API HACKING SECRETS PART 1

In this series, I will be teaching you my methodology of hacking API. I will share all my knowledge that I learned in the past 3 years and made 1000 dollars hacking web applications and mobile API.

Let's get started...

In this series, I will tell you from the very **basics of hacking API** the secrets that were hidden from past decades. I will share my reports and other people API's reports to give you the 360-degree view of API hacking.

Before moving forward let us first know the basics of API.

Because if you want to hack something you should be familiar with ins and outs of that thing.

In our case it's the API so we should know everything about API, **how it works, where you can find certain vulnerability, etc.** Without this knowledge you can find some vulnerabilities in API's but not always, so let's get started with the basic introduction of APPLICATION PROGRAMING LANGUAGE.

WHAT IS API?

API means **Application Programming Language**.

An **API** is a set of definitions and protocols for building and integrating application software. In simple words it's a service using which two applications can talk to each other. If you need to understand more check this video on what is API? (Link: https://www.youtube.com/watch?v=BF0hvorwuWQ&t=19s)

So, let's get started with API hacking now…

There are lots of APIs, but we will focus on only web APIs, which is only two mentioned below.

- SOAP
- REST APIs

Before going ahead let quickly understand the main difference between **SOAP AND REST API.**

SOAP(Simple Object Access Protocol)

SOAP is a protocol, and it follows a strict standard to allow communication.

SOAP uses only XML for exchanging information in its message format.

SOAP user web services like WSDL

REST (REpresentational State Transfer.)

REST is an architectural style that doesn't follow any strict standard

REST is not restricted to XML and it can use anything like XML, JSON, PLAIN-TEXT etc.

REST USES URI like path

API HACKING SECRETS PART 2

Now let us take a quick look at the **WEB SERVICE COMPONENT** before we move forward and see the real hacking methods. Basically, there are three web components that are used but for web application API we will be using only two one for SOAP and other for Rest API. The documentation standard that is used for SOAP is called WSDL and the documentation standard used for REST API is WADL

WSDL -Documentation Standard for SOAP

WADL-Documentation Standard for REST

Now some of you might think instead of demonstrating some cool trick to hack API I am showing you the documentation. but later you will understand how important these two files are. So now you know about the documentation standard of both SOAP and REST API. Let's us move forward from the next writeup we will start focusing on process and techniques to find the vulnerability in API.

API HACKING SECRETS PART 3

Finding the WSDL file to extract ENDPOINTS

The most important thing in API is endpoints, most the time you will just play with the endpoint to find a vulnerability. So, in this part we will learn how to extract endpoints from the WSDL file. The first thing you need to do is you need to do recon and API recon is the easiest. You only need to find the documentation and few google search and your task is done.

RECON:

Suppose you have a target website *example.com* which you want to hack and as mentioned above we need to do find the WSDL file so we can extract endpoints from that there are more ways also to find endpoint that I will cover in later part, for now, let us only focus on finding WSDL file.

If you are lucky you will get the WSDL file just by adding "**?wsdl"** at the end of the base API. From the above example we have taken of example.com and considering that behind the website *example.com* the API service running is *api.examle.com* then you can easily find the wsdl file just by adding

https://api.example.com/api/**?wsdl**

You can also get a similar result just by doing the google search which is **www.example.com filetype:WSDL**

• site:target.com filetype:wsdl

• ext:svc inurl:wsdl

• filetype:wsdl wsdl

•Filetype: ?wsdl

• inurl:asmx?wsdl OR inurl:jws?wsdl

• inurl:_vti_bin/sites.asmx?wsdl | intitle:_vti_bin/sites.asmx?wsdl

If both the techniques fail to give you the result then you should definitely take the help of all-time favorite tool Burpsuite.

And with burp suite you need an addon WSDL wizard which will automatically find the WSDL file from the crawled URL. the link of the addon is below.

WSDL Wizard

This extension scans a target server for WSDL files. After performing normal mapping of an application's content, right…

portswigger.net

If now also you are not able to find the WSDL file there is two possibility

1. The WSDL file is not present it was removed by the web owner
2. The WSDL file is not present at that particular end-point

Apart from these two there is one more scenario that is responsible if you are not able to find the WSDL file Do you know what it is?

The website is not using SOAP API.

I will request all of my readers if you know some more techniques to find WSDL do reply in the comment section so we all can learn few more techniques to find WSDL file. Apart from that if you come across any google dork that can be helpful in finding WSDL file please share.

Video demonstration of the above techniques is available on my YouTube channel API hacking playlist

https://www.youtube.com/watch?v=Nd-cFZ_0-fU

API HACKING SECRETS PART 4

In this part, I will teach you how to extract endpoints from the WSDL file so let's get started.

So, for finding the endpoint you need a WSDL file and if you are reading this you must have read my previous parts of API hacking and i assume that you have the WSDL file. If you don't know how to find the WSDL file you can go to the API hacking secret part three and you will know how to find the WSDL file.

So, Let's get started…

For parsing the endpoint from WSDL file you need three things

WSDL file

Burp Suite

WSDLER

We know about the WSDL file and Burp suite but

What is WSDLER?

WSDLER is a burp suite extension used to parse the WSDL file, so let us first see how to install wsdler in burp suite properly.

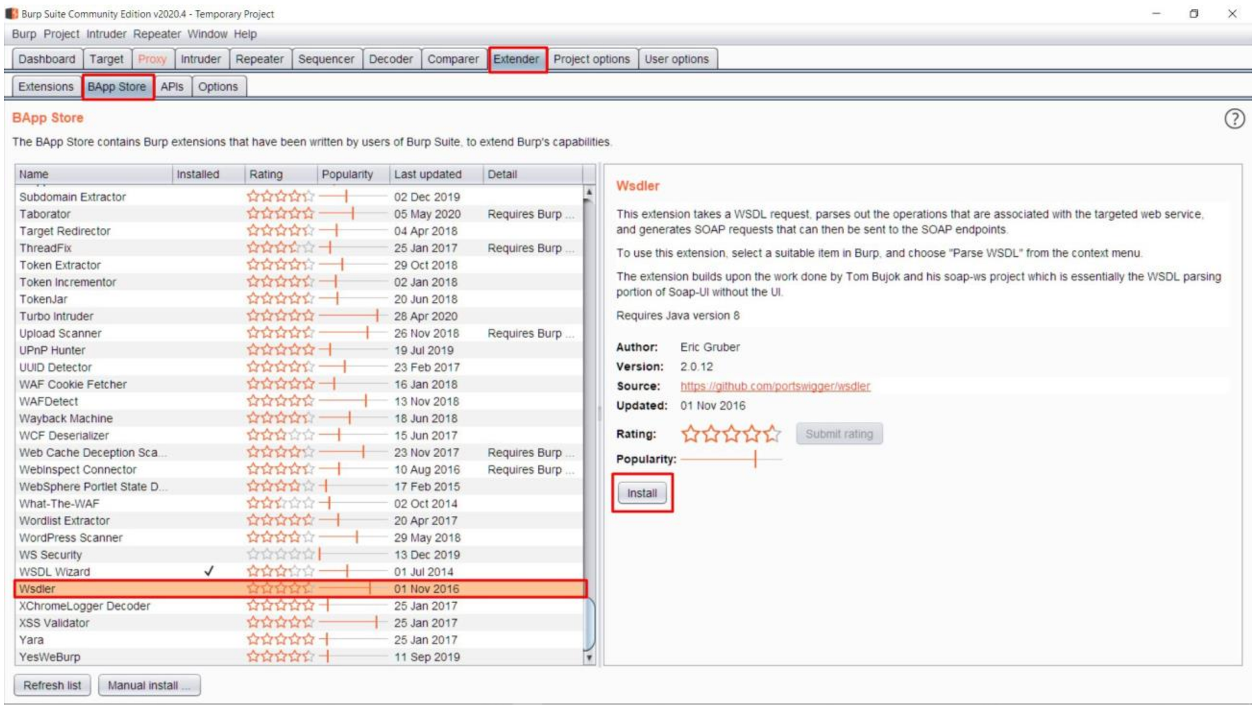
Bwapp store wsdler link

https://portswigger.net/bappstore/594a49bb233748f2bc80a9eb18a2e08f

How to install WSDLER in burp suite?

STEP 1: Fire up Burp Suite and navigate to the Extender tab and then click on the Bwapp store.

STEP 2: Find wsdler addon by scrolling down in bwapp store and click on Install button as shown in the image below.



Burp Suite Community Edition v2020.4 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard | Target | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options

Extensions | BApp Store | APIs | Options

BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Name	Installed	Rating	Popularity	Last updated	Detail
Subdomain Extractor				02 Dec 2019	
Taborator				05 May 2020	Requires Burp ...
Target Redirector				04 Apr 2018	
ThreadFix				25 Jan 2017	Requires Burp ...
Token Extractor				29 Oct 2018	
Token Incrementor				02 Jan 2018	
TokenJar				20 Jun 2018	
Turbo Intruder				28 Apr 2020	
Upload Scanner				26 Nov 2018	Requires Burp ...
UPnP Hunter				19 Jul 2019	
UUID Detector				23 Feb 2017	
WAF Cookie Fetcher				16 Jan 2018	
WAFDetect				13 Nov 2018	
Wayback Machine				18 Jun 2018	
WCF Deserializer				15 Jun 2017	
Web Cache Deception Sca...				23 Nov 2017	Requires Burp ...
WebInspect Connector				10 Aug 2016	Requires Burp ...
WebSphere Portlet State D...				17 Feb 2015	
What-The-WAF				02 Oct 2014	
Wordlist Extractor				20 Apr 2017	
WordPress Scanner				29 May 2018	
WS Security				13 Dec 2019	
WSDL Wizard	✓			01 Jul 2014	
Wsdler				01 Nov 2016	
XChromeLogger Decoder				25 Jan 2017	
XSS Validator				25 Jan 2017	
Yara				25 Jan 2017	
YesWeBurp				11 Sep 2019	

Refresh list | Manual install ...

Wsdler

This extension takes a WSDL request, parses out the operations that are associated with the targeted web service, and generates SOAP requests that can then be sent to the SOAP endpoints.

To use this extension, select a suitable item in Burp, and choose "Parse WSDL" from the context menu.

The extension builds upon the work done by Tom Bujok and his soap-ws project which is essentially the WSDL parsing portion of Soap-UI without the UI.

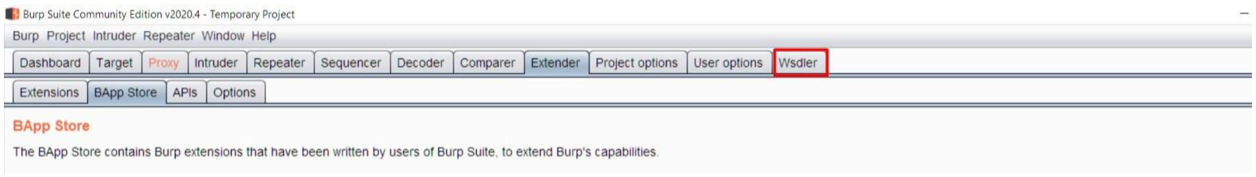
Requires Java version 8

Author: Eric Gruber
Version: 2.0.12
Source: https://github.com/portswigger/wsdler
Updated: 01 Nov 2016
Rating: Submit rating
Popularity:

Install

It will take up to 1 min depending upon your internet speed to download and install it into your Burp Suite.

Post-installation you will see one more tab named WSDLER as shown in the image below.



Burp Suite Community Edition v2020.4 - Temporary Project

Burp Project Intruder Repeater Window Help

Dashboard | Target | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Wsdler

Extensions | BApp Store | APIs | Options

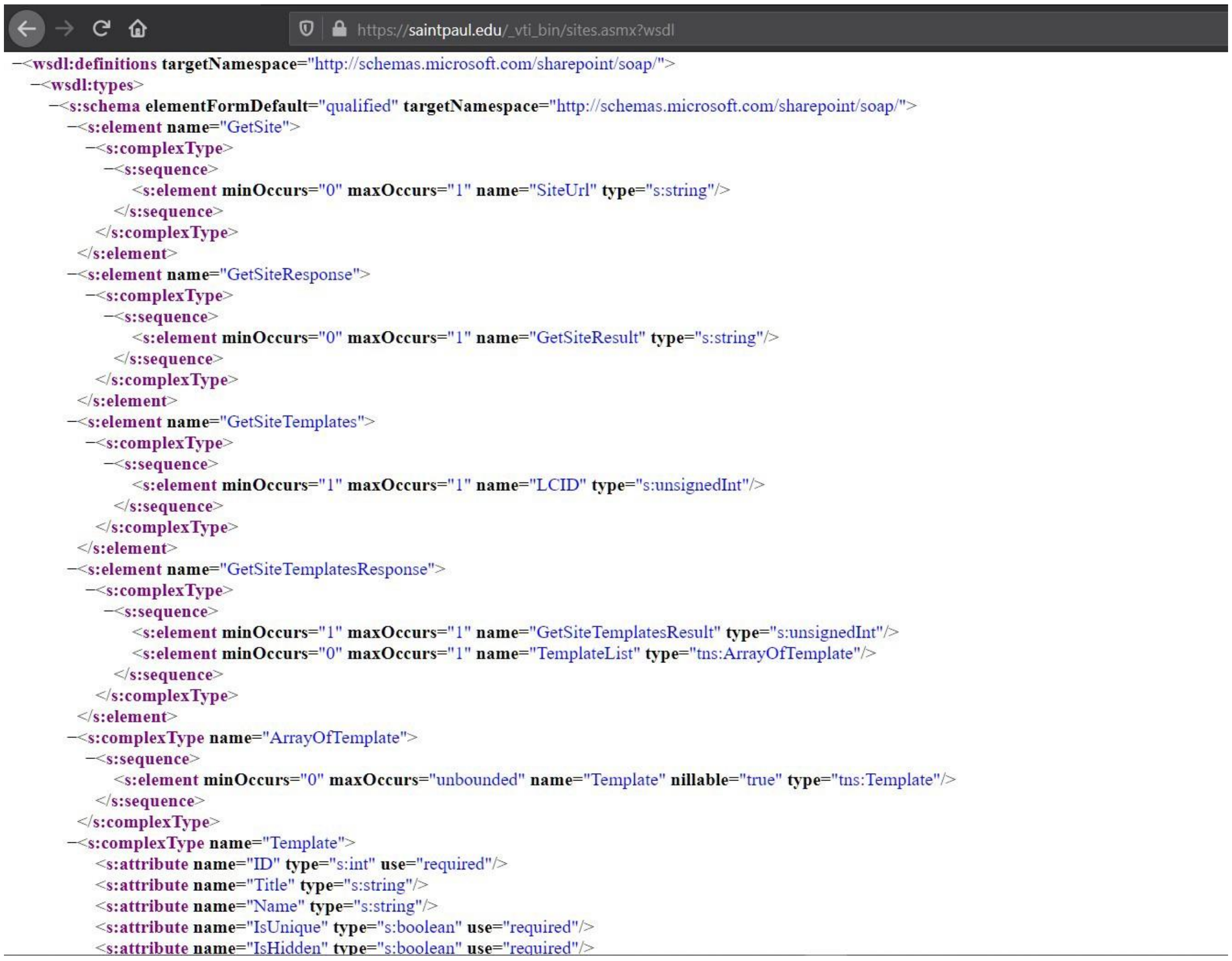
BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Now we have the required tool to parse the END-POINT let us now move forward and see how to extract endpoint from the WSDL file.

FINDING THE ENDPOINTS

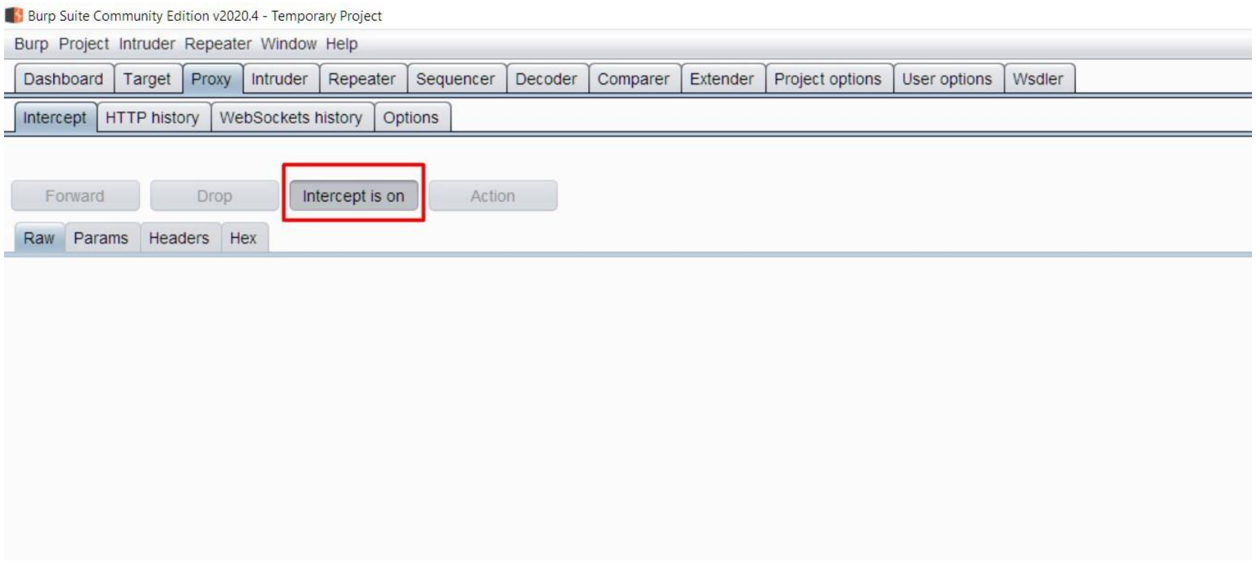
STEP 1: Find the base URI where the WSDL file is stored. As shown in the below image.



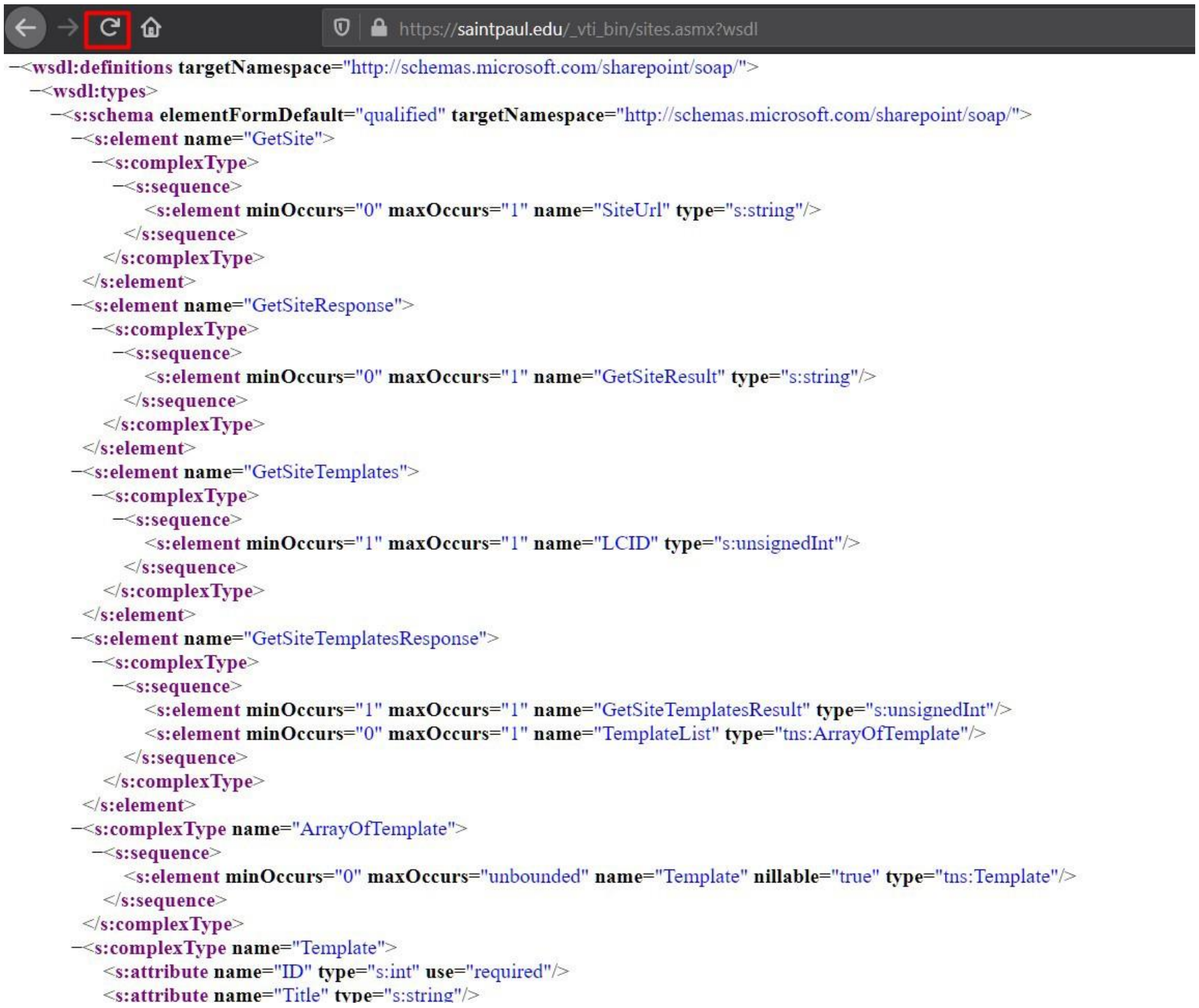
https://saintpaul.edu/_vti_bin/sites.asmx?wsdl

```
−<wsdl:definitions targetNamespace="http://schemas.microsoft.com/sharepoint/soap/">
  −<wsdl:types>
    −<s:schema elementFormDefault="qualified" targetNamespace="http://schemas.microsoft.com/sharepoint/soap/">
      −<s:element name="GetSite">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="SiteUrl" type="s:string"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:element name="GetSiteResponse">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="GetSiteResult" type="s:string"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:element name="GetSiteTemplates">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="LCID" type="s:unsignedInt"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:element name="GetSiteTemplatesResponse">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="GetSiteTemplatesResult" type="s:unsignedInt"/>
              <s:element minOccurs="0" maxOccurs="1" name="TemplateList" type="tns:ArrayOfTemplate"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:complexType name="ArrayOfTemplate">
        −<s:sequence>
            <s:element minOccurs="0" maxOccurs="unbounded" name="Template" nillable="true" type="tns:Template"/>
          </s:sequence>
        </s:complexType>
      −<s:complexType name="Template">
          <s:attribute name="ID" type="s:int" use="required"/>
          <s:attribute name="Title" type="s:string"/>
          <s:attribute name="Name" type="s:string"/>
          <s:attribute name="IsUnique" type="s:boolean" use="required"/>
          <s:attribute name="IsHidden" type="s:boolean" use="required"/>
```

STEP 2: Now once you have the base URI let us now proxy it through our burp suite just by turning our proxy on in burp and refresh the browser. As shown in the below image:


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Wsdler
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Params Headers Hex

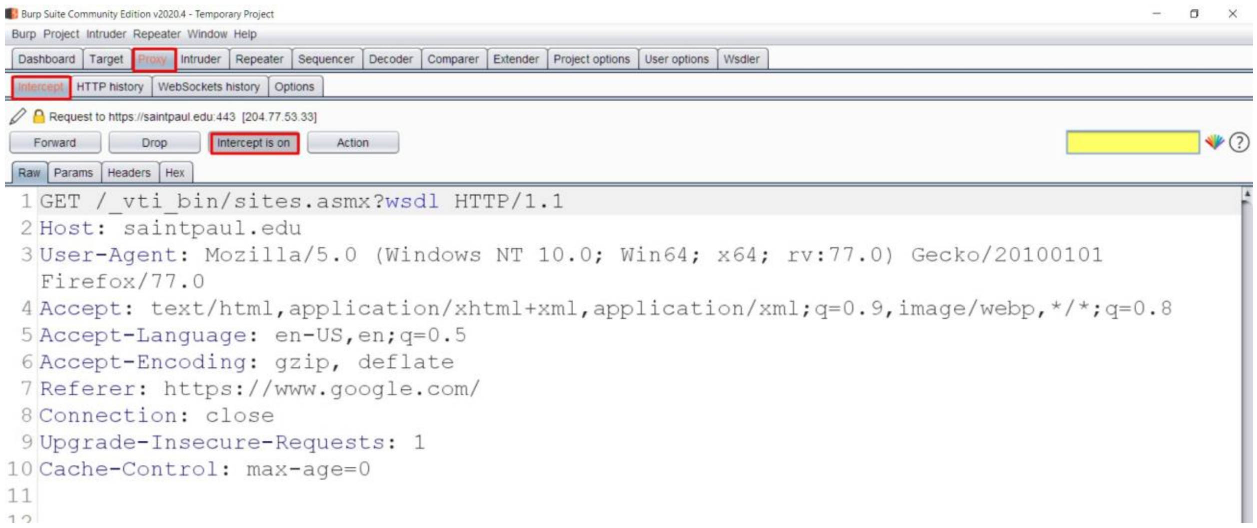
The first thing to intercept request you must turn on the proxy by clicking on Intercept is on the button in the proxy tab. Now we need to go to our browser and click on the refresh. As shown in below image:



https://saintpaul.edu/_vti_bin/sites.asmx?wsdl

```
−<wsdl:definitions targetNamespace="http://schemas.microsoft.com/sharepoint/soap/">
  −<wsdl:types>
    −<s:schema elementFormDefault="qualified" targetNamespace="http://schemas.microsoft.com/sharepoint/soap/">
      −<s:element name="GetSite">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="SiteUrl" type="s:string"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:element name="GetSiteResponse">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="GetSiteResult" type="s:string"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:element name="GetSiteTemplates">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="LCID" type="s:unsignedInt"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:element name="GetSiteTemplatesResponse">
        −<s:complexType>
          −<s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="GetSiteTemplatesResult" type="s:unsignedInt"/>
              <s:element minOccurs="0" maxOccurs="1" name="TemplateList" type="tns:ArrayOfTemplate"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      −<s:complexType name="ArrayOfTemplate">
        −<s:sequence>
            <s:element minOccurs="0" maxOccurs="unbounded" name="Template" nillable="true" type="tns:Template"/>
          </s:sequence>
        </s:complexType>
      −<s:complexType name="Template">
          <s:attribute name="ID" type="s:int" use="required"/>
          <s:attribute name="Title" type="s:string"/>
```

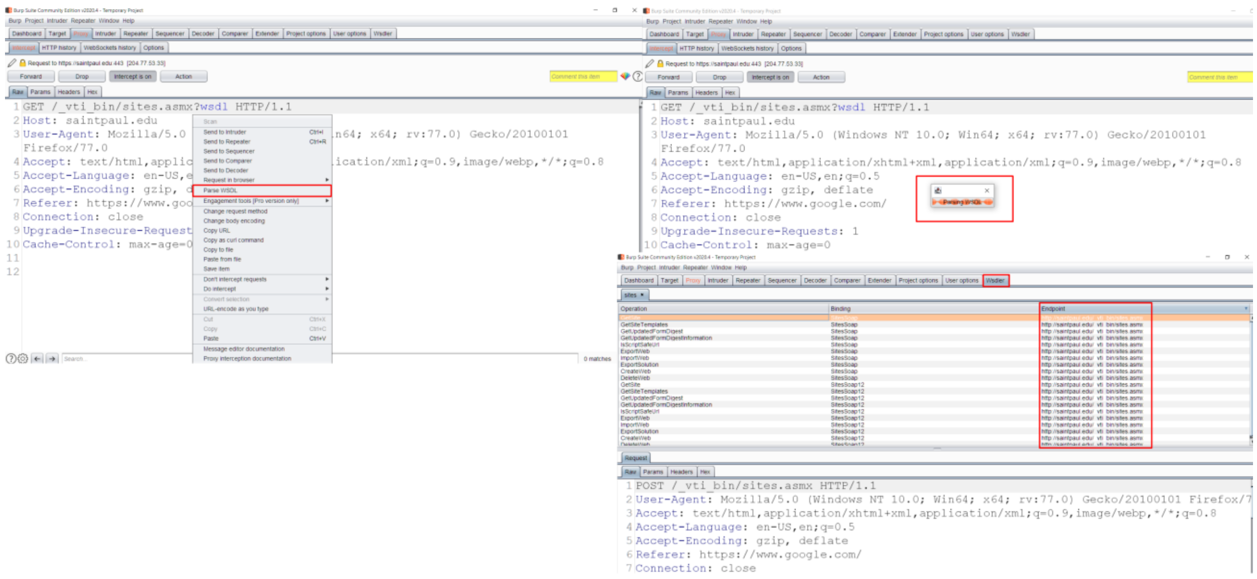
Now As soon as you will click on refresh burp will intercept the request and it will prompt up like as shown in the below image:


Burp Suite Community Edition v2020.4 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Wsdler
Intercept HTTP history WebSockets history Options
Request to https://saintpaul.edu:443 [204.77.53.33]
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
GET /_vti_bin/sites.asmx?wsdl HTTP/1.1
Host: saintpaul.edu
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101
Firefox/77.0
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: https://www.google.com/
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Now once you see this screen in burp you done everything perfectly now let us see how to extract the endpoints.

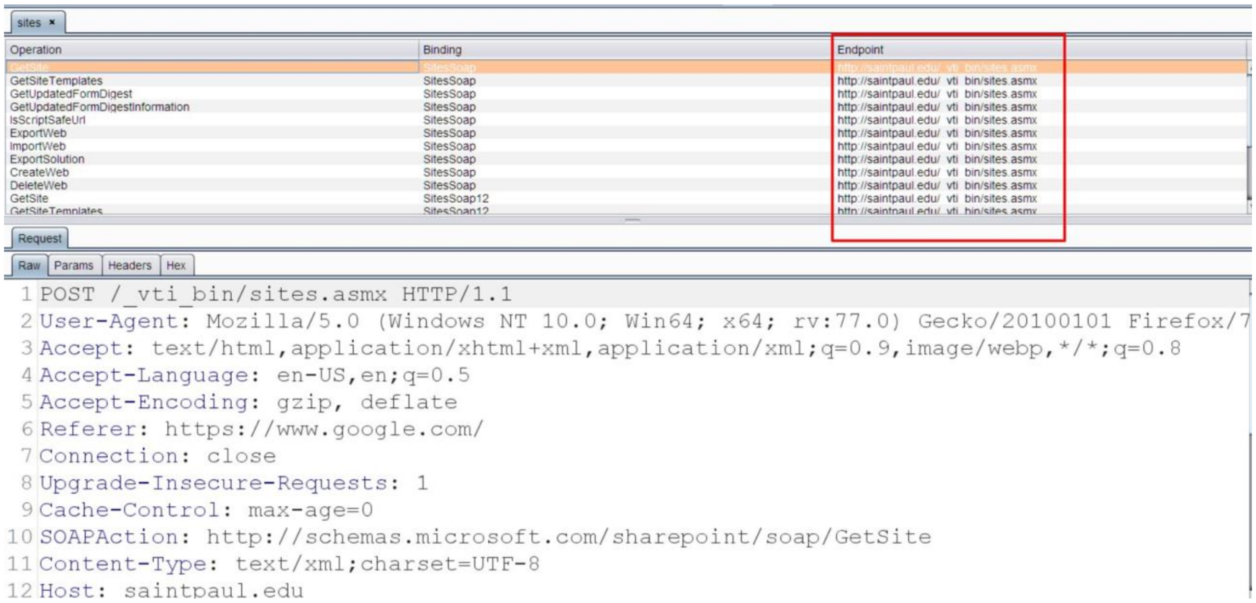
So, for extracting the endpoint you will right click your mouse on the proxied request as shown in the below image. and then click on parse wsdl it will take a while processing



Operation	Binding	Endpoint
GetSite	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetSiteTemplates	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetUpdatedFormDigest	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetUpdatedFormDigestInformation	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
IsScriptSafeUrl	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
ExportWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
ImportWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
ExportSolution	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
CreateWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
DeleteWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetSite	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
GetSiteTemplates	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
GetUpdatedFormDigest	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
GetUpdatedFormDigestInformation	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
IsScriptSafeUrl	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
ExportWeb	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
ImportWeb	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
ExportSolution	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
CreateWeb	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
DeleteWeb	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx

```
POST /_vti_bin/sites.asmx HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/7
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: https://www.google.com/
Connection: close
```

Upon the parse is complete you will be greeted with the endpoints. As shown in the below image:



Operation	Binding	Endpoint
GetSite	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetSiteTemplates	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetUpdatedFormDigest	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetUpdatedFormDigestInformation	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
IsScriptSafeUrl	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
ExportWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
ImportWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
ExportSolution	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
CreateWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
DeleteWeb	SitesSoap	http://saintpaul.edu/_vti_bin/sites.asmx
GetSite	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx
GetSiteTemplates	SitesSoap12	http://saintpaul.edu/_vti_bin/sites.asmx

```
POST /_vti_bin/sites.asmx HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/7
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: https://www.google.com/
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
SOAPAction: http://schemas.microsoft.com/sharepoint/soap/GetSite
Content-Type: text/xml;charset=UTF-8
Host: saintpaul.edu
```

So, now you can see you have the endpoints.

In this post, I want to stick to How to parse the WSDL file and how to extract endpoints in the next writeup I will take you deep inside endpoint.

If still you face some issue installing and parsing endpoint visit my YouTube channel and in API hacking playlist you will find the video for parsing the endpoint:

https://www.youtube.com/watch?v=fQKQ2DMFelg

Some people ask me to skip basic things, but I want to cover everything as most of my readers are new in bug bounty and hacking and as I remember one line from stock - sharing is caring.