CS 155: Spring 2013
June 11, 2013

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)

☐ **GRADUATING?**

_____

(Print your name, legibly!*)*

Prob	# 1	# 2	# 3	# 4	# 5	# 6	Total
Score							
Max	16	27	14	14	15	14	100

1. (*16 points*) ..... True or False

_____ (a) Cryptography can prevent TOCTOU bugs.

_____ (b) Cryptography help defend against network eavesdropping.

_____ (c) Stack canaries cannot defend against heap-based buffer overflows.

_____ (d) Syncookies prevent massive (>200Gbs) DoS floods.

_____ (e) It is fundamentally harder to create a program analysis tool with a low rate of false negatives than a low rate of false positives.

_____ (f) Fuzzing is a technique that can find security vulnerabilities in the system tested, without requiring access to source code.

_____ (g) Implementing a new web application using prepared statements is effective in preventing SQL injection attacks.

_____ (h) The same-origin policy prevents javascript from reading the value of a cookie.

_____ (i) A stateless packet filter can prevent internal hosts from connecting to an external DNS server.

_____ (j) A stateless packet-filter can block unrequested UDP packets.

_____ (k) A firewall with an application-layer proxy can scan for viruses.

_____ (l) DNSSEC is effective against DNS-rebinding attacks.

_____ (m) A stateless packet filter is effective against DNS-rebinding attacks.

_____ (n) In Unix, a `setuid` system call can always be used to change the effective user ID to the stored user ID.

_____ (o) Static analysis of source code can take all possible program inputs into account.

_____ (p) If a web site uses secure cookies for session management, this will prevent CSRF attacks.

2. (*27 points*) ..... Short Answer

(a) (*3 points*) Suppose an operating system uses a TCP implementation that generates predictable TCP initial sequence numbers. Can this be used to defeat the confidentiality and integrity properties of the HTTPS protocol?

(b) (*3 points*) Suppose a bug in the implementation of `/dev/random` causes it to output an infinite sequence of bytes where all the bytes are equal to '7'. In this environment does ASLR improve system security?

(c) (*3 points*) Same as previous question: buggy `/dev/random`. Would DEP improve system security?

(d) (*3 points*) Buffer overflow and XSS attacks exploit a poor design vulnerable to something called "mixing program control and user data." For both buffer overflows and XSS, say what the user data is and how this affects program control.

(e) (*3 points*) Briefly explain how you would use access control lists or capabilities to enforce the principle of least privilege.

(f) (*3 points*) You run a web vulnerability scanner against your web site and the tool finds no vulnerabilities in your site. Can you conclude that your site is secure?

(g) (*6 points*) Some web attacks have general solutions that help prevent them for all websites. List two such attacks and describe how a web application framework (like Ruby on Rails) can automatically enforce these solutions

(h) (*3 points*) In a TPM-based file encryption system such as BitLocker explain what happens if the master boot record (MBR) which loads the operating system is infected with a root kit. Explain in detail why the Operating System will not boot.

3. (*14 points*) ..... Sandboxing

`Seccomp` is a feature of the Linux kernel that is enabled in (most) contemporary Linux distributions. It restricts a thread to a small number of system callbs: read(), write(), exit(), sigreturn(). If the thread calls any other system call the *entire process*, including all threads in that process, are terminated. This is desirable for security because any failure in the sandbox will likely terminate the process.

Chrome uses `seccomp` to sandbox HTML renderers so that a compromise of the renderer cannot be used to compromise the Chrome controller. Unfortunately, the HTML renderer needs support for more than just the four system calls allowed by `seccomp`. To address the problem Chrome runs an additional *trusted thread* in the same process address space as the sandboxed HTML renderer thread. Whenever the renderer thread needs to make a non-authorized system call, it sends the request to the trusted thread who checks the argument to the system call, and if authorized, makes the call on behalf of the renderer. By running both threads under the same process Chrome achieves overall good performance while sandboxing the renderer. *However, this means that the renderer thread can read and modify the trusted thread's memory.* For the rest of the question we assume the renderer thread has been compromised and is running malicious code.

(a) (*3 points*) Suppose the renderer requests the trusted thread to open a file on its behalf and the filename is stored in memory. The trusted thread reads the file name from memory, verifies that it points to a benign file (e.g. in `/tmp`), and then issues a system call to open the file. Specifically, it does:

```
if (benign(filename))
   fp = open(filename, "r");
```

How could a malicious renderer exploit this design to open a protected file? What is this type of vulnerability called?

(b) (*4 points*) Propose a simple solution to the problem identified above that has little impact on performance.

(c) (*3 points*) Another problem with the code above is that the renderer thread can easily manipulate the trusted thread's stack. How would an attacker exploit this to defeat the code above?

(d) (*4 points*) How would you solve the problem of an untrusted stack?

4. (*14 points*) ..... Unintended Functionality

When a file F is uploaded to Dropbox from the user's machine, the dropbox client software first computes a hash of the file, $h(F)$, and sends it to the Dropbox servers. If the hash matches a hash that Dropbox already received before (possibly from another user), Dropbox assumes that the files are the same and does not trigger the client to upload the new file. When the user syncs (a.k.a downloads) the file to another machine, Dropbox sends the copy of the file it has on its servers.

(a) (*3 points*) Suppose the hash function h used by Dropbox is such that for any file F and malware M it is easy to find a suffix S such that $h(F) = h(M.S)$ where dot denotes concatenation. Describe how an attacker can use this to easily spread the malware M to many Dropbox users from his own machine. For simplicity, assume the attacker obtains a copy of a popular movie before it is released.

(b) (*3 points*) What standard property should the Dropbox hash function satisfy to prevent this attack?

(c) (*4 points*) Suppose a movie studio is about to release a blockbuster (very popular) movie M. The studio wants to test if the movie has been pirated and uploaded to Dropbox. Explain how this can be done using the Dropbox client software, without any special Dropbox server access.

(d) (*3 points*) The Dropbox "feature" you used to answer the previous question can be used to explore which files are stored on Dropbox. If you modify the Dropbox system to prevent this, describe the increase in bandwidth cost to the Dropbox servers?

5. (*15 points*) ..... The HTML canvas element

The canvas HTML element creates a 2D rectangular area and lets Javascript draw whatever it wants in that area. Canvas is used for client-side graphics such as drawing a path on a map loaded from Google maps. For the purpose of the associated same-origin policy, the origin of a canvas is the origin of the content that created it. In the map example, the origin of the Javascript that creates the canvas is Google. Canvas lets Javascript read pixels from any canvas in its origin using the GetImageData() method.

(a) (*5 points*) Canvas lets Javascript embed images from any domain in the canvas. Suppose a user has authenticated to a site that displays private information. Describe an attack that would be possible if Javascript from one domain could embed an image from another domain in the canvas and then use GetImageData() to read pixels from that image.

(b) (*3 points*) How would you restrict GetImageData() to prevent the attack above?

(c) (*4 points*) A canvas element can be placed anywhere in the browser content area and can be made transparent so that the underlying content under the canvas shows through. What security problem arises if calling GetImageData() returned the actual pixels shown on the screen at that position?

(d) (*3 points*) How would you design GetImageData() to prevent it from reading screen content? Propose a design that does not require the browser to test if the requested pixel is over content from another origin.

6. (*14 points*) ..... Android WebView Security

WebView is an Android Library class that provides web browser functionality to app developers. An app may interact with web content by creating an object that is an instance of the WebView class and navigating it to web pages using the same basic principles as a user navigating a web browser.

The WebView class defines a function `loadUrl` that takes a single `String` parameter. If the parameter value is a URL like `www.google.com` then the WebView instance will navigate to that page. If the parameter value is of the form `javascript:code_goes_here` then the WebView instance will execute the javascript code `code_goes_here` in the context of the currently open web page.

(a) (*6 points*) Suppose an Android app wants to let other apps send it URLs to view. It creates an Activity that receives an Intent containing a String, which it passes to loadUrl(). It might do this with the following code that is called whenever the Activity receives an Intent of the appropriate type.

```
public void onReceive(Context ctx, Intent intent){
    String url = intent.getExtras("url");
    myWebViewInstance.loadUrl(url);
}
```

Describe how a malicious app running on a phone with this browser app installed could take advantage of this behavior to steal the user's cookie for `www.bank.com` and send it to `attacker.com`.

(b) (*8 points*) A powerful WebView feature that is unique to mobile phones is called the *javascript interface*. This interface allows apps to expose Java objects to the javascript code running inside a WebView instance. The app calls `myWebViewInstance.addJavascriptInterface(myObject, "obj")` and then javascript code can access the object referenced by `myObject` through the name `"obj"`, as illustrated in the example code below. Objects added using the javascript interface are not protected by Same Origin Policy rules. All origins with content rendered in the WebView instance can access these objects.

```
// In the java code for the app
class Foo{
    public void printStuff(){
        // Prints some stuff
    }
}
...
myWebViewInstance.addJavascriptInterface(new Foo(), "foo");

// In the javascript code running in WebView
foo.printStuff() // Prints stuff in the app
```

i. (*4 points*) Assume that an app developer does not know about how the Same Origin Policy interacts with javascript interface objects and adds the ContactManager to the Javascript Interface. What vulnerability does this introduce? Explain an attack.

ii. (*4 points*) What trust assumption must an app developer make about the websites he will access with his WebView instance if he wants to use the javascript interface?