OSTINATO: Cross-host Attack Correlation Through Attack Activity Similarity Detection

Sutanu Kumar Ghosh(✉), Kiavash Satvat, Rigel Gjomemo, and V. N. Venkatakrishnan

University of Illinois Chicago, Chicago, IL, USA
{sghosh34,ksatva2,rgjome1,venkat}@uic.edu

Abstract. Modern attacks against enterprises often have multiple targets inside the enterprise network. Due to the large size of these networks and increasingly stealthy attacks, attacker activities spanning multiple hosts are extremely difficult to correlate during a threat-hunting effort. In this paper, we present a method for an efficient cross-host attack correlation across multiple hosts. Unlike previous works, our approach does not require lateral movement detection techniques or host-level modifications. Instead, our approach relies on an observation that attackers have a few strategic mission objectives on every host that they infiltrate, and there exist only a handful of techniques for achieving those objectives. The central idea behind our approach involves comparing (OS agnostic) activities on different hosts and correlating the hosts that display the use of similar tactics, techniques, and procedures. We implement our approach in a tool called OSTINATO and successfully evaluate it in threat hunting scenarios involving DARPA-led red team engagements spanning 500 hosts and in another multi-host attack scenario. OSTINATO successfully detected 21 additional compromised hosts, which the underlying host-based detection system overlooked in activities spanning multiple days of the attack campaign. Additionally, OSTINATO successfully reduced alarms generated from the underlying detection system by more than 90%, thus helping to mitigate the threat alert fatigue problem.

1 Introduction

Modern advanced persistent threats (APT) often spread stealthily across multiple hosts in their target enterprises. Detecting APT activities across multiple hosts inside such networks is very challenging. Approaches that deal with this challenge are often *network-based* [22,38,50]. They focus on finding a strong presence of attack artifacts in network data (e.g., DDOS, botnets). However, modern APTs are increasingly stealthy and usually have a minimal footprint on network logs, and are often characterized as "slow and low". Often, most of their actions occur *inside* hosts, while activities like scanning internal hosts or gaining access to new hosts happen over a long period of time.

To be able to detect suspicious in-host activities, *host-based* solutions are needed. Current *host-based* approaches and Intrusion Detection Systems (IDSes)

V. R. Badarla et al. (Eds.): ICISS 2022, LNCS 13784, pp. 1–22, 2022.
https://doi.org/10.1007/978-3-031-23690-7_1

[27,28,45] rely on audit logs to detect attack activities represented as Indicators of Compromise (IOCs) or Tactics Techniques and Procedures (TTPs). However, they are focused on single-host detection and the alerts they raise are mostly about activities inside single hosts. To be able to deal with multi-host attacks, alerts raised on single hosts must be *correlated* with one another.

One way to correlate alerts from multiple hosts involves understanding and detecting *lateral movement* tactics, techniques, and procedures (TTPs) employed by attackers [4,20,34]. In particular, if a *lateral movement* TTP is detected, the two hosts involved in that TTP can be assumed to be victims of the same campaign. However, because such TTPs may be based on zero-day exploits or because of *threat alert fatigue* in human operators of Security Operation Centers [14,16], they may not be detected and the alerts from multiple hosts may not be connected with one another.

In this paper, we present OSTINATO, a tool for efficient cross-host attack correlation across multiple hosts. OSTINATO's design relies on the key observation that a specific APT group uses a finite (possibly large) set of tools during a campaign. In fact, according to MITRE's ATT&CK page listing the cyber threat groups observed in the wild, a vast majority of those groups employ only a handful of techniques and procedures [1,13].

Based on this observation, we design an approach that compares (OS-agnostic) activities on different hosts and correlates those hosts that display similar suspicious techniques used to achieve similar objectives across those hosts. In particular, if similar tactics appear on two different hosts, then it is likely that the two hosts are victims of the same attack and they are, therefore, correlated.

The main challenge in realizing this approach lies in defining an *activity similarity computation* method that can be applied independently of attack peculiarities and thus be used in a general setting in networks with a large number of hosts. To address this challenge, OSTINATO first models the attacker's techniques and the underlying operational procedures as *tagged provenance graphs*, which represent audit logs as graphs that are tagged with attacker-related procedures. Next, OSTINATO defines a novel approximate graph similarity computation method that can be applied to the set of tagged provenance graphs in a pairwise fashion. The main contributions of the paper are as follows.

Graph- and Similarity-Based Correlation. We propose a novel approximate graph similarity-based alert correlation technique by addressing the (often overlooked) problem of determining when entities (e.g., processes, files, sockets, and its respective information flow) associated with alerts from different hosts are similar during an attack campaign. This is particularly useful for cross-host attack correlation involving hundreds of hosts in an enterprise network.

Threat Hunting Application. The second contribution of this paper is the detection of compromised hosts through the correlation between detected attacker activities and other activities across multiple hosts using graph similarity. In this kind of application, OSTINATO can enhance the threat hunting capabilities of existing Security Information and Event Management (SIEM) systems.

Threat Alert Fatigue Mitigation Application. Threat alert fatigue is a common problem in Security Operation Centers (SOC), where human operators pour over hundreds of thousands of alerts generated by the network- and host-based systems. OSTINATO can boost alert scores related to attack activities that are similar across multiple hosts and thus help reduce alert fatigue.

Along with these contributions, we perform the experimental evaluation (Sect. 4) where we use two different datasets collected from several red team engagements organized by DARPA. In the first dataset, the red teams performed various attacker activities across a network of 500 Windows hosts, resembling modern APTs. In this evaluation, the single host-based detection system either missed attacker activities having small footprints that evade its detection threshold or produced many false positives at lower thresholds. In turn, because of similarities among these activities OSTINATO was able to detect 21 additional hosts compromised by the attacker. We also created and evaluated an extensive dataset of more than 1000 graphs (of different sizes) generated by varying the detection threshold of the underlying IDS in each of the hosts from the same data. In the second dataset, we further evaluated OSTINATO on a different attack scenario involving multiple hosts of different OSes and successfully correlated cross-host attacker activities.

This paper is structured as follows: Sect. 2 provides a high level description of the problem. Section 3 describes the approach and architecture, Sect. 4 contains the evaluation, Sect. 5 describes the related work and Sect. 6 the conclusion.

2 Problem Description

In a multi-host system, one of the primary methods for expanding threat-hunting activities to new hosts relies on detection of *lateral movement* activities. In particular, if lateral movement events are seen on one host – e.g., suspicious traffic to a remote Windows SMB host – SOC operators may decide to escalate the alerts on both those hosts to more scrutiny. However, this strategy relies on *known* lateral movement indicators, and it may not always work if those indicators are missing, incorrectly modeled, if attackers modify their tactics so that they do not match known indicators, or if they move laterally via existing benign network communications (living off the land).

Alert fatigue is another cause for failing to process lateral movement indicators. In modern SOC centers, with thousands of hosts and hundreds of thousands of alerts (the majority of which are false positives), without a (relatively) strong signal about lateral movement or initial compromise on a host, it is counterproductive to escalate alerts to a higher level of scrutiny. Operations in such centers are finely tuned to deal with *alert fatigue*, and almost every system incorporates techniques, filtering mechanisms, and knobs that adjust the signals to forward to human operators [10]. Setting such filters to low values ensures reducing false negatives at the cost of having more false positives. Setting them at higher values reduces false positives but can potentially miss true positives. As a result, legitimate lateral movement indicators may be ignored by analysts.

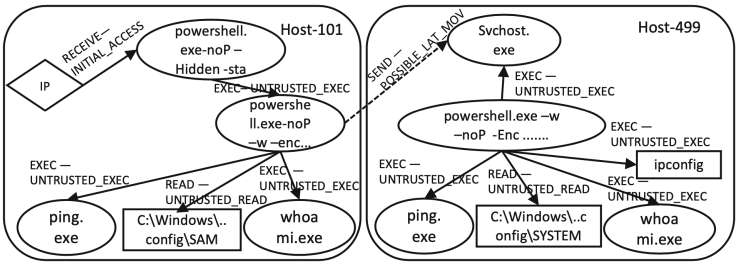

Fig. 1. Example of alarms raised by underlying IDS in multiple hosts in the form of tagged provenance graph

To illustrate the problem, consider the following running example (Fig. 1). An attacker obtains an initial foothold in a host (Host-101) inside a network of several hundred hosts. There (s)he performs several actions on the compromised host using powershell commands. These include pinging other hosts, monitoring running processes, reading sensitive system files, password hashes, and so on. The host-based intrusion detection system (IDS) raises alerts for some of these activities. These alerts are related to events in the audit logs that have some suspicious connotations. However, because they are similar to benign activities, they do not pass the threshold needed to be forwarded to a human operator or, if they do, they may appear together with many other false positive alerts, and thus be missed by human operators. Next, the attacker uses compromised passwords from Host-101 to gain access to another host (Host-499), which is usually accessed remotely by a benign user from Host-101. Because the attacker is mimicking a benign activity, this lateral movement remains undetected. The attacker performs similar actions in the new host including pinging other hosts, monitoring daily activities, sensitive file systems, and password hashes. The IDS running on Host-499 is identical to that on Host-101 and raises similar alerts. However, because the connection between Host-101 and Host 499 is considered benign, alerts are again missed. As a result, the attack is not detected or ignored.

Problem Statement. Our problem statement is as follows: *How can we correlate alerts across hosts without relying on lateral movement detection in a network with hundreds of hosts? How can we obtain an additional suspicious signal related to correlation for SOC operators?* Our key observation to solve this problem is that those attacker activities that are observable in audit logs are a manifestation of the attackers' overall goals (i.e., kill-chain steps) and related techniques [5]. Often, these goals overlap across hosts. For instance, for every host that is compromised, there **must** exist an *initial access* step. Often, initial access is followed by a *discovery* step, where the attacker explores the newly compromised host. To spread to a new host, the attacker **must** perform *lateral movement*. To be able to maintain their presence in the hosts for a long time, the attackers **must** execute some form of *persistence*.

Another *key insight* at the basis of our solution is that the tactics and procedures available to carry out these common goals in multiple hosts during an APT

campaign are not infinite but are limited in number. The attackers must, therefore, execute similar procedures in several hosts. In particular, by the pigeonhole principle, the larger the number of compromised hosts in a network, the more likely it is that similar activities are carried on those hosts. While it is certainly possible that attackers use different procedures for the same goals on different hosts, this would significantly raise the bar of difficulty for the attackers as this would require exploiting several vulnerabilities and increasing the chances of being detected by the underlying IDSes. Several research and survey papers, in fact, confirm the validity of our insight in the wild [3,9,15]. As evidenced by some observations [25], creating novel TTPs often requires significant resources and motivation from attackers.

The main challenge, in correlating cross-host alerts resides in producing a similarity definition for alerts that is general enough to be used across multiple hosts in a network. In particular, because each host can execute processes in many different ways, we must be able to capture the similarity between processes' behaviors inside different hosts. We solve this challenge with OSTINATO, a system that can detect similar behaviors present in the alerts generated by IDS-es, and create additional alerts. OSTINATO's goal is to be used as companion to existing IDS-es and provide an additional signal for attack detection.

3 Approach and Architecture

Threat Model. We assume that the attacker is able to initially compromise a host and, starting from that host, spread to other hosts inside the network, either via relying on vulnerable processes or by using existing tools from the compromised host (e.g., remote desktop services, SSH, etc.). We assume that there is an intrusion detection system in each host that is generating alerts, which may detect part of the attacker's activities. However, these alerts are also buried inside a large number of false positive alerts. Similar to prior research in this area, we also assume that the audit logs data are trustworthy and not modified by the attackers. We also assume that the alerts are derived from existing audit logs systems (ETW, Auditd) and that they contain the system calls generated by the running processes and the process invocations with their command line arguments. We represent the information in these logs and alerts as tagged provenance graphs.

3.1 Tagged Provenance Graphs

Provenance graphs [27,28,35,37,39,44,45,52] are well-known, widely-popular representation of audit logs, where nodes represent system entities (processes, files, registry entries, sockets) labeled by the entity names or paths together with command line arguments (in the case of processes) and directed edges represent system events and system calls (and are labeled by the system call name, e.g., read, write, fork, mmap) that connect those entities.

To represent both the high-level attacker goals and their low-level operational details at the same plane, OSTINATO enhances *provenance graphs* with

tags (additional labels) on the edges of the graph representing semantic level details (attacker goals, tactics names, and others). This enhancement is done by Ostinato based on the graph and its respective alarms generated by the underlying IDS where the edge names are augmented with additional information before storing them in a common database. Examples of such graphs are shown in Fig. 1. The nodes represent different system entities, while edges are labeled by both *system call labels* (`exec`, `remove`) and *suspiciousness labels*, which capture the attacker's goal (`Untrusted Exec`, `Untrusted_Remove`), as a TTP [2] name. This novel enhanced representation, which we call *tagged provenance graphs*, allows us to represent alerts including both system behavior and attacker goals and include these high level details in the search for similar alerts.

Using *tagged provenance graphs*, Ostinato models the process of alert correlation across different hosts as a search for similar tagged provenance graphs representing those alerts. In particular, Ostinato first determines *node similarity* between nodes in different graphs. Next, it uses the edge labels and tags to determine edge similarity, and finally combines the nodes and edge similarity values into an overall similarity score for a pair of tagged provenance graphs.

Ostinato's architecture is shown at the top half of Fig. 2. At the bottom of the figure, we show the hosts of an enterprise network. The IDSes inside each of these hosts produce alerts that are next transformed into tagged provenance graphs and stored in a central database by Ostinato. It serves as a companion to these IDSes and utilizes the respective tagged provenance graphs. Each of these tagged provenance graphs in the central database are processed by the first phase of our approach, *Node Similarity Detection* (Sect. 3.2), which is responsible for grouping different nodes from different graphs into *buckets* containing similar nodes. Next, the *Graph Similarity Detection* step uses these buckets and a set of edge label similarity rules to compute the final similarity value among the tagged provenance graphs (Sects. 3.3, 3.4). Finally, if the similarity value crosses a specific threshold, Ostinato raises an alert. The details of each of these stages along with their challenges are explained in the remainder of this section.

3.2 Identifying Similar Nodes

There is a large body of work dedicated to computing graph similarity and related problems, including graph isomorphism [42], iterative (or structural) methods [29], graph pattern matching [24]. At the foundation of these algorithms, often there is an *assumption that an initial mapping between nodes on different graphs already exists*. Such mapping informs these algorithms on which nodes in one graph are similar or match which nodes in another graph. They often assume that nodes have simple labels (e.g., single letters of the alphabet), which can be trivially used to provide an initial similarity measure between nodes. In our setting, however, nodes in the provenance graphs represent different entities, including processes, files, sockets, etc. An initial mapping that can inform about similar nodes across hosts does not exist. Therefore, one question at the core of our problem is: how can we produce such mapping? When can we claim that, for instance, two processes from two different hosts are the same or similar? Is,

for instance, a *PowerShell* process from one host similar to a *PowerShell* process from another host? To answer this question, we focus on two aspects of nodes in the graphs: *node label content* and *approximate node behavior*.

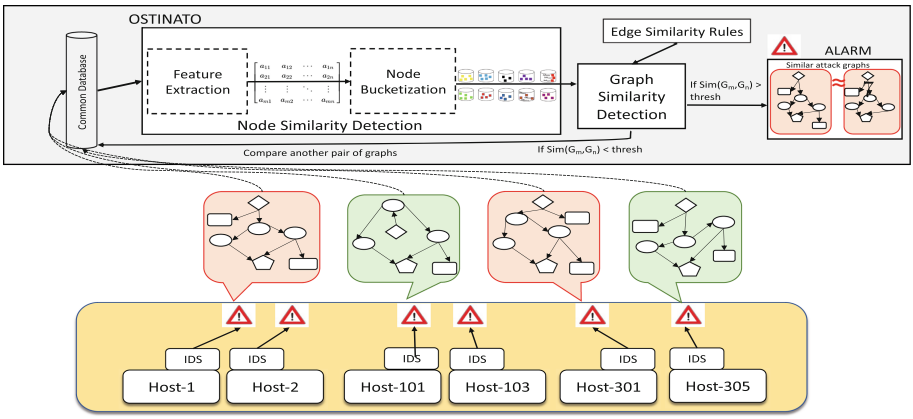

Fig. 2. OSTINATO architecture.

Node Label Content. Node labels consist of text extracted from audit logs information. They typically contain identifiers, e.g., the names and paths of the entities, command line invocation (for processes), flags, and other entity definitions. Because these entities are in different hosts, such labels may not be the same, even for processes often presumed to be similar. For instance, exact string comparison would be unable to identify the two nodes from two different graphs `C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell -noP -w 1` and `C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell.exe -NoP -NonI -w` as similar when actually they behave similarly in the host level. Furthermore, attackers may also invoke processes differently by using different order of similar command-line arguments and other means. An effective node similarity computation method must take into consideration all these factors.

Approximate Node Behavior. Different variations of string comparison would also not result in accurate node similarity. For instance, one can extract only the file name from the path of an image before performing the string comparison so that different directory structures do not interfere with the matching. In this case, labels like `C:\Windows\System32\WindowsPowerShell\v1.0\PowerShell.exe` would match. While such a solution might perform better than string comparison in certain cases, it would perform poorly for complex processes such as PowerShell, python that act as interpreters. These processes can exhibit multiple different behaviors depending on their input and command-line arguments and cannot be assumed to be similar only because they have the same name.

Solution. In our solution, we consider the node labels as textual representations of the nodes' behavior and use a notion of text similarity to determine node similarity. Beyond the two naive string-based approaches that we mentioned, there are several other approaches used for dealing with text similarity like Bag of Words [31], Word2Vec [43], or ConceptNet [40]. These approaches, however, incorporate concepts derived from natural text, such as synonyms, and cannot be directly used in our problem where the text is composed of processes, file names, and command-line options. In practice, a good solution should: 1) be applicable to the domain of terms appearing in audit logs and not rely or depend on assumptions of natural text, 2) use general intuitions about anomalies in data related to attacker activities.

We first consider each node label as a text document and use the *Term Frequency-Inverse Document Frequency (TF-IDF)* method [32] to create a feature matrix that captures the presence of words inside the nodes and their importance. TF-IDF makes no assumptions on the kind of text it acts on, and it allows evaluating how relevant a phrase is to a document in a group of documents as a statistical measure. Next, we use Locality-Sensitive Hashing (LSH) on the feature matrix [53] to create *similarity buckets*, where nodes in the same bucket are similar to each other. We describe these two steps next.

3.2.1 Feature Extraction

The first step in our approach creates a matrix representation of the node labels in the tagged provenance graphs, which can be used in the next step of the approach. This matrix, which we call *feature matrix* is built by considering the node labels as text documents and applying the TF-IDF measure over them. TF-IDF is a measure for determining how relevant a word is inside a set of documents [32]. The TF part represents the number of times a term appears in a single document (node). The IDF part, on the other hand, represents the informativeness of a term. In particular, a term that appears more frequently in all the nodes is expected to be less informative compared to one that rarely appears in those nodes. In our approach, we define as 'terms' the words inside the node labels separated by white spaces, and we consider each node as a separate document. In particular, each node (i.e., subject or object) is a document d_i: $d_i = \{t_1, t_2, t_3, ..., t_n\}$. We create two sets of such documents D_{sub}, representing all subject nodes (processes), and D_{obj} representing all object nodes (registry, files, IP). In the following details, we describe only the steps related to D_{sub} for space reasons. Steps for D_{obj} are identical. In the first step, we calculate the TF-IDF score for each term appearing in the documents in D_{sub}. This score is the product of the two features: The term frequency of term t in document d_i is shown in Equation 1 where the numerator is the number of times t appears in d_i, and the denominator represents the total number of terms (t') in d_i. The inverse document frequency of the term is shown in Equation 2 where the numerator is the total number of documents (subject nodes), and the denominator represents the number of documents (subject nodes) that contain term t.

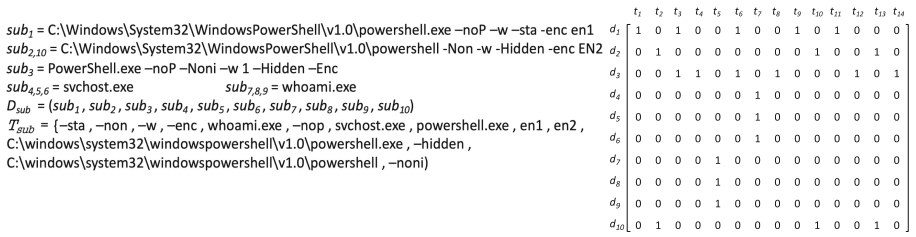

Fig. 3. Example feature matrix. The rows represent node labels. The columns represent words and the cells represent words with TF-IDF higher than median TF-IDF of the corresponding document.

$$\text{tf}(t, d_i) = \frac{f_{t,d_i}}{\sum_{t' \in d_i} f_{t',d_i}} \tag{1}$$

$$\text{idf}(t, D_{sub}) = \log \frac{|D_{sub}|}{|\{d_i \in D_{sub} : t \in d_i\}|} \tag{2}$$

After calculating the TF-IDF score for all terms in the documents in D_{sub}, for each $d_i \in D_{sub}$, we calculate the median score $\hat{\mu}$ of the TF-IDF values of its terms. Next, we build a matrix M_{sub} with dimensions $m \times n$ where m equals the number of documents $|D_{sub}|$ and n equals the total number of terms appearing in the documents in D_{sub}. A row in the matrix represents a document, while a column represents a term. If term t exists in d_i:

$$M_{d_i t_i} = \begin{Bmatrix} 1, \text{ if } \textit{TF-IDF}_t \geq \hat{\mu} \\ 0, \textit{ otherwise} \end{Bmatrix}$$

The M_{sub} matrix represents the relevance of each term inside each document. The intuition behind this matrix is that we want to keep track of the terms which have appeared in the nodes and, at the same time, are important ($\geq\hat{\mu}$). This means two 1 in two different nodes (i.e., rows) in the same column indicate the presence of an important (relatively rare) term which is $\geq\hat{\mu}$ in both nodes. On the other hand, the terms that are non-relevant and therefore less informative and important ($<\hat{\mu}$) will be represented as zeros. Figure 3 depicts an example of a M_{sub} matrix. The 10 subject nodes in the tagged provenance graphs are shown at the top of the figure. These correspond to 10 rows in the matrix. Each column in the matrix represents one of the words presented in the node labels. The matrix cells represent the presence (1) or absence (0) of a TF-IDF value that is larger than the median TF-IDF in each row.

3.2.2 Node Bucketization

The next step in our approach is to determine the (approximate) similarity between nodes, represented by the rows of the feature matrix M_{sub} (and M_{obj} for the object nodes). In particular, we want to cluster similar nodes into similar 'buckets'. To do this, we compute the similarity between each pair of rows in the TF-IDF matrix M by using the Jaccard similarity measure among them [46].

This measure is calculated using both the number of elements that the two rows share and the number of elements they do not. However, using this measure directly on the rows of matrix M would present some scalability issues. For a matrix of m rows and n columns, the time complexity of these comparisons is $O(nm^2)$. Given that there could be millions of nodes and several (hundreds in some cases) terms, this method would be computationally expensive.

To deal with this issue, we use a version of Locality-Sensitive Hashing (LSH) with Minhash [21]. LSH traditionally employs shingling, which breaks down large documents into sequences of length k of characters called k-shingles. Used traditionally for detecting near-duplicate documents (e.g., plagiarism detection), LSH methods hash data records into buckets such that records similar to each other are placed in the same bucket with a high probability. In contrast, records distant from each other are likely to be placed in separate buckets.

In Ostinato, we adapt LSH to solve our problem by using the TF-IDF feature matrix M rows instead of k-shingles as input. Because the TF-IDF feature matrix encodes a semantic representation of documents (i.e., nodes) that k-shingles do not have, we believe this is a better approach than using only the LSH method over the documents. In particular, its *Minhash* function, can project high-dimensional binary vectors like M_{sub} to a low-dimensional vector of integers H by reducing the sparseness of the former. This transformation has the property that if the Jaccard index, $J(d_i, d_j)$ between two rows of M_{sub} is high, then the probability value $Pr(H(d_i) == H(d_j))$ is also high. After creating the signature matrix H_M, we calculate pairwise row similarities using the formula:

$$Sim(H(d_i), H(d_j)) = \frac{|H(d_i) \cap H(d_j)|}{D}$$

where the numerator is the size of the row intersection operator (over integers) and the denominator is the size of the rows. The value of this similarity is between 0 and 1. We finally place two nodes in the same bucket if their corresponding similarity is above a threshold J_T. This threshold is specific to the kind of data a system produces and can be tweaked by a domain expert based on their knowledge of the hosts and audit logs they produce.

An evasion technique that attackers may try to use is to change the number of command line arguments in order to have two subject nodes in different buckets. This technique, however, is not likely to be successful for several reasons. To carry this out, the attacker has to include the command line arguments that carry out the objectives in the two subject nodes. To be able to place the nodes in different buckets, the attacker must change the values of the corresponding terms so that the terms' TF-IDF values are below the median in one node (so as to be represented as a 0 in the feature matrix) and above the median in the other node (so as to be represented as a 1 in the feature matrix). We point out that, due to the presence of the IDF, these median values cannot be controlled by the attacker but are a parameter of the system as a whole. Thus, if for instance, an attacker tries to modify the TF term by adding more values in the command line, they would also inherently change the IDF term. This effectively raises the bar of difficulty for the attacker.

Table 1. Edge label similarity rules. *S(name)* denotes the suspiciousness label, *Label.sub* is the subject and *Label.obj* is the object. ≈ denotes string containment.

Information flow similarity	Prerequisites
$E_i \equiv E_j$	System call labels E_i and E_j are the same
Load ≡ Exec	For all cases
Fork ≡ Exec	For all cases
Write ≡ Create	For all cases
Read ≡ Exec	$(Read.sub \approx$ PowerShell$) \wedge (Exec.sub \approx$ PowerShell$)$ $\wedge$ $Read.obj \approx$ {.ps1, .psd1, .psm1} $\wedge$ $Exec.obj \approx$ {.ps1, .psd1, .psm1}
TaskStart ≡ ProcessCreate	$(S(TaskStart) \in$ {Untrusted_Exec} $\wedge S(OpenProcess) \in$ {Untrusted_Exec})
Read ≡ Load	if $(Read.obj \in$ {shared_objects} $\wedge$ $Load.obj \in$ {shared_objects})

3.3 Edge Label Similarity

The edge labels can be very valuable in determining the similarity among tagged provenance graphs. In particular, *system call* labels can inform us about activity similarity at OS level, while *suspiciousness* labels carry much more meaningful information about attackers' goals. To capture edge label similarity, we incorporate several matching rules in OSTINATO. Given the finite number of suspiciousness and system call labels, this task does not need to be automated and can take advantage of domain knowledge. The edge label similarity rules that are used in OSTINATO are shown in Table 1. The first column shows the similarity between edges using system call label names, while the second column shows the prerequisites that must be met for two edges to be considered similar. We also require that the suspiciousness labels are the same for all edge pairs (we do not show this in the table for space reasons). For instance, two edges with *exec* system calls labels are considered similar only if their suspiciousness labels are also the same (e.g., *Untrusted_Exec*). In Table 1 the first row represents the trivial cases where both types of labels are the same (e.g., *read* and *read)*. The following rows represent cases where edges with different labels can be considered similar. For instance, the fifths row represents a rule that states that a `read` in host i is equivalent to an *exec* in host j if either subject contains (≈) 'PowerShell' and if the suspiciousness label of either edge is different from *Initial_Compromise*. This rule captures the duality of PowerShell scripts, which can be both read and execute. We point out that this table only deals with similarity among edge labels without considering the nodes. In other words, the table only captures information flow similarity. To fully evaluate if an event is similar to another, we also need to make sure that the nodes connected by that edge are similar to one another. We provide the details about this procedure in the next section.

Algorithm 1: Graph Similarity Algorithm.

function SIMILARITY
  Input: G_l, G_s, Buckets map $B : nodes \rightarrow buckets$, Edge label similarity rules E_L, MPS $= \{(N_s, N_l) | N_s \in G_s \wedge N_l \in G_l \wedge B(N_s) = B(N_l)\}$, $Len_{MPS} = |MPS|$
  Output: Final_Sim(G_l, G_s)
  for $(N_s, N_l) \in MPS$ **do**
    MPS = MPS $\setminus$ (N_s, N_l)
    Total_Acc += Parallel_BFS(N_s, N_l)
  Final_Sim$(G_l, G_s) = \frac{Total_Acc}{Len_{MPS}}$
function PARALLEL_BFS(N_s, N_l)
  Sim = 0
  Enqueue(N_s, Q_s); Enqueue(N_l, Q_l)
  while $(Q_s \neq \varnothing \wedge Q_l \neq \varnothing)$ **do**
    N_s = Dequeue(Q_s); N_l = Dequeue(Q_l)
    MPS = MPS $\setminus$ (N_s, N_l)
    $NN_s = \{V | (N_s, V) \in E(Gs) \vee (V, N_s) \in E(G_s)\}$
    $NN_l = \{V | (N_l, V) \in E(G_l) \vee (V, N_l) \in E(G_l)\}$
    for $v_1 \in NN_s$ **do**
      for $v_2 \in NN_l$ **do**
        if $(v_1, v_2) \in MPS$ **then**
          Enqueue(v_1, Q_s); Enqueue(v_2, Q_l)
          if $E_L(N_s, v_1) == E_L(N_l, v_2)$ **then**
            Sim+= W_1
          else
            Sim+= W_2
        else
          if $E_L(N_s, v_1) == E_L(N_l, v_2)$ **then**
            Sim+= W_3
    return Sim

3.4 Graph Similarity Detection

The final step of OSTINATO, is to determine whether two tagged provenance graphs belonging to two different hosts are similar or not. These graphs, however, can: 1) have widely different sizes, depending on the number of suspicious activities detected in each host, 2) be composed of different activities that may or may not be similar. To determine the final similarity score between two tagged provenance graphs, we use Algorithm 1, which performs in parallel two modified breadth first searches over the two tagged provenance graphs while updating a similarity score value during the traversal. This algorithm uses both the bucket information representing the node mappings and the edge label similarity rules to determine whether an initial attack graph is similar to another graph (or a set of graphs) in comparison to the attack behavior and the structure of the graph.

Algorithm 1 takes in input the two tagged provenance graphs, the edge label similarity rules E_L (Table 1), and Matched Pairs Set (MPS), which is the set of pairs of nodes from the two graphs that are in the same bucket. The algorithm chooses one such pair of nodes and performs a breadth first search traversal on

each graph using those nodes as roots. Before the traversal, it removes that pair of nodes from the set, so that it does not traverse them a second time later. During the traversal, it only follows the nodes of the two neighborhoods that are in the same bucket (lines 11–16). At any iteration of the loop in line 16, considers three cases of similarity, to which it assigns three different weights: 1) W_1 corresponding to complete edge matching (nodes and edge labels), 2) W_2 corresponding to the two nodes matching but the edge labels being different (E.g., firefox writes to a file in one host and firefox reads from the same file in another host), 3) W_3 corresponding to the case where the subject node and edge labels match but the object names do not match. This approach works across hosts with different OS because even though the names of processes are varied across different OSes, the malicious behaviour and its usage would place the nodes into the respective similar bucket. We use different weights in order to take into account the differences in the number of buckets of subjects and of objects discussed earlier (see end of Sect. 3.2.2). The weights we used in our evaluation for W_1, W_2 and W_3 are 1, 0.2 and 0.8 respectively. From our evaluation, we conclude that these values can be generalized for different OSes or platforms. Additionally, the value of these weights can be customized further by analysts to look for specific nodes during forensic analysis or threat hunting.

After the final similarity score between two graphs is determined in line 7, we raise an alert if it is higher than a predefined similarity threshold. The value of this threshold depends on several factors, including the systems and the filtering actions of the local IDS detectors. We include a discussion about this threshold and others in the Evaluation.

4 Evaluation

This section evaluates OSTINATO by two different experiments using different datasets generated by DARPA red team exercises. The first experiment is part of large-scale 3-day long red team exercise [12] in an environment containing 500 Windows hosts in which the major attacker activities were concentrated in the first two days. The details of this experiment are discussed in Sect. 4.1. We further evaluate OSTINATO on a second experiment which contains two separate multi-host attack campaigns involving hosts with different OSes [11].

We deployed OSTINATO on a desktop with Intel Xeon W CPU @ 3.2 GHz and 32 GB memory running macOS Big Sur. As a local IDS, we used HOLMES, which we obtained from its developers [45]. HOLMES uses rules of *connected* TTPs to detect attacks unfolding inside a single host. Its final output consists of provenance graphs representing the activities detected as TTPs. These graphs are next sent to OSTINATO, which determines similarities among them.

Results Summary. We performed our experiments in a *threat hunting* scenario where, given some attacker activities in one host, we use OSTINATO to find similar activities in other hosts. Thanks to this kind of search, OSTINATO was able to uncover attacks in 14 more hosts than HOLMES on the first day (detailed description in Sect. 4.1) and 7 more (21 in total) on the second day. This is due

to the lighter footprint of the attacks on the additional hosts, which fall under Holmes' detection threshold. In fact, to make Holmes detect the same attacks as Ostinato on those additional hosts, we had to lower Holmes' detection threshold significantly, producing several hundreds of alerts and false positives.

4.1 Ostinato Efficacy

Dataset Overview. We evaluate Ostinato over two datasets: first, OpTC-NCR2 a large dataset [12] of audit logs produced as part of DARPA's CHASE program. The dataset was collected over a period of two days on 500 hosts. During these days, a red team performed several APT-like attacks on 24 of those hosts. Benign activities were generated both manually and by running scripts. The second dataset was collected as part of DARPA's Transparent Computing (TC) program [11]. During this engagement, the attackers replicated APT-type scenarios across multiple hosts on different platforms.

Ground Truth. The data are accompanied by PDF documents written by the red team describing the attackers' activities performed on each host. The ground truth was built from these descriptions and the process ids contained in those descriptions. In particular, if a tagged provenance graph contains one or more of those process id-s it is considered as an *attack graph*. In addition, we build a ground truth of pairs of similar attack graphs manually.

Detailed Results. Table 2 shows Ostinato's results for the first two days of the OpTC-NCR2 dataset. The left table (a) contains the results of the first day, while the right table the results of the second day. The tables contain pair-wise similarity scores among tagged provenance graphs that were a part of the attackers' activities and the maximum similarity score (Column B_{max}), and mean similarity score (Column BM_1) between each graph that represents attacker activities and the other provenance graphs that represent benign activities. In Table 2(b), G_a, G_b, and G_c represent 3 tagged provenance graphs generated by Holmes (and enhanced by Ostinato) in its default optimal detection threshold setting, which produces true positives and a low number of false positives. These were present on only one host, hence comparisons among them are not calculated. The rest of the tagged provenance graphs from 7 distinct hosts ($G_d - G_j$) represent activities with a smaller footprint, which were not detected as attacks by Holmes in its default detection threshold. In our experiments, we reduced Holmes's detection threshold obtaining a total of 689 more graphs from 500 hosts. Using Ostinato, we identified 7 (from 7 distinct hosts) out of 689 graphs that were similar to the initial 3 tagged provenance graphs as part of the attacker's activities, while the rest were false positives. In these hosts, the attack's footprint was smaller because the attackers performed only a small number of malicious activities like running some PowerShell scripts in some hosts or communicating to an untrusted C2 server in other hosts. There were several benign graphs generated in those 7 affected hosts, for which Ostinato generated low similarity scores as per expectations. Ostinato was able to successfully correlate attacker activities found in the initial attack graphs (G_a, G_b, G_c) among hundreds of other graphs.

Table 2. (a) Day-1 evaluation results, (b) Day-2 evaluation results. Similarity matrices representing the similarity score between attack graph G_x (represented in diagonal) and other attack graphs. The columns B_{mean} represent the average similarity score between the attack graph (G_x) and other benign graphs, and B_{max} represent the maximum similarity score between the attack graph (G_x) and other benign graphs.

(b)

Graph id	G_a	G_b	G_c	G_d	G_e	G_f	G_g	G_h	G_i	G_j	B_{mean}	B_{max}
G_a	x	x	x	0.85	0.81	0.9	0.8	0.8	0.88	0.8	0.15	0.26
G_b	x	x	x	0.97	0.88	0.9	0.8	0.8	0.88	0.68	0.23	0.54
G_c	x	x	x	0.82	0.86	0.9	0.9	0.93	0.88	0.91	0.23	0.28
G_d	x	x	x	x	0.87	0.9	0.8	0.8	0.88	0.6	0.18	0.62
G_e	x	x	x	x	x	0.9	0.8	0.8	0.88	0.68	0.13	0.37
G_f	x	x	x	x	x	x	0.8	0.8	0.9	0.8	0.18	0.52
G_g	x	x	x	x	x	x	x	0.9	0.9	0.8	0.32	0.55
G_h	x	x	x	x	x	x	x	x	0.93	0.86	0.08	0.14
G_i	x	x	x	x	x	x	x	x	x	0.82	0.35	0.57
G_j	x	x	x	x	x	x	x	x	x	x	0.31	0.51

(a)

B_{max}	B_{mean}	G_{18}	G_{17}	G_{16}	G_{15}	G_{14}	G_{13}	G_{12}	G_{11}	G_{10}	G_9	G_8	G_7	G_6	G_5	G_4	G_3	G_2	G_1	Graph id
0.29	0.14	x	G_{18}																	
0.53	0.27	0.66	x	G_{17}																
0.57	0.23	0.66	0.66	x	G_{16}															
0.44	0.18	0.66	0.66	0.66	x	G_{15}														
0.19	0.06	0.66	0.95	0.66	0.66	x	G_{14}													
0.62	0.37	0.66	0.66	0.66	0.66	0.66	x	G_{13}												
0.22	0.18	0.66	0.66	0.66	0.66	0.66	0.66	x	G_{12}											
0.27	0.16	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_{11}										
0.59	0.24	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_{10}									
0.58	0.36	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_9								
0.31	0.11	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_8							
0.24	0.09	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_7						
0.52	0.29	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_6					
0.39	0.17	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_5				
0.56	0.29	0.66	0.93	0.66	0.66	0.93	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	x	G_4			
0.41	0.15	0.66	0.86	0.66	0.66	0.95	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.95	x	G_3		
0.64	0.31	0.66	0.91	0.66	0.66	0.85	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.88	0.84	x	G_2	
0.68	0.25	0.66	0.78	0.66	0.66	0.88	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.66	0.85	0.90	x	x	G_1

Table 2(a) represents the results of the first day of activities. During this day, at its "optimal" threshold, HOLMES detected only four initial attack graphs from three distinct hosts in its default detection threshold with no false positives. However, the attackers conducted activities in several other hosts. These activities do not cross HOLMES' detection threshold. We subsequently reduce HOLMES' detection threshold to capture all possible attacker activities. As a result, we generated 424 more graphs from those 500 hosts. OSTINATO correlated 14 graphs (G_5–G_{18}) out of those 424 graphs from 14 distinct hosts to be similar to the initial starting points (G_1–G_4). As can be seen from the table, the average pair-wise similarity values obtained among the attack graphs are significantly higher than those obtained when an attack graph is compared with benign graphs (even with other benign graphs produced from those 14 compromised hosts).

Evaluation on TC Dataset. We evaluated OSTINATO on one additional dataset generated as part of Engagement-5 organized by DARPA [11]. The results are shown in Table 3. The red team used secure ssh sessions to move from one host to another, starting from a pivot host. In each host, the red team performed some suspicious operations (like `nmap, ls, ifconfig`) and exfiltrated a file (`passwd`) back to the pivot machine before moving to a new host. Cadets-1 was a FreeBSD machine in this setup, Theia-1 and Trace-2 were Linux machines, and FiveD-3 was a windows machine. Although the attacks on these different hosts were not identical, due to similar attacker-created processes and invocations across the 4 hosts of different platforms, OSTINATO was able to detect the similarity across hosts successfully.

Table 3. Engagement-5 evaluation results. Multi-host ssh campaign.

Host Name	Cadets-1	Theia-1	Trace-2	FiveD-3	Benigns Mean
Cadets-1	x	0.93	0.86	0.85	0.0
	Theia-1	x	0.87	0.86	0.0
		Trace-2	x	0.93	0.0
			FiveD-3	x	0.0

Similarity Threshold, Precision and Recall. We define a false positive as a comparison between a benign graph and an attack graph that results in a value above a specific similarity threshold. In turn, a false negative is a comparison between two attack graphs that results in a value below the similarity threshold. To determine the optimal threshold in our dataset, we varied the threshold over a specific range and collected the false positives, false negatives, true positives, and true negatives using the ground truth. The results of this experiment are shown in Fig. 4(a), which depicts the values of the precision, recall, F1-score, and accuracy as a function of the similarity threshold. The accuracy metric measures the ratio of correct outcomes over the total number of outcomes ($Accuracy = (TP + TN)/(TP + TN + FP + FN)$). As can be seen in Fig. 4(a) OSTINATO achieves a high accuracy (∼0.97), Evidently, the optimal value for the F1-score is for values of the similarity threshold around 0.5, which produces a total of 15 false positives over both days.

4.2 Node Similarity Accuracy

In this subsection, we describe an independent evaluation of the approach described in Sects. 3.2.1 comparing it with other possible similarity detection methods, in particular, string matching (SM) and k-means clustering technique with a different number of clusters.

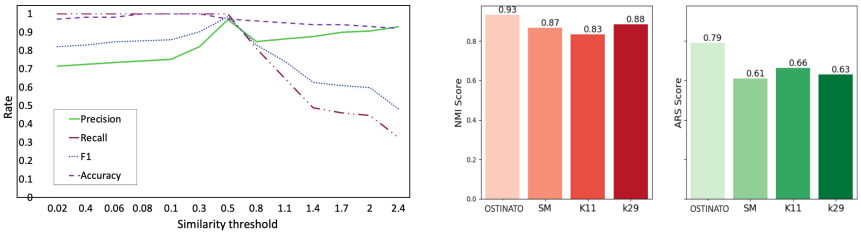

Fig. 4. (a) Precision, Recall, F-Score, Accuracy as a function of the similarity detection threshold. (b) Clustering performance comparison between OSTINATO and other methods using NMI and ARS.

Ground-Truth Dataset. To measure our approach's performance, we created a ground truth dataset of expected buckets for a subset of nodes in the dataset of 3700 subject nodes. We asked multiple security experts to assign each node to a bucket of similar nodes that represents a specific behavior. After this step,

the experts met to discuss their assignments, and if there were disagreements, a new step of assignments were executed. This cycle of assignment-discussions was repeated until consensus was reached.

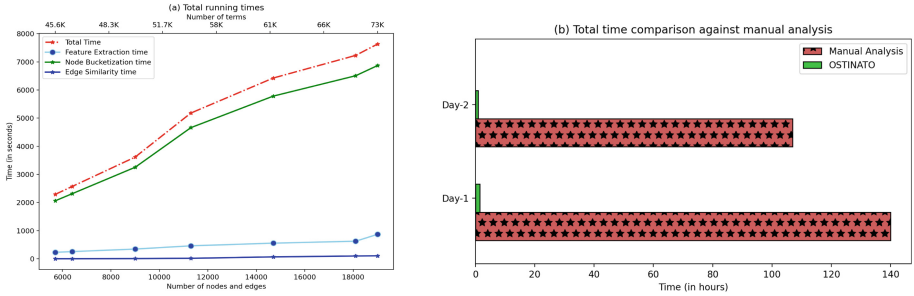

Fig. 5. (a) OSTINATO performance representing the running times of different steps. (b) Total time comparison against manual analysis.

Comparison Against Other Approaches. We compare our approach against several common approaches, including SM and the common clustering approach of k-means clustering with TF-IDF with different k values. To evaluate our approach against the k-means clustering, the most fundamental step is to define the optimum number of clusters (i.e., k). For this step, we chose two approaches. First, we used the number of clusters based on the number of clusters in the ground truth (i.e., $k = 29$). Second, to choose the number of clusters, we used the elbow method [33], a common heuristic approach to determine the optimum number of k which picks the elbow of the curve as $k = 11$ as the optimum k. Choosing two values for k enables us to evaluate our approach against the two probable number of clusters, 1) expected number of clusters based on the ground truth 2) suggested number of (optimum) clusters by elbow method.

To measure the performance of our bucketizing approach against other approaches, we use two standard quality metrics for clustering algorithms: the Adjusted Rand Score (ARS) and the Normalized Mutual Information (NMI) metrics [23] which use different methods to compare the quality of clustering algorithms when the number of clusters in ground truth clustering and that in the prediction are different. The overall results are shown in Fig. 4(b). As can be seen from this figure, our approach outperforms both SM and k-means for different values of k. The main reason for the better performance of our approach is the LSH step, which is able to better capture approximate similarity.

4.3 Run-Time Performance

We measure the run-time performance of OSTINATO by creating sets of tagged provenance graphs of different sizes by varying the underlying IDS detection threshold on the *Day 1* campaign data. The run time performance of the different steps of OSTINATO is shown in Fig. 5(a). To obtain different datapoints, we group

the graphs into 7 sets of increasing sizes. The number of nodes and edges in each of these sets is shown in the primary x-axis. The secondary x-axis at the top reflects the total number of words (or terms) present in the nodes of each set. As can be seen from the figure, the most expensive part of the approach is node bucketization, amounting to approximately 90% of the total time when comparing thousands of nodes. This is mainly due to the large size of the feature matrix. Graph similarity (Algorithm 1), represented by the blue line, is the fastest component, usually taking just a few seconds.

4.4 Threat Alert Fatigue Mitigation

Our evaluation of Ostinato for threat alert fatigue mitigation shows promising results. In situations where local host-based detection systems produce a large number of alerts, Ostinato can help cyber analysts to pinpoint hosts where similar attacker activities are occurring, filtering out thousands of benign alerts (or false positives from IDS). Across the attack campaign [12] for two days, the underlying IDS produced more than 1000 alerts, which is really unfeasible for manual analysis. Comparatively, when those graphs are fed into Ostinato along with the 7 initial attack graphs, it successfully correlated to 21 alerts from distinct hosts where it found similar attack behavior. According to several studies [6–8], it usually takes about 10–30 min to investigate an alert manually by cyber analysts. Assuming 15 min on average for each alarm, a cyber analyst would require 140 h to investigate the average alarms of each day produced by the IDS in our experiment. Alternatively, as shown in Fig. 5(b), Ostinato takes around 167 min to complete the analysis of all the alarms generated, reducing false positives of the underlying IDS by more than 90%.

4.5 Comparison with Other Tools

We compare some of Ostinato's aspects with some popular graph matching approaches. Ostinato is much better suited for cross-host attack correlation than compared to other popular graph matching techniques. The features that stand out in comparison with other graph pattern matching approaches is that Ostinato can perform accurate node label approximations even when similar nodes exhibit different behaviors, does not require training to implement, and performs context relative edge comparison, which is essential for cross-host attack correlation purposes. We outline the qualitative comparison against the existing tools in Table 4. Since majority of such tools are not open source or easily available, an experimental comparison of Ostinato with those approaches is unfeasible. Out of these only SimGNN [18] is publicly available, however the nodes and edges are much more simpler in it's evaluated datasets and only contains of integers instead of actual names of processes, objects or edges.

5 Related Work

Several approaches have been proposed to deal with cross-host attack detection via cross-host information tracking. These approaches rely on the presence of

Table 4. OSTINATO vs. other approaches

Approach	Node label approximation	Node embeddings	Context-relative edge comparison	Training required
SimGNN [18]	✗	✓	✗	✓
Poirot [44]	✗	✗	✗	✗
Deltacon [36]	✗	✓	✗	✗
OSTINATO	✓	✓	✓	✗

information flow data between entities (e.g., processes) across hosts [30,51]. This approach, however, requires fine-grained taint tracking, which relies on system instrumentation it requires some modifications to existing systems.

Log-Based Threat Hunting: A wide variety of systems leverages different types of logs for threat-hunting purposes. Hercule [48] is a log-based detection system modeled on the community discovery problem. It correlates logs from multiple sources and detects attack communities. Oprea et al. [47], Romero-Gomez et al. [49] leveraged DNS, web-proxy logs in order to detect and visualize threats in a network. Bilge et al. [19] leveraged NetFlow logs to detect Botnet C&C servers and distinguish them from the benign traffic. The DNS logs are also leveraged extensively [17] for the detection of malicious domains. Several systems [27,41,45] make use of different logs just as OSTINATO for efficient threat hunting, forensic analysis, or real-time detection of cyberattacks. Most of the mentioned approaches that deal with cross-host activities rely on network logs, however, while OSTINATO is used over audit host logs.

Provenance Graph Analysis: BackTracker [35] first introduced the concept of generating a provenance graph from the kernel audit logs. In recent years, significant progress has been made for log reduction, compression techniques and tracking OS-level dependencies [26,28,37] in order to facilitate detection of benign events from the suspicious ones as well as to reduce storage overheads. Moreover, recent studies have used provenance graphs effectively for a wide variety of security problems such as identification of zero-day attack paths in ZePro [54], automated provenance triage in NoDoze [27], real-time attack detection and attack scenario reconstruction [45]. While sharing use of provenance graphs, OSTINATO's approach is different from these works. In fact, OSTINATO looks for similar subgraphs across multiple provenance graphs as a signal for multi-host attack correlation.

6 Conclusion

We present OSTINATO, which is based on the intuition that attackers have similar goals on multiple hosts during a campaign. OSTINATO implements an approach for correlating similar attacker activities across different hosts and implements a novel approximate node matching technique. It further uses the attack semantics

to detect similarities among tagged provenance graphs. We successfully evaluate Ostinato on two datasets created by DARPA red team engagements.

References

1. 2021: Year in review. https://thedfirreport.com/2022/03/07/2021-year-in-review/
2. Adversarial tactics, techniques and common knowledge. https://attack.mitre.org/
3. Apt cybercriminal campagin collections. https://bit.ly/364iN8U
4. Detecting lateral movement with windows event logs. https://bit.ly/3hQyF1D
5. Mandiant (2013). https://bit.ly/3MA0N7b
6. Alert fatigue: 31.9% anaysts ignore alerts. https://bit.ly/3MyE9fA (2017)
7. Automated incident response (2017). https://bit.ly/3hPm3Ia
8. New research from advanced threat analytics finds MSSP incident responders overwhelmed by false-positive security alerts (2018). https://prn.to/37hqsS9
9. Destructive attack "dustman" (2019). https://bit.ly/3tHX7YC
10. Dramatic reductions in alert fatigue with crowdscore (2019). https://bit.ly/3IZD9is
11. Tc engagement-5 (2019). https://github.com/darpa-i2o/Transparent-Computing
12. Optc dataset (2020). https://github.com/FiveDirections/OpTC-data
13. Groups (2021). https://attack.mitre.org/groups/
14. Lateral movement (2021). https://bit.ly/3t63ru1
15. Lateral tool transfer (2021). https://attack.mitre.org/techniques/T1570/
16. What makes lateral movement so hard to detect? (2021). https://bit.ly/3hUl0qg
17. Antonakakis, M., et al.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: 21st {USENIX} Security Symposium ({USENIX}) (2012)
18. Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., Wang, W.: SimGNN: a neural network approach to fast graph similarity computation. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 384–392 (2019)
19. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: Proceedings of the 28th Annual Computer Security Applications Conference (2012)
20. Bowman, B., Laprade, C., Ji, Y., Huang, H.H.: Detecting lateral movement in enterprise computer networks with unsupervised graph {AI}. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses. RAID (2020)
21. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations. J. Comput. Syst. Sci. **60**(3), 630–659 (2000)
22. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection framework. In: Proceedings 2002 IEEE Symposium on Security and Privacy (2002)
23. Emmons, S., Kobourov, S., Gallant, M., Börner, K.: Analysis of network clustering algorithms and cluster quality metrics at scale. PLoS One **11**(7), e0159161 (2016)
24. Gallagher, B.: Matching structure and semantics: a survey on graph-based pattern matching. In: AAAI Fall Symposium: Capturing and Using Patterns for Evidence Detection, pp. 45–53 (2006)
25. Hajizadeh, M., Phan, T.V., Bauschert, T.: Probability analysis of successful cyber attacks in SDN-based networks. In: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), pp. 1–6. IEEE (2018)
26. Hassan, W.U., Bates, A., Marino, D.: Tactical provenance analysis for endpoint detection and response systems. In: 2020 IEEE Symposium on Security and Privacy (2020)

27. Hassan, W.U., et al.: NoDoze: combatting threat alert fatigue with automated provenance triage. In: Network and Distributed Systems Security Symposium (2019)
28. Hossain, M.N., Sheikhi, S., Sekar, R.: Combating dependence explosion in forensic analysis using alternative tag propagation semantics. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 1139–1155. IEEE (2020)
29. Jeh, G., Widom, J.: Simrank: A measure of structural-context similarity. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543 (2002). https://bit.ly/3HXbqgQ
30. Ji, Y., et al.: Enabling refinable cross-host attack investigation with efficient data flow tagging and tracking. In: 27th {USENIX} Security Symposium ({USENIX} Security 18) (2018)
31. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0026683
32. Joachims, T.: A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Technical report, Carnegie-Mellon Univ., Pittsburgh, PA, Dept. of CS (1996)
33. Ketchen, D.J., Shook, C.L.: The application of cluster analysis in strategic management research: an analysis and critique. Strateg. Manag. J. **17**, 441–458 (1996)
34. King, D.: Spotting the signs of lateral movement (2018). https://splk.it/3vTiQ2C
35. King, S.T., Chen, P.M.: Backtracking intrusions. In: SOSP. ACM (2003)
36. Koutra, D., Vogelstein, J.T., Faloutsos, C.: DeltaCon: a principled massive-graph similarity function. In: Proceedings of the 2013 SIAM International Conference on Data Mining. SIAM (2013)
37. Krishnan, S., Snow, K.Z., Monrose, F.: Trail of bytes: efficient support for forensic analysis. In: Proceedings of the 17th ACM CCS, pp. 50–60 (2010)
38. Kruegel, C., Valeur, F., Vigna, G.: Intrusion Detection and Correlation: Challenges and Solutions, vol. 14. Springer, New York (2004). https://doi.org/10.1007/b101493
39. Lee, K.H., Zhang, X., Xu, D.: LogGC: garbage collecting audit log. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (2013)
40. Liu, H., Singh, P.: ConceptNet-a practical commonsense reasoning tool-kit. BT Technol. J. **22**(4), 211–226 (2004)
41. Liu, Y., et al.: Towards a timely causality analysis for enterprise security. In: NDSS (2018)
42. McKay, B.D., Piperno, A.: Practical graph isomorphism, II. J. Symb. Comput. **60**, 94–112 (2014)
43. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
44. Milajerdi, S.M., Eshete, B., Gjomemo, R., Venkatakrishnan, V.: Poirot: aligning attack behavior with kernel audit records for cyber threat hunting. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)
45. Milajerdi, S.M., Gjomemo, R., Eshete, B., Sekar, R., Venkatakrishnan, V.: Holmes: real-time apt detection through correlation of suspicious information flows. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 1137–1152. IEEE (2019)

46. Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S.: Using of Jaccard coefficient for keywords similarity. In: Proceedings of the International Multiconference of Engineers and Computer Scientists, vol. 1, pp. 380–384 (2013)
47. Oprea, A., Li, Z., Yen, T.F., Chin, S.H., Alrwais, S.: Detection of early-stage enterprise infection by mining large-scale log data. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 45–56. IEEE
48. Pei, K., et al.: Hercule: attack story reconstruction via community discovery on correlated log graph. In: Proceedings of the 32nd ACSAC, pp. 583–595 (2016)
49. Romero-Gomez, R., Nadji, Y., Antonakakis, M.: Towards designing effective visualizations for DNS-based network threat analysis. In: 2017 IEEE Symposium on Visualization for Cyber Security (VizSec), pp. 1–8. IEEE (2017)
50. Sadoddin, R., Ghorbani, A.: Alert correlation survey: framework and techniques. In: Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, pp. 1–10 (2006)
51. Sahabandu, D., Xiao, B., Clark, A., Lee, S., Lee, W., Poovendran, R.: Dift games: dynamic information flow tracking games for advanced persistent threats. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 1136–1143. IEEE (2018)
52. Satvat, K., Gjomemo, R., Venkatakrishnan, V.: Extractor: extracting attack behavior from threat reports. In: 2021 IEEE European Symposium on Security and Privacy (EuroS P), pp. 598–615 (2021)
53. Shrivastava, A., Li, P.: In defense of MinHash over SimHash. In: Artificial Intelligence and Statistics, pp. 886–894. PMLR (2014)
54. Sun, X., Dai, J., Liu, P., Singhal, A., Yen, J.: Using Bayesian networks for probabilistic identification of zero-day attack paths. IEEE Tran. Inf. Forensics Secur. **13**, 2506–2521 (2018)