PART III Network Security Application

CHAPTER 17 Web Security

17.1 Web Security Considerations

17.2 Secure Socket Layer (SSL) and Transport Layer Security (TLS)

17.1 Web Security Considerations

Web Security Threats

- Integrity : Cryptographic Checksum
  - Modification of user data, memory, message traffic in transit
  - Trojan horse browser
- Confidentiality : Encryption Web proxies
  - Eavesdropping on the net
  - Theft of info from server or client
  - Info about network configuration
  - Info about which client talks to server
- Denial of Service : Difficult to prevent
  - Killing of user threads
  - Flooding machine with bogus requests
  - Filling up disk or memory
- Authentication : Cryptographic Techniques
  - Impersonation of Legitimate user
  - Data forgery

17.1 Web Security Considerations

Web Traffic Security Approaches

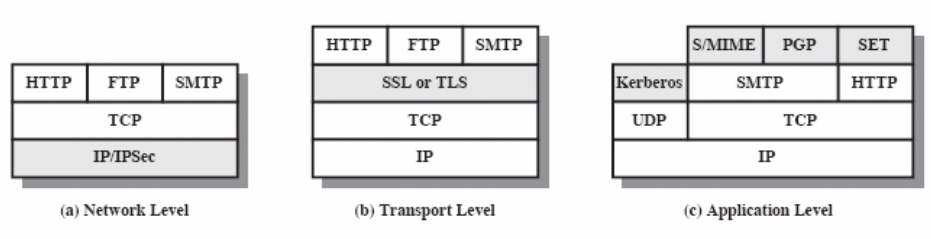


Fig. 17.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

17.2 SSL & TLS

- SSL 개요
- SSL Handshake Protocol
- SSL Alert Protocol
- SSL Chage Cipher Spec protocol
- SSL Record protocol
- TLS 소개
- WTLS 소개

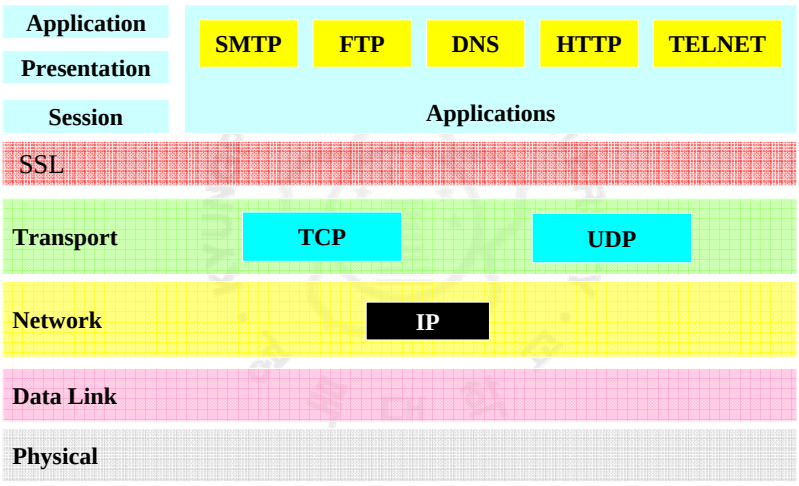
17.2 SSL & TLS

SSL 개요

- 네스케이프(Netscape)사에서 처음으로 제안, 웹 보안의 대명사로 알려져 있는 보안 프로토콜.
- 웹과 같은 특정 응용을 위한 보안 프로토콜이 아닌 일반적인 인터넷 보안 프로토콜로 사용
- TCP/IP계층과 어플리케이션 계층(HTTP, TELNET, FTP)사이에 위치하여 종단간 보안 서비스를 제공

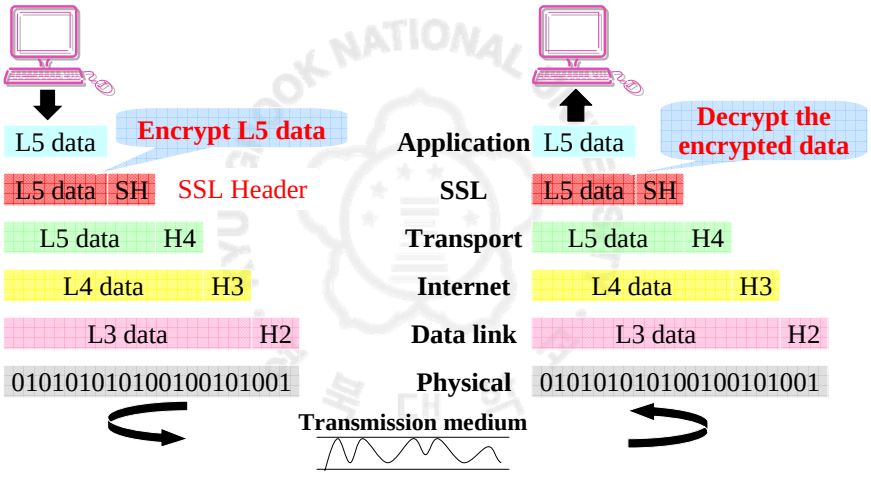
17.2 SSL & TLS

TCP/IP Layers



17.2 SSL & TLS

The Position of SSL in TCP/IP Protocol Suite



17.2 SSL & TLS

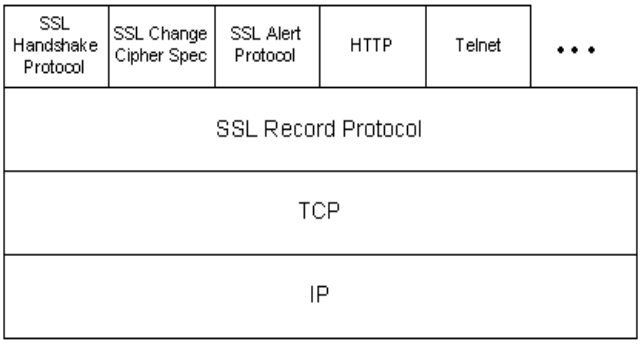
SSL 개요

- 두 응용간의 기밀성(Confidentiality) 서비스
  - DES, RC4 등과 같은 대칭키 암호화 알고리즘;
  - 사용되는 비밀키는 Handshake Protocol을 통해 생성
- Client와 Sever의 상호인증(Mutual authentication)
  - RSA와 같은 비 대칭키 암호화 알고리즘,
  - DSS와 같은 전자서명 방식과 X.509 인증서가 사용
- 메시지 무결성(Message integrity) 서비스
  - MAC(Message Authentication Code)기법이 사용
- SSLv1 designed by Netscape; SSLv2 shipped with Navigator 1.0; SSLv3 was peer-reviewed;
- RFC 2246 : TLS protocol version 1.0

17.2 SSL & TLS

SSL 개요

- SSL의 2계층 구조
  - Handshake Protocol ,Change Cipher Spec Protocol, Alert Protocol, Record Protocol



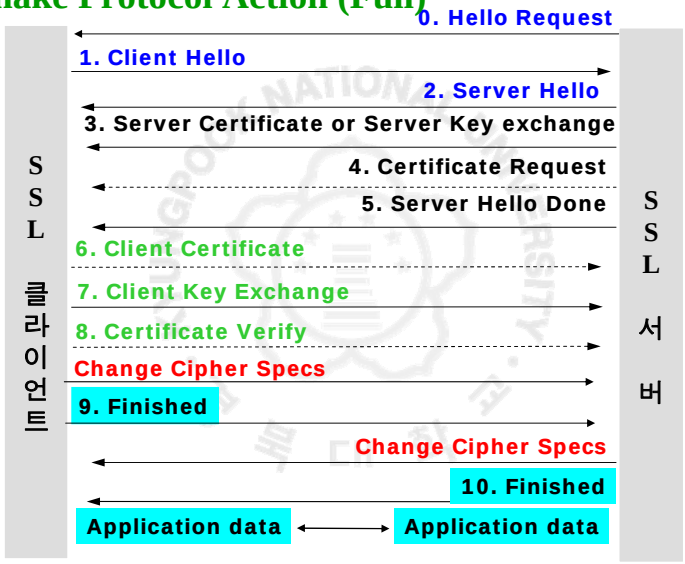
17.2 SSL & TLS

SSL Handshake Protocol

- Client와 Server 양쪽사이의 암호화적 parameters 생성
  - Session identifier: Server에 의해 생성된 랜덤 시퀀스
  - Peer certificate: X.509 인증서
  - Compression method: 압축에 이용되는 알고리즘
  - Cipher spec : 암호화 알고리즘
  - Master secret : Server와 Client 간의 비밀스럽게 공유된 48 byte 시퀀스
- SSL Record Protocol의 앞에서 수행
- Server와 Client간의 합의내용
  - Protocol Version (SSL)
  - Cryptographic algorithms
  - 상호인증 [Option]
  - 공개키 기술을 이용한 pre_master_secret를 공유

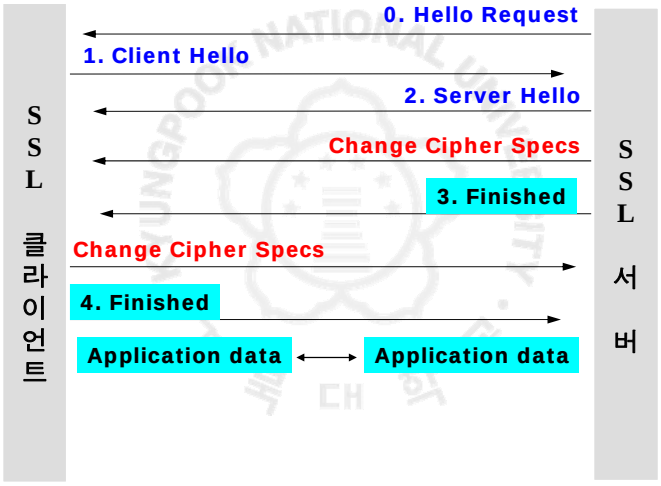
17.2 SSL & TLS

Handshake Protocol Action (Full)



17.2 SSL & TLS

Handshake Protocol Action (Resume)



17.2 SSL & TLS

Handshake Protocol Message

0) Hello Request

이 메시지는 server가 client에게 보낼 수 있는 메시지이지만 handshake 프로토콜이 이미 진행 중이면 client는 이 메시지를 무시.

1) Client Hello

Client는 server에 처음으로 연결을 시도할 때 Client Hello 메시지를 통해 client SSL 버전, client에서 생성한 임의의 난수, 세션 식별자, Cipher Suit 리스트, client가 지원하는 압축방법 리스트 등의 정보를 서버에 전송.

17.2 SSL & TLS

1) Client Hello

- **Client_version** : Client가 사용하는 SSL의 버전
- **Random** (ClientHello.Random)
  - Gmt_unix_time : 서버의 응답시간
  - Random_byte : 28 byte (random number generator)
- **Session id**
  - Client가 이 세션에서 원하는 세션 식별자
  - 기존에 서버와 연결된 적이 있다면 같은 Session id와 그에 해당하는 암호학적 파라메터 이용
  - 만약 이 필드의 값이 없다면 Server에서 다시 설정
- **Cipher_suites**
  - Client측에서 사용하는 암호 알고리즘의 리스트 및 size

17.2 SSL & TLS

- **Compression_methods**
  - Client측에서 사용하는 압축방법의 리스트
- **Cipher Suites** :

SSL_키교환알고리즘_WITH_암호알고리즘_해쉬알고리즘

- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
- SSL_RSA_WITH_IDEA_CBC_SHA

.
.
.

17.2 SSL & TLS

2) Server Hello

Server는 Client Hello 메시지를 수신 후, handshake_failure Alert 메시지 or Server Hello 메시지를 전송.

Server는 server의 SSL 버전, server에서 생성한 난수, 세션 식별자, client가 보낸 Cipher Suit list 중에서 선택한 하나의 Cipher Suit, client의 압축방법 list에서 선택한 압축 방법 정보를 client에 전송

17.2 SSL & TLS

2) Server Hello

- **Server_version** : Server에서 이용되는 SSL 버전
- **Random** (ServerHello.Ramdom) : Client와 별도로 생성
- **Session_id**
  - ClientHello.Session_id가 있다면, 맞는지 검사 후 같다면 같은 값 리턴
  - ClientHello.Session_id가 비었다면, 해당하는 Session_id를 찾음
- **Cipher_suite** : ClientHello.cipher_suite로부터 하나를 선택
- **Compression_method** : ClientHello.cipher_method로부터 하나를 선택

17.2 SSL & TLS

3) Server Certificate or Server Key Exchange

Sever 인증을 위해서 자신의 공개키 인증서를 가지고 있다면, Server Certificate 메시지를 즉시 client에 전송. 일반적으로 X.509 버전 3 인증서를 사용

- **Server certificate** [optional]
  - Server가 자신의 인증서를 Client에게 전송
  - 인증서의 정보는 선택된 cipher suite의 키 교환 알고리즘에 이용
- SSL 2.0 : RSA만 이용, SSL 3.0 : RSA, D-H 이용
- **Server key exchange message** [optional]

17.2 SSL & TLS

- 인증서가 없을 경우 사용
  - 만약 cipher suite에서 RSA를 선택했다면
    - Rsa_modulus
    - Rsa_exponent
  - 만약 cipher suite에서 D-H를 선택했다면
    - DH_p, DH_g, DH_Y
  - 만약 cipher suite에서 Fortezza를 선택했다면
    - R_s

17.2 SSL & TLS

4) Certificate Request

Server는 기본적으로 client에게 서버 자신을 인증을 할 수 있도록 함.

이와 마찬가지로 server는 client의 인증서를 요구하여 신뢰할 수 있는 client인지 확인할 수 있음.

이 단계는 선택적으로 동작하지만 client와 server는 상호인증이 가능.

- Certificate request [optional]

17.2 SSL & TLS

5) Server Hello Done

Server에서 보낼 메시지를 모두 보냈음을 의미.

이 메시지를 받은 client는 server로부터 더 이상의 메시지 전송을 없음을 알 수 있게 된다.

Client의 응답을 기다림

Client는 Server hello done 메시지가 도착할 때까지 기다림

17.2 SSL & TLS

6) Client Certificate

Server로부터 client의 인증서를 보내라는 요청이 있을 경우 client 자신의 인증서를 보내게 됨:[optional].

7) Client Key Exchange

Client는 seesion key를 생성하는데 이용되는 임의의 비밀정보인 48바이트 pre_master_secret를 생성.

다음에 선택된 공개키 알고리즘(RSA, Fortezza, Diffie-Hellma 중 하나)에 따라 pre_master_secret 정보를 암호화 하여 server에 전송.

Pre_master_secret는 master_secret 를 생성하는데 이용

17.2 SSL & TLS

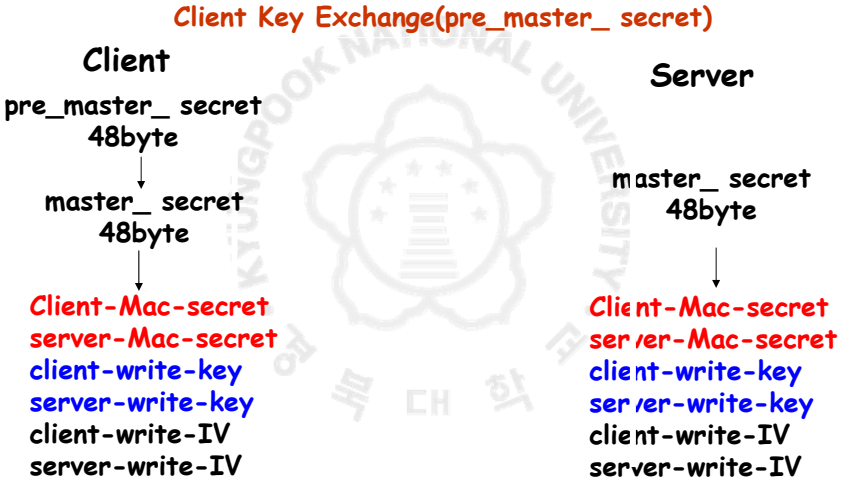
8) Certificate Verify

Server의 요구에 의해 전송하는 client의 인증서를 server가 쉽게 확인할 수 있도록 client는 메시지를 전자 서명하여 전송.

이 메시지를 통해 server는 client의 인증서에 포함된 공개키가 유효한지 확인

17.2 SSL & TLS

키 생성 및 교환 절차



17.2 SSL & TLS

Master_secret 생성 방법

- Master_secret의 생성 : 48byte

MD5(pre_master_secret + SHA('A' + pre_master_secret +
ClientHello.random + ServerHello.random)) +

MD5(pre_master_secret + SHA('BB' + pre_master_secret +
ClientHello.random + ServerHello.random)) +

MD5(pre_master_secret + SHA('CCC' + pre_master_secret +
ClientHello.random + ServerHello.random))

17.2 SSL & TLS

key_block의 생성 : 필요한 길이만큼 생성

MD5(master_secret + SHA('A' + master_secret +
ClientHello.random + ServerHello.random)) +

MD5(master_secret + SHA('BB' + master_secret +
ClientHello.random + ServerHello.random)) +

MD5(master_secret + SHA('CCC' + master_secret +
ClientHello.random + ServerHello.random))

MD5(master_secret + SHA('….' + master_secret +
ClientHello.random + ServerHello.random))

key_block

17.2 SSL & TLS

Key 생성

- Keyblock을 원하는 길이만큼 분할해서 키를 만듬

client-MAC-Secret [cipher_spec.hash_size] = **Key_block**[0-15]
server-MAC-secret [cipher_spec.hash_size] = **Key_block**[16-31]

client-write-key [cipher_spec.key_material] = **Key_block**[32-36]
server-write-key [cipher_spec.key_material] = **Key_block**[37-41]

client-write-IV [cipher_spec.IV_size] = **size 8 byte**
server-write-IV [cipher_spec.IV_size] = **size 8 byte**

17.2 SSL & TLS

SSL에서의 대칭키 알고리즘

- 암호화와 SSL record의 무결성을 이용하는데 이용
- DES와 MD5를 주로 이용
- Master_secret를 이용해 MAC keys, DES keys, Ivs를 생성
- Key의 종류
  - Client write MAC secret, Server write MAC secret
  - Client write key, Server write key
  - Client write IV, Server write IV

17.2 SSL & TLS

9) Change Cipher Specs, Finished (from Client)

10) Chang cipher Specs, Finished (from server)

Change Cipher Specs 메시지는 handshake protocol의 메시지는 아님.

Client (server)는change Cipher Specs 메시지를 server (client)에 전송.

이후에 전송되는 모든 메시지는 협상된 알고리즘과 키를 이용하여야 함을 알리게 됨.

Finished 메시지(협상된 알고리즘 및 키로 암호)를 생성하여 server (client)에 전송.
Handshake의 종료

17.2 SSL & TLS

Change Cipher Spec

- SSL Handshake protocol과 독립적
- 선택된 암호 알고리즘과 MAC 알고리즘을 정의
- 키 교환과 인증과정이 성공적이었다고 증명
- 이 메시지 이후에 전송되는 메시지는 새롭게 형성된 알고리즘과 키를 이용할 것임을 지시하게 됨
- Finished 메시지는 Change Cipher Specs 메시지 이후에 전송되며, 협상된 알고리즘과 키가 처음으로 적용됨

17.2 SSL & TLS

Alert protocol

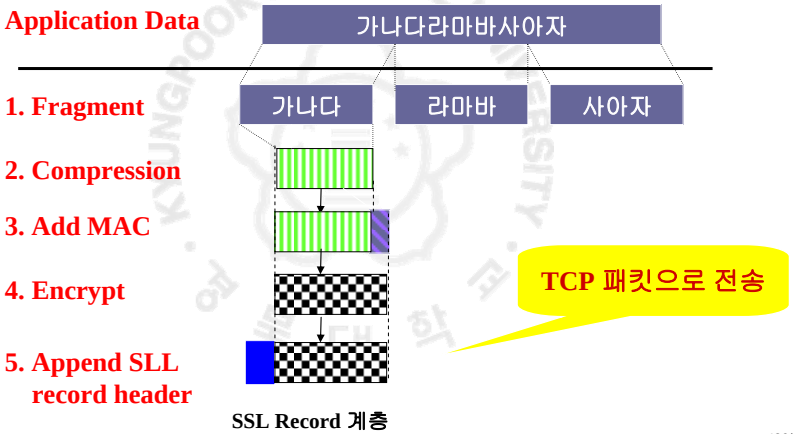
- Alert protocol은 압축 및 암호화 오류, MAC 오류, handshake protocol 실패, 인증서 오류 등에 대한 메시지와 설명(alert message)을 상대방에게 전송.

- The Alert protocol defines two fields a severity level (2 level) and an alert description.

17.2 SSL & TLS

SSL Record protocol

- 응용 계층에 보안 서비스 제공;기밀성, 무결성



SSL Record 계층

17.2 SSL & TLS

SSL Record protocol

1) Fragmentation

- 레코드 계층은 응용 계층에서 전달된 데이터를 2^{14} bytes 또는 그 이하의 크기로 분할하여 S니 Plaintext 레코드를 생성

2) Compression

- 모든 레코드는 현재의 세션 상태(session state)에 정의되어 있는 압축 알고리즘을 사용하여 압축
- 현재 SLL v3 & TLS = null, 사용 않함.

17.2 SSL & TLS

3) MAC 생성

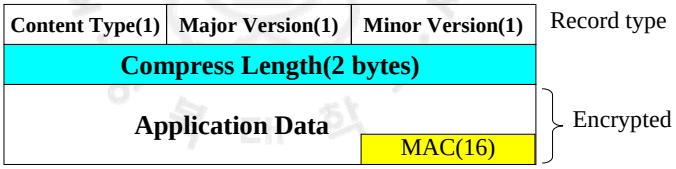
hash(MAC_write_secret + pad_2 +
hash(MAC_write_secret + pad_1 + seq_num +
SSLCompressed.type + SSLCompressed.length +
SSLCompressed.fragment))

MAC_write_secret :	공유 키(Client_MAC_Secret, Server_MAC_Secret)
pad_1	0x36 문자를 MD5에서는 48번 반복값, SHA에서는 40번 반복한 값
pad_2	0x5c 문자를 MD5에서는 48번 반복값, SHA에서는 40번 반복한 값
seq_num	메시지의 순서번호
hash	해쉬 알고리즘

17.2 SSL & TLS

4) Encryption

블록 암호 알고리즘		스트림 암호 알고리즘	
IDEA	128bits	RC4-40	40 bits
RC2-40	40 bits	RC4-128	128 bits
DES-40	40 bits		
DES	56 bits		
3DES	168 bits		
Fortezza	80 bits		



5) Append SSL Record Header

17.2 SSL & TLS

SSL에서 이용되는 알고리즘

- 전자서명 & 키 교환 알고리즘
  - RSA, DH/DHE-DSS, DH-anon, Fortezza
- 암호 알고리즘
  - RC4, RC2, IDEA, DES, 3DES, Fortezza
- 해쉬함수 (MAC을 위한)
  - MD5, SHA-1, (H)MAC over session key, seq#, length, content

17.2 SSL & TLS

TLS

- SSL 3.0을 표준화하기 위해 제안
- IETF TLS WG에서 표준화; V1.0이 발표
- SSL 3.0과 차이점
  - Fortezza 알고리즘이 삭제
  - RSA 알고리즘을 선택적으로 사용 (특허문제)
  - DSS와 D-H를 반드시 등재
  - HMAC 이용; HMAC_MD5, HMAC_SHA-1
  - Master_secret의 계산식이 틀림; PRF (pseudo-random function) 이용

17.2 SSL & TLS

WTLS

- WAP (Wireless Application Protocol)에서 사용되는 TLS
- TLS와의 차이점
  - 키 교환 프로토콜로 ECC 추가
  - Record 단계에서 Fragmentation 및 Compression을 하지 않음
  - 여러 가지 인증서 패턴 사용; X.509 Cert, WTLS Cert, X968 Cert
  - 랜덤 수 크기의 차이
    - Pre_master_secret와 Master_secret는 20 byte
    - Server, Client random는 16 byte
  - Key_block의 계산식이 틀려짐