

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 **root** and password **toor**.

**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.
        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 root access.

# **pwd**

# cd /root/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

What is the root password? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Text

Description automatically generated

* + 1. Extract the IoT device firmware binary file into a new directory.
       1. Change directories to the IoT device firmware file:

# cd /root/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:

# binwalk -e iotdev\_firmware.bin

What is the file system type? squashfs

Text

Description automatically generated

* + 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.

SquashFS is a compressed, read-only filesystem for Linux that can also be used as a flexible, general purpose, compressed archive format, optimized for fast random access with support for Unix permissions, sparse files and extended attributes.

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

# ls -l

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

\_iotdev\_firmware.bin.extracted

Text

Description automatically generated

* + - 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?

squashfs-root, ls -l

* + 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?

ls

Graphical user interface, text, application, chat or text message

Description automatically generated

* + - 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?

MIPS architecture (MIPS R30ch 00)

Text

Description automatically generated

* + 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. 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.

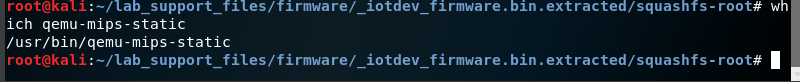
**Instructor Note**: QEMU should already be installed on the Kali Linux virtual image.

* + - 1. The QEMU binary that we will be using needs to be copied to the squash-fs-root directory. Before we copy it, we need to know where it is located:

# which qemu-mips-static

Where is this binary file located?

/usr/bin/qemu-mips-static



* + - 1. Copy this file to the current working directory which should be squash-fs-root:

# cp /usr/bin/qemu-mips-static .

* + - 1. Try running bin/busybox and other binaries without QEMU and notice that they will not be successful:

# bin/busybox

# bin/ping 127.0.0.1

These commands did not work since the QEMU emulator is required to run binaries originally compiled on different architectures.

Text

Description automatically generated

* + - 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:

# chroot . ./qemu-mips-static bin/busybox date

# chroot . ./qemu-mips-static bin/ping –c 3 127.0.0.1~

Text

Description automatically generated

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

# chroot . ./qemu-mips-static bin/sh

How did the prompt change?

It is now displaying the path as ‘/’ followed by the # for the root prompt

Graphical user interface, text

Description automatically generated

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

# pwd

What is the current path?



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.

* + 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

A screenshot of a computer

Description automatically generated with medium confidence

# ls -l usr/bin

A picture containing graphical user interface

Description automatically generated

# ls -l bin/

A picture containing graphical user interface

Description automatically generated

# ls -l usr/sbin

A picture containing graphical user interface

Description automatically generated

# ls –l etc/

Text

Description automatically generated

# cat etc/passwd

Text

Description automatically generated

# cat etc/shadow

Text

Description automatically generated

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

Text

Description automatically generated

* + - 1. What other commands did you use to explore and look for possible vulnerabilities?

mount command

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?

An IoT device can be vulnerable to attacks like Buffer Overflow attacks, DOS attacks, distributed denial of service, firmware modifications and key handling vulnerabilities.

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

IoT devices with default usernames could be logged into remotely. To fix this vulnerability usernames and passwords may be changed. Passwords can also be cracked so we should also pay attention to that.

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

Adding a backdoor to an existing script in the firmware of the device. Hackers can repackage the firmware into a new downloadable binary file, which can then be installed on an IoT device by an unsuspecting user. It is possible to verify that the firmware binary files have not been modified by using a hash.

* 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.

The name of the program is firmware-mod-kit and yes, this program exist on the Kali Linux operating system.