FlowHacker: Detecting Unknown Network Attacks in Big Traffic Data using Network Flows

Luis Sacramento[1,3] Ibéria Medeiros[2] João Bota[3] Miguel Correia[1]
[1]INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[2]LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal
[3]Vodafone Portugal, Portugal
luis.sacramento@tecnico.ulisboa.pt, imedeiros@di.fc.ul.pt, joao.bota@vodafone.com, miguel.p.correia@tecnico.ulisboa.pt

***Abstract*—Traditional Network Intrusion Detection Systems (NIDSs) inspect the payload of the packets looking for known intrusion signatures or deviations from normal behavior, but inspecting traffic at the current speed of Internet Service Provider (ISP) networks is difficult or even unfeasible. This paper presents an approach to detect malicious traffic and identify malicious hosts by inspecting flows, leveraging a combination of unsupervised machine learning and threat intelligence, without requiring either previous knowledge about attacks or traffic without attacks. The approach was implemented in the FlowHacker NIDS and evaluated with two kinds of traffic flows: synthetic traffic flows and real ISP traffic flows.**

***Index Terms*—Intrusion detection, flows, machine learning**

I. Introduction

Network security is becoming increasingly important since the inception of computer communications. Network security systems have to be able to keep up with this challenge that grows with the speed of the Internet Service Providers (ISPs) links, which now have backbones with bandwidths in the order of 1 to 10 Gbps. Faster links support new forms of communication and services not possible in the past, but also create the opportunity for new attacks that can pass unnoticed by protection mechanisms such as Network Intrusion Detection Systems (NIDSs).

Traditional NIDSs are either signature-based or behaviour-based and usually operate using deep packet inspection (DPI), meaning that they analyse the payload of the packets passing through specific points of the network (e.g., an edge or border router), looking for a certain signature or behavioral pattern. These approaches require, respectively, knowledge about existing attacks and traffic without attacks for training purposes, neither of which is simple to obtain [1]. This kind of analysis is feasible in reasonably slow link connections, but not in modern high-speed backbones. Moreover, nowadays most of the traffic payload is encrypted due to the use of protocols such as HTTPS and IPSec, which makes this kind of inspection even harder and less useful [9].

An alternative to these approaches is the inspection of *network flows* [2], [3]. The concept of flow was first proposed by Cisco in the context of its NetFlow router feature [2]. A flow can be defined as a sequence of packets with a common set of features, passing through an observation point, in a given period of time. Flows are a way of monitoring communication without inspecting the content of the packets, using instead high-level information about connections (e.g., source/destination IP address, source/destination port) but not the data transferred itself. Analyzing this information is more efficient than doing DPI in terms of protection of the privacy of users and consumption of computational resources, once flows do not carry payload content, requiring thus less processing to be analyzed. Network flow analysis allows, for example, the detection of internal and external actions such as network misconfiguration and policy violation [8]. They can detect many network layer and transport layer attacks, but not application layer attacks such as SQL injection or buffer overflows.

Machine learning has been used in NIDS in the context of behaviour-based detection, also called anomaly detection [7], [5]. Either traditional knowledge-based NIDSs or Netflow-based systems can also use machine learning techniques, but the precision and accuracy of these systems depend on the completeness of the knowledge they have about the threats that they will detect, as they need to be fed and trained with that knowledge. Machine learning techniques aim to provide knowledge to such systems, allowing them to discover hidden patterns in input data based on the knowledge they learned, and classify that data. However, even using machine learning, there are challenges in analyzing network flow data, such as the huge amount of traffic flow becoming from larger and faster networks as, for instance, connection links of ISPs.

This paper presents an approach to detect malicious traffic, even if as part of new attacks, and to identify the malicious hosts involved by inspecting network flows. The approach uses a combination of unsupervised machine learning techniques, without *a priori* knowledge, and threat intelligence information to achieve its goal. *The approach is based on the following key insights: there is much more normal traffic than malicious traffic; malicious traffic is qualitatively different from normal traffic; and similar traffic within each cathegory (normal, malicious) can be summarized using unsupervised machine learning.* The approach involves splitting traffic (flows) in clusters, with the larger corresponding to normal traffic and the smaller are the ones we have to worry about. For the latter, the approach proposes a classification method based on unsupervised machine learning to classify them as malicious or benign, detecting thus malicious traffic and identify malicious hosts. This classification allows reducing drastically the amount of time spent in analyzing the flows, reducing the size

of the problem of processing the amount of traffic at the speed of ISP networks. The approach works in loop, iteratively and continuously detecting network attacks and malicious hosts. Between iterations, clusters are classified and learned, so that this knowledge can be used in the following iterations. This form of learning provides increasing autonomy to the system and may significantly reduce the network managers' constant need for intervention, although not being completely free from human intervention, as no NIDS is. In fact human intervention is unavoidable when the goal is to *detect attacks without requiring either previous knowledge about attacks (signatures) or traffic without attacks (clean traffic for training)*.

The paper also presents the FLOWHACKER NIDS that implements our approach. This software uses Hadoop MapReduce [4], [11] to summarize networks flows and a set of machine learning algorithms to process these summaries, besides providing visual tools for human analysis. We evaluated FLOWHACKER with two kinds of traffic (synthetic and real) and it identified botnet command and control servers, SSH brute-force attacks, and denial of service events.

The main contributions of the paper are: (1) an approach for improving network security based on the inspection of network flows by using a combination of unsupervised machine learning techniques to detect intrusions; (2) an iterative learning process; (3) a NIDS that implements the approach; (4) an experimental evaluation that shows the ability of the system to detect intrusions in computers communication using network flows.

II. FLOW-BASED NIDS APPROACH

The approach involves processing network flows using a combination of unsupervised machine learning algorithms and threat intelligence to detect unknown network attacks. The approach follows the assumption that *most of the traffic is legitimate, so malicious traffic is much less, as well as that malicious traffic is qualitatively different from normal traffic*. Having this assumption in mind, the application of the unsupervised machine learning algorithm allows splitting the malicious traffic from the clean traffic, so that the biggest clusters are those containing normal traffic, whereas the smaller ones are those that may be malicious (although that is not mandatory; there may small clusters of legitimate traffic). These assumptions in combination with the use of flows allow to cover (1) the difficulty of reacting to an unknown pattern when real traffic is analyzed, and (2) the slow processing and analysis of the traffic payload. The first drawback may be countered by using an unsupervised machine learning algorithm, and the second by doing the analysis at flow-level.

The approach acts in loop for improving the knowledge that has been acquired (learned) and for achieving better performance in terms of detection. This means that for each loop iteration all phases involved in the approach are executed. Therefore, for each set of collected flows, the approach gains insights from them, improves such insights with threat intelligence information, applies an unsupervised algorithm on the improved insights for getting clusters, and then classifies the smaller clusters as being malicious or benign hosts by using a classifier method based on a unsupervised algorithm. Lastly, these new classified clusters are added to the existing knowledge for the unsupervised data set to be used in the next loop iterations. This learning phase between loop iterations makes the approach self-learning, allowing it to increase its knowledge gradually with every iteration.

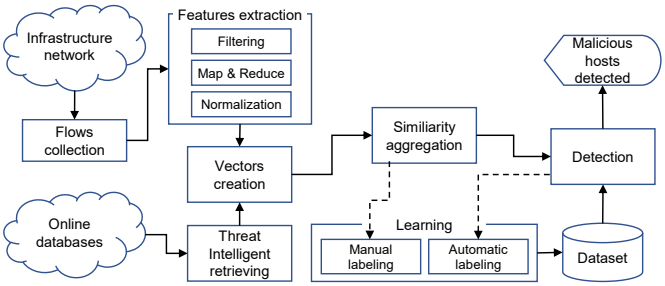

Fig. 1: Overview on approach architecture

The approach comprises six phases, as shown in Fig. 1:

1) *Flows collection*: to collect flows from different hosts or routers belonging to a network infrastructure, corresponding to a certain period of time (e.g., a day). Each flow summarizes a set of packets collected from a host during a period of time.
2) *Features extraction*: to extract data from the collected flows in order to create vectors of features that allow characterizing the flows. Flows are filtered to extracting relevant data, afterwards this data goes through MapReduce for getting statistics and summarizing their values, then they are normalized and the vectors created.
3) *Threat intelligence retrieving*: to automatically retrieve information about threats from online databases, namely blacklists of subnets and IP addresses. Afterwards, this information is used to complement and complete the data of vectors of features.
4) *Similarity aggregation*: to apply an unsupervised machine learning algorithm over the feature vectors to aggregate similar vectors (i.e., with similar feature values), resulting in clusters that represent hosts having a similar behavior. The biggest clusters are assumed to contain clean traffic and the smaller clusters have to be further analyzed.
5) *Detection*: to detect unknown network attacking hosts. Based on the knowledge acquired in the learning phase a classification method using unsupervised machine learning classifies the small clusters as being malicious or clean traffic. In the first loop iteration, the classification is done manually, but automatically subsequently when the system has already knowledge from the previous classifications. Clusters are labeled with its classification and the classified data set is updated with them.
6) *Learning*: to learn the new cluster classifications resulting from the detection phase. For each loop iteration, the clusters classified as malicious are learned for later to be used in the next loop iterations by the classifier.

III. FLOWHACKER

We implemented our approach in the FLOWHACKER NIDS, which we developed in Python. The system is composed by two modules – *similarity* and *classification* –, for aggregating feature vectors in clusters and detecting malicious hosts, respectively. In addition, FLOWHACKER interacts with the Hadoop framework for running MapReduce over the flows, obtaining aggregated flows and vectors of features.

Before using FLOWHACKER, the first step is to gather the flow collection, obtained using NetFlow-enabled routers placed at the border routers between the core network and the connection to the *ISP*. For the sake of analyzing and treating these flows, all of this data is converted to the SiLK format. Next, a filtering to the flows is performed for getting the features needed to compose the feature vectors (see Table I). Afterwards, these filtered features are processed by the Hadoop framework. Hadoop is an open-source framework that features both distributed storage and parallel processing of Big Data, making it very scalable to very large amounts of data. To support the parallel data processing, Hadoop implements Google's MapReduce algorithm [4]. This model can be divided in two main steps, *mapping* and *reducing*, that realize the processing operations.

TABLE I: Set of features that describe an aggregated flow in the form of a vector.

Feature		Description	#
L2	AggregationKey	IP address used as identifier	1
	NumSIPs / NumDIPs	The number of IP addresses contacted	
	LocationCode	Code for the country associated with the address	
L3	NumSports	The number of different source ports contacted	2
	NumDport	The number of different destination ports contacted	3
	ICMPRate	Ratio of ICMP packets, and total number of packets	13
	SynRate	Ratio of packets with SYN flag over the total	14
L4	NumHTTP	The number of packets to/from port 80 (HTTP)	4, 8
	NumIRC	Number of packets to/from ports 194 or 6667 (IRC)	5, 9
	NumSMTP	Number of packets to/from port 25 (SMTP)	6
	NumSSH	Number of packets to/from port 22 (SSH)	7, 10
Statistic	TotalNumPkts	Total number of packets exchanged	11
	TotalNumBytes	Overall sum of bytes	15
	PktRate	Ratio of the number of packets sent and its duration	12
	AvgPktSize	Average packet size	16

FLOWHACKER starts with the similarity module which allows the users managing the clusters, such as change or calculate the number of clusters (k), generate the clusters, and visualize their contents, and then the classification module gets the smaller clusters resulting of the first module, verifies if they are malicious or not, classifying them, and updates the dataset with the malicious ones for further classifications. Moreover, the tool has a terminal interface that was developed to facilitate the similarity aggregation tasks and visualize the outcomes.

IV. EXPERIMENTAL EVALUATION

The objective of the experimental evaluation was to answer the following questions: (1) Is FLOWHACKER able to detect attacks against synthetic data and real data? (2) Is FLOWHACKER able to identify the type of attacks performed? (3) How does it perform in terms of false positives and false negatives? In order to validate our approach, we evaluate FLOWHACKER with two datasets. In Section IV-A the ISCX synthetic dataset[1] is used, and in Section IV-A the evaluation uses real data provided by the large Portuguese ISP. Both sections answer to questions 1 to 3.

A. Attack Detection with Synthetic Data

The ISCX dataset consists in flows collected during one week, and aims to provide a complete testbed for IDSs [10]. All flows in this dataset are labeled, therefore allowing for a validation of the accuracy of the FLOWHACKER. Upon the cluster generation and respective manual labeling of this data, the results were compared to the ground truth provided by the original dataset. After the data that was labeled, we proceeded to train our classifier, which is the classifier for further flows to be analyzed. The next two sections provide an analysis under the similarity task results and classification results, whereas the last section discusses the validation results.

1) Cluster Analysis: The ISCX IDS dataset is divided in 6 sets, each one representing a weekday, from Saturday to Thursday. Attacks were detected in all of these days, except for Wednesday, that was found to be a regular intrusion-free day. For the data of each day, we proceeded with its filtering, extraction, and normalization. Next, it was processed by Hadoop, and then we used FLOWHACKER configured for 10 clusters. Next we present the clustering results.

Saturday. By analyzing cluster contents, we found one cluster that presented features that are rather alarming. In this one, the number of different source ports used and number of connections through the SSH port are highlighted, being the number of connections through the SSH port at its absolute maximum value. A study on Brute-Force SSH attacks [6] has shown that these two features together are representative of a Brute-Force SSH attack. Given that the rest of the traffic presents feature values that are rather normal (i.e., none of them is indicting the presence of an intrusion), we considered that the flow present in this cluster was performing such an attack, therefore highlighting it as an intrusion flow, as the remainder of the traffic was considered to be normal.

Sunday. The results for this day showed a rather different pattern. Unlike the previous day, now we see that almost every cluster presents very high feature values. Features such as the average packet size, the number of source ports and the number of HTTP connection are high in the great majority of the clusters. Also, the number of SMTP connections was also found to be very high in two different clusters. This behavior shows us that something is not right, as the SMTP connections are usually grouped together in a single cluster, and this analysis shows us that two different clusters have these characteristics. Taking this into account we assume that these flows, although having this feature with very high values, were grouped into different clusters because they have a different behavior, and therefore showing us that these flows are not normal. As for the remaining clusters, we found 4 that have very high values for the the number of HTTP

[1] http://www.unb.ca/research/iscx/dataset/iscx-IDS-dataset.html

connections, alongside with the number of different source ports and average packet sizes. These three features together may indicate that a large volume attack is being perpetrated, exploring the HTTP protocol, therefore also labeling these clusters as attacks.

Monday. Moving on to Monday, when looking at their clusters' content, we found two clusters that immediately distinguish themselves from the rest. The first one has a mean value of 0.998 for the ICMP Rate, being it the cluster with the biggest dimension (it contains 375 different flows); the second one has the number of destination ports and number of SMTP and IRC connections at its highest value possible. However, this is not considered an alarming behavior, because even though the value for ICMP Rate in indeed at a very high value, no other feature in that cluster was showing a high value; as for the second cluster, throughout the whole evaluation of the system, there was always one cluster with such characteristics, and we can infer that this cluster corresponds only to regular clients using email services. Apart from these two, other four clusters also present an alarming pattern. All these clusters share high values for the number of different source ports, number of HTTP connection and also for the average packet sizes. Such pattern may be attributed to a DoS attack, as each host is send a great amount of packets from many different ports, all direct to the port 80 (or port 8080, in some cases), with an high average packet size. This is the case of the DoS HTTP Flood attack. However, this is an attack that is easily identifiable by inspecting its payload, and this flow-based approach does no allow us to perform such an analysis, being these features our only way to hint the presence of such an attack.

Tuesday. Analyzing Tuesday, we observed that there were multiple clusters with a very high value for the ICMP Rate. However, this feature appear alone, i.e., it is the only feature in these clusters that has a relevant high value, no other features show up, apart from one cluster that also has a high value for the average packet size, which also does not correspond to a recognizable pattern. From all these clusters, the one that grabs our attention is the tenth, which features a high value for the number of source ports, HTTP connections and average packet size. Also, it has a high packet rate, average packet size and total number of bytes. From what we have seen so far, this can only correspond to an attack, and therefore the content of this cluster was labeled as being an attack.

Thursday. Reaching the last day with intrusions, Thursday, just like when we analyzed Saturday, there is one cluster that was found to have an absolute maximum value for the number of SSH connection alongside with a high value of number of different source ports, thus indicating us the presence of a Brute-Force SSH attack. Also, three other clusters have high values for the number of different source ports, number of HTTP connections and also a high average packet size, also possibly indicating the presence of an attack. Therefore, these two clusters were also labeled as malicious.

TABLE II: FLOWHACKER results with the synthetic dataset, showing how results improve with the number of iterations.

	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday
True Positive	1	3	2	0	0	1
False Positive	0	1	0	1	5	4
False Negative	0	17	4	3	0	0

2) Unsupervised Classification: Parallel to this daily analysis, the system may also autonomously identify malicious activities using the classification method implemented, which classifies the smaller clusters resulting from the similarity module based on an unsupervised algorithm and a training dataset. Before the FLOWHACKER is able to classify data it is needed, at least, malicious labeled data from the first day, which results from manual intervention. From this day on, the classifier is able to produce results on its own, and these results may be refined with every iteration of the system (for the purposes of this work, an iteration corresponds to the period of one day), by training the system again, as new patterns are identified and manually labeled.

The classifier was trained for the first day with data from the analysis for Saturday, as seen in Section IV-A1. As expected when the system processed the clusters for that day, it correctly identified the malicious flow. However, when trying to classify the results for Sunday, the classifier did not found any sort of malicious activity. This was due to the fact that the system's only knowledge about the intrusions observed during Saturday, which does not give sufficient information to the system to detect other attacks. After training the system once again with the analysis done for Sunday, the classifier was now able to identify the malicious activities, although it could not identify them all. This same behavior was found when classifying the remainder of dataset throughout the rest of the days.

Table II describes this analysis in further detail. Along the system iterations, FLOWHACKER detected 7 attacks, which correspond to flows whose were perpetrating the attack with greater intensity, i.e., producing large volumes of traffic, whereas the remainder of the attacks were not successfully identified. Although we observed 24 false negatives, it is visible that they were decreased during the iterations, which is justified by increasing of knowledge that the system was learned. Regarding the Wednesday, the system misclassified 5 flows (i.e., false positives) as being malicious, when the traffic relative to Wednesday is all normal, intrusion-free traffic. This is due to the fact that Wednesday was one of the day that had the largest amount of traffic, and therefore the flows that belong to it also produced higher features values, leading to it being perceived as malicious. At the end the system misclassified 11 flows as being malicious. This means that it needs some refine in the classification method. However, we prefer to have a system giving false positives than unreported attacks, i.e., with false negatives.

3) Results Validation: The ISCX dataset is a dataset with known attacks put in a database containing information about them. The dataset is used as the ground truth validation, so we compared the FLOWHACKER results with this ground truth to

validate them and find out the accuracy and precision of our system. Each flow belonging to ISCX is identified by a unique ID, meaning that the flows contained in our clusters have this ID. Therefore, through this ID we were able to traceback it to its IP address (which is stored in the database), and this way we were able to identify the malicious hosts.

We were able to verify that all clusters containing malicious flows identified by our system actually such flows are malicious, meaning thus that our system is able to detect and classify correctly attacks under traffic analysis. Also, we observe that correctly we identify the type of attack evolved in such flows. However, we also verified that our system generates some false positives and misses some malicious flows (false negatives), such as Table II shows.

These results suggest a positive answer to questions 1 to 3.

B. Attack Detection with Real Data

The following results were obtained from data provided by the above-mentioned ISP. Our NIDS approach assumes that data is collected using NetFlow-enabled routers, e.g., placed at the border routers between the core network and the connection to the ISP. All the data that reaches the Portuguese company network arises from the ISP, that connects to Portuguese company via routers that are placed at the borders of the core network. These routers are NetFlow-enabled, and are protected by firewalls, in order to filter any wanted data, according to their security policies, therefore ensuring that only supposedly clean data reaches its clients. However, not all bad traffic is filtered, which is why there are needed extra security measures, such as NIDSs. So, the data that reaches these routers from the outside (i.e. the data incoming from the ISP) is collected by NetFlow, for further analysis. For the purpose of this evaluation, a data sample was collected for a few hours.

1) Data Analysis: After performing the filtering and MapReduce phases, we obtained two datasets comprising the source flows and the destination flows, respectively. Therefore, the following analysis reports to each of these datasets. The clustering was performed with 30 clusters (i.e., $k = 30$).

Source aggregation key clustering.

By looking at the source aggregation key clustering content, we observe five clusters that present high feature values, as shown in Table IV (clusters 13, 15, 17, 21 and 30). The first presents a high number of different source ports, as well as a high number of total bytes sent. However, such pattern was not found to be suspicious, as the number of source ports itself does not represent an alarming network threat, as opposed to the number of destination ports, and no address found in this cluster was found to be in any IP blacklist. When analyzing the second one, we see that is presents a high connectivity to various users, under various ports, receiving communication on an IRC port, and communicating through HTTP, with a high number of packets sent, as well as a high number of bytes. This leads us to assume that this machine is either a major spammer, or it could be a DoS attack, given its traffic pattern, and it was thus labeled as being a malicious host. Moving on the third cluster, it was found to have a high number of SSH communications alone, which could represent a Brute-Force SSH attack, in just like had observed in the previous section, thus also being labeled as malicious hosts. The fourth presented a high number of IRC (which is used as a portal for botnet's C&C communications) communications, alongside with an high average packet size. This feature distribution led us to consider that this could a botnet communicating, and thus labeling it as malicious hosts. Moreover, we decided to track the IP addresses, and check them against a number of public available blacklists, which confirmed and categorized the IP addresses as sources of Botnets/Spam. One of the IP addresses lead us directly to an authentication page of a C&C. When analyzing the last alarming cluster, we observed that it presented a high number of SMTP communications, but when analyzing its IP addresses, we found that these were only mail servers communication, and we found no harm in it. Prior to this analysis, all of the IP addresses present in the malicious clusters, were found to be present in several blacklist, thus confirming our suspicion.

Destination aggregation key clustering.

Regarding the destination aggregation key clustering, we observe that there are 5 clusters (16, 20, 22, 25, 29) with alarming features (Table IV). Analyzing cluster 16, we see that it has a feature distribution that is similar to what we had understood as a DoS HTTP Flood attack when analyzing the ISCX data, except that this cluster is missing a high value for the number of HTTP connections. Therefore, this could also represent a DoS attack, but directed to other applications, e.g., DNS. We cannot be sure of this attack, because none of the monitored ports are presenting high values, and so we cannot infer anything more about it. Cluster 20 presents a high number of different source IP addresses, destination ports and number of bytes. Because these flows do not have a high average packet size, it could possibly indicate that this a network scan, as these flow contacted many different port of many different IP addresses, resulting in a high value of bytes sent throughout this process. Cluster 22, on the other hand, presents a feature distribution that is similar to what had previously perceived as a DoS attack: it has a high number of different source IP addresses, number of source ports, number of HTTP connections, and a number of bytes sent. However, it still lacks a high value for the average packet size. Therefore, this may be, just like cluster 20, a network scan, but this time directed to the HTTP application, i.e. it may be a probing of a website in order to locate some vulnerability, for example. Cluster 25 presents a high number of different IP addresses, average packet sizes and number of bytes sent. These features alone do not seem to correspond to a malicious behavior, as we interpreted them a simple burst of traffic. At last, cluster 29 hold a have number of source IP addresses, number of destination ports, number of source HTTP connections, average packet sizes and number of bytes sent. This pattern very similar to what we have seen for the DDoS IRC botnet

TABLE III: Source key clustering content using K-Means

Clusters	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
# Flows	1605	51773	6485	13305	529	1730	1729	21507	8523	8522	1498	4686	10	5653	1	824	5	4606	1864	1676	12	107	13	2233	2264	8091	10	23897	16843	35
Features																														
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-	-	-	-	0.368	-	1.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.667	-	-	-	-	-	-	-	0.384 ; 0.184	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.237 ; 0.158
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.542 ; 0.18	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.626 ; 0.21	-	-	-	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-	-	-	-	0.61 ; 0.208	-	0.843	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.261	-	-	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

TABLE IV: Destination key clustering content using K-Means

Clusters	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
# Flows	34565	1447	7734	888	312	17180	53767	497	7	2442	2177	3089	232	62515	644	12	1242	1699	4987	84	93	23	10883	754	36	22543	16	503	3	8680
Features																														
1	-	-	-	-	-	-	-	-	0.840 ; 0.110	-	-	-	-	-	-	0.413 ; 0.067	-	-	-	-	0.159 ; 0.069	-	0.209 ; 0.047	-	-	0.150 ; 0.093	-	-	-	0.156 ; 0.09	-
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.229 ; 0.012	-	-	-	-	-	-	-	-	0.236 ; 0.093	-
3	-	-	-	-	-	-	-	-	0.857 ; 0.155	-	-	-	-	-	-	0.589 ; 0.178	-	-	-	-	-	0.517 ; 0.141	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.838 ; 0.154	-
5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.396 ; 0.181	-	-	-
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-	0.260 ; 0.162	-	-	-	-	-	-	-	-	-	-	-	-	0.116 ; 0.108	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	0.854 ; 0.167	-	-	-	-	-	-	0.445 ; 0.108	-	-	-	-	-	-	-	-	-	0.123 ; 0.054	-	-	-	0.102 ; 0.014	-
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.226 ; 0.013	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	0.258 ; 0.074	-	-	-	-	-	-	-	-	-	-	0.147	-	0.212 ; 0.069	-	-	0.449 ; 0.180	-	-	-	0.400 ; 0.046	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

attacks, expect for the number of IRC connections. Therefore, this may also correspond to infected hosts that are being used as a third party for attacks, but contacting its botmaster through a C&C server other than an IRC, or they could be victims of an attacker who is using spoofed IP addresses to use them as a third party.

2) Discussion: Table V summarizes the information regarding the intrusions detected throughout this analysis. As presented in the table, the FLOWHACKER NIDS was able to detect and identify attacks in real traffic of a company, both in incoming and outgoing traffic. Such results allow us to verify that real threats are effective and a concern for companies. Also, they allow us to verify that our system is effective in detection of such threats and can be used by companies for avoiding them. In addition, they allow to answer positively to questions 1 to 3.

TABLE V: Real data analysis results

Cluster #	Aggregation Key	Highlighted Features	Type of Attack
15	Source	1, 3, 5, 8, 11, 15	Spam / DoS
16	Destination	1, 3, 6	DoS
17	Source	10	Brute-Force SSH
20	Destination	1, 2, 15	Network Scan
21	Source	9, 16	Botnet Communication
22	Destination	1, 3, 8, 15	Web Application Probing
27	Source	1, 2, 5, 8, 11, 15	DDoS IRC Botnet
29	Destination	1, 2, 4, 11, 15	DDoS Botnet

V. CONCLUSION

The paper presents a system able to analyze traffic from faster networks, such as the fast connection links of Internet Service Providers (ISPs). The system is based on analysis of netflows, turning it capable of analyzing such connection links. The approach behind of the system allows detecting malicious hosts without requiring previous knowledge about what we were looking for or clean training data. A combination of data mining techniques for the feature extraction from netflows, and machine learning techniques for data analysis allows the detection of malicious behaviors without requiring specific training, except for the inevitable human intervention in a first run of the system. The FLOWHACKER NIDS implements the approach and it was evaluated with synthetic and real data.

Acknowledgements. This work was partially supported by the EC through project H2020-700692 (DiSIEM), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/00408/2013 (LASIGE) and UID/CEC/50021/2013 (INESC-ID). We warmly thank Henrique Santos for feedback on a previous version of this work.

REFERENCES

[1] G. Bruneau. The history and evolution of intrusion detection, SANS institute, 2001. https://www.sans.org/reading-room/whitepapers/detection/history-evolution-intrusion-detection-344.

[2] B. Claise. Cisco systems netflow services export version 9. RFC 3954, RFC Editor, October 2004. http://www.rfc-editor.org/rfc/rfc3954.txt.

[3] B. Claise, B. Trammell, and P. Aitken. Specification of the ip flow information export (ipfix) protocol for the exchange of flow information. STD 77, RFC Editor, September 2013. http://www.rfc-editor.org/rfc/rfc7011.txt.

[4] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[5] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion detection systems. *Computer Networks*, 31(8):805–822, Apr. 1999.

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

[7] H. Javitz and A. Valdes. The SRI IDES statistical anomaly detector. *Proceedings IEEE Computer Society Symposium on Research in Security and Privacy*, 1991.

[8] B. Li, J. Springer, G. Bebis, and M. H. Gunes. A survey of network flow applications. *Journal of Network and Computer Applications*, 36(2):567–581, 2013.

[9] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. Blindbox: Deep packet inspection over encrypted traffic. *ACM SIGCOMM Computer Communication Review*, 45(4):213–226, 2015.

[10] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.

[11] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *IEEE 26th Symposium on Mass Storage Systems and Technologies*, pages 1–10, 2010.