Midterm 1

CMSC 414: Computer and Network Security

March 12, 2015

Name __________________________________________

Instructions

Do not start until told to do so!

- This exam has 9 pages (including this one); make sure you have them all
- You have 75 minutes to complete the exam
- The exam is worth 100 points. Allocate your time wisely: some hard questions are worth only a few points, and some easy questions are worth a lot of points.
- If you have a question, please raise your hand and wait for the instructor.
- You may use the back of the exam sheets if you need extra space.
- In order to be eligible for partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Definitions</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2 Security goals</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>3 Design principles</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4 Memory safety</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>5 SQL injection</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>6 CSRF attacks</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>7 XSS attacks</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. Definitions (10 points total, 1 point each)
For each of the descriptions below, give the term (from the table) that best describes it.

<table>
<thead>
<tr>
<th>Buffer overflow</th>
<th>Defense in depth</th>
<th>Polymorphic</th>
<th>Security by obscurity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canary</td>
<td>Heartbleed</td>
<td>Reflected XSS</td>
<td>Stored XSS</td>
</tr>
<tr>
<td>Complete mediation</td>
<td>Least privilege</td>
<td>Return-to-libc</td>
<td>Stuxnet</td>
</tr>
<tr>
<td>Cookie</td>
<td>Logic bomb</td>
<td>Rootkit</td>
<td>TOCTTOU vulnerability</td>
</tr>
<tr>
<td>CSRF</td>
<td>Metamorphic</td>
<td>Same origin policy</td>
<td>Time bomb</td>
</tr>
</tbody>
</table>

(a) One of the things that made this noteworthy was that it involved four zero-day exploits.

**Answer:** *Stuxnet*

(b) This type of virus can be detected with straightforward signature-based schemes, but only by running it (in a sandbox).

**Answer:** *Polymorphic*

(c) The notion that one can gain more robust security by using multiple mechanisms of different types.

**Answer:** *Defense in depth*

(d) This helps to overcome the fact that HTTP is stateless.

**Answer:** *Cookie*

(e) The notion that, when checking access control policies, one must ensure that every access to every object is checked.

**Answer:** *Complete mediation*

(f) A software flaw that occurs when a program tests for access to an object separate from when it provides access.

**Answer:** *TOCTTOU vulnerability*

(g) This attack exploits the trust that a website has in the URLs issued to it by its clients.

**Answer:** *CSRF*

(h) An in-browser defense that ensures that a website cannot obtain cookies from a different domain.

**Answer:** *Same-origin policy*

(i) A countermeasure to making stack memory non-executable.

**Answer:** *Return-to-libc*

(j) An attack based on embedding a script into a URL, such that when a server processes the URL, its reply includes the script within it.

**Answer:** *Reflected XSS*
2. (Security goals, 12 points total, 4 points each)

In each of the following scenarios, there are vulnerabilities against at least one of the three basic security goals: (C)onfidentiality, (I)ntegrity, and/or (A)vailability. Circle all that apply, and provide a brief description of an attack that would violate each applicable security goal.

(a) C I A
Thanks to some in-class advertising, socks.com has really gotten popular! To deal with the influx of queries, it only allows at most 1000 concurrent connections at any point in time.

Answer: A (just open many connections and don’t close them, e.g., slow down the transfer)

(b) C I A
Android applications can read the current battery level of the user’s mobile device, and report it back to the application developer’s server. Recent results have shown that, using only these readings, one can infer your location within 100 meters (even if the user denies permission to any explicit location information). Taking readings can result in the phone having to stay awake longer, thus draining battery. Moreover, users are currently not asked whether or not to give permission to these readings.

Answer: C (location tracking), A (battery drain)

(c) C I A
Transferring high-definition video can consume a lot of network bandwidth, which is expensive for Internet Service Providers (ISPs). To combat this, some ISPs “transcode” streaming video that traverses their network, lowering the definition of the video so they have less to send. Some even go so far as to insert their own advertisements to make up for the extra bandwidth costs.

Answer: I (integrity of the video).

Also possibly A (unfair access to lower quality, especially if they’re inserting long ads).

A weak (half-credit) argument for C (the network provider knows what you are viewing, but they know that anyway).
3. (Design principles, 8 points total, 4 points each)

For the next set of problems, consider the following design principles:

(i) Use fail-safe defaults  
(ii) Separation of responsibilities  
(iii) Security is economics

(iv) Defense in depth  
(v) Account for human factors  
(vi) Ensure complete mediation  
(vii) Least privilege

(a) When you go to Costco, the cashier who checks you out provides you with an itemized receipt. Then, when you go to walk out the sole public exit, another employee checks your itemized receipt and, if all the items in your cart are on the receipt, draws a line on the receipt and allows you to leave with your goods.

Identify two security principles applied in this scenario. For each, provide in a single sentence a description of why it meets that design principle.

Answer:

- Ensure complete mediation: everyone needs to go through the door with the person checking receipts.
- Separation of responsibilities: one creates, the other checks, so you would need to collude with more than one employee.
- Arguably also defense in depth (though somewhat weaker).
- Others answers permitted, so long as they have compelling supporting arguments.

(b) socks.com used to require all of its employees to change their passwords every week. Strangely enough, after a regular security audit, they notice that the vast majority of their break-ins are because attackers are guessing employees’ passwords! The security team halts development of their latest secure shopping cart to instead install fingerprint scanners on all employees’ machines, to be used instead of passwords.

Identify two security principles applied in this approach. For each, provide in a single sentence a description of why it meets that design principle.

Answer:

- Human factors matter: the system is far more usable now.
- Security is economics: diverted attention to the biggest threat.
4. Memory safety (28 points total)

For each of the following functions (except for part (d)):

i. Provide an input that would violate its postcondition (the “ensures” statement) without violating the precondition (the “requires” statement).

ii. Provide a fix (you may rewrite the code or just annotate it; feel free to use the back of the page) so that the postcondition holds for all inputs that meet the precondition. If it is ever impossible to satisfy the post-condition, just return safely or exit.

(a) (7 points)

/* Ensures: Returns a copy of the ASCII string message (‘msg’) or NULL on failure. */
/* Requires: msg != NULL */
char *heartbeat(const char *msg, size_t msg_len)
{
    char *echo = malloc(msg_len);
    if(echo == NULL) return NULL;
    memcpy(echo, msg, msg_len);
    return echo;
}

Answer: Length greater than message. Fix by taking strlen instead.

(b) (7 points)

/* Requires: orig_string != NULL and valid pointer */
/* Ensures: shorter_string is the same as orig_string, but with the last printable character removed */
char *truncate(char *orig_string)
{
    size_t num_bytes = strlen(orig_string) - 1;
    char *shorter_string = malloc(num_bytes);
    if(shorter_string != NULL)
        strncpy(shorter_string, orig_string, num_bytes);
    return shorter_string;
}

Answer: strlen could be zero, in which case it would allocate the maximum number of bytes. Return NULL in such a case.
(c) (7 points)

```c
struct TrackInfo /* As in the tracks of a record */
{
    char track_name[512];
    int runtime;
};

struct TrackInfo tracks[16]; /* initialized elsewhere */

/* Ensures: Prints the info of the track 'track_number' */
void print_info(int track_number)
{
    printf(tracks[track_number].track_name);
    printf(" %d minutes\n", tracks[track_number].runtime);
}
```

Answer: Doesn’t check bounds: should add track_number < 16

(d) (7 points) The following code is compiled using a 32-bit x86 compiler that implements stack canaries:

```c
char fun(int arg1, int arg2)
{
    int local[2];

    /* body of function follows: we draw the stack diagram at this point */
    ...
}
```

In the following stack diagram (note that higher addresses are on top), fill in each empty box with the value that would be held there once we reach the point specified in the code above. Use the following values:

<table>
<thead>
<tr>
<th>Canary</th>
<th>arg1</th>
<th>arg2</th>
<th>local[0]</th>
<th>local[1]</th>
<th>Saved %ebp</th>
<th>Saved %eip</th>
</tr>
</thead>
</table>

Answer: Sorry, I originally had this as arg1 then arg2: if you had that and got points off, come in for a regrade.

From high addresses to low addresses: arg2, arg1, Saved %eip, Saved %ebp, Canary, local[1], local[0]
5. SQL Injection (16 points total)

Recall the following syntax (note that this database permits single or double quotes):

- `SELECT * FROM Grades WHERE Course = "CMSC414" AND (FirstName="John" OR LastName="Doe");`
- `UPDATE Grades SET StudentGrade="F" WHERE Course = "CMSC414";`
- `UPDATE Grades SET Access='Yes' WHERE (Course='CMSC414' AND Uname = 'admin');`
- `SELECT * FROM Grades; -- This is a comment`

Once again, I have gone and given everyone in CMSC414 an F. The bad news: we won’t change this. The good news: we didn’t sanitize our inputs.

We’ve set up a website where you can enter your username and password and view your (failing) grades. However, this time I’ve tried to add access control: the idea is that only users with the `Access` field equal to `Yes` (as in the above example) are allowed to view grades. To enforce this, on the backend, we store the username and password in variables `$_username` and `$_password`, respectively, and run the following SQL query:

```
SELECT * FROM Grades WHERE (Access='Yes' AND (Course='CMSC414' AND Uname = '$_username' AND Pass = '$_password'));
```

It then takes the output of this command and presents it to you on a website.

For each of the below answers clearly delineate any important but not readily apparent whitespace with \_.

(a) (5 pts.) Provide a username and password that would allow you to view everyone’s grades.

**Answer:** `'); OR 1=1; -- _`

(b) (5 pts.) Provide a username and password that would give *you* an A and *everyone else* in CMSC414 a B.

**Answer:** `'); UPDATE Grades SET StudentGrade='B' WHERE Course='CMSC414';
UPDATE Grades SET StudentGrade='A' WHERE (FirstName = 'actual first name' AND LastName = 'actual last name'); -- _`

(c) (5 pts.) Having caught onto your schemes, we update our backend processing to escape single quotes: the way we do this is by replacing each instance of `'` with `\'` (i.e., we place a single backslash in front of it) for both the `$_username` and `$_password` variables. After that, we run the above SQL query.

Provide a username/password that, under this scheme, will change all CMSC414 students’ grades to an A.

**Answer:** `\\'); UPDATE Grades SET StudentGrade="A" WHERE Course="CMSC414"); -- _`

(d) (1 point) Name a countermeasure we could have employed to keep your answer from part (c) from working.

**Answer:** *Escape better; or prepare/bind statements.*
6. CSRF attacks (14 points total)

Consider a web-based forum in which a new topic can be posted using a GET request for the following URL:


Suppose further that, when a user logs in, the website provides a session cookie named \( \text{sid} \), and that a message is posted only if a valid \( \text{sid} \) is provided.

(a) (4 points) Briefly explain why the above URL is susceptible to CSRF attacks.

\textbf{Answer:} It is easy to predict and has a side effect.

(b) (4 points) Briefly describe what the website can do to defend against CSRF attacks while maintaining posting functionality (i.e., “shut off the site” is not a sufficient answer!).

\textbf{Answer:} Make the URL difficult to predict: modify it to include \&sid=…

(c) (2 points each) For each of the following three ways for generating the \( \text{sid} \), label it as \textbf{Secure} or \textbf{Insecure} and give a one-sentence explanation.

- A fixed global \( \text{sid} \) for all users.

\textbf{Answer:} Insecure: any user can log in and get the session cookie for all other users.

- A fixed \( \text{sid} \) for each user across all sessions of that user, but use different \( \text{sid} \)’s for different users.

\textbf{Answer:} Slightly more secure, but the problem is that if you steal it once, you can forge requests from that user forever, whether that user logs out or not.

- A fresh, random \( \text{sid} \) for each user and for each active session.

\textbf{Answer:} The only secure way to go.
7. XSS attacks (12 points total, 4 points each)

(a) State whether you Agree, or Disagree, with the following statement, and provide a brief explanation for your answer:

Websites that do not use cookies are not susceptible to XSS attacks.

Answer: Disagree: XSS can be used to run any code on the client, not just those that access cookies. For instance, you could use XSS to change the website’s contents (e.g., inject your own ads).

For the next two parts of this question, consider a browser that does the following:

- Any time it accesses a URL, it scans the URL itself for the regular expression /<[^>]*>/ – that is, it will find an open-tag character (<), followed by any number of characters, and then a close-tag character (>).
- If the regular expression matched, then the browser stores the matching text: let’s call it $T$. When loading the page, if it finds $T$ anywhere, then it processes it as raw HTTP, refusing to execute any script that may be included inside it.

(b) Briefly describe to what extent this mechanism protects against reflected XSS attacks.

Answer: If the URL contains the open/close-tag characters, then it could protect against this attack. However, if the browser is processing the URL, then the tags would be URL-encoded, and the regular expression would not match.

(c) Briefly describe to what extent this mechanism protects against stored XSS attacks.

Answer: No protection: the script would not be included in the URL.