DOI:
10.1016/j.comnet.2016.05.018

Document Version
Peer reviewed version

Analysis of high volumes of network traffic for Advanced Persistent Threat detection

Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, Alessandro Guido

Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia, Italy

Abstract

Advanced Persistent Threats (APTs) are the most critical menaces to modern organizations and the most challenging attacks to detect. They span over long periods of time, use encrypted connections and mimic normal behaviors in order to evade detection based on traditional defensive solutions. We propose an innovative approach that is able to analyze efficiently high volumes of network traffic to reveal weak signals related to data exfiltrations and other suspect APT activities. The final result is a ranking of the most suspicious internal hosts; this rank allows security specialists to focus their analyses on a small set of hosts out of the thousands of machines that typically characterize large organizations. Experimental evaluations in a network environment consisting of about 10K hosts show the feasibility and effectiveness of the proposed approach. Our proposal based on security analytics paves the way to novel forms of automatic defense aimed at early detection of APTs in large and continuously varying networked systems.

Keywords: Security analytics, Traffic analysis, Advanced Persistent Threats, Data exfiltration.

1. Introduction

Advanced Persistent Threats [1, 2] (APTs) represent the most critical menace to modern organizations. Unlike automated broad-range attacks, APTs are human-driven infiltrations, perpetrated over long periods of time, customized for the targeted organization after some intelligence analyses, possibly on open sources, and can even leverage unknown exploits to infiltrate vulnerable systems [3]. The economic cost for an organization that is a victim of an APT can reach even millions of dollars [4], and its reputation may be compromised. Since large corporate networks continue to increase in terms of traffic and number of connected devices, it is a tough research challenge to design and implement advanced network monitoring systems and security analytical algorithms that can detect APT attacks in a rapid way. Traditional security solutions based on pattern matching (e.g., [5]) work well for detecting known attacks, but they cannot identify APTs because attackers typically exploit unknown vulnerabilities, and use standard protocols and encrypted communications (e.g., HTTPS) to evade detection [1]. Moreover, existing traffic analyzers are able to detect common types of attacks (e.g., distributed denial of service and worms [6, 7, 8, 9, 10]), but they are inadequate to identify APTs because an expert attacker mimics normal behavior and compromises a limited number of specific hosts thus avoiding spreading infections as typical automatic malware does. Another problem of present detection systems installed in large architectures is represented by the huge numbers of generated alarms, at least in the order of thousands per day. A similar context would require either a large number of dedicated security analysts or, more likely, the need to overlook most alarms. As an additional observation, our focus on traffic logs reflects a realistic enterprise scenario in which host-based logs (e.g., system calls) would be extremely expensive to collect and analyze.

The real goal of this paper should be clear: we do not aim to identify the hosts that are surely compromised because this is an unrealistic goal in the APT case. Instead, we want to detect -out of thousands hosts characterizing the information system of a large organization- the few hosts that show some suspicious activities. In such a way, we allow security analysts to be more effective because they can focus their competence and attention on a limited number of hosts. As a means to reach this goal, we propose a novel framework that is able to gather and analyze huge volumes of traffic and can detect some of the key phases of APT-related activities corresponding to data exfiltrations [1]. These features are extracted and evaluated over time for all internal hosts, each corresponding to a point in a multi-dimensional feature space. Suspiciousness of movements and positions is evaluated for each host by inspecting its past and by comparing it to the other hosts of the observed network. The final output is a ranked list of hosts, that allows the security analysts to concentrate their expertise on the top-k suspicious ones.

Special attention is given to the scalability and efficiency of the proposed framework, as most analyses can be executed in parallel for each internal host. By analyzing the network flows instead of raw traffic data, our approach achieves high performance and limited computational and storage costs. As additional remarkable feature, the proposed framework can work even for encrypted communications because it does not need to

Email address: {mirco.marchetti, fabio.pierazzi, michele.colajanni, alessandro.guido}@unimore.it (Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, Alessandro Guido)

inspect payload data.

The main contributions of this paper can be summarized as following:

- we characterize network statistics of real and large network environments, and define a model aimed to detect APT-related activities with specific attention to data exfiltrations;
- we propose a set of algorithms that are able to score suspiciousness of APT activities by evaluating movements and positions of the internal hosts in a multidimensional feature space over time;
- we design and implement a prototype that is applied to a real networked system consisting of about 10K hosts and that demonstrates the feasibility and effectiveness of the proposed approach.

To the best of our knowledge, this paper presents the first proposal of models, algorithms and analyzers integrated in a real prototype that can support security analysts to detect the most suspicious hosts that may be involved in APT-related activities and specifically in data exfiltrations. Indeed, despite the relevant risk represented by APTs, few results exist for detecting similar attacks. Most papers [11, 12, 13, 14, 15] consider one or more case studies of famous APTs, describe their main features, and propose some best practice with no attempt to consider a general framework for automatic or semi-automatic detection. Other solutions [16, 17, 18] are focused on the design of the main components of a framework for identifying APTs, but they do not propose detection rules and leave details to future works. Friedberg et al. [19] present a solution that considers the analysis of host-based security logs and is complementary to our network-based approach. Moreover, in our experience, in large network scenarios it is almost impossible to gather and analyze host-based security logs because this activity requires the deployment of software agents on each computer and has high costs in terms of performance and management overhead.

The remainder of the paper is structured as follows. Section 2 compares our work with related literature. Section 3 describes the typical lifecycle of an APT and the challenges related to their detection. Section 4 presents an overview of the proposed framework. Section 5 motivates the choice for features that are tailored to efficiently detect possible data exfiltrations in large network environments. Section 6 defines how suspicious movements in the feature space are evaluated and how the different metrics are combined into a ranking score. Section 7 presents an experimental evaluation on a real large network environment consisting of about 10K hosts. Finally, Section 8 presents conclusions and possible directions for future research.

2. Related Work

We propose a novel framework for ranking internal hosts that are likely to be involved in APT attacks by monitoring high volumes of network traffic efficiently and effectively. Our proposal combines heuristics based on knowledge related to known APTs with behavioral and statistical models that are able to capture suspicious network activities.

The main motivation behind our work relies in the observation that traditional security solutions based on pattern matching (e.g., solutions based on network intrusion detection systems [5, 20, 21]) are not adequate for identifying APTs, due to the usage of encrypted covert communication channels and zero-day attacks. Most proposals based on statistical traffic analyses that could work in presence of encryption are mainly focused on anomaly detection [22] of more popular types of attacks (e.g., distributed denial of service [9, 10] or worms [23, 24]), and do not consider APTs; moreover, the fact that attackers in APTs try to mimic normal behavior [1] further complicates detection through statistical analyses.

We can consider three main related areas in the literature: APT detection, botnet detection, and insider threat detection.

2.1. APT detection

Despite the relevance of the APT problem, the literature focusing on this topic is still limited. Most articles [11, 1, 2] describe and analyze popular and publicly disclosed APT cases, such as Stuxnet [12], Duqu [13] and Flame [14]. However, these studies only discuss some mitigation techniques mainly based on best practices, and do not discuss solutions for automatic detection of APTs.

Other works [25, 16] try to formalize the APT detection problem. In [25], the authors propose a 7-phase detection model to identify different steps of an APT. In [16], the authors propose an *attack pyramid* that aims to capture movements of an attacker through different domains (e.g., physical, network, application). In [17] the authors propose possible building blocks for a framework of APT detection. However, all these approaches are limited to the proposal of guidelines that should be used to build methods for APT detection, but the definition of detection rules and approaches is left to future work.

A more practical work based on graph analytics [26] proposes a new metric that measures the vulnerability of a network environment with respect to the risk of privilege escalation. However, this work focuses on the proposal of a single graph-based metric, that is intended only as a means to evaluate the vulnerability of a network to APTs (especially when granting new privileges to users) but does not help in detection of APTs in operational environments.

In another interesting work [19] the authors propose an anomaly detection system for identifying APTs from several security logs. However, their approach requires a huge number of logs (also collected on individual hosts) that are often impractical to obtain, and the output of their proposal may be extremely difficult to interpret for a security analyst, since their approach is agnostic to the given input. On the other hand, our focus on network traffic is more practical and our output is easier to interpret, as it consists of a ranked list of hosts that performed suspicious network activities possibly related to data exfiltrations or key phases of APTs.

Other related works focus specifically on the detection of data exfiltrations. Some of them [27, 28] require a big amount

of host-based log data that would be unfeasible to collect in large organizations, whereas our focus is on network traffic that can be easily collected through some network probes inside the organization. Bertino et al. [29] focus on the analysis of *DataBase Management System* (DBMS) access logs in order to detect suspicious patterns for possible exfiltrations, but do not consider network traffic and their approach is limited to the detection of possible DBMS-based exfiltrations. Liu et al. [30] propose a framework for detecting data exfiltrations through analysis of network communications, and their approach is based on automatic signature generation but has two major shortcomings. First, the authors assume that attackers perform plaintext communications that can be matched with their signatures, whereas most APTs use encrypted or obfuscated communications [1]. Moreover, they assume that all sensitive data is known a-priori in order to be able to generate the signatures for detecting their exfiltration. On the contrary, our proposal can be applied even if the attacker uses encrypted communications and standard protocols such as HTTPS, since it does not require payload analysis. In addition, we do not require to define a-priori the set of sensitive data that could be exfiltrated.

2.2. Botnet detection

It is also interesting to compare the problem of APT detection with the *botnet detection* domain [31]. A botnet is a huge set of distributed compromised hosts, controlled by one or more command and control servers. Several approaches (e.g., [32, 33, 34]) have been proposed in the literature in the last few years for detecting infected hosts and command and control servers related to botnet activities. However, there are some core differences that prevent the adoption of botnet methods in the APT domain.

First, the scale of the problem is completely different: while botnets consist of thousands or millions of hosts, APTs are human-driven attacks directed at a specific organization. Hence, botnet approaches that aim to detect similar behaviors in groups of hosts (e.g., through clustering of traffic features) cannot be applied in the APT domain. This is because only a few internal hosts are infected, and command and control servers may use targeted protocols with only a subset of victim hosts. Hence, it is not possible to perform broad-range clustering analyses as those proposed in the botnet detection literature in order to determine huge volumes of hosts with anomalous traffic patterns. Moreover, infection strategies are different: whereas APTs often use spear phishing and zero-day exploits, botnets may try to replicate by themselves in a more aggressive way. Our framework is specifically tailored to the APT domain, and takes into account the specific limitations and challenges related to the identification of suspicious hosts.

2.3. Insider threat

Insider threat research [35] shares some similarities with the APT problem as well. Indeed, an APT aims to take control of a legitimate host inside of an organization, and the attacker will try to emulate normal behavior in order to avoid detection. However, an important difference is that an insider may not need to exfiltrate the data through a network, hence many approaches of insider threat detection focus on host-based logs [36] and honeypot strategies [37] instead of analyzing network traffic as it is done in our approach. An important observation is that the framework proposed in this paper could be easily integrated in insider threat detection systems, while its approach based on traffic analyses can contribute significantly to existing insider threat solutions, although insider threat detection is not its primary objective.

3. Scenario

In this section we introduce the lifecycle of modern APTs and the main intuitions behind the proposed framework for ranking suspicious hosts.

3.1. APT lifecycle

APTs have peculiar characteristics that make their detection extremely challenging [2]. Unlike traditional attacks, they are targeted to a specific organization, often use zero-day vulnerabilities, and span over long periods of time (a "low-and-slow" approach). They attempt to simulate normal behaviors and the use of strategies like *social engineering* complicates their detection even more.

Another peculiar aspect is that APTs are often conducted by cybercrime organizations and not by individuals [1]. Each member of the group has peculiar skills and knowledge that increases difficulty of detection for defenders and security analysts. For the sake of simplicity, in the remainder of this paper we will refer to them with singular form: *attacker*.

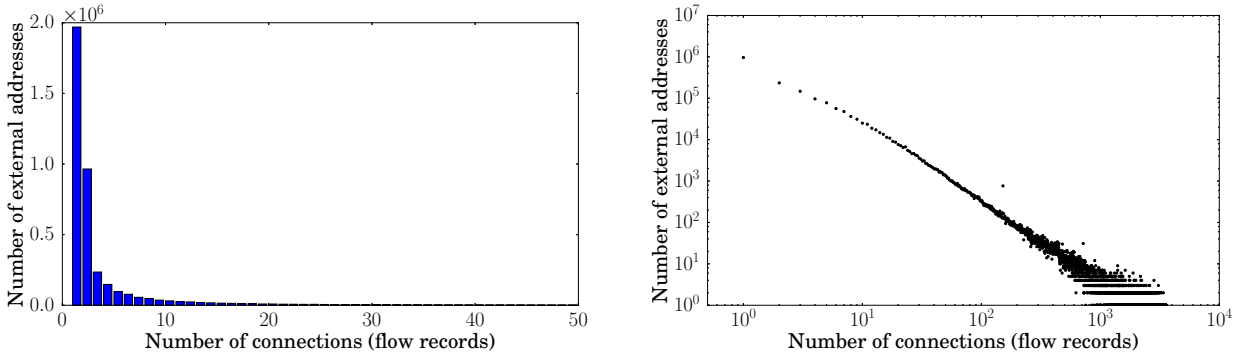
We now discuss the APT characteristics that might be captured by inspecting traffic data and security logs. In particular, we refer to the definition of APTs proposed by [1], that identifies five main phases:

1) reconnaissance;
2) compromise;
3) maintaining access;
4) lateral movement;
5) data exfiltration.

In the *reconnaissance* phase, the attacker aims to identify possible entry points in the victim organization. From a traffic perspective, it may involve *scan operations* from external networks.

In the *compromise* phase, the attacker typically creates a *spear phishing email* that contains an attachment that, if opened, exploits a zero-day vulnerability to infect a victim machine, with the purpose of infiltrating the target network. In particular, the malicious software that installs on the victim host is often referred to as *RAT*: Remote Access Trojan, or Remote Administration Tool.

In the *maintaining access* phase, the RAT contacts a command and control server (CnC) of the attacker in order to receive commands that must be performed on the target network. A peculiarity of this phase is that the connection is initialized



(a) linear scale

(b) log scale

Figure 1: Distribution of the number of connections to external hosts initiated by internal hosts.

by the victim host and not by the attacker [1]. This occurs for two main reasons: (i) connections initialized by internal hosts are usually allowed by the organization firewall, and (ii) in this way the attacker can improve his chances of not being detected. This is because a download from an external host would be extremely suspicious, and it would be easily detected by traditional security solutions such as intrusion detection systems [5].

In the *lateral movement* phase, the attacker tries to gain access to other hosts inside the target network with greater privileges that might be required to access valuable resources. For example, the RAT may try to perform an internal scan on the observed network, or try to initialize new connections with other internal hosts (e.g., via SSH).

Finally, in the *data exfiltration* phase, the stolen data is sent to one or more remote servers controlled by the attacker. This phase can be performed either in a *burst*, in which all data are exfiltrated at once, or in a *low-and-slow* way, if the attacker wants to continuously steal data. Some recent examples [3] of data exfiltration are represented by the *Adobe Leak* in 2013, in which 9 GB of encrypted password data were stolen, and *Ashley Madison* in 2015, in which attacker leaked their databases (the size of one of the databases was about 40 GB).

Each phase has particular characteristics that may leak some information that may be captured from traffic logs. In the upcoming subsection, we show why it is extremely challenging to detect phases of APTs with respect to traditional external attacks, with special attention to the data exfiltration phase.

3.2. Challenges in APT detection

There are several characteristics of the APT scenario that make it extremely more challenging than any other external threat. We can identify the following main motivations:

- *imbalanced data*: APTs hide in weak signals among huge amounts of data, and are hence very difficult to detect; unlike botnets [31], in APTs usually only a few hosts get infected, hence command and controls (CnC) and victims are harder to detect;
- *base-rate problem*: since APTs are related to very rare events that span over long periods of time, the number of false positives can easily be extremely high [38];
- *lack of publicly available data*: this well-known shortcoming for network security is even more critical when considering the APT domain, since most companies that are victim of APTs have no interest in releasing logs and details about such events;
- *use of encryption and standard protocols* (e.g., HTTPS) prevents efficacy of commonly adopted network security solutions, such as signature-based intrusion detection systems [5].

To better understand the issues related to APT detection, let us consider the following example. An intuitive approach for detecting communications with CnCs, RATs or possible attempts of data exfiltration would be to isolate and analyze connections with external hosts that have been *rarely contacted* from the internal hosts of an organization. This idea looks plausible because an APT attacker does not want to be discovered, hence he will try to perform only a limited number of communications.

However, analyses that we performed on a real and large network environment consisting of about 10K hosts revealed that most network statistics still follow *heavy-tailed* distributions. For example, let us consider in Figure 1 the distribution of the number of contacts established by internal hosts toward external IP addresses. The X-axis represents the number of contacts from internal hosts to external IP addresses during one day. The Y-axis represents the number of unique external IP addresses that were contacted exactly x times. As an example, the first 3 bars in the histogram 1(a) show that about 2M external IP addresses were contacted only once, about 1M were contacted twice, and about 220K were contacted 3 times. Figures 1(a) and 1(b) report the same distribution on a linear and logarithmic scale, respectively. Other network statistics and results referring to different time granularities (e.g., hour, week) are not reported for space reasons and correspond to similar results.

From these figures, we can conclude that most of the external hosts are contacted only once or a few times.

This heavy-tailed nature of network statistics is partially related to the recent widespread adoption of *cloud providers*, that are now commonly used for remote storage and Web applications. By analyzing the characteristics of the observed network environment, we have verified that even during short periods of time (e.g., one hour) the variety of external addresses corresponding to content delivery networks and cloud providers contacted by internal hosts is really high. Moreover, the majority of external IP addresses are contacted only a few times. This result was unexpected, because when considering the external addresses contacted by all internal hosts in a large network environment, we would expect that the set of external addresses corresponding to cloud providers would be limited or at least contacted many times, since all the organization internal hosts belong to the same geographical area. This high variability of range of IP addresses holds for different cloud providers and for different time windows (hour, day, month). A possible solution would be to *whitelist* all cloud providers as *benign*, but if the objective is the identification of APTs, this is not a viable option, since the attacker can easily create cloud accounts as CnCs or as data exfiltration points [39, 40]. Hence, the presence of an extremely dynamic range of IP addresses related to different cloud providers further complicates APT detection.

This nature of the network statistics complicates the application in the APT domain of several well-known statistical approaches for anomaly detection [22], such as:

- *threshold-based approaches* [41], that would not work well because no effective threshold can be set on a heavy-tailed distribution;
- *clustering approaches* [42], that would likely find two main clusters, one related to the head, and one related to the tail of the distribution;
- *boxplot rule* [41], that would not be effective for outlier detection as the underlying distributions are not Gaussian.

Since we are aware that traditional anomaly detection approaches are extremely difficult to adopt in modern network environments [43, 44], we propose an innovative framework in the APT domain that allows to *rank* the hosts involved in suspicious activities possibly related to APTs. The proposed approach models the behavior of individual hosts as feature points in a multidimensional space, and compares internal host statistics both with respect to their past and with respect to all the other internal hosts. The proposed framework then assigns a score to each internal host on the basis of the suspiciousness of its movements and positions in the feature space. This allows the security analysts to focus only on the top-k suspicious hosts, instead of manually inspecting huge volumes of data. In particular, in this paper we focus on the ranking of hosts possibly related to data exfiltration activities, as a first effort of a more comprehensive framework for APT detection.

An overview and the details of the proposed framework will be described in the upcoming sections.

4. Framework overview

In this section, we present an overview of the proposed framework for ranking internal hosts possibly involved in data exfiltrations as part of APTs. The final output is an ordered list of hosts that performed suspicious network activities possibly related to APTs. In this way, security analysts can focus only on the manual investigation of the top-k suspicious hosts.

The main objective of our framework is to overcome challenges posed by APT detection through an innovative approach with the following characteristics:

- unlike many existing APT detection solutions that require analyses of huge amount of host-based logs, we focus on traffic information that can be easily collected through network probes deployed in the observed network environment;
- to efficiently deal with high volumes of network traffic, we extract and analyze *flow records*; this design choice achieves better results both in terms of *storage occupation* and in terms of *computational cost* for the analysis;
- unlike the majority of existing anomaly detection approaches that focus mainly on network-wide statistics, we focus on individual hosts and on comparative statistics to identify the hosts that perform suspicious activities (i) with respect to their past behavior, and (ii) with respect to the other internal hosts of the organization;
- we propose a set of features that is tailored to identify hosts possibly involved in data exfiltrations;
- moreover, the proposed ranking approach does not rely on deep packet inspection, hence it can work even on encrypted traffic.

We refer to a scenario of an enterprise network consisting of a few thousands of hosts. Figure 2 reports the main phases of the proposed framework:

1) flow collection and storage;
2) feature extraction;
3) feature normalization;
4) computation of suspiciousness scores;
5) ranking.

We observe that the second, third and fourth phases can be executed independently for each internal host, hence the proposed approach can scale to very large networks comprising more than hundreds of thousands of hosts by executing these phases in parallel.

Since raw traffic data would be impractical to analyze and store, the *first phase* aims to collect *flow records* [6, 7]. Each record contains some information extracted from the IP header, such as source and destination addresses and ports, and protocol (e.g., TCP, UDP, ICMP). The adoption of flow records instead of raw traffic data brings several advantages in terms of performance for the approach:

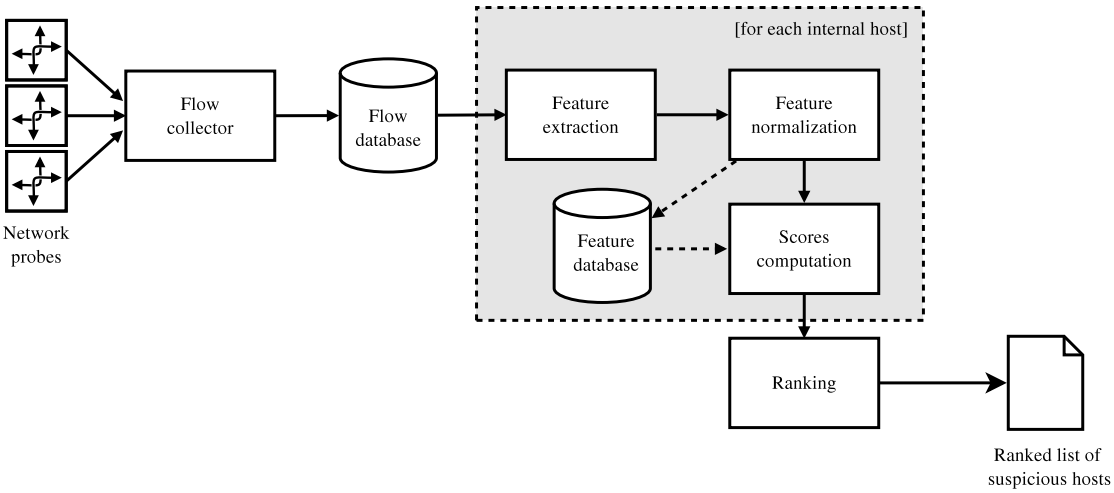


Figure 2: Framework overview.

- flow records can be efficiently stored and compressed, even when they are collected for long periods of time (e.g., years);
- processing of flow records is computationally more feasible than analyzing huge volumes of raw traffic data [6].

Since in this paper we are interested in the ranking of suspicious activities possibly related to data exfiltrations, we focus only on *outgoing traffic* generated by internal hosts, that is, on the behavior of traffic flows directed from internal to external hosts. We recall that this choice is under the realistic assumption that all malicious connections related to an APT will be initiated by internal hosts, as an APT attacker wants to evade detection [1]. Any connection initiated by external hosts to internal hosts not corresponding to servers can be marked as suspicious by adopting traditional intrusion detection systems [5].

The *second phase* of our framework involves *extraction* of features that are relevant for data exfiltration. These features are computed for each internal host every time interval T (e.g., once per day). Details, examples and motivations on the chosen features will be discussed in Section 5. We highlight that feature extraction is performed by a dedicated server that analyzes network flows, hence this activity does not consume computational resources on any production host in the monitored network environment.

The *third phase* involves a *normalization* of the different features, since each of them is characterized by a different heavy-tailed distribution, hence they must be normalized for comparison purposes. This is achieved through a normalization metric taken from [45], that is specifically tailored to normalize heavy-tailed distributions with different characteristics.

For each internal host and for each time t_i, the *fourth phase* involves the computation of statistics related to host movement in the feature space, in order to evaluate suspiciousness by considering both magnitude and direction. In particular, our approach considers three points in the feature space:

- the feature values of the host at present time, t_i;
- the centroid of the feature values of the same host in a historical window of size W, that is, between t_{i-1-W} and t_{i-1};
- the centroid of the feature values of all hosts at time t_i.

The historical window is used to represent the past behavior of the hosts in the network. The *direction* of the movement with respect to the recent past is also taken into account, so that movements along uncommon directions are considered suspicious.

Finally, in the *fifth phase* the ranking algorithm assigns a suspiciousness score to each internal host, that allows it to be compared with all the other hosts. In particular, the level of suspiciousness is evaluated as a *linear combination* of:

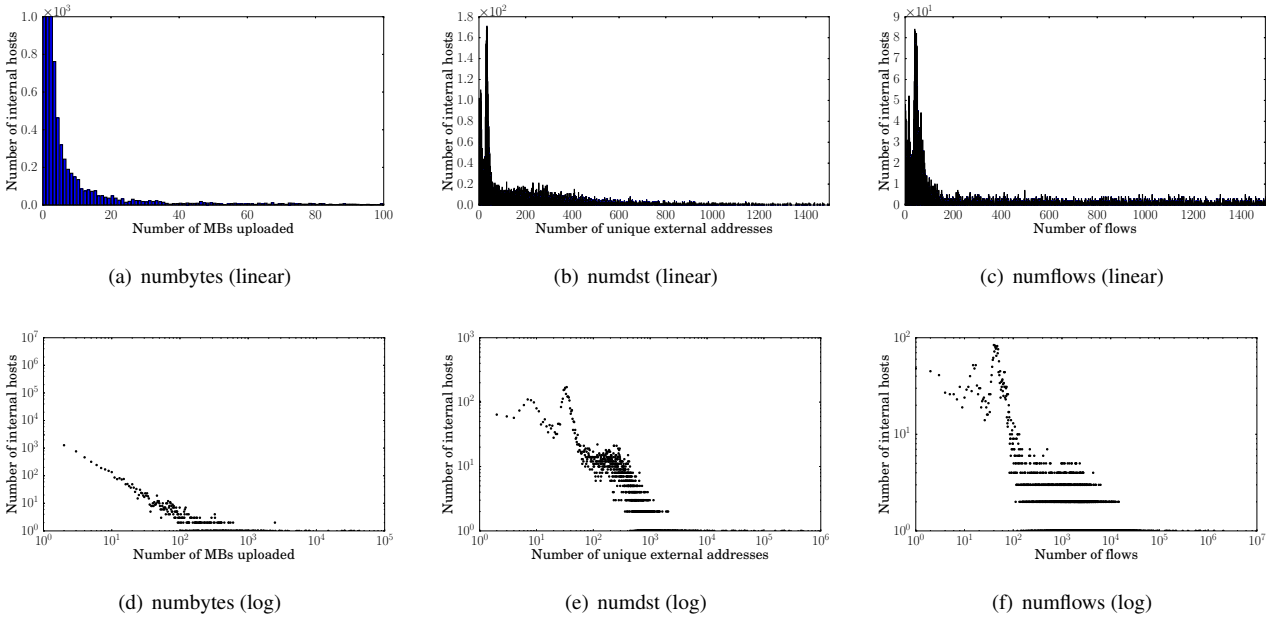
- normalized distance of an internal host with respect to the centroid of the feature space;
- percentage increment of the magnitude of the movement;
- unlikelihood of the direction of the movement, with respect to the movements of all the internal hosts in the observed network environment.

The result is a ranked list of suspicious hosts, that analysts can use to focus only on the top-k suspicious ones.

The details for each of these phases will be discussed in the upcoming sections, along with several examples and evaluations on flow records collected from a real network consisting of about 10K hosts.

5. Feature extraction and normalization

In this section, we present and motivate a set of features that we extract for each internal host with the purpose of detecting possible data exfiltrations. Then, we discuss how the components of each feature vector are normalized and compared.



(a) numbytes (linear) (b) numdst (linear) (c) numflows (linear)

(d) numbytes (log) (e) numdst (log) (f) numflows (log)

Figure 3: Distribution of feature vectors for the whole internal hosts population.

5.1. Feature extraction

For each internal host in the observed network environment, we extract a set of features that is tailored to detect data exfiltrations through analysis of suspicious and rare movements in the feature space.

Let us define H_I and H_E as the sets of *internal* and *external* hosts, respectively. For each internal host $h \in H_I$, a *feature vector* $\mathbf{x}_t(h)$ is computed that is defined as the following ordered tuple:

$$\mathbf{x}_t(h) = \left(x_t^1(h), x_t^2(h), \ldots, x_t^N(h)\right) \tag{1}$$

where $x_t^i(h)$ corresponds to the value of the i-th component of the feature vector at time t, with respect to internal host h. In particular, the feature vector values $x_t^i(h)$ are computed every *sampling period* T. This allows building a *time series* with the feature vector values for each internal host $h \in H_I$. These series will be used in the next phases for evaluating the movements of a host over time. For the sake of simplicity, unless otherwise specified, in the remainder of the paper we use a simplified notation in which we omit h, that is: the feature vector of internal host h is referred to as $\mathbf{x}_t(h) = \mathbf{x}_t$, with components $x_t^i(h) = x_t^i$.

Without loss of generality, in this paper we refer to a time granularity $T = 1$ day, because we have verified that this is a good granularity for several reasons, among which:

- APTs involve operations that could go on for days or months [1], hence excessively fine time granularities (e.g., minutes) could generate too much noise in the analysis;
- by producing a ranked list of suspicious hosts once per day, the security analyst can easily focus on the top-k suspicious hosts for manual investigation.

It is also possible to consider other time granularities, according to the characteristics of the observed network environment and to possible domain knowledge of the security analyst.

We now propose a *set of features* that are aimed to recognize hosts involved in suspicious network activities possibly related to the *data exfiltration* phase of an APT:

1) **numbytes**: *number of megabytes uploaded* by internal hosts to external addresses (i.e., possible points of exfiltration);
2) **numflows**: *number of flows* (typically connections) to external hosts initiated by internal hosts;
3) **numdst**: *number of external IP addresses* related to a connection initiated by an internal host.

We now motivate the choice for each of these features.

numbytes allows us to monitor deviations of uploaded bytes, as they may correspond to data exfiltrations. For example, if a host exhibits a tenfold increase in the amount of uploaded bytes, it may be involved in APT-related activities. In this paper, the amount of bytes uploaded by internal hosts is represented in megabytes.

numflows is used to monitor data transfers initiated by internal hosts [1, 2]. Exfiltrations are initiated by internal hosts for two main reasons: (i) outgoing connections are not blocked by most firewalls; (ii) connections initiated by external hosts would look suspicious and could be easily detected through traditional signature-based intrusion detection systems [5].

numdst makes it possible to identify anomalous behaviors that involve a change in the number of distinct destinations contacted by each internal host. As an example, if the number of external IPs contacted within a given time window by an internal host remains stable while the number of uploaded bytes or connections greatly increases, it may correspond to a data exfiltration or APT-related activities.

We are aware that *numflows* and *numdst* are correlated to some extent: if the number of flows initiated by an internal host increases greatly, the number of destinations is expected to increase as well. The opposite is true for a decreasing number of flows. Although the machine learning community usually recommends to choose uncorrelated features to maximize the information contained in a feature vector [46], we observe that we are not interested in a *classification* problem, but rather in identifying suspicious movements in the feature space. Hence, we willingly choose two correlated features to capture the presence of internal hosts moving in directions of the feature space that violate the expected correlation between *numflows* and *numdst*.

A representation of the distribution of these features with respect to about 10K hosts of a real network environment is reported in Figure 3. In particular, we have that Figures 3(a), 3(b) and 3(c) report on the X-axis numbytes, numdst, and numflows, respectively, as a function of the number of internal hosts on the Y-axis in linear scale. Figures 3(d), 3(e) and 3(f) report the same features on a logarithmic scale. We observe that for $x > 100$ all the features follow a long-tailed distribution. As discussed in Section 3, this complicates the application of many anomaly detection algorithms, such as threshold-based ones [22].

A 3D representation of the proposed feature space is reported in Figure 4, where the three axes represent the different features in logarithmic scale. It is possible to observe that, when considered as a whole, the internal hosts cover several positions in the feature space. By monitoring their movements and their distances both from their history and from the centroid of the feature space, we aim to detect the most anomalous hosts, that might be related to data exfiltration activities.

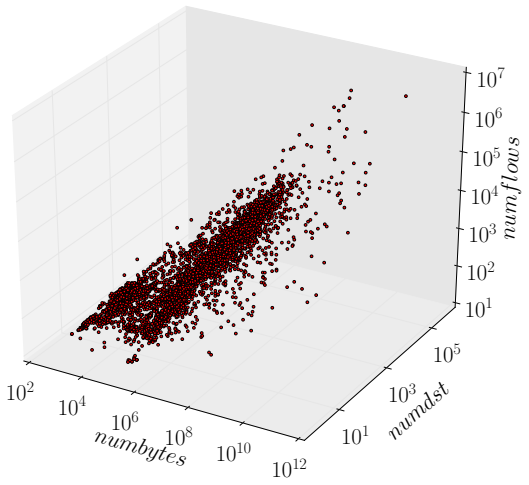
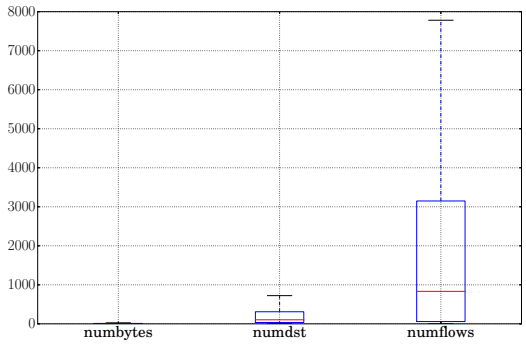


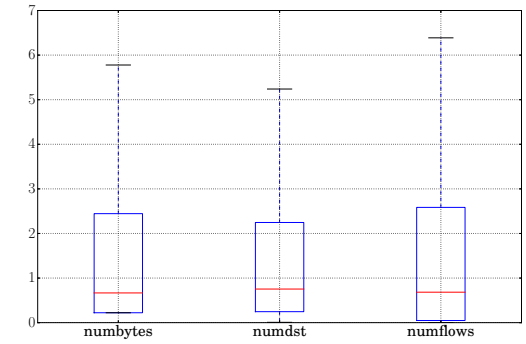
Figure 4: Example of representation of the internal hosts in the feature space with logarithmic scale.

5.2. Feature normalization

To perform a fair comparison of the movements and positions of the internal hosts, we have to normalize their distributions. One of the most common normalization techniques is *range normalization* [41], that maps a distribution in a range between 0 and 1 by normalizing with respect to the maximum and the minimum of a range of values. However, we showed that in the considered context each feature is characterized by a different long-tailed distribution (Figures 3). Hence, out-of-scale values are frequent, and range normalization would yield poor results because most values would be near 0.



(a) Before QWM normalization



(b) After QWM normalization

Figure 5: Scale normalization with two-sided quartile weighted median.

To overcome this issue we adopt the *two-sided quartile weighted median* (QWM) metric [45, 47, 48], that is defined as:

$$QWM(D) = \frac{Q_{.75}(D) + 2 \cdot Q_{.50}(D) + Q_{.25}(D)}{4} \quad (2)$$

where $Q_k(D)$ corresponds to the k-th quantile of the dataset D. We observe that this measure takes into account both the median of the values ($Q_{.50}$), and the variance of the data, that is represented by the first and third quartiles. This makes the QWM robust and independent of the distribution of the data, and also suitable to normalize values of heavy-tailed distributions to similar central tendencies.

If we consider the feature vector $\mathbf{x}_t$ at time t related to an internal host h, we can obtain the normalized feature vector $\bar{\mathbf{x}}_t$ by performing on each component x_t^i, $i \in \{1, 2, \ldots, N\}$ the

following normalization:

$$\bar{x}_t^i = \frac{x_t^i}{QWM(\mathbf{X}_t^i)} \quad (3)$$

where $\mathbf{X}_t$ is the set of feature vectors of all internal hosts $h \in H_I$, and $\mathbf{X}_t^i$ is the set of the i-th components. We observe that normalization is performed with respect to the whole population to fairly compare the behaviors of all internal hosts.

The effectiveness of QWM normalization for our scenario is demonstrated by Figures 5(a) and 5(b). Figure 5(a) represents through boxplots [41] the distributions of the three features (*numbytes* is expressed in MB, while *numdst* and *numflows* are pure numbers), computed on a real network of about 10K hosts. Figure 5(b) shows that the scales and the major descriptive statistics of the different distributions become comparable after QWM normalization. In the upcoming sections we will assume that all the feature values $\mathbf{x}_t$ have already been normalized.

6. Computation of suspiciousness scores

Let us consider the normalized *feature vector* of the internal host h at time t:

$$\mathbf{x}_t = (x_t^1, x_t^2, x_t^3) \quad (4)$$

where x_t^i are *numbytes* (x_t^1), *numdst* (x_t^2) and *numflows* (x_t^3).

At each time t we compute for each internal host three *suspiciousness scores*:

- s_t^1: *distance* from the centroid of the feature space;
- s_t^2: *magnitude* of the movement in the feature space;
- s_t^3: unlikelihood of movement *direction*.

Sections 6.1, 6.2 and 6.3 present the details and motivations for the computation of these scores. Finally, Section 6.4 discusses how these three scores are combined to compute the final suspiciousness score of each internal host.

6.1. Distance from feature space centroid

First, we compute a score s_t^1 that depends on the position of the host h at time t with respect to all the other internal hosts. The purpose is to determine whether a host at time t is situated in an anomalous region of the multidimensional feature space.

Let us consider $\mathbf{X}_t$ as the set of all positions of internal hosts in the feature space at time t, that is:

$$\mathbf{X}_t = \{\mathbf{x}_t(h) : h \in H_I\} \quad (5)$$

We then define $\mathbf{c}(\mathbf{X}_t)$ as the *centroid* of the feature space at time t. In particular, it is computed as:

$$\mathbf{c}(\mathbf{X}_t) = \left(\frac{\sum_h x_t^1(h)}{|\mathbf{X}_t|}, \frac{\sum_h x_t^2(h)}{|\mathbf{X}_t|}, \frac{\sum_h x_t^3(h)}{|\mathbf{X}_t|} \right), h \in H_I \quad (6)$$

where $x_t^i(h)$ is the i-th component of the feature vector $\mathbf{x}_t$ of internal host $h \in H_I$, and $|\mathbf{X}_t|$ represents the cardinality of $\mathbf{X}_t$. Hence, the centroid is a feature vector resulting from the mean of the components of the feature vectors associated with all the hosts.

Finally, for each internal host h we can compute the score s_t^1 as the Euclidean distance d_t between the feature vector $\mathbf{x}_t$ and the centroid of the feature space $\mathbf{c}(\mathbf{X}_t)$:

$$s_t^1 = d_t(\mathbf{x}_t(h), \mathbf{c}(\mathbf{X}_t)) = \sqrt{\sum_{i=1}^{3} \left(x_t^i(h) - c^i(\mathbf{X}_t)\right)^2} \quad (7)$$

The higher the value of s_t^1, the farther an internal host is from the centroid of the feature space.

We observe that the feature vector normalization through the QWM discussed in Section 5.2 is fundamental for a fair computation of the magnitude of d_t in Eq. 7. If the magnitude was computed over non-normalized features, there would always be one feature overwhelming the others (see also Figures 5).

6.2. Magnitude of movement in the feature space

The purpose is to identify a distance metric that is suitable for measuring suspiciousness of movements of internal hosts in the feature space.

The movement of $\mathbf{x}_t$ from $t-1$ to t is represented as a *distance vector* in a Euclidean space:

$$\mathbf{x}_t - \mathbf{x}_{t-1} = (x_t^1 - x_{t-1}^1, x_t^2 - x_{t-1}^2, x_t^3 - x_{t-1}^3) \quad (8)$$

A drawback of this representation is that the magnitude of the distance vector will most likely be higher for internal hosts that are far from the origin of the feature space. For instance, if a host usually uploads about 10GB of data per day with a standard deviation of 1GB, then its distance vector would be always higher than a host that normally uploads a few MB per day and then suddenly uploads 100MB, although a tenfold increment would be really suspicious with respect to possible data exfiltrations.

Moreover, if we compare $\mathbf{x}_t$ with respect to $\mathbf{x}_{t-1}$ only, we are implicitly assuming that $\mathbf{x}_{t-1}$ is representative of the past behavior of the host. Such assumption may not be appropriate in real networks. A *time window* is a more feasible approach and creates an evolving *behavioral baseline* for each internal host. We define $\beta_{t-1}(W)$ as the *centroid* of the set of features vectors $\{\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-1-W}\}$, where W is the size of the time window. $\beta_{t-1}(W)$ is defined as:

$$\beta_{t-1}(W) = \left(Q_{.50}(\cup_j x_j^1), Q_{.50}(\cup_j x_j^2), Q_{.50}(\cup_j x_j^3)\right) \quad (9)$$

where $j \in \{t-W-1, \ldots, t-1\}$, and each i-th component of $\beta_{t-1}(W)$ corresponds to the median of the last W values of the component x_t^i of feature vector $\mathbf{x}_t$.

We can now define a distance metric to measure movements in the feature space. To take into account *relative* deviations and movements, we define the *movement vector* $\mathbf{m}_t$ as the *relative difference* between the feature point $\mathbf{x}_t$ and the centroid $\beta_{t-1}(W)$, that is:

$$\mathbf{m}_t = \frac{\mathbf{x}_t - \beta_{t-1}(W)}{\beta_{t-1}(W)} = \quad (10)$$
$$= \left(\frac{x_t^1 - \beta_{t-1}^1(W)}{\beta_{t-1}^1(W)}, \frac{x_t^2 - \beta_{t-1}^2(W)}{\beta_{t-1}^2(W)}, \frac{x_t^3 - \beta_{t-1}^3(W)}{\beta_{t-1}^3(W)} \right)$$

This definition of $\mathbf{m}_t$ allows to fairly determine how much a host deviated from his previous positions.

Finally, the second score s_t^2 is the magnitude of $\mathbf{m}_t$:

$$s_t^2 = \| \mathbf{m}_t \| = \sqrt{\sum_{i=1}^{3} \left(\frac{x_t^i - \beta_{t-1}^i(W)}{\beta_{t-1}^i(W)} \right)^2} \tag{11}$$

The score s_t^2 is important to determine *how much* a host has changed its position with respect to its recent history. For example, if a host suddenly increases his upload rate with respect to its past values, it may be exfiltrating information to one or more external addresses. A similar risk exists if the number of external addresses contacted by that host suddenly decreases greatly while the amount of megabytes uploaded remains stable.

6.3. Likelihood of movement direction in the feature space

The magnitude of $\mathbf{m}_t$ by itself is not sufficient to determine the suspiciousness of a host movement in the feature space. This score is designed to take into account the direction of the movement vector. Uncommon directions (i.e., directions with low probability) should be considered more suspicious. For example, a movement in a direction where the number of flows increases while the number of destinations decreases is unusual.

The direction of $\mathbf{m}_t$ related to an internal host is represented by the *unit vector* $\hat{\mathbf{m}}_t$ that is defined as the following ratio:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{\| \mathbf{m}_t \|} = (u_t, v_t, w_t) \tag{12}$$

where u_t, v_t and w_t are the components of the unit vector $\hat{\mathbf{m}}_t$.

In Figure 6, we report a real-world example of movement directions related to a large network environment consisting of about 10K hosts. Each line in this figure represents a different unit vector $\hat{\mathbf{m}}_t$ related to a different internal host, where the axes correspond to u_t, v_t and w_t. To improve readability, this figure only shows 1000 unit vectors that were randomly sampled among all unit vectors (one for each active host) generated in a day. It is possible to observe that the unit vectors together form a sphere, thus implying that several directions are explored at least once. However, we highlight that the distribution of unit vectors is not uniform, and some regions are much more populated than others. This implies that movements in some directions are more common than movements in other directions.

To understand the distribution of unit vectors of internal hosts, it is convenient to use *spherical coordinates* [41], that can be obtained from the components u_t, v_t and w_t through the following equations:

$$\rho = \sqrt{u_t^2 + v_t^2 + w_t^2} \tag{13}$$

$$\varphi = \arccos\left(\frac{w_t}{\sqrt{u_t^2 + v_t^2 + w_t^2}} \right) \tag{14}$$

$$\theta = \arctan\left(\frac{v_t}{u_t} \right) \tag{15}$$

where $\rho \geq 0$ is the length (magnitude) of the vector, $0° \leq \varphi \leq 180°$ and $-180° \leq \theta \leq 180°$ are two angles that describe the direction of the movement in the feature space. Since all unit vectors $\hat{\mathbf{m}}_t$ have $\rho = 1$ by definition, only two variables (φ and θ) are required to represent the direction of the unit vector.

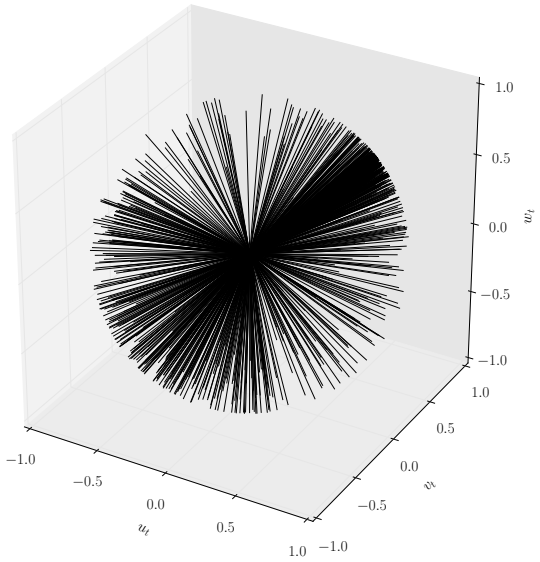


Figure 6: Representation in the (u_t, v_t, w_t)-space of the unit vectors $\hat{\mathbf{m}}_t$ corresponding to the movement directions within a day for a network with about 10K hosts.

Figure 7 shows a histogram representing the number of internal hosts with a certain direction (φ, θ) on the Z-axis, and the values of φ and θ on the other two axes. This figure refers to the statistics of about 10K hosts, computed over a day. Similar results are achieved for the other days in the observed environment, and are not reported only for space reasons.

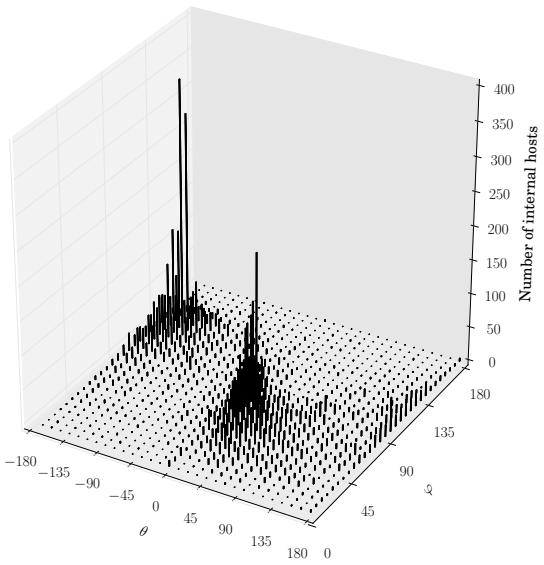


Figure 7: Histogram of likelihood of directions φ and θ in the feature space related to 10K internal hosts.

From Figure 7, it is clear that most internal hosts move just in a small subset of the possible directions. Hence, movements in less common directions represent a viable indicator of suspiciousness. In particular, we can observe two space regions characterized by a much higher population.

To better understand the histogram in Figure 7, we report the 2D projections for φ and θ in Figures 8(a) and 8(b), respectively.

One of the most populated regions in Figure 7 corresponds to values of φ between $110°$ and $135°$ and values of θ between $-180°$ and $-130°$. This space region includes movement vectors for which all three features decrease. In particular, the highest spike within this region represents vectors for which the three components decrease proportionally with respect to each other. Intuitively, this region captures all hosts for which the network activity (uploaded bytes, number of destinations and number of flows) decreases with respect to their recent history.

The other highly populated region in Figure 7 is characterized by values of φ between $80°$ and $90°$ and values of θ between $0°$ and $30°$. This space region captures movements for which all three features increase. In particular u_t (corresponding to the number of uploaded bytes) increases more than v_t (number of destinations) and w_t (number of flows), while v_t and w_t grow proportionally with respect to each other. We can conclude that a considerable increase in the number of uploaded bytes with respect to the previous history is quite common, while strong increases in the number of destinations or in the number of flows are less frequent.

The other less populated regions of Figure 7 correspond to movements in unlikely directions. In particular, for low values of both φ and θ the number of uploaded bytes and of destinations decreases, while the number of flows increases. Moreover, for high values of both φ and θ the number of uploaded bytes and the number of flows decreases, while the number of destinations increases.

The third score s_t^3 is defined as:

$$s_t^3 = 1 - Pr(\hat{\mathbf{m}}_t) \tag{16}$$

where $Pr(\hat{\mathbf{m}}_t)$ represents the probability of a certain direction in the feature space, computed as the value of a bin divided by the total number of internal hosts. Hence, $1 - Pr(\hat{\mathbf{m}}_t)$ is its *complement probability*, that represents the *unlikelihood* of moving in a certain direction in the feature space. The higher s_t^3, the more suspicious is the direction followed by the host.

6.4. Computation of the final score

The final step of our framework for APT detection is to compute the *final score* for each host of the internal network by combining the three suspiciousness scores described in Sections 6.1, 6.2 and 6.3.

The *final score* S_t is computed as a *linear combination* of scores s_t^1, s_t^2 and s_t^3. In particular, we adopt the following formula:

$$S_t = \sum_{j=1}^{3} \left(\delta_t^j \cdot s_t^j \right) \tag{17}$$

where δ_t^j is a normalization weight associated with the j-th score. Since the three scores are characterized by different bounds, scales and distributions, we normalize them by defining δ_t^j through the QWM metric [45, 47]:

$$\delta_t^j = \frac{\sum_{k,k\neq j} QWM(s_t^k)}{\sum_k QWM(s_t^k)}, k \in \{1,2,3\} \tag{18}$$

We note that the QWM values related to the different scores are normalized with respect to the sum of all QWMs.

The final output of our framework is a list of internal hosts ranked in descending order with respect to the final score S_t. Security analysts can use this list to prioritize manual and time consuming scrutiny of network and system activities of suspicious internal hosts.

7. Experimental evaluation

In this section we evaluate the performance and effectiveness of the proposed framework by implementing and deploying a prototype on a real and large network consisting of about 10K hosts. In particular, we applied the framework to five months of network flow records and report the most significant results.

The proposed evaluation aims to demonstrate three main aspects:

- feasibility of *execution times* and *storage requirements* for our framework in a real operational environment;
- *ability to identify hosts* that exhibit suspicious behaviors compatible with data exfiltrations;
- *sensitivity* of the proposed approach with respect to different classes of hosts and different sizes of exfiltrations.

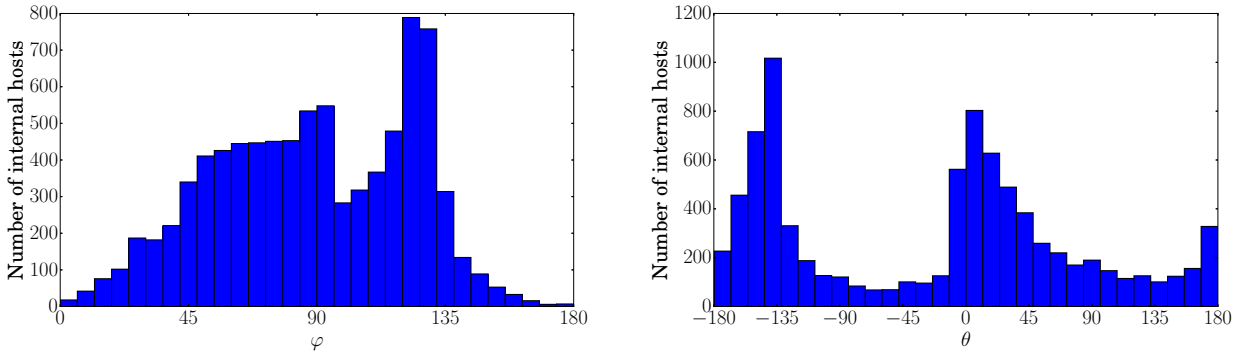
Section 7.1 describes the testbed and the experimental setup, along with details about execution times and storage requirements of the proposed framework. In Section 7.2 we present some detailed experimental results referring to different exfiltrations on a particular day, whereas Section 7.3 presents a comprehensive analysis and comparison of the results of the proposed framework over a five months period with respect to different classes of hosts, different sizes of exfiltration, and the common ranking approach used in traditional security solutions. Finally, Section 7.4 summarizes and discusses experimental results.

7.1. Experimental testbed and framework performance

We first describe the experimental setting and the performance and storage requirements of our approach.

We build a prototype of the proposed framework (Section 4) using different programming languages and tools for each phase:

- the first phase is realized through the *nprobe* tool [49] that collects flow records in the observed large network environment;



(a) movements on φ (degrees)

(b) movements on θ (degrees)

Figure 8: Likelihood of movement directions in the feature space with respect to φ and θ.

- the second and third phases are implemented in *Go*, where we read and extract the features from the flow records generated by nprobe;
- the fourth and fifth phases are realized in *Python*, where we extract, elaborate and rank the suspiciousness scores of each internal host.

The prototype is then deployed on a machine equipped with eight 2.5GHz cores, 128GB of RAM and a 512GB SSD hard drive.

Without loss of generality, we consider a sampling period $T = 1$ day (i.e., the host ranking is performed once per day), and a historical window $W = 14$ days (i.e., for each host we build its behavioral baseline on the basis of a two-week window).

In particular, the flow records have been collected in a real large network environment (class B) with the characteristics reported in Table 1.

Characteristic	**Average value**
number of hosts	$\approx 10,000$ active internal hosts
bitrate	≈ 600 Mbps (business hours)
number of flows	≈ 140 millions records per day

Table 1: Main characteristics of the observed network environment.

We observe that the ranking is performed for all the 10K hosts by analyzing over 140 millions flow records per day. The main performance and storage requirements referring to $T = 1$ day and $W = 14$ days are reported in Table 2.

Statistic	**Time/Storage required [for one day of data]**
Storage of flow records	≈ 1.7 GB
Feature extraction and normalization	$\approx 70 \div 80$ seconds
Score computation and ranking	$\approx 10 \div 20$ seconds

Table 2: Main performance and storage requirements statistics of the proposed framework related to the analysis of one day of data.

We observe that hardware requirements and execution times of the proposed solution are low and compatible with realistic use cases. Moreover, even though the monitored network generates high volumes of network traffic, the storage space requirements make long-term logging and analysis feasible. In our environment, the storage occupation is of about ≈ 47 GB per month, and ≈ 500 GB per year.

7.2. Detection of artificially injected exfiltrations

We now consider the effectiveness of the proposed approach by injecting data exfiltrations in the observed real network environment. The purpose of this experiment is to verify whether the proposed approach is able to capture data exfiltrations despite the huge noise related to the traffic statistics of 10K hosts.

To inject exfiltrations we first selected a random working day $\bar{t}$ (February 5th, 2016) and the host $\bar{h}$ that in $\bar{t}$ uploaded a number of bytes equal to the median of the number of bytes uploaded by all internal hosts. We then executed two experiments that simulate two well known security incidents:

- exfiltration of 40GB of data, approximately the same size of one of the databases exfiltrated from Ashley Madison [3];
- exfiltration of 9GB of data, approximately the amount of data in the Adobe password leak case [3].

We also observe that the main parameters are $T = 1$ day, $W = 14$ days. Moreover, the *optimal number of bins* for the score s_t^3 from Eq. 16 has been computed on the basis of the *Freedman–Diaconis rule* [50] iterated over different days. In the observed environment the optimal number of bins is 10×5 (ten bins for the θ axis and five for the φ axis).

Table 3 and Table 6 report the first 10 hosts ranked by suspiciousness score (for the 40 GB and 9 GB exfiltrations, respectively), together with the values of each of the three partial scores. The columns of this table report the total scores S_t and the details of the individual scores s_t^1, s_t^2 and s_t^3, respectively. The scores associated with the injected exfiltrations are represented in bold.

Host	Score S_t	s_t^1	s_t^2	s_t^3
$\bar{h}$	**6499.72**	**6141.14**	**6249.63**	**0.85**
h_1	4083.40	636.69	4224.00	0.85
h_2	652.88	7258.66	12.55	0.85
h_3	633.70	7173.14	0.32	0.87
h_4	341.73	10.94	356.72	0.76
h_5	280.47	3166.87	0.41	0.85
h_6	185.51	2005.66	8.32	0.74
h_7	158.17	1224.06	51.84	0.85
h_8	136.76	1522.31	1.79	0.85
h_9	128.50	1441.60	0.59	0.85

Table 3: Top-10 hosts ranked by suspiciousness score, exfiltration of 40 GB.

Feature	$\beta_{\bar{t}-1}(W)$	$x_{\bar{t}}$
numbytes	$\approx$ 6.4 MB	40004 MB
numdst	$\approx$ 185.7	168
numflows	$\approx$ 1982.4	1457

Table 4: Feature vector values and history for host $\bar{h}$, exfiltration of 40 GB.

Moreover, in Table 4 we report the feature vector $\mathbf{x}_{\bar{t}}$ of $\bar{h}$, and the centroid $\beta_{\bar{t}-1}(W)$ that represents its past behavior.

It is possible to observe that in this case our approach placed the host $\bar{h}$ at the first place, with a final score S_t, well above those of the other internal hosts. While this result may seem trivial, we highlight that for the day $\bar{t}$ the top uploader was h_3 (ranked 4th), that uploaded more than 47 GB, and the second uploader was h_2 (ranked 3rd), that uploaded about 46 GB. Host $\bar{h}$ uploaded about 4 MB, and after the injection reached about 40 GB. If hosts were ranked according to the number of uploaded bytes, $\bar{h}$ would be in the third position. On the other hand, using our approach host $\bar{h}$ is ranked in the first place due to high values of all three partial scores. In particular, the high value of s_1 is determined by the large amount of bytes uploaded by $\bar{h}$ after having injected the exfiltration, that places $\bar{h}$ far from the centroid of the feature space that represents all internal hosts. The high value of the score s_2 is caused by the sudden increase of uploaded bytes with respect to the recent history of $\bar{h}$. Finally, the value of s_3 shows that the movement of $\bar{h}$ with respect to its previous history followed an uncommon direction, because the number of uploaded bytes increased considerably while the number of destinations and flows decreased (as shown in Table 4).

For the sake of comparison, Table 5 reports the complete feature values of all the other hosts included in the top-10 (hosts h_1 to h_9).

Results related to the exfiltration of 9 GB are presented in Table 6.

In this experiment host $\bar{h}$ was ranked as the second most suspicious host, before h_2 that uploaded 46 GB. We observe that in traditional security systems that commonly sort hosts according to the number of uploaded bytes, $\bar{h}$ would only achieve the seventh position. Table 7 reports the centroid and the feature vector for $\bar{h}$ after the injection of the 9 GB exfiltration.

Host	numbytes	numdst	numflows
h_1	47267	5362	100326
h_2	46712	11	787
h_3	20623	38673	179658
h_4	18829	792	14571
h_5	13122	259	82637
h_6	9457	601	6868
h_7	8043	189	13902
h_8	6264	1059	16651
h_9	4225	68	591

Table 5: Feature values for hosts h_1 to h_9 at day $\bar{t}$.

Host	Score S_t	s_t^1	s_t^2	s_t^3
h_1	4070.95	637.97	4224.00	0.85
$\bar{h}$	**1466.94**	**1373.19**	**1405.88**	**0.85**
h_2	709.83	7259.93	12.55	0.85
h_3	690.03	7174.41	0.32	0.87
h_4	340.30	10.54	356.72	0.76
h_5	305.41	3168.14	0.41	0.85
h_6	201.31	2006.93	8.32	0.74
h_7	167.67	1225.34	51.84	0.85
h_8	148.69	1522.44	1.79	0.85
h_9	139.91	1442.87	0.59	0.85

Table 6: Top-10 hosts ranked by suspiciousness score, exfiltration of 9 GB.

Feature	$\beta_{\bar{t}-1}(W)$	$x_{\bar{t}}$
numbytes	$\approx$ 6.4 MB	9004 MB
numdst	$\approx$ 185.7	168
numflows	$\approx$ 1982.4	1457

Table 7: Feature vector values and history for host $\bar{h}$, exfiltration of 9 GB.

7.3. Sensitivity analysis and comparative evaluation

In this section we evaluate the effectiveness of the proposed approach for different classes of internal hosts. The following experiments are not limited to a single day, but cover the whole set of available data, which is about five months of network flow records. We recall that the number of bytes uploaded by internal hosts follows a long-tailed distribution (see Figures 3(a) and 3(d)). To provide a comprehensive evaluation, for each day we selected seven *representative* hosts, each characterized by an amount of uploaded bytes corresponding to a specific quantile in the daily uploaded bytes distribution. We consider the following quantiles: 0.01, 0.05, 0.25, 0.50, 0.75, 0.95, 0.99. This choice allows us to evaluate the effectiveness of the proposed approach for *low uploaders*, *mid uploaders* and *big uploaders*.

After the selection, each representative host is chosen as source of artificially injected exfiltrations. To evaluate the sensitivity of the proposed approach, for each day and for each representative host we injected data exfiltrations of different sizes: 50 MB, 100 MB, 200 MB, 500 MB, 1 GB, 2 GB, 5 GB and 10 GB. These experiments allowed us to simulate the full spectrum of possible data exfiltrations, from *low-and-slow* to *burst*

Exfiltration size	$r_{0.01}$	$r_{0.05}$	$r_{0.25}$	$r_{0.5}$	$r_{0.75}$	$r_{0.95}$	$r_{0.99}$
50 MB	99.29	99.29	97.86	85.00	43.57	35.00	32.86
100 MB	99.29	98.57	97.86	82.86	42.14	37.14	33.57
200 MB	99.29	100.00	97.86	87.14	50.71	37.86	34.29
500 MB	99.29	100.00	99.29	89.29	57.14	37.86	31.43
1 GB	99.29	100.00	99.29	92.86	67.86	38.57	30.00
2 GB	99.29	100.00	99.29	96.43	76.43	41.43	29.29
5 GB	99.29	100.00	99.29	97.14	84.29	43.57	32.86
10 GB	99.29	100.00	99.29	98.57	90.71	52.86	35.71

Table 8: Percentage of instances in which the position by score computed with our framework is higher than the position obtained with common-ranking approach.

Exfiltration size	$r_{0.01}$	$r_{0.05}$	$r_{0.25}$	$r_{0.5}$	$r_{0.75}$	$r_{0.95}$	$r_{0.99}$
50 MB	42.50	43.50	45.00	117.50	202.50	225.00	50.00
100 MB	23.00	23.50	25.00	64.00	125.50	149.00	49.50
200 MB	11.00	11.00	12.00	37.50	72.50	96.00	46.00
500 MB	**5.00**	6.00	6.00	17.00	32.00	48.50	36.50
1 GB	**3.00**	**3.00**	**3.00**	9.50	17.00	31.00	29.00
2 GB	**2.00**	**2.00**	**2.00**	**4.50**	9.50	22.50	20.50
5 GB	**1.00**	**1.00**	**1.00**	**2.00**	**5.00**	10.50	11.50
10 GB	**1.00**	**1.00**	**1.00**	**2.00**	**3.00**	**5.00**	7.00

Table 9: Median position by score obtained with our framework.

Exfiltration size	$r_{0.01}$	$r_{0.05}$	$r_{0.25}$	$r_{0.5}$	$r_{0.75}$	$r_{0.95}$	$r_{0.99}$
50 MB	353.00	353.00	350.00	323.00	255.00	129.50	38.00
100 MB	164.00	164.00	164.00	159.00	142.00	103.00	37.00
200 MB	97.50	97.50	97.50	96.00	92.00	77.00	33.50
500 MB	54.00	54.00	53.50	53.50	53.00	51.00	26.50
1 GB	34.50	34.50	34.00	34.00	33.50	33.00	20.00
2 GB	18.50	18.50	18.50	18.50	18.00	17.00	14.00
5 GB	10.00	10.00	10.00	10.00	10.00	9.50	8.00
10 GB	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**

Table 10: Median position obtained with common-ranking approach.

data leaks. Finally, for each day and for all combinations of the 7 representative hosts and the 8 different kind of data exfiltration, we computed the score for all internal hosts and ranked them accordingly.

The most common approach for detecting data exfiltration is to leverage standard network analysis tools and rank internal hosts with respect to the number of uploaded bytes [51, 52, 53]. In the following, we refer to this method as *common-ranking*. To verify the effectiveness of our approach we also compare against the results of common-ranking.

The comparison is reported in Table 8, where each column refers to a different representative host; that is, $r_{0.01}$ represents the host corresponding to 0.01-quantile, $r_{0.05}$ represents the host corresponding to 0.05-quantile and so on. Each row denotes a different exfiltration size. Each cell represents the number of times in which the ranking based on the proposed approach led to better results (that is, a higher position) with respect to common-ranking. As an example, the value 99.29% contained in the top-left cell means that for 139 out of 140 days, the ranking based on the score value resulted in a higher position than the one obtained by common-ranking. Table 8 shows that the proposed approach leads to better ranking for all exfiltration sizes and for all hosts up to the 0.5-quantile (column $r_{0.5}$). Columns $r_{0.75}$ and $r_{0.95}$ show mixed results, where the proposed approach fares better for exfiltration sizes higher than 200 MB. As expected, column $r_{0.99}$ favors common-ranking, because high-uploaders are always among the top positions in a ranking based on uploaded bytes, independently of possible data exfiltrations.

This is a significant achievement because: (i) our solution performs better for a majority of the internal hosts; (ii) it highlights suspicious behaviors in hosts that would not have been considered otherwise, because a ranking based on uploaded bytes is always dominated by the minority of big uploaders.

Besides comparing the two different rankings, it is important to analyze their results. Since both approaches prioritize hosts that should undergo further time-consuming analysis, injected exfiltrations should be associated to the highest priori-

ties. Median ranks computed through the proposed approach and through common-ranking are presented in Tables 9 and 10, respectively.

Each cell in these table contains the median rank for a combination of a representative host and an exfiltration size, computed over 140 days. As an example, the first cell in Table 9 that contains value 42.50 means that an exfiltration of 50 MB for a host corresponding to the 0.01-quantile of daily uploaded bytes has a position higher or equal than 42.50 for at least half of the days. On the other hand, Table 10 shows that the same host for the same exfiltration size has a median rank of 353.

Since both the ranking approaches support security analysts in focusing on the most suspicious hosts, their results are effective only if hosts subject to data exfiltrations are ranked within the top-k positions, where k depends on the amount of machines that a security analyst can analyze and is constrained by available time and resources. In this evaluation, we consider two different scenarios: the first one represents an environment where resources allow checks on $k = 50$ machines each day (that is, about 0.05% of the hosts in the considered network); the second scenario represents a more resource-constrained scenario in which an analyst can analyze only $k = 5$ machines per day. In both cases, we consider as *useful* all positions included in the top-k. In Tables 9 and 10, useful results for the first scenario are highlighted in gray, while useful results for the second scenario are written in bold.

In the first scenario we can observe that our proposal generates useful results for all exfiltration sizes (including those smaller than 1 GB) for representative hosts up to 0.25-quantile and for $r_{0.99}$. For representative hosts $r_{0.5}$, $r_{0.75}$ and $r_{0.95}$, the proposed approach detects exfiltrations of at least 200 MB, 500 MB and 500 MB, respectively. On the other hand, common-ranking is never able to detect exfiltrations smaller than 1 GB for any representative host up to 0.95-quantile.

In the second scenario, the proposed approach clearly outperforms common-ranking for all representative hosts up to the 0.95-quantile. In the considered network environment, a daily upload of 10 GB is enough to be ranked as the fifth highest uploader, hence common-ranking is only able to identify exfiltrations of at least 10 GB.

7.4. Summary of results

All experimental results demonstrate that the proposed approach is an improvement with respect to common-ranking. In particular:

- it allows security analysts to quickly identify low-and-slow exfiltrations for the majority of internal hosts;
- for *low-* and *mid-uploaders* the proposed approach yields consistently better results with respect to common-ranking;
- our approach is not biased towards *big-uploaders* that always occupy the highest positions in the common-ranking, independently of exfiltration activities;
- even *low-* and *mid-uploaders* can be classified as the most suspicious hosts.

From the perspective of an attacker, if the network is monitored only through common-ranking, an exfiltration of 500 MB per day would not be detected in most of the cases. On the other hand, if our approach is applied, an exfiltration of 500 MB per day would be always detected. To execute an exfiltration while avoiding detection, an attacker has to carefully choose the host from which the exfiltration takes place. The rational choice for the attacker is to execute low-and-slow exfiltrations from *mid-to-high* uploaders that are above the 0.5-quantile and below the 0.99-quantile. This choice does not depend only on the uploads of a single compromised machine, but also on the network traffic generated by all the other hosts in the network. It is quite unlikely that even an expert attacker is able to gather this information. Hence, we can claim that our proposal makes it extremely difficult to evade detection.

8. Conclusions

We have proposed the first framework that is able to identify and rank suspicious hosts possibly involved in data exfiltrations related to APTs. Our approach gathers and analyzes only network traffic data. We propose a set of features that is specifically tailored to detect possible data exfiltrations, and we define a suspiciousness score for each internal host. The final output is a ranked list of suspicious hosts possibly involved in data exfiltrations and other APT-related activities. The effectiveness of the proposed solution has been proved by implementing a prototype that is deployed on a real large network environment. The proposed approach is able to analyze about 140 millions of flows related to approximately 10,000 internal hosts in about 2 minutes. Experimental results demonstrate the ability of the framework to identify burst and low-and-slow exfiltrations. Our proposal paves the way to novel forms of efficient and automated traffic analyses related to APT activities. Future work includes the integration of correlation systems with respect to other network security assets, such as data flows and alerts coming from intrusion detection systems.

Acknowledgments

We thank the anonymous reviewers for their insightful comments, that guided many modifications in the paper and that helped in clarifying its scope and innovative results.

This research has been funded by the European Commission within the project "EUOF2CEN: European On-line Fraud Cyber Centre and Expert Network". Funded by EU in agreement n. HOME/2014/ISFP/AG/CYBR/7172.

References

[1] R. Brewer, Advanced persistent threats: minimising the damage, Network Security 2014 (4) (2014) 5–9.

[2] I. Jeun, Y. Lee, D. Won, A practical study on advanced persistent threats, in: Computer Applications for Security, Control and System Engineering, Springer, 2012, pp. 144–152.

[3] World most popular data breaches., http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/ (visited in September 2015).

[4] Ponemon Study: The Economic Impact of Advanced Persistent Threats (APTs)., https://securityintelligence.com/media/2014-ponemon-study-economic-impact-advanced-persistent-threats-apts/ (visited in September 2015).

[5] D. E. Denning, An intrusion-detection model, IEEE Transactions on Software Engineering (2) (1987) 222–232.

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

[7] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, B. Stiller, An overview of ip flow-based intrusion detection, Communications Surveys & Tutorials, IEEE 12 (3) (2010) 343–356.

[8] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, in: ACM SIGCOMM Computer Communication Review, Vol. 34, ACM, 2004, pp. 219–230.

[9] G. Thatte, U. Mitra, J. Heidemann, Parametric methods for anomaly detection in aggregate traffic, IEEE/ACM Transactions on Networking (TON) 19 (2) (2011) 512–525.

[10] S. T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks, Communications Surveys & Tutorials, IEEE 15 (4) (2013) 2046–2069.

[11] N. Virvilis, D. Gritzalis, The big four-what we did wrong in advanced persistent threat detection?, in: Availability, Reliability and Security (ARES), 2013 Eighth International Conference on, IEEE, 2013, pp. 248–254.

[12] T. M. Chen, S. Abu-Nimeh, Lessons from stuxnet, Computer 44 (4) (2011) 91–93.

[13] B. Bencsáth, G. Pék, L. Buttyán, M. Félegyházi, Duqu: Analysis, detection, and lessons learned, in: ACM European Workshop on System Security (EuroSec), Vol. 2012, 2012.

[14] B. Bencsáth, G. Pék, L. Buttyán, M. Felegyhazi, The cousins of stuxnet: Duqu, flame, and gauss, Future Internet 4 (4) (2012) 971–1003.

[15] Kaspersky, Labs, "Red October" Diplomatic Cyber Attacks Investigation., https://securelist.com/analysis/publications/36740/red-october-diplomatic-cyber-attacks-investigation/ (visited in September 2015).

[16] P. Giura, W. Wang, A context-based detection framework for advanced persistent threats, in: Cyber Security (CyberSecurity), 2012 International Conference on, IEEE, 2012, pp. 69–74.

[17] J. De Vries, H. Hoogstraaten, J. van den Berg, S. Daskapan, Systems for detecting advanced persistent threats: A development roadmap using intelligent data analysis, in: Cyber Security (CyberSecurity), 2012 International Conference on, IEEE, 2012, pp. 54–61.

[18] S. Torii, M. Morinaga, T. Yoshioka, T. Terada, Y. Unno, Multi-layered defense against advanced persistent threats (apt), FUJITSU Sci. Tech. J 50 (1) (2014) 52–59.

[19] I. Friedberg, F. Skopik, G. Settanni, R. Fiedler, Combating advanced persistent threats: from network event correlation to incident detection, Computers & Security 48 (2015) 35–57.

[20] B. Mukherjee, L. T. Heberlein, K. N. Levitt, Network intrusion detection, Network, IEEE 8 (3) (1994) 26–41.

[21] M. Roesch, et al., Snort: Lightweight intrusion detection for networks., in: LISA, Vol. 99, 1999, pp. 229–238.

[22] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computing Surveys (CSUR) 41 (3) (2009) 15.

[23] C. C. Zou, W. Gong, D. Towsley, Code red worm propagation modeling and analysis, in: Proceedings of the 9th ACM conference on Computer and communications security, ACM, 2002, pp. 138–147.

[24] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, Inside the slammer worm, IEEE Security & Privacy 1 (4) (2003) 33–39.

[25] P. Bhatt, E. Toshiro Yano, P. M. Gustavsson, Towards a framework to detect multi-stage advanced persistent threats attacks, in: Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on, IEEE, 2014, pp. 390–395.

[26] J. R. Johnson, E. Hogan, et al., A graph analytic metric for mitigating advanced persistent threat, in: Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on, IEEE, 2013, pp. 129–133.

[27] T. Sasaki, Towards detecting suspicious insiders by triggering digital data sealing, in: Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on, IEEE, 2011, pp. 637–642.

[28] J. Grier, Detecting data theft using stochastic forensics, digital investigation 8 (2011) S71–S77.

[29] E. Bertino, G. Ghinita, Towards mechanisms for detection and prevention of data exfiltration by insiders: keynote talk paper, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ACM, 2011, pp. 10–19.

[30] Y. Liu, C. Corbett, K. Chiang, R. Archibald, B. Mukherjee, D. Ghosal, Sidd: A framework for detecting sensitive data exfiltration by an insider attack, in: System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on, IEEE, 2009, pp. 1–10.

[31] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, in: Emerging Security Information, Systems and Technologies, 2009., IEEE, 2009, pp. 268–273.

[32] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, W. Lee, Bothunter: Detecting malware infection through ids-driven dialog correlation., in: Usenix Security, Vol. 7, 2007, pp. 1–16.

[33] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, G. Vigna, Your botnet is my botnet: analysis of a botnet takeover, in: Proceedings of the 16th ACM conference on Computer and communications security, ACM, 2009, pp. 635–647.

[34] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al., Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection., in: USENIX Security Symposium, Vol. 5, 2008, pp. 139–154.

[35] A. Azaria, A. Richardson, S. Kraus, V. Subrahmanian, Behavioral analysis of insider threat: A survey and bootstrapped prediction in imbalanced data, Computational Social Systems, IEEE Transactions on 1 (2) (2014) 135–155.

[36] F. L. Greitzer, D. A. Frincke, Combining traditional cyber security audit data with psychosocial data: towards predictive modeling for insider threat mitigation, in: Insider Threats in Cyber Security, Springer, 2010, pp. 85–113.

[37] B. M. Bowen, S. Hershkop, A. D. Keromytis, S. J. Stolfo, Baiting inside attackers using decoy documents, Springer, 2009.

[38] S. Axelsson, The base-rate fallacy and the difficulty of intrusion detection, ACM Transactions on Information and System Security (TISSEC) 3 (3) (2000) 186–205.

[39] N. Villeneuve, J. Bennett, Detecting apt activity with network traffic analysis, Trend Micro Incorporated [pdf] Available at:¡ http://www. trendmicro. com/cloud-content/us/pdfs/securityintelligence/white-papers/wp-detecting-apt-activity-with-network-traffic-analysis. pdf¿[Accessed 31 October 2013].

[40] F. Pierazzi, A. Balboni, A. Guido, M. Marchetti, The network perspective of cloud security, in: Proc. 4th IEEE Symp. Network Cloud Computing and Applications, 2015.

[41] T. T. Soong, Fundamentals of probability and statistics for engineers, John Wiley & Sons, 2004.

[42] J. A. Hartigan, Clustering algorithms.

[43] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: Security and Privacy (SP), 2010 IEEE Symposium on, IEEE, 2010, pp. 305–316.

[44] F. Pierazzi, S. Casolari, M. Colajanni, M. Marchetti, Exploratory security analytics for anomaly detection, Computers & Security 56 (2016) 28 – 49.

[45] N. G. Duffield, F. L. Presti, Multicast inference of packet delay variance at interior network links, in: INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 3, IEEE, 2000, pp. 1351–1360.

[46] C. M. Bishop, Pattern recognition and machine learning, springer, 2006.

[47] C. Canali, M. Colajanni, R. Lancellotti, Hot set identification for social network applications, in: Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International, Vol. 1, IEEE, 2009, pp. 280–285.

[48] M. Andreolini, S. Casolari, M. Colajanni, Models and framework for supporting runtime decisions in web-based systems, ACM Transactions on the Web (TWEB) 2 (3) (2008) 17.

[49] nProbe: An Extensible NetFlow v5/v9/IPFIX Probe for IPv4/v6., http://www.ntop.org/products/netflow/nprobe/ (visited in September 2015).

[50] D. Freedman, P. Diaconis, On the histogram as a density estimator: L 2 theory, Probability theory and related fields 57 (4) (1981) 453–476.

[51] High-Speed Web-based Traffic Analysis and Flow Collection., http://www.ntop.org/products/traffic-analysis/ntop/ (visited in March 2016).

[52] Solarwind Netflow Traffic Analyzer., http://www.solarwinds.com/netflow-traffic-analyzer.aspx/ (visited in March 2016).
[53] AlienVault Unified Security Management., http://www.solarwinds.com/netflow-traffic-analyzer.aspx/ (visited in March 2016).

Mirco Marchetti received his Ph.D. in Information and Communication Technologies (ICT) in 2009. He holds a post-doc position at the Interdepartmental Research Center on Security and Safety (CRIS) of the University of Modena and Reggio Emilia. He is interested in intrusion detection, cloud security and in all aspects of information security. Home page: http://weblab.ing.unimo.it/people/marchetti/

Fabio Pierazzi is a Ph.D. student at the International Doctorate School in Information and Communication Technologies (ICT) of the University of Modena and Reggio Emilia, Italy. He received the Master Degree in computer engineering from the same University in 2013. His research interests include security analytics and cloud architectures. Home page: http://weblab.ing.unimo.it/people/fpierazzi/

Michele Colajanni is full professor in computer engineering at the University of Modena and Reggio Emilia since 2000. He received the Master degree in computer science from the University of Pisa, and the Ph.D. degree in computer engineering from the University of Roma in 1992. He manages the Interdepartmental Research Center on Security and Safety (CRIS), and the Master in "Information Security: Technology and Law". His research interests include security of large scale systems, performance and prediction models, Web and cloud systems. Home page: http://weblab.ing.unimo.it/people/colajanni/

Alessandro Guido is a Ph.D. student at the University of Modena and Reggio Emilia, Italy. He received the Master Degree in computer engineering from the same University in 2012. His research interests include network security and all aspects related to information security. Home page: http://weblab.ing.unimo.it/people/guido/