ÖN BİLGİ

Bu belge

- https://www.exploit-db.com/docs/turkish/41888-[turkish]-web-services-penetration-testing.pdf

resmi adresindeki veya linkin kırık olması ihtimaline karşın alternatif olarak

- https://www.includekarabuk.com/kitaplik/indirmeDeposu/Siber_Guvenlik_Teknik_Makaleler/Teori/BaskalarinaAitMakaleler/Web%20Servislerine%20Y%C3%B6nelik%20S%C4%B1zma%20Testleri.pdf

adresindeki makaleye çalışılarak elde edilen notlarımı kapsamaktadır. Bu çıkarılan notlar belgemde alıntılar ve/veya kişisel ilavelerim mevcuttur.

0)

Bu yazıda kullanılan web servislerinin tespit edilmesiyle birlikte sık karşılaşılan teknik ve mantıksal açıklıklara değinilecektir.

(Page 4)

1)

Web Servisi Nedir?
Kullanıcı isteklerine karşılık XML, JSON cevapları dönen yapıya web servisi denir.

(Page 3)

2)

Web servisi denildiğinde akla gelen bazı kavramlar şunlardır:

- SOAP
- REST
- WSDL
- WADL

(Page 3)

3)

What is SOAP
SOAP (Simple Object Access Protocol) is a messaging protocol that allows web programs that run on disparate operating systems (such as Windows and Linux) to communicate using Hypertext Transfer Protocol (HTTP) and its Extensible Markup Language (XML). Since Web protocols are installed and available for use by all major operating system platforms, HTTP and XML provide an at-hand solution that allows programs running under different operating systems in a network to communicate with each other.

SOAP specifies exactly how to encode an HTTP header and an XML file so that a web program in one computer can call a web program in another computer and pass along information. That is, **SOAP is a standard for encoding messages in XML.** SOAP also specifies how the called web program can return a response.

SOAP is analogous to Remote Procedure Calls (RPC) that enables applications to call functions from other applications, running on any hardware platform, regardless of different operating systems or programming languages.

As a result, SOAP is a standard for web client and server applications so that they can communicate with each other.

(http://searchsoa.techtarget.com/definition/SOAP)

4)

What is SOAP (Cont.)

- SOAP is an application communication protocol.
- SOAP is a format for sending and receiving messages.
- SOAP is platform independent.
- SOAP is based on XML.
- SOAP is a W3C recommendation

It is important for web applications to be able to communicate over the Internet. The best way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this. SOAP provides a way to communicate between applications **as a service** running on different operating systems, with different technologies and programming languages.

(http://www.w3schools.com/xml/xml_soap.asp)

5)

What is REST
REST is an architecture style for designing networked applications. The idea is that, rather than using complex mechanisms such as SOAP, CORBA or RPC (Remote Procedure Calls) to connect between machines, simple HTTP is used to make calls between machines.

RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. Thus, REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations. REST is a lightweight alternative of mechanisms like RPC (Remote Procedure Calls) and Web Services (SOAP, WSDL, et al.).

(http://www.acme.com/phonebook/UserDetails/12345)

6)

A request example using web services and SOAP:

Web Service Data in Server:

```
--------------------------------------------------
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
 <soap:body pb="http://www.acme.com/phonebook">
  <pb:Get UserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:Get UserDetails>
 </soap:Body>
</soap:Envelope>
--------------------------------------------------
```

With REST, the query will probably look like this:

> http://www.acme.com/phonebook/**UserDetails**/12345

Note that, this request has no any parameter. It's just a URL. This URL is sent to the server using a simpler GET request, and the HTTP reply is the raw result data -- not embedded inside anything, just the data you need in a way you can directly use.

Note how the URL's "method" part is not called "GetUserDetails", but simply "**UserDetails**". It is a common convention in REST design to use nouns rather than verbs to denote simple resources.

(http://rest.elkstein.org)

7)

What is WSDL
Açılımı Web Service Description Language olan WSDL web servislerinin methodlarını, method tanımlamalarını, veri tiplerini,.. kullanıcıya sunan XML formatında yazılımış bir dildir. Genel olarak SOAP için kullanılır.

(http://www.javauzmani.com/web-servisleri-soap-uddi-wsdl/)

8)

What is WADL
Açılımı Web Application Description Language olan WADL tıpkı WSDL gibi web servislerinin tanımlamalarında kullanılan bir dildir. Genel olarak REST için kullanılır.

(Page 3)

9)

Hedef Sunucunun Web Servisini Keşfetme
Web uygulamalarının kullandıkları web servisler aşağıdaki yöntemlerle keşfedilebilmektedir:

a. Bir proxy yazılımı ile araya girerek elde edilen trafiğin incelenmesiyle
b. Arama motorlarına girilecek dork'lar ile
c. Crawling yazılımlarıyla
d. Fuzzing Testleri aracılığıyla

(Page 5)

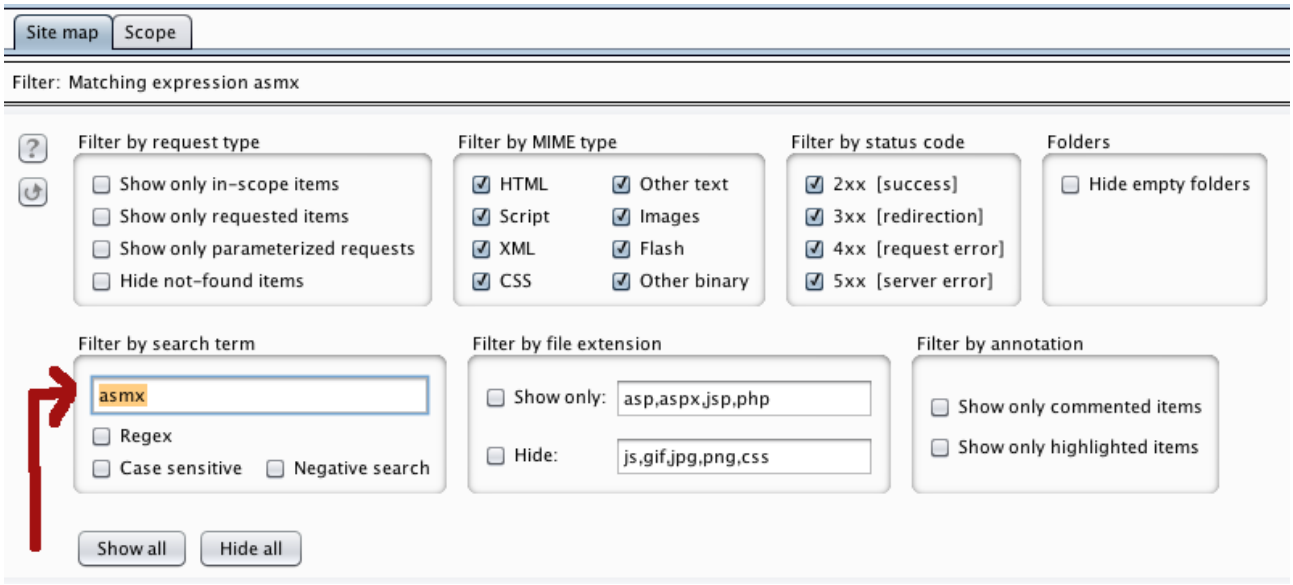
10)

What is Fuzzing Testing?
Fuzz testing or fuzzing is a software testing technique used to discover coding errors and security loopholes in software, operating systems or networks by inputting massive amounts of random data, called fuzz, to the system in an attempt to make it crash. If a vulnerability is found, a tool called a fuzz tester (or fuzzer) indicates potential causes.

(http://searchsecurity.techtarget.com/definition/fuzz-testing)

11)

Burpsuite ile Web Servis Keşfi
Aşağıda burpsuite kullanılarak capture edilen trafik üzerinde “asmx” keyword'ü ile basit bir filtre uygulanarak web servis tespitinde bulunulmaya çalışılmıştır.



Filtre olarak asmx yerine “.php?wsdl”, “.exe?wsdl”, “.dll?wsdl” veya “.ashx?wsdl” gibi keyword'ler de kullanılabilir.

NOT: aspx : Kontroller ve Event'ları içeren bir web sayfasıdır.
asmx: ASP.NET'te bir bilgi paylaşım methodu. yani bir web servisi türüdür.
ashx : ASP.NET'te generic (Üretken, dinamik) web sayfalarını handle eden custom bir hizmettir.

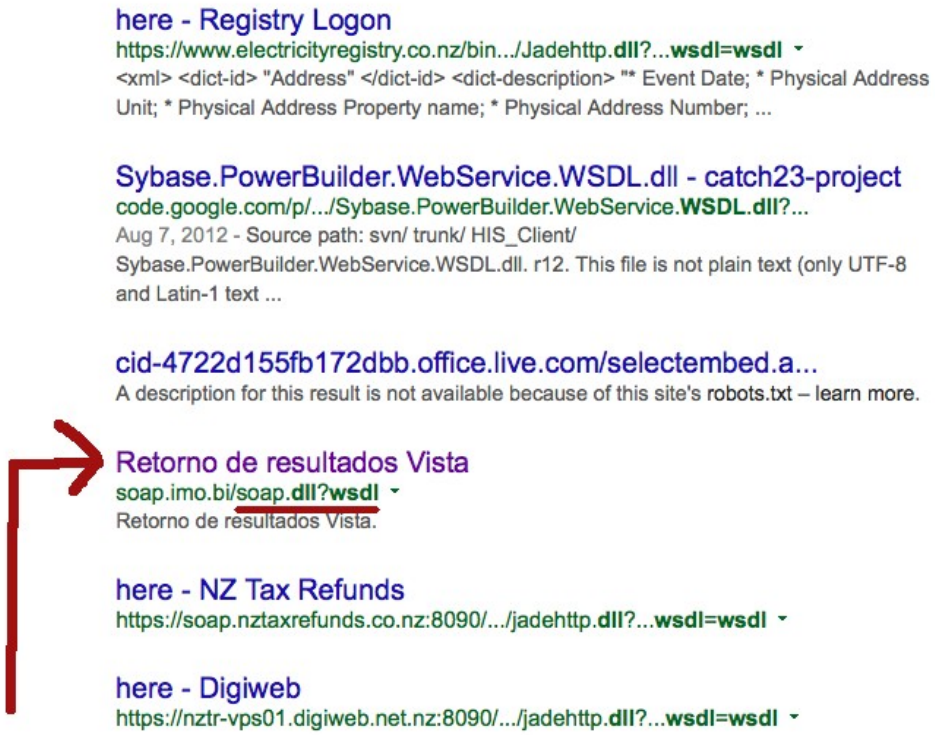
12)

Google Dorking ile Web Servis Keşfi
Aşağıdaki dork dosya türü asmx olan URL'de belirtilen keyword'lerden birini içeren web sayfalarını listeler.

> filetype:asmx inurl:(_vti_bin | api | webservice | ws)

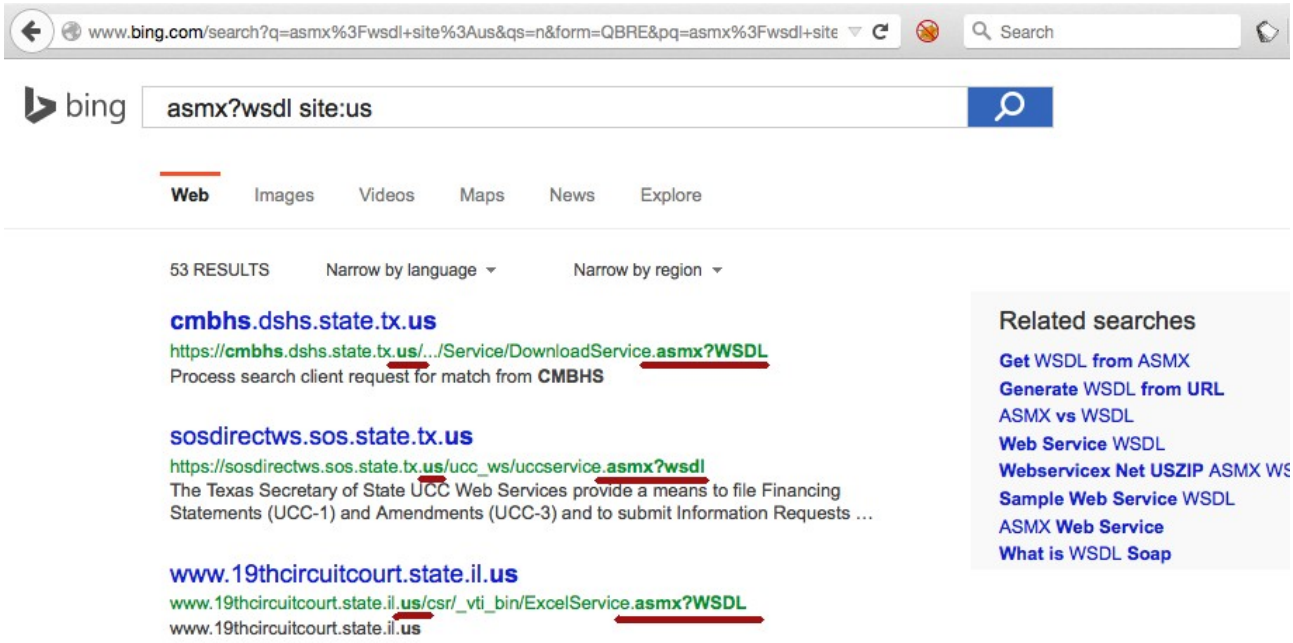
URL'sinde dll?wsdl parçacığını barındıran, dosya türü dll olan web sayfalarını listeler.

> allinurl:dll?wsdl filetype:dll



Bing arama moturunda da dork'lar (payload'lar) ile web servisi tespiti yapılabilir. Örneğin aşağıdaki payload url'sinde asmx?wsdl barındıran ve domain'i us'la biten web sayfalarını listeler.

> asmx?wsdl site:us



(Page 6-7)

13)

Fuzzing
Bir uygulamada zafiyet tespit edebilmek için uygulamaya rastgele veriler göndermeye ve gönderilen bu isteklere dönen cevapları analiz etmeye fuzzing (fuzz testing) denir. Bu işlemi yapmaya yarayan tool'lara ise fuzzer denilmektedir. Örneğin URL (dizin) keşfetmek için bir fuzzer olan wfuzz kullanılabilir. Wfuzz dışında bu işi OWASP DirBuster, nikto gibi tool'lar da yapabilmektedir.

NOT: wfuzz eski kali'lerde mevcut değil.

(http://webguvenligi.org/dergi/FuzzTesting-Agustos2009-OnurYilmaz.pdf)

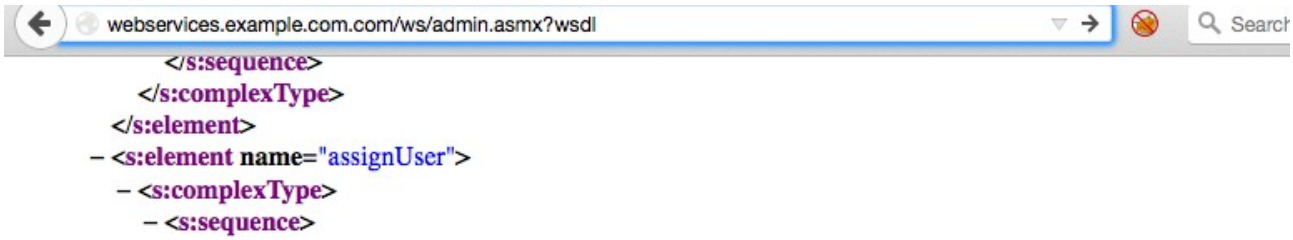
14)

Bir fuzzer olan wfuzz ile hedef web uygulamasının servisi keşfedilebilir. Bunun için web servislerinin kullandığı URL uzantılarından oluşan bir sözlük oluşturulması gerekmektedir ve wfuzz gibi bir tool ile bu sözlükten çekilen ve hedef sitenin URL'sine eklenerek yapılan taleplerin 200 OK yanıtı döndürmesine göre hedef web uygulamasının kullandığı web servisinin ne olduğu tespit edilebilir. Bu işleme aşağıdaki fuzzing örnek olarak verilebilir:

> wfuzz -p 127.0.0.1:8080 -c --hc 404,XXX -z list,ws-webservice-webservisler -z file,../general/common.txt -z file,ws-files.txt http://webservices.example.com/FUZZ/FUZ2ZFUZ3Z

[Yukarıdaki parmetrelerin anlamını ve wfuzz'ın kullanımını merak ediyorsan araştırıver]

Fuzzing sonucu keşfedilen url sayesinde hedef sistenin kullandığı web servisi teknolojisi aşağıdaki resimde göründüğü gibi öğrenilebilmiştir:



NOT: Url .asmz?wsdl ile bitiyor.

(Page 8)

15)

Web servis testleri için kullanılabilecek araçlarların listesi aşağıdaki gibidir:

- WebScarap
- SoapUI
- WCFStorm
- SOA Cleaner
- WSDigger
- wsScanner
- Wfuzz
- RESTClient
- BurpSuite
- WS-Attacker
- ZAP
- Metasploit
- WSDL Analyzer

(Page 13)

16)

Web Servislerinin kullandığı methodlara (POST'a, GET'e, PUT'a, CONNECT'e,...) ait parametreler manipule edilerek çeşitli açıklar bulunabilir.

(Page 10)

17)

http://zero.webappsecurity.com/web-services/infoService?wsdl

adresli web servisine yapılacak sızma testi aşamalarını ve ondan önce bu işle alakalı background'ı sayfa 10'dan 15'e kadarki sayfalardan okuyabilirsin. Olay tetkik edilemediğinden buraya not alınmamıştır.

18)

Bir Web Uygulamasında keşfedilen “Kullanıcı Hesaplarının Tahmin Edilebilmesi (User Enumeration yapılabilmesi) veya “Yerel Dizin İfşası (Full Path Disclosure) bulgusu gibi açıklara bir Web Servisi içerisinde de rastlanabilmektedir. Yani Web Uygulamasında olabilecek bir açık aynı zaman da Web Servisinde de olabilmektedir. Dolayısıyla Web Servisi testleri yaparken Web Uygulama Testleri de mutlaka yapılmalıdır.

(Page 16)

19)

Web Servislerinde SQL Injection Zafiyeti

● Web servislerinde olan SQL Injection zafiyetinin Web Uygulamalarında karşılaşılan SQL Injection zafiyetinden bir farkı yoktur.

● Web servislerinde SQL Injection (SQLi) zafiyeti tespitinde bulunabilmek için web servisine ait tüm fonksiyonların tüm parametreleri detaylı bir şekilde incelenmelidir.

(Page 17)

20)

Web Servisinde Adım Adım SQL Injection Arama
SQL zafiyeti için http://www.thomas-bayer.com/sqlrest/ adresinde bulunan ve vulnearable bir sistem olan RESTful web servisi örnek olarak kullanılacaktır. SQL zafiyetinin tespiti için Firefox üzerindeki RESTClient eklentisi kullanılacaktır.

http://www.thomas-bayer.com/sqlrest/CUSTOMER/$id adresinde id parametresi yerine bazı SQL payloadları gönderilecek ve dönen cevaplar yorumlanacaktır. Örneğin $id yerine 1, 2, 3 vb. şeyler girildiğinde ekrana veritabanından ilgili kullanıcılara ait bilgiler yansıtıldığı görülmektedir.

http://www.thomas-bayer.com/sqlrest/CUSTOMER/1
http://www.thomas-bayer.com/sqlrest/CUSTOMER/2
http://www.thomas-bayer.com/sqlrest/CUSTOMER/3

Mesela 23 değeri için aşağıdaki gibi bir çıktı ekrana gelmektedir.

NOT: Dikkat edersen 23 sayısı sanki bir dizinmiş (klasörmüş) gibi URL'ye eklendi. Yani index.php?id=23 şeklinde değil de /CUSTOMER/23/ şeklinde eklendi. Bunun nedeni hedef sitenin bir framework kullanıyor oluşundandır. Framework'lerde Controller içerisindeki fonksiyonların adı URL'ye böyle dizin ismi gibi gelir. CUSTOMER bir fonksiyon adıdır. Ondan sonra gelen 23 değeri ise işlenen bir parametre niteliği taşımaktadır.

23 yerine 23 AND 1 = 1 girildiğinde ise herhangi bir hata sayfası ile karşılaşılmamış, yani SQL sorgusundaki ilave edilen mantıksal ifadenin çalışmasına izin verildiği tespitine varılmıştır.

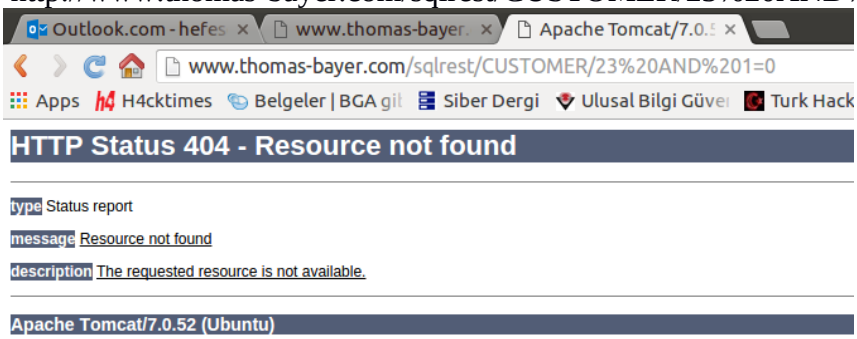
Payload : 23 AND 1=1

Site : http://www.thomas-bayer.com/sqlrest/CUSTOMER/23%20AND%201=1/

Payload olarak aşağıdaki kullanıldığında ise mantıksal açıdan SQL sorgu koşulunun sağlanmaması dolayısıyla 404 cevabının geldiği görülmüştür.

Payload : 23 AND 1=0

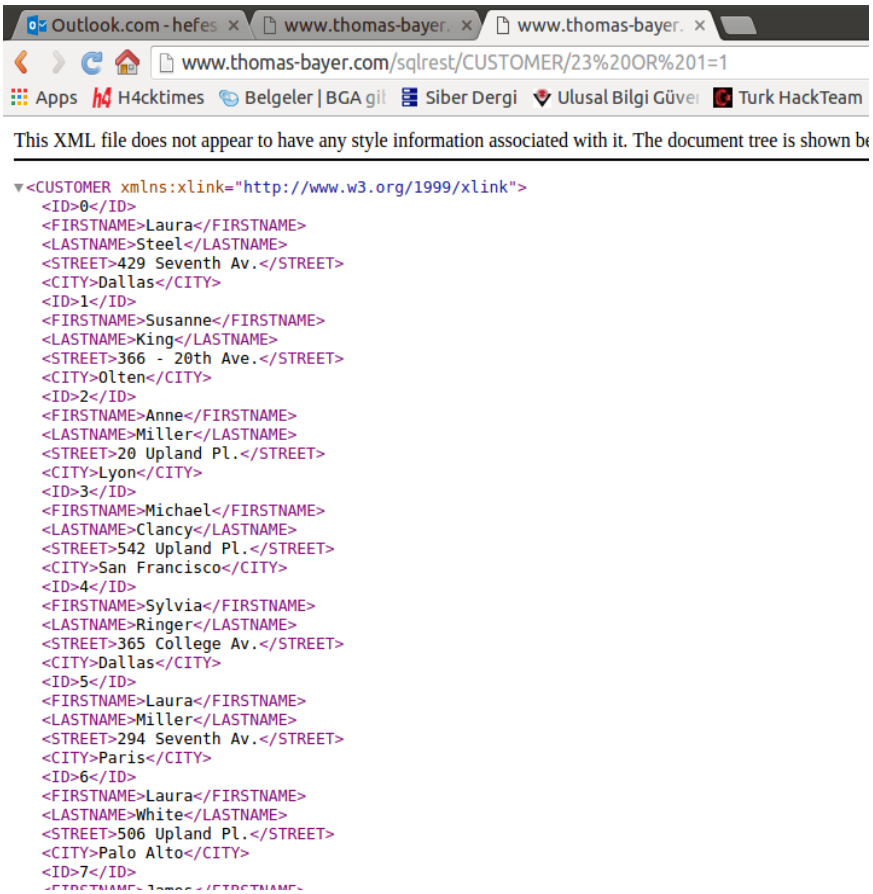
Site : http://www.thomas-bayer.com/sqlrest/CUSTOMER/23%20AND%201=0



Son olarak aşağıdaki payload gönderilmiş ve sistem üzerindeki tüm kullanıcı isimleri dönen cevap içerisinde görüntülenmiştir.

Payload : 23 OR 1=1

Site : http://www.thomas-bayer.com/sqlrest/CUSTOMER/23%20OR%201=1



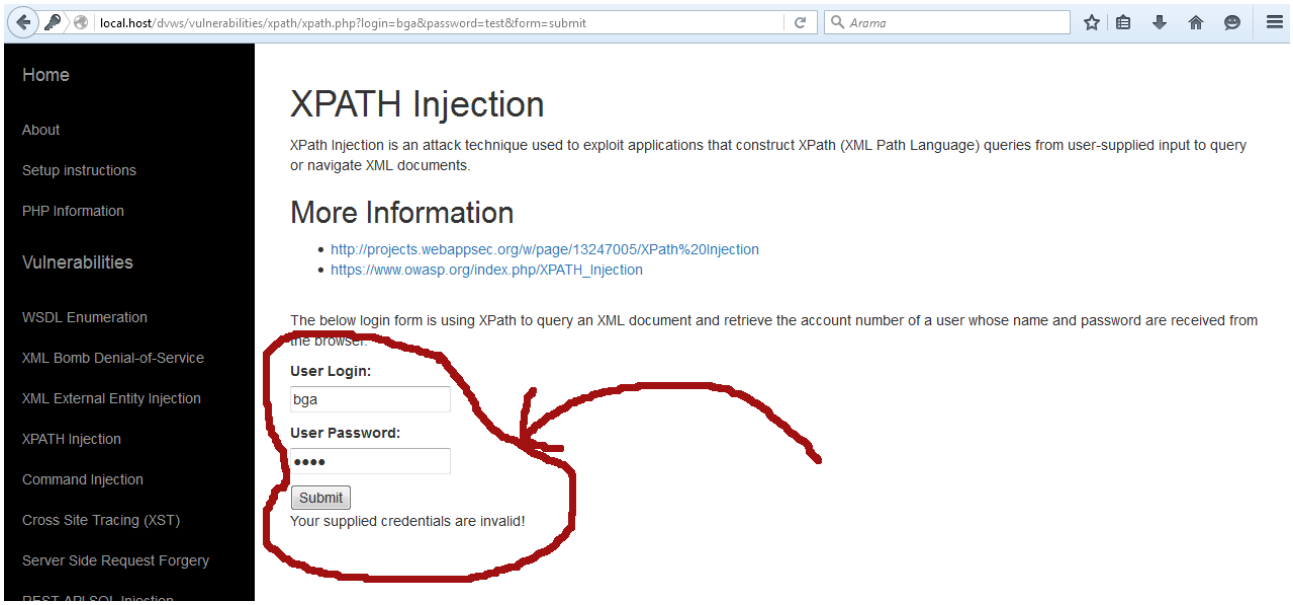
Bu şekilde manuel olarak SQLi (Sql Injection) tespiti için hedef sisteme basit payloadlar gönderilerek gelen yanıtlar incelenip ilgili web servis “fonksiyonunun” (CUSTOMER'ın) SQLi zafiyetini barındırıp barındırmadığı kanaatine varılabilir.

(Page 17-19)
21)

Xpath Injection Zafiyeti
Xpath sunucuda ikamet eden XML formatında saklı veriler üzerinde sorgulama işleminin yapılmasını sağlayan bir yan dildir. Bu dil üzerinden kaynaklanan zafiyeti anlamak adına

https://github.com/snoopythesecuritydog/dvws/

uygulamasını kullanalım ve uygulamanın sahip olduğu XPATH Injection zafiyetini sömürelim. Uygulamanın sunduğu login ekranı ve arayüzü şu şekildedir:



Bu login ekranının arkaplanındaki PHP kodlaması ise şu şekildedir:

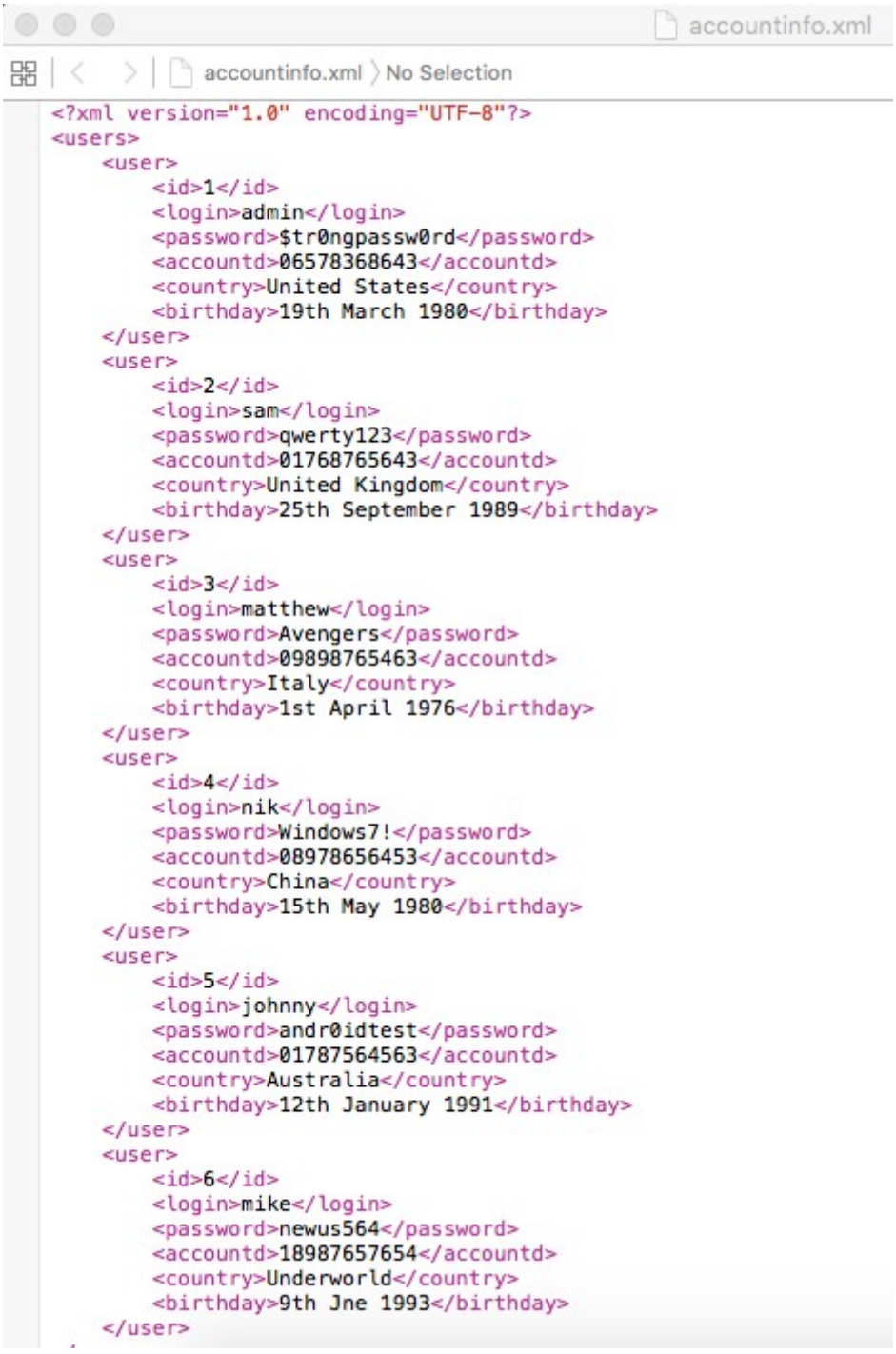
```
85
86
87  if(isset($_REQUEST["login"]) & isset($_REQUEST["password"]))
88  {
89  //take values from the HTML form
90      $login = $_REQUEST["login"];
91      $password = $_REQUEST["password"];
92
93
94      // Loads the XML file saved in the same directory
95      $xml = simplexml_load_file("accountinfo.xml");
96
97      // Executes the XPath search
98      $result = $xml->xpath("/users/user[login='".$login."' and password='".$password."']");
99
100     if($result)
101     {
102
103         $message =  "<br>Accepted User: <b>" . ucwords($result[0]->login) . "</b><br>Your Account Number: <b>" . $result[0]->accountd . "</b>";
104         echo $message;
105     }
106
107     else
108     {
109
110         $message = "Your supplied credentials are invalid!";
111         echo $message;
112     }
113    }
114 ?>
115 </html>
```

PHP kodlamasındaki seçili satıra dikkat edelim. Şöyle bir yapıya sahiptir:

string(//user[username/text()='girdi1' and password/text()='girdi2']/account/text())

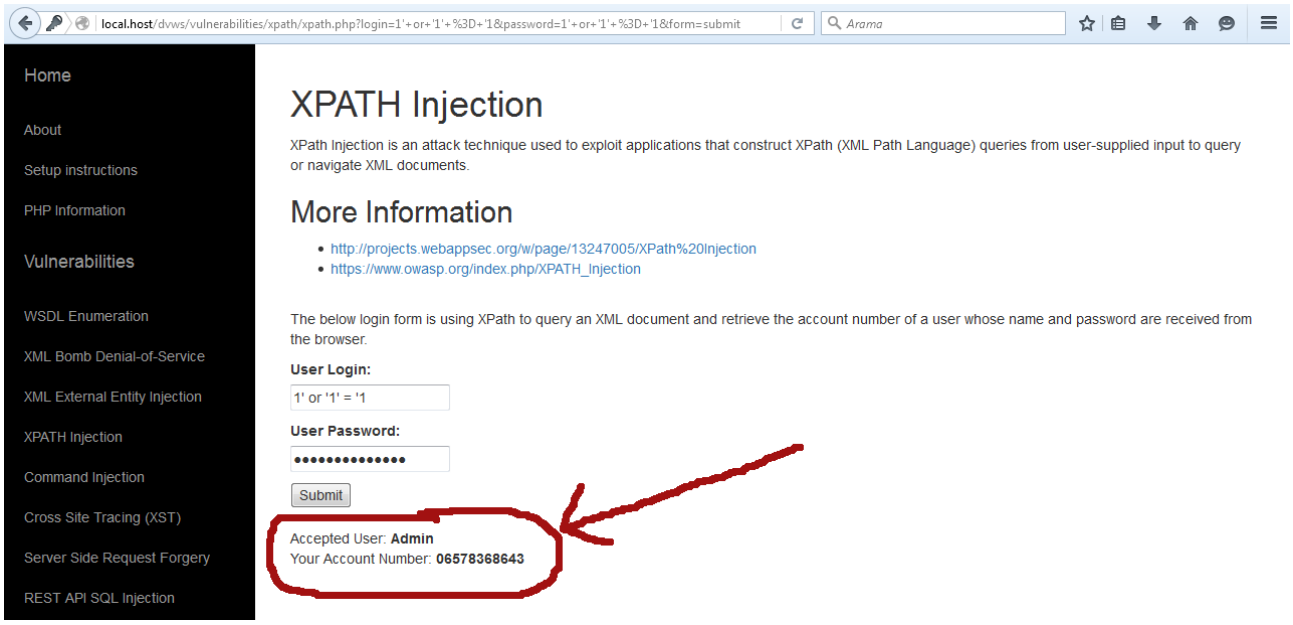
Bu bir Xpath sorgusudur. Nasıl SQL sorgusunun WHERE koşuluna kolon adı ve kullanıcıdan gelen girdi kıyaslamaları konuyorsa bu yukarıdaki Xpath'te de kıyaslanmak üzere girdiler konmaktadır. Bu sorguda kullanılan girdi alanlarına 1’ and ‘1’=’1 ve 1’ and ‘1’=’2 payload'ları gönderildiğinde bu sorgu eğer mantıksal işlem sağlanırsa “TRUE”, sağlanmazsa false yanıtını döndürecektir. Böylelikle kodun devamındaki if ya da else koşuluna girilerek oturumun açıldığı ya da açılamadığı bir arayüz bizi bekleyecektir.

Bahsettiğimiz bu Xpath sorgusu ve iç yapısındaki kıyaslamadan önceki satırlarda PHP kodlamasıyla $xml değişkenine “accountinfo.xml” adlı xml dosyasının içeriği yüklenmektedir. Bu xml dosyasının içeriği şu şekildedir:



Yukarıdaki xml dosyası göründüğü üzere login bilgilerinin depolu olduğu bir tür veritabanı gibi bir yapıya ve değerlere sahiptir. İşte Xpath bu veritabanı (xml) üzerinde sorgulama yapmaya yarayacaktır. Yani uygulamanın arayüzündeki login kutularına örneğin xml dosyasında barınmayan bir username değeri ve password değeri girersek ve button ile sayfayı tekrar yükletirsek PHP kodlaması ve Xpath yardımıyla $xml değişkenine yüklenen xml dosyası taranacaktır ve eşleşme bulunduğunda if koşuluna girilerek oturum açılacak, bulunamadığında ise else koşuluna girilerek oturum açılamayacaktır.

Eğer login ekranındaki metin kutularına SQL Injection'dan hatırladığımız 1' or '1' = '1 kodları girilirse aşağıdaki gibi oturum açılabildiği gözlenecektir.



1' or '1' = '1 kodu Xpath'in TRUE dönmesine sebebiyet verdiği gibi aynı zamanda XML dosyasındaki tüm kayıtlarla eşleşme var şeklinde bir sonucun $result değişkenine atanmasına da sebebiyet vermektedir. Yani tıpkı SQL sorgularında WHERE koşulunun 1' or '1' = '1 kodu ile iptal edilip ekrana tüm kayıtların basılması gibi Xpath'te de tüm kayıtların seçilmesine neden olmaktadır. Dolayısıyla Xpath'in true dönmesi sonrası ve $result değişkeninin tüm XML kayıtlarıyla dolması sonrası girilen if koşulunda $result değişkeninin sıfırıncı kaydı ($result[0]) oturum değişkeni olarak kullanıldığından xml dosyasında hangisi denk geliyorsa o kullanıcı ile oturum açılmış olacaktır. Yukarıdaki resimde 1' or '1' = '1 kodu ile oturumun Admin olarak açıldığını görmektesiniz.

Sonuç olarak Web Uygulamalarında gözlemlenen SQL Injection açığına Web Servislerinde Xpath Injection denmektedir. İkisi de temelde aynı zafiyeti ifade etmektedir. İkisi de input denetleme mekanizmasının olmayışlarından ya da yetersiz kalışlarından kaynaklanmaktadır. Fakat birisi veritabanı ve sql kullanırken diğer xml ve xpath kullandığından ayrı isimlendirmeye tabi tutulmuşlardır.

(Page 19-22)

22)

XML Injection
XML injection lafı xml formatında gönderilecek verinin içerisine ekstradan XML düğümü ilave etmeye denir. Aynı zamanda sunucudan gelecek xml formatındaki verinin manipule edilmesine de denmektedir. Örneğin bir web servisi düşünün ve metin kutusuna girdiğiniz sayı değerine göre ilgili xml düğümünün sunucudan geldiğini varsayın.

Input : 2

Response:

```
<product>
    <name>Computer</name>
    <count>2</count>
    <price>200$</price>
</product>
```

Görüldüğü üzere girilen 2 değerinin response içerisindeki <count> düğümünde yazdığı görülmektedir. Bu girdi değerine aşağıdaki gibi kodlar eklendiğinde response'la dönen <price> düğümünün değeri değiştirilebilir:

Input : 2</count> <price>0$</price> </product>

Response:

```
<product>
    <name>Computer</name>
    <count>2</count> <price>0$</price> </product>
```

Input'taki </product> tag'ı sayesinde geri kalan orijinal <price> ve </product> tag'larının sunucudan gelmemesini sağlamış olduk ve kendi <price> ve </product> tag'larını onların yerine koyabilmiş olduk.

(Page 22-23)

23)

XXE (XML External Entity) Zafiyeti
XML External Entity XML alışverişinin kullanıldığı web servislerinde Entity özelliğinin istismarına dayalı bir zafiyettir. Zafiyeti idrak edebilmek için bir örnek üzerinden gidelim. Diyelim ki hedef sunucuda şöyle bir PHP kodlaması mevcut:

```
<?php
$xml = simplexml_load_string(file_get_contents("php://input"));
echo $xml;
?>
```

Önce bu kodu adım adım açıklayalım. Kodlamadaki php://input kodu bir nevi $_POST'tur. Tıpkı $_POST gibi kullanıcıdan gelen veriyi çeker. Fakat $_POST json ve xml verilerini henüz handle

edebilecek kabiliyette olmadığı için php://input kullanımı daha münasiptir. simplexml_load_string fonksiyonu kullanıcıdan gelen xml verisini bir xml nesnesine dönüştürmeye yarar. $xml değişkenine ise bu oluşturulan nesne atanır. Son olarak echo $xml ile de $xml nesnesinin content'i ekrana basılır. Yani kullanıcıdan php://input ile gelen xml verisi echo ile ekrana basılmaktadır. Örneğin kullanıcının aşağıdaki xml verisini gönderdiğini varsayarsak.

```
<?xml version="1.0" ?>
<tag>test</tag>
```

ekrana “test” string'i yazdırılacaktır. İşte buradaki sıkıntı kullanıcıdan gelen verinin olduğu gibi ekrana basılmasıdır. Arada hiçbir filtreleme, denetleme vesaire yoktur. XXE açığı bu sebeple vardır. Eğer saldırgan böylesi bir PHP kodlamasına sahip sunucuya dosya içeriği okuyabilen bir XML Entity kodu gönderirse sunucunun yapacağı şey input olarak aldığı dosya dizinine bakıp ilgili dosyayı kendi sunucusunda bulup içeriğini ekrana basmak olucaktır. Örneğin;

```
<?xml version="1.0"?>
<!DOCTYPE tag [
    <!ENTITY bingo SYSTEM "file:///etc/passwd">
]>
<tag>&bingo;</tag>
```

yukarıdaki xml verisi saldırgan tarafından sunucuya gönderildiği takdirde sunucu ilgili xml'in content'ini ekrana basabilmek için önce content'i parse etmek isteyecektir. Çünkü content'te bir entity (&bingo;) vardır. Bu nedenle xml verisindeki <!ENTITY tag'ı sunucuda çalıştırılacaktır. Bunun üzerine /etc/passwd dizinindeki dosya okunacak ve içeriği &bingo; entity'sinin olduğu yere koyulacaktır. Son olarak bu content ekrana echo ile basılacaktır. Böylece saldırgan hedef sunucunun /etc/passwd dosyasını web taraycısından okuyabilmiş olacaktır. /etc/passwd okumak dışında daha kritik işlemlerde gerçekleştirilebilir. Örneğin dosya ismi yerine saldırgan hedef web sitesinin bir linkini koyabilir ve bu sayede hedef sitenin ilgili linkine ait sayfanın kaynak kodunu (PHP'li halini) ekranına basabilir. Bunun yanısıra saldırgan XXE açığı ile ENTITY üzerinden hedef sunucuda sistem komutu dahi çalıştırabilir.

```
<?xml version="1.0"?>
<!DOCTYPE tag [<!ENTITY bingo SYSTEM "expect://whoami">]>
<tag>&bingo;</tag>
```

Yukarıdaki ENTITY tag'ı hedef sunucuda whoami komutunu çalıştıracaktır ve dönen yanıt ekrana basılacaktır. Görüldüğü üzere bu işin sonu komple sunucunun hack'lenmesine kadar gidebilir.

XXE diye kısaltılan web servis zafiyeti eski olmasına rağmen Facebook gibi sosyal medya devinin sunucularında bulunmasıyla gündeme oturmuştur. XML üzerinden dosya okuma ve sistem komutu çalıştırma gibi özellikler aslında XML'in esnekliğini arttırmak için geliştirilmiş birer özelliklerdir, fakat bu esneklik önemli güvenlik risklerini de beraberinde getirmiştir. Bu özellikler üzerinden saldırıya uğramamak için ya kullanıcıdan gelen xml verilerini denetlemeye tabi tutmalıyız ya da bu esnek özellikleri xml kütüphaneleri yardımıyla kapatmalıyız.

Yararlanılan Kaynak : http://blog.bga.com.tr/2014/02/xxe-xml-external-entity-guvenlik.html

(Page 23-24)

24)

Web Servislerinde DoS Zafiyeti
İstemci tarafından gönderilen XML veriler sunucu tarafında bir XML parser yardımıyla çözümlenir ve işlenir. İki tip XML çözümleyici(parser) bulunmaktadır. Bunlar SAX tabanlı ve DOM tabanlı XML çözümleyicilerdir.

SAX tabanlı çözümleyiciler, çözümleme işlemi esnasında yapıları gereği bellekte maksimum 2 elementi tutmaktadırlar. Bu durumda servis dışı bırakma zafiyeti SAX tabanlı çözümleyiciler için söz konusu olmamaktadır.

DOM tabanlı çözümleyiciler, istemci tarafından gelmiş olan tüm XML veriyi belleğe alır. Bu sebeple bellek içerisinde çok büyük nesneler oluşmasına sebep olabilirler. Bu durumda, servis dışı bırakma saldırıları bu çözümleyici için kaçınılmaz olmaktadır. XML'e ait düğüm, attribute gibi değerlerin boyut ve adetlerinin kontrol edilmemesi durumu bunu tetiklemektedir.

Örnek 1 (Content ile DoS) :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <TEST>
      <BGABGABGABGABGABGABGABGABGABGABGABGABGABGABGABGA...
        ...1000'lerce BGA stringi buraya...BGABGABGABGABGABGABGABGABGABGA>
    </TEST>
  </soapenv:Body>
</soapenv:Envelope>
```

Örnek 2 (Attribute ile DoS) :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <TEST>
      <BGA attribute="BGABGABGABGABGABGABGAB..... BİNLERCE BGA STRINGI
      BURAYA....BGABGABGABGABGABGABGABGABGABGA"></BGA>
    </TEST>
  </soapenv:Body>
</soapenv:Envelope>
```

Örnek 3 (Entity ile DoS) :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE bga [
  <!ELEMENT ANY >
  <!ENTITY bga1 "bga1">
  <!ENTITY bga2 "&bga1;&bga1;&bga1;&bga1;&bga1;&bga1;">
  <!ENTITY bga3 "&bga2;&bga2;&bga2;&bga2;&bga2;&bga2;">
  <!ENTITY bga4 "&bga3;&bga3;&bga3;&bga3;&bga3;&bga3;">
  <!ENTITY bga5 "&bga4;&bga4;&bga4;&bga4;&bga4;&bga4;">
  <!ENTITY bga6 "&bga5;&bga5;&bga5;&bga5;&bga5;&bga5;">
]>
<bga>&bga6;</bga>
```

Örnek 3'te saldırgan bazı entity'ler tanımlamış ve en son durumda bga6'yı çağırmıştır. Çağırılan bga6 varlığı içerisinde bga5 varlığını 6 defa çağırmıştır. Aynı şekilde her bir bga5 varlığıda bga4 varlığını 6 defa çağırmıştır. Bu varlık ve çağırım adetleri daha da artırılarak böylesi bir payload gönderildiğinde XML parser'ın üzerine düşen yük çok fazla olacak ve sunucu bir zamandan sonra cevap veremeyecek duruma gelecektir. Böylece DoS saldırısı hedefine varmış olacaktır.

(Page 32-33)