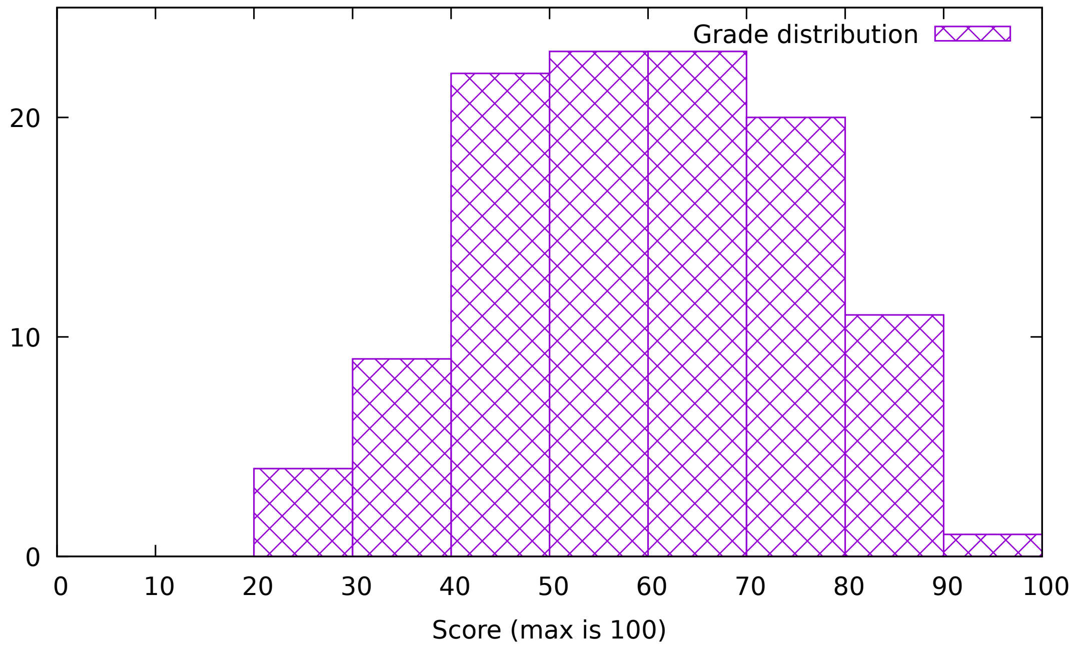
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Spring 2020

Quiz II Solutions

Mean 60.1 Standard deviation 15.9



I Paper Reading / Lecture Questions

1. [8 points]: Based on Max Burkhardt's guest lecture, which of the following statements are true? **(Circle True or False for each choice; we subtract points for incorrect answers.)**

A. True / False The cloud allows security engineers to scale making software secure.

Answer: False; because of the lack of scalability security is an attractive job.

B. True / False Network intrusion detection software is directly usable to detect intrusion of a web app.

Answer: False; because you need app-specific logs and statistics.

C. True / False Detecting missing access-control checks in Web apps can be done by keeping statistics on Direct Object References, and monitoring those statistics.

Answer: True; this was one of attacks he discussed that AirBnB stopped using the intrusion detection software he discussed.

D. True / False To develop intrusion detection for Web apps, it is important to have a strong background in machine learning techniques.

Answer: False; no background in data science/machine learning is required; for example, Max doesn't have one.

2. [8 points]: Based on Max Krohn's guest lecture about Keybase, which of the following statements are true about the Keybase chat application? **(Circle True or False for each choice; we subtract points for incorrect answers.)**

A. True / False The administrators of the servers for `keybase.com` can read the chat messages of all users.

Answer: False, because all messages end-to-end encrypted.

B. True / False The administrators of the servers for `keybase.com` can modify Alice's chat message without Bob noticing it. (You can assume Bob's knows Alice's public key).

Answer: False, because all messages have end-to-end authentication.

C. True / False By periodically publishing the Merkle root of keybase in the Bitcoin ledger, administrators of the servers for `keybase.com` cannot perform a permanent fork attack.

Answer: True.

D. True / False If a user has a single keybase device and an attacker steals that device, then the user can revoke that device by logging into `keybase.com` and remove the stolen device.

Answer: False; a user needs an existing keybase device to remote another device. Keybase doesn't store user's private keys.

II SSL/TLS

The SSL 3.0 handshake that is evaluated by Wagner and Schneier in the "Analysis of the SSL 3.0 protocol" is as follows:

```
 1. C -> S: Hello: client version, randomC, session_id, ciphers
 2. S -> C: Hello: server version, RandomS, session_id, ciphers
 3. S -> C: ServerCertificate: cert list
 4. S -> C: HelloDone
 5. C -> S: ClientKeyExchange: encrypt (pre_master_secret, PK_S)
 6. C -> S: ChangeCipherSpec
 7. C -> S: Finished, MAC({master_secret ++ msg 1,2,3,4,5}, C_key) (and encrypted)
 8. S -> C: ChangeCipherSpec
 9. S -> C: Finished MAC({master_secret ++ msg 1,2,3,4,5,7}, S_Key)  (and encrypted)
10. C -> S: encrypt(sign(data, MAC_secret), C_key)
```

3. [8 points]: The paper observes that a weakness in the protocol is that message 6 isn't included in the MAC in messages 7 and 9, and on an authenticated-only SSL connection an adversary can launch a MITM attack: delete the ChangeChipherSpec message and strip off record-layer authentication fields from Finished message and session data. The authors argue that as long as the server doesn't accept a Finished message before receiving ChangeChipherSpec message this MITM attack isn't possible. Explain briefly why this small implementation change avoids the exploit?

Answer: With the fix the server will use cryptographic authentication, and the attacker cannot remove the authentication fields from the record-layer without the server noticing because when the server recomputes the MAC it won't checkout with the data it has received.

4. [8 points]: Assume that the SSL implementation is modified as described in the previous question. Consider now a client that performs the handshake for an encrypted and authenticated SSL session with strong ciphers. Suppose the attacker doesn't delete the ChangeCipherSpec message but replaces it with a ChangeCipherSpec listing a weak cipher. Would this attack allow the attacker to get the server to accept a weak cipher and learn the content of the SSL session by breaking the weak cipher? (Briefly explain your answer.)

Answer: This question isn't quite right. In the SSL 3.0 protocol the ChangeCipherSpec message doesn't list the cipher. It is just a signal to the other party that the source switched to the pending SSL state, which includes the ciphers negotiated in the earlier steps.

If the ChangeCipherSpec would list the cipher, then the intended answer is as follows. No; if the client uses a strong encryption cipher to start with, it will be difficult for the attacker to replace message 7 with a weakly-encrypted version that the server will accept, because the attacker must decrypt the message 7 that client sent to learn its content to make a plausible new message 7 that the server will accept, but message 7 is encrypted with the strong cipher. Furthermore, messages 1 and 2 are included in the MAC, and the attacker cannot replace the ciphers listed in those messages without the server noticing.

III Spectre

Spectre V2 as described in the paper "Spectre attacks: Exploiting Speculative Execution" uses the following gadget to learn a secret `v`:

```
    if (off < sz) {
        v = array1[off]
        (*f)(v)
    }
```

The variable `f` is a pointer to a function. The idea is that the attacker trains the branch predictor so that on a miss on "sz" the processor will speculatively execute a function `f` of the attacker's choosing. (You may assume the attacker can reverse engineer how the branch predictor works.)

Assume there is a large `array2` that the attacker can read and a function `g`:

```
void g(unsigned char v) {
  v1 = array2[v*v]
}
```

5. [8 points]: Briefly explain if `(*f)()` invoked `g`, is `g` suitable for learning the value `v`? (You may assume that the attacker has set up everything up correctly to handle alignment, cache lines, etc. correctly and that `array2` is large enough.)

Answer: We accepted 2 possible answers:

Yes. To check if `v` has the value 2, for example, the attacker measures the 4th entry of `array2`, and so on. Each $v * v$ is unique.

No. Small values of `v` will access different entries in `array2`, but still touch the same cache line so it won't be possible to distinguish them.

Consider the following function `h`:

```
uint32 z(unsigned char v) {
  // Efficiently return a pretty strong
  // cryptographic hash of v.
}

void h(unsigned char v) {
  v1 = array2[z(v)]
}
```

6. [8 points]: Briefly explain if `h` is suitable for learning the value `v`, and, if so, how the attacker would learn the value of `v`? (You can make the same assumptions as in the previous question.)

Answer: Yes. To check if `v` has the value 2, the attacker measures the `z(2)`'th entry of `array2`, and so on. Since `z` is a strong cryptographic hash function `z(v)` is likely to be unique.

IV Messaging Security

Consider the following messaging protocol. If sender A (with private key SK_A) wants to send message m to recipient B (with public key PK_B), A generates a fresh symmetric key K, and transmits the following to B:

- Encrypt(PK_B, K)
- Sign(SK_A, K)
- MAC(K, m)
- m

7. [7 points]: Does the protocol provide authenticity (that is, B should be convinced that the message m came from A, assuming A is honest)? Explain why or why not.

Answer: No, the protocol does not provide authenticity if A ever sends a message to someone other than B. If A ever sends a message to some other user C, then C can take the shared key K signed by A, and send it to B, along with an arbitrary message of C's choice, and a correctly computed MAC value (because C knows K).

8. [7 points]: Does the protocol provide deniability (that is, there is no way for B to provide cryptographic evidence that A must have sent m)? Explain why or why not.

Answer: The protocol provides deniability. B can compute MAC(K,m') for any other message m', so C would not be able to determine whether any particluar m came from A or not.

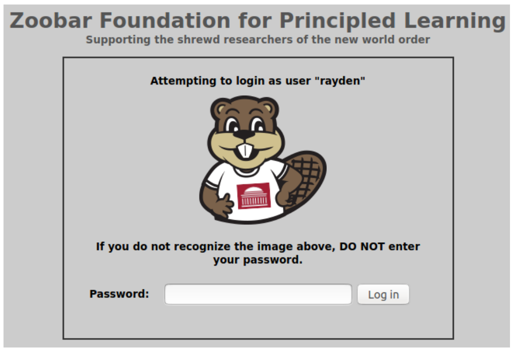
V Web Security

This question is based on the Zoobar lab assignments, the web security lecture, and the “SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements” paper.

Ben Bitdiddle runs a restaurant called Ben’s Bites in downtown Boston. Due to the coronavirus lockdown, he decides to set up a website for business continuity, where customers can make takeout and delivery orders. He needs to set up the website quickly, so he uses the entirety of the 6.858 Zoobar application as a template, and parks it under the domain name `bensbites.com`.

To start off, Ben patched all cross-site scripting vulnerabilities in the Zoobar application. In addition, the online login process has been modified to include the following "Secret Image" security control:

- At Registration: The user has to choose ONE (1) image out of a large library of preset images, which will be associated with user’s username. The choice of image is kept secret.
- At Login: Initially, the user is required to type in ONLY their username. The website then shows the password field and an image. The user is instructed to enter their password IF AND ONLY IF they see the image they chose at the time of registration. (See Figure 1)


Zoobar Foundation for Principled Learning
Supporting the shrewd researchers of the new world order

Attempting to login as user "rayden"

If you do not recognize the image above, DO NOT enter your password.

Figure 1: Example `bensbites.com` login page after entering username.

Feature	Details
Public/Private Key	3072-bit RSA
Protocol Support	$\geq$ TLS 1.2
Revocation Information	CRL, OCSP
OCSP Stapling	NOT enabled
Public Key Pinning (HPKP)	NOT enabled
HTTPS-only Pinning (HSTS)	NOT enabled
`TLS_FALLBACK_SCSV`	ENABLED
TLS Compression	NOT enabled

Figure 2: Some TLS configuration details about `bensbites.com`.

Ben is very concerned about security, so he enforces TLS on all connections to the site as shown in Figure 2. Assume that the TLS cipher suites supported by `bensbites.com` are **cryptographically secure**, and that the certificate was issued by a well-known and trusted CA.

Rayden is a loyal customer of Ben’s Bites and has created an account on `bensbites.com`. One day, he decides to order lunch on the website. Unfortunately for Rayden, Noah carried out a man-in-the-middle (MITM) attack and stole Rayden’s password.

9. [8 points]: Briefly describe how and why Noah is able to carry out an MITM attack to steal Rayden's password when Rayden browses to `bensbites.com`.

Answer: HSTS is not enabled, so browsers will not enforce TLS connection on subsequent visits to `bensbites.com` [1pt]. MITM attack possible by "stripping" TLS layer away so that Noah can see HTTP traffic in the clear: Rayden $\leftrightarrow$ (HTTP) Noah (HTTPS) $\leftrightarrow$ (HTTPS) `bensbites.com` [1pt]

You tell Ben Bitdiddle about the attack in Question 9, and he patches it so that it no longer works. Not to be outplayed, Noah creates a malicious site `evil.com` and manages to steal Rayden's password for a second time via a phishing attack!

10. [8 points]: How is Noah able to carry out a phishing attack on Rayden using `evil.com`, despite the fact that Rayden entered his password only when he saw the secret image he chose at the time of registration?

Answer: Proctor the connection via `evil.com`. When Rayden enters his username at `evil.com`, `evil.com` sends the username to `bensbites.com` and fetches the secret image, which is relayed to Rayden. Rayden sees that the image is the secret image that he selected at registration and proceeds to enter his password, which is captured by Noah.

In response to Noah's phishing attack, Ben modifies the "Secret Image" security control to be more secure. Here is the modified control with the changes in **bold**:

- At Registration: The user has to choose ONE (1) image out of a large library of preset images, which will be associated with user's username. The choice of image is kept secret. **A browser cookie named `_secimg_nonce` will be set on the user's browser for `bensbites.com` with a 64-byte string value initialized from a CSPRNG, and the cookie value is bound to the user's username in the database.**
- At Login: Initially, the user is required to type in ONLY their username. **The user's browser then sends the `_secimg_nonce` cookie to the server which verifies that the cookie value matches the bound value in the database. If the cookie is not set or it does not match the bound value, then an error is raised and the login process halts.** Otherwise, the website shows the password field and an image. The user is instructed to enter their password IF AND ONLY IF they see the image they chose at the time of registration. (See Figure 1)

11. [8 points]: Does Ben's modified security control mitigate Noah's phishing attack in Question 10? **Explain why.** Be sure to explicitly state any assumptions for your security model.

Answer: Yes. [1pt] SOP prevents `evil.com` from accessing cookies set for `bensbites.com`, so it cannot find a valid `_secimg_nonce` value. [1pt] Valid string values are long and random which makes brute-forcing infeasible for threat actors. [1pt] In addition, while `evil.com` may direct the user's browser to send requests to `bensbites.com`, SOP prevents `evil.com` from reading the responses from `bensbites.com`. [1pt]

In order to better utilize his server network bandwidth, Ben decides to enable TLS compression using the DEFLATE compression algorithm, which replaces repeated byte sequences with a pointer to the first instance of that sequence. The longer the repeated sequences, the higher the compression ratio. For example, the string "`Ooooooooompa Looooooooompa`" compresses to "`Ooooooooompa L(-14,12)`" since the repeated "`ooooooooompa`" sequence is 12 characters long and located at relative offset -14.

When Rayden successfully logs in to `bensbites.com`, Rayden's browser sends requests in the following format (shown before any encryption:

```
GET / HTTP/1.1
Host:  bensbites.com/zoobar/index.cgi
Cookie:  PyZoobar=c30e8a3b8ed2323bccfaf2224a35716d
...(other headers, HTML body)...
```

Here, the `PyZoobar` cookie is a session token that uniquely identifies a user to `bensbites.com`.

12. [8 points]: Describe how Noah, who is passively eavesdropping on encrypted TLS traffic between Rayden and `bensbites.com`, can steal Rayden's `PyZoobar` session cookie when Rayden visits `evil.com`. Assume that Rayden logged on to `bensbites.com` minutes before visiting `evil.com`.

Answer: `evil.com` can direct Rayden's browser to send `GET` requests to `bensbites.com`, making use of the compression ratio information leakage from the DEFLATE compression. We start with the request `https://bensbites.com/zoobar/index.cgi?PyZoobar=`. Since the query string is in the body of the GET request, brute-forced characters at the end that correspond to Rayden's `PyZoobar` cookie will cause a decrease in the size of TLS encrypted data.

```
GET / HTTP/1.1
Host:  bensbites.com/zoobar/index.cgi
Cookie:  PyZoobar=c30e8a3b8ed2323bccfaf2224a35716d
...
```

`PyZoobar=a` ← size of encrypted data decreases when we hit 'c', which is part of the actual cookie value, during which we move on to the 2nd character

We carry out this brute-force attack, moving on to the next character whenever we observe such a decrease, and terminating when we reach the known length of the cookie value.

VI SUNDR

Ben Bitdiddle is trying to optimize the serialized SUNDR protocol, as described in section 3.3 of the SUNDR paper. Ben thinks he can leverage the fact that the serialized SUNDR protocol allows only one user to make a change at a time. Instead of storing a full version vector in the version structure (see Figure 3 in the SUNDR paper), Ben's version structure stores just two elements of the version vector: the version counter for the user that created the version structure, and the version counter for the previous user that made a change. The rules for validating version vectors remain the same.

For example, if Alice made two changes, and then Bob made a change, Bob's version vector would contain 2 for Alice and 1 for Bob. If Bob made three more changes in a row, he would still keep Alice in his version vector, and Bob's version structure would contain 2 for Alice and 4 for Bob.

13. [10 points]: Does Ben Bitdiddle's variation provide fork consistency? Explain why or why not. Assume that only one user performs an operation at a time.

Answer: No, Ben's SUNDR variant does not provide fork consistency. Assume the system has four users (A, B, C, and D) with empty version structures to begin with. A and B make a series of alternating changes (issuing new version structures), so that their version structures have version vectors (A:10,B:10) and (A:10,B:11) respectively. C makes a change, so its version vector is now (B:11,C:1). Now A and B observe C's change and make another series of alternating changes, so their structures have version vectors (A:20,B:20) and (A:20,B:21). At this point, if D looks at the file system, an adversary could give it A's and B's latest version structures, together with the empty version structure for C. This would look OK to D, but would violate fork consistency, because A and B observed C's change, but D does not.

VII 6.858

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

14. [2 points]: Are there any papers in the second part of the semester that you think we should definitely remove next year? If not, feel free to say that.

Answer: 8x Messaging; 7x HTTPS/Certificates; 7x Spectre; 7x Tangled Web; 5x SSL; 5x Keybase (wanted a better paper); 5x SUNDR; 4x TCP 1x Max Burkhardt; 1x Tor.

Also, 2x Didn't like Q&A guest lecture format, and 1x Present crypto basics sooner (e.g., before Komodo).

15. [2 points]: Are there topics that we didn't cover this semester that you think 6.858 should cover in future years?

Answer: 15x real-world attacks; 12x cryptocurrency/blockchain; 10x malware, viruses, botnets; 4x IoT; 4x distributed systems; 4x hardware security; 4x zoom; 4x web attacks; 2x modern buffer overflows; 2x DNS; 2x crypto; 2x penetration testing; 2x more side-channel attacks (power, crypto); 2x practical network attacks (wifi); 2x kerberos; 1x autonomous vehicles and robots; 1x where is security going in 5-10 years; 1x election security; 1x corporate security; 1x certificate transparency; 1x practical symbolic execution; 1x database security; 1x practical OS security (linux, windows, macos); 1x cpu bugs; 1x ml security; 1x math proofs

End of Quiz