UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

ZeroDays v2
Sistema de Gestão de Ameaças de Cibersegurança Dia-Zero (Exploits e Software Malicioso Dia-Zero)

Tomás Rafael Martins Ferreira

Mestrado em Segurança Informática

Versão Provisória

Trabalho de Projeto orientado por:
Prof. Doutor Nuno Fuentecilla Maia Ferreira Neves
Eng. José António dos Santos Alegria

2022

”Um passo para trás, depois de fazer uma
curva errada, é um passo na direção certa.”

- Kurt Vonnegut

Agradecimentos

Se estas palavras estão escritas, são completamente devidas aos meus pais e aos meus avós, que embora não sejam responsáveis por este trabalho, são sim responsáveis pela minha maneira de pensar devido à educação e aos valores que me deram. Essa educação e valores que sempre permitiram que eu interrogasse o porquê das coisas, e assim dando-me as ferramentas para me dotarem da capacidade de aprender sozinho e de questionar a razão das coisas. Esta educação não foi exclusivamente dada pelos meus pais, mas também pela minha irmã, que foi-me dando sempre outros pontos de vista, sendo que me ensinou a analisar as questões no seu todo e não num único prisma. Também devo um agradecimento ao meu cunhado e ao meu sobrinho por me ajudarem a simplificar os problemas que lhes trago, e ajudam a consolidar as minhas ideias. Este trabalho é tanto meu como deles todos, pelo que lhes devo um especial obrigado.

Devo também um grande agradecimento à equipa da direção de cibersegurança da Altice Portugal, que possibilitou este projeto. Sem eles, este trabalho não teria sido possível. Quero agradecer à Sara Nascimento, que me ajudou muito durante a concretização do projeto; Ao Eng.º Jorge Silva, pelo suporte e disponibilidade que sempre teve para me ajudar, e foi essencial na minha integração na Altice, e na DCY em particular; Ao Eng.º José Alegria, meu coorientador, pela confiança que teve em mim, até nos momentos em que menos a tive. Estas são algumas das pessoas que me ajudaram muito neste projeto, mas estes agradecimentos são extensíveis a toda a DCY.

Devo também um especial agradecimento ao Professor Nuno Neves, pela disponibilidade em ajudar em todas as etapas do projeto, mas acima de tudo, ao seu conhecimento e à sua sensibilidade, que me ofereceu mais liberdade quando precisava dela, mais suporte quando era preciso, e que me deu as ferramentas para me tornar um melhor profissional de cibersegurança.

Por último, agradeço também aos meus amigos, que me acompanharam e apoiaram durante esta fase da minha vida, e sem eles este projeto teria sido muito mais difícil.

Abstract

In 2020, in order to contain the spread of the COVID-19 virus, the government declared a state of emergency. One of the measures applied was the adoption of remote work whenever possible. The growth of the number of employees working from home increased the amount of cyberattacks suffered by companies. To diminish these attacks, in 2021 the development of the ZeroDays v1 system began, which had the purpose of collecting information about zeroday vulnerabilities which weren't already covered by the antivirus and vulnerability scan tools used by the company.

To continue the development of the project, the ZeroDays v2 project was developed, which aims to increase the types of threats considered by adding exploits and malware to the scope of the project. For this, an analysis of the essential fields for each type of threat, in order to determine which information sources would be relevant for the system's implementation. Changes were made to the current ZeroDays prototype to allow it to collect data about exploits and malware from previously chosen sources and enrich that information with the knowledge of which internal systems are vulnerable to a given threat, as well as determining if the threat was known to the company's internal threat detection tools. The notification system was also adapted to be able to send alerts to the CyberSOC team of the Cybersecurity Department, so that they can take informed decisions in a timely manner. This project also had as an objective the reengineering of the past implementation, in order to make it work once again and to increase its fault tolerance.

With the new system, we were able to obtain information about zeroday threats earlier, and also were detected threats that a month past weren't yet correctly captured by the threat detection tools used internally. With this information, notifications were created with the relevant information to better understand the threat, and so that it could be analysed.

Key-Words: Zero-Day, Exploits, Malware, Proactive Security

Resumo

Em 2020 foi decretado o estado de emergência para permitir a concretização de medidas para controlar a propagação do vírus COVID-19. Uma dessas medidas obrigou à adoção do regime de teletrabalho. O alargamento do número de funcionários a ligarem-se remotamente causou um crescimento nos ataques à cibersegurança observados nas empresas. Para mitigar estes ataques, no ano de 2021, iniciou-se o desenvolvimento do sistema ZeroDays v1, com o fim de obter informações sobre vulnerabilidades dia-zero, ou seja, que não estão cobertas pelos sistemas de antivírus e outras ferramentas de pesquisa de vulnerabilidades utilizadas na empresa.

De forma a dar continuidade ao trabalho desenvolvido, foi desenvolvido o projeto ZeroDays v2, que visa alargar as ameaças consideradas, nomeadamente aos exploits e ao software malicioso. Para o projeto foram analisados os campos fundamentais disponíveis nas fontes de informação de cada um dos tipos de ameaças, com o fim de determinar quais fontes seriam relevantes para a implementação do sistema. Realizaram-se alterações à versão existente do protótipo ZeroDays de maneira a que consiga agregar informação sobre *exploits* e software malicioso dia-zero de fontes pré-selecionadas e enriquecer estes dados com conhecimento sobre que sistemas informáticos internos estão vulneráveis, bem como descobrir se as ferramentas existentes saberiam das ameaças. Adaptou-se o sistema de notificações existente no projeto ZeroDays para que este possa enviar alertas à equipa do CyberSOC da Direção de Cibersegurança da empresa, para que esta possa tomar conhecimento das ameaças de modo a que consiga realizar decisões informadas de uma forma atempada. Também foi objetivo deste projeto a reengenharia da implementação anterior, de modo a que este voltasse a funcionar e aumentasse a sua robustez.

Com o novo sistema verificou-se uma antecipação na obtenção de informações sobre ameaças dia-zero, bem como foram detetadas ameaças que passado um mês ainda não tinham sido corretamente descritas pelas ferramentas de proteção internas. Com esta informação foram geradas notificações com a informação necessária para entender o problema, para que este pudesse ser tido em conta.

Palavras-chave: Dia-Zero, Exploits, Software Malicioso, Segurança Próativa

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

CVE	**C**ommon **V**ulnerabilities and **E**xposures
CVSS	**C**ommon **V**ulnerability **S**coring **S**ystem
KB	**K**nowledge **B**ase
IOC	**I**ndicadores de **C**ompromisso
EDB	**E**xploit **D**ataBase
ZDT	**0D**ay.**T**oday
PSS	**P**acket **S**torm **S**ecurity
VUL	**Vul**ners
MS	**M**al**S**hare
BAch	Malware **Ba**zaar.**ch**
INQ	**InQ**uest Labs
VB	**V**irus**B**ay
SM	**S**oftware **M**alicioso
FCr	**F**onte de **Cr**uzamento
FCo	**F**onte de **Co**nfrontação

Capítulo 1

Introdução

O sistema ZeroDays v2 visa a automação e estandardização de um processo *ad hoc* que era feito manualmente, por membros da equipa do CyberSOC na direção de cibersegurança - a recolha de informação confiável sobre novas ameaças de *exploits* e software malicioso que são desconhecidos internamente à organização. Deste modo, ao criar um processo formal que efetua esta recolha e filtragem de dados, é possível reduzir a latência que existe entre a produção dos dados e a obtenção do conhecimento de ameaças reais que podem afetar os sistemas computacionais da empresa.

Com este fim em vista, foi desenhado e desenvolvido o projeto ZeroDays v2, para poder informar a direção de cibersegurança da empresa de novas ameaças de forma a que esta tenha mais tempo para agir sobre elas. O valor deste sistema é proporcional à qualidade das fontes de informação - quanto mais recente e relevante for a informação, mais útil será para a empresa. Esta melhoria no tempo de recolha de informação pode ser crucial e definir a diferença entre uma ameaça concretizada com sucesso ou uma ameaça frustrada.

Este projeto difere dos restantes sistemas propostos e implementados anteriormente, pois junta um diferente tipo de ameaças com a validação interna, tendo em foco princípios como o da simplicidade, modularidade, certeza da informação fornecida, e velocidade de ação.

1.1 Motivação

Com o aumento da capacidade computacional dos sistemas, tem-se observado um crescimento na dimensão e complexidade do software. Um exemplo claro disto é o Kernel Linux, cujo código-fonte em 1994 (versão 1.0) ocupava na ordem dos 1.5 megabytes, mas hoje ocupa 245 megabytes (versão 5.19, lançada a 31 de julho). Tal representa um crescimento de 163 vezes do seu código-fonte, aumentando assim a complexidade do software.

Com este aumento da complexidade, é natural que sejam introduzidos mais erros no código-fonte, sendo que uma estimativa aceite é de que existem entre 5 a 50 erros por cada 1000 linhas de código [25]. Se pegarmos no exemplo anterior do Kernel Linux, a mudança da versão 5.18 para a versão 5.19 inclui a adição de 495 mil linhas novas [3], podendo-se estimar que foram acrescentados entre 2475 e 24750 novos erros. Dentro desses novos erros, vários poderão corresponder a novas vulnerabilidades. Deste modo, são abertos novos vetores de ataque que podem ser explorados com programas apelidados de *exploits*, possibilitando a introdução de malware nos sistemas.

A 12 de maio de 2017, muitas organizações foram apanhadas desprevenidas por um ataque de ransomware, apelidado de WannaCry. Este ataque propagou-se a partir de um *exploit*, o EternalBlue [14], que explora uma vulnerabilidade na implementação

do protocolo SMB [1] [35] nos sistemas operativos da Microsoft. Esta vulnerabilidade era conhecida pela National Security Agency (NSA) desde 2012 [41], e existia uma correção para a vulnerabilidade desde 14 de março de 2017 [24], e foi catalogada na NVD no dia 16 de março de 2017 [30], dois meses antes de o ataque WannaCry se ter consumado. Um mês antes de o ataque se realizar, a 14 de abril de 2017, o *exploit* EternalBlue foi disponibilizado publicamente pelo grupo de hackers apelidado de Shadow Brokers [18]. Durante um mês, o *exploit* foi público até que foi utilizado no ataque WannaCry [20].

Como muitos sistemas não tinham feito a atualização de segurança que corrigia esta vulnerabilidade, o ataque teve grande sucesso. Estima-se que num espaço de quatro dias cerca de 200 mil computadores em 150 países foram afetados [40], causando uma perda estimada de 4 mil milhões de dólares americanos [38]. No entanto, este ataque não deixou de existir após estes quatro dias de maior atividade. Todos os sistemas que não têm a atualização de segurança que remenda esta vulnerabilidade ainda estão expostos a esta ameaça, seja pelo WannaCry ou por outro ataque que use a mesma vulnerabilidade [12].

Nos anos recentes constatou-se que com a crescente introdução de novos dispositivos, e da sua ligação às redes, aumenta a superfície de ataque das organizações [37], o que torna natural a necessidade de uma atitude mais pró-ativa em relação à cibersegurança. Esta necessidade agravou-se com a pandemia de COVID-19, uma vez que obrigou a um confinamento obrigatório para alguns setores. Nessa altura, em que milhares de funcionários tiveram de começar a trabalhar remotamente, ocorreu um crescimento significativo nos dispositivos conectados remotamente às redes internas das organizações, elevando o nível de risco face a ataques externos.

Assim, a cibersegurança deve ser tratada adequadamente numa empresa com variados sistemas informáticos. Quando surgem ameaças, é possível ajudar a empresa para que esta consiga reduzir os danos reputacionais, evitar perdas monetárias devido à indisponibilidade inesperada dos sistemas. Permite também proteger dados confidenciais, a partir do seu foco nos atributos da confidencialidade, integridade, e disponibilidade dos sistemas. Ao ter uma atitude pró-ativa em relação à cibersegurança, uma organização consegue ser mais ágil em relação a novas ameaças, quer seja na minimização de custos de resolução de incidentes ou na prevenção, para que estas ameaças não se cheguem a consumar. Com este intuito, a empresa sentiu a necessidade de criar uma solução para organizar informação sobre vulnerabilidades a que os seus ativos estão expostos, mas que não estão cobertas pelas ferramentas de teste em uso (consideradas dia-zero).

Deste modo, foi desenvolvido o ZeroDays v1, uma ferramenta que permite a recolha de uma forma automatizada de dados provenientes de fontes de informação de vulnerabilidades consideradas dia-zero [42]. Após a conclusão do projeto, a Altice Portugal seguiu o passo lógico de estender a funcionalidade desta ferramenta, de modo a que esta consiga acrescentar informações sobre *exploits* e software malicioso. Com esta informação adicional, a organização consegue manter-se o mais atualizada, de forma a conseguir atuar em tempo útil face às novas ameaças que se apresentam continuamente.

1.2 Objetivo

O projeto ZeroDays v2 corresponde a uma evolução do ZeroDays original, com o intuito de adicionar funcionalidade a recolha e correlação de dados sobre *exploits*

[1] SMB é um protocolo de rede que permite a partilha de ficheiros, de impressoras, e outros recursos.

e software malicioso novos, de forma a informar a organização dos ataques mais recentes. Adicionalmente, existe o desejo de reorganizar a atual implementação do ZeroDays, para que esta seja mais robusta.

Como tal, o objetivo deste projeto é o desenvolvimento de um sistema de gestão de ameaças dia-zero, nomeadamente de *exploits* e de software malicioso. Irá ser criado um sistema que fará a recolha, interpretação, e notificação da existência de ameaças dia-zero para a direção de cibersegurança da organização, para que esta possa tomar decisões atempadas sobre esta informação de maneira a reduzir o risco associado à materialização destas ameaças.

Este projeto não contempla a gestão de vulnerabilidades dia-zero, dado que esse sistema já foi criado no ZeroDays original [42]. Este projeto tem como objetivo a sua expansão e reengenharia, mas não tem como objetivo a sua recriação. Também se encontra fora do âmbito deste projeto a automatização da ação sobre a informação recolhida, mas sim o fornecimento do conhecimento necessário para poder mitigar uma ameaça.

1.3 Contribuições

O projeto ZeroDays v2 foi desenvolvido com o intuito de facilitar uma deteção atempada de ameaças de cibersegurança (mais especificamente, *exploits* e software malicioso dia-zero). As principais contribuições deste projeto são as seguintes:

- A **reengenharia** da implementação da versão anterior do ZeroDays;
- A **recolha automática** de *exploits* e de software malicioso de fontes de informação selecionadas;
- A **identificação** de informação relevante dentro dos dados colecionados;
- A produção de informação **concreta**, **confiável**, e **conclusiva** sobre estas ameaças;
- A **notificação em tempo útil** da direção de cibersegurança de novos *exploits* e software malicioso dia-zero.

Em suma, com este projeto pretende-se fornecer informação dia-zero relevante à direção de cibersegurança, de forma a que esta possa agir sobre esta em tempo útil, reduzindo assim a latência existente entre a sua publicação na web e a sua descoberta pelos elementos da equipa de CyberSOC da direção de cibersegurança da organização.

1.4 Estrutura do documento

O presente documento segue a seguinte estrutura:

- No capítulo 2, será dado algum contexto sobre o problema que irá ser enfrentado, serão esclarecidos alguns conceitos necessários, e serão introduzidos trabalhos relacionados.
- No capítulo 3, serão enunciadas características que as fontes de informação devem possuir, as fontes de informação que foram escolhidas para a implementação do projeto, e as fontes de cruzamento e confrontação a ser utilizadas.
- No capítulo 4, será explicada a arquitetura do sistema ZeroDays v2, bem como serão descritos alguns aspetos fundamentais da sua implementação.

- No capítulo 5, serão avaliados e analisados os resultados da execução das duas componentes do sistema, a componente de *exploits* e a componente de software malicioso.
- No capítulo 6, irá ser feito um sumário do trabalho realizado, serão tecidas considerações sobre os resultados obtidos, e serão propostas algumas possíveis evoluções futuras com o fim de melhorar o sistema.

Capítulo 2

Contexto

2.1 Noções gerais

Nesta secção irão ser fornecidas algumas noções relativas ao problema das vulnerabilidades, *exploits* e software malicioso. Assim, serão definidos esses mesmos conceitos nas Subsecções 2.1.1, 2.1.2 e 2.1.4, respetivamente. Também será apresentado o ciclo de vida de uma vulnerabilidade (Subsecção 2.1.3), que inclui a criação de um *exploit*, e o ciclo de vida de um software malicioso (Subsecção 2.1.5). Serão de seguida esclarecidos os conceitos de superfície de ataque e de remendos e mitigações, nas Subsecções 2.1.6 e 2.1.7. Por último, serão apresentados alguns trabalhos relacionados com o ZeroDays v2, mas que apresentam diferenças chave no seu funcionamento, impedindo que possam ser consideradas como análogos a este sistema.

2.1.1 Vulnerabilidade

Uma vulnerabilidade é definida pelo National Institute of Standards and Technology (NIST) como um ponto fraco num sistema de informação, quer seja por falhas no desenho, implementação, ou por utilização incorreta do mesmo [36], que causam que o sistema esteja vulnerável a ataques que violem a sua segurança [31].

Uma vulnerabilidade por desenho do sistema ocorre durante a fase de desenho do sistema, ainda antes de ter iniciado a implementação. Um exemplo deste tipo de vulnerabilidade seria a escolha de um algoritmo errado para um sistema, como utilizar o algoritmo MD5 [32] para sintetizar palavras-passe [1].

Quando uma vulnerabilidade é introduzida durante a programação do sistema, esta é denominada de vulnerabilidade de implementação. Um exemplo deste tipo de vulnerabilidade seria a implementação errada de um algoritmo criptográfico, tornando usando apenas os primeiros 8 bits da chave.

Por último, uma vulnerabilidade pode ainda ser criada por utilização incorreta de um sistema. Um exemplo deste caso seria a utilização de palavras-passe fracas ou configurando as comunicações sem suporte de canais seguros.

O projeto ZeroDays [42] original, cujo desenvolvimento iniciou em 2021, tratava da recolha deste tipo de informações - as vulnerabilidades dia-zero.

2.1.2 *Exploit*

Um *exploit* é um software ou pedaço de código que permite explorar uma vulnerabilidade. Estes normalmente são categorizados de acordo com os resultados que se obtém depois de se explorar a vulnerabilidade - seja por execução de código arbitrário, escalonamento de privilégios, entre outros [23].

[1] Em 2004 foi provado que este algoritmo não é resistente a colisões, e em 2014 estimou-se que provocar uma colisão custava 0,65 dólares americanos, mais impostos [6]

Um *exploit* pode ser considerado como uma Prova de Conceito (PoC) se for desenvolvido apenas para demonstrar o impacto que um ataque que explora essa vulnerabilidade teria num sistema. Apesar de não apresentar um perigo imediato, o PoC requer alguma cautela pois traz maior destaque a uma vulnerabilidade através da demonstração que é viável a sua exploração que muitas vezes tal acontece antes de esta ser corrigida, aumentando assim a atenção dada por entidades maliciosas, que poderão desenvolver *exploits* para certos sistemas.

2.1.3 Ciclo de vida de uma vulnerabilidade

Uma vulnerabilidade passa por vários estados até à sua resolução, como ilustrado na Figura 2.1.

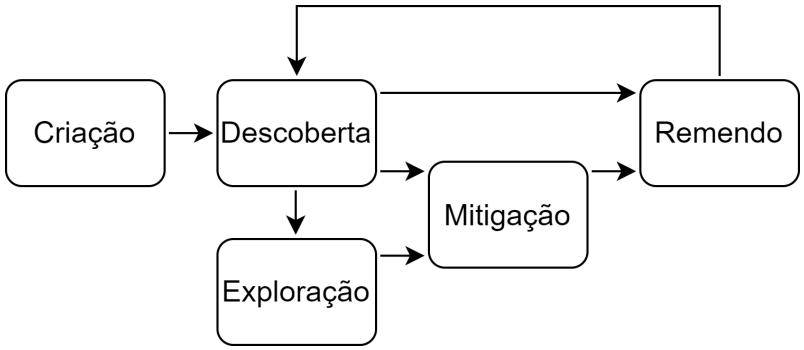

Figura 2.1: Ciclo de Vida de uma Vulnerabilidade.

O estado inicial é a sua *criação*. Isto normalmente acontece por um erro de implementação ou de desenho da aplicação / sistema.

Quando esta vulnerabilidade é detetada, seja pelos programadores da aplicação / sistema ou por elementos externos, esta transita para o estado de *descoberta*.

Se a vulnerabilidade for detetada por um agente externo, este deverá reportar a mesma ao fabricante, para que se possa corrigir o problema com um remendo. Em alternativa, este pode contactar uma organização que trate do processo de relatar ao fabricante a vulnerabilidade de uma maneira responsável, como por exemplo a Zero Day Initiative [15].

No entanto, se a vulnerabilidade for descoberta por um agente com intenções maliciosas, este provavelmente não irá divulgar a informação. Dependendo da aplicação, poderão existir incentivos financeiros para vender a descrição da vulnerabilidade a entidades terceiras, ou desenvolver o seu próprio *exploit*.

Após o fabricante da aplicação ou sistema tomar conhecimento do problema, este deve iniciar a criação de um remendo para corrigir a vulnerabilidade. Como um passo intermédio pode, até à sua aplicação, desenvolver mitigações de modo a controlar o seu risco. Após a aplicação do remendo, a vulnerabilidade encontra-se inativa, podendo no futuro ser reativada, quer seja por um remendo deficiente ou por esta vulnerabilidade ser um sintoma de um problema maior.

2.1.4 Software malicioso

Um software malicioso é um programa não autorizado que irá afetar negativamente um sistema informático [10]. Estes podem realizar uma intrusão nos sistemas informáticos por via de *exploits*, abusando de vulnerabilidades que os donos das aplicações ou sistemas muitas vezes não sabem que existem.

Existem diversos tipos de software malicioso, denominados de estirpes. Alguns exemplos destes são uma aplicação que serve anúncios à vítima (*Adware*), software que envia as atividades do utilizador para o criador do programa (*Spyware*), e um programa que age normalmente, para depois realizar atividades maliciosas (*Trojan*, ou em português, *Cavalo de Tróia*) [26].

Dentro destes tipos de software malicioso existem ainda subtipos, apelidados de famílias, em que os programas são variações de um original, seguindo um dado padrão de ações. Um exemplo é o software malicioso *RedLine Stealer*, que rouba informações de browsers, como as credenciais guardadas, dados de introdução automática, e informações bancárias [33].

2.1.5 Ciclo de vida de um software malicioso

Tal como uma vulnerabilidade, o software malicioso também tem um ciclo de vida de acordo com a Figura 2.2.

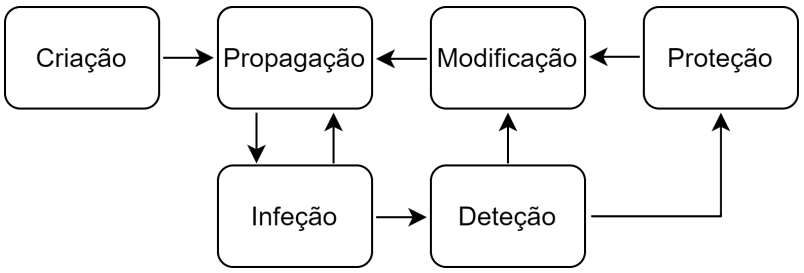

Figura 2.2: Ciclo de Vida de um Software Malicioso.

O software malicioso começa pela sua *Criação*. Nesta etapa, o criador do software malicioso escolhe o objetivo do seu programa, e o método de entrada no sistema. Este associa o software a um *exploit*, para que este se possa instalar e assim conseguir obter maior acesso ao sistema.

De seguida, na fase de *Propagação*, o software malicioso é propagado para a vítima, de modo a que esta seja infetada. Dependendo do tipo de software malicioso, e de como opera, esta fase pode ser simultânea com a seguinte, ou separada, se o programa se mantiver dormente numa fase inicial.

Na fase *Infeção*, o software malicioso realiza as ações que são o seu objetivo no sistema infetado, como por exemplo, o envio de informação sobre atividades da vítima. Depois desta fase, o software malicioso pode ainda voltar à fase de *Propagação*, se este for o seu objetivo, o que é o caso das *worms*, um tipo de software malicioso que se propaga sem interação humana.

A fase de *Deteção* ocorre quando a comunidade da cibersegurança começa a dar conta do sucedido, e inicia a criação de proteções para este software malicioso, seja ao identificar os indicadores de compromisso ou ao adicionar a sua assinatura a uma lista negra.

Na fase de *Proteção*, os sistemas começam a ser protegidos contra o software malicioso, levando a que a sua eficácia comece a reduzir.

Para evitar a proteção dos sistemas contra o software malicioso, o autor do software malicioso pode proceder à *Modificação* do mesmo para que este não seja detetado, iniciando novamente a fase de *Propagação*.

2.1.6 Superfície de ataque

A superfície de ataque é definida como o conjunto de pontos que um sistema possui que podem ser atacados por um agente malicioso. Isto significa que, quanto menor for o âmbito do sistema, e quanto menos funcionalidades desnecessárias tiver, mais limitada será a sua superfície de ataque, e por consequência, dificultará a materialização de uma ameaça num ataque com sucesso.

Um exemplo associado a um servidor que serve como sistema de mensagens seria ter mais portos abertos que os que são precisos, para acesso a serviços como o *Secure Shell* (SSH) ou o *File Transfer Protocol* (FTP). Apesar de poder eventualmente ajudar a manutenção destes sistemas, estes serviços não necessitam de estar expostos à internet, mas ao estarem disponíveis, aumentam a superfície de ataque do sistema, possibilitando a um agente malicioso aproveitar-se de vulnerabilidades que eventualmente existam nestes serviços para explorar o sistema.

2.1.7 Remendos e mitigações

Quando um fabricante toma conhecimento de uma vulnerabilidade e a quer reparar, para que consiga assegurar uma maior segurança aos utilizadores do seu produto, este normalmente disponibiliza um remendo (do inglês *patch*) para corrigir o erro no software. Este remendo é um pequeno pedaço de código que não apresenta mais nenhuma funcionalidade sem ser essa correção, e pode ser incluído numa *release* do software, ou o fabricante pode ainda realizar uma atualização só com esse remendo, caso se justifique.

Enquanto o remendo não é desenvolvido e aplicado, a organização pode ainda preparar e aplicar mitigações para que o impacto da vulnerabilidade seja reduzido.

No entanto, nem sempre um remendo é eficaz, e nem sempre as vulnerabilidades são completamente corrigidas, podendo voltar a surgir a mesma vulnerabilidade no futuro.

2.2 Conceitos particulares ao âmbito do projeto

Na presente secção serão dispostos alguns conceitos mais particulares ao âmbito do projeto, o problema das ameaças dia-zero. Nas Subsecções 2.2.1 e 2.2.2 irão ser explicados os termos *ameaça* e *dia-zero* aplicados ao âmbito do projeto, e nas Subsecções 2.2.3 e 2.2.4 serão dadas as definições de fontes de cruzamento e fontes de confrontação interna, que serão fundamentais para a definição dos termos anteriores.

2.2.1 Ameaça

Para o presente trabalho, entende-se como uma ameaça tudo o que pode afetar a confidencialidade, integridade ou disponibilidade de um sistema. Assim, como o âmbito deste projeto são os *exploits* e software malicioso, este termo será utilizado quando for preciso mencionar de uma forma genérica ambos os tipos de ameaças.

2.2.2 Dia-zero

Uma ameaça dia-zero consiste numa ameaça que não é conhecida internamente à organização. Neste caso, para a empresa, são denominadas como ameaças dia-zero os *exploits* e os softwares maliciosos que não são foram ainda identificadas pelas ferramentas utilizadas pela empresa.

Isto deve-se ao facto que uma ameaça pode ser desconhecida pela generalidade dos utilizadores, mas conhecida por um número reduzido de entidades que não têm como objetivo a divulgação e a proteção dos sistemas. Este tipo de ameaças apresenta um risco severo, pois podem acontecer inesperadamente sem que a organização tenha meios adequados para se defender.

2.2.3 Fontes de Cruzamento

Para a realização deste projeto, serão utilizadas fontes de informação denominadas como Fontes de Cruzamento (FCr). Estas fontes de informação são utilizadas internamente, e serão empregues neste projeto para definir se a informação recolhida já é do conhecimento da organização. A existência destas fontes de informação é essencial para determinar quais ameaças são de dia-zero, e quais ameaças são previamente compreendidas.

2.2.4 Fontes de Confrontação Interna

Neste projeto também serão utilizadas fontes de informação apelidadas de Fontes de Confrontação Internas (FCi). Estas fontes contêm informações sobre os sistemas utilizados na empresa, permitindo assim definir se a organização é ou não afetada por determinada ameaça.

Esta confrontação será melhor ou pior consoante o detalhe dos dados disponíveis, ou seja, dos cadastros internos da empresa sobre os seus sistemas. Se não houver informação suficiente ou incorreta, podem ser gerados alertas desnecessários.

2.3 Partilha de descobertas

Por último, será clarificado o processo de descoberta e de partilha de ameaças. O conhecimento da operação destes processos é importante para entender a escassez de informação sobre ameaças que existe, e de como a informação é partilhada. Primeiramente, na Subsecção 2.3.1 serão dados alguns exemplos de possíveis acontecimentos e desfechos da descoberta de ameaças, e na Subsecção 2.3.2 serão descritos mercados onde são partilhados estes dados.

2.3.1 Descoberta de ameaças

Se uma ameaça for descoberta por um agente com intenções alinhadas com a cibersegurança, como o fabricante ou dono do sistema, a vulnerabilidade tem uma maior probabilidade de ser remendada em tempo útil, de maneira a evitar que essa ameaça se cumpra.

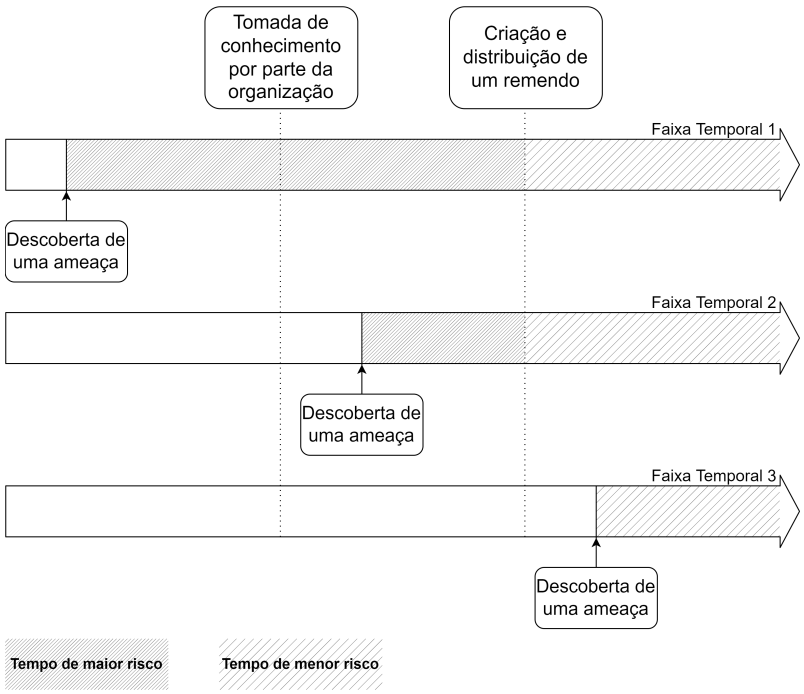

FIGURA 2.3: Efeito do tempo e do conhecimento de uma organização sobre o risco de uma ameaça.

Como representado na Figura 2.3, uma ameaça pode seguir várias rotas, quando é descoberta por um utilizador:

- É divulgada diretamente ao fabricante, para que este consiga criar um remendo para que esta ameaça consiga ser minimizada ou desaparecer;
- Não é divulgada a ninguém, nem é utilizada;
- É utilizada para criar um *exploit* que explora essa vulnerabilidade para obter acesso ao sistema;
- É divulgada por canais gratuitos sem contactar o fabricante;
- É vendida a terceiros, para que estes possam usufruir dessa informação confidencial.

Todos estes caminhos geram níveis de risco diferentes, sendo que quanto mais tarde o fabricante tomar conhecimento de uma ameaça, mais impacto na segurança poderá ter esta vulnerabilidade, pois pode ser criado um *exploit* para a explorar.

A qualquer momento após o conhecimento do fabricante, a vulnerabilidade pode ser remendada, reduzindo o risco substancialmente, mas não o nulifica, pois podem continuar a existir sistemas que não são remendados. O remendo também pode ser imperfeito, não resolvendo a vulnerabilidade de uma forma completa.

Quando uma entidade com intenções maliciosas descobre uma vulnerabilidade ocorre o pior caso, pois a ameaça será consumada antes de a organização tomar conhecimento da sua existência, não conseguindo assim mitigá-la ou eliminá-la.

2.3.2 Mercados de ameaças

Para efeitos deste relatório, aceitamos o termo *dark web* para definir todos os recursos que não são possíveis de aceder sem recurso ao TOR, e *deep web* como todos os recursos que não são indexados por motores de busca [29].

Como informações sobre *exploits* e software malicioso são importantes para entidades com intenções maliciosas, observa-se que existe um mercado à volta do desenvolvimento e partilha de novos *exploits* e software malicioso [39]. Nestes mercados operam agentes que descobrem ameaças e que as vendem à melhor oferta. Um exemplo é uma vulnerabilidade dia-zero que afetava os produtos baseados no sistema operativo iOS, que foi comercializada por 8 milhões de euros [28].

É possível encontrar vários mercados de compra e venda deste tipo de informação na *dark web* [19], pois são atividades ilícitas que pedem um certo nível de anonimato, que a rede TOR permite.

Constata-se que devido à dificuldade associada ao desenvolvimento de novos *exploits* e software malicioso, e o elevado preço de venda que às vezes atingem em mercados paralelos, frequentemente existe uma motivação forte para que novas descobertas sejam mantidas escondidas dos fabricantes.

2.4 Outros sistemas de gestão de ameaças

Devido ao baixo incentivo oferecido à divulgação de *exploits* e software malicioso, existe pouca disponibilidade de informação sobre este tipo de ameaças. Assim, torna-se complicada a realização de projetos semelhantes aos tratados nesta tese.

É possível encontrar empresas que disponibilizam produtos que procuram conferir uma maior segurança com base em informação sobre ameaças dia-zero, recorrendo a ferramentas desenvolvidas internamente. Muitas destas ferramentas não disponibilizam a informação em si,para evitar correr o risco de poder estar a alimentar as empresas concorrentes com dados de conhecimento restrito e de difícil recolha.

Devido a estes fatores, não existem produtos similares no mercado que realizem as mesmas funções que o ZeroDays v2. Iremos descrever em seguida alguns sistemas com objetivos relacionados.

2.4.1 Taranis

O Taranis [7] é um projeto desenvolvido pelo Centro Nacional de Cibersegurança dos Países Baixos. Esta ferramenta facilita o processo de monitorizar e analisar artigos e notícias, para recolher informação relevante para segurança informática. As fontes usadas são programáveis, sendo possível definir o sítio onde o sistema vai buscar as suas informações. Este projeto também efetua esta recolha de uma maneira modular,

tal como o ZeroDays v1. Depois de obtida a informação, esta é agrupada e enviada para listas de emails internas.

Este projeto difere do ZeroDays v2 pois não trata do cruzamento com dados já presentes nos sistemas da organização, nem trata do processo de notificação às partes interessadas.

2.4.2 D-Miner

O D-Miner [19] é uma *framework* para a recolha de informação em mercados de *exploits* dia-zero na *dark web*. Os autores sugerem que ao se obter esta informação de uma forma automatizada poderia ajudar a trazer mais conhecimento a partes interessadas. Nesse âmbito, desenvolvem uma metodologia de modo a encontrar uma solução para este problema, e propõem uma possível implementação.

Um conjunto de experiências demonstram que o sistema apresentado consegue recolher informação de uma forma automatizada de várias fontes, nomeadamente com dois mercados de ameaças na *dark web*.

Apesar da versatilidade do D-Miner, este não contempla o cruzamento com informação interna à empresa, pelo que não é possível qualificar a informação recolhida de acordo com a sua relevância interna.

2.4.3 Follow The Blue Bird

No *Follow The Blue Bird* [4] é realizada uma análise sobre a viabilidade da utilização da rede social Twitter como uma fonte de informação. Os autores concluem que esta rede social é uma fonte de informação viável, onde muitas vulnerabilidades reportadas antecipadamente têm um impacto elevado, podendo ainda ter informações sobre como mitigar esses problemas.

A inclusão de fontes de informação deste género (não curadas) não está a ser considerada pela empresa, pois não é possível garantir a validade das ameaças relatadas, nem dos dados relatados sobre estas. Um outro detalhe importante é também o facto dos autores não terem avaliado a sua qualidade para fornecer informações sobre outros tipos de ameaças, como *exploits* e software malicioso.

2.4.4 SYNAPSE

A SYNAPSE [8] monitoriza publicações em redes sociais, com recurso a aprendizagem automática. Na implementação apresentada, os autores (os mesmos do *Follow The Blue Bird*) escolheram recolher informações do Twitter, tendo tido sucesso considerável no processo de recolha de informação útil, com uma taxa de verdadeiros positivos, ou seja, de verdadeiras ameaças, acima dos 90%.

Esta ferramenta difere do ZeroDays v2 no sentido em que neste projeto é requisito que as fontes de informação sejam verificadas, ou que tenham um elevado nível de confiança. Um outro aspeto negativo de implementar um motor de recolha de informações deste género seria o aumento significativo do âmbito deste projeto, pelo que esta solução não está a ser considerada.

Capítulo 3

Fontes de informação e os seus dados

O sistema ZeroDays v2, realizando um processo de recolha, cruzamento e disseminação de dados, é apenas tão bom quanto a qualidade das suas fontes de informação. Deste modo, neste capítulo serão apresentados os dados que as fontes de informação devem conter sobre as ameaças, nomeadamente os exploits e software malicioso, bem como as fontes de cruzamento e de confrontação que serão utilizadas.

3.1 Dados relevantes para recolha

Para que possam ser cruzados os dados recolhidos sobre exploits e software malicioso, é conveniente que estas contenham campos que possibilitam a sua identificação (idealmente de uma maneira única). Estes campos irão ser úteis para o cruzamento da informação entre fontes internas e externas à organização. Assim, nesta secção serão enunciados alguns dados que permitem a análise e correlação de informação de exploits e software malicioso.

3.1.1 *Common Vulnerabilities and Exposures*

O projeto *Common Vulnerabilities and Exposures* (CVE) [27] tem como objetivo catalogar vulnerabilidades divulgadas publicamente, de modo a que estas sejam facilmente identificáveis.

O processo de catalogação de vulnerabilidades inicia com a descoberta da vulnerabilidade. De seguida, o agente que a descobre reporta a mesma ao fabricante, que pede um novo identificador (ID) à organização que gere o CVE, que identifica univocamente a vulnerabilidade. O CVE atribui o novo ID ao fabricante para que este possa preencher os detalhes da vulnerabilidade, e depois, a vulnerabilidade é catalogada.

Esta informação é útil para o projeto para que se possa relacionar um *exploit* com a vulnerabilidade que este explora. Deste modo, é possível seguir o estado da vulnerabilidade, e saber quando esta se encontra remendada. A descrição é útil para se compreender o comportamento de um software malicioso, e se saber que exploits este utiliza (para se propagar, para movimento lateral, para escalar privilégios, entre outros).

3.1.2 *Common Vulnerability Scoring System*

O Common Vulnerability Scoring System (CVSS) é um standard para categorizar a gravidade de vulnerabilidades [13]. Neste momento existem duas versões em grande

utilização - o CVSSv2 e o CVSSv3.1. Como a versão CVSSv3.1 é mais completa, esta será utilizada preferencialmente no projeto.

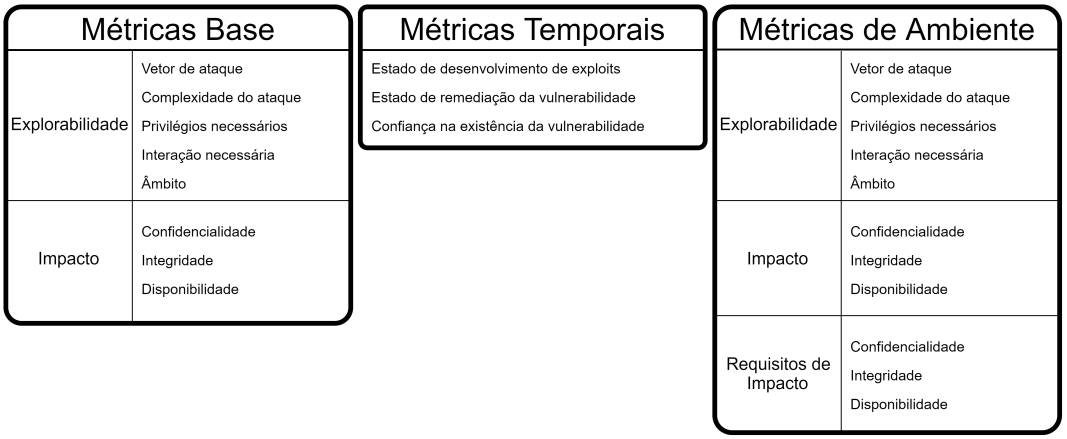

FIGURA 3.1: Representação das categorias e parâmetros do CVSSv3.1.

A metodologia definida pelo CVSS associa um conjunto de métricas às vulnerabilidades. O CVSS é composto por três tipos de métricas: as métricas de base, as métricas temporais e as métricas de ambiente (ver figura 3.1).

As métricas base são atribuídas a partir das características da vulnerabilidade em causa, e são imutáveis. Estas são compostas por dois tipos de métricas, as métricas de explorabilidade e as métricas de impacto.

Dentro das métricas de explorabilidade estão contidas informações como:

- O vetor de ataque, ou seja, o ponto de entrada onde um ataque pode ser efetuado contra o sistema. Estes são, por ordem de escala, a rede, a rede adjacente, localmente, ou por acesso físico;
- A complexidade do ataque necessário para explorar a vulnerabilidade. Os valores possíveis para esta componente são *baixo* ou *alto*;
- Os privilégios necessários que um atacante deve ter para poder explorar esta vulnerabilidade. Esta componente é caracterizada por um de três valores, *nenhum*, *baixo*, ou *alto*;
- A interação necessária por parte da vítima, ou seja, se é preciso que a vítima complete uma ação, ou se é independente da vítima. Os valores possíveis desta componente são *nenhum* e *necessário*;
- O âmbito da vulnerabilidade se explorada, isto é, a capacidade de um atacante poder impactar recursos para além dos que supostamente a aplicação teria privilégios. Esta componente é caracterizada por *não alterado*, e *alterado*.

Existem também métricas sobre o impacto de vulnerabilidades, em que cada uma tem um valor entre *nenhum*, *baixo*, e *alto* que são:

- O impacto sobre a **confidencialidade** do sistema;
- O impacto sobre a **integridade** do sistema;
- O impacto sobre a **disponibilidade** do sistema;

Em conjunto estas métricas formam as métricas base de risco de uma vulnerabilidade. As métricas são traduzidas para um número de 0 a 10, que define o risco associado à vulnerabilidade. O risco é atualizado de acordo com as métricas temporais, e depois modificado consoante as suas métricas ambientais.

As métricas temporais são compostas por um único tipo que contém:

- O estado de desenvolvimento de um exploit que tome partido da vulnerabilidade. Este componente é caracterizado por um de cinco tipos: *não definido* quando não se conhece o seu estado, *existência não provada*, *existe um PoC*, *existe um exploit funcional*, e *alta*;
- O estado de remediação da vulnerabilidade, que é caracterizado por *não definido*, *remendo oficial*, *remendo temporário*, *solução de contorno*, e *não disponível*;
- A confiança na existência da vulnerabilidade e nos seus detalhes, que pode ser definida por um de quatro valores. Estes são: *não definido*, *desconhecido*, *razoável*, e *confirmado*.

Se um dos parâmetros é *não definido*, então deve assumir o valor mais alto. Esta solução permite minimizar uma possível subavaliação do risco, podendo assim oferecer um resultado mais alto, face o desconhecido.

As métricas de ambiente são compostas por três grupos: as métricas de explorabilidade, as métricas de impacto, e as métricas de requisitos de impacto. As métricas de explorabilidade e as métricas de impacto são as mesmas que as métricas base, no entanto, são modificadas para ter em consideração o seu papel em relação à empresa e ao ambiente em que estes sistemas se situam. As métricas de requisitos de impacto são definidas pelo requisito que a empresa tem para cada critério de impacto (confidencialidade, integridade e disponibilidade). Estes valores podem ser *não definido*, *baixo*, *médio*, e *alto*.

Desta maneira, ao implementar a metodologia definida pelo CVSS, e ao juntar a informação sobre estas métricas, é possível obter um valor de 0 a 10 associado ao risco de cada vulnerabilidade, de uma maneira estandardizada. Quanto maior for este valor, maior risco a vulnerabilidade apresenta.

3.1.3 Indicadores de compromisso

Um Indicador de Compromisso [22] (apelidado normalmente de IoC, do inglês *Indicator of Compromise*) são informações encontradas no sistema que podem indicar que houve uma intrusão. Como tal, os IoC são utilizados por profissionais de cibersegurança para detetar software malicioso de uma forma automática. Um exemplo de um IoC seria, por exemplo, uma ligação a um determinado endereço IP, ou a alteração de um determinado ficheiro.

3.1.4 Síntese de um ficheiro

A síntese de um ficheiro é um número de tamanho fixo,gerado por uma função criptograficamente segura sobre o ficheiro. Uma pequena alteração neste ficheiro resulta na criação de um número completamente diferente (com elevada probabilidade). Assim, este número pode ser utilizado para identificar (ou representar) ficheiros diferentes. Logo, apesar ser possível dois ficheiros diferentes terem o mesmo número, tal é altamente improvável, facilitando a tarefa de identificação de ficheiros maliciosos.

3.1.5 Plataforma afetada

Uma outra informação fundamental é a plataforma afetada pelo *exploit*. Este dado é essencial para a confrontação interna, de modo a definir quais sistemas informáticos são afetados por esta ameaça. Devido à inexistência de um cadastro da empresa que contenha os CPE [1], este campo é ainda mais importante, pois não pode ser aferido o programa vulnerável a partir do CVE-ID da vulnerabilidade explorada.

Uma plataforma pode ser um sistema operativo, como por exemplo, *Windows*, *macOS*, ou *Linux*, bem como uma linguagem de programação, como *Lua*, *Java*, ou *PHP*, ou ainda um programa específico, como *Excel*, *Firefox*, ou *Adobe Flash Player*.

3.1.6 MIME e Tipo de um ficheiro

O MIME é um standard que permite identificar o tipo de ficheiro com base na sua extensão. O tipo de um ficheiro pode ajudar na identificação de um software malicioso. Por exemplo, se o MIME de um ficheiro for *application/x-dosexec* ou *PE32*, é possível aferir que a plataforma afetada poderá ser o sistema operativo *Windows*. Esta conclusão poderá não estar sempre correta, pelo que estas informações não substituem a informação sobre a plataforma afetada.

3.1.7 Outros campos relevantes

Também são importantes outros campos que habitualmente aparecem junto da descrição do exploit, como o título da informação, o seu autor, a sua descrição, e a data da sua publicação. Todos estes campos ajudam a fornecer contexto sobre a ameaça ao utilizador final do sistema.

3.2 Dados importantes para exploits

Conforme a explicação na Secção 3.1, os dados importantes para o tratamento da informação de exploits são os seguintes:

- O CVE-ID da vulnerabilidade que este explora;
- A plataforma afetada por este exploit;
- O nível de acesso que a ameaça possibilita;
- Identificadores únicos da ameaça - seja um Hash, ou qualquer outro tipo de pedaço de texto que permita a sua identificação de uma forma a obter uma correspondência perfeita;
- A disponibilização do código do exploit;
- Meta-dados variados como o nome do autor do exploit, a data de criação, um título e uma descrição da sua operação.

Dispondo desta informação, é possível identificar um exploit e atribuir-lhe um nível de risco, bem como obter informações sobre o seu funcionamento para poder tomar medidas que reduzam o seu impacto.

[1] CPE, ou Common Platform Enumeration, é um esquema de nomes utilizado para identificar um determinado software.

Se uma fonte de informação possuir todas as informações acima detalhadas, então esta é ideal para o sistema ZeroDays v2, e eventualmente deverá ser integrada. Infelizmente, na realização deste projeto, não foi encontrada nenhuma fonte que contivesse todos estes dados, pelo que estes dados representam o que seria uma fonte perfeita, sem qualquer tipo de adaptação necessária ou compromisso.

3.3 Dados importantes para software malicioso

Tal como foi feito na secção anterior, os dados importantes para o tratamento da informação de software malicioso são os seguintes:

- O CVE-ID da vulnerabilidade explorada pelo software malicioso (seja para obter um ponto de entrada, para movimento lateral na rede, aumento de privilégios, entre outros);
- A plataforma afetada por este;
- A estirpe e família do software malicioso;
- Identificadores únicos para cada software malicioso - preferencialmente, algo que permita associar o software malicioso a um tipo comportamental, ou um padrão de atuação mais específico (por exemplo, se é uma variação do software malicioso *WannaCry*), ou um identificador único para o ficheiro;
- O tipo de ficheiro;
- A data da primeira vez que foi observado;
- Indicadores de compromisso associados.

Com estas informações é possível traçar um perfil do software malicioso, e responder a questões como a origem do software malicioso, como o identificar, como lidar com ele, e como agir caso se verifique uma infeção por este.

Tal como na Secção 3.2, se uma fonte possuir todos estes dados e os dispuser de uma forma atempada e verdadeira, então a fonte poderá ser integrada no sistema ZeroDays v2. Infelizmente, tal como na secção prévia, não foram encontradas fontes que apresentassem todas estas informações, pelo que terão de ser feitas adaptações e compromissos.

3.4 Fontes de informação

De forma a obter os apresentados anteriormente, nesta secção serão descritas as fontes de informação que foram consideradas para a realização do sistema ZeroDays v2. Na Subsecção 3.4.1 serão enumeradas as quatro fontes que foram investigadas para a recolha de informação de exploits e na Subsecção 3.4.2 será feito o mesmo, mas para as quatro fontes de informação para a recolha de dados de software malicioso. De seguida, serão descritas as fontes de cruzamento e confrontação utilizadas pelo sistema ZeroDays v2 para confirmar se uma ameaça é dia-zero e se afeta *assets* da empresa.

3.4.1 Fontes de exploits

Para a realização deste trabalho foram investigadas fontes que fornecem dados com as características chave descritas na Subsecção 3.2. Tendo em conta os critérios discutidos, e a restrição de apenas serem analisadas fontes não presentes na deep web nem na dark web, foram assinaladas quatro fontes como sendo de interesse: a 0day.today, a Exploit-DB, a PacketStorm Security, e a Vulners. Nesta subsecção será feita uma análise destas quatro fontes de informação, bem como uma discussão sobre a sua viabilidade para inclusão no sistema ZeroDays v2.

3.4.1.1 0day.today

O 0day.today [1] (ou ZDT) é uma das bases de dados de exploits mais populares, cuja informação é recolhida a partir de submissões voluntárias. Esta plataforma surgiu em 2008, e desde então tem fornecido informação útil sobre exploits, como a categoria em que este se insere, uma pequena descrição de como funciona, a sua plataforma, a data em que foi inserido na plataforma, e o CVE, caso este seja conhecido. Todos os conteúdos publicados são verificados pela equipa do 0day.today, tornando rara a situação de ocorrências em que são reportadas falsas ameaças.

A plataforma contempla ainda um mercado de exploits, onde alguns utilizadores podem vender o seu exploit em troca de *gold*, a moeda de troca utilizada neste site (1 USD equivale a 1 *gold*). Ao realizar transações numa soma total de 1000 *gold*, um utilizador pode obter acesso a todas as áreas da plataforma, nomeadamente a exploits mais recentes aos categorizados como mais perigosos. Mesmo sem aceder a esta porção do website, é possível obter gratuitamente bastante informação útil a partir desta plataforma, sendo assim considerada como uma fonte viável para a realização do projeto.

3.4.1.2 Exploit-DB

A Exploit-DB [11] (ou EDB) é uma base de dados mantida pela Offensive Security. Tal como o 0day.today, a informação é obtida a partir de submissões voluntárias da comunidade de cibersegurança.

Esta plataforma é completamente gratuita, e contém informações importantes para o motor de informação, como a plataforma afetada, a data em que foi inserido na plataforma, o tipo de exploit, o CVE da vulnerabilidade que este explora e o código do exploit.

Uma desvantagem em relação ao 0day.today reside na verificação de conteúdos - a quantidade de exploits verificados pela empresa que mantém o Exploit-DB é muito menor, comparativamente.

3.4.1.3 Packet Storm Securitsy

A Packet Storm Security [34] (ou PSS) é uma plataforma que engloba uma base de dados de ficheiros e um blog de notícias. Esta base de dados contém diversos conteúdos, como ferramentas, documentos técnicos, e exploits, sendo que este último é o mais relevante para este projeto.

Os conteúdos desta base de dados podem ser submetidos por qualquer pessoa, no entanto, estes são verificados antes de serem publicados. Os conteúdos contém informações consideradas úteis para a realização deste projeto, como o CVE da vulnerabilidade que o exploit explora, uma síntese (no sentido criptográfico) do exploit,

	Fontes			
	ZDT	EDB	PSS	VUL
ID único para cada entrada	✓	✓	✓	✓
Título	✓	✓	✓	✓
Autor	✓	✓	✓	✓
Descrição	✓		✓	✓
Data de Publicação	✓	✓	✓	✓
Categoria	✓	✓	✓	
Plataforma	✓	✓	✓	
Síntese			✓	
CVE (quando disponível)	✓	✓	✓	✓
Métricas de Envolvimento	✓			
Métrica de Risco Própria	✓			✓
Disponibiliza Código	✓	✓	✓	✓
Verifica Informação	✓	✓	✓	
Exclusivamente Informação Verificada			✓	
Websites	✓	✓	✓	✓
API				✓
Conteúdo Pago	✓			
Exclusivamente Conteúdo Pago				

Tabela 3.1: Fontes de Informação de Exploits e o seu Conteúdo

o código do exploit, o seu nome, e uma pequena descrição acerca do exploit e de como este opera.

3.4.1.4 Vulners

A Vulners [44] (ou VUL) é uma empresa que fornece uma base de dados de vulnerabilidades, exploits, e recompensas por encontrar bugs (ou seja, uma empresa pode pagar para que utilizadores encontrem bugs no seu sistema). Na secção de exploits, a Vulners recolhe dados de exploits de vários sites, como o Exploit-DB, o PacketStorm Security, e o 0Day Today, e também tem um processo de procura no GitHub de novos exploits.

É possível obter dados dela a partir de uma subscrição a alguns tópicos por email. Os emails podem ser obtidos gratuitamente, com um limite de subscrição a 5 tópicos. A Vulners também fornece planos pagos, que permitem realizar interações com a sua API, podendo assim obter-se mais informações sobre um sistema específico de uma forma programática.

A Vulners não acrescenta informação difícil de obter, sendo que todos os dados presentes podem ser conseguidos ao cruzar com a informação do CVE a ser explorado (como por exemplo, o CVSS de uma vulnerabilidade).

3.4.1.5 Discussão das fontes de exploits

Na Tabela 3.1 estão representadas as quatro fontes de informação de exploits consideradas, e os dados que estas podem fornecer. Estes dados estão separados por categorias, sendo a primeira relativa aos dados gerais. Nesta categoria encontram-se campos como:

- O título do exploit, que é uma breve descrição em poucas palavras do exploit;

- O seu autor, ou seja, quem submeteu o exploit nesta fonte de informação;
- A descrição do exploit, que contém uma explicação mais alargada do exploit e do seu funcionamento;
- A data de publicação do exploit na fonte de informação;
- A plataforma afetada por este;
- Uma síntese (no sentido criptográfico) do exploit, para que este seja identificável;

Destes dados gerais o mais importante é a plataforma afetada pelo exploit, uma vez que facilita o cruzamento interno. Este cruzamento interno poderá ser auxiliado com informações que possam estar presentes no título ou descrição do exploit. A síntese do exploit é um dado relevante para se poder distinguir exploits, sendo assim possível determinar a sua unicidade.

As informações sobre métricas de envolvimento sobre o conteúdo publicado poderão ser uma alternativa para julgar a relevância da informação, a partir de comentários e de cliques na publicação. Esta informação não deve ser utilizada unicamente para definir a relevância do *exploit*, mas poderá vir a ser utilizada para a complementar.

De seguida, insere-se a categoria de standards de vulnerabilidades e de risco, em que estão contidos os seguintes dados:

- O identificador do CVE correspondente à vulnerabilidade explorada;
- Se a fonte de informação disponibiliza métricas de envolvimento dos utilizadores com o conteúdo, como por exemplo, a quantidade de cliques e de comentários que uma entrada obteve.
- O nível de risco da vulnerabilidade, calculado pela fonte de informação.

O CVE da vulnerabilidade explorada é, destes três, a informação mais importante de obter, pois com este conseguimos obter informação sobre o seu CVSS. Em alternativa, em casos em que este identificador não está disponível, poderá ser útil usar uma possível métrica de risco própria, ou seja, o risco que este exploit apresenta de acordo com a fonte. Esta informação nunca irá substituir o CVSS, pois não é nem estandardizada nem controlada, podendo uma fonte indicar um risco muito mais alto que a realidade.

Como se pode verificar na Tabela 3.1, as fontes de informação de exploits enunciadas apresentam conjuntos de informação diferentes. A 0day.today (ZDT na tabela) é uma das fontes que passará para a fase de implementação, devido à quantidade de informação que esta fornece. No entanto, não é perfeita, e tem alguns defeitos:

- Nem todas as entradas têm descrição;
- A introdução de um identificador do CVE não é obrigatória, havendo entradas que este esteja disponível. No entanto, apresenta em todos os exploits uma métrica própria de risco;
- Não publica só informação verificada, apesar de verificar toda a informação postada recentemente;
- Não tem uma API.

A Exploit-DB (EDB na Tabela 3.1) também é uma fonte que será utilizada na fase de implementação, apesar de também ter alguns detalhes menos bons:

- Nem todas as entradas têm descrição, e quando existe, a descrição não está formatada num campo especial - esta informação está presente no corpo do exploit;
- Esta fonte verifica muito poucas entradas, apesar de postar apenas informação que passou um processo de moderação;
- A única maneira de aceder à plataforma é a partir do seu website.

A PacketStorm Security (PSS na Tabela 3.1) é também uma fonte que será utilizada na fase de implementação, pois também apresenta alguns detalhes úteis quando comparada com outras duas fontes:

- Não apresenta a plataforma, apesar desta ser possível de inferir a partir do título do exploit;
- Apresenta um resumo do exploit, podendo atribuir assim um identificador único ao ficheiro;
- Apenas é publicada informação verificada na plataforma.

A Vulners não é, para já, uma fonte considerada para este projeto. Isto deve-se ao facto de grande parte da informação apresentada ser republicações das três fontes referidas anteriormente. Um outro aspeto negativo reside no facto de esta fonte não verificar a informação que publica, indo assim contra as características desejáveis das fontes de informação.

3.4.2 Fontes de software malicioso

Para a secção de recolha de dados sobre software malicioso foram encontradas algumas plataformas consideradas como relevantes. As quatro plataformas identificadas foram a MalShare, o MalwareBazaar, a Inquest Labs, e a Virusbay. Nas seguintes subsecções serão descritas as fontes de informação, e de seguida será feita uma discussão sobre a sua viabilidade para inclusão no sistema ZeroDays v2.

3.4.2.1 MalShare

A MalShare [21] (MS) é uma base de dados de colaboração livre, que permite que os utilizadores publiquem e visualizem amostras de software malicioso. Esta fonte de informação facilita a livre colaboração de uma forma gratuita, mas não possui validação de entradas.

Após ter sido considerada, esta fonte de informação foi descartada por falta de dados que fornece sobre cada entrada.

3.4.2.2 MalwareBazaar

O MalwareBazaar [2] (BAch) é um projeto da Abuse.ch, que por sua vez é um projeto da Universidade de Ciências Aplicadas de Berna. Este tem como obtetivo servir de plataforma para recolha e partilha de amostras de software malicioso, de modo a promover a segurança contra estes ataques informáticos.

Este projeto tem uma base de dados de software malicioso, que é atualizada com uma elevada frequência. Esta plataforma contempla informações úteis para a realização deste projeto, como por exemplo o tipo e a família do software malicioso (por família entenda-se o software malicioso original de que este é uma variação).

3.4.2.3 InQuest Labs

A InQuest Labs [16] (INQ) é uma plataforma da InQuest.net, uma empresa dos Estados Unidos da América do ramo da cibersegurança. Esta plataforma contém três bases de dados diferentes: A *Deep File Inspection* (DFI), a *Aggregate Reputation Database* (REP-DB), e a *Indicators of Compromise Database* (IOC-DB). Para efeitos deste projeto, a base de dados que está em consideração é a DFI, no entanto as outras bases de dados poderão ser utilizadas futuramente para enriquecimento da informação recolhida por outras fontes.

A DFI [17] é uma base de dados que analisa automaticamente um determinado tipo de ficheiros (ficheiros do Microsoft Office, Open Office, folhas de cálculo e apresentações com um tamanho até 15MB), mas que no futuro pretende expandir a outros tipos ficheiros. Esta plataforma tem um elevado número de entradas, mas com a versão gratuita apenas é possível observar as últimas 1337 publicações. Cada entrada é classificada com um de três estados: *malicioso* quando é confirmado que é um ficheiro malicioso, *suspeito* quando é um ficheiro que apresenta alguns indicadores de que pode ser malicioso, mas que ainda não há certeza, e *desconhecido* quando ainda não existe informação suficiente para tomar uma decisão. Esta fonte de informação, embora um pouco limitada na quantidade de ficheiros, pode ser útil, pois apresenta informação útil sobre os mesmos.

3.4.2.4 VirusBay

A VirusBay [43] (VB) é uma plataforma de colaboração entre vários profissionais de cibersegurança. Esta plataforma contempla um fórum de discussão, um sistema de pedidos de classificação de amostras, e uma base de dados de softwares maliciosos identificados por esta comunidade.

Esta base de dados não contempla informações essenciais, como a plataforma afetada, o tipo de software malicioso, e a sua família, pelo que estes dados (quando existem) são postados nas etiquetas de cada entrada. Esta fonte de informação não confirma a informação publicada, e tem uma baixa quantidade de entradas. De acordo com o seu website, na altura da escrita deste relatório, esta contém apenas 6735 entradas (à data de 19 de setembro de 2022), com menos de 20 entradas no último ano.

3.4.2.5 Discussão das fontes de software malicioso

	Fontes			
	MS	BAch	INQ	VB
Tamanho		✓	✓	
Síntese	✓	✓	✓	✓
Plataforma				
Ficheiro Disponível	✓	✓	✓	✓
Tipo de Ficheiro	✓	✓	✓	
MIME do Ficheiro		✓	✓	
Data do Primeiro Avistamento		✓		✓
CVE				
Tipo de Software Mal.		✓	✓	
Família de Software Mal.		✓		
Verifica Informação		✓	✓	
Apenas Informação Verificada				
Website	✓	✓	✓	✓
API	✓	✓	✓	

Tabela 3.2: Fontes de Informação de Software Malicioso e o seu Conteúdo

Na Tabela 3.2 estão representadas as quatro fontes de informação de software malicioso consideradas, e os dados que estas podem fornecer. Os dados estão separados por categorias, sendo a primeira os *dados gerais*. Nesta categoria encontra-se:

- O tamanho do ficheiro com o software malicioso;
- Uma síntese (no sentido criptográfico) do software malicioso;
- A plataforma afetada pelo software malicioso;
- A disponibilização por parte da plataforma do software malicioso;
- O tipo de ficheiro que o software malicioso utiliza;
- O MIME do tipo de ficheiro;
- A data do seu primeiro avistamento;
- O CVE-ID das vulnerabilidades que são exploradas por este software malicioso.

Nesta categoria, os dados mais importantes são a plataforma afetada, a data do primeiro avistamento, e o CVE-ID associado. A informação sobre a plataforma afetada é relevante para poder ser feita a confrontação interna, e a informação sobre a data do primeiro avistamento pode ajudar a compreender a maturidade do software malicioso e as defesas que poderão já existir contra este. Infelizmente, nenhuma das fontes fornece dados sobre o CVE-ID nem possui a plataforma afetada, pelo que a confrontação com os *assets* da empresa será limitada de uma forma bastante negativa, senão mesmo inviabilizada.

Na secção seguinte encontram-se os *dados sobre o software malicioso*. Nesta secção encontram-se:

- O tipo de software malicioso;

- A sua família.

Estes dados são importantes para obter mais informação à volta do software malicioso e do seu comportamento, pois indica o padrão geral que seguem (por exemplo, no caso de um *spyware*, observa as ações da vítima).

De seguida, são apresentados *dados sobre o tratamento da informação*. Nesta categoria incluem-se:

- Se a fonte de informação verifica a informação que publica;
- Se esta apenas publica informação que verificou.

A distinção entre estes dois aspetos é fundamental, pois uma fonte pode publicar informação e depois verificá-la, e outra fonte pode publicar apenas informação depois de a verificar. No primeiro caso, o processo é expedito, pois não existe uma verificação. No segundo caso, a informação é mais fiável, pois é validada.

Por último, são representadas as maneiras de aceder à informação que as fontes de informação disponibilizam:

- Se a fonte de informação tem um website;
- Se é disponibilizada uma API.

Esta informação é relevante para a implementação, pois permite aceder à informação presente nas fontes de informação de uma forma mais facilitada e consistente: se uma fonte de informação modificar o seu website, ou se criar entraves à recolha automatizada do website, pode tornar inviável o uso de certa fonte de informação, pois pode deixar de ser possível recolher informação desta. Felizmente, todas as fontes de informação possuem uma API, à exceção da VB.

A tabela foi preenchida de acordo com a informação disponibilizada por cada uma das fontes de informação consideradas, de modo a poderem ser tiradas conclusões. Existem fontes que publicam aspetos mais importantes que outras, como é o caso do MalwareBazaar (na tabela, BAch) que contém muita informação útil, em contraste com a VirusBay e a MalShare, que fornecem um número reduzido de características. A VirusBay também possui uma grande falta de atividade, pelo que também reduz a sua viabilidade.

Apesar da completude da informação que a InQuest Labs consegue oferecer, esta fonte é limitada no seu âmbito, apenas tratando documentos que possam estar infetados.

Devido à falta de verificação de informação aliada à falta de informação descrita anteriormente por parte da MalShare, esta não pode ser utilizada como fonte de recolha de informação.

Ponderando as várias características das fontes, foram selecionadas como fontes o MalwareBazaar e o InQuest Labs. A VirusBay e a MalShare atualmente encontram-se descartadas, mas se estas melhorarem as características que impedem a sua viabilidade, poderão vir a ser incluídas no futuro.

3.4.3 Fontes de cruzamento

Como referido na Secção 2.2.3, para o funcionamento do sistema ZeroDays v2, é necessário a definição e interação com fontes de cruzamento, para que seja possível aferir o conhecimento interno da empresa sobre uma determinada ameaça. Sem estas, é impossível aferir se uma dada ameaça é conhecida internamente.

3.4.3.1 Fontes de cruzamento de exploits

Para confirmar se um exploit é de dia-zero ou não, será utilizada como fonte de cruzamento interna à organização a base de dados do produto Qualys VMDR 2.0. Nesta fonte de informação é possível pesquisar uma vulnerabilidade a partir do seu CVE, e saber o estado do conhecimento de exploits desta. Como não é possível obter um hash estandardizado do exploit, a utilização da outra fonte de cruzamento, a IBM X-Force Exchange, fica comprometido e não poderá ser realizado.

3.4.3.2 Fontes de cruzamento de software malicioso

Assim como é necessário definir o conhecimento interno da empresa para decretar se um exploit é dia-zero ou não, também é necessário fazer o mesmo para software malicioso. Como as fontes usadas atualmente não possuem informações sobre o CVE-ID da vulnerabilidade que os os software maliciosos exploram, não é possível fazer o cruzamento destas informações com a base de conhecimento oferecida pelo Qualys VMDR 2.0. No entanto, como se consegue obter um hash dos ficheiros de software malicioso, é possível utilizar a IBM X-Force Exchange para aferir a novidade de uma ameaça, bem como viabiliza o uso da base de conhecimento do VirusTotal, que também é utilizado para decretar se uma ameaça é dia-zero.

3.4.4 Fontes de confrontação interna

Para realizar a confrontação interna é necessário obter informações sobre todos os *assets* da empresa. Infelizmente, não existe um cadastro completo com todo o software e hardware que a empresa dispõe, havendo apenas uma listagem do sistema operativo de cada *asset* num índice na plataforma ElasticSearch. Assim, tal como no ZeroDays v1, terá de ser feito um cruzamento com palavras chave, que mapeia palavras que estão relacionadas com determinados sistemas operativos com identificadores desses mesmos sistemas operativos. Realizar este tipo de cruzamento é possível quando existe informação suficiente, o que é o caso dos exploits. No caso do software malicioso, tal cruzamento é impossível de realizar sem tomar grandes assunções, pelo que não será feito.

Capítulo 4

ZeroDays v2

Com vista a definição de um processo de monitorização de exploits e malware dia-zero, foi proposta e implementada uma solução que permite a sua recolha, análise, monitorização, e notificação, de forma a informar a Direção de Cibersegurança de novas ameaças.

4.1 Proposta da solução

Nas seguintes subsecções irá ser feita a proposta da solução ZeroDays v2. Esta solução é restrita pelos requisitos definidos na Subsecção 4.1.1, que definem características que o projeto deve satisfazer. De seguida, é feita uma descrição da arquitetura dos motores do sistema, bem como a sua interligação.

4.1.1 Requisitos

Para este projeto foram definidos os seguintes requisitos, que delinearam as características fundamentais que o sistema ZeroDays v2 deve assegurar:

- **Modularidade** - A modularidade dos componentes de recolha de informação deve ser não só preservada, como se possível aumentada. Isto permite que a simplicidade e facilidade de manutenção do sistema seja mantida ou acrescida.
- **Simplicidade** - O projeto deve optar pela simplicidade sempre que possível, de modo a evitar erros de implementação, bem como facilitar a sua manutenção.
- **Confiança** - As fontes de informação escolhidas devem ter como característica um grau elevado de confiança, de forma a que os dados recolhidos e tratados sejam relevantes.
- **Conclusividade** - A solução tem de produzir dados detalhados e conclusivos, de forma a que possa ser possível atuar sobre estas informações.
- **Velocidade** - O processo de notificação deve ser realizado em tempo útil, para que a informação relatada seja relevante quando é produzida.
- **Certeza** - As notificações só devem ser feitas quando existir um grau elevado de certeza, ou quando o risco for alto o suficiente que se justifique. Isto permite que não seja relatada informação inútil, evitando assim desperdício de tempo, bem como uma situação de dessensibilização aos alertas gerados.

4.1.2 Arquitetura

Com o objetivo de obter e propagar informação de ameaças dia-zero, é proposta a seguinte arquitetura, composta por três componentes (representado na Figura 4.1):

- A **Componente de Vulnerabilidades**, desenvolvida no âmbito do projeto ZeroDays v1;
- A **Componente de Exploits**, criada para recolher e gerir novas ameaças do tipo exploit;
- A **Componente de Software Malicioso**, criada para recolher e gerir novas ameaças do tipo software malicioso.

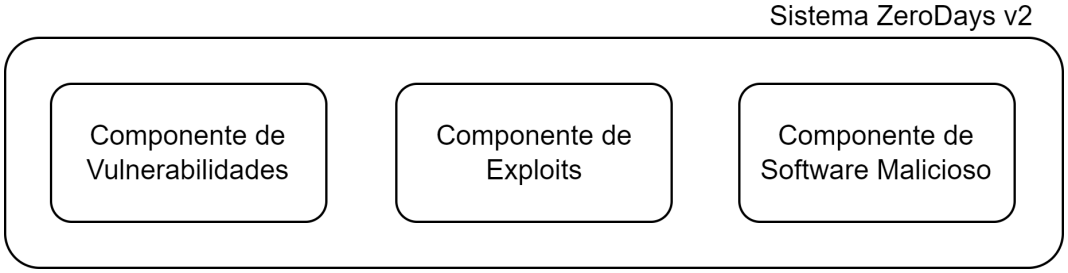

Figura 4.1: Esquema geral do sistema ZeroDays v2.

As duas últimas componentes foram desenvolvidas no âmbito do projeto ZeroDays v2 de uma maneira similar, de maneira a conseguir aumentar a simplicidade e modularidade do sistema. Cada uma destas componentes é composta por três motores (representado na Figura 4.2): O motor de recolha, o motor de processamento, e o motor de notificação.

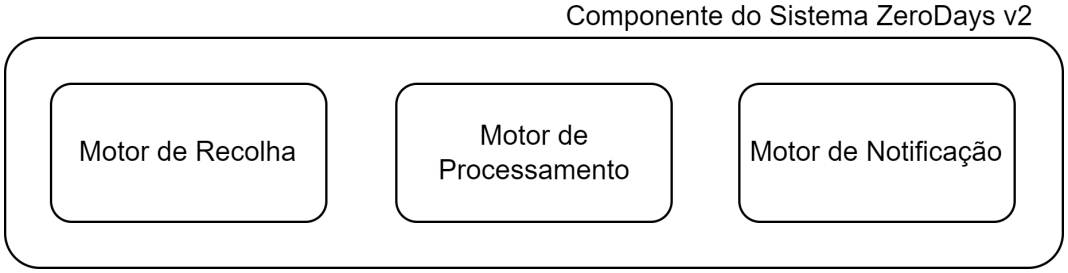

Figura 4.2: Esquema geral de uma das componentes do sistema ZeroDays v2.

O motor de recolha desempenha a função da interação com as fontes de informação, independentemente da forma de obtenção dos dados. Este motor faz a coleta da informação, e fornece um ficheiro *csv* com os dados da fonte de informação. O motor de recolha é composto por módulos de recolha, em que cada um acumula os dados de uma só fonte de informação. Desta maneira, se existir uma falha na execução em qualquer um dos módulos (causada, por exemplo, por uma mudança na disposição da informação na fonte) esta não deverá comprometer a execução dos restantes módulos de recolha de informação.

O motor de processamento trata da informação. Isto inclui a sua normalização, bem como o relacionamento com as fontes de cruzamento, fontes de confrontação, e com as execuções anteriores do processo. Este recebe os ficheiros *csv* produzidos pelo motor de recolha, e guarda a informação numa base de dados.

Por último, o motor de notificação assegura a notificação das partes interessadas, através do envio de emails com a informação relevante. Estes emails devem conter a

informação sobre a ameaça dia-zero, o URL de onde foram obtidos os dados, e informações sobre os sistemas afetado por esta. Estas notificações devem apenas ocorrer quando o sistema encontra uma ameaça que categoriza como dia-zero, e que classifica como relevante para os sistemas informáticos da organização. Como esta informação tem um elevado grau de sensibilidade, esta informação deve enviada exclusivamente à equipa do CyberSOC da Direção de Cibersegurança da empresa.

4.1.2.1 Arquitetura do motor de recolha

O motor de recolha é composto por módulos especializados que coletam informação, sendo executados sequencialmente mas independentes uns dos outros. Isto na prática significa que os módulos de recolha de informação podem falhar de um modo independente, em que a paragem de um módulo não vai impedir que os outros que estão funcionais cumpram o seu propósito. Esta arquitetura é a mesma quer na componente de exploits como na de software malicioso.

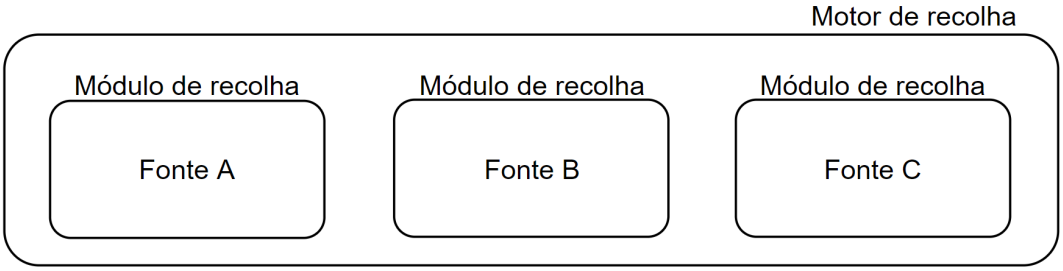

FIGURA 4.3: Esquema geral do motor de recolha

Para o desenvolvimento dos módulos de recolha de informação, optou-se por uniformizar o processo de modo a que seja simples estender a fonte a outras fontes. Cada módulo interage com três ficheiros:

- Ficheiro de **log** - Este ficheiro guarda os dados da última entrada que foi obtida, para que o módulo saiba onde terminou a recolha anterior.
- Ficheiro **.bp.log** - Para tornar a aferição do funcionamento de cada módulo mais simples, após todas as ações do módulo é acrescentada informação sobre essa ação ao ficheiro .bp.log, que é reescrito sempre que o módulo inicia a sua execução. Ao criar este ficheiro, obtém-se efetivamente informação de debug das ações do processo, que permite uma maior facilidade de manutenção e resolução de problemas do módulo.
- Ficheiro CSV (***.csv***) - Neste ficheiro é guardado o resultado da recolha de informação, em formato *csv* (Comma Separated Values), para que seja exportado para o motor de processamento.

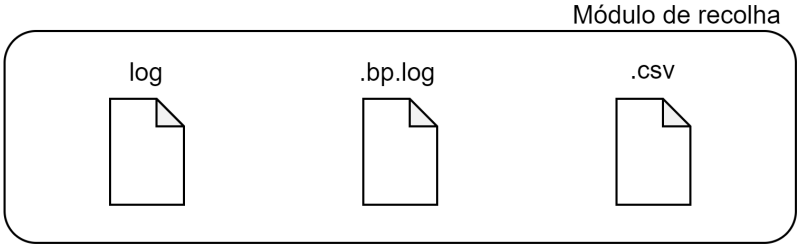

FIGURA 4.4: Ficheiros com que cada módulo de recolha interage

O módulo de recolha segue um esquema geral de execução que também foi replicado para as diversas fontes (como representado na Figura 4.5):

- **Leitura de Ficheiro de Log** - Nesta etapa o processo lê a informação incluída no ficheiro de log, e guarda na memória o seu conteúdo.
- **Recolha de informação** - Interage com a fonte acumulando a informação disponível e mais recente. Esta ação também realiza alguma limpeza dos campos recolhidos, para que a informação produzida esteja uniformizada.
- **Guardar num ficheiro *csv*** - Se a fonte de informação tiver produzido novos resultados, estes são guardados num ficheiro *csv*.
- **Exportar ficheiro *csv*** - O processo exporta o ficheiro gerado pela recolha para o motor de processamento.
- **Atualizar ficheiro de log** - O processo reescreve o ficheiro de log com a informação atualizada relativa à última recolha.

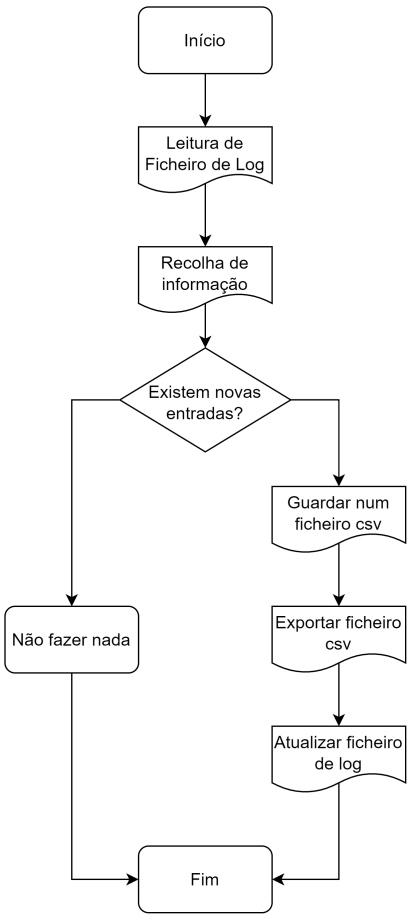

FIGURA 4.5: Esquema geral da arquitetura dos módulos de recolha de informação

Deste modo, ao realizar a recolha de dados desta maneira, é possível assegurar a modularidade do software. Esta também permite uma manutenção facilitada no futuro, caso se deseja adicionar ou retirar fontes de informação.

4.1.2.2 Arquitetura do motor de processamento

O motor de processamento é composto por dois segmentos, o segmento de Triagem e Interpretação, e o segmento de Cruzamento e Confrontação.

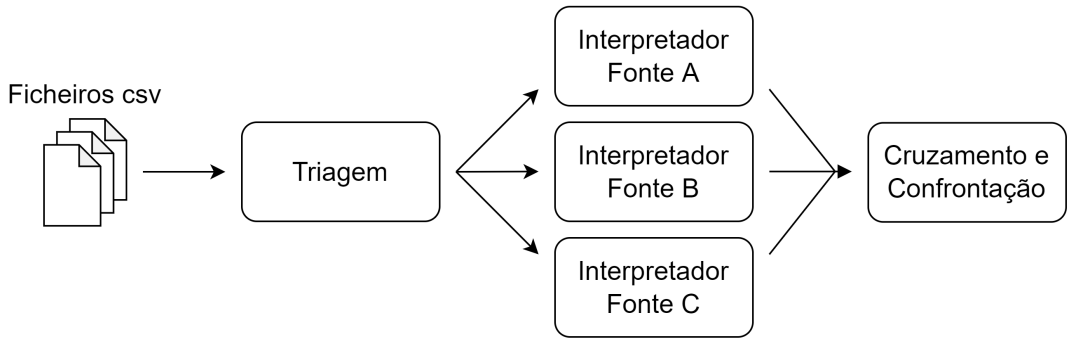

FIGURA 4.6: Primeira parte do motor de processamento - Triagem e Interpretação

No segmento de Triagem e Interpretação, o processo lê os ficheiros produzidos pelo motor de recolha de informação, e atribui esse ficheiro a um interpretador que compreende o formato dos dados dessa fonte.

A existência dos interpretadores aumenta o número de mudanças que será necessário desenvolver para adicionar uma nova fonte de informação. No entanto, introduz flexibilidade na ferramenta porque permite adicionar e alterar informação com base na fonte que foi recolhida, como também acrescentar metainformação, como a hora em que a entrada foi analisada. Exemplos disto são a atribuição de um nível de confiabilidade às fontes, e a correção de formatos de dados diferentes, tornando-os semelhantes.

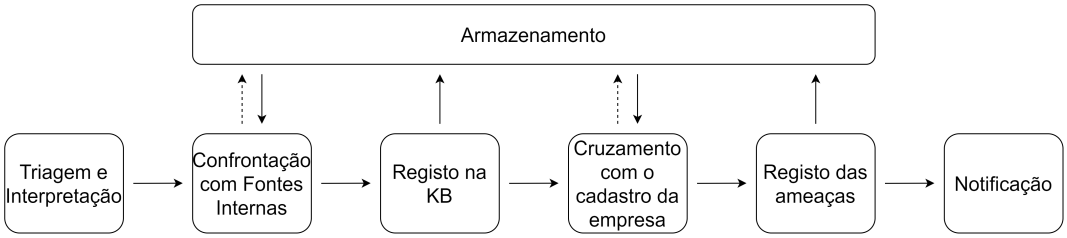

FIGURA 4.7: Segunda parte do motor de processamento - Cruzamento e Confrontação

Depois da Triagem e Interpretação da informação, os dados entram na segunda parte do motor de processamento, o Cruzamento e Confrontação. Nesta fase, os dados são comparados com as fontes internas de informação, que irão ajudar a determinar se a ameaça já é conhecida pela organização. Após esta confrontação, os dados são armazenados numa KB associada a esse tipo de ameaça, para que possam ser acedidos posteriormente.

Depois de armazenadas na KB, as ameaças são cruzadas com o cadastro da empresa, de modo a aferir que assets estão vulneráveis. Essa informação é guardada num registo que indica que assets estão vulneráveis a determinada ameaça, para que depois possa ser consultada pelas entidades responsáveis.

4.1.2.3 Arquitetura do motor de notificação

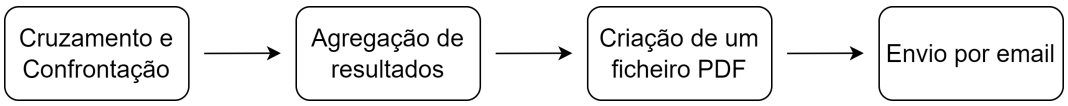

Figura 4.8: Esquema geral do motor de notificação

Embora o motor de notificação seja composto por um único segmento, este cumpre um dos aspetos fundamentais do projeto, a notificação das entidades competentes.

O motor de notificação primeiro agrupa os dados sobre as ameaças, para poder mostrar toda a informação que tem sobre uma ameaça no mesmo sítio. De seguida, cria um ficheiro PDF com a informação sobre a ameaça. O critério de agrupamento na componente de exploits é o seu hash de comparação, e na componente de software malicioso o *sha-256* do ficheiro. Este ficheiro PDF é gerado quando o processo encontra ameaças dia-zero que são categorizadas como relevantes para a organização, pelo que não é gerado nenhum ficheiro quando estas não são detetadas.

No caso em que a ameaça é um exploit, as informações enviadas são o seu título, o CVE-ID da vulnerabilidade que explora, a sua descrição, o seu autor, o nome da fonte de de onde foi recolhida, os URLs para poder aceder ao conteúdo do exploit e para aceder à entrada na fonte, a razão pelo qual foi determinado como dia-zero, e o sistema informático que foi detetado como vulnerável.

Em alternativa, na situação em que a informação recolhida sobre a ameaça é sobre um software malicioso, a informação relatada é composta pelo *sha-256* do ficheiro, a data do seu primeiro avistamento, a extensão do ficheiro, o seu tamanho, o seu MIME, a sua assinatura, e um campo de informações adicionais disponibilizado por uma das fontes de informação, que ajuda a entender o software malicioso.

De seguida, o sistema envia o ficheiro PDF com as informações sobre as ameaças por email à equipa do CyberSOC da Direção de Cibersegurança da empresa.

Com o fim do motor de notificação também vem o fim do processo ZeroDays v2, aguardando até à sua próxima iteração.

4.2 Implementação

Com base na arquitetura descrita anteriormente foi realizada a concretização do sistema ZeroDays v2. Nesta secção serão definidas as tecnologias e ferramentas que foram utilizadas na implementação do sistema (Subsecção 4.2.1), será descrita a reengenharia necessária do projeto ZeroDays v1 (Subsecção 4.2.2), e será explicado o desenvolvimento dos sistemas de recolha e tratamento de exploits e de software malicioso (Subsecções 4.2.3 e 4.2.4).

4.2.1 Tecnologias e ferramentas utilizadas

Para o armazenamento de informação foi utilizada a Elastic Stack [9], um conjunto de tecnologias que tem como vantagens a fácil visualização de dados, bem como a facilidade de armazenamento de grandes quantidades de informação. Este ecossistema é composto pelas seguintes ferramentas:

- **Kibana** - Permite a visualização de informação a partir de uma componente gráfica, bem como a sua representação através de gráficos.

- **ElasticSearch** - Trata do armazenamento e da pesquisa de informação por via de uma API RESTful.
- **LogStash** - Tem como função o armazenamento de grandes quantidades de dados (esta difere da anterior no sentido em que o propósito principal é o armazenamento, sendo que é mais utilizada para armazenar registos).
- **Beats** - Embora não tenha sido usada, possibilita a obtenção de métricas de utilização.

Para implementar o motor de recolha, foi utilizada a solução Blue Prism RPA [5], uma ferramenta de Robotic Process Automation. Esta permite a automatização de processos por via de "robôs"que simulam a ação humana. Deste modo, os módulos do motor de informação foram todos desenvolvidos com recurso a esta ferramenta, que permite a modularização da recolha da dados.

O uso desta ferramenta simplifica a manutenção dos módulos de recolha de informação dado que requer pouco código e pouca complexidade de uso, permitindo ainda realizar operações mais complexas a partir da execução de código escrito em Visual Basic. No entanto, esta possui menor desempenho que o desenvolvimento de módulos escritos em Ruby. Como um dos requisitos do projeto é a simplicidade e facilidade de manutenção, optou-se por desenvolver o motor de recolha com recurso ao Blue Prism.

Para a implementação dos restantes motores, foi utilizada a linguagem de programação Ruby, assim como a seguinte seleção de gemas (do inglês *gems*, são bibliotecas de código que podem ser reutilizadas para realizar determinadas funções):

- **dotenv** para a leitura de variáveis de ambiente;
- ***csv***, ***json***, ***yaml*** e ***safe_yaml*** [1] para interação com os ficheiros de seu nome;
- ***fileutils*** para interação com o sistema de ficheiros, como por exemplo copiar, mover e apagar ficheiros;
- ***nokogiri***, ***nethttp*** e ***uri*** para a recolha de informações de websites necessária pelo motor de processamento;
- ***time*** para a interação com datas e horas;
- ***digest*** para a criação de hashes de texto;
- ***singleton*** para a aplicação do padrão de design com o mesmo nome;
- ***logger*** para o registo de ações em ficheiros de log;
- ***erb*** para a criação de templates de texto fáceis de editar;
- ***prawn*** para a criação de ficheiros PDF;
- ***mail*** para o envio de emails;
- ***elasticsearch*** para a interação com a plataforma com que partilha o nome.

[1] A utilização desta gema em conjunto com a anterior é desnecessária - no entanto, um dos scripts em Ruby da versão anterior do sistema ZeroDays utilizava a última, enquanto que um outro utilizava a primeira. Optou-se por não alterar este detalhe, porque não afeta os resultados produzidos.

4.2.2 Reengenharia da atual implementação

Como um dos objetivos do projeto é a melhoria do ZeroDays original, optou-se por começar por esta etapa de forma a aumentar a sua robustez.

Uma vez que uma fonte de informação alterou a maneira como apresentava os dados, a recolha de informação não estava a funcionar. Para apurar a razão do problema, desenvolveu-se uma ferramenta que cria um registo num ficheiro sobre cada ação executada com sucesso pelo módulo de recolha. Assim, foi possível determinar que uma das fontes de informação de vulnerabilidades, neste caso a VulDB, estava inoperacional, por uma mudança na sua API.

No entanto, a inoperacionalidade de uma fonte estava previsto no projeto ZeroDays inicial, dado ser um dos requisitos do sistema [42]. No entanto, por um erro de implementação, esta modularidade não foi preservada, pois não prevê uma falha na recolha de informação de uma das fontes. Depois de corrigida esta falha, a recolha de informação retomou à normalidade, ficando a decorrer de acordo com a calendarização que foi definida.

Quando foi efetuada esta correção, observou-se uma irregularidade no sistema - o motor de processamento não executava com a regularidade que era de esperar. Este problema devia-se ao método com que o módulo realizava a deteção dos novos ficheiros recolhidos. Na implementação do projeto ZeroDays inicial, definiu-se que o motor de processamento iria iniciar, e depois ficaria a aguardar por modificações em ficheiros numa pasta definida. Este procedimento é ilustrado na Figura 4.9.

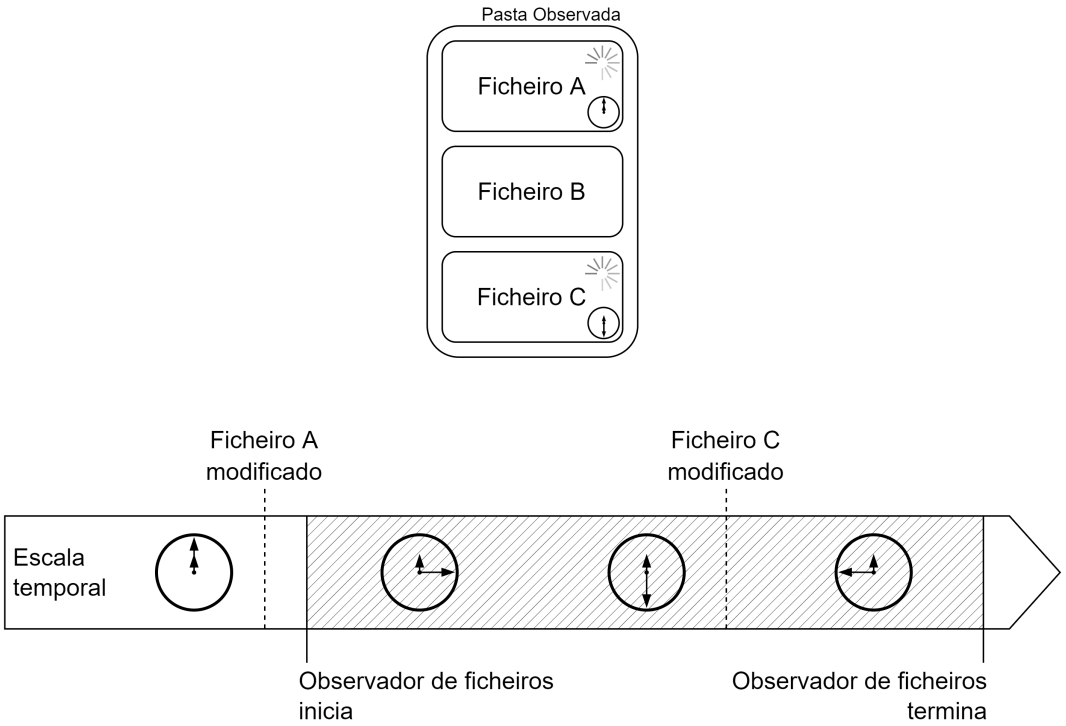

FIGURA 4.9: Erro proveniente da implementação com base num observador de ficheiros. Se o Ficheiro A é atualizado antes de o observador começar a execução, então este não deteta a sua modificação.

Assim, assumindo que os relógios entre os sistemas estão sincronizados, o motor de recolha de informação e o motor de processamento começam a execução ao mesmo tempo, tendo o motor de processamento uma janela de tolerância de 15 minutos. Ao fim dessa janela de tolerância, o motor de processamento deixa de vigiar a pasta, não

detetando modificações futuras. Contudo, por erro de calendarização, o processo de recolha de informação estava a iniciar-se uns minutos antes do esperado, o que impedia que o motor de processamento detetasse qualquer mudança no ficheiro produzido.

Para corrigir este problema, modificou-se o horário do motor de processamento para executar uns minutos antes de o processo de recolha começar a trabalhar. Esta correção não é robusta, embora resolva o problema. Na implementação do ZeroDays v2, optou-se por utilizar uma maneira diferente de detetar mudanças nos ficheiros, evitando-se esta questão.

O motor também possuía um processo que verificava se estava a ser executado numa das horas predefinidas. Como o cálculo das horas tinha erros, foi necessário refazer esses cálculos.

Também foi necessária uma correção à produção de logs do sistema anterior. Da maneira que estava implementado, o motor de processamento escrevia num ficheiro todas as ações que realizava ao nível da interação com o elasticsearch, sem nunca apagar o ficheiro escrito anteriormente - ao início do dia, o programa renomeava a extensão do ficheiro onde registava as suas ações para este conter a data do dia anterior.

Este procedimento impactou especialmente o funcionamento do sistema, pois quando uma operação falhava, o programa tentava novamente de uma forma incondicional. Isto apresenta um problema grave, pois quando existia um erro na interação com o elasticsearch, o programa escrevia no ficheiro de registo até não ter mais espaço disponível.

Este ciclo incondicional foi corrigido ao adicionar um número fixo de repetições que cada operação pode realizar, e atingindo este o programa termina com erro. Também foi adicionado um sistema de limpeza destes ficheiros, em que os registos com mais de 7 dias são apagados.

Deste modo, após realizadas as melhorias referidas acima, começou a ser implementado o projeto ZeroDays v2.

4.2.3 Exploits

O desenvolvimento do ZeroDays v2 iniciou-se com a implementação do componente relativo aos *exploits*. Para este tipo de informação foi definido que o processo iria executar três vezes ao dia, de oito em oito horas (neste caso, foram definidas como horas de execução as 0h, 8h, e 16h). Este horário foi escolhido para ir de encontro com o horário que já era utilizado no ZeroDays v1. Esta calendarização poderá ser alterada se o caudal de informação ou a redução desse intervalo de tempo se justificar.

4.2.3.1 Recolha de informação de exploits

A componente da recolha de informação seguiu a arquitetura descrita na secção anterior (Figura 4.5). Esta é composta por três módulos de recolha de informação, um para cada fonte de informação.

Para proceder à recolha de informação, o módulo abre um browser, que durante a implementação ficou definido como sendo o Google Chrome, devido à sua fácil integração com a ferramenta Blue Prism. O módulo depois abre o website de onde vai ser recolhida a informação, recolhe a informação presente em cada uma das novas entradas, e guarda a sua informação na memória. Assim que o processo deteta que está a aceder a uma página que ainda não contém nenhuma entrada, este tenta aceder às cinco páginas seguintes, e se nenhuma delas for uma entrada válida, então o processo termina a recolha de dados do website. Depois de terminada a recolha de

informação, o módulo fecha o browser, de modo a não criar problemas de continuidade para os módulos seguintes.

Para poder diferenciar os exploits entre si e encontrar os que são iguais, seria necessário que as fontes de informação disponibilizassem um resumo do ficheiro com o exploit. Como apenas a fonte PacketStorm Security o fornece, é impossível proceder de uma maneira tão simples. Criar uma hash com base no ficheiro completo também foi equacionado, mas não funciona de modo imediato porque as fontes de informação, apesar de disponibilizarem o mesmo exploit, às vezes acrescentam texto ao corpo do exploit. Um exemplo disto é o 0day.today, que remove a data dos explois, para a dispor no fim do ficheiro.

Na figura 4.10 é possível visualizar uma ocorrência deste problema, em que as três fontes de informação possuem o mesmo exploit, embora devido à formatação e à adição de texto causam que os ficheiros sejam diferentes.

```
$>diff -u EDB_49601.txt  PSS_161586.txt  | diffstat -m
 PSS_161586.txt |   37 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 1 file changed, 37 modifications(!)
$>diff -u EDB_49601.txt  ZDT_35878.txt  | diffstat -m
 ZDT_35878.txt |   54 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 1 file changed, 54 modifications(!)
$>diff -u PSS_161586.txt  ZDT_35878.txt  | diffstat -m
 ZDT_35878.txt |   54 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 1 file changed, 54 modifications(!)
```

FIGURA 4.10: Diferenças entre três entradas iguais (entradas com o ID 49601 do Exploit-DB, ID 161586 do PacketStorm Security e ID 35878 do 0day.today), com recurso às ferramentas *diff* e *diffstat*

Assim, foi desenvolvido um sistema de comparação de exploits, em que é feito um resumo do conteúdo do exploit, depois de este ser limpo. Esta limpeza foi criada e adaptada de maneira a remover as modificações comuns que cada uma das fontes efetua, bem como eliminar todos os caracteres especiais que estão no corpo do exploit, e que podem originar erros. Os caracteres apagados correspondem a todos os caracteres invisíveis, e ainda os caracteres de espaço e de nova linha. Também são removidas todas as linhas do ficheiro que comecem pelo caracter '#' (cardinal), que as fontes de informação utilizam para denotar um comentário. Esta limpeza é realizada a partir de uma expressão regex, que está representada no Apêndice A. Assim, ao submeter os ficheiros a estas modificações, é possível equiparar dois exploits de fontes diferentes através do resumo SHA-256 que partilham. A este resumo foi atribuido o nome **Comparison Hash** (CHash), em português **Hash de Comparação**, para não ser confundido com os resumos que o PacketStorm Security fornece.

Outras alternativas a este sistema de comparação foram consideradas:

- Guardar o texto do exploit, e realizar comparações sobre estes - esta alternativa foi descartada pois envolvia guardar o exploit, o que podia comprometer a segurança interna da empresa. Em segundo lugar, a comparação entre textos seria relativamente lenta em termos computacionais, quando empregue uma solução baseada no algoritmo que calcula a distância Levenshtein. Um outro grande problema seria a sua viabilidade a longo prazo, em que o desempenho do sistema seria comprometido conforme o número de entradas na tabela fosse aumentando, pois cada nova entrada teria de ser comparada com todas as anteriores.

- Realizar um hash do corpo do exploit sem modificações - se as fontes de informação fossem perfeitas, e publicassem a informação sempre de forma igual,

esta alternativa seria viável. No entanto, se a comparação fosse realizada desta maneira, nenhuma entrada da fonte ZDT seria equiparada noutra fonte de informação, e muitas das entradas nas outras fontes não seriam equiparadas entre si, devido a pequenas modificações no texto.

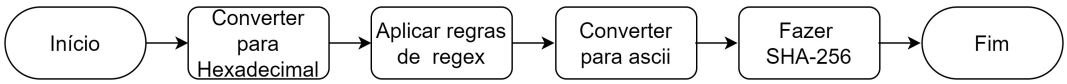

FIGURA 4.11: Esquema da limpeza do texto do exploit e de criação do Hash de Comparação

Depois de criado um método para comparar exploits, é possível diminuir o número de entradas duplicadas que antes eram identificadas como não o sendo, bem como avaliar as fontes na sua capacidade de produção de informação nova ou exclusiva.

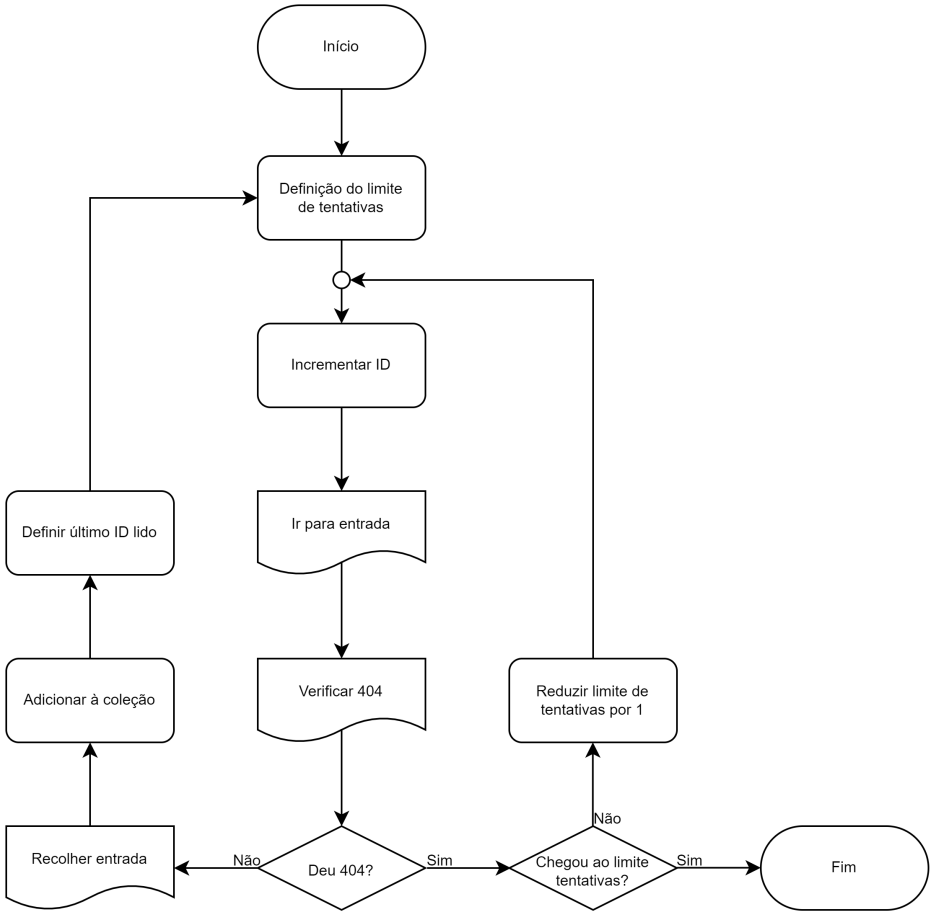

FIGURA 4.12: Esquema da recolha de informação relativa à fonte de informação ExploitDB

Na Figura 4.12 são ilustrados os principais passos na recolha de informação da fonte ExploitDB. A fonte apresenta a informação na web seguindo o esquema geral introduzido na Figura 4.13. O processo é composto por várias etapas, que também existem nos outros módulos de outras fontes:

- **Definição do limite de tentativas** - É definida uma variável que controla o número de tentativas que o processo irá realizar. Se esta já estiver definida, ela irá voltar a ter o seu valor inicial. Para efeitos da implementação, foi definido

o valor máximo de tentativas como 5, sendo possível adaptá-lo no futuro. Na prática, isto significa que após 5 ocorrências de seguida do erro 404, o processo determinará que chegou ao fim da recolha.

- **Incrementar ID** - A variável que guarda o ID é incrementada por 1. Esta variável será utilizada para identificar a entrada que está a ser recolhida.
- **Ir para entrada** - O processo remete o browser para a entrada da fonte de informação que possui o ID atualmente definido.
- **Verificar 404** - Depois de ir para a entrada, o processo verifica se a página indicada contém uma mensagem de 404. Esta página pode ocorrer caso a entrada tenha sido apagada, ou caso ainda não tenha sido preenchida.
- **Recolher Entrada** - O processo recolhe os campos da entrada, e adiciona-os à coleção dos dados recolhidos. Depois disto, é definida uma variável que contém o último ID recolhido que será escrita no ficheiro de log que guarda a última iteração.
- Se o website retornar uma página de 404, e o número de tentativas não tiver chegado a zero, este será subtraído por um. Se o número de tentativas tiver chegado a zero, então o processo termina a recolha.

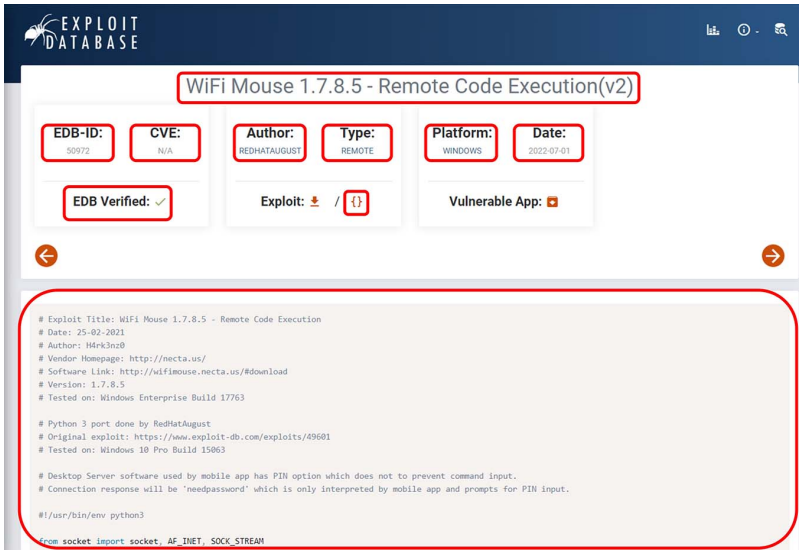

FIGURA 4.13: Principais componentes de cada entrada do ExploitDB que são recolhidas, com um quadrado vermelho à sua volta.

Dentro do passo Recolher Entrada, são obtidos os seguintes campos do website:

- **Título**, **ID**, **autor**, **tipo de exploit**, **plataforma**, **categoria** e **data de publicação** do exploit, que estão sempre presentes em todas as entradas da fonte de informação.
- **Estado da verificação** do exploit, ou seja, se a fonte verificou a informação.
- **ID do CVE** da vulnerabilidade que o exploit explora, se disponível.
- **Link de referência**, que corresponde à página onde se obteve a informação, e **link para o exploit**, onde podemos ver o código fonte do exploit.

- É temporariamente recolhido o texto do exploit, de forma a criar uma **hash de comparação**.

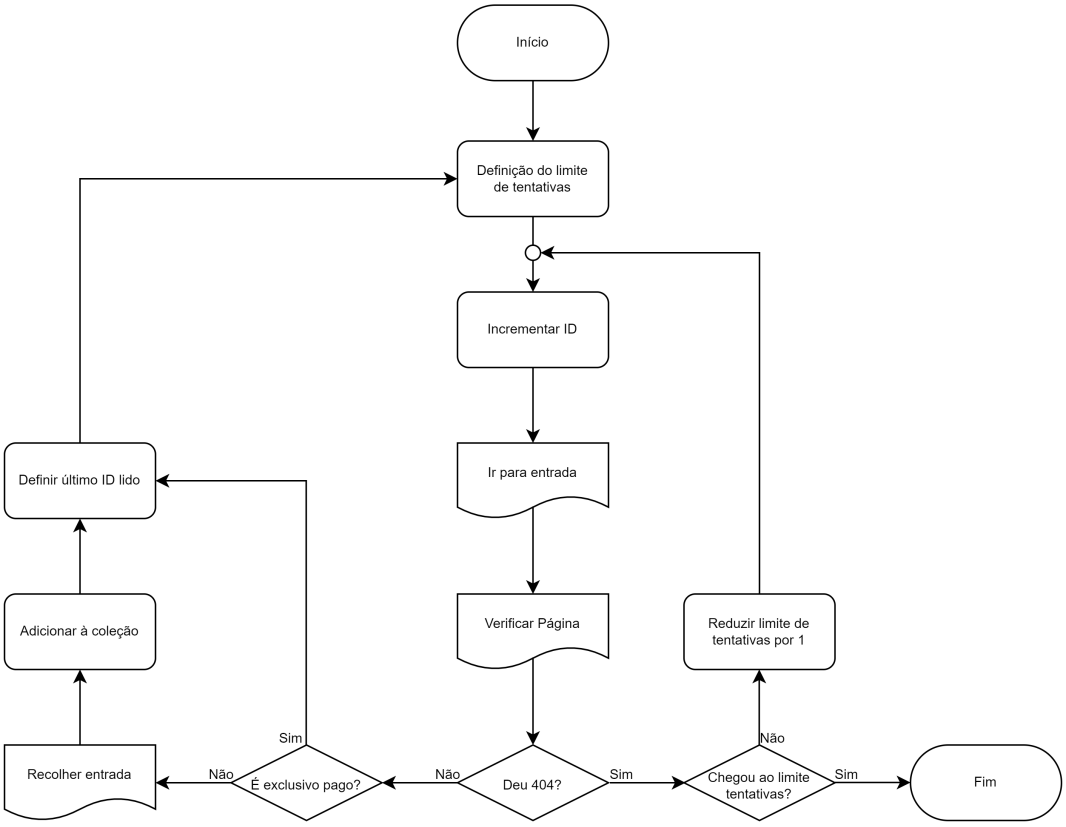

FIGURA 4.14: Esquema da recolha de informação relativa à fonte de informação 0day.today

O módulo de recolha de informação da fonte de informação 0day.today é similar (ver Figura 4.14), mas difere num único passo: Em vez de verificar se a página retornou uma mensagem de 404, esta também verifica se a fonte devolveu uma entrada que é exclusiva a utilizadores pagos. Se for o caso, esta assinala que a entrada não pode ser lida, e prossegue a sua execução. Se no futuro se constatar que é viável o pagamento por informação sobre exploits de esta fonte, este passo pode ser alterado ou mesmo removido, para que esta situação esteja contemplada.

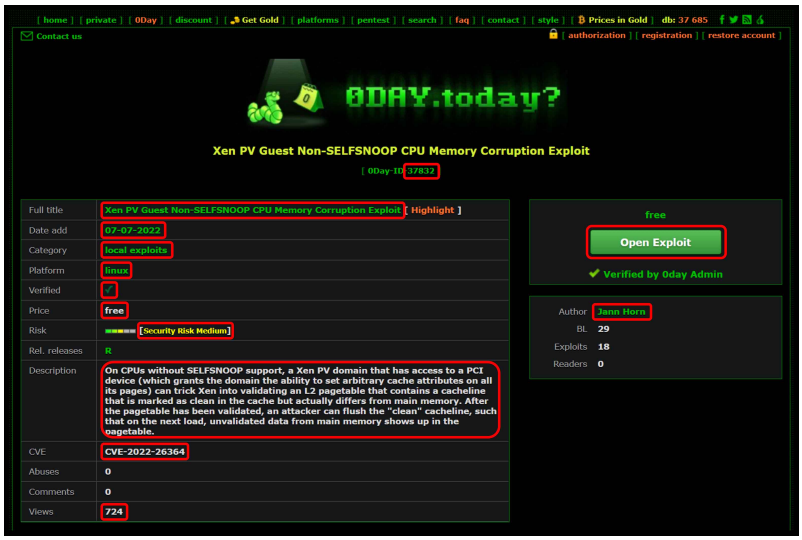

FIGURA 4.15: Algumas componentes de cada entrada do 0day.today que são recolhidas, com um quadrado vermelho à sua volta.

Em todas as páginas do 0day.today existe uma tabela com duas colunas, que contém o título do campo à esquerda e o seu conteúdo à direita (ver Figura 4.15). Nem sempre são dispostas todas estas informações, pelo que quando não existem, a linha desse campo é omitido da tabela. Nas entradas que não disponibilizam o texto do exploit gratuitamente, o campo presente no lado direito da página não disponibiliza o seu conteúdo, mas sim uma página que pede um pagamento para a poder visualizar.

Tal como no módulo de recolha de informação do ExploitDB, dentro do passo Recolher Entrada obtêm-se os seguintes campos:

- **Título**, **ID**, **autor**, **tipo de exploit**, **plataforma**, **categoria**, **data de publicação** e **preço** do exploit, que estão sempre presentes em todas as entradas da fonte de informação.
- **Estado da verificação** do exploit, ou seja, se a fonte verificou a informação. Neste caso, só as entradas muito antigas é que não estão verificadas.
- **ID do CVE** da vulnerabilidade que o exploit explora, se disponível.
- **Descrição** do exploit, se disponível.
- **Número de visualizações** à data da recolha, bem como o **nível de risco atribuído pela fonte** ao exploit.
- **Link de referência**, que corresponde à página onde se obteve a informação, e **link para o exploit**, onde podemos ver o código fonte do exploit.
- Tal como no módulo de recolha do ExploitDB, é temporariamente recolhido o texto do exploit, de forma a criar uma **hash de comparação**.

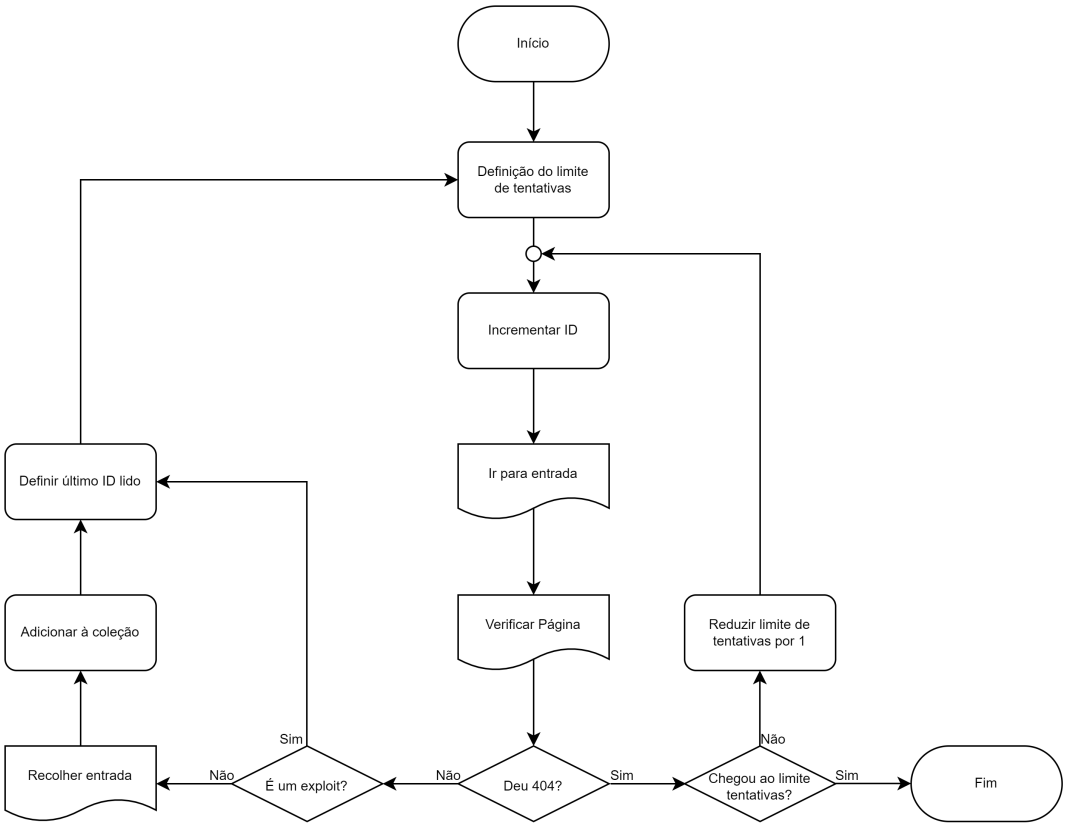

FIGURA 4.16: Esquema da recolha de informação relativa à fonte de informação PacketStorm Security

Por último, o módulo de recolha de informação da fonte PacketStorm Security também é similar aos anteriores, embora tenha uma pequena modificação necessária à recolha de informação exclusivamente de exploits. Como a fonte publica várias entradas com diversos tipos de informação, e a sua enumeração é feita de uma forma global, não existe um identificador para apenas os exploits publicados por esta, sendo partilhado com o resto das publicações. Assim, é necessário efetuar uma verificação da categoria da publicação antes de a recolher, para ter a certeza que está a ser recolhida informações sobre um exploit e não de uma ferramenta, por exemplo.

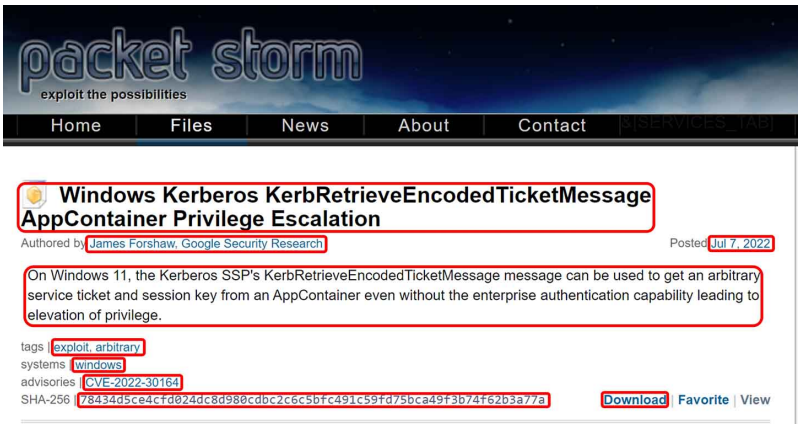

FIGURA 4.17: Algumas componentes de cada entrada do PacketStorm Security que são recolhidas, com um quadrado vermelho à sua volta.

Assim como nos módulos de recolha de informação apresentados anteriormente, no PacketStorm Security obtêm-se campos semelhantes.

Depois de realizar estas etapas, o módulo de recolha guarda a coleção num ficheiro *csv*, e de seguida envia-o por um canal seguro para o motor de processamento, de acordo com as informações definidas no processo.

Para o envio da informação foram consideradas as seguintes alternativas:

- **Criar uma API que sirva de *apartado*** - Foi considerada a criação de uma API no sistema, para que esta recebesse os ficheiros provenientes da recolha de informação. Esta opção foi descartada por duas razões: (1) Poderia originar erros de implementação que poriam em causa a segurança do sistema; (2) Aumentaria o esforço para a manutenção futura do sistema, pois não estaria a ser utilizado um software estandardizado.

- **Não enviar o ficheiro, e ir buscá-lo ativamente** - À primeira vista, esta alternativa seria a mais óbvia, no entanto, seria preciso adicionar dependências desnecessárias ao sistema (ao nível da interação com outros sistemas de ficheiros em rede).

Assim, foi escolhido o método de enviar o ficheiro por ssh. O acesso deste deve ser limitado para apenas permitir a introdução de ficheiros, para evitar abusos que podem originar através do roubo do acesso. O processo também deve ser compartimentado, de forma a que caso exista um roubo do acesso, este tenha um menor impacto no sistema, e afete só o processo ZeroDays.

4.2.3.2 Triagem e interpretação da informação de exploits

Como o processo de recolha de informação está calendarizado para ocorrer a cada 8 horas, seria possível definir o processo de triagem e interpretação da informação para executar simultaneamente. Este foi o procedimento adotado no ZeroDays v1, e este criou problemas devido à produção dos dados por vezes não ser realizada na janela de tempo definida (ver Figura 4.9). Assim, foi definido que esta estratégia seria de evitar, para que o sistema consiga ser mais resiliente a este tipo de atrasos.

Deste modo, foi necessário encontrar um método de detetar a existência de nova informação disponível dentro dos ficheiros *csv*. Foi considerada a hipótese de realizar o tratamento do ficheiro com base na sua data de modificação, mas esta foi descartada pois não assegura que o ficheiro contém informação diferente da que foi recolhida anteriormente. Foi também considerado calcular um hash do ficheiro e guardar este num ficheiro à parte. Este método permitiria verificar se existem modificações, mas não deteta se foi produzida informação já recolhida anteriormente (apesar de isto não ser provável devido à arquitetura dos módulos de recolha, pode ocorrer se houver um problema com os ficheiros log).

Assim foi decidido que, tendo acesso aos identificadores das entradas produzidas pelas fontes, e como estes são sequenciais, será guardado o maior identificador presente no ficheiro recolhido num ficheiro à parte, para que seja possível ao processo verificar se existe nova informação dentro do ficheiro recolhido. Quando o processo inicia, este verifica se para cada um dos ficheiros *csv* existe um ficheiro ".lastID"correspondente. Caso o ficheiro exista, o processo verifica se o identificador guardado é inferior ao maior identificador recolhido. Se for este o caso, ou se o ficheiro não existir, a informação é recolhida normalmente, e o ficheiro é atualizado quando o processo termina. Caso exista um ficheiro ".lastID", e este não for inferior ao maior identificador presente no ficheiro *csv*, o processo ignora o ficheiro recolhido.

Com este procedimento é possível evitar possíveis incompatibilidades de calendarização, pois o processo de tratamento de informação pode executar com uma maior frequência que o processo de recolha, evitando-se assim esperas ativas. Apesar disto, este procedimento apresenta algumas contrapartidas:

- Se for introduzida uma nova fonte que não usa um sistema com base em identificadores, ou se alguma fonte já existente deixar de o usar, terá de ser criada uma exceção para essa. Nesse caso, teria de ser guardada um hash do conteúdo do ficheiro *csv*, ou caso o sistema operativo e o sistema de ficheiros o permita, poderá ser utilizada a data de modificação do ficheiro.
- Ao criar um ficheiro para cada fonte de informação, embora este possua pouca informação, este irá tendencialmente ocupar o tamanho mínimo definido para um ficheiro, que irá ser o tamanho de um cluster do sistema. Isto pode ser corrigido ao juntar a informação toda num único ficheiro, no entanto, pode impactar a clareza do processo.

De seguida, depois de aferir se há nova informação, o processo envia o ficheiro para um interpretador, que irá ler o exploit e adicioná-lo a um vetor já com a sua informação uniformizada. Este passo pode ser paralelizado para aumentar o desempenho do processo. No entanto, optou-se por não realizar tal optimização, devido ao reduzido número de fontes atualmente implementadas, e à relativa rapidez com que a parte de interpretação dos resultados executa.

4.2.3.3 Cruzamento da informação de exploits

Depois de uniformizada, a informação é cruzada com as ferramentas de deteção de ameaças internas, para aferir se o exploit recolhido é de facto dia-zero. O cruzamento da informação é feito de acordo com o diagrama exposto na Figura 4.18.

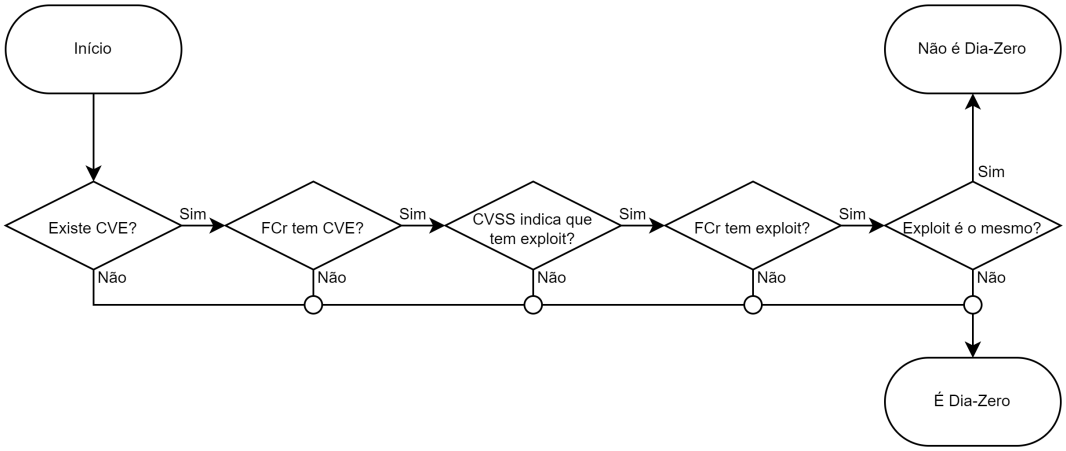

Figura 4.18: Procedimento do cruzamento da informação com as Fontes de Cruzamento (FCr)

Para conseguir realizar o cruzamento, é necessário obter informação de identificação da ameaça. Tendo em consideração os identificadores usados nas fontes internas, os únicos identificadores usáveis para o *exploit* são os identificadores CVE das vulnerabilidades que este explora. Sem este não é possível assegurar o seu conhecimento interno.

Assim, foram definidos os seguintes critérios para aferir se um exploit é dia-zero:

- Se não existir um ID do CVE que identifique a vulnerabilidade explorada.
- Se a fonte de cruzamento não possuir informação sobre o CVE explorado pelo exploit.
- Se o CVSS associado ao CVE não indicar que existe um exploit.
- Se a fonte de cruzamento não possuir nenhum exploit para esse CVE.
- Se a lista de exploits indicada pela fonte de cruzamento não tiver uma entrada cujo Hash de Comparação seja igual ao que foi recolhido.

O exploit é dia-zero se cumprir todos os critérios acima. Adicionalmente, se o exploit já tiver sido recolhido numa iteração anterior do processo ZeroDays v2, o exploit é assumido como não sendo dia-zero.

No final, a informação é inserida num índice do elasticsearch, para que esta informação seja armazenada para referência futura, bem como visualizada.

4.2.3.4 Confrontação da informação de exploits

Depois de realizado o cruzamento da informação, a informação é confrontada com o cadastro interno de assets da empresa. Como as categorias atribuídas nas fontes de informação aos exploits são muito genéricas, optou-se por efetuar uma atribuição de categorias a partir de palavras-chave presentes no título de cada exploit. Esta função é necessária, pois permite uma aferir o que está vulnerável a uma determinada ameaça com uma maior certeza. No entanto, esta função também possui algumas limitações, como a possibilidade da existência de variações de nomes e erros ortográficos, que, a menos que sejam especificadas previamente, não serão detetadas pelo processo.

Assim, optou-se por organizar as palavras-chave por ordem de popularidade. Isto permite que uma ameaça para, por exemplo, o software *Microsoft SQL Server* seja detetada como tal, em vez de ser determinada como *SQL*, um termo muito mais comum.

Depois de atribuída uma categoria ao exploit, a informação é cruzada com o cadastro interno da empresa, podendo assim definir que sistemas estão potencialmente vulneráveis a essa ameaça. A informação é então inserida num índice, cujas entradas contêm o asset vulnerável e a ameaça a que está exposto.

4.2.3.5 Notificação de exploits

Depois de terminada a confrontação dos exploits, a informação que é dia-zero e que poderá ter impacto em sistemas vulneráveis é recolhida e agrupada para ser inserida num ficheiro PDF. Esse ficheiro é depois enviado para uma lista de email pré-definida, em que dada a sensibilidade da informação se optou por enviar apenas para quem precisa desta informação, neste caso, a equipa de CyberSOC da Direção de Cibersegurança.

4.2.4 Software malicioso

Após implementada a componente de exploits, iniciou-se a implementação da componente de software malicioso. Tal como na componente de exploits, foi idealizado que a recolha de informação iria ocorrer periodicamente ao longo do dia. No entanto, por motivos relacionados com a produção de informação por parte das fontes, foi definido que a recolha de informação iria ser executada mais frequentemente, a cada meia hora, para um total de 48 vezes ao dia.

4.2.4.1 Recolha de informação de software malicioso

A componente de recolha de informação da componente de software malicioso seguiu a arquitetura descrita na Secção 4.1.2.1, sendo composta por dois módulos de recolha de informação.

Como as fontes de informação selecionadas (InquestLabs e MalwareBazaar) possuem uma API que torna possível a recolha de dados de uma forma programática, esta foi utilizada. Deste modo, ambos os módulos de recolha de informação tornam-se mais simples, facilitando a resolução de possíveis erros, e não tendo de interagir com browsers para a recolha de informação.

A recolha de informação de software malicioso não sofre do mesmo problema de diferenciação entre entradas que a recolha de informação de exploits - as fontes de informação de software malicioso dispõem todas de um *hash* do ficheiro, permitindo assim identificar de forma unívoca uma ameaça sem que seja preciso a criação de um método especial para as comparar.

Tendo isto em consideração, foram criados os dois módulos de recolha de informação para as duas fontes usadas no projeto ZeroDays v2 - a InquestLabs e a MalwareBazaar.

Figura 4.19: Esquema de recolha de informação das fontes de informação InquestLabs e MalwareBazaar

Na Figura 4.19 está ilustrado o processo de recolha de informação proveniente da InquestLabs. Este obtém as últimas entradas, compara as entradas recolhidas com as que tem guardadas no ficheiro de log, e exclui as que não são novas. Esta utilização do ficheiro log difere da que foi implementada nos módulos criados para processar

os exploits, pois os softwares maliciosos não possuem identificadores sequenciais nas fontes de software malicioso. Logo, não é possível definir uma última entrada onde o processo determina que já leu essa informação.

Devido à produção de informação pelas fontes ser feita de uma forma temporal (isto é, a API da InquestLabs retorna as últimas 5000 entradas, e a API do MalwareBazaar retorna as entradas publicadas na última hora), é possível guardar num ficheiro os hashes dos dados que foram recolhidos na última iteração, e verificar quais entradas são novas. Tal torna-se possível, dado que uma fonte de informação não irá omitir dados numa iteração e voltar a relatá-los na iteração seguinte, pois quando é removida esta é substituida por uma nova entrada.

Apesar de parecer simples, foi necessário experimentar algumas optimizações porque algumas funções do Blue Prism têm um desempenho mais baixo, tendo este um procedimento por detrás mais complexo do que inicialmente aparenta.

Esta optimização deseja melhorar a velocidade com que o Blue Prism itera coleções - inicialmente esta função era muito lenta. Nas recolhas de informação, antes de realizadas as melhorias, o processo demorava cerca de 20 minutos a comparar os dados armazenados com as informações novas. Isto era agravado pela quantidade de iterações que era preciso fazer sobre a informação, como também era o caso de converter a informação em formato *json* (formato multi-dimensional, ou seja, permite a existência de vetores dentro de uma entrada) para *csv* (uni-dimensional). Como esta lentidão influencia todos os outros passos do processo, era preciso repensar a operação.

Para converter o formato *json* para *csv*, foi criada uma função em Visual Basic que realiza operações de *regex* para obter os dados do ficheiro *json*, e dispô-los de uma forma uni-dimensional. Esta função foi necessária, pois quando se convertia a informação em *json* para *csv* a partir das funções disponibilizadas pelo Blue Prism, perdiam-se dados em campos que tivessem vetores (neste caso, para a InquestLabs, perdia-se dados sobre informações adicionais que a fonte dispunha em cada entrada).

Como as operações realizadas em blocos de código em Visual Basic são muito mais rápidas que as outras operações, foi desenvolvida uma função que compara as coleções dos dados recolhidos com os dados do ficheiro log, e retorna as que são novas. O procedimento é o seguinte:

1. Inicialmente, são retirados todos os dados da coleção acabada de recolher (com exceção do hash do software) e são guardados numa nova coleção, que será denominada como *Coleção A*.

2. Depois são agregados os dados da recolha anterior e os da *Coleção A* numa só, e ordenados alfabeticamente. Deste modo, todas as entradas repetidas aparecem juntas na sequência.

3. De seguida, são retiradas todas as entradas que estão presentes na lista mais que uma vez e guardados numa coleção nova, que será denominada como *Coleção B*.

4. Após a criação da *Coleção B*, esta é somada à coleção dos dados da recolha anterior e são ordenadas alfabeticamente, para que tal como no Passo 2 as entradas repetidas apareçam agrupadas.

5. Depois, são excluídas todas as entradas que não estão presentes na lista mais que uma vez e guardados numa coleção nova, que será denominada como *Coleção C*, que contém os sha256 das entradas que são novas e que não foram recolhidas anteriormente.

6. De seguida, as entradas da *Coleção C* são somadas à coleção acabada de recolher, e são excluídas todas as entradas que não tenham duas ocorrências do mesmo sha256. Estas entradas são guardadas numa coleção nova, que será denominada como *Coleção D*.

Depois dos passos descritos acima, é possível obter todos os dados novos numa única coleção, a *Coleção D*, bem como a coleção que será guardada como o registo da última iteração que será guardado no ficheiro log, a *Coleção A*. Este procedimento está também ilustrado de uma maneira simplificada na Figura 4.20.

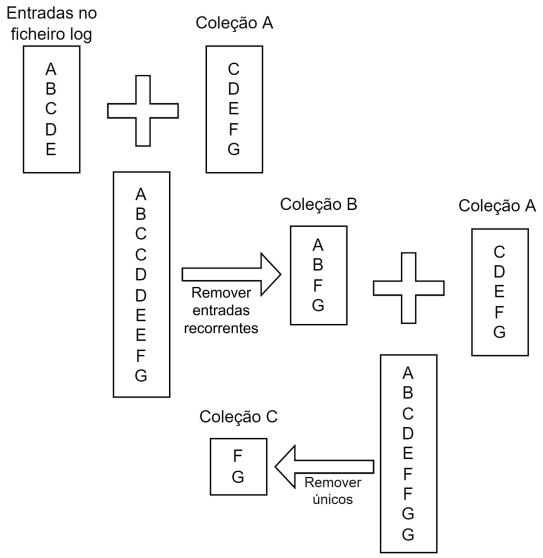

Figura 4.20: Procedimento de exclusão de informação já conhecida no processo de recolha de informação da InquestLabs

Com recurso a estes dois procedimentos, foi possível diminuir o tempo de recolha de informação de aproximadamente 40 minutos para cerca de 20 segundos.

Comparativamente, o módulo de recolha de informação do MalwareBazaar é muito mais simples. Este também dispõe do mesmo método de guardar as entradas no ficheiro log, mas numa escala mais reduzida - devido aos *endpoints* que a API do MalwareBazaar disponibiliza, apenas é possível a obtenção dos dados da última hora, ou das últimas 100 entradas. Como não se consegue determinar através da API se já foram produzidas 100 novas entradas, foi definido que serão obtidos os dados da última hora, e será executada a recolha de informação de meia em meia hora. Assim, caso uma iteração falhe, se a seguinte não falhar, não serão perdidos dados. Caso se verifique um elevado número de falhas nas recolhas de informação, esta calendarização poderá ser recalibrada, de modo a que seja tolerante a mais faltas.

Assim, o módulo de recolha acaba por ser semelhante ao da Figura 4.19. Como a quantidade de informação recolhida é substancialmente menor, não foi necessário criar os procedimentos para a conversão e identificação de dados úteis. Se no futuro se verificar um aumento na produção de dados do MalwareBazaar que justifique a implementação destes procedimentos, tal poderá ser realizado facilmente através da adaptação dos já existentes.

4.2.4.2 Triagem e interpretação da informação de software malicioso

A triagem e interpretação da informação de software malicioso foi implementada da mesma maneira que na componente de exploits, de modo a manter a simplicidade,

modularidade, e integridade entre os componentes do sistema.

Uma vez que existe uma quantidade elevada de informação gerada pelas fontes, optou-se por considerar apenas as entradas presentes em ambas as fontes. Esta escolha permite que a informação relatada tenha uma maior fiabilidade, embora possa reduzir a sensibilidade do sistema a novas informações sobre software malicioso.

A informação é inicialmente importada para um índice no *elasticsearch*, e de seguida são verificadas quais das entradas recolhidas são duplicadas (ou seja, quais destas entradas estão presentes no índice mais que uma vez provenientes das duas fontes de informação). Desta maneira, num passo simples, é possível identificar as entradas que foram simultaneamente pelas duas fontes, bem como entradas que foram recolhidas e que já tinham sido relatadas numa iteração anterior.

4.2.4.3 Cruzamento da informação de software malicioso

Depois de filtrada, a informação é cruzada com as duas fontes de cruzamento internas: a VirusTotal, e a IBM X-Force Exchange. Para a pesquisa de informação nestas fontes, foi utilizado o campo com o hash do software, pois ambas as fontes indexam as ameaças de forma a permitir a pesquisa por este campo.

Assim, como o objetivo é definir o conhecimento interno de uma determinada ameaça, se alguma das fontes de informação indicar que a ameaça já foi identificada, então esta não será dia-zero. Se ambas indicarem que a ameaça é desconhecida, então esta será considerada como dia-zero.

4.2.4.4 Confrontação da informação de software malicioso

Infelizmente, devido à falta de informação das fontes que permita definir com certeza que sistemas computacionais estão vulneráveis a que ameaças, não foi possível criar um método que realize a confrontação interna com um alto nível de confiança.

4.2.4.5 Notificação de software malicioso

Depois do cruzamento da informação de software malicioso, a informação sobre ameaças que foram consideradas como dia-zero é enviada por email para a equipa do CyberSOC da direção de cibersegurança da organização.

Capítulo 5

Resultados e avaliação

Para poder analisar o desempenho do sistema ZeroDays v2, as suas componentes foram executadas durante um mês (componente de exploits) e uma semana (componente de software malicioso). Idealmente, os dados seriam recolhidos durante mais tempo, mas tal não foi feito por restrições de tempo.

Como tal, a avaliação da componente de exploits será feita na Secção 5.1, e a avaliação da componente de software malicioso será realizada na Secção 5.2. Estas secções são compostas pela avaliação da sua recolha de informação, das suas fontes de informação, pela avaliação do desempenho dos seus motores de processamento, e também por uma avaliação geral da componente num todo.

5.1 Componente de exploits

Para avaliar a componente de exploits, esta foi executada durante um mês (entre 25 de julho e 25 de agosto), e foram recolhidos os dados produzidos para que possam ser analisados.

5.1.1 Resultados obtidos da recolha de exploits

Nesta subsecção pretende-se avaliar os resultados produzidos pelo processo de recolha de informação de exploits.

Como representado na Figura 5.1, foram recolhidas 187 entradas das três fontes de informação. Dentro destas 187 entradas, não foi possível obter o código de 3, sendo que só foi possível calcular o hash de comparação de 184 entradas. Destas, o processo classificou 120 como sendo únicas, tendo 75 entradas que foram relatadas por apenas uma das fontes, 29 entradas que foram relatadas por duas fontes, e 17 entradas que se encontraram nas três.

Após uma inspeção manual, foram verificadas 98 entradas únicas, tendo assim o sistema falhado na criação do hash de comparação em 22 ocasiões, tendo assim uma margem de erro de aproximadamente 12%.

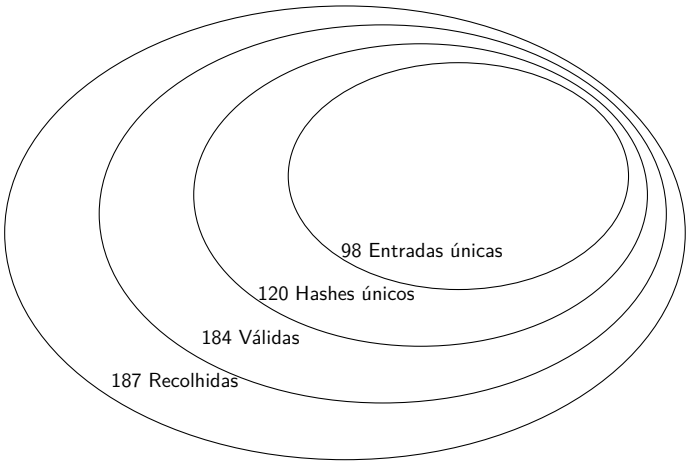

FIGURA 5.1: Entradas recolhidas avaliadas quanto à performance do Hash de comparação.

Dentro das 22 entradas cujo hash de comparação diferiu das publicações noutras fontes de informação, 10 originaram de um cabeçalho ou informação de rodapé inesperada no texto do exploit, que foi removido por uma das fontes de informação. Seis entradas não assinaladas como repetidas tiveram como causa entradas que foram truncadas na fonte de informação, que contém cerca de metade do seu conteúdo. A fonte ZDT foi quem realizou todos estes cortes de informação, tendo assim entradas incompletas. Três das entradas repetidas que não foram apontadas como tal correspondem a erros de implementação, que irão ser corrigidos, pois o algoritmo utilizado para criar hashes de comparação foi desenhado de forma a apanhar as diferenças presentes entre estas entradas. As últimas duas entradas repetidas tiveram origem na maneira como a informação foi exposta: numa delas, dentro do corpo do exploit, foi utilizada codificação de URL, enquanto que na sua contraparte não foi; na outra entrada repetida, quem realizou a introdução dos dados na fonte de informação dividiu o seu conteúdo em duas entradas, enquanto que as outras fontes publicaram tudo na mesma entrada.

Em termos de frequência de publicação, observou-se em média aproximadamente 4 entradas únicas por dia, com um pico máximo de 38 entradas num dia que correspondeu também ao valor máximo de entradas únicas (27). Verificou-se uma maior frequência nos dias de semana, existindo um maior número de resultados às terças feiras (ver Figura 5.2). Se for excluído o valor extremo referido anteriormente, a frequência de entradas à segunda-feira baixa drasticamente, tornando-se o segundo dia com menor frequência de entradas.

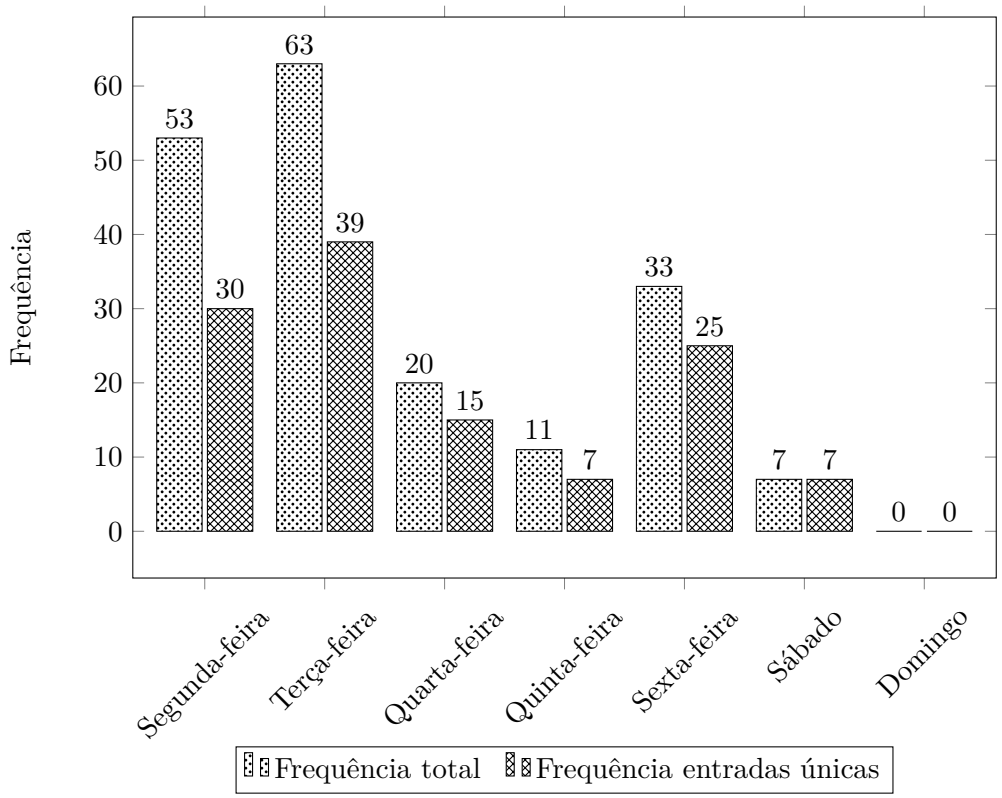

FIGURA 5.2: Comparação do número total de entradas de exploits recolhidas conforme o dia da semana.

Como não são possíveis de obter as horas exatas de publicação nas fontes de exploits, é impossível afirmar ao certo quanto tempo o processo demorou até detetar uma nova ameaça. Como tal, o incremento mais pequeno que é possível analisar é de um dia.

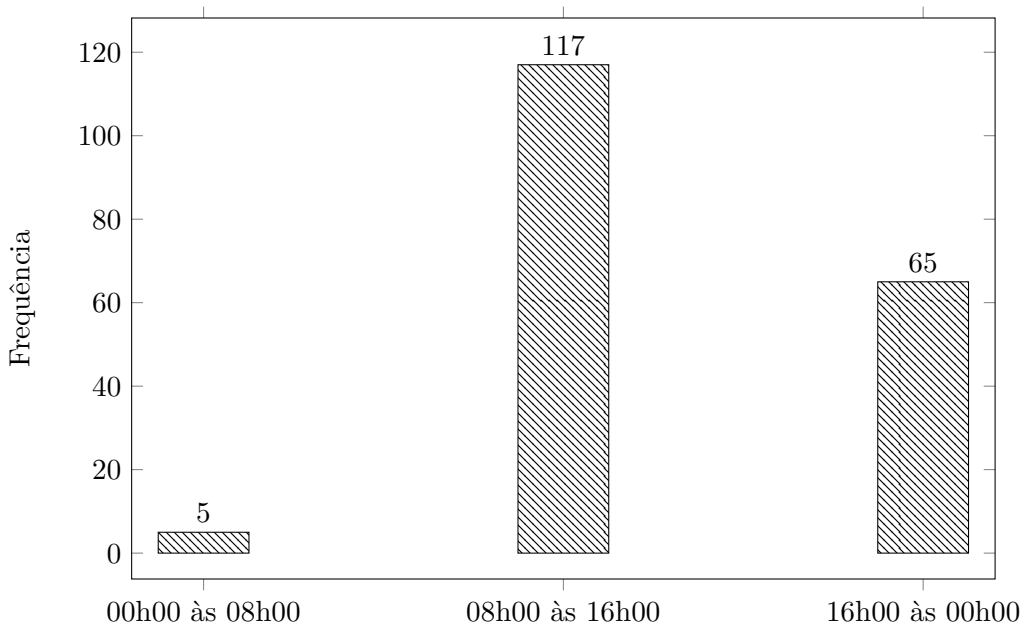

FIGURA 5.3: Distribuição da quantidade de entradas recolhidas ao longo das horas do dia.

Quanto à distribuição ao longo das três iterações diárias, verificou-se que existem poucos dados produzidos entre a meia noite e as oito da manhã, com apenas 5 entradas produzidas neste horário. No entanto, entre as oito da manhã e as dezasseis horas foram produzidas 117 entradas, cerca de 63% do que foi recolhido, sendo que as restantes 65 entradas foram recolhidas entre as dezasseis horas e a meia noite. Com estes dados podemos concluir que deve ser feita uma modificação no horário de funcionamento do sistema, de forma a executar mais frequentemente num certo espaço horário (de acordo com os resultados em análise, deve ser aumentada a frequência de recolha de informação entre as 8 e as 16 horas). Esta modificação permitiria estar mais atualizado, mas não correndo o risco de estar a executar a recolha de informação sem haver informação nova.

5.1.2 Avaliação das fontes de informação de exploits

Para avaliar a qualidade de cada uma das fontes de informação, foi efetuada uma verificação da unicidade das entradas. Os resultados encontram-se representados na Figura 5.4.

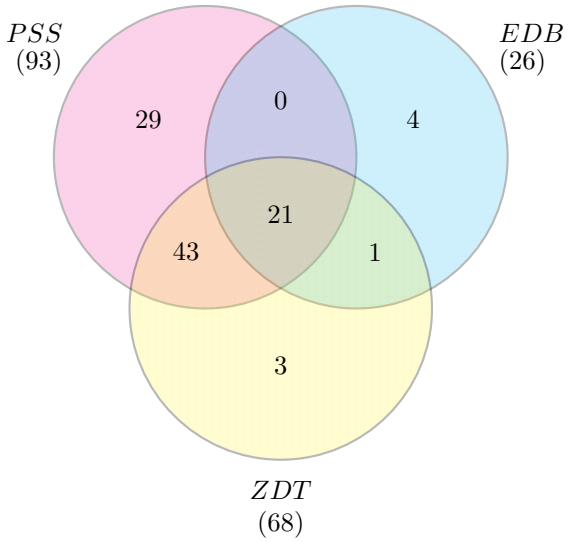

Figura 5.4: Coincidência das entradas nas fontes de informação utilizadas na implementação do motor de recolha de exploits.

Classificando as fontes quanto à sua unicidade, a fonte PSS apresenta o maior número de entradas únicas, seguido a EDB e da ZDT, com um número de entradas únicas muito similar. A ZDT possui um número elevado de exploits coincidentes com outras fontes de informação, constatando-se que partilha grande parte da sua informação com a PSS.

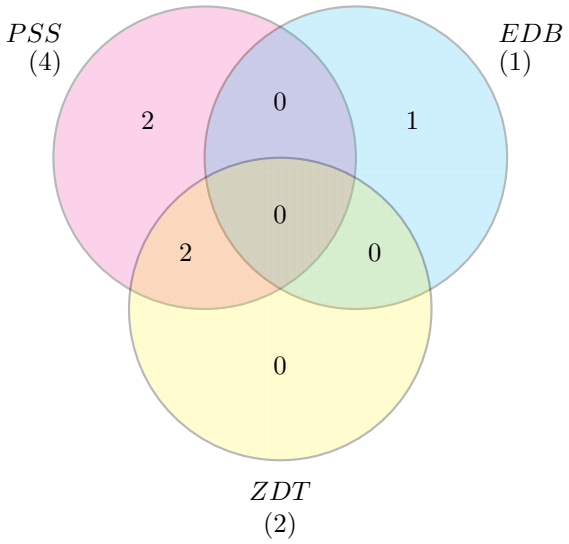

Figura 5.5: Alertas gerados por cada uma das fontes de informação de exploits.

De seguida, foram analisados os alertas gerados pelo sistema. Estes alertas correspondem a ameaças dia-zero que foram relatadas, por representarem uma ameaça a sistemas computacionais da empresa. Olhando para os cinco alertas que foram gerados (ver Figura 5.5), apenas um não teve origem numa entrada presente na PSS, sendo exclusivo à EDB. Nos quatro alertas gerados a partir de informação da fonte PSS, dois foram também fornecidos ZDT. No entanto, como o número de alertas é baixo, afirmar que uma fonte é mais útil que as restantes pode ser perigoso, pelo que deixamos esta questão para o futuro.

Comparando o tempo de resposta (ou seja, o tempo que cada fonte demora a publicar informação que foi disponibilizada noutra fonte) de cada fonte nas entradas duplicadas, podemos verificar que:

- A fonte ZDT sofre constantes atrasos - nas 65 publicações comuns, esta fonte foi a última a publicar a informação em 31 dos casos. O atraso mínimo observado foi de um dia, o máximo de três e o atraso médio de aproximadamente dois dias. A ZDT apenas publicou uma entrada mais rápido que qualquer outra fonte (a EDB), e esta foi produzida ao mesmo tempo que a PSS. O tempo de resposta médio da ZDT é 0.89 dias depois das outras fontes.

- A PSS consistentemente realizou as publicações ao mesmo tempo que a EDB - nas 21 entradas que têm em comum, a EDB apenas se atrasou uma única vez (o caso descrito anteriormente). Este atraso foi de um dia, o valor mínimo possível de verificar.

- Na única ocorrência em que houve uma entrada exclusivamente comum entre a EDB e a ZDT, a última sofreu um atraso de dois dias face a primeira.

Estes resultados fornecem uma indicação que apesar de as três fontes serem importantes (dado que todas possuem entradas únicas), não têm todas a mesma relevância, sendo que a melhor fonte é claramente a PSS, devido à quantidade de informações únicas que esta publica. Em segundo lugar, a EDB é uma fonte muito útil, porque apresenta as entradas sem sofrer grandes atrasos. Por último, a ZDT, que apesar de apresentar atrasos na divulgação dos *exploits* e de por vezes possuir informação incompleta, também publica entradas únicas e tem uma grande compatibilidade com

as outras fontes de informação. Desta maneira é possível utilizá-la para enriquecer a informação já disponível.

5.1.3 Avaliação do motor de processamento de exploits

O motor de processamento de exploits teve um comportamento positivo, não tendo apresentado falsos cruzamentos. Isto na prática significa que tudo o que foi apontado como sendo uma ameaça dia-zero se confirmou como tal (com a exceção das ameaças referidas anteriormente que não foram classificadas corretamente como duplicadas).

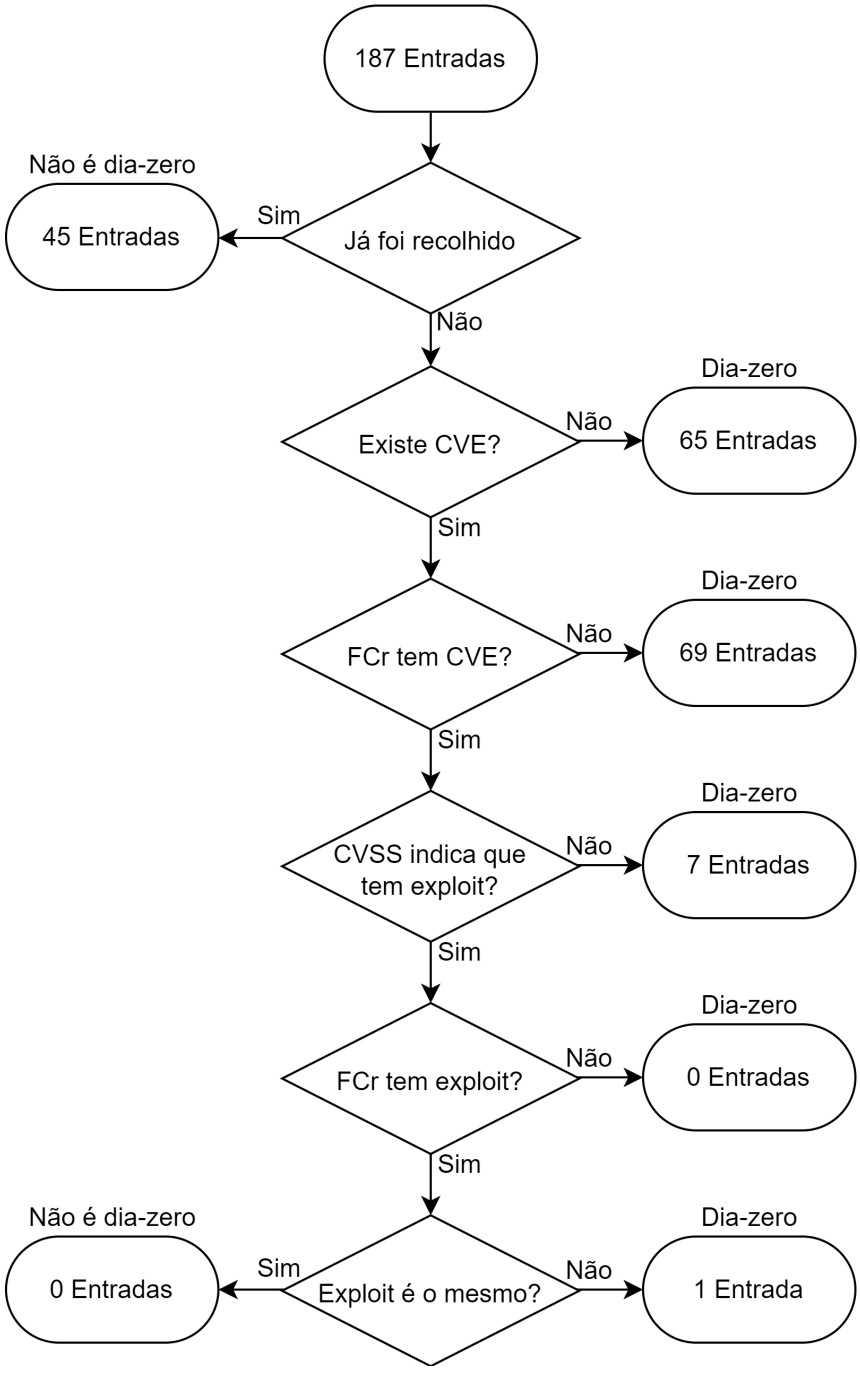

Figura 5.6: Comportamento da verificação com a fonte de cruzamento.

Como ilustrado na Figura 5.6, nenhuma dos 187 *exploits* foram classificados como conhecidos internamente. Isto deve-se principalmente a dois fatores: a falta de informação sobre a vulnerabilidade que um exploit explora, e à baixa quantidade de vulnerabilidades que a fonte de cruzamento dispõe.

Em relação à confrontação com os assets da empresa, o processo apresentou resultados não tão satisfatórios - a única palavra chave que foi cruzada com sucesso foi *windows*, devido à inexistência de um cadastro estandardizado com o software que os assets utilizam. Por causa desta limitação, foi preciso utilizar um sistema de pesquisa por palavras chave, sendo que nenhuma das notificações geradas adveio de confrontações com assets, mas sim de palavras chave criadas com o objetivo de alertar caso um exploit possa ter interesse. Este filtro não captou entradas que foram manualmente classificadas como irrelevantes, mas também não captou algumas entradas que foram classificadas manualmente como úteis, necessitando de ser melhorado no futuro. No total foram criados 5 alertas únicos com base nas palavras chave, enquanto que manualmente se verificaram 7 entradas únicas com informação relevante à Direção de Cibersegurança.

5.1.3.1 Revisão de resultados da componente de exploits

Passado um mês da experiência, foram revistos os resultados, e constatou-se que dentro das 51 entradas que foram verificadas manualmente como sendo únicas e que contêm CVE-ID (porque as que não contêm este identificador não são possíveis de identificar na fonte de cruzamento, sendo sempre dia-zero), 7 destas sofreram uma atualização de estado na fonte de cruzamento, e 70 mantiveram o seu nível de conhecimento interno (excluindo o conhecimento definido pelo próprio sistema). Dentro destas 7, só uma não reconhece a existência de um exploit, sendo que todas as outras consideram que existe um exploit. Nessas seis, duas reconhecem o exploit reconhecido, deixando assim de constituir ameaças dia-zero.

Nas duas entradas que reconhecem o exploit que foi recolhido, verificaram-se atrasos do lado da fonte de cruzamento de 26 dias e 14 dias. Isto significa que, se o sistema ZeroDays v2 não existisse, a informação demoraria em média mais 20 dias a chegar à equipa do CyberSOC da Direção de Segurança da empresa.

5.1.4 Avaliação geral da componente de exploits

Com vista o objetivo de identificar ameaças antes de estas se consumarem e de recolher informação sobre estas para informar a empresa de ameaças previamente desconhecidas, o sistema ZeroDays v2 apresentou resultados bastante positivos.

		Verificação Manual	
		Verdadeiro	Falso
Sistema	Positivo	7	0
	Negativo	179	1

Tabela 5.1: Matriz dos alertas gerados pela componente de exploits.

Como é possível verificar na Tabela 5.1, onde se encontram dispostas as entradas consoante o estado de alerta, o processo apresentou uma precisão e exatidão de 100% e uma sensibilidade de 87,5%. A origem deste resultado reside na fonte de confrontação, que não contém os dados necessários para efetuar uma confrontação perfeita das informações. A combinação destes valores resulta num F-Score de 93,3%. É de

destacar que estes resultados são obtidos com uma amostra reduzida de entradas. Existiu apenas um falso positivo, correspondente a uma entrada única (ou seja, apenas ocorreu uma vez, incluindo repetições - ver Apêndice C). As fórmulas utilizadas para calcular estas métricas encontram-se dispostas no apêndice B.

Depois de obtidos estes resultados, e verificado que o sistema conseguiu antecipar algumas ameaças, é possível afirmar que a componente de exploits teve sucesso no seu objetivo de alertar a equipa do CyberSOC de exploits dia-zero, de forma a fornecer mais tempo necessário para entender e lidar com essas ameaças.

Após a análise dos resultados, foi também descoberta uma irregularidade no processo de recolha que impedia algumas iterações de executar corretamente. Depois de corrigida, o processo de recolha de informação não voltou a apresentar qualquer deficiência. Independentemente deste problema, tal como descrito anteriormente, irá ser modificada a frequência das recolhas de informação para reduzir ainda mais a latência entre a produção de dados das fontes de informação e a importação destes pelo sistema.

5.2 Componente de software malicioso

A componente de software malicioso foi executada durante uma semana (de 21 a 28 de setembro) e foram extraídos os dados recolhidos, para poder avaliar o seu desempenho.

5.2.1 Resultados obtidos da recolha de software malicioso

Durante o espaço de tempo estabelecido, foram recolhidas 8679 entradas no total, sendo que 5606 destas foram provenientes da fonte InquestLabs, e 3073 da fonte MalwareBazaar. Destas, 38 entradas existiram em ambas as fontes, e foi verificado o conhecimento interno destas na empresa.

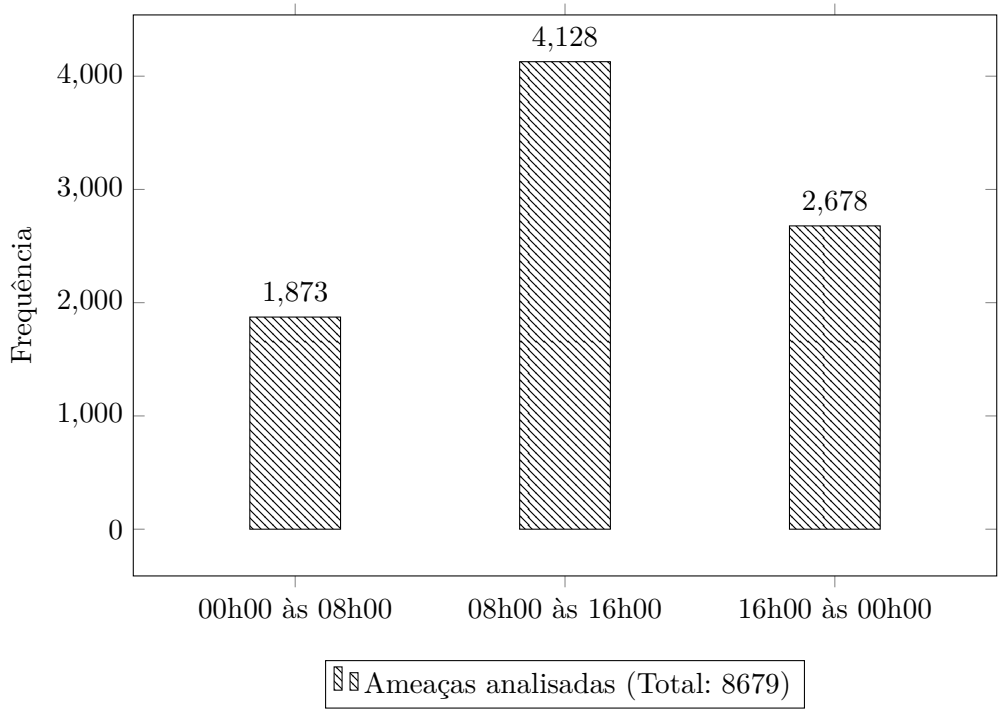

Figura 5.7: Distribuição da quantidade de entradas recolhidas ao longo das horas do dia.

Em termos de frequência de publicação ao longo do dia, verificou-se que há uma maior frequência de publicações entre as 8 e as 16 horas, sendo que neste intervalo de tempo houve mais de o dobro das entradas que ocorreram entre a meia noite e as 8h, constituindo cerca de 47,5% das entradas totais.

Apesar disto, ao observar os dados recolhidos é possível verificar que o espaço horário entre as 8h e as 16h é bastante influenciado por duas ocorrências em que foram recolhidas muitas entradas, face o normal. Se estes dois valores forem removidos, a distribuição assemelha-se a uma distribuição uniforme (27,3% das entradas ocorreram entre a meia noite e as 8h, 33,8% entre as 8 e as 16 horas, e 38,9% entre as 16 horas e a meia noite).

É possível concluir então que, ao contrário da componente de exploits, a frequência da recolha de informação não precisa de uma adaptação especial (por exemplo, o processo não beneficiaria de executar mais vezes entre as 0h e as 8h mais do que executar mais vezes entre as 8h e as 16h). No entanto, devido à quantidade elevada de informação, é sempre preferível que este processo de recolha de informação corra o mais frequentemente possível, de modo a manter o sistema atualizado com novas ameaças que acabam de ser relatadas.

5.2.2 Avaliação das fontes de informação de software malicioso

Ao analisar os dados produzidos pelas fontes de informação, conclui-se que estas são valiosas, embora tenham pouca coincidência. Durante os 7 dias da experiência, verificou-se que a fonte InquestLabs produziu 5606 entradas, e a MalwareBazaar 3073 entradas. A sua distribuição ao longo dos dias da semana foi a que é possível verificar na Figura 5.9, em que a segunda e terça-feira são os dias com maior atividade (cerca de 1900 entradas em ambos os dias), em contraste com domingo, em que se verificou uma baixa frequência de novas publicações (437 entradas).

Figura 5.8: Coincidência das duas fontes de informação de software malicioso.

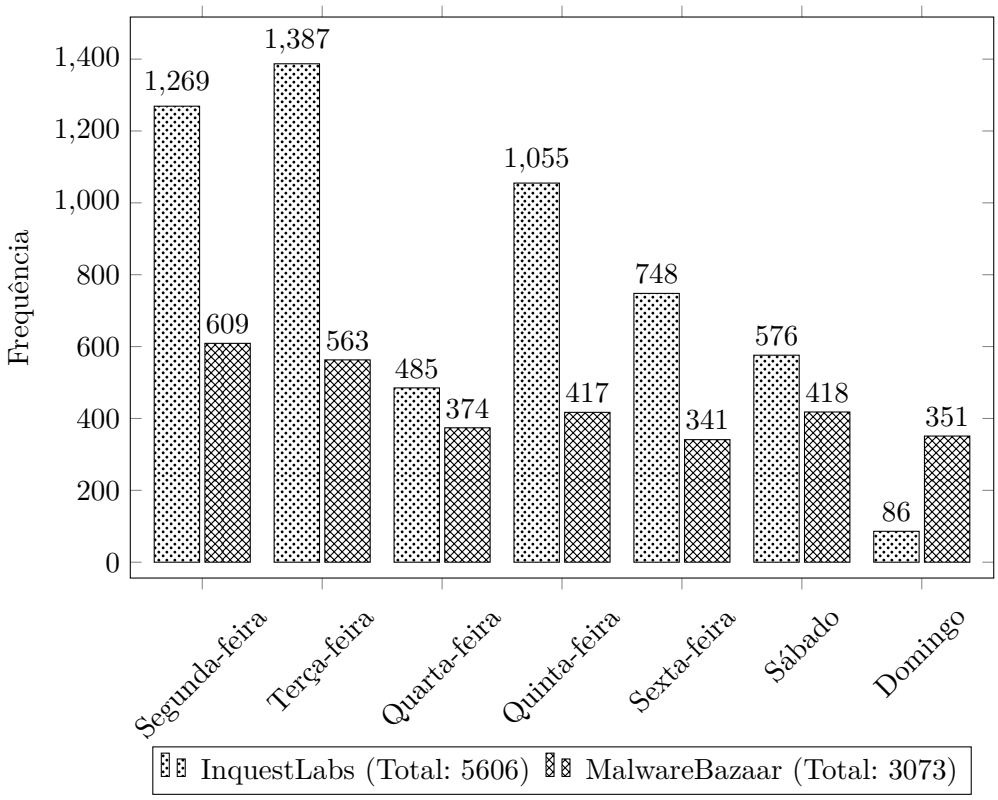

FIGURA 5.9: Comparação do número total de entradas de software malicioso produzidas por cada fonte de informação conforme o dia da semana.

É também possível afirmar que, dentro da amostra de resultados recolhidos, a InquestLabs é mais inconstante no seu caudal de informação, e que a MalwareBazaar é mais consistente (em comparação, a InquestLabs apresentou um desvio padrão de 429,5, contra os 97,6 da MalwareBazaar).

5.2.3 Avaliação do motor de processamento de software malicioso

O motor de processamento decorreu com sucesso, tendo classificado 38 ameaças de acordo com o seu conhecimento interno. Em termos de frequência de ameaças que se verificaram em ambas as fontes, verificou-se uma maior quantidade na quinta-feira, sendo que nesse dia verificaram-se mais ameaças dia-zero do que nos restantes dias combinados. Esta discrepância pode estar relacionada com o tamanho da experiência, que por ser mais curta, pode levar a uma maior volatilidade dos dados apresentados.

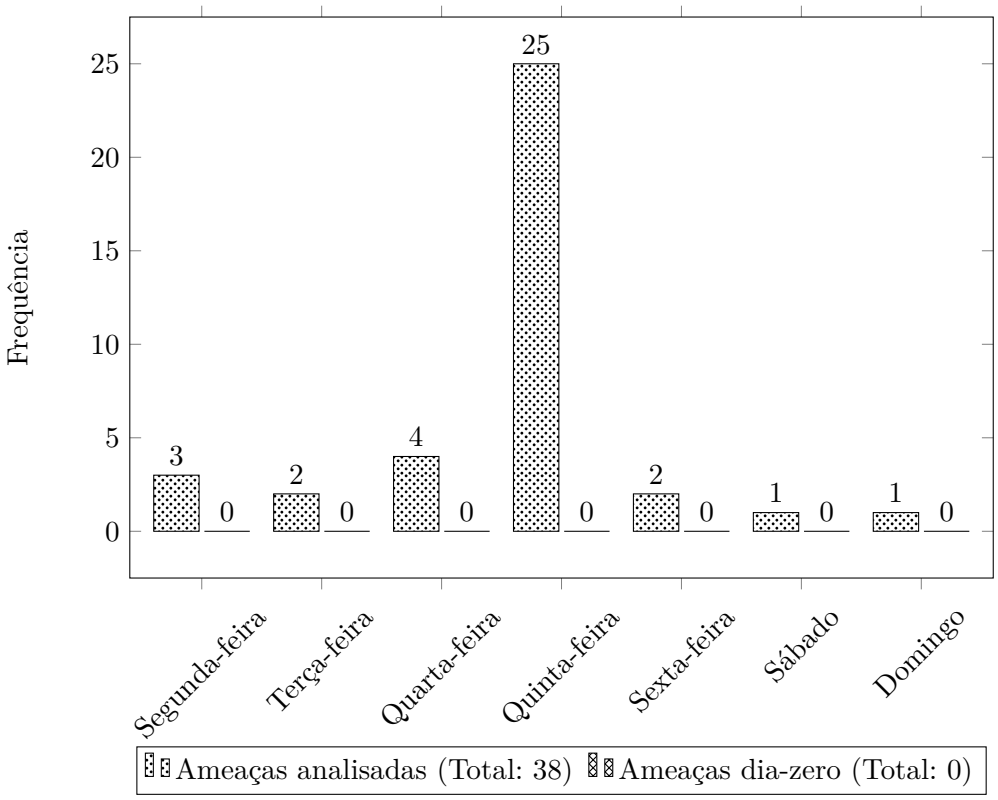

FIGURA 5.10: Comparação do número total de entradas de software malicioso cruzadas internamente conforme o dia da semana.

Durante a semana em que foi realizada a experiência, não foram detetadas ameaças dia-zero pelo sistema. No entanto, as informações agregadas pelo sistema, ao serem guardadas para consulta interna, podem ser úteis para aumentar o conhecimento da empresa sobre uma dada ameaça.

5.2.4 Avaliação geral da componente de software malicioso

Assim, com os dados analisados, é possível verificar que houve um número baixo de ameaças dia-zero relatadas. Apesar disto, o sistema permite acrescentar informações valiosas às ameaças já conhecidas internamente. Para melhorar o número de ameaças dia-zero detetadas, é necessário obter mais fontes de informação que sejam rápidas a relatar softwares maliciosos novos, e que contenham a plataforma afetada. Deste modo, seria possível obter cruzar mais informação valiosa, podendo encontrar mais ameaças dia-zero de uma forma mais rápida.

Infelizmente, por restrições temporais, a análise de resultados foi realizada sobre uma amostra reduzida de resultados, pelo que estes podem ser diferentes do comportamento do sistema a longo prazo.

Capítulo 6

Conclusões e trabalho futuro

6.1 Conclusões

Com este trabalho pretendeu-se criar e implementar um sistema de recolha e tratamento de informação que permitisse avisar a Direção de Cibersegurança da empresa de ameaças (mais especificamente, exploits e software malicioso) dia-zero. Foram realizadas correções ao sistema com vista a sua robustez e manutenção de funcionamento.

Foi desenhado e desenvolvido o ZeroDays v2, de forma a conseguir recolher estes dados. Foi definido que a componente de exploits recolheria informações de três fontes: O 0day.today, a Exploit-DB e a PacketStorm Exchange. Foi também decidido que a componente de software malicioso iria obter os seus dados de duas fontes: A InquestLabs e o MalwareBazaar. A arquitetura do sistema é composta por três motores - os motores de recolha (que realiza a obtenção de informação), processamento (que cruza a informação com as fontes internas e que define se visa sistemas informáticos da empresa) e notificação (que envia alertas sobre ameaças dia-zero). A recolha de informação de ambas as componentes foi calendarizada com a mesma frequência que a implementação do ZeroDays v1 (8 em 8 horas), com o objetivo de definir um ponto de partida pelo qual será criado um horário melhor, mais adaptado ao ritmo de produção de informação das fontes.

Foram definidos um conjunto de requisitos fundamentais para a realização do trabalho: Modularidade, simplicidade, confiança, conclusividade, velocidade, e certeza. Este trabalho cumpriu estes requisitos e os objetivos a que se propôs, sendo que os seus resultados são limitados pelas ferramentas de cadastro disponíveis na empresa e pela falta de dados de algumas das fontes de informação integradas.

Os resultados apresentados pelo sistema foram positivos, sendo que no primeiro mês de atividade da componente de exploits este relatou 7 ameaças dia-zero relevantes para a organização. A componente de software malicioso, embora tenha sido analisada durante muito menos tempo, apresentou ameaças que, apesar de já serem conhecidas internamente, permitem modificar o seu nível de risco bem como o seu tratamento, ao fornecer mais conhecimento sobre estas à equipa de CyberSOC da Direção de Cibersegurança.

Com este projeto, concluiu-se que a recolha e tratamento de dados sobre exploits e software malicioso dia-zero a partir de fontes de informação pré-selecionadas pode permitir a uma empresa manter-se a par destas de uma forma sistemática, podendo assim reagir mais atempadamente a ameaças que podem por em causa a sua segurança.

6.2 Trabalho futuro

No futuro, acredita-se que o sistema ZeroDays v2 possa ser aperfeiçoado nos seguintes aspetos:

- **Deteção** - O sistema ZeroDays v2 é tão bom quanto as fontes que possui. Deste modo, o processo melhorará com a adição de mais fontes de informação, desde que possuam os requisitos identificados na secção 4.1.1.
- **Integração num *playbook*** [1] - A empresa pode no futuro beneficiar de uma integração do projeto num *playbook* que permita agilizar ainda mais o processo de gestão de ameaças dia-zero.
- **Horário** - O horário de recolha de dados pode ainda ser optimizado, de forma a que fique adaptado ao ritmo de produção de informação de cada fonte de informação (por exemplo, se uma fonte de informação publicar informação muito frequentemente, este pode ser adaptado para se ajustar ao seu caudal).
- **Notificação** - Podem ser incorporados mais processos de notificação ao sistema ZeroDays v2, de modo a que possa informar as partes interessadas por outros meios. Isto é importante, não só para reduzir o tempo de reação humana à disponibilização da informação, bem como para evitar possíveis falhas nos sistemas de notificação já implementados.
- **Visualização de dados** - Os dados disponibilizados pelo ZeroDays v2 podem, no futuro, ter uma componente de visualização gráfica, de maneira a tornar a informação gerada mais fácil de compreender e interpretar.
- **Mais tipos de ameaças** - O sistema ZeroDays v2 poderia, no futuro, integrar mais tipos de ameaças que os que já estão implementados, como por exemplo, ataques de phishing e ataques de denial of service.
- **Monitorização de fontes de informação não curadas** - Se for modificado um dos axiomas do projeto ZeroDays v2, de forma a que este passe a contemplar informações recolhidas de fontes não curadas, como por exemplo, o Twitter (tal como os autores do artigo Follow the Blue Bird [4] sugeriram em 2020), podem potencialmente ser recolhidas mais informações sobre as ameaças, bem como melhorar o seu tempo de publicação, a troco de alguma fiabilidade da informação.
- **Utilização de um algoritmo de fuzzy hashing** - No futuro, para poder detetar as entradas duplicadas na componente de exploits, poderá ser implementado um sistema de fuzzy hashing, que permite atribuir um nível de diferença entre o código fonte das entradas. Isto ajudaria a diminuir a quantidade de ameaças que são assinaladas como novas, mas que já foram relatadas.

[1] Um *playbook* é uma lista de procedimentos criados para simplificar e estratificar a resposta a incidentes e ameaças numa organização.

Apêndice A

Expressões RegEx para limpeza do corpo dos exploits

```
/((0A-23-20-20-30-64-61-79-2E-74-6F-64-61-79).*(23))
|((23-20).*?(0A))|([^2-7\-][0-9A-F])|(20)|(7F)
|((?<=\-)\-{1,})|(^-)
|((-23-30-64-61-79-2E-74-6F-64-61-79-5B-){1}
([2-7\-][0-9A-F]\-)*(5D-23){1})
|(((?<=[0-9A-F])\-{1,}\s*$))/
```

Explicação:

```
1. ((0A-23-20-20-30-64-61-79-2E-74-6F-64-61-79).*(23)) - Obtém
o texto que começa por "#  0day.today";

2. ((23-20).*?(0A)) - Obtém todas as linhas que começam por "# ";

3. ([^2-7\-][0-9A-F]) - Obtém todos os caracteres que não estão
entre 2F e 7F;

4. (20)|(7F) - Obtém os caracteres " " (espaço) e DEL;

5. ((?<=\-)\-{1,})|(^-) - Obtém os caracteres "-" que aparecem
duplicados, ou no início do texto;

6. ((-23-30-64-61-79-2E-74-6F-64-61-79-5B-){1} ([2-7\-][0-9A-F]\-)*
(5D-23){1}) - Obtém o pedaço de texto que começa por "#0day.today["
e termina com "]#";

7. (((?<=[0-9A-F])\-{1,}\s*$)) - Obtém todos os caracteres "-"
repetidos, presentes no início do texto, ou no fim do texto.
```

Apêndice B

Fórmulas das métricas utilizadas

VP = Verdadeiros Positivos
VN = Verdadeiros Negativos
FP = Falsos Positivos
FN = Falsos Negativos

$$\text{Precisão} = \text{p} = \frac{VP}{VP + FP}$$

$$\text{Sensibilidade} = \text{s} = \frac{TP}{TP + FN}$$

$$\text{F-Score} = \text{Média Harmónica}(p, s) = \frac{2 * p * s}{p + s}$$

Apêndice C

Ameaça dia-zero não detetada

Na seguinte tabela (Tabela C.1), está representada a entrada que não foi detetada pela componente de exploits como sendo relevante (não foi associada a uma palavra-chave relevante). Esta entrada é considerada relevante para os sistemas informáticos da empresa, porque explora uma vulnerabilidade numa aplicação base existente num sistema operativo relevante.

	ZDT-37910
Data de recolha	2022-08-16 15:05:03
Fonte de informação	ZDT
ID	37910
Plataforma	Linux
Título	Powershell Code Arbitary Execution Builder FUD Exploit
Hash de comparação	Não foi possível obter
Preço	0.042 BTC / 1 000 USD
Descrição	A desired powershell(.ps1) hides the payload with special methods. It allows it to run secretly on the installed computer. Bypasses all modern antivirus protections. Completely FUD

Tabela C.1: Primeira entrada não detetada como dia-zero pela componente de *exploits*.

Referências Bibliográficas

[1] 0day Today Team. "0day.today exploit database", 2021. https://0day.today/ (visitado a: 09/09/2022).

[2] abuse.ch. MalwareBazaar, 2021. https://bazaar.abuse.ch/about/ (visitado a: 09/09/2022).

[3] Airlie, D. [git pull] drm for 5.19-rc1. Linux Kernel Mailing List, Maio 2022.

[4] Alves, F., Andongabo, A., Gashi, I., Ferreira, P. M., and Bessani, A. Follow the blue bird: A study on threat data published on twitter. In *Proceedings of the European Symposium on Research in Computer Security, September 18, 2020*. pp. 217–236.

[5] BluePrism. BluePrism RPA, 2021. https://www.blueprism.com/pt/ (visitado a: 09/09/2022).

[6] Buchanan, W. J., Woodward, A., and Helme, S. Cryptography across industry sectors. *Journal of Cyber Security Technology 1*, 3-4 (2017). pp. 145–162.

[7] Centro Nacional de Cibersegurança dos Países Baixos. Taranis Wiki, 2021. https://github.com/NCSC-NL/taranis3/wiki (visitado a: 09/09/2022).

[8] Dionísio, N., Alves, F., Ferreira, P. M., and Bessani, A. Cyberthreat detection from twitter using deep neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (2019).

[9] Elastic. Elk Stack, 2021. https://www.elastic.co/what-is/elk-stack/ (visitado a: 09/09/2022).

[10] ENISA. Enisa threat landscape 2021. *ENISA Threat Landscape 2021* (Outubro 2021). https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021 (visitado a: 09/09/2022).

[11] Exploit-DB. Exploit database - exploits for penetration testers, researchers, and ethical hackers, 2021. https://www.exploit-db.com/ (visitado a: 09/09/2022).

[12] Fayi, S. Y. A. What petya/notpetya ransomware is and what its remidiations are. In *Information technology-new generations*. 2018, pp. 93–100.

[13] FIRST. Common vulnerability scoring system version 3.1: Specification document, 2021. https://www.first.org/cvss/specification-document (visitado a: 09/09/2022).

[14] Greenberg, A. The untold story of notpetya, the most devastating cyberattack in history. *Wired August 2018 22*.

[15] Initiative, Z. D. Zero day initiative, 2021. https://www.zerodayinitiative.com/ (visitado a: 09/09/2022).

[16] InQuest Labs. InQuest Labs, 2021. https://labs.inquest.net/about (visitado a: 09/09/2022).

[17] InQuest Labs. InQuest Labs - DFI, 2021. https://labs.inquest.net/dfi (visitado a: 09/09/2022).

[18] Koujalagi, A., Patil, S., and Akkimaradi, P. The wannacry ransomeware, a mega cyber attack and their consequences on the modern india. *International Journal of Management Information Technology and Engineering 14* (2018).

[19] Lawrence, H., Hughes, A., Tonic, R., and Zou, C. D-miner: A framework for mining, searching, visualizing, and alerting on darknet events. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)* (2017), pp. 1–9.

[20] Liu, Z. Working mechanism of eternalblue and its application in ransomworm. *CoRR abs/2112.14773* (2021).

[21] MalShare. MalShare Project, 2021. https://malshare.com/about.php (visitado a: 09/09/2022).

[22] Menéndez, H. D. Malware: the never-ending arm race. *Open Journal of Cybersecurity 1*, 1 (2021). pp. 1–25.

[23] Micro, T. exploit, 2021. https://www.trendmicro.com/vinfo/us/security/definition/exploit (visitado a: 09/09/2022).

[24] Microsoft. Microsoft security bulletin summary for march 2017. https://docs.microsoft.com/en-us/security-updates/securitybulletinsummaries/2017/ms17-mar (visitado a: 02/08/2022).

[25] Miguel Pupo Correia e Paulo Jorge Sousa. *Segurança no Software - (2ª Edição)*. FCA, 2017.

[26] Mishra, U. An introduction to computer viruses. *Available at SSRN 1916631* (2010).

[27] MITRE. CVE, 2021. https://www.cve.org/About/Overview (visitado a: 09/09/2022).

[28] Morrison, R. Apple zero day vulnerability on the market for €8 million, Agosto 2022. https://techmonitor.ai/technology/cybersecurity/apple-zero-day-vulnerability-android (visitado a: 01/09/2022).

[29] Nunes, E., Diab, A., Gunn, A., Marin, E., Mishra, V., Paliath, V., Robertson, J., Shakarian, J., Thart, A., and Shakarian, P. Darknet and deepnet mining for proactive cybersecurity threat intelligence. In *Proceedings of the IEEE Conference on Intelligence and Security Informatics*. 2016.

[30] NVD. CVE-2017-0144. https://nvd.nist.gov/vuln/detail/CVE-2017-0144 (visitado a: 02/08/2022).

[31] Pfleeger, C. P., and Pfleeger, S. L. *Analyzing computer security: A threat/vulnerability/countermeasure approach*. Prentice Hall Professional, 2012.

[32] Rivest, R. L. The MD5 message-digest algorithm. RFC 1321, April 1992. http://www.rfc-editor.org/rfc/rfc1321.txt.

[33] Saleous, H., Ismail, M., AlDaajeh, S. H., Madathil, N., Alrabaee, S., Choo, K.-K. R., and Al-Qirim, N. Covid-19 pandemic and the cyberthreat landscape: Research challenges and opportunities. *Digital Communications and Networks* (2022).

[34] Security, P. S. Packet storm security, 2021. `https://packetstormsecurity.com` (visitado a: 09/09/2022).

[35] Sharpe, R. Just what is SMB? https://www.samba.org/cifs/docs/what-is-smb.html (Visitado a 27/09/2022).

[36] Shirey, R. Internet security glossary. RFC 2828, IETF, Maio 2000.

[37] Sillaber, C., Sauerwein, C., Mussmann, A., and Breu, R. Data quality challenges and future research directions in threat intelligence sharing practice. In *Proceedings of the Workshop on Information Sharing and Collaborative Security* (Out. 2016). pp. 65–70.

[38] Stevenson, J. The wannacry ransomware attack (2017). *Strategic Comments 23*, 4 (2017). pp. vii–ix.

[39] Stockton, P. N., and Golabek-Goldman, M. Curbing the market for cyber weapons. *Yale Law & Policy Review 32* (2013). pp. 239.

[40] Trautman, L. J., and Ormerod, P. C. Wannacry, ransomware, and the emerging threat to corporations. *Tenn. L. Rev. 86* (2018), 503.

[41] Valea, O., and Oprişa, C. Towards pentesting automation using the metasploit framework. In *Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing* (2020). pp. 171–178.

[42] Vigário, J. Zerodays: Sistema de gestão de ameaças de ciber-segurança de diazero. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2021.

[43] VirusBay. About - VirusBay, 2021. `https://virusbay.io/about` (visitado a: 09/09/2022).

[44] Vulners. Vulners - vulnerability data base, 2021. `https://vulners.com/` (visitado a: 09/09/2022).