Final Review Session

1 Overview

Part 1 of the course covered many cryptographic primitives. You should know them! But Part 2 of the course (covering platform security, software security, and human/end-user security) provided a survey on many systems, attacks, and general security practices that are important to understand.

- Lec 17 - software trust
  - Imports, updates, secure boot
- Lec 18 - hardware attacks
  - Side channel, row hammer Physical attacks
- Lec 19 - iOS security
  - app sec, secure boot, BOOT ROM, secure effaceable storage, Secure Enclave
- Lec 20 - Software Security
  - Bugs: buffer overflow, use after free, encode (SQL), XXS, concurrency bug
- Lec 21 - Privilege Seperation
  - examples: logging, keys, codec, NTP, OpenSSH
- Lec 22 - Bug finding
  - Fuzzing, symbolic execution
- Lec 23 - Runtime Defenses
  - Stack canary, fat pointer, ASLR, non-exe stack
- Lec 24 - Schnorr
  - Discrete Log Problem review, Schnorr, Zero-Knowledge
- Lec 25 - Diff. Privacy
  - Laplace Mechanism for noise

2 Review Problems

Problem 1: Bug Finding
Let's assume we are using a symbolic execution system in order to find any bugs in the following small piece of code:
We notice that when a = 0, b = 2, and c = 1, our code crashes.

```
int x=0, y=0, z=0;
if(a) {
    x = -2;
}
if (b < 5) {
    if (!a && c) { y = 1; }
    z = 2;
}
assert(x + y + z != 3);
```

Draw an execution tree for symbolic execution on the above code. How many different symbolic execution paths are there in the above code?

Part b. Now, let's say we are using a fuzzer to find the same bug. Let's assume that the code coverage mechanism and corpus for said Fuzzer does not significantly impact runtime of the fuzzing process, and effectively the Fuzzer picks Fuzzed inputs uniformly at random (which is not true in practice). If inputs `a,b,c` are all obtained from `ConsumeUInt(2)`), what is the expected number of inputs before our Fuzzer finds the bug?

Part c. What are the is a pro of using a Fuzzer over Symbolic Execution?

Part d. What is a pro of using Symbolic Execution over a Fuzzer?

Problem 2: Schnorr
Recall the Schnorr signature game from lecture, say the Prover knows c=1 before the protocol begins. Describe how, without knowing x, the Prover could still pass the game, tricking the Verifier.

Say the verifier could run the protocol twice, once with c=0 and once with c=1, how describe how the verifier could obtain the value of x.

Consider a modified version of Schnorr's signature in which the signing nonce r is computed as $r \leftarrow H(m)$, where $H : 0, 1^* \rightarrow Zq$ is a hash function, m is the message to be signed, and q is the order of the group used for the signature scheme. Is this deterministic version of Schnorr's signature scheme is secure?

Problem 3: Differential Privacy
Your Laboratory, Happiness Inc, conducts a top secret survey, which polls the age and average reported happiness of 8, very private individuals. You want to employ differential privacy techniques while releasing some statistics about your data.

age	happiness
14	.1
10	.5
20	.8
30	.7
40	.1
61	.1
43	.2
12	.1

Part a. Let's say we want to apply differential privacy with ϵ to report the mean of our `age` data. What Laplace noise would we want to use? What about for the mean of our *happiness* data?

Part b. Let's say we want to apply differential privacy with ϵ to report the maximum of our age data. What Laplace noise would we want to use?

Part c. Which statistic requires more noise to report?

Problem 4: Isolation
Let's say we are implementing an isolated system using Time Multiplexing, for example the video game system discussed in class. Does the system provide integrity, non-leakage and non-interference? What assumptions do you need to make in order to acheive each of these isolation goals?

Problem 5: iOS
Match the following properties to the component. A property may apply to more than one component.

Components
Boot ROM
Secure Storage
Secure Enclave
Bootloader

Properties
Effacable
Non-Writeable
Contains Guess Counter
Contains a Verification Key
Contains an Encryption Key
Enforces PIN Delay
Part of Secure Boot
Part of User Data Protection

Problem 6: Privilege Separation
Let's say that the NTP (Network Time Protocol), instead has the Time Service and Network service running in one process. What could potentially go wrong with this plan?

Recall the Logging Privledge Seperation plan from lecture. Let's say our application server has direct access to our Logging database, what could potentially go wrong with this plan?

Problem 7: Runtime Defenses
Assume that p is a fat pointer. Which of these scenarios would fail/succeed?
Scenario 1:

```
p = malloc(4);
*p = 1;
```

Scenario 2:

```
p = malloc(4);
q = p;          // copy fat ptr to q.
q = q + 8;      // add 8 to q.curr.
*q = 1;
```

Scenario 3:

```
p = malloc(4);
q = p;          // copy fat ptr to q.
q = q + 2;      // add 2 to q.curr.
*q = 1;
```

Scenario 4:

```
p = malloc(4);
q = p;          // copy fat ptr to q.
q = q - 1;      // subtract 1 from q.curr.
*q = 1;
```

Name 2 downsides to using fat pointers.