Session Resume의 기한 연장을 이용한 SSL/TLS Handshake 프로토콜의 성능 개선

박지철° 한명진 이경현
부경대학교 전자계산학과
{webhunt°, sparkeye}@unicorn.pknu.ac.kr, khrhee@pknu.ac.kr

Performance Improvement of SSL/TLS Handshake Protocol through extension of a Session Resume

Ji-Cheol Park°, Myong-Jin Han, Kyung-Hyune Rhee
Dept. of Computer Science, PuKyong Nat'l University

요 약

전송계층에서의 안전한 통신을 위한 Secure Sockets Layer(SSL)와 Transport Layer Security(TLS)의 Handshake 프로토콜에서 Session ID의 저장이 매우 짧은 시간 동안 저장됨으로 전체적인 Full Handshake의 횟수가 증가한다. 따라서, 안전한 Session resume 보장함으로 서버의 session cache 기한을 연장할 수 있으며 전체적인 Full Handshake 프로토콜의 횟수를 줄일 수 있다. 본 논문에서는 Handshake 프로토콜의 성능 개선을 위하여 S/Key와 같은 해쉬의 일방향 성질을 이용하는 개선된 Session resume의 방안을 제안한다.

1. 서 론

컴퓨터 네트워크 기술의 발달과 인터넷의 활용 영역이 다양한 분야로 확대되면서 안전한 데이터의 전송이 많은 어플리케이션의 주요 요구사항으로 대두되었으며, 인터넷과 같은 공중망을 사용함에 있어서 사용자의 인증, 데이터의 기밀성과 무결성 그리고 부인봉쇄와 같은 보안 서비스를 제공하기 위해 대부분의 어플리케이션에서 SSL/TLS[1][2] 보안 프로토콜을 채택하고 있다.

SSL/TLS는 클라이언트와 서버 사이에 교환되는 데이터를 안전하게 보호하기 위하여 공개키 인증서를 통해 사용자 인증, 비밀키 암호 시스템(예, DES, 3DES, RC4)을 통한 데이터 기밀성을 제공한다. 이때, 비밀키 암호화에서 사용되는 이 비밀키는 키 교환 알고리즘으로 교환된 비밀 값(master_secret)으로 유도된다. 그리고 데이터의 위/변조를 막기 위하여 메시지 인증 코드(MAC : Message Authentication Code)를 사용한다.

본 논문에서는 SSL/TLS의 handshake 과정에서 발생할 수 있는 Session Resume을 분석하고, 이를 보완하여 성능이 향상된 해쉬의 일방향 성질을 이용한 n회의 안전한 Session resume Handshake 프로토콜을 제안한다. 본 논문의 구성은 2장에서 관련 연구를 논의하고, 3장에서는 SSL/TLS Session Resume의 성능 개선 방안을 제안한다. 마지막으로 4장에서 결론 및 향후과제에 대해 논의하고자 한다.

2. 관련연구

2.1. SSL/TLS Handshake 프로토콜의 개요

SSL/TLS는 TCP/IP 계층과 응용계층 사이에 위치하는 프로토콜로서, 다양한 응용 프로그램들에게 보안서비스를 제공해 준다. SSL/TLS는 레코드 계층 위에 Change Cipher Spec, Alert, Handshake, Application Data 프로토콜로 구성된 계층 구조를 가진다. 클라이언트는 Handshake 프로토콜로 안전한 통신을 서버에게 요청하게 되고 서버는 클라이언트의 요청에 응답을 하여 통신에 필요한 파라미터들을 설정한다. Handshake 프로토콜에 의해서 설정된 파라미터들은 Change Cipher Spec 프로토콜에 의해 사용 할 수 있도록 활성화되고 데이터들은 레코드 계층에 의해 보호되어 전송된다.

통신과정에서 발생한 오류들은 Alert 프로토콜로 처리된다. 각 프로토콜의 모든 데이터는 레코드 계층을 통하여 전송된다. 레코드 계층에서 사용되는 파라미터들은 Handshake 프로토콜에서 다음 그림1과 같이 설정된다.

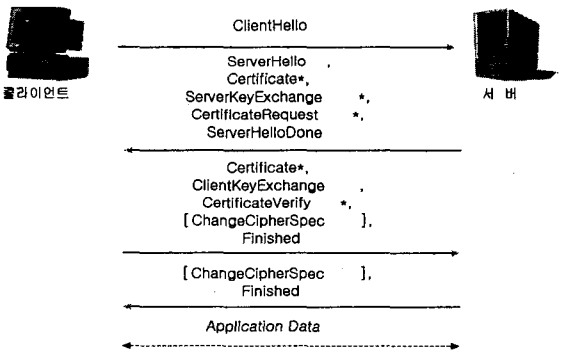

그림 1. Full Handshake 프로토콜

클라이언트가 새로운 Session을 생성할 경우 비어있는 Session ID를 서버에 전송하게 되고, 이미 존재하는 Session을 사용하여 새로운 연결(connection)을 생성할 경우에는 클라이언트는 ClientHello에서 재사용하기를 원하는 Session ID를 전송하게 된다. 서버는 클라이언트가 보낸 Session ID를 자신의 Session Cache에서 일치하는 것이 있는지 확인하여 일치하는 Session ID를 찾으면 그림2와 같이 간단하게 Abbreviate Handshake 프로토콜(Session Resume)이 진행된다. 일치하는 Session ID를 찾을 수 없으면 서버는 새로운 Session ID를 보내거나 비어있는 Session ID와 함께 Alert를 보내게 되어 Full Handshake를 수행한다.

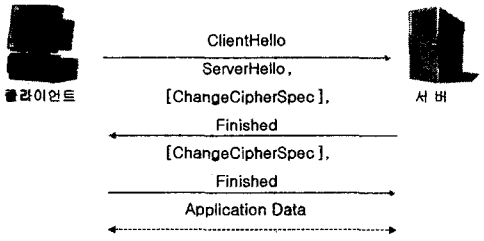

그림 2. Abbreviate Handshake 프로토콜

새로운 Session Resume이 수행될 때는 새로운 서버와 클라이언트의 랜덤 값, cache에 저장되어있던 기존의 암호화키(Master_secret)를 재 사용하여 새로운 키 블록을 생성한다.

2.2. S/KEY 일회용 패스워드 시스템의 개요

S/KEY 일회용 패스워드 시스템[3][4]은 네트워크상의 도청/재생 공격으로부터 안전한 인증을 제공한다. 이 시스템은 기존의 one-time 혹은 multi-use 인증 시스템들에 비해 여러 가지 장점을 가지고 있다.

2.2.1. 특징

S/KEY 일회용 패스워드 시스템은 다음과 같은 특징들을 가지고 있다.

-기본 OPT(One Time Password)의 특징을 가지고 있다.
-사용하기에 간단하다.
-비밀 패스워드를 기억하도록 한다.
-자동화가 되어질 수 있다.
-알고리즘이 공개되어 있다.(MD4 또는 MD5 일방향 해쉬 함수)
-어떤 비밀 정보도 호스트에 보관되어지지 않는다.

2.2.2. 알고리즘

기본적인 알고리즘은 일방향 해쉬 함수를 여러번 적용함으로 계속해서 생성되어진다. f(x) = y 에서 x를 알면 y를 쉽게 구할 수 있으나 y를 알고 x를 알아내기란, 즉, $x = f^{-1}(y)$는 거의 불가능하다.

첫 번째 OTP는 사용자의 비밀 패스워드(S)를 정해진 특정 수 (n)만큼의 일방향 해쉬 함수를 수행함으로 생성되어진다.

$$P(1) = H_n(H_{n-1}(....H_2(H_1(S))))$$

다음 번에는 n-1번을 수행함으로 생성되어지는 P를 사용할 것이다.

$$P(2) = H_{n-1}(....H_2(H_1(S)))$$

그렇기 때문에 도청자가 P(i)를 알게 되더라도, 다음 패스워드, P(i+1)을 알아낸다는 것이 불가능하게 된다. 처음에 서버는 OTP의 복사본을 저장한다. 그리고 그것을 일방향 해쉬 함수로 계산하여, 복사본과 비교한다. 만약 일치하게 된다면, 시스템 패스워드 파일의 사용자의 엔트리는 OTP의 복사본으로 갱신되어진다.

사용자에 의해 일방향 해쉬 함수의 수가 하나씩 줄어들게 되므로 어느 시점에 다다르면 초기화를 시켜주어야 한다. 사용자의 OTP는 노트북이나 palm-top등을 포함하는 다양한 기종의 pc에서 수행될 수 있다. 플로피 디스크에 저장되어 수행되거나, 미리 계산을 하여 프린트를 한 후 사용할 수 있다.

3. Session Resume의 성능 개선 방안

본 장에서는 2장에서 소개된 SSL/TLS Handshake 프로토콜 Session Resume의 성능 개선 방안을 제시한다. Session은 이전 Session ID가 서버의 Session cache에 존재할 때만 resume을 할 수 있다. 그러나 비교적 부하가 높은 서버의 실제 구현에서 Session ID의 저장은 매우 짧은 시간 동안만 저장됨으로, Full Handshake가 빈번히 수행된다. 본 논문에서는 S/Key에서와 같은 해쉬의 일방향성을 이용한 n회의 안전한 Session Resume을 보장함으로서 Session cache 기한을 기존보다 오랜 시간 동안 지속될 수 있게된다. 따라서, 기존 SSL/TLS 보다 전체적인 Full Handshake의 횟수를 줄일 수 있게됨으로서 서버와 클라이언트 모두 계산량과 전송량을 줄일 수 있다.

3.1. 개선 방안

기존 SSL/TLS Handshake 프로토콜은 서버와 클라이언트 사이의 Full Handshake 이후에 Session을 짧은 시간 동안만 저장한다. 이와 같은 짧은 Session cache의 기한으로 인해 많은 Full Handshake를 요구하게 된다. 이러한 한계를 보완하기 위해서 본 논문에서는 기존 SSL/TLS 보다 오랜 시간동안 안전하게 Session cache를 할 수 있는 "해쉬의 일방향성을 이용한 n회의 안전한 Session Resume"을 제안한다.

기존의 Handshake 프로토콜은 Session ID가 값을 가질 경우에 이를 Session ID cache값과 비교하여 값이 맞을 경우에는 Session Resume을 실시한다. 그러나 본 논문에서는 Session ID의 의미를 변경하였다.

기존의 Session ID를 재사용하지 않고 다음과 같이 S/Key를 이용하여 서버와 클라이언트 모두 새로운 Session ID(n,m)를 생성한다.

[표기]
(n, m) : Full Handshake 도중에 결정된 일방향 해쉬 함수 수

행 횟수(m 단계별로 n회를 수행)
$H_{n,m}$(secret) : 비밀값 secret를 특정 m 단계에서 n 만큼 일방향 해쉬 함수 수행.
Session ID(0,1) : 최초 Full Handshake 이후의 세션 ID.
Session ID(i,m) : m 단계의 i 번째 Session ID.
(n, m, i >= 1)

Session ID(1,m)= $H_{(n,m)}$(master_secret||Session ID(0,m))
...
Session ID(i,m)= $H_{(n-(i-1),m)}$(master_secret||Session ID(0,m))
Session ID(i+ 1,m)= $H_{(n-i,m)}$(master_secret||Session ID(0,m))
...
Session ID(i,m)= $H_{(n-i,m)}$(master_secret||Session ID(0,m))(i=n-1)
Session ID(1,m+ 1)= $H_{(n,m+1)}$(master_secret||Session ID(n-1,m))
...

즉, (n,m)이 (100,10)이라고 가정하면, Full Handshake에서 Session ID(0,1), 비밀값 master_secret, 일방향 해쉬 함수 수행 횟수 n,m이 결정된다. Session Resume의 첫 번째 Session ID(1,1)는 위 식을 이용하여 계산되어진다.

Session ID(1,1)= $H_{100,1}$(master_secret||Session ID(0,1))
Session ID(2,1) = $H_{99,1}$(master_secret || Session ID(0,1))
...
Session ID(99,1) = $H_{1,1}$(master_secret || Session ID(0,1))
Session ID(1,2) = $H_{100,2}$(master_secret || Session ID(99,1))
...

다음의 Session ID(2,1)는 일방향 해쉬 함수 99번 수행함으로 생성된다.

Session ID(i,m)의 사용을 모니터하고 있는 도청자는 다음 Session ID(i+ 1,m)를 생성해 낼 수 없을 것이다. 처음 시점에서의 master_secret를 알지 못하면 도청이 불가능하게 된다.

Session Resume에서 클라이언트는 ClientHello에 Session ID(i,m)를 계산하여 전송하고 이를 받은 서버는 자신이 계산한 Session ID(i,m)'와 일치하지 않으면 Session Resume을 실패하여 Full Handshake를 실시하게 된다. 만약 서로 일치하면, 서버의 Session Cache에 Session ID(i,m)'로 갱신한다. 따라서, Session ID는 기존의 서버와 클라이언트 사이의 단순한 Session 확인의 의미였으나 본 제안에서는 인증의 의미를 포함하게 된다.

본 논문에서는 해쉬의 일방향성을 이용한 n회 안전한 Session Resume 프로토콜을 강화하여 보다 많은 Session Resume 기한(즉, 서버의 Session ID cache 기한)을 연장함으로서 기존보다 적은 Full Handshake를 실시하게 된다. 그러므로, 개선방안에서는 기존 SSL/TLS보다 적은 계산량과 전송량이 요구된다

계산 성능이 낮은 장치를 사용하는 무선 환경에서 보다 효율적으로 적용 가능 할 것으로 기대 된다.

3.2. 성능

기존 SSL/TLS의 Session Resume 연산에 간단한 해쉬 연산의 추가로서 많은 계산량을 요구하는 공개키 암호 알고리즘 연산을 줄일 수 있으며, Full Handshake를 실행하지 않으므로 전송량 또한 줄일 수 있다.

클라이언트에 의해 수행된 단 방향 함수의 수(n)와 각 단계(m)가 하나씩 줄어들기 때문에, 어느 시점에 다다르면 클라이언트는 다시 서버와의 Full Handshake를 통해 재 초기화를 해야만 한다.

최초의 Session Resume부터 재 초기화되기까지의 횟수를 x라고 하면, x번의 Session Resume을 보장받을 수 있을 것이다. 이 x가 기존 cache 기간내의 resume의 수 보다 y배의 차가 난다면, y-1 만큼의 Full Handshake의 수를 줄일 수 있을 것이다. 이렇게 x의 크기를 조절함으로서 기존 보다 Full Handshake의 수를 줄일 수 있게 된다. 실제 구현에서는 이런 x의 크기를 설정하는 것이 중요하다. 설정 방법으로는 서버의 관리자가 자신 서버 환경에 맞는 적절한 값 x를 지정하여 매번 이 값을 사용하도록 한다.

4. 결론 및 향후과제

기존 SSL/TLS Handshake 프로토콜은 서버와 클라이언트 사이의 Full Handshake 이후에 짧은 Session cache의 기한 한계를 보완하기 위해서 본 논문에서는 기존 SSL/TLS 보다 오랜 시간동안 Session cache을 할 수 있는 해쉬의 일방향성을 이용한 n회의 안전한 Session Resume을 제안하였다.

본 논문에서는 간단 한 해쉬 연산의 추가로 인한 보다 많은 Session ID cache 기한을 연장 할 수 있다. 또한 전체적인 Full Handshake 프로토콜의 횟수를 줄일 수 있음을 보았다. 따라서 공개키 알고리즘 계산과 같은 많은 계산을 요구하는 연산을 하지 않게 됨으로 서버와 클라이언트는 계산적으로 성능이 향상되며 전송량 또한 줄일 수 있다.

향후 일정시간 내에 Session Resume의 통계적 횟수와 master_secret의 보안강도를 고려하여 적당한 n의 값을 도출하여야 할 것이다.

[참고문헌]

[1] A. Freier, P.Karlton and P. Kocher, "The SSL Protocol version 3.0", Nov. 1996, InternetDraft, available at http://home.netscape.com/eng/ssl3/draft302.txt.

[2] T. Dierks and C. Allen, "The TLS Protocol version 1.0", Jan. 1999, RFC 2246, available at http://www.cis.ohio-state.edu/rfc/rfc2246.txt.

[3] Haller, N., "The S/KEY One-Time Password System", February 1995, RFC 1760.

[4] N. Haller, C. Metz, P. Nesser and M. Straw, "A One-Time Password System", Feb. 1998, RFC 2246, available at http://www.ietf.org/rfc/rfc2289.txt