Finding Forensic Evidence for Several Web Attacks

Nataša Šuteva, Aleksandra Mileva
Faculty of Computer Science
University Goce Delčev
Štip, Republic of Macedonia
{natasa.suteva, aleksandra.mileva}@ugd.edu.mk

Mario Loleski
Forensic Department
Ministry of Interior of the Republic of Macedonia,
Skopje, Republic of Macedonia
mario_loleski@moi.gov.mk

Abstract—**Symantec Internet Security Threat Report 2014 is showing a horrified fact, that when an attacker looked for a site to compromise, one in eight sites made it relatively easy to gain access. Digital forensics is one of our biggest line of defense against cyber criminals, because it provides evidence against them. For attacks against web applications, web application forensics is the branch which gives most of the answers. First, the victim machine usually gives some data, which are then used for identifying possible suspects, and this is followed by forensic analysis of suspects' devices, like computers, laptops, tablets, and even smart phones. In this paper, we use an attack scenario against the known vulnerable web application WackoPicko, using several web attacks: SQL Injection, stored and reflected XSS, remote file inclusion, and command-line injection. We use post-mortem computer forensic analysis of attacker and victim machine to find some artifacts in them, which can help to identify and possible to reconstruct the attack, and most important, to obtain valid evidence which holds in court. We assume that the attacker was careless and did not perform any anti-forensic techniques on its machine.**

Keywords- Web Application Forensics; SQL Injection; File Inclusion; stored XSS; reflected XSS; command-line injection.

I. Introduction

Vulnerability scans of public websites carried out in 2013 by Symantec's Website Vulnerability Assessment Services found that 77 percent of sites contained vulnerabilities, and 16 percent of them were classified as critical vulnerabilities that could allow attackers to access sensitive data, alter the website's content, or compromise visitors' computers (Symantec, 2014). The OWASP (Open Web Application Security Project) Top Ten (2013) each year offers a list of the most critical Web application vulnerabilities for the previous year, and for the 2013, the list includes different types of injection, broken authentication and session management, cross-site scripting, secure misconfiguration, etc. Many organizations lose their reputation or revenue, because of various hackers' attacks. Hacking government institutions' web sites is today means of war and terrorism. The cybercrime is a global problem, and the computer forensics is one way to combat it. Computer forensics prepares legal evidences and give answers to many questions of legal systems related to computers. Analyzed forensic images are the primary evidence. Web Application Forensics is a special branch of the Digital Forensics, which deals with web attacks. It usually starts with an analysis of various log files in the victim machine (Segal, 2002).

We chose to investigate five types of attacks, SQL injection, stored and reflected XSS, remote file injection, and command-line injection, which are usually conducted through a web browser. We are interested in what kind of post-mortem forensic artifacts can be found after performing each attack separately, on the attacker and victim machine. Also, we assume that the attacker did not perform any anti-forensic techniques (format, wipe, log edits, etc.) on both machines. As a tested web application, we use known vulnerable WackoPicko, first introduced by Doupe et al. (2010). We are aware that conducted research is very platform specific, so our results holds for the dominant Apache web server and Kali Linux attacker's machine. But similar artifacts can be also expected on other related attacker/victim platforms, too.

We showed that all types of conducted attacks leave many traces on both machines, most of them in log files on the victim and web history in the attacker's machine. Maybe, command-line injection attack leaves the fewest artifacts on the attacker' machine.

After Introduction Section, Section II is devoted to attacking scenario, including a short description of vulnerable web application WackoPicko, and detailed description of the five performed attacks: SQL injection, stored and reflected XSS, remote file inclusion and command-line injection. In Section III we give a brief overview of performing forensic analysis of both machines, followed by discussion of the results and final conclusions.

A. Previous work

To our knowledge, there are no many papers for forensic analysis of specific web attacks. Andrade and Gan (2012) investigate passive attacks for determination of vulnerabilities of Linux Ubuntu server, using Linux BackTrack 5 tools, including Metasploit, Nessus, Whatweb, Nmap, PHP-Backdoor and Weevely. They use netstat tool and server log files for forensic investigation of the attacks. Good forensics analysis of Linux RAM is given in (Urrea, 2006). Shulman and Waidner (2014) show how digital signatures from DNSSEC can be useful in forensic analysis. Snow et al. (2011) propose a new framework for enabling fast and accurate detection and forensic analysis of code injection attacks.

II. ATTACK SCENARIO

For the attack, we use virtual WMware machine with installed Kali Linux version 1.0.9a (which is Debian derivate) and with IP address 192.168.60.163.

A. Vulnerable web application WackoPicko

The vulnerable WackoPicko application is a photo sharing and photo-purchasing site, where users can upload photos, browse other user's photos, comment on photos, and purchase the rights to a high-quality version of a photo. It has 10 vulnerabilities accessible without authentication (reflected and stored XSS, reflected XSS behind JavaScript, predictable Session ID for admin, weak admin password, reflected SQLI, command-line injection, file inclusion, unauthorized file exposure, and parameter manipulation), and 6 vulnerabilities accessible after logging into the web site (multi-step stored XSS, stored SQLI, directory traversal, forceful browsing, logic flaw and reflected XSS behind Flash).

The web server hosting WackoPicko and used in our experiments was run on the OWASP Broken Web Applications Project virtual machine [8], which has numerous intentionally vulnerable applications (we ignore other applications). The following technologies are used: Apache 2.2.14 on Linux Ubuntu 10.04.1, PHP/5.3.2-1ubuntu4.5 with Suhosin-Patch, and MySQL 5.0.67. The IP address of the victim server is 192.168.60.141.

B. Conducted attacks

We conducted five types of attacks on the WackoPicko, including:

- SQL Injection on the login form;
- Stored XSS on the guestbook page;
- Reflected XSS on the search form;
- Remote file inclusion with null byte injection;
- Command-line injection on the password field.

SQL Injection on the login form is done by injecting a known string **1'or 1=1#**, which allow us to login without a password as Sample User.

On the guestbook page visitors can leave comments to every picture. These comment fields are not properly escaped, so we can use them for stored XSS attack. We insert the following script **<script>location=" http://www.ugd.edu.mk/ index.php/en;"</script>** in the comment on the picture ugd_logo.jpg (we uploaded this picture previously, and it is saved as 4 images on the server: ugd_logo.jpg, ugd_logo.128.jpg, ugd_logo.128_128.jpg and ugd_logo.550.jpg). Whenever user visits the guestbook page and try to see the targeted picture in full size, the attack is triggered and the JavaScript code will be executed, redirecting the visitor to the link www.ugd.edu.mk.

WackoPicko has search.php script with search field in which, the query parameter is vulnerable to reflected XSS attack. We have used one of the standard examples for performing this kind of attack, which returns the PHPSESSID session cookie in the dialog box of the user's web browser:

<script>alert(document.cookie)</script>

In WackoPicko web application, the admin page is vulnerable to file inclusion. We use remote file inclusion in two ways. In the first way, using the browser, we upload two public accessible shells b374k-shell.php version 2.8 (used as b374k-2.8.php) and c99shell.php and one picture ugd.jpg, using the form for uploading pictures. We can run the shells using admin page and null byte injection, for example

http://192.168.60.141/WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/b374-2.8/b374-2.8%00.php.

Using this shell, we have removed the folder cart (**rm -rf cart**) and we have created new folder papka (**mkdir papka**). Using the shell c99shell.php, we have edited the page http://192.168.60.160/WackoPicko/test.php, by adding a new link to the page http://www.stip.gov.mk. We chose two different shells, because b374k-2.8.php sends commands to the server as part of the POST request body, and c99shell.php sends commands as parameters in the URL.

Other remote file inclusion attack uses the known hacker tool Metasploit (from Kali Linux) and its Reverse TCP Payload command, for generating the reverse shell payload /root/napad.php on the attacker's machine. The procedure for uploading and using this shell on the victim machine is the same as previous. Using this shell, we have deleted the file error.php (**rm error.php**) and uploaded the file stip.html (**upload /root/stip.html**).

There is a passcheck.php script for checking password strength. Its password field is vulnerable to command-line injection. Using this vulnerability, we have deleted the file tos.php (**123|rm -f tos.php #**).

Usually, attackers use IP spoofing, but in this case we are not doing that, because we want to see the artifacts left on the attacker's machine, by performing some web attacks.

III. Performing forensic analysis

In forensics expertise there are three main phases: acquisition, analysis, presentation.

In the acquisition phase, the state of digital system, with all allocated and unallocated areas, is saved for later analysis. This copy is called an image. For the preserving integrity of the image, the hash result of the image is calculated and saved. For making the image we used the free AccessData FTK Imager 3.1.

The analysis phase takes the acquired data and involves examination of every piece of data from the evidence. We are starting with hypothesis for what kind of attack can be found. This phase includes search with keywords (names of files or folder that we assume that are produced from the attack). For analysis we are using two forensic tools - Autopsy (The Sleuth Kit, freeware), and WinHex (freeware).

In the last presentation phase, conclusions are made from the analysis and it is necessary to prepare a documentation that can be in a readable format for people who work in courts. This document is called forensic report. Clearly, the final part with forensic report is not done in this research.

A. Analysis of the victim server

The analysis of the victim image is done with the following forensic techniques:

- File system analysis
- Recovering of deleted files and folders
- Log file analysis
- Keyword search
- Overview and keyword search of the swap area

In terms of the file system analysis and recovering of deleted data, the investigator seeks for unusual entries in the log files and specially POST data, script abuses and unusual parameters in the URL, files created and/or modified around the suspected time of the attack, many accesses from the same IP address, etc. We have found that several new php scripts are present in the system, and that some files are recently changed. We can use their names in keyword search for both machines. We have examined the log files: error.log, access.log, mysql.log, and their versions with different extensions, for example, log.1, or log.1.gz. Again, we assume that we are dealing with careless attacker that did not change the log files. From the log files analysis, we can first determine that the attack on this server is performed from the IP address 192.168.60.163; and looking at user agent string (**"Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"**), we can determine that the attacker's machine runs on 64-bit version of Linux operating system, with Mozilla Firefox 24.0 web browser based on Gecko. Additionally, Iceweasel is a Debian rebranded version of Firefox, which means that the Linux operating system is Debian derivate. We have obtained also traces that show the use of b374-2.8.php, c99shell.php and napad.php shells, and even the modification of the file test.php with c99shell (see Table I) from 192.168.60.163.

In mysql.log file one can find the following traces of SQL injection and stored XSS attacks, which were conducted on December 12, 2014 from 11:28:06 till 11:48:12:

141220 11:28:06 558 *Connect wackopicko@localhost on*

558 Init DB wackopicko

*558 Query SELECT * from `users` where `login` like '-1'or 1=1#' and `password` = SHA1(CONCAT('', `salt`)) limit 1*

141220 11:43:16 571

571 Query INSERT INTO `pictures` (`id`, `title`, `width`, `height`, `tag`, `filename`, `price`, `high_quality`, `created_on`, `user_id`) VALUES (NULL, 'ugd_logo', '128', '128', 'ugd_logo', 'ugd_logo/ugd_logo', '2', 'MzEzMjM5Mw==', NOW(), '12')

572 Query SELECT pictures.filename, pictures.id, pictures.user_id, users.login from pictures, users where pictures.id != '17' and pictures.tag like 'ugd_logo' and pictures.user_id = users.id order by RAND() limit 2

573 Query INSERT INTO `comments_preview` (`id`, `text`, `user_id`, `picture_id`, `created_on`) VALUES (NULL, '<script>location=\"http://www.ugd.edu.mk/index.php/en/\";</script>', '12', '17', NOW())

```
141220 11:48:12 574
574 Query INSERT INTO `comments` (`id`, `text`, `user_id`, `picture_id`, `created_on`) VALUES (NULL, '<script>location=\"http://www.ugd.edu.mk/index.php/en/\";</script>', '12', '17', '2014-12-20 11:48:12')
```

The first one, show us how the attacker’s SQL injection query is processed, the second one is for uploading of a picture, and the third one tells us the creation date and time for stored XSS attack as a comment under the picture with id 17 from the user with user_id 12. These traces can be connected to the machine 192.168.60.163 using logs from access.log file that show accesses to the WackoPicko’s scripts login.php, upload.php, preview_comment.php and add_comment.php in the same times. One example is the following trace for uploading a file

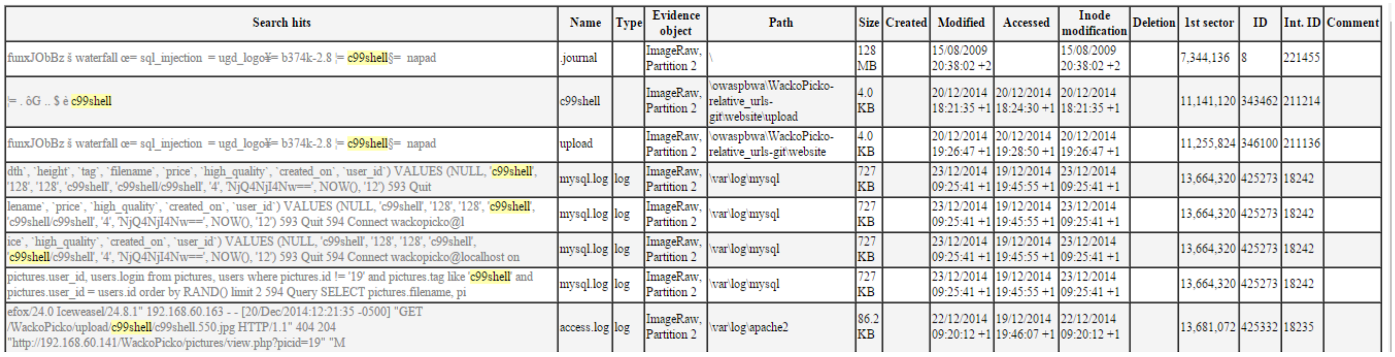
```
192.168.60.163 - - [20/Dec/2014:11:43:16 -0500] "POST /WackoPicko/pictures/upload.php HTTP/1.1" 303 20 "http://192.168.60.141/WackoPicko/pictures/upload.php" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"

192.168.60.163 - - [20/Dec/2014:11:43:16 -0500] "GET /WackoPicko/pictures/view.php?picid=17 HTTP/1.1" 200 1211 "http://192.168.60.141/WackoPicko/pictures/upload.php" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"
```

TABLE I PART OF THE KEYWORD SEARCH ON THE VICTIM MACHINE WITH COMMENTS, PRESENTED ALSO AS ENTRIES IN ACCESS.LOG

No.	Search hints	Name	Path	Modified	Accessed	Inode Modification	Comment
1	192.168.60.163 - - [20/Dec/2014:12:03:17 -0500] "GET /WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/b374k-2.8/b374k-2.8%00.php HTTP/1.1" 200 1780 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"	access.log	\var\log\apache2	22/12/2014 09:20:12 +1	19/12/2014 19:46:07 +1	22/12/2014 09:20:12 +1	Successful use of the script b374.php with null byte injection and GET request from IP address 192.168.60.163
2	192.168.60.163 - - [20/Dec/2014:12:05:31 -0500] "POST /WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/b374k-2.8/b374k-2.8%00.php& HTTP/1.1" 200 4008 "http://192.168.60.141/WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/b374k-2.8/b374k-2.8%00.php" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"	access.log	\var\log\apache2	22/12/2014 09:20:12 +1	19/12/2014 19:46:07 +1	22/12/2014 09:20:12 +1	Successful use of the script b374.php with null byte injection and POST request from IP address 192.168.60.163
3	192.168.60.163 - - [20/Dec/2014:12:35:05 -0500] "GET /WackoPicko/admin/index.php?page=http%3A%2F%2F192.168.60.141%2FWackoPicko%2Fupload%2Fc99shell%2Fc99shell%00.php&act=f&f=test.php&ft=edit&d=%2Fowaspbwa%2FWackoPicko-relative_urls-git%2Fwebsite%2F HTTP/1.1" 200 3593 "http://192.168.60.141/WackoPicko/admin/index.php?page=http%3A%2F%2F192.168.60.141%2FWackoPicko%2Fupload%2Fc99shell%2Fc99shell%00.php&act=f&f=test.php&d=%2Fowaspbwa%2FWackoPicko-relative_urls-git%2Fwebsite&" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"	access.log	\var\log\apache2	22/12/2014 09:20:12 +1	19/12/2014 19:46:07 +1	22/12/2014 09:20:12 +1	Successful use of the script c99shell.php with null byte injection and POST request from IP address 192.168.60.163. From the URL, one can see that file test.php is edited (ft=edit).
4	192.168.60.163 - - [20/Dec/2014:13:31:18 -0500] "GET /WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/napad/napad%00.php HTTP/1.1" 200 29 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"	access.log	\var\log\apache2	22/12/2014 09:20:12 +1	19/12/2014 19:46:07 +1	22/12/2014 09:20:12 +1	Successful use of the script pay with null byte injection and GET request from IP address 192.168.60.163

From the log files found in the location presented before, a keyword list can be prepared. In our research, the keyword list consists of the following terms: 1=1#, b374k, c99shell, %00.php, ugd, papka, napad, stip.html, tos.php, www.stip.gov.mk, and www.ugd.edu.mk, cart, error.php.

Part of the useful results of the keyword search with WinHex were the same as those found in log files, and are presented in the Table I. Figure I shows how the results of the keyword search look in WinHex. The keyword www.ugd.edu.mk, beside in the mysql.log, is found also in the comments.myd and comments_preview.myd files with part of database' data. The keywords papka and stip.html are found in the .journal file.



Search hits	Name	Type	Evidence object	Path	Size	Created	Modified	Accessed	Inode modification	Deletion	1st sector	ID	Int. ID	Comment
funxJObBz š waterfall œ= sql_injection = ugd_logo¥= b374k-2.8 ¦= c99shell§= napad	.journal		ImageRaw, Partition 2	\	128 MB		15/08/2009 20:38:02 +2		15/08/2009 20:38:02 +2		7,344,136	8	221455	
¦= . ôG .. $ è c99shell	c99shell		ImageRaw, Partition 2	\owaspbwa\WackoPicko-relative_urls-git\website\upload	4.0 KB		20/12/2014 18:21:35 +1	20/12/2014 18:24:30 +1	20/12/2014 18:21:35 +1		11,141,120	343462	211214	
funxJObBz š waterfall œ= sql_injection = ugd_logo¥= b374k-2.8 ¦= c99shell§= napad	upload		ImageRaw, Partition 2	\owaspbwa\WackoPicko-relative_urls-git\website	4.0 KB		20/12/2014 19:26:47 +1	20/12/2014 19:28:50 +1	20/12/2014 19:26:47 +1		11,255,824	346100	211136	
dth`, `height`, `tag`, `filename`, `price`, `high_quality`, `created_on`, `user_id`) VALUES (NULL, 'c99shell', '128', '128', 'c99shell', 'c99shell/c99shell', '4', 'NjQ4NjI4Nw==', NOW(), '12') 593 Quit	mysql.log	log	ImageRaw, Partition 2	\var\log\mysql	727 KB		23/12/2014 09:25:41 +1	19/12/2014 19:45:55 +1	23/12/2014 09:25:41 +1		13,664,320	425273	18242	
lename`, `price`, `high_quality`, `created_on`, `user_id`) VALUES (NULL, 'c99shell', '128', '128', 'c99shell', 'c99shell/c99shell', '4', 'NjQ4NjI4Nw==', NOW(), '12') 593 Quit 594 Connect wackopicko@l	mysql.log	log	ImageRaw, Partition 2	\var\log\mysql	727 KB		23/12/2014 09:25:41 +1	19/12/2014 19:45:55 +1	23/12/2014 09:25:41 +1		13,664,320	425273	18242	
ice`, `high_quality`, `created_on`, `user_id`) VALUES (NULL, 'c99shell', '128', '128', 'c99shell', 'c99shell/c99shell', '4', 'NjQ4NjI4Nw==', NOW(), '12') 593 Quit 594 Connect wackopicko@localhost on	mysql.log	log	ImageRaw, Partition 2	\var\log\mysql	727 KB		23/12/2014 09:25:41 +1	19/12/2014 19:45:55 +1	23/12/2014 09:25:41 +1		13,664,320	425273	18242	
pictures.user_id, users.login from pictures, users where pictures.id != '19' and pictures.tag like 'c99shell' and pictures.user_id = users.id order by RAND() limit 2 594 Query SELECT pictures.filename, pi	mysql.log	log	ImageRaw, Partition 2	\var\log\mysql	727 KB		23/12/2014 09:25:41 +1	19/12/2014 19:45:55 +1	23/12/2014 09:25:41 +1		13,664,320	425273	18242	
efox/24.0 Iceweasel/24.8.1" 192.168.60.163 - - [20/Dec/2014:12:21:35 -0500] "GET /WackoPicko/upload/c99shell/c99shell.550.jpg HTTP/1.1" 404 204 "http://192.168.60.141/WackoPicko/pictures/view.php?picid=19" "M	access.log	log	ImageRaw, Partition 2	\var\log\apache2	86.2 KB		22/12/2014 09:20:12 +1	19/12/2014 19:46:07 +1	22/12/2014 09:20:12 +1		13,681,072	425332	18235	

Figure 1 Part of the keyword search results on victim machine as given by WinHex

For command-line injection, we look for successful POST requests of the passcheck.php script in the access.log, like

192.168.60.163 - - [22/Dec/2014:03:03:34 -0500] "POST /WackoPicko/passcheck.php HTTP/1.1" 200 1015 "http://192.168.60.141/WackoPicko/passcheck.php" "Mozilla/5.0 (X11; Linux x86_64; rv:24.0) Gecko/20140924 Firefox/24.0 Iceweasel/24.8.1"

An enormous interest was expected to be the swap file (which had 400MB of size) and inside that file were not found any hits, which could be further treated as clues/traces for the interest in the analysis. Or in other words, the SWAP area has proven as a useless target for the task.

B. Analysis of the attacker's machine

The analysis of the suspect's image is done with the following forensic techniques using Autopsy:

- File system analysis and hash comparing
- Log file analysis and internet history files
- OS and third party software artifacts/CLI history overview
- Keyword search
- Preparation of forensic report

According to the report of the incident and the previous analysis on the server, the forensic expert can have a starting point for answering the question - Is the suspect the attacker itself? We actually have the attacker's machine, and the following results are found:

File system analysis and hash comparing

In this case, a complete file search activity is done in which every file and folder is hashed. From the attacked side, the forensic expert can find names and hash values of the files which one thinks that are produced from the incident, and see if some of those values can be found in the suspect's image. If so, there is a starting point for determining that this is the actual attacker. In this case, we have found that the file content and their hash values are appropriate for both sides, so we have a starting point. From Table II, one can see that the shells used in the attack, are the same files on both machines, and also uploaded files ugd_logo.jpg and stip.html are the same. This type of forensic analysis is proven to be very useful.

We can conclude that careless attacker that saves and use shells for an attack, lives evidences for their existence on both machines.

Log file analysis and internet history files

The mentioned activity in general is focused on locating internet artifacts which can be found in web browser temporary storage and browser's history files. In this case, several URL records and cached content is found, which proves that the user was visiting the attacked site. Because in our case, FireFox was used by the attacker, there are several very interesting files. The places.sqlite file keeps track of visited pages and saved bookmarks, the cookies.sqlite file is used for temporary storage of cookie updates, and the formhistory.sqlite file stores values that the user enters into form input fields.

Since our attacking methods are conducted through the interface of the web browser, this analysis can prove the act of the first, third and fourth attack, the methodology (used scripts) and the files that were the object of the intrusion. Sample results are given in Table III and Table IV. Figure 2 shows the forensic artefacts for SQL injection, remote file inclusion and reflected XSS attack, found in the formhistory.sqlite file.

TABLE II SOME RESULTS OF THE FILE SYSTEM ANALYSIS AND HASH COMPARING USING FTK IMAGER AND AUTOPSY

Victim machine file system		**Attacker machine file system**	
Location	**MD5 Hash**	**Location**	**MD5 Hash**
[root]\owaspbwa\WackoPicko-relative_urls-git\website\upload\b374k-2.8\b374k-2.8	cc8d0f697435783610a4c17278e3c51c	[root]\root\Desktop\b374k-2.8.php	cc8d0f697435783610a4c17278e3c51c
[root]\owaspbwa\WackoPicko-relative_urls-git\website\upload\c99shell\c99shell	83f83bb415b98d41fbcae9193b47c984	[root]\root\Desktop\c99shell.php	83f83bb415b98d41fbcae9193b47c984
[root]\owaspbwa\WackoPicko-relative_urls-git\website\upload\ugd_logo\ugd_logo	842753e783da94542dc53d053c4e3687	[root]\root\Desktop\ugd_logo.jpeg	842753e783da94542dc53d053c4e3687
[root]\owaspbwa\WackoPicko-relative_urls-git\website\upload\napad\napad	76da7d37759542cf11316e44ed9c54eb	[root]\root\napad.php	76da7d37759542cf11316e44ed9c54eb
[root]\owaspbwa\WackoPicko-relative_urls-git\website\stip.html	4d34178c417daa8e9d160365e150566f	[root]\root\stip.html	4d34178c417daa8e9d160365e150566f

TABLE III SOME RESULTS FROM LOG ANALYSIS AND INTERNET HISTORY FILES FROM ATTACKER'S MACHINE WITH AUTOPSY

URL	**Program**	**Source File**	**Date Accessed**	**Tags**	**Comment**
http://192.168.60.141/WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/b374k-2.8/b374k-2.8%00.php	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/places.sqlite	2014/12/20 18:03:20	Mozilla History	Using of the script b374.php on the victim machine.
http://192.168.60.141/WackoPicko/admin/index.php?page=http%3A%2F%2F192.168.60.141%2FWackoPicko%2Fupload%2Fc99shell%2Fc99shell%00.php&act=f&f=test.php&ft=edit&d=%2Fowaspbwa%2FWackoPicko-relative_urls-git%2Fwebsite%2F#	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/places.sqlite	2014/12/20 18:41:47	Mozilla History	Using of the script c99shell.php with null byte injection on the victim machine. From the URL, one can see that file test.php is edited (ft=edit).
http://192.168.60.141/WackoPicko/admin/index.php?page=http://192.168.60.141/WackoPicko/upload/napad/napad%00.php	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/places.sqlite	2014/12/20 19:31:21	Mozilla History	Using of the script napad.php with null byte injection on the victim machine.
http://192.168.60.141/WackoPicko/pictures/search.php?query=%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E&x=37&y=12	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/places.sqlite	2014/12/20 19:55:35	Mozilla History	Reflected XSS attack
http://192.168.60.141/WackoPicko/passcheck.php	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/places.sqlite	2014/12/20 20:08:10	Mozilla History	Using of passcheck.php script (for possible command-line attack)

TABLE IV SOME RESULTS FROM WEB COOCKIES FROM ATTACKER'S MACHINE WITH AUTOPSY

URL	Date/Time	Name	Value	Program	Source File
192.168.60.141	2014/12/20 19:33:13	b374k_included	1	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/cookes.sqlite
192.168.60.141	2014/12/20 19:33:13	b374k	fb621f5060b9f65acf8eb4232e3024140dea2b34	FireFox	/img_KaliLinux.E01/vol_vol2/root/.mozilla/firefox/qr82uf4g.default/cookes.sqlite

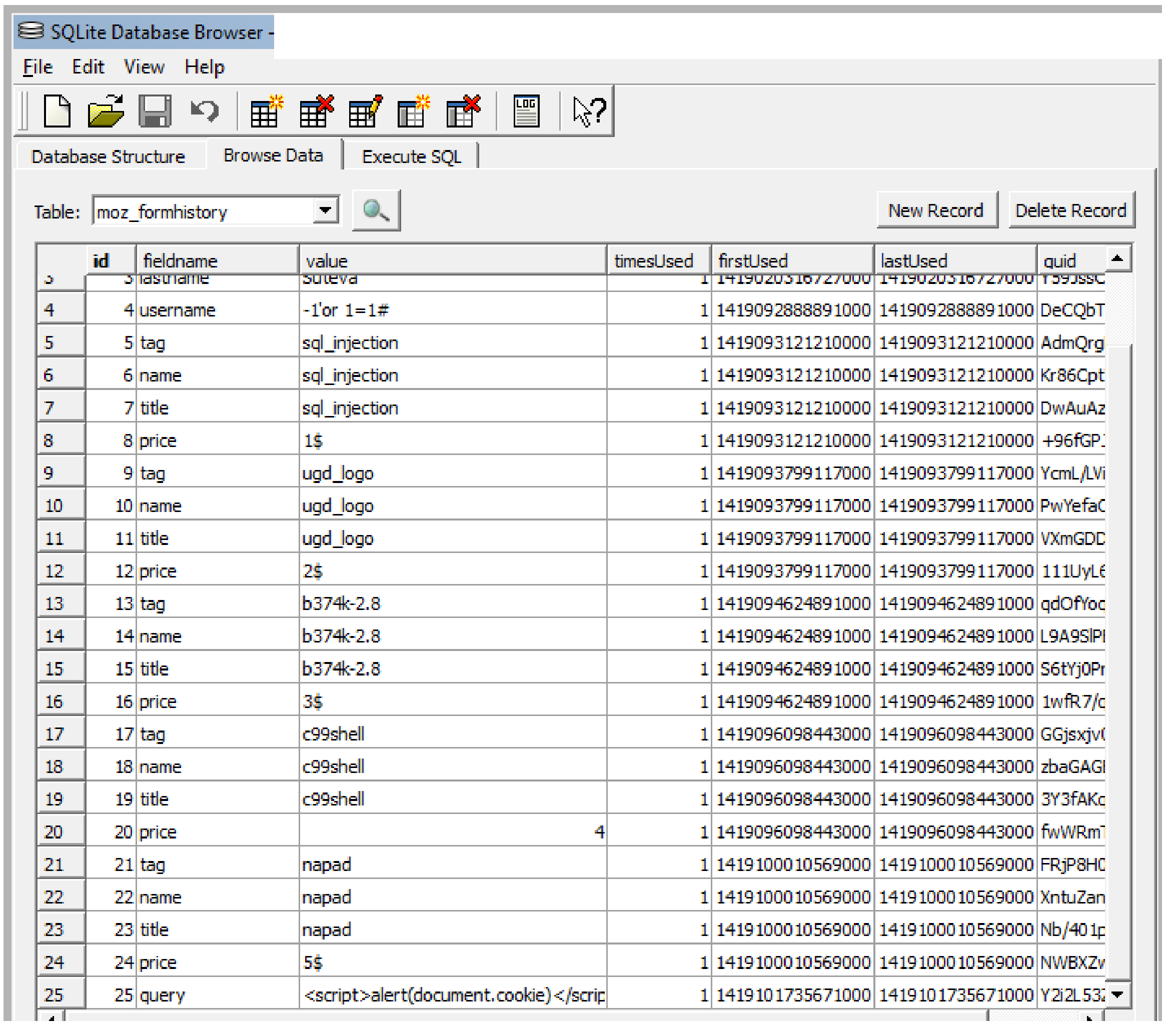


Table: moz_formhistory

	id	fieldname	value	timesUsed	firstUsed	lastUsed	guid
3	3	lastname	Suteva	1	1419020316727000	1419020316727000	Y59JssC
4	4	username	-1'or 1=1#	1	1419092888891000	1419092888891000	DeCQbT
5	5	tag	sql_injection	1	1419093121210000	1419093121210000	AdmQrg
6	6	name	sql_injection	1	1419093121210000	1419093121210000	Kr86Cpt
7	7	title	sql_injection	1	1419093121210000	1419093121210000	DwAuAz
8	8	price	1$	1	1419093121210000	1419093121210000	+96fGP
9	9	tag	ugd_logo	1	1419093799117000	1419093799117000	YcmL/LVi
10	10	name	ugd_logo	1	1419093799117000	1419093799117000	PwYefaC
11	11	title	ugd_logo	1	1419093799117000	1419093799117000	VXmGDD
12	12	price	2$	1	1419093799117000	1419093799117000	111UyL6
13	13	tag	b374k-2.8	1	1419094624891000	1419094624891000	qdOfYoc
14	14	name	b374k-2.8	1	1419094624891000	1419094624891000	L9A9SlPI
15	15	title	b374k-2.8	1	1419094624891000	1419094624891000	S6tYj0Pr
16	16	price	3$	1	1419094624891000	1419094624891000	1wfR7/c
17	17	tag	c99shell	1	1419096098443000	1419096098443000	GGjsxjv
18	18	name	c99shell	1	1419096098443000	1419096098443000	zbaGAGI
19	19	title	c99shell	1	1419096098443000	1419096098443000	3Y3fAKc
20	20	price	4	1	1419096098443000	1419096098443000	fwWRmT
21	21	tag	napad	1	1419100010569000	1419100010569000	FRjP8HC
22	22	name	napad	1	1419100010569000	1419100010569000	XntuZan
23	23	title	napad	1	1419100010569000	1419100010569000	Nb/401p
24	24	price	5$	1	1419100010569000	1419100010569000	NWBXZv
25	25	query	<script>alert(document.cookie)</scrip	1	1419101735671000	1419101735671000	Y2i2L53

Figure 2 Forensic artifacts found in the formhistory.sqlite

Because web browser is the primary way of performing selected attacks, it is normal that we have found evidence in Internet history, and in the saved web cookies, again, if the attacker is careless.

<u>OS and third party software artifacts</u>

A useful forensic artifacts are found in the bash history file (Table V), which show the creation of napad.php script. Apparently the forensic expert can conclude that the user has cleaned the other history of the shell activity, or the shell was not used for other attacks. Also, additional interesting forensic artifact can be the IP address of the suspect's machine, but in this case, the operating system of the suspect machine was using an automatic assigned IP address of the virtual machine. Therefore, the file at the location etc/network/interfaces in this case does not prove anything.

If the attacker had used the command prompt for carrying the attack, the bash_history file keeps the records of executed commands (if it is not erased by the attacker). So, for example, if the attacker uses the Metasploit console, every command will be recorded in the bash_history file.

TABLE V INTERESTED RESULT FROM OS ARTIFACT ANALYSIS

File name	Location	Content
.bash_history	/img_KaliLinux.E01/vol_vol2/root/.bash_history	msfpayload php/meterpreter/reverse_tcp LHOST=192.168.60.163 LPORT=4444 R >/root/napad.php msfconsole

<u>Keyword search</u>

From the keyword search approach we can obtain additional information, some of them are presented in Table VI (for keyword hack.html). Some of the search hits are in temporary database files as meta.db-wal, other are in database file as formhistory.sqlite and places.sqlite, other in the free space, in journal files, etc.

TABLE VI SOME RESULTS FROM KEYWORD SEARCH ANALYSIS ON ATTACKER'S MACHINE

Search hits	Name	Path	Created	Modified	Accessed
⁻20T18:08:04Z †£ ‹] 2014-12-20T18:08:04Z †£ ‹] H ‹] †£ ‹] 5file:///root/napad.php †£ ‹] 9application/x-php †£ ‹] Í †£ ‹] B †¡ †£ ‹] @true š	tracker-store.journal	\root\.local\share\tracker\data	19/12/2014 21:04:50 +1	22/12/2014 08:52:21 +1	22/12/2014 08:50:17 +1
//root/Desktop/c99shell.php ‹[, Y file:///root/Desktop/SQL_Injection.jpg ‹V$ I file:///root/Desktop/stip.html ‹\(Q file:///root/Desktop/ugd_logo.jpeg ‹W 9 file:///root/napad.php ‹]$ I file:///usr/share/appli	meta.db-wal	\root\.cache\tracker	19/12/2014 21:12:57 +1	22/12/2014 08:52:21 +1	22/12/2014 08:51:43 +1
napad ©ø Ý(©ø Ý(FRjP8H0ZTuC6C3Wc. -price4 © ÙŽø © ÙŽøfwWRmTJMSE6ZpjqG5 -titlec99shell © ÙŽø © ÙŽø3Y3fAKq/QMmAalb04 -namec99shell © ÙŽø © ÙŽøzbaGAGEMQueAXXLj3 -tagc	formhistory.sqlite	\root\.mozilla\firefox\qr82uf4g.default	19/12/2014 21:18:11 +1	20/12/2014 19:55:36 +1	22/12/2014 09:06:58 +1
/bookmark:applications _/metadata_ _/info_ _/bookmark_ _bookmark href="file:///root/napad.php" added="2014-12-20T18:09:57Z" modified="2014-12-20T18:26:00Z" visited="2014-12-20T18:09:57Z"_ _	Free space				
ad/napad/M@ • http://192.168.60.141/WackoPicko/pictures/view.php?picid=20L" I file:///root/Desktop/stip.htmlK& Q place:type=6&sort=14&maxResults=10 A place:sort=8&maxResults=10 2 i place:sort=14&type=6&maxR	places.sqlite	\root\.mozilla\firefox\qr82uf4g.default	19/12/2014 21:13:33 +1	22/12/2014 09:11:35 +1	22/12/2014 09:11:35 +1
G· ¹ _Ú ` G· □ „ G· ‚ ¨ G· ¯ Ì G· t ð G·ugd_logo.jpeg c99shell.php b374k-2.8.php SQL_Injection.jpg stip.html VMwareTools-9.2.3-1031360.tar.gz a	home	\root\.local\share\gvfs-metadata	22/12/2014 08:51:20 +1	22/12/2014 08:51:20 +1	22/12/2014 08:51:20 +1

IV. DISCUSSIONS

From the obtained results, we can summarize several results. First, remote file inclusion and use of shells leave many traces on the attacker's and the victim's machine. If the shell carries out commands through a POST request, its successful use is documented (without content of POST body) in victim log files and in the attacker files places.sqlite, formhistory.sqlite and cookies.sqlite. Even more, if the shell carries out commands as parameters in the URL, they are documented in the same way, and even the attack can be reconstructed correctly. If the attacker use Metasploit console, he leaves traces in a batch_history file on its machine.

Examples of our SQL injection and stored XSS attacks use the body of the POST request, so they do not leave any traces in the attacker's browser history files. But SQL injection leaves traces in formhistory.sqlite file and in the mysql.log file. The script from the stored XSS attack is saved in the backend database and it leaves traces in the mysql.log file also, which includes a time stamp and user_id which identifies the user who create this entry. For both attacks, there are traces in the victim log files, which show accesses to the related php scripts (login.php, upload.php, preview_comment.php and add_comment.php) by the attacker machine. This can be used as forensic evidences for these attacks.

Reflected XSS attack where the parameter in the URL is used for conducting the attack, leaves direct forensic evidence in the Internet history of the attacker's machine and in the formhistory.sqlite file.

Command-line injection, conducted as in our attack scenario, does not leave any direct forensic evidence in the attacker machine, except the use of passcheck.php script in the Internet history. Again, the reason is the use of the POST request for carrying the command.

V. CONCLUSIONS

Using post-mortem computer forensic analysis of attacker and victim machine, we found several direct and/or indirect forensic evidences of them, for each conducted attacks in our scenario. Careless attacker that saves and use shells for his attack, leaves evidence on both machines. On the attacker's machine, traces were found in the browser's history files, browser' temporary storage, and bash_history_file. On the victim's machine, traces were found in the file system and in the log files. These artifacts can help to identify and sometimes to reconstruct performed attacks, and even more, they can represent a valid evidence for the court.

REFERENCES

Andrade, J. J. B., and Gan, D. (2012) 'A Forensics Investigation into Attacks on Linux Servers', In: *Issues in cybercrime, security and digital forensics*. University of Strathclyde Continuing Education Centre, Glasgow, UK, pp. 73-80. ISBN 9780947649852.

Autopsy 3.1.0. [online] http://www.sleuthkit.org/autopsy/ (Accessed 1 September 2014).

b374k-2.8.php. [online] https://code.google.com/p/b374k-shell/downloads/detail?name=b374k-2.8.php (Accessed 1 November 2014)

c99shell.php. [online] http://www.4shared.com/file/-sHx3aFm/c99shell.html?locale=en (Accessed 1 September 2014)

Doupe, A., Cova, M., and Vigna, G. (2010) 'Why Johnny Can't Pentest: An Analysis of Black-box Web Vulnerability Scanners'. In C. Kreibich, M. Jahne (Eds.) *Proceedings of the 7th International conference on Detection of Intrusions and Malware, and Vulnerability Assessment - DIMVA'10*, pp. 111-131, Springer Berlin Heidelberg.

FTK Imager 3.1. [Online]. Available: http://accessdata-ftk-imager.software.informer.com/3.1/. (Access Date: 1 September 2014)

Open Web Application Security Project, OWASP Broken Web Applications Project. [online] https://www.owasp.org/index.php/ OWASP_Broken_Web_Applications_Project#tab=Main (Accessed 1 November 2014)

Open Web Application Security Project, "OWASP Top Ten Project". [online] http://www.owasp.org/index.php/Category: OWASP Top Ten Project (Accessed 1 September 2014).

Segal, O. (2002) *Web Application Forensics: The Uncharted Territory.* Sanctum Inc. http://www.cgisecurity.com/lib/WhitePaper_Forensics.pdf (Accessed 1 September 2014).

Shulman, H., and Waidner, M. (2014) 'DNSSEC for cyber forensic', *EURASIP Journal on Information Security 2014,* 2014:16 (doi:10.1186/s13635-014-0016-2).

Snow, K.Z., Krishnan, S., Monrose, F. and Provos, N. (2011) 'SHELLOS: Enabling Fast Detection and Forensic Analysis of Code Injection Attacks'. In Proceedings of *USENIX Security Symposium.* https://www.usenix.org/legacy/event/sec11/tech/full_papers/Snow.pdf (Accessed 1 September 2014).

Symantec (2014) *Internet Security Threat Report 2014,* April 2014. http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf (Accessed 1 September 2014).

Urrea, J. M. (2006) *An Analysis of Linux RAM forensics*, MSc thesis, Naval Postgraduate School, Monterey, USA.

WackoPicko. [online] https://github.com/adamdoupe/ WackoPicko/archive/master.zip (Accessed 1 September 2014).

WinHex. [online] http://www.x-ways.net/winhex/ (Accessed 1 September 2014).