CS2107 Tutorial 5
(Security Protocol: Renegotiation Attack on TLS)

School of Computing, NUS

30 September – 4 October 2019

This tutorial looks into attacks on protocol. The discussed attack below illustrates the subtlety and difficulty of designing a secure network protocol. You have to prepare this tutorial by first reading the article "*Understanding the TLS Renegotiation Attack*" on TLS from `http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html`. (You can optionally see a description of the TLS handshake protocol. Yet, we do not need to know the full details on the handshake to complete this tutorial.) The described attack allows an attacker to inject traffic into a legitimate client-server exchange, so that the TLS server will accept it as if it came from the client.

Answer the following questions related to the described attack.

1. What is the role of the Server's public/private key? Note that there is no involvement of the Client's public/private key in the authentication process. Why?

2. How the Attacker and Client get to know the Server's public key?

3. Let k_1 and k_2 be the symmetric keys established during the first and second handshake, respectively. Does the Attacker knows k_2?

4. Which key is used to encrypt the "Initial Traffic"?

5. The second handshake is partially encrypted. Which key is used to encrypt the second handshake?

6. Which key is used to encrypt the "Client Traffic"?

7. The vulnerability was discovered in 2009, and RFC 5746 was released in Feb 2010 to update the TLS/SSL protocol specification. Suppose that, sometime during late 2009, Alice uploaded her report via IVLE (during the "Client Traffic" in the renegotiation attack). Would the confidentiality of the report being compromised under the attack?

8. Bob was tasked to fix the found vulnerability of the TLS protocol. Bob suggested the following:

   In the original TLS's handshake, the very first message was:

   - Client → Server: "Hello, I want to connect".

Bob wanted to change the above to:

- Client → Server: "Hello, I want to connect and this is the $\langle x \rangle$ handshake".

where $\langle x \rangle$ can be `first`, `second`, and so on. Whenever the Server notices an inconsistency, it must then immediately abort. Bob claimed that the above protocol modification would prevent the renegotiation attack. Show that Bob was wrong.

9. Bob realised the mistake. He noticed that in this attack, the Client was the victim. So, he should give the victim the power to control the situation. In his second attempt, he then suggested the following:

   In the original TLS's handshake, the second message was:

   - Server → Client: "Ok, I am happy to connect. Here is my certificate and other information".

   Bob wanted to change the above to:

   - Server → Client: "Ok, I am happy to connect. Here is my certificate and other information. This is our $\langle x \rangle$ handshake".

   where $\langle x \rangle$ can be `first`, `second`, and so on. Whenever the Client notices an inconsistency, it must then immediately aborts. Show that Bob was still wrong.

10. Based on Bob's second attempt, suggest a way to prevent the renegotiation attack.

11. While this instance of TLS performs a unilateral authentication, in many web applications, the Clients still need to be authentication in some ways. In the pizza shop application, how does the client get authenticated?

12. You were the web application developer who built the pizza shop online site. When the renegotiation attack was made known, you knew that your application was vulnerable to the attack. Should you rely on the web server to fix and mitigate the attack (and thus you don't have to do anything), or should you "harden" your web application by modifying it? If you choose the latter, how should you prevent the attack? Recall that in the original application, the following is sent from the Client's browser:

    ```
    GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1
    Cookie: victimscookie
    ```

    What should be sent instead?

13. When the renegotiation attack was made known, a very simple and effective suggested security measure was to disable the renegotiation in TLS. Any implications and limitations to simply immediately disabling the TLS renegotiation feature?

— End of Tutorial —