C2107 Tutorial 5 (PKI and SSL)

School of Computing, NUS

March 9, 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[1]. 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*[2], 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?

   Solution

   Suppose a token is stolen and the user is not aware of it. Without expiry date, the token can be used forever. With expiry date, the attacker are forced to re-authenticate and they would not able to get the refresh token (since the attacker does not know the password).

   (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

[1] We assume that the client had already conducted unilateral authentication with the server and a secure channel is already established. Here, the client is sure that the server is authentic, whereas the server does not know the authenticity of the client and thus ask for the password. This is a typical setting in web application. E.g. The client first establishes HTTPS with a bank server through unilateral authentication. Next, the bank server asks the client to manually key in password.

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

- $H(\cdot)$: a collision resistant hash.

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.

Solution

i. Assuming Kerckhoff's principle, adversaries know the algorithm and format of the token (in fact, it is very easy for a curious attacker to reverse engineer the format). With the knowledge, the attacker can construct a valid token for any user id.

ii. Replace the unkeyed hash $H(\cdot)$ with a MAC. That is $y = MAC_k(m)$ where k is a secret key known by the server only.

(c) Another server uses a 128-bit string t as token. When a user successfully manually logged in, the server generates a random 128-bit r. Next the token $t = \langle r, d, u \rangle$ is inserted into a database $\mathcal{D}$, where d and u is the creation date/time and user id respectively[3]. 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.

Is this version more secure than your variant in question (b)?

Solution

i. The variant in question (b) relies on the security of the MAC. If the key is stolen, or the MAC has vulnerability (which is unlikely if implemented correctly), then attacker can generate any valid token. This version does not rely on security of any cryptographic primitive.

ii. Here, after a user changed his/her password, all entries with u will be deleted. Hence, after password changed, all previous tokens generated cannot be used. But for the variant in (b), non-expired tokens generated prior to the change (which could be stolen) still can be used.

In practice, this version is not desired. Why?

[3] 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.

Solution

This version requires keeping a database and searching during authentication. Thus more expensive.

(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.

Solution

Let's call browser version B, and the token version T.

i. If the browser is compromised, in B the password would be stolen, whereas in T the token would be stolen. The password could be valid forever, but token has an expiry date.

ii. In B, the user is involved in the login process and thus aware that some sensitive information could be sent or accessed by the process. In T, some background processes could be modified or accessed (e.g. Facebook posts created) sensitive information without the user's knowledge.

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.

(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: ___ (b) public key: ___ (c) signature: ____, and sends this certificate to Alice during TLS's handshake.

   ii. Alice will accept the the certificate generated by M, because ______. The authenticated key-exchange will be successfully carried out, because M indeed knows the _____ of the public key ____.

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

(d) M then performs another unilateral authentication with Bob, whereby _____ uses ______'s public key to verify _____'s authenticity. After the successful authentication, M and Bob establish another session key pair $k_{\mathtt{b}}$, $t_{\mathtt{b}}$. Subsequent communication between M and Bob will be encrypted using _____ and authenticated using _____.

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

Solution

(a) The monitor M sits in the middle of Alice and Bob. Hence M can be a *man-in-the-middle*. (*Note*: In fact, M is a very powerful man-in-the-middle since it knows the `SchoolCA`'s private key k_d.)

(Note: As mentioned in the lecture notes, HTTPS is used to perform a unilateral authentication. The outcome of the employed SSL/TLS handshake protocol is to derive session keys for encrypting and protecting the authenticity of subsequent communication.)

(b) i. M generates a certificate with the content **(i)** `www.happytooth.com` **; (ii)** k_e**; (iii); Validity period; (iv) Signature done using** k_d **of the** `SchoolCA`**'s certificate**.

ii. Alice will accept the the certificate generated by M, because Alice accept M as a root CA. The authenticated key-exchange will be successfully carried out, because M indeed knows the private key 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 $(k_\mathtt{a}, t_\mathtt{a})$. Subsequent communication M and Alice will be encrypted using $k_\mathtt{a}$ and authenticated using $t_\mathtt{a}$.

(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 $k_\mathtt{b}$, $t_\mathtt{b}$. Subsequent communication between M and Bob will be encrypted using $k_\mathtt{b}$ and authenticated using $t_\mathtt{b}$.

(e) Now, whenever Alice wants to send a message m to Bob, it is to be encrypted using $k_\mathtt{a}$. Since M is the man-in-the-middle, M can intercept the encrypted m. M decrypts it using $k_\mathtt{a}$, inspects it, and then re-encrypts it using $k_\mathtt{b}$, 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 $k_\mathtt{b}$ and sent to M. M decrypts it using $k_\mathtt{b}$, inspects it, and then re-encrypts it using $k_\mathtt{a}$, 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).

Solution

S1 would be very difficult to implement since it requires a browser change. This requirement poses a serious problem since the school needs to make the required modifications on various types of popular Web browsers on different OS platforms, and their numerous available versions. Also, browsers do get updated very frequently by their developers, including for security reasons. The solution is therefore not feasible.