

# BURPSUITE FOR PENTESTER



```
linecount = "v" + id;
printf("Int(rc)");
console write(count) =
{
  printf("id" count
}
case if read(C1 = code
assigned int.i);
sizeof(int
ss; ss, =
sh, p, = c
```

```
int i = 6;
copy flinecount(); <1;
sizeof( innt)(<1=t;
trapp = linecount+;
cdet == tol;
return c= s"o";
printf("rezo_ac);
```

## Active Scan++

[www.ignitetechnologies.in](http://www.ignitetechnologies.in)



## Contents

Introduction .....	3
Exploring & Initializing Active Scan++ .....	3
Installing Active Scan++ from the BApp Store.....	3
Enhancing the Audit Functionalities .....	6
Audit the application.....	7
Auditing specific Injection Points .....	9
Defining Insertion Points in Burp Intruder .....	10
Scan Results and Vulnerability Insights.....	12



## Introduction

*Using Burp Suite as an automated scanner? Wondering right, even some pentesters do not prefer it, due to the fewer issues or the vulnerabilities it carries within. But what, if the burp scanner itself could identify the least common vulnerabilities along with core findings.*

So, today in this article we'll explore one of the most popular burp plugins "**Active Scan++**" which thereby merges up with the burp's scanner engine in order to enhance its scanning capabilities to identify the additional issues within an application.

## Exploring & Initializing Active Scan++

*Advanced vulnerabilities require advanced scanning techniques.*

Thereby, Active Scan++ one of the most of most popular burp's extension designed for the **Burp's Professional users** by "**James Kettle**" in order to improvise the burp's active and passive scanning capabilities.

However, this plugin gets integrated within the burp scanner such that it could help in the issue discovering part for the **Host Header Attacks, Password Reset Poisoning, Cache Poisoning, DNS Rebinding, XML Injection, Arbitrary Header Injection, Template Injeciton, Blind Code Injection** and the list goes on.

Moreover, this plugin also identifies the insertion points for **HTTP Basic Authentication**.

### Installing Active Scan++ from the BApp Store

Being so much effective, so let's find it out at the **bApp** store first. And we know where to navigate for that. Over at the **Extender** section, switch to the **bApp store** and then you'll find this tool at the top with the highest rating.

Name	Installed	Rating	Popularity	Last updated	Detail
.NET Beautifier		☆☆☆☆☆	— —	23 Jan 2017	
Active Scan++		☆☆☆☆☆	— —	11 Dec 2020	Pro extension
Add & Track Custom Issues		☆☆☆☆☆	— —	03 Mar 2020	Pro extension
Add Custom Header		☆☆☆☆☆	— —	08 Jul 2020	
Additional CSRF Checks		☆☆☆☆☆	— —	14 Dec 2018	
Additional Scanner Checks		☆☆☆☆☆	— —	21 Dec 2018	Pro extension
Adhoc Payload Processors		☆☆☆☆☆	— —	06 Nov 2019	
AES Payloads		☆☆☆☆☆	— —	28 Aug 2015	Pro extension
Anonymous Cloud, Configurat...		☆☆☆☆☆	— —	11 Sep 2020	Pro extension
Anti-CSRF Token From Referer		☆☆☆☆☆	— —	28 Feb 2020	
Asset Discovery		☆☆☆☆☆	— —	12 Sep 2019	Pro extension





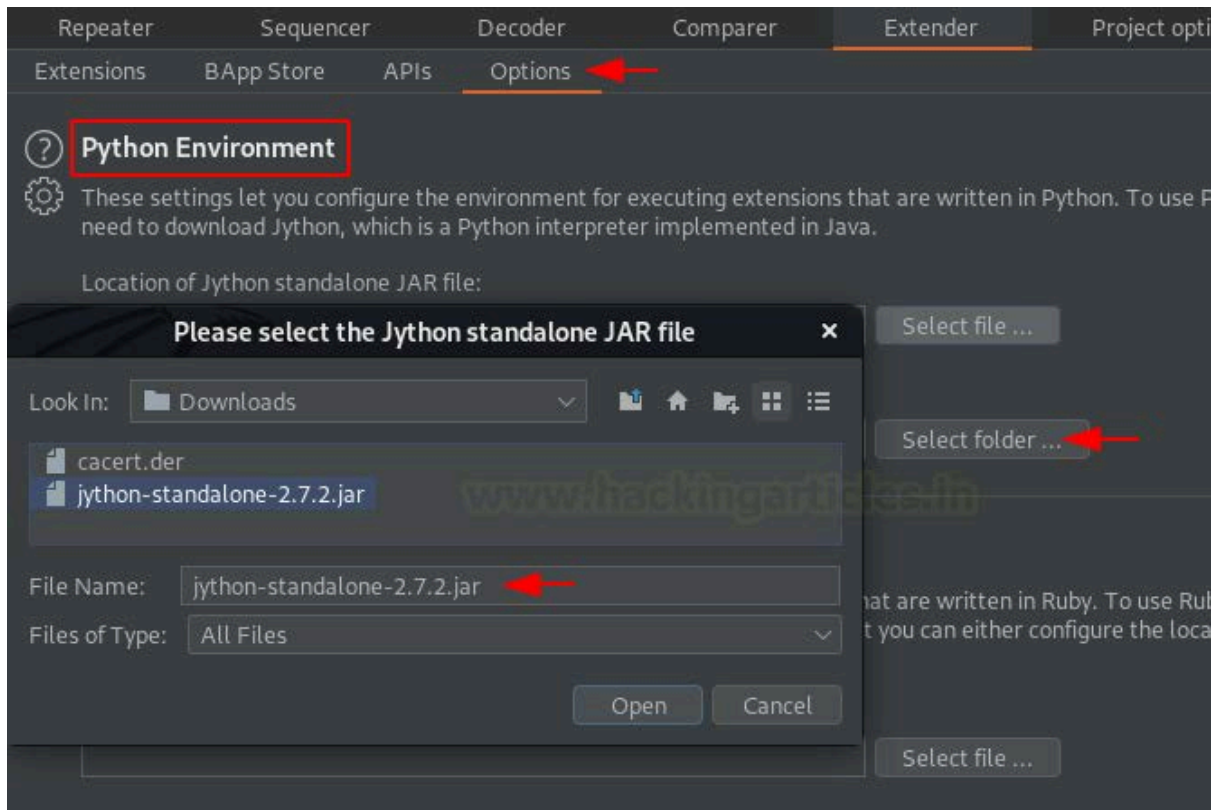
Let's now switch to the left panel in order to identify the **Install** button. But wait !! It requires Jython, so let's install and configure that first.

The screenshot shows the Burp Suite interface with the 'Extender' tab selected. The 'BApp Store' is open, displaying a list of extensions. The 'Active Scan++' extension is highlighted. The details for this extension are shown on the right, including its author (James Kettle), version (1.0.21d), source (https://github.com/portswigger/active-scan-plus-plus), and update date (11 Dec 2020). The 'Install' button is highlighted with a red box. Below the 'Install' button, there is a 'Download Jython' button, which is pointed to by a red arrow.

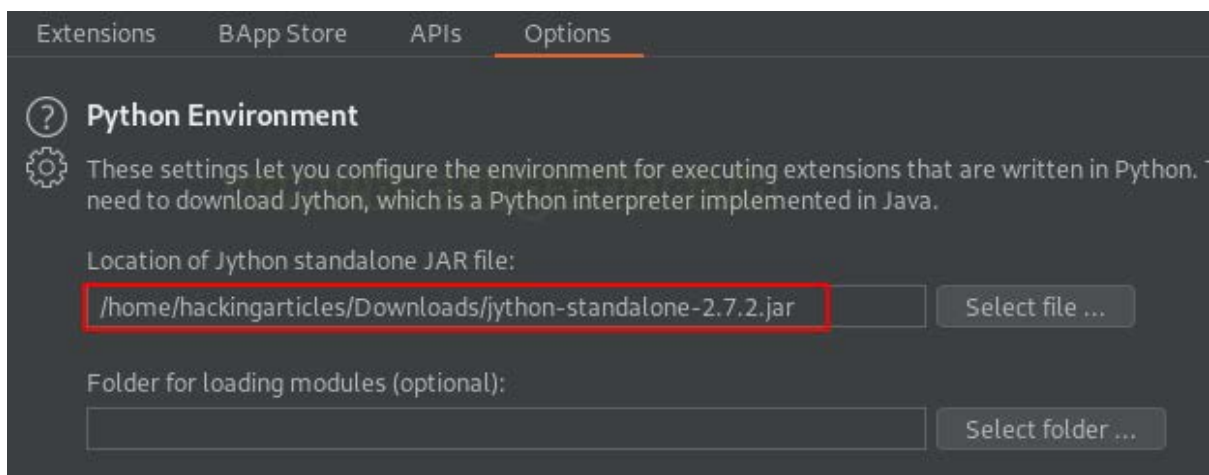
Head to Jython's official website, download the **Standalone Jython's** file.

The screenshot shows the Jython official website. The browser address bar displays 'https://www.jython.org/download.html'. The page has a navigation bar with links to Home, News, Download, Documentation, Development, and Links. The 'Current Version' section is highlighted, stating: 'The current version of Jython is 2.7.2 It can be downloaded here:'. Below this, there are three bullet points: 'Jython Installer - Use this to install Jython. (metadata)', 'Jython Standalone - Use this to run Jython without installing or to embed Jython in a Java application. (metadata)', and 'You may cite Jython 2.7.2 as a dependency in your Maven or Gradle build.' A red arrow points to the 'Jython Standalone' link.

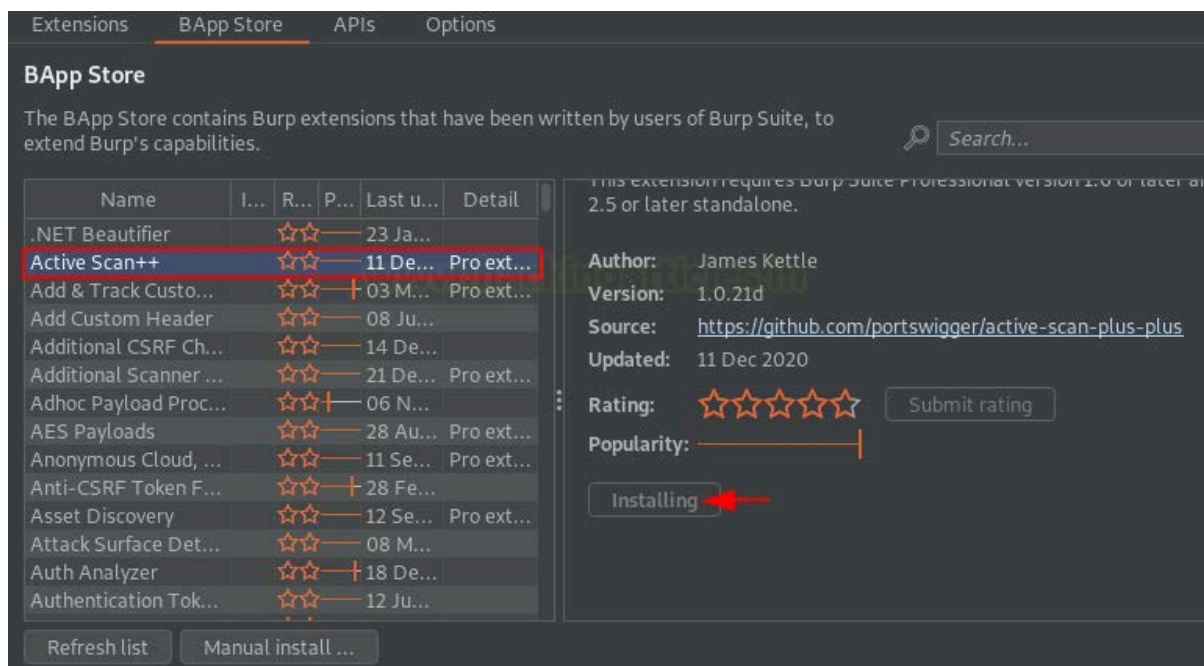
As soon as the file got stored up over at the local machine, we'll embed it with our Burp Suite application. Back at the extender tab, navigate to the **Options** section there and scroll down for the **"Python Environment"**, hit the **select file** button, and then opt for the downloaded file.



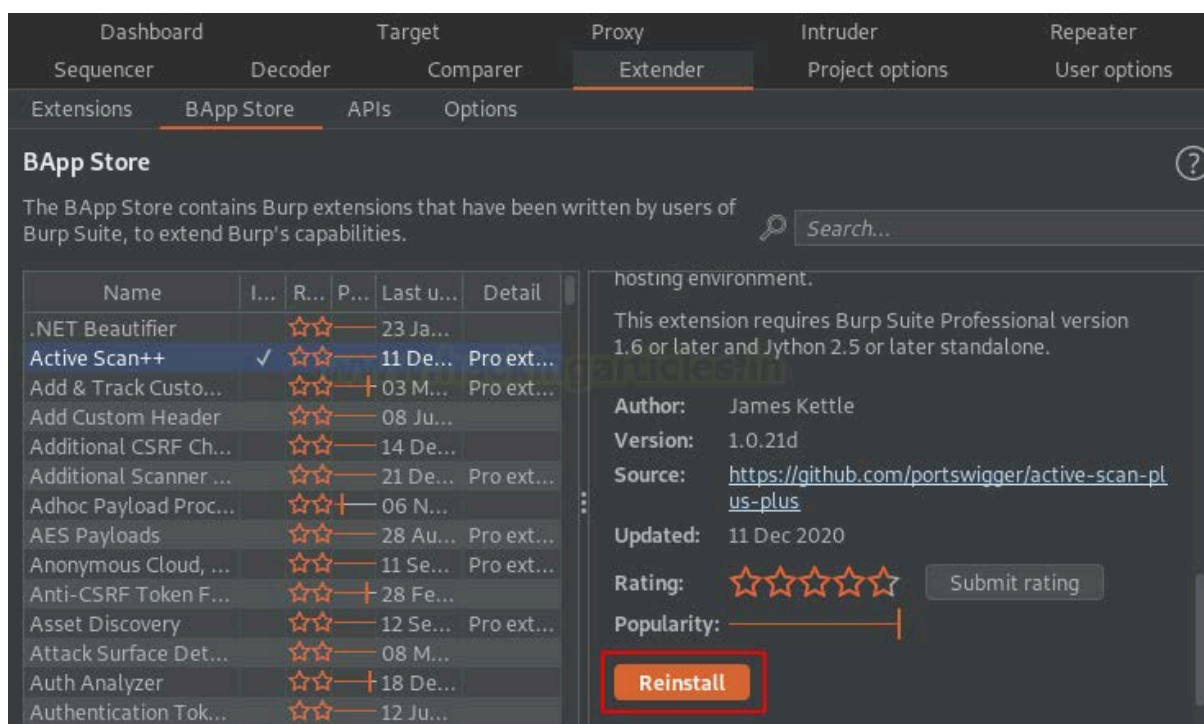
Once done with all this, you'll have your screen somewhat similar to the below image. But, the Jython's configuration is yet not over, **restart your burp** in order to get the changes reflected.



Now head back to the bApp store and open the **Active Scan++** right-side portal. From the below image you can see that the **Install** button is now active, let's fire it to initiate the installation.



Great!! We got the **Reinstall** button, seems like the extension had been set up successfully.



## Enhancing the Audit Functionalities

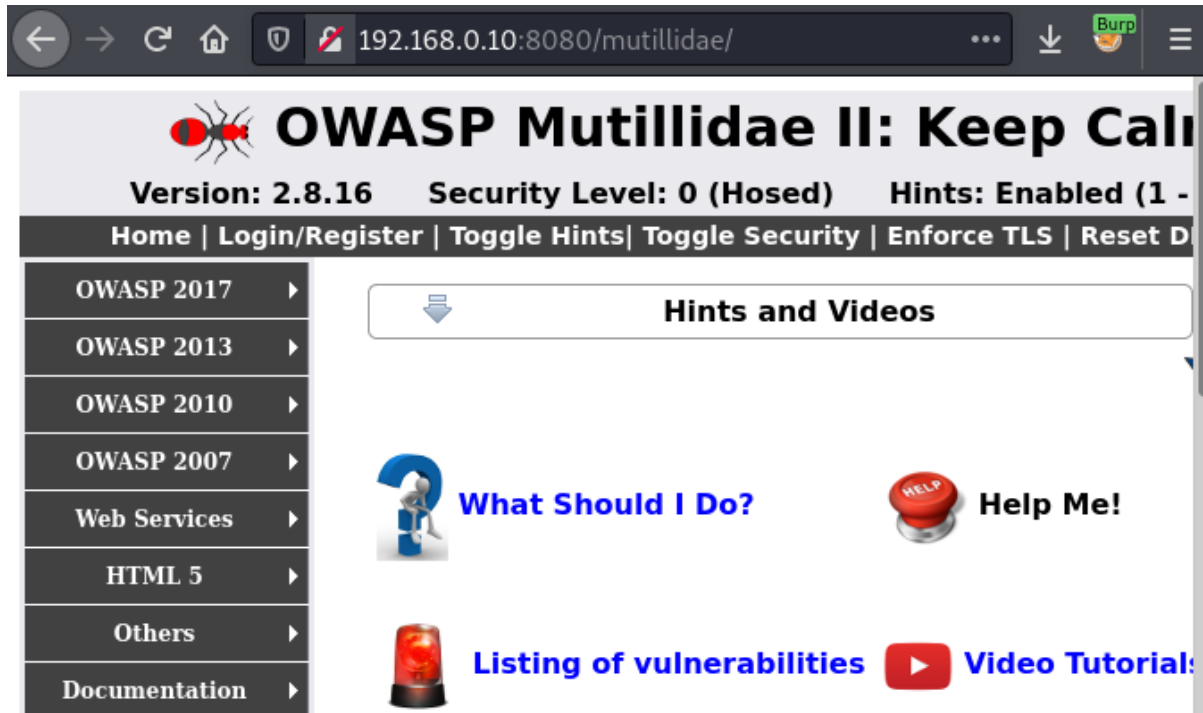
You might be wondering, like being the most popular, Active Scan++ should have its own place at the top panel, so where is it?

As discussed earlier that **Active Scan++ integrates with the burp's scanner** such in order to assist it to identify additional vulnerabilities. Thereby, we do not have any specific location to find it. However, we can analyze its working while performing an active or passive scan.

So, let's see what additional it dumps out when we initiate a scan over at the entire application.

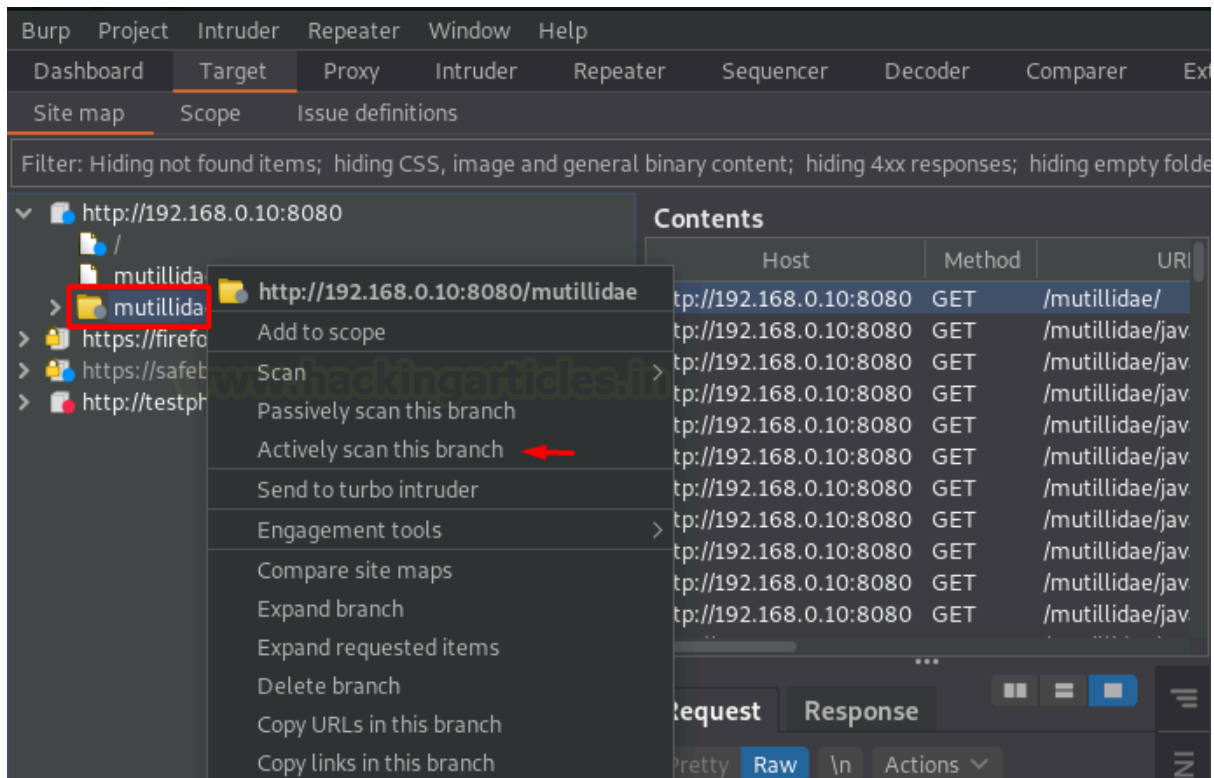
## Audit the application

Turn **On** the Browser's **Proxy Service** and then surf the OWASP's Mutillidae vulnerable application.



Now, let's navigate to the **Target** option over at our burp suite monitor, further head to the **Site map**.

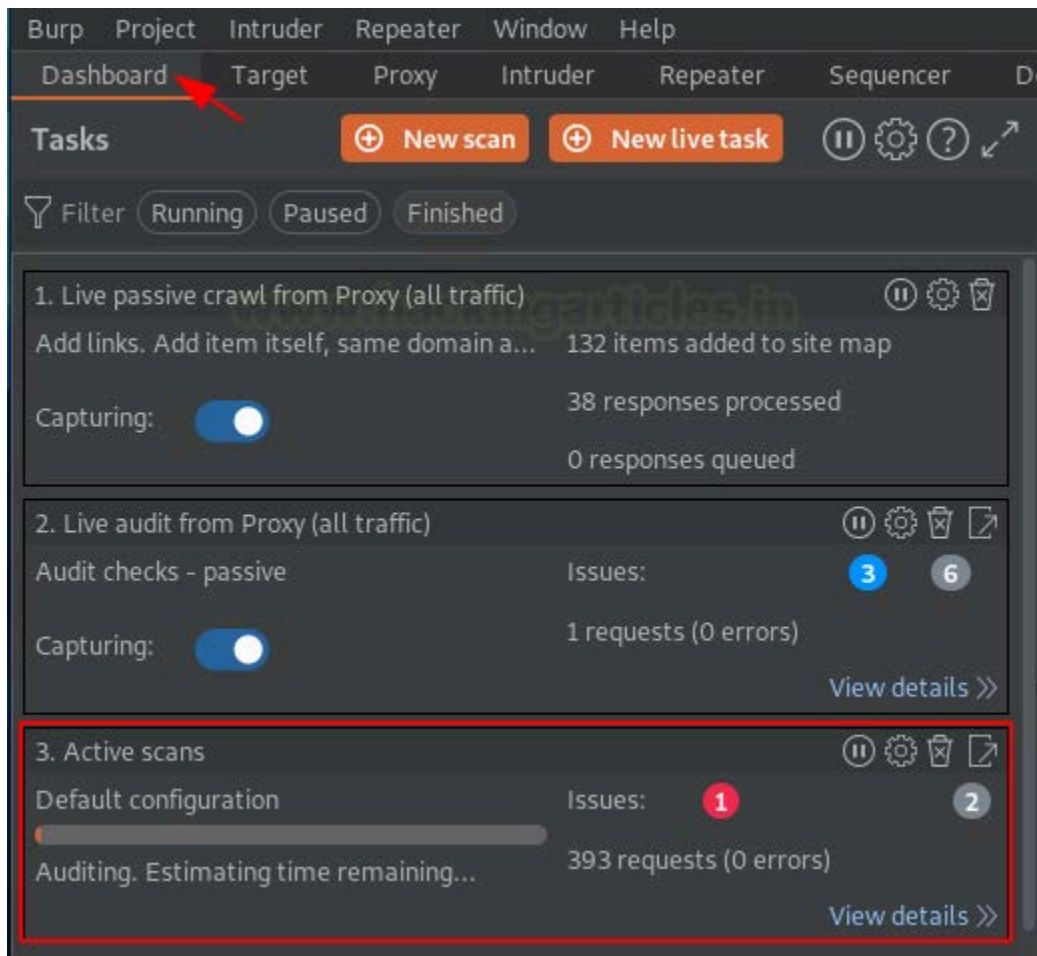
The Left panel carries up the web application's hierarchy, let's opt the **root branch**, and then we'll hit a right-click to opt "**Actively scan this branch**" such in order to share the application for the scanning part.







As soon as do so, we got our auditing task aligned at the “Dashboard”.



And within a few minutes, you can see a number of issues segregated at the right panel, let's check them out.

*You might find most of the issues discovered with the burp's scanner but there are some like cache poisoning, DNS rebinding, Host header Injection, all these additional ones are identified with the Active Scan++.*





**Issue activity**

Filter: High Medium Low Info Certain Firm Tentative Search...

#	Task	Time	Action	Issue type
19	3	21:12:54 5 Jan 2021	Issue round	Cross-domain Referer leakage
18	3	21:12:54 5 Jan 2021	Issue deleted	Cross-domain Referer leakage
17	3	21:12:54 5 Jan 2021	Issue found	Cross-domain POST
16	3	21:12:54 5 Jan 2021	Issue found	Cross-domain Referer leakage
15	3	21:12:54 5 Jan 2021	Issue found	Cross-domain Referer leakage
14	3	21:12:54 5 Jan 2021	Issue found	Cross-domain POST
13	3	21:12:54 5 Jan 2021	Issue found	Frameable response (potential Clickjackin
12	3	21:12:05 5 Jan 2021	Issue found	HTTP TRACE method is enabled
11	2	21:09:35 5 Jan 2021	Issue found	Cross-domain POST

Advisory Request 1 Response 1 Request 2 Response 2 Request 3 Response 3

Issue: **Cross-domain Referer leakage**  
Severity: **Information**  
Confidence: **Certain**  
Host: **http://192.168.0.10:8080**  
Path: **/mutillidae**

**Issue detail**

The application contains the following link to another domain from URLs containing a query string:

- <http://www.youtube.com/user/webpwnized>

This issue was found in multiple locations under the reported path.

## Auditing specific Injection Points

What if, you don't want to test the entire application's branch neither a specific web-page, but you want an injection point to be audited and if it's possible then does **Active Scan++** still collaborates with the burp scanner?

As soon as we hit the install button at the bApp store, the very first-second Active Scan++ bound itself with Burp's Scanner. So, whether it's about auditing the entire application or a specific injection point, Active Scan++ always involves itself within it.

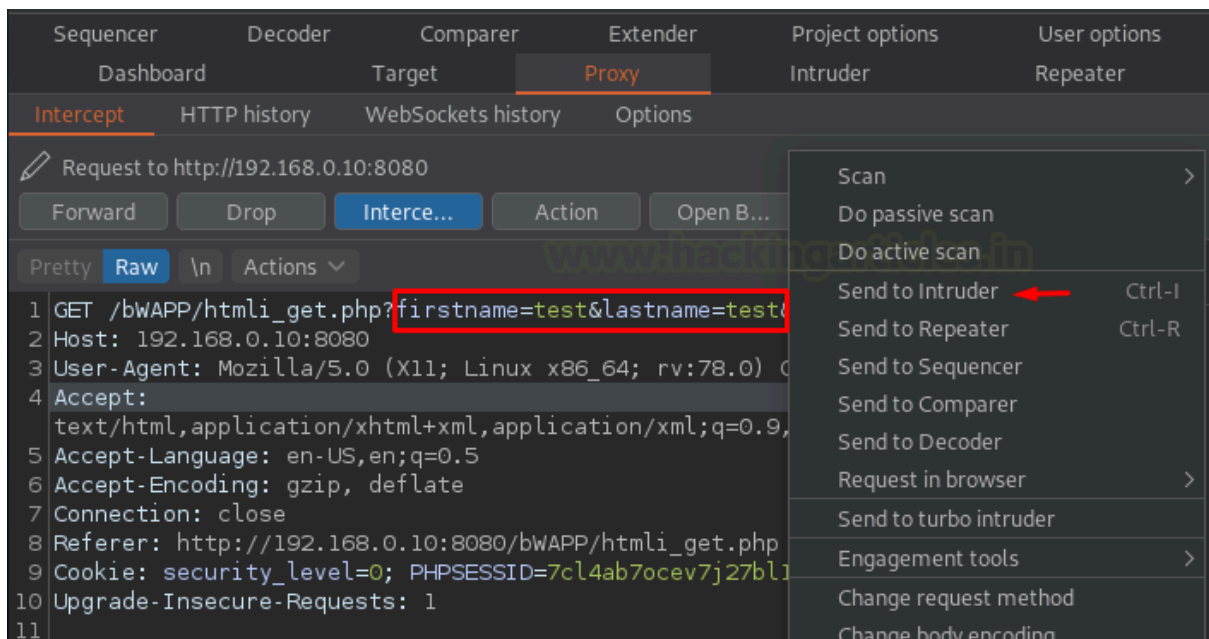
And about the Injection point auditing, so let's do that.

Initiate the bWAPP application with **bee: bug** and then navigate to the **HTML Injection (Reflected)** webpage. Further, we'll enter some test credentials and hit the **Go** button with our **Proxy Service** turned **"ON"**

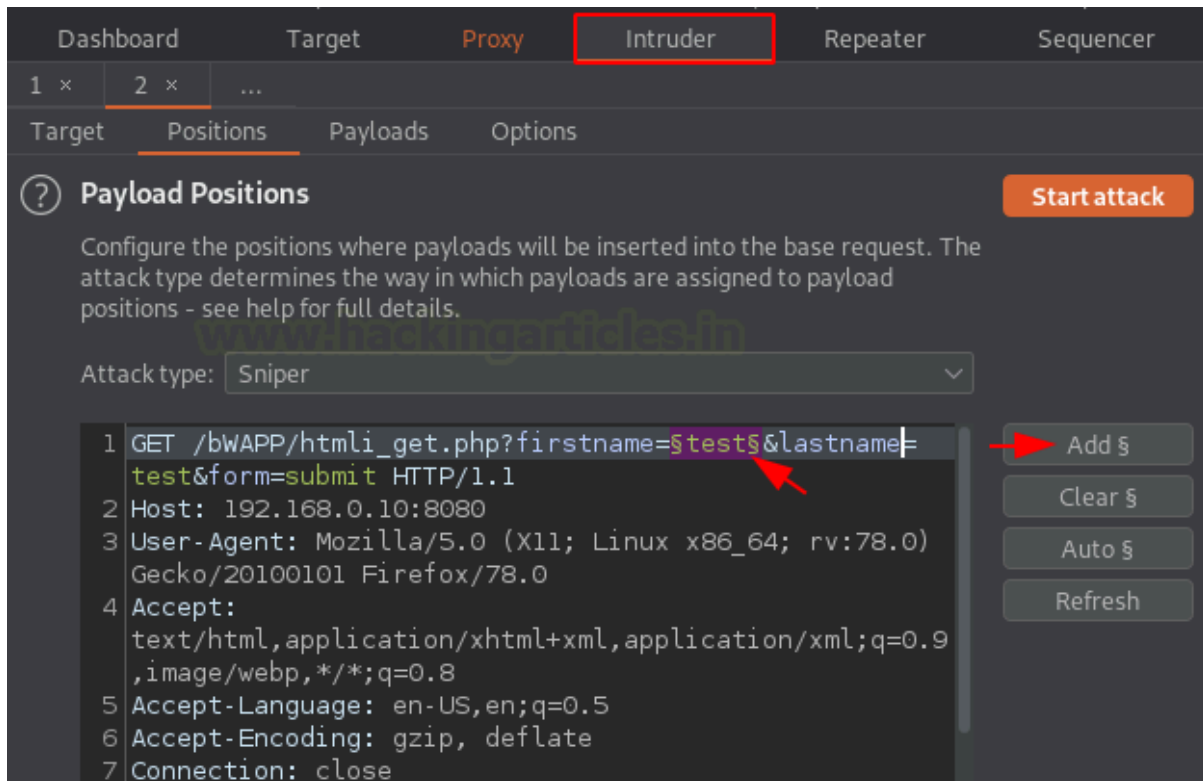


### Defining Insertion Points in Burp Intruder

Over at our burp suite monitor, we got the ongoing HTTP request captured, let's share it with the intruder.



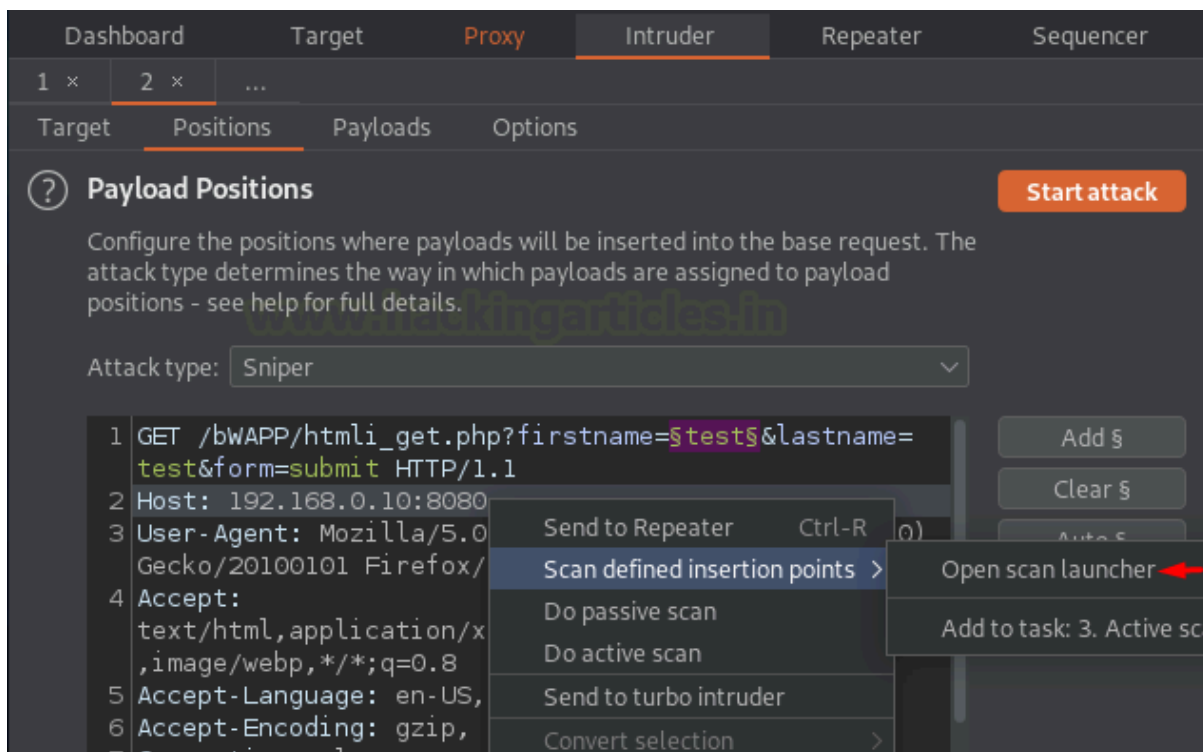
Once done, let's navigate to the **Position** section over at the intruder tab in order to set the injection point. Simply clear the default selections and add the one you want the scanner to audit.



What now? A simple right-click is always our helping hand. So, let's hit the mouse right button and then opt **"Scan defined insertion points"**.

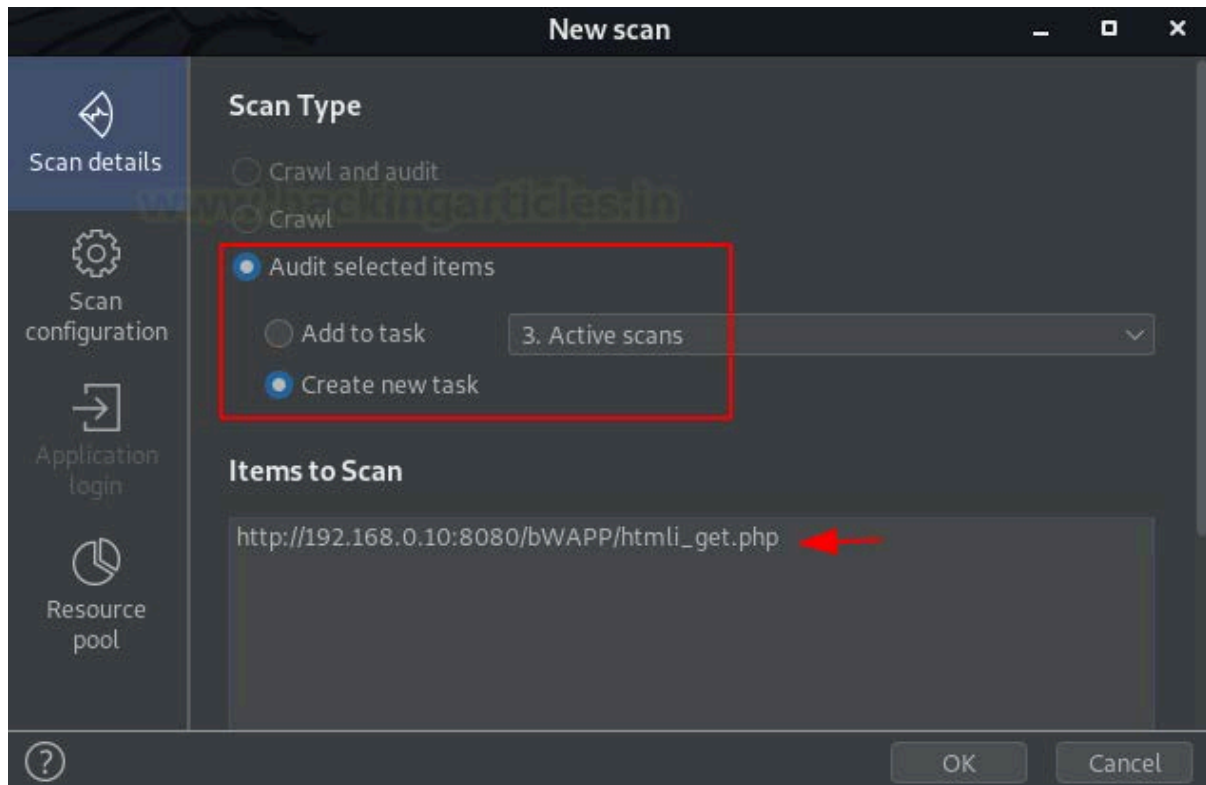
As we haven't deleted our previous task, thereby the burp drops two option either to send this request with the ongoing Audit or to start a new one for this.

Let's hit the **Open Scan launcher** in order to start and customize the new scan.





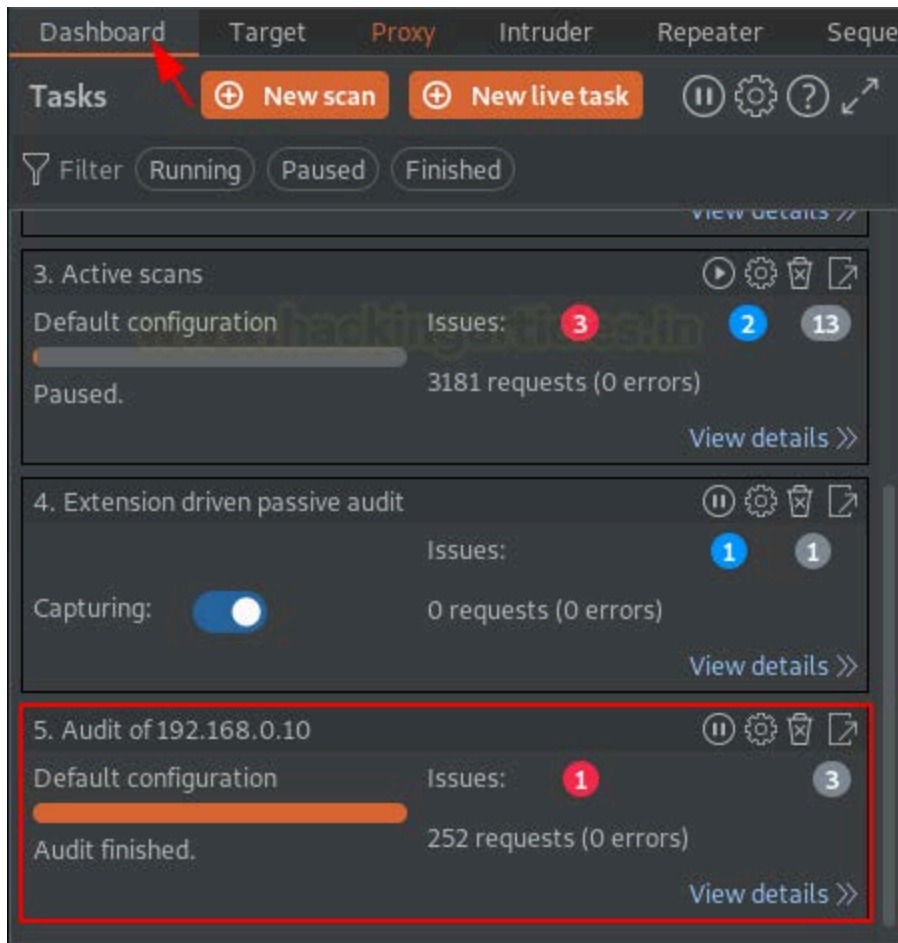
As soon as we do so, we got a window that popped up in front of us stating “**New Scan**”. Let’s move forward with the default configuration by hitting the **OK** button. Check our [previous article](#) if you want to customize your scanner accordingly.



### Scan Results and Vulnerability Insights

From the below image we can see that our scanner captured about **3 Basic** and **1 Critical issue** by sharing about 250+ requests at the application's injection point.





Time to analyze. From the below image, we can see that the burp scanner tested Cross-Site Scripting for our injection point and dumped the issue details with the exploiting payload and the mitigation steps.



### Issue activity

Filter

HighMediumLowInfoCertainFirmTentative

Search...

Action	Issue type	Host
Issue found	Path-relative style sheet import	http://192.168.0.10:8080 /bWAPP/htmli_get.php
Issue found	Cross-site scripting (reflected)	http://192.168.0.10:8080 /bWAPP/htmli_get.php
Issue found	Input returned in response (reflected)	http://192.168.0.10:8080 /bWAPP/htmli_get.php
Issue found	Open redirection (DOM-based)	http://192.168.0.10:8080 /mutillidae/
Issue found	Cross-domain Referer leakage	http://192.168.0.10:8080 /bWAPP/htmli_get.php
Issue found	Cross-site scripting (reflected)	http://192.168.0.10:8080 /mutillidae/hints-page-
Issue found	Input returned in response (reflected)	http://192.168.0.10:8080 /mutillidae/hints-page-
Issue found	SQL injection	http://192.168.0.10:8080 /mutillidae/hints-page-

Advisory

Request

Response

Cross-site scripting (reflected)

Issue:

Cross-site scripting (reflected)

Severity:

High

Confidence:

Certain

Host:

http://192.168.0.10:8080

Path:

/bWAPP/htmli\_get.php

Issue detail

The value of manual insertion point 1 is copied into the HTML document as plain text between tags. The payload `cx0th<script>alert(1)</script>j2vf1` was submitted in the manual insertion point 1. This input was echoed unmodified in the application's response.

14 | Page

# JOIN OUR TRAINING PROGRAMS

