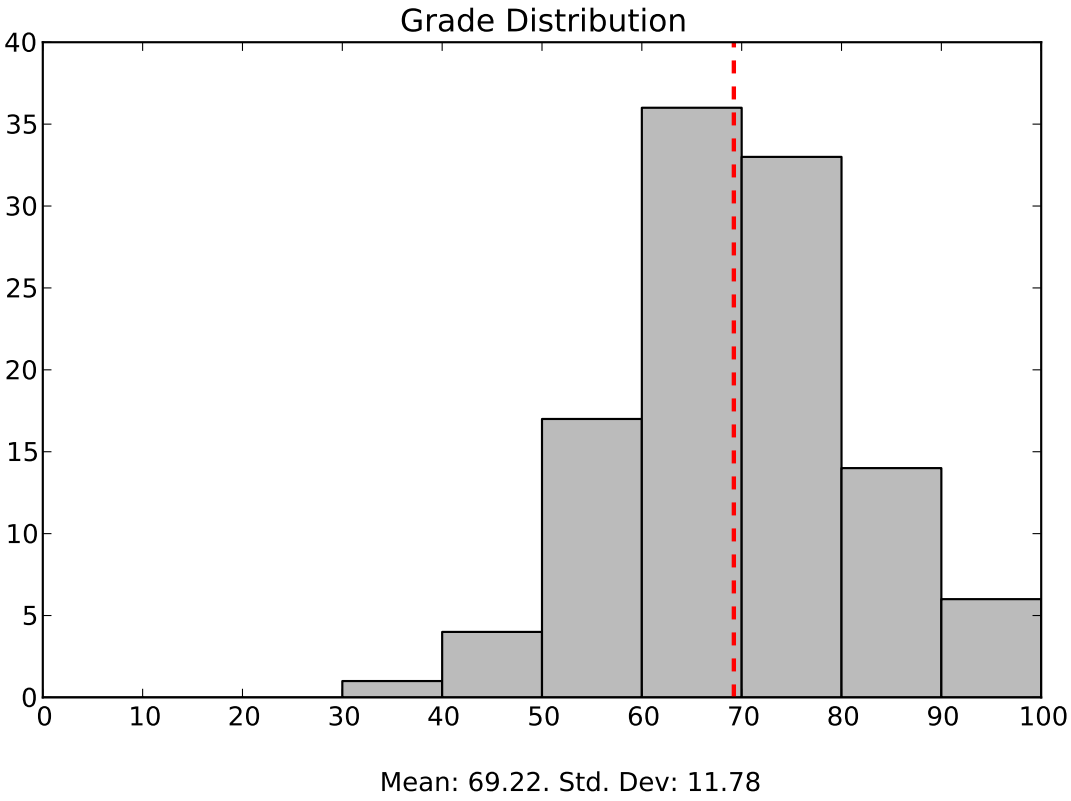
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2013

Quiz II Solutions



Histogram of grade distribution

I Web security

1. [8 points]:

Recall that Zoobar has a cross-site scripting vulnerability that allows an adversary to construct a URL so that the server's response contains arbitrary Javascript code (from the URL itself). In lab 5, you used this vulnerability to steal a victim's cookie, by loading the `sendmail.php` script from our web server (`css.csail.mit.edu`).

Ben Bitdiddle is building a web browser, and he comes up with a plan to stop cross-site scripting attacks like the one above. His idea is to add an extra HTTP header, set by the web server, that prevents the browser from making *any* cross-origin requests from that page—including `IMG` tags, `SCRIPT` tags, forms, links, etc. If this header is set, Ben reasons that a cross-site scripting exploit like the one above will be unable to get to `sendmail.php`, and will not be able to steal the victim's cookie.

Explain how an adversary can steal a victim's cookie in Zoobar (say, running at `zoobar.com`), given a cross-site scripting vulnerability, without making cross-origin requests (i.e., bypassing Ben's plan), without network or DNS attacks, and without exploiting any other vulnerabilities.

Answer: Save the victim's cookie into the victim's zoobar profile. The adversary can later read the victim's cookie by going to the victim's zoobar profile page.

We did not accept answers that set `window.location` because that amounts to cross-origin requests; Ben's extension blocks all cross-origin interaction, including links, etc.

2. [6 points]:

Consider the following Javascript code that returns twice the value of its argument `x`:

```
function double(x) {
  var y = x;
  var cmd = "y=y+" + x;
  eval(cmd);
  return y;
}
```

Propose a design that extends lab 6 to safely support `eval`, such as in the above example, without allowing an adversary to execute arbitrary Javascript code in the web browser. Your design should support invoking `eval` on strings that are computed at runtime and cannot be determined statically, as in the above example. Your design should support the same subset of Javascript in `eval` as in lab 6 itself.

Don't worry about writing out exact Javascript code. We are looking for a precise yet succinct description of how such a design would work—a few sentences.

Answer: Implement lab 6's rewriter in Javascript (part of the trusted library), export a `sandbox_eval` function, which first invokes the rewriter on the string passed to it, and then passes it to the real `eval`.

Another answer: implement a Javascript interpreter, and when accessing a variable `y` in that interpreter, access the variable `sandbox_y` in the outer "world".

II File system encryption

Ben Bitdiddle is designing an alternative to BitLocker that performs encryption at the file system level instead of disk sector level. Ben's encryption scheme works at the level of inodes (i.e., files or directories). Each inode is structured as follows:

```
struct inode {
  /* Encrypted parts of the inode */
  int inode_type;
  /* ... */
  uint64_t data_sectors[64];

  /* Non-encrypted parts of the inode */
  mac_tag inode_tag;
  mac_tag data_tag;
};
```

Each `struct inode` (except for the two tags at the end) is encrypted with AES-CBC using the master encryption key K and an IV of $E_K(inum)$, where *inum* is the inode number. The data sectors that comprise a file or directory are pointed to by the `data_sectors` array (which stores up to 64 sector numbers, in this simplified design). The contents of each data sector is encrypted with AES-CBC using the master encryption key K and an IV of $E_K(offset)$, where *offset* is the offset of that sector's data in the containing file or directory.

The inode's `inode_tag` is a MAC tag over the contents of the encrypted part of the inode, using the master authentication key A. The `data_tag` is a MAC over the contents of all data sectors of that inode, using the master authentication key A.

3. [8 points]:

Suppose an adversary obtains a laptop running Ben's encrypted file system, and cannot log in, but the laptop's software allows the adversary to create temporary files with arbitrary contents. How could the adversary gain access to Ben's data? Assume the adversary knows the inode number and sector numbers for any system file, such as `/etc/passwd`, since they are typically predictable. The adversary can power off the laptop at any time, take out the disk, and inspect/modify raw disk contents.

Answer: Write a replacement `passwd` file as a temporary file, containing a root account with a known password. Then find the inode and data sectors corresponding to `/etc/passwd`, and copy the temporary file's data sectors and data MAC tag to the data sectors and data MAC tag of `/etc/passwd`. Then boot up the laptop and log in as root.

III Network security

According to Mark Silis, if a firewall was deployed at MIT, would it make the following system less vulnerable to attacks, more vulnerable to attacks, or the same? Circle the best answer for each question.

4. [3 points]: MIT's building control systems, in the context of remote buffer overflow attacks.

A. More vulnerable.

B. Less vulnerable.

C. Just as vulnerable.

Answer: B. The firewall makes it more difficult for outside adversaries to connect to vulnerable building control systems.

5. [3 points]: MIT's web servers, in the context of DDoS attacks.

A. More vulnerable.

B. Less vulnerable.

C. Just as vulnerable.

Answer: A. The firewall itself may be unable to deal with a large volume of traffic from a DDoS attack, even if the web servers are able to deal with that traffic.

6. [3 points]: MIT's domain name, in the context of attacks like the hijacking in January 2013.

A. More vulnerable.

B. Less vulnerable.

C. Just as vulnerable.

Answer: C. The hijacking attack did not involve MIT's network at all; it took place at the EDUCAUSE registrar.

IV User authentication

The Secure Remote Password (SRP) is a well-understood, public-domain client/server password scheme based on a challenge-response protocol that has the following nice properties:

- User and server authenticate each other without any other parties involved;
- Password is never sent over the wire;
- Password (or equivalent) is not stored at the server;
- Adversary who steals server data cannot masquerade as client; and
- Using the same password with two different servers results in different server-side data.

7. [4 points]:

Which of the security criteria from "The Quest to Replace Passwords" does SRP meet (meaning solid circle from the paper)? Answer "False" for open-circle (almost offers the benefit).

(Circle True or False for each choice.)

A. True / False S1 Resilient-to-Physical-Observation. **Answer:** False.

B. True / False S4 Resilient-to-Unthrottled-Guessing. **Answer:** False.

C. True / False S5 Resilient-to-Internal-Observation. **Answer:** False.

D. True / False S6 Resilient-to-Leaks-from-Other-Verifiers. **Answer:** True.

E. True / False S9 No-Trusted-Third-Party. **Answer:** True.

F. True / False S10 Requiring-Explicit-Consent. **Answer:** True.

G. True / False S11 Unlinkable. **Answer:** True.

Ben Bitdiddle is worried that PBKDF2 is not good enough for storing passwords in his web site, since it takes just 50 msec to compute the hash for a given password and salt combination. Instead, Ben comes up with the following scheme (in Python), which applies PBKDF2 in a chain to each letter in the password, using the previous letter's hash value as the salt (and using the initial salt for the first letter):

```
def store(pw, salt):
  result = [salt]
  for c in pw:
    h = pbkdf2.PBKDF2(c, result[-1]).hexread(32)
    result.append(h)
  return result

def check(pw, stored):
  if len(pw) != len(stored) - 1:
    return False
  for cpos, c in enumerate(pw):
    h = pbkdf2.PBKDF2(c, stored[cpos]).hexread(32)
    if stored[cpos+1] != h:
      return False
  return True
```

To store a password, Ben's web site stores the result of the `store()` function, and to check if a password supplied by some web browser is correct, Ben's web site passes it to the `check()` function, along with the previously stored hash of the user's password, and returns an error to the web browser if the password does not match (i.e., `check` returned `False`).

8. [10 points]: Explain how an adversary can break into any account on Ben's web site, which uses Ben's modified password scheme described on the previous page. Assume the adversary does **not** steal the stored passwords from the server.

Answer: The password can be extracted through a timing attack. First, submit passwords of different lengths, and detect which one takes relatively longer to check; that one will be a password of the right length. Then vary all possible first letters in the password; one of them should take longer, which means the `for` loop moved on to the second letter in that case. Repeat to guess each letter in turn.

V Privacy

9. [6 points]:

Tor uses TLS (which provides confidentiality and integrity) between onion routers, and encrypts cells traversing these routers with AES (see Figure 1). In addition, Tor checks integrity at the edges of each stream. Why is it necessary to perform end-to-end stream integrity in addition to TLS between routers?

Answer: Without end-to-end integrity checking, malicious routers may be able to change the ciphertext of cells in sensible ways that are undetectable at the edges. This will allow a malicious router in the middle of a circuit to corrupt packets and perhaps observe which outgoing packets from some exit node get corrupted as a result.

VI Android security

The manifest for the FriendTracker application is as follows:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="org.siislab.tutorial.friendtracker"
          android:versionCode="1" android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".FriendTrackerControl" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <provider android:authorities="friends" android:name="FriendProvider"
                  android:readPermission="org.siislab.tutorial.permission.READ_FRIENDS"
                  android:writePermission="org.siislab.tutorial.permission.WRITE_FRIENDS">
        </provider>

        <service android:name="FriendTracker" android:process=":remote"
                 android:permission="org.siislab.tutorial.permission.FRIEND_SERVICE">
        </service>

        <receiver android:name="BootReceiver">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"></action>
            </intent-filter>
        </receiver>
    </application>

    <permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></permission>
    <permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></permission>
    <permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></permission>
    <permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE_ADD"></permission>
    <permission android:name="org.siislab.tutorial.permission.FRIEND_NEAR"
                android:label="@string/permlab_friendNear"
                android:description="@string/permdesc_friendNear"
                android:protectionLevel="dangerous"></permission>
    <permission android:name="org.siislab.tutorial.permission.BROADCAST_FRIEND_NEAR"></permission>

    <uses-permission android:name="org.siislab.tutorial.permission.READ_FRIENDS"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.WRITE_FRIENDS"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.FRIEND_SERVICE"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.FRIEND_NEAR"></uses-permission>
    <uses-permission android:name="org.siislab.tutorial.permission.BROADCAST_FRIEND_NEAR"></uses-permiss
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
</manifest>
```

10. [6 points]: The manifest, shown on the previous page, states several permissions that FriendTracker needs (e.g., the `[...].ACCESS_FINE_LOCATION`). `[...].ACCESS_FINE_LOCATION` is a dangerous permission. What or who decides whether FriendTracker should be allowed to have this dangerous permission?

Answer: The user, when installing the application.

11. [6 points]: What permissions should the manifest of the FriendViewer application request to be able to write to the `friends` provider in the FriendTracker application?

Answer: org.siislab.tutorial.permission.WRITE_FRIENDS.

12. [6 points]: Consider the setup in Figure 2 of the Android paper from lecture. Where else, in addition to the shown manifest, does the `BROADCAST_FRIEND_NEAR` permission show up? Be specific: which applications, where in those applications (code or manifest), and why does `BROADCAST_FRIEND_NEAR` show up there?

Answer: The sender of broadcast intents (in the FriendTracker app) will use it in its code, and the receivers of broadcast intents will use it in their manifests.

VII CryptDB

The proxy rewrites SQL queries to perform them over encrypted columns stored on server. An encrypted column may be an onion with several layers of encryption. Assume that the server has a `employee` table with three columns: `name`, `SSN`, and `salary`, and that all three are encrypted using an onion.

13. [6 points]: To what query does the proxy rewrite the query `select SUM(salary) from employee where name="Alice"`? (If CryptDB removes any onion layers, please explain which onion layers are removed.)

Answer: Need to remove the RND layer of the DET onion for `name`. The resulting query will be: `select CRYPTDB_SUM(salary_hom) where name_det = DET(JOIN("Alice"))`, with column names properly anonymized.

14. [6 points]: Even with several onions, CryptDB cannot rewrite the query `select * from employee where salary + ssn > 50`. Explain why and propose an extension of CryptDB that would be able to execute it.

Answer: Computing salary+SSN on the server will produce a HOM value, which cannot be compared using > because it's not OPE. Extensions: fetch both salary and SSN to the client and finish computing there. Or pre-compute an OPE column containing the sum salary+SSN. Using FHE is not a viable alternative, because FHE does not allow the server to determine if the `salary+ssn>50` condition is true or false for a given row; FHE produces an encrypted boolean result of that condition, which the server cannot decrypt.

15. [6 points]: Recall that CryptDB's design as described in the paper uses three different onions for each encrypted column (each with different layers). Consider an alternative design which uses a single onion with three layers: OPE (innermost), DET (middle), and HOM (outermost). For each of the three operations (+ or `SUM`; =; and >), explain whether they can be performed using this onion, and why. If some operation cannot be performed with this alternative design, but *can* be performed with CryptDB, give an example query.

Answer: Can perform OPE and DET, but cannot perform HOM, because it would add DET ciphertexts, instead of the plaintext values. Query: `SELECT sum(salary) FROM employee;`

VIII Code obfuscation

16. [9 points]:

For each of the following techniques used in the "Looking inside Dropbox" paper, would the technique be useful in de-obfuscating a web application written in Javascript, given the Javascript source code running in the user's web browser?

(Circle True or False for each choice.)

A. True / False Reverse-engineering opcode remapping would be useful. **Answer:** False.

B. True / False Monkey-patching would be useful. **Answer:** True.

C. True / False Intercepting network requests before they are encrypted with SSL would be useful. **Answer:** True.

IX 6.858

We'd like to hear your opinions about 6.858. Any answer, except no answer, will receive full credit.

17. [4 points]: We have noticed that lecture attendance is noticeably lower than quiz attendance. Why do you think this is, and what do you think might make lectures worth attending?

Answer:

- 25x Lectures should cover material not in papers
- 13x Lectures are too long
- 12x More demos/examples
- 11x Later hours
- 6x Can watch lectures outside of class
- 4x In-lecture quizzes
- 4x Different room
- 3x More interactive discussions
- 2x Too many papers/didn't read papers
- 2x Recitations
- 2x Participation in-class
- 2x More interesting papers
- 2x Make lectures relate to labs
- 1x More technical lectures
- 1x Cover more recent stuff
- 1x Better notes

End of Quiz

6.858 Computer Systems Security

Fall 2014