CS 155: Spring 2016
June 2, 2016

CS 155 Final Exam

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 hours.**

Print your name legibly and sign and abide by the honor code written below. All of the intended answers may be written in the space provided. 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	20	10	15	15	90

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

_____ (a) Isolation using separate UIDs is a good starting point for applying the principle of least privilege in Unix-based systems.

_____ (b) Any process in a Unix-based system make the `setuid` system call. However, the operating system will only allow non-root processes to change between a small number of possible UIDs.

_____ (c) A network attacker is not as powerful as a web attacker.

_____ (d) `postMessage` is a powerful browser-based communication mechanism that allows any frame to broadcast messages that will be accepted by any other frame, regardless of frame origin.

_____ (e) System call interposition is used to prevent covert channels.

_____ (f) `httpOnly` cookies are never sent over HTTPS.

_____ (g) The logout process in a Web application should mark the session as expired on the server.

_____ (h) High bandwidth DDoS attacks are often mounted using benign network services, such as NTP and DNS.

_____ (i) Remote attestation, as used in Trusted Computing, is a way to prove to a remote party what is the current state of the PCR registers.

_____ (j) Certificate pinning in the Chrome browser prevents rogue CAs from issuing a certificate for `gmail.com` that will be accepted by the browser.

2. (*20 points*) ..... Questions From All Over With a Short Answer

(a) (*4 points*) Suppose A and B are two frames in a browser that are loaded from different origins. Why is it a reasonable security policy to allow A to navigate B to another origin based only on whether the display area of A contains the display area of B and A has control over that area?

(b) (*4 points*) What should Facebook do if it finds a certificate for `facebook.com` signed by a trusted CA, but where the subject's public key is not a public-key that belongs to Facebook?

(c) (*4 points*) What is the purpose of the `Strict-Transport-Security` header in HTTP? Please make sure to explain what attack is being prevented by this header.

(d) (*4 points*) In class we saw an example of a BGP path hijack. What is the purpose of this? Why would an adversary try to do it? How can end-users protect themselves from this?

(e) (*4 points*) In the last lecture Brad Arkin made the point that mitigation techniques are more effective than bug fixing. What was his argument?

3. (*20 points*) ..... Attacks on TLS

Recall that the TLS protocol works as follows (simplified):

- the browser sends a `client-hello` indicating the list of cipher-suites and protocol version that the client understands,
- the server responds with a `server-hello` indicating the chosen cipher-suite and its public-key certificate,
- the browser chooses a random pre-master secret and sends a `client-key-exchange` message to the server containing the RSA encryption of the pre-master secret, encrypted under the server's public-key (this is called an RSA key exchange),
- the server decrypts using its secret-key, at which point both sides derive session-keys from the pre-master secret. All subsequent traffic is encrypted using the session-keys. The server sends back a `finished` message which contains a hash of the entire protocol transcript so far,
- the browser compares the hash in the finished message to its own hash of the protocol transcript and aborts if the two defer. Otherwise. the browser sends back a similar finished message and the session is established.

(a) (*5 points*) An enterprise wishes to decrypt and examine all HTTPS traffic going through its border gateway. To do so it creates a CA public/private key pair and installs the CA public-key on all employee computers, so that employee browsers will trust certificates issued by this CA. The border gateway has the corresponding CA secret key. Explain the process by which the border gateway eavesdrops on an HTTPS connection established by a browser inside the enterprise and connecting to an external HTTPS web site. Make sure to explain exactly what the gateway does at each step of the HTTPS protocol.

(b) (*3 points*) How can the user tell that her connection to, say `facebook.com`, is being intercepted as described in part (**3**a)? Please confine your answer to the existing browser UI.

(c) (*4 points*) At some point in your answer to part (**3**a) the gateway uses its CA signing key to issue a certificate for the remote domain being contacted (say, `bank.com`). Suppose the gateway does so by creating an identical certificate to the one received from the remote site where the only difference is that the CA name is changed, the subject's public-key is changed, and the CA's signature is changed. The gateway makes no other changes to the incoming certificate and no other checks. It is effectively relying on the user's browser to validate the certificate. Explain why this exposes the end user to a man-in-the-middle attack by someone outside the enterprise. Describe the attack.

(d) (*4 points*) An old version of TLS, called SSLv2, is vulnerable to a devastating attack due to Daniel Bleichenbacher: an active attacker can get the web server to decrypt any RSA ciphertext encrypted under the server's RSA public-key, by initiating multiple SSLv2 connections to the server. Unfortunately, many web servers still support SSLv2, in addition to TLS 1.2. Explain how an active man-in-the-middle can exploit this to learn the secret session key of a modern TLS 1.2 session, even for a modern browser that *only* speak TLS 1.2. You may assume the server has a single public-key certificate used for both protocol versions. Make sure to draw the exact message flow and procedure used by the attacker.

(e) (*4 points*) How would you defend against the attack in the previous question? Disabling SSLv2 on the server is a good idea, but let's assume that this is not possible because it will break old clients. Is there another solution?

4. (*10 points*) ..... Blind SQL injection

Consider the following server-side PHP code fragment:

```
$sql = "SELECT firstname, lastname FROM MyUsers
                WHERE username='" . $_GET["username"] . "'";
$result = $conn->query($sql);   // issue SQL query
if ($result->num_rows > 0) {
    print("Welcome back")    // if matching record found
} else {
    print("User not found")     // otherwise
}
```

Here `$_GET["username"]` is the username provided by the browser in the HTTP request. `print` writes its argument to the Web page sent back to the browser.

Clearly this code has a SQL injection vulnerability. However, the response to the SQL query is not displayed in the resulting page, so it is not immediately clear how to exploit this vulnerability.

(a) (*6 points*) Suppose the `MyUsers` table also has a password column, in addition to the three columns referenced in the code above. Show that an attacker can extract the password of any user it wants just using the code fragment above.
Hint: try using SQL queries of the form

```
SELECT firstname, lastname FROM MyUsers
                WHERE username='victim' AND password LIKE 'abc%'
```

The phrase `password LIKE 'abc%'` matches all records where the password begins with 'abc'.

(b) (*4 points*) Consider another example server-side code fragment.

```
$sql = "SELECT firstname, lastname FROM MyUsers
                WHERE username='" . $_GET["username"] . "'";
$result = $conn->query($sql);   // issue SQL query
if ($result->num_rows > 0)
    $found = true;   // record result
// Prepare Web page without referencing $found
```

Here the response sent to the browser is oblivious to the SQL query results. Show that even now an attacker can extract the password of any user it wants.
Hint: try using SQL queries of the form

```
SELECT firstname, lastname FROM MyUsers
     WHERE username='victim' AND
     IF(SUBSTRING(password,3,1)=CHAR(65), SLEEP(10), NULL)
```

Now, if the third character of password is `CHAR(65)` (i.e., 'A'), the response will be delayed by 10 seconds. This is called a time-based SQL injection.

Note: this is not quite proper SQL, but this approach can be made to work in SQL.

5. (*15 points*) ..... Stealing remote files using a media format

FFmpeg is a multimedia framework designed to decode, encode, transcode, mux, demux, stream, filter and play various kinds of media formats. This framework is currently used in a number of Linux, Mac OS X and Windows platforms. In his guest lecture, Alex Stamos mentioned an FFmpeg zero-day vulnerability that lets a successful attacker read local files on a remote machine and send them over the network.

The attack uses HTTP Live Streaming (HLS). HLS sends audio and video as a series of small files, typically of about 10 seconds duration. These small files are called media segment files. An index file (a.k.a playlist) provides an ordered list of the URLs of the media segment files. The URL of the index file is accessed by a client over http. The client then requests the media segment files in sequence, where each file is loaded from the server where it is hosted.

(a) (*2 points*) Suppose that Alice writes HLS playlists and posts them on a web site. A user Bob accesses one of these playlists using his FFmpeg player to play it. How could the author Alice of a playlist use this mechanism to cause Bob's client to communicate an 8-digit number (e.g., 12345678) to a URL owned by Charlie? Say, Alice is trying to make Bob send a message to Charlie because Alice cannot contact Charlie directly due to Internet censorship. You may assume that Charlie operates a web server that responds to URLs of the form `http://www.Charlie.com/mediafile?messge=12345678`.

(b) (*5 points*) Suppose that HLS allows URLs to be constructed using lines from files accessed by the player. For example, a playlist could include a directive that says:

> *Concatenate the first line of the file at URL1 to the first line of the file at URL2 and use the resulting string as the next URL of this media stream.*

How could Alice cause Bob to send Charlie a line of a file from `secret.txt` from Bob's file system?

(c) (*4 points*) One proposed mitigation for FFmpeg vulnerabilities is to run FFmpeg (and Web servers) in an isolated environment based on chroot. What aspects of the attack outlined in this problem be prevented if Bob used chroot?

(d) (*4 points*) Suppose Alice places her playlists on an external website and Bob and Charlie are on the same enterprise (e.g., university) network protected from the external internet by a stateful packet-filter firewalls. Will this help mitigate the problem from part (**5**b)? (Assume Bob is allowed to access external web sites.)

6. (*15 points*) ..... Android WebView

Android allows apps to embed a custom webkit browser, called a WebView. WebViews can render web content obtained either from the Internet or loaded from files stored on the mobile device. Apps load specific content in the WebView by calling the methods `loadUrl`, `loadData`, `loadDataWithBaseUrl`, or `postUrl` and passing either strings containing HTML content or URLs as parameters.

If an app calls method `addJavascriptInterface(obj, "name")` on a WebView instance then *all* JavaScript code in that WebView can call `name.foo()` to cause the app to execute the Java object objs method `foo`, and return its result to the JavaScript code.

Android apps communicate using intents. An implicit intent is delivered to any app that has set an intent filter that matches properties of the intent. An intent filter can declare zero or more `<data>` elements, such as mimeType and scheme
(e.g., `android:mimeType="video/mpeg" android:scheme="http"`).

(a) (*4 points*) Suppose a Yelp app is designed to provide a UI for the same user functions as the Yelp website, tailored to the smaller screen of a phone. Assume the app makes a Java object accessible to Yelp JavaScript, with Yelp data accessible through this object. If the app WebView loads a page from the Yelp website that contains an advertisement in an iframe, how can this advertisement access Yelp data in a way that would *not* be possible if a browser loaded this page and advertisement?

(b) (*4 points*) Android allows developers to intercept and prevent unsupported web resources from loading by implementing two callback methods. A WebView calls `shouldOverrideUrlLoading` before loading a new page in a top level frame and `shouldInterceptRequest` before making any additional web request, including iframe and image loads. In both cases the app has an opportunity to prevent the resource load by returning true in the case of `shouldOverrideUrlLoading` or null in the case of `shouldInterceptRequest`.

Explain how an app developer can use these methods to restrict a Stanford app so that it can only load content from a `stanford.edu` URL.

(c) (*5 points*) The OAuth protocol is designed for web-based authentication through a browser. OAuth is used by Google, Facebook, LinkedIn and other identity providers. Since a WebView provides a form of browser, it is natural for app developers to try to use OAuth through webview. When a webview uses OAuth, a form of session token is returned as part of a URL using the protocol scheme `my_oauth`. This URL is returned to the app through an implicit intent with a filter that specifies this scheme. Describe a method for a malicious app running on an Android platform to steal a session token from an app that uses OAuth via webview in this way.

(d) (*2 points*) By default, an embedded WebView will cancel loading a resource over HTTPS when there is a certificate error. However, developers are able to change this behavior by implementing the method onReceivedSslError. Many posts on StackOverflow provide implementations of onReceivedSslError that ignore certificate errors. What information does a browser provide to a user in this situation that an app loading a page using a WebView does not?