System Security
Trusted Execution Environments

Not Graded

Distribution: 02.11.2023

1 TEE fundamentals

(a) What is the TCB of an application? What is the rationale for using Trusted Execution Environments?

(b) What are the three TEE primitives? Briefly explain their purpose.

(c) What are the trust assumptions of Intel SGX, AMD SEV and ARM TrustZone, respectively? How does each of them enforce runtime isolation?

(d) What are the advantages and disadvantages of SGX, SEV, and TrustZone?

2 Side-Channels against SGX

Enclaves running inside Intel SGX might be vulnerable to side channel attacks if the OS is compromised. This paper describes an attack: `http://ieeexplore.ieee.org/document/7163052/`. You don't have to read it completely, but you should get the overall idea of the side-channel functionality.

(a) Which information about the execution of an SGX enclave does the paper use for the side-channel attack? Which steps does the malicious OS need to take to obtain this information?

(b) How can the OS use this information to infer what is being executed inside the enclave?

(c) What can be done on the application level to prevent such side channels? You can check the paper, read online or think about possible solutions.

For the next questions, we will consider an SGX enclave E. E accepts an AES-encrypted message `cipher` together with a counter value `ctr` through an ECALL, decrypts the message using a secret `key` stored within the enclave, and then processes the resulting plaintext further. The ECALL can be called arbitrarily often. For the purpose of this question, we assume that `key`, `cipher`, and `ctr` are all 8 byte (64 bit) long. The first steps of the decryption process can be abstracted in the following way:

```
decrypt(byte[8] ctr, byte[8] cipher, byte[8] key){
    byte[8] temp = {0};
    //first step of the first round of AES in counter mode
    for (int i = 0; i < 8; i++) {
        // '^' is XOR
        temp[i] = sbox[ctr[i] ^ key[i]];
    }
    //continue with the rest of the AES steps and rounds...
}
```

`sbox` is a byte array of length 256 that is stored statically in enclave memory. The array is distributed among two memory pages: The first half (indices from 0x00 to 0x7F) is on page P_0, the second half (indices from 0x80 to 0xFF) is on page P_1.

(d) Assume the secret `key` is 0x94 b7 b9 3b 6a 45 4b 44, the user-provided `ctr` is 0x7c 9b 05 05 9b 28 3e fe, and `cipher` is 0x34 c6 bb 26 fa 61 56 e9. What is the page trace for P_0 and P_1 produced by the first steps of the decryption?

(e) Provide a `key` which does not have any byte values in common with the original key, but produces the same page trace under the assumption that `ctr` and `cipher` stay the same.

(f) Which conclusions can an attacker draw about the key from the observed page trace? How many brute force attempts does the attacker need to do in the worst case given the side channel information they obtained?

3 New TEE developments

In this exercise, you will get to know two more recently developed TEE technologies: The Realm Extensions for ARM CPUs and Keystone, a TEE Framework for RISC-V.

3.1 ARM Realms

With ARMv9, ARM announced the Realm Management Extensions (RME), which allows running attestable isolated execution environments, so-called realms, on ARM. ARM provides a high-level description of RME and its purpose here: `https://developer.arm.com/documentation/den0126/latest`. By reading the (short) Sections 1 and 2, you should be able to get an initial overview.

(a) Briefly describe the design of RME. How is the isolation of execution contexts achieved?

(b) What is the TCB of (i) software in the non-secure world, (ii) software in secure world, (iii) software running in a realm?

(c) Which component is in charge of assigning resources to realms?

(d) How does RME compare to ARM TrustZone?

(e) Do you see similarities to another TEE technology that you already know?

3.2 RISC-V Keystone

Keystone is a TEE proposal stemming from academia: `https://dl.acm.org/doi/abs/10.1145/3342195.3387532`. Read the paper until and including Section 4.

(a) Briefly describe the design of Keystone. How is the isolation of execution contexts achieved?

(b) What is the TCB of (i) the untrusted host OS, (ii) an eapp?

(c) Recall the side-channel attack on Intel SGX discussed above. Is such an attack still possible with Keystone? Why/why not?

(d) Do you see similarities to another TEE technology that you already know?