Combining Dynamic and Static Attack Information for Attack Tracing and Event Correlation

Conference Paper · August 2018

CITATIONS
0

READS
24

1 author:

Ankur Chowdhary
Arizona State University
25 PUBLICATIONS **74** CITATIONS

SEE PROFILE

Combining Dynamic and Static Attack Information for Attack Tracing and Event Correlation

Adel Alshamrani, Ankur Chowdhary, Oussama Mjihil, Sowmya Myneni, Dijiang Huang
Arizona State University
Email: adel.alshamrani, ankur.chaudhary, oussama.mjihil, sowmya.myneni, dijiang.huang@asu.edu

***Abstract*—Many sophisticated attacks, e.g. Advanced Persistent Threats (APTs), have emerged with a variety of different attack forms. APT employs a wide range of sophisticated reconnaissance and information-gathering tools, as well as attack tools and methods. Thus, in this paper, we design a solution that is based on multi-source data combination to learn the adversarial behavior of suspicious users as well as to optimally select a proper countermeasure.**

***Index Terms*—Advanced Persistent Threats, Intrusion Detection Systems, Attack Graph**

I. Introduction

Nowadays, attacks are not only increasing in number; the way how they are launched is based on sophisticated and complex methods (e.g., APTs), which indicates that these attacks may be supported by large organizations and in some cases by foreign governments [1]. Distinguishing between APT and legitimate activities is very difficult and can result in difficulty in detecting hidden attackers in an organization. According to [2], attackers were able to remain hidden for more than a month after the attackers first gained access to the system by stealing credentials from an HVAC and refrigeration company, Fazio Mechanical Services, through stealing Fazio Mechanical credentials via infected emails with malware.

Existing deployed security systems mostly are optimized for processing large amount of system data (e.g., event logs) and are therefore highly automated. This makes the detection of sophisticated, intelligent, and complex attacks difficult. Although these systems are able to identify individual characteristics of an attack, it is necessary to perform an additional investigation of complex attacks to reveal important hidden details in the system that can lead to detecting intelligent malicious activities. Therefore, our purpose is to design a solution that involves multiple components, such as audit logs, system vulnerability information, network configurations, machine configurations, intrusion detection alerts, etc, each with the ability to derive important observations that, when combined, can lead to detecting, tracing, locating, and optimally responding to potential APT attacks. These components look for parts of attack sequences that are visible within their own data.

In this paper, we utilize the following points: *First*, investigate the system events, that can be obtained from different logs, to learn about the users' activities currently and in the past. *Log events* can provide the recorded activities of users and systems that can be used together with the initial indicators (e.g., IP addresses or user names) to trace malicious activities and to identify the different steps of an attack [3]. The analysis of log events can be a time-consuming and complex process, since to ensure an effective analysis of events, this process needs to be iteratively done over time. However, to overcome this issue, it is possible to analyze only necessary and relevant log events for an indication of abnormal activity in our system. *Second*, represent the system's current vulnerabilities and links between them in an attack graph. Each path in the attack graph is considered an APT scenario. The more vulnerabilities in the attack graph, the more multiple APT scenarios can be created, in which each represents a path from the root of the attack graph downstream to the target node. Therefore, we believe that investigating log events to construct correlated attack activities (evidence) and mapping that constructed evidence to the available information in the attack graph will enhance our attack detection system.

The contributions of this paper can be summarized as follows. First, we analyze system information to detect low frequency attack activities "stealthy activities". Second, we correlate detected stealthy activities (dynamic information) with the system configuration vulnerability (static information). Third, we extend the representation of attack graph to include dynamic information and reduce the current complexity of using the attack graph to support security risk assessment. Fourth, we design a real time security countermeasure methodology.

The rest of the paper is organized as follows. Section III describes our approach. A test case and evaluation of this work are described in Section IV and Section V, respectively. The related work is presented in Section II. Finally, Section VI concludes this paper.

II. Related Work

Most of the existing work only describe, analyze, and discuss popular and publicly disclosed APT cases, such as Stuxnet [4], Duqu [5] and Flame [6]. These studies do not discuss solutions for automatic detection of APTs [7]. Unlike many works that have been surveyed in [8], [7], where proposed approaches correlate attack events and then generate evidence graph based on those attack events only, our work is different since we, in advance, correlate existing system vulnerabilities with possible and suspicious attack events to extract all possible attack paths (*APT scenarios*). In other words, most of the evidence graph-based intrusion detection approaches create evidence graph through identifying the

attack events which we believe they can give only limited view of attacker's behavior.

Authors in [9] propose a framework that is based on correlating alerts together when the attackers launch attacks to prepare for other attacks later. This approach can help to detect the prerequisites of the coming attacks but not with other possible attacks can be launched but no sign has been shown yet. Brogi et al. [10] design an approach for detecting long lived attack campaign APT. They claim they can trace series of attacks that are consequential attempted. However, they did not show the effectiveness of their work through an evaluation. Many other approaches have been discussed in literature [11], [12], [13], [14]. However, they mainly focus either on delegating hosts and vulnerabilities risk analysis or on the number and types of vulnerabilities and how they affect the probability of attack success.

III. Our Approach

A. Threat Model

We consider covert attackers who have the motivation to targeting enterprise networks with the goal to deliver malware to vulnerable people and systems through social engineering technique. We assume that he gets useful information about the organization's employees and what websites they heavily visit. The attacker then can start checking those preferred websites and inject malicious contents to a vulnerable one, this is known as *"watering hole attack"*. Once the targeted user visits the infected website, the malicious code is loaded and the system got compromised. Now, the attacker has gained an access in the system through the malware, and then can access more sensitive components such as servers. Finally, the targeted system's data will be exfiltrated. The presence of the malware on the compromised host can change its behavior by running unexpected processes, requesting unauthorized services, scanning the network, or attempting to authenticate to other hosts. Therefore, we can consider these suspicious activities as triggers to investigate the existence of APT attack in our system.

B. Approach Design

Our approach consists of multiple components, as shown in Figure 1, and each component provides useful information about the system and users activities to the APT Evidence Unit (APTEU). Consequently, the APTEU makes a decision if the existed vulnerability is unexplored, explored, or exploited. Based on the decision output, the countermeasure selection module selects the optimal countermeasure to mitigate the APT attack. Our approach mainly consists of four phases: a) *Event phase* which is responsible for network traffic collection, processing, and extracting the attack events. It continuously feeds attack events information to the second phase (b); b) *Vulnerability phase* which checks the current vulnerabilities in the system and related suspicious events and then visualizing them in a graphical fashion. Also it generates the possible APT scenarios; c) *Decision phase* which is responsible for making a decision if the system vulnerabilities have been explored, or exploited and then pass the result to the countermeasure phase; and d) *Countermeasure phase* which is responsible for selecting an optimal countermeasure from a countermeasure pool. The description of our approach as depicted in Figure 1 is summarized as follows:

Step 1) Collect network traffic and system data and pass these data to the traffic processing and intrusion detection system.

Step 2) In this step, our system processes the collected data to identify abnormal and malicious activity (alert generation).

Step 3a) Extract important facts from alerts and pass them to the extended MulVAL engine. Facts can be timestamp, IP Source/Destination, port Source/Destination, and description of the malicious activity, etc.

Step 3b) Extract the correlations between malicious events and relevant logged information and also extract important facts and pass them to the extended MulVAL engine. These correlated events can share in common these features: source MAC address, destination MAC address, source IP address, and destination IP address, actions.

Step 4) Generate attack graph based on the system vulnerabilities and their connectivity information as well as the received facts from the alerts and the logs.

Step 5) Generate APT scenarios that belong to all possible paths from the root to the downstream nodes. It is possible to have multiple APT scenarios, where each represents a path from the root of the attack graph downstream to the target node.

Step 6) Pass the output of ***Step 5*** to the decision making components.

Step 7) Check the vulnerabilities against the malicious events and generate a base knowledge that can give a clear evidence if any vulnerability has a related action in the malicious events or not.

Step 8) Pass the final decision to the countermeasure phase.

Step 9) Select a countermeasure from the pool of countermeasures shown in Table I.

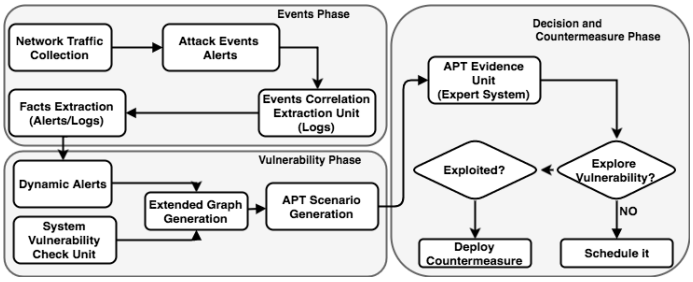


Fig. 1: The Approach Process Flow

C. Event Correlation Model

It is important noting that the primary focus of this model is oriented towards correlating the attack events among multiple event logs. However, differentiating between abnormal and normal activities (events) is a critical prerequisite that is beforehand needed to effectively correlate logged events.

Therefore, since this is not the main motivation of this paper, we assume that our system distinguishes between normal and abnormal activities. Setting ground truth sets to distinguish between normal and abnormal events can be done by a system administrator where the knowledge of the entire system helps in defining the ground truth set that shows the normal activities in the system. Information like, predefined cron job ids and their intervals, users of the system and their privileges of access, activities that are run by root, DNS requests by the Network manager, etc. Once we have these information as ground truth, we can use this to filter syslog. After filtering, we are left with logs that we define as abnormal activities. The more the system is investigated, the more information can be added to the ground truth sets. This is possible if our system can track the attacker's activities in the system overtime and extract a pattern of a malicious event. The attack events, processed by the events correlation extraction unit, are assumed to include the related information to the malicious activities (abnormal). However, it might be possible that we process normal events as falsely marked abnormal (false positive). Our system is designed to eliminate this erroneousness by checking the confidence of collected evidence, this is described later in this paper. The events correlation extraction unit collects the attack events and their correlated information from one log and correlate them with relevant information from multiple logs to reveal related suspicious activities and other information that cannot be detected by current alert systems. These attack events must share common information, for example, the identifiers of both attacker and target machines, events time dependency, attack strategy, etc.

D. Dynamic Attack Alert Correlation Model

The attack graph (AG) in it's current setting is able to identify static system vulnerabilities such as *local buffer overflow*, *remote Code Execution*, etc. Based on the interaction rules defined in MulVAL knowledge base [15], the tool is able to make inference about possible attack paths. However, the dynamic network attacks and system access control violations such as *FTP login failure*, *rhosts write to home directory* have not been incorporated in the current scope of the tool. We utilized evidence based on network events and system logs to create a dynamic attack graph which utilizes static vulnerability information and attackers lateral movement in the network over an extended period of time. We extended static attack graph (AG) to incorporate dynamic attack scenarios and extended vulnerability based AG to include APT scenarios discovered by correlation of static system vulnerabilities and logs evidence. The reasoning engine in MulVal scales well $(O(n^2))$ with the size of the network. The details of reasoning engine in MulVal, which we follow to correlate dynamic and static information, can be refereed to [15], and it was skipped due to space constraints.

E. Evidence Model

This model includes the APT evidence unit, presented in Figure 1, which is responsible for investigating all generated APT scenarios (paths) to collect the necessary information about the vulnerabilities and their pre and post conditions to acknowledge what has been explored, and/or exploited.

Definition 1. ***Intrusion Evidence*** *is a piece of information that can result in confidence of an intrusion against enterprise network. The intrusion (attack) evidence is a five-tuple* $I = (\_srcIP, \_dstIP, timestamp, weight, description)$ *where* $\_srcIP$ *is the event initiator and the* $\_dstIP$ *is the event target, timestamp is the activate and latest timestamps of the evidence, weight is the a value between [0, 1] to show the impact of evidence on the attack, for example, scan a network from an internal normal user after downloading of a suspicious software will have a high weight value, and the description is the type of abnormal event.*

Definition 2. ***Global Evidence*** *Combining multiple source of intrusion evidence results into a final global evidence graph that provides advanced information about the location of the attacker and what abnormal actions that have been taken but logged as normal activities. Whereas, in reality those actions are a sequence of steps toward launching APT attack.*

F. Countermeasure Methodology

Once the attack path is selected, the algorithm ranks the vulnerabilities according to their criticality based on the following characteristics: *a*) The severity of the vulnerability (Base Score(BS)). *b*) The structural importance of the vulnerability in the attack graph based on in and out reachability links. *c*) Attacker's distance from the target vulnerability.

Table I shows the list of possible countermeasures, that our system is able to perform for instance, and their assigned costs and complexity. Table II shows the indexed CVE ID of each vulnerabilities and its base score (BS) that we consider in our countermeasure metrics. Countermeasures are characterized by the following measures: Countermeasure Cost (CC), Countermeasure Deployment Complexity (CDC).

Definition 3. ***Countermeasure Cost (CC)*** *which is evaluated by the defender based on the countermeasure application and its consequences. This cost may depends on the resource consumption or the service availability after the application of a given countermeasure. Moreover, service dependency may increase the countermeasure's cost. Based on the previous factors, the cost will take a value ranging from 1 to 10.*

Definition 4. ***Countermeasure Deployment Complexity (CDC)*** *is the effort required from the defender to apply the countermeasure which depends on the amount of components that are involved in the application of a specific countermeasure. Also it depends on the need of the technical requirements and the skills of the staff. Based on the previous factors, the CDC will take a value ranging from 1 to 10.*

The goal of the algorithm 1 is to track, in real-time, the attack paths and vulnerabilities that should be fixed immediately and it measures also the Return Of Investment (ROI) after deploying a countermeasure. Since it is supposed that, at this

stage, we should already have the attack graph AG and the information representing the actual position of the attacker. Thus, the defender can deploy the optimal countermeasure with consideration of the previous state conditions. The benefit in the ROI function is represented by the rank index of the vulnerability. In other words, there will be more benefit, and hence, more important ROI for treating immediate and urgent vulnerabilities than others which are faraway from the attacker's position or not on the path of the attacker.

Once the values of CC, CDC, and Benefit are known, the ROI is calculated and the decision-making model can decide which countermeasure can be taken. Using the available information we can learn about possible attacks and our system is aware of the possible upcoming actions. This information can be easily predicted based on the already taken actions and current positions of the attacker in APT scenario as well as the vulnerability status: (unexplored, explored, or exploited).

TABLE I: Possible countermeasures and their costs and complexity.

No.	Countermeasure	CC	CDC
1	Traffic redirection	6	7
2	Traffic isolation	3	2
3	Block port	4	1
4	Software patch	7	6
5	Network reconfiguration	7	9
6	Service migration	6	8

TABLE II: Vulnerability analysis report.

CVE ID	Base Score	Consequences
CVE-1999-0651	7.5-High	rlogin
CVE-1999-0038	7.2-High	Buffer overflow
CVE-1999-0454	10-High	Nmap scanning
CVE-1999-0013	7.5-High	SSH remote access
CVE-1999-0547	10-High	Authentication through .rhosts file
CVE-2000-0143	4.6-Medium	sshdc service redirection
CVE-2000-0813	5-Medium	ftp connection
CVE-2008-3855	4.6-Medium	privileges escalation
CVE-2006-5270	9.3-High	Remote code execution

The following three terms (NbrHop, Degree, Complexity) respectively mean: distance to the target machine from the current location of the attacker, the value of in/out reachability links from the current location of the attacker, and complexity to exploit a vulnerability.

The real-time measurement of the attacker's progress is given by the ratio of what the attacker accomplished (exploited or explored) on a given path divided by the maximum value the path can have. Accordingly, the value of a path is reflecting the actions and steps performed by the attacker. We distinguish between two types of actions: *a*) ***Vulnerabilities exploit:*** Each vulnerability has a Base Score, obtained from online National Vulnerability Database (NVD), which determines its severity or, in other words, the reward that an attacker can gain after exploiting a vulnerability. *b*) ***Events:*** Each event on the APT path is characterized by a score, which determines its severity. We assign scores to events based on their scopes, exportability, and their impact. This metric is an adopted version of Base Score V3 [16]. Both vulnerabilities and event will be having the same scoring metric. The scope is showing whether the vulnerability/event is having a local or global influence, so if the scope changes, there will be producing a higher Base Score. The impact of a vulnerability/event is describing the quantity of damage that it can cause to the system. The exploitability is reflecting the required technical skills to exploit the vulnerability or to raise the event.

Algorithm 1 Real-time APT Tracking

Require:
APTPaths[] ▷ List of possible APT Scenarios
MaxValPath[] ▷ The maximum value of the Path
ValPath[] ▷ The Current value of the Path
function MAXIMUMATTACKPATHVALUE
  for $eachPath : APTPaths$ **do**
    for $each\ e : Path$ **do** ▷ e : Event or Vulnerability
      $MaxValPath_i = MaxValPath_i + e.BS$
    end for
  end for
end function
function REALTIMEATTACKPATHVALUE
  for $each\ event : logFile$ **do**
    For all paths containing the event
    $ValPath_i = ValPath_i + element.BS$
  end for
  $Attack - Progress[Path_i] = \frac{ValPath_i}{MaxValPath_i}$
  if $AttackProgress[Path_i] \leq threshold$ **then**
    $RealTimeAttackPathValue <>$
  else
    $CountermeasureApplication < Path_i >$
  end if
end function
function BENEFIT_FUNCTION(vul)
  $Distance \leftarrow NbrHop * Complexity$
  return $\frac{Degree*BS}{Distance}$
end function
function COUNTERMEASUREAPPLICATION(Path)
  for $all\ vul : Path$ **do**
    $Benefit = benefit\_function < vul >$
    $ROI = \frac{Benefit}{CC+CDC}$
    $\lambda = \max_{ROI}(vul)$
  end for
end function

IV. A TEST CASE

In this section we present the analysis of the intrusions events and the related information for each malicious activity. We also provide the attack graph and the extracted APT scenarios and the correlation of suspicious activities.

A. Intrusions Detection

In this part, we provide the analysis of events log, attack graph, and the correlation between their data.

1) **Event Log:** Table III contains some logged events and their base score.
2) **Evidence Correlation:** Since XSB [17] environment is used to support tabled execution of Prolog programs for MulVAL, we can benefit of this and query its database through ODBC connections to extract some facts that can be used as inputs for our evidence checking process. According to the work in [15], when the vulnerability scanner runs on each machine it outputs: *a*) the network configuration (represented as host access-control lists); *b*) the deduction rules; *c*) and the administrator-defined

security policy, which will be loaded into an XSB environment.

TABLE III: A sequence of stealthy actions (events) and their assigned BS.

ID	Events	Base Score(BS)
1	login from known host using public key authentication	5
2	download malware from remote host	7.9
3	local buffer overflow	6.3
4	remote user shell login	8.3
5	port scanning	5.3
6	ftp_connect	7.1
7	write to ftp home directory	7.9
8	remote user shell login	8.3
9	local buffer overflow	6.3
10	data exfiltration	10

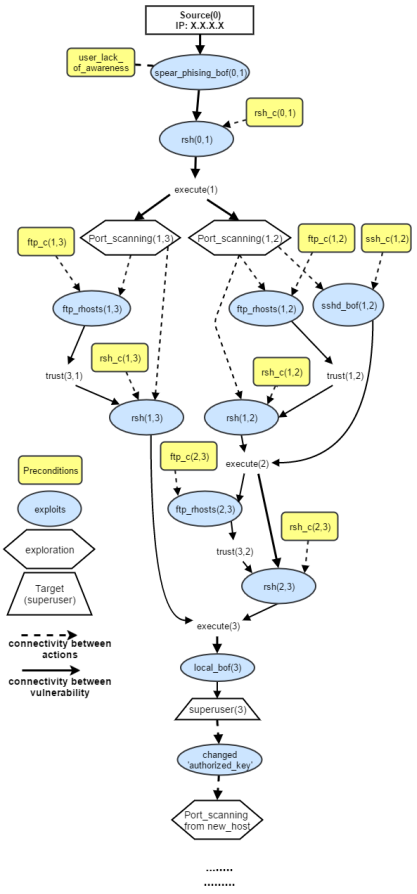


Fig. 2: Extended Attack Graph Simplified Example

Now we parse the received alerts of abnormal activities and related logged information to extract key features that can have matches in the XSB files. Abnormal alerts and related logged information can provide machine ID, timestamp, IP Source/Destination, port Source/Destination, and description of the malicious activity. Various alerts may share some common useful information which can be used as evidences too. Logged information can increase the confidence when we investigate the related logged information with the received alerts, for example, machine ID that is shown in the alert is an attribute can be used to check the related events in the log. Querying XSB, we can search for policy violation, and other related information of each vulnerability. For example, the vulnerability information, the *Host* contains the vulnerability, connectivity information, preconditions and postconditions of an attack, *Protocol* and *Port* number.

To illustrate this idea through an example, Figure 2 represents an attack graph for small network topology of only three machines (1,2,3) with several services and possible attacks, such as file transfer protocol FTP, secure shell SSH, and remote shell RSH. This system design is borrowed from [18] with some modifications. We assume that the attacker already tricked a user in organization and deployed a malware through spear phishing (social engineering) attack to use these exploits in a stealthy fashion to penetrate system to have root access to the target machine. Later the adversary can take actions to scan what services the targeted system is running. In this example, the adversary tries to have a root access (*superuser*) on the target machine(3). To gain the access he/she has, as shown in the attack graph, multiple APT scenarios with various preconditions, colored in yellow in Figure 2, holding on the VM from which the exploit is supposed to be initiated. These events may not trigger the common security systems to generate alerts. However, such events are logged and can be analyzed. Therefore, to identify the possible path the attacker can follow, correlating the vulnerabilities from the attack graph with their pre and post conditions, alerts, and the logged malicious events, in advance, is an important step and our major concern in this work. Due to page limitation, Table IV only shows few of the APT scenarios that cab be extracted from Figure 2.

TABLE IV: List of APT scenarios from the attack graph

No.	APT Scenario
1	rsh(0,1) → ftp_rhosts(1,2) → rsh(1,2) → ftp_rhosts(2,3) → sshd_bof(1,2) → rsh(2,3) → local_bof(3) → full_access(3)
2	rsh(0,1) → ftp_rhosts(1,2) → rsh(1,2) → rsh(2,3) → local_bof(3) → full_access(3)
3	rsh(0,1) → ftp_rhosts(1,2) → rsh(1,2) → ftp_rhosts(2,3) → rsh(2,3) → local_bof(3) → full_access(3)

V. Evaluation

A. Experimental Environment

We set up a centralized syslog server for 6 hosts (1 Windows 7 VM, 1 Fedora 23 VM, 1 CentOS 6.5 VM, 1 Kali Linux 2.0 VM, 1 Metasploitable 2.0 VM, 1 Ubuntu 16.04 VM) in an Openstack based cloud networking environment

The set up consists of following assumptions

- Linux tools used to identify system normal users, super users and network services on each individual VM.
- Auditing policy was defined by system admin e.g, "check normal escalation (uid=0), check rlogin attempts on open ports, check large files uploads and exe downloads by users".
- A team of 7 students was asked to perform normal and paranormal activity that would create logs for either the normal system usage (e.g- authenticated user logs into FTP, ssh access by user) or malicious activity distributed over a long period of time (e.g.- SYN probe at instant

t=0, priv escalation attempt at t=1, rlogin on open port at t=2, etc). The goal of attacker will be to perform the malicious activities in a way to evade normal signature based detection agents like Snort.

B. Advanced Persistent Threat Pattern Mining

We conducted analysis of changes to the attack pattern based on lateral movement of attacker in the network. The results below show (Static Exploit Pattern) where exploitation of system described above is purely based on system vulnerabilities. The system logs were analyzed over a 3 days period May,07, 2017 to May,09,2017 to identify activities of a malicious user who tries to perform stealthy activities to gain access to system and at the same time bypassing a intrusion detection system which we installed to identify signature based attacks.

1) ***Target: metasploitable VM, attacker: user****:* We found the attack pattern for goal node *metasploitable VM* to be multi-hop attack where attacker will need to exploit intermediate nodes to reach target node.

$$RULE5(multi - hopaccess) \rightarrow execCode('192.168.4.8', user) \quad (1)$$

In case of dynamic attack, where attacker is utilizing dynamic payload apart from system vulnerabilities

$$RULE3(remote\ exploit\ for\ a\ client\ program) \rightarrow execCode('192.168.4.8', user) \quad (2)$$

This shows attacker is trying to pass malicious payload to client program to gain privilege, which will be much easier when it is compared to a multihop attack.

2) ***Target: metasploitable VM, attacker: someUser****:* We analyzed the exploitation of system using a "someUser" account with minimal privileges. The user is not able to find any exploit path purely based on system vulnerabilities. In case attacker utilizes malicious input to remote server program, which was *rlogin* in case of VM 192.168.4.8(metasploitable), attacker is able to exploit system using attack pattern

$$RULE2(remote\ exploit\ of\ a\ server\ program) \rightarrow execCode('192.168.4.8', someUser) \quad (3)$$

This shows that the attacker is more successful using both static system information and dynamic stealthy activity over an extended period of time.

C. Dynamic Attack Graph

We analyzed the lateral movement of attacker from IDS and application logs. The logs collected at different time periods are combined with attack graph based on system vulnerability information. Based on information from attack graph analysis we were able to construct patterns ominous in attack graph. We have highlighted the events taking place across time using Table V. The attacker starts by probing and tries to target key network open services, e.g., ftp, telnet, etc. Once attacker gains access to vulnerable service e.g. vsftpd *row 4*, he targets system vulnerabilities and tries to mount multi-hop attack to other service which are more important *rows 5, 6*. The time series analysis of attack graph information, in Table V, shows that Ubuntu-Client downloaded some large file on Ubuntu-Client *row 1*, also as shown in *row 2*. The VM Ubuntu-Client was used to perform *rlogin* on Metasploitable VM which succeeded *row 5*, whereas file system mount on Metasploitable failed *row 6*. These sequence of activities, depicting lateral movement of attacker, are extremely hard to detect for a normal Intrusion Detection System (IDS), but they can be mined using Dynamic APT pattern mining leveraged by us using graph representation of time series data in this work.

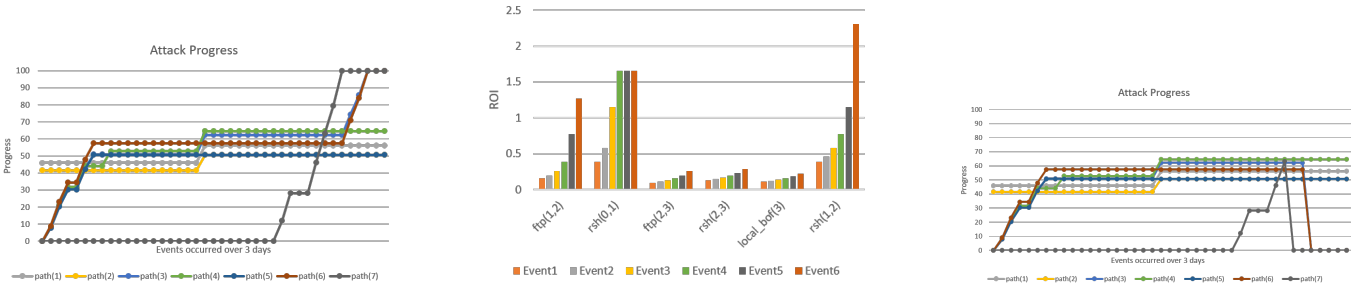
TABLE V: Log Attack Pattern

Time stamp	VM	Process	Log Data	Type
May, 7,7:35:02	Ubuntu-Client	cracklib:	Download File	Application
May, 7,16:56:15	Metasploitable-Client	in.telnetd [9215]:	warning:	Application
May, 7,21:27:16	Ubuntu-Client	groupadd [17670]:	group	System
May, 8,18:30:59	Ubuntu-Client	vsftpd [4943]:	anonymous	Application
May, 8,21:29:47	Metasploitable-Client	in.rlogind [13569]:	connect	Application
May, 8,21:37:57	Metasploitable-Client	mountd [4190]:	refused System	System

D. Real Time Identification of Active Attack Scenarios

To be able to track the attacker's position and identify the ongoing attacks, we use the abnormal events to update the criticality values related to each attack scenario in real-time. As consequence of this operation we will be able to detect, in real time, the paths which are currently under attack. Figure 3a illustrates the real-time evolution of the criticality of attack paths and, hence the progress made by the attacker based on the information collected from log files. The time window we presented in the previous figure is extracted from a real-time assessment of our experimental networked system, so some paths (e.g., path(1) and Path(2)) start already with a non-null progress value, which means that the attacker performed some progress before time zero of the graph. APT paths may have some shared vulnerabilities and events, for this reason the experiment showed that some events and vulnerability exploits are impacting the progress value for multiple paths. In Figure 3a, we didn't consider the countermeasure application when the attack progress exceeds the threshold, so some paths (e.g., path(7), path(3), and path(6)) get fully traversed by the attacker. Our framework is able to detect if an attacker was hidden in the system, for a period of time, and when he starts using previously gained information to continue his progress (e.g., path(6)), which confirms that the attack is persistent (APT). During a time window (period) the progress of attack on some paths can be negligible or null (e.g., path(1)), but its value will be persistent. The persistence of the attack progress value is the key feature that is leading to APT discovery.

Figure 3b depicts the ROI after applying a countermeasure (here, we choose *traffic redirection* as an example) on a specific vulnerability. The sequence of events raise the ROI of vulnerabilities for the simple reason that the attacker is getting



(a) Attack Progress. (b) Return Of Investment. (c) Attack Progress.

Fig. 3: Evaluation results.

closer to them. A path is selected for the countermeasure application when the attack progress exceeds the threshold. Figure 3c shows countermeasure application with 70 percent as maximum threshold. According to our algorithm and the threshold, the first path that has been selected for countermeasure is path(7) and its new value will become zero. The progress value of path(7) is showing a fast increase because of its short length. Consequently the attacker is making a significant advancement each time he performs a successful step towards the target. Some paths, which have common vulnerabilities will be mitigated in the same time (e.g., path(3) and path(6)). After selecting a path(3) for countermeasure, rsh shown high ROI and, hence a countermeasure has been selected to mitigate it. Consequently, the path(6) which contains the same vulnerability has been impacted.

VI. Conclusion

In this paper, we presented a new approach that can be used to identify relations between events and correlate them with system configuration vulnerabilities to detect stealthy type of attacks. The proposed approach utilizes the attack graph to draw an extended and informative graph that provides a comprehensive overview of all attack steps and allows an investigator to reconstruct the event relations to better understanding of the system current situation and ease the deployment of countermeasure selection. Our approach reduces the complexity of the attack graph, since it only considers the most critical paths in the extended attack graph. For the evaluation of our proposed approach, we simulated APT attack with real world methods and tools. The results of combining multiple sources of information and the utilization of AG show that our approach is able to identify the hidden and low frequency malicious activities of the simulated attacker. In the future work, we plan to collect more system data over long time and perform multiple scenarios of APTs.

References

[1] Colin Tankard. Advanced persistent threats and how to monitor and deter them. *Network security*, 2011(8):16–19, 2011.

[2] SJ Rockefeller. A kill chain analysis of the 2013 target data breach. Technical report, tech. rep., Committee on Commerce, Science and Transportation, 2014.

[3] Martin Ussath, Feng Cheng, and Christoph Meinel. Event attribute tainting: A new approach for attack tracing and event correlation. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 509–515. IEEE, 2016.

[4] Thomas M Chen and Saeed Abu-Nimeh. Lessons from stuxnet. *Computer*, 44(4):91–93, 2011.

[5] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*, volume 2012, 2012.

[6] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Mark Felegyhazi. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet*, 4(4):971–1003, 2012.

[7] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127–141, 2016.

[8] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, Daesung Moon, and Jong Hyuk Park. A comprehensive study on apt attacks and countermeasures for future networks and communications: challenges and solutions. *The Journal of Supercomputing*, pages 1–32, 2016.

[9] Peng Ning, Yun Cui, and Douglas S Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 245–254. ACM, 2002.

[10] Guillaume Brogi and Valérie Viet Triem Tong. Terminaptor: Highlighting advanced persistent threats through information flow tracking. In *New Technologies, Mobility and Security (NTMS), 2016 8th IFIP International Conference on*, pages 1–5. IEEE, 2016.

[11] Candace Suh-Lee and Juyeon Jo. Quantifying security risk by measuring network risk conditions. In *Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on*, pages 9–14. IEEE, 2015.

[12] Yu Liu, Nasato Goto, Akira Kanaoka, and Eiji Okamoto. Privacy preserved rule-based risk analysis through secure multi-party computation. In *Information Security (AsiaJCIS), 2015 10th Asia Joint Conference on*, pages 77–84. IEEE, 2015.

[13] Doudou Fall, Takeshi Okuda, Youki Kadobayashi, and Suguru Yamaguchi. Security risk quantification mechanism for infrastructure as a service cloud computing platforms. *Journal of Information Processing*, 23(4):465–475, 2015.

[14] Su Zhang, Xinwen Zhang, and Xinming Ou. After we knew it: empirical study and modeling of cost-effectiveness of exploiting prevalent known vulnerabilities across iaas cloud. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 317–328. ACM, 2014.

[15] Xinming Ou, Sudhakar Govindavajhala, and Andrew W Appel. Mulval: A logic-based network security analyzer. In *USENIX security*, 2005.

[16] Common Vulnerability Scoring System Version 3.0. https://www.first.org/cvss/user-guide/, 2017.

[17] Prasad Rao, Konstantinos Sagonas, Terrance Swift, David S Warren, and Juliana Freire. Xsb: A system for efficiently computing well-founded semantics. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 430–440. Springer, 1997.

[18] Anoop Singhal and Xinming Ou. *Security risk analysis of enterprise networks using probabilistic attack graphs*. Citeseer, 2011.