DOI: 10.3745/KIPSTC.2008.15-C.5.359

SSL 프로토콜의 CipherSuite 설정 문제점과 해결 방안

이 윤 영[†] · 허 순 행[†] · 박 상 주[††] · 신 동 휘[†††] · 원 동 호[††††] · 김 승 주[†††††]

요 약

인터넷과 정보통신기술의 사용이 일반화되면서 인터넷 상에서 중요한 데이터를 안전하게 전송해야 함에 따라 SSL 프로토콜의 사용은 필수요소가 되었다. SSL 프로토콜은 메시지 위/변조 공격과 같은 능동적인 공격에 안전하도록 설계가 되었지만 CipherSuite 설정 변경은 별다른 제약 없이 수정이 가능하다. 공격자가 Client의 시스템의 오작동을 유도하여 CipherSuite 설정을 키 길이가 낮은 대칭키 알고리즘으로 변경한다면 도청하여 데이터를 해독할 수 있다. 본 논문은 키 길이가 낮은 대칭키 알고리즘 설정에 대한 국내 웹사이트의 보안세션 생성 조사 및 CipherSuite 설정 문제점에 대한 해결책을 제시한다.

키워드 : SSL 프로토콜, Ciphersuite, 네트워크 보안

CipherSuite Setting Problem of SSL Protocol and It's Solutions

Yunyoung Lee[†] · Soonhaeng Hur[†] · Sangjoo Park[††] · Donghwi Shin[†††] · Dongho Won[††††] · Seungjoo Kim[†††††]

ABSTRACT

As the use of Internet and information communication technology is being generalized, the SSL protocol is essential in Internet because the important data should be transferred securely. While the SSL protocol is designed to defend from active attack such as message forgery and message alteration, the cipher suite setting can be easily modified. If the attacker draw on a malfunction of the client system and modify the cipher suite setting to the symmetric key algorithm which has short key length, he should eavesdrop and cryptanalysis the encrypt data. In this paper, we examine the domestic web site whether they generate the security session through the symmetric key algorithm which has short key length and propose the solution of the cipher suite setting problem.

Keywords : SSL Protocol, Ciphersuite, Network Security

1. 서 론

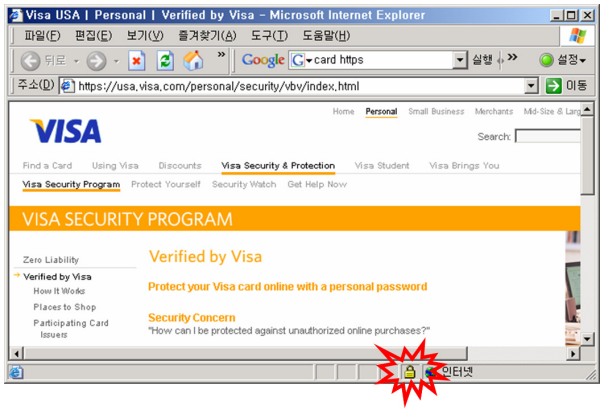
SSL(Secure Sockets Layer) 프로토콜은 네트워크상에서 안전한 데이터 전송을 위해 널리 사용되는 표준 프로토콜이다[1]. 인터넷과 정보통신기술의 사용이 일상화되면서 인터넷 상에서 중요한 데이터를 안전하게 전송해야 함에 따라 SSL 프로토콜의 사용은 필수요소가 되었다. SSL 프로토콜은 공격자가 메시지를 위/변조하는 공격과 같은 능동적인 공격에 강하게 설계되었기 때문에 네트워크상에서 전송되는 SSL 프로토콜의 메시지를 공격하는 방법으로는 의미 있는 결과를 얻기는 힘들다. 그러나 SSL 프로토콜의 CipherSuite 설정 변경은 별다른 제약 없이 수정이 가능하기 때문에 공격자가 Client의 시스템의 오동작을 유도하여 CipherSuite 설정을 변경한다면 공격이 가능하게 할 수 있다. 공격자가 악성코드가 첨부된 메일 또는 악성코드가 설치된 웹 사이트로의 접속을 유도하는 등의 방법을 이용하여 Client의 시스템에 CipherSuite 설정을 키 길이가 낮은 대칭키 알고리즘으로 설정을 하여 도청한다면 데이터를 해독할 수 있다.

본 논문의 2장에서는 SSL 프로토콜에 대해서 살펴본다. 3장에서는 SSL 프로토콜의 CipherSuite 설정의 문제점을 분석하고 이를 이용한 공격 시나리오 및 방법에 대해서 알아본다. 4장에서는 제시한 공격방법을 이용하여 SSL 프로토콜을 주로 사용하는 국내 웹 사이트의 보안 접속 기능을 분석하고 5장에서는 해결책을 제시한다. 그리고 6장에서 결론을 맺는다.

2. SSL 프로토콜

SSL(Secure Sockets Layer) 프로토콜은 네트워크상에서

※ 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-C1090-0801-0028)
※ 본 연구는 지식경제부 및 정보통신연구진흥원의 대학IT연구센터 지원사업의 연구결과로 수행되었음 (IITA-2008-C1090-0801-0016)
† 준 회 원 : 성균관대학교 전자전기컴퓨터공학과 석사과정
†† 준 회 원 : LG전자 연구원
††† 정 회 원 : 한국정보보호진흥원 암호응용팀 연구원
†††† 종신회원 : 성균관대학교 정보통신공학부 교수
††††† 종신회원 : 성균관대학교 정보통신공학부 교수(교신저자)
논문접수 : 2008년 5월 20일
수 정 일 : 2008년 8월 29일
심사완료 : 2008년 9월 4일

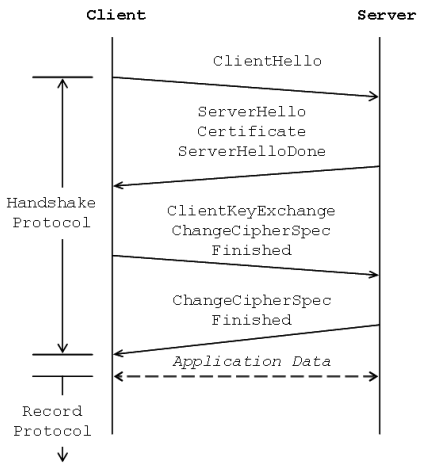


(그림 1) SSL 프로토콜 동작화면

안전한 데이터 전송을 위해 널리 사용되는 표준 프로토콜이다[1]. SSL은 RFC에 등재되면서 TLS(Transport Layer Security)로 이름이 바뀌었는데 본 논문에서는 SSL로 통칭하여 부르기로 하겠다. SSL 프로토콜은 (그림 1)과 같이 Web상의 데이터를 보호하기 위해 많이 쓰이고 있다. 예를 들어 전자상거래 시, 카드번호, 개인정보 등을 안전하게 전송하기 위해 사용된다[5]. 웹브라우저에서 SSL 프로토콜이 동작하면 (그림 1)의 아래 부분과 같이 자물쇠가 표시되는 것을 확인 할 수 있으며 주소창의 웹 주소가 'https'로 시작되는 것을 확인할 수 있다.

SSL 프로토콜은 크게 Handshake 프로토콜과 Record 프로토콜로 나뉜다[9]. Handshake 프로토콜은 SSL 보안 세션을 시작하기 전에 두 통신 매체 간의 각종 보안 파라미터와 세션키를 협의하는 과정이다. Record 프로토콜은 Handshake 프로토콜에서 협의된 암호 알고리즘을 이용하여 데이터를 암호화하여 전송하는 과정이다.

(그림 2)는 클라이언트(Client)와 서버(Server) 사이의 SSL 프로토콜을 간략하게 도식화한 것이다[8]. ClientHello 메시지부터 서버(Server)에서 보낸 Finished 메시지까지가 SSL의 Handshake 프로토콜 부분이고 Application Data가 전송되는 부분이 Record 프로토콜 부분이다. (그림 2)의 각



(그림 2) SSL 프로토콜

메시지 중에서 보안 파라미터가 협의되는 부분은 ClientHello와 ServerHello 메시지 부분이다. ClientHello에서 클라이언트는 자신이 제공할 수 있는 각 보안 파라미터가 정의된 CipherSuite를 서버에 전송한다. ClientHello 메시지를 전달받은 서버는 CipherSuite 중에서 자신이 제공할 수 있고 보안 강도가 가장 강한 CipherSuite를 선택하여 ServerHello 메시지에 실어 클라이언트에 전송한다. ServerHello 메시지를 전달받은 클라이언트는 선택된 CipherSuite를 기초로 서버와 세션 키를 공유하고 Application Data를 암호화한다.

CipherSuite는 공개키, 대칭키, 해쉬 알고리즘의 묶음으로 구성된다[6]. 아래 항목은 SSL 프로토콜에서 대표적으로 제공하는 공개키, 대칭키, 해쉬 알고리즘이다. 보통 공개키, 해쉬 알고리즘은 세션키를 교환하기 위해 사용되며, 대칭키 알고리즘은 Application Data를 암호화하기 위해 사용된다.

- 공개키 알고리즘 : RSA, Diffie-Hellman
- 대칭키 알고리즘 : RC2, RC4, DES, 3DES, IDEA, AES
- 해 쉬 알고리즘 : SHA, MD5

SSL은 위와 같은 공개키, 대칭키, 해쉬 알고리즘을 하나로 묶어 아래와 같이 CipherSuite로 표현한다. 아래는 CipherSuite 중에 하나를 예로 나타낸 것이다. 아래 예제는 TLS 프로토콜 사용하고 RSA를 공개키 알고리즘으로 DES를 대칭키 알고리즘으로 사용하며 CBC 모드로 데이터를 암호화하고 SHA를 해쉬 알고리즘으로 사용하는 CipherSuite이다.

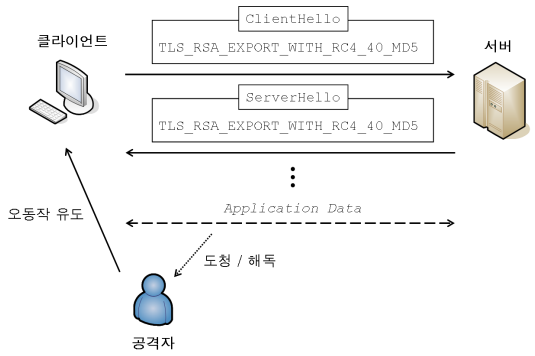
- TLS_RSA_WITH_DES_CBC_SHA

만약 클라이언트와 서버가 Handshake 프로토콜을 통해 위의 CipherSuite를 협의했다면 상호간에 전송되는 Application Data는 DES로 암호화되어 전송되게 된다.

3. 공격 시나리오 및 방법

SSL 프로토콜은 공격자가 메시지를 위/변조하는 공격과 같은 능동적인 공격에 강하게 설계되었기 때문에 네트워크상에서 전송되는 SSL 프로토콜의 메시지를 공격하는 공격 방법으로 의미 있는 결과를 얻기는 매우 힘들다고 볼 수 있다. 하지만 SSL 프로토콜을 사용하는 시스템의 오동작을 유도한다면 공격이 가능할 수도 있다. 공격 시나리오는 (그림 3)과 같다.

① 공격자는 클라이언트가 ClientHello 메시지에 보안 강도가 약한 CipherSuite 만을 서버로 전송하도록 오동작을 유도한다. 예를 들어 대칭키 암호로 RC4 40bit가 있는 CipherSuite 만을 전송하도록 한다.
② ClientHello 메시지를 전달받은 서버는 선택할 수 있는 CipherSuite가 하나 밖에 없으므로 보안 강도가 약한 CipherSuite를 선택하게 된다. 서버는 선택된 CipherSuite를 ServerHello 메시지에 실어서 클라이언트로 보낸다.



(그림 3) 공격 시나리오

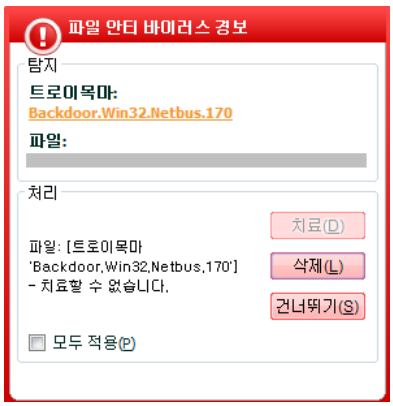
③ 클라이언트와 서버는 보안강도가 약한 CipherSuite를 기반으로 보안 세션을 맺게 되고 Application Data는 RC4 40bit로 암호화 되어 전송된다.

④ 공격자는 RC4 40bit로 암호화된 Application Data를 도청하여 데이터를 해독한다.

이 공격 방법은 CipherSuite rollback 공격과 유사하다. CipherSuite rollback 공격은 클라이언트와 서버간의 Hello 메시지가 평문으로 전송됨으로써 가능한 공격이다. 공격자는 전송되는 ClientHello 메시지를 가로채 클라이언트가 전송하는 CipherSuite 메시지의 CipherSuite 리스트를 수정하여 공격자가 원하는 알고리즘으로 대체하여 공격하는 방식이다. 이는 Handshake 프로토콜의 Finished 메시지에 Handshake 프로토콜에서 사용했던 모든 메시지에 대한 해쉬값을 생성하여 메시지 인증을 함으로써 공격을 방지할 수 있다. 그러나 제안한 공격방법의 경우, CipherSuite 리스트를 수정하는 점은 같지만 전송되는 ClientHello 메시지를 가로채지 않고 클라이언트 시스템의 오동작을 유도하여 SSL을 사용하는 응용 프로그램의 평문으로 저장된 CipherSuite 설정 파일을 수정하여 변경한다는 점이 다르다. 이는 클라이언트에서 보내는 ClientHello 메시지 자체가 변경된 CipherSuite 리스트를 보내기 때문에 Finished 메시지에 의해서 공격이 탐지되지 않는다.

공격자가 클라이언트 시스템의 오동작을 유도하는 방법에는 악성코드가 첨부된 메일을 전송하여 실행을 유도하는 방법, 악성코드가 설치된 웹 사이트로 접속을 유도하는 방법 등 다양한 방법이 있을 수 있다. 실제로 웹 상에서 "컴퓨터 속도 향상 프로그램", "상용 소프트웨어 제품 키 생성 프로그램" 등 악성코드가 포함된 프로그램을 게시하여 다운받아 설치하게 한 후, 백도어(Backdoor), 레지스트리 등록 등 악성코드를 설치하는 프로그램을 쉽게 찾을 수 있다. 이를 이용하여 클라이언트 시스템의 CipherSuite 설정을 변경한다면 쉽게 공격할 수 있다. (그림 4)는 실제 웹 상에 떠돌고 있는 프로그램을 다운받아 설치를 하려고 했을 때 감지된 악성코드를 보여준다.

위와 같은 공격법은 악성코드가 프로세스로 주기적으로 실행되어 시스템에 해를 미치는 일반적인 방법과는 달리 한 번만 실행하여 오동작을 유도하기만 하면 된다. 클라이언트와 서버 시스템은 보안 강도가 낮은 보안 세션이 생성되는



(그림 4) 악성코드 감지

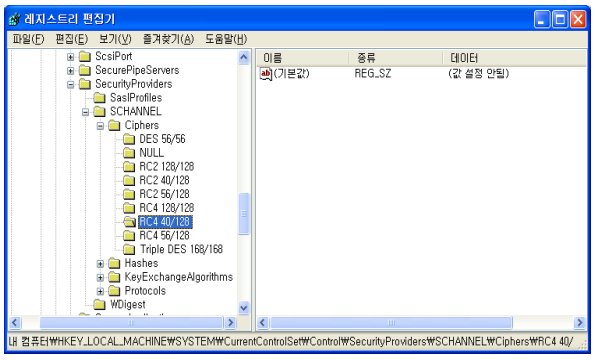
것을 정상적인 상태로 인식할 것이다. 따라서 위 공격법은 일반적인 사용자가 현재 오동작의 상태를 감지하기 힘들며 알려지지 않은 악성코드일 경우 바이러스 백신에서 이를 감지하기도 힘들 수 있다.

3.1 Internet Explorer

Microsoft사의 Windows 운영체제는 SSL 프로토콜을 지원하기 위해 RSA 사에서 개발한 'Schannel.dll' 라이브러리를 사용한다. Windows 운영체제의 레지스트리를 수정하는 방법으로 'Schannel.dll'에서 사용할 CipherSuite를 조정할 수 있다[2]. Internet Explorer 역시 해당 라이브러리를 사용하기 때문에 레지스트리를 수정하면 Internet Explorer도 같은 방식으로 동작하게 된다.

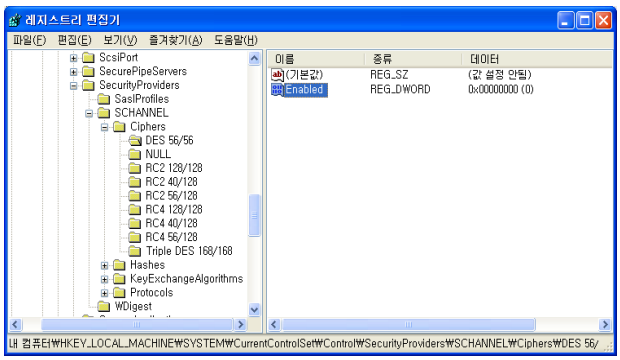
레지스트리의 아래 경로를 따라 들어가면 (그림 5)와 같이 공개키, 대칭키, 해쉬 알고리즘을 설정할 수 있는 부분이 나온다.

· HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers



(그림 5) Schannel.dll의 CipherSuite 수정

이때 Cipher에 속에 있는 "RC4 40/128"키를 제외한 나머지 모든 키에 "Enabled"라는 이름의 "DWORD" 값을 생성하여 "0x00000000" 값을 할당한다(그림 6). 이렇게 되면 RC4 40bit 대칭키 알고리즘을 제외한 나머지 알고리즘은 사용할 수 없게 된다.

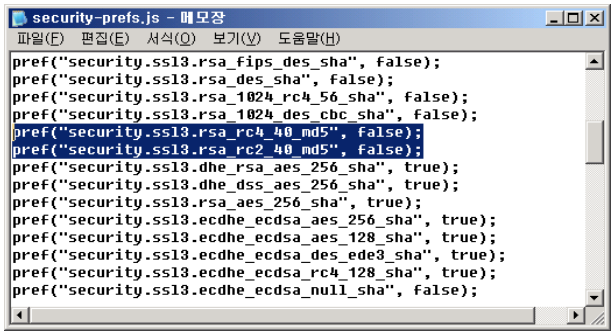


(그림 6) DES 56/56 암호 알고리즘 Disable

3.2 Firefox

Firefox 역시 Internet Explorer와 같이 사용할 CipherSuite를 제한할 수 있다. 아래 경로의 파일을 열면 (그림 7)과 같이 CipherSuite를 선택할 수 있다.

· C:\Program Files\Mozilla Firefox\greprefs\security-prefs.js



(그림 7) Firefox의 CipherSuite 수정

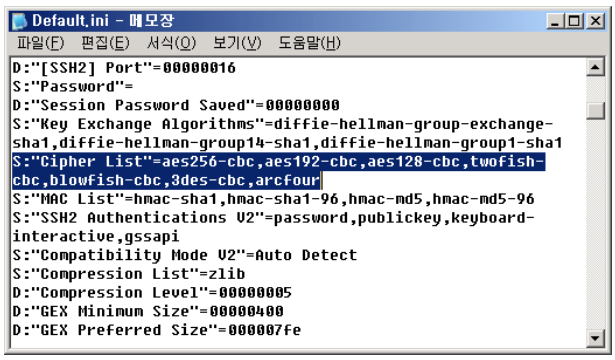
Firefox v.2.0.0.11을 기준으로 RC4 40bit를 지원하는 CipherSuite는 디폴트로 사용할 수 없게 되어있다. 하지만 일반적인 텍스트 파일이기 때문에 이를 수정하기는 쉽다. (그림 7)에서 'ssl3.rsa_rc4_40_md5'가 있는 필드의 'false'값을 'true'변경하면 되기 때문이다.

3.3 SecureCRT

SecureCRT는 유닉스 서버에 ssh 프로토콜을 이용하여 접속할 수 있도록 해주는 터미널 프로그램이다. ssh 프로토콜은 데이터를 안전하게 전송할 수 있도록 SSL 프로토콜을 사용한다.

SecureCRT도 역시 CipherSuite를 제한할 수 있다. 아래 경로의 파일을 열면 (그림 8)과 같이 CipherSuite를 선택할 수 있다. 아래 경로에서 <사용자 ID>는 Windows 시스템에 로그인한 사용자의 ID이다. (그림 8)의 'Cipher List'에서 RC4(arcfour)를 제외하고 나머지를 모두 지운다면 RC4만 동작하게 된다.

· C:\Documents and Settings\<사용자 ID>\Application Data\VanDyke\Config\Sessions\Default.ini



(그림 8) SecureCRT의 CipherSuite 수정

4. 웹 사이트 보안 세션 생성 현황

국내에서 SSL 프로토콜을 주로 사용하는 경우로 웹 사이트의 보안 접속 기능이 있을 수 있다. 일반적으로 웹 사이트에 ID와 패스워드를 입력한 후 로그인 할 때 해당 ID와 패스워드는 평문으로 클라이언트에서 서버로 전송된다. 따라서 이를 방지하기 위해서 보안 접속 기능을 이용하여 SSL 프로토콜로 해당 ID와 패스워드를 보호하는 기능이 주로 이용된다. Internet Explorer의 SSL 보안 설정을 변경한 뒤 국내 사이트를 대상으로 실험을 하였다. 실험환경은 다음과 같다.

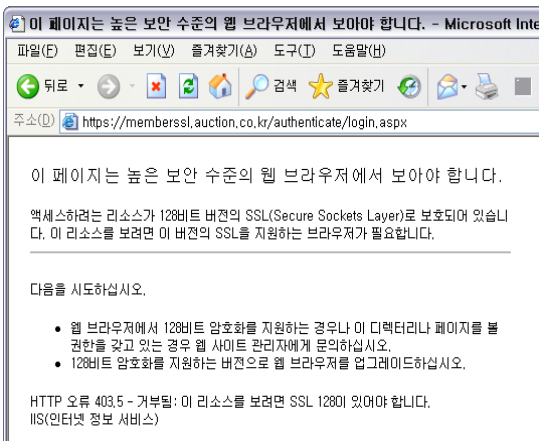
· 소프트웨어: Internet Explorer
· CipherSuite: TLS_RSA_EXPORT_WITH_RC4_40_MD5

위의 환경에서 국내 사이트의 보안 접속(로그인)을 시도하여 네트워크 패킷을 Ethereal 소프트웨어로 캡쳐하여 확인하였다. 실험결과 대부분의 국내 사이트가 RC4 40bit 대칭키 알고리즘으로 보안세션이 생성되는 것을 확인할 수 있었다 <표 1>.

실험결과 중 특이한 점은 Auction 사이트의 경우 RC4 40bit로 보안 세션을 수립했을 때 사이트에서 128bit로 업그레이드 하라는 메시지를 사용자에게 보여준다는 것이다(그림 9).

〈표 1〉 국내 사이트 RC4 40bit 보안 세션 실험 결과

웹 사이트	40비트 통신가능여부
Auction	X
Dnshop	O
Gmarket	O
Interpark	O
Daum	O
Empas	O
MSN	O
NATE	O
Naver	O
Sayclub	O
Yahoo	O
Dreamwiz	O



(그림 9) Auction 보안 경고

5. 문제점 해결 방안

SSL 보안 설정에 대한 문제점을 정리하면 다음과 같다.

· SSL 프로토콜의 호환성에 대한 문제점
· SSL을 사용하는 클라이언트 시스템의 보안성을 확신할 수 없음
· SSL을 사용하는 SW에 대한 보안 파라미터들을 누구나 수정할 수 있음

SSL 프로토콜은 지속적으로 업그레이드되면서 이전 버전에 대한 호환성을 지원하고 있다. RC4 40bit 보안 세션이 아직도 사용될 수 있는 이유가 호환성을 지원하고 있기 때문이다.

그리고 이러한 공격이 가능한 이유는 클라이언트 시스템이 서버 시스템에 비해서 상대적으로 보안이 취약하기 때문이다. 따라서 공격자는 상대적으로 보안이 취약한 클라이언트 시스템의 SSL 보안 설정 부분을 공격하여 SSL의 보안 강도를 낮출 수 있는 것이다. 공격자가 클라이언트 시스템의 SSL 보안 설정을 변경하여도 SSL 프로토콜 상에서는 이를 감지할 수 없기 때문에 이러한 문제가 발생한다.

또한 클라이언트 시스템의 SSL 보안 설정이 평문으로 기록되어 있기 때문에 누구나 쉽게 보안 설정을 변경할 수 있는 것이 문제점이다. 공격자는 시스템에 설치된 소프트웨어의 SSL 보안 설정을 별다른 제약 없이 수정할 수 있다는 것이다.

5.1 SSL 프로토콜 수정

앞서 분석한 문제점을 해결하기 위한 방안 중 첫 번째로 SSL 프로토콜이 보안에 취약한 세션을 성립할 수 없도록 수정을 가하는 것이다. 현재 SSL 프로토콜에서 지원하는 대칭키 알고리즘은 RC2, RC4, DES, 3DES, IDEA, AES 등이 있다. 일반적으로 60bit 이하의 길이를 가지는 키를 사용하고 있는 대칭키 알고리즘을 취약하다고 알려져 있다. 따라서 취약한 키 길이를 가지고 있는 대칭키 알고리즘을 사용할 수 없도록 SSL 프로토콜을 수정하는 작업이 필요하다. 그 대상으로 아래의 대칭키 알고리즘을 사용할 수 없도록 제한해야 한다.

· DES (56bits), RC2 (40,56 bits), RC4 (40,56 bits)

5.2 SSL 서버 설정 수정

SSL 프로토콜은 서버와 클라이언트 간의 협의를 통해 보안 세션을 설정하므로 서버와 클라이언트에 대한 보완책이 필요하다. 본 절에서는 SSL 서버의 보안 설정에 대해서 알아본다.

5.2.1 서버 소프트웨어 설정 수정

Apache 웹 서버의 경우 SSL 모듈을 추가하면 웹 서버에서 SSL 프로토콜을 지원할 수 있다. 이 경우 SSL 보안 설정이 디폴트로 모든 CipherSuite를 지원할 수 있도록 되어 있다. 보안 강도가 약한 CipherSuite가 클라이언트로부터 제공될 경우 서버는 이를 지원하도록 되어 있는 것이다. 따라서 서버 관리자는 SSL 모듈 설치 시에 보안 강도가 강한

```
[uylee@apache]$openssl ciphers -v
DHE-RSA-AES256-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(256)  Mac=SHA1
DHE-DSS-AES256-SHA      SSLv3 Kx=DH       Au=DSS  Enc=AES(256)  Mac=SHA1
AES256-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(256)  Mac=SHA1
EDH-RSA-DES-CBC3-SHA    SSLv3 Kx=DH       Au=RSA  Enc=3DES(168) Mac=SHA1
EDH-DSS-DES-CBC3-SHA    SSLv3 Kx=DH       Au=DSS  Enc=3DES(168) Mac=SHA1
DES-CBC3-SHA            SSLv3 Kx=RSA      Au=RSA  Enc=3DES(168) Mac=SHA1
DES-CBC3-MD5            SSLv2 Kx=RSA      Au=RSA  Enc=3DES(168) Mac=MD5
DHE-RSA-AES128-SHA      SSLv3 Kx=DH       Au=RSA  Enc=AES(128)  Mac=SHA1
DHE-DSS-AES128-SHA      SSLv3 Kx=DH       Au=DSS  Enc=AES(128)  Mac=SHA1
AES128-SHA              SSLv3 Kx=RSA      Au=RSA  Enc=AES(128)  Mac=SHA1
RC2-CBC-MD5             SSLv2 Kx=RSA      Au=RSA  Enc=RC2(128)  Mac=MD5
DHE-DSS-RC4-SHA         SSLv3 Kx=DH       Au=DSS  Enc=RC4(128)  Mac=SHA1
EXP-KRB5-RC4-MD5        SSLv3 Kx=KRB5     Au=KRB5 Enc=RC4(40)   Mac=MD5  export
EXP-KRB5-RC4-SHA        SSLv3 Kx=KRB5     Au=KRB5 Enc=RC4(40)   Mac=SHA1 export
KRB5-RC4-MD5            SSLv3 Kx=KRB5     Au=KRB5 Enc=RC4(128)  Mac=MD5
KRB5-RC4-SHA            SSLv3 Kx=KRB5     Au=KRB5 Enc=RC4(128)  Mac=SHA1
RC4-SHA                 SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=SHA1
RC4-MD5                 SSLv3 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=MD5
RC4-MD5                 SSLv2 Kx=RSA      Au=RSA  Enc=RC4(128)  Mac=MD5
KRB5-DES-CBC3-MD5       SSLv3 Kx=KRB5     Au=KRB5 Enc=3DES(168) Mac=MD5
KRB5-DES-CBC3-SHA       SSLv3 Kx=KRB5     Au=KRB5 Enc=3DES(168) Mac=SHA1
RC4-64-MD5              SSLv2 Kx=RSA      Au=RSA  Enc=RC4(64)   Mac=MD5
EXP1024-DHE-DSS-DES-CBC-SHA SSLv3 Kx=DH(1024) Au=DSS  Enc=DES(56)   Mac=SHA1 export
EXP1024-DES-CBC-SHA     SSLv3 Kx=RSA(1024) Au=RSA  Enc=DES(56)   Mac=SHA1 export
EXP1024-RC2-CBC-MD5     SSLv3 Kx=RSA(1024) Au=RSA  Enc=RC2(56)   Mac=MD5  export
KRB5-DES-CBC-MD5        SSLv3 Kx=KRB5     Au=KRB5 Enc=DES(56)   Mac=MD5
KRB5-DES-CBC-SHA        SSLv3 Kx=KRB5     Au=KRB5 Enc=DES(56)   Mac=SHA1
EDH-RSA-DES-CBC-SHA     SSLv3 Kx=DH       Au=RSA  Enc=DES(56)   Mac=SHA1
EDH-DSS-DES-CBC-SHA     SSLv3 Kx=DH       Au=DSS  Enc=DES(56)   Mac=SHA1
DES-CBC-SHA             SSLv3 Kx=RSA      Au=RSA  Enc=DES(56)   Mac=SHA1
DES-CBC-MD5             SSLv2 Kx=RSA      Au=RSA  Enc=DES(56)   Mac=MD5
EXP1024-DHE-DSS-RC4-SHA SSLv3 Kx=DH(1024) Au=DSS  Enc=RC4(56)   Mac=SHA1 export
EXP1024-RC4-SHA         SSLv3 Kx=RSA(1024) Au=RSA  Enc=RC4(56)   Mac=SHA1 export
EXP1024-RC4-MD5         SSLv3 Kx=RSA(1024) Au=RSA  Enc=RC4(56)   Mac=MD5  export
EXP-KRB5-RC2-CBC-MD5    SSLv3 Kx=KRB5     Au=KRB5 Enc=RC2(40)   Mac=MD5  export
EXP-KRB5-DES-CBC-MD5    SSLv3 Kx=KRB5     Au=KRB5 Enc=DES(40)   Mac=MD5  export
EXP-KRB5-RC2-CBC-SHA    SSLv3 Kx=KRB5     Au=KRB5 Enc=RC2(40)   Mac=SHA1 export
EXP-KRB5-DES-CBC-SHA    SSLv3 Kx=KRB5     Au=KRB5 Enc=DES(40)   Mac=SHA1 export
EXP-EDH-RSA-DES-CBC-SHA SSLv3 Kx=DH(512)  Au=RSA  Enc=DES(40)   Mac=SHA1 export
EXP-EDH-DSS-DES-CBC-SHA SSLv3 Kx=DH(512)  Au=DSS  Enc=DES(40)   Mac=SHA1 export
EXP-DES-CBC-SHA         SSLv3 Kx=RSA(512) Au=RSA  Enc=DES(40)   Mac=SHA1 export
EXP-RC2-CBC-MD5         SSLv3 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC2-CBC-MD5         SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
EXP-RC4-MD5             SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40)   Mac=MD5  export
[uylee@apache]$
```

(그림 10) openSSL에서 지원하는 CipherSuite

```
# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate.
# See the mod_ssl documentation for a complete list.
# SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
SSLCipherSuite EXP-RC4-MD5
```

(그림 11) Apache CipherSuite 설정

CipherSuite만을 지원할 수 있도록 하여 문제점을 해결 할 수 있다.

다음은 Apache 웹 서버에서 CipherSuite를 수정하는 방법이다. (그림 10)는 openSSL에서 지원하는 CipherSuite를 보여준다. 이것을 참조하여 Apache의 apache/conf/extra/httpd-ssl.conf 파일을 수정하여 저장한 후 Apache를 재시작하면 적용된다[7]. (그림 11)은 TLS_RSA_EXPORT_WITH_RC4_40_MD5만을 서버가 지원하겠다고 하는 설정이다.

5.2.2 서버 서비스 수정

웹 사이트의 보안 접속(로그인) 기능을 볼 때 SSL 프로토콜을 사용하여 사이트에 접속하는 기능을 제공한다. SSL 프로토콜을 사용하여 ID와 패스워드를 암호화하는 기능를 제공하는데 어떠한 보안 강도로 암호화 할 지는 사용자가 선택할 수 없는 형태이다. 따라서 사용자가 자신이 접속할 때 사용할 SSL 프로토콜의 보안 강도를 선택할 수 있도록 서비스를 구성하여 문제점을 해결하는 방법을 생각해 볼 수 있다.

서버의 보안 접속 기능을 이용하고자 할 때 사용자가 선택할 수 있는 보안 강도를 나열하여 이를 사용자로 하여금 선택하도록 하는 기능이다. 예를 들어 사용자가 AES-256 bit의 보안 강도를 선택하였다면 서버는 이를 감지하여 ClientHello 메시지의 CipherSuite 중에서 AES-256 bit를 지원하는 CipherSuite를 선택하여 ServerHello 메시지에 넣어 SSL 보안 강도를 결정하는 것이다. 따라서 이러한 방식을 사용할 경우 Application 계층과 분리되어 있는 SSL 프로토콜이 서비스로 구현되면서 Application 계층과 통합된 형태로 운영되어야 할 필요성이 생긴다.

5.3 클라이언트 설정 수정

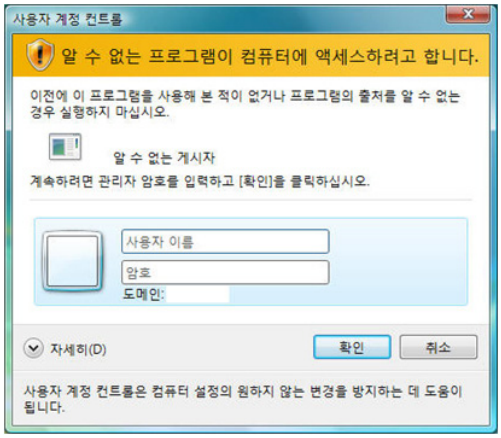
클라이언트의 경우 SSL 보안 설정을 누구나 수정할 수 있으므로 이를 제어할 필요가 있다. 이를 제어하는 방법으로 접근 제어를 이용한 방법, 보안 설정 부분의 암호화, 사용자의 확인 등이 있을 수 있다.

5.3.1 접근 제어

클라이언트의 SSL 보안 설정을 아무나 수정할 수 없도록 운영체제 상에서 접근 제어를 지원하는 방법이 있을 수 있다. Internet Explorer의 경우 SSL 보안 설정 부분이 레지스트리에 있으므로 접근 제어를 이용하기가 용이할 수 있다. 실제로 Windows Vista의 경우 이러한 기능을 제공하고 있다.

Windows Vista는 User Account Control(UAC)를 통해 운영체제 상의 사용자 계정과 관리자 계정을 분리하고 있다[3]. UAC는 프로세스가 관리자의 권한이 필요한 작업을 할 시에 관리자의 패스워드를 물어보는 기능을 통해 악성코드 등이 임의로 시스템 작업을 하는 것을 막고 있다(그림 12).

Windows Vista에서 관리자의 권한이 필요한 대표적인 경우는 다음과 같다[3]. 이중에서 '레지스트리 중 HKEY_LOCAL_MACHINE에 대한 쓰기 작업'은 Internet Explorer의 SSL 보안 설정을 변경하기 위해 필요한 작업이다. 따라서 공격자



(그림 12) Windows Vista의 UAC

가 임의로 SSL 보안 설정을 변경하려 할 때 UAC 기능을 통해 이를 방지할 수 있다.

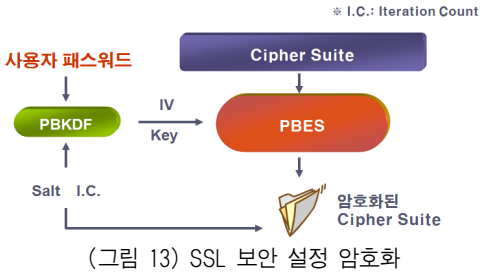
- %Systemdrive% 대한 핸들링
- %Systemroot%(Windows), Program Files 폴더에 대한 쓰기 작업
- 레지스트리 중 HKEY_LOCAL_MACHINE에 대한 쓰기 작업
- Internet Explorer에 대한 각종 설정 변경 등
- Windows 시스템의 각종 설정들 (정확하게 Administrators에게 권한이 부여된 모든 작업)

접근 제어를 통해 SSL 보안 설정을 방지하는 방법은 운영체제가 이를 지원해야 한다는 제약점이 있다. 현재 Windows Vista를 제외한 Windows 계열 운영체제들은 이러한 접근제어 기능을 제공하고 있지 않기 때문에 접근 제어를 이용한 방식은 제약사항이 있다.

5.3.2 SSL 보안 설정 암호화

SSL 프로토콜을 지원하는 소프트웨어의 SSL 보안 설정을 암호화하는 방법이 있을 수 있다. 앞서 제기한 문제점의 근본적인 원인은 SSL 보안 설정이 평문으로 기록되어 있는 것이었다. 따라서 SSL 보안 설정을 암호화해서 저장하여 다른 누군가가 이를 변경하는 것을 방지할 수 있다.

SSL 보안 설정을 암호화하는 방법 중 하나로 사용자 패스워드로부터 키를 추출하여 SSL 보안 설정을 대칭키 알고리즘으로 암호화하는 방법이다(그림 13). 사용자 패스워드를



(그림 13) SSL 보안 설정 암호화

PBKDF 함수에 넣어 Salt와 I.C.를 통해 IV와 Key를 생성한다. 그리고 CipherSuite를 PBES 함수에 넣어 이를 암호화하여 암호화된 CipherSuite를 파일로 생성하고 여기에 Salt와 I.C.를 같이 기록하여 이후에 복호화 할 때 사용할 수 있도록 하는 것이다. PBKDF 함수는 RSA사에서 개발한 함수로 패스워드로부터 Key를 추출하는 함수이고 PBES 함수 역시 RSA사에서 개발한 함수로 패스워드로부터 추출한 Key를 이용하여 데이터를 암호화 하는 것을 표준으로 정의하고 있다. 암호화된 CipherSuite는 SSL Handshake 프로토콜 이전에 복호화 되어 사용된다.

이 방식이 실제 구현 될 경우 소프트웨어는 실행되기 전에 SSL 보안 설정 부분을 복호화 하기 위해 사용자의 패스워드를 묻는 절차를 필요로 하게 될 것이다. 사용자의 패스워드로 암호화된 보안 설정을 복호화 하여 이후 SSL 프로토콜 진행시에 사용해야 하기 때문이다. 따라서 이러한 방식을 사용할 경우 Application 계층과 분리되어 있는 SSL 프로토콜이 소프트웨어로 구현되면서 Application 계층과 통합된 형태로 운영되어야 할 필요성이 생긴다.

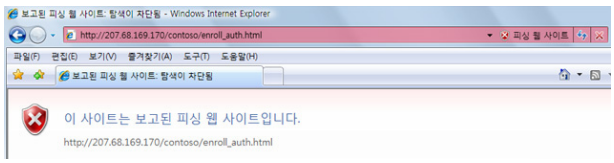
5.3.3 SSL 보안 세션의 사용자 확인

SSL 프로토콜을 지원하는 소프트웨어의 경우 보안 세션이 수립될 때 보안 강도가 어떻게 되고 사용되는 대칭키 알고리즘이 어떠한 것인지 사용자가 명시적으로 확인하기 힘든 경우가 있다. 이러한 경우 보안 강도가 약한 보안 세션이 수립되더라도 사용자가 이를 확인하기 힘들 수 있기 때문에 문제가 발생한다. 따라서 SSL 보안 세션이 수립될 경우 소프트웨어에서 이를 명시적으로 확인할 수 있도록 해주는 방법이 필요하다.

Internet Explorer 7의 경우 현재 웹 사이트의 상태를 주소 창의 색상으로 나타내어주는 기능이 있다[4]. SSL 인증서가 있는 웹 사이트에 접속했을 경우 녹색, 피싱 사이트에 접속했을 경우에 빨간색으로 나타내는 것이다(그림 14).

이와 유사한 방법으로 각각의 보안강도에 대해 색상을 설정하여 사용자가 자신이 어떠한 보안강도를 이용하여 통신을 하고 있는지 알 수 있도록 해 줄 수 있다. 보안강도가 약할 경우 빨간색, 보안강도가 강할 경우 파란색 등을 예로 들 수 있다. 따라서 소프트웨어가 SSL 보안 세션을 맺을 때 현재의 보안 강도가 어느 정도인지 사용자는 확인할 수 있으며 연결된 SSL 보안 세션의 강도가 낮을 경우 사용자에게 경고 메시지를 보낼 수 내 현재 연결된 SSL 보안 세션의 보안 강도를 확인 할 수 있도록 할 수 있다.

해결 방안을 정리하여 장, 단점을 기술하면 <표 2>와 같다.



(그림 14) Internet Explorer의 피싱 관련 보안 상태 표시줄

〈표 2〉 해결방안의 장 · 단점

해결책		장점	단점
SSL 프로토콜 수정		• SSL을 지원하는 모든 프로그램에 대해 적용 가능	• SSL 프로토콜을 자체를 수정해야 함
서버 설정 수정	소프트웨어 설정수정	• 클라이언트의 수정 불필요	• 지원가능한 CipherSuite가 제한됨
	서비스 수정	• 클라이언트가 원하는 보안강도를 제공	• 프로그램과의 연동 필요
클라이언트 설정 수정	접근제어	• 파일에 대한 권한이 있는 사용자만이 프로그램의 보안 설정을 변경할 수 있음	• 운영체제에서 지원해야 함
	보안설정 파일 암호화	• 키를 알고있는 사용자만이 프로그램의 보안 설정을 변경할 수 있음	• 프로그램과의 연동 필요
	보안세션 확인	• 생성되는 보안세션마다 보안강도를 확인할 수 있음	• 실제 보안세션이 생성된 후 사용자가 보안강도를 알 수 있음

6. 결 론

SSL 프로토콜은 안전한 데이터 전송을 위해 사용되는 표준 프로토콜로서 공격자가 메시지를 위/변조하는 등의 능동적인 공격에 강하게 설계되었다. Handshake 과정에서 클라이언트는 ClinetHello를 통하여 자신이 제공할 수 있는 각 보안 파라미터가 정의된 CipherSuite를 서버에 전송을 하고 서버는 CipherSuite 중에서 자신이 제공할 수 있는 가장 강한 CipherSuite를 선택하여 SeverHello 메시지에 실어 클라이언트에게 전송을 한다. 그 후 클라이언트는 선택된 보안 파라미터를 기초로 서버와 세션 키를 공유하고 Application Data를 암호화한다. 그러나 SSL 프로토콜의 CipherSuite 설정 변경은 별다른 제약 없이 수정이 가능하여 공격자가 Clinet의 시스템의 오동작을 유도하여 CipherSuite 설정을 변경한다면 전송되는 메시지를 도청하여 해독이 가능하다.

본 논문에서는 SSL 프로토콜의 호환성에 대한 문제점, SSL을 사용하는 클라이언트 시스템의 보안성을 확신할 수 없음, 그리고 SSL을 사용하는 SW에 대한 보안파라미터들을 누구나 수정할 수 있다는 문제점을 지적하고 그에 따른 해결책을 제시하였다. 제시한 해결책을 SSL 프로토콜에 적용하여 통신을 한다면 서버와 클라이언트는 더 안전한 보안 세션을 생성하여 데이터를 전송할 수 있을 것이다.

참 고 문 헌

[1] Dierks, T. and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1”, RFC 4346, April 2006.

[2] "How to Restrict the Use of Certain Cryptographic Algorithms and Protocols in Schannel", Microsoft, 2004.
[3] 백승주, "Windows Vista의 사용자 계정 컨트롤 (User Account Control : UAC)", http://www.microsoft.com/korea/technet/resources/Technetcolumn/column_uac1.mspx, Microsoft, 2006.
[4] "Internet Explorer7 홈페이지", http://www.microsoft.com/korea/windows/products/winfamily/ie/default.mspx, Microsoft, 2007.
[5] Ashraf Elgohary, Tarec S. Sobh, M. Zaki, "Design of an enhancement for SSL/TLS protocols", Computers & Security, Vol.25, Issue 4, pp.297-306 June 2006,
[6] Secure password-based cipher suite for TLS.
[7] John Viega, Matt Messier, Pravir "Network security with OpenSSL", O'REILLY.
[8] Eric Rescorla, "SSL and TLS Designing and Building Secure Systems", Addison-Wesley.
[9] David Wagner, Bruce Schneier, "Analysis of the SSL 3.0 protocol", USENIX Workshop on Electronic Commerce, ACM.

이 윤 영

e-mail : yylee@security.re.kr
2007년 성균관대학교 정보통신공학부 (학사)
2007년~현 재 성균관대학교 전자전기컴퓨터공학과 석사과정
관심분야 : 정보보호, 네트워크 보안 등

허 순 행

e-mail : shhur@security.re.kr
2007년 성균관대학교 정보통신공학부 (학사)
2007년~현 재 성균관대학교 전자전기컴퓨터공학과 석사과정
관심분야 : 정보보호, 보안성 평가 등

박 상 주

e-mail : spark@security.re.kr
2006년 성균관대학교 정보통신공학부 (학사)
2008년 성균관대학교 전자전기컴퓨터공학과(공학석사)
2008년~현 재 LG전자 연구원
관심분야 : 정보보호, 네트워크 보안 등

신 동 휘

e-mail : dhshin@security.re.kr
2002년 성균관대학교 자연과학부(학사)
2002년 성균관대학교 전자전기컴퓨터공학부(학사)
2008년 성균관대학교 전자전기컴퓨터공학과(공학석사)
2008년~현 재 한국정보보호진흥원 연구원
관심분야 : 네트워크 보안, 침투 테스트, 정보보호 응용 등

원 동 호

e-mail : dhwon@security.re.kr
1976년~1988년 성균관대학교 전자공학과 (학사, 석사, 박사)
1978년~1980년 한국전자통신연구원 전임연구원
1985년~1986년 일본 동경공업대 객원연구원
1988년~2003년 성균관대학교 교학처장, 전기전자 및 컴퓨터공학부장, 정보통신대학원장, 정보통신기술연구소장, 연구처장
1996년~1998년 국무총리실 정보화추진위원회 자문위원
2002년~2003년 한국정보보호학회장
2002년~현 재 대검찰청 컴퓨터범죄수사 자문위원, 감사원 IT 감사 자문위원
2007년~현 재 성균관대학교 정보통신공학부 교수, 한국정보보호학회 명예회장, 정보통신부지정 정보보호인증기술연구센터 센터장, 정보통신대학원장
관심분야 : 암호이론, 정보이론, 정보보호

김 승 주

e-mail : skim@security.re.kr
1994년~1999년 성균관대학교 정보공학과 (학사, 석사, 박사)
1998년~2004년 한국정보보호진흥원(KISA) 팀장
2004년~현 재 성균관대학교 정보통신공학부 부교수
2004년 1월~현 재 한국정보보호학회 이사
2005년 6월~2006년 6월 교육인적자원부 유해정보차단 자문위원
2006년~Marquis Who's Who in Asia (2007: 1st Edition) 인명사전 등재
2006년 3월~국가정보원장 국가사이버안전업무 유공자 표창
2007년~Marquis Who's Who in the World (2008: 25th Edition) 인명사전 등재
2007년 5월~현 재 대검찰청 디지털수사자문위원
2007년 12월~현 재 전자정부서비스보안위원회 사이버침해사고대응실무위원
2008년 1월~현 재 기술보증기금 외부자문위원
2008년 2월~현 재 수원시 지역 정보화 촉진 협의회 위원
2008년 4월~현 재 한국은행 금융정보화추진분과위원회 자문위원
2008년 4월~현 재 법무부 법률서비스산업 경쟁력강화 위원회 위원
관심분야 : 암호이론, 정보보호표준, 정보보호제품 및 스마트카드 보안성 평가, PET