# How does server use session key after tls handshake?

Asked 2 years, 11 months ago   Active 2 years, 11 months ago   Viewed 1k times

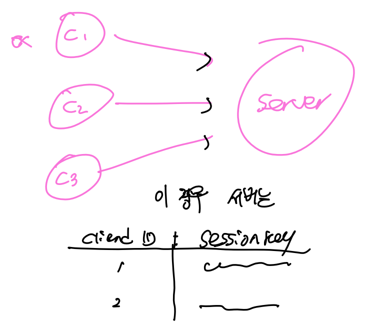
Q tls handshake 이후 Pms에 의해 생성된 대칭키 (session key)는 어디에 저장되나요?

3

After TLS handshake both server and client agreed about `session key` to use as symmetric key. So server must store several `session keys` which each one belongs to a specific client.

How server determines for a received request which `session key` must be used(i.e `session key` of which client)? (Is this true: Client appends its `session id` in each request(somewhere like header?) and server has a map between `session id` and `session key`.)

How long a server uses a `session key`? Is it bad practice to use a session key for long time? Why they change `session key` while it transferred in a secure(i.e encrypted with public key) manner?



이 경우 서버는

어떤식 저장?

4

Normally, once the TLS handshake is complete, the client and server can exchange encrypted traffic on the same connection without having to communicate the session id. This is because the server associates the SSL context information (including the session key) with the TCP socket on its end. You can see this in this simplified example of a TCP server: https://wiki.openssl.org/index.php/Simple_TLS_Server

Ans)
TLS handshake 이후 서버는 SSL context information 과 TCP 소켓을 연결 시키기 때문에, 클라이언트는 tcp 연결을 계속 하고 있다. (keep-alive)

그러므로 session key는 저장할 필요가 없다.

However, there are two scenarios where the session id is communicated after the initial handshake - session resumption and session renegotiation. 뭐지!

Session resumption allows the same TLS session (containing all the parameters agreed in the handshake) on a *new* TCP connection after closing the original one. This means the connection could time out, and your browser would detect this case and resume the TLS session on a new connection by sending the session id in a new Client Hello. More details on session resumption here: http://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077.html

Session renegotiation is used to change the parameters to be different from the initial handshake, but on the *same* TCP connection.

As you mention, the storage of a session cache on the server compromises security. An attacker could gain access to the session cache and then decrypt traffic. More details on this here - https://blog.compass-security.com/2017/06/about-tls-perfect-forward-secrecy-and-session-resumption/

Share  Improve this answer  Follow

answered Sep 29 '18 at 21:28
Ben Rowland
156 1

Thanks for your detailed answer. So for each request(post, get,put,delete,...) it must do a handshake? I previously assumed since handshake is a cpu intensive operation it should do a handshake and use its result for multiple requests... – Bonje Fir Sep 30 '18 at 5:30

1 @BonjeFir: you are mixing several things together. TLS is a protection for the TCP connection. get, put ... are actions at the application protocol level. Depending on the application protocol it might be possible to have several actions within a single TCP connection. Specifically with HTTP/1 (when using keep-alive) or with HTTP/2 a TCP connection will be created, upgraded to TLS and then several HTTP requests and responses will be done using the same TCP connection (and the same TLS session) - i.e. one TLS handshake and multiple HTTP requests. – Steffen Ullrich Sep 30 '18 at 6:02

HTTP 1. / HTTP 2
$\downarrow$
connection이 유지된다면, 여러 http 요청에 대하여 같은 TCP connection을 사용한다.

@SteffenUllrich Thanks, very helpful clarification. – Bonje Fir Sep 30 '18 at 6:28

Add a comment