Project 1

Yanshu Hong, Nat Roth
CS255: Cryptography and Computer Security

February 12, 2015

Question 1. All the passwords are padded to 65 bytes (one more byte than the largest possible password allowed) before being encrypted. This means that all passwords have the same length and thus the encryption does not leak the lengths of the passwords.

Question 2. Each padded password is concatenated with the HMAC of its domain before being encrypted. The value in the KVS is the encryption of this concatenation. Every time the user requests the password for a domain, the value is decrypted and split into padded password and HMAC (because the padded password has a constant length, the splitting point is known). Then this HMAC will be compared with the HAMC of the domain the user passed in. If they're different, an exception will be thrown.

If a swap attack occurs, then the HMAC of the key will not match the HMAC encrypted inside the value. Therefore, the GET() function can detect any swap.

Question 3. Yes, the trusted location must exist in order to defend rollback attacks.

Suppose the keychain is secure against rollbacks but there is no trusted location. All data is stored on the vulnerable disk which is accessible to the attacker. Say the attacker wants to perform a rollback attack, he could save the disk image D_1 at time t_1. Later on at time t_2, the attacker could replace the entire disk image by D_1 and the entire keychain could be rolled back to the state at time t_1. Because all data (including SHA-256 or data of any other authentication scheme) the keychain relies on for authentication must be saved on the vulnerable disk, if the keychain is able to detect the rollback attack when reading from the tampered disk image after t_2, it should throw out a similar exception at time t_1 because it is facing exactly the same disk image at t_1 and t_2. This is a contradiction because we had a valid state at t_1 and throwing out an exception at t_1 was impossible. Therefore, there must be a trusted location in order to be secure against rollback attacks.

If the trusted location exists, some data can be stored there. Therefore, the attacker can not completely restore the keychain state at time t_1 using the disk image D_1.

Question 4. In general, MAC might not be collision resistant. Therefore, we might be possible to find two different domains d_1 and d_2 that map to the same tag under the MAC with non-negligible probability. Once d_1 and d_2 are found, the attacker can perform the following attack:

1) specify password p_0 or p_0 to be added under domain d_1
2) specify password p_0 or p_1 ($p_0 \neq p_1$) to be added under domain d_2
3) specify domain d_1 for authentication and receive the password p'
4) compare p' to p_0 and p_1 and output the world bit

The attacker can perform step 3 because step 1 is the last query to domain d_1 and the attacker sent in identical passwords. Because d_1 and d_2 map to the same HMAC, the value of domain d_1 will

be overwritten in step 2, namely, by either the encryption of p_0 or the encryption of p_1. Therefore, the attacker can find out the world bit at step 4 with non-negligible probability.

HMAC is implemented by SHA-256 and it should be collision resistant. However, another secure MAC might be not collision resistant and therefore might result the password manager to be insecure.

Question 5. If we can assume some reasonable upper-bound for the number of records, we can always add dummy records to the KVS so that all KVSs will have the same number of entries. In general, people will have not have a lot passwords. The reasonable upper-bound might be pretty small, potentially in the thousands. And this should not be too memory-intensive.

To construct dummy records, we just choose random strings as domains and random strings as passwords. The possibility that a randomly chosen domain will be mapped to a user chosen one under HMAC is negligible, assuming the user has reasonable number of passwords.