World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:12, No:12, 2018

Searching for Forensic Evidence in a Compromised Virtual Web Server against SQL Injection Attacks and PHP Web Shell

Gigih Supriyatno

***Abstract*—**SQL injection is one of the most common types of attacks and has a very critical impact on web servers. In the worst case, an attacker can perform post-exploitation after a successful SQL injection attack. In the case of forensics web servers, web server analysis is closely related to log file analysis. But sometimes large file sizes and different log types make it difficult for investigators to look for traces of attackers on the server. The purpose of this paper is to help investigator take appropriate steps to investigate when the web server gets attacked. We use attack scenarios using SQL injection attacks including PHP backdoor injection as post-exploitation. We perform post-mortem analysis of web server logs based on Hypertext Transfer Protocol (HTTP) POST and HTTP GET method approaches that are characteristic of SQL injection attacks. In addition, we also propose structured analysis method between the web server application log file, database application, and other additional logs that exist on the webserver. This method makes the investigator more structured to analyze the log file so as to produce evidence of attack with acceptable time. There is also the possibility that other attack techniques can be detected with this method. On the other side, it can help web administrators to prepare their systems for the forensic readiness.

***Keywords*—**Web forensic, SQL injection, web shell, investigation.

I. Introduction

WEB servers and web applications are the most frequently targeted by attackers in cybercrime cases. They are easier to access and do not need any special connection or state-sponsored resources. In this regard, they are easier to hack than operating systems or network devices like routers and switches. In many cases, once a website is compromised, they serve as a beachhead for other major attacks and allow attackers to gain more sensitive information from internal resources. The attack vector is the starting point that attacker can exploit a system and it come in a variety of forms. OWASP (Open Web Application Security Project) each year offers to list the most critical web application vulnerabilities. The list consists of the latest vulnerabilities, threats, attack vectors, and detection recommendations.

Report from Positive-Technologies in 2017, SQL Injection (sqli) is one of the most common attacks in web applications [1]. This attack method is used to access sensitive information on a system or run operating system commands for further penetration of a system. In the worst case, this attack can bypass the authentication and authorization mechanisms and even pass through a firewall that protects the web system. A web application becomes vulnerable to sqli attack when they to handle user inputs in a secure manner such as in situations of poor input validation [2]-[4]. In addition, the use of plugin web applications that are not provable secure or have zero-day vulnerability cause the web server to be vulnerable to sqli [5].

After a successful attack, the attacker will usually manage a broken system to perform advanced attacks. Web shell is malicious code that forms a Remote Access Tool (RAT) or backdoor that provides a means for an attacker to interact with a web server. This will save the attacker time and effort each time access to the compromised server is required. US-CERT categorizes the web shell as a very dangerous program because it is often used in cybercrime and can be a hole for injecting Advanced Persistent Threats (APTs) [6]. Web shell could hide as a single file among the thousands of files present on a web server or as a single line of code in a legitimate page on a site. Web shell can be created in several programming languages such as PHP, ASP, Java, and JavaScript depends on the server environment. The most common web shell is written in PHP since the majority of web systems support this language.

Web forensic is a branch of the computer forensic that deals with web attack [7]. Computer forensics prepares legal evidence related to cybercrime so that it can be used in a court of law. In that case, investigators are trying to reveal digital evidence in the victim server and attacker machine, to find out when it happened and how the attacker attacked the web system. Web forensics is generally carried out on both client side and server side. While the server side forensic evidence helps an investigator progress towards a conclusion, the client side evidence provides potentially very strong and detailed evidence.

Server side forensic process usually starts with an analysis of various log files in the victim machine [7]-[9]. A log file records events and actions that take place during the runtime of a service or application. Even when someone performs an attack, the log file lets store the traces left behind. It can help forensic investigators unfold the chain of events that may have led to a malicious activity. Investigation to find out incident of web application attacks can be started in one of the following files: web server(s) and application server(s) logs, application server(s) configuration files, web server(s) and any third party installed software logs, the operating system logs, and server side scripts which are used by the web application [9]. In many

Gigih Supriyatno is with the School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia (e-mail: gigih.supriyatno@gmail.com).

cases, log file contains a large number of log entries. It will be difficult and takes time to extract relevant evidence, analyze, track and understand existing information efficiently and reliably [9], [8].

This paper discusses the steps to find digital evidence using log files against SQL injection attack and detect backdoor web shells on compromised servers. It will help forensic investigators to retrieve important information from the server through simple operating procedures, and to offer investigation of collections and reports of analysis related to digital evidence. We use the attack simulation using SQL injection vulnerability and then perform injection backdoor into server.

II. Literature Review

In order to conduct web forensic, it is necessary to use special methods for forensic purposes to be achieved. Following a standard methodology is crucial to perform a successful forensic investigation. The European Union Agency for Network and Information Security (ENISA) makes general forensic process models that can be applied to a web server including collection phases, data testing, data analysis and reporting [10]. Suteva et al. perform forensic web servers using three-stage forensic processes including data acquisition, analysis, and reporting [7]. They simulate frequent attacks on websites such as SQL injection, XSS, Remote File Inclusion, and command-line injection. Their forensic process is done manually against multiple log files using known keywords to detect attacks.

Common methods of forensic web server process carried out include the following stages [9], [11] :

1. Protect the web application during the forensic examination to prevent any modification of the evidence files. Protection can be done for multiple servers depending on the condition of the object forensics.
2. Discover all files needed for the forensic investigation.
3. Perform forensic analysis of the files considering the order of events and the degree of compromise, and then create chain of events.
4. Create a report of the results of the investigation of the data files obtained from web applications.
5. Prepare recommendations for post-event actions.

Previous researches are common case when attacks occur on the web server. We found that there is still an open space for the development and research on the technical forensic web server in special cases such as SQL attack and backdoor injection.

III. Attack Scenario

Focus of this paper is to detect SQL injection attacks and successful backdoor uploads on web server. We use attack scenarios against web servers. Each step is viewed at the application layer level. The importance of understanding the attack technique is by understanding how an attacker performs an action will find it easier to find digital evidence left behind. This simulation ignores the stage of scanning and gathering information that is commonly done when performing penetration tests, although at that stage there will be many traces left by attackers in the server log.

The web server used in this simulation uses Ubuntu 12.04 32-bit, Apache 2.2.22, MySQL 5.5.35, PHP 5.3.10-1, and bWAPP web applications. The web server and the attacker's computer are on a LAN network. The web server is operated in Virtual Machine (VM) using VMware Fusion. The dummy domain used in simulation as a website is www.presiden.info with IP address 192.168.179.142. Simulated attacks were carried out through two stages i.e. SQL Injection and backdoor upload. The detailed working of the simulated attacks is described as a flow chart shown in Fig. 1.

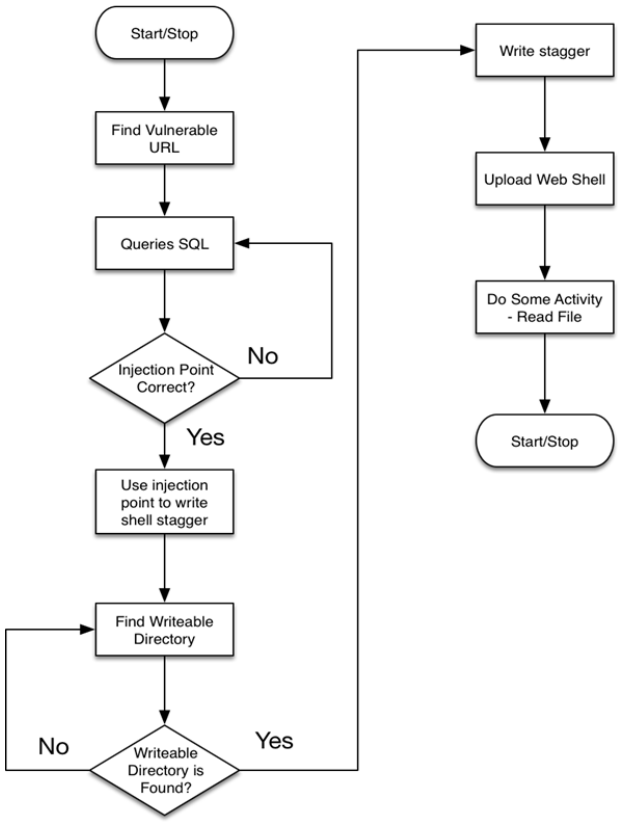

Fig. 1 Flowchart attack scenario

A. SQL Injection

SQL injection is carried out using the tools sqlmap to create malicious queries of SQL. The attacker is assumed to have knowledge of the target server URL that is vulnerable to SQL injection. HTTP POST and HTTP GET methods used in these simulations to see the characteristics of the traces left behind on the server.

B. Upload Backdoor

The attacker sends shell stager file to the web server before uploading the main shell's web. The shell stager searches for the writeable permissions on the web server. The location of the stager shell should be accessible by the attacker's browser. Shell stager is written through SQL query and encoded according to successful injection query in web server execution.

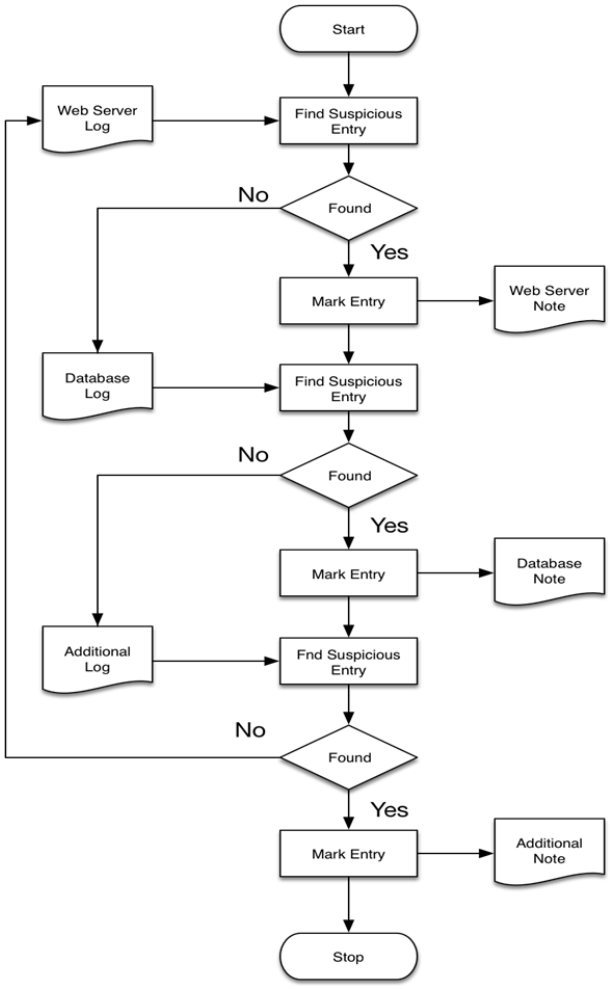

Fig. 2 Log analysis flowchart

IV. Method for Searching Evidence

In this section, we describe how we investigate evidence from a suspected web server. Our method is highly dependent on the information recorded on the log file. Log files that we need primarily come from web servers and database applications. We use several assumptions related to log configurations that run on the web server.

- Log files that created by apache did not record HTTP POST parameters.
- Mysql application's log file records all the queries that occur.
- Entries in the log file are the original of the authorized application. The attacker does not re-write the contents of log files with fake entries.
- The attacker performs a simple delete operation against the log files.

The first phase, we copied the hard drive from the evidence. At this stage we describe the acquisition on the virtual machine server but not in detail. In the second phase, we perform log analysis using the procedure shown in the Fig. 2. Our method investigates the events recorded on a log file and performs cross-checking with other log files. When conducting an investigation of event information, there is an opportunity for investigators to use two analytical methods that is using rule-based software detection or manual detection. The final phase is the creation of reports that the timeline of attacks and evidence found.

A. Data Acquisition

Data acquisition of the VM web server begins with a snapshot of the last condition of the web server [12]. After the snapshot is finished, the server is suspended for a while then copies the hard disk artifact into forensic media storage. In virtual environment, performing bit-to-bit copy is same as copying the VM files (such as vmdk, vmx, and vmem) to other storage media. The VM artifacts are duplicated for the analytical process and then record the hash value to maintain the integrity of the data.

B. Data Examination and Analysis

There are two options that forensic investigators can do when analyzing VMs artifacts:

a) Continue a suspended VM and analyze live machines using normal procedures. Using this method, the investigator can search and analyze evidence in a flexible manner such as conducting live forensic testing. The disadvantage of this technique is that during resume process many files stored on the hard disk might change including information stored in memory, which may damage evidence.
b) Analyze VM files without resuming suspended machines. This technique works directly with the VM files that are stored on forensic media.

The second technique used an FTK Imager to retrieve files that we need from the VMDK file and then we perform an application analysis. In terms of the application analysis, we seek for unusual entries in log files, check for files created or modified around suspected time of attack, and cross-check between them to find relationships between them. We considered two main logs to be analyzed first; that is web server log and database log. If no attack indication has been found from them, additional log files are required. It can also be used to combine the information found in the two files before. The log analysis step is described in the flowchart shown in Fig. 2.

1) Web Server Log File

In Apache, there are two important files used as forensic material that is access.log and error.log. Usually, they can be found in the same folder i.e. /var/log/apache/. We start with the hypothesis of any hint of sqli attack. We can start an investigation of the access.log file because it contains all requests that occur to the web server. After that, we can analyze the error.log file to look for other evidence. In cases of access.log or error.log with small size, we can observe the logs manually looking for a specific keyword using things like grep or other searching software. But when we deal with a large size of file log, we should use automated software such as scalp, with manual inspection. Scalp is an open source web log analyzer for Apache web server. It reads the apache log and performs log analysis for possible attacks using provided rule

sets [13]. This can minimize the time required for the investigator to find the attack fingerprint.

To detect sqli we check existence of specific keyword in access.log such as “union”, “select”, single-quote “‘”, “into file”, and some other SQL keywords. In the case of the GET method, all parameters that go to the server are logged to access.log so that sqli traces are easy to be found. But access.log file does not contain all data of the HTTP POST method. We only found the header form of POST method in the access.log, so we used mysql.log to see if the POST parameters sent were malicious queries. There are a few things to be considered when doing an investigation on the access.log file:

- The keywords used to detect sqli should be appropriate. Investigators can use regular expressions for detection of a typical sqli attack.
- If any suspected sqli entries have been found in the log, look for the initial entry of the traffic in the log file to create a timeline of the attack. Record the IP address of the client as a suspect candidate.
- Investigator needs to observe the status code of the HTTP request. A status of 404 can also give an indication that the attacker is doing web page enumeration.
- In case of sqli with POST method has the characteristics that there are POST headers with the similar URL accessed sequentially within the specified time range. The trace is created when an attacker performs a POST method attack on a URL even with different size payloads.

Record important information obtained from the investigation such as IP address, date, time, size of payload, HTTP method, and URL. The record becomes a result of the examination of the web server log. We can obtain information based on Fig. 3.

a) IP address **192.168.179.1** is suspected attacker.
b) The attack starts at 03/Apr/2018 23:20:23 +0700 UTC.
c) The attack using GET method to exploit URL “www.presiden.info/sqli_1.php?title=”.

Access.log **GET** method:
192.168.179.1 - - [03/Apr/2018:23:20:23 +0700] "**GET** /sqli_1.php?title=**admin' or '1'='1**&action=search HTTP/1.1" 200 1382 "http://www.presiden.info/sqli_1.php?title=Joe&action=search" "Mozilla/5.0 AppleWebKit/537.36 Chrome/65.0.3325.181 Safari/537.36"

Fig. 3 An indication of SQL Injection using GET method

```
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 1321 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 4372 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 4386 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 4386 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 4407 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 4425 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:48:01 +0700] "POST /sqli_6.php HTTP/1.1" 200 4464 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:49:26 +0700] "POST /sqli_6.php HTTP/1.1" 200 4419 "-" "Mozilla/5.0 (X11; U; Linux i686;
192.168.179.1 - - [03/Apr/2018:22:49:40 +0700] "POST /sqli_6.php HTTP/1.1" 200 4372 "-" "Mozilla/4.0 (compatible; MSIE 8.0
192.168.179.1 - - [03/Apr/2018:22:49:40 +0700] "POST /sqli_6.php HTTP/1.1" 200 4478 "-" "Mozilla/4.0 (compatible; MSIE 8.0
192.168.179.1 - - [03/Apr/2018:22:53:24 +0700] "POST /sqli_6.php HTTP/1.1" 200 4372 "-" "Mozilla/5.0 (Windows; U; Windows
192.168.179.1 - - [03/Apr/2018:22:53:24 +0700] "POST /sqli_6.php HTTP/1.1" 200 4397 "-" "Mozilla/5.0 (Windows; U; Windows
192.168.179.1 - - [03/Apr/2018:22:54:59 +0700] "POST /sqli_6.php HTTP/1.1" 200 1346 "-" "Mozilla/5.0 (Windows; U; Windows
192.168.179.1 - - [03/Apr/2018:22:54:59 +0700] "GET /tmpuecpj.php HTTP/1.1" 404 469 "-" "Mozilla/5.0 (Windows; U; Windows
```

Fig. 4 An indication of SQL Injection using POST method

TABLE I
SUSPICIOUS ENTRIES RECORDED IN ACCESS.LOG

No	Log entries
1	192.168.179.1 - - [03/Apr/2018:22:54:59 +0700] "**POST** /sqli_6.php HTTP/1.1" **200 1346** "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/4.0.202.0 Safari/532.0"
2	192.168.179.1 - - [03/Apr/2018:22:54:59 +0700] "**GET** /tmpuecpj.php HTTP/1.1" **404 469** "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/4.0.202.0 Safari/532.0"
3	192.168.179.1 - - [03/Apr/2018:22:55:14 +0700] "**POST** /sqli_6.php HTTP/1.1" **200 4397** "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/4.0.202.0 Safari/532.0"

SQL Injection using POST method is often difficult to detect because they are similar to other normal POST traffic in access.log files. But we can pay close attention to a similar URL accessed sequentially within the specified time range, as shown in Fig. 4. There are various payload sizes recorded as POST parameters. We can verify that POST payload using other information was founded in the database log.

When performing POST method analysis on the access.log file, investigator persistence is required. Based on Table I, the log number 1 indicates a header with POST method that looks like normal access. We did not know what kind of packet the IP address had sent to the server via the POST parameter. In log number 2, with a very close time difference, the same IP address tries to access the /tmpuecpj.php page using the GET method but the page does not exist and the response is 404. Log number 3 with the time contiguous to log number 1 and log number 2 tries to access the sqli_6.php page with a larger payload size from log number 1. That log is our concern for the next action investigation.

2) Database Log File

Mysql has three standard log files that can be used as forensic materials. They are general query logs, error logs, and slow query logs. By default, mysql does not enable the log facility, therefore it is recommended to the administrator to enable the feature. General query logs and error logs are the two most important features of the investigation process. General query logs are usually stored in the mysql.log file. The mysql.log file provides information about query activity not

found in the web server log. In the case of sqli attacks using the POST method, the investigator can perform a query search stored within the POST parameter through the file. While error.log contains information about errors that occur while the server is running.

The investigator can start his investigation through the mysql config file, my.cnf, to check whether the mysql logging facility is enabled and the file name used. In this case, the log feature is enabled and stored in files named mysql.log, error.log, and mysql-slow.log. We perform investigation priorities with mysql.log, error.log and mysql-slow.log. Based on Table I, POST parameters of log number 1 can be traced using keywords based on time range of events, 03/Apr/2018 22:54:59. Based on Table II, we found the relationship that log number 1 and log number 2 in Table I are sequences of bound events. The attacker tries to write a file named "tmpuecpj.php" in the "/var/www/presiden/" directory via the SQL query in the POST parameter of log number 1 and then they try to access the tmpuecpj.php file. But their efforts were unsuccessful, as we saw from the response server 404.

TABLE II
RECORDED QUERY BASED ON TIME RANGE OF EVENTS

Query Log
180403 22:54:59 88 Connect root@localhost on 88 Init DB presiden 88 Query SELECT * FROM movies WHERE title LIKE '%%' LIMIT 0,1 INTO OUTFILE '**/var/www/presiden/tmpuecpj.php'** LINES TERMINATED BY 0x3c3f7068700a6966202869737365742824 5f524551554553545b2275706c6f616420e616d653d75706c6f61642076616c75653d75706c6f61643e3c2f666f726d3e223b7d3f3e0a-- #%' 88 Quit

We apply searches based on keywords commonly used by attackers when writing or accessing files through SQL injection vulnerabilities like outfile, dumpfile, and load_file. Then, we found evidence when the attacker wrote a malicious file. We can check the existence of that file in the known directory or we can re-examine the web server logs to obtain any other suspicious activity.

TABLE III
SUSPICIOUS QUERY

Query Log
180501 7:12:05 52 Connect root@localhost on 52 Init DB presiden 52 Query SELECT * FROM movies WHERE title LIKE '%=joe' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,0x61646d696e61646d696e00202020203c68c2f74643e0a202020203c2f54523e223b0a202020207d0a202020207d0a20202020 6563686f20223c2f5441424c453e223b0a202020203f3e0a0a202020203c2f626f64793e0a202020203c2f68746d6c3e,NULL INTO DUMPFILE **'/var/www/html_presiden/upload.php'**#%' 52 Quit

TABLE IV
SUSPICIOUS ENTRIES IN ACCESS.LOG

No	Log entries
1	192.168.179.1 - - [01/May/2018:07:12:05 +0700] "**POST /sqli_6.php** HTTP/1.1" **200 4011** "http://www.presiden.info/sqli_6.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"
2	192.168.179.1 - - [01/May/2018:07:13:45 +0700] "**GET /upload.php** HTTP/1.1" **200 3254** "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"
3	192.168.179.1 - - [01/May/2018:07:14:17 +0700] "**POST /upload.php** HTTP/1.1" **200 3299** "http://www.presiden.info/upload.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"
4	192.168.179.1 - - [01/May/2018:07:14:23 +0700] "**GET /rcd.php** HTTP/1.1" **200 1065** "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"

We found connection between queries in Table III with logs in Table IV. Queries in Table III are the contents of the POST parameter of log number 1. It is confirmed from log number 2 that shows the attacker attempted to access the upload.php page result from log number 1 with responses 200 from the server. Unfortunately we cannot find the contents of the POST parameter log number 3, because the server does not record the contents of the POST parameters. However, we can deduce from log number 4 that log number 3 sends an rcd.php file to the server.

3) Additional Logs

Web forensic analysis can be done by examining additional queries that come from other logs in the web server. There are certain web applications that include special log features for themselves. Investigators can use that log file to obtain indications of attack. It can be another source of information that confirms evidence at the previous source. In this paper, we do not explain in detail how to analyze logs from the other sources, but its opportunities as forensic materials are wide open such as antivirus server logs, event logs, service logs, firewall logs and system logs also help in tracking a web server incident.

4) Searching Web Shell

Web shell is a post-exploitation tool, attacker first has to find a vulnerability in the web server, exploit it, and upload their web shell to the server [14]. They will leave enough footprints except for those who are already skilled and eliminating tracks. We use two techniques to trace the existence of a web shell within the server. The first technique uses evidence that we found on the web server logs and database logs before. We use time range analysis of events. We utilized access.log and mysql.log of the web server as primary source. From the evidence we find earlier, there are some files that are inserted by the attacker inside the web server. There are

"tmpuecpj.php", "upload.php", and "rcd.php". Then we check the existence of the file and we perform static code analysis.

The difficulty of the first technique is if an attacker creates a file name similar to a normal file name on the web server. Investigators should search from thousands of files on the web server. But, this can be minimized by analyzing the time range of attacks that occur so as to obtain initial conclusions about the web shell location. Therefore, analysis of web server logs and database logs become important as the starting points for the investigator to find a web shell.

The second technique utilized an automatic web shell detector. There are some commercial or freeware web shell detectors available. Some web shell detectors use a web signature database known to identify a web shell, while other detectors perform code analysis based on commonly used functions as part of a web shell. The disadvantage of this technique is when webshell uses obfuscation technique in order to avoid detection, web shell detector cannot detect it. Because the webshell's detection method doesn't work properly, so it makes false positive results.

We used web shell detector to detect malicious PHP file in our web page directory. Web shell detector detected "rcd.php" as a suspicious file, but not "upload.php", as shown in Fig. 5. The rcd.php contains obfuscated program code obfuscated using "eval(gzinflate (base64_decode ()))" method. While upload.php is a stager file used to upload the rcd.php web shell. This confirms the correlation of log number 3 and log number 4 of Table IV.

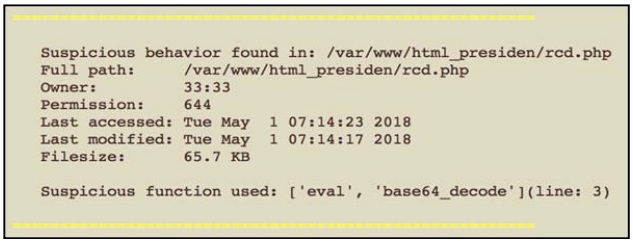

Fig. 5 The rcd.php detection results

C. Reporting

The last stage of the forensic process is make a report. We create reports in timeline format as in Table V.

TABLE V
INVESTIGATION RESULTS IN TIMELINE EVENTS

Time	Log entries	Comment
4/3/2018 10:54:59 PM	192.168.179.1 - - [03/Apr/2018:22:54:59 +0700] "POST /sqli_6.php HTTP/1.1" 200 1346 "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/4.0.202.0 Safari/532.0"	First attack attempt; Confirm at query in mysql.log; Write tmpuecpj.php into server
4/3/2018 10:54:59 PM	88 Query SELECT * FROM movies WHERE title LIKE '%%' LIMIT 0,1 INTO OUTFILE '/var/www/presiden/tmpuecpj.php'	
4/3/2018 10:54:59 PM	192.168.179.1 - - [03/Apr/2018:22:54:59 +0700] "GET /tmpuecpj.php HTTP/1.1" 404 469 "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US) AppleWebKit/532.0 (KHTML, like Gecko) Chrome/4.0.202.0 Safari/532.0"	Unsuccessful file writing in previous step.
4/3/2018 11:20:23 PM	192.168.179.1 - - [03/Apr/2018:23:20:23 +0700] "GET /sqli_1.php?title=admin' or '1'='1&action=search HTTP/1.1" 200 1382 "http://www.presiden.info/sqli_1.php?title=Joe&action=search" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36"	SQL Injection GET method
5/1/2018 7:12:05 AM	192.168.179.1 - - [01/May/2018:07:12:05 +0700] "POST /sqli_6.php HTTP/1.1" 200 4011 "http://www.presiden.info/sqli_6.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"	SQL Injection POST method; Confirm at query in mysql.log; Write upload.php into server.
5/1/2018 7:12:05 AM	52 Query SELECT * FROM movies WHERE title LIKE '%=joe' UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,0x61646d69793e0a202020203c2f68746d6c3e,NULL INTO DUMPFILE '/var/www/html_presiden/upload.php'#%'	
5/1/2018 7:13:45 AM	192.168.179.1 - - [01/May/2018:07:13:45 +0700] "GET /upload.php HTTP/1.1" 200 3254 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"	Successful file writing of upload.php; Get 200 response server.
5/1/2018 7:14:17 AM	192.168.179.1 - - [01/May/2018:07:14:17 +0700] "POST /upload.php HTTP/1.1" 200 3299 "http://www.presiden.info/upload.php" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"	upload.php page send unknown data to server using POST method.
5/1/2018 7:14:23 AM	192.168.179.1 - - [01/May/2018:07:14:23 +0700] "GET /rcd.php HTTP/1.1" 200 1065 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.117 Safari/537.36"	Unknown page exists after POST method using upload.php.
-	/var/www/tmpuecpj.php	File does not exist in server.
-	/var/www/upload.php	File exists; Suspected as malicious code; Created at 5/1/2018 7:12:05 AM.
-	/var/www/rcd.php	File exists; Suspected as malicious code; Created at 5/1/2018 7:14:17 AM; File contains encoding base_64.

V. DISCUSSION AND CONCLUSION

We can summarize several conclusions from the obtained results. SQL Injection attack and use of web shell leave traces on a web server. There are some very important log files as forensic materials. Our method is to utilize log files from web server applications and database applications as the primary

source. We perform cross-analysis of the information found on access.log and mysql.log using suspicious time and keyword parameters to seek for evidence of sqli attack and web shell. These results can help to reconstruct the timeline of attacks and it can be also as valid evidence in the court of law.

In the scenario we created, the forensic methods we did successfully identified sqli attacks and detected a PHP web shell inside the compromised server. With the limitation of our work, there is space for researchers to investigate a better method to conduct web forensic analysis.

REFERENCES

[1] P. Technologies, "Web Application Attack Statistics 2017." (Online). Available: https://www.ptsecurity.com/upload/corporate/ww-en/analytics/WebApp-Attacks-2017-eng.pdf.

[2] R. U. Putri and J. E. Istiyanto, "Analisis Forensik Jaringan Studi Kasus Serangan SQL Injection pada Server Universitas Gadjah Mada," *Indones. J. Comput. Cybern. Syst.*, vol. 6, no. 2, p. 12, 2012.

[3] R. J. Manoj, D. A. Chandrasekhar, and M. D. A. Praveena, "An Approach to Detect and Prevent Tautology Type SQL Injection in Web Service Based on XSchema validation," *Int. J. Eng. Comput. Sci.*, no. 1, p. 5, 2014.

[4] B. Nagpal, N. Chauhan, and N. Singh, "A Survey on the Detection of SQL Injection Attacks and Their Countermeasures," *J. Inf. Process. Syst.*, vol. 13, no. 2017, p. 14.

[5] B. Dickson, "Why are web applications attractive targets for hackers?," *TechTalks*, 29-Feb-2016. (Online). Available: https://bdtechtalks.com/2016/02/29/why-are-web-applications-attractive-targets-for-hackers/. (Accessed: 10-Apr-2018).

[6] "Web Shells – Threat Awareness and Guidance." (Online). Available: https://www.us-cert.gov/ncas/alerts/TA15-314A. (Accessed: 01-Apr-2018).

[7] N. Šuteva, A. Mileva, and M. Loleski, "Finding forensic evidence for several web attacks," *Int. J. Internet Technol. Secur. Trans.*, vol. 6, no. 1, p. 64, 2015.

[8] A. Fry, "A Forensic Web Log Analysis Tool: Technique and Implementation," Thesis Dissertation, Department of Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada, 2011.

[9] A. Lazzez and T. Slimani, "Forensics Investigation of Web Application Security Attacks," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 3, pp. 10–17, Feb. 2015.

[10] L. Palkmets, "Forensic Analysis," *ENISA*, p. 68, 2016.

[11] O. Segal, "Web Application Forensics: The Uncharted Territory." Sanctum, 2002.

[12] M. Hirwani, Y. Pan, B. Stackpole, and D. Johnson, "Forensic Acquisition and Analysis of VMware Virtual Hard Disks," Rochester Institute of Technology, 2012.

[13] S. Zakharchenko, "apache-scalp: Scalp!/Anathema is a log analyzer for web server (Apache, nginx) (Python3)," 01-Apr-2018. (Online). Available: https://github.com/nanopony/apache-scalp. (Accessed: 01-May-2018).

[14] S. Agisilaos, "Detecting Malicious Code in a Web Server," Departemen Of Digital Systems, University Of Pireaus, Athens, 2016.