Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2012

Quiz II

You have 80 minutes to answer the questions in this quiz. In order to receive credit you must answer the question as precisely as possible.

Some questions are harder than others, and some questions earn more points than others. You may want to skim 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 and legible. If we can't understand your answer, we can't give you credit!

Write your name and submission website username (typically your Athena username) on this cover sheet.

This is an open book, open notes, open laptop exam.
NO INTERNET ACCESS OR OTHER COMMUNICATION.

Please do not write in the boxes below.

I (xx/17)	II (xx/16)	III (xx/10)	IV (xx/18)	V (xx/24)	VI (xx/12)	VII (xx/6)	Total (xx/103)

Name:

Submission website username:

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.

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.

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).

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?

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**).

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.

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?

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 $\langle$trinket-id, counter-id$\rangle$ 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(..)`?

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)?

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)?

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?

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?

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

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?

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

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

End of Quiz

6.858 Computer Systems Security

Fall 2014