CS 155: Spring 2012
June 11, 2012

CS 155 Final Exam

This exam is open books 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	# 7	Total
Score								
Max	10	12	9	17	14	9	14	85

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

_____ (a) Control hijacking attacks are possible only when there is an overflow of a buffer located on the stack.

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

_____ (c) Fuzzing is a bug-finding technique that finds all security vulnerabilities in the system tested, without requiring access to source code.

_____ (d) Web servers can instruct browsers to store a given cookie for many years.

_____ (e) Firefox's private browsing mode will prevent web sites from discovering your name or IP address.

_____ (f) Client-side validation of form data is as secure as server-validation; the only difference is that it happens on the client's machine.

_____ (g) Implementing your web application using prepared statements is effective against SQL injection attacks.

_____ (h) Bot nets can be used for denial of service attacks.

_____ (i) With Trusted Computing disk encryption keys are only given to authorized boot loaders and therefore Trusted Computing prevents users from booting Linux on their machines.

_____ (j) During its initial phase of propagation, a well-designed worm may spread exponentially fast.

2. (*12 points*) ..... Short Answer

(a) (*3 points*) Briefly explain the difference between an HTTPS session protected using a regular certificate and an HTTPS session protected using an extended-validation certificate.

(b) (*3 points*) Why are compartmentalization and isolation needed in order to support the principle of least privilege?

(c) (*3 points*) You are setting up a home computer for a friend and you want to provide some firewall protection. One option is a small hardware device that operates as a stand-alone firewall. Another is firewall software that runs on your friend's computer. What are the relative advantages of each?

(d) (*3 points*) It was recently discovered that a vulnerability at a certificate authority resulted in an unknown entity obtaining a certificate for `gmail.com`. Explain why this is a problem.

3. (*9 points*) ..... Code Injection

Code Injection attacks involve inserting code, chosen by the attacker, into a system or application which then executes it. Circle "yes" or "no" for each attack below, to indicate whether it is a form of code injection attack.

(a) YES NO TOCTTOU (time-of-check-to-time-of-use)

(b) YES NO Buffer overflow

(c) YES NO Port scanning

(d) YES NO SQL injection

(e) YES NO DNS cache poisoning via Kaminsky's attack

(f) YES NO Reflected XSS

(g) YES NO CSRF

(h) YES NO TCP SYN flooding

(i) YES NO Clickjacking

4. (17 points) ..... Apple iOS Sandboxing

Apple iOS is a Unix-based operating system used on iPhone and iPad platforms. A recently released *iOS Security* document describes application sandboxing and emphasizes the following points:

- All third-party apps are "sandboxed," so they are restricted from accessing files stored by other apps or from making changes to the device.
- Each app has a unique home directory for its files, which is randomly assigned when the app is installed.
- If a third-party app needs to access information other than its own, it does so only by using application programming interfaces (APIs) and services provided by iOS.
- The majority of iOS runs as the non-privileged user "mobile," as do all third-party apps.
- Access by third-party apps to user information and features such as iCloud is controlled using declared entitlements. Entitlements are key/value pairs that are signed in to an app and allow authentication beyond runtime factors like unix user ID. Since entitlements are digitally signed, they cannot be changed.
- The combination of required code signing, sandboxing, and entitlements in apps provides solid protection against viruses, malware, and other exploits that compromise the security of other platforms.

Questions:

(a) (*2 points*) The first sentence says that "third-party apps . . . are restricted . . . from making changes to the device." Name one design choice listed above that contributes to this goal.

(b) (*3 points*) Why does iOS assign *random* home directory names instead of standard names based on the name of the application? *Hint:* You may recall from your operating system class that there are 5 basic Unix system calls for file I/O.

```
1.  int open(char *path, int flags [ , int mode ] );
2.  int close(int fd);
3.  int read(int fd, char *buf, int size);
4.  int write(int fd, char *buf, int size);
5.  off_t lseek(int fd, off_t offset, int whence);
```

The first one, `open`, returns a file descriptor corresponding to a path; the others all require a file descriptor as an argument.

(c) (*3 points*) How does the user id "mobile" used for apps in iOS compare with the user id assigned an Android app? Which supports the Principle of Least Privilege better? Why?

(d) (*1 point*) What standard access control concept, often contrasted with access control lists, appears to be used in the iOS concept of entitlements?

(e) (*3 points*) Give one compelling reason why the iOS designers might have chosen to implement this concept cryptographically instead of using OS control?

(f) (*2 points*) How are entitlements likely to be used by the APIs and services provided by iOS? (*Hint:* you can infer the answer from the information given. *Be brief.*)

(g) (*3 points*) The Apple document explains iOS protects against viruses and other malware. Do you expect there to be malware attacks against iOS devices? If so, explain one kind of vulnerability that is not addressed by the design of iOS.

5. (*14 points*) ..... Memory management

The iOS `_MALLOC(size_t size, int type, int flags)` function allocates `size` bytes on the heap. Internally blocks are represented as a length field followed by a data field:

```
struct _mhead {
  size_t  mlen;
  char    dat[0];  }
```

The `mlen` field is used by the `free()` function to determine how much space needs to be freed. In iOS 4.x the `_MALLOC` function was implemented as follows:

```
1   void * _MALLOC(size_t size, int type, int flags)  {
2     struct _mhead  *hdr;
3     size_t memsize = sizeof (*hdr) + size;
4     hdr = (void *)kalloc(memsize);  // allocate memory
5     hdr->mlen = memsize;
6     return  (hdr->dat);
7   }
```

In iOS 5.x the following two lines were added after line 3:

```
int o = memsize < size ? 1 : 0;
if (o)  return (NULL);
```

(a) (*5 points*) Why were these lines added in iOS5.x? Describe an attack that may be possible without these lines.

(b) (*3 points*) Suppose two blocks are allocated one next to the other. Later an overflow in the first buffer overwrites the `mlen` field of the second buffer (the data field of the second buffer is unaffected). What goes wrong if after the overwrite the `mlen` field contains a smaller value than before the overwrite? It suffices to briefly describe the type of error rather than present an explicit attack.

(c) (*3 points*) What goes wrong if after the overwrite the `mlen` field contains a greater value than before the overwrite? It suffices to briefly describe the type of vulnerability rather than present an explicit attack.

(d) (*3 points*) In iOS the kernel heap is divided into zones. When the zone allocator is called it allocates a new zone in a random location in kernel space. What security purpose does randomization serve in this context?

6. (*9 points*) ..... Certificates

John Smith generates a public/private key pair, and buys from a trusted CA (e.g. Symantec) a personal certificate for *John Smith* on his public key. He then generates a second public/private key pair, and uses his *John Smith* private key to sign a certificate on the second public key where the common name is set to `www.amazon.com`.

Now, John, being the malicious sort, intercepts an SSL connection to Amazon.com. John presents his forged certificate for `www.amazon.com` to the intercepted user. The intercepted user's browser incorrectly recognizes the fake Amazon certificate as legitimate because there is a valid certification path to the trusted CA.

(a) (*4 points*) Why does this scenario present an attack? How can it be exploited?

(b) (*5 points*) How would you fix this problem?

(Last July it was discovered that the iPhone's mobile Safari is vulnerable to this attack)

7. (*14 points*) ..... Transparent proxies

Browsers generally restrict network access given to web content: Web content can only make requests to remote servers using HTTP and the response is accessible only if the remote site is in the same origin as the content.

(a) (*3 points*) What would go wrong if browsers gave web content read/write access to TCP network sockets?
Hint: Suppose the browser is behind a corporate firewall. Explain how `attacker.com`, outside the firewall, could host a web page that when visited will exfiltrate data from internal web servers.

(b) (*3 points*) Despite the risk, there is strong demand for giving socket-level access to web content. This is needed, for example, for peer-to-peer games and video chats where the overhead of the HTTP protocol is undesirable. Java, Flash player, and WebSocket all provide facilities for giving web content direct access to TCP sockets. To prevent the attack from part (a), when a script asks for socket-level access, Flash player will first ask the destination server whether it permits web content to talk to it over a TCP socket. If the server says yes then Flash player gives the Flash application read/write access to a socket connected to the target server.

Suppose a Flash app requests socket-level access to a server, but when Flash player queries the server, the server never responds. What should Flash player do? Justify your answer.

(c) (*5 points*) Corporate networks often use transparent proxies to cache static content. These proxies examine browser HTTP requests and if the object requested is in the proxy's cache, the proxy responds from cache rather the forwarding the request to the target web server. If the object is not in cache, the proxy forwards the request to the server without modifying the packets sent by the browser (this is why these proxies are said to be *transparent*). Here we assume the proxy forwards the request to the server named in the `host` header of the incoming HTTP request.

Show that an attacker at `attacker.com` can write a script that when run in the client's Flash player can send HTTP requests and receive responses from any server in the corporate LAN. Assume that all corporate servers *deny* socket-level access when Flash player queries them. You may assume the proxy lives inside the corporate LAN.
Hint: Use the fact that the transparent proxy forwards connections to the server named in the `Host` HTTP header.

(d) (*3 points*) How would you prevent the attack from part (c)?