iGNITE Technologies

Burp Suite for Pentester

Sequencer

TABLE OF CONTENTS

Abstract

*Whenever we log into an application, the server issues a **Session ID or a token**, and all over from the internet we hear that the session ID we get is unique, but what, if we could guess the next unique session ID which the server will generate?*

Today, in this article we'll try to overtake the application's algorithm that helps them to generate a **randomized session ID** for a specific user and will try to log in inside the application impersonating that user with our predicted session ID.

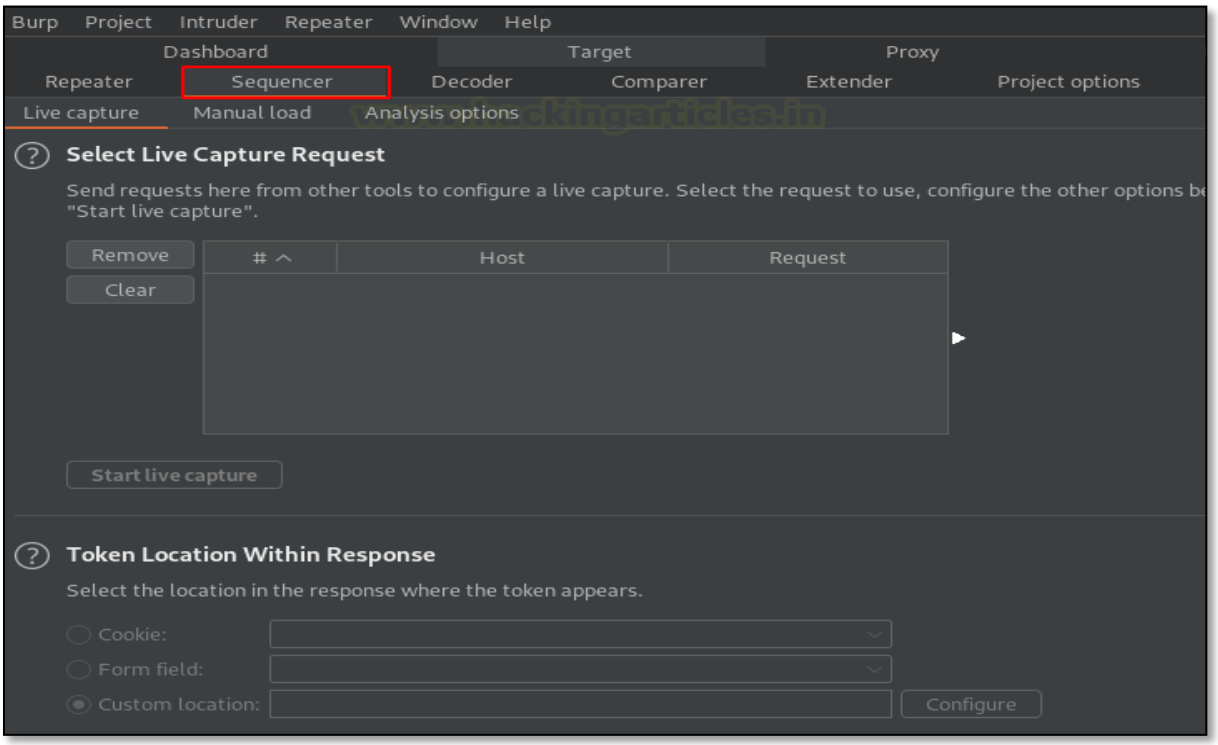
Introduction to Burp Sequencer

Burp Sequencer is a tool for analyzing the **quality of randomness** in a sample of data items. The data items can either be application's session ID's, CSRF tokens, password reset or forget password tokens or any specific unpredictable ID generated by the application.

The Burp Sequencer is one of the most amazing tools that try to **capture the randomness** or the **variances in the session ID's** by employing some standard statistical tests which are based on the principle of testing a hypothesis against a sample of evidence, and calculating the probability of the observed data occurring.

However, the tool tests the given sample in a number of different scenarios whether it is a character-level analysis or a Bit-level one, the analyzed output would be in the best-segregated format. To learn more about how the sequencer tests the randomness, check the sequencer's documentation from **here**.

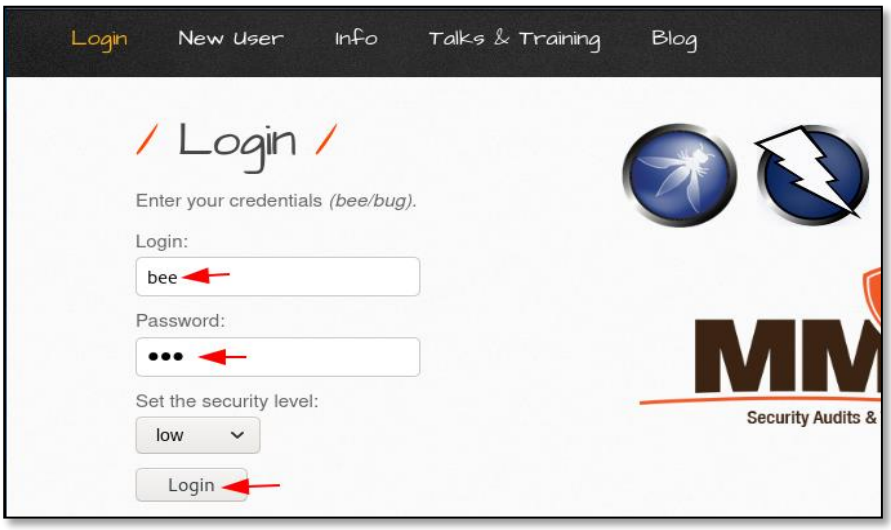
The best part of this tool is that it is available for both the editions i.e. for the **Professional and community version**, you just need to tune in your burp suite application and navigate to the **Sequencer tab** over at the top panel.



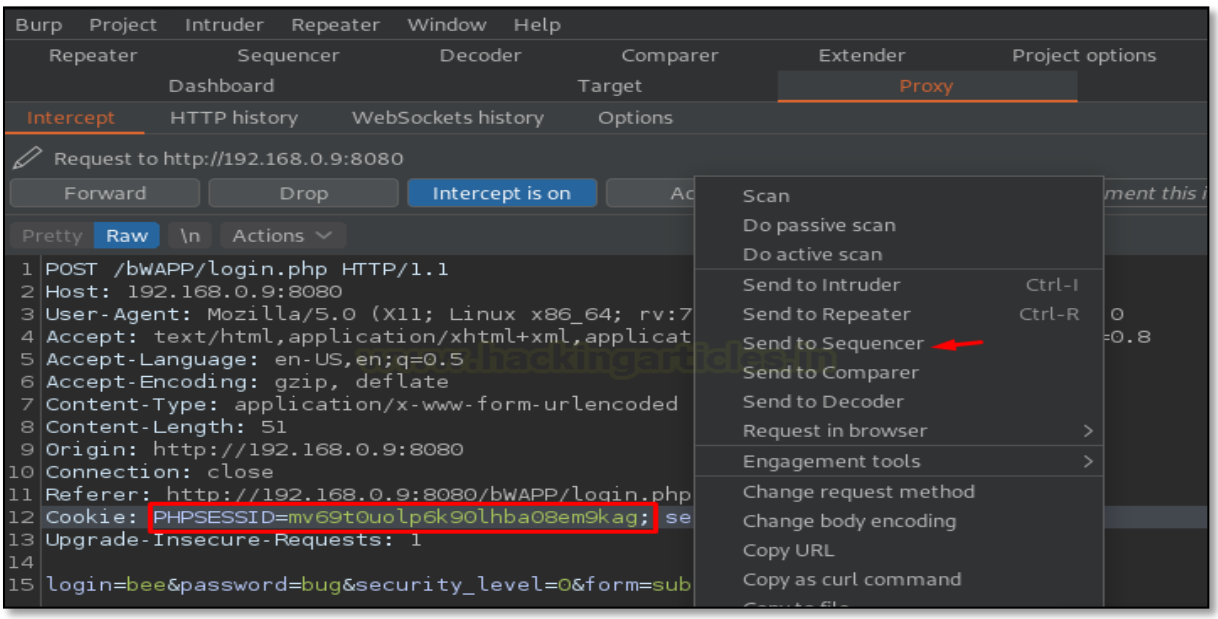
Exploitation Using Burp Sequencer

Session ID exploitation via Sequencer

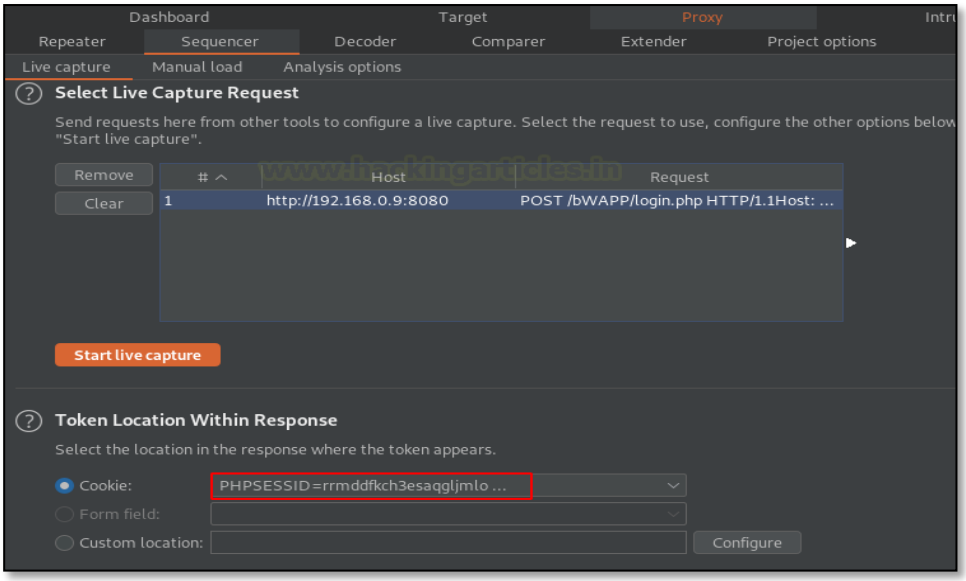
Whether it's a basic Session ID or a token generated from the server-side, the burp's sequencer analysis that all, as the tool's only requirement is a **"Request"** shared with it.
So, let's initiate the analyzer within the burp sequencer by capturing and sharing a login Session ID over from our favourite vulnerable application i.e. bWAPP.
Feed the target IP in the browser and login with **bee: bug**.
Wait!! We talked about login session ID right, so let's turn the proxy service here only, and then hit the **"Login"** button.



Let's check our burp suite, whether it captured the request or not. From the captured request we can see that a **PHPSESSID** is in the **Cookie header,** let's share the complete request with the **Sequencer** by hitting right-click over on the white space.

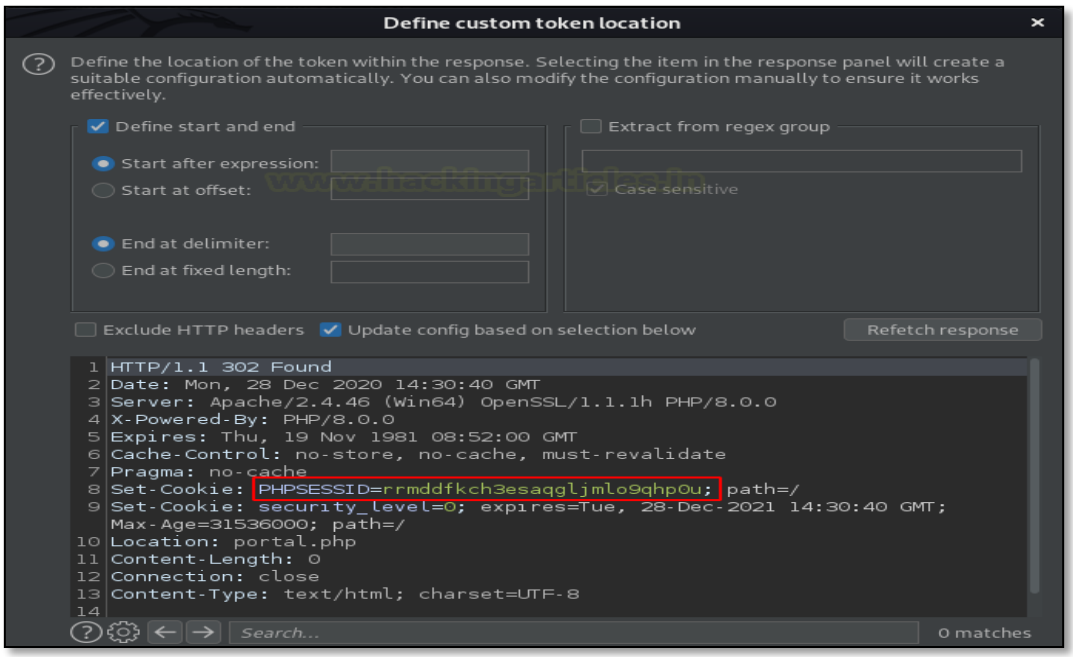


As soon as the Sequencer receives the request, the empty fields got filled up directly with the Token ID that goes with the **"Response"** for the specific shared Request.

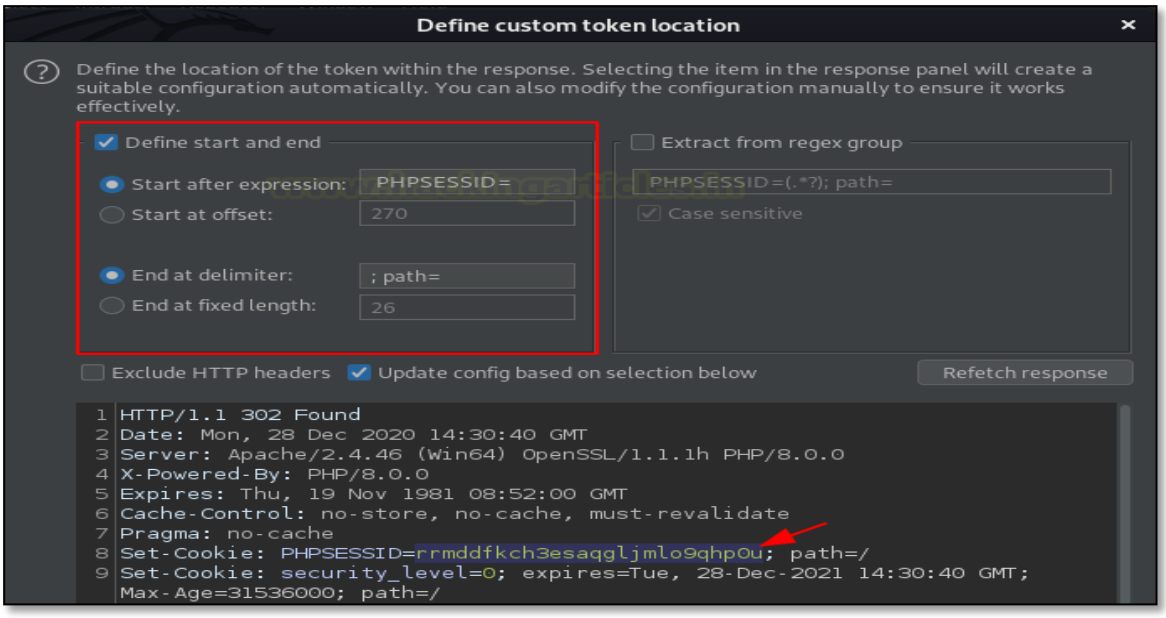


However, there are times when we want the sequencer to analyze some different value. Thus, over in such cases, the burp suite's creators give us the opportunity to define the **Custom location** within the response. Let's check that out.

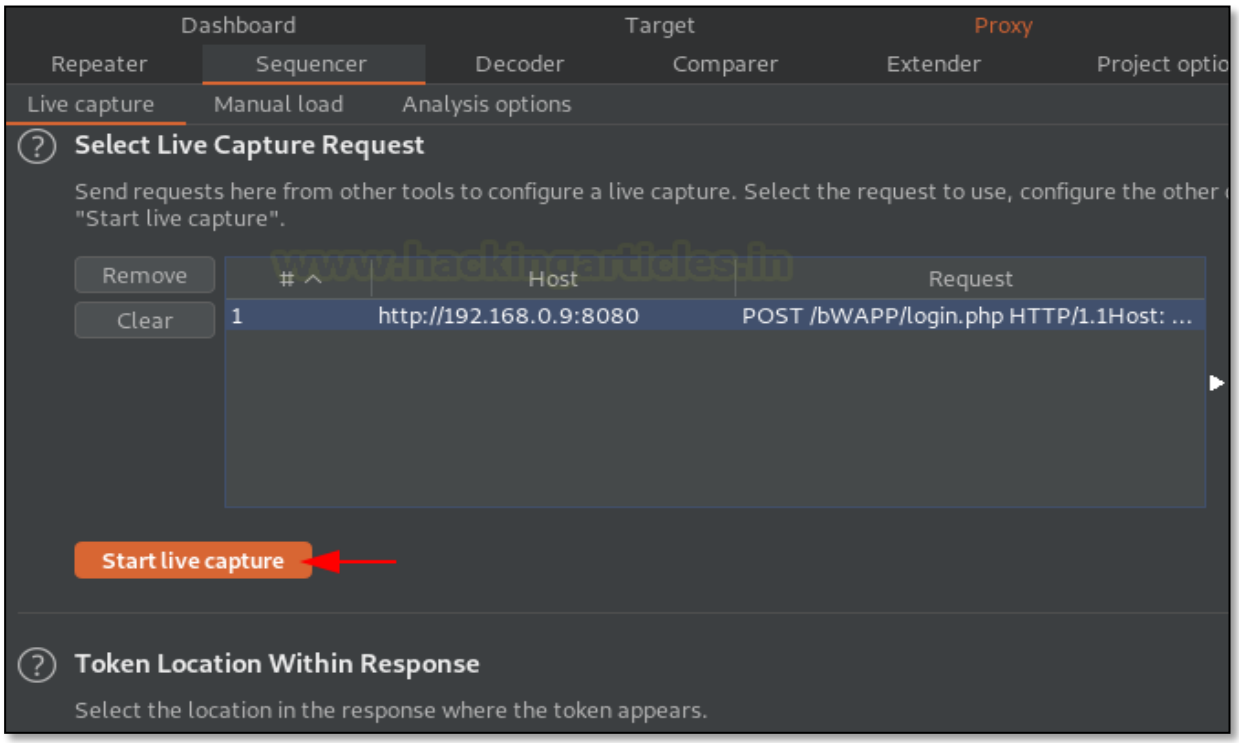
Hit the **Custom location** radio button below at the cookie option and then navigate with **Configure**. As soon as we do so, we got a new window opened as **"Define Custom token location"** where we're having the response of our shared request.



Over into this, we can opt either the defined options whether **"Define start & end"** or **"Extract from regex group".** Let's check the first one.
As we hover and select the value from the shared Response, we got some manipulation at the Start and the End delimiters. Thus, hit the **Save button** and our custom location will be defined up at the panel.



Now simply hit the **"Start Live Capture"** button having the **Cookie** option selected at the **"Token Location within Response"** section.

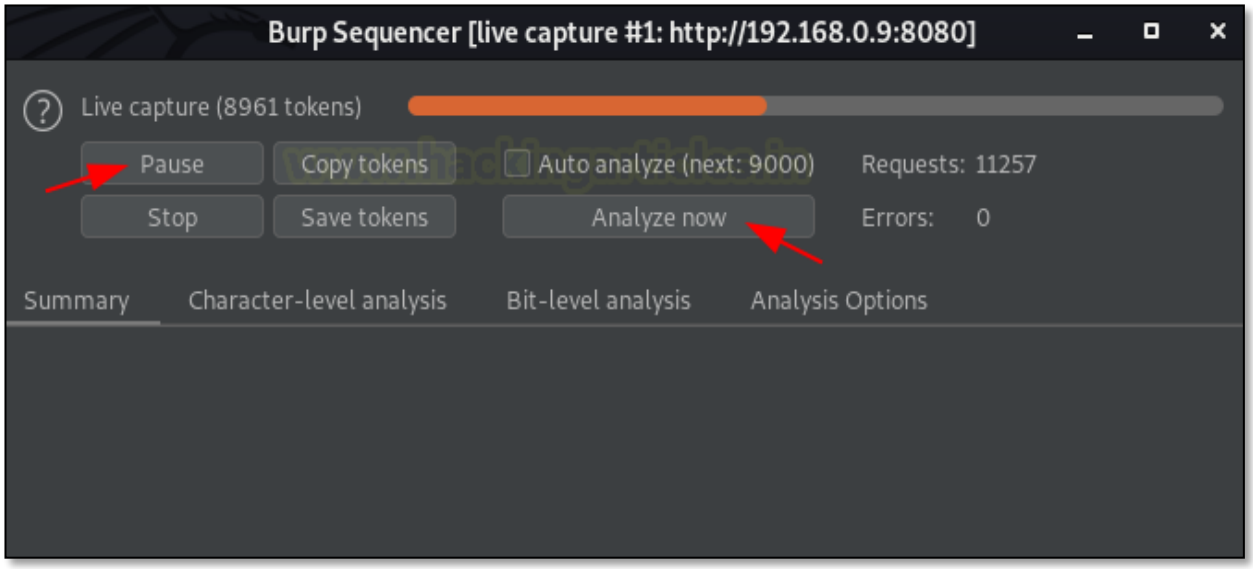


And, as we did that a new window **[Live capture #1]** got popped up. *Okay!! But what's this?* Let's define it all.

As soon as the **"Start Live Capture"** button got fired up, *Burp repeated the original request (potentially about a thousand times) and thereby extract all the tokens received from the responses.*

However, within all this, once the **Capture page** boots up, a **progress bar** is displayed with a counter of tokens generated and the requests made by the sequencer. A number of buttons also contributes into this Live Capture window as –

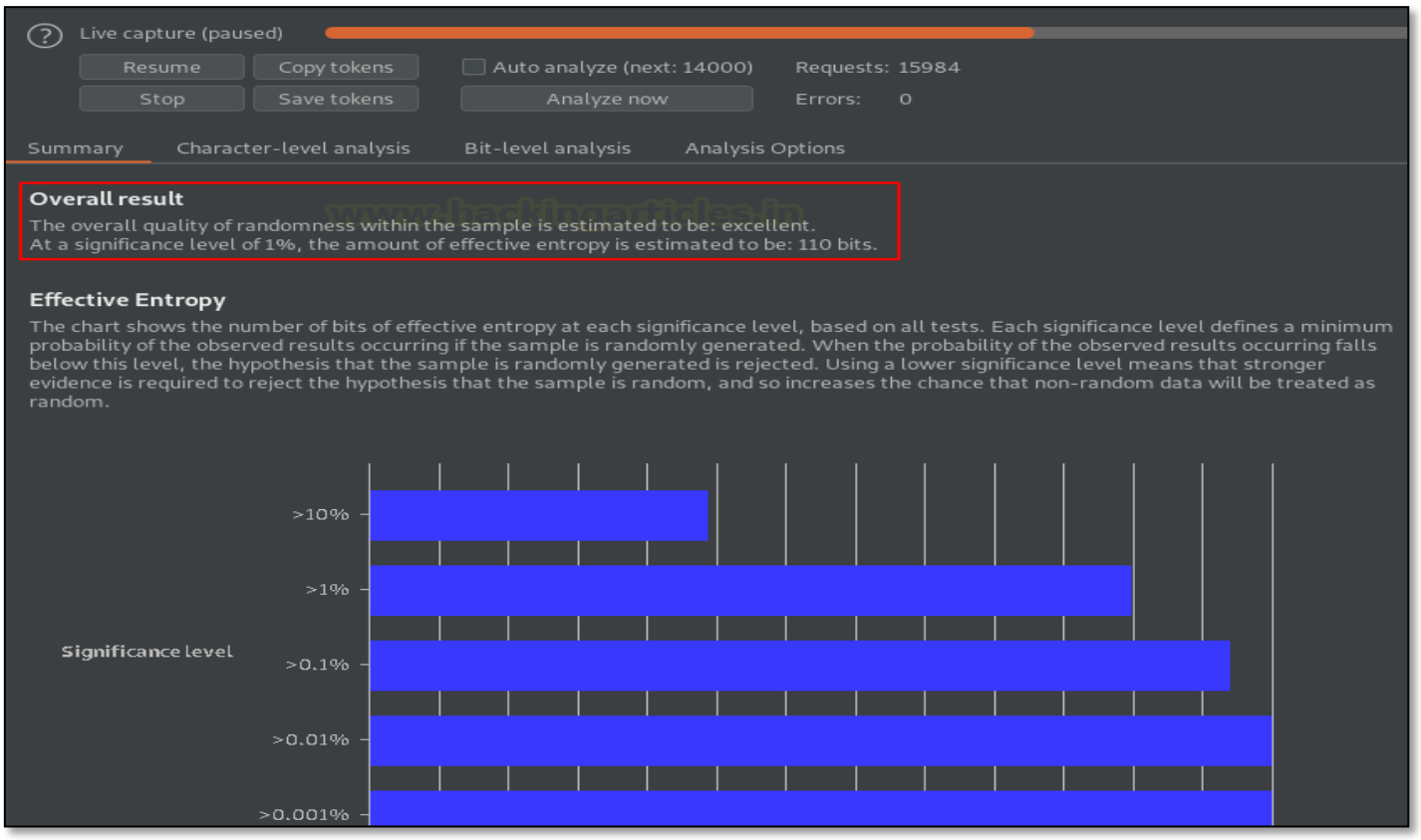
- **Pause/Resume:** This temporarily pauses the capturing request and the counter, such in order to help the pentester to analyze the requests generated till then.
- **Copy Tokens:** It helps to copy all the randomized tokens generated.
- **Stop:** A major road block for the live capture analyzer.
- **Save Tokens:** Output can be dropped down as in the form of randomized generated tokens into a defined file.
- **Auto analyze:** This check box (if enabled) will thus help the Sequencer to dump the analyzed results as soon as a specific number of tokens are generated.
- **Analyze now:** This button is only available when at least 100 tokens have been generated and if clicked, will thus print out the analyzed report on the screen.



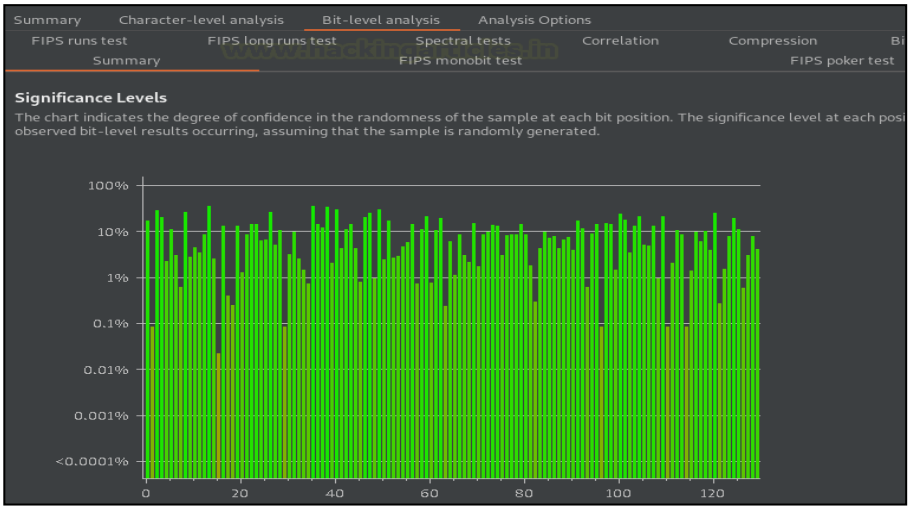
So, let's **pause** the sequencer and then we'll hit the **Analyze now** button, such in order to determine what it gathers.
From the below image, we can see that the sequencer had dropped a report by analyzing about **16000 requests,** having an overall quality of randomness within a sample estimated to be **"excellent".**
*We could have either got randomness quality to be "**poor**" if the web application's session ID is repetitive.*
However, the amount of effective entropy is **110 bits** which are considered to be a good one as the least is **"64"** and the best is **"128".**

NOTE –

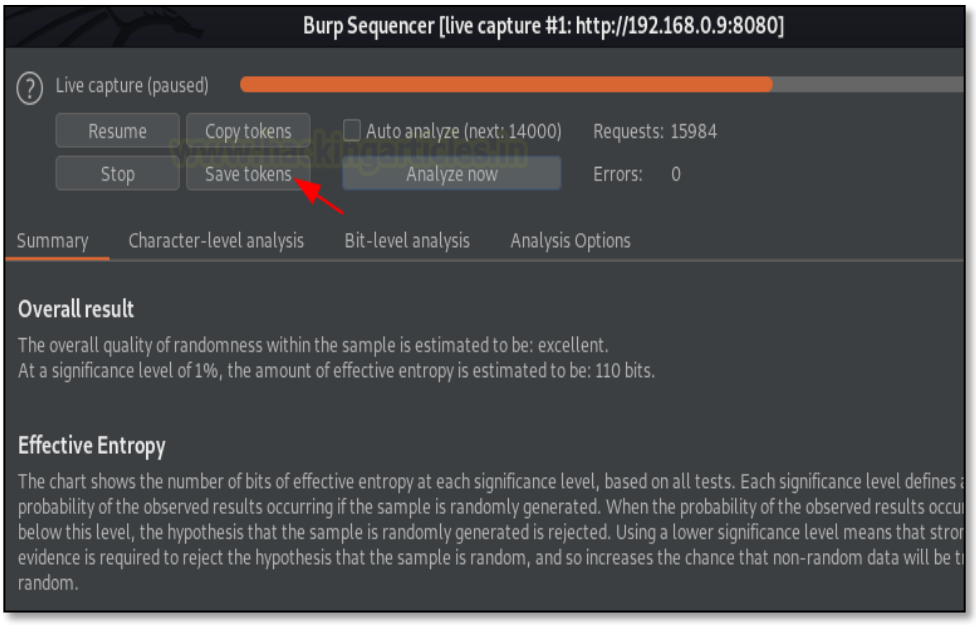
*Burp Sequencer works over **Sample,** thereby a large number of captured tokens will thus drop a better and a precise result. Thus, it is recommended to have at least a 1000-2000 sample tokens before analyzing the application's session ID randomness.*



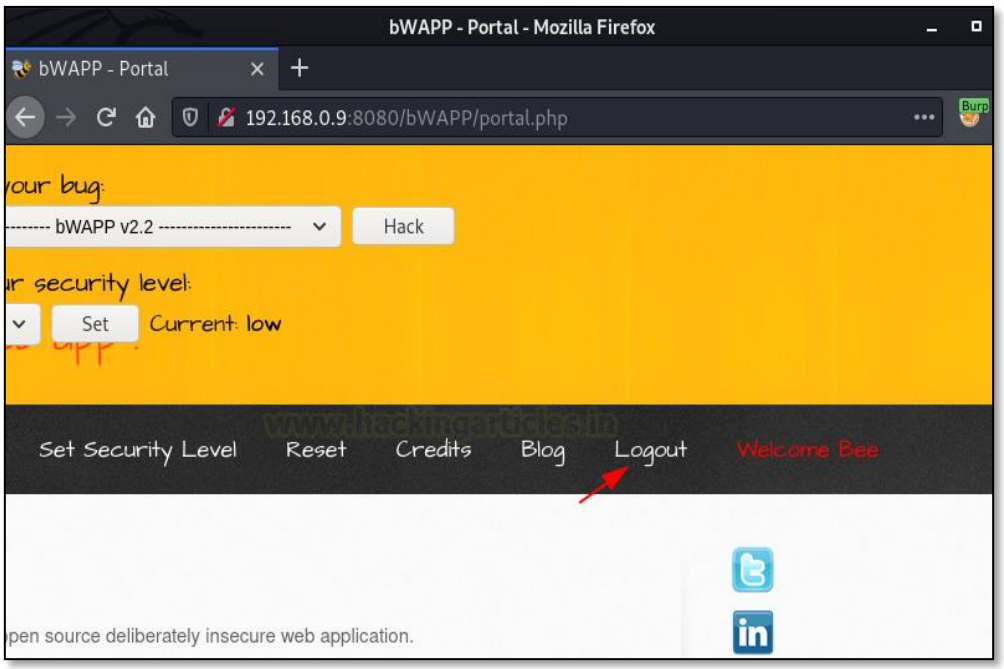
A number of sub-sections are available which thus could help us to analyze the application-properly, but being a pentester we just need to analyze the reliability of the results. Thereby, in order to learn more about it, check the Port Swigger's documentation from **here**.



Let's hit the **"Save Tokens"** button and dump the generated token values into **token.txt.**



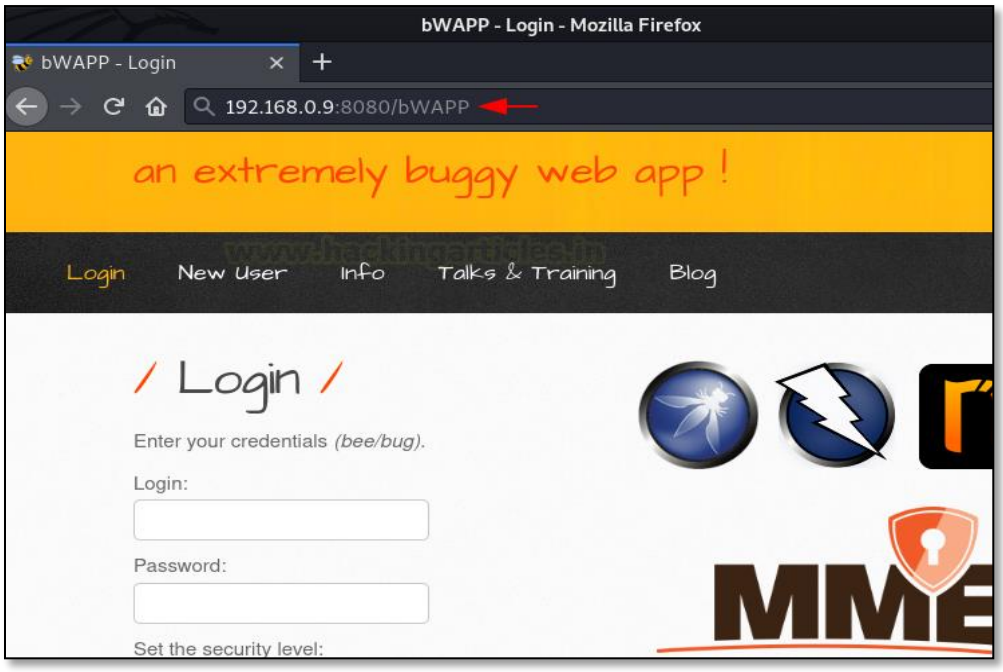
Till the file is saving in the background, let's get back to out bWAPP application and will **logout the user** such in order to end the session.



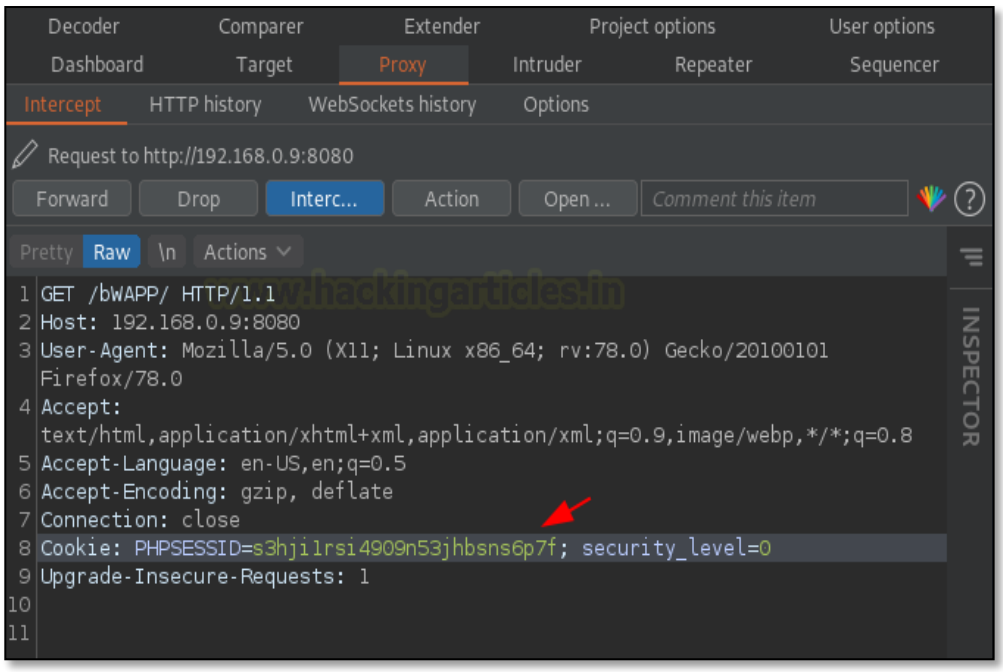
Now, as soon as we redirect back to the login page, we'll modify the URL within it i.e. setting it to

```
http://192.168.0.9:8080/bWAPP
```

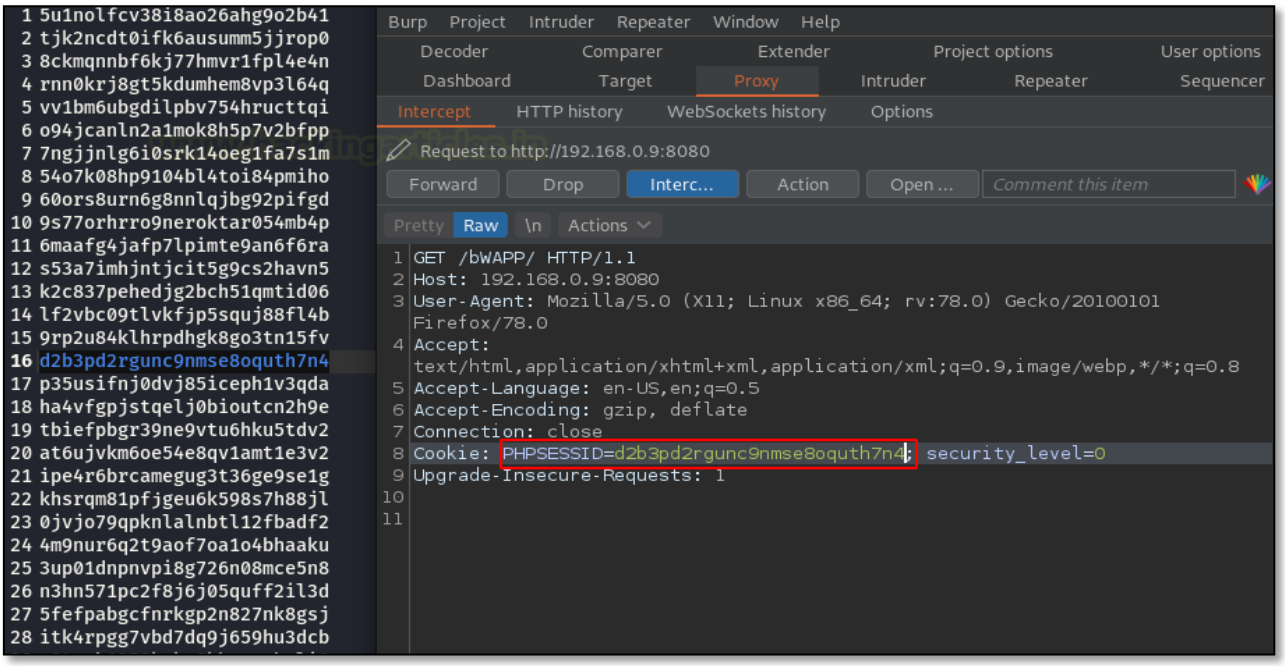
And we'll **capture the ongoing HTTP Request** over at our burpsuite.



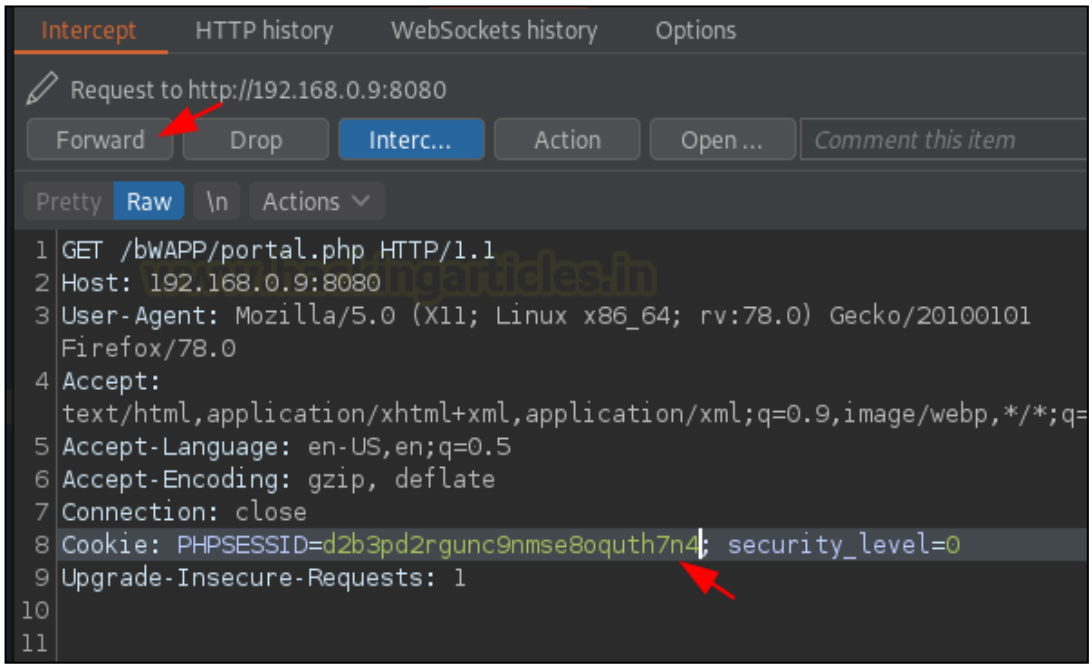
Back into our burpsuite's monitor, we can see that there is a session ID in the cookie header.



Let's manipulate the Session ID with one of our saved results.

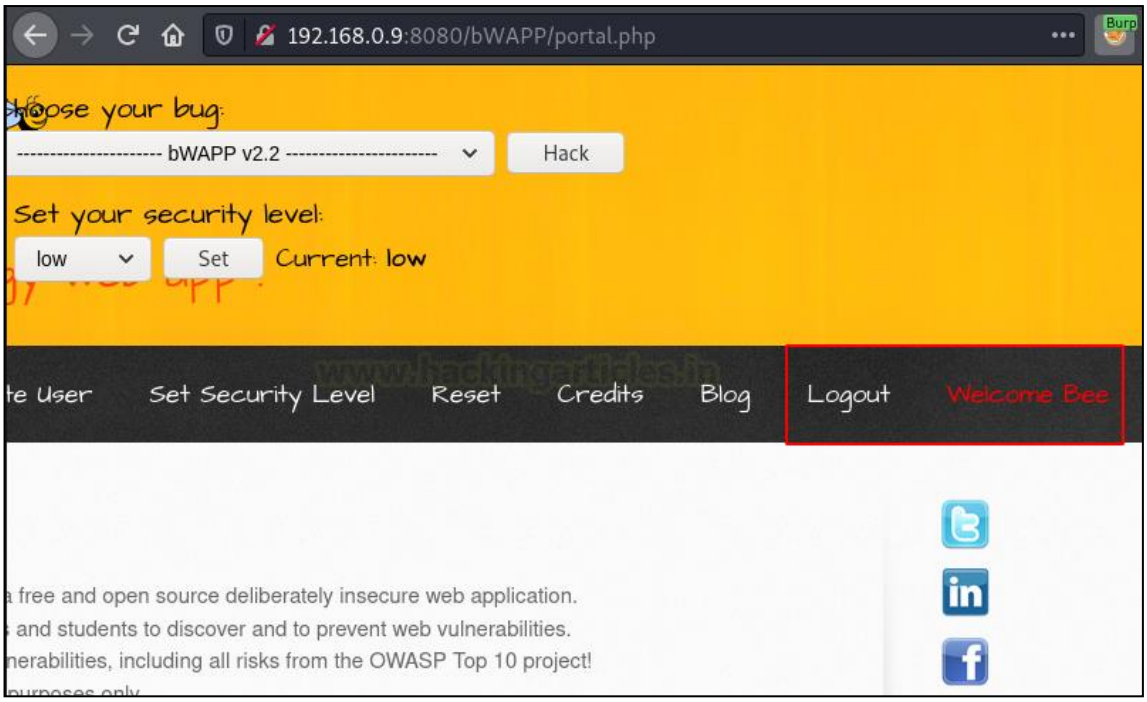


As soon as we hit the **forward button**, we got redirected to **portal.php** page and a session ID is required here too, let's manipulate it with the same.



And as we fire the button again, over at the browser we got logged in as **"Bee"** and this time it was without the credentials.

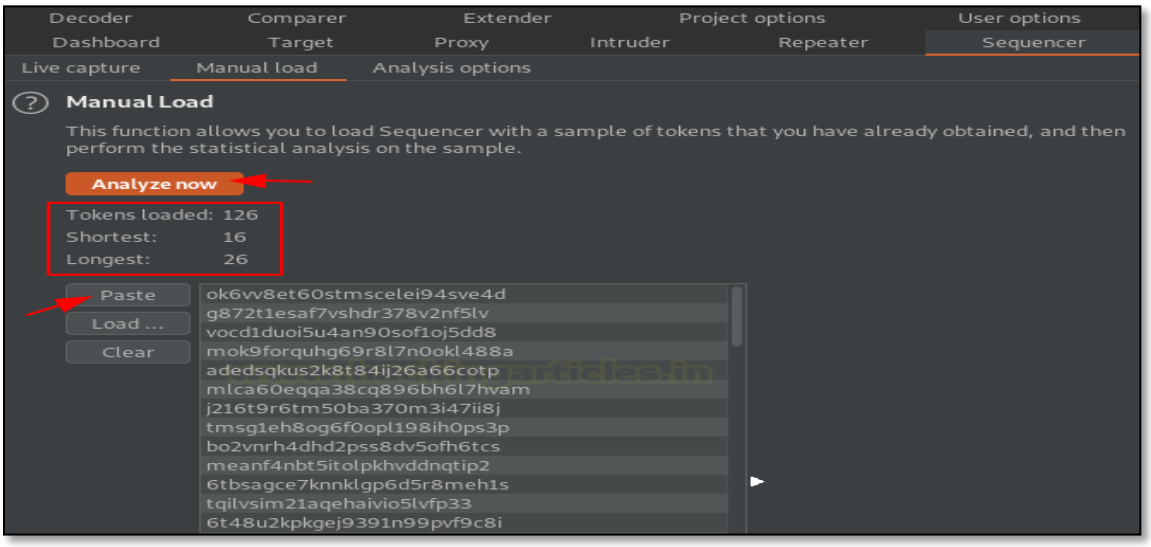
*There are times when the session ID that we manipulate might not valid, thus in such cases, we can use the entire **tokens.txt** with our **intruder** and will hunt for a successful login.*



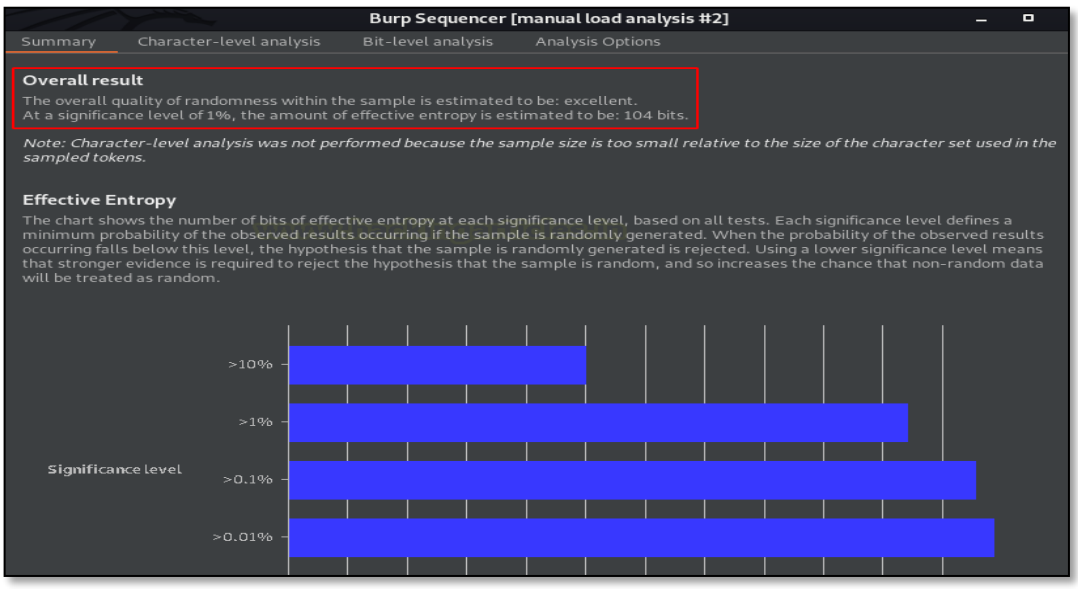
Manual Request Analysis

What, if we're not having any specific live web-application, but we're having a sample of tokens or Session ids and we want to analyze or depict its randomness?
Whether the sample is from a live-application or not, the burp's sequencer is always there for you to perform a **statistical analysis** such in order to determine the randomness.
Over at the dashboard, navigate to the Sequencer tab and move to the **Manal load** option, hit the **paste** button if the sample is in your clipboard or the **load** button if the tokens are within a specific file and then hit the **"Analyze now"** button to initiate the sequencer.

Note –
The sample or the number of tokens should be more than 100 in order to initiate the manual analysis.

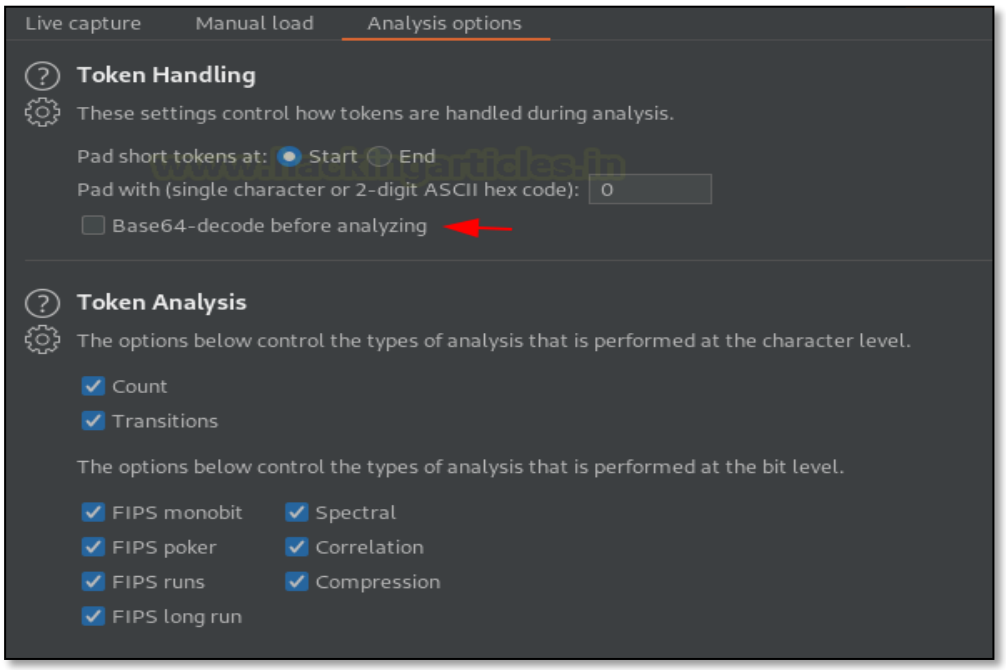


However, within a few seconds, we got the result dumped over in the new window as **"manual load analysis".**



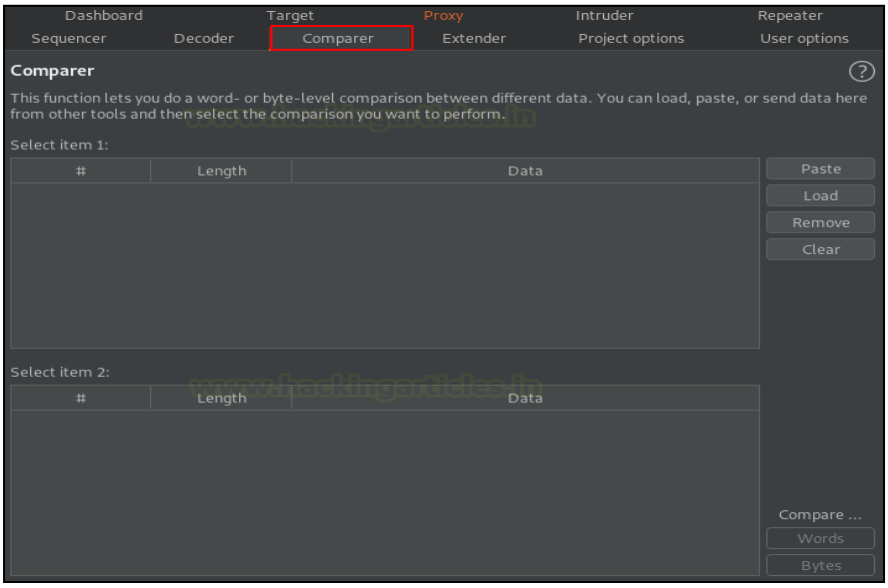
Aren't you wondering like what if the application's session ID or the token was bounded with base64 encoding scheme?

However, if such situations arise, the burp's sequencer is there for us, over at the **Sequencer's dashboard** as we switch to the **analysis option** tab we just need to enable the **"base-64 decode before analyzing"** option and we're happy to go.

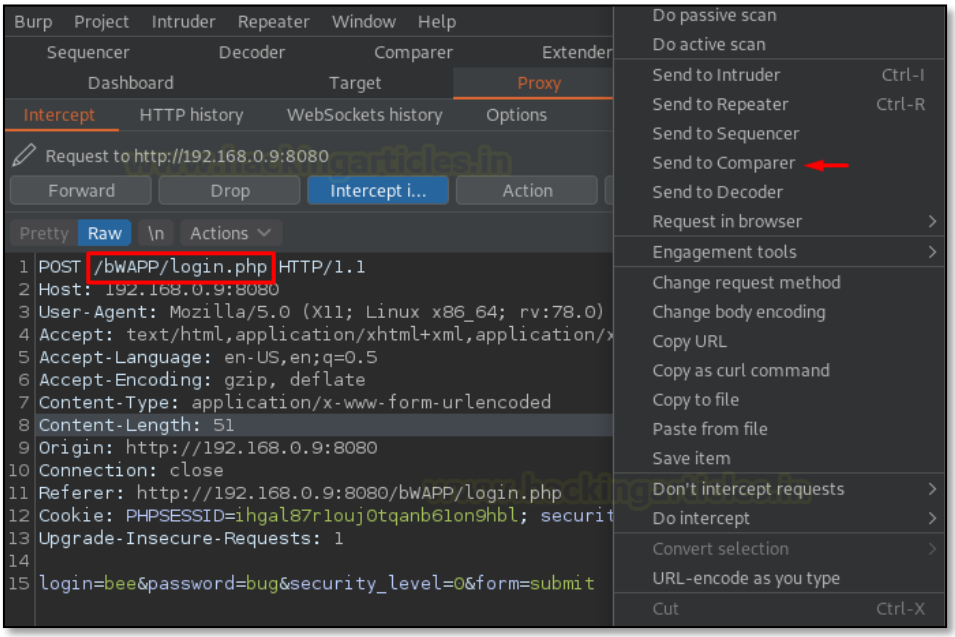


Comparing the Captured Tokens

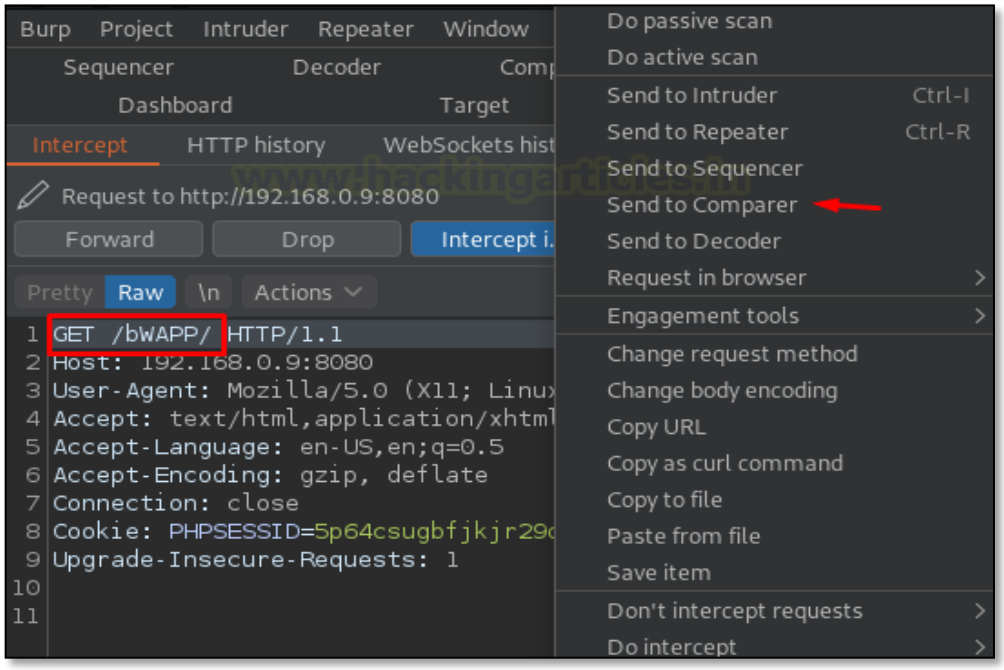
Burp Comparer is a tool to perform a comparison between two requests or responses against one another, this somewhere helps us to analyze the different responses in a way simpler.
However, this tool is the most friendly as it works with almost every other burp's section whether it's with proxying the requests, or fuzzing with the intruder, or capturing response with the repeater.
Being the most helpful, it thereby has its own space at the burp's panel. Simply open your burp suite and you'll find it right in front of you.



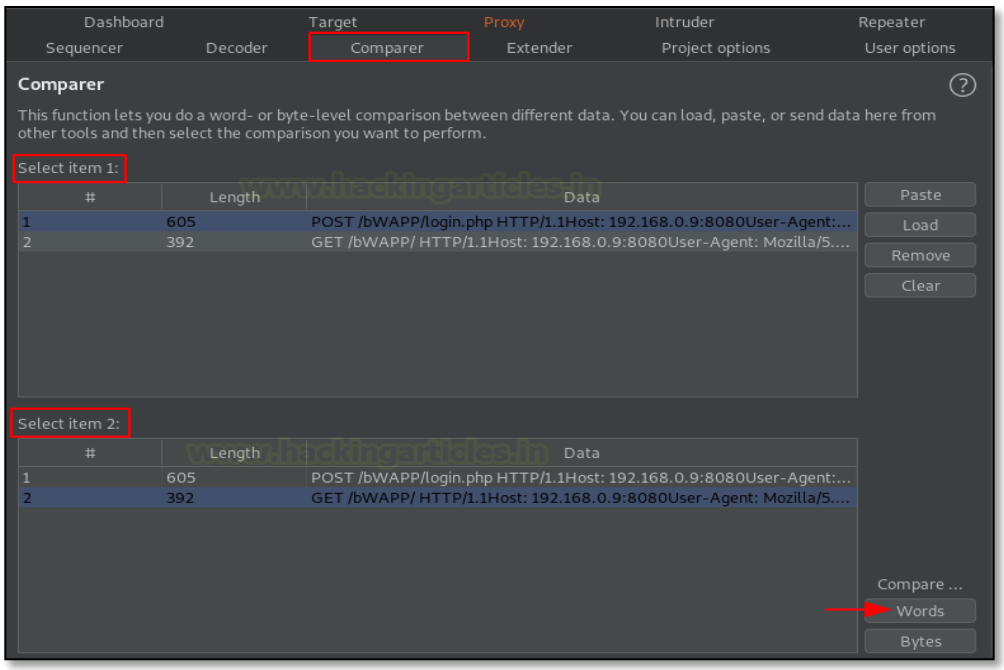
So, let's use this Comparer tool with our Proxy tab and will compare the two intercepted requests captured within it.
Simply do a right-click near the white space of the captured request and share it with the comparer.



Further, intercept the next request and share the same.



Now switch to the comparer tab from the panel. Over within it, we'll see that both of our requests are aligned within their specific regions as **Item 1** & **Item 2.**
However, we can even paste the request or response directly from our clipboard by hitting the **paste button** or either we could have loaded the files simply by firing the **load button.**
The comparer tab offers us two options, either a word-by-word compare or by a bit-by-bit one. Let's initiate with the **Words** one.



f

As soon as we select the comparison option, we got a new window popped up displaying both the requests in-front and highlighting the **Modified, Deleted** & the **Added** keywords.
Over the right-bottom, a check-box as **Sync Views (**if enabled**)** will help us to analyze and scroll the two requests or responses simultaneously where the content within them is a bit lengthy.

From the below image, we can determine that the session ID's are unique and different throughout the web-application.

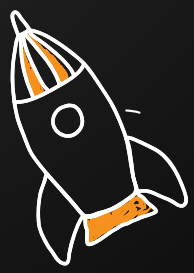
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

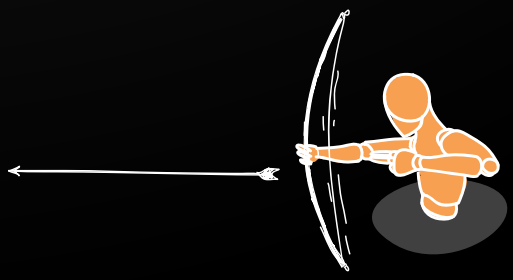
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



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