Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Fall 2010

Quiz II

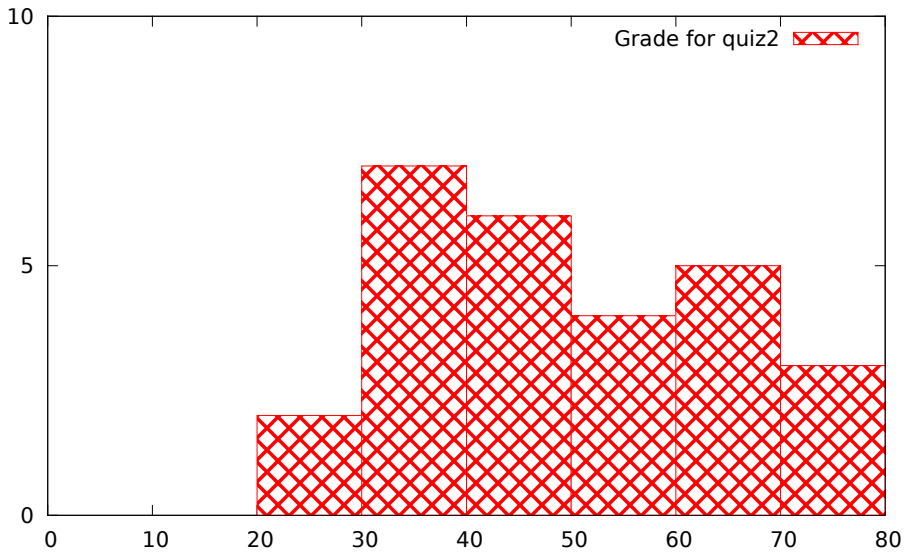
All problems are open-ended questions. In order to receive credit you must answer the question as precisely as possible. You have 80 minutes to finish this quiz.

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).

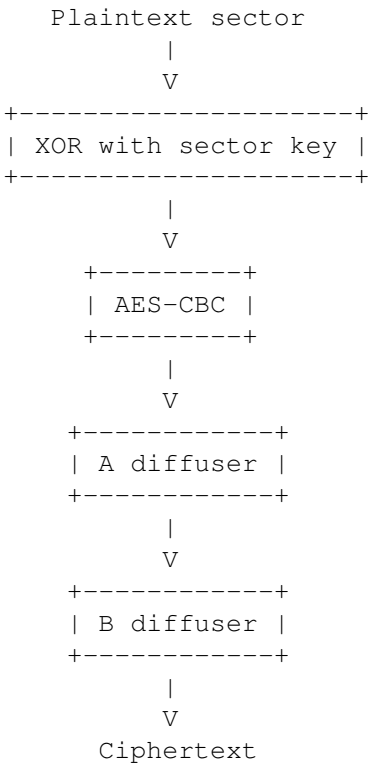
7. [5 points]: What, precisely, can a client safely assume about its SSL connection and about Ben's server, if it validates the response (steps 5a, 5b, and 5c)? Assume that no attackers have compromised any TPM chips or certificate authorities.

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