Final Solutions

Question	Parts	Points
1: True or False	6	12
2: Encryption scheme	1	10
3: Authentication schemes	2	20
4: Isolation	2	20
5: Symbolic execution	1	15
6: Google Chrome security	2	10
7: Law and policy	2	10
8: Lab 4: WASI escape	1	15
9: Lab 5: timing attack	2	17
10: iPhone security	5	15
11: Splitting trust	3	30
12: Course survey	2	6
Total:		180

Name: ____

This exam is printed double-sided!

Problem 1. [12 points] **True or False** (6 parts)
Please write **T** or **F** for the following. *No justification is needed (nor will be considered).*

(a) [2 points] We know how to construct a public-key encryption scheme assuming only one-way functions.

Solution: False.

(b) [2 points] $F(k, x) = k \oplus x$ is a secure PRF.

Solution: False.

(c) [2 points] If H_1 and H_2 are collision-resistant hash functions then $H(x) = H_1(H_2(x))$ is also collision-resistant.

Solution: True.

(d) [2 points] A CPA-secure encryption scheme must be randomized.

Solution: True.

(e) [2 points] Given a secure signature scheme for n-bit messages, one can construct a secure signature scheme for messages of length $2n$ bits, by partitioning the $2n$-bit message into two equal chunks (each of length n), and signing each chunk using the underlying secure signature scheme.

Solution: False.

(f) [2 points] If a mechanism A provides ϵ-differential privacy, then the mechanism A also provides 2ϵ-differential privacy.

Problem 2. [10 points] **Encryption scheme** (1 part)
Assume that one-way functions exist. Does there exist a CPA-secure encryption scheme that takes as input a 128-bit message and a random string, and outputs a 128-bit ciphertext? Explain your answer.

Solution: The answer is no, since then each message must have a unique ciphertext (by the pigeon-hole principle), and then one can attack as explained in class.

Problem 3. [20 points] **Authentication schemes** (2 parts)
Suppose we are given a PRF that outputs a single bit, with a key space K and a message space M; namely, $F : K \times M \to \{0, 1\}$.

(a) [10 points] Is the following a secure MAC, for the same key space K and message space M: for a key $k \in K$ and message $m \in M$, $\mathrm{MAC}(k, m) := F(k, m)$?

Solution: No, one can guess the tag with probability $1/2$.

(b) [10 points] Suppose that the message space M above is $M = \{0, 1\}^{135}$. Show how to use the PRF F to construct a secure MAC scheme with message space $\{0, 1\}^{128}$.

Solution: $\mathrm{MAC}(k, m) = (F(k, m||i))_{i=1}^{128}$, where each $i \in [128]$ is encoded in binary and thus is an element in $\{0, 1\}^7$.

Problem 4. [20 points] **Isolation** (2 parts)

Ben Bitdiddle wants to run two applications strongly isolated from one another, so he runs them on two separate computers. However, he has just one storage server. His storage server implements a simple network API, whose pseudo-code is shown below (here `src` indicates which of the two separate computers are making the request; assume the adversary cannot tamper with the `src` argument):

```
class Storage:
  def __init__(self):
    self.files = {}

  def write(self, src, filename, contents):
    self.files[filename] = (src, contents)

  def read(self, src, filename):
    if filename not in self.files:
      return ErrorNotFound()
    (owner, contents) = self.files[filename]
    if owner == src:
      return contents
    else:
      return ErrorNotAllowed()
```

(a) [10 points] Does Ben's design provide integrity for the isolated applications, as defined in the isolation lecture? Explain why or why not.

Solution: No: one application can corrupt another application's data by writing over files with the same filename.

(b) [10 points] Does Ben's design provide non-leakage for the isolated applications, as defined in the isolation lecture? Explain why or why not.

Solution: No: one application can determine the names of files used or not used by the other application, by probing different file names using `read()`.

Partial credit for yes: if only one application is running, it cannot read the data of another application. This effectively assumes that the applications use a well-known set of file names that is not dependent on the application's state.

Problem 5. [15 points] **Symbolic execution** (1 part)
Ben Bitdiddle is running a symbolic execution tool on some function `f(`x`,` y`)`, where x and y are arbitrary symbolic 32-bit values passed as arguments to `f`. The symbolic execution tool issues the following queries to the SAT solver:

- $(x > 0)$
- $(x > 0) \wedge (x + y > 0)$
- $(x > 0) \wedge (x + y > 0) \wedge (x == 0)$
- $(x > 0) \wedge (x + y > 0) \wedge \neg(x == 0)$
- $(x > 0) \wedge \neg(x + y > 0)$
- $\neg(x > 0)$
- $\neg(x > 0) \wedge (x + y < 0)$
- $\neg(x > 0) \wedge \neg(x + y < 0)$

Write down a sketch of Ben's function `f` that would generate these queries under symbolic execution:

```
void f(unsigned int x, unsigned int y) {
```

Solution:

```
if (x > 0) {
  if (x+y > 0) {
    if (x == 0) {
      ...
    }
  }
} else {
  if (x+y < 0) {
    ...
  }
}
```

```
}
```

Problem 6. [10 points] **Google Chrome security** (2 parts)

(a) [5 points] What is the approximate black-market cost for a zero-day vulnerability in Google Chrome that allows a remote adversary to execute arbitrary code and escape from Chrome's sandbox? Circle the best answer.

- $10K
- $100K
- $1M
- $10M

Solution: $1M.

(b) [5 points] If the black-market price of a zero-day vulnerability in Google Chrome went down, would the Chrome security team view this as a success? Explain why or why not.

Solution: No: if bugs become cheaper, it means there's either fewer people that want them (so Chrome is less important) or there's more bugs (so Chrome has more vulnerabilities).

Problem 7. [10 points] **Law and policy** (2 parts)

(a) [5 points] Explain in 1-2 sentences what “going dark” is, according to Jennifer Granick, in the context of end-to-end encryption.

Solution: “Going dark” is the term law-enforcement agencies have often used to describe the fact that they cannot decrypt end-to-end encrypted traffic.

(b) [5 points] Earlier this year, Apple proposed a system that would scan photos on users’ iPhones for illegal/exploitative material. Explain in 1-2 sentences why, according to Jennifer Granick, privacy experts objected to this technology.

Solution: One concern was that the system could be abused by domestic or foreign governments for repressive purposes.

Problem 8. [15 points] **Lab 4: WASI escape** (1 part)
Is it possible to exploit the WASI file system sandbox bug that you used in lab 4 part 2 (i.e., access a file outside of the sandbox directory) without using either the `symlink()` or `openat()` system calls? Describe how, or explain why not.

Solution: Not possible. The bug in the WASI sandboxing plan is that the depth for an open file or directory might be wrong if that file or directory was moved in the file system tree after it was opened. But without being able to open a pathname relative to an existing directory file descriptor (which is only done by openat), there is no way to take advantage of this bug; no other code looks at the (possibly incorrect) depth in a file descriptor.

Problem 9. [17 points] **Lab 5: timing attack** (2 parts)

(a) [8 points] Ben Bitdiddle is developing his attack for lab 5, implementing the code for the function `steal_secret_token(l)`. Recall that the argument `l` is the number of random bytes that were used to generate the secret token that the attack code must guess. Ben's plan is to guess one of 256 possible values for each of the `l` bytes at a time, but he discovers that his attack is taking way too long. What should Ben be doing instead to speed up his attack? (We are looking for a big speedup—at least $2\times$ faster.)

Solution: He should be guessing one of the `2*l` nibbles (from 0 to 15) at a time, since the token is hex-encoded and checked one hex digit at a time.

(b) [9 points] Alyssa P. Hacker is also working on her lab 5 attack. She gets her solution mostly working, and is able to guess almost all of the token, but for some reason, her attack does not work for the very last byte of the token. What is Alyssa missing, and how should she fix her attack?

Solution: For the last character of the token, the checking code executes in about the same time regardless of whether the last character is correct or not. Alyssa's attack should instead try each of the possible values and see which one causes the server to reply with success vs failure.

Problem 10. [15 points] **iPhone security** (5 parts)
An iPhone app developer finds an exploitable buffer overflow in the iPhone's kernel on the application processor.

For each of the following questions, answer either **True** or **False** and give a one-sentence explanation.

(a) [3 points] The app developer may be able to exploit this overflow to get root on the phone's application processor.

Solution: True.

(b) [3 points] A website that the phone's owner visits may be able to exploit this vulnerability to get root on the phone's application processor.

Solution: True.

(c) [3 points] A malicious app developer can exploit this vulnerability to persistently corrupt the application-processor's kernel, such that the attacker's kernel code continues to run even after restarting the phone.

Solution: False.

(d) [3 points] If an attacker can exploit this overflow, they can corrupt the OS in a way that prevents the phone from booting.

Solution: True.

(e) [3 points] If an attacker can exploit this overflow, it can corrupt the phone's BootROM.

Solution: False.

Problem 11. [30 points] **Splitting trust** (3 parts)

Apple has a secret key that it uses to sign its iOS operating system updates. Apple engineers want to split this key into n pieces and to give one piece to each of n engineers in a way that ensures that:

- no strict subset of engineers can produce a valid signature but
- all n engineers together can sign a release.

(a) [10 points] Say that there are $n = 4$ engineers and that the signing key k is 256 bits long. One engineer proposes splitting the key into four 64-bit chunks and giving one chunk to each of the engineers. Is this scheme secure? Why or why not?

Solution: No. If three engineers collude, they can learn all but 64 bits of the key and can brute-force search to find the rest.

(b) [10 points] Explain how the engineers can split the key into n pieces (for any n) such that

1. it is possible to recover the key given all n pieces but
2. no one piece leaks any information about the key.

Solution: Pick random $k_1, k_2, k_3, k_3 \in \{0, 1\}^{256}$ such that $k = k_1 \oplus k_2 \oplus k_3 \oplus k_4$. Security follows from security of the one-time pad.

(c) [10 points] Explain how the engineers can split the key into n pieces (for any $n > 3$) such that

1. it is possible to recover the key given ANY THREE pieces but
2. no one piece leaks any information about the key.

Solution: For each size-three subset of users, run the scheme of part (b). Give one share to each user.

Problem 12. [6 points] **Course survey** (2 parts)
We would like your feedback on how to improve this class when we teach it next year. **Any answer, except a blank answer, will receive full credit.**

(a) [3 points] What aspects of the guest lectures did you find valuable to you? Are there other things you wished the guest lectures covered?

(b) [3 points] Which lab assignment did you find to be the least valuable? What else would you have wanted to see in lab assignments instead?