iGNITE Technologies

Burp Suite for Pentester

Collaborator

TABLE OF CONTENTS

Abstract

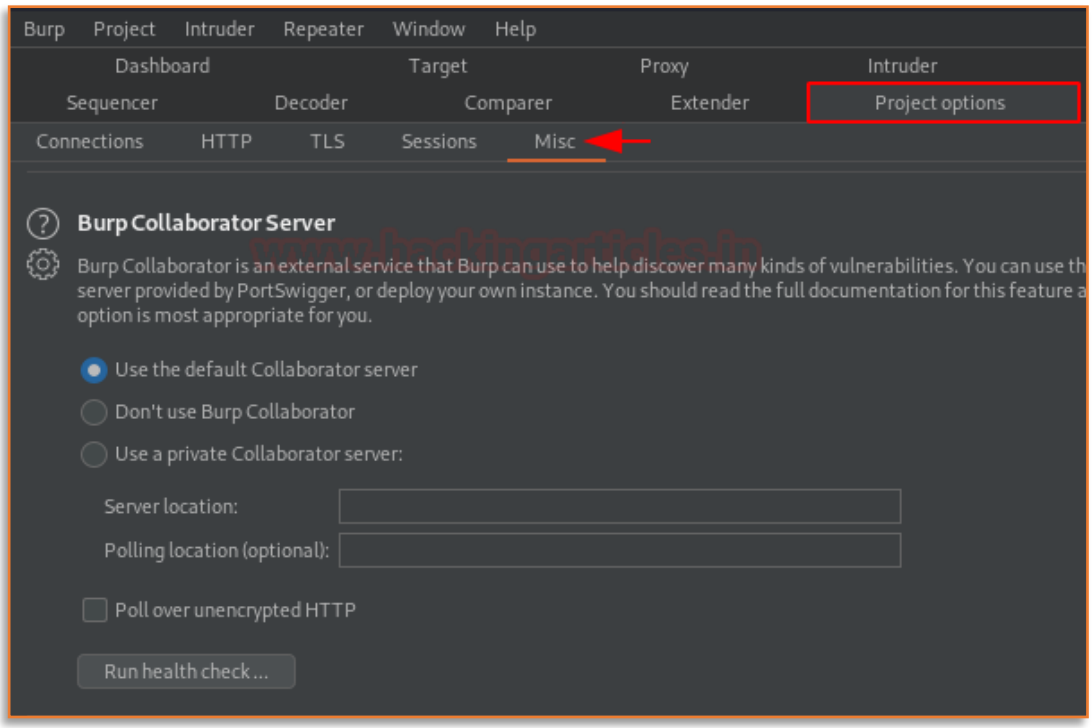
A number of vulnerabilities exist over the web, but the majority of them are not triggered directly as they do not reproduce any specific output or an error. So, is the output or the error is the only solution to determine that the vulnerability exist or not?
So, today in this publication of the series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

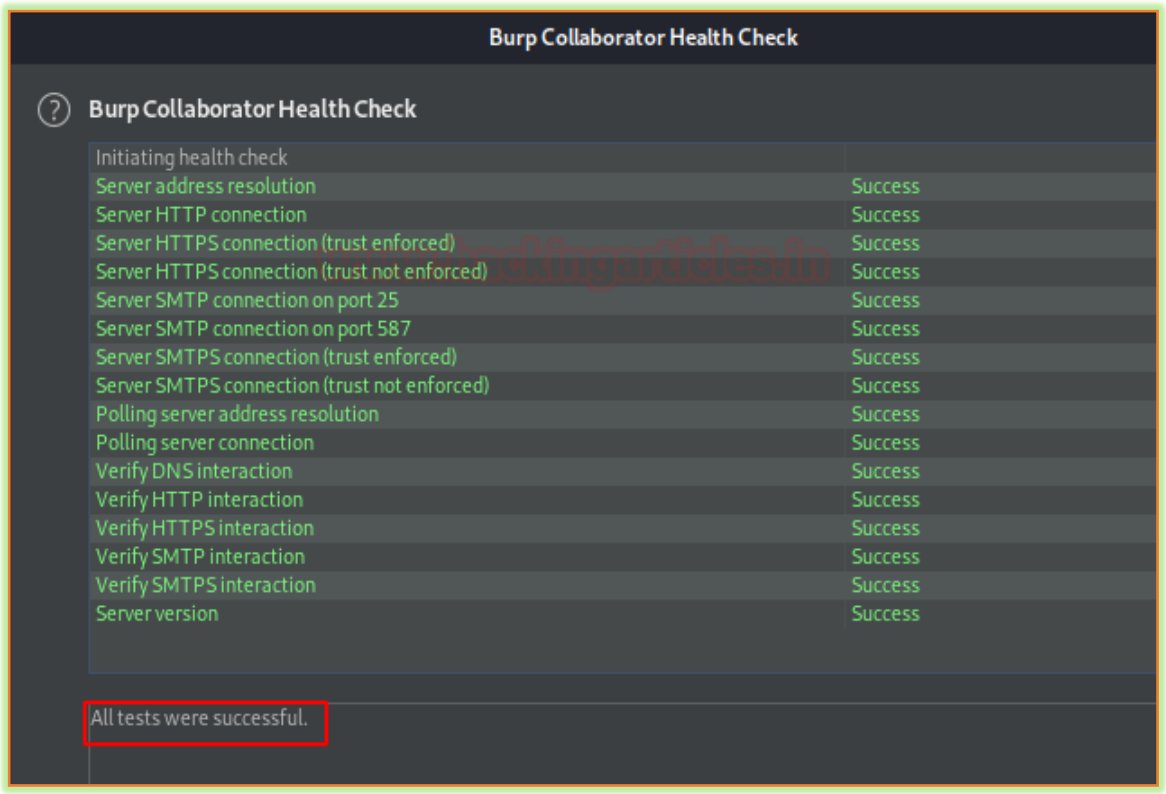
Burp Suite's Professional edition offers one of its best features as **"Burp Collaborator**" to determine or detect vulnerabilities that try to interact with external services but do not cause any difference in the content of the application's responses when specific payloads are injected.
However, in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

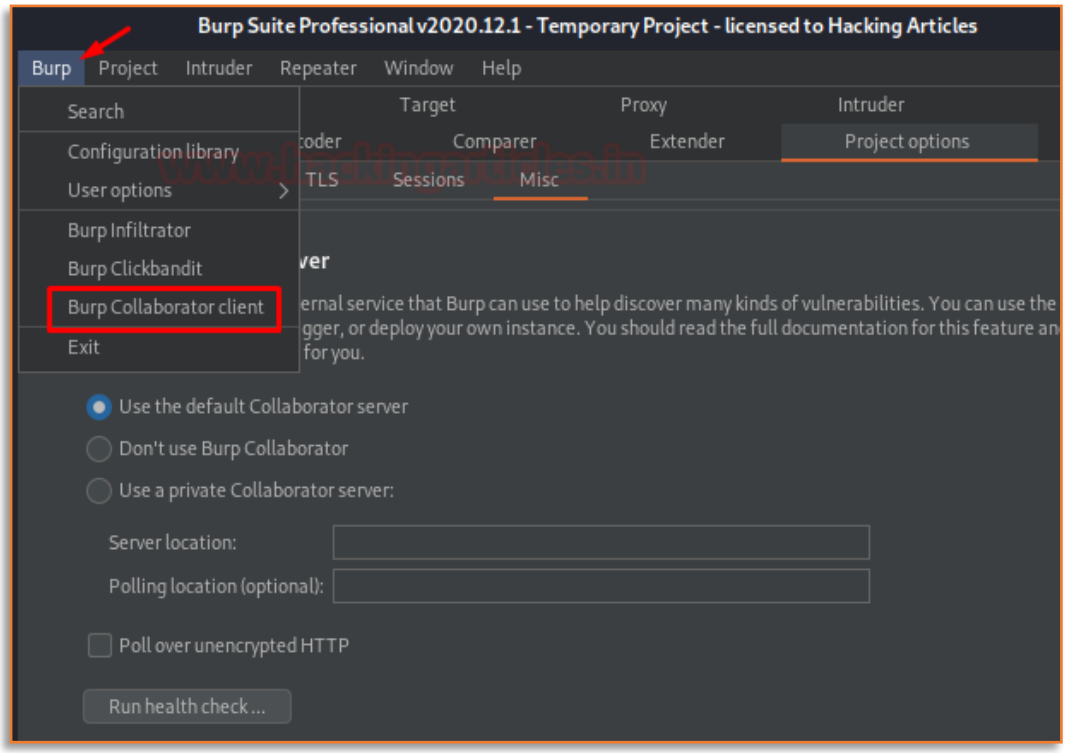


However, for this section, we'll be using the default collaborator server. But before following up further, let's do a **health check** by hitting the **Run heath check** button in order to be sure that our collaborator is working properly.
Great from the below image we can see that our collaborator server had passed all the tests.



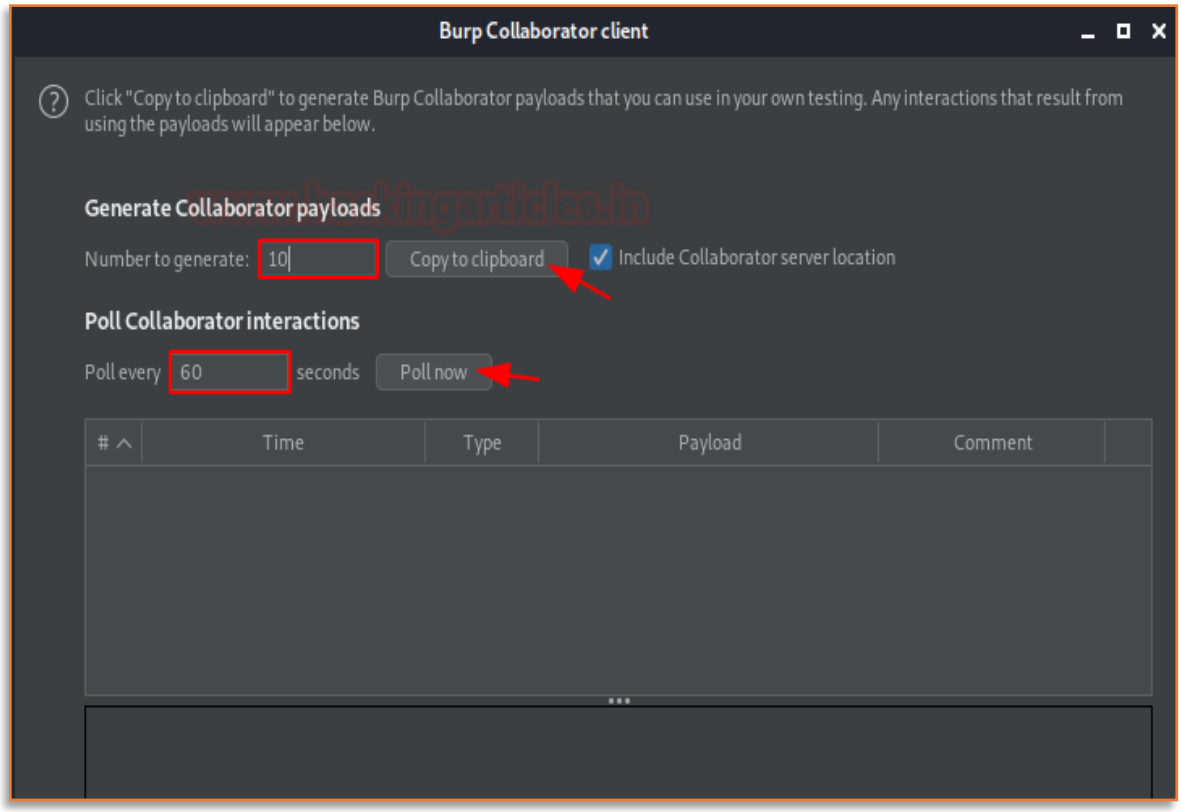
Burp Collaborator Client

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.

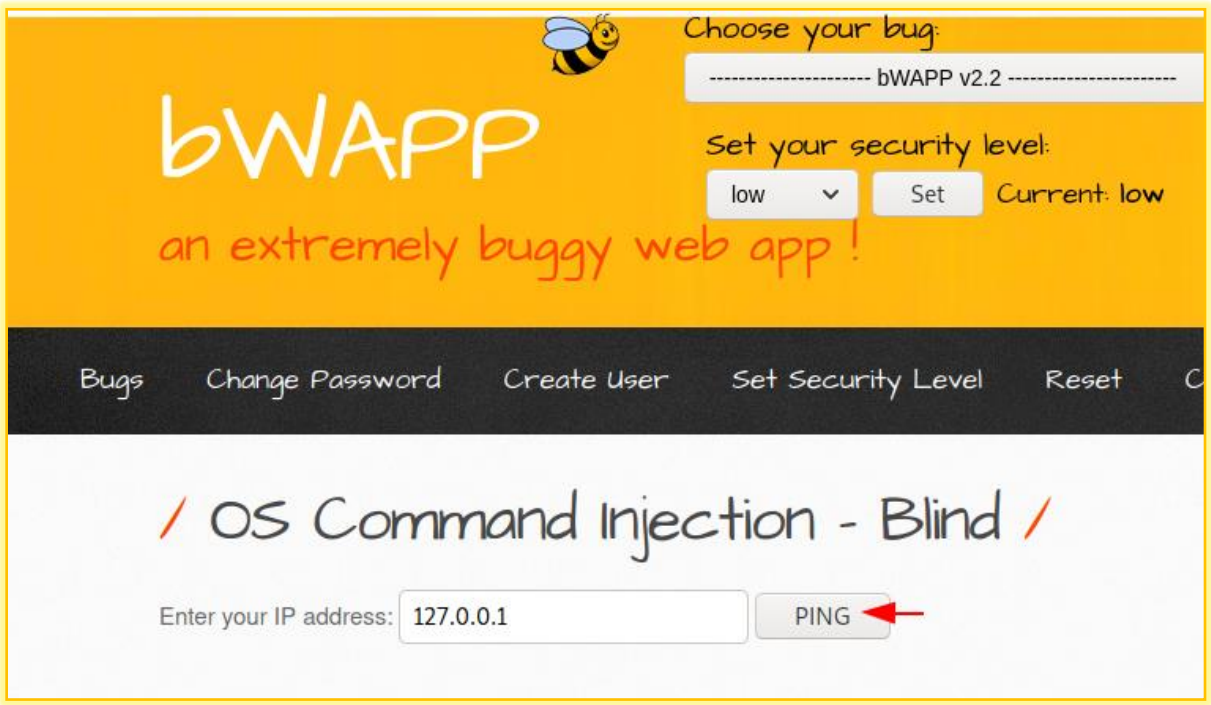


Detecting vulnerabilities with Collaborator Client

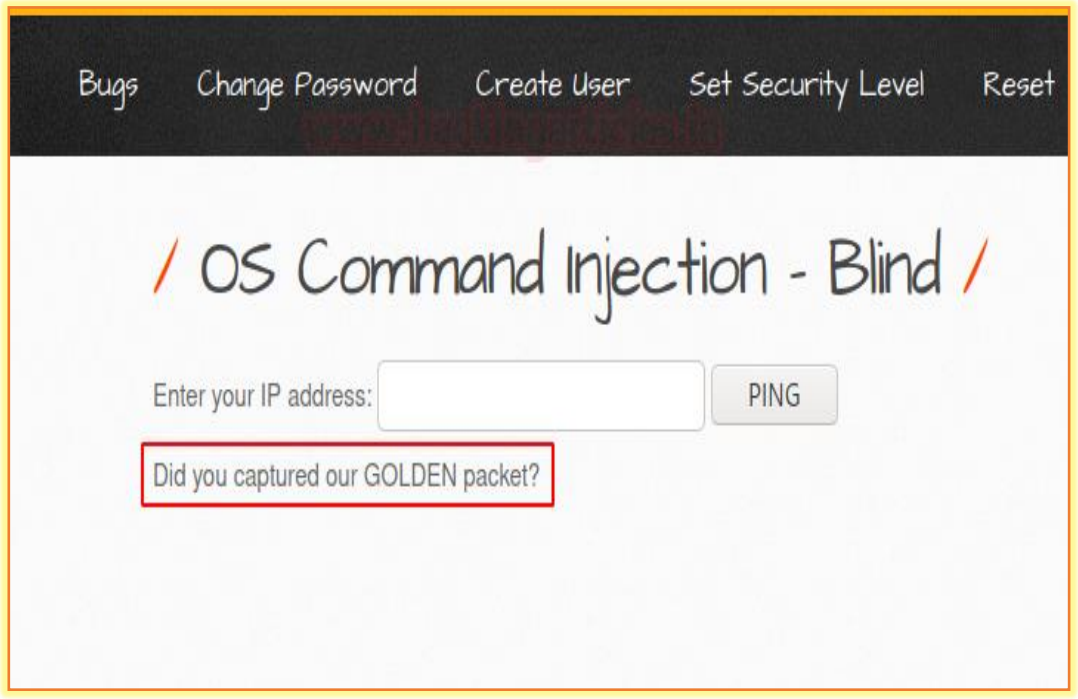
Up till now, you might be having a clear vision about what is Burp Collaborator and its client, and where to find them. So, let's use this amazing feature to have some practical exposure by detecting and finding the most crucial out-of-band vulnerabilities.

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.



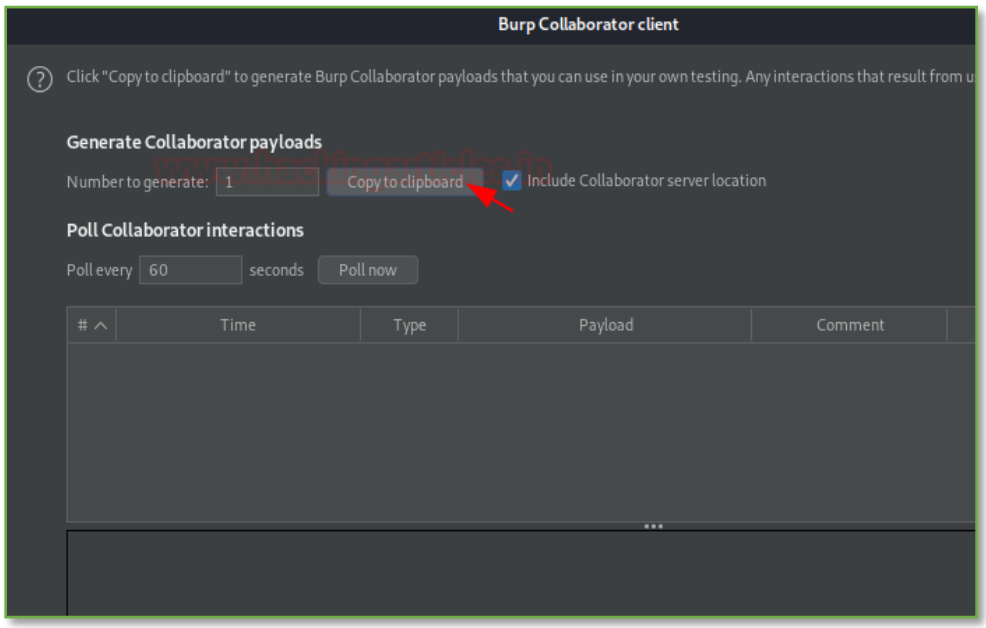
Over there, as we tried to ping our local machine with **"127.0.0.1"**, we didn't get any output neither any error, just we got a message **"Did you captured our GOLDEN packet?".**



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

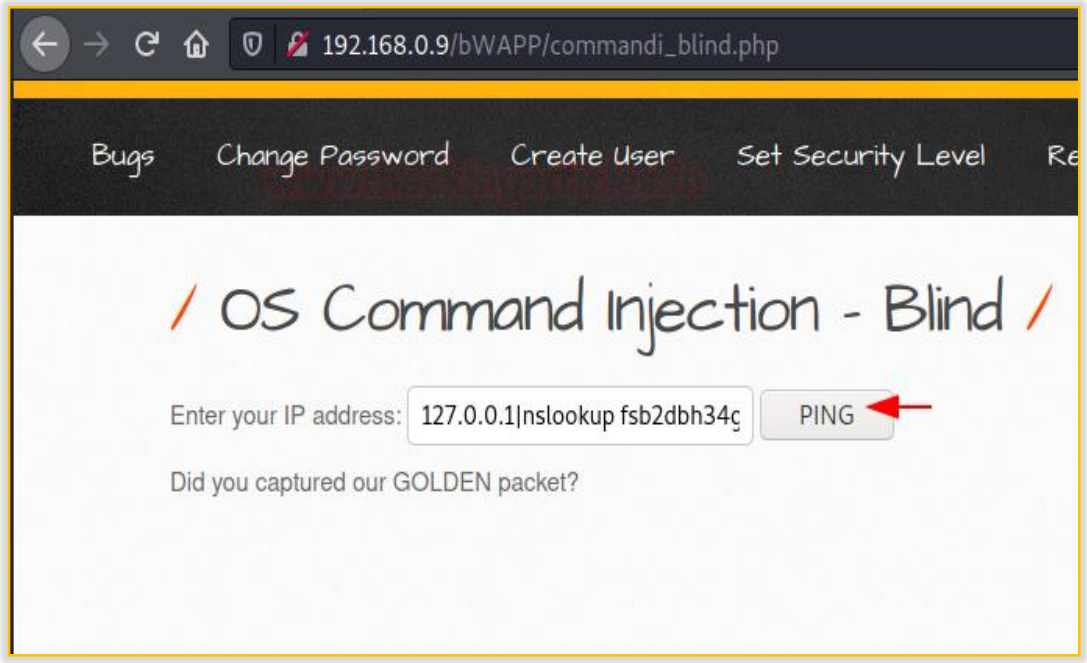
Thereby in order to confirm this, we'll be using the Burp collaborator, which will pop out the response if the application somewhere or the other tries to interact with any other external service (DNS; HTTP; SMTP).

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

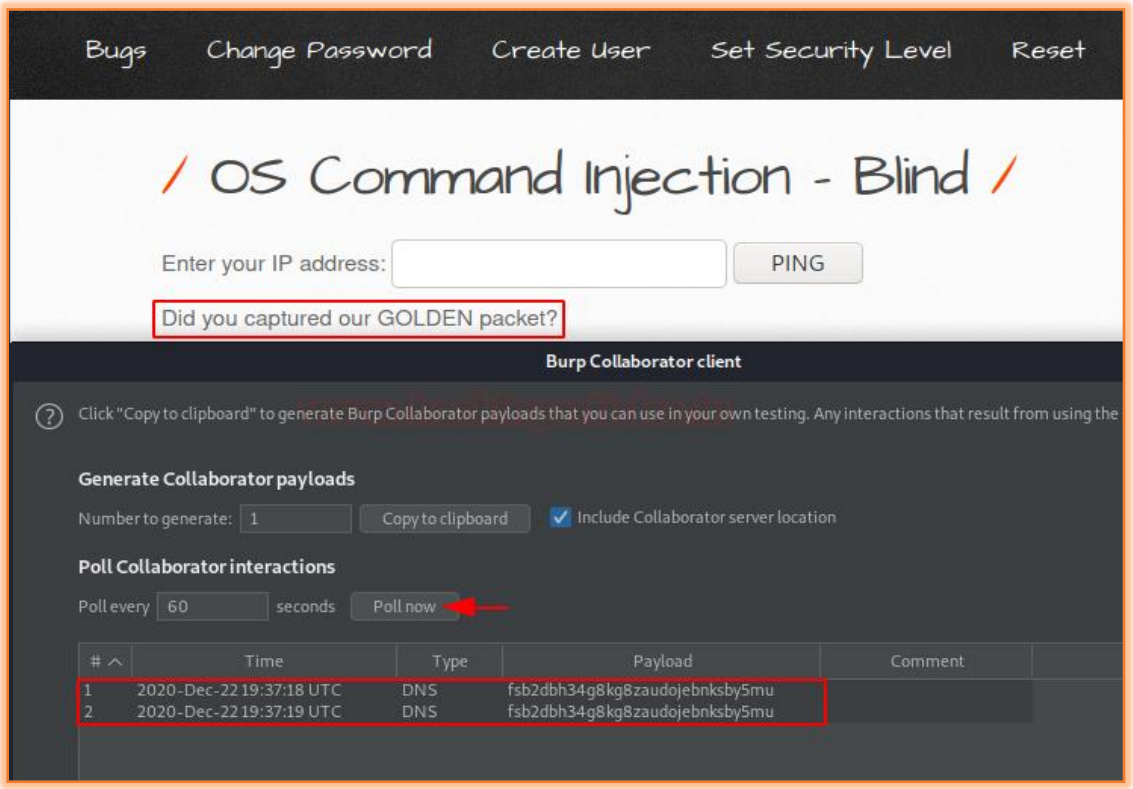


Now this time we'll modify the input value, i.e., along with the IP we'll inject the collaborator payload too and will hit the **"Ping"** button.

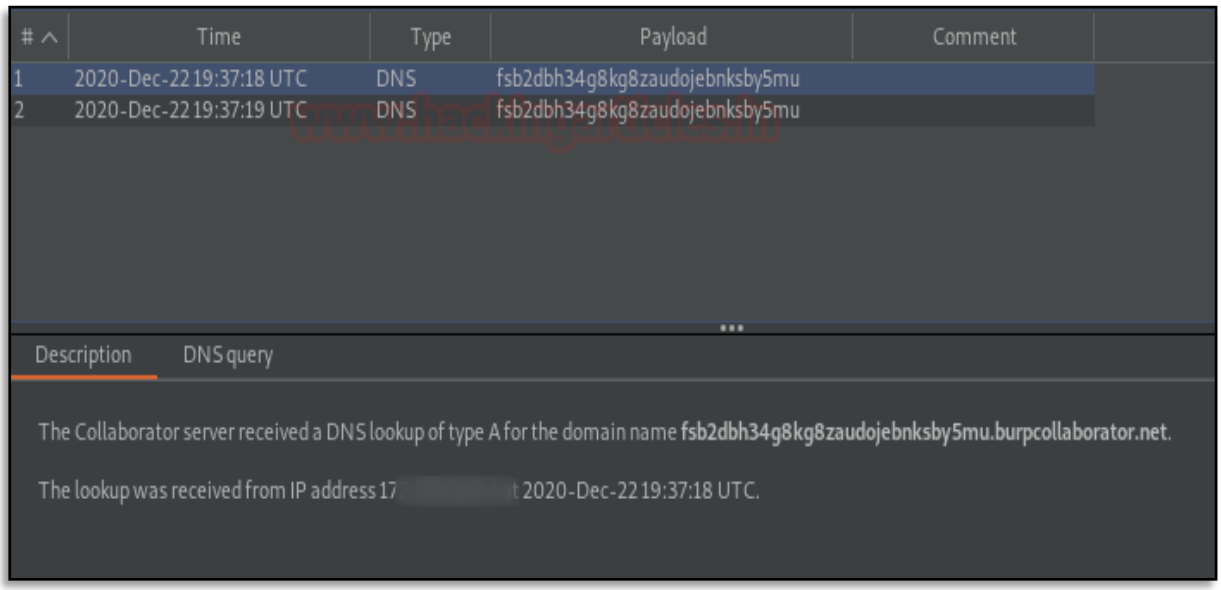
```
127.0.0.1|nslookup <Burp Collaborator copied payload>
```



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.

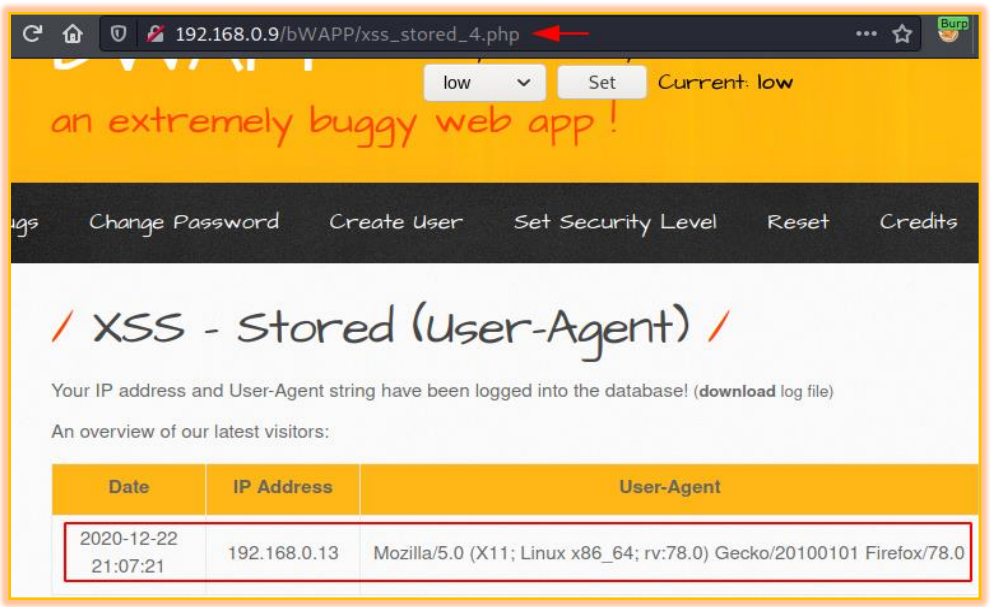


With the first response, we could notice that the application used the **DNS service** as the payload was bounded with the **nslookup** command. Thereby this confirms that the web-page is vulnerable to OS Command Injection.

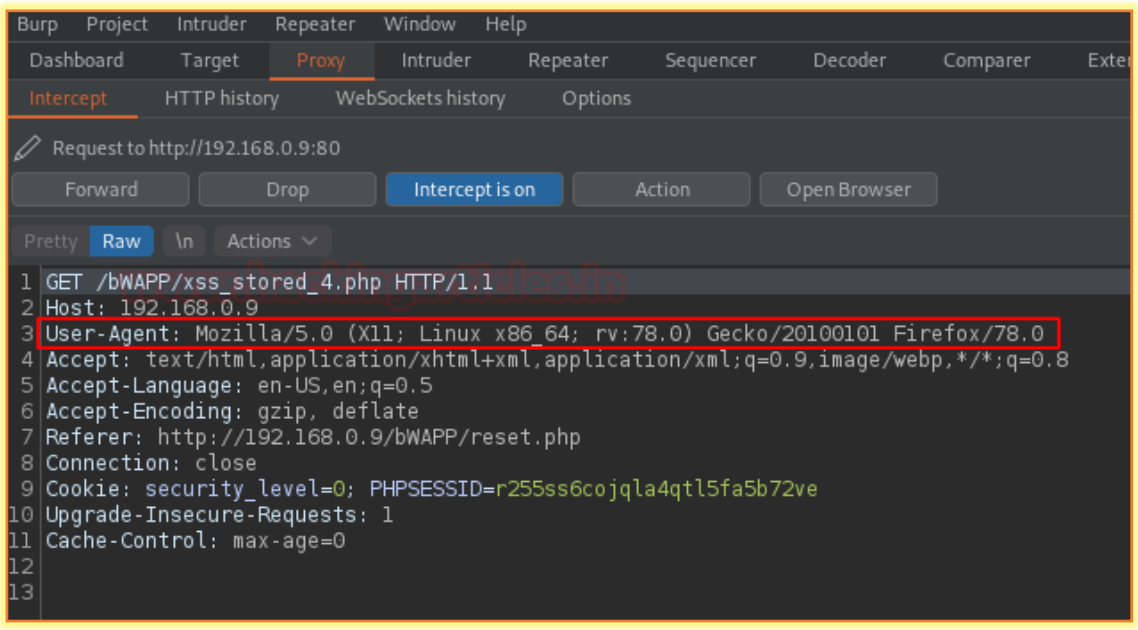


Cross-Site Scripting Detection

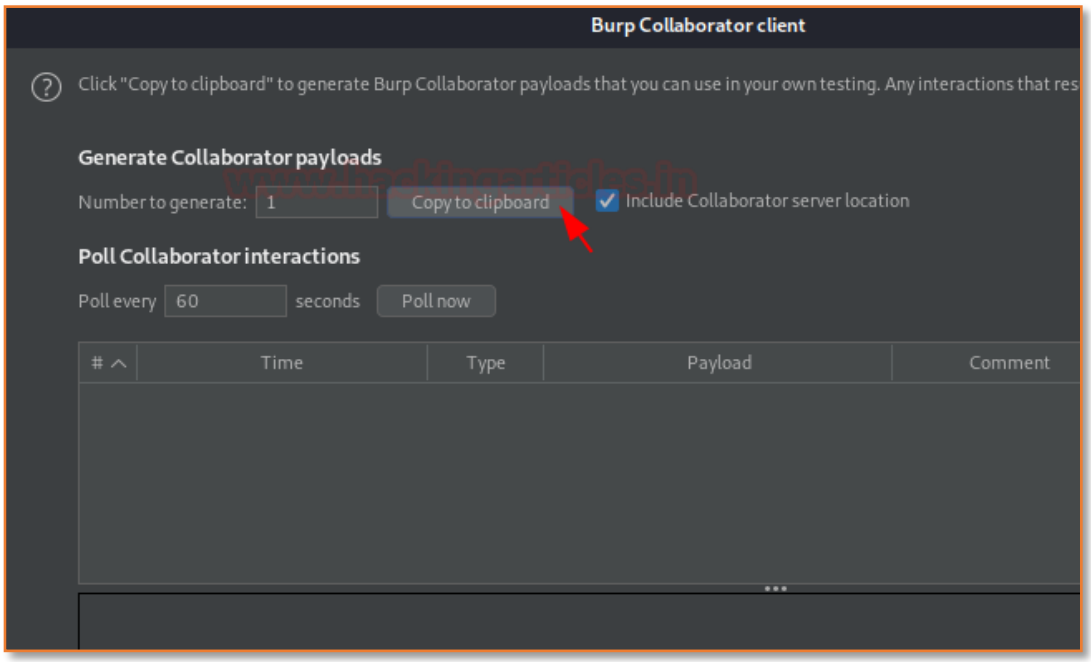
Cross-Site Scripting is a client-side code injection attack where malicious scripts are injected into trusted websites and are triggered when the user visits the specific suffering web-page.
But many times, we never know, where our injected payload will end up or when it will be executed. Thereby, in 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.



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

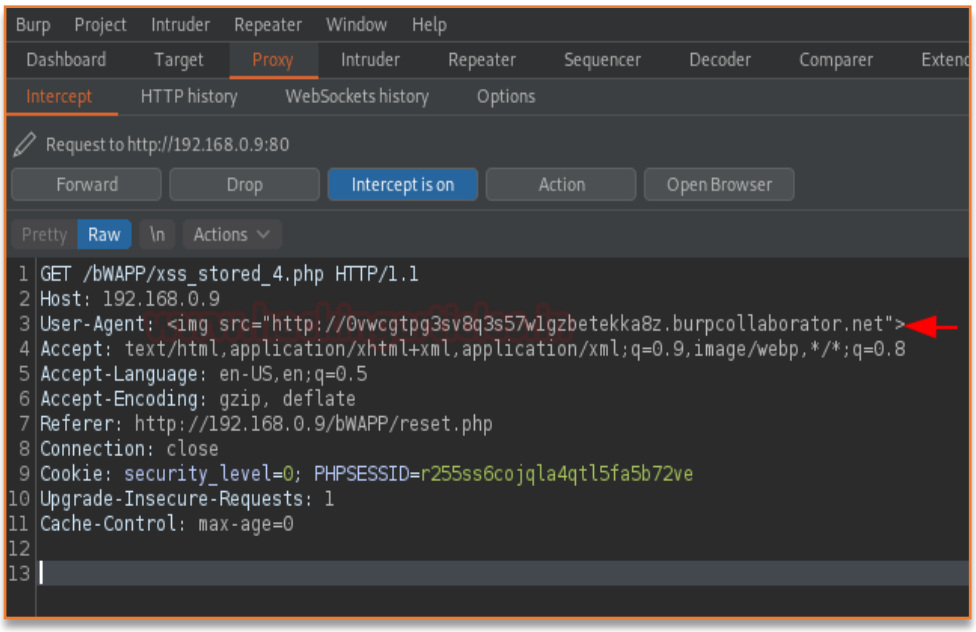


So, in order to manipulate this User-Agent, let's first copy the Collaborator payload by moving the Collaborator Client window.

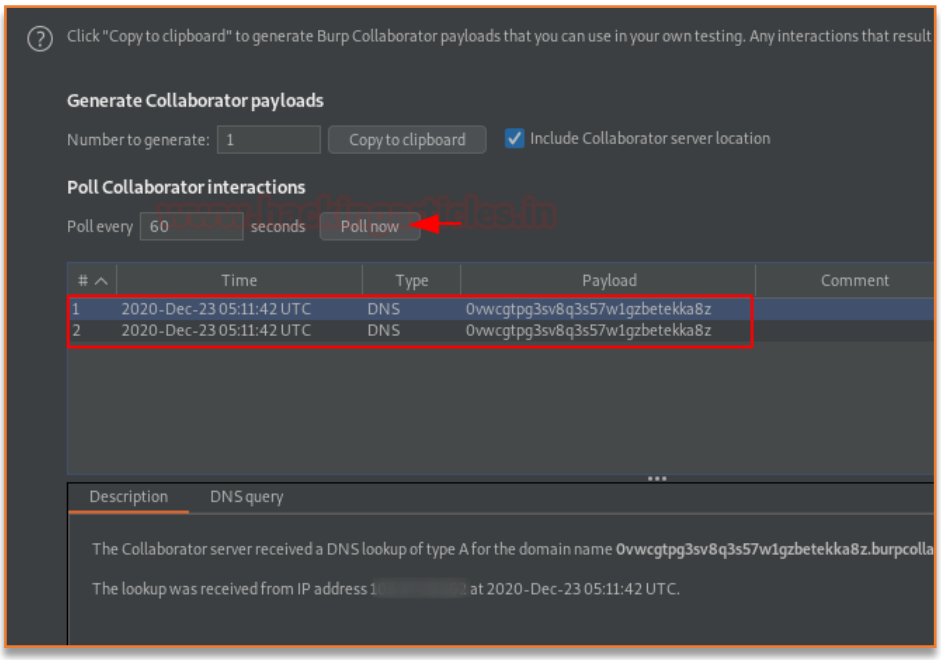


Back into the proxy tab & **modify the User-Agent** with the image tag and hit the **forward button.**

```
<img src="http://copied_payload_value">
```



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.



NOTE –

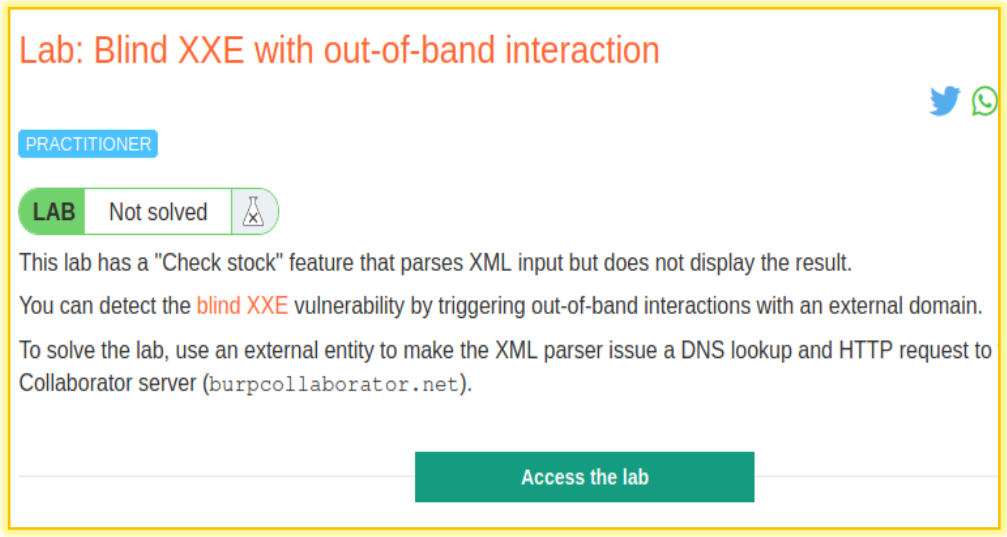
However, this is a basic demonstration as the vulnerable application has been hosted locally, but over in real-life scenarios, it takes time to get executed by the administrator or the developer, as many applications stores user-agents in their databases.

Blind XXE

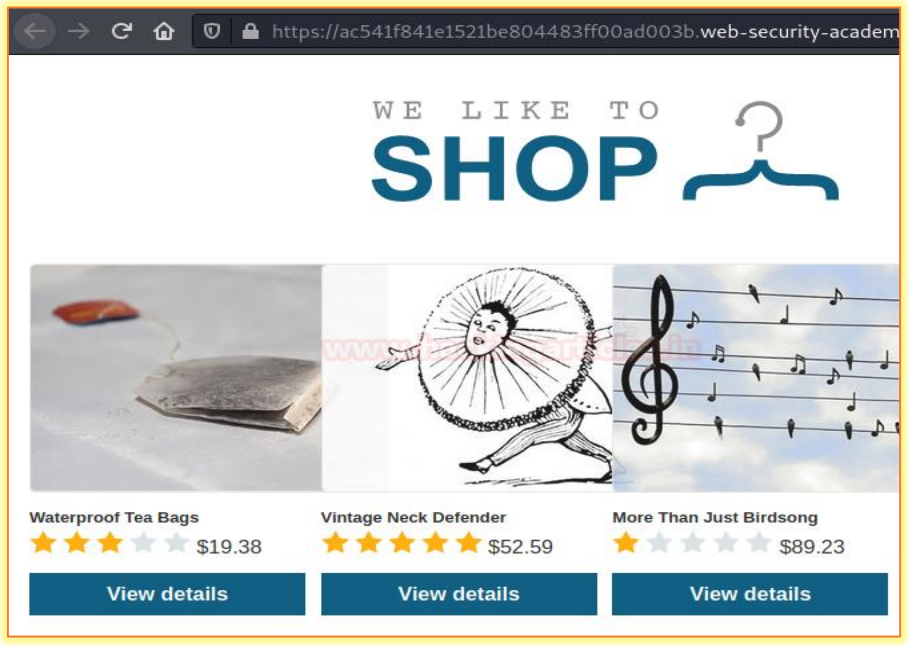
XML External Entity (XXE) attacks are the most common in today's era, as almost every application carries up XML inputs and parse them. However, the majority of these are blind ones as the weakly configured XML parser parses the malicious content but do not render it directly on the visitor's screen.

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.

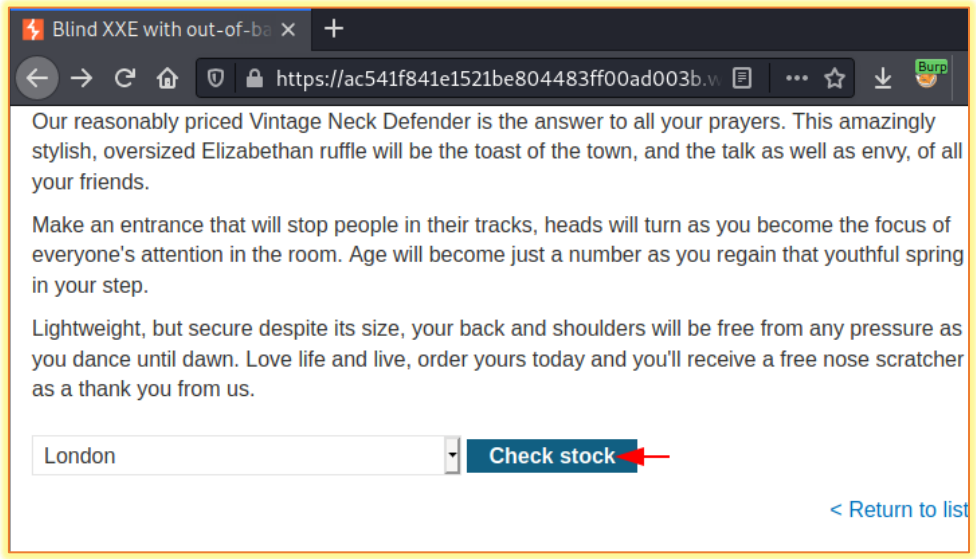
Login into the **PortSwigger academy** and drop down till **XML external entity (XXE) injection** and further choose the lab as "**Blind XXE with out-of-band interaction"** and hit **"Access the lab"** button.



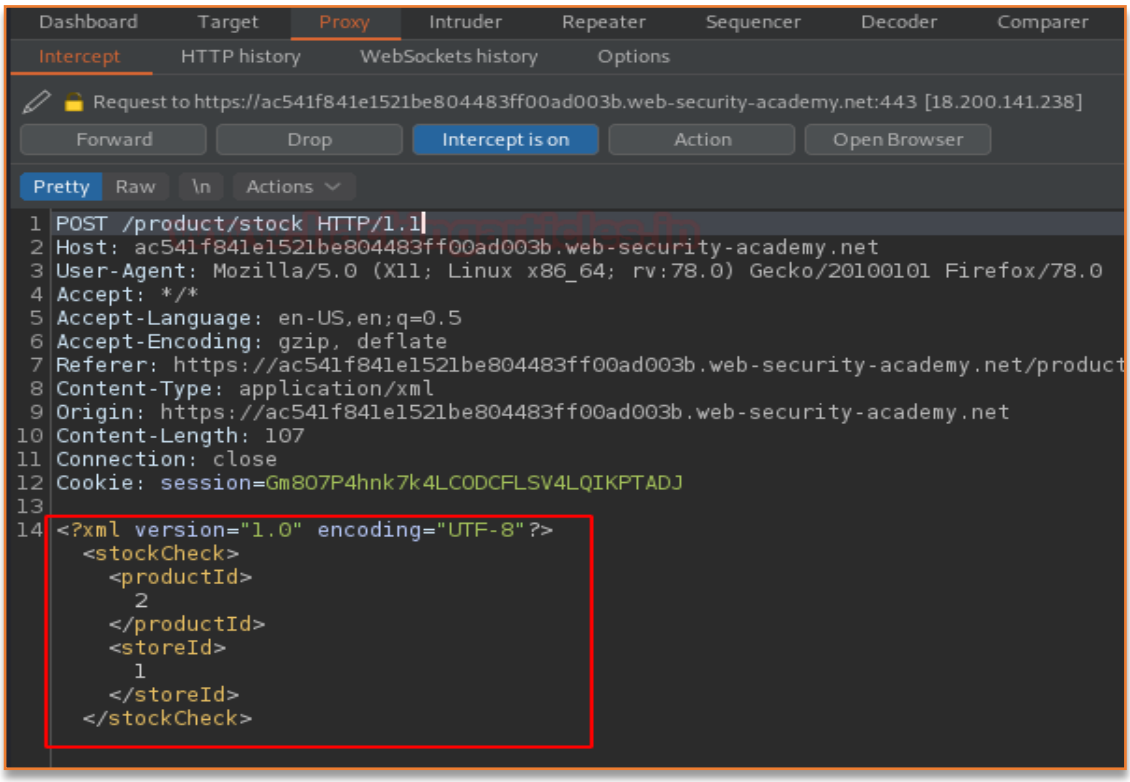
There as we do so, we'll be redirected to an e-commerce web-page; let's check out the details of the **"Vintage Neck defender".**



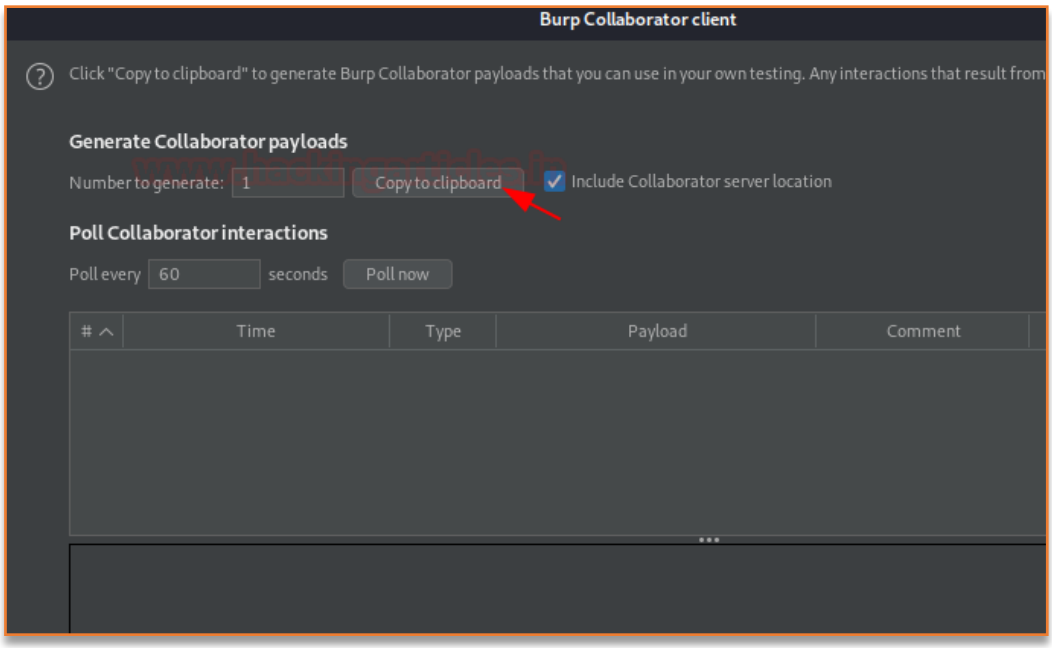
Over in the details of the item, as we scroll down to the bottom, we got something like **"Check Stock".** Seems to be suspicious, let's turn ON our burpsuite monitor and capture the on-going HTTP Request for the Stock values.



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

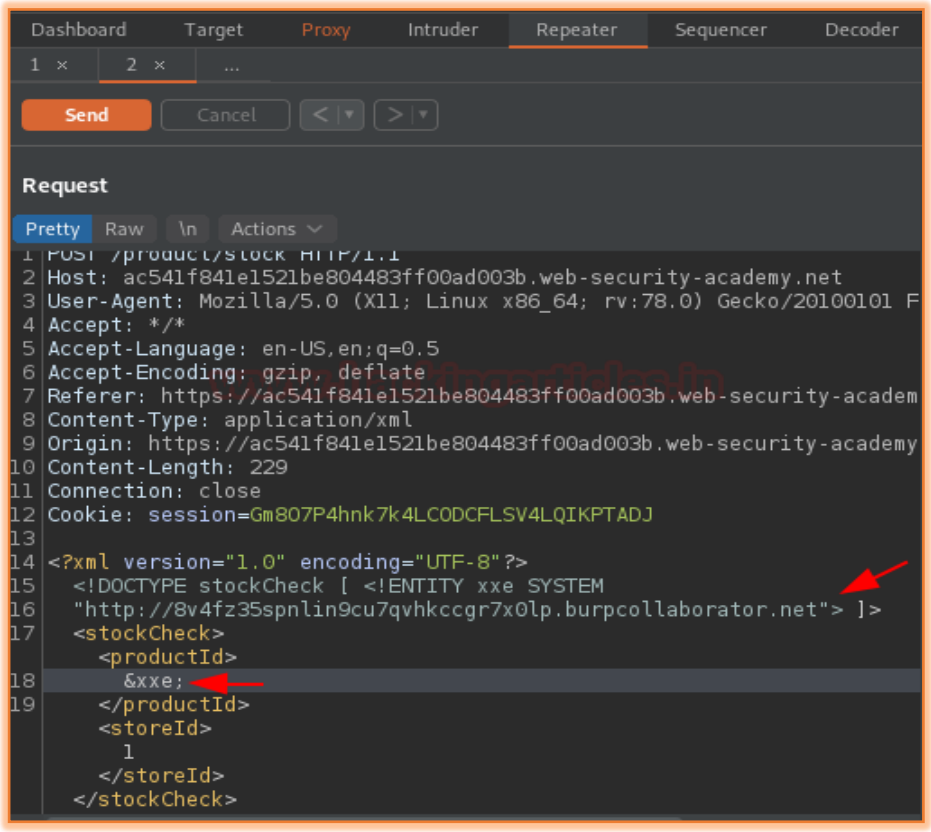


Time to modify the content and check for an XXE vulnerability, thereby in order to detect let's copy the Collaborator payload value!!

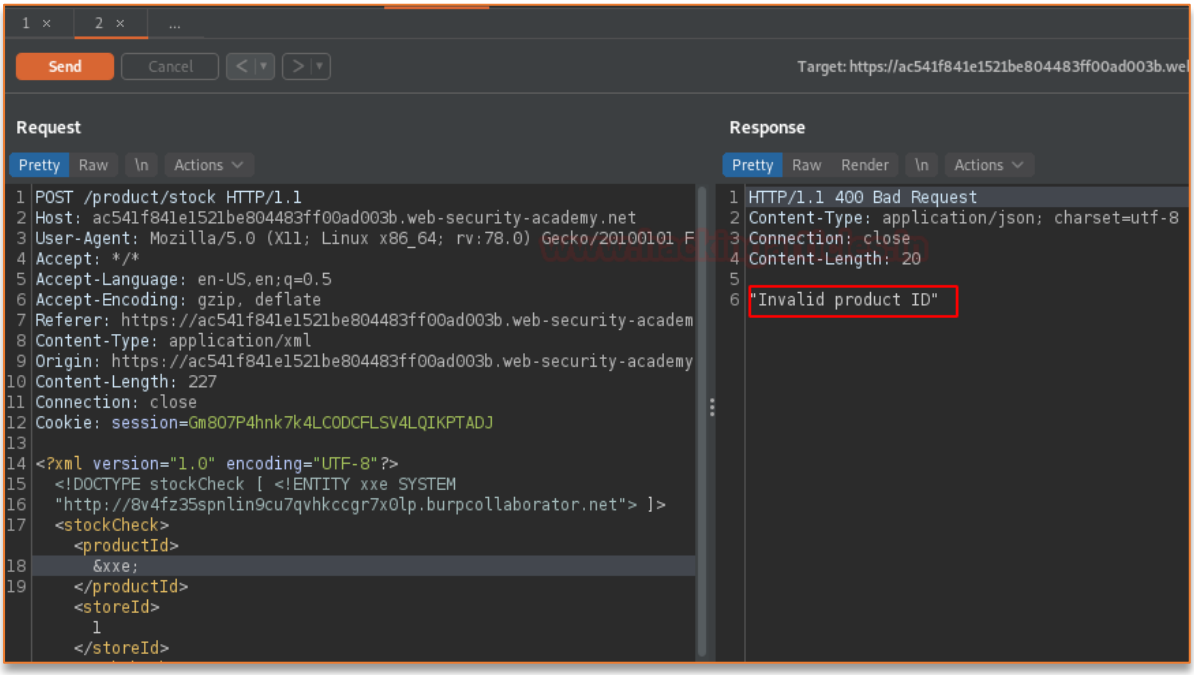


Now, let's first share the request to the Repeater tab and then we'll inject the copied payload into the XML data and call up the value within the productID as shown in the below image –

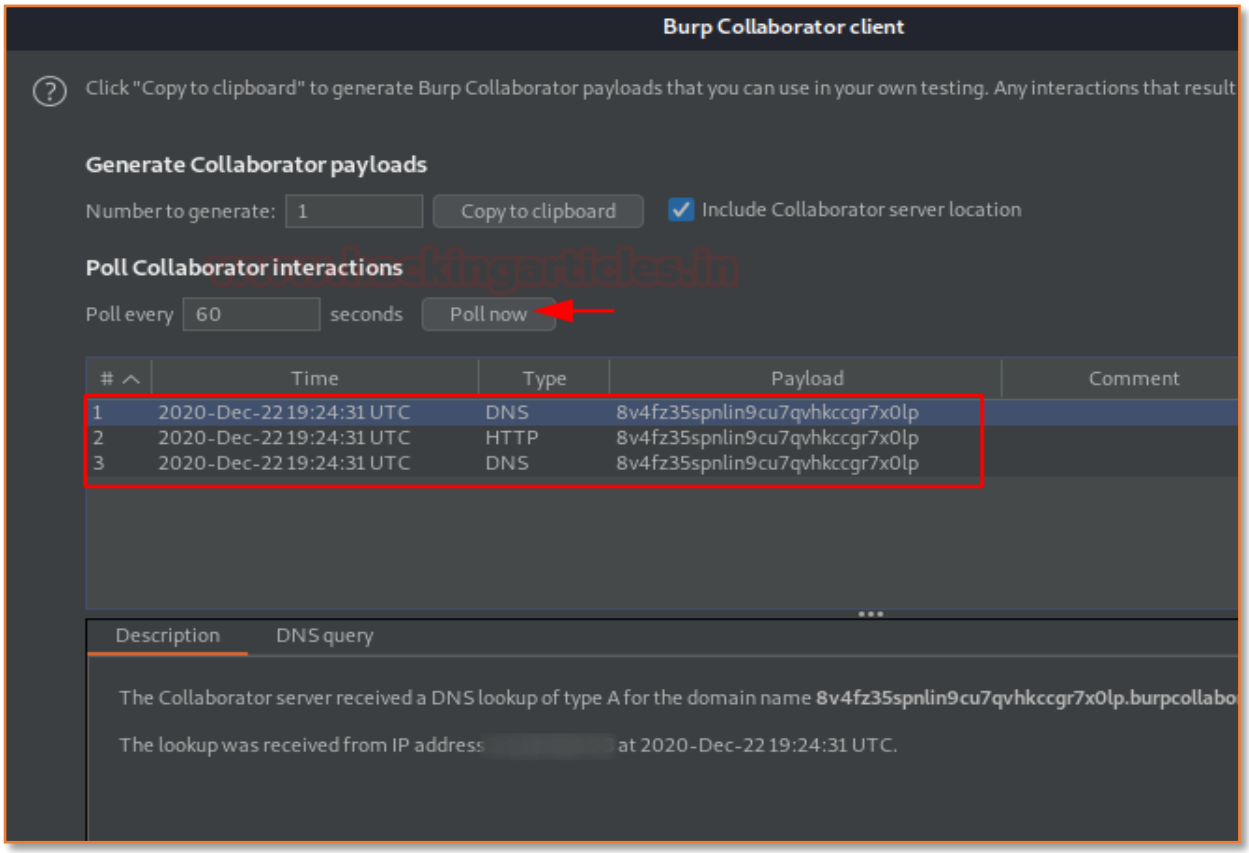
<!DOCTYPE stockCheck [<!ENTITY xxe SYSTEM "http://payload_value_.burpcollaborator.net">]>



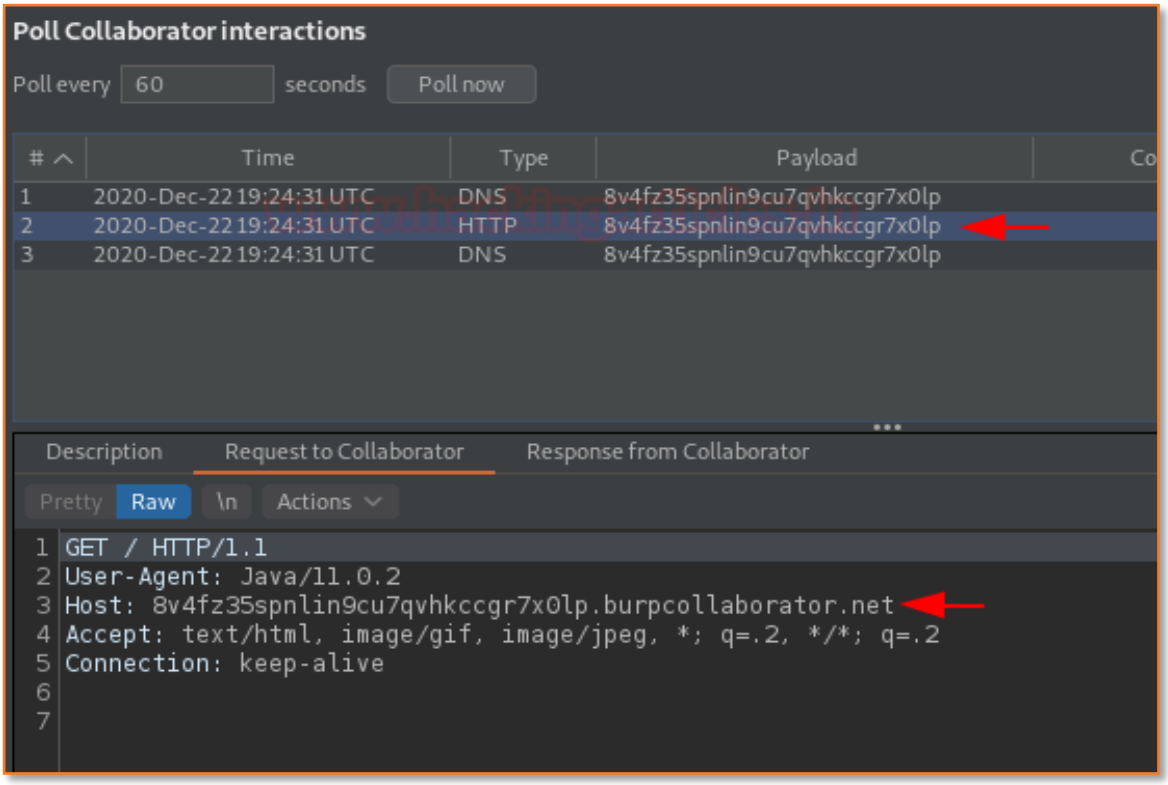
Now, as soon as we hit the **"Send"** button we got the response in the Repeater tab.



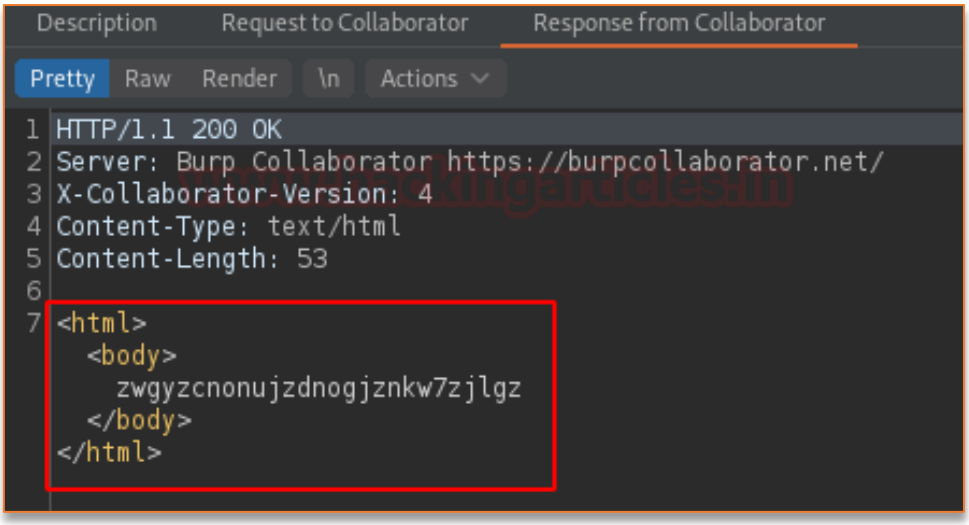
But wait, it's an error an **"Invalid Product ID"**!! Now what?
Let's check out the collaborator, did the request hit at the background or not. And there it is, this time, not only the DNS but also an HTTP service has also been triggered.



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



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.

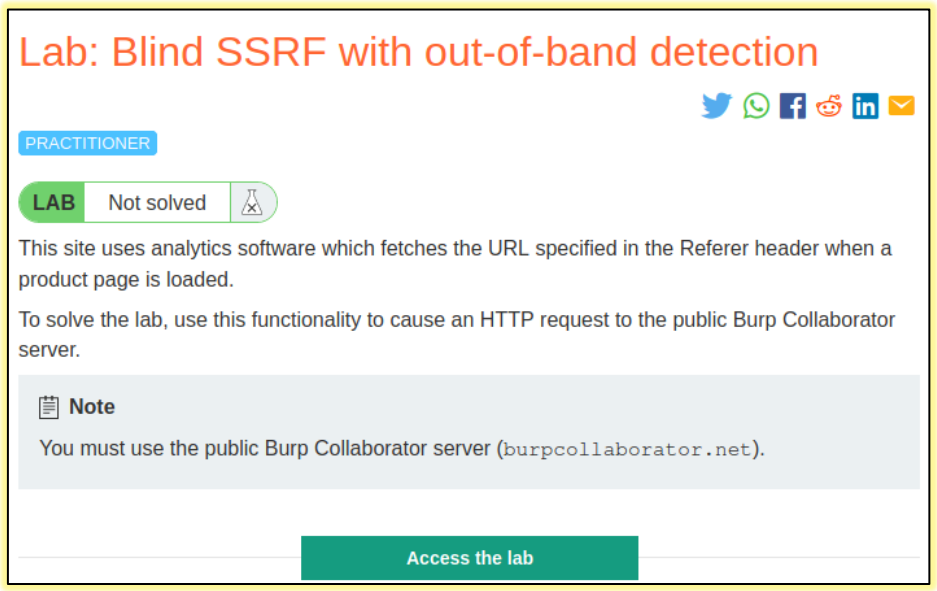


Server-Side Request Forgery

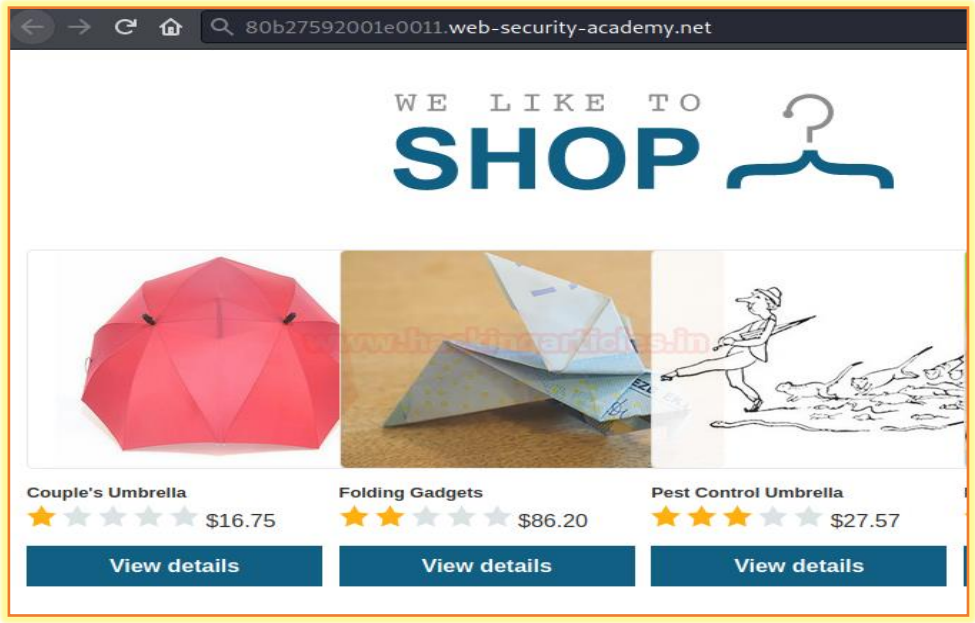
SSRF or **S**erver-**S**ide **R**equest **F**orgery is the most crucial web-application vulnerability where the attacker is able to use the vulnerable application to send **crafted HTTP Request** to a third-party server or application in order to **fetch internal information, banners or open ports**. 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.

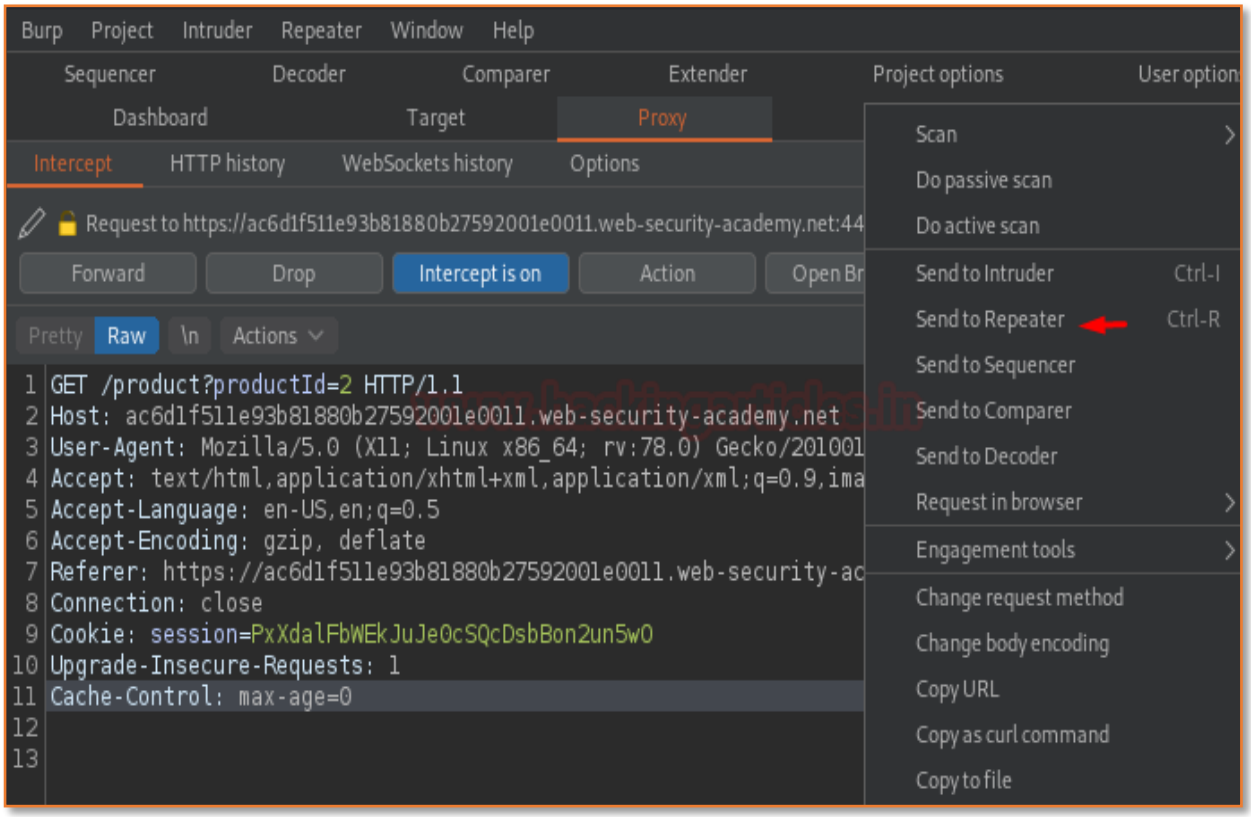
Back into the PortSwigger Academy, switch to Blind SSRF with out-of-band detection and hit the **Access the lab** button.



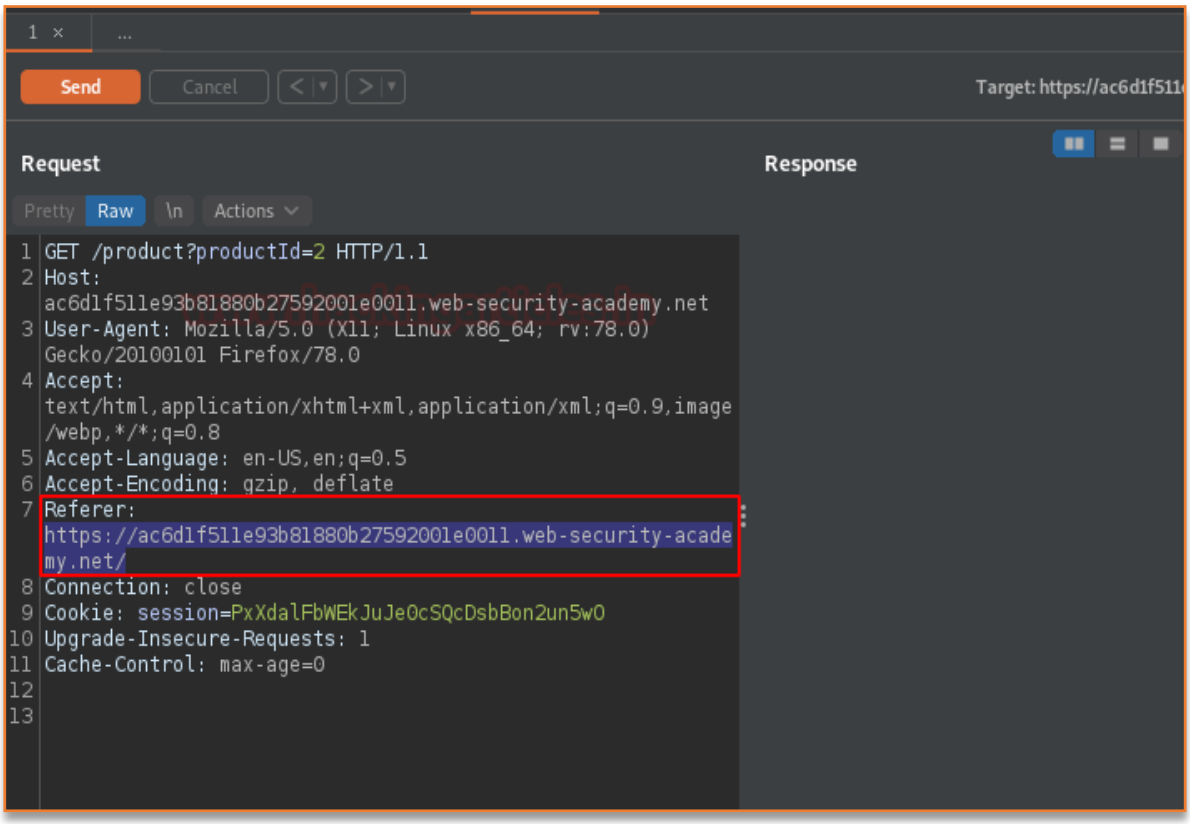
As soon as we do so, we'll be redirected back into the shop page, but this time content is somewhat different. Let's check the second item within it.



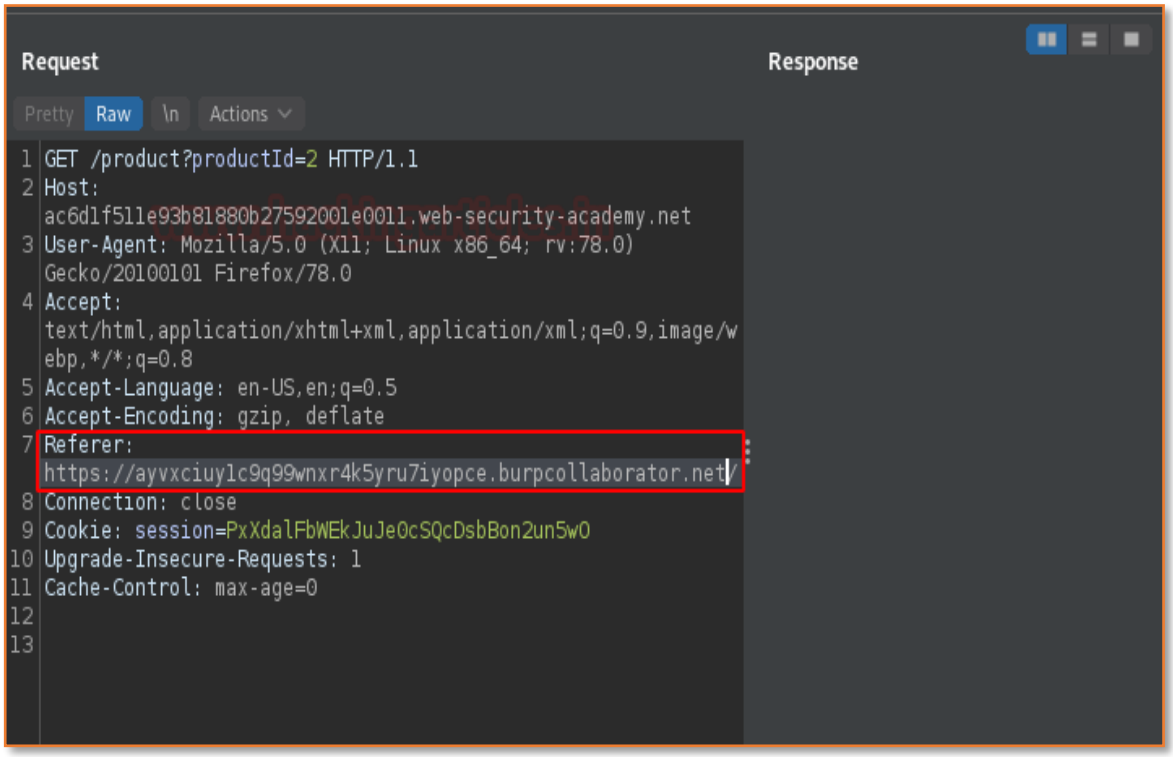
But before accessing the item, let's turn our Burp monitor and capture the ongoing HTTP Requests for it, and as it captures them, we'll throw it directly to the repeater.



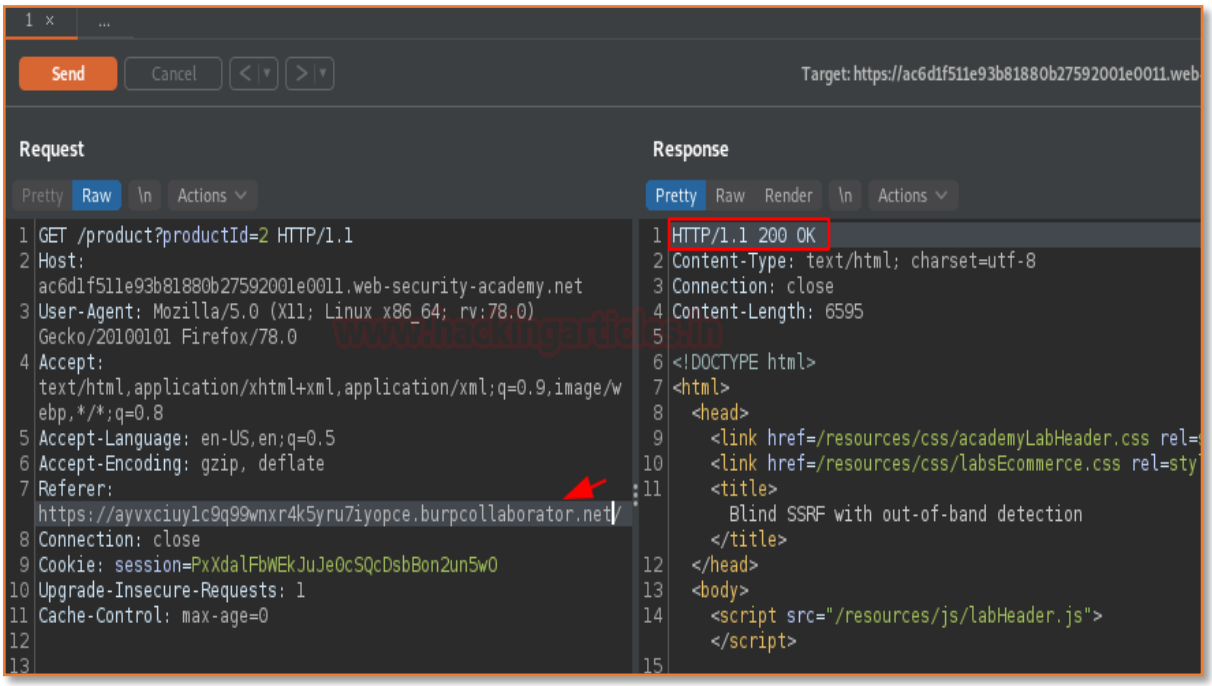
Over in the request section, the referer value seems to be injectable let's manipulate it.



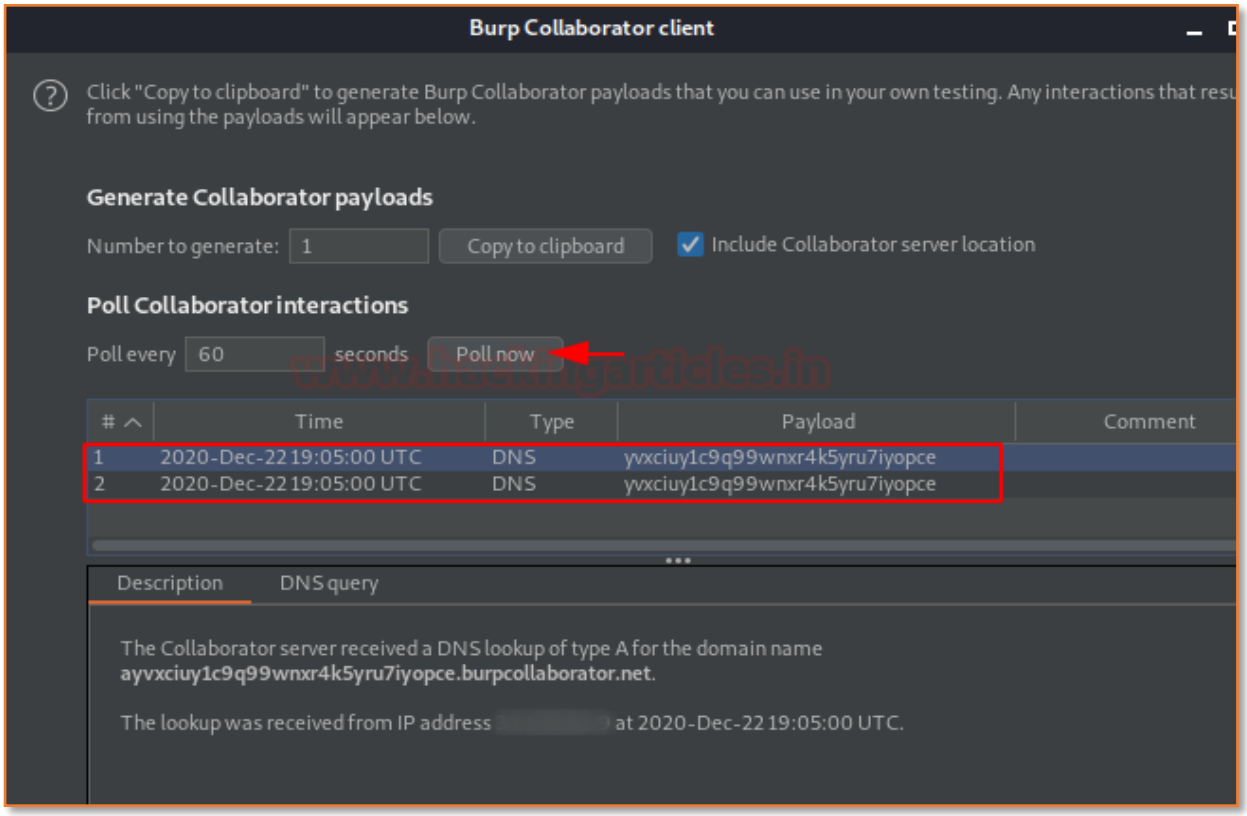
Back with the collaborator client window, copy the payload, and modify the referer with it.



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??



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.

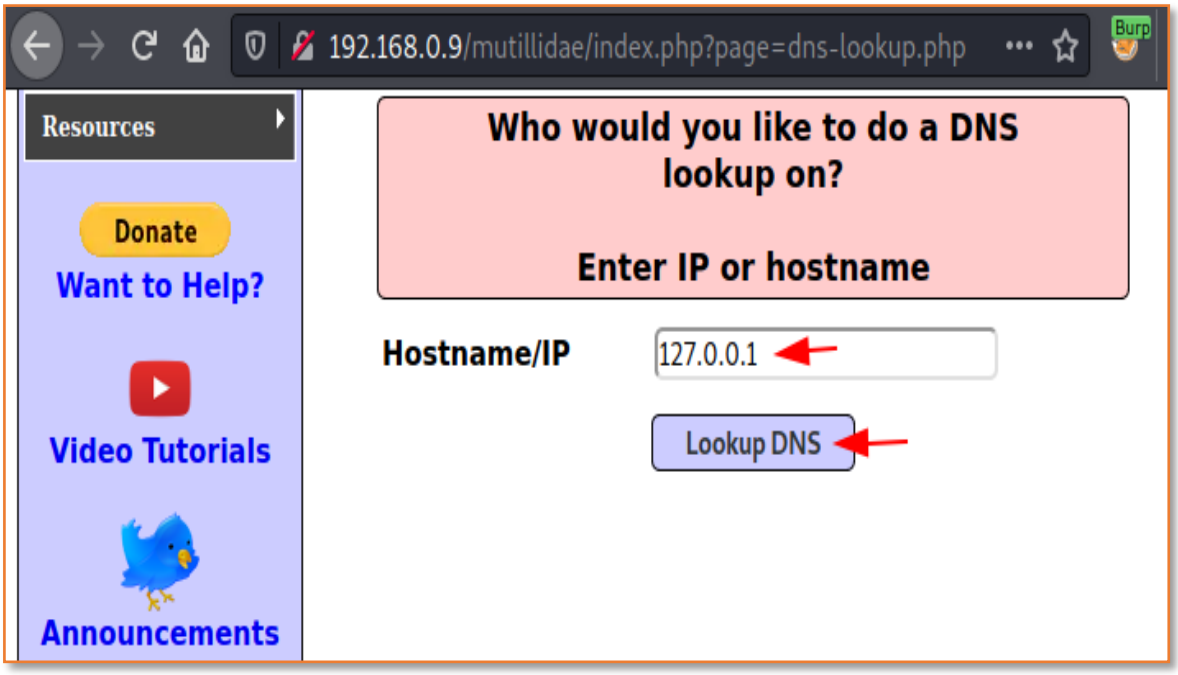


Let's open the browser again. Cool the lab has been cleared.

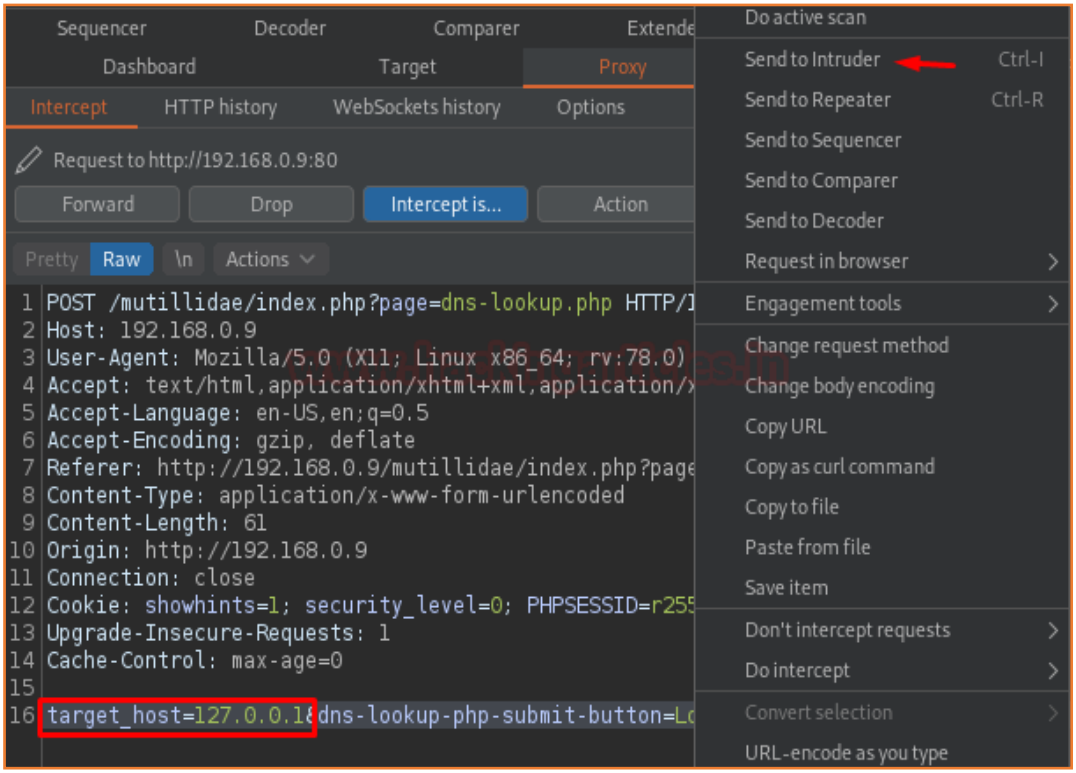
Fuzzing for SSRF Detection

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.

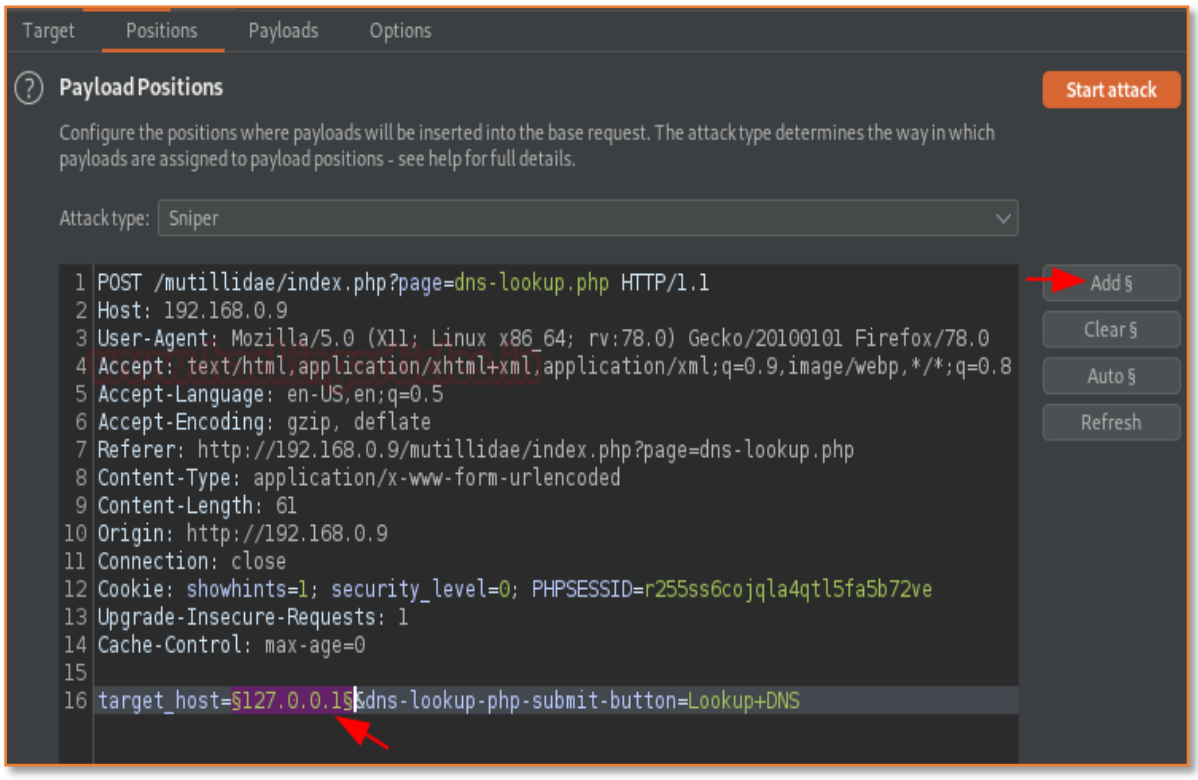
As soon as we reach there, we got an input field to enter the IP Address, let's enter the localhost IP.



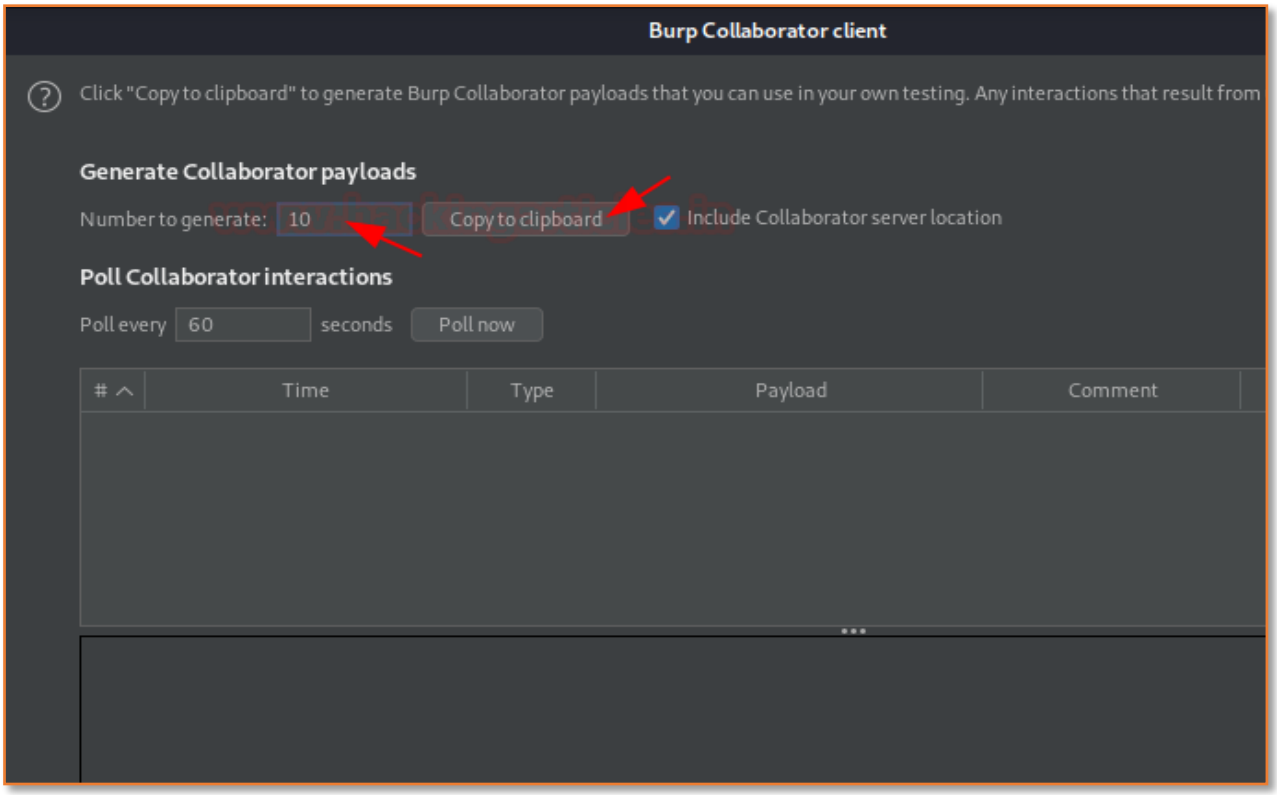
Before hitting the Lookup DNS button, let's configure our burp monitor in order to intercept the passing HTTP request and as soon as it captures it up, we'll share it to the Intruder tab.



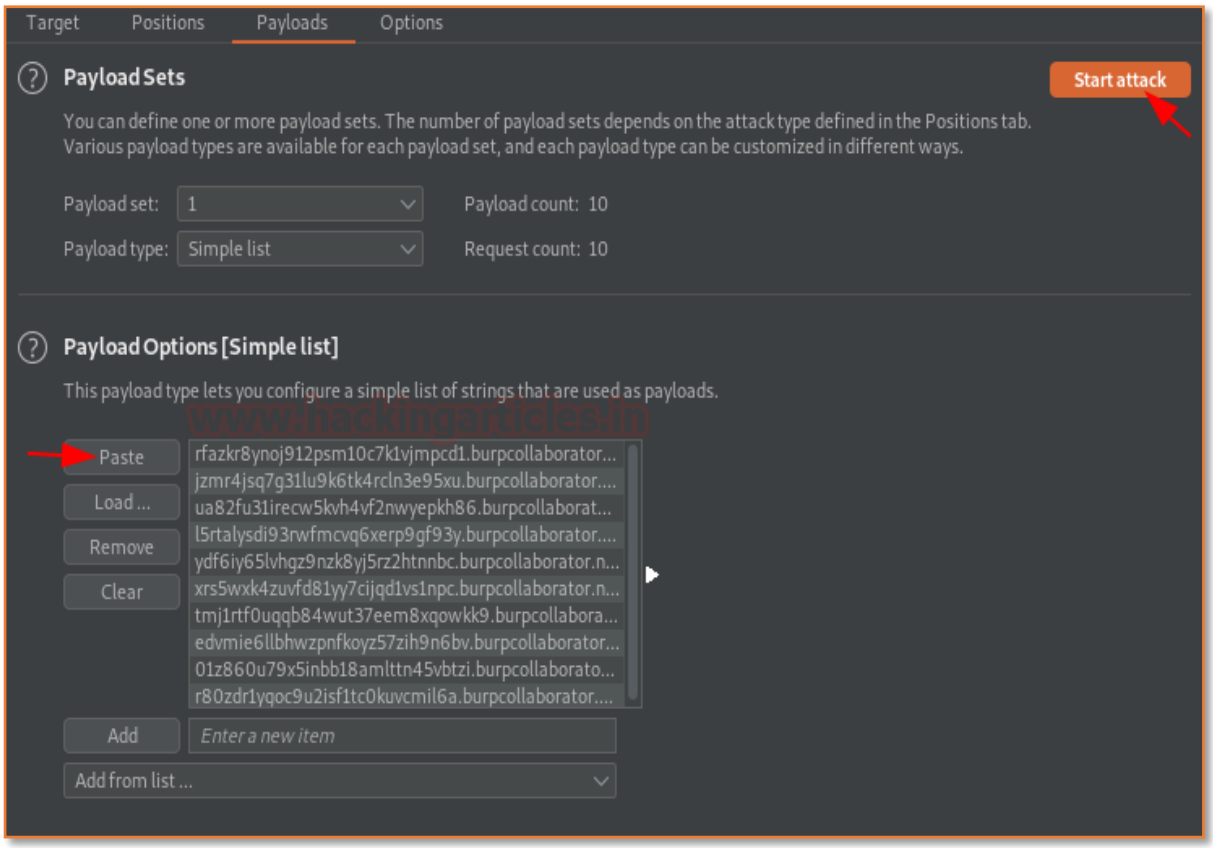
Over in the intruder tab, we'll modify the injection point and set the payload position at the IP address we've entered.



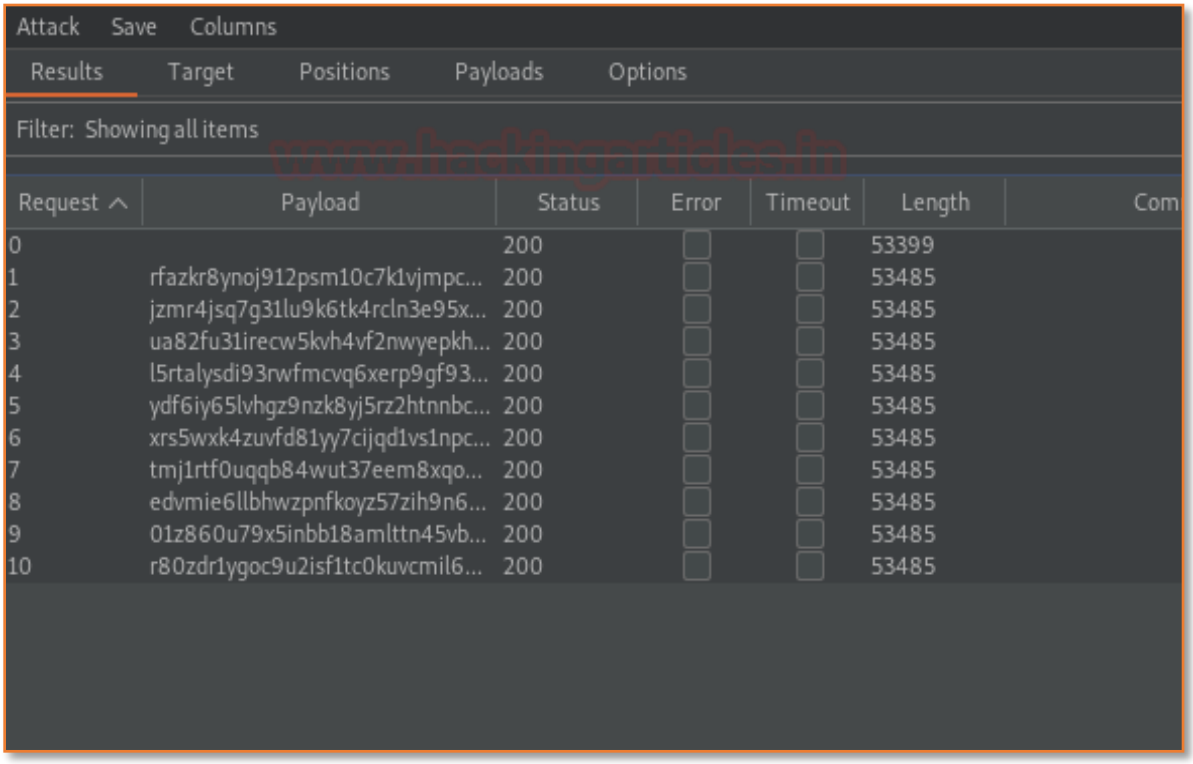
Now time to take help. Back into the collaborator client window, set the number to generate to 10 and hit the copy to clipboard button.



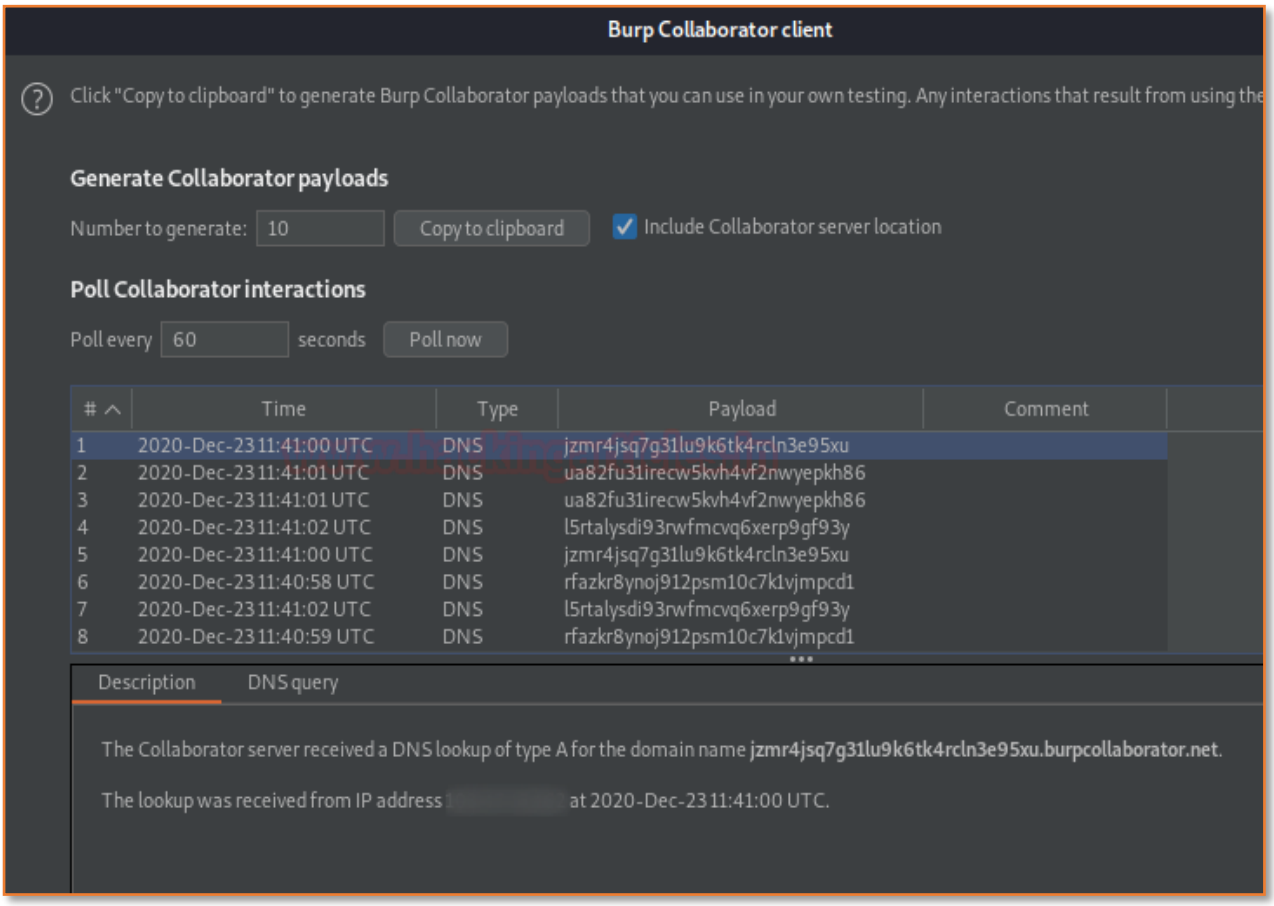
Over into the payload section, simply hit the **Paste** button in order to move all the copied payloads in the empty box, further hit the **"Attack"** button to initiate the fuzzer.



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



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.



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/

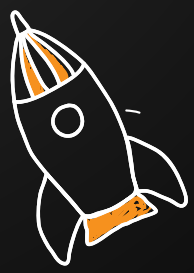
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

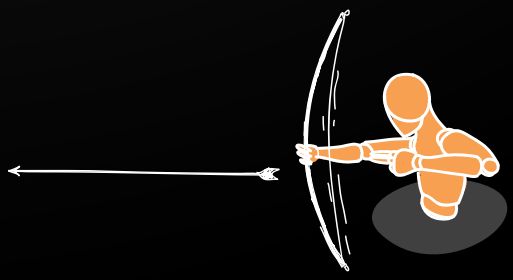
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux