Scalable architecture for online prioritization of cyber threats

Fabio Pierazzi
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
fabio.pierazzi@unimore.it

Giovanni Apruzzese
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
giovanni.apruzzese@unimore.it

Michele Colajanni
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
michele.colajanni@unimore.it

Alessandro Guido
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
alessandro.guido@unimore.it

Mirco Marchetti
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia
Modena, Italy
mirco.marchetti@unimore.it

Abstract: Detecting advanced attacks is increasingly complex and no single solution can work. Defenders can leverage logs and alarms produced by network and security devices, but big data analytics solutions are necessary to transform huge volumes of raw data into useful information. Existing anomaly detection frameworks either work offline or aim to mark a host as compromised, with high risk of false alarms. We propose a novel online approach that monitors the behavior of each internal host, detects suspicious activities possibly related to advanced attacks, and correlates these anomaly indicators to produce a list of the most likely compromised hosts that is provided to human analysts. Due to the huge number of devices and traffic logs, we make scalability one of our top priorities. Therefore, most computations are independent on the number of hosts and can be naively parallelized. A large set of experiments demonstrate that our proposal can pave the way to novel forms of detection of advanced malware.

Keywords: Autonomous triage; early prioritization; security analytics; scalability.

1. INTRODUCTION

The information systems of modern organizations are subject to a multitude of cyberattacks conceived by a wide range of attackers with different goals, capabilities and motivations. Despite all efforts spent in preventive defenses, the reality is that attacks occur every day and no organization can consider itself secure. This paper shifts the focus from the *prevention* to the *detection* phase.

Existing proposals in academic literature perform detection of specific attacks through heuristics and statistical analysis (e.g., [1, 2, 3, 4]). Most approaches (e.g., [2] [5]) rely on offline post-event analyses. Other online anomaly detectors assume that statistically detectable changes involve huge numbers of hosts (e.g., worm propagation in [6, 7]) or that compromised hosts share similar behavior (e.g., botnet detection in [8, 9, 10, 11]). However, these assumptions are not true anymore in modern human-driven advanced cyberattacks [12], hence existing proposals can be affected by many false positive and false negative alarms.

As no security operator accepts to be annoyed by hundreds of alarms notified at the same priority level, we take a different direction and focus on ranking suspicious hosts instead of detecting compromised hosts. To this purpose, our online analysis begins by monitoring the behavior of individual hosts over time and by identifying suspicious events involving even single or few hosts. These indicators are finally aggregated to produce a ranking of the most suspicious hosts, which are then provided to the security operator in a timely fashion.

Due to the amount of data to be managed online, we propose a scalable design and implementation of our approach. All initial phases before the final aggregation scale linearly with the number of hosts and can be parallelized. The proposed approach is general enough to be adopted with different types of data (such as internal traffic, external traffic, alarms coming from IDS and SIEM), yet the goal of this paper is not to present a complete framework, but rather to propose the idea that the combination of autonomous triage with manual inspection increases the probability of detecting even advanced attacks. For these reasons, we present our approach relying only on network flows of internal corporate traffic, whose effectiveness is shown through experiments applied to networks of more than one thousand hosts. We consider five main attack scenarios, representative of the activities that an attacker will likely perform from a compromised internal host: reconnaissance, data transfer to a dropzone, man in the middle, watering hole through DNS spoofing, and lateral movement through pivoting.

The remaining part of this paper is organized as follows. Section 2 presents related work. Section 3 outlines the main components and functions of the proposed approach. Section 4 describes the analytics core that extracts useful information and builds layer models from raw network data. Section 5 presents five examples of

prioritization algorithms that leverage outputs produced by the analytics core, along with results from real testbed networks. Section 6 concludes the paper with some final remarks and future work.

2. RELATED WORK

Detecting advanced cyberattacks is increasingly difficult as attackers have several ways to penetrate a network and to hide their activities. The huge volume of logs generated by the multitude of servers, firewalls and devices are useful only when they are integrated with security analytics systems for automatic detection and triage. Considering the attacker ability and the difficulty of signaling an infected host without causing false alarms, in the area of security analytics we propose an innovative approach. Instead of signaling an impossible "guaranteed" detection, our system ranks the most suspicious hosts and leaves to the security analyst the task of inspecting a manageable number of hosts. Additional features include online processing for early prioritization and scalability over thousands of hosts as most analyses can be carried out independently for each host. The proposed approach can be applied on alerts and logs derived by IDS [13, 14], SIEM and other security appliances, and can be integrated with external traffic analyses, but in this paper we present a brand-agnostic approach based exclusively on flows of internal network traffic.

We identify three main areas of related works: offline forensics analysis, advanced malware detection, online traffic monitoring.

The large majority of related proposals in literature concern offline analysis for forensics purposes that differ from our online approach. Just to give some representative examples, we can cite [2] on heterogeneous logs analyses, [5] for its original graph-based approach for forensics, *BeeHive* [3] that correlates logs through histogram analysis to identify suspicious activities and corporate policy violations, [15] on forensics for cloud environments, and [16] on mobile forensics. Literature on advanced malware detection focuses on specific attack sequence patterns based on past APT campaigns (e.g., [17, 18] [19, 20, 21]) instead of detecting suspicious activities in each possible phase of an attack. Other more general solutions (e.g., [22, 23]) share our idea of prioritizing suspicious hosts, but they are designed for offline or batch analysis.

The proposals based on online analyses focus on detection of DDoS [24, 25, 26], worm propagation and botnets, where the last two are the most related to our work. In worm propagation detection [7, 27], the internal network is usually modeled as a graph, where huge changes in the overall structure are identified as possible infection propagations. These works differ from our proposal because they focus on a specific threat, and their analyses look for huge changes in traffic volumes and patterns,

whereas we prioritize signals of malicious activities related to behavioral changes of individual hosts. Moreover, our solution is scalable with respect to the number of monitored hosts, while worm propagation analyses depend on the size of the network graph. Botnet detection proposals [8, 9, 10, 11] are based on online scalable solutions for finding hosts that are possibly compromised. However, their underlying assumption is that a large number of hosts are compromised and share a similar network behavior, which is not true in the case of advanced cyberattacks where only few hosts may be compromised and malicious actions are often human-driven. In summary, we can outline the major contributions that differentiate our work with respect to the state of the art:

- *ranking* of suspicious activities instead of specific detection(s);
- *online* analysis instead of offline post-mortem analysis;
- analysis based on *individual hosts* behavior that guarantees parallel analyses and scalability;
- possibility of capturing suspicious actions involving even *few hosts*.

3. FRAMEWORK OVERVIEW

We aim to detect anomalous network activities concerning each host of a corporation, and to use this information to rank the most suspicious hosts. In this section we outline the proposed architecture and design choices for achieving scalability. Figure 1 emphasizes that the input is represented by raw network data gathered by internal probes. Without loss of generality, in this paper we consider just network flows of traffic among internal hosts, which are feasible to collect and analyze for online contexts [28]. These logs are processed through three main steps: *analytics core*, *attack prioritization* and *autonomous triage*. The final output is a list of internal hosts ranked by a risk score representing the likelihood that each host is involved in one or more attacks.

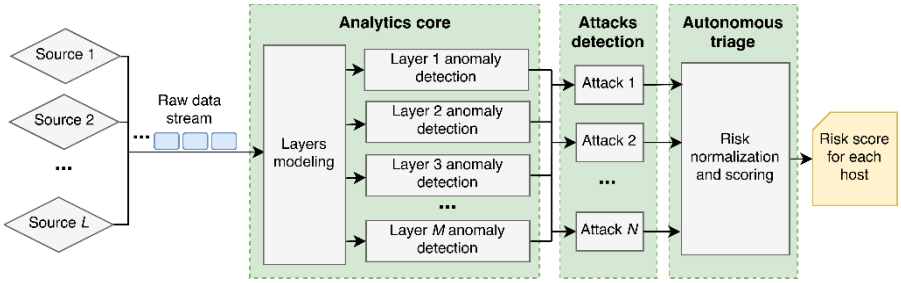

Figure 1: Framework overview.

Starting from raw network data, the *analytics core* builds different *layers* that are graph models in which nodes represent internal hosts and edges represent a metric of interest. Each layer portrays a different perspective of the events occurring in the monitored network. For example, if we consider three layers, then edges may represent the number of packets, the number of bytes, and the average duration of the transmissions between two hosts, respectively. Then, the analytics core applies anomaly detection algorithms on the activities of each internal host within each layer. This fine-grained analysis is motivated by the observation that an attack related to a single host within a large internal network cause very small alterations that are not visible in an aggregated model comprising all layers and all hosts. Similar "global" approaches work well only to identify massive attacks or network-wide anomalies [29, 4].

As a further advantage, since anomaly detection on different layers and hosts can be performed in parallel, the analytics core scales linearly with respect to the number of monitored hosts and layers. In such a way, we can extend and improve an instance of the framework by adding more layers and/or nodes without having to change the information flow and the overall architecture. The algorithms adopted by the analytics core for layers modeling and anomaly detection are presented in Section 4.

The *attack prioritization* module takes as its input the anomalies identified by the analytics core, and correlates them with the goal of detecting different attack scenarios, each one corresponding to activities that an attacker may perform from a compromised internal host. It is also possible to include novel attack detection algorithms with limited computational effort because they can leverage the common fine-grained analyses already performed by the analytics core. The details of the attack prioritization algorithms are discussed in Section 5.

The output of the attack prioritization module is a risk score assigned to each internal host for each considered attack. Attack specific risk scores for all hosts represent the input of the *autonomous triage* module that aids security operators by visualizing the few hosts with higher ranks and the attacks in which they are likely involved.

4. ANALYTICS CORE

This section describes the algorithms used by the analytics core for *layers modeling* and for *anomaly detection* within each layer. The objective is to identify statistical anomalies for each host on all the layers, which will be correlated and ranked by the attack prioritization module. The analytics core is designed for *online* processing and *scalability*.

4.1. LAYERS MODELING

Raw data are collected from the probes as soon as they are produced, and temporarily stored for a time defined by the *current time window* of size Δ. If t denotes the current time, then the layers modeling module maintains all raw data generated between $t - \Delta$ and t. Since literature shows that most network activities are characterized by a daily periodicity [29, 23], it is convenient to set Δ equal to one day. At every *sampling interval* τ, all raw data in the current time window are used to compute the *current representation* of all layers. Since anomalies can be detected only after their appearance in the current representation of a layer, "early" prioritization is influenced by the choice of the parameter τ that is conveniently chosen in the order of few minutes. Lower values cause useless oversampling of data (as an example, Netflow records [30] related to long-lived connections are refreshed every 2 minutes), while higher values introduce detection delays. We use the notation $L_i(t)$ to identify the current representation of the layer i that is built using raw data in the current time window.

As shown in Figure 2, each $L_i(t)$ is modeled as a graph, in which the nodes represent hosts of the internal network, and the edges denote some specific features of network activities occurring between the two hosts. As an example, a layer representing the number of bytes exchanged between internal hosts can be defined as a directed and weighted graph, in which edge direction denotes the direction of data transfer (from source to destination) and the weight represents the amount of transferred bytes.

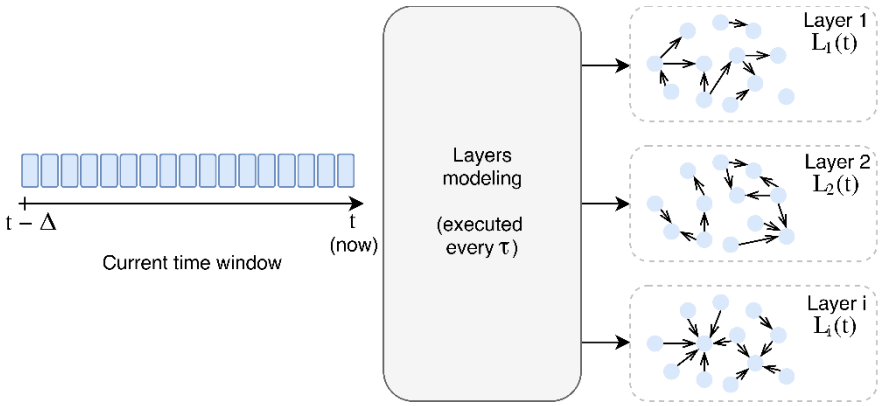

Figure 2: Activities of the layer modeling module.

Table 1 reports the list of considered layers and their descriptions. These characteristics are commonly adopted to identify anomalies in traffic [31]. For example, time series of flows, packets, bytes and ports are used to identify reconnaissance activities [6] and data exfiltration [23]; graphs of internal

communications are adopted for identification of worm propagation [7]; ARP messages can be useful for detecting eavesdropping activities [32].

Table 1. Considered layers.

	Layer	Description
L_1	*Packets*	Directed weighted graph. Nodes are internal hosts and edges connect two nodes that exchange packets using any protocol. Direction is from source to target, and the weight of the edge is the total number of packets transmitted.
L_2	*Bytes*	Directed weighted graph. Nodes are internal hosts and edges connect two nodes that exchange packets using any protocol. Direction is from source to target, and the weight of the edge is the total number of bytes transmitted.
L_3	*Flows*	Directed weighted graph. Nodes are internal hosts and edges connect two nodes that exchange packets using any protocol. Direction is from source to target, and the weight of the edge is the total number of network flows.
L_4	*Ports*	Directed weighted graph. Nodes are internal hosts and edges connect two nodes communicating through TCP or UDP protocols. Direction is from source node to target node, and the weight of the edge is the number of different destination port numbers.
L_5	*Durations*	Directed weighted graph. Nodes are internal hosts and edges connect two nodes that exchange packets using any protocol. Direction is from source to target, and the weight of the edge is the average duration of network flows.
L_6	*Conns*	Directed unweighted graph. Nodes are internal hosts and edges connect two nodes that exchange IP datagrams. Direction is from source to target.
L_7	*Paths*	Bipartite directed graph. Both sets of nodes represent internal hosts. Edges connect each host from the first set, to the hosts of the second set that are reachable by it through a "path" composed of at least 3 hosts.
L_8	*DNS*	Bipartite directed graph. One set of nodes represents hostnames of internal hosts, the other set of nodes represents IP addresses. Edges connect a hostname to the associated IP address in DNS resolutions.
L_9	*ARP*	Bipartite directed graph. One set of nodes represents IP addresses of internal hosts, the other set of nodes represents MAC addresses. Edges connect the IP address and the MAC address that are bound as part of an ARP transaction.

The representations of all layers are passed in input to the processing modules that perform anomaly detection.

4.2. LAYER ANOMALY DETECTION

The goal is to identify hosts that exhibit anomalous behaviors in any of the layers. This step does not depend on the identifiable attacks nor on the feature represented by each layer, hence all layers are subject to the same anomaly detection algorithms, which can be executed in parallel and independently. For each layer, we adopt two complementary detection approaches, as shown in Figure 3: the former identifies quantitative anomalies and state changes; the latter detects novel or uncommon events.

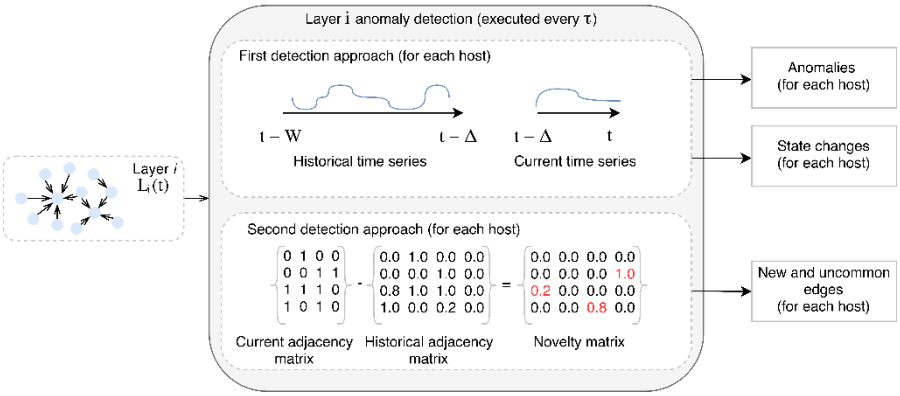

Figure 3: Structure of a layer anomaly detection module.

The former approach processes all current layer representations $L_i(t)$. The goal is to extract scalar values from graphs and to build time series. For each host, the framework computes two scalar values: the weighted in-degree and the weighted out-degree [33] representing the number of incoming and outgoing connections of each host in the current layer, respectively. Since a new $L_i(t)$ is received by the layer anomaly detection module (one at every sampling interval τ), scalar values for consecutive $L_i(t)$ are used to build two current time series representing recent values of in-degree and out-degree for each host. If t denotes the current instant of time, the current time series includes values between $t - \Delta$ and t. Moreover, to perform anomaly detection, it is necessary to build two historical time series including older scalar values between $t - W$ and $t - \Delta$ (excluded), where W represents the size of the historical window. W should be large enough (in the order of few weeks) to have a reliable baseline for the past behavior of each host [28]. Traffic among internal hosts exhibits more stability with respect to traffic among internal and external hosts, characterized by higher variability and consequent difficulties to achieve stable baseline models [29]. Anomaly detection is performed on each current time series through the online and adaptive detection algorithm proposed in [28] trained over the period W. This algorithm identifies both point anomalies and state changes [1] that reflect different kinds of relevant deviations between the current and past behaviors of an internal host.

The latter approach (cf. Figure 3) identifies new edges that never appeared in the historical window. For example, these edges may represent novel persistent connections of an attacker trying to perform lateral movement. For each $L_i(t)$, the detection algorithm computes its *current adjacency matrix* [33], which is a mathematical representation of the edges in $L_i(t)$, whose rows and columns

represent the internal hosts: the matrix element (j, k) is set to 1 if $L_i(t)$ has an edge from host j to k, to 0 otherwise.

Older versions of the current adjacency matrix, built on previous $L_i(t)$ belonging to the historical time window W, are used to compute the *historical adjacency matrix*. Its values are rational numbers between 0 and 1. In particular, (j, k) denotes the frequency of occurrence of the edge from j to k in the older instances of $L_i(t)$. For example, the value (j, k) is set to 1 if all older instances of $L_i(t)$ layer contain an edge from j to k; if one fifth of older $L_i(t)$ layers include an edge from j to k, then the value (j, k) is set to 0.2. The historical time window is updated every Δ.

At every sampling interval τ, the detection algorithm subtracts the historical adjacency matrix to the current adjacency matrix. The result, defined as *novelty matrix*, allows an immediate identification of new or uncommon edges that are present in $L_i(t)$, but never or seldom appeared in the historical time window. Uncommon edges having a low value in the historical adjacency matrix will result in values that are close to 1 in the novelty matrix; common edges with high values in the historical adjacency matrix will result in values close to 0 in the novelty matrix. The layer anomaly detection algorithm can sum the values included in each row of the novelty matrix to evaluate the "novelty" of all the edges starting from the corresponding host. Similarly, the "novelty" of all edges that end in any internal host is computed by summing the values on the corresponding column of the novelty matrix.

Anomalies, state-changes and novel edges detected by the analytics core are then used by the algorithms that evidence malicious activities and prioritize them.

5. ATTACK PRIORITIZATION AND RANKING

The main goal is to prioritize signals of malicious activities that may be part of an advanced attack. To this purpose, we correlate the anomalies, state-changes and novel edges detected by the analytics core.

5.1. EXPERIMENTAL TESTBED

There are several possible indicators associated with malicious activities. In this paper, we consider: reconnaissance (R), data transfer to a dropzone (DTD), Man in the Middle (MITM), watering hole through DNS spoofing (WH), lateral movement through pivoting (LM). Table 2 indicates which *layer models* are included in the analysis of each attack scenario. The presence of multiple layers increases confidence that a suspicious activity is actually occurring. The attack prioritization module evaluates a *risk score* for each internal host by combining the anomalies,

state-changes and novel edges detected by the analytics core. We refer to the following notations:

- $\boldsymbol{A_{L_i}^{in}}$ (resp. $\boldsymbol{A_{L_i}^{out}}$) denotes the intensity of the biggest point anomaly in the incoming (resp. outgoing) time series related to layer L_i of an internal host during the observed window [3]. For example, a burst in the outgoing bytes.
- $\boldsymbol{C_{L_i}^{in}}$ (resp. $\boldsymbol{C_{L_i}^{out}}$) denotes the intensity of possible state-changes in the incoming (resp. outgoing) time series related to layer L_i of an internal host. For example, a state-change is detected if the average number of packets in the current window doubles for a long period (hence, it is not only a point anomaly [4]).
- $\boldsymbol{N_{L_i}^{in}}$ (resp. $\boldsymbol{N_{L_i}^{out}}$) denotes the number of new incoming (resp. outgoing) edges of an internal host in the graph of layer L_i. For example, it can be used to detect the number of newly contacted hosts in the current time window.

All formulas and scores in this section are computed for each host.

Table 2. Layers used to prioritize different types of attacker activities.

	L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8	L_9
Attack	**Packets**	**Bytes**	**Flows**	**Ports**	**Durations**	**Conns**	**Paths**	**DNS**	**ARP**
Reconnaissance			x	x	x	x			
Data transfer	x	x			x	x			
MITM	x	x	x			x			x
Watering hole					x	x		x	
Lateral movement					x		x		

In the experiments, we consider an internal network consisting of more than 1,000 hosts composed of about 800 clients and 200 servers. The client machines have heterogeneous operating systems including several versions of Mac OS, Linux, and Windows. The server machines host mainly websites and DBMS, but also high performance computations, code versioning and NAS storage. We place monitoring probes in the main 1Gbit switches of the network. Our algorithms are executed on a cluster of eight blades, each having an Intel Xeon 2.6GHz CPU and 16GB of RAM. Network flows are sampled every five minutes.

To evaluate scalability, we consider three scenarios consisting of 96, 287 and 1,012 hosts, respectively. One cluster node is sufficient for computations related to 96 and 287 hosts, while four nodes are necessary for the scenario with 1,012 hosts. This scalability is achieved because all computations of the analytics core are performed independently for each host and for each layer. Operations of the attack prioritization module do not scale linearly, but their computational cost is negligible with respect to the anomaly detection algorithms of the analytics core. We present the details about prioritization of the five attack scenarios, and how risk scores are shown to the security operators.

5.2. PRIORITIZATION OF SUSPICIOUS ACTIVITIES

For each scenario, we inject multiple attacks in some hosts of the network, we apply our analytics and evaluate a *risk score* for each host.

5.2.1 Reconnaissance in internal network

An attacker having control of an internal host likely scans neighbor hosts looking for (known or zero-day) vulnerabilities [6, 26]. We define the risk score R for reconnaissance as follows:

$$R = \frac{A^{out}_{Flows} + A^{out}_{Ports} + N^{out}_{Conns}}{1 + A^{out}_{Durations}}$$

where a higher value of R denotes a higher likelihood that an internal host is performing a scan. Intuitively, when an internal host performs a reconnaissance activity, the average duration of its connections decreases (due to many volatile communications) while the numbers of flows, ports and contacted hosts increase. To evaluate the risk score for this attack, we carry out reconnaissance activities from 10 hosts by varying the scan intensity in terms of number of scanned hosts and ports, as described in Table 3.

Table 3. Reconnaissance attacks injected in the internal network from 10 hosts.

Attack	***#ports scanned***	***#hosts scanned***
horizontal scan	1 single port	from 50 to 1,000 distinct hosts
vertical scan	from 50 to 1,000 distinct ports	1 single host
block scan	from 50 to 1,000 distinct ports	from 50 to 1,000 distinct hosts

Since our approach produces a ranking, we evaluate how many times an internal host performing the attack is prioritized within the top-K hosts. Table 4 reports the results of multiple experiments executed over several weeks, where each row represents the percentage of times an internal host performing the attack has been ranked within the top-K. Each column corresponds to horizontal, vertical, or block scan experiments as described in Table 3.

Table 4. Percentage of times a host performing a reconnaissance is ranked within the top-K.

In top-K	***Horizontal scan***	***Vertical scan***	***Block scan***
in top-5	94.3%	92.4%	99.2%
in top-10	99.5%	99.1%	99.7%
in top-25	100%	99.7%	100%
in top-50	100%	100%	100%

Table 4 shows that in more than 99% of the cases a host performing a reconnaissance activity is ranked within the top-10. Horizontal scans are easier to detect because they span over multiple hosts, whereas vertical scans are harder because it is more common for clients to contact servers on multiple ports if they offer more than one service. As expected, block scans have higher rankings, because they span both over multiple ports and multiple hosts.

Figure 4 reports an example of the traffic time series used to compute the risk score *R*, extracted from the layers *Ports*, *Flows*, *Conns*, *Durations*. The X-axis represents time, and the Y-axis reports the value of different metrics. Small arrows highlight significant anomalies: a horizontal scan around 12:48 and two vertical scans around 17:55 and 22:20.

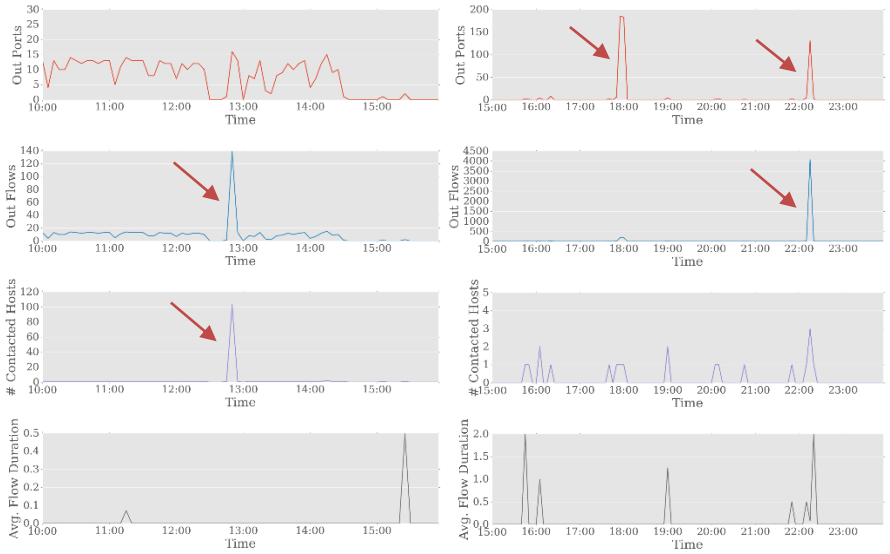

Figure 4. Time series of an internal host performing horizontal and vertical scans.

5.2.2 Data transfer to dropzone before exfiltration

Attackers often move data to be exfiltrated towards an internal *dropzone* [23, 12], used as intermediate point from which the exfiltration is easier to perform. These activities can be detected through the risk score DTD defined as follows:

$$DTD = \frac{A^{out}_{Durations} + A^{out}_{Bytes} + A^{out}_{Packets}}{1 + N^{out}_{Conns}}$$

where a higher value of DTD suggests that an internal host is likely transferring data to an internal dropzone. In the numerator, we consider point anomalies instead of

state changes because the higher bandwidth of internal networks – typically in the order of *Gbps* – allows for short transfer times. In the denominator, we consider N_{Conns}^{out} to rule out legitimate intensified network activity, such as p2p protocols.
We perform several experiments in which we simulate DTD attacks of increasing transfer sizes from 10MB-50MB to 100MB-1GB. We use five controlled hosts as possible attackers and we transfer data to a Web server of the organization as an emulated dropzone. Table 5 reports the percentage of times a host performing a DTD is ranked within the top-K: for small amounts of data (10-50MB), in about 92% of the cases the hosts are ranked within the top-10. This is the most challenging scenario for detection because clients may use multiple hosts/devices for backup, hence it is tough to identify anomalous transfers unless we integrate anomaly detection with white/black lists of internal hosts (where storage is or is not allowed), but similar integrations are out of the scope of this paper. The most compelling result is that in 99% of the cases, the 50-100MB internal transfers are ranked within the top-10.

Table 5. Percentage of times a host performing a DTD is ranked within the top-K.

In top-K	***10-50MB***	***50-100MB***	***100MB-1GB***
in top-5	87.5%	95.4%	99.7%
in top-10	91.7%	99.1%	100%
in top-25	95.6%	99.8%	100%
in top-50	99.5%	100%	100%

As an example, Figure 5 reports time series that show evolution of several layers referring to an internal host used for injecting data transfers to dropzone. The X-axis represents time, and the Y-axis reports the different metrics. Two arrows highlight peaks of about 100MB and 1GB, respectively – corresponding to the injected data exfiltration. We also observe an increment of the average flow duration in correspondence of the two data transfers, whereas other statistics (e.g., number of contacted hosts) remain stable.

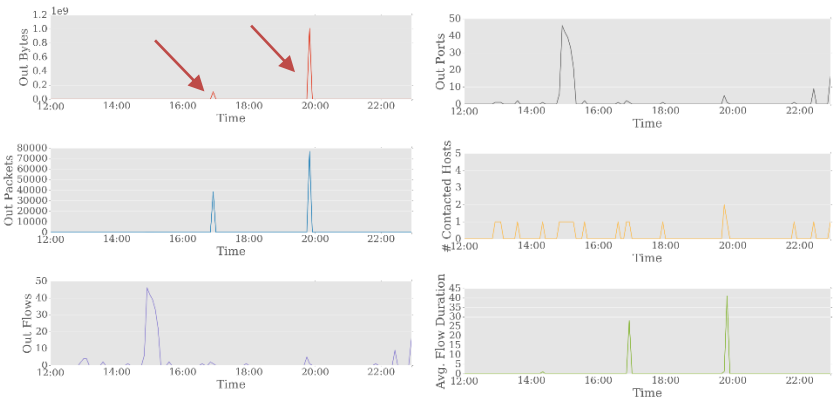

Figure 5. Time series of an internal host in which two DTDs of 100MB and 1GB are injected.

5.2.3 MITM: Man in the Middle attack

Man in the Middle (MITM) attacks are important to perform advanced reconnaissance or to steal credentials, because an attacker can eavesdrop communications of hosts within the same subnet. Here, we consider one of the most subtle forms of MITM performed through ARP spoofing [34]. In this scenario, an attacker sends fake correspondences between IP and MAC addresses with the goal of acting as "hidden" proxy between a victim and the gateway of its subnet. Netflows record no explicit communication between the eavesdropper and the victim, but our experiments evidence that once a host becomes victim of MITM, then all packets sent and received by the victim pass twice through the switch. This attack can be captured by the state-change detection algorithm of the analytics core. In order to prioritize possible victims of MITM we define the following risk score:

$$MITM = \left(\frac{C^{in}_{Bytes} + C^{out}_{Bytes} + C^{in}_{Pkts} + C^{out}_{Pkts}}{1 + C^{in}_{Flows} + C^{out}_{Flows} + N^{in}_{Conns} + N^{out}_{Conns}} \right) * N^{out}_{ARP}$$

In the numerator we consider state-changes instead of point anomalies because MITM is usually an activity that lasts for some time to get useful information. The parameter N^{out}_{ARP} is a multiplicative factor because if N^{out}_{ARP} =0 there is no new correspondence in the ARP layer (see Section 4). In the denominator, we include state-changes and novel edges in *Flows* and *Conns* layers, because they must remain approximately stable with respect to a past window even if MITM is occurring.

Experimental results are achieved through controlled Man in the Middle attacks of varying durations where we use one host as the eavesdropper and other 10 hosts as victims. From Table 6, we can observe that in more than 95% of the cases, even MITM lasting for just 15-30 minutes are prioritized in the top-10; if an attack lasts

for at least 1 hour, the victim hosts are ranked within the top-5 in more than 98% of the cases.

Table 6. Percentage of times a host victim of a MITM is ranked within the top-K.

In top-K	***15-30min***	***1-2hr***	***12-24hr***	***24-72hr***
in top-5	89.8%	98.2%	99.4%	99.8%
in top-10	95.4%	99.1%	99.8%	100%
in top-25	99.0%	99.8%	100%	100%
in top-50	99.7%	100%	100%	100%

As a motivation, in Figures 6 we report the time series of a host related to *Packets* and *Bytes* layers, where the Y-axis denotes the different metrics, and the X-axis reports time. The plots report two days separated by a vertical dashed line. When the MITM is occurring on the second day, it is possible to observe that the number of packets and bytes increases significantly.

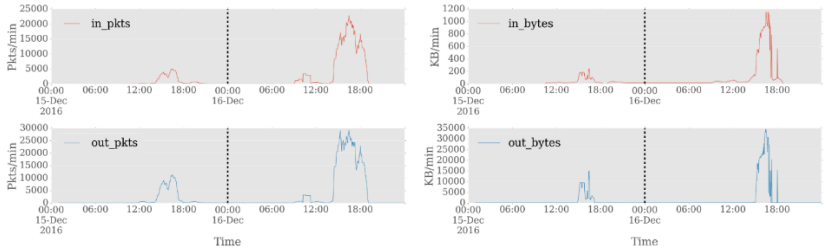

Figure 6. Comparison showing changes in Packets and Bytes layers when MITM occurs.

5.2.4 Watering hole through DNS spoofing

Watering hole is a technique used by attackers to increase their coverage and persistence by infecting multiple hosts of an organization simultaneously. We consider a particular type of *watering hole* attack performed through DNS spoofing [35], where the attacker spoofs DNS responses to redirect victims to a compromised sever. To prioritize internal hosts that may correspond to watering holes, we define the risk score WH as follows:

$$WH = \left(N^{in}_{Conns} + C^{in}_{Conns} + C^{in}_{Durations}\right) * N^{out}_{DNS}$$

where a high value of WH represents a higher likelihood that a host is performing a watering hole through DNS spoofing. Intuitively, this can be prioritized when a host has many new incoming connections, its IP corresponds to a new DNS resolution, and has a state-change in the number and duration of incoming connections. We

observe that N_{DNS}^{out} is a multiplicative factor, because $N_{DNS}^{out} = 0$ implies that no DNS spoofing occurred.

To evaluate the risk score WH, we use five internal clients as "spoofers" of three internal Web servers offering different services: *Server 1 (small)*, *Server 2 (medium)* and *Server 3 (large)* that are used by an average number of clients per hour of about 10, 50 and 250, respectively. We perform a DNS spoofing at times 10am, 2pm, 4pm, 6pm. Table 7 reports the percentage of times a "watering hole" host (that was redirecting traffic to itself through DNS spoofing) has been ranked within the top-K hosts in the different scenarios. This table shows that for Server 1 (having small activity) the spoofer is prioritized in the top-10 in more than 96% of the cases, while this percentage is even higher for servers with high number of clients where the intensity of the redirect is more evident.

Table 7. Percentage of times a host performing "watering hole" is ranked within the top-K.

In top-K	***Server 1 (small)***	***Server 2 (medium)***	***Server 3 (large)***
in top-5	94.7%	98.9%	99.9%
in top-10	96.2%	99.8%	100%
in top-25	99.5%	100%	100%
in top-50	99.8%	100%	100%

As an example, Figure 7 reports a bipartite graph representation of the *Conns* layer over different days: on day 6, the dotted circle highlights a host that started performing a watering hole attack through DNS spoofing. We can observe an increase in the number of the incoming communications.

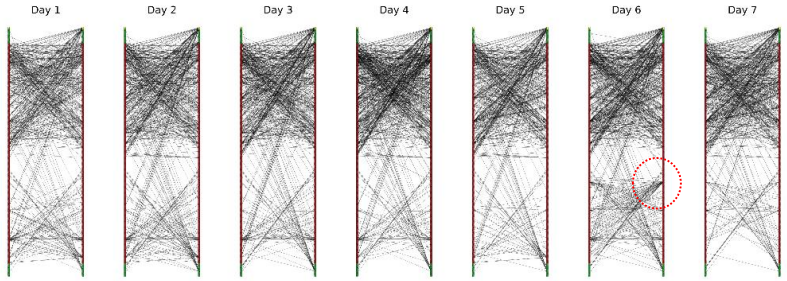

Figure 7. Bipartite communications graph derived from Conns layer over 7 different days.

5.2.5 Lateral movement through pivoting

To get closer to his target, an attacker tends to compromise several internal hosts with higher privileges or to access to the most internal parts of the corporate network.

This activity is called *lateral movement* [12]. Figure 8 reports an example through a common technique named *pivoting* [36], where an attacker creates a tunnel of communications among multiple hosts (namely, *pivoters*) to access a LAN that cannot be reached directly from the external.

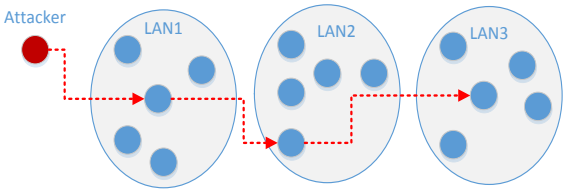

Figure 8. Example of lateral movement through pivoting.

To prioritize lateral movements, we define the risk score LM as follows:

$$LM = (N^{in}_{Paths} + N^{out}_{Paths} + C^{in}_{Duration} + C^{out}_{Duration})$$

where N^{in}_{Paths} and N^{out}_{Paths} take into account new paths in the communications graph (see Section 4), while $C^{in}_{Duration}$ and $C^{out}_{Duration}$ check whether an increment in the average duration of the flows has occurred. (We recall that a *pivoting* tunnel has to last for some time [36]). We perform several experiments involving up to 10 controlled clients as intermediate pivoter hosts in the lateral movement (see Figure 8). Table 8 reports the percentages of times a pivoter host is ranked within the top-K risky nodes. The different columns correspond to different *lengths* of the pivoting tunnel. For example, Figure 8 presents a tunnel of length 2 with two pivoter hosts.

Table 8. Percentage of times a host performing LM is ranked within the top-K.

In top-K	***1 pivoter***	***3-5 pivoters***	***8-10 pivoters***
in top-5	96.2%	99.7%	99.9%
in top-10	97.9%	99.9%	100%
in top-25	99.1%	100%	100%
in top-50	99.8%	100%	100%

5.3. AUTONOMOUS TRIAGE TO SUPPORT SECURITY ANALYSTS

We present how results can be combined to produce an overall ranking useful for security analysts to focus on few suspicious hosts.

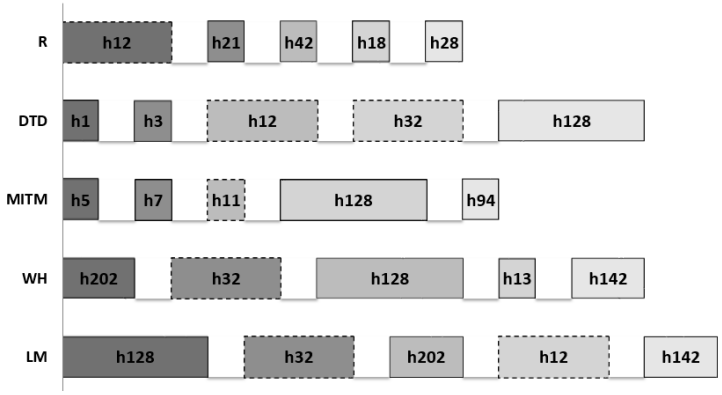

Figure 9. Online autonomous triage of internal hosts for different attack scenarios.

Figure 9 reports the overall rankings, with each line corresponding to a different attack: R for reconnaissance, DTD for data transfer to dropzone, MITM for Man in the Middle, WH for watering hole through DNS spoofing, LM for lateral movement through pivoting. On the leftmost (resp. rightmost) side, there is the host with higher (resp. lower) risk score on that line. All hosts are represented as rectangles, where the size is proportional to the number of top-K rankings in which a host appears. For example, host *h202* appears in two top-five rankings (first for WH, and third for LM), hence its rectangle has a double size. As ranking operations are evaluated online, rectangles with a dashed outline denote hosts that recently entered a top-five ranking. A similar visualization supports security analysts in monitoring several cyber threats occurring in the core of large networks. The number of top-K hosts to show can be chosen adaptively depending on the size of the organization and the amount of human resources. As a final remark, it is important to observe that our proposal makes it hard for an attacker to elude ranking, because we monitor changes in activity of each *individual host* with respect to its history, and we produce an overall ranking considering all hosts of the internal network. Hence, an attacker would require a *complete* view of all hosts behaviors/history to evade prioritization successfully.

6. CONCLUSIONS

In this paper, we propose a novel approach based on ranking and prioritization instead of "guaranteed" detection. We consider an innovative perspective in which we start by analyzing *individual host* behaviors, and then post-correlate outputs to compute various indicators corresponding to different attacker activities. A prioritized list of likely compromised hosts is passed to human analysts, who can focus their attention only on the most suspicious hosts and activities. Experimental

evaluations and use-case examples in real-world internal networks of more than 1,000 hosts demonstrate the feasibility and scalability of the proposed approach for online autonomous triage of different attack scenarios. Future works include the integration of attack indicators from external traffic, such as text analysis and statistical characterization of DNS queries to identify possible C&Cs.

7. REFERENCES

[1] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, 2009.

[2] E. S. Pilli, R. C. Joshi and R. Niyogi, "Network forensic frameworks: Survey and research challenges," *Elsevier Digital Investigation*, 2010.

[3] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels and E. Kirda, "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," in *Proceedings of the 29th ACM Annual Computer Security Applications Conference*, 2013.

[4] M. Andreolini, M. Colajanni and M. Marchetti, "A collaborative framework for intrusion detection in mobile networks," *Elsevier Information Sciences*, 2015.

[5] W. Wang and T. E. Daniels, "A graph based approach toward network forensics analysis," *ACM Transactions on Information and System Security (TISSEC)*, 2008.

[6] S. J. Stolfo, "Worm and attack early warning: piercing stealthy reconnaissance," *IEEE security & privacy*, 2004.

[7] M. P. Collins and M. K. Reiter, "Hit-list worm detection and bot identification in large networks using protocol graphs," in *International Workshop on Recent Advances in Intrusion Detection*, 2007.

[8] M. Bailey, E. Cooke, F. Jahanian, Y. Xu and M. Karir, "A survey of botnet and botnet detection," in *Conference For Homeland Security, CATCH'09, Cybersecurity Applications \& Technology*, 2009.

[9] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proceedings of the ACM 28th Annual Computer Security Applications Conference*, 2012.

[10] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong and W. Lee, "BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation," in *Usenix Security*, 2007.

[11] G. Gu, R. Perdisci, J. Zhang and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection," in *USENIX Security Symposium*, 2008.

[12] R. Brewer, "Advanced persistent threats: minimising the damage," *Network Security*, pp. 5-9, 2014.

[13] M. Colajanni, D. Gozzi and M. Marchetti, "Enhancing interoperability and stateful analysis of cooperative network intrusion detection systems," in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2007.

[14] M. Marchetti, M. Colajanni and F. Manganiello, "Framework and Models for Multistep Attack Detection," *International Journal on Security and Its Applications*, 2011.

[15] K. Ruan, J. Carthy and T. Kechadi, "Survey on cloud forensics and critical criteria for cloud forensic capability: A preliminary analysis," in *Proceedings of the Conference on Digital Forensics, Security and Law*, 2011.

[16] J. Grover, "Android forensics: Automated data collection and reporting from a mobile device," *Elsevier Digital Investigation*, 2013.

[17] P. Bhatt, E. Toshiro Yano and P. M. Gustavsson, "Towards a Framework to Detect Multi-stage Advanced Persistent Threats Attacks," in *IEEE International Symposium on Service Oriented System Engineering (SOSE)*, 2014.

[18] E. M. Hutchins, M. J. Cloppert and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," in *Proceedings of the 6th International Ionference on i-Warfare and Security*, 2011.

[19] R. Brewer, "Advanced persistent threats: minimising the damage," *Network Security*, pp. 5-9, 2014.

[20] I. Jeun, Y. Lee and D. Won, "A practical study on advanced persistent threats," *Computer Applications for Security, Control and System Engineering*, pp. 144-152, 2012.

[21] N. Virvilis and D. Gritzalis, "The big four-what we did wrong in advanced persistent threat detection?," in *IEEE International Conference on Availability, Reliability and Security (ARES)*, 2013.

[22] M. Marchetti, F. Pierazzi, A. Guido and M. Colajanni, "Countering Advanced Persistent Threats through security intelligence and big data analytics," in *Cyber Conflict (CyCon), 2016 8th International Conference on*, Tallin, Estonia, 2016.

[23] M. Marchetti, F. Pierazzi, M. Colajanni and A. Guido, "Analysis of high volumes of network traffic for Advanced Persistent Threat detection," *Elsevier Computer Networks*, 2016.

[24] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti and M. Christensen, "Portvis: a tool for port-based detection of security events," in *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, 2004.

[25] V. Sekar, N. G. Duffield, O. Spatscheck, J. E. van der Merwe and H. Zhang, "LADS: Large-scale Automated DDoS Detection System," in *USENIX Annual Technical Conference, General Track*, 2006.

[26] M. H. Bhuyan, D. Bhattacharyya and J. K. Kalita, "Surveying port scans and their detection methodologies," *The Computer Journal*, 2011.

[27] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip and D. Zerkle, "GrIDS-a graph based intrusion detection system for large networks," in *Proceedings of the 19th national information systems security conference*, 1996.

[28] S. Casolari, S. Tosi and F. Lo Presti, "An adaptive model for online detection of relevant state changes in Internet-based systems," *Performance Evaluation*, pp. 206-226, 2012.

[29] F. Pierazzi, S. Casolari, M. Colajanni and M. Marchetti, "Exploratory security analytics for anomaly detection," *Computers & Security*, pp. 28-49, 2016.

[30] "nProbe," [Online]. Available: http://www.ntop.org/products/netflow/nprobe/.

[31] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras and B. Stiller, "An overview of IP flow-based intrusion detection," *IEEE communications surveys & tutorials*, 2010.

[32] P. Goyal, S. Batra and A. Singh, "A literature review of security attack in mobile ad-hoc networks," *International Journal of Computer Applications*, 2010.

[33] M. Newman, Networks An introduction, Oxford University Press, 2010.

[34] V. Ramachandran and S. Nandi, "Detecting ARP spoofing: An active technique," in *International Conference on Information Systems Security*, 2005.

[35] U. Steinhoff, A. Wiesmaier and R. Araujo, "The State of the Art in DNS Spoofing," in *Proc. 4th Intl. Conf. Applied Cryptography and Network Security (ACNS)*.

[36] "Pivoting," 2016. [Online]. Available: https://offensive-security.com/metasploit-unleashed/pivoting/.