IoT Security and Privacy

Lab 2 – ESP32 Flash Hack
(70 points)

Instructions

1. This is a group assignment. Each group can have at most two students. There should be a single submission for each group.
2. Answer each question following the original question. Do NOT delete the original question.
3. Refer to Print screen on how to take a screenshot.

Questions

1. Please refer to [1] for the use of *esptool*. The following command will retrieve the partition table of the IoT kit flash in the binary format:
   a. *python C:\Users\$USER\.platformio\packages\tool-esptoolpy\esptool.py read_flash 0x8000 0xc00 ptable.img*
   where 0x8000 is the start address of the partition table and 0xc00 is the length of the partition table. $USER is the username of the user that installed platformio. The binary partition table is saved in *ptable.img*. Please provide a screenshot of the command running. [10 point]

```
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab1> cd ..
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy> cd lab2
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab2> python C:\Users\Owner\.platformio\packages\tool-esptoolpy\espt
ool.py read_flash 0x8000 0xc00 ptable.img
esptool.py v3.1
Found 3 serial ports
Serial port COM5
Connecting.....____.....
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 9c:9c:1f:ca:75:58
Uploading stub...
Running stub...
Stub running...
3072 (100 %)
3072 (100 %)
Read 3072 bytes at 0x8000 in 0.3 seconds (81.0 kbit/s)...
Hard resetting via RTS pin...
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab2>
```

   b. Please refer to [2] for the use of *gen_esp32part.py*. The following command will print out the partition table of our IoT kit in the CSV (comma-separated values) format. The partition table shows how the flash is partitioned.
   python C:\Users\$USER\.platformio\packages\framework-espidf\components\partition_table\gen_esp32part.py ptable.img
   Please provide a screenshot of the command running. [10 point]

```
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab2> python "C:\Users\Owner\.platformio\packages\framework-espidf@
~3.30101.0\components\partition_table\gen_esp32part.py" ptable.img
Parsing binary partition input...
Verifying table...
# Espressif ESP32 Partition Table
# Name, Type, SubType, Offset, Size, Flags
nvs,data,nvs,0x9000,16K,
otadata,data,ota,0xd000,8K,
phy_init,data,phy,0xf000,4K,
factory,app,factory,0x10000,1M,
ota_0,app,ota_0,0x110000,1M,
ota_1,app,ota_1,0x210000,1M,
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab2>
```

c. Refer to [2] and explain the partition table that is printed out. [10 point]

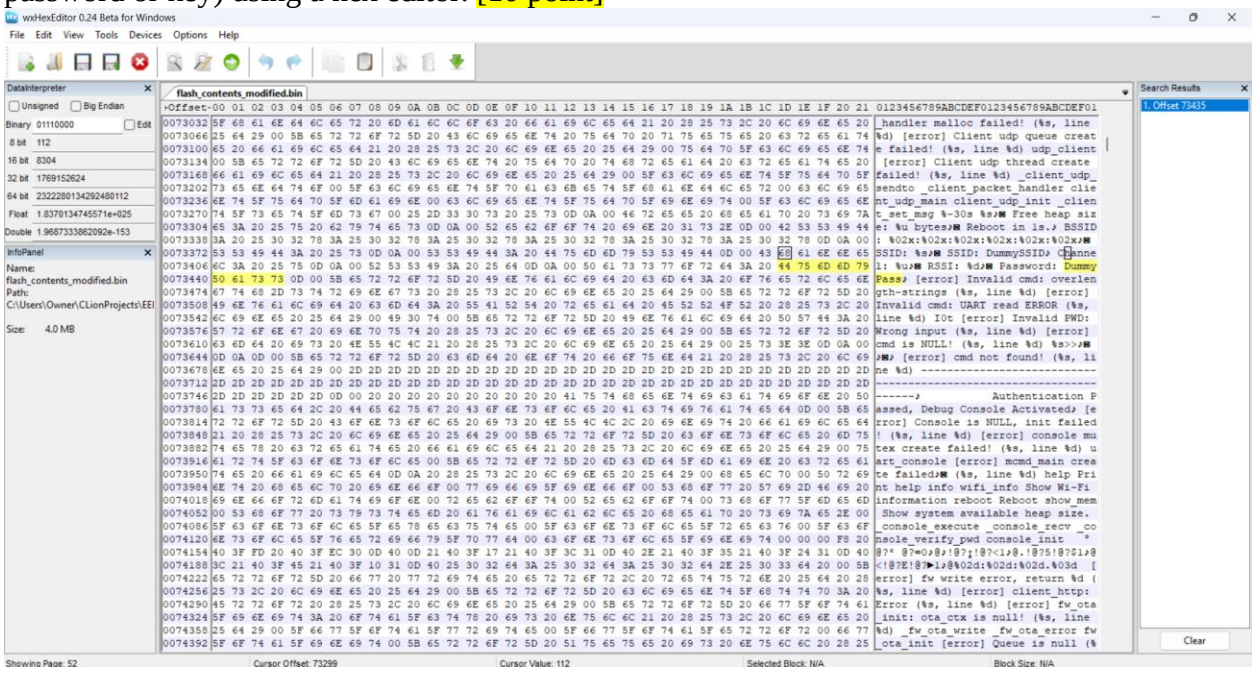
Name	Type	SubType	Offset	Size	Purpose
nvs	data	nvs	0x9000	16K	Stores **Non-Volatile Storage (NVS)**, which holds key-value pairs such as **Wi-Fi credentials, device settings, and calibration data**. The ESP32 SDK uses NVS for persistent storage.
otadata	data	ota	0xd000	8K	Stores **OTA (Over-The-Air) update status**. This partition helps the ESP32 decide which OTA firmware slot (ota_0 or ota_1) is active and ensures proper booting after an update.
phy_init	data	phy	0xf000	4K	Stores **Wi-Fi and Bluetooth PHY initialization parameters**. This data is needed for proper radio operation and is usually preconfigured during flashing.
factory	app	factory	0x10000	1M	Contains the **factory firmware** that is loaded during initial flashing. This firmware is the default boot image when the ESP32 starts unless OTA updates are enabled.
ota_0	app	ota_0	0x110000	1M	Stores the **first OTA firmware slot**. If an OTA update is performed, the new firmware may be written here.
ota_1	app	ota_1	0x210000	1M	Stores the **second OTA firmware slot**. The ESP32 can switch between ota_0 and ota_1 during updates to ensure **fail-safe OTA updates**.

NVS (Non-Volatile Storage): Allows storage of small persistent data across reboots.
OTA Updates: ESP32 can store multiple firmware versions and switch between them.
Factory Partition: The original firmware remains intact, ensuring recovery if OTA updates fail.
PHY Init Data: Ensures proper radio functionality for Wi-Fi and Bluetooth.

2. The following command retrieves the whole flash content although it is also possible to refer to the partition table and select only the occupied part of the flash.
    a. *python C:\Users\Owner \.platformio\packages\tool-esptoolpy\esptool.py read_flash 0 0x400000 flash_contents.bin*
    where *0* is the starting address and *0x400000* is the length of the flash of the ESP32-WROOM-32 surface-mount module board that our IoT kit uses. The whole flash in the binary format is saved in *flash_contents.bin*. Please provide a screenshot of the command. [10 point]

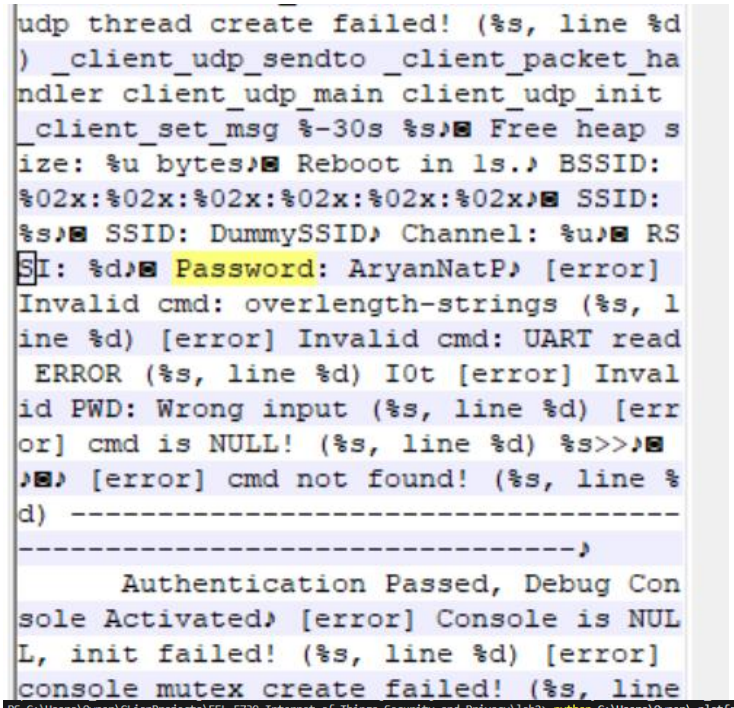
```
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy> python C:\Users\Owner\.platformio\packages\tool-esptoolpy\esptool.p
y read_flash 0 0x400000 flash_contents.bin
esptool.py v3.1
Found 3 serial ports
Serial port COM5
Connecting.....____.
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
WARNING: Detected crystal freq 41.01MHz is quite different to normalized freq 40MHz. Unsupported crystal in use?
Crystal is 40MHz
MAC: 9c:9c:1f:ca:75:58
Uploading stub...
Running stub...
Stub running...
4194304 (100 %)
4194304 (100 %)
Read 4194304 bytes at 0x0 in 381.5 seconds (87.9 kbit/s)...
Hard resetting via RTS pin...
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy>
```

    b. Students can use a hex editor (e.g. wxhexeditor, imhex) to search the WiFi credentials in the flash dump. Please provide a screenshot of found WiFi credentials (e.g. password or key) using a hex editor. [10 point]



```
_handler malloc failed! (%s, line
%d) [error] Client udp queue creat
e failed! (%s, line %d) udp_client
 [error] Client udp thread create
failed! (%s, line %d) _client_udp_
sendto _client_packet_handler clie
nt_udp_main client_udp_init _clien
t_set_msg %-30s %s♪◙ Free heap siz
e: %u bytes♪◙ Reboot in 1s.♪ BSSID
: %02x:%02x:%02x:%02x:%02x:%02x♪◙
SSID: %s♪◙ SSID: DummySSID♪ Channe
l: %u♪◙ RSSI: %d♪◙ Password: Dummy
Pass♪ [error] Invalid cmd: overlen
gth-strings (%s, line %d) [error]
Invalid cmd: UART read ERROR (%s,
line %d) I0t [error] Invalid PWD:
Wrong input (%s, line %d) [error]
cmd is NULL! (%s, line %d) %s>>♪◙
♪◙♪ [error] cmd not found! (%s, li
ne %d) --------------------------
--------------------------------
```

c. In an extreme case, an adversary may attempt to write another firmware to the IoT kit with esptool.py. Please demonstrate writing back to the IoT kit with esptool.py. Students should modify the downloaded flash to change the password to be some combination of all group members' names. Take care that the modified password is the same length as the original. Students must then try to write the changed flash dump to the IoT kit. Please provide a screenshot of the command running. Describe the results (e.g. observe the board, try communicating with it, repeat Lab 1, etc.) and discuss potential reasons for what was observed. [20 points]
Hint: Here is a command writing the flash_contents_modified.bin to the IoT kit
python C:\Users\$USER\.platformio\packages\tool-esptoolpy\esptool.py write_flash 0 flash_contents_modified.bin




```
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab2> python C:\Users\Owner\.platformio\packages\tool-esptoolpy\esptool.py write_flash 0 flash_contents_modified.bin
esptool.py v3.1
Found 3 serial ports
Serial port COM5
Connecting.....____.....____..
Detecting chip type... ESP32
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 9c:9c:1f:ca:75:58
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00000000 to 0x003fffff...
Compressed 4194304 bytes to 1000485...
Wrote 4194304 bytes (1000485 compressed) at 0x00000000 in 96.1 seconds (effective 349.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
PS C:\Users\Owner\CLionProjects\EEL-5739_Internet-of-Things-Security-and-Privacy\lab2>
```

Successfully Password Change:

Conclusion: -

By observing the IoT kit's behaviour after flashing the modified firmware, we can determine how well the device resists unauthorized modifications. Security measures such as firmware integrity checks, secure boot, and signed firmware enforcement can prevent adversarial modifications. If the device allowed the password change, it highlights a potential vulnerability that could be exploited for unauthorized access. The state of the UART board is normal and the red light is not flashing instead it is stable now. After the last command which is given above the board is normal and the system has also Hash verified the details and Hard resetting via RTS pin.

To be double secure regarding the password is modified I used this read command again after the whole process is completed and viewed the flash_contents.bin file to see what password I can see now, and the password change is successfully done as I can see the new password which I have set

References

[1] esptool.py, Accessed on Mar. 6, 2021
[2] Partition Tables, Accessed on Feb 9, 2023