# SECURITY ANALYSIS OF THE ARCHITECTURAL COMPONENTS WITHIN A WEB-BASED REPORTING APPLICATION

Dan Bogdan ZAH[1)] Bogdan MIHAILESCU[2)] Andrei Bogdan RUS[1)] Kurt EDER[2)] Virgil DOBROTA[1)]
[1)] *Technical University of Cluj-Napoca, Communications Department*
*28 Memorandumului, Cluj-Napoca, Romania, +40 (264) 401-200,*
*bogdan_bobb@yahoo.com, {bogdan.rus, virgil.dobrota}@com.utcluj.ro*
[2)] *Frequentis Romania SRL*
*47 Taietura Turcului, Cluj Napoca, Romania, +40 (720) 110-768,*
*{bogdan.mihailescu, kurt.eder}@frequentis.com*

**Abstract: In the work presented herein we conducted a security audit of the architectural components included in any web-based data reporting application. Overall we implemented a total of eleven attacks: four of them targeted the server, four the web application on the client device, two envisaged the wireless router and one exploited the vulnerabilities of the database server. Along with the successful Denial of Service attacks, we managed to extract confidential information (i.e. password hashes) from the system using sniffing, Structured Query Language injection and Cross-Site Scripting attacks. For any successful security breaking presented herein, we proposed a set of techniques that can mitigate the specific vulnerability.**

***Keywords:*** *Denial of Service, Security, web-based application.*

## I. INTRODUCTION

More and more users nowadays connect to different web-based applications to search for needed information, exchange messages, interact with each other, conduct business, pay taxes, perform financial operations etc. On the other hand every moment the web-based services are targeted by malicious programs or attackers intending to exploit possible vulnerabilities. They can cause not only the disruption of services but they gather confidential and valuable information [1]. These vulnerabilities are directly related to implementation flaws that open unwanted backdoors into the application and various misconfigurations of system components upon their deployment. The OWASP (Open Web Application Security Project) community rated the injection vulnerability as being the first one in the Top Ten 2013 Project, with the most common prevalence, the easiest exploitability and the most severe impact [2]. Actually, the SQL (Structured Query Language) injection is a code exploitation technique that tries to get benefits from the vulnerabilities of the interface between web applications and database servers [3].

The XSS (Cross-Site Scripting) is another widespread attack that is targeting web-based applications, being the third one within the OWASP's list. The absence of sufficient validation of user inputs provides the perfect environment for XSS mechanisms to succeed. It has become even more severe with the advent of Web 2.0 technologies, i.e. AJAX (Asynchronous JavaScript and Extensible Markup Language) [4].

The DoS (Denial of Service) attack is a dangerous threat that is performed against various providers because it tries to deprive legitimate users from the services offered by them. DDoS (Distributed Denial of Service) is a particular version of it, involving a cluster of devices that are assaulting the same victim. As a consequence, the attack is enhanced and it is harder to mitigate [6]. DoS attacks were originally classified in two categories: a) flooding-based, sending a large amount of packets to the victim and exhausting its resources; b) vulnerability-based, sending a specially crafted message to the attacked host and thus forcing it to deny or make unavailable the offered service.

Regarding the available tools that can be used to implement and launch attacks, BackTrack Linux is a very popular operating system created by Offensive Security organization that offers various security related tools. Another available option is the solution called Kali Linux that provides a larger arsenal of tools, being the most advanced penetration-testing environment nowadays [7].

Amongst the multitude of attack and penetrations tools available in Kali Linux, Metasploit Framework is one that should be mentioned herein. It allows the execution of exploit code against a remote target machine, thus enabling users to launch a multitude of attacks. Another software tool, designed for database penetration testing is called SQLMap. It offers a powerful SQL Injection detection engine, database fingerprinting and data fetching features. For performing MitM (Man-in-the-Middle) attacks, Kali distribution provides applications such as Arpspoof and Ettercap. Moreover, suppose network exploitation software is needed in a certain scenario to take advantage of weaknesses in a large variety of network protocols (e.g. DHCP Dynamic Host Configuration Protocol). Thus Yersinia is the perfect candidate to be used. The Browser exploitation Framework offered by Kali Linux is a penetration-testing tool that uses XSS and CSRF for launching attacks to various targets online, thus verifying their level of security.

*Manuscript received September 17, 2014; revised December 2, 2014*

The rest of the paper is organized as follows: Chapter II presents an overview of the analyzed attacks followed in Chapter III by the experimental results performed in the testbed, together with the methods that to mitigate the discovered vulnerabilities. The last chapter concludes the work.

## II. AN OVERVIEW OF THE TESTED ATTACKS

Web application security does not refer only to secure coding of the application itself, but also to the safety of the architectural components implied. Therefore, the security analysis should be performed on each of the components included in the scenario. In *Figure 1* we depicted a generic architecture that can be found behind any deployment that includes a web-based data accessing application. The main components are the following: the client device (i.e. tablet, PC, laptop), the router that connects the client device to the Internet and the Server side applications (i.e. the web and database services).

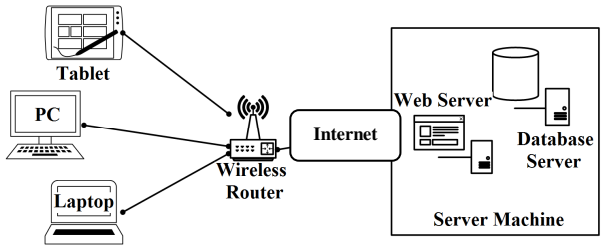


*Figure 1. Architectural components of a web-based application*

| Attack | Target Component | Affected Entities | OSI Layer |
|---|---|---|---|
| SQL Injection | DBMS | DBMS, iPad | Application |
| Man in the Middle | Wireless router, iPad | LAN, iPad | Data Link |
| SYN Flooding | Server machine | Server machine, iPad | Transport |
| DHCP Starvation | Wireless Router | Wireless Router, iPad | Application |
| Zero Window Connection | Server machine | Server machine, iPad | Transport |
| Slow HTTP Headers | Web Server | Server machine, iPad | Application |
| IPv6 Router Advertise-ment Flood | iPad | iPad | Network Data Link |
| Session Hijacking | iPad | iPad | Application, Data Link |
| XSS | iPad, web server | iPad, web server | Application |
| Cross-Site Request Forgery | iPad | iPad | Application |
| Heartbleed | Server machine | Server machine | Application |

*Table 1. The list of the eleven types of attacks evaluated*

*1. SQL Injection*

Most web applications nowadays are using SQL databases to store confidential information like usernames and passwords. The main problem of such solutions is that they are vulnerable to SQL injection which is a type of attack that exploits the user input vulnerabilities of the software interface between a web application and the database server [3]. The attack relies on the fact that the application accepts malicious SQL queries that will be parsed and executed. These vulnerabilities occur when the user's inputs are not checked for string literal escape characters embedded in SQL statements, before being sent to the database server [8]. Thus, the malicious SQL commands are injected into the user input variables and sent to the database management system through GET, POST, cookie parameters of an HTTP request or via HTTP User-Agent request headers. This is possible due to improper or unsecure coding and development of web applications. In conclusion, although the architectural component that is affected by an SQL injection attack is the database server (see *Figure 1*), the injection vulnerability is a flaw in the web application code: the database is attacked through the use of the web application's vulnerabilities. The SQL Injection attacks can be classified into the following categories:

a) *Tautologies:* The scope is to bypass authentication, identify injectable parameters or extract valuable data. This type of attack supposes to make the WHERE clause always evaluate to true, which will result in bypassing all the conditions inside the SQL statement and thus retrieving all the rows in the targeted database table.
b) *Illegal/ Logically Incorrect Queries:* The aim is to collect information about the schematic and structure of tables and fields inside the database. The attack is carried out by injecting wrong or incorrect SQL queries into the web application with the purpose of receiving information from the database in the form of an internal database error message. This is possible mainly due to incorrect handling of errors at the database level.
c) *UNION Queries:* The goal is to have access to data from other tables of the database, by joining two independent queries. This is achieved by the use of the UNION operator in order to inject a second SELECT query inside the original one.
d) *Piggy-Backed Queries:* The purpose is to make the database receive and execute multiple distinct malicious queries after a legitimate one. The query delimiter ";" is actually the vulnerability exploited. The semicolon is used to mark the end of a query and the start of the new one, allowing attackers to append malicious code to the original and legitimate one.

*2. Man-in-the-Middle*

This is a way of actively eavesdropping on the traffic being exchanged during a communication session between two entities. It intends to establish independent connections with the attacked entities and to relay the traffic exchanged between them [2], as shown in *Figure 2*.

Within a Local Area Network, the IP (Internet Protocol) address of the destination device is resolved into a MAC (Medium Access Control) address by using the ARP (Address Resolution Protocol) protocol. The resulting bindings are stored in the ARP cache, updated at regular time intervals via the exchange of two messages: ARP Request and ARP Reply.

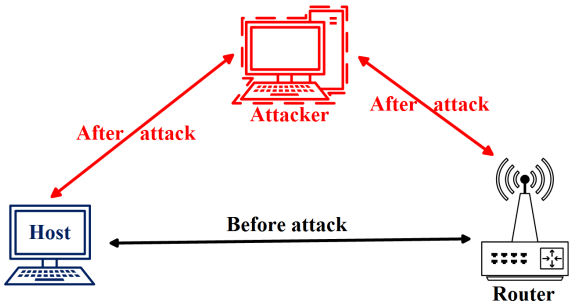


*Figure 2. The Man-in-the-Middle attack*

The ARP Request message is used to query the MAC address that corresponds to a specific IP address. The ARP Reply messages contain the MAC to IP translation information. Being a stateless protocol, ARP enables an attacker to send fake and unsolicited ARP Reply messages by spoofing its real identity. An illustration of the attack is presented in *Figure 3*. The Attacker makes unsolicited replies to the Host and Router telling the Host that the Router has the Attackers MAC, while informing the Router that the Host has the Attackers MAC. This process is called ARP poisoning and the hacker's machine becomes an intermediary point between the host and the router. He/she is able to sniff the entire traffic exchanged between the two entities. The impact of a successful MitM on these components is high due to the fact that the attacker is afterwards able to launch other attacks such as DNS (Domain Name System) spoofing, session hijacking, DoS attacks etc.

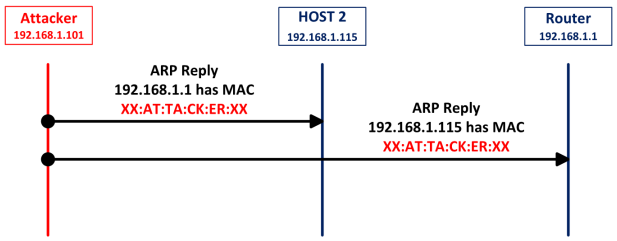


*Figure 3. Attacker sends unsolicited ARP Replies*

*3. SYN Flooding*

Denial of Service is a big category of attacks due to the fact that includes multiple approaches, all having the same purpose of denying services to legitimate users [2]. There are two groups of DoS attacks: a) single source against single target; and b) multiple sources against single target, also known as DDoS [9]. In this paper we are focused only on assessing the impact of the single source against single target type.

In TCP SYN flooding a large number of network packets are being sent to the web server with the purpose of making a service unavailable. It is based on exploiting the exchange of messages between the clients and the server, performed at every connection establishment, called TCP three-way handshake. The attack exploits the half-open state of the server when it is waiting for the client's ACK packet. In this state, the server allocates resources to store the information about the half-opened connection and these resources will only be released upon arrival of the client's ACK packet or at connection timeout. If the connection requests are performed at a much higher rate than the rate at which the half-opened connections expire, the target server will no longer be able to accept incoming connections and thereby will not be able to provide any service to legitimate users.

*4. DHCP Starvation*

This is another type of DoS attack but it does not require a very large number of packets to be injected into the network. This technique exploits the vulnerabilities of the Dynamic Host Configuration Protocol, specifically the way the DHCP Server dynamically allocates IP addresses and host configuration parameters to user devices [10]. The hacker sends a number of DHCP DISCOVER messages using spoofed MAC Addresses to the DHCP Server until the number of available IP addresses in the server pool is exhausted. Therefore, the DHCP Server will no longer be able to assign IP addresses to legitimate users thus causing a Denial of Service [11].

*5. Zero Window Connection*

This security break aims to cause a web server to enter a DoS state by establishing as many open TCP connections as it can. It is similar to SYN Flooding in the sense that it is an asymmetric resource consumption technique. Because it uses raw sockets, no resources are consumed at the attacker's side. Unlike SYN Flooding attack, the Zero Window Connection completes the TCP three-way handshake by sending an ACK packet to the server, advertising a Zero Window size. The targeted victim interprets this as a situation where the client is not currently able to accept data. Thereby, the server will allocate resources for the opened connections and the sockets could remain opened indefinitely. If the hacker makes enough connections so that all of the resources of the server are used, the attack is successful and the server will enter a Denial of Service state, not being able to accept other connection request from legitimate users [12].

*6. Slow HTTP Headers*

This Application Layer attack exploits the vulnerabilities of the HTTP (HyperText Transfer Protocol) servers. The goal is to make a web server unreachable to legitimate users by exhausting all available connections on the target device. The intruder sends incomplete/pending requests to the target to saturate its resources [6]. When sending the HTTP request message, the header must end with the carriage return character followed by the line feed one: "<CR><LF>" or "\r\n". If the client does not send the final line, the server will wait for a timeout interval before closing the connection. This timeout will restart if the client sends a new incomplete HTTP message (see *Figure 4*).

*7. IPv6 Router Advertisement Flood*

This DoS attack forces machines on the same IPv6 enabled network, to use all of their computational resources, thus becoming unresponsive. It exploits the lack of security in the exchange of Router Solicitation/Advertisement messages, whenever the nodes are trying to configure their own IP address. The attacker device floods the network with Router Advertisement messages containing random source MAC addresses and IPv6 prefixes. This action causes all devices in the network to compute a new IPv6 suffix and to update their routing table to reflect the accepted announcement. Thus the victim's CPU becomes 100% occupied for legitimate users.

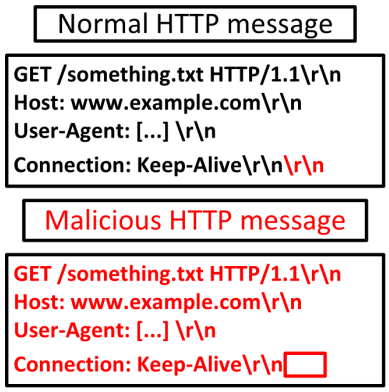


*Figure 4. Legitimate vs. Malicious HTTP message*

*8. Session Hijacking*

The attacker attempts to impersonate himself/herself as a legitimate user. It consists in the exploitation of the control mechanism of web sessions which is managed for a session token or cookie. HTTP communications between a client and a web server may use many TCP connections and thus, the web server needs to recognize every user's connection. This is performed by the use of a session token or cookie which is sent to the client, by the web server after the client successfully authenticates itself.

*9. XSS*

This is a special case of code injection that has the purpose of compromising private information, stealing cookies, creating requests that can be interpreted as those of a valid user, executing malicious code on the web browser of the targeted users etc. The attack consists in the exploitation of the vulnerabilities that lie in the code of the web application, allowing the hacker to send malicious content from the end-user's browser and collect data from the victim [2].

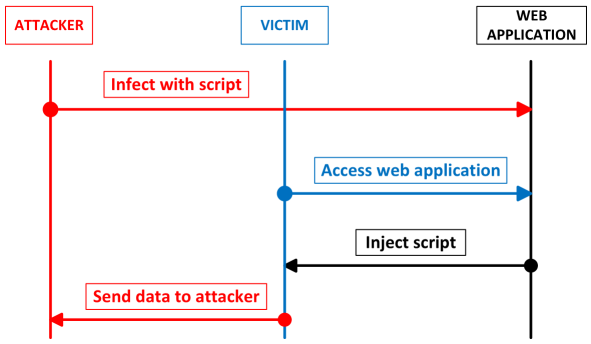


*Figure 5. Typical XSS (Cross Site Scripting) Attack*

As presented in *Figure 5*, infecting a legitimate web application with a malicious client-side script is the action performed by the basic XSS attack. When a user accesses the web application, the script is injected and downloaded into the client's browser and executed.

*10. CSRF (Cross-Site Request Forgery)*

This affected entities such as web users. An intruder causes a logged-on victim's browser to perform an unwanted action on a trusted website [5].

*11.Heartbleed*

This attack recently discovered by Neel Mehta of Google Security and it is a coding flaw in the implementation of the SSL/TLS protocols in the popular OpenSSL software [14]. It enables attackers to read the RAM memory of the systems that use vulnerable versions of these applications. According to *Figure 6*, the malicious Heartbeat request contains a very small payload (e.g. one byte) and the payload size field is being set to 65,536 bytes. Therefore the attacker tells the server that it will send 64 KB whilst it only sends one. The server will copy that byte received from the attacker into its memory and it will send back to the attacker 65,536 bytes from the memory, in the Heartbeat Response message. As a consequence, the attacker can steal confidential and sensitive information from the server's memory such as: secret encryption keys, usernames, and passwords.

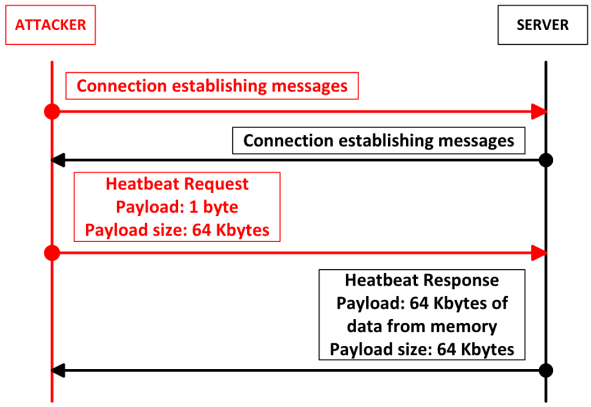


*Figure 6. Heartbleed attack message exchange*

## III. EXPERIMENTAL RESULTS

The testbed architecture used in our work in order to deploy the series of attacks is depicted in *Figure 7*. The tablet PC is the client machine, the Linksys router acts as Internet/LAN access device, whilst a Red Hat Enterprise Server on which Apache Tomcat and MySQL services are running is the destination. A third computer (i.e. Attacker) is being used to launch attacks against all the components of the considered architecture.

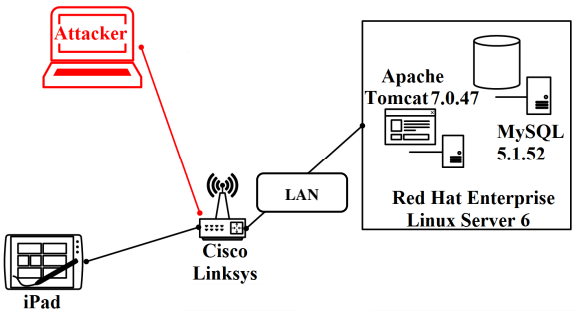


*Figure 7. Architecture components of the analyzed web-based system*

Using the Sqlmap tool, we managed to inject a "WHERE" or "HAVING" clause in the HTTP GET request URL and perform malicious queries to the database. For instance, the version of the database management system (DBMS) installed on the server together with a list of

available databases were extracted after the SQL injection attacks was performed (see *Figure 8*).

```
---
back-end DBMS: MySQL >= 5.0.0
available databases [13]:
[*] foo
[*] frequentis
[*] FrqMobServer
[*] information_schema
[*] mysql
[*] phpmyadmin
[*] pizza
[*] pma
[*] serjonu
[*] test
[*] testing
[*] webserver
[*] zabbix
```

*Figure 8. Back-end DBMS, databases discovered*

Moreover, we managed to display all the tables within the `mysql` database and from the user table we obtained the password hashes and the assigned privileges attached to each user account. The impact of the SQL Injection attack on the database server is severe. Thus the hacker is able to directly connect to the database and to steal, to add, to alter or to remove information. To defend it, the web application must filter out all characters with special meaning in SQL and it must use parameterized statements where the variable parts of the SQL are replaced by a marker instead of the user input.

When deploying the Man-in-the-Middle into the tested environment, the intruder's device sends unsolicited ARP Replies to both the Linksys Router and the client (i.e. iPad). This way it impersonates the tablet when communicating with the router, and the Linksys router when communicating with the iPad tablet. As a consequence, the attacker gains Man-in-the-Middle position, being actually a relay for the traffic exchanged between the two legitimate entities. In the second step, using a sniffing tool (e.g. Ettercap, Wireshark) the attacker captures all the traffic exchanged in the network between the two victims, thus being able to extract valuable information.

When executing a SYN Flooding attack on the testbed presented in *Figure 7*, we managed to send a number of 567,922 packets, 174 bytes each, in a time interval of 98 seconds, with an average rate of 5,792.85 packets/ second (see *Figure 9*). The impact of the attack on the web server is a major one, requiring less than 60 seconds to cause the DoS state and making it very hard to track the origin of the attack due to the randomly spoofed source IP addresses.

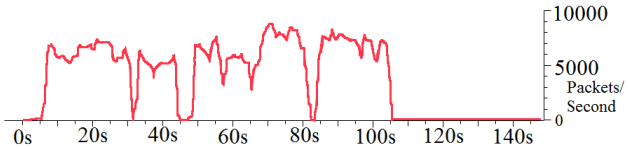


*Figure 9. The TCP output transfer rate when an attack is deployed with the hping3 tool.*

Nevertheless, after the attack was stopped, the web server recovered in a matter of seconds, users being able to access its services again. There are a number of detection and defense solutions to this attack amongst which we can mention: the use of load balancing, sending SYN cookies, using Firewalls and recycling oldest half-open connections.

When performing the DHCP starvation attack towards the Linksys router, we successfully sent 43,724 DHCP Discover packets to the DHCP server on the router, in 4.5 seconds, with an average rate of 9,740 packets/second. The DHCP server responded to the malicious DISCOVER packets with 40 DHCP OFFER messages, reserving 40 IP addresses (all addresses available in the IP pool) for the attackers spoofed MAC addresses. The response messages were sent in 40 seconds, therefore offering the attacker IP addresses and network configuration parameters with a rate of 1 packet/second as represented in *Figure 10*. The attack was successful because we managed to exhaust all of the IP addresses that were still available in the server pool (in the current configuration the DHCP server was set to have a number of 50 available addresses and 10 of them being already reserved or occupied by other devices in the network).

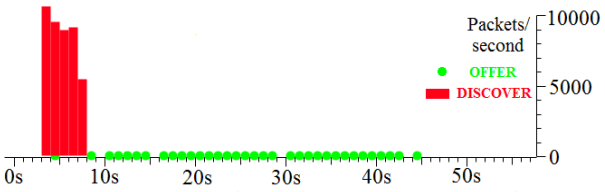


*Figure 10. The DHCP starvation attack*

To defend against such attacks, a solution would be to set a threshold on the number of DHCP Request messages processed in a certain period of time. Just specifying at the DHCP server side a set of Layer 2 addresses that can dynamically get an IP address is not enough because the attacker could spoof a trusted MAC. When performing the Zero Window Connection security break, the Sockstress tool managed to establish 135,500 connections with the server, at a rate of 141.44 connections/ second. As a consequence we observed that the cached memory increases dramatically and remains occupied even when the attack is terminated. Actually, during the security break the level of RAM memory used increased dramatically and when the attack was stopped, only the level of cached memory occupied remained high. From the user's perspective, during the attack, the server did not respond to any legitimate requests, regardless of the service that tried to be accessed. A practical defense mechanism against this is to limit the rate of connections to the server by blocking an IP address after it opens more than 10 connections to a certain port in a time interval of 20 seconds.

Another attack that successfully caused a DoS situation in our testbed was the Slow HTTP Headers. It consisted of 17,786 HTTP requests sent to the web server in a time interval of 12 minutes with a low average rate of 23 packets/ second (see *Figure 11*).

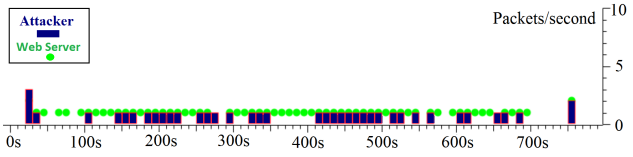


*Figure 11. Messages exchanged between the attacker and the web server when performing Slow HTTP Headers*

The attack was successful in exhausting all of the available parallel connections supported by the web server. After that, it recovered from the DoS state in approximately one minute (i.e. the time needed for the server to close all of the established connections with the attacker). To defend against this, the hardware load balancers accepting only full HTTP connections should be used. Moreover, by limiting the allowed number of connections for a single host and specifically for the Apache HTTP server installing the `mod_antiloris` module should be used.

When performing the XSS attack, we managed to 'hook' the web browser of the victim with the Fiddler AutoResponder feature. The hook.js script was sent to the client from the web server. With the use of the command "Get Cookie" from the BeEF tool, the 'hooked' browser sent to the attacker's web server the cookies of the web application. Afterwards, using "Webcam HTML5", the BeEF tool sent a request to the attacked browser to use the webcam. When the request was accepted, the webcam was activated and the victim started to send real time video stream to the attacker. The CSRF attack on the Cisco Linksys Router E900 managed to send the victim's browser a popup message requesting to authenticate to the routers web-based administration page. If the target client types in merely the username, the attack will succeed and the password will be changed to the one specified by the attacker. A possible solution to defend the web application and its users against this is to encrypt the HTTP requests by using SSL/TLS.

Unfortunately the Heartbleed attack was unsuccessful, no matter the protocol type sent by the `openssl_heartbleed` module. The reason the server did not respond to the Heartbeat Request with a Heartbeat Response was related to the version of OpenSSL installed on the server (i.e. an older one that did not have the Heartbeat extension). Although the server was not vulnerable to the Heartbleed security break due to its older version of the protocol, this is not a solution because if no updates are performed, it could be vulnerable to other attacks that are patched in the newer versions. Also IPv6 Router Advertisement Flood did not affect iPad, as the vulnerability has been patched in the iOS 7 version of its operating system. In a similar manner, the performances of Windows 7/Windows 8-based machines are not influenced by this attack.

## IV. CONCLUSIONS

The security analysis presented herein shows that the architectural components of the tested web-based system are all vulnerable to a series of attacks if no security defense methods and solutions are being employed. The SQL Injection attack performed on the database management systems had the most severe impact compared to all others. It managed to extract the table of database users, therefore allowing an attacker to further gain control over the DBMS and to compromise the confidential data within it. Two Man-in-the-Middle attacks had as a target the wireless router. ARP poisoning achieved the goal of sniffing all of the traffic to/from the router, therefore being able to intercept any data that was not encrypted. DHCP Starvation attack managed to deny access to any new hosts that were willing to connect to the router. Overall, we had nine successful attacks and two unsuccessful ones (IPv6 Router Advertisement Flood, Heartbleed) due to patched software versions of different products. The vulnerabilities that rely on the coding flaws of the web application were exploited by four of the implemented attacks, while the vulnerabilities that rely in the operation of protocols were exploited by six of the implemented attacks. A vulnerability that consists in erroneously implementation of the TLS protocol was also verified. Some defense mechanisms to solve the vulnerabilities were discussed: updating and patching software systems, installing hardware load balancers, and configuring the system's security features. The employment of SSL/TLS over HTTP to encrypt the transmitted data proved to be the solution to mitigate four of the nine successful attacks.

## REFERENCES

[1] N. Teodoro, C. Serrao, "Web Application Security: Improving Critical Web-Based Applications Quality through In-Depth Security Analysis", *Proceedings of the International Conference on Information Society (i-Society)*, 27-29 June 2011, pp.457-462, 2011

[2] *Top 10 2013*, OWASP Foundation, 2013 [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10 [Accessed: August 2014].

[3] W. Du, "SQL Injection Attack Lab", *National Science Foundation's Course, Curriculum, and Laboratory Improvement* (CCLI), 2014, [Online]. Available: http://www.cis.syr.edu/~wedu/seed/Labs/Attacks_SQL_Injection/SQL_Injection.pdf/, [Accessed: August 2014].

[4] V.K. Malviya, S. Saurav, A. Gupta, "On Security Issues in Web Applications through Cross Site Scripting (XSS)", *Proceedings of the 20th Software Engineering Conference APSEC 2013*, 2-5 Dec. 2013, Asia-Pacific, Vol. 1, pp.583-588

[5] L. Xiaoli, P. Zavarsky, R. Ruhl, D. Lindskog, "Threat Modeling for CSRF Attacks", *Proceedings of the International Conference on Computational Science and Engineering, CSE 2009,* 29-31 Aug. 2009, Vol. 3, pp.486-491, DOI: 10.1109/CSE.2009.372

[6] E. Cambiaso, G. Papaleo, M. Aiello, "Taxonomy of Slow DoS Attacks to Web Applications", *Communications in Computer and Information Science*, Volume 335, 2012, pp 195-204

[7] Kali Linux Documentation, *Offensive Security Ltd.* 2014 [Online]. Available: http://docs.kali.org/, [Accessed: September 2014].

[8] A. V. Uskov, "Hands-On Teaching of Software and Web Applications Security", *Proceedings of the Interdisciplinary Engineering Design Education Conference IEDEC 2013*, 4-5 March 2013, vol.3, pp.71-78

[9] A. Bajpai, "Securing Apache, Part 8: DoS & DDoS Attacks", *The OpenSourceForU,* April 2011, [Online]. Available: http://www.opensourceforu.com/2011/04/securing-apache-part-8-dos-ddos-attacks/, [Accessed: September 2014]

[10] "DHCP Starvation", Hakipedia, December 2008, [Online]. Available: http://hakipedia.com/index.php/DHCP_Starvation, [Accessed: September 2014]

[11] M. Yaibuates, R. Chaisricharoen, "ICMP based Malicious Attack Identification Method for DHCP", *Proceedings of the 4th Joint International Conference on Information and Communication Technology, Electronic and Electrical Engineering JICTEE 2014*, 5-8 March 2014, pp.1-5

[12] "Sockstress DDoS Attack: TCP-Connection Clogger", July 2013, [Online]. Available: http://www.ddosattacks.biz/attacks/sockstress-ddos-attack-tcp-connection-clogger/, [Accessed: July 2014]

[13] S. Kapoor, *Session Hijacking Exploiting TCP, UDP and HTTP Sessions*, 2014, [Online]. Available: http://www.infosecwriters.com/text_resources/pdf/SKapoor_SessionHijacking.pdf , [Accessed: September 2014]

[14] N. Metha, *OpenSSL Security Advisory*, 2014, [Online]. Available: https://www.openssl.org/news/secadv_20140407.txt