[Applied Computing and Informatics 14 (2018) 28–36](http://dx.doi.org/10.1016/j.aci.2017.04.002)



Contents lists available at [ScienceDirect](http://www.sciencedirect.com/science/journal/22108327)

Applied Computing and Informatics

journal homepage: [www.sciencedirect.com](http://www.sciencedirect.com/)

Detection of attack-targeted scans from the Apache HTTP Server access logs



Merve Ba s Seyyar [a](#_bookmark0),[⇑](#_bookmark2), Ferhat Özgür Çatak [b](#_bookmark1), Ensar Gül [a](#_bookmark0)

a *I*\_*stanbul S* *ehir University, I*\_*stanbul, Turkey*

b *Tubitak/Bilgem, Kocaeli, Turkey*

# a r t i c l e i n f o

*Article history:*

Received 8 January 2017

Revised 25 April 2017

Accepted 26 April 2017

Available online 28 April 2017

*Keywords:*

Rule-based model Log analysis

Scan detection

Web application security XSS detection

SQLI detection

# a b s t r a c t

A web application could be visited for different purposes. It is possible for a web site to be visited by a regular user as a normal (natural) visit, to be viewed by crawlers, bots, spiders, etc. for indexing purposes, lastly to be exploratory scanned by malicious users prior to an attack. An attack targeted web scan can be viewed as a phase of a potential attack and can lead to more attack detection as compared to traditional detection methods. In this work, we propose a method to detect attack-oriented scans and to distinguish them from other types of visits. In this context, we use access log files of Apache (or ISS) web servers and try to determine attack situations through examination of the past data. In addition to web scan detec- tions, we insert a rule set to detect SQL Injection and XSS attacks. Our approach has been applied on sam- ple data sets and results have been analyzed in terms of performance measures to compare our method and other commonly used detection techniques. Furthermore, various tests have been made on log sam- ples from real systems. Lastly, several suggestions about further development have been also discussed.

© 2017 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University. This is an

open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Contents

1. Introduction 29
2. Related work 29
3. System model 30
   1. Assumptions 30
   2. Data and log generation 30
      1. Web servers 30
      2. Damn Vulnerable Web Application (DVWA) 30
      3. Web vulnerability scanners 30
   3. Rules and methodology. 31
4. Results 35
   1. Experimental setup 35
   2. Model evaluation 35
   3. Scan detection on live data 35
5. Conclusion 36

Appendix A. Supplementary material 36

References 36

\* Corresponding author.

*E-mail addresses:* [merveseyyar@std.sehir.edu.tr](mailto:merveseyyar@std.sehir.edu.tr) (M. Bas Seyyar), [ozgur.catak@tubitak.gov.tr](mailto:ozgur.catak@tubitak.gov.tr) (F.Özgür Çatak), [ensargul@sehir.edu.tr](mailto:ensargul@sehir.edu.tr) (E. Gül).

Peer review under responsibility of King Saud University.

**Production and hosting by Elsevier**

<http://dx.doi.org/10.1016/j.aci.2017.04.002>

2210-8327/© 2017 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.

This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

The dependency to web systems; consisting of web services and web applications, is growing with time. From health sector to elec- tronic commerce (e-commerce), internet usage is needed in all areas of life. Due to incremental utilization of cloud technology; there is no doubt that this dependency will increase even more. However, web environment is hosting billions of users including malicious ones like script kiddies and cyber terrorists. Malicious users misuse highly efficient automated scan tools to detect vul- nerabilities in web applications. Obtaining diagnostic information about web applications and specific technologies thanks to these tools is known as ‘‘Reconnaissance” in penetration testing method- ologies and standards like The Penetration Testing Execution Stan- dard (PTES). This information gathering phase is the first phase of all attacks just before exploitation because security vulnerabilities would lead to threats either directly or indirectly [[1]](#_bookmark33). Because, an overlooked vulnerability scan may result in a large scale problems such as information leakage and privacy violation. As a matter of fact, the detection of these malicious scans becomes very crucial to prevent web applications from exploitation and to take effective countermeasures almost immediately.

According to European Network and Information Security Agency (ENISA) Threat Landscape 2015 (ETL 2015) [[2]](#_bookmark19), web based and web applications attacks are ranked as number two and three in cyber-threat environment, and their rankings have remained unchanged between 2014 and 2015. Since web security related threats have been perpetually evolving, web applications are more disposed to security risks [[3]](#_bookmark19). Also, attack methods to web applica- tions are very diverse and their trends continue for a long time. For instance, although Structured Query Language (SQL) injection and Cross-Site Scripting (XSS) seem to be at a decreasing rate in 2014, an increase in their exposures is seen in 2015. Therefore, one may easily deduce that web systems are in the focus of cyber criminals. To detect all of mentioned attacks and scans, analyzing the log files is usually preferred, because anomalies in users’ requests and related server responses could be clearly identified. Two primary reasons for this preference are that log files are easily available, and there is no need for expensive hardware for analysis [[4]](#_bookmark19). In addition, logs may provide successful detection especially for encrypted protocols such as Secure Sockets Layer (SSL) and Secure Shell Daemon (SSHD) [[5]](#_bookmark19). However, the heavier the website’s traf- fic is, the more difficult the examination of the log files gets. There- fore, the need for an user-friendly web vulnerability scan detection

tool by analyzing log files seems pretty obvious.

Therefore, the objectives of this study can be summarized as follows:

* to detect vulnerability scans.
* to detect XSS and SQLI attacks.
* to examine access log files for detections.

Accordingly, the contributions of the work can be expressed as

follows:

* The motivation of the relevant work is quite different, typically focusing on machine-learning based predictive detection of

malicious activities. Actually, all machine learning algorithms have training phase and training data to built a classification model. In order to increase accuracy of machine learning classi- fier model, a large scale input training data is needed. In turn, an increase in memory consumption would occur. As a result, either the model would turn out to be not trainable, or training phase would last for days. On the other hand, executing the pro- posed rule set on access logs does not cause any memory con- sumption problems. Our script simply runs on Ubuntu terminal with a single line of code.

* Another negative aspect of focusing on machine learning is overfitting; referring to a model that models the training data

too well. Using a very complex models may result in overfitting that may negatively influence the model’s predictive perfor- mance and generalization ability [[6]](#_bookmark19). Nevertheless, we design our rules to operate on past data which allows a detailed anal- ysis of a user’s actions [[4]](#_bookmark19) so that the complexity of our approach is not too high.

* The proposed model addresses the detection of web vulnerabil-

ity scans on web applications by analyzing log files retrieved

from web servers. Since most of the web servers log HTTP requests by default, data is easily available to be analyzed. Thus, any extra configuration, installation, purchase or data format modification are not needed. Furthermore, our analysis is based upon rule-based detection strategy and we built our rule set on several features of log entries. As opposed to relevant work, the number of these features is low enough to make input data less complex.

* Finally, our work contributes to a better understanding of cur-

rent web security vulnerabilities. For example, we can detect

web vulnerability scanners and learn about vulnerability itself at the same time.

The rest of the paper is organized as follows: The related work is presented in [Section 2](#_bookmark3). [Section 3](#_bookmark4) presents our system model in details. Our model evaluation and real system test results are pre- sented in [Section 4](#_bookmark18). The concluding remarks are given in [Section 5](#_bookmark20).

1. Related work

Within this section, the most related researches for vulnerabil- ity scan detection have been reviewed.

Auxilia and Tamilselvan suggest a negative security model for intrusion detections in web applications [[7]](#_bookmark19). This method is one of the dynamic detection techniques that is anomaly-based. The authors propose to use Web Application Firewall (WAF) with a rule set protecting web applications from unknown vulnerabilities. When analyzed their rules for Hypertext Transfer Protocol (HTTP) attacks detection, the rules appears to be generated by checking the values of some important HTTP header fields, Uniform Resource Identifier (URI) strings, cookies, etc. Associating WAF, Intrusion Detection System (IDS), rule engine reasoning together makes this article interesting.

Goseva-Popstojanova et al. [[8]](#_bookmark19) propose a method to classify malicious web sessions through web server logs. Firstly, the authors constitute four different data sets from honeypots; on which several web applications were installed. Afterwards, 43 dif- ferent features were extracted from web sessions to characterize each session and three machine learning methods that are Support Vector Machine (SVM), J48 and Partial Decision Trees (PART) were used to make the classifications. The authors assert that when all 43 features used in learning period, their method to distinguish between attack and vulnerability scan sessions attains high accu- racy rates with low probability of false alarms. This comprehensive research provides significant contribution in the area of web security.

Different from log analysis, Husák et al. [[9]](#_bookmark19) analyze extended network flow and parse HTTP requests. In addition to some Open Systems Interconnection (OSI) Layer 3 and Layer 4 data, the extracted HTTP information from network flow includes host name, path, user agent, request method, response code, referrer and content type fields. To group network flow in three classes such as repeated requests, HTTP scans, and web crawlers; source Internet Protocol (IP), destination IP, and requested Uniform Resource Locator (URL) split into domain and path are used. One

of the interesting results they obtain is that the paths requested for HTTP scans are also requested for brute-force attack as well. How- ever, not only HTTP requests but also HTTP responds should also be analyzed to get more effective results.

After a learning period of non-malicious HTTP logs, Zolotukhin et al. [[10]](#_bookmark21) analyze HTTP requests in an on-line mode to detect net- work intrusions. Normal user behavior, anomalies related features and intrusions detection are extracted from web resources, queries attributes and user agent values respectively. The authors compare five different anomaly-detection methods; that are Support Vector Data Description (SVDD), K-means, Density-Based Spatial Cluster- ing of Applications with Noise (DBSCAN), Self-Organizing Map (SOM) and Local Outlier Factor (LOF), according to their accuracy rates in detecting intrusions. It is asserted that simulations results show higher accuracy rates compared to the other data-mining techniques.

Session Anomaly Detection (SAD) is a method developed by Cho and Cha [[11]](#_bookmark22) as a Bayesian estimation technique. In this model, web sessions are extracted from web logs and are labelled as ‘‘nor- mal” or ‘‘abnormal” depending on whether it is below or above the assigned threshold value. In addition, two parameters that are page sequences and their frequency are investigated in training data. In order to test their results; the authors use Whisker v1.4 as a tool for generating anomalous web requests and it is asserted that The Bayesian estimation technique has been successful for detect- ing 91% of all anomalous requests. Therefore, two points making this article different from the others are that SAD can be cus- tomized by choosing site-dependent parameters; and the false positive rates gets lower with web topology information.

Singh et al. [[12]](#_bookmark23) have presented an analysis of two web-based attacks which are i-frame injection attacks and buffer overflow attacks. For analysis, log files created after attacks are used. They compare the size of the transferred data and the length of input parameters for normal and malicious HTTP requests. As a result, they just have carried out descriptive statistics and have not men- tioned any detection techniques.

In their work, Stevanovic et al. [[13]](#_bookmark24) use SOM and Modified Adaptive Resonance Theory 2 (Modified ART2) algorithms for training and 10 features related to web sessions for clustering. Then, the authors label these sessions as human visitors, well- behaved web crawlers, malicious crawlers and unknown visitors. In addition to classifying web sessions, similarities among the browsing styles of Google, MSN, and Yahoo are also analyzed in this article. The authors obtain lots of interesting results, one of which is that 52% of malicious web crawlers and human visitors are similar in their browsing strategies; which means that it is hard to distinguish each other.

Another completely different propose a semantic model that is named ontological model [[14]](#_bookmark25). They assert that attack signatures are not independent from programming languages and platforms. As a result, signatures may become invalid after some changes in business logic. In contrary, their model is extendible and reusable and could detect malicious scripts in HTTP requests and response. Also, thanks to ontological model, zero day attacks could be effec- tively detected. Their paper also includes a comparison between the proposed Semantic Model and ModSecurity.

There are several differences between our work and the above mentioned works. Firstly, as in the most of the related works, checking only the user-agent header field from a list is not enough to detect web crawlers in the correct way. Correspondingly, we add extra fields to check to make the web crawler detection more accu- rate. Additionally, unlike machine learning and data-mining, rule-based detection has been used in the proposed model. Finally, in contrast to other works, we prefer to use combined log format in order to make the number of features larger and to get more con- sistent results.

1. System model

In this section, we describe how we construct and design the proposed model in detail. Also, we present our rules with underly- ing reasons.

* 1. *Assumptions*
* In access logs, POST data can not get logged. Thus, the proposed method cannot capture this sort of data.
* Browsers or application servers may support other encodings. Since only two of them are in the context of this work, our script

cannot capture data encoded in other styles.

* Our model is designed for detection of two well-known web application attacks and malicious web vulnerability scans, not

for prevention. Thus, working on-line mod is not included in the context of our research.

* 1. *Data and log generation*

In this section, tools, applications, virtual environment used throughout this work and their installation and configuration set- tings are explained.

* + 1. *Web servers*
       1. *HTTP Server.* As mentioned earlier, Apache/2.4.7 (Ubuntu) Server is chosen as a web server. Apache is known to be the most commonly used web server. According to the W3Techs (Web Tech- nology Surveys) [[15]](#_bookmark26), as of December 1, 2016; Apache is used by

51.2 percent of all web servers. In addition, it is open source, highly scalable and has a dynamically loadable module system. Apache installation is made via apt-get command-line package manager. Any extra configuration is not necessary for the scope of this work.

* + - 1. *Apache Tomcat.* The Apache Tomcat being an implementa- tion of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies, is an open source software [[16]](#_bookmark27). In this work, Apache Tomcat Version 8.0.33 is used. Atlassian JIRA Standalone Edition (Jira 3.19.0-25-generic #26) is used as a web application. Access log configuration of Tomcat is set to be similar to access log entries in Apache.
    1. *Damn Vulnerable Web Application (DVWA)*

DVWA is a vulnerable PHP/MySQL web application. It is designed to help web developers find out critical web application security vulnerabilities by hands on activity. Different from illegal website-hacking, it offers a totally legal environment to exploit for security people. Thanks to DVWA; Brute Force, Cross Site Request Forgery (CSRF), Command Execution, XSS (reflected) and SQL Injec- tion vulnerabilities could be tested for three security levels; low, medium, high.

In this work, DVWA 1.0.8 version (Release date: 11/01/2011) is used. To install this web application, Linux Apache MySQL PHP (LAMP) Server; including MySql, PHP5, and phpMyAdmin, has been installed. The reasons for studying with DVWA are to better understand XSS and SQL Injection attacks and to find out related payloads substituted in query string part of URIs. In this way, rule selection to detect these attacks from access logs could be correctly determined. Also, web vulnerability scanners used in this work, have scanned this web application for data collection purposes.

* + 1. *Web vulnerability scanners*
       1. *Acunetix.* Acunetix is one of the most commonly used com- mercial web vulnerability scanners. Acunetix scans a web site according to the determined configurations, produces a report about the existing vulnerabilities, groups them as high, medium,

low and informational; and identifies the threat level of the web application with the related mitigation recommendations. In the context of this work, Acunetix Web Vulnerability Scanner (WVS) Reporter v7.0 has been used with default scanning configurations in addition to site login information.

* + - 1. *Netsparker.* Netsparker is a web application security scan- ner that is commercial too. Netsparker detects security vulnerabil- ities of a web application and produces a report including mitigation solutions. In addition, detected vulnerabilities could be exploited to confirm the report results. In the context of this work, Netsparker Microsoft Software Library (MSL) Internal Build

4.6.1.0 along with Vulnerability Database 2016.10.27.1533 has been used with special scanning configurations including custom cookie information.

* + - 1. *Web Application Attack and Audit Framework (W3AF).* W3AF is an open source web application security scanner. W3AF is devel-

oped using Python and licensed under General Public License (GPL) v2.0. Framework is designed to help web administrators secure the web applications. W3AF could detect more than 200 vulnerabilities [[17]](#_bookmark28). W3AF has several plug-ins for different operations such as crawling, brute forcing, and firewall bypassing. W3AF comes by default in Kali Linux and could be found in ‘‘Applications/Web Application Analysis/Web Vulnerability Scanners”. W3AF version

1.6.54 has been used with ‘‘fast-scan” profile through audit, crawl, grep and output plugins.

* 1. *Rules and methodology*

As mentioned earlier, our script runs on access log files. The main reason for this choice is the opportunity for detailed analysis about users actions. By examining past data, information security policies for the web applications could be correctly created and implemented. Additionally, further exploitations could be pre- vented in advance. Unlike the proposed model, Network Intrusion

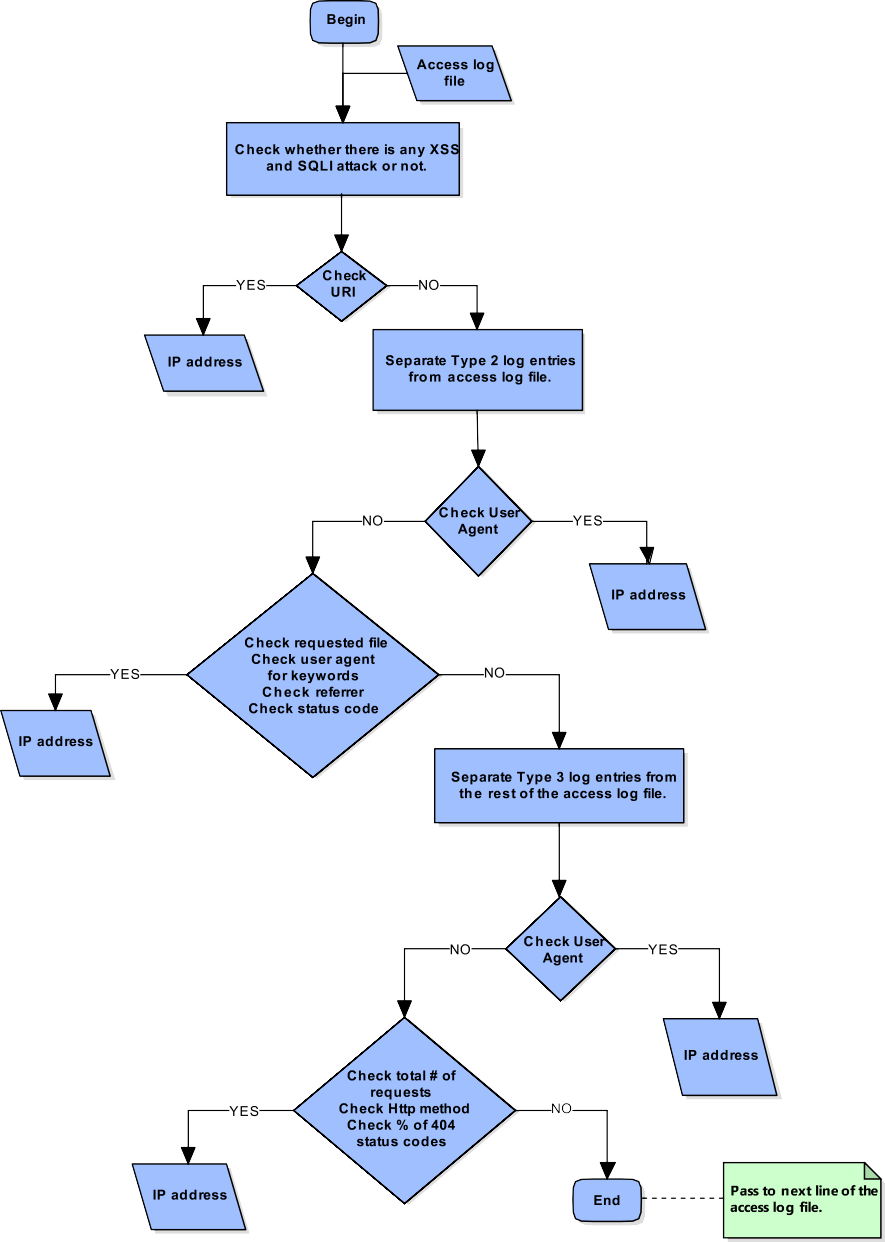


Fig. 1. Flow chart of the proposed rule-based model.

Detection System (NIDS) may not detect attacks when HTTPS is used [[4]](#_bookmark19). However, working with logs has some disadvantages. Since log files do not contain all data of HTTP request and response, some important data could not be analyzed. For example, POST parameters that are vulnerable to injections attacks could not be logged by web servers. Another negative aspects are the size of logs and parsing difficulty. Nevertheless, to solve this problem, we sep- arate the access log files on a daily basis. Therefore, web adminis-

Table 1

HTTP methods in Acunetix.

HTTP method Number

Connect 2

Get 2758

Options 2

Post 668

Trace 2

Track 2

Total 3434

Table 2

HTTP methods in Netsparker.

trators might run our script every day to check for an attack. Lastly, real-time detection and prevention is not possible with the pro- posed method which runs off-line. Thus, we could not guarantee to run on-line. In fact, this approach is conceptually sufficient for the scope of this work. Differently from the test environment; an extra module that directly accesses logs, or a script that analyses logs faster could be developed to use our approach in a live or real environment.

Our method could be described as rule-based detection. Unlike anomaly based detection, our rules are static including both black- list and whitelist approaches. In detail, XSS and SQL injection detection part of our method is a positive security model; on the other hand, the rest is a negative security model. Thus, data eva- sion is tried to be kept at a minimum level. In order to classify IP addresses in the access log file, we identify three different visitor types as follows:

Table 6

Details of classified data sets.

Visitor type Log file Line number IP number Type 1 Normal 62,539 15

Type 2 Web robot 28,804 143

Type 3 Acunetix 6539 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| HTTP method | Number | Type 3 | Netsparker | 7314 | 1 |
| Get | 3059 | Type 3 | W3AF | 3996 | 2 |
| Head | 590 | Type 1, 2 and 3 | Total | 109,192 | 162 |
| Netsparker | 1 |  | | | |
| Options | 14 |  | | | |
| Post | 956 | Table 7 | | | |
| Propfind | 14 | Confusion matrix. | | | |

Total 4634

Table 3

HTTP status codes in Netsparker.

HTTP status code Number

200 177

301 1

302 23

404 494

500 6

Total 701

Actual: Type 3 Actual: Type 1 or 2

Predicted: Type 3 TP = 3 FN = 1

Predicted: Type 1 or 2 FP = 0 TN = 158

Table 8

Summary of results for general data set.

IP number Accuracy Precision Recall F1

162 99.38% 100.00% 75.00% 85.71%

1200

Table 4

HTTP status codes in W3AF.

1000

800

**Running Time (in seconds)**

600

400

200

0

|  |  |
| --- | --- |
| HTTP status code | Number |
| 200 | 91 |
| 302 | 8 |
| 404 | 30 |
| 500 | 6 |
| Total | 135 |

Table 5

HTTP status codes in Acunetix.

HTTP status code Number

**Log Lines**

sed method.

|  |  |  |
| --- | --- | --- |
| 200 | 598 | Fig. 2. Time performance of the propo |
| 301 | 38 |  |
| 302 | 686 | Table 9 |
| 400 | 44 | Details of log samples. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 403  404 | 16  2022 | Log file | Log duration | File size | Line number | IP number |
| 405 | 4 | Data Set 1 | 5 days | 43 MB | 202,145 | 3910 |
| 406 | 2 | Data Set 2 | 210 days | 13.4 MB | 34,487 | 9269 |
| 417 | 2 | Data Set 3 | 270 days | 7.2 MB | 36,310 | 4719 |
| 500 | 20 | Data Set 4 | 90 days | 1.3 MB | 5936 | 1795 |
| 501 | 2 | Data Set 5 | 90 days | 0.48 MB | 3554 | 579 |
| Total | 3434 | Total | 665 days | 65.37 MB | 282,432 | 20,272 |

Table 10

Data sets test results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data set | Period | IP number | Type 3 IP number | Type 3 percentage (%) |
| Data Set 1 2004 | 10/March | 370 | 13 | 3.51 |
|  | 11/March | 786 | 20 | 2.54 |
|  | 12/March | 1002 | 22 | 2.20 |
|  | 13/March | 1960 | 39 | 1.99 |
|  | 14/March | 1079 | 21 | 1.95 |
| Data Set 2 2004 | April | 3140 | 1 | 0.03 |
|  | May | 4546 | 3 | 0.07 |
|  | June | 701 | 6 | 0.86 |
|  | July | 735 | 4 | 0.54 |
|  | August | 189 | 1 | 0.53 |
|  | September | 280 | 0 | 0.00 |
|  | October | 106 | 1 | 0.94 |
| Data Set 3 2005 | June | 663 | 1 | 0.15 |
|  | July | 755 | 1 | 0.13 |
|  | August | 577 | 0 | 0.00 |
|  | September | 731 | 1 | 0.14 |
|  | October | 452 | 0 | 0.00 |
|  | November | 623 | 19 | 3.05 |
|  | December | 181 | 1 | 0.55 |
|  | January | 652 | 45 | 6.90 |
|  | February | 802 | 34 | 4.24 |
| Data Set 4 2005 | 1–15/June | 160 | 1 | 0.63 |
|  | 16–30/June | 497 | 0 | 0.00 |
|  | 1–15/July | 503 | 0 | 0.00 |
|  | 16–30/July | 280 | 1 | 0.36 |
|  | 1–15/August | 284 | 0 | 0.00 |
|  | 16–30/August | 282 | 0 | 0.00 |
| Data Set 5 2005 | 16–31/January | 28 | 0 | 0.00 |
|  | 1–15/February | 176 | 0 | 0.00 |
|  | 16–28/February | 112 | 0 | 0.00 |
|  | 1–15/March | 225 | 3 | 1.33 |
|  | 16–30/March | 28 | 0 | 0.00 |

* Type 1: Regular (normal) users with a normal (natural) visit.
* Type 2: Crawlers, bots, spiders or robots.
* Type 3: Malicious users using automated web vulnerability scanners.

As shown in [Fig. 1](#_bookmark6) in Phase 1, our first step is to detect SQL injection and XSS attacks. Although different places of HTTP (the HTTP body, URI) could be used to exploit a vulnerability [[4]](#_bookmark19); we will analyze path and query parts of the requested URI for detection.

In detail; for XSS, we use regular expressions to recognize some patterns such as HTML tags, ‘src’ parameter of the ‘img’ tag and some Javascript event handlers. Likewise; for SQL injection, we check the existence of the singlequote, the doubledash, ‘#’, exec() function and some SQL keywords. In addition, since there is a pos-

sibility for URL obfuscation, Hex and UTF-8 encodings of these pat- terns are also taken in consideration.

Afterwards, we continue by separating IP addresses of Type 2 from the rest of the access log file in Phase 2. To do this, two differ- ent approaches are used. Firstly, user-agent part of all log entries is compared with the user-agent list from robots database that is publicly available in [[18]](#_bookmark29). However, since this list may not be up- to-date, another bot detection rules are added. In order to identify these rules, we use the following observations about web robots:

1. Most of the web robots make a request for ‘‘/robots.txt” file [[19]](#_bookmark30).
2. Web robots have higher rate of ‘‘4xx” requests since they usu- ally request unavailable pages [[20–23]](#_bookmark31).
3. Web robots have higher unassigned referrer (‘‘–”) rates [[23–25]](#_bookmark32).

4

3.5

3

2.5

Type 3 (%)

2

1.5

1

0.5

0

Days

1

0.9

0.8

0.7

0.6

Type 3 (%)

0.5

0.4

0.3

0.2

0.1

0

Apr May Jun Jul Aug Sep Oct

Months

Fig. 3. Data Set 1 test results. Fig. 4. Data Set 2 test results.

1. According to the access logs that we analyzed, user-agent header field of web robots may contain some keywords such as bot, crawler, spider, wanderer, and robot.

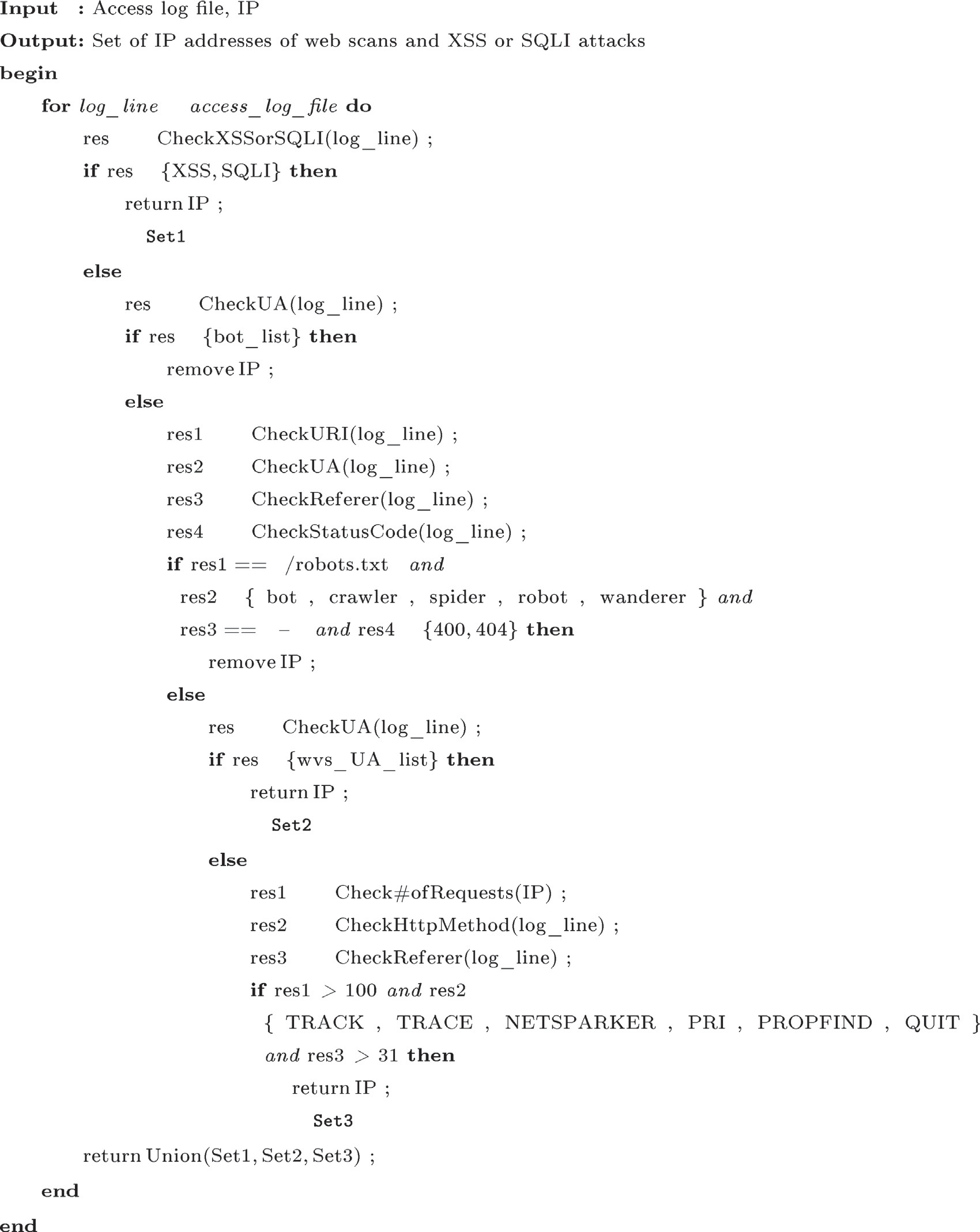
As a result of above mentioned observations, we add some extra rules to correctly distinguish Type 2 from other visitors.

For the rest our rule set as indicated at Phase 3 in [Fig. 1](#_bookmark6), we con- tinue by investigating our access log files formed as a result of vul- nerability scanning mentioned in the previous section. As shown in [Tables 1 and 2](#_bookmark7), our first immediate observation is that as compared to Type 2 and Type 1, Type 3’s requests include different HTTP methods; such as Track, Trace, Netsparker, Pri, Propfind and Quit. Secondly, as shown in [Table 3](#_bookmark8), [Tables 4 and 5](#_bookmark12); we deduct that

status codes of Type 3 differ from Type 2 and Type 1. In fact, Type 3 has higher rate of ‘‘404” requests, average of which for Acunetix, Netsparker and W3AF is 31% in our data set. Thus, we generate a rule to check the presence of these HTTP methods and the percent- age of ‘‘404” requests. User-agent header fields of Type 3 could generally be modified and obfuscated manually at the configura- tion phase before vulnerability scan. Even so, we made a list of well-known automated web vulnerability scanners, and compare it with user-agent header fields. Finally, we notice that these scan- ners make at least more than 100 HTTP requests in a certain time, we select this value as a threshold for Type 3 detection.

The pseudo code of the proposed model is shown in [Algorithm 1](#_bookmark17):

Algorithm 1. Pseudo-Code for Proposed Model.



*∈*

*←−*

*∈*

*D*

*←−*

*∈*

*←−*

*←−*

*←−*

*←−*

”

*∈* ” ” ”

” ”

”

” ”

*∈*

” ” ” ”

”

*←−*

*∈*

*D*

*←−*

*←−*

*←−*

”

” ”

*∈*

” ”

” ” ” ”

” ”

”

*D*

1. Results

This section is based on the evaluation of our model against some important metrics. Moreover, test results of attack detection on live data are also included.

* 1. *Experimental setup*

To implement our rules, Python programming language version

3.5 has been chosen. Script is executed on Ubuntu operating sys- tem mentioned in Section [3.2.2](#_bookmark5) via terminal. To parse log lines, ‘‘apache-log-parser 1.7.0” which is a Python package has been used. As well as, we benefit from python libraries that are collec- tions, datetime, numpy, ua-parser and argparse.

Since there are not any actual, publicly available and labelled data sets to evaluate our model, we create our data sets. In fact, we deploy two different web applications on two different web servers to form Type 1 and Type 3 traffics. Details are expressed in Section [3.2.2](#_bookmark5).

Type 1 (normal traffic) is the data set collected from Jira Soft- ware as a web application running on Tomcat web server during 4 days.The size of the related access log file is 16.3 MB. As shown [Table 6](#_bookmark9), log file contains 62,539 log entries from 15 different IP addresses. These requests are generated in a local network.

For Type 2 traffic, an external traffic that is open to the internet is needed. To this end, we make use of three different access log files retrieved from a company website. In detail, log files contain crawling data collected during 13 days from requests of several web robots. The size of the related access log files is totally

6.4 MB, and log files contain 28,804 log entries from 143 different IP addresses as shown [Table 6](#_bookmark9).

To generate Type 3 traffic, DVWA running on Apache HTTP Ser- ver is used as a web application. Before scanning, the security level of DVWA is configured as low security. Moreover, we scan this application via Acunetix, Netsparker and W3AF as web vulnerabil- ity scanners. Firstly, DVWA is scanned for 22 min and 22 s with Acunetix. Secondly, DVWA is scanned for 19 min and 56 s with Netsparker. Lastly, DVWA is scanned for 2 min and 6 s with W3AF. The details of the related access log files are summarized as Type 3 in [Table 6](#_bookmark9).

For the evaluation of the proposed model, we combine all men- tioned access log files into one file that is our general data set. Then, we run our Python script on the mentioned data set.

* 1. *Model evaluation*

Initially, to evaluate the proposed model, we compute the con- fusion matrix where TP, FN, FP, and TN denote true negatives, false negatives, false positives, and true negatives respectively as shown in [Table 7](#_bookmark10).

After, we evaluate the following measures:

accuracy(acc (TN + TP)

## (TN + FN + FP + TP)

)=

(TP)

recall. As a result, our model has 99.38% accuracy, 100.00% preci- sion, 75.00% recall and finally 85.71% F1 score as we can see in [Table 8](#_bookmark11).

[Fig. 2](#_bookmark13) illustrates the relation between the line number of the log files and the running time. It is clear that the running time rises steadily as the number of the lines increases.

* 1. *Scan detection on live data*

We have built or model according to the data sets mentioned in Section [4.1](#_bookmark18). Additionally, we test our model according to several large-scale, live, not labelled and publicly available data sets. In this section, we share our test results illustrated in tables and graphs.

In accordance with this purpose, we have used log samples from real systems [[26]](#_bookmark34). As stated in the related web source, these samples are collected from various systems, security devices, applications, etc.; and neither Chuvakin nor we did not sanitize, anonymized or modified them in any way. Since they include HTTP access logs, we have chosen the log samples named Bundle 9, Bun- dle 7, Bundle 1, Bundle 4 and Bundle 3. For the rest of the work, these bundles are expressed as Data Set 1, Data Set 2, Data Set 3, Data Set 4 and Data Set 5 respectively. Details of these data sets are shown in [Table 9](#_bookmark14).

In order to test the log samples, Data Set 1, Data Set 2, Data Set 3, Data Set 4 and Data Set 5 are divided into daily, monthly,

7

6

5

4

Type 3 (%)

3

2

1

0

Jun Jul Aug Sep Oct Nov Dec Jan Feb

Months

Fig. 5. Data Set 3 test results.

0.7

0.6

0.5

Type 3 (%)

precision(prec)=

## (TP + FP)

0.4

)=

recall(rec (TP)

## (TP + FN)

F1 score = (2TP) (2TP + FP + FN)

(1)

0.3

0.2

More specifically, the accuracy provides the percentage of Type 3 that are detected correctly. The precision determines the fraction of IP addresses correctly classified as Type 3 over all IP addresses classified as Type 3. The recall (a.k.a. sensitivity) is the fraction of IP addresses correctly classified as Type 3 over all IP addresses of Type 3. Finally, the F1-score is a harmonic mean of precision and

0.1

0

1 2 3 4 5 6

15-Day Period

Fig. 6. Data Set 4 test results.

1.4

1.2

1

0.8

Type 3 (%)

0.6

0.4

0.2

0

1 2 3 4 5

15-Day Period

Fig. 7. Data Set 5 test results.

Intrusion Kill Chains, vol. 1, API, 2011, URL [<https://books.google.com.](https://books.google.com.tr/books?id=oukNfumrXpcC) [tr/books?id=oukNfumrXpcC>](https://books.google.com.tr/books?id=oukNfumrXpcC).

1. European Union Agency for Network and Information Security (ENISA), ENISA Threat Landscape 2015. URL <[https://www.enisa.europa.eu/publications/](https://www.enisa.europa.eu/publications/etl2015) [etl2015](https://www.enisa.europa.eu/publications/etl2015)>, 2016 (accessed November 29, 2016).
2. D.V. Bernardo, Clear and present danger: interventive and retaliatory approaches to cyber threats, Appl. Comput. Infor. 11 (2) (2015) 144–157, <http://dx.doi.org/10.1016/j.aci.2014.11.002>, URL [<http://](http://www.sciencedirect.com/science/article/pii/S2210832714000386) [www.sciencedirect.com/science/article/pii/S2210832714000386>](http://www.sciencedirect.com/science/article/pii/S2210832714000386).
3. R. Meyer, Detecting Attacks on Web Applications from Log Files. URL <[https://](https://www.sans.org/reading-room/whitepapers/logging/detecting-attacks-web-applications-log-files-2074) [www.sans.org/reading-room/whitepapers/logging/detecting-attacks-web-](https://www.sans.org/reading-room/whitepapers/logging/detecting-attacks-web-applications-log-files-2074) [applications-log-files-2074](https://www.sans.org/reading-room/whitepapers/logging/detecting-attacks-web-applications-log-files-2074)>, 2008 (accessed December 12, 2016).
4. D.B. Cid, Log Analysis using OSSEC. URL <[http://www.academia.edu/8343225/](http://www.academia.edu/8343225/Log_Analysis_using_OSSEC) [Log\_Analysis\_using\_OSSEC](http://www.academia.edu/8343225/Log_Analysis_using_OSSEC)>, 2007 (accessed November 29, 2016).
5. Wikipedia, Overfitting. URL <<https://en.wikipedia.org/wiki/Overfitting>>, 2016

(accessed December 27, 2016).

1. M. Auxilia, D. Tamilselvan, Anomaly detection using negative security model in web application, in: 2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM), 2010, pp. 481–486, <http://dx.doi.org/10.1109/CISIM.2010.5643461>.
2. K. Goseva-Popstojanova, G. Anastasovski, R. Pantev, Classification of malicious web sessions, in: 2012 21st International Conference on Computer Communications and Networks (ICCCN), 2012, pp. 1–9, [http://dx.doi.org/](http://dx.doi.org/10.1109/ICCCN.2012.6289291) [10.1109/ICCCN.2012.6289291](http://dx.doi.org/10.1109/ICCCN.2012.6289291).

monthly, 15-day and 15-day periods respectively. Related details

are expressed in [Table 10](#_bookmark15).

Type 3 percentage of each data set is shown in [Figs. 3–7](#_bookmark16).

1. Conclusion

In this work, we studied web vulnerability scans detection through access log files of web servers in addition to detection of XSS and SQLI attacks. In accordance with this purpose, we used rule-based methodology. Firstly, we examined the behavior of the automated vulnerability scanners. Moreover, we implemented our model with a Python script. Afterwards, our model has been evaluated based on data we have collected. Finally, we tested our model on the log samples from real systems.

It is clear that our method has very high probability of detection and low probability of false alarm. More specifically, the accuracy and the precision rates of our model are 99.38%, 100.00% respec- tively. More importantly, malicious scans can be captured more precisely because different types of scanning tools including both open source and commercial tools were examined. Therefore, our results indicates that static rules can detect successfully web vul- nerability scans. Besides, we have observed that our model func- tions properly with larger and live data sets and correctly detects Type 3 IP addresses.

As shown in the [Fig. 2](#_bookmark13), the relation between the number of lines of the log files and the running time is linear. As a result, how long a log file would be analyzed, could be predicted in advance.

The results presented in this work may enhance researches about malicious web scans and may support the development of attack detection studies. Also, if security analysts or administrators execute the proposed python script several times within the same day, he/she could prevent most of the web related attacks.

Future work considerations related to this work are twofold. In the first place, one could make our model possible to analyze other log files such as audit log and error log. Secondly, in addition to the scope of this work; different from SQLI and XSS attacks, other well- known web application attacks like CSRF could be addressed too.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at <http://dx.doi.org/10.1016/j.aci.2017.04.002>.

References

[1] E.M. Hutchins, M.J. Cloppert, R.M. Amin, Intelligence-driven Computer Network Defense Informed by Analysis of Adversary Campaigns and

1. M. Husák, P. Velan, J. Vykopal, Security monitoring of http traffic using extended flows, in: 2015 10th International Conference on Availability, Reliability and Security, 2015, pp. 258–265, [http://dx.doi.org/10.1109/](http://dx.doi.org/10.1109/ARES.2015.42) [ARES.2015.42](http://dx.doi.org/10.1109/ARES.2015.42).
2. M. Zolotukhin, T. Hämäläinen, T. Kokkonen, J. Siltanen, Analysis of http requests for anomaly detection of web attacks, in: 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, 2014, pp. 406–411, <http://dx.doi.org/10.1109/DASC.2014.79>.
3. S. Cho, S. Cha, Sad: web session anomaly detection based on parameter estimation, Comput. Secur. 23 (4) (2004) 312–319, [http://dx.doi.org/10.1016/](http://dx.doi.org/10.1016/j.cose.2004.01.006) [j.cose.2004.01.006](http://dx.doi.org/10.1016/j.cose.2004.01.006), URL [<http://www.sciencedirect.com/science/article/pii/](http://www.sciencedirect.com/science/article/pii/S0167404804000264) [S0167404804000264>](http://www.sciencedirect.com/science/article/pii/S0167404804000264).
4. N. Singh, A. Jain, R.S. Raw, R. Raman, Detection of Web-Based Attacks by Analyzing Web Server Log Files, Springer India, New Delhi, 2014, [http://dx.doi.](http://dx.doi.org/10.1007/978-81-322-1665-0_10) [org/10.1007/978-81-322-1665-0\_10](http://dx.doi.org/10.1007/978-81-322-1665-0_10), pp. 101–109.
5. D. Stevanovic, N. Vlajic, A. An, Detection of malicious and non-malicious website visitors using unsupervised neural network learning, Appl. Soft Comput. 13 (1) (2013) 698–708, [http://dx.doi.org/10.1016/j.](http://dx.doi.org/10.1016/j.asoc.2012.08.028) [asoc.2012.08.028](http://dx.doi.org/10.1016/j.asoc.2012.08.028), URL [<http://www.sciencedirect.com/science/article/pii/](http://www.sciencedirect.com/science/article/pii/S1568494612003778) [S1568494612003778>](http://www.sciencedirect.com/science/article/pii/S1568494612003778).
6. A. Razzaq, Z. Anwar, H.F. Ahmad, K. Latif, F. Munir, Ontology for attack detection: an intelligent approach to web application security, Comput. Secur. 45 (2014) 124–146, <http://dx.doi.org/10.1016/j.cose.2014.05.005>, URL [<http://](http://www.sciencedirect.com/science/article/pii/S0167404814000868) [www.sciencedirect.com/science/article/pii/S0167404814000868>](http://www.sciencedirect.com/science/article/pii/S0167404814000868).
7. W3Techs (Q-Success DI Gelbmann GmbH), Usage Statistics and Market Share of Apache for Websites. URL <[https://w3techs.com/technologies/details/ws-](https://w3techs.com/technologies/details/ws-apache/all/all) [apache/all/all](https://w3techs.com/technologies/details/ws-apache/all/all)>, 2009–2017 (accessed December 12, 2016).
8. The Apache Software Foundation, Apache Tomcat. URL <[http://tomcat.apache.](http://tomcat.apache.org/) [org](http://tomcat.apache.org/)> (accessed December 24, 2016).
9. w3af.org, w3af. URL <[http://w3af.org](http://w3af.org/)>, 2013 (accessed December 12, 2016).
10. The Web Robots Pages, Robots Database. URL <[http://www.robotstxt.org/db.](http://www.robotstxt.org/db.html) [html](http://www.robotstxt.org/db.html)> (accessed September 4, 2016).
11. M.C. Calzarossa, L. Massari, D. Tessera, An extensive study of web robots traffic, in: Proceedings of International Conference on Information Integration and Web-based Applications &#38; Services, IIWAS ’13, ACM, New York, NY, USA, 2013, pp. 410:410–410:417, [http://dx.doi.org/10.1145/](http://dx.doi.org/10.1145/2539150.2539161) [2539150.2539161](http://dx.doi.org/10.1145/2539150.2539161).
12. M.D. Dikaiakos, A. Stassopoulou, L. Papageorgiou, An investigation of web crawler behavior: characterization and metrics, Comput. Commun. 28 (8) (2005) 880–897, <http://dx.doi.org/10.1016/j.comcom.2005.01.003>, URL

[<http://www.sciencedirect.com/science/article/pii/S0140366405000071>](http://www.sciencedirect.com/science/article/pii/S0140366405000071).

1. M. Dikaiakos, A. Stassopoulou, L. Papageorgiou, Characterizing Crawler Behavior from Web Server Access Logs, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, <http://dx.doi.org/10.1007/978-3-540-45229-4_36>.
2. M.C. Calzarossa, L. Massari, Analysis of Web Logs: Challenges and Findings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, [http://dx.doi.org/](http://dx.doi.org/10.1007/978-3-642-25575-5_19) [10.1007/978-3-642-25575-5\_19](http://dx.doi.org/10.1007/978-3-642-25575-5_19), pp. 227–239.
3. D. Stevanovic, A. An, N. Vlajic, Feature evaluation for web crawler detection with data mining techniques, Expert Syst. Appl. 39 (10) (2012) 8707–8717, <http://dx.doi.org/10.1016/j.eswa.2012.01.210>, URL [<http://](http://www.sciencedirect.com/science/article/pii/S0957417412002382) [www.sciencedirect.com/science/article/pii/S0957417412002382>](http://www.sciencedirect.com/science/article/pii/S0957417412002382).
4. A.G. Lourenço, O.O. Belo, Catching web crawlers in the act, in: Proceedings of the 6th International Conference on Web Engineering, ICWE ’06, ACM, New York, NY, USA, 2006, pp. 265–272, <http://dx.doi.org/10.1145/1145581.1145634>.
5. D. Stevanovic, A. An, N. Vlajic, Detecting Web Crawlers from Web Server Access Logs with Data Mining Classifiers, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, <http://dx.doi.org/10.1007/978-3-642-21916-0_52>, pp. 483–489.
6. A. Chuvakin, Public Security Log Sharing Site. URL <[http://log-](http://log-sharing.dreamhosters.com/) [sharing.dreamhosters.com](http://log-sharing.dreamhosters.com/)>, 2009 (accessed December 15, 2015).