BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 67 (71), Numărul 4, 2021
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ
DOI:10.2478/bipie-2021-0020

APPLICATION OF ASSOCIATION RULE MINING IN PREVENTING CYBERATTACKS

BY

CĂTĂLIN MIRONEANU[∗], ALEXANDRU ARCHIP and GEORGIANA ATOMEI

“Gheorghe Asachi” Technical University of Iași,
Faculty of Automatic Control and Computer Science

Received: December 11, 2021
Accepted for publication: December 29, 2021

Abstract. Designing a security solution should rely on having a good knowledge of the protected assets and better develop active responses rather than focus on reactive ones. We argue and prove that malicious activities such as vulnerabilities exploitation and (D)DoS on Web applications can be detected during their respective initial phases. While they may seem distinct, both attack scenarios are observable through abnormal access patterns. Following on this remark, we first analyze Web access logs using association rule mining techniques and identify these malicious traces. This new description of the historical data is then correlated with Web site structure information and mapped over *trie* data structures. The resulted *trie* is then used for every new incoming request and we thus identify whether the access pattern is legitimate or not. The results we obtained using this proactive approach show that the potential attacker is denied the required information for orchestrating successful assaults.

Keywords: cybersecurity; data mining; association rule mining; proactive security; IDPS.

[∗]Corresponding author; *e-mail*: catalin.mironeanu@academic.tuiasi.ro

1. Introduction

Today's cyber-attacks are increasingly complex and run in several phases. There are two complementary defensive approaches: monitoring and analyzing hosts activity, usually solved by HIDS (Host-based Intrusion Detection Systems), and network activity, which is the duty of NIDS (Network-based Intrusion Detection Systems). Both solutions are commonly known as IDS. Main goal of IDS is detecting abnormal activities or known malware occurrences for issuing alerts. Modern solutions include an active response component for blocking or even tangling sources of attacks. These defense approaches also include an offensive module which offers a reaction to known threats and are known as IDPS (Intrusion Detection and Prevention Systems). All earlier mentioned solutions are usually integrated into SIEMs (Security Information and Event Management) for further analysis and near real-time automated responses.

It is known that ensuring computer systems security is a goal that is never fully achieved, but it must be done in a proper way that blocks almost all cyberattacks. Common defense techniques are known by experienced attackers that make misconfigured IDS/IDPS become obsolete on their own. It is a quite common situation in which IDS/IDPS does not detect attacks or are bypassed by attackers' evasion techniques. Thus, defenders must continuously analyze their systems and be aware of all attacker’s methods and techniques. From this perspective, Bruce Schneier’s statement from the early 2000s that “security is a process, not a product” (Schneier, 2000) is still relevant. To strengthen this idea, according to the National Institute of Standards and Technology, the NIST Cybersecurity framework was crafted in a recursive manner on its core components – Identify, Protect, Detect, Respond, and Recover. The main purpose of this framework is to reduce the cyber risk and to improve the security to critical infrastructure (Barrett, 2018).

The first idea spawned from the “security is a process” concept, was carried out by the Lockheed Martin Corporation in 2011 in the development of Cyber Kill Chain Framework (CKC) – an IT reworking of the military Find Fix Track Target Engage Assess (F2T2EA) term (Hutchins *et al.*, 2011). CKC described 7 stages: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command & Control, and Actions on Objectives. Each stage describes the potential actions that an attacker might engage to conduct a potentially malicious attempt. Each stage is observable on the so-called victim of the attack apart from the Weaponization phase which is usually hidden. It has been noticed that even modern security solutions are rather reactive: defensive measures are noticeable during and after the Delivery stage. Paul Pols extends the CKC into the Unified Kill Chain (UKC) in (Pols, 2017). Pols proposes a 17 phases description of attacks divided into 3 major groups: Initial Foothold, Network Propagation and Action on Objectives. UKC is focused more on modelling advanced persistent threats and includes indirect attack patterns that target not only the computing systems, but also the human users of those systems. Both UKC and CKC frameworks begin with the same three phases (Hutchins *et al.*, 2011; Pols, 2017). This makes perfect sense since orchestrating a successful attack relies on analyzing the target to identify potential weaknesses, on establishing a course of action and then on delivering malicious payloads. While the actions included in the Reconnaissance phase may not be incriminating, we argue that they provide valuable data. A thorough analysis of these data could prove to be a valuable asset in preventing cyberattacks (Mironeanu *et al*., 2021; Mironeanu, 2021). The solution we present in this paper is one such example of proactive prevention.

The second fundamental concept that spawned from the "Security is a process" idea is to focus on understanding the huge amount of gathered log data by using DM (Data-Mining), big data and AI/ML (Artificial Intelligence/Machine Learning) techniques (Kabanda, 2020). The increasing number of cyber-attacks and related security events whose material traces are founded in logs, and the increasing variety of attack patterns are met with renewed research efforts in developing new, better, and more robust preventing solutions.

Gathering massive amounts of network traffic data and then analyzing it through DM techniques to determine valuable information related to Reconnaissance and Delivery phases is nowadays accessible. The evolution of hardware distributed resources and better algorithms that can be implemented in modern solutions support blocking attack sources and hindering attackers' efforts to gain knowledge on the defended systems and thus could prove invaluable in undermining a full weaponization phase. Because some attacker actions are mixed in Reconnaissance and Delivery phases, we will consider both as consistent sources of data.

2. Related Work and Similar Approaches

Data Mining (DM) and Machine Learning (ML) have some common features, but others that are fundamentally different. The common core is data. Both processes are used for solving complex problems. Another common feature is that both processes use the same algorithms for discovering patterns in data. The main difference between DM and ML is the target of the results. While DM techniques offer results useful for human users, ML focuses on automating data analysis and decision systems. ML is thus a next stage over DM. It is a well-known fact (Han *et al.,* 2012), especially for data mining techniques, that these methods of data analysis are classified into descriptive and predictive techniques. This grouping is generated by the type of results that are supplied. Thus, descriptive techniques are used to obtain new features and perspectives on the data already accumulated, while predictive techniques are aimed at characterizing new elements.

The idea of using DM/ML techniques in strengthening security solutions is not new. In a series of 11 articles, Wenke Lee starting with (Lee and Stolfo, 1998) and ending with (Lee *et al.*, 2002; Lee, 2002) emphases the idea that most analytical approaches are based on classic DM tools and on using these results for improving IDS/IDPS solutions. In (Lee and Stolfo, 1998) the focus is on audit data, such as access logs, protocol-specific data (*i.e.*, FTP and SMTP), or dedicated monitoring commands (such as tcpdump) to build classifiers or to mine association rules. The purpose of these tasks is to contribute to building a set of decision rules for a potentially automated agent-based IDS. Further research (Lee *et al.*, 2002) shows how modified association rule mining and frequent episodes can be improved and therefore strengthen the aforementioned IDS. The achieved results are the base building-blocks in determining features of intrusion patterns, features that are then used for intrusion detection models. Wenke Lee summarizes both the progress and the issues that also arise from using DM and incipient forms of ML in IDS/IDPS: efficiency and credibility. DM/ML are mostly statistical instruments that could yield false-positives and false-negatives (Lee, 2002).

Both descriptive and predictive DM techniques are employed in security analysis (Jin *et al.*, 2019). In a comprehensive comparison between 12 approaches that identify the strengths and drawbacks related to different types of attacks, the survey concluded with two important remarks. The first is that one must pay a great deal of attention to the actual preprocessing stages of data selection and data modelling when dealing with DM tools and algorithms. These stages definitely impact on the quality of the results and on the success of the solution. The second remark is that a single analysis technique is usually insufficient in providing trustworthy information. Both descriptive and predictive solutions must be considered and mixed before reaching a security decision. A rather recent and comprehensive review on the usage of these techniques is included in (Dasgupta *et al.*, 2020). The authors presented the most commonly used ML algorithms in cybersecurity by considering the basics of the algorithms, the underlying DM techniques, and the applications in an extensive analysis of the classification techniques used in the field of cyber security.

Modern security solutions, that follow the “security is a process” idea, must be designed having two main concepts in mind, as we have previously shown. The first one is to *understand* the entity that needs to be protected against unwanted/malicious access. A thorough knowledge on the hardware and software technologies that form the protected target is indeed required. It may allow a much better understanding of the strengths and of the weaknesses/ vulnerabilities that could be exploited. The second main concept is to think proactively and focus on prevention rather than detection. One should focus on understanding how a potential attacker might think and on identifying any potential malicious activity as early as possible. The following two sections of the paper describe our proposed prototype for actively identifying threats on Web applications and the preliminary results we have achieved. The reason for focusing our attention on this type of application is that the Web has become an ever-important tool in our day-to-day lives. This "openness" is also an Achiles' heel: Web applications are the most exposed entities to potential attackers (Widup *et al.*, 2021).

3. A Theoretical Perspective of Our Approach

The CKC and its derivatives clearly show that attack patterns may be observed during phases 1 ("Reconnaissance") and 3 ("Delivery") respectively. While the second phase ("Weaponization") is hidden from the victim, any attacker must first identify its target vulnerabilities and then deliver the so-called payload of the attack. Let us consider the case of a Web server hosting a PHP (Hypertext Preprocessor) application and two common attack patterns: vulnerability exploitation and (Distributed) Denial of Service ((D)DoS).

To exploit some known application weakness, the potentially malicious actor must first find that the target vulnerability is present within the Web application. (Shustin, 2019) and (Johannes, 2021) present such examples for the Web application included in some IPTV sets and list both .php and .js files. (Jost, 2021) also lists a vulnerable WordPress plugin which is, again, developed using PHP. All three authors target SQL injection and Remote Code Execution (RCE) attack patterns. A successful malicious attempt must first identify that the vulnerable files or the vulnerable software version is present on the target device. This implies that the hosting server would yield log entries having either HTTP GET or POST requests that directly target the known susceptible application files. This is a key observation since it allows us to find the traces of such attempts through using frequent access patterns in log entries.

(D)DoS attacks are presumably well known and considered to be mostly "Command and Control" and/or "Actions on Objectives" phases with respect to CKC (Sfakianakis *et al.*, 2018). Complex attacks could use (D)DoS as a "Reconnaissance" tool to test the target's response and defense capabilities. (D)DoS might also be employed to conceal the delivery of more destructive payloads (Mironeanu, 2021; Sfakianakis *et al.*, 2018). HTTP based (D)DoS is usually performed through GET and POST verbs – the same predicates that are commonly used for vulnerability exploitation. HTTP GET Floods – a (D)DoS type of attack – are of particular interest. Such assaults are usually performed through many requests per second that target the victim Web server. The interesting part of HTTP GET Flood is the actual vulnerability that is exploited. Attackers are not usually concerned with the actual Web resource they target, but rather with missing resources and slow/poorly configured support databases. The reason supporting this claim is that the Web server would be constrained to process faulty requests more often than legitimate ones. Consider, for instance, the case of an SQL database with no indexing: targeting a non-existent record would incur a slow response from the DB server, which would, in turn, keep at least one of the Web server's processing threads in a running state until a 404 Not Found response may be formulated back to the calling client. These flood attacks are, again, noticeable through frequent access patterns.

Let us now consider the case of how PHP works in delivering a response to the calling client. PHP allows developers a high degree of modularity. A PHP script could include other PHP *modules* that implement the required functions. The result is usually presented as an HTML page that, in turn, includes various links to .js, .css and media files. We call this first HTML page an *aggregated resource* (or *aggregator*), and all the included links *components*. A client's browser (or user agent) would then analyze the *aggregator's* code and would then formulate further HTTP requests to the target Web server to acquire its respective *components*. Considering this behavior, the following key remarks are the reason for our approach:

1. the client's browser would include the *aggregator* as the referrer for each *component* it would require; this value is passed to the server in the HTTP Referer request header and it is noticeable in the server log files (Fielding and Reschke, 2014);
2. media *components* (such as images and/or movies) may be acquired independently from the *aggregator*;
3. the PHP *modules* that contribute to the *aggregator* are not requested by the client's browser and should therefore never be noticeable in the server log files.

Furthermore, a legitimate client request would either target the URI of the *aggregator* or an URI of a media *component*. In such cases, a legitimate request may not include an HTTP Referer request header. An attacker attempting to exploit a known vulnerability would most likely try to determine if the vulnerable *module(s)* or *component(s)* are present on the victim server – the "Reconnaissance" phase previously described. The malicious actor would most likely use automated site scrapper which would target known *modules* and/or *components* without the required HTTP Referer header. A (D)DoS attack based on HTTP GET would most likely target invalid URIs to either *aggregators*, either *components*/*modules* as we have shown.

To summarize, our theoretical model is built on the following access patterns:

1. *legitimate clients*:
    a. **target** *aggregator URI* **with/without** a valid referrer;
    b. **target** *component URIs* **with** a valid referrer;
    c. **never target** *module URIs*;
2. *potential attackers*:
    a. **target** *non-existent aggregator URIs*;

b. **target** *component URI* **without** valid referrers;
c. **target** *modules*.

We therefore deduce that a legitimate access would use a valid HTTP method, that would either target an *aggregator* (or *valid* or *external*) resource, either employ a valid referrer for *components* (or *internal* resources). Given the fact that a legitimate client would most likely browse a site using the hypermedia links included in pages (and assuming that those pages do not have misspelled links), then there is an extremely low probability that a legitimate client would repeatedly request invalid *public* or *external* resources. Any potential perpetrator attempting a vulnerability scan or an HTTP GET (D)DoS would either target *modules* directly or *components* (or *internal* resources) without valid referrer URIs, either have a considerably higher rate of requesting invalid *public* or *external* resources. We require only three items to model these access patterns: HTTP_METHOD, REFERRER_TYPE and TARGET_TYPE. Their corresponding values are presented in Table 1.

Table 1
Item labels, Values and Reasoning

Label	Values	Target Access Pattern	Reasoning
HTTP_METHOD	*Admissible HTTP method names or INVALID*	*all*	*A valid HTTP method is present in all scenarios. An INVALID method indicates an attack or otherwise bad request.*
REFERRER_TYPE	*VALID*	*1.a÷c*	*A legitimate client request includes the corresponding referrer for components.*
	INVALID | NULL	*1.a; 2.a÷c*	*A legitimate client may target an aggregator or a media component without a referrer. An attacker would have NULL or INVALID referrals for modules and non-media components.*
TARGET_TYPE	*VALID | EXTERNAL*	*all*	*Both legitimate clients and attackers could target existing aggregators.*
	INTERNAL	*2.b÷c*	*Attackers attempt and identify vulnerabilities by directly requesting components.*
	INVALID	*2.a; 2.c*	*An attacker could use module identifiers as targets.*

We would like to explain a bit further the idea of a ***VALID*** referrer URL. Let us consider two pages, denoted A and B. Page B includes a link to a file named “sample.js”, while page A *does not include* the same JavaScript file. In such a case, B *is* a ***valid*** referrer for *component* “sample.js” and, at the same time, A *is not* a ***valid*** referrer for the same component. This strengthens the reasoning of our approach, hinting on the context of an aggregated resource: modules and components are used for designated aggregators. It would therefore make no sense to have referrer information outside the scope of an aggregator for such a scenario.

We first validated this model by analyzing Apache logs using the association rule mining descriptive DM technique. This analysis had been performed using a custom implementation of the Apriori algorithm (Agrawal & Srikant, 1994) which we have derived from (Li *et al.*, 2012; Mao and Guo, 2013). This implementation had been run on an Apache Hadoop (***, 2021; Dean and Ghemawat, 2008). We analyzed itemsets made up of 3 items (see Table 1), having the from:

{HTTP_METHOD, REFERRER_TYPE, TARGET_TYPE}.

We have also set up a fixed form for the association rules:
antecedent
{HTTP_METHOD, REFERRER_TYPE}
consequent
{TARGET_TYPE}.

We noticed that for the two studied attacks (vulnerability scanning and (D)DoS) the consequent of the rule had the *INVALID* value for most cases describing a malicious attempt. This supports the claims of the proposed model since it clearly distinguishes between legitimate and illegitimate access. More on these results will be discussed in the following Section 4.

The next step we addressed was to transform the descriptive association rules into pro-active instruments that would allow a prompt detection and a corresponding response for such attacks. Our target is to perform near-real time detection on client network traffic. To reach our target, we make use of *trie* data structures (La Rocca, 2021):

- each **level** in the *trie* represents a possible transition between the items in an interesting itemset;

- each **edge** between a *parent* and a *child* of the *trie* is labeled with the admissible values of the corresponding item – see Table 1:
    - *trie root* to *1st level children*: admissible values for the HTTP_METHOD item (in antecedent);
    - *1st level children* to *2nd level children*: admissible values for the REFERRER_TYPE item (in antecedent);
    - *2nd level children* to *3rd level children*: admissible values for the TARGET_TYPE item (in consequent);
- each of the **trie's data pointers** holds the confidence of the corresponding rule and a label showing whether the rule describes an attack or a legitimate access.

Fig. 1 describes the necessary steps to map an association rule into a trie path. The *transform* function (Fig. 1, line 1) is a basic split-like function that isolates the items in the antecedent and the consequent of the rule. Fig. 2 includes the algorithm we have devised to perform an automated labeling of the *data pointers*.

```
Require: trie_root, rule, confidence
1: $trie\_path, data \leftarrow transform(rule, confidence)$
2: $navptr \leftarrow trie\_root$
3: for $path\_item \in trie\_path$ do
4:     if $navptr[path\_item] = NULL$ then
5:         $navptr[path\_item] \leftarrow new\_node$
6:     end if
7:     $navptr \leftarrow navptr[path\_item]$
8: end for
9: $navptr.data \leftarrow data$
```

Fig. 1 – Rule insertion algorithm.

```
Require: rule, confidence
1: $trie\_path \leftarrow rule.antecedent \cup rule.consequent$
2: $data \leftarrow \{confidence\}$
3: if ($trie\_path[referrer] = NULL$ AND $rule.consequent \in \{INTERNAL, INVALID\}$)
   OR $trie\_path[http\_method] \notin \{allowed\_http\}$ then
4:     $data \leftarrow data \cup \{ATTACK\}$
5: else
6:     $data \leftarrow data \cup \{NORMAL\}$
7: end if
8: return $trie\_path, data$
```

Fig. 2 – Data pointer labeling algorithm.

Fig. 3 and Fig. 4 show symbolic examples of the desired output.

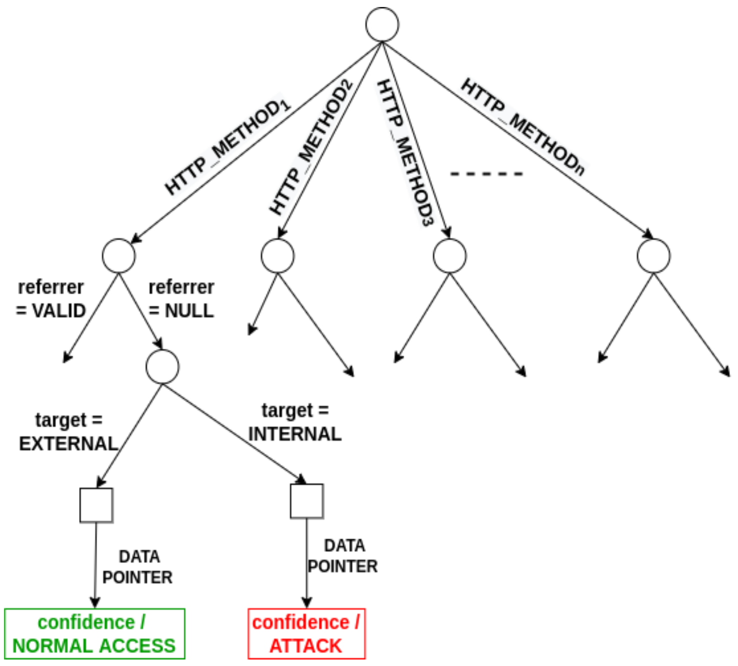

Fig. 3 – Trie output for the “Reconnaissance” stage of vulnerability scanning attack.

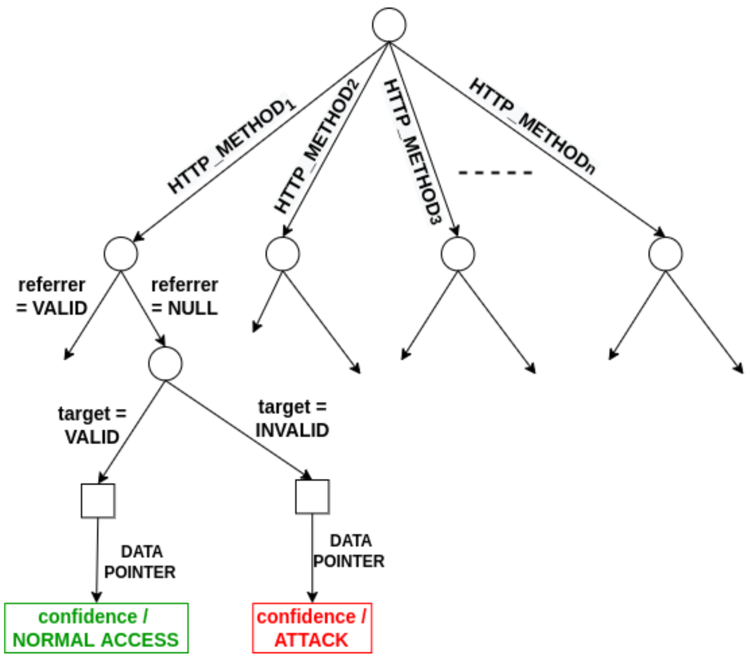

Fig. 4 – Trie output for (D)DoS attacks.

Due to the many similarities between the two attack patterns, we were also able to merge the two *tries* into one – Fig. 5. This is indeed a desirable outcome since it allows us to identify both types of malicious actions in a single stage rather than being forced to perform two distinct searches on two distinct data structures.

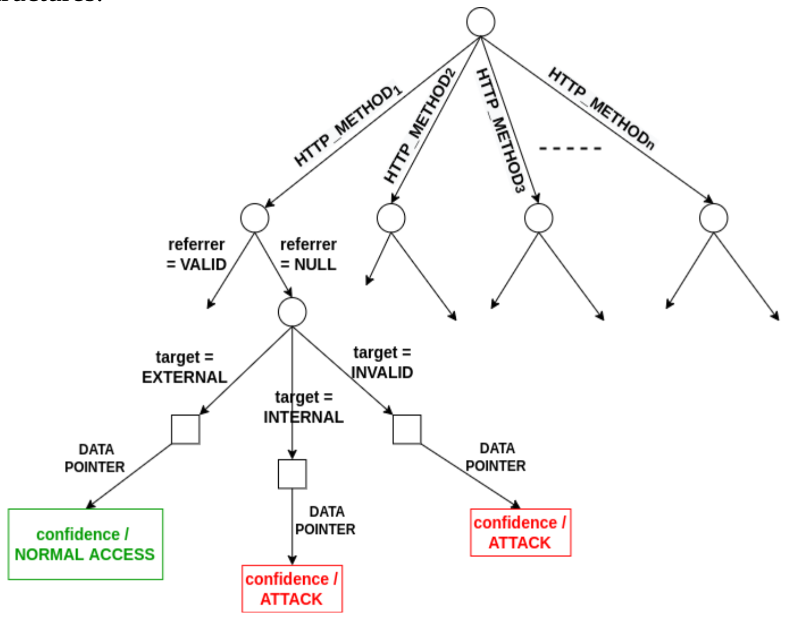

Fig. 5 – Combined trie output for both studied attacks.

One could easily check incoming requests using well-known tools such as tcpdump, tshark and so on. Having access to such a request, the HTTP method and the target resource are directly accessible in the *HTTP request line* and the *HTTP Referer header* is, yet again, directly accessible within that same request data. We then select and label these values to form an *access pattern* that closely resembles the format of the itemsets. Subsequently, we use the *access pattern* to search the *trie* as presented in the algorithm in Fig. 6.

```
Require: trie_root, access_pattern
1: $path \leftarrow to\_path(access\_pattern)$
2: $navptr \leftarrow trie\_root$
3: for $path\_item \in path$ do
4:     if $navptr[path\_item] = NULL$ then
5:         return $NULL$
6:     end if
7:     $navptr \leftarrow navptr[path\_item]$
8: end for
9: return $navptr.data$
```

Fig. 6 – Trie search algorithm.

If the search path yields a *data pointer* labeled “NORMAL ACCESS”, then the calling IP is allowed to follow through with its respective request. If, on the other hand, we find an “ATTACK” label, then the calling IP is blocked and any later request is not allowed any further.

4. Preliminary Results

The test scenario included the following components. The so-called *victim* (or target) of the attacks was a server hosted on CentOS Linux running Apache 2.4.6 with PHP 7.4.21. The Apache log format was configured to use the well-known *custom* format for the *access logs*. The actual website was delivered using the popular WordPress CMS version 5.7. The web server also hosted the ECAD monitoring agents (Mironeanu *et al.*, 2021; Mironeanu, 2021) responsible for real-time network traffic capture and for implementing the decisions issued by the underlying layers. Normal access patterns have been both simulated using httrack on three different workstations and human client interaction. The attacks were simulated using the following tools:

- **OpenVas** (Greenbone OS 6.0.10) - vulnerability scanning and semantic DoS;
- **Nikto –** vulnerability scanning and DoS (“-Tuning 6” argument set);
- **GoBuster –** vulnerability scanning and DoS (by brute forcing with “raft-medium-words-lowercase” wordlist string argument set).

All these tools had been run using three different workstations. During the first testing phase, the ECAD agents were turned off. We have obtained an access log file that stored a total of 485,798 entries. Over 400,000 entries corresponded to vulnerability scanning (“Reconnaissance” stage) or DoS attacks (“Reconnaissance” / “Delivery” stages). All these entries have been remodeled as presented in Section 3 and then processed using the MapReduce Apriori solution we have mentioned earlier. We obtained a total of 16 relevant association rules with a minimum confidence threshold of 0.7 – Table 2.

The labels in the last column have been set up using the algorithm presented in Fig. 2. The correctness of our approach is proven by considering the following remarks:

1. rules no. 1 and 7 through 16 are obtained from log entries that do not include an admissible HTTP method for the request;
2. rules no. 2 and 4 are obtained from log entries that include requests that target specific PHP modules (which ***should not*** be present in valid requests) or internal components without a valid HTTP Referer header;
3. rules no. 3, 5 and 6 are obtained from log entries that include valid, external web pages (or *aggregator resources*) that may be accessed by all legitimate clients.

Table 2
Identified association rules

No.	Antecedent	Consequent	Confidence	Label
1	*0_- 1_null*	*2_invalid*	*1.0*	*ATTACK*
2	*0_GET 1_null*	*2_invalid*	*1.0*	*ATTACK*
3	*0_GET 1_valid*	*2_external*	*0.82*	*NORMAL*
4	*0_POST 1_null*	*2_invalid*	*1.0*	*ATTACK*
5	*0_POST 1_valid*	*2_external*	*1.0*	*NORMAL*
6	*0_TRACE 1_valid*	*2_external*	*1.0*	*NORMAL*
7	*0_VTTEST 1_valid*	*2_external*	*1.0*	*ATTACK*
8	*0_\x16\x03 1_null*	*2_invalid*	*1.0*	*ATTACK*
9	*0_\x16\x03\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
10	*0_\x16\x03\x01\x03\xa1\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
11	*0_\x16\x03\x01\x03\xb9\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
12	*0_\x16\x03\x02\x03\xa1\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
13	*0_\x16\x03\x02\x03\xb9\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
14	*0_\x16\x03\x03\x03\xc7\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
15	*0_\x16\x03\x03\x03\xdf\x01 1_null*	*2_invalid*	*1.0*	*ATTACK*
16	*0_some 1_null*	*2_invalid*	*1.0*	*ATTACK*

Another interesting remark is that *all* rules labeled "attack" have a 100% confidence score. This implies a strong correlation between the items included in the antecedent and consequent of the rule, respectively, and further shows the validity of the proposed access pattern. Furthermore, please notice rule no. 3 in Table 2. The antecedent of the rule includes a valid, safe HTTP method (namely GET) and the target of the request in the consequent is a valid *aggregator*/*external* resource. One may be certain that such a request is normal and would expect a 100% confidence degree. Despite this apparent behavior, we only achieved a confidence score of only 82%. Indeed, some attack patterns may be hidden behind seemingly legitimate traffic.

The second set of tests we performed targeted the near-real time detection of the attack patterns we studied. We have started the ECAD agents on the monitored Web server and reinitiated the attacks. All incoming requests were processed in agreement with the desired pattern we have presented in Section 3. The resulting data was then sent to the ECAD decision agents that hosted the *trie* data structure. The results – see the algorithm in Fig. 6 – were sent back to the monitoring agents which, in turn, allowed the traffic to pass (NORMAL access) or issued a *DROP connection* like action (if an ATTACK rule had been matched). Vulnerability scanning workstations were banned after only 5 successful requests, while DoS ones were dropped after 9 to 15 successful requests.

This is a particularly significant result. We have studied the data collected by the vulnerability scanning workstations. There had been no meaningful output related to the WordPress CMS and the monitored Web server, which means that a potential attacker did not gather any significant information during the “Reconnaissance” stage. Also, the active ECAD agents denied the DoS attacks after at most 15 malicious attempts. If we consider the fact that more than 400,000 entries in the gathered logs were DoS attempts, stopping such attempts in just under 1% of the total traffic volume is a great achievement. While HTTP GET based DoS is easily denied by modern IDPS solutions and while we are aware that these results had been achieved in a simulated, controlled scenario, it is still a remarkable output for the solution we have presented.

5. Conclusion and Further Development

The present study is a fork of the research conducted while developing the ECAD framework – previously published in (Mironeanu e*t al.*, 2021). For this present study, we have focused on the analytical modules included in ECAD. A new perspective in using descriptive DM techniques (association rule mining) and a strong knowledge on the monitored Web server’s behavior allow us to develop an innovative approach for preventing cyberattacks. We emphasize the fact that knowing the entity one tries to defend is of utmost importance in implementing Schenier’s “security is a process” concept. We relied our reasoning on the way a PHP Web application is built and ran on Apache, on how an *aggregator* resource uses internal modules to formulate responses and on identifying the relation between the .js and .css components and the actual Web pages they belong to. We also consider the adoption of the *trie* data structure to map association rules and then use them to predict a client’s behavior to be a novel approach in applying DM techniques in security applications. This is a proactive behavior in cybersecurity allowing active responses to threats. One last remark on the current state of our research is that we have relied on historical data (*i.e.*, server log files) to achieve our goals.

There are still many future directions to further this study. The ECAD framework allows a multi-criterial analysis of incoming requests. We have focused only on application-level data (*i.e.*, Layer 7 data with respect to the OSI stack), but different attacks may also yield valuable information on other OSI stack layers. We could also analyze the way the underlying TCP connection is established or finished (Layer 3 data). If we correlate this information with the behavior of the client in receiving the server's response, we may yet better distinguish between normal and attack patterns. This would allow us to correlate HIDS and NIDS solutions and achieve a higher degree of prevention and protection. Also, one may check the actual behavior of Web clients in accessing a Web site and determine whether they follow a legitimate pattern or not. Historical log data processed through descriptive DM techniques is an invaluable source of information that could be used in strengthening SOC operations.

Acknowledgements. Cătălin Mironeanu has developed the ECAD framework architecture and formulated the theoretical and practical models for the attack patterns. Alexandru Archip has developed the theoretical model for the itemsets and association rules and defined and implemented the trie data structure. Georgiana Atomei contributed to the design of the MapReduce Apriori solution and has handled its practical implementation, testing and results.

REFERENCES

Agrawal R., Srikant R., *Fast Algorithms for Mining Association Rules in Large Databases*, In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 487-499.

Barrett M., *Framework for Improving Critical Infrastructure Cybersecurity Version 1.1*, NIST Cybersecurity Framework, 2018 (available online, https://doi.org/10.6028/NIST.CSWP.04162018, last accessed: October 2021).

Dasgupta D., Akhtar Z., Sen S., *Machine Learning in Cybersecurity: A Comprehensive Survey*, The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, **19**, 1, 57-106 (2020).

Dean J., Ghemawat S., *MapReduce: Simplified Data Processing on Large Clusters*, Commun. Association for Computing Machinery, New York, NY, USA, **51**, 1, 107-113 (2008).

Fielding R.T., Reschke J., Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, RFC 7231, ISSN: 2070-1721, June 2014.

Han J., Kamber M., Pei J., *Data Mining: Concepts and Techniques*, ITPro Collection, Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann Publishers, 3rd Edition, 2012.

Hutchins E., Cloppert M., Amin R., *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains*, Proceedings of the 6th International Conference on I-Warfare and Security, Washington, DC, USA, 17–18 March 2011, 113-125.

Jin Z., Cui Y., Yan Z., *Survey of Intrusion Detection Methods Based on Data Mining Algorithms*, In Proceedings of the 2019 International Conference on Big Data Engineering (BDE 2019). Association for Computing Machinery, New York, NY, USA, 98-106.

Johannes U., *Who Is Hunting for Your IPTV Set-Top Box?*, SANS ISC InfoSec Forums, 2021 (available online: https://isc.sans.edu/forums/diary/Who+Is+Hunting+For+Your+IPTV+SetTop+Box/27912/, last accessed: October 2021)

Jost R., *WordPress Plugin Secure Copy Content Protection and Content Locking 2.8.1 - SQL-Injection (Unauthenticated) - CVE 2021-24931*, Exploit Database, 2021 (available online: https://www.exploit-db.com/exploits/50733, last accessed: December 2021).

Kabanda G., *Performance of Machine Learning and other Artificial Intelligence Paradigms in Cybersecurity*, Oriental Journal of Computer Science and Technology, **13**, 1, 1–21 (2020).

La Rocca M., *Advanced Algorithms and Data Structures*, Manning Publications, 173-217, 2021.

Lee W., Stolfo S.J., *Data-Mining Approaches for Intrusion Detection*. In 7th USENIX Security Symposium, SSYM'98, USENIX Association, Berkeley, CA, USA, 1998, volume 7, 6–21.

Lee W., Stolfo S.J., Mok K.W, *Algorithms for Mining System Audit Data*, In data-mining, Rough Sets and Granular Computing; Physica-Verlag GmbH: Heidelberg, Germany, 2002, 166-189 (2002).

Lee W., *Applying Data-Mining to Intrusion Detection: The Quest for Automation, Efficiency, and Credibility,* SIGKDD Explor. 2002, **4**, 35–42.

Li N., Zeng L., He Q., Shi Z., *Parallel Implementation of Apriori Algorithm Based on MapReduce,* 2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 08-10 August 2012, Kyoto, Japan, 236–241.

Mao W., Guo W., *An Improved Association Rules Mining Algorithm Based on Power Set and Hadoop*, 2013 International Conference on Information Science and Cloud Computing Companion, 07-08 December 2013, Guangzhou, China, 236-241.

Mironeanu C., Archip A., Amarandei C.M., Craus M., *Experimental Cyber Attack Detection Framework*, Electronics, **10**, 14:1682 (2021).

Mironeanu C., *Prevenirea atacurilor cibernetice cu tehnici de data mining,* Teza de doctorat, Universitatea Tehnică "Gheorghe Asachi" din Iaşi, 2021.

Pols P., *The Unified Kill Chain - Designing a Unified Kill Chain for Analyzing, Comparing and Defending Against Cyber Attacks*, MSc. Degree Thesis, Delft University of Technology, 2017.

Schneier B., *The Process of Security,* 2000 (available online: https://www.schneier.com/essays/archives/2000/04/the_process_of_secur.html, last accessed: October 2021).

Sfakianakis A., Douligeris C., Marinos L., Lourenço M., Raghimi O., *ENISA Threat Landscape Report 2018*, Report O.1.2.1, European Union Agency for Network and Information Security, Heraklion, Greece, 47-53.

Shustin R., *We Decide What You See: Remote Code Execution on a Major IPTV Platform*, Check Point Research, 2019 (available online: https://research.checkpoint.com/2019/we-decide-what-you-see-remote-code-execution-on-a-major-iptv-platform/, last accessed: October 2021).

Widup S., Pinto A., Hylender C.D., Basset. G., Langlois P., *Verizon Data Breach Investigations Report*, Verizon USA, 2021 (available online:

https://www.verizon.com/business/resources/reports/dbir/, last accessed: October 2021).
*** *Apache Hadoop*, https://hadoop.apache.org/, last visit on October 2021.

APLICAŢII ALE REGULILOR DE ASOCIERE ÎN PREVENIREA ATACURILOR CIBERNETICE

(Rezumat)

Proiectarea unei soluții de securitate ar trebui să se bazeze pe o bună cunoaștere a elementelor protejate și să fie axată pe răspunsuri active în detrimentul celor reactive. Susținem și dovedim că activitățile rău intenționate, cum ar fi exploatarea vulnerabilităților și atacurile de tip (D)DoS asupra aplicațiilor web, pot fi detectate încă din fazele inițiale corespunzătoare. Deși pot părea distincte, ambele scenarii de atac sunt observabile prin modele de acces anormale. În baza acestei observații, analizăm în prima etapă jurnalele de acces Web utilizând tehnici de extragere a regulilor de asociere și identificăm indiciile unor activități rău intenționate. Această nouă descriere a datelor istorice este apoi corelată cu informațiile referitoare la structura site-ului Web și modelate folosind structuri de date *trie*. Fiecare nouă solicitare primită este prelucrată prin parcurgea *trie*-urilor rezultate. Astfel identificăm dacă această nouă cerere este legitimă sau nu. Rezultatele obținute folosind această abordare proactivă demonstrează faptul că un potențial atacator nu poate obține informațiile necesare pentru orchestrarea unor atacuri de succes.