웹서비스 보안기술의 향상방안에 대한 제안

이대하*, 김록원*, 정승우*, 송병열*, 조현규*
*전자통신연구원 지능형 웹기술 연구팀
e-mail:bigsum@etri.re.kr

The proposal for improvement of Webservice security

Dae-ha Lee*, Rock-won Kim*, Seung-woo Jung*,
Byoung-Youl Song*, Hyun-kyu Cho*
*Intelligent Web Technology Research Team, ETRI

요 약

최근 이 기종 시스템 및 애플리케이션의 상호호환을 위해 웹 기반 서비스에 대한 관심이 고조되어 가고 있다. 웹의 편리함과 동시에 누구에게나 접근 가능한 웹의 개방성은 보안에 대한 문제점을 야기시키는데, 웹서비스 보안기술은 이에 대한 해결책을 제시하고 있다. 본고에서는 웹서비스 보안기술에 대해 살펴보고, 웹서비스 보안 기술의 취약점 및 이를 보완할 수 있는 향상방안에 대해 기술하고자 한다.

1. 서론

이 기종 시스템이나 소프트웨어간의 통합이나 호환을 위해 이전부터 여러 가지 노력이 있어왔으나, 벤더들 간의 상호협조 부족 및 관련표준 통합의 어려움 등으로 인해 아직까지 뚜렷한 결과를 볼 수는 없는 실정이다. 최근 웹 기반 기술의 발달과 주요 벤더들 간의 상호이해를 바탕으로 사용자 친화적이면서 웹의 폭발적인 파급 효과 등을 고려 하여 웹 기반 하에서 이러한 통합작업을 이루려는 움직임이 일고 있다. 대표적으로 Microsoft와 IBM 등이 주축이 된 웹서비스 관련 기술들이 그러한 예인데, 이에는 SOAP[1], UDDI[2], WSDL[3] 등이 있다.

웹 기반 하에서의 작업은 사용자 편리성 및 애플리케이션 개발에 있어 많은 장점을 제공하지만, 이와 아울러 웹의 개방적인 특성으로 인해 보안에 대한 문제점도 동시에 부각되어진다. 웹서비스 보안기술은 웹서비스를 통한 이 기종 시스템 및 애플리케이션 통합작업에 있어 그 선결조건으로 해결되어야 할 중요한 기술로 받아들여지고 있으며, 이에 대한 표준 및 개발 작업도 활발히 진행 중에 있다.

웹서비스 기술에 대한 초기 표준작업은 대부분 Microsoft와 IBM 등에 의해 이루어졌으며, 현재는 W3C, OASIS, IETF 등의 주요 표준화 기구에서 표준화 작업이 이루어지고 있다.
웹서비스 보안기술에 대한 표준화 작업은 OASIS의 WSS TC(Web Services Security Technical Communities)에서 이루어지고 있다. 현재 Working Draft 상태로 SOAP Message Security[4] 라는 이름으로 진행되어지고 있다.

본 고에서는 SOAP Message Security 에 대해 살펴보고, 이에 대한 문제점 및 그 해결방안에 대해 기술할 것이다.

2. Web Services Security

웹서비스 보안은 SOAP 메시지에 대한 보안에 중점을 두고 있는데, SOAP 메시지의 무결성과 기밀성, 메시지의 인증을 통하여 SOAP 메시지를 보호하게 된다. 웹서비스 보안 메커니즘은 기존에 존재하는 다양한 보안 모델과 암호 기술을 수용하는 형태로 설계되어졌다. 이는 또한 메시지와 관련된 보안 토큰에 대한 일반적인 메커니즘을 제공한다.
웹서비스 보안은 특별한 형태의 보안 토큰에 구애받지 않고, 다양한 형태의 보안 토큰에 적합하게 확장 가능한 형태로 설계되어졌다. 부가적으로 보안 토큰을 어떻게 인코드할 것인지에 대해서도 기술하고 있는데, 특별히 스펙에서는 X.509 인증서[5]와 Kerberos Ticket[6]에 대한 인코딩 방법을 기술하고 있으며, 암호화된 키를 어떻게 포함할 것인지에 대해서도 기술하고 있다.

표 1은 SOAP Message Security 실제 예를 나타낸다. 각 부분에 대해 간략히 설명하면,

- **Routing Information** : SOAP 메시지의 수신지와 송신지에 대한 정보를 나타낸다.
- **Timestamp** : 보안 정보의 재사용을 방지하기 위

해 사용되며, 보안 정보의 생성시간 및 유효기간 등으로 구성되어진다.

- **Security Token** : 보안에 관련된 정보로서, 이는 다시 Unsigned Security Token과 Signed Security Token 두 가지로 나뉜다.

Unsigned Security Token은 인증기관에 의해 승인되지 않은 Security Token으로, 보안등급이 낮은 경우에 적용할 수 있는 정보로서 예를 들면, 사용자 이름(Username) 등을 들 수 있다.

Signed Security Token은 인증기관에 의해 승인되고, 그 인증기관에 의해 암호학적으로 서명되어진 Security Token으로서, 이에는 X.509인증서[5]나 Kerberos Ticket[6] 등이 있다.

- **Encrypted Key** : SOAP Body에 위치하는 데이터를 암호화한 세션 키가 SOAP 메시지의 수신자의 공개키로 다시 암호화된 것을 말한다. 이는 SET[7]에서 사용된 전자봉투와 같은 개념이다.
- **Signature** : XML Digital Signature[8] 알고리즘을 이용하여 SOAP Body 데이터를 서명한 부분으로 데이터 무결성과 부인방지 기능을 제공한다.
- **Encrypted Data** : XML Encryption[9] 알고리즘을 이용하여 SOAP Body 데이터를 암호화한 부분으로 데이터의 기밀성을 제공한다.

그림 1은 Encrypted Key 메커니즘을 나타낸다.

대칭키 암호 알고리즘은 공개키 암호 알고리즘보다 암/복호화 계산 속도가 빠른데, 대칭키 암호 알고리즘인 DES[10]는 비대칭키 암호 알고리즘인 RSA[11]보다 약 1000배 정도 암/복호화 계산속도가 빠르다. 이런 계산속도 차이 때문에, 일반적으로 데이터 암/복호화 에는 대칭키 암호 알고리즘이 사용된다. 하지만 대칭키 암호 알고리즘은 키 교환 시에 키 분실에 대한 우려가 있어 키 관리에 대한 문제점이 있게 된다. 비대칭키 암호 알고리즘에서는 암호화할 때 공개키를 사용하고, 복호화할 때는 개인키를 사용하는데,

일반적으로 공개키는 인증기관에 공개하고, 개인키는 자신이 소유하게 된다. 따라서 암/복호화를 수행할 때마다 키 교환을 수행해야하는 대칭키 암호 알고리즘보다 키 관리에 대한 부담이 줄게 된다. SET[7]에서는 전자봉투라는 메커니즘을 이용해서 사용자의 중요정보를 대칭키 암호 알고리즘으로 암호화하고, 여기서 사용된 비밀키를 수신자의 공개키로 암호화하여 수신자에게 보내게 되는데, 여기서 공개키로 암호화된 비밀키를 전자봉투라고 한다. 공개키라는 봉투에 싸여진 비밀키임을 암시한다. 수신자는 암호화된 메시지와 전자봉투를 받아서, 먼저 수신자의 개인키로 전자봉투를 복호화해서 비밀키를 얻은 다음, 그 비밀키로 암호화된 메시지를 복호화하게 된다.

대칭키 암호 알고리즘의 계산속도와 비대칭키 암호 알고리즘의 안전한 키 관리의 장점을 잘 활용한 메커니즘이라고 할 수 있다.

Encrypted Key 메커니즘도 SET[7]에서의 전자봉투 메커니즘을 따랐는데, 일반적으로 메시지 내용이 긴 SOAP Body 메시지는 암/복호화 속도가 빠른 대칭키 암호 알고리즘으로 암호화해서 Encrypted Data로 만들고, 여기서 사용된 비밀키는 SOAP 메시지 수신자의 공개키로 암호화해서 일종의 전자봉투라 할 수 있는 Encrypted Key를 만들어서 SOAP Header에 위치시키게 된다.

표 1. SOAP Message Security 예

<S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"> <S:Header> <m:path xmlns:m="http://schemas.xmlsoap.org/rp/"> <m:action>http://fabrikam123.com/getQuote</m:action> <m:to>http://fabrikam123.com/stocks</m:to> <m:from>mailto:johnsmith@fabrikam123.com</m:from> <m:id>uuid:84b9f5d0-33fb-4a81-...</m:id> </m:path>	①
<wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext"	
<wsu:Timestamp wsu:Id="Timestamp-12"> <wsu:Created>2001-10-13T09:00:00Z</wsu:Created> <wsu:Expires>2001-10-13T09:00:00Z</wsu:Expires> </wsu:Timestamp>	②
<wsse:BinarySecurityToken ValueType="wsse:X509v3" EncodingType="wsse:Base64Binary"> MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i... </wsse:BinarySecurityToken>	③
<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"> <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/> <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#"> <wsse:SecurityTokenReference> <wsse:KeyIdentifier>FHejk.. </wsse:KeyIdentifier> </wsse:SecurityTokenReference> </ds:KeyInfo> <xenc:CipherData> <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0... </xenc:CipherValue> </xenc:CipherData> <xenc:ReferenceList> <xenc:DataReference URI="#enc1"/> </xenc:ReferenceList> </xenc:EncryptedKey>	④
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"> <ds:SignedInfo> <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/> <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/> <ds:Reference> <ds:Transforms> <ds:Transform Algorithm="http://...#RoutingTransform"/> <ds:Transform Algorithm="http://www.w3.org/2001/10/ xml-exc-c14n#"/> </ds:Transforms> <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/> <ds:DigestValue>LyLsF094hPi4wPU... </ds:DigestValue> </ds:Reference> </ds:SignedInfo> <ds:SignatureValue> Hp1ZkmFZ/2kQLXDJbchm5gK... </ds:SignatureValue> <ds:KeyInfo> <wsse:SecurityTokenReference> <wsse:Reference URI="#X509Token"/> </wsse:SecurityTokenReference> </ds:KeyInfo> </ds:Signature>	⑤
</wsse:Security> </S:Header> <S:Body>	
<xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" xmlns:wsu="http://schemas.xmlsoap.org/ws/2002/07/utility" Type="http://www.w3.org/2001/04/xmlenc#Element" wsu:Id="enc1"> <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#3des-cbc"/> <xenc:CipherData> <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0... </xenc:CipherValue> </xenc:CipherData> </xenc:EncryptedData>	⑥
</S:Body> </S:Envelope>	
① Routing Information ② Timestamp ③ Security Token ④ Encrypted Key ⑤ Signature ⑥ Encrypted Data	

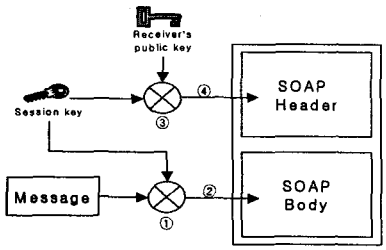

그림 1. Encrypted Key 메커니즘

SOAP 메시지를 받은 수신자는 자신의 개인키로 Encrypted Key를 복호화해서 비밀키를 얻은 다음, 이 비밀키로 EncryptedData를 복호화해서 SOAP Body 메시지를 얻게 된다. 참고로, 비밀키는 그 길이가 길지 않기 때문에 비대칭키 암호 알고리즘으로 암/복호화 하는데 많은 시간이 걸리지 않는다. 예를 들어, DES[10]의 경우 비밀키의 길이는 64비트이고, SSL[12] 에서는 40~128비트 이내의 세션키를 사용한다.

SOAP Message Security 구성 순서를 살펴보면, 그림 2는 SOAP Message Security 생성 과정을 나타내는데, 우선 SOAP Body 에 실어 보낼 메시지를 생성하고, 이 메시지 수신자에 대한 라우팅 정보를 구성하여 Routing Information을 만든다. 그 다음에 Timestamp 및 Security Token을 생성한다. 이 때 Security Token이 Signed Security Token인 경우엔 인증기관에 의뢰하여 정보를 얻어 구성한다. SOAP Body 데이터에 제 3자에게 공개해서는 안되는 정보가 있다면, 데이터를 암호화하여 기밀성을 유지한다. 암호화 과정은 XML Encryption[9] 알고리즘을 따른다. 암호화된 데이터는 Encrypted Data가 되고, 암호화한 키는 수신자의 공개키로 암호화하여 Encrypted Key를 만든다. 마지막으로 데이터에 대한 무결성 및 신원확인을 위해 디지털 서명을 행한다. 이 때 디지털 서명은 XML Digital Signature[8] 알고리즘에 의해 수행된다.

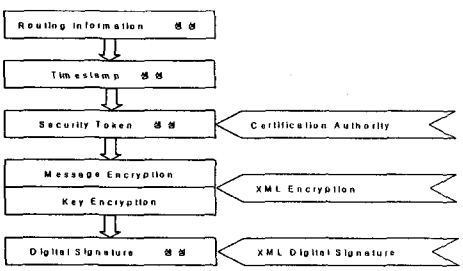

그림 2. SOAP Message Security 생성

그림 3은 수신단 에서의 SOAP Message Security 검증 과정을 나타낸다.

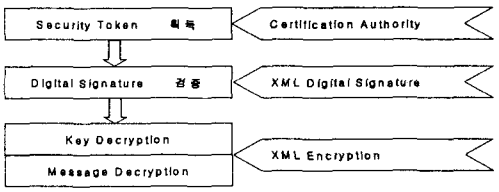

그림 3. SOAP Message Security 검증

수신자는 우선 전자서명을 검증하기 위해, SOAP 메시지 헤더나 외부의 CA로부터 인증서를 얻고, 이 인증서를 가지고 서명을 검증한다. 서명이 검증되고 나면, 암호화된 데이터를 복호화해야 하는데, 먼저 수신자의 공개키로 암호화된 키를 복호화하고, 그 다음에 복호화된 키로 암호화된 데이터를 복호화한다.

3. 문제점

이제까지 살펴본 SOAP Message Security[4]는 디지털 서명과 암호화 알고리즘을 사용하여 메시지의 신원확인 및 기밀성을 얻을 수 있고, 전자봉투 개념을 도입하여 효율적인 메시지 암호화 및 안전한 키 교환을 할 수 있는 구조로 설계되었음을 알 수 있었다.

하지만, SOAP Message Security[4]에서는 제 3자에 의해 서명문이 교체되어질 수 있는 가능성이 있다. 그림 4는 이런 문제점을 나타내고 있다.

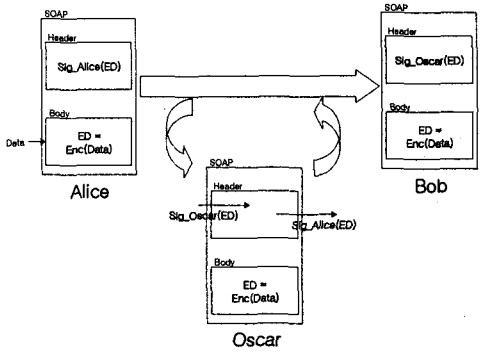

그림 4. SOAP Message Security 문제점

위 그림에서 보면, Alice는 암호화한 데이터 ED(=Enc(Data))를 서명하여 Sig_Alice(ED)를 생성하여 Bob에게 전송한다. 이 때 Oscar는 전송로 상에서 Alice가 보낸 SOAP 메시지를 얻어, Alice에 의해 서명된 부분인 Sig_Alice(ED)를 자신의 서명인 Sig_Oscar(ED)로 교체한다. Oscar는 수정된 SOAP 메시지를 다시 Bob에게 보낸다. Bob은 이런 교체 사실을 모른 채, 수신 받은 SOAP 메시지가 Alice가 아닌 Oscar에 의해 서명되었다고 생각하게 된다. Oscar는 데이터의 내용을 복호화할 필요 없이 중간에서 서명을 교체하여, 데이터 주체에 대한 신원을 위장하게 된다.

4. 해결방안

SOAP Message Security[4]에서 발견된 문제점은

데이터의 기밀성의 여부와 상관없이 외부에 서명이 노출되어 있는 상황에서는 제 3자에 의해 서명이 교체될 수 있다는 것이다. 이러한 서명교체를 막기 위해서, 서명부분을 암호화하여 전송한다. 제 3자는 암호화된 서명을 비밀키가 없인 쉽게 볼 수 없으므로, 서명 위조할 수 없게 된다. 수신측 에서는 암호화된 서명을 복호화한 다음 서명검증을 행하게 된다. 그림 5는 개선된 SOAP Message Security[4] 생성과정을 나타내는데, 메시지를 암호화한 비밀키를 이용하여 서명부분을 암호화한다. 그런 다음 수신자의 공개키로 비밀키를 암호화한다.

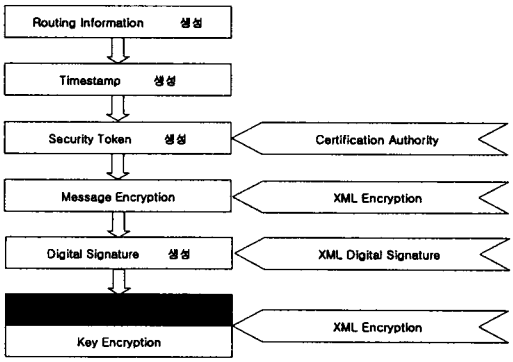

그림 5. 개선된 SOAP Message Security 생성

그림 6은 개선된 검증 과정을 나타내는데, 암호화한 서명을 수신자의 개인키로 복호화한 다음 서명검증을 하게 된다.

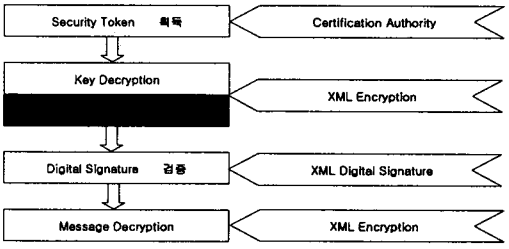

그림 6. 개선된 SOAP Message Security 검증

5. 결론

본 고에서는 최근 이 기종 시스템 및 애플리케이션의 상호호환을 위해 채택되고 있는
웹서비스 보안에 대해 살펴보았는데, 웹서비스 보안은 새로운 보안 메커니즘을 소개하기보다 기존의 보안 모델이나 보안 알고리즘을 수용하는 형태로 구성되었음을 알 수 있었다. 웹서비스 보안은 SOAP 메시지에 대한 보안을 그 목적으로 하고 있는데, 여기에는 크게 두 가지 기술, 즉, XML Digital Signature[8] 와 XML Encryption[9]이 사용되었다.

SOAP 메시지의 무결성과 신원확인을 위해 디지털 서명이 사용되었는데, 이는 XML Digital Signature[8] 알고리즘에 따라 수행되었고, SOAP 메시지에서 기밀성을 유지해야할 부분은 암호화하였는데, XML Encryption[9] 알고리즘이 적용되었다. 거기다 전자봉투 개념을 도입하여, 메시지를 암호화한 비밀키를 안전하게 수신자에게 전달할 수 있게 하였다.

하지만, 서명 부분이 노출되어 제 3자에 의해 서명 위조의 가능성이 있게 되었다. 이에 대한 해결책으로 노출된 서명 부분을 암호화하여 제 3자가 쉽게 서명에 접근할 수 없게 하였다.

웹서비스를 통한 시스템 통합 및 서비스 표준화의 요구가 계속 증가하고 있고, 이로 인한 여러 분야의 웹서비스화는 더욱 가속될 것이다. 하지만, 웹 서비스의 개방성으로 인해 보안에 대한 우려도 커지고 있다. 현재 웹서비스 보안에 대한 작업들이 활발히 이루어지고 있지만, 여전히 많은 문제점들을 가지고 있다. 향후에는 SOAP 메시지 자체에 대한 보안은 물론, 접근제어나 정책에 관련된 부분도 많은 연구가 이루어져야 할 것이다.

참고문헌

[1] W3C, "SOAP Version 1.2 Part0:Primer", Recommendation 24 June 2003, http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[2] OASIS, "UDDI Version 3.0", 19 July 2002, http://uddi.org/pubs/uddi-v3.00-published-20020719.htm

[3] W3C, "Web Services Description Language(WSDL) Version 1.2 Part 1: Core Language", Working Draft 11 June 2003, http://www.w3.org/TR/2003/WD-wsdl12-20030611

[4] OASIS, "Web Services Security : SOAP Message Security", Working Draft 21 April 2003, http://www.oasis-open.org/committees/documents.php

[5] IETF, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", Jan 1999, http://www.ietf.org/rfc/rfc2459.txt

[6] IETF, "The Kerberos Network Authentication Service(V5)", Sep 1993, http://www.ietf.org/rfc/rfc1510.txt

[7] Visa and MasterCard, "Secure Electronic Transaction", 1997

[8] W3C, "XML-Signature Syntax and Processing", Recommendation Feb 2002, http://www.w3.org/TR/2002/REC-xmldsig-core-20020212

[9] W3C, "XML Encryption Syntax and Processing", Candidate Recommendation March 2002, http://www.w3.org/TR/2002/CR-xmlenc-core-20020304

[10] NBS FIPS 46, "Data Encryption Standard", National Bureau of Standards, U.S. Department of Commerce, Jan 1977

[11] R.L.Rivest, A.Shamir,and L.Adleman, "A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (1978), no.2, 120-126

[12] A.Freier,P.Karlton,and P.Kocher, "The SSL Protocol Version 3.0", Internet Draft, work in progress, 1996