Jung Yoon (jey283)
CS 361S - Network Security and Privacy
Spring 2015
Professor O. Jensen

Homework #1

Write the number of late days you are using: 3

1) **Molvania**

   n * (7 * 26)^2

   - 26 because it says that it transforms to uppercase so we're assuming that it's only uppercase letters that are involved

2) **Hash Function Properties**

   a. Only thinking in terms of Jerko's credibility and not considering the possibility of Zlad trying to cheat (so assuming that Zlad is a completely fair player, otherwise we'd need one-wayness and weak-collision resistance to protect the answer), it needs collision-resistance so that Jerko can have credibility as far as proving that he really did have the correct solution earlier. If it didn't have collision resistance then who is to say that Jerko and Zlad didn't have two different answers but ended up with the same hash?

   b. Weak-collision resistance (second-preimage resistance) so that potential malicious users will find it difficult to find ways to overwrite protected files without detection. It might also be best to have collision-resistance so that we can prevent having the possibility of having potential hash matches, thus making it even more difficult for malicious users to find ways to overwrite protected files without detection, since images are not random.

   c. One-wayness. You don't want Zlad to be able to work backwards to find the signatures.

   d. Weak-collision resistance so that it's difficult for Zlad to find and pass on a message that's in the log file. Again, like (b) it might just be better to have collision-resistance when considering the entire system instead of just Zlad's case so that things like this can be avoided in the first place.

   e. Collision resistance because it makes it difficult for the array's values to be overwritten by multiple strings. That is, without collision resistance we could have multiple strings (s1,s2,s3...sn) try to insert into the same a[i] where i = hash(these_multiple_strings s1,s2,s3...sn). This would render the "datastructure" faulty.

3) **MMAC**

The MH function takes in two parameters and returns a 320-bit, whereas the SHA-1 has one parameter and returns half of that, 160 bits. We could make it so that the MH function encodes the two parameters in two batches using SHA-1 (so the key would return 160 bits and the message would return 160 bits) and then concatenates the two to return 320 bits. This makes sure to keep the design one-way, collision resistant, hide the key, but it makes it vulnerable to forging, just as the SHA-1 has its vulnerabilities.

4) **Zbox Half**

a. Jerko is a network attacker. If Fyodor is not using a strong enough cryptographic encryption service or if the data isn't encrypted in the first place, Jerko can exploit the signal leak by traversing the network and reading Fyodor's data, Jerko now has access to whatever messages Fyodor is sending, which includes sent passwords, etc. With this new data, Jerko can modify his Zbox's serial number and he will also have access to Fyodor's password so he would be able to use Fydor's Half-Alive server for free. It's really Zbox Half-Alive's fault for having such a crummy authentication system and Fyodor's fault for not using a secure network.

b. A possible user authentical scheme that prevents passive attackers from exploiting eavesdropped messages between the Zbox and the Half-Alive server would be a mutual authentication scheme. Since Jerko "cannot modify them or introduce new messages" anyway and won't be able to use the man-in-the-middle tactic, we could use a key exchange scheme.

   Perhaps we could use the Diffie-Hellman key exchange. Without explaining the entirety of the protocol since it's pretty well known, basically the idea would be that each sides would have a secret key/integer and through a series of "disguised" messages they would both get a shared secret key/integer, "disguised" because they never actually send the secret integers to each other directly.

   Then, using this secret key, the client could send over passwords and serial codes in a format Authen(key, message) where the message is either the password or the serial code. Note that this wouldn't happen every time the client connected but rather only the very first time that they connected, meaning that the shared key/integer would stay the same indefinitely in order to prevent passive attackers from exploiting eavesdropped messages.

   This is far from being a perfect fix because the server/client authentication scheme is still vulnerable to other types of attacks but it gets the job done.

5) **T3N3X**

Since “recovering from a page fault requires swapping a page into memory and thus takes much longer than reading a page that is already in physical memory”, we can actually exploit the time difference in reading from the same page and reading from a page not in physical memory to guess passwords. This works since “your timer is granular enough to detect a pagefault.”

That is, we could put the first character of the proposed password (passwordArgument) in one page and the rest of the characters in another page. If we get a page fault, as detected by the timer, we know that the first character was correct. If we don’t get a page fault, we know that the first character is incorrect. Then, if it was correct we move the second character to first page along with the first character and repeat the process over and over again until the entirety of the passwordArgument is on the first page and the password is correct.

6) **HMAC**

a. sdf
b. sdf

7) **Brobra.mi**

a. We can’t do sidejacking or surf jacking since the victim is offline and unreachable. There is no authenticator and the passwords are not hashed in any way so given a username, we get retrieve it since it’s stored as a plaintext. Of course, we’re assuming that the victim’s computer is unreachable so that might be more difficult.

   Given a username, it might be easier to tamper with an actual cookie and send it back to the server with the new password (oldUsername: newPassword) so that now, we have access to the site with our same username. The site only keeps track of the username and verifies that the password saved in the cookie matches the password entered so there should not be a problem logging into the account.

b. Adding an authenticator would solve the issue. We’d add a HMAC to every cookie such that HMAC(server’s secret key, session information). This means that with each request, browser presents the cookie; server recomputes and verifies the authenticator. This makes sure that the exchanges between the user and the server are authentic and makes sure that the Authenticators are unforgeable and tamper-proof.