Nicholas Clement

ECEN5032 Homework3: Authentication

1a. A central sign-on facility would increase security by taking normalizing authentication and ensuring that all other university sites do not need to implement their own authentication. It would blanket all university websites and ensure the same security standards across each website, leaving no website any more vulnerable than the next.

1b. This same idea might hurt security by allowing all university websites to be compromised if the central facility is compromised. Similar to the compromising of a Certificate Authority if the central facility is compromised then all tokens it issued cannot be trusted, just like all certificates from a compromised CA cannot be trusted.

1c. If site A is controlled by an attacker the attacker can use the username (u) and Sign(u) to determine the central private key, and thus the Sign(u) for all other sites. This allows the attacker impersonate the user to all other university sites trusting the protocol, without getting authorization from the central facility.

1d. A good way to prevent this security risk would be to ensure that the attacker cannot use sign(u) + the public key to determine the central sign-on private key. One way to accomplish this task is to ensure that all university websites are secure, and sanitized.

1e. Mallory can send bob a random nonce, and then store the nonce. Next she sends the nonce to Alice, on Alice’s reply Mallory is given the HMAC(nonce) using the private key k. Mallory then sends this to Bob and Bob thinks that she is authenticated.

1f. In order to continue using a shared secret we could force the initiator to prove their identity first prior to preforming the HMAC. This would remove the possibility of Mallory using Alice or anyone else in the group to use the private key to allow Mallory fake authentication.

2a. a-z + A-Z + 0-9 = 26+26+10 possible characters for each character of the password. This comes out to 62 possible characters. The total number of possible combinations for this password are 62^8. This comes out to 2.1834011e+14 possible passwords. Given 4 million attempts per second to ensure cracking the password it would take our adversary 2.1834011e+14/4,000,000 seconds. This comes out to 54585027.5 seconds to ensure cracking of the password. This is equal to 909750 seconds or 15162 hours or 606 days to ensure cracking of the password. This is 303 days on average.

2b. Assuming that cracking the password = exhausting all possible combinations of the hash then it would take 7581 bots working on different combinations of the password to ensure completion in one hour on average.

2c. Knowing that our 8-character password = 8 bytes, we can see that we will need to store (2.1834011e+14 \* 8) + (2.1834011e+14 \*32) bytes for a comprehensive table. This is 8.7336044 petabytes of data.

2d. Assuming that k=number of passwords per chain then the space complexity taken by our rainbow table is (N/K) 2\*8 bytes.

2e. Given that total N = 2.1834011e+14 possible passwords and chain length K = 5000 we see that our total size is (2.1834011e+14/5000)\*2\*8 = 698.688352 gigabytes.

2f. Knowing that the attacker will need to preform K(K-1)/2 hash operations for each column we can calculate the number of columns to determine the time necessary to run through the entire table. (5000(4999)/2)/2,000,000 = 6.24875 seconds Per column.

2g. 6.24 seconds is much more feasible than 606 days to determine a password and its hash (guaranteed). In 606 days it is likely that the users password has been reset, or that the user may not be active anymore. It is highly unrealistic to have 15162 bots working in unison, and it is just as unrealistic to have 8.7 petabytes of data dedicated to cracking passwords with modern drives. The rainbow table is a perfect balance of resources to crack passwords in realistic time and space constraints.

2h. Adding 32 bits to the front of the password would increase the complexity of getting all sha-256 hashes of the now longer password. However after determining the hash of a working password the server secret can be known for all passwords, and it is then useless.

2i. If the server had a newly generated secret for each password, it would protect the other passwords in the event that one password is decrypted.

2j. Given the ability to compute 10^9 hashes per second, and 2.1834011e+14 possible passwords, it will take (2.1834011e+14/10^9) seconds or 2.52708461 days to ensure cracking a password.