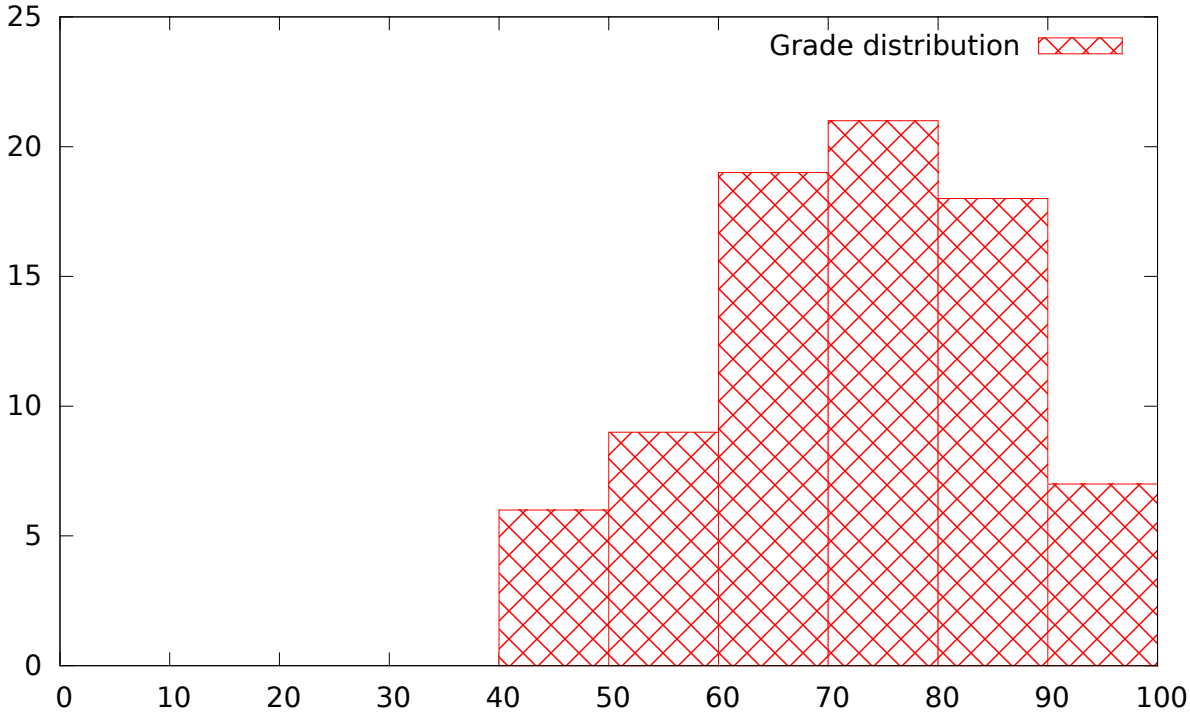
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2014

Quiz II Solutions



Histogram of grade distribution

I User authentication

1. [8 points]: A company named Vault wants to offer a secure, cloud-based backup system. When the user updates a local file, her Vault client opens a TCP connection to a Vault server, and uses the Diffie-Helman protocol to establish a secret symmetric key K with the server. Then, the client generates this string s:

$$s = \langle \text{ documentName, documentContent, userName, userPassword, randomNumber } \rangle$$

and sends the following message to the Vault server:

$$E_K(s, \text{HMAC}_K(s))$$

where $E_K(m)$ denotes encrypting message m using key K, and $\text{HMAC}_K(m)$ denotes computing an HMAC message authentication code of message m using key K.

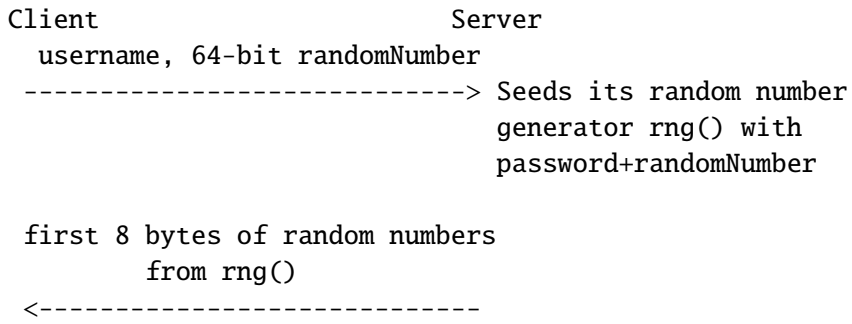
The server decrypts the message, verifies the user's password, and verifies the integrity of the message using the HMAC. If all of the checks succeed, the server stores the document. If the server sees more than 10 messages with the wrong password, all future accesses to that account are blocked.

How can a network attacker reliably obtain the user's password?

Answer: The Vault server does not authenticate itself to the client. Thus, a man-in-the-middle attacker can impersonate the Vault server and steal the user's password.

2. [8 points]:

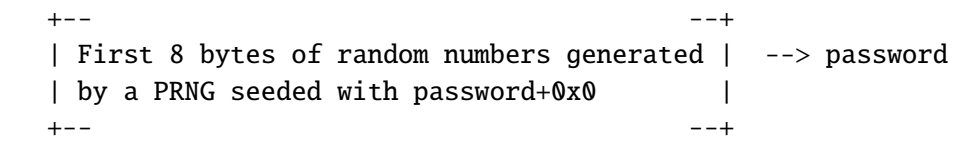
Suppose that a user wants to verify that a server knows the user's password. The user and the server engage in the following challenge/response:



Everyone knows which algorithm the server uses for `rng()`, so the client can validate that the server's response contains the expected bytes.

Suppose that many different servers use this protocol. Further suppose that the attacker has a list of popular usernames, and a separate list of popular passwords. Explain how an attacker can abuse the protocol (without otherwise compromising any servers) to build a rainbow table of passwords and easily determine the passwords of many users.

Answer: The attacker gets to pick the random number, so he always sets it to the same value, say, 0x0. The attacker can then iterate through a list of popular passwords, and, for each one, build a table entry like this:



The attacker goes to a server and, for each popular username, submits <username, 0x0>. The attacker then looks up the returned bytes in the table to determine the password.

II Tor and Private browsing

Imagine that the website `https://foo.com` embeds a JavaScript file from `http://attacker.com`. Suppose that a user's browser allows an HTTPS page to run JavaScript code fetched from an HTTP URL; however, the code cannot read or write any persistent client-side state. For example, the JavaScript code cannot read or write cookies, nor can it read or write DOM storage. The browser also ensures that the `attacker.com` web server cannot set client-side cookies using the `Set-Cookie` header, or receive client-side cookies in the HTTP request for the JavaScript file.

Suppose that the user visits `https://foo.com` once in private browsing mode, closes the tab, and then visits the site again in regular browsing mode; in both cases, the user's web traffic goes through Tor. The `attacker.com` web server would like to determine with high likelihood that the same user has visited the site twice. However, the attacker does not control any Tor nodes.

3. [8 points]: Why is it unlikely that the `attacker.com` server can use TCP fingerprinting to identify the user? Recall that TCP fingerprinting involves looking at TCP connection parameters, such as the initial TCP window size, TCP options, TCP flags, etc.

Answer: The user's computer will establish a TCP connection to the first Tor relay node, not to `attacker.com`. The TCP connection seen by `attacker.com`, and thus all of the TCP fingerprinting features, will come from the exit node in the user's circuit, which is independent of the user's TCP stack.

4. [8 points]: Describe a way that the attacker can fingerprint the user with a much higher likelihood of success.

Answer: The attacker's JavaScript can use Panopticlick-style techniques to fingerprint the user's browser. The JavaScript sends this information back to the attacker server using AJAX. The attacker keeps a database of fingerprints and tracks when the same fingerprint is seen twice.

5. [8 points]: Browsing the web through Tor can be slow. This is because user traffic is forwarded between volunteer computers that are scattered across the world, overburdened with traffic, and potentially situated behind slow network connections.

Suppose that CloudCo, a large technology company with datacenters scattered across the world, offers some of its machines to the Tor community for use as entry and exit nodes. CloudCo machines have plentiful RAM and CPU; CloudCo machines also have low latency, high-bandwidth network connections to all major ISPs. By using CloudCo machines as Tor entry and exit nodes, users could ensure that Tor congestion would only exist in the middle of a circuit.

Assume that CloudCo genuinely wants to help Tor users, and that CloudCo configures its machines to faithfully execute the Tor protocol. Why is it still a bad idea for users to employ CloudCo machines as entry and exit nodes?

Answer: If the attacker monitors or subverts the first and last nodes in a Tor circuit, the attacker can deanonymize the circuit's user via statistical techniques that correlate the traffic that enters the entry node and leaves the exit node. So, if an attacker manages to subvert CloudCo, he can deanonymize a large set of users. Even if the attacker can only determine where CloudCo entry and exit nodes are located, the attacker can launch statistical attacks.

III TaintDroid

6. [10 points]: TaintDroid defines various sources of taint, and a unique taint flag for each of those sources (e.g., `IMEI`, `PHONE_NUM`, `CAMERA`, etc.).

For the application code below, list the set of taint flags that each variable will have *after* each line of code has executed (for example, `"IMEI, CAMERA"` or `"PHONE_NUM"`). If a variable will not have any taint flags, write an Ø.

```
int x = android.getIMEI();                  x:_____________
int y = android.getPhoneNum();              y:_____________
int z;
if(x > 5000){
    z = 0;                                  z:_____________
}else{
    z = 1;                                  z:_____________
}

int arr[] = new int[2];                     arr:___________
arr[0] = x;                                 arr:___________
arr[1] = y;                                 arr:___________

char buf[] = android.getCameraPicture();    buf:___________
int val = buf[x%2];                         val:___________

x = 42;                                     x:_____________
```

	`int x = android.getIMEI();`	x: IMEI
	`int y = android.getPhoneNum();`	y: PHONE_NUM
	`int z;`	
	`if(x > 5000){`	
	`  z = 0;`	z: Ø
	`}else{`	
	`  z = 1;`	z: Ø
	`}`	
	`int arr[] = new int[2];`	arr: Ø
	`arr[0] = x;`	arr: IMEI
	`arr[1] = y;`	arr: IMEI, PHONE_NUM
	`char buf[] = android.getCameraPicture();`	buf: CAMERA
	`int val = buf[x%2];`	val: IMEI, CAMERA
Answer:	`x = 42;`	x: Ø

IV Timing side channels

Consider the timing attack on OpenSSL RSA keys, described in the paper by Brumley and Boneh.

7. [8 points]: Suppose that OpenSSL was modified to never use Karatsuba multiplication (and instead always use "normal" multiplication), but was still using Montgomery multiplication as described in the paper. Would you expect the attack to still work with a few million queries? Explain why or why not.

Answer: Should still work: the zero-one gap will remain positive for all bits.

8. [8 points]: Suppose that OpenSSL was modified to never use Montgomery multiplication, but was still using both Karatsuba and normal multiplication as described in the paper. Would you expect the attack to still work with a few million queries? Explain why or why not.

Answer: Probably hard to make it work: no zero-one gap for the first 32 bits, so need to guess all 32 high bits at once, requiring on the order of 2^32 queries.

V Embedded device security

9. [8 points]: Ben Bitdiddle has a smartphone with an always-on voice recognition system, which runs any commands that it hears.

Alyssa P. Hacker wants to trick Ben's phone into running a command, but Ben turns off all wireless radios on the phone as a precaution to prevent Alyssa from breaking in over the network, and also keeps his phone in a locked sound-proof room in hopes of foiling Alyssa. After hearing Kevin Fu's guest lecture, Alyssa figures out how she can get Ben's phone to run a command of her choice, without breaking into Ben's room. What is Alyssa's plan?

Answer: Inject EM interference with the voice command modulated at the appropriate frequency so as to produce the desired sound waveform in the microphone on Ben's smartphone.

VI Android security

Ben Bitdiddle is still excited about his phone's voice recognition system, and decides to set up a system that allows third-party applications to handle user voice commands (e.g., his music player might want to support a command like "next song", and his Facebook application might want to support a command like "post my current location on Facebook").

Ben's design runs on Android, and involves a single voice recognizer application that runs with access to the microphone. This voice recognizer transcribes whatever sounds it hears into ASCII text messages, and hands these messages off to third-party applications.

10. [12 points]: Describe a design for allowing the voice recognizer to securely send the transcribed messages to third-party applications, and allowing the third-party applications to securely receive them. Be as specific as possible; do not worry about software bugs. To help you structure your answer, consider the following:

First, what new permission labels do you propose to create? For each new permission label needed in your design, specify (1) its name, (2) who creates the label, and (3) what the type of the permission label is.

Answer: The voice recognizer should create two labels: recognizer.SEND (signature) and recognizer.LISTEN (dangerous).

Continued on the next page.

Second, how should the voice recognizer securely send the transcribed messages? Describe (1) what component(s) the recognizer should have, (2) what permission(s) the application should ask for, (3) what label(s) should protect the recognizer's component(s), and (4) what other security-relevant steps the recognizer's code has to take.

Answer: The recognizer should have a listening service. The voice recognizer should ask for the SEND permission. When sending a message, it should do `sendBroadcast(msg, recognizer.LISTEN)` to ensure that the message is received only by applications that have the LISTEN privilege.

Third, how should the third-party applications securely receive the messages? Describe (1) what component(s) each application should have, (2) what permission(s) the application should ask for, (3) what label(s) should protect the application's component(s), and (4) what other security-relevant steps the application code has to take.

Answer: Applications should create a broadcast receiver component, and protect it with the SEND label, to ensure that only the voice recognizer can send it messages. Applications should ask for the LISTEN permission to ensure they can get the messages.

VII Labs

11. [8 points]: Ben Bitdiddle's lab 6 code rewrites the following profile code:

```
var x = y[z];
```

into:

```
var sandbox_x = sandbox_y[sandbox_z];
```

Write down an exact piece of profile code that an adversary could use to call `alert(123);`.
Answer: function bogus() ; var x = 'constructor'; var y = bogus[x]; y('alert(123)')();

VIII 6.858

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

12. [6 points]: Now that you are almost done with 6.858, how would you suggest we improve the class in future years?

Answer:

Organizational structure. 7x Smaller homeworks or in-class assignments. 3x More labs. 2x More office hours (e.g., Friday). 2x Introduce recitations; suggest supplemental background reading. Discuss labs in class or recitation. Bring back James Mickens. Come up with some way to try out group members before final project.

Administrative stuff. 4x Post lecture videos quicker; post all before quiz. 2x Avoid labs due on quiz week. Post guest lecture slides. Better organized lecture notes.

Quiz. 2x Avoid tricky quiz questions; line up quiz and lecture material. Avoid quiz on Monday after Thanksgiving.

Labs. 6x More attack-oriented assignments / labs. 6x Start the projects earlier; more checkpoints and feedback. 4x Add an Android/iOS lab (maybe instead of lab 6). 4x Network attack lab. 3x Balance out lab difficulty (very uneven now); e.g., labs 5 vs 6. 3x Give up on trying to make lab 6 secure. 2x Intermediate deadline / more guidance for lab 5. 2x Less repetitive lab exercises (later parts of lab 2). 2x Make lab 4 more substantial. 2x Clarify lab 3 instructions. 2x More context for lab 6. Randomize project groups. CTF lab. More labs attacking other students' code. Improve make check for lab 5, Chrome vs Firefox. More guidance, explanations for lab assignments. Learn how to prevent CSRF in some lab. Add a lab on Tor.

Reading assignments. 11x Move the 10pm question deadline later. 2x Give more feedback on submitted paper questions. 2x Ask more relevant reading questions; answer them in lecture. Make it optional to come up with a reading question. Allow submitting lecture questions after deadline, even if that means getting no credit.

Lectures. 6x More in-class examples / demos. 3x Shorter lectures, or 5-minute break in the middle. 3x Post key points / takeways for papers. 2x Spend part of lecture giving intro / background for next paper. 2x Fewer papers. 2x Make lectures overlap less with assigned reading. Guiding notes on complex papers. Schedule the lectures later in the day. Differentiate between broad background and specific paper focus in lectures.

Topics / papers. 4x More connection between lab and lecture topics. 3x Bring back BitLocker / FS encryption from last year. 3x Guest lectures were cool. 3x Talk about social engineering. 2x Discuss more contemporary security bugs and risks, such as stuxnet. 2x More about web security. 2x Have lectures cover more than just the readings. Remove medical devices lecture. Use an up-to-date Kerberos paper. Android/TaintDroid was boring. More about designing secure protocols. Timing attack paper was way too difficult. More case studies of real-world systems. Less overlap of lab and readings. More on security mindset, how to approach security problems. More language-based security. More guidance on how to design secure software. Tangled Web and UrWeb hard to read (but cool systems). More on network attacks (ARP spoofing). Bring back CryptDB. Private browsing paper was useless.

End of Quiz