**Firmware Analysis**

This blog aims to offer a thorough guide on firmware analysis focusing on static methods. We'll use "IoTGoat" from the OWASP Foundation as our vulnerable firmware sample. We've made this choice to sidestep the security and legal issues that come with exploiting a production sample. Even so, all the techniques and exploitation methods we'll show you are true to life and can be applied to real-world samples.

IoTGoat works as a security training tool similar to WebGoat for web security practice. This Internet of Things (IoT) device has built-in weaknesses on purpose. It gives security experts, coders, and researchers a safe place to learn about and test common IoT flaws. By copying real-world problems like weak login systems, unsafe communication, and poor access controls, IoTGoat helps users understand and fix these security issues. You can set it up in a virtual space offering a hands-on platform to train, check security, and create IoT-specific security tools and methods. Using IoTGoat well lets users learn how to secure IoT devices and apps, which matters a lot now that we have more connected devices. If you're into IoT security, working with IoTGoat gives you a chance to boost and sharpen your skills.

To analyze firmware, you need to set up the right environment. This guide recommends AttifyOS, a Linux system built to exploit embedded devices. Setting up IoTGoat for static analysis also needs some extras, like the static image file and the provided virtual machine. Here's a list of everything you need to download.

**Remember:** *Download the IoTGoat image file inside your AttifyOS machine, but get the IoTGoat virtual machine on your main computer.*

**Required Downloads:**

IotGoat Image File <https://github.com/OWASP/IoTGoat/releases/download/v1.0/IoTGoat-raspberry-pi2.img>

IotGoat Virtual Machine- <https://github.com/OWASP/IoTGoat/releases/download/v1.0/IoTGoat-x86.vdi>

AttifyOS- <https://github.com/adi0x90/attifyos>

**$ sudo apt install john sqlite3 arp-scan**

**$ cd tools/ && git clone https://github.com/craigz28/firmwalker.git**

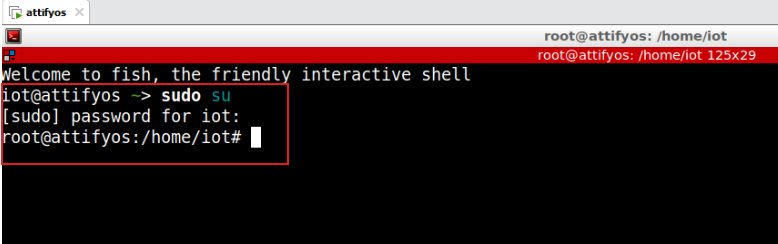
**OPEN THE TERMINAL:**

Once you're in your AttifyOS virtual machine (the root password is "attify"), open a terminal and type these commands

**Step 1:**  Open the terminal and write

$sudo su

Attify

****

**Static Analysis**

Emulating firmware samples often presents difficulties. This document will examine static analysis using dynamic analysis to check findings.

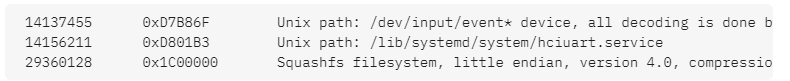
To kick off the analysis, we'll use Binwalk, a robust toolkit for extracting and analyzing firmware. Start by running this command to pull out the file system from the image file:

Binwalk -e imagefile.img

**Step 2:** $ binwalk -e IoTGoat-raspberry-pi2.img

****

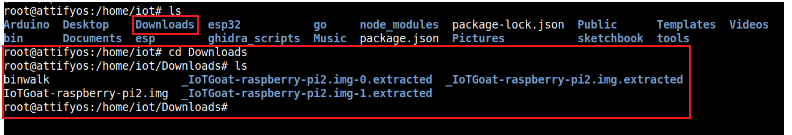
When you run this command, Binwalk will attempt to extract the filesystem and display important information, such as the following:

****

**Step 3**: To check the file,

$ ls

$ Downloads

****

This output makes it evident that we are dealing with a Linux-based firmware sample. Most firmware samples are indeed Linux-based, but this particular one employs the SquashFS compressed filesystem. After you extract the filesystem, the first thing to do is to look for hardcoded user credentials. They occur frequently & are easy to find. To start, head to the “etc directory. That's where you’ll find both the “passwd” and “shadow” files.

**Step 3: U**se the IoTGoat-raspberry-pi2.img-0.extracted

Open it and find a directory name squashfs-root.

$ cd IoTGoat-raspberry-pi2.img-0.extracted

$ ls

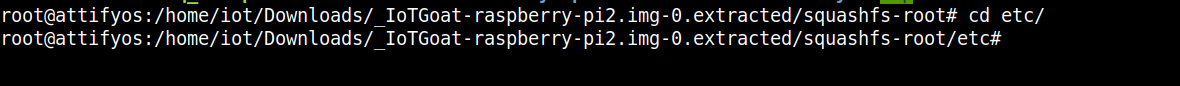


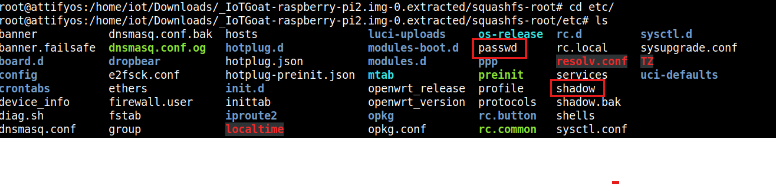
**Step 4:**  $ cd squashfs-root

$ ls

****

**Step 5:** $ cd etc/ - where both the “passwd” and “shadow” files are present**.**

****

****

**Step 6:** To combine the information from the "shadow" and "passwd" files, you can use a password cracking tool "John the Ripper" which performs brute-force on password hashes.

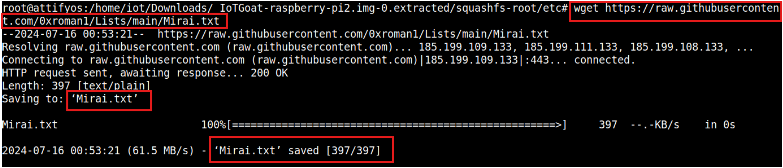
$ unshadow passwd shadow > iotgoat.db

****

**Step 7:** Once the file is setup, now you have to perform a brute-force attack. For this attack, you need a dictionary that contains a simple text file containing words to test against the user's hashes. In this case, the wordlist from the Mirai botnet will work just fine. Go ahead & download it from

here: <https://raw.githubusercontent.com/0xroman1/Lists/main/Mirai.txt>

$ wget <https://raw.githubusercontent.com/0xroman1/Lists/main/Mirai.txt>

****

After you've got the text file downloaded, you can begin the brute-force process by running this command:

**Step 8:** john iotgoat.db --wordlist=**textfilename.txt**

$ john iotgoat.db –wordlist=Mirai.txt

From what we see now, it's clear that the **"iotgoatuser**" has a password of "**7ujMko0vizxv**." Although we’ve found some basic user credentials, there are still many more significant findings waiting within the filesystem before we move on to exploiting these credentials.

****

Next up is using our previously installed "firmwalker" tool. Open the directory where you have this utility stored & run the script while providing it with the path to your extracted filesystem.

****

****

Now, in your terminal, navigate to that database path mentioned earlier. This is a sqlite3 database; all you have to do next is run these commands:

**sqlite3 sensordata.db**

**.tables**

**SELECT \* FROM sensors;**

When you do this, you’ll see the result:

****

****

The contents of this file are below:

$ cat **iotgoat.lua**

****

****

**Conclusion:**

In conclusion, the firmware analysis of the IoTGoat practical brings light critical insights regarding vulnerabilities found in Io devices. By carefullying and examining the firmware, we discovered possible weaknesses. These include hardcoded credentials, unencrypted data storage, and insecure update mechanisms that could be exploited by malicious actors.