IoT Lab

Firmware Analysis

Objectives:

- *Analysis of a firmware binary*
    - *Filesystem extraction*
    - *Basic analysis*
    - *Execution of binaries compiled for other architectures*
- *Device simulation*
    - *Searching for things of interest*
    - *Network service discovery & access*

Notes:

This Lab is based on a Cisco Netacad IoT Security 1.1 Lab (Lab activity: 3.2.2.7). The second part is using material found in the VM provided in the corresponding course but not used in any proposed Lab by Cisco at the time of writing. The IoT device firmware and its simulation files were retrieved from the Kali VM used in the course. However, the presented story was not part of the initial Lab and is given to gamify the Lab.

The story…

You have been asked to analyze a potential attack attempt targeting a specific type of IoT device used within your company.

To support your investigation, the SOC team has provided the following files:

- The latest backup of the firmware deployed by the IoT department: **iotdev_firmware.bin**

- An archive containing various files recovered from a hacker's computer: **emulated.tgz**

Your mission is to identify the vulnerabilities present in the original firmware and determine what the attacker was planning to do.

Happy hunting…

Part I: Analysis of the deployed firmware binary

Extracting the original filesystem from the firmware

In this part, you are going to use the tool ***binwalk***.

As stated on the GitHub page of the project, "*Binwalk is a fast, easy to use tool for analyzing, reverse engineering, and extracting firmware images.*"
In practice, ***binwalk*** can scan a firmware image for many different embedded file types and file systems.

With the proper complementary system tools, it is also able to extract the files it finds, making it remarkably easy to analyze firmware images.

Analyzing the filesystem

- Unzip the *Firmware.zip* file and go to the Firmware directory
- Run ***binwalk*** (without any option) on the *iotdev_firmware.bin* file. What information did you retrieve?
- Using the *-e* option, use ***binwalk*** to extract the full filesystem from the firmware
- Locate and get into the directory where these files have been extracted

Now that you have access to the original file system, you should try to perform some basic analysis.

Here is a non-exhaustive list of "classical" things you might want to check:

1. Check the original login / password files
2. Investigate user accounts
3. Check for available remote access services
4. Check for standard available programs and configuration files
5. Try to execute and analyze the execution of some programs
6. Check for log files

Some guidelines

1. Credentials in a UNIX system are normally found inside two files: */etc/password* and */etc/shadow.* Using the system man pages, figure out the structure of those files and analyze them.
   Things you might want to do:
   - Take note of interesting usernames
   - Check for accounts with no password or with easy to crack passwords
     - Tip: To check for easy to crack passwords, you might want to use a password cracker such as ***john***
       -
2. Analyze the content of */root* and all the directories found in */home*
   - Check for unusual files
   - Check for hidden files
   - Check for access control files such as files found in the *.ssh* directories

3. Check for activated network services in */etc/init.d* and their corresponding configuration files. For example, you might want to check for standard services such as:
    - telnet
    - ssh
    - ftp
    - http services
4. Check for unusual access rights on files.

5. Check the standard bin directories *(/bin*, */sbin*, */usr/bin*, */usr/local/bin*…) and users' home directories for unusual programs.
   Note: To check, debug or even test a program found in the filesystem, you might have to execute it. However, the architecture of the device might not match the one found of your machine. In that case, you will have to use an emulator such as ***qemu***. Here is how to do it.
   As an exercise, you are going to execute the program *"/bin/busybox*"
    - Find out what this program does
    - Why are almost all binaries found in the firmware image linked to it?
    - Check the actual architecture of the binary using the *readelf* program:
        - Get into the root directory of the firmware
        - Type: *readelf -h bin/busybox*
        - Check the machine type; in your case, the architecture should be *MIPS R3000*
        - Optionally, check what happens if you try to execute the program without ***qemu***, typing: *bin/busybox*
    - Locate and copy the ***qemu*** program corresponding to the actual architecture for which *busybox* was compiled into the base directory of the firmware extraction

      In your case, you should copy: */usr/bin/qemu-mips .*
        - *Note:* depending on the Linux distribution you are using, you might have to copy and paste *qemu-mips-static* instead of *qemu-mips*.
    - Execute *busybox* in a "*chrooted*" environment
        - Type: *chroot . ./qemu-mips ./bin/busybox*
          Note: Make sure you understand the meaning of the *chroot* command and consequently, why you needed to copy the static version of ***qemu***
        - Try to execute the following command:
          chroot . ./qemu-mips ./bin/busybox sh
          What happens? Why?
        - Try to execute the shell program of the firmware:
          chroot . ./qemu-mips ./bin/sh
        - Explain the prompt and the results
6. The log files in a standard UNIX system are found in the */var/log* directory.
   You might to check its content for information.

Of course, at any step, you might dig out information forcing you to restart part of your analysis all over again.

Useful UNIX commands to help you with your investigations:

- *grep*: look for regular expressions in files
- *find*: search the filesystem for files based on different attributes (names, rights, timestamps, …)
- *awk*: display filtered information from files

Part II: IoT Device simulation

As already said, the SOC team was able to retrieve some files from the computer of a possible hacker. These files are found in the *emulated.tgz* archive. Your mission is to further analyze the IoT device simulation and:

- Figure out which network services are actually running and how they work
- Try to guess what the hacker was planning, and the risks involved

Running the simulated IoT device

- Untar the *emulated.tgz* archive
- Get into the *emulated/mips32* directory
- A prior analysis of the files by the SOC team showed that the hacker was simulating the IoT device with ***qemu*** and that he created a script named *start_device.sh* to automate the entire process. This script is creating the virtual environment, a virtual network and then starts ***qemu***.
    - Preliminary work: The version of ***qemu*** used by the hacker appears to be older than the one available in your environment. Edit the *start_device.sh* script and delete any reference ***vlan=0***
- Run the simulation and connect to the device using the admin password you found in the first part
- Analyze which services are available and how they work.
- Look into the home directory of the *default* user and find and analyze what seems to be an interesting add-on to the original image…
- Based on your investigations conclude about the motivations of the hacker
- To properly stop the simulation, log into the simulated device as *root* and type: *poweroff*

Some useful UNIX commands in the context of the investigation:

- To execute a program in the background, just add *&* at the end of the program. E.g.: *./myprog &*
- To list the running processes: *ps -aux*
- To kill a process: *kill process_id*
  The *process_id* can be retrieved with the *ps -aux* command
  Should the previous kill command be unsuccessful, you can try: *kill -KILL process_id*
- To get the IP configuration of the network interfaces: *ifconfig*
- To list the TCP network connections (active connections and listening ones): *netstat -tan*
- To connect to a remote machine through the TCP protocol: *nc*
  For example, if you want to connect to port 3333 on a remote machine with IP address 192.168.12.6, you will type: *nc 192.168.12.6 3333*