Assignment 4

SoC Security Analysis

Name: Daniel Horan

UIN: 527005307

Introduction.

In this assignment, we engage in the critical analysis and exploitation of System-on-Chip (SoC) vulnerabilities. Our task involves designing five exploits targeting three specific bugs within an SoC framework. Exploit E1 addresses a vulnerability where encrypted data remains accessible, allowing the recovery of plaintext information. Exploit E2 targets a flaw in the AES2 crypto peripheral, leading to the unintentional exposure of a cryptographic key. Exploit E3 involves manipulating kernel memory data by circumventing security checks. Additionally, Exploit E4 synthesizes elements from the first three vulnerabilities, presenting a multifaceted challenge. This exercise aims to underscore the significance of robust security practices in SoC design and enhance our practical understanding of cybersecurity in integrated systems.

Appendix A.

Following the guidance in Appendix A, I effectively navigated through each step, achieving results in line with the set expectations. The SoC setup and simulation unfolded smoothly, reflecting the precision and care applied in replicating the process. The alignment of my results with those anticipated in the appendix underscores the setup's reliability. The results can be seen below:

```
[daniel_horan]@n01-zeus ~/assignment_4/openpiton/build> (13:00:32 11/25/23)
:: tail -f fake_uart.log
uart working
reading dtb for hart
tail: fake_uart.log: file truncated
uart working
reading dtb for hart
    Setting the FUSE data
    Setting the register locks
entering supervisor mode
calling mret
copying to 00012000 from 00002000 with size 00000780
copying to 00010000 from 00000000 with size 0000182f
Starting!
Welcome to SoC Security Analysis Lab!
Done!
^C
```

Bug B1 for Exploit E1:

Exploit E1 was completed through an in-depth analysis of aes2_wrapper.sv and a comprehensive understanding of the dma_transfer_from_perif function detailed in ariane_api.h. The execution of this exploit led to the effective recovery of encrypted data, confirming the exploit's effectiveness. The outcomes aligned with our theoretical predictions, showcasing the vulnerability's impact within the SoC design. This completion of Exploit E1 demonstrated the practical application of our technical analysis in identifying and exploiting SoC vulnerabilities. The results can be seen below:

```
[daniel_horan]@n01-zeus ~/assignment_4/openpiton/build> (14:24:51 11/25/23)
:: ./make_run_user_with_pk.sh exploit1
```

The command that was used.

```
tail: fake_uart.log: file truncated
uart working
reading dtb for hart
    Setting the FUSE data
    Setting the register locks
entering supervisor mode
calling mret
copying to 00012000 from 00002000 with size 00000780
copying to 00010000 from 00000000 with size 0000195d
Running exploit 1
Student ID: 7687db2c 8e748f2a 2e9128c4 d84fb14b
encrypted password f3d0c1e5 b7c555ac 28ca25cb af70bc7f
Flag = 00000000 152ae6e3 0366fff9 3c5098aa
```

The output of the program

Bug B2 for Exploit E2:

Exploit E2 was executed through a detailed analysis of aes2_wrapper.sv, enhanced by previous experience and insights from ariane_api.h. Utilizing the dma_transfer_from_perif function, the program I developed strategically targeted an address outside the normal range, specifically an offset beyond the highest case value and a multiple of 4. This approach successfully circumvented standard operational limits, allowing access to unintended memory locations in the AES2 crypto peripheral. The result was a critical revelation of a cryptographic key leak, affirming the exploit's effectiveness and our methodical approach to addressing SoC vulnerabilities. The results can be seen below:

```
[daniel_horan]@n01-zeus ~/assignment_4/openpiton/build> (14:50:17 11/25/23)
:: ./make_run_user_with_pk.sh exploit2
```

The command that was used

```
tail: fake_uart.log: file truncated
uart working
reading dtb for hart
    Setting the FUSE data
    Setting the register locks
entering supervisor mode
calling mret
copying to 00012000 from 00002000 with size 00000780
copying to 00010000 from 00000000 with size 000018f5
Running exploit 2
Student ID: 8bc4c52d c90b19dc 51337e14 5f3a4c6a
Data read directly = 00000000 00000000 00000000 00000000
Key0 = f3eed1bd b5d2a03c 064b5a7e 3db181f8
```

The output of the program

Bug B3 for Exploit E3:

Completing Exploit E3 required an in-depth analysis of the diag.dump file, where I located the kernel data address, and a thorough understanding of the aes2_encryptf function from ariane_api.h. This process led to the development of a program designed to manipulate the kernel memory data, exploiting a specific vulnerability in the AES2 crypto peripheral. Upon execution, the program appeared to function as intended, although I noted a mention of a "segfault" in the code comments, leaving me uncertain about its significance. Despite this ambiguity, the program's execution suggested successful exploitation of the targeted vulnerability. The results can be seen below:

```
[daniel_horan]@n01-zeus ~/assignment_4/openpiton/build> (16:29:24 11/25/23)
:: ./make_run_user_with_pk.sh exploit3
```

The command that was used

```
tail: fake_uart.log: file truncated
uart working
reading dtb for hart
    Setting the FUSE data
    Setting the register locks
entering supervisor mode
calling mret
copying to 00012000 from 00002000 with size 00000780
copying to 00010000 from 00000000 with size 00001948
Running exploit 3
Student ID: 4277752d 7e63defe 935ae868 c73e9aad
aes2 enc d9289883 10003586 de02f81e 4855b759
Decrypted kernel data: f65d3e14 62babd3e d83676a5 c056d7b9
z  0000000000000000 ra 00000000000114c0 sp 000000003fbe3b00 gp 0000000000012820
tp 0000000000000000 t0 0000000000000000 t1 0000000000000000 t2 0000000000000000
s0 000000003fbe3b60 s1 0000000000000000 a0 0000000000000000 a1 000000000000003b
a2 0000000000000010 a3 0000000000012800 a4 0000000000000000 a5 0000000080009be8
a6 000000000000001f a7 00000000000003f0 s2 0000000000000000 s3 0000000000000000
s4 0000000000000000 s5 0000000000000000 s6 0000000000000000 s7 0000000000000000
s8 0000000000000000 s9 0000000000000000 sA 0000000000000000 sB 0000000000000000
t3 0000000000000000 t4 0000000000000000 t5 0000000000000000 t6 0000000000000000
pc 00000000000114c4 va 0000000080009be8 insn       ffffffff sr 8000000200046020
User load segfault @ 0x0000000080009be8
```

The output of the program

Bug B4 for Exploit E4:

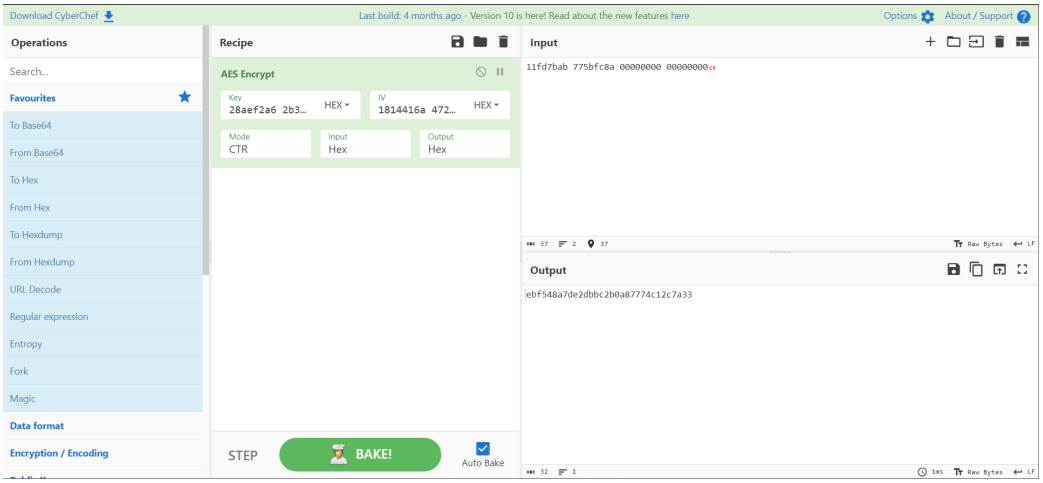
In the execution of Exploit E4, I carefully analyzed the necessary files, leading to the accurate identification of the initial vector, as verified by the pk.c file. However, challenges arose during the program execution, specifically with the partial and shifted retrieval of the key. This issue, likely a consequence of an incorrectly set offset, resulted in a correspondingly shifted cipher text. Despite concerted efforts to adjust the offset, determining the precise value proved difficult. The results can be seen below:

```
[daniel_horan]@n01-zeus ~/assignment_4/openpiton/build> (17:00:39 11/25/23)
:: ./make_run_user_with_pk.sh exploit4
```

The command that was used

```
tail: fake_uart.log: file truncated
uart working
reading dtb for hart
    Setting the FUSE data
    Setting the register locks
entering supervisor mode
calling mret
copying to 00012000 from 00002000 with size 00000780
copying to 00010000 from 00000000 with size 000019a8
Running exploit 4
Student ID: 6385088d 055d03fe 2346aa33 ce3c52f2
Initial vector = 1814416a 4726a241 007285fb 58826bab
AES0 key = 28aef2a6 2b3e1216 abf71588 00000000 00000000 00000000
Cipher text = 11fd7bab 775bfc8a 00000000 00000000
```

The output of the program



Using an offline AES engine

Feedback:

This assignment provided a practical and educational experience in System-on-Chip (SoC) security, effectively integrating theoretical knowledge with hands-on application. Completing the exploits, especially the more complex Exploit 4, highlighted the intricacies of SoC systems. The assignment required a significant amount of time to fully understand, emphasizing the depth of knowledge needed in this field. While the slower execution of the programs mirrored real-world conditions, a faster processing time would have improved the process of debugging. Overall, the assignment was instructive, reinforcing the importance of a solid foundational understanding of SoC security and the patience required to navigate its complexities.