Web Servisleri

Kılavuzun bu bölümünde web servisi geliştiricilerinin karşılaştığı genel konular ve bu genel konuları işaret eden metotlar inceleniyor. Yer sıkıntısından dolayı, bu çerçevedeki bütün konulara, bakılsa herbiri ayrı birer kitap olur, ayrıntılı olarak bakılamıyor. Bunun yerine, okuyucuya uygun kullanım biçimleri gösterilmeye çalışılıyor ve yol üzerindeki potansiyel engeller hakkında okuyucu uyarılıyor.

Web servisleri basında çok fazla yazıldı ve bunun sonucunda da gerçekten ne oldukları hakkında büyük karışıklıklar ve yanlış anlaşılmalar meydana geldi. Bazıları web servislerini, webin kendisinden sonra ortaya çıkan en büyük teknolojik gelişme olarak görürken; başka bir grup, biraz daha şüpheci yaklaşarak biraz evrimleşmiş web uygulamaları olarak gördü. Her iki durumda da, web uygulaması güvenliğini ilgilendiren konular web servisleri için de geçerlidir.

En basit seviyede, web servisleri, sunum şekli açısından farklı olan, özelleşmiş web uygulamaları olarak görülebilir. Web uygulamaları tipik olarak HTML-tabanlı iken, web servisleri XML-tabanlıdır. B2C hareketlerinin interaktif kullanıcıları normalde web uygulamlarına erişirken, web servisleri, SOA modeli kullanarak B2B zincirleri oluşrutururlar, ve bu web uygulamaları için yapı taşları olarak kullanılırlar. Web servisleri genel olarak, programlardan çağrılabilir, kamuya açık fonksiyonel bir arabirime sahipken; web uygulamasları daha zengin özelliklere sahip olma ve çoğu zaman içeriğe bağlı çalışma eğilimindedirler.

Web Servislerinin Güvenliği

Web servisleri, diğer dağıtık uygulamalar gibi farklı seviyelerde korunur:

- SOAP mesajları, kablolardan, gizli olarak ve üzerinde değişiklik yapılamadan dağıtılmalıdır.
- Sunucu kimin konuştuğu ve konuşan istemcilerinin hakları konusularında emin olmalıdır.
- İstemciler doğru sunucu ile konuştuklarından ve bir balık ağına yakalanmadıklarından(phishing) emin olmalıdırlar.
- Sistem mesajlarının kayıtları olaylar zincirinin güvenilebilir bir şekilde tekrar edilebilmesi ve kimliği doğrulanmış ziyaretçileri takip edebilecek kadar yeterli bilgiye sahip olmalıdırlar.

Aynı şekilde, ileriki bölümlerde incelenen, ve dağıtık uygulamaların güvenliğine yönelik yüksek seviye yaklaşımları, uygulama ayrıntılarındaki ufak değişiklerle web servisleri için de geçerlidir.

Web servisleri geliştiriciler için güzel haber ise bu yaklaşımlar altyapı seviyesinde görevler gerektirir, dolayısıyla, teorik olarak, bu konularla sistem yöneticilerinin ilgilenmesi gerekir. Fakat, bu bölümün ileriki kısımlarında bahsedilen bazı sebeplerden

dolayı, web servisi geliştiricileri en azından bu risklerden haberdar olmalı ve çoğu zaman koruma işlemine bizzat katılmalıdırlar.

İletişim Güvenliği

Ortak olarak benimsenen ve daha yaygın bir şekilde uygulanan bir yargı vardır: "her türlü iltişimimizi korumak için SSL kullanıyoruz, ve biz güvendeyiz." Aynı zamanda, birçok makale "kanal güvenliğine karşı parçacık(token) güvenliği" konusu üzerinde yayınlanmaktadır ki bu görüşleri burada tekrarlamak anlmasızdır. Bu yüzden aşağıdaki listede kanal güvenliğini yalnız bir şekilde kullanırken ortaya çıkabilecek gizli tehlikelerin özeti bulunmaktadır:

- Sadece "noktadan noktaya" güvenlik sağlar.

Birden fazla noktadan oluşan herhangi bir iletişim, bu yol üzerindeki herbir iletişimde bulunan nokta arasında ayrı birer kanal (ve güvenlik) kurulmasını gerektirir. Ayrıca güvenin geçişliliği gibi hemen göze çarpmayan konular da vardır, mesela {A,B} ve {B,C} nokta ikilileri arasındaki güven, otomatik olarak {A,C} arasındaki güveni gerektirmez.

- Bellek meselesi

Mesajlar sunucu tarafında alındıktan sonra (kastedilen alıcı olmasa da), en azından kısa süreliğine de olsa düz yazı biçiminde bulunur. Taşınan bilginin aradaki noktalarda kayıt edilmesi, veya kayıt dosyalarındaki hedef sunucu (herkes tarafından görüntülenebilir) ve yerel önbellek problemi daha da kötü hale getirir.

- Birlikte çalışılabilirliğin eksikliği

SSL taşıma korunmasında standart bir mekanizma sağlasa da, uygulamalar kimlik bilgilerinin taşınmasında, güvenli kanaldan taşınan verinin bütünlük, gizlilik ve tazelik gibi hususlarının sağlanmasında yüksek derecede özel mekanizmalar kullanır. Farklı bir sunucu kullanmak, işlev itibarıyle değişmediği halde aynı bilgileri farklı formatlarda kabul ettiği için istemcilerin değişmesine ve B2B servis zincirlerinin oluşamamasına sebep olur.

Standart-tabanlı parçacık (token) korunumu, birçok durumda, mesaj-bazlı Web Servisi SOAP iletişim modeline karşı ileri bir alternatiftir.

Bu demektir ki, bugünkü Web Servislerinin çoğu halen kanal güvenlik mekanizmalarının bir versiyonu ile korunmaktadır ki bu tek başına basit bir iç uygulama için yeterlidir. Fakat, bu yaklaşımın sınırları iyice anlaşılmalıdır ve tasarım sırasında gerekli düzenlemeler yapılmalıdır, kanal güvenliğinin mi yoksa parçacık (token) güvenliğinin mi, yoksa bu ikisinin bir karışımının mı bu özel durum için gerekliliği düşünülmelidir.

Kimlik Bilgilerinin Taşınması

Kimlik bilgilerinin karşılıklı değiştirilmesi ve Web Servislerinde kimlik doğrulamanın yapılabilmesi için, geliştiriciler aşağıdaki konuları hedef almalıdır.

Öncelikle, SOAP mesajları XML-tabanlı olduğundan, değiştirilen bütün bilgiler yazı biçiminde olmalıdır. Bu, kullanıcı adı/şifre gibi kimlik bilgilerinde sorun yaratmazken, yürütülebilir dosyalar(mesela X.509 sertifikaları veya Kerberos parçacıkları-tokens-) gönderilmeden ve alınmadan önce Base64 kodlama sistemi ile kodlanmalı ve çözülmelidir.

İkinci olarak da, kimlik bilgilerinin taşınması bu bilgilerin kaybedilmesi riskini hep taşır- iletim sırasında dinlenebilir veya sunucu kayıtları incelenebilir. Bu yüzden parola veya gizli anahtar gibi bilgilerin şifrelenmiş bir şekilde gönderilmesi veya hiçbir zaman düz yazı şeklinde gönderilmemesi gerekir. Hassas kimlik bilgilerinin yollanmasındaki genel yöntem, kriptografik **hash** ve/veya imzalama yöntemidir.

Mesaj Tazeliğini Sağlamak

"Yeniden oynatma" saldırısında kullanıldığında geçerli bir mesaj bile tehlike arz edebilir- mesela, sunucuya istenilen işlemin tekrar yaptırılması için birkaç kez gönderilmesi gibi. Mesaj değiştirilmeye karşı yeteri kadar sağlam olsa bile, saldırı için kullanılan mesajın kendisi olduğu için (bkz. XML Enjeksiyonu bölümü.), bu saldırı mesajın tamamının ele geçirilmesi ile uygulanabilir.

Yeniden yollanan mesajlardan korunmak için kullanılan genel yöntemler, ya mesajda benzersiz tanımlayıcı kullanmak ve işlenenlerin kayıtlarını tutmak, veya nispeten daha kısa geçerlilik süresi kullanmaktır. Web Servisleri dünyasında ise, mesajın hazırlandığı zamanın bilgisi genellikle mesaja eklenerek iletilir, ve bunun yanında fazladan bilgi de eklenebilir, örneğin mesajın sona erme zamanı, veya belirli durumlar gibi.

İkinci çözümde, uygulaması daha kolay olmasına rağmen, senkronize bir saat gerektirir ve "sunucu zamanında kaymaya" çok duyarlıdır ki; sunucu ve istemci saatlerinde kayma çok olur ve zamanlı mesaj iletimini engeller; günümüz bilgisayarlarında önemli bir problem arz etmediği halde. Daha büyük problem, sunucu tarafındaki mesaj işleme sırasında, sunucu meşgul olduğundan veya cevap veremediğinden dolayı beklemekten süresi dolan mesajlarda yaşanır.

Mesaj Bütünlüğünün Sağlanması

Bir mesaj bir web servisi tarafından alındığında, servis her zaman şu iki soruyu sormalıdır: "gönderene güvenmeli miyim?," "bu mesajı o mu gönderdi?". Gönderene güvenin bir şekilde sağlandığını kabul edersek, sunucu elindeki mesajın gerçekten gönderen kişi tarafından oluşturulduğundan ve yolda (bilinçli veya bilinçsiz) değiştirilmediğinden emin olmalıdır. Bu SOAP mesajının teknik özelliklerini, örneğin

zaman etiketini, veya içeriğini, örneğin bankadan çekilecek olan miktar gibi, etkileyebilir. Açıkça görülüyor ki, iki değişiklik de sunucu tarafından farkedilmelidir.

İletişim protokollerinde, paket bütünlüğünü teyit eden parçacıklar bulunur(checksum). Fakat bu yeterli olmayacaktır, halka açık Web Servisleri dünyasında, bu toplam parçacıkları kolayca değiştirilebilir ve güvenle sahibine ulaşımı sağlayamaz. Gerekli ilişki HMAC kullanarak kurulabilir, veya hash'lerin kriptografik imzalarla veya gizli anahtar şifreleriyle (şifrelerin gizli olduğunu kabul edersek) birleştirilmesi ile en ufak bir değişikliğin farkına varılabilir.

Mesaj Gizliliğinin Korunması

Çoğu zaman, bütünlüğü sağlamak yeterli değildir, bir çok durumda saklanan veya taşınan bilginin gizliliğinin de sağlanması gerekebilir. Bu işlenen mesajın tamamına uygulanabileceği gibi, belirli kısımlarına da uygulanabilir ve her iki durumda da içeriği saklamak için bir çeşit şifreleme gereklidir. Normalde, büyük miktardaki verileri şifrelemek için simetrik şifreleme algoritmaları kullanılır, çünkü asimetrik şifrelemeden daha hızlıdırlar. Asimetrik şifreleme de simetrik oturum anahtarlarını korumak için kullanılır, bu anahtarlar bir çok uygulamada bir konuşma süresince geçerlidirler ve konuşma sonunda atılırlar.

Şifre kullanma, konuşacak birimlerin sertifika ve anahtar doğrulama, güvenebilecekleri anahtarları bilme, iletişim için kullanacakları anahtarları bilme gibi işlerin halledilebilmesi için geniş kapsamlı bir kurulum çalışması gerektirir.

Birçok durumda, bütünlük ve gizliliğin sağlanması için şifreleme imza ile birleştirilir. Normalde, imzalayan anahtarlar şifreleyenlerden farklıdır, çünkü anahtarların hayat süreleri farklıdır. İmzalayan anahtarlar, sahipleri ile ilişkilendirilmiş kalıcı anahtarlardır, fakat şifreleyen anahtarlar mesaj alış-verişinden sonra geçersiz hale getirilebilir. Bir diğer sebep de iş yükümlülüklerinin ayrılması olabilir, imzalayan otorite (ve karşılık gelen anahtar) bir bölüme veya kişiye aitken, şifreleme anahtarları BT bölümünden sorumlu elemanların kontrol ettiği sunucular tarafından üretilir.

Erişim Kontrolü

Mesaj alındıktan ve başarıyla geçerli kılındıktan sonra, sunucu şunlara karar vermelidir:

- İşlemi talep edeni tanıyor muyum?(Tanımlama)
- Gönderenin, iddia ettiği kişi olduğuna güveniyor muyum?(Kimlik Doğrulama)
- Gönderene bu işlemi yapması için izin veriyor muyum?(Yetkilendirme)

Bu aşamada Web Servislerine özel çok fazla aktivite yok-sadece kimlik doğrulama sırasındaki kimlik bilgilerinin iletimi için yeni birkaç yol. Çoğu zaman, yetkilendirme görevleri tamamen Web Servislerinin dışında halledilir ve bütün etki alanını koruyan politika sunucusunda yapılır.

Burada önemli bir sorun daha-geleneksel HTTP güvenlik duvarları Web Servislerine yapılan saldırıları engellemeye yardımcı olmaz. Organizasyon, uygulama-seviyesinde web sunucusundaki trafiği analiz edebilen ve SOAP mesajlarını hedeflerine geçirip geçirmemekte akıllı kararlar veren XML| SOAP güvenlik duvarlarına ihtiyaç duyacaktır. Okuyucunun, bu önemli konuda, sadece bir bölüm içinde incelenmesi zor olduğundan, başka kitaplara ve yayınlara başvurması gerekebilir.

Denetleme

Tipik olarak deneteleme için gerekli olan genel görevlerden biri, belirli bir probleme yol açan olaylar zincirini tekrar oluşturmaktır. Normalde, bu sunucu kayıtlarının güvenli bir yerde kaydedilmesi ve sadece BT yöneticilerine ve sistem denetçilerine bu kayıtlardan "denetleme takibi" yapılmasına izin verilmesiyle başarılabilir. Web servisleri, bu duruma bir istisna değildir, ve diğer Web uygulamalarının genel yaklaşımlarını takip eder.

Bir başka denetleme hedefi, inkar edilememezliği sağlamaktır, yani mesajın teyit edilebilecek bir şekilde gönderenine kadar izi sürülebilir. Standart yasal uygulamalara göre, elektronik dökümanlar bir çeşit elektronik imza bulundurmalıdırlar, fakat bu imzanın tanımı çok geniş bir şekilde yapılmış olduğundan pratikte her manaya gelebilir - ve birçok durumda isminizi ve doğum tarihinizi girmeniz bile elektronik imza tanımına uyacaktır. Web servisleri açısından bu derecedeki bir koruma yetersiz ve kolay kırılabilirdir. Standart uygulamada bir dökümanın yasal olarak bağlayıcı olabilmesi için kriptografik dijital imza bulundurması gerekir –ve eğer böyle bir imzaya sahip bir döküman denetleme kayıtlarında bulunursa; bu, güvenilir bir şekilde imzalayan anahtarın sahibine ulaştırabilir.

Web Servisleri Güvenlik Hiyerarşisi

Teknik açıdan bakılırsa, Web Servisleri XML-tabanlı bir gramer ile tanımlanan ve Web Servisleri Tanımlama Dili (Web Services Description Language-WSDL, bkz. http://www.w3.org/TR/2005/WD-wsdl20-20050510) adı verilen, oldukça basit ve esnek bir yapıya sahip XML-tabanlı iletişim yöntemidir. WSDL, XML şeması şeklinde ifade edilen, mesajlar ve işlemler gibi servis arayüzlerinin, kullanılan medya biçimi ile arasındaki bağı sağlar. Herhangi bir şekilde gereklilik olmamasına rağmen, tercih edilen biçim HTTP üzerinden SOAP şeklidedir. Bu, Web Servisleri arayüzleri, HTTP rotokolü üzerinden taşınan gelen ve giden SOAP mesajları şeklinde tanımlanır.

Standartlar Komitesi

Standartları incelemeden önce, bu standartları geliştiren ve destekleyen organizasyonlara kısaca göz atmakta fayda var. Bu alanda çalışan çok az miktarda grup ve konsorsiyum bulunur. Bunların en önelimleri:

W3C(bkz. http://www.w3c.org), en çok bilinen, Web ile ilgili birçok standardı elinde bulunduran ve Working Group biçiminde geliştiren gruptur. Bu bölümle ilgilendiği konularsa, XML Şemaları, SOAP, XML-dsig, XML-enc ve WSDL standartları(W3C deyimiyle, tavsiyeler).

OASIS (bkz. http://www.oasis-open.org) daha çok, güvenlikle ilgili veya değil, Web Servislerine özel standartlarla ilgilenir. Komite bazlı çalışır ve geliştirilecek standartlar için Teknik Komiteyi oluşturur. Bu konuşma ile ilgili olarak da, OASIS WS-Security(Web Servisleri Güvenliği) ve SAML standartlarını elinde bulundurur.

Web Services Interoperability grubu (WS-I, bkz http://www.ws-i.org/) birlikte çalışabilir web servisleri için genel bir yapı oluşturmak için kuruldu. İşleri daha çok, başka yaygın şekilde kabul edilen standardı alıp, Web-Servis uygulamalarına uyumları için gereken profilleri ve gereksinimler geliştirmedir. Özel olarak, BSP(Basic Security Profile-Temel Güvenlik Profili)'leri OASIS'in WS-Security (WS-Güvenlik) standartlarına dayanır ve birlikte çalışabilirlik iddia eden Web Servislerindeki gereken ve isteğe bağlı olan güvenlik özelliklerini belirler.

Liberty Alliance(LA, bkz. http://projectliderty.org) konsorsiyumu, birlikte çalışabilir bir Identity Federation iskeleti geliştirmek ve ilerletmek için kurlumuştur. Bu iskelet kesin bir şekilde Web Servislerine özel olmasa da, genelde, bu konu için önemlidir çünkü OASIS'in geliştirdiği SAML standartları ile yakından ilişkilidir.

Yukarıda listelenen organizasyonların yanında, Web Servisleri güvenlik aktivitelerini ilerleten, kalıcı kurulan ama az yaşayan, başka endüstri kuruluşları da vardır. Bunlar genelde yazılım endüstrisinin önde gelen, örneğin Microsoft, IBM, Verisign, BEA, Sun, ve diğerleri gibi, ve belirli konularda çalışmak üzere katılan kuruluşlardır. Bu çoklu aktivitelerin sonuçları, belirli bir olgunluğa ulaştığında, yeni endüstri standardı olarak standardlar komitesine sunulur.

SOAP

Simple Object Access Protocol(SOAP, bkz. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/) servisler arasında yapılandırılmış bilgi alışverişini sağlayan XML-tabanlı bir iskelet sağlar. Bu bilgi, Header(Başlık) ve Body(Gövde) şeklide biçimlenir, teorik olarak birçok iletişim protokolü üzerinden taşınabilir, fakat sadece HTTP-bağı resmi olarak tanımlanmıştır ve bugün kullanımdadır. SOAP Remote Procedure Call(RPC, Uzak Prosedür Çağrısı) stilinde etkileşim sağlar, ve uzak fonksiyon çağrıları ve Döküman-stili iletişim gibi, ve mesaj içeriği sadece WSDL'deki XML Şema tanımına göre hazırlanır. Çağrı sonuçları isteğe bağlı olarak cevap mesajı ile birlikte döndürülür, veya hata mesajı döndürülebilir ki bu geleneksel programlama dillerindekine kabaca benzerdir.

SOAP protokolü, iletişim iskeletini oluştururken, mesaj alışverişini korumak açısından hiçbir yardım sağlamaz- bu iletişim ya güvenli bir kanaldan yapılmalıdır veya ileride bu bölümde anlatılacağı gibi koruma mekanizmaları kullanılmalıdır.

XML Güvenlik Spesifikasyonları (XML-dsig & Şifreleme)

XML İmzası (XML-dsig, bkz. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/), ve XML Şifreleme (XML-enc, bkz. http://www.w3.org/TR/2002/REC-xmlenccore-20021210/) düz XML dökümanlarına kriptografik koruma kazandırır. Bu spesifikasyonlar bütünlük, mesajın ve imzalayıcısının doğrulanması gibi özelliklerin eklenmesi ile birlikte bütün XML dökümanının veya içinden bazı elemanların şifrelenmesi/çözülmesi konusunda destek sağlar.

Bu standartların gerçek değeri, işlenen veriye (XML dökümanına göre hem dahili hem de harici), ya da gizli anahtarlara veya anahtar ikililerine, ulaşmaya yarayan ve de imzalama/şifreleme işlemlerinin sonucunu XML şeklinde, orjinal dökümana ekleyerek/yer değiştirerek sunmak için geliştirilen yüksek esneklikteki iskeletten alır.

Fakat, kendi başlarına, XML-dsig ve XML-enc, SOAP-tabanlı Web Servis etkileşimlerini koruma problemini çözmez; çünkü istemci ve servis önce işlemlerin sırası, imza için nereye bakılacağı, kriptografik parçacıkların nasıl alınacağı, mesajın hangi parçacıkları imzalanmalı, şifrelenmeli, geçerli bir mesajın uzunluğu, vs. gibi konularda anlaşmalıdırlar. Bu konular, takip eden bölümlerin incelediği, daha üst seviye spesifikasyonlarla çözümlenmiştir.

Güvenlik Spesifikasyonları

Yukarıdaki standartlara ek olarak, Web Servisleri işlemlerinin değişik boyutlarıyla alakalı geniş çaplı güvenlik spesifikasyonları geliştirilmektedir.

Bunlardan biri, kimlik, özellik ve kimlik doğrulama bilgilerinin katılan servisler arasında güvenli ve birlikte çalışabilir bir şekilde nasıl alınıp verileceğini tanımlayan SAML'dir.

Microsoft ve IBM önderliğindeki geniş bir konsorsiyum, Verisign, RSA Security (Güvenlik), vs gibi firmalardan gelen girdilerle, ve diğer katılımcılarla, birlikte "Web Servisleri Haritası" olarak bilinen bir spesifikasyonlar ailesini geliştirdi. Bu spesifikasyonların temeli, WS-Security (WS-Güvenlik) tarafından OASIS'e teslim edilmiş ve 2004 yılında OASIS standardı olmuştur. Bu ailenin diğer önemli spesifikasyonları geliştirme sürecinin değişik aşamalarında halen kullanılmakta olup teslim planları açıklanmamasına rağmen, güvenlik politikalarında (WS-Policy), güven konularında, güvenlik parçacık değişiminde (WS-Trust), güvenli iletişim için altyapı oluşturulmasında(WS-SecureConversation) kapsanmaktadır. Bu ailedeki sepesifikasyonlardan birisi, WS-Federation, doğrudan LA konsorsiyumunda yapılan çalışmayla yarışır ve Windows'a Wista sürümüyle birlikte eklenmesi gerekirken, büyük ölçüde geciktiği için ve şu anda endüstri tarafından desteklenmesiği için, mevcut durumda geleceği belirsizdir.

WS-Security(WS-Güvenlik) Standardı

WS-Security spesifikasyonları (WSS), aslında "Harita"'nın bir parçacığı olarak Microsoft, IBM ve Verisign tarafından geliştirilmiştir, ve daha sonra Web Servisleri Mimarisi (Web Services Architecture, WSA) olarak adlandırılmıştır. WSS bu alandaki bütün diğer spesifikasyonlar için temel oluşturmuş, mesaj-tabanlı güvenlik alışverişi için altyapıyı hazırlamıştır. Birlikte çalışabilir Web Servisleri oluşturmadaki öneminden dolayı, OASIS e verilmiş ve gerekli komite işlemlerinden sonra, resmi olarak kabul edilmiş bir standart olmuştur. Şu anki versiyonu 1.0'dır ve 1.1 üzerindeki çalışmanın 2005 yılının ikinci yarısında bitmesi hedeflenmektedir.

Standardın Organizasyonu

WSS standardının kendisi, birçok ayrıntıyı profil dökümanlarına bırakarak, bir çok temel güvenlik alanıyla ilgilenir. Standart tarafından geniş bir şekilde tanımlanan temel alanlar:

- SOAP zarflarına güvenlik başlıkları(WSSE Başlıkları) eklemenin yolları
- Mesaja güvenlik parçacıklarının ve kimlik bilgilerinin eklenmesi
- Zaman etiketinin eklenmesi
- Mesajın imzalanması
- Mesajın şifrelenmesi
- Genişletilebilirlik

WS-Security standardının esnekliği, genişletilebilir olmasından gelir, bu şekilde yeni geliştirilen güvenlik parçacıklarına ve protokollerine uyum sağlayabilir. Bu esneklik, WSS iskeletine yeni güvenlik parçacıkları eklemek için tanımlanan fazla profiller sayesinde kazanılır. Standartların imzalama ve şifreleme kısımları önemli değişiklikler gerektirmezken (sadece altta bulunan XML-dsig ve XML-enc güncellendiğinde), WSS mesajlarında kullanılan parçacık türleri ve bu türlerin mesaja eklenme biçimleri önemli ölçüde değişebilir. Yüksek seviyedeki WSS standartları WSS başlıklarına eklenebilen 3 çeşit güvenlik parçacığı tanımlar: kullanıcı adı/şifre, ikili(binary) ve XML parçacıkları. Bu türlerin herbiri bir (veya birden fazla) profil dökümanında detaylı bir şekilde (güvenlik parçacığının özellikleri, elemanları) tanımlanır.

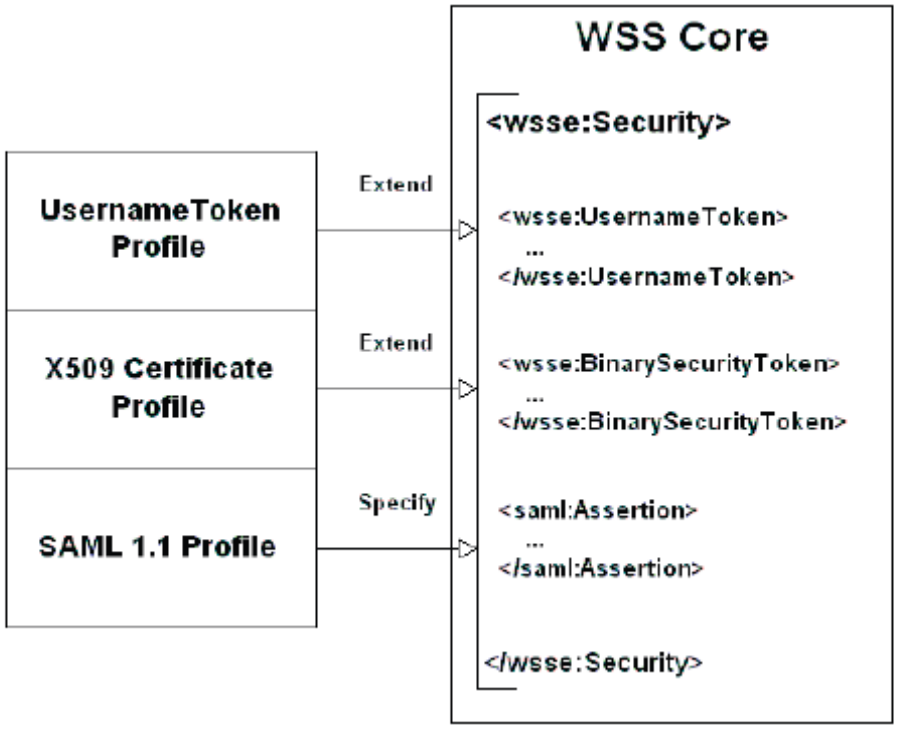

Figure 1 WSS Spesifikasyon hiyerarşisi

Amaç

WSS standardının temel amacı mesaj-seviyesindeki iletişimi koruyan araçlar sağlamaktır. Her mesajda, önemli mesaj özelliklerini, tanımlanmışlık, bütünlük ve tazelik gibi, teyit edecek kadar ve şifrelenmiş mesaj parçacıklarını deşifre etmeye yarayacak kadar güvenlik bilgisi taşır. Bu, geleneksel kanal güvenliğinin tam karşıtı bir korumadır, geleneksel kanal güvenliğinde iletişim öncesinde kararlaştırılan güvenlik yapısı bütün iletişimde kullanılır. "Harita"'da, WS-SecureConversation gibi standartların uygulanmasıyla bu tür servislerin sağlanması beklenir.

Başından beri, WSS standardı yüksek kademedeki protokollerde güvenli veri dağıtımını sağlamak üzere hazırlanmış mesaj-seviyesinde bir araç kiti olarak görülür.WS-Policy, WS-Trust, Librty Alliance gibi standartlar tabanlı bu protokollerde, geçiş kontrolü politikalarını(access control policies) uygulamak üzere iletilen sembollere(token) güvenilir. Buna rağmen, yalnız alındığında, WSS standardı herhangi belirli bir güvenlik politikasının kullanılmasını zorlamaz, ve bu standardın ad-hoc modundaki uygulamalarında gizli açıklıklar bulunur ki, bunlara da bu bölümün ilerleyen kısımlarında bahsedeceğiz.

WS-Security Yapı Taşları

WSS standardı aslında birkaç dokümandan oluşur- güvenlik başlıklarının SOAP zarflarına nasıl ekleneceğini açıklayan ve geçerli bir güvenlik başlığında bulunması gereken bütün yüksek seviye bloklarını açıklayan temel bir doküman bunlardan biridir. Profil dokümanlarının iki görevi vardır: ilgilendikleri sembol (token) tiplerinin geniş

açıklaması, ekstra özellik ve elemanlar sağlayarak; diğer bir görevi de temel tanımlamaların dışında kalan ilişkileri açıklamak, örneğin eklenti kullanımı gibi.

Temel WSS 1.0 dokümanında, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0, birçok türdeki güvenlik sembolü(token); manası, ulaşma yolları, zaman etiketleri, güvenlik başlığına XML-dsig ve XML-enc uygulama yolları açıklanır- XML Dsig bölümünde genel yapısı hakkında daha fazla bilgi bulabilirsiniz..

İlişkili dokümanlar:

- Kullanıcı adı profili, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0, temel kullanıcı-adı sembolüne birçok değişik parola-alakalı genişletmeler getirir.
- X.509 sertifika sembolü profili, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0, X.509 sertifikalarının BinarySecurityToken içerisine nasıl geçileceğini açıklar.
- SAML sembol profili, http://docs.oasis-open.org/wss/2004/01/oasis-wss-samltoken-profile-1.0.pdf, WSS başlıklarına nasıl XML-tabanlı SAML sembollerinin yerleştirilebileceğini açıklar.

Veri Nasıl Taşınıyor

WSS güvenlik tanımlamaları, 2 ayrı veri türüyle ilgilenir; güvenlik bilgisi, güvenlik sembolü, imza vs. taşıyan; ve mesaj verisi, SOAP mesajında taşınan ve geriye kalan herşey. XML-tabanlı bir standart olarak, WSS, XML-elemanları halinde gruplanmış yazı içeren verilerle çalışır. Binary haldeki herhangi bir veri, base64 encoding/decoding denilen özel bir değişmden geçirilir, örneğin imza verisi, veya kerberos sembolü gibi. Bu değişim binaryden ASCII biçimine doğrudan bir çevirmedir. Aşağıda, binary verinin nasıl göründüne dair bir örnek bulunuyor:

```
cCBDQTAeFw0wNDA1MTIxNjIzMDRaFw0wNTA1MTIxNjIzMDRaMG8xCz
```

Binary elemanı encode ettikten sonra, algoritmanın tanımlamasını yapan bir eleman, XML taşıyan veriye eklenir, ve böylelikle alıcı taraf okumak için doğru çözücüyü kullanabilir. Bu tanımlayıcılar WSS dokümanında bulunmaktadır.

Güvenlik Başlığının Yapısı

Bir mesajdaki güvenlik başlığı mesajı saran zarf gibi düşünülebilir, mektubu mühürler ve korur, fakat içeriği ile ilgilenmez. Bu ilgisizlik diğer yönde de geçerlidir, yani mektup (SOAP mesajı) zarfı (WSS başlığı) hakkında birşey bilmemelidir ve ilgilenmemelidir de, çünkü bu bilgiler farklı kişileri (veya uygulamaları) hedeflerler.

Bir SOAP başlığının, farklı aktörleri (SOAP 1.1) veya farklı rolleri (SOAP 1.2) hedefledikçe, birden çok güvenlik başlık bilgisi taşımasında bir sakınca yoktur. İçerikleri de birbirlerine atıfta bulunabilir, fakat bu atıflar doğru deşifre/teyit sırasını bulma sırasında karmaşık problemler yaratabilir, ve genel olarak kaçınılmalıdır. WSS güvenlik

başlığının kendisi dağınık ve güvensiz bir yapıya sahiptir, çünkü dokümanın kendisi herhangi bir elemanın bulunmasını gerektirmez, sonuç olarak boş bir mesajın minimum başlığı şu şekilde görünür:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
      <soap:Header>
            <wsse:Security xmlns:wsse="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wsswssecurity-
utility-1.0.xsd" soap:mustUnderstand="1">
            </wsse:Security>
      </soap:Header>
      <soap:Body>
      </soap:Body>
</soap:Envelope>
```

Buna rağmen, kullanışlı olması için, bazı bilgileri taşıması gerekir, ve bu bilgiler mesajı korumada yardımcı olur. Bu bir veya birden fazla güvenlik sembolü, XML içermek manasına gelir, mesela XML imzası ve XML şifreleme elemanları, tabi mesaj imzalanıyorsa ve/veya şifreleniyorsa. Böylece, tipik bir başlık daha çok şu şekilde olur:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
   <soap:Header>
      <wsse:Security xmlns="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsse="http://docs.oasisopen.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wsswssecurity-
utility-1.0.xsd" soap:mustUnderstand="1">
         <wsse:BinarySecurityToken EncodingType="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-
token-profile-1.0#X509v3" wsu:Id="aXhOJ5">MIICtzCCAi...
         </wsse:BinarySecurityToken>
         <xenc:EncryptedKey
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
            <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
         <dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
            <wsse:SecurityTokenReference>
                  <wsse:Reference URI="#aXhOJ5"
ValueType="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
            </wsse:SecurityTokenReference>
         </dsig:KeyInfo>
         <xenc:CipherData>
            <xenc:CipherValue>Nb0Mf...</xenc:CipherValue>
            </xenc:CipherData>
         <xenc:ReferenceList>
         <xenc:DataReference URI="#aDNa2iD"/>
         </xenc:ReferenceList>
      </xenc:EncryptedKey>
```

```
      <wsse:SecurityTokenReference wsu:Id="aZG0sG">
         <wsse:KeyIdentifier ValueType="http://docs.oasisopen.
org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-
1.0#SAMLAssertionID"
wsu:Id="a2tv1Uz"> 1106844369755</wsse:KeyIdentifier>
         </wsse:SecurityTokenReference>
      <saml:Assertion AssertionID="1106844369755" IssueInstant="2005-
01-
27T16:46:09.755Z" Issuer="www.my.com" MajorVersion="1" MinorVersion="1"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
...
      </saml:Assertion>
      <wsu:Timestamp wsu:Id="afc6fbe-a7d8-fbf3-9ac4-f884f435a9c1">
      <wsu:Created>2005-01-27T16:46:10Z</wsu:Created>
      <wsu:Expires>2005-01-27T18:46:10Z</wsu:Expires>
      </wsu:Timestamp>
      <dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
Id="sb738c7">
         <dsig:SignedInfo Id="obLkHzaCOrAW4kxC9az0bLA22">
...
            <dsig:Reference URI="#s91397860">
...
      <dsig:DigestValue>5R3GSp+OOn17lSdE0knq4GXqgYM=</dsig:DigestValue>
      </dsig:Reference>
            </dsig:SignedInfo>
      <dsig:SignatureValue
Id="a9utKU9UZk">LIkagbCr5bkXLs8l...</dsig:SignatureValue>
      <dsig:KeyInfo>
         <wsse:SecurityTokenReference>
            <wsse:Reference URI="#aXhOJ5"
ValueType="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
            </wsse:SecurityTokenReference>
      </dsig:KeyInfo>
      </dsig:Signature>
         </wsse:Security>
      </soap:Header>
      <soap:Body xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="s91397860">
         <xenc:EncryptedData
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Id="aDNa2iD" Type="http://www.w3.org/2001/04/xmlenc#Content">
         <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
            <xenc:CipherData>
<xenc:CipherValue>XFM4J6C...</xenc:CipherValue>
            </xenc:CipherData>
         </xenc:EncryptedData>
      </soap:Body>
</soap:Envelope>
```

Sembol Çeşitleri

Bir WSS başlığında şu türde güvenlik sembolleri bulunur:

- Kullanıcı-adı Sembolü:

Kullanıcı adı ve, tercihe bağlı olarak parola, geçme mekanizmalarını tanımlar- parola kısmı kullanıcı-adı profilinde anlatılır. Tüm sembol şifrelenmediği sürece, düzyazı şeklinde parola içeren her mesaj güvenli kanal üzerinden taşınmalıdır. Hedef web sunucusunun doğrulama için düzyazı parolalara erişimi olduğu durumlarda(bu LDAP veya başka diğer kullanıcı dizinlerinde geçerli değildir), zaman etiketi ve rastgele bir değerle birlikte alınan hash değeri tarcih edilir.

Parola = Base64(SHA-1 (nonce + zaman etiketi + parola))

- Binary Sembolü:

Yazı şeklinde binary veri taşımak için kullanılır, X.509 sertifikaları gibi; varsayılan biçim Base64'tür. Temel doküman BinarySecurityToken elemanını açıklarken, profil dokümanı birçok değişik sembolün eklenmesi ile ilgili fazladan özellikler ve alt-elemanlar tanımlar. Şu anda, X.509 profili benimsendi, ve çalışma kerberos profiline doğru ilerliyor.

```
<wsse:BinarySecurityToken EncodingType="http://docs.oasisopen.
org/wss/2004/01/oasis-200401-wss-soap-message-security-
1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-
token-profile-1.0#X509v3" wsu:Id="aXhOJ5">
MIICtzCCAi...
</wsse:BinarySecurityToken>
```

- XML Sembolü:

Bu herhangi bir tür XML-tabanlı sembol için geliştirildi, fakat öncelikli olarak SAML cümleleri için kullanılır. Temel doküman sadece bu sembollerin kullanılması ihtimalinden bahsederken, ayrıntıları profil dokümanına bırakıyor. Şu anda, SAML 1.1 OASIS tarafından kabul edilmiş durumdadır.

```
<saml:Assertion AssertionID="1106844369755" IssueInstant="2005-
01-
27T16:46:09.755Z" Issuer="www.my.com" MajorVersion="1" MinorVersion="1"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
...
</saml:Assertion>
```

Teknik olarak bür güvenlik sembolü olmamasına rağmen, mesajın tazeliğini belirtmek üzere bir zaman etiketi güvenlik başlığına eklenebilir. İleriki okuma kısımlarına bu konuda tasarım modelleri görebilirsiniz.

Mesaj Parçacıklarına Atıf

Mesajda geçen güvenlik sembollerini almak için ya da imzalanmış ve şifrelenmiş mesaj parçacıklarını tanımlamak için, temel dokümanda özel bir özelliğin kullanımı benimsenmiştir: wsu:Id. Bu özellikteki tek gereksinim, tanımlandığı XML-dokümanındaki bu Idlerin değerlerinin tek olmasıdır. Uygulaması ara işlemciler için çok

önemli bir avantajdır, çünkü çalışması msajın XML-şemasını anlamayı gerektirmiyor. Fakat ne yazık ki XML imza ve şifreleme tanımları özellik genişlemesine izin vermiyor (kapalı şemaya sahipler), bu yüzden, imzayı veya şifreleme elemanlarını belirlerken imza ve şifre elemanlarının yerel ID'leri öncelikli olarak düşünülür.

Wss temel dokümanı ayrıca güvenlik sembollerine SecurityTokenReference elemanını kullanarak erişmeye yarayan genel bir mekanizma da tanımlar. Bu elemana bir örnek, aşağıdaki SAML cümlesindedir:

```
      <wsse:SecurityTokenReference wsu:Id="aZG0sGbRpXLySzgM1X6aSjg22">
            <wsse:KeyIdentifier ValueType="http://docs.oasisopen.
org/wss/2004/XX/oasis-2004XX-wss-saml-token-profile-
1.0#SAMLAssertionID"
wsu:Id="a2tv1Uz">
1106844369755
            </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
```

Bu eleman WSS başlığının içindeki veya dısındaki harhangi bir sembol tipi için tasarlanmış olsa da (şifreleme anahtarları, sertifikalar, SAML cümleleri, gibi), oldukça karmaşıktır. Doküman iki muhtemel kullanım tipini tavsiye eder: Doğrudan kullanımı (URI yolu ile) ve anahtar tanımlaması için (bir çeşit sembol tanımı). Profil dokümanları (SAML, W.509 gibi) bu mekanizmalara değişik sembol türlerinin özelliklerini kullanma genişliği sağlar.

İletişim Korum Mekanizmaları

Önceden de açıklandığı gibi (bkz. 0), kanal güvenliği, önemli servisler sağlarken, Web Servisi geliştiricilerin karşılaştığı her sorunu da çözmez. WSS, aşağıdaki bölümlerde tanımlanan mekanizmaları kullanarak bu sorunların bazılarının SOAP seviyesinde çözülmesine yardımcı olur.

Bütünlük

WSS dokümanı mesaj bütünlüğünü sağlamak için XML-dsig standardını kullanırken, belli bazı durumlarda fonksiyonelliğini kısıtlıyor, örneğin sadece açıkça basvurulan elemanlar imzalanabilir (mesela, gömme veya gömülü imza modlarına izin verilmez). Bir XML dokümanını imzalamadan önce, dokümanın benzer bir versiyonunu oluşturmak için bir dönüşüm gereklidir. W3C'de XML Dijital İmza WG tarafından tanımlanmış, namespace tanımlarının işlenmesine göre, iki ana dönüşüm bulunur: Inclusive ve Exclusive dönüşüm. İmzayı değiştirmeden imzalı XML dokümanlarının diğer dokümanlara kopyalanmasına izin verdiğinden, temel WSS dokümanında özel olarak EXC-C14N kullanımı tavsiye ediliyor.

İmzalı sembolleri adreslemek için tek bir yol sağlamak için, WSS bir Security Token Reference(STR) Dereference Transform seçeneği ekler; bu bazı programlama dillerindeki belirli veri türlerini dereference etmeye benzer. Aynı şekilde, XML imzası şeklinde tanımlı anahtar imzalamaya ek olarak, WSS, STR mekanizması yoluyla

güvenlik sembolü imzalamaya da izin verir. Tipik imza örneği daha önce, bölüm 0'da verilmişti.

Tipik olarak, kullanıcı bilgileri ile birlikte SOAP gövdesi ve zaman etiketi gibi güvenli olması gereken elemanlara XML imzası uygulanır. Bir eleman hem imzalandığında hem de şifrelendiğinde değişik bir durum söz konusudur, çünkü bu işlemler herhang bir sıra takip edebilirler, ve bu sıranın bilinmesi imza teyiti için gereklidir. Bu konu için, WSS temel dokümanında her elamanın güvenlik başlığına ön-ek olarak eklenmesi gerektiği, ve işlemlerin doğal sırasının bu olması gerektiği yazar. İstisnai bir problem, bir SOAP mesajında üsteüste gelen imza ve şifre blokları kullanan birden fazla güvenlik başlığı bulunması durumunda ortaya çıkar, bu durumda doğru sıralamayı gösterecek herhangi bir şey yoktur.

Gizlilik

Gizliliğinin korunması için, WSS başka bir standarda güvenir, XML şifreleme. XML-sig'e benzer olarak, bu satndart SOAP mesajının seçilen elemanları üzerinde işlem görür, fakat şifrelenmiş elemanın verisini <xenc:ExcryptedData> alt elemanında ki şifrelenmiş byte grubu ile değiştirir. Şifreleme verimi için, doküman eşsiz bir anahtar kullanımını tavsiye eder, bu anahtar daha sonra alıcının açık anahtarı ile şifrelenerek güvenlik başlığına <xenc:EncryptedKey> elemanı içerisinde ön-ek olarak eklenir. Şifrelenmiş bir gövdeye sahip bir SOAP mesajı bölüm 0'da gösterilmiştir.

Tazelik

SOAP mesajlarının tazeliği zaman etiketi mekanizması ile adreslenir- her güvenlik başlığı sadece bir eleman bulundurur, ve bu eleman UTC zaman formatında, güvenlik başlığının oluşturulma ve süre dolma zamanlarını belirtir. Zaman etiketinin SOAP mesajının kendisine değil zaman etiketine uygulandığının farkına varmak önemlidir, ve bu şekilde birden fazla güvenlik başlığı ve farklı zaman etiketleri bulundurulabilir. Bu "tek zaman etiketi" ile ilgili çözülmemiş bir problem bulunuyor; zaman etiketi oluşturulup imzalandığında,varolan imzasını bozmadan güncellemek mümkün değildir, WSS başlığında olması gereken bir değişiklik için bile.

```
<wsu:Timestamp wsu:Id="afc6fbe-a7d8-fbf3-9ac4-f884f435a9c1">
 <wsu:Created>2005-01-27T16:46:10Z</wsu:Created>
<wsu:Expires>2005-01-27T18:46:10Z</wsu:Expires>
</wsu:Timestamp>
```

Eğer zaman etiketi bir mesaja eklenirse, değiştirmeye ve tekrar edilmeye karşı imzalanır. Saat ayarlamasını hedefleyen herhangi bir mekanizma yoktur (önceden de söylendiği gibi, modern sistemlerde ele alınmaz) – WSS mekaniğine göre bu konu dışında incelenmelidir. Bu prolemi ele alan okuma bölümüne bakınız.

Erişim Kontrol Mekanizmaları

Konu erişim kontrol karalarına geldiğinde, web servislerinin kendileri özel koruma mekanizmaları sağlamazlar – sadece sembolleri ve veri yüklerini güvenli bir şekilde kaynak ve hedef SOAP uç noktaları arasında taşımak için gerekli unsurları bulundururlar.

Erişim kotnrol görevleri ile ilgili ayrıntılı bilgi için bu rehberin diğer bölümlerine bakınız.

Tanımlama

Tanımlama belirli bir kimlik sahibi olunduğunu iddia etmektir ve mesaja bu bilgiyi eklemek ile ifade edilir. Bu bir kullanıcı adı, SAML cümlesi, Kerberos etiketi, veya başka bilgi parçacığı olabilir, ve bu parçacıktan servis çağrıyı yapanın kimliğini çıkartabilir.

WSS, mesaja değişik türden semboller ekleyebildiği için (bkz 0), bu bilginin yayımı için güzel bir yol sunar. Eklenen sembolü açıp hangi kimliğe ait olduğunu bulmak, veya sembol bulamazsa mesajı reddetmek alıcının sorumlulundadır.

Kimlik Doğrulama

Kimlik doğrulama iki şekilde yapılabilir: kimlik bilgilerinin teyidi veya sembol geçerlemesi. Bu ikisi arasındaki gizli farklılık da sembollerin bir şekilde önceden yapılmış bir kimlik doğrulama işleminden sonra kullanılmasıdır, ve genellikle kullanıcının kimliğiyle birlikte kimliğin bütünlük kanıtı birlikte taşınır.

WSS, protokole özel sembollerinbağlanma mekanizmalarını tanımlayarak ve güvenilir bir şekilde gönderici ile ilişkilendirerek, birçok standart kimlik doğrulama protokolünü destekler. Bununla birlikte, çağrı sahibinin iddia ettiği kişi olduğunun kanıtının mekaniği tamamen Web Servisinin takdirine kalır. Sağlanan kullanıcı adı ve parolanın hash değerini alıp, kullanıcı veritabanında kontrol edebilir, veya X.509 sertifikasından konu ismini çıkartabilir ve sertifika zincirini teyit edebilir; şu anda hangi yöntemle yapılması gerektiği konusunda herhangi bir gereksinim veya standart yoktur.

Yetkilendirme

XACML yetkilendirme kurallarını tanımlamak için kullanılabilir, fakat kullanımı Web-servislerine özel değildir-çok daha geniş bir kapsamı vardır. Bu yüzden, sunucu tarafında halihazırda kullanılan herhangi bir politika veya rol-tabanlı yetkilendirme mekanizması web-servisleri için de kullanılabilir. Uygulamaya bağlı olarak, sunucu tarafında birçok yetkilendirme katmanı bulunabilir. Örneğin, JSR'ın 224(JAX-RPC 2.0) ve 109 (Implementing Enterprise Web Services), Web servislerinin java bağlantılarını tanımlar, web servislerinin J2EE konteynırlarında geliştirilmesini uygun görür. Bu demektir ki, bir web servisine erişimde bulunulduğunda, J2EE konteynırında url yetki kontrolü olacaktır, ve sonrasında da web-servisi spesifik kaynağın web servisi katmanında bir kontrol olacaktır. Bu kontrollerin sayısı, uygulama spesifiktir, ve herhangi

bir standartta belirtilmez. Windows aleminde de benzer bir şekilde gerçekleşir, IIS gelen HTTP çağrılarını, ASP.NET'e ulaşmadan önce bir kontrol yapar, daha sonra SOAP mesajları daha küçük parçalara ayrılıp incelenir.

Politika Anlaşması

Normalde, web servislerinin iletişimi, WSDL dosyasında belirtildiği gibi, kamu arayüzleri aracılığıyla olur. Dokümanda SOAP bağlantı gereksinimleri yeterli ayrıntıda anlatılmıştır, fakat herhangi bir güvenlik parametresi belirtilmemiştir, ve bu yük, yani bu arayüzlerde güvenlik gereksinimlerini belirlemek ve sağlamak, web servisi geliştiricilerine kalmıştır.

Bu eksiklikleri örtmek için, WS-Policy dokümanı karmaşık politika gereksinimlerini ve kalitelerini tanımlamak üzere tasarlanmıştır. Yayınlanan politika ile SOAP uç noktaları kendi güvenlik gereksinimlerini yayabilir ve istemcileri de taleplerini hazırlarken gereken uygun mesaj koruma önlemlerini alabilirler. Genel WS-Policy dokümanında (aslında 3 ayrı dokümandan oluşur) ayrıca özel politika türleri için de eklentiler vardır, örneğin güvenlik için WS-SecurityPolicy.

Eğer talep eden gereken parçacıklara sahip değilse, bu parçacıkları güven mekanizmaları yoluyla alabilirler. WS-trust servisleri, birçok güvenli parçacık değişimi talebinde çağrılır.

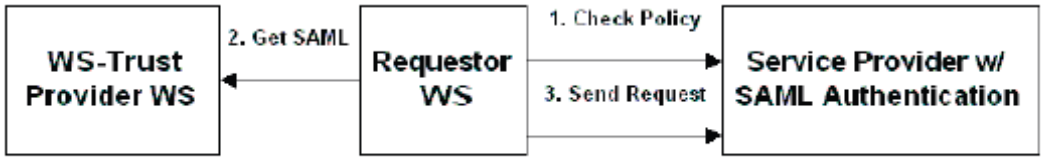

Figure 3. Using Trust service

Ne yazık ki, WS-Policy ve WS-Trust tanımlamaları standartlaştırılması için kamuya açılmamıştır ve geliştirme süreçleri birçok şirketin işbirliği sonunda gerçekleştirilmiştir, her ne kadar diğer katılımcılara da açık olmuş olsa da. Artı olarak da, bu tanımlamalar için birçok birlikte çalışmalar yapılmıştır, bu yüzden Web Servislerinin altyapısındaki bu önemli bağlantıların geliştirme süreci tamamen kapalı kutu değildir.

Web Servis Zinciri Oluşturma

Varolan veya planlanan SOA veya B2B sistemlerinin çoğu, emir almadan, üretime hatta dağıtıma kadar birçok görevini yapabilmek üzere dinamik Web Servisleri zincirine güvenir.

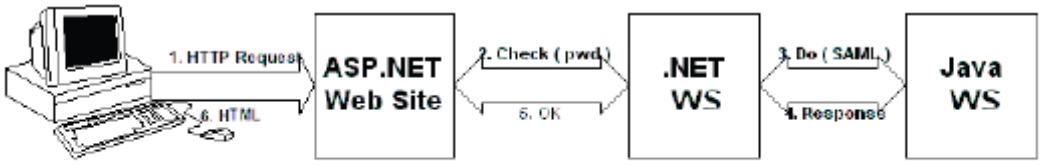

Figure 4: Service chain

Bu teoride doğrudur. Pratikte ise, bu yolda gizli birçok engel bulunur, ve bu engellerin en büyüğü, intra veya Inter-net tabanlı istemcilere açılan işlem fonksiyonlarının güvenliği.

Web servislerinin etkileşimini engelleyen konulardan bazıları – kullanıcıların kimlik doğrulaması ve yetkilendirilmeleri için uyumlu olmayan modeller kullanmaları, servislerin kendi aralarındaki güven ilişkisi ve bu ilişkinin kurulumu, güvenli bağlantılar sağlama, ve kullanıcı dizinlerinin senkronizasyonu, veya kullanıcıların bilgilerinin alış-verişi. Bu konulara sıradaki paragraflarda değiniliyor.

Uyumsuz Kullanıcı Eşimi Kontrol Modelleri

Önce de açıklandığı bigi, bölüm 0'da, web sevislerinin kendileri erişim kontrolü için ayrı eklentiler bulundurmazlar, bunun yerine varolan güvenlik mekanizmalarına güvenirler. Fakat, bunun yerine SOAP servisinin güvenlik gereksinimlerini öğrenir ve tanımlarlar (WS-policy sayesinde), ve gerekli güvenlik bilgilerini WS-Trust tabanlı servisler sayesinde elde ederler.

Servislerin Güven İlişkisi

İstemci ve sevis arasında karşılıklı güven sağlanması için, iki taraf da birbirlerinin politika gereksinimlerini karşılamalıdırlar. Bunu sağlayan basit ve popüler bir model SSLdir, fakat SSL açık servis modelleri için ölçeklenebilir değildir, ve sadece bir kimlik doğrulama tipini destekler. Daha çok esneklik gerektiren servisler, kullanıcılar ile konuşmadan önce yaptığı erişim kontrol mekanizmasına benzer bir mekanizma kullanmalıdırlar.

Güvenli Bağlantılar

İki taraf arasında güven oluşturulduğunda, bu güvenin her etkileşimden önce teyit edilmesini gerektirmek çok pratik olmaz. Bunun yerine, güvenli bir istemci-sunucu bağlantısı oluşturulur ve istemcinin oturumu aktif iken bu bağlantı tutulur. Yine, böyle bir bağlantı için kullanılan en popüler mekanizma SSLdir, fakat bu web servislerine özel bir mekanizma değildir, ve SOAP iletişimlerinde kullanıldığında bazı eksikleri görülmüştür, 0'da anlatıldığı gibi.

Kullanıcı Dizinlerinin Senkronizasyonu

Bu çapraz-etki alanı uygulamaları ile uğraşırken karşılaşılan önemli bir problemdir, çünkü faklı etki alanlarındaki kullanıcı sayıları çok değişkendir. Bu yüzden,

etki alanı B'deki bir servis, etki alanı A'daki bir kullanıcının A etki alanında kimlik doğrulamasının yapıldığına nasıl güvenecek? Bu problemin değişik açıları vardır. Öncelikle- genel bir SSO mekanizması, kullanıcının iki etki alanında da tanınmasını gerektirir(senkronizasyon veya başka bir yolla), ve bir etki alanından alınan kimlik doğrulama parçacığının diğerinde kabul edilir olması gerekir. Web servisleri dünyasında, bu kullanıcının SAML veya Kerberos parçacığının dolaştırılması ile gerçekleşir.

Etki Alanı Birleşmeleri

Problemin bir başka açısı da kullanıcılar etki alanları arasında paylaşılmazken, fakat bir etki alanında, belirli bir ID ile kimliği doğrulanmış bir kullanıcının, büyük kuruluşlarda olduğu gibi, kullanıcı bilgilerini paylaşmadan ortaklık oluşturma isteği olabilir. Bu talebi kabul etmede alınan karar etki alanları arası prosedürlere bağlıdır, özel güven ilişkileri kurmak ve bu kapalı parçacıkları değiştirmek gibi. Bu gayretlerden en göze batanı Library Alliance projesidir, ve şu anda SAML 2.0 dokümanları için temel oluşturur. Bu alandaki çalışma halen bitim aşamasından çok uzaktadır, ve varolan yerleştirmelerin çoğu POC'tan başka birşey değildir, veya şirketler arası yerleştirme değil de, yerel pilot projelerdir, her ne kadar LA'in web sayfasında büyük çlçekli projeler hakkında çalışmalar olsa da.

Varolan Uygulamalar

En başından farketmek gerekir ki, herhangi bir güvenlik standardı tek başına mesaj alış verişi konusunda güvenlik sağlamaz – bunu yapan yüklenmiş uygulamalardır. Gelen SOAP mesajlarının standartlara uygunluğunu değerlendirdiği gibi, giden mesajlara da güvenlik mekanizmaları uygulayarlar.

.NET – Web Servis Eklentisi

Yeni standartlar yarış halinde geliştirildiğinden, .Net platformu hepsine yetişmek yerine, Web Servis Eklentilerini (Web Service Extensions - WSE) kullanır. WSE, şu anda versiyon 2.0'dadır, ve son Web Servisleri güvenlik standartlarına geliştirme ve uygulama desteği sağlar, hala "ilerleme aşamasında" olmalarına rağmen. Standartlar olgunlaştığında ise, destekleri .Net platformunun yeni versiyonlarına eklemek suretiyle devam eder, ve bu .Net 2.0 görevini tamamladığında olacaktır. WSE'nin yeni versiyonu, 3.0, VS2005 ile birlikte gelicek ve mesajlaşma ve Web Uygulaması alanında .Net 2.0 platformunun en yeni buluşlarını kullanacaktır.

Micrsoft'un Web Servislerinin güvenliği konusunda en aktif oyunculardan bir olduğunu ve endüstrideki etkisini düşünürsek, WSE büyük ihtimalle en tamamlanmış ve en güncel uygulama olacaktır ve şiddetle tavsiye edilir ki, en azından WSE-güvenilir .Net Web Servis istemcisi ile kısa bir birlikte çalışma kontrolü yapılmalıdır. Eğer java-tabanlı bir web servisiniz varsa, ve birlikte çalışabilirlik bir gereksinim ise (ki genelde öyledir), güvenlik testlerinin yanında, java ve .Net Web servislerinin temel birlikte çalışabilirliğinin de göz önünde bulundurulması gerekir.

Bu özellikle önemlidir, çünkü .Net Web Servis araçlarının şu andaki versiyonu WS Security'yi ve ilgili OASIS XML şemalarını temiz bir şekilde idare edilmez, bu yüzden, Web Service tasarımcısının biraz yaratıcı olması gerekir. Bunu söylemişken, WSE paketinin kendisinin, oldukça zengin ve güzel-yapılandırılmış bir fonksiyonalitesi vardır, ve ASP.Net tabanlı ve özerk Web-servisi istemcileri tarafından, gelen SOAP mesajlarını kontrol etmek ve giden mesajların güvenliğini sağlamak için kullanılabilir, ve Web Servis programcılarının bu ayrıntıları bilmesine gerek yoktur. Bunların dışında, WSE 2.0 en son WS-Policy ve WS-Security profillerini destekler, ve temel mesaj güvenliğini sağlar. Bunlar güvenli alışveriş sağlamak için gereklidir, tıpkı SSL'in taşıma katmanında (transport layer) yaptığı gibi, fakat mesaj-tabanlı iletişime uygulanır.

Java Araç Kitleri

Varolan Java araç kitlerinin çoğu XML güvenliği seviyesinde çalışır, örneğin XML-dsig ve XML-enc gibi, mesela IBM'in Security Suite ve Apache'nin XML Security projesi. Java'nın JSR 105 ve JSR 106(halen tamamlanmadı) imzalar ve sifrelemeler ile ilgili Java bağlantılarını tanımlar, ve bu, uygulamaların, JSR'lar tamamlandığında, JCA sağlayıcılar gibi bağlanmalarına izin verecek.

Bir kademe üste çıkarsak, Web Servislerinin kendisine, resim biraz daha bulanıyor, şu anda birçok uygulama birçok tamamlanmama kademesinde bulunuyor. Örneğin, Apache şu anda WSS4J projesi üzerinde çalışıyor, ve bu çalışma biraz yavaş ilerliyor, ayrıca birçok uygulama problemi bulunan Phaos'un(şu anda Oracle'a ait değil) ticari yazılımı bulunuyor.

Günümüz Web servisi geliştiricileri arasında popüler bir seçim olan Sun'ın JWSDP'si Web Servis güvenliği için destek içerir. Fakat, versiyon 1.5'daki Web Servis güvenliği dokümanı desteği temel WSS standardı ile kullanıcı adı ve X.509 sertifika profili ile kısıtlıdır. Güvenlik özellikleri JAX-RPC iskeletinin bir parçası olarak eklenmiş ve konfigürasyon ile çalışır, ve bu Web Servis uygulamasından temiz bir şekilde ayrılmasına izin verir.

Donanım, yazılım sistemleri

Bu kategori araç kitleri veya iskeletlerden çok tamamlanmış sistemleri içerir. Bir tarafta, genelde hazır ve sengin bir fonksiyonalite sağlarlar, diğer tarafta kullanım modeli çözümün mimarisi ve uygulaması ile katı bir şekilde sınırlıdır. Bu araç kitlerine zıt olarak, kendi başlarına herhangi bir servis sağlamazlar, fakat sistem geliştiricilerine arzulanan Web Servis güvenlik özelliklerini ürünlerine içermeleri için gerekli araçları sağlarlar.

Bu sistemler altyapı kademesinde gelen mesajların hedef Web Servislerine gönderilmeden önce efektif politikalar, kontrol imzaları, parçacıklar, vs. gibi açılardan teyit edilmesinde kullanılabilir. Giden SOAP mesajlarına uygulandığında, proxy gibi hareket eder, mesajları gerekli güvenlik elemenlarını içerecek şekilde değiştirir, ve imzalar ve/veya şifreler.

Yazılım sistemleri önemli konfigürasyon esnekliği fakat yavaş işlemleriyle karakterize edilebilir. İyi tarafta, çoğu zaman varolan altyapı ile birlikte yüksek seviyede entegrasyon sağlar. Bu servislere bir örnek, eski adıyla Netegrity'den TransactionMinder – arkasındaki Web Servisleri için bir Politika Yaptırım Noktası ile, politika sunucusunun üzerinde çalışan ve politika kararlarını konfigure depolardan ve politikalardan çıkarttığı bilgileri kontrol ederek veren bir servistir.

Donanım sistemleri için, performans anahtar kelimedir; gigabyte işlem eşiği çoktan geçilmiştir, büyük dokümanların gerçek zamanlı işlenmesine izin veren ve birçok son sürüm Web Servis güvenlik standardına göre tasarlanmıştır. Bu sistemlerde kullanım kolaylığı bir başka çekici noktadır- birçok sıradan durumda, donanım kutusu getirilir, bağlanır ve anında çalıştırılır. Bu kalite tabi ki bir bedel karşılığında gelir, fakat, bu performans ve kolaylık, kullanıcı donanım kutusunun konfigürasyon sınırlarında kaldığı sürece sağlanır. Arka depolarla entegre olmaya çalıştığı anda bu avantajların çoğu kaybolur. Bu şekil donanım cühazlarına bir örnek olarak, hem güvenlik duvarı hem de XML trafiği için proxy olabilen, DataPower'ın sağladığı XS40 XML Güvenlik Kapısı'dır.

Problemler

Önceki bölümden de anlaşılacağı gibi, Web Servisleri halen birçok türbülans yaşamaktadır, ve bu durum düzelene kadar biraz zaman geçmesi gerekir. Burada, şu an varolan güvenlik standartlarında ve onların uygulamalarında bulunan problemlere kısa bir bakış sunacağız.

Standartların Hamlığı

Standartların çoğu ya çok yeni (en çok birkaç yıllık), ya da halen geliştirilmektedir. Standart geliştirmeler komiteler halinde yapılmasına rağmen, ki bu tamamının gözden geçirilmesi riskini azaltır, bazı hata senaryoları periyodik olarak gerçekleşir, çünkü hiçbir teori binlerce geliştiricinin çalıştığı bir projenin testleriyle birebir eşleşmez.

Buna ek olarak, bazı politik sebeplerden dolayı bu standartlar kamudan saklanmıştır, WSA arenasındaki birçok standartta olduğu gibi, ve bazı uğraşlar iki kez gerçekleştirilmiştir, LA ve WS-Federation dokümanlarında olduğu gibi.

Performans

XML gramerinin incelenmesi yavaş bir görevdir, ve bu kabul edilen bir gerçektir, ve SOAP işlemleri bunu daha da yavaşlatır. Şimdi ise, pahalı kriptografik ve yazı çevirim işlemleri bu karışıma katılmıştır, ve böylelikle bu görevler birer dar boğaz haline

gelmiştir -en son kriptografi ile bile- ve günümüzde XML-işleyen donanım çözümleri sunulmaya başlanmıştır. Şu anda markette bulunan bütün ürünler bu durumla karşı karşıyadır, ve bunu değişik derecelerdeki başarı oranlarıyla çözmeye çalışmaktadırlar.

Donanım çözümleri, performansı oldukça arttırırken, her zaman en uygun çözüm olarak kullanılamaz, çünkü varolan arka-plan yazılım altyapısına kolayca entegre edilemezler, en azından – performans fedaları yapılmadan. Donanım-tabanlı çözümlerin doğru olup olmadığına dair üzerinde düşünülen bir diğer konu ise- donanım çözümleri genelde yaptıkları işlerde yüksek derecede uzmanlaşmışlarken, modern Uygulama Sunucuları ve güvenlik iskeletleri genellikle çok daha fazla miktarda değişik koruma mekanizması sunar, sadece Web servislerini değil diğer kurulu uygulamaları da tutarlı ve benzer biçimde korur.

Karmaşıklık ve Birlikte Çalışırlık

Önceki bölümlerden de anlaşılacağı gibi, Web Servis güvenlik standartları oldukça karmaşıktır, ve oldukça dik bir öğrenme eğrisine sahiptir. Şu andaki ürünlerin çoğu, Web Servislerin güvenliğiyle ilgilenen, altlarında yatan yapının karmaşıklığından kaynaklanan orta karar bir kullanılabilirlikten yakınırlar. Bütün bu değişik politikaları, anahtarları, protokolleri konfigüre etme ve alakalı teknolojinin anlaşılması bir sürü zaman alır, ve çoğu zaman son kullanıcının gördüğü hatalar garip ve anlaşılmaz açıklamalara sahiptir.

Yöneticilere yardım etmek için ve servis yanlış konfigürasyonundan doğabilecek güvenlik risklerini azaltmak için, bir çok firma politika şablonları geliştirir, ve gelen ve giden SOAP mesajlarını korumak için alınacak en iyi önlemleri gruplar. Ne yazık ki, bu çalışma henüz hiçbir standartın ilgi alanında bulunmuyor, ve görülen o ki, bu şablonlar kısa bir zaman içinde kamunun kullanımına açılacak. Bu uğraşa en yakını, WS-I'ın Basic Security Profile(BSP- Temel Güvenlik Profili) sayılabilir, ve bu profil, WSS gibi bir çok standarttan çıkarılan genel güvenlik özelliklerinin bir alt kümesini kullanarak, Web Servislerinin en iyi birlikte çalıabilirlik için gereken kuralları tanımlar. Fakat, bu çalışma yöneticilere en popüler kullanım alanlarınını eşleyen güvenlik şablonlarını sağlamayı değil de, bu şablonların en küçük ortak katlarını kurmayı hedefler.

Anahtar Yönetimi

Anahtar yönetimi, çoğu koruma mekanizması bi şekilde kriptografik anahtarlara güvendiğinden, genellikle herhangi başka bir güvenlik aktivitesinin oluşturulmasında gerçeklenir. Web servisleri anahtar dağıtımı için XKMS protokolüne sahipken, yerel anahtar yönetimi halen bir çok durumda çok büyük bir zorluktur. Bu sistemler anahtar yönetimi için kendi mekanizmalarını kullanma taraftarıdırlar ve bu birçok durumda önemli ölçüde risk taşır. Gizli ve özel anahtarların saklanması, güncellenmesi, kurtarılması gibi sorular hiç olmadığı kadar çok sorulmaya başlanmıştır ve bu çözümlerde yeterli cevaplar bulamamışlardır.

İlave Okunabilecekler

- Piliptchouk, D., WS-Security in the Enterprise, O'Reilly ONJava

http://www.onjava.com/pub/a/onjava/2005/02/09/wssecurity.html
http://www.onjava.com/pub/a/onjava/2005/03/30/wssecurity2.html

- WS-Security OASIS site
  http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- Microsoft, *What's new with WSE 3.0*
  http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx?pull=/libr
  ary/en-us/dnwse/html/newwse3.asp
- Eoin Keary, Preventing DOS attacks on web services
  https://www.threatsandcountermeasures.com/wiki/default.aspx/ThreatsAndCo
  untermeasu
  resCommunityKB.PreventingDOSAttacksOnWebServices

Dökümanın Aslı : OWASP GUIDE 2.0.1

Çeviri : Emre Çakır - cakiremre@gmail.com