there are 2 kinds of birthday attacks

S1 = 2^7

S2 = m (2^9)

C2107 Tutorial 6 (Protocol: Renegotiation attack)

School of Computing, NUS

Answer is the same as last tutorial formula for bday attack

March 11, 2021

$2^7 \times 2^{16} = Q$

$2^9 \times 2^{16} = Att$

$1 - e^{-\frac{(2^{9+16})^2}{2^{34}}} = 1?$

1. Consider DNS spoofing. An attacker was able to trick a victim to send out 2^7 DNS queries asking the ip address of a particular domain, say `www.aaa.com`. The attacker was not able to sniff the query, but was able to send spoofed DNS responses to the victim. Since the attacker was unable to sniff, the attacker would not know the QID of each query and thus unable to construct valid responses.

   Nevertheless, this attacker, after tricked the victim to send out the queries, still sent 2^9 spoofed response to victim, each with a randomly generated QID.

   What would be the probability that the attack succeeded? (that is, at least of the responses matched one of the queries).

   How is this question related to the size of nonce in strong authentication?

2. **Renegotiation attack.**
   This attack illustrates the subtly and difficulty in designing a security protocol. You have to prepare for this tutorial. Read Renegotiation attack on TLS :

   `http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html`

   renegotiation just means another TLS handshake under the 1st encrypted session key, to discuss the 2nd session key

   This attack exploits a vulnerability in how TLS handle the handshake, and it provides a way for the attacker to ~~append~~ any message. (see lecture note for TLS's handshake)

   (a) Is this a unilateral or mutual authentication? If unilateral, which entity, Server or the Client, carries out the verification?

   (b) How the Client and Attacker get to know the server's public key?

   (c) Let (k_1, t_1) and (k_2, t_2) be the sessions keys established during the first and second handshake respectively. Does the Attacker knows k_2?

   (d) Which key is used to encrypt the "Initial Traffic"?

   (e) The second handshake is partially encrypted. Which key is used to encrypt the second handshake?

   (f) Which key is used to encrypt the "Client Traffic"?

after sending the client key exchange, encrypted using public key, the server will check the key, then both of them will send the same message "Finished" to each other to ensure that they have confirmed to use same key

(g) The vulnerability was discovered in 2009, and RFC 5746 was released in Feb 2010 to update TLS/SSL protocol specification. Suppose that, sometime during late 2009, Alice uploaded her report via Luminus. Later, Alice was very worried that someone would peep into her report. Could the confidentiality of the report be compromised?

(h) Bob was tasked to fix the vulnerability in the TLS handshake 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 x-th handshake".

where x can be `1`, `2`, and so on. Whenever the server notices inconsistency, it must immediately abort. Bob claimed that the above protocol modification would prevent the renegotiation attack. Show that Bob was wrong.

(i) 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 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 x-th handshake".

Whenever the client notices inconsistency, it must immediately aborts. Show that Bob was still wrong.

(j) Based on Bob's second attempt, suggest a way to prevent renegotiation attack.

(k) While this instance of TLS performs unilateral authentication, in many web applications, the clients still need to be authentication in some ways. In the pizza shop application, how is the client authenticated?

(l) 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 "harden" your web application by modifying your application? 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.

changing format of the M

idea is to ensure that the request message will have the secret instead of the cookie being sent at the end

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

What should be sent instead?

(m) When the Renegotiation attack was made known, a very simple and effective security was to disable renegotiation in TLS. Any implications and limitations to simply disabling TLS renegotiation?

Importance of modularity: So that there is a separation between layers, means application developer does not need to worry about a fix in the TLS vulnerability