Latte: Large-Scale Lateral Movement Detection

Qingyun Liu⋆, Jack W. Stokes†, Rob Mead††, Tim Burrell††, Ian Hellen‡, John Lambert‡, Andrey Marochko†, and Weidong Cui†
⋆University of California at Santa Barbara, Santa Barbara, CA 93106 USA
†Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA
††Microsoft Corporation, Cheltenham, Gloucestershire, GL50 1TY, UK
‡Microsoft Corp., One Microsoft Way, Redmond, WA 98052 USA

Abstract—**The frequency of recent headlines indicates that attacks on governmental and corporate computer networks are increasing. Once they infect one computer, the attackers are quite likely to explore the network by accessing additional computers. Such "lateral movement", *i.e.*, the process attackers use to move from one computer to the next in a compromised network, increases the difficulties of preventing data exfiltration. To deal with challenges from large-scale data and little knowledge of the attackers, we propose *Latte*, a graph-based detection system to discover potential malicious lateral movement paths. We model computers and accounts as nodes, and computer-to-computer connections or user logon events as edges. We address the lateral movement problem in two ways. Starting with an infected computer or account, *forensic analysis* quickly identifies other compromised computers. To discover a new attack, *general detection* identifies unknown lateral movement across nodes which are not known to be compromised. A key component for general detection is a remote file execution detector which filters out the majority of the rare paths in the network. We provide separate algorithms for these subproblems and validate their effectiveness and efficiency on two, large-scale datasets, including one with a confirmed attack and one from a penetration test.**

Index Terms—**Lateral Movement, Advanced Persistent Threat**

I. Introduction

Attackers are successfully penetrating governmental and corporate computer networks with the intent of exfiltrating sensitive data at an alarming rate. Attacks that are directed towards organizations often begin with a spearphishing campaign aimed at targeted individuals which attempts to coerce the potential victims into installing malware on their computers by opening an attachment or clicking a malicious link. Once executed, the malware typically drops a backdoor providing the attacker with remote control over the infected computer. Next, the attacker begins to use *lateral movement* techniques to explore the network, *i.e.*, the process an attacker uses to move from one computer to the next in a compromised network [1].

Lateral movement can be viewed as a set of malicious paths corresponding to the attacker's activity within a large graph of benign connections created by the daily operation of the organization's users. Prior studies have shown that graph-based algorithms are promising for monitoring the network to discover vulnerabilities [2], [3] or detect anomalies [4], [5], [8]. However, they often have strong assumptions, either for the network, *e.g.*, each computer has a certain defined vulnerability state [1], or for the targeted scenario, *e.g.*, anomalous behaviors will result in unusual substructural patterns [5]. Also, most previous studies employ datasets that are either small or contain simulated attacks.

The two most important problems that organizations face related to lateral movement include forensics analysis and general detection.

Forensic Analysis Problem. Forensic analysis is needed when an organization determines that its network includes a compromised account or computer which is exhibiting confirmed evidence of a severe threat. In this case, the infected organization must quickly mobilize to identify and disable all compromised accounts and computers. For the forensic analysis problem depicted in Figure 1, we consider both inbound and outbound paths including the compromised computer. For inbound path analysis, we search for the malicious path or paths that lead to the infected computer. Similarly for outbound paths, we search for the malicious path or paths originating from the known infected computer.

General Detection Problem. While the forensic analysis problem deals with the case where an infected computer is already known, general detection involves identifying infected computers or compromised user accounts and the malicious lateral movement paths in the network connection graph without any prior *confirmed* detections. Without prior detections, analysts spend much of their time hunting for evidence of possible intrusion including lateral movement. In other words, the task is to first discover the *initial* infected computer or compromised user account.

To address both the forensics analysis and general detection problems, we propose *Latte*, a new graph-based, *lat*eral movement de*te*ction system to discover malicious lateral movement paths. *Latte* analyzes large-scale event logs collected from operational networks. In our system, we model computers and accounts as nodes, and computer-to-computer connections or user logon events as directed edges.

The system first uses Kerberos [6] service ticket request events to construct the network connection graph. Kerberos is a computer network authentication protocol which allows nodes communicating over a non-secure network to prove their identity to another in a secure manner. For the forensics analysis scenario, *Latte* uses this connection graph in combination with a list of confirmed malicious nodes to identify possible malicious lateral movement. For general detection, which is illustrated in Figure 2, *Latte* first correlates a number of system

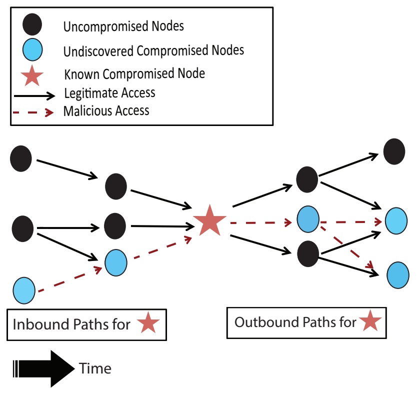

Fig. 1. Forensic Analysis. A known compromised node (denoted as a red star) is discovered and the task is to identify the unknown compromised computers and user accounts (blue nodes) along the malicious access paths (red dashed lines). Searching for potentially compromised nodes which connect to the known compromised node considers inbound paths, while discovering downstream nodes which are connected to by the known compromised node studies outbound paths.

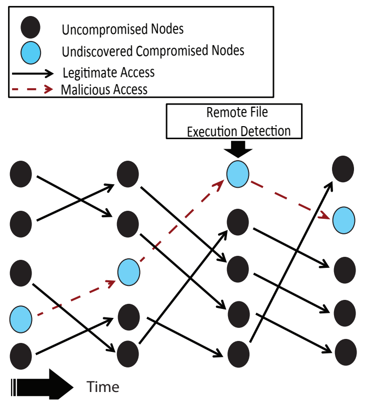
Uncompromised Nodes
Undiscovered Compromised Nodes
Legitimate Access
Malicious Access
Remote File Execution Detection
Time

Fig. 2. General Detection. The figure represents all the user accounts and computers in a large-scale network without any prior *confirmed* intrusions. The figure indicates that one computer in the graph is suspicious due to the detection of a possible remote file execution. The general detection task is to identify the malicious lateral movement path involving the blue nodes.

and security events to indicate possible remote file execution. It then combines these possible remote file execution detections with rare, anomalous paths in the network connection graph to significantly filter out benign lateral movement activity. To aid in the detection of malicious lateral movement, the figure indicates that one suspicious computer in the network has generated an alert indicating a potential remote file execution (RFE), a key component of lateral movement. When operating in isolation on a large graph, both the RFE and the rare path detectors can lead to many false positives. However, by *combining* these separate detection event streams, *Latte* can significantly reduce the number of false positives related to detecting the initial compromised node. The lateral movement path is then recovered from the rare paths which include the computer involved with the remote file execution detection. While monitoring the entire network, *Latte* generates alerts when there is suspicious lateral movement detected by our algorithm. In some conditions, *Latte* could be used to automatically disable an account or a computer's network access if the detection confidence is high enough.

Unlike an earlier graph-based system designed to detect potential lateral movement [7], an important design goal for *Latte* is not to require any kernel mode components to be installed on each host computer. This allows our system to be more easily deployed in large-scale networks without the increased attack surface, instability, and maintenance introduced by a kernel mode component. To achieve this goal, all of *Latte's* inputs are system and security events are generated by the production operating system. The previous system proposed in [7] requires potential malicious lateral movement subgraphs to be detected in a kernel mode agent. These subgraphs are then aggregated by a central system. As noted by the authors in [7] and confirmed by our experiments in Section IV, the number of potential graphs increases exponentially with the path length. *Latte* instead runs in a reasonable amount of time because the most time-consuming graph processing blocks are implemented on a large-scale MapReduce platform.

We evaluate *Latte* on two datasets. Both datasets include one instance of lateral movement. The first dataset contains events from an actual attack and was provided to us by an anonymized organization. In some cases, the Microsoft Threat Intelligence Center (MSTIC) conducts joint operations with customers to better understand targeted attacks. The second dataset contains data from an internal penetration test on Microsoft's computer network. Both datasets contain at least 90 days of Kerberos service ticket request events indicating user-computer logons and computer-to-computer connections. Moreover, the second dataset also contains additional Windows system and security events which we use to detect possible remote file execution activity.

We validate the effectiveness and efficiency of our system on these datasets separately. For forensic analysis, our method can promote the malicious lateral movement paths as the most suspicious paths in most cases. For general detection, *Latte* is able to initially discover nodes that were infected, and the corresponding lateral movement paths, during the penetration test. Key contributions of our work include the following:

- We utilize graph concepts for the problem of tracking malicious lateral movements across computers and accounts.
- We provide an algorithm to aid analysts in the forensic analysis scenario to help identify lateral movement paths into and out of a newly discovered compromised computer or account.
- We introduce a general detection algorithm for identifying previously unknown malicious lateral movement on a network by a combination of a remote file execution detector and a rare path anomaly detection algorithm.
- We validate our algorithms on two large-scale datasets with known lateral movement collected on operational networks.

II. System Design

We now describe *Latte*, our lateral movement detection system which is illustrated in Figure 3. System and security events are used to construct the network connection graph and detect potential remote file executions. *Latte* does not process the events stored on the local machine. To reduce the likelihood of log tampering on the host by the attackers, these events are instead forwarded to Windows Event Forwarding

(WEF) servers and stored in separate logs in a MapReduce file system. The system and security events are analyzed to identify potential Remote File Executions (RFEs) which can be a key component of lateral movement. A detailed discussion of the Remote File Execution Detector is presented later in Section III. From the network connection graph, we propose a path-rate score which provides a measure of how rare (*i.e.*, anomalous) a path is in a computer network. Given a list of one or more compromised computers or accounts, this path-rate score is used to assist analysts in discovering additional compromised nodes in the *Forensic Analysis Module*. In the absence of any known detections, the *General Detection Module* combines the output of the *Remote File Execution Detector* and the path-rate score to discover new compromised nodes. Ranking the outputs from the *Forensic Analysis Module* or the *General Detection Module* can be used to aid analysts in the discovery of unknown compromised nodes. In addition, the results of the *General Detection Module* can lead to automatic disabling of compromised user accounts or computers. *Latte* is fully implemented on a MapReduce platform to efficiently process the large-scale system and security event inputs. We next discuss the details of several of the high-level blocks in the figure in the remainder of this section.

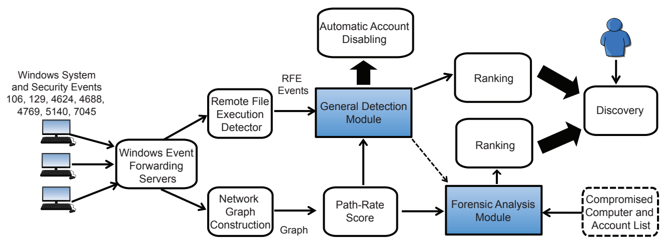
Automatic Account Disabling
Windows System and Security Events 106, 129, 4624, 4688, 4769, 5140, 7045
RFE Events
Remote File Execution Detector
General Detection Module
Ranking
Discovery
Windows Event Forwarding Servers
Ranking
Network Graph Construction
Graph
Path-Rate Score
Forensic Analysis Module
Compromised Computer and Account List

Fig. 3. *Latte* system overview. Each box with a solid outline represents a high-level block, while the two boxes with a blue background color represent our two detection modules. The box with the dashed boundary is a required input feature for the forensic analysis module. The thick black arrows correspond to system outputs.

Network Graph Construction. The network connection graph is $G = (V, E)$ where V denotes the set of accounts or computers $(v_1, v_2, ...)$, called *nodes*. E is the set of connection relationships $(e_1, e_2, ...)$ between pairs of nodes, called *edges*, constructed from the Kerberos Service Ticket Request events (*e.g.*, 4769 Windows security events). Each Kerberos record includes a source node, a destination node, and a timestamp indicating when this connection occurred. Since there is an order between the connection of a pair of nodes, we employ directed edges in the model. We then model lateral movement as a *path* which is a sequence of edges connecting a set of nodes, *e.g.*, $v_1, e_1, v_2, e_2, ...e_K, v_{K+1}$. Since the number of edges on this path is K, this path is called a *K-hop* path.

The main purpose of *Latte* is to identify malicious lateral movement subgraphs in the overall network connection graph. Lateral movement paths can take several forms including linear paths, directed acyclic graphs (DAGs), or cyclic graphs. To facilitate accurate and efficient processing, we first search for malicious subpaths (*i.e.*, rare K-hop paths), and then construct the overall malicious graph by joining these subpaths together based on an exact match of the source node, destination node, and timestamp. It should be noted that since the timestamps are required to reconstruct the malicious lateral movement graph, we cannot aggregate multiple paths to reduce the scale of the collected data.

Thus, one input parameter to the system is K, the desired subpath length. From experiments in Section V, we show that the number of K-hop paths in the network increases exponentially as K increases. From our experience, longer subpaths with length $K \geq 3$ miss shorter lateral movement activity and make it more difficult to identify malicious lateral movement paths. As a result, we set $K = 2$. *Latte* also generates all results for 1-hop paths in case the attacker fails to make a second movement to a third node.

Some servers located on the network contribute an extremely large number of connections. In our data, the computers with such a high indegree (inbound) or outdegree (outbound) are domain controllers (DCs), AppLocker servers and WEF servers. To prune the search space further in the general detection scenario, we introduce an *optional* filtering mechanism, *Node Filtering*, to remove nodes with an inbound or outbound which is greater than or equal to the threshold F. *Node Filtering* is not required in the forensic analysis case because the graph is already filtered based on the known compromised node. It can be disabled in the general detection scenario but requires more time to generate and process the connection graph. It should be noted that by including *Node Filtering*, *Latte* will fail to detect any rare connections to or from the high degree node. In our experiments, setting $F = 10000$ is a good compromise value between reducing the size of the graph and the false negative rate.

Path-Rate Score. To facilitate the discovery of rare paths in the network connection graph, *Latte* computes a path-rate score which represents the probability of occurrence of the entire path over some period of time. We first assign a weight to each directed edge, $w(v, v') = \frac{C(v,v')}{X}$, which represents the probability of a daily connection from v to v' over an X-day history of data, where $C(v, v')$ denotes the number of days v connects to v' during a period of X days. Ideally, X will be as large as possible to search for activity in the distant past. In practice, X is chosen based on the costs associated with storing the connection logs and the computation time required to construct and process the graph.

The path-rate score, p, which represents to probability of a K-hop path, is then calculated by the multiplication of the weights of all of its constituent edges:

$$p = w(v_1, e_1, v_2, e_2, ...e_K, v_{K+1}) = \Pi_{i=1}^{K} w(v_i, v_{i+1}). \qquad (1)$$

For mathematical simplicity, we compute the path-rate score in (1) assuming the edges are independent. An idea which is similar to the path-rate score was previously proposed in [8].

Since paths are intended to model lateral movement across nodes within a network, we add a time constraint to filter out impossible paths. Specifically, we require edges to be created in sequence with $time(e_i) < time(e_{i+1})$ for i = 1,2, $\cdots$, k-1 where $time(e_i)$ denotes the timestamp when edge e_i is created.

Given enough computational and storage resources, no additional constraints are required. However even after eliminating impossible paths, the graph can still be extremely large.

To further prune the graph, *Latte* includes an *optional* time constraint for each pair of edges. This constraint is motivated by discussions with analysts who believe that attackers do not typically remain active on the network for an extended period of time to avoid detection. Formally, this constraint requires that each pair of edges in a path must be created within a certain period of time, $|time(e_i) - time(e_{i-1})| < T$, where $|\cdot|$ represents the absolute value, and T is a user-defined input threshold representing the maximum amount of time the attacker uses to make a pair of consecutive lateral movement hops on the network. Since T only constrains a pair of edges, this model is able to capture much longer lateral movement activity within a single session. For example, consider 2-hop paths $e1 \longrightarrow e2$ and $e2 \longrightarrow e3$. If $e1 \longrightarrow e2$ is determined to be malicious, $e2 \longrightarrow e3$ can quickly be evaluated based on an exact match of $e2$.

Varying T is a tradeoff between potential false positives, false negatives, and the number of results returned by *Latte*. When T is small, the filtered subgraph is small and will only detect quick lateral movements. However, *Latte* will miss any lateral movement (*i.e.*, a false negative) if the attacker introduces large delays between hops. On the other hand, using large values of T results in a large number of potential lateral movement paths which can lead to increased false positives.

Forensic Analysis Module. The forensic analysis scenario assumes we have a list of compromised nodes as an additional input feature, which contains at least one known compromised computer or user account. The goal in forensic analysis is to quickly identify all malicious lateral movement paths into and out of each compromised node. To do so, we set this infected node as either the starting or ending node in the 2-hop path, and apply the path-rate score to generate all possible outbound or inbound paths. Because we only consider a single malicious node, the majority of the rare 2-hop paths are filtered out revealing an extremely limited number of suspicious paths to investigate manually. Once we confirm a malicious lateral movement subgraph, we add the two newly discovered nodes to the compromised node list. When there are multiple infected nodes in the list, *Latte* repeats the process for each additional node. In some cases, these individually confirmed subpaths can be combined to reveal the entire malicious lateral movement graph.

General Detection Module. For the general detection problem, the task is to initially detect malicious lateral movement without the knowledge of any previously discovered compromised nodes. Initially, we tried to address the general detection problem by simply ranking the path-rate score. However, this approach generates too many rare, but legitimate, connection paths to discover malicious activity.

Instead, we first analyze the event logs for signs of remote file execution, a key component of lateral movement. While the *Remote File Execution Detector* generates far fewer false positives, it still requires significant manual analysis. Furthermore, identifying remote file execution alone does not identify the potential lateral movement paths. To overcome these issues, the *General Detection Module* searches for lateral movement using the path-rate score in combination with possible remote file execution detections on a network. The core idea is that, by combining these remote file execution detections with the rare connection path detections, we can increase the probability of detecting malicious lateral movement without generating a large number of false positives. There can be a feedback loop from the general detection module to the forensic analysis module to further analyze potential malicious lateral movements.

Implementation. Given the large-scale datasets, *Latte* cannot be implemented using standard techniques and efficiently executed on a single computer. Since there are millions of nodes and hundreds of millions of edges, we implement all of our algorithms in a large cluster using Microsoft's Cosmos MapReduce framework. This framework supports a SQL-like syntax, and includes a distributed storage component. After parsing the users' input code, it generates a parallel, optimized "execution plan" for the defined queries. Latte requires 44 minutes to build the Network Connection Graph and Compute the Path-Rate Scores, 31 minutes to execute the Remote File Execution Detector, and 3 minutes to correlate the RFE Detector and Path-Rate Score.

III. Remote File Execution Detection

We next discuss how we combine the Windows system and security events to detect possible remote file execution. Remote file execution occurs when a user on one computer runs a program on a second computer and is often used by attackers during lateral movement. The PsExec tool is commonly used by IT administrators on Windows computer networks to execute a file on a remote computer. An account that has administrative privileges on the remote computer - typically a domain administrator - logs on and mounts the ADMIN$ share, writes an executable to this share or one of its subdirectories, remotely installs a service pointing to the executable, and remotely starts the service. The result is the remote execution of a file performing the configured actions of the IT administrator. While mechanisms such as the PsExec example exist to facilitate remote administration of computers on a domain, this and similar techniques are also used by attackers to achieve lateral movement on a computer network. Once an attacker has obtained the credentials to a privileged account, such as a domain administrator, Remote File Execution (RFE) can be used to infect other computers on the network at will.

Classes of RFE extend beyond service installation. Misuse of the remote registration of scheduled tasks, the WMI Win32 Process class and the Windows Remote Management (WinRM) have also been observed in the pursuit of lateral movement.

Remote service installations or task registrations are immediately preceded by a network logon from the user performing the action because of the associated RPC endpoint interaction. Additionally, when the service or task is started following the installation, we may observe a process creation related to the service or task that was installed. At the core of this detection technique are the service installation (7045) and task registration (106) events. From these events, we can correlate backward-in-time with the logon event to identify whether there were any network logon success events (4624) by the user who installed the service or task. Finding a match between these within a short time-frame is highly correlated with a remote service installation or task registration. To bring further context to these events, we then extend the correlation by looking for a process creation event (4688) and any share access event (5140) that may be associated with this activity. In summary, we are looking for a combination of the following: a remote logon, a service installation or task registration resulting from that logon, a process creation event of the executable pointed to by the service or task, and optionally, an ADMIN$ share access attempt.

Logon Correlation. For a remote service installation event (7045), the Event/System/Security@UserID contains the security identifier (SID) of the user account that installed the service. We search backward-in-time from the installation event to find a matching Network Logon Success (4624) event where the TargetUserSid values match. In the case of a task registration, we are provided with the user account that registered the task in the UserContext field of the event. We use this information in a similar way to trace the registration back to any remote network logon success event for that user by correlating UserContext with the TargetUserName and TargetDomainName fields.

Task Correlation. In the case of the task registration we are left with some extra work to do as the path to the executable is not provided in this event. The Event/System/Correlation@ActivityID field in the XML provides a robust mechanism to find any associated "Task Created Process" (106) events for this task. Performing a forward-in-time correlation on the ActivityID, or where this is not available the TaskName, gives us a reliable method of obtaining the executable's path.

Forward Process Execution Correlation. Once we have identified where a remote task or service installation has taken place, we can then correlate this forward-in-time with any associated process execution (4688) events. Given the name, path and ID of the process from the service or task data, we can look for the first process execution event that matches this criterion. Additionally, restricting the user and logon ID associated with the process execution to be a system session with a logon ID of '0x3e7' in the event indicates that it is a service installation.

Share Access Correlation. Between the time of the remote logon and the service or task installation, we also look for a Share Access event (5140) for the \\*\ADMIN$ share. We use this as evidence that a file may have been written remotely at the time of its execution. The Subject user details in the 5140 event correlates with the Target user details from the 4624 logon event, in addition to the IP address that is the source of the share access and logon. The result of this process is a high fidelity detection of a remote service or task installation, enriched with the full logon event, resulting process execution and, if available, the ADMIN$ share access used to write the file that is executed. This refined RFE data can be used to detect lateral movement on a computer network by identifying rare or uncommon instances, as well as provide a rich, valuable data source for analysts and incident responders.

IV. Datasets

We utilize two datasets in this study to analyze the proposed *Latte* system. Both datasets consist of Windows security and system events plus labels from analysts for malicious activity.

An anonymized organization provided us with the first, *Attack Dataset* containing a confirmed attack exhibiting lateral movement among a collection of four computers in a large network. This dataset only contains the Kerberos Service Ticket Request (4769) event logs with a total of 1,190,639 active nodes and 250,614,631 connections (edges) collected over a 90-day period leading up to the detected attack. Since this dataset contains the connection events from the 4769 Kerberos Service Ticket request logs, it enables us to only evaluate the *Forensic Analysis Module*.

The second, *Penetration Test Dataset* was generated by Microsoft and includes an attack with lateral movement conducted by a penetration tester. This dataset includes all of the Windows events found in Section III. As a result, this dataset allows us to evaluate the performance of *Latte's General Detection Module*. The penetration tester was not aware of how *Latte* detects lateral movement. For a six-month period leading up to the penetration test, the number of source nodes is 3,412,030, and the minimum, maximum, and mean out degree is 1, 1,631,308, and 336, respectively. Similarly, the minimum, maximum, and mean in degree is 1, 3,273,616, and 346, respectively.

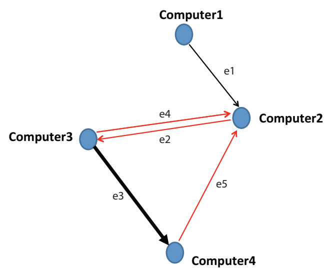

Fig. 4. The lateral movement of a real-world attack. The arrows indicate the connection direction. The red arrows correspond to a one-time connection, the thin black arrow indicates there were several connections before the attack day, and the bold black arrow means there was a regular connection pattern before the attack day. Connections are ranked in chronological order.

Target Paths	$e1 \longrightarrow e2$		$e2 \longrightarrow e3$	
Path Direction	Outbound from Computer1	Inbound to Computer3	Outbound from Computer2	Inbound to Computer4
# of all possible paths	1,467,837	9,243,802	1,476,721	10,009,740
# of paths within time constraints	22,793	447,828	28,219	566,693
Ranking of the target path	1	1	15	4
# of paths tied for the same rank	1	11	2	4
Top-1 Precision / Recall	1.0 / 1.0	Multiple Ties	Multiple Ties	0.0 / 0.0
Top-5 Precision / Recall	0.2 / 1.0	Multiple Ties	0.25 / 0.5	Multiple Ties
Top-25 Precision / Recall	0.04 / 1.0	0.04 / 1.0	0.08 / 1.0	0.25 / 1.0

TABLE I
Ranking of the target paths (lateral movements by compromised nodes) when different source nodes are processed by the *Forensic Analysis Module*.

V. Forensic Analysis Results

In this section, we evaluate the performance of *Latte* for the forensic analysis scenario. We test our system on one confirmed attack and one penetration test attack initiated by a professional tester on a very large computer network.

Path Direction	Outbound from Computer1	Inbound to Computer3
# of paths within time constraints	16,584,643	42,079,507
Ranking of the target path	1	1
# of paths tied for the same rank	10	125
Top-10 Precision / Recall	0.1 / 1.0	Multiple Ties

TABLE II
RANKING OF THE TARGET PATHS IN THE FORENSIC ANALYSIS SCENARIO FOR DETECTING LATERAL MOVEMENT DURING A PENETRATION TEST WHEN THE *Forensic Analysis Module* IS APPLIED TO DIFFERENT REFERENCE NODES.

Target Paths	$e2 \longrightarrow e3$	$e4 \longrightarrow e3$	$e1 \longrightarrow e5$	$e1 \longrightarrow e6$
Day	2	2	3	3
# of paths within time constraints	264,581,288	264,581,288	608,156,343	608,156,343
Path-rate score	1.2345679e-4	1.2345679e-4	3.7037037e-4	3.7037037e-4
Ranking of the target path	1	1	219,749,246	219,749,246
# of paths tied for the same rank	44,087,827	44,087,827	15,211,115	15,211,115

TABLE III
RANKING OF TARGET PATHS (LATERAL MOVEMENTS BY SUSPICIOUS NODES) BY THE PATH-RATE SCORE FOR DIFFERENT MALICIOUS PATHS.

System Setup. We first describe the parameter settings we use for *Latte* both for forensic analysis and, later, for general detection. For all experiments, we set $X = 90$ days which is the maximum duration of one of our datasets.

Path Length. We analyzed how many K-hop paths can be constructed for different values of K for the *Attack DataSet*, and found 3,162,870, 70,242,941, 1,642,110,520, and 51,625,886,182 paths for K ranging from 1 to 4, respectively. The number of paths increases exponentially with a factor ranging from 22 to 31 as K increases. For all values of K, we apply aggressive node filtering with an inbound or outbound threshold of $F = 1{,}000$. This low threshold value was required in order to compute these results for $K = 4$ hops. The table indicates that there are 3,162,870 1-hop paths. This number is significantly less than the 250 million edges reported for this dataset in Section IV because of this aggressive filtering. There are clearly too many 4-hop paths for consideration; therefore, we exclude all 4-hop paths from our remaining experiments.

Time Constraint. Unless otherwise specified, we include the *optional*, consecutive hop time constraint and set $T = 2$ hours to reduce the computational and storage burdens. While we set the threshold between two successive movements (moving from one node to another) to be 2 hours, it is possible that attackers may slow their lateral movement to escape detection. However, slowing the lateral movement also greatly increases the risks of being caught, since attackers are forced to remain on the network much longer.

Based on these results, we only consider 2-hop paths for *Latte*, *i.e.*, paths with three nodes (computers or accounts) and two edges (a connection from one node to another). In general, we find that using 2-hop paths leads to good detection rates, and this parameter setting is much more efficient than running *Latte* using path lengths with $K \geq 3$.

Case Study 1: Confirmed Lateral Movement Attack. We first analyze the *Attack Dataset* using the *Forensic Analysis Module* for the forensic analysis case. As depicted in Figure 4, the cybercriminal first logged on to Computer1, *i.e.*, a computer in a meeting room, then to another computer, *i.e.*, Computer2, in a different meeting room. The attacker next moves to Computer3, and finally to Computer4. Afterwards, the attacker explores the network by jumping back and forth among the four compromised computers. The entire lateral movement process lasts for 1 hour and 8 minutes.

For the two, 2-hop paths involving malicious connections, we apply the *Forensic Analysis Module* to the two reference nodes separately, and provide the results in Table I. We include both the outbound and inbound rankings for the same 2-hop path to investigate the effects of whether the first or the last node was identified to be malicious. For example, when it is an outbound path for the 2-hop path from $e1$ to $e2$, the analyst would be told that Computer1 is the malicious node. Similarly, Computer3 would be the identified as the malicious node for the inbound path results. We have several key observations. First, after adding the time constraints in the second row, the total number of possible paths is reduced by more than 90% compared to results in the first row without the time constraint. Second, the outbound and inbound rankings for the same 2-hop path show that the number of possible connections can vary significantly depending on the malicious node. Third, our *Forensic Analysis Module* generates good ranking results for different paths and reference nodes, where all malicious paths rank at or near the top out of millions of possible paths. Unlike typical detection tasks, forensics analysis is a ranking task, and the goal is for the malicious path to be included near the top of the ranked results. Multiple 2-hop paths may have the same path-rate score, and are therefore tied for the same rank. We include the number of tied paths to indicate that the analysts only need to manually investigate a small number of potential malicious outbound or inbound paths. For this dataset, the 2-hop path from $e1$ to $e2$ is either first or tied for the first in both path directions. Also, the outbound, 2-hop path from $e2$ to $e3$ is ranked 15th and another path shares its path-rate score. This data indicates that 14 paths have a lower path-rate score, some of which may be tied. Since this is a ranking task, we include the Top-K precision and recall results. Finally, the *Forensic Analysis Module* is robust even when there are regular connections between two computers ($e3$), and our model can still promote the malicious paths to the top of the ranked list. This is important because it indicates that both edges in the 2-hop path do not necessarily need to have the lowest possible path-rate score to be identified. Note that in the case of identifying the outbound path from $e2$ to $e3$, the target malicious path is not ranked first. However, we did find that the top ranked path is a loop from $e2$ to $e4$, which also involves malicious lateral movement. These results validate the effectiveness of our method in the forensic analysis scenario.

VI. General Detection Results

To investigate whether *Latte* can address the general detection task at large-scale, we conducted several experiments on the *Penetration Test Dataset*. This dataset contains all the events noted in Section III collected during a penetration test on a large-scale network conducted by a professional tester. The lateral movement graph from the penetration test is depicted in Figure 5, and the lateral movement activity involves one user account and four computers. The first connection, $e1$, was made on the first day and the third day. The connections represented by $e2$, $e3$ and $e4$ occurred on the second day while the $e5$ and $e6$ connections were generated on the third day. We first evaluate the detection results of the graph's 2-hop path results ranked by the path-rate score and *Remote File Execution Detector* in isolation. Then we present *Latte's* results, which processes all the events in the full dataset. While we did not use the optional *Node Filtering* for forensic analysis, we do include it for these general detection experiments, with a connection threshold $F = 10000$, to prune the graph.

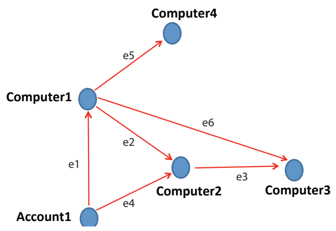

Fig. 5. Connection graph for the *Penetration Test Dataset*.

Case Study 2: Path-Rate Score Ranking and RFE Detections in Isolation. First, we consider graph-based detection by only ranking the path-rate score, and the results are summarized in Table III. After filtering all potential paths as described in Section II, this dataset has 264 million, 2-hop paths on Day 2 and 608 million, 2-hop paths on Day 3. Among them, 88,175,654 paths are tied on Day 2 for the first rank and share the same minimum path-rate score, $1/X^2 = 1.235e-4$, since both edges on the path have just one connection during the observation window ($X = 90$ days).

The results from this experiment indicate that the path-rate score in isolation is not a good candidate for general detection. One reason is that many of the paths are tied for the top rank. While we found that the penetration tester's movements were rare in general, many other paths on the network were equally rare or even more rare. Even though one 2-hop path was ranked first, it was tied with millions of other paths with the same path-rate score. Therefore, we conclude that only ranking the path-rate score does not have high enough precision to be used for general detection.

Next, we report the performance of the *Remote File Execution Detector*. The *Remote File Execution* detections on the network significantly reduce the amount of raw data that analysts need to manually inspect to detect the penetration testers. As shown in Table IV, there were 45 alerts of possible remote file execution on the network on the second day of the exercise, and two of these were due to the penetration tester's activity on nodes Computer3 and Computer4 in Figure 5. There were no indications that any of the other alerts were due to malicious lateral movement activity and are considered to be false positives. It is worth noting that the attack graph in Figure 4 does not include any user nodes that the analysts could identify. Presumably this is because the attacker had already installed a backdoor on the computer sometime in the past. This result demonstrates that the *Remote File Execution Detector* can be used for general detection with daily manual inspection by analysts.

Metric	Value
# of potential remote file execution events generated in the second day	45
Ranking of malicious path $e2 \longrightarrow e3$	1
Ranking of malicious path $e4 \longrightarrow e3$	1
# of paths tied for the top rank which include a node with an RFE detection	2
Combined Top-1 Precision / Recall	1.0 / 0.2
Combined Top-5 Precision / Recall	0.4 / 0.4

TABLE IV
RANKING OF THE TARGET PATHS BY THE *General Detection Module* FOR THE PENETRATION TEST WITH ALL THE WINDOWS SYSTEMS AND SECURITY EVENTS FOUND IN SECTION III.

Case Study 3: General Detection Results Using All the Data. While the previous results indicate that the RFE detector can be used to identify malicious remote file execution with manual inspection, there are still too many false positives to use it for automatic account disabling. In addition, analysts often tire from doing repetitive manual analysis day after day without discovering malicious activity. For the twenty days of logs preceding and including the lateral movement in Figure 5, 777 *Remote File Execution* events were generated on the network. To further improve the system, we next investigate combining the RFE detection results with the potential lateral movement graph in the *General Detection Module*.

We correlated all the RFE events on Day 2 with *Latte's* daily, network connection graph. A summary of the results for this experiment is provided in Table IV. Of the 45 potential RFE events that were generated on the second day, we manually inspected all the 2-hop paths which included any of these nodes. As noted in Table III, we found that the two, 2-hop paths ($e2 \longrightarrow e3$, $e4 \longrightarrow e3$) with lateral movement had the smallest path-rate score — *i.e.*, they ranked at the top of list. In addition, these malicious 2-hop paths were the only two items that were tied with the smallest path-rate score; none of the 2-hop paths involving the other RFE false positive nodes had the smallest path-rate score. Although not conclusive, this exercise provides evidence that it is possible to automatically discover a detection by intersecting rare RFE events with rare paths in *Latte's* connection graph. In other words, this experiment demonstrates that we need to combine the RFE detection with the path-rate model to overcome the high false positive rate for the RFE detector. RFE detection alone is not sufficient.

VII. Related Work

Lateral Movement Detection. The detection of lateral movement is an understudied problem, although a few papers have explored the topic. Neil, et al. [8] propose a graph-based model to detect anomalous paths in a graph. An anomaly score based on the connection's p-value is first learned. Then a hidden Markov model is used to identify anomalous paths. To make our system scalable, we instead use probabilities to construct the path score. Also, we add the RFE detector to more effectively detect intrusions. We tested our system on actual attack data and red team activity with confirmed lateral movement, where Neil, *et al.* tested their algorithm on simulated and network data. No attacks were confirmed in 38 detections in their network data during a 20 day period. Authentication data is modeled in [9] using a bipartite graph, and the authors measure the risk of credential hopping by computing the largest connected component of this graph. In [1], Purvine et al. assume each computer has a vulnerability state, construct a reachability graph of the network, and then define an impact metric to monitor the network's security. The authors then experiment on datasets with artificially injected vulnerabilities. In [10], the authors propose a defense-based zero-sum game to prevent, or at least slow down, an attacker from reaching a target node on a computer network. Fawaz, et al.[7] propose a hierarchical system to detect lateral movement on a graph. The lowest-level agents in the system propose subgraphs for each host of potential lateral movement. A central agent then combines these local subgraphs across hosts to recover potential lateral movement paths. Therefore, this system requires that a custom agent be installed on each host. *Latte*, on the other hand, only requires standard system and security events already included in recent releases of the Windows operating system. In addition this work does not specifically address the forensic or general detection scenarios and does not detect remote file execution. Soria-Machado, et al. [11] suggest rules for possibly detecting pass-the-hash attacks for the NTLM protocol and pass-the-ticket attacks for the Kerberos protocol based on system and security events on Windows Vista, Windows 7, and Windows Server 2008 environments. These rules attempt to identify potential lateral movement as anomalous connections to high value computers (*e.g.*, domain controllers).

Cybersecurity. Several graph-based detection algorithms have been proposed for computer security, and many of them focus on detecting certain network vulnerabilities. Heat-ray [2] tries to determine potential configuration changes in a network regarding computers and user accounts as nodes, and the privileges of source nodes on destination nodes as edges. Attack graphs are generated in [3] where nodes represent possible attack states while edges represent changes of state.

Recently, more studies focus on malware detection on real large-scale data, which targets the security problem in large enterprises. Examples include [12], [13], [14], [15], [16], [17].

VIII. Conclusions

In this paper, we propose *Latte*, a new graph-based system to discover malicious lateral movement paths in a compromised computer network. The network connection graph is constructed from Windows security events generated by Kerberos service ticket requests. Our findings suggest the effectiveness and efficiency of graph-based algorithms in detecting lateral movement. For the forensic analysis problem, we show that using the network connection graph processed with the path-rate score and the *Forensic Analysis Module* is sufficient for generating the lateral movement graph with the detection of one or more confirmed malicious computers or user accounts. For the general detection problem, relying on the path-rate score alone leads to too many potential, but legitimate, rare paths for manual investigation by analysts. To overcome this problem, we propose the *General Detection Module* which combines the path-rate score with possible *Remote File Execution* detections to filter out most of the benign lateral-movement paths. We demonstrate this idea by using a remote file execution detector which correlates six additional Windows system and security events caused by the installation of a new service or service, or the creation of a new process. These new tools for detecting lateral movement activity, as well as the initial detection of a compromised node, are very helpful for analysts in combatting this extremely challenging problem.

References

[1] E. Purvine, J. R. Johnson, and C. Lo, "A graph-based impact metric for mitigating lateral movement cyber attacks," in *Proceedings of the ACM Workshop on Automated Decision Making for Active Cyber Defense (SafeConfig)*, October 2016.

[2] J. Dunagan, A. X. Zheng, and D. R. Simon, "Heat-ray: combating identity snowball attacks using machinelearning, combinatorial optimization and attack graphs," in *Proceedings of the 22nd symposium on Operating systems principles (SOSP)*, October 2009.

[3] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in *Proceedings of DARPA Information Survivability Conference &amp; Exposition II*, 2001.

[4] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[5] C. C. Noble and D. J. Cook, "Graph-based anomaly detection," in *Proceedings of the 9th International Conference on Knowledge discovery and data mining (KDD)*, August 2003.

[6] M. I. of Technology, "Kerberos: The network authentication protocol," https://web.mit.edu/kerberos/, Last accessed February 16, 2017.

[7] A. Fawaz, A. Boharay, C. Chehy, and W. H. Sanders, "Lateral movement detection using distributed data fusion," in *Proceedings of IEEE 35th Symposium on Reliable Distributed Systems (SRDS)*, 2016.

[8] J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie, "Scan statistics for the online detection of locally anomalous subgraphs," *Technometrics*, pp. 403–414, 2013.

[9] A. Hagberg, N. Lemons, A. Kent, and J. Neil, "Connected components and credential hopping in authentication graphs," in *Proceedings of the 10th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS)*, November 2014.

[10] M. A. Noureddine, A. Fawaz, W. H. Sanders, and T. Başar, "International conference on decision and game theory for security," in *Proceedings of International Conference on Decision and Game Theory for Security (GameSec)*, 2016, pp. 294–313.

[11] M. Soria-Machado, D. Abolins, C. Boldea, and K. Socha, "Detecting lateral movements in windows infrastructure," http://cert.europa.eu/static/WhitePapers/CERT-EU_SWP_17-002_Lateral_Movements.pdf, Last accessed August 7, 2017.

[12] 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 Annual Computer Security Applications Conference (ACSAC)*, 2013.

[13] T.-F. Yen, V. Heorhiadi, A. Oprea, M. K. Reiter, and A. Juels, "An epidemiological study of malware encounters in a large enterprise," in *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*, 2014.

[14] A. Oprea, Z. Li, T.-F. Yen, S. H. Chin, and S. Alrwais, "Detection of early-stage enterprise infection by mining large-scale log data," in *Procedings of the 45th International Conference on Dependable Systems and Networks (DSN)*, 2015.

[15] S. T. King, Z. M. Mao, D. G. Lucchetti, and P. M. Chen, "Enriching intrusion alerts through multi-host causality," in *Proceedings of the Network and Distributed Systems Security Symposium*, 2005.

[16] J. Rennie and A. McCallum, "Using reinforcement learning to spider the web efficiently," in *Proceedings of the International Conference on Machine Learning*, 1999.

[17] J. Cho, H. Garcia-Molina, and L. Page, "Efficient crawling through url ordering," in *Proceedings of the Seventh International Conference on World Wide Web*, 1998.