Host-based Cyber Attack Pattern Identification on Honeypot Logs Using Association Rule Learning

Angelos Papoutsis, Christos Iliou, Dimitris Kavallieros, Theodora Tsikrika,
Stefanos Vrochidis, and Ioannis Kompatsiaris
Information Technologies Institute, *CERTH*, Thessaloniki, Greece
{apapoutsis,iliouchristos,dim.kavallieros,theodora.tsikrika,stefanos,ikom}@iti.gr

***Abstract*—Attack pattern identification is a significant step for protecting organisations from cyber-threats, as it can be used to reveal valuable patterns, enabling the better detection and analysis of the respective attacks that can be leveraged for the development of effective and efficient Intrusion Detection Systems. In this work, Association Rule Learning (ARL), a data mining technique, is used for the identification of attack patterns from data collected from a public honeypot. Using the FP-Growth ARL algorithm, we identified different patterns of attacks and correlated the respective commands executed by various attackers. To our knowledge, this is the first time ARL has been used to extract attack patterns from commands run by the attackers using real-world log data collected at the host level.**

***Index Terms*—cyber-attack patterns, machine learning, rule-based learning, pattern identification, attack command correlation, honeypots, Dionaea, host-based log data**

I. Introduction

Nowadays, cyber-attacks have evolved in both complexity and diversity, with attackers using sophisticated and computationally powerful techniques to cause operational damage to organisations and individuals. To protect their assets, the affected parties leverage in their daily operations defence systems such as Intrusion Detection Systems (IDS), i.e., programs or devices that scan a system or network and detect potentially malicious actions [1]. However, due to the considerably high and continuously increasing number of cyber-attacks, cyber-security is not a trivial task, as it requires the allocation of a significant amount of human and monetary resources [2].

Towards cyber-security against such attacks, attack pattern identification can play an important role as it can reveal patterns followed by attackers, enabling the better detection and analysis of the respective attacks. To this end, data mining techniques can be used to analyse large amounts of data and reveal the underlying patterns of attacks, thus providing valuable insights that can be leveraged for the development of effective and efficient IDS [3]. One such data mining method that can be employed for attack pattern identification is Association Rule Learning (ARL) [4]–[6], a method that is also considered as unsupervised machine learning [7].

ARL is part of a more extensive family of *rule learning-techniques* used in the cyber-security domain for *rule-learning based IDS* [6], either for *descriptive rule learning* focusing mainly on pattern identification on specific datasets without any assumption about new data, or for *predictive rule learning* capable of predicting rules that generalise to new data [8], [9].

Different research efforts have used ARL methods for rule-learning based IDS operating either on the network or host environment of an infrastructure. Such efforts use publicly available datasets of raw network binary TCP data[1] [3], or combine data from various sources collected at host level, such as IDS, firewall, and system logs [10]. Data collected from hosts can be used to obtain richer information about the behaviour of attackers (such as specific commands that they run) which can then be used to extract more advanced intelligence about their actions [10].

Thus, in this work, we take advantage of the effectiveness of ARL in pattern identification and focus on host-based logs, since more advanced attack patterns can be extracted from them, when compared to network-based data. Different from current approaches that extract generic attack patterns from host-based logs (such as DDoS attacks), this work focuses on extracting patterns from the specific commands of the attackers. Specifically, we collected data using a Dionaea honeypot[2] deployed publicly on a cloud Virtual Machine (VM) on Amazon Web Services (AWS)[3]. These data include logs from several services provided by Dionanea, with only the MSSQL service though being supported in a way that allows high level interactions from attackers (i.e., supporting different commands run by the attackers). Using the FP-Growth ARL algorithm [11], we recognised several attack patterns given that a large set of real-world data (many attack events for nearly a month) was used. To our knowledge, this is the first time ARL is used to extract patterns from commands run by attackers on a honeypot and build correlations of these attack commands using real-world host-based log data.

The rest of this paper is structured as follows. Section II presents related work on attack pattern identification. Section III presents the methodology followed regarding data collection, command extraction, and the ARL algorithm we used. Section IV presents the evaluation procedures followed in the conducted experiments, the metrics used, and the constructed dataset. Section V presents our experimental results. Last, Section VI discusses our main findings and conclusions.

[1]http://kdd.ics.uci.edu//databases/kddcup99/kddcup99.html
[2]https://github.com/DinoTools/dionaea
[3]https://aws.amazon.com/

II. Related Work

Various research efforts have used ARL-based methods to address the problem of attack pattern identification. For example, a framework to detect network intrusion using association rules was proposed [3] where the rules were generated with the use of a modified version of the Apriori algorithm [12] and audit data (TCP data) derived from the KDD intrusion detection contest dataset. The constructed rules were then employed by an anomaly detection system to recognise intrusions.

The Apriori algorithm was also used for extracting attack patterns by analysing traffic destined to unused Internet addresses [13]. More specifically, different characteristics of the data were analysed, e.g., the packets distribution or its used transport, network, and application layer protocols. Threats and their correlations were also analysed, with denial of service attempts, buffer overflow exploits, and unsolicited VPN access being the most severe threats found.

In [14], an automatic way was proposed to reduce the false alarms of a Network IDS, with the use of the FP-Growth algorithm [11] and honeypot log similarity analysis on three months of data collected from a deployed honeypot. With these techniques only suspicious traffic would be present in the honeypots, resulting in the false alarms being reduced. In particular, mainly malware behaviour and DDoS attacks were identified, but not specific patterns regarding the commands.

In [10] association rules were extracted from multi-source security logs (e.g., snort, firewall, and system logs) captured in a cloud environment to identify attackers' intrusion behaviour. Data were gathered through simulation of intrusion attacks. The Adaptive-Miner algorithm [15] based on Apache Spark was used to built a security log rule base for intrusion detection and vulnerability assessment and lower the complexity generated by the different logs being analysed. On this base, different security rules were identified and also a distributed association rule mining-based MapReduce calculation model was developed for efficient implementations. The system was evaluated for real-time network intrusion detection and, despite some delay in the recognition, the approach was better than other methods in terms of precision, recall, and f-measure.

Contrary to aforementioned research, we focus our analysis on the specific commands that the attackers run, collected from host systems where richer information about attacks can be found (such as SQL commands), as opposed to network data. Also, in our work, we use a recently collected dataset, an important aspect in this line of research, as the cyber-attack landscape evolves continuously and new attacks appear.

III. Methodology

This section describes the methodology of the approach proposed in this work for the identification of attack patterns using an ARL algorithm on commands executed at the host level. Initially a honeypot implementation is used for attracting attacks by malicious actors, while an agent that operates on the honeypot monitors continuously the various log paths and identifies any new log entry in real-time. Then, these log data are stored and analysed for the detection of important features of an attack and some preprocessing steps are also applied. Last, these logs are analysed by an ARL algorithm.

A. Data Collection

To gather data, we utilise a honeypot implementation. Honeypots can mimic the behaviour (e.g., the services) of a typical system and therefore, in the view of attackers, honeypots are just another endpoint that can be attacked; however, honeypots are not actual systems and can lure attackers into performing malicious actions [16]. After extensive research, we decided to use the Dionaea honeypot deployed on a cloud VM located on the AWS platform. Dionaea has many benefits, including support for configurations that store only high level information about the attacks excluding specific details to reduce noise, while it can also store information about the attacks in JSON format, facilitating any further preprocessing steps.

To store the data from Dionaea and extract the relevant attack information, the Wazuh[4] security platform was used. Wazuh is a free, open-source security monitoring solution for threat detection, incident response, and compliance, which consists of (i) the Wazuh manager responsible for storing and analysing the data, and (ii) the Wazuh agent responsible for collecting the data from the target host; more than one agents can be registered (i.e., connected) to one manager. To allow for the collection of the data from Dionaea, a Wazuh agent is installed on the same VM where the Dionaea honeypot is installed. The agent periodically monitors Dionaea's logs and returns any new log entries to the Wazuh manager. The latter analyses the gathered logs to extract attack information.

Specifically, the Wazuh manager analyses the data collected in Dionaea using rule-based techniques (e.g. regular expressions) to extract information that provides insights concerning the attackers (e.g., IP address, source port, browser agent version) and the cyber-attack itself on the honeypot (e.g., destination IP and port). After identifying incidents, the relevant information is stored in JSON format. Then, periodically, all data collected up to a certain point are used as input to the ARL module that uses the FP-Growth algorithm for identifying patterns and correlations of attacks from these data.

B. Command Extraction

Different attributes can provide information about a cyber-attack, such as the commands executed by attackers or the systems ports targeted during the attack. By analysing the gathered data, we identified that not all attributes entailed information that can be used to extract useful patterns from attacks. Several of the services provided by the Dionaea honeypot (e.g., FTP, SMB, MongoDB, HTTP, etc.) were not fully supported, thus the collected commands were limited. Therefore, in our work, we used the following attributes: (i) the *data.cmd*, which describes the SQL commands used by attackers, (ii) the *data.connection.protocol* which defines the protocol used by attackers, and (iii) the *data.connection.local.port* which describes the system connection port that attackers targeted.

[4]https://wazuh.com/

The *data.cmd* entails long sequences of SQL commands.To construct shorter and eventually clearer rules, every sequence of SQL commands is separated into multiple commands based on different keywords that depict common SQL commands executed by different attackers, such as the "EXEC" keyword which is used to "execute" a selected procedure. Examples of such keywords include "EXEC", "exec" "DECLARE", "SELECT", "Drop", and "Create". For example, the following sequence of SQL Commands captured by our honeypot:

{*exec sp_server_info 1 exec sp_server_info 2*
exec sp_server_info 500 select 501,NULL,1
where 'a'='A' select 504,c.name,c.description,c.definition
from master.dbo.syscharsets c,master.dbo.syscharsets c1,
master.dbo.sysconfigures f where f.config=123 and
f.value=c1.id and c1.csid=c.id set textsize 2147483647
set arithabort on}

is split into the following five different SQL commands:

(i) *exec sp_server_info 1*
(ii) *exec sp_server_info 2*
(iii) *exec sp_server_info 500*
(iv) *select 501,NULL,1 where 'a'='A'*
(v) *select 504,c.name,c.description, c.definition from master.dbo.syscharsets c,master.dbo.syscharsets c1,master.dbo.sysconfigures f where f.config=123 and f.value=c1.id and c1.csid=c.id set textsize 2147483647 set arithabort on*}

After these steps, all attributes are fitted to our ARL method.

C. Pattern Identification using ARL

ARL has been applied to different domains with the goal to derive interesting patterns and correlations between variables in a database. Consequently, ARL can also be used to reveal interesting patterns in attackers' actions [17], where ARL along with sequential rules are the main techniques that have been applied for intrusion detection [18].

ARL was first proposed in commercial environments to model customers' purchasing behaviour [4]. In such settings, the products a customer buys are stored as objects that can be considered as distinct items and the main goal is to identify sets of items that are usually purchased together; these are expressed as rules. The same logical procedure can be transferred to the cyber-security domain where a command that an attacker executes can be viewed as an item and the final goal is the identification of items (e.g., commands) that an attacker executes together. As a result, attack patterns followed by attackers can be identified by correlating the executed commands. These attack patterns can be used either as part of an IDS to facilitate the detection of the attacks [10] or to further analyse the behaviour of attackers and extract insights.

In ARL, an itemset is a group of two or more objects (i.e., items) that appear together. If an itemset appears frequently over a specified threshold known as *support*, it is considered to be a frequent itemset (also known as candidate itemset) or frequent pattern [13]. A rule that depicts the correlation between different items can be expressed in the form of an If-Then structure. The *if* and *then* parts are generally known as *antecedent* and *consequent*, respectively. For example, let's consider three different commands (i.e., items): A, B, and C. A rule can be expressed in the form: If A and B, then C. This rule states that if commands A and B are executed together, command C is also executed with some probability [3].

In this work, we used the FP-Growth algorithm [11], which has many advances over the popular Apriori algorithm [12]. Apriori starts by identifying all unique items (length of one) in a database and then follows an iterative process to find all itemsets that have support value equal or higher to a predefined threshold, so that frequent patterns of length two or more are used for generating the association rules; a rule can then be identified between the items of a frequent pattern or between different frequent patterns. The length of the rule depicts the number of commands (i.e, items) that a rule entails from the left side (antecedent part) to the right side (consequent part). In our datasets, we identify length of commands from 2 to 10, as a unique command cannot be considered as rule by itself.

However, Apriori suffers from two main drawbacks: (i) the number of candidate itemsets at each iteration can become very large, thus adding complexity as more rules are generated and need to be eventually analysed by security experts, and (ii) the whole database must be scanned at each iteration, adding computational complexity that depends on the length of the longest mined frequent itemset. Thus, Apriori presents high memory consumption and execution time, especially for large datasets. On the other hand, FP-Growth does not generate candidate itemsets. Instead, it uses a tree-based representation of the original dataset, known as FP-tree, from which the frequent itemsets are mined. The fact that no candidate generation is needed and that the database is scanned twice in total addresses the two main drawbacks of Apriori.

Overall, ARL is a good candidate for our problem, as it can identify attack patterns without time-consuming and complex prepossessing steps, which is particularly important in the modern, complex cyber-security landscape, while FP-Growth can assist in finding many attack patterns on large datasets without the drawbacks of the Apriori algorithm.

IV. Experimental Setup

A. Dataset

The data were captured continually for 25 days from 2022-02-04 to 2022-02-28. Every 24 hours we built a different dataset, thus ending up with 25 datasets. The length of the datasets varies depending on the traffic that the honeypots attracted, with each dataset being composed of different samples and attributes. Each sample in our dataset represents a security incident in the deployed honeypot. Table I depicts the characteristics of the captured datasets regarding the SQL commands executed by attackers.

B. Evaluation Methodology

As this research aims to extract attack patterns from the commands executed at the Dionaea honeypot, we initially

TABLE I
DATASETS COLLECTED IN THE DIONAEA HONEYPOT FOR 25 CONTINUOUS DAYS

	Dataset	1	2	3	4	5	12	13	14	15	16	17	18
SQL Commands	**# of samples**	1496	2033	1767	1290	462	882	1494	1631	1768	1141	1427	883
	# of unique	130	154	142	121	85	102	130	136	160	115	127	103

extract the commands of each attack from each generated dataset and then this information, along with the respective protocol and port are used, as input to the ARL module for the extraction of the respective attack patterns.

In our experiments, we used the FP-Growth implementation in the mlxtend library[5]. The representation of the raw data is performed using the pandas library[6], where each collected dataset is considered as a dataframe, each sample as a dataframe's row, and each attribute as a dataframe's feature; in the remaining paper, these terms will be used interchangeably.

However, not all datasets, each collected during a 24-hour period, were useful for our experiments. For example, for specific days, some datasets had little attack information as our honeypot did not attract enough traffic, while others had enough information, but not of interest to us, such as SQL commands. Overall, among the 25 datasets, only 12 (depicted in Table I) were useful to us. For our experiments, we first concatenated these datasets into one with a length of 16472 entries (i.e, SQL commands). From this dataset, we extacted 259 frequent itemsets using a support of 0.15%, meaning that 259 sets of items occur together in at least 15% of all items.

C. Evaluation Metrics

In ARL, different rules can be generated depending on the dataset size and the complexity of commands executed by attackers. To select specific rules, different metrics can be used [19], [20]. In this work, we consider the following metrics: (i) *Support* $\in \{0, 1\}$ of an itemset X defined as the proportion of transactions in the dataset that contain the itemset, showing how many times a generated rule appears in a dataset [12]; (ii) *Confidence* $\in \{0, 1\}$ showing the percentage of cases in which a consequent Y appears, given that the antecedent X has occurred, thus measuring the reliability of the rule; it is calculated as the number of transactions containing X and Y, divided by the number of transactions containing X [4]; and (iii) *Lift* $\in \{0, \infty\}$ showing the ratio of the interdependence of the observed values, i.e, the ratio of observed support to expected support if X and Y were independent; if the lift is equal to one, the rule and the items are independent, since no statistically proven relationship is shown, while if the lift is more than one, it indicates a higher dependency [21].

V. RESULTS

In this section, we present the results of the experiments on our dataset using the FP-Growth algorithm. Despite the fact that we gathered data for nearly a month, we identify patterns and correlations only between SQL commands given that the Dionaea honeypot does not allow users (including attackers) to execute all types of commands, but mainly SQL commands.

Based on the identified frequent itemsets, we generated 6020 rules. We filtered these rules using the confidence metric to keep rules that depicted high reliability; by only considering rules with a confidence of 1, we ended up with 3031 rules. Finally, we deleted rules that had lift of 1, as these depict no correlation and cannot considered as useful. Therefore, these evaluation metrics were used to identify rules having a strong relationship between their items (i.e., one item was executed with high probability along with another item) and eventually reduce the number of generated rules thus lowering the complexity accompanying such large numbers of rules.

Table II presents examples of generated rules (not all rules are shown due to space limitations). More specifically, four rules with the highest lift value (i.e., 6.57) and two rules with the lowest lift value (i.e., 5.3) are depicted. Figure 1 depicts the different values of the lift metric. The higher the lift value, the stronger the rule is stronger dependency between the antecedent and consequent parts). From these rules, we can evaluate actions of attackers that are executed together and in a particular sequence. For example the first rule in Table II indicates that if one attacker executes the commands:

1) `EXEC sp_OASetProperty, @objFull,'ControlFlags',4`
2) `Create, 'WbemScripting.SWbemLocator', @objLocator, OUTPUT`

then the attacker will execute with high probability the following commands:

3) `EXEC sp_OAMethod, @objWmi,'Get',@objFull, OUTPUT,'Win32_SecurityDescriptor`
4) `EXEC sp_OAMethod, @objLocator, 'ConnectServer',@objWmi,OUTPUT,'.', 'root\\cimv2`

So with the lift values, we can infer that commands $3, 4$ are more likely to be executed together with commands $1, 2$ rather than commands $1, 2$ to be executed alone. More specifically, in this example, commands $3, 4$ are nearly 7 times more likely to be executed together with commands $1, 2$. In other words, lift declares that the presence of commands $1, 2$ increases the probability that the commands $3, 4$ will be executed. Table II depicts also confidence of 1 on each rule, i.e, if the command on the left part of the rule is present, then the specific command on the right part will follow 100% of the time.

Such generated rules depict sequences of commands executed by different malicious actors in their attempt to achieve a specific goal, such as to gain access to the system by

[5]https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/fpgrowth/
[6]https://pandas.pydata.org/

TABLE II
RULES GENERATED BY ARL THAT INDICATE THE ATTACK PATTERNS FOLLOWED BY THE ATTACKERS

Antecedents	Consequents	Confidence	Lift
EXEC sp_OASetProperty @objFull, ’ControlFlags’,4, Create ’WbemScripting.SWbemLocator’, @objLocator OUTPUT	”EXEC sp_OAMethod @objWmi,’Get’, @objFull OUTPUT,’Win32_SecurityDescriptor’ ”, ”EXEC sp_OAMethod @objLocator, ’ConnectServer’,@objWmi OUTPUT,’.’,’root\\cimv2’ ”	1	6.57
”Create ’WbemScripting.SWbemLocator’, @objLocator OUTPUT ” ’mssqld’	’1433.0’, ”EXEC sp_OAMethod @objWmi,’Get’, @objFull OUTPUT,’Win32_SecurityDescriptor’ ”, ”EXEC sp_OASetProperty @objFull, ’ControlFlags’,4 ”, ”EXEC sp_OAMethod @objPermiss, ’SetSecurityDescriptor’,NULL,@objFull ”, ”EXEC sp_OAMethod @objLocator, ’ConnectServer’,@objWmi OUTPUT,’.’,’root\\cimv2’ ”	1	6.57
”Create ’WbemScripting.SWbemLocator’, @objLocator OUTPUT ”, ”EXEC sp_OAMethod @objPermiss, ’SetSecurityDescriptor’,NULL,@objFull ”, ”EXEC sp_OAMethod @objWmi,’Get’, @objFull OUTPUT, ’Win32_SecurityDescriptor’ ”	”EXEC sp_OAMethod @objLocator, ’ConnectServer’,@objWmi OUTPUT,’.’,’root\\cimv2’ ”	1	6.57
’Drop Procedure sp_password ’, ”Create ’WbemScripting.SWbemLocator’, @objLocator OUTPUT ”, ”EXEC sp_OAMethod @objPermiss, ’SetSecurityDescriptor’,NULL,@objFull ”, ”EXEC sp_OAMethod @objWmi,’Get’, @objFull OUTPUT,’Win32_SecurityDescriptor’ ”	”EXEC sp_OASetProperty @objFull, ’ControlFlags’,4 ”, ”EXEC sp_OAMethod @objLocator, ’ConnectServer’,@objWmi OUTPUT,’.’,’root\\cimv2’ ”, ’mssqld’	1	6.57
...	..	...	...
”EXEC sp_OAMethod @objPermiss, ’SetSecurityDescriptor’, NULL,@objFull ”, ”EXEC sp_OAMethod @objWmi,’Get’, @objFull OUTPUT,’Win32_SecurityDescriptor’ ”, ”EXEC sp_OASetProperty @objFull,’ControlFlags’,4 ”	Drop Procedure sp_password ’, ’mssqld’	1	5.3
”EXEC sp_OAMethod @objLocator, ’ConnectServer’,@objWmi OUTPUT,’.’,’root\\cimv2’	’Drop Procedure sp_password ’, ’1433.0’, ’mssqld’	1	5.3

e.g., first creating an object and then retrieving the security properties of that object. Thus, such rules are very valuable as they can be used by security analysts to identify repeatable attack patterns and consequently assist them in developing the appropriate defence measures against them for protecting their organisation. Hence, the use of ARL in the current attack landscape is particularly beneficial as it can generate patterns of attacks by analysing large-scale data which is crucial given the high and continuously increasing number of daily attacks.

Overall, we identified several patterns and correlations of commands. The number of generated rules depends on the size of the captured dataset and the attacks information used. For example, we used 12 datasets and three different attributes of an attack; depending on research objectives, less or more attack information (attributes) can be used. The final rules that can be used for further analysis also depend on the research needs. In other situations, for example, it is probably beneficial to pay more attention to the frequent item generation part and, more specifically, to the rules' length specified at this stage, since an analyst may be interested in lengths that entail many commands, and thus filters the final rules accordingly.

In addition, this work generated 259 frequent itemsets, while someone may be interested to generate more itemsets to explore even rare or hidden patterns. This can be achieved with the use of the support metric; the lower the value of support, the more the generated frequent itemsets and eventually the more the rules that get generated. However, this increases both the computational complexity, as more memory resources are needed, and the analysis effort by a security expert as many rules must be evaluated for their usability.

Finally, in this work, we joined the data of different days to extract rules from a unified dataset, rather than extract rules from distinct datasets. This may not always be possible though, as unified datasets can increase the memory requirements and complexity of the ARL methods. This is a well known drawback of ARL and different research efforts have tried to deal with it [22], [23]. In such cases, the extraction of rules from distinct datasets would probably be preferable, but more extensive and diverse datasets can arguably lead to more conclusions about attackers' actions.

VI. CONCLUSIONS

In this work, we proposed a framework for identifying attack patterns and building correlations of commands from real attackers targeting a public honeypot by utilising an ARL algorithm and a honeypot. To our knowledge, this is the first time ARL has been used to extract attack patterns from commands executed by attackers using real-world log data

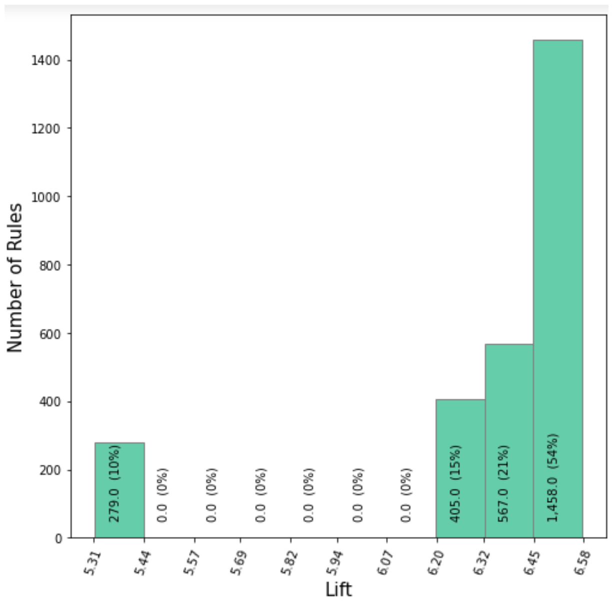

Fig. 1. Distribution of lift values of rules

collected at the host level. We gathered attack data for nearly a month (25 days). In order to determine the attackers' patterns, we use several characteristics of their attacks, such as the SQL commands they executed, the connection protocol they used, and the connection port they targeted. The results derived from the experiment were evaluated using different evaluation metrics, namely support, confidence, and lift. The results showed that the ARL procedure is suitable for identifying different attackers' actions as the generated rules reveal several attack patterns. Our next step is to map the generated rules to the MITRE ATT&CK framework to allow for (i) the better organisation of the extracted information, thus facilitating the identification of the general behaviour and motivation behind the attacks patterns that we identified, and (ii) the storage of the collected intelligence in a format that can more easily be shared and exploited by other organisations. In addition, as our dataset is constantly updated with new data, we plan to utilise datasets of more extended periods and analyse the trends in the detected patterns and behaviour of attackers. Finally, besides the real-world host-based data used in this work, we also plan to use network-based data to reveal additional attack patterns.

Acknowledgment

This work was supported by the FORESIGHT (H2020-833673) and ECHO (H2020-830943) projects, funded by the European Commission.

References

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[2] M. H. Nasir, S. A. Khan, M. M. Khan, and M. Fatima, "Swarm intelligence inspired intrusion detection systems—a systematic literature review," *Computer Networks*, p. 108708, 2022.

[3] F. S. Tsai, "Network intrusion detection using association rules," *International Journal of Recent Trends in Engineering*, vol. 2, no. 2, p. 202, 2009.

[4] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, 1993, pp. 207–216.

[5] M. Husák, J. Kašpar, E. Bou-Harb, and P. Čeleda, "On the sequential pattern and rule mining in the analysis of cyber security alerts," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 2017, pp. 1–10.

[6] Q. Liu, V. Hagenmeyer, and H. B. Keller, "A review of rule learning-based intrusion detection systems and their prospects in smart grids," *IEEE Access*, vol. 9, pp. 57 542–57 564, 2021.

[7] K. Sindhu Meena and S. Suriya, "A survey on supervised and unsupervised learning techniques," in *International Conference on Artificial Intelligence, Smart Grid and Smart City Applications*. Springer, 2019, pp. 627–644.

[8] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of rule learning*. Springer Science & Business Media, 2012.

[9] J. Fürnkranz and T. Kliegr, "A brief overview of rule learning," in *International symposium on rules and rule markup languages for the semantic web*. Springer, 2015, pp. 54–69.

[10] P. Lou, G. Lu, X. Jiang, Z. Xiao, J. Hu, and J. Yan, "Cyber intrusion detection through association rule mining on multi-source logs," *Applied Intelligence*, vol. 51, no. 6, pp. 4043–4057, 2021.

[11] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[12] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215. Citeseer, 1994, pp. 487–499.

[13] C. Fachkha, E. Bou-Harb, A. Boukhtouta, S. Dinh, F. Iqbal, and M. Debbabi, "Investigating the dark cyberspace: Profiling, threat-based analysis and correlation," in *2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)*. IEEE, 2012, pp. 1–8.

[14] C.-B. Jiang, I.-H. Liu, Y.-N. Chung, and J.-S. Li, "Novel intrusion prediction mechanism based on honeypot log similarity," *International Journal of Network Management*, vol. 26, no. 3, pp. 156–175, 2016.

[15] S. Rathee and A. Kashyap, "Adaptive-miner: an efficient distributed association rule mining algorithm on spark," *Journal of Big Data*, vol. 5, no. 1, pp. 1–17, 2018.

[16] S. Kumar, B. Janet, and R. Eswari, "Multi platform honeypot for generation of cyber threat intelligence," in *2019 IEEE 9th International Conference on Advanced Computing (IACC)*. IEEE, 2019, pp. 25–29.

[17] S. Mabu, C. Chen, N. Lu, K. Shimada, and K. Hirasawa, "An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming," *IEEE transactions on systems, man, and cybernetics, part C (Applications and Reviews)*, vol. 41, no. 1, pp. 130–139, 2010.

[18] K. M. M. Aung and N. N. Oo, "Association rule pattern mining approaches network anomaly detection," in *Proceedings of 2015 International Conference on Future Computational Technologies (ICFCT'2015) Singapore*, 2015, pp. 164–170.

[19] J. M. Luna, M. Ondra, H. M. Fardoun, and S. Ventura, "Optimization of quality measures in association rule mining: an empirical study," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, p. 59, 2018.

[20] M. Hahsler, "A probabilistic comparison of commonly used interest measures for association rules," *United States. Southern Methodist University*, 2015.

[21] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic itemset counting and implication rules for market basket data," in *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, 1997, pp. 255–264.

[22] H. Li and P. C.-Y. Sheu, "A scalable association rule learning heuristic for large datasets," *Journal of Big Data*, vol. 8, no. 1, pp. 1–32, 2021.

[23] S. Patel Tushar, P. Mayur, L. Dhara, K. Jahnvi, D. Piyusha, P. Ashish, P. Reecha, S. Tushar, P. Mayur, and L. Dhara, "An analytical study of various frequent itemset mining algorithms," *Res J Computer & IT Sci*, vol. 1, no. 1, pp. 6–9, 2013.