CS 155: Spring 2018
June 7, 2018

CS 155 Final Exam

Print your name legibly and sign and abide by the honor code written below. This exam is open book and open notes. You may use course notes and documents that you have stored on a laptop, but you may NOT use the network connection on your laptop in any way, especially not to search the web or communicate with a friend. **You have 2.5 hours.**

The space allocated in this printed exam indicates the length of a good correct answer. Think carefully and answer clearly and succinctly. You may use the back of a page for scratch work. If you use the back side of a page to write part of your answer, be sure to mark your answer clearly.

The following is a statement of the Stanford University Honor Code:

A. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*

   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

C. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

I acknowledge and accept the Honor Code.

_____

(Signature)

_____ _____

(SUNet ID) *(Print your name,* legibly!*)*

☐ **GRADUATING?**

Prob	# 1	# 2	# 3	# 4	# 5	# 6	Total
Score							
Max	10	20	15	20	15	20	100

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

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

_____ (a) Covert channels are used to leak secret information from a high privilege service to a low privilege service.

_____ (b) Control flow integrity is an approach to prevent return oriented programming.

_____ (c) HSTS is designed to defend against wrongly-issued certificates.

_____ (d) Referer validation is sufficient to prevent CSRF exploits.

_____ (e) Source IP spoofing is a key component of a DDoS amplification attack via exposed memcached servers.

_____ (f) When a page at origin `a.com` uses `<script src="">` to load a script from origin `b.com`, the loaded script has access to the DOM and cookies of origin `a.com`.

_____ (g) Positive security indicators (such as pictures of locks on the screen) properly help users assess the security risk of an action they are about to take (such as typing a password into a login field).

_____ (h) System call interposition can be used to prevent an application from accessing the network.

_____ (i) A smartphone that can be unlocked with either a PIN code or a fingerprint is *more* secure than an identical phone that is configured to only accept a PIN code.

_____ (j) Mixed content errors occur when web content on a page is loaded from two different domains.

2. (*20 points*) ..... Questions from all over with a short answer

(a) (*4 points*) Recall that, in the browser, the variable `Window.opener` is the window that opened the current window (or NULL if no such window). In particular, any origin can call `Window.opener.location = 'phishing.com'` to navigate the window that opened the current window to a phishing site. This works even if the opener window contains content from a different origin. Explain why this does not violate the same origin policy.

(b) (*4 points*) Using afl-fuzz, Bob found a buffer overflow in tar (the archiving utility on Unix). Bob has created a malformed tar file named `give-me-a-shell.tar.gz` such that running the command `tar -xf give-me-a-shell.tar.gz` causes tar to execute shellcode and launch a shell. Bob shares his findings with Jane, and says that because the tar program is owned by root on myth.stanford.edu, he can SSH into myth and run the above command to get a root shell. Jane disagrees, arguing that running the command above on myth would give a shell running with Bob's privileges, not root privileges. Who is right, and why?
Note that running `ln -l /bin/tar` gives

```
-rwxr-xr-x 1 root root 383632 Nov 17  2016 /bin/tar
```

(c) (*4 points*) The "`require-sri-for script`" content security policy (CSP) directive ensures that all loaded scripts have a sub-resource integrity (SRI) attribute. Scripts loaded as `<script src="script.js">` will not be loaded. Explain what problem this directive is intended to prevent.

(d) (*4 points*) This question asks about the relationship between a smartphone PIN code and MDM (Mobile Device Management). A smartphone could be configured to automatically lock itself and delete all data after 10 unsuccessful attempts to unlock it with an incorrect PIN.

i. (*2 points*) Why is locking the phone after 10 unsuccessful attempts a good security measure?

ii. (*2 points*) How can MDM help prevent data loss with this configuration?

(e) (*4 points*) TOCTOU refers to a class of time-of-check/time-of-use vulnerabilities. In his guest lecture on Spectre, Paul Kocher talked about a code sample that looks like this:

```
if (x < array1_size)    y = array2[array1[x] * 256];
```

i. (*2 points*) In speculative execution based on branch prediction, which two parts of this code sample may be executed at the same time?

ii. (*2 points*) Why is this a form of TOCTOU error?

3. (*15 points*) ..... User tracking

A web site `xyz.com` wants to tell if a visitor to its site has visited before. This can be easily done with cookies, but cookies can be blocked or cleared. In this question we look at alternate ways for a web site to test if a visitor has visited before.

(a) (*2 points*) The web site's home page can embed an iframe that is marked as static content. The iframe will be cached on the user's browser. Explain how this browser feature can be used to test if the user visited the site before.

(b) (*5 points*) Suppose that when the visitor first visits the web site, the site assigns the visitor a 128-bit unique ID. Explain how the web site can use the method from part (a) to learn the 128-bit ID of a re-visiting user. In other words, not only can the web site tell that the visitor is a repeat visitor, it can learn the assigned user ID.

(c) (*3 points*) In class we discuss HSTS, a flag that a web site can send to the browser to indicate that the site understands HTTPS. When the HSTS flag is set in the browser, the browser will refuse to connect to the site over HTTP. Explain how this browser feature can be used to test if the user visited the site before. Write the browser-side code that the web site can use for this.

(d) (*5 points*) As in part (b), suppose the site assigns to each visitor a 128-bit unique ID. Explain how the web site can use HSTS to learn the 128-bit ID of the visiting user. You can use subdomains of `xyz.com` if needed.

4. (*20 points*) ..... Password checking

(a) (*5 points*) *Weak password checker.* What is the problem with this password-checking code, resembling the C code used in an early version of Unix? Describe a *simple* attack that allows you to log in. Assume this code is compiled using StackGuard. LEN is some constant, say 8. Here `*enteredPassword` is the password supplied by the user.

```
void check_passwd(char *username, *enteredPasswd) {
    char buffer1[LEN];
    char buffer2[LEN];
    /* omitted code to place user's stored password in buffer 1 */
    strcpy(buffer2, enteredPasswd);
    /* compare buffer1 and buffer2 */
    if (!memcmp(buffer1, buffer2, LEN)) {
     /* the buffers are equal so allow login */
    }
    else { /* disallow login */
    }
}
```

Recall that the stack grows downward as local variable are allocated.

Note: For simplicity in this question we assume that the web site stores passwords in the clear. This should never be done in the real-world. Passwords must be hashed and salted before storing. This issue is not relevant to answering this question.

(b) (*5 points*) *More weak password checkers.* What is the problem with the following password-checking code, also derived (somewhat) from real systems? Assume that the caller ensures that `*givenPwd` and `*correctPwd` are exactly LEN characters.

```
void check_passwd(char *username, *enteredPasswd, *correctPwd) {
    for(i=0,  i<LEN, ++i) {
        if (givenPasswrd[i] == correctPwd[i]) {
            sleep(.1);      /* sleep for 0.1 sec */
        } else {
            return(-1);     /* disallow login */
    }}
    return(0);   /* allow login */
}
```

Describe a *simple* attack that lets an attacker extract the correct password associated with the given username using at most $\texttt{LEN} * 256$ attempted logins.

(c) (*5 points*) Write a password checker with the same interface as in part (b) that avoids the problem you identified in part (b).

(d) (*5 points*) *The importance of memory integrity.* Suppose that the conditional `if` instruction in the code from part (b) is implemented in assembly as follows:

```
8a 03                   mov    al,BYTE PTR [rbx]   // load byte into al
8a 0a                   mov    cl,BYTE PTR [rdx]   // load byte into cl
84 c8                   test   al,cl               // compare al, cl
0f 84 10 00 00 00       je     0x1c                // branch if equal
```

The left most column shows the x64 machine code (12 bytes total) and the middle column is the corresponding assembly. This code is stored in DRAM memory. Show that a single bit flip in memory, caused by a stray cosmic ray or by an attacker, can cause all passwords to be accepted as valid. Explain which bit in the code needs to be flipped. Your answer shows the importance of memory integrity for security.

Hint: Here is another way to implement the same conditional `if` and the corresponding machine code.

```
8a 0b                   mov    cl,BYTE PTR [rbx]   // load byte into cl
8a 02                   mov    al,BYTE PTR [rdx]   // load byte into al
84 c8                   test   al,cl               // compare al, cl
0f 84 10 00 00 00       je     0x1c                // branch if equal
```

5. (*15 points*) ..... CSP and the browser XSS filter

Content Security Policy (CSP) directives can be passed in an http header or using a `<meta>` tag in the page html, as in
`<meta http-equiv="Content-Security-Policy" content="script-src 'self' ">.`

(a) (*2 points*) What restriction does the directive `script-src 'self'` cause? (*Hint:* It's analogous to `default-src 'self'`.)

(b) (*2 points*) Assume the following page is loaded from the domain `example.com`. What command in the page will the CSP tag block?

```
<!DOCTYPE html>
<html>
<head>
  <title>CSP Test</title>
  <meta http-equiv="Content-Security-Policy" content="script-src 'self'">
</head>
<body>
  <script type="text/javascript" src="https://evil.com/..."></script>
</body>
</html>
```

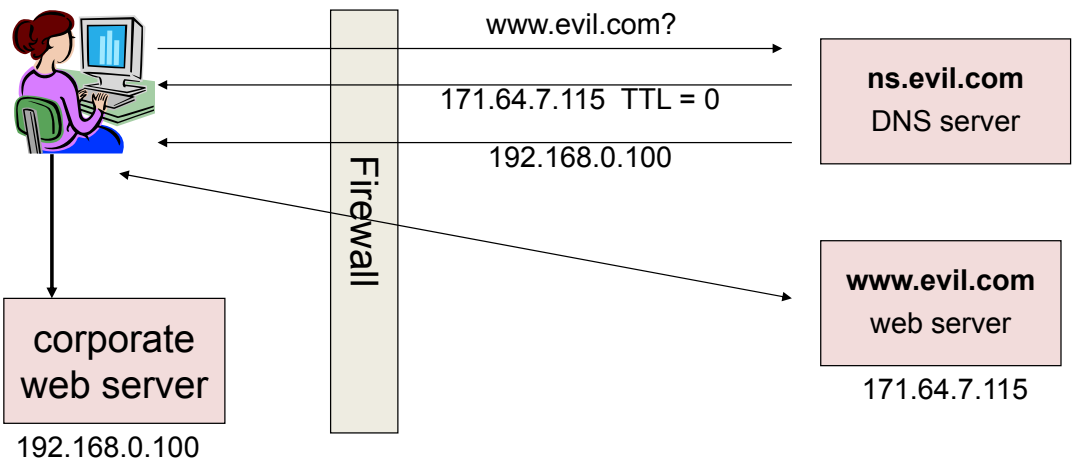
(c) (*4 points*) Microsoft introduced a browser-side XSS filter first in Internet Explorer 8. The purpose of the XSS filter is to mitigate reflective XSS attacks. When the browser loads a URL, the XSS filter first checks if some part of the URL looks like a script, using a set of regular expressions. For example, in the URL `http://example.com/index.php?id=<script>alert(1)</script>`, the value of parameter `id` may be part of an XSS attack because it matches the form of a script. Strings that contain "`<meta>`" also match one of the regular expressions for identifying possible scripts.

To block a reflective XSS attack, the browser checks if the page HTML contains a script that appeared in a URL parameter. For example, if the URL contained the string `<script>alert(1)</script>`, the browser treats this script in the response HTML as a reflected XSS attack and changes every occurrence of this string in the response to `<sc#ipt>alert(1)</script>`. The filter may similarly change `<meta>` to `<me#ta>`.

Question: What will a browser running the XSS filter do to HTML content that is received in response to the following url:
`http://example.com/xss.html?<meta http-equiv="Content-Security-Policy" content="...">` ?
That is, how will the HTML be changed?

(d) (*5 points*) Assume that the page shown in part (b) of this question is located at `http://example.com/page.html`. Write a version of this url that has an additional query parameter that will defeat the purpose of the CSP directive in the page. Do not worry about URL encoding. You can write spaces in your answer instead of `%20`, and so on.

(e) (*2 points*) How would an attacker exploit your URL from part (d)?



6. (*20 points*) ..... DNS Rebinding

The figure above shows a browser behind a corporate firewall. In the first steps shown in the figure, the browser renders web content from `www.site.com` that contains an embedded frame `<iframe src="http://www.evil.com">`. Because the user has not requested a connection to `evil.com` before, the user's system asks the DNS server `ns.evil.com` for the IP address of `www.evil.com`. The DNS server returns a resource record (RR) giving the address as 171.64.7.115, with TTL = 0. The user's system uses the address 171.64.7.115 to load the frame.

A short time later, the user's browser makes a second request to `www.evil.com` to load a second frame of the same page, using a similar `iframe` line as the one above. Because the TTL of the previous RR was 0, the user's system goes back to `ns.evil.com` and asks a second time for the IP address of `www.evil.com`. This time, the DNS server returns a resource record with the address 192.168.0.100, which happens to be the IP address of a corporate web server behind the firewall. This causes the user's browser to contact the corporate web server at 192.168.0.100. You may assume that all http server requests are directed to port 80.

(a) (*1 point*) For the same-origin policy governing DOM access in the browser, what are the three components of the origin?

(b) (*1 point*) What is the origin of the first frame loaded by the page?

(c) (*1 point*) What is the origin of the second frame loaded by the page?

(d) (*2 points*) Normally, content from `evil.com` cannot access DOM elements associated with content from the corporate web server. Explain how and why content from `evil.com` can read content from the corporate web server, in the setup above.

(e) (*3 points*) Explain how the `evil.com` name server can be configured so that every step of this attack meets the requirements of DNSSEC.

(f) (*4 points*) One solution to the problem is called *DNS pinning*. In DNS pinning, the user's browser will not query DNS again, even if the TTL is exceeded. Instead, once a frame is loaded from `www.evil.com`, the same IP address will be used for all subsequent requests. Write two complete sentences describing one compelling disadvantage of DNS pinning and why it matters.

(g) (*4 points*) Suppose you want to solve the problem at the firewall *without* blocking DNS traffic based on TTL. If there is a stateless firewall policy that will prevent this attack, write *"Stateless"* as part of your answer and give a stateless policy. If not, explain why state is needed and describe a stateful policy.

(h) (*2 points*) Would blocking DNS RRs with TTL=0 prevent attacks of this form?

(i) (*2 points*) Of the solutions in parts (f), (g), and (h), which is the best one given all these considerations?