Split-and-Merge: Detecting Unknown Botnets

Agathe Blaise*†, Mathieu Bouet†, Stefano Secci‡ and Vania Conan†
*Sorbonne Université, CNRS LIP6, Paris, France
†Thales, Gennevilliers, France. Email: {name.surname}@thalesgroup.com
‡Cnam, Paris, France. Email: stefano.secci@cnam.fr

***Abstract*—On September 20th, 2016, Mirai struck the Internet in a massive surprise attack. This is just but one example of unknown botnets, not detected nor mitigated at the time they happened. Current anomaly detection techniques can only detect them after they have spread. However such attacks are generally preceded by several stages, including infection of hosts or fingerprinting of devices. Being able to capture this activity would allow their early detection. In this paper, we propose a strategy aimed at the early detection of unknown botnets, which acts by (i) splitting and merging distributed monitoring data and related metrics, (ii) monitoring at the port-level, with a simple yet efficient change-detection algorithm based on a modified Z-score measure. The analysis of port usage is first split over different parts of the network, and then merged to retain only similar anomalies. This ensures detection of large-scale attacks and drastically reduces false positives. We validate the approach on the MAWI data set, which provides daily traces of a transpacific backbone link. We demonstrate that the solution generates a very low number of false positives. We show how it detects some main attacks (including Mirai) arisen the last three years that traditional anomaly detectors have not seen. Details about noticed anomalies are provided to help the administrator qualifying them through specific features.**

I. Introduction

In 2016, the Mirai botnet [1] launched a massive attack towards DNS servers of major Internet providers, cutting access to high-profile websites during several hours. Beforehand, it reunited nearby 50,000 devices in its bot army, but has not been detected until too late. Mirai acted like a revolutionary IoT-based malware since the release of its source code [2] that led to huge increase in other botnets development. Actually, malware that targets Internet-of-Things (IoT) devices is responsible for many Distributed Denial-of-Service (DDoS) attacks. It exploits the lack of security of connected objects to create botnets, spreading extremely fast. We expect to see an increase in such IoT attacks, along with the explosion of IoT devices that could grow up to 125 billion by 2030 [3]. Recently, DDoS attacks significantly increased in terms of number and duration; indeed, the first half of 2018 saw seven times more large attacks (higher than 300 Gbps) compared to the same period in 2017, as noted in a Kaspersky report [4]. Furthermore, these botnets slightly propagate and affect whole networks without even being noticed, until they reach their real target. There is thus an urgent need to detect this kind of threats as soon as possible.

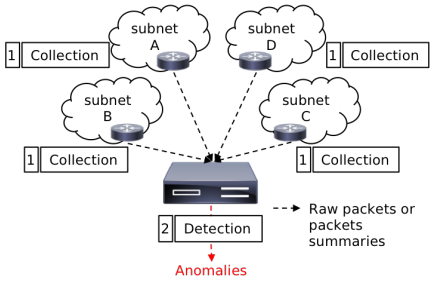
Designed to ensure cyber-security in networks, Intrusion Detection Systems (IDSs) aim to identify malicious activities and related threats. However, as a matter of fact most botnets go under the radars. Indeed, current IDSs work at different traffic granularities, e.g., flow, host or packet, to detect anomalies. However, they miss global changes on application ports that are involved during the infection and 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 [5], and therefore new attempts on one port can be observed simultaneously in the whole Internet. Moreover, current IDSs work on small variations of traffic, generally using time-sliding windows that last a few seconds. Therefore, they cannot build long-term profiles for port and detect major changes in their usage. Finally, current 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 watches network trends and spots main changes in the usage of a single port. This aims at detecting botnets that use protocols in a particular way during their lifecycle. A split-and-merge technique ensures that changes in one port are distributed and not due to random traffic variations (that can be caused by ephemeral ports). Local detections 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) would do (Figure 1b). The number of false positives can hence be significantly lower as only anomalies detected in several places are confirmed.

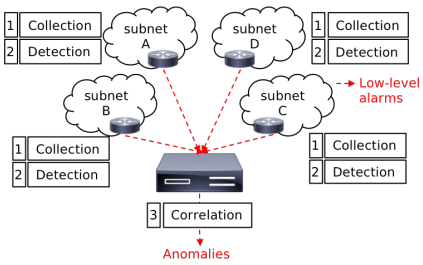
Intuitively, the most obvious way to identify an attack is to observe a sudden rise in traffic towards a port. However, this may be inefficient as it can be a well-known vulnerable port, already massively scanned. Actually, some stealthy changes in its behavior can be identified as anomalies. In our algorithm, several features representing port usages are computed. Large packets batches picked at a frequency of several days enable to profile the evolution of features over time, then simple statistical measures spot anomalies in the features time-series. In addition, our solution does not require parsing packets payload, contrary to most botnet detection tools. Therefore it can be used in a network with end-to-end encryption.

For our evaluation, we use the MAWI dataset [6], containing daily traffic traces of a transpacific backbone link. The dataset is restricted to a single ISP, hence corresponds to what could be used at the ISP-level. We do not use it as background traffic but aim at detecting real attacks from it, providing a better knowledge of the dataset at the same time. Our source code is publicly available at [7]. We present the intrusion detection

978-3-903176-15-7 © 2019 IFIP



(a) Local collection and central detection.



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

Fig. 1: Two possible approaches for large-scale NIDS.

results against known attacks arisen the last three years, not detected by the MAWILab detection algorithm [8], and also what appear as unknown events. Subsequently, we demonstrate that our algorithm greatly reduces the number of false positives compared to a single IDS running on the whole dataset.

This paper is structured as follows. Section II addresses related work. Section III presents our methodology to detect distributed changes in port usages. In Section IV, we present the evaluation, highlighting the benefits of our method in terms of false detection rate and detection accuracy, and proposing a classification of the detected anomalies. Finally, Section V concludes the paper.

II. Related works

In this section, we propose a classification of intrusion detection techniques and review related works.

A. Intrusion detection methodologies

Many algorithms are proposed in the literature for network intrusion detection [9]. We can classify them in two main families: knowledge-based and anomaly-based techniques.

Knowledge-based (or signature-based) solutions such as Snort [10] and Bro [11] 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, with precise information about the detection logic. Nevertheless, they are not able to detect zero-day attacks, i.e., attacks exploiting unknown vulnerabilities, for which no patch is available.

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. BotSniffer [12] utilizes statistical methods to detect C&C botnets, by observing coincident behaviors among hosts, like messages to servers, network scan or spam. Machine learning techniques can be classified in three families: (i) *Supervised* ones learn from a labelled dataset what constitutes either normal traffic or attacks (ii) *Unsupervised* approaches learn by themselves what is normal or abnormal – among them, MAWILab [13] finds anomalies by combining detectors that operate at different traffic granularities (the results against the MAWI dataset are in [13]); numerous works compare themselves to MAWILab, as for instance change-detection techniques [14], [15] (defining an anomaly as a sudden change compared to a model), and ORUNADA [16] (relying on a discrete time-sliding window to continuously update the feature space and cluster events); (iii) *Hybrid* approaches benefit from only a small part of labelled traffic, meant to be enough to learn from.

B. Large-scale intrusion detection

Coordinated attacks arise in multiple networks simultaneously and include large-scale stealthy scans, worm outbreaks and DDoS attacks [17]. 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 they 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 split in different subnetworks and in running a central detection engine against aggregated data, as shown in Figure 1a. Raw packets are transmitted from the flow collectors to the detection engine [18]. Solutions exist to avoid the collection traffic overhead, as done by Jaal [19], 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. There are three types of CIDSs depending on their communication architecture:

1) *Centralized* CIDSs are composed of several monitors that transmit the alerts to a central correlation engine, as illustrated in Figure 1b.
2) *Hierarchical* CIDSs use a pyramidal 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, [20] 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. [21] 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.

C. Alert correlation design

A common issue behind CIDS is its alerts correlation. Alert correlation algorithms can be divided into three categories [22]: (i) similarity-based algorithms 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 rely on a database of attacks definitions; (iii) probabilistic algorithms use similar statistical attributes to correlate attacks.

In our work, we leverage on the CIDS principles to build our system, and in particular centralized CIDS, and in terms of alert correlation we attempt at simplifying the search space using application ports, and more precisely destination ports. Up to our knowledge, we are the first to use the CIDS concept applied to port-centric detection. 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 [5], the authors propose a survey of the current methods to detect port scans. [23] aims to show the correlation between port scans and attacks. [24] analyzes the period during the release of a zero-day attack and its patching. [23], [24] 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.

III. Split-and-Merge Port-centric Network Anomaly Detection

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

A. 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, in order not to be influenced by weekly or daily variations. 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 provide in details the different steps of the 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):

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

TABLE I: Notations.

Data collection: incoming packets are collected in a single group of N_{batch} elements; only the following data is stored: the source and destination IP addresses and ports, the packet size and the TCP flags configuration.

Features computation: packet references are aggregated by destination port, filtering out those ports with less than N_{min} packets in order not to be influenced by very light traffic on one port. Features are then computed for the remaining ports.

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 I 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.

B. 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 II. *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. *portDivIndex* reflects the diversity in source ports. A variation in the *meanSize* feature suggests a change of packets nature, like crafted packets sent by bots. A variation in the *stdSize* feature can be caused by a change of 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 which kills other processes bound to the same port. Finally, a variation in *perSYN* implies an increase or decrease in port scan.

We denote the time series of feature i on N days (i.e., N_{days}) for port p as $f^p_{i,N} = (x^p_{i,1}, ..., x^p_{i,j}, ..., x^p_{i,N})$, with $x^p_{i,j}$

Feature	Description
srcDivIndex	Percentage of unique source IP addresses
destDivIndex	Percentage of unique destination IP addresses
portDivIndex	Percentage of unique source ports
meanSize	Mean packets size
stdSize	Standard deviation of packets sizes
perSYN	Percentage of SYN packets
nbPackets	Number of packets

TABLE II: Split-and-Merge features.

being the value of feature i for port p on day j. The detection modules dispose of features tables to save the vectors. The tables constantly contain N entries so that for every new capture, the former value is deleted and the new one added.

C. 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. Hence we model the time series $f^p_{i,N} = (x^p_{i,1},...,x^p_{i,N})$ 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:

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

In order to automatically detect a change in the time series of a port-specific feature, *Z-score* is a well-known simple statistical-based algorithm. More precisely, the Z-score 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. The larger the Z-score, the more unusual the value. For the given time series $f^p_{i,N}$, 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 of the standard deviation from the median, instead of the classical mean and the 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 $N+1$ 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 then 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].

Therefore, the modified Z-score is used to identify anomalies on all features, except *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

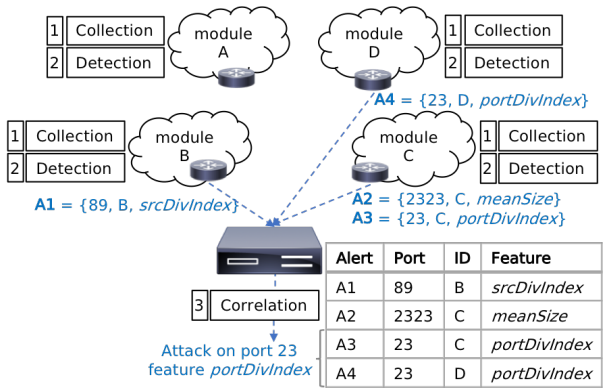


Fig. 2: Split-and-Merge example. Local modules run at different points in the network and send alerts to the central correlation module, which spots an anomaly on port 23.

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 $j \epsilon [1,N]$.

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 Figure 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.

D. 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 Figure 2, several detection modules send alerts to the correlation module; among them, two subnets report a change in the *portDivIndex* feature on port 23. Hence the correlation module spots an anomaly on port 23.

It is even better if similar anomalies have been noticed on the same port for several features. We define the *anomaly score* as the number of anomalies noticed for one port by all monitors and for all features; e.g., if one monitor detects anomalies on two features, and another on six features, the anomaly score is 8; or if nine monitors see an anomaly on a single feature, the anomaly score is 9. A score higher than the number of monitors is already concerning, but we can define several levels of concern based on the anomaly score.

When the correlation module identifies top-level anomalies, it warns all detection modules about the anomalous ports. Thus they will 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.

IV. Evaluation

In this section, we evaluate the performance of our split-and-merge anomaly detection process using real traffic traces. The detection accuracy is compared between two approaches: split-and-merge detection and a central detection made at a single point. We open source the source code used for split-and-merge analysis in [7].

A. Network traffic dataset

The WIDE project provides researchers with daily traces of a transpacific link, named the MAWI archive [6]. 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.

However, the anonymization key changes everyday, so there is a need to retrieve the same subnetworks each day. We notice that from one day to another, several anonymizations of the same subnetwork always share the same first byte. Using this tip and a mapping with subnetworks masks, we can retrieve all MAWI subnetworks whatever the period. Finally, we found 7 different AS totalling 17 subnetworks in MAWI. We use only the 9 subnetworks containing incoming traffic, and apply our algorithm as 9 distinct monitoring points.

B. Local anomaly detection

First, we launch the local detection modules 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 in the model is $N_{days} = 10$. Therefore, the initialization stage needed to compute the feature vectors the first time lasts 10 weeks, and the detection begins on June 9. These parameters are used for next simulations too, in Sections IV-C and IV-D.

Below on Figure 3 is an example of the modified Z-score evolution for the *srcDivIndex* feature on port 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.

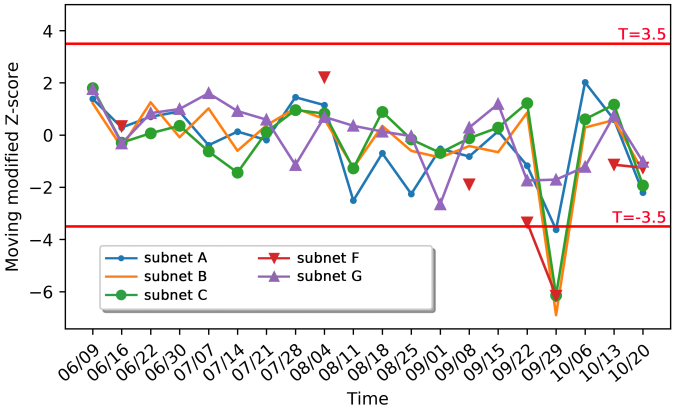


Fig. 3: Evolution of the modified Z-score of feature *srcDivIndex* on port 3389 over time (2016).

C. Split-and-Merge

This time, we run the split-and-merge detection on two different periods and analyze the occurrences of anomalies. The first period goes from March 31 to Oct. 20, 2016, and the second period goes from Nov. 26, 2017, to April 26, 2018, including the initialization stage.

Figure 4 and 5 show the occurrences of anomaly scores for these two periods. In the squares are given the numbers of ports with this anomaly score this day. The highest the score, the most colored the case to highlight significant detections. Note that the y-axis is not linear. We choose to describe only anomalies whose score is strictly higher than 15. For each of them, the port and the volume percentage of traffic towards it are indicated: this highlights that our algorithm notices anomalies even on ports with light traffic. Hereafter we further described the detected anomalies, along with our assumption about their cause.

2016 period. Four noticeable scores appear on Figure 4 depicting this first period.

i) The IoT Mirai botnet [1] 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 determines the environment and executes architecture-specific malware. The victim is now infected by Mirai 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. However, other IDSs did not detect it, while our algorithm detected the scans as soon as they arose. Indeed, the 26-score on Aug. 4 corresponds to the Mirai scan on port 23 and the 28-score on Sept. 15 corresponds to the Mirai scan on port 2323.

ii) The 20-score on June 30 corresponds to an exploit on port 6379 Redis, an in-memory key-value store used as a

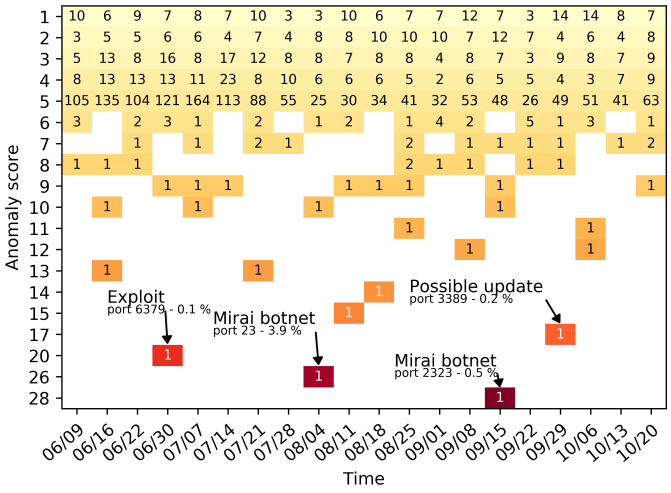


Fig. 4: Anomaly scores for the 2016 period. In the coloured squares are given the numbers of ports with this anomaly score this day.

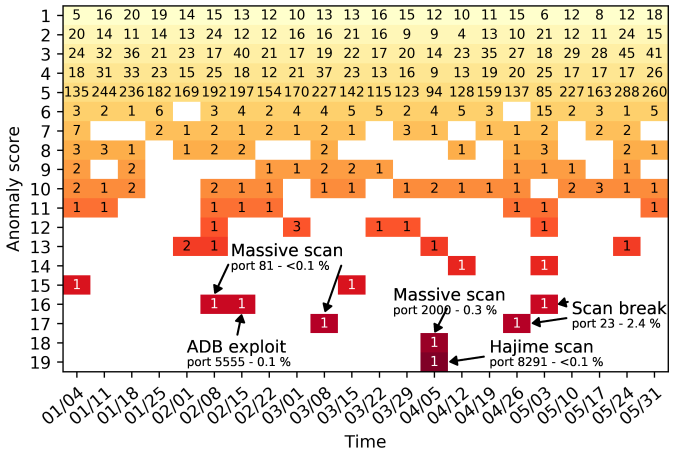


Fig. 5: Anomaly scores for the 2018 period.

database or a cache. This day a large SYN scan is detected from the same source IP address, targeting numerous hosts in several ASes of the MAWI network. This is either a large scan targeting the whole IPv4 space, through a tool like the ZMap tool [26] that performs Internet-wide network scan in under 45 minutes, or someone who wants to penetrate the MAWI network specifically. 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) The 17-score on Sept. 29 reveals an anomaly on port 3389, of the Remote Desktop Protocol (RDP). This proprietary protocol enables a user to remotely connect to another computer with a graphical interface over a network connection. This day, as for the Redis anomaly, the same source IP address exchanged TCP packets with numerous hosts in several AS of MAWI network, with the same source port. It may be an update by an administrator connected to all machines through RDP, or a large scan to penetrate the network.

2018 period. Seven scores distinguish themselves during this second period on Figure 5.

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 all MAWI subnetworks. This may be once again an Internet-wide network scan (e.g., by ZMap) or an attacker that targets the MAWI network 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 [27]. Hence this day, numerous IP addresses sent TCP SYN packets to various hosts in the MAWI network using different source port numbers, as seen for the Mirai botnet in 2016.

iii) An 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 network. 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 [28].

iv) An 19-score on Apr. 5 highlights a scan on port 8291, carried out by Hajime bots. Hajime [29] 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. 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.

v) Finally, 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* falls. 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 instead of port 23 because vulnerabilities on this port are progressively patched and devices are armed against possible exploits on this port.

D. Comparison between central and split-and-merge detection

To evaluate the benefits in splitting the detection process, we apply the same detection method under two different approaches and compare the results. The two approaches are:

Split-and-merge detection: in the first case, we split the detection between several detection modules, then we aggre-

gate the results. Here an anomaly is defined for one feature if at least two detection modules compute a modified Z-score higher than the threshold. Thus its anomaly score corresponds to the number of detection modules that detected that.

Central detection: here we apply the same detection method, but at a single point in the network. An anomaly is defined for one feature if the modified Z-score is higher than the threshold while working on packets of the whole network. In this case, the anomaly score is equal to 1, else 0.

The evaluations below aim to compare both approaches by considering two main aspects: the number of false positives and the capacity to detect attacks.

1) Evaluation of the number of false positives: First, we compare the number of anomalies between both approaches. Table III shows the number of anomalies found for each feature on a day randomly chosen, on Aug. 4, 2016.

Feature	Central	Split-and-merge
srcDivIndex	169	1
destDivIndex	175	3
portDivIndex	172	2
meanSize	170	2
stdSize	152	1
perSYN	16	1

TABLE III: Number of detected anomalies in central and split-and-merge approaches on Aug. 4, 2016.

For all features, the number of anomalies for the split-and-merge detection is significantly lower than for the central detection. Indeed it appears that the split-and-merge approach is very selective and enables to eliminate numerous alerts by keeping only the distributed ones. Operators at the Security Operation Center (SOC) of information systems are overwhelmed by alarms and thus tend to under utilize detection suites. Hence our solution could be considered in that case.

2) Evaluation of the detection accuracy of known attacks: We now compare the detection accuracy between split-and-merge and central detections on some attacks identified in the previous subsection. Tables IV to VI show the results. The two last columns show the total scores for all features. To get an idea of the rarity detection, the number of ports that notice a score higher or equal to the actual one is indicated in cases $\#ports \geq$S. The total number of ports corresponds to the number of ports obtained after the filter on N_{min} packets is applied. It is then different each day.

We indicate also for each attack if MAWILab detected it based on MAWILab anomaly reports. Indeed, these attacks are not traditionally detected by current IDSs for several reasons:

- Current IDSs work on small variations of traffic, generally using time-sliding windows that last a few seconds or less. Therefore, they fail at detecting major but progressive changes in the traffic.
- They aggregate packets either by source IP or destination IP addresses, while these attacks focus on ports.
- ISP-scale attacks stay invisible at a single network scale.

Mirai botnet. Table IV presents the results of the Mirai botnet detection, on port 23 on August 4^{th}, 2016. With the central detection, there are anomalies for five features, a score noticed for 132 ports in total. For split-and-merge detection, the score is equal to 26 out of 51, a score noticed only for this port (see Figure 4), so this is very rare and visible by the network administrator. The Mirai case is interesting since the Telnet port was already massively scanned before the Mirai scan. Therefore, it is not identifiable by sudden high traffic on this port, but it requires the use of precise features to notice a change. Moreover, MAWILab did not detect the scan.

ADB exploit. Table V presents the results of the ADB exploit detection. The central detection produces a total score of 4 out of 6. This score is not relevant enough since 455 other ports registered a score higher or equal to this one. On the contrary, the split-and-merge detection produces a score of 16 out of 51, a score highly relevant because noticed for only this port (see Figure 5). The MAWILab report of this day is not available, so we are not able to check if MAWILab detectors detected the attack.

Hajime botnet. Table VI presents the results of the Hajime botnet detection. The central detection produces a total score of 5 out of 6. Despite its high value, this score cannot be interpreted as relevant and distinct enough since 203 ports registered a score higher or equal to this one. On the contrary, the split-and-merge detection produces a score of 19 out of 51, which is truly relevant because noticed for only this port (see Figure 5). Furthermore, this day MAWILab report did not notify anything relatively to port 8291.

For all attacks, we observe that the split-and-merge detection is far more efficient than the central detection. Therefore there is a benefit in detecting anomalies by subnetwork and then aggregating low-level alerts, rather than performing anomaly detection with packets from the whole network.

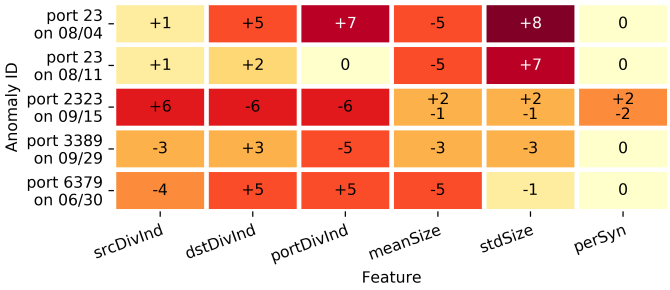


Fig. 6: Characterization of anomalies in 2016. In the squares are given the numbers of subnetworks that detected a modified Z-score over (+) or below (-) the threshold for each feature.

3) Detailed characterization of major anomalies: Each anomaly is unique regarding the port usage before the anomaly and the type of change, thus any classification of anomalies would be non-exhaustive. However, looking at features conjointly enables insights on the anomalies. Figure 6 shows on the y-axis the most noticeable anomalies discovered in 2016, along with their destination port and date. For each one, the number of monitors that detected an anomaly for the feature in the x-axis is given in the coloured squares. We propose

Detection	*srcDivIndex*		*destDivIndex*		*portDivIndex*		*meanSize*		*stdSize*		*perSYN*		**6 features**	
	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	**S**	**#ports≥S**
Central	1/1	169/20267	1/1	175	1/1	172	1/1	179	1/1	152	0/1	20267	**5/6**	**132**
Split-Merge	1/9	41/17781	5/9	1	7/9	1	5/9	1	8/9	1	0/9	17781	**26/51**	**1**

TABLE IV: Evaluation on port 23 (Mirai botnet) the 4^{th} of August, 2016 for central and split-and-merge detections.

Detection	*srcDivIndex*		*destDivIndex*		*portDivIndex*		*meanSize*		*stdSize*		*perSYN*		**6 features**	
	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	**S**	**#ports≥S**
Central	1/1	536/25400	1/1	515	0/1	25400	1/1	487	1/1	457	0/1	25400	**4/6**	**455**
Split-Merge	3/9	3/21999	3/9	3	4/9	1	4/9	1	1/9	218	1/9	14	**16/51**	**1**

TABLE V: Evaluation on port 5555 (ADB exploit) the 15^{th} of February, 2018 for central and split-and-merge detections.

Detection	*srcDivIndex*		*destDivIndex*		*portDivIndex*		*meanSize*		*stdSize*		*perSYN*		**6 features**	
	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	S	#ports≥S	**S**	**#ports≥S**
Central	1/1	256/25400	1/1	253	1/1	254	0/1	25400	1/1	224	1/1	30	**5/6**	**203**
Split-Merge	4/9	2/21999	5/9	1	5/9	1	1/9	122	3/9	4	1/9	16	**19/51**	**1**

TABLE VI: Evaluation on port 8291 (Hajime botnet) the 5^{th} of April, 2018 for central and split-and-merge detections.

hereafter the description of three anomalies.

i) The anomalies on port 23 represent the Mirai scan. There is no anomaly in *perSYN*, as this port was already known for its vulnerabilities before Mirai, and thus largely scanned. *destDivIndex* rises in a majority of subnetworks as there are more hosts targeted by a scan than before. *srcDivIndex* rises as well but not over the threshold, and thus does not appear here. Also, *meanSize* decreases because crafted packets have a lower size than regular ones. *stdSize* rises, as the difference between Mirai and non-Mirai packets sizes expands. This is not shown in the picture but some days later, *stdSize* falls in several subnetworks as Mirai tends to stop other processes bound to this port, thus the diversity among packets is lower.

ii) The qualification of the Mirai scan on port 2323 is quite different. This time, *perSYN* rises because the port was not popular before and is now highly scanned. *srcDivIndex* rises as well since there are far more infected bots targeting port 2323 than previously.

iii) Finally, the RDP anomaly characteristics on port 3389 depict another kind of anomaly. We notice that *srcDivIndex* falls in several subnetworks while *destDivIndex* rises, therefore these are numerous connections coming from the same host. We also note that the diversity among packets is lower by observing *stdSize*, and that anomalous packets have a lower size than regular ones with *meanSize*.

V. Conclusion

Our split-and-merge port-driven anomaly detection procedure focuses on major attacks targeting servers and connected objects around the world. We propose a collaborative scheme to detect main changes in the usage of ports. The detection results are very promising, since our algorithm detected a number of world-wide attacks from 2016 to 2018, including Mirai. It is also able to provide precise details about the anomalies to help the administrator characterizing them. In contrast, current IDSs, among which the notorious MAWILab, have not detected them. Moreover, we demonstrate that our algorithm produces a very low number of false positives. At the same time, we provided a better knowledge of the MAWI dataset, raising the main attacks that happened the last three years, in open source.

In the future, we plan to implement a system that runs in near real-time. Hence, instead of running the algorithm at an appropriate frequency, we would have to remove the seasonal trend from time-series. This way, a dimensioning of our algorithm in terms of memory, storage and computing power could be elaborated too. We also aim at doing a better characterization of attacks by analyzing conjointly the features evolution. Finally, one can refine the detection accuracy by analyzing how the parameters impact the results.

References

[1] M. Antonakakis, T. April *et al.*, "Understanding the Mirai botnet," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2017, pp. 1093–1110.

[2] (2016) Mirai source code. [Online]. Available: https://github.com/jgamblin/Mirai-Source-Code/

[3] (2018) Netscout threat intelligence report. [Online]. Available: https://www.netscout.com/threatreport

[4] E. B. Alexander Khalimonenko, Oleg Kupreev. (April 2018) DDoS attacks in Q1 2018. [Online]. Available: https://securelist.com/ddos-report-in-q1-2018/85373/

[5] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Surveying port scans and their detection methodologies," *The Computer Journal*, vol. 54, no. 10, pp. 1565–1581, April 2011.

[6] (2018) MAWI working group traffic archive. [Online]. Available: http://mawi.wide.ad.jp/mawi/

[7] (2018) Source code for split-and-merge detection algorithm. [Online]. Available: https://github.com/a-blaise/split-and-merge

[8] R. Fontugne, P. Borgnat *et al.*, "Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the International COnference on emerging Networking EXperiments and Technologies (Co-NEXT)*, 2010.

[9] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[10] Snort - network intrusion detection & prevention system. [Online]. Available: https://www.snort.org/

[11] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, Dec 1999.

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

[13] MAWILab database. [Online]. Available: http://www.fukuda-lab.org/mawilab
[14] A. G. Tartakovsky, A. S. Polunchenko, and G. Sokolov, "Efficient computer network anomaly detection by changepoint detection methods," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 1, pp. 4–11, Feb 2013.
[15] C. Callegari, S. Giordano, and M. Pagano, "Entropy-based network anomaly detection," in *Proceedings of International Conference on Computing, Networking and Communications (ICNC)*, 2017.
[16] J. Dromard, G. Roudiere, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, March 2017.
[17] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124–140, Feb 2010.
[18] P. K. Shanmugam, N. D. Subramanyam *et al.*, "DEIDtect: towards distributed elastic intrusion detection," in *Proceedings of the ACM SIGCOMM workshop on Distributed cloud computing (DCC)*, 2014.
[19] A. Aqil, K. Khalil *et al.*, "Jaal: Towards network intrusion detection at isp scale," in *Proceedings of the International COnference on emerging Networking EXperiments and Technologies - (CoNEXT)*, 2017.
[20] D. Singh, D. Patel *et al.*, "Collaborative IDS framework for cloud," *International Journal of Network Security*, vol. 18, pp. 699–709, 2015.
[21] J. Francois, I. Aib, and R. Boutaba, "FireCol: A collaborative protection network for the detection of flooding DDoS attacks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1828–1841, Dec 2012.
[22] S. A. Mirheidari, S. Arshad, and R. Jalili, "Alert correlation algorithms: A survey and taxonomy," in *Cyberspace Safety and Security*, 2013, pp. 183–197.
[23] S. Panjwani, S. Tan *et al.*, "An experimental evaluation to determine if port scans are precursors to an attack," in *2005 International Conference on Dependable Systems and Networks (DSN)*, 2005.
[24] C.-N. Kao, Y.-C. Chang *et al.*, "A predictive zero-day network defense using long-term port-scan recording," in *2015 IEEE Conference on Communications and Network Security (CNS)*, Sep 2015.
[25] B. Iglewicz and D. Hoaglin, "How to detect and handle outliers," in *The ASQC Basic References in Quality Control: Statistical Techniques*, P. Edward F. Mykytka, Ed., 1993, vol. 16.
[26] E. W. Zakir Durumeric and J. A. Halderman, "ZMap: Fast internet-wide scanning and its security applications," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2013.
[27] (Feb 2018) ADB.Miner: More information. [Online]. Available: http://blog.netlab.360.com/adb-miner-more-information-en/
[28] (2018) IoTroop botnet: The full investigation. [Online]. Available: https://research.checkpoint.com/iotroop-botnet-full-investigation/
[29] S. Edwards and I. Profetis. (2016) Hajime: Analysis of a decentralized internet worm for IoT devices. [Online]. Available: https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf