Discovering Attack Strategies Using Process Mining

Sean Carlisto de Alvarenga, Bruno Bogaz Zarpelão, Sylvio Barbon Junior
Computer Science Department
State University of Londrina (UEL)
Londrina, Paraná, Brazil
E-mail: {sean, brunozarpelao, barbon}@uel.br

Rodrigo Sanches Miani
School of Computer Science (FACOM)
Federal University of Uberlândia (UFU)
Uberlândia, Minas Gerais, Brazil
E-mail: miani@ufu.br

Michel Cukier
A. James Clark School of Engineering
University of Maryland
College Park, Maryland, USA
E-mail: mcukier@umd.edu

***Abstract*— Intrusion Detection Systems generate alerts which depend on manual analysis of a specialist to determine a response plan. However, these systems usually trigger thousands of alerts per day. Investigating unmanageable amounts of alerts manually becomes burdensome and error-prone. Besides, it complicates the analysis of critical alerts. In this paper, an approach is proposed to facilitate the investigation of huge amounts of intrusion detection alerts by a specialist. The proposed approach makes use of process mining techniques to discover attack strategies observed in intrusion alerts, which are presented to the network administrator in friendly visual models. Tests were performed using a real dataset from the University of Maryland. The results show that the proposed approach combines visual features along with quantitative measures that help the network administrator to analyze the alerts in an easy and intuitive manner.**

Keywords-intrusion detection; security visualization; alert mining; heuristic mining.

I. INTRODUCTION

In recent years, the increase of security vulnerabilities has concerned companies and organizations. In 2014 alone, almost 8000 new vulnerabilities were found in software applications and operating systems, as shown by the National Vulnerability Database (NVD) statistics [1]. The more increases the number of new vulnerabilities, the greater the likelihood of increase in the frequency of computer security violations. That is where the security measures come into play.

Intrusion Detection Systems (IDS) are devices that play an important role in the set of security policies in information systems. IDS monitor the network and system activities for any security violations. When it detects a security violation, it reports the event to a network administrator, who assesses the threat and initiates a response [2]. Unfortunately, IDS sensors generate huge amounts of alerts that makes it difficult to analyze them and identify relevant alerts [3]. To address this problem, alert correlation techniques [3][4][5] have been proposed to extract high-level descriptions of huge amounts of alerts.

The idea of using high level descriptions and graphical models in security assessment is not exclusive of alert correlation research, but it is also employed in the theory of attack trees and attack graphs. Attack trees and attack graphs have been extensively used to a variety of purposes such as attack and defense assessment, as well as for metrics quantification (e.g., cost, time, impact, probabilities, etc.). However, these representations usually require some expert knowledge of the network (e.g., topology, hosts) to generate the model.

In this paper, an approach is proposed to the IDS alert analysis problem from a process-oriented perspective. Alerts are considered as events of a process and they are analyzed with process mining techniques to generate a process model. The process model is a high-level visual representation of attack strategies observed in IDS alerts.

The proposed approach has the following benefits. At first, specific data acquisition is not necessary since companies and organizations usually employ IDS sensors to protect their networks. Secondly, process models provide an intelligible and intuitive way to interpret complex information such as IDS alerts. Thirdly, it is possible to model different perspectives from the alerts, e.g., the attackers' perspective, giving the network administrator a comprehensive view of the network. Moreover, it supports different levels of granularity in analysis as it is possible to filter the most frequent behavior observed in the alerts. Finally, the proposed approach shows the strategies that attackers are employing to compromise the network, helping network administrators to determine preventive measures.

The rest of the paper is organized as follows. Section II reviews related work. Section III defines the preliminary concepts used in this paper. Section IV shows the proposed approach and its operation. Section V presents the results obtained in the evaluation of the proposed approach. Finally, the Section VI contains concluding remarks and future work possibilities.

II. Related work

In this section, an overview of previous work on attack modeling and IDS alert analysis is presented. Previous work that used real IDS alerts data to discover attack strategies with process mining techniques was not found in literature. Therefore, approaches that use visual representation for attack modeling and data mining for IDS alerts analysis will be presented.

One of the great advantages of using higher level graphical models is that they are intuitive and facilitate threat assessment and attack scenario understanding. Attack trees and attack graphs are the most common methods used to modeling attack threats. As introduced by Schneier [6], attack trees are a visual representation that aims at modeling an attack in a tree structure. The attacker's goal is specified as the root of the tree. Branches in the tree represent attack subgoals, which can be represented as disjunctive or conjunctive nodes. Disjunctive nodes depict different alternative paths that an attacker can follow to achieve his goal. Conjunctive nodes represent different steps an attacker needs to take in order to achieve a goal [7].

Unlike the approach proposed in this work, attack trees are often modeled manually, a labor-intensive and error-prone process. In [8][9][10], this problem is addressed by methods to automate attack trees generation. Moore et al. [11] use attack trees to represent security attacks and document information, aiding security analysts to identify attack patterns. Tidwell et al. [12] enhance attack trees to represent multi-stage attacks behavior with an attack specification language.

Attack trees have some limitations regarding attacks modeling. This type of representation is static and can not take temporal aspects, such as dynamic time variations and order or priority of actions [7]. Therefore, this representation is not suitable for the proposed approach that takes these aspects into account.

Attack graphs are another way to represent and analyze security attacks. The term was first introduced by Phillips and Swiler [13]. In an attack graph, the nodes represent the network state and the edges represent an action of the attacker that changes the state. Weights can be assigned to the edges to enrich the model and algorithms can be applied to graph analysis, e.g., shortest path, to find which paths are more likely to succeed, time to success and other metrics [7]. Swiler et al. [14] developed a tool to generate attack graphs. Researches in [15][16] addressed the scalability problem of the graph size. Attack graphs are generated based on information about the attack, the system and the attacker profile [7]. This requires some background knowledge that is not always known. The approach proposed in this work generates the model based only on IDS alerts and hence does not require such knowledge.

Researchers have also studied how to extract attack information from huge volumes of IDS alerts. In [3], Ning and Xu published one of the first researches in this field. They proposed a model that builds graphs from IDS alerts to represent attack strategies. The authors also presented a method to measure the similarity between different attack strategy graphs. In more recent work, Lagzian et al. [4] and Xuewei et al. [5] used data mining techniques. Lagzian et al. presented a framework that, at first, aggregates the alerts in graphs. Then, it applies the Bit-AssocRule algorithm to mine the most frequent patterns in the graphs. Xuewei et al., on the other hand, proposed to identify causal relationships between the alerts with Markov models.

III. Background Information

A. *Intrusion Detection Systems*

An IDS is a software or a hardware device that monitors computers or network traffic for malicious activities or intrusive behavior. Once a malicious activity is detected, IDS can either raise an alert or log the event [17]. IDS can be classified into two categories, namely network-based and host-based. Moreover, it can use one of these three techniques: signature-based detection, anomaly-based detection or hybrid [18].

Signature-based detection is the process of comparing patterns or signatures that corresponds to a known threat against observed network events to identify malicious activity. This technique uses a database of already known attack signatures for detecting intrusions. Signature-based IDS is very effective to detect known attacks, already defined in its database. On the other hand, it can not detect attacks that do not have a previous signature, e.g., zero-day attacks or modified attacks. This limitation is circumvented by adding new signatures and keeping the database up to date.

An anomaly-based IDS works by distinguishing an abnormal behavior from what is considered to be normal. Therefore, this technique builds a model of normal traffic and raises an alert for any traffic that deviates from this model. A great advantage of this method is the detection of new attacks without any prior knowledge. The weakness of anomaly detection is the difficulty to define a model for what is normal, what is malicious and the boundaries between them.

A hybrid method combines the qualities of both signature-based and anomaly-based detection and integrates them in a single system.

B. *Process Mining*

Process mining depicts a set of methods and approaches that combine data mining techniques and business process modeling and analysis [19]. Process mining uses information recorded in a log to extract knowledge and represent it as process models. Therefore, it is important that logs have relevant and proper information as they are the starting point for process mining techniques.

For process mining, each record in the log is considered an event, the reason the logs are known as event logs. Furthermore, to extract information from the event logs, some characteristics must be considered [20]:

- Each event in the log corresponds to an *activity*, i.e., an action that was performed in the process [20]. As an example, suppose a user registration system that records all its actions in a log. Each recorded action, e.g., *Create User*, *Update User*, *Delete User*, etc., can represent an activity in the process.

- Each event in the process has to refer to a process instance or *case*. A *case* defines the process scope, i.e., where the process starts and where it ends. In the example of a user registration system, a set of events associated to the registration of a particular user can compose a *case*.
- Events can have attributes such as *activity*, *time* and *resource*. The attribute *activity* shows the event action, as mentioned before. The attribute *time* records the event timestamp. Finally, the attribute *resource* presents the responsible for performing the event.
- Events within a case are ordered as they occur, e.g., according to their timestamp. The occurring sequence of events is crucial because process mining algorithms determine causal dependencies between events to build the model.

There are three main areas in process mining, namely process discovery, conformance checking and model enhancement. Process discovery is related to how to transform an event log into a model. A process discovery technique receives as input an event log and returns as output a process model, so the model is representative for the behavior observed in the event log [19]. This is the main focus of the approach proposed in this work. Conformance checking uses metrics such as fitness and precision to evaluate the process model in the context of a log. Model enhancement uses new information to improve the process model.

In the next subsections, some process discovery techniques are briefly discussed. It is out of scope to discuss in details how these algorithms work, but benefits and drawbacks of each one will be pointed out. Further details can be found in [19]-[26].

C. *The α-Algorithm*

Proposed by van der Aalst et al. [23] in 2003, the α-Algorithm is one of the first algorithms designed for process mining and its ideas contributed to the development of more powerful discovery algorithms currently in use. The algorithm produces as output a WorkFlow net (WF-net), which is a subclass of Petri nets. In a WF-net, all nodes are on a path from the source place (unique place where the process starts) to the sink place (unique place where the process ends). The α-Algorithm examines the event log for four ordering relation between activities: directly follows relation, dependency relation, non-parallel relation and parallel relation. Refer to [22][23] for a complete description of the algorithm.

Under some specific conditions, the α-Algorithm works well. However, it has problems to deal with some situations (control-flow constructs) found in real life event logs. For instance, short loops, i.e., loops of length one or two, make the algorithm to derive an incorrect WF-net. Short loops occur when the same activity or two activities are executed multiple times in sequence. Considering IDS alerts, this may happen when the attacker attempts to perform the same violation several times until succeed.

The α-Algorithm has other limitations as it may not derive a correct WF-net when dealing with noise, i.e., event log with rare events that do not represent the process behavior, and incompleteness, i.e., the event log does not have enough events to discover a model. Therefore, this technique is not suitable in most real life processes.

D. *The α-Algorithm extensions*

To overcome the α-Algorithm limitations, many extensions have been proposed. Each of them extends the α-Algorithm to add support to some constraint. The α^{+}-Algorithm deals with the short loop problem. The Tsinghua-α-Algorithm focuses on event logs containing activities associated to transactional life-cycle. The α^{++}-Algorithm seeks to support non-free-choice control-flow construct. The $\alpha^{\#}$-Algorithm and the α^{*}-Algorithm concentrates on discovering some Petri nets that are not in the class of WF-nets and hence can not be discovered by the basic algorithm. Refer to a survey in [26] for more details.

E. *Heuristic Mining*

As mentioned in Section III-C, one of the limitations of the α-Algorithm is it can not deal with noise. However, noise is common in real life event logs due to information incorrectly logged and occurrence of exceptional events [23]. The Heuristic Mining algorithm handles this problem by taking the frequencies of events into account. Therefore, the algorithm can express the main behavior observed in the log without including the low frequency behavior from the noise into the model. Short loops are also overcome by the use of dependency/frequency table (D/F-table) and the dependency score [25]. The D/F-table contains metrics about the frequency of ordering relations occurrence, e.g., number of times one activity is directly followed by another activity. Based on these metrics, the dependency score, a numeric value between -1 and 1, is computed. The dependency score represents how strong the dependency relation between activities is. For instance, if the dependency score between activity *a* and itself is close to 1, then *a* is often the cause of *a*, suggesting a loop. These metrics along with dependency score and a threshold can be used to refine the output model.

IV. PROPOSED APPROACH

In this section, the proposed method to automate the discovery of attack strategies using a process mining discovery algorithm will be introduced. The proposed approach consists of four steps. In the first step, alerts with common features are aggregated. In the second step, the aggregated alerts are converted in a suitable format for process discovery algorithms. In the third step, the process discovery algorithm is executed to build the attack model. Finally, in the last step, the resulting attack model is analyzed. Figure 1 shows the four steps that compose the proposed approach.

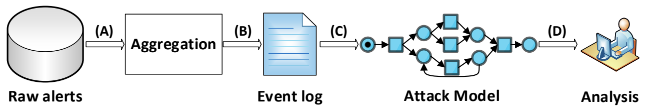

Figure 1. The four steps of the proposed approach.

In the following subsections, the details of each step are described. Then, in the next section, the method is evaluated using a real dataset of IDS alerts, discussing the results and some considerations.

A. *IDS alerts aggregation*

In attack strategy discovery, the goal is to discover how the attackers are attempting to compromise the network. After attack strategy discovery, network administrators can know each step attackers often take and the dependencies between these steps. Therefore, in the first stage of the proposed approach, alerts with common features are aggregated, aiming to group the alerts that compose each attack strategy. Then, in the next stage, the process discovery algorithm will investigate the relationships between these alerts. To aggregate the alerts, the perspective to be represented is taken into account. The perspective denotes how the alerts will be associated in the aggregation process. For instance, to represent the attackers' perspective, one can aggregate the alerts originating from the same source IP address. Similarly, to represent the targets' perspective, one can aggregate the alerts with the same destination IP address. The flexibility to represent different perspectives in this step can be explored to provide the network administrator a comprehensive view of the network.

B. *Conversion of aggregated alerts to an event log*

As aforementioned, for process mining, the input dataset should consist of events recorded in a log. Therefore, since the intention is to use process mining in IDS alerts analysis, the second step of the proposed approach is to convert the aggregated alerts into an event log. IDS collect information that may vary according to the type of device. This information may include source IP address, destination IP address, source port, destination port, Autonomous System Number (ASN) information, signature severity, attack type group for each signature, etc.

To analyze IDS alerts under a process mining perspective, each individual alert is considered an event. Each event attribute (i.e., the attributes of the alerts) will be analyzed to build the attack model. Because the objective is to discover attack strategies, the alert attributes that provide information about the attacks must be chosen. Then, this information will be used to build an event log with the characteristics required by process mining such as the concepts of *case*, *activity* and *time* (see Section III-B).

At first, event activity (i.e., the action performed in the process) has to be defined. The event activity is an important information as it will be denoted by the nodes in the attack model. The nodes in the model represent the steps performed in the attack-flow and help the identification and visualization of sequences and dependencies of attacks in the model. Usually, IDS record information about what triggered the alert, e.g., some signature identification or description of the violation. In the context of IDS alerts, the signature can be considered the action of the attacker as it depicts his intentions to compromise the network. Therefore, the signature is defined as the event activity.

Moreover, in an event log, events should be grouped in a case (i.e., each event in the process belongs to a case). The case defines the scope of the process. During the process discovery, several cases are compared among each other to determine the causal dependencies between activities. In the proposed approach, a case is defined as a group of alerts that were aggregated in the first step (see Section IV-A) and occurred within a time span *t*. As an example, suppose that the alerts are aggregated according to the source IP address and the time span *t* is set as 1 day. Then, all alerts with source IP address *x.x.x.x* triggered in day *m* will belong to case *i*. All alerts with source IP address *x.x.x.x* triggered in day *n* will belong to case *j*. Finally, all alerts with source IP address *y.y.y.y* triggered in day *m* will belong to case *k*. In this manner, each attacker composes a case and his attack steps (i.e., its alerts occurred within *t*) are the events of the case. Finally, in an event log, events in a case must be ordered as they occur. In the IDS alerts context, the timestamp information is used to order the alerts.

The event log format adopted in the proposed approach is the eXtensible Event Stream (XES). XES is an eXtensible Markup Language (XML)-based standard used to store event logs supported by most process mining tools including the ProM Framework [27] used in this research.

C. *Attack model discovery*

To build the model, the process discovery algorithm that will take the event log as input and generate the attack model as output must be defined. As mentioned before, process discovery algorithms have limitations regarding the control-flow constructs they can discover. Different algorithms may generate different attack models. Furthermore, some algorithms may generate attack models that are not able to represent the behavior observed in the event log and consequently may lead to wrong conclusions about the attacks.

In IDS alerts, loops may take place in the model, since events that compose a case may have repeated activities in sequence (e.g., situations in which the attacker executed the same violation until succeed or attempted to compromise multiples hosts such as in a *botnet*). The discovery algorithm should be able to detect these repeated activities and represent them not as individual activities in sequence but as a loop in the model. On the other hand, duplicate tasks (e.g., situations in which two different violations have the same signature) will unlikely be a problem because IDS alerts are atomic entities (e.g., a *buffer overflow* exploit will not have the same signature of a *nimda attack* in the log). Therefore, the proposed approach uses the Heuristic Mining algorithm as it can deal with these characteristics.

D. *Model evaluation*

After the model has been generated, an expert analysis is required. Through the model, different aspects can be observed. In the next section, a case study and some analysis will be presented and discussed.

V. RESULTS

To evaluate the proposed approach, raw IDS alerts generated by a signature-based device deployed at the University of Maryland, whose network has about 40000 computers, were used. These alerts were triggered between April and December 2012 for inbound and outbound network traffic of the University. The alerts raised in October were chosen to evaluate the method.

To perform the first step of the proposed approach, alerts with the same source IP address were aggregated. Then, the cases were defined, setting the time span *t* to 1 day. Consequently, alerts with the same source IP address that were triggered in the same day were associated to the same case. Moreover, only inbound alerts, i.e., alerts originating from traffic addressed to the University were considered. To represent the frequent behavior of the attackers, exceptional situations were filtered (similar to the filtering performed in [3]). Therefore, cases containing a single event (isolated alert) or cases containing multiple events associated to the same violation (i.e., same signature) were not included in the model. These cases do not depict attack strategies used by attackers, as they show an attack-flow with a single step and do not provide enough information on the behavior of the attacker, as illustrated by Figure 2.

Similarly, cases containing more than 50 events in which almost all the events have the same signature were not included in the model.

The ProM Framework [27] was used to generate the process model with the Heuristic Mining algorithm. Figure 3 shows the results of tests performed on October 7th. In this day, there were 97 events (i.e., triggered alerts) with 8 different activities (i.e., distinct signatures) organized in 9 cases.

Analysis of Figure 3 indicates that:

- Within that day, the attacks started in one of four violations: (i) *Malicious PHP Program Access*, (ii) *Malicious SMB Probe/Attack*, (iii) *Possible nmap Scan (XMAS (FIN PSH URG))* and (iv) *Impossible Flags (SFRPAU)*. Each of them lead to a different attack-flow.
- Among the 9 cases, there is one case that starts with *Malicious PHP Program Access*, two cases that start with *Malicious SMB Probe/Attack*, three cases that start with *Possible nmap Scan (XMAS (FIN PSH URG))* and three cases that start with *Impossible Flags (SFRPAU)*. Similarly, one case ends with *PHP Code Injection*, one case ends with *Windows PlugnPlay Request Anomaly*, four cases end with *Possible nmap Scan (XMAS (FIN PSH URG))* and three cases end with *Impossible Flags (SFRPAU)*.
- In (i), an attack can be clearly observed. First, the attacker executes a *Malicious PHP Program Access*. Afterwards, the attacker executes a *PHP Code Injection* and then the two activities come into loop. This shows that some attacker is injecting code (e.g., eval injection) into a PHP server located at the University network and then some user/visitor is accessing the server and executing the code. This attack-flow shows a possible unknown vulnerability that the network administrator has to fix.
- In (ii), in one of the attack-flows, the attacker performs a *Malicious SMB Probe/Attack* followed by *Windows PlugnPlay Request Anomaly*. Although not directly related, both attacks have something in common: they are associated to Microsoft Operating System (OS) and exploit vulnerabilities that allow remote code injection and elevation of privileges. These vulnerabilities, if successfully exploited, can allow the attacker to take control of the compromised system as reported by Microsoft Security Bulletin [28][29].
- In (iii), a possible attack attempting is presented. The attacker performs a port scan (*TCP Xmas scan*), probing the server or host for open ports. Port scan is a well known technique used in pre-attack phases to gather information about the target and be able to exploit them. After the port scan, the attack-flow splits into three paths. One path leads to (ii). The other path leads to (iv). In the third path, the attacker performs *NULL OS Fingerprinting Probe*, an attempt to collect information about the target OS and thus know what vulnerabilities he can/can not exploit (e.g., if the vulnerability was already patched in this OS version). After that, the path leads to (iv). This attack-flow indicates that the attacker is conducting a reconnaissance of the target before executing the attack.
- In (iv), the attack-flow is similar to (iii). The *Impossible Flags (SFRPAU)* are TCP packets with all flags (*SYN*, *FIN*, *RST*, *Push*, *ACK*, *UrgPtr*) set. These packets might be unintentional produced by poorly implemented applications but are more likely (considering the attacks in the paths it splits) from a *Full Xmas scan*.

It is possible to obtain other information by analyzing the model. For example, the *Impossible Flags (SFRPAU)* signature was the most executed attack (30 times). Next, there is the *Possible nmap Scan (XMAS (FIN PSH URG))* attack (21 times). The reason for this behavior is the loop between the attacks, showing that many port scans were executed in this day. In addition, the model provides an intuitive and easy way to investigate the alerts, showing the attack strategies that would hardly be discovered investigating almost 100 alerts manually.

As mentioned before, the attack model presented in Figure 3 represents the attackers' perspective, i.e., how multiple source IP addresses (i.e., the attackers) are attempting to compromise several targets in the University network. However, this representation may not be the ideal for all situations and other perspectives can be explored for a deep

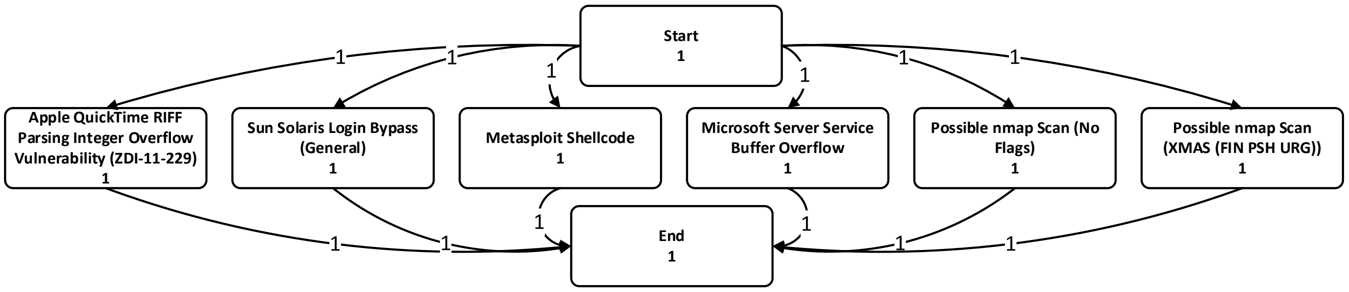
Start
1
Apple QuickTime RIFF Parsing Integer Overflow Vulnerability (ZDI-11-229)
1
Sun Solaris Login Bypass (General)
1
Metasploit Shellcode
1
Microsoft Server Service Buffer Overflow
1
Possible nmap Scan (No Flags)
1
Possible nmap Scan (XMAS (FIN PSH URG))
1
End
1

Figure 2. Example of the behavior of isolated alerts on October 7th.

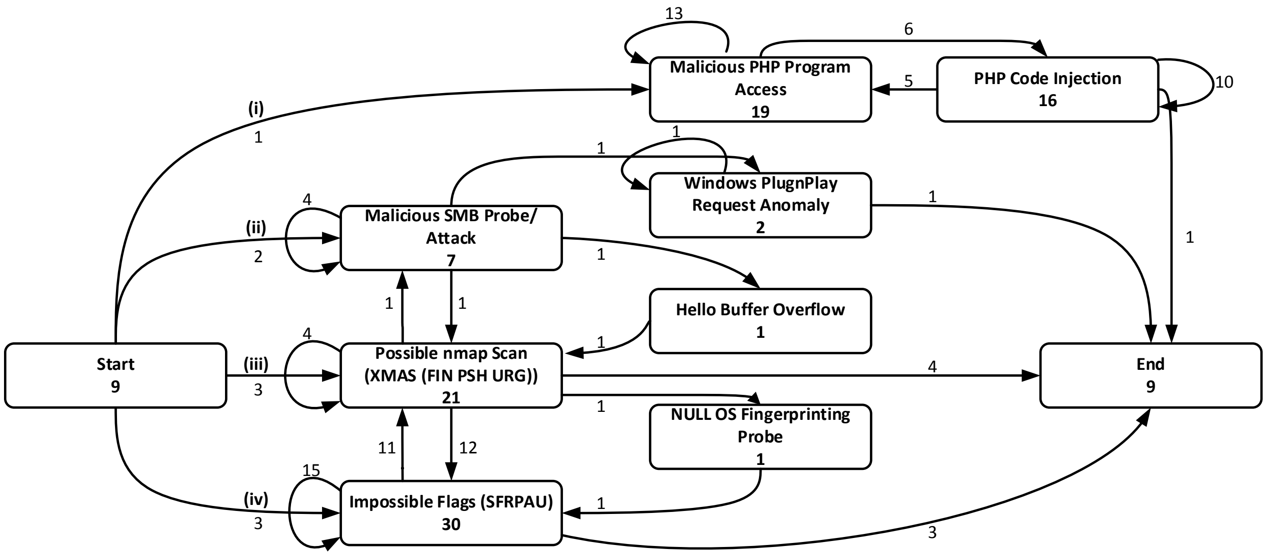
Start
9
Malicious PHP Program Access
19
PHP Code Injection
16
Windows PlugnPlay Request Anomaly
2
Malicious SMB Probe/ Attack
7
Hello Buffer Overflow
1
Possible nmap Scan (XMAS (FIN PSH URG))
21
NULL OS Fingerprinting Probe
1
Impossible Flags (SFRPAU)
30
End
9
(i)
(ii)
(iii)
(iv)

Figure 3. Attack model that represents the behavior of the attackers on October 7th.

investigation into the attacks. For instance, to represent the behavior of distributed attacks (many-to-one attacks), the targets' perspective can be explored (i.e., cases with events associated to the same destination IP address).

VI. CONCLUSION AND FUTURE WORK

This paper has addressed the problem of analyzing huge amounts of IDS alerts. A four step method that uses process mining techniques to mine the alerts and generate a process model, a high-level graphical representation of the attackers' behavior observed in the alerts, was proposed. The method was evaluated on a real IDS alert dataset from University Maryland. The results showed that the resulting model has an intuitive and user-friendly representation that can be used by network administrators as an alternative to the manual investigation of alerts.

As future work, the objective is to extend the attack perspectives and analyze the alerts from another viewpoint (e.g., the target perspective). Besides, it was observed that some models become complex as the number of distinct signatures increases. Therefore, clustering techniques may be employed to reduce the complexity of those models and conformance checking metrics, such as simplicity, may be employed to evaluate the model.

ACKNOWLEDGMENT

The authors would like to thank Gerry Sneeringer and the Division of Information Technology at the University of Maryland for allowing and supporting the described research.

REFERENCES

[1] "National vulnerability database," [Online]. Available: https://web.nvd.nist.gov/view/vuln/statistics. [Retrieved: April, 2015].
[2] S. O. Al-Mamory and H. Zhang, "Intrusion detection alarms reduction using root cause analysis and clustering," Computer Communications, 2009, vol. 32, no. 2, pp. 419-430.
[3] P. Ning and D. Xu, "Learning attack strategies from intrusion alerts," in Proceedings of the 10th ACM Conference on Computer and Communications Security. ACM, 2003, pp. 200-209.
[4] S. Lagzian, F. Amiri, A. Enayati and H. Gharaee, "Frequent item set mining-based alert correlation for extracting multi-stage attack scenarios," in Telecommunications (IST), 2012 Sixth International Symposium on. IEEE, 2012, pp. 1010-1014.
[5] F. Xuewei, W. Dongxia, H. Minhuan and S. Xiaoxia, "An approach of discovering causal knowledge for alert correlating based on data mining," in Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on. IEEE, 2014, pp. 57-62.
[6] B. Schneier, "Attack trees: Modeling security threats," Dr. Dobb's Journal, December 1999. [Online]. Available:

https://www.schneier.com/paper-attacktrees-ddj-ft.html. [Retrieved: April, 2015].

[7] B. Kordy, L. Piètre-Cambacédès and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," Computer Science Review, 2014, vol. 13–14, pp. 1-38.

[8] R. Vigo, F. Nielson and H. R. Nielson, "Automated generation of attack trees," in Computer Security Foundations Symposium (CSF), 2014 IEEE 27th. IEEE, 2014, pp. 337-350.

[9] S. Paul, "Towards automating the construction & maintenance of attack trees: a feasibility study," in Proceedings First International Workshop on Graphical Models for Security, GraMSec 2014, 2014, pp. 31-46.

[10] H. Birkholz, S. Edelkamp, F. Junge and K. Sohr, "Efficient automated generation of attack trees from vulnerability databases," in Working Notes for the 2010 AAAI Workshop on Intelligent Security (SecArt), 2010, pp. 47-55.

[11] A. P. Moore, R. J. Ellison and R. C. Linger, "Attack modeling for information security and survivability," Technical Note CMU/SEI-2001-TN-001, Carnegie Mellon University, 2001.

[12] T. Tidwell, R. Larson, K. Fitch and J. Hale, "Modeling internet attacks," in Proceedings of the 2001 IEEE Workshop on Information Assurance and security, 2001, pp. 54-59.

[13] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in Proceedings of the 1998 Workshop on New Security Paradigms, 1998, pp. 71-79.

[14] L. P. Swiler, C. Phillips, D. Ellis and S. Chakerian, "Computer-attack graph generation tool," in DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01. Proceedings. IEEE, 2001, pp. 307-321.

[15] S. Jajodia, S. Noel and B. O'Berry, "Topological analysis of network attack vulnerability," in Managing Cyber Threats, 2005, vol. 5, pp. 247-266.

[16] X. Ou, W. F. Boyer and M. A. McQueen, "A scalable approach to attack graph generation," in Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, 2006, pp. 336-345.

[17] A. Patel, Q. Qassim and C. Wills, "A survey of intrusion detection and prevention systems," Information Management & Computer Security, 2010, vol. 18, pp. 277-290.

[18] J. Vacca, Computer and Information Security Handbook, Second Edition, Morgan Kaufmann, 2013.

[19] W. M. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer Science & Business Media, 2011.

[20] W. van der Aalst and C. Giinther, "Finding structure in unstructured processes: The case for process mining," in Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on, 2007, pp. 3-12.

[21] R. P. J. C. Bose and W. M. van der Aalst, "Context Aware Trace Clustering: Towards Improving Process Mining Results," SDM, 2009, pp. 401-412.

[22] A. de Medeiros, W. van der Aalst and A. Weijters, "Workflow mining: Current status and future directions," in On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, 2003, pp. 389-406.

[23] W. van der Aalst, T. Weijters and L. Maruster, "Workflow mining: discovering process models from event logs," Knowledge and Data Engineering, IEEE Transactions on, 2004, vol. 16, no. 9, pp. 1128-1142.

[24] A. Weijters and J. Ribeiro, "Flexible heuristics miner (FHM)," in Computational Intelligence and Data Mining (CIDM), 2011 IEEE Symposium on, 2011, pp. 310-317.

[25] A. Weijters and W. van der Aalst, "Rediscovering workflow models from event-based data using little thumb," Integrated Computer-Aided Engineering, 2003, pp. 151-162.

[26] B. van Dongen, A. Alves de Medeiros and L. Wen, "Process mining: Overview and outlook of petri net discovery algorithms," in Transactions on Petri Nets and Other Models of Concurrency II, 2009, pp. 225-242.

[27] B. van Dongen, A. de Medeiros, H. Verbeek and A. Weijters, "The prom framework: A new era in process mining tool support," in Applications and Theory of Petri Nets 2005, 2005, pp. 444-454.

[28] Microsoft, "Microsoft Security Bulletin MS05-039: Vulnerability in plug and play could allow remote code execution and elevation of privilege (899588)," August 2005, [Online]. Available: https://technet.microsoft.com/library/security/ms05-039. [Retrieved: April, 2015].

[29] Microsoft, "Microsoft Security Bulletin MS11-019: Vulnerabilities in SMB client could allow remote code execution (2511455)," April 2011, [Online]. Available: https://technet.microsoft.com/library/security/ms11-019. [Retrieved: April, 2015].