

Write down the answers and demonstrate to teacher when you have finished the lab.

In some questions, reading/searching Netacad chapter’s will also help.

Lab – Compromise IoT Device Firmware

1. Objectives

Part 1: Performing Threat Modeling Activities to Evaluate IoT Device Firmware

Part 2: Reflection and Discussion of Threats to IoT Device Firmware

1. Background / Scenario

IoT devices are susceptible to attacks like many other Internet connected devices running an operating system. In the past, hackers have taken over smart home devices including security cameras, refrigerators, ovens, air conditioners, and even vehicles while they were driving. IoT devices with cameras can potentially be used as spying devices within one’s home. Many popular IoT devices run Linux as their OS.

Vulnerabilities with device firmware include, but are not limited to: key handling vulnerabilities, hard coded passwords, default username/passwords, clear text password transmission through WiFi, buffer overflow attacks, distributed denial of service attacks, and firmware modification attacks.

This lab will require students to download a device firmware binary file and decompress/decode it to its file system and kernel components. Students will be required to analyze the firmware and investigate how vulnerabilities can be exploited.

In this particular lab, we will be exploring an IoT device’s firmware file by using a wide variety of tools/commands on a Kali Linux virtual image.

1. Required Resources

* Laptop or PC running Oracle Virtual Box with a virtual image of Kali Linux loaded
* Internet connection

1. Performing Threat Modeling Activities to Evaluate IoT Device Firmware
   * 1. Set up the environment and open a terminal window.
2. Start the IoTSec Kali VM and log in using username **kali** and password **kali**.

**Note**: If the Kali Linux machine has not been installed, please refer to a previous lab in this course.

1. Open up a terminal window.
   * 1. Crack root password using john.

In this scenario you have managed to extract a copy of Linux IoT device root file system. You have the rootfs contents as file “rootfs.ext2”

* + - 1. Issue the **strings** command on the rootfs.ext2 file system and search for “root” to see if there are any password entries. This file is essentially an image file that could be written directly to a drive and booted up using the appropriate hardware. We will look for “root:$” which is how a typical entry would appear in the shadow file, and then save it to a temporary file in the /tmp directory. However, when we filter for “root:$” using the ‘grep’ command, we have to delimit the “$” with two backslashes (\\) because the $ has a special meaning in the command.

**Note**: The Linux prompts in the examples of this lab have been shortened to just $, indicating that you have user kali access right. **You need to have installed the lab-support-files as described in previous lab**.

$ **pwd**

$ cd /home/kali/lab\_support\_files/emulated/mips32

$ strings rootfs.ext2 | grep root:\\$ > /tmp/passwd

$ more /tmp/passwd

* + - 1. Run john on the file **/tmp/passwd** to crack the password.

$ **john /tmp/passwd**

Created directory: /root/.john

Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"

Use the "--format=aix-smd5" option to force loading these as that type instead

<some output omitted>

Use the "--show" option to display all of the cracked passwords reliably

Session completed

You could also use the **show** option to display the cracked passwords.

$ john --show /tmp/passwd

The output is in **shadow file format**. (you can google that). It is the way how linux users and their passwords are stored in linux. See kali users in vm using ‘sudo cat /etc/shadow’ . The root admin can see hashes of all other users, but not their plaintext passwords.

What is the root password? device.

* + 1. Extract the IoT device firmware binary file into a new directory.

In this scenario, you have extracted a compressed image of a filesystem for study. Typically embedded devices have a read-only compressed filesystem on persistent flash memory, and in system boot the filesystem is decompressed in RAM, from which it is run. RAM access times are much faster than flash memory access by factor 10-100, so it makes sense to use it for runtime. When the system is powered off, RAM contents are erased, but flash contents persist.

* + - 1. Change directories to the IoT device firmware file:

$ cd /home/kali/lab\_support\_files/firmware

* + - 1. List the content of the folder and obtain the name of the firmware file:

$ ls -l

* + - 1. The following command will extract the file system from the firmware and will allow you to navigate and explore anything that can be targeted as a potential vulnerability  
         Note newest binwalk requires a fix <https://gist.github.com/thanoskoutr/4ea24a443879aa7fc04e075ceba6f689>

$ binwalk -e iotdev\_firmware.bin

What is the file system type? Squashfs filesystem.

* + 1. Examine the output of the binwalk command.
       1. What is the significance of this file system type? Do a web search to find more information.

R: It’s a read only compressed file system usually used in embedded systems (routes, etc.).

* + - 1. Verify what new subdirectory was added:

$ ls -l

What was the name of the new directory that was added?

R: \_iotdev\_firmware.bin.extracted

* + - 1. Change into this new directory:

$ cd \_iotdev\_firmware.bin.extracted

What is the name of the new directory within this subdirectory? What command did you use to determine that?

R: Using the command ls -la, we get 2 new directories, “squashfs-root” and “squashfs-root-0“.

* + 1. Navigate and explore this extracted file system
       1. Verify the directory you are in and list the content of the directory:

$ pwd

$ ls -l

* + - 1. Move into the new squashfs-root directory:

$ cd squashfs-root

What are some of the subdirectories in this directory? What command did you use?

R: Using ls, we can see, for example, “bin”, “usr”, “sys” among other. So it seems like it is based of unix.

* + - 1. The **readelf** command can be used to determine what architecture a binary file was originally compiled to be run on. Use **readelf** to determine what architecture the chkntfs binary file (located in sbin) is supposed to be run on. This command will help us determine which qemu binary to run in the next step.

$ readelf -h sbin/route

What is the Machine architecture listed?

R: MIPS R3000.

* + 1. Use the QEMU open source machine emulator and virtualizer

QEMU will allow you to emulate binaries for other architectures different from what your Kali Linux image is currently running on (Intel/AMD 64). QEMU supports the MIPS architecture.

All of the binaries in the squashfs-root directory that were decompiled/decompressed from the IoT device’s firmware file were compiled for the MIPS architecture. They will not run on this Kali Virtual Image because it is on a different architecture. In order to run these binaries, we will need to use QEMU.

$ **sudo apt update**

$ **sudo apt install qemu-user-static**

* + - 1. Check that the binaries really are for MIPS architecture

$ file bin/busybox

* + - 1. The following uses the **chroot** command. The **chroot** command changes the root directory to the current directory as identified with the **.** (dot)

Enter the commands below to demonstrate that we can run MIPS binaries on the Kali Linux OS:

$ sudo chroot . bin/busybox date

$ sudo chroot . bin/ping -c 3 203.0.113.1  
(note that copy-pasting the commands can change ‘–’ character to something else -> help is displayed)

* + - 1. Start a new shell within the squashfs-root directory with the following command:

$ sudo chroot . bin/sh

How did the prompt change? It only displays “/ # ” instead of my kali hostname.

* + - 1. Verify that new shell is using the current path as the root directory.

# pwd

What is the current path? We are in the root directory, “ / ”.

Look at other programs/binaries in the bin and sbin directories. Try running at least 3 other programs and document their results. Type ‘exit’ when you have completed exploration to leave the shell.

R: “ls” which lists the current directory, “pwd” which prints the working directory and “ipaddr” which shows information about the machine connection to the internet.

* + 1. Explore and look at other important files on this decompiled/extracted file system
       1. Verify you are currently within the squashfs-root directory:

# pwd

* + - 1. Use the following commands to explore and look for possible vulnerabilities:

# ls -l usr

# ls -l usr/bin

# ls -l bin/

# ls -l usr/sbin

# ls –l etc/

# cat etc/passwd

# cat etc/shadow

# cat etc/ssh/ssh\_host\_rsa\_key

* + - 1. Based on previous command outputs (and using other tools introduced in this lab), explain two methods how an attacker can get root access to an another instance of this device?

R: It seems that the shadow file has the root encrypted password that we could take, crack and use to get access. Also, we have the private ssh key, which we can use to get a ssh connection to the machine. Since we are on the root account and we have the ssh, we could use it, as it would give us root access.

1. Reflection and Discussion of Threats to IoT Device Firmware

Part 1 of this lab demonstrates how to decompile/disassemble a binary firmware file and navigate the file system for vulnerabilities. Hackers are able to use widely available tools to modify the firmware with backdoors or Trojan horses and repackage the firmware into a new firmware image.

1. Reflection
   1. What kinds of threats or vulnerabilities are IoT device firmware susceptible to?

R: Firmware must be kept up to date, otherwise we might end with firmware that has vulnerabilities.

The lack of encryption can be dangerous too.  
 Insecure authentication is also something we must think about.

* 1. Describe one type of IoT device firmware attack and what measures could be used to help prevent this type of attack.

R: Weak, Guessable, or Hardcoded Passwords is when we leave the default credentials, allowing anyone to try to brute force them, getting access to our device. To prevent this, we must change all the default accounts that might come with our devices.

* 1. Describe another type of IoT device firmware attack and what measures could be used to help prevent this type of attack.
  2. R: Lack of Secure Update Mechanism is when we have lack of firmware validation on

device, lack of secure delivery and so on. We can prevent this by, for example, validate our firmware with some sort of signing method and encrypt our firmware delivery.

* 1. What is the name of the program that hackers could use to modify firmware with a backdoor and repackage it back into another firmware binary? Does this program exist on the Kali Linux operating system that you are using? Hint: Use Google to find the answer.

Firmware mod kit can be used to make changes in firmware. It can be used to place a backdoor on the system and repackage it, for example. It does not exist in our operating system.