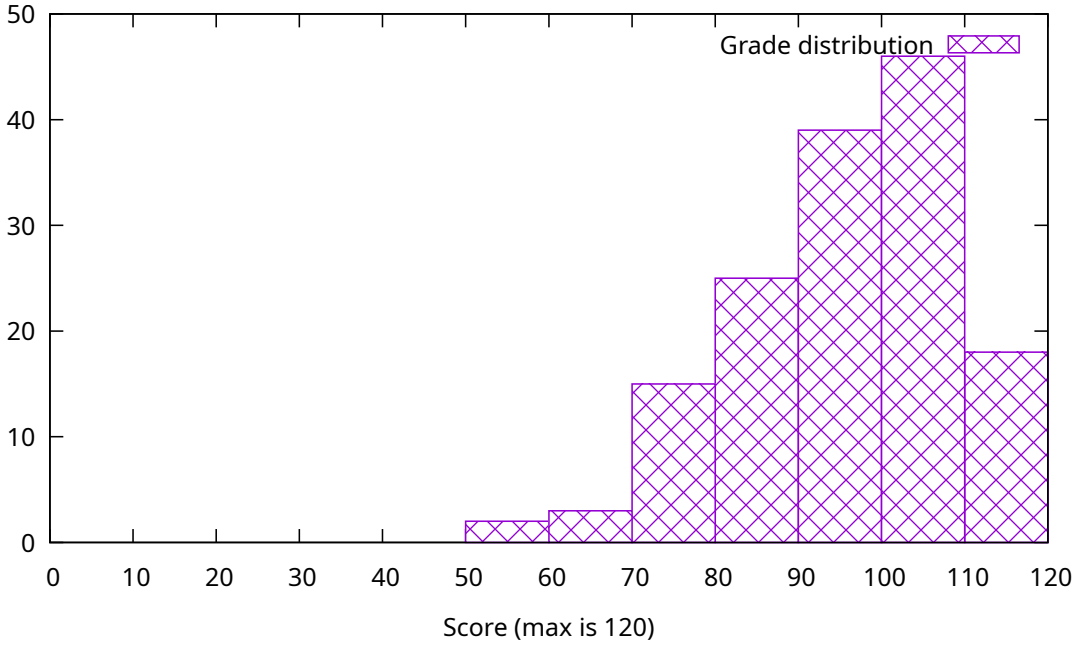
*Department of Electrical Engineering and Computer Science*

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

**6.858 Spring 2022**

# Quiz II Solutions

Mean 96.4 Standard deviation 13.2



# I Web security

Ben Bitdiddle has an account at BensBank, `https://bensbank.com/`. Like most web sites, BensBank uses cookies to track the sessions of logged-in users. However, BensBank developers forgot to set the `Secure` attribute on the session cookie for `bensbank.com`.

**1. [8 points]:** How can a network adversary gain access to Ben's account? Describe what a network adversary would need to do, and what they would need to trick Ben into doing. Assume that Ben will not type his password into any site other than `https://bensbank.com/`, and will not type any Javascript code into his browser; assume there are no bugs in the BensBank web application or in the web browser.

**Answer:** Trick Ben into visiting `http://bensbank.com`, which will cause Ben's browser to send Ben's cookie over HTTP, since it's not marked Secure. Since the adversary has access to the network, the adversary can then learn Ben's cookie and use it to access Ben's account.

Alyssa P. Hacker is developing a web application hosted at `https://alyssa.com/`. She uses a Javascript spell-checking library from her friend Carl, by including the following HTML tag in the `https://alyssa.com/` web page:

```
<SCRIPT SRC="https://carls-code.com/spell.js"></SCRIPT>
```

Unfortunately, an adversary was able to corrupt the `spell.js` file hosted at `carls-code.com`.

Now, a victim user visits `https://alyssa.com/`.

**2. [8 points]:** Can the adversary described above obtain this victim's cookies for `alyssa.com`? Explain how or why not.

**Answer:** Yes, because Carl's code runs in the `alyssa.com` origin.

# II SSL and TLS

**3. [12 points]:** Eric uses HTTPS (HTTP over TLS) to download a file with contents $F$ from server S, whose name and public key are well known to everyone. Can Eric provide cryptographic evidence to convince his friend Fred that the server really sent Eric the bytes for $F$, without having Fred download the file from the server himself? Explain how or why not.

**Answer:** No, because the data sent over TLS is authenticated using a MAC, with a key shared between the client and the server. A client can compute a valid MAC value for any data, not just the data that was sent by the server, so a MAC tag does not provide cryptographic proof that the data was sent by the server. This is the same reason why deniable encryption works in the context of messaging.

# III Messaging security

Ben Bitdiddle develops a messaging protocol as follows (assume A is the sender of a message and B is the recipient). B generates an ephemeral public/private key pair ($PK_e$, $SK_e$) and sends $\langle PK_e, Sign(SK_B, H(PK_e))\rangle$ to A, where $SK_B$ is B's signing key, and H is a secure hash function (say, SHA-256). When A receives B's message, she checks the signature using Verify($PK_B$, ...); assume that A knows the correct $PK_B$ corresponding to B's $SK_B$. A then sends B $\langle Enc(PK_e, m), Sign(SK_A, H(m))\rangle$, where *m* is the message she wants to send and $SK_A$ is A's signing key. B then decrypts the message and checks the signature on the hash of *m* (assume that B knows $PK_A$). In short, the protocol is:

- A $\leftarrow$ B: $PK_e$, Sign($SK_B$, H($PK_e$))
- A $\rightarrow$ B: Enc($PK_e$, m), Sign($SK_A$, H(m))

Ben gets many users to install and use his messaging application. In the following questions, assume an adversary who controls the network, and controls the computer of one of the friends of A and B.

**4. [12 points]:** Describe how an adversary can violate confidentiality – that is, how can an adversary get the contents of a message sent by A, even if the adversary is not B?

**Answer:** Adversary would need to trick B into sending the adversary the message containing a public key of the adversary's choice (e.g., adversary asks B to please copy-paste this blob in a message to the adversary). This gives the adversary Sign($SK_B$, H($PK_{adv}$)), which the adversary can then send to A as if this was the first step of the protocol between A and B. Now A will send her message for B encrypted with $PK_{adv}$.

**5. [12 points]:** Describe how an adversary can violate authenticity – that is, how can an adversary send a message to B as if the message came from A, even if A did not send that message to B?

**Answer:** The adversary talks to A in order to get a signature on some message *m* that A sent to the adversary, and now the adversary can send the same message to B, as if A sent it to B.

# IV Spectre

Ben Bitdiddle is developing a new CPU and is worried about Spectre attacks. He decides to implement the following defense strategy. The CPU tracks all of the memory addresses accessed during speculative instructions (regardless of whether they hit in the cache or not), and if the instruction ends up being aborted (due to mis-speculation), the CPU will evict all of the cache lines for addresses accessed by that instruction from the CPU caches.

**6. [12 points]:** Describe how an adversary can still mount a Spectre-like attack against a CPU with Ben's defense to learn arbitrary memory contents. Be specific.

**Answer:** The attack is much like the Spectre attack, except that, instead of looking for cache lines that seem to be present in the cache from array2, the attack looks for cache lines that got evicted from the cache from array2, which indicates that address was accessed during speculative execution.

# V Guest lectures

**7. [8 points]:** According to Max Burkhardt's guest lecture, what is the main attack vector that Figma's security engineers worry about?

**(Circle the best choice; we subtract points for incorrect answers.)**

**A.** Exploiting buffer overflow vulnerabilities in Figma's C++ server software.

**B.** Guessing the password of one of Figma's security engineers.

**C.** Phishing attacks against Figma's employees.

**D.** Fraudulent TLS certificates for Figma's domain.

**Answer:** C.

**8. [8 points]:** Based on Galen Hunt's guest lecture about Microsoft Azure Sphere, which of the following are security benefits for an IoT device manufacturer from using Azure Sphere?

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

**A. True / False** Azure Sphere ensures the Linux OS is updated to fix newly discovered security vulnerabilities.

**Answer:** Yes, this is a big point for Azure Sphere.

**B. True / False** Azure Sphere ensures that the device-specific application code does not have buffer overflow vulnerabilities.

**Answer:** No, Azure Sphere has little to do with the internals of the application code.

**C. True / False** Azure Sphere manages the private keys that identify the IoT device.

**Answer:** Yes, Azure Sphere's hardware root of trust provides the device identity, which is used to authenticate and identify the device to the cloud servers.

**D. True / False** Azure Sphere prevents compromised device-specific application code from gaining control of the entire IoT device.

**Answer:** Yes, Azure Sphere has multiple levels of isolation to prevent buggy app code from compromising the kernel, security monitor, or the Pluton chip.

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

**9. [8 points]:** Based on bunnie's guest lecture about hardware security, which of the following are correct statement about the precursor device?

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

**A. True / False** Using a physical keyboard avoids the need to trust the firmware of a touch-screen controller.

**Answer:** True.

**B. True / False** Disassembling the device allows the user to detect supply-chain attacks where an adversary adds a new chip to the motherboard.

**Answer:** True.

**C. True / False** Using an FPGA makes it impossible for an adversary to interpose on the system's I/O using a thru-silicon via (TSV) physical implant.

**Answer:** False.

**D. True / False** Using an FPGA guarantees that the RISC-V CPU has not been modified to add any backdoors.

**Answer:** We intended the answer to be False (because the FPGA does not make such a guarantee, and because there's nothing that guarantees the FPGA itself has not been tampered with or programmed correctly), but we accepted any answer because another interpretation is that, with an FPGA, the rest of the device can try to check if the bitstream being programmed into the FPGA (implementing the RISC-V CPU) has been tampered with (even if the FPGA can't guarantee that the rest of the device is doing this check correctly).

**10. [8 points]:** Based on Max Krohn's guest lecture about Zoom security, what is the most significant reason for why Zoom does not support end-to-end encryption in Zoom's web-based client?

**(Circle the best choice; we subtract points for incorrect answers.)**

**A.** It is difficult to implement cryptographic operations in a web browser using Javascript.

**B.** It is difficult to prevent a web server from sending different Javascript code to different users.

**C.** It is difficult to reliably encrypt audio and video in a web browser.

**D.** It is difficult to avoid cross-site scripting vulnerabilities.

**Answer:** B: a compromised web server can send malicious Javascript code to users, and this Javascript code can gain access to the users' private keys, without anyone else knowing about this attack. This is in contrast to the desktop or mobile apps, where the malicious code would have to be visible to many other users and potentially to security researchers or auditors.

# VI Lab 3

**11. [10 points]:** Consider the following code snippet, similar to `check-symex-int.py` from lab 3.

```
def f(x):
    if x > 500:
        return 33
    if x // 7 == 7:
        return 1234
    if x*2 == x+1:
        return 100
    return 40

def test_f():
    i = fuzzy.mk_int('i', 0)
    v = f(i)
    return v

f_results = fuzzy.concolic_execs(test_f, verbose=1)
```

Daniel is working on the lab and has **correctly** implemented everything. The call to `concolic_execs` will loop through the branches and make calls to `concolic_find_input` on several constraints. Circle all such possible constraints, then explain why the uncircled constraints are not explored. (Pay careful attention to the `Not`'s)

**A.** `And(True, Not((i > 500) == False))`

**B.** `And(And(True, (i > 500) == False), Not((i/7 == 7) == False))`

**C.**
```
And(And(And(True, (i > 500) == False), (i/7 == 7) == False),
    Not((i*2 == i + 1) == False))
```

**D.**
```
And(And(And(True, (i > 500) == False), Not((i/7 == 7) == False)),
    Not((i*2 == i + 1) == False))
```

**E.**
```
And(And(And(True, Not((i > 500) == False)), Not((i/7 == 7) == False)),
    Not((i*2 == i + 1) == False))
```

**Answer:** A, B, and C are called. D and E correspond to impossible paths: e.g., in D, it would mean that the third if statement ran even though the code went into the second if statement (and executed that return), and in E, the second and third if statements ran even though the code went into the first if statement (and executed that return).

# VII Lab 4

Ben Bitdiddle implements an additional layer of security to protect against the cookie-stealing attacks you implemented in lab 4. Ben uses the `SameSite=strict` attribute of the cookie to prevent the browser from sending this cookie along with cross-site requests. Ben also sets the `path` attribute to the zoobar login page so that the cookie can only be accessed from the login page.

**12. [10 points]:** Can an attacker still obtain the victim's cookie with these new protections, using techniques from lab 4? Either explain why this is not possible, or describe a specific attack (with HTML and Javacsript) snippets that the attacker could implement.

**Answer:** Yes: the adversary can still use an XSS attack (such as in exercises 3-4) to steal the cookie.

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

# VIII 6.858

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

**13. [2 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:** 23x Azure Sphere; 14x hardware security; 9x TCP/IP (too old); 6x Zoom; 6x information security in the real world; 5x Tor; 4x HTTPS; 3x Secure messaging; 3x SSL; 2x OWASP top 10; 1x Spectre.

Other comments: several said the Tor lecture was difficult to follow; Spectre paper was superficial and not enough detail; want additional material for the HTTPS paper; SoK papers are boring; many students were excited about the hardware security lecture, but several also commented the material was a stretch from other topics and hard to follow; Spectre should have been in the first half of the class; many said that there's too much going on in the Secure Messaging paper; several said that TCP then TLS then HTTPS felt too long; SSL 3.0 analysis paper felt dated; would be nice to have Zoom and Secure messaging back-to-back or combined given the overlap; lab 3 was just turning pseudocode into code.

**14. [2 points]:** Are there topics that we didn't cover this semester that you think 6.858 should cover in future years?

**Answer:** 34x blockchains / Bitcoin / Ethereum; 12x recent high-profile real-world attacks/vulnerabilities; 9x more details/overview of crypto; 9x more (and more in-depth) hardware security; 8x Kerberos; 8x more side channels; 5x more on web security, focusing on more modern issues and modern web frameworks; 5x security in ML and using ML in security; 4x SUNDR; 3x botnets and DDoS; 3x webauthn in more detail; 2x privacy-focused blockchains / ZK; 2x FHE; 2x MIT security; 2x SQL injection; 2x deeper dive on Tor; 2x more hands-on crypto; 2x modern secure messaging, like Signal, Skype, Facetime, and comparing them; 2x public policy; 2x post-quantum security; 2x IoT security; 2x penetration testing; 2x intrusion detection / monitoring; 2x CTFs (e.g., password cracking); 1x PKI; 1x wifi security; 1x paper on web security (nonexistant, sadly) instead of Mozilla docs; 1x better introduction to TLS; 1x web browsers on mobile phones; 1x Qubes / Tails / Whonix; 1x Cloud security; 1x social engineering; 1x security of third-party extensions; 1x peer-to-peer systems; 1x what do companies do to protect against attacks; 1x OS security; 1x PIR; 1x return-oriented programming; 1x security associated with CI/CD pipelines; 1x ransomware; 1x healthcare security; 1x low-level mitigations; 1x zero-day exploits; 1x serialization / deserialization bugs; 1x theory of what is not achievable and complexity of protocols; 1x what happens after a breach; 1x recitation on web security; 1x formal verification for security (like Komodo).

Other comments: 2x would be cool to have a lab on side channels; would be cool to have a lab on hardware security; would be cool to implement a TCP/TLS/HTTPS client; more background for hardware security lecture; would be nice for intro lecture to acknowledge broader security problems (phishing, humans) and not just bugs / buffer overflows.

# End of Quiz