

Burp Suite for Pentester

# Web Scanner & Crawler



## Contents

<b>The Burp's Crawler .....</b>	<b>3</b>
What is Crawler? .....	3
Customizing the Crawler .....	9
Configuring Out of Scope URL's .....	9
<b>Vulnerability Scanning Over Burpsuite .....</b>	<b>14</b>
Auditing with the default configuration .....	15
Defining Audit Configurations.....	20
<b>Crawling &amp; Scanning with an Advanced Scenario .....</b>	<b>26</b>
<b>Deleting the Defined Tasks.....</b>	<b>27</b>

## The Burp's Crawler

### What is Crawler?

The term **web-crawler** or **web-spider** is the most common and is been used several times while testing a web application. So, what this crawler is??

Carrying with its name we can depict that a crawler **surveys a specific region** slowly and deeply and then drops down the output with a defined format.

So, is the Burp's Crawler the same thing??

According to port swigger "The crawl phase involves navigating around the application, following links, submitting forms, and logging in, to catalog the content of the application and the navigational paths within it."

In simpler words, we can say that the burp crawler programmatically moves within the entire web application, follows the redirecting URLs, logs inside the login portals, and then adds them all in a **tree-like structure** over in the Site Map view in the **Target tab**.

However, this crawler functions as similar to the "Dirb" or the "DirBuster" tools – the web content scanners, which brute-force the web-server to dump the visited, non-visited, and hidden URLs of the web application.

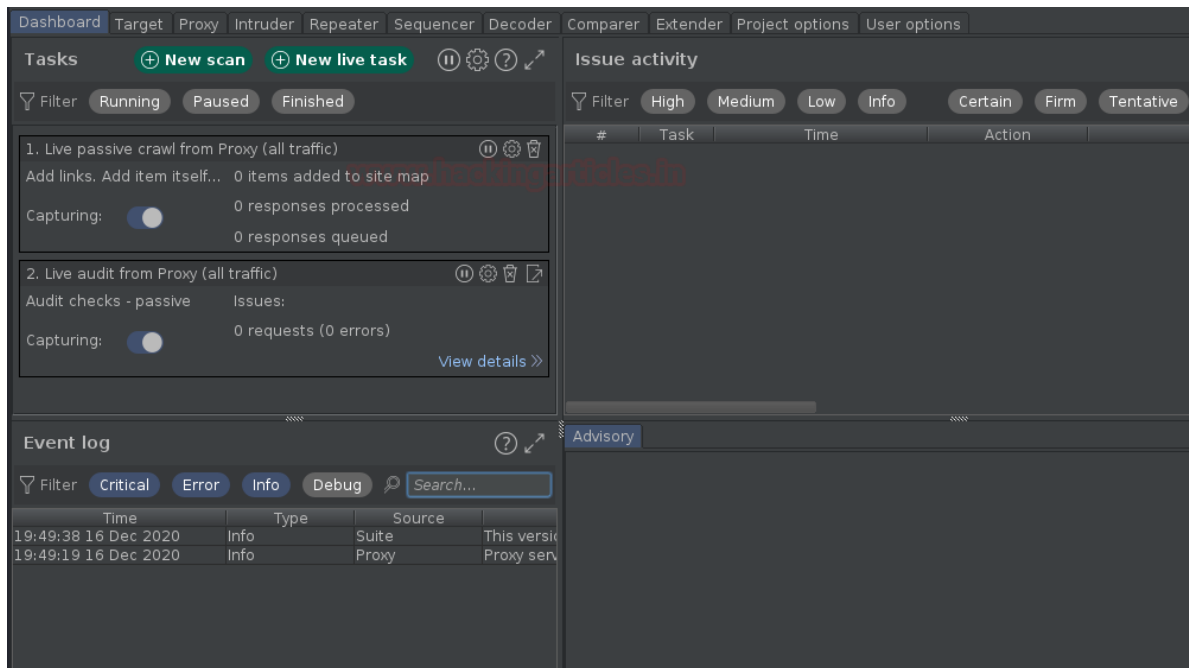
Earlier over in the previous versions of burp suite say **"1.7"**, we got this crawler termed as **"Spider"**. So why does this happen, what new features did the burp crawler carries that made the spider vanish off??

### Let's dig it out!!

#### Crawl with default configurations!!

If you're familiar with the spider feature, then you might be aware, that, the spider holds up a specific tab within the burpsuite's panel. But with the enhancements, the burp's crawler comes pre-defined within the **dashboard section**. However, it thus helps us to monitor and control the burp's automated activities in a single place.

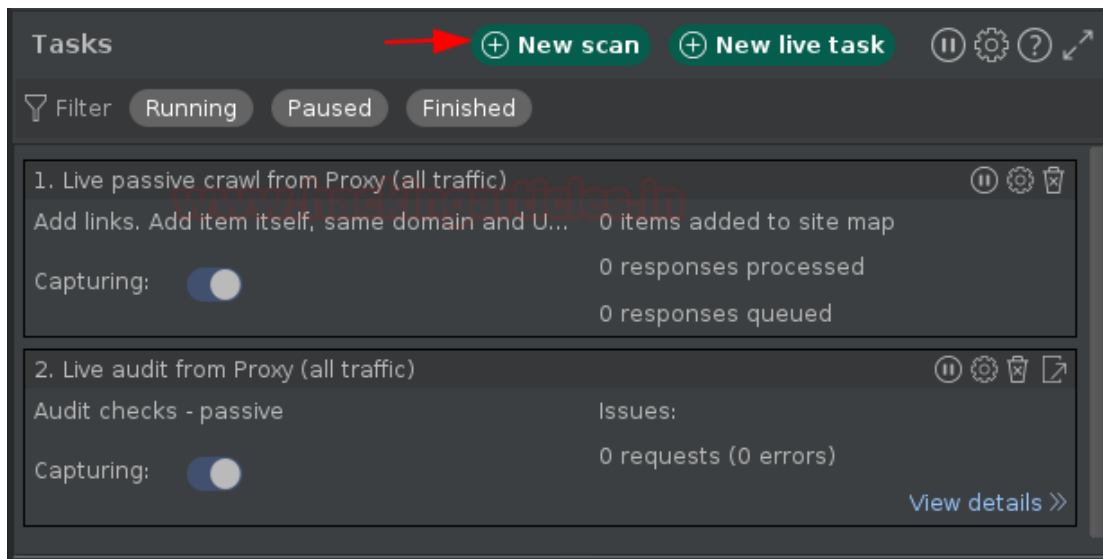
So, in order to initiate with the crawler, let's turn ON our burp suite and redirect to the Dashboard section there.



As soon as we land at the dashboard panel, we can see the number of subsections specified. Let's explore them in detail:

1. **Tasks** – The “Tasks” section carries the summary of all the running crawls and scans, whether they are user-defined or automated ones. Here, we can pause and resume the individual tasks, or all tasks together, and even we can view the detailed versions of a specific crawl or audit too.
2. **Event log** – The Event log feature generates all the events that the burp suite follows like if the proxy starts up the event will be generated for it, or a specific section is not working properly, then an event log with the will be generated.
3. **Issue Activity** – This section drops out the common vulnerabilities within the application that the burp suite scans up and further we can segregate them all by applying the defined filters according to their severity and destructiveness.
4. **Advisory** – This is one of the most important sections of the burp's dashboard as it demonstrates the selected vulnerability in the expanded form such by defining the payload with a Request & Response, mentioning the cause of its existence, defining the mitigation steps, and dropping the reference and the CVSS Scores for our review.

Thereby, to dig web-application we need to hit the “**New Scan**” button placed at the top of the **Tasks** section.



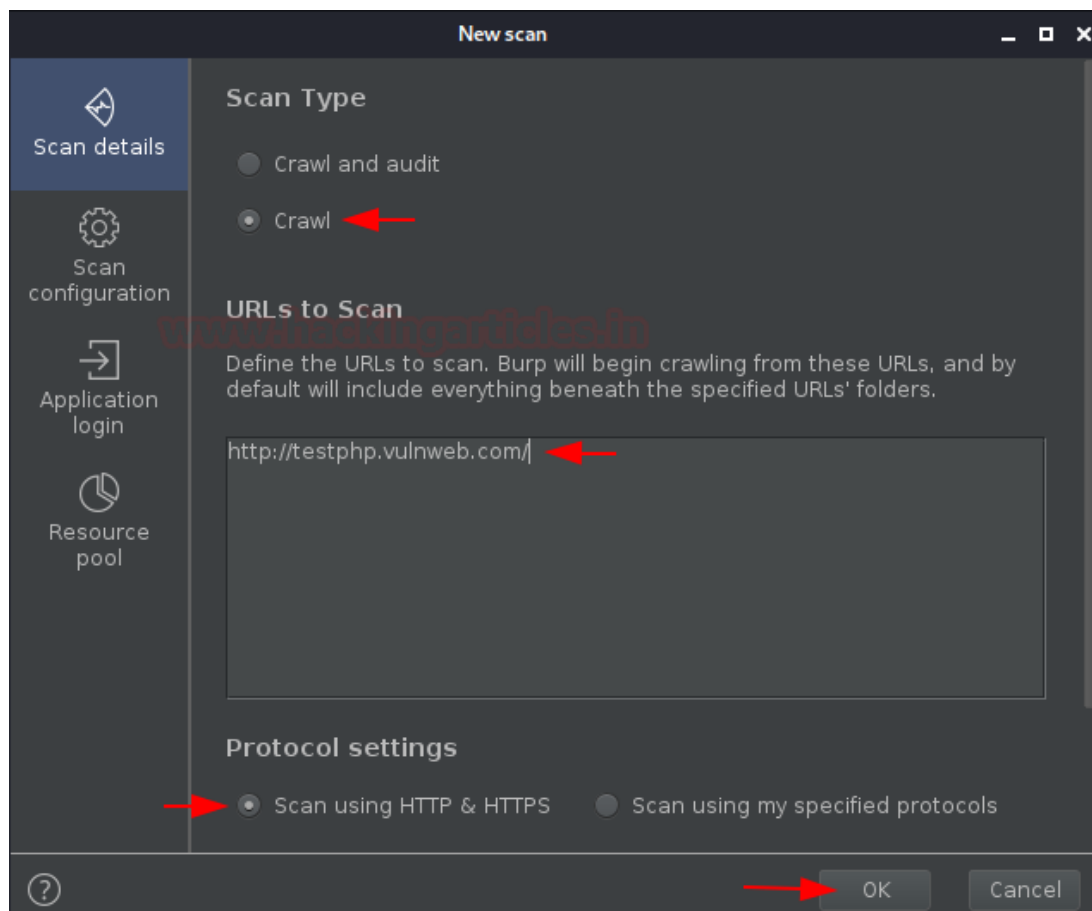
As soon as we do so, we'll be redirected to a new popped-up window stating "**New Scan**".

There we'll be welcomed with two options –

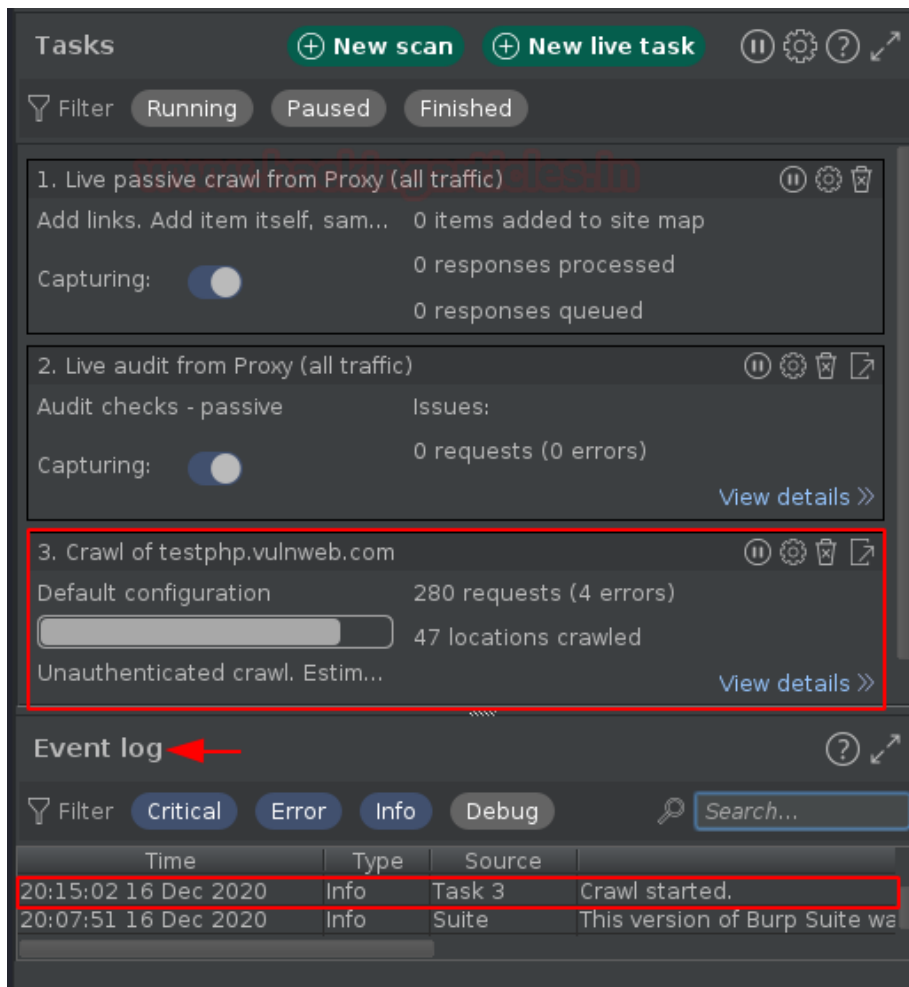
- Crawl & Audit
- Crawl

However, for this section, we'll make it to "**Crawl**" only. And the other one, we'll discuss later in this article.

As we're heading with the default configurations thus we'll simply type the **testing URL** i.e. "http://testphp.vulnweb.com/" and will hit the "**OK**" button.



As we do so, the window will get disappeared and over in the dashboard we'll get our new task aligned as **"Crawl of test.vulnweb.com"**, and in the event log, we can see that we got the event **"Crawl started"**.



Within a few minutes, the crawling task will get finished up and we'll get the notification there. *But where's the result??*

As defined earlier the crawler, dumps the result in a **tree-like format** in the **Site Map view** in the **Target tab**, let's move there.

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options xssValidator

Site map Scope Issue definitions

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

Contents

Host	Method	URL	Params	Status	Length
http://testphp.vulnweb.com	GET	/		200	5175
http://testphp.vulnweb.com	GET	/AJAX/index.php		200	4453
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/		200	1191
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/BuyPr...		200	316
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/BuyPr...		200	291
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/BuyPr...		200	308
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/Detail...		200	529
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/Detail...		200	535
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/Detail...		200	495
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/RateP...		200	316
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/RateP...		200	291
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/RateP...		200	308
http://testphp.vulnweb.com	GET	/artists.php		200	5545
http://testphp.vulnweb.com	GET	/artists.php?artist=1	✓	200	6468
http://testphp.vulnweb.com	GET	/artists.php?artist=2	✓	200	6410
http://testphp.vulnweb.com	GET	/artists.php?artist=3	✓	200	6410
http://testphp.vulnweb.com	GET	/cart.php		200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/categories.php		200	6332

Great!! We got what we desire. Over in the right panel we're having about almost **every URL of the webpage**, along with that, it carries up the **HTTP methods** and a **parameter section** that defines which URL requires a Params value within it.

Several major vulnerabilities exist due to the un-sanitized input fields thereby with this dumped data we can simply **segregate the URLs that contain the Input values** which thus can be further tested on. And for this simply double click the "Params" field.

Host	Method	URL	Params	Status	Length
http://testphp.vulnweb.com	GET	/artists.php?artist=1	✓	200	6468
http://testphp.vulnweb.com	GET	/artists.php?artist=2	✓	200	6410
http://testphp.vulnweb.com	GET	/artists.php?artist=3	✓	200	6410
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/cart.php	✓	200	5120
http://testphp.vulnweb.com	POST	/guestbook.php	✓	200	5627
http://testphp.vulnweb.com	GET	/hpp/?pp=12	✓	200	599
http://testphp.vulnweb.com	GET	/hpp/params.php?aaaa%2...	✓	200	214
http://testphp.vulnweb.com	GET	/hpp/params.php?p=valid...	✓	200	221
http://testphp.vulnweb.com	GET	/listproducts.php?artist=1	✓	200	8211
http://testphp.vulnweb.com	GET	/listproducts.php?artist=2	✓	200	5410
http://testphp.vulnweb.com	GET	/listproducts.php?artist=3	✓	200	4916
http://testphp.vulnweb.com	GET	/listproducts.php?cat=1	✓	200	8097
http://testphp.vulnweb.com	GET	/listproducts.php?cat=2	✓	200	5528
http://testphp.vulnweb.com	GET	/listproducts.php?cat=3	✓	200	4916
http://testphp.vulnweb.com	GET	/listproducts.php?cat=4	✓	200	4916
http://testphp.vulnweb.com	GET	/product.php?pic=1	✓	200	6645
http://testphp.vulnweb.com	GET	/product.php?pic=2	✓	200	6585
http://testphp.vulnweb.com	GET	/product.php?pic=3	✓	200	6618
http://testphp.vulnweb.com	GET	/product.php?pic=4	✓	200	6670

However, if we want to check the pages or a specific directory, we can simply navigate the left side and select our desired option there.



The screenshot displays a web application interface on the left and a network request log on the right. The file list on the left includes various PHP files and folders, with 'guestbook.php' highlighted by a red arrow. The network panel on the right shows a POST request to '/guestbook.php' with a status of 200. The raw request data is visible, showing the URL and the request body: 'name=anonymous+user&text=604111&submit=add+message'.

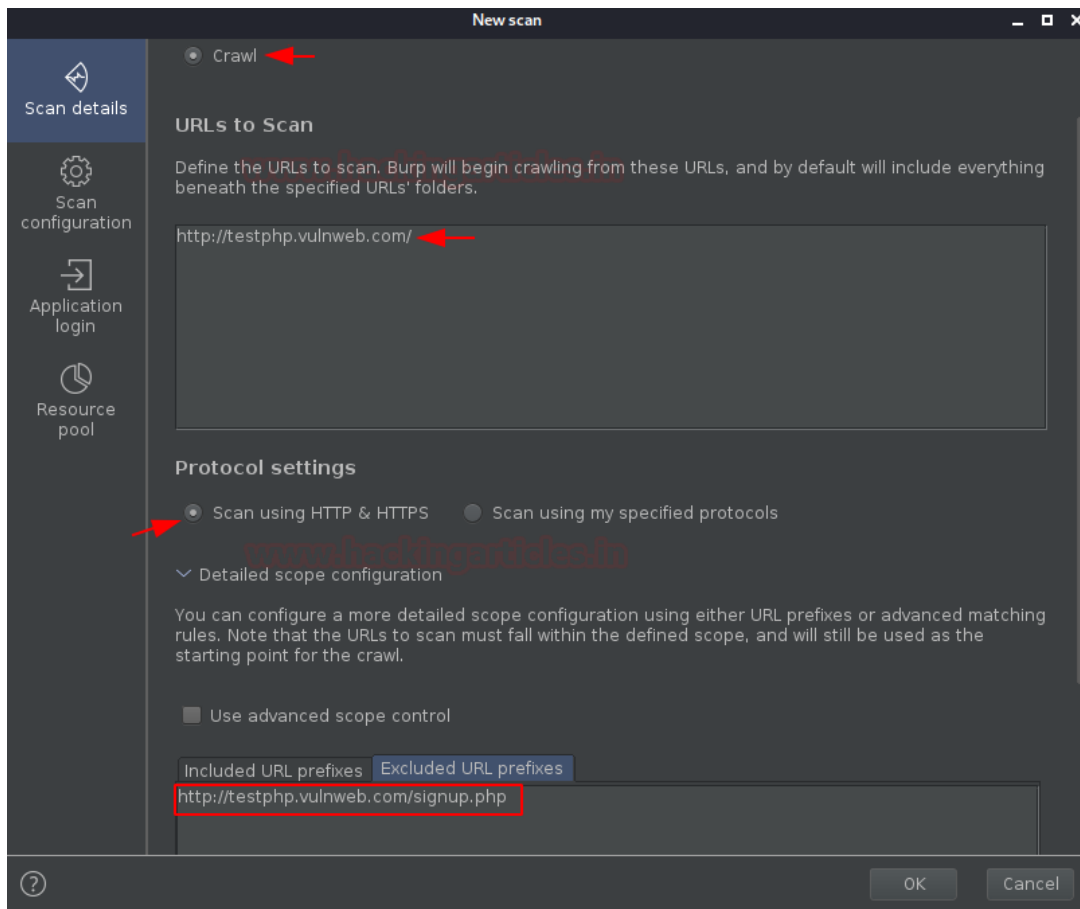
## Customizing the Crawler

What, if some specific web pages are **Out of Scope**? Or the website needs some **specific credentials** to surf the restricted web pages?

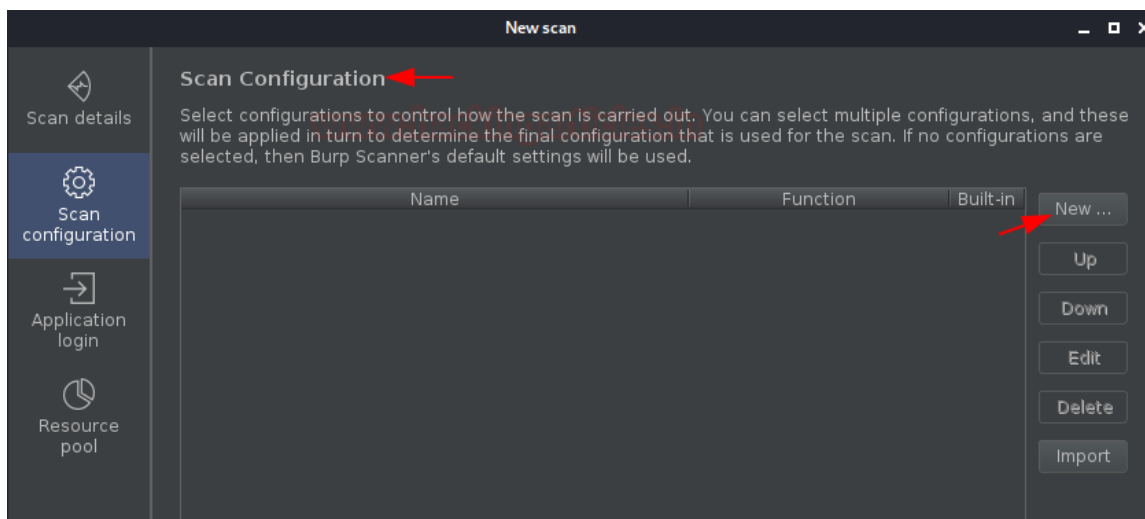
Therefore, in such cases, we need to configure our crawler, such that, it could work as we want it to. So, to do this, let's get back to the dashboard and select the **"New Scan"** option again. But for this time we won't hit **"OK"** after setting the URL.

## Configuring Out of Scope URL's

Below at the protocol setting, there is an option for the **Detailed Scope Configuration**, where we'll simply navigate to the **"Excluded URL prefixes"** and will enter the out of Scope URL i.e. <http://testphp.vulnweb.com/signup.php>

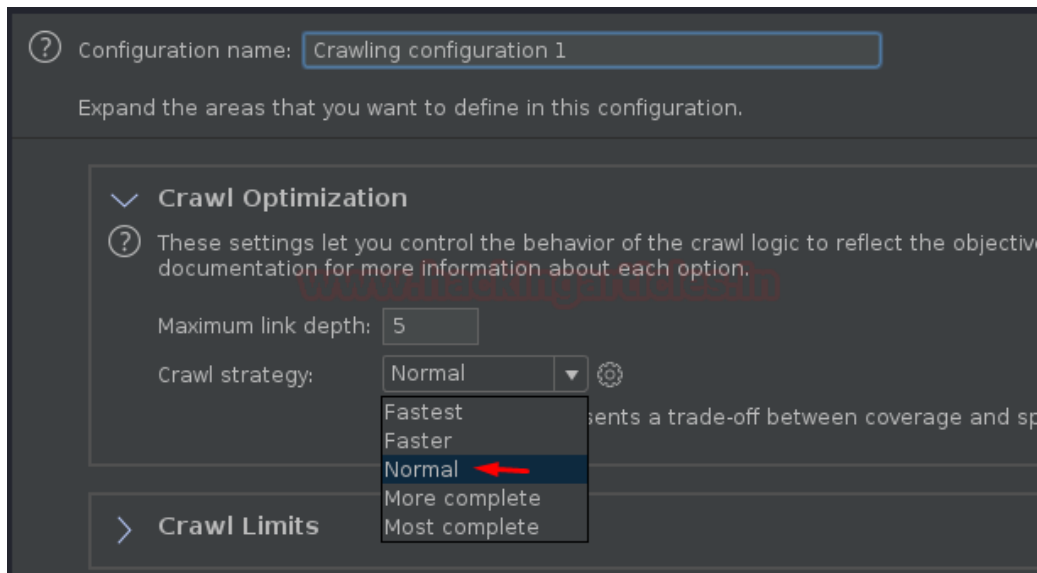


For further customization, we'll thus move to the **Scan Configuration** option. And there we'll hit the **"New"** button to set up a new crawler.

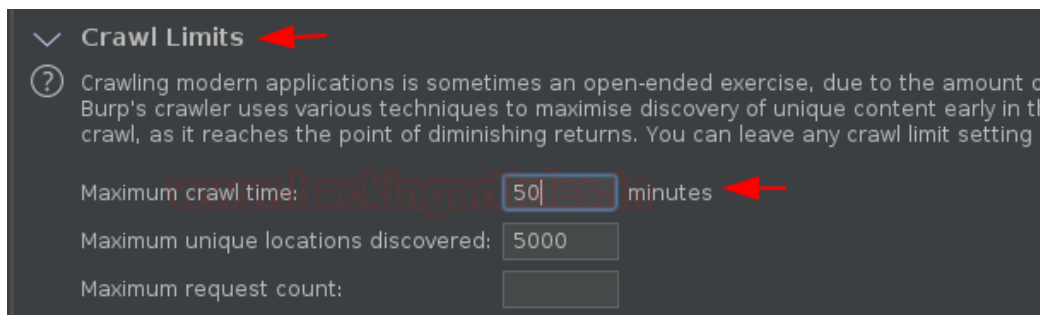


As soon as we do so, we'll thus get **another window open** with the **configuration options**. Let's keep the configuration name as the default, however, you can change it if you wish so.

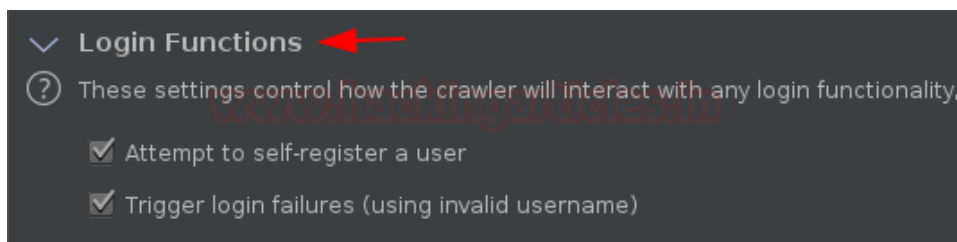
Further, the Crawl optimization option segregates within the **"Fastest to the Deepest"**, thereby we'll thus change it according to our requirement.



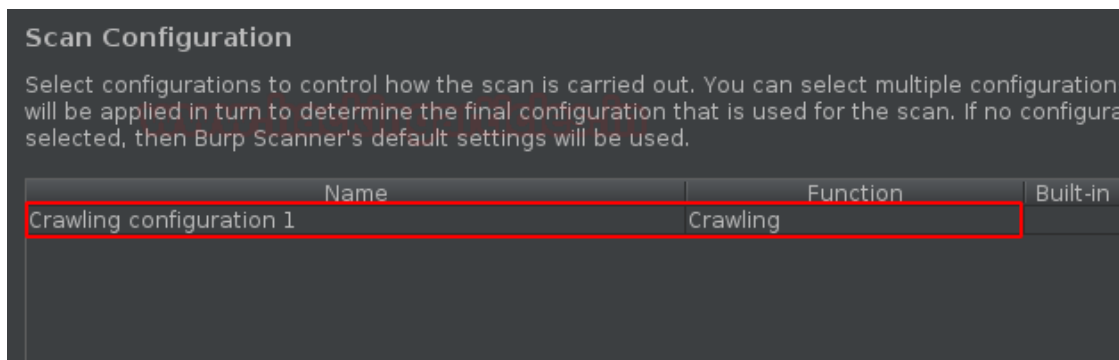
Crawl Limit is considered to be an important factor as it determines the time required and the depth to crawl an application. Thereby we'll set the **maximum crawl limit to 50 minutes** and the **Maximum unique locations discovered to 5000**.



Some applications carry **user registration or login portals**, thus checking both the options will thus guide the burp's crawler to **self-register** with some random values if encounters up with a signup portal and even use wrong credentials at the login portals to determine the website's behavior.

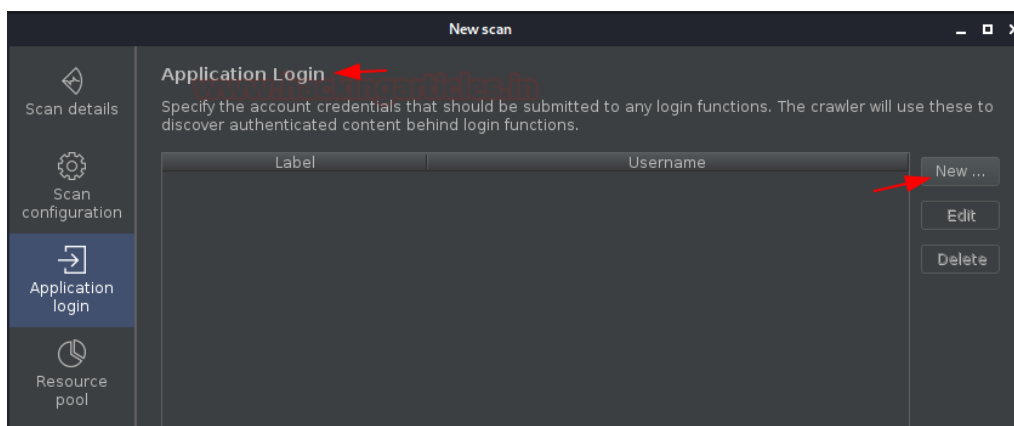


Now with all these configurations as soon as we hit the **"Save"** button we thus get our crawler listed at the **New scan dashboard**.

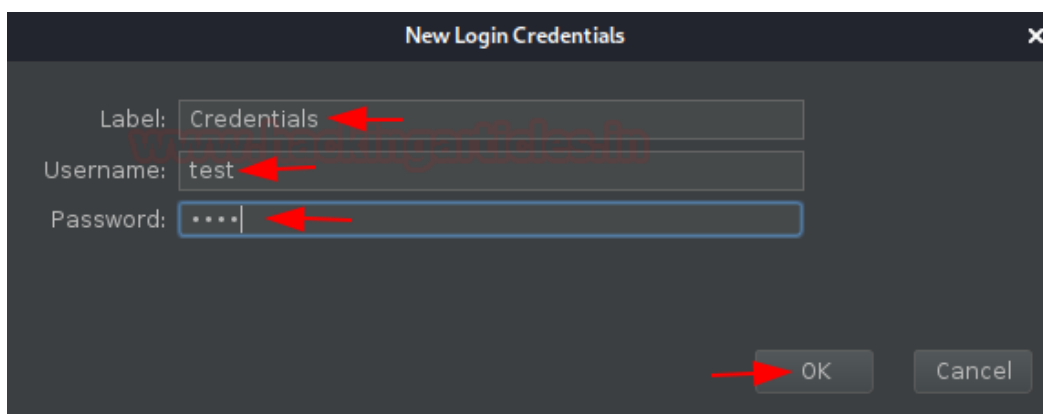


What, if the crawler encounters the restricted pages? Or an admin portal? Thereby, for such situations, let's feed up some default credentials so that the crawler can use them!!

Navigate to the “**Application login**” section and click on “New”.



Over in the pop-up box, enter the desired credentials & hit the “OK” button.



Along with all these things, we’re having one more option within the “**New Scan dashboard**”, i.e. “**Resource Pool**”.

A resource pool is a section defined for the **concurrent requests** or in simpler terms, we can say about how many requests the crawler will send to the application in one go, and what would be the time gap between the two requests.

Therefore, if you're testing a fragile application that could get down with an excessive number of requests, thus then you can configure it accordingly, but as we're testing the demo application thereby we'll set them to default.

**New scan**

**Resource Pool** ←

Specify the resource pool in which the scan will be run. Resource pools are used to manage the usage of system resources across multiple tasks.

☒ Use existing resource pool

Selected	Resource pool	Max concurrent requests	Delay between requests
<input checked="" type="radio"/>	Default resource pool	10	

☐ Create new resource pool

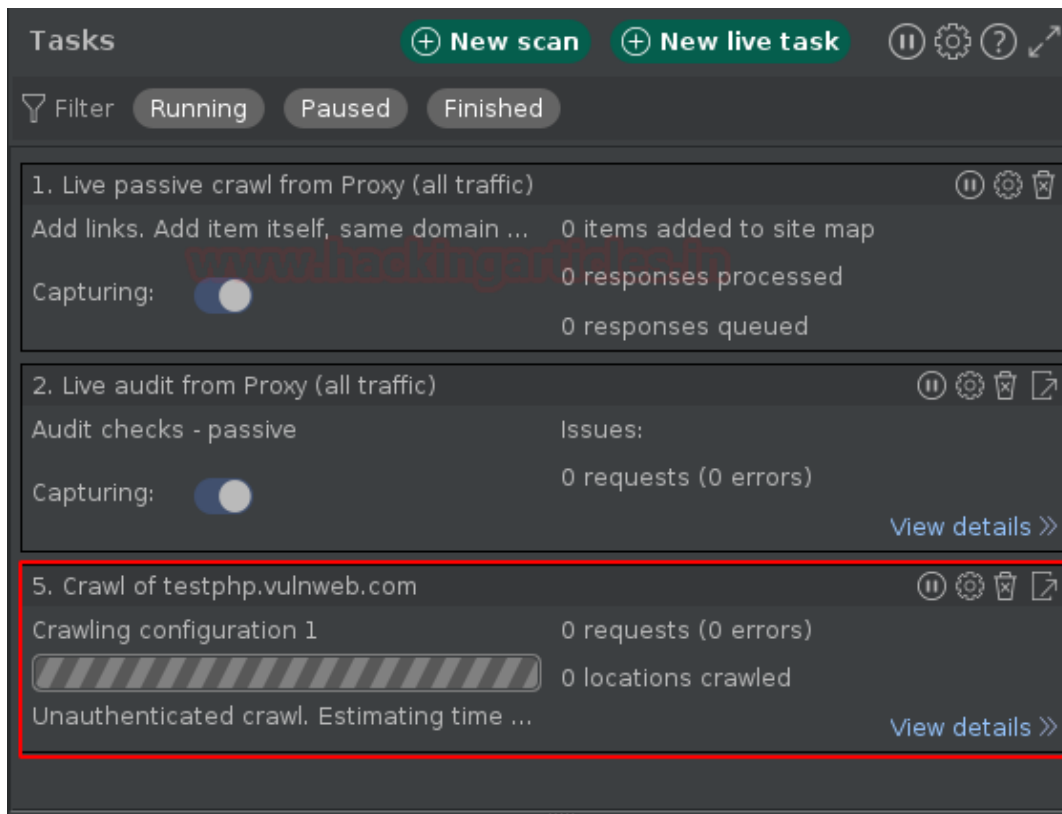
Name:

☐ Maximum concurrent requests:

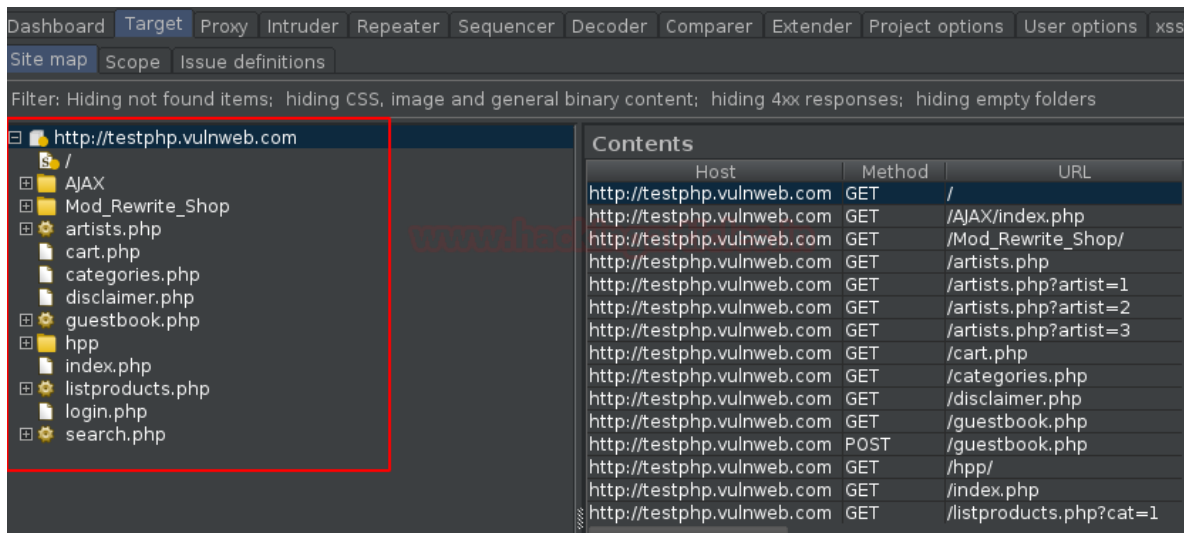
☐ Delay between requests:  milliseconds

☐ Add random variations

Now as we hit the **“OK”** button, our crawler will start which thus could be monitored at the dashboard.



Now, let's wait for it to get END!! As we navigate to the **Target tab** we'll thus get our output listed, and there we can notice that the **signup page is not mentioned**, which states that our configuration worked properly.



## Vulnerability Scanning Over Burpsuite

Rather than being an incepting tool, the burp suite acts as a vulnerability scanner too. Thereby, it scans the applications with a name as **"Audit"**. There are several vulnerability scanners over the web and the burp suite is one of them, as it is designed to be used by the security testers, and to fit in closely with

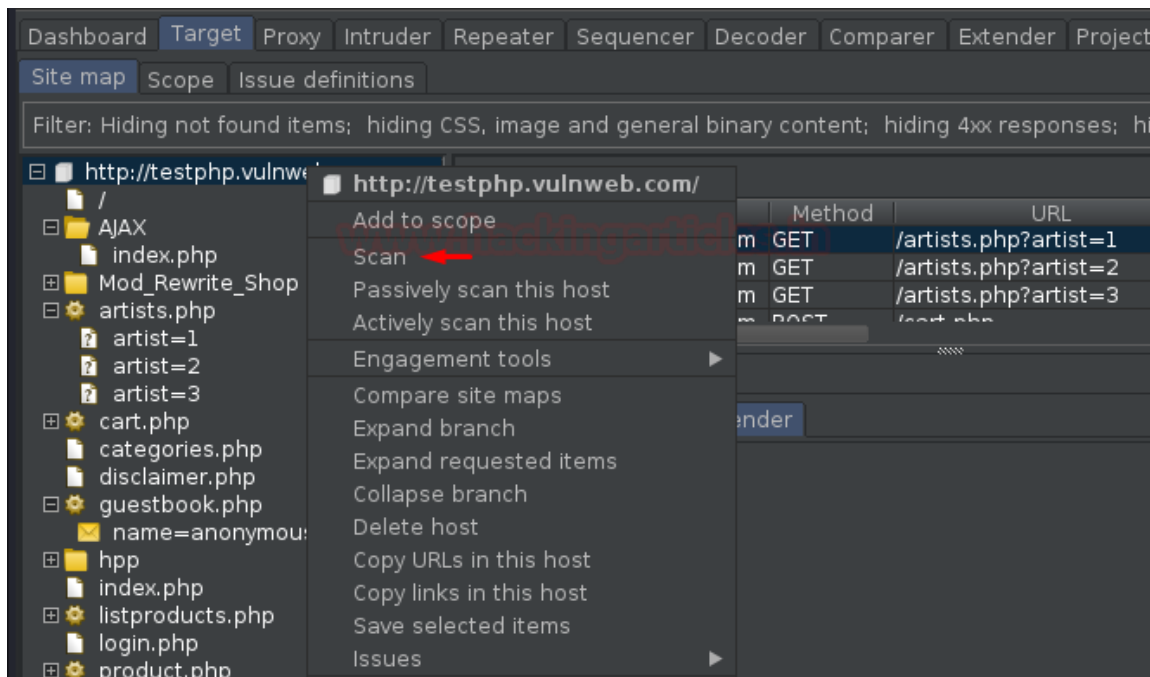
the existing techniques and methodologies for performing manual and semi-automated penetration tests of web applications.

So let's dig the "**testphp.vulnweb**" vulnerable application and check out what major vulnerabilities it carries within.

## Auditing with the default configuration

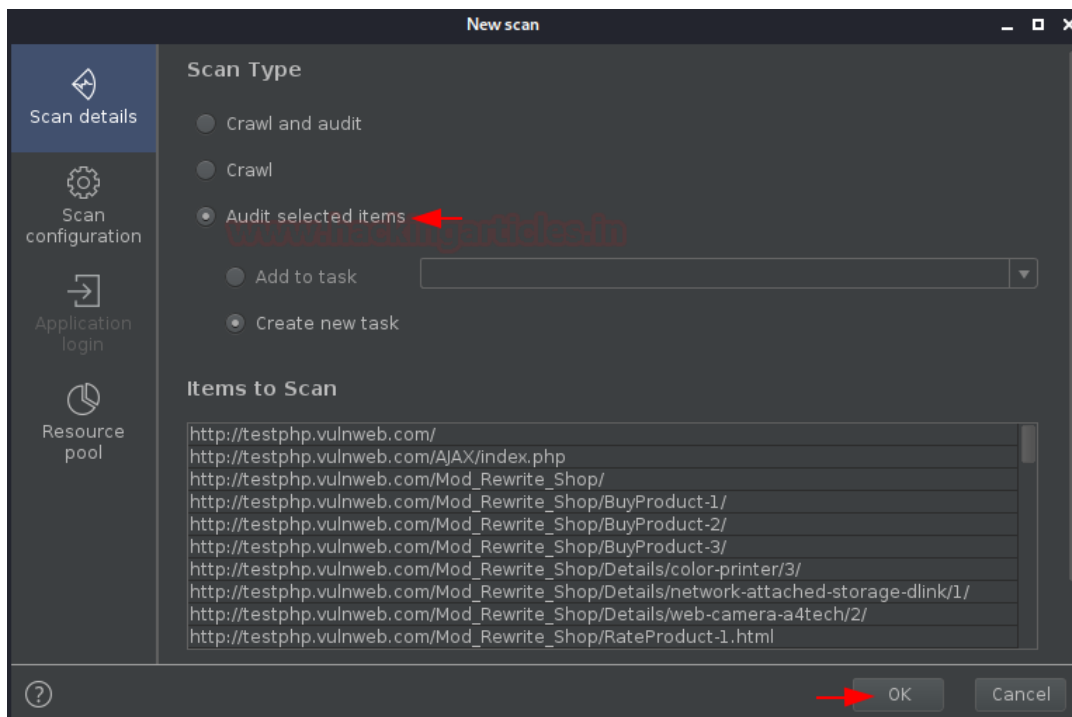
As we've already crawled the application thus it would be simpler to audit it, however, **to launch a scanner all we need is a URL**, whether we get it by intercepting the request, or through the target tab.

From the screenshot, you can perceive that we've sent the base URL by doing a right-click and opting the "**Scan**".



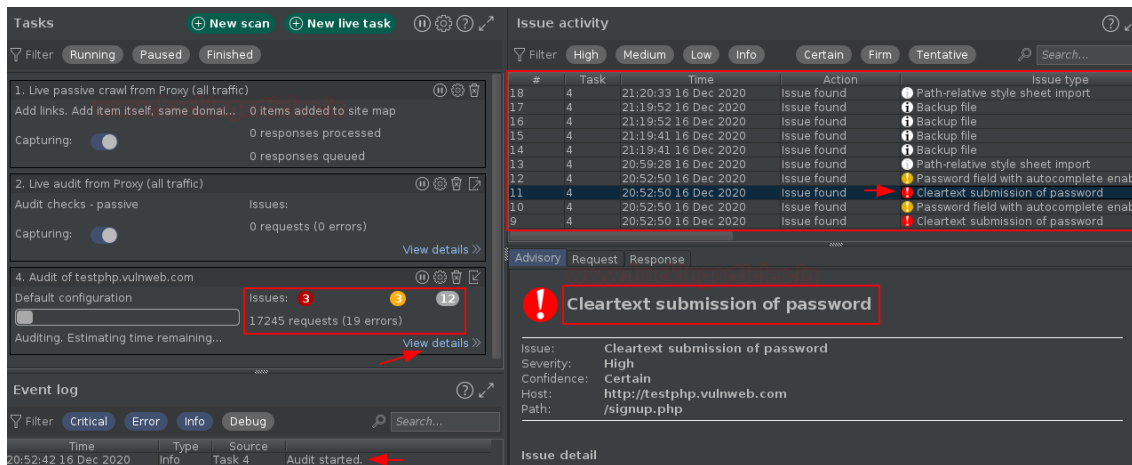
As soon as we do so, we'll thus be redirected back to the **New Scan's Dashboard**. But wait!! This time we're having one more option i.e. "**Audit Selected items**", as soon as we select it we'll thus get all the URLs within the **Item to Scan** box (This happens because we've opted for the base request).

As we're dealing with the default auditing, we'll thus simply hit the "**OK**" button there.



And now I guess you know where we need to go. Yes!! The **Dashboard** tab.

This time not only the **Tasks** section and the **Event log** is changed but we can see the variations in the **Issue activity** and the **advisory** sections too.

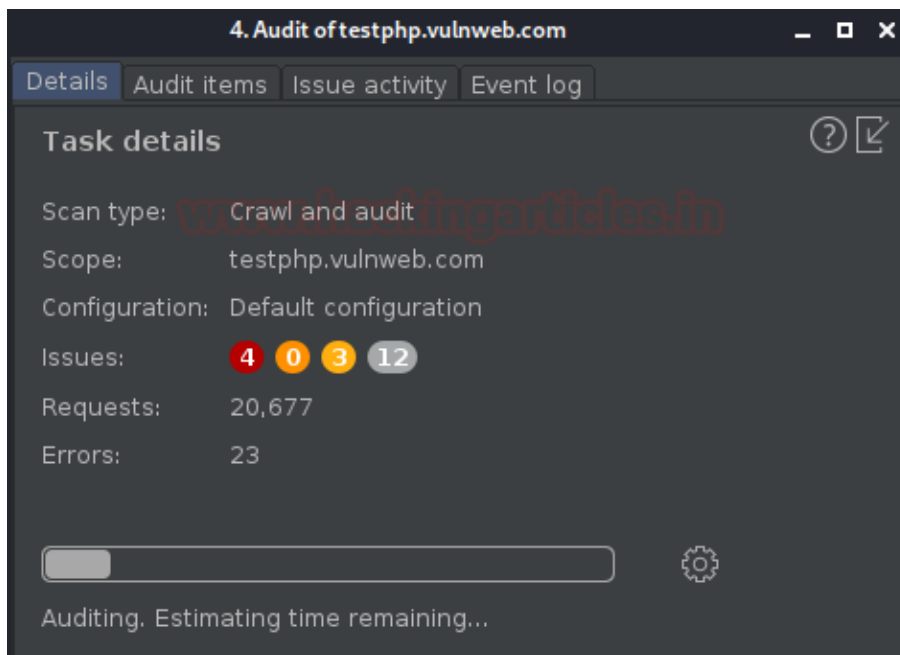


From the above image, we can see that within a few minutes our scanner has sent about 17000 requests to the web application and even dumped several vulnerabilities according to their severity level.

What if we want to see the detailed version??

To do so, simply click on the **View Details** section placed at the bottom of the defined task, and will thus get redirected to a new window will all the refined details within it.





Cool!! Let's check the Audited Items.

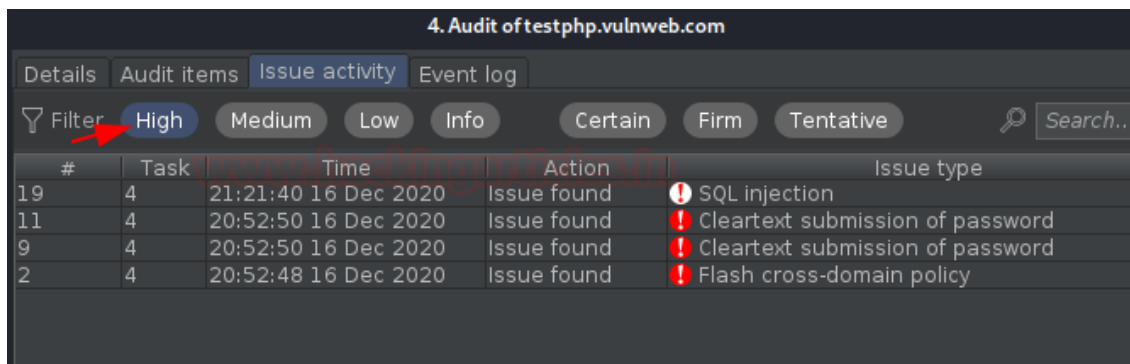
And as we hit the Audit Items tab, we'll thus get landed up to the detailed version of the audited sections, where we'll get the statutes, Active & Passive phases, Requests per URLs, and many more.

Details	Audit items	Issue activity	Event log					
URL	Status	Passive ph...	Active phases	JavaScript ph...	Issues	Requests	Errors	Insertion point
/	Scanning	1 2	1 2 3 4 5	1 2 3	1	560		3
/AJAX/index.php	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		1254	2	6
/Mod_Rewrite_Shop/	Scanning	1 2	1 2 3 4 5	1 2 3	1	656	1	4
/Mod_Rewrite_Shop/BuyProduct-1/	Errors: unknown h...	1 2	1 2 3 4 5	1 2 3		773	2	5
/Mod_Rewrite_Shop/BuyProduct-2/	Scanning	1 2	1 2 3 4 5	1 2 3		26		5
/Mod_Rewrite_Shop/BuyProduct-3/	Scanning	1 2	1 2 3 4 5	1 2 3		26		5
/Mod_Rewrite_Shop/Details/color-printer/3/	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		1281	4	7
/Mod_Rewrite_Shop/Details/network-attached-sto...	Scanning	1 2	1 2 3 4 5	1 2 3		1230	1	7
/Mod_Rewrite_Shop/Details/web-camera-a4tech/2/	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		1332	1	7
/Mod_Rewrite_Shop/RateProduct-1.html	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	2	1043	3	5
/Mod_Rewrite_Shop/RateProduct-2.html	Errors: request tim...	1 2	1 2 3 4 5	1 2 3		769		5
/Mod_Rewrite_Shop/RateProduct-3.html	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	2	1066	1	5
/artists.php	Errors: unknown h...	1 2	1 2 3 4 5	1 2 3		206	1	5
/artists.php	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	1	584	2	6
/artists.php	Scanning	1 2	1 2 3 4 5	1 2 3	1	504		
/artists.php	Errors: request tim...	1 2	1 2 3 4 5	1 2 3	1	590	1	6
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3	1	477		5
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		424		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		362		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		236		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3		171		8
/cart.php	Scanning	1 2	1 2 3 4 5	1 2 3				

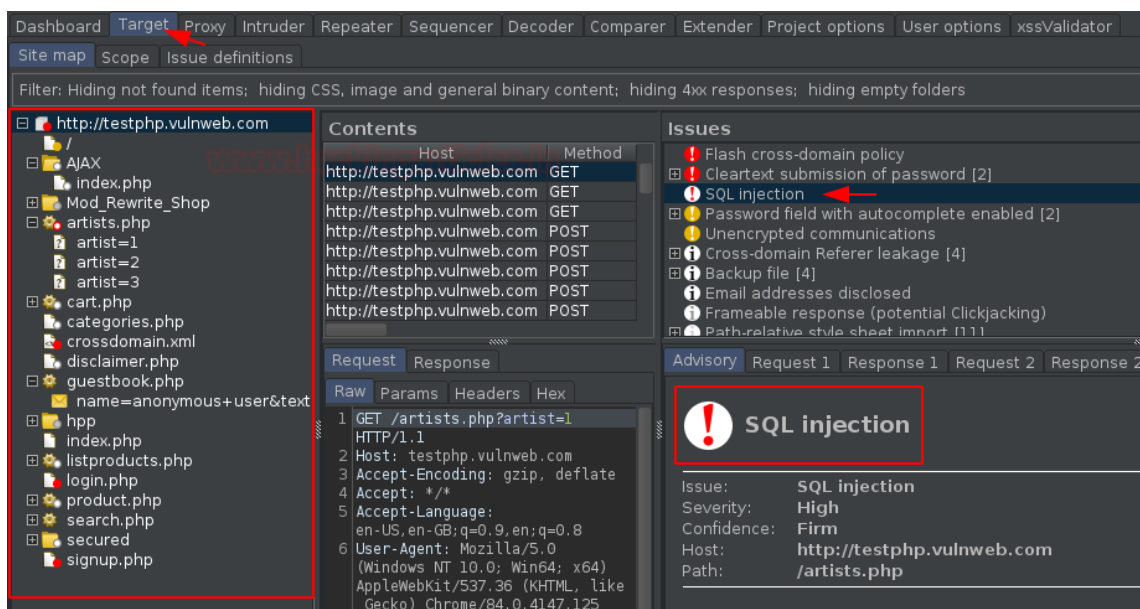
Further, we can even check the in-detailed Issues that have been found in the web application.

Details		Audit items		Issue activity		Event log		
Filter		High	Medium	Low	Info	Certain	Firm	Tentative
#	Task	Time	Action	Issue type		Host	Path	
21	4	21:26:08 16 Dec 2020	Issue found	Path-relative style sheet import		http://testphp.vulnweb.com	/artists.php	
20	4	21:24:24 16 Dec 2020	Issue found	Path-relative style sheet import		http://testphp.vulnweb.com	/cart.php	
19	4	21:21:40 16 Dec 2020	Issue found	SQL injection		http://testphp.vulnweb.com	/artists.php	
18	4	21:20:33 16 Dec 2020	Issue found	Path-relative style sheet import		http://testphp.vulnweb.com	/AJAX/index.php	
17	4	21:19:52 16 Dec 2020	Issue found	Backup file		http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-3.htm	
16	4	21:19:52 16 Dec 2020	Issue found	Backup file		http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-3.htm	
15	4	21:19:41 16 Dec 2020	Issue found	Backup file		http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-1.htm	
14	4	21:19:41 16 Dec 2020	Issue found	Backup file		http://testphp.vulnweb.com	/Mod_Rewrite_Shop/RateProduct-1.htm	
13	4	20:59:28 16 Dec 2020	Issue found	Path-relative style sheet import		http://testphp.vulnweb.com	/	
12	4	20:52:50 16 Dec 2020	Issue found	Password field with autocomplete enabled		http://testphp.vulnweb.com	/signup.php	
11	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password		http://testphp.vulnweb.com	/signup.php	
10	4	20:52:50 16 Dec 2020	Issue found	Password field with autocomplete enabled		http://testphp.vulnweb.com	/login.php	
9	4	20:52:50 16 Dec 2020	Issue found	Cleartext submission of password		http://testphp.vulnweb.com	/login.php	
8	4	20:52:50 16 Dec 2020	Issue found	Cross-domain Referer leakage		http://testphp.vulnweb.com	/http/	
7	4	20:52:50 16 Dec 2020	Issue found	Cross-domain Referer leakage		http://testphp.vulnweb.com	/	
6	4	20:52:50 16 Dec 2020	Issue found	Cross-domain Referer leakage		http://testphp.vulnweb.com	/	

Although we can even filter them according to their defined severity levels.



Not only these things, over in the **target tab**, something is waiting for us i.e. the Issues and the Advisory are also mentioned there, but if we look at the **defined tree** at the left panel we can see some colorful dots majorly **red and grey** indicating that these URL's are having **high and informative existing vulnerabilities** respectively.



However, from the below image, with the **Advisory option of SQL Injection**, there is a specific panel for **Request & Response**, let's check them and determine how the scanner confirms that there is an SQL Injection existing.

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

## SQL injection

Compare responses

Issue: SQL injection  
Severity: High  
Confidence: Firm  
Host: http://testphp.vulnweb.com  
Path: /artists.php

### Issue detail

The **artist** parameter appears to be vulnerable to SQL injection attacks. The payloads **71972544** or **4095=04095** and **90514291** or **9068=9069** were each submitted in the artist parameter. These two requests resulted in different responses, indicating that the input is being incorporated into a SQL query in an unsafe way.

Note that automated difference-based tests for SQL injection flaws can often be unreliable and are prone to false positive results. You should manually review the reported requests and responses to confirm whether a vulnerability is actually present.

Additionally, the payload **(select\*from(select(sleep(20)))a)** was submitted in the artist parameter. The application took **20156** milliseconds to respond to the request, compared with **0** milliseconds for the original request, indicating that the injected SQL command caused a time delay.

The database appears to be MySQL.

### Issue background

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe

As we navigate to the 3<sup>rd</sup> Request, we got an SQL time-based query injected in the “**artist=**” field.

And as we **shared this request with the browser**, we got a **delay of about 20 seconds**, which confirms that the vulnerabilities dumped with the scanner are triggerable.

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

Raw Params Headers Hex

```

1 GET /artists.php?artist=(select*from(select(sleep(20)))a) HTTP/1.1
2 Host: testphp.vulnweb.com
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/84.0.4147.125 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9 Referer: http://testphp.vulnweb.com/artists.php
10
11

```

You might be wondering like okay I got the vulnerability, but I’m not aware of it – what more could I get with or how could I chain it to make a crucial hit.

Therefore, to solve this issue, we got an Issue definition section, where we can simply go through with the defined or captured vulnerability.

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options xssValidator

Site map Scope **Issue definitions**

### Issue Definitions

This listing contains the definitions of all issues that can be detected by Burp Scanner.

Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
ASP.NET tracing enabled	High	0x00100280
File path traversal	High	0x00100300
XML external entity injection	High	0x00100400
LDAP injection	High	0x00100500
XPath injection	High	0x00100600
XML injection	Medium	0x00100700
ASP.NET debugging enabled	Medium	0x00100800
HTTP PUT method is enabled	High	0x00100900
Out-of-band resource load (HTTP)	High	0x00100a00
File path manipulation	High	0x00100b00
PHP code injection	High	0x00100c00
Server-side JavaScript code injection	High	0x00100d00
Perl code injection	High	0x00100e00
Ruby code injection	High	0x00100f00
Python code injection	High	0x00100f10
Expression Language injection	High	0x00100f20
Unidentified code injection	High	0x00101000
Server-side template injection	High	0x00101080
SSI injection	High	0x00101100
Cross-site scripting (stored)	High	0x00200100
HTTP request smuggling	High	0x00200140
Web cache poisoning	High	0x00200180
HTTP response header injection	High	0x00200200
Cross-site scripting (reflected)	High	0x00200300
Client-side template injection	High	0x00200308
Cross-site scripting (DOM-based)	High	0x00200310

#### OS command injection

##### Description

Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.

OS command injection vulnerabilities are usually very serious and may lead to compromise of the server hosting the application, or of the application's own data and functionality. It may also be possible to use the server as a platform for attacks against other systems. The exact potential for exploitation depends upon the security context in which the command is executed, and the privileges that this context has regarding sensitive resources on the server.

##### Remediation

If possible, applications should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks, which cannot be manipulated to perform additional commands than the one intended.

If it is considered unavoidable to incorporate user-supplied data into operating system commands, the following two layers of defense should be

## Defining Audit Configurations

Similar to the Crawling option, we can simply configure this Audit too, by getting back to the “New Scan” dashboard with a right-click on the defined URL & hitting Scan.

**New scan**

**Scan details**

**Scan configuration**

**Application login**

**Resource pool**

### Scan Type

☐ Crawl and audit

☐ Crawl

☒ Audit selected items

☐ Add to task

☒ Create new task

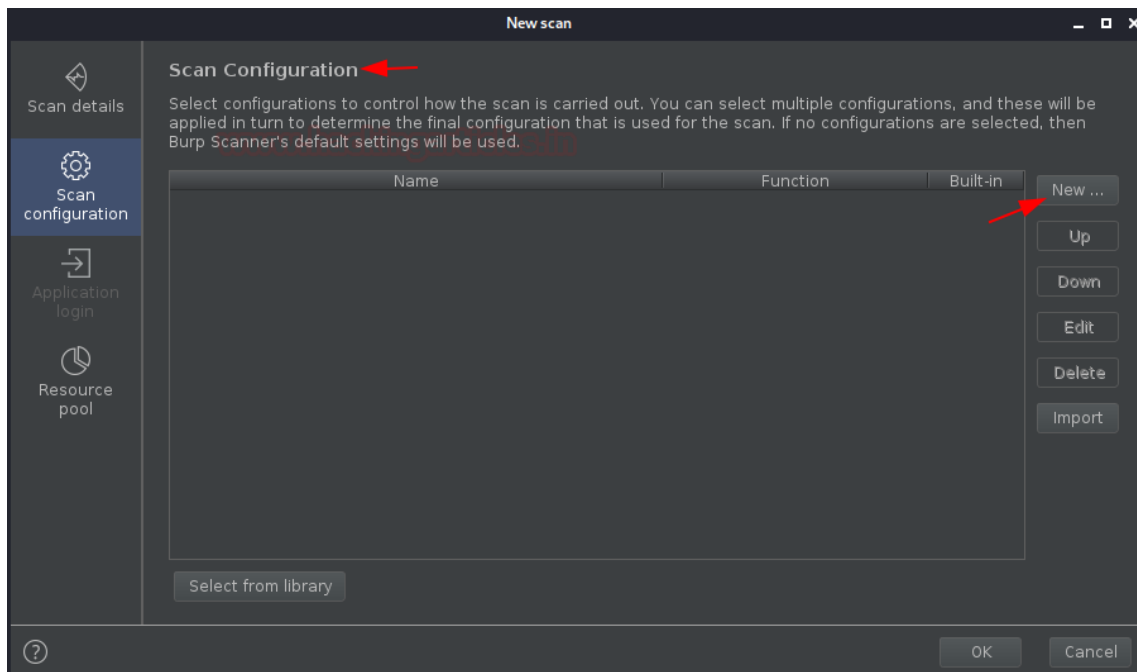
### Items to Scan

http://testphp.vulnweb.com/  
 http://testphp.vulnweb.com/AJAX/index.php  
 http://testphp.vulnweb.com/Mod\_Rewrite\_Shop/  
 http://testphp.vulnweb.com/artists.php  
 http://testphp.vulnweb.com/artists.php  
 http://testphp.vulnweb.com/artists.php  
 http://testphp.vulnweb.com/artists.php  
 http://testphp.vulnweb.com/cart.php  
 http://testphp.vulnweb.com/categories.php  
 http://testphp.vulnweb.com/disclaimer.php

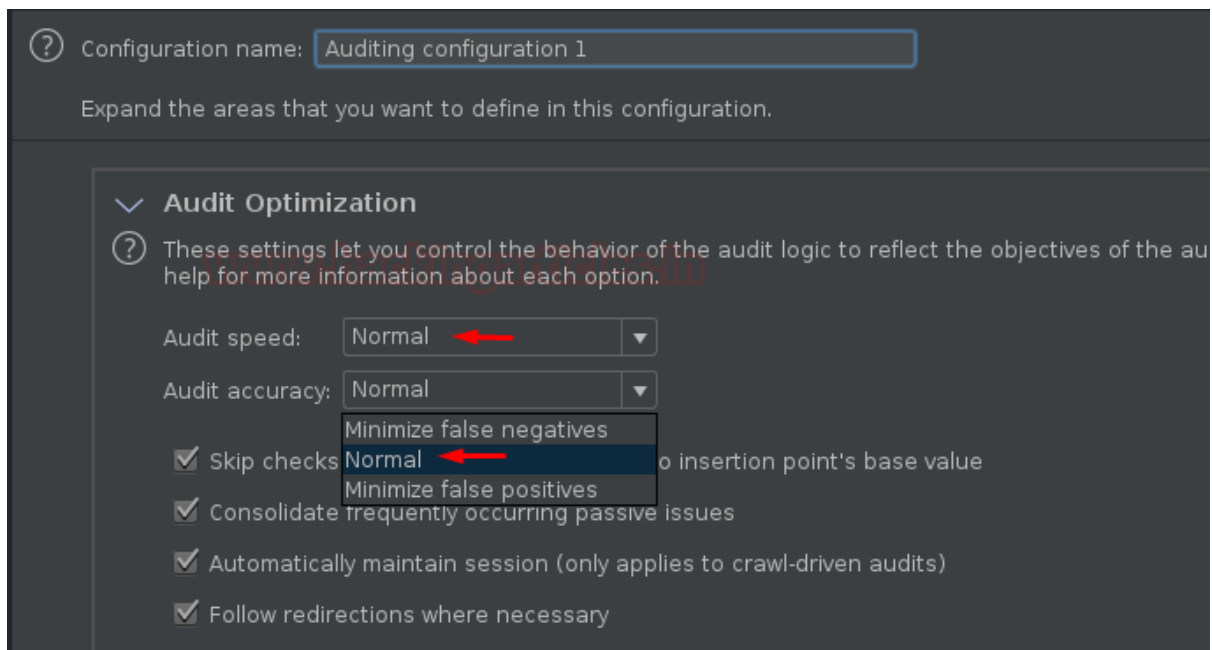
OK Cancel

Here, in the above image, if we scroll down, we'll thus get the same option to set the **out of Scope** URL as was in the Crawl section.

Now, moving further with the scan configurations, hit the “New” button as we did earlier.



Setting the configuration name to default and manipulating the audit accuracy to normal, you can define it according to your need.



Now comes to the most important section to **define the Issues reported** by selecting the “**Scan Type**”. Here to complete the scan faster, I’m simply taking the **Light active scan** option, but you can opt for any of the following –

- **Passive** – These issues are detected simply by inspecting the application’s behavior of requests and responses.
- **Light active** – Here this detects issues by making a small number of benign additional requests.


- **Medium active** – These are issues that can be detected by making requests that the application might reasonably view as malicious.
- **Intrusive active** – These issues are detected by making requests that carry a higher risk of damaging the application or its data. For example, SQL injection.
- **JavaScript analysis** – These are issues that can be detected by analyzing the JavaScript that the application executes on the client side.

Issues Reported

? These settings control which issues Burp will check for. You can select issues by scan type or individually. If you see the detection methods that are used for some types of issues.

☒ Select by scan type:

☐ Passive

☒ Light active 

☐ Medium active

☐ Intrusive active

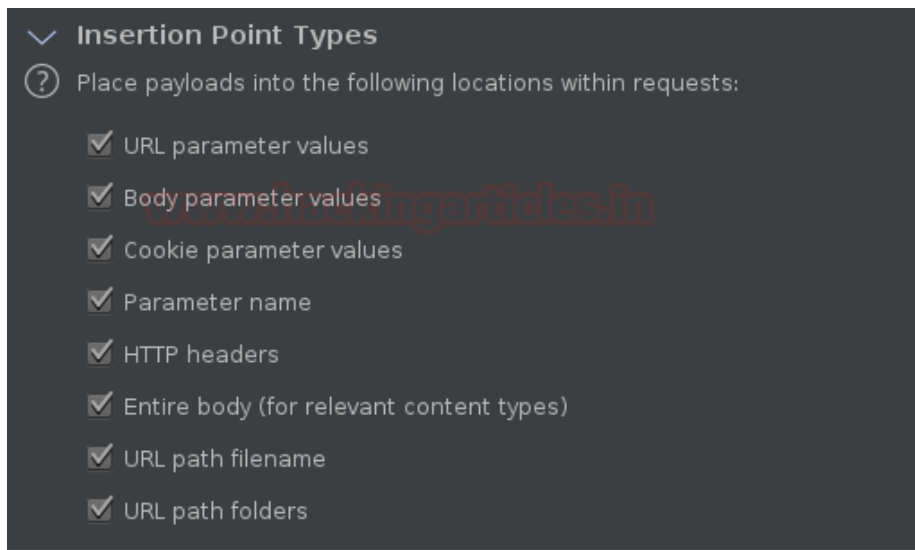
☐ JavaScript analysis

☐ Select individual issues:

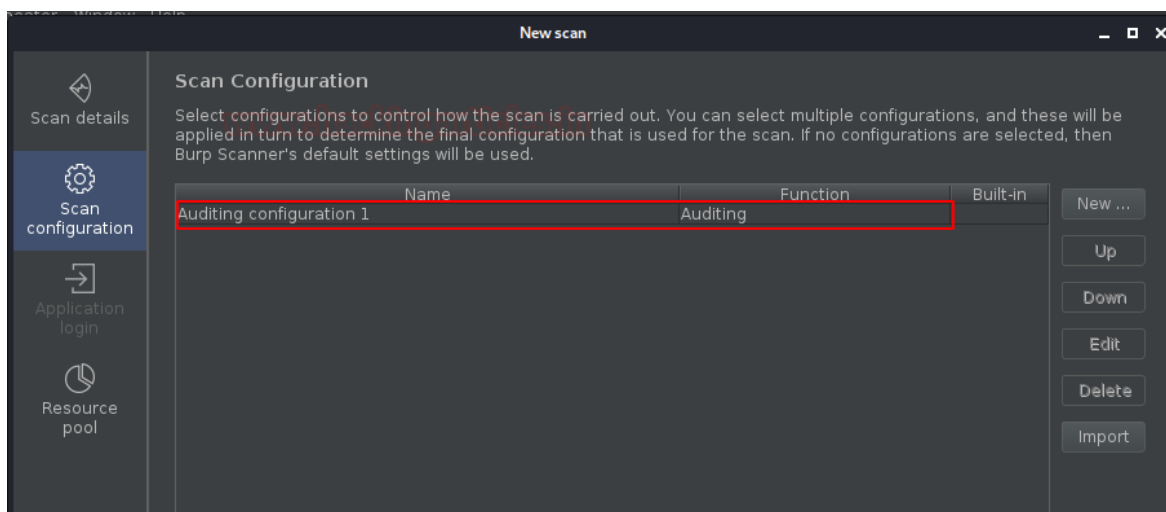
Filter **Passive** **Light** **Medium** **Intrusive** **JavaScript**

Enabled	Name	Passive	Light	Medium	Intrusive	JavaScript	Typical severity
<input checked="" type="checkbox"/>	OS command injection			•			High
<input checked="" type="checkbox"/>	SQL injection				•		High
<input checked="" type="checkbox"/>	SQL injection (second order)				•		High
<input checked="" type="checkbox"/>	ASP.NET tracing enabled			•			High
<input checked="" type="checkbox"/>	File path traversal				•		High
<input checked="" type="checkbox"/>	XML external entity injection			•			High
<input checked="" type="checkbox"/>	LDAP injection				•		High
<input checked="" type="checkbox"/>	XPath injection				•		High
<input checked="" type="checkbox"/>	XML injection				•		Medium
<input checked="" type="checkbox"/>	ASP.NET debugging enabled			•			Medium
<input checked="" type="checkbox"/>	HTTP PUT method is enabled				•		High
<input checked="" type="checkbox"/>	Out-of-band resource load (HTTP)			•			High

You might be aware of the concept of insertion points, as they are the most important sections to the vulnerability to get hit. They are locations within the requests where the payloads are injected. However, the burp's scanner even audits the insertion points too, and thus could also be manipulated in this phase.



Now as we're done with the configuration and we hit the **"Save"** button, our customized audit is thus gets listed up in the **New Scan's dashboard**.



However, the option of Application login is disabled in this section as there is no specific need to log in to an application just for vulnerability testing.

Therefore, now we know what's next, i.e. **hitting the OK button** and **moving to the dashboard**. And as soon as we reach there, we'll get the result according to our configuration with about **2700 requests**.

But this time, the major issue is only **"1"**

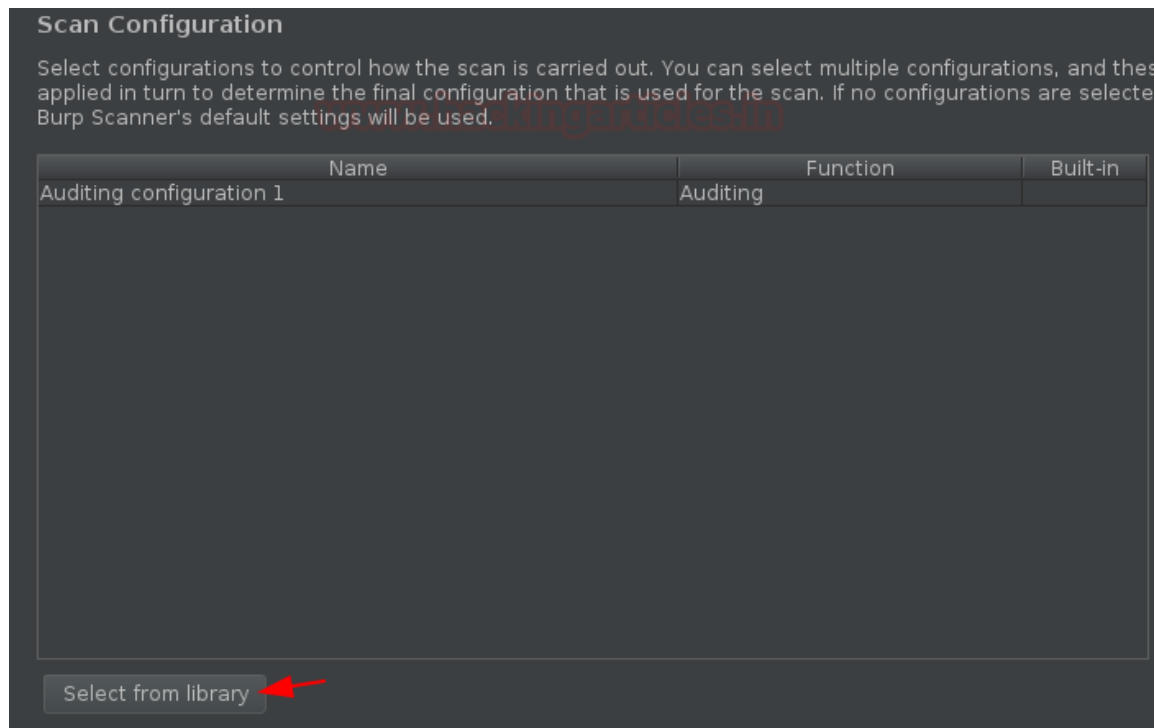
The screenshot shows the Burp Suite interface. On the left, the 'Tasks' panel lists three tasks: '1. Live passive crawl from Proxy (all traffic)', '2. Live audit from Proxy (all traffic)', and '4. Audit of testphp.vulnweb.com'. Task 4 is selected and shows 'Auditing configuration 1' with 'Issues: 1' and '2687 requests (0 errors)'. The 'Audit finished.' status is highlighted with a red box. On the right, the 'Issue activity' panel shows a list of issues found, including 'Path-relative style sheet import' and 'Input returned in response (reflected)'. The 'Issue type' column is highlighted with a red box.

Now, if we move back to the Target tab and select any request from the left panel and do a right-click over there, we'll get **2 options rather than "1"**, i.e. the **last customization we configure** will thus get into this field and if we share any request within it, it will start auditing accordingly.

The screenshot shows the Burp Suite interface with a right-click context menu open over a request in the 'Contents' panel. The menu options include 'Add to scope', 'Scan', 'Passively scan this branch', 'Actively scan this branch', 'Engagement tools', 'Compare site maps', and 'Expand branch'. The 'Scan' option is highlighted with a red box, and its sub-menu is visible, showing 'Open scan launcher' and 'Add to task: 4. Auditing configuration 1'. The 'Add to task: 4. Auditing configuration 1' option is also highlighted with a red box.

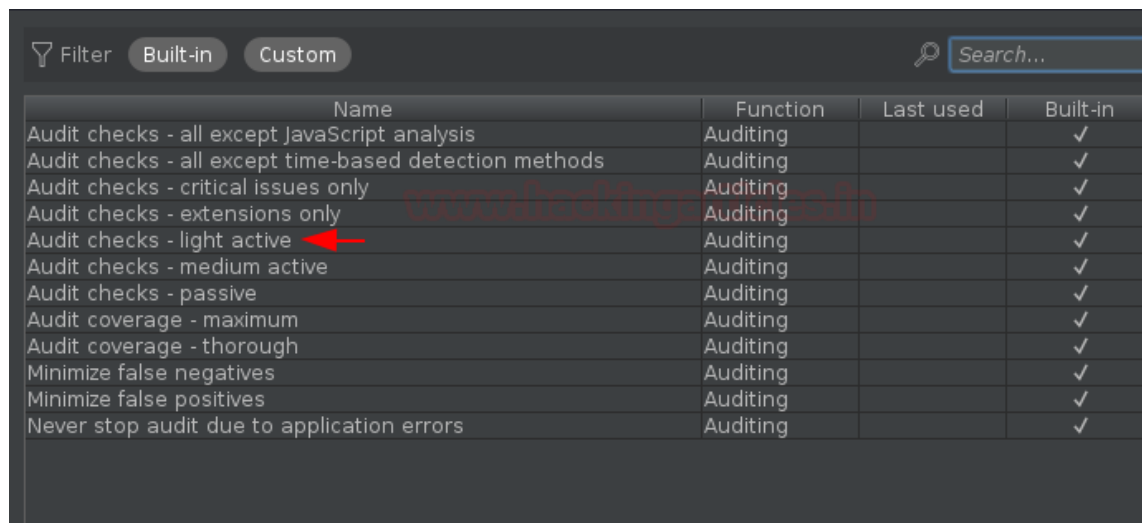
Thereby, we'll opt for the Open scan launcher again to check the other features too. As we head back, we're welcomed with our previous customized audit, but at the bottom, there is a **"Select from library"** option, click there and check what it offers.



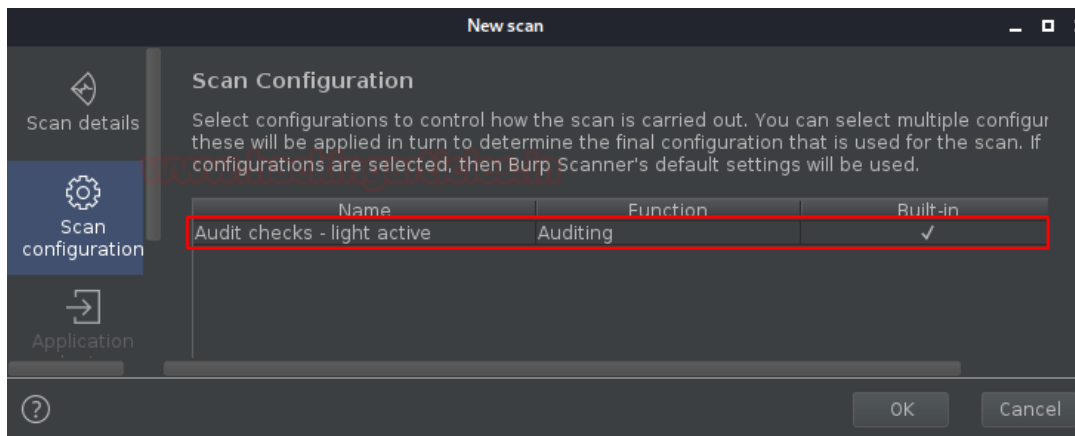


So, wasn't it a bit confusing to configure the audit by manipulating every option it has??

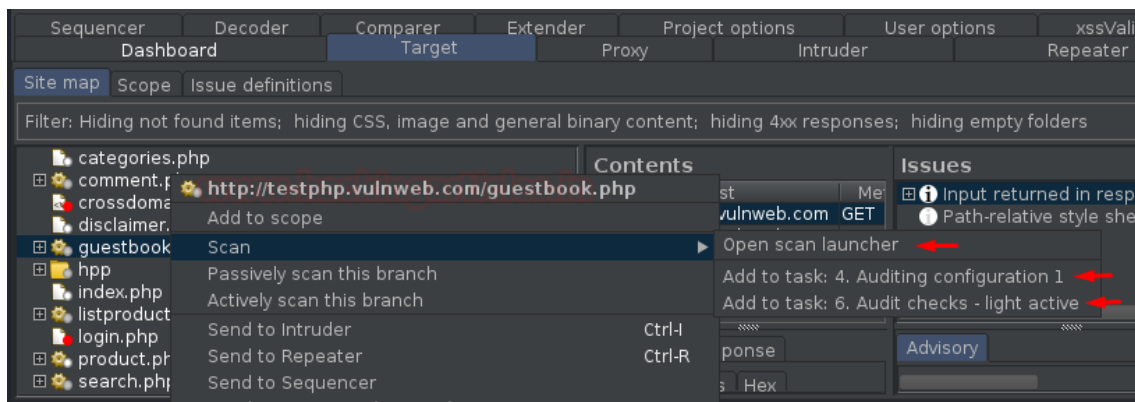
Thereby, to get rid of this, the burp suite offers one more great feature to opt for a **built-in Audit check**, where we simply need to select any and continue.



And as we select one, we'll thus get our option listed back into the New Scan dashboard.



Hit **“OK”** and check the result in the dashboard!! Further, now if we navigate to the **Target** tab and do a right-click on any request we’ll thus **get 3 options rather than 2.**



## Crawling & Scanning with an Advanced Scenario

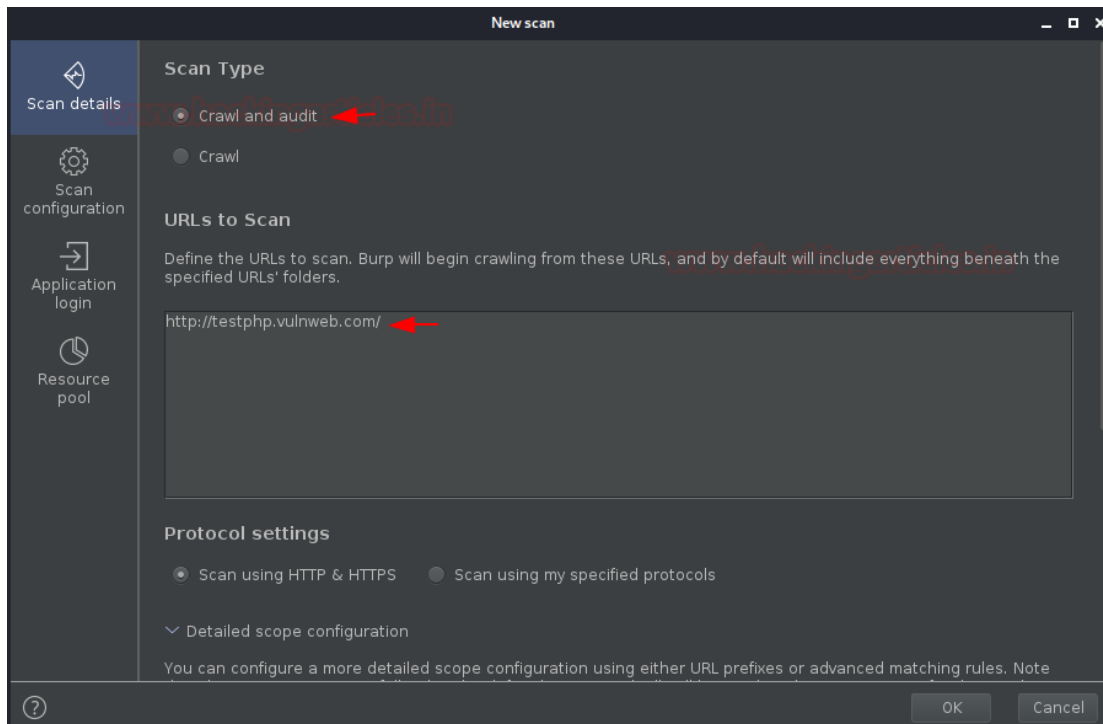
Up till now, we’ve used the scanner and the crawler individually, but what if we want to do both things together. Thereby to solve this problem too, the burp suite creators give us an End-to-End scan opportunity, where our burp suite will –

1. First Crawl the application and discover the contents and the functionalities within it.
2. Further, it will start auditing it for the vulnerabilities.

Thereby, to do all this, all it needs is a **“URL”**.

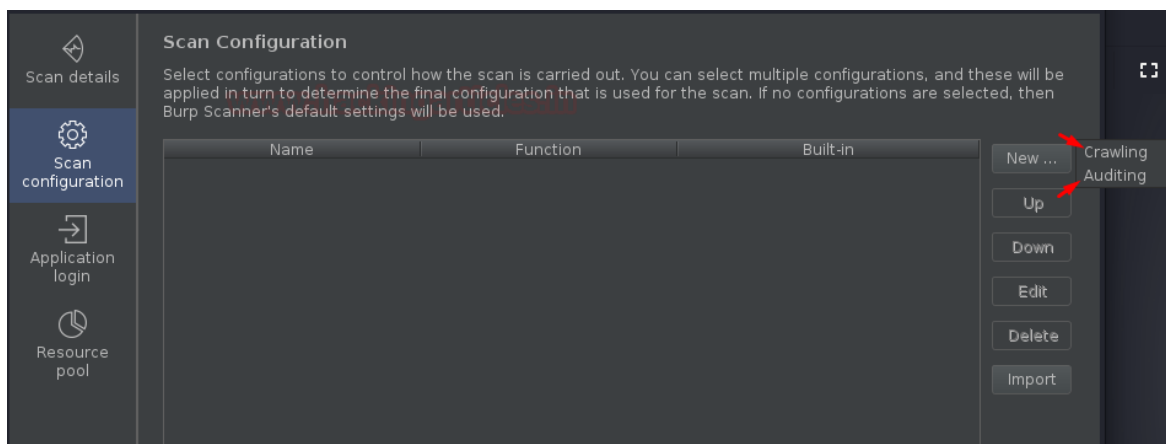
**Let’s check how we can do it.**

Back on the dashboard, select **“New Scan”**, and now this time opt **“Crawl & Audit”**, further mention the URL within it.



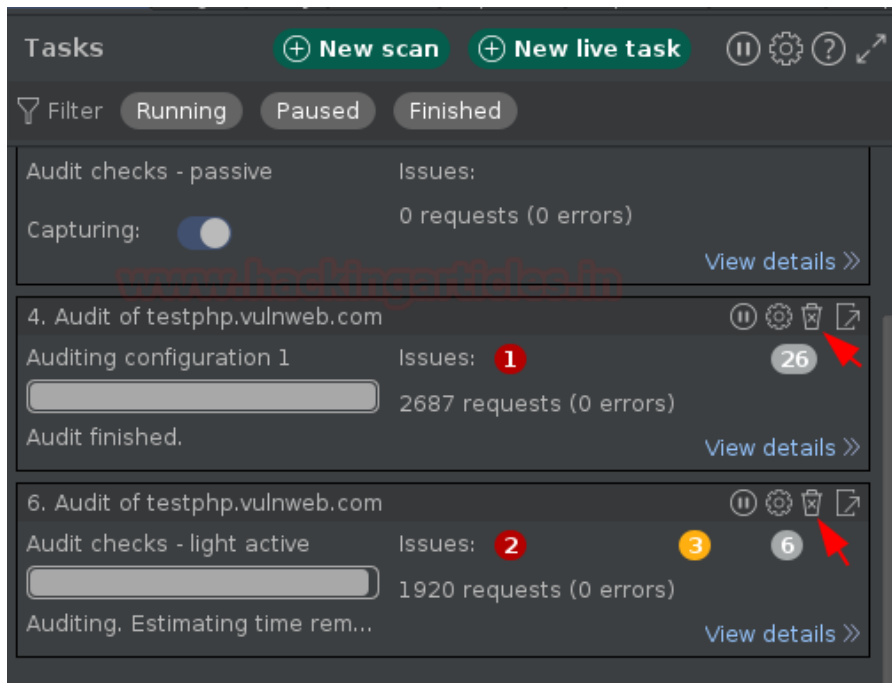
Great!! Now let's check the **Scan Configuration options**, as we move there and when we click on the **"New"** button, rather than redirecting us to the customization menu it asks us about where to go, for crawl optimization or audit configuration.

However, all the internal options are the same.

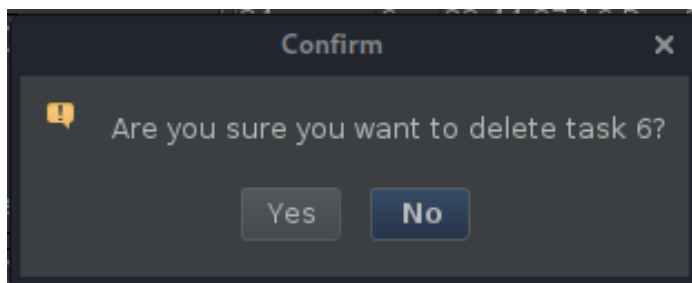


## Deleting the Defined Tasks

Rather not only knowing how to start or configure the things up, but we should also be aware of how to end them all. Thereby let's click on the Dustbin icon defined up as a Task option, to delete our completed or incompleted tasks.



And as we do so, we got the confirmation pop-up as



# JOIN OUR TRAINING PROGRAMS

