EECS 398 December 14, 2010
Intro to Computer Security Final Exam

Final Exam

You must work by yourself and abide by the College of Engineering Honor Code. You may consult your notes; you are also permitted to use books or Internet references, though this should not be necessary to answer the questions. For each response, you must **cite any sources** that you used, other than your notes.

This exam is due **Tuesday, December 21 at 5pm**. Email your answers to `eecs398@umich.edu` with the subject line "Exam Submission". Please attach your solutions in a `.txt` or `.pdf` file named "*`your uniqname`*`_final`". You should receive a confirmation email within 15 minutes.

Answer all four of the numbered questions below. Show your work.

1. **Hashes and passwords.** Suppose you are in charge of security for a major web site, and you are considering what would happen if an attacker stole your database of usernames and passwords. You have already implemented a basic defense: instead of storing the plaintext passwords, you store their SHA-256 hashes.

   Your threat model assumes that the attacker can carry out 4 million SHA-256 hashes per second. His goal is to recover as many plaintext passwords as possible from the information in the stolen database.

   Valid passwords for your site may contain only characters a–z, A–Z, and 0–9, and are exactly 8 characters long. For the purposes of this homework, assume that each user selects a random password.

   (a) On average, how long would it take for the attacker to crack a single password by brute force? How large a botnet would he need to crack one password per hour, assuming each bot can compute 4 million hashes per second?

   (b) Based on your answer to part (a), the attacker would probably want to adopt more sophisticated techniques. You consider whether he could compute the SHA-256 hash of every valid password and create a table of $(\textit{hash}, \textit{password})$ pairs sorted by hash. With this table, he would be able to take a hash and find the corresponding password very quickly.

   How many bytes would the table occupy?

   It appears that the attacker probably won't have enough disk space to store the exhaustive table from part (b). You consider another possibility: he could use a *rainbow table*, a space-efficient data structure for storing precomputed hash values.

   A rainbow table is computed with respect to a specific set of N passwords and a hash function H (in this case, SHA-256). We construct a table by computing *m chains*, each of fixed length k and representing k passwords and their hashes. The table is constructed in such a way that only the first and last passwords in each chain need to be stored: the last password (or *endpoint*) is sufficient to recognize whether a hash value is likely to be part of the chain, and the first password is sufficient to reconstruct the rest of the chain. When long chains are used, this arrangement saves an enormous amount of space at the cost of some additional computation.

   Chains are constructed using a family of *reduction functions* $R_1, R_2, \ldots, R_k$ that deterministically but pseudorandomly map every possible hash value to one of the N passwords. Each chain begins with a different password p_0. To extend the chain by one step, we compute $h_i := H(p_{i-1})$ then apply the ith reduction function to arrive at the next password, $p_i = R_i(h_i)$. Thus, a chain of length 3 starting with the password `hax0r` would consist of (`hax0r`, $R_1(H($`hax0r`$))$, $R_2(H(R_1(H($`hax0r`$))))$).

After building the table, we can use it to quickly find a password p_* that hashes to a particular value h_*. The first step is to locate a chain containing h_* in the table; this requires, at most, about $k^2/2$ hash operations. Since h_* could fall in any of $k-1$ positions in a chain, we compute the password that would *end up* in the final chain position for each case. If we start by assuming h_* is right before the end of the chain and work backwards, the possible endpoints will be $R_k(h_*)$, $R_k(H(R_{k-1}(h_*)))$, $\ldots$. We then check if any of these values is the endpoint of a chain in the table.

If we find a matching endpoint, we proceed to the second step, reconstructing this chain based on its initial value. This chain is very likely to contain a password that hashes to h, though collisions in the reduction functions cause occasional false positives.

(c) For simplicity, make the optimistic assumption that the attacker's rainbow table contains no collisions and each valid password is represented exactly once. Estimate the size of the table in terms of the chain length k. If $k = 5000$, how many bytes will the attacker's table occupy? Roughly estimate how long it takes to construct the table if the attacker can add 2 million chain elements per second. Compare these size and run-time estimates to your results from (a) and (b).

(d) Based on your analysis, you consider making the following change to the web site: instead of storing SHA-256(*password*) it will store SHA-256(*server_secret*||*password*), where *server_secret* is a randomly generated 32-bit secret stored on the server. (The same secret is used for all passwords.)

How well does this design defend against a rainbow table attack? Can you adjust the design to provide even stronger protection?

2. **Web security.** A large Midwestern research university wants to implement a central sign-on facility where users authenticate themselves to an official site then receive a token that confirms their identity to all other campus sites.

(a) Assuming the protocol is competently implemented and deployed, how might deploying this service improve security on campus? How might it hurt security?

(b) Besides using HTTPS, what else could the sign-on site do to help prove its identity to users?

(c) Suppose the sign-on protocol proceeds as follows: When the user visits site A, which requires authentication, site A redirects the user to the central sign-on site. Following authentication, the central sign-on site redirects the user's browser back to an HTTPS URL at site A with the following parameters: u, the user's username, and $\text{Sign}_{\text{SecKey}}(u)$, a digital signature produced with the sign-on site's secret key. (Assume that the corresponding public key is widely known.)

If site A is controlled by an attacker, how can it impersonate the user to other sites that trust the sign-on protocol?

(d) Propose a simple change to the protocol that would fix the problem identified in (c).

3. **Authorization** Wikipedia defines a *credential* as “an attestation of qualification, competence, or authority issued to an individual by a third party with a relevant or *de facto* authority or assumed competence to do so.” Each of the items below effectively serves as a credential. For each, describe: (i) the authority or access that is granted, (ii) the third party that issues the credential, (iii) the authorized user of the credential.

   (a) Your uniqname and password

   (b) A self-signed public key certificate generated by you

   (c) A dollar bill

   (d) A $50 Zingerman’s gift card

   (e) The URL http://www.google.com/

   (f) The VeriSign SSL Certificate Authority’s private key

4. **Short essay.** Respond to each of the following prompts in two or three paragraphs.

   (a) “Security through obscurity,” keeping parts of a system’s design secret so that potential weaknesses are not exposed, is often criticized as a naïve security strategy. What’s wrong with it? When can it nevertheless be beneficial?

   (b) Tomorrow you discover a flaw in the DNS protocol that allows a remote attacker to easily poison an ISP’s DNS cache and redirect its users to impostor sites. What harm can this cause? Whom do you contact? How and when do you disclose the problem?

   (c) Imagine you are in charge of security for the Wikileaks project. What threats would you worry about? What defenses would you deploy? Justify your answers in terms of risk level, cost, and benefit.

   (d) Imagine you are in charge of security for the U.S. State Department’s document repository. What threats would you worry about? What defenses would you deploy? Justify your answers in terms of risk level, cost, and benefit.