

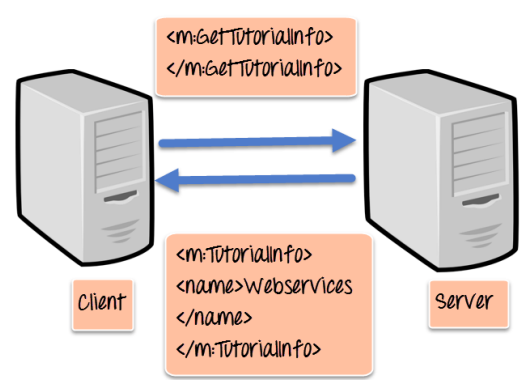
Web Servis Nedir?

İçindekiler

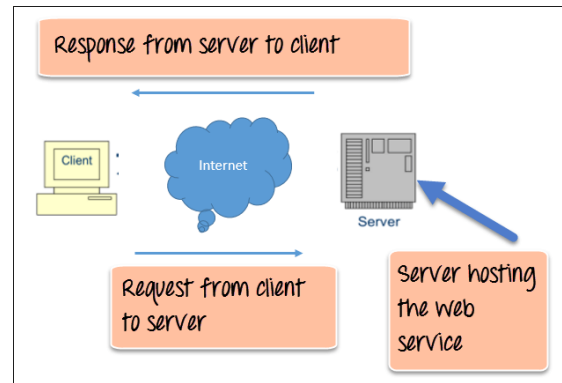
- Web Servis Tanımı
 - Web Servis Roller
 - Web Servis Çalışma Şekli
 - Web Servis Türleri
 - Web Servislerin Arayüzleri Olmaması Hk
 - Web Servis Arayüz / Kapsam Dosyalarının Güvenliğe Dokunan Yanı
 - API ve Web API (Web Servis) Arasındaki Fark
 - Web Servisleri Test Etme
 - Web Servis Ek Notlar
- Uygulama [SoapUI Yazılımı ile SOAP Web Servis Test Etme]
Uygulama [SoapUI Yazılımı ile SOAP Web Servis Test Etme 2]
Uygulama [SoapUI ile Rest Web Servis Test Etme]
Uygulama [Burpsuite WSDLER Eklentisi ile Soap Web Servis Test Etme]
Uygulama [Burpsuite ile Rest Web Servis Test Etme]
Uygulama [Netsparker Yazılımı ile Soap Web Servis Test Etme]
Uygulama [Netsparker Yazılımı ile Rest Web Servis Test Etme]
Uygulama [Chrome Web Tarayıcı Eklentisi İle Soap Web Servis Test Etme]
>>> Web Servisleri Saha Görevlerinde Test Etme Hakkında (Genel Değerlendirme)

a. Web Servis Tanımı

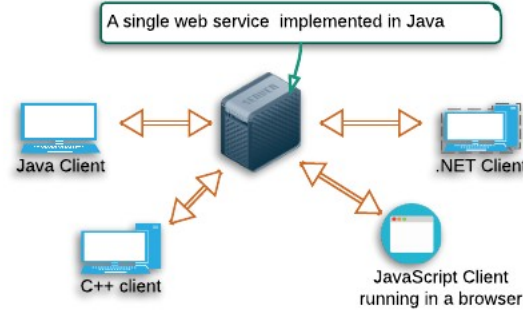
Web servisi web sunucularda yer alan, herhangi bir programlama dili (örn; php, java, c#,...) ile yazılabilen, herhangi bir web sunucu platformunda (örn; apache, iis, nginx, ... de) yer alabilen, farklı platformlardaki ve programlama dillerindeki (örn; windows-based, unix-based, web-based, mobile-based,... platformlardaki ve php, java, c#,... dillerindeki) uygulamaların XML mesajlaşma sistemiyle haberleşerek kullandıkları bir yazılım parçasıdır.



(Örnek Bir Web Servis Haberleşmesi)



(Web Servisler Web Sunucuda Host Edilir)



(Çeşitli İstemcilerin Web Servisle Haberleşmesi)

Web servisi bir programlama dilinin bir başka programlama diliyle haberleştiği bir teknolojidir. Yani web servisi teknolojilerin birbirleriyle bir arada olabildikleri bir yol sunar. Bunu XML ve JSON kullanımı ile yaparlar. Teknolojiler bu ortak platform bağımsız veri formatlarını kullanarak birbirleriyle haberleşebilirler. Sıralanan resimlerde sol tarafta web servis kullanan / tüketen tarafla web servis sunucusu tarafının haberleşme şekli, sağ tarafta web servislerin web sunucularda host edildiği bilgisi, aşağı tarafta ise çeşitli teknolojilerdeki web servisi kullanan / tüketen uygulamaların xml haberleşme şekliyle web servisle haberleşmesi gösterilmiştir.

Web servisi 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 ile konuşması veya iOS bir mobil uygulamanın php ile yazılmış bir web servis ile konuşması örnek olarak verilebilir, veya aynı türden teknolojilerin haberleşmesine dvws (damn vulnerable web services) web servisi kullanan / tüketen uygulamanın web-based bir web tarayıcıdaki php uygulaması olması ve web servisinin de php olması örnek olarak verilebilir.

b. Web Servis Roller

Web serviste

- service requestor ve
- service provider

adlı roller vardır. Bu rollerden service requestor web servisi tüketendir / kullanandır. Herhangi bir platformda / ortamda yer alabilir. Service provider ise web servisi sağlayan ve internetten erişilir kılındır. Web sunucuda yer alır. Service requestor örneğin web-based olabilir, windows-based olabilir, unix based olabilir,..., farklı türden platformlara özgü uygulamalar olabilir. Service provider ise örneğin web sunucudaki java bir uygulama olabilir, asp.net bir uygulama olabilir, php bir uygulama olabilir. Service requestor'lar service provider'a ağda bağlantı açarlar ve XML talepleri yollarlar. Yanıt olarak da XML yanıtları alırlar.

c. Web Servis Çalışma Şekli

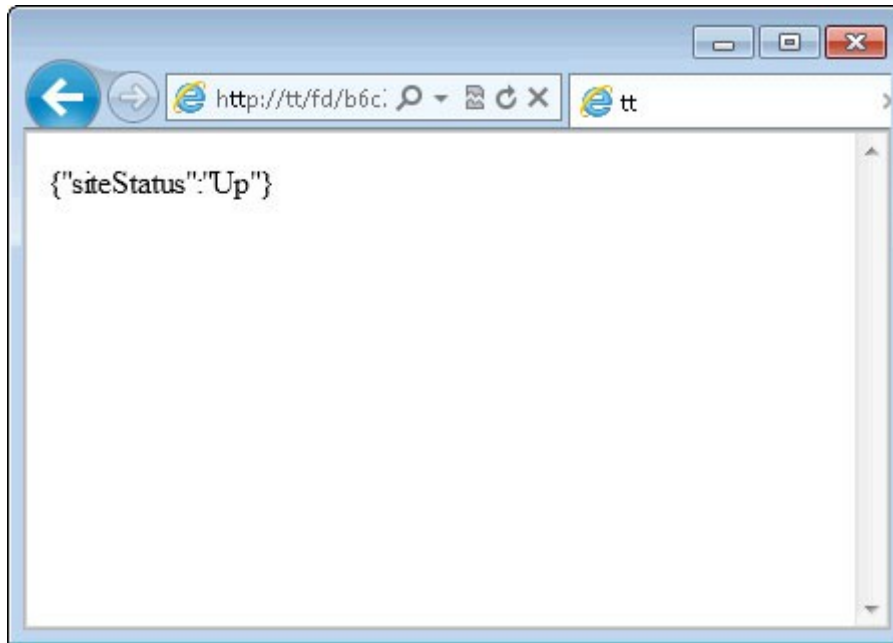
Service requestor'lar service provider'la xml talepleri ve yanıtlarıyla konuşurlar. Örneğin aşağıda php ile yazılmış bir web servisinin, yani service provider'ın web sunucudaki bir kod bloğu gösterilmektedir.

```

"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)
?>

```

Kodlamaya 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 yanıt olarak gönderiliyor.



Burada service requestor web based bir uygulama olduğundan tarayıcıda yanıt görüntülenmektedir. Web uygulamalarda yanıt paketleri html dökümanı döner iken web servislerde xml / json dönülür. Bu örnekte web servisindeki bu türden dönüş için `print_json_encode()` metodu kullanılmıştır.

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 ikinci örnek için service requestor'ın gönderdiği geçerli hesap bilgilerindeki login olma paketi ve aldığı yanıt örneği,

Örnek Http Request 1:

```
POST /mockServiceSoapBinding HTTP/1.1
...
...
...
...
<soapenv:Header/>
<soapenv:Body>
<sam:login>
<username>Login</username>           // Geçerli Credential
<password>Login123</password>       // Geçerli Credential
</sam:login>
</soapenv:Body>
</soapenv:Envelope>
```

Örnek Http Yanıtı 1:

```
... // Headers
... // Headers
... // Headers
... // Headers

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

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

Örnek Http Request 2:

```
POST /mockServiceSoapBinding HTTP/1.1
...
...
...
...
<soapenv:Header/>
<soapenv:Body>
<sam:login>
<username>Login</username>           // Geçersiz Credential
<password>deneme</password>       // Geçersiz Credential
</sam:login>
</soapenv:Body>
```


Örnek Http Yanıtı 2:

```
... // Headers
... // Headers
... // Headers
... // Headers

<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 uygulamalarda yapılan her işlem yapılabilir. Web servisler bu işlevleri xml (veya json) etiketleri üzerinden gerçekleşen alışverişle yaparlar. Dolayısıyla web uygulamalardaki brute force, sql enjeksiyonu, komut çalıştırma,... gibi saldırılar web servislerde de yapılabilir ve web uygulamalarda çıkan açıklıklar web servislerde de çıkabilir.

d. Web Servis Türleri

Web servisleri ağırlıklı bir şekilde kullanıma göre iki türdür. Bunlar,

- SOAP Web Servisi
- REST Web Servisi

şeklinde. Bu iki web servisi piyasadaki en baskın iki türdür. SOAP web servisi zaman zaman 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 servislere 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.

e. Web Servislerin Arayüzleri Olmaması Hk

Web servisler kendi başlarına bir arayüze sahip değildirler. Örneğin web uygulamalarda gelen yanıt paketlerinde spidering / crawling yapılarak arayüz / kapsam belirlenir. 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 tarayıcılarının 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 optional (seçime bağlı) olarak WSDL dosyası kullanılır. 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 yanıt olarak bekleniyor, ... gibi bilgileri öğrenilir. Bu şekilde kapsam elde edilir ve saldırı testleri veya fonksiyonel çalışırılık testleri uygulanabilir. SOAP web servisler WSDL'e bağımlıdır ve SOAP web servislerini keşfetmek adına başka bir mekanizmaya sahip değildirler. Yani SOAP'ta WSDL tek çözümdür. Fakat soap web servislerin WSDL kullanımını zorunlu değildir.

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 her bir 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 dökümantasyon yapmak 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 bir 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 gönderildiğini, hangi status code'un yanıt olarak beklendiğini, ... v.b. bilgilerini öğrenir. Böylece rest web servisler bu şekilde arayüzleri / kapsamları keşfedilebilir olur ve saldırı testleri ve ayrıca fonksiyonel çalışırılık 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.

f. Web Servis Arayüz / Kapsam Dosyalarının Güvenliğe Dokunan Yanı

Web servislerde soap için wsdl veya rest için wadl, openapi (swagger),... gibi arayüz / kapsam dosyalarının public olarak web servis adresinden erişilebilir olması güvenlik riski oluşturmaktadır. Bu durum örneğin soap web servisleri WSDL Enumeration adı verilen açıklığa karşı savunmasız bırakacaktır. Çünkü public olarak arayüz / kapsam dosyalarının erişilebilir olması saldırganların gizli fonksiyonları bulabilmesine yol açacaktır ve muhtemel hassas verilere erişebilmesine yol açabilir. Bu nedenle arayüz / kapsam dosyaları public olarak erişilebilir olmamalıdır. Arayüz / kapsam dosyasına erişim kısıtlanmalıdır ve arayüz / kapsam dosyalarında gereksiz tanımlamalardan sakınılmalıdır.

g. API ve Web Servis API Arasındaki Fark

API (yani Application Programming Interface) birbiriyle benzerliği olmayan iki uygulamanın haberleşebilmesini sağlayan yazılım bileşenine denir. Eğer API veri göndermek için network'ten yararlanıyor ise bu API'lere de web API denir. API ile Web API arasındaki fark Web API'nin bir API alt türü olmasıdır. Yani API tanımsal olarak daha geniş bir kategoriye tekabül eder. Kök kategoridir. Web API ise bir API türüdür.

API ve web API farklı uygulamalar arasında veri transferi için kullanılan iki teknolojidir. API bir uygulamanın verilerini dış uygulamalara sunan bir arabirimken web API bir uygulamanın verilerini dış uygulamalara sunan daha katı gereksinimlere sahip bir arabirimdir. Web API'nin sahip olduğu bu gereksinimler network üzerinden iletişiminin olması, "birincil" iletişim protokolü olarak SOAP kullanılıyor olması ve genelde public olarak daha az erişime izin veriyor olmasıdır.

Aşağıda API türü olan web API'nin sahip olduğu türler listelenmiştir.

Web API Türleri:

- Open APIs
- Partner APIs
- Internal APIs
- Composite APIs
- Web service APIs (*)

Web Servis API bir web API'dir. Web Servis API'ye örnek olarak şu popüler türler verilebilir.

Web Servis API Türleri:

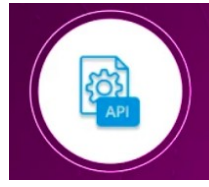
- SOAP
- XML-RPC
- JSON-RPC
- REST

Tüm web API'ler birer API'dirler, fakat her API web API değildir. Çünkü web api dışındaki api'ler network ile haberleşmezler. Lokalden haberleşirler.

Aşağıda web api ve api arasındaki network farkını göstermek adına temsili resimler verilmiştir:



Web API



API

Sonuç olarak api ve web api (daha spesifik ifadeyle web servis api) arasındaki fark network'ün varlığıdır.

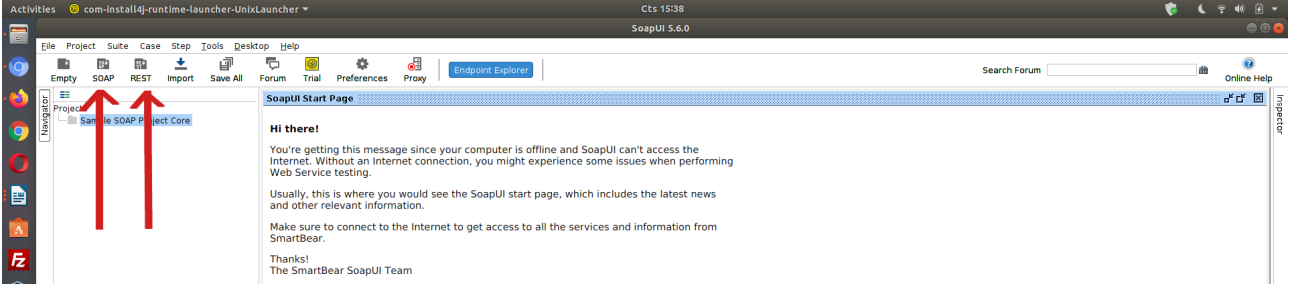
h. Web Servisleri Test Etme

Web servisler çeşitli web zafiyet tarayıcı araçlarla test edilebilir. Örneğin soap ve rest web servisler için geliştirilmiş SoapUI (Soap User Interface) yazılımı, Burpsuite WSDLER plugin'i, Netsparker veya web tarayıcı eklentileri gibi araçlar test uygulayabilirler.

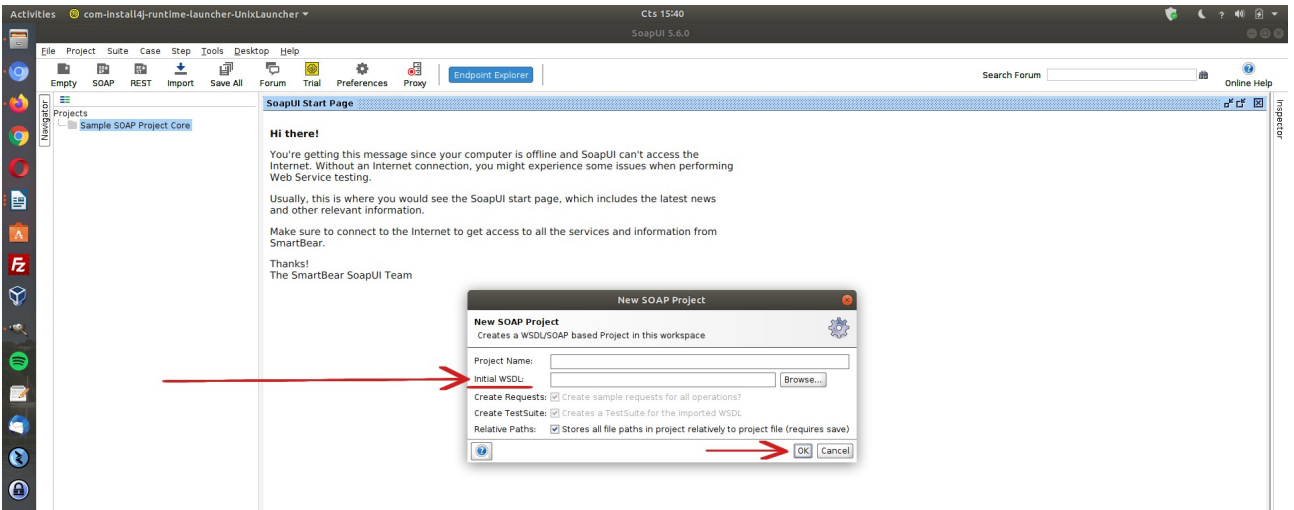
* SoapUI

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

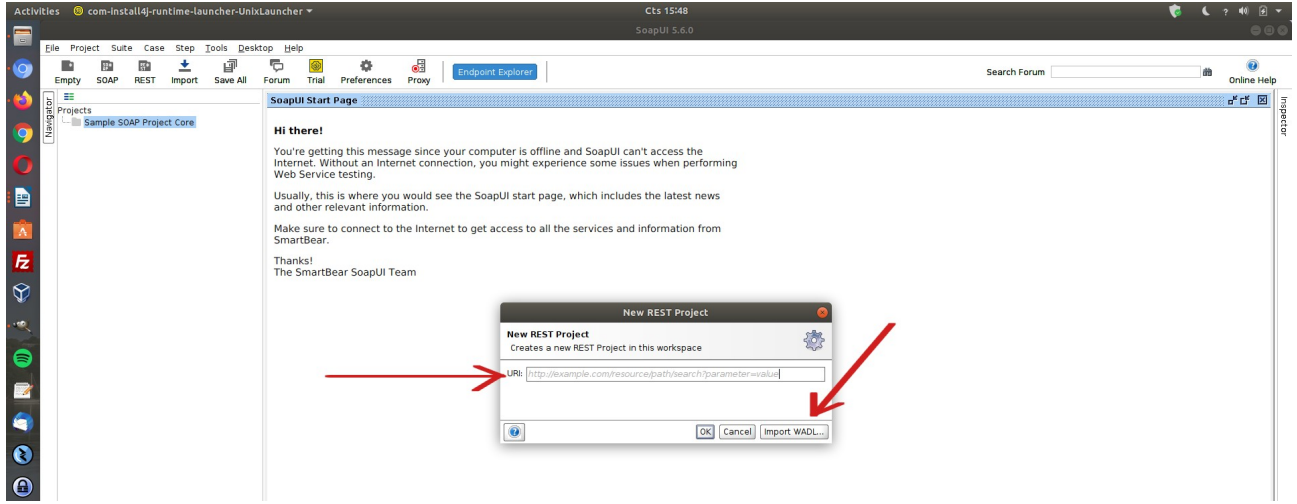
SoapUI'de Soap ve Rest web servisleri test etme şu şekilde gerçekleşir:



Üst menüde SOAP ve REST butonları vardır. SOAP web servisi test edileceği zaman SOAP'a tıklanır ve açılan pencerede WSDL 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 dendindiğinde 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 seçime bağlı olarak WADL dosyası istenmektedir.

WSDL: Web Services Description Language // SOAP'ın Definition Dosyası
WADL: Web Application Definition Language // REST'in Definition Dosyası

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

* Burpsuite

Burpsuite'te wsdler eklentisi yardımıyla soap web servislere ve url adresi yoluyla rest web servislere güvenlik testi uygulanabilmektedir.

Soap web servisler için burpsuite wsdler 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 wsdler 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 Wsdler 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.

Not: Burpsuite wsdler 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 proxy ayarıyla burp'te açarak bu sorunu aşabiliriz ve test uygulayabiliriz. Burpsuite wsdl eklentisi SoapUI ihtiyacını kaldırmak için, ve SoapUI olmadan burpsuite'te soap web servisleri test etmek için geliştirilmiş bir eklentidir.

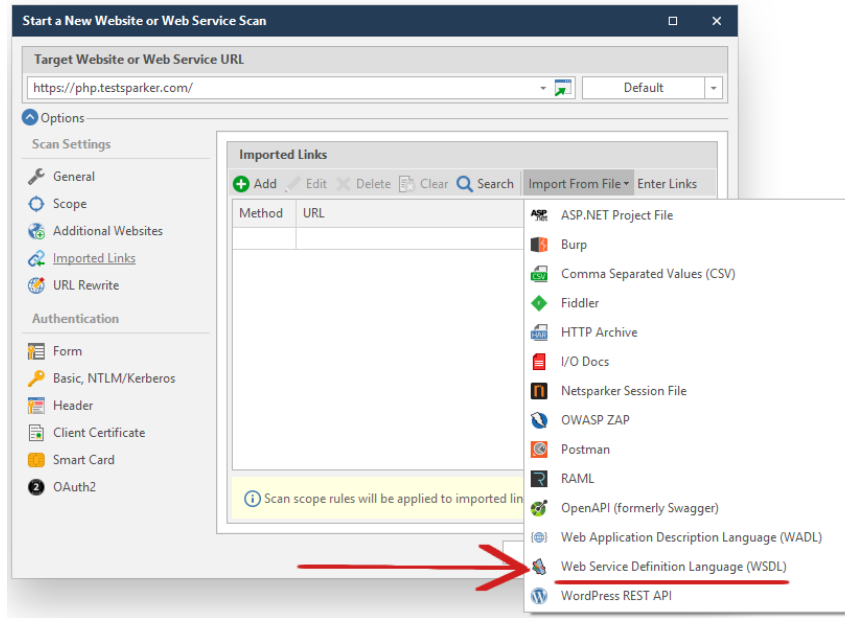
Rest web servisler için burpsuite'te rest web url adresi üzerinden güvenlik testi yapılmaktadır. Bunun için rest web servisin url listesine ihtiyaç vardır. Rest web serviste bir url için talep paketini alıp paket başlıkları ve gövdesini kurcalayarak repeater'da, fuzzing ile intruder'da güvenlik testleri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

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

* Netsparker

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.

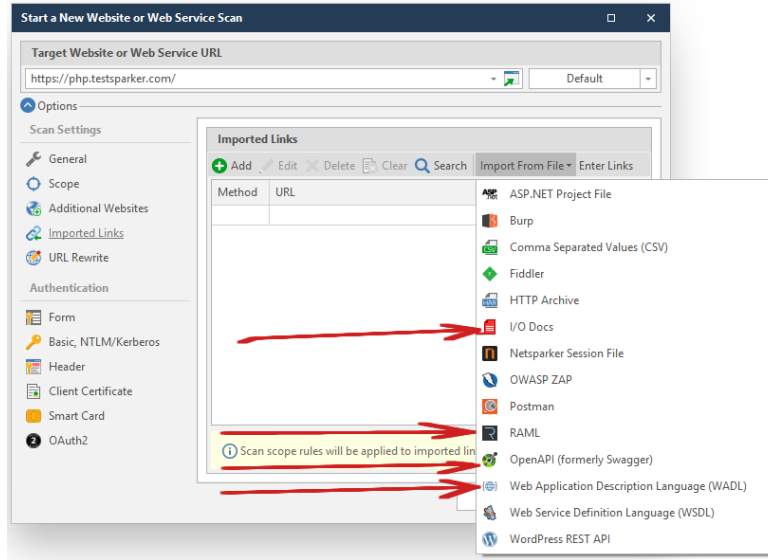
Örneğin soap web servisler Netsparker'da Import Links seçeneğinden WSDL dosyasını okutturarak arayüzün / kapsamın dahil edilmesiyile taranabilmektedir.



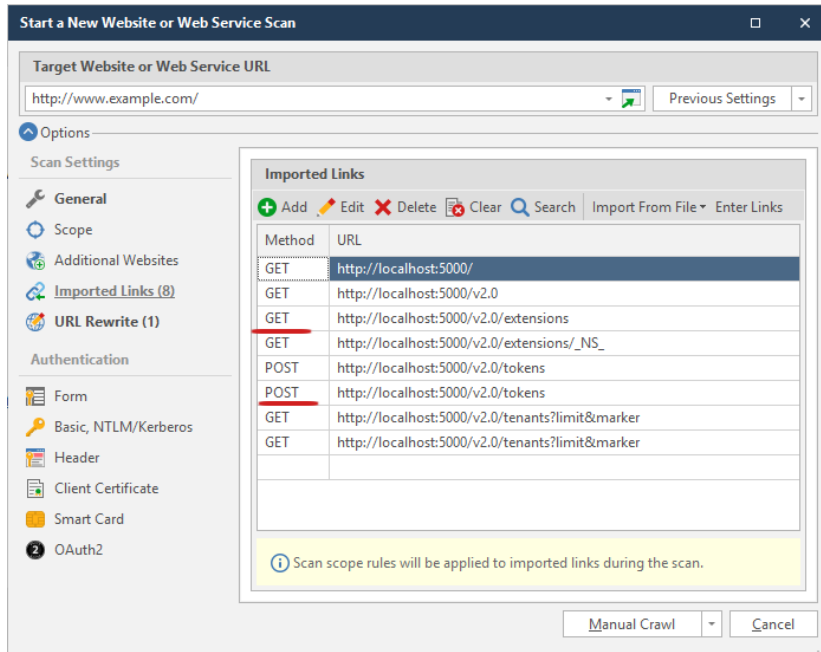
Örneğin REST web servisler netsparker'da şu üç yolla taranabilir:

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

- 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)

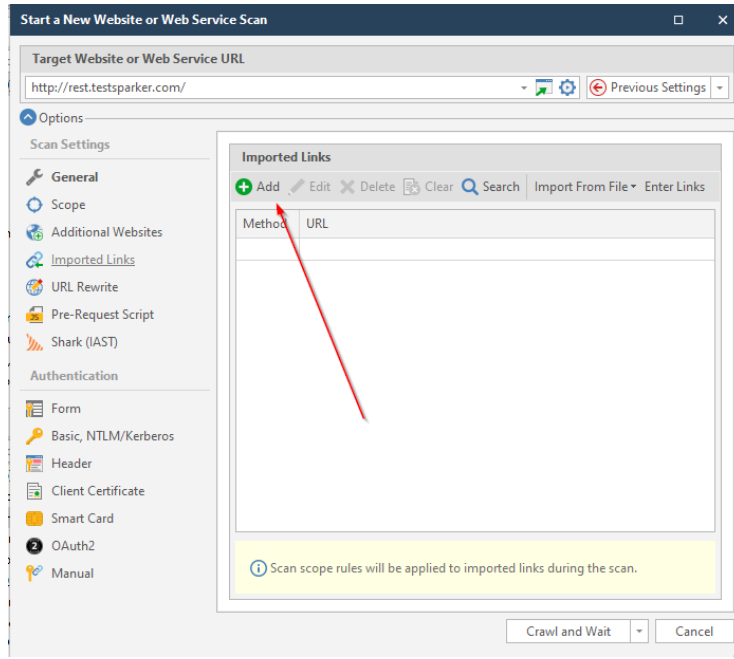


(URL Listesi Belirlenmiş ve Taramaya Hazır REST Web Servisi Örneği)

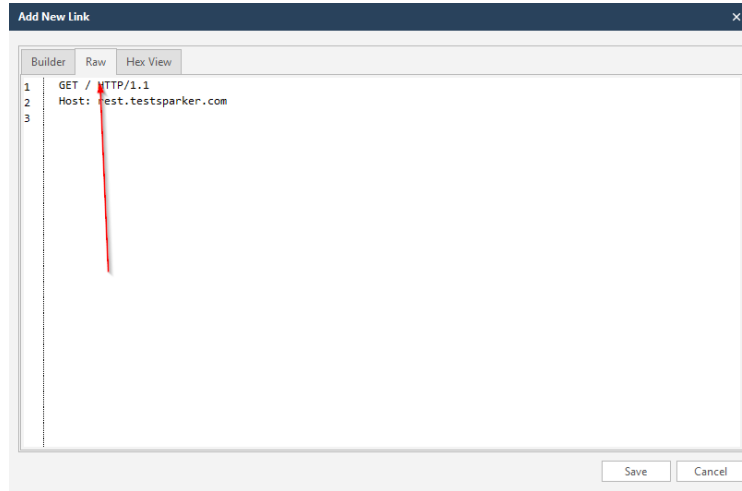
URL listesi belirlenmiş rest web servisi bu şekilde taranabilecektir.

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

- 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. Ayrıyetten 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 testleri pratik uygulaması bu dökümanda uygulama başlığı altında gösterilmiştir.

* 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 talepleri kullanılarak testler yürütülebilir.

* Wireshark

Web servisi tüketen / kullanan uygulamanın trafiği dinlenerek ve hedef web servis adres filtrelemesine gidilerek web servis talep paketleri “follow tcp stream” ile elde edilebilir ve bu şekilde ham paket burpsuite’e örneğin atılarak güvenlik testleri yürütülebilir veya Netsparker’a örneğin atılarak sadece belirtilen ham paket üzerinden otomatize saldırı testleri uygulanabilir. Bu yöntemde kapsam elde edildiği kadardır ve testler elde edilebilen kapsam kadar uygulanır.

Bir Tecrübe:

Bir saha görevinde *SK’da bilgisayarda wireshark açılmıştır ve akan paketler içerisinde post metotlu bir paket görülmüştür. Bakıldığında xml gövdeli bir paket olduğu anlaşılmıştır ve soap xml node’ları yer almaktadır. O paketteki soap xml node’larından bilgisayarın arkasında uzaktaki bir web servisle konuştuğu paket olduğu anlaşılmıştır. Pakete sağ tık yapıp Follow->TCP Stream yaparak talep paketi kopyalanmıştır ve Burpsuite’te repeater’a yapıştırılmıştır. Paket burp’te repeater’da gönderildiğinde yanıt olarak başarılı bir soap yanıt mesajı dönmüştür. Dolayısıyla pakete burpsuite’te sağ tık yapıp Intruder yapılarak testler genişletilebilir veya Scan ile otomatize tarama yapılabilir. Bu şekilde wireshark’tan elde edilen paketler kapsamınca Burpsuite’te testler yürütülebilir. Veya paket Netsparker’a atılarak otomatize testler yürütülebilir.

* Proxy

Web servisi tüketen / kullanan uygulamanın haberleştiği web servis adresi için işletim sistemi /etc/hosts dosyasından karşılık olarak localhost girilerek ve uygulama trafiği lokalde burpsuite’e bu şekilde yönlendirilerek uygulamadan gelen xml talep paketlerini kurcala ve gönder ile güvenlik testleri yürütülebilir. Bu yöntemde kapsam elde edildiği kadardır ve testler elde edilebilen kapsam kadar uygulanır.

i. Web Servis Ek Notlar

- Burpsuite WSDL Wizard eklentisi hedef soap web servisinde wsdl keşfi çalışması yapmaktadır.
- ASP.NET web servisler web sunucuda .asmx uzantısı kullanılmaktadırlar.
- Endpoint URL web servisin nerede erişilebilir olduğunu ifade eder.

Uygulama [SoapUI Yazılımı ile SOAP Web Servis Test Etme]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

Bu uygulamada SoapUI kullanılarak SoapUI tutorials'lardaki demo soap web servisi projesi açılacaktır ve soap web servisi testi yapılacaktır.

Gereksinimler

Ubuntu 18.04 LTS	// Ana Makine
SoapUI	// Web Servis Test Yazılımı
Burpsuite	// Proxy Yazılımı

SoapUI yazılımının kurulumu şu şekildedir:

// Download

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

(Kurulum dosyası Downloads klasöründe yer almaktadır: SoapUI-x64-5.6.0.sh)

// Install

Ubuntu 18.04 LTS Terminal:

```
> sudo su
> chmod +x SoapUI-x64-5.6.0.sh
> ./SoapUI-x64-5.6.0.sh
```

(Kurulumda home dizinine kurulum yapılır, ve Tutorials da yüklemeye dahil edilir.
Tutorials'da soap ve rest için örnek soapui projeleri mevcut)

Kurulum sonrası yazılımı başlatma şu şekildedir:

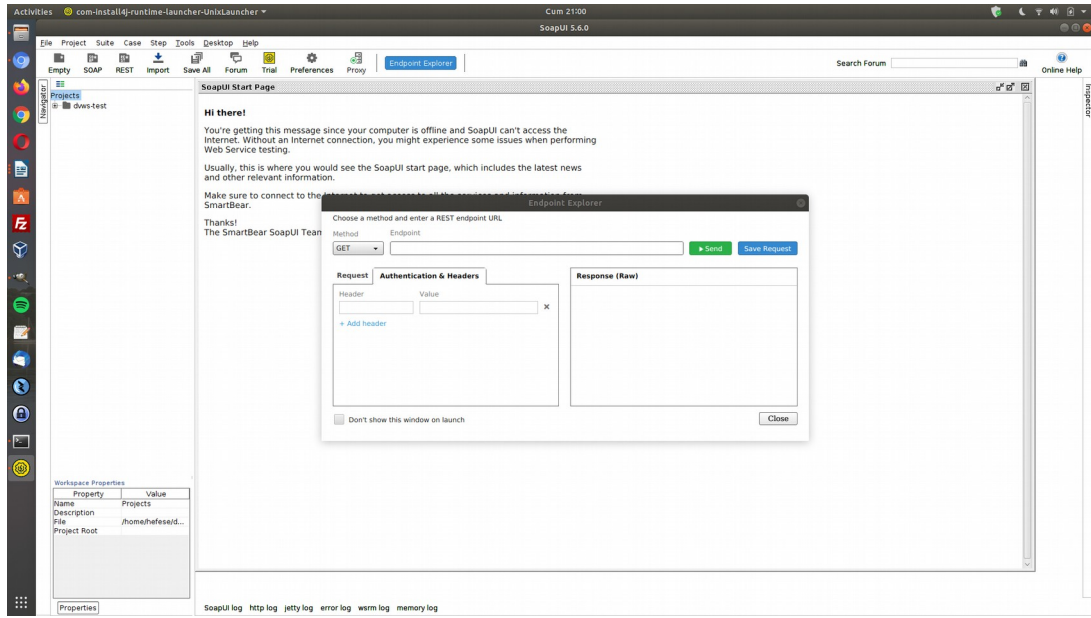
// Run

Ubuntu 18.04 LTS:

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

(Email kaydı skip'lenebilir)

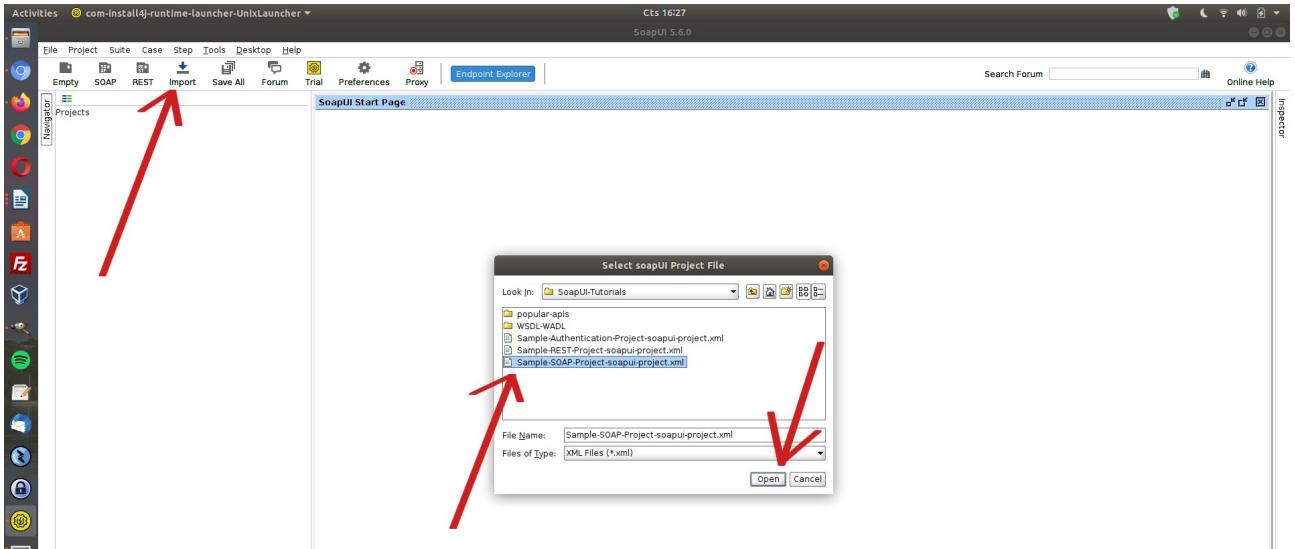
Çıktı:



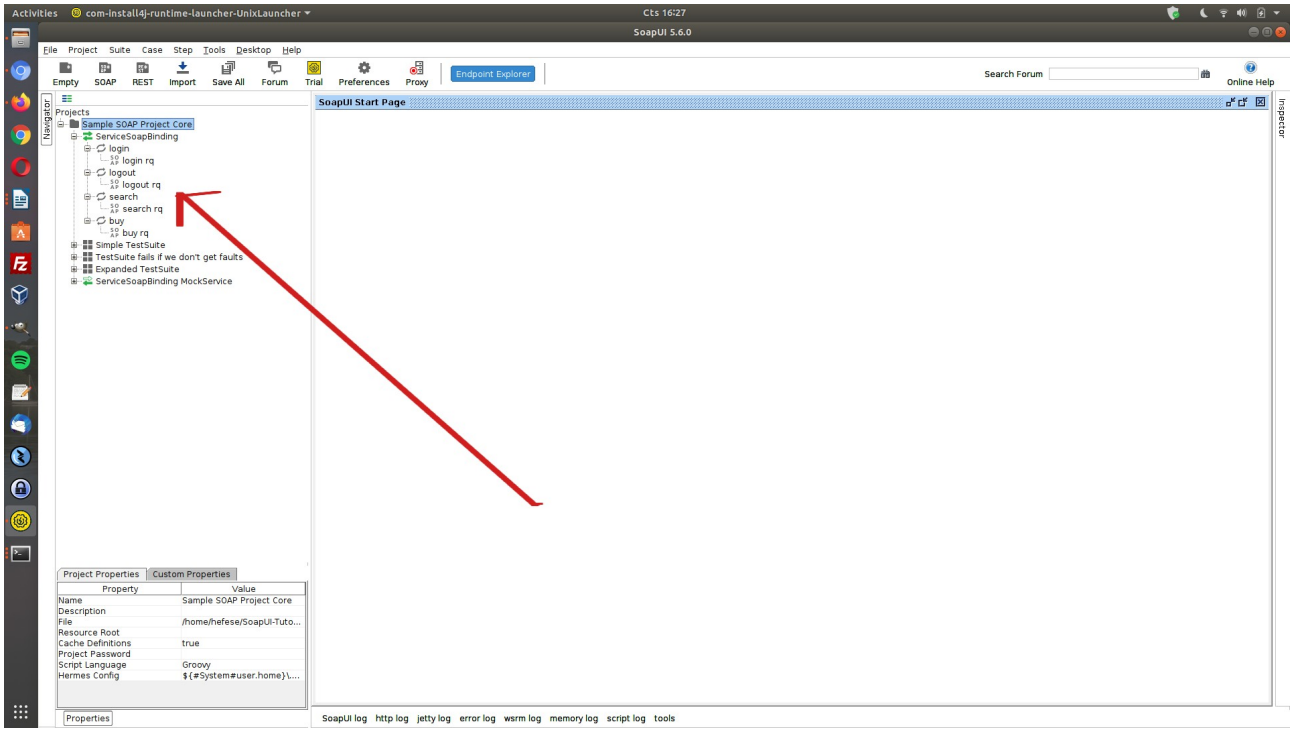
(SoapUI Açılır)

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

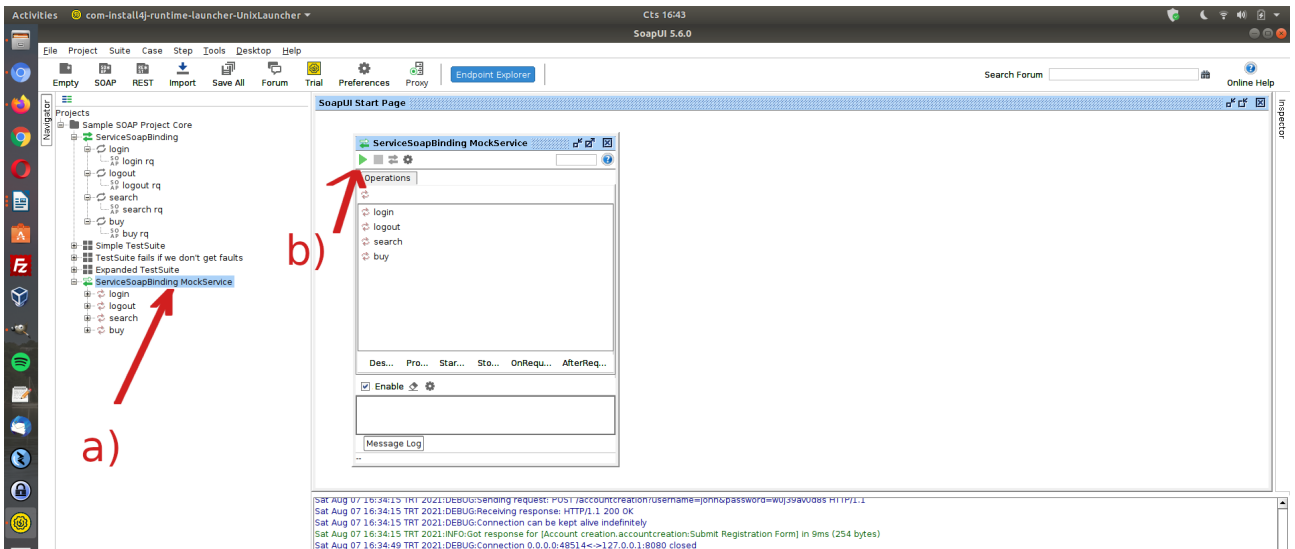
SOAP Örnek Web Servisi SoapUI Proje Dosyası Konumu: /home/hefese/SoapUI-Tutorials/

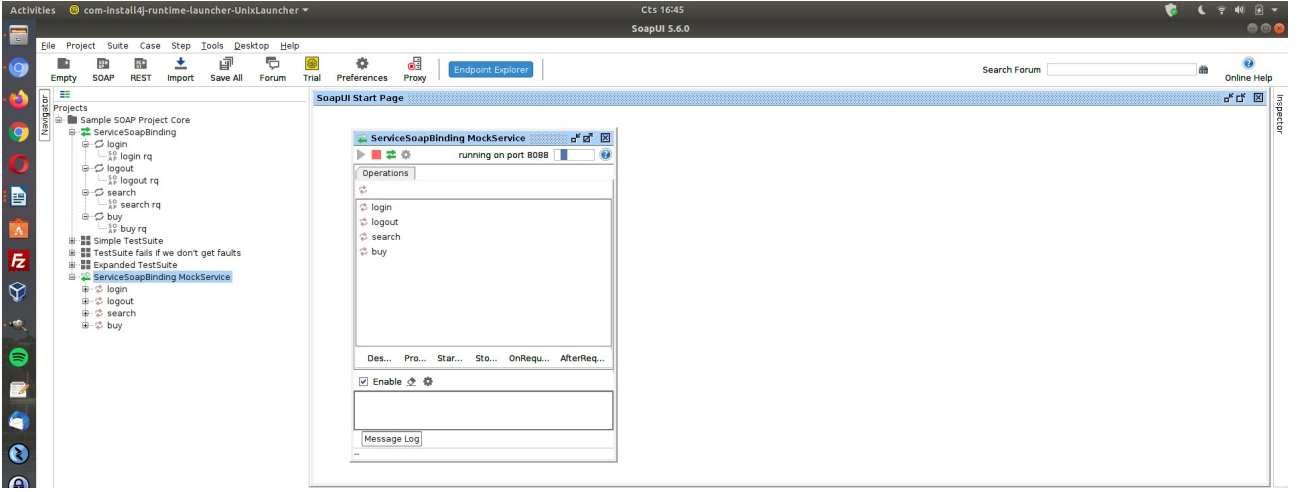


Import ile soapui proje dosyası yüklenir ve örnek soap web servisi arayüzü / kapsamı sol sütunda listelenir.

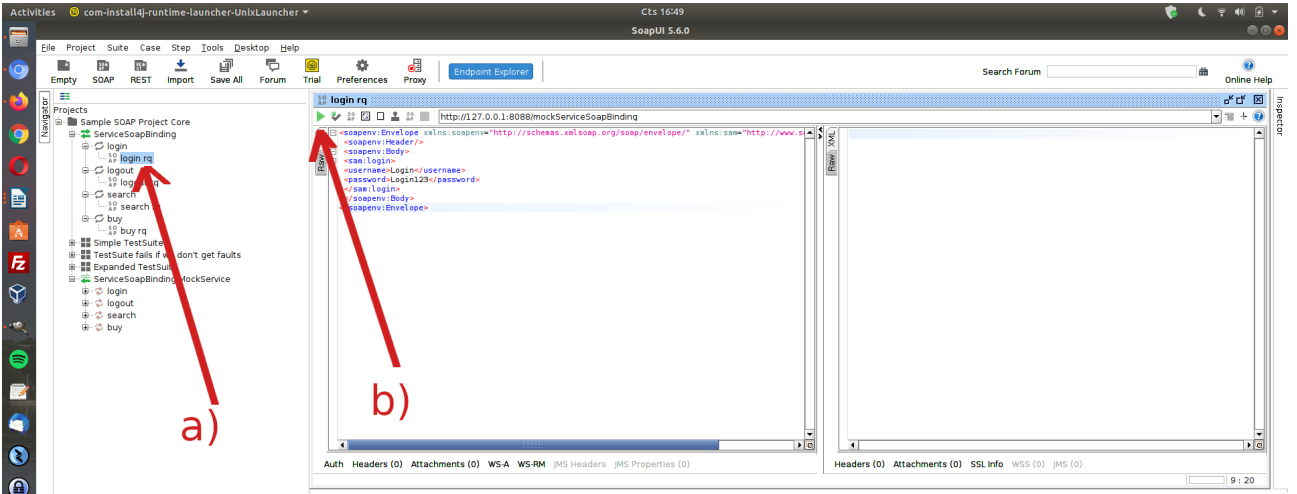


Bu uygulamada hedef soap web servisini canlı hale (çalışır hale) getirmek için localhost'ta bir web sunucu ayağa kaldırılması gerekmektedir. Normal şartlarda buna gerek olmayacaktır. Çünkü normal şartlarda hedef web servisi uzak bir yerden zaten çalışır olacaktır. Bu normal duruma göre ekstra konumundaki 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.

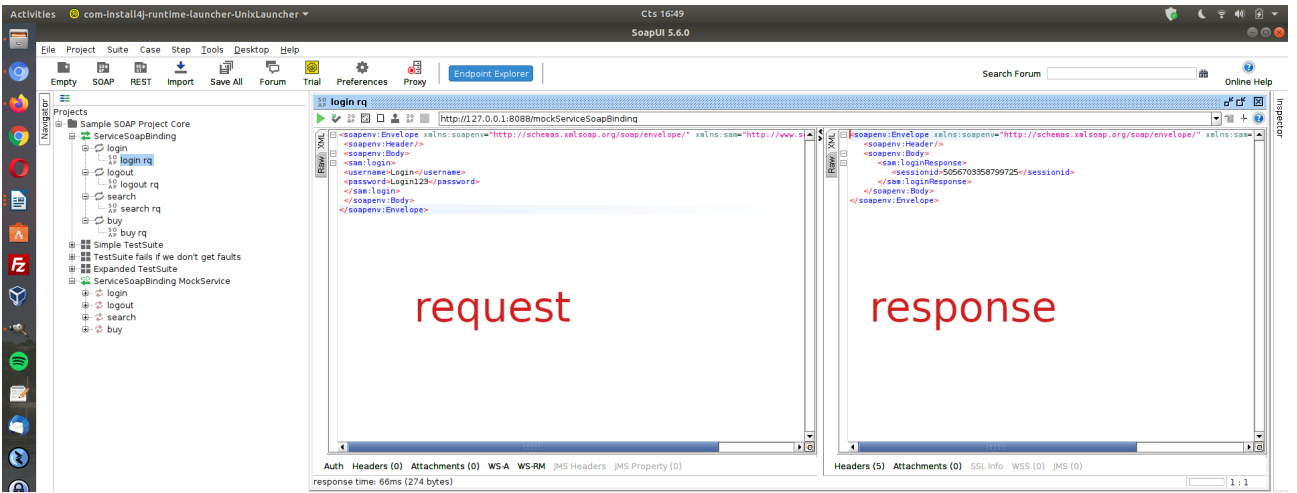




Böylece göndereceğimiz örnek xml talepleri hedefine gidebilecektir ve yanıt alınabilecektir. Şimdi demo soap web servisi teste hazırdır. Sol sütunda sıralı örnek xml taleplerinden login parametrelili olanı gönderelim ve yanıtı gözlemleyelim.



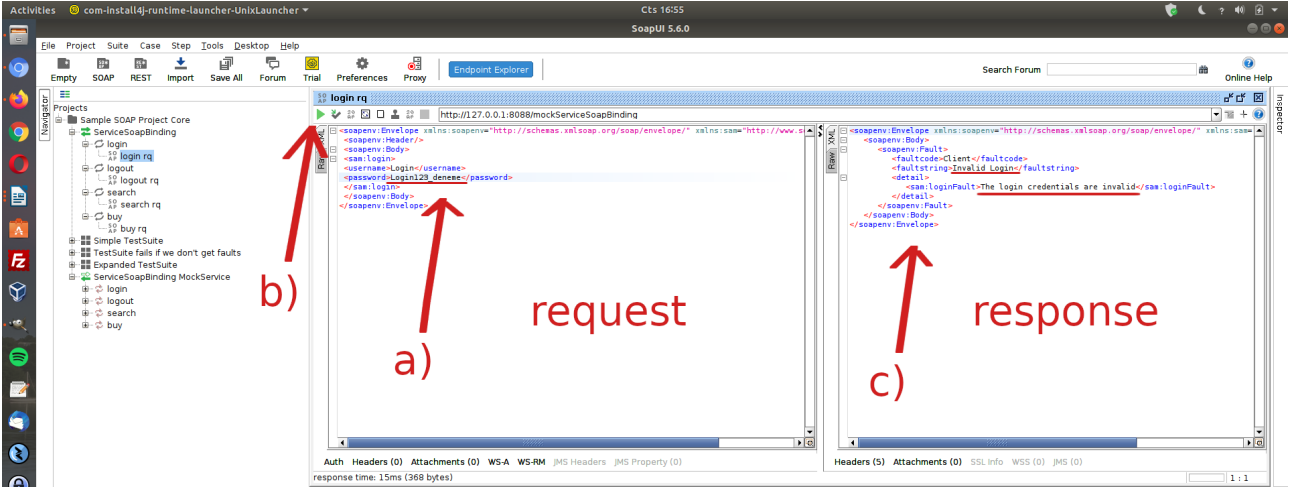
(Örnek Xml Taleplerinden Biri Seçilir)



(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.

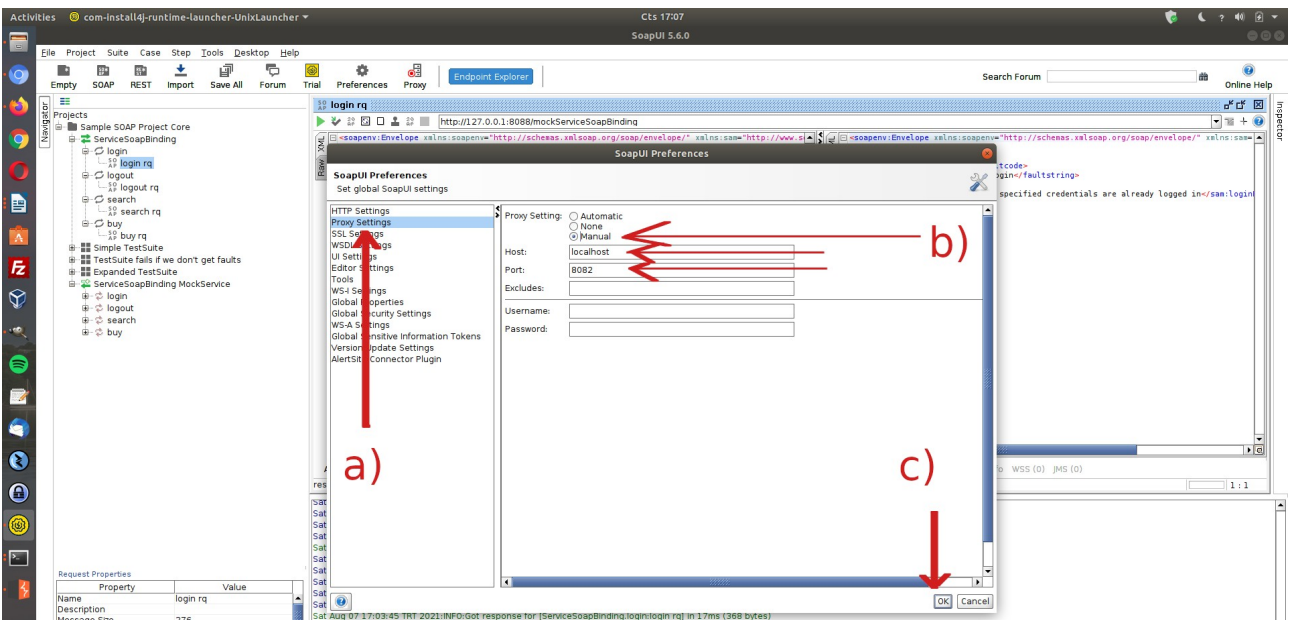
Login mekanizmasını test etmek maksatlı login olma isteği olan talep paketindeki xml node'larından parola bilgisini geçersiz bir parola yapalım ve talep paketini tekrar gönderelim.



Görüldüğü gibi girilen kullanıcı bilgileri geçersizdir yanıtı 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.

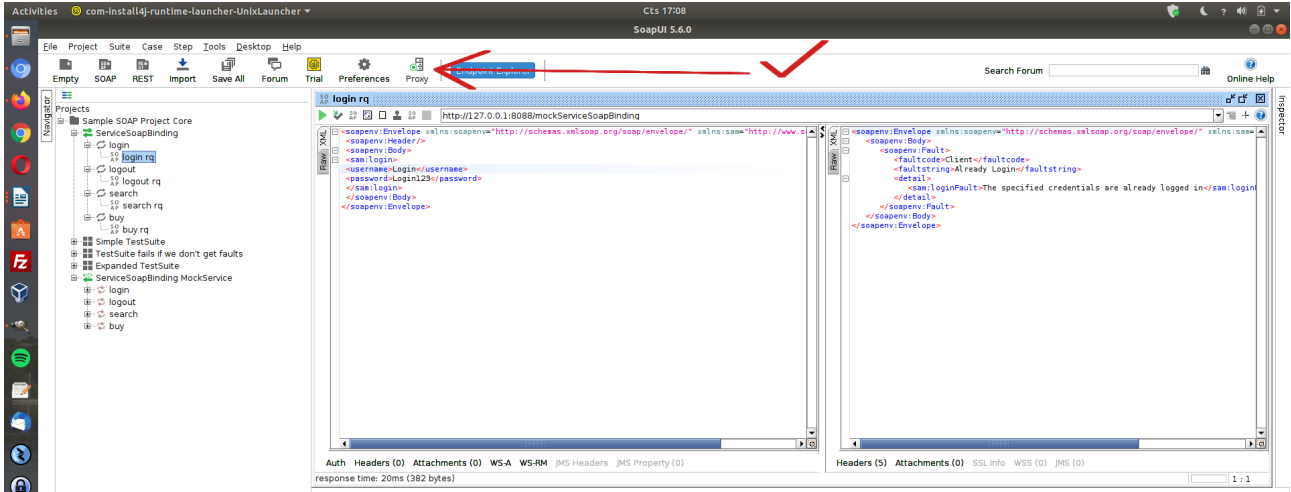
Ayrıca SoapUI'nin sunduğu proxy ayarı kullanılarak paketler Burpsuite proxy'sine yönlendirilebilir ve Burpsuite'te otomatize daha detaylı güvenlik testleri uygulanabilir. Örneğin login isteği olan xml talep paketi için proxy ayarı ile Burpsuite'e alınabilir ve paketin gövdesindeki ilgili xml node'ları Intruder'da işaretlenerek brute force ve dictionary saldırısı düzenlenebilir. Böylece örnek xml talebi login isteği paketi 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.



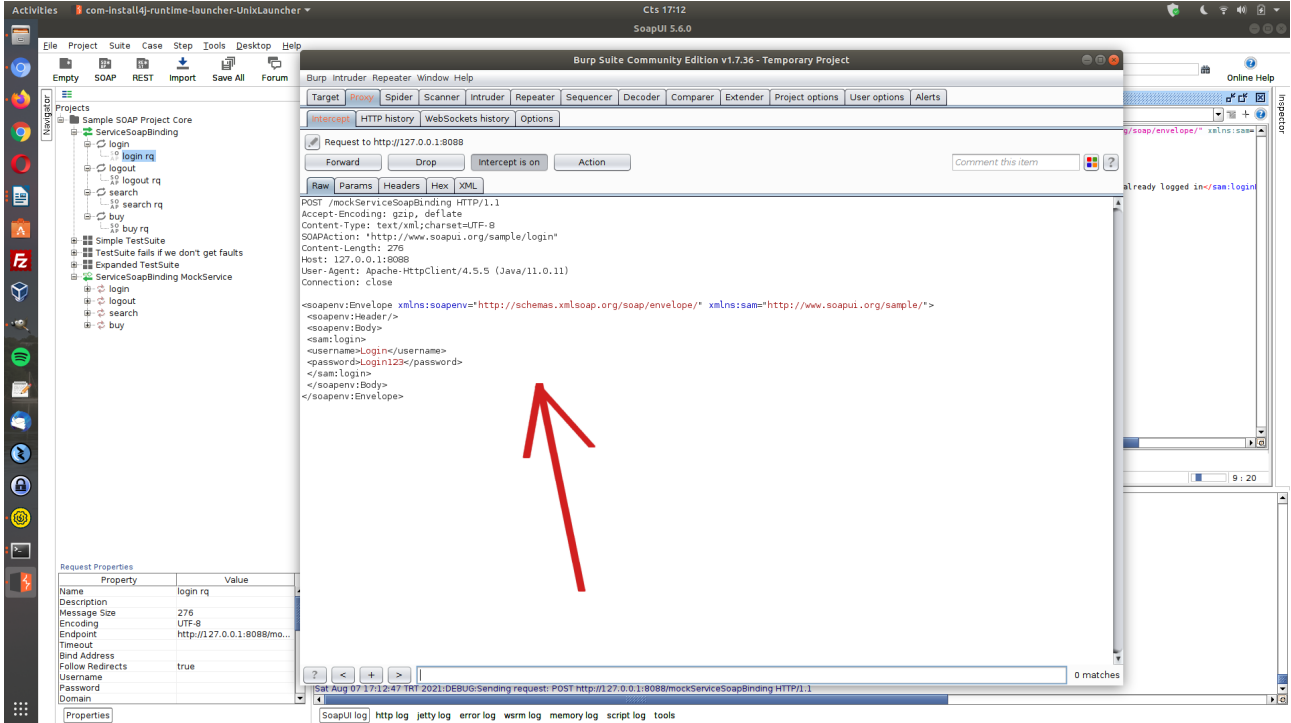
(SoapUI Proxy Ayarı Girilir)

Not: Proxy ayarı 8082 girilir, çünkü 8080’de localhost’ta ayağa kaldırılan tutorials’daki demo soap servisi çalışmaktadır.



(SoapUI ‘de Proxy Açık Durumdadır)

Ardından Burpsuite yazılımı localhost 8082’de çalıştırılır. Böylece SoapUI arayüzünden login isteği yapan xml talep tekrar gönderildiğinde paketi Burpsuite yakalar.

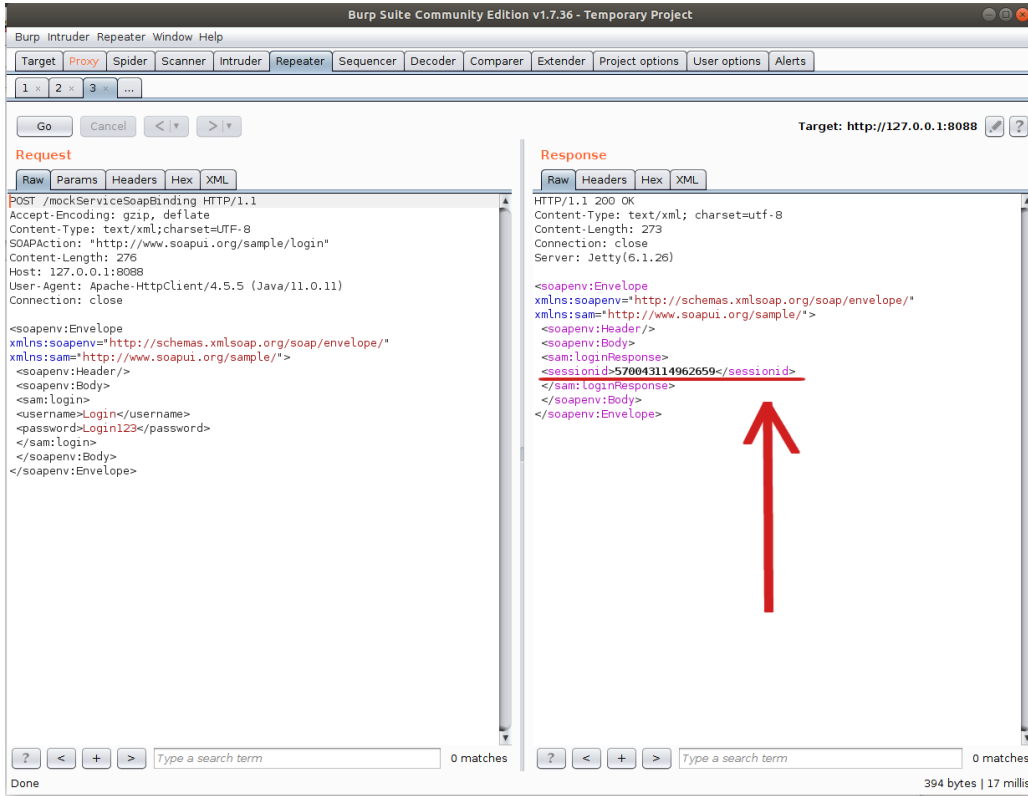


Şimdi login olma isteği paketini geçerli hesap bilgileriyle gönderdiğimizde gelen yanıt paketini Burpsuite repeater’den görelim.

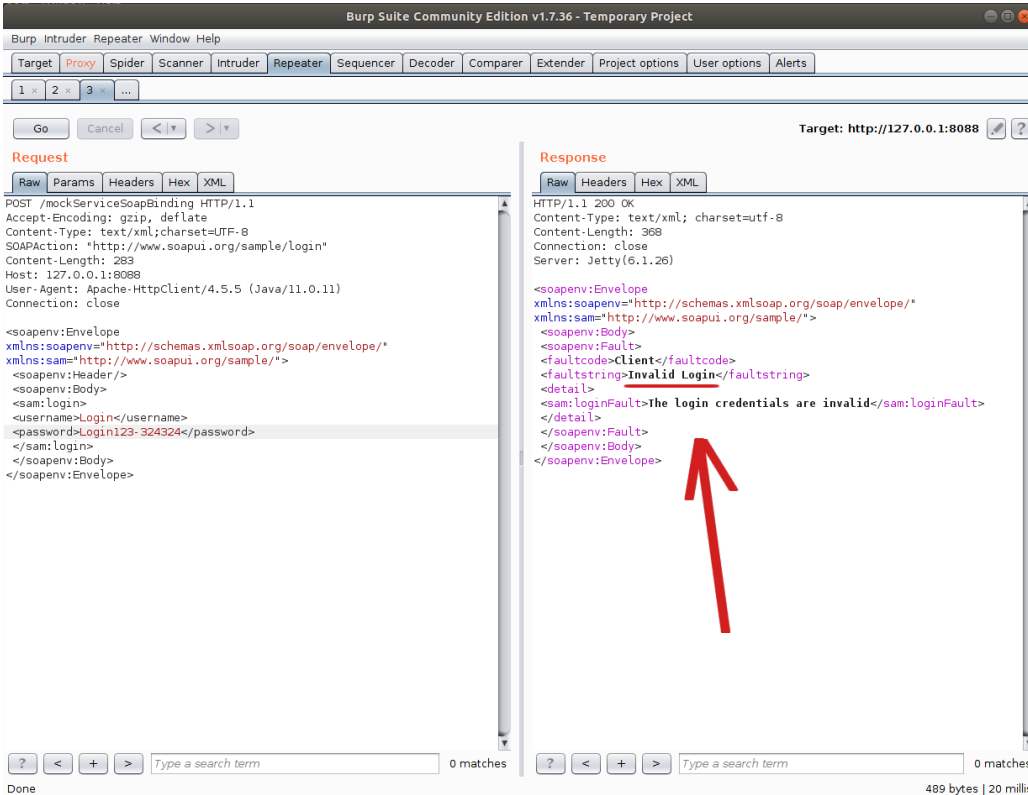
Not:

Geçerli hesap bilgileriyle proxy ayarı girmeden önce talep paketi gönderildiğinden localhost’ta ayağa kaldırılmış web sunucuda oturum açıktır. Bu nedenle xml login isteği talep paketi repeater’den gönderildiğinde ve yanıt paketine bakıldığında sessionID gelmesi yerine zaten oturum açık yanıt mesajı gelecektir. Bu nedenle localhost’ta ayağa kaldırdığımız web sunucuyu SoapUI sol sütundan “ServiceSoapBinding MockService”e çift

tık yapıp restart'layarak login isteği xml talebini tekrar gönderdiğimizde oturumun yeni açılışında gelen yanıt paketini görebiliriz.

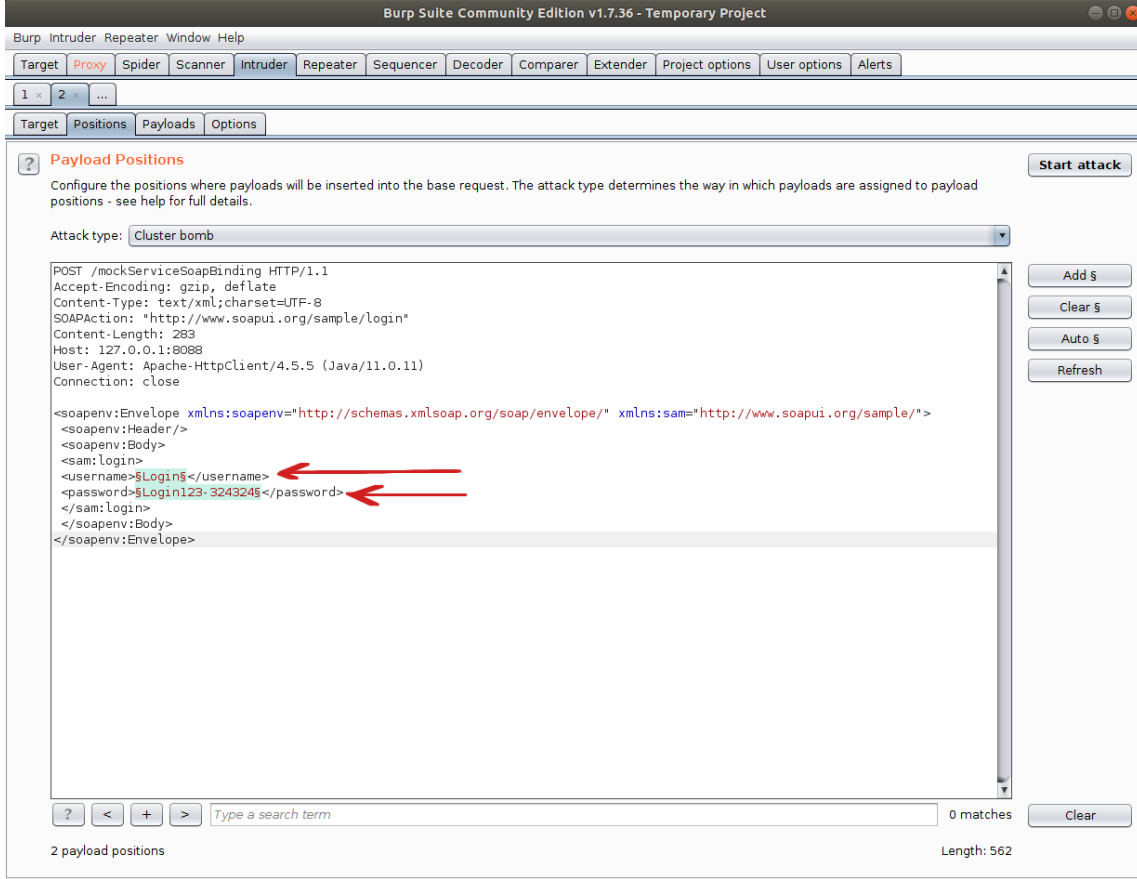


Dönen pakette sessionid xml node'u yer almaktadır. Yani session gelmektedir. Geçersiz bir hesap ile talep paketini gönderdiğimizde ise gelen yanıt paketini görelim.



Dönen pakette geçersiz hesap bilgisi hatası yer almaktadır. Buradaki hata ifadesini brute force / dictionary saldırısında flag olarak alabiliriz ve denenen olası hesaplardan başarısız olanlar bu flag'i tick'lesin, başarılı olan ise bu flag'i 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 oluşturulmuş ve içlerine rastgele kullanıcı adları ve parolalar ile doldurulmuş. 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

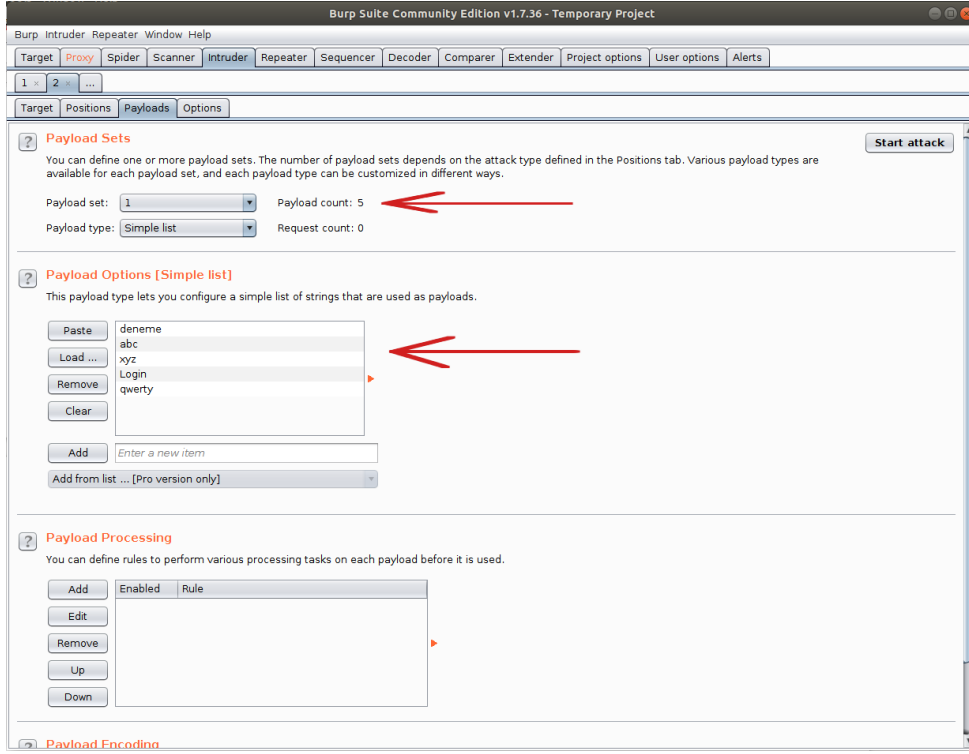
parola12*

Login123

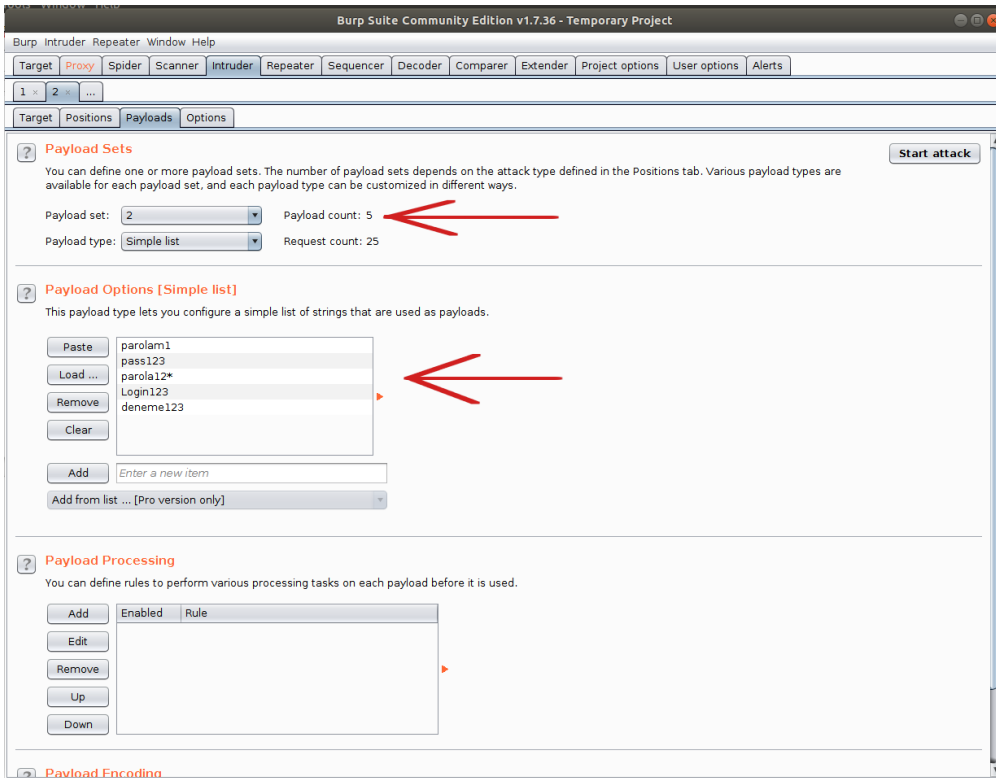
deneme123

// Doğru Olan

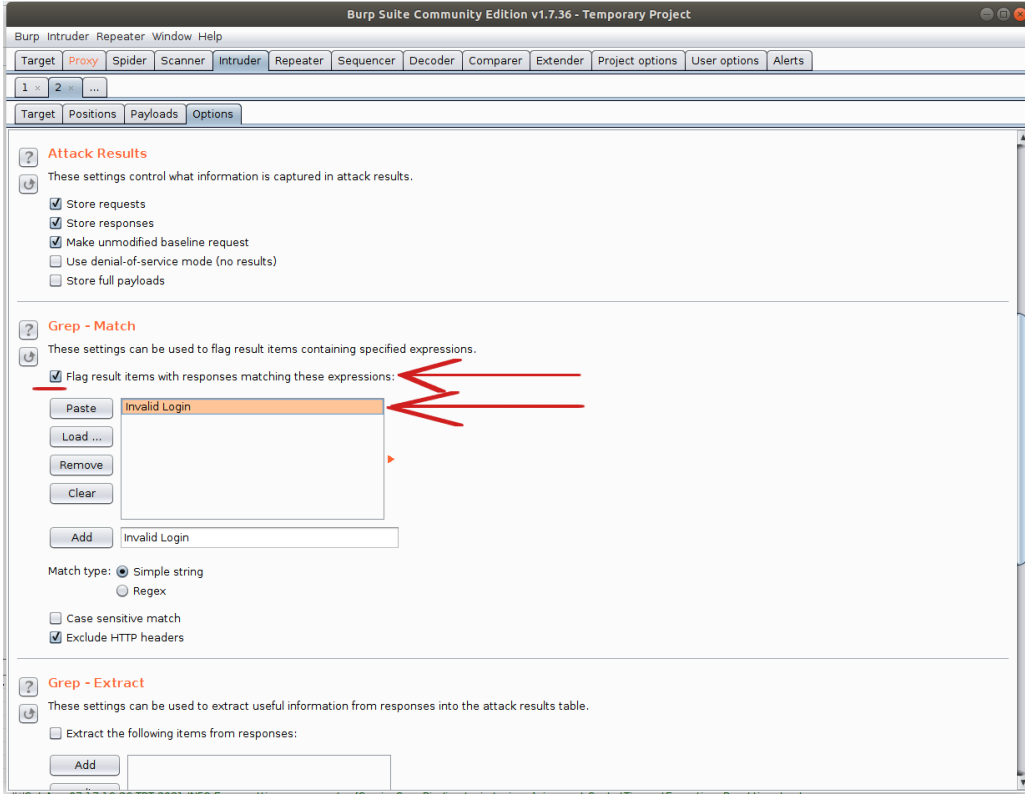
İşaretili alanlara payload'ları koyalım ve ardından yanlış hesap bilgisi denendiğinde gelen hatayı flag yapalım.



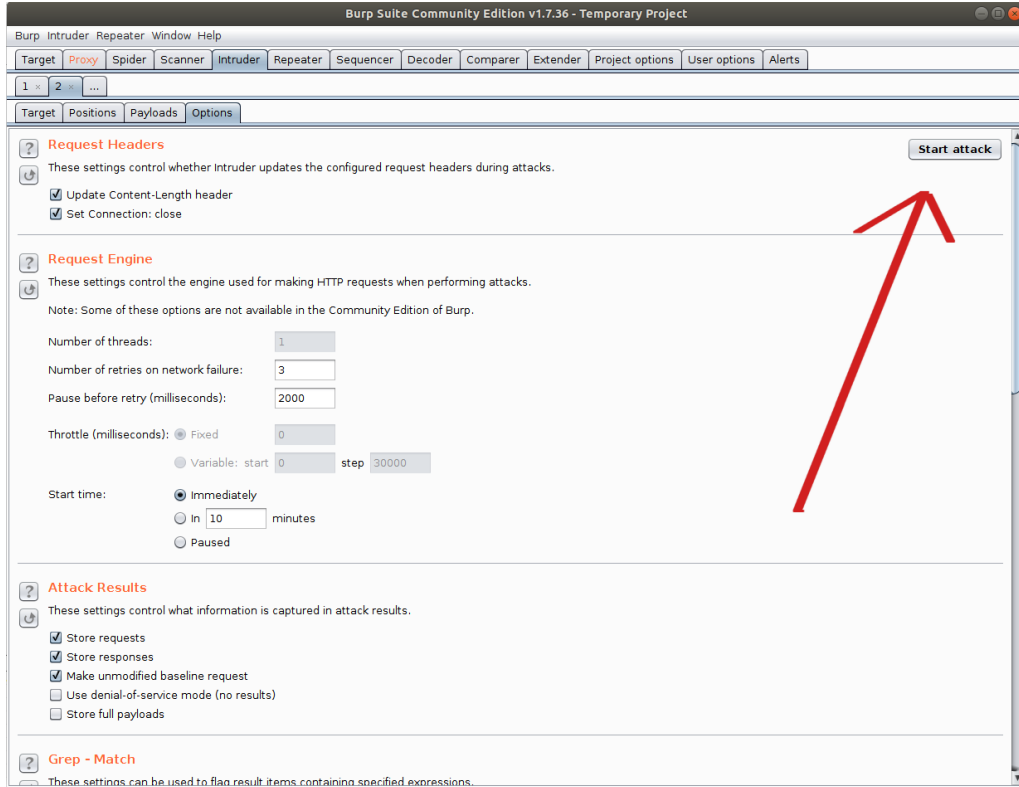
(Olası Kullanıcı Adı Listesi Birinci Payload Olarak Belirlenir)



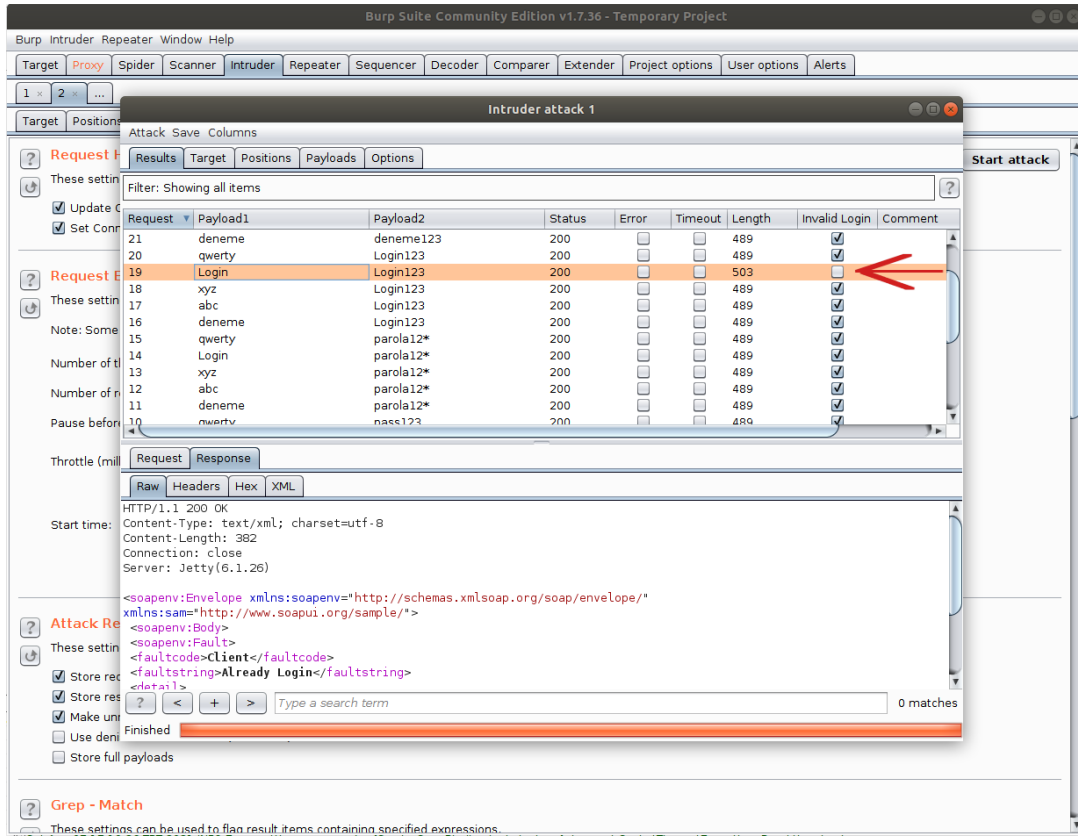
(Olası Parolalar Listesi İkinci Payload Olarak Belirlenir)



(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 kurcalanarak web servislerine saldırı testleri uygulanabilir.

Uygulama [SoapUI Yazılımı ile SOAP Web Servis Test Etme 2]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

Bu uygulamada SoapUI yazılımı kullanılarak kasıtlı zafiyetler içeren DVWS web servisine ait bir ders sayfasında sunulan WSDL dosyası alınacaktır ve DVWS'nin ilgili sayfasında işlevsel soap web servisini test etme gösterilecektir.

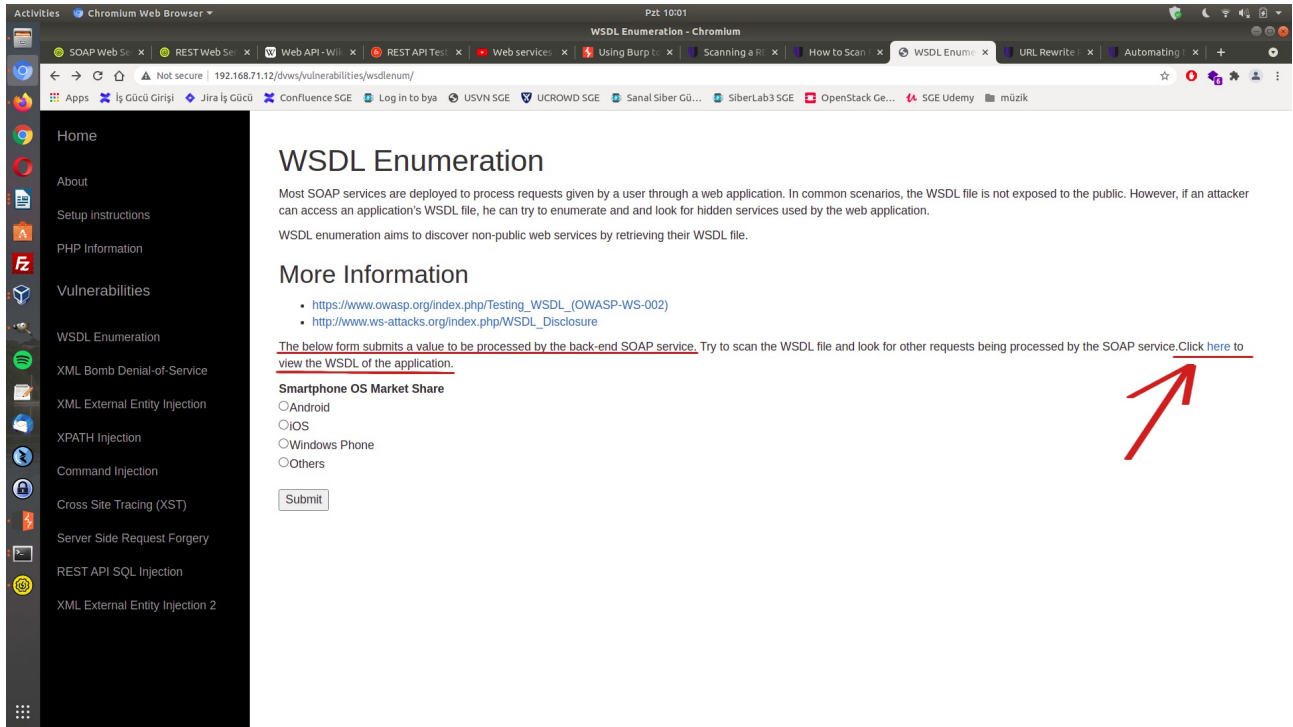
Gereksinimler

Ubuntu 18.04 LTS	// Ana Makine
SoapUI	// Web Servis Test Yazılımı
Burpsuite	// Proxy Yazılımı
DVWS - Windows 7 Home Premium VM	// Hedef Web Servisi - Sanal Makine

Not: SoapUI kurulumu ve başlatma Uygulama [SoapUI Yazılımı ile SOAP Web Servis Test Etme] başlığında bahsedilmiştir.

Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

Örneğin DVWS web servisinin bir ders sayfasında wsdl enumeration saldırısı senaryolaştırılmıştır ve sayfanın dediği üzere dvws'nin arka ucundaki soap web servisin WSDL dosyasının elde edildiği varsayılmaktadır. Bu wsdl dosyasının url'si ders sayfasında verilmiştir:



WSDL Enumeration

Most SOAP services are deployed to process requests given by a user through a web application. In common scenarios, the WSDL file is not exposed to the public. However, if an attacker can access an application's WSDL file, he can try to enumerate and look for hidden services used by the web application.

WSDL enumeration aims to discover non-public web services by retrieving their WSDL file.

More Information

- [https://www.owasp.org/index.php/Testing_WSDL_\(OWASP-WS-002\)](https://www.owasp.org/index.php/Testing_WSDL_(OWASP-WS-002))
- http://www.ws-attacks.org/index.php/WSDL_Disclosure

The below form submits a value to be processed by the back-end SOAP service. Try to scan the WSDL file and look for other requests being processed by the SOAP service. [Click here to view the WSDL of the application.](#)

Smartphone OS Market Share

Android

iOS

Windows Phone

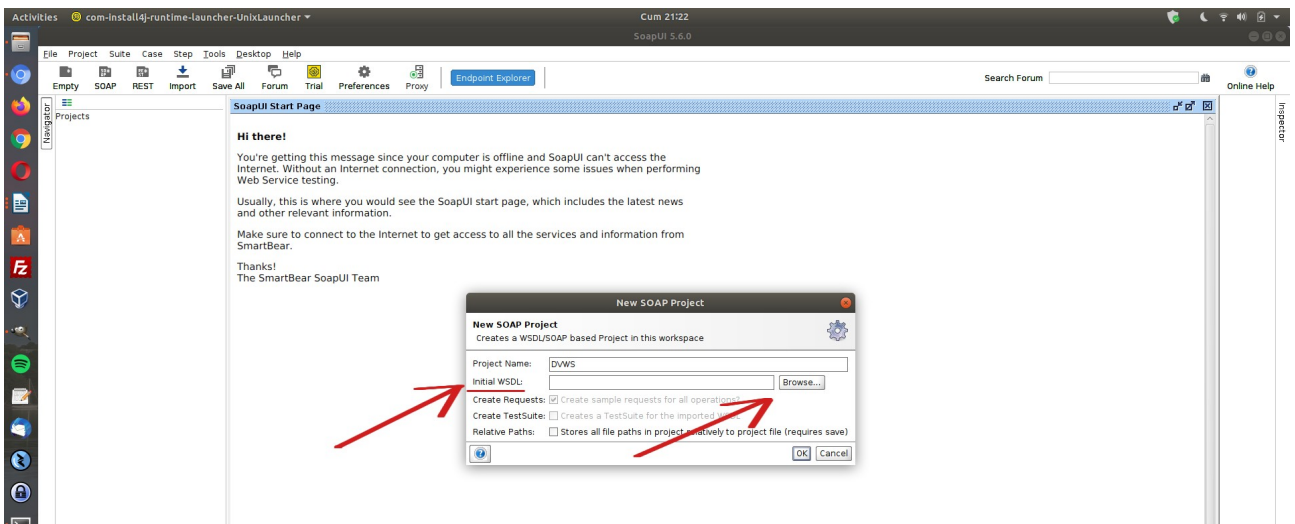
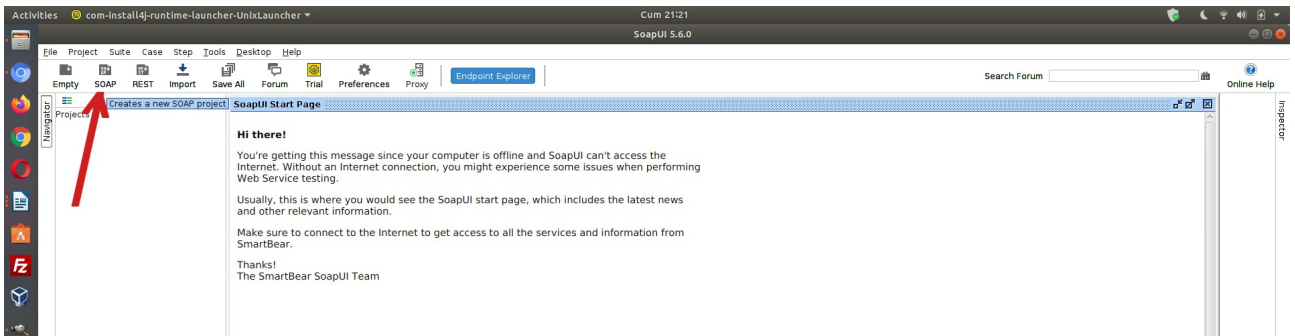
Others

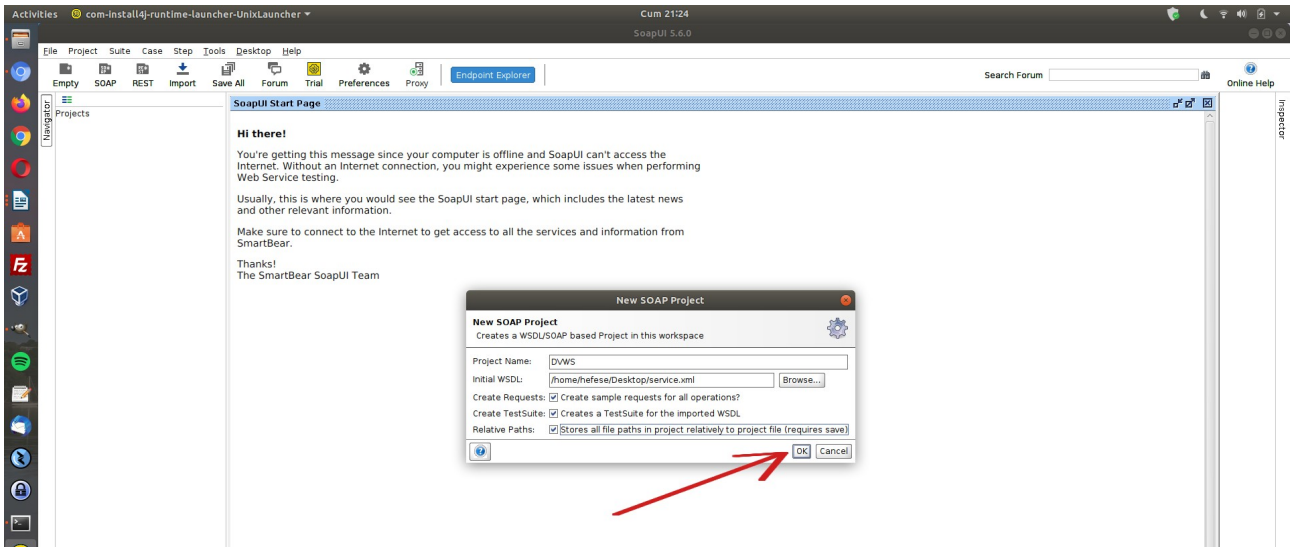
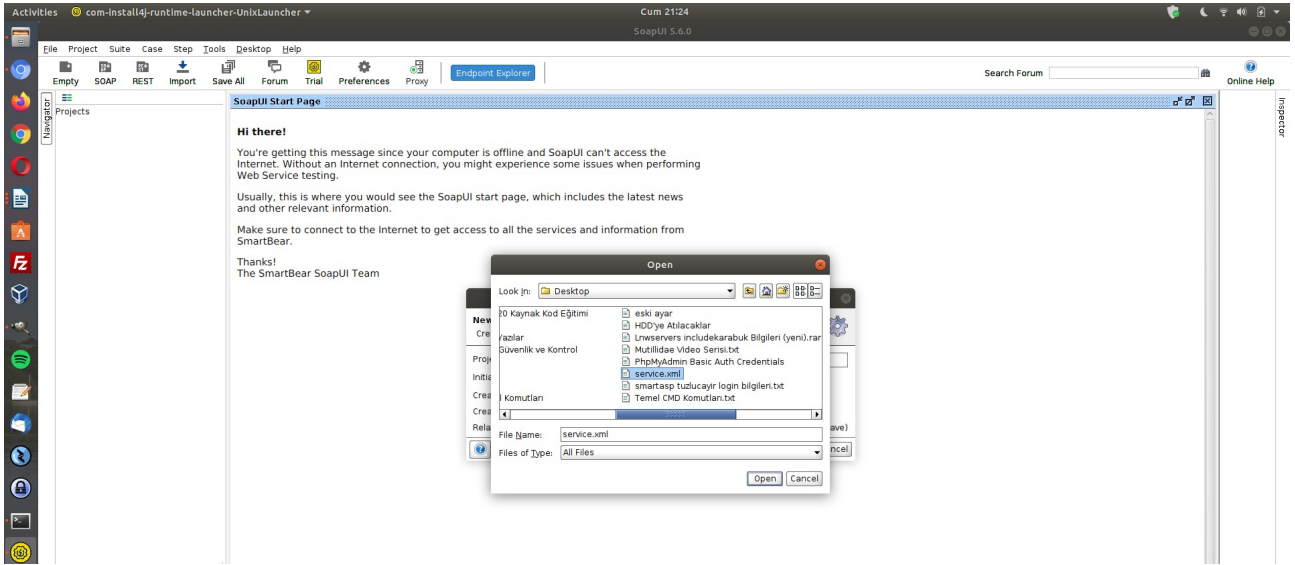
localhost/dvws/vulnerabilities/wsdlenum/service.php?wsdl - Chromium

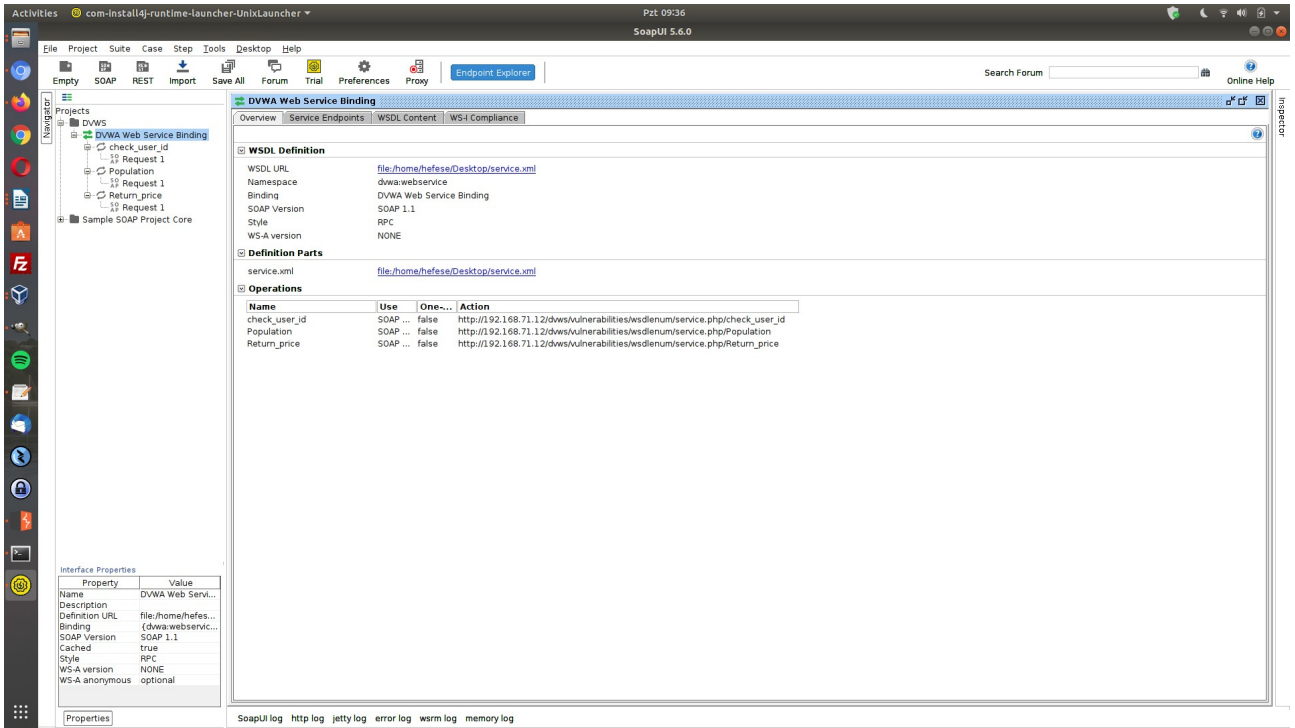
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="dwa:webservice" xmlns:soap="http://schemas.xmlsoap.org/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="dwa:webservice">
  <types>
    <xsd:schema targetNamespace="dwa:webservice">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
    </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="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>
```

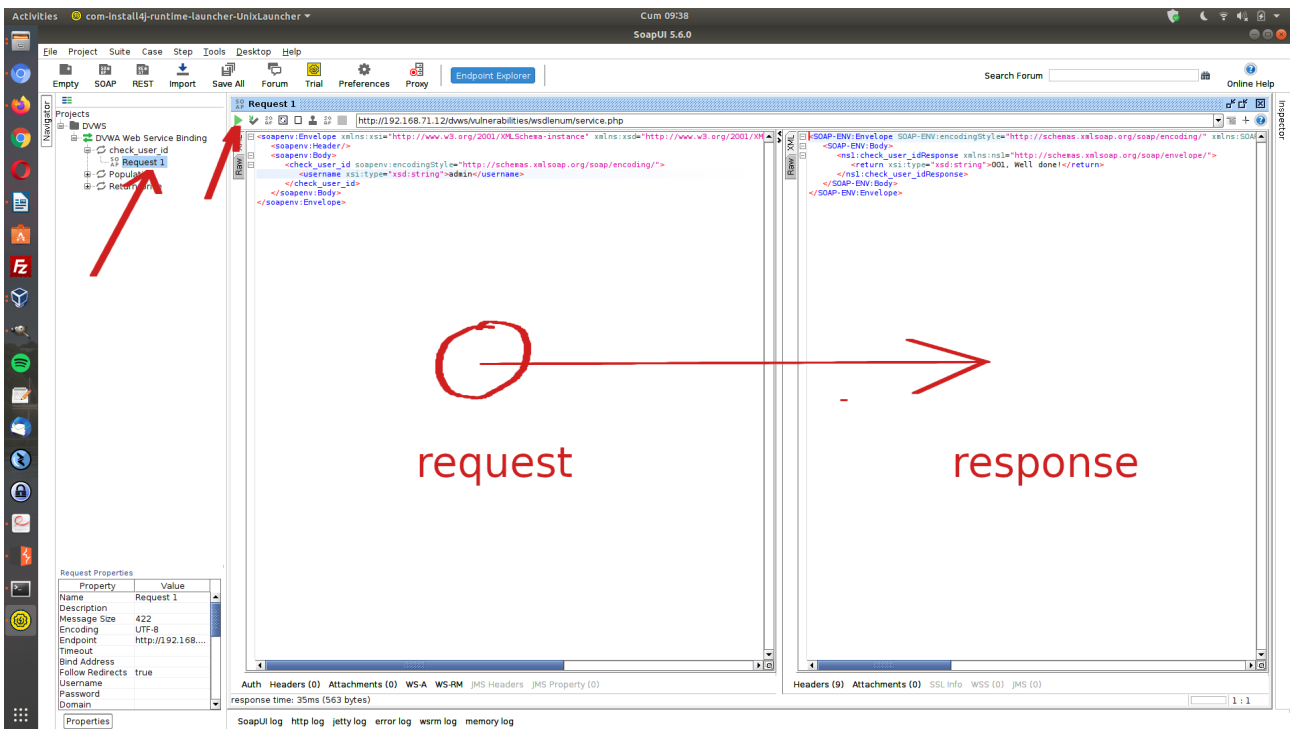
SoapUI yazılımına bu soap web servisin wsdl dosyasını verelim ve soap web servisin arayüzünü / kapsamını görelim.



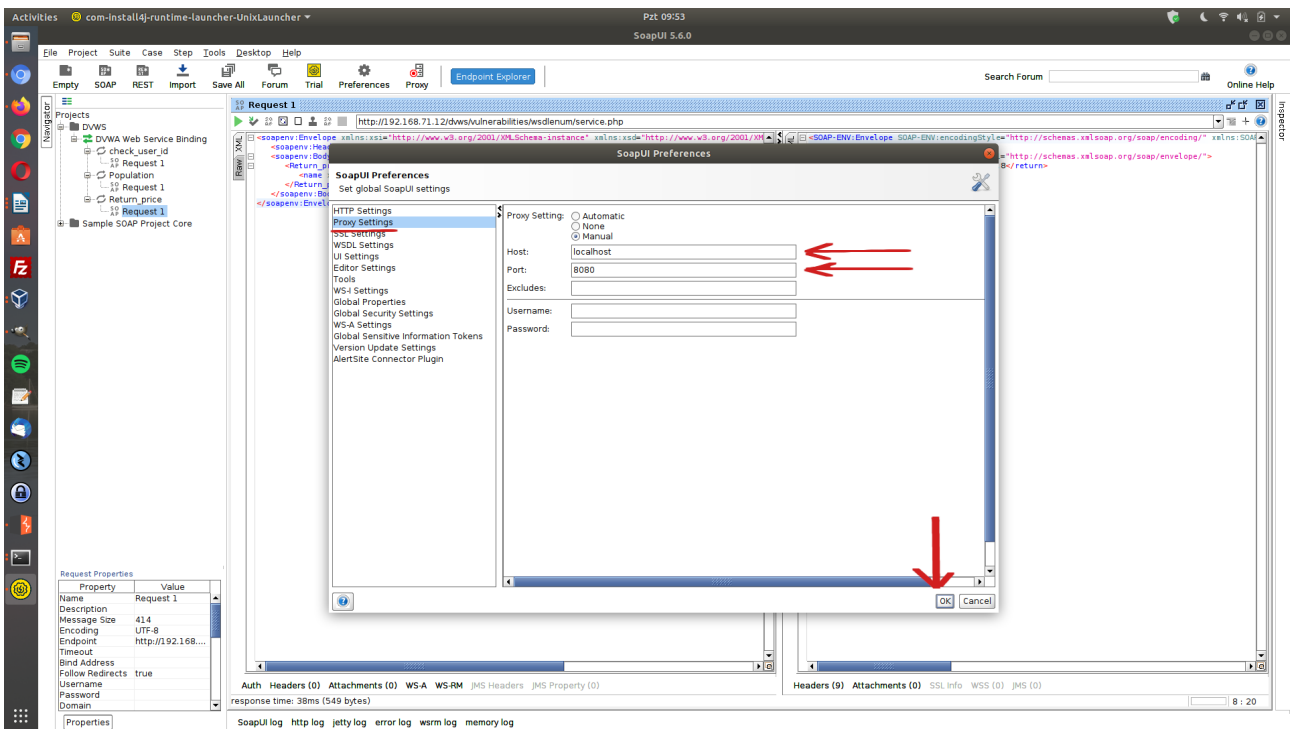
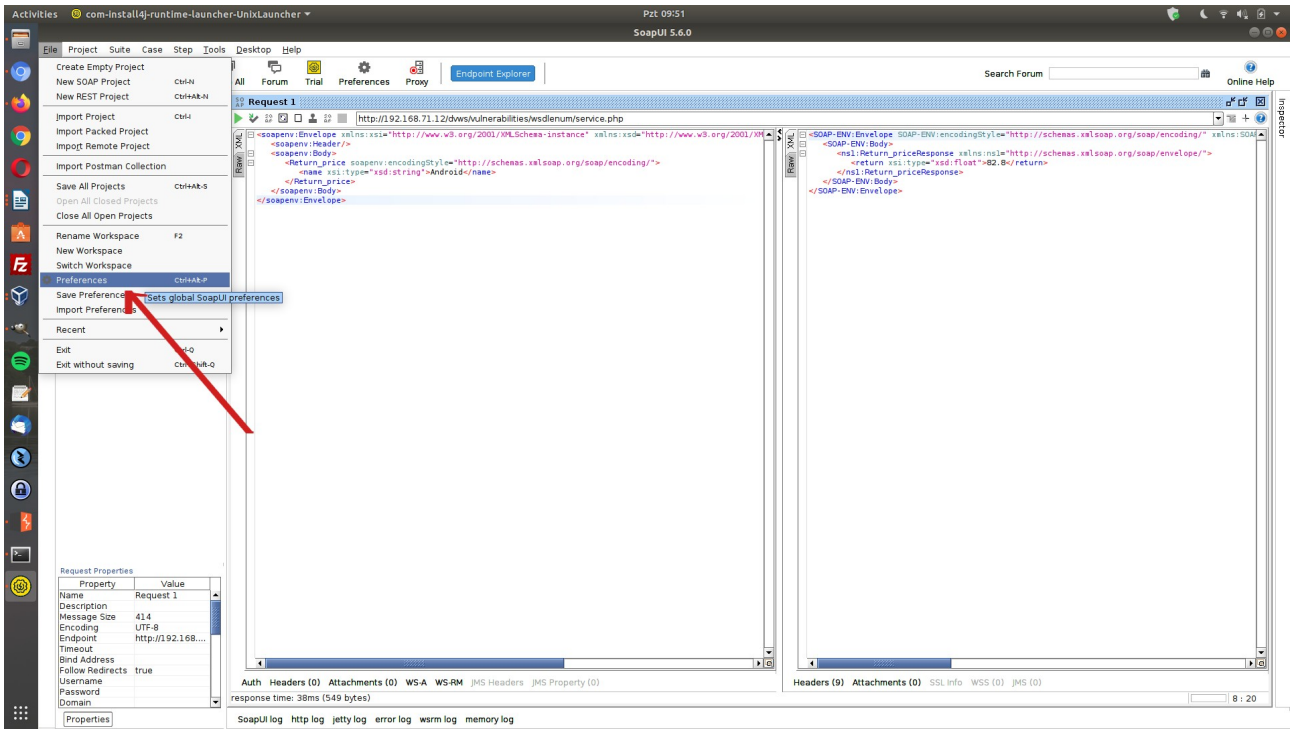




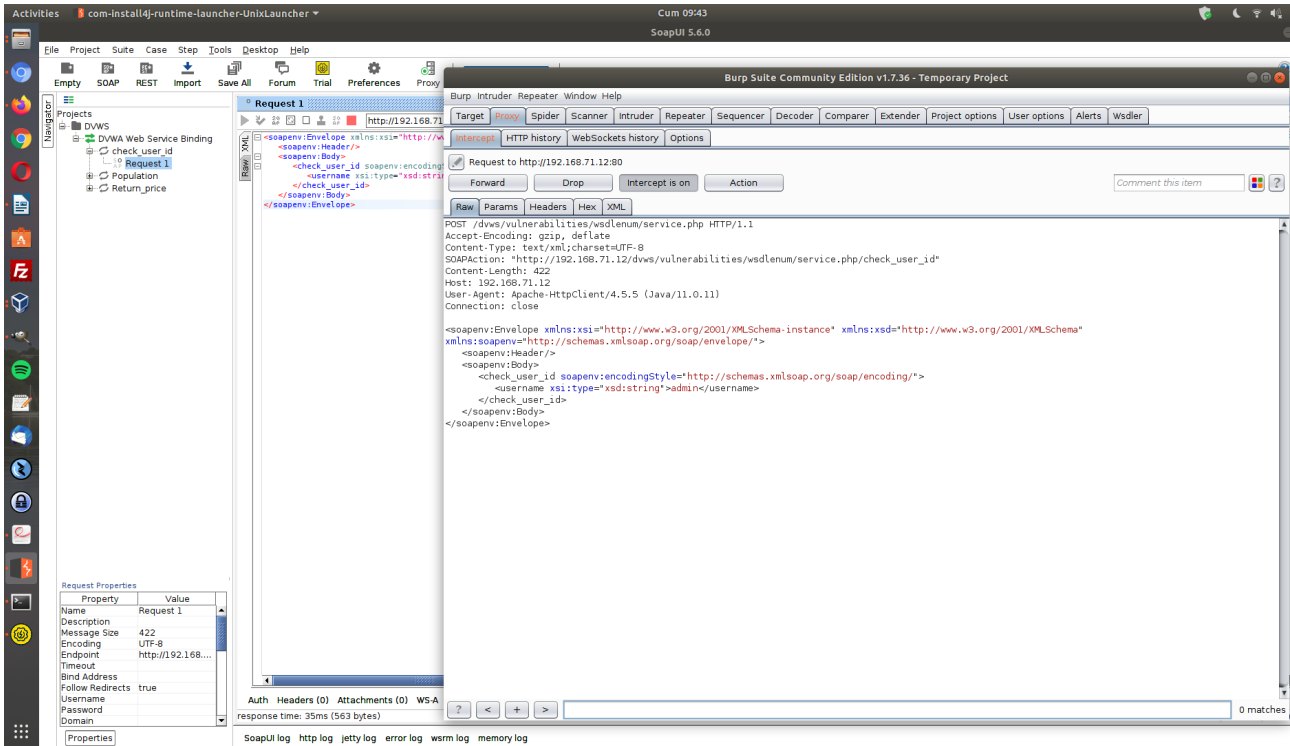
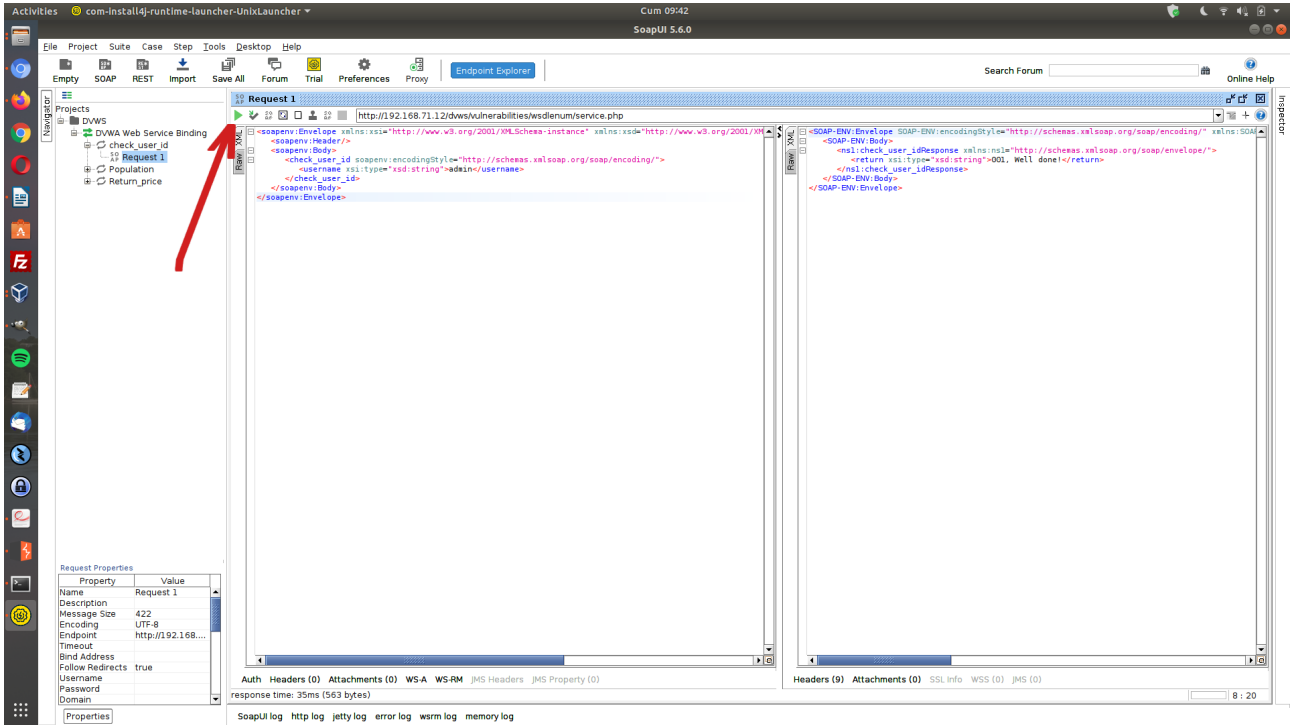
Bu adımlar neticesinde dvws'deki soap web servisin arayüzü / kapsamı yüklemesi ile örnek xml talepleri sol sütunda sıralanır: check_user_id, Population, Return_price. Sol sütundaki request 'lere tıklanıldığında gönderilmeye hazır xml talepleri açılacaktır.



Burada SoapUI ayarlardan proxy ayarı girerek gönderilen paketi yakalayabilir ve Burpsuite'te Intruder'a göndererek çeşitli güvenlik testleri uygulanabilir. SoapUI'de proxy ayarı girmek için;



adımları takip edilir ve sonra ekrandaki xml talep ve yanıt penceresindeki yeşil butonla xml talebi gönderilir. Bu yapıldığında talebi localhost 8080 portunu dinleyen burpsuite alacaktır.



Yakalanan paketi repeater'a gönderelim ve yanıt paketlerini gözlemleyerek güvenlik testini uygulayalım.

```
Request
POST /dwvs/vulnerabilities/wsdlenum/service.php HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction:
"http://192.168.71.12/dwvs/vulnerabilities/wsdlenum/service.php/check_u
ser_id"
Content-Length: 422
Host: 192.168.71.12
User-Agent: Apache-HttpClient/4.5.5 (Java/11.0.11)
Connection: close

<soapenv:Envelope
xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns: xsd="http://www.w3.org/2001/XMLSchema"
xmlns: soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
<check_user_id
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<username xsi:type="xsd:string">admin</username>
</check_user_id>
</soapenv:Body>
</soapenv:Envelope>

Response
HTTP/1.1 200 OK
Date: Fri, 20 Aug 2021 06:45:46 GMT
Server: Apache/2.4.23 (win32) OpenSSL/1.0.2h PHP/5.5.38
X-Powered-By: PHP/5.5.38
X-SOAP-Server: NuSOAP/0.9.5 (1.123)
Content-Length: 563
Connection: close
Content-Type: text/xml; charset=ISO-8859-1

<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns: SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:B
ody><ns1:check_user_idResponse
xmlns: ns1="http://schemas.xmlsoap.org/soap/envelope/"><return
xsi:type="xsd:string">001. Well
done!</return></ns1:check_user_idResponse></SOAP-ENV:Body></SOAP-ENV:E
nvelope>
```

```
Request
POST /dwvs/vulnerabilities/wsdlenum/service.php HTTP/1.1
Accept-Encoding: gzip, deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction:
"http://192.168.71.12/dwvs/vulnerabilities/wsdlenum/service.php/check_u
ser_id"
Content-Length: 421
Host: 192.168.71.12
User-Agent: Apache-HttpClient/4.5.5 (Java/11.0.11)
Connection: close

<soapenv:Envelope
xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns: xsd="http://www.w3.org/2001/XMLSchema"
xmlns: soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
<check_user_id
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<username xsi:type="xsd:string">root</username>
</check_user_id>
</soapenv:Body>
</soapenv:Envelope>

Response
HTTP/1.1 200 OK
Date: Fri, 20 Aug 2021 06:45:56 GMT
Server: Apache/2.4.23 (win32) OpenSSL/1.0.2h PHP/5.5.38
X-Powered-By: PHP/5.5.38
X-SOAP-Server: NuSOAP/0.9.5 (1.123)
Content-Length: 563
Connection: close
Content-Type: text/xml; charset=ISO-8859-1

<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns: SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:B
ody><ns1:check_user_idResponse
xmlns: ns1="http://schemas.xmlsoap.org/soap/envelope/"><return
xsi:type="xsd:string">002. Well
done!</return></ns1:check_user_idResponse></SOAP-ENV:Body></SOAP-ENV:E
nvelope>
```

(Not: Check_User_ID arguman değeri kasıtlı zafiyetler içeren dwvs web servisinin pdf dökümanından elde edilmiştir. Bkz. /var/www/dwvs/Attacking Damn Vulnerable Web Services.pdf)

Bu şekilde burp'te talep paketinin gövdesindeki node'ların değeri ile, paketin başlıkları ile, ve url'de yer alan parametreler ile oynanarak çeşitli güvenlik testi adımları uygulanabilir.

Uygulama [SoapUI ile Rest Web Servis Test Etme]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

Bu uygulamada SoapUI yazılımı kullanılarak kasıtlı zafiyetler içeren DVWS web servisindeki bir ders sayfasında sunulan rest web servisini test etme yolu gösterilecektir.

Gereksinimler

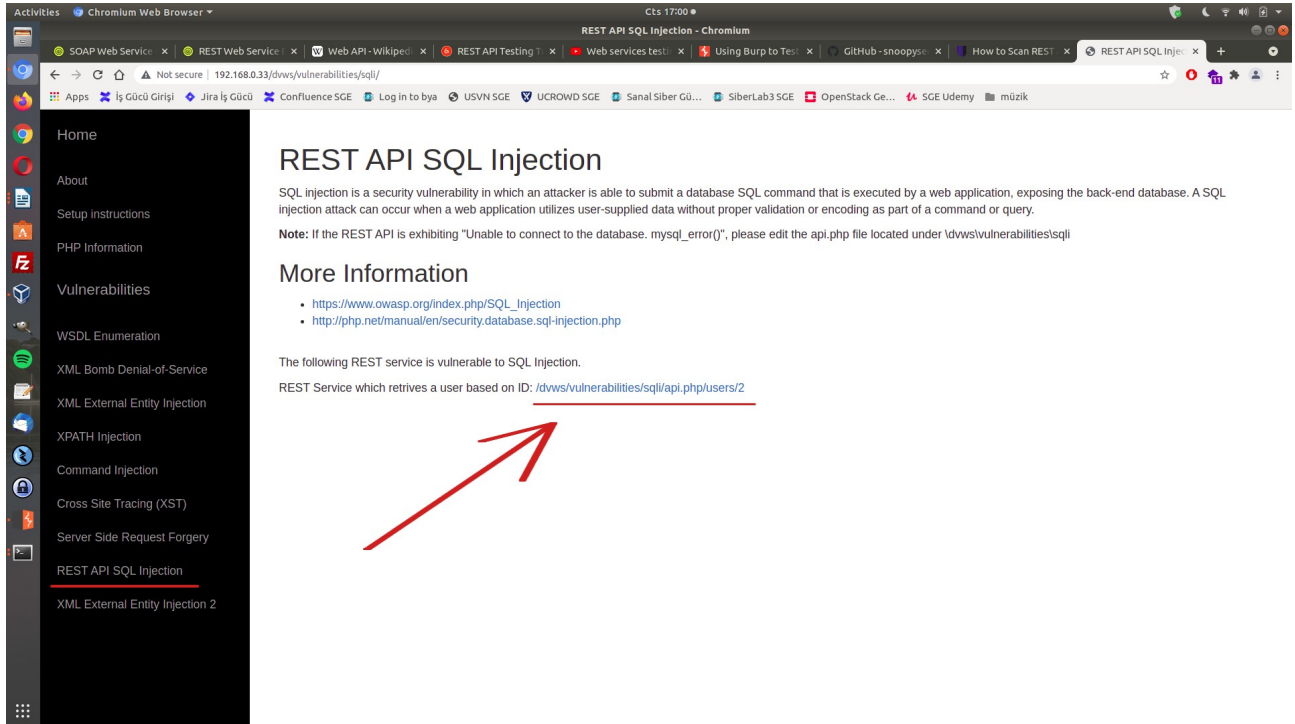
Ubuntu 18.04 LTS	// Ana Makine
SoapUI	// Web Servis Test Yazılımı
Burpsuite	// Proxy Yazılımı
DVWS - Windows 7 Home Premium	// Hedef Web Servisi - Sanal Makine

Not: SoapUI kurulumu ve başlatma Uygulama [SoapUI Yazılımı ile SOAP Web Servis Test Etme] başlığında bahsedilmiştir.

Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışırdır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

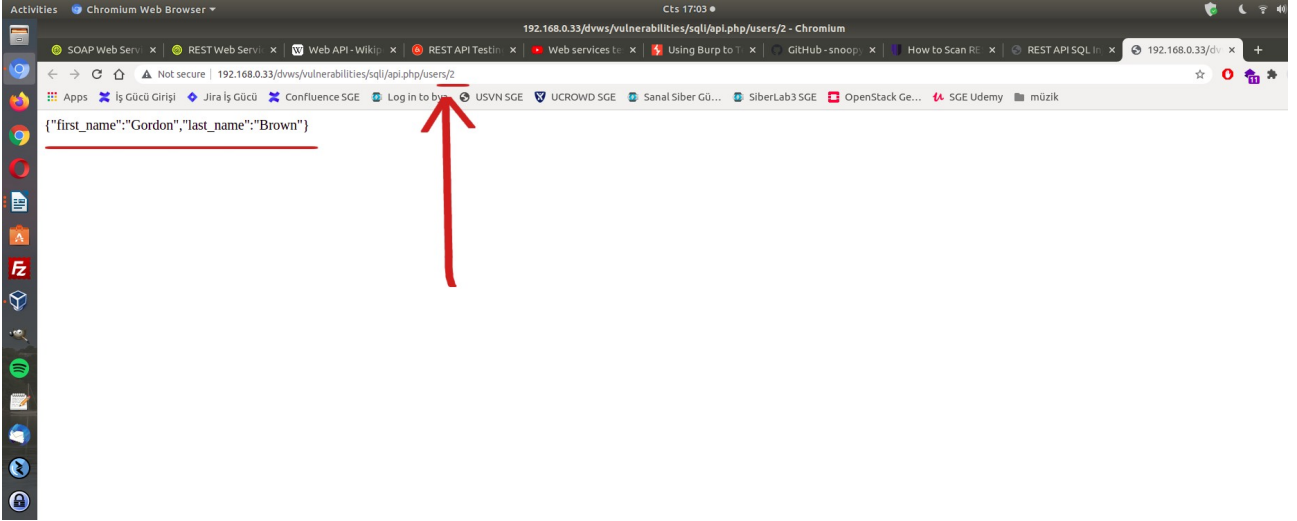
DVWS web servisi kompakt halde test amaçlı hem SOAP web servis hem de REST web servis barındırmaktadır. REST web servisi için bir tanımlama dosyası bulundurmamaktadır. Bunun yerine bir adet url şeklinde arayüz / kapsam sunmaktadır. Bu nedenle soapui'ye arayüz / kapsam bu bir url ile yüklenecektir.

Öncelikle dvws web servisindeki ilgili sayfaya göz atalım.

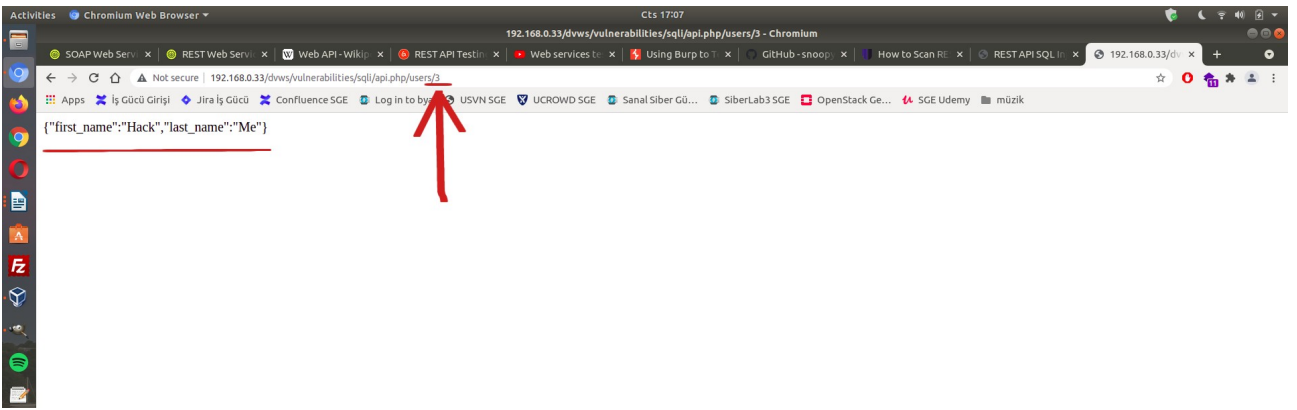


(DVWS Web Servisindeki REST Web Servisi Kısmı)

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



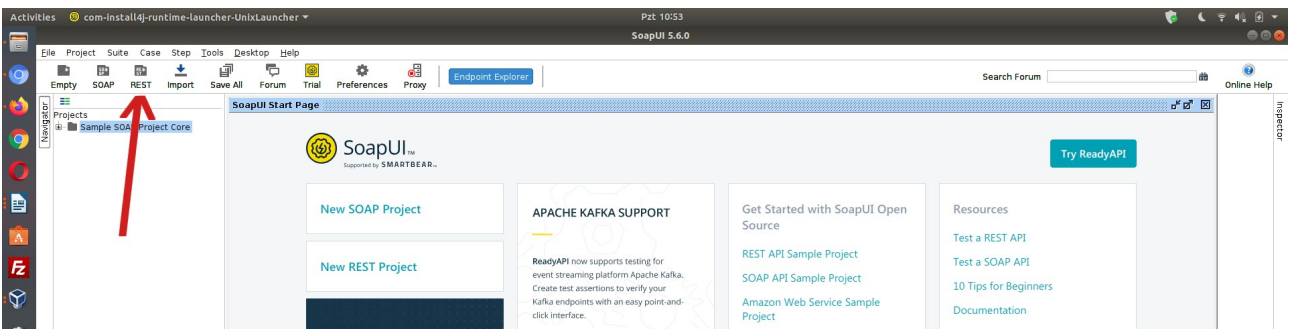
(REST Web Servis URL Parametresi 2 iken Gelen Veri)

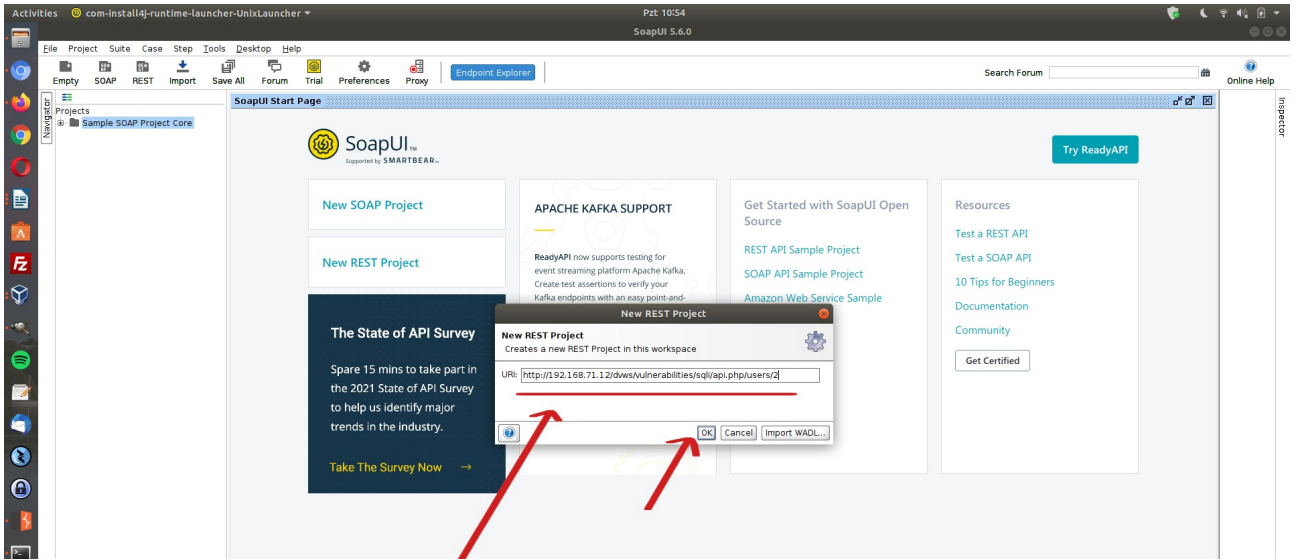


(Rest Web Servis URL Parametresi 3 iken Gelen Veri)

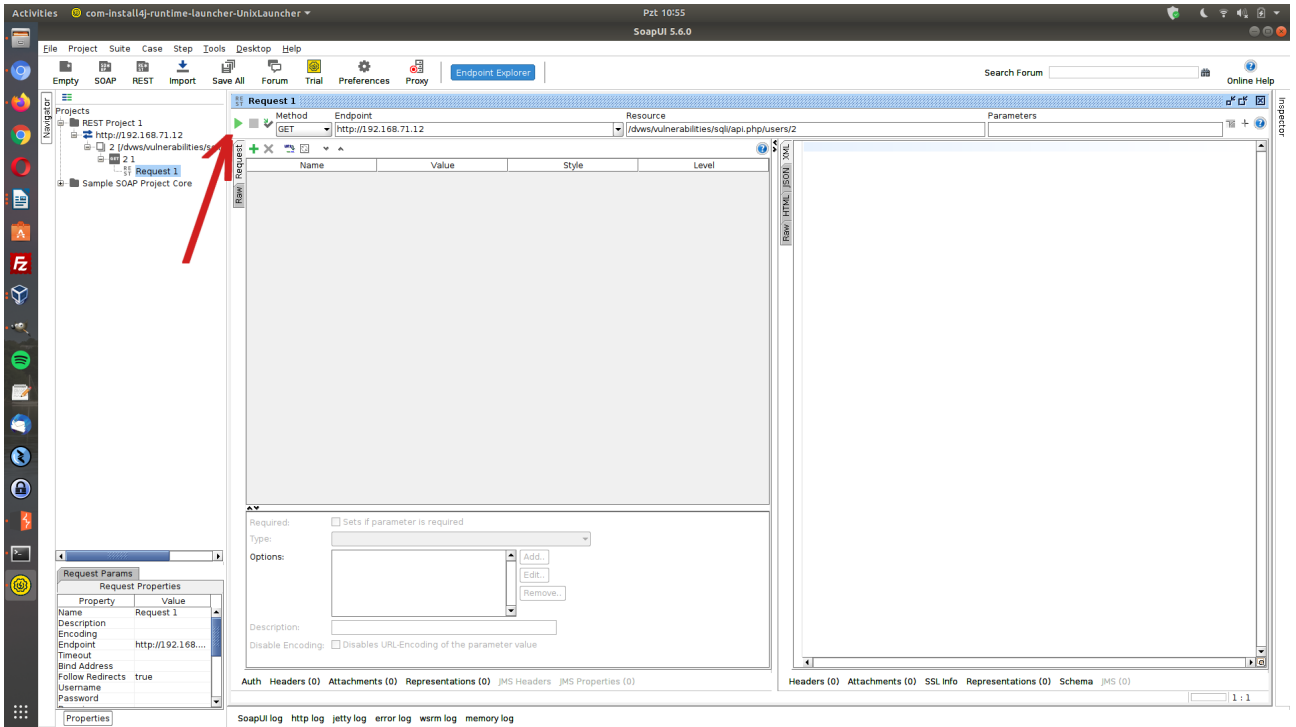
DVWS'nin bu ders sayfası ekranında rest web servisine ait url'deki 2 parametresinde sql enjeksiyonu açıklığı sunulmaktadır.

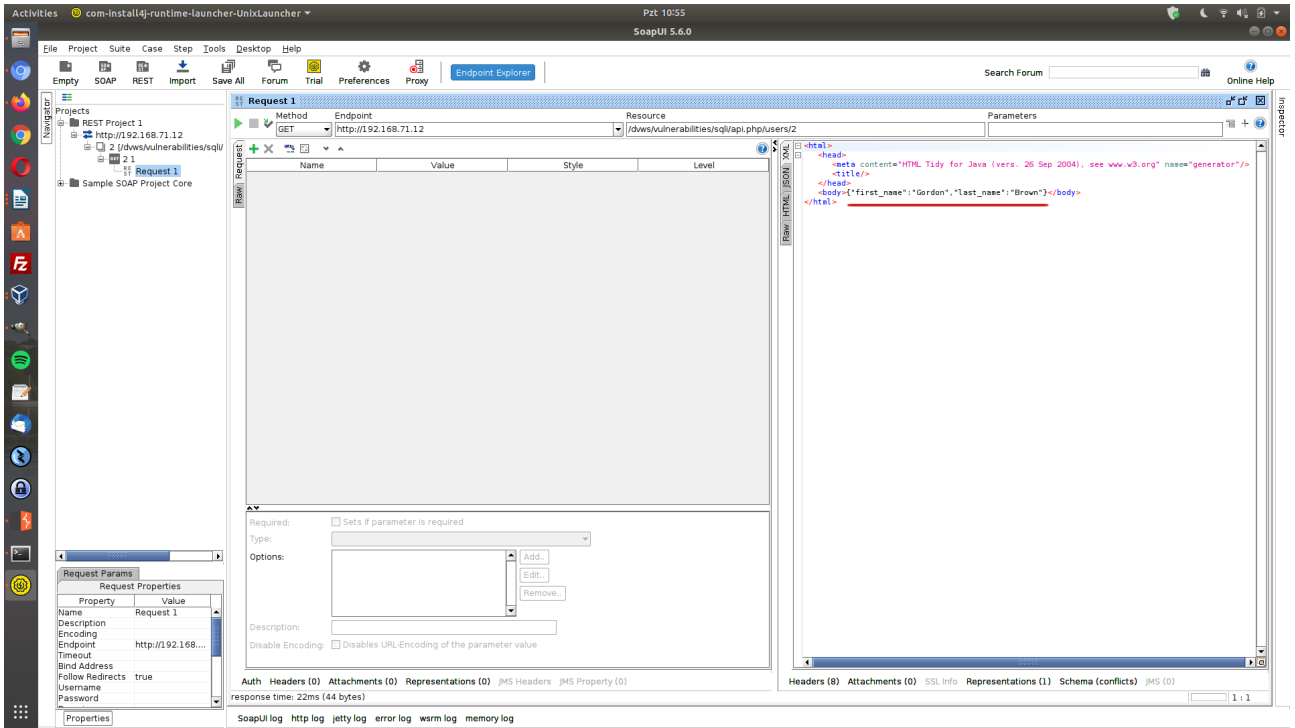
SoapUI yazılımı ile REST web servisini bu arayüzü / kapsamı göstererek test edelim ve sql enjeksiyonu açıklığını tespit edelim.





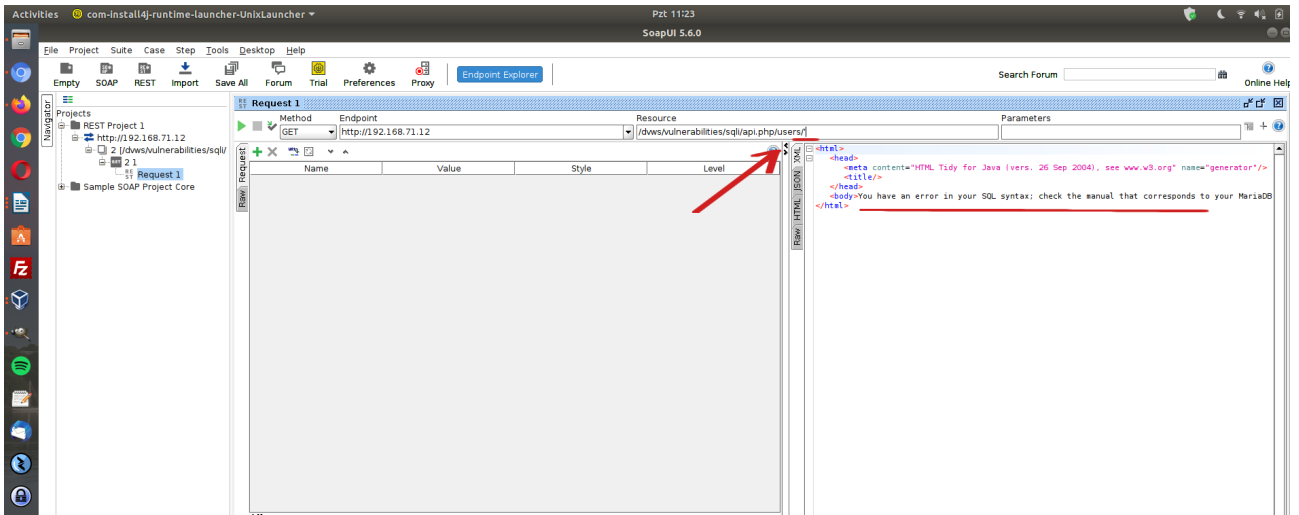
Hedef web servisi tanımlama dosyası WADL sunmadığından sadece bir url ile arayüz / kapsam girilir ve Ok denir. Böylece örnek bir talep paketi oluşacaktır. Bu paket gönderilerek hedef web servisine talep yapılır ve yanıt döner.

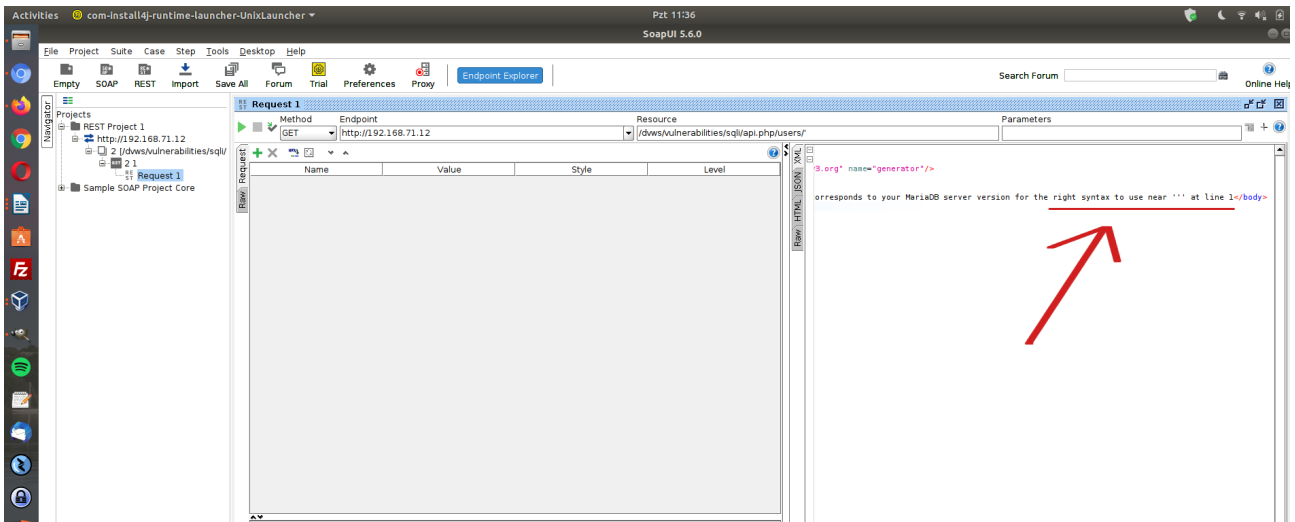




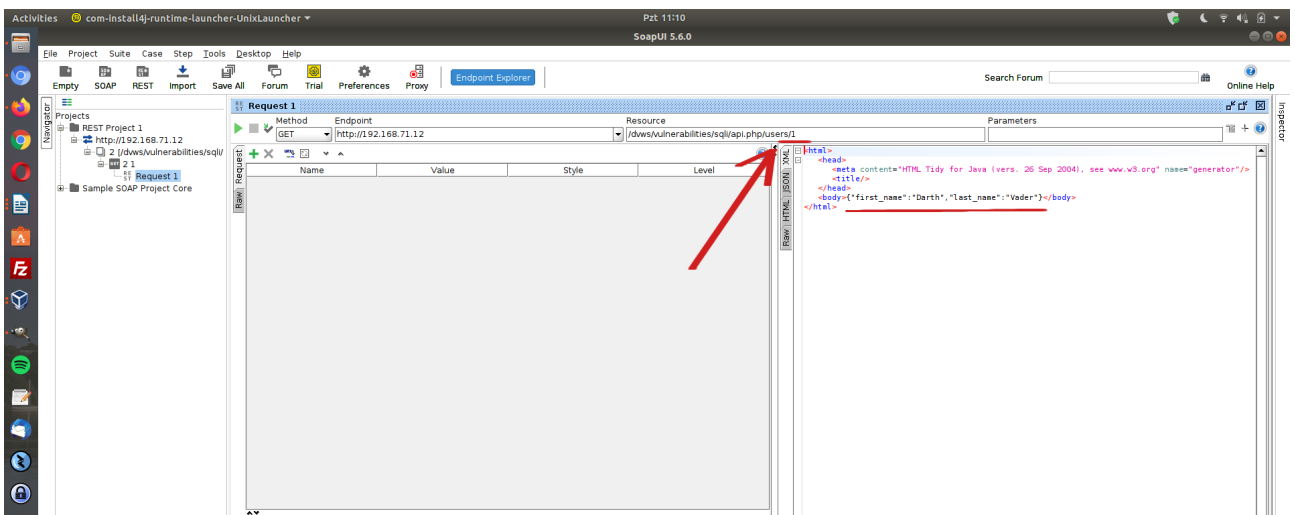
Şimdi 2 parametresine sql enjeksiyonu payload'ları girelim ve açıklık var mı tespit edelim.

Girilen Payload 1: '

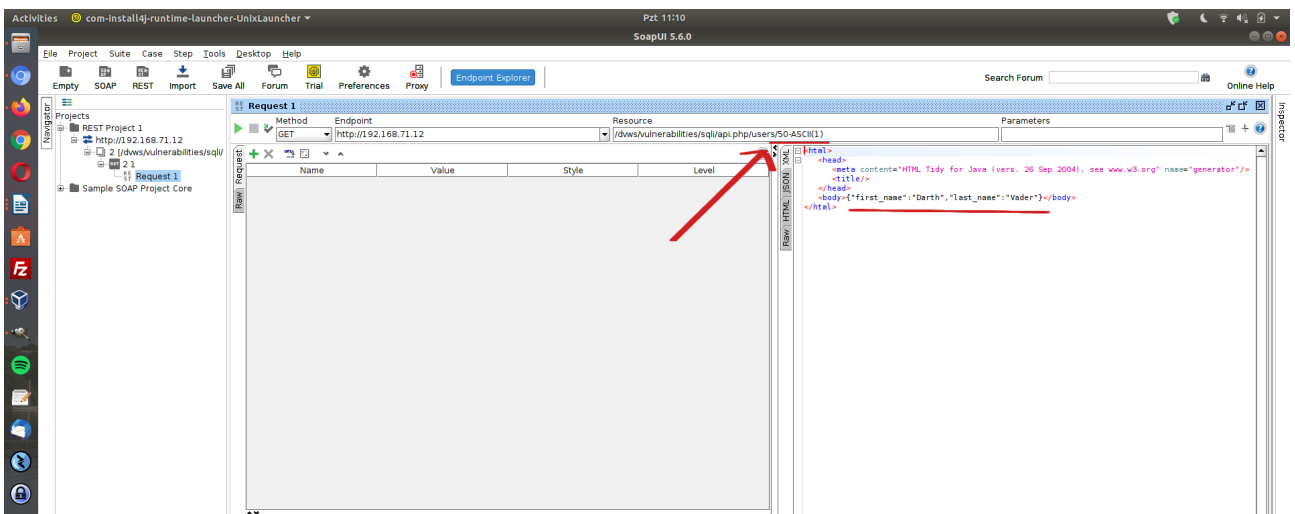




Girilen Payload 2: 1



Girilen Payload 3: 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 yine 1 kaydı yanıt olarak dönecektir. 1 payload'u ile 50-ASCII(1) payload'u aynı çıktıyı verdiğiinden demek ki ASCII() sql ifadesi çalışmıştır. Sql enjeksiyonu açıklığı var tespiti sonrası sql ifadeleri uygun şekilde girilerek sql enjeksiyonu gerçekleştirilebilir.

Bilgi:

Netsparker ile bu rest web servis url'si taranmıştır (bkz. Uygulama [Netsparker ile REST Web Servis Test Etme]) ve uygun payload'lar ile veritabanı ismi, sürümü, veritabanı kullanıcısı gibi bilgiler elde edilmiştir.

Eğer WADL, OpenAPI (Swagger),... v.b. REST web servise ait arayüz / kapsam dosyası olsaydı bu arayüz / kapsam dosyası SoapUI'ye yüklendiğinde sol sütunda örnek talepler listelenecekti ve ona göre rest web servis güvenlik denetimleri uygulanabilecekti.

Uygulama [Burpsuite WSDLER Eklentisi ile Soap Web Servis Test Etme]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

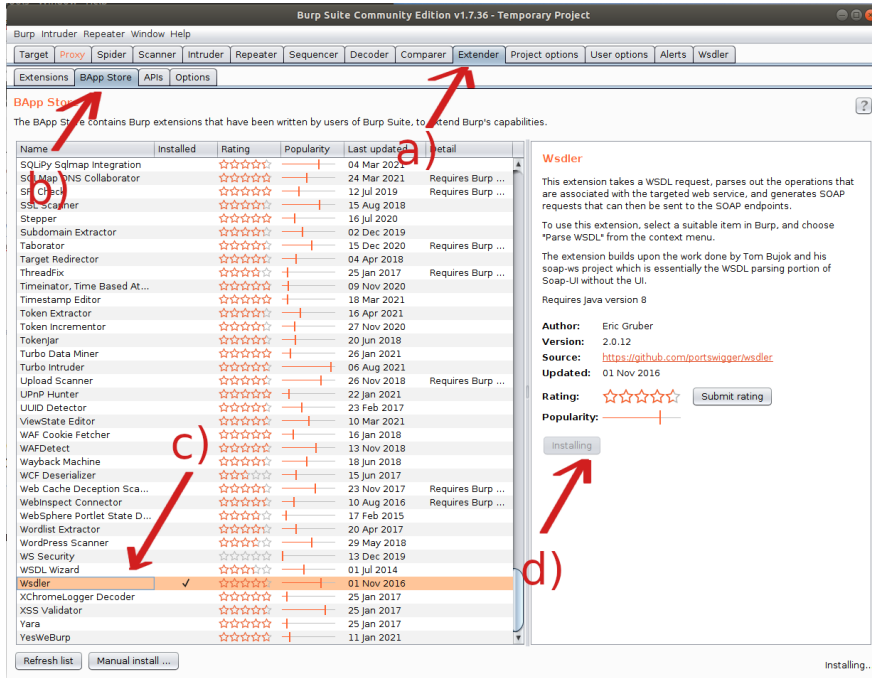
Bu uygulamada burpsuite'in wsdler eklentisi yardımıyla dvws kasıtlı zafiyetler içeren web servisinin bir ders sayfasındaki arka uçta yer alan hedef soap web servisi test edilecektir.

Gereksinimler

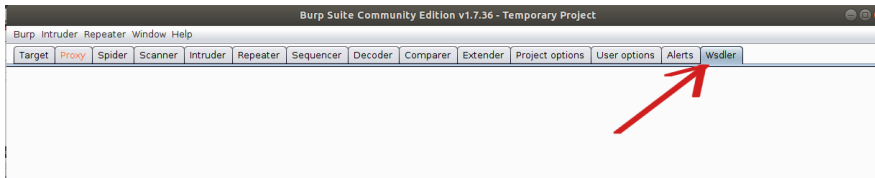
Ubuntu 18.04 LTS // Ana Makina
Burpsuite, Wsdler Plugin // Güvenlik Testi Aracı
DVWS - Windows 7 Home Premium // Web Servisi - Sanal Makine

Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışırdır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

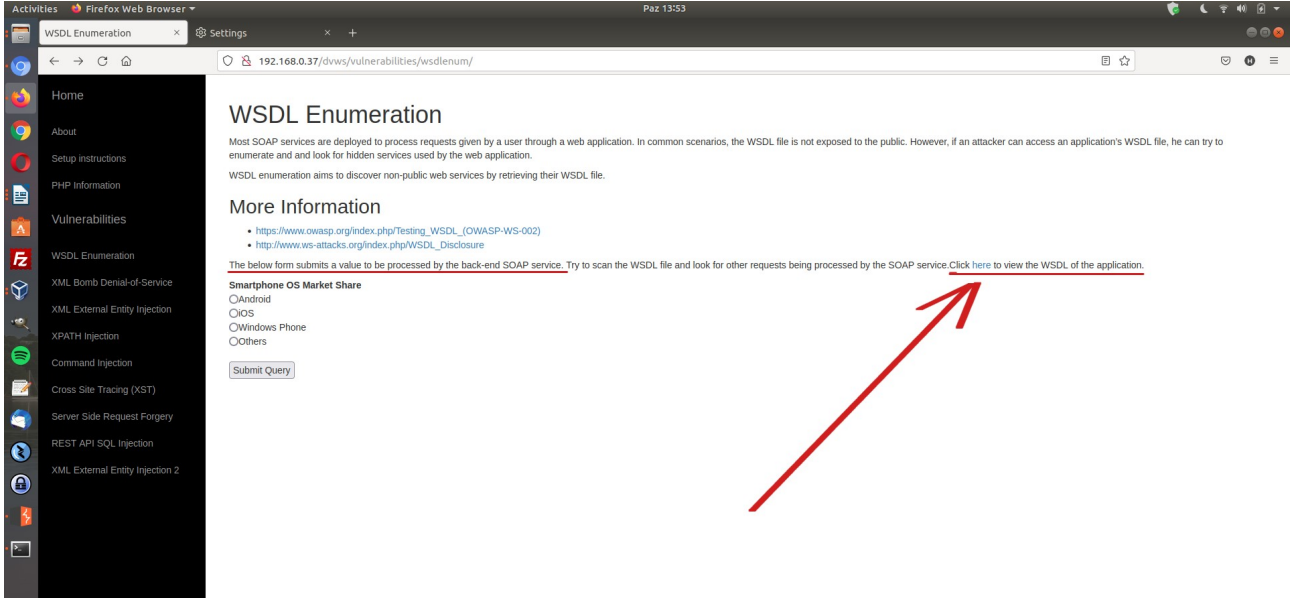
Öncelikle Burpsuite'e Wsdler eklentisini kuralım.



Bu kurulum sonrası Wsdler sekmesi burpsuite menülerine gelecektir.

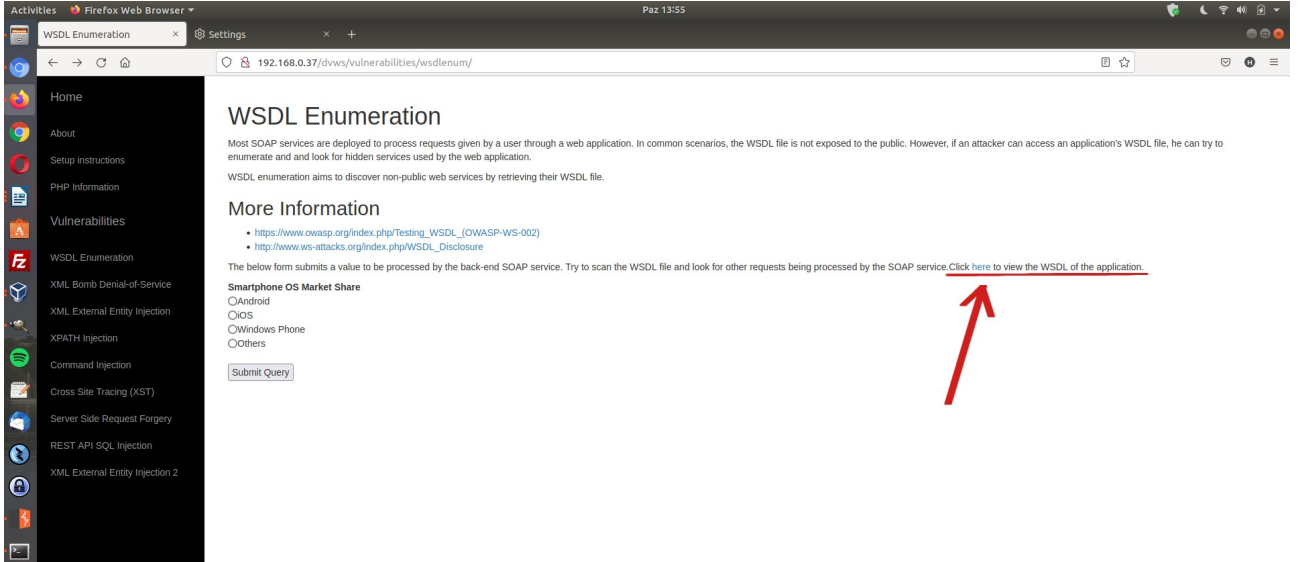


Hedef dvws web servisinin ilgili sayfasına bir göz atalım.



DVWS'nin bu sayfasında arka uçta 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 wsd dosyasını barındırdığı url'yi ziyaret edip paketi burpsuite'te yakalayalım.



(WSDL Dosyası URL'si Paylaşımı)

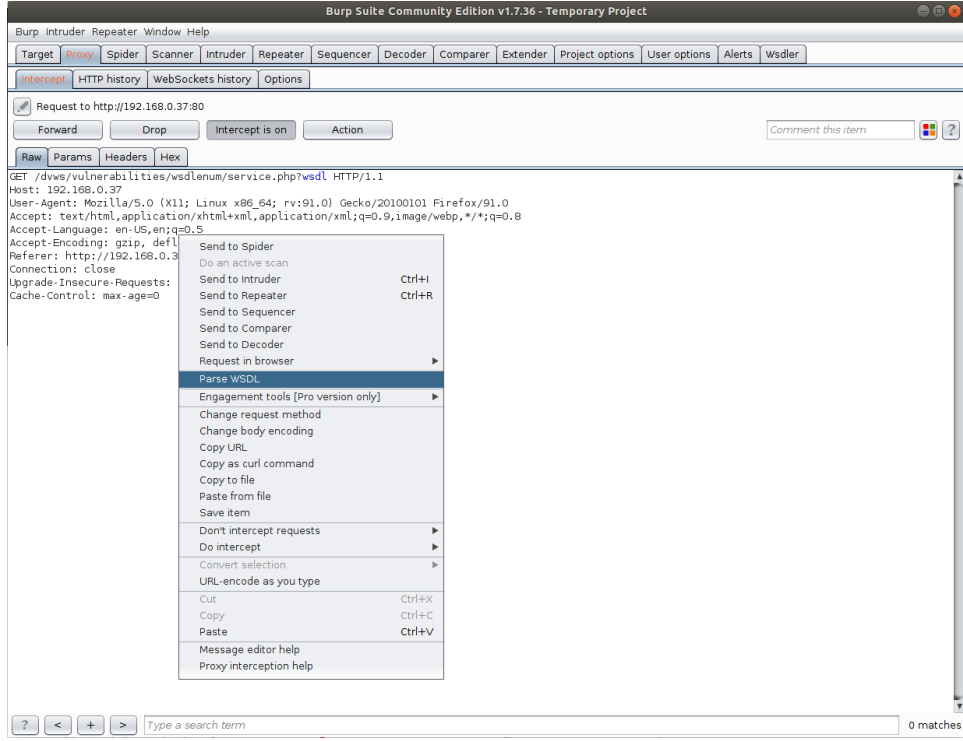
```
--<definitions targetNamespace="dvwa:webservice">
  <types>
    <xsd:schema targetNamespace="dvwa:webservice">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
    </xsd:schema>
  </types>
  <message name="Return_priceRequest">
    <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="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://192.168.0.37/dvws/vulnerabilities/wsdlenum/service.php/Return_price" style="rpc"/>
    </operation>
  </binding>
</definitions>
```

(Hedef Web Servisi Arayüzü / Kapsamı Dosyası)

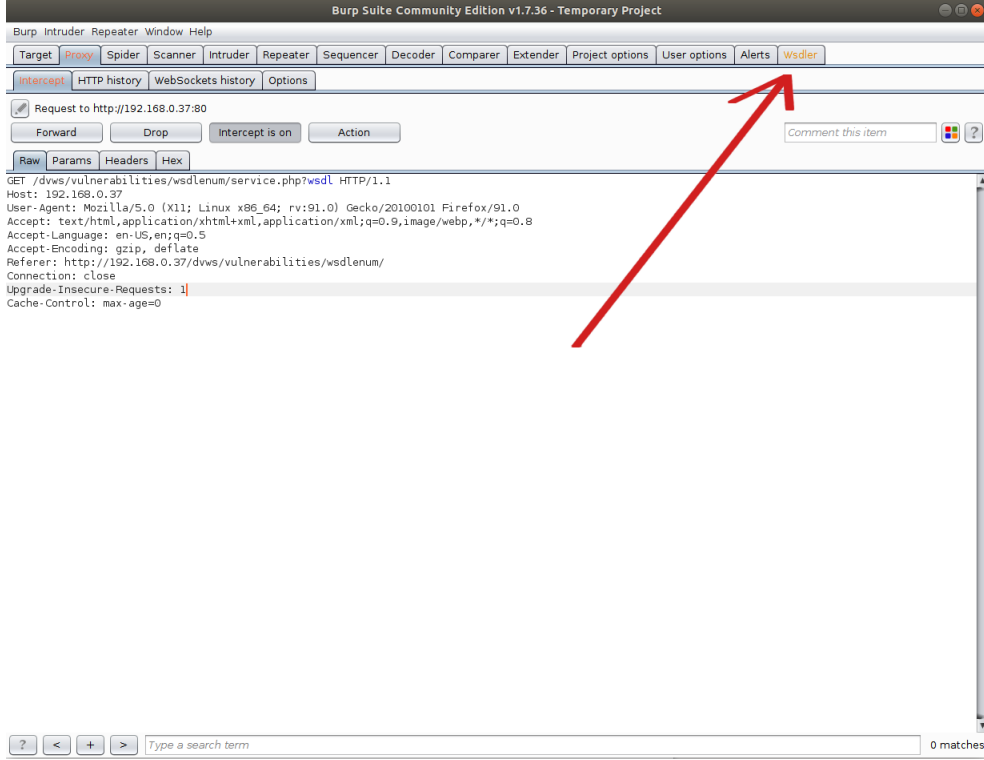
```
GET /dvws/vulnerabilities/wsdlenum/service.php?wsdl HTTP/1.1
Host: 192.168.0.37
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.37/dvws/vulnerabilities/wsdlenum/
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

(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.

Operation	Binding	Endpoint
Return_price	DVWA Web Service Binding	http://192.168.0.37/dvws/vulnerabilities/wsdlenum/service.php
check_user_id	DVWA Web Service Binding	http://192.168.0.37/dvws/vulnerabilities/wsdlenum/service.php
Population	DVWA Web Service Binding	http://192.168.0.37/dvws/vulnerabilities/wsdlenum/service.php

Request

Raw Params Headers Hex XML

```
POST /dvws/vulnerabilities/wsdlenum/service.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.37/dvws/vulnerabilities/wsdlenum/
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
SOAPAction: http://192.168.0.37/dvws/vulnerabilities/wsdlenum/service.php/Return_price
Content-Type: text/xml;charset=UTF-8
Host: 192.168.0.37
Content-Length: 414

<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <Return_price soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <name xsi:type="xsd:string">gero et</name>
    </Return_price>
  </soapenv:Body>
</soapenv:Envelope>
```

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.

Request

Raw Params Headers Hex XML

```
POST /dvws/vulnerabilities/wsdlenum/service.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.37/dvws/vulnerabilities/wsdlenum/
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
SOAPAction: http://192.168.0.37/dvws/vulnerabilities/wsdlenum/service.php/Return_price
Content-Type: text/xml;charset=UTF-8
Host: 192.168.0.37
Content-Length: 414

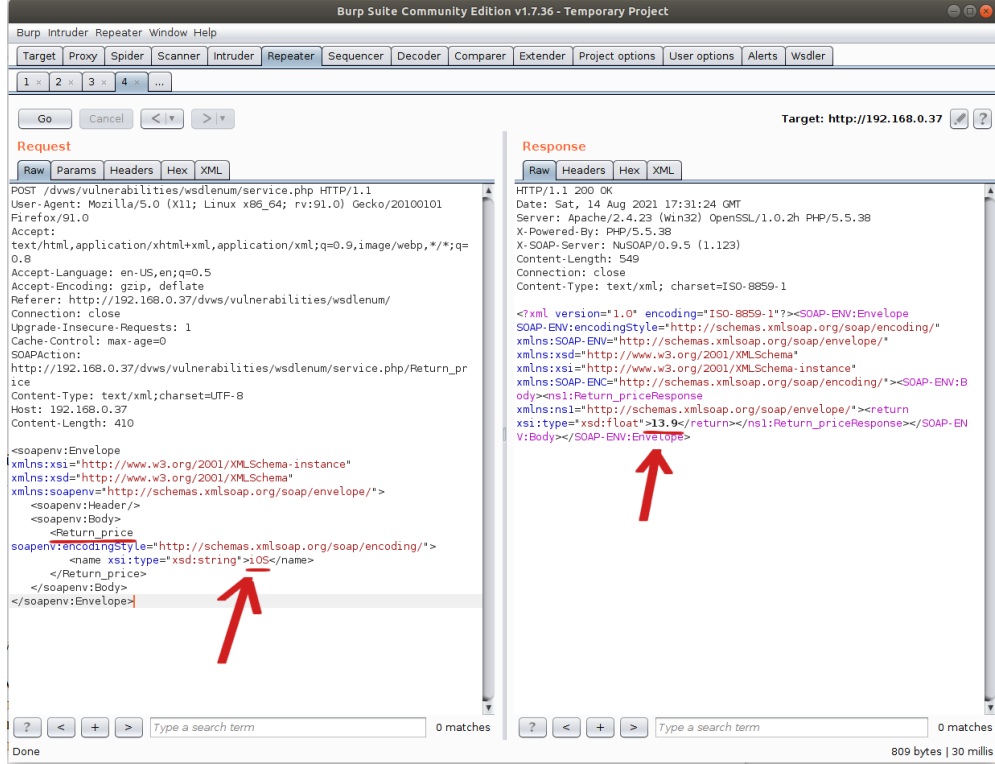
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <Return_price soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <name xsi:type="xsd:string">Android</name>
    </Return_price>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

Raw Headers Hex XML

```
HTTP/1.1 200 OK
Date: Sat, 14 Aug 2021 17:31:00 GMT
Server: Apache/2.4.23 (Ubuntu) OpenSSL/1.0.2h PHP/5.5.38
X-Powered-By: PHP/5.5.38
X-SOAP-Server: NuSOAP/0.9.5 (1.129)
Content-Length: 549
Connection: close
Content-Type: text/xml; charset=ISO-8859-1

<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><ns1:Return_priceResponse xmlns:ns1="http://schemas.xmlsoap.org/soap/envelope/"><return xsi:type="xsd:float">82.8</return></ns1:Return_priceResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```



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

Bu řekilde testler uygulanarak gelen yanıtlar izlenebilir ve bir ađıklık aranabilir.

Bilgi:

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

Uygulama [Burpsuite ile Rest Web Servis Test Etme]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

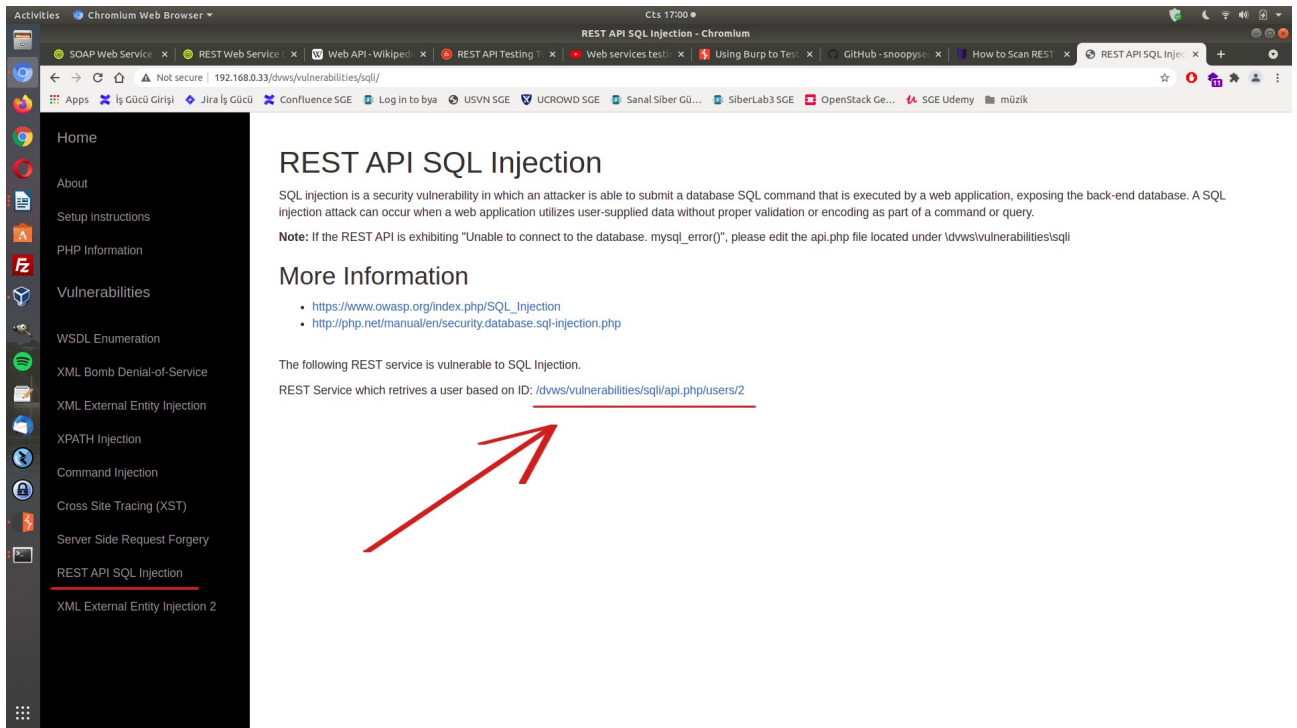
Bu uygulamada burpsuite ile kasıtlı zafiyetler içeren dvws web servisinin bir ders sayfasındaki arka uçta yer alan hedef rest web servisi test edilecektir. Not: Bu uygulama Burpsuite'in resmi sayfasında rest web servisleri test etme makalesinde uygulama olarak gösterilmektedir. (bkz. <https://portswigger.net/support/using-burp-to-test-a-rest-api>)

Gereksinimler

Ubuntu 18.04 LTS // Ana Makina
Burpsuite // Güvenlik Testi Aracı
DVWS - Windows 7 Home Premium // Web Servisi - Sanal Makine

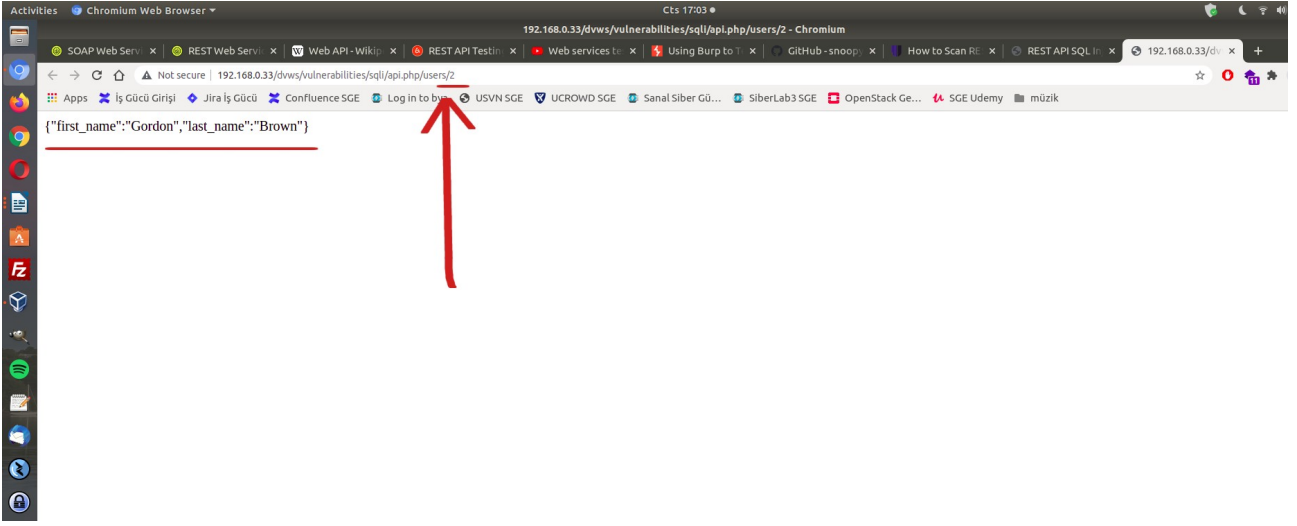
Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışırdır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

Öncelikle dvws web servisindeki ilgili sayfaya bir göz atalım.

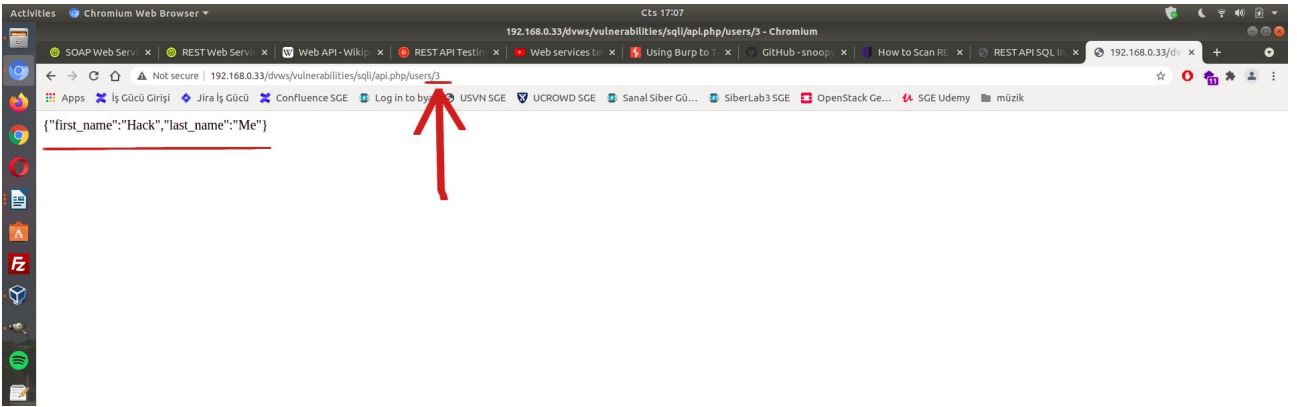


(DVWS Web Servisindeki REST Web Servisi Kısmı)

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



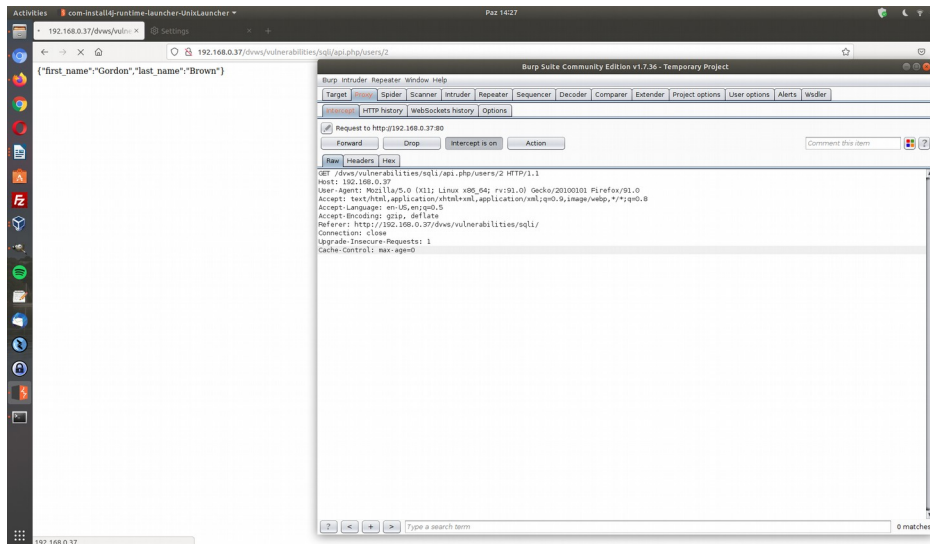
(REST Web Servis URL Parametresi 2 iken Gelen Veri)



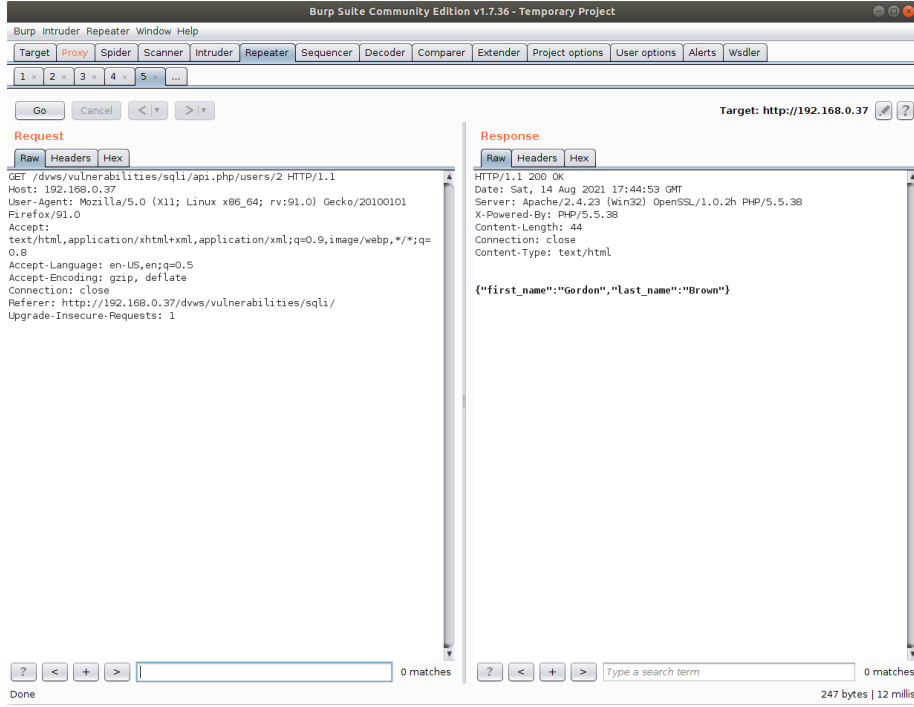
(Rest Web Servis URL Parametresi 3 iken Gelen Veri)

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.



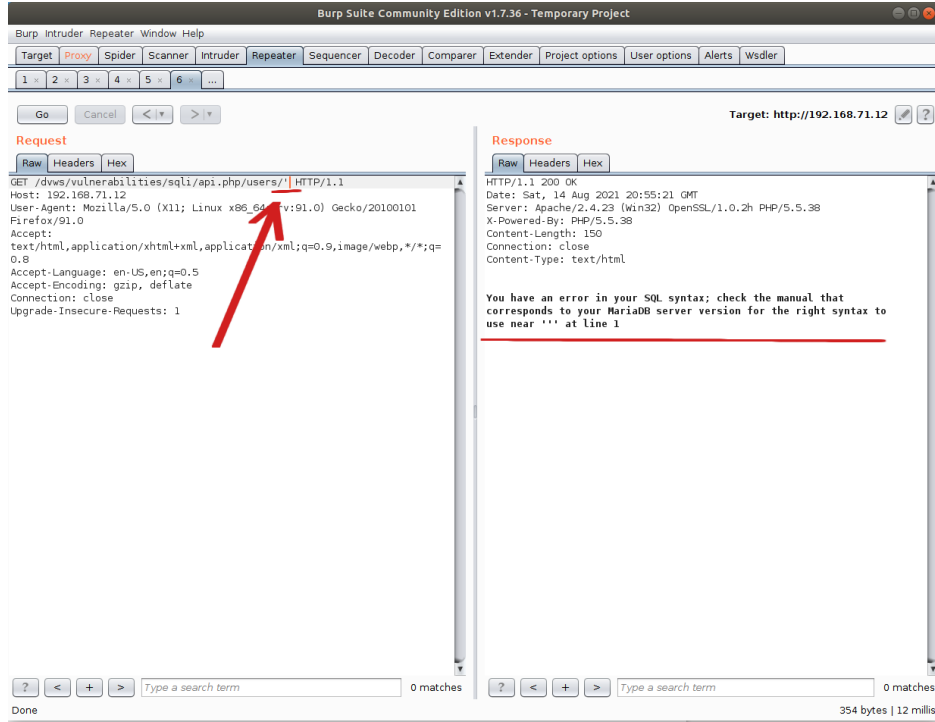
(Rest Web Servis Endpoint URL'si Http Talep Paketi Yakalanır)



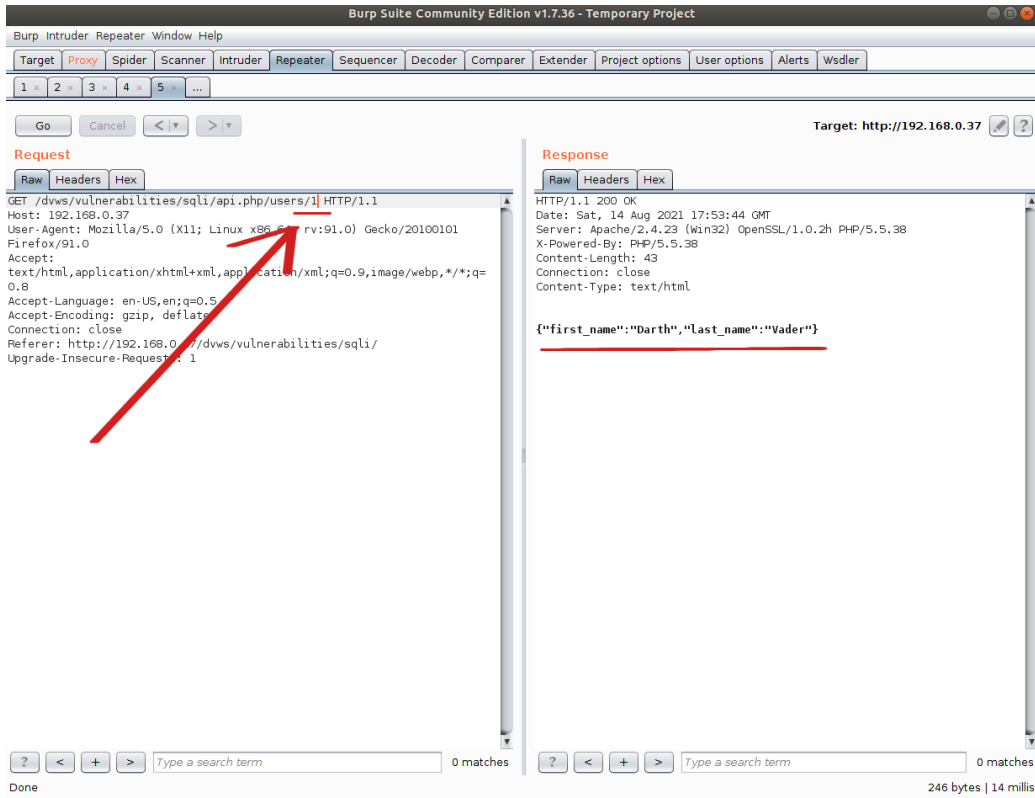
(Rest Web Servis Endpoint URL'si Http Talep Paketi Repeater'a Atılır)

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

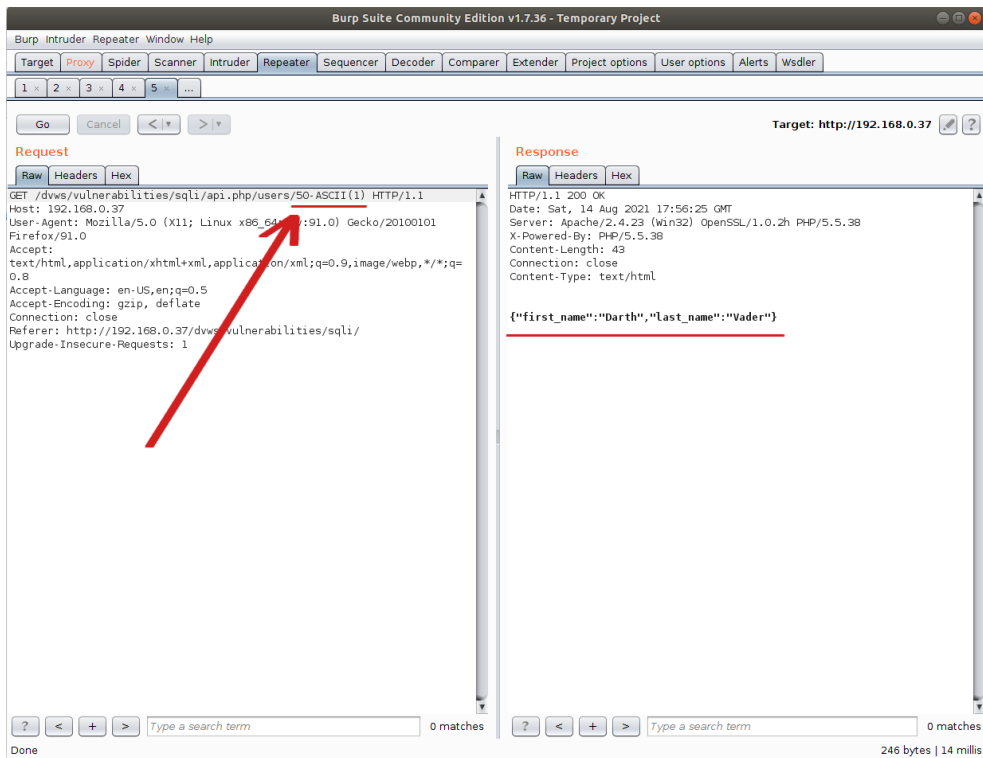
Girilen Payload: '



Girilen Payload: 1



Girilen 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 verdiğiinden demek ki ASCII() sql ifadesi çalışmıştır. Sql enjeksiyonu açıklığı var tespiti sonrası sql ifadeleri uygun şekilde girilerek sql enjeksiyonu gerçekleştirilebilir.

Bilgi:

Netsparker ile bu rest web servis url'si taranmıştır (bkz. Uygulama [Netsparker ile REST Web Servis Test Etme]) ve uygun payload'lar ile veritabanı ismi, sürümü, veritabanı kullanıcısı gibi bilgiler elde edilmiştir.

Bu şekilde Burpsuite ile rest web servisi güvenlik testleri uygulanabilir.

Uygulama [Netsparker Yazılımı ile Soap Web Servis Test Etme]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

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

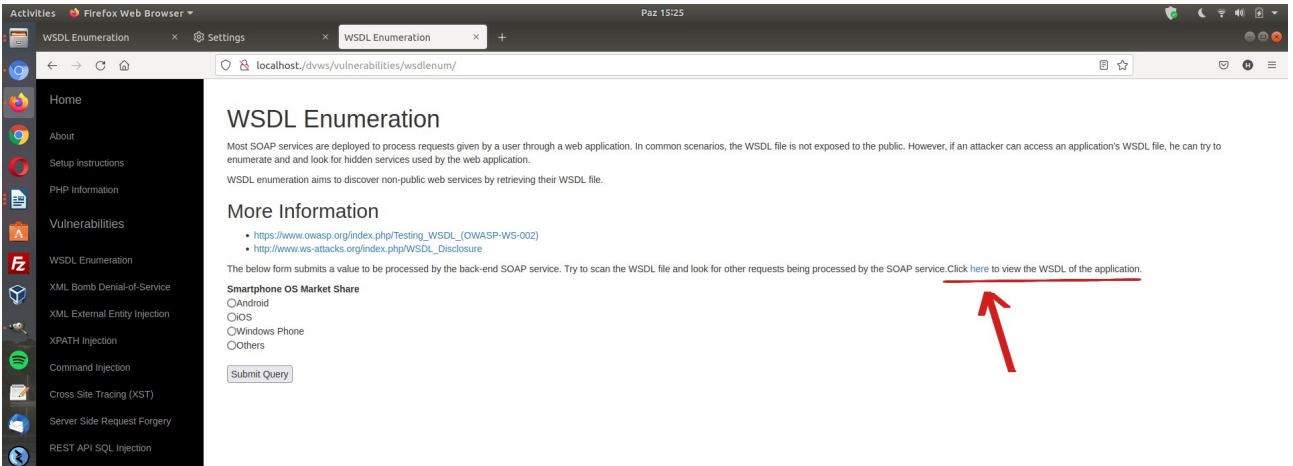
Gereksinimler

Ubuntu 18.04 LTS	// Ana Makina
Netsparker	// Güvenlik Testi Aracı
DVWS - Windows 7 Home Premium	// Hedef Web Servisi - Sanal Makine

Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

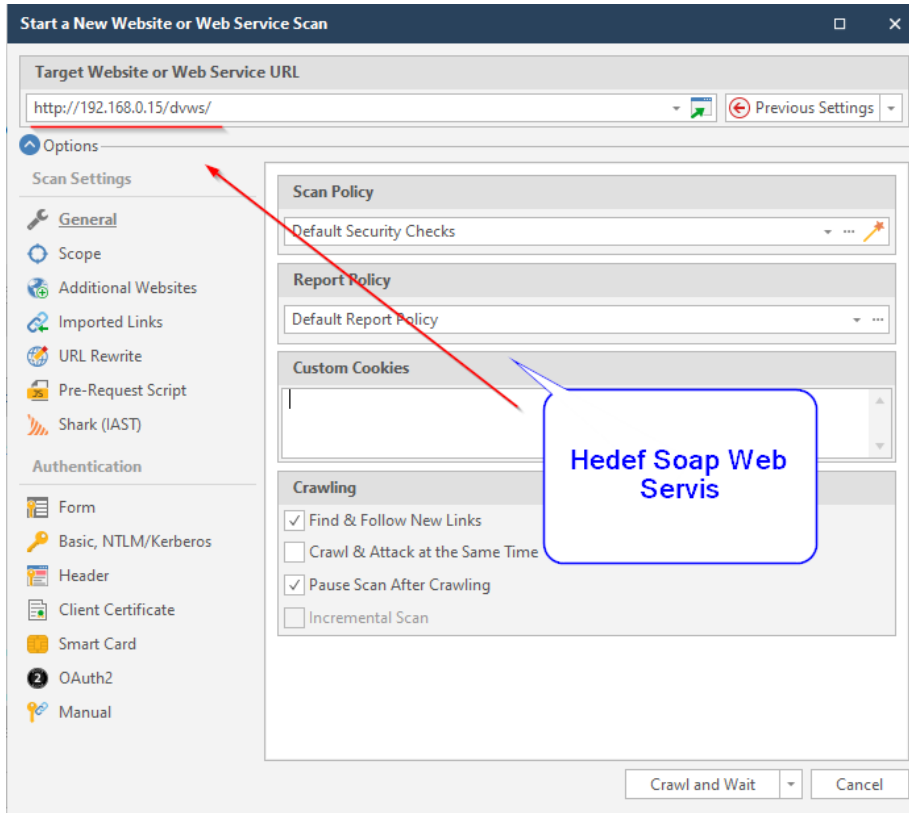
Netsparker ile soap web servis tarayabilmek için soap web servislerde arayüz / kapsam sunan WSDL dosyasını Netsparker tarama ayarlarından Import Links seçeneği ile yüklemek gerekmektedir. Bu şekilde Netsparker otomatize tarama aracı soap web servisin arayüzünü / kapsamını görebilecektir ve saldırı testlerini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

Öncelikle hedef dvws soap web servisinin WSDL dosyasını elde edelim.

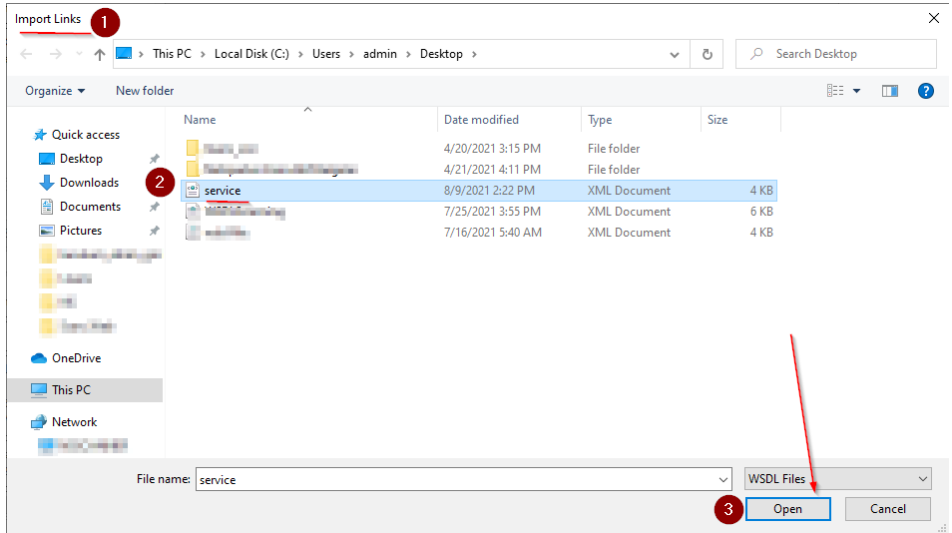
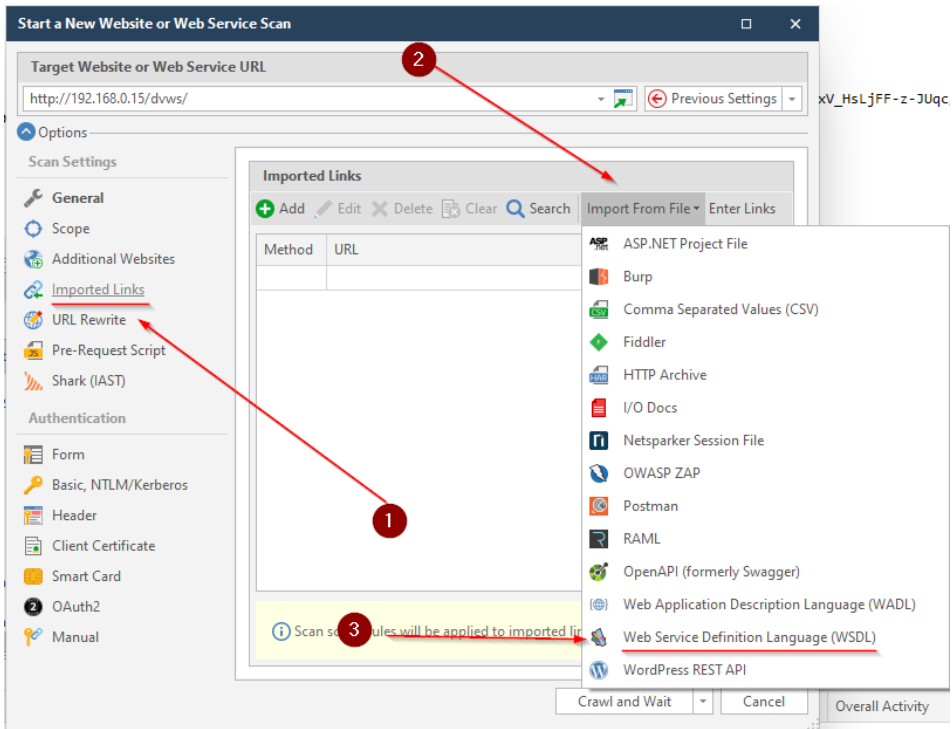


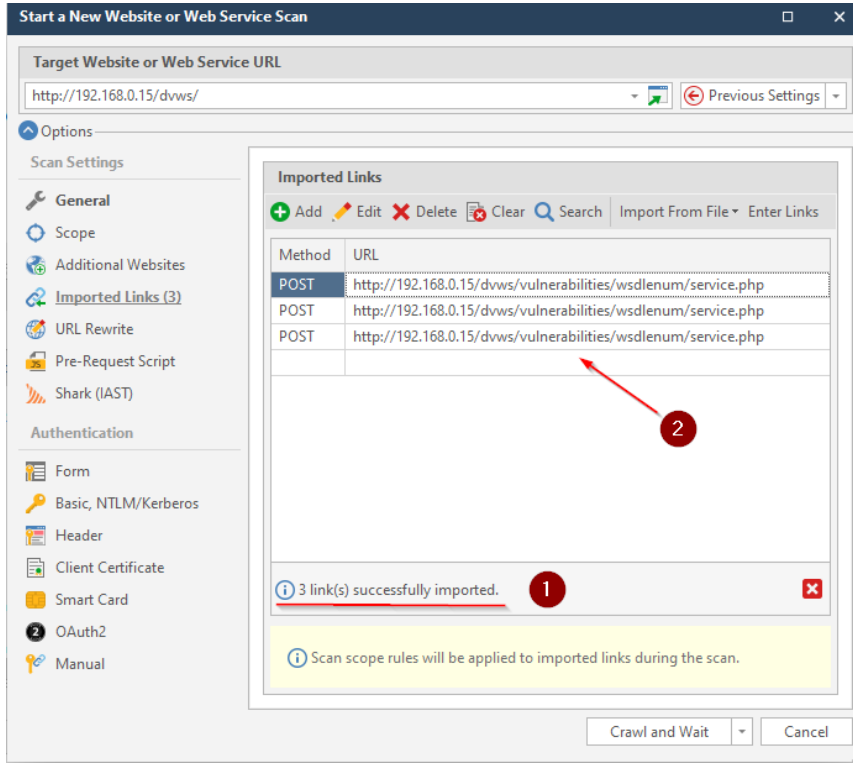
```
Activities Firefox Web Browser Paz 15:26
WSDL Enumeration Settings localhost/dvws/vulnerabil...
localhost/dvws/vulnerabilities/wsdlenum/service.php?wsdl
This XML file does not appear to have any style information associated with it. The document tree is shown below.
--<definitions targetNamespace="dvwa:webservice">
  <types>
    <xsd:schema targetNamespace="dvwa:webservice">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
      <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="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>
      </xsd:schema>
    </types>
  </definitions>
--</input>
```

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

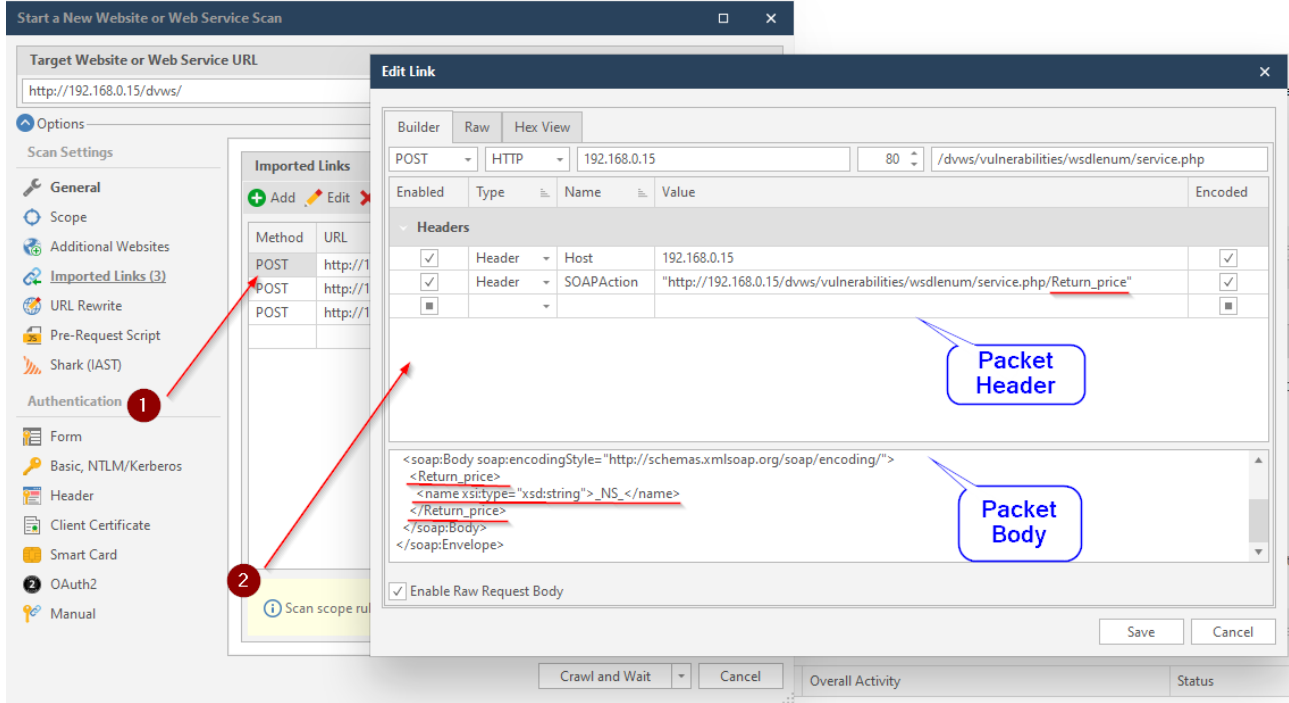


Daha sonra hedef soap web servisin arayüzünü / kapsamını yükleyelim.

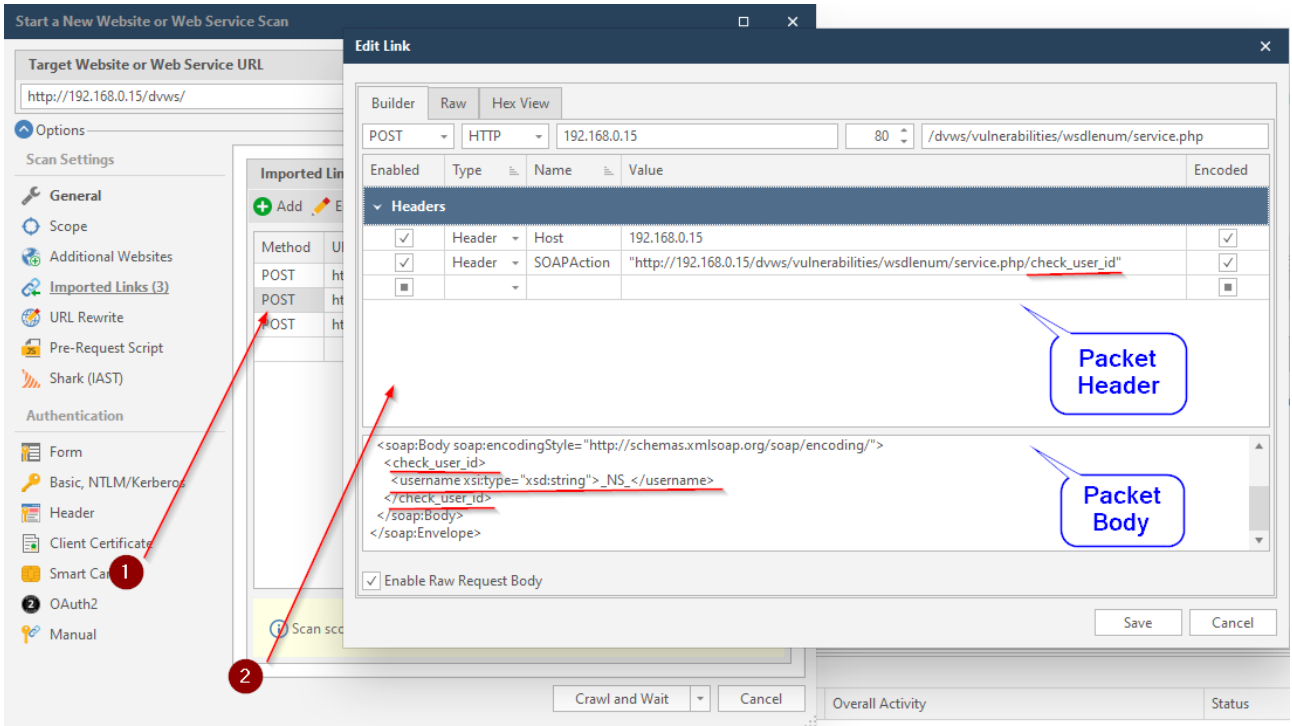




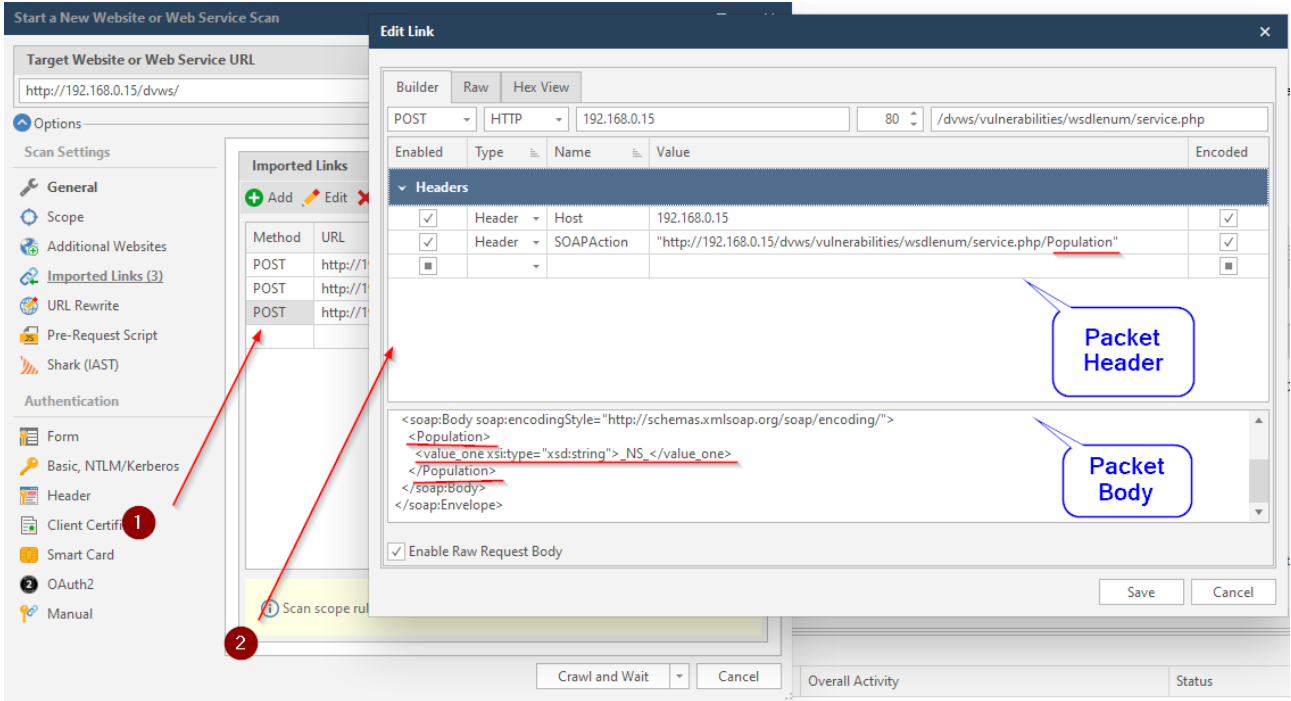
Arayüz / Kapsam dosyası yüklemesi sonrası hedef dvws soap web servise ait 3 adet link eklenir. Bu linkler hedef dvws soap web servisinin kabul ettiği xml talep paketleri şeklinde listelenecektir.



(Eklenen Birinci Örnek XML Talep Paketi)

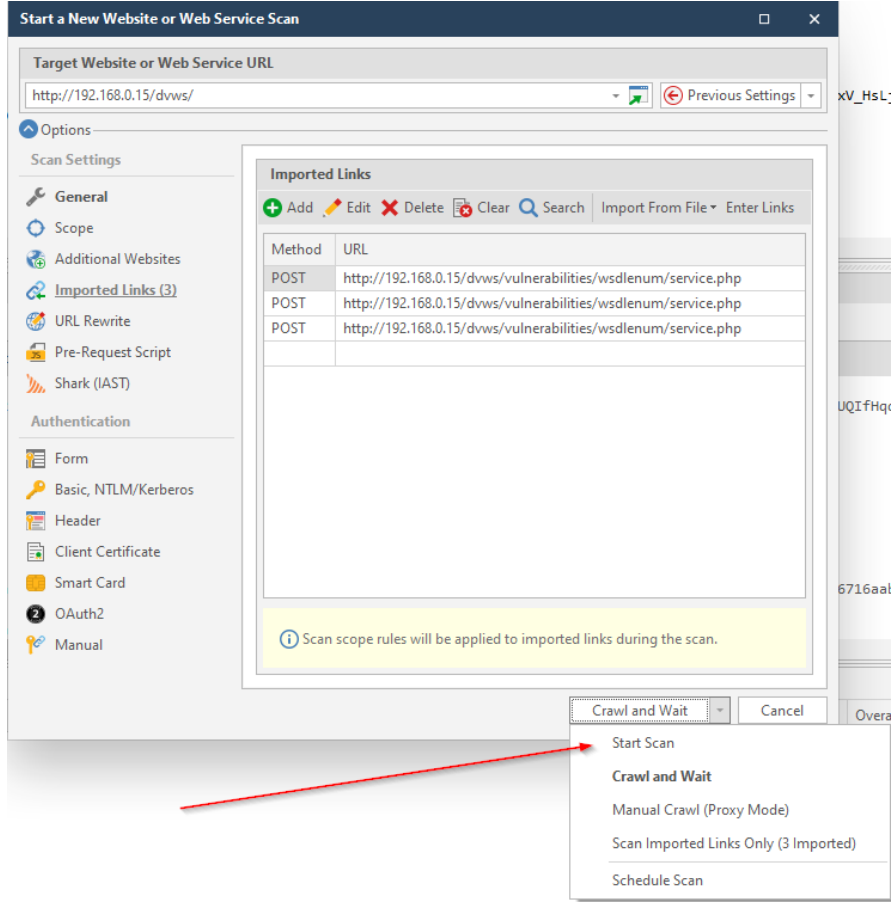


(Eklenen İkinci Örnek XML Talep Paketi)



(Eklenen 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.

Soap Web Servisi Tespit Edildi Bilgisi

Soap Web Servisi Tespit Edildi Bilgisi

Operation	Parameter	Return_price
check_user_id	/soap:Envelope[1]/soap:Body[1]/check_user_id[1]/username[1]/text () [1]	/soap:Envelope[1]/soap:Body[1]/Return_price[1]/name [1]/text () [1]
Population	/soap:Envelope[1]/soap:Body[1]/Population[1]/value_one[1]/text () [1]	/soap:Envelope[1]/soap:Body[1]/Return_price[1]/name [1]/text () [1]

Method	Target	Parameter	Duration	Current Activity	Overall Activity	Status
POST	https://192.168.0.15/dvwa/vulnerabilities/cvss/...	(value=DVWS)	1s			Loading (DOM/S)
GET	https://192.168.0.15/dvwa/vulnerabilities/paths/...		1s			Parsing (DOM/S)
GET	https://192.168.0.15/dvwa/vulnerabilities/paths/...		1s			Loading (DOM/S)
GET	https://192.168.0.15/dvwa/vulnerabilities/cmd/...		1s			Analyzing

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 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.

Uygulama [Netsparker Yazılımı ile Rest Web Servis Test Etme]

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

Gereksinimler

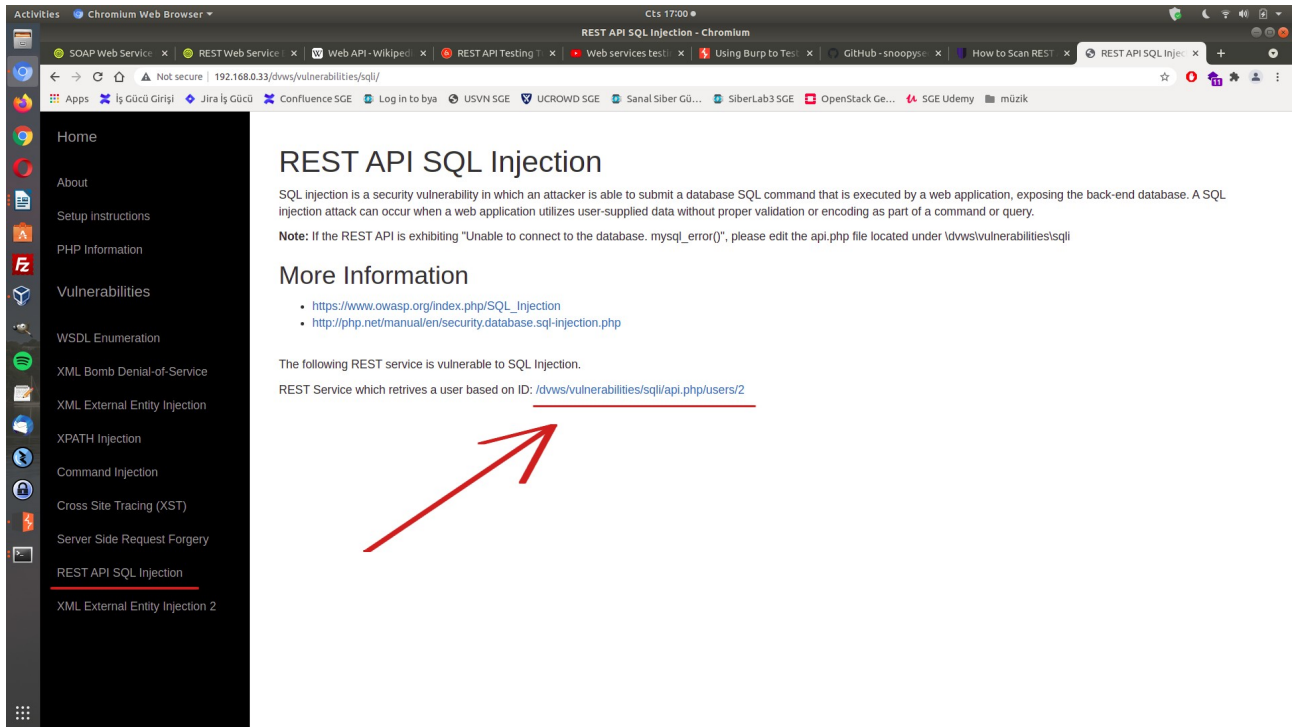
Ubuntu 18.04 LTS	// Ana Makina
Netsparker	// Güvenlik Testi Aracı
DVWS - Windows 7 Home Premium	// Hedef Web Servisi - Sanal Makine

Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışırdır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

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.

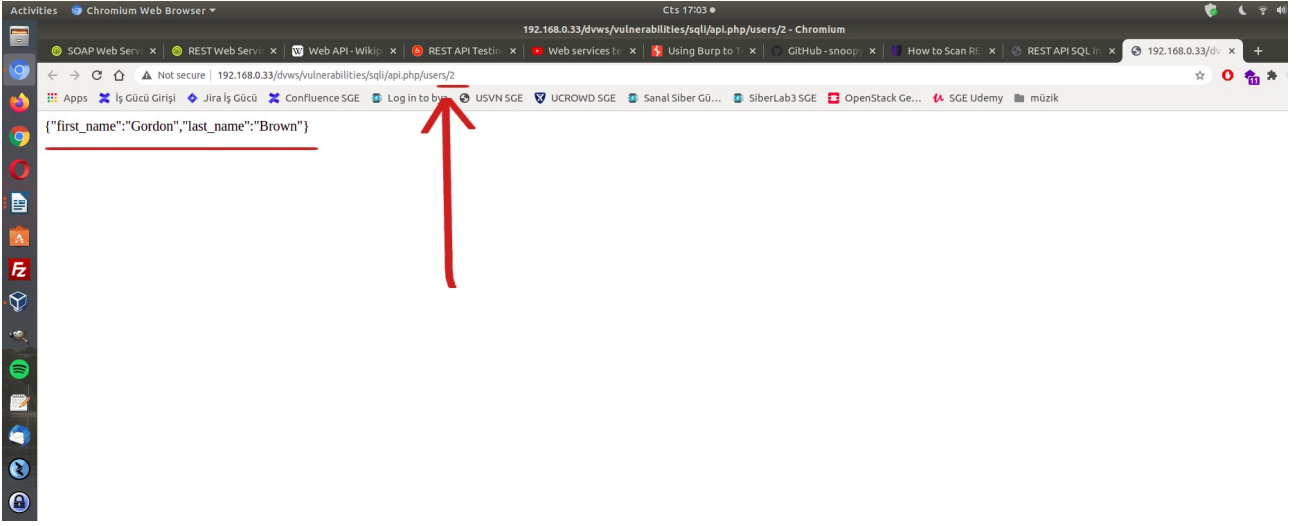
DVWS web servisi kompakt halde test amaçlı hem SOAP web servis hem de REST web servis barındırmaktadır. REST web servisi için bir tanımlama dosyası bulundurmamaktadır. Bunun yerine bir adet url şeklinde arayüz / kapsam sunmaktadır. Bu nedenle netsparker'a arayüz / kapsam bu bir url ile eklenecektir.

Öncelikle dvws web servisindeki ilgili sayfayı görelim.

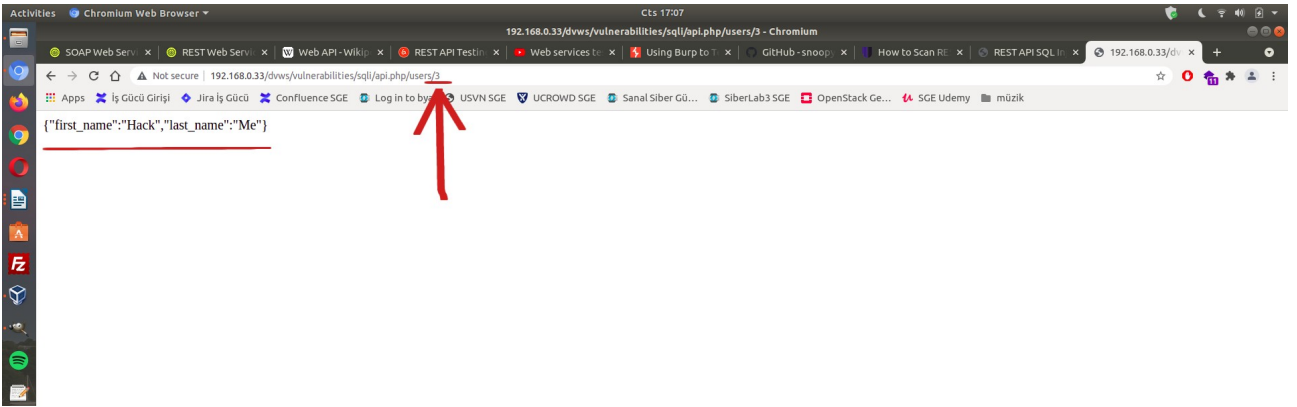


(DVWS Web Servisindeki REST Web Servisi Kısmı)

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



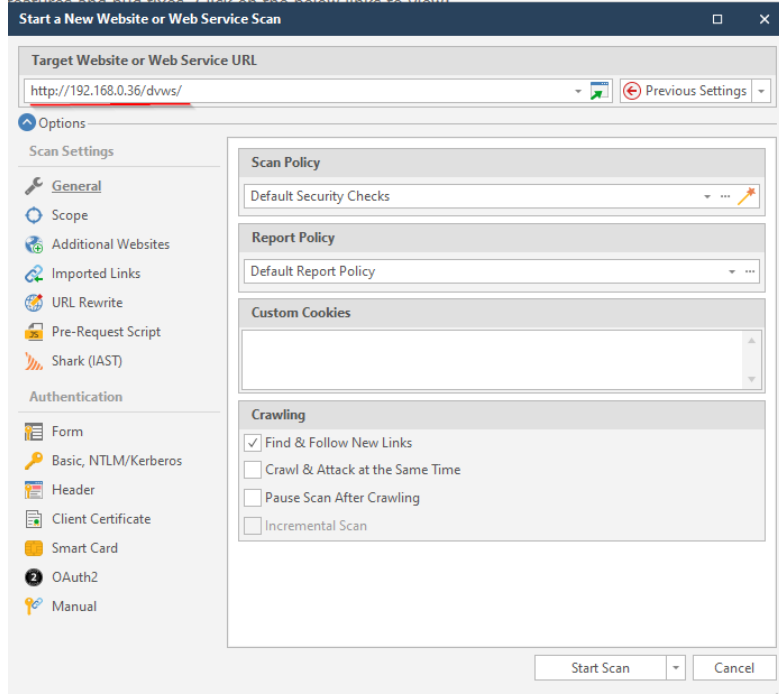
(REST Web Servis URL Parametresi 2 iken Gelen Veri)



(Rest Web Servis URL Parametresi 3 iken Gelen Veri)

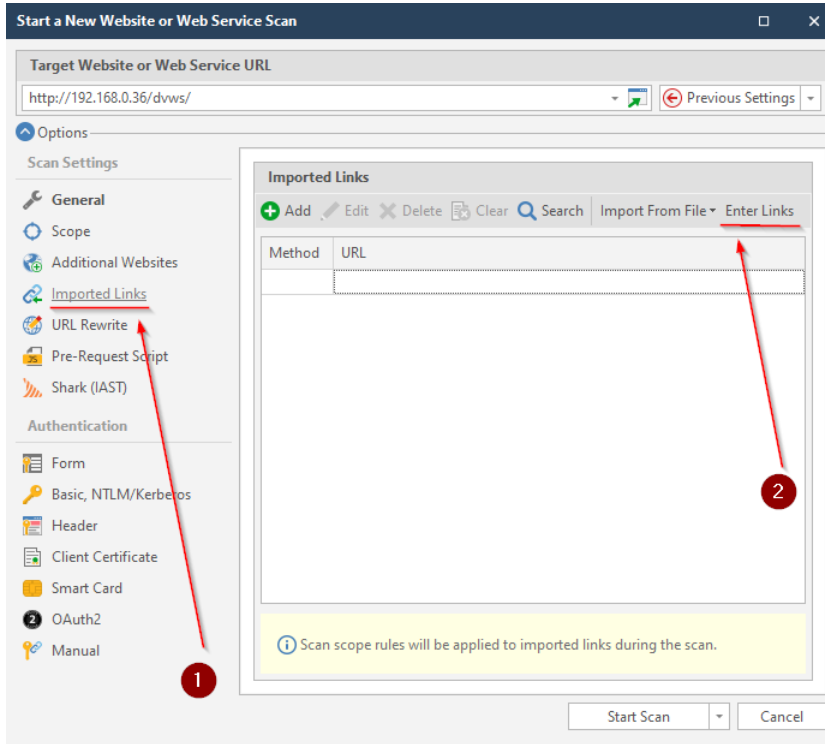
DVWS'nin bu ders sayfası ekranında rest web servis url'sindeki 2 parametresinde sql enjeksiyonu açıklığı sunulmaktadır.

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

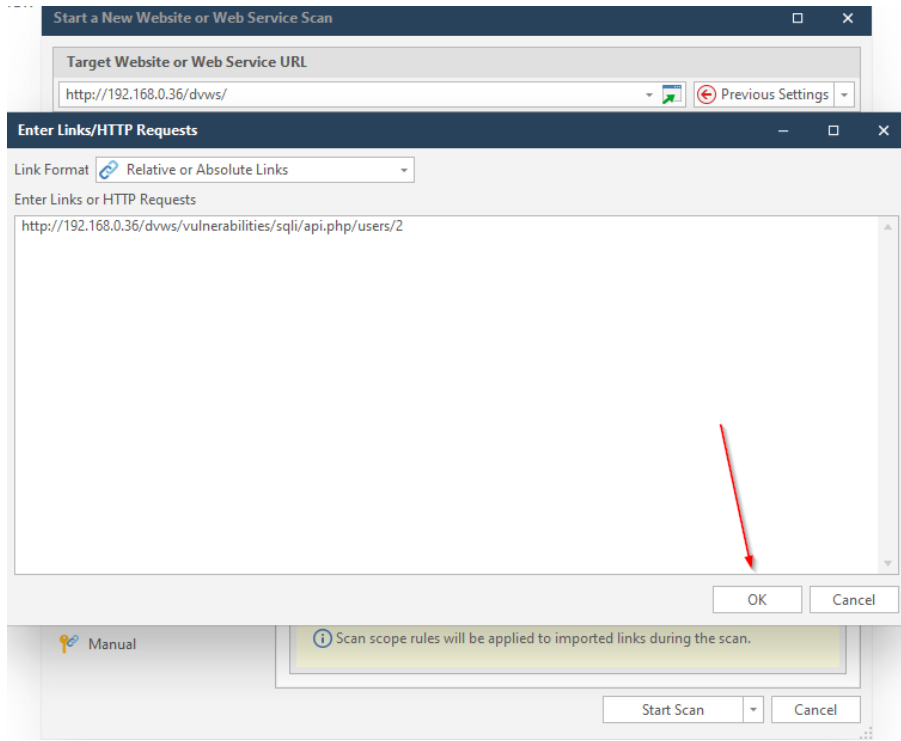


(Netsparker Taramaya Web Servis Adresi Girilir)

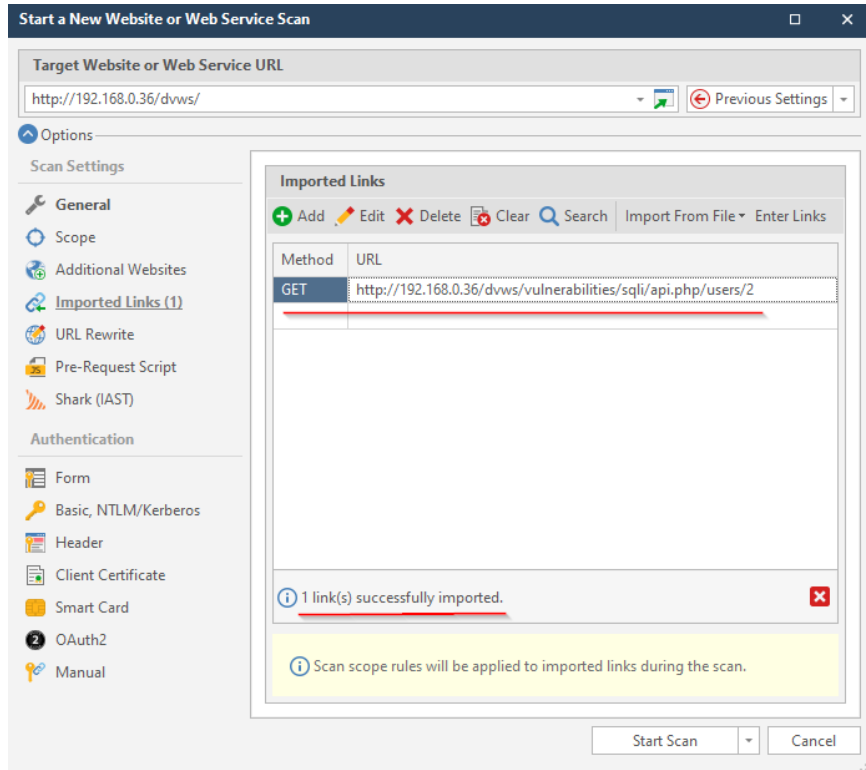
Ardından rest web servis arayüzü / kapsamı tanımlama dosyası olmadığından Import Links seçeneğinde arayüz / kapsam dosyası import etmek yerine bir adet url şeklinde ekleme yapalım.



(Web Servis Arayüz / Kapsam Yükleme Ekranına Gidilir)



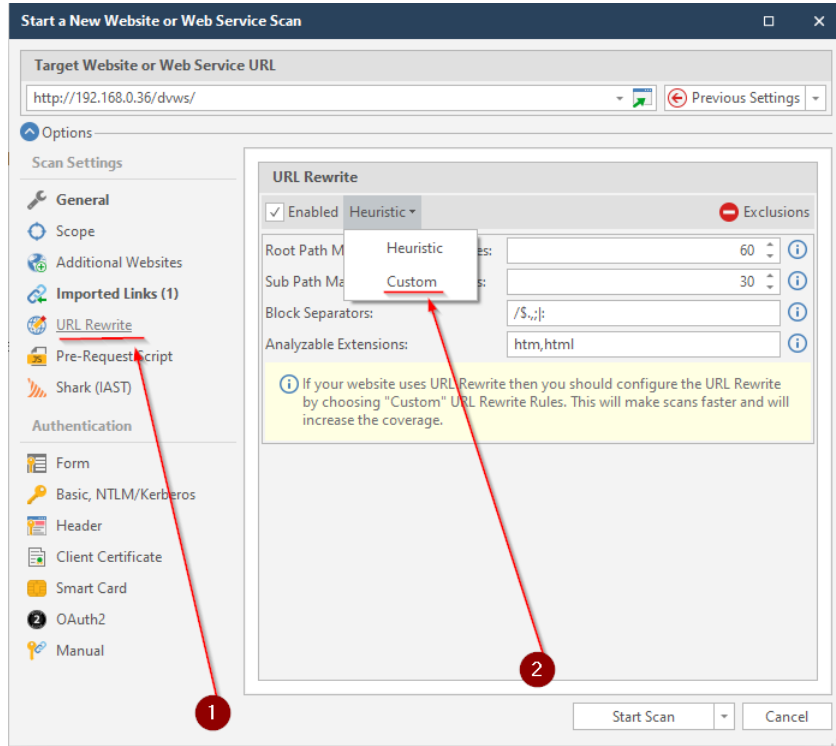
(Endpoint URL'i Girilir)



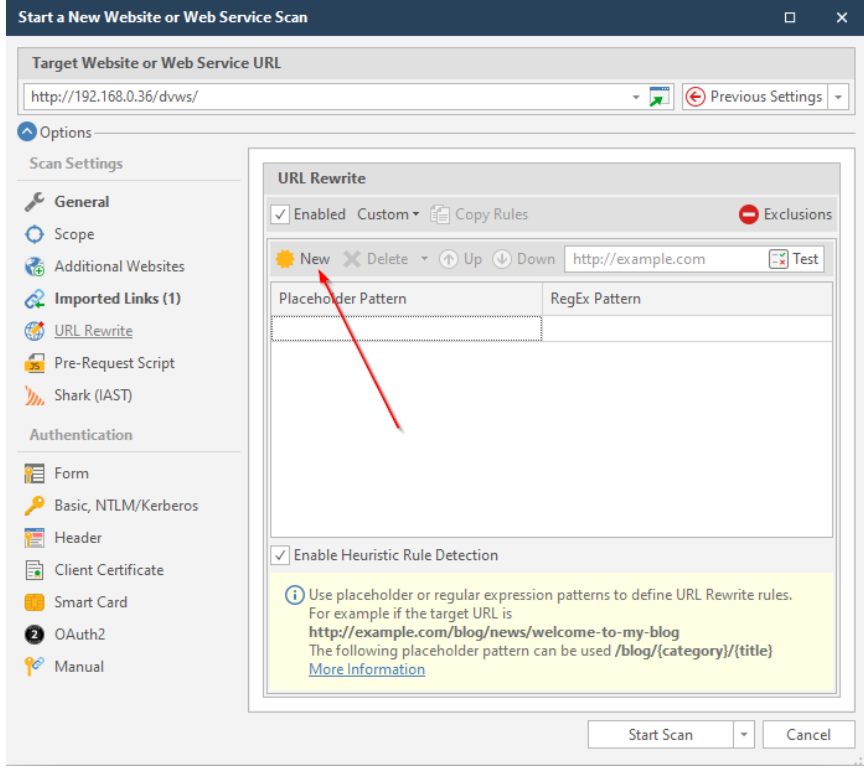
(Rest Web Servis Arayüzü / Kapsamı Eklenir)

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.

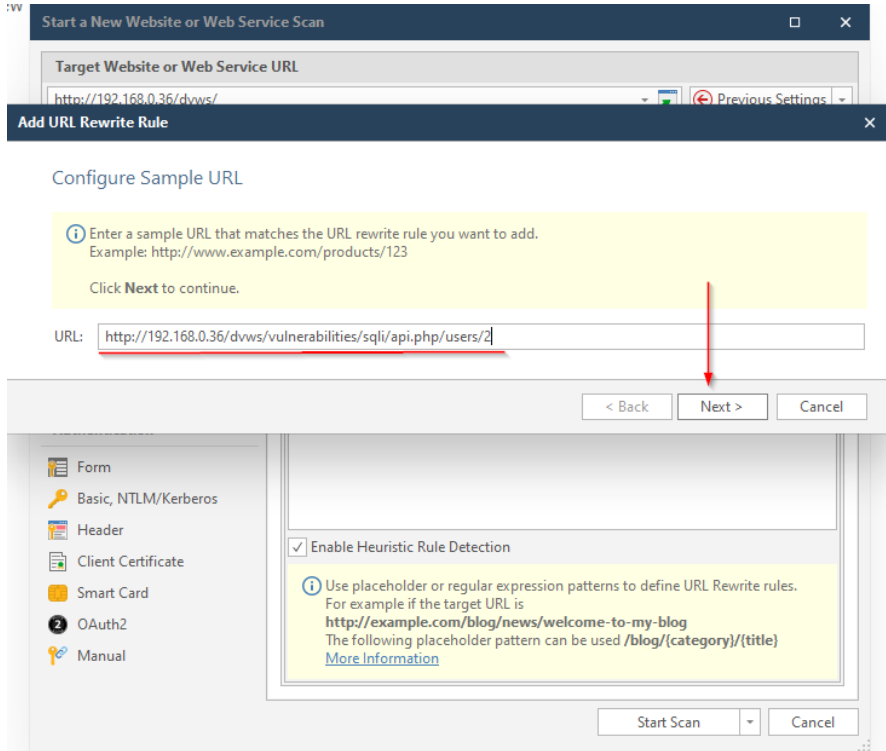
Rest web servis URL’sindeki Parametreyi kapsama dahil etmek için URL Rewrite sekmesine gidilir ve Custom seçeneğine tıklanır.



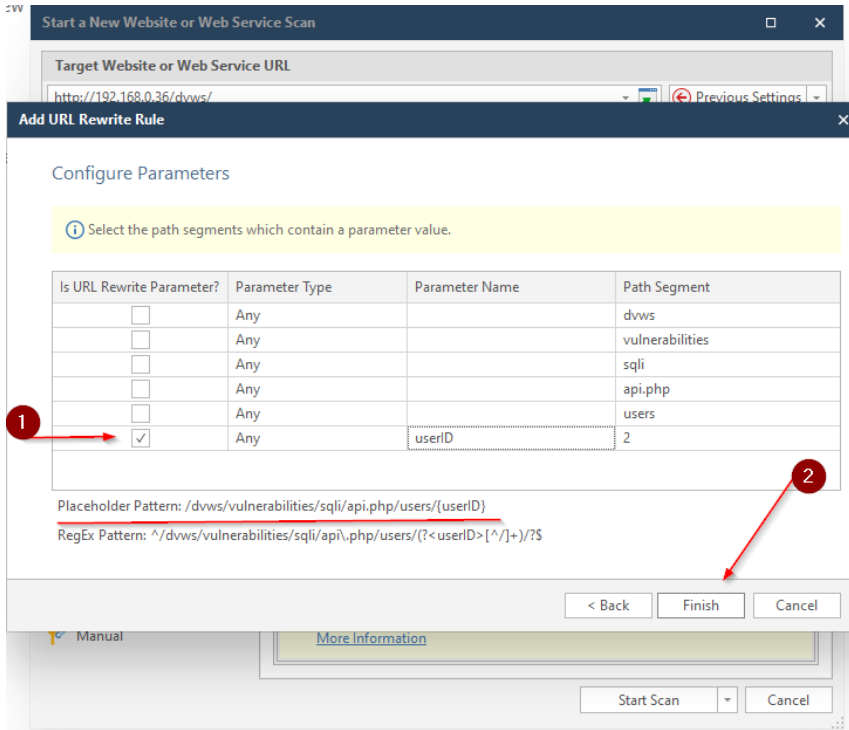
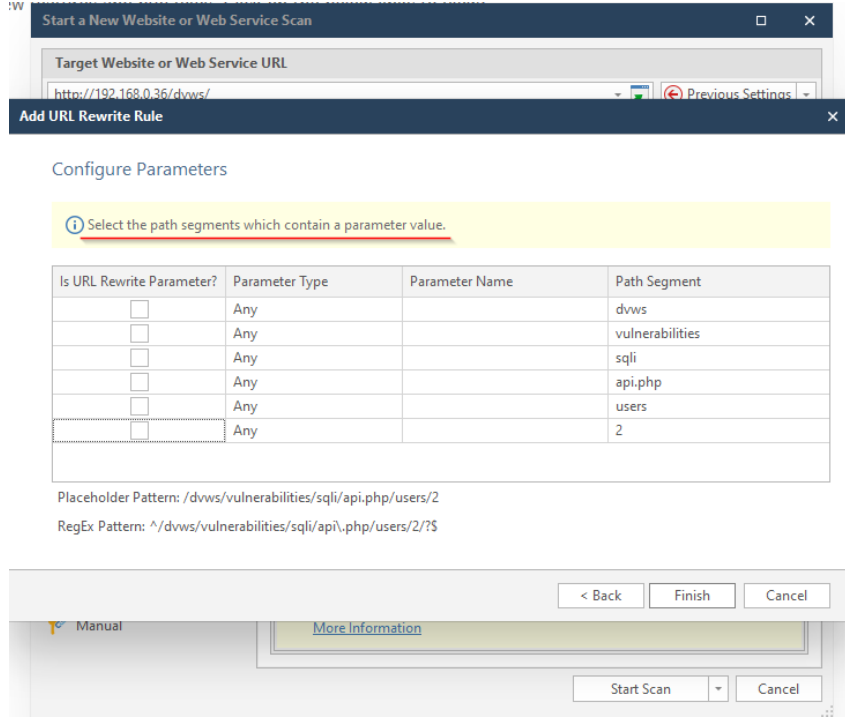
New ile kural penceresi açılır.



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.

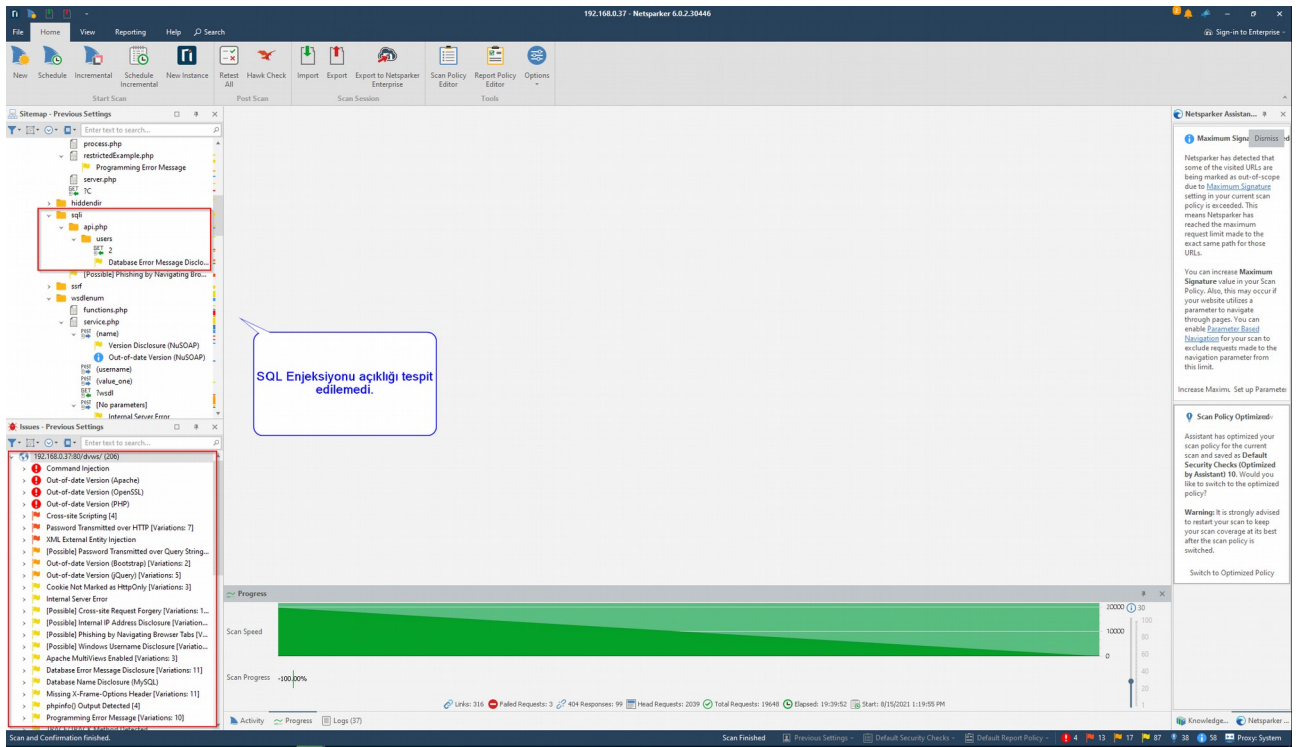
Yukarıdaki resimde birinci olarak

<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 varsayılabacağından denetlenmeyecektir ve sql injection zafiyeti tespit edilemeyecektir. Aşağıda url rewrite ile 2 parametresi işaretlemesi yapılmadan tarama yapıldığında sql enjeksiyonunun tespit edilemediği gösterilmiştir.



Uygulama [Chrome Web Tarayıcı Eklentisi İle Soap Web Servis Test Etme]

(+) Birebir denenmiştir ve başarıyla uygulanmıştır.

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

Gereksinimler

Ubuntu 18.04 LTS	// Ana Makina
Boomerang SOAP and REST Client Plugin	// Web Servis Güvenlik // Testi Chrome Eklentisi
DVWS - Windows 7 Home Premium	// Hedef Web Servisi - Sanal Makine

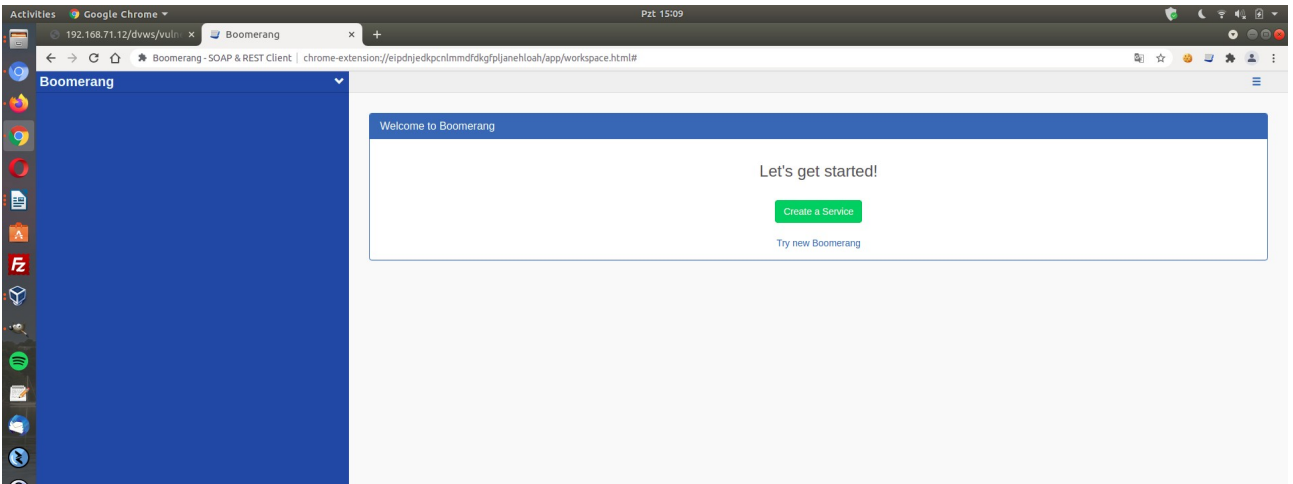
Not: Kasıtlı zafiyetler içeren DVWS web servisi outdated olduğundan sadece eski php versiyon 5.5.38'de her sayfası düzgün çalışır. Bu nedenle XAMPP php 5.5.38 kurulumu ile DVWS web servisi DVWS - Windows 7 Home Premium sanal makinesinde yayındadır.

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

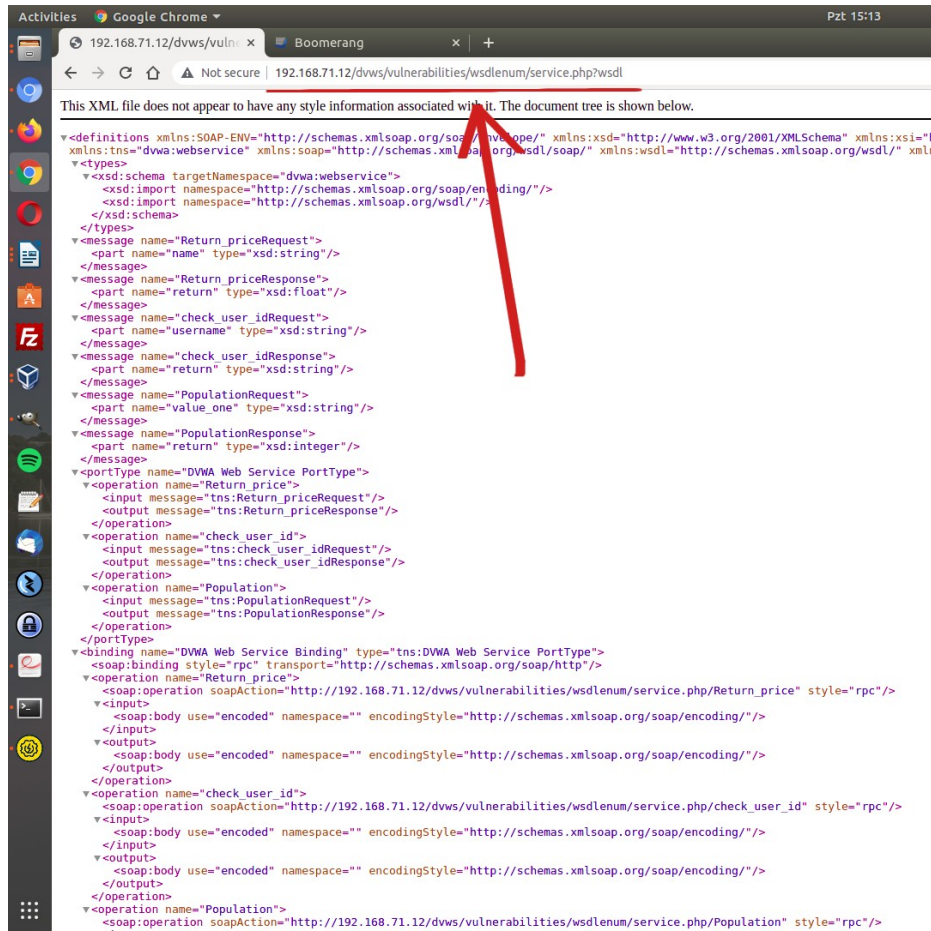
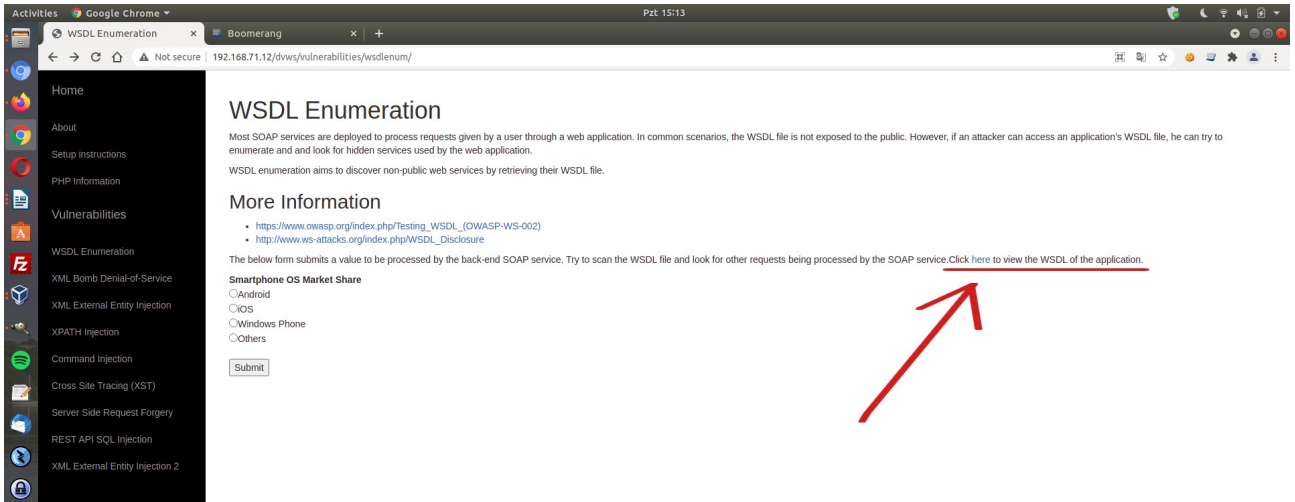
- Boomerang SOAP and REST Client (Chrome Eklentisi)

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.

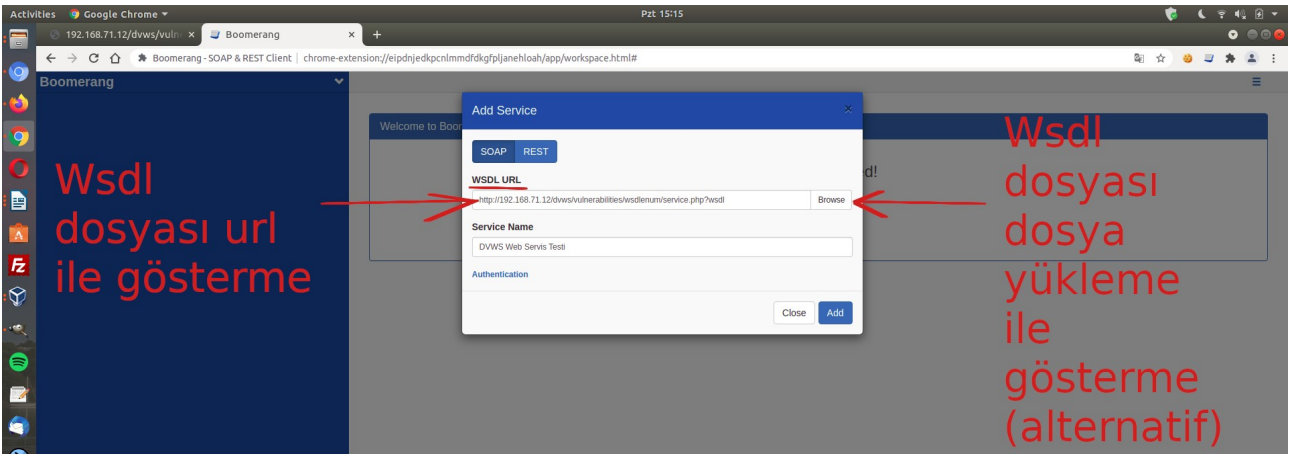
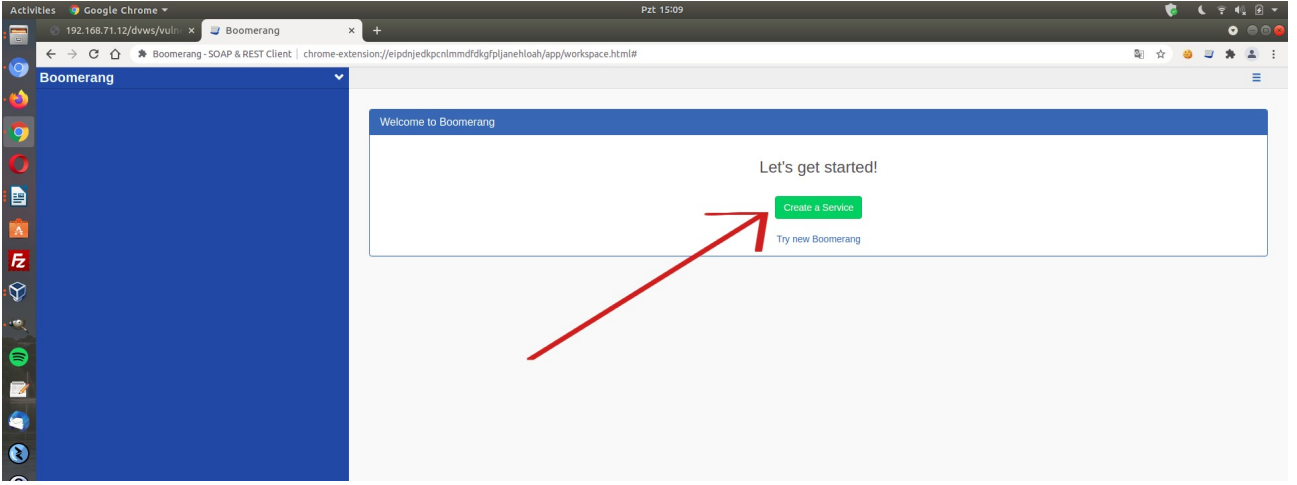
Boomerang SOAP and REST Client eklentisi chrome web tarayıcılara kurulduğunda şöyle bir arayüzle ekrana gelmektedir.



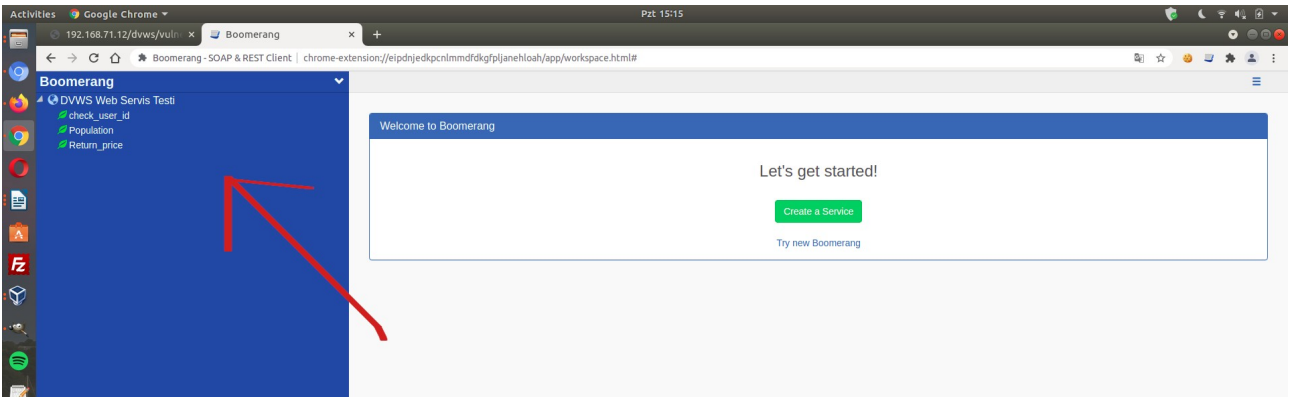
Öncelikle test edilecek DVWS soap web servisinin wsdl dosyası url'sini alalım.



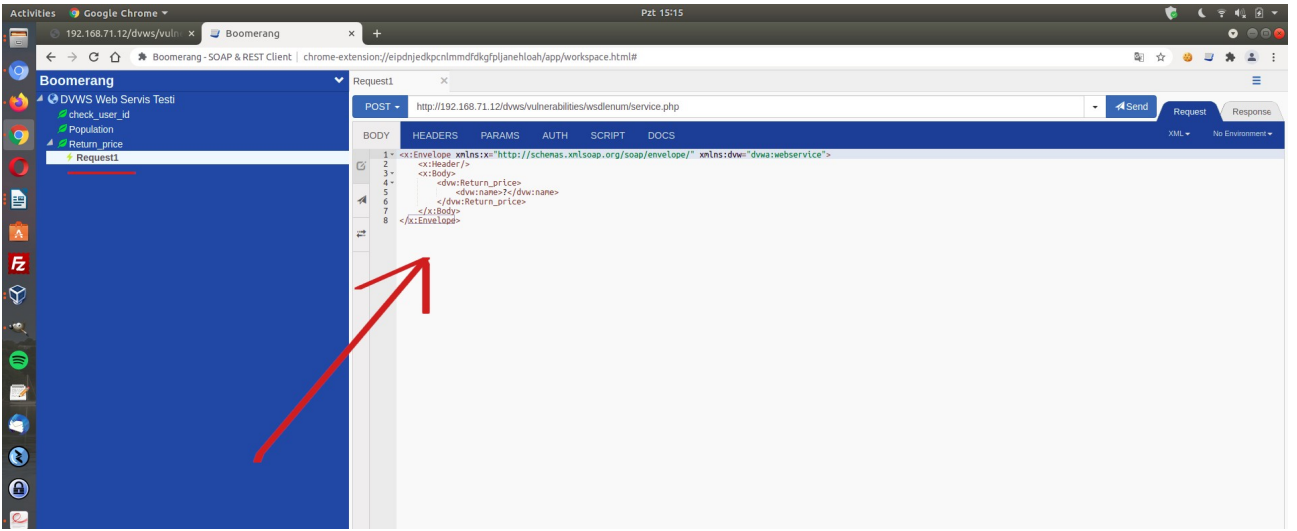
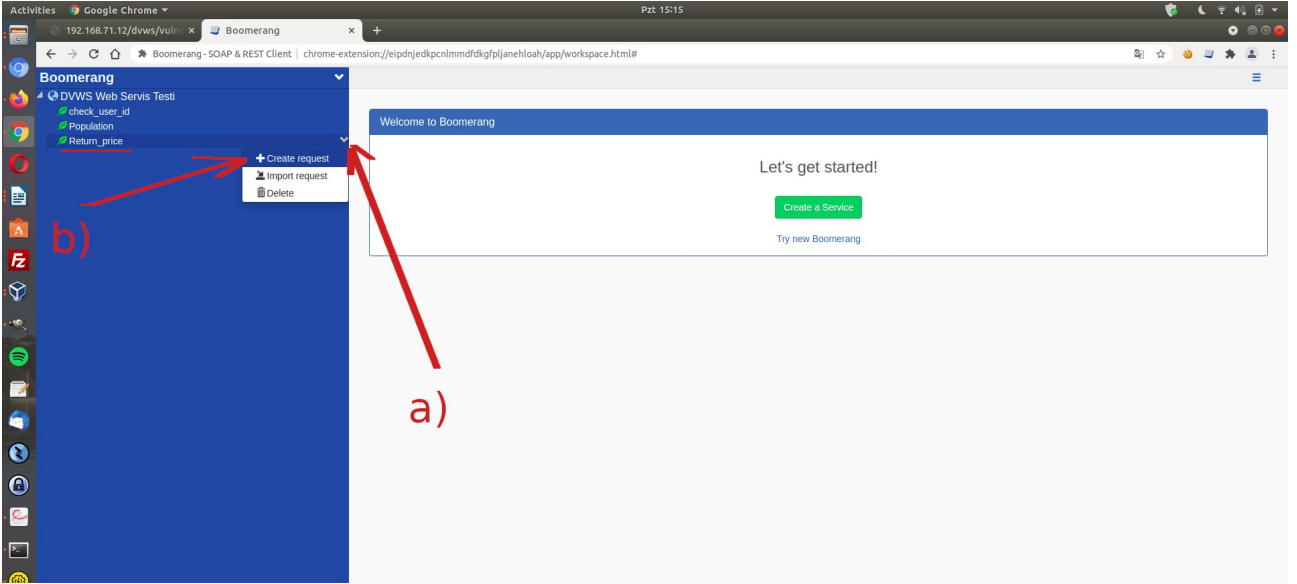
Ardından Chrome eklentisi arayüzünde Create a Service diyerek test edilecek Soap web servisinin WSDL dosyası url'sini girelim veya dosya elde mevcutsa dosya yüklemesi Browse seçeneğini kullanarak wsdl dosyasını yükleyelim.



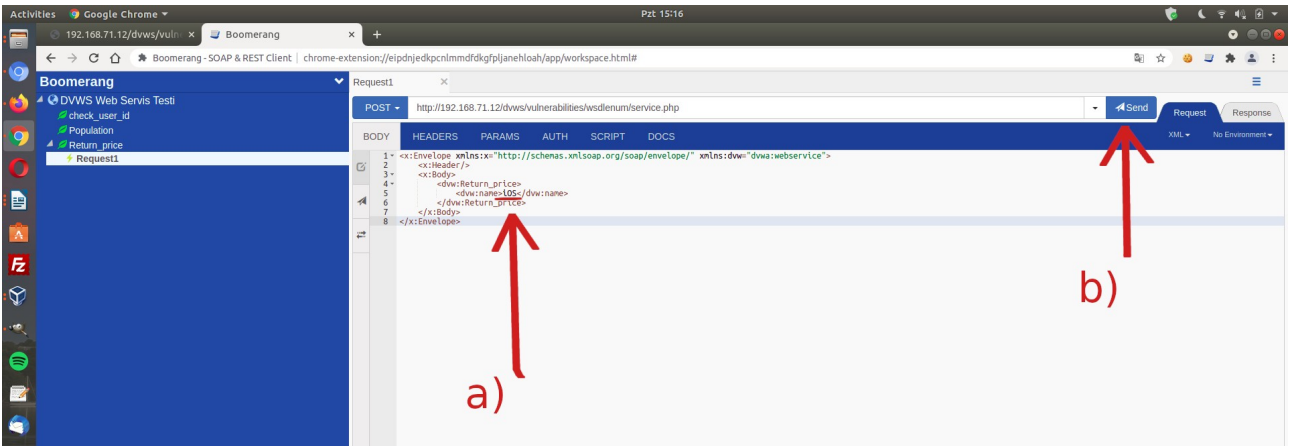
Bu şekilde hedef dvws soap web servisin arayüzü / kapsamı eklentiye (web servisi tüketen / kullanan uygulamaya) yüklenmiş olacaktır.



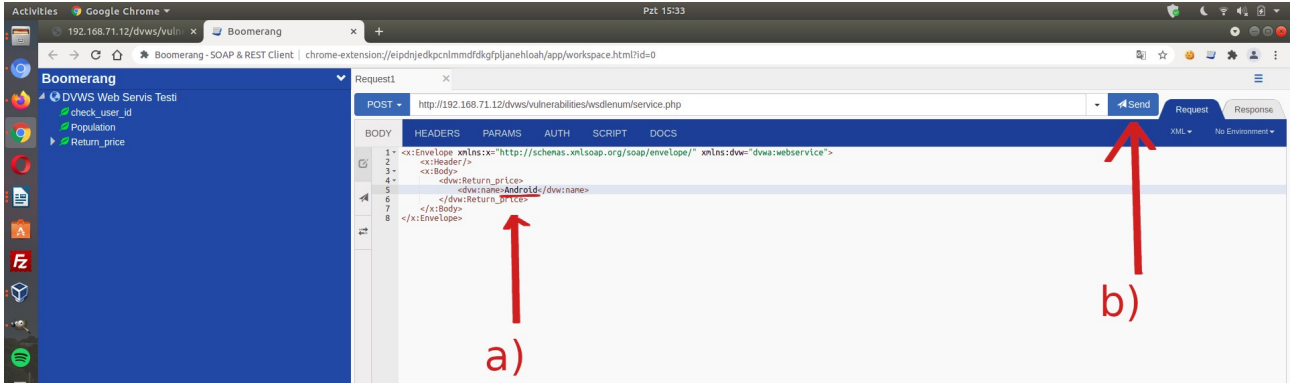
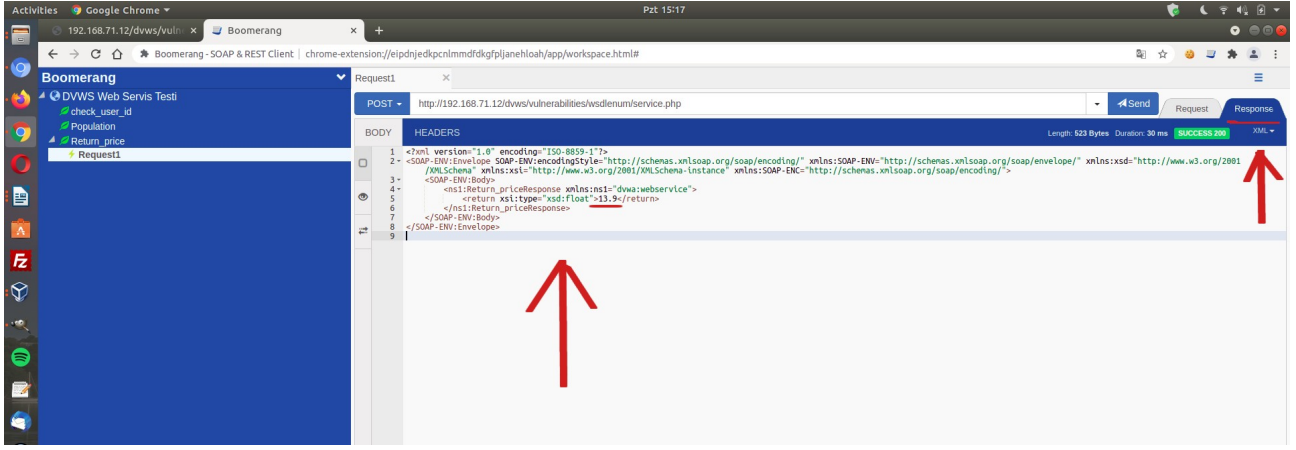
Sol sütunda hedef soap web servisin kabul ettiği parametreler sıralanacaktır. Bu parametrelerden biri için örnek bir xml talebi otomatik şekilde oluşturuyorum.



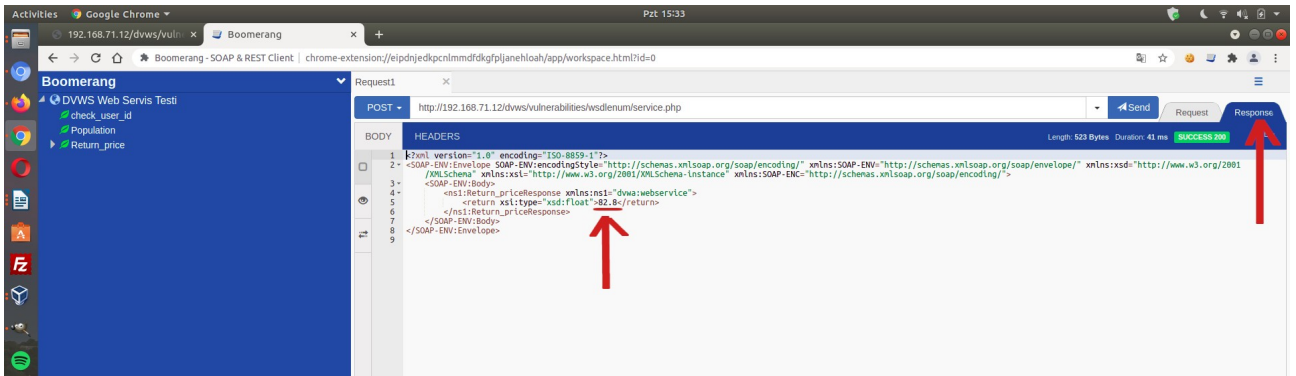
Ardından oluşan örnek xml talebini kurcalayarak hedef dvws soap web servise gönderelim.



(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'larından elde edilmiřtir).



Bu řekilde chrome eklentisi yoluyla wsdl dosyası y¼kleme ve hedef soap web servislerde g¼venlik denetimleri uygulama yapılabilir.

>>> Web Servisleri Saha Görevlerinde Test Etme Hakkında (Genel Değerlendirme)

[Netsparker]

Web servis testlerinde eğer web servis arayüzü / kapsamı dosyası - soap için wsdl ve rest için wadl, openapi (swagger),... - web servis url'si üzerinden indirilerek veya doğrudan web servis geliştiricilerinden alınarak elde edilebiliyorsa Netsparker'da kapsam ekranında import edilebilir ve tarama yapılabilir.

[SoapUI]

Web servis testlerinde eğer web servis arayüzü / kapsamı dosyası - soap için wsdl ve rest için sadece wadl - web servis url'si üzerinden indirilerek veya doğrudan web servis geliştiricilerinden alınarak elde edilebiliyorsa SoapUI yazılımında kapsam olarak yüklenebilirler ve kapsama göre oluşacak örnek xml taleplerini değiştire değiştirerek göndererek testler yürütülebilir. Ayrıca SoapUI'de yapılacak proxy ayarı ile talep paketleri burpsuite'e yönlendirilerek test genişletilebilir.

[Burpsuite]

Web servis testlerinde soap web servisler için wsdl dosyası "web servis url'si üzerinden" elde edilebiliyorsa burpsuite wsdler eklentisi ile arayüz / kapsam edinilebilir ve burpsuite üzerinden repeater, intruder, scan, ... gibi sekmelerden testler yürütülebilir.

[Chrome & Firefox Plugin'leri]

Web servis testlerinde eğer web servis arayüzü / kapsamı dosyası örneğin wsdl elde edilebilmekteyse web tarayıcı eklentileri aracılığıyla da (örn; Boomerang Soap and REST Client - Chrome Eklentisi gibi) tanımlama dosyaları parse edilebilir ve oluşan örnek xml taleplerinde oynamalar yaparak gönderme ile güvenlik denetimleri uygulanabilir.

[Wireshark]

Web servis testlerinde eğer web servis arayüzü / kapsamı dosyası mevcut değilse web servis geliştiricisinden web servisi tüketen / kullanan uygulama istenebilir. Bu şekilde uygulamayı kullanırken oluşan trafik wireshark ile okunabilir ve web servis web sunucu adresi filtrelemesine gidilerek "follow tcp stream" ile uygulamanın web servise yaptığı ham bir talep alınabilir. Bu ham talep Burpsuite'te repeater'a atılarak ve örneğin Intruder'a gönderilerek testler yürütülebilir. Aynı şekilde Netsparker'a atılarak sadece belirtilen ham paket üzerinden otomatize testler yürütülebilir.

Not:

WSDL dosyası soap web servisler geliştirilir geliştirilmez oluşan dosyalardır. Bu nedenle soap için wsdl dosyası almak garanti olabilir, fakat rest için aynı şey söz konusu değildir.

[Proxy]

Web servis testlerinde eğer web servis arayüzü / kapsamı dosyası mevcut değilse web servis geliştiricisinden web servisi tüketen / kullanan uygulama istenebilir. Bu şekilde uygulamayı kullanırken işletim sistemi /etc/hosts dosyasından ilgili hedef web servis urls'sine karşılık localhost verilerek uygulama trafiği lokalde burpsuite'e aktarılabilir ve uygulamadan gelen paketlerle web servis test edilebilir.

Sonuç

Sonuç olarak web servisler ancak arayüzleri / kapsamı çıktığında taranabilir. Web uygulamalara göre manuel yollardan arayüzü / kapsamı çıkarmak güçtür. Bu nedenle manuel yollardan gidildiğinde arayüz / kapsam ancak çıktığı kadarıyla testler yürütülebilir. Yani ancak elde edilen talepler üzerinden açıklık denemeleri testleri uygulanabilir. Örneğin web servisi tüketen / kullanan uygulamanın trafiği wireshark ile dinlenerek elde edilen talepler kapsamınca testler uygulanması gibi. Bu şekilde web servis testi yürütülebilir. Eğer web servis geliştiricisinde arayüz / kapsam dosyaları mevcutsa elde edilecek talepler bütün kapsamı vereceğinden testler daha kaliteli yürütülebilir.

Kaynaklar

<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://cheatsheetsseries.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/>