C2107 Tutorial 5 (PKI and SSL)

School of Computing, NUS

March 8, 2021

1. **Single Signed On (SSO).** Suppose a user is using a new laptop A to visit a website, say Facebook, the website would prompt the user to manually key in the password. After the user has successfully logged in, the website does not prompt the user for the password during subsequent visits. E.g. when the user wants to post a message to Facebook, there is no further prompt for password. This is very convenient as the user only needs to *manually* login once, and hence the term "Single Signed On".

   Although the user doesn't manually login, the laptop automatically carries out authentication with the server. The following is way to achieve SSO:

   S1. After a successful manual login, the web server generates a string t called *authentication token*[1], and sends t to A.

   S2. For each subsequent visit, A automatically sends t to the server without the user involvement. If the authentication token is correct, the website would accept and do not prompt the user for password.

   (a) An authentication token typically has an expiry date.Why?

   (b) A server uses $t = \langle m, y \rangle$ as the token, where

   - $m = d \| r \| u$;
   - $y = H(m)$;
   - d : date and time of the token creation;
   - r : a randomly selected 128-bit string. This would serve as salt;
   - u : the user id; and
   - $H(\cdot)$: a collision resistant hash.

   checking validity of token:

   1) belongs to user

   2) should not be expired

   When server receive t, it verifies the information in m, and verify that $H(m)$ is indeed equal to y.

   Explain why this method is not secure. Give a secure variant.

   (c) Another server uses tuple $t = \langle r, d, u \rangle$ as token. When a user successfully manually logged in, the server generates a random 128-bit r, and d and u is the creation date/time and user id respectively[2]. Next the token t is inserted into a database $\mathcal{D}$. After a user u changed the password, all entries with u in $\mathcal{D}$ will be deleted. Now, to verify a t, the server first searches $\mathcal{D}$. If t is not in $\mathcal{D}$ or expired, it would ask the user to manually login.

[1] In web, the token is sent as "cookie", which will be covered later in the topic of web security.

[2] Remark: In practice, likely there is a device id and the server would make sure that there is only one entry per device in $\mathcal{D}$. In this question, for simplicity, we ignore that.

Mac relies on cryptographic assumptions - like factoring is too difficult

Is this version more secure than your variant in question (b)? In practice, this version is not desired. Why?

token	autofill
-automatic	-
-temporary	-

(d) SSO is different from an alternative method where the browser remembers/stores the password. In this alternative, after a successful login, the browser stores the password. For each subsequent login, the browser automatically fills in the password. Hence, the login page would still appear and the user has to manually click an "ok" button to continue.

Compare the security of SSO and this alternative. Give scenario that SSO is more "secure", and vice vera.

2. **(Role of PKI in TLS)** A school has a local area network that is connected to the Internet via a gateway. All in-coming and out-going traffic must go through the gateway. The school wants to inspect all the communication, and thus installed a monitor M at the gateway.

   (a) Alice is a student in the school. Alice frequently visits websites via https, e.g. the webpage

   `https://www.happytooth.com`

   Furthermore, many of those websites do not support http and thus can only be visited through https.

   (b) Since all traffic via the gateway must be inspected, hence, whenever the monitor M spots https connection, it will drop them. Explain why the gateway is unable to inspect the content of the web traffic?

   (c) The students protest. As a compromise, the school now allows students to visit webpages with https, but with the condition that the monitor is able to decrypt and inspect the web traffic. The students accepted this arrangement. The school approached your company for a solution. Your team derived two possible solutions.

   modifying the browser

   -send session key to M

   to monitor

   (d) Solution S1: All students must install a program (developed by your company) in their machines. Together with the monitor M, the following is carried out whenever a student's browser (let's call it A) wants to make a https connection.

      i. M lets A completes the TLS handshake without interfering.
      ii. A sends the established TLS's session key k to M.
      iii. M uses k to inspect subsequent messages.

   (e) Solution S2:

   Step 1. All students must accept a self-signed certificate signed by an entity with the name `SchoolCA` and public key k_e. This certificate states that `SchoolCA` can issue certificate, that is, it is a CA. The school and the monitor know the private key k_d of k_e.

Step 2. Now, whenever a browser wants to visit a https site, the monitor carries out "proxy-re-encryption" to decrypt, inspect and re-encrypt the traffic. For simplicity, let's call the browser Alice and the website Bob.

Using the following step-by-step guide, explain how Step 2 in S2 is to be carried out.

(a) Alice wants to visit Bob, and the monitor M sits in the middle of Alice and Bob.

(b) First, Alice has to carry out TLS's handshake, which is a unilateral authentication with Bob. At this point, M wants to impersonate Bob and carries out the handshake with Alice.

i. M generates a certificate with the content of (a) name: school CA ____ (b) public key: k_e (c) signature: $Enc_{kd}(\ )$, and sends this certificate to Alice during TLS's handshake.

ii. Alice will accept the the certificate generated by M, because the self-signed cert cert. The authenticated key-exchange will be successfully carried out, because M indeed knows the k_d of the public key k_e.

(c) After the successful authenticated key-exchange, M and Alice establish the session key, which is the pair $(k_\mathsf{a}, t_\mathsf{a})$. Subsequent communication M and Alice will be encrypted using k_a and authenticated using t_a.

(d) M then performs another unilateral authentication with Bob, whereby M uses Bob's public key to verify Bob's authenticity. After the successful authentication, M and Bob establish another session key pair k_b, t_b. Subsequent communication between M and Bob will be encrypted using k_b and authenticated using t_b.

(e) Now, whenever Alice wants to send a message m to Bob, it is to be encrypted using k_a. Since M is the man-in-the-middle, M can intercept the encrypted m. M decrypts it using k_a, inspects it, and then re-encrypts it using k_b, and finally forwards to Bob. Similar steps are carried out for the mac.

(f) Likewise, messages from Bob to Alice are processed in a similar way. Bob's message is to be encrypted using k_b and sent to M. M decrypts it using k_b, inspects it, and then re-encrypts it using k_a, and finally forwards to Alice. Similar steps are carried out for the mac.

3. Which solution, S1 or S2, is preferred? (Consider implementation cost, usability, etc).