6.858 Quiz 1

Huy C. Dai

TOTAL POINTS

67 / 80

QUESTION 1

Google 8 pts

1.1 Google A 2 / 2

+ **0 pts** "True" is incorrect. The correct answer is "false".

✓ **+ 2 pts** **"False" is correct.**

+ **1 pts** 1 point for no answer.

+ **2 pts** Stated assumption about attacker having end-user permission ticket.

+ **1 pts** Special case.

1.2 Google B 2 / 2

✓ **+ 2 pts** **"False" is correct.**

+ **1 pts** No answer.

+ **0 pts** "True" is incorrect.

1.3 Google C 2 / 2

✓ **+ 2 pts** **"True" is correct.**

+ **1 pts** No answer.

+ **0 pts** "False" is incorrect.

+ **2 pts** Correct by assumption about exactly how the decryption works.

1.4 Google D 2 / 2

✓ **+ 2 pts** **"False" is correct**

+ **1 pts** No answer.

+ **0 pts** "True" is incorrect.

+ **2 pts** Full credit by assumption that Gmail must work on user data while user is not logged in.

+ **2 pts** Special case

QUESTION 2

2 U2F 9 / 9

✓ **+ 9 pts** **Correct**

+ **5 pts** Replay attack

+ **2 pts** Describe what specific steps the attacker takes, e.g., record the signed message by a compromised browser or a phishing website.

+ **2 pts** Describe what the attacker achieves.

+ **0 pts** Click here to replace this description.

+ **5 pts** Phishing attack

+ **5 pts** Man in the middle attack

+ **5 pts** Attackers can generate valid challenge-signature pairs using only the public key.

+ **3 pts** Conceptually correct but missing the details of the attacking steps and what the attacker can achieve.

+ **5 pts** DDoS attack

+ **1 pts** Conceptually correct but missing the details of the attacking steps and what the attacker can achieve.

QUESTION 3

3 Baggy 9 / 9

✓ **+ 9 pts** **Correct**

+ **3 pts** 256 Power-of-2 allocation size

+ **2 pts** Pointer arithmetic staying between +/- slot_size/2 of allocation.

+ **3 pts** Understanding pointers are not OOB if they are < 256 but > 130

+ **1 pts** Crash line is right

+ **0 pts** Wrong

QUESTION 4

4 OKWS 8 / 8

✓ **+ 8 pts** **Correct: prevent ability to issue arbitrary DB queries**

+ **4 pts** Split service needed

+ **0 pts** Split service not needed

QUESTION 5

5 Firecracker: read syscall 5 / 5

✓ **+ 5 pts Cannot violate: Adversary can compromise the kernel in their AWS Lambda MicroVM but adversary cannot invoke read() on host kernel.**

+ 3 pts Cannot violate + inaccurate explanation

+ 0 pts Can violate

QUESTION 6

6 Firecracker: virtio 0 / 5

+ 5 pts Cannot violate

Because of Jailing

+ 5 pts Can violate

It allows escaping outside of MicroVM

✓ **+ 0 pts Wrong/Empty/No explanation**

+ 0 pts Incorrect explanation.

+ 3 pts Special case

QUESTION 7

7 Firecracker: kvm bug 5 / 5

✓ **+ 5 pts Access Linux kernel**

+ 3 pts Can violate

+ 0 pts Cannot violate

QUESTION 8

8 WebAssembly 5 / 10

✓ **+ 5 pts Point out bounds checking issues**

+ 2.5 pts Some explanation of wasm bounds checking (section 7.1)

+ 2.5 pts Full explanation (section 7.1 code specialization - memsize can be hardcoded in generated code, which is no longer safe if memsize can shrink)

+ 0 pts No answer or incorrect answer

QUESTION 9

9 iOS 9 / 9

✓ **+ 9 pts Assumptions stated (attacker has/doesn't have signed version of old iOS, respectively) and correct attack (possible/impossible, respectively)**

+ 4 pts Some explanation, but not quite correct/complete

+ 0 pts No answer or incorrect answer

QUESTION 10

10 EXE 7 / 10

+ 10 pts Correct (7 paths)

✓ **+ 7 pts Correct reasoning but wrong number of paths**

+ 3 pts Some reasoning, but incorrect/incomplete

+ 0 pts No answer or wrong answer w/ no explanation or wrong explanation

QUESTION 11

11 Feedback 2 / 2

✓ **+ 2 pts Thank you for the feedback.**

+ 0 pts Blank answer.

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Spring 2022

Quiz I

You have 80 minutes to answer the questions in this quiz. In order to receive credit you must answer the question as precisely as possible.

Some questions are harder than others, and some questions earn more points than others. You may want to skim them all through first, and attack them in the order that allows you to make the most progress.

If you find a question ambiguous, be sure to write down any assumptions you make. Be neat and legible. If we can't understand your answer, we can't give you credit!

Write your name and submission website email address on this cover sheet.

This is an open book, open notes, open laptop exam.
NO INTERNET ACCESS OR OTHER COMMUNICATION.

This quiz is printed double-sided.

Please do not write in the boxes below.

I (xx/8)	II (xx/9)	III (xx/9)	IV (xx/8)	V (xx/15)	VI (xx/10)	VII (xx/9)	VIII (xx/10)	IX (xx/2)	Total (xx/80)

Name: Huy Chi Dai

Submission website email address: huydai@mit.edu

You can answer the feedback questions on the back of the quiz before the official start time.

This page intentionally left blank.

I Google security architecture

1. [8 points]: Suppose an adversary compromises the storage service responsible for storing the Gmail data for some Google user (call them X), meaning that the adversary's code is already running on that storage service. In which of the following scenarios would the adversary be able to obtain X's Gmail messages?

Circle True or False for each choice. We will give 2 points for a correct answer, 1 point for no answer, and 0 points for the wrong answer.

A. **True** / **False** If the inter-service communication configuration allows the adversary to issue RPCs to the compromised storage service.

B. **True** / **False** If user X logs into their Google account and accesses another service (such as Google Calendar).

C. **True** / **False** If user X logs into their Google account and accesses their Gmail service.

D. **True** / **False** If the adversary compromises the Gmail service in addition to the storage service (but user X never logs in).

This page intentionally left blank.

II U2F

Consider the U2F protocol as described in "Security Keys: Practical Cryptographic Second Factors for the Modern Web." Ben Bitdiddle wants to optimize U2F by not requiring the server to send a randomly-generated challenge during the authentication phase. Instead, in Ben's modified U2F protocol, the client generates its own challenge, uses it in the same way as regular U2F, and sends it to the server along with the signature. The server then uses the challenge sent by the client for verification. Assume that Ben makes no other changes to the server except for using the client-supplied challenge.

2. [9 points]: Describe an attack that an adversary could perform, within the U2F threat model, against Ben's modified U2F which is prevented in the original U2F design. Be specific: describe what the attacker does, what steps they take, and what they achieve.

Ben's modified U2F is vulnerable to a Man in the middle attack. Since the server cannot bind a specific challenge to a certain user authentication session, an adversary can open a fake website which the user opens (through a phishing attack) and attempts to log in w/ U2F as 2FA. The adversary collects the signed challenge sent by user and forwards it to the true server, which then will allow the adversary to log in as the user (as signed challenge should be correct). ~~The adversary can also send the server's response to the user so~~

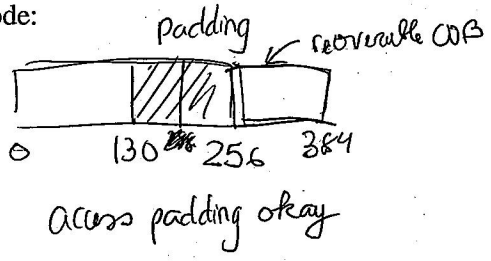
This page intentionally left blank.

III Baggy bounds

Consider the 32-bit version of Baggy Bounds as described in the paper "Baggy Bounds Checking: An Efficient and Backwards-Compatible Defense against Out-of-Bounds Errors" by Akritidis et al (plus the errata). Assume a slot size of 256 bytes (instead of 16 as in the paper).

3. [9 points]: An application runs the following code:

```
1 char *q = malloc(130);
2 char *r = q + 300;
3 char *s = r - 100;
4 char a = *s;
5 *r = a;
6 char *n = NULL;
7 *n = a;
```



Assume that the call to `malloc` succeeded. What line will the program crash on, and why?

The program will crash on line 5. Line 2 is okay because the pointer is w/in slot size/2 from end of allocation boundary. Similary, line 3 is ok because it is w/in the 256 bytes boundary. We allow for read/write access to padding (line 4) in baggy bounds.

Line 5 fails because it attempts to write to the OOB pointer of (*r), and thus will result in an error.

This page intentionally left blank.

IV OKWS

Ben Bitdiddle is using OKWS to build his web application, but he is writing all of his services using Rust, a memory-safe programming language that does not allow programmers to make mistakes that lead to buffer overflows. (He avoids any "unsafe" code in his Rust applications, which otherwise could have allowed programmer mistakes to lead to memory corruption.) As a result, he decides he doesn't need to run the services separate from one another: he runs all services in one process, with one connection to the database.

4. [8 points]: Does Ben's choice of running all services together in one process reduce security, given that all of his service code is written in Rust? Explain why the split-service design is no longer needed, or explain what attacks a split-service design might still prevent. Be specific.

While Ben's code might be safer in terms of guarding against buffer overflow attacks, an attackere might still be able to exploits bugs elsewhere or unintentional bad permissions handling done by the code. For example, if his server serves files to the user where the user provids file path as part of the request, if Ben is not careful, an attacker may be able to leak sensitive source code, database, or even system files. This can be prevented with a split-service design, where you can chroot a certain service (like fileservice) and limit its access to non-relevant files. Also, if the attacker figures out a way to get a certain part of system to crash (e.g. go into infinite loop due to computation bug), the attacker can DOS / crash the full service. This is less of an issue w/ process separation, since you can restart service individually.

This page intentionally left blank.

V Firecracker

Consider the paper "Firecracker: Lightweight Virtualization for Serverless Applications." For the following scenarios, describe what security goals of the AWS Lambda service running Firecracker an adversary might be able to violate (and how they might be able to do that), or explain why the scenario does not give the adversary the ability to do any additional damage. For each question, assume that this is the only exploitable bug that the adversary knows about.

Circle at most one of "Can violate" or "Cannot violate", and explain.

5. [5 points]: The adversary finds an exploitable buffer overflow in the Linux kernel implementation of the `read()` system call.

Can violate / Cannot violate

The guest kernel is ~~ab~~ abstracted away from the true system kernel, and is also untrusted and so exploiting this buffer overflow will only affect the container's kernel but does not allow it to affect other ~~the~~ containers on the system.

6. [5 points]: The adversary finds an exploitable buffer overflow in how the virtio network device implementation in the Firecracker process (say, in the unsafe portions of the Rust code) handles the guest-VM-accessible virtual device memory.

Can violate / Cannot violate

The adversary can send special exploits to its virtio network device that will cause buffer overflow in the system QEMU, which then allows it to potentially write to ~~and~~ other containers virtual ~~de~~ memory and create additional damage / crash other containers.

7. [5 points]: The adversary finds an exploitable buffer overflow in the Linux KVM handler for VM traps.

Can violate / Cannot violate

The adversary can use that buffer overflow to ~~potentially~~ control ~~left~~ KVM to emulate other instructions for it or to peer into the direct hardware page tables of other containers.

This page intentionally left blank.

VI WebAssembly

Consider the paper "Bringing the Web up to Speed with WebAssembly."

8. [10 points]: Alyssa P. Hacker extends WebAssembly to support de-allocating memory, by adding a `shrink_memory` *n* instruction that reduces the allocated memory size by *n* bytes, the opposite of `grow_memory` *n*. How can this change lead to a security problem in a JIT-based WebAssembly runtime that was secure before this change? Assume the runtime executes just one WebAssembly program, once.

When the memory is shrinked, the data that was stored there still remains. This can cause a security issue when ~~the~~ say, another part g the program (another module) requests for the memory to be extended again. In this case, that chunk g previously used memory has data that is no longer tracked by the WebAssembly runtime, and so the current module can easy read from it, exposing unintended data. This ~~causes a security problem of~~ not intro limiting shared between mod

We assume that other module has access to this memory space from an import/ export statement

This page intentionally left blank.

VII iOS security

9. [9 points]: Suppose that Apple suddenly discovers that many iPhones manufactured since 2020 were assigned the same ECID. An adversary physically steals someone's iPhone, with the same ECID as the attacker's phone, and wants to break into it by installing an older version of iOS, from 2015, which has an exploitable vulnerability. Explain how the adversary can perform this attack, or explain why it's not possible.

The attacker can peryorm this attack by ~~directly~~ first ~~loading copying~~ getting the old version g iOS installed on his phone, which is possible if the adversary's phone is old enough that it can be upgraded legitimately to the 2015 version. Following that, the adversay can simply write / copy his OS image to the victim's phone. The system then is expected to boot correctly. Since the only protection for OS image is legitimate at bootup is if the ECID on it matches the ECID stored on the bootROM. check Afterwards, the attack can exploit the vulnerability in the iOS to get access. However, he may still phon find that he cannot access most files since they require the passcode derived key to unlock, and this key is erased after a period g time and/or restart.

This page intentionally left blank.

VIII EXE

Consider the following code fragment in C in which stepN is a symbolic variable, running in the EXE system as described in the paper "EXE: Automatically generating inputs of death".

```
1  char buf[10];
2
3  int f() {
4    int stepN;
5    int sum = 0;
6    int i;
7
8    markSymbolic(&stepN);
9
10   for (i = 0; i < 10/stepN; i++) {
11     sum += buf[i*stepN];
12   }
13   return sum;
14 }
```

10, 5, 3, 2,

10. [10 points]: How many execution paths does EXE explore when executing `f()`?

2

EXE, when it reaches for loop, can choose to go into it or not (2).

If it goes into it, the for loop can take max 10 iterations, each time creating a branch depending on wheter conditon is satisfied. So there's total of ~~1 + 10 = 11 total~~ execution paths EXE can take (no for loop) (stop at which step) error

~~$2^0 + 2^1 + \dots + 2$~~

Final answer:

~~1 + 10 = 11 total paths~~

1 + 4 = 5 total paths

(4 possible values that 10/stepN can take on)

IX 6.858

We'd like to hear your opinions about 6.858. Any answer, except no answer, will receive full credit.

11. [2 points]: Is there one paper out of the ones we have covered so far in 6.858 that you think we should definitely remove next year? If not, feel free to say that.

Baggy Bounds → not really sure how useful it is as a technique anymore and there's a lot of limitations to what it can prevent/detect.

End of Quiz