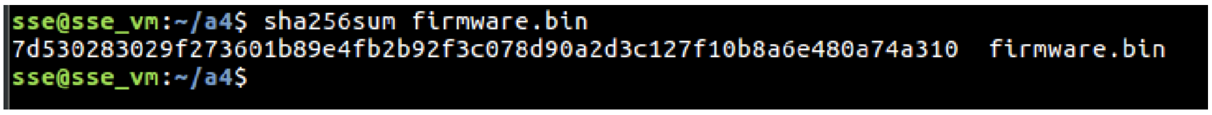
Assignment 4 - Secure systems engineering CS6570

Team: Trojan
Members:
Pradeep Peter Murmu (CS24M033), Vishnu K. (CS24M022)

In This assignment we are given a Nor Flash from which we have to extract the firmware and analyze the code and solve the given questions.

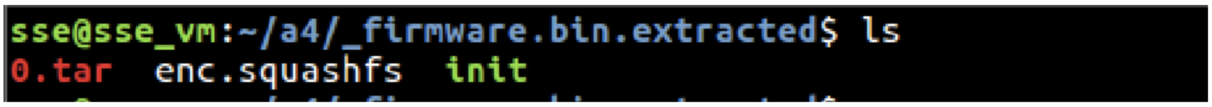
Task 1: While we were trying to understand what was present on the flash, we extracted and read the contents of the flash into a binary blob with sha256sum 7d530283029f273601b89e4fb2b92f3c078d90a2d3c127f10b8a6e480a74a310. Check if you read it correctly?
We extracted the firmware.bin from the flashrom using CH431 programmer. We used sha256sum to verify it as given in the question.



```
sse@sse_vm:~/a4$ sha256sum firmware.bin
7d530283029f273601b89e4fb2b92f3c078d90a2d3c127f10b8a6e480a74a310  firmware.bin
sse@sse_vm:~/a4$
```

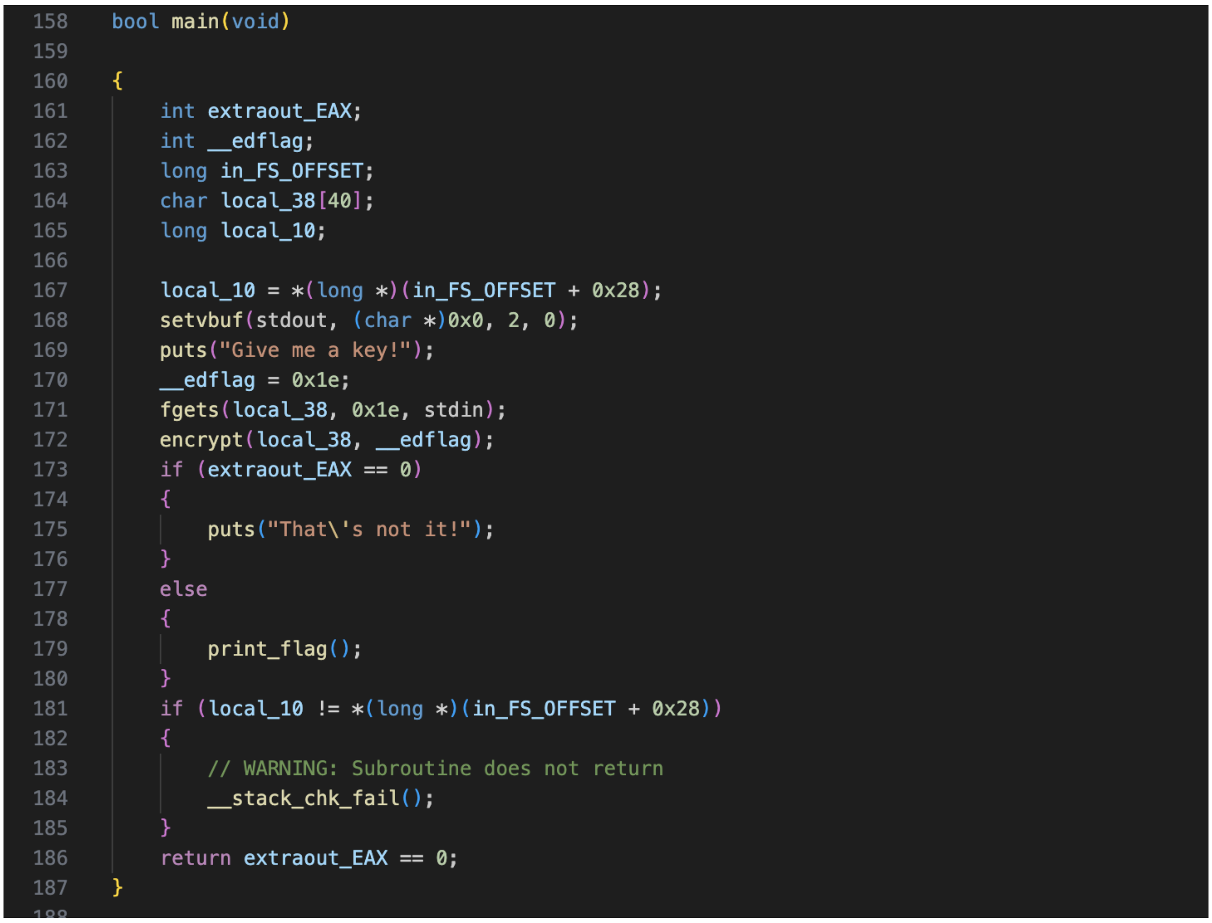
Task-2 : Report the SHA-256 checksum for the correctly processed firmware based on the init code. (10 marks)

We used our handy tool binwalk to extract the firmware.bin and got three files.



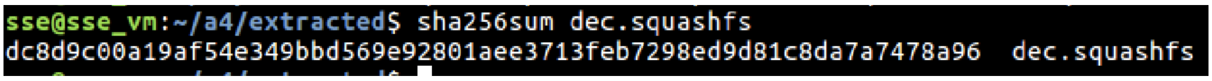
```
sse@sse_vm:~/a4/_firmware.bin.extracted$ ls
0.tar  enc.squashfs  init
```

The enc.squashfs was encrypted so we needed a key. We decompiled the “init” file to search for some hints. We also found that the enc.squashfs was MIPS architecture hence we could use qemu-mips to execute it. Since init code was essentially a key which set some offsets. We used “qemu-mips-static ./init enc.squashfs dec.squashfs” to decrypt the enc.squashfs and store in “dec.squashfs” file.



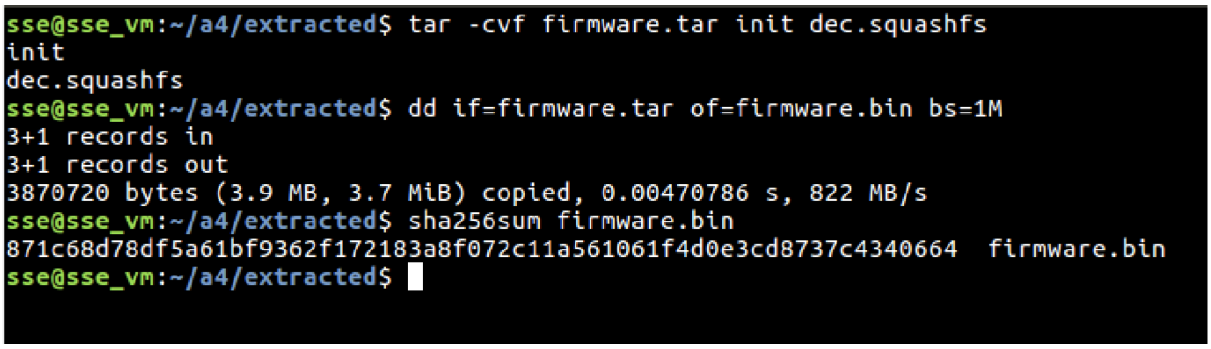
```c
bool main(void)

{
    int extraout_EAX;
    int __edflag;
    long in_FS_OFFSET;
    char local_38[40];
    long local_10;

    local_10 = *(long *)(in_FS_OFFSET + 0x28);
    setvbuf(stdout, (char *)0x0, 2, 0);
    puts("Give me a key!");
    __edflag = 0x1e;
    fgets(local_38, 0x1e, stdin);
    encrypt(local_38, __edflag);
    if (extraout_EAX == 0)
    {
        puts("That\'s not it!");
    }
    else
    {
        print_flag();
    }
    if (local_10 != *(long *)(in_FS_OFFSET + 0x28))
    {
        // WARNING: Subroutine does not return
        __stack_chk_fail();
    }
    return extraout_EAX == 0;
}
```

The sha256sum of decrypted file of squashfs is below:



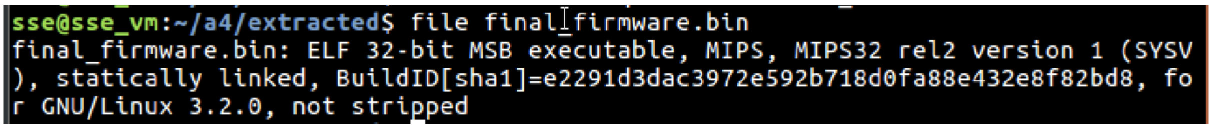
```
sse@sse_vm:~/a4/extracted$ sha256sum dec.squashfs
dc8d9c00a19af54e349bbd569e92801aee3713feb7298ed9d81c8da7a7478a96  dec.squashfs
```

We are also repacking the decrypted file with init to find the hash of processed firmware using following commands.



```
sse@sse_vm:~/a4/extracted$ tar -cvf firmware.tar init dec.squashfs
init
dec.squashfs
sse@sse_vm:~/a4/extracted$ dd if=firmware.tar of=firmware.bin bs=1M
3+1 records in
3+1 records out
3870720 bytes (3.9 MB, 3.7 MiB) copied, 0.00470786 s, 822 MB/s
sse@sse_vm:~/a4/extracted$ sha256sum firmware.bin
871c68d78df5a61bf9362f172183a8f072c11a561061f4d0e3cd8737c4340664  firmware.bin
sse@sse_vm:~/a4/extracted$
```

Task-3: Now that you have the correct firmware, what is the CPU architecture and endianness that the device runs on? (5 marks)



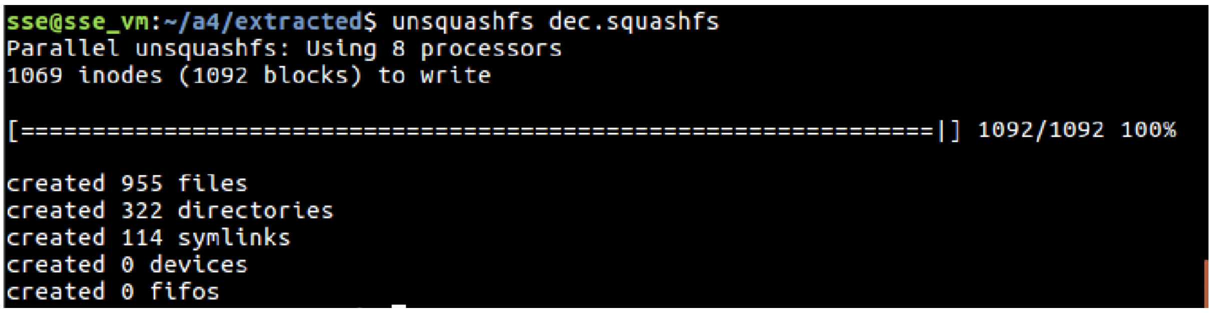
```
sse@sse_vm:~/a4/extracted$ file final_firmware.bin
final_firmware.bin: ELF 32-bit MSB executable, MIPS, MIPS32 rel2 version 1 (SYSV
), statically linked, BuildID[sha1]=e2291d3dac3972e592b718d0fa88e432e8f82bd8, fo
r GNU/Linux 3.2.0, not stripped
```

CPU Architecture:
The file is an ELF 32-bit executable. It is compiled for MIPS (MIPS32 rel2). This means the device runs on a MIPS 32-bit processor.

Endianness:
The output mentions "MSB executable", which stands for Most Significant Byte first. This indicates that the file is compiled for Big Endian architecture.

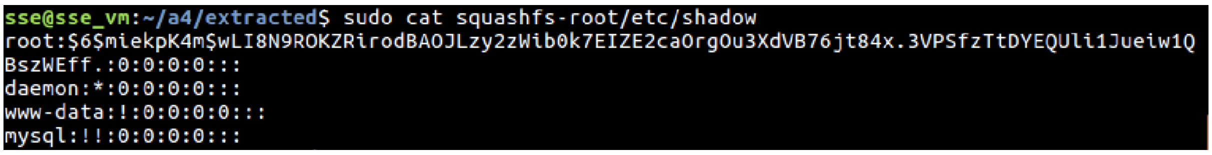
Task-4 : During our testing phase the router gave us a warning that said "the root password was changed", we are wondering what that could be? Can you help identify what the password for the user root was changed to? Clearly explain your approach in the report. (10 marks)

We unsquashed the decrypted squashfs file with using "unsquashfs" command.



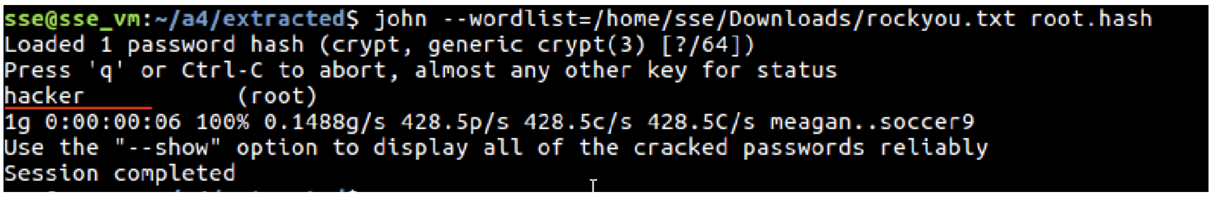
```
sse@sse_vm:~/a4/extracted$ unsquashfs dec.squashfs
Parallel unsquashfs: Using 8 processors
1069 inodes (1092 blocks) to write

[=============================================================|] 1092/1092 100%

created 955 files
created 322 directories
created 114 symlinks
created 0 devices
created 0 fifos
```

We found the extracted root system. To search for the password, we accessed the /etc/shadow inside the directory. We found the hashed value of the password.



```
sse@sse_vm:~/a4/extracted$ sudo cat squashfs-root/etc/shadow
root:$6$miekpK4m$wLI8N9ROKZRirodBAOJLzy2zWib0k7EIZE2caOrgOu3XdVB76jt84x.3VPSfzTtDYEQUli1Jueiw1Q
BszWEff.:0:0:0:0:::
daemon:*:0:0:0:0:::
www-data:!:0:0:0:0:::
mysql:!!:0:0:0:0:::
```

The hash field uses SHA512 crypt (indicated by $6$). Using password cracking tools like john the ripper, we found the password to be **"hacker".**



```
sse@sse_vm:~/a4/extracted$ john --wordlist=/home/sse/Downloads/rockyou.txt root.hash
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
hacker           (root)
1g 0:00:00:06 100% 0.1488g/s 428.5p/s 428.5c/s 428.5C/s meagan..soccer9
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Task-5: We noticed some malicious activity on the network, find and report the IP addresses of these devices along wih their malicious activity. (5 marks)

1. IP Address: 192.168.1.103

   Activity: Attempted unauthorized access to /etc/passwd.

2. IP Address: 192.168.1.104

   Activity: Downloaded /etc/shadow via SCP which is a sensitive file Download.

3. IP Address: 172.16.0.10

   Activity: Multiple failed login attempts from external IP address indicates brute force or DDoS attack.

4. IP Address: 192.168.1.50

   Activity:
   Port scanning detected suggesting probing for vulnerable services.
   Connection request on port 23 (commonly used for Telnet).

5. IP Address: 192.168.1.113

   Activity:
   Accessed a URL containing a command injection attempt:
   telnetd -l /bin/sh -p 2333 -b 0.0.0.0
   This command attempts to open a Telnet service bound to port 2333 and execute /bin/sh as a shell.

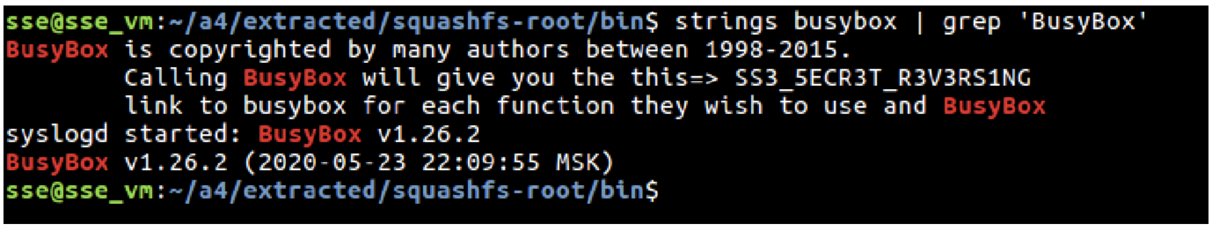
Task-6: There is a specific device which is trying to gain a remote shell access. Can you identify the IP address and program that it is trying to execute and the port number that it is trying to open?

2025-02-28 06:03:15 [INFO] [192.168.1.113] User 'guest' accessed 'www.router.local/showdevices=?new&telnetd -l /bin/sh -p 2333 -b 0.0.0.0'

The IP of the deceive is : 192.168.1.113
The port number it is trying to open: 2333
It is trying to execute "/bin/sh" as a shell via telnet which will give access to remote shell.

Task-7: What actually happens when the malicious user sends these malicious commands, can you trace down the steps of their execution and find the flag that they might have left behind?

We found that BusyBox client is being used for the telnet connection in the previous question. We found the busybox binary in the bin folder.
Using strings in busybox binary we found the secret flag "SS3_5ECR3T_R3V3RS1NG"



```
sse@sse_vm:~/a4/extracted/squashfs-root/bin$ strings busybox | grep 'BusyBox'
BusyBox is copyrighted by many authors between 1998-2015.
        Calling BusyBox will give you the this=> SS3_5ECR3T_R3V3RS1NG
        link to busybox for each function they wish to use and BusyBox
syslogd started: BusyBox v1.26.2
BusyBox v1.26.2 (2020-05-23 22:09:55 MSK)
sse@sse_vm:~/a4/extracted/squashfs-root/bin$
```

Contributions:

We shared hardware with team "Deadsec" and had discussions.

Pradeep P.M. - Explored the firmware bin and wrote the first 4 tasks in this report.
Vishnu K. - Decompiled binaries. Worked on the last 3 tasks of this report.

Acknowledgement:

1. Qemu
2. Perplexity AI for debugging
3. Ghidra and online decompilers