Detecting Network Threats using OSINT Knowledge-based IDS

Ivo Vacas[1,2] Ibéria Medeiros[1] Nuno Neves[1]
[1]*LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal*
[2]*Centro Nacional de Cibesegurança, Portugal*
ivo.vacas@cncs.gov.pt, imedeiros@di.fc.ul.pt, nuno@di.fc.ul.pt

***Abstract*—Cybercrime has steadily increased over the last years, being nowadays the greatest security concern of most enterprises. Institutions often protect themselves from attacks by employing intrusion detection systems (IDS) that analyze the payload of packets to find matches with rules representing threats. However, the accuracy of these systems is as good as the knowledge they have about the threats. Nowadays, with the continuous flow of novel forms of sophisticated attacks and their variants, it is a challenge to keep an IDS updated. Open Source Intelligence (OSINT) could be explored to effectively obtain this knowledge, by retrieving information from diverse sources. This paper proposes a fully automated approach to update the IDS knowledge, covering the full cycle from OSINT data feed collection until the installation of new rules and blacklists. The approach was implemented as the `IDSoSint` system and was assessed with 49 OSINT feeds and production traffic. It was able to identify in real time various forms of malicious activities, including botnet C&C servers communications, remote access applications, brute-force attacks, and phishing events.**

***Keywords*—Threat detection, Intrusion detection systems, OSINT, Threat intelligence, Network security**

I. Introduction

Nowadays, institutions are often being targeted by cybercrime groups. Attacks on different hosts (from servers to devices) are getting increasingly frequent and with a profound impact, many times taking the form of distributed denial of service (DDoS), spamming, and malware. For example, it is estimated that in 2019 the costs associated with these criminal actions can reach 2 trillion dollars [1]. Therefore, cybersecurity has become one of the higher concerns and priorities of institutions, independently of their business areas [2].

Many times attacks are carried out by first infecting the victims' hosts with malware or a backdoor, for later remotely control them to perform malicious actions [3]. Botnet attacks have increased over the last years. For example, according to the 2016 and 2017 Akamai reports, such attacks have risen 129% from year 2015 to 2016 [4], and year 2017 closed with an increase of 10% in web application attacks and 14% in DDoS attacks [5]. The reason behind the DDoS attacks growth is in part explained by the larger number of connected IoT (Internet of Things) devices, which have reached approximately 3 billions in 2017 [6]. For instance, Mirai was the first known botnet that used a considerable number of IoT devices. In 2016, Mirai attacked the Dyn servers (organization that controls DNS) with 100.000 IoT devices that generated a traffic volume in order of 1.2 Tbps, causing Internet unavailability across Europe and USA [7].

Institutions normally protect themselves from attacks by deploying IDS that monitor the network traffic, analyzing the payload of the packets to find matches with rules about threats, identifying anomalies and generating alerts [8]–[10]. The most common IDSs employ signatures, generating an alert whenever the observed traffic is similar to one of the behaviours represented in a rule [11]. Sometimes IDSs also resort to machine learning techniques to improve precision [12], [13]. Other approaches just inspect network flows, i.e., they do not check the payload of the packets, but only their headers [14]–[16]. In all cases, however, the accuracy of an IDS is as good as the knowledge it has about the threats. However, it is a challenge to keep IDSs updated with the continuous emergence of novel forms of sophisticated attacks. In addition, usually for each kind of attack, there are many slight variants that can cause missing detection (false negatives) [10]. Knowledge about recent attacks might also be hard to obtain because it is considered a business secret [17].

OSINT could be explored to collect rapidly (and often affordably) knowledge about threats, by retrieving information from diverse sources (both public and private). OSINT is normally composed of shared data provided for instance by the examination of honeypots and other security event descriptions about any kind of damage on services and systems, such as credential theft, phishing or DDoS [11]. Sometimes the data included in individual events is not enough to characterize real threats, being useless for detection either by behaviour and by patterns. However, if the data from multiple events/sources could be aggregated and complemented with other information, it would be possible to create indicators of attacks (IoA) [18] that could clearly characterize certain threats, allowing their discovery.

This paper presents an approach to enhance attack discovery capabilities, by automatically updating the IDS threat knowledge. Our solution covers the full cycle, from collecting OSINT feeds until installing rules at the intrusion detector. It involves the processing and correlation of OSINT data in order to create IoAs, which then support the generation of rules and blacklists, and their integration in the IDS. The current prototype is called `IDSoSint`, and it was assessed with 49 OSINT sources and real traffic. During a period of time it was used to analyze the traffic of a few of the main network links of the University of Lisbon, scanning a large amount of packets. IDSoSint was able to identify in real time various forms of malicious activities, including botnet C&C servers communications, remote access applications, brute-force attacks, and phishing events.

In summary, the main contributions of the paper are: (1) an approach for improving IDS capabilities by leveraging OSINT; (2) a process to generate IoAs based on OSINT processing; (3) a rule and blacklist generator for IDS from IoA's; (4) a system, IDSoSint, that implements the approach, and its experimental evaluation with real traffic, showing the ability to detect various types of attacks in a (near) production environment.

II. Context and related Work

A. Botnets, attacks and IDSs

Botnets are networks of vulnerable devices (e.g., computers and mobile devices) that were compromised, and then controlled by criminal entities – botmasters – through a command and control center (C&C) [19]. Botnets are one of the main roots of SPAM, malware propagation, ransomware, phishing, and DDoS attacks. Botnets C&C are a valuable target, both for entities who try to detect and destroy them and for other cybercriminals who attempt to steal them for their benefit. Botmasters use cryptographic mechanisms to protect the communications within the botnet, and dissimulation mechanisms to prevent their identification and capture [20] [21]. Therefore, it is difficult to eliminate such threats because they are getting more sophisticated, and each day they appear in many variants, precluding easy forms of detection.

IDS are a huge contribution for the discovery of those threats. They can analyze the infrastructure of an organization and identify attack patterns. There are two kinds of IDS, network-based and host-based. While the former can validate the network traffic to find malicious communication patterns, the latter analyzes the system itself (e.g., computer). Both IDS types can operate by signatures or by behavior, but they are as good as the knowledge they have about the threats. In this paper, we will focus on network-based IDSs.

Signature-based IDS use a database where each entry contains a representation of some characteristic of an attack, which is called a signature. When a packet is processed by the IDS, its content is compared with the database entries in order to find a matching signature, detecting in this way the malicious traffic. This kind of solution only identifies known threats, i.e., those that have their signatures stored in the database. For this reason, this solution has the benefit of having a low number of false positives, and can provide good alert information about the identified threats. On the other hand, it can have a high number of false negatives, namely when ”unknown” threats do not appear in the database [10], [11].

Behavior-based IDS are based on the behavior of systems it knows and monitors. It starts by learning how a system should behave, and it detects suspicious and incorrect actions when the monitored system deviates from the expected behavior, and generates an alarm every time this occurs. These systems are complex and require an extensive and continuous study about the system under analysis. Contrarily to signature-based IDS, this solution can have a lower false negative rate because it can detect unknown and uncommon behaviors. On the other hand, it has a high false positive rate because any behavioral deviation generates an alarm, even if there is no illicit action.

B. Intrusion Detection Tools

Zeng et al. proposed a hybrid architecture to detect botnet communications by matching network traffic and host information. The approach first analyzes network traffic to find machines that have a different behavior than expected, and then it inspects those hosts to discover malicious artefacts. This combination allows better coverage and evaluation of the monitored systems as it can detect common botnet communication behaviors and inspect hosts individually [8].

Sperotto et al. [22] proposed an IDS based on inspecting network traffic packet headers instead of their packet payloads. This IDS only detects attacks that can be found by header analysis. On contrast, the approach we present inspects packet payloads, allowing to detect botnets and other types of anomalies such as phishing.

A survey on botnet detection showed that, despite the current level of expertise and the technologies used by IDS, only a few methods are effective, and the better ones are based on anomaly approaches. Nevertheless, the detection of botnets remains difficult due to the cloaking techniques used by botmasters continuously [10]. Aviv et al. proposed a cooperation of entities, since they have confidential and private information that, although it can not be shared and made public, it can be used to mitigate the impact of botnets by detecting them [17].

FeatureSmith detects malware in android applications based on knowledge gathered from these applications and employing data mining techniques. A dataset was obtained from google scholar scientific papers related to android malware, and the data mining technique was applied to semantic relationships and malware behaviour. The system detected 92.5% of malware and had a false positive rate of 1% [9]. Unlike FeatureSmith, the approach we propose detects threats by using OSINT data, which is processed to construct IDS rules and obtain blacklists, and to be integrated in IDS as knowledge.

MISP is a collaborative platform to share information about security threats. Its architecture is split into organization and community. These groups allow to share information in a secure, simple and controlled way at the organization level or in a given community [23]. MISP follows an approach similar to ours but does not perform correlations among the collected OSINT, and the rules it generates when applied to an IDS decrease performance, turning it unusable.

III. OSINT-based IDS Approach

Our approach enhances the attack detection capabilities of an organization by renewing the information about threats configured in the IDSs, inserting novel rules and blacklists

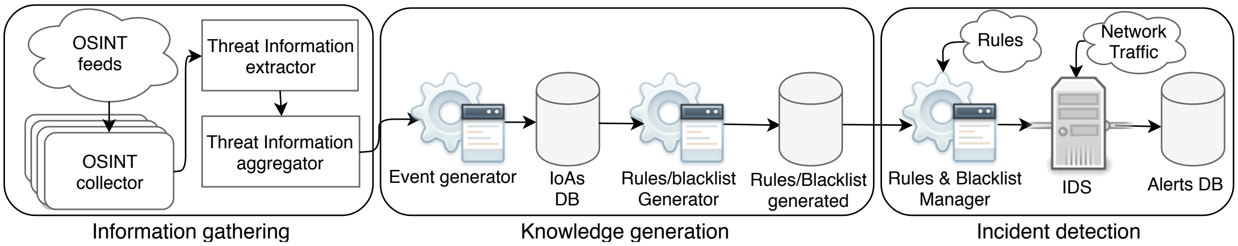


Fig. 1: OSINT-based IDS architecture overview.

built automatically from OSINT data. It performs the complete cycle of IDS knowledge update, from collecting threat intelligence, generating rules, and installing them.

OSINT-based threat events are provided by many feeds and Internet sources, and they have diverse formats and data about security related incidents. The rules that we intend to create vary significantly in terms of complexity depending on the attack. This means that sometimes a single OSINT event contains enough data to construct a simple rule. However, in other cases, the derivation of more complex rules requires the combination of multiple OSINT events and other external information. Still, it often happens that these complex rules are the ones necessary to actually support the detection of more sophisticated attacks.

Therefore, we process OSINT data with the goal of verifying the interrelation among different security events in order to complement each other and obtain the relevant and complete information necessary to describe attacks. Moreover, based on this information, IDS rules (both simple and complex) are built automatically, ready to be integrated into an IDS. As OSINT reports characterize many threat categories, rules for quite distinct types of attacks are constructed, improving the IDS capabilities in two directions: the detection of an expressive number of different attacks, and the robust discovery of sophisticate attacks.

The approach acts continuously in a loop, updating the IDS with the knowledge that was acquired through OSINT, in order to achieve better performance of detection. This means that for each loop iteration all phases of the approach are executed, as shown in Fig. 1. These phases are summarized as follows:

1) *Information gathering*: collects the threat intelligence provided by OSINT feeds, and aggregates the security events by threat categories.
2) *Knowledge generation*: processes the aggregated data, establishing associations between security events and other external information, and generating knowledge that can describe attacks clearly. Based on this knowledge, IDS rules and IP blacklists are constructed.
3) *Incident detection*: updates the IDS knowledge database with the rules and IP blacklists, allowing the system to remain up to date and able to identify new malicious activities. Incidents are registered based on matching the observed traffic with the rules and blacklists.

The first two phases are carried out periodically, within a predefined time range (e.g., daily) that is dependent on the availability of OSINT updates, to process event feeds and generate knowledge, whereas the third phase is always in execution.

The next sections explain the phases in detail.

A. Information gathering

This phase comprises three steps implemented by the components: *OSINT collector*, *threat information extractor*, and *threat information aggregator*. These components are respectively responsible for gathering the OSINT events from the pre-configured sources, as indicated by the SOC (Security Operations Center) analyst; extracting the data from the events; and aggregating the information from related events.

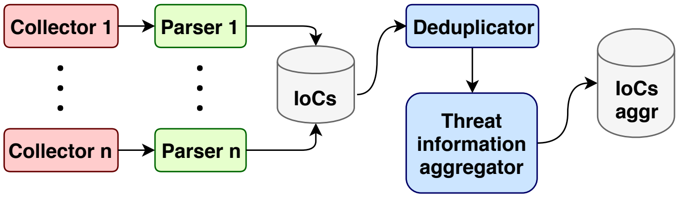
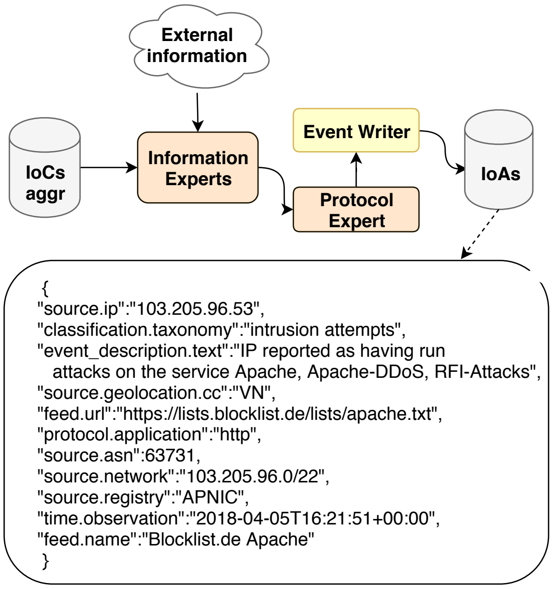


Fig. 2: Information gathering data flow in detail.

The data flow of these steps is displayed in Fig. 2. It starts with several *collectors*, each associated with a particular feed (e.g., *phishtank*, *malwaredomainlist*), where they obtain an influx of OSINT events. Multiple collectors can be configured for the same threat category (phishing, malware domain) depending on the number of feeds the SOC analyst wants to gather. Therefore, various events can relate to the same threat category. However, OSINT data is available on different formats (e.g., cvs, txt, doc), and thus it is necessary to harmonize events to a single format before processing. To do so, the individual *parsers* analyze each of these events, classifying them into one of the pre-defined threat categories, and translating the events into the indicators of compromise (IoC). IoC are standardized forensic information artifacts used in the identification of potential malicious activities of a compromised system or network [18]. The IoC structure is specified as a JSON object composed of a set of `<key:value>` pairs. Each pair contains data that can characterize malicious activity, and the set of pairs is the union of data collected from a part of an event, since an event can contain various IoCs related with the same security issue. Afterwards, a *deduplicator* checks for repeated IoCs by utilizing pattern matching because feeds from the same category can emit equal threats, ensuring thus that similar IoCs are erased and achieving a better performance on all this process. Next, an



```
{
"source.ip":"103.205.96.53",
"classification.taxonomy":"intrusion attempts",
"event_description.text":"IP reported as having run
  attacks on the service Apache, Apache-DDoS, RFI-Attacks",
"source.geolocation.cc":"VN",
"feed.url":"https://lists.blocklist.de/lists/apache.txt",
"protocol.application":"http",
"source.asn":63731,
"source.network":"103.205.96.0/22",
"source.registry":"APNIC",
"time.observation":"2018-04-05T16:21:51+00:00",
"feed.name":"Blocklist.de Apache"
}
```

Fig. 3: Indicator of attacks (IoA) generation data flow.

extractor examines the IoCs, detecting and extracting relevant data about threats (e.g, domains, malicious IP for a specific protocol), and the *threat information aggregator* groups them by category of threat.

B. Knowledge generation

This phase builds knowledge that will improve the detection capability of the IDS. It is composed of two main steps implemented by the *event generator* and the *rules & blacklist generator*. They are responsible for processing the aggregated IoCs and creating rules & blacklists based on such data.

1) Event generator: The actions carried out by the event generator are detailed in Fig. 3. The aggregated IoCs are processed by the *information experts* to enhance them with other external data, not coming from OSINT feeds. Experts are configured by administrators, for example, to run network commands to obtain detailed data about the IP provided in the IoC, such as *whois*, geo location, and *asn* source, or to obtain features about the malware domain, such as the port. Next, the *protocol expert* correlates all the information, identifying interconnection points between them and enhancing them, with the objective of getting together the most complete description of an attack. The various data is then merged as an indicator of attack (IoA). IoAs allow the attacks to be identified on systems executing at runtime [18], [24]. The IoA structure follows the structure of an IoC, i.e., a set of `<key:value>` pairs, where each pair contains data that characterizes an attack, and the set of pairs is the union of data collected from the various security events. In addition, the *protocol expert* adds to the IoAs the communication protocol associated with the attack and the *event writer* stores the IoAs in a database. The reader should note that although we create the IoAs mainly to construct the IDS rules, they are not limited to this task and can be used for other purposes (e.g., decide to quarantine certain hosts; use of stronger authentication measures).

The bottom of Fig. 3 shows an example of an IoA. This IoA describes attacks against the *Apache service*, namely DDoS and RFI (remote file inclusion) attacks. To construct this IoA, the data it contains was obtained as follows: a collector was configured to get OSINT data from feed `https://lists.blocklist.de/lists/apache.txt`, named `Blocklist.de Apache`, and with a description `IP reported as having run (...) RFI-Attacks`. Notice that this feed only provides a list of IPs that attacked the Apache service, so all the information contained in this IoA do not come from it. Next, after the parser extracting the `103.205.96.53` IP from that list, an IoC is created to it, and then aggregated with another IoCs associated to Apache service. In this case, this IoC is unique. However, if there were another IoCs belonging to the same IP network, they would be represented (aggregated) as a sub-network or an IP range. The *information experts* get external information about that IP (source fields in the IoA). In addition, the field `protocol.application` (sixth pair in the example) contains the protocol targeted by the attack, which was added by the *protocol expert*.

2) Rules & blacklist generator: This component uses the data contained in the IoAs to build IDS rules and compose lists of IP addresses considered malicious (blacklist). Fig. 4 displays this process. To create a rule, first it checks the protocol indication contained in `protocol.application` to determine both the source and destination ports. Next, it constructs the rule with this data, adding the remaining information from the IoA to make the rule more specific for a particular attack. Next, it stores the rule in the rules database. In parallel, if the IoA only contains IPs tagged as malicious, the generator creates an IP blacklist with them, and stores the result in the blacklist database.

In general, an IDS rule is specified as follows:

```
alert [action][proto][srcIP][srcPort] ->
[destIP][destPort] ([rule options])
```

where `proto` indicates the protocol that the rule refers to, such as TCP, UDP or ICMP; `srcIP/destIP` and `srcPort/destPort` specify the source/destination IP and port; `rule options` defines the patterns used to analyze the packet payload, checking if it matches with some of them

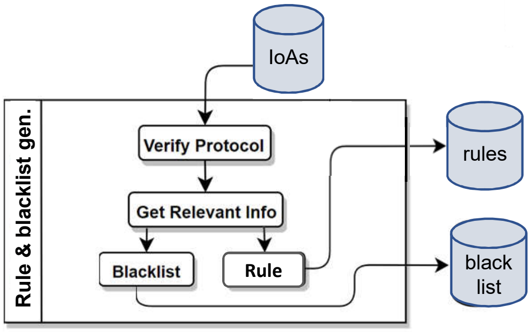


Fig. 4: Data flow for the rules & blacklist generator.

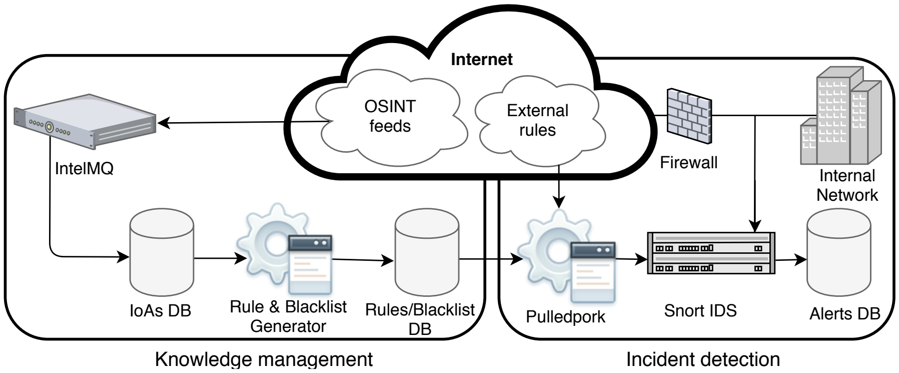


Fig. 5: IDSoSint system divided into knowledge management (left) and incident detection (right) components.

and detecting an attack. If so, the defined action is executed. In such case, an alert message is emitted stating that the rule was triggered, and the packet payload is stored temporarily so that later it can be analyzed by the SOC team.

For example, from the IoA in Fig. 3 the generated rule is:

```
alert tcp 103.205.96.53 22 -> $HOME_NET 80
(msg:"Intrusion attempts to Apache; sid 100")
```

in which it is identified by ID 100 (sid 100) and the message Intrusion attempts to Apache being emitted when TCP traffic coming from 103.205.96.53 machine through 22 source port tries to reach the Apache service running on port 80 in the UL network identified by variable $HOME_NET. Notice that the port 80 is determined based on the information given by the protocol.application.

C. Incident detection

The detection of malicious activities by the IDS is done with the help of the rules and blacklists generated by the previous phase. However, before updating the IDS with this knowledge, the *rules & blacklist manager* needs to perform some checks.

This manager (see Fig. 1) administers all rules and blacklists with the objective of avoiding the duplication of rules in the IDS, enhancing thus its performance, and handling the IP's reputation. The checking task is carried out at each pre-defined time interval (e.g., daily) before updating the IDS. Afterwards, the new rules and blacklisted IPs are deployed into the IDS, and it starts running with them.

For its part, before the IDS analyzes the packet payload, it first ascertains the IP headers using the blacklists. If no match occurs, next, it uses the rules. This sequence of analysis improves the IDS performance, since the validation of packet headers is faster than looking at the packet contents. The actions and alerts are triggered if any rule finds illicit activity. Following the example described above, that rule is inserted into IDS, and then if the host 103.205.96.53, via a SSH connection (port 22), tries to communicate with the UL web server (Apache service) which is running on port 80 in a host belonging to the $HOME_NET network, an alert will be emitted and the correspondent packet stored for later analyzes.

IV. IDSoSint System

The approach was implemented in our IDSoSint system, which is illustrated in its current form in Fig. 5. It is divided into two parts – *knowledge management* and *incident detection* – where the former implements the first two phases of the approach and includes three modules we have developed, and the second component corresponds to the third phase.

IDSoSint uses a few open-source tools and some modules that we programmed in Python to facilitate the integration with other software packages. We setup the IntelMQ platform [25] to collect various OSINT feeds and to produce the IoCs. Then, the system is composed of three modules: the *events correlator*, for correlating IoCs in order to get relevant information that characterizes attacks; the *IoA generator*, for generating IoAs based on the correlation results; and the *rule & blacklist generator* to get the IDS rules and blacklist based on the IoAs. In addition, we configured the Pulledpork platform [26] to manage the rules and blacklists, and the IDS Snort [27] to process the packets with our rules.

V. Experimental Evaluation

The objective of the experimental evaluation is to answer the following questions: (1) Is IDSoSint able to detect attacks by processing real data? (2) Is IDSoSint able to identify the type of attacks performed? (3) Is IDSoSint able to explore OSINT data to generate knowledge that can improve the detection capabilities of IDSs?

In order to validate our approach, we evaluate IDSoSint in a production environment, by scanning during 8 days the traffic of a few of the main network links of the University of Lisbon (UL), while analyzing a large amount of packets.

The rest of the section is organized as follows: in Subsection V-A the system set up is presented, focusing on the OSINT feeds; and in Subsections V-B and V-C are described and discussed the results. These last two subsections answer questions 1 to 3.

A. Setting-up IDSoSint

The IDSoSint system was set up in a HP ProLiant DL360 G6 machine, with 2 Intel Xeon CPU X5550 at 2.67GHz, 12GB of RAM, 165GB hard disk, and 3 network interfaces (a 10Gb/s optic fibre channel and 2 1Gb/s ethernet interfaces).

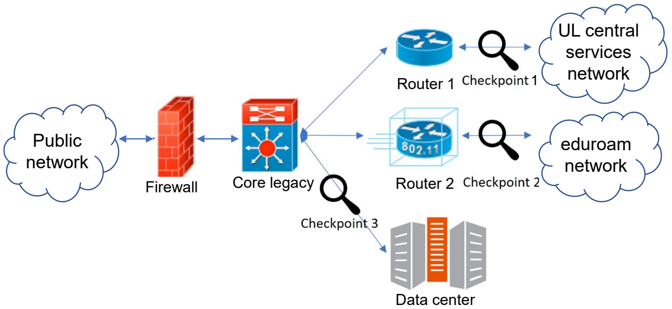


Fig. 6: University of Lisbon network overview with the three links that were evaluated by *IDSoSint*.

The installed operating system was the Security Onion Linux distribution, which is already fully equipped with network security analysis applications such as Snort. By default Snort does not come with rules defined. Although the Talos Cisco team releases a Snort rule-set, we opted by using only the rules that IDSoSint generated, since the former are released once per week and ours are constructed and deployed daily.

TABLE I: Distribution of feeds by OSINT events.

OSINT event	**Feed**	**OSINT event**	**Feed**
Blacklist IP	16	Malicious IP against MAIL server	2
IPs trying remote access	9	Malicious IP against FTP server	1
Malicious domain	9	Malicious IP against SNMP server	1
Malicious IP against VoIP server	4	Phishing or malicious links	7

The IntelMQ threat intelligence platform was installed in a virtual machine with 4GB of RAM, and 17 GB of hard disk. IntelMQ was configured to collect 49 OSINT feeds from different categories, provided by 44 public and open source repositories belonging to 19 entities.[1]

Some of these feeds are well known, such as Phishtank [28] that reports phishing incidents, CINSScore [29] for blacklisted IPs, and Malware Domain [30] for share malware domains. Table I distributes the 49 OSINT feeds by our 8 event categories, which allowed the collection of threat intelligence information about diverse kinds of security incidents.

As stated, we analyzed a few connection links of the computer infrastructure of the UL. Fig. 6 gives an overview of the network, displaying the three links where we deployed `IDSoSint` (checkpoints in the figure). The UL central services network includes for example the administrative services and human resources; the data center stores the data of most of the institution (e.g., websites); and eduroam is the academic European wireless network. We selected these links because of their size and characteristics, as they connect parts of the infrastructure that have distinct purposes, therefore ensuring that a wider range of traffic is observed.

B. Incident Record

We analyzed the rules and blacklists that were created, and it was possible to verify that they were correctly constructed by the *rule & blacklist generator*, and that they could be imported by Snort. During the evaluation period, on average, 5.290 rules and 14.590 blacklist entries were generated daily, with a total of 42.320 and 116.720 entries, respectively. `IDSoSint` registered an overall 5.5 million incidents when it applied these rules and blacklists to the observed traffic. For those alerts triggered by rules, the analyzed packets were stored for later investigation. However, since the rules used by Snort were the ones generated by the *rule & blacklist generator* and are based on reputational OSINT feeds, all the incidents registered were effective and most of them regarded access attempts to unavailable services on UL network.

TABLE II: Number of registered incidents (8 days).

Day 1	**Day 2**	**Day 3**	**Day 4**	**Day 5**	**Day 6**	**Day 7**	**Day 8**
410.795	523.005	411.344	505.541	472.309	840.062	605.033	1.683.571
Total: 5.451.660							

Table II summarizes the number of incidents detected in each day. On average, 540.000 alerts were emitted on the initial 7 days, whilst in the the last day (day 8) there was an important increase on the alerts. The explanation for this increase is most probably due to a novel high impact malware that appeared in that day.

TABLE III: Number of collected OSINT events that belonged to each of the eight categories used to classify the rules & blacklisted IPs, and the corresponding incidents that were detected.

Category	**OSINT event**	**Incident**
Blacklist IP	116.313.458	3.779.638
IPs trying remote access	864	1,588,024
Malicious domain	4.223	45.843
Malicious IP against VoIP server	118	23.823
Malicious IP against MAIL server	397	12.005
Malicious IP against FTP server	23	1,536
Malicious IP against SNMP server	48	791
Phishing or malicious links	19.453	0
Total	**116.338.584**	**5.451.660**

Table III presents the number of OSINT events that were collected and processed in each one of our threat categories, i.e., extracted from the 49 OSINT feeds (column 2). It also displays how the registered incidents were distributed per category (column 3). It is possible to observe that the number of collected OSINT events was much larger than the number of entries that were set in the IDS, respectively, around 116 million and 160 thousand. This reduction is explained, on one hand, by an efficient detection of duplicates, and on the other hand, by the successful aggregation that can be achieved by representing the threats as IoAs. Recall that an IoA describes an attack using the information included in a set of related events. In addition, the table shows the effectiveness of the rules at producing diverse kinds of alerts, where the most common incident was related to communications of blacklisted IP. This gives evidence that OSINT data can be explored to create knowledge to be used by protection assets in a network infrastructure, such as an IDS. Moreover, these numbers demonstrate how cybercrime is well present in public networks, and that it has to be an actual concern for organizations.

[1] 8 of them: https://zeustracker.abuse.ch, https://www.openphish.com, https://www.spamhaus.org, https://reputation.alienvault.com, https://lists.blocklist.de, https://www.malwaredomainlist.com, https://www.cinsscore.com, https://www.team-cymru.org.

TABLE IV: Number of incidents by category.

Event	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Alert
Traffic made by a blacklisted IP	399652	378032	375775	482454	454109	770526	568797	350293	3,779,638
SSH communication by a malicious IP	7928	137997	30030	5788	5076	60203	9091	1319138	1,575,251
Telnet communication by a malicious IP	101	117	136	127	140	173	128	81	1,003
Communication with a malicious domain	663	2476	1564	10533	7008	3492	19726	381	45,843
Web communication by a malicious IP	228	42	58	156	0	0	750	10536	11,770
IMAP communication by a malicious IP	333	220	421	1251	2271	1452	1214	68	7,230
SMTP communication by a malicious IP	64	1597	636	52	125	542	799	960	4,775
FTP communication by a malicious IP	7	14	22	1415	33	0	36	9	1,536
SNMP communication by a malicious IP	171	150	84	85	108	115	58	20	791
SIP OPTIONS request by a malicious IP	1648	2210	2470	3270	3295	2997	4365	1871	22,126
SIP REGISTER request by a malicious IP	0	75	74	410	0	486	69	214	1,328
SIP communication by a malicious IP	0	75	74	0	144	76	0	0	369
Total	410795	523005	411344	505541	472309	840062	605033	1683571	5,451,660

A more detailed analysis of the rules and registered incidents shows that in some cases, a few rules of certain categories, as FTP for example, were sufficient to detect a considerable number of illicit actions. However, in other cases, although the rules to detect such actions were correct, no incidents were recorded (as in the *phishing or malicious links* category). This absence is justified by the pre-processing made by Snort, where the packet headers are checked before the analysis of their payloads. In that case, if the phishing IP belongs to the list of malicious IPs, Snort detects the problem in the pre-processing phase, emits an alert, and ends the analysis for that packet. In this situation, the associated rules ended up not being used during the packet validation.

Table IV presents the 5.5 millions of recorded incidents stratified by category, i.e., by the type of rule that triggered the incident. The traffic generated by blacklisted IPs is responsible for approximately 70% of recorded incidents. We recall that the blacklisted IPs come from feeds that only report malicious IPs and they are used by Snort pre-processor. Therefore, this result is justified by the fact that the Snort's pre-processor stops any other validation when the IPs of the packets belong to the blacklists, which is useful from a performance optimization perspective. The remaining 30% incidents were emitted by Snort's rule processor. Here, the most common incidents were *SSH communications from IPs tagged as malicious*. In the *communication with malicious domain* category, some of the observed patterns were associated to bots and C&C communications. Such observations were made based on the investigation of the analyzed packet that were stored when rules were triggered. The same observation is also true with regard to the generated alerts for IMAP, SMTP, and FTP communications. Some of the SIP OPTIONS requests were verified, and they mostly came from a known botnet called Friendly-Scanner [31] which uses the SIPVicious tools [32] to carry out illicit actions. This sort of behaviour was also seen in the SIP REGISTER requests. Some SNMP requests were also manually verified. It was possible to discover some *get* and *get-next* commands, which support reconnaissance maneuvers to gather information about the infrastructure, facilitating future intrusions on the scanned devices. It is visible in day 8 the accentuated activity of malicious IPs trying to connect to the SSH service. Also, there were many attempts to carry out web communications. The SSH attacks could be caused by ransomeware, since this malware has as goal the remote access of a machine to control its resources, for instance, encrypting its hard drive and later asking for rescue money.

TABLE V: Number of alerts created due to blacklists, organized by source and destination ports.

Source port	Alert
0	125324
135	1647
443	2150
5081	3543
6000	23225
9090	2110
9224	4523
10000	75759
21888 – 48284	54410
50263 – 65535	406591
Total	699282

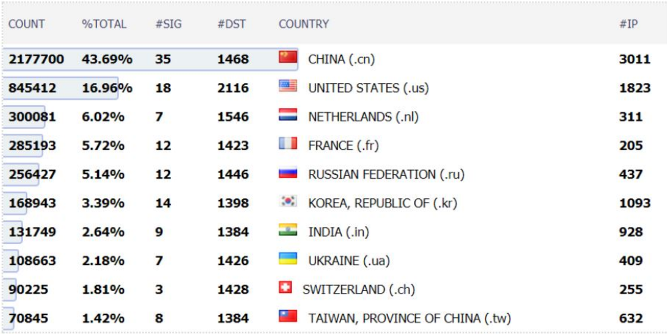
Destination port	Alert
0	125315
22	668036
23	689073
25	42086
53	104006
80	108549
123	16392
443	78799
1433	672640
1900	101632
2323	19761
5060	225279
43526	228788
Total	3080356

C. Blacklisted IPs Analysis

To understand what kind of incidents were raised by the Snort's pre-processor, we analyzed the alerts generated by the blacklist entries. Table V shows the number of alerts taking into account the source and destination ports (and therefore the associated services).

Source ports (first two columns of the table) are mostly dynamic (last row), others are not assigned (last row but one), and the remaining are used by specific services. For example, port 0 is sometimes maliciously employed to gather information although it is a forbidden port [33], port 6000 is a known port used by the X11 system, and port 10000 is utilized by NDMP. Both ports 6000 and 10000 are also used by some Trojans [34], therefore this fact can justify why they appear together with blacklisted IPs. Moreover, the not assigned ports might have been used by a botnet.

The destination ports (last two columns of the table) show that the most accessed services by blacklisted IPs were Telnet (23), Microsoft SQL server (1433), and SSH (22), with a number of incidents very close to each other. This means that cybercriminals attempted to connect remotely some UL machines through the (insecure) telnet protocol and the (secure) SSH protocol, and to access the institution databases, for instance by executing SQL injection attacks. Curiously, the forth most accessed port (43526) is assigned dynamically.



COUNT	%TOTAL	#SIG	#DST	COUNTRY	#IP
2177700	43.69%	35	1468	CHINA (.cn)	3011
845412	16.96%	18	2116	UNITED STATES (.us)	1823
300081	6.02%	7	1546	NETHERLANDS (.nl)	311
285193	5.72%	12	1423	FRANCE (.fr)	205
256427	5.14%	12	1446	RUSSIAN FEDERATION (.ru)	437
168943	3.39%	14	1398	KOREA, REPUBLIC OF (.kr)	1093
131749	2.64%	9	1384	INDIA (.in)	928
108663	2.18%	7	1426	UKRAINE (.ua)	409
90225	1.81%	3	1428	SWITZERLAND (.ch)	255
70845	1.42%	8	1384	TAIWAN, PROVINCE OF CHINA (.tw)	632

Fig. 7: Top 10 countries that produce the most alerts.

This could indicate that a group of machines might have been compromised and are communicating with a specific port to blacklisted IPs. The remaining most accessed ports were VoIP (5060), HTTP (80), DNS (53), SSDP (1900), and HTTPS (443). These results give evidence that several of the IPs tagged as malicious (blacklisted) could be blocked at any institution boarder firewall, as they attempt to access reserved services such as Telnet.

Finally, the Fig. 7 gives an overview of the 10 countries most involved in the incidents, where the top 3 rank is: China, USA, and Netherlands. These results confirm the statistics included in Akamai's 2017 report [5].

Based on the results discussed and analyzed in Sections V-B and V-C, the questions 1 to 3 have a positive answer.

VI. Conclusion

The paper presents an approach to improve the detection capabilities of IDS by resorting to threat intelligence data gathered from OSINT feeds. Our approach automatically processes OSINT data, aggregating and correlating it in order to generate IoAs. Afterwards, these IoAs are used to build IDS rules and blacklist, which are then installed in the IDS. We implemented the `IDSoSint` system and evaluated the approach with production traffic from a few links of the UL. The results show that OSINT data is useful to generate new forms of representing knowledge of threat intelligence and can be applied in defence mechanisms. `IDSoSint` is able to detect illicit activities by employing the generated knowledge, alerting for problems such as botnet communications and remote access applications.

Acknowledgements. This work was partially supported by the EC through funding of the H2020 DiSIEM project (H2020-700692), and by the LASIGE Research Unit (UID/CEC/00408/2013). We warmly thank CNCS for availability on this work.

References

[1] S. Morgan, "Cyber crime costs projected to reach $2 trillion by 2019," https://www.forbes.com/sites/stevemorgan/2016/01/17/cyber-crime-costs-projected-to-reach-2-trillion-by-2019/.

[2] I. Society, "Global internet report 2016," https://www.internetsociety.org/globalinternetreport/2016/.

[3] P. Bächer, T. Holz, M. Kötter, and G. Wicherski, "Know your enemy: Tracking botnets," https://www.honeynet.org/book/export/html/50.

[4] Akamai, "[State of Internet] / Security. Q2 2016 Report," Akamai, Tech. Rep., 2016.

[5] ——, "[State of Internet] / Security. Q4 2017 Report," Akamai, Tech. Rep., 2017.

[6] statista, "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions)," https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[7] SearchSecurity, "Details emerging on dyn dns ddos attack, mirai iot botnet," http://searchsecurity.techtarget.com/news/450401962/Details-emerging-on-Dyn-DNS-DDoS-attack-Mirai-IoT-botnet.

[8] Y. Zeng, X. Hu, and K. G. Shin, "Detection of botnets using combined host-and network-level information," in *IEEE/IFIP International Conference on Dependable Systems & Networks*, 2010, pp. 291–300.

[9] Z. Zhu and D. T., "Featuresmith: Automatically engineering features for malware detection by mining the security literature," in *In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 767–778.

[10] S. Silva, R. Silva, R. Pinto, and R. Salles, "Botnets: A survey," *Computer Networks*, vol. 57(2), no. 2, pp. 378–403, 2013.

[11] G. Bruneau, "The history and evolution of intrusion detection," SANS Institute, Tech. Rep., 2001.

[12] B. Li, J. Springer, G. Bebis, and M. H. Gunes, "A survey of network flow applications," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 567–581, 2013.

[13] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *Communications Surveys & Tutorials, IEEE*, vol. 12, no. 3, pp. 343–356, 2010.

[14] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A flow-based SSH intrusion detection system," in *IFIP International Conference on Autonomous Infrastructure, Management and Security*, 2012, pp. 86–97.

[15] A.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *IEEE/IFIP Network Operations and Management Symposium*, vol. 1. IEEE, 2004, pp. 599–612.

[16] T. Dubendorfer, A. Wagner, and B. Plattner, "A framework for real-time worm attack detection and backbone monitoring," in *1st IEEE International Workshop on Critical Infrastructure Protection*, 2005.

[17] A. H. Haeberlen and A. Aviv, "Challenges in experimenting with botnet detection systems," in *In Proceedings of the 4th Conference on Cyber Security Experimentation and Test*, 2011, pp. 6–15.

[18] I. S. McAfee, "Indicators of attack (ioa)," Intel Security McAfee, Tech. Rep., 2014.

[19] N. Gamer, "The state of botnets in late 2015 and early 2016," http://blog.trendmicro.com/the-state-of-botnets-in-late-2015-and-early-2016/.

[20] J. M. Butle, "Finding hidden threats by decrypting ssl," SANS Institute, Tech. Rep., 2013.

[21] M. Rouse, "Metamorphic and polymorphic malware," http://searchsecurity.techtarget.com/definition/metamorphic-and-polymorphic-malware.

[22] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 12, pp. 343–356, 2010.

[23] C. Wagner, A. Dulaunoy, G. Wagener, and A. Iklody, "Misp: The design and implementation of a collaborative threat intelligence sharing platform," in *In Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security*, 2016, pp. 49–56.

[24] N. Lord, "What are indicators of compromise?" https://digitalguardian.com/blog/what-are-indicators-compromise.

[25] n0where, "Automate incident handling process: Intelmq," https://n0where.net/automate-incident-handling-process-intelmq/.

[26] Shirkdog, "Pulledpork," https://github.com/shirkdog/pulledpork.

[27] Snort, "Snort," https://www.snort.org.

[28] OpenDNS, ""phishtank," https://www.phishtank.com.

[29] CINScore, "The cins score," http://cinsscore.com.

[30] MDL, "Malware domain list," https://www.malwaredomainlist.com.

[31] M. Kezys, "Sip attack: Friendly-scanner," http://blog.kolmisoft.com/sip-attack-friendly-scanner/.

[32] S. Gauci, "Sipvicious," https://github.com/EnableSecurity/sipvicious.

[33] B. Mitchell, "Port 0 in tcp and udp," https://www.lifewire.com/port-0-in-tcp-and-udp-818145.

[34] Speedguide, ""speedguide"," https://www.speedguide.net/port.php?port=6000.