Computer Networks 180 (2020) 107391

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Detection of zero-day attacks: An unsupervised port-based approach☆

Agathe Blaise[a,b,*], Mathieu Bouet[a], Vania Conan[a], Stefano Secci[c]

[a] Thales, Gennevillers, France
[b] Sorbonne Université, CNRS, Paris, LIP6 France
[c] Cnam, Cedric, 75003 Paris, France

A B S T R A C T

Last years have witnessed more and more DDoS attacks towards high-profile websites, as the Mirai botnet attack on September 2016, or more recently the memcached attack on March 2018, this time with no botnet required. These two outbreaks were not detected nor mitigated during their spreading, but only at the time they happened. Such attacks are generally preceded by several stages, including infection of hosts or device fingerprinting; being able to capture this activity would allow their early detection. In this paper, we propose a technique for the early detection of emerging botnets and newly exploited vulnerabilities, which consists in (i) splitting the detection process over different network segments and retaining only distributed anomalies, (ii) monitoring at the port-level, with a simple yet efficient change-detection algorithm based on a modified Z-score measure. We argue how our technique, named Split-and-Merge, can ensure the detection of large-scale zero-day attacks and drastically reduce false positives. We apply the method on two datasets: the MAWI dataset, which provides daily traffic traces of a transpacific backbone link, and the UCSD Network Telescope dataset which contains unsolicited traffic mainly coming from botnet scans. The assumption of a normal distribution – for which the Z-score computation makes sense – is verified through empirical measures. We also show how the solution generates very few alerts; an extensive evaluation on the last three years allows identifying major attacks (including Mirai and memcached) that current Intrusion Detection Systems (IDSs) have not seen. Finally, we classify detected known and unknown anomalies to give additional insights about them.

1. Introduction

Back in September 2016, the Mirai botnet [2] struck the internet with a massive distributed denial of service (DDoS) attack. During several months, it spread slowly and reunited nearby 50,000 bots distributed over various parts of the internet, without being noticed. More recently, a record-breaking DDoS attack hit Github on February 2018 with a new amplification attack vector: UDP-based memcached traffic [1]. The caching system is supposed to be used internally, but sometimes runs on servers exposed without any authentication protection; several days later, most memcached servers have been patched, making the attack not efficient anymore [50]. Actually, malwares targeting Internet-of-Things (IoT) devices and misconfigured servers are responsible for many Distributed Denial-of-Service (DDoS) attacks [53]. Detecting these botnets and exploited vulnerabilities during their spreading could avoid many harms. There is thus an urgent need to detect this kind of threats as soon as possible, and current anomaly detection tools appear deficient in this respect.

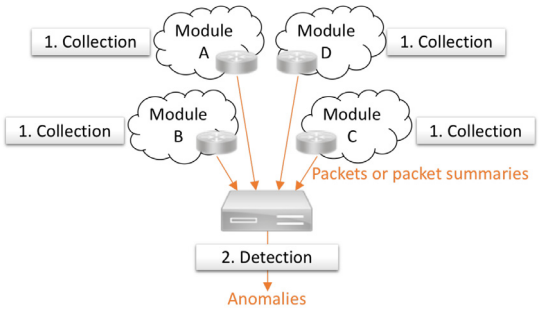
Ensuring cyber-security in networks, Intrusion Detection Systems (IDSs) monitor network traffic for malicious activities and related threats. However, as a matter of fact most botnets go under the radars for three reasons: (i) Current IDSs work at different traffic granularities, e.g., flow, host or packet. However, they miss global changes on application ports that are involved during the propagation of botnets. Ports can be scanned to fingerprint the target machine, to exploit known vulnerabilities, or to communicate with a Command-and-Control (C&C) server [6]. The sole common denominator for a botnet coming from very distinct sources and targeting lots of hosts is the port it scans. However, an IDS working on IP addresses would be unable to notice the anomalous port. (ii) Most IDSs work on small variations of traffic, generally using time-sliding windows of several seconds. Therefore, they cannot build long-term profiles per port and detect major changes in their usage. (iii) IDSs are usually deployed at a single point in the network, while ISP-scale attacks are only visible by looking at a holistic view of a wide area network.

In this paper, we propose an anomaly detection technique that spots main changes in the usage of a single port to identify botnets. Intuitively, the most obvious way to identify it is to observe a sudden rise in traffic towards a port. However, this may not be sufficient as it can be a well-known vulnerable port, already massively scanned. For example, before the Mirai attack, many TCP SYN scans targeted the Telnet port whose vulnerabilities were already known and exploited. Then, when the Mirai attack was actually hitting, one could not observe an increase in the number of scans targeting this port. Our goal is to detect early stealthy changes in the scans behavior, as an increase in the number of distinct attackers (i.e., source IP addresses) or an increase in port spoofing, to then spot them as unknown botnets or newly exploited vulnerabilities, even on ports already scanned before.

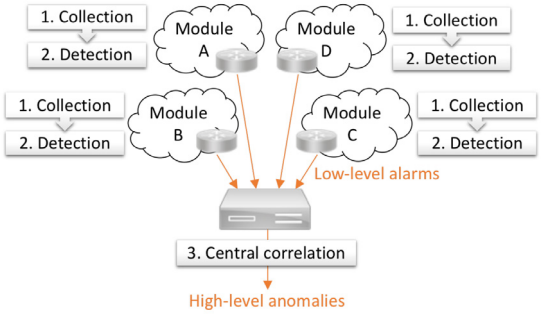
☆ A preliminary version of this paper has been presented at IFIP/IEEE IM 2019 [7].

* Corresponding author.
E-mail addresses: agathe.blaise@lip6.fr (A. Blaise), mathieu.bouet@thalesgroup.com (M. Bouet), vania.conan@thalesgroup.com (V. Conan), stefano.secci@cnam.fr (S. Secci).

https://doi.org/10.1016/j.comnet.2020.107391
Received 27 January 2020; Received in revised form 28 April 2020; Accepted 29 June 2020



(a) Local collection and central detection.



(b) Centralized CIDS: local collection and detection, and central correlation.

Fig. 1. Two possible approaches for large-scale IDS.

In our method, we use features representing particular port usages; large packets batches picked at a frequency of several days enable to profile the evolution of features over time, then statistical measures can spot anomalies in the features time-series. A port-based approach may generate a large number of alarms, as for instance each ephemeral port used in a arbitrary manner would produce an anomaly. Therefore, we adopt a collaborative scheme to ensure that changes in one port are distributed and are not due to random or localized traffic variations. In our approach, called Split-and-Merge, local detection modules, geographically split in the network, collect traffic and send anomalies to a central controller in charge of aggregating them, like a Collaborative IDS (CIDS) [45] would do (Fig. 1b). The number of false positives can so be significantly reduced as only anomalies detected in several places are taken into consideration. Our contributions differ from existing botnet's detection approaches given the following reasons. First, it targets long-term anomaly detection enabling to detect major changes in the use of ports, and thus underlying botnets. As a matter of fact, current approaches for botnets' detection [3,9,12,15,19,22,24,26,38,44–46,48,54] focus on real-time intrusions and may miss stealthy changes visible at a several days scale. Second, it focuses on destination ports, compared to other approaches [3,9,12,15,19,22,24,44–46,48,54] aggregating packets per flow or IP address, which thus are not able to detect scans coming from very distinct source IP addresses and targeting a large variety of destination IP addresses. Third, it leverages on several detection modules geographically split in the network, in order to reduce the number of false positives. While most IDSs are localized at a single vantage point [9,15,18,20,22,24,26,38,46,48,54], we explore a collaborative IDS approach only marginally adopted at the state of the art [3,12,19,44,45]. Finally, its features are computed over diversity indices, packet size and TCP flags, using a change-point detection system, which is not done in [3,9,12,15,18,20,19,22,24,26,38,44–46,54].

For our evaluation, we use the MAWI dataset [31] which provides daily traces of a transpacific backbone link. The dataset is restricted to a single Internet Service Provider (ISP), hence corresponds to what could be used at the ISP-level. Differently than the common approach that uses real traces to generate background traffic, we use the MAWI traces as they are, with the aim at detecting real attacks from it, providing a better knowledge of the dataset at the same time. We also use the UCSD Network Telescope dataset which consists of a globally routed, but lightly utilized /8 network prefix. Inbound traffic to non-existent machines is unsolicited and results from a wide range of events, including misconfiguration, scanning of address space by attackers or malware looking for vulnerable targets and backscatter from randomly spoofed denial-of-service attacks. This way, we are able to compare the anomalies found in both datasets. We present the intrusion detection results against known attacks arisen the last three years, not detected by the MAWILab detection algorithm [16], and we show that we can detect some unknown anomalies as well; in order to classify anomalies, we observe the simultaneous evolutions of features. We experimentally show that our algorithm greatly reduces the number of false positives compared to a single IDS running on the whole dataset. For the sake of reproducibility and further research, our source code is publicly available at [21].

This paper is organized as follows. Section 2 surveys the related work. Section 3 presents our solution detecting distributed changes in port usages, along with the analysis of its complexity. In Section 4, we present results from numerical evaluation, highlighting the benefits Split-and-Merge can grant in terms of false detection rate and detection accuracy, and also proposing a classification of the noticed anomalies. Finally, Section 6 concludes the paper.

2. Related work

In this section, we present intrusion detection methodologies and review related work.

2.1. Intrusion detection methodologies

Many algorithms are proposed in the literature for network intrusion detection [5]. We can classify them in two main families: knowledge-based and anomaly-based techniques.

Knowledge-based (or signature-based) solutions such as Snort [13] and Bro [39] rely on a signature database to find attacks that match given patterns, such as malicious byte sequences or known malware signatures. Up to now, most companies rely on signature-based IDSs as they are expressive and understandable by network administrators. Nevertheless, they are not able to detect zero-day attacks, i.e., attacks exploiting unknown vulnerabilities, for which no patch is available [28]

Anomaly-based approaches attempt to detect zero-day attacks, in addition to known ones. They model the normal network traffic and qualify an anomaly as a significant deviation from it, with statistical or machine learning techniques. In such a case, we talk about anomalies rather than attacks. BotSniffer [22] utilizes statistical methods to detect C&C botnets, by observing coincident behaviors among hosts, like messages to servers, network scan or spam. The authors in [27] observe changes in feature distributions to identify anomalies. Entropy and/or volume are such metrics used for this purpose.

Machine learning techniques can be classified in three families:

1. *Supervised* ones learn from a labelled dataset what constitutes either normal traffic or attacks – there exists different techniques such as SVM-based classifiers, rule-based classifiers and ensemble-learning detectors [46].
2. *Unsupervised* approaches learn by themselves what is normal or abnormal – among them, MAWILab [18,20] finds anomalies by combining detectors that operate at different traffic granularities (the results against the MAWI dataset are in [18,20]); numerous works compare themselves to MAWILab, as for instance change-detection techniques [9,48] (defining an anomaly as a sudden change compared to a model), and ORUNADA [15] (relying on a discrete time-sliding window to continuously update the feature space and cluster events).

3. *Hybrid* approaches benefit from only a small part of labelled traffic, meant to be enough to learn from, as proposed in [29].

2.2. Large-scale intrusion detection

Coordinated attacks arise in multiple networks simultaneously and include large-scale stealthy scans, worm outbreaks and DDoS attacks [57]. Traditional IDSs tend to fail at detecting these attacks as they commonly monitor only a limited portion of network. Large-scale IDSs, instead, have a global view over the network, and can better scale by distributing the computational load between several detection agents. Two large-scale IDS approaches can be identified.

The first IDS approach consists in distributing flow collectors in different subnetworks and in running a central detection engine against aggregated data, as shown in Fig. 1a. Raw packets are transmitted from the flow collectors to the detection engine [44]. Solutions exist to avoid the collection traffic overhead, as done by Jaal [3], which creates and sends concise packets summaries to the detector - with Jaal, one reaches a 35% bandwidth overhead to get an acceptable true positive rate, which is still important.

The second IDS approach consists in the already mentioned CIDS, that is a two-level anomaly detection system where monitors are physically split in the network to perform local detection. They generate low-level alerts then aggregated to produce a high-level intrusion report. Three types of CIDSs exist depending on the communication architecture:

1. *Centralized* CIDSs are composed of several monitors that transmit the alerts to a central correlation engine, as illustrated in Fig. 1b.
2. *Hierarchical* CIDSs use a multistage structure of monitors to achieve an increasingly higher alert aggregation until the alerts reach the top correlation engine.
3. *Distributed* CIDSs share the detection and correlation tasks between all monitors. This approach can be set up by a peer-to-peer network.

For instance, [45] presents a centralized CIDS framework composed of IDS clusters implementing both the detection and the correlation; Snort signatures are therein used to detect known attacks, while an unsupervised learning algorithm detects unknown attacks. [19] proposes a sort of distributed CIDS, composed of Intrusion Prevention Systems forming rings around the hosts to protect, in order to collaborate and forward the traffic adaptively depending on their findings.

Inherent in the CIDSs, alert correlation algorithms can be divided into three categories [32]: (i) similarity-based algorithms, which compute the similarity between an alert and a cluster of alerts, and based on the result either merge it with the cluster or create a new one; (ii) knowledge-based algorithms, which rely on a database of attacks definitions; (iii) probabilistic algorithms, which use similar statistical attributes to correlate attacks.

2.3. Botnet detection

In the past years, several novel algorithms for botnet detection have been proposed, which can be classified into packet/flow-based ones and graph-based ones. Among them, [24] compares the performances of four different approaches: Snort, BotHunter and two data-mining based system ones, either based on the packet header/payload or on flows. They run their algorithm on public datasets, including the Conficker dataset from CAIDA, the ISOT-UVic dataset, and Zeus botnet datasets from Snort, NETRESEC and NIMS. As a result, they get detection rates approaching up to 100%. BotMark [54] exploits both statistical flow-based traffic features and graph-based features to build their detection model, then considers similarity and stability between flows as measurements in the detection. They test their algorithm by simulating five newly propagated botnets, including Mirai, Black energy, Zeus, Athena and Ares, and achieve 99.94% in terms of detection accuracy. In [12], the authors create a complete characterization of the behavior of legitimate hosts that can be used to discover previously unseen botnet traffic. They employ the ISCX botnet dataset, a publicly available dataset composed of various IRC, P2P and HTTP based botnets. They find that their framework can detect bots in a network with 1.00 TPR and 0.082 FPR.

It is worth noting that the aforementioned algorithms perform their analysis at network-level, on traffic generated by botnets. Their objective is to distinguish between benign hosts and bots, to then draw a confusion matrix and evaluate their classifier. Contrary to these algorithms, our analysis runs at an Internet carrier link level (or on data collected into a darknet for the UCSD dataset). It focuses on analyzing the current trends in Internet traffic over several years, including trends in terms of botnets. It also enables to spot vulnerabilities exploited following their disclosure and targeted attacks not part of a botnet. Therefore, (i) we do not benefit from labelled flows and ground-truth, (ii) we do not limit our goal to botnet detection, (iii) we cover a three-year period compared to those algorithms that are limited to several hours only. For these reasons it would not be consistent to directly compare Split-and-Merge to those algorithms.

2.4. Our contribution

Let us position our contribution with respect to the described related work. To detect port-based anomalies, we benefit from an unsupervised anomaly detection algorithm, that does not require labeled data. We leverage on the CIDS principles to build our system, and in particular centralized CIDS characteristics. In terms of alert correlation, we attempt at simplifying the search space using application ports, and more precisely destination ports. Up to our knowledge, our direction of using centralized CIDS scheme with port-centric detection is novel. Aggregating alerts based on destination ports as we propose can strongly ease the aggregation challenge, avoiding too complex algorithms for that purpose.

A few works specifically focus on port-based detection but they do not apply to CIDS. In [6], the authors propose a survey of the current methods to detect port scans[38]. aims to show the correlation between port scans and attacks. [26] examines the period during the release of a zero-day attack and its patching. Also, [26,38] analyze port-usage but they do not use destination ports as primary key. Actually, this last setting generates a high number of false positives, which can be mitigated by CIDS as we are doing.

About our feature choice, we leverage on diversity indices, defined as the proportion of unique elements in a set. In the literature, the number of unique source IP addresses and unique active /24 blocks are used to detect Internet outages [23] and large-scale spoofing [14]. Compared to these works, we compute the proportion (instead of the count) of unique source IP addresses, but also of unique destination IP addresses (to identify large-scale scan techniques used by attackers) and unique source port numbers (to detect spoofed ports), each of them computed on a per destination port basis. In addition, we use features computed from TCP flags and packet size, and we integrate them into a change-point detection system, i.e., a system that identifies when the probability distribution of a feature time series changes.

We describe the detection of new vulnerabilities and emerging botnets, using a port-based change detection algorithm in a preliminary conference paper [7]. In this paper, we extend the original article in regards to several aspects. First, we perform a more extensive evaluation (over three years instead of 6 months in the first version) and we include an additional dataset, the UCSD Telescope one, to compare the anomalies that we found from both datasets. For this purpose, we draw a retrospective analysis of major botnets and attacks arisen these last years. We also validate our hypothesis regarding the normal distribution, through empirical measures on the MAWI dataset, and analyze the detection accuracy looking at how the parameters and the features impact the results. We provide a more detailed characterization of the attacks by analyzing conjointly the features evolution, raising 8 differ-

ent kinds of anomalies. Finally, we include a complexity analysis of our algorithm in terms of space and time complexity, and evaluate the time required to run the detection process.

3. Split-and-Merge port-centric network anomaly detection

We present our anomaly detection proposal, detailing the reference CIDS architecture and the features design.

3.1. Rationale

We already anticipated some of our key modeling choices: we aggregate traces based on destination ports, in a distributed CIDS setting, and target to design features minimizing the degree of arbitrarity in their choice and interpretation. Our objective is to model the usage of each port, by computing features each time the same day at the same daytime slot. The features characterize the port usage, e.g., if it is mainly targeted by port scan or not, if the hosts are numerous or not, etc. We work on a limited time window over a day, which we assume to represent port usages this day.

In our reference distributed CIDS setting, several *detection module* agents run on different subnetworks so that they can capture subnetwork peculiarities and cover the CIDS network context completely. Based on the time evolution of the features of a port, the detection modules detect anomalies and report them to a *correlation module*. Hereafter we detail the different steps of our detection module logic, as well as the anomaly aggregation logic of the correlation module. At each daytime slot, every detection module performs several tasks in a row (each task is then further detailed in the following subsections).

3.1.1. Data collection

First, the detection module collects packets in its scope in a single group of N_{batch} elements, and stores packet attributes in lightweight Collection Tables (CTs). For each incoming packet, it identifies the destination port and updates four key-value CTs and a counter for the given port:

- CT1: unique source IP addresses;
- CT2: unique destination IP addresses;
- CT3: unique source ports;
- CT4: unique size of packets;
- Counter: number of SYN packets.

Each entry in one CT (e.g., a source IP address in CT1) is associated with a counter of occurrences.

3.1.2. Features computation

After data collection, a filter is applied on CTs so that only the ports with at least $N_{\min}$ packets stored are kept to be analyzed. For every remaining destination port, the detection module computes some features based on CTs and updates the Features Table (FT) with new values. The FT constantly contains N_{days} entries (we use one day per week in our tests) so that for every new capture, the former value is deleted and the new one added.

3.1.3. Anomaly detection

Lastly, the local detection module analyzes the port-specific features time-series over N_{days} in order to detect an anomaly with a change-detection algorithm. When an anomaly is spotted, based on a warning threshold T_i on a given feature i, an alert is created and transmitted to the central correlation module. The collection and detection parameters resumed in Table 1 are to be customized. At the end of the detection process, the correlation module aggregates the alerts received from all detection modules. It is then able to deduce and qualify an attack by noticing the distributed alerts.

Table 1
Parameter notations.

Notation	Definition
N_{batch}	Number of packets collected per day
$N_{\min}$	Minimum number of packets per port
N_{days}	Number of days in the sliding window
T_i	Threshold to spot an anomaly for feature i

Table 2
Features definition.

Feature	Computed from	Description
srcDivIndex	CT1	% of unique source IP addresses
destDivIndex	CT2	% of unique destination IP addresses
portDivIndex	CT3	% of unique source ports
meanSize	CT4	Mean packets size
stdSize	CT4	Standard deviation of packets sizes
perSYN	Counter	% of SYN packets
nbPackets	Any CT	Number of packets

3.2. Features design

To observe an anomaly on a port, looking at the number of packets over time is not sufficient. Indeed, subtle changes in the nature of packets can happen on a port already massively scanned. Therefore, we need to design significant features.

Our features choice is resumed in Table 2. *nbPackets* represents the number of packets stored for this port and enables to see if a port is suddenly massively used. *srcDivIndex* and *destDivIndex* highlight significant variations in the proportion of unique source and destination IP addresses. An increase in *srcDivIndex* may be an attack perpetrated by bots, while its decrease can indicate an attack led by only a few actors. A rise in *destDivIndex* may represent a large number of victims, as a botnet scanning random IP addresses or the whole IPv4 range would cause. *portDivIndex* reflects the diversity in source ports, its diminution may represent the usage of a spoofed port. A variation in the *meanSize* feature suggests a change in packets nature, like crafted packets sent by bots. A variation in the *stdSize* feature can be caused by a change in packets nature as well, and in addition is not easy to fool for an attacker: if it increases, the diversity among packets is higher, so probably there are suddenly both crafted and regular packets; if it decreases, the diversity among packets is lower, hence the traffic more specific. This can be caused by a malicious software that kills other processes bound to the same port. Finally, a variation in *perSYN* implies an increase or decrease in port scan. Therefore each port p at a given day is characterized by the set of features computed from CTs, shown in Table 2.

We denote the time-series of feature i containing N days (i.e., N_{days}) for port p as $f^p_{i,N} = (x^p_{i,1}, \dots, x^p_{i,j}, \dots, x^p_{i,N})$, with $x^p_{i,j}$ being the value of feature i for port p on day j. Features are computed at a given frequency, set to once every week in the following simulations (in particular the same day at the same daytime slot, in order not to be influenced by weekly or daily variations).

Algorithm 1 below shows how to update the $FT\{ports^* features^* N_{days}\}$ by computing features by port from packet attributes found in CTs.

Algorithm 1 updateFT(CTs, FT, $N_{\min}$).

```
Delete first column of FT and shift others
for all port p ∈ ports do
    // Check condition on the number of packets
    if length(CT1[p]) > N_min then
        for all att ∈ attributes do
            feature f ← relativeMetric(att) // 1 or 2 features per attribute (e.g., mean and std for packet size)
            FT[p][f][currentDay] = CTf[p].apply(f)
return FT
```

3.3. Local anomaly detection

Assuming a feature is more or less likely to vary (standard deviation) depending on its type, and usually around the same (mean) value, the normal distribution logically quite fits as its distribution. The validity of this assumption is assessed later in Section 4.2. We model the time-series $f^p_{i,N} = (x^p_{i,1}, \dots, x^p_{i,N})$ over N days as a normal distribution $\mathcal{N}(\mu^p, \sigma^{p^2})$ of mean μ^p and standard deviation σ^p such that:

$$\mu^p = \sum_{j=1}^{N} x^p_{i,j} \text{ and } \sigma^p = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x^p_{i,j} - \mu^p)^2}. \quad (1)$$

The *Z-score* is a well-known simple statistical-based algorithm, commonly used to automatically detect a change in time-series. More precisely, it is the measure of how many standard deviations below or above the mean a data point is. Basically, a Z-score equal to zero means that the data point is equal to the mean and the larger the Z-score, the more unusual the value. For the given time-series $f^p_{i,N} = (x^p_{i,1}, \dots, x^p_{i,N})$ approximated by a normal distribution $\mathcal{N}(\mu^p, \sigma^{p^2})$, the Z-score of the new value $x^p_{i,N+1}$ of feature i at time $N+1$ is computed as follows:

$$Z^p_{i,N+1} = \frac{x^p_{i,N+1} - \mu^p}{\sigma^p}. \quad (2)$$

However, the Z-score is computed from the mean, a metric influenced by outliers and especially extreme values. Alternatively, the *modified Z-score* uses the median and the median absolute deviation (MAD) from the median, instead of the classical mean and standard deviation respectively, which makes it outlier-resistant [25].

Given the time-series median $\widetilde{f}^p_{i,N}$, the modified Z-score $M^p_{i,N+1}$ of the new value $x^p_{i,N+1}$ of feature i at time $N+1$ is computed as:

$$M^p_{i,N+1} = \frac{0.6745 \cdot (x^p_{i,N+1} - \widetilde{f}^p_{i,N})}{median(|x^p_{i,N+1} - \widetilde{f}^p_{i,N}|)} \quad (3)$$

An anomaly is detected if the absolute value of the modified Z-score exceeds a threshold T_i. For all i, we adopt a threshold value of 3.5 as recommended in [25]. Algorithm 2 presents the anomaly detection process taking place in each local detection module, to detect anomalies from features time-series found in FT.

Algorithm 2 runDetection(FT, N_{days}).

```
median ← 0
mad ← 0 // median absolute deviation
mZ ← 0 // modified Z-score
list anomalies
for all port p ∈ FT.ports do
    for all feature f ∈ FT.features do
        series ← FT[p][f]
        orderedSeries ← quickSort(series)
        median ← orderedSeries[(N_days+1)/2]
        sum ← 0
        for all value ∈ series do
            sum ← sum + |value - median|
        mad ← sum / N_days
        mZ ← (0.6745 · (series[currentDay] − median)) / mad
        if mZ > 3.5 then anomalies.add({p, f})
return anomalies
```

The modified Z-score is used to identify anomalies on all features, except for *nbPackets*: it is only used to spot emerging ports, i.e., ports that were not in use before. That is, an anomaly is spotted if at least a given number of packets $N_{\min}$ is collected on one port for the first time in N_{days}, so that $x^p_{i,N+1} \geq N_{\min}$ and $x^p_{i,j} < N_{\min}$ for each $j \epsilon [1, N]$.

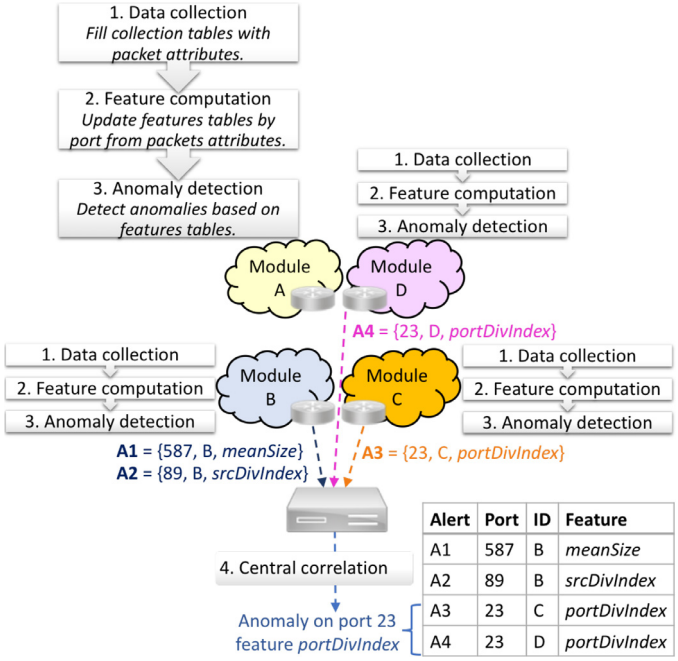


Fig. 2. Architecture example. Local modules run at different points in the network and send alerts to the central correlation module. The controller will then be able to keep only distributed ones. Here it spots an anomaly on port 23 for feature *portDivIndex*, coming from two different places.

Once all features of all ports are analyzed, the detection module sends the content of the anomalies to the correlation module as alerts. For each alert, the module specifies its ID m, the anomalous port p, the involved feature i, the time-series $f^p_{i,N}$ and the new anomalous value $x^p_{i,N+1}$. An alert is so defined by a 5-tuple $\{p, m, i, f^p_{i,N}, x^p_{i,N+1}\}$. For example, in Fig. 2, the detection module B notices an anomaly on port 89 for feature *srcDivIndex*. It also provides the time-series of feature $f^p_{i,N}$ and $x^p_{i,N+1}$, though not written on the Figure.

3.4. Central correlation

The correlation module receives the low-level alerts from all detection modules. The distinction between localized (noticed in one subnet) and distributed (noticed in several subnets) alerts is made here. As we are searching for distributed attacks, the correlation module groups the low-level alerts to keep only the ones reported by at least k subnets; we set $k = 2$ in this work. In the example of Fig. 2, several detection modules send alerts to the correlation module; among them, two subnetworks report a change in the *portDivIndex* feature on port 23. Hence the correlation module induces an anomaly on this port. It is even better if similar anomalies have been noticed on the same port for several features.

We define the *Anomaly Score* (*AS*) as the number of anomalies noticed for one port by all monitors and for all features; e.g., if for one port a monitor detects anomalies on two features and another on six features, the *AS* is 8. The correlation module is able to compute the *AS* after having received alerts from all monitors during the same time slot. When it identifies top-level anomalies, it warns all detection modules about the anomalous ports. Thus they are able to analyze these ports as a priority next time. Ad-hoc actions can also be taken, as a function of the programmability of the local network, such as port blocking, mirroring, deep-packet-inspection, for the sake of reporting in a possible further detailed analysis.

4. Evaluation

In this section, we evaluate the performance of the Split-and-Merge detection process using real traffic traces. First, we aim to validate our assumption that the feature data is normally distributed around its median. We also analyze the results to adequately determine the features and parameters. Finally, we look at the anomalies during the last three years and aim to classify them. The source code used for the detection and evaluation is available in [21].

4.1. Network traffic datasets

The WIDE project provides researchers with daily traces of a transpacific link, named the MAWI archive [31]. Traces are collected between their network and the upstream ISP. Each file contains 15 minutes of traffic flows, captured between 14:00:00 and 14:15:00 local time. This represents usually between 4 and 10 GB of traffic for one file. Before being released, traces are anonymized so that no personal information can be extracted. Specifically, the application data is removed and IP addresses are scrambled with a modified version of *tcpdpriv* following two principles: 1) it is collision-free so that there is a one-to-one mapping between IP addresses before and after anonymization; 2) it is prefix-preserving so that if two IP addresses share k bits before anonymization, the two anonymized IP addresses will also share k bits. This enables to retrieve the subnetworks after anonymization.

In addition, the Center for Applied Internet Data Analysis (CAIDA) provides the UCSD Network Telescope dataset [8]. It consists of a globally routed, but lightly utilized /8 network prefix, that is, 1/256th of the whole IPv4 address space. It contains a few legitimate hosts; inbound traffic to non-existent machines - so called Internet Background Radiation (IBR) - is unsolicited and results from a wide range of events, including misconfiguration, scanning of address space by attackers or malware looking for vulnerable targets, backscatter from randomly spoofed denial-of-service attacks, and the automated spread of malware. CAIDA continuously captures this anomalous traffic discarding the legitimate traffic packets destined to the few reachable IP addresses in this prefix. They provide two types of files: raw data stored into pcap files for the ongoing month, and hourly FlowTuple files for the last 13 years – that we used. Note that this dataset does not contain several subnetworks as the MAWI one does. Thus the number of false positives (i.e., detecting an attack targeting only this subnetwork but not the whole Internet) may be higher. The objective in using this second dataset is to evaluate the efficiency of our method as of detected botnets and, most of all, to compare the anomalies found in each of the datasets and to study the common ones.

Note that by default, we refer to the MAWI dataset when we do not specify which one we use. As the UCSD dataset only contains one vantage point, we do not use it for all experiments, instead we use it to cross-check the anomalies found in the MAWI dataset since there is no ground truth.

4.2. Normal distribution fitting

It is important to empirically assess the validity of a key Split-and-Merge assumption, that is that the features data is expected to be normally distributed around the median. Indeed, we use the modified Z-score to support the detection logic.

It is well known from the state of the art that Internet traffic exhibits a power-law behavior [30] for the packet counts. Among Split and Merge features, we consider the *nbPackets* feature only to characterize ports behaviour (for the ports with sufficient traffic). Moreover, other features represent diversity indices, attributes means and standard deviations.

To assess the assumption that normal distribution is a well fit for the *nbPackets* feature, and that it is better than the power-law distribution, we compute the mean square error (MSE) between the measured and synthetically generated histogram [10], for each tuple of port and feature so that

$$MSE = \frac{1}{N_{bins}} \sum_{b=1}^{N_{bins}} [H^p_{i,N}(b) - \widehat{H^p_{i,N}}(b)]^2 \quad (4)$$

where $H^p_{i,N}$ denotes the normalized histogram of the N days time-series $f^p_{i,N}$ of feature i and port p, $\widehat{H^p_{i,N}}$ is the histogram with matching mean and standard deviation, and N_{bins} is the number of bins in the histograms. The latter is chosen according to Sturges' rule stating that the number of bins K should be equal to $K = 1 + 3.322(\log_{10}(N))$ with N the number of samples. Using this method, we used 4 bins for 10 samples.

Given the several thousands of ports to analyze each day for each feature, the Cumulative Distribution Function (CDF) represents the cumulative probability for one feature to reach a given MSE by taking into account all ports. We plot the empirical CDF of the MSE by considering four different regressions: a normal distribution with matching mean and standard deviation in Fig. 3a, a normal distribution with matching median and median absolute deviation in Fig. 3b, a log-normal distribution with matching mean and standard deviation in Fig. 3c, and a log-normal distribution with matching median and median absolute deviation in Fig. 3d. The reported results are those for 2016 traffic (we observe similar results in 2017 and 2018).

We observe that: (i) the regression using the log-normal distribution gives far worse results than the normal distribution, (ii) for the normal distribution, the regression using the median and the median absolute deviation gives a better approximation than the one with the mean and the standard deviation, (iii) for the normal distribution, all features produce more or less the same MSE.

By using a normal distribution, we found out that the MSE is very low for all features, which is an empirical validation of this assumption.

4.3. Local anomaly detection

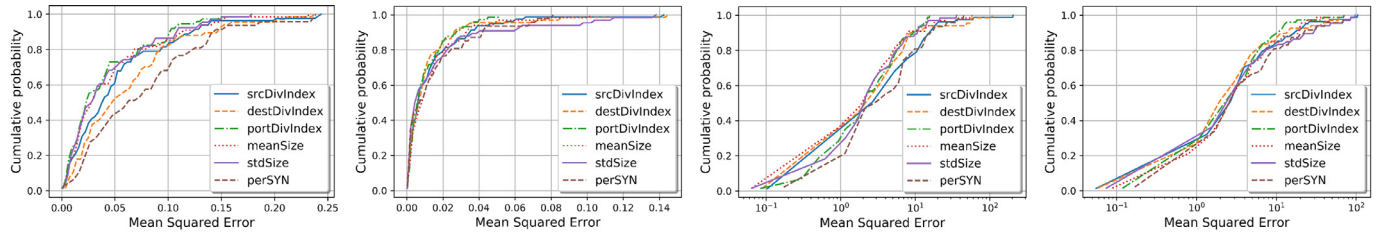
This section gives the outcome of several local detection modules running simultaneously, each of them being situated in a MAWI subnetwork. We pick each Thursday from March 31, to Oct. 20, 2016. Thresholds T_i for an anomaly are all set to 3.5. The minimum number of packets $N_{\min}$ is set to 20. The number of days we chose is $N_{days} = 10$. We will tune these two last values later in Section 4.7.2.

Fig. 4 gives an example of the modified Z-score evolution for the *srcDivIndex* feature on port TCP/3389. On Sept. 29, the absolute value of the modified Z-score is over the threshold for four detection modules situated in different subnetworks, resulting in an anomaly. The subnetwork F contains only a few points because most of the time, there is little (less packets than $N_{\min}$) or no traffic on port 3389 in this subnetwork. The same explanation applies to subnetworks that do not appear at all in the legend.

4.4. Comparison between aggregated and split views

In this experiment, we compute the number of alarms for each feature considering two approaches: (i) an aggregated view where an anomaly is observed considering the traffic from all subnetworks aggregated, (ii) a split view where only distributed anomalies (i.e., seen in at least two subnetworks) are conserved. The results are presented in Table 3 for 2016, while similar findings have been observed in 2017 and 2018. We observe that the number of anomalies – thus the number of false positives – is significantly lower with the split view.

The example of feature *srcDivIndex* is shown in Fig. 5, with the number of anomalies expressed in logarithmic scale. With a split view, we observe that considering only distributed alerts considerably diminishes the number of anomalies to deal with. Indeed, the number of anomalies for a single variation (score of 1) is 100 times higher than for a distributed variation (score of 2), decreasing the number of alerts from 3918 to 66.



(a) Normal distribution based on mean.

(b) Normal distribution based on median.

(c) Log-normal distribution based on mean.

(d) Log-normal distribution based on median.

Fig. 3. Empirical CDF of the MSE between the true distribution and the regression.

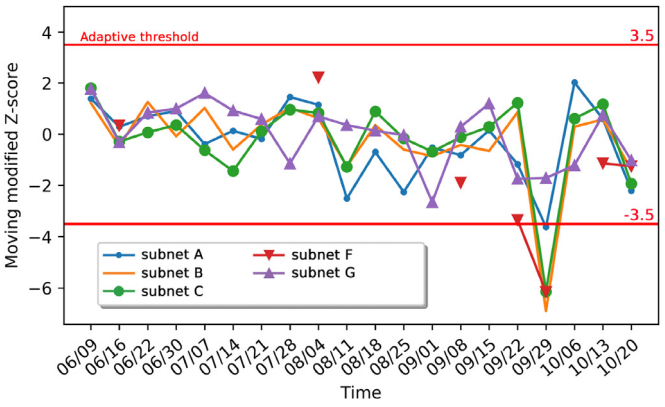


Fig. 4. Evolution of the modified Z-score in 5 subnetworks for feature *srcDivIndex* on port 3389 over time (2016).

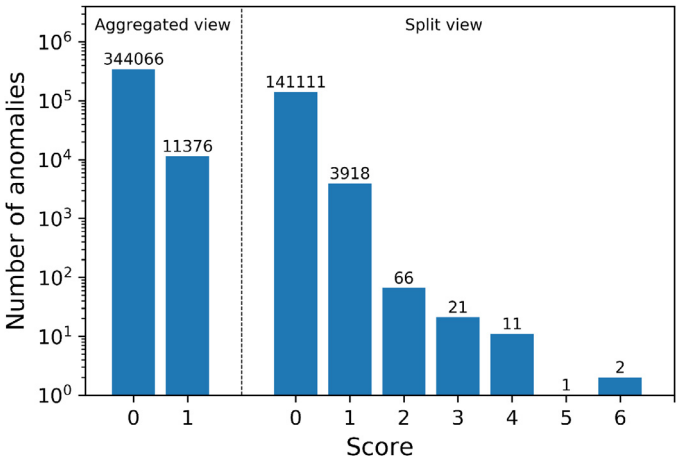


Fig. 5. Number of anomalies for feature *srcDivIndex*. In aggregated view, the score is 1 if there is an anomaly on the whole traffic, else 0. In split view, it is the number of anomalous subnetworks.

4.5. Last years panorama

In this subsection, we launch the anomaly detection process on a large period to examine the type of anomalies we can detect. We describe hereafter the main anomalies arisen these last three years, in the MAWI dataset and the UCSD dataset.

4.5.1. In the MAWI dataset

Figs. 6 a (2016), 6 b (2017) and 6 c (2018) show the number of ports with a given anomaly score each day, highlighting the main anomalies arisen these last years. In all cases, we observe very few alarms each day, which is quite convenient for the network administrator, as too numerous alerts is considered as one reason why IDS are underused. Furthermore, none of these anomalies have been detected by MAWILab. Also, we tag events observed in both datasets with a red frame in Fig. 6. Note also that lots of ports score 5 anomalies, as a significant variation on one port in one subnetwork generates simultaneous alerts for all features (except for the feature SYN that gives poor results as shown later in Section 4.7.1), i.e., 5 alerts. We therefore describe the main anomalies, whose anomaly score is the highest, and we also indicate if we retrieved these anomalies in the UCSD dataset.

2016 period. Eight noticeable scores appear in Fig. 6b depicting this first period.

Table 3
Number of anomalies for both approaches (2016).

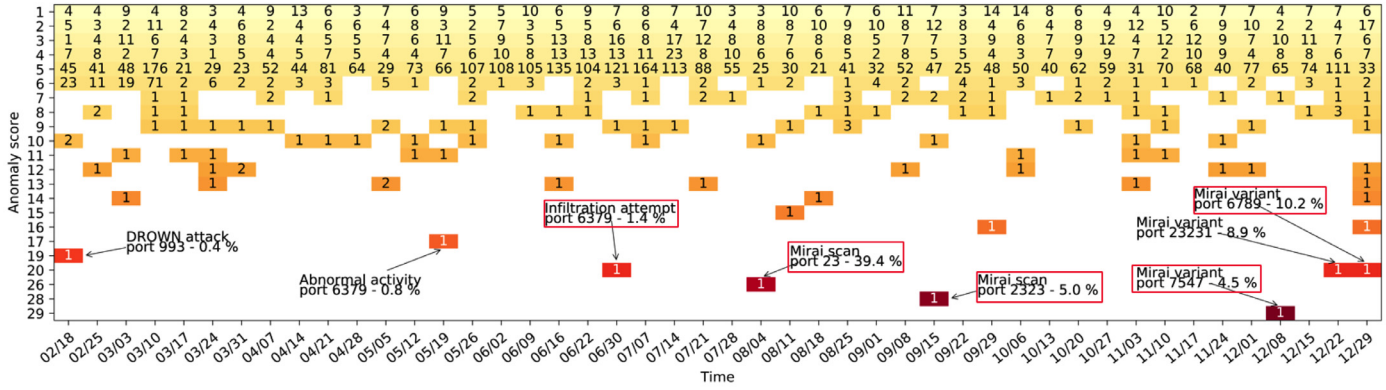
Feature	Aggregated view	Split view
srcDivIndex	11,376	101
destDivIndex	11,409	96
portDivIndex	11,375	102
meanSize	10,978	91
stdSize	10,549	67
perSYN	851	98

i) The 19-score on Feb. 19 is a scan prior to the DROWN attack [36], exploiting a vulnerability in Secure Sockets Layer version 2.0 (SSLv2) (CVE-2016-0800).

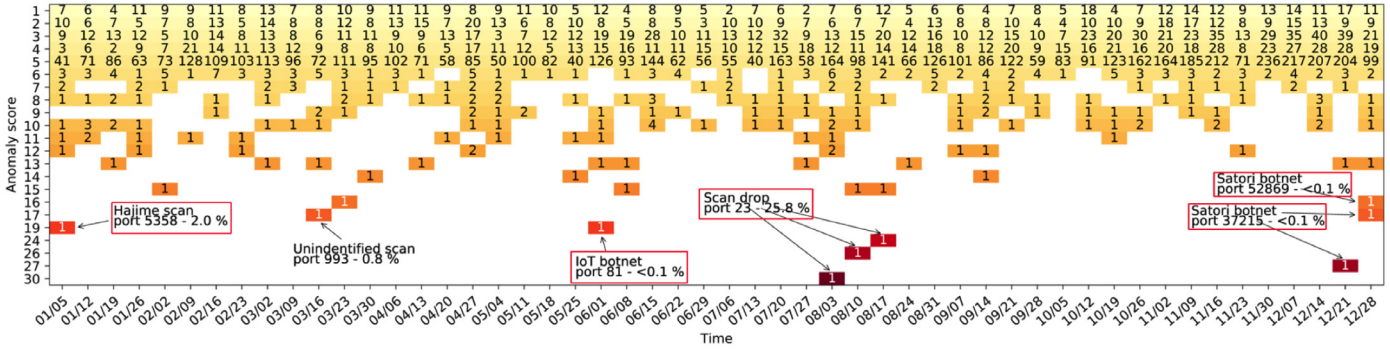
ii) The 17-score on May 19 corresponds to an exploit on port 6379 Redis, an in-memory key-value store used as a database or a cache. This day, numerous IP addresses with different source port numbers targeted this port. It could be a botnet or numerous different persons scanning for vulnerable devices. Indeed, Redis servers do not require authentication by default and therefore are easy victim of this type of scan. Also, this happens only a few days after buffer overflow vulnerabilities were discovered, leading to arbitrary code execution (CVE-2016-8339, CVE-2016-10517).

iii) Once again, the 20-score on June 30 corresponds to an exploit on port 6379. This day, a large SYN scan is observed coming from the same source IP address and targeting numerous hosts in several ASes of MAWI. This is either a large scan targeting the whole IPv4 space, through a tool like ZMap [55] that performs Internet-wide network scan in under 45 minutes, or someone trying to penetrate the MAWI network. This anomaly has been detected with a score of 4 in the UCSD dataset.

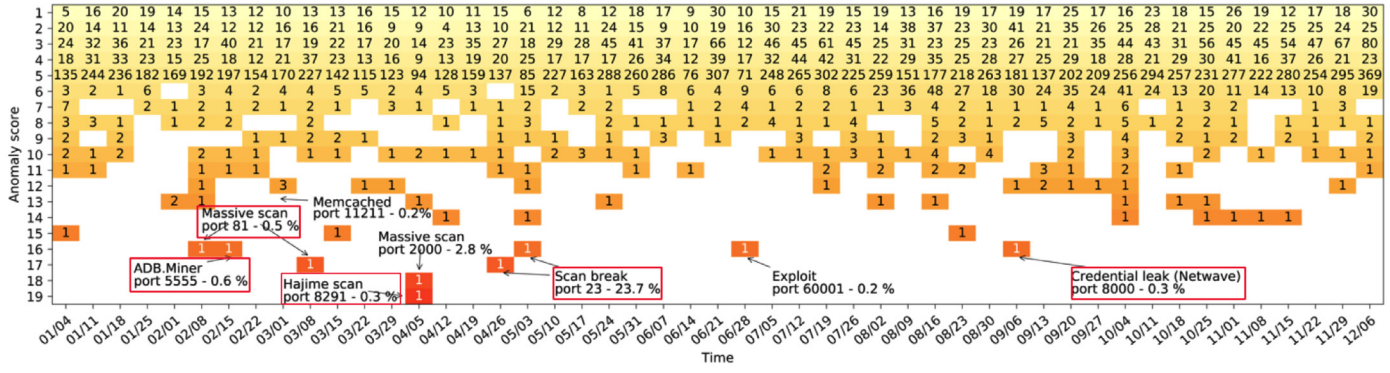
iv) The IoT Mirai botnet [2] is a major attack arisen in 2016. First, Mirai infected hosts send TCP SYN packets to random IP addresses on Telnet ports 23 and 2323, except those on a blacklist. Hosts whose Telnet port is open send back a SYN/ACK packet. Then, infected hosts try to establish a Telnet connection to them using a hard-coded list of credentials, and send the credentials to another server if it is successful. From there, a separate program executes architecture-specific malware. The victim is now infected and listens for attack commands from the C&C server, then starts scanning to infect other hosts. This is how Mirai spread into connected objects and form a worldwide army of bots. Indeed, the 26-score on Aug. 4 corresponds to the Mirai scan



(a) In 2016.



(b) In 2017.



(c) In 2018.

Fig. 6. Anomaly scores for the MAWI dataset (number of anomalies for one port, taking into account all features and all monitors). In the colored squares are given the numbers of ports with this anomaly score this day. Events detected in both datasets are tagged with a red frame.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

on port 23 and the 28-score on Sept. 15 relates to port 2323. Numerous Mirai variants exploit vulnerabilities on other ports later in 2016, as evidenced by the 29-score on Dec. 8 on port 7547, the 20-score on Dec. 22 on port 23,231 and the 20-score on Dec. 29 on port 6789. The attack has been detected in the UCSD dataset with a score of 4 for port 2222, of 5 for ports 23 and 7547, and of 6 for ports 2323 and 6789.

2017 period. Nine noticeable scores appear in Fig. 6b showing anomaly detection results in 2017.

i) A 19-score on Jan. 5 highlights a scan on port 8291, carried out by Hajime bots. Hajime [16] is an IoT worm revealed only a few days after the release of the source code for Mirai. The botnet is continuously evolving, taking advantage of newly released vulnerabilities. At the beginning of 2017, the efficient SYN scanner implementation scans for open ports 5358 (WSDAPI) [47]. The same way Mirai did on port 23, the extension module tries to exploit the victim using brute-force shell login. This anomaly has been detected with a score of 6 in the UCSD dataset.

ii) The 17-score on March 16 corresponds to a massive scan on port 993 which deals with secure IMAP (IMAPS). This day, a massive scan coming from the same IP address from port 993 as well has been observed. It may be a ZMap scan, or an attacker trying to infiltrate the MAWI network specifically. Indeed, IMAPS belongs to the list of ports scanned by default by Nmap [37] and is permanently stuck with the vulnerabilities in SSL 3.0.

iii) The June 1 a 19-score is observed, corresponding to a new IoT botnet spreading and exploiting vulnerability in security cameras [33], several days after the researcher Pierre Kim released a vulnerability analysis report on GoAhead and other OEM cameras. This anomaly has been detected with a score of 6 in the UCSD dataset.

iv) The 30-score, 26-score and 24-score, respectively on Aug. 3, Aug. 10 and Aug. 17 correspond to a sensible drop in the scan perpe-

trated by Mirai on port 23. This may be due to the Internet of Things (IoT) Cybersecurity Improvement Act of 2017 [52], adopted on Aug. 1, 2017. The latter seeks to improve the security of internet-connected devices, so that devices do not contain any known security vulnerabilities and are conceived using standard protocols. It also claims that the eventual patches should be applied even retroactively, which can explain the scan drops. These changes have also been noticed in the UCSD dataset, producing anomalies with scores of 4 and 5.

v) The 27-score on Dec. 21, and 15-score and 16-score on Dec. 28 involve ports 37,215 and 52869, that received numerous scans from the newest version of Satori (a Mirai variant) [34]. These anomalies have been detected in the UCSD dataset with scores of 4 and 5.

2018 period. Nine noticeable scores appear in Fig. 6a depicting the 2018 year. Compared to previous years, large scores are much rarer this time. Indeed, the maximum *AS* is up to 19, compared to 29 and 30 respectively in 2016 and 2017. We can still identify several main anomalies.

i) On Feb. 8 and March 8, exploits on port 81 are noticed. These days, almost the same IP address launched TCP SYN scans from the same source port number, targeting numerous MAWI subnetworks. This may be once again an Internet-wide network scan (e.g., by ZMap) or an attacker that targets the MAWI dataset specifically.

ii) The 16-score on Feb. 15 is actually a scan on port 5555. It comes from the ADB.Miner botnet, which identifies Android devices with Android Debug Bridge turned on, to control them and make them execute commands [35]. Hence, this day, numerous IP addresses sent SYN packets to various hosts in the MAWI network using different source port numbers, as seen for the Mirai botnet in 2016. This anomaly has been detected in the UCSD dataset with a score of 4.

iii) The 12-score on March 1 corresponds to the memcached attack on port 11211. Prior to the huge DDoS attack towards Github, large TCP SYN scans across the world targeted port 11,211 in order to identify memcached servers exposed without any authentication protection (CVE-2018-1000115). The anomaly observed this day is a mark of this large scan. These changes have also been noticed in the UCSD dataset, producing anomalies with scores of 4 and 5.

iv) The 18-score on Apr. 5 corresponds to a large scan on port 2000 coming from various source IP addresses with different source port numbers, and targeting many IP addresses from several ASes in the MAWI dataset. Cisco Skinny Call Control Protocol (SCCP) is often bound to this port, allowing terminal control for voice over IP. This scan is symptomatic of an IoT botnet, willing to exploit the few vulnerabilities disclosed last years for this protocol, and maybe IoTroop [11].

v) The 19-score on Apr. 5 highlights a scan on port 8291, carried out by Hajime bots. On May 2018, it exploits a vulnerability (CVE-2018-7445) published 13 days before. First, infected hosts scan random IP addresses on port 8291 to identify MikroTik devices. Once the bot has identified one device, it tries to infect it with a public exploit package sent via port 80 or an alternate port. If successful, the device infects new victims in turn under the same protocol. This day, as for the Mirai botnet, our program saw many IP addresses targeting the MAWI network on port 8291, using various source port numbers. This anomaly has been detected with a score of 4 in the UCSD dataset.

vi) On Apr. 26 and March 3, two anomalies on port 23 are detected. We observe that these days, *meanSize* considerably rises while *srcDivIndex* and *destDivIndex* fall. The number of packets is also lower than usual. Thus it looks like there are less malicious scans towards this port these days. Actually, botnets tend to use alternate ports because vulnerabilities are progressively patched and devices are armed against possible exploits on port 23. These changes have also been noticed in the UCSD dataset, producing anomalies with scores of 4 and 5.

vii) On June 28, a 16-score is stored for port 60001, probably corresponding to a Trojan named Trinity. It first connects to one of 11 IRC servers on UnderNet. The Trojan then joins a chat room and waits for commands to attack individual agents on the channel. The noticed anomaly probably results from that trojan, that we identified coming from two different IP addresses.

viii) On Sept. 6, a 16-score anomaly is observed on port 8000. Several days before, the possibility for an unauthenticated attacker to exfiltrate sensitive information about the network configuration (network SSID and password) has been made possible by an information disclosure in Netwave IP camera (CVE-2018-11653 and CVE-2018-11654). This anomaly has been detected in the UCSD dataset with a score of 4.

4.5.2. In the UCSD network telescope dataset

Figs. 7 a (2016), 7 b (2017) and 7 c (2018) show the number of ports with a given anomaly score each day, highlighting the main anomalies found these last years in the UCSD dataset. Once again, we frame common anomalies (found in both datasets) in red, and describe the anomalies detected uniquely in this one hereafter.

2016 period. Several noticeable scores appear in Fig. 7a showing anomaly detection results in 2016.

i) On Feb. 18, we observe many scans targeting destination ports from 36,242 to 36,560 (even numbers only), producing 6-score anomalies. The scans probably come from a botnet, because the source and destination IP addresses are very diverse. It is hard to determine the nature of these scans, because they target registered ports with no known vulnerabilities. It may be a stealthy scan technique to first determine if the host is up, to then focus on some ports showing vulnerabilities (as the Reaper botnet does, cf. [41]).

ii) On March. 17 and 24, a decrease in the scans targeting the range 36242–36560 is observed, producing 6-score anomalies.

iii) Once again, several scans targeting the range 19328–19454, producing 6-score anomalies have been noticed on Dec. 1.

iv) On Dec. 8, an anomaly is detected with a score of 5, highlighting a scan performed by Mirai on port TCP/2222, running Rockwell Automation ControlLogix whose several vulnerabilities are known [51]. The MAWI dataset contains traffic from a backbone link, and port 2222 may also be used as an alternative port for SSH, producing noise in data. This may explain why we did not detect the scan in the MAWI dataset

2017 period. Several noticeable scores appear in Fig. 7b showing anomaly detection results in 2017.

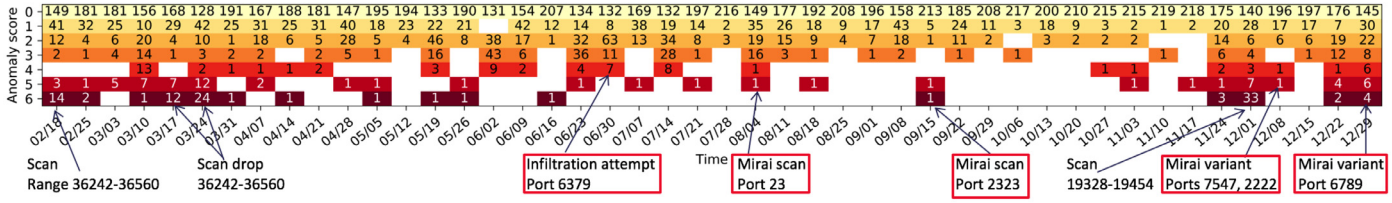
i) A 6-score anomaly on Jan. 5 highlights a scan on port TCP/27017 that runs MongoDB. A few days later, it has been reported and confirmed that many unsecured MongoDB databases have been scanned and vandalized around the world [49]. This attack affects only those databases which maintain default configurations, which leaves the database open to external connections via the Internet. We observe also a scan of the same nature on Feb. 16.

ii) We observe on March. 30 several scans targeting destination ports 88, 443, 3389, 6666 and 54313, coming from very distinct source IP addresses and with no flags (a network scanning technique known as Inverse TCP Flag Scanning). All of these ports present known vulnerabilities. A vulnerability on port 54,313 to exploit a Netis Router Backdoor has been detected and exploited by botnets back in Aug. 2016[43]. We did not find any information about a botnet targeting all these ports conjointly in the literature, thus we assume it is a small-scale attack.

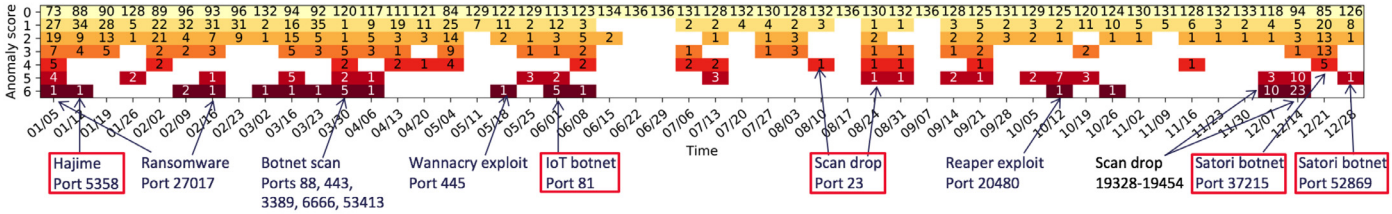
iii) A 6-score on May. 18 highlights a scan on port TCP/445 that runs Server Message Block (SMB) known for its ”EternalBlue” vulnerability. On May. 17, the recent WannaCry ransomware takes advantage of this vulnerability to compromise Windows machines [17].

iv) A 6-score on Oct. 12 highlights a scan on port 20480, probably from the Reaper botnet targeting a sequence of destination ports including 20480 [41]. However, instead of doing aggressive, asynchronous SYN scans for open Telnet ports, Reaper performs a more elaborate, conservative TCP SYN scan on a series of different ports, one IP at a time, targeting uncommon ports. Only after the first wave of scans on the victim, a second wave starts consisting of potential IoT web service ports: 80, 81, 82, 83, 84, 88, etc.

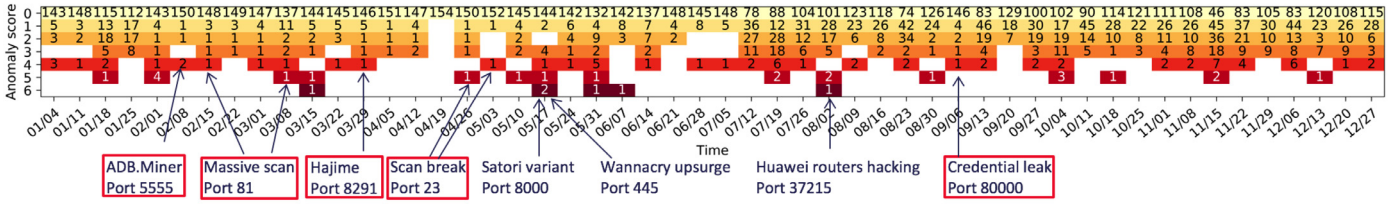
v) As in 2016, we observe a decrease in the scans targeting the range 19328–19454, producing 6-score anomalies on Dec. 7 and 14.



(a) In 2016.



(b) In 2017.



(c) In 2018.

Fig. 7. Anomaly scores for the UCSD Network Telescope dataset (number of anomalies for one port, taking into account all features and all monitors). In the colored squares are given the numbers of ports with this anomaly score this day. Events detected in both datasets are tagged with a red frame. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2018 period. Several noticeable scores appear in Fig. 7c showing anomaly detection results in 2018.

i) A 6-score on May. 17 highlights a scan on port TCP/8000, performed by the Satori botnet exploiting a buffer overflow vulnerability, tracked as CVE-2018-10088 [42]. The exploit could be used by remote attackers to execute arbitrary code by sending a malformed package via ports 80 or 8000. The upsurge of malicious scanning activity has been observed on June 15, but we could think that we detected a preliminary scan campaign from Satori developers. Moreover, this scan was probably not targeting the whole IPv4 range because we did not detect it in the MAWI dataset.

ii) A 6-score on May. 17 highlights a scan on port TCP/445, a port known for its "EternalBlue" vulnerability exploited by Wannacry. We noticed at this time an important increase in the botnet scan activities.

iii) A 6-score on Aug. 2 highlights a scan on port TCP/37215, a port used by Huawei HG532 routers. At the end of July, an IoT hacker identifying himself as "Anarchy" claimed to have hacked about 18000+ Huawei routers [4]. It works by exploiting an already known vulnerability which CVE is 2017–17215, used in Satori.

4.6. Anomaly score distribution

Last subsection shows that there are only a few emerging anomalies, and a lot more with small anomaly scores. We can so wonder which anomalies are greater enough to be analyzed. Fig. 8 shows the mean anomaly score occurrence per day in logarithmic scale for each year. For 6 features, we observe different levels of 6 items, e.g., *AS* from 1 to 6 are close to 10, then from 7 to 12 close to 0.3, then the number of anomalies progressively declines. Once again, the occurrence $AS = 5$ is high because a given variation on one port in one subnetwork generates one anomaly per feature, except for *perSYN*. For next evaluations, we set the threshold to define an anomaly either to 6 or 12, depending on if we want to detect a large variety of anomalies or only significant ones.

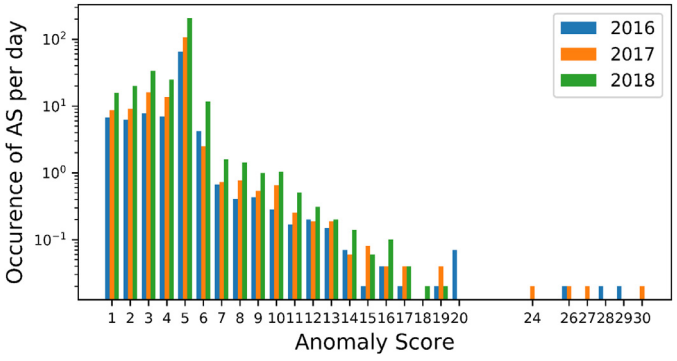


Fig. 8. Occurrences of each anomaly score during the last three years, showing a low number of significant anomalies, for the MAWI dataset.

In addition, Fig. 9 shows the distribution of anomaly scores resulted from the evaluation made on the UCSD dataset. We observe a very low number of significant anomalies, i.e., with a score at least equal to 4. For the MAWI dataset, we observed a decrease starting from $AS = 7$ (thus at least two impacted subnetworks) and not 4, probably because it contains more false positives due to traffic generated by internal hosts (while there is no such traffic in the UCSD dataset).

4.7. Features and parameters choice

In this subsection, we aim to refine the detection accuracy by analyzing how the features and the parameters impact the results.

4.7.1. Feature selection

We use two metrics to determine which features are the most useful in the anomaly detection process. The two following experiments are made on the whole traffic from 2016 to 2018. For both, a large number

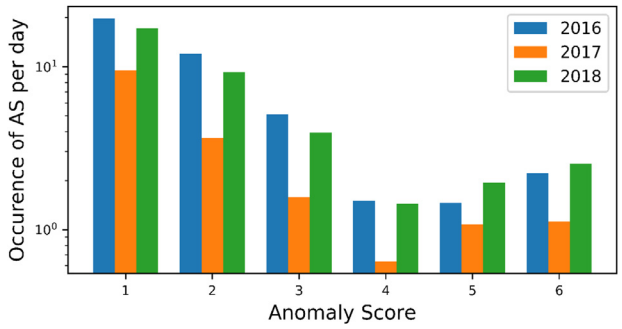


Fig. 9. Occurrences of each anomaly score during the last three years, showing a low number of significant anomalies, for the UCSD dataset.

of anomalies, i.e., with an $AS > 6$ is first considered, then only major ones, i.e., with an $AS > 12$.

F-Test based feature selection: F-Test is a statistical test used to compare between models, here between the input (features) and the output (anomaly detection results). It is useful in feature selection as we get to know the significance of each feature in improving the model. First, the anomalies are generated by launching our solution, and only the ones whose *AS* is higher than *T* are kept. Then, we observe the impact of each variation of feature on the anomalies found. Table 4 gives the results of F-Test applied on the produced anomalies, for each feature variation. We observe that the *perSYN* feature, that scores 0.114 / 0.009 and 0.152 / 0.039 respectively for an increase and a decrease, has the least impact on the results.

Number of absolute variations: we analyze for each anomaly whether the variations of features are distributed in several subnetworks, i.e., if the feature only rises or decreases in several subnetworks. This kind of variations is wanted as it means a distributed change, and not some random, differing variations. In Table 5 is given the number of absolute anomalies, i.e., where one given feature is only rising in several subnetworks, or decreasing in several subnetworks. We observe that the *perSYN* feature contains a majority of random variations, contrary to other features.

Table 4
F-Test scores per feature, among anomalies whose *AS* is higher than T. The higher the F-Test score of a feature is, the greater impact the feature has on final results.

Feature	F-Test score ($T = 6$)	F-Test score ($T = 12$)
+srcDivIndex	0.269	0.136
-srcDivIndex	0.689	1.000
+destDivIndex	0.599	0.378
-destDivIndex	1.000	0.610
+portDivIndex	0.633	0.188
-portDivIndex	0.648	0.899
+meanSize	0.321	0.158
-meanSize	0.932	0.463
+stdSize	0.692	0.302
-stdSize	0.478	0.825
+perSYN	0.114	0.009
-perSYN	0.152	0.039

Table 5
Number of absolute variations per feature (either rise in several subnetworks, or decrease in several subnetworks), among anomalies whose *AS* is higher than T.

Feature	# absolute variations ($T = 6$)	# absolute variations ($T = 12$)
srcDivIndex	318	12
destDivIndex	296	17
portDivIndex	334	16
meanSize	260	14
stdSize	211	7
perSYN	154	6

We can induce from these two experiments that the *perSYN* feature has the lowest impact on final results and does not often rise or decrease in the majority of subnetworks. This observation is interesting as this feature is widely used in TCP SYN scan detection algorithms.

4.7.2. Parameters tuning

Our algorithm has four parameters (see Table 1). In this subsection, we evaluate and discuss the impact of the key ones: the window size (N_{days}) and the minimum number of packets on one port ($N_{\min}$).

Number of days in the window (N_{days}): first, we wonder how the number of elements in the time-window impacts the validity of a normal distribution based on median. For each feature, we compute the median MSE (for all ports) between the true distribution and the normal distribution based on median, by making N_{days} vary. The result is shown in Fig. 10. We observe that the more days in the model, the more normally distributed the set of data, making the model with 20 days the best approximation. However, it may be possible to find a trade-off between a good approximation and a minimized time execution and memory allocation.

Looking at the anomaly detection results can also give evidence about the appropriate window size. The objective is to find an optimal window size that is not too large, but still gives a satisfactory detection accuracy. In Fig. 11a is shown the number of anomalies in 2016 depending on the threshold (minimum *AS* value) for diverse N_{days} values. We observe that 10 days (in green) enable to see all substantial anomalies, i.e., with an *AS* higher than 12. We keep $N_{days} = 10$ for the following simulations to minimize the window size.

Minimum number of packets on one port ($N_{\min}$): this time, we show the number of anomalies in 2016 depending on the threshold (minimum *AS* value) for diverse $N_{\min}$ values in Fig. 11b. We observe that for a threshold higher than 12, the number of anomalies is the same no matter $N_{\min}$. It means that anomalies happen most of the time on ports with a sufficient amount of traffic (more than 100 packets). Therefore we choose $N_{\min} = 100$ to limit the number of ports to deal with, hence reducing the time execution and memory allocation.

To complete the study, Fig. 12 shows the number of ports to deal with in each subnetwork in 2016, by making $N_{\min}$ vary. With $N_{\min} = 100$, the maximum amount of ports to analyze is around 6000 for subnetwork E, which is quite reasonable.

4.8. Anomalies classification

We define a set of commonly-called " expert rules" with some conditions about the features evolution. For each characteristic, the variation has to be noticed in at least two subnetworks, while the opposite varia-

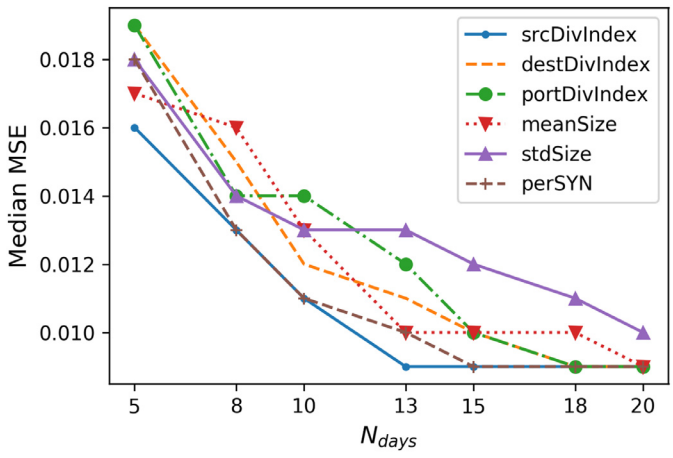
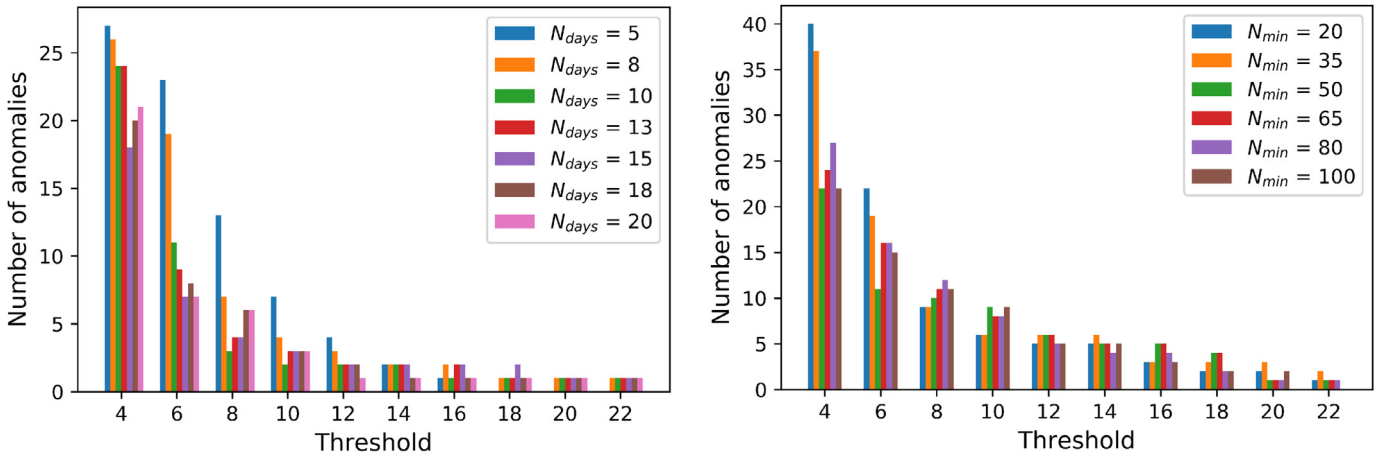


Fig. 10. Median MSE (between the true and the normal distribution) per feature, with the window size (N_{days}) varying.



(a) Varying the number of days in the model (N_{days}). (b) Varying the minimum number of packets on one port (N_{min}).

Fig. 11. Number of anomalies depending on a parameter for diverse thresholds (2016).

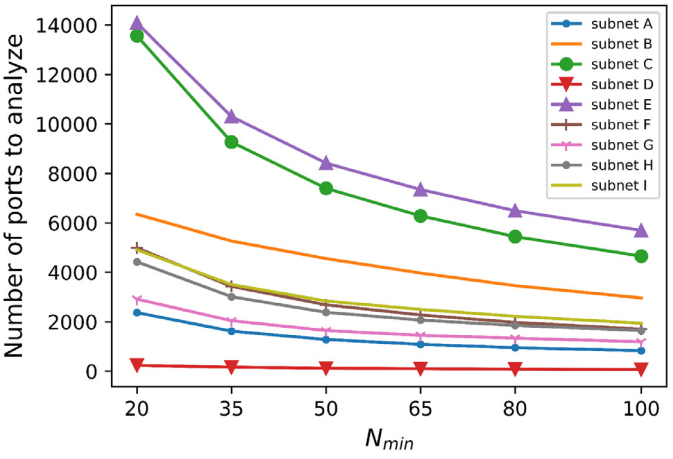


Fig. 12. Number of ports to analyze in every subnetwork, with the minimum number of packets per port (N_{min}) varying.

tion must not be noticed in any subnetwork. For example, if the characteristic is *-srcDivIndex*, an anomaly is verified if *srcDivIndex* decreases in at least two subnetworks (*-srcDivIndex* ≥ 2) and if it does not rise in any subnetwork ($+srcDivIndex = 0$). Furthermore, the feature *portDivIndex* is not self-speaking and thus not used in the classification. However it gives additional information about the likelihood for the source port to be spoofed or not. Therefore, if *portDivIndex* rises in at least two subnetworks while it does not decrease in any subnetwork, then the source port may be randomly chosen. If the exact opposite is observed, the port may be spoofed. In addition, the *perSYN* feature is not used for the classification either, because of its small impact on the results (see Section 4.7.1).

In Table 6 are given several classes of anomalies observed the last few years, followed by their characteristics. They are ordered so that the most specific rules override the lower ones. Some anomalies referring to attacks are identified, such as "forged packets", "large scan" probably launched with ZMap, "DDoS attack", "targeted scan", "botnet scan" and "botnet expansion" that kills other processes bound to a port. There are also anomalies caused by a drop in malicious activities, such as "normal packets" and "less botnet scan".

Only significant anomalies (whose $AS > 12$) are kept to be classified. The fact that some anomalies do not belong to any class is not worrying, as it means a port whose nature evolves rapidly, rather than a port targeted by one clearly identified attack. We observe a nature of traffic seemingly different between 2016 and 2018, with a majority of scans and DDoS attacks in 2016 and more normal activity in 2018.

4.9. Ground-truth

Since we are detecting anomalies on a long-term scale and based on destination port numbers, we are not able to compare our work to other intrusion detection systems on a fair basis. We are only comparing our results to the MAWILab database, showing that it did not detect the main anomalies that we identified, either in the MAWI dataset or the UCSD dataset. Moreover, as a matter of fact we focus on detecting major botnets and attacks arisen these last years and we do not benefit from a labelled dataset of a several-years period.

Nevertheless, we attempted to provide a retrospective analysis of the major botnets and attacks arisen between 2016 and 2018. The literature in this field is not prolific so we combined different attack data sources [40,56] to create the list; we then evaluated whether we detected them using different datasets. Table 7 hereafter shows the IoT and Malware Botnets reported between 2016 and 2018, and the anomaly score of the event in case we detected it.

The false detection rate is computed as the number of benign events classified as malicious over the total number of benign events. However, we do not have labelled data in the available datasets and thus it is not feasible to compute the false positive rate. For instance, we cannot distinguish between a false positive and a small attack which targets only this network and which is not referenced as a botnet. However, we estimate hereafter the number of **unknown anomalies**, defined as major detected anomalies which are not part of a known botnet. For the MAWI dataset, we consider as major anomalies those starting from an anomaly score (AS) of 12 (therefore the equivalent of 2 subnetworks impacted with 6 anomalous features); for the UCSD dataset, which only contains one vantage point, we consider as major anomalies those with an AS of 6 (i.e., with all features anomalous). We count 71 (respectively 22, 25 and 24 in 2016, 2017 and 2018) unknown anomalies for the MAWI dataset and 26 (respectively 14, 9 and 3 in 2016, 2017 and 2018) for the UCSD dataset.

As a result, we detected a slightly higher number of anomalies in the UCSD dataset: this happens likely because it is less noisy and only contains unsolicited traffic. On the contrary, the MAWI dataset shows the advantage of containing proportionally less unknown anomalies. This is likely due to the several subnetworks in the MAWI dataset; this enables to discard the local anomalies (i.e., seen in only one subnetwork), which are caused by occasional traffic peaks and may be considered as false alerts, and to keep only distributed anomalies.

Table 6
Definition of classes and their characteristics, with their occurrences each year.

Classes and characteristics	2016	2017	2018
More normal packets *+meanSize,+stdSize*	1	5	12
More forged packets *-meanSize,-stdSize*	0	1	1
Large scan *-srcDivIndex,+destDivIndex, -meanSize*	3	5	0
DDoS *+srcDivIndex,-destDivIndex*	6	1	3
Botnet scan *+srcDivIndex,+destDivIndex,-meanSize*	5	2	6
Botnet expansion *+srcDivIndex,+destDivIndex,-stdSize*	2	2	2
Targeted scan *-srcDivIndex,-destDivIndex*	1	2	4
Less botnet scan *-srcDivIndex,-dstDivIndex, +meanSize,+stdSize*	0	5	3
Total	18/21	23/32	31/40

Table 7
List of most impactful botnets reported these last three years.

Botnet	Port	Year	MAWI dataset	UCSD dataset
Mirai (IoT botnet)	23	2016	✓(26/54)	✓(5/6)
Mirai (IoT botnet)	2323	2016	✓(28/54)	✓(6/6)
Mirai (IoT botnet)	7547	2016	✓(29/54)	✓(5/6)
Mirai (IoT botnet)	6789	2016	✓(20/54)	✓(6/6)
Mirai (IoT botnet)	2222	2016	X	✓(5/6)
Mirai (IoT botnet)	23231	2016	✓(20/54)	X
Hajime (Malware botnet)	5358	2017	✓(19/54)	✓(6/6)
Reaper (IoT botnet)	20480	2017	X	✓(6/6)
Satori (IoT botnet)	37215	2017	✓(27/54)	✓(4/6)
Satori (IoT botnet)	52869	2017	✓(17/54)	✓(5/6)
ADB.Miner (IoT botnet)	5555	2018	✓(19/54)	✓(4/6)
Memcached (Malware botnet)	11211	2018	✓(12/54)	X
Hajime (Malware botnet)	8291	2018	✓(19/54)	✓(4/6)
Satori (IoT botnet)	8000	2018	X	✓(6/6)
Total			11/14 detected ($TPR = 78.6\%$)	12/14 detected ($TPR = 85.7\%$)

Table 8
Complexity notations.

Notation	Definition
p	Number of packets collected in the subnet
a	Number of attributes per collected packet
p'	Number of ports after applying $N_{\min}$ filter
f	Number of features
l	Number of anomalies during one time slot
n_i	Number of entries in CTi (for feature i)

5. Complexity and performances analysis

5.1. Complexity analysis

To evaluate the scalability of our algorithm, we provide the space and time complexity analysis of each step. Table 8 hereafter aims at simplifying the notations to express the complexity.

5.1.1. Data collection

Space complexity: the program stores a attributes per packet, hence the complexity is $O(a \cdot p)$.

Time complexity: packets are collected on the fly, and packets attributes are instantly stored. Therefore the execution of the algorithm is near real-time.

5.1.2. Features computation (see algorithm 1)

Space complexity: the memory space needed to compute features from attributes is O(1) for all features, i.e., the three diversity indices, the mean, the standard deviation, the percentage and the number of packets. Therefore, the total space complexity is $O(f)$.

Time complexity: first, we need to parse ports and compute a set of features for each one. The feature computations, i.e., three diversity indices, a mean, a standard deviation, a percentage and a counter, multiplied by the number of ports, generates a total complexity of $O(p' \cdot \sum_{i=1}^{f} n_i)$.

5.1.3. Anomaly detection (see algorithm 2)

Note that steps 2 and 3 can be merged to avoid parsing twice each tuple of ports and features.

Space complexity: the median computation requires to order values using quick sort, whose memory space allocated is logarithmic: $O(log(N_{days}))$. Then, one need to store values for the median, the median absolute deviation, the modified Z-score and the anomalies. These values are updated for each feature and port, thus the space complexity is $O(3 + l)$.

Time complexity: The time complexity for the quick sort needed to compute the median is $O(N_{days} \cdot \log(N_{days}))$.

Finally, as $a < < p$, the total space complexity for all steps is equal to $O(p + \log(N_{days}))$. The total time complexity is $O(p' \cdot \sum_{i=1}^{f} n_i + N_{days} \cdot \log(N_{days}))$.

Both space and time complexity directly depend on $N_{\min}$ (which determines the number of ports p' to deal with) and on N_{days}. That is why, in the following evaluations, we better minimize these values, while keeping a good detection rate.

5.2. Execution performance

We performed our experiments on a 2017 MacBook Pro-with 2.3 GHz Intel Core i5 Processor and 16GB RAM. The time spent to detect anomalies depends on the number of analyzed ports, thus on $N_{\min}$, the minimum number packets on a port to consider it. We consider here the UCSD dataset, composed of 152 days picked from 2016 to 2018, each one containing one million of packets. The learning phase (building per-port profiles) and the detection phase (computing the Z-score) are run simultaneously. In fact, we dispose of a sliding window, thus the model is updated each time and the detection is made on the updated model on the fly. In total, it took 6390 seconds to run the full algorithm (i.e., the learning and detection phases) on 152 days, covering 17,152 ports and one subnetwork.

6. Conclusion and perspectives

The port-based anomaly detection algorithm we proposed is able to detect known and unknown attacks targeting connected objects and servers around the world. Early stages of the attacks, namely the exploited vulnerabilities, can be detected beforehand.

Our method, called Split-and-Merge detection, leverage on statistics to detect main changes in the usage of application ports. The empirical results obtained applying our method to real traces are very promising, since our algorithm detected a number of world-wide attacks from 2016 to 2018. In contrast, current IDSs, among which the notorious MAWILab, have not detected them. We evaluated the scalability of our algorithm by computing its complexity in terms of space and time. Moreover, we showed that our algorithm produces a very low number of false positives. We demonstrated the validity of our statistical model assumptions, showing how leveraging on distribution analysis of features we can refine the detection accuracy by analyzing how the parameters and features impact the results. We provide a classification of detected attacks given expert rules by analyzing jointly the features evolution.

The objective of this work is to detect the early apparition of botnets, newly exploited vulnerabilities and other diverse attacks within a given network. However, we noticed that we could obtain a comprehensive list of major botnets that emerged last years by coupling different data sources. For example, some preliminary scans target a limited IPv4 range only and thus are visible only on some datasets. As a result, we found that some attacks detected in one real traffic source (the MAWI dataset), containing both attack traffic and background traffic, were not found in another source containing only attack or unsolicited traffic (the UCSD dataset). We thus believe it would be interesting in future work to leverage these different data sources in order to describe the landscape of major attacks and botnets in recent years.

As a further work, we also plan to implement our proposal in an SDN environment, using a controller and several switches running the algorithm. This way the identified attacks could be mitigated, for example by patching the appropriate services or network programming.

Declaration of Competing Interest

1. Agathe Blaise
   - Thales SIX GTS France (all)
   - Sorbonne Université, LIP6, PHARE team (all)
2. Mathieu Bouet
   - Thales SIX GTS France (all)
3. Vania Conan
   - Thales SIX GTS France (all)
4. Stefano Secci
   - Cnam, CEDRIC team (all)
   - Sorbonne Université, LIP6, PHARE team (all)

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comnet.2020.107391

CRediT authorship contribution statement

Agathe Blaise: Conceptualization, Methodology, Software, Validation, Writing - original draft. **Mathieu Bouet:** Conceptualization, Writing - review & editing, Supervision. **Vania Conan:** Conceptualization, Writing - review & editing, Supervision. **Stefano Secci:** Conceptualization, Writing - review & editing, Supervision.

References

[1] Akamai, Memcached UDP reflection attacks, 2018. Online: https://blogs.akamai.com/2018/02/memcached-udp-reflection-attacks.html.

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, Understanding the Mirai botnet, in: Proceedings of the USENIX Security Symposium (USENIX Security), 2017, pp. 1093–1110.

[3] A. Aqil, K. Khalil, A.O. Atya, E.E. Papalexakis, S.V. Krishnamurthy, T. Jaeger, K.K. Ramakrishnan, P. Yu, A. Swami, Jaal: Towards network intrusion detection at isp scale, in: Proceedings of the International COnference on emerging Networking EXperiments and Technologies - (CoNEXT), 2017, doi:10.1145/3143361.3143399.

[4] Avira, 18000 routers taken hostage in less than a day, 2018. Online: https://blog.avira.com/18000-routers-taken-hostage-in-less-than-a-day/.

[5] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools, IEEE Commun. Surv. Tutor. 16 (1) (2014) 303–336, doi:10.1109/surv.2013.052213.00046.

[6] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Surveying port scans and their detection methodologies, Comput. J. 54 (10) (April 2011) 1565–1581, doi:10.1093/comjnl/bxr035.

[7] A. Blaise, M. Bouet, S. Secci, V. Conan, Split-and-Merge: detecting unknown botnets, in: Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), 2019.

[8] CAIDA, UCSD Network Telescope Aggregated Flow Dataset, 2020. Online: https://www.caida.org/data/passive/telescope-flowtuple.xml.

[9] C. Callegari, S. Giordano, M. Pagano, Entropy-based network anomaly detection, in: Proceedings of International Conference on Computing, Networking and Communications (ICNC), 2017, doi:10.1109/iccnc.2017.7876150.

[10] M. Celenk, T. Conley, J. Willis, J. Graham, Predictive network anomaly detection and visualization, IEEE Trans. Inf. Forensics Secur. 5 (2) (2010) 288–299, doi:10.1109/tifs.2010.2041808.

[11] CheckPoint, IoTroop botnet: The full investigation, 2018. Online: https://research.checkpoint.com/iotroop-botnet-full-investigation/.

[12] J.Á. Cid-Fuentes, C. Szabo, K. Falkner, An adaptive framework for the detection of novel botnets, Comput. Secur. 79 (2018) 148–161, doi:10.1016/j.cose.2018.07.019.

[13] Cisco, Snort - network intrusion detection & prevention system, 2018. Online: https://www.snort.org/.

[14] A. Dainotti, K. Benson, A. King, kc claffy, M. Kallitsis, E. Glatz, X. Dimitropoulos, Estimating internet address space usage through passive measurements, ACM SIGCOMM Comput. Commun. Rev. 44 (1) (2013) 42–49, doi:10.1145/2567561.2567568.

[15] J. Dromard, G. Roudiere, P. Owezarski, Online and scalable unsupervised network anomaly detection method, IEEE Trans. Netw. Serv. Manage. 14 (1) (March 2017) 34–47, doi:10.1109/tnsm.2016.2627340.

[16] S. Edwards, I. Profetis, Hajime: Analysis of a decentralized internet worm for IoT devices, 2016. Online: https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf.

[17] FireEye, Smb exploited: Wannacry use of "eternalblue", 2017. Online: https://www.fireeye.com/blog/threat-research/2017/05/smb-exploited-wannacry-use-of-eternalblue.html.

[18] FukudaLab, MAWILab database, 2019. Online: http://www.fukuda-lab.org/mawilab.

[19] J. Francois, I. Aib, R. Boutaba, Firecol: a collaborative protection network for the detection of flooding DDos attacks, IEEE/ACM Trans. Networking 20 (6) (Dec 2012) 1828–1841, doi:10.1109/tnet.2012.2194508.

[20] FukudaLab, MAWILab database, 2019. Online: http://www.fukuda-lab.org/mawilab.

[21] Github, Source code for Split-and-Merge detection algorithm, 2019. Online: https://github.com/a-blaise/split-and-merge.

[22] G. Gu, J. Zhang, W. Lee, BotSniffer: Detecting botnet command and control channels in network traffic, in: Proceedings of the Network and Distributed System Security Symposium (NDSS), 2008.

[23] A. Guillot, R. Fontugne, P. Winter, P. Mrindol, A. Dainotti, C. Pelsser, Chocolatine: Outage detection for internet background radiation, in: Network Traffic Measurement and Analysis Conference (TMA), 2019.

[24] F. Haddadi, D.L. Cong, L. Porter, A.N. Zincir-Heywood, On the Effectiveness of Different Botnet Detection Approaches, in: Information Security Practice and Experience, Springer International Publishing, 2015, pp. 121–135, doi:10.1007/978-3-319-17533-1_9.

[25] B. Iglewicz, D. Hoaglin, How to Detect and Handle Outliers, in: P. Edward F. Mykytka (Ed.), The ASQC Basic References in Quality Control: Statistical Techniques, 16, 1993.

[26] C.-N. Kao, Y.-C. Chang, N.-F. Huang, I.S. S, I.-J. Liao, R.-T. Liu, H.-W. Hung, A predictive zero-day network defense using long-term port-scan recording, in: 2015 IEEE Conference on Communications and Network Security (CNS), 2015, doi:10.1109/cns.2015.7346890.

[27] A. Lakhina, M. Crovella, C. Diot, Mining anomalies using traffic feature distributions, in: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM, ACM Press, 2005, doi:10.1145/1080091.1080118.

[28] A.G.P. Lobato, M.A. Lopez, I.J. Sanz, A.A. Cardenas, O.C.M.B. Duarte, G. Pujolle, An adaptive real-time architecture for zero-day threat detection, in: IEEE International Conference on Communications (ICC), 2018, doi:10.1109/icc.2018.8422622.

[29] W. Lu, H. Tong, Detecting Network Anomalies Using CUSUM and EM Clustering, in: Advances in Computation and Intelligence, 2009, pp. 297–308, doi:10.1007/978-3-642-04843-2_32.

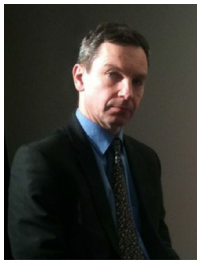
[30] A. Mahanti, N. Carlsson, A. Mahanti, M. Arlitt, C. Williamson, A tale of the tails: power-laws in internet measurements, IEEE Netw 27 (1) (2013) 59–64, doi:10.1109/mnet.2013.6423193.

[31] MAWI, MAWI working group traffic archive, 2019. Online: http://mawi.wide.ad.jp/mawi/.

[32] S.A. Mirheidari, S. Arshad, R. Jalili, Alert Correlation Algorithms: A Survey and Taxonomy, in: Cyberspace Safety and Security, 2013, pp. 183–197, doi:10.1007/978-3-319-03584-0_14.
[33] Netlab360, New threat report: A new IoT botnet is spreading over http 81 on a large scale, 2017.
[34] Netlab360, Warning: Satori, a Mirai branch is spreading in worm style on port 37215 and 52869, 2017. Online: http://blog.netlab.360.com/warning-satori-a-new-mirai-variant-is-spreading-in-worm-style-on-port-37215-and-52869-en/.
[35] Netlab360, ADB.Miner: More information, 2018. Online: http://blog.netlab.360.com/adb-miner-more-information-en/.
[36] Nimrod Aviram, DROWN Attack, 2016. Online: https://drownattack.com/.
[37] Nmap, Nmap: the network Mapper, 2018. Online: https://nmap.org/.
[38] S. Panjwani, S. Tan, K. Jarrin, M. Cukier, An experimental evaluation to determine if port scans are precursors to an attack, in: 2005 International Conference on Dependable Systems and Networks (DSN), 2005, doi:10.1109/dsn.2005.18.
[39] V. Paxson, Bro: a system for detecting network intruders in real-time, Comput. Netw. 31 (23–24) (Dec 1999) 2435–2463, doi:10.1016/s1389-1286(99)00112-7.
[40] PentaSecurity, Top 5 botnets of 2017, 2017. Online: https://www.pentasecurity.com/blog/top-5-botnets-2017/.
[41] Radware, Why the world is under the spell of IoTReaper, 2017. Online: https://blog.radware.com/security/2017/10/iot_reaper-botnet/.
[42] Radware, Satori iot botnet variant, 2018. Online: https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/satori-iot-botnet/.
[43] SANS ISC InfoSec Forums, Surge in exploit attempts for netis router backdoor (udp/53413), 2017. Online: https://isc.sans.edu/forums/diary/Surge+in+Exploit+Attempts+for+Netis+Router+Backdoor+UDP53413/21337/.
[44] P.K. Shanmugam, N.D. Subramanyam, J. Breen, C. Roach, J.V. der Merwe, DEIDtect: towards distributed elastic intrusion detection, in: Proceedings of the ACM SIGCOMM workshop on Distributed cloud computing (DCC), 2014, doi:10.1145/2627566.2627579.
[45] D. Singh, D. Patel, B. Borisaniya, C. Modi, Collaborative IDS framework for cloud, Int. J. Netw. Secur. 18 (2015) 699–709.
[46] M.-Y. Su, G.-J. Yu, C.-Y. Lin, A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach, Comput. Secur. 28 (5) (2009) 301–309, doi:10.1016/j.cose.2008.12.001.
[47] Symantec, Hajime worm battles mirai for control of the internet of things, 2017. Online: https://www.symantec.com/connect/blogs/hajime-worm-battles-mirai-control-internet-things.
[48] A.G. Tartakovsky, A.S. Polunchenko, G. Sokolov, Efficient computer network anomaly detection by changepoint detection methods, IEEE J. Sel. Top. Signal Process. 7 (1) (2013) 4–11, doi:10.1109/jstsp.2012.2233713.
[49] TechRepublic, Massive ransomware attack takes out 27,000 mongodb servers, 2017. Online: https://www.techrepublic.com/article/massive-ransomware-attack-takes-out-27000-mongodb-servers/.
[50] TechRepublic, How to stop Memcached DDoS attacks with a simple command, 2018. Online: https://www.techrepublic.com/article/how-to-stop-memcached-ddos-attacks-with-a-simple-command/.
[51] US CERT, Ics advisory (icsa-13-011-03), rockwell automation controllogix plc vulnerabilities, 2019. Online: https://www.us-cert.gov/ics/advisories/ICSA-13-011-03.
[52] US Congress, Internet of Things (IoT) Cybersecurity Improvement Act of 2017, 2017. Online: https://www.congress.gov/115/bills/s1691/BILLS-115s1691is.pdf.
[53] A. Wang, W. Chang, S. Chen, A. Mohaisen, Delving into internet DDos attacks by botnets: characterization and analysis, IEEE/ACM Trans. Netw. 26 (6) (2018) 2843–2855, doi:10.1109/tnet.2018.2874896.
[54] W. Wang, Y. Shang, Y. He, Y. Li, J. Liu, Botmark: automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors, Inf. Sci. 511 (2020) 284–296, doi:10.1016/j.ins.2019.09.024.
[55] E.W. Zakir Durumeric, J.A. Halderman, ZMap: Fast internet-wide scanning and its security applications, in: Proceedings of the USENIX Security Symposium (USENIX Security), 2013.
[56] ZDNet, A decade of malware: Top botnets of the 2010s, 2019. Online: https://www.zdnet.com/article/a-decade-of-malware-top-botnets-of-the-2010s/.
[57] C.V. Zhou, C. Leckie, S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection, Comput. Secur. 29 (1) (2010) 124–140, doi:10.1016/j.cose.2009.06.008.

Agathe Blaise is currently a Ph.D. student at Thales Communications & Security (Gennevilliers, France) and LIP6, Sorbonne University (Paris, France). She received her Engineering degree in Computer Science from ISEN (Lille, France) in 2017. Her research interests are in the field of data analysis applied to network security.

Mathieu Bouet received the Ph.D. degree in Computer Science and the Habilitation degree from Sorbonne University (formerly UPMC – Paris VI) in 2009 and 2017, respectively. He is a Research Expert in networking and communications with Thales, France, where he currently manages research activities on network softwarization with the Networking Laboratory, Advanced Studies Department. His research interests are mainly focused on network virtualization and network optimization.



Vania Conan received the Engineering and Ph.D. degrees in Computer Science from Mines ParisTech in 1990 and 1996, respectively, and the Habilitation degree from Sorbonne University, Paris in 2012. He is a Senior Research Expert in networking and communications with Thales, France. He is currently the Head of the Networking Laboratory, Advanced Studies Department in Thales. He has been conducting research in the fields of software-defined communications and wireless networking. He has published over 100 international conference and journal papers and holds 10 patents in networking technologies. His current research topics include mobile network protocols and virtualized network design.

Stefano Secci is professor of networking at Cnam (Conservatoire national des arts et metiers), Paris, France. He received the M.Sc. Degree in telecommunications engineering from Politecnico di Milano, Italy, in 2005, and a dual Ph.D. Degree in computer science and networks from Politecnico di Milano and Telecom ParisTech, France, in 2009. He was associate professor at LIP6, UPMC from 2010 to 2018. His current interests cover novel routing and switching architectures and network virtualization. Webpage: http://cedric.cnam.fr/~seccis/.