UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

IMPROVING VULNERABILITY DETECTION OF WAP

Miguel Amorim Falé

MESTRADO EM INFORMÁTICA

Dissertação orientada por:
Prof. Doutor Nuno Fuentecilla Maia Ferreira Neves
e co-orientada pela Prof. Doutora Ibéria Vitória de Sousa Medeiros

2017

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisors, Prof. Nuno Fuentecilla Maia Ferreira Neves and Prof. Ibéria Vitória de Sousa Medeiros, for their patience towards some of my difficulties over the course of this dissertation. Their guidance and feedback were crucial, and helped me regain focus on priorities.

I would like to thank Henrique Mendes, Miguel Rodrigues, Inês Gouveia, Paulo Antunes and Ricardo Costa for shedding some positive light and support during challenging and stressful times. Even though we are currently distant, I would also like to thank my family for their advice in taking on risks and uncertainties.

This work was partially supported by the EC through project FP7-607109 (SEGRID) and H2020-643964 (SUPERCLOUD), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference to UID/CEC/00408/2013 (LaSIGE).

Resumo

As aplicações web apresentam um nível de sofisticação que foi gradualmente melhorado no decorrer de duas décadas. Em particular, estas aplicações evoluíram de simples aglomerados de documentos hipermédia, para sistemas altamente complexos e extensíveis, desempenhando um papel fundamental no acesso a uma miríade de serviços. No entanto, esta evolução manifesta-se também pela procura e desenvolvimento de aplicações avançadas em prazos restritos, um fenómeno que é desencadeado pela competitividade agressiva e dinamismo próprios da Web moderna. Esta mudança de paradigma leva muitos programadores a seguir atalhos no desenvolvimento de aplicações, tal como o uso de linguagens de programação populares (ex. PHP), a integração de bibliotecas e extensões de origem dúbia, e a negligência para com os bons padrões de desenvolvimento de software. Infelizmente, tais práticas encontram-se frequentemente associadas à ocorrência de vulnerabilidades no código fonte destas aplicações, que comprometem a segurança das mesmas.

A gravidade deste problema induz assim uma urgência crescente no que toca ao desenvolvimento de aplicações seguras. Contudo, o sucesso das medidas de segurança depende tanto dos conhecimentos de quem as elabora, bem como da correta utilização das linguagens de programação. É por isso que uma longa história de trabalho em segurança de aplicações web tem vindo a acompanhar tal evolução.

As ferramentas de análise estática de código são utilizadas para detetarem vulnerabilidades nos programas de forma automática. Por excelência, estas ferramentas são as mais utilizadas por conseguirem uma maior cobertura do código analisado, poderem ser utilizadas durante o ciclo de desenvolvimento de aplicações e por não necessitarem de executar o código da aplicação. Porém, a qualidade da análise realizada por estas ferramentas na deteção de vulnerabilidades assenta na correta codificação do conhecimento sobre as vulnerabilidades a detetar e implementação das técnicas de análise estática de código, tal como a análise de comprometimento. Isso significa que, por um lado, estas ferramentas apenas procuram vulnerabilidades no código fonte para as quais foram codificadas, sendo incapazes de encontrar os restantes tipos de vulnerabilidades. Por outro lado, podem gerar falsos positivos (falsas vulnerabilidades) e falsos negativos (vulnerabilidades não detetadas) devido à não completude das técnicas de análise estática nelas

implantadas. O principal objetivo desta dissertação assenta no melhoramento das capacidades de deteção de vulnerabilidades presentes na ferramenta *Web Application Protection (WAP)* para análise de código PHP. A concretização deste objetivo pressupõe três passos.

O primeiro passo consiste, numa primeira instância, no estudo das vulnerabilidades de validação de input de aplicações web e das formas como são detetadas por análise estática de código. Seguidamente, no levantamento de falhas de implementação num conjunto de ferramentas de análise estática de código fonte em PHP e em particular na WAP. É importante perceber que falhas estão presentes nas ferramentas atuais e de que forma se correlacionam com os falsos positivos e negativos gerados nas análises. Para efetuar este levantamento, é necessário executar cada ferramenta em estudo e comparar os resultados de ferramentas diferentes, utilizando como alvo de análise os mesmos artefactos de software em PHP. É de igual importância analisar o código manualmente, pelo que constitui a única forma de detetar falsos negativos e confirmar falsos positivos simultaneamente presentes em todas as ferramentas. As ferramentas são aplicadas no processamento de código fonte PHP encontrado em pacotes de software da plataforma WordPress.

O segundo passo consiste no estudo de um dos problemas identificados. O problema selecionado para o efeito assenta na identificação e resolução de *dependências circulares* no código fonte, as quais podem prejudicar a qualidade da análise, provocando até a paragem inesperada da ferramenta. As dependências circulares são oriundas da inclusão recursiva de código contido em ficheiros (ex. ficheiro ***a*** inclui ficheiro ***b*** e ficheiro ***b*** inclui ficheiro ***a***). Inicialmente, é feito um levantamento das capacidades de inclusão disponíveis na linguagem PHP com base na documentação oficial. Seguidamente, são definidos e avaliados casos de estudo que utilizam estas capacidades de modo a produzir dependências circulares. Observam-se os respetivos comportamentos das ferramentas no processamento destes casos. Simultaneamente, é feita uma ponte com o comportamento demonstrado pelo PHP Zend, pois fornece um contexto prático que serve para completar as ambiguidades identificadas na documentação oficial.

O terceiro passo consiste na resolução do problema das dependências circulares na ferramenta WAP. Apresentamos propostas de resolução do problema das dependências circulares, passando pela identificação de ficheiros raiz, caminhos de inclusão e causadores de ciclo dado um conjunto aleatório de ficheiros PHP. Este passo culmina no desenvolvimento de dois algoritmos que detetam e resolvem ciclos num projeto PHP, respetivamente. Oferecemos uma avaliação experimental das melhorias implementadas na ferramenta WAP, com base em pacotes de software da plataforma WordPress. Por um lado, a avaliação pretende verificar a capacidade de deteção de ciclos e vulnerabilidades por parte da versão melhorada da WAP. Por outro, permite a confirmação dos comportamentos identificados no segundo passo, bem como a identificação de novos problemas, relacionados com falhas de implementação das ferramentas Pixy, RIPS e phpSAFE.

A nova versão da ferramenta WAP (*WAP++*) permite identificar e resolver 16 de-

pendências circulares e detetar 891 vulnerabilidades (6 vulnerabilidades adicionais). Para além disso, a ferramenta WAP++ não apresenta nem os comportamentos erróneos da versão original, nem os comportamentos erróneos observados nas outras ferramentas de análise estática. Em contraste, a versão original da ferramenta WAP não identifica quaisquer dependências circulares.

Palavras-chave: análise estática de código fonte, vulnerabilidades de validação de entrada, dependências circulares, segurança de aplicações web, segurança de software

Abstract

For over two decades, the web has been evolving from a simple set of hypermedia documents to a complex ecosystem of web applications that are supported by various frameworks. This paradigm shift has been promoting a series of practices that lead to an increasing number of vulnerabilities, which can compromise the security of web applications. One of the main contributing factors lies in vulnerable source code, written in unsafe languages such as PHP.

In order to mitigate the problem, a large research effort on web application security has occurred over the past years. Source code static analysis tools perform the task of finding program vulnerabilities in an automated fashion. These tools offer superior code coverage, easier integration into the application development cycle, and do not require the actual code to be executed. They instead perform source code analysis, looking for potential bugs while inspecting the program code. However, the analysis performed by these tools depends on their knowledge of the classes of vulnerabilities and the implementation of analysis techniques, such as taint analysis. This means that, on one hand, the tools only search for vulnerabilities in the source code that they hold knowledge of, being unable to find other kinds of problems. On the other hand, the tools may generate false positives and false negatives, due to the limitations and incompleteness of implemented analysis techniques.

One of such tools is the *Web Application Protection (WAP)*. The main objective of this dissertation is to identify problems with WAP and improve its vulnerability detection capabilities, when processing open source PHP code. Four static analysis tools - WAP, Pixy, phpSAFE and RIPS - are evaluated against a set of WordPress plugins that are known to be vulnerable, in order to collect examples of incorrect processing of the tools which lead, for instance, to false negatives. Additionally, we define and evaluate several use cases for a common found limitation, which consists in the identification and circumvention of *circular dependencies* (i.e., recursive inclusion of code) in the source code. If circular dependencies are not treated correctly, they may lead to unexpected tool behaviors and incorrect analyses. These assessments help reflecting upon new solutions to address WAP's shortcomings. A new version of WAP is implemented, and evaluated with the same original WordPress plugins.

This dissertation offers the following contributions. A list of vulnerabilities is compiled through manual analysis of the plugins, in a format that allows comparison between the chosen tools, and the identification of common false negatives. An enhanced version of WAP is implemented, with improved detection capabilities that reduce both false positives and false negatives. Two evaluations concerning WAP and a set of WordPress plugins are present, comparing the results before and after the enhancements, respectively.

Keywords: source code static analysis, input validation vulnerabilities, circular dependencies, web application security, software security

Contents

List of Figures

List of Tables

List of Acronyms

AST Abstract Syntax Tree. 19, 21, 22, 45–47, 49

CA Code Analyzer. 3, 5, 21

CFG Control Flow Graph. 22

CMS Content Management System. 1, 2, 8

DBMS Database Management System. 10, 11

DOM Document Object Model. 18

EP Entry point. 8–10, 13, 93, 94

EPC Execution path constraint. 8, 11, 13, 94, 95

EPVC Entry point validation constraint. 8, 13, 94, 95

IVV Input validation vulnerability. 8

MIFT Main Include File Table. 42, 45, 46, 48–50, 55

MST Main Symbol Table. 42, 45, 46, 48–50, 52, 53, 55

OOP Object-oriented programming. 24, 59

phpSAFE PHP Security Analysis For Everyone. iv, vi, ix, xv, 4–6, 22–24, 27–29, 33, 34, 36, 39, 40, 42, 44, 59–61, 63, 65

RIPS Re-Inforce PHP Security. iv, vi, ix, xv, 4–6, 22–25, 27, 28, 33, 34, 36, 39, 40, 42, 44, 59–62, 65

SCSA Source code static analysis. 1, 3, 6, 7, 19, 36, 65

SF Sanitization function. 8, 10

SQLI SQL injection. xiii, xv, 7, 9–12, 23, 33, 59, 60

SS Sensitive sink. 8–10, 93, 94

ST Symbol Table. 21, 42, 45–47, 49, 51–55, 57, 63

TEPT Tainted Execution Path Tree. 21, 46

TI Tainted input. 8

TST Tainted Symbol Table. 21, 95

UD Untainted Data. 21

VV Vulnerability. 7, 9, 57, 59

WAP Web Application Protection. iv–vii, x, xiii, 2–6, 20–25, 27, 28, 32–34, 36, 37, 39, 40, 42, 44–46, 57, 59, 60, 63, 65, 66, 94, 95

XSS Cross-site scripting. xv, 7, 11, 12, 15, 23, 39, 58–60

Chapter 1

Introduction

Nowadays, web applications play a fundamental role in the access to a wide range of services, presenting a level of sophistication that was enhanced in the course of two decades. These applications evolved from simple hypermedia documents (in the Web 1.0 era), to complex and interconnected systems, partially due to the integration of plugins and widgets (in the Web 2.0 era) [42]. Ever since this transition, the demand for richer web applications has been increasing, which, combined with the frenetic competition and dynamism of the new Web, puts pressure on programmers to create them in constrained time intervals. Under these conditions, two common practices consist in using easy to learn languages (such as PHP) and plugins, to cut development time and enhance customization. For example, WordPress, the most popular Content Management System (CMS) [32], offers a panoply of these extensions and plugins of diverse subjects (e.g., e-commerce, maps, contacts, calendars and testimonials).

An unfortunate outcome of this paradigm is that web applications suffer from many security issues [3]. The process of securing web applications depends on the knowledge of the developers, as well as on the correct use of the underlying programming language. Developers often do not know how to build secure code, and therefore they deploy their software with flaws. This fact is also visible in third-party software included in web applications, such as plugins developed for WordPress [41][38]. Despite the tremendous amount of research concerning web application security over the last two decades [6][30][61][67], reaching such goal remains a non-trivial task.

Fortunately, nowadays, source code static analysis (SCSA) tools [35] perform the task of finding these vulnerabilities in an automated fashion. Compared to manual code inspection, this process has the advantage of saving time and cost, allowing developers to remain focused on writing the basic program logic. The disadvantage is that these tools tend to generate false positives and false negatives, depending on the approach and techniques applied by each tool.[1] In addition, the vulnerabilities are required to be manually

[1] False positives consist in signaled vulnerabilities that do not actually exist, while false negatives consist in undetected vulnerabilities.

removed, something that might be unfeasible for developers that do not have appropriate knowledge about secure coding.

WAP is a tool that discovers and removes input validation vulnerabilities in the source code of PHP applications [37]. It uses knowledge about entry points (user inputs) and sensitive sinks (functions that perform some kind of sensitive operation). The tool resorts to taint analysis (a form of static analysis) to find candidate vulnerabilities, and data mining to predict the existence of false positives. Finally, the candidate vulnerabilities considered as being real vulnerabilities are automatically fixed by the tool, by modifying the source code. The main objective of this dissertation is to improve WAP's vulnerability detection capabilities by identifying and correcting a few of its limitations, focusing on plugins from the WordPress CMS.

1.1 Motivation

The problem of web application security is related to the foundation and evolution of the World Wide Web. In the 1990s, most websites consisted in a set of static text pages with the inclusion of other media (e.g., images). The restrictions inherent to this platform reduced the opportunities for elaborate web-based attacks, as well as their impact.

The advent of Web 2.0 spawned more opportunities and tools that promoted two important properties. For one, *complexity*, allowed for the creation of richer applications. Such complexity was achieved through the use of server-side scripts that produce dynamic, user-specific content. Server-side script languages (e.g., PHP, Ruby) rely heavily on databases, files and operating file systems, in order to accomplish advanced tasks. This property inevitably led to an increasing number of bugs contained within these applications. The other property, *extensibility*, supports the combination of components, commonly found today as untrustworthy third-party extensions and plugins that can be installed in browsers and/or back end systems [29]. One practical instance of extensibility is observable in the use of plugins that are integrated within CMSs [42].

WordPress is currently the most used CMS [32][71] and it is programmed in PHP, which is the most used server-side script language for development of web applications [32][70]. WordPress supports the integration of plugins that offer very diverse features, saving programming time when it is necessary to include these features. PHP is an appealing script language due to its accessible documentation [53], support for other technologies (e.g., many database management systems, compression and multimedia formats [47]), availability on major operating systems [49] and apparent ease of learning.

The downside is that WordPress and other CMSs *do not apply security checks on plugins*, aside from issuing a few warning messages [75]. PHP might be an easy language to learn, but it *contains many sensitive functions* (i.e., functions that are susceptible to be exploited by some malcrafted data) that when exploited can cause unexpected and unde-

sirable results. Examples of such functions are the *shell_exec* and *system* functions, which execute commands on the underlying operating system, and the *eval* function, which allows the execution of arbitrary PHP code. Another factor that makes the code easy to exploit is that the language is weakly typed, thus allowing for the injection of unexpected data [7]. It is common for programmers to deploy these plugins without having knowledge about secure coding. In a production environment, data is received from users with arbitrary intentions. Additionally, vulnerabilities found in one plugin may compromise several applications using that plugin. As a result, when these extensions are targeted by attacks, the results range from sensitive data being leaked to the actual web server being completely compromised.

In order to tackle these issues, web security mechanisms were developed, some relying on runtime checks [61] while others focusing on the discovery of vulnerabilities through the source code analysis [35]. SCSA tools, which fall in the latter category, are more easily integrated into the application development life cycle, providing better code coverage and managing to find the actual source of security problems [41]. However, they may also report many false positives. This problem is due to undecidability, meaning that the static analysis tools are unable to determine how the target software behaves for all inputs [36]. On the other hand, false negatives can occur, for instance, when the tools do not have the proper knowledge about framework-specific entry points and sensitive sinks.

WAP performs static analysis of PHP programs. In addition to searching for input validation vulnerabilities, the tool also reports some of the vulnerability's properties (e.g., the set of affected instructions and files) and applies fixes in the code, correcting and removing the identified bugs. The correction of source code effectively saves a lot of time and effort on the developer's side. This means that, while the process of vulnerability removal can be automated, the programmer has the opportunity to learn how vulnerabilities occur, and how they can be corrected.

Recent experience with the WAP tool has identified some problems, as it appears to have some flaws that prevent the analysis of certain programs or the detection of specific vulnerabilities. As later explained in the document, these problems are also common in other static analysis tools, thus, it is important to keep researching this topic.

1.2 Objectives

First, we aim at finding and documenting problems with the existing SCSA tools in terms of detection of vulnerabilities in PHP open source applications, with deeper emphasis on WAP. Within its logic, WAP contains a Code Analyzer (CA) module, which is where most problems apparently stem from, so it is important to grasp some familiarity with its source code. A complementary task consists on evaluating the tools against a set of WordPress plugins that are known to be vulnerable, and collecting various examples of false positives

and negatives. Since false negatives seem to be a prevalent problem for WAP, we need to understand the reason why they occur by studying how the code is processed, and by identifying common patterns on vulnerable source code that are not being detected. In addition to WAP, we chose *Pixy*, *PHP Security Analysis For Everyone (phpSAFE)* and *Re-Inforce PHP Security (RIPS) 0.55* for this study.

Second, we aim at improving the WAP tool with new approaches and techniques. In particular, we focus on one of the identified problems, related to the existence of *circular dependencies* (i.e., recursive inclusion of code) in the source code. If not treated correctly, circular dependencies may affect the static analysis process, causing false positives and negatives or even the crash of the tool. We define and evaluate several use cases in order to understand the precise effects of circular dependencies.

Third, we solve the problem of circular dependencies in WAP, through the development of algorithms that identify the root points in a set of PHP files, discover all valid inclusion paths, and circumvent the found circular dependency points.

This new version of WAP was also evaluated to assess its improvements.

1.3 Contributions

The dissertation presents the following contributions:

- An experimental identification of problems in static analysis tools. This contribution consists in the compilation of a vulnerability list (i.e., from a set of WordPress plugins known to be vulnerable) which aids in identifying key patterns and characteristics concerning the vulnerabilities. These patterns will explain the occurrence of errors (false positives and false negatives) in WAP and the other tools. Additionally, new solutions may arise by understanding the nature of these patterns, and they will be proposed in order to fix the found issues.

- A study concerning a common limitation found in several static analysis tools. One particular problem is triggered by the existence of circular dependencies within the source code (i.e., file ***a*** includes file ***b***, and file ***b*** includes file ***a***). Static analysis tools analyze the given source code, thus they are sensitive to circular dependencies caused by inclusion of external files. If such dependencies are not processed correctly, they may lead to invalid vulnerability detections, in the form of false positives and false negatives. Initially, this study presents several use cases in which circular dependencies occur, as well as the expected outcome according to PHP's documentation. Afterwards, the study documents the behavior of four static analysis tools, as well as the PHP Zend engine, when processing these use cases.

- A solution concerning the identification of inclusion paths to be considered during the analysis, in the form of an algorithm. These paths may or may not encounter cir-

cular dependencies. If circular dependencies are indeed present in the source code, they imply the existence of points which are responsible for creating them. The algorithm is capable of identifying and solving these circular dependency points without breaking the analysis.

- Implementation of the studied solutions in the WAP tool and experimental evaluation, which consists in taking the improved version of WAP and comparing it with its previous version, as well as with the other tools.

1.4 Publications

The work performed on this dissertation allowed us to publish a communication paper (8 pages) for the *Symposium on Informatics - INForum 2017*, in the *Security of Computational Systems and of Communication Networks* track [9].

1.5 Organization

This document is organized as follows:

- Chapter 2 describes the context and related work. It begins with the detailed explanation of two vulnerability classes, and complements such explanation with examples. More importantly, this chapter presents the static analysis process, and which techniques are most commonly employed by static analysis tools when implementing this process. WAP's CA module is explained with greater detail, since it is used in the following chapters. The chapter ends with an assessment of key static analysis features, and whether or not they are available throughout the four chosen static analysis tools.

- Chapter 3 presents several problems that were found in static analysis tools. First, it describes the manual analysis process against a set of WordPress plugins with aim to produce a set of vulnerabilities with detailed characteristics. This assessment is crucial in order to identify the strengths and weaknesses of the WAP's current implementation, and to reflect upon new solutions to mitigate these weaknesses. It also presents some additional experiments that helped to identify other bugs, and to find a connection between them and their origins in the actual WAP source code. Pixy, phpSAFE and RIPS 0.55 are studied in a similar fashion.

- Chapter 4 includes an in-depth behavioral study on a specific problem found in Chapter 3, which consists in the incorrect processing of circular dependencies during the analysis of PHP source code. Several use cases are illustrated, according to the expected behavior found in PHP's documentation. These use cases are then

processed by the four chosen static analysis tools, and the PHP Zend engine, in order to understand how their behavior may differ from the expected outcome.

- Chapter 5 describes known approaches to identify and solve circular dependencies, and their respective limitations. We also present our own contribution on the matter, in the form of an algorithm that detects cycles in a PHP project (i.e., the circular dependency points that generate these cycles), and another algorithm that solves the identified cycles. These algorithms were implemented in the WAP tool.

- Chapter 6 contains an experimental evaluation of the features implemented in the WAP tool. The tool is evaluated against several plugins from the WordPress platform, in order to detect the existence of cycles and vulnerabilities. This process is complemented with the evaluation of four other SCSA tools - Pixy, RIPS, phpSAFE and WAP 2.1 - in order to compare results and document any behavioral errors found. The chapter ends with an overall assessment of the results.

- Chapter 7 presents the conclusion to this dissertation. The work developed throughout this dissertation is summarized and reflected upon. Some possibilities for future work are also discussed.

Chapter 2

Context and Related Work

This chapter presents several concepts and definitions that are crucial to the understanding of the vulnerability detection process. The chapter also references representative works on the matter, particularly, those that present a vulnerability detection approach and implement it in the form of a SCSA tool.

The chapter begins with a general description of the vulnerability classes contained within the scope of the dissertation, as well as an enumeration of properties that are common among them. Following this high-level explanation, the chapter elaborates upon the specific mechanics of each class (i.e., one section per class), complete with a series of examples taken from the manual analysis. Subsequently, the chapter explains the process of SCSA, as well as known approaches and implementations. The concepts of *semantic analysis*, *data flow analysis* and *taint analysis*, within the context of static analysis, are explained. The chapter ends with an assessment of key features in static analysis, explaining if they are present in each tool.

2.1 Vulnerability classes

This section presents two main vulnerability classes, SQL injection (SQLI) and Cross-site scripting (XSS). They are placed within the top three positions of the OWASP Top 10 in 2013 [74], a report which depicts the ten most critical web application security risks.

As explained in Section 1.1, most web applications are programmed in a server-side script language that allows for the creation of dynamic web pages, depending on external input. Most of these languages also support the management of user-generated content through the use of files and databases. If the source code written in these languages does not consider security aspects when using these features, the applications can become vulnerable to malicious actions. Attackers may be able to read and modify sensitive data, build fake web pages (usually meant to steal user credentials) and perform privilege escalation. These flaws are what we call *vulnerabilities (VVs)*, where the SQLI and XSS classes are the ones most often left in web applications.

A common pattern among these two vulnerability classes is that their attacks consist in sending malicious user inputs originating from a client (such as a web browser) to a target web server, in which the inputs are unchecked and used in some sensitive manner. This means that they fit on the meta-class of *input validation vulnerabilities (IVVs)*[1] [37][34], and as such this dissertation will focus on it.

Before we explain the classes in higher detail, it is important to clarify the properties of an input validation vulnerability:

- *Entry points (EPs)* consist in any variables and functions from which external user inputs enter the application, such as *$_GET* and *$_POST*. Usually, EPs do not perform validation measures against their inputs, unless they are special functions tailored for certain PHP frameworks, such as the *get_post* function from the WordPress CMS, which supports sanitization filters [82].

- *Tainted Inputs (TIs)* consist in maliciously crafted EP data that can be used in sensitive operations. Note that these inputs may propagate taintedness to other variables and functions of the program, therefore creating *entry point dependencies*.

  Since most web applications are ubiquitous, their outside environment must not be considered trustworthy. As such, *entry point data is considered tainted by default* (more details in Section 2.2).

- *Sensitive sinks (SSs)* are functions in the program that perform a sensitive operation, such as the output of dynamic information to the web browser, or a database operation. When some TI reaches a SS, it can cause security problems, such as data leaks [45] and the injection of malicious scripts [2].

- *Sanitization functions (SFs)* are meant to sanitize tainted data, thus making it untainted, i.e., nullify the harmful effects of TIs.

- *Entry point validation constraints (EPVCs)* are conditions that validate data from an EP and/or their dependencies. Depending how effective these constraints are, they may share similar purposes with SFs. The main difference is that constraints prevent the flow of execution from entering their respective blocks (i.e., in case the input does not meet the requirements).

- *Execution path constraints (EPCs)* are conditions that split the main flow of execution into several independent paths.

  The latter two notions are crucial, since a SS may be placed before or within a constraint. A vulnerability may be triggered under many different circumstances, through different paths. The effectiveness of SFs and constraints depends on the input's type and on the vulnerability class being considered.

[1] Alternatively known as *taint-style vulnerabilities*

Examples of these constructs will be presented throughout this section (see Tables 2.1 and 2.2). Code slices extracted from the manual analysis will also be described, in order to illustrate the specific mechanics of each vulnerability class.

2.1.1 SQL injection

SQLI attacks target parameters that are inserted into *structured query language* (SQL) commands (usually called *queries*) with little to no validation, which are then sent to the underlying database. An attacker may be able to manipulate expressions and predicates (see Figure 2.1) using special characters, therefore modifying the original syntax of the query.

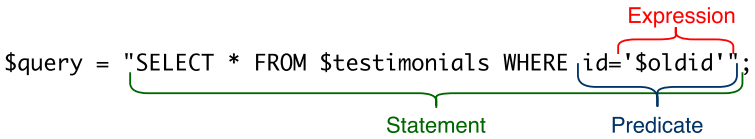

Figure 2.1: Basic elements of an SQL statement.

Figure 2.2 presents a SQLI vulnerability, using the same statement from Figure 2.1. In this case, the targeted parameter is *$oldid*, which corresponds, through attribution, to the entry point *$_GET['oldid']*. *$_GET* is an associative array of values which enter the application through the URL parameters of an HTTP request [48]. Thus, it is one of the many EPs available in PHP, in addition to the ones listed in the *Entry point* column of Table 2.1.

```
                    0' UNION SELECT * FROM users --
$oldid = $_GET['oldid'];
$query = "SELECT * FROM $testimonials
      WHERE id='0' UNION SELECT * FROM users -- '";
$results = mysql_query($query);
```

Figure 2.2: SQLI vulnerability example. The payload, colored in red, targets potentially sensitive user records.

Furthermore, *mysql_query* is responsible for executing commands against the underlying database [51], which may contain sensitive records. Due to the arbitrary nature of its operation, and the fact that it does not sanitize its arguments, *mysql_query* is considered a SS. The *Sensitive sink* column of Table 2.1 contains examples of other SSs.

Neither the entry point nor its dependency (*$oldid*) are sanitized before reaching contact with the sensitive sink. For this reason, we call this flaw a VV, i.e., an entry point or its dependency that reaches a sensitive sink without any sanitization and/or validation, which

may be exploited through payloads such as the ones exemplified in the *Tainted input* column of Table 2.1. Note the use of special symbols, such as the prime character (') and the comment indicator (- -). From a malicious perspective, both can be used to terminate the original query abruptly. The prime character allows for the injection of new operators and predicates, while the comment indicator makes sure that any other statement parts succeeding it are disregarded.

The execution of the modified query may access or modify unauthorized data. In addition to business-related records, certain injections may cause data leaks concerning the database schema (e.g., tables, columns, types), which may help the attacker understand how the database is organized, and perform more sophisticated attacks. An attacker may also be interested in discovering the type and version of the database, a process known as *database fingerprinting* [43].

The syntax of SQLI attacks depends on the Database Management System (DBMS) and the server-side script language being used. WordPress uses MySQL as its underlying DBMS [76], which will be the focus of this document.

Entry point	Tainted input	Sensitive sink	Sanitization function	Validation constraint
$_GET	0 OR 1=1 - -	mysql_query	mysql_real_escape_string	is_null
$_POST	passwd' OR pass <> 'passwd	mysqli_query	mysqli_real_escape_string	is_numeric
$_REQUEST	' UNION SELECT secretcode FROM... - -	$wpdb->query	$wpdb->prepare	!= ''
$_SERVER			$wpdb->esc_like	
$_COOKIE				
$_FILES				

Table 2.1: Examples of SQLI vulnerability properties

2.1.1.1 SQLI vulnerability example

Figure 2.3 depicts the (simplified) analysis of a SQLI vulnerability, taken from the manual analysis compilation (Slice #9 of *collision-testimonials.3.0* plugin, please check the appendix).

The analysis shows that the EP *$_GET['oldid']*, listed in the first table of the figure, is neither verified by a constraint nor sanitized by a SF. It creates two dependencies: *$oldid* and *$query*, which are listed in the second table of the figure. The latter reaches a SS, *mysql_query*.

Sensitive sink

- Name: mysql_query
- Line: 177
- File: collision- testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	File
175	GET	$_GET['oldid']	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	File
175	variable	$oldid	$_GET['oldid']	1	1	collision-testimonials.3.0/pages/testimonials.php
176	variable	$query	$oldid	1	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	T/F	File
170	($_GET['action'] == "edit") && (!isset($_POST['editQuote']))	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET[′action′] == "edit") && (!isset($_POST[′editQuote′]))){	collision-testimonials.3.0/pages/testimonials.php
175	$oldid = $_GET[′oldid′];	collision-testimonials.3.0/pages/testimonials.php
176	$query = "SELECT * FROM $testimonials WHERE id=′$oldid′";	collision-testimonials.3.0/pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php

Figure 2.3: Sample analysis of a SQLI vulnerability.

The last table of the figure presents the slice (i.e., excerpt of the source code) containing the vulnerability. Although it contains some constraints (line 170) that perform some validation, they are not associated to the entry point; so the entry point remains tainted. These EPCs are listed in the third table.

If the entry point data is malicious, it can manipulate the query's structure so that it retrieves more data than it should, when executed by the DBMS. However, by applying a sanitization function, such as a cast to the integer type, or *mysql_real_escape_string* (for MySQL DBMS), the effect of a malicious input is eliminated. The *Sanitization function* column of Table 2.1 contains examples of other sanitization functions. Most drivers nowadays support prepared statements as well, though these require more complex modifications to the code.

2.1.2 Cross-site scripting

XSS attacks consist in the injection of malicious client-side script code (usually JavaScript) into the unprotected output of a web application. The code is embedded in the resulting

HTML document and executed in the victim's browser. Unlike SQLI, this type of attack attempts to harm users, rather than the web application.

There are three classes of XSS attacks depending on how the payload interacts with the application, which affects the way it is sent to the victim [44]. They are Reflected XSS, Stored XSS and DOM-based XSS. The DOM-based XSS class specifically targets the client-side, without interacting with the back end system. As such, it will only be given a slight explanation, since the scope of the chosen tools concerns the server-side of web applications.

Entry point	Tainted input	Sensitive sink	Sanitization function	Validation constraint
$_GET	<script>alert('XSS')</script>	echo	htmlentities	is_null
$_REQUEST	<p onmouseover=alert (document.cookie)>HERE!</p>	die	esc_js	preg_match
$_POST	<script>window.open("http://evil.pt/ steal.php?cookie="+document.cookie) </script>	exit	wp_json_encode	!= "
file_get_contents* mysql_query*		file_put_contents		

Table 2.2: XSS example vulnerabilities. Stored XSS specifics are marked with *.

2.1.2.1 Reflected XSS

A reflected XSS vulnerability occurs when the web application contains a sensitive function that outputs (i.e., reflects) unsanitized information to the browser. An attacker can craft an URL that assigns malicious code to the relevant, unsanitized parameters (e.g., variables that will be concatenated within error or greeting messages). An unsuspecting user needs to be convinced into clicking the link, in order to trigger the effect (e.g., e-mail messaging of the link). The link then accesses the application, and its code is reflected back to the browser, through functions such as *echo* or *die*. The attacker may craft malicious code for various purposes:

- By subverting a web page's appearance with a fake login form, the attacker can trick the user into revealing its credentials.

- By replacing download links, the user can be tricked into downloading an harmful program, such as a trojan horse. Similarly, the user can be redirected to a malicious web-site.

- User cookies with session information can be stolen (i.e., sent to the attacker's site). Authentication cookies are included in requests, and they are used by web servers in order to identify which user is making the request. They aid in the implementation of the site's business logic (e.g., separation between "normal users" and "premium users", e-commerce features such as shopping carts, user preferences). An attacker

that obtains a victim's cookie can effectively impersonate the victim, and access the application with the same user privileges.

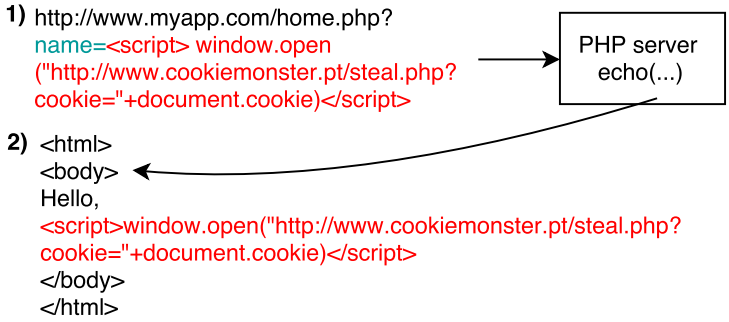

Figure 2.4: Example of a Reflected XSS vulnerability that allows a cookie theft.

Figure 2.4 illustrates a Reflected XSS vulnerability. The entry point is a variable in the *home.php* script that holds URL parameters. The sensitive sink, *echo*, prints arbitrary contents into the resulting document. The payloads, exemplified in the *Tainted input* column of Table 2.2, contain HTML elements that hold malicious scripts. Note that, even though the use of the <*script*> tag is common, it is not obligatory to use such tag. In fact, an attacker can take advantage of HTML events and inject code in the event attributes (e.g., onclick, onmouseover) of other elements, such as <*p*> or even <*body*>.

Reflected XSS example

Figure 2.5 depicts the (simplified) analysis of a Reflected XSS vulnerability, taken from the manual analysis compilation (Slice #5 of *ajaxgallery.3.0* plugin, in the appendix).

The analysis shows that the EP *$_GET['gId']*, listed in the first table of the figure, is validated by a constraint. Constraints related to entry points or dependencies are what we call of EPVCs, which are listed in the third table. However, this particular instance only checks for the existence of the value (i.e., the value is not unset or NULL). From an information security standpoint, it is a weak constraint that allows typical XSS payloads into the application. The EP generates a dependency called *$ggId* (in the second table), which reaches the sensitive sink *echo*.

The last table of the figure presents the slice containing the vulnerability. Note that the EPVC is also an EPC, since its *if* block contains the sink (i.e., it begins in line 159 and ends somewhere after line 164).

The attacker can include and execute arbitrary JavaScript code into the page with varying objectives, as mentioned in Section 2.1.2.1. By applying encoding measures to the output, such as *htmlentities* and *wp_json_encode*, these effects can be nullified. Another approach consists in sanitizing the input, such as imposing a white-list of accepted HTML elements, thus discarding any other type of element. Both approaches may be combined.

Sensitive sink

- Name: echo
- Line: 164
- File: ajaxgallery.3.0/utils/options.php

Entry points:

L	Type	Name	File
160	GET	$_GET['gId']	ajaxgallery.3.0/utils/options.php
160	POST	$_POST['gId']	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	File
160	variable	$ggId	$_GET['gId']	isset($_GET['gId'])	1	ajaxgallery.3.0/utils/options.php
160	variable	$ggId	$_POST['gId']	isset($_GET['gId'])	0	ajaxgallery.3.0/utils/options.php

Entry Point Validation Constraints

L	Condition	Nested by	T/F	File
159	isset($_GET['gId']) || isset($_POST['gId'])		1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	T/F	File
159	isset($_GET['gId']) || isset($_POST['gId'])	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
159	if(isset($_GET[′gId′]) || isset($_POST[′gId′])){	ajaxgallery.3.0/utils/options.php
160	$ggId = isset($_GET[′gId′]) ? $_GET[′gId′] : $_POST[′gId′];	ajaxgallery.3.0/utils/options.php
164	<textarea>[ajax_gallery id=<?php echo $ggId;?>]</textarea> 	ajaxgallery.3.0/utils/options.php

Figure 2.5: Sample analysis of a Reflected XSS vulnerability.

2.1.2.2 Stored XSS

Stored XSS attacks store the payload in a persistent storage, such as a file system or a database, on the target web server. As such, they require two main steps:

- The first step consists in taking advantage of operations that modify information. Web applications usually perform write operations to a database and/or file system, in order to store business data (e.g., the SQL INSERT or UPDATE commands, PHP extensions which manipulate files). If these operations are not complemented with sanitization measures, a malicious input may enter the application and be stored inside legitimate structures (e.g, database tables, folders, configuration files). This is called the *write* step.

- The second step consists in taking advantage of operations that output information. Web applications perform read operations through the use of SQL queries, or PHP extensions that retrieve file contents. If such operations are not complemented with encoding and/or sanitization measures, they open the opportunity for stored malicious code to be executed, thus accomplishing the attack. This is called the *read* step.

Any user browsing the application may be a potential victim in this scenario, since there is no need to click a specially-crafted link. The payload may be retrieved by a static query (i.e., a query without user-provided arguments) or file access (i.e., the file is loaded without depending on user input). As a result, stored XSS vulnerabilities are much more dangerous than other known types of XSS vulnerabilities.

The browser may request content from a subset (i.e., table column, file line) of the persistence layer at any time. If this subset is housing the harmful script, the browser will execute the code, causing the issues mentioned in Section 2.1.2.1.

Stored XSS example

Figures 2.6 and 2.7 depict the (simplified) analysis of a Stored XSS vulnerability, taken from the manual analysis compilation (Slice #7 of *ajaxgallery.3.0* plugin, please check the appendix). Once again, observe that this vulnerability class requires two steps: a *write* step that stores malicious scripts in a persistent environment (without sanitizing it), and a *read* step that outputs the stored malicious script (without encoding it).

The *UPDATE* command in Figure 2.6 is vulnerable due to the use of a tainted input, *$value*, which is assigned to the *option_value* field in rows where *option_name* has the value *'agItem'*. The SELECT command in Figure 2.7 retrieves all fields where *option_name* is 'agItem' (from the same table), including option_value. This value's taintedness would then be propagated to the *$options* array (since the contents were stored in colon-separated fashion). Finally, *option*'s elements are argument to echo, the sink of this vulnerability.

These vulnerabilities can be eliminated with the same measures mentioned in Section 2.1.2.1. However, accurate detection of these vulnerabilities can be difficult, as certain fields (e.g., numeric *AUTO_INCREMENT* fields) are not affected by Stored XSS. Additionally, complex sanitization measures may take place before persisting the data (e.g., *preg_replace* of certain metacharacters), leading to false positives.

Sensitive sink

- Name: echo
- Line: 60,61,62,63
- File: ajaxgallery.3.0/utils/list.php

WRITES (1st step)
Entry points:

L	Type	Name	File
42	POST	$_POST['user']	ajaxgallery.3.0/utils/options.php
43	POST	$_POST['album']	ajaxgallery.3.0/utils/options.php
44	POST	$_POST['rows']	ajaxgallery.3.0/utils/options.php
45	POST	$_POST['cols']	ajaxgallery.3.0/utils/options.php
46	POST	$_POST['align']	ajaxgallery.3.0/utils/options.php
47	POST	$_POST['pag']	ajaxgallery.3.0/utils/options.php
48	POST	$_POST['showp']	ajaxgallery.3.0/utils/options.php
49	POST	$_POST['thumbsize']	ajaxgallery.3.0/utils/options.php
51	POST	$_POST['gId']	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	File
42	variable	$value	$_POST['user']	isset($_POST['user']) && trim($_POST['user']) != ""	1	ajaxgallery.3.0/utils/options.php
43	variable	$value	filterAlbum ($_POST['album'])	isset($_POST['album']) && trim($_POST['album']) != ""	1	ajaxgallery.3.0/utils/options.php
44	variable	$value	$_POST['rows']	isset($_POST['rows']) && trim($_POST['rows']) != ""	1	ajaxgallery.3.0/utils/options.php
45	variable	$value	$_POST['cols']	isset($_POST['cols']) && trim($_POST['cols']) != ""	1	ajaxgallery.3.0/utils/options.php
46	variable	$value	$_POST['align']	isset($_POST['align']) && trim($_POST['align']) != ""	1	ajaxgallery.3.0/utils/options.php
47	variable	$value	$_POST['pag']	isset($_POST['pag']) && trim($_POST['pag']) != ""	1	ajaxgallery.3.0/utils/options.php
48	variable	$value	$_POST['showp']	isset($_POST['showp']) && trim($_POST['showp']) != ""	1	ajaxgallery.3.0/utils/options.php
49	variable	$value	$_POST['thumbsize']	isset($_POST['thumbsize']) && trim($_POST['thumbsize']) != ""	1	ajaxgallery.3.0/utils/options.php
51	variable	$sql	$value, $_POST['gId']	1	1	ajaxgallery.3.0/utils/options.php

Entry Point Validation Constraints

L	Condition	Nested by	T/F	File
50	isset($_POST['gId'])	39	1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	T/F	File
39	isset($_POST['save'])	1	ajaxgallery.3.0/utils/options.php
50	isset($_POST['gId'])	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
32	function filterAlbum($str){	ajaxgallery.3.0/utils/options.php
33	$str = str_replace(" ", "", $str);	ajaxgallery.3.0/utils/options.php
34	$str = str_replace(".", "", $str);	ajaxgallery.3.0/utils/options.php
35	$str = str_replace(",", "", $str);	ajaxgallery.3.0/utils/options.php
36	return $str;	ajaxgallery.3.0/utils/options.php
37	}	ajaxgallery.3.0/utils/options.php
39	if(isset($_POST['save'])){	ajaxgallery.3.0/utils/options.php
42	$value = (isset($_POST['user']) && trim($_POST['user']) != ""? $_POST['user'] : "user") . ":";	ajaxgallery.3.0/utils/options.php
43	$value .= (isset($_POST['album']) && trim($_POST['album']) != "" ? filterAlbum($_POST['album']) : "") . ":";	ajaxgallery.3.0/utils/options.php
44	$value .= (isset($_POST['rows']) && trim($_POST['rows']) != "" ? $_POST['rows'] : "4") . ":";	ajaxgallery.3.0/utils/options.php
45	$value .= (isset($_POST['cols']) && trim($_POST['cols']) != "" ? $_POST['cols'] : "4") . ":";	ajaxgallery.3.0/utils/options.php
46	$value .= (isset($_POST['align']) && trim($_POST['align']) != "" ? $_POST['align'] : "center") . ":";	ajaxgallery.3.0/utils/options.php
47	$value .= (isset($_POST['pag']) && trim($_POST['pag']) != "" ? $_POST['pag'] : "0") . ":";	ajaxgallery.3.0/utils/options.php
48	$value .= (isset($_POST['showp']) && trim($_POST['showp']) != "" ? $_POST['showp'] : "true") . ":";	ajaxgallery.3.0/utils/options.php
49	$value .= (isset($_POST['thumbsize']) && trim($_POST['thumbsize']) != "" ? $_POST['thumbsize'] : "160");	ajaxgallery.3.0/utils/options.php
50	if(isset($_POST['gId'])){	ajaxgallery.3.0/utils/options.php
51	$sql = "UPDATE $wpdb->options SET option_value='$value' WHERE option_name='agItem' and option_id=".$_POST['gId'];	ajaxgallery.3.0/utils/options.php
52	}else{	ajaxgallery.3.0/utils/options.php
55	$wpdb->query($sql)	ajaxgallery.3.0/utils/options.php

Figure 2.6: First step of the sample analysis of a Stored XSS vulnerability.

READS (2nd step)
Entry points:

L	Type	Name	File
54	BD	$wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	ajaxgallery.3.0/utils/list.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	File
54	variable	$gItems	$wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	0	1	ajaxgallery.3.0/ utils/list.php
55	variable	$gItem	$gItems	0	1	ajaxgallery.3.0/ utils/list.php
56	variable	$gItem->option_value	$gItem	0	1	ajaxgallery.3.0/ utils/list.php
56	variable	$gItem->option_value	$gItem->option_value	1	1	ajaxgallery.3.0/ utils/list.php
56	variable	$options	$gItem->option_value	1	1	ajaxgallery.3.0/ utils/list.php

Entry Point Validation Constraints

L	Condition	Nested by	T/F	File
55	foreach ($gItems as $gItem)		1	ajaxgallery.3.0/utils/list.php

Execution path constraints

L	Condition	T/F	File
55	foreach ($gItems as $gItem)	1	ajaxgallery.3.0/utils/list.php

Slice:

L	Instruction	File
54	$gItems = $wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	ajaxgallery.3.0/utils/list.php
55	foreach ($gItems as $gItem){	ajaxgallery.3.0/utils/list.php
56	$options = split(":", $gItem->option_value);	ajaxgallery.3.0/utils/list.php
60	<td><?php echo $options[0]; ?></td>	ajaxgallery.3.0/utils/list.php
61	<td><?php echo $options[1]; ?></td>	ajaxgallery.3.0/utils/list.php
62	<td><?php echo $options[2]; ?></td>	ajaxgallery.3.0/utils/list.php
63	<td><?php echo $options[3]; ?></td>	ajaxgallery.3.0/utils/list.php

Figure 2.7: Second step of the sample analysis of a Stored XSS vulnerability.

2.1.2.3 DOM-based XSS

The previous two classes require the web application to perform the dynamic creation of a response page containing the malicious input. DOM-based XSS payloads are not used in the actual server; instead, they take advantage of script vulnerabilities in the response page, at the client-side.

The Document Object Model (DOM) represents an HTML page, containing its elements and attributes. The attacker targets DOM attribute references that are susceptible of being used by JavaScript for certain client-side operations (e.g., document.URL, document.location). The effects take place when the browser receives the response page, and fills DOM attributes with malicious data provided in the URL. Figure 2.8 provides a brief example of a vulnerable web page, plus a possible exploit.

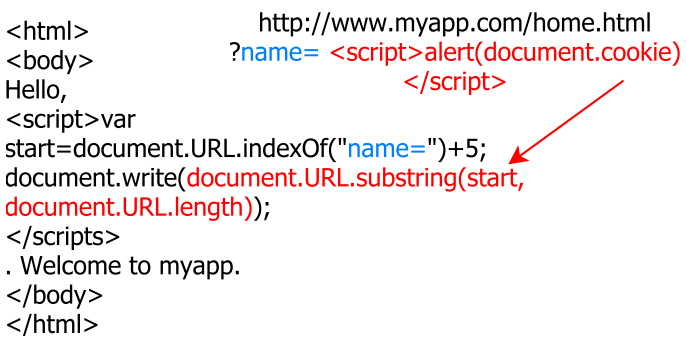

Figure 2.8: Example of a DOM-based XSS vulnerability and possible exploit.

2.2 Source code static analysis

The research corpus related to static analysis is very broad, reaching back to at least two decades [73][35][62][7]. Generally speaking, static analysis is a technique that may process source code, binary code, or intermediate code. This technique may be used for a variety of ends, such as vulnerability detection, complexity measuring, and redundancy detection. However, this section will only focus on static analysis for the purpose of detecting web application vulnerabilities in the source code. This section also presents some of the main SCSA tools for the PHP language.

SCSA extracts meaning from the source code without running it. Among the various techniques within SCSA, *semantic analysis* plays a fundamental role in the design of major static analysis tools. The semantic analysis process operates in two main steps. Initially, the *lexical analysis* step, performed by a *lexer*, breaks the code into atomic parts and returns the sequence of atoms. These atoms are named *tokens* (e.g., logical and mathematical operators, keywords, strings), and the sequence of atoms provided by the lexer is called a *token stream*. Given the token stream and a set of grammatical rules, an *abstract syntax tree (AST)* is constructed through the operation of a *semantic analyzer*, commonly referred to as *parser*. A good parser should be able to discern PHP language constructs from one another. For instance, it should distinguish a variable called *$gets* from a call to function *gets*, or a use of the word *"echo"* in a string from the use of the *echo* construct. *Data flow analysis* traces the input data flows taken inside the program to detect vulnerabilities. *Taint Analysis* is the most often employed type of data flow analysis. This technique considers all external *input* originating from an *entry point* as *tainted*. The analysis follows the use of these external values within the program, considering the various possible paths and dependencies, and detects if these values reach a sensitive function. If this occurs then a vulnerability exists in the source code, which could be exploited by some malicious input. During the analysis, each tracked variable may have its state changed from *tainted* to *not tainted*, depending on the code locations visited by the inputs [37]. The items listed in Section 2.1 are inspired by concepts belonging to taint analysis.

2.3 Source code static analysis tools

SCSA tools are used to detect errors in programs, including vulnerabilities. As an example, Figure 2.9 presents the main components of a static analysis tool for PHP programs. The source code (in our case, written in PHP) is taken, and its components (e.g., variables, instructions) are organized in a model, which contains data structures representing the program. The model is then analyzed for vulnerability detection, with the help of knowledge about entry points and sensitive sinks (e.g., a text document with a list of entry points or a sensitive sink database). Such analysis employs a set of rules and tracking mecha-

nisms in order to identify whether or not a vulnerability occurs, as mention in Section 2.2. The analysis may use additional techniques, such as context-sensitive, inter-procedural data flow analysis, in order to obtain higher precision. Finally, the vulnerabilities that were found are listed in the results.

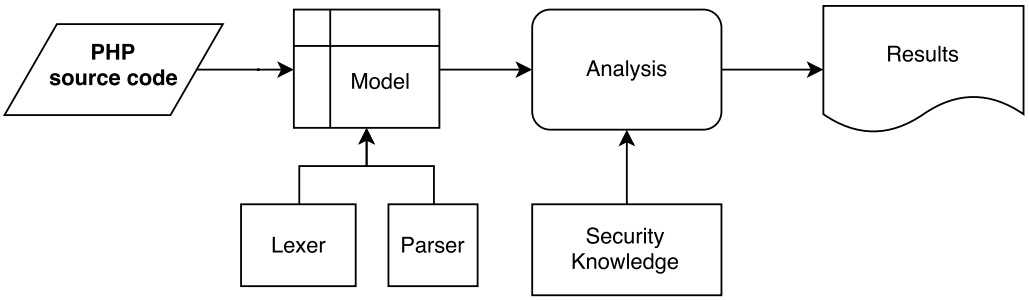

Figure 2.9: Main components of a static analysis tool for PHP programs.

2.3.1 WAP

Section 1.1 introduced a simple view of WAP and its functionalities. The approach followed by WAP is based on the security of language-based *information flows* [31] [63] [39]. The tool detects problematic information flows in the source code, in the form of vulnerability detection from entry points to sensitive sinks. The tool also blocks these flows by removing the identified vulnerabilities via source code modification. This notion covers two important characteristics in information security [64]. The first, *confidentiality*, states that private information should not be disclosed while flowing through public objects. The second, *integrity*, states that untrusted data should not flow through trusted objects without being carefully scrutinized. Web application attacks exemplified throughout Section 2.1 attempt to circumvent these two properties, thus violating information-flow security. Figure 2.10 illustrates some of these violations.

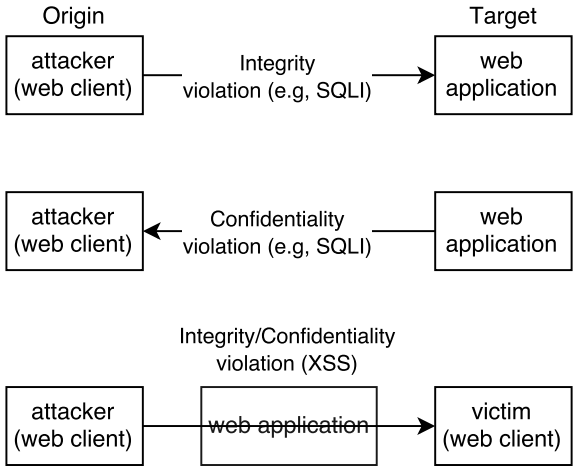

Figure 2.10: Problematic information flows and their targets.

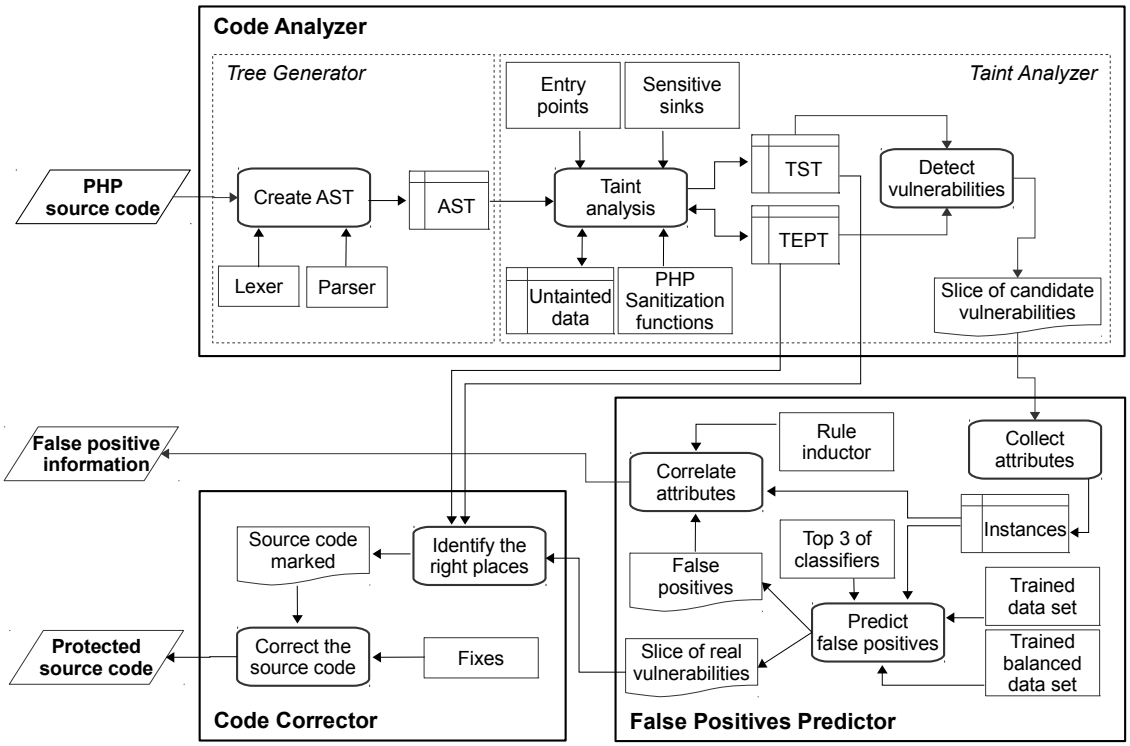

Figure 2.11: WAP's architecture, containing its main modules and supporting structures (taken from [37]).

WAP's architecture, shown in Figure 2.11, is built on top of the basic architecture from Figure 2.9. Inside the CA module, the *Tree Generator* produces ASTs, which are high-level tree models representing the code, created by the combined efforts of the lexer with the parser. Then, the ASTs are navigated by *tree walkers*, and enriched with several annotations (e.g., instructions, instruction locations, taintedness values, aliases). These annotations, as well as the associated subtrees of the AST, are stored in a *Symbol Table (ST)*. In the *Taint Analyzer*, the tool performs taint analysis to detect input validation vulnerabilities. Each ST node is analyzed, verifying if its data contains entry points or dependencies. In such case, the data is marked as being tainted and its taintedness is propagated to nodes related with the nodes containing tainted data. The results of taint analysis are stored in a Tainted Symbol Table (TST), which is the ST with its nodes marked as tainted. While building these tables, the tree walkers also create *Tainted Execution Path Trees (TEPTs)*. Each branch contains a tainted variable, and a sub-branch for each line in the code where the variable becomes tainted. Each sub-branch contains a list of dependencies that the tainted variable propagated its taintedness into, in the aforementioned line. Variables are moved into an *Untainted Data (UD)* structure (not shown in the figure) whenever their value changes from tainted to untainted. Each variable is marked as tainted back in the TST iff it belongs to TEPT but not UD. The symbiotic work between TST, TEPT and the manually crafted security knowledge aids the tool in understanding the propagation of

taintedness between variables. The end result is a set of candidate vulnerable control-flow paths (referred to as *slices* in the figure), from an entry point to a sensitive sink [37]. The *False Positives Predictor* and *Code Corrector* modules contain features that further complement the idea of information flow security. However, these modules remain outside of the scope of this dissertation.

2.3.2 Other static analysis tools

Pixy [34], phpSAFE [41] and RIPS 0.55 [6] are tools that perform static analysis. Like WAP, they track entry points, checking if some of them reach some sensitive sink and taking in consideration data sanitization. In certain aspects, they are similar to WAP, but differ in others as Section 2.3.3 explains.

Pixy. Pixy is the first open source static analysis tool for PHP among the chosen tools. Given a PHP file, the tool uses a modified version of *JFlex* for lexical analysis [10], and a modified version of *CUP* for semantic analysis [4]. The result is a parse tree of the input file, which is then translated to a Control Flow Graph (CFG). The taint analysis, complemented with alias analysis, operates on these CFGs. Each node on the CFG contains an instruction and the previous value associated with that instruction's receiving variable (i.e., in case of an update) [35][34].

PhpSAFE. phpSAFE performs lexical analysis using a PHP extension called the *Tokenizer* [58]. The Tokenizer provides the *token_get_all* function. Given the contents of a target script, it breaks its source code into tokens, and returns these tokens. Each token in the result contains its identifier, value and line number [57]. An AST of the source code is created from these tokens. Similarly to WAP, phpSAFE extracts information about user-defined functions, parameters, function calls, among others, from the AST. Taint analysis is performed through management of the data flow history of each variable. This history is stored into the *parser_variables* array. Each entry contains the variable name, source file name, line number, dependencies, applied functions, among others items. During taint analysis, phpSAFE parses every generated AST, and makes decisions based on the found source code constructs (e.g., assignments, expressions, function calls and returns, etc.). The *parser_variables* array is updated accordingly [41].

RIPS. RIPS also performs lexical analysis using the Tokenizer, and builds an abstract syntax tree of the source code. RIPS also extracts information about user-defined functions, parameters, function calls, among others, from the AST. Each AST is transformed into a CFG by splitting conditional program flow into linked blocks. Generally speaking, whenever a conditional statement, function call or file inclusion is detected, a new block

is created and linked with the previous block. Data flows are simulated by the relationships between blocks. Whenever a function call is detected, the function name is checked against a list of sensitive sinks. If the function is indeed a sensitive sink, the relevant call arguments are traced backwards through these blocks. If any of these arguments happen to be some user input or dependency, then there is a potential vulnerability, unless some sanitization measure has been applied somewhere along the trace [6].

2.3.3 Comparison between static analysis tools

All tools support the following properties:

- **Inter- and Intra-procedural analysis.** Inter-procedural analysis implies that data flows need to enter functions, whenever they are called, regardless of whether the function exists in the calling file. Such analysis removes the need for erroneous presumptions about the called function's behavior, enhancing precision and thus reducing the number of false positives and false negatives. Intra-procedural analysis implies that data flows need to enter an individual function, without considering the context from which it is called [41][37][5].

- **Global analysis.** This analysis considers the propagation of variables defined with global scope.

- **Context sensitivity.** Context-sensitive tools take into account the context in which a function call or file include was made [37].

The remainder of this section presents differences between taint analysis tools that will be used in the context of this dissertation, for the evaluation of WAP. Table 2.3 provides an overview of these differences.

Supported vulnerability classes. Taint analysis tools search for errors depending on the supported vulnerability classes. Pixy and phpSAFE only support two vulnerability classes, SQLI and XSS [35][41]. In addition to these two classes, RIPS and WAP deal with others, such as *Remote file inclusion (RFI)*. A crucial difference between tools is that WAP's architecture offers a modular way to add new vulnerability classes [38].

Alias analysis. In addition to entry point dependencies, PHP also supports the creation of *aliases* [60]. Aliases are variables that, at a certain point in the program, store their value in the same memory location that of other variables. Among the tools, Pixy and WAP identify these aliases, allowing for a more precise analysis that reduces false positives and false negatives [37][35][6]. Pixy contains a slight limitation, in that its alias analysis does not cover the use of aliases for arrays or array elements.

Object-oriented support. Unlike phpSAFE and WAP [41][37], Pixy and RIPS do not support Object-oriented programming (OOP) features [6][34]. Pixy and RIPS evaluate object-oriented constructs optimistically, i.e., they consider that such constructs cannot be tainted.

Project analysis. Among the chosen tools, only WAP and RIPS perform full project, sub-directory analysis [37][6].

PHP frameworks. Among the chosen tools, only WAP and phpSAFE recognize WordPress specific functions [41][38]. WAP supports extension for other frameworks without the need to modify its main code.

Automatic code correction. Among the chosen tools, only WAP performs automatic code correction, modifying the source code with *fixes* that remove vulnerabilities found. WAP's *Code Corrector* module uses vulnerability information provided by the Code Analyzer module (e.g., vulnerability class and the code slice) in order to decide which fixes should be applied, and the location in the source code where fixes should be inserted. Code fixes mainly perform validation or sanitization measures against user input, nullifying the harmful effects of dangerous metacharacters or metadata before reaching a sensitive sink. As a result, these fixes are either inserted directly in the line of the sensitive sink, or before it. The former is usually a better choice. Applying fixes as further away from the entry point as possible avoids interference with other code that also performs some sort of sanitization or validation [37].

Tools	Classes	Alias analysis	OO support	Project analysis	PHP Frameworks	Code correction
Pixy	SQLI, Reflected XSS	Yes	No	No	None	No
phpSAFE	SQLI, Reflected XSS, Stored XSS [1]	No	Yes	No	WordPress	No
RIPS	SQLI, Reflected XSS, Stored XSS [2], others[6]	No	No	Yes	None	No
WAP	SQLI[3] Reflected XSS Stored XSS others[38]	Yes	Yes	Yes	WordPress	Yes

Table 2.3: Summary of differences between tools

Notes regarding Table 2.3:

- 1. phpSAFE follows a pessimistic philosophy in regards to stored injections, thus treating data from any persistence source as tainted. This may result in false positives.

- 2. RIPS will signal potential Stored XSS vulnerabilities if option *2. file/DB tainted + 1* is enabled.
- 3. WAP assumes the background database to be MySQL, DB2, or PostgreSQL.

Chapter 3

Identification of Problems in Source Code Static Analysis Tools

This chapter presents some problems and limitations that were found in static analysis tools, with focus on WAP. It begins by listing the set-up process for WAP's development environment and the software stack that is required in order to run WAP's production environment. It also lists the software stack required in order to run the Pixy, phpSAFE and RIPS tools. They are deployed in a separate operating system from that of WAP, in an attempt avoid cluttering, overhead and extra dependencies (e.g., RIPS and phpSAFE require a web server to run).

Subsequently, the chapter explains the reporting format employed for the manual analysis, based on important patterns in the source code. It presents two tables summarizing the results. Table 3.1 is organized by vulnerability class, while Table 3.2 is organized by both tool and vulnerability class. The chapter ends with a summary of the problems identified in each tool during the manual analysis process.

3.1 Development and production environments

3.1.1 WAP

The development environment for WAP includes:

- A folder containing the grammar files.
- A collection of scripts that create and transfer parsers to the WAP tool.
- A script that transfers packages back to the parser, so that any new methods developed on WAP's side can be recognized.
- The source code for the tool, and its dependencies. It may integrated with an IDE, such as Netbeans 8.1.

The software stack required to run WAP is as follows:

- The Operating system, Xubuntu 16.04.1 LTS.
- The Java Development Kit, version 1.8.0_111

Progress concerning the development of WAP is being saved through a version control system, *git* [11], in a BitBucket repository [1].

3.1.2 RIPS, phpSAFE and Pixy

The software stack is as follows:

- The Operating system, Xubuntu 14.04.3 LTS.
- The Java Development Kit, version 1.7.0_121
- XAMPP for Linux 5.6.12-0, which includes:
  - Apache web server, version 2.4.12
  - PHP, version 5.6.11
  - MySQL, version to 5.6.25
  - phpMyAdmin, version 4.4.12

3.2 Manual analysis

In order to reach our first goal, we turned to manual analysis of WordPress plugins. It served as a base for identifying important patterns in the source code, and understanding the influence of control structures and functions on the information flow. We analyzed six plugins using WAP 2.1 [65], RIPS 0.55 [26], Pixy [13] and phpSAFE [12], which gave us information about sensitive sink locations and the evaluations performed by the tools (i.e, vulnerable, not vulnerable). Table 3.1 indicates the names of the plugins that were considered. The data for WAP helped us find problems with its behavior when dealing with certain situations, and reflect upon new solutions we could implement, to make it more effective.

Iteratively, we built a format containing the following items[1]:

- **Sensitive sink** contains the name of the sink (i.e. *mysql_query*) and its location.
- **Entry points** is a table that contains the names, types and locations of entry points. The manual analysis process has shown that most developers simply use native PHP entry points, such as *$_GET*.

[1] Each item is explained more thoroughly in the appendix.

- **Entry point dependencies** is a table containing variables to which taintedness was propagated to, under certain conditions.
- **Execution path constraints** contains the control flow context for the slice, indicating whether it contains the sensitive sink, whether the constraint is contained (”nested”) inside another constraint, and the ”true/false” value that should be met under the desired control flow context.
- **Entry point validation constraints** contains validations performed against the entry points or dependencies.
- **Slice** contains the relevant trace of code, encompassing the entry points, dependencies, constraints, validations, and, finally, the sink.

We then proceeded to compile information about the WordPress plugins using this format. One report per plugin was created, each containing the list of vulnerabilities, their characteristics, and a final verdict for each (i.e., vulnerable or not vulnerable). At the end of each report, there are usually 3 sections:

- **VV definition**: General conditions in which a vulnerability occurs.
- **WAP false negatives**: A list indicating discovered false negatives.
- **Misc.:** Comments regarding unusual situations, WordPress sanitization functions, native PHP sanitization functions, concept clarifications, etc.

Note that some slices are omitted from the summary tables due to lack of relevance (i.e., re-running a slightly updated version of the tool resulted in the correct detection of the vulnerability) or because they are duplicates. The duplicates occur because phpSAFE considers that a sink containing n tainted inputs simultaneously will produce n vulnerabilities. The following slices were omitted:

- ajaxgallery.3.0 slice #4.
- calculated-fields-form.1.0.10 slice #4.
- collision-testimonials.3.0 slice #27, #32.
- community-events.1.2.9 slices #6, #8, #11, #16 and #18.

Table 3.1 presents a summary of the results of the manual analysis. Note that the high number of false positives for Stored XSS is due to the retrieval of a harmless integer typed column set with *AUTO_INCREMENT* (usually an identifier representing a unique entity). These fields were wrongly detected as tainted. Table 3.2 presents a summary of the results per tool.

Plugin	**Slices**	**SQLI**		**Reflected XSS**		**Stored XSS**	
		TP	**FP**	**TP**	**FP**	**TP**	**FP**
ajaxgallery.3.0	19	3	0	2	0	10	4
calculated-fields-form.1.0.10	11	2	1	0	0	0	8
collision-testimonials.3.0	30	10	1	3	0	11	5
community-events.1.2.9	34	3	0	20	1	4	6
contact-form.2.7.5	26	5	0	6	0	3	12
easy2map.1.2.4	17	0	0	7	4	2	4
TOTAL	137	23	2	38	5	30	39

Table 3.1: Summary of the manual analysis

Plugin	Slices	phpSAFE										WAP										RIPS										Pixy									
		SQLI			Reflected XSS			Stored XSS			SQLI			Reflected XSS			Stored XSS			SQLI			Reflected XSS			Stored XSS			SQLI			Reflected XSS			Stored XSS		
		TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
ajaxgallery.3.0	19	2	0	1	2	0	0	0	4	10	2	0	1	2	0	0	0	0	10	0	0	3	2	0	0	0	0	10	0	0	3	2	0	0	0	0	6	0	4
calculated -fields-form .1.0.10	11	2	1	0	0	0	0	0	8	0	2	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	0
collision -testimonials .3.0	30	8	1	2	3	0	0	0	0	11	7	1	3	3	0	0	0	0	11	7	0	3	3	0	0	11	5	0	6	0	4	3	0	0	0	0	0	0	11
community -events .1.2.9	34	3	0	0	4	0	16	4	6	0	0	0	3	2	1	18	0	0	4	0	0	3	7	0	13	0	0	4	0	0	3	2	0	18	0	0	0	0	4
contact-form .2.7.5	26	4	0	1	6	0	0	3	12	0	4	0	1	6	0	0	1	0	2	0	0	5	6	0	0	0	1	3	0	0	5	1	0	5	0	0	0	0	3
easy2map .1.2.4	17	0	0	0	7	3	0	1	4	1	0	0	0	2	1	5	0	0	2	0	0	0	3	0	4	0	0	2	0	0	0	0	0	7	0	1	0	1	2
TOTAL		19	2	4	22	3	16	8	34	22	15	2	8	15	2	23	1	0	29	7	0	16	21	0	17	11	6	19	6	0	17	8	0	30	6	2	24

Table 3.2: Summary of the manual analysis per tool

3.3 Limitations in WAP

While performing the analysis, we discovered that certain PHP projects are not processed as expected. Due to file inclusion problems, WAP may end abruptly by either not identifying any files, or through a Java error. Most limitations lie in the code analyzer's taint analysis phase, in which many vulnerabilities are left undetected due to specific code patterns that WAP does not process correctly. Finally, the code corrector is unable to fix certain situations as expected, and does not accurately report which entry point(s) are being addressed when processing certain patterns. Overall, the main classes of problems that were observed are described below.

Incorrect processing of circular dependencies. Circular dependencies involving the use of code inclusion directives (i.e., ***a*** includes ***b***, and ***b*** includes ***a***) will result in invalid vulnerability detections, in the form of false positives and false negatives (e.g, abrupt termination of analysis, *StackOverflowError* errors due to infinite recursion [8]). As such, circular dependencies need to be identified and resolved. It is essentially a problem of locating and preventing loops in directed graphs. Chapter 4 provides an in-depth study with more details on this matter.

Incorrect concatenation operation. WAP may lose track of taintedness propagation, especially during the processing of complex sensitive sinks. These are usually composed by a set of concatenation operations. Issues with string concatenations are probably related to the operation of the WAP's parser.

Lack of conditional awareness Existing approaches for information flow security [37] define a track path as an execution path that contains a sensitive sink - a construct that can be exploited for malicious purposes - and one or several entry points - sources containing potentially unsafe user input. However, an important factor is missing from this definition, as a single execution path can be split into several paths depending on the control flow statements along the way. This poses a problem for the internal structures of WAP, as the symbol tables do not keep track of the control flow context of variables in the PHP source code. As a result, false negatives occur when the sensitive sink is preceded by multiple execution paths that share a common tainted variable, and that variable is used as input for the sensitive sink. In such cases, only the last branch is considered by the Taint Analysis.

Code corrector issues Some issues concerning the Code Corrector module were identified, but most should subside by the correction of the Taint Analysis process. Regardless, they are documented here for completeness:

- When the sensitive sink occupies more than one line (through concatenation), and the tainted input is not on the first line, the code corrector will not fix the tainted input.

- When multiple strings containing tainted inputs are joined, only one of the strings is fixed. Note that, for SQLI, the fix is applied on the last line where the query is composed, a process that may disregard previous concatenations.

- Entry points used in loop constructs (e.g, *foreach*) are not reported; instead, the code corrector will report that the entry point exists in a different line, if used in another location of the code.

3.4 Limitations in other tools

The remaining three tools were also evaluated in order to detect problems in their processing. Some of the found limitations are also present in WAP, and they are as follows:

- **Incorrect processing of circular dependencies.** This issue is also present in Pixy, phpSAFE and RIPS. Pixy and RIPS treat the *include* and *include_once* directives identically, which may cause false negatives. phpSAFE crashes with infinite recursion, when processing circular uses of the *include* directive. One important factor to consider is that Pixy and phpSAFE do not support full project analysis (i.e., only one input file is considered per analysis). Such characteristic makes the problem less effectual for these tools. Once again, Chapter 4 provides more details on this matter.

- **Incorrect concatenation operation.** This issue is also present in Pixy and RIPS.

- **Lack of conditional awareness.** This issue is also present in Pixy, phpSAFE and RIPS. Pixy will only consider the first branch of the execution paths, while the other tools only consider the last.

- **Incorrect processing of non-literal includes.** Unlike the other static analysis tools, phpSAFE does not resolve non-literal *include* targets, thus ignoring them. This limitation hinders the tool's inter-procedural capabilities, causing false negatives.

- **Lack of taintedness propagation between files.** phpSAFE does not propagate taintedness into included files. This limitation hinders the tool's inter-procedural capabilities, causing false negatives.

3.5 Summary of Limitations

The problems are summarized in Table 3.3. The character *x* marks the presence of a problem in a tool. *N/A* signifies that the tool does not support the feature which suffers from the identified problem.

Problem	WAP	Pixy	phpSAFE	RIPS
Issues with circular dependencies	x	x	x	x
Issues with concatenation	x	x		x
Lack of conditional awareness	x	x	x	x
Code correction issues	x	N/A	N/A	N/A
Issues with non-literal includes			x	
Issues with inter-procedural taintedness propagation			x	

Table 3.3: Summary of problems and limitations in static analysis tools

We can see that none of the four tools handle circular dependencies correctly, for a variety of reasons explained in Section 3.4. As a result, it is a relevant limitation that needs to be addressed. In addition, the four tools do not consider all branching paths in a control structure, resulting in an incomplete analysis. The remaining problems are less critical overall. Among the four tools, phpSAFE shows the most robustness when dealing with concatenation operations. However it cannot resolve non-literal includes, and does not propagate taintedness between included files and their parents. Only WAP contains code correction capabilities.

Chapter 4

Circular Dependencies in Static Source Code Analysis

This chapter studies a particular limitation concerning the incorrect use and/or analysis of code inclusion directives in PHP. Most programming languages support the concept of *file inclusion*, which consists in an indication that the contents of a target file should be incorporated in the current flow of execution. This is a very useful practice for code reuse in several parts of an application. Each include file may, for instance, store the the initialization of common global variables, and the declaration of user-defined functions with similar responsibilities. These constructs may then be called by multiple application modules, at any time.

4.1 Include cycles in PHP

The PHP language implements the includes with two types of directives: *require/include* and *require_once/include_once*. The first type, in the form of *require* or *include*, copies the contents (e.g., functions, code) of the target file and inserts these contents into the current flow of execution, when the directive is called. The included file is made aware of the previous program context (i.e., before the call to *require/include*). In other words, the included file is able to access variables and functions created by its ancestors. All functions and classes in the included file have a *global scope*, meaning it is possible to call these functions and classes from a different file, as long as both files belong to the same flow [55][27]. The second directive, in the form of *require_once* or *include_once*, tracks which files have already been included, and only includes each file once [56]. Duplicate checking is performed, which considers multiple denominations of the same file. If, for instance, the target of an *include_once* directive contains a relative path, it will be resolved to an absolute path and compared against the set of already included files [28]. Static analysis tools only analyze the PHP source code without running the actual interpreter, so they must be prepared to simulate these content inclusion behaviors.

However, certain applications contain *circular dependencies*, in which the included

file is simultaneously its own child and parent (e.g., ***a*** includes ***b***, and ***b*** includes ***a***). A more elaborate instance of circular dependency occurs when the included file already belongs to the execution chain, making it its own successor and ancestor (e.g., ***a*** includes ***b***, ***b*** includes ***c***, and ***c*** includes ***a***). Static analysis tools are sensitive to this phenomenon, since they process include directives as well, for the sake of completeness and precision. The non-contemplation of potential circular dependencies during static analysis may cause undesirable results, such as incorrect vulnerability detection, abrupt termination of analysis, or excessive memory consumption due to infinite recursion. This characteristic further aggravates the tendency for tools to produce false positives and false negatives due, for instance, to the language's undecidability, or to the lack of inter- or intra-procedural analysis. We intend to study the behavior of the four SCSA tools for the PHP language, in order to understand if potential occurrences of circular dependencies are contemplated. We also intend to understand if these behaviors are faithful to the descriptions in the official PHP documentation.

Our tests and discussions lead to the identification of four important cases, based on cycles caused by these directives and included files containing code or user-defined functions. If there is a cycle, it means that there is a file responsible for causing it, when being included. We denominate this file a *cycle maker*. The cycle maker can contain code or user-defined functions. If it only contains user-defined functions, there is no need to revisit it. Since function definitions attain global scope, they are only collected once [55][27]. However, if the cycle maker contains some code in addition to the user-defined functions, we treat it identically to a cycle maker that only contains code. New vulnerabilities may arise in the code by revisiting the cycle maker.

Cycle caused by a file containing code.

- ***include_once/require_once.*** It is necessary to include the target file only once, as suggested in [56]. An important characteristic of a PHP project is that it may have several *roots* (i.e., files that do not have ancestors). Unlike Pixy and phpSAFE, which only receive one input file for analysis, WAP and RIPS are capable of full project, sub-directory analysis. Discarding duplicate file inclusions in linear fashion is an incorrect approach for the latter case. The management of included files should be made relatively to each possible root in the project. This problem is discussed in-depth at the beginning of Chapter 5.

- ***include/require.*** This is by far the most ambiguous case, since the PHP documentation does not specifically explain the behavior in case of an infinite cycle [55]. In addition to the aforementioned root resolution problem, static analysis tools should be equipped with an additional mechanism to reduce the halting problem.

Cycle caused by a file containing user-defined functions.

- ***include_once/require_once.*** The desirable outcome in static analysis would be to collect all of the file's functions without causing an infinite cycle. Fortunately, this case has a rather trivial solution, compatible with PHP's expected behavior. We previously mentioned that all functions and classes of the included file can be called from a different file within the same flow, during execution. On the other hand, static analysis tools, such as WAP, have a tendency to construct the program model before performing taint analysis. This means the program's functions are collected and modeled preemptively, through semantic analysis. As such, there is no need to revisit a cycle maker that only contains functions, and any duplicate include directives can be safely ignored (i.e., treated as redundant) [55].

- ***include/require.*** Documentation for the PHP language refers that it is not possible to redefine a previously declared function. Given that this type of directive can include the same file multiple times, it is unclear whether duplicate declarations should be ignored, or a redeclaration error should be shown [59].

Given the documentation's ambiguity, especially in the *include/require* cases, it is important to understand what happens in practice. For such purpose, we defined a series of test cases divided into two batches. The main objective is to understand the behavior of the chosen static analysis tools, as well as the PHP Zend engine, when processing these test cases. In particular, we are looking to answer the following questions:

1. How does each tool resolve cycles? Does it treat *include* and *include_once* identically, or is the halting problem reduced in some other way? Is it faithful to the behavior found in the documentation for PHP, and/or in the PHP Zend engine?

2. Which is the order of instruction blocks? Are repeated inclusions fully ignored, or is the flow directed in some other way?

3. How do static analysis tools behave when finding duplicate function declarations? Are they faithful to the behavior found in the PHP Zend engine?

4.2 Cycle caused by a file containing code

The first batch of tests is based on the format illustrated in Figure 4.1. This format implies the existence of three files containing code (***a***, ***b*** and ***c***) and an include relationship between them, represented by solid arrows. The dotted arrows indicate that the execution reaches the end of file, thus returning to the previous parent file. The tests themselves are illustrated in Table 4.1, and consist in a series of configurations with activated/deactivated code parts, beginning in file ***a***. The twelve tests are valid for both *require/include* and

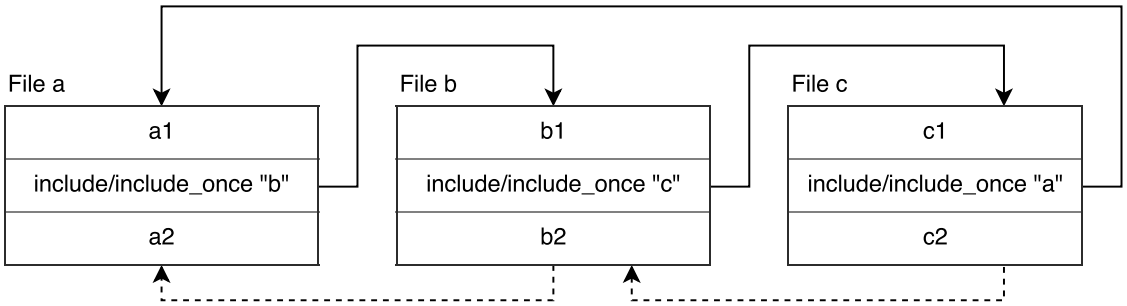

Figure 4.1: Baseline use case where cycle maker contains code.

require_once/include_once directives. For example, test #3 begins in file ***a***, by executing the block ***a1***. After ***a1***, the execution proceeds by including file ***b*** (***inc "b"***) and executing its first instruction, ***b1***. During this file's execution, after ***b1***, file ***c*** is included (***inc "c"***). In this example, only ***c1*** is executed (i.e., there is no circular dependency). This means the execution returns to file ***b***, for the execution of block ***b2***. Note that the instructions ***inc "a"***, ***c2***, and ***a2*** are deactivated in this example (i.e., grayed out). They do not take part in this execution chain. More importantly, examples #5 through #12 imply the existence of a cycle between ***c*** and ***a***. Overall, the tests are expected to propagate taintedness in this general order: ***a1***, ***b1***, ***c1***, ***c2***, ***b2***, ***a2***. Furthermore, we make three reasonable assumptions:

- For the examples #5 through #12, a cycle is eventually formed during the inclusion of file ***a*** by file ***c***. We do not know, however, when such cycle will occur. According to the documentation, the desirable outcome should be ***a1***, ***include_once b***, ***b1***, ***include_once c***, ***c1***, ***c2***, ***b2***, ***a2*** for *include_once*. The second inclusion of ***a*** is ignored, since it already exists in the execution chain (i.e., not as an include file, but as the first file). For *include*, it should be ***a1***, ***include b***, ***b1***, ***include c***, ***c1***, ***infinite cycle*** (the second inclusion of ***a*** will likely lead to the infinite cycle, as this directive does not check for duplicate files in the chain) [55].

- Taintedness is propagated between available parts of the program. This is also supported by the documentation, since each included file is aware of its parent's context (e.g., variable values).

- The three files exist in the same directory, meaning the include targets are resolved directly, and correctly.

#	Active / Omitted parts	#	Active / Omitted parts	#	Active / Omitted parts
1	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	2	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	3	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2
4	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	5	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	6	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2
7	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	8	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	9	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2
10	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	11	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2	12	a1 | b1 | c1 inc "b" | inc "c" | inc "a" a2 | b2 | c2

Table 4.1: A set of twelve tests, each representing a different combination of instructions.

Table 4.2 depicts the behavior of the four static analysis tools and the PHP Zend engine, when processing the examples in Table 4.1. The columns *Outcome* present the behavior of the tool/engine in question, when processing the corresponding line's example. The columns *Status* indicate whether the resulting behavior is correct and/or desirable (i.e., *OK* or *NOT OK*). Red symbols indicate that the respective file is being included (it is only used for the inclusion of file ***a*** by file ***c***). Bold text represents an incorrect behavior.

Pixy resolves include directives correctly. We can observe that the tool ignores cycle makers, regardless of the type of include directive applied. This approach may lead to false negatives for *include*. For instance, a sink that is not reached by any tainted input during the first inclusion may become reachable on future inclusions, depending on new entry points that may occur along the chain of execution.

phpSAFE resolves include directives correctly. However, it contains a bug; it does not keep track of variable taintedness between files. If the *include_once* directive is used, the tool will revisit the cycle maker if it is also used as the input file. Such mechanism is unnecessary for *include_once*. If the *include* directive is used, it will trigger infinite file inclusion. This is an undesirable behavior, as it leads to excessive memory consumption that crashes phpSAFE.

RIPS resolves include directives correctly. The tool will revisit the cycle maker if it is also used as the input file, regardless of the type of include directive applied. Such mechanism is unnecessary for *include_once*. It is also not the best approach for *include*, since it only revisits one file.

WAP resolves include files correctly. However, it does not manage its internal symbol tables properly, in case of a cycle. This situation leads to empty analyses, regardless of the type of include directive applied. We also noted that the propagation of taintedness between files is not assured in the case of XSS vulnerabilities.

The PHP Zend engine ignores cycle makers as expected, when processing *include_once*

directives. However, it presents infinite file inclusion when processing *include* directives. Once again, this behavior is not desirable, as it leads to excessive memory consumption.

Tool/Engine	#	include_once		include	
		Outcome	Status	Outcome	Status
Pixy	1	a1, b1, a2	OK	a1, b1, a2	OK
	2	a1, b1, c1, a2	OK	a1, b1, c1, a2	OK
	3	a1, b1, c1, b2	OK	a1, b1, c1, b2	OK
	4	a1, b1, c1, b2, a2	OK	a1, b1, c1, b2, a2	OK
	5	a1, b1, c1, a	OK	a1, b1, c1, a	NOT OK
	6	a1, b1, c1, a, a2	OK	a1, b1, c1, a, **a2**	NOT OK
	7	a1, b1, c1, a, b2	OK	a1, b1, c1, a, **b2**	NOT OK
	8	a1, b1, c1, a, b2, a2	OK	a1, b1, c1, a, **b2, a2**	NOT OK
	9	a1, b1, c1, a, c2	OK	a1, b1, c1, a, **c2**	NOT OK
	10	a1, b1, c1, a, c2, a2	OK	a1, b1, c1, a, **c2, a2**	NOT OK
	11	a1, b1, c1, a, c2, b2	OK	a1, b1, c1, a, **c2, b2**	NOT OK
	12	a1, b1, c1, a, c2, b2, a2	OK	a1, b1, c1, a, **c2, b2, a2**	NOT OK
phpSAFE	1	a1, b1, a2	OK	a1, b1, a2	OK
	2	a1, b1, c1, a2	OK	a1, b1, c1, a2	OK
	3	a1, b1, c1, b2	OK	a1, b1, c1, b2	OK
	4	a1, b1, c1, b2, a2	OK	a1, b1, c1, b2, a2	OK
	5	a1, b1, c1, a, **a1**	NOT OK	**crash - infinite recursion**	NOT OK
	6	a1, b1, c1, a, **a1, a2, a2**	NOT OK	**crash - infinite recursion**	NOT OK
	7	a1, b1, c1, a, **a1, b2**	NOT OK	**crash - infinite recursion**	NOT OK
	8	a1, b1, c1, a, **a1, a2, b2, a2**	NOT OK	**crash - infinite recursion**	NOT OK
	9	a1, b1, c1, a, **a1, c2**	NOT OK	**crash - infinite recursion**	NOT OK
	10	a1, b1, c1, a, **a1, a2, c2, a2**	NOT OK	**crash - infinite recursion**	NOT OK
	11	a1, b1, c1, a, **a1, c2, b2**	NOT OK	**crash - infinite recursion**	NOT OK
	12	a1, b1, c1, a, **a1, a2, c2, b2, a2**	NOT OK	**crash - infinite recursion**	NOT OK
RIPS	1	a1, b1, a2	OK	a1, b1, a2	OK
	2	a1, b1, c1, a2	OK	a1, b1, c1, a2	OK
	3	a1, b1, c1, b2	OK	a1, b1, c1, b2	OK
	4	a1, b1, c1, b2, a2	OK	a1, b1, c1, b2, a2	OK
	5	a1, b1, c1, a, **a1**	NOT OK	a1, b1, c1, a, **a1**	NOT OK
	6	a1, b1, c1, a, **a1, a2, a2**	NOT OK	a1, b1, c1, a, **a1, a2, a2**	NOT OK
	7	a1, b1, c1, a, **a1, b2**	NOT OK	a1, b1, c1, a, **a1, b2**	NOT OK
	8	a1, b1, c1, a, **a1, a2, b2, a2**	NOT OK	a1, b1, c1, a, **a1, a2, b2, a2**	NOT OK
	9	a1, b1, c1, a, **a1, c2**	NOT OK	a1, b1, c1, a, **a1, c2**	NOT OK
	10	a1, b1, c1, a, **a1, a2, c2, a2**	NOT OK	a1, b1, c1, a, **a1, a2, c2, a2**	NOT OK
	11	a1, b1, c1, a, **a1, c2, b2**	NOT OK	a1, b1, c1, a, **a1, c2, b2**	NOT OK
	12	a1, b1, c1, a, **a1, a2, c2, b2, a2**	NOT OK	a1, b1, c1, a, **a1, a2, c2, b2, a2**	NOT OK
WAP	1	a1, b1, a2	OK	a1, b1, a2	OK
	2	a1, b1, c1	NOT OK	a1, b1, c1	NOT OK
	3	a1, b1, c1	NOT OK	a1, b1, c1	NOT OK
	4	a1, b1, c1	NOT OK	a1, b1, c1	NOT OK
	5	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	6	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	7	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	8	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	9	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	10	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	11	**empty analysis**	NOT OK	**empty analysis**	NOT OK
	12	**empty analysis**	NOT OK	**empty analysis**	NOT OK
PHP Zend	1	a1, b1, a2	OK	a1, b1, a2	OK
	2	a1, b1, c1, a2	OK	a1, b1, c1, a2	OK
	3	a1, b1, c1, b2	OK	a1, b1, c1, b2	OK
	4	a1, b1, c1, b2, a2	OK	a1, b1, c1, b2, a2	OK
	5	a1, b1, c1, a	OK	**crash - infinite recursion**	NOT OK
	6	a1, b1, c1, a, a2	OK	**crash - infinite recursion**	NOT OK
	7	a1, b1, c1, a, b2	OK	**crash - infinite recursion**	NOT OK
	8	a1, b1, c1, a, b2, a2	OK	**crash - infinite recursion**	NOT OK
	9	a1, b1, c1, a, c2	OK	**crash - infinite recursion**	NOT OK
	10	a1, b1, c1, a, c2, a2	OK	**crash - infinite recursion**	NOT OK
	11	a1, b1, c1, a, c2, b2	OK	**crash - infinite recursion**	NOT OK
	12	a1, b1, c1, a, c2, b2, a2	OK	**crash - infinite recursion**	NOT OK

Table 4.2: Instruction order when processing the examples in Table 4.1, and whether it meets the expected outcome.

4.3 Cycle caused by a file containing functions

The batch of tests is based on the format illustrated in Figure 4.2. This format implies the existence of three files (***a***, ***b*** and ***c***). Files ***a*** and ***c*** perform function calls to file ***b***, which is the cycle maker. Once again, include relationships are represented by solid arrows, while the dotted arrows indicate that the execution reaches the end of file, thus returning to the previous parent file. The twelve examples present in Table 4.3 are valid for both *require/include* and *require_once/include_once* directives. If function ***b2***'s declaration is activated, it will be called instead of ***b1***, as seen in examples #3 and #4.

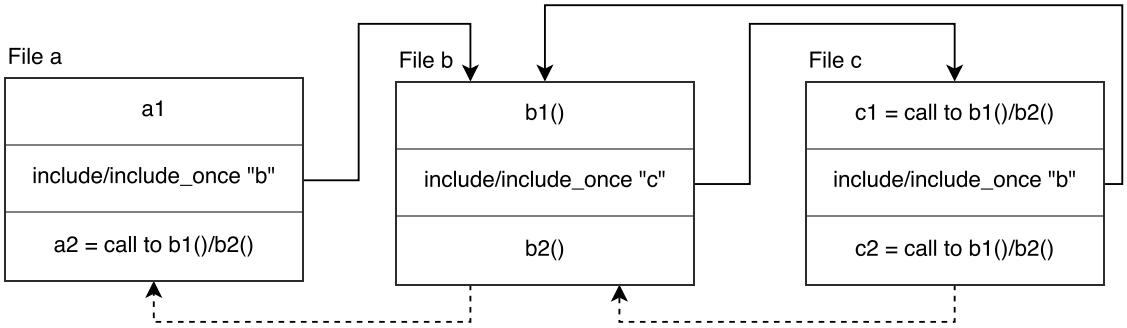

Figure 4.2: Baseline use case where cycle maker contains functions only.

For example, test #5 begins in file ***a*** by executing the block ***a1***, which is the only instruction that does not contain a function call. Such is due to no function being available at that point (i.e., it is necessary to visit ***b*** first, since it contains the function definitions). After ***a1***, the execution proceeds by including file ***b*** (***inc "b"***) and collecting function ***b1***. Note that ***b2*** would also be collected at this point, if it existed within the context of the example. During this file's execution, after the functions are gathered, file ***c*** is included (***inc "c"***), and function ***b1*** is called. After this function call, the second inclusion of ***b*** may trigger a cycle between these two files. Note that the lower blocks of this example are deactivated (i.e., grayed out). That is, the call to ***b1*** in file ***a***, function ***b2***, and the second call to ***b1*** in file ***c*** do not take part in this execution chain. We make the following assumptions:

- For the examples #5 through #12, a cycle is eventually formed during the inclusion of file ***b*** by file ***c***. According to the documentation, the desirable outcome should be ***a1***, ***include_once b***, ***b1***, ***include_once c***, ***c1***, ***c2***, ***b2***, ***a2*** for *include_once* (the second inclusion of ***b*** is ignored, since it already exists in the execution chain). For *include*, it should be ***a1***, ***include b***, ***b1***, ***include c***, ***c1***, ***infinite cycle*** (the second inclusion of ***b*** will likely lead to the infinite cycle, as this directive does not check for duplicate files in the chain) [55].

- Since all functions are collected once file ***b*** is included, they will be available for the remainder of the execution (i.e., they have global scope [55]).

- The three files exist in the same directory, meaning the include targets are resolved directly, and correctly.

#	Active / Omitted parts	#	Active / Omitted parts	#	Active / Omitted parts
1	a1 | b1 | b1() inc "b" | inc "c" | inc "b" b1() | b2 | b1()	2	a1 | b1 | b1() inc "b" | inc "c" | inc "b" b1() | b2 | b1()	3	a1 | b1 | b2() inc "b" | inc "c" | inc "b" b2() | b2 | b2()
4	a1 | b1 | b2() inc "b" | inc "c" | inc "b" b2() | b2 | b2()	5	a1 | b1 | b1() inc "b" | inc "c" | inc "b" b1() | b2 | b1()	6	a1 | b1 | b1() inc "b" | inc "c" | inc "b" b1() | b2 | b1()
7	a1 | b1 | b2() inc "b" | inc "c" | inc "b" b2() | b2 | b2()	8	a1 | b1 | b2() inc "b" | inc "c" | inc "b" b2() | b2 | b2()	9	a1 | b1 | b1() inc "b" | inc "c" | inc "b" b1() | b2 | b1()
10	a1 | b1 | b1() inc "b" | inc "c" | inc "b" b1() | b2 | b1()	11	a1 | b1 | b2() inc "b" | inc "c" | inc "b" b2() | b2 | b2()	12	a1 | b1 | b2() inc "b" | inc "c" | inc "b" b2() | b2 | b2()

Table 4.3: A set of twelve tests, each representing a different combination of instructions.

Table 4.4 depicts the behavior of the four static analysis tools and the PHP Zend engine, when processing the examples in Table 4.3. Blue symbols indicate the first inclusion of the file containing functions (it is always ***b***). Overall, the results in Table 4.4 manifest similarities with the results in Table 4.2. Pixy and phpSAFE present the same basic symptoms and order of operations. Even though RIPS still revisits the cycle maker in the *include_once* column, it has no effect on the analysis, since the cycle maker only contains functions. The big differences are observable in WAP and the PHP Zend engine.

WAP manifests a different symptom (NullPointerException) from the one in Table 4.2 (empty analysis). This discrepancy is due to the different nature of the function tests, since they imply that the cycle begins with the inclusion of file ***b*** by file ***c*** (i.e., instead of the inclusion of file ***a*** by file ***c***). In order to understand how NullPointerException is triggered, one must be acquainted with two important structures of WAP - the Main Symbol Table (MST) and the Main Include File Table (MIFT). The MST keeps track of all the files (represented as STs) created by the Taint Analyzer. As the name suggests, it holds the main files to be used in the beginning of the taint analysis phase. The MIFT also contains STs, yet the files being represented are included by the ones in MST. Internally, ***b*** is moved from WAP's MST to the MIFT. When it attempts to move ***b*** into the MIFT for the second time, WAP crashes with NullPointerException, since the file no longer exists in its original location, the MST. The inclusion of file ***a*** by file ***c*** does not suffer from this issue, since ***a*** serves as the initial input file. However, the logic behind both problems lies in an identical part of the code. This notion is further discussed in the Chapter 5.

In order to understand other details of PHP Zend's behavior, we also made a set of alternative tests. These alternative tests contain *redeclaration guards* in file ***b***. This tech-

nique prevents errors regarding function redeclaration, and may be used in practice instead of *include_once* for inclusion of function files. It consists in the use of a guard which encloses the function body (i.e., *if !function_exists("b1") then declare function "b1"*). The objective was to check whether an infinite cycle may occur, instead of the function redeclaration error at the end of Table 4.4. As such, we developed alternative versions of examples #5 through #12. They are, once again, based on Table 4.3, except a guard is added before ***b1*** and/or ***b2***. Alternative tests #5, #6, #9 and #10 result in a crash due to infinite recursion. The other alternative tests, #7, #8, #11 and #12, crash due to nonrecognition of function ***b2***. The latter batch presents an interesting phenomenon. Their regular counterparts (at the end of Table 4.4, *include_once* column) show that all functions are collected once the functions file is visited for the first time. However, that is no longer the case if the function is enclosed with a redeclaration guard, as ***b2*** is not collected before the inclusion of file ***c***. Hence, when ***b2*** is called from file ***c***, it is not recognized. According to the results, the existence of redeclaration guards does not influence the outcome of the tests when using static analysis tools.

Tool/Engine	#	include_once		include	
		Outcome	Status	Outcome	Status
Pixy	1	a1, b, a2->b1	OK	a1, b, a2->b1	OK
	2	a1, b, c1->b1, a2->b1	OK	a1, b, c1->b1, a2->b1	OK
	3	a1, b, c1->b2	OK	a1, b, c1->b2	OK
	4	a1, b, c1->b2, a2->b2	OK	a1, b, c1->b2, a2->b2	OK
	5	a1, b, c1->b1, b	OK	a1, b, c1->b1, b	NOT OK
	6	a1, b, c1->b1, b, a2->b1	OK	a1, b, c1->b1, b, **a2->b1**	NOT OK
	7	a1, b, c1->b2, b	OK	a1, b, c1->b2, b	NOT OK
	8	a1, b, c1->b2, b, a2->b2	OK	a1, b, c1->b2, b, **a2->b2**	NOT OK
	9	a1, b, c1->b1, b, c2->b1	OK	a1, b, c1->b1, b, **c2->b1**	NOT OK
	10	a1, b, c1->b1, b, c2->b1, a2->b1	OK	a1, b, c1->b1, b, **c2->b1, a2->b1**	NOT OK
	11	a1, b, c1->b2, b, c2->b2	OK	a1, b, c1->b2, b, **c2->b2**	NOT OK
	12	a1, b, c1->b2, b, c2->b2, a2->b2	OK	a1, b, c1->b2, b, **c2->b2, a2->b2**	NOT OK
phpSAFE	1	a1, b, a2->b1	OK	a1, b, a2->b1	OK
	2	a1, b, c1->b1, a2->b1	OK	a1, b, c1->b1, a2->b1	OK
	3	a1, b, c1->b2	OK	a1, b, c1->b2	OK
	4	a1, b, c1->b2, a2->b2	OK	a1, b, c1->b2, a2->b2	OK
	5	a1, b, c1->b1, b	OK	**crash - infinite recursion**	NOT OK
	6	a1, b, c1->b1, b, a2->b1	OK	**crash - infinite recursion**	NOT OK
	7	a1, b, c1->b2, b	OK	**crash - infinite recursion**	NOT OK
	8	a1, b, c1->b2, b, a2->b2	OK	**crash - infinite recursion**	NOT OK
	9	a1, b, c1->b1, b, c2->b1	OK	**crash - infinite recursion**	NOT OK
	10	a1, b, c1->b1, b, c2->b1, a2->b1	OK	**crash - infinite recursion**	NOT OK
	11	a1, b, c1->b2, b, c2->b2	OK	**crash - infinite recursion**	NOT OK
	12	a1, b, c1->b2, b, c2->b2, a2->b2	OK	**crash - infinite recursion**	NOT OK
RIPS	1	a1, b, a2->b1	OK	a1, b, a2->b1	OK
	2	a1, b, c1->b1, a2->b1	OK	a1, b, c1->b1, a2->b1	OK
	3	a1, b, c1->b2	OK	a1, b, c1->b2	OK
	4	a1, b, c1->b2, a2->b2	OK	a1, b, c1->b2, a2->b2	OK
	5	a1, b, c1->b1, b	OK	a1, b, c1->b1, b	NOT OK
	6	a1, b, c1->b1, b, a2->b1	OK	a1, b, c1->b1, b, **a2->b1**	NOT OK
	7	a1, b, c1->b2, b	OK	a1, b, c1->b2, b	NOT OK
	8	a1, b, c1->b2, b, a2->b2	OK	a1, b, c1->b2, b, **a2->b2**	NOT OK
	9	a1, b, c1->b1, b, c2->b1	OK	a1, b, c1->b1, b, **c2->b1**	NOT OK
	10	a1, b, c1->b1, b, c2->b1, a2->b1	OK	a1, b, c1->b1, b, **c2->b1, a2->b1**	NOT OK
	11	a1, b, c1->b2, b, c2->b2	OK	a1, b, c1->b2, b, **c2->b2**	NOT OK
	12	a1, b, c1->b2, b, c2->b2, a2->b2	OK	a1, b, c1->b2, b, **c2->b2, a2->b2**	NOT OK
WAP	1	a1, b, a2->b1	OK	a1, b, a2->b1	OK
	2	a1, b, c1->b1, a2->b1	OK	a1, b, c1->b1, a2->b1	OK
	3	a1, b, c1->b2	OK	a1, b, c1->b2	OK
	4	a1, b, c1->b2, a2->b2	OK	a1, b, c1->b2, a2->b2	OK
	5	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	6	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	7	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	8	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	9	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	10	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	11	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
	12	**crash - NullPointerException**	NOT OK	**crash - NullPointerException**	NOT OK
PHP	1	a1, b, a2->b1	OK	a1, b, a2->b1	OK
	2	a1, b, c1->b1, a2->b1	OK	a1, b, c1->b1, a2->b1	OK
	3	a1, b, c1->b2	OK	a1, b, c1->b2	OK
	4	a1, b, c1->b2, a2->b2	OK	a1, b, c1->b2, a2->b2	OK
	5	a1, b, c1->b1, b	OK	**crash - redeclared function**	NOT OK
	6	a1, b, c1->b1, b, a2->b1	OK	**crash - redeclared function**	NOT OK
	7	a1, b, c1->b2, b	OK	**crash - redeclared function**	NOT OK
	8	a1, b, c1->b2, b, a2->b2	OK	**crash - redeclared function**	NOT OK
	9	a1, b, c1->b1, b, c2->b1	OK	**crash - redeclared function**	NOT OK
	10	a1, b, c1->b1, b, c2->b1, a2->b1	OK	**crash - redeclared function**	NOT OK
	11	a1, b, c1->b2, b, c2->b2	OK	**crash - redeclared function**	NOT OK
	12	a1, b, c1->b2, b, c2->b2, a2->b2	OK	**crash - redeclared function**	NOT OK

Table 4.4: Instruction order when processing the examples in Table 4.3, and whether it meets the expected outcome.

Chapter 5

Solving Circular Dependencies

This chapter presents the proposed approaches to solve the issues identified in Chapter 4. It begins with the approach to distinguish between the *include/require* and *include_once/require_once* directives in the semantic analysis. We present the *cycle maker identification* algorithm, which solves the incorrect management of STs via identification of roots and separation of file dependencies in the PHP project. The algorithm also enhances the completeness of the analysis by identifying potential *ring components* of the PHP project. A ring component occurs when there are no roots among files that participate in a circular dependency. For example, when ***a*** includes ***b***, ***b*** includes ***c***, and ***c*** includes ***a***. Lastly, we present the *solving circular dependencies* algorithm, which circumvents cycles caused by circular dependencies, during taint analysis.

5.1 WAP's Internal Structures

These approaches were implemented in the WAP tool. However, in order to understand the cause of these problems and where changes have to be made to solve them, some notions concerning WAP's internal structures are necessary. WAP's analysis consists of two main phases:

1. Semantic analysis. During this phase, the lexer breaks the source code into tokens, producing a token stream. The PHP parser then takes these tokens, as well as a set of grammatical rules, and builds an AST. Just before the taint analysis phase, the ASTs are navigated by tree walkers for the production of STs (as explained in Section 2.3.1). The tree walkers are also parsers, and are supported by a set of grammatical rules separate from that of PHP's parser. These STs may be moved to either WAP's MST, or to WAP's MIFT. The MST keeps track of files that will be used in the beginning of the taint analysis phase, while the MIFT is only meant to store files which are included by other files in the MST.

   WAP's results in Table 4.2 and Table 4.4 indicate that the tool does not distinguish between *include/require* and *include_once/require_once* directives. That part of the

problem lies in the operation of the tree walkers, since this phase is also responsible for collecting information about file dependencies. In addition, Table 4.2 shows that WAP does not proceed to the taint analysis phase in the case of circular dependencies. In particular, the tool does not process *ring components*. For example, when ***a*** includes ***b***, ***b*** includes ***c***, and ***c*** includes ***a***, all of the three files are moved to the MIFT, since they are all include files. There is no main file from which to start, causing the taint analysis phase not to occur. We conclude that the mismanagement of the STs also occurs during the operation of the tree walkers. Premature exchange of STs between the MST and the MIFT may also explain the symptoms documented in Table 4.4. Solving problems in this phase is not enough, however. Even with a proper main file in the MST, and correct exchange of symbol tables, circular dependencies will cause infinite cycles during taint analysis, since WAP does not feature a cycle resolution mechanism.

2. Taint analysis. This phase follows the entry point values within the program, considering the various possible paths and dependencies, and detects if these values reach a sensitive sink. Tainted Execution Path Trees (TEPT's) are populated with taintedness information during taint analysis. Each branch contains a tainted variable and a sub-branch for each line in the code where the variable becomes tainted. Each sub-branch contains a list of dependencies that the tainted variable propagated its taintedness into. Naturally, this phase is sensitive to circular dependencies that may occur along the paths, since it needs to follow the values between files. A cycle resolution feature should be put in place. In addition, the remaining *NOT OK* cases in Table 4.2 are likely caused by implementation errors during this phase.

5.2 Distinction between Include Directives

As previously explained, PHP supports two file inclusion directives, *include/require* and *include_once/require_once*. According to WAP's results in Table 4.2 and Table 4.4, the tool does not seem to distinguish between the two types of directives, as it presents the exact same behavior when processing both. The distinction between the two types of directives should be done during semantic analysis, and the logic for processing each of the two cases should be included within taint analysis. During semantic analysis, if the processed PHP file contains a call to *include/require* or *include_once/require_once*, then the corresponding AST would contain a node representing such call. The node is then used by the tree walkers to store file relationship information in the respective ST. Consequently, our first solution consists in modifying the tree walker grammar files to make the distinction at the node level.

The tree walker contains a rule called *simplerequire*, which is activated whenever the current node matches the format *^(RequireOperator expression)*. WAP's original im-

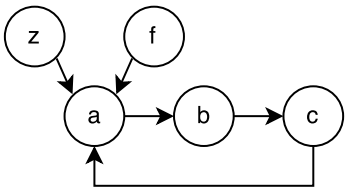

Figure 5.1: A set of files and their relationships. It contains two starting points - ***z*** and ***f*** - and a cycle between ***c*** and ***a***.

plementation does not make a distinction between these types of operators, treating and identifying all four possible operators in the ST as *include*. By checking the existence of the *once* substring in $RequireOperator.text construct (the construct containing PHP include data in the AST), the distinction becomes possible. Finally, the operator type is saved in the current ST. This is a simple, yet important step for the correct operation of two novel algorithms that we have developed, which are explained in more detail in the rest of the chapter.

5.3 Cycle Maker Identification Algorithm

We have seen that circular dependencies are the main cause of incorrect management of STs during semantic analysis. In order to solve the the circular dependency issue, we propose an algorithm capable of identifying cycle makers. The algorithm identifies *file inclusion paths* and associates circular dependency points (i.e., the cycle makers) within such paths. We define a file inclusion path as being a chain of files composed by inclusion files and terminated by cycle makers. This means that a file can include one or mores files, thus creating a chain which stops when a cycle maker is identified. Before the correct identification of these paths, it is mandatory to discover its starting points, i.e., the files that are not included by any other file. We call such starting points *root files*. However, ring components are particular cases in which there is not clearly a root file, because all files include at least one file. In these cases, one of the participants in the ring takes the role of the root file. Identifying the root files is crucial for the precise and correct detection of vulnerabilities, since the taint analysis process must begin at the correct starting point. Figure 5.1 exemplifies five files and their inclusion dependencies. The algorithm identifies two paths $\{$***z, a, b, c, a***$\}$ and $\{$***f, a, b, c, a***$\}$, where ***f*** and ***z*** are signaled as being root files, and ***a*** as the cycle maker. If ***f*** and ***z*** files did not exist we would have a ring formed by $\{$***a, b, c***$\}$. In that case, one of the files would be chosen as the root. For example, if ***b*** was chosen we would have the $\{$***b, c, a, b***$\}$ path, with ***b*** as the cycle maker. In certain ring configurations, some files may not be able to reach every other file. The algorithm is ready for such possibility by calculating the number of inclusions for each file. The candidate with the most connections is chosen as the root.

name	successors	ancestors	operator type	cycle maker
z	a	-	-	-

Figure 5.2: A node representing a single file and its relationships.

As a PHP project may contain multiple starting points, it may have several file inclusion paths, as many as the number of starting points. However, it is not possible to make the assumption that a file that was already included in a path can not be included again in another path, since the project analysis would be incomplete or erroneous. For example, in Figure 5.1, consider the include file ***a***, which is called by ***z*** and ***f*** with the *include_once* directive. Just because ***a*** is included once by ***z***, it does not mean ***a*** should be ignored when included by ***f***. Clearly, it is necessary to analyze each possible path, given each possible starting point. The existence of circular dependencies should be verified relative to each path. After finding the root files, inclusion paths, circular dependencies in said paths, and potential ring components, it becomes safe to move the include files from the MST to the MIFT.

As such, we present an algorithm that implements these notions. Algorithm 1 receives a set of input files (or a PHP project directory), which contain dependency information within them. The objectives are fivefold:

1. Find all root files;
2. Find all file inclusion paths, starting in each root file identified;
3. Find all cycle makers in file inclusion paths (i.e., the files responsible for causing circular dependency points);
4. Move the appropriate include files from MST to MIFT;
5. Check for ring components and define root files for them, and their inclusion paths.

Each file in the inclusion paths is represented by a node. A node is characterized by its name, its successors, its ancestors, the type of include operator and the cycle maker. The *name* is identical to the name of the file being represented. The *successors* field refers to a file's direct children (i.e., dependencies which are explicitly stated in the parent file). The *ancestors* field refers to a file's direct and indirect parents (i.e., all of the previous files in the chain). While the *successors* field merely serves as a guide for recursively building the chain, the *ancestors* field is the key to identify circular dependencies. If the current node name matches the direct parent's name, or any of the other ancestors' names, that means the current node is a circular dependency point. As such, the parent node's *cycle maker* field is updated. Finally, the *operator type* field simply refers to the type of directive used to include the current file. Observe the Figure 5.2, once again based on Figure 5.1. We

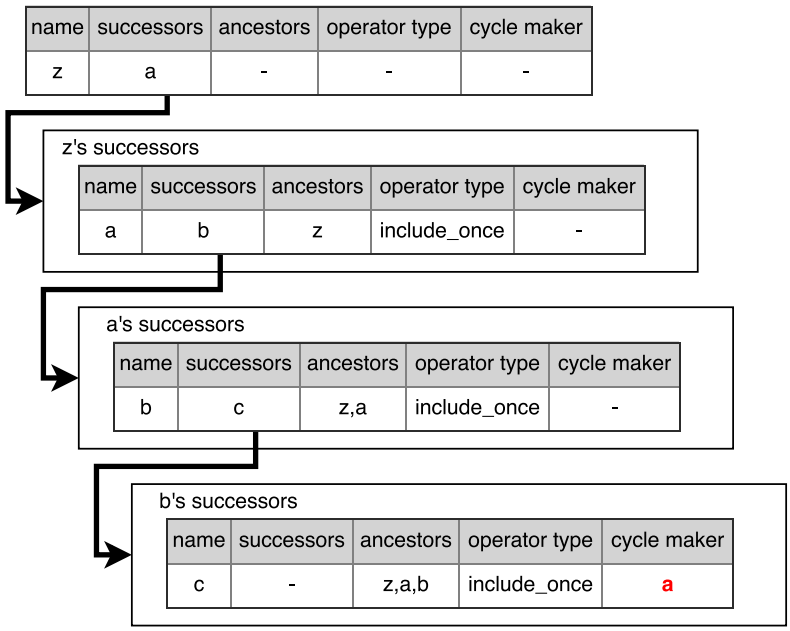

Figure 5.3: A chain of nodes representing a single path.

have already established that the algorithm identifies two paths, one of them being {***z, a, b, c, a***}. Consider the file ***z***, which is represented by the node with *name* ***z***. It contains ***a*** as its direct successor, as illustrated in Figure 5.1. It contains no ancestors, since it is a root file. As a result, it does not have an associated operator type. Finally, there are no cycle makers at this point in the chain, and the field is empty.

Figure 5.3 illustrates the creation of the chain of nodes for the example. Each inclusion path begins with a root node. The example uses ***z*** as the root node, but recall that another path starting with ***f*** should also exist.

Each chain is formed by the file relationships identified in the beginning of the semantic analysis phase. The chain ends when it identifies a circular dependency. A node causes circular dependency if its name matches the direct parent's name, or any of the other ancestors' names. Such is the case with file ***a*** when included by file ***c*** (i.e., it becomes ***c***'s ancestor and successor simultaneously). The knowledge concerning where circular dependencies occur is fundamental for the correct functioning of the remainder of semantic analysis (i.e., moving STs from the MST to the MIFT in an orderly fashion). It is also useful for the taint analysis phase, which will decide how to deal with these circular dependencies.

The Algorithm 1 presents the code of the algorithm that we have been describing. For a better understanding of this main algorithm, it has been split in six parts. Initially, the algorithm obtains an AST and an empty ST per each input file (lines 5-6). Each ST is added to the MST (line 7). After these structures have been created, the include instructions are iterated (line 8), their information is extracted (lines 10-11) and stored in the current file's ST (line 12). At this point, we know the basic relationships between files.

Algorithm 1 Detection of root files, file inclusion paths and circular dependencies, given a set of files and their relationships.

Input: Directory of PHP files *directory*
Output: A list of file inclusion paths with identified cycle makers *paths*

```
 1: MST ← create MST
 2: roots ← empty list
 3: paths ← empty list
 4: for each file ∈ directory do
 5:     AST ← create file AST
 6:     ST ← create ST
 7:     MST ← MST + ST
 8:     for each include ∈ AST do
 9:         child ← create child
10:         child.name ← include.name
11:         child.operator ← include.operator
12:         ST.children ← ST.children + child
13:     end for
14: end for
15: for each st1 ∈ MST do
16:     isRoot ← 1
17:     for each st2 ∈ MST do
18:         for each child ∈ st2.children do
19:             if st1.name = child.name then
20:                 isRoot ← 0
21:                 break
22:             end if
23:         end for
24:         if isRoot = 0 then
25:             break
26:         end if
27:     end for
28:     if isRoot = 1 then
29:         roots ← roots + st1
30:     end if
31: end for
32: for each root ∈ roots do
33:     p ← create empty node path
34:     BUILDPATH(root, p, root.children, nil, nil, nil, MST)
35:     paths ← paths + p
36: end for
37: CALCRINGDEPENDENCIES(MST, roots, paths)
```

We can proceed to line 15, which iterates the MST with the intent of searching for root files (lines 17-27) and adding them to a list of root files (line 29). At this point, we are ready to produce file inclusion paths from the roots. Line 32 iterates over the list of root files in order to discover inclusion paths and circular dependencies (cycle makers) among the files (line 34). Each newly produced path is added to a list of paths in line 35. At the end of this process, it is mostly safe to move files from the MST to the MIFT. However, we discussed that ring components do not contain root files, so the files involved cannot be processed by lines 17-27. We need to classify certain files in the ring components as roots, so that they become part of the analysis. Such task is performed in line 37.

Algorithm 2 Recursive extraction of a file inclusion path given a root file and its children. Auxiliary for Algorithm 1 and Algorithm 3.

Input: File for path extraction *cs*
Node representation of file *node*
File's children *l_children*
Node's ancestors *anc*
Node's parent node *nodeparent*
Include operator used to include file *op*
The set of STs *MST*
Output: Recursive file inclusion path *node*

```
1:  function BUILDPATH(cs, node, l_children, anc, nodeparent, op, MST)
2:      for each a ∈ anc do
3:          if cs.name = a.name ∨ nodeparent.name = cs.name then
4:              nodeparent.cycle_maker ← cs
5:              return
6:          end if
7:      end for
8:      if node = nil then
9:          node ← create node
10:         if nodeparent ≠ nil then
11:             nodeparent.successors ← nodeparent.successors + node
12:             node.ancestors ← node.ancestors + nodeparent.ancestors
13:             node.ancestors ← node.ancestors + nodeparent
14:             node.include_operator ← op
15:         end if
16:     end if
17:     node.name ← cs.name
18:     for each child ∈ l_children do
19:         BUILDPATH(MST.get(child), nil, MST.get(child).children,
                node.ancestors, node, cs.children.get(child).operator)
20:     end for
21: end function
```

Consider Algorithm 2 for the extraction of file inclusion paths and cycle makers. This algorithm contains the *BuildPath* function, which is called by Algorithm 1 (in line 34) and Algorithm 3 (in line 18). A file causes circular dependency if it possesses an ancestor that is also its successor (lines 2 to 7). If a circular dependency is detected, its information is stored in the parent node (line 4), and we can safely abandon the cycle maker (line 5). Otherwise, we proceed by populating the current file's inclusion path node (lines 11-14,17 populate a single node). These nodes posses the format previously exemplified in Figure 5.2. The algorithm works recursively by processing each of the current file's children (lines 18-20). The output is a file inclusion path, an example of which is present in Figure 5.3.

Algorithm 3 Detection of ring dependencies within the MST. Auxiliary for Algorithm 1.

Input: The set of STs *MST*
A set of root files *roots*
A set of file include paths *paths*
Output: A list of file inclusion paths with identified cycle makers *paths*

```
1: function CALCRINGDEPENDENCIES(MST, roots, paths)
2:     while true do
3:         partofmst ← MISSINGSTS(MST, roots)
4:         if partofmst = nil then
5:             break
6:         end if
7:         bestWeight ← −1
8:         bestTable ← null
9:         for each st ∈ partofmst do
10:            temp ← CALCNUMEDGES(st, null, MST)
11:            if temp > bestWeight then
12:                bestWeight ← temp
13:                bestTable ← st
14:            end if
15:        end for
16:        roots ← roots + bestTable
17:        p ← create empty node path
18:        BUILDPATH(bestTable, p, bestTable.children, nil, nil, nil, MST)
19:        paths ← paths + p
20:    end while
21: end function
```

Consider Algorithm 3 for the detection of ring dependencies within the MST. The *CalcRingDependencies* function is called by Algorithm 1 in line 37. Line 3 searches for STs which are not covered by any of the previously identified roots. If there are none missing (line 4), this algorithm ends in line 5. Lines 7-15 check which of the missing ring component STs is the most appropriate candidate for a root. The candidate with the highest score is added to the set of roots, in line 16. The candidate is used in line 18 to build a file inclusion path and eventually detect a circular dependency. Line 19 updates the list of paths with the newly built path.

Consider Algorithm 4, which finds STs not covered by any roots. The *missingSTs* function is called by Algorithm 3 in line 3. Lines 3-14 test each ST in the MST individually. Lines 5-10 check if any of the roots cover the tested file. In particular, line 6

Algorithm 4 Detection of missing STs (i.e., files the are not covered by any roots). Auxiliary for Algorithm 3 for detection of ring dependencies.

Input: The set of STs *mst*
A set of roots *roots*
Output: A set of STs which are not covered by the roots *partofmst*

```
1: function MISSINGSTS(mst, roots)
2:     partofmst ← empty list
3:     for each st ∈ mst do
4:         found ← 0
5:         for each rst ∈ roots do
6:             if rst.name = st.name ∨ CONTAINSST(st.name, rst, null, mst) then
7:                 found ← 1
8:                 break
9:             end if
10:        end for
11:        if found = 0 then
12:            partofmst ← partofmst + st
13:        end if
14:    end for
15:    return partofmst
16: end function
```

Algorithm 5 Recursive calculation of file coverage given a root. Auxiliary to Algorithm 4 for the detection of missing STs.

Input: The filename to look up *stname*
The current file to compare against *current*
A list of visited files *exclude*
The set of STs *mst*
Output: Indication of whether the file was found *bool*

```
1:  function CONTAINSST(stname, current, exclude, mst)
2:      bool ← false
3:      if exclude ≠ nil ∧ current.name ∈ exclude then
4:          return 0
5:      end if
6:      if current.name = stname then
7:          return 1
8:      end if
9:      newExclude ← nil
10:     if exclude ≠ nil then
11:         newExclude ← copy(exclude)
12:     else
13:         newExclude ← empty list
14:     end if
15:     newExclude ← newExclude + current.name
16:     for each child ∈ current.children do
17:         bool ← bool ∨ CONTAINSST(stname, mst.get(child), newExclude, mst)
18:     end for
19:     return bool
20: end function
```

checks whether the currently evaluated ST matches the current root **or** if a mention of the ST exists somewhere along the current root's includes, as well as the respective children's includes. If the search is successful, we know that the tested file exists within the context of the roots, and we may proceed to the next ST (line 8). Otherwise, we add the ST to a partial MST (line 12), qualifying it as a ring dependency root candidate for Algorithm 3.

Consider Algorithm 5, which checks if a given file is referenced somewhere within a file's includes, or in the respective children's includes. The *containsST* function is called by Algorithm 4 in line 6. It implies a name comparison operation (line 6) between the filename and each file in the chain, while avoiding recursion by keeping track of each file visited (lines 11 and 15). If at least one comparison is successful (line 17), that means the file is accessible by a root.

Algorithm 6 Recursive calculation of edges within a ring dependency. Auxiliary to Algorithm 3 for the detection of ring dependencies.

Input: A file *st*
A list of visited files *exclude*
The set of STs *mst*
Output: Sum of the file's exiting edges *sum*

```
1:  function CALCNUMEDGES(st,exclude,mst)
2:      if exclude ≠ nil ∧ st.name ∈ exclude then
3:          return 0
4:      end if
5:      sum ← 0
6:      newExclude ← nil
7:      if exclude ≠ nil then
8:          newExclude ← copy(exclude)
9:      else
10:         newExclude ← empty list
11:     end if
12:     newExclude ← newExclude + st.name
13:     for each child ∈ st.children do
14:         sum ← sum + 1 + CALCNUMEDGES(mst.get(child), newExclude, mst)
15:     end for
16:     return sum;
17: end function
```

Consider Algorithm 6, which calculates the number of exiting edges of a file. It is called by Algorithm 3 in line 10, via the *calcNumEdges* function. The bulk of this algorithm is present in line 14, in which a sum is calculated recursively. The sum depends on the number of exiting directed edges of the ST, as well as the number of exiting edges from its respective children. Once again, recursion is avoided by keeping track of each file visited (lines 8 and 12).

5.4 Solving Circular Dependencies Algorithm

Before we discuss this algorithm, it is important to recall the four main cases of cycles, discussed at the beginning of Chapter 4.

Include_once/require_once. For a certain path, if a cycle maker only contains functions, we may ignore the file altogether, because function information is obtained beforehand, during semantic analysis. If a cycle maker contains instructions, the file is to be included only once, i.e., future inclusions will be ignored. Thus, the taint analysis phase needs a small modification in order to stop the analysis.

Include/require. We can observe in Table 4.4 that the PHP Zend engine stops program execution when finding duplicate function definitions. On the other hand, static analysis tools simply ignore these duplicate declarations. As such, the tools could benefit from a balanced approach, in which they throw a warning whenever the cycle maker with functions is repeatedly included, while not halting the remainder of the analysis. The warning message could explain that, in a production environment, PHP will throw an error and stop program execution for such cases. As for the cycle makers with code, PHP Zend

Algorithm 7 Cycle circumvention during taint analysis.

Input: A set of paths *paths*
The set of STs *MST*
Output: None

```
1: for each path ∈ paths do
2:     TAINTANALYSIS(path, MST.get(path.name).instructions)
3: end for
4: function TAINTANALYSIS(node,instructions)
5:     for each instruction ∈ instructions do
6:         if instruction.isIncludeRequire = 1 then
7:             include ← instruction
8:             if include.name ∈ node.successors then
9:                 child ← node.successors.get(child)
10:                TAINTANALYSIS(child, MST.get(child.name).instructions)
11:            else
12:                if include.operator = "include_once" then
13:                    return
14:                else
15:                    isFunctionsOnly ← 1
16:                    for each instruction ∈ MST.get(include.name).instructions do
17:                        if instruction.isIncludeRequire = 1 then
18:                            continue
19:                        end if
20:                        if instruction.isFunctionBody = 0 then
21:                            isFunctionsOnly ← 0
22:                            break
23:                        end if
24:                    end for
25:                    if isFunctionsOnly = 0 then
26:                        REVISIT(MST.get(include.name), MST.get(include.name).children)
27:                    end if
28:                end if
29:            end if
30:        else
31:            resolve instruction
32:        end if
33:    end for
34: end function
```

engine does not offer a standard protection against infinite recursion. Static analysis tools present a tendency to treat this type of directive identically to *include_once*. Such approximation protects the tools from infinite recursion while sacrificing precision, causing false positives and false negatives. For instance, a certain instruction may not be vulnerable in a specific point of the execution, yet, it may become vulnerable depending on new entry points or dependencies that occur subsequently. Thus, it would be interesting for these tools to present alternative mechanics and heuristics.

Algorithm 7 presents the algorithm for solving circular dependencies. The algorithm receives the file inclusion paths (each represented by a root node) returned by Algorithm 1 as input. Initially, each path is analyzed independently (lines 1-3); recall that the root files remain in the MST, while include files were moved into the MIFT. Every time an include directive is found (line 6), the algorithm checks if the target is included within the list of successors (line 8). If so, the analysis may proceed as usual (lines 9 and 10). Otherwise, the target can only exist in the *cycle_maker* field of the current node. A file may only be marked as a cycle maker if it already appeared in the node chain. Every instance of *include_once* is ignored (lines 12 and 13). However, the treatment of *include* directives is more elaborate, beginning in line 14, and ending in line 24. Lines 15-24 check if the file only contains function bodies. If the file contains contents other than these function

Algorithm 8 Revisiting combinations of cycle participants. Auxiliary to Algorithm 7.

Input: A cycle maker *cyclemaker*
Cycle maker's children *cyclenodes*
Output: None

```
 1: function REVISIT(cyclemaker,cyclenodes)
 2:     for each include ∈ cyclemaker.children do
 3:         node ← CYCLENODES.GET(include.name)
 4:         if node ≠ nil then
 5:             for each instruction ∈ MST.get(node.name).instructions do
 6:                 resolve instruction
 7:             end for
 8:             for each cyclenode ∈ cyclenodes do
 9:                 if cyclenode ≠ node then
10:                     newcyclenodes ← newcyclenodes + cyclenode
11:                 end if
12:             end for
13:             REVISIT(node, newcyclenodes)
14:         end if
15:     end for
16:     return;
17: end function
```

bodies (line 20), then the file is marked for revisit (line 21). Lastly, lines 25-27 circumvent a cycle maker containing only code. Given the nodes involved in a circular dependency, Algorithm 8 is called by Algorithm 7 in line 26, via the *Revisit* function. It performs taint analysis of each combination of nodes (lines 3 and 13).

Chapter 6

Experimental Evaluation

This chapter presents a validation of the *cycle maker identification* and the *solving circular dependencies* algorithms, introduced in Chapter 5 as Algorithm 1 and Algorithm 7 respectively. We present an experimental evaluation of their implementation, written in Java and integrated into a new version of WAP (denoted *WAP++*). The evaluation is based on a set of WordPress plugins known to be vulnerable or to contain cycles.

Table 6.1 offers an overview of the results. We found that WAP 2.1 signaled 885 vulnerabilities in 224 WordPress plugins and web applications. However, the tool struggled with plugins containing circular dependencies. The number of associated cycles are present in the *Cycles* column. Observe that the first five entries in the *Plugin* column contain at least one cycle. Such characteristic results in the crash of the tool when attempting to manage the project's STs. As such, no vulnerabilities were detected for those entries by WAP 2.1 (columns 3 to 5). Our solution, in WAP++, detects these circular dependencies using the *cycle maker identification* algorithm. Section 6.1 contains Table 6.2, which presents the specific circular dependency participants detected by this algorithm. Furthermore, the *solving circular dependencies* algorithm prevents the interruption of the taint analysis, which detects 6 additional vulnerabilities, as shown by the last 3 columns of Table 6.1.

Plugin	**Cycles**	**WAP 2.1**			**WAP++**		
		SQLI	**XSS**	**RFI**	**SQLI**	**XSS**	**RFI**
tweet-old-post.3.2.5	2						
proplayer.4.7.9.1	8						
related-sites.2.1	2				**+2**		
jetpack.5.0	2					**+1**	
wordfence 6.3.10	2					**+3**	
204 web plugins	0	165	489	108			
15 web applications	0	33	68	22			
TOTAL	**16**	**198**	**557**	**130**	**+2**	**+4**	
		885 VVs			**+6 VVs**		

Table 6.1: Analysis of 224 WordPress plugins and web applications using WAP 2.1 and WAP++.

6.1 Evaluation of the Cycle Maker Identification Algorithm

Plugin	Identified circular dependencies	Analysis results
jetpack.5.0	/sal/class.json-api-platform-jetpack.php ->/sal/class.json-api-platform.php	Reflected XSS signaled in /modules/widgets/eu-cookie-law/widget.php:10, **<input type="hidden" name="redirect_url" value=" <?php echo esc_attr($_SERVER['REQUEST_URI']);?>"** False positive due to lack of knowledge of the *esc_attr* function, which encodes malicious characters.
	/_inc/lib/tracks/class.tracks-client.php ->/_inc/lib/tracks/class.tracks-client.php	
tweet-old-post.3.2.5	/tweet-old-post.php ->/top-admin.php	Correct separation between root files and dependencies. No vulnerabilities reported, however.
	/top-excludepost.php -> /tweet-old-post.php	
related-sites.2.1	/BTE_RW_core.php -> /RelatedWebsites.php	Two SQLI vulnerabilities detected. In /BTE_RW_webajax.php:46 and /BTE_RW_siteajax.php:46 **$wpdb->query($sql);** Tainted by entry point $_POST['guid'];
	/BTE_RW_admin.php -> /RelatedWebsites.php	
proplayer.4.7.9.1	/unit-tests/simpletest/errors.php ->/simpletest/invoker.php	Correct separation between root files and dependencies. No vulnerabilities reported, however.
	/unit-tests/simpletest/errors.php ->/unit-tests/simpletest/test_case.php	
	/unit-tests/simpletest/scorer.php ->/unit-tests/simpletest/invoker.php	
	/unit-tests/simpletest/default_reporter.php -> /unit-tests/simpletest/simpletest.php	
	/unit-tests/simpletest/test_case.php -> /unit-tests/simpletest/invoker.php	
	/unit-tests/simpletest/test_case.php -> /unit-tests/simpletest/scorer.php	
	/unit-tests/simpletest/test_case.php -> /unit-tests/simpletest/simpletest.php	
	/unit-tests/simpletest/exceptions.php -> /unit-tests/simpletest/invoker.php	
wordfence.6.3.10	/lib/wordfenceClass.php-> /lib/wfScanEngine.php	Three Reflected XSS vulnerabilities signaled. In /lib/diffResult.php:18, /lib/diffResult.php:24 and /lib/diffResult.php:26. False positives due to the use of *wp_kses* with empty *allowed_html* parameter, which strips away any tags. The content is printed outside of other elements, making it impossible to perform XSS via HTML events.
	/lib/wfAPI.php-> /lib/wordfenceClass.php	

Table 6.2: Plugin behaviors when processed by WAP++.

Table 6.2 presents the circular dependency participants identified in the first five plugins from Table 6.1. For each plugin, the *Identified circular dependencies* column shows unique relationships between files that participate in circular dependencies. For example, *jetpack5.0* contains two unique relationships.

We can observe that circular dependency relationships were correctly identified. This means the enhanced logic for the semantic analysis phase is working. Upon closer inspection of the files, all identified situations fit in the *include_once* cases. In particular, the cycle makers either only contain functions, or they contain functions plus some code. Example #12 from Tables 4.1 and 4.3 holds the most resemblance with the identified sit-

uations, as include directives are commonly surrounded by other constructs. We have not seen *include* situations in practice, even though they are theoretically possible.

6.2 Evaluation of the Cycle Maker Identification and Solving Circular Dependencies Algorithm

Table 6.2 also presents the effect of the *solving circular dependencies* algorithm in WAP++. The *Analysis results* column presents an overview of the analysis behavior, showing the additional detected vulnerabilities. WAP++ does not crash for any of the plugins, and signals additional vulnerabilities if present.

The XSS vulnerability in *jetpack5.0* is actually a false positive. This problem is due to WAP's (and WAP++'s) lack of knowledge of the *esc_attr* function.

The two SQLI vulnerabilities in *related-sites.2.1* are true positives. The taintedness is indirectly propagated from the entry point *$_POST['guid']*.

The three XSS vulnerabilities in *wordfence.6.3.10* are false positives due to the use of the *wp_kses* sanitization function. The function is applied to a tainted input, and uses an empty array for the *allowed_html* parameter. This means no HTML tags are allowed, and the function strips away any tags. The content is printed outside of other HTML elements, making it impossible to perform XSS via manipulation of HTML events (i.e., injection of Javascript code in the element to create or modify event handlers).

6.3 Analysis with five tools

This section presents a comparison across five source code static analysis tools: Pixy, RIPS, phpSAFE, WAP 2.1 and WAP++. The tools were evaluated using the five plugins from Table 6.2. The objective of this evaluation is to compare WAP++'s performance with the other static analysis tools (i.e., for detection of cycles and vulnerabilities) when analyzing plugins which are known to contain cycles. This task is performed in order to validate our implementation. Table 6.3 contains the detailed results for Pixy, RIPS and phpSAFE, in terms of code slices, vulnerability classes, presence of errors, and the VVs detected in the slices. Table 6.4 contains the detailed results for WAP 2.1 and WAP++.

We can observe that Pixy only detected two cycles. Outside of the *tweet-old-post.3.2.5* plugin, the tool manifests a syntax error near class definitions. Recall our assessment in Section 2.3.3, which indicates that the tool does not support the analysis of plugins with OOP features. Pixy detected 75 vulnerabilities, in which 72 are false positives. In addition, there are 19 false negatives. The high number of false positives is due to the lack of knowledge concerning the WordPress encoding and sanitization functions. The high number of false negatives is due to other implementation errors, exhibited via the aforementioned syntax error.

The RIPS tool detected 16 cycles. However, it manifests the same circular dependency problem identified in Chapter 4, Table 4.2. This erroneous behavior was identified for all plugins. RIPS detected 83 vulnerabilities, in which 64 are false positives. There are 3 false negatives. The high number of false positives is, once again, explained by the lack of knowledge concerning the WordPress encoding and sanitization functions. There is a distinct lack of SQLI vulnerabilities detected by Pixy and RIPS.

The phpSAFE tool identified 6 cycles. It exhibits symptoms akin to those identified in Chapter 4 (Table 4.2) for *related-sites.2.1*, *tweet-old-post.3.2.5* and *wordfence.6.3.10*. However, the analyses of the remaining plugins yield a different problem, in which no include dependencies are visited, likely due to a different implementation bug. The tool detected 20 vulnerabilities, in which 15 are false positives. There are 17 false negatives. The high number of false positives for SQLI and Stored XSS is due to phpSAFE's pessimistic approach regarding ambiguous inputs, such as the ones originating from a database. The high number of false negatives is likely influenced by the newly found include dependencies problem.

The WAP 2.1 tool does not detect any of the cycles. As a result, the tool crashes abruptly, as seen in Chapter 4 (Table 4.4). No vulnerabilities are reported, resulting in 22 false negatives.

Finally, the WAP++ tool solves the issues found in WAP 2.1. It identifies and solves 16 cycles without error. In addition, it identifies 6 vulnerabilities, in which 4 are false positives. The number of false negatives is reduced to 20. We found that the tool lacks knowledge of some of the more recent WordPress encoding functions, thus explaining the false positives. The false negatives are explained by the additional limitations (i.e., unrelated to circular dependencies) present in Section 3.3.

An extremely important consideration lies in the fact that Pixy and phpSAFE only perform single file analysis. This limitation is very cumbersome for complex projects and plugins, such as *jetpack5.0* and *wordfence.6.3.10*. In fact, there is a tendency for plugins to grow even bigger, as discussed in Chapter 1. Many of the reported vulnerabilities cannot be easily found by the tools, requiring additional software and even modification in order to scan entire projects. WAP++ is able to circumvent cycles and signal these additional vulnerabilities via project analysis. Another important factor lies in the fact that RIPS and phpSAFE explicitly require a web server to run, consuming many resources for the analysis of the larger plugins. WAP++ imposes no such requirement.

Plugin	Slices	Pixy										RIPS										phpSAFE									
		Error	Cycles	SQLI			Reflected XSS			Stored XSS			Error	Cycles	SQLI			Reflected XSS			Stored XSS			Error	Cycles	SQLI			Reflected XSS			Stored XSS		
				TP	FP	FN	TP	FP	FN	TP	FP	FN			TP	FP	FN	TP	FP	FN	TP	FP	FN			TP	FP	FN	TP	FP	FN	TP	FP	FN
jetpack .5.0	107	X[1]					1	65	3				X[2]	2				3	30	1		11		X[3]						2	4		5	
proplayer .4.7.9.1	23	X[1]					2		9				X[2]	8				11	11					X[3]							11			
related-sites .2.1	4	X[1]				2							X[2]	2			2							X[5]	2	2	2							
tweet-old -post.3.2.5	9		2						5				X[2]	2				5						X[5]	2				3	3	2			
wordfence .6.3.10	15	X[1]						7					X[2]	2					11			1		X[5]	2		1			2				
TOTAL	158	4	2			2	3	72	17				5	16			2	19	52	1		12		5	6	2	3		3	7	17		5	

Table 6.3: Detailed analysis results for Pixy, RIPS and phpSAFE.

Plugin	Slices	WAP 2.1										WAP++									
		Error	Cycles	SQLI			Reflected XSS			Stored XSS			Error	Cycles	SQLI			Reflected XSS			Stored XSS		
				TP	FP	FN	TP	FP	FN	TP	FP	FN			TP	FP	FN	TP	FP	FN	TP	FP	FN
jetpack .5.0	107	X[4]							4					+2					+1				
proplayer .4.7.9.1	23	X[4]							11					+8									
related-sites .2.1	4	X[4]				2								+2	+2		-2						
tweet-old -post.3.2.5	9	X[4]							5					+2									
wordfence .6.3.10	15	X[4]												+2					+3				
TOTAL	158	5				2			20					+16	+2		-2		+4				

Table 6.4: Experimental evaluation of the WAP++'s Cycle Maker Identification and Solving Circular Dependencies algorithm.

The *Error* column in Tables 6.3 and Tables 6.4indicates if a static analysis tool behaved incorrectly when processing the respective plugin. If an error is indeed present, it will be signaled with a cross (X) and a superscript with the identifier of the problem. The following erroneous behaviors were identified:

- #1. **Syntax error in Pixy** when analyzing the following files, possible due to incorrect processing of OOP constructs:
  - jetpack/sal/class.json-api-platform-jetpack.php
  - jetpack/sal/class.json-api-platform.php
  - jetpack/_inc/lib/tracks/class.tracks-client.php
  - proplayer.4.7.9.1/unit-tests/simpletest/errors.php
  - proplayer.4.7.9.1/unit-tests/simpletest/invoker.php
  - proplayer.4.7.9.1/unit-tests/simpletest/test_case.php
  - proplayer.4.7.9.1/unit-tests/simpletest/scorer.php
  - proplayer.4.7.9.1/unit-tests/simpletest/simpletest.php
  - proplayer.4.7.9.1/unit-tests/simpletest/exceptions.php
  - proplayer.4.7.9.1/unit-tests/simpletest/default_reporter.php
  - related-sites.2.1/RelatedWebsites.php
  - related-sites.2.1/BTE_RW_core.php
  - related-sites.2.1/BTE_RW_admin.php
  - wordfence/lib/wordfenceClass.php
  - wordfence/lib/wfScanEngine.php
  - wordfence/lib/wfAPI.php
- #2. **Cycle maker revisited with include_once in RIPS**. This symptom is similar to the behaviors displayed by examples #5 through #12 from Chapter 4, in Table 4.2. It occurs given the following files:
  - jetpack/sal/class.json-api-platform-jetpack.php
  - jetpack/sal/class.json-api-platform.php
  - jetpack/_inc/lib/tracks/class.tracks-client.php
  - proplayer.4.7.9.1/unit-tests/simpletest/errors.php
  - proplayer.4.7.9.1/unit-tests/simpletest/invoker.php
  - proplayer.4.7.9.1/unit-tests/simpletest/test_case.php

  - proplayer.4.7.9.1/unit-tests/simpletest/scorer.php
  - proplayer.4.7.9.1/unit-tests/simpletest/simpletest.php
  - proplayer.4.7.9.1/unit-tests/simpletest/exceptions.php
  - proplayer.4.7.9.1/unit-tests/simpletest/default_reporter.php
  - related-sites.2.1/RelatedWebsites.php
  - related-sites.2.1/BTE_RW_core.php
  - related-sites.2.1/BTE_RW_admin.php
  - tweet-old-post.3.2.5/tweet-old-post.php
  - tweet-old-post.3.2.5/top-admin.php
  - tweet-old-post.3.2.5/top-excludepost.php
  - wordfence/lib/wordfenceClass.php
  - wordfence/lib/wfScanEngine.php
  - wordfence/lib/wfAPI.php

- #3. When analyzing the following files in **phpSAFE**, **no include dependencies are found**:

  - jetpack/sal/class.json-api-platform-jetpack.php
  - jetpack/sal/class.json-api-platform.php
  - proplayer.4.7.9.1/unit-tests/simpletest/errors.php
  - proplayer.4.7.9.1/unit-tests/simpletest/invoker.php
  - proplayer.4.7.9.1/unit-tests/simpletest/test_case.php
  - proplayer.4.7.9.1/unit-tests/simpletest/scorer.php
  - proplayer.4.7.9.1/unit-tests/simpletest/simpletest.php
  - proplayer.4.7.9.1/unit-tests/simpletest/exceptions.php
  - proplayer.4.7.9.1/unit-tests/simpletest/default_reporter.php

- #4. **Crash during management of STs in WAP 2.1**. This symptom is similar to the behaviors displayed by examples #5 through #12 from Chapter 4, in Table 4.4.

- #5. **Cycle maker revisited with include_once in phpSAFE**. This symptom is similar to the behaviors displayed by examples #5 through #12 from Chapter 4, in Table 4.2. It occurs given the following files:

  - related-sites.2.1/RelatedWebsites.php
  - related-sites.2.1/BTE_RW_core.php

- related-sites.2.1/BTE_RW_admin.php
- tweet-old-post.3.2.5/tweet-old-post.php
- tweet-old-post.3.2.5/top-admin.php
- tweet-old-post.3.2.5/top-excludepost.php
- wordfence/lib/wordfenceClass.php
- wordfence/lib/wfScanEngine.php
- wordfence/lib/wfAPI.php

Chapter 7

Conclusion

In this dissertation, we discussed approaches to improve the vulnerability detection capabilities of WAP when processing PHP source code. We begun by visiting related works in the field of SCSA and documenting their vulnerability detection strengths and limitations. A body of manual analysis was compiled for such purpose, evidencing erroneous patterns in the results that need to be corrected. The most common patterns include false negatives, either due to lack of knowledge about certain types of vulnerabilities or implementation bugs. We have also seen examples of approximations that typically lead to false positives, due to the ambiguity of certain language constructs. We chose one of the identified limitation for an in-depth study. We identified different symptoms related to the processing of circular dependencies in all of the tools, and even questionable behaviors displayed by the PHP Zend engine. We understood the relevance of root files, file include paths and circular dependency points for this problem. A major solution was developed, taking these notions into account. We developed and implemented two algorithms for the identification and resolution of circular dependencies, taking into account the different root files and paths in a PHP project. These algorithms were implemented in the WAP tool, generating a new version called WAP++. An experimental evaluation was performed using 224 WordPress plugins and web applications. The static analysis tools RIPS, Pixy, phpSAFE, WAP 2.1 and WAP++ took part in this evaluation. The results show that WAP++ improves upon WAP 2.1, in both circular dependency identification and vulnerability detection. The results also exhibit a series of problems in RIPS, Pixy and phpSAFE, mostly related to file inclusion. Such problems are resolved in WAP++.

7.1 Future Work

Immediate possibilities for future work are twofold, and they are based in the tool limitation assessments made in Sections 3.3 and 3.4. Besides the processing of circular dependencies in the source code, the most prevalent problem within the chosen static analysis tools refers to the lack of conditional awareness. A single execution path can be

split in several paths (i.e., each with an associated conditional constraint) that need to be addressed to minimize the number of false negatives. These control structures and their domains can be very complex, requiring enhancements in both the semantic analysis and taint analysis phases of WAP. Another prevalent problem is the incorrect processing of concatenation operations, which may disregard parts of important instructions, such as the sensitive sinks. This problem needs to be addressed at the semantic analysis level, likely via correction and improvement of the parser.

Bibliography

[1] Bitbucket.org. Bitbucket — The Git solution for professional teams. https://bitbucket.org/, 2016.

[2] G. Cluley. Popular WordPress plugins found vulnerable to XSS attacks. https://heatsoftware.com/blog/10033/popular-wordpress-plugins-found-vulnerable-to-xss-attacks/, 2015. Accessed: 2016-12-20.

[3] Symantec Corporation. Internet Security Threat Report. Technical report, April 2016. Accessed: 2016-11-25.

[4] CUP. CUP: LALR Parser Generator in Java. http://www2.cs.tum.edu/projects/cup/, 2005. Accessed: 2017-4-24.

[5] J. Dahse. RIPS - A Static Source Code Analyser for Vulnerabilities in PHP Scripts. http://www.nds.rub.de/media/nds/attachments/files/2010/09/rips-paper.pdf, 2010. Accessed: 2016-11-1.

[6] J. Dahse and T. Holz. Simulation of Built-in PHP Features for Precise Static Code Analysis. In *Proceedings of the Symposium on Network and Distributed System Security*, 2014.

[7] N. L. de Poel. Automated Security Review of PHP Web Applications with Static Code Analysis. Master's thesis, University of Groningen, May 2010.

[8] Docs.oracle.com. StackOverflowError (Java Platform SE 7). https://docs.oracle.com/javase/7/docs/api/java/lang/StackOverflowError.html, 2016. Accessed: 2016-10-6.

[9] M. Falé, I. Medeiros, and N. Neves. Resolução de Dependências Circulares em Inclusão de Código em Análise Estática de Código. In *Proceedings of the INFORUM Symposium on Informatics*, 2017.

[10] Flex. JFlex: The Fast Scanner Generator for Java. http://jflex.de/, 2005. Accessed: 2017-4-24.

[11] Git-scm.com. Git. https://git-scm.com/, 2016.

[12] GitHub.com. JoseCarlosFonseca/phpSAFE. https://github.com/JoseCarlosFonseca/phpSAFE, 2016. Accessed: 2016-11-20.

[13] GitHub.com. oliverklee/pixy. https://github.com/oliverklee/pixy, 2016. Accessed: 2016-11-20.

[14] GitHub.com. phpSAFE/class-vulnerable-filter.php at master · JoseCarlosFonseca/phpSAFE · GitHub. https://github.com/JoseCarlosFonseca/phpSAFE/blob/master/class-vulnerable-filter.php, 2016. Accessed: 2016-11-20.

[15] GitHub.com. phpSAFE/class-vulnerable-input.php at master · JoseCarlosFonseca/phpSAFE · GitHub. https://github.com/JoseCarlosFonseca/phpSAFE/blob/master/class-vulnerable-input.php, 2016. Accessed: 2016-11-20.

[16] GitHub.com. phpSAFE/class-vulnerable-output.php at master · JoseCarlosFonseca/phpSAFE · GitHub. https://github.com/JoseCarlosFonseca/phpSAFE/blob/master/class-vulnerable-output.php, 2016. Accessed: 2016-11-20.

[17] GitHub.com. pixy/model_sqlsanit.txt at master · oliverklee/pixy · GitHub. https://github.com/oliverklee/pixy/blob/master/config/model_sqlsanit.txt#L73, 2016. Accessed: 2016-11-24.

[18] GitHub.com. pixy/model_xsssanit.txt at master · oliverklee/pixy · GitHub. https://github.com/oliverklee/pixy/blob/master/config/model_xsssanit.txt#L1, 2016. Accessed: 2016-11-24.

[19] GitHub.com. pixy/ProgramConverter.java at master · oliverklee/pixy · GitHub. https://github.com/oliverklee/pixy/blob/master/src/at/ac/tuwien/infosys/www/pixy/conversion/ProgramConverter.java#L81, 2016. Accessed: 2016-11-24.

[20] GitHub.com. pixy/sinks_file.txt at master · oliverklee/pixy · GitHub. https://github.com/oliverklee/pixy/blob/master/config/sinks_file.txt, 2016. Accessed: 2016-11-24.

[21] GitHub.com. pixy/sinks_sql.txt at master · oliverklee/pixy · GitHub. https://github.com/oliverklee/pixy/blob/master/config/sinks_sql.txt, 2016. Accessed: 2016-11-24.

[22] GitHub.com. pixy/sinks_xss.txt at master · oliverklee/pixy · GitHub. https://github.com/oliverklee/pixy/blob/master/config/sinks_xss.txt, 2016. Accessed: 2016-11-24.

[23] GitHub.com. rips-scanner/securing.php at master · robocoder/rips-scanner · GitHub. https://github.com/robocoder/rips-scanner/blob/master/config/securing.php, 2016. Accessed: 2016-11-20.

[24] GitHub.com. rips-scanner/sinks.php at master · robocoder/rips-scanner · GitHub. https://github.com/robocoder/rips-scanner/blob/master/config/sinks.php, 2016. Accessed: 2016-11-20.

[25] GitHub.com. rips-scanner/sources.php at master · robocoder/rips-scanner · GitHub. https://github.com/robocoder/rips-scanner/blob/master/config/sources.php, 2016. Accessed: 2016-11-20.

[26] GitHub.com. robocoder/rips-scanner. https://github.com/robocoder/rips-scanner, 2016. Accessed: 2016-11-20.

[27] GitHub.com. php/php-langspec: The include Operator. https://github.com/php/php-langspec/blob/master/spec/10-expressions.md#the-include-operator, 2017. Accessed: 2017-6-1.

[28] GitHub.com. php/php-langspec: The include_once Operator. https://github.com/php/php-langspec/blob/master/spec/10-expressions.md#the-include_once-operator, 2017. Accessed: 2017-6-1.

[29] G. Hoglung and G. McGraw. *Exploiting Software: The Achilles' Heel of CyberDefense*. 2004.

[30] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai. Web Application Security Assessment by Fault Injection and Behavior Monitoring. In *Proceedings of the International Conference on World Wide Web*, 2003.

[31] Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D.-T. Lee, and S.-Y. Kuo. Securing Web Application Code by Static Analysis and Runtime Protection. In *Proceedings of the 13th International Conference on World Wide Web*, 2004.

[32] Imperva. Web Application Attack Report #6. Nov. 2015.

[33] Josephkeeler.com. Joseph Keeler » PHP Security – Escape proof SQL injection in ORDER BY clause. http://josephkeeler.com/2009/05/php-security-sql-injection-in-order-by/, 2016. Accessed: 2016-11-24.

[34] N. Jovanovic, C. Kruegel, and E. Kirda. Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper). In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[35] N. Jovanovic, C. Kruegel, and E. Kirda. Precise Alias Analysis for Static Detection of Web Application Vulnerabilities. In *Proceedings of the Workshop on Programming Languages and Analysis for Security*, 2006.

[36] W. Landi. Undecidability of Static Analysis. *ACM Letters on Programming Languages and Systems*, December 1992.

[37] I. Medeiros, N. Neves, and M. Correia. Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining. *IEEE Transactions on Reliability*, March 2016.

[38] I. Medeiros, N. Neves, and M. Correia. Equipping WAP with WEAPONS to Detect Vulnerabilities: Practical Experience Report. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, June 2016.

[39] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans. Automatically Hardening Web Applications using Precise Tainting. In *Proceedings of the Security and Privacy in the Age of Ubiquitous Computing*, 2005.

[40] Notsosecure.com. Injection in Order by, Group by Clause - NotSoSecure. `https://www.notsosecure.com/injection-in-order-by-clause/`, 2016. Accessed: 2016-11-24.

[41] P. Nunes, J. Fonseca, and M. Vieira. phpSAFE: A Security Analysis Tool for OOP Web Application Plugins. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015.

[42] T. O'Reilly. What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. MPRA Paper 4578, University Library of Munich, Germany, March 2007.

[43] OWASP.org. OWASP Backend Security Project DBMS Fingerprint. `https://www.owasp.org/index.php/OWASP_Backend_Security_Project_DBMS_Fingerprint`, 2008. Accessed: 2016-12-20.

[44] OWASP.org. Cross-site Scripting (XSS). `https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)`, 2016. Accessed: 2016-12-20.

[45] P. Paganini. Kapustkiy hacked the National Assembly of Ecuador website. `http://securityaffairs.co/wordpress/54068/data-breach/`

national-assembly-of-ecuador-hacked.html, 2016. Accessed: 2016-12-20.

[46] PHP.net. PHP: addslashes - Manual. https://secure.php.net/manual/en/function.addslashes.php, 2016. Accessed: 2016-10-21.

[47] PHP.net. PHP: Function Reference - Manual. https://secure.php.net/manual/en/funcref.php, 2016. Accessed: 2016-12-1.

[48] PHP.net. PHP: $_GET - Manual. https://secure.php.net/manual/en/reserved.variables.get.php, 2016. Accessed: 2016-10-21.

[49] PHP.net. PHP: Installation and Configuration - Manual. https://secure.php.net/manual/en/install.php, 2016. Accessed: 2016-12-1.

[50] PHP.net. PHP: mysqli::real_escape_string - Manual. https://secure.php.net/manual/en/mysqli.real-escape-string.php, 2016. Accessed: 2016-10-21.

[51] PHP.net. PHP: mysql_query - Manual. https://secure.php.net/manual/en/function.mysql-query.php, 2016. Accessed: 2016-10-21.

[52] PHP.net. PHP: mysql_real_escape_string - Manual. https://secure.php.net/manual/en/function.mysql-real-escape-string.php, 2016. Accessed: 2016-10-21.

[53] PHP.net. PHP: PHP Manual - Manual. https://secure.php.net/manual/en/index.php, 2016. Accessed: 2016-12-1.

[54] PHP.net. PHP: str_replace - Manual. https://secure.php.net/manual/en/function.str-replace.php, 2016. Accessed: 2016-11-10.

[55] PHP.net. PHP: include - Manual. https://secure.php.net/manual/en/function.include.php, 2017. Accessed: 2017-6-1.

[56] PHP.net. PHP: include_once - Manual. https://secure.php.net/manual/en/function.include-once.php, 2017. Accessed: 2017-6-1.

[57] PHP.net. PHP: token_get_all - Manual. https://secure.php.net/manual/en/function.token-get-all.php, 2017. Accessed: 2017-6-1.

[58] PHP.net. PHP: Tokenizer - Manual. https://secure.php.net/manual/en/book.tokenizer.php, 2017. Accessed: 2017-6-1.

[59] PHP.net. PHP: User-defined functions - Manual. https://secure.php.net/manual/en/functions.user-defined.php, 2017. Accessed: 2017-6-1.

[60] PHP.net. PHP: What References Are - Manual. https://secure.php.net/manual/en/language.references.whatare.php, 2017. Accessed: 2017-6-1.

[61] T. Pietraszek and C. Berghe. Defending against Injection Attacks through Context-Sensitive String Evaluation. In *Proceedings of the Recent Advances in Intrusion Detection*, 2005.

[62] C. R. Prause, R. Reiners, and S. Dencheva. Empirical Study of Tool Support in Highly Distributed Research Projects. In *Proceedings of the IEEE International Conference on Global Software Engineering*, Aug 2010.

[63] A. Sabelfeld and A. C. Myers. Language-based Information-flow Security. *IEEE Journal on Selected Areas in Communications*, September 2006.

[64] R. Sandhu. Lattice-Based Access Control Models. *Computer*, November 1993.

[65] Sourceforge.net. Web Application Protection. https://sourceforge.net/projects/awap/?source=navbar, 2016. Accessed: 2016-11-20.

[66] Stackoverflow.com. Addslashes safe to protect against xss in array? http://stackoverflow.com/a/10538721, 2016. Accessed: 2016-11-24.

[67] F. Valeur, D. Mutz, and G. Vigna. A Learning-based Approach to the Detection of SQL Attacks. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2005.

[68] Virtuesecurity.com. Preventing Cross-site Scripting in PHP - Virtue Security. https://www.virtuesecurity.com/blog/preventing-cross-site-scripting-php/, 2016. Accessed: 2016-11-24.

[69] W3schools.com. PHP stripcslashes() Function. http://www.w3schools.com/php/func_string_stripcslashes.asp, 2016. Accessed: 2016-10-24.

[70] W3techs. Historical trends in the usage of server-side programming languages, December 2016. https://w3techs.com/technologies/history_overview/programming_language, 2016. Accessed: 2016-12-1.

[71] W3techs. Usage Statistics and Market Share of Content Management Systems for Websites, December 2016. https://w3techs.com/technologies/overview/content_management/all, 2016. Accessed: 2016-12-1.

[72] D. Wichers. [Owasp-topten] [Owasp-leaders] OWASP Top Ten: Project Activity? http://lists.owasp.org/pipermail/owasp-topten/2015-June/001310.html, 2015. Accessed: 2016-12-1.

[73] B.A. Wichmann, A.A. Canning, D.W.R. Marsh, D.L. Clutterbuck, L.A. Winsborrow, and N.J. Ward. Industrial Perspective on Static Analysis. *Software Engineering Journal*, March 1995.

[74] J. Williams and D. Wichers. OWASP Top 10 - 2013 rcl - The Ten Most Critical Web Application Security Risks. Technical report, 2013. Accessed: 2016-11-25.

[75] WordPress.org. PHP Warning: Missing argument 2 for wpdb::prepare(). https://make.wordpress.org/core/2012/12/12/php-warning-missing-argument-2-for-wpdb-prepare/, 2012. Accessed: 2016-11-14.

[76] WordPress.org. About » Requirements — WordPress. https://wordpress.org/about/requirements/, 2016. Accessed: 2016-12-20.

[77] WordPress.org. Data Validation « WordPress Codex. https://codex.wordpress.org/Data_Validation#Database, 2016. Accessed: 2016-10-24.

[78] WordPress.org. Data Validation « WordPress Codex. https://codex.wordpress.org/Data_Validation#HTML.2FXML, 2016. Accessed: 2016-11-24.

[79] WordPress.org. Database Description « WordPress Codex. https://codex.wordpress.org/Database_Description#Table:_wp_options, 2016. Accessed: 2016-10-24.

[80] WordPress.org. Editing wp-config.php « WordPress Codex. https://codex.wordpress.org/Editing_wp-config.php#table_prefix, 2016. Accessed: 2016-10-24.

[81] WordPress.org. esc_attr() — Function — WordPress Developer Resources. https://developer.wordpress.org/reference/functions/esc_attr/#more-information, 2016. Accessed: 2016-10-24.

[82] WordPress.org. get_post() — Function — WordPress Developer Resources. https://developer.wordpress.org/reference/functions/get_post/, 2016. Accessed: 2016-10-24.

[83] WordPress.org. How to fix the intentionally vulnerable plugin. https://make.wordpress.org/support/handbook/breakfix-lessons/how-to-fix-the-intentionally-vulnerable-plugin/#sql-injection, 2016. Accessed: 2016-10-24.

[84] WordPress.org. Options API « WordPress Codex. https://codex.wordpress.org/Options_API, 2016. Accessed: 2016-10-24.

[85] WordPress.org. wp-db.php in tags/4.6/src/wp-includes – WordPress Trac. https://core.trac.wordpress.org/browser/tags/4.6/src/wp-includes/wp-db.php#L1941, 2016. Accessed: 2016-10-24.

[86] WordPress.org. wpdb::_real_escape() — Method — WordPress Developer Resources. https://developer.wordpress.org/reference/classes/wpdb/_real_escape/, 2016. Accessed: 2016-10-24.

Appendix A

Explaining the Manual Analysis Format

The manual analysis format contains the following items:

- **Sensitive sink** contains the name of the sink (i.e. *mysql_query*) and its location.
- **Entry points** is a table that contains the names, types and locations of entry points. The *Reaches Sink Directly?* column, *RSD*, indicates whether the input reaches the sensitive sink directly (i.e., without being altered, without creating dependencies). The manual analysis process has shown that most developers simply use native PHP entry points, such as *$_GET*.
- **Entry point dependencies** is a table that contains variables to which taintedness was propagated to, under certain conditions. The *True/False?* column, *T/F*, indicates if the taint condition is true or false. The *Nested?* column, *Ntd*, indicates if the variable is being updated since the previous instruction. Once again, *RSD* indicates whether the input reaches the sensitive sink directly (i.e., without being altered, without creating dependencies).
- **Execution path constraints** contains the control flow context for the slice, indicating whether it contains the sensitive sink, whether the constraint is contained (”nested”) inside another constraint, and the ”true/false” value that should be met under the desired control flow context. The *Nested?* (*Ntd*) and *True/False?* (*T/F*) columns are important because control flow contexts can stack up on one another, affecting the direction of the input until it reaches a sink. The *Encloses sink?* column, *ES*, indicates if the constraint encloses the sensitive sink.
- **Entry point validation constraints** contains validations performed against the entry points or dependencies. The *True/False?* column, *T/F*, indicates the value that should be met for the desired control flow context. The *Encloses sink?* column, *ES*, indicates if the constraint encloses the sensitive sink.
- **Slice** contains the relevant trace of code, encompassing the entry points, dependencies, constraints, validations, and, finally, the sink.

Appendix B

Plugin: ajaxgallery.3.0

B.1 Type: SQLI

B.1.1 Slice #1

B.1.1.1 Verdict: 1 (vulnerable)

wap SQLI Slice #3
phpSAFE SQLI Slice #2

Sensitive sink

- Name: $wpdb->query
- Line: 31
- File: ajaxgallery.3.0/utils/list.php

Entry points:

L	Type	Name	RSD	File
31	GET	$_GET['gId']	1	ajaxgallery.3.0/utils/list.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
30	isset($_GET['gId'])	1		1	ajaxgallery.3.0/utils/list.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
30	isset($_GET[‘delete’]) && isset($_GET[’gId’])	1	0	1	ajaxgallery.3.0/utils/list.php

Slice:

L	Instruction	File
30	if(isset($_GET[’delete’]) && isset($_GET[’gId’])){	ajaxgallery.3.0/utils/list.php
31	$wpdb->query(”DELETE FROM $wpdb->options WHERE option_name=’agItem’ and option_id=”.$_GET[’gId’]);	ajaxgallery.3.0/utils/list.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE’s decision: 1
WAP’s decision: 1

B.1.1.2 Explanation

A vulnerability occurs because the entry point (*$_GET[’gId’]*) reaches a sensitive sink (*$wpdb->query*) directly. *$wpdb->query* is not known to sanitize its input [77]. *$wpdb->options* holds the name of the WordPress options table [84][79].

B.1.2 Slice #2

B.1.2.1 Verdict: 1 (vulnerable)

wap SQLI Slice #2
phpSAFE SQLI Slice #3

Sensitive sink

- Name: $wpdb->query
- Line: 55
- File: ajaxgallery.3.0/utils/options.php

Entry points:

L	Type	Name	RSD	File
42	POST	$_POST[’user’]	0	ajaxgallery.3.0/utils/options.php
43	POST	$_POST[’album’]	0	ajaxgallery.3.0/utils/options.php
44	POST	$_POST[’rows’]	0	ajaxgallery.3.0/utils/options.php
45	POST	$_POST[’cols’]	0	ajaxgallery.3.0/utils/options.php
46	POST	$_POST[’align’]	0	ajaxgallery.3.0/utils/options.php
47	POST	$_POST[’pag’]	0	ajaxgallery.3.0/utils/options.php
48	POST	$_POST[’showp’]	0	ajaxgallery.3.0/utils/options.php
49	POST	$_POST[’thumbsize’]	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
42	variable	$value	$_POST['user']	isset($_POST['user']) && trim($_POST['user']) != ""	1	0	0	ajaxgallery.3.0/ utils/options.php
43	variable	$value	filterAlbum ($_POST['album'])	isset($_POST['album']) && trim($_POST['album']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
44	variable	$value	$_POST['rows']	isset($_POST['rows']) && trim($_POST['rows']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
45	variable	$value	$_POST['cols']	isset($_POST['cols']) && trim($_POST['cols']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
46	variable	$value	$_POST['align']	isset($_POST['align']) && trim($_POST['align']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
47	variable	$value	$_POST['pag']	isset($_POST['pag']) && trim($_POST['pag']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
48	variable	$value	$_POST['showp']	isset($_POST['showp']) && trim($_POST['showp']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
49	variable	$value	$_POST['thumbsize']	isset($_POST['thumbsize']) && trim($_POST['thumbsize']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
53	variable	$sql	$value	1	1	0	1	ajaxgallery.3.0/ utils/options.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
50	isset($_POST['gId'])	0	39	0	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
39	isset($_POST['save'])	1	0	1	ajaxgallery.3.0/utils/options.php
50	isset($_POST['gId'])	0	1	0	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
32	function filterAlbum($str){	ajaxgallery.3.0/ utils/options.php
33	$str = str_replace(” ”, ””, $str);	ajaxgallery.3.0/ utils/options.php
34	$str = str_replace(”.”, ””, $str);	ajaxgallery.3.0/ utils/options.php
35	$str = str_replace(”,”, ””, $str);	ajaxgallery.3.0/ utils/options.php
36	return $str;	ajaxgallery.3.0/ utils/options.php
37	}	ajaxgallery.3.0/ utils/options.php
39	if(isset($_POST[’save’])){	ajaxgallery.3.0/ utils/options.php
40	$sql = ””;	ajaxgallery.3.0/ utils/options.php
42	$value = (isset($_POST[’user’]) && trim($_POST[’user’]) != ””? $_POST[’user’] : ”user”) . ”:”;	ajaxgallery.3.0/ utils/options.php
43	$value .= (isset($_POST[’album’]) && trim($_POST[’album’]) != ”” ? filterAlbum($_POST[’album’]) : ””) . ”:”;	ajaxgallery.3.0/ utils/options.php
44	$value .= (isset($_POST[’rows’]) && trim($_POST[’rows’]) != ”” ? $_POST[’rows’] : ”4”) . ”:”;	ajaxgallery.3.0/ utils/options.php
45	$value .= (isset($_POST[’cols’]) && trim($_POST[’cols’]) != ”” ? $_POST[’cols’] : ”4”) . ”:”;	ajaxgallery.3.0/ utils/options.php
46	$value .= (isset($_POST[’align’]) && trim($_POST[’align’]) != ”” ? $_POST[’align’] : ”center”) . ”:”;	ajaxgallery.3.0/ utils/options.php
47	$value .= (isset($_POST[’pag’]) && trim($_POST[’pag’]) != ”” ? $_POST[’pag’] : ”0”) . ”:”;	ajaxgallery.3.0/ utils/options.php
48	$value .= (isset($_POST[’showp’]) && trim($_POST[’showp’]) != ”” ? $_POST[’showp’] : ”true”) . ”:”;	ajaxgallery.3.0/ utils/options.php
49	$value .= (isset($_POST[’thumbsize’]) && trim($_POST[’thumbsize’]) != ”” ? $_POST[’thumbsize’] : ”160”);	ajaxgallery.3.0/ utils/options.php
52	}else{	ajaxgallery.3.0/ utils/options.php
53	$sql = ”INSERT INTO $wpdb->options (option_name, option_value) VALUES(’agItem’,’$value’)”;	ajaxgallery.3.0/ utils/options.php
54	}	ajaxgallery.3.0/ utils/options.php
55	$wpdb->query($sql)	ajaxgallery.3.0/ utils/options.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE’s decision: 1
WAP’s decision: 1

B.1.2.2 Explanation

A vulnerability occurs because the entry point(s) participate in a series of concatenations to the variable *$value* which, in turn, participates in an expression that is assigned to *$sql*. Finally, *$sql* reaches a sensitive sink (*$wpdb->query*) unsanitized. For both tools, only one vulnerability was reported concerning this sink, but consider the following:

- *$sql* may come from the *else* branch, in which case it requires *$value*, and performs an *INSERT* (as seen in this slice);

- *$sql* may come from the *if* branch, in which case it requires both *$_POST[‘gId’]* and *$value*, and performs an *UPDATE* (see Slice #3 in B.1.3).

Re-running wap-2.1 with this file will result in only one vulnerability being detected. **It only considers the else branch in both the taint analysis and code correction phases.** Please check the end of this plugin (B.5) for more examples.

B.1.3 Slice #3

B.1.3.1 Verdict: 1 (vulnerable)

Sensitive sink

- Name: $wpdb->query
- Line: 55
- File: ajaxgallery.3.0/utils/options.php

Entry points:

L	Type	Name	RSD	File
42	POST	$_POST[’user’]	0	ajaxgallery.3.0/utils/options.php
43	POST	$_POST[’album’]	0	ajaxgallery.3.0/utils/options.php
44	POST	$_POST[’rows’]	0	ajaxgallery.3.0/utils/options.php
45	POST	$_POST[’cols’]	0	ajaxgallery.3.0/utils/options.php
46	POST	$_POST[’align’]	0	ajaxgallery.3.0/utils/options.php
47	POST	$_POST[’pag’]	0	ajaxgallery.3.0/utils/options.php
48	POST	$_POST[’showp’]	0	ajaxgallery.3.0/utils/options.php
49	POST	$_POST[’thumbsize’]	0	ajaxgallery.3.0/utils/options.php
51	POST	$_POST[‘gId’]	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
42	variable	$value	$_POST[’user’]	isset($_POST[’user’]) && trim($_POST[’user’]) != “”	1	0	0	ajaxgallery.3.0/ utils/options.php
43	variable	$value	filterAlbum ($_POST[’album’])	isset($_POST[’album’]) && trim($_POST[’album’]) != “”	1	1	0	ajaxgallery.3.0/ utils/options.php
44	variable	$value	$_POST[’rows’]	isset($_POST[’rows’]) && trim($_POST[’rows’]) != “”	1	1	0	ajaxgallery.3.0/ utils/options.php
45	variable	$value	$_POST[’cols’]	isset($_POST[’cols’]) && trim($_POST[’cols’]) != “”	1	1	0	ajaxgallery.3.0/ utils/options.php
46	variable	$value	$_POST[’align’]	isset($_POST[’align’]) && trim($_POST[’align’]) != “”	1	1	0	ajaxgallery.3.0/ utils/ options.php
47	variable	$value	$_POST[’pag’]	isset($_POST[’pag’]) && trim($_POST[’pag’]) != “”	1	1	0	ajaxgallery.3.0/ utils/options.php
48	variable	$value	$_POST[’showp’]	isset($_POST[’showp’]) && trim($_POST[’showp’]) != “”	1	1	0	ajaxgallery.3.0/ utils/options.php
49	variable	$value	$_POST[’thumbsize’]	isset($_POST[’thumbsize’]) && trim($_POST[’thumbsize’]) != “”	1	1	0	ajaxgallery.3.0/ utils/options.php
51	variable	$sql	$value, $_POST[‘gId’]	1	1	0	1	ajaxgallery.3.0/ utils/options.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
50	isset($_POST[‘gId’])	0	39	1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
39	isset($_POST[‘save’])	1	0	1	ajaxgallery.3.0/utils/options.php
50	isset($_POST[‘gId’])	0	1	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
32	function filterAlbum($str){	ajaxgallery.3.0/utils/options.php
33	$str = str_replace(” ”, ””, $str);	ajaxgallery.3.0/utils/options.php
34	$str = str_replace(”.”, ””, $str);	ajaxgallery.3.0/utils/options.php
35	$str = str_replace(”,”, ””, $str);	ajaxgallery.3.0/utils/options.php
36	return $str;	ajaxgallery.3.0/utils/options.php
37	}	ajaxgallery.3.0/utils/options.php
39	if(isset($_POST[’save’])){	ajaxgallery.3.0/utils/options.php
40	$sql = ””;	ajaxgallery.3.0/utils/options.php
42	$value = (isset($_POST[’user’]) && trim($_POST[’user’]) != ””? $_POST[’user’] : ”user”) . ”:”;	ajaxgallery.3.0/utils/options.php
43	$value .= (isset($_POST[’album’]) && trim($_POST[’album’]) != ”” ? filterAlbum($_POST[’album’]) : ””) . ”:”;	ajaxgallery.3.0/utils/options.php
44	$value .= (isset($_POST[’rows’]) && trim($_POST[’rows’]) != ”” ? $_POST[’rows’] : ”4”) . ”:”;	ajaxgallery.3.0/utils/options.php
45	$value .= (isset($_POST[’cols’]) && trim($_POST[’cols’]) != ”” ? $_POST[’cols’] : ”4”) . ”:”;	ajaxgallery.3.0/utils/options.php
46	$value .= (isset($_POST[’align’]) && trim($_POST[’align’]) != ”” ? $_POST[’align’] : ”center”) . ”:”;	ajaxgallery.3.0/utils/options.php
47	$value .= (isset($_POST[’pag’]) && trim($_POST[’pag’]) != ”” ? $_POST[’pag’] : ”0”) . ”:”;	ajaxgallery.3.0/utils/options.php
48	$value .= (isset($_POST[’showp’]) && trim($_POST[’showp’]) != ”” ? $_POST[’showp’] : ”true”) . ”:”;	ajaxgallery.3.0/utils/options.php
49	$value .= (isset($_POST[’thumbsize’]) && trim($_POST[’thumbsize’]) != ”” ? $_POST[’thumbsize’] : ”160”);	ajaxgallery.3.0/utils/options.php
50	if(isset($_POST[’gId’])){	ajaxgallery.3.0/utils/options.php
51	$sql = ”UPDATE $wpdb->options SET option_value=’$value’ WHERE option_name=’agItem’ and option_id=”.$_POST[’gId’];	ajaxgallery.3.0/utils/options.php
52	}else{	ajaxgallery.3.0/utils/options.php
55	$wpdb->query($sql);	ajaxgallery.3.0/utils/options.php

Name of the enclosing function: No function

B.1.3.2 Explanation

A vulnerability occurs because *$value* and *$_POST[‘gId’]* participate in an expression that is assigned to a variable (*$sql*) which, in turn, reaches a sensitive sink (*$wpdb->query*). **False negative for all tools**.

B.1.4 Slice #4

B.1.4.1 Verdict: 0 (not vulnerable)

wap SQLI Slice #4

Sensitive sink

- Name: $wpdb->get_results
- Line: 54
- File: ajaxgallery.3.0/utils/list.php

Slice:

L	Instruction	File
54	$gItems = $wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	ajaxgallery.3.0/utils/list.php

Name of the enclosing function: No function

B.1.4.2 Explanation

This slice was produced by mistake.

B.2 Type: Reflected XSS

B.2.1 Slice #5

B.2.1.1 Verdict: 1 (vulnerable)

wap XSS Slice #2
pixy XSS Slice #2

Sensitive sink

- Name: echo
- Line: 164
- File: ajaxgallery.3.0/utils/options.php

Entry points:

L	Type	Name	RSD	File
160	GET	$_GET['gId']	0	ajaxgallery.3.0/utils/options.php
160	POST	$_POST['gId']	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
160	variable	$ggId	$_GET[‘gId’]	isset($_GET[‘gId’])	1	0	1	ajaxgallery.3.0/utils/options.php
160	variable	$ggId	$_POST[’gId’]	isset($_GET[‘gId’])	0	0	1	ajaxgallery.3.0/utils/options.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
159	isset($_GET[‘gId’]) || isset($_POST[’gId’])	1		1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
159	isset($_GET[‘gId’]) || isset($_POST[’gId’])	1	0	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
159	if(isset($_GET[’gId’]) || isset($_POST[’gId’])){	ajaxgallery.3.0/ utils/options.php
160	$ggId = isset($_GET[’gId’]) ? $_GET[’gId’] : $_POST[’gId’];	ajaxgallery.3.0/ utils/options.php
164	<textarea>[ajax_gallery id=<?php echo $ggId;?>]</textarea> 	ajaxgallery.3.0/ utils/options.php

Name of the enclosing function: No function

phpSAFE’s decision: 1
WAP’s decision: 1
Pixy’s decision: 1
RIPS’s decision: 1

B.2.1.2 Explanation

A vulnerability occurs because the entry point (*$_GET[’gId’]* or *$_POST[’gId’]*) is embedded in an expression that is assigned to a variable (*$ggId*) which, in turn, reaches a sensitive sink (*echo*). It is possible to insert and run malicious Javascript code in this *textarea*.

B.2.2 Slice #6

B.2.2.1 Verdict: 1 (vulnerable)

pixy XSS Slice #3
wap XSS Slice #3

Sensitive sink

- Name: echo

- Line: 169
- File: ajaxgallery.3.0/utils/options.php

Entry points:

L	Type	Name	RSD	File
160	GET	$_GET['gId']	0	ajaxgallery.3.0/utils/options.php
160	POST	$_POST['gId']	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
160	variable	$ggId	$_GET['gId']	isset($_GET['gId'])	1	0	1	ajaxgallery.3.0/utils/options.php
160	variable	$ggId	$_POST['gId']	isset($_GET['gId'])	0	0	1	ajaxgallery.3.0/utils/options.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
159	isset($_GET['gId']) || isset($_POST['gId'])	1		1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
159	isset($_GET['gId']) || isset($_POST['gId'])	1	0	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
159	if(isset($_GET['gId']) || isset($_POST['gId'])){	ajaxgallery.3.0/ utils/options.php
160	$ggId = isset($_GET['gId']) ? $_GET['gId'] : $_POST['gId'];	ajaxgallery.3.0/ utils/options.php
169	<input name="gId" type="hidden" value="<?php echo $ggId;?>" />	ajaxgallery.3.0/ utils/options.php

Name of the enclosing function: No function
phpSAFE's decision: 1
WAP's decision: 1
Pixy's decision: 1
RIPS's decision: 1

B.2.2.2 Explanation

A vulnerability occurs because the entry point (*$_GET['gId']* or *$_POST['gId']*) is embedded in an expression that is assigned to a variable (*$ggId*) which, in turn, reaches a sensitive sink (*echo*). It is possible to insert and run malicious Javascript code after the <*input/*> element.

B.3 Type: Stored XSS

B.3.1 Slices #7, #8, #9, #10

B.3.1.1 Verdict: 1 (vulnerable)

Sensitive sink

- Name: echo
- Line: 60,61,62,63
- File: ajaxgallery.3.0/utils/list.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
42	POST	$_POST['user']	0	ajaxgallery.3.0/utils/options.php
43	POST	$_POST['album']	0	ajaxgallery.3.0/utils/options.php
44	POST	$_POST['rows']	0	ajaxgallery.3.0/utils/options.php
45	POST	$_POST['cols']	0	ajaxgallery.3.0/utils/options.php
46	POST	$_POST['align']	0	ajaxgallery.3.0/utils/options.php
47	POST	$_POST['pag']	0	ajaxgallery.3.0/utils/options.php
48	POST	$_POST['showp']	0	ajaxgallery.3.0/utils/options.php
49	POST	$_POST['thumbsize']	0	ajaxgallery.3.0/utils/options.php
51	POST	$_POST['gId']	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
42	variable	$value	$_POST['user']	isset($_POST['user']) && trim($_POST['user']) != ""	1	0	0	ajaxgallery.3.0/ utils/options.php
43	variable	$value	filterAlbum ($_POST['album'])	isset($_POST['album']) && trim($_POST['album']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
44	variable	$value	$_POST['rows']	isset($_POST['rows']) && trim($_POST['rows']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
45	variable	$value	$_POST['cols']	isset($_POST['cols']) && trim($_POST['cols']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
46	variable	$value	$_POST['align']	isset($_POST['align']) && trim($_POST['align']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
47	variable	$value	$_POST['pag']	isset($_POST['pag']) && trim($_POST['pag']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
48	variable	$value	$_POST['showp']	isset($_POST['showp']) && trim($_POST['showp']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
49	variable	$value	$_POST['thumbsize']	isset($_POST['thumbsize']) && trim($_POST['thumbsize']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
51	variable	$sql	$value, $_POST['gId']	1	1	0	1	ajaxgallery.3.0/ utils/options.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
50	isset($_POST[‘gId’])	0	39	1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
39	isset($_POST[‘save’])	1	0	1	ajaxgallery.3.0/utils/options.php
50	isset($_POST[‘gId’])	0	1	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
32	function filterAlbum($str){	ajaxgallery.3.0/utils/options.php
33	$str = str_replace(” ”, ””, $str);	ajaxgallery.3.0/utils/options.php
34	$str = str_replace(”.”, ””, $str);	ajaxgallery.3.0/utils/options.php
35	$str = str_replace(”,”, ””, $str);	ajaxgallery.3.0/utils/options.php
36	return $str;	ajaxgallery.3.0/utils/options.php
37	}	ajaxgallery.3.0/utils/options.php
39	if(isset($_POST[’save’])){	ajaxgallery.3.0/utils/options.php
40	$sql = ””;	ajaxgallery.3.0/utils/options.php
42	$value = (isset($_POST[’user’]) && trim($_POST[’user’]) != ””? $_POST[’user’] : ”user”) . ”:”;	ajaxgallery.3.0/utils/options.php
43	$value .= (isset($_POST[’album’]) && trim($_POST[’album’]) != ”” ? filterAlbum($_POST[’album’]) : ””) . ”:”;	ajaxgallery.3.0/utils/options.php
44	$value .= (isset($_POST[’rows’]) && trim($_POST[’rows’]) != ”” ? $_POST[’rows’] : ”4”) . ”:”;	ajaxgallery.3.0/utils/options.php
45	$value .= (isset($_POST[’cols’]) && trim($_POST[’cols’]) != ”” ? $_POST[’cols’] : ”4”) . ”:”;	ajaxgallery.3.0/utils/options.php
46	$value .= (isset($_POST[’align’]) && trim($_POST[’align’]) != ”” ? $_POST[’align’] : ”center”) . ”:”;	ajaxgallery.3.0/utils/options.php
47	$value .= (isset($_POST[’pag’]) && trim($_POST[’pag’]) != ”” ? $_POST[’pag’] : ”0”) . ”:”;	ajaxgallery.3.0/utils/options.php
48	$value .= (isset($_POST[’showp’]) && trim($_POST[’showp’]) != ”” ? $_POST[’showp’] : ”true”) . ”:”;	ajaxgallery.3.0/utils/options.php
49	$value .= (isset($_POST[’thumbsize’]) && trim($_POST[’thumbsize’]) != ”” ? $_POST[’thumbsize’] : ”160”);	ajaxgallery.3.0/utils/options.php
50	if(isset($_POST[’gId’])){	ajaxgallery.3.0/utils/options.php
51	$sql = ”UPDATE $wpdb->options SET option_value=’$value’ WHERE option_name=’agItem’ and option_id=”.$_POST[’gId’];	ajaxgallery.3.0/utils/options.php
52	}else{	ajaxgallery.3.0/utils/options.php
55	$wpdb->query($sql);	ajaxgallery.3.0/utils/options.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
54	BD	$wpdb->get_results(“SELECT * FROM $wpdb->options WHERE option_name=’agItem’”);	0	ajaxgallery.3.0/utils/list.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
54	variable	$gItems	$wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	0	1	0	0	ajaxgallery.3.0/ utils/list.php
55	variable	$gItem	$gItems	0	1	0	0	ajaxgallery.3.0/ utils/list.php
56	variable	$gItem->option_value	$gItem	0	1	0	0	ajaxgallery.3.0/ utils/list.php
56	variable	$gItem->option_value	$gItem->option_value	1	1	0	0	ajaxgallery.3.0/ utils/list.php
56	variable	$options	$gItem->option_value	1	1	0	1	ajaxgallery.3.0/ utils/list.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
55	foreach ($gItems as $gItem)	1		1	ajaxgallery.3.0/utils/list.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
55	foreach ($gItems as $gItem)	1	0	1	ajaxgallery.3.0/utils/list.php

Slice:

L	Instruction	File
54	$gItems = $wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	ajaxgallery.3.0/utils/list.php
55	foreach ($gItems as $gItem){	ajaxgallery.3.0/utils/list.php
56	$options = split(":", $gItem->option_value);	ajaxgallery.3.0/utils/list.php
60	<td><?php echo $options[0]; ?></td>	ajaxgallery.3.0/utils/list.php
61	<td><?php echo $options[1]; ?></td>	ajaxgallery.3.0/utils/list.php
62	<td><?php echo $options[2]; ?></td>	ajaxgallery.3.0/utils/list.php
63	<td><?php echo $options[3]; ?></td>	ajaxgallery.3.0/utils/list.php

Name of the enclosing function: No function

B.3.1.2 Explanation

The *UPDATE* command is vulnerable due to the use of a tainted input, *$value*, which is assigned to the *option_value* field in rows where *option_name* has the value *'agItem'*. The *SELECT* command retrieves all fields where *option_name* is 'agItem' (from the same table), including *option_value*. This value's taintedness would then be propagated to the *$options array* (since the contents were stored in colon-separated fashion). Finally, *option*'s elements are argument to *echo*, the sink of this vulnerability. **False negative for all tools**.

B.3.2 Slices #11, #12, #13, #14, #15, #16

B.3.2.1 Verdict: 1 (vulnerable)

Sensitive sink

- Name: echo
- Line: 93,100,111,118,145,151
- File: ajaxgallery.3.0/utils/options.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
42	POST	$_POST['user']	0	ajaxgallery.3.0/utils/options.php
43	POST	$_POST['album']	0	ajaxgallery.3.0/utils/options.php
44	POST	$_POST['rows']	0	ajaxgallery.3.0/utils/options.php
45	POST	$_POST['cols']	0	ajaxgallery.3.0/utils/options.php
46	POST	$_POST['align']	0	ajaxgallery.3.0/utils/options.php
47	POST	$_POST['pag']	0	ajaxgallery.3.0/utils/options.php
48	POST	$_POST['showp']	0	ajaxgallery.3.0/utils/options.php
49	POST	$_POST['thumbsize']	0	ajaxgallery.3.0/utils/options.php
51	POST	$_POST['gId']	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
42	variable	$value	$_POST['user']	isset($_POST['user']) && trim($_POST['user']) != ""	1	0	0	ajaxgallery.3.0/ utils/options.php
43	variable	$value	filterAlbum ($_POST['album'])	isset($_POST['album']) && trim($_POST['album']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
44	variable	$value	$_POST['rows']	isset($_POST['rows']) && trim($_POST['rows']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
45	variable	$value	$_POST['cols']	isset($_POST['cols']) && trim($_POST['cols']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
46	variable	$value	$_POST['align']	isset($_POST['align']) && trim($_POST['align']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
47	variable	$value	$_POST['pag']	isset($_POST['pag']) && trim($_POST['pag']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
48	variable	$value	$_POST['showp']	isset($_POST['showp']) && trim($_POST['showp']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
49	variable	$value	$_POST['thumbsize']	isset($_POST['thumbsize']) && trim($_POST['thumbsize']) != ""	1	1	0	ajaxgallery.3.0/ utils/options.php
51	variable	$sql	$value, $_POST['gId']	1	1	0	1	ajaxgallery.3.0/ utils/options.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
50	isset($_POST['gId'])	0	39	1	ajaxgallery.3.0/utils/options.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
39	isset($_POST[‘save’])	1	0	1	ajaxgallery.3.0/utils/options.php
50	isset($_POST[‘gId’])	0	1	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
32	function filterAlbum($str){	ajaxgallery.3.0/utils/options.php
33	$str = str_replace(” ”, ””, $str);	ajaxgallery.3.0/utils/options.php
34	$str = str_replace(”:”, ””, $str);	ajaxgallery.3.0/utils/options.php
35	$str = str_replace(”,”, ””, $str);	ajaxgallery.3.0/utils/options.php
36	return $str;	ajaxgallery.3.0/utils/options.php
37	}	ajaxgallery.3.0/utils/options.php
39	if(isset($_POST[’save’])){	ajaxgallery.3.0/utils/options.php
40	$sql = ””;	ajaxgallery.3.0/utils/options.php
42	$value = (isset($_POST[’user’]) && trim($_POST[’user’]) != ””? $_POST[’user’] : ”user”) . ”:”;	ajaxgallery.3.0/utils/options.php
43	$value .= (isset($_POST[’album’]) && trim($_POST[’album’]) != ”” ? filterAlbum($_POST[’album’]) : ””) . ”:”;	ajaxgallery.3.0/utils/options.php
44	$value .= (isset($_POST[’rows’]) && trim($_POST[’rows’]) != ”” ? $_POST[’rows’] : ”4”) . ”:”;	ajaxgallery.3.0/utils/options.php
45	$value .= (isset($_POST[’cols’]) && trim($_POST[’cols’]) != ”” ? $_POST[’cols’] : ”4”) . ”:”;	ajaxgallery.3.0/utils/options.php
46	$value .= (isset($_POST[’align’]) && trim($_POST[’align’]) != ”” ? $_POST[’align’] : ”center”) . ”:”;	ajaxgallery.3.0/utils/options.php
47	$value .= (isset($_POST[’pag’]) && trim($_POST[’pag’]) != ”” ? $_POST[’pag’] : ”0”) . ”:”;	ajaxgallery.3.0/utils/options.php
48	$value .= (isset($_POST[’showp’]) && trim($_POST[’showp’]) != ”” ? $_POST[’showp’] : ”true”) . ”:”;	ajaxgallery.3.0/utils/options.php
49	$value .= (isset($_POST[’thumbsize’]) && trim($_POST[’thumbsize’]) != ”” ? $_POST[’thumbsize’] : ”160”);	ajaxgallery.3.0/utils/options.php
50	if(isset($_POST[’gId’])){	ajaxgallery.3.0/utils/options.php
51	$sql = ”UPDATE $wpdb->options SET option_value=’$value’ WHERE option_name=’agItem’ and option_id=”.$_POST[’gId’];	ajaxgallery.3.0/utils/options.php
52	}else{	ajaxgallery.3.0/utils/options.php
55	$wpdb->query($sql);	ajaxgallery.3.0/utils/options.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
64	BD	$wpdb->get_row(”SELECT * FROM $wpdb->options WHERE option_name=’agItem’ and option_id=”.$ggId)	0	ajaxgallery.3.0/utils/options.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
64	variable	$gItem	$wpdb->get_row("SELECT * FROM $wpdb->options WHERE option_name='agItem' and option_id=".$ggId)	0	1	0	0	ajaxgallery.3.0/ utils/options.php
65	variable	$gItem-> option_value	$gItem	0	1	0	0	ajaxgallery.3.0/ utils/options.php
65	variable	split(":", $gItem-> option_value)	$gItem-> option_value	1	1	0	0	ajaxgallery.3.0/ utils/options.php
65	variable	$gItemsopc	split(":", $gItem-> option_value)	1	1	0	0	ajaxgallery.3.0/ utils/options.php
66	variable	$gItemsopc[0]	$gItemsopc	1	1	0	0	ajaxgallery.3.0/ utils/options.php
66	variable	$options['user']	$gItemsopc[0]	1	1	0	1	ajaxgallery.3.0/ utils/options.php
67	variable	$gItemsopc[1]	$gItemsopc	1	1	0	0	ajaxgallery.3.0/ utils/options.php
67	variable	$options['album']	$gItemsopc[1]	1	1	0	1	ajaxgallery.3.0/ utils/options.php
68	variable	$gItemsopc[2]	$gItemsopc	1	1	0	0	ajaxgallery.3.0/ utils/options.php
68	variable	$options['rows']	$gItemsopc[2]	1	1	0	1	ajaxgallery.3.0/ utils/options.php
69	variable	$gItemsopc[3]	$gItemsopc	1	1	0	0	ajaxgallery.3.0/ utils/options.php
69	variable	$options['cols']	$gItemsopc[3]	1	1	0	1	ajaxgallery.3.0/ utils/options.php
71	variable	$gItemsopc[5]	$gItemsopc	1	1	0	0	ajaxgallery.3.0/ utils/options.php
71	variable	$options['pag']	$gItemsopc[5]	1	1	0	1	ajaxgallery.3.0/ utils/options.php
73	variable	$gItemsopc[6]	$gItemsopc	1	1	0	0	ajaxgallery.3.0/ utils/options.php
73	variable	$options['show Pagination']	$gItemsopc[6]	1	1	0	1	ajaxgallery.3.0/ utils/options.php

Execution path constraints:

L	Condition	ES	Ntd	T/F	File
62	isset($_GET['gId']) || isset($_POST['gid'])	0	0	1	ajaxgallery.3.0/utils/options.php

Slice:

L	Instruction	File
62	if(isset($_GET['gId']) || isset($_POST['gid'])){	ajaxgallery.3.0/utils/options.php
63	$ggId = isset($_GET['gId']) ? $_GET['gId'] : $_POST['gId'];	ajaxgallery.3.0/utils/options.php
64	$gItem = $wpdb->get_row("SELECT * FROM $wpdb->options WHERE option_name='agItem' and option_id=".$ggId);	ajaxgallery.3.0/utils/options.php
65	$gItemsopc = split(":", $gItem->option_value);	ajaxgallery.3.0/utils/options.php
66	$options['user'] = $gItemsopc[0];	ajaxgallery.3.0/utils/options.php
67	$options['album'] = $gItemsopc[1];	ajaxgallery.3.0/utils/options.php
68	$options['rows'] = $gItemsopc[2];	ajaxgallery.3.0/utils/options.php
69	$options['cols'] = $gItemsopc[3];	ajaxgallery.3.0/utils/options.php
70	$options['aligntable'] = $gItemsopc[4];	ajaxgallery.3.0/utils/options.php
71	$options['pag'] = $gItemsopc[5];	ajaxgallery.3.0/utils/options.php
73	$options['showPagination'] = $gItemsopc[6];	ajaxgallery.3.0/utils/options.php
75	{	ajaxgallery.3.0/utils/options.php
93	<input name="user" type="text" value="<?php echo $options['user'];?>" />	ajaxgallery.3.0/utils/options.php
100	<input name="album" type="text" value="<?php echo $options['album'];?>" />	ajaxgallery.3.0/utils/options.php
111	<input name="rows" type="text" value="<?php echo $options['rows'];?>" />	ajaxgallery.3.0/utils/options.php
118	<input name="cols" type="text" value="<?php echo $options['cols'];?>" />	ajaxgallery.3.0/utils/options.php
145	<input name="pag" type="text" value="<?php echo $options['pag'];?>" />	ajaxgallery.3.0/utils/options.php
151	<select name='showp' value="<?php echo $options['showPagination'];?>">	ajaxgallery.3.0/utils/options.php

Name of the enclosing function: No function
Pixy's decision: 1

B.3.2.2 Explanation

See B.3.1.2.

B.3.3 Slices #17, #18, #19, #20

B.3.3.1 Verdict: 0 (not vulnerable)

- Name: echo
- Line: 59, 64, 65, 66
- File: ajaxgallery.3.0/utils/list.php

WRITES (1st step)
None of the writes affect the *option_id* field from the *$wpdb->options* table. Check the explanation below.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
54	BD	$wpdb->get_results("SELECT * FROM $wpdb->options WHERE option_name='agItem'");	0	ajaxgallery.3.0/utils/list.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
54	variable	$gItems	$wpdb->get_results(“SELECT * FROM $wpdb->options WHERE option_name=’agItem”’);	0	1	0	0	ajaxgallery.3.0/ utils/list.php
55	variable	$gItem	$gItems	0	1	0	0	ajaxgallery.3.0/ utils/list.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
55	foreach ($gItems as $gItem)	1		1	ajaxgallery.3.0/utils/list.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
55	foreach ($gItems as $gItem)	1	0	1	ajaxgallery.3.0/utils/list.php

Slice:

L	Instruction	File
54	$gItems = $wpdb->get_results(”SELECT * FROM $wpdb->options WHERE option_name=’agItem”’);	ajaxgallery.3.0/utils/list.php
55	foreach ($gItems as $gItem){	ajaxgallery.3.0/utils/list.php
59	<th scope=”row” style=”text-align: center”><?php echo $gItem->option_id;?></th>	ajaxgallery.3.0/utils/list.php
64	<td>[ajax_gallery id=<?php echo $gItem->option_id; ?>]</td>	ajaxgallery.3.0/utils/list.php
65	<td><a href=”admin.php?page=ajaxgallery/utils/options.php& gId=<?php echo $gItem->option_id;?>”>Editar</td>	ajaxgallery.3.0/utils/list.php
66	<td><a href=”admin.php?page=ajaxgallery/utils/list.php& gId=<?php echo $gItem->option_id;?>&delete”>Eliminar</td>	ajaxgallery.3.0/utils/list.php

Name of the enclosing function: No function
phpSAFE’s decision: 1

B.3.3.2 Explanation

False positive. The *option_id* field is never affected by user input, in this plugin.

B.4 VV

A vulnerability is defined as an execution path in which an entry point (EP) reaches a sensitive sink (SS) under these conditions:

- The Entry point (EP) enters the flow of execution through a native PHP variable that handles HTTP requests (e.g., *$_GET*, *$_POST*), or similar constructs provided by PHP frameworks.
- The EP does not participate in any expression before reaching the Sensitive sink (SS), thus not creating dependencies. (see Slice #1)

- The entry point is not directly modified before reaching the SS.
- If the sink is enclosed within an execution path constraint (EPC), the constraint must be satisfied (see line 39 in Slice #2).
- If the sink exists outside any EPC's before it, but is shared by the different paths **in which taintedness was propagated to**, we have a distinct vulnerability per path (Slice #2 and Slice #3). Thus, each vulnerability must either match an EPC, or the negation of that EPC.
- No entry point validation constraints (EPVC's) are verified against the entry point or its dependencies.
- Entry point validation constraints are verified against the entry point or its dependencies, but they are weak constraints that lack context-awareness (i.e. *isset* of a value intended to be numeric, instead of *isset(value) && is_numeric(value)*)
- Neither the entry points or its dependencies are an argument to a function whose output will be the input for the sensitive sink, as shown in most vulnerabilities in this plugin.
- The entry points or dependencies are arguments to a function, but the function does not perform sanitization, (*$_POST['album']* in *filterAlbum($_POST['album'])*).
- *$wpdb->query* is a sensitive sink, since it is not known to sanitize its input [77].
- We have a single vulnerability iff the same sink can be reached by multiple EP's simultaneously, within the same execution path (see Slice #3's *$value* and *$_POST['gId']*)
- [Stored XSS] A stored XSS vulnerability occurs if, for a **write** operation that inserts vulnerable data into a storage unit (such as a database or a file) under a set of conditions (we could call it a "domain" X), we can find at least one **read** operation that retrieves data from the same domain and the same storage unit, **and** that data reaches a sensitive sink without being encoded.

B.5 WAP false negatives

In addition to the false negative in Slice #3, we crafted many examples in which taintedness is propagated to multiple branches, yet WAP will only consider the last branch in both taint analysis and code correction phases. In the following example, only the *else* clause will be considered by WAP. When removed, the previous clause will be considered, and so on...

L	Instruction
3	$a=$_GET['data'];
4	$b=$_GET['optype'];
5	if ($b === "search") {
6	$sql = "SELECT * FROM Customers WHERE CustomerName LIKE '".$a."'";
7	} else {
8	$sql = "DELETE FROM Customers WHERE CustomerName LIKE '".$a."'";
9	}
10	$r = mysql_query($sql);

The problem is caused by a limitation in WAP's TST, when the same symbol is shared by different branches; in this case, it is the variable *$sql*.

B.6 Misc.

- Entry point validation constraints (EPVC's) are execution path constraints (EPC's), but the reverse is not necessarily true (see line 39 in Slice #2)

Appendix C

Plugin: calculated-fields-form.1.0.10

C.1 Type: SQLI

C.1.1 Slice #1

C.1.1.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #4
wap SQLI Slice #15
Sensitive sink

- Name: $wpdb->query
- Line: 69
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry points:

L	Type	Name	RSD	File
69	GET	$_GET['u']	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
69	GET	$_GET["name"]	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
69	function	esc_sql($_GET["name"])		0	1	0	1	calculated-fields-form.1.0.10/ cp_calculatedfieldsf_admin_ int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
67	isset($_GET['u']) && $_GET['u'] != ''	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
67	isset($_GET['u']) && $_GET['u'] != ''	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
67	else if (isset($_GET['u']) && $_GET['u'] != '')	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
68	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
69	$wpdb->query('UPDATE `'.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE.'` SET form_name="'.esc_sql($_GET["name"]).'" WHERE id='.$_GET['u']);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

wap's decision: 1
phpSAFE's decision: 1

C.1.1.2 Explanation

Tainted input (*$_GET['u']*) reaches sensitive sink without sanitization. In regard to *$_GET["name"]*, the *wpdb->esc_sql()* function can be used to escape input that will be used within quotes [83].

C.1.2 Slice #2

C.1.2.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #5
wap SQLI Slice #16
Sensitive sink

- Name: $wpdb->query
- Line: 74

- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry points:

L	Type	Name	RSD	File
74	GET	$_GET['d']	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
72	isset($_GET['d']) && $_GET['d'] != ''	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
72	isset($_GET['d']) && $_GET['d'] != ''	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
72	else if (isset($_GET['d']) && $_GET['d'] != '')	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
73	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
74	$wpdb->query('DELETE FROM `'.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE.'` WHERE id='.$_GET['d']);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

wap's decision: 1
phpSAFE's decision: 1

C.1.2.2 Explanation

Tainted input reaches sensitive sink without sanitization.

C.1.3 Slice #3

C.1.3.1 Verdict: 0 (not vulnerable)

phpSAFE SQLI Slice #6
wap SQLI Slice #17
Sensitive sink

- Name: $wpdb->query
- Line: 99
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry points:

L	Type	Name	RSD	File
88	GET	$_GET[“chs”]	1	calculated-fields-form.1.0.10/ cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
88	variable	$target_charset	$_GET[“chs”]	1	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
86	$_GET[“chs”] != ”	1	83	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
83	isset($_GET[’ac’]) && $_GET[’ac’] == ‘st’	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
86	$_GET[“chs”] != ”	1	1	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
90	foreach ($tables as $tab)	1	1	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
93	foreach ($myrows as $item)	1	1	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
97	preg_match(“/^varchar\((\d+)\)$/i”, $type, $mat) \|\| !strcasecmp($type, “CHAR”) || !strcasecmp($type, “TEXT”) \|\| !strcasecmp($type, “MEDIUMTEXT”)	1	1	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

Slice:

L	Instruction	File
83	} else if (isset($_GET['ac']) && $_GET['ac'] == 'st')	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
84	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
86	if ($_GET["chs"] != '')	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
87	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
88	$target_charset = $_GET["chs"];	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
90	foreach ($tables as $tab)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
91	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
93	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
94	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
97	if (preg_match("/^varchar\((\d+)\)$/i", $type, $mat) \|\| !strcasecmp($type, "CHAR") || !strcasecmp($type, "TEXT") \|\| !strcasecmp($type, "MEDIUMTEXT"))	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
98	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
99	$wpdb->query("ALTER TABLE {$tab} CHANGE {$name} {$name} {$type} COLLATE {$target_charset}");	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

phpSAFE's decision: 1
wap's decision: 1

C.1.3.2 Explanation

False positive. Even though *$target_charset* is tainted, the *ALTER TABLE* command will only accept valid collations (whitelist).

C.1.4 Slice #4

C.1.4.1 Verdict: 0 (not vulnerable)

wap SQLI Slice #14
Sensitive sink

- Name: $wpdb->insert
- Line: 16
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry points:

L	Type	Name	RSD	File
17	GET	$_GET["name"]	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
17	function	stripcslashes ($_GET[“name”])	$_GET[“name”]	1	1	0	1	calculated-fields-form.1.0.10/ cp_calculatedfieldsf_admin_ int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
11	isset($_GET[’a’]) && $_GET[’a’] == ‘1’	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	Path
11	if (isset($_GET[’a’]) && $_GET[’a’] == ’1’)	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
12	{	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
16	$wpdb->insert($wpdb->prefix.CP_ CALCULATEDFIELDSF_FORMS_TABLE, array(	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php
17	’form_name’ => stripcslashes($_GET[”name”]),	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Name of the enclosing function: No function
wap’s decision: 1

C.1.4.2 Explanation

The *$wpdb->insert($table, $data, $format)* function takes unescaped input [77] and performs sanitization before insertion. It requires the table name and the data to be stored in separate variables, and provides them to a prepared statement correctly [85]. Thus, it is being used correctly in this situation. *stripcslashes()* removes backslashes [69], which is necessary for the correct usage of *$wpdb->insert*. **This slice was produced by mistake.**

C.2 Type: Stored XSS

C.2.1 Slice #5

C.2.1.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2775
Sensitive sink

- Name: echo

- Line: 196
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)
None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
191	BD	$wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
191	variable	$myrows	$wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
196	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
196	<td nowrap><?php echo $item->id; ?></td>	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE's decision: 1

C.2.1.2 Explanation

False positive. *$wpdb->prefix* is the database table prefix, which is "wp_" by default; it supports only letters, numbers or underscore, and is defined in wp-config.php [80]. The *id* field (from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table) is never affected by user input, in this plugin.

C.2.2 Slice #6

C.2.2.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2778
pixy XSS Slice #256
Sensitive sink

- Name: echo
- Line: 197
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
69	GET	$_GET['u']	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
69	GET	$_GET["name"]	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
69	function	esc_sql ($_GET["name"])	$_GET["name"]	1	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
67	isset($_GET['u']) && $_GET['u'] != ''	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
67	isset($_GET['u']) && $_GET['u'] != ''	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
67	else if (isset($_GET['u']) && $_GET['u'] != '')	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
68	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
69	$wpdb->query('UPDATE `'.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE.'` SET form_name="'.esc_sql($_GET["name"]).'" WHERE id='.$_GET['u']);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

READS (2nd step)
Entry points:

L	Type	Name	RSD	File
191	BD	$wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
191	variable	$myrows	$wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
197	<td nowrap><input type="text" name="calname_<?php echo $item->id; ?>" id="calname_<?php echo $item->id; ?>" value="<?php echo esc_attr($item->form_name); ?>" /></td>	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php

Name of the enclosing function: No function
pixy's decision: 1
phpSAFE's decision: 1

C.2.2.2 Explanation

False positive. Even though the field *form_name* for this table may contain malicious data, *esc_attr()* [81] is being used to escape data for the *value* attribute.

C.2.3 Slice #7

C.2.3.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2779
Sensitive sink

- Name: echo
- Line: 200
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)

None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below.

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
191	BD	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
191	variable	$myrows	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results(”SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
200	<input type=”button” name=”calupdate_<?php echo $item->id; ?>” value=”Update” onclick=”cp_updateItem(<?php echo $item->id; ?>);” />	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE's decision: 1

C.2.3.2 Explanation

False positive (see Slice #5).

C.2.4 Slice #8

C.2.4.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2780
Sensitive sink

- Name: echo
- Line: 201
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)
None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
191	BD	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
191	variable	$myrows	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_ CALCULATEDFIELDSF_ FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results(”SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
201	<input type=”button” name=”calmanage_<?php echo $item->id; ?>” value=”Settings” onclick=”cp_manageSettings(<?php echo $item->id; ?>);” />	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE’s decision: 1

C.2.4.2 Explanation

False positive (see Slice #5).

C.2.5 Slice #9

C.2.5.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2781
Sensitive sink

- Name: echo
- Line: 202
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)
None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
191	BD	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
191	variable	$myrows	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
202	<input type="button" name="calmanage_<?php echo $item->id; ?>" value="Messages" onclick="cp_viewMessages(<?php echo $item->id; ?>);" />	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE's decision: 1

C.2.5.2 Explanation

False positive (see Slice #5).

C.2.6 Slice #10

C.2.6.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2782
Sensitive sink

- Name: echo
- Line: 203
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)
None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below.
READS (2nd step)

Entry points:

L	Type	Name	RSD	File
191	BD	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	Ntd	RSD	File
191	variable	$myrows	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results(”SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
203	<input type=”button” name=”calclone_<?php echo $item->id; ?>” value=”Clone” onclick=”cp_cloneItem(<?php echo $item->id; ?>);” />	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE's decision: 1

C.2.6.2 Explanation

False positive (see Slice #5).

C.2.7 Slice #11

C.2.7.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2783
Sensitive sink

- Name: echo
- Line: 204
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)
None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below. READS (2nd step)
Entry points:

L	Type	Name	R S D	File
191	BD	$wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
191	variable	$myrows	$wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_ CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php
204	<input type="button" name="caldelete_<?php echo $item->id; ?>" value="Delete" onclick="cp_deleteItem(<?php echo $item->id; ?>);" />	calculated-fields-form.1.0.10/cp_ calculatedfieldsf_ admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE's decision: 1

C.2.7.2 Explanation

False positive (see Slice #5).

C.2.8 Slice #12

C.2.8.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2790
Sensitive sink

- Name: echo
- Line: 206
- File: calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

WRITES (1st step)

None of the writes affect the *id* field from the *$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE* table. Check the explanation below.

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
191	BD	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
191	variable	$myrows	$wpdb->get_results(“SELECT * FROM ”.$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	variable	$item	$myrows	0	1	0	0	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
197	variable	$item->id	$item	0	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
192	foreach ($myrows as $item)	1		1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
192	foreach ($myrows as $item)	1	0	1	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Slice:

L	Instruction	File
191	$myrows = $wpdb->get_results("SELECT * FROM ".$wpdb->prefix.CP_CALCULATEDFIELDSF_FORMS_TABLE);	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
192	foreach ($myrows as $item)	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
193	{	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php
206	<td nowrap>[CP_CALCULATED_FIELDS id="<?php echo $item->id; ?>"]</td>	calculated-fields-form.1.0.10/cp_calculatedfieldsf_admin_int_list.inc.php

Name of the enclosing function: No function
phpSAFE's decision: 1

C.2.8.2 Explanation

False positive (see Slice #5).

C.3 VV

A vulnerability occurs if (cont.):

- The entry points or its dependencies, if they are already sanitized, are arguments to a de-sanitization function[1] (such as *stripcslashes()* [69]) whose output will be input for the sensitive sink;
- **[For SQLI and numeric input]**, the *wpdb->esc_sql()*, *mysqli_real_escape_string()* or *mysql_real_escape_string()* functions are applied to numeric entry points or dependencies in the same way that are applied to strings, but their results unchange the numeric inputs [83];
- **[For SQLI]**, the *$wpdb->prepare()* function is used incorrectly - there is a vulnerability iff this function is used with only one argument that contains a tainted input [75].

C.4 Misc.

- The *mysql_real_escape_string()* family does not offer protection against XSS attacks. For instance, it is still possible to inject some JavaScript code and HTML

[1] Also known as *revert function* [41]

elements, as the ‘<’ and ‘>’ characters are not sanitized, among others [52][50][86].

- Functions from the *mysql_real_escape_string()* family, when used as an anti-SQLI measure, can only be safely used to escape input that will be used within quotes [83].
- The *$wpdb->insert($table, $data, $format)* function takes unescaped input [77] and performs sanitization before insertion. It requires the table name and the data to be stored in separate variables, and provides them to a prepared statement correctly [85].
- Revert functions may be used to produce a de-sanitized value on purpose. Such practice is sometimes necessary in order to avoid interference with other functions that also perform sanitization. For example, *stripcslashes()* is sometimes used to remove backslashes, which is necessary for the correct usage of *$wpdb->insert*.

Appendix D

Plugin: collision-testimonials.3.0

D.1 Type: SQLI

D.1.1 Slice #1

D.1.1.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #7
Sensitive sink

- Name: $wpdb->query
- Line: 103
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/ pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/ pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/ pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/ pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/ pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/ pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/ pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/ pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/ pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http://' ? 'http://' : '') . $website;	collision-testimonials.3.0/ pages/testimonials.php
94	}	collision-testimonials.3.0/ pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/ pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/ pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/ pages/testimonials.php
101	}	collision-testimonials.3.0/ pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/ pages/testimonials.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1

D.1.1.2 Explanation

$_POST['id'] is the main concern; its taintedness is propagated to *$id*, reaching the sensitive sink without any sanitization. *$testimonials* is simply the name of the WordPress table. Line 93 checks whether *$website* contains the HTTP protocol prefix. In regard to other variables, they are transformed by *$wpdb->escape*. This function is deprecated, and uses *addslashes*, which is vulnerable to SQLI under certain conditions (check Misc. at the end of this plugin for more information).
wap-2.1 did not detect this vulnerability. The problem may be due to string concatenation. Revisiting the resolver methods is necessary.

D.1.2 Slice #2

D.1.2.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #8
Sensitive sink

- Name: $wpdb->query
- Line: 110
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
108	POST	$_POST	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
108	variable	$id	$_POST	foreach($_POST as $id)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
109	variable	$featsql	$id	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
108	foreach($_POST as $id)	1	107	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
107	isset($_POST['featQuote'])	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
108	foreach($_POST as $id)	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
107	if (isset($_POST[′featQuote′])) {	collision-testimonials.3.0/ pages/testimonials.php
108	foreach($_POST as $id){	collision-testimonials.3.0/ pages/testimonials.php
109	$featsql = ”UPDATE $testimonials SET featured=1 WHERE id=$id”;	collision-testimonials.3.0/ pages/testimonials.php
110	$results = $wpdb->query($featsql);	collision-testimonials.3.0/ pages/testimonials.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
wap's decision: 1

D.1.2.2 Explanation

Tainted input reaches sensitive sink without sanitization.
wap-2.1 detects this vulnerability, despite being unable to fix it.

D.1.3 Slice #3

D.1.3.1 Verdict: 1 (vulnerable)

pixy SQLI Slice #2
RIPS SQLI Slice #5
phpSAFE SQLI Slice #9
Sensitive sink

- Name: mysql_query
- Line: 115
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
114	GET	$_GET['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
114	variable	$id	$_GET['id']	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
113	isset($_GET['featQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
113	if (isset($_GET['featQuote'])) {	collision-testimonials.3.0/ pages/testimonials.php
114	$id = $_GET['id'];	collision-testimonials.3.0/ pages/testimonials.php
115	mysql_query("UPDATE $testimonials SET featured=1 WHERE id =$id");	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side. (Blind exploitation)

———————————————

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
pixy's decision: 1
RIPS's decision: 1
wap's decision: 1

D.1.3.2 Explanation

Tainted input reaches sensitive sink without sanitization.
wap-2.1 detects this vulnerability, despite being unable to fix it.

D.1.4 Slice #4

D.1.4.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #10
Sensitive sink

- Name: $wpdb->query
- Line: 122
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
120	POST	$_POST	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	RSD	File
120	variable	$id	$_POST	foreach($_POST as $id)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
121	variable	$featsql	$id	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
120	foreach($_POST as $id)	1	119	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
119	isset($_POST['unfeatQuote'])	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
120	foreach($_POST as $id)	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
119	if (isset($_POST[′unfeatQuote′])) {	collision-testimonials.3.0/ pages/testimonials.php
120	foreach($_POST as $id){	collision-testimonials.3.0/ pages/testimonials.php
121	$featsql = ”UPDATE $testimonials SET featured=0 WHERE id=$id”;	collision-testimonials.3.0/ pages/testimonials.php
122	$results = $wpdb->query($featsql);	collision-testimonials.3.0/ pages/testimonials.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
wap's decision: 1

D.1.4.2 Explanation

Tainted input reaches sensitive sink without sanitization.
wap-2.1 detects this vulnerability, despite being unable to fix it.

D.1.5 Slice #5

D.1.5.1 Verdict: 1 (vulnerable)

RIPS SQLI Slice #6
phpSAFE SQLI Slice #11
pixy SQLI Slice #3
Sensitive sink

- Name: mysql_query
- Line: 127
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	R S D	File
126	GET	$_GET['id']	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
126	variable	$id	$_GET['id']	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
125	isset($_GET['unfeatQuote'])	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
125	if (isset($_GET[′unfeatQuote′])) {	collision-testimonials.3.0/ pages/testimonials.php
126	$id = $_GET[′id′];	collision-testimonials.3.0/ pages/testimonials.php
127	mysql_query("UPDATE $testimonials SET featured=0 WHERE id =$id");	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side. (Blind exploitation)

———————————————

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

RIPS's decision: 1
pixy's decision: 1
phpSAFE's decision: 1
wap's decision: 1

D.1.5.2 Explanation

Tainted input reaches sensitive sink without sanitization.
wap-2.1 detects this vulnerability, despite being unable to fix it.

D.1.6 Slice #6

D.1.6.1 Verdict: 0 (not vulnerable)

phpSAFE SQLI Slice #12
Sensitive sink

- Name: $wpdb->query
- Line: 149
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
146	POST	$_POST	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
146	variable	$id	$_POST	foreach($_POST as $id)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
148	variable	$statussql	$id	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
146	foreach($_POST as $id)	1	131	1	collision-testimonials.3.0/ pages/testimonials.php
147	is_numeric($id)	1	146	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
131	isset($_POST['bulkQuote'])	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
146	foreach($_POST as $id)	1	1	1	collision-testimonials.3.0/ pages/testimonials.php
147	is_numeric($id)	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
131	if (isset($_POST['bulkQuote'])){	collision-testimonials.3.0/ pages/testimonials.php
146	foreach($_POST as $id){	collision-testimonials.3.0/ pages/testimonials.php
147	if (is_numeric($id)){	collision-testimonials.3.0/ pages/testimonials.php
148	$statussql = "UPDATE $testimonials SET status=$status WHERE id=$id";	collision-testimonials.3.0/ pages/testimonials.php
149	$results = $wpdb->query($statussql);	collision-testimonials.3.0/ pages/testimonials.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: function

phpSAFE's decision: 1
wap's decision: 1

D.1.6.2 Explanation

Not vulnerable due to the *is_numeric* restriction. Curiously, **wap-2.1 classified it as a vulnerability**, and **applied a fix to *$status*, instead of *$id* (*$status* is untainted).**

D.1.7 Slice #7

D.1.7.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #13
pixy SQLI Slice #4
RIPS SQLI Slice #7
Sensitive sink

- Name: mysql_query
- Line: 157
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
156	GET	$_GET['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
156	variable	$id	$_GET['id']	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
155	isset($_GET['deleteQuote'])	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
155	if (isset($_GET[′deleteQuote′])) {	collision-testimonials.3.0/ pages/testimonials.php
156	$id = $_GET[′id′];	collision-testimonials.3.0/ pages/testimonials.php
157	mysql_query(”DELETE FROM $testimonials WHERE id=′$id′”);	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results——-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side. (Blind exploitation)

———————————————

Misc. (enclosing modules & functions)

- Name of the enclosing function: function

RIPS's decision: 1
pixy's decision: 1
phpSAFE's decision: 1
wap's decision: 1

D.1.7.2 Explanation

Tainted input reaches sensitive sink without sanitization.**wap-2.1 detects this vulnerability, and applies the fix correctly**.

D.1.8 Slice #8

D.1.8.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #14
pixy SQLI Slice #5

RIPS SQLI Slice #8
Sensitive sink

- Name: mysql_query
- Line: 163
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
162	POST	$_POST	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
162	variable	$id	$_POST	foreach($_POST as $id)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
162	foreach($_POST as $id)	1	161	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
161	isset($_POST['deleteQuote'])	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
162	foreach($_POST as $id)	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
161	if (isset($_POST[′deleteQuote′])) {	collision-testimonials.3.0/ pages/testimonials.php
162	foreach($_POST as $id) {	collision-testimonials.3.0/ pages/testimonials.php
163	mysql_query(”DELETE FROM $testimonials WHERE id=′$id′”);	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results——-
Additional comment:

Userinput reaches sensitive sink. For more information, press the help icon on the left side. (Blind exploitation)

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
RIPS's decision: 1
pixy's decision: 1
wap's decision: 1

D.1.8.2 Explanation

Tainted input reaches sensitive sink without sanitization. **wap-2.1 detects this vulnerability and fixes it correctly, but prints an incorrect trace. Line 156's *$_GET['id']* is reported instead of line 162's *$_POST*.**

D.1.9 Slice #9

D.1.9.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #15
pixy SQLI Slice #6
RIPS SQLI Slice #9
Sensitive sink

- Name: mysql_query
- Line: 177
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
175	GET	$_GET['oldid']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
175	variable	$oldid	$_GET['oldid']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
176	variable	$query	$oldid	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
170	($_GET['action'] == "edit") && (!isset($_POST['editQuote']))	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET[′action′] == "edit") && (!isset($_POST[′editQuote′]))){	collision-testimonials.3.0/ pages/testimonials.php
175	$oldid = $_GET[′oldid′];	collision-testimonials.3.0/ pages/testimonials.php
176	$query = "SELECT * FROM $testimonials WHERE id=′$oldid′";	collision-testimonials.3.0/ pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
RIPS's decision: 1
pixy's decision: 1
wap's decision: 1

D.1.9.2 Explanation

Tainted input reaches sensitive sink without sanitization. **wap-2.1 detects this vulnerability, and applies the fix correctly**.

D.1.10 Slice #10

D.1.10.1 Verdict: 1 (vulnerable)

pixy SQLI Slice #7
RIPS SQLI Slice #10
Sensitive sink

- Name: mysql_query

- Line: 333
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
318	GET	$_GET['sort']	0	collision-testimonials.3.0/ pages/testimonials.php
326	GET	$_GET['dir']	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
318	variable	$sort	$_GET['sort']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
326	variable	$dir	$_GET['dir']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
332	variable	$query	$sort, $dir	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
317	isset($_GET['sort']) && !empty($_GET['sort'])	0		1	collision-testimonials.3.0/ pages/testimonials.php
325	isset($_GET['dir']) && !empty($_GET['dir'])	0		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
317	isset($_GET['sort']) && !empty($_GET['sort'])	0	0	1	collision-testimonials.3.0/pages/testimonials.php
325	isset($_GET['dir']) && !empty($_GET['dir'])	0	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
317	if (isset($_GET[′sort′]) && !empty($_GET[′sort′])){	collision-testimonials.3.0/ pages/testimonials.php
318	$sort = $_GET[′sort′];	collision-testimonials.3.0/ pages/testimonials.php
319	}	collision-testimonials.3.0/ pages/testimonials.php
325	if (isset($_GET[′dir′]) && !empty($_GET[′dir′])){	collision-testimonials.3.0/ pages/testimonials.php
326	$dir = $_GET[′dir′];	collision-testimonials.3.0/ pages/testimonials.php
327	}	collision-testimonials.3.0/ pages/testimonials.php
332	$query = ”SELECT * FROM $testimonials ORDER BY $sort $dir”;	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1
pixy's decision: 1

D.1.10.2 Explanation

Tainted input reaches sensitive sink without sanitization. **wap-2.1 does not detect this vulnerability**. Once again, when multiple branches share the same symbol (e.g., lines 318 and 321 share *$sort*, lines 326 and 329 share *$dir*), only the last branch is considered. In this case, the last branch does not hold a tainted version of the variable (i.e., *$sort = "id"; $dir = "asc";* in lines 321 and 329 respectively), thus resulting in no vulnerabilities detected for the entirety of the conditional construct.

D.2 Type: Reflected XSS

D.2.1 Slice #11

D.2.1.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2791
RIPS XSS Slice #32
pixy XSS Slice #257
Sensitive sink

- Name: echo
- Line: 190
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
190	SERVER	$_SERVER['REQUEST_URI'])	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
170	($_GET['action'] == "edit") && (!isset($_POST['editQuote']))	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
179	while ($data = mysql_fetch_array($results))	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET['action'] == "edit") && (!isset($_POST['editQuote']))){	collision-testimonials.3.0/ pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
190	<form method="post" action="<?php echo str_replace('%7E', '~', $_SERVER['REQUEST_URI']); ?>">	collision-testimonials.3.0/ pages/testimonials.php

Name of the enclosing function: No function

Additional RIPS results———-

Additional comment:

Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

pixy's decision: 1

RIPS's decision: 1

phpSAFE's decision: 1

wap's decision: 1

D.2.1.2 Explanation

Tainted input reaches sensitive sink without sanitization; the request URL, stored in *$_SERVER['REQUEST_URI']*, can be malicious.

For *str_replace* [54], the first argument is the search string, the second argument is the replace string, and the third is the content that is subject to modification (i.e., whenever a match occurs, the subject is modified so that the replace string overwrites the matched content). This particular use of *str_replace* is not enough to prevent a reflected XSS attack.

wap-2.1 classified this situation as a potential false positive due to the use of *str_replace*. However, the existence of *str_replace*, alone, is not enough to block taintedness.

D.2.2 Slice #12

D.2.2.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2792
pixy XSS Slice #258
RIPS XSS Slice #38
Sensitive sink

- Name: echo
- Line: 238
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
238	SERVER	$_SERVER['REQUEST_URI']	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
170	($_GET['action'] == "edit") && (!isset($_POST['editQuote']))	1	0	0	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
234	<?php } } else { ?>	collision-testimonials.3.0/pages/testimonials.php
238	<form method="post" action="<?php echo str_replace('%7E', '~', $_SERVER['REQUEST_URI']); ?>">	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function
Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

RIPS's decision: 1
pixy's decision: 1
phpSAFE's decision: 1
wap's decision: 1

D.2.2.2 Explanation

See Slice #11.

D.2.3 Slice #13

D.2.3.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2793
RIPS XSS Slice #39
pixy XSS Slice #259
Sensitive sink

- Name: echo
- Line: 284
- File: collision-testimonials.3.0/pages/testimonials.php

Entry points:

L	Type	Name	RSD	File
284	SERVER	$_SERVER['REQUEST_URI'])	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
284	<form method="post" action="<?php echo str_replace('%7E', '~', $_SERVER['REQUEST_URI']); ?>">	collision-testimonials.3.0/ pages/testimonials.php

Name of the enclosing function: No function
Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

pixy's decision: 1
phpSAFE's decision: 1
RIPS's decision: 1
wap's decision: 1

D.2.3.2 Explanation

See Slice #11.

D.2.4 Slice #14

D.2.4.1 Verdict: 1 (vulnerable)

Sensitive sink

- Name: mysql_query
- Line: 179
- File: collision-testimonials.3.0/collision-testimonials.php

Entry points:

L	Type	Name	RSD	File
159	parameter	$id	0	collision-testimonials.3.0/collision-testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
168	!empty($id)	1		1	collision-testimonials.3.0/collision-testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
168	!empty($id)	1	0	1	collision-testimonials.3.0/collision-testimonials.php

Slice:

L	Instruction	File
159	function collision_testimonials($id="){	collision-testimonials.3.0/ collision-testimonials.php
168	if (!empty($id)){ // If the id has been set, show that testimonial instead!	collision-testimonials.3.0/ collision-testimonials.php
178	$query = "SELECT * FROM $testimonials WHERE status!='1' AND status!='2' AND id IN($id) ORDER BY FIELD(id, $id)";	collision-testimonials.3.0/ collision-testimonials.php
179	$results = mysql_query($query);	collision-testimonials.3.0/ collision-testimonials.php

Name of the enclosing function: collision_testimonials($id="")
RIPS's decision: 1

D.2.4.2 Explanation

The entry point is the function parameter *$id*, which participates (unsanitized) in a *SELECT* command. **It is mainly a SQLI vulnerability** (it was mistakenly placed within the Reflected XSS section). However, an interesting observation is that reflected XSS attacks are indeed possible, in line 178. An attacker could craft the first instance of *$id* with the following payload:

999999) union all select ‘a’, ‘b’, ‘<script>alert(\’hi\’);</script>’, ‘d’ . . . - -

Through the use of the *UNION ALL* operator, the attacker can add new rows to the result set. Note that a space at the end of the comment indicator is required. It is also required to have dummy columns (in this case, ‘a’,’b’, etc.) in order to match the exact number of columns from the original query. Recall that the *SELECT* statement from line 178 retrieves all columns, so it would be a matter of trial and error until discovering the correct number of columns - and which columns hold data that can be displayed - that make such attack possible. **False negative for all tools except RIPS.**

D.3 Type: Stored injection

D.3.1 Slice #15

D.3.1.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #30
Sensitive sink

- Name: echo
- Line: 203
- File: collision-testimonials.3.0/collision-testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST[’status’]	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST[’name’]	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST[’location’]	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST[’quote’]	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST[’website’]	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST[’id’]	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/ pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/ pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/ pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/ pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/ pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/ pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/ pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/ pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/ pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http://' ? 'http://' : '') . $website;	collision-testimonials.3.0/ pages/testimonials.php
94	}	collision-testimonials.3.0/ pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/ pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/ pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/ pages/testimonials.php
101	}	collision-testimonials.3.0/ pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/ pages/testimonials.php

Name of the enclosing function: No function READS (2nd step)

Entry points:

L	Type	Name	R S D	File
159	parameter	$id	0	collision-testimonials.3.0/ collision-testimonials.php
171	BD	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_prefix"', ARRAY_A);	0	collision-testimonials.3.0/ collision-testimonials.php
174	BD	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_suffix"', ARRAY_A);	0	collision-testimonials.3.0/ collision-testimonials.php
179	BD	mysql_query($query)	0	collision-testimonials.3.0/ collision-testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint c.	T/F	Ntd	RSD	File
171	variable	$testimonial_prefix	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_prefix"', ARRAY_A)	0	1	0	0	collision-testimonials.3.0/collision-testimonials.php
172	variable	$testimonial_prefix	$testimonial_prefix['value']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
174	variable	$testimonial_suffix	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_suffix"', ARRAY_A)	0	1	0	0	collision-testimonials.3.0/collision-testimonials.php
175	variable	$testimonial_suffix	$testimonial_suffix['value']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
178	variable	$query	$id	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
179	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
181	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
183	function	stripslashes($data['name'])	$data['name']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
183	variable	$name	stripslashes ($data['name'])	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
184	function	stripslashes($data['date'])	$data['date']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
184	variable	$date	stripslashes($data['date'])	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
185	function	stripslashes($data['location'])	$data['location']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
185	variable	$location	stripslashes($data['location'])	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
187	variable	$location	$location	!empty ($location)	1	0	0	collision-testimonials.3.0/collision-testimonials.php
189	function	stripslashes($data['quote'])	$data['quote']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
189	variable	$quote	stripslashes ($data['quote'])	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
192	variable	$website	$data['website']	1	1	0	0	collision-testimonials.3.0/collision-testimonials.php
194	variable	$websitepre	$website	!empty ($website)	1	0	0	collision-testimonials.3.0/collision-testimonials.php
199	variable	$the Testimonials	$testimonial_prefix, $quote, $websitepre, $name, $location, $testimonial_suffix	1	1	0	1	collision-testimonials.3.0/collision-testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
168	!empty($id)	1		1	collision-testimonials.3.0/collision-testimonials.php
181	while ($data = mysql_fetch_array($results))	0	168	1	collision-testimonials.3.0/collision-testimonials.php
186	!empty($location)	0	181	1	collision-testimonials.3.0/collision-testimonials.php
193	!empty($website)	0	181	1	collision-testimonials.3.0/collision-testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
168	!empty($id)	1	0	1	collision-testimonials.3.0/collision-testimonials.php
181	while ($data = mysql_fetch_array($results))	0	1	1	collision-testimonials.3.0/collision-testimonials.php
186	!empty($location)	0	1	1	collision-testimonials.3.0/collision-testimonials.php
193	!empty($website)	0	1	1	collision-testimonials.3.0/collision-testimonials.php

Slice:

L	Instruction	File
159	function collision_testimonials($id=“”){	collision-testimonials.3.0/ collision-testimonials.php
168	if (!empty($id)){ // If the id has been set, show that testimonial instead!	collision-testimonials.3.0/ collision-testimonials.php
171	$testimonial_prefix = $wpdb->get_row(”SELECT value FROM $testimonials_settings WHERE name=′testimonial_prefix′”, ARRAY_A);	collision-testimonials.3.0/ collision-testimonials.php
172	$testimonial_prefix = $testimonial_prefix[′value′];	collision-testimonials.3.0/ collision-testimonials.php
174	$testimonial_suffix = $wpdb->get_row(”SELECT value FROM $testimonials_settings WHERE name=′testimonial_suffix′”, ARRAY_A);	collision-testimonials.3.0/ collision-testimonials.php
175	$testimonial_suffix = $testimonial_suffix[′value′];	collision-testimonials.3.0/ collision-testimonials.php
178	$query = ”SELECT * FROM $testimonials WHERE status!=′1′ AND status!=′2′ AND id IN($id) ORDER BY FIELD(id, $id)”;	collision-testimonials.3.0/ collision-testimonials.php
179	$results = mysql_query($query);	collision-testimonials.3.0/ collision-testimonials.php
181	while ($data = mysql_fetch_array($results)){	collision-testimonials.3.0/ collision-testimonials.php
183	$name = stripslashes($data[′name′]);	collision-testimonials.3.0/ collision-testimonials.php
184	$date = stripslashes($data[′date′]);	collision-testimonials.3.0/ collision-testimonials.php
185	$location = stripslashes($data[′location′]);	collision-testimonials.3.0/ collision-testimonials.php
186	if (!empty($location)){	collision-testimonials.3.0/ collision-testimonials.php
187	$location = ”, ”.$location;	collision-testimonials.3.0/ collision-testimonials.php
188	}	collision-testimonials.3.0/ collision-testimonials.php
189	$quote = stripslashes($data[′quote′]);	collision-testimonials.3.0/ collision-testimonials.php
192	$website = $data[′website′];	collision-testimonials.3.0/ collision-testimonials.php
193	if (!empty($website)) {	collision-testimonials.3.0/ collision-testimonials.php
194	$websitepre = ′<a href=”′.$website.′”>′;	collision-testimonials.3.0/ collision-testimonials.php
196	}	collision-testimonials.3.0/ collision-testimonials.php
199	$theTestimonials .= $testimonial_prefix . $quote . ′ — ′ . $websitepre . $name . $websitesuf . ′′ . $location . ”” . $testimonial_suffix;	collision-testimonials.3.0/ collision-testimonials.php
200	}	collision-testimonials.3.0/ collision-testimonials.php
203	echo $theTestimonials;	collision-testimonials.3.0/ collision-testimonials.php

Name of the enclosing function: collision_testimonials($id=“”)
Additional RIPS results———-
Additional comment:
Userinput reaches sensitive sink when function collision_testimonials is called.

———————————————

RIPS’s decision: 1

D.3.1.2 Explanation

A Stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . “testimonials”*** throughout the files in the plugin. In the file

collision-testimonials.3.0/pages/testimonials.php, an *UPDATE* command is performed on records that contain a certain *$id* value. *$wpdb->escape* applies *addslashes* to its arguments, which is not enough to prevent the injection of malicious data in the table. Back in *collision-testimonials.3.0/collision-testimonials.php*, as long as the updated *status* is neither 1 nor 2, and *$id* matches the value used on the *UPDATE*, the data is retrieved by the *SELECT* statement, and its contents are displayed (executed) by *echo* in line 203. ***$testimonials_settings*'s *value*** field suffers from a similar problem.

D.3.2 Slice #16

D.3.2.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #31
Sensitive sink

- Name: echo
- Line: 245
- File: collision-testimonials.3.0/collision-testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST[′editQuote′])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST[′status′];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST[′name′];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST[′location′];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST[′quote′];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST[′website′];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST[′id′];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = ””;	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != ′http://′ ? ′http://′ : ′′) . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = ”UPDATE ” . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	” SET status=′” . $wpdb->escape($status) . ”′, name=′” . $wpdb->escape($name) . ”′, location=′” . $wpdb->escape($location) . ”′, quote=′” . $wpdb->escape($quote) . ”′, website=′” . $wpdb->escape($website_final) . ”′ WHERE id=′”. $id .”′”;	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
159	parameter	$id	0	collision-testimonials.3.0/ collision-testimonials.php
210	BD	$wpdb->get_row(“SELECT value FROM $testimonials_settings WHERE name='testimonial_num”’, ARRAY_A)	0	collision-testimonials.3.0/ collision-testimonials.php
215	BD	mysql_query($query)	0	collision-testimonials.3.0/ collision-testimonials.php
235	BD	$wpdb->get_row(“SELECT value FROM $testimonials_settings WHERE name='testimonial_prefix”’, ARRAY_A)	0	collision-testimonials.3.0/ collision-testimonials.php
238	BD	$wpdb->get_row(“SELECT value FROM $testimonials_settings WHERE name='testimonial_suffix”’, ARRAY_A)	0	collision-testimonials.3.0/ collision-testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
210	variable	$testimonial _num	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_num"', ARRAY_A)	0	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
211	variable	$testimonial _num	$testimonial_num['value']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
214	variable	$query	$testimonial_num	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
215	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
217	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
220	function	stripslashes ($data['name'])	$data['name']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
220	variable	$name	stripslashes($data['name'])	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
221	function	stripslashes ($data['location'])	$data['location']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
221	variable	$location	stripslashes($data['location'])	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
223	variable	$location	$location	!empty ($location)	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
225	function	stripslashes ($data['quote'])	$data['quote']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
225	variable	$quote	stripslashes ($data['quote'])	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
228	variable	$website	$data['website']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
230	variable	$websitepre	$website	!empty ($website)	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
235	variable	$testimonial_prefix	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_prefix"', ARRAY_A)	0	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
236	variable	$testimonial_prefix	$testimonial_prefix['value']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
238	variable	$testimonial_suffix	$wpdb->get_row("SELECT value FROM $testimonials_settings WHERE name='testimonial_suffix"', ARRAY_A)	0	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
239	variable	$testimonial_suffix	$testimonial_suffix['value']	1	1	0	0	collision-testimonials.3.0/ collision-testimonials.php
242	variable	$theTestimonials	$testimonial_prefix, $quote, $websitepre, $name, $location, $testimonial_suffix	1	1	0	1	collision-testimonials.3.0/ collision-testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
168	!empty($id)	1		0	collision-testimonials.3.0/collision-testimonials.php
217	while ($data = mysql_fetch_array($results))	0	168	1	collision-testimonials.3.0/collision-testimonials.php
222	!empty($location)	0	217	1	collision-testimonials.3.0/collision-testimonials.php
229	!empty($website)	0	217	1	collision-testimonials.3.0/collision-testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
168	!empty($id)	1	0	0	collision-testimonials.3.0/collision-testimonials.php
217	while ($data = mysql_fetch_array($results))	0	1	1	collision-testimonials.3.0/collision-testimonials.php
222	!empty($location)	0	1	1	collision-testimonials.3.0/collision-testimonials.php
229	!empty($website)	0	1	1	collision-testimonials.3.0/collision-testimonials.php

Slice:

L	Instruction	File
159	function collision_testimonials($id=“”){	collision-testimonials.3.0/ collision-testimonials.php
208	else { // Nothing set, let′s randomize it!	collision-testimonials.3.0/ collision-testimonials.php
210	$testimonial_num = $wpdb->get_row(”SELECT value FROM $testimonials_settings WHERE name=′testimonial_num′”, ARRAY_A);	collision-testimonials.3.0/ collision-testimonials.php
211	$testimonial_num = $testimonial_num[′value′];	collision-testimonials.3.0/ collision-testimonials.php
214	$query = ”SELECT * FROM $testimonials WHERE status!=′1′ AND status!=′2′ ORDER BY RAND() LIMIT $testimonial_num”;	collision-testimonials.3.0/ collision-testimonials.php
215	$results = mysql_query($query);	collision-testimonials.3.0/ collision-testimonials.php
217	while ($data = mysql_fetch_array($results)){	collision-testimonials.3.0/ collision-testimonials.php
220	$name = stripslashes($data[′name′]);	collision-testimonials.3.0/ collision-testimonials.php
221	$location = stripslashes($data[′location′]);	collision-testimonials.3.0/ collision-testimonials.php
222	if (!empty($location)){	collision-testimonials.3.0/ collision-testimonials.php
223	$location = ”, ”.$location;	collision-testimonials.3.0/ collision-testimonials.php
224	}	collision-testimonials.3.0/ collision-testimonials.php
225	$quote = stripslashes($data[′quote′]);	collision-testimonials.3.0/ collision-testimonials.php
228	$website = $data[′website′];	collision-testimonials.3.0/ collision-testimonials.php
229	if (!empty($website)){	collision-testimonials.3.0/ collision-testimonials.php
230	$websitepre = ′<a href=”′.$website.′”>′;	collision-testimonials.3.0/ collision-testimonials.php
232	}	collision-testimonials.3.0/ collision-testimonials.php
235	$testimonial_prefix = $wpdb->get_row(”SELECT value FROM $testimonials_settings WHERE name=′testimonial_prefix′”, ARRAY_A);	collision-testimonials.3.0/ collision-testimonials.php
236	$testimonial_prefix = $testimonial_prefix[′value′];	collision-testimonials.3.0/ collision-testimonials.php
238	$testimonial_suffix = $wpdb->get_row(”SELECT value FROM $testimonials_settings WHERE name=′testimonial_suffix′”, ARRAY_A);	collision-testimonials.3.0/ collision-testimonials.php
239	$testimonial_suffix = $testimonial_suffix[′value′];	collision-testimonials.3.0/ collision-testimonials.php
242	$theTestimonial = $testimonial_prefix . $quote . ′ — ′ . $websitepre . $name . $websitesuf . ′′ . $location . ”” . $testimonial_suffix;	collision-testimonials.3.0/ collision-testimonials.php
245	echo $theTestimonial;	collision-testimonials.3.0/ collision-testimonials.php

Name of the enclosing function:collision_testimonials($id=“”)
Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput reaches sensitive sink when function collision_testimonials() is called.

———————————————

RIPS’s decision: 1

D.3.2.2 Explanation

A stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . "testimonials"*** throughout the files in the plugin. In the file ***collision-testimonials.3.0/pages/testimonials.php***, an UPDATE command is performed. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. Back in *collision-testimonials.3.0/collision-testimonials.php*, as long as the updated *status* is neither 1 nor 2, the data is retrieved by the SELECT statement, and its contents are displayed (executed) by echo in line 245. ***$testimonials_settings*'s *value*** field suffers from a similar problem.

D.3.3 Slice #17

D.3.3.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #33
Sensitive sink

- Name: echo
- Line: 196
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST[′editQuote′])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST[′status′];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST[′name′];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST[′location′];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST[′quote′];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST[′website′];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST[′id′];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = ””;	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != ′http://′ ? ′http://′ : ′′) . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = ”UPDATE ” . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	” SET status=′” . $wpdb->escape($status) . ”′, name=′” . $wpdb->escape($name) . ”′, location=′” . $wpdb->escape($location) . ”′, quote=′” . $wpdb->escape($quote) . ”′, website=′” . $wpdb->escape($website_final) . ”′ WHERE id=′”. $id .”′”;	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
175	GET	$_GET['oldid']	0	collision-testimonials.3.0/pages/testimonials.php
177	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
175	variable	$oldid	$_GET['oldid']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
176	variable	$query	$oldid	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
177	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
179	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
182	function	stripslashes ($data['name'])	$data['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
182	variable	$name	stripslashes ($data['name'])	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
179	while ($data = mysql_fetch_array($results)) {	1	170	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
170	if (($_GET['action'] == “edit”) && (!isset($_POST['editQuote']))){	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET[′action′] == ”edit”) && (!isset($_POST[′editQuote′]))){	collision-testimonials.3.0/ pages/testimonials.php
175	$oldid = $_GET[′oldid′];	collision-testimonials.3.0/ pages/testimonials.php
176	$query = ”SELECT * FROM $testimonials WHERE id=′$oldid′”;	collision-testimonials.3.0/ pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
182	$name = stripslashes($data[′name′]);	collision-testimonials.3.0/ pages/testimonials.php
196	<input type=”text” id=”name” name=”name” class=”regular-text” value=”<?php echo $name; ?>” />	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-

Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.3.2 Explanation

A stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . "testimonials"*** throughout the files in the plugin. In the same file, an UPDATE command is performed on records that contain a certain *$id* value. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. As long as *$oldid* matches the value used on the UPDATE, the data is retrieved by the SELECT statement, and its contents are displayed (executed) by echo in line 196.

D.3.4 Slice #18

D.3.4.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #34
Sensitive sink

- Name: echo
- Line: 202
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/ pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/ pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/ pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/ pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/ pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/ pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/ pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/ pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/ pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http://' ? 'http://' : '') . $website;	collision-testimonials.3.0/ pages/testimonials.php
94	}	collision-testimonials.3.0/ pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/ pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/ pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/ pages/testimonials.php
101	}	collision-testimonials.3.0/ pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/ pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	R S D	File
175	GET	$_GET['oldid']	0	collision-testimonials.3.0/pages/testimonials.php
177	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
175	variable	$oldid	$_GET['oldid']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
176	variable	$query	$oldid	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
177	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
179	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
183	function	stripslashes ($data['location'])	$data['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
183	variable	$location	stripslashes ($data['location'])	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
179	while ($data = mysql_fetch_array($results)) {	1	170	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
170	if (($_GET['action'] == "edit") && (!isset($_POST['editQuote']))){	1	0	1	collision-testimonials.3.0/pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	1	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET['action'] == "edit") && (!isset($_POST['editQuote']))){	collision-testimonials.3.0/pages/testimonials.php
175	$oldid = $_GET['oldid'];	collision-testimonials.3.0/pages/testimonials.php
176	$query = "SELECT * FROM $testimonials WHERE id='$oldid'";	collision-testimonials.3.0/pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
183	$location = stripslashes($data['location']);	collision-testimonials.3.0/pages/testimonials.php
202	<input type="text" id="location" name="location" class="regular-text" value="<?php echo $location; ?>" />	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1

D.3.4.2 Explanation

A stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . "testimonials"*** throughout the files in the plugin. In the same file, an UPDATE command is performed on records that contain a certain *$id* value. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. As long as *$oldid* matches the value used on the UPDATE, the data is retrieved by the SELECT statement, and its contents are displayed (executed) by echo in line 202.

D.3.5 Slice #19

D.3.5.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #35
Sensitive sink

- Name: echo
- Line: 208
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http://' ? 'http://' : '') . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
175	GET	$_GET['oldid']	0	collision-testimonials.3.0/pages/testimonials.php
177	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
175	variable	$oldid	$_GET['oldid']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
176	variable	$query	$oldid	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
177	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
179	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
184	function	stripslashes ($data['quote'])	$data['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
184	variable	$quote	stripslashes ($data['quote'])	1	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
179	while ($data = mysql_fetch_array($results)) {	1	170	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
170	if (($_GET['action'] == "edit") && (!isset($_POST['editQuote']))){	1	0	1	collision-testimonials.3.0/pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	1	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET[′action′] == "edit") && (!isset($_POST[′editQuote′]))){	collision-testimonials.3.0/pages/testimonials.php
175	$oldid = $_GET[′oldid′];	collision-testimonials.3.0/pages/testimonials.php
176	$query = "SELECT * FROM $testimonials WHERE id=′$oldid′";	collision-testimonials.3.0/pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
184	$quote = stripslashes($data[′quote′]);	collision-testimonials.3.0/pages/testimonials.php
208	<textarea class="large-text code" id="quote" cols="50" rows="3" name="quote"><?php echo $quote; ?></textarea>	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1

D.3.5.2 Explanation

A stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . "testimonials"*** throughout the files in the plugin. In the same file, an UPDATE command is performed on records that contain a certain *$id* value. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. As long as *$oldid* matches the value used on the UPDATE, the data is retrieved by the SELECT statement, and its contents are displayed (executed) by echo in line 208.

———————————————

D.3.6 Slice #20

D.3.6.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #36
Sensitive sink

- Name: echo
- Line: 214
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http://' ? 'http://' : '') . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
175	GET	$_GET['oldid']	0	collision-testimonials.3.0/pages/testimonials.php
177	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
175	variable	$oldid	$_GET['oldid']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
176	variable	$query	$oldid	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
177	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
179	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
185	variable	$website	$data['website']	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
179	while ($data = mysql_fetch_array($results)) {	1	170	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
170	if (($_GET['action'] == “edit”) && (!isset($_POST['editQuote'])))){	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET[′action′] == ”edit”) && (!isset($_POST[′editQuote′])))){	collision-testimonials.3.0/pages/testimonials.php
175	$oldid = $_GET[′oldid′];	collision-testimonials.3.0/pages/testimonials.php
176	$query = ”SELECT * FROM $testimonials WHERE id=′$oldid′”;	collision-testimonials.3.0/pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
185	$website = $data[′website′];	collision-testimonials.3.0/pages/testimonials.php
214	<input type=”text” id=”website” name=”website” class=”regular-text” value=”<?php echo $website; ?>” />	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————

Name of the enclosing function: No function
RIPS's decision: 1

D.3.6.2 Explanation

A stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . "testimonials"*** throughout the files in the plugin. In the same file, an UPDATE command is performed on records that contain a certain *$id* value. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. As long as *$oldid* matches the value used on the UPDATE, the data is retrieved by the SELECT statement, and its contents are displayed (executed) by echo in line 214.

D.3.7 Slice #21

D.3.7.1 Verdict: 0 (not vulnerable)

RIPS XSS Slice #37
Sensitive sink

- Name: echo
- Line: 229
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
None of the writes affect the *id* field.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
175	GET	$_GET['oldid']	0	collision-testimonials.3.0/pages/testimonials.php
177	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
175	variable	$oldid	$_GET['oldid']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
176	variable	$query	$oldid	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
177	variable	$results	$query	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
179	variable	$data	$results	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
180	variable	$id	$data['id']	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
179	while ($data = mysql_fetch_array($results)) {	1	170	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
170	if (($_GET['action'] == “edit”) && (!isset($_POST['editQuote']))){	1	0	1	collision-testimonials.3.0/ pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	1	1	1	collision-testimonials.3.0/ pages/testimonials.php

Slice:

L	Instruction	File
170	if (($_GET[′action′] == ”edit”) && (!isset($_POST[′editQuote′]))){	collision-testimonials.3.0/pages/testimonials.php
175	$oldid = $_GET[′oldid′];	collision-testimonials.3.0/pages/testimonials.php
176	$query = ”SELECT * FROM $testimonials WHERE id=′$oldid′”;	collision-testimonials.3.0/pages/testimonials.php
177	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
179	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
180	$id = $data[′id′];	collision-testimonials.3.0/pages/testimonials.php
229	<input type=”hidden” value=”<?php echo $id; ?>” name=”id” />	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS’s decision: 1

D.3.7.2 Explanation

False positive! None of the writes affect the *id* field.

D.3.8 Slice #22

D.3.8.1 Verdict: 0 (not vulnerable)

RIPS XSS Slice #40
Sensitive sink

- Name: echo
- Line: 353
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
None of the writes affect the *id* field.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
336	variable	$id	$data['id']	0	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
336	$id = $data['id'];	collision-testimonials.3.0/pages/testimonials.php
353	<input type="checkbox" value="<?php echo $id; ?>" class="administrator" id="<?php echo $id; ?>" name="<?php echo $id; ?>"/>	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.8.2 Explanation

False positive! None of the writes affect the *id* field.

D.3.9 Slice #23

D.3.9.1 Verdict: 0 (not vulnerable)

RIPS XSS Slice #41
Sensitive sink

- Name: echo
- Line: 356
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
None of the writes affect the *id* field.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/pages/testimonials.php
336	variable	$id	$data['id']	0	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = ”SELECT * FROM $testimonials ORDER BY $sort $dir”;	collision-testimonials.3.0/pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
336	$id = $data[′id′];	collision-testimonials.3.0/pages/testimonials.php
356	<?php echo $id; ?>	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS’s decision: 1

D.3.9.2 Explanation

False positive! None of the writes affect the *id* field.

D.3.10 Slice #24

D.3.10.1 Verdict: 0 (not vulnerable)

RIPS XSS Slice #42
Sensitive sink

- Name: echo
- Line: 359

- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
None of the writes affect the *id* field.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
336	variable	$id	$data['id']	0	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php
359	$featured == 1	1	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
336	$id = $data['id'];	collision-testimonials.3.0/ pages/testimonials.php
359	<?php if ($featured == 1){ ?><a href="?page=testimonials&unfeatQuote&id=<?php echo $id; ?>" class="feat">*	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.10.2 Explanation

False positive! None of the writes affect the *id* field.

D.3.11 Slice #25

D.3.11.1 Verdict: 0 (not vulnerable)

RIPS XSS Slice #43
Sensitive sink

- Name: echo
- Line: 360
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
None of the writes affect the *id* field.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
336	variable	$id	$data['id']	0	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php
360	if ($featured == 0)	1	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = ”SELECT * FROM $testimonials ORDER BY $sort $dir”;	collision-testimonials.3.0/pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
336	$id = $data[′id′];	collision-testimonials.3.0/pages/testimonials.php
360	<?php } else if ($featured == 0){ ?><a href=”?page=testimonials&featQuote&id=<?php echo $id; ?>” class=”unfeat”>*<?php } ?>	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS’s decision: 1

D.3.11.2 Explanation

False positive! None of the writes affect the *id* field.

D.3.12 Slice #26

D.3.12.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #44
Sensitive sink

- Name: echo

- Line: 363
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/ pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/ pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/ pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/ pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/ pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/ pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/ pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/ pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/ pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http://' ? 'http://' : '') . $website;	collision-testimonials.3.0/ pages/testimonials.php
94	}	collision-testimonials.3.0/ pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/ pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/ pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/ pages/testimonials.php
101	}	collision-testimonials.3.0/ pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/ pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
346	function	stripslashes ($data['name'])	$data['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
346	variable	$name	stripslashes ($data['name'])	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = ”SELECT * FROM $testimonials ORDER BY $sort $dir”;	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
346	$name = stripslashes($data[′name′]);	collision-testimonials.3.0/ pages/testimonials.php
363	<?php echo $name; ?>	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS’s decision: 1

D.3.12.2 Explanation

A stored XSS vulnerability occurs. ***$testimonials*** contains the value ***$wpdb->prefix . “testimonials”*** throughout the files in the plugin. In the same file, an UPDATE command is performed. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. This data is retrieved by the SELECT command, and its contents are displayed (executed) by echo in line 363.

D.3.13 Slice #27

D.3.13.1 Verdict: 0 (not vulnerable)

RIPS XSS Slice #45
Sensitive sink

- Name: echo

- Line: 364
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
None of the writes affect the *id* field.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
336	function	$id	$data['id']	0	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
336	$id = $data['id'];	collision-testimonials.3.0/ pages/testimonials.php
364	<a href="?page=testimonials&action=edit&oldid=<?php echo $id; ?>" title="Edit this post">Edit | <a class='submitdelete' title='Delete this testimonial' href='?page=testimonials&deleteQuote&id=<?php echo $id; ?>' onclick="if (confirm('You are about to delete a testimonial by \'<?php echo $name; ?>\'\n \'Cancel\' to stop, \'OK\' to delete.')) { return true;}return false;">Delete	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1

D.3.13.2 Explanation

False positive! None of the writes affect the *id* field.

D.3.14 Slice #28

D.3.14.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #46
Sensitive sink

- Name: echo
- Line: 364
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	R S D	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST[′editQuote′])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST[′status′];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST[′name′];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST[′location′];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST[′quote′];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST[′website′];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST[′id′];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = ””;	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != ′http://′ ? ′http://′ : ′′) . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = ”UPDATE ” . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	” SET status=′” . $wpdb->escape($status) . ”′, name=′” . $wpdb->escape($name) . ”′, location=′” . $wpdb->escape($location) . ”′, quote=′” . $wpdb->escape($quote) . ”′, website=′” . $wpdb->escape($website_final) . ”′ WHERE id=′”. $id .”′”;	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function

READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
346	function	stripslashes ($data['name'])	$data['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
346	variable	$name	stripslashes ($data['name'])	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
346	$name = stripslashes($data['name']);	collision-testimonials.3.0/ pages/testimonials.php
364	<a href="?page=testimonials&action=edit&oldid=<?php echo $id; ?>" title="Edit this post">Edit | <a class='submitdelete' title='Delete this testimonial' href='?page=testimonials&deleteQuote&id=<?php echo $id; ?>' onclick="if (confirm('You are about to delete a testimonial by \'<?php echo $name; ?>\'\n \'Cancel\' to stop, \'OK\' to delete.')) { return true;}return false;">Delete	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.14.2 Explanation

A stored XSS vulnerability occurs. $testimonials contains the value $wpdb->prefix . "testimonials" throughout the files in the plugin. In the same file, an UPDATE command is performed. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. This data is retrieved by the SELECT command, and its contents are displayed (executed) by echo in line 364.

D.3.15 Slice #29

D.3.15.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #47
Sensitive sink

- Name: echo
- Line: 367
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	R S D	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST[′editQuote′])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST[′status′];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST[′name′];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST[′location′];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST[′quote′];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST[′website′];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST[′id′];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = ””;	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != ′http://′ ? ′http://′ : ′′) . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = ”UPDATE ” . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	” SET status=′” . $wpdb->escape($status) . ”′, name=′” . $wpdb->escape($name) . ”′, location=′” . $wpdb->escape($location) . ”′, quote=′” . $wpdb->escape($quote) . ”′, website=′” . $wpdb->escape($website_final) . ”′ WHERE id=′”. $id .”′”;	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	Nt d	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
347	function	stripslashes ($data['location'])	$data['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
347	variable	$location	stripslashes ($data['location'])	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	Nt d	T / F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
347	$location = stripslashes($data['location']);	collision-testimonials.3.0/ pages/testimonials.php
367	<?php echo $location; ?>	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.15.2 Explanation

A stored XSS vulnerability occurs. $testimonials contains the value $wpdb->prefix . "testimonials" throughout the files in the plugin. In the same file, an UPDATE command is performed. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. This data is retrieved by the SELECT command, and its contents are displayed (executed) by echo in line 367.

D.3.16 Slice #30

D.3.16.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #48
Sensitive sink

- Name: echo
- Line: 370
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote']))	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http:// ? 'http:// : '') . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/pages/testimonials.php
348	function	stripslashes ($data['quote'])	$data['quote']	1	1	0	0	collision-testimonials.3.0/pages/testimonials.php
348	variable	$quote	stripslashes ($data['quote'])	1	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = ”SELECT * FROM $testimonials ORDER BY $sort $dir”;	collision-testimonials.3.0/pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
348	$quote = stripslashes($data[′quote′]);	collision-testimonials.3.0/pages/testimonials.php
370	<?php echo $quote; ?>	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.16.2 Explanation

A stored XSS vulnerability occurs. $testimonials contains the value $wpdb->prefix . "testimonials" throughout the files in the plugin. In the same file, an UPDATE command is performed. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. This data is retrieved by the SELECT command, and its contents are displayed (executed) by echo in line 370.

D.3.17 Slice #31

D.3.17.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #49
Sensitive sink

- Name: echo
- Line: 373
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	R S D	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST[′editQuote′])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST[′status′];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST[′name′];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST[′location′];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST[′quote′];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST[′website′];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST[′id′];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = ””;	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != ′http://′ ? ′http://′ : ′′) . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = ”UPDATE ” . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	” SET status=′” . $wpdb->escape($status) . ”′, name=′” . $wpdb->escape($name) . ”′, location=′” . $wpdb->escape($location) . ”′, quote=′” . $wpdb->escape($quote) . ”′, website=′” . $wpdb->escape($website_final) . ”′ WHERE id=′”. $id .”′”;	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
333	variable	$results	mysql_query($query)	0	1	0	0	collision-testimonials.3.0/pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/pages/testimonials.php
349	variable	$website	$data['website']	1	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/pages/testimonials.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/pages/testimonials.php
349	$website = $data['website'];	collision-testimonials.3.0/pages/testimonials.php
373	<a target="_blank" href="<?php echo $website; ?>"><?php echo str_replace("http://", "", $website); ?>	collision-testimonials.3.0/pages/testimonials.php

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.17.2 Explanation

A stored XSS vulnerability occurs. $testimonials contains the value $wpdb->prefix . "testimonials" throughout the files in the plugin. In the same file, an UPDATE command is performed. $wpdb->escape() applies addslashes() to its arguments, which is not enough to prevent the injection of malicious data in the table. This data is retrieved by the SELECT command, and its contents are displayed (executed) by echo in line 373.

D.3.18 Slice #32

D.3.18.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #50
Sensitive sink

- Name: echo
- Line: 373
- File: collision-testimonials.3.0/pages/testimonials.php

WRITES (1st step)
Entry points:

L	Type	Name	R S D	File
84	POST	$_POST['status']	0	collision-testimonials.3.0/pages/testimonials.php
85	POST	$_POST['name']	0	collision-testimonials.3.0/pages/testimonials.php
86	POST	$_POST['location']	0	collision-testimonials.3.0/pages/testimonials.php
87	POST	$_POST['quote']	0	collision-testimonials.3.0/pages/testimonials.php
88	POST	$_POST['website']	0	collision-testimonials.3.0/pages/testimonials.php
89	POST	$_POST['id']	0	collision-testimonials.3.0/pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
84	variable	$status	$_POST['status']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
85	variable	$name	$_POST['name']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
86	variable	$location	$_POST['location']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
87	variable	$quote	$_POST['quote']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
88	variable	$website	$_POST['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
89	variable	$id	$_POST['id']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
93	variable	$website_final	$website	!empty($website)	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
98	variable	$update	$id	!empty($name) && !empty($quote)	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
92	!empty($website)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php
96	!empty($name) && !empty($quote)	0	83	1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
83	isset($_POST['editQuote'])	1	0	1	collision-testimonials.3.0/pages/testimonials.php
92	!empty($website)	0	1	1	collision-testimonials.3.0/pages/testimonials.php
96	!empty($name) && !empty($quote)	0	1	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
83	if (isset($_POST['editQuote'])) {	collision-testimonials.3.0/pages/testimonials.php
84	$status = $_POST['status'];	collision-testimonials.3.0/pages/testimonials.php
85	$name = $_POST['name'];	collision-testimonials.3.0/pages/testimonials.php
86	$location = $_POST['location'];	collision-testimonials.3.0/pages/testimonials.php
87	$quote = $_POST['quote'];	collision-testimonials.3.0/pages/testimonials.php
88	$website = $_POST['website'];	collision-testimonials.3.0/pages/testimonials.php
89	$id = $_POST['id'];	collision-testimonials.3.0/pages/testimonials.php
91	$website_final = "";	collision-testimonials.3.0/pages/testimonials.php
92	if (!empty($website)) {	collision-testimonials.3.0/pages/testimonials.php
93	$website_final = (substr(ltrim($website), 0, 7) != 'http:// ? 'http:// : '') . $website;	collision-testimonials.3.0/pages/testimonials.php
94	}	collision-testimonials.3.0/pages/testimonials.php
96	if (!empty($name) && !empty($quote)){	collision-testimonials.3.0/pages/testimonials.php
98	$update = "UPDATE " . $testimonials .	collision-testimonials.3.0/pages/testimonials.php
99	" SET status='" . $wpdb->escape($status) . "', name='" . $wpdb->escape($name) . "', location='" . $wpdb->escape($location) . "', quote='" . $wpdb->escape($quote) . "', website='" . $wpdb->escape($website_final) . "' WHERE id='". $id ."'";	collision-testimonials.3.0/pages/testimonials.php
101	}	collision-testimonials.3.0/pages/testimonials.php
103	$results = $wpdb->query($update);	collision-testimonials.3.0/pages/testimonials.php

Name of the enclosing function: No function
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
333	BD	mysql_query($query)	0	collision-testimonials.3.0/ pages/testimonials.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	RSD	File
333	variable	$results	mysql_query ($query)	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
335	variable	$data	$results	0	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
349	variable	$website	$data['website']	1	1	0	0	collision-testimonials.3.0/ pages/testimonials.php
373	function	str_replace ("http://", "", $website)	$website	1	1	0	1	collision-testimonials.3.0/ pages/testimonials.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
335	while ($data = mysql_fetch_array($results))	1		1	collision-testimonials.3.0/ pages/testimonials.php

Execution path constraints

L	Condition	ES	N t d	T / F	File
335	while ($data = mysql_fetch_array($results))	1	0	1	collision-testimonials.3.0/pages/testimonials.php

Slice:

L	Instruction	File
332	$query = "SELECT * FROM $testimonials ORDER BY $sort $dir";	collision-testimonials.3.0/ pages/testimonials.php
333	$results = mysql_query($query);	collision-testimonials.3.0/ pages/testimonials.php
335	while ($data = mysql_fetch_array($results)) {	collision-testimonials.3.0/ pages/testimonials.php
349	$website = $data['website'];	collision-testimonials.3.0/ pages/testimonials.php
373	<a target="_blank" href="<?php echo $website; ?>"><?php echo str_replace("http://", "", $website); ?>	collision-testimonials.3.0/ pages/testimonials.php

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput returned by function mysql_fetch_array() reaches sensitive sink.

Name of the enclosing function: No function
RIPS's decision: 1

D.3.18.2 Explanation

A Stored XSS vulnerability occurs. *$testimonials* contains the value *$wpdb->prefix . "testimonials"* throughout the files in the plugin. In the same file, an *UPDATE* command is performed. *$wpdb->escape()* applies *addslashes()* to its arguments, which is not enough to prevent the injection of malicious data in the table. This data is retrieved by the *SELECT* command, and its contents are displayed (executed) by *echo* in line 373 (*str_replace* simply removes the protocol prefix, the variable remains tainted).

D.4 VV

- The entry point comes through a native PHP variable for dealing with HTTP requests: *$_GET*, *$_POST*. **The entry point may also come as a function parameter**. **If the variable is an array, we will assume at least one of the elements is tainted.**

- Taintedness is propagated into a loop construct and each of its iterations iff the entity being iterated upon is tainted AND at least one item of the entity being iterated upon is accessed.

D.5 WAP false negatives

- Slice #10 shows that, if multiple branches are preceding the sink, **and share a common variable**, only the last one is considered. Since no taintedness is propagated into the last branch, wap does not detect the vulnerability **even if previous branches were, indeed, tainted. This is a limitation concerning the Symbol Table structure.**

- **Multi-line concatentation strings** usually present a problem for WAP. It seems the resulting string is cut off at some treshold, causing some sinks to be partially skipped, when the tainted input appears on the part that is cut off.

- wap-2.1 classifies real vulnerabilities as false positives if *str_replace*[54] is used on a tainted input. However, the existence of *str_replace*, alone, is not enough to

block taintedness.

D.6 Misc.

- **Running wap-2.1 with this plugin (option *-p*) will not return any vulnerabilities**. Running it with individual files returns vulnerabilities as expected.
- It is been proven that *addslashes* is not a very useful function for XSS protection. The function only escapes a very limited range of characters [46]. It is missing crucial characters such as '<' or '>', making the examples in [66] and [68] successful. Wordpress provides better functions, such as *wp_kses* or *esc_**[78].
- Reflected XSS can be induced indirectly through an SQL injection by using the union operator. The union part of the injection must match the number of fields in the normal query [40]. A limitation to this attack occurs if *ORDER BY* is used before *UNION*, as we get the following error:
  Incorrect usage of UNION and ORDER BY
  The only way around this limitation is to intentionally trigger a SQL syntax error, where the error report contains (and therefore, executes) the malicious code (error reporting needs to be activated).
- I could not reproduce the first two examples of [40],[33], which are related to ORDER BY injections. Nevertheless, I propose we do not let any untrusted input participate in a select statement.
- In Vulnerability #8, wap considers line 156's *$_GET['id']* to be the entry point, which is incorrect. The real entry point, in this situation, consists in an array of values provided in a POST request. This seems to be a problem on the presentation of results.

Appendix E

Plugin: community-events.1.2.9

E.1 Type: SQLI

E.1.1 Slice #1

E.1.1.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #16
Sensitive sink

- Name: $wpdb->query
- Line: 381
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	R S D	File
381	GET	$_GET['deletecat']	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
375	isset($_GET['deletecat'])	1	365	1	community-events.1.2.9/community-events.php
379	$catexist	1	375	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
365	$_GET['page'] == 'community-events-event-types'	1	0	1	community-events.1.2.9/community-events.php
375	isset($_GET['deletecat'])	1	1	1	community-events.1.2.9/community-events.php
379	$catexist	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
352	function on_show_page() {	community-events.1.2.9/community-events.php
365	elseif ($_GET[′page′] == ′community-events-event-types′)	community-events.1.2.9/community-events.php
366	{	community-events.1.2.9/community-events.php
375	elseif (isset($_GET[′deletecat′]))	community-events.1.2.9/community-events.php
376	{	community-events.1.2.9/community-events.php
379	if ($catexist)	community-events.1.2.9/community-events.php
380	{	community-events.1.2.9/community-events.php
381	$wpdb->query("DELETE from " . $wpdb->prefix . "ce_category WHERE id = " . $_GET[′deletecat′]);	community-events.1.2.9/community-events.php

Misc. (enclosing modules & functions)

- Name of the enclosing function:on_show_page

phpSAFE's decision: 1

E.1.1.2 Explanation

Tainted input reaches sensitive sink without sanitization.

E.1.2 Slice #2

E.1.2.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #17
Sensitive sink

- Name: $wpdb->query
- Line: 407
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	RSD	File
407	GET	$_GET['deletevenue']	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
401	isset($_GET['deletevenue'])	1	391	1	community-events.1.2.9/community-events.php
405	$venueexist	1	401	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
391	$_GET['page'] == 'community-events-venues'	1	0	1	community-events.1.2.9/community-events.php
401	isset($_GET['deletevenue'])	1	1	1	community-events.1.2.9/community-events.php
405	$venueexist	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
352	function on_show_page() {	community-events.1.2.9/community-events.php
391	elseif ($_GET['page'] == 'community-events-venues')	community-events.1.2.9/community-events.php
392	{	community-events.1.2.9/community-events.php
401	elseif (isset($_GET['deletevenue']))	community-events.1.2.9/community-events.php
402	{	community-events.1.2.9/community-events.php
405	if ($venueexist)	community-events.1.2.9/community-events.php
406	{	community-events.1.2.9/community-events.php
407	$wpdb->query("DELETE from " . $wpdb->prefix ."ce_venues WHERE ce_venue_id = " . $_GET['deletevenue']);	community-events.1.2.9/community-events.php

Misc. (enclosing modules & functions)

- Name of the enclosing function:on_show_page

phpSAFE's decision: 1

E.1.2.2 Explanation

Tainted input reaches sensitive sink without sanitization.

E.1.3 Slice #3

E.1.3.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #18
Sensitive sink

- Name: $wpdb->query

- Line: 454
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	R S D	File
454	GET	$_GET['deleteevent']	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
448	isset($_GET['deleteevent'])	1	438	1	community-events.1.2.9/community-events.php
452	$eventexist	1	448	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
438	$_GET['page'] == 'community-events-events'	1	0	1	community-events.1.2.9/community-events.php
448	isset($_GET['deleteevent'])	1	1	1	community-events.1.2.9/community-events.php
452	$eventexist	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
352	function on_show_page() {	community-events.1.2.9/community-events.php
438	elseif ($_GET[′page′] == ′community-events-events′)	community-events.1.2.9/community-events.php
439	{	community-events.1.2.9/community-events.php
448	elseif (isset($_GET[′deleteevent′]))	community-events.1.2.9/community-events.php
449	{	community-events.1.2.9/community-events.php
452	if ($eventexist)	community-events.1.2.9/community-events.php
453	{	community-events.1.2.9/community-events.php
454	$wpdb->query("DELETE from " . $wpdb->prefix . "ce_events WHERE event_id = " . $_GET[′deleteevent′]);	community-events.1.2.9/community-events.php

Misc. (enclosing modules & functions)

- Name of the enclosing function:on_show_page

phpSAFE's decision: 1

E.1.3.2 Explanation

Tainted input reaches sensitive sink without sanitization.

E.2 Type: Stored injection

E.2.1 Slice #4

E.2.1.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2699
Sensitive sink

- Name: echo
- Line: 1323
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1304	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
1304	variable	$venues	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1319	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1323	variable	$venue->nbitems	$venue	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1306	$venues	1		1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1306	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1306	$venues	1	0	1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1260	function event_venues_meta_box($data) {	community-events.1.2.9/community-events.php
1304	<?php $venues = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	community-events.1.2.9/community-events.php
1306	if ($venues): ?>	community-events.1.2.9/community-events.php
1319	<?php foreach($venues as $venue): ?>	community-events.1.2.9/community-events.php
1323	<td style='background: #FFF;text-align:right'><?php echo $venue->nbitems; ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_venues_meta_box($data)
phpSAFE's decision: 1

E.2.1.2 Explanation

Not vulnerable. *$venue->nbitems* is an alias for *count(e.event_id)*, as seen in line 1304. It is simply a counter which poses no threat.

E.2.2 Slice #5

E.2.2.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2700
Sensitive sink

- Name: echo
- Line: 1325
- File: community-events.1.2.9/community-events.php

WRITES (1st step)

None of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table.

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
1304	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	Ntd	RSD	File
1304	variable	$venues	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1319	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1325	variable	$venue->ce_venue_id	$venue	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1306	$venues	1		1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1306	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1306	$venues	1	0	1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1	1	community-events.1.2.9/community-events.php
1324	$venue->nbitems == 0	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1260	function event_venues_meta_box($data) {	community-events.1.2.9/community-events.php
1304	<?php $venues = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	community-events.1.2.9/community-events.php
1306	if ($venues): ?>	community-events.1.2.9/community-events.php
1319	<?php foreach($venues as $venue): ?>	community-events.1.2.9/community-events.php
1324	<?php if ($venue->nbitems == 0): ?>	community-events.1.2.9/community-events.php
1325	<td style='background:#FFF'><a href='admin.php?page=community-events-venues&deletevenue=<?php echo $venue->ce_venue_id; ?>'	community-events.1.2.9/community-events.php

Name of the enclosing function:event_venues_meta_box($data)
phpSAFE's decision: 1

E.2.2.2 Explanation

Not vulnerable. None of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table.

E.2.3 Slice #6

E.2.3.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2711
Sensitive sink

- Name: echo
- Line: 1472
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
1463	BD	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_category ORDER by event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
1463	variable	$cats	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_category ORDER by event_cat_name");	0	1	0	0	community-events.1.2.9/community-events.php
1465	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1472	variable	$cat->event_cat_id	$cat	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
1465	foreach ($cats as $cat)	1		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
1465	foreach ($cats as $cat)	1	0	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1377	function events_meta_box($data) {	community-events.1.2.9/community-events.php
1463	<?php $cats = $wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_category ORDER by event_cat_name");	community-events.1.2.9/community-events.php
1465	foreach ($cats as $cat)	community-events.1.2.9/community-events.php
1466	{	community-events.1.2.9/community-events.php
1472	echo "<option value='" . $cat->event_cat_id . "' " . $selected-string . ">" . stripslashes($cat->event_cat_name) . "\n";	community-events.1.2.9/community-events.php

Name of the enclosing function:events_meta_box($data)
phpSAFE's decision: 1

E.2.3.2 Explanation

Vulnerable. Even though none of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table, a vulnerability occurs due to *$cat->event_cat_name* (see Slice #7).

E.2.4 Slice #7

E.2.4.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2712
Sensitive sink

- Name: echo
- Line: 1472
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
581	POST	$_POST['name']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	Nt d	RSD	File
581	function	array("event_cat_name" => $_POST['name'])	$_POST['name']	1	1	0	0	community-events .1.2.9/ community-events.php
581	variable	$newcat	array("event_cat_name" => $_POST['name'])	1	1	0	1	community-events .1.2.9/ community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
580	isset($_POST['name'])	0	579	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Nt d	T / F	File
579	isset($_POST['newcat']) \|\| isset($_POST['updatecat'])	1	0	1	community-events.1.2.9/community-events.php
580	isset($_POST['name'])	0	1	1	community-events.1.2.9/community-events.php
588	isset($_POST['newcat'])	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
569	function on_save_changes_eventtypes() {	community-events.1.2.9/community-events.php
579	if (isset($_POST['newcat']) \|\| isset($_POST['updatecat'])) {	community-events.1.2.9/community-events.php
580	if (isset($_POST['name']))	community-events.1.2.9/community-events.php
581	$newcat = array("event_cat_name" => $_POST['name']);	community-events.1.2.9/community-events.php
588	if (isset($_POST['newcat']))	community-events.1.2.9/community-events.php
589	{	community-events.1.2.9/community-events.php
590	$wpdb->insert($wpdb->prefix.'ce_category', $newcat);	community-events.1.2.9/community-events.php

Name of the enclosing function:on_save_changes_eventtypes()
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1463	BD	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_category ORDER by event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
1463	variable	$cats	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_category ORDER by event_cat_name");	0	1	0	0	community-events.1.2.9/community-events.php
1465	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1472	variable	$cat->event_cat_name	$cat	1	1	0	0	community-events.1.2.9/community-events.php
1472	function	stripslashes($cat->event_cat_name)	$cat->event_cat_name	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1465	foreach ($cats as $cat)	1		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1465	foreach ($cats as $cat)	1	0	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1377	function events_meta_box($data) {	community-events.1.2.9/community-events.php
1463	<?php $cats = $wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_category ORDER by event_cat_name");	community-events.1.2.9/community-events.php
1465	foreach ($cats as $cat)	community-events.1.2.9/community-events.php
1466	{	community-events.1.2.9/community-events.php
1472	echo "<option value='" . $cat->event_cat_id . "' " . $selectedstring . ">" . stripslashes($cat->event_cat_name) . "\n";	community-events.1.2.9/community-events.php

Name of the enclosing function:events_meta_box($data)
phpSAFE's decision: 1

E.2.4.2 Explanation

A vulnerability occurs because a tainted input is assigned to *event_cat_name* within an insert command on *$wpdb->prefix.ce_category*. The SELECT command retrieves all fields from the table, and the vulnerable *event_cat_name* value becomes an (unescaped) argument to echo, the sink of this vulnerability.

E.2.5 Slice #8

E.2.5.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2735
Sensitive sink

- Name: echo
- Line: 1488
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1479	BD	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_venues ORDER by ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	RSD	File
1479	variable	$venues	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_venues ORDER by ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1481	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1488	variable	$venue->ce_venue_id	$venue	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1481	foreach ($venues as $venue)	1		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1481	foreach ($venues as $venue)	1	0	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1377	function events_meta_box($data) {	community-events.1.2.9/community-events.php
1479	<?php $venues = $wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_venues ORDER by ce_venue_name");	community-events.1.2.9/community-events.php
1481	foreach ($venues as $venue)	community-events.1.2.9/community-events.php
1482	{	community-events.1.2.9/community-events.php
1488	echo "<option value='" . $venue->ce_venue_id . "' " . $selectedstring . ">" . stripslashes($venue->ce_venue_name) . "\n";	community-events.1.2.9/community-events.php

Name of the enclosing function:events_meta_box($data)
phpSAFE's decision: 1

E.2.5.2 Explanation

Vulnerable. Even though none of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table, *$venue->ce_venue_name* is vulnerable (see Slice #9).

E.2.6 Slice #9

E.2.6.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2736
Sensitive sink

- Name: echo
- Line: 1488
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
682	POST	$_POST['ce_venue_name']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	Nt d	RSD	File
682	function	array("ce_venue_name" => $_POST['ce_venue_name'], "ce_venue_address" => $_POST['ce_venue_address'], "ce_venue_city" => $_POST['ce_venue_city'], "ce_venue_zipcode" => $_POST['ce_venue_zipcode'], "ce_venue_phone" => $_POST['ce_venue_phone'], "ce_venue_email" => $_POST['ce_venue_email'], "ce_venue_url" => $_POST['ce_venue_url'])	$_POST['ce_venue_name']	1	1	0	0	community-events.1.2.9/community-events.php
682	variable	$newvenue	array("ce_venue_name" => $_POST['ce_venue_name'], "ce_venue_address" => $_POST['ce_venue_address'], "ce_venue_city" => $_POST['ce_venue_city'], "ce_venue_zipcode" => $_POST['ce_venue_zipcode'], "ce_venue_phone" => $_POST['ce_venue_phone'], "ce_venue_email" => $_POST['ce_venue_email'], "ce_venue_url" => $_POST['ce_venue_url'])	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
681	isset($_POST['ce_venue_name'])	0	679	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
679	isset($_POST['newvenue']) \|\| isset($_POST['updatevenue'])	1	0	1	community-events.1.2.9/community-events.php
681	isset($_POST['ce_venue_name'])	0	1	1	community-events.1.2.9/community-events.php
696	isset($_POST['newvenue'])	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
611	function on_save_changes_venues() {	community-events.1.2.9/community-events.php
679	elseif (isset($_POST['newvenue']) \|\| isset($_POST['updatevenue'])) {	community-events.1.2.9/community-events.php
681	if (isset($_POST['ce_venue_name']))	community-events.1.2.9/community-events.php
682	$newvenue = array("ce_venue_name" => $_POST['ce_venue_name'],	community-events.1.2.9/community-events.php
696	if (isset($_POST['newvenue']))	community-events.1.2.9/community-events.php
697	{	community-events.1.2.9/community-events.php
698	$wpdb->insert($wpdb->prefix.'ce_venues', $newvenue);	community-events.1.2.9/community-events.php

Name of the enclosing function:on_save_changes_venues()

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
1479	BD	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_venues ORDER by ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
1479	variable	$venues	$wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_venues ORDER by ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1481	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1488	variable	$venue->ce_venue_name	$venue	1	1	0	0	community-events.1.2.9/community-events.php
1488	function	stripslashes($venue->ce_venue_name)	$venue->ce_venue_name	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1481	foreach ($venues as $venue)	1		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	N t d	T/F	File
1481	foreach ($venues as $venue)	1	0	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1377	function events_meta_box($data) {	community-events.1.2.9/community-events.php
1479	<?php $venues = $wpdb->get_results("SELECT * from " . $wpdb->prefix. "ce_venues ORDER by ce_venue_name");	community-events.1.2.9/community-events.php
1481	foreach ($venues as $venue)	community-events.1.2.9/community-events.php
1482	{	community-events.1.2.9/community-events.php
1488	echo "<option value='" . $venue->ce_venue_id . "'" . $selected-string . ">" . stripslashes($venue->ce_venue_name) . "\n";	community-events.1.2.9/community-events.php

Name of the enclosing function:events_meta_box($data)
phpSAFE's decision: 1

E.2.6.2 Explanation

A vulnerability occurs because a tainted input is assigned to *ce_venue_name* within an insert command on *$wpdb->prefix.ce_venues*. The SELECT command retrieves all fields from the table, and the vulnerable *ce_venue_name* value becomes an (unescaped) argument to echo, the sink of this vulnerability.

E.2.7 Slice #10

E.2.7.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2799
Sensitive sink

- Name: echo
- Line: 1223
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
1206	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
1206	variable	$cats	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	1	0	0	community-events.1.2.9/community-events.php
1221	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1223	variable	$cat->event_cat_id	$cat	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
1208	$cats	1		1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1208	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
1208	$cats	1	0	1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1190	function event_types_meta_box($data) {	community-events.1.2.9/community-events.php
1206	<?php $cats = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	community-events.1.2.9/community-events.php
1208	if ($cats): ?>	community-events.1.2.9/community-events.php
1221	<?php foreach($cats as $cat): ?>	community-events.1.2.9/community-events.php
1223	<td class='name column-name' style='background: #FFF'><?php echo $cat->event_cat_id; ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_types_meta_box($data)
phpSAFE's decision: 1

E.2.7.2 Explanation

Not vulnerable. None of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table.

E.2.8 Slice #11

E.2.8.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2805
Sensitive sink

- Name: echo
- Line: 1224
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1206	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	RSD	File
1206	variable	$cats	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	1	0	0	community-events.1.2.9/community-events.php
1221	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1224	variable	$cat->event_cat_id	$cat	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1208	$cats	1		1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1208	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1208	$cats	1	0	1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1190	function event_types_meta_box($data) {	community-events.1.2.9/community-events.php
1206	<?php $cats = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	community-events.1.2.9/community-events.php
1208	if ($cats): ?>	community-events.1.2.9/community-events.php
1221	<?php foreach($cats as $cat): ?>	community-events.1.2.9/community-events.php
1224	<td style='background: #FFF'><a href='admin.php?page=community-events-event-types&editcat=<?php echo $cat->event_cat_id; ?>'><?php echo stripslashes($cat->event_cat_name); ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_types_meta_box($data)
phpSAFE's decision: 1

E.2.8.2 Explanation

Vulnerable. Even though none of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table, a vulnerability occurs due to *$cat->event_cat_name* (see Slice #12).

E.2.9 Slice #12

E.2.9.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2806
Sensitive sink

- Name: echo
- Line: 1224
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
581	POST	$_POST['name']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
581	function	array("event_cat_name" => $_POST['name'])	$_POST['name']	1	1	0	0	community-events.1.2.9/community-events.php
581	variable	$newcat	array("event_cat_name" => $_POST['name'])	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
580	isset($_POST['name'])	0	579	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
579	isset($_POST['newcat']) \|\| isset($_POST['updatecat'])	1	0	1	community-events.1.2.9/community-events.php
580	isset($_POST['name'])	0	1	1	community-events.1.2.9/community-events.php
588	isset($_POST['newcat'])	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
569	function on_save_changes_eventtypes() {	community-events.1.2.9/community-events.php
579	if (isset($_POST['newcat']) \|\| isset($_POST['updatecat'])) {	community-events.1.2.9/community-events.php
580	if (isset($_POST['name']))	community-events.1.2.9/community-events.php
581	$newcat = array("event_cat_name" => $_POST['name']);	community-events.1.2.9/community-events.php
588	if (isset($_POST['newcat']))	community-events.1.2.9/community-events.php
589	{	community-events.1.2.9/community-events.php
590	$wpdb->insert($wpdb->prefix.'ce_category', $newcat);	community-events.1.2.9/community-events.php

Name of the enclosing function:on_save_changes_eventtypes()

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
1206	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
1206	variable	$cats	$wpdb->get_results(“SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM ” . $wpdb->prefix . “ce_category c LEFT JOIN ” . $wpdb->prefix . “ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name”);	0	1	0	0	community-events.1.2.9/community-events.php
1221	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1224	variable	$cat->event_cat_name	$cat	1	1	0	0	community-events.1.2.9/community-events.php
1224	function	stripslashes($cat->event_cat_name)	$cat->event_cat_name	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
1208	$cats	1		1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1208	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
1208	$cats	1	0	1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1190	function event_types_meta_box($data) {	community-events.1.2.9/community-events.php
1206	<?php $cats = $wpdb->get_results(”SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM ” . $wpdb->prefix . ”ce_category c LEFT JOIN ” . $wpdb->prefix . ”ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name”);	community-events.1.2.9/community-events.php
1208	if ($cats): ?>	community-events.1.2.9/community-events.php
1221	<?php foreach($cats as $cat): ?>	community-events.1.2.9/community-events.php
1224	<td style=′background: #FFF′><a href=′admin.php?page=community-events-event-types&editcat=<?php echo $cat->event_cat_id; ?>′><?php echo stripslashes($cat->event_cat_name); ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_types_meta_box($data)
phpSAFE’s decision: 1

E.2.9.2 Explanation

A vulnerability occurs because a tainted input is assigned to *event_cat_name* within an insert command on *$wpdb->prefix.ce_category*. The SELECT command retrieves *event_cat_name* (among other fields) from the table, and the vulnerable *event_cat_name* value becomes an (unescaped) argument to echo, the sink of this vulnerability.

E.2.10 Slice #13

E.2.10.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2812
Sensitive sink

- Name: echo
- Line: 1225
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1206	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
1206	variable	$cats	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	1	0	0	community-events.1.2.9/community-events.php
1221	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1225	variable	$cat->nbitems	$cat	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1208	$cats	1		1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1208	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1208	$cats	1	0	1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1190	function event_types_meta_box($data) {	community-events.1.2.9/community-events.php
1206	<?php $cats = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	community-events.1.2.9/community-events.php
1208	if ($cats): ?>	community-events.1.2.9/community-events.php
1221	<?php foreach($cats as $cat): ?>	community-events.1.2.9/community-events.php
1225	<td style='background: #FFF;text-align:right'><?php echo $cat->nbitems; ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_types_meta_box($data)
phpSAFE's decision: 1

E.2.10.2 Explanation

Not vulnerable. *$cat->nbitems* is an alias for *count(e.event_id)*, as seen in line 1206. It is simply a counter which poses no threat.

E.2.11 Slice #14

E.2.11.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2819
Sensitive sink

- Name: echo
- Line: 1227
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *event_cat_id* field from the *$wpdb->prefix.ce_category* table.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1206	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
1206	variable	$cats	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	0	1	0	0	community-events.1.2.9/community-events.php
1221	variable	$cat	$cats	0	1	0	0	community-events.1.2.9/community-events.php
1227	variable	$cat->event_cat_id	$cat	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1208	$cats	1		1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1208	1	community-events.1.2.9/community-events.php
1226	$cat->nbitems == 0	1	1221	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1208	$cats	1	0	1	community-events.1.2.9/community-events.php
1221	foreach ($cats as $cat)	1	1	1	community-events.1.2.9/community-events.php
1226	$cat->nbitems == 0	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1190	function event_types_meta_box($data) {	community-events.1.2.9/community-events.php
1206	<?php $cats = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, c.event_cat_id, c.event_cat_name FROM " . $wpdb->prefix . "ce_category c LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_category = c.event_cat_id GROUP BY c.event_cat_name");	community-events.1.2.9/community-events.php
1208	if ($cats): ?>	community-events.1.2.9/community-events.php
1221	<?php foreach($cats as $cat): ?>	community-events.1.2.9/community-events.php
1226	<?php if ($cat->nbitems == 0): ?>	community-events.1.2.9/community-events.php
1227	<td style='background:#FFF'><a href='admin.php?page=community-events-event-types&deletecat=<?php echo $cat->event_cat_id; ?>'	community-events.1.2.9/community-events.php

Name of the enclosing function:event_types_meta_box($data)
phpSAFE's decision: 1

E.2.11.2 Explanation

Not vulnerable. None of the writes affect the event_cat_id field from the $wpdb->prefix.ce_category table.

E.2.12 Slice #15

E.2.12.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2820
Sensitive sink

- Name: echo
- Line: 1321
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1304	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
1304	variable	$venues	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1319	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1321	variable	$venue->ce_venue_id	$venue	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
1306	$venues	1		1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1306	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
1306	$venues	1	0	1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1260	function event_venues_meta_box($data) {	community-events.1.2.9/community-events.php
1304	<?php $venues = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	community-events.1.2.9/community-events.php
1306	if ($venues): ?>	community-events.1.2.9/community-events.php
1319	<?php foreach($venues as $venue): ?>	community-events.1.2.9/community-events.php
1321	<td class='name column-name' style='background: #FFF'><?php echo $venue->ce_venue_id; ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_venues_meta_box($data)
phpSAFE's decision: 1

E.2.12.2 Explanation

Not vulnerable. None of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table.

E.2.13 Slice #16

E.2.13.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2826
Sensitive sink

- Name: echo
- Line: 1322
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
None of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1304	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	Nt d	RSD	File
1304	variable	$venues	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1319	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1322	variable	$venue->ce_venue_id	$venue	0	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1306	$venues	1		1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1306	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
1306	$venues	1	0	1	community-events.1.2.9/community-events.php
1319	foreach($venues as $venue)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1260	function event_venues_meta_box($data) {	community-events.1.2.9/community-events.php
1304	<?php $venues = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	community-events.1.2.9/community-events.php
1306	if ($venues): ?>	community-events.1.2.9/community-events.php
1319	<?php foreach($venues as $venue): ?>	community-events.1.2.9/community-events.php
1322	<td style='background: #FFF'><a href='admin.php?page=community-events-venues&editvenue=<?php echo $venue->ce_venue_id; ?>'><?php echo stripslashes($venue->ce_venue_name); ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_venues_meta_box($data)
phpSAFE's decision: 1

E.2.13.2 Explanation

Vulnerable. Even though none of the writes affect the *ce_venue_id* field from the *$wpdb->prefix.ce_venues* table, *$venue->ce_venue_name* is vulnerable (see Slice #17).

E.2.14 Slice #17

E.2.14.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2827
Sensitive sink

- Name: echo
- Line: 1322
- File: community-events.1.2.9/community-events.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
682	POST	$_POST['ce_venue_name']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
682	function	array(“ce_venue_name” => $_POST[’ce_venue_name’], “ce_venue_address” => $_POST[’ce_venue_address’], “ce_venue_city” => $_POST[’ce_venue_city’], “ce_venue_zipcode” => $_POST[’ce_venue_zipcode’], “ce_venue_phone” => $_POST[’ce_venue_phone’], “ce_venue_email” => $_POST[’ce_venue_email’], “ce_venue_url” => $_POST[’ce_venue_url’])	$_POST[’ce_venue_name’]	1	1	0	0	community -events.1.2.9 /community -events.php
682	variable	$newvenue	array(“ce_venue_name” => $_POST[’ce_venue_name’], “ce_venue_address” => $_POST[’ce_venue_address’], “ce_venue_city” => $_POST[’ce_venue_city’], “ce_venue_zipcode” => $_POST[’ce_venue_zipcode’], “ce_venue_phone” => $_POST[’ce_venue_phone’], “ce_venue_email” => $_POST[’ce_venue_email’], “ce_venue_url” => $_POST[’ce_venue_url’])	1	1	0	1	community -events.1.2.9 /community -events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
681	isset($_POST[’ce_venue_name’])	0	679	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	N t d	T / F	File
679	isset($_POST[’newvenue’]) \|\| isset($_POST[’updatevenue’])	1	0	1	community-events.1.2.9/community-events.php
681	isset($_POST[’ce_venue_name’])	0	1	1	community-events.1.2.9/community-events.php
696	isset($_POST[’newvenue’])	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
611	function on_save_changes_venues() {	community-events.1.2.9/community-events.php
679	elseif (isset($_POST[′newvenue′]) \|\| isset($_POST[′updatevenue′])) {	community-events.1.2.9/community-events.php
681	if (isset($_POST[′ce_venue_name′]))	community-events.1.2.9/community-events.php
682	$newvenue = array(”ce_venue_name” => $_POST[′ce_venue_name′],	community-events.1.2.9/community-events.php
696	if (isset($_POST[′newvenue′]))	community-events.1.2.9/community-events.php
697	{	community-events.1.2.9/community-events.php
698	$wpdb->insert($wpdb->prefix.′ce_venues′, $newvenue);	community-events.1.2.9/community-events.php

Name of the enclosing function:on_save_changes_venues()
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
1304	BD	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	Ntd	RSD	File
1304	variable	$venues	$wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	0	1	0	0	community-events.1.2.9/community-events.php
1319	variable	$venue	$venues	0	1	0	0	community-events.1.2.9/community-events.php
1322	variable	$venue->ce_venue_name	$venue	1	1	0	0	community-events.1.2.9/community-events.php
1322	function	stripslashes($venue->ce_venue_name)	$venue->ce_venue_name	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1306	$venues	1		1	community-events.1.2.9/community-events.php
1319	foreach ($venues as $venue)	1	1306	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1306	$venues	1	0	1	community-events.1.2.9/community-events.php
1319	foreach ($venues as $venue)	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1260	function event_venues_meta_box($data) {	community-events.1.2.9/community-events.php
1304	<?php $venues = $wpdb->get_results("SELECT count(e.event_id) AS nbitems, v.ce_venue_id, v.ce_venue_name FROM " . $wpdb->prefix . "ce_venues v LEFT JOIN " . $wpdb->prefix . "ce_events e ON e.event_venue = v.ce_venue_id GROUP BY v.ce_venue_id ORDER by v.ce_venue_name");	community-events.1.2.9/community-events.php
1306	if ($venues): ?>	community-events.1.2.9/community-events.php
1319	<?php foreach($venues as $venue): ?>	community-events.1.2.9/community-events.php
1322	<td style='background: #FFF'><a href='admin.php?page=community-events-venues&editvenue=<?php echo $venue->ce_venue_id; ?>'><?php echo stripslashes($venue->ce_venue_name); ?></td>	community-events.1.2.9/community-events.php

Name of the enclosing function:event_venues_meta_box($data)
phpSAFE's decision: 1

E.2.14.2 Explanation

A vulnerability occurs because a tainted input is assigned to *ce_venue_name* within an insert command on *$wpdb->prefix.ce_venues*. The SELECT command retrieves *ce_venue_name* from the table (among other fields), and the vulnerable *ce_venue_name* value becomes an (unescaped) argument to echo, the sink of this vulnerability.

E.3 Type: XSS (reflected)

E.3.1 Slice #18

E.3.1.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2744
RIPS XSS Slice #51
Sensitive sink

- Name: echo
- Line: 432
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	RSD	File
432	GET	$_GET['importrowscount']	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
391	$_GET['page'] == 'community-events-venues'	1	0	1	community-events.1.2.9/community-events.php
412	isset($_GET['messages'])	1	1	1	community-events.1.2.9/community-events.php
416	foreach ($messagelist as $message)	1	1	1	community-events.1.2.9/community-events.php
418	switch($message)	1	1	1	community-events.1.2.9/community-events.php
431	case '9':	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
352	function on_show_page() {	community-events.1.2.9/community-events.php
391	elseif ($_GET['page'] == 'community-events-venues')	community-events.1.2.9/community-events.php
392	{	community-events.1.2.9/community-events.php
412	if (isset($_GET['messages']))	community-events.1.2.9/community-events.php
413	{	community-events.1.2.9/community-events.php
416	foreach ($messagelist as $message)	community-events.1.2.9/community-events.php
417	{	community-events.1.2.9/community-events.php
418	switch($message) {	community-events.1.2.9/community-events.php
431	case '9':	community-events.1.2.9/community-events.php
432	echo " " . $_GET['importrowscount'] . " " . __('row(s) found', 'link-library') . ". " . $_GET['successimportcount'] . " " . __('link(s) imported successfully', 'link-library') . ". ";	community-events.1.2.9/community-events.php

Additional RIPS results——-
Alternative trigger files:

- community-events.1.2.9/community-events.php
- community-events.1.2.9/get-events-admin.php
- community-events.1.2.9/get-events.php

Additional comment:
Userinput reaches sensitive sink when function on_show_page() is called.

———————————————

Name of the enclosing function:on_show_page()
RIPS's decision: 1
phpSAFE's decision: 1

E.3.1.2 Explanation

Tainted input reaches sensitive sink.

E.3.2 Slice #19

E.3.2.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2749
Sensitive sink

- Name: echo
- Line: 432
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	RSD	File
432	GET	$_GET['successimportcount']	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
391	$_GET['page'] == 'community-events-venues'	1	0	1	community-events.1.2.9/community-events.php
412	isset($_GET['messages'])	1	1	1	community-events.1.2.9/community-events.php
416	foreach ($messagelist as $message)	1	1	1	community-events.1.2.9/community-events.php
418	switch($message)	1	1	1	community-events.1.2.9/community-events.php
431	case '9':	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
352	function on_show_page() {	community-events.1.2.9/community-events.php
391	elseif ($_GET['page'] == 'community-events-venues')	community-events.1.2.9/community-events.php
392	{	community-events.1.2.9/community-events.php
412	if (isset($_GET['messages']))	community-events.1.2.9/community-events.php
413	{	community-events.1.2.9/community-events.php
416	foreach ($messagelist as $message)	community-events.1.2.9/community-events.php
417	{	community-events.1.2.9/community-events.php
418	switch($message) {	community-events.1.2.9/community-events.php
431	case '9':	community-events.1.2.9/community-events.php
432	echo " " . $_GET['importrowscount'] . " " . __('row(s) found', 'link-library') . ". " . $_GET['successimportcount'] . " " . __('link(s) imported successfully', 'link-library') . ". ";	community-events.1.2.9/community-events.php

Name of the enclosing function:on_show_page()
phpSAFE's decision: 1

E.3.2.2 Explanation

Tainted input reaches sensitive sink.

E.3.3 Slice #20

E.3.3.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #55
pixy XSS Slice #260
phpSAFE XSS Slice #2755
wap XSS Slice #17
Sensitive sink

- Name: echo
- Line: 14
- File: community-events.1.2.9/get-events.php

Entry points:

L	Type	Name	R S D	File
5	GET	$_GET['year']	0	community-events.1.2.9/get-events.php
6	GET	$_GET['outlook']	0	community-events.1.2.9/get-events.php
7	GET	$_GET['showdate']	0	community-events.1.2.9/get-events.php
8	GET	$_GET['maxevents']	0	community-events.1.2.9/get-events.php
9	GET	$_GET['moderateevents']	0	community-events.1.2.9/get-events.php
10	GET	$_GET['searchstring']	0	community-events.1.2.9/get-events.php
14	GET	$_GET['dayofyear']	1	community-events.1.2.9/get-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
5	variable	$year	$_GET['year']	1	1	0	0	community-events.1.2.9/get-events.php
6	variable	$outlook	$_GET['outlook']	1	1	0	0	community-events.1.2.9/get-events.php
7	variable	$showdate	$_GET['showdate']	1	1	0	0	community-events.1.2.9/get-events.php
8	variable	$maxevents	$_GET['maxevents']	1	1	0	0	community-events.1.2.9/get-events.php
9	variable	$moderateevents	$_GET['moderateevents']	1	1	0	0	community-events.1.2.9/get-events.php
10	variable	$searchstring	$_GET['searchstring']	1	1	0	0	community-events.1.2.9/get-events.php
14	function	$my_community _events_plugin-> eventlist($year, $_GET['dayofyear'], $outlook, $showdate, $maxevents, $moderateevents, $searchstring, $options['fullscheduleurl'], $options['addeventurl'], $options['allowuserediting'], $options['displayendtimefield'])	$year, $_GET['dayofyear'], $outlook, $showdate, $maxevents, $moderateevents, $searchstring	1	1	0	1	community-events.1.2.9/get-events.php
1729	parameter	$year	$year	1	1	0	0	community-events.1.2.9/community-events.php
1729	parameter	$dayofyear	$_GET['dayofyear']	1	1	0	0	community-events.1.2.9/community-events.php
1831	variable	$output	$dayofyear, $year	1	1	1	0	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
1736	$searchstring != ''	1		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
1736	$searchstring != ''	1	0	1	community-events.1.2.9/community-events.php
1752	$events	1	1	1	community-events.1.2.9/community-events.php
1830	count($events) > $maxevents	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
5	$year = $_GET['year'];	community-events.1.2.9/get-events.php
6	$outlook = $_GET['outlook'];	community-events.1.2.9/get-events.php
7	$showdate = $_GET['showdate'];	community-events.1.2.9/get-events.php
8	$maxevents = $_GET['maxevents'];	community-events.1.2.9/get-events.php
9	$moderateevents = $_GET['moderateevents'];	community-events.1.2.9/get-events.php
10	$searchstring = $_GET['searchstring'];	community-events.1.2.9/get-events.php
12	$options = get_option('CE_PP');	community-events.1.2.9/get-events.php
14	echo $my_community_events_plugin->eventlist($year, $_GET['dayofyear'], $outlook, $showdate, $maxevents, $moderateevents, $searchstring, $options['fullscheduleurl'],	community-events.1.2.9/get-events.php
15	$options['addeventurl'], $options['allowuserediting'], $options['displayendtimefield']);	community-events.1.2.9/get-events.php
1729	function eventlist ($year, $dayofyear, $outlook = 'true', $showdate = 'false', $maxevents = 5, $moderateevents = 'false', $searchstring = '', $fullscheduleurl = '',	community-events.1.2.9/community-events.php
1730	$addeventurl = '', $allowuserediting = false, $displayendtimefield = false) {	community-events.1.2.9/community-events.php
1734	$output = "<table class='ce-7day-innertable' id='ce-7day-innertable'>\n";	community-events.1.2.9/community-events.php
1736	if ($searchstring != '')	community-events.1.2.9/community-events.php
1737	{	community-events.1.2.9/community-events.php
1752	if ($events)	community-events.1.2.9/community-events.php
1753	{	community-events.1.2.9/community-events.php
1830	if (count($events) > $maxevents)	community-events.1.2.9/community-events.php
1831	$output .= "<tr><td>See all events for " . date("l, M jS", strtotime('+ ' . $dayofyearforcalc . 'days', mktime(0,0,0,1,1,$year))) . "</td></tr>\n";	community-events.1.2.9/community-events.php
1832	}	community-events.1.2.9/community-events.php
1833	}	community-events.1.2.9/community-events.php
2165	return $output;	community-events.1.2.9/community-events.php
2166	}	community-events.1.2.9/community-events.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Misc. comparisons

- wap_ori: y
- pixy: 3
- pixy_ori: fn

wap's decision: 1
RIPS's decision: 1
pixy's decision: 1
phpSAFE's decision: 1

E.3.3.2 Explanation

A vulnerability occurs because the function return value, in the form of the variable *$output*, is tainted by user input and reaches the sensitive sink, *echo*, in line 14. Note that the tools make an approximation in which they ignore the user-defined function and simply consider that taintedness is propagated because of the taintedness of its arguments. **There are other vulnerable paths, but only this one in particular was included for the sake of brevity**

E.3.4 Slice #21

E.3.4.1 Verdict: 1 (vulnerable)

wap XSS Slice #16
pixy XSS Slice #15
phpSAFE XSS Slice #2756
RIPS XSS Slice #56
Sensitive sink

- Name: echo
- Line: 15
- File: community-events.1.2.9/get-events-admin.php

Entry points:

L	Type	Name	R S D	File
5	GET	$_GET['currentyear']	0	community-events.1.2.9/get-events-admin.php
6	GET	$_GET['currentday']	0	community-events.1.2.9/get-events-admin.php
7	GET	$_GET['page']	0	community-events.1.2.9/get-events-admin.php
8	GET	$_GET['moderate']	0	community-events.1.2.9/get-events-admin.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
5	variable	$currentyear	$_GET['currentyear']	1	1	0	0	community-events.1.2.9/get-events-admin.php
6	variable	$currentday	$_GET['currentday']	1	1	0	0	community-events.1.2.9/get-events-admin.php
7	variable	$page	$_GET['page']	1	1	0	0	community-events.1.2.9/get-events-admin.php
8	variable	$moderate	$_GET['moderate']	1	1	0	0	community-events.1.2.9/get-events-admin.php
11	variable	$moderate	true	0	1	0	0	community-events.1.2.9/get-events-admin.php
13	variable	$moderate	false	0	1	0	0	community-events.1.2.9/get-events-admin.php
15	function	$my_community_events _plugin->print_event _table($currentyear, $currentday, $page, $moderate);	$currentyear, $currentday, $page	1	1	0	1	community-events.1.2.9/get-events.php
868	parameter	$currentyear	$currentyear	1	1	0	0	community-events.1.2.9/community-events.php
868	parameter	$currentday	$currentday	1	1	0	0	community-events.1.2.9/community-events.php
868	parameter	$page	$page	1	1	0	0	community-events.1.2.9/community-events.php
936	variable	$output	$page	1	1	1	0	community-events.1.2.9/community-events.php
940	variable	$output	$page	1	1	1	0	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
927	$events	0	0	1	community-events.1.2.9/community-events.php
929	foreach($events as $event)	0	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
5	$currentyear = $_GET['currentyear'];	community-events.1.2.9/get-events-admin.php
6	$currentday = $_GET['currentday'];	community-events.1.2.9/get-events-admin.php
7	$page = $_GET['page'];	community-events.1.2.9/get-events-admin.php
8	$moderate = $_GET['moderate'];	community-events.1.2.9/get-events-admin.php
10	if ($moderate == 'true')	community-events.1.2.9/get-events-admin.php
11	$moderate = true;	community-events.1.2.9/get-events-admin.php
12	else	community-events.1.2.9/get-events-admin.php
13	$moderate = false;	community-events.1.2.9/get-events-admin.php
15	echo $my_community_events_plugin->print_event_table($currentyear, $currentday, $page, $moderate);	community-events.1.2.9/get-events-admin.php
868	function print_event_table($currentyear, $currentday, $page, $moderate = false) {	community-events.1.2.9/community-events.php
908	$output .= " \n";	community-events.1.2.9/community-events.php
927	if ($events)	community-events.1.2.9/community-events.php
928	{	community-events.1.2.9/community-events.php
929	foreach($events as $event) {	community-events.1.2.9/community-events.php
936	$output .= "\t\t\t<td style='background: #FFF'>" . stripslashes($event['event_name']) . "</td>\n";	community-events.1.2.9/community-events.php
940	$output .= "\t\t\t<td style='background:#FFF'><a href='admin.php?page=community-events-events&deleteevent=" . $event['event_id'] . "&pagecount=" . $page . "'\n";	community-events.1.2.9/community-events.php
943	}	community-events.1.2.9/community-events.php
944	}	community-events.1.2.9/community-events.php
961	return $output;	community-events.1.2.9/community-events.php
962	}	community-events.1.2.9/community-events.php

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Misc. comparisons

- wap_ori: y
- pixy: 1
- pixy_ori: y

RIPS's decision: 1
pixy's decision: 1
phpSAFE's decision: 1
wap's decision: 1

E.3.4.2 Explanation

A vulnerability occurs because the function return value, in the form of the variable *$output*, is tainted by user input and reaches the sensitive sink, *echo*, in line 15. Note that the tools make an approximation in which they ignore the user-defined function and simply consider that taintedness is propagated because of the taintedness of its arguments. No further paths to explore.

E.3.5 Slice #22

E.3.5.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #57
phpSAFE XSS Slice #2757
Sensitive sink

- Name: echo
- Line: 538
- File: community-events.1.2.9/rss.genesis.php

Entry points:

L	Type	Name	RSD	File
538	FILES	file_get_contents ($name)	1	community-events.1.2.9/rss.genesis.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
522	!$file	0	0	0	community-events.1.2.9/rss.genesis.php

Slice:

L	Instruction	File
492	function organizeData() {	community-events.1.2.9/rss.genesis.php
495	$this->rss_feed .= $this->rss_header;	community-events.1.2.9/rss.genesis.php
496	$this->rss_feed .= $this->channel_data;	community-events.1.2.9/rss.genesis.php
497	$this->rss_feed .= $this->image_data;	community-events.1.2.9/rss.genesis.php
498	$this->rss_feed .= $this->input_data;	community-events.1.2.9/rss.genesis.php
501	foreach ($this->item_data as $item) :	community-events.1.2.9/rss.genesis.php
503	$this->rss_feed .= $item;	community-events.1.2.9/rss.genesis.php
505	endforeach;	community-events.1.2.9/rss.genesis.php
508	$this->rss_feed .= $this->rss_footer;	community-events.1.2.9/rss.genesis.php
510	}	community-events.1.2.9/rss.genesis.php
513	function createFile ($name = ”my.rss”) {	community-events.1.2.9/rss.genesis.php
516	$this->organizeData();	community-events.1.2.9/rss.genesis.php
519	$file = @fopen ($name, ”w”);	community-events.1.2.9/rss.genesis.php
522	if (!$file) :	community-events.1.2.9/rss.genesis.php
524	die (”Critical Error: Unable to create: $name”);	community-events.1.2.9/rss.genesis.php
526	endif;	community-events.1.2.9/rss.genesis.php
529	fwrite ($file, $this->rss_feed);	community-events.1.2.9/rss.genesis.php
538	echo file_get_contents ($name);	community-events.1.2.9/rss.genesis.php

Additional RIPS results——-
Alternative trigger files:

- community-events.1.2.9/rssfeed.php

Additional comment:
Userinput reaches sensitive sink when function createfile() is called.

———————————————

Name of the enclosing function:createFile ($name = “my.rss”)
phpSAFE’s decision: 1
RIPS’s decision: 1

E.3.5.2 Explanation

A vulnerability occurs because data from RSS feed reaches a sensitive sink.
Note that both the entry point and the sensitive sink are in the same line, which caused some confusion

---

E.3.6 Slice #23

E.3.6.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #52
Sensitive sink

- Name: echo
- Line: 1612

- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	RSD	File
1608	GET	$_GET['pagecount']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
1608	variable	$pagecount	$_GET['pagecount']	1	1	0	0	community-events.1.2.9/community-events.php
1612	function	$this->print_event_table($currentyear, $currentday, $pagecount, false)	$pagecount	1	1	0	1	community-events.1.2.9/community-events.php
868	parameter	$page	$pagecount	1	1	0	0	community-events.1.2.9/community-events.php
936	variable	$output	$page	1	1	1	0	community-events.1.2.9/community-events.php
940	variable	$output	$page	1	1	1	0	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1607	$_GET['pagecount'] != ""	0		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1607	$_GET['pagecount'] != ""	0	0	1	community-events.1.2.9/community-events.php
927	$events	0	0	1	community-events.1.2.9/community-events.php
929	foreach($events as $event)	0	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1377	function events_meta_box($data) {	community-events.1.2.9/community-events.php
1607	if ($_GET['pagecount'] != "")	community-events.1.2.9/community-events.php
1608	$pagecount = $_GET['pagecount'];	community-events.1.2.9/community-events.php
1612	echo $this->print_event_table($currentyear, $currentday, $pagecount, false); ?>	community-events.1.2.9/community-events.php
868	function print_event_table($currentyear, $currentday, $page, $moderate = false) {	community-events.1.2.9/community-events.php
908	$output .= " \n";	community-events.1.2.9/community-events.php
927	if ($events)	community-events.1.2.9/community-events.php
928	{	community-events.1.2.9/community-events.php
929	foreach($events as $event) {	community-events.1.2.9/community-events.php
936	$output .= "\t\t\t<td style='background: #FFF'>" . stripslashes($event['event_name']) . "</td>\n";	community-events.1.2.9/community-events.php
940	$output .= "\t\t\t<td style='background:#FFF'><a href='admin.php?page=community-events-events&deleteevent=" . $event['event_id'] . "&pagecount=" . $page . "'\n";	community-events.1.2.9/community-events.php
943	}	community-events.1.2.9/community-events.php
944	}	community-events.1.2.9/community-events.php
961	return $output;	community-events.1.2.9/community-events.php
962	}	community-events.1.2.9/community-events.php

Additional RIPS results———-
Alternative trigger files:

- community-events.1.2.9/community-events.php
- community-events.1.2.9/get-events-admin.php
- community-events.1.2.9/get-events.php

Additional comment:
Userinput reaches sensitive sink when function events_meta_box() is called.

———————————————

Name of the enclosing function:events_meta_box($data)
RIPS's decision: 1

E.3.6.2 Explanation

A vulnerability occurs because the function return value, in the form of the variable *$output*, is tainted by user input and reaches the sensitive sink, *echo*, in line 1612. Note that the tools make an approximation in which they ignore the user-defined function and simply consider that taintedness is propagated because of the taintedness of its arguments. No further paths to explore.

E.3.7 Slice #24

E.3.7.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #53

Sensitive sink

- Name: echo
- Line: 1614
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	RSD	File
1608	GET	$_GET['pagecount']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
1608	variable	$pagecount	$_GET['pagecount']	1	1	0	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
1607	$_GET['pagecount'] != ""	1		1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
1607	$_GET['pagecount'] != ""	1	0	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
1377	function events_meta_box($data) {	community-events.1.2.9/community-events.php
1607	if ($_GET['pagecount'] != "")	community-events.1.2.9/community-events.php
1608	$pagecount = $_GET['pagecount'];	community-events.1.2.9/community-events.php
1614	<input type='hidden' name='eventpage' id='eventpage' value='<?php echo $pagecount; ?>' />	community-events.1.2.9/community-events.php

Additional RIPS results——-
Additional comment:
Userinput reaches sensitive sink when function events_meta_box() is called.

Name of the enclosing function:events_meta_box($data)
RIPS's decision: 1

E.3.7.2 Explanation

User input reaches sensitive sink without sanitization

E.3.8 Slices #25, #26-#38

E.3.8.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #54
Sensitive sink

- Name: echo
- Line: 2586
- File: community-events.1.2.9/community-events.php

Entry points:

L	Type	Name	RSD	File
2333	GET	$_GET['eventday']	0	community-events.1.2.9/community-events.php
2334	GET	$_GET['eventyear']	0	community-events.1.2.9/community-events.php
2492	GET	$_GET['search']	0	community-events.1.2.9/community-events.php
2500	GET	$_GET['venue']	0	community-events.1.2.9/community-events.php
2505	GET	$_GET['category']	0	community-events.1.2.9/community-events.php
2510	GET	$_GET['location']	0	community-events.1.2.9/community-events.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	Nt d	RSD	File
2333	variable	$queryday	$_GET['eventday']	1	1	0	0	community-events.1.2.9/community-events.php
2334	variable	$queryyear	$_GET['eventyear']	1	1	0	0	community-events.1.2.9/community-events.php
2586	variable	$eventquery	$queryyear, $_GET['search'], $_GET['venue'], $_GET['category'], $_GET['location'], $queryday	1	1	1	1	community-events.1.2.9/community-events.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
2330	isset($_GET['eventyear']) && isset($_GET['eventday'])	0		1	community-events.1.2.9/community-events.php
2490	isset($_GET['search']) && ($_GET['search'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2498	isset($_GET['venue']) && ($_GET['venue'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2503	isset($_GET['category']) && ($_GET['category'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2508	isset($_GET['location']) && ($_GET['location'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2525	isset($_GET['search']) && ($_GET['search'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2533	isset($_GET['venue']) && ($_GET['venue'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2538	isset($_GET['category']) && ($_GET['category'] != '')	0	2483	1	community-events.1.2.9/community-events.php
2543	isset($_GET['location']) && ($_GET['location'] != '')	0	2483	1	community-events.1.2.9/community-events.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
2330	isset($_GET['eventyear']) && isset($_GET['eventday']) && isset($_GET['dateset'])	0	0	1	community-events.1.2.9/community-events.php
2483	while ($loopcount > 0)	1	0	1	community-events.1.2.9/community-events.php
2490	isset($_GET['search']) && ($_GET['search'] != '')	0	1	1	community-events.1.2.9/community-events.php
2498	isset($_GET['venueset']) && isset($_GET['venue']) && ($_GET['venue'] != '')	0	1	1	community-events.1.2.9/community-events.php
2503	isset($_GET['categoryset']) && isset($_GET['category']) && ($_GET['category'] != '')	0	1	1	community-events.1.2.9/community-events.php
2508	isset($_GET['locationset']) && isset($_GET['location']) && ($_GET['location'] != '')	0	1	1	community-events.1.2.9/community-events.php
2525	isset($_GET['search']) && ($_GET['search'] != '')	0	1	1	community-events.1.2.9/community-events.php
2533	isset($_GET['venueset']) && isset($_GET['venue']) && ($_GET['venue'] != '')	0	1	1	community-events.1.2.9/community-events.php
2538	isset($_GET['categoryset']) && isset($_GET['category']) && ($_GET['category'] != '')	0	1	1	community-events.1.2.9/community-events.php
2543	isset($_GET['locationset']) && isset($_GET['location']) && ($_GET['location'] != '')	0	1	1	community-events.1.2.9/community-events.php
2584	$loopcount == 90	1	1	1	community-events.1.2.9/community-events.php

Slice:

L	Instruction	File
2309	function ce_full($moderateevents = true, $fullvieweventsperpage = 20, $fullviewmaxdays = 90, $fullscheduleurl = ′′, $addeventurl = ′′, $allowusereditingt = false, $displayendtimefield = false) {	community-events.php
2330	if (isset($_GET[′eventyear′]) && isset($_GET[′eventday′]) && isset($_GET[′dateset′]))	community-events.php
2331	{	community-events.php
2333	$queryday = $_GET[′eventday′];	community-events.php
2334	$queryyear = $_GET[′eventyear′];	community-events.php
2336	}	community-events.php
2483	while ($loopcount > 0)	community-events.php
2484	{	community-events.php
2488	$eventquery .= ”where YEAR(event_start_date) = ” . $queryyear;	community-events.php
2490	if (isset($_GET[′search′]) && ($_GET[′search′] != ′′))	community-events.php
2491	{	community-events.php
2492	$eventquery .= ” and ((event_name like ′%” . $_GET[′search′] . ”%′)”;	community-events.php
2493	$eventquery .= ” or (ce_venue_name like ′%” . $_GET[′search′] . ”%′)”;	community-events.php
2494	$eventquery .= ” or (ce_venue_city like ′%” . $_GET[′search′] . ”%′)”;	community-events.php
2495	$eventquery .= ” or (event_description like ′%” . $_GET[′search′] . ”%′))”;	community-events.php
2496	}	community-events.php
2498	if (isset($_GET[′venueset′]) && isset($_GET[′venue′]) && ($_GET[′venue′] != ′′))	community-events.php
2499	{	community-events.php
2500	$eventquery .= ” and ce_venue_id = ” . $_GET[′venue′];	community-events.php
2501	}	community-events.php
2503	if (isset($_GET[′categoryset′]) && isset($_GET[′category′]) && ($_GET[′category′] != ′′))	community-events.php
2504	{	community-events.php
2505	$eventquery .= ” and event_category = ” . $_GET[′category′];	community-events.php
2506	}	community-events.php
2508	if (isset($_GET[′locationset′]) && isset($_GET[′location′]) && ($_GET[′location′] != ′′))	community-events.php
2509	{	community-events.php
2510	$eventquery .= ” and ce_venue_city = ′” . $_GET[′location′] . ”′”;	community-events.php
2511	}	community-events.php
2513	$eventquery .= ” and DAYOFYEAR(DATE(event_start_date)) = ” . $queryday . ” ”;	community-events.php
2525	if (isset($_GET[′search′]) && ($_GET[′search′] != ′′))	community-events.php
2526	{	community-events.php
2527	$eventquery .= ” ((event_name like ′%” . $_GET[′search′] . ”%′)”;	community-events.php
2528	$eventquery .= ” or (ce_venue_name like ′%” . $_GET[′search′] . ”%′)”;	community-events.php
2529	$eventquery .= ” or (ce_venue_city like ′%” . $_GET[′search′] . ”%′)”;	community-events.php
2530	$eventquery .= ” or (event_description like ′%” . $_GET[′search′] . ”%′)) and”;	community-events.php
2531	}	community-events.php
2533	if (isset($_GET[′venueset′]) && isset($_GET[′venue′]) && ($_GET[′venue′] != ′′))	community-events.php
2534	{	community-events.php
2535	$eventquery .= ” ce_venue_id = ” . $_GET[′venue′] . ” and ”;	community-events.php
2536	}	community-events.php
2538	if (isset($_GET[′categoryset′]) && isset($_GET[′category′]) && ($_GET[′category′] != ′′))	community-events.php
2539	{	community-events.php
2540	$eventquery .= ” event_category = ” . $_GET[′category′] . ” and ”;	community-events.php
2541	}	community-events.php
2543	if (isset($_GET[′locationset′]) && isset($_GET[′location′]) && ($_GET[′location′] != ′′))	community-events.php
2544	{	community-events.php
2545	$eventquery .= ” ce_venue_city = ′” . $_GET[′location′] . ”′ and ”;	community-events.php
2546	}	community-events.php
2548	$eventquery .= ”((YEAR(event_start_date) = ” . $queryyear . ” and YEAR(event_end_date) = ” . $queryyear;	community-events.php
2549	$eventquery .= ” and DAYOFYEAR(DATE(event_start_date)) <= ” . $queryday . ” ”;	community-events.php
2550	$eventquery .= ” and DAYOFYEAR(DATE(event_end_date)) >= ” . $queryday . ”) OR ”;	community-events.php
2552	$eventquery .= ”(YEAR(event_start_date) = ” . $queryyear . ” and YEAR(event_end_date) > ” . $queryyear;	community-events.php
2553	$eventquery .= ” and DAYOFYEAR(DATE(event_start_date)) <= ” . $queryday . ”) OR ”;	community-events.php
2555	$eventquery .= ” (YEAR(event_start_date) < ” . $queryyear;	community-events.php
2556	$eventquery .= ” and DAYOFYEAR(DATE(event_end_date)) >= ” . $queryday;	community-events.php
2557	$eventquery .= ” and DAYOFYEAR(YEAR(event_end_date)) = ” . $queryyear . ”) OR ”;	community-events.php
2559	$eventquery .= ” (YEAR(event_start_date) < ” . $queryyear;	community-events.php
2560	$eventquery .= ” and DAYOFYEAR(YEAR(event_end_date)) > ” . $queryyear . ”) OR ”;	community-events.php
2562	$eventquery .= ” (YEAR(event_end_date) > ” . $queryyear . ” ”;	community-events.php
2563	$eventquery .= ” and YEAR(event_start_date) = ” . $queryyear . ” ”;	community-events.php
2564	$eventquery .= ” and DAYOFYEAR(DATE(event_start_date)) <= ” . $queryday . ”) OR ”;	community-events.php
2566	$eventquery .= ” (YEAR(event_end_date) > ” . $queryyear . ” ”;	community-events.php
2567	$eventquery .= ” and YEAR(event_start_date) < ” . $queryyear . ”) OR ”;	community-events.php
2569	$eventquery .= ” (YEAR(event_end_date) = ” . $queryyear;	community-events.php
2570	$eventquery .= ” and YEAR(DATE(event_start_date)) < ” . $queryyear;	community-events.php
2571	$eventquery .= ” and DAYOFYEAR(DATE(event_end_date)) >= ” . $queryday . ”)) ”;	community-events.php
2584	if ($loopcount == 90)	community-events.php
2585	{	community-events.php
2586	echo ”<!-- ” . $eventquery . ” -->”;	community-events.php

Additional RIPS results———-
Alternative trigger files:

- community-events.1.2.9/community-events.php
- community-events.1.2.9/get-events-admin.php
- community-events.1.2.9/get-events.php

Additional comment:
Userinput reaches sensitive sink when function ce_full() is called.

———————————————

RIPS's decision: 1 (#25)

E.3.8.2 Explanation

Tainted variable reaches sensitive sink (#25). There are additional 13 vulnerabilities depending on the control structures (#26-#38).

E.4 VV

- If a tainted value reaches a sensitive sink through a user-defined function, there is a vulnerability iff the return value is tainted.

E.5 WAP false negatives

- Out of 16 vulnerabilities, WAP only managed to signal 1. Running it with *community-events.1.2.9/community-events.php* will not yield any results.

E.6 Misc.

- The *die* function exits the current script. This behaviour should be taken into consideration.

Appendix F

Plugin: contact-form.2.7.5

F.1 Type: SQLI

F.1.1 Slice #1

F.1.1.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #19
Sensitive sink

- Name: $wpdb->query
- Line: 5
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
5	POST	$_POST['form_id']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	1	contact-form.2.7.5/admin/editform.php
4	isset($_POST['delete_form'])	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
3	if ((!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))) {	contact-form.2.7.5/admin/editform.php
4	if (isset($_POST['delete_form'])) {	contact-form.2.7.5/admin/editform.php
5	$wpdb->query("DELETE FROM $table_name WHERE ID = ".$_POST['form_id']);	contact-form.2.7.5/admin/editform.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
wap's decision: 1

F.1.1.2 Explanation

Tainted input reaches sensitive sink. For the record, *$table_name = $wpdb->prefix."easyform";* (*contact-form.2.7.5/define.php*)
wap-2.1 detects and fixes this vulnerability

F.1.2 Slice #2

F.1.2.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #20
Sensitive sink

- Name: $wpdb->query
- Line: 6
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
6	POST	$_POST['form_id']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	1	contact-form.2.7.5/admin/editform.php
4	isset($_POST['delete_form'])	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
3	if ((!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))) {	contact-form.2.7.5/admin/editform.php
4	if (isset($_POST['delete_form'])) {	contact-form.2.7.5/admin/editform.php
6	$wpdb->query("DELETE FROM $settings_table_name WHERE form_id = ".$_POST['form_id']);	contact-form.2.7.5/admin/editform.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
wap's decision: 1

F.1.2.2 Explanation

Tainted input reaches sensitive sink.
wap-2.1 detects and fixes this vulnerability

F.1.3 Slice #3

F.1.3.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #21
Sensitive sink

- Name: $wpdb->query
- Line: 63
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
63	POST	$_POST['form_id']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
61	isset($_POST['wpcf_easyform_update_submit'])	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
61	if (isset($_POST['wpcf_easyform_update_submit'])) {	contact-form.2.7.5/admin/editform.php
63	$wpdb->query("DELETE FROM $settings_table_name WHERE form_id = ".$_POST['form_id']);	contact-form.2.7.5/admin/editform.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1
wap's decision: 1

F.1.3.2 Explanation

Tainted input reaches sensitive sink.
wap-2.1 detects and fixes this vulnerability

F.1.4 Slice #4

F.1.4.1 Verdict: 1 (vulnerable)

Sensitive sink

- Name: $wpdb->query
- Line: 64
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
67	POST	$_POST['formname']	1	contact-form.2.7.5/admin/editform.php
68	POST	$_POST['destinatary']	1	contact-form.2.7.5/admin/editform.php
69	POST	$_POST['show_label_inside']	1	contact-form.2.7.5/admin/editform.php
70	POST	$_POST['form_id']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
61	isset($_POST['wpcf_easyform_update_submit'])	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
61	if (isset($_POST['wpcf_easyform_update_submit'])) {	contact-form.2.7.5/admin/editform.php
64	$wpdb->query($wpdb->prepare("	contact-form.2.7.5/admin/editform.php
65	UPDATE $table_name	contact-form.2.7.5/admin/editform.php
66	SET	contact-form.2.7.5/admin/editform.php
67	name = '".$_POST['formname']."',	contact-form.2.7.5/admin/editform.php
68	destinatary = '".$_POST['destinatary']."',	contact-form.2.7.5/admin/editform.php
69	show_label_inside = '".$_POST['show_label_inside']."'	contact-form.2.7.5/admin/editform.php
70	WHERE ID = ".$_POST['form_id']));	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
wap's decision: 1

F.1.4.2 Explanation

A vulnerability occurs because a prepared statement is being used incorrectly. False negative for all tools **except wap-2.1** (but fails to apply the fix).

F.1.5 Slice #5

F.1.5.1 Verdict: 1 (vulnerable)

phpSAFE SQLI Slice #22
Sensitive sink

- Name: $wpdb->query
- Line: 14
- File: contact-form.2.7.5/requests/sort_row.request.php

Entry points:

L	Type	Name	RSD	File
7	POST	$_POST['field_num']	0	contact-form.2.7.5/requests/sort_row.request.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
7	variable	$position	$_POST['field_num']	1	1	0	0	contact-form.2.7.5/requests/sort_row.request.php
7	variable	$field_id	$_POST['field_num']	1	1	0	0	contact-form.2.7.5/requests/sort_row.request.php
10	variable	$query	$position,$field_id	1	1	0	1	contact-form.2.7.5/requests/sort_row.request.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
7	foreach ($_POST['field_num'] as $position=>$field_id)	1		1	contact-form.2.7.5/requests/sort_row.request.php
9	$field_id > 0	1	7	1	contact-form.2.7.5/requests/sort_row.request.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
7	foreach ($_POST['field_num'] as $position=>$field_id)	1	0	1	contact-form.2.7.5/requests/sort_row.request.php
9	$field_id > 0	1	1	1	contact-form.2.7.5/requests/sort_row.request.php

Slice:

L	Instruction	File
7	foreach ($_POST[′field_num′] as $position=>$field_id) {	contact-form.2.7.5/requests/sort_row.request.php
9	if ($field_id > 0) {	contact-form.2.7.5/requests/sort_row.request.php
10	$query = ”	contact-form.2.7.5/requests/sort_row.request.php
11	UPDATE $settings_table_name	contact-form.2.7.5/requests/sort_row.request.php
12	SET position = ′”.$position.”′	contact-form.2.7.5/requests/sort_row.request.php
13	WHERE ID = $field_id”;	contact-form.2.7.5/requests/sort_row.request.php
14	$wpdb->query($query);	contact-form.2.7.5/requests/sort_row.request.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

phpSAFE's decision: 1

F.1.5.2 Explanation

A vulnerability occurs due to tainted variable(s) *$position* and/or *$field_id*. Due to PHP's weak typing, constraints that involve numeric comparison can be bypassed by strings, as long as the first character is a number that satisfies the constraint. *if($field_id > 0)*, from line 9, can be bypassed with, for instance, the value
“2<script>alert(document.cookie);</script>”.
$position is also vulnerable if the provided associative array has a SQL injection payload as one of its keys.

F.2 Type: XSS (reflected)

F.2.1 Slice #6

F.2.1.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #58
phpSAFE XSS Slice #2758
Sensitive sink

- Name: echo
- Line: 31
- File: contact-form.2.7.5/admin/addform.php

Entry points:

L	Type	Name	R S D	File
31	SERVER	$_SERVER['REQUEST_URI']	1	contact-form.2.7.5/admin/addform.php

Slice:

L	Instruction	File
31	<form method="POST" accept-charset="utf-8" target="_self" action="<?php echo $_SERVER['REQUEST_URI']; ?>">	contact-form.2.7.5/admin/addform.php

Additional RIPS results———-
Alternative trigger files:

- contact-form.2.7.5/admin/options-page.php

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1
phpSAFE's decision: 1
pixy's decision: 1
wap's decision: 1

F.2.1.2 Explanation

Tainted input reaches sensitive sink. **wap-2.1 detects this vulnerability** but cannot fix it.

F.2.2 Slice #7

F.2.2.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #59
phpSAFE XSS Slice #2766
Sensitive sink

- Name: echo
- Line: 11
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
11	SERVER	$_SERVER['REQUEST_URI']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
3	if ((!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))) {	contact-form.2.7.5/admin/editform.php
11	<form method="POST" accept-charset="utf-8" target="_self" action="<?php echo $_SERVER['REQUEST_URI']; ?>">	contact-form.2.7.5/admin/editform.php

Additional RIPS results———-
Alternative trigger files:

- contact-form.2.7.5/admin/options-page.php

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1
wap's decision: 1
phpSAFE's decision: 1

F.2.2.2 Explanation

Tainted input reaches sensitive sink. **wap-2.1 detects this vulnerability** but cannot fix it.

F.2.3 Slice #8

F.2.3.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #60
phpSAFE XSS Slice #2776
Sensitive sink

- Name: echo
- Line: 111
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
111	SERVER	$_SERVER['REQUEST_URI']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
111	<form method="POST" accept-charset="utf-8" target="_self" action="<?php echo $_SERVER['REQUEST_URI']; ?>">	contact-form.2.7.5/admin/editform.php

Additional RIPS results———-
Alternative trigger files:

- contact-form.2.7.5/admin/options-page.php

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

Name of the enclosing function: No function
RIPS's decision: 1
wap's decision: 1
phpSAFE's decision: 1

F.2.3.2 Explanation

Tainted input reaches sensitive sink. **wap-2.1 detects this vulnerability** but cannot fix it.

F.2.4 Slice #9

F.2.4.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #61
phpSAFE XSS Slice #2777
Sensitive sink

- Name: echo
- Line: 157
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
88	POST	$_POST['form_id']	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
88	variable	$shortcode	$_POST['form_id']	1	1	0	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
88	$shortcode = '[easyform id="'.$_POST['form_id'].'"]';	contact-form.2.7.5/admin/editform.php
157	<?php echo $shortcode; ?>	contact-form.2.7.5/admin/editform.php

Additional RIPS results———-
Alternative trigger files:

- contact-form.2.7.5/admin/options-page.php

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1
wap's decision: 1
phpSAFE's decision: 1

F.2.4.2 Explanation

Tainted variable reaches sensitive sink. **wap-2.1 detects this vulnerability** but cannot fix it.

F.2.5 Slice #10

F.2.5.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #62
phpSAFE XSS Slice #2834
Sensitive sink

- Name: echo
- Line: 256
- File: contact-form.2.7.5/admin/editform.php

Entry points:

L	Type	Name	RSD	File
256	POST	$_POST['form_id']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
256	<input class="button-primary" type="submit" name="wpcf_easyform_update_submit" value="<?php _e('Save Form'); ?>" /><input type="hidden" name="form_id" value="<?php echo $_POST['form_id']; ?>" />	contact-form.2.7.5/admin/editform.php

Additional RIPS results——-
Alternative trigger files:

- contact-form.2.7.5/admin/options-page.php

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

RIPS's decision: 1
wap's decision: 1
phpSAFE's decision: 1

F.2.5.2 Explanation

Tainted input reaches sensitive sink. **wap-2.1 detects this vulnerability** but cannot fix it.

F.2.6 Slice #11

F.2.6.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #63
phpSAFE XSS Slice #2839
Sensitive sink

- Name: echo
- Line: 19

- File: contact-form.2.7.5/admin/settings.php

Entry points:

L	Type	Name	R S D	File
19	SERVER	$_SERVER['REQUEST_URI']	1	contact-form.2.7.5/admin/settings.php

Slice:

L	Instruction	File
19	<form method="POST" accept-charset="utf-8" target="_self" action="<?php echo $_SERVER['REQUEST_URI']; ?>">	contact-form.2.7.5/admin/settings.php

Additional RIPS results———-
Alternative trigger files:

- contact-form.2.7.5/admin/settings.php

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1
wap's decision: 1
phpSAFE's decision: 1

F.2.6.2 Explanation

Tainted input reaches sensitive sink. **wap-2.1 detects this vulnerability** but cannot fix it.

F.3 Type: Stored injection

F.3.1 Slice #12

F.3.1.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2784
Sensitive sink

- Name: echo
- Line: 214
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
None.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
214	variable	$field->ID	$field	0	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	N t d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results(”	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	”);	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
214	<tr id=”field_num_<?php echo $field->ID; ?>”>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.1.2 Explanation

Not vulnerable. None of the writes affect the *ID* field from the *$settings_table_name* table. For the record, *$settings_table_name = $wpdb->prefix.“easyform_settings”;* (*contact-form.2.7.5/define.php*)

F.3.2 Slice #13

F.3.2.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2785
Sensitive sink

- Name: echo
- Line: 216
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
See explanation.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(” SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ”)	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
216	variable	$field->name	$field	0	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
216	<input class="field required" type="text" name="field_name[]" value="<?php echo $field->name; ?>" field="field_name"/>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.2.2 Explanation

Not vulnerable. Insertions and updates regarding the *name* field from the *$settings_table_name* table are protected against Stored XSS through a particular use of preg_replace:

- preg_replace("/[^a-z0-9_]/i", "", $_POST['field_name'][$index]);

This function returns a modified version of *$_POST['field_name'][$index]* so that it only contains letters, numbers and/or underscore.

F.3.3 Slice #14

F.3.3.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2794
Sensitive sink

- Name: echo
- Line: 219
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
77	POST	$_POST['field_label'][$index]	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Nt d	RSD	File
77	variable	$easyform_settings_row['label']	$_POST['field_label'][$index]	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
72	foreach ($_POST['field_name'] as $index=>$value)	1	61	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
61	isset($_POST['wpcf_easyform_update_submit'])	1	1	1	contact-form.2.7.5/admin/editform.php
72	foreach ($_POST['field_name'] as $index=>$value)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
61	if (isset($_POST['wpcf_easyform_update_submit'])) {	contact-form.2.7.5/admin/editform.php
72	foreach ($_POST['field_name'] as $index=>$value) {	contact-form.2.7.5/admin/editform.php
77	$easyform_settings_row['label'] = $_POST['field_label'][$index];	contact-form.2.7.5/admin/editform.php
83	$wpdb->insert($settings_table_name, $easyform_settings_row);	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
219	variable	$field->label	$field->label	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results(”	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	”);	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
219	<input class=”field” type=”text” name=”field_label[]” value=”<?php echo $field->label; ?>” field=”field_label”/>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.3.2 Explanation

Vulnerable. It is possible to place malicious code after the *<input>* element because *$field->label* is not escaped.

F.3.4 Slice #15

F.3.4.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2795
Sensitive sink

- Name: echo
- Line: 223
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(” SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ”)	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
223	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
223	<option value="text"<?php echo($field->type == "text" ? ′ selected="selected"′ : ′′); ?>><?php _e(′Text′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.4.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *' selected="selected" '* or *"* depending on the result of the comparison.

F.3.5 Slice #16

F.3.5.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2800
Sensitive sink

- Name: echo
- Line: 224
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
224	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	N t d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results(”	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	”);	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
224	<option value=”textarea”<?php echo($field->type == ”textarea” ? ′ selected=”selected”′ : ′′); ?>><?php _e(′Text Area′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.5.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *‘ selected=“selected” ’* or *”* depending on the result of the comparison.

F.3.6 Slice #17

F.3.6.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2807
Sensitive sink

- Name: echo
- Line: 225
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
101	BD	$wpdb->get_results(” SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ”)	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
225	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
225	<option value="email"<?php echo($field->type == "email" ? ′ selected="selected"′ : ′′); ?>><?php _e(′Email′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.6.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *' selected="selected" '* or *"* depending on the result of the comparison.

F.3.7 Slice #18

F.3.7.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2808
Sensitive sink

- Name: echo
- Line: 226
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
226	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results(”	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	”);	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
226	<option value=”url”><?php echo($field->type == ”url” ? ′ selected=”selected”′ : ′′); ?>><?php _e(′URL′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.7.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *‘ selected=“selected” ’* or *”* depending on the result of the comparison.

F.3.8 Slice #19

F.3.8.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2813
Sensitive sink

- Name: echo
- Line: 227
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
101	BD	$wpdb->get_results(” SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ”)	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
227	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
227	<option value="dropdown"<?php echo($field->type == "dropdown" ? ′ selected="selected"′ : ′′); ?>><?php _e(′Dropdown (1 choice)′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.8.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *' selected="selected" '* or *"* depending on the result of the comparison.

F.3.9 Slice #20

F.3.9.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2814
Sensitive sink

- Name: echo
- Line: 228
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
228	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	N t d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results(”	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	”);	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
228	<option value=”checkbox”<?php echo($field->type == ”checkbox” ? ′ selected=”selected”′ : ′′); ?>><?php _e(′Checkbox (Multi choice)′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.9.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *‘ selected=“selected” ’* or *”* depending on the result of the comparison.

F.3.10 Slice #21

F.3.10.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2821
Sensitive sink

- Name: echo
- Line: 229
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
101	BD	$wpdb->get_results(” SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ”)	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
229	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
229	<option value="radio"<?php echo($field->type == "radio" ? ′ selected="selected"′ : ′′); ?>><?php _e(′Radio Button (1 choice)′); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.10.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *' selected="selected" '* or *"* depending on the result of the comparison.

F.3.11 Slice #22

F.3.11.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2822
Sensitive sink

- Name: echo
- Line: 230
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Irrelevant. See explanation.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
230	variable	$field->type	$field->type	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	N t d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
230	<option value="submit"<?php echo($field->type == "submit" ? ' selected="selected"' : ''); ?>><?php _e('Submit Button'); ?></option>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.11.2 Explanation

Not vulnerable. *echo* does not print *$field->type*. It prints either *' selected="selected" '* or *"* depending on the result of the comparison.

F.3.12 Slice #23

F.3.12.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2828
Sensitive sink

- Name: echo
- Line: 234
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
79	POST	$_POST['field_value'][$index]	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
79	variable	$easyform_settings_row['value']	$_POST['field_value'][$index]	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
72	foreach ($_POST['field_name'] as $index=>$value)	1	61	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
61	isset($_POST['wpcf_easyform_update_submit'])	1	1	1	contact-form.2.7.5/admin/editform.php
72	foreach ($_POST['field_name'] as $index=>$value)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
61	if (isset($_POST['wpcf_easyform_update_submit'])) {	contact-form.2.7.5/admin/editform.php
72	foreach ($_POST['field_name'] as $index=>$value) {	contact-form.2.7.5/admin/editform.php
79	$easyform_settings_row['value'] = $_POST['field_value'][$index];	contact-form.2.7.5/admin/editform.php
83	$wpdb->insert($settings_table_name, $easyform_settings_row);	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
234	variable	$field->value	$field->value	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	");	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
234	<input type="text" class="field" name="field_value[]" value="<?php echo $field->value; ?>" field="field_value"/>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.12.2 Explanation

Vulnerable. It is possible to place malicious code after the *<input>* element because *$field->value* is not escaped.

F.3.13 Slice #24

F.3.13.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2833
Sensitive sink

- Name: echo
- Line: 240
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)

Irrelevant. See explanation.

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
101	BD	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
101	variable	$form_fields	$wpdb->get_results(" SELECT * FROM $settings_table_name WHERE form_id = $form_id ORDER BY position ")	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	function	stripslashes_deep ($form_fields)	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
108	variable	$form_fields	stripslashes_deep($form_fields)	0	1	0	0	contact-form.2.7.5/admin/editform.php
211	variable	$field	$form_fields	0	1	0	0	contact-form.2.7.5/admin/editform.php
238	variable	$field->required	$field->required	1	1	0	0	contact-form.2.7.5/admin/editform.php
238	variable	$checked	$field->required	0	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
211	foreach ($form_fields as $field)	1	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field)	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
101	$form_fields = $wpdb->get_results(”	contact-form.2.7.5/admin/editform.php
102	SELECT *	contact-form.2.7.5/admin/editform.php
103	FROM $settings_table_name	contact-form.2.7.5/admin/editform.php
104	WHERE form_id = $form_id	contact-form.2.7.5/admin/editform.php
105	ORDER BY position	contact-form.2.7.5/admin/editform.php
106	”);	contact-form.2.7.5/admin/editform.php
108	$form_fields = stripslashes_deep($form_fields);	contact-form.2.7.5/admin/editform.php
211	foreach ($form_fields as $field) {	contact-form.2.7.5/admin/editform.php
238	$checked = $field->required == 1 ? ′ checked=”checked”′ : ′′;	contact-form.2.7.5/admin/editform.php
240	<input type=”checkbox” class=”checkbox” value=”1” name=”field_required[<?php echo $i; ?>]” field=”field_required”<?php echo $checked; ?>/><?php _e(′Yes′); ?>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
phpSAFE's decision: 1

F.3.13.2 Explanation

Not vulnerable. *echo* does not print *$field->required*. It prints either ‘ *checked=“checked” ’* or ” depending on the result of the comparison in line 238.

F.3.14 Slice #25

F.3.14.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #2838
Sensitive sink

- Name: echo
- Line: 32
- File: contact-form.2.7.5/admin/editform.php

WRITES (1st step)
Entry points:

L	Type	Name	R S D	File
67	POST	$_POST['formname']	1	contact-form.2.7.5/admin/editform.php
68	POST	$_POST['destinatary']	1	contact-form.2.7.5/admin/editform.php
69	POST	$_POST['show_label_inside']	1	contact-form.2.7.5/admin/editform.php
70	POST	$_POST['form_id']	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	0	contact-form.2.7.5/admin/editform.php
61	isset($_POST['wpcf_easyform_update_submit'])	1	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
60	else {	contact-form.2.7.5/admin/editform.php
61	if (isset($_POST[′wpcf_easyform_update_submit′])) {	contact-form.2.7.5/admin/editform.php
64	$wpdb->query($wpdb->prepare("	contact-form.2.7.5/admin/editform.php
65	UPDATE $table_name	contact-form.2.7.5/admin/editform.php
66	SET	contact-form.2.7.5/admin/editform.php
67	name = ′".$_POST[′formname′]."′,	contact-form.2.7.5/admin/editform.php
68	destinatary = ′".$_POST[′destinatary′]."′,	contact-form.2.7.5/admin/editform.php
69	show_label_inside = ′".$_POST[′show_label_inside′]."′	contact-form.2.7.5/admin/editform.php
70	WHERE ID = ".$_POST[′form_id′]));	contact-form.2.7.5/admin/editform.php

Misc. (enclosing modules & functions)

- Name of the enclosing function: No function

READS (2nd step)
Entry points:

L	Type	Name	R	S	D	File
21	BD	$wpdb->get_results(" SELECT * FROM $table_name ORDER BY name ");	0			contact-form.2.7.5/admin/editform.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Nt d	RSD	File
21	variable	$results	$wpdb->get_results(" SELECT * FROM $table_name ORDER BY name ");	0	1	0	0	contact-form.2.7.5/admin/editform.php
27	variable	$form	$results	0	1	0	0	contact-form.2.7.5/admin/editform.php
28	variable	$form->ID	$form->ID	0	0	0	0	contact-form.2.7.5/admin/editform.php
28	variable	$form->name	$form->name	1	0	0	0	contact-form.2.7.5/admin/editform.php
28	variable	$options	$form->ID, $form->name	1	1	0	1	contact-form.2.7.5/admin/editform.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
27	foreach ($results as $form)	0	3	1	contact-form.2.7.5/admin/editform.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
3	(!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))	1	0	1	contact-form.2.7.5/admin/editform.php
27	foreach ($results as $form)	0	1	1	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
3	if ((!isset($_POST['wpcf_easyform_submit']) AND !isset($_POST['wpcf_easyform_update_submit'])) OR (isset($_POST['wpcf_easyform_submit']) AND isset($_POST['delete_form']))) {	contact-form.2.7.5/admin/editform.php
21	$results = $wpdb->get_results("	contact-form.2.7.5/admin/editform.php
22	SELECT *	contact-form.2.7.5/admin/editform.php
23	FROM $table_name	contact-form.2.7.5/admin/editform.php
24	ORDER BY name	contact-form.2.7.5/admin/editform.php
25	");	contact-form.2.7.5/admin/editform.php
27	foreach ($results as $form) {	contact-form.2.7.5/admin/editform.php
28	$options .= '<option value="'.$form->ID.'">'.$form->name.'</option>';	contact-form.2.7.5/admin/editform.php
29	}	contact-form.2.7.5/admin/editform.php
32	<?php echo $options; ?>	contact-form.2.7.5/admin/editform.php

Name of the enclosing function: No function
wap's decision: 1
phpSAFE's decision: 1

F.3.14.2 Explanation

Vulnerable. It is possible to place malicious code in (or after) the *<option>* element's text, because *$form->name* is not escaped.

F.3.15 Slice #26

F.3.15.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #2844
Sensitive sink

- Name: echo
- Line: 144
- File: contact-form.2.7.5/form.php

WRITES (1st step)
Entry points:

L	Type	Name	R S D	File
41	FILES	file(CF_PLUGIN_BASE_DIR . '/contact-form-wordpress/include/ratings.txt')	0	contact-form.2.7.5/form.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
41	variable	$file	file(CF_PLUGIN_BASE_DIR . '/contact-form-wordpress/include/ratings.txt')	1	1	0	0	contact-form.2.7.5/admin/editform.php
41	variable	$stringData	$file	1	1	0	0	contact-form.2.7.5/admin/editform.php

Slice:

L	Instruction	File
39	define ('CF_PLUGIN_BASE_DIR', WP_PLUGIN_DIR, true);	contact-form.2.7.5/form.php
40	function install() {	contact-form.2.7.5/form.php
41	$file = file(CF_PLUGIN_BASE_DIR . '/contact-form-wordpress/include/ratings.txt');	contact-form.2.7.5/form.php
42	$num_lines = count($file)-1;	contact-form.2.7.5/form.php
43	$picked_number = rand(0, $num_lines);	contact-form.2.7.5/form.php
44	for ($i = 0; $i <= $num_lines; $i++)	contact-form.2.7.5/form.php
45	{	contact-form.2.7.5/form.php
46	if ($picked_number == $i)	contact-form.2.7.5/form.php
47	{	contact-form.2.7.5/form.php
48	$myFile = CF_PLUGIN_BASE_DIR . '/contact-form-wordpress/include/standard.txt';	contact-form.2.7.5/form.php
49	$fh = fopen($myFile, 'w') or die("can't open file");	contact-form.2.7.5/form.php
50	$stringData = $file[$i];	contact-form.2.7.5/form.php
51	fwrite($fh, $stringData);	contact-form.2.7.5/form.php

Name of the enclosing function: No function

READS (2nd step)

Entry points:

L	Type	Name	R S D	File
139	FILES	fopen($myFile, 'r')	0	contact-form.2.7.5/form.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
139	variable	$fh	fopen($myFile, 'r')		1	0	0	contact-form.2.7.5/form.php
140	function	fread($fh, 50)	$fh		1	0	0	contact-form.2.7.5/form.php
140	variable	$theData	fread($fh, 50)		1	0	0	contact-form.2.7.5/form.php
143	function	fstr_replace("n", "", $theData)	$theData		1	0	0	contact-form.2.7.5/form.php
143	variable	$theData	fstr_replace("n", "", $theData)		1	0	1	contact-form.2.7.5/form.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
136	$stringData > “100”	1	0	1	contact-form.2.7.5/form.php

Slice:

L	Instruction	File
136	if ($stringData > ”100”) {	contact-form.2.7.5/form.php
137	function thecredits(){	contact-form.2.7.5/form.php
138	$myFile = CF_PLUGIN_BASE_DIR . ′/contact-form-wordpress/include/standard.txt′;	contact-form.2.7.5/form.php
139	$fh = fopen($myFile, ′r′);	contact-form.2.7.5/form.php
140	$theData = fread($fh, 50);	contact-form.2.7.5/form.php
143	$theData = str_replace(”n”, ””, $theData);	contact-form.2.7.5/form.php
144	echo ′Contact Form plugin created by ′;echo $the-Data;echo ′</small> ′;	contact-form.2.7.5/form.php

Name of the enclosing function: No function
phpSAFE’s decision: 1
RIPS’s decision: 1

F.3.15.2 Explanation

$myFile holds the value *CF_PLUGIN_BASE_DIR . ‘/contact-form-wordpress/include/standard.txt’;* in two instances of this file (lines 48 and 138). Line 51 writes rating values from *CF_PLUGIN_BASE_DIR . ‘/contact-form-wordpress/include/ratings.txt’* into $myFile. There are no other uses of the ratings file.

F.4 VV

- Due to PHP’s weak typing, constraints that involve numeric comparison can be bypassed by strings, as long as the first character is a number that satisfies the constraint. An example: *if($field_id > 0)* can be bypassed with the value *“2<script>alert(document.cookie);</script>”*. A tainted string remains tainted if its numeric constraint does not apply type checking (i.e. is_numeric)
- Regarding preg_replace[1]:

- [SQLI] If the third argument is tainted, and none of the special characters (\x00, \n, \r, \, ’, ” or \x1a) are replaced (i.e. they are not sought for in the regular expressions (blacklist) or are allowed by the whitelist), the taintedness remains.

- [XSS] If the third argument is tainted, and none of the special characters (<, >, &, ” and ‘) are replaced, the taintedness remains.

- In a ternary operation, taintedness is only propagated if the candidate value for the attribution is tainted.

F.5 Misc.

- wap-2.1 detected many (true) vulnerabilities, but no records exist in the dataset.

F.6 Sources

- [1]. `http://php.net/manual/en/function.preg-replace.php`

Appendix G

Plugin: easy2map.1.2.4

G.1 Type: XSS (reflected)

G.1.1 Slice #1

G.1.1.1 Verdict: 1 (vulnerable)

RIPS XSS Slice #265
phpSAFE XSS Slice #1309
Sensitive sink

- Name: die
- Line: 405
- File: easy2map.1.2.4/includes/Functions.php

Entry points:

L	Type	Name	R S D	File
405	REQUEST	$_REQUEST	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
405	function	Easy2Map _MapPinFunctions ::Save_map_pin ($_REQUEST)	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php
405	function	json_encode (Easy2Map _MapPinFunctions:: Save_map_pin ($_REQUEST))	Easy2Map_MapPinFunctions ::Save_map_pin($_REQUEST)	1	1	0	1	easy2map.1.2.4/includes/Functions.php
337	parameter	$Items	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php
349	variable	$mapPointID	$Items[“mapPointID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
345	isset($Items[“mapPointID”]) && (int) $Items[“mapPointID”] != 0	0		1	easy2map.1.2.4/includes/Functions.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
345	isset($Items[“mapPointID”]) && (int) $Items[“mapPointID”] != 0	0	0	1	easy2map.1.2.4/includes/Functions.php
351	!$wpdb->query($wpdb->prepare(” UPDATE $mapPointsTable SET LatLong = ‘%s’, PinImageURL = ‘%s’, Title = ‘%s’, Settings = ‘%s’, DetailsHTML = ‘%s’ WHERE ID = %s;“, $Items[’latLong’], urldecode($Items[’icon’]), $Items[’pinTitle’], $Items[’pinSettingsXML’], urldecode($Items[”pinHTML”]), $mapPointID))	0	1	0	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
404	public static function Save_map_pin() {	easy2map.1.2.4/includes/Functions.php
405	die(json_encode(Easy2Map_MapPinFunctions::Save_map_pin($_REQUEST)));	easy2map.1.2.4/includes/Functions.php
406	}	easy2map.1.2.4/includes/Functions.php
337	public static function Save_map_pin($Items) {	easy2map.1.2.4/includes/Functions.php
342	$mapPointsTable = $wpdb->prefix . ”easy2map_map_points”;	easy2map.1.2.4/includes/Functions.php
345	if (isset($Items[”mapPointID”]) && (int) $Items[”mapPointID”] != 0) {	easy2map.1.2.4/includes/Functions.php
349	$mapPointID = $Items[”mapPointID”];	easy2map.1.2.4/includes/Functions.php
351	if (!$wpdb->query($wpdb->prepare(”	easy2map.1.2.4/includes/Functions.php
352	UPDATE $mapPointsTable	easy2map.1.2.4/includes/Functions.php
353	SET LatLong = ′%s′, PinImageURL = ′%s′,	easy2map.1.2.4/includes/Functions.php
354	Title = ′%s′, Settings = ′%s′, DetailsHTML = ′%s′	easy2map.1.2.4/includes/Functions.php
355	WHERE ID = %s;”, $Items[′latLong′], urldecode($Items[′icon′]), $Items[′pinTitle′], $Items[′pinSettingsXML′], urldecode($Items[”pinHTML”]), $mapPointID))) {	easy2map.1.2.4/includes/Functions.php
356	return 0;	easy2map.1.2.4/includes/Functions.php
357	}	easy2map.1.2.4/includes/Functions.php
359	} else {	easy2map.1.2.4/includes/Functions.php
381	return $mapPointID;	easy2map.1.2.4/includes/Functions.php
382	}	easy2map.1.2.4/includes/Functions.php

Additional RIPS results——-

Alternative trigger files:

- N/A

Additional comment:
Userinput reaches sensitive sink when function save_map_pin() is called.

Name of the enclosing function:Save_map_pin()
RIPS's decision: 1
phpSAFE's decision: 1

G.1.1.2 Explanation

Vulnerable. In the if branch, $Items["mapPointID"] may be able to taint $mapPointID **as *(int)* will return a value != 0 if the payload is, for instance, *"4 "*** . *$mapPointID* is then used to filter the *ID* field; it seems that, in certain configurations (PHP + a MySQL driver), database queries that filter for numeric IDs will be successful if the provided input is a string that contains a valid, existing ID as its first character (see Misc. for more information). The format specifier used for ID, in line 355, is %s, so a payload like *"4 "* would be allowed. If a record with ID=4 exists on the database, the query will update that record; if at least one record is affected by this update, the constraint in line 351 fails. Line 381 returns this tainted value to *json_encode()*, which severely limits the variety of successful XSS attacks (for instance, it will ignore code inside a <script> element). However, the payload *"4 "* is able to bypass *json_encode()* untouched, as **scripts do not have to be inside script tags**. Finally, this payload is provided to *die()*, the sink of this vulnerability.

G.1.2 Slice #2

G.1.2.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1310
Sensitive sink

- Name: die
- Line: 413
- File: easy2map.1.2.4/includes/Functions.php

Entry points:

L	Type	Name	R S D	File
413	REQUEST	$_REQUEST	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
413	function	Easy2Map_MapFunctions ::Save_map($_REQUEST)	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php
413	function	json_encode (Easy2Map_MapFunctions:: Save_map($_REQUEST))	Easy2Map_MapFunctions:: Save_map($_REQUEST)	1	1	0	1	easy2map.1.2.4/includes/Functions.php
152	parameter	$Items	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php
157	variable	$mapID	$Items[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
160	intval($mapID) != 0	0		1	easy2map.1.2.4/includes/Functions.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
160	intval($mapID) != 0	0	0	1	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
412	public static function Save_map() {	easy2map.1.2.4/includes/Functions.php
413	die(json_encode(Easy2Map_MapFunctions::Save_map($_REQUEST)));	easy2map.1.2.4/includes/Functions.php
414	}	easy2map.1.2.4/includes/Functions.php
152	public static function Save_map($Items) {	easy2map.1.2.4/includes/Functions.php
157	$mapID = $Items[”mapID”];	easy2map.1.2.4/includes/Functions.php
158	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/Functions.php
160	if (intval($mapID) != 0) {	easy2map.1.2.4/includes/Functions.php
163	$wpdb->query(sprintf(”	easy2map.1.2.4/includes/Functions.php
164	UPDATE $mapsTable	easy2map.1.2.4/includes/Functions.php
165	SET TemplateID = ′%s′,	easy2map.1.2.4/includes/Functions.php
166	MapName = ′%s′,	easy2map.1.2.4/includes/Functions.php
167	Settings = ′%s′,	easy2map.1.2.4/includes/Functions.php
168	LastInvoked = CURRENT_TIMESTAMP,	easy2map.1.2.4/includes/Functions.php
169	CSSValues = ′%s′,	easy2map.1.2.4/includes/Functions.php
170	CSSValuesList = ′%s′,	easy2map.1.2.4/includes/Functions.php
171	CSSValuesHeading = ′%s′,	easy2map.1.2.4/includes/Functions.php
172	MapHTML = ′%s′,	easy2map.1.2.4/includes/Functions.php
173	IsActive = 1,	easy2map.1.2.4/includes/Functions.php
174	ThemeID = ′%s′	easy2map.1.2.4/includes/Functions.php
175	WHERE ID = %s;”,	easy2map.1.2.4/includes/Functions.php
176	$Items[′mapTemplateName′],	easy2map.1.2.4/includes/Functions.php
177	$Items[′mapName′],	easy2map.1.2.4/includes/Functions.php
178	urldecode($Items[′mapSettingsXML′]),	easy2map.1.2.4/includes/Functions.php
179	urldecode($Items[”mapCSSXML”]),	easy2map.1.2.4/includes/Functions.php
180	urldecode($Items[”listCSSXML”]),	easy2map.1.2.4/includes/Functions.php
181	urldecode($Items[”headingCSSXML”]),	easy2map.1.2.4/includes/Functions.php
182	urldecode($Items[”mapHTML”]),	easy2map.1.2.4/includes/Functions.php
183	$Items[′mapThemeName′],	easy2map.1.2.4/includes/Functions.php
184	$mapID));	easy2map.1.2.4/includes/Functions.php
185	} else {	easy2map.1.2.4/includes/Functions.php
259	return $mapID;	easy2map.1.2.4/includes/Functions.php
260	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Save_map()
phpSAFE's decision: 1

G.1.2.2 Explanation

Vulnerable. The constraint in line 160 uses *intval()*, which has a similar problem to *(int)*; it will return a value != 0 if the payload is, for instance, *“4 ”*. Regardless of the outcome for the UPDATE command, line 259 returns the tainted value to *json_encode()*, which severely limits the variety of successful XSS attacks (for instance, it will ignore code inside a <script> element). However, the payload *“4 ”* is able to bypass *json_encode()* untouched, as **scripts do not have to be inside script tags**. Finally, this payload is provided to *die()*, the sink of this vulnerability.

G.1.3 Slice #3

G.1.3.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1317
Sensitive sink

- Name: die
- Line: 485
- File: easy2map.1.2.4/includes/Functions.php

Entry points:

L	Type	Name	R S D	File
485	REQUEST	$_REQUEST[“mapID”]	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
485	function	Easy2Map_MapFunctions ::Retrieve_map_settings ($_REQUEST[“mapID”])	$_REQUEST[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php
485	function	json_encode (Easy2Map_MapFunctions:: Retrieve_map_settings ($_REQUEST[“mapID”]))	Easy2Map_MapFunctions ::Retrieve_map_settings ($_REQUEST[“mapID”])	1	1	0	1	easy2map.1.2.4/includes/Functions.php
104	parameter	$mapID	$_REQUEST[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
110	intval($mapID) === 0	0		0	easy2map.1.2.4/includes/Functions.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
110	intval($mapID) === 0	0	0	0	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
484	public static function Retrieve_map_settings_callback() {	easy2map.1.2.4/includes/Functions.php
485	die(json_encode(Easy2Map_MapFunctions::Retrieve_map_settings ($_REQUEST[”mapID”])));	easy2map.1.2.4/includes/Functions.php
486	}	easy2map.1.2.4/includes/Functions.php
104	public static function Retrieve_map_settings($mapID) {	easy2map.1.2.4/includes/Functions.php
107	$mapTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/Functions.php
108	$themesTable = $wpdb->prefix . ”easy2map_themes”;	easy2map.1.2.4/includes/Functions.php
110	if (intval($mapID) === 0) {	easy2map.1.2.4/includes/Functions.php
118	}	easy2map.1.2.4/includes/Functions.php
120	$mapSettings = $wpdb->get_results($wpdb->prepare(”SELECT $mapTable.*,	easy2map.1.2.4/includes/Functions.php
121	IFNULL($themesTable.Styles,′′) AS Styles	easy2map.1.2.4/includes/Functions.php
122	FROM $mapTable	easy2map.1.2.4/includes/Functions.php
123	LEFT JOIN $themesTable ON $mapTable.ThemeID = $themesTable.ID	easy2map.1.2.4/includes/Functions.php
124	WHERE $mapTable.ID = ′%s′;”, $mapID));	easy2map.1.2.4/includes/Functions.php
126	foreach ($mapSettings as $row) {	easy2map.1.2.4/includes/Functions.php
128	$settings = new e2mMapItem($row->ID, $row->TemplateID, $row->MapName, $row->DefaultPinImage, $row->Settings, $row->CSSValues, $row->CSSValuesList, $row->CSSValuesHeading, $row->PolyLines, $row->ThemeID, $row->Styles);	easy2map.1.2.4/includes/Functions.php
130	return $settings;	easy2map.1.2.4/includes/Functions.php
131	}	easy2map.1.2.4/includes/Functions.php
133	return null;	easy2map.1.2.4/includes/Functions.php
134	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Retrieve_map_settings_callback()
phpSAFE's decision: 1

G.1.3.2 Explanation

Not vulnerable to reflected XSS. The function will either return *null* or *$settings*. *$settings* is an object, which means *json_encode* will return a JSON structure that mirrors that object's properties.

G.1.4 Slice #4

G.1.4.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1318
Sensitive sink

- Name: die
- Line: 489
- File: easy2map.1.2.4/includes/Functions.php

Entry points:

L	Type	Name	R S D	File
489	REQUEST	$_REQUEST[“mapID”]	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
489	function	Easy2Map_MapFunctions ::Retrieve_map_templates ($_REQUEST[“mapID”])	$_REQUEST[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php
489	function	json_encode (Easy2Map_MapFunctions:: Retrieve_map_templates ($_REQUEST[“mapID”]))	Easy2Map_MapFunctions ::Retrieve_map_templates ($_REQUEST[“mapID”])	1	1	0	1	easy2map.1.2.4/includes/Functions.php
18	parameter	$mapID	$_REQUEST[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
488	public static function Retrieve_map_templates_callback() {	easy2map.1.2.4/includes/Functions.php
489	die(json_encode(Easy2Map_MapFunctions::Retrieve_map_templates ($_REQUEST[”mapID”])));	easy2map.1.2.4/includes/Functions.php
490	}	easy2map.1.2.4/includes/Functions.php
18	public static function Retrieve_map_templates($mapID) {	easy2map.1.2.4/includes/Functions.php
21	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/Functions.php
22	$templatesTable = $wpdb->prefix . ”easy2map_templates”;	easy2map.1.2.4/includes/Functions.php
23	$returnValue = array();	easy2map.1.2.4/includes/Functions.php
25	$templates = $wpdb->get_results($wpdb->prepare(”SELECT A.ID,	easy2map.1.2.4/includes/Functions.php
26	IFNULL(B.TemplateID,94) AS SelectedTemplate,	easy2map.1.2.4/includes/Functions.php
27	A.TemplateName	easy2map.1.2.4/includes/Functions.php
28	, A.ExampleImage	easy2map.1.2.4/includes/Functions.php
29	, IFNULL(B.CSSValues, A.CSSValues) AS CSSValues	easy2map.1.2.4/includes/Functions.php
30	, IFNULL(B.CSSValuesList, IFNULL(A.CSSValuesList,′′)) AS CSSValuesList	easy2map.1.2.4/includes/Functions.php
31	, IFNULL(B.CSSValuesHeading, IFNULL(A.CSSValuesHeading,′′)) AS CSSValuesHeading	easy2map.1.2.4/includes/Functions.php
32	, IFNULL(B.MapHTML, A.TemplateHTML) AS TemplateHTML	easy2map.1.2.4/includes/Functions.php
33	, IFNULL(A.StyleParentOnly,0) AS StyleParentOnly	easy2map.1.2.4/includes/Functions.php
34	FROM $templatesTable A	easy2map.1.2.4/includes/Functions.php
35	LEFT JOIN $mapsTable B ON (A.ID = B.TemplateID AND B.ID = %s)	easy2map.1.2.4/includes/Functions.php
36	WHERE A.Active = 1	easy2map.1.2.4/includes/Functions.php
37	ORDER BY A.DisplayOrder;”, $mapID));	easy2map.1.2.4/includes/Functions.php
39	foreach ($templates as $template) {	easy2map.1.2.4/includes/Functions.php
41	$mapTemplate = new e2mMapTemplate($template->ID,	easy2map.1.2.4/includes/Functions.php
42	$template->SelectedTemplate,	easy2map.1.2.4/includes/Functions.php
43	$template->TemplateName,	easy2map.1.2.4/includes/Functions.php
44	$template->ExampleImage,	easy2map.1.2.4/includes/Functions.php
45	stripcslashes($template->CSSValues),	easy2map.1.2.4/includes/Functions.php
46	stripcslashes($template->CSSValuesList),	easy2map.1.2.4/includes/Functions.php
47	stripcslashes($template->CSSValuesHeading),	easy2map.1.2.4/includes/Functions.php
48	stripcslashes($template->TemplateHTML),	easy2map.1.2.4/includes/Functions.php
49	$template->StyleParentOnly);	easy2map.1.2.4/includes/Functions.php
51	array_push($returnValue, $mapTemplate);	easy2map.1.2.4/includes/Functions.php
52	}	easy2map.1.2.4/includes/Functions.php
54	return $returnValue;	easy2map.1.2.4/includes/Functions.php
55	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Retrieve_map_templates_callback()
phpSAFE’s decision: 1

G.1.4.2 Explanation

Not vulnerable, as *$mapID* is used for filtering purposes in a prepared statement. The return is an array of *e2mMapTemplate* objects.

G.1.5 Slice #5

G.1.5.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1319
Sensitive sink

- Name: die
- Line: 493
- File: easy2map.1.2.4/includes/Functions.php

Entry points:

L	Type	Name	R S D	File
493	REQUEST	$_REQUEST[“mapID”]	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T / F	N t d	R S D	File
493	function	Easy2Map_MapFunctions ::Retrieve_map_themes ($_REQUEST[“mapID”])	$_REQUEST[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php
493	function	json_encode(Easy2Map _MapFunctions ::Retrieve_map_themes ($_REQUEST[“mapID”]))	Easy2Map_MapFunctions:: Retrieve_map_themes ($_REQUEST[“mapID”])	1	1	0	1	easy2map.1.2.4/includes/Functions.php
58	parameter	$mapID	$_REQUEST[“mapID”]	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
492	public static function Retrieve_map_themes_callback() {	easy2map.1.2.4/includes/Functions.php
493	die(json_encode(Easy2Map_MapFunctions:: Retrieve_map_themes($_REQUEST[”mapID”])));	easy2map.1.2.4/includes/Functions.php
494	}	easy2map.1.2.4/includes/Functions.php
58	public static function Retrieve_map_themes($mapID) {	easy2map.1.2.4/includes/Functions.php
61	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/Functions.php
62	$themesTable = $wpdb->prefix . ”easy2map_themes”;	easy2map.1.2.4/includes/Functions.php
63	$returnValue = array();	easy2map.1.2.4/includes/Functions.php
65	$themes = $wpdb->get_results($wpdb->prepare(”SELECT A.ID,	easy2map.1.2.4/includes/Functions.php
66	IFNULL(B.ThemeID,1) AS SelectedTheme,	easy2map.1.2.4/includes/Functions.php
67	A.ThemeName	easy2map.1.2.4/includes/Functions.php
68	, A.Styles	easy2map.1.2.4/includes/Functions.php
69	FROM $themesTable A	easy2map.1.2.4/includes/Functions.php
70	LEFT JOIN $mapsTable B ON (A.ID = B.ThemeID AND B.ID = %s)	easy2map.1.2.4/includes/Functions.php
71	ORDER BY A.ID;”, $mapID));	easy2map.1.2.4/includes/Functions.php
73	foreach ($themes as $theme) {	easy2map.1.2.4/includes/Functions.php
75	$mapTheme= new e2mMapTheme($theme->ID,	easy2map.1.2.4/includes/Functions.php
76	$theme->ThemeName,	easy2map.1.2.4/includes/Functions.php
77	$theme->Styles);	easy2map.1.2.4/includes/Functions.php
79	array_push($returnValue, $mapTheme);	easy2map.1.2.4/includes/Functions.php
80	}	easy2map.1.2.4/includes/Functions.php
82	return $returnValue;	easy2map.1.2.4/includes/Functions.php
83	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Retrieve_map_themes_callback()
phpSAFE's decision: 1

G.1.5.2 Explanation

Not vulnerable, as *$mapID* is used for filtering purposes in a prepared statement. The return is an array of *e2mMapTheme* objects.

G.1.6 Slice #6

G.1.6.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1320
Sensitive sink

- Name: echo
- Line: 113
- File: easy2map.1.2.4/includes/MapImport.php

Entry points:

L	Type	Name	R S D	File
9	GET	filter_input(INPUT_GET, 'map_id')	1	easy2map.1.2.4/includes/MapImport.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
28	intval($mapID) === 0	0	26	0	easy2map.1.2.4/includes/MapImport.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
16	is_uploaded_file($_FILES["xmlfile"]['tmp_name'])	1	0	1	easy2map.1.2.4/includes/MapImport.php
23	isset($xmlObject->map)	0	1	1	easy2map.1.2.4/includes/MapImport.php
26	!filter_input(INPUT_GET, 'markersonly')	0	1	1	easy2map.1.2.4/includes/MapImport.php
28	intval($mapID) === 0	0	1	0	easy2map.1.2.4/includes/MapImport.php

Slice:

L	Instruction	File
9	$mapID = filter_input(INPUT_GET, 'map_id');	easy2map.1.2.4/includes/MapImport.php
13	$mapsTable = $wpdb->prefix . "easy2map_maps";	easy2map.1.2.4/includes/MapImport.php
14	$markersTable = $wpdb->prefix . "easy2map_map_points";	easy2map.1.2.4/includes/MapImport.php
16	if (is_uploaded_file($_FILES["xmlfile"]['tmp_name'])) {	easy2map.1.2.4/includes/MapImport.php
18	try {	easy2map.1.2.4/includes/MapImport.php
21	$xmlObject = simplexml_load_string(file_get_contents ($_FILES["xmlfile"]['tmp_name']));	easy2map.1.2.4/includes/MapImport.php
23	if (isset($xmlObject->map)) {	easy2map.1.2.4/includes/MapImport.php
26	if (!filter_input(INPUT_GET, 'markersonly')) {	easy2map.1.2.4/includes/MapImport.php
28	if (intval($mapID) === 0) {	easy2map.1.2.4/includes/MapImport.php
63	$mapID = $row->NewMapID;	easy2map.1.2.4/includes/MapImport.php
66	} else {	easy2map.1.2.4/includes/MapImport.php
81	}	easy2map.1.2.4/includes/MapImport.php
82	}	easy2map.1.2.4/includes/MapImport.php
111	}	easy2map.1.2.4/includes/MapImport.php
113	echo '<script> jQuery(function() { window.location = "?page=easy2map&action=edit&map_id=' . $mapID . '";});</script>';	easy2map.1.2.4/includes/MapImport.php

Name of the enclosing function: No function
phpSAFE's decision: 1

G.1.6.2 Explanation

A vulnerability occurs if the constraint in line 28 is negated. *$mapID* remains untouched and tainted, reaching a sensitive sink.

G.1.7 Slice #7

G.1.7.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1321
Sensitive sink

- Name: echo
- Line: 126
- File: easy2map.1.2.4/includes/MapImport.php

Entry points:

L	Type	Name	RSD	File
9	GET	filter_input(INPUT_GET, 'map_id')	1	easy2map.1.2.4/includes/MapImport.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
28	intval($mapID) === 0	0	26	0	easy2map.1.2.4/includes/MapImport.php

Execution path constraints

L	Condition	ES	Nt d	T/F	File
16	is_uploaded_file($_FILES[“xmlfile”][’tmp_name’])	0	0	1	easy2map.1.2.4/includes/MapImport.php
23	isset($xmlObject->map)	0	1	1	easy2map.1.2.4/includes/MapImport.php
26	!filter_input(INPUT_GET, ‘markersonly’)	0	1	1	easy2map.1.2.4/includes/MapImport.php
28	intval($mapID) === 0	0	1	0	easy2map.1.2.4/includes/MapImport.php
125	!filter_input(INPUT_GET, ‘markersonly’)	1	0	1	easy2map.1.2.4/includes/MapImport.php

Slice:

L	Instruction	File
9	$mapID = filter_input(INPUT_GET, ′map_id′);	easy2map.1.2.4/includes/MapImport.php
13	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/MapImport.php
14	$markersTable = $wpdb->prefix . ”easy2map_map_points”;	easy2map.1.2.4/includes/MapImport.php
16	if (is_uploaded_file($_FILES[”xmlfile”][′tmp_name′])) {	easy2map.1.2.4/includes/MapImport.php
18	try {	easy2map.1.2.4/includes/MapImport.php
21	$xmlObject = simplexml_load_string(file_get_contents ($_FILES[”xmlfile”][′tmp_name′]));	easy2map.1.2.4/includes/MapImport.php
23	if (isset($xmlObject->map)) {	easy2map.1.2.4/includes/MapImport.php
26	if (!filter_input(INPUT_GET, ′markersonly′)) {	easy2map.1.2.4/includes/MapImport.php
28	if (intval($mapID) === 0) {	easy2map.1.2.4/includes/MapImport.php
63	$mapID = $row->NewMapID;	easy2map.1.2.4/includes/MapImport.php
66	} else {	easy2map.1.2.4/includes/MapImport.php
81	}	easy2map.1.2.4/includes/MapImport.php
82	}	easy2map.1.2.4/includes/MapImport.php
111	}	easy2map.1.2.4/includes/MapImport.php
117	}	easy2map.1.2.4/includes/MapImport.php
125	<?php if (!filter_input(INPUT_GET, ′markersonly′)) { ?>	easy2map.1.2.4/includes/MapImport.php
126	action=”?page=easy2map&action=mapimport&map_id=<?php echo $mapID; ?>”	easy2map.1.2.4/includes/MapImport.php

Name of the enclosing function: No function
phpSAFE’s decision: 1

G.1.7.2 Explanation

A vulnerability occurs if the constraint in line 28 is negated. *$mapID* remains untouched and tainted, reaching a sensitive sink.

G.1.8 Slice #8

G.1.8.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1329
Sensitive sink

- Name: echo
- Line: 128
- File: easy2map.1.2.4/includes/MapImport.php

Entry points:

L	Type	Name	RSD	File
9	GET	filter_input(INPUT_GET, ‘map_id’)	1	easy2map.1.2.4/includes/MapImport.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
28	intval($mapID) === 0	0	26	0	easy2map.1.2.4/includes/MapImport.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
16	is_uploaded_file($_FILES[“xmlfile”][’tmp_name’])	0	0	1	easy2map.1.2.4/includes/MapImport.php
23	isset($xmlObject->map)	0	1	1	easy2map.1.2.4/includes/MapImport.php
26	!filter_input(INPUT_GET, ‘markersonly’)	0	1	0	easy2map.1.2.4/includes/MapImport.php
125	!filter_input(INPUT_GET, ‘markersonly’)	1	0	0	easy2map.1.2.4/includes/MapImport.php

Slice:

L	Instruction	File
9	$mapID = filter_input(INPUT_GET, ′map_id′);	easy2map.1.2.4/includes/MapImport.php
13	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/MapImport.php
14	$markersTable = $wpdb->prefix . ”easy2map_map_points”;	easy2map.1.2.4/includes/MapImport.php
16	if (is_uploaded_file($_FILES[”xmlfile”][′tmp_name′])) {	easy2map.1.2.4/includes/MapImport.php
18	try {	easy2map.1.2.4/includes/MapImport.php
21	$xmlObject = simplexml_load_string(file_get_contents ($_FILES[”xmlfile”][′tmp_name′]));	easy2map.1.2.4/includes/MapImport.php
23	if (isset($xmlObject->map)) {	easy2map.1.2.4/includes/MapImport.php
26	if (!filter_input(INPUT_GET, ′markersonly′)) {	easy2map.1.2.4/includes/MapImport.php
82	}	easy2map.1.2.4/includes/MapImport.php
111	}	easy2map.1.2.4/includes/MapImport.php
117	}	easy2map.1.2.4/includes/MapImport.php
127	<?php } else { ?>	easy2map.1.2.4/includes/MapImport.php
128	action=”?page=easy2map&action=mapimport&markersonly =true&map_id=<?php echo $mapID; ?>”	easy2map.1.2.4/includes/MapImport.php

Name of the enclosing function: No function

phpSAFE’s decision: 1

G.1.8.2 Explanation

Tainted input reaches sensitive sink

G.1.9 Slice #9

G.1.9.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1342
wap XSS Slice #54
RIPS XSS Slice #267
Sensitive sink

- Name: echo
- Line: 45
- File: easy2map.1.2.4/includes/MapPinImageSave.php

Entry points:

L	Type	Name	RSD	File
28	GET	$_GET[“map_id”]	0	easy2map.1.2.4/includes/MapPinImageSave.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
28	variable	$imagePlusLocation	$_GET[“map_id”]	1	1	0	1	easy2map.1.2.4/includes/MapPinImageSave.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
11	is_uploaded_file($_FILES[“pinicon”][’tmp_name’])	0	0	1	easy2map.1.2.4/includes/MapPinImageSave.php
26	$_FILES[“pinicon”][’size’] < 5000000	0	1	1	easy2map.1.2.4/includes/MapPinImageSave.php
41	strlen($errorMessage) > 0	1	0	0	easy2map.1.2.4/includes/MapPinImageSave.php

Slice:

L	Instruction	File
7	$errorMessage = ””;	easy2map.1.2.4/includes/MapPinImageSave.php
11	if (is_uploaded_file($_FILES[”pinicon”][′tmp_name′])) {	easy2map.1.2.4/includes/MapPinImageSave.php
26	if ($_FILES[”pinicon”][′size′] < 5000000) {	easy2map.1.2.4/includes/MapPinImageSave.php
28	$imagePlusLocation = WP_CONTENT_URL . ”/uploads/easy2map/images/map_pins/uploaded/” . $_GET[”map_id”] . ”/” . $arrSmallImage[0];	easy2map.1.2.4/includes/MapPinImageSave.php
30	}	easy2map.1.2.4/includes/MapPinImageSave.php
31	}	easy2map.1.2.4/includes/MapPinImageSave.php
43	<?php } else { ?>	easy2map.1.2.4/includes/MapPinImageSave.php
45	window.parent.jQuery(′#draggable′).attr(′src′, ′<?php echo $imagePlusLocation; ?>′);	easy2map.1.2.4/includes/MapPinImageSave.php

Misc. comparisons

- wap_ori: y
- pixy: 3
- pixy_ori: fn

Additional RIPS results———-
Alternative trigger files:

- N/A

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS's decision: 1
wap's decision: 1
phpSAFE's decision: 1

G.1.9.2 Explanation

Tainted input reaches sensitive sink.

G.1.10 Slice #10

G.1.10.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1350
wap XSS Slice #55
RIPS XSS Slice #266
Sensitive sink

- Name: echo
- Line: 5
- File: easy2map.1.2.4/includes/MapPinImageSave.php

Entry points:

L	Type	Name	R S D	File
4	GET	$_GET[“map_id”]	0	easy2map.1.2.4/includes/MapPinImageSave.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
4	variable	$imagesDirectory	$_GET[“map_id”]	1	1	0	1	easy2map.1.2.4/includes/MapPinImageSave.php

Slice:

L	Instruction	File
4	$imagesDirectory = WP_CONTENT_DIR . ”/uploads/easy2map/images/map_pins/uploaded/” . $_GET[”map_id”] . ”/”;	easy2map.1.2.4/includes/MapPinImageSave.php
5	echo $imagesDirectory;	easy2map.1.2.4/includes/MapPinImageSave.php

Misc. comparisons

- wap_ori: y
- pixy: 3
- pixy_ori: fn

Additional RIPS results——-
Alternative trigger files:

- N/A

Additional comment:
Userinput reaches sensitive sink. For more information, press the help icon on the left side.

———————————————

Name of the enclosing function: No function
RIPS’s decision: 1
wap’s decision: 1
phpSAFE’s decision: 1

G.1.10.2 Explanation

Tainted input reaches sensitive sink.

———————————————

G.1.11 Slice #11

G.1.11.1 Verdict: 0 (not vulnerable)

wap XSS Slice #53
Sensitive sink

- Name: file_get_contents
- Line: 21
- File: easy2map.1.2.4/includes/MapImport.php

Entry points:

L	Type	Name	RSD	File
21	FILES	$_FILES[“xmlfile”][’tmp_name’]	1	easy2map.1.2.4/includes/MapImport.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
16	is_uploaded_file ($_FILES[“xmlfile”][’tmp_name’])	1		1	easy2map.1.2.4/includes/MapImport.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
16	is_uploaded_file($_FILES[“xmlfile”][’tmp_name’])	1	0	1	easy2map.1.2.4/includes/MapImport.php

Slice:

L	Instruction	File
16	if (is_uploaded_file($_FILES[”xmlfile”][′tmp_name′])) {	easy2map.1.2.4/includes/MapImport.php
21	$xmlObject = simplexml_load_string(file_get_contents ($_FILES[”xmlfile”][′tmp_name′]));	easy2map.1.2.4/includes/MapImport.php

Name of the enclosing function: No function
wap’s decision: 1

G.1.11.2 Explanation

Not vulnerable to reflected XSS, as its contents are not reflected back.

G.2 Type: Stored injection

G.2.1 Slice #12

G.2.1.1 Verdict: 1 (vulnerable)

phpSAFE XSS Slice #1330
Sensitive sink

- Name: echo
- Line: 104
- File: easy2map.1.2.4/includes/MapManager.php

WRITES (1st step)
Entry points:

L	Type	Name	RSD	File
413	REQUEST	$_REQUEST	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
413	function	Easy2Map_MapFunctions::Save_map($_REQUEST)	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php
413	function	json_encode(Easy2Map_MapFunctions::Save_map($_REQUEST))	Easy2Map_MapFunctions::Save_map($_REQUEST)	1	1	0	1	easy2map.1.2.4/includes/Functions.php
152	parameter	$Items	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
160	intval($mapID) != 0	0	0	1	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
412	public static function Save_map() {	easy2map.1.2.4/includes/Functions.php
413	die(json_encode(Easy2Map_MapFunctions::Save_map($_REQUEST)));	easy2map.1.2.4/includes/Functions.php
414	}	easy2map.1.2.4/includes/Functions.php
152	public static function Save_map($Items) {	easy2map.1.2.4/includes/Functions.php
157	$mapID = $Items[”mapID”];	easy2map.1.2.4/includes/Functions.php
158	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/Functions.php
160	if (intval($mapID) != 0) {	easy2map.1.2.4/includes/Functions.php
163	$wpdb->query(sprintf(”	easy2map.1.2.4/includes/Functions.php
164	UPDATE $mapsTable	easy2map.1.2.4/includes/Functions.php
165	SET TemplateID = ′%s′,	easy2map.1.2.4/includes/Functions.php
166	MapName = ′%s′,	easy2map.1.2.4/includes/Functions.php
167	Settings = ′%s′,	easy2map.1.2.4/includes/Functions.php
168	LastInvoked = CURRENT_TIMESTAMP,	easy2map.1.2.4/includes/Functions.php
169	CSSValues = ′%s′,	easy2map.1.2.4/includes/Functions.php
170	CSSValuesList = ′%s′,	easy2map.1.2.4/includes/Functions.php
171	CSSValuesHeading = ′%s′,	easy2map.1.2.4/includes/Functions.php
172	MapHTML = ′%s′,	easy2map.1.2.4/includes/Functions.php
173	IsActive = 1,	easy2map.1.2.4/includes/Functions.php
174	ThemeID = ′%s′	easy2map.1.2.4/includes/Functions.php
175	WHERE ID = %s;”,	easy2map.1.2.4/includes/Functions.php
176	$Items[′mapTemplateName′],	easy2map.1.2.4/includes/Functions.php
177	$Items[′mapName′],	easy2map.1.2.4/includes/Functions.php
178	urldecode($Items[′mapSettingsXML′]),	easy2map.1.2.4/includes/Functions.php
179	urldecode($Items[”mapCSSXML”]),	easy2map.1.2.4/includes/Functions.php
180	urldecode($Items[”listCSSXML”]),	easy2map.1.2.4/includes/Functions.php
181	urldecode($Items[”headingCSSXML”]),	easy2map.1.2.4/includes/Functions.php
182	urldecode($Items[”mapHTML”]),	easy2map.1.2.4/includes/Functions.php
183	$Items[′mapThemeName′],	easy2map.1.2.4/includes/Functions.php
184	$mapID));	easy2map.1.2.4/includes/Functions.php
185	} else {	easy2map.1.2.4/includes/Functions.php
259	return $mapID;	easy2map.1.2.4/includes/Functions.php
260	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Save_map()

READS (2nd step)

Entry points:

L	Type	Name	RSD	File
86	BD	$wpdb->get_results(“SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”)	0	easy2map.1.2.4/includes/MapManager.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint cond.	T/F	Ntd	RSD	File
86	variable	$results	$wpdb->get_results(“SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”)	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
89	variable	$result	$results	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
91	function	stripslashes ($result->MapName)	$result->MapName	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
91	variable	$name	stripslashes($result->MapName)	1	1	0	1	easy2map.1.2.4/includes/MapManager.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
89	foreach ($results as $result)	1		1	easy2map.1.2.4/includes/MapManager.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
89	foreach ($results as $result)	1	0	1	easy2map.1.2.4/includes/MapManager.php

Slice:

L	Instruction	File
46	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/MapManager.php
86	$results = $wpdb->get_results(”SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”);	easy2map.1.2.4/includes/MapManager.php
89	foreach ($results as $result) {	easy2map.1.2.4/includes/MapManager.php
91	$name = stripslashes($result->MapName);	easy2map.1.2.4/includes/MapManager.php
104	<td style=”width:30%;font-size:16px;font-weight:bold;”><?php echo $name; ?></td>	easy2map.1.2.4/includes/MapManager.php

Name of the enclosing function: No function
phpSAFE’s decision: 1

G.2.1.2 Explanation

The field *MapName* from the *$wpdb->prefix . “easy2map_maps”* table is vulnerable to stored XSS.

G.2.2 Slice #13

G.2.2.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1331
Sensitive sink

- Name: echo
- Line: 105
- File: easy2map.1.2.4/includes/MapManager.php

WRITES (1st step)
None of the writes affect the *ID* field.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
86	BD	$wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;")	0	easy2map.1.2.4/includes/MapManager.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	Ntd	RSD	File
86	variable	$results	$wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;")	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
89	variable	$result	$results	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
90	variable	$id	$result->ID	0	1	0	1	easy2map.1.2.4/includes/MapManager.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
89	foreach ($results as $result)	1		1	easy2map.1.2.4/includes/MapManager.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
89	foreach ($results as $result)	1	0	1	easy2map.1.2.4/includes/MapManager.php

Slice:

L	Instruction	File
46	$mapsTable = $wpdb->prefix . "easy2map_maps";	easy2map.1.2.4/includes/MapManager.php
86	$results = $wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;");	easy2map.1.2.4/includes/MapManager.php
89	foreach ($results as $result) {	easy2map.1.2.4/includes/MapManager.php
90	$id = $result->ID;	easy2map.1.2.4/includes/MapManager.php
105	<td nowrap><p nowrap style="text-align:center;font-size:1.2em;color:#033c90;padding:5px;background-color:#e7e7e7;border:1px solid #5b86c5;border-radius:3px;width:180px;">[easy2map id="<?php echo $id; ?>"]</p>	easy2map.1.2.4/includes/MapManager.php

Name of the enclosing function: No function
phpSAFE's decision: 1

G.2.2.2 Explanation

False positive. None of the writes affect the *ID* field.

G.2.3 Slice #14

G.2.3.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1332
Sensitive sink

- Name: echo
- Line: 107
- File: easy2map.1.2.4/includes/MapManager.php

WRITES (1st step)
None of the writes affect the *ID* field.
READS (2nd step)
Entry points:

L	Type	Name	RSD	File
86	BD	$wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;")	0	easy2map.1.2.4/includes/MapManager.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T/F	Ntd	RSD	File
86	variable	$results	$wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;")	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
89	variable	$result	$results	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
90	variable	$id	$result->ID	0	1	0	1	easy2map.1.2.4/includes/MapManager.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
89	foreach ($results as $result)	1		1	easy2map.1.2.4/includes/MapManager.php

Execution path constraints

L	Condition	ES	Ntd	T/F	File
89	foreach ($results as $result)	1	0	1	easy2map.1.2.4/includes/MapManager.php

Slice:

L	Instruction	File
46	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/MapManager.php
86	$results = $wpdb->get_results(”SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”);	easy2map.1.2.4/includes/MapManager.php
89	foreach ($results as $result) {	easy2map.1.2.4/includes/MapManager.php
90	$id = $result->ID;	easy2map.1.2.4/includes/MapManager.php
107	<td style=”width:15%;text-align:center;vertical-align:middle;”><a href=”?page=easy2map&action=edit&map_id=<?php echo $id; ?>”>	easy2map.1.2.4/includes/MapManager.php

Name of the enclosing function: No function
phpSAFE's decision: 1

G.2.3.2 Explanation

False positive. None of the writes affect the *ID* field.

G.2.4 Slice #15

G.2.4.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1340
pixy XSS Slice #277
Sensitive sink

- Name: echo
- Line: 109
- File: easy2map.1.2.4/includes/MapManager.php

WRITES (1st step)
None of the writes affect the *ID* field.
READS (2nd step)
Entry points:

L	Type	Name	R S D	File
86	BD	$wpdb->get_results(“SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”)	0	easy2map.1.2.4/includes/MapManager.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
86	variable	$results	$wpdb->get_results(“SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”)	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
89	variable	$result	$results	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
90	variable	$id	$result->ID	0	1	0	1	easy2map.1.2.4/includes/MapManager.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T/F	File
89	foreach ($results as $result)	1		1	easy2map.1.2.4/includes/MapManager.php

Execution path constraints

L	Condition	ES	N t d	T/F	File
89	foreach ($results as $result)	1	0	1	easy2map.1.2.4/includes/MapManager.php

Slice:

L	Instruction	File
46	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/MapManager.php
86	$results = $wpdb->get_results(”SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;”);	easy2map.1.2.4/includes/MapManager.php
89	foreach ($results as $result) {	easy2map.1.2.4/includes/MapManager.php
90	$id = $result->ID;	easy2map.1.2.4/includes/MapManager.php
109	<td style=”width:15%;text-align:center;vertical-align:middle;”><a onclick=”areYouSure(<?php echo $id; ?>);” href=”#”><img src=”<?php echo easy2map_get_plugin_url(′/images/e2m_icon_delete.png′); ?>”></td>	easy2map.1.2.4/includes/MapManager.php

Name of the enclosing function: No function
phpSAFE’s decision: 1
pixy’s decision: 1

G.2.4.2 Explanation

False positive. None of the writes affect the *ID* field.

G.2.5 Slice #16

G.2.5.1 Verdict: 0 (not vulnerable)

phpSAFE XSS Slice #1341
Sensitive sink

- Name: echo
- Line: 96
- File: easy2map.1.2.4/includes/MapManager.php

WRITES (1st step)
None of the writes affect the *ID* field.
READS (2nd step)

Entry points:

L	Type	Name	RSD	File
86	BD	$wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;")	0	easy2map.1.2.4/includes/MapManager.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	RSD	File
86	variable	$results	$wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;")	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
89	variable	$result	$results	1	1	0	0	easy2map.1.2.4/includes/MapManager.php
90	variable	$id	$result->ID	0	1	0	1	easy2map.1.2.4/includes/MapManager.php

Entry Point Validation Constraints

L	Condition	ES	Nested by	T / F	File
89	foreach ($results as $result)	1		1	easy2map.1.2.4/includes/MapManager.php

Execution path constraints

L	Condition	ES	N t d	T / F	File
89	foreach ($results as $result)	1	0	1	easy2map.1.2.4/includes/MapManager.php

Slice:

L	Instruction	File
46	$mapsTable = $wpdb->prefix . "easy2map_maps";	easy2map.1.2.4/includes/MapManager.php
86	$results = $wpdb->get_results("SELECT * FROM $mapsTable WHERE IsActive = 1 ORDER BY LastInvoked DESC;");	easy2map.1.2.4/includes/MapManager.php
89	foreach ($results as $result) {	easy2map.1.2.4/includes/MapManager.php
90	$id = $result->ID;	easy2map.1.2.4/includes/MapManager.php
96	<tr id="trMap<?php echo $id; ?>">	easy2map.1.2.4/includes/MapManager.php

Name of the enclosing function: No function
phpSAFE's decision: 1

G.2.5.2 Explanation

False positive. None of the writes affect the *ID* field.

G.2.6 Slice #17

G.2.6.1 Verdict: 1 (vulnerable)

Sensitive sink

- Name: die
- Line: 485
- File: easy2map.1.2.4/includes/MapManager.php

WRITES (1st step)

Entry points:

L	Type	Name	R S D	File
413	REQUEST	$_REQUEST	0	easy2map.1.2.4/includes/Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
413	function	Easy2Map _MapFunctions ::Save_map($_REQUEST)	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php
413	function	json_encode(Easy2Map _MapFunctions ::Save_map($_REQUEST))	Easy2Map _MapFunctions ::Save_map($_REQUEST)	1	1	0	1	easy2map.1.2.4/includes/Functions.php
152	parameter	$Items	$_REQUEST	1	1	0	0	easy2map.1.2.4/includes/Functions.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
160	intval($mapID) != 0	0	0	1	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
412	public static function Save_map() {	easy2map.1.2.4/includes/Functions.php
413	die(json_encode(Easy2Map_MapFunctions::Save_map($_REQUEST)));	easy2map.1.2.4/includes/Functions.php
414	}	easy2map.1.2.4/includes/Functions.php
152	public static function Save_map($Items) {	easy2map.1.2.4/includes/Functions.php
157	$mapID = $Items[”mapID”];	easy2map.1.2.4/includes/Functions.php
158	$mapsTable = $wpdb->prefix . ”easy2map_maps”;	easy2map.1.2.4/includes/Functions.php
160	if (intval($mapID) != 0) {	easy2map.1.2.4/includes/Functions.php
163	$wpdb->query(sprintf(”	easy2map.1.2.4/includes/Functions.php
164	UPDATE $mapsTable	easy2map.1.2.4/includes/Functions.php
165	SET TemplateID = ′%s′,	easy2map.1.2.4/includes/Functions.php
166	MapName = ′%s′,	easy2map.1.2.4/includes/Functions.php
167	Settings = ′%s′,	easy2map.1.2.4/includes/Functions.php
168	LastInvoked = CURRENT_TIMESTAMP,	easy2map.1.2.4/includes/Functions.php
169	CSSValues = ′%s′,	easy2map.1.2.4/includes/Functions.php
170	CSSValuesList = ′%s′,	easy2map.1.2.4/includes/Functions.php
171	CSSValuesHeading = ′%s′,	easy2map.1.2.4/includes/Functions.php
172	MapHTML = ′%s′,	easy2map.1.2.4/includes/Functions.php
173	IsActive = 1,	easy2map.1.2.4/includes/Functions.php
174	ThemeID = ′%s′	easy2map.1.2.4/includes/Functions.php
175	WHERE ID = %s;”,	easy2map.1.2.4/includes/Functions.php
176	$Items[′mapTemplateName′],	easy2map.1.2.4/includes/Functions.php
177	$Items[′mapName′],	easy2map.1.2.4/includes/Functions.php
178	urldecode($Items[′mapSettingsXML′]),	easy2map.1.2.4/includes/Functions.php
179	urldecode($Items[”mapCSSXML”]),	easy2map.1.2.4/includes/Functions.php
180	urldecode($Items[”listCSSXML”]),	easy2map.1.2.4/includes/Functions.php
181	urldecode($Items[”headingCSSXML”]),	easy2map.1.2.4/includes/Functions.php
182	urldecode($Items[”mapHTML”]),	easy2map.1.2.4/includes/Functions.php
183	$Items[′mapThemeName′],	easy2map.1.2.4/includes/Functions.php
184	$mapID));	easy2map.1.2.4/includes/Functions.php
185	} else {	easy2map.1.2.4/includes/Functions.php
259	return $mapID;	easy2map.1.2.4/includes/Functions.php
260	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Save_map()

READS (2nd step)

Entry points:

L	Type	Name	R S D	File
120	BD	$wpdb->get_results($wpdb->prepare(“SELECT $mapTable.*, IFNULL($themesTable.Styles,’‘) AS Styles FROM $mapTable LEFT JOIN $themesTable ON $mapTable.ThemeID = $themesTable.ID WHERE $mapTable.ID = ’%s’;”, $mapID))	0	easy2map.1.2.4/ includes/ Functions.php

Entry point dependencies:

L	Type	Name	Tainted by	Taint condition	T / F	N t d	R S D	File
120	variable	$mapSettings	$wpdb->get_results($wpdb->prepare("SELECT $mapTable.*, IFNULL($themesTable.Styles,'') AS Styles FROM $mapTable LEFT JOIN $themesTable ON $mapTable.ThemeID = $themesTable.ID WHERE $mapTable.ID = '%s';", $mapID))	0	1	0	0	easy2map.1.2.4/ includes/ Functions.php
126	variable	$row	$mapSettings	0	1	0	0	easy2map.1.2.4/ includes/ Functions.php
128	variable	$row->MapName	$row->MapName	1	1	0	0	easy2map.1.2.4/ includes/ Functions.php
128	variable	$settings	$row->MapName	1	1	0	1	easy2map.1.2.4/ includes/ Functions.php

Entry Point Validation Constraints

L	Condition	E S	Nested by	T / F	File
126	foreach ($mapSettings as $row)	0		1	easy2map.1.2.4/includes/Functions.php

Execution path constraints

L	Condition	E S	N t d	T / F	File
110	intval($mapID) === 0	0	0	0	easy2map.1.2.4/includes/Functions.php
126	foreach ($mapSettings as $row)	0	0	1	easy2map.1.2.4/includes/Functions.php

Slice:

L	Instruction	File
484	public static function Retrieve_map_settings_callback() {	easy2map.1.2.4/includes/Functions.php
485	die(json_encode(Easy2Map_MapFunctions::Retrieve_map_settings ($_REQUEST["mapID"])));	easy2map.1.2.4/includes/Functions.php
486	}	easy2map.1.2.4/includes/Functions.php
104	public static function Retrieve_map_settings($mapID) {	easy2map.1.2.4/includes/Functions.php
107	$mapTable = $wpdb->prefix . "easy2map_maps";	easy2map.1.2.4/includes/Functions.php
108	$themesTable = $wpdb->prefix . "easy2map_themes";	easy2map.1.2.4/includes/Functions.php
110	if (intval($mapID) === 0) {	easy2map.1.2.4/includes/Functions.php
118	}	easy2map.1.2.4/includes/Functions.php
120	$mapSettings = $wpdb->get_results($wpdb->prepare("SELECT $mapTable.*,	easy2map.1.2.4/includes/Functions.php
121	IFNULL($themesTable.Styles,'') AS Styles	easy2map.1.2.4/includes/Functions.php
122	FROM $mapTable	easy2map.1.2.4/includes/Functions.php
123	LEFT JOIN $themesTable ON $mapTable.ThemeID = $themesTable.ID	easy2map.1.2.4/includes/Functions.php
124	WHERE $mapTable.ID = '%s';", $mapID));	easy2map.1.2.4/includes/Functions.php
126	foreach ($mapSettings as $row) {	easy2map.1.2.4/includes/Functions.php
128	$settings = new e2mMapItem($row->ID, $row->TemplateID, $row->MapName, $row->DefaultPinImage, $row->Settings, $row->CSSValues, $row->CSSValuesList, $row->CSSValuesHeading, $row->PolyLines, $row->ThemeID, $row->Styles);	easy2map.1.2.4/includes/Functions.php
130	return $settings;	easy2map.1.2.4/includes/Functions.php
131	}	easy2map.1.2.4/includes/Functions.php
133	return null;	easy2map.1.2.4/includes/Functions.php
134	}	easy2map.1.2.4/includes/Functions.php

Name of the enclosing function:Retrieve_map_settings_callback()

G.2.6.2 Explanation

JSON object returned by *Retrieve_map_settings* contains tainted values. *$mapID* needs to contain the same value in both steps. False negative for all tools.

G.3 VV

- The entry point comes through a native PHP variable for dealing with HTTP requests: $_GET, $_POST, **SOME $_SERVER** values[1], and $_FILES. The entry point may also come as a function parameter. If the variable is an array, we will assume at least one of the elements is tainted.
- [SQLI] sprintf (native php) is being used with unsanitized tainted inputs and non-numeric format specifiers, and its output is input to a sensitive sink.
- Tainted inputs that are checked against validation constraints such as (int) or intval() remain tainted.

G.4 WAP false negatives

- WAP does not consider *filter_input*[2] as an entry point.

G.5 Misc.

- The following files do not exist:

- easy2map.1.2.4/includes/forms/delete.php
- easy2map.1.2.4/includes/forms/position.php

- (See Slice#1) Some database configurations seem to retrieve results based on query predicates meant to filter integer ID values, but with the given argument being a string in which the first character is a valid, existing record ID, ignoring the rest of the argument. The same payload given in Slice #1 was tested directly in phpMyAdmin. Both of the following examples retrieve the same result:

- SELECT * FROM ‘wp_easy2map_map_points‘ WHERE ID=4
- SELECT * FROM ‘wp_easy2map_map_points‘ WHERE ID=“4 ”

G.6 Sources

- [1]. `http://stackoverflow.com/questions/6474783/which-server-variables-are-safe`
- [2]. `http://php.net/manual/en/function.filter-input.php`