Received August 20, 2020, accepted August 26, 2020, date of publication August 31, 2020, date of current version September 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020507

A Distributed Architecture for DDoS Prediction and Bot Detection

BRUNO MARTINS RAHAL, ALDRI SANTOS, AND MICHELE NOGUEIRA
Center for Computational Security sCience (CCSC), Department of Informatics, Federal University of Paraná (UFPR), Curitiba 81531-980, Brazil

Corresponding author: Michele Nogueira (michele@inf.ufpr.br)

This work was supported in part by the National Council for Scientific and Technological Development (CNPq/Brazil) under Grant #309129/2017-6 and Grant #432204/2018-0, and in part by the joint NSF and RNP HealthSense project under Grant #99/2017.

ABSTRACT Distributed Denial-of-Service (DDoS) attacks disrupt servers, services and the network, overloading the target resources and denying normal traffic. In order to defend from this attack, mitigation actions usually overprovision and sinkhole malicious traffic. Sooner the attack is detected, better is the mitigation. Hence, we advocate for using a prediction technique aiming to anticipate actions against the possible attack, before it effectively starts. Then, this article contributes to advance the state-of-the-art presenting a distributed architecture that identifies early signals of a possible DDoS attack and detects bots composing a botnet. The architecture identifies the malicious actors (bots) participating in the attack. The bot detection technique is triggered by the prediction of DDoS supported by early signals. Prediction identifies signals of attack on the network before it reaches advanced stages. Based on the metastability theory, it provides unsupervised statistical learning and identifies the imminence of DDoS attacks. The botnet detection is challenging because of the high dimension of data involved and because of resource constraints (memory and processing) in network devices. Network devices are clustered based on features extracted from the traffic and based on the causality between devices. Detection is performed per cluster. Performance evaluations took as input the CTU-13 Czech Republic University, CAIDA and Botnet 2014 datasets, efficiently detecting the bots in the dataset with an accuracy of 99.9%.

INDEX TERMS Attack prediction, botnet detection, cybersecurity, network traffic analysis.

I. INTRODUCTION

Botnets pose a huge threat to Internet users and devices. Given the easiness in accessing the Internet, social and commercial relations rely on the online world. Services and social relation moved to the Internet (as banking, commerce, health, interpersonal relations, news and recreation) [1], [2]. With more people and services connected and generating information, data has become a valuable asset. However, as in the physical world, there are bad actors. Malicious software infects different devices, transforming them in robots (bots) that work in networks (botnets) targeting users, services and devices, looking for generating profit to operators and developers of these malicious softwares.

Once a malicious software infects a machine, it may execute commands controlled by a remote attacker. The remote attacker tries to breach and compromise as many devices as possible, including each bot into a botnet. Botnets are able to perform much more powerful attacks than a single machine would do. These bad actors, the bots, harm data confidentiality, integrity, and/or data and service availability.

The associate editor coordinating the review of this manuscript and approving it for publication was Moayad Aloqaily.

A well-known attack lies in Distributed Denial-of-Service (DDoS), in which an attacker (botmaster) commands all bots of a botnet to flood the bandwidth and target resources. Hence, the resources of the targeted system are denied to legitimate users. The largest DDoS attack ever recorded has targeted Amazon AWS reaching a stunning 2.3 Tbps [3]. Detecting, controlling, and restraining DDoS attacks pose a huge challenge to systems and network administrators.

Measures can be taken to avoid that a device becomes part of a botnet (hence avoiding its participation in such attacks), such as patching devices with security updates, not using default credentials on devices and services, using antivirus and not installing unknown software. In local networks, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) detect devices that might have been breached and became part of a botnet. Internet Service Providers (ISPs) also want to identify malicious actions on their networks, avoiding bad reputation and lowering costs (*e.g.* DDoS attacks might incur in high bandwidth usage, hence higher costs). However, once a botnet starts its attack, in particular, a DDoS one, there is no much to be done by the targeted system [4]. Mitigation approaches have sought to reduce the impact of ongoing DDoS attacks. Overprovisioning and sinkholing malicious traffic are some measures that can be taken, but they are more effective the sooner the targeted system knows it has to act.

This article presents an architecture to jointly predict DDoS attack and detect bots. The architecture is funded on the fact that as early as a system has indications about the imminence of the DDoS attack and detects its bots, better the system can act against the attack [4]. The proposed architecture operates hierarchically in two levels: local and Internet to anticipate botnet signals and prevent DDoS flooding attacks generated by botnets. On the local level, the early signals of a DDoS attack are identified before it reaches advanced stages. These signals are based on leading indicators as return rate, autocorrelation, coefficient of variance, and skewness, as in [5]. In contrast with other works, leading indicators are used in this article to predict trends of DDoS attacks. On the Internet level, the proposed system follows a hybrid bot detection method. This hybrid method uses both clustering techniques and signal processing on graphs to analyze traffic on the network. It performs statistical analysis on the traffic, selecting features that better characterize the analyzed traffic. Next, it uses the selected features and an unsupervised machine learning technique to cluster devices with similar behavior. Then, it identifies devices that behave like bots, in the clusters, separating them from benign nodes.

For both local and Internet levels, the proposed architecture has been instantiated in a case study. In the case study, the system based on the proposed architecture captures the traffic sent from and to devices and uses it to build and analyze time series. Time series comprises data from features selected from the traffic. On the local level, a time series comprises data from the network load as packet size through time. On the Internet level, several other features are extracted to support statistical analysis, clustering, and bot detection. Results show a set of behavior that can predict trends of a disruptive change in the network state. This disruption, detected by the early signals of a DDoS, triggers the bot detection mechanism on the Internet level that, in its turn, performs a robust and more detailed analysis on the traffic, looking for bots.

Evaluations employ CTU-13, CAIDA and Botnet 2014 datasets. These datasets are commonly employed in the literature for botnet detection and present scenarios with DDoS attacks. Hence, the evaluated system receives as input 60-second windows of network traffic. Based on the 60-second windows the leading indicators are calculated. Once the system points out a critical state change, it extracts and selects features. Hence, clusters are built and the system analyzes the devices in each cluster. Finally, detection accuracy are derived from the number of true-positives, true-negatives, false-positives and false-negatives. Results show that, based on the early signals, the system could predict a DDoS attack. This early prediction triggers the further investigation and bot detection. Results show accuracy and feasibility in the bot detection.

The main contribution of this article lies in a distributed architecture to both predict and detect botnets. The architecture starts with a local network traffic analysis supported by leading indicators, i.e., statistical measures calculated from simple data of the network traffic such as packet size or packet transmission frequency. Such leading indicators support the detection of early signals of DDoS attacks caused by botnets. Also, the use of hybrid features (extracted from traffic and graphs representing the identified structure of interrelations between nodes) provides a wider number of inputs that can characterize the traffic. Traffic features are not only extracted but also selected based on the relevance to detect botnets. Lastly, we employ a deterministic method to detect bots in nodes that are clustered, providing a better performance to the detection, in the internet level, that can combine traffic from multiple local networks.

This article proceeds as follows. Section II reviews related works. Section III describes the architecture of the solution, its inputs and outputs and how the data is handled on multiple levels to predict and detect bots causing DDoS attacks. Section IV describes the early signals technique and the bot detection method, and how we use them to build the proposed architecture. Section V details the evaluation methodology and discusses the results. Section VI concludes the article.

II. RELATED WORK

Early warning systems have been researched in many fields. Recently, researchers started focusing on networking and computational systems. In [6], Ramaki and Atani presented a survey of architectures and techniques for early warning threats in Information Technology (IT). The authors classified the early warning systems (EWS) in commercial or under research and development. They pointed out a set of challenges, such as data collection, data correlation, and post-event data correlation [6]. The authors reinforced the need of designing proactive solutions to predict threats and attacks before they occur, using data analytics.

Other studies have particularly tackled the problem of developing early warning techniques for DDoS attacks [7]–[9]. In [8], Xiao, Chen, and He proposed a cooperative system to produce warning signals. The system is based on a Bloom filter technique. The authors goal lies in reducing storage and computational resources consumption. In [7], Tsai, Chang and Huang presented a multi-layer system based on time delay neural networks. It is a cooperative system in which each device in the network monitors its neighborhood. In a timely manner, the device sends the collected data to an expert module which analyzes all collected data and attempts to match received data with known DDoS patterns.

In [9], Korczyński *et al.* presented a cooperative and self-organized anomaly detection system inspired by honey bee colonies. Their goal was to provide dynamic thresholds to detect anomalous patterns in network traffic to improve early intrusion detection that could assist attack mitigations. In [5], Pelloso and Nogueira employed leading indicators to detect the presence of volumetric DDoS bots on the network. Their work, however, did not focus on the identification of the nodes that might be causing the attack, only coming to the conclusion that leading indicators may provide early signals that a DDoS attack might occur in the near future. In [10], Moussa, Nogueira, and Guedes presented α-*investing*$^+$, an algorithm for statistical analysis of network traffic features. The algorithm identifies relevant features on the traffic. They presented a feature selection method for streaming DDoS attack data. The method is able to sequentially add and test features, performing the selection in an ongoing attack. This can be a technique for DDoS detection based on the passive monitoring of the network.

In [11], Daya, Salahuddin, Limam, and Boutaba proposed a machine learning bot detection system. They presented a two-phased, graph-based bot detection system which leverages both unsupervised and supervised machine learning techniques. The first phase prunes presumably benign hosts, while the second phase achieves bot detection with high accuracy. They compare multiple techniques, both on the unsupervised and supervised machine learning phases. Similarly, in [12], the authors employed a self-organizing map (an unsupervised machine learning technique) to cluster nodes in the network based on graph based features. They employ Support Vector Machine (a supervised machine learning technique) to further classify bots. In [13], the authors employed protocol attributes and frequency distributions to characterize host behaviour, and learning and classification techniques, based on distances to labeled clusters.

The state-of-the-art in networking is always evolving. Software-Defined Networks (SDN) is a new approach to design networks. In [14], a DDoS attack detection system for SDNs is proposed using two levels of security. First a signature detection based is used. Then, machine learning techniques (SVM - Support Vector Machine and DNN - Deep Neural Network) detect attacks on the network. Similarly, another trend in networking is cloud computing. In [15], Karan B. Virupakshar, Manjunath Asundi, Kishor Channal, Pooja Shettar, Somashekar Patil, D. G. Narayan used Decision tree, KNN (K-nearest neighbor), Naive Bayes and Deep Neural Network (DNN), machine learning techniques, to detect DDoS attacks on cloud computing infrastructure, with OpenStack integrated firewall and raw socket programming for monitoring the network traffic.

In [16], Mei and Moura proposed a deterministic framework which identifies inter-relations in time series and intra-relations in time series. This technique, named Causal Graph Process (CGP), derives a low dimensional representation from a multidimensional feature set. This is represented by a directed and weighted graph that can possibly capture causal relations on the traffic. Their work, however, did not focus on network traffic analysis or botnet detection. We apply this technique to early detect botnets and causal relations that bots might exert into their victims.

III. ARCHITECTURE

Prediction is the act of knowing in advance about an event, based on special knowledge [17]. In nature, diseases are avoided with early identification of risk factors, offering clinical setting, as in Acute Kidney Injury [18]. The same takes place on cardiovascular diseases where diabetes mellitus is associated with a significant increase in risk for cardiovascular disease, but this risk is not uniform [19]. Even on infectious diseases, evaluation of internal and external drivers, that trigger emergence events, is increasingly helpful for predicting these events [20].

The goal of predicting lies in "get ahead of the curve", allowing individuals and health system to react accordingly and promptly respond to outcomes. The same happens on network traffic, where trends are observed and predicted. When it comes to detecting anomalies, algorithms can bundle normal data behavior and infer anomalous behavior. These methodologies have a very high percentage of false positives and may disrupt the performance of the underlying system, if it triggers a very high number of false diagnostics (imagine treating everyone with possible indicators of Acute Kidney Injury as it will result in morbidity and mortality).

Instead of treating every possible patient with early risk factors, these early signals indicate the necessity of further analysis in patient condition. In health systems, blood tests and tomography assist in critical cases, showing or not the necessity of an investigative surgery. But these actions are triggered by early indicators in the individual health system. Early indicators focus on a fraction of organs but further investigations are broader and may focus on the whole system (the human body). Initial investigations may focus on a particular organ that is not the cause of malfunction, needing to target more organs to find bad actors.

This article presents an architecture to jointly predict DDoS attack and detect bots. The architecture is funded on the fact that as early as possible the targeted system knows about the imminence of the DDoS attack and detect its bots, better the system can act against the attack [4]. The proposed architecture operates hierarchically in two levels: local and Internet to anticipate botnet signals and prevent DDoS flooding attacks generated by botnets. On the local level, the early signals of a DDoS attack are identified before it reaches advanced stages. These signals are based on leading indicators as return rate, autocorrelation, the coefficient of variance, and skewness, as in [5]. In contrast with other works, we apply the leading indicators in this article to predict trends of DDoS attacks.

We follow this analogy and we propose an architecture that applies a two-tier approach, i.e., local and Internet levels, to predict attacks and detect bots. The architecture is funded on the fact that as early as possible the targeted system knows about the imminence of the DDoS attack and detects its bots, better the system can act against the attack [4]. Figure 1 presents the five modules that compose the architecture: Data Collection, Feature Extraction, Processing, Analysis and Notification. The **Data Collection Module** integrates network sensors and preprocessing components. The first component continuously captures the traffic from the network, using a sensor or a sniffer. The second component preprocesses the collected network traffic, transforming binary

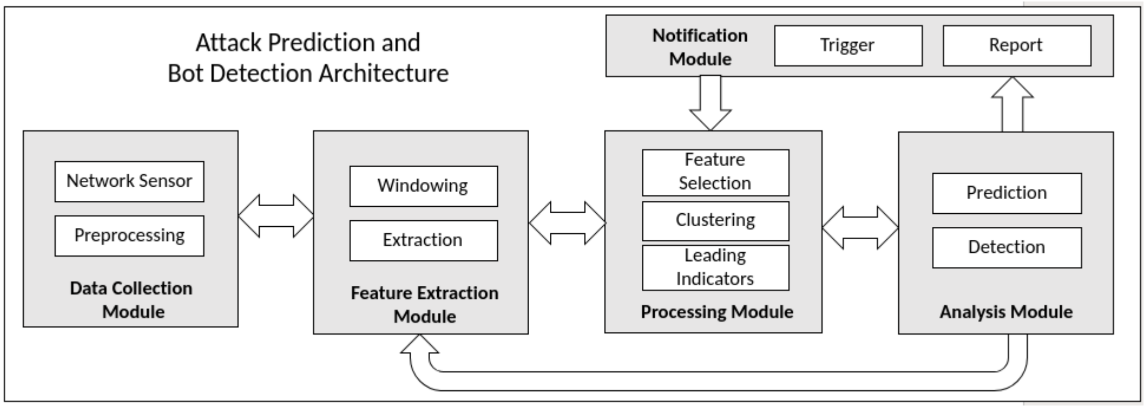

FIGURE 1. Architecture for early signals calculation and bot detection.

data in human-readable data, in a way to offer a relevant input to the **Feature Extraction Module**. This module reads the preprocessed data and splits it into windows, i.e., sets of data generally determined by time or by the quantity of elements (e.g., number of network packets). This module also performs the extraction of features that characterize the traffic in that window. It gathers information from traffic packets and its protocol, summarizes and counts some of those data, building a feature that is representative of that traffic.

The **Processing Module** performs feature selection, clustering and traffic assessment by leading indicators. The Processing Module uses data from the windows generated on the Feature Extraction Module and calculates the selected features, the clusters and the leading indicators. The Feature Selection Component identifies the most relevant features in network traffic and outputs a set of suitable features to perform the bot detection. The Clustering Component groups network nodes with similar behavior, according to an unsupervised machine learning technique. Leading indicators offer information about a change in network state. Calculating these indicators is fast and less costly than doing a deep analysis. This may indicate a DDoS, without, however, doing an analysis of each calculation output.

The **Analysis Module** continuously performs attack prediction and bot detection. It has as inputs the leading indicators calculated by the Processing Module. The clusters defined by the Processing Module serve as the basis for bot detection. Both the Processing Module and the Analysis Module communicate with the Notification Module. This module is responsible for triggering the Detection Analysis once an attack as DDoS is predicted. The Analysis Module adjusts window size, from the Feature Extraction Module, for fine tuning, when necessary. The **Notification Module** also reports results to system administrators or an automated system that acts into the network.

Once a state change on the network traffic is verified on the Prediction Component, the Notification Module triggers the Botnet Detection Component that detects which devices on the network that have a suspicious behavior. Four leading indicators warn for state change in the network. This state change may provide an indicator of the occurrence of a DDoS in the future. As calculating leading indicators is faster and lightweight, it is done on local networks with the computational resources of a domestic router, for example. Once the state change is detected, indicating the imminence of a possible DDoS, then an extended analysis is triggered to detect the malicious actors that may harm the network.

The Botnet Detection Component, once triggered, uses the collected data from the networks and processed, to identify the bots on the network causing a DDoS attack. This further analysis is broader, with more features being analyzed and targeting more devices. It uses the Internet traffic to classify and identify malicious devices. The Bot Detection Component operates not only on the traffic of a single local network, but with Internet traffic, compounded from multiple local network collected traffic. It can be triggered from a change on the state of any of the local network behavior. Once it is triggered, it uses collected traffic and processed data, as input, and classifies devices as bots or not. The detection acts in each cluster, possibly in parallel, analyzing causality between network nodes. The bot detection analysis is triggered once a change on the state of the network is detected on the Prediction Analysis Component. The output for the Detection Analysis Module is the classification of devices in bots or not.

Combining these two techniques, prediction and detection, as in health systems, this system is able to early identify suspicious activity, with a fast method, easy on computational resources. Detection is broader and deeper, needing a more powerful computational device to operate its detection. Using prediction and detection, in a distributed fashion, it is possible to identify bots on a reliable and scalable way, without the overhead of continuously performing a costly bot detection.

IV. CASE STUDY

This section details a case study for the proposed architecture. It starts describing how to collect network data and preprocess it into time series that are subsequently processed and analyzed. Next, it details the employed prediction mechanism (i.e., early signals on local networks) and bot detection. With data from local network, the implemented mechanism in the Prediction Component has as input the leading indicators calculated on the built time series and has as output a signal that indicates a significant change in the network state. When this change occurs, in any local network, it triggers the implemented mechanism in the Detection Component, through the Notification Module. The bot detection mechanism combines traffic from multiple local networks to detect malicious devices. The Data Collection Module stores the data from the traffic so it can be further analyzed.

Specific behaviors on a network may indicate a volumetric flooding-based DDoS attack. To measure such behaviors, this case study employs a set of classical statistical leading indicators: return rate, autocorrelation, coefficient of variance, and skewness [21]. Leading indicators (calculated by the Processing Module) are calculated to identify metastable states changes (verified by the Analysis Module) and identify such attacks. Leading indicators are well known in Statistics [21]. Here, they offer a prediction about a disruptive transition on the network state. This is based on certain properties from metastability theory that uses leading indicators to predict trends of volumetric DDoS attacks [5].

The Bot Detection Component takes as input the network traffic (from the Data Collection Module) and the extracted features from packet headers (from the Feature Extraction Module). This component uses only the selected features for cluster the nodes (Processing Module). The Feature Selection Component (in the Processing Module) identifies the most relevant features in the network traffic and outputs a set of suitable features to detect bots. The Clustering Component (in the Processing Module) groups network nodes with similar behavior, according to an unsupervised machine learning technique. The Detection Component (in the Analysis Module) acts in each cluster, preferentially in parallel, analyzing causality between network nodes, by a deterministic method.

Figure 2 presents an overview of the distributed architecture, when running on multiple devices, on the local and Internet levels. The figure shows data being collected and analyzed on multiple local network devices and the trigger for botnet detection in a centralized device.

Traffic from local network devices (desktops, smart TVs, smartphones, tablets, notebooks) are forwarded to a router, a hub or a modem, that, being part of the network, captures the traffic in real time. This traffic is collected by the device through a network sensor. Data (binary data) is pre-processed, enabling further analysis, in human-readable format. All the pre-processed traffic is split into windows, that is analyzed and it characterizes the time frame. From the traffic data of a window, leading indicators are calculated and the prediction component may indicate a critical transition in the network or not. If a critical transition is detected, the local network device triggers the botnet detection system. This prediction is light on computational resources and provides early signals of a possible DDoS attack in the network.

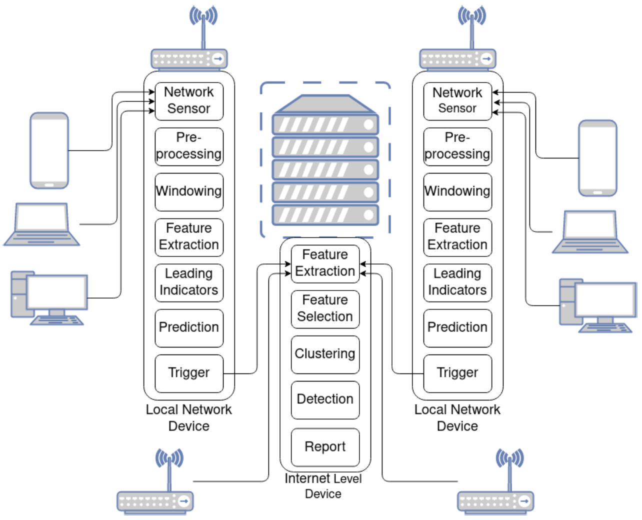

FIGURE 2. General overview on multiples devices.

Once a critical transition is detected, in any local network state, the goal is to investigate possible bots on that traffic. New features, different from the local network features, are extracted. These features require more computational power to be extracted. After that, features are selected, being only the statistically relevant features used for clustering. After nodes are grouped in clusters, detection indicates nodes that may or may not have a bot behavior in that window (triggered by the local network device). The result of the detection is then reported to a network administrator or, possibly, an automated tool that will act in the network, avoiding damage that the bot may cause.

A. THE DATA COLLECTION AND FEATURE EXTRACTION MODULES

Both the early signals and the botnet detection mechanism uses time series as input. A time series is a series of data points indexed by time. In order to predict and detect bots, the network is monitored and the module and its components act (*i*) collecting data from the network, (*ii*) defining window size and (*iii*) extracting features. Packets are collected from the network with a network monitor or sniffer and the raw packet (bits) are preprocessed into relevant human-readable data to be stored. Windows are used to segment data streams that can be analyzed in a scenario where data is continuously generated. Common segmentation methods are sliding window approach with a fixed window size and timely fixed windows. For that, a window size must be defined based on the evaluated data features. Two ways of sizing the windows are fixing the duration for the window in a time unit (*e.g.* a window is 60 seconds of data collection) or fixing the amount of collected samples (*e.g.* 1000 network collected packets). Once a window is defined, its size could be dynamically adjusted, matching a given network requirement (the Analysis Module could indicate the need of a bigger or smaller window). This dynamic adjustment makes feasible the analysis of a large data amount, without compromising the detection performance [22].

Each packet size is observed from the perspective of the packet destination, and it is indexed per time unit t (a time series). Each packet is collected sequentially, hence each packet is labeled by a unique timestamp. For each time window, observations are organized in a matrix, in which cells contain the packet size per time unit (matrix column) and packet destination (matrix row). The packet size is chosen because of the general behavior of a volumetric DDoS attack. The packet size is a network feature susceptible to attacker manipulation. Attackers can easily produce fake network packets containing bigger sizes in order to overload the network bandwidth. Hence, this feature is used to analyze and investigate the network behavior and patterns over time.

After collecting data, the Feature Extraction Module will process the data and extract information from the traffic so it can be further processed and analyzed. A feature is a representation of the collected data. Each possible feature represents the data (hence, the traffic), in a certain way. For example, the sum of packet lengths on the network may represent the volume of data that is flowing in the network. Feature extraction occurs until a window limit is reached (may it be by time or sample). Once a window is filled, it can be processed to bring knowledge about the network, based on the features, and be analyzed. A feature may or not be linked to a particular node: it can represent the overall behavior of the network being analyzed or the behavior of a given node.

For the leading indicators calculation, the interest is analyzing the overall behavior of the network, not of particular nodes. For the bot detection (and its Processing Components), network traffic is collected and features extracted from it to describe the node on the network and its behavior. Instead of building one time series (one feature) that describes the network traffic, for detecting a bot, a higher number of features is required than to calculate the leading indicators. It is possible to extend the use of as many features as a system should require to improve the detection performance. Features in the Bot Detection Component are those that better characterize a node. Similarly to the leading indicator calculation, data is collected in fixed time or quantity window size. But for better analyzing bot behaviors, instead of counting the number of packets, a different approach is required based on the number of nodes (e.g. a window must have traffic from 50 nodes). This happens because machine learning techniques group and compare the behavior of nodes, instead of packets. As each node is associated with one time series for each feature, these time series are further processed and used for bot detection.

1) FEATURE EXTRACTION

This module extracts features from a graph built from the traffic and other features are created straight out from the network traffic. Extracting and selecting relevant features for clustering nodes are essential for accuracy in detecting bots [23]. Features play a key role on how the clusters are composed, being decisive to detection accuracy. Two types of features are extracted from the traffic: (*i*) features from the traffic itself and (*ii*) features from relations between nodes.

The features are derived from packet headers. The employed features are summarized in Table 1. The *protocol quantity* feature counts how many protocols each node uses to communicate with other nodes. As ICMP, DNS, TCP and UDP are common, their percentage in relation to the total of used protocols lead to the following features: *ICMP Percent*, *DNS Percent*, *TCP Percent*, and *UDP Percent*, respectively. Certain botnet-based generated attacks may use primarily some of these protocols (*e.g., Ping Flood* – ICMP protocol and Dyn Attack – DNS protocol). Hence, these percentages are relevant to the analysis. For other protocols, the feature *Percent Others* records this information.

TABLE 1. Features extracted from the network.

Features	Description
Protocol Quantity	Quantity of protocols each node uses
Average TTL	Average *Time to live* of the sent packets
TCP Window Size	Size of the windows in the TCP protocol
Percent TCP	Percentage of packets that use TCP protocol
Percent UDP	Percentage of packets that use UDP protocol
Percent DNS	Percentage of packets that use DNS protocol
Percent ICMP	Percentage of packets that use ICMP protocol
Percent Others	Percentage of packets that use other protocols
Source Privileged Ports	Quantity of ports on the source that are privileged (<1024)
Source Not Privileged Ports	Quantity of ports on the source that are not privileged (>1024)
Destination Port Quantity	Quantity of destination ports contacted by the node
Frame Length	Packet's frame length (Layer 2 on the OSI model)
In Degree	Graph feature described in the text
Out Degree	Graph feature described in the text
In Degree Weight	Graph feature described in the text
Out Degree Weight	Graph feature described in the text
Node betweenness centrality	Graph feature described in the text
Local clustering coefficient	Graph feature described in the text
Eigenvector centrality	Graph feature described in the text

The *TCP Window Size* and *Frame Length* features contain the average of TCP window size and frame length for nodes. *TCP Window Size* indicates the byte-size sent before an *ACK* is received by a node. The *frame length* refers to the frame on the data link layer. Certain bots may benefit from having access to privileged ports on the compromised device (lower than 1024) while others may not have access to those ports (sending packets from port numbers higher than 1024). The features *Privileged Source Ports* and *Not Privileged Source Ports* summarize the number of packets sent from privileged ports and non privileged ports, respectively. The *quantity of ports* each node tries to reach is summarized because some bots may communicate mainly through a restricted number of ports. The *average Time To Live (TTL)* of each packet is also employed since some bots may want to hide sending packets by setting low or high TTLs.

In addition to features extracted from the network traffic, others are extracted from relations between nodes. A directional graph is built in order to derive for these interactions. For each packet exchanged from a source node to a destination node, IP addresses from source and destination are captured and stored as vertices in the graph. A directed edge is inserted from the source node vertex to the destination node vertex. If an edge already exists between the same source and destination pair, a new one is not created. The source and destination addresses from the packets are the basis to build a graph $G_1 = (V, E)$, where V is the set of

IP addresses in the observed traffic, and E is the set of edges indicated by the ordered pair of source and destination IP addresses of a packet in the traffic. The following features are extracted from this graph to characterize the node behavior: *out degree*, *in degree*, *out degree weight*, *in degree weight*, *local clustering coefficient*, *node betweenness centrality* and *eigenvector centrality*.

In degree and *out degree* indicate the number of nodes that communicate with other nodes. *In degree* is the number of nodes that have sent at least one packet to a given node, while *out degree* is the number of nodes to whom a given node has sent at least one packet. *In degree* and *out degree* are important because if many bots try to reach a given node, this node could be acting as a botmaster, having a high value in the *in degree* and/or *out degree* features. *In degree weight* and *out degree weight* are similar to the previous features but they are pondered by the quantity of packets sent to and from a given node, respectively. For each exchanged packet, existence of an edge is checked on the graph being built with the same source and destination. If there is an edge, the current value is incremented, otherwise an edge is created and the values are initialized.

Node betweenness centrality quantifies the number of times a given node acts as a link to the shortest path between two other nodes. The higher the *node betweenness centrality*, the more central the node is in a graph, which is important to detect bots that may act in a decentralized way, such as in a Peer-to-Peer (P2P) bot network [12]. *Local clustering coefficient* indicates how concentrated is the neighborhood of a given node, trying to identify how close the neighbors are in the graph. *Eigenvector centrality* is a criterion that identifies the influence of a node to other nodes, indicating a normalized weight for a given node.

B. THE PROCESSING MODULE

The Processing Module fetches data from the Feature Extraction Module and calculates indicators that are analyzed on the Analysis Module. Some of its processing may be triggered by the Notification Module, once an analysis indicates a possible DDoS attack starting in the network. Three components compose this module: Feature Selection, Clustering, and Leading Indicators. The Feature Selection Component identifies the most relevant features in network traffic and outputs a set of suitable features to perform bot detection. The Clustering Component groups network nodes with similar behavior, according to an unsupervised machine learning technique. Leading Indicator Component offers information about a change in network state. Calculating the leading indicators is fast and less costly than doing a deep analysis. This may indicate a DDoS, without, however, doing an analysis of each calculation output. The next subsections detail how these components have been implemented in this case study.

1) FEATURE SELECTION

All 19 features extracted from the Feature Extraction Module are normalized, which is important for the clustering. However, these features may or may not be relevant to the whole analysis. It might seem that having more features is always interesting, but in machine learning (employed for clustering) having many features may incur in the Curse of Dimensionality, as described in [24]. This indicates the existence of a bound on the number of relevant features. Finding a relevant subset of features is critical for analysis. This is the goal of the selecting features: having the most relevant and most informative subset of features about the network traffic.

The α*-investing*$^+$ algorithm (Algorithm 1) [10] is used to select the most relevant features. This algorithm has the goal to reduce false-positive rates in the discovery of new features on DDoS traffic. In this case, a false-positive consists in a newly discovered feature wrongly chosen and added to the subset of relevant features until a given time t. Following α*-investing*$^+$, a feature is added to the subset of selected features if there is no dependence between the analyzed feature and any other feature already in the subset. A threshold α ($l.5$ on Algorithm 1) estimates the probability of including an irrelevant feature in the subset. Furthermore, a weight w ($l.2$) represents the percentage of false-positives that eventually could be acceptable in the subset across the algorithm iterations. A false-positive, then, is a feature with some dependency relation with another in the subset of selected features. The p-value is the probability of two features being independent and it is calculated through a Chi-squared test that can work on categorical and non-parametric data, while measuring feature independence.

Algorithm 1 α-investing$^+$ Algorithm

```
Data: Network Data
Result: Selected features set
1  $W \leftarrow [0.5]$; // Weight set
2  $w_0 \leftarrow W_0$; // Initial weight
3  $F \leftarrow \{\}$; // Detected features set
4  $S \leftarrow \{\}$; // Selected features set
5  $\alpha_\Delta \leftarrow 0.5$;
6  i ← 1;
7  while New features detected do
8      addFeature($f_i$, $F$);
9      $\alpha_i \leftarrow w_i/2 * i$;
10     repeat
11         table ← createContingenceTable($f_i$, $m$);
12         result ← chiSquare(table);
13         if result[p-value] $\leq \alpha_i$ then
14             $w_{i+1} \leftarrow w_i - \alpha_i$;
15         else
16             addFeature($f_i$, $S$);
17             $w_{i+1} \leftarrow w_i + \alpha_\Delta - \alpha_i$;
18         end
19     until each $s \in S$ that $s \neq f_i$;
20     i ← i + 1;
21 end
```

In order to add a new feature to the set of relevant features, the result of the statistical test should be higher than the current α threshold (l.13 to l.18) with the feature being analyzed. If the result is higher than the threshold, the current weight w increases (l.17). Otherwise, when there is a dependency between features, w decreases (l.18). The weight w affects directly the threshold α (l.9), the higher the number of selected features, the stricter the threshold becomes to avoid adding false-positives. For each of the 19 features, a vector (named feature vector) with the values attributed to the features is built for each node. The algorithm has as inputs the feature vectors and it is executed until no more features are detected. All the 19 generated features are continuously evaluated. The α-*investing*$^+$ algorithm reduces the data set to be subsequently analyzed, improving the method performance without compromising data representativeness. The algorithm output is a set of features that better represents the analyzed traffic.

2) CLUSTERING

Self-Organizing Map (SOM) technique is employed to cluster nodes. SOM is an unsupervised machine learning technique that reduces data dimensionality, without the need of data classification (labels). Clustering aims at grouping nodes based on the selected features. Each node provides as input to the clustering algorithm a feature vector. The SOM technique compares each feature vector to the ones from other nodes, through the Euclidean distance between vectors. Given two vectors $Q_1 = (q_1, q_2, \ldots, q_{19})$ and $S_1 = (s_1, s_2, \ldots, s_{19})$, the distance is calculated as the square root of the sum of the squared differences of the distances between the vector terms (Eq. 1). The vector terms are the values observed from the network traffic of the selected features per node. A node is included in a cluster if its feature vector is closer to the center of that cluster.

$$d_E(Q, S) = \sqrt{(q_1 - s_1)^2 + \cdots + (q_{19} - s_{19})^2} = \sqrt{\sum_{i=1}^{n}(q_i - s_i)^2} \tag{1}$$

The Euclidean distances designates if a node is associated to a cluster. With the distances, SOM returns a 5×5 grid. Each item of this grid is a cluster, with the node's IP addresses associated to each cluster. A cluster is a collection of similar nodes. At each iteration, cluster's centers are readjusted by recomputing the centroid of the cluster as it grows. Iterations run until the cluster centers do not change (and, consequently, no new nodes are added to the cluster) or a number of iterations is reached (we set the limit to 1000).

Clustering reduces the dimensionality of the problem, granularity now being the number of the clusters instead of the number of nodes. This also reduces significantly the space of the analysis, with a lower number of nodes analyzed each time (per cluster) compared to the complete dataset analysis. Hence, the high amount of input data is now rearranged into a finite number of clusters. Clusters are distributed in a grid that enables the understanding of the node relations, mainly when there is a high volume of data, as is often the case.

It is expected that some clusters have a much higher number of nodes than others. Clusters with a higher number of nodes have a benign behavior in the network [12]. Malicious/abnormal behaviors are less often in the real world. Also, node infection is not immediate and malicious behavior is not immediate on nodes, happening over time [25]. For example, for Wannacry ransomware, that quickly infected many companies and hosts around the world, about 30% of hosts that had TCP port 445 open to the Internet (the way this malware infected the devices) were vulnerable [26]. This assumption allows the detection to start from the smaller cluster, speeding up bot detection, with a much smaller computational cost than carrying out the analysis on the whole network traffic at once. Detection starts on the smallest cluster (i.e., the one with the smallest number of nodes), followed by the second one, until checking all clusters.

3) LEADING INDICATORS

This case study employs four leading indicators to predict a state change on a computer network: return rate, autocorrelation, coefficient of variance, and skewness. We associate the metastability phenomenon with a computer network under attack. In a network that is in a metastable state, a DDoS attack produces a disturbance in the network state, for instance. Disturbances can push the state towards values that are close to conquering the barrier between two metastates. The Analysis Module, particularly the Prediction Component, verifies the metastates, taking as input the four leading indicators. Small disturbances may cause changes in the network state that may not be enough to make it to overcome a barrier. If the disturbance is big enough the network state potentially experiences a critical transition. Network states are analyzed based on the network packet sizes in a time window (the Data Collection Module). Hence, a network state is characterized by the behavior of a set of network packet sizes during a time window (Feature Extraction Module).

The first leading indicator is return rate. It measures the rate of return or recovery to an equilibrium meta-state. It is calculated by the dominant eigenvalues from the matrix composed by the observations of a time window. Dominant eigenvalues characterize the rate of change around a metastable state. It can indicate the proximity of a critical transition. Particularly, it can indicate the irreversibility of a transition, i.e., if the network state is drifting from a metastable state to another, breaking the critical barrier that separates them. The higher the return rate, the faster the network recovers from small disturbances around its current state and, consequently, the network has a higher resilience to change. The return rate is reduced when the network approaches a critical transition, decreasing smoothly to zero as the disruptive transition approaches [21]. The smooth reduction of the return rate close to a critical transition is called Critical Slowing Down (CSD), a term coined from dynamical systems theory.

The second leading indicator is autocorrelation. It is a metric employed to evaluate correlation between observations. It measures how much the network states have become increasingly similar between consecutive observations. In [21] the authors observed that autocorrelation in time series increases when critical transitions approach. Disruptive transitions tend to increase the correlation at low lags (also known as 'short-term memory') between observations on a time series. Autocorrelation is calculated at lag-1, i.e., the correlation of the time series to itself shifted one time-step back. In Equation 2, the variables z_t and z_{t+1} represent two consecutive observations at times t and $t+1$, respectively, μ is the mean in a given time window, and σ is the variance of the variable z_t.

$$\rho_1 = \frac{E[(z_t - \mu)(z_{t+1} - \mu)]}{\sigma_z^2} \tag{2}$$

The third leading indicator is the coefficient of variation. It is a statistical measure that indicates the level of variance in a time series. It is calculated as $CV = \frac{SD}{\mu}$ where SD is the empirical standard deviation calculated by $SD = \sqrt{\frac{1}{n-1}\sum_{t=1}^{n}(z_t - \mu)^2}$. Critical transitions and CSD phenomenon increase the variance in a time series [21]. With other leading indicators, the coefficient of variance can assist in predicting trends of a critical transition, in this case a volumetric DDoS attack.

The fourth leading indicator is skewness, a well-known statistical measure that indicates asymmetry observations distribution. If the skewness value is increasing, the observations' distribution are becoming asymmetric. Skewness is calculated as shown in Equation 3. Like variance, skewness can also increase because of a critical transition, meaning that the asymmetry in the observations increases. This happens because the dynamics close to the barrier become slow. [21] observed a rise in the skewness in different types of time series when they are close to a critical transition. Skewness is employed as a leading indicator and it is analyzed together with the other three.

$$\gamma = \frac{\frac{1}{n}\sum_{t=1}^{n}(z_t - \mu)^3}{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(z_t - \mu)^2}} \tag{3}$$

These statistical calculations are employed as leading indicators on attacks' trends. The individual analysis of a single leading indicator should not be taken for conclusions, in order to reduce false positives or false negatives. The analysis should be performed on all indicators together. In [21], the authors demonstrated that this set of leading indicators defines a specific behavior at the imminence of a critical transition. They observed that at the imminence of a disruptive change (*i*) the return rate decreases, (*ii*) the autocorrelation at-lag-1 increases, (*iii*) the coefficient of variance increases, and (*iv*) the skewness increases. This set of leading indicators early predicts the beginning of a volumetric attack.

C. THE ANALYSIS MODULE

This subsection presents a description of the Analysis Module in this case study. The next subsections focus on describing the two main components of this module.

1) THE PREDICTION COMPONENT

The Prediction Component identifies a change on the network state, through the identification of metastates. A system is metastable when it tends to spend a long time in a particular intermediate state A, as known as metastable state (or metastate), and then eventually it drifts swiftly to another state B, where it remains now for a long time (or indefinitely). /brThe system's dynamics can be split into three distinct time-scales: (*i*) long-time at metastate A, (*ii*) rapid transition to a different state B, (*iii*) staying on state B for a long (possibly an infinite amount of) time. Characterizing metastability requires determining the metastates, how long the system stays on each metastate, and how fast is the transition. The literature on the subject is mostly of empirical studies, as a formal analysis is impractical. Observing empirically macroscopic metastability is far more achievable than analytically studying the microscopic interactions among agents [27].

Metastability often arises in systems that exhibit two or more possible stable states. For instance, consider the framework of diffusion of two (for simplicity, but in practice usually more) opinions A and B in a social network of individuals, where every individual exerts some influence upon their peers' opinions and is influenced by theirs as well. There are at least two stable states: everyone adopts opinion A or everyone adopts opinion B. The system may swing swiftly between the two states depending on exogenous perturbations. A stochastic system undergoes perturbations about a stable state, and if the deviations are large enough, it overcomes a barrier and drift quickly to another stable state. A relevant measure of the time it takes for a system to move from one to another stable state is the return rate.

As a metastable system undergoes a fast transition – possibly, to an undesirable state (e.g., blackout in a power grid, sudden compromising of IoT services, cardiac arrest) – it is reasonable to expect early-warning signal as precursors of the transition, which in turn can help to control it. These precursors may be used to flag when the probability of a transition becomes unusually high. It is an important and subtle aspect of the problem to characterize such macroscopic observations. Depending on the application, several heuristic statistics are used in the literature to forecast a transition beyond the four leading indicators referred to before [28]. As a general rule, in particle systems [27] (the framework that formally studies metastability), there are no broad universal approaches and predictive statistics are usually crafted targeting each scenario. One exception is the seemingly universal usefulness of the return rate. It measures the irreversibility of a transition, i.e., if the system is drifting from a metastable state to another.

2) THE DETECTION COMPONENT

Inside each cluster, correlations between nodes are calculated to identify interrelations and their evolution across time. For each feature and node, a time series is built, *i.e.*, a collection of values distributed in regular time intervals, ranging from the start of the evaluation to its end. The collected values are regularly spaced in time, so that the temporal sample size is the same for all the evaluated nodes and all the series have the same amount of data in the same analysis. Out of the 19 extracted features, each one has its own time series. Only the feature *out degree weight* is employed to assess the correlation. As output, an influence matrix is generated, making it possible to identify causal relations in the traffic generated by the nodes and enabling bot identification in traffic. Each input in the matrix matches the weight attributed to the relation between two nodes (row and column).

With the time series defined, autoregression is applied to estimate the influence matrix. The autoregression coefficients are graph filters that allow an even smaller data search space [29]. The influence degree between the identified nodes is calculated without prior knowledge to data relation. However, the value of a feature in a given time k is influenced by some feature in the k-1 time window. This influence lies in the identification provided by the autoregressive model and is exposed in the incidence matrix. Eq. 4 describes the mathematical details to calculate the influence matrix [29].

$$\begin{aligned} x[k] &= w[k] + \sum_{i=1}^{M} P_i(A, c)x[k-1] \\ &= w[k] + \sum_{i=1}^{M}\sum_{j=0}^{i}(c_{ij}A^j)x[k-1] \\ &= w[k] + (c_{10}I + c_{11}A)x[k-1] \\ &\quad + (c_{20}I + c_{21}A + c_{22}A^2)x[k-2] + \ldots \\ &\quad + (c_{M0}I + \ldots + c_{MM}A^M)x[k-M] \end{aligned} \tag{4}$$

where $x[k]$ is the feature value at time k, $P_i(\mathbf{A}, \mathbf{c})$ is a polynomial on the matrix A with order i (*i.e.*, $P_i(\mathbf{A}, \mathbf{c})$ are graph filters [29]); $w[k]$ is statistical noise to evaluate the precision of the autoregression; c_{ij} are polynomial scalar coefficients, with $\mathbf{c} = (c_{10}, c_{11}, \ldots c_{ij}, \ldots, c_{MM})$ a vector of all the c_{ij}, and M is the autoregression order [16]. All are known, except the influence matrix $\mathbf{A}$.

Given that there is a correlation between the data, a graph $G_1 = (V, \mathbf{A})$ defines this correlation, with V being the vertices that match the network nodes and an unknown $\mathbf{A}$, that defines this correlation. To calculate $\mathbf{A}$, it is done:

1) Solve for $R_i = P_i(\mathbf{A}, c)$: Apply graph filters, having as input the temporal series
2) Retrieve the structure of $\mathbf{A}$ with one of these two approaches: using $\hat{A} = \hat{R}$ as in

$$\begin{aligned} \hat{\mathbf{R}}_i = \underset{\hat{\mathbf{R}}}{\operatorname{argmin}} \frac{1}{2}\sum_{k=M}^{k-1} \| x[k] - \sum_{i=1}^{M} R_i x[k-j] \|_2^2 \\ + \lambda_1 \| vec(\hat{R}_1) \|_1 + \lambda_3 \sum_{j\neq i} \| [R_i, R_j] \|_F^2 \end{aligned} \tag{5}$$

or compounding all the $\hat{\mathbf{R}}_i$ to find $\mathbf{A}$,

$$\begin{aligned} \hat{\mathbf{A}} = \underset{\mathbf{A}}{\operatorname{argmin}} \| \hat{\mathbf{R}}_1 - \mathbf{A} \|_2^2 + \lambda_1 \| vec(\mathbf{A}) \|_1 \\ + \lambda_3 \sum_{i=2}^{M} \| (\mathbf{A}, \hat{\mathbf{R}}_i) \|_F^2 \end{aligned} \tag{6}$$

3) Estimate all c_{ij} with one of these two approaches: estimating $\mathbf{c}$ or starting from $\hat{\mathbf{A}}$ and $\hat{\mathbf{R}}_i$ as in

$$\hat{\mathbf{c}}_i = \underset{\mathbf{c_i}}{\operatorname{argmin}} \frac{1}{2} \| vec(\hat{\mathbf{R}}_i) - \mathbf{Q_i c_i} \|_2^2 + \lambda_2 \| \mathbf{c_i} \|_i \tag{7}$$

where

$$\mathbf{Q}_i = (vec(\mathbf{I})vec(\hat{\mathbf{A}}) \ldots vec(\hat{\mathbf{A}}^i)), \quad c_i = (c_{i1}c_{i2} \ldots c_{ii}) \tag{8}$$

or from $\hat{\mathbf{A}}$ and the data $\mathbf{X}$ as in

$$\hat{\mathbf{c}}_i = \underset{\mathbf{c}}{\operatorname{argmin}} \frac{1}{2} \| \mathbf{Y}(\hat{\mathbf{A}}) - \mathbf{B}(\hat{\mathbf{A}})\mathbf{c} \|_F^2 + \lambda_2 \| \mathbf{c} \|_i \tag{9}$$

At the end of this process, the correlation degrees between the nodes are identified in the adjacency matrix. The higher the coordination between nodes, the higher the correlation between them. This results in a high probability to identify a botnet [30]. The coordination between nodes lies in the magnitude of the values on the adjacency matrix. The farthest from zero, the higher is the influence between nodes identified in the rows and columns of the matrix.

As stated before, bots are detected per cluster. Thus, the volume of analyzed data is smaller than in other applications of this autoregression technique [31]. For each cluster, the relations for each node can be calculated, having as output an $N \times N$ matrix, where N is the number of nodes in the cluster. When the influence matrix is calculated, bots have a much higher magnitude order for calculated values than the nodes that do not have any correlation among them. Cluster analysis can benefit from parallel computing, improving performance detection. Each matrix (i.e., each cluster) is individually evaluated, identifying its bots. Nodes are classified in bots or not. Hence, results can be reported by cluster for immediate action after its analysis.

D. THE NOTIFICATION MODULE

The Notification Module interfaces components that may be distributed on the network and also reports results from the analysis to other systems, in a human-readable format or to further action on the network. Once a prediction analysis is completed, if it identifies a state change on the network, a deeper analysis to identify the bots is started. The Analysis Module indicates to the Notification Module to start the processing of data in order to detect the bots. The Notification Module triggers the Processing Module.

Once a detection is completed, an action on the network is recommended to block action of malicious actors. The Notification Module can indicate to system administrators and/or network managers which nodes may be compromised,

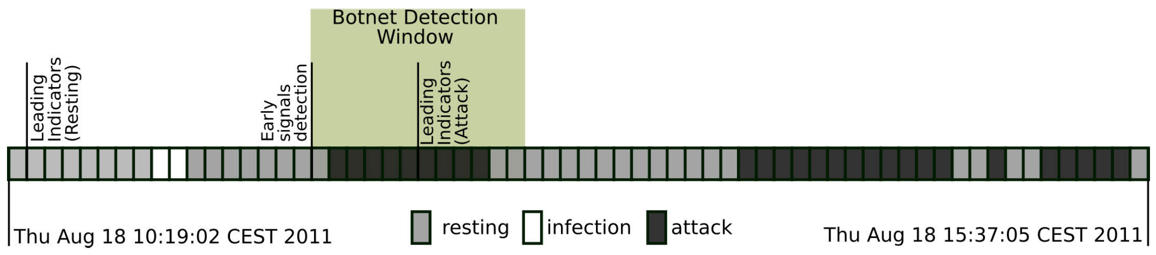
Botnet Detection Window
Leading Indicators (Resting)
Early signals detection
Leading Indicators (Attack)
Thu Aug 18 10:19:02 CEST 2011
resting
infection
attack
Thu Aug 18 15:37:05 CEST 2011

FIGURE 3. CTU-13 - Scenario 10 attack time frame.

with the behavior of a bot, so they can act on the network. The Notification Module can also provide other system data so they can act on the network, blocking its traffic flow, reducing possible damage of a DDoS attack.

V. PERFORMANCE EVALUATION

This section presents the results for the early signals prediction, that calculates the leading indicators, and bot detection, that detects bots only when triggered by the early signals. The goal is to detect bots, with high accuracy, reducing false-positives and false-negatives. Results from the evaluation of the CTU-13 dataset, scenario 10, are discussed and explained on Subsection V-A. This dataset was created from a virtual environment deployed by the Czech Republic University, with 13 different scenarios. Results from Scenarios 04 and 11 from this dataset, which also contains DDoS attacks, are presented on Subsection V-B.

CTU-13 dataset, on scenario 10, presents traffic with a volumetric DDoS attack, with 10 infected hosts. The dataset contains logs of the network traffic following the setting: a Rbot malware infects the devices and those infected devices comprise a DDoS attack. The Rbot malware exploits security vulnerabilities in Microsoft operating systems. The whole scenario lasted 4.75 hours with more than 90 million packets exchanged in the network. The attack happens in multiple time periods during the capture, allowing us to perform the methods under a resting scenario, without the communication of the infected hosts, before the attack starts and during it.

A window size of 60 seconds was used (from 0 to 59.99 seconds). From the second window, the Prediction Component keeps as history 50% of the previous window. While the Prediction Component analyzes the next window leading indicators, the botnet detection accumulates the traffic from prior windows while it is detecting the influence of one node over the other, indicating the presence of bots. Figure 3 shows the timeframe of the capture for scenario 10 of CTU-13 and highlights the capture period employed to plot the results shown in the subsequent figures. The system implemented for this case study runs continuously, calculating the leading indicators for each window and executing the botnet detection method when triggered by the early signals of DDoS attacks.

To calculate the leading indicators and analyze the network behavior, time series are built with packet size and its timestamp. The goal lies in employing relevant but simple features to predict the imminence of an attack. Features such as packet generation frequency and packet size can strongly characterize a volumetric DDoS attack. High frequency packet generation is generally associated with DDoS attack. The increase in packet size usually indicates that the attacker is trying to cause network saturation. Packet sizes usually are smaller than 250 bytes. However, one observes packet sizes reaching 1500 bytes, mostly when the network is under attack.

It is possible to observe a fast increase in the packet size on the built time series (Figure 4, right). While in the first time series (Figure 4, left) the size of the majority of the packets are 60 bytes, in the second time series (Figure 4, right) it is possible to see the transition (indicated at the red dashed vertical line) from a period in which packets are of 60 bytes to a period in which the majority of the packets are 1500 bytes. Another interesting aspect lies in the two peaks (highlighted by the red boxes) presented in the first time series (Figure 4, left). Despite the fact that this time series is from a non-attack phase, few packets of 1500 bytes can be seen. This behavior was observed in other analyzed time series. We suspect that they correspond to bots preparing and testing themselves, a few minutes before launching the attack.

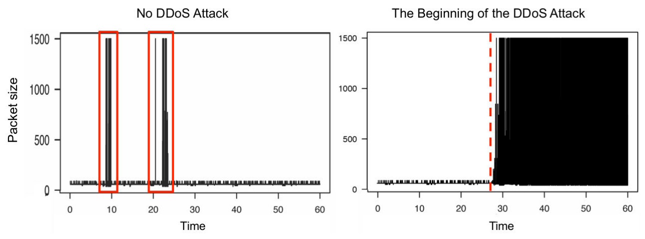

FIGURE 4. DDoS attack.

A. RESULTS

The calculation of leading indicators is done for each window of 60 seconds that was split from the *pcap* file that contains all traffic. Three behaviors can be observed: in a resting scenario, moments before an attack starts (where there is already bot communication) and during an attack. Moments before the attack, there is a state change on the network, calculated by the leading indicators, providing us a signal that something malicious might be happening on the network. With this, botnet detection starts to log the traffic to be able to identify the bots.

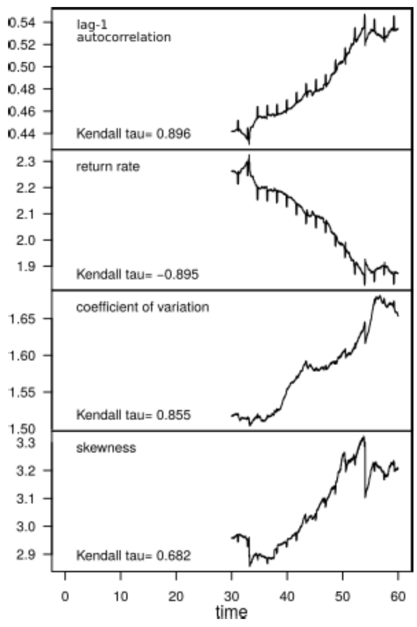

FIGURE 5. Leading indicators in the imminence of an attack.

Figure 5 shows the results for the leading indicators, before the attack starts (from Thu Aug 18 11:46:13 CEST 2011 to Thu Aug 18 11:47:13 CEST 2011). The indicators show an imminent beginning of a DDoS attack, i.e., there is a decrease of the return rate curve, while there is an increase of the autocorrelation curve, the coefficient of variation and skewness. The decrease of the return rate indicates the presence of significant oscillations on the network traffic flow, then, not being able to keep in a metastable state. The trend of high increase for autocorrelation index (positive Kendall tau of 0.896) indicates a similarity on the packet sizes though time, meaning that there is a trend to the packets stay close to 1500 bytes. This packet size is used during an attack to overload the bandwidth by the attack as it is the maximum size of a packet. The increase in the coefficient of variation reveals a strong instability on the network due the presence of extreme values and the high variation on the packet sizes. Finally, the positive skewness, with its increasing trend, points the concentration of the packet size close to 1500 bytes. Considering the indicators, a critical state change happens, pointing out an imminent DDoS attack.

To illustrate the behavior when no critical state change is verified, Figure 6, left, shows the state on the network on a resting stage. At this point, the traffic capture had begun, but without the presence of the bots that perform DDoS attacks. Similarly, Figure 6, right, shows the leading indicators in the middle of the DDoS attack, where also there is no critical state change (the attack tends to continue). The return rate indicates continuity of the metastable state due to low variance of the data. Both autocorrelation and the coefficient of variation indicate low data similarity, with low variance, pointing to a change on the packet average size.

The state change, shown in Figure 5 triggers the botnet detection. Figure 3 shows the window that data was accumulated and continuously scanned for bots. Next, the results show the botnet detection method performing the feature extraction and selection, the clustering and, finally, indicating bots that behave like bots or not. During the analysis, traffic from 35477 nodes were seen on the traffic.

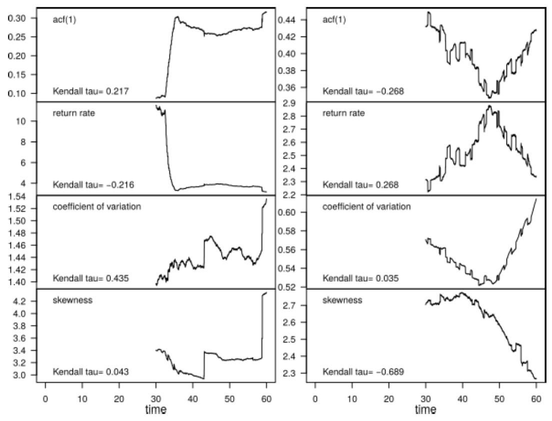

FIGURE 6. Leading indicators while not in the imminence of an attack.

For all the nodes, 19 features were continuously extracted and analyzed. Table 2 shows the result of the feature selection, having as output selected and not selected features. It is observed a reduction of 48% in the features total number. The feature *node betweenness centrality*, not selected, is the one that has the highest computational cost to calculate. However, it is unknown, in advance, if this feature should or should not be selected. In some tests, this feature took 99.9% of the time spent to perform the calculation of all the features. In all cases, it took over 95% of time, as in [11].

TABLE 2. Selected and not selected features during attack.

Not selected	Selected
TCP Windows Size	Average TTL
Percent DNS	Destination Port Quantity
Percent TCP	Frame Length
Source Privileged Ports	Percent ICMP
Local clustering coefficient	Percent UDP
Out Degree Weight	Percent Outros
Out Degree	Source Not Privileged Ports
Node Betweenness Centrality	In Degree Weight
Protocol Quantity	In Degree
	Eigenvector Centrality

The Clustering Component creates 25 clusters using the selected features. All the 10 bots of this dataset were clustered on the same cluster with another node. Table 3 highlights the cluster with the bots. Starting the analysis from the smaller cluster and the detection might be able to detect the bots after analyzing 27 other nodes (on 7 clusters). This speeds up the detection and reduces the time to an action to be taken and mitigate the attack.

Bot detection starts with the smallest cluster. Based on the matrix of influence, it is evaluated as the influence between nodes on each cluster, using as feature the total number of

TABLE 3. CTU-13 Scenario 10: Clusters during the attack.

	0	1	2	3	4
0	21	23	9	0	3
1	100	27	3	11	3
2	2963	48	20	0	3
3	7570	132	23	13	3
4	24113	339	35	12	3

packets exchanged between pairs of nodes. When there is no correlation between nodes, their corresponding values on the matrix of influence are close to zero, in absolute values. Figure 7 illustrates the matrix of influence and its higher values for a given node.

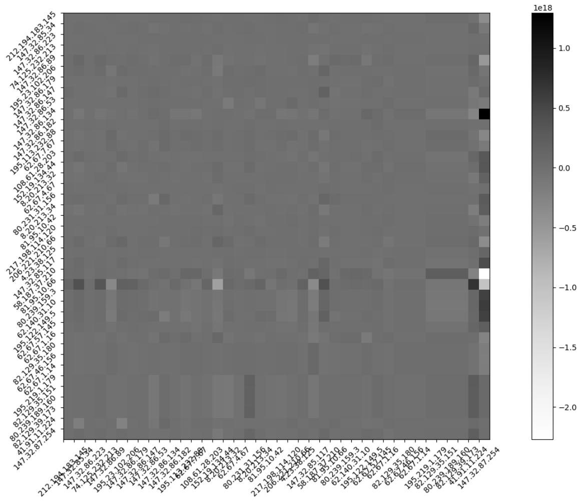

FIGURE 7. Influence matrix on a cluster. Last column node has higher influence on nodes than other nodes on cluster.

Results also show influence among nodes in other clusters. When this influence is observed, the node is identified as bot. On the cluster with the bots (10 bots on 11 nodes), the detection correctly classified 9 out of the 10 bots, also correctly identifying the benign node on that cluster. Therefore, Table 4 shows the confusion matrix after the detection is done. Nine nodes were correctly classified as bots (true-positives), 35464 nodes were correctly classified as benign (true-negatives), seven benign nodes were falsely classified as bots (false-positives) and one bot were falsely classified as benign (false-negative).

TABLE 4. Confusion matrix after bot detection.

		Real	
		Bot	Benign
Prediction	Bot	9	7
	Benign	1	35460

Also, if the Detection Component needs to check all the nodes on the dataset in the search for bots, the execution time would turn the analysis impractical. This cluster-based approach and doing the detection only when triggered by early signals reduces the amount of analyzed data, without compromising the results.

B. OTHER RESULTS

This section presents results from scenarios 04 and 11 of the CTU-13 (CTU-13/S04 and CTU-13/S11, respectively) and CAIDA datasets. These datasets also contain DDoS attacks. The scenario 04 of CTU-13 has 4.21 hours of duration, from August 15, 2011 (10:59:23 CEST to 15:11:46 CEST). This dataset has generated over 62 million packets and has Rbot as its malicious bot. The scenario 11 of CTU-13 database is just 15 minutes long, generating over 6 million packets. The CAIDA dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007 (20:50:08 UTC to 21:56:16 UTC). This denial-of-service attack has attempted to block access to the targeted server by consuming computing resources on the server and consuming all of the bandwidth of the network connecting the server to the Internet.

On the CAIDA dataset and also for CTU-13, scenarios 04 and 11, it was possible to verify a critical state change on the network before any attack began. Figure 8 presents the leading indicators for these datasets. A critical transition is detected before the attack starts, that is, when there is a decrease in return rate, and an increase in autocorrelation, coefficient of variation, and skewness.

Table 5 shows the results of the feature selection by the Processing Module, for each dataset. It can be seen that the set of selected features is not static but rather changing for each traffic. It is worth noticing that the feature *node betweenness centrality* was not selected in any dataset. This feature has the highest computational cost to calculate among all. However, it is unknown in advance if this feature should or should not be calculated. The clustering processing outputs a 5 × 5 grid with the nodes associated with each cluster.

TABLE 5. Selected features for each dataset.

Features	CTU-13/S04	CTU-13/S11	CAIDA
Protocol Quantity			
Average TTL	✓	✓	✓
TCP Window Size			
Percent TCP			✓
Percent UDP	✓	✓	
Percent DNS	✓	✓	
Percent ICMP	✓	✓	✓
Percent Others			✓
Source Privileged Ports	✓	✓	
Source Not Privileged Ports			✓
Destination Port Quantity	✓	✓	✓
Frame Length	✓	✓	
In Degree			✓
Out Degree	✓	✓	
In Degree Weight			✓
Out Degree Weight	✓	✓	
Node betweenness centrality			
Local clustering coefficient		✓	
Eigenvector centrality	✓	✓	

For CTU-13/S04, there is only one bot. The bot is in a cluster with another 71 nodes, as can be seen in Table 6.

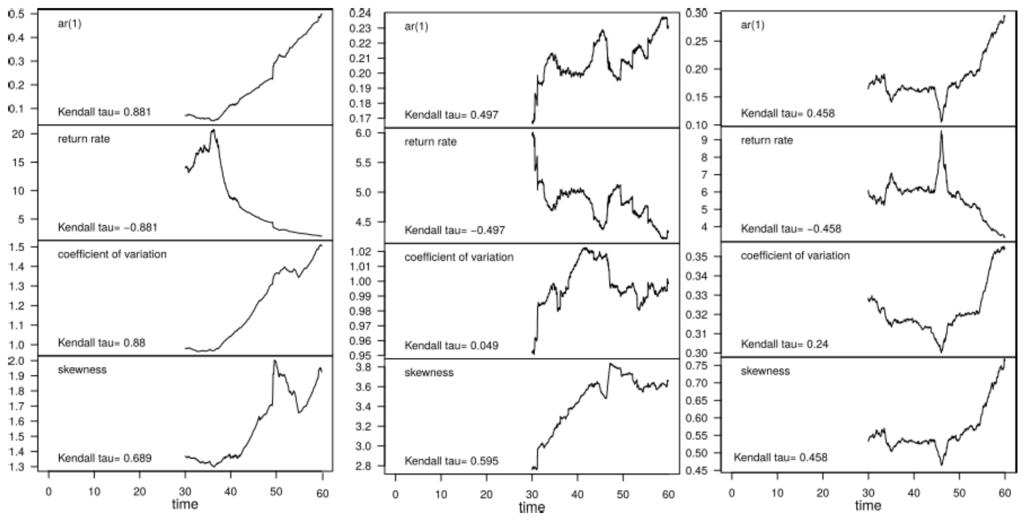

FIGURE 8. Leading indicators on the CTU-13 datasets. Critical transitions detected before the attack starts on Scenarios 04 (left), 11 (center) and CAIDA (right).

TABLE 6. CTU-13 Scenario 04: Clusters - bot in cluster (0,4).

	0	1	2	3	4
0	168268	3172	31	17	72
1	919	367	25	11	2
2	243	115	27	8	4
3	34	44	18	10	8
4	61	41	20	40	20

TABLE 7. CTU-13 Scenario 11: Clusters - 2 bots at cluster (0,1) and 1 bot at cluster (3,4).

	0	1	2	3	4
0	3	7	3	13	23
1	4	0	10	6	2
2	12	14	5	25	51
3	22	56	49	170	457
4	2735	3396	7183	4062	21547

For CTU-13/S11, (Table 7), there are three bots: two can be seen in a cluster with another five nodes and the other bot is found with 456 other hosts. Having the nodes clustered speeds up the detection, as the bot can be detected before reaching the clusters with a higher number of nodes, which takes more computational time and power. The number of bots in the CAIDA dataset is unknown. However, instead of analyzing all the 9251 nodes in one run, clustering allows a quick detection and action in the network, reducing damages caused by the action of the malicious bots. Table 8 shows the clustering result for the CAIDA dataset.

TABLE 8. CAIDA: Clusters.

	0	1	2	3	4
0	114	58	1	0	1
1	136	44	81	0	0
2	78	80	84	6	0
3	130	95	117	49	3762
4	1794	854	169	857	741

The detection component was able to successfully find bots on all datasets. On CTU-13, scenario 04, the detection component identified a significant influence between all the nodes on the cluster that also had the bot. That explains the high number of false-positives in this scenario. Also, this scenario is the one that had the most analyzed nodes, with a total of 186,206. Table 9 summarizes the results.

TABLE 9. Confusion matrix after bot detection on Scenario 04.

		Real	
		Bot	Benign
Prediction	Bot	1	104
	Benign	0	186101

For CTU-13, scenario 11, the detection component marks 6 out of 7 nodes as bots on the cluster with the two real bots, being 4 incorrectly. Also, it did not recognize the bot being the other 456 nodes, on the cluster with the other real bot. Also, on smaller clusters, a significant influence between nodes was identified. 17 nodes were incorrectly classified as bots, while one bot was incorrectly classified as benign. The total number of analyzed nodes is 41,892.

On the CAIDA dataset, traffic from 9251 nodes was present. This traffic is only from the network that the victim is part of, sending or receiving packets. As this dataset is not labeled, the number of bots is unknown. The detection mechanism detected high coordination between 2738 nodes, marking them as bots.

C. COMPARING TECHNIQUES

We use Botnet 2014 [32] dataset to compare our proposed architecture and machine learning techniques, in particular Decision Tree. This database has been chosen because its dataset is split into training and test. This dataset was built merging three others datasets (ISOT dataset [33], ISCX 2012 IDS dataset [34] and Botnet traffic generated by the

TABLE 10. Confusion matrix after bot detection on Scenario 11.

		Real	
		Bot	Benign
Prediction	Bot	2	17
	Benign	1	41872

Malware Capture Facility Project [35]), aiming to improve generality, realism and representativeness. If we had to manually generate a training and test dataset from the other tested datasets (CTU-13 and CAIDA), we might interfere in the results, masquerading the real analysis. Thus, having a dataset built with the separation in training and testing is better.

Decision Tree was chosen to compare our results because literature has shown it provides better results than other techniques [11]. Also, Decision Tree provides a natural feature selection, performing feature selection while it does performance evaluation (in a method known as hybrid/embedded [23]). Thus, as our distributed architecture also performs selection (in a method known as filter), it is fair to use Decision Tree to compare.

However, it must be pointed out that, to analyze the Decision Tree results, no clustering was done. Also, no prediction on local networks was performed. The training dataset was used to train the machine learning algorithm and, then, results gathered on the testing dataset. As our distributed architecture requires no training phase, results are from applying the deterministic technique only in the testing dataset. Results for the Decision Tree algorithm can be found at Table 11.

TABLE 11. Confusion matrix with decision tree to botnet 2014 dataset.

		Real	
		Bot	Benign
Prediction	Bot	7	3
	Benign	24	14173

The prediction component is highly time dependent. Even if windows are not set by time (windows split every 60 seconds), but by packet count (windows split every 10000 packets), the leading indicators are calculated based on timestamps. This is due the fact that a critical transition is evaluated at a given time, in an interval. The way the dataset was built, the packets were sparse distributed in time. Splitting the dataset every 60 seconds (or even 3600 seconds!), only one (or a few) packet was inserted in that window. If packet count was used, as the packets are sparse distributed, leading indicators still could not be calculated for every window. By packet count windows, some windows were calculated and we were able to detect critical transitions. However, we have decided to ignore the prediction component in the following results, as we could not reliably rely on packet timestamps.

Considering that there was a trigger from the prediction component, features would be selected. For this dataset, the following features were selected: Average TTL, Destination Port Quantity, Frame Length, Percent ICMP, Percent DNS, Percents TCP, Percent Others, Source Not Privileged Ports, In Degree and In Degree Weight. The feature selection reduced the data to be analyzed in 48%. These features are the most statistically relevant features for this dataset. The Clustering Component groups the nodes. Table 12 presents the 5 × 5 clusters For the Botnet 2014 dataset.

TABLE 12. Botnet 2014: Clusters.

	0	1	2	3	4
0	19	34	27	2	14
1	99	72	60	28	18
2	369	86	66	41	142
3	299	36	32	1536	8900
4	684	114	754	364	411

Similarly to the other datasets, the detection component classifies the nodes in bots or not. The final classification, after the clustering phase can be seen at Table 13. Comparing results from the Decision Tree algorithm, there was an improvement in the detection of real bots, with more benign nodes classified as bots.

TABLE 13. Confusion matrix with prediction and detection.

		Real	
		Bot	Benign
Prediction	Bot	12	43
	Benign	19	14133

VI. CONCLUSION

This article presented a two-tier distributed architecture for DDoS attack prediction and bot detection. The architecture promotes a distributed and integrated analysis of the local network traffic and the Internet traffic. The architecture defines the use of prediction methods, such as the early signals of volumetric DDoS attacks to trigger bot detection methods. By a case study, we evaluate the proposed architecture. In the case study, we employ early signals such as leading indicators, calculated from the network traffic. These early signals have been chosen because domestic routes can easily calculate them. Once an early signal predicts a possible imminence of a DDoS attack, the system triggers the bot detection to identify malicious nodes in the network.

Hence, this article showed that it is possible to identify changes in the network state that might indicate a DDoS attack. Following this two-tier, the volume of analyzed data particularly to bot detection is reduced and occurs per cluster of nodes, without losing detection efficiency. Performance evaluation employed the CTU-13 dataset, scenario 10 and the results showed the efficiency in detecting malicious nodes, infected by bots, controlled to launch DDoS attacks.

A. FUTURE WORK

The proposed DDoS prediction and bot detection architecture follows a local network and Internet traffic analysis. Early signals of a volumetric DDoS attack trigger a hybrid botnet detection method. For the early detecting signals, we have employed four leading indicators that are commonly used in biological systems to indicate a transition. However, other indicators could be explored to be added.

Further features could be extracted, both from traffic and graphs, to better characterize the traffic. Window sizes were empirically chosen, in a trade-off between performance and accuracy, without having a final conclusion on the best window size. Studying the best window size parameter for each traffic will surely improve results.

Other feature selection mechanisms could be investigated. We have employed the α-*investing*$^{+}$ algorithm, that operates as a filter method. Wrapper and embedded methods could be analyzed. We have compared our proposal with a Decision Tree algorithm. Other machine learning techniques could also be explored, to better emphasize the results obtained in our architecture. Finally, the vast literature on botnet detection focuses on machine learning techniques instead of deterministic methods. A hybrid method could be explored using both, combined, to improve the accuracy of the detection methods.

ACKNOWLEDGMENTS

The authors thank the agencies CAPES, CNPq, and the joint NSF and RNP HealthSense project, grant #99/2017.

REFERENCES

[1] D. J. Atkin, L. W. Jeffres, and K. A. Neuendorf, "Understanding Internet adoption as telecommunications behavior," in *J. Broadcast. Electron. Media*, vol. 42, no. 4, pp. 475–490, Sep. 1998.

[2] E. AbuKhousa, N. Mohamed, and J. Al-Jaroodi, "E-health cloud: Opportunities and challenges," *Future Internet*, vol. 4, no. 3, pp. 621–645, Jul. 2012.

[3] M. Humphries. (2020). *Amazon Mitigates Biggest Ever DDoS Attack*. Accessed: Jul. 2, 2020. [Online]. Available: https://www.pcmag.com/news/amazon-mitigates-biggest-ever-ddos-attack

[4] A. A. Santos, M. Nogueira, and J. M. F. Moura, "A stochastic adaptive model to explore mobile botnet dynamics," *IEEE Commun. Lett.*, vol. 21, no. 4, pp. 753–756, Apr. 2017.

[5] M. Pelloso, A. Vergutz, A. Santos, and M. Nogueira, "A self-adaptable system for DDoS attack prediction based on the metastability theory," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[6] A. Ahmadian Ramaki and R. Ebrahimi Atani, "A survey of IT early warning systems: architectures, challenges, and solutions," *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4751–4776, Oct. 2016.

[7] C.-L. Tsai, A. Y. Chang, and M.-S. Huang, "Early warning system for DDoS attacking based on multilayer deployment of time delay neural network," in *Proc. 6th Int. Conf. Intell. Inf. Hiding Multimedia Signal Process.*, Oct. 2010, pp. 704–707.

[8] B. Xiao, W. Chen, and Y. He, "A novel approach to detecting DDoS attacks at an early stage," *J. Supercomput.*, vol. 36, no. 3, pp. 235–248, Jun. 2006.

[9] M. Korczynski, A. Hamieh, J. H. Huh, H. Holm, S. R. Rajagopalan, and N. H. Fefferman, "Hive oversight for network intrusion early warning using DIAMoND: A bee-inspired method for fully distributed cyber defense," *IEEE Commun. Mag.*, vol. 54, no. 6, pp. 60–67, Jun. 2016.

[10] A. A. Moussa, M. Nogueira, and A. L. Guedes, "[Translated from Portuguese] Online feature selection in streaming based on alpha-investing to DDoS attacks," in *Proc. Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, 2019, pp. 43–56.

[11] A. A. Daya, M. A. Salahuddin, N. Limam, and R. Boutaba, "A graph-based machine learning approach for bot detection," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 144–152.

[12] S. Chowdhury, M. Khanzadeh, R. Akula, F. Zhang, S. Zhang, H. Medal, M. Marufuzzaman, and L. Bian, "Botnet detection using graph-based feature clustering," *J. Big Data*, vol. 4, no. 1, p. 14, Dec. 2017.

[13] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Botnet fingerprinting: A frequency distributions scheme for lightweight bot detection," *IEEE Trans. Netw. Service Manage.*, early access, May 21, 2020, doi: 10.1109/TNSM.2020.2996502.

[14] K. B. V., N. D. G., and P. S. Hiremath, "Detection of DDoS attacks in software defined networks," in *Proc. 3rd Int. Conf. Comput. Syst. Inf. Technol. for Sustain. Solutions (CSITSS)*, Dec. 2018, pp. 265–270.

[15] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. G. Narayan, "Distributed denial of service (DDoS) attacks detection system for OpenStack-based private cloud," *Procedia Comput. Sci.*, vol. 167, pp. 2297–2307, Jan. 2020.

[16] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, Apr. 2017.

[17] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Proc. Int. Conf. Wireless Commun., Netw. Mobile Comput.*, vol. 2, Sep. 2005, pp. 1041–1044.

[18] R. Kate, R. Perez, D. Mazumdar, K. Pasupathy, and V. Nilakantan, "Prediction and detection models for acute kidney injury in hospitalized older adults," *BMC Med. Inform. Decis. Making*, vol. 16, no. 1, p. 39, Dec. 2016.

[19] G. Wander, M. Bansal, and R. Kasliwal, "Prediction and early detection of cardiovascular disease in South Asians with diabetes mellitus," *Diabetes Metabolic Syndrome: Clin. Res. Rev.*, vol. 14, no. 4, pp. 385–393, Jul./Aug. 2020.

[20] N. Ogden, P. AbdelMalik, and J. Pulliam, "Emerging infectious diseases: Prediction and detection," *Canada Communicable Disease Rep.*, vol. 43, no. 10, pp. 206–211, Oct. 2017.

[21] M. Scheffer, J. Bascompte, W. Brock, V. Brovkin, S. Carpenter, V. Dakos, H. Held, E. Nes, M. Rietkerk, and G. Sugihara, "Early-warning signals for critical transitions," *Nature*, vol. 461, no. 7260, pp. 53–59, Sep. 2009.

[22] M. Nogueira Lima, A. Luiz dos Santos, and G. Pujolle, "A survey of survivability in mobile ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 1, pp. 66–77, 4th Quart., 2009.

[23] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," *Mach. Learn.*, vol. 101, pp. 59–84, Oct. 2015.

[24] R. Bellman, "Dynamic programming," *Science*, vol. 153, nos. 37–31, pp. 34–37, 1966.

[25] C. Milling, C. Caramanis, S. Mannor, and S. Shakkottai, "Network forensics: Random infection vs spreading epidemic," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, pp. 223–234, Jun. 2012.

[26] Rapid7. (2017). *WannaCry Update*. Accessed: May 8, 2020. [Online]. Available: https://blog.rapid7.com/2017/05/16/update-on-wannacry-vulnerable-smb-sh ares-are-widely-deployed-and-people-are-scanning-for-them/

[27] E. Olivieri and M. Vares, *Large Deviations and Metastability*. Cambridge, U.K.: Cambridge Univ. Press, Feb. 2005.

[28] R. Wang, "First-principles prediction of ferroelastic phase transition in AlPO4," *Solid State Commun.*, vol. 155, pp. 88–91, Feb. 2013.

[29] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[30] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. 10th IEEE Int. Conf. Netw. Protocols, . Proceedings.*, Nov. 2002, pp. 312–321.

[31] A. E. G. Ferreira and M. Nogueira, "[Translated from Portuguese] Identifying volumetric DDoS botnets based on graph signaling processing," in *Proc. Workshop de Gerência e Operação de Redes e Serviços (WGRS)*, 2018, pp. 1–14.

[32] E. Biglar Beigi, H. Hadian Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 247–255.

[33] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.

[34] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.

[35] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.

BRUNO MARTINS RAHAL received the B.Sc. degree in control and automation engineering from the Universidade Federal de Santa Catarina. He is currently pursuing the master's degree in computer science with the Universidade Federal do Paraná. His research interests are botnet detection and network security. He is a member of the Center for Computational Science (CCSC) Research Team.

ALDRI SANTOS received the Ph.D. degree in computer science from the Federal University of Minas Gerais. He is currently a Professor with the Department of Informatics, Federal University of Paraná (UFPR). He is the Leader of the Wireless and Advanced Networks (NR2) Research Team. His research interests are network management, fault tolerance, security, data dissemination, and sensor networks. He has served as a member of the technical committee in security information from the Brazilian Computer Society.

MICHELE NOGUEIRA received the Ph.D. degree in computer science from the UPMC-Sorbonne University, Laboratoire d'Informatique de Paris VI (LIP6). From 2016 to 2017, she was on a Sabbatical Leave at Carnegie Mellon University. She is currently an Associate Professor with the Computer Science Department, Federal University of Paraná. Her research interests include wireless networks, network security, and resilience. She was an Associate Technical Editor for the *IEEE Communications Magazine* and the *Journal of Network and Systems Management*.