CS155 Final exam

George Charles Davis

TOTAL POINTS

83 / 100

QUESTION 1

1 True or False 9 / 10

- **0 pts** Correct

✓ **- 1 pts** *(a) incorrect*

- **1 pts** (b) incorrect
- **1 pts** (c) incorrect
- **1 pts** (d) incorrect
- **1 pts** (e) incorrect
- **1 pts** (f) incorrect
- **1 pts** (g) incorrect
- **1 pts** (h) incorrect
- **1 pts** (i) incorrect
- **1 pts** (j) incorrect

QUESTION 2

Short answer 24 pts

2.1 Cookie deletion 4 / 4

✓ **- 0 pts** *Correct*

- **2 pts** Partially correct (doesn't specify that cookie's expiration date can be set to some time in the past).
- **3 pts** Significant error.
- **4 pts** Not attempted.
- **1 pts** Click here to replace this description.

2.2 ProPolice 4 / 4

✓ **- 0 pts** *Correct*

- **2 pts** Partially correct (said something true about stack protections but didn't describe iterative extraction, or claimed that canary cannot be extracted but still describes iterative extraction process).
- **3 pts** Incorrect (states that canary cannot be extracted, doesn't mention extraction process).
- **4 pts** Not attempted.

2.3 Confused Deputy 4 / 4

✓ **- 0 pts** *Correct*

- **1 pts** Partially correct (describes intent, doesn't explain that it can be passed in as a callback).
- **2 pts** Partially correct (said something true about Android security model but didn't describe passing malicious code through callback).
- **4 pts** Not attempted.

2.4 Certificate Transparency 4 / 4

✓ **- 0 pts** *Correct*

(CT does not prevent attacks, but can allow for discovering that a rogue certificate was issued afterwards)

- **2 pts** Partially incorrect answer (Either one of claiming that the attack would be prevented OR an incorrect explanation on CT.)
- **3 pts** Significant error (Incorrectly claims two or more CT log servers need to be verified and

thus prevents the attack, since one compromised CA can publish onto multiple servers.)

- 4 pts Incorrect answer (Both claimed that the attack would be prevented and wrong explanation, or was not attempted)

2.5 TOCTOU Bug **4 / 4**

✓ **- 0 pts** *Correct (TOCTOU = Time of Check Time of Use Bug. Explains bug or shows example)*

- 3 pts Does not correctly explain security vulnerability

- 4 pts Fully incorrect or not attempted.

2.6 Apple iPhone Security **0 / 4**

- 0 pts Correct (Firmware signed by Apple's signing key, and Apple's public key is embedded in the secure processor)

- 2 pts Partially Incorrect (No mention of using a cryptographic key)

✓ **- 4 pts** *Incorrect or not attempted.*

QUESTION 3

Memory tagging 20 pts

3.1 **4 / 4**

✓ **- 0 pts** *Correct*

- 4 pts No answer.

- 2 pts Not quite. The hardware memory system will throw an exception.

- 2 pts When the string is 32 bytes we have a buffer overflow, and we want the hardware to raise an exception.

- 2 pts Having the process crash is better than suffering a buffer overflow attack.

- 1 pts Granule tags are not overwritten on an overflow. The tags are stored as metadata in the memory system.

3.2 **4 / 4**

✓ **- 0 pts** *Correct*

- 4 pts No answer.

- 2 pts This is a user-after-free vulnerability.

3.3 **2 / 2**

✓ **- 0 pts** *Correct*

- 2 pts No answer.

- 2 pts The granule is tagged 0x2, but the pointer s is still tagged 0xA, so the memory system will raise an exception.

3.4 **6 / 6**

✓ **- 0 pts** *Correct*

- 2 pts Presumably delete will change the tag to a random *different* tag from the existing tag.

- 4 pts There are still cases when a use-after-free may not raise an exception.

- 2 pts This is not a use-after-free. The pointer points to a correctly allocated object.

- 6 pts No answer

- 4 pts How?

- 2 pts This is not use-after-afree.

3.5 **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts An attacker can mount an overflow attack within a single stack frame.

- 2 pts Why?

- 4 pts No answer

QUESTION 4

Text fragments 16 pts

4.1 **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts SOP not specified

- 2 pts Incorrect

4.2 **2 / 2**

✓ **- 0 pts** *Correct*

- 0.5 pts Incorrect / No URL

- 1 pts Incorrect Test

- 1 pts Did not use Web text fragment

- 2 pts Incorrect

4.3 **1 / 2**

- 0 pts Correct: mentions (or clearly alludes to) same origin policy's effect on DOM access

✓ **- 1 pts** *Minor Error: does not clearly mention or allude to same origin policy, but describes isolation between evil.com and bank.com that would result from SOP; or other minor inaccuracy*

- 2 pts Incorrect: does not mention or allude to same origin policy at all, or other major inaccuracy

- 2 pts Missing

Chrome does not need to change scrolling behavior on cross-origin requests since the DOM SOP would prevent ScrollTop from being read by a cross-origin site.

4.4 **4 / 4**

✓ **- 0 pts** *Correct*

- 1.5 pts Does not clearly state that evil.com should read the scrollTop property of one of its own components, since scrolling the frame scrolls the whole page (and it cannot access the frame's scrollTop due to SOP)

- 1.5 pts Does not clearly state that attacker should search for text by loading web text fragment link (either through iframe src or in javascript)

- 1 pts Implies user must still click web text fragment link or take some other action for attack to work; this is not true since the web text fragment can be set to the source of the iframe

- 1 pts Contains irrelevant and/or inaccurate info

- 2 pts Falsely states (or implies) that evil.com can read DOM tree of embedded bank.com iframe (this would violate the SOP)

- 4 pts Incorrect and/or carries out different attack than that in (b)

- 4 pts Missing

4.5 **2 / 2**

✓ **- 0 pts** *Correct*

- 2 pts Not Attempted

- 1 pts Slightly Incorrect

4.6 **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts No mention of making search take constant time

- 4 pts Not Attempted

QUESTION 5

Secure communication 14 pts

5.1 **1.5 / 3**

- 0 pts Correct

✓ **- 1.5 pts** *Correct flaw, incorrect protocol fix*

- 3 pts Blank / Incorrect

5.2 **0 / 3**

- 0 pts Correct response; identifies the TCP handshake

- 1 pts Solid response, but with a minor mistake, misunderstanding, or significant lack of clarity

✓ **- 3 pts** *Incorrect or blank*

5.3 **0 / 2**

- 0 pts Correctly identifies risk of compromising other hosts in the organization's networks

✓ **- 2 pts** *Incorrect (e.g., assumes spoofing ability) or blank*

5.4 **0 / 3**

- 0 pts Correct

✓ **- 3 pts** *Not Attempted*

- 1 pts No mention of BGP or DNS/Domain Hijack

5.5 **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts No explanation

- 2 pts No example of information that is not protected

- 3 pts No answer

QUESTION 6

Tor network 16 pts

6.1 **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts Incorrect explanation, incomplete explanation, or no explanation

- 3 pts Incorrect answer

6.2 **3 / 3**

✓ **- 0 pts** *Correct*

- 2.5 pts Incorrectly states that attacker can view traffic content or destination

- 3 pts No answer.

6.3 **1.5 / 3**

- 0 pts Correct

- 1.5 pts Information incorrect

✓ **- 1.5 pts** *Protection incorrect*

- 1 pts Information partially correct

- 1 pts Protection partially correct

6.4 **3 / 3**

✓ **- 0 pts** *Correct*

- 3 pts Incorrect

- 1.5 pts Partially correct

6.5 **4 / 4**

✓ **- 0 pts** *Correct*

- 4 pts Incorrect

- 2 pts One of the sides incorrect

1. *(10 points)* ..... True or False

For each question, please write `T` or `F` in the space provided. No explanation needed.

F (a) Heap spraying is a technique used to exploit memory corruption in the heap.

T (b) A format string vulnerability lets the attacker read/write locations in memory that it should not have access to.

N (c) Marking the stack and the heap as non-executable memory prevents control hijacking attacks. → bypassed by ROP

Y (d) Consider the Spectre attack discussed in lecture 12. If memory caching were turned off, so that every memory read and write would go to main memory, then the attack would not be possible.

T (e) A cookie with no explicitly defined `Domain` will only be sent to the exact domain that set the cookie.

F (f) CORS performs a pre-flight check for all `POST` requests made through Javascript.

F (g) The purpose of Server Name Indication (SNI) field in the HTTPS `client-hello` message is to hide the domain name that the browser is requesting from the server.

F (h) DNSSEC protects users against DNS rebinding attacks.

T (i) Amplification-based DOS attacks can be prevented by all ISPs deploying ingress filtering for all their direct customers.

F (j) SMS-based two-factor authentication protects against phishing attacks.

1 True or False **9 / 10**

- **0 pts** Correct

✓ **- 1 pts** *(a) incorrect*

- **1 pts** (b) incorrect
- **1 pts** (c) incorrect
- **1 pts** (d) incorrect
- **1 pts** (e) incorrect
- **1 pts** (f) incorrect
- **1 pts** (g) incorrect
- **1 pts** (h) incorrect
- **1 pts** (i) incorrect
- **1 pts** (j) incorrect

2. *(24 points)* ..... Questions from all over with a short answer

(a) *(4 points)* Suppose a web site sets a cookie called `temp` in the browser. How can the web site instruct the browser on a subsequent visit to delete the cookie `temp`?

Your answer:

set an expiration date with "Expires:" attribute or "Max Age" attribute with a short TTL so that the browser deletes the cookie before the user can revisit the website

=> exp date can also be set in the past to remove the cookie

(b) *(4 points)* Suppose a web server has an exploitable memory corruption vulnerability. The web server was compiled using ProPolice to use random stack canaries. The web server is configured to restart automatically after a crash, without changing the canary value. Can an adversary extract the stack canary value from the server? If so, explain how. If not, explain why not.

Your answer:

yes; if the values don't change, the attacker can write into the canary one byte at a time; if the server doesn't crash, they know they got the right byte & move on to the next one. eventually they can determine the entire canary and move on to overwriting the pointers

(c) *(4 points)* A *confused deputy* vulnerability refers to a service that runs in a "high" security context. An attacker running in a "low" security context can call this service with an unexpected argument and cause an invalid operation to take place in the high security context. Slide 18 in lecture 18 gave an example of confused deputy using Android intents. Explain how the code from the lecture running in a victim app on the phone can be exploited by a malware app running on the same phone.

Your answer:

a malicious app can call an intent on the victim app with a callback reference that is internal to the victim app; the OS sees that the victim app "generated" the new intent and effectively gives the attacker to call internal-only intents (native to the victim app)

2.1 Cookie deletion **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts Partially correct (doesn't specify that cookie's expiration date can be set to some time in the past).

- 3 pts Significant error.

- 4 pts Not attempted.

- 1 pts Click here to replace this description.

2. *(24 points)* ..... Questions from all over with a short answer

(a) *(4 points)* Suppose a web site sets a cookie called `temp` in the browser. How can the web site instruct the browser on a subsequent visit to delete the cookie `temp`?

Your answer:

set an expiration date with "Expires:" attribute or "Max Age" attribute with a short TTL so that the browser deletes the cookie before the user can revisit the website

=> exp date can also be set in the past to remove the cookie

(b) *(4 points)* Suppose a web server has an exploitable memory corruption vulnerability. The web server was compiled using ProPolice to use random stack canaries. The web server is configured to restart automatically after a crash, without changing the canary value. Can an adversary extract the stack canary value from the server? If so, explain how. If not, explain why not.

Your answer:

yes; if the values don't change, the attacker can write into the canary one byte at a time; if the server doesn't crash, they know they got the right byte & move on to the next one. eventually they can determine the entire canary and move on to overwriting the pointers

(c) *(4 points)* A *confused deputy* vulnerability refers to a service that runs in a "high" security context. An attacker running in a "low" security context can call this service with an unexpected argument and cause an invalid operation to take place in the high security context. Slide 18 in lecture 18 gave an example of confused deputy using Android intents. Explain how the code from the lecture running in a victim app on the phone can be exploited by a malware app running on the same phone.

Your answer:

a malicious app can call an intent on the victim app with a callback reference that is internal to the victim app; the OS sees that the victim app "generated" the new intent and effectively gives the attacker to call internal-only intents (native to the victim app)

2.2 ProPolice **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts Partially correct (said something true about stack protections but didn't describe iterative extraction, or claimed that canary cannot be extracted but still describes iterative extraction process).

- 3 pts Incorrect (states that canary cannot be extracted, doesn't mention extraction process).

- 4 pts Not attempted.

2. *(24 points)* ..... Questions from all over with a short answer

(a) *(4 points)* Suppose a web site sets a cookie called `temp` in the browser. How can the web site instruct the browser on a subsequent visit to delete the cookie `temp`?

Your answer:

set an expiration date with "Expires:" attribute or "Max Age" attribute with a short TTL so that the browser deletes the cookie before the user can revisit the website

=> exp date can also be set in the past to remove the cookie

(b) *(4 points)* Suppose a web server has an exploitable memory corruption vulnerability. The web server was compiled using ProPolice to use random stack canaries. The web server is configured to restart automatically after a crash, without changing the canary value. Can an adversary extract the stack canary value from the server? If so, explain how. If not, explain why not.

Your answer:

yes; if the values don't change, the attacker can write into the canary one byte at a time; if the server doesn't crash, they know they got the right byte & move on to the next one. eventually they can determine the entire canary and move on to overwriting the pointers

(c) *(4 points)* A *confused deputy* vulnerability refers to a service that runs in a "high" security context. An attacker running in a "low" security context can call this service with an unexpected argument and cause an invalid operation to take place in the high security context. Slide 18 in lecture 18 gave an example of confused deputy using Android intents. Explain how the code from the lecture running in a victim app on the phone can be exploited by a malware app running on the same phone.

Your answer:

a malicious app can call an intent on the victim app with a callback reference that is internal to the victim app; the OS sees that the victim app "generated" the new intent and effectively gives the attacker to call internal-only intents (native to the victim app)

2.3 Confused Deputy **4 / 4**

✓ **- 0 pts** *Correct*

- 1 pts Partially correct (describes intent, doesn't explain that it can be passed in as a callback).

- 2 pts Partially correct (said something true about Android security model but didn't describe passing malicious code through callback).

- 4 pts Not attempted.

(d) (*4 points*) Suppose an attacker successfully steals the signing key of a certificate authority (CA). This lets the attacker issue certificates for any domain of its choice. Does Certificate Transparency (CT) prevent the attacker from using such a rogue certificate to carry out a man-in-the-middle (MiTM) attack against a web site that the attacker does not control? In particular, will a browser that only accepts CT certificates, accept the rogue certificate and establish an HTTPS connection with the attacker's site? Explain why or why not. Recall that the attacker can validly register its rogue certificate with a CT log, as required for CT compliance.

Your answer:

yes; CT doesn't detect if a certificate was mis-issued by a rogue certificate authority

(e) (*4 points*) What is a TOCTOU bug and why is it a security vulnerability?

Your answer:

it's a race condition bug; the check of condition occurs first, but the status of "condition" can change after the check but before doSomething() runs; this can allow an attacker to run code they shouldn't have access to

```
if (condition) {
    doSomething();
}
```

2.4 Certificate Transparency **4 / 4**

✓ **- 0 pts** *Correct*

(CT does not prevent attacks, but can allow for discovering that a rogue certificate was issued afterwards)

- 2 pts Partially incorrect answer (Either one of claiming that the attack would be prevented OR an incorrect explanation on CT.)

- 3 pts Significant error (Incorrectly claims two or more CT log servers need to be verified and thus prevents the attack, since one compromised CA can publish onto multiple servers.)

- 4 pts Incorrect answer (Both claimed that the attack would be prevented and wrong explanation, or was not attempted)

(d) (*4 points*) Suppose an attacker successfully steals the signing key of a certificate authority (CA). This lets the attacker issue certificates for any domain of its choice. Does Certificate Transparency (CT) prevent the attacker from using such a rogue certificate to carry out a man-in-the-middle (MiTM) attack against a web site that the attacker does not control? In particular, will a browser that only accepts CT certificates, accept the rogue certificate and establish an HTTPS connection with the attacker's site? Explain why or why not. Recall that the attacker can validly register its rogue certificate with a CT log, as required for CT compliance.

Your answer:

yes; CT doesn't detect if a certificate was mis-issued by a rogue certificate authority

(e) (*4 points*) What is a TOCTOU bug and why is it a security vulnerability?

Your answer:

it's a race condition bug; the check of condition occurs first, but the status of "condition" can change after the check but before doSomething() runs; this can allow an attacker to run code they shouldn't have access to

```
if (condition) {
    doSomething();
}
```

2.5 TOCTOU Bug **4 / 4**

✓ **- 0 pts** *Correct (TOCTOU = Time of Check Time of Use Bug. Explains bug or shows example)*

- 3 pts Does not correctly explain security vulnerability

- 4 pts Fully incorrect or not attempted.

(f) (*4 points*) **iPhones use a secondary secure processor that isolates secure components of the phone (e.g., Apple Wallet) and exposes only a hardened API to the primary processor. When the phone boots, the secure processor loads its firmware (i.e., the code it executes) from non-volatile memory. How do you think Apple prevents an attacker from replacing the firmware stored in non-volatile memory with malicious firmware that leaks sensitive data?**
Hint: **Think of a cryptographic mechanism that might help. Your solution should not prevent Apple from periodically installing firmware updates on the phone.**

Your answer:

2.6 Apple iPhone Security **0 / 4**

- 0 pts Correct (Firmware signed by Apple's signing key, and Apple's public key is embedded in the secure processor)

- 2 pts Partially Incorrect (No mention of using a cryptographic key)

✓ **- 4 pts** *Incorrect or not attempted.*

3. *(20 points)* ..... Memory tagging

ARM recently added a new security capability to ARM processors called *Memory Tagging Extension* or MTE. We discussed MTE in Lecture 3. Recall that MTE divides physical memory into granules, where each granule is 16 bytes long. Each 16-bytes granule is "tagged" by a 4-bit tag that is stored as metadata for the granule. New ARM instructions let an application set the 4-bit tag for a granule. In addition, the four top bits of every pointer contain a 4-bit tag. When a pointer is used to read or write an address in memory, the ARM memory management hardware checks that the 4-bit tag embedded in the pointer is equal to the 4-bit tag associated with the granule being accessed. If not, the memory management hardware raises an exception, which may cause the application to crash.

(a) *(4 points)* Consider the following C++ code:

```
1:   int func(char *inp) {
2:     char *s = new char(15);
3:     strcpy(s, inp);
4:     // do something
     }
```

Suppose that `malloc` on line (2) allocates the buffer `*s` on a single granule on the heap, and assigns a random 4-bit tag, say `0xA`, to that granule. The next 16-byte granule is assigned a different random 4-bit tag, say `0x5`. Moreover, `malloc` tags the pointer `s` with the 4-bit tag `0xA`. When MTE is enabled, explain what happens when the function `func` is called

- with `inp` pointing to a string of length 12 bytes;
- with `inp` pointing to a string of length 32 bytes.

Is this a desirable behavior?

Your answer:

1) MTE checks that the tag for inp matches the memory tag for its respective granule, and also checks that the tag for s' matches the tag of the granule being written to for each byte written (which it does); func() works properly

2) when strcpy() starts to write into the granule with the tag 0x5, MTE notices the tag mismatch and prevents the buffer overflow

⇒ prevents overflows ⇒ desireable

3.1 **4 / 4**

✓ **- 0 pts** *Correct*

- 4 pts No answer.

- 2 pts Not quite. The hardware memory system will throw an exception.

- 2 pts When the string is 32 bytes we have a buffer overflow, and we want the hardware to raise an exception.

- 2 pts Having the process crash is better than suffering a buffer overflow attack.

- 1 pts Granule tags are not overwritten on an overflow. The tags are stored as metadata in the memory system.

(b) (*4 points*) Suppose the subsequent code in the function `func` does

```
5:      delete [] s;              // free the string *s
6:      Name *p = new Name;       // allocate an object on the heap
7:      s[2] = 'a';
```

→ memory gets retagged

memory is retagged again to match tag of p

Moreover, suppose that the object `*p` in line (6) is allocated in the same position as the recently freed buffer `*s`. What is the name of the potential vulnerability that line (7) introduces? Briefly explain how an attacker might be able to exploit this vulnerability. You can make whatever assumptions you need about the object `*p` and its location in memory.

↳ in the heap

Your answer:

1) use after free

2) assuming that s retains the same tag as the memory allocated by p, the MTE will still allow the attacker to access this memory using s.

(c) (*2 points*) Now, suppose that the `delete [] s` operation on line (5) frees the memory at `*s` on the heap and also changes the 4-bit tag assigned to that granule from `0xA` to a different random 4-bit tag, say `0x2`. For simplicity, assume that line (6) does not change the 4-bit tag assigned to the granule. Explain what will happen when the code from part (b) is executed. Is this the desired behavior?

Your answer:

MTE will prevent use after free because s still has the old tag (0xA) and MTE will catch when it tries to write to memory tagged with 0x2 ⇒ desirable

3.2 **4 / 4**

✓ **- 0 pts** *Correct*

- 4 pts No answer.

- 2 pts This is a user-after-free vulnerability.

(b) (*4 points*) Suppose the subsequent code in the function `func` does

```
5:      delete [] s;              // free the string *s
6:      Name *p = new Name;       // allocate an object on the heap
7:      s[2] = 'a';
```

→ memory gets retagged

memory is retagged again to match tag of p

Moreover, suppose that the object `*p` in line (6) is allocated in the same position as the recently freed buffer `*s`. What is the name of the potential vulnerability that line (7) introduces? Briefly explain how an attacker might be able to exploit this vulnerability. You can make whatever assumptions you need about the object `*p` and its location in memory.

↳ in the heap

Your answer:

1) use after free

2) assuming that s retains the same tag as the memory allocated by p, the MTE will still allow the attacker to access this memory using s.

(c) (*2 points*) Now, suppose that the `delete [] s` operation on line (5) frees the memory at `*s` on the heap and also changes the 4-bit tag assigned to that granule from `0xA` to a different random 4-bit tag, say `0x2`. For simplicity, assume that line (6) does not change the 4-bit tag assigned to the granule. Explain what will happen when the code from part (b) is executed. Is this the desired behavior?

Your answer:

MTE will prevent use after free because s still has the old tag (0xA) and MTE will catch when it tries to write to memory tagged with 0x2 ⇒ desirable

3.3 **2 / 2**

✓ **- 0 pts** *Correct*

- 2 pts No answer.

- 2 pts The granule is tagged 0x2, but the pointer s is still tagged 0xA, so the memory system will raise an exception.

(d) (*6 points*) Will re-tagging on free, as explained in part (c), prevent all the vulnerabilities of the type you identified in part (b)? If so, explain why. If not, give a short code snippet that contains a vulnerability of the type in part (b) which is still vulnerable despite the use of re-tagging on free. Briefly explain why your code snippet is vulnerable.

Your answer:

not quite; because the new tag value is random, the new granule tag could randomly match the tag of the old pointer, (1/16 chance) allowing S to access the memory

code would be the same; just run it over and over again and eventually the new granule tag will match the old pointer's tag

(e) (*4 points*) Suppose that all granules in a single stack frame are tagged with the same tag. However, adjacent stack frames are tagged with different tags. Will this prevent buffer overflow attacks on the stack? If so, explain why. If not, briefly explain how an attack might work.

Your answer:

no; the attacker can overwrite into the return pointer and point it to somewhere in the caller's stack frame (so that the ptr and memory tags match) where the attacker previously put malicious code

3.4 **6 / 6**

✓ **- 0 pts** *Correct*

- 2 pts Presumably delete will change the tag to a random *different* tag from the existing tag.

- 4 pts There are still cases when a use-after-free may not raise an exception.

- 2 pts This is not a use-after-free. The pointer points to a correctly allocated object.

- 6 pts No answer

- 4 pts How?

- 2 pts This is not use-after-afree.

(d) (*6 points*) Will re-tagging on free, as explained in part (c), prevent all the vulnerabilities of the type you identified in part (b)? If so, explain why. If not, give a short code snippet that contains a vulnerability of the type in part (b) which is still vulnerable despite the use of re-tagging on free. Briefly explain why your code snippet is vulnerable.

Your answer:

not quite; because the new tag value is random, the new granule tag could randomly match the tag of the old pointer, (1/16 chance) allowing S to access the memory

code would be the same; just run it over and over again and eventually the new granule tag will match the old pointer's tag

(e) (*4 points*) Suppose that all granules in a single stack frame are tagged with the same tag. However, adjacent stack frames are tagged with different tags. Will this prevent buffer overflow attacks on the stack? If so, explain why. If not, briefly explain how an attack might work.

Your answer:

no; the attacker can overwrite into the return pointer and point it to somewhere in the caller's stack frame (so that the ptr and memory tags match) where the attacker previously put malicious code

3.5 **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts An attacker can mount an overflow attack within a single stack frame.

- 2 pts Why?

- 4 pts No answer

4. (16 points) ..... Web text fragments and covert channels

Chrome 80 added a feature, called Web text fragments, that allows one to encode a search string in a URL. When the browser opens that URL, if the search string is found on the page, the page automatically scrolls to that location, and the string is highlighted. For example, to open the CS155 page at the "Projects" section, use the URL

`https://cs155.stanford.edu/#:~:text=Projects`

Feel free to try it out. If the string is not found on the page, no scrolling takes place.

As we saw throughout the course, benign-looking features can have significant security implications. In this question we will explore the risks associated with the Web text fragments feature.

Consider a page at `evil.com` that contains a link to a fictional site

```
<a href="https://bank.com" target="_blank"> Click me </a>
```

Suppose Bob visits `evil.com` and clicks on this link. The result is a new browser window at `bank.com`.

(a) (2 points) First, what browser security mechanism prevents `evil.com` from reading Bob's activity at the banking site?

Your answer:

same origin policy

(b) (2 points) Every DOM element has a `ScrollTop` property that indicates the number of pixels that the element has been scrolled to from its top position. Suppose `evil.com` could read the `ScrollTop` property of the newly opened window. Explain how `evil.com` could then use a Web text fragment to test if Bob is currently logged into `bank.com`. You may assume that if Bob is logged in then the word "Logout" appears on the page, and otherwise the word is not on the page.

Your answer:

evil.com put put a Web text fragment in the link

e.g. https://bank.com/#:~:text=Logout

and read the ScrollTop property to determine if the page scrolled or not; if it did, the attacker knows the user is logged in

4.1 **2 / 2**

✓ **- 0 pts** *Correct*

- 1 pts SOP not specified

- 2 pts Incorrect

4. (16 points) ..... Web text fragments and covert channels

Chrome 80 added a feature, called Web text fragments, that allows one to encode a search string in a URL. When the browser opens that URL, if the search string is found on the page, the page automatically scrolls to that location, and the string is highlighted. For example, to open the CS155 page at the "Projects" section, use the URL

`https://cs155.stanford.edu/#:~:text=Projects`

Feel free to try it out. If the string is not found on the page, no scrolling takes place.

As we saw throughout the course, benign-looking features can have significant security implications. In this question we will explore the risks associated with the Web text fragments feature.

Consider a page at `evil.com` that contains a link to a fictional site

```
<a href="https://bank.com" target="_blank"> Click me </a>
```

Suppose Bob visits `evil.com` and clicks on this link. The result is a new browser window at `bank.com`.

(a) (2 points) First, what browser security mechanism prevents `evil.com` from reading Bob's activity at the banking site?

Your answer:

same origin policy

(b) (2 points) Every DOM element has a `ScrollTop` property that indicates the number of pixels that the element has been scrolled to from its top position. Suppose `evil.com` could read the `ScrollTop` property of the newly opened window. Explain how `evil.com` could then use a Web text fragment to test if Bob is currently logged into `bank.com`. You may assume that if Bob is logged in then the word "Logout" appears on the page, and otherwise the word is not on the page.

Your answer:

evil.com put put a Web text fragment in the link

e.g. https://bank.com/#:~:text=Logout

and read the ScrollTop property to determine if the page scrolled or not; if it did, the attacker knows the user is logged in

4.2 **2 / 2**

✓ **- 0 pts** *Correct*

- 0.5 pts Incorrect / No URL

- 1 pts Incorrect Test

- 1 pts Did not use Web text fragment

- 2 pts Incorrect

(c) (*2 points*) How do you think Chrome prevents the attack you described in part (b)?

Your answer:

they check to see if the request came from a different origin; if so, they don't scroll to the element; alternatively, open the new page with a "noopener" context

(d) (*4 points*) For security reasons, the browser will only search for the text fragment search string in the top level frame. It will not search for the search string in an embedded iframe. Explain how `evil.com` could mount your attack from part (b) if the search were applied to an embedded iframe. You may assume that `evil.com` can open `bank.com` as an iframe in the `evil.com` page. Moreover, you may assume that scrolling the iframe scrolls the entire page.

Your answer:

evil.com opens the above link in an iframe on evil.com; if the user is logged into bank.com and visits evil.com, the iframe will automatically scroll to "logout", scrolling the entire page down => attacker can check if the page scrolled to see if the user is logged in

4.3 **1 / 2**

- 0 pts Correct: mentions (or clearly alludes to) same origin policy's effect on DOM access

✓ **- 1 pts** *Minor Error: does not clearly mention or allude to same origin policy, but describes isolation between evil.com and bank.com that would result from SOP; or other minor inaccuracy*

- 2 pts Incorrect: does not mention or allude to same origin policy at all, or other major inaccuracy

- 2 pts Missing

Chrome does not need to change scrolling behavior on cross-origin requests since the DOM SOP would prevent ScrollTop from being read by a cross-origin site.

(c) (*2 points*) How do you think Chrome prevents the attack you described in part (b)?

Your answer:

they check to see if the request came from a different origin; if so, they don't scroll to the element; alternatively, open the new page with a "noopener" context

(d) (*4 points*) For security reasons, the browser will only search for the text fragment search string in the top level frame. It will not search for the search string in an embedded iframe. Explain how `evil.com` could mount your attack from part (b) if the search were applied to an embedded iframe. You may assume that `evil.com` can open `bank.com` as an iframe in the `evil.com` page. Moreover, you may assume that scrolling the iframe scrolls the entire page.

Your answer:

evil.com opens the above link in an iframe on evil.com; if the user is logged into bank.com and visits evil.com, the iframe will automatically scroll to "logout", scrolling the entire page down => attacker can check if the page scrolled to see if the user is logged in

4.4 **4 / 4**

✓ **- 0 pts** *Correct*

- 1.5 pts Does not clearly state that evil.com should read the scrollTop property of one of its own components, since scrolling the frame scrolls the whole page (and it cannot access the frame's scrollTop due to SOP)

- 1.5 pts Does not clearly state that attacker should search for text by loading web text fragment link (either through iframe src or in javascript)

- 1 pts Implies user must still click web text fragment link or take some other action for attack to work; this is not true since the web text fragment can be set to the source of the iframe

- 1 pts Contains irrelevant and/or inaccurate info

- 2 pts Falsely states (or implies) that evil.com can read DOM tree of embedded bank.com iframe (this would violate the SOP)

- 4 pts Incorrect and/or carries out different attack than that in (b)

- 4 pts Missing

(e) (*2 points*) Suppose that using a covert channel, `evil.com` could measure the time that it takes the browser to open a link in a new window. Again, explain how `evil.com` could test if Bob is currently logged into the banking site. You may assume that the search for the string terminates quickly when the string is found at the top of the page compared to when the string is not found on the page at all.

Your answer:

the attacker measures the load time when the user clicks on the malicious link; if the load time is short, the attacker knows the text was found quickly => knows the user is logged in; this attack is easier if the attacker chooses to search for text near the top of the page

(f) (*4 points*) How would you suggest that Chrome defend against your attack from part (e)?

Your answer:

enforce a constant search time so that the search for the string doesn't terminate until the set time occurs, even if the text is found early => makes it so that the attacker can't differentiate between logged in & not logged in

For completeness, we note that a web page can prevent text fragments from being applied to it by including the HTTP header

```
Document-Policy: force-load-at-top
```

This header forces the page to always open at the top scroll position.

4.5 **2 / 2**

✓ **- 0 pts** *Correct*

- 2 pts Not Attempted

- 1 pts Slightly Incorrect

(e) (*2 points*) Suppose that using a covert channel, `evil.com` could measure the time that it takes the browser to open a link in a new window. Again, explain how `evil.com` could test if Bob is currently logged into the banking site. You may assume that the search for the string terminates quickly when the string is found at the top of the page compared to when the string is not found on the page at all.

Your answer:

the attacker measures the load time when the user clicks on the malicious link; if the load time is short, the attacker knows the text was found quickly => knows the user is logged in; this attack is easier if the attacker chooses to search for text near the top of the page

(f) (*4 points*) How would you suggest that Chrome defend against your attack from part (e)?

Your answer:

enforce a constant search time so that the search for the string doesn't terminate until the set time occurs, even if the text is found early => makes it so that the attacker can't differentiate between logged in & not logged in

For completeness, we note that a web page can prevent text fragments from being applied to it by including the HTTP header

```
Document-Policy: force-load-at-top
```

This header forces the page to always open at the top scroll position.

4.6 **4 / 4**

✓ **- 0 pts** *Correct*

- 2 pts No mention of making search take constant time

- 4 pts Not Attempted

5. *(14 points)* ..... Secure Communication

SMTP is a network protocol that's used to deliver email from one organization to another. In the protocol, the sending mail server accepts messages from a sending user, establishes a TCP connection with the destination domain's email server, and then delivers the message. Unfortunately, SMTP did not include any built-in security protections. Instead, later additional protocols were introduced to improve email security.

DomainKeys Identified Mail (DKIM) is an opt-in protocol in which a mail server can cryptographically sign the email messages it sends with a private key and then attach the signature to the messages it sends. If a recipient receives a message with a signature, it looks up the sender's corresponding public key using a specially named DNS record. Using the server's public key, the recipient validates that the message originated from the server and that it hasn't been altered in transit.

Sender Policy Framework (SPF) is a complimentary email security protocol that allows organizations to whitelist a specific set of IP addresses that are allowed to send mail on behalf of a domain. This prevents attackers from sending mail on behalf of the organization. When a recipient any mail server receives any message, it looks up the allowed IP addresses for the domain through a specifically named record and drops messages that are not from one of the IPs.

(a) *(3 points)* What is a security fundamental flaw in the DKIM protocol and how would you extend the protocol to prevent this attack? Hint: Think about how the attacker might modify the message.

Your answer:

the attacker could add an additional "from" field; when the recipient verifies the signature, they do so using the DNS record for the 1st (original) "from" field => message is valid, but when they reply to the message they also reply to the attacker

5.1 **1.5 / 3**

- **0 pts** Correct

✓ **- 1.5 pts** *Correct flaw, incorrect protocol fix*

- **3 pts** Blank / Incorrect

(b) (*3 points*) **What about the SMTP protocol prevents off-path attackers from being able to spoof SMTP connections from an SPF-whitelisted IP address in an undetectable manner?**

Your answer:

(c) (*2 points*) **Organizations oftentimes whitelist all IP addresses that belong to the organization in their SPF records. How might an attacker take advantage of that error to send spam even if the mail server is secure?**

Your answer:

an attacker may have access to a subdomain and can send an email from the parent domain that appears validated

(d) (*3 points*) **Even if all of a senders' infrastructure is perfectly secure, an attacker may be able to send spam as that organization by mounting an attack against *other* Internet infrastructure or protocols. Describe how an attacker could send spam as an organization without compromising any of that organization's infrastructure.**

Your answer:

5.2 **0 / 3**

- 0 pts Correct response; identifies the TCP handshake

- 1 pts Solid response, but with a minor mistake, misunderstanding, or significant lack of clarity

✓ **- 3 pts** *Incorrect or blank*

(b) (*3 points*) **What about the SMTP protocol prevents off-path attackers from being able to spoof SMTP connections from an SPF-whitelisted IP address in an undetectable manner?**

Your answer:

(c) (*2 points*) **Organizations oftentimes whitelist all IP addresses that belong to the organization in their SPF records. How might an attacker take advantage of that error to send spam even if the mail server is secure?**

Your answer:

an attacker may have access to a subdomain and can send an email from the parent domain that appears validated

(d) (*3 points*) **Even if all of a senders' infrastructure is perfectly secure, an attacker may be able to send spam as that organization by mounting an attack against *other* Internet infrastructure or protocols. Describe how an attacker could send spam as an organization without compromising any of that organization's infrastructure.**

Your answer:

5.3 **0 / 2**

- 0 pts Correctly identifies risk of compromising other hosts in the organization's networks

✓ **- 2 pts** *Incorrect (e.g., assumes spoofing ability) or blank*

(b) (*3 points*) **What about the SMTP protocol prevents off-path attackers from being able to spoof SMTP connections from an SPF-whitelisted IP address in an undetectable manner?**

Your answer:

(c) (*2 points*) **Organizations oftentimes whitelist all IP addresses that belong to the organization in their SPF records. How might an attacker take advantage of that error to send spam even if the mail server is secure?**

Your answer:

an attacker may have access to a subdomain and can send an email from the parent domain that appears validated

(d) (*3 points*) **Even if all of a senders' infrastructure is perfectly secure, an attacker may be able to send spam as that organization by mounting an attack against *other* Internet infrastructure or protocols. Describe how an attacker could send spam as an organization without compromising any of that organization's infrastructure.**

Your answer:

5.4 **0 / 3**

- 0 pts Correct

✓ **- 3 pts** *Not Attempted*

- 1 pts No mention of BGP or DNS/Domain Hijack

(e) (*3 points*) End-to-End Encryption protocols like PGP, OTR, and Signal encrypt the contents of individual messages over email and chat. What information do they fail to protect? Why might this be a concern to a whistleblower?

Your answer:

they don't protect the size of the content being encrypted or the times that the messages are sent; proof that a large encrypted message (potentially containing sensitive docs) was sent at a certain time on a network could be enough to pin down who the whistleblower is

5.5 **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts No explanation

- 2 pts No example of information that is not protected

- 3 pts No answer

6. (*16 points*) ..... Tor Network

As described in Lecture 16, Tor is community-run network that enables anonymous communication.

(a) (*3 points*) Is it safe to visit websites over (unencrypted) HTTP using Tor? Explain why or why not, and if not, which nodes in the Tor network is the user vulnerable from attack from.

Your answer:

no; data sent between you and tor is encrypted, but the data sent from the last node in the relay (the "exit node") will be unencrypted (if http) => the user is vulnerable to attack from this exit node

(b) (*3 points*) If the NSA compromised a single popular Tor entrance node, what information could they collect about users? What will they not be able to see?

Your answer:

they can identify the users and collect the identities of the middle nodes in the relay chains, but can't identify the exit nodes or the destination sites

6.1 **3 / 3**

✓ **- 0 pts** *Correct*

- 1 pts Incorrect explanation, incomplete explanation, or no explanation

- 3 pts Incorrect answer

6. (*16 points*) ..... Tor Network

As described in Lecture 16, Tor is community-run network that enables anonymous communication.

(a) (*3 points*) Is it safe to visit websites over (unencrypted) HTTP using Tor? Explain why or why not, and if not, which nodes in the Tor network is the user vulnerable from attack from.

Your answer:

no; data sent between you and tor is encrypted, but the data sent from the last node in the relay (the "exit node") will be unencrypted (if http) => the user is vulnerable to attack from this exit node

(b) (*3 points*) If the NSA compromised a single popular Tor entrance node, what information could they collect about users? What will they not be able to see?

Your answer:

they can identify the users and collect the identities of the middle nodes in the relay chains, but can't identify the exit nodes or the destination sites

6.2 **3 / 3**

✓ **- 0 pts** *Correct*

- 2.5 pts Incorrectly states that attacker can view traffic content or destination

- 3 pts No answer.

(c) (*3 points*) If an attacker was able to view all traffic that entered and exited the Tor network but none of the traffic within the Tor network, what information might an attacker be able to guess about a website? How could the Tor network protect against this attack?

Your answer:
they could perform timing attacks; multiple connections in quick succession from the same exit node are likely the same user and can be traced to multiple quick connections to a entry node w/ the same pattern

(d) (*3 points*) Does using four Tor relays provide any additional tradeoffs over using three relays?

Your answer:
(+) one additional encryption cycle;
(+) your data makes an additional hop; can make it more difficult to do traffic analysis/ timing attacks

(-) increased latency

(e) (*4 points*) Malware has started to use Tor hidden services to hide command and control (C2) servers that compromised bots connect to to request instructions. What benefits does provide attackers? Consider both the benefits on the compromised bot-side (2 pts) and the C2-side (2 pts).

Your answer:
C2-side:
- much harder to identify and take down the centralized C2 server

bot-side
- bots don't need to know actual identity of C2 server <> C2 server is safer
- harder to identify bots by looking at traffic
- bots have access to large network they can use to perform distributed attacks on

6.3 **1.5 / 3**

- **0 pts** Correct
- **1.5 pts** Information incorrect

✓ **- 1.5 pts** *Protection incorrect*

- **1 pts** Information partially correct
- **1 pts** Protection partially correct

(c) (*3 points*) If an attacker was able to view all traffic that entered and exited the Tor network but none of the traffic within the Tor network, what information might an attacker be able to guess about a website? How could the Tor network protect against this attack?

Your answer:
they could perform timing attacks; multiple connections in quick succession from the same exit node are likely the same user and can be traced to multiple quick connections to a entry node w/ the same pattern

(d) (*3 points*) Does using four Tor relays provide any additional tradeoffs over using three relays?

Your answer:
(+) one additional encryption cycle;
(+) your data makes an additional hop; can make it more difficult to do traffic analysis/ timing attacks

(-) increased latency

(e) (*4 points*) Malware has started to use Tor hidden services to hide command and control (C2) servers that compromised bots connect to to request instructions. What benefits does provide attackers? Consider both the benefits on the compromised bot-side (2 pts) and the C2-side (2 pts).

Your answer:
C2-side:
- much harder to identify and take down the centralized C2 server

bot-side
- bots don't need to know actual identity of C2 server <> C2 server is safer
- harder to identify bots by looking at traffic
- bots have access to large network they can use to perform distributed attacks on

6.4 **3 / 3**

✓ **- 0 pts** *Correct*

- 3 pts Incorrect

- 1.5 pts Partially correct

(c) (*3 points*) If an attacker was able to view all traffic that entered and exited the Tor network but none of the traffic within the Tor network, what information might an attacker be able to guess about a website? How could the Tor network protect against this attack?

Your answer:
they could perform timing attacks; multiple connections in quick succession from the same exit node are likely the same user and can be traced to multiple quick connections to a entry node w/ the same pattern

(d) (*3 points*) Does using four Tor relays provide any additional tradeoffs over using three relays?

Your answer:
(+) one additional encryption cycle;
(+) your data makes an additional hop; can make it more difficult to do traffic analysis/ timing attacks

(-) increased latency

(e) (*4 points*) Malware has started to use Tor hidden services to hide command and control (C2) servers that compromised bots connect to to request instructions. What benefits does provide attackers? Consider both the benefits on the compromised bot-side (2 pts) and the C2-side (2 pts).

Your answer:
C2-side:
- much harder to identify and take down the centralized C2 server

bot-side
- bots don't need to know actual identity of C2 server <> C2 server is safer
- harder to identify bots by looking at traffic
- bots have access to large network they can use to perform distributed attacks on

6.5 **4 / 4**

✓ **- 0 pts** *Correct*

- 4 pts Incorrect

- 2 pts One of the sides incorrect

CS155: Computer Security **Spring 2022**

Take-home Final Exam

Instructions:

- **Please answer all six questions. You have three hours.**
- You may take the exam at any time during the exam window. You have three hours from the moment you begin until the moment you submit your answers on Gradescope.
- The exam is open book, open notes, and open laptops. However, you are expected to do the exam on your own. You may not interact, collaborate, or discuss the exam with another person during the exam window.
- To submit your answers please either (i) use the provided LaTeX template, or (ii) print out the exam and write your answers in the provided spaces, or (iii) write your answers on blank sheets of paper, but please make sure to start each question on a new page. When done, please upload your solutions to Gradescope (course code `N86BR6`).
- The LaTeX template for the final is available here. Please do not share the link with others.
- If you have questions, please post them privately on Ed and we will answer them as quickly as we can.
- Students are bound by the Stanford honor code. In particular, you are expected to do the exam on your own.