WEB SERVİS GÜVENLİĞİNDE TOKEN BAZLI YENİ YAKLAŞIM

Mehmet CİNCİ
201402202

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Ceren ÇUBUKÇU CERASİ

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Eylül, 2022

WEB SERVİS GÜVENLİĞİNDE TOKEN BAZLI YENİ YAKLAŞIM

Mehmet CİNCİ
201402202
ORCID: 0000-0003-4256-9388

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Ceren ÇUBUKÇU CERASİ

İstanbul
T.C. Maltepe Üniversitesi
Lisansüstü Eğitim Enstitüsü
Eylül, 2022



JÜRİ VE ENSTİTÜ ONAYI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" ile bildirilen 6698 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

ETİK İLKE VE KURALLARA UYUM BEYANI

Bu belge, Yükseköğretim Kurulu tarafından 19.01.2021 tarihli "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" ile bildirilen 6698 Sayılı Kişisel Verilerin Korunması Kanunu kapsamında gizlenmiştir.

TEŞEKKÜR

Bu çalışmanın gerçeklemesinde, yardımlarını esirgemeyen, değerli bilgilerini benimle paylaşan ve yönlendiren, her zaman yanımda olan ve sonsuz desteğini hissettiren, çok kıymetli danışmanın Dr. Öğr. Üyesi Ceren ÇUBUKÇU ÇERASİ'ye şükranlarımı sunarım.

Tez çalışmam için her zaman yanımda olan akademik başarımının devamında bana yol gösteren ve teşvik eden AR-GE Müdürüm Dr. Muaz GÜLTEKİN'ne çok teşekkür ederim.

Son olarak tüm kalbi ile beni destekleyen eşim Şenay'a ve her zaman benim yanımda olan sevgili aileme sonsuz teşekkürler.

Mehmet CİNCİ

Eylül 2022

ÖZ

WEB SERVİS GÜVENLİĞİNDE TOKEN BAZLI YENİ YAKLAŞIM

Mehmet Cinci
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı
Danışman: Dr. Öğr. Üyesi Ceren Çubukçu Çerasi
Maltepe Üniversitesi Lisansüstü Eğitim Enstitüsü, 2022

Günümüzde internetin yaygınlaşması ile kurumlar arasında verilerin iletilmesinde web servisler yoğun bir şekilde kullanılmaktadır. Web servisler ile veri transferlerinde üçüncü kişilerin erişimini önlemek için SSL / TLS kullanılmaktadır. Bu güvenlik çözümü sayesinde istemci ile sunucu arasındaki haberleşme güvenliği sağlanmış olunacaktır. Web servislerin üçüncü kişiler tarafından zafiyet saldırılarına uğradıkları gözlemlenmiştir. Bu atakların en popüler olanları XML Injection, XPath Injection, SQL Injection, Spoofing ve Denial of Service günümüzde de devam etmektedir. Bu saldırılara karşı bizim tavsiye etiğimiz XAdES tabanlı çözüm olacaktır. Burada karşılaştığımız en büyük problem; talep edilen Envelope'un istenilen XML formatı, Dijital imza ve imza türüne göre hazırlanıp sunucu tarafına iletilmesidir. Ayrıca XAdES oluşturmada en büyük problem, dijital imzalamada bulunan ve imzalama sırasında kullanılan Private Key'in satıcı kurum tarafından asimetrik olarak güvenlik nedenlerinden dolayı açık sunulmamasıdır. Genellikle kurumun sunduğu veya üçüncü şahıslar tarafından üretilen API'lere ihtiyaç duymaktadırlar. API'lerin kullandığı imzalama algoritmaları veya yapıların bazen yetersiz olduğu gözlemlenmiştir. Bundan dolayı kurumlar arasında özel kütüphaneler ile çözüme gidilmeye çalışılmıştır. Kurumlar arasındaki yapılan veri aktarımları istemci ile sunucu arasında Geliştirme, Test, Kalite Kontrol ve Canlı süreçlerinin zaman planlamasının yapılamamasından kaynaklanmaktadır. Ayrıca SOAP mesajının içerisinde yer alan Timestamp bilgisinin ömrünün çok kısa olması, konum, yer ve zaman farklılığın olmasından dolayı proje maliyetinin de hesaplanamadığı gözlemlenmiştir. Bu sistemlerin devlet kurumları tarafından zorlama yolu ile özel kurumlarda yaygınlaştırılması sağlanmıştır.

Anahtar Sözcükler: Web Servis Güvenliği; Dijital İmza; XAdES; SOAP; WSG-Prefiks

ABSTRACT

TOKEN BASED NOVEL APPROACH TO WEB SERVICE SECURITY

Mehmet Cinci
Master Thesis
Department of Computer Engineering
Computer Engineering Arts Masters Programmer
Advisor: Asst. Prof. Ceren Çubukçu Çerasi
Maltepe University Graduate School, 2022

Today, with the widespread use of the internet, data exchange between institutions has started to be done with web services. SSL / TLS was started to be used to prevent third party access in data transfers and only communication security between client and server was ensured. It has been observed that web services have been subjected to vulnerability attacks by third parties. The most popular of these attacks as of today are XML Injection, XPath Injection, SQL Injection, Spoofing, Denial of Service. Our recommendation against these attacks would be XAdES. The biggest problem faced here is that the requested Envelope is prepared according to the desired XML format, digital signature and signature type and transmitted to the server side. In addition, the biggest problem in creating XAdES is that the Private Key used in digital signing and used during signing is not presented asymmetrically by the vendor for security reasons. They usually need APIs provided by the institution or produced by third parties. Signing algorithms or structures used by APIs have sometimes been observed to be insufficient. For this reason, it has been tried to reach a solution with private libraries among institutions. Data transfers between institutions are due to the inability to schedule DEV, TEST, QA and PROD processes between the client and server. It has been observed that the project cost cannot be calculated due to the short life of the Timestamp information in the SOAP message, in addition to the location and time difference.

Keywords: Web Service Security; Digital Sign; XAdES; SOAP; WSS-Prefix

İÇİNDEKİLER

JÜRİ VE ENSTİTÜ ONAYI i
ETİK İLKE VE KURALLARA UYUM BEYANI ii
TEŞEKKÜR iii
ÖZ iv
ABSTRACT v
İÇİNDEKİLER vi
TABLOLAR LİSTESİ ix
ŞEKİLLER LİSTESİ x
KISALTMALAR xi
1. GİRİŞ 1
1.1 Web Servis 1
1.2 XML 4
1.3 SOAP 6
1.3.1 Zarf (Envelope) 7
1.3.1.1 Başlık (Header) 7
1.3.1.2 Gövde (Body) 8
1.3.1.3 Hata (Fault) 8
1.4 WSDL 9
1.5 UDDI 11
1.6 Servis Odaklı Mimari (SOA) 14
1.6.1 SOA faydaları 17

1.6.2 SOA sınırlamaları ve sorunları 18
1.7 Web Servis Güvenliği 19
1.8 Web Servis Güvenliğinde Saldırı Türleri 19
1.8.1 XML enjeksiyon 19
1.8.2 XML bombası 20
1.8.3 SQL enjeksiyon 20
1.8.4 Ortadaki adam saldırısı (MITM) 21
1.8.5 Hizmet reddi (DoS) 22
1.8.6 Sahtekarlığı (Spoofing) 22
1.9 Web Servis Güvenliğinde Sınırlıkları 22
1.10 Amaçlana Araştırmanın Özeti 22
2. YÖNTEM 24
2.1 Web Servis Güvenliğinde Envelope Belirlenmesi 24
2.1.1 Body hazırlama 25
2.1.2 X.509 v3 dijital sertifikasının e-imza üzerinden alınası 27
2.1.3 Header bilgisinin tanımlanması 27
2.1.4 Body'nin hazırlanması 28
2.1.5 Timestamp belirlenmesi 28
2.1.6 E-İmzanın belirlenmesi 29
2.1.7 İmzalama 29
2.1.8 İmzalama doğrulama 29
2.1.9 Schematron kontrolü 30

2.2 Sunucu'ya İletilmesi 30
2.3 Web Servis Güvenliğinde Sonuçlar 30
2.4 Web Servis Güvenliğinde Literatür Karşılaştırması 32
3. BULGULAR VE YORUMLAR 34
3.1 Web Servis Güvenliği için Klasik ve Geliştirilmiş Yöntem İşleyişi 34
3.1.1 Birinci modül 34
3.1.2 İkinci modül 35
3.2 Geliştirilmiş ve Klasik Önerimiz Hakkında 37
3.3 Aldığımız Veriler ve Verilerin Yorumlanması 38
3.4 Lokasyon ve Çalışma Vardiyası Sorunu 40
4. SONUÇ ve ÖNERİLER 43
4.1 Özet 43
4.2 Yargı 43
4.3 Öneriler 44
4.4 Çalışma Kısıtları ve Gelecekteki Çalışmamız 44
KAYNAKLAR 46
ÖZGEÇMİŞ 50

TABLOLAR LİSTESİ

Tablo 2. 100 adet SOAP zarf mesaj tablosu 39
Tablo 2. 1.000 adet SOAP zarf mesaj tablosu 39
Tablo 3. 10.000 adet SOAP zarf mesaj tablosu 40
Tablo 4. 6 ülke seçilerek bir gündeki mesai tablosu 42

ŞEKİLLER LİSTESİ

Şekil 1. Web Servis Mimarisi (Haviluddin, Edy, & Nur, 2019) 2
Şekil 2. Web Hizmetleri Mimarisi (Yongxin & Qin, 2016) 3
Şekil 3. İki kullanıcı bilgisinin bulunduğu basit bir XML dosyası 5
Şekil 4. Id, Ad ve Soyad bilgileri bulunan basit bir XMLSchema dosyası 5
Şekil 5. Tarih kontrolü içeren basit bir Schematron dosyası 6
Şekil 6. Basit bir SOAP Zarfı 7
Şekil 7. Basit bir hatalı bilgisi içeren SOAP Zarfı 8
Şekil 8. WSDL Bileşenlerinin Hiyerarşisi (Booth, 2007). 11
Şekil 9. Dağıtılmış UDDI sisteminin bilgi etkileşimi (Yongxin & Qin, 2016). 12
Şekil 10. Merkezileştirilmiş UDDI sistem mimarisinin tasarımı (Yongxin & Qin, 2016) 14
Şekil 11. NVivo 10 yazılımı tarafından oluşturulan ağırlıklı odak bulutu (Niknejad, 2020) 15
Şekil 12. SQL Enjeksiyon ile yapılan bir saldırı örneği. 21
Şekil 13. Boş bir Zarf ve SOAP versiyon 1.2'den oluşur. 24
Şekil 14. Muhasebe'ye web servis güvenliği üzerinden gönderilen zarf. 25
Şekil 15. Finans'sa web servis güvenliği üzerinden gönderilen zarf. 25
Şekil 16. İstemci tarafından iletilecek satış faturası Zarf. 26
Şekil 17. İstemci tarafından iletilen satış faturası sonuç sorgulama Zarf. 26
Şekil 18. E-imza tanımlamasında bulunan ve Body tagı ile referans edilmiş ds:Referans. 26
Şekil 19. Timestamp zaman aralığının belirlenmesi. 27
Şekil 20. Tekil paket sorgulama. 28
Şekil 21. Klasik referans tanımlanması kullanımı. 36
Şekil 22. Gelişmiş referans tanımlanması kullanımı. 37

.

KISALTMALAR

API : Uygulama Programlama Arayüzü

BS : Bilgi Sistemi

BT : Bilgi Teknolojisi

HP **:** Hewlett-Packard

HTTP : Hiper-Metin Aktarım İletişim Protokolü

HTTPS : Güvenli Hiper Metin Aktarım İletişim Protokolü

HSM : Donanım Güvenlik Modülü

IBM : Uluslararası İş Makineleri Merkezi

JAVA : Programlama Dili

JAX-RS : Jakarta RESTful Web Servis

JAX-WS : Jakarta XML Web Servis

MathML : Matematiksel Biçimlendirme Dili

Microsoft : Microsoft Corporation

OASIS : OASIS Konsorsiyum

REST : Temsili Durum Transferi

RPC : Uzaktan yordam çağrısı

RSS : Zengin Site Özeti

SGML : Standart Genelleştirilmiş İşaretleme Dili

SOA : Servis odaklı mimari

SOAP : Basit Nesne Erişim Protokolü

UDDI : Evrensel Tanımlama, Keşif ve Entegrasyon

UTC : Eşgüdümlü Evrensel Zaman

W3C : World Wide Web Consortium

WS : Web Servis

WSDL : Web Servisleri Tanımlama Dili

XML : Genişletilebilir İşaretleme Dili

XPATH : XML dokümanındaki bilgiyi bulmak için kullanılan bir dildir.

XSD : XML Şema Tanımı

XSLT : Genişletilebilir Biçimlendirme Dili Dönüşümleri

X.509 : Açık anahtar sertifikalarının formatını tanımlayan bir standarttır

1. GİRİŞ

1.1 Web Servis

Web servis, bir ağ üzerinden makineler arası birlikte çalışabilen ve etkileşimi desteklemek için tasarlanmış bir yazılım sistemidir. (Debasish & Sristy, 2017). XML, SOAP, WSDL, UDDI gibi açık standartları kullanarak, internet protokolü omurgası üzerinden web tabanlı uygulamaları bütünleştiren bir hizmettir. XML verileri işaretlemek, SOAP verileri aktarmak, WSDL mevcut hizmetleri tanımlamak ve UDDI kullanılabilir hizmetleri listelemek için kullanılmaktadır (Haviluddin, Edy, & Nur, 2019; Debasish & Sristy, 2017). Web servis çözümlerinde açık standartlar kullanılmasından dolayı yazılım ve işletim sistemi bağımsızdır. İnternet veya intranet üzerinden farklı uygulamalar arasında iletişim kurmak için ortam sağlayan bir uygulamadır. Gevşek bağlantılıdır, dinamiktir ve platformlar arası ortamı destekler (Debasish & Sristy, 2017). Web servislerde veri değiş tokuşu XML tabanlı olan SOAP protokolü üzerinden yapılır. SOAP iletişim protokolü OASIS, Microsoft, IBM, HP ve vb. konsorsiyumlar, firmalar ve kurumlar tarafından desteklenmektedir. SOAP protokol geliştirmeleri ve dokümanları bu konsorsiyumlar tarafından yayınlanmaktadır. Günümüz de popüler olarak kullanılan iki tür web servisi vardır. Bunlar SOAP ve REST tabanlı web servisler olarak belirtilmektedir.

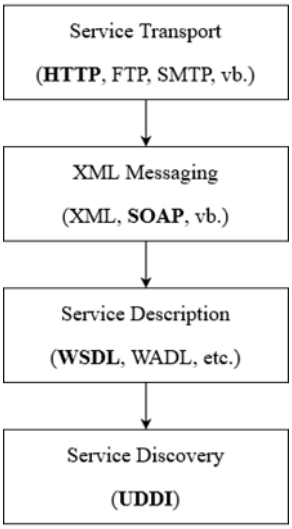
SOAP web servislerinin birbirleriyle nasıl etkileşime girebileceğini tanımlayan bir protokoldür. SOAP protokolü XML tabanlıdır. Kesin olarak belirlenmiş katı standartları mevcuttur. REST web servis çözümleri SOAP web servisleri gibi katı standartları takip etmemektedir. Ancak Roy Fielding tarafından tezinde de açıkladığı kısıtlamaları takip eder (Roy, 2000). SOAP tabanlı web servislerin dilden ve platformdan bağımsız olmasından dolayı, güvenlikleri WS-Security'de tanımlanmaktadır.

REST, web servisleri geliştirmek için bir mimari tarzdır. REST web servisinde, istemci kaynakları kullanarak istek gönderir ve sunucu istemciye yanıt verir fakat yanıt verme zorunluluğu yoktur, veri tek yönlü olarak iletilebilir. Bu iletişim standart bir yaklaşım olmamasından dolayı tek taraflıdır. REST web servisleri herhangi bir protokole bağlı değildir, ancak hemen hemen her REST servisi, kaynaklar üzerindeki işlemleri

düzenlemek için kullanılan HTTP yapısının kullanımını sağlar. REST web servislerinde, veri transferinde kaynaklar; videolar, web sayfaları, resimler gibi bilgisayar tabanlı sistemde izin verilen her şey olabilir. SOAP'ta Envelope kullanılmaktadır fakat REST'te böyle bir zorunluluk yoktur. REST servisler SOAP servislere göre daha hızlıdır fakat daha güvensizdir.

Her iki çözümde de iletişim için çok hızlıdır, ancak bu ikisi arasında küçük bir fark vardır. SOAP, REST 'in web servislerini geliştirmek için mimari bir stil olduğu bir protokolü takip eder. JAX-RS ve JAX-WS, sırasıyla REST ve SOAP için Java API'leridir. Her iki hizmetin de operasyonun karakteristiğine göre avantajları ve dezavantajları vardır. Bu nedenle, hangi durumda daha iyi sonuçlar elde etmek için hangi hizmetin kullanılabileceğini bulmak daha iyidir (Haviluddin, Edy, & Nur, 2019).

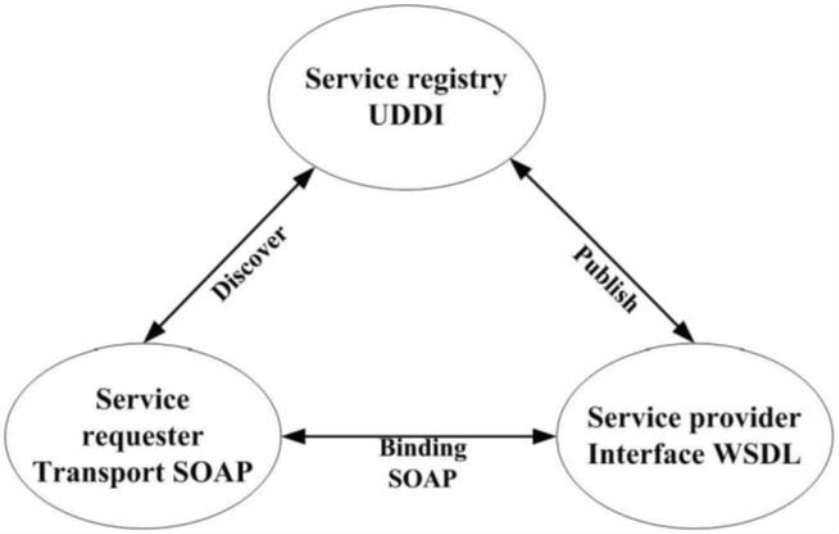
Web servisler, farklı platformlarda çalışan uygulamaları bir araya getirerek veri alışverişini mümkün kılar. Web servislerin platform bağısız olması bu sistemin en büyük tercih nedenlerinden birisidir. Web servislerin platform bağımsız olarak çalışabilmesinin en büyük nedeni XML, SOAP, WSDL ve UDDI standartlarını kullanmasıdır.



Şekil 1. Web Servis Mimarisi (Haviluddin, Edy, & Nur, 2019)

Web servis mimarisi **Şekil 1**'deki gibi sırasıyla HTTP, SOAP, WSDL ve UDDI şeklinde sırasıyla islenmektedir. Bu süreç günümüzde kullanılan tüm güncel servis türleri için geçerlidir.

Web servis modelinin standart uygulaması, hizmet sağlayıcının, hizmet açıklamasını tanımlayan ve ardından açıklamayı hizmet kayıt defterinde yayınlayan bir ağ yazılım modülü aracılığıyla erişilebilir olmasıdır. Hizmet talep eden, hizmet açıklamasını almak için hizmet kayıt merkezini bulur, bilgileri hizmet sağlayıcı ile olan bağlantıyı tamamlamak için kullanır, etkileşimini sağlar ve işlemleri çağırır. Web servis sisteminde, tüm uygulamalar, üç rol ve üç tür işlem içeren hizmetlere soyutlanır. Aşağıdaki şekilde Web Servislerinin mimarisini göstermektedir (Yongxin & Qin, 2016).



Şekil 2. Web Hizmetleri Mimarisi (Yongxin & Qin, 2016)

Web hizmetleri mimarisi **Şekil 2**'deki gibi hizmet sağlayıcısı, hizmet talep eden ve servis kayıt merkezi birbirleri ile olan bağlantı verilmiştir.

Hizmet sağlayıcı: ticari açıdan bakıldığında, hizmet sağlayıcı, hizmetlerin geliştirilmesinde ticari bir varlıktır; Mimari açısından hizmet sağlayıcı, yönetilen erişim hizmetleri için bir platform veya hizmet çalıştıran bir ortamdır (Yongxin & Qin, 2016).

Hizmet talep eden: mimari açıdan bu, hizmetleri bulmak ve çağırmak için bir uygulamadır (Yongxin & Qin, 2016).

Servis kayıt merkezi: Servis sağlayıcılar kendi tarif ettikleri servisleri bunun üzerinde yayınlar (Yongxin & Qin, 2016).

Yayınlama: Hizmeti kullanılabilir hale getirmek için hizmet sağlayıcıların, hizmet kullanıcılarının bulabilmesini sağlamak için hizmet açıklamasını yayınlaması gerekir.

Sorgulama: Sorgulama işleminde, hizmet talebinde bulunan kişi doğrudan hizmet tanımını alabilir veya hizmet kayıt merkezinde gerekli hizmet türünü sorgulayabilir.

Bağlama: bağlama işleminde, hizmet talebinde bulunan kişi, hizmetleri bulmak, iletişim kurmak ve çağırmak için hizmet açıklamasıyla ilgili bağlama ayrıntılarını kullanır, böylece çalışma zamanında hizmetle etkileşimi başlatır (Yongxin & Qin, 2016).

1.2 XML

XML, HTML'le çok benzeyen bir dildir. HTML'den çok daha esnektir çünkü kendi özel etiketlerinizi oluşturmanıza olanak tanır. Ancak, XML'in sadece bir dil olmadığını anlamak önemlidir. XML bir meta dildir: diğer dilleri yaratmamıza veya tanımlamamıza izin verir. Örneğin; XML ile RSS, MathML (bir matematiksel biçimlendirme dili) gibi başka diller ve hatta XSLT gibi araçlar oluşturabiliriz (Prof., 2017).

1970 yılında IBM, SGML'yi (Standart Genelleştirilmiş İşaretleme Dili) tanıttı. SGML, 1960'ların sonlarında IBM tarafından geliştirilen Genel İşaretleme Dili'nden (GML) geliştirilmiştir. SGML, metin belgeleri için anlamsal ve yapısal bir dildir ancak çok karmaşıktır. HTML, SGML'nin bir alt kümesidir (Prof., 2017).

1996 yılında W3C altında XML çalışma grubu kuruldu. World Wide Web Konsorsiyumu (W3C), üye kuruluşların, tam zamanlı bir personelin ve halkın Web standartlarını geliştirmek için birlikte çalıştığı uluslararası bir konsorsiyumdur. W3C, 1989'da World Wide Web'i de icat eden Tim Berners-Lee tarafından 1994'te oluşturuldu. 1998'de W3C, XML 1.0'ı tanıttı (Prof., 2017).

```
<?xml version="1.0" encoding="UTF-8"?>
- <Kullanicilar>
  - <Kullanici>
      <Id>634f7472-f1bd-4ce1-b17e-5508f90d4e6f</Id>
      <Ad>Ahmet</Ad>
      <Soyad>Ayaz</Soyad>
    </Kullanici>
  - <Kullanici>
      <Id>3fd0b105-936b-4415-ad48-09dafe421e3d</Id>
      <Ad>Hsan</Ad>
      <Soyad>Tok</Soyad>
    </Kullanici>
</Kullanicilar>
```

Şekil 3. İki kullanıcı bilgisinin bulunduğu basit bir XML dosyası

XML dosyası **Şekil 3**'te ki gibi kullanıcılar listesi içerisinde iki farklı kullanıcını Id(tekil numatası), Ad ve Soyad bilgilerinin tutulduğu Ahmet Ayaz ve Hasan Tok bilgileri bulunan bir örnektir.

XML, yazılımlar arası veri transferi yapmak için kullanılan ve hiyerarşik bir yapıya sahip olan bir yazım dilidir. XML oluşturulmasında serialize ve deserialize işlemi çok sıklıkla kullanılır hale gelmiştir. Her XML'in XMLSchema özeliği vardır. XML yapısı ve veri tipi için XMLSchema kullanılır.

```
<?xml version="1.0" encoding="UTF-8"?>
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  - <xs:element name="Kullanicilar">
    - <xs:complexType>
      - <xs:sequence>
        - <xs:element name="Kullanici" maxOccurs="unbounded">
          - <xs:complexType>
            - <xs:sequence>
                <xs:element name="Id" type="xs:string"/>
                <xs:element name="Ad" type="xs:string"/>
                <xs:element name="Soyad" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

Şekil 4. Id, Ad ve Soyad bilgileri bulunan basit bir XMLSchema dosyası

XMLSchema **Şekil 4**'te oluşturulacak XML dosyasının yapı taşlarını belirlemektedir. XML'lin Kullanıcılar baş elemanından oluşarak, içerisinden birden fazla Kullanıcı bilgisinin oluştuğu ve her kullanıcısının tekil bilgileri Id, Ad ve Soyad bilgilerinden oluştuğu ve alfanumerik berisinin girilebileceğini açıklamaktadır.

Schematron, bir XML doğrulama dilidir. Belgeleri XML kalıplarının varlığına veya yokluğuna göre doğrulayan kuralları tanımlar. Bu kurallar kısadır, basittir ve kullanıcı dostu mesajların yazdırılmasına izin verir. Schematron'un sözdizimi nispeten basittir. Doğrulama yeteneği kurallara dayanmaktadır. Schematron modeli en az bir tane kural içermelidir. Her kuralın, iddiaları çalıştırdığı bir bağlamı olmalıdır. Schematron iki tür iddia içerir olumlu (iddia) ve olumsuz (iddia-raporu) olarak sınıflandırılır. Bir kuralın herhangi bir iddiası başarısız olursa kural başarısız olur ve belge geçersiz olarak rapor ile işaretlenir. En önemli özeliği de Schematron doğrulana bilgisinin de kuralların XML içerisinde tanımlanmış olmasıdır.

Schematron temel görevi; veri bağımlılıklarını doğrulamak, veri kardinalitesini kontrol etmek ve veri algoritmasını kullanır. Kullanıldığı alanlar; finans, sigorta, devlet kurumları ve teknik yayıncılar olarak örnek verilebilir.

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <pattern>
      <title>Tarih kuralları</title>
      <rule context="Contract">
         <assert test="ContractDate &lt; current-date()">
            Sözleşme Tarihi geçmişte olmalıdır çünkü gelecekteki sözleşmelere izin verilmez.
         </assert>
      </rule>
   </pattern>
</schema>
```

Şekil 5. Tarih kontrolü içeren basit bir Schematron dosyası

Schematron **Şekil 5**'te anlatılmak istenilen bir tarih kuralıdır. Buradaki kontrol Sözleşme tarihinin işlem yapılacak bir olayın sözleşme tarihinden önce olmaması kuralıdır. Örnek vermemiz gerekirse; Cep telefonu numarası almadan ve sözleşme imzalanmada o cep telefona fatura kesilmemesi gerektiğinden eğer böyle bir eylem gerçekleştirilmek istenirse buna sistem izin vermemesi sağlanmış olur ve kontroller için maliyet azaltılmasına gidilmiş olur.

Schematron en büyük amaçlarından bir tanesi de iletilecek verilerin talep edilen şekilde olup olmaması ve veri kirliliğin azaltılmasını sağlar.

1.3 SOAP

SOAP web servis, uygulamalar arasında XML formatında iletişim için kullanılan bir protokoldür. Standart ve daha etkili bir web iletişim yolu ve kurumsal web

uygulamalarının diğer sistemler veya kuruluşlarla daha esnek bir şekilde iletişim kurması için kullanılır. SOAP'ın en büyük dezavantajı ağır mesaj alışverişi ve düşük performans sağlayan yüklü işlevleridir. Performansını iyileştirmek, onu en çok kabul gören iletişim çözümlerinden biri haline getirebilir (Abdul & Ahmed , 2017). SOAP gövde bloğunda teslim edilecek gerçek bildirim mesajını içerir.

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope">
    <env:Header>
        <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
            <n:priority>1</n:priority>
            <n:expires>2022-01-15T14:00:00-05:00</n:expires>
        </n:alertcontrol>
    </env:Header>
    <env:Body>
        <m:alert xmlns:m="http://example.org/alert">
            <m:msg>Berker'i saat 2'de okuldan al</m:msg>
        </m:alert>
    </env:Body>
</env:Envelope>
```

Şekil 6. Basit bir SOAP Zarfı

SOAP Envelope **Şekil 6**'te gösterildiği gibi Header ve Body kısımlarından oluşmaktadır. Header kısmında birinci özelik seviyesi 1 olarak ve son Envelope'un son kullanım tarihi verilmiştir. Body kısmında ise son kullanım tarihine kadar yapılması gereken olay bilgisi verilmiştir. Bu olay da Berker'in saat 2'de okuldan alınması işlem talimatıdır.

1.3.1 Zarf (Envelope)

SOAP mesajında ne olduğunu ifade etmek için genel bir çerçeve tanımlar. Kiminle iletişime geçecek veya zorunlu olup olmadığı ile ilgili bilgiler bulunur. Envelope SOAP'ın en dış katmanıdır. Header, Body ve sunucu tarafından iletilen hatalı iletiler için Fault parçalarını da barındırır.

1.3.1.1 Başlık (Header)

Header parçası, SOAP'ın ilk alt öğesi olarak kodlanmıştır. Başlık ögesinin tüm acil alt öğeleri başlık girişleri olarak adlandırılır. Bu kısımda metadata denilen belge bilgi bulundurulur.

1.3.1.2 Gövde (Body)

Body kısmı SOAP mesajındaki en önemli kısmı oluşturur. Bu kısım web servisteki metotların adları, parametreleri ve geri dönüş değerleri gibi hayati öneme sahip bilgileri taşımaktadır. Mesajın alıcısına yönelik zorunlu bilgi alışverişinde bulunmak için basit bir mekanizma sağlar. Header parçasının tipik kullanımları arasında marshalling RPC çağrıları ve hata raporlamayı barındırır.

1.3.1.3 Hata (Fault)

Sunucu tarafından iletilen hata mesajıdır. İçerisinde detail(detay), reason(sebep) ve code(kod) içerisinde hata mesajları döndürür.

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header>(...)
    </soapenv:Header>
    <soapenv:Body>
        <soapenv:Fault>
            <soapenv:faultcode>soapenv:Client</soapenv:faultcode>
            <soapenv:faultstring>Varlık vEntityName Kaydı işlenirken hata zaten var.</soapenv:faultstring>
            <soapenv:detail>
                <ns1:ExceptionInfoList xmlns:ns1="http://schemas.myapp.net/functions">
                    <ns1:ExceptionInfo functionid="MP0085">
                        <ns1:Exception name="com.myapp.EntityAlreadyExistsException">
                        <ns1:ReasonCode>1</ns1:ReasonCode>
                        <ns1:Message>Zaten var.</ns1:Message>
                        </ns1:Exception>
                    </ns1:ExceptionInfo>
                </ns1:ExceptionInfoList>
            </soapenv:detail>
        </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

Şekil 7. Basit bir hatalı bilgisi içeren SOAP Zarfı

Sunucu tarafından iletilen **Şekil 7**'da hata mesajında hata kodu olarak soapend:Client, hata açıklaması bilgileri Varlık vEntityName Kaydı hata zaten ver bilgisi verilmektedir. Ayrıca detay açıklama fonksiyon bilgisi olarak MP0085, hatanın kaynağı com.myapp.EnttiyAlreadyExistsException, hatanın sebep kodu 1 ve hata mesajı Zaten var bilgisi ile hata mesajında hata bilgisi ve detayı en açık bir şekilde sunucu tarafından

iletilmiştir. Her hata mesajının yapısı aynı olmamakla beraber Body kısmı vazgeçilmez ve zorumlu Envelope için vazgeçilmez bir durum olarak karşımıza çıkmaktadır.

1.4 WSDL

Web Servisleri Tanımlama Dili (Web Services Description Language (WSDL)), bir web hizmetini tanımlamak için bir XML gösterimidir. Bir WSDL tanımı, bir istemciye bir web hizmeti isteğinin nasıl oluşturulacağını söyler ve web hizmeti sağlayıcısı tarafından sağlanan arabirimi tanımlar. SOAP ve REST servislerde WSDL kullanılmaktadır. Web servislerde WSDL metadata olarak kullanılmaktadır. Her SOAP web servisin WSDL dosyası olma zorunluluğu yoktur fakat olması istemci tarafından yapılacak istekler için bir gösterge olarak tanımlanır. Sunucu tarafında bulunan WSDL sayesinde sadece istenilen veya izin verilen bilgilere ulaşım, WSDL özelleştirilmesi ile sağlanabilmektedir. Günümüzde modern uygulamalar web servis yapısına göre otomatik olarak WSDL dosyasını oluşturmaktadır.

WSDL dosyalarının esas kullanım amacı bir web servisi tarif etmektir. Web serviste bulunan fonksiyonlar, bu fonksiyonların aldığı parametreler, bu parametrelerin tipleri ve alabilecekleri değerler gibi bütün bilgiler WSDL dosyalarında mevcuttur. WSDL dosyalarının web servisin bir ara yüzü olarak tanımlanabilir (Chinnici R. e., 2007).

İletişim protokolleri ve mesaj formatları web topluluğunda standartlaştıkça, iletişimleri yapısal bir şekilde tanımlayabilmek giderek daha mümkün ve önemli hale getirmektedir. WSDL, ağ hizmetlerini mesaj alışverişi yapabilen iletişim uç noktaları koleksiyonları olarak tanımlamak için bir XML dilbilgisi tanımlayarak bu ihtiyacı giderir. WSDL hizmet tanımları, dağıtılmış sistemler için belgeler sağlar ve uygulama iletişiminde yer alan ayrıntıları otomatikleştirmek için bir reçete görevi görür.

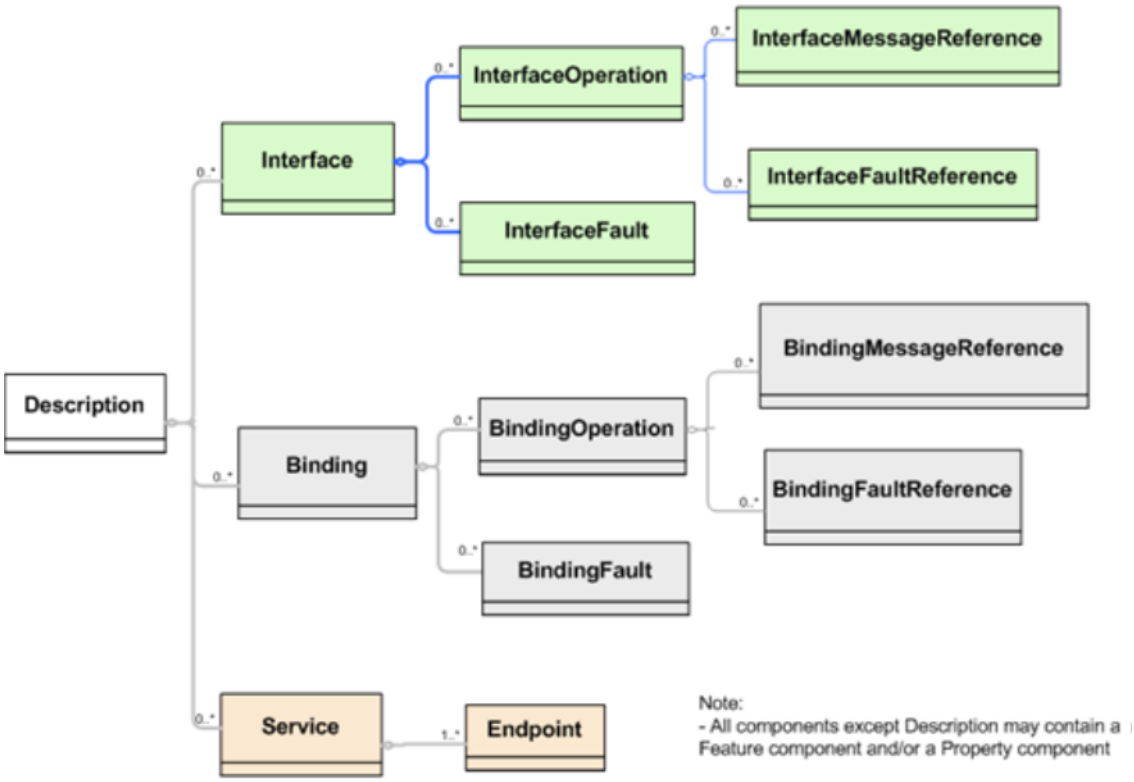
WSDL belgesi, hizmetleri ağ uç noktaları veya bağlantı noktaları koleksiyonları olarak tanımlar. WSDL de, uç noktaların ve mesajların soyut tanımı, somut ağ dağıtımlarından veya veri formatı bağlamalarından ayrılır. Bu, soyut tanımların yeniden kullanılmasına izin verir. Değiş tokuş edilen verilerin soyut açıklamaları olan mesajlar ve soyut işlem koleksiyonları olan belirli bir bağlantı noktası türü için somut protokol ve veri biçimi belirtimleri, yeniden kullanılabilir bir bağlama oluşturur. Bir bağlantı noktası, bir ağ adresini yeniden kullanılabilir bir bağlamayla ilişkilendirerek tanımlanır ve bir bağlantı

noktası koleksiyonu bir hizmeti tanımlar. Bu nedenle, bir WSDL belgesi, ağ hizmetlerinin tanımında aşağıdaki öğeleri kullanır:

- Türler (Types): bazı tür sistemlerini (XSD gibi) kullanan veri türü tanımları için bir kapsayıcı.
- Mesaj (Message): iletilen verilerin soyut, yazılı tanımı.
- Operasyon (Operation): hizmet tarafından desteklenen bir eylemin soyut bir açıklaması.
- Bağlantı Noktası Türü (Port Type): bir veya daha fazla uç nokta tarafından desteklenen soyut bir işlemler kümesi.
- Bağlama (Binding): belirli bir bağlantı noktası türü için somut bir protokol ve veri biçimi belirtimi.
- Bağlantı noktası (Port): bir bağlama ve bir ağ adresinin birleşimi olarak tanımlanan tek bir uç nokta.
- Servis (Service): ilgili uç noktaların bir koleksiyonu (E., F., G., & S., 2001).

W3C tarafında oluşturulan WSDL in temel amacı web servislerde bir standart oluşturulmasıdır. WSDL tıpkı XML de olduğu gibi hiyerarşik (parent-children) bir yapıya sahiptir ve kendi içerisinde belirli gruplar halindedir. Her grubun kendine has bir işlevi bulunmaktadır (Booth, 2007).

Aşağıdaki **Şekil 8**'deki şema WSDL bileşenlerinin hiyerarşik yapısına genel bakışı göstermektedir.

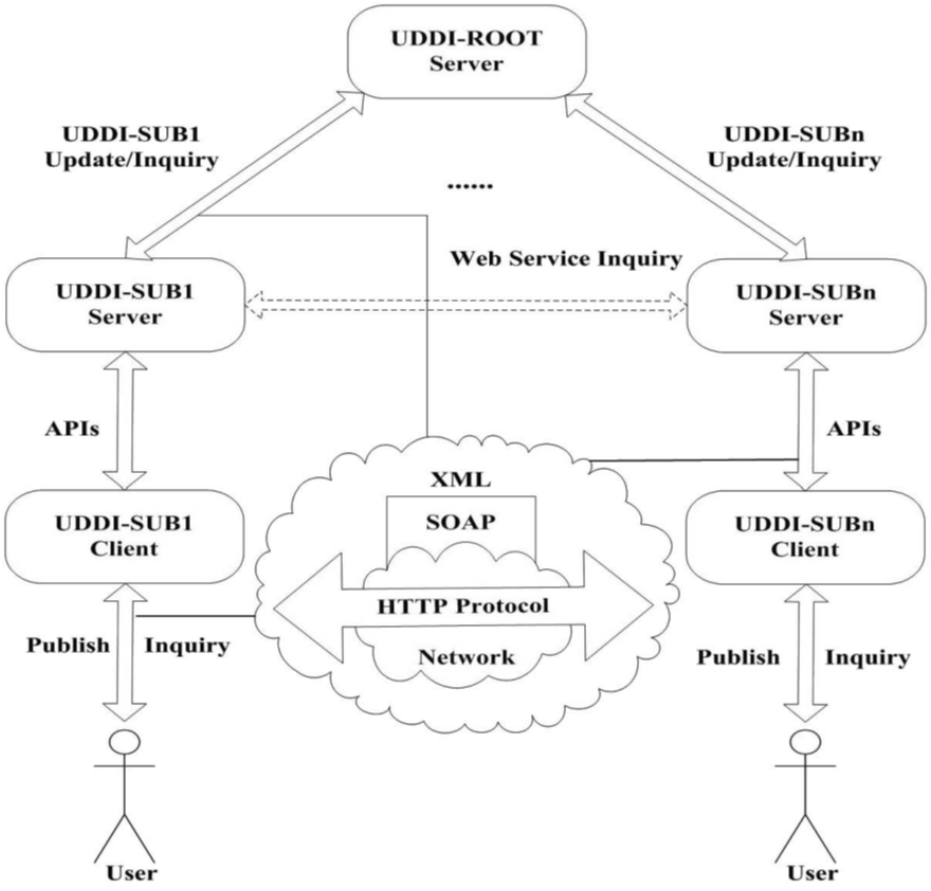


Şekil 8. WSDL Bileşenlerinin Hiyerarşisi (Booth, 2007).

1.5 UDDI

Evrensel Tanımlama, Keşif ve Entegrasyon (Universal Description, Discovery, and Integration (UDDI)) kayıt merkezidir. Büyük miktarda veri için paralel işleme yeteneğini

artıran ve ayrıca verilerin doğruluğunu sağlayan hizmet sağlayıcı ve hizmet talep eden arasında hayati bir bağlantıdır. UDDI, SOA'nın önemli bir bileşenidir.



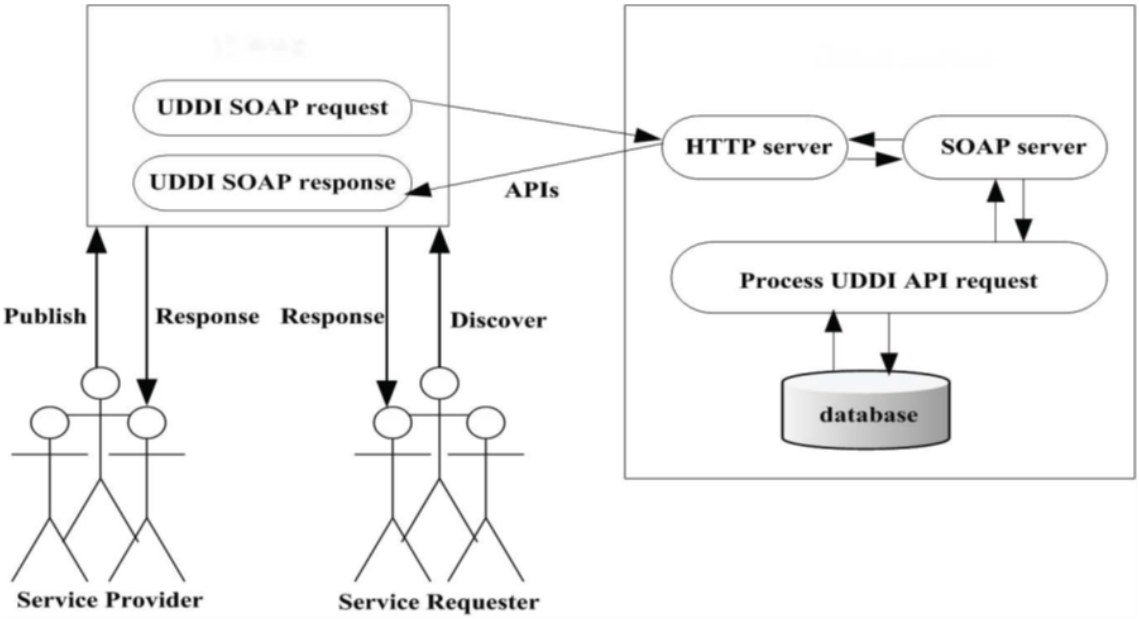
Şekil 9. Dağıtılmış UDDI sisteminin bilgi etkileşimi (Yongxin & Qin, 2016).

Kullanıcılar bir yan kuruluş UDDI-SUB(n) ile İstemci tarafı kayıt sisteminde oturum açarlar, UDDI-SUB(n) İstemcisinin hizmet yönetim modülü, kapsüllenmiş talebi XML biçimindeki SOAP mesajına paketler, daha sonra açılış ile Web Hizmetini yayınlar veya sorgular ve ardından iletir. HTTP protokolü aynı zamanda, UDDI-SUBI İstemcisi, UDDI-SUB 1 Sunucu düğümünün API'lerini çağırır veya UDDI-SUB 1 Sunucusu, UDDI-ROOT Sunucusuna sorgulama hizmeti talebini başlatır. UDDI-ROOT Sunucusu ayrıca HTTP protokolü aracılığıyla SOAP isteği iletir ve paketleri yanıtlar (Yongxin & Qin, 2016).

UDDI, WSDL dosyasının URI'sini sağlayarak işletmelerin, kuruluşların ve diğer web hizmetleri sağlayıcılarının hizmetleri keşfetmelerine ve bunlara erişmelerine yardımcı olur. Bu durum, kalitesine göre bir web hizmeti seçmek için bir mekanizma sunmaz. Ayrıca standart web servislerinin içeriğinde yeterli semantik tanımlamadan yoksundur, bu eksiklik analiz, arama ve eşleştirme süreçlerinde uygun web servislerinin bulunmasını ve oluşturulmasını zorlaştırmaktadır. Ek olarak, merkezi bir UDDI, tek bir merkezi nokta sorunu ve yüksek bakım maliyetinden muzdariptir.

Web servislerin ana teknik desteği olan Evrensel Tanımlama, Keşif ve Entegrasyon (UDDI), SOA'da (Hizmet Odaklı Mimari) giderek daha önemli bir rol oynamaktadır. Blok zincirine dayalı akıllı sözleşme teknolojisi, UDDI kayıtları arasında veri çoğaltma ve paylaşma sorununu çözmek için kullanılır. UDDI kaydı, yalnızca basit hizmet tanımı ve blok zincirine referans bilgilerini depolar ve kimlik yönetimi, yetki yönetimi ve diğer işlevleri sağlar. Servis sağlayıcının ve sunduğu hizmetlerin detaylı bilgileri blok zincirinde saklanır. UDDI kaydının maliyeti büyük ölçüde azalır ve verilerin paylaşımı ve depolanması kolayca gerçekleştirilebilir. Bu arada, verilerin güvenliği ve denetlene bilirliği, blok zincirinin özellikleri tarafından garanti edilir (Zhao, 2018).

Şu anda, kayıt sistemlerinin uygulanması çoğunlukla merkezi bir tasarıma dayanmaktadır. Bir sunucu ve bir istemciden oluşur ve tüm ağ boyunca kayıtlı tüm hizmet bilgilerini sunucu veri tabanında saklar. Merkezileştirilmiş UDDI sistem mimarisi aşağıda **Şekil 10**'da gösterilmektedir.



Şekil 10. Merkezileştirilmiş UDDI sistem mimarisinin tasarımı (Yongxin & Qin, 2016)

Bu sistemde, hizmet sağlayıcı ve hizmet talebinde bulunan kişi genellikle hizmet yayını veya hizmet keşfini işlemek için UDDI Sunucusu tarafından sağlanan SOAP API'sini çağırması gereken UDDI İstemcisinde hizmet yayını veya hizmet bulma işlemlerini uygular. UDDI Sunucusu, API UDDI Sorgulama, API UDDI Yayını, API UDDI Güvenliği, API UDDI Gözetim Aktarımı, API UDDI Aboneliği ve UDDI Aboneliği UDDI Çoğaltma dahil olmak üzere altı SOAP API'sinde uygulanabilir. Bu altı çeşit API işlevinde en yaygın olarak kullanılanlar UDDI Security, UDDI Publication ve UDDI Inquiry'dir (Feng).

1.6 Servis Odaklı Mimari (SOA)

Yazılım dünyasında, Servis Odaklı Mimari (SOA) hizmetlerden oluşur ve tipik olarak İnternet veya İntranet üzerinde diğer bileşenlere koordineli bir şekilde hizmet eder. SOA'nın hizmet verdiği temel yapı taşları servislerdir. Bu servisler birbirleri ile mimarisi gereği gevşek bağlıdır. SOA en büyük özeliği yazılım ve platform bağımsız olmasıdır. Günümüzde SOA mimarisini desteklemeyen modern yazılım dili kalmamıştır. Kullanılan yazılım dilleri (C#, Java ve vb.) olarak örnek verilebilir. Cross Platform ile işletim sistemi bağımlılığı SOA tarafından bağımsız bir şekilde çalışılması sağlanmıştır. İşletim sistemleri olarak da günümüzde en popüler kullanılan (Windows, Linux, macOS ve vb.) işletim sistemlerini sayabiliriz. SOA mimarisi devlet kurumları, özel/tüzel kurumlarda

hatta kullandığımız cep telefon uygulamalarında da vazgeçilmez bir mimari olarak karşımıza çıkmaktadır. Kurumlar için eski uygulamaların SOA mimarisine geçişi sırasında yazılım dili ve işletim sistemi bağımlılığı olmamasından maliyet, hız ve süre problemlere bu anlamda SOA tarafından çözümler sunulmuştur. Yazılım dünyasında ve insanlar için hız ve sonuç kavramı önemli olmasından vazgeçilmez bir mimari olarak günümüzde kullanılmaya devam etmektedir. SOA mimarisini kullanmayan yazılım firması yok denilecek kadar az duruma gelmiştir. Kurumlar için SOA mimarisi her zaman avantajlı hale gelmemiştir, bazı bakım durumlarında da dezavantajlar gözlemlenmiştir, bu konuda örnek olarak hizmet bakımlar örnek olarak verilebilir.



Şekil 11. NVivo 10 yazılımı tarafından oluşturulan ağırlıklı odak bulutu (Niknejad, 2020)

Kuruluşların yazılım sistemini çağdaşlaştırmak için eski bir sistemden SOA tabanlı bir sisteme geçiş ana akım bir trend haline geldi (Abdellatif, 2018; Gupta, 2018). Birçok çalışmada Nesnelerin İnterneti (IOT), Bulut Bilişim (CC) ve Mikro Servisler (MS) gibi dünya lideri yeni teknolojilerin geliştirilmesinde SOA'nın kullanılmasının faydalarını vurgulanmıştır. Bunun nedeni, SOA'nın hizmet tabanlı modüler mimarisi nedeniyle esnek entegrasyon ve hizmetlerin yeniden kullanılabilirliği sağlamasıdır. SOA, birçok uygulamayı ve veri kaynağını bir kara kutu biçiminde kapsadığı için şeffaflık da sunmaktadır. Bu şekilde, çeşitli teknolojilerin, dil kodlarının, işlevlerin ve platformların varlığına rağmen, bütünleşmiş bir Bilgi Teknolojisi (BT) kaynakları havuzuna hala erişilebilir durumdadır (Niknejad, 2020).

Endüstriyel sektörlerle ilgili olarak, şimdiye kadar, SOA'nın bankacılık, sağlık, ulaşım vb. gibi birçok sektörde kilit bir paradigma olduğu kanıtlanmıştır. Birçok faydasının yanı sıra, tanımlanan birincil çalışmaların bazıları, kuruluşların gerçekleştiremediğini ortaya

koymuştur. SOA'nın benimsenmesinin tüm faydaları, çeşitli nedenlerden dolayı. Pek çok engel arasında, SOA'nın benimsenmesi ve uygulanması için kritik başarı faktörleri hakkında bilgi ve belge eksikliği başarısızlığın temel nedenleriydi (Sima & Raziye, 2013). Ancak, başarılı SOA'nın benimsenmesi ve uygulanması için kritik etkili kriterleri paylaşan ayrıntılı bir sistematik çalışma yoktur. Bu boşluğu doldurmak için, kuruluşlarda SOA'nın benimsenmesinin önemli faktörlerinin araştırılmasının hayati olduğuna inanmışlardır, çünkü bu faktörlerin anlaşılması, bu kuruluşların SOA uygulamasının faydalarını en üst düzeye çıkarmasına yardımcı olacaktır (Niknejad, 2020).

Birçok araştırmacı SOA'yı farklı bakış açılarına göre (teknoloji, işletme ve mimari gibi) tanımlamıştır, dolayısıyla bu terimin kesin bir tanımı yoktur. SOA, BT'nin karmaşıklığının üstesinden gelebilecek bir teknoloji, ürün veya hızlı bir çözüm değildir. Ayrıca SOA, mevcut tüm Bilgi Teknolojisi (BT) / Bilgi Sistemi (BS) zorluklarını ele alabileceğini garanti etmez. Yine de SOA'nın bir bütün olarak işletmelerde, BT'de, BS'de ve işletmelerde avantajlı olarak kullanılabilecek bir kavram olduğu söylenebilir (Marks, 2008).

Bir arayüz tanımı ile başlayan ve tüm uygulama topolojisini ara yüzler, ara yüz uygulamaları ve ara yüz çağrılarının bir topolojisi olarak oluşturan bir yazılım mimarisidir. SOA, hizmetler ve hizmet tüketicilerinin bir ilişkisidir, her iki yazılım modülü de eksiksiz bir iş işlevini temsil edecek kadar büyüktür (Natis, 2003).

SOA, kurumsal iş çözümlerini tasarlamanın, inşa etmenin ve oluşturmanın temel birimi olarak iş odaklı kurumsal hizmet kavramını destekleyen bir mimari tarz olarak tanımlanabilir (Lublinsky, 2007).

SOA, değişen iş önceliklerini ele almak için yeniden kullanılabilen ve birleştirilebilen güvenli, standartlaştırılmış bileşenler-hizmetler olarak iş süreçlerini entegre etmek ve BT altyapısını desteklemek için bir çerçevedir. SOA, sistemler arasında gevşek bağlantı, yeniden kullanım ve birlikte çalışabilirliği destekleyen kurumsal çapta bir BT mimarisidir (Bieberstein, 2006).

Servis yönelimi, yaptığınız şeyi çerçeveleyen bir paradigmadır. Servis yönelik mimari (SOA), hizmet yöneliminin uygulanmasından kaynaklanan bir mimari türüdür. Kuruluşların değişen iş ihtiyaçları doğrultusunda artan çeviklik ve maliyet etkinliği ile sürekli olarak sürdürülebilir iş değeri sunmalarına yardımcı olmak için hizmet odaklılık mimaridir (A. Arsanjani, 2009).

1.6.1 SOA faydaları

Küreselleşme, daha sıkı ekonomiler, iş süreçlerinde dış kaynak kullanımı, sürekli artan düzenleyici ortamlar ile bilgili tüketiciler, büyük işletmeleri işlerini ve hizmetlerini sağlama biçimlerini değiştirmeye zorlamaktadır. İşletmelerin çevik ve esnek olmaları gerekiyor ve BT yöneticilerinden mevcut BT yatırımından yararlanırken gelişmiş işlevsellik sağlamaları isteniyor. Bu ortamda SOA, Kurumsal Uygulama Entegrasyonu ve aradıkları diğer çözümler için çekici bir yaklaşım olduğunu kanıtlıyor. SOA, BT'nin iş ile daha iyi uyumlaştırılması, etkin yeniden kullanım, birlikte çalışabilirlik ve daha düşük geliştirme maliyetleri vaat ediyor. Bununla birlikte, herhangi bir yaklaşım gibi, sınırlamaları vardır ve bu nedenle birtakım zorluklara sahiptir.

Windows Servis (WS) teknolojisi ve SOA, azaltılmış maliyetler, daha kolay bakım, daha fazla esneklik ve iyileştirilmiş ölçeklenebilirlik gibi ek avantajlarla birlikte kurumsal uygulama entegrasyonu için daha iyi fırsatlar sunar. SOA, gevşek bağlı yapısıyla, işletmelerin yeni hizmetler eklemesine veya mevcut hizmetleri yükseltmesine ve iş çevikliğini artırma ve talep üzerine yanıt verebilme fırsatı sunar. İşletmelerin ve BT sistemlerinin iş ve çevre değişikliklere karşı daha çevik olmalarını sağlar.

SOA, değişen teknolojilere ve iş gereksinimlerine uyum sağlama esnekliği sunarken, geniş ölçekli birlikte çalışabilirlik elde etme fırsatı sağlar. Doğru şekilde uygulanırsa, SOA aşağıdaki faydaları sunar:

- Gevşek bağlı uygulamalar ve konum şeffaflığı
- Uygulama bağlantısı ve birlikte çalışabilirlik
- BT'nin işin ihtiyaçları etrafında hizalanması
- Mevcut varlıkların ve uygulamaların geliştirilmiş yeniden kullanımı

• Süreç merkezli mimari

• Paralel ve bağımsız gelişme

• Daha iyi ölçeklenebilirlik ve zarif evrimsel değişiklikler

• Uygulama geliştirme ve entegrasyon maliyetlerinin düşürülmesi

• Daha kolay bakım

• Azaltılmış satıcı kilitlenmeleri (Mahmood, 2007)

1.6.2 SOA sınırlamaları ve sorunları

SOA, dağıtılmış istemci sunucu mimarisine kıyasla çok daha iyi bir mimaridir ve kodun yeniden kullanımı, daha iyi entegrasyon ve iş gereksinimlerine daha iyi yanıt verme biçiminde büyük faydalar sağlayabilir. Bununla birlikte, esneklik ve verimlilik arasındaki sonsuz savaş, her zaman olduğu gibi aynı şekilde mevcuttur. SOA ayrıca teknoloji ve geliştirme yoluyla büyük bir ön yatırım gerektirir. Çok pahalıya mal olabilir ve yatırım getirisinin gerçekleşmesi uzun zaman alabilir (Overall, 2006).

SOA'nın aşağıda olumsuz yönlerine değinelim:

• Herhangi bir hizmet diğerini çağırabileceğinden, her hizmetin her giriş parametresini tamamen doğrulaması gerekir. Bunun yanıt süresi ve genel makine yükü açısından etkileri olacaktır.

• İyi kullanılan bir hizmette ortaya çıkan bir hata veya bozulmanın, yalnızca tek bir uygulamayı değil, tüm sistemi etkilemesi mümkündür.

Hizmet meta verilerini yönetmek, bir başka bariz zorluktur. Servisler, görevleri yerine getirmek için mesaj alışverişinde bulunmaya devam ettikçe, bu mesajların sayısı tek bir uygulama için bile milyonları bulabilmektedir. Web Servisleri mantıklı bir uygulama platformu sağlasa da birçok altyapı servisi (ör. güvenlik, sistem yönetimi, arayüz sözleşmeleri) henüz tam olarak tanımlanmamıştır. Doğru soyutlama düzeyinde bir hizmet bulmak da bir zorluktur (Mahmood, 2007).

1.7 Web Servis Güvenliği

İnternetin yaygınlaşmadı ve web servislerin hemen hemen her alanda kullanılması ile önemli ve gizli bilgilerin paylaşılması sırasında kullanılmasından dolayı güvenlik zafiyetleri oluşturmuştur. Kuruluşların veya bireylerin hizmetler arası işlevselliği yeniden kullanmalarına yardımcı olan bir ağ üzerinden farklı verileri aktarmanın bir yolunu sağlar. Web servisler çeşitli uygulamaları sağlık, bankacılık ve e-ticaret operasyonlarından oluşmaktadır. Web servislerinin geliştirilmesinde yer alan çeşitli protokoller ve çerçeveler nedeniyle, genellikle saldırılara karşı savunmasızdır. Bunları sırasıyla saymak gerekirse XML Enjeksiyonu (XML Injection), SQL Enjeksiyonu (SQL Injection), XPath Enjeksiyonu (XPath Injection), Ortadaki Adam Saldırısı (Man in The Middle Attack (MITM)), Hizmet Reddi (Denial of Service (DOS)) ve Sahtekarlığı (Spoofing) içerir. Web servis güvenliği için OASIS, Microsoft, IBM vb. konsorsiyumlar tarafından geliştirdikleri gözlemlenmiştir.

Günümüzde SOA yaklaşımını kullanan birçok uygulama geliştirme aşamasındadır. Bu yaklaşımın uygulanması, birbirleri arasında kolayca bilgi alışverişi yapmak için web servislerini kullanmaktır. Web servisler, kuruluşların veya bireylerin hizmetler arası işlevselliği yeniden kullanmalarına yardımcı olan bir ağ üzerinden farklı verileri aktarmanın bir yolunu sağlar. Web servislerinin geliştirilmesinde yer alan çeşitli protokoller ve çerçeveler nedeniyle, genellikle saldırılara karşı savunmasızdır. (Eriyanto & Fadhil, 2020). Bundan dolayı web servislerde güvenlik gün geçtikçe sorunlu hale gelmiştir.

1.8 Web Servis Güvenliğinde Saldırı Türleri

Günümüzde web servisler için pek çok saldırı türü vardır. Bu saldırı türlerini sırasıyla inceleyelim.

1.8.1 XML enjeksiyon

İstemci tarafında iletilen veri güvenlik protokolleri yok ile sunucuya iletilen XML verisinin değiştirilerek sunucu tarafından farklı algılanması ve farklı sonuçlara sonuçlanması anlamına gelmektedir. Genellikle bu saldırı durumları güvensiz web

servislerde yapılan bir saldırı türüdür. Bu saldırı sonucu ön görülmeyen sonuçlara neden olacaktır.

1.8.2 XML bombası

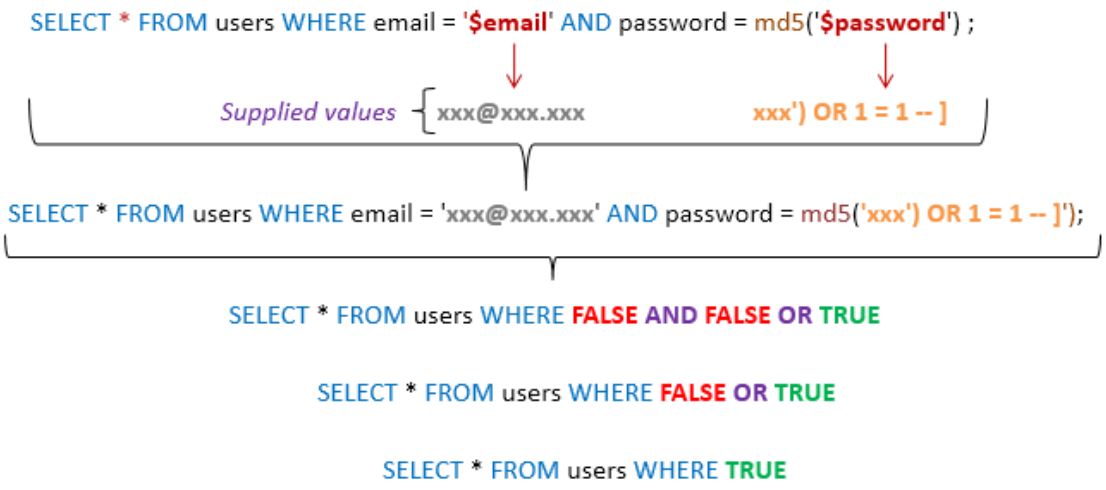
XML bombası, XML dosyalarını ayrıştıran programı ezmek amacıyla oluşturulan ve gönderilen küçük ama tehlikeli bir mesajdır. XML ayrıştırıcısı bir XML bombasını işlemeye çalıştığında, veriler kendi kendini besler ve katlanarak büyür. Bu, bir Web sitesini veya ISP'yi (İnternet servis sağlayıcısı) kapatabilir ve bilgisayar korsanları tarafından hizmet reddi saldırılarını gerçekleştirmek için kullanılan birçok yöntemden biridir.

1.8.3 SQL enjeksiyon

En yaygın web servis saldırıları SQL Enjeksiyon saldırısıdır ve bu saldırı yoluyla bilgisayar korsanları belirli bir web servisin veritabanı üzerinde yetkisiz kontrol elde etmeye çalışır ve tüm önemli veriler veritabanından alabilir. Web servisler tarafından kullanılan popüler verirabanlarından olan MySQL, Oracle, SQL Server gibi veritabanı kullanan web siteleri SQL Enjeksiyon güvenlik açığından etkilenebilir. Saldırganlar SQL sorgularını manipüle ederek tüm veritabanına erişebilir ve kullanıcıyla ilgili tüm önemli bilgileri alabilir, SQL Enjeksiyon saldırganları bir web servisin veritabanındaki kayıtları listeleme, silme, ekleme ve değiştirme CRUD (Oluşturma, Okuma, Güncelleme ve Silme) işlemi yapıldığı gözlemlenir. Bu nedenle, SQL Enjeksiyon, diğer web servisleri güvenlik açıkları arasında en eski ve en yaygın web servis saldırıları olarak kabul edilir (Ankit, Sharma , Sharma , Kaushik, & Bhushan , 2019).

Aşağıda bir SQL Enjeksiyon örneği verilmiştir. Budata web servisi kullanarak veritabanında erişen kullanıcının yetkilerine göre saldırı yapılabilir. Bu saldırılar web servisine değil doğrudan veritabanında olan saldırı olarak düşünülür.

SQL enjeksiyon saldırısı, saldırganın SQL sorgularına bazı sahte komutlar ekleyerek hatalı çalışmaya neden olduğu saldırı türüdür. Aşağıda **Şekil 12**'de örnek bir saldırı verilmiştir.



Şekil 12. SQL Enjeksiyon ile yapılan bir saldırı örneği.

XPath Language (XPATH) XML dokümanları içinde sorgu yapmak için kullanılan yorumlayıcı sorgulama dilidir. Web servis uygulamaları genellikle veri, konfigürasyon veya parametre verilerini saklamak için XML dosyalarını kullanır. XML dosyaları üzerinde işlem yapılırken XPath enjeksiyon saldırıları için önlem alınmamışsa; düğümlere erişmek için kullanılan sorgu komutları ile XML veritabanındaki tüm verilere ulaşmak mümkün olacaktır. XML verilerine ulaşıldıktan sonra hak yükseltme ve diğer veritabanlarına erişim gibi daha yıkıcı saldırılar ile karşı karşıya kalınmaktadır (Daş & Burak, 2020).

1.8.4 Ortadaki adam saldırısı (MITM)

MITM saldırısı, kullanıcıların gerçek güvenli bağlantıya mı yoksa benzer bir güvenli olmayan bağlantıya mı bağlandıklarını anlamalarını zorlaştıracak şekilde çalışır. Kullanıcı ağ ile bağlantı kurmaya çalıştığında, önce kullanıcı cihazı ile ilgili bilgileri içeren paketleri gerekli ağa gönderir. Ardından ağ, şifrelenmiş bağlantı anahtarını ve kullanıcı cihaz adresini içeren bir dijital sertifika oluşturur. Bağlantı başlatma sırasında geçirilen sertifika güvensiz olduğundan, saldırgan dijital sertifikaya kolayca erişebilir ve sertifikanın onayını kullanıcıya bırakarak sertifikadaki bilgileri değiştirebilir. Pek çok kullanıcı, sahte ve mükerrer sertifikaların nerede olduğunu ve bunlara karşılık gelen saldırıları kontrol etmek için yeterli bilgiye sahip değildir, bu nedenle sertifikaları kabul eder ve saldırganların saldırıyı gerçekleştirmesini sağlamak için güvenli olmayan ağa bağlantıya izin verir (Kapil, Manoj, & Jay, 2016).

1.8.5 Hizmet reddi (DoS)

Birçok biçimde ortaya çıkan DoS saldırıları, sistem kullanılabilirliğini azaltarak meşru kullanıcıların sistem erişimini engellemeye yönelik açık girişimlerdir. Bu girişimler genellikle Web servislerin internet bant genişliğinde geçen veri boyutunun azaltılmasında karşılaşılmaktadır.

Yazılım yaması bazı saldırılara karşı koruma sağlasa da İnternet paketlerinin düzensiz iletilmesinden yararlanan DoS sel saldırılarına karşı koruma sağlamaz. Hem saldırı tespiti hem de karşı önlemleri içeren ikincil bir savunma gereklidir (Carl, Kesidis, Brooks, & & Rai, 2006).

1.8.6 Sahtekarlığı (Spoofing)

Spoofing: Bilgisayarda başkalarını taklit etmek ve başka bir kullanıcının kimlik doğrulama mesajına (kullanıcı adı ve parola gibi) yasa dışı olarak erişmek ve bunları kullanmak anlamına gelir (JIANG, CHEN, DENG, & ZHONG, 2011).

1.9 Web Servis Güvenliğinde Sınırlıkları

Bu çalışmada kullanılan XAdES yapısının JAVA ortamında daha aktif kullanıldığı gözlemlenmiştir. Fakat yaptığımız çalışmada .Net veya diğer modern dillerde de kullanabileceği ön görülmüştür. Bundan dolayı bu çalışmada .Net dili kullanılmıştır. Fakat bundan sonraki çalışmalar .Net yerine başka dilleri kullanarak bu çalışmayı tekrar edebilir veya geliştirebilirler.

1.10 Amaçlana Araştırmanın Özeti

Web servisler kurumlar tarafından veri transferi ve sorgulamada en yaygın kullanılan yöntemdir. Web servislerin İnternet'te açık olmasından dolayı ister istemez bazı durumlarda saldırılara uğramaktadır. Saldırılar sonucunda kurumların web servis güvenliği açıklarından dolayı, veri hırsızlığı, veri kaybı ve öngörülmeyen maliyetler oluşması kaçınılmaz bir durum olarak gözlemlenmiştir. Bundan dolayı web servis güvenliğinde klasik (prefiks'siz) ve gelişmiş (prefiks'li) hakkında Bölüm 2 de ayrıntılı bilgi verilecektir. Ayrıca gelişmiş (prefiks'li) yapılan çalışmada karşımıza çıkan

problemler hakkında ve aldığımız önlemler ve yaptığımız geliştirmeler hakkında detaylı bilgi verilecektir.

Yaptığımıza çalışma sonucunda web servis güvenliği kurumlar arasında yaygınlaşmamasının en büyük sebebi Geliştirme, Test, Kalite Kontrol ve Ürünün canlı sisteme alınmasındaki sürecin kurumlar arasında hesaplanamamasıdır. Hesaplanamamasının en büyük bir diğer nedeni de birbirleri ile çalışacak kurumların buna hazır olup olmaması durumudur.

Ayrıca bu çalışmada kullandığımız Uygulama Programlama Arayüzü (Application Programming Interface (API))'lerin gelişmiş (prefiks'li) yapılara uygun olmamasından dolayı geliştirdiğimiz yapı üzerinden anlatılması hedeflenmektedir.

2. YÖNTEM

Web servisler mimarisi nedeniyle İnternet'te herkese açık ya da kurumlara özel olarak yayınlanmaktadır. Bu durum kurumların belirlediği çalışma şeklinden kaynaklanmaktadır. Kurumların çalışma şekillerine göre ister istemez web servislerde bazı güvenlik zafiyetleri getirmektedir. Güvenlik zafiyetleri kurumun belirlediği politikalara göre değişmektedir.

Web servisler İnternet'te açık olmasından dolayı ister istemez art niyetli saldırılar yapılmaktadır. Yapılan veya yapılabilecek saldırılardan kaçınmak için bizim tarafımızdan önerdiğimiz XAdES teknolojisi kullanarak, E-İmza ile imzalayarak OASIS tarafından tavsiye edilen klasik (prefiks'li) ve geliştirilmiş (prefiks'siz) yöntemler kullanılarak kurumlar için oluşabilecek maliyetleri en aza indirilmesi hedeflenmektedir.

2.1 Web Servis Güvenliğinde Envelope Belirlenmesi

Günümüzde aktif olarak kullanılan SOAP versiyonları SOAP versiyon 1.1 ve SOAP versiyon 1.2'dir. Bu çalışmada tercih edilen versiyon SOAP versiyon 1.2'dir. Envelope boş olarak **Şekil 13**'teki gibi oluşur. Bu çalışmada yapılacak işlemler **Şekil 13**'teki boş Envelope üzerinden devam edilecektir. SOAP 1.1 ve SOAP 1.2 veri transferleri için kullanılan kodlama Text ve Mesaj İletim Optimizasyon Mekanizması (Message Transmission Optimization Mechanism (MTOM)) olarak iki farklı tür kullanılmaktadır. MTOM kodlama Text kodlamadan farkı verinin taşınması veya verinin sorgulanması base64 formatında olmaksındır. MTOM network trafiği, boyut ve kodlama problemlerin oluşmamasından dolayı kurumların maliyetlerini düşürmektedir. Web servisler üzerinden gönderilecek PDF, XML, DOCX vb. dosyaların büyük boyutta olmasından dolayı MTOM kodlama kullanılarak oluşabilecek veri transferi zaman ve maliyet kazanımı sağlamış olacaktır.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"/>
```

Şekil 13. Boş bir Zarf ve SOAP versiyon 1.2'den oluşur.

Boş olarak oluşturulan Envelope **Şekil 13**'te ki gibi sadece SOAP versiyon bilgisi içermektedir. İstemci ile sunucu arasında hangi kodlama (Text, MTOM) ile verinin

iletileceği bilgisi içermemektedir. Kodlama bilgisi istemciden sunucuya veri gönderme veya sorgulama sırasında tanımlanır. Kodlama konusunu verinin sunucuya iletilmesi sırasında tanımlanmaktadır. Text ve MTOM seçimi sunucu tarafından yapılmaktadır. İletilecek dosya boyutuna göre de karar verilebilir. Küçük boyuttaki bilgiler Text web servisten iletilirken, büyük boyuttaki dosyalar için genellikle MTOM web servisleri seçilmektedir. Aynı web servis hem Text ve MTOM olarak çağırılması mümkündür.

Kurum tarafından talep edilen Envelope farklı projelerde kullanılıyorsa bunun için proje adı ve kurum adı bilgileri kullanılarak farklılaştırama gidilebilir. Bu özellik zorunlu olmamakla beraber ayırt edici bir özelliktir.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ns="http://accounting.abc.com/"/>
```

Şekil 14. Muhasebe'ye web servis güvenliği üzerinden gönderilen zarf.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ns="http://finance.abc.com/"/>
```

Şekil 15. Finans'sa web servis güvenliği üzerinden gönderilen zarf.

Muhasebe ve Finans güvenli web servislerini kullanılmak istenildiğinde gönderilen zarf hatalı bir servise gönderilip gönderilmediğini ayırt etmek için kullanılır. Bu kontrol imzalama ve imza doğrulama maliyetli olmasından dolayı öncelikle Schematron kontrolünde yapılmaktadır. Schematron kontrolünde yapılarak sunucu tarafında kuşkusuz oluşabileceği network trafiği ve IO kullanımlarından doğacak maliyetleri azaltmak için kullanılabilir. Ayırt edici özellik zorunlu olmamasına rağmen birden fazla serviste kullanılması durumunda bunu maliyetleri azaltmak için önerilmektedir. Ayırt edici attribute tanımlaması Body kısmında da kullanıldığı gözlemlenmiştir.

2.1.1 Body hazırlama

Body kısmı istemci tarafından iki farklı yöntem için kullanılır. Birinci yöntem istemci tarafından veri iletilmesi ikinci yöntem ise sunucudan veri talep etmektir. Her iki durum için de imzalama sırasında klasik (prefiks'li) ve güncel (prefiks'siz) yöntem kullanılarak, imzalama sırasında referans edilerek imzalanan değerleri alınarak imza içerisinde referans numarası ile saklanacaktır.

```
▼<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ns="http://finance.abc.com/">
 ▼<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="BodyId">
  ▼<ns:setBillOfSale>
     <paketID>55XXXXXXX74-BIN2022000000001</paketID>
     <paketValue>MIIF...</paketValue>
   </ns:setBillOfSale>
  </soap:Body>
 </soap:Envelope>
```

Şekil 16. İstemci tarafından iletilecek satış faturası Zarf.

Zarf içerisinde buluna Body tag **Şekil 16**'te bir satış faturasının 55XXXXXXX74 kişi veya kurum tarafından BIN2022000000001 satış faturasının base64 formatta sunucuya iletilmesi için kullanılacaktır prefiks'li wsu:Id:"BodyId" imzalama sırasında referans olarak BodyId tanımlaması yapılmıştır. İmza referansı için aşağıdaki gibi URI="#BodyId" imza içerisinde tekil bir değer olarak saklanmıştır.

Şekil 17. İstemci tarafından iletilen satış faturası sonuç sorgulama Zarf.

Zarf içerisinde bununa Body tag **Şekil 17**'te istemci tarafından başarılı bir şekilde iletilmiş olan faturanın sonuç bilgilerini ve durumunu öğrenmek için sorgulanan Zarf örneğidir. Bu örnekte 55XXXXXXX74 kişi veya kurum tarafından BIN2022000000001faturanın durum bilgisi alınır. Bu talep web servis güvenliği ile yapılması, üçüncü kişi tarafından bilgi gizliliği oluşturmuştur.

E-imza ve X.509 sertifikası kullanarak imzalanmak istenilen ve imzalamada referans edilen Body tag aşağıdaki gibi imza bilgileri içerisinde bulunur. Her referansın dijital değeri ve dijital algoritması DigitsMethod ve DigitsValue değerlerinde tutulur.

```
▼<ds:Reference xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="ReferenceBodyId" URI="#BodyId">
 ▼<ds:Transforms>
    <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>VtEmtQL1E2qE6KG/8O6SsM16xw0=</ds:DigestValue>
 </ds:Reference>
```

Şekil 18. E-imza tanımlamasında bulunan ve Body tagı ile referans edilmiş ds:Referans.

Body tag için **Şekil 18**'da imza referans bilgisi verilmiştir. Bu bilgiler Schematron ve E-imza doğrulama sırasında kullanılacaktır. Daha önceden de bahsettiğimiz URI="#BodyId" imza içerisindeki referans değeri ile bağlanılmıştır. Bir e-imzada birden

fazla referans bulunabilmektedir. Burada önemli olan referanslar arasında kopuk olmamasıdır.

2.1.2 X.509 v3 dijital sertifikasının e-imza üzerinden alınası

Her E-imza üzerinde X.509 sertifikası bulunmaktadır. İki farklı e-imza türü vardır, nitelikli ve niteliksiz. Her iki e-imza türü aynı e-imza da bulunmamaktadır. Sertifika ayrıca Açık Anahtar (Public Key) olarak da tanımlanır. X.509 sertifikası imzalanacak Zarf üzerinde veya imzalanan imza bilgileri üzerinde bulunmaktadır. Bu sertifika ile e-imza doğrulaması yapılmaktadır ve her zarf üzerinde en az bir tane X.509 sertifikası bulunması zorunludur. X.509 sertifikaları seri ve paralel imzalamalarda da kullanılmaktadır.

2.1.3 Header bilgisinin tanımlanması

Header tag'ı Zarf içerisine eklenmeden önce e-imza içerisinde bulunan sertifika bilgilerine ulaşmak için PIN kodu ile e-imzaya giriş işlemi yapılarak sertifika bilgileri alınır ve bu sertifika imzalama ve imzala doğrulamasında kullanılacaktır. Alınan sertifika Security tag'ın altında bulunan BinarySecurityToken tag'ında base64 formatında aklanılır. Saklanan sertifika bilgileri ile açık anahtar olarak kullanılmaktadır ve e-imza doğrulaması yapılmaktadır. Header içerisinden en kritik tag'lar dan bir tanesi de zarfnin geçerlilik zamanın belirleyen Timestamp özeliğidir. Bu zarfın oluşturma ve geçerlilik süresini belirler. Bu verilen örnekte 50 saniye içerisinde tüketilesi gereken bir talep olarak gözlemlenmektedir. Bu 50 saniye içerisinde XML'lin imzalanması ve iletilmesi süresi olarak tanımlanır. Tanımlanan zaman dışında bir istek gönderilemez. Sunucu tarafından hata zarfı ile istemci tarafından yapılan istek bilgilendirilir.

```
▼<soap:Header xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  ▼<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
   xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
     <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.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#X509PKIPathv1" wsu:Id="X509-92EB8E5C86227B3244140429163031146">MZ....</wsse:BinarySecurityToken>
    ▼<wsu:Timestamp wsu:Id="TS-32">
       <wsu:Created>2014-07-02T09:00:30.307Z</wsu:Created>
       <wsu:Expires>2014-07-02T09:01:20.307Z</wsu:Expires>
     </wsu:Timestamp>
   </wsse:Security>
 </soap:Header>
```

Şekil 19. Timestamp zaman aralığının belirlenmesi.

2.1.4 Body'nin hazırlanması

Body tagının hazırlanması sırasında aşağıdaki gibi bilgiler tanımlama yapılır ve imzalama sırasında referans edilecek kısım olarak wsu:Id=”id-28” referansı tanımlanır. OASIS tarafından wsu:Id tanımlanan prefiks'li bir yöntemdir. Fakat sunucu tarafından bu yöntemin kabul edilmeden de Id üzerinden de gerçekleştirile bilinmektedir. Body tag'ı ve alt tag'lar serbest olarak bırakılmıştır. Body tag'ın da en önemli özellik imzalayan ile imzalanan verinin birbiri ile bir bağlantısının olup olmadığı tekilliğin tespit edilmesinin sağlamaktır. Bu tekillik gönderilen verinin sorgulanmasında kullanılmaktadır.

```
▼<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
 xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:web="http://webservice.barsan.com/" wsu:Id="id-28">
 ▼<web:getBatchStatus>
    <paketID>8e96ce76-8be9-4520-9d23-f27e70edf852</paketID>
  </web:getBatchStatus>
 </soap:Body>
```

Şekil 20. Tekil paket sorgulama.

2.1.5 Timestamp belirlenmesi

SOAP mesajının içerisinde yer alan Timestamp bilgisinin ömrü çok kısadır. Bu bilgi istemci tarafından oluşturulan ve imzalanan zaman dilimi ve e-imza içerisinde bulunan imza değerinin Londra saatine göre yapılarak Lokasyon farklılıklarında oluşan zamana göre tanımlamaların, yanlış olacağından dolayı Londra saati ön görülerek zaman hesabının yapılması sağlanmıştır. Türkiye'den imzalanarak iletilen Envelope 22:31 14 Mart 2022, Pazartesi (GMT+3) göstermesi ve Kanada Ottava'da 15:31 14 Mart 2022, Pazartesi (GMT-4) ile iletilen zamanların farklı ve yanlış olmasından. Asıl olması gereken Londra, Birleşik Krallık konumunda saat 19:31 14 Mart 2022, Pazartesi (GMT) Timestamp içerisinde tanımlı olmamasından dolayı, istemci tarafından iletilen zarf sunucu tarafından talep edilen zaman aralığında olmamasından 22:31 ve 15:31 beklenmemesinden dolayı sunucu tarafından Timestamp varidasyon 19:31 beklendiğinden hata vermektedir. Eğer 19:31 Timestamp içerisinde iletildiyse ve sunucu tarafından belirlenen gecikme süresi aralındaysa, bir sonraki işleme geçer ve bu işlem için başarılı sonuç dönecektir. Burada zarf oluşturma ve sonlandırma zaman aralığı hizmet veren sunucu tarafından belirlenmektedir. 50 sn. veya 5 dakika olabileceği gibi zaman uzaması durumunda sunucu tarafında saldırıların daha fazla olması kaçınılmaz bir

duruma sokacaktır. Bundan dolayı yaptığımız çalışmada 5 dakika bir iletilmesi ideal bir zaman olarak öngörülmektedir.

2.1.6 E-İmzanın belirlenmesi

XAdES imzalamada kullanılacak imza türü günümüzde en çok kullanılan imzalama türü Enveloped imzalama türüdür. Enveloping imzalama türü genellikle kullanılmamaktadır. Fakat güvenli bir ayrım için de seçile bilinmektedir. Bizim tarafımızdan önerilen imzalama türü Enveloped'tır. Çalışmamızda sadece Enveloped türü kullanılarak tüm işlemlere devam edilecektir.

2.1.7 İmzalama

İmzalama günümüzde çok farklı türleri ve algoritma türleri içermektedir. CADES, XAdES, PADES, ASiC olarak sınıflanmaktadır, e-imzalama türlerinin desteklediği standartlar. ETSI TS 101 733 CADES standardında elektronik imza formatı (ASN veri yapısı). ETSI TS 102 778 PADES standardında elektronik imza formatı (PDFveri yapısı). ETSI TS 101 903 XADES standardında elektronik imza formatı (XML veri yapısı). ETSI TS 102 918 ASiC standardında elektronik imza formatı olarak örnek verilmiştir. Bu desteklenen standartlar iç içe ve bağımlı olarak da kullanılabilmektedir. Örnek vermek gerekirse web servis güvenliği te iletilen veri XAdES olarak kullanılmaktadır. Body içerisinde iletilen veri base64 formatında CADES, PADES, ASiC olarak da kullaılmaktadır. Bu durum ayrıca CADES imzalama içerisinde PADES imzası da bulunabilmektedir. Bu durum kullanım ve talep edilen duruma göre değişiklikler göstermektedir.

2.1.8 İmzalama doğrulama

E-imza doğrulama birden fazla standart içermektedir. Her standart farklı algoritmalar içermesinden talep edilen imza standardı ve algoritma türüne göre yapılmalıdır. Bu karar sunucu tarafındadır. İmza algoritma güçlüğü de önemlidir. İmzalama da İnternet'te ihtiyaç duyulduğundan imza doğrulama nispeten daha kolaydır. İmza doğrulamada da internete gereksinim duyulmadığından nispetten daha hızlıdır. Bir e-imza ile yapılan imzalama 1 ile 2 sn. alırken doğrulama ise 0,03 sn. gibi bir değer alması imza doğrulamanın daha hızlı olduğundan bahsedilmiştir.

2.1.9 Schematron kontrolü

Schematron kontrolü imzalamadan önce en önemli kısımlarından biridir. Burada talep edilen zarf istenilen formatta olup olmadığı, tagların boş olmadığı ve doğru bir sırada olup olmadığı. Tag'ların içerisinde bulunan attribut'ların eksik olup olmadığı. Tag'ların boş olup olmadığı. İmzalama sırasında kullanılan imza türünün doğru ve eksiksiz yazılıp yazılmadığı. İmza XML'in imzasız XML'lin prefiks'li veya prefiks'siz referans Id'lerin olup olmadığı. Referans edilen imza türü ile imzalanan e-imza türünün eş olup olmadığı gibi kontrollerin yapılması için kullanılmaktadır. Bu işlem performans için vazgeçilmez bir durumdur. Schematron tüm web servisler için önerilmiştir.

2.2 Sunucu'ya İletilmesi

Sunucuya iletilen her talep için bir başarılı veya başarısız bir cevap iletilmesi zorunlu olmamasına rağmen beklenmektedir. Sunucu aldığı talebi yaptığı kontrol akışı sonucunda iletilen talebi kabul veya reddeder. Bunun sonucunda istemci tarafından alınan sonuca göre karar verilerek sunucu tarafından iletilen başarılı veya başarısız ileti istemci tarafından yorumlanarak karar verilmektedir. Sunucu iletme işlemini istemci tarafından iletilen ileti ile gerçekleştirilmektedir.

2.3 Web Servis Güvenliğinde Sonuçlar

E-imza kullanılarak; web servislerin güvenliği, sunucu ve istemci tarafında verilerin asimetrik olarak iletilmesi şeklinde ifade edilebilir. Bu durumda veri paketlerinin güvenliği sunucu tarafında sağlanmıştır. Bu durum teknik açıdan avantajlı olarak görülmektedir. Fakat performans açısından istemci tarafında yapılacak olan yorumlamalar teknik açıdan daha zordur ama daha avantajlıdır.

Veri gönderiminin yoğunlaşması durumunda veya isteklerin fazla olması durumunda performans iyileştirilmesi adına istemci tarafında güvenlik kontrolünün yapılması önemlidir. Ayrıca sunucu tarafında ölçeklendirilebilirlikten yararlanılarak bu performanstan kazanç sağlanabileceği öngörülmüştür. Ama bu durum ek maliyet kalemlerini beraberinde getirmektedir. Ayrıca operasyon ağının genişlemesinden dolayı teknik olarak iyileştirmeler zorlaşacaktır.

Sunucu tarafına gönderilen verilerde asimetrik imzalama türü kullanılarak sunucu tarafından basit bir zarf türünde istemciye iletilmesi daha uygun olarak gözükmektedir. Bu durumun performans açısından yapılması uygundur.

Web servis güvenliğinde şifre kullanılması, konuşacak birimlerin sertifika ve anahtar doğrulaması, güvenebilecekleri anahtarları bilmesi gibi işlerin halledilebilmesi için geniş kapsamlı bir kurulum çalışmasını gerekmektedir. Ayrıca servis güvenliğinde birçok durumda, bütünlük ve gizliliğin sağlanması için şifreleme imza ile birleştirilmiştir.

Güncel servis güvenliği çözümlerinde imzalayan anahtarlar şifreleyen anahtarlardan farklıdır. Bu durumun ana kaynağı ise anahtarların hayat sürelerinin farklı olmasındandır. İmzalayan anahtar sahipleri kalıcı anahtarlar oluşturulmuştur. Fakat şifreleyen anahtarlar mesaj alış-verişinden sonra geçersiz hale gelebilmektedir.

E-imza içerisinde sertifika bulunmaktadır. Bu sertifikaların ise belli bir geçerlilik tarihi vardır. Genellikle 1 ile 3 yıl olarak üretilmektedir. Bu bağlamda imzalama aşamasında tarihe bağlı olarak sertifikaların doğrulanması gerekmektedir. E-imza içerisinde bulun sertifika ve imzalama doğru hatta imza türü de doğru olmasına rağmen gönderilecek zarf için de belirtilen zaman aralığının dışında ise bu ilgili mesajı başarısız olacaktır. Ayrıca mesaj içerisinde belirtilen zaman aralıkları çok kısadır. Bu nedenle de bu işlemin istemci tarafında yapılması operasyonun sağlıklı ilerletilebilmesi için büyük avantaj sağlayacaktır.

Bu çalışma ile klasik OASIS tarafından önerilen SOAP zarf mesajı hazırlanmasında öncelikle X.509 sertifika e-imza üzerinden alınıp referans edilmesi yerine, imzalama sırasında eklenmesi ile zaman kazanılacağı öngörüldü. Bunu içinde gelişmiş yöntem sunuldu. Ayrıca çalışma literatüre sunucu tarafından talep edilen SOAP zarf mesajın farklılaşabileceği gözlemlenmiştir. Bu farklılaşmada güvenlikten taviz verilmeden SHA-256, SHA-384 veya SHA-512 şifreleme yöntemlerinin kullanılabileceği gözlemlenmiştir.

2.4 Web Servis Güvenliğinde Literatür Karşılaştırması

Web servisler ve APİ'ler yazılım süreçlerinin en önemli parçalarından biridir. Özel ve kamu kurumları müşteri ve kullanıcı ile olan veri alışverişlerinde bu çözümleri kullanmaktadır. Bu bağlamda servislerin güvenliği ve performansı operasyonların devamı adına hayati önem arz etmektedir.

Kamu kurumları özel kurumlara sundukları hizmetler için geliştirdikleri servislerin güvenliğini sağlamak adına çeşitli çözümler geliştirmektedirler. Bu çözümler e-imza tabanlı çözümlerden oluşabilmektedir. Bu çözümlerde imzalama doğrulaması sunucu tarafında yapılmaktadır. Sunucuya gönderilen SOAP mesajının zarfın içinde belirtilen tarihlerde geçerli olan e-imza eğer belirtilen zaman aralığında iletilmezse operasyonu başarısız olmasına neden olabilmektedir. Bu durumda sunucu kaynaklarını gereksiz kullanmış olmaktayız. Eğer bu kontrol istemci tarafında yapılıp ona göre mesaj iletilseydi server kaynakları doğru bir şekilde kullanılmış olunulacaktır. Ayrıca veri güvenliği ve bütünlüğü operasyonun başlaması ile garanti altına alınmış olacaktı. Fakat bu işlem teknik açıdan zor ve karmaşıktır. Geliştirdiğimiz modelde prefix'siz tabanlı bir çözüm kullandık. Bu işlemi başarılı bir şekilde gerçekleştirdik. Bu konuda yapılan çalışmalardan bazıları bu bölümde kısaca açıklanmıştır.

Nils Engelbertz ve arkadaşları yapmış oldukları çalışmada DSS (Digital Signature Services) yöntemini kullanarak, XML ve XAdES şifreleme yöntemini ile çalışan web servis ve web servis güvenliği saldırılarının bertaraf ettiklerini öne sürmüşlerdir (Nils Engelbertz, 2019). Fakat bizim yaptığımız çalışmada Body ile iletilen verinin imzalı ve zaman damgalı validasyonları ile XAdES ile yapılan sahte kayıtlar için önlem alınabileceği gözlemlenmiştir.

Wawrzyniak ve arkadaşları yaptıkları çalışmada E-imza imzalama ile yapılan imzalamalarda referans bağlantı problemlerin ve zafiyetler hakkında bilgi verilmiştir (Fray, 2020). Bizim yaptığımıza çalışmada referans bilgilerinin azaltarak, imza içerisinde bulunan referans kopukluklarını en aza indirmektedir.

Eriyanto Adhi Setyawan ile arkadaşları 2020'de yaptıkları çalışmada web servis güvenliği için dinamik analiz, statik analiz, filtreler kullanarak güvenlik mekanizması, kimlik

doğrulama, şifreleme veya güvenli kodlama ve model tabanlı teknikler hakkında bilgi verilmişlerdir (Hidayat, 2020). Buradaki çalışmada kimlik doğrulası, şifreleme ve kodlama tekniklilikerinin kullanıldığı tespit edilmiştir.

Bukovetskyi ve arkadaşları 2022'de yaptıkları çalışmada MITM saldırıların arttığı gözlemlemişlerdir. Bu ataklar için analizler ve alınabilecek yöntemler hakkında bilgi verilmiştir (Rizak., 2022). Bizim çalışmamızda, imza doğrulama ve Schematron kontrolü ile oluşabilecek ataklar bertaraf edilmektedir.

Rohlmann ve arkadaşları yapmış oldukları çalışmada, Mircosoft Office ve DSS güvenli olduklarını fakat yapmış oldukları 18 tane saldırıdan 16 tanesinin 66/66/6666 gibi bir saldırıyı bile bertaraf edemediklerini gözlemişler (Simon Rohlmann, 2022). Bu çalışmada wsu:Timestamp tagı içerisinde bulunan wsu:Created ve wsu:Expires bilgileri Schematron sırasında kontrol ile bertaraf edilmiştir.

3. BULGULAR VE YORUMLAR

Web servis güvenliğinde OASIS konsorsiyum tarafından sunulan klasik (prefiks'li (wsu:Id)) referans tanımlanması kullanılmıştır. Kullanılan klasik referans tanımlaması yerine gelişmiş (prefiks'siz (Id)) referans tanımlamasının kullanılmasının geliştiriciler için önemli kaynak kazanımının sağlayacağını öngörülmüştür. Bu önerimizin sebebi e-imzadan alınan X.509 sertifikası sadece imzalama sırasında kullanılmasıdır. Fakat OASIS tarafından imzalama yapılmadan önce zarf içerisinde tanımlanması ve daha sonra imzalanması talebi olmasından T1 sürede yapılacak işlem ister istemez T1+T2 sürede gerçekleştirilmiş olacaktır. Ayrıca günümüzde kullanılan üçüncü parti API'ler klasik referans tanımlama imzalamalarında çok aktif olarak kullanılmadığından Geliştirme, Test, Kalite Kontrol ve Canlı ortamında oluşan personel, zaman ve uzun süreç nedeniyle maliyet artışından dolayı kurumlar tarafından pek kullanılmadığını gözlemlenmiştir. Bizim önerdiğimiz gelişmiş yöntem ile imzalanan zarf da aynı güvenlik standartlarda olacaktır.

3.1 Web Servis Güvenliği için Klasik ve Geliştirilmiş Yöntem İşleyişi

OASIS tarafından önerilen klasik referans tanımlama tabanlı çözüm gösterilmiştir. Bu yöntem temelde iki ana modülden oluşmaktadır. Birinci modül istemci tarafındaki süreci ifade etmektedir. İkinci modül ise sunucu tarafındaki süreci ifade etmektedir.

3.1.1 Birinci modül

ModulClient yedi temel ana adımdan oluşmaktadır.

- İlk adım, istemci tarafında zarf için veri yüklemenin veya sorgulamanın karar verildiği adımdır.
- İkinci ve üçüncü adım, e-imza üzerinden X.509 sertifikasının alındığı ve XML'in oluşturulduğu adımdır.
- Dördüncü adım, wsse:BinarySecurityToken tagına X.509 sertifikası için base64 formatında oluşturulan verinin XML'e eklendiği adımdır.
- Beşinci ve altıncı adım, oluşturulan zarf XML'linin imzalandığı ve imza değerinin kontrolünün yapıldığı adımdır.

- Son adım olan yedinci adımda ise zarf sunucuya iletilmeden önce Schematron kontrolüne tabi tutulan adımdır.

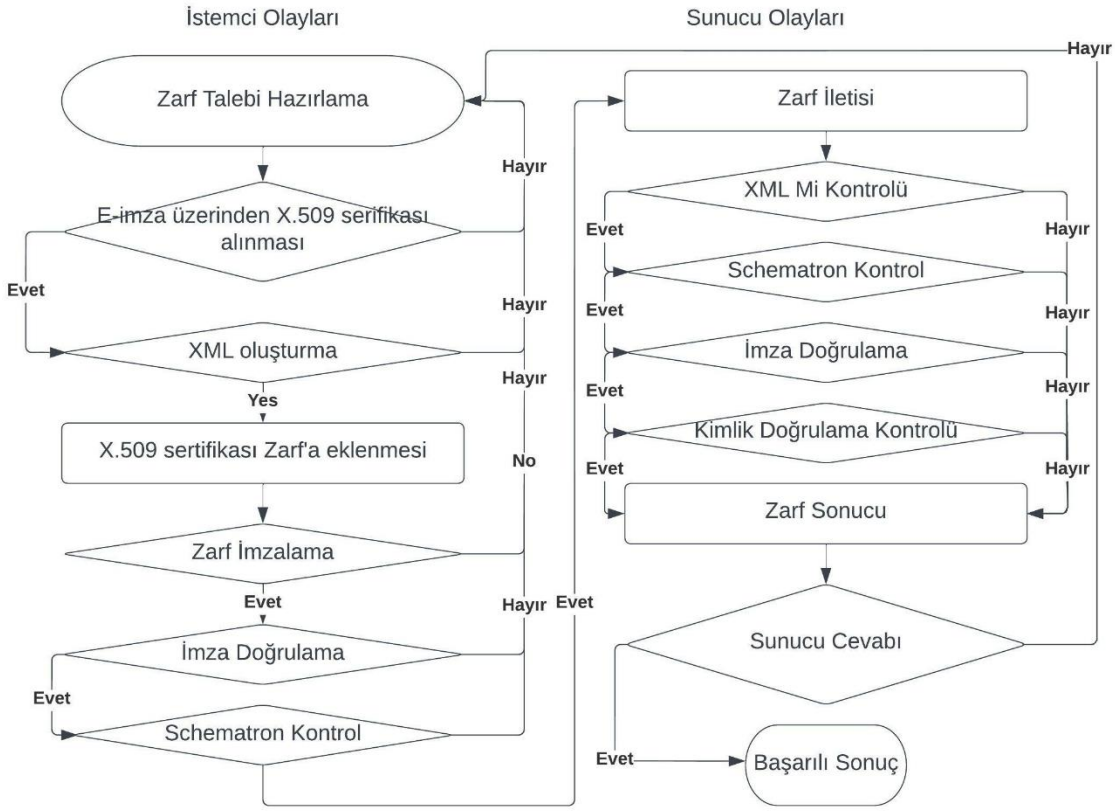
Sunucuya Envelope request mesajı iletilir. Böylece istemci tarafındaki temel adımlar tamamlanmış olur. İstemci tarafındaki işlemlerin tamamlanmasından sonra süreç sunucu tarafında ikinci modül ile devam eder.

3.1.2 İkinci modül

İkinci modül sunucu tarafında yapılacak olan işlemlerin gerçekleştirildiği modüldür. Temelde 7 adımdan oluşmaktadır.

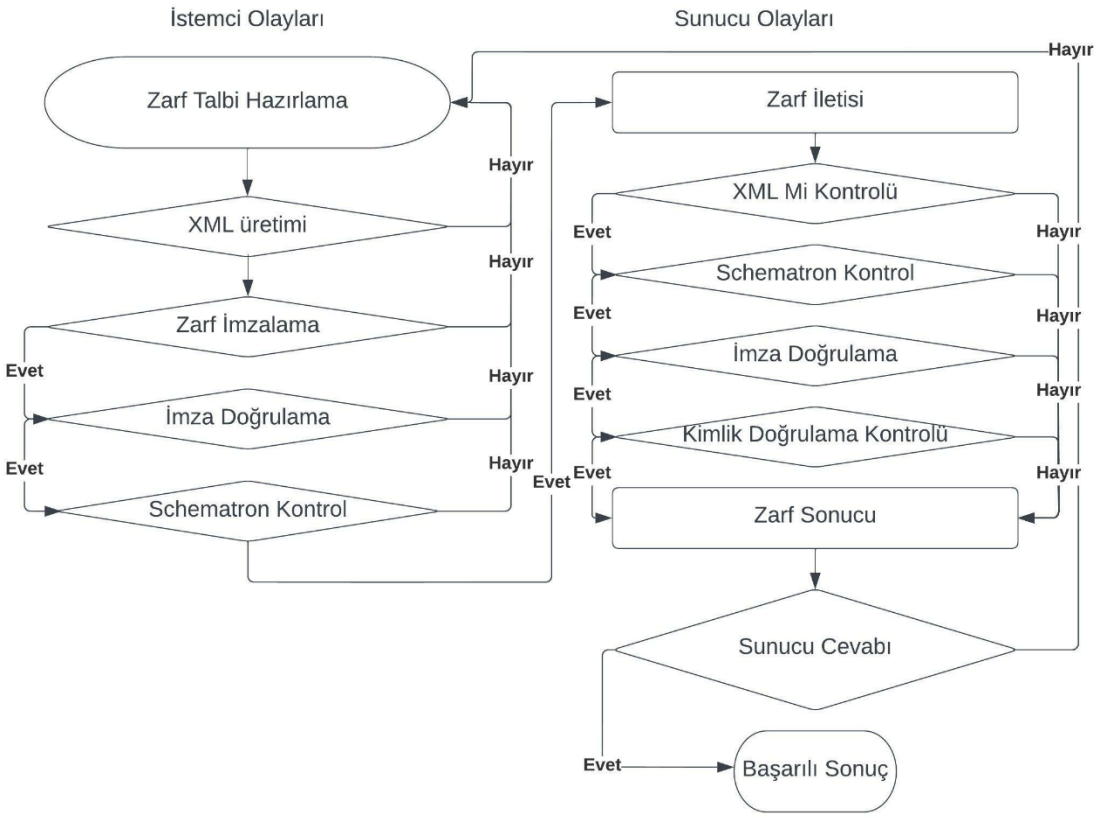
- Birinci adım istemci tarafından iletilen istek (request) mesajının alındığı adımdır.
- İkinci adımdan altıncı adımda kadar XML kontrolü, Schematron kontrolü, imza doğrulam, istemci kontrolü (Authentication) kontrolleri yapılır.
- Son adımda ise Sunucu tarafından oluşturulan Envelope cevabı (response) XML'li istemciye başarılı veya başarısız olarak döndürülür.

Bu modüldeki bütün süreçler server tarafında gerçekleştirilmiştir.



Şekil 21. Klasik referans tanımlanması kullanımı.

Klasik yöntem prefix'li yöntemdi. E-imza üzerinde bulunan X.509 sertifikası alınarak zarf üzerine eklenmektedir. Eklenen X.509 sertifikası imzalama sırasında referans edilmektedir. Bu işlem OASIS tarafından tavsiye edilen yöntemdir. Klasik yöntem referans tanımlamada nispeten geliştirilmiş prefiks'siz yönteme göre daha karışık olarak tanımlanmıştır.



Şekil 22. Gelişmiş referans tanımlanması kullanımı.

Bu çalışmada klasik yöntem daha karmaşık ve adımların daha çok olduğu gözlemlenmiştir. Geliştirilmiş yöntem ile daha az adım ile güvenlikten taviz verilmeden de tamamlanacağı gözlemlenmiştir.

3.2 Geliştirilmiş ve Klasik Önerimiz Hakkında

Bizim tarafımızda önerilen geliştirilmiş referans tanımlama tabanlı çözüm gösterilmiştir. Bu yöntem klasik yöntem gibi iki ana modülden oluşmaktadır. Bizim modelimizde istemci tarafında kullanılan klasik sistemde kullanılan Dijital İmza üzerinden X.509 sertifikasının alınması ve wsse:BinarySecurityToken tagına X.509 sertifikası için base64 formatında oluşturulan verinin XML'le eklendiği adımlar çıkartılarak oluşturulmuştur.

Birinci modül istemci beş temel ana adımdan oluşmaktadır.

- İlk adım, istemci tarafında Envelope için veri yüklemenin veya sorgulamanın karar verildiği adımdır.
- İkinci adım, XML'in oluşturulduğu adımdır.

- Üçüncü ve dördüncü adım, oluşturulan Envelope XML'linin imzalandığı ve imza değerinin kontrolünün yapıldığı adımdır.
- İstemci için klasik yöntemde kullanılan diğer adımlara devam edilmiştir.

İkinci modül sunucu tarafında yapılacak olan işlemlerin gerçekleştirildiği modüldür. Temelde 7 adımdan oluşmaktadır. Bu adımlar klasik yöntemde ikinci modülde yapılan işlemler ile aynıdır. İkinci modülde Schematron kontrol ve imza doğrulama kısımları güncellenerek ds:Signature üzerinde ds:X509Certificate X.509 sertifika alınması sağlanmıştır.

Günümüzde kullanılan API'lerin geliştirilmiş yöntemde kullanılan imzalamada daha aktif kullanılmaktadır. API'ler yaydın olarak imzalama işlemi için geliştirilmiş referans çözümü kullanılmaktadır. Geliştirilmiş çözümünde bakım masrafları daha azalacağını ön görmekteyiz. Sunucu tarafında ise klasik sistemde kullanılan imzalama doğrulama ve Schematron kontrolleri geliştirilmiş yöntem ile değişime uğrayacağını ön gördük. OASIS tarafından sunulan yöntem imzalama sırasında bizim tarafımızdan önerilen yönteme göre daha zor ve karmaşık olmasından dolayı, sunmuş olduğumuz yöntem Web servis güvenliği için kullanabileceğini bütünlük ve güvenlik seviyesin de benzer olacağı tespit ettik.

3.3 Aldığımız Veriler ve Verilerin Yorumlanması

Web servis güvenliğinde OASIS konsorsiyum tarafından klasik metot ile geliştirilmiş metot arasındaki yaptığımız çalışma sonuçları **Tablo 1**, **Tablo 2** ve **Tablo 3**' te verilmiştir.

Tablo 1. 100 adet SOAP zarf mesaj tablosu

100 ADET SOAP ZARF MESAJ TABLOSU		
İstemci Etkinlikleri	**100 SOAP Zarf**	
	Klasik	**Geliştirilmiş**
E-İmzadan X.509 Sertifikası Alınması	5,03 sn	0,00 sn
XML Üretimi	1,01 sn	1,01 sn
Zarfa X.509 Sertifikası Ekleme	1,00 sn	0,00 sn
Zarf İmzalama	120 sn	120 sn
Zarf İmza Doğrulama	0,30 sn	0,30 sn
Schematron Kontrol	0,20 sn	0,20 sn
Ortalama Toplam:	**127,54 sn**	**121,51 sn**

Tablo 1'de verilen değerler SOAP Envelope mesajının dosya boyutuna göre değişmektedir. İstemci tarafında yaptığımız çalışmada klasik çözüm ile geliştirilmiş çözüm arasında zaman farklılığı gözlemlenmiştir. Bu farklılık E-imza üzerinden alınan X.509 sertifikasının SOAP Envelope mesajının klasik çözümde eklenmesidir. Fakat geliştirilmiş çözümde bu işlemi yapılasına gereksinim olmadığından zaman kazanımı oluşmuştur. Sonuç olarak klasik çözüm ile geliştirilmiş çözüm istemci tarafında yapılan işlemlerde zaman farklılıkları gözlemlenmiştir. Sunucu tarafından ise her SOAP Envelope mesajı için tüm aşamalar ortalama 0,03 saniyedir.

Tablo 2. 1.000 adet SOAP zarf mesaj tablosu

1.000 Adet SOAP Zarf mesaj TABLOSU		
İstemci Etkinlikleri	**1.000 SOAP Zarf**	
	Klasik	**Geliştirilmiş**
E-İmzadan X.509 Sertifikası Alınması	5,03 sn	0,00 sn
XML Üretimi	8,49 sn	8,49 sn
Zarfa X.509 Sertifikası Ekleme	5,78 sn	0,00 sn
Zarf İmzalama	1200 sn	1200 sn
Zarf İmza Doğrulama	3.00 sn	3.00 sn
Schematron Kontrol	2,00 sn	2,00 sn
Ortalama Toplam:	**1221 sn**	**1210 sn**

Tablo 2'de 1.000 tane SOAP Envelope mesajı üzerinden yapılan sonuçlar aktarılmıştır. Buradaki zaman her SOAP Envelope mesajın E-imza ile imzalama sırasında geçen zamandan kaynaklanmaktadır. Bu durum klasik ve geliştirilmiş süreçlerde aynıdır.

Tablo 3. 10.000 adet SOAP zarf mesaj tablosu

10.000 ADET SOAP ZARF MESAJ TABLOSU		
İstemci Etkinlikleri	**1.000 SOAP Zarf**	
	Klasik	**Geliştirilmiş**
E-İmzadan X.509 Sertifikası Alınması	5,03 sn	0,00 sn
XML Üretimi	79,58 sn	79,58 sn
Zarfa X.509 Sertifikası Ekleme	46,85 sn	0,00 sn
Zarf İmzalama	120,00 sn	120,00 sn
Zarf İmza Doğrulama	26,58 sn	3.00 sn
Schematron Kontrol	19,54 sn	19,54 sn
Ortalama Toplam:	**3 saat**	**3 saat**

Tablo 3'te 10.000 tane SOAP Envelope mesajı üzerinden yapılan sonuçlar aktarılmıştır. 1.000.0000 ve üzeri gönderimlerde imzalamadan uzun sürmesinden 3-5 gün arası bir süre alacağından dolayı farklı bir yöntem ise bu süre kısaltılmıştır. Bunu için E-imza türü değişikliğine gidilerek HSM (Hardware Security Module) cihazı kullanılması tasfiye edilmiştir. HSM da yapılan imzalama, E-imza cihazında 50 ile 100 kat daha hızlı yapılmaktadır. Bundan dolayı oluşabilecek yüksek veri transferlerinde HSM ile yapılması tavsiye edilmiştir.

3.4 Lokasyon ve Çalışma Vardiyası Sorunu

Günümüzde kullanılan tüm E-imzalarda imzalama işlemi için X.509 sertifikaları kullanılmaktadır. X.509 sertifikaları Public Key olarak da kabul edilmektedir. E-imza sağlayıcıları, imzalama sırasında Private Key Infrastructure metodolojiden yararlanarak imzalama hizmeti vermektedir. Bu işlem gerçekleştirilirken şifrelemede asimetrik yaklaşımları kullanılmaktadır. E-İmza içerisinde bulunan Private Key 256-bitlik bir şifre ile saklanmaktadır. Böylece üçüncü şahısların erişimi engellenmesi sağlanacaktır.

X.509 sertifikalarının geçerli olup olmadığı, E-imza sağlayıcıları tarafından belirlenmektedir. E-imza ile XML dosyasını imzalamak için E-imza sağlayıcılarına erişiliyor olmalıdır. E-imza sağlayıcısı tarafından sağlanan X.509 sertifikasının geçerliliğine ihtiyaç duyulmaktadır. SOAP Envelope mesajını imzalamak için internette gereksinim duyulmaktadır. Fakat X.509 sertifikası geçerli imza ile imzalanan SOAP Envelope mesajını üzerinde dolayı internete ihtiyaç duyulmadan, imza doğrulaması yapılabilmektedir. Bu işlem için de X.509 sertifikası Public Key olarak kullanılmaktadır.

Public Key imzalamadan sonra Envelope üzerinde olmasından internet ihtiyacı veya bir kuruluşa başvurma zorunluluğu oluşmayacaktır.

Klasik yaklaşımda en önemli kısıtlamalarından bir tanesi Lokasyon ve ülkelerin çalışma sürelerinin farklı olması durumudur. Devlet kurumlarında kullanılan servisler 7/24 prensibine göre çalışmaktadır. Fakat devlet kurumlarından alınacak destekler kurumun belirlediği yöntem ve çalışma zamanı farklılığından ister istemez web servis güvenliği için yapılan entegrasyonlarda zaman kaymasının olduğu gözlemlenmiştir. Geliştirme yapan kurumlar için çalışma saatleri projedeki entegrasyona göre belirlenmelidir. Bu durumda doğru planlama yapılmaması durumunda projenin uzaması ve maliyetlerin artması kaçınılmaz olacaktır.

Kamu kurumlar ile yapılacak olan bu tür entegrasyonlarda en çok karşılaşılan problemlerden bir tanesi de personellerin bu konuda yetersiz kalması durumudur. Personellerin bilgisinin yetersiz kalması durumunda ve böyle bir entegrasyonun ilk defa yapılması sürecinde tarafların birbirleri ile yaptıkları bilgi paylaşımında problemler yaşanabilmektedir. Kullanılan teknolojinin de farklı olması durumunda da bilgi transferinde zorluklar olacağı aşikardır. Ayrıca taraflar tarafında kullanılan güncel teknolojilerin birbirleri ile uyumlu olması sürecin başarısı için önemlidir.

Web servis güvenliğinde en kritik süreç geliştirme sürecidir. Geliştirme sürecine başlamadan önce bu süreci tamamlamış kurum ile çalışılması önerilmiştir. Web servis güvenliğinin en yaygın olarak uygulandığı yerler devlet kurumları ve bankalardır. Bu bağlamda bu kurumlar ile yapılacak olan entegrasyonlarda geliştirme süreci tamamlandıktan sonra test, kalite kontrol ve canlı süreçleri geçişleri, geliştirme sürecine göre daha hızlı bir şekilde gerçekleştirile bilinecektir.

Tablo 4. 6 ülke seçilerek bir gündeki mesai tablosu

Lokasyon		UTC	Bir günlük mesaide çalışma saatlerin belirlenmesi																							Destek Oranı
ABD	Washington, Ds	UTC -4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	62,50%
Rusya	Moskova	UTC +3	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	37,50%
Çin	Pekin	UTC +8	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	7	8	9	10	0,00%
İngiltere	Londra	UTC +1	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	62,50%
Fransa	Paris	UTC +2	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	25,00%
Trkiye	Ankara	UTC +3	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	37,50%

Tablo 4'te Amerika Birleşik Devletleri'nde yapılan bir çalışmanın diğer ülkelerden alacağı destek oraları verilmiştir. ABD, Rusya, Çin, İngiltere, Fransa ve Türkiye gibi ülkelerin başkentleri tanımlanmıştır.

Amerika Birleşik Devletleri'nde, kurum tarafında 09:00 ile 17:00 günlük çalışma saati olarak tanımlanmıştır. Bu çalışmamızda farklı ülkeler için web servis güvenliği entegrasyonu alınabilecek ve verilebilecek destekler hakkında zaman çizelgesi verilmiştir. Her ülke için UTC farklılıklarından dolayı destek oran farklılığı yüzdelik olarak verilmiştir. Web servis güvenliği çalışmasında avantajlı olan kısım UTC±00.00'nin zaman diliminin kullanılmasıdır. Dezavantajlı kısım ise iletilecek SOAP Envelope mesajının wsu:Timestamp zaman aralığının beş ile on dakika gibi zaman aralığının çok kısa olmasıdır.

Geliştiriciler tarafından karşılaşılan hatalar ve talep edilen bilgilerin kurumların çalışma zamanı farklılıklarından dolayı uzayabilmektedir. Bu tür entegrasyonlarda çalışma zamanı, çalışılacak ülkeye göre farklılık arz edebilecektir. Bu tür entegrasyonlarda vardiyalı çalışma prensibinin kullanılması gerekmektedir. Planlamanın düzgün yapılmaması durumunda da DEV, TEST, QA, PROD entegrasyon süreci uzayacaktır. Bu nedenle zaman planlaması ve çalışacak kişi veya kişilerin vardiya planına göre çalışıyor olması talep edilmektedir.

Bu çalışma sonucunda kullanılan web servis güvenliği klasik yöntem dışında geliştirilmiş yöntem ile de aynı düzeyde başarıya ulaşabileceği ve hatta daha da yaygınlaşabileceği ön görmekteyiz. Bu çalışma sonucunda Devlet kurumları ve Banka'lar dışında da çık büyük maliyetlere gerek kalmadan geliştirilmiş yöntem ile de güvenlikten taviz verilmeden daha az adımla da istenilen sonuçlara ulaşabileceği öngördük.

4. SONUÇ ve ÖNERİLER

4.1 Özet

Günümüzde kurumlar arasında veri değiş tokuşu ve sorgulama işlemi için en popüler kullanılan yöntem kuşkusuz web servislerdir. Web servis türlerinde en popüler olan SOAP ve REST'tir. Her iki servisinde çalışma şekilleri farklı olmasına rağmen veri değiş tokuşu ve sorgulamada kullanılmasından dolayı amaçları aynıdır. Bu durum da kullanılmak istenilen servis türünde avantajları ve dezavantajları mevcuttur. Web servisin tür seçimi kurumların ihtiyaçlarına ve aldıkları politikalar ile belirlenmektedir. Bu çalışmada devlet kurumları ile özel/tüzel kurumlar arasında kullanılacak web servislerin güvenli çalışmaları gözlemlenmiştir. Sadece devlet ile değil aynı zamanda tüm kurumların web servis güvenliğini tavsiye edilmiştir. Web servis güvenliği için de XAdES ve e-imza ile zarf imzalanması önerilmiştir.

Web servisler saldırılara açık olarak günümüzde devam etmesinden dolayı XAdES ve e-imza ile oluşturulan zarf bilgilerinin tekrar kullanılmaması için belirtilen Timestamp özeliği ile zaman kısıtlaması yapılarak saldırıları en aza indirilmesi hedeflenmiştir. E-imza ile imzalanan zarfın belirtilen zaman içerisinde sadece bir defa kullanılması hedeflenmiştir. Web servis güvenliğinde istemci tarafından yapılan işlemlerin sunucu tarafından kontrollerden daha uzun süreceğinden sunucu saldırılara karşı daha az işlem yapmasını ve web servis güvenliğini kullanan kurumlar için maliyetleri azaltılması hedeflenmiştir.

4.2 Yargı

Günümüzde kullandığımız bazı imzalama API'lerde tespit etmiş olunan referans bilgilerinin prefiks'siz olarak kullanılırken, bazı referansların ise prefiks'li kullanılmasından dolayı imzalama ve imzalama doğrulamasında talep edilen imzayı oluşmadığı ve imzalama yapılmış olmasında rağmen doğrulanamadığı gözlemlenmiştir.

Bu durum ister isteme kurumların web servis güvenliğinde alacakları yöntem de önemli hale gelmiştir. Kullanılacak yöntem klasik (prefiks'siz) ve gelişmiş(prefiks'li) kurumların alacakları, karar verecekleri kabul yöntemi ister istemez tercih sebebi oluşturacaktır.

4.3 Öneriler

Kurumlar tarafından güvenli web servis geçişleri sırasında karşılaşılan en büyük problem, talep edilen ve talep eden kuruluşlar arasında oluşturulan Geliştirme, Test, Kalite Kontrol ve Canlıya geçiş aşamasında ilk maliyetlerin yüksek olmasından dolayı günümüzde Devlet kurumlarında bu sistemlerin daha yaygın kullanıldığı gözlemlenmiştir. Devlet kurumları dışından da en çok kullanılan kurumlar ise bankalardır. Bu tezimizde devlet kurumları ve bankaları dışında özel/tüzel kurumalar ile de aralarında bu sistemi geliştirmeleri gerektiğini gözlemlenerek, önerilmiştir.

Ayrıca web servisler günümüzde yazılım çözümlerinin en önemli parçasıdır. Yapılan bu çalışma ile SOAP web servislerinin kamusal alandaki kullanımda karşılaşan zorlukların üstesinden nasıl gelinebileceği konusunda yeni bir yaklaşım sunulmuştur. Sunulan bu yaklaşım "klasik tabanlı" modeline aksine "geliştirilmiş modelin" kullanılmasına dayanmaktadır. Önerilen yaklaşım firmaların uluslararası arenada kamu kurumları ile entegre olunmasında önemli bir kolaylık getirmektedir. Bu yaklaşım ile Geliştirme, Test, Kalite Kontrol ve Canlı ortamlarında kamu kurumları ile entegre olunması durumunda sıklıkla karşılaşan Timestamp problemini çözülmesi başarılmıştır. E-imza ve HSM cihazları tercihi hakkında da bilgi verilmiştir. Günümüzde kullanılan API'ler hakkına daha efektif kullanım için alınması gereken yol hakkında bilgi verilmiştir. Web servis güvenliği metodolojileri hakkında kurumlar aydınlatılmaya çalışılmıştır. Web servislerde SSL/TLS sertifikaları sadece istemci ile sunucu arasındaki uç uca güvenlik için ideal olduğu fakat saldırılara karşı güvensiz ve yetersiz olduğu bilgisi verilerek, web servis güvenliği kullanımını yaygınlaştırılmasını önerilmiştir.

4.4 Çalışma Kısıtları ve Gelecekteki Çalışmamız

Bu çalışmada yapılan araştırma sonucunda; Devlet kurumları ve bankalarda yaygın olarak kullanıldığı fakat özel/tüzel kurumlarda sık kullanılmadığı gözlemlenmiştir. Bu çalışmada bu eksiklik gözlemlenmiştir. Buradaki amaç özel/tüzel kurumlarsa yaygınlaştırarak web servis güvenliğini farklı yöntemler ile XAdES kullanımı artırmaktır.

Yapılan bu çalışmada, devlet kurumları ve bankalar dışında da özel kurumlar arasından web servis güvenliğini artırılarak veri gönderme ve veri sorgulama işlemlerini daha hızlı ve güvenli bir şekilde yapılmasının, ayrıca kurumlar tarafından, XAdES bilinçli bir şekilde kullanımını sağlanması hedefleyerek oluşabilecek zaman ve maliyetleri kaybını azaltmak için alınacak yöntemler hakkında bilgi verilmiştir.

Bu çalışma sonucunda; kurumlar tarafından web servis güvenliği için alacakları ve izletecekleri yöntemler hakkında detaylı bilgi verilerek oluşabilecek ön maliyetin zamanla daha avantajlı bir duruma geleceği anlatılmıştır. Bu çalışmada kurumların karşılaşabileceği zorlukları nasıl bertaraf edilebileceğini ve karar aşamasında hangi yöntem daha sağlıklı bir şekilde olduğu anlatılarak, çok karmaşık yapıların da aslında ne kadar kolay bir şekilde gerçek hayata geçebileceğini gözlemlenmiştir.

En büyük hedef; web servislerde yapılan veri değiş tokuşu ve sorgulama işlemlerinin inkâr edilememesi ve bu işlem için kurumların yöntemlerin en basit karmaşıklıktan uzak bir şekilde gerçekleştirildiğini gözlemlemektir.

KAYNAKLAR

A. Arsanjani, G. B. (2009). *SOA manifesto*. 1 1, 2017 tarihinde http://www.soa-manifesto.org/ adresinden alındı

Abdellatif, M. e. (2018). State of the practice in service identification for soa migration in industry. (s. 634–650). International Conference on Service-Oriented Computing. Springer, Cham, 2018.

Abdul, W. M., & Ahmed , M. Z. (2017). Web services SOAP optimization techniques. *In 2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, 1-5.

Ankit, S., Sharma , N., Sharma , A., Kaushik, I., & Bhushan , B. (2019). Taxonomy of attacks on web based applications. *In 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)* (s. 1231-1235). IEEE.

Bieberstein, N. e. (2006). Service-oriented architecture compass: business value, planning, and enterprise roadmap. *FT Press*.

Booth, D. a. (2007). Web services description language (WSDL) version 2.0 part 0: Primer. *W3C recommendation 26*, 39-41.

Carl, G., Kesidis, G., Brooks, R. R., & & Rai, S. (2006). Denial-of-service attack-detection techniques. *IEEE Internet computing 10(1)*, 82-89.

Chinnici, R. e. (2007). Web services description language (wsdl) version 2.0 part 1: Core language. *W3C recommendation 26.1*, 19.

Daş, R., & Burak, B. (2020). Analysis of Different Types of Network Attacks on the GNS3 Platform. *Sakarya University Journal of Computer and Information Sciences*, 210-230.

Debasish, C., & Sristy, S. N. (2017). Web Service Performance Enhancement for Portable. *20th International Conference of Computer and Information Technology (ICCIT)*.

E., C., F., C., G., M., & S., W. (2001). *Web services description language (WSDL) 1.1*.

Eriyanto, A. S., & Fadhil, H. (2020). Web Services Security and Threats: A Systematic Literature Review. *In 2020 International Conference on ICT for Smart Society (ICISS)*, 1-6.

Fray, G. W. (2020). *New xml signature scheme that is resistant to some attacks. .* IEEE Access.

Gupta, P. e. (2018). Event-driven SOA-based IoT architecture. (s. 247-258). International Conference on Intelligent Computing and Applications. Springer, Singapore, 2018.

Haviluddin, H., Edy, B., & Nur, F. H. (2019). A Database Integrated System Based on SOAP Web Service. *TEM Journal*, 782-787.

Hidayat, E. A. (2020). Web services security and threats: A systematic literature review. *In 2020 International Conference on ICT for Smart Society (ICISS)*, 1–6.

JIANG, L., CHEN, H., DENG, F., & ZHONG, Q. (2011). A Security Evaluation Method Based on Threat Classification for Web Service. *JOURNAL OF SOFTWARE, VOL. 6, NO. 4*, 595-603.

Kapil, M. J., Manoj, V. J., & Jay, L. B. (2016). A Survey on Man in the Middle Attack. *IJSTE-International J. Sci. Technol. Eng, 2(09)*, 277-280.

Lublinsky, B. (2007). Defining SOA as an architectural style: Align your business model with technology. *Systems Journal*, 1-12.

Mahmood, Z. (2007). Service oriented architecture: potential benefits and challenges. *Proceedings of the 11th WSEAS International Conference on COMPUTERS*.

Marks, E. A. (2008). Service-oriented architecture: a planning and implementation guide for business and technology. John Wiley & Sons.

Natis, Y. (2003). *Service-oriented architecture scenario*. http://dx.doi.org/ adresinden alındı

Niknejad, N. e. (2020). Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation. Information Systems 91: 101491.

Nils Engelbertz, V. M. (2019). *Nurullah Erinola, and Jorg Schwenk. Security analysis of xades validation in the cef digital signature services (dss). .* Open Identity Summit 2019.

Overall, D. (2006). Have we been there before.

Prof., S. B. (2017). *Learning XML.* An Imprint of Laxmi Publications Pvt. Ltd.

Rizak., V. B. (2022). Developing the algorithm and soft- ware for access token protection using request signing with temporary secret. . *Eastern-European Journal of Enterprise Technologies, 1(9):115*.

Roy, T. F. (2000). *Anchitectural Style and the Design of Netword-based Software Architectures.* California: University Of California, Irvine.

Sima, E., & Raziye, H. H. (2013). Critical Factors in the Effective of Service-Oriented Architecture. *ACSIJ Advances in Computer Science: an International Journal, Vol. 2, Issue 3, No. , 2013,* 26-30.

Simon Rohlmann, C. M. (2022). Oops... code execution and content spoofing: The first comprehensive analysis of opendocument signatures. . *Ruhr University Bochum*.

Yongxin, F., & Qin, L. (2016). The distributed UDDI system model based on service oriented architecture. *In 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 585-589.

Zhao, Y. W. (2018). Blockchain-based UDDI data replication and sharing. *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD)).*

ÖZGEÇMİŞ

Mehmet CİNCİ

Eğitim

Derece	*Yıl*	*Üniversite, Enstitü, Anabilim/Anasanat Dalı*
Y.Ls.	2014	Maltepe Üniversitesi, Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı
Ls.	2016	Anadolu Üniversitesi, İktisat Fakültesi İktisat
Ö.Ls.	2000	Trakya Üniversitesi, Edirne Meslek Yüksek Okulu Bilgisayar Programcılığı
Lise	1997	Kağıthane Lisesi

İş/İstihdam

Yıl	*Görev*
2022 -	Kıdemli Yazılım Mühendisi, Sovos Digital Planet A.Ş.
2019 – 2022	Yazılım Uzamanı, Barsan Global Lojistik A.Ş.
2018 – 2019	Kıdemli Yazılım Uzamanı, ATEZ Yazılım Teknolojileri A.Ş.
2013 – 2017	Kıdemli Yazılım Uzamanı, Güler Dinamik Gümrük Müşavirliği

Kongre/Sunum/Bildiri/Konferans

"Icecet 20-22 July 2022 Czech Republic ," **Token Based Novel Approach To Web Service Security**., 2022.