Programming Project 1 - Short Answers

Leo Martel, Christine Li

Friday, Feb 13, 2015

1. The adversary is prevented from learning information about the lengths of the passwords because the passwords are all padded to 64 bytes (maximum password length), salted with $HMAC(domain)$, which is always 256 bits, and finally salted with a random additional salt of constant length (64 bits). AES is a block cipher, so given a fixed-size input will produce a fixed-size output. Therefore the encrypted passwords are all the same length, and no length information is leaked.

2. We use the HMAC'd domain (the keyring key) as part of the password (keyring value) salt. When $get(domain)$ is called, the salted password is decrypted and the stored domain is compared to the parameter. If the domain does not match, we throw a tampering exception. Since a key-value store by definition only contains one password for each domain, swap attacks are impossible.

3. Yes, the trusted location is necessary. If we store the digest on disk, an adversary could modify the database and then modify the digest to match, causing our integrity check to produce a false positive. For example, the adversary could successfully execute a rollback attack by rolling back both the database and the digest to corresponding earlier versions.

4. Using another MAC could jeopardize the security of this system. Specifically, as shown in class CBC-MAC is secure only for fixed-length plaintexts. Domains are not fixed length. They can be arbitrarily long (and the assignment specification doesn't provide a maximum length), so padding is insufficient. Different-length plaintexts produce different-length ciphertexts, leaking information about domain length; in addition, an adversary can win the existential forgery game by submitting plaintexts of varying length.

5. We could pad the password manager with bogus entries such that it is always at its maximum capacity (or at some practical large enough capacity). Before encryption, we could put in an indicator in either the URL or password ($fake||http://$ or $real||http://$) that indicates whether the entry is real or fake. These prefixes would not jeopardize security because we add them before running through HMAC. When adding a new entry, we could iterate through all of the keys until we find a fake one, then delete it. Then, we could add in our new entry to bring the total record count back up to the maximum.