



Burpsuite

Web application Pentesting

Original Author(s): *Raj & Megha*



Table of Contents

Abstract.....	3
Introduction.....	4
Burpsuite for Pentester: Authorize	4
Common vulnerabilities detected by Authorize	4
Understanding the working of Authorize	5
Installation and Setup	6
Navigating and Configuration Options	8
Practical Demonstration of Authorize in Action	13
Burpsuite for Pentester: Logger++	21
Setting Up & Navigating	21
Navigating	22
Query-Based Filter	24
Magical Filter	26
Filter Library	32
Regex-Based Filter	36
Export Data Feature	39
Conclusion	43
References	43



Abstract

In order to protect online assets, web application security testing is an essential element of safeguarding them. Burp Suite has been a leader in this area for many years and it's still being used by safety professionals as well as Ethical hackers.

The primary role of Burp Suite is to assist security professionals, such as penetration testers and ethical hackers, in identifying and addressing vulnerabilities in web applications, which will be the subject of this report.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.



Introduction

In the first section of this report, we'll explore a helpful extension called "Authorize," which makes the process of testing authentication and authorization in web application security easier.

Moving on to the second part, we'll take a look at another useful extension known as "Burp Logger++," which assists many web experts in identifying issues on websites.

Burpsuite for Pentester: Authorize

Authorize = Authenticate + Authorize

Authorization includes any method by which a system grants or revokes permission to access specific data or actions. Meanwhile, Authentication is a process by which an individual or system authenticates themselves as being who they claim to be.

- Common vulnerabilities detected by Authorize
- Understanding the Functionality
- Installation and Setup
- Navigation and Configuration options
- Practical Demonstration of Authorize in Action

Common vulnerabilities detected by Authorize

It is primarily focused on identifying authorization-related vulnerabilities. It can help to identify some of the main types of vulnerabilities, such as:

- **Inadequate Role-Based Access Control (RBAC):** It can uncover issues where user roles or permissions are not properly enforced, allowing users to access functionality or data they shouldn't have access to.
- **Broken Access Controls:** It can identify instances where access controls are not correctly implemented, leading to unauthorized access to resources or actions.
- **Insecure Direct Object References (IDOR):** It can find situations where attackers can manipulate input to access other users' data or perform actions, they shouldn't be able to.
- **Forced Browsing:** It can help identify cases where an attacker can navigate directly to restricted areas of the application by manipulating URLs.



- **Insufficient Authorization:** It may detect situations where user roles or permissions are not properly enforced, allowing unauthorized actions to be performed.
- **Horizontal and Vertical Privilege Escalation:** It can find vulnerabilities that enable attackers to escalate their privileges within the application, either by impersonating other users or gaining additional permissions.
- **Business Logic Flaws:** Authorize may discover business logic vulnerabilities, where application workflows can be manipulated in unintended ways, potentially leading to unauthorized actions or data exposure.

Remember that the effectiveness of Authorize depends on how well it is configured and your tests are carried out.

Understanding the working of Authorize

Let's understand how Authorize works. Suppose, for instance, a web application implements user-based roles and supports cookie-based authentication.

Normal User: has access to general functionality but is not allowed to access admin functions and database (read-only access).

Admin User: has access to all functionality (read/write access).

Capture the normal user cookies and add them to Authorize. Re-log in with the Admin user access all the admin functionality and update some data to the database.

What will Authorize be doing now? Authorize is capturing all requests and changing the administrator cookie with your normal user's cookies when you are browsing an application, then sending them to server. See the server response, if the server behaves in the same way as legitimate Admin (like 200 OK in response) and no errors have been detected. The request was highlighted as a Red Bypass! Another request shows as a Green Enforced!.

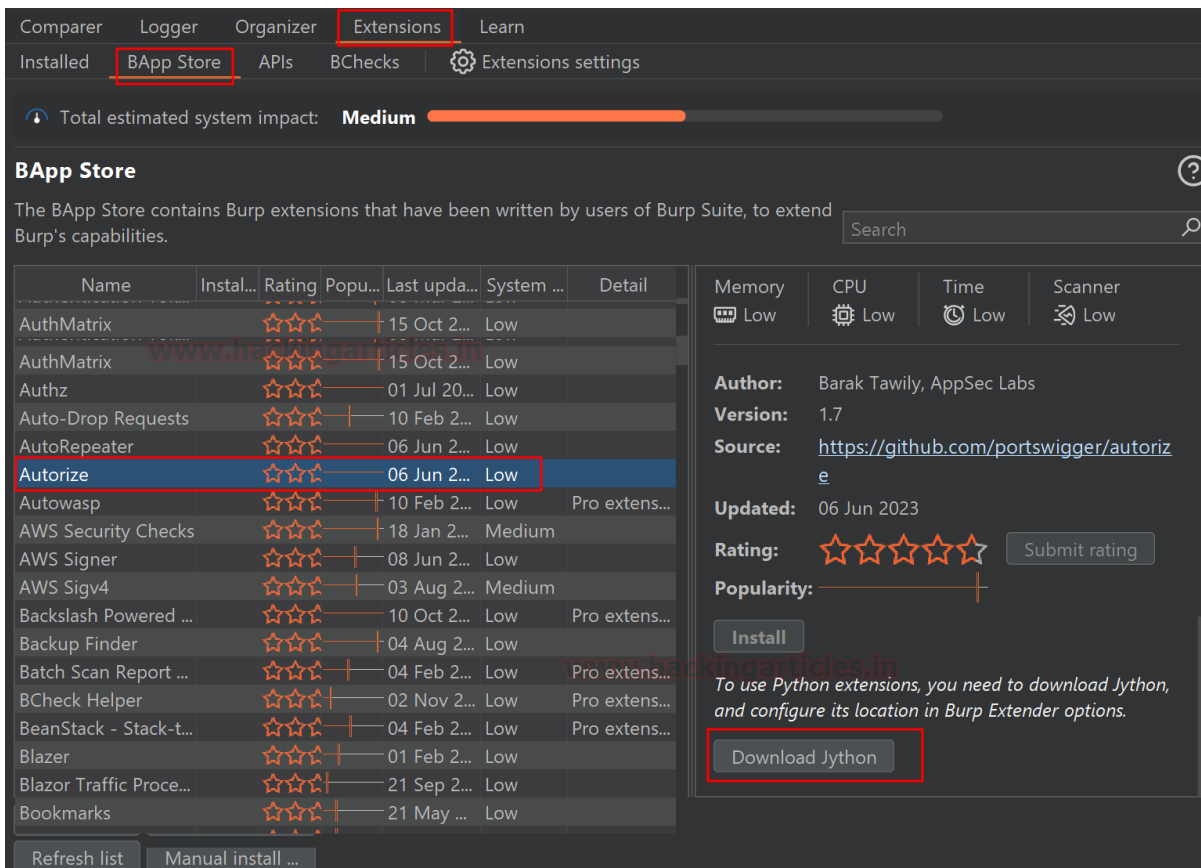
For every request sent to the server from a client, it will perform an automated test. With a large application, with over 30+ dynamic webpages, it's going to ease our work. There are a lot of URLs you need to test manually, so Authorize will do it for you.

Similarly, Authorize also detects an API endpoint problem in the same way. The authentication method must be checked for the API. Let's say an API uses a JWT token, you can control that by modifying its authorization header and identifying the authentication bypass issues with the APIs.

Installation and Setup

From the Bapp Store, you can download and install the extension. Select Bapp Store in Extensions. You can search for 'Authorize', or you can just look down. Click on it, scroll down to the right side.

The extension is built in Python, you will see that 'Jython' needs to be installed first.



BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Search

Name	Instal...	Rating	Popu...	Last upda...	System ...	Detail
AuthMatrix		☆☆☆		15 Oct 2...	Low	
AuthMatrix		☆☆☆		15 Oct 2...	Low	
Authz		☆☆☆		01 Jul 20...	Low	
Auto-Drop Requests		☆☆☆		10 Feb 2...	Low	
AutoRepeater		☆☆☆		06 Jun 2...	Low	
Authorize		☆☆☆		06 Jun 2...	Low	
Autowasp		☆☆☆		10 Feb 2...	Low	Pro extens...
AWS Security Checks		☆☆☆		18 Jan 2...	Medium	
AWS Signer		☆☆☆		08 Jun 2...	Low	
AWS Sigv4		☆☆☆		03 Aug 2...	Medium	
Backslash Powered ...		☆☆☆		10 Oct 2...	Low	Pro extens...
Backup Finder		☆☆☆		04 Aug 2...	Low	
Batch Scan Report ...		☆☆☆		04 Feb 2...	Low	Pro extens...
BCheck Helper		☆☆☆		02 Nov 2...	Low	Pro extens...
BeanStack - Stack-t...		☆☆☆		04 Feb 2...	Low	Pro extens...
Blazer		☆☆☆		01 Feb 2...	Low	
Blazor Traffic Proce...		☆☆☆		21 Sep 2...	Low	
Bookmarks		☆☆☆		21 May ...	Low	

Memory Low CPU Low Time Low Scanner Low

Author: Barak Tawily, AppSec Labs
Version: 1.7
Source: <https://github.com/portswigger/authorize>
Updated: 06 Jun 2023
Rating: ☆☆☆☆☆ Submit rating
Popularity: ————

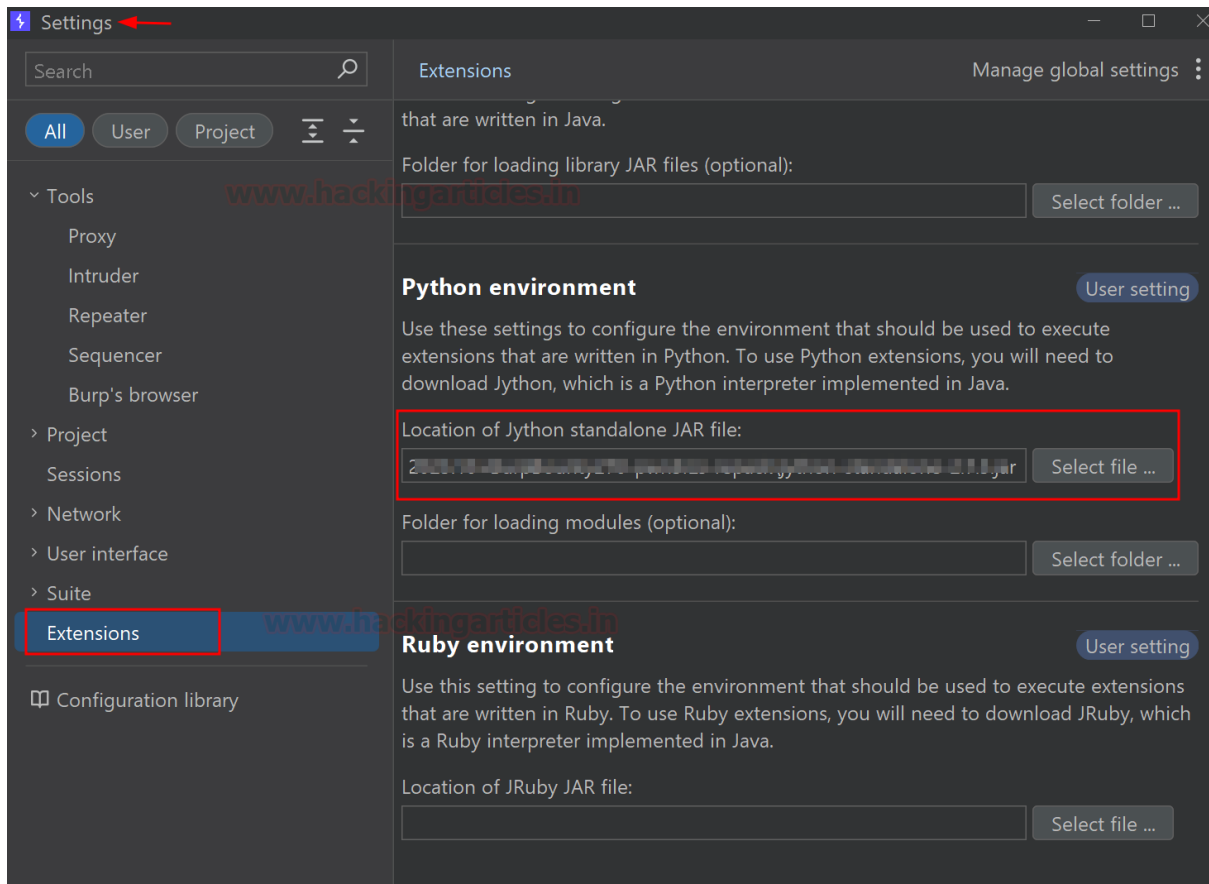
Install

To use Python extensions, you need to download Jython, and configure its location in Burp Extender options.

Download Jython

Browse the below link and download 'Jython Standalone'.

After downloading go to Setting > Extension > on the right side under Python Environment browser the Jython file. This environment has been successfully set up for Jython.



Restart the Burp program and follow this path to install Authorize on BApp Store. You'll notice that the install button is highlighted. You can click on it and install it.

Authorize

Authorize is an extension aimed at helping the penetration tester to detect authorization vulnerabilities. It is sufficient to give to the extension the cookies of a low privileged user and navigate the website with the extension and detects authorization vulnerabilities.

It is also possible to repeat every request without any cookies in order to detect authentication vulnerabilities.





The plugin works without any configuration, but is also highly customizable, allowing configuration of the extension. It is possible to save the state of the plugin and to export a report of the authorization tests in HTML format.

The reported enforcement statuses are the following:

1. Bypassed! - Red color
2. Enforced! - Green color
3. Is enforced??? (please configure enforcement detector) - Yellow color

Estimated system impact

Overall: **Low** 

Memory	CPU	Time	Scanner
 Low	 Low	 Low	 Low

Author: Barak Tawily, AppSec Labs

Version: 1.7

Source: <https://github.com/portswigger/authorize>

Updated: 06 Jun 2023

Rating: 

Popularity: 

Install

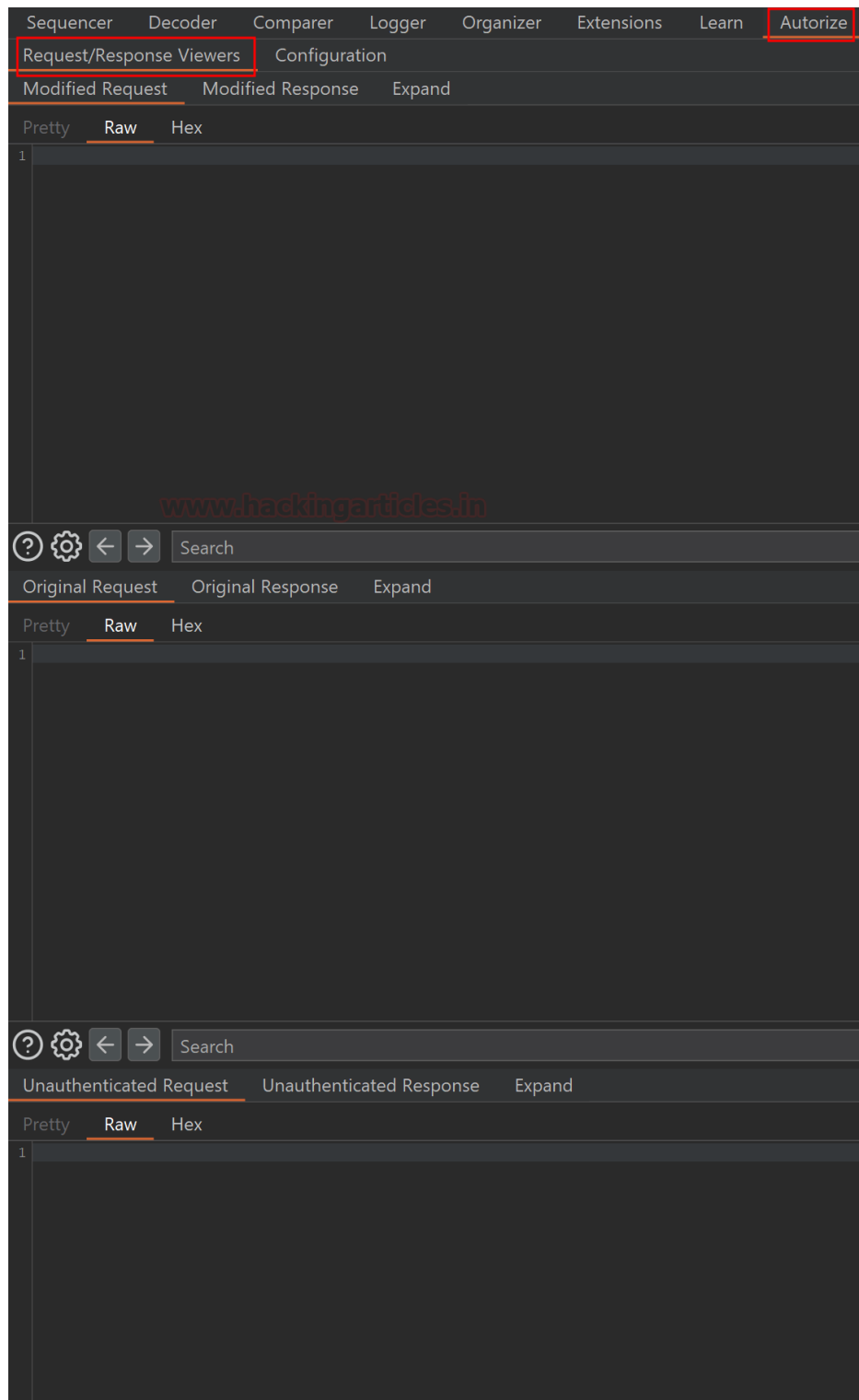
The Authorize tab will appear in the bar after successful installation.

Navigating and Configuration Options

There are two tabs under the Authorize section, the first one is Request/Response Viewers tab and the other one Configuration tab.

Request/Response Viewers: The Request/Response tab will display complete information about the particular request you capture within Authorize and choose. The manipulated

request will be displayed under the Modified Request section, the Original Request tab will display the original/unmodified request, and the Unauthenticated request will display the unauth request.

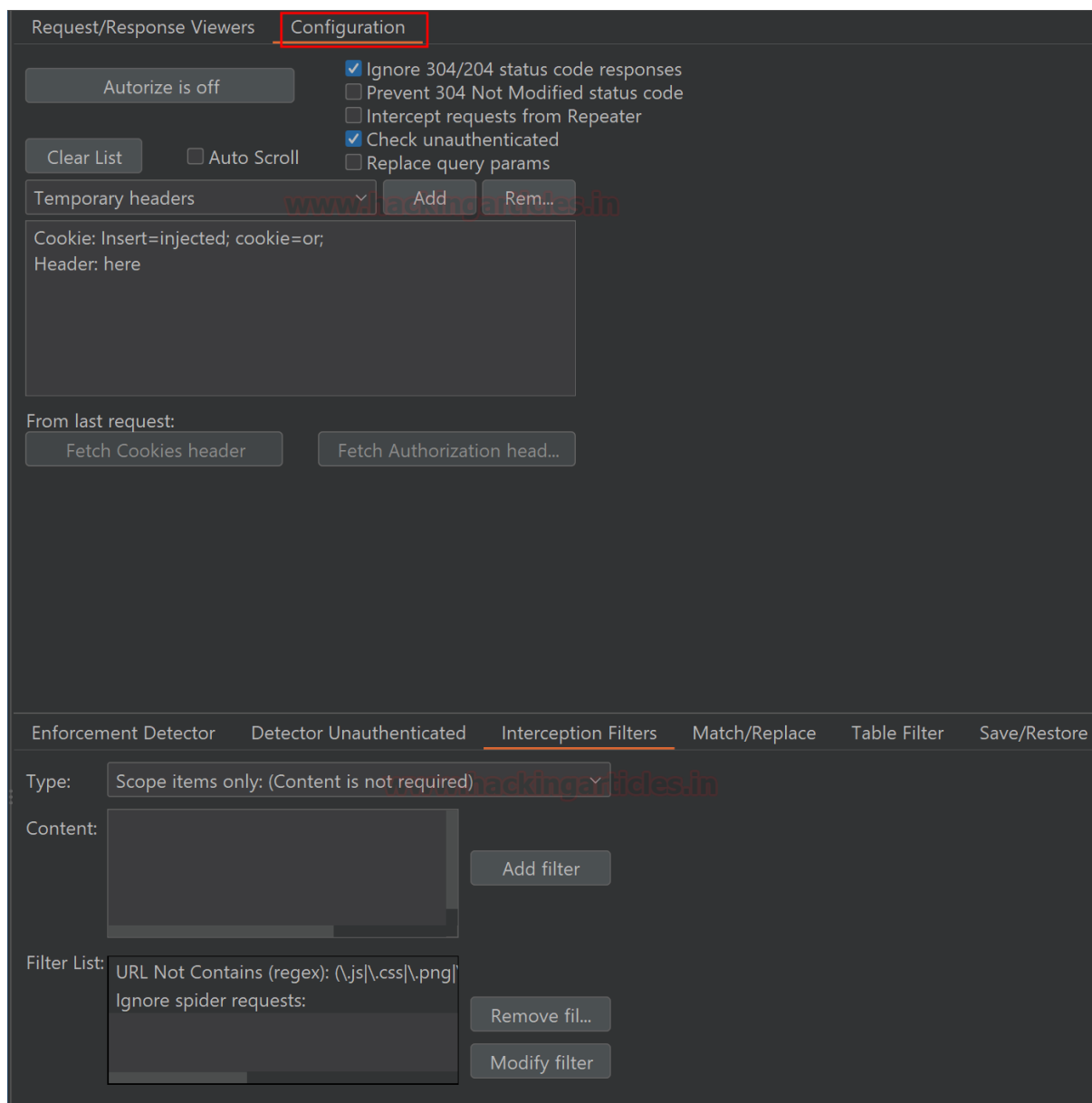


Configuration: Under the configuration tab you will see Authorize is off by default, when you are ready to capture the request first put Authorize on. There are also some configurations for capturing a request and server status code. Depending on your preference, you can select it.

Here, under the Temporary header box; you need to put the normal user token/cookies/header value that you want to replace within the actual request i.e. if any application is using a JWT token for auth mechanism you need to put that value here.

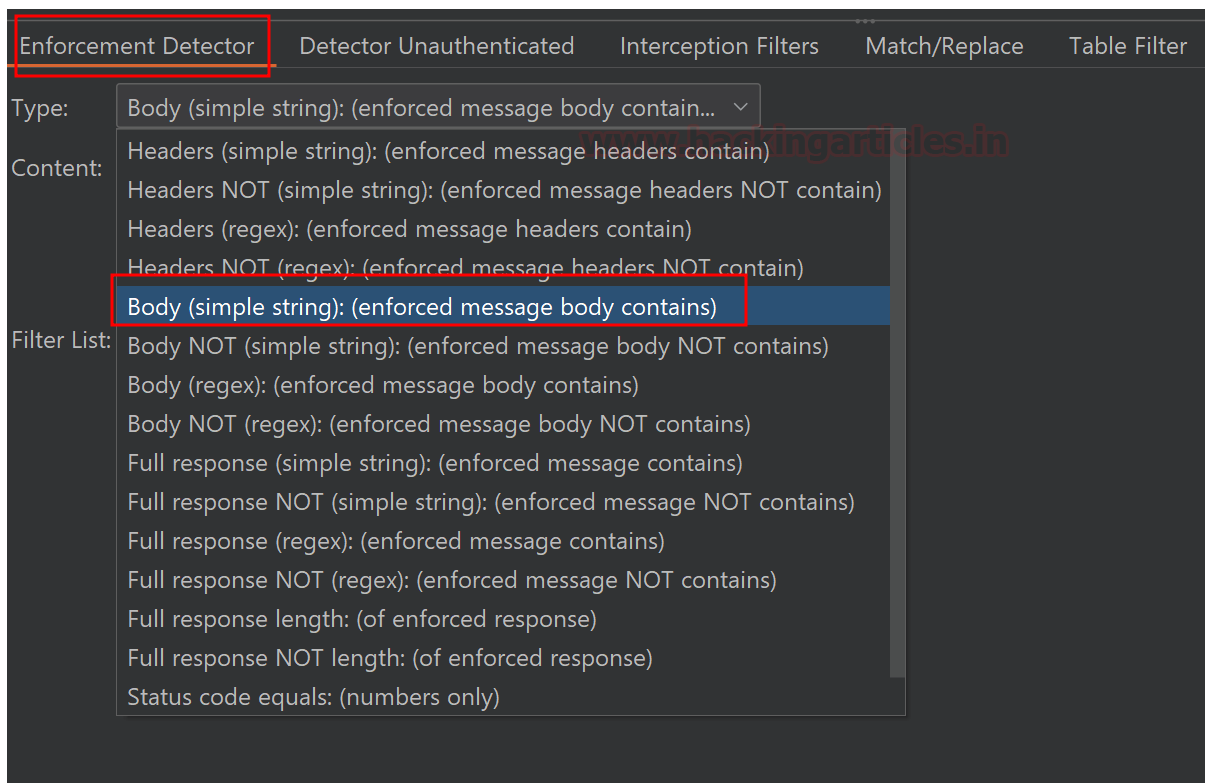
Either you can manually add the auth value or below is the option to fetch it from the last request. If you want to add the cookies header from the last request – click on ‘Fetch Cookies header’ or if you want to add Authorization header – click on ‘Fetch Authorization header’.

Generally, the session cookies are under Cookies Header and the auth token comes under Authorization Header.



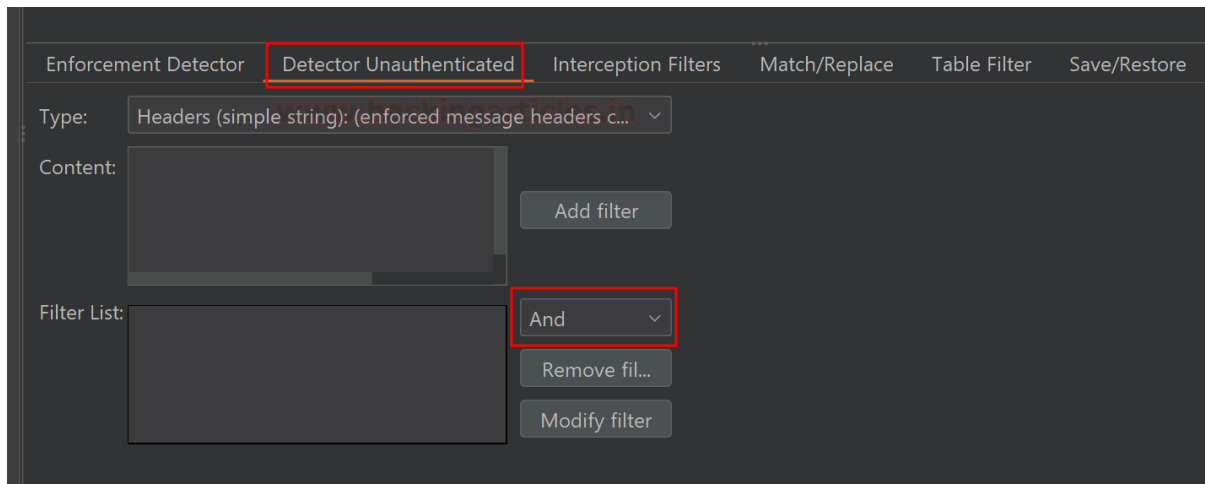
Once the session cookies are loaded, it is essential to instruct Authorize on which requests to intercept and establish the standard behaviour for the application when dealing with unauthorized requests or those with insufficient permissions.

Commencing with the Enforcement Detector, input a characteristic of the application's response that can be anticipated when a user with limited privileges tries to perform an action, they lack sufficient permissions. In my practice, I've found that utilizing the "Body (simple string): enforced message body contains" option is the simplest to set up and functions effectively. Choose the type and content that aligns with your specific needs and remember to click the "Add filter" button.

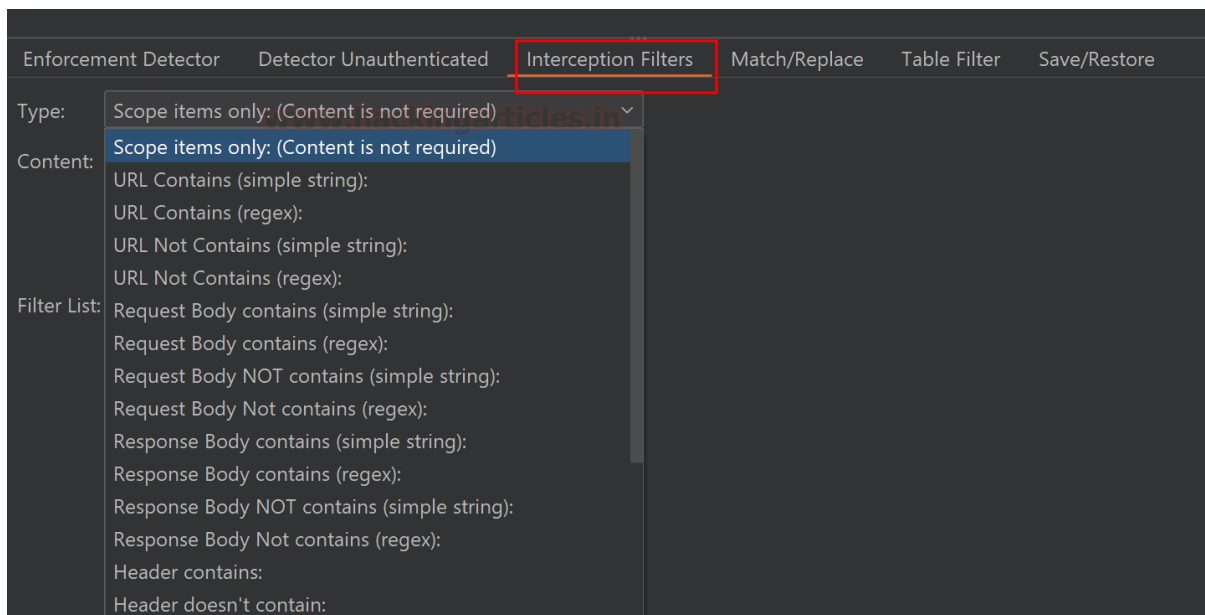


Moreover, it is necessary to understand that it automatically sets the default comparison to "And" when assessing multiple filters. Therefore, if the application generates distinct error messages, such as one for trying to read a file and another for attempting to access administrative features, you should create a filter for each scenario and switch the "And" to "Or."

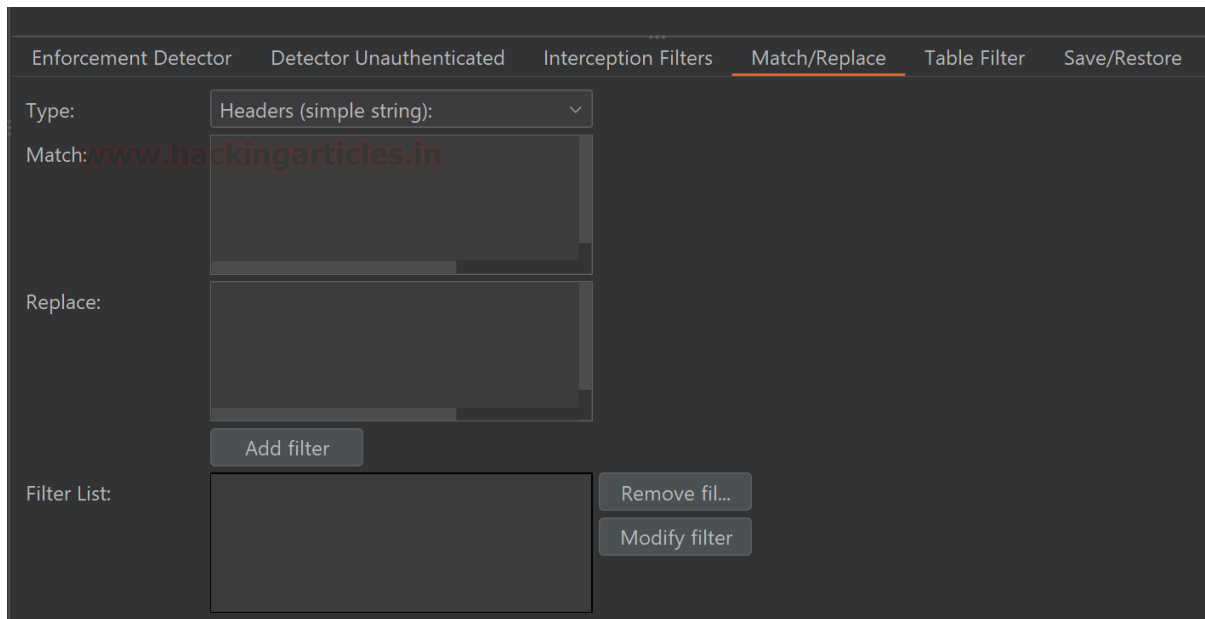
Follow the same procedure for the Unauthenticated Detector



The interception filter will intercept “Scope items only” regardless of content and from those requests, it will ignore spider requests and URLs containing image extensions. You may select on your preference and click “Add filter” when type is selected.

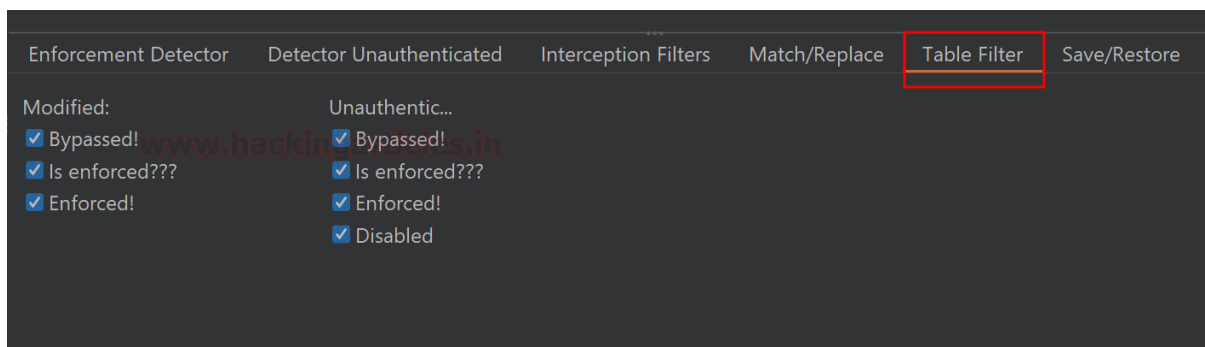


This is another additional feature Match/Replace. You can select it from this site if you need to change any specific header or body parameter on the Authorize request. Suppose there is a parameter name ‘u.name’ on the request body, and it has to be replaced by an Admin EID (i.e.:=”a.name”) for proper access circumvention. You can tell Authorize via adding here.



You can select the type of requests that you want to see under the Table Filter bar,

- bypassed!: the endpoint may be vulnerable to IDOR,
- Is enforced!: endpoint seems to be protected but re-check once,
- Enforcing!: against IDOR, the endpoint is clearly protected.

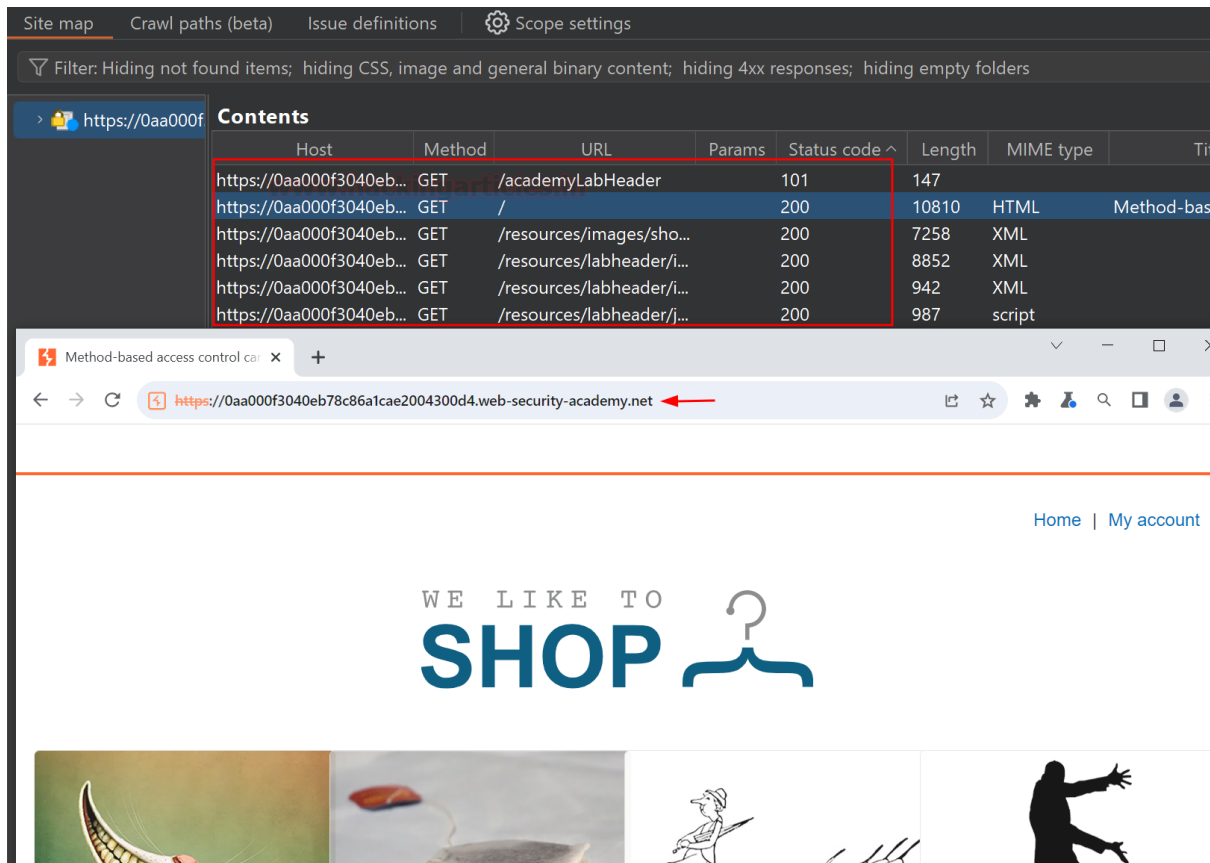


You can save and export the data for further analysis under the Save/Restore tab.

Practical Demonstration of Autorize in Action

Let's do a quick demonstration to understand in an easy way, to perform this practical we are going to use a pre-setup Port Swigger lab "Method-based access control can be circumvented". Click on access the lab and browser the application.

This will show a Broken Access Control vulnerability with two users that have different role higher and lower privilege users. The same concept can be applied to same-level users.



The screenshot shows a Burp Suite interface at the top with a 'Contents' table. Below it is a web browser window displaying a web application.

Host	Method	URL	Params	Status code	Length	MIME type	Title
https://0aa000f3040eb...	GET	/academyLabHeader		101	147		
https://0aa000f3040eb...	GET	/		200	10810	HTML	Method-based
https://0aa000f3040eb...	GET	/resources/images/sho...		200	7258	XML	
https://0aa000f3040eb...	GET	/resources/labheader/i...		200	8852	XML	
https://0aa000f3040eb...	GET	/resources/labheader/i...		200	942	XML	
https://0aa000f3040eb...	GET	/resources/labheader/j...		200	987	script	

The browser window shows the URL `https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net`. The page content includes a navigation bar with 'Home' and 'My account' links, a main heading 'WE LIKE TO SHOP' with a hanger icon, and a row of four images: a green plant, a red flower, a cartoon character, and a silhouette of a person.

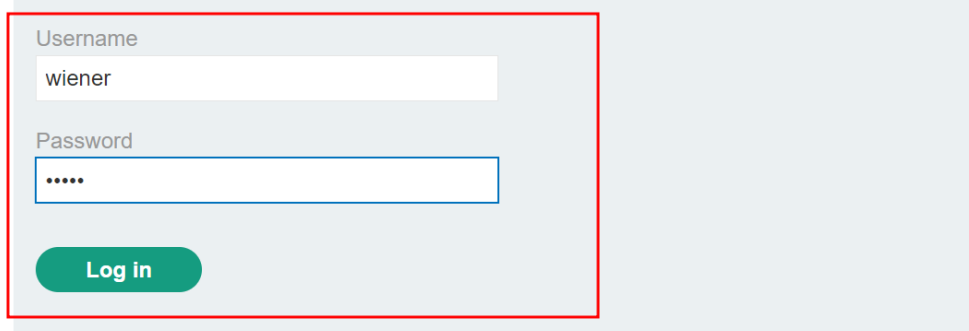
First, we have to capture the cookies for low privileged user (normal user). We are using the default normal user credentials,

Wiener:peter

And logged into the application to capture session cookie.



Login

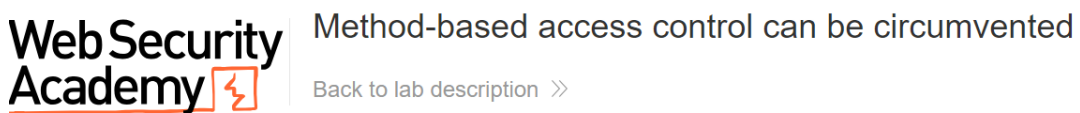


Username

Password

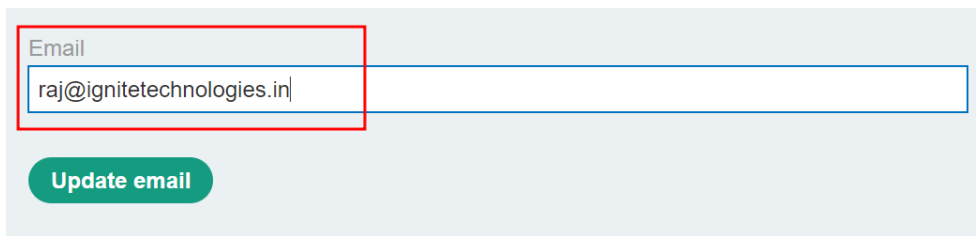
[Log in](#)

Updated some more details.



My Account

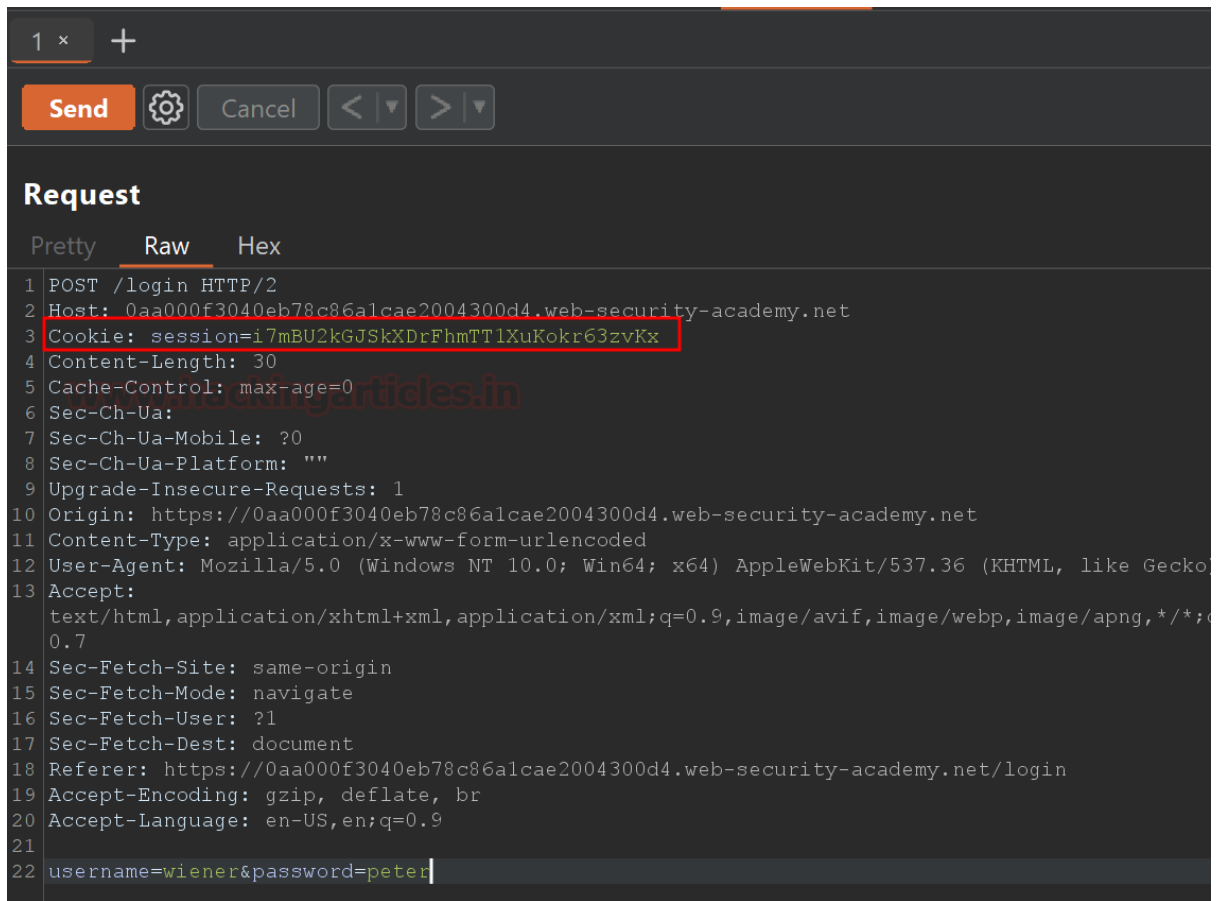
Your username is: wiener



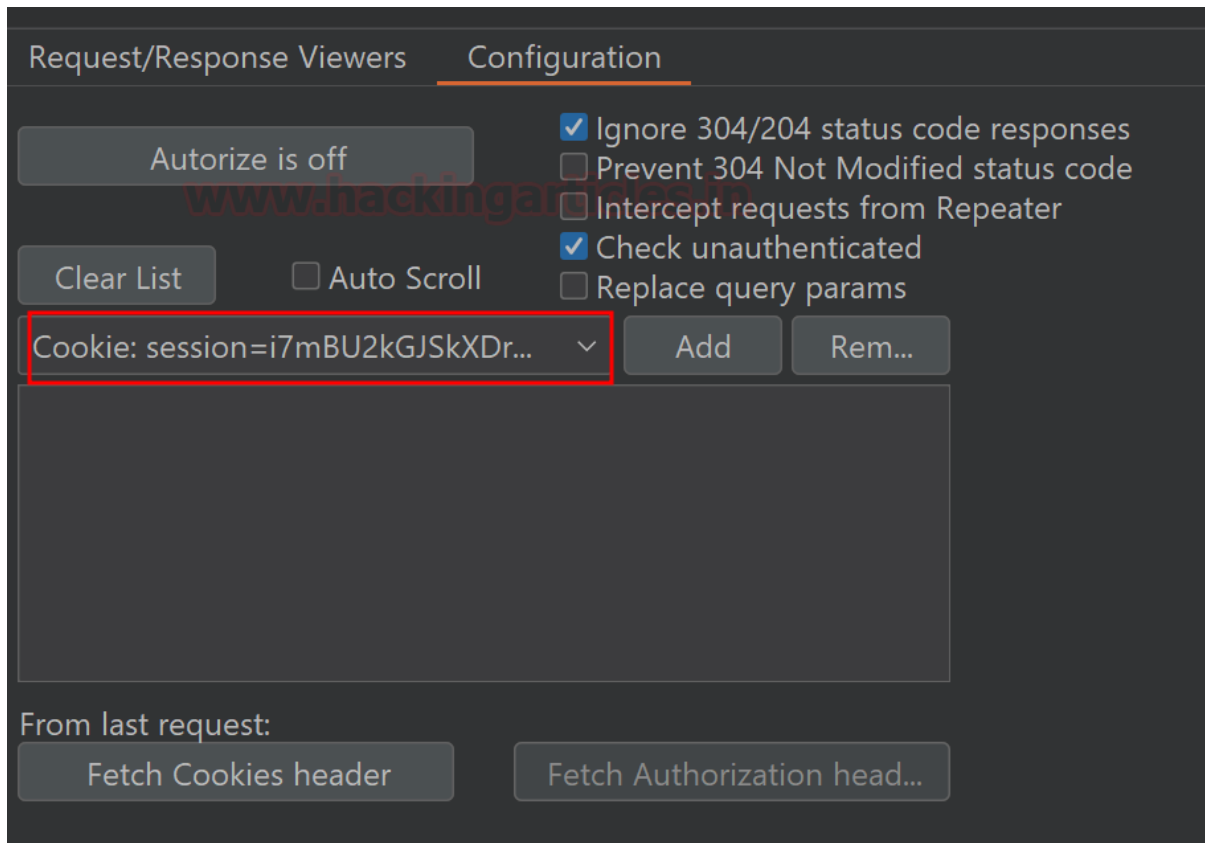
Email

[Update email](#)

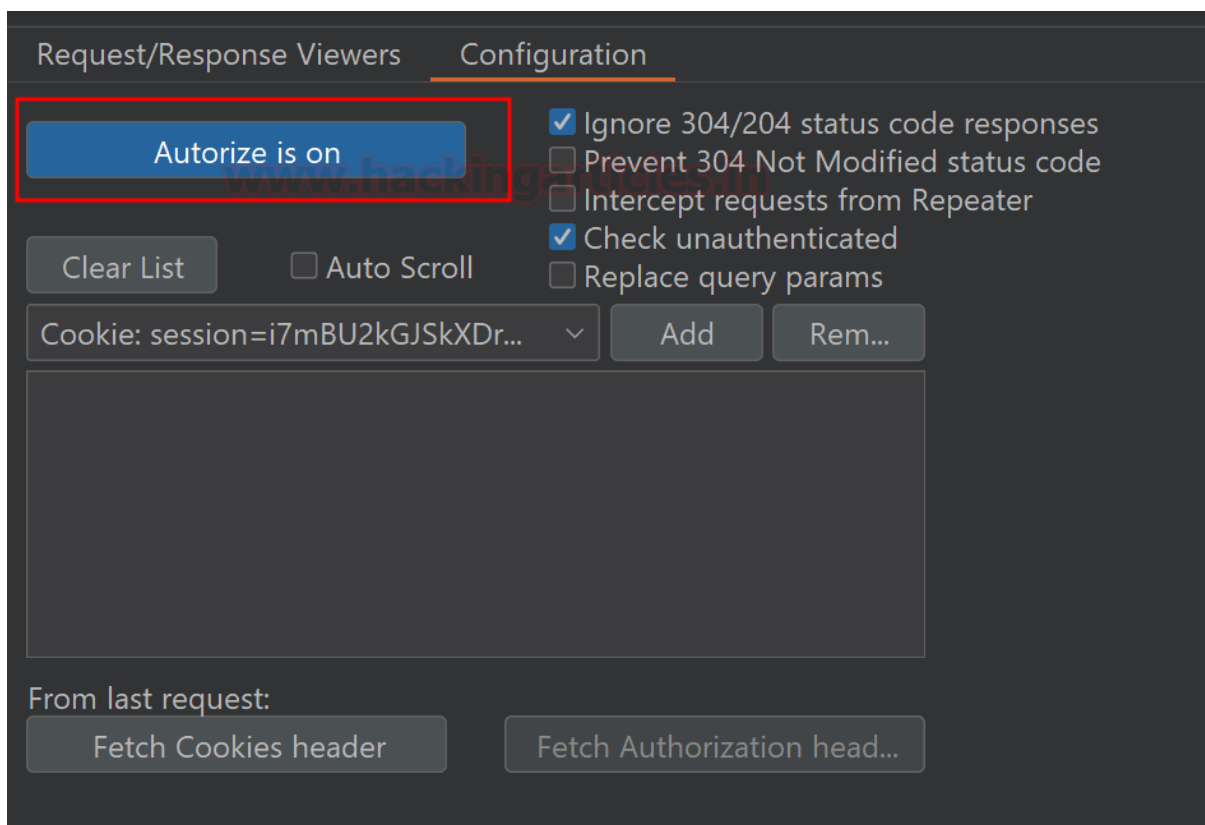
You will see the below capture session cookie in to the login request. Now copy this cookie header.



Add this cookie header value to Authorize tab as shown below,



And keep Authorize on.



In order to, check the auth bypass now we have to log in with high privilege (admin user). Go to login page again and use admin credentials to log in,

Administrator:admin



<https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net/login>

Web Security
Academy 

Method-based access control can be circum

[Back to lab description >>](#)

Login

Username

administrator

Password

....

Log in

After successfully logging in and browsing the all admin-only URLs. You can see under the Authorize tab some highlighted requests

The **Authz. Status** indicates which endpoints are accessible to wiener (normal user).

The **Unauth. Status** pertains to unauthorized users, effectively eliminating the cookie and all authorization headers. You can opt to disable this feature by deselecting the “Check unauthenticated” option in the Authorize configuration tab.

Red [Bypassed!] : endpoint could be vulnerable to access control/IDOR issues.

Orange [Is enforced!] : endpoint seems to be protected but cross-check manually by replacing the cookies value.

Green [Enforced!] : endpoint is clearly protected against access control/IDOR issues.

...	URL	Authz. Status	Unauth. Status
1 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/login	0	0	Bypassed!	Bypassed!
2 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/my-account?id=administrator	Bypassed!	Bypassed!
3 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/academyLabHeader	0	...	Enforced!	Enforced!
4 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/admin	Bypassed!	Bypassed!
5 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/academyLabHeader	0	...	Enforced!	Enforced!
6 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/admin-roles	0	0	Bypassed!	Bypassed!
7 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/admin	Bypassed!	Bypassed!
8 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/academyLabHeader	0	...	Enforced!	Enforced!
9 ...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/admin-roles	0	0	Bypassed!	Bypassed!
...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/admin	Bypassed!	Bypassed!
...	https://0aa000f3040eb78c86a1cae2004300d4.web-security-academy.net:443/academyLabHeader	0	...	Enforced!	Enforced!

As visible in above image, request 1, 2, 6, and 7 are having Broken access control issue.

Keep in mind that do not blindly follow the Authorize result, The Red highlight requests do not mean that all endpoints are vulnerable or bypassed. There may be false positives; You must do a cross-check.

Some other possible scenarios, suppose you are testing auth issues with the two same level of users. As a result, you will see Authz. Status shows Bypassed! And Unauth. Status shows Enforced! In that case improper authorization can be found on the request which shows that the specific endpoint can be accessed by the 2nd user but has correctly implemented authorization for any unauthorized users.

When you select any highlighted request, on the right side you will see the detailed information about modified, original & unauthenticated request and responses.



Organizer Extensions Learn Autorize

Request/Response Viewers Configuration

Modified Request Modified Response Expand

Pretty Raw ex

```

1 POST /my-account/change-email HTTP/2
2 Host: 0a2000f4041f53818018353200cb00fd.web-security-academy.net
3 Content-Length: 34
4 Cache-Control: max-age=0
5 Sec-Ch-Ua:
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: ""
8 Upgrade-Insecure-Requests: 1
9 Origin: https://0a2000f4041f53818018353200cb00fd.web-security-academy.net
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: https://0a2000f4041f53818018353200cb00fd.web-security-academy.net/my-ac
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-US,en;q=0.9
20 Cookie: session=e02BUCQvmsEIqb87XJBCiACSHsdXD9t1$
21

```

? ⚙️ ⬅️ ➡️ Search

Original Request Original Response Expand

Pretty Raw Hex

```

1 POST /my-account/change-email HTTP/2
2 Host: 0a2000f4041f53818018353200cb00fd.web-security-academy.net
3 Cookie: session=dxqZCSrDns2hw1OFoRl866ROgo4Gv6Cb
4 Content-Length: 34
5 Cache-Control: max-age=0
6 Sec-Ch-Ua:
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: ""
9 Upgrade-Insecure-Requests: 1
10 Origin: https://0a2000f4041f53818018353200cb00fd.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a2000f4041f53818018353200cb00fd.web-security-academy.net/my-ac
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-US,en;q=0.9
21

```

? ⚙️ ⬅️ ➡️ Search

Unauthenticated Request Unauthenticated Response Expand

Pretty Raw Hex

```

1 POST /my-account/change-email HTTP/2
2 Host: 0a2000f4041f53818018353200cb00fd.web-security-academy.net
3 Content-Length: 34
4 Cache-Control: max-age=0
5 Sec-Ch-Ua:
6 Sec-Ch-Ua-Mobile: ?0
7 Sec-Ch-Ua-Platform: ""
8 Upgrade-Insecure-Requests: 1
9 Origin: https://0a2000f4041f53818018353200cb00fd.web-security-academy.net
10 Content-Type: application/x-www-form-urlencoded
11 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
12 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-User: ?1
16 Sec-Fetch-Dest: document
17 Referer: https://0a2000f4041f53818018353200cb00fd.web-security-academy.net/my-ac
18 Accept-Encoding: gzip, deflate, br
19 Accept-Language: en-US,en;q=0.9
20
21 email=rrai%40ignitetechnologies.in

```

? ⚙️ ⬅️ ➡️ Search



Burpsuite for Pentester: Logger++

Suppose you are a web explorer, and you want to know everything about a website. Burp Logger++ is like your trusty notebook. It is super helpful because it has a magical filter. You can tell it what kind of information you are looking for, and it will only show you those things.

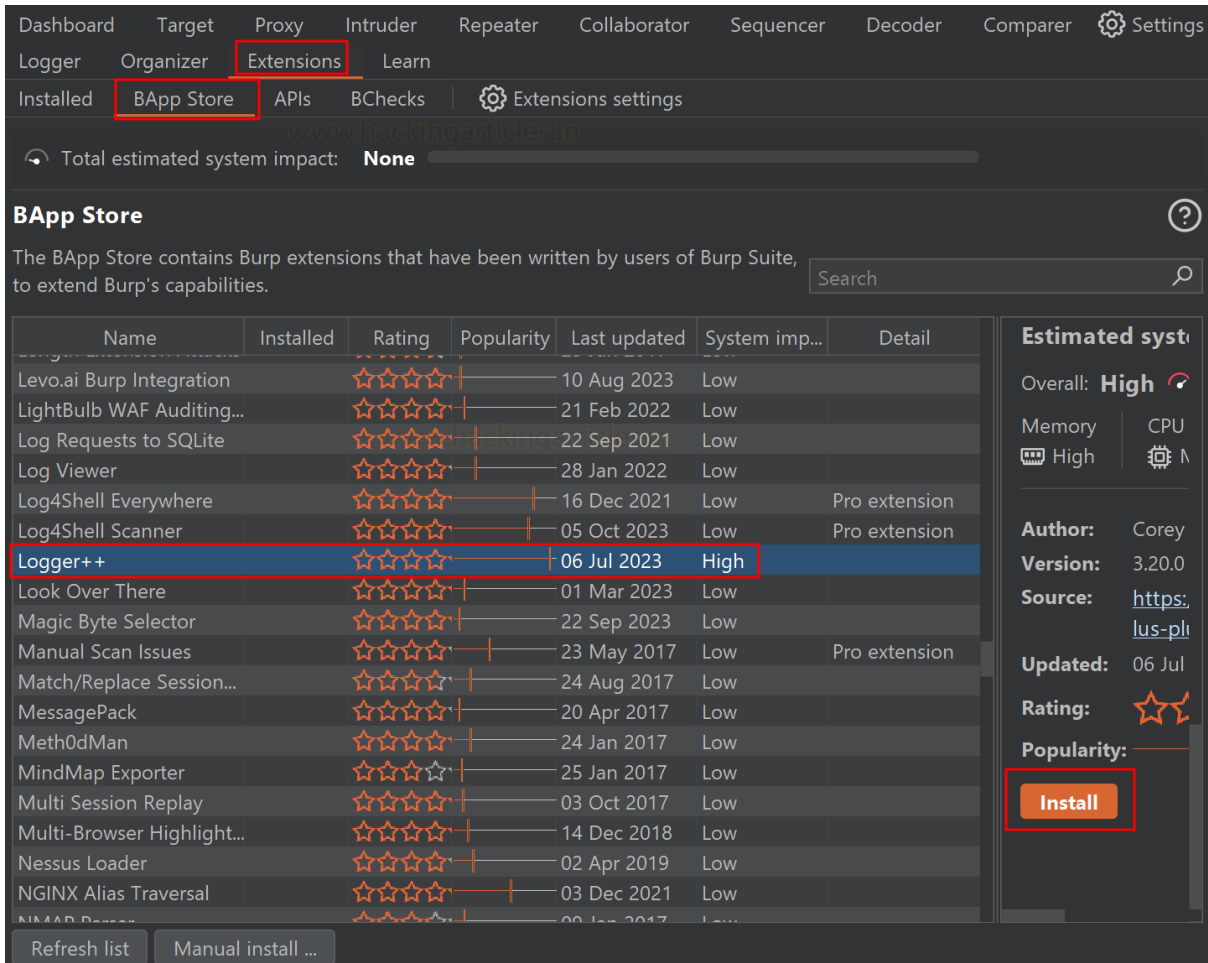
With Burp Logger++, you can also color-code things. Think of it like using different colors to highlight the most important parts of a picture. This helps you spot the important stuff quickly.

- **Setting Up & Navigating**
- **Query-Based Filter**
- **Filter Library**
- **Regex-Based Filter**
- **Export Data Feature**

Setting Up & Navigating

You can download and install the extension from the BApp Store. Go to Extensions > Bapp Store. Here, search for Logger++ or simply scroll down.

Click on it, on the right side scroll down and install it.



Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Settings

Logger Organizer Extensions Learn

Installed BApp Store APIs BChecks Extensions settings

Total estimated system impact: **None**

BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Name	Installed	Rating	Popularity	Last updated	System imp...	Detail
Levo.ai Burp Integration		☆☆☆☆		10 Aug 2023	Low	
LightBulb WAF Auditing...		☆☆☆☆		21 Feb 2022	Low	
Log Requests to SQLite		☆☆☆☆		22 Sep 2021	Low	
Log Viewer		☆☆☆☆		28 Jan 2022	Low	
Log4Shell Everywhere		☆☆☆☆		16 Dec 2021	Low	Pro extension
Log4Shell Scanner		☆☆☆☆		05 Oct 2023	Low	Pro extension
Logger++		☆☆☆☆		06 Jul 2023	High	
Look Over There		☆☆☆☆		01 Mar 2023	Low	
Magic Byte Selector		☆☆☆☆		22 Sep 2023	Low	
Manual Scan Issues		☆☆☆☆		23 May 2017	Low	Pro extension
Match/Replace Session...		☆☆☆☆		24 Aug 2017	Low	
MessagePack		☆☆☆☆		20 Apr 2017	Low	
MethOdMan		☆☆☆☆		24 Jan 2017	Low	
MindMap Exporter		☆☆☆☆		25 Jan 2017	Low	
Multi Session Replay		☆☆☆☆		03 Oct 2017	Low	
Multi-Browser Highlight...		☆☆☆☆		14 Dec 2018	Low	
Nessus Loader		☆☆☆☆		02 Apr 2019	Low	
NGINX Alias Traversal		☆☆☆☆		03 Dec 2021	Low	

Refresh list Manual install ...

Estimated system

Overall: **High**

Memory: High CPU: N

Author: Corey

Version: 3.20.0

Source: <https://lus-pli>

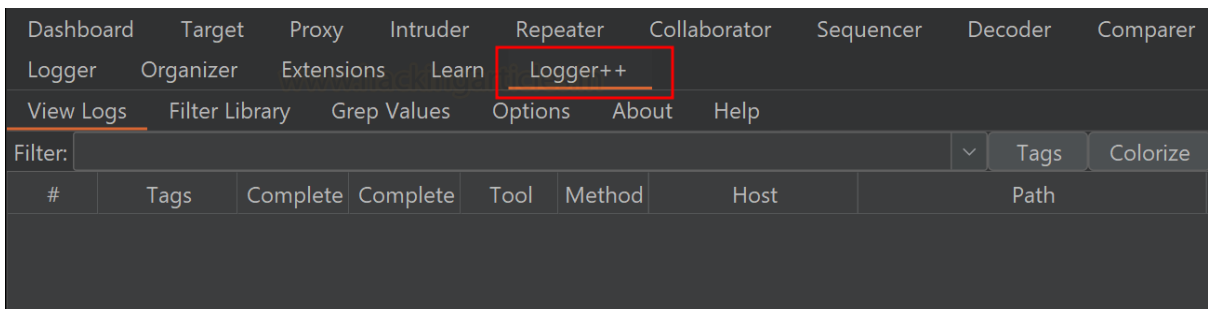
Updated: 06 Jul

Rating: ☆☆☆

Popularity:

Install

After successful installation, it will appear on the toolbar.



Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer

Logger Organizer Extensions Learn **Logger++**

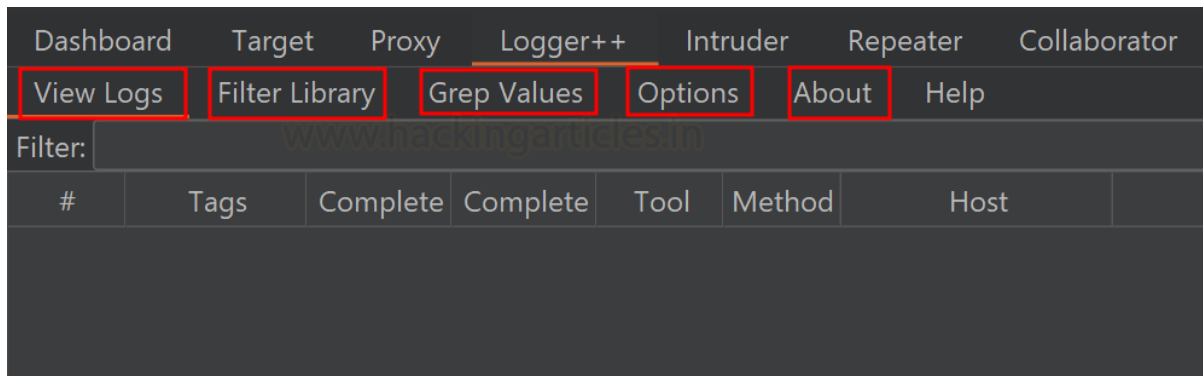
View Logs Filter Library Grep Values Options About Help

Filter: Tags Colorize

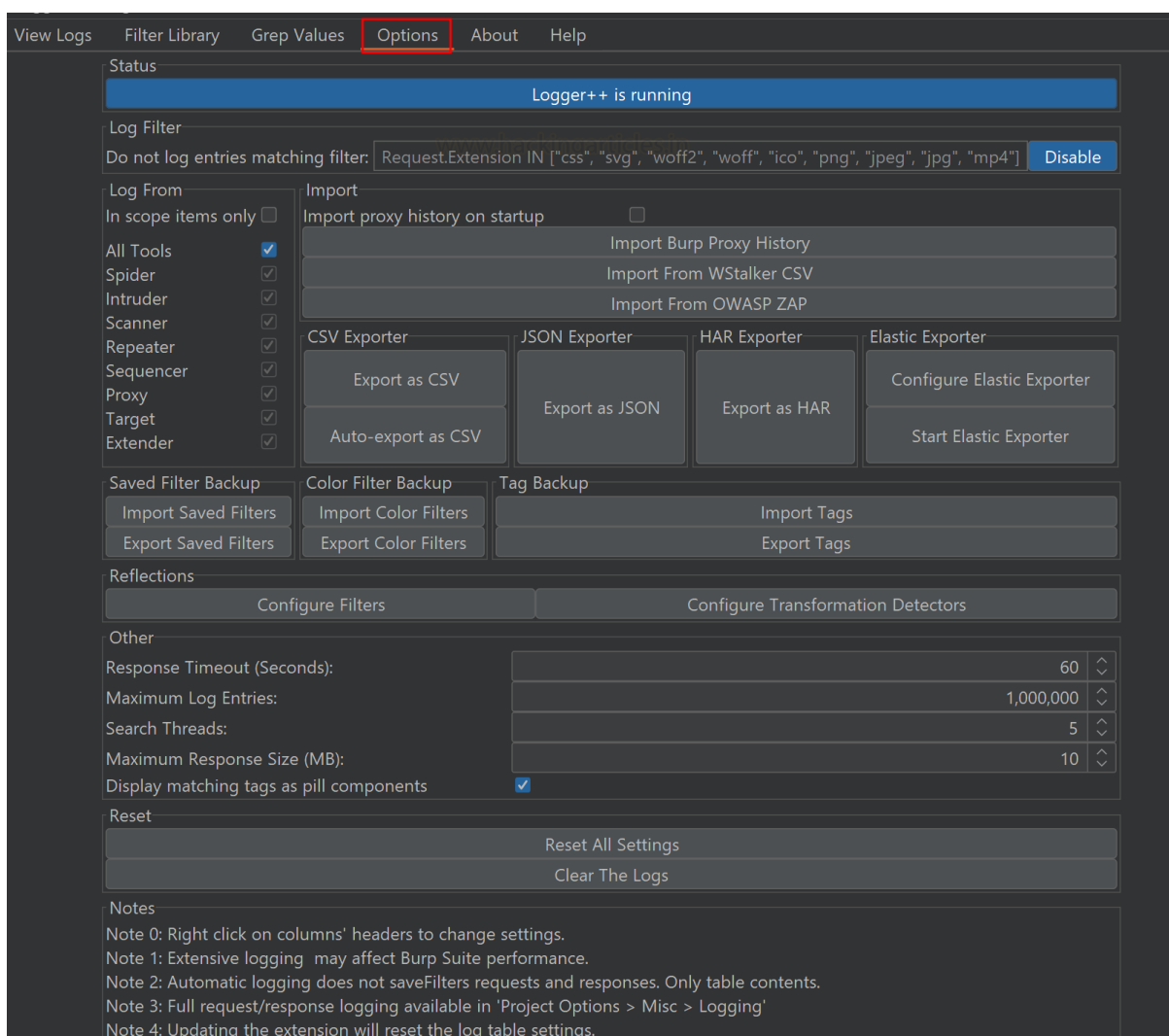
#	Tags	Complete	Complete	Tool	Method	Host	Path
---	------	----------	----------	------	--------	------	------

Navigating

There are a lot of options visible to you. First, let's explore the "Options" tab to discover what advanced settings are included in this extension.



Navigate to “Options” to see the various log filter options. It allows you to customize logging setting as per your preference.



Logger++ is running by default. Here are some other important settings:

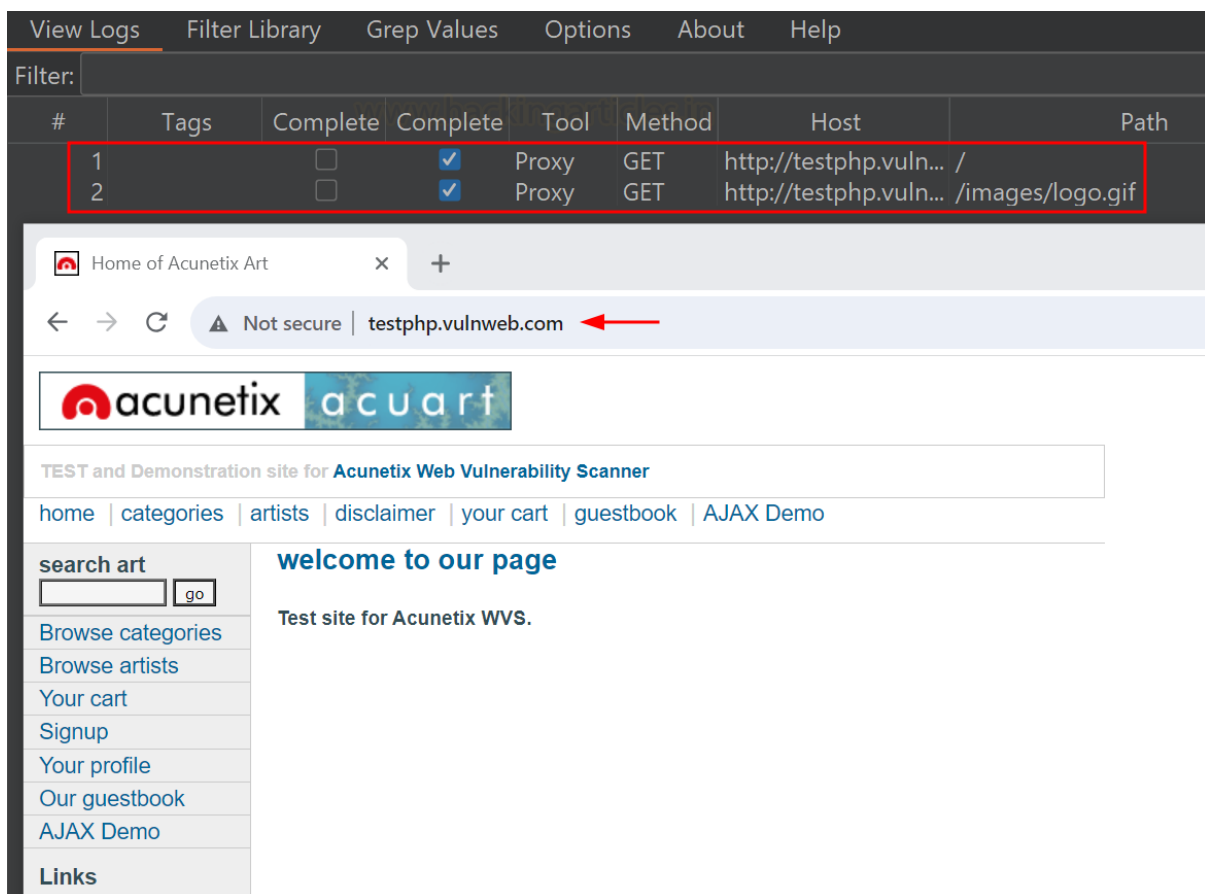
- **Log Filter:** This feature lets you specifically choose the requests that you don't need to record for analysis, or you may turn it off when not in use.
- **Log From:** It enables you to capture data from the specific logs that you want to capture.

- Import: You can import log data from CSV and OWASP ZAP reports with this function.
- Export: The log data can be exported for further analysis.

Depending on your preferences, you can use different configurations. We are sticking with the default settings for the time being.

Query-Based Filter

The View Log tab contains all the logs. Using this website “vulnweb” as an example, browse it and simply scan the entire site; all logs will show up here under the View Logs page.



The screenshot displays the 'View Logs' interface. At the top, there are tabs: 'View Logs', 'Filter Library', 'Grep Values', 'Options', 'About', and 'Help'. Below these is a 'Filter:' input field. A table of logs is shown with columns: '#', 'Tags', 'Complete', 'Complete', 'Tool', 'Method', 'Host', and 'Path'. Two log entries are visible, both with 'Complete' checked and 'Tool' set to 'Proxy'. The first entry has a path of '/', and the second has a path of '/images/logo.gif'. Below the table is a browser window showing the 'Home of Acunetix Art' website. The browser's address bar shows 'testphp.vulnweb.com' with a red arrow pointing to it. The website content includes the 'acunetix' and 'acuart' logos, a navigation menu, and a 'welcome to our page' message.

#	Tags	Complete	Complete	Tool	Method	Host	Path
1		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Proxy	GET	http://testphp.vuln...	/
2		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Proxy	GET	http://testphp.vuln...	/images/logo.gif

Now, go to Signup. To capture the logs for credentials, enter the test login details.

Username: test

Password: test



then click on “Login”.

Filter:

#	Tags	Complete	Complete	Tool	Method	Host
1		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Proxy	GET	http://testphp.vuln... /
2		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Proxy	GET	http://testphp.vuln... /images/log
3		<input type="checkbox"/>	<input checked="" type="checkbox"/>	Proxy	GET	http://testphp.vuln... /categories

login page

Not secure | testphp.vulnweb.com/login.php

 acunetix 

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#)

search art

[Browse categories](#)
[Browse artists](#)
[Your cart](#)
[Signup](#)
[Your profile](#)
[Our guestbook](#)
[AJAX Demo](#)

Links
[Security art](#)
[PHP scanner](#)
[PHP vuln help](#)
[Fractal Explorer](#)

If you are already registered please enter your login information below:

Username :
Password :

You can also [signup here](#).
Signup disabled. Please use the username **test** and the password **test**.

Let's update some more details to capture more requests for further analysis.



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
Logout test

search art

[Browse categories](#)

[Browse artists](#)

[Your cart](#)

[Signup](#)

[Your profile](#)

[Our guestbook](#)

[AJAX Demo](#)

Links

[Security art](#)

[PHP scanner](#)

[PHP vuln help](#)

[Fractal Explorer](#)

(test)

On this page you can visualize or edit you user information.

Name:

Credit card number:

E-Mail:

Phone number:

Address:

You have 3 items in your cart. You visualize you cart [here](#).

About Us | Privacy Policy | Contact Us | ©2016 Acunetix Ltd

You can see that all requests have been captured here in View Logs.

#	Method	Host	Title	Path	New Cookies	Inferred Type	Reflected Params	Response Length	Parameter Count	Status
1	GET	http://testphp.vuln...	Home of Acu...	/		HTML		4958	0	200
2	GET	http://testphp.vuln...		/images/logo.gif		IMAGE_GIF		6660	0	200
3	GET	http://testphp.vuln...	picture categ...	/categories.php		HTML		6115	0	200
4	GET	http://testphp.vuln...	artists	/artists.php		HTML		5328	0	200
5	GET	http://testphp.vuln...	you cart	/cart.php		HTML		4903	0	200
6	GET	http://testphp.vuln...		/userinfo.php		PLAIN_TEXT		14	0	302
7	GET	http://testphp.vuln...	login page	/login.php		HTML		5523	0	200
8	GET	http://testphp.vuln...	login page	/login.php		HTML		5523	0	200
9	GET	http://testphp.vuln...	guestbook	/guestbook.php		HTML		5390	0	200
10	GET	http://testphp.vuln...		/images/remark.gif		IMAGE_GIF		79	0	200
11	GET	http://testphp.vuln...	ajax test	/AJAX/index.php		HTML		4236	0	200
12	GET	http://testphp.vuln...	login page	/login.php		HTML		5523	0	200
13	POST	http://testphp.vuln...	user info	/userinfo.php	[login=test%2Ft...	HTML	[uname, pass]	5980	2	200
14	POST	https://passwordslea...		/v1/leaks/lookupSingle		SCRIPT		179	5	400
15	POST	http://testphp.vuln...	user info	/userinfo.php		HTML	[uname, ucc, uemai...	6009	6	200

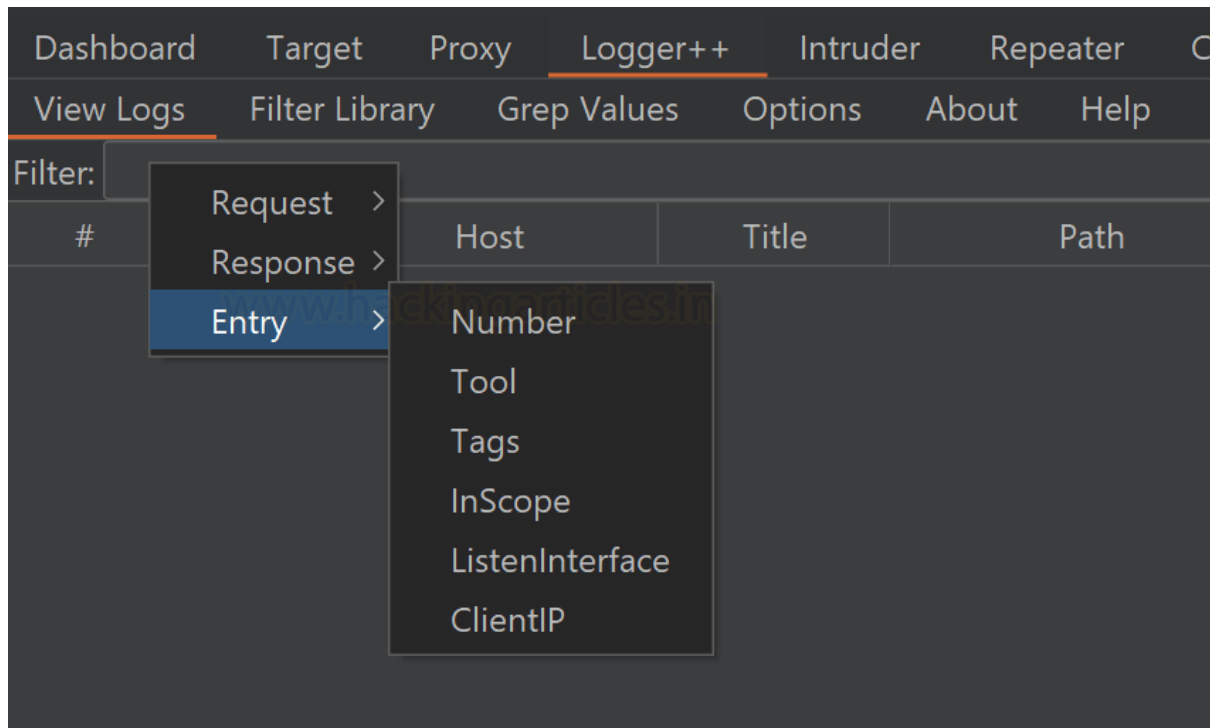
Magical Filter

You can use filter to selectively view or manipulate HTTP requests and responses. These filters help you focus on specific aspects of the web traffic and are especially useful during

security testing. The working is based on query string. It accepts a logical query and returns output based on them.

You have some advanced choices with the filter options:

Entry: You can apply filters according to number, tool, tags, InScope, and other criteria.



Request: It lets you filter just the request itself using many options such as header, body, URL, method, parameters, cookies, etc. As shown below:



View Logs Filter Library Grep Values Options About Help

Filter:

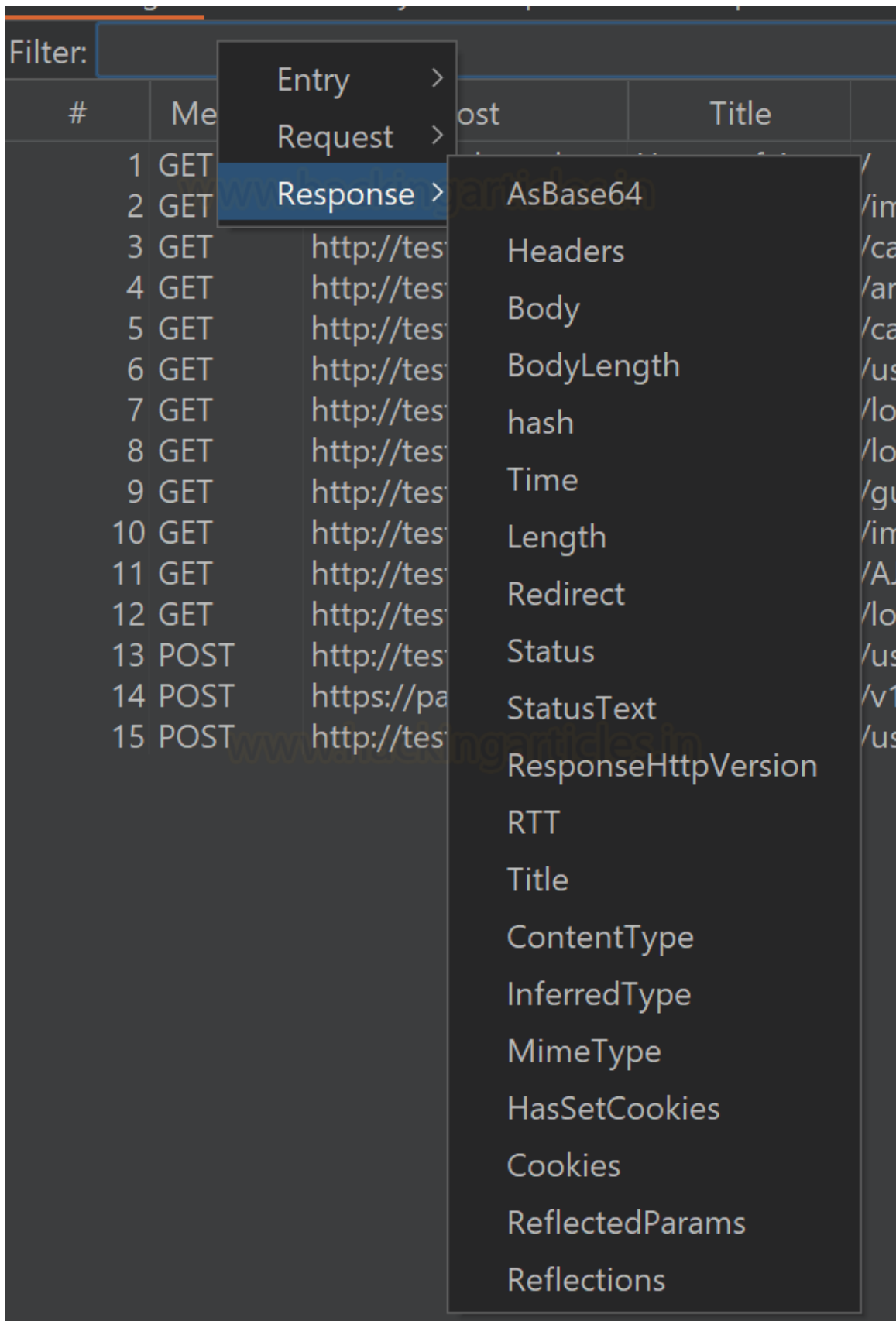
#	Method	Host	Entry
1	GET	http://testphp.v	Request
2	GET	http://testphp.v	Response
3	GET	http://testphp.vulnw... picture c	AsBase64
4	GET	http://testphp.vulnw... artists	Headers
5	GET	http://testphp.vulnw... you cart	Body
6	GET	http://testphp.vulnw...	BodyLength
7	GET	http://testphp.vulnw... login pag	Time
8	GET	http://testphp.vulnw... login pag	Length
9	GET	http://testphp.vulnw... guestbo	Tool
10	GET	http://testphp.vulnw...	Comment
11	GET	http://testphp.vulnw... ajax test	Complete
12	GET	http://testphp.vulnw... login pag	URL
13	POST	http://testphp.vulnw... user info	Method
14	POST	https://passwordslea...	Path
15	POST	http://testphp.vulnw... user info	Query

PathQuery
Protocol
IsSSL
UsesCookieJar
Hostname
Host
Port
ContentType
RequestHttpVersion
Extension
Referrer
HasParams
HasGetParam
HasPostParam
HasSentCookies
CookieString
ParameterCount
Parameters
Origin

Pretty Raw Hex

1

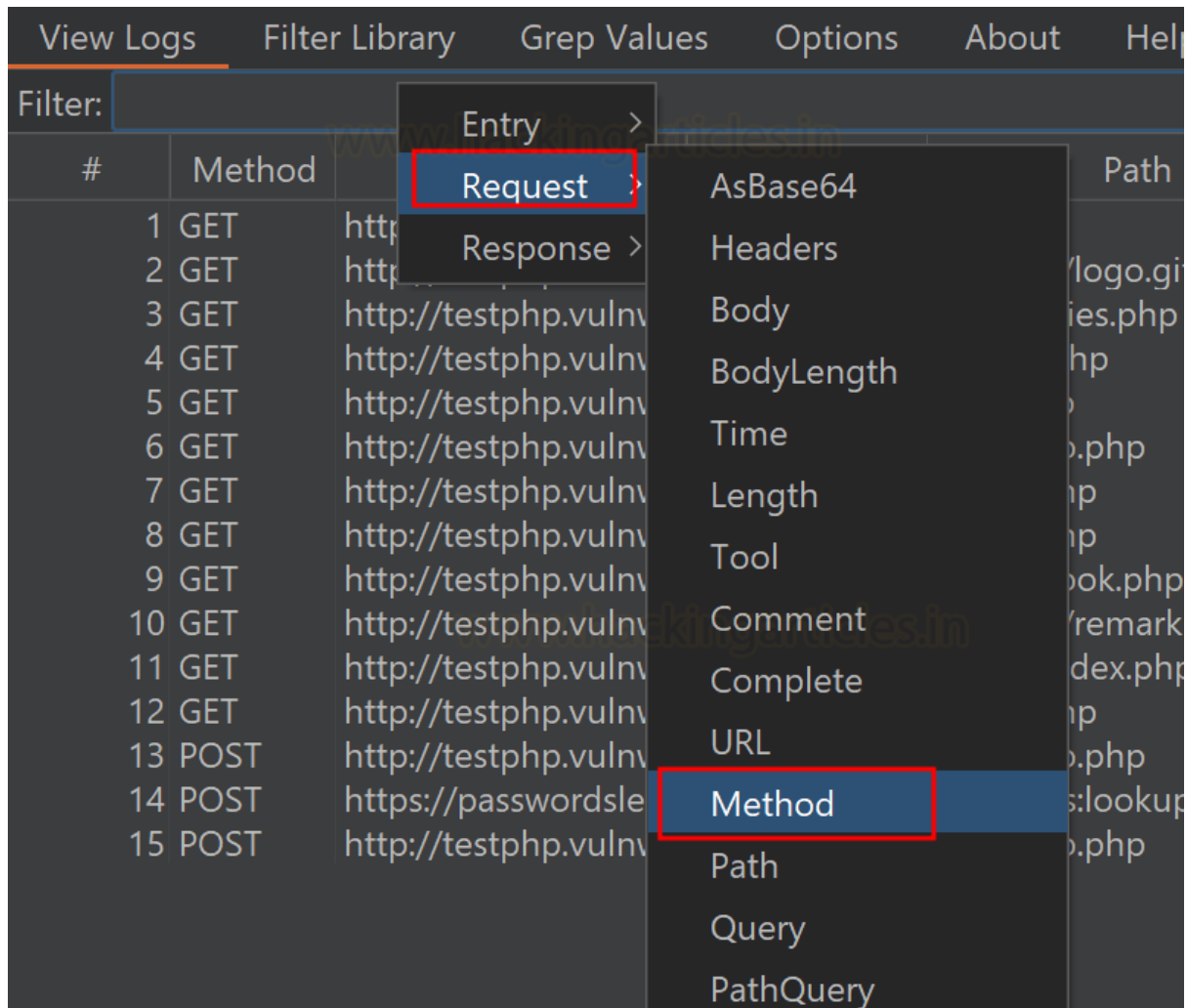
Response: It lets you filter just the response by using various options such as header, body, Inferred Type, Method, Parameters, cookies etc. As shown below:



Scenario 1: Let's suppose you just want to view HTTP POST requests from all logs. It is understood that HTTP POST parameters are in HTTP Request.

Go to Filter bar > right click > Select Request > Select Method

The method has been chosen and is visible in the filter bar.



`Request.Method == "POST"`

And hit enter. As result, Only HTTP POST Method requests appear.

View Logs Filter Library Grep Values Options About Help					
Filter: Request.Method == "POST" ←					
#	Method	Host	Title	Path	New Cookies
13	POST	http://testphp.vulnw...	user info	/userinfo.php	[login=test%2Ft
14	POST	https://passwordslea...		/v1/leaks:lookupSingle	[]
15	POST	http://testphp.vulnw...	user info	/userinfo.php	[]

Scenario 2: Taking another example, suppose we just want to view the requests which contains any username information from all logs.

Go to Filter bar > right click > Select Request > Select Body

Request.Body CONTAINS "uname"

As a result, the following request is highlighted:

View Logs Filter Library Grep Values Options About Help					
Filter: Request.Body CONTAINS "uname" ←					
#	Method	Host	Title	Path	New Cookies
13	POST	http://testphp.vulnw...	user info	/userinfo.php	[login=test%

Pretty Raw Hex		
1	POST /userinfo.php HTTP/1.1	
2	Host: testphp.vulnweb.com	
3	Content-Length: 20	
4	Cache-Control: max-age=0	
5	Upgrade-Insecure-Requests: 1	
6	Origin: http://testphp.vulnweb.com	
7	Content-Type: application/x-www-form-urlencoded	
8	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4399.24 Safari/537.36	
9	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8	
10	Referer: http://testphp.vulnweb.com/login.php	
11	Accept-Encoding: gzip, deflate, br	
12	Accept-Language: en-US,en;q=0.9	
13	Connection: close	
14		
15	uname=test&pass=test	

Below are some useful queries which are helpful during penetration testing.

JSON Injection (Check for only JSON request)

Response.InferredType == "json"



Injection Attack (Check for HTML, XML, JSON)

Response.InferredType IN ["json", "html", "xml"]

Sensitive File Exposed

Response.Body CONTAINS [".git", ".config", ".zip", ".swf", ".doc", ".pdf", ".xlsx", ".csv",]

Sensitive Path Exposed

Request.Path CONTAINS ["/git", "/etc", "/var"]

Request.Path MATCHES "/account*"

Sensitive Parameter in Query String

Request.Path CONTAINS ["id", "username", "password", "role", "isAdmin"]

Sensitive Parameter in Request

Request.Body CONTAINS ["id", "username", "password", "token", "role", "EnterpriseID", "isAdmin"]

Server Information Disclosed

Response.header CONTAINS "Server:"

CORS Misconfiguration

Response.Header MATCHES "Access-Control-Allow-Origin: *"

Check for CSRF Token

Request.Method == "POST" AND Request.Body CONTAINS "csrf"

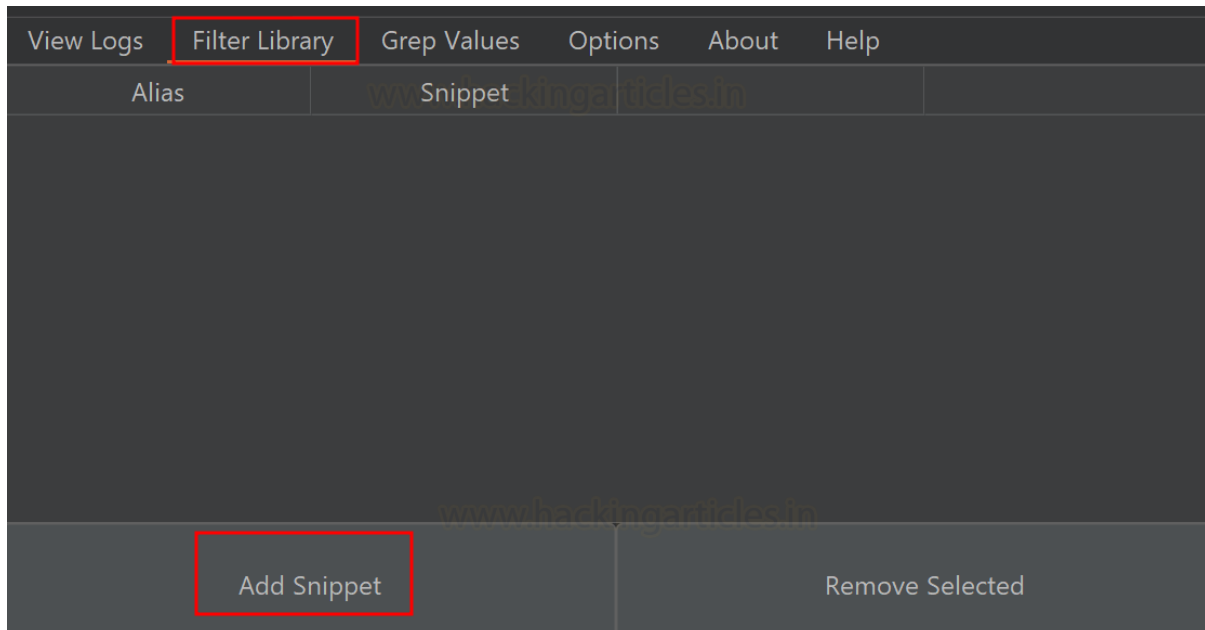
Missing Robots.txt

Request.Path MATCHES "/robots.txt"

URL Redirection

Request.Path CONTAINS ["redirect=", "page=", "url=", "index.page="]

We can use the saved or pre-configured filters from the library directly with the help of the Filter Library. When you start testing, you do not have to manually type or remember the query string of filter pattern.

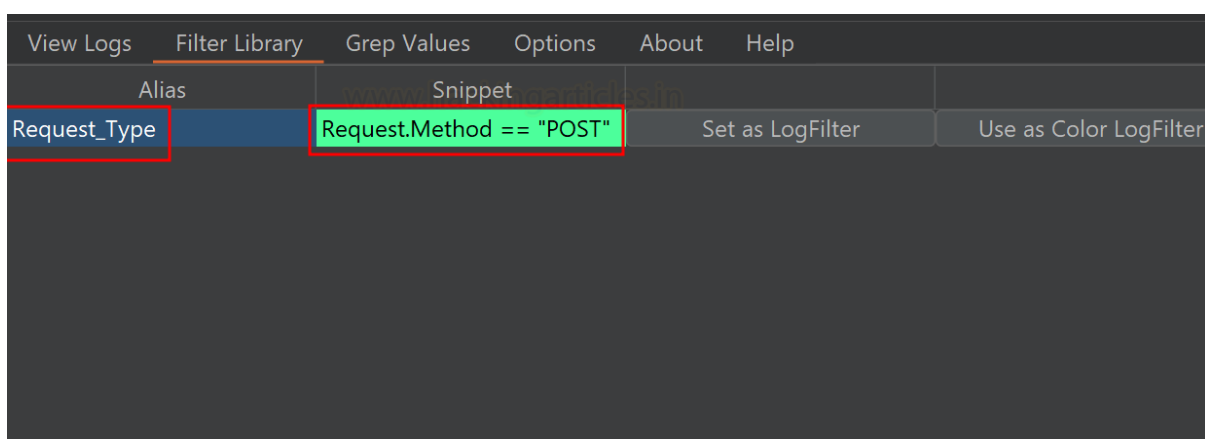


Click on “Add Snippet”. Here are two values that must be added.

- **Alias:** Put any Alias name for your query string.
- **Snippet:** Add query string here.

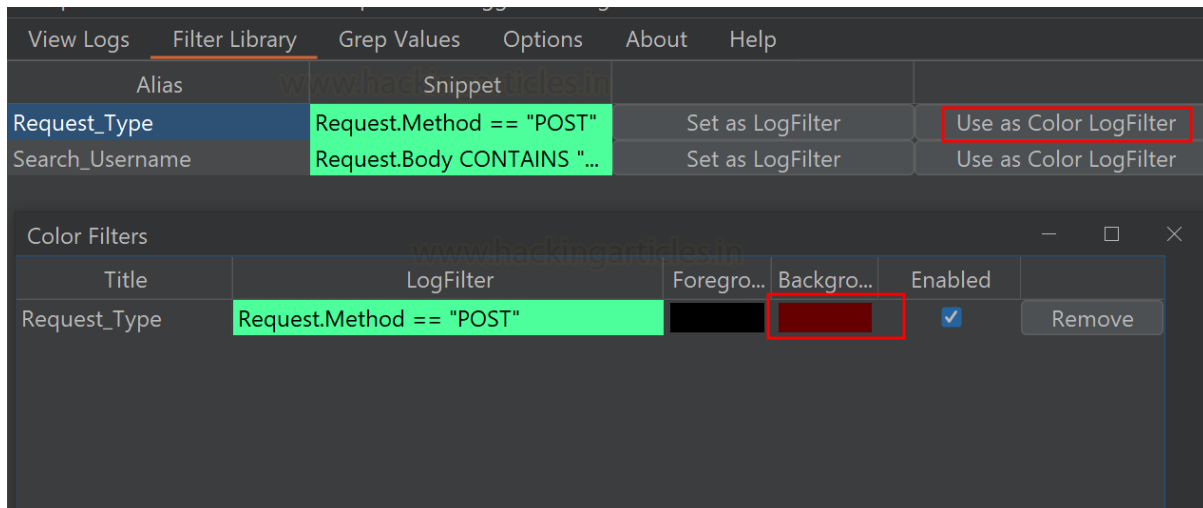
As you can see below, I have added a filter for

`Request_Type: Request.Method == "POST"`



You no longer have to type repeatedly to find only POST requests. You can color-code this request so that the highlighted request stands out among all captured requests on the View Logs page.

Click on Use as **Color LogFilter** > **Select Background Color** > check Enable and save it.



All POST requests are now highlighted in “Dard-Red” on this page.

Filter:						
#	Method	Host	Title	Path	New Cookies	
1	GET	http://testphp.vulnw...	Home of Acu...	/	[]	
2	GET	http://testphp.vulnw...		/images/logo.gif	[]	
3	GET	http://testphp.vulnw...	picture categ...	/categories.php	[]	
4	GET	http://testphp.vulnw...	artists	/artists.php	[]	
5	GET	http://testphp.vulnw...	you cart	/cart.php	[]	
6	GET	http://testphp.vulnw...		/userinfo.php	[]	
7	GET	http://testphp.vulnw...	login page	/login.php	[]	
8	GET	http://testphp.vulnw...	login page	/login.php	[]	
9	GET	http://testphp.vulnw...	guestbook	/guestbook.php	[]	
10	GET	http://testphp.vulnw...		/images/remark.gif	[]	
11	GET	http://testphp.vulnw...	ajax test	/AJAX/index.php	[]	
12	GET	http://testphp.vulnw...	login page	/login.php	[]	
13	POST	http://testphp.vulnw...	user info	/userinfo.php	[login=test%2F	
14	POST	https://passwordslea...		/v1/leaks:lookupSingle	[]	
15	POST	http://testphp.vulnw...	user info	/userinfo.php	[]	

Similarly, you can save whole test scenarios in the Filter Library. There is two ways to call the saved filter:

Method 1: In Filter Library, click on Set as LogFilter.

View Logs	Filter Library	Grep Values	Options	About	Help
Alias	Snippet				
Request_Type	Request.Method == "POST"	Set as LogFilter	Use as Color LogFilter		
Search_Username	Request.Body CONTAINS "..."	Set as LogFilter	Use as Color LogFilter		

It will directly run the query and the desired result will be displayed.

View Logs	Filter Library	Grep Values	Options	About	Help
Filter:	Request.Method == "POST"				
#	Method	Host	Title	Path	New Cookies
13	POST	http://testphp.vulnw...	user info	/userinfo.php	[login=test%2Ft...
14	POST	https://passwordslea...		/v1/leaks:lookupSingle	[]
15	POST	http://testphp.vulnw...	user info	/userinfo.php	[]

Method 2: Use “#” with Alias name directly in filter bar.

View Logs	Filter Library	Grep Values	Options	About	Help
Filter:	#Request_Type				
#	Method	Host	Title	Path	New Cookies
1	GET	http://testphp.vulnw...	Home of Acu...	/	[]
2	GET	http://testphp.vulnw...		/images/logo.gif	[]
3	GET	http://testphp.vulnw...	picture categ...	/categories.php	[]
4	GET	http://testphp.vulnw...	artists	/artists.php	[]
5	GET	http://testphp.vulnw...	you cart	/cart.php	[]
6	GET	http://testphp.vulnw...		/userinfo.php	[]
7	GET	http://testphp.vulnw...	login page	/login.php	[]
8	GET	http://testphp.vulnw...	login page	/login.php	[]
9	GET	http://testphp.vulnw...	guestbook	/guestbook.php	[]
10	GET	http://testphp.vulnw...		/images/remark.gif	[]
11	GET	http://testphp.vulnw...	ajax test	/AJAX/index.php	[]
12	GET	http://testphp.vulnw...	login page	/login.php	[]
13	POST	http://testphp.vulnw...	user info	/userinfo.php	[login=test%2Ft...
14	POST	https://passwordslea...		/v1/leaks:lookupSingle	[]
15	POST	http://testphp.vulnw...	user info	/userinfo.php	[]

And hit enter. The equivalent outcome will appear as follows:



View Logs Filter Library Grep Values Options About Help					
Filter: #Request_Type					
#	Method	Host	Title	Path	New Cookies
13	POST	http://testphp.vuln...	user info	/userinfo.php	[login=test%2Ft...
14	POST	https://passwordslea...		/v1/leaks:lookupSingle	[]
15	POST	http://testphp.vuln...	user info	/userinfo.php	[]

Regex-Based Filter

Burp Logger's regex filter is a powerful feature that helps web security professionals pinpoint specific data within the vast sea of information during security testing.

You need to specify the regular expression (regex) pattern. This pattern acts like a search query, telling Burp Logger++ what kind of data you want to capture. You can create regex expression pattern to find data as like Email Address, IP Address, Server-side error messages, Software version disclosed, Any API Key exposed etc.

Go to Logger++, click on Grep Values tab. Here, you can see more filters to limit the search criteria.

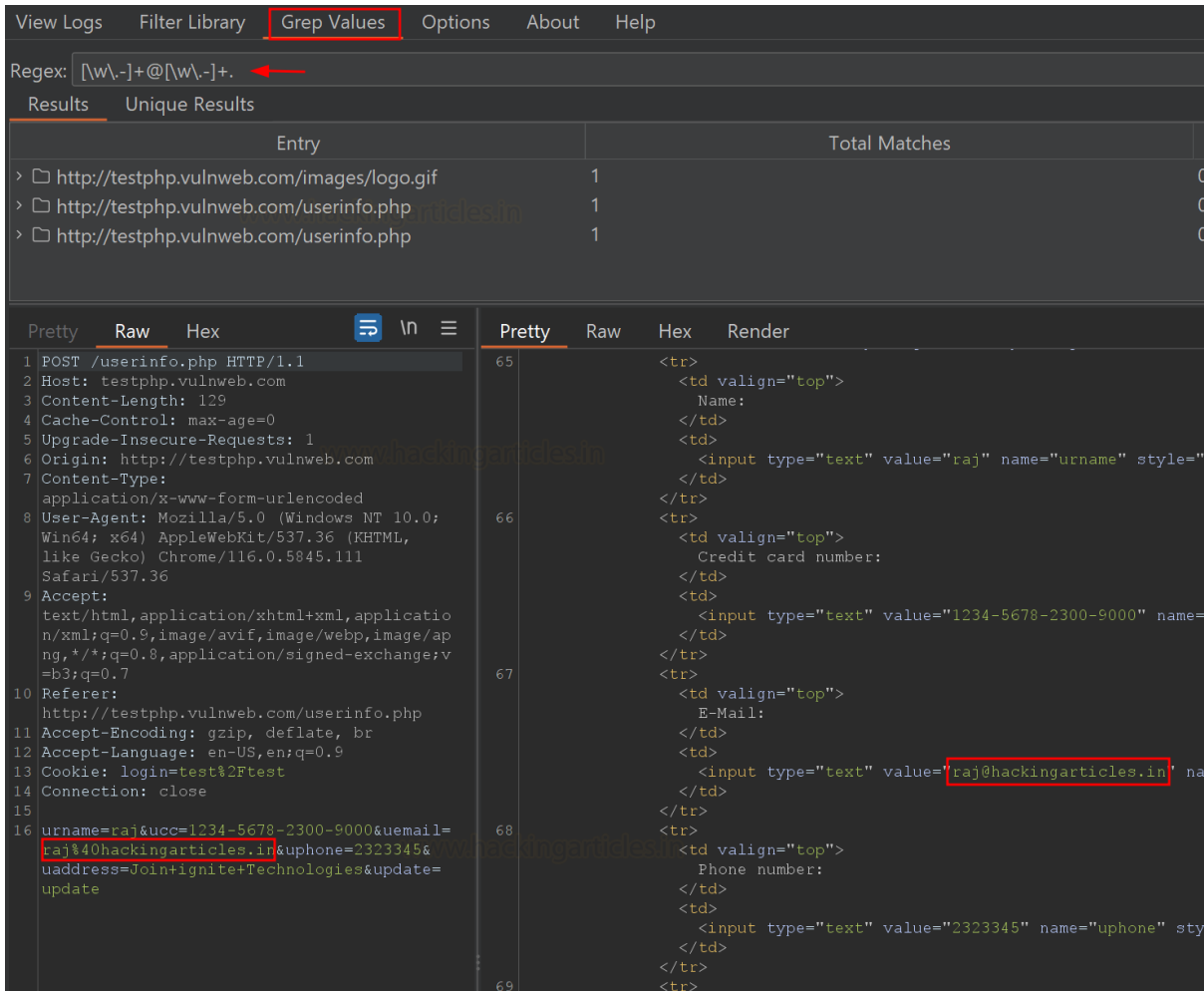
- Search Response = It will perform search only in responses.
- Search Request = It will perform search only in requests.
- In Scope Only = If you added the target URL in Scope only then it will only search within the scoped target.

For the time being, choose to search through every request and response. Let's take an example, if you want to find email addresses in web traffic, your regex pattern might look like

Regex:

```
[\\w\\.-]+@[\\w\\.-]+.
```

Directly paste this expression under Regex bar and press enter.



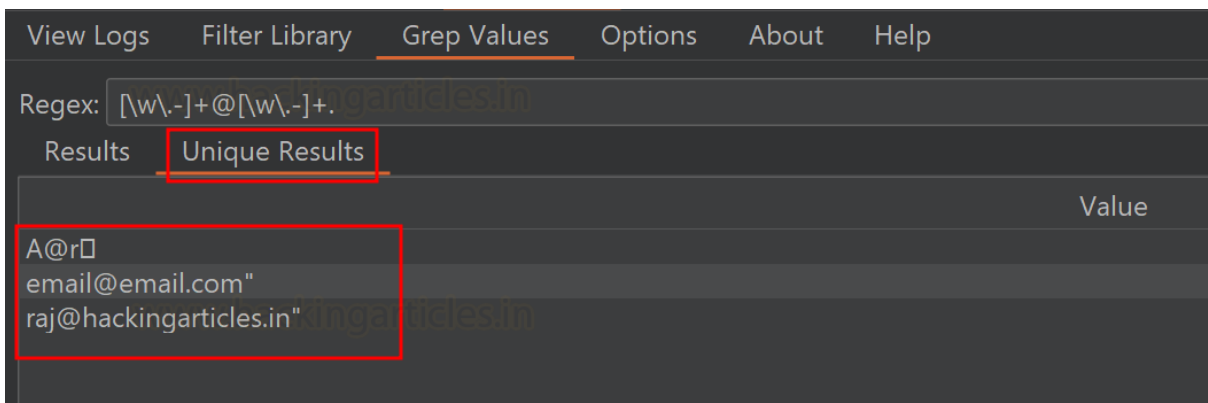
The screenshot shows the Burp Suite Grep Values window. The 'Grep Values' tab is selected. The 'Regex' field contains the pattern `[\\w\\.-]+@[\\w\\.-]+`. The 'Results' tab is active, displaying a table with two columns: 'Entry' and 'Total Matches'. The results are as follows:

Entry	Total Matches
http://testphp.vulnweb.com/images/logo.gif	1
http://testphp.vulnweb.com/userinfo.php	1
http://testphp.vulnweb.com/userinfo.php	1

Below the table, the 'Raw' tab is selected, showing the raw HTTP request and response data. The request is a POST to `/userinfo.php` with a `Content-Type` of `application/x-www-form-urlencoded`. The response is an HTML form with fields for 'Name', 'Credit card number', 'E-Mail', and 'Phone number'. The email field contains the value `raj@hackingarticles.in`, which is highlighted with a red box.

Consequently, the `/userinfo.php` request — which includes the email mentioned above is displayed.

You have two ways: Manually search through the complete request/response or click on Unique Result. The results that match the regex expression will be displayed only in Unique Results.



The screenshot shows the Burp Suite Grep Values window with the 'Unique Results' tab selected. The 'Regex' field contains the pattern `[\\w\\.-]+@[\\w\\.-]+`. The 'Unique Results' tab is active, displaying a table with two columns: 'Value' and 'Total Matches'. The results are as follows:

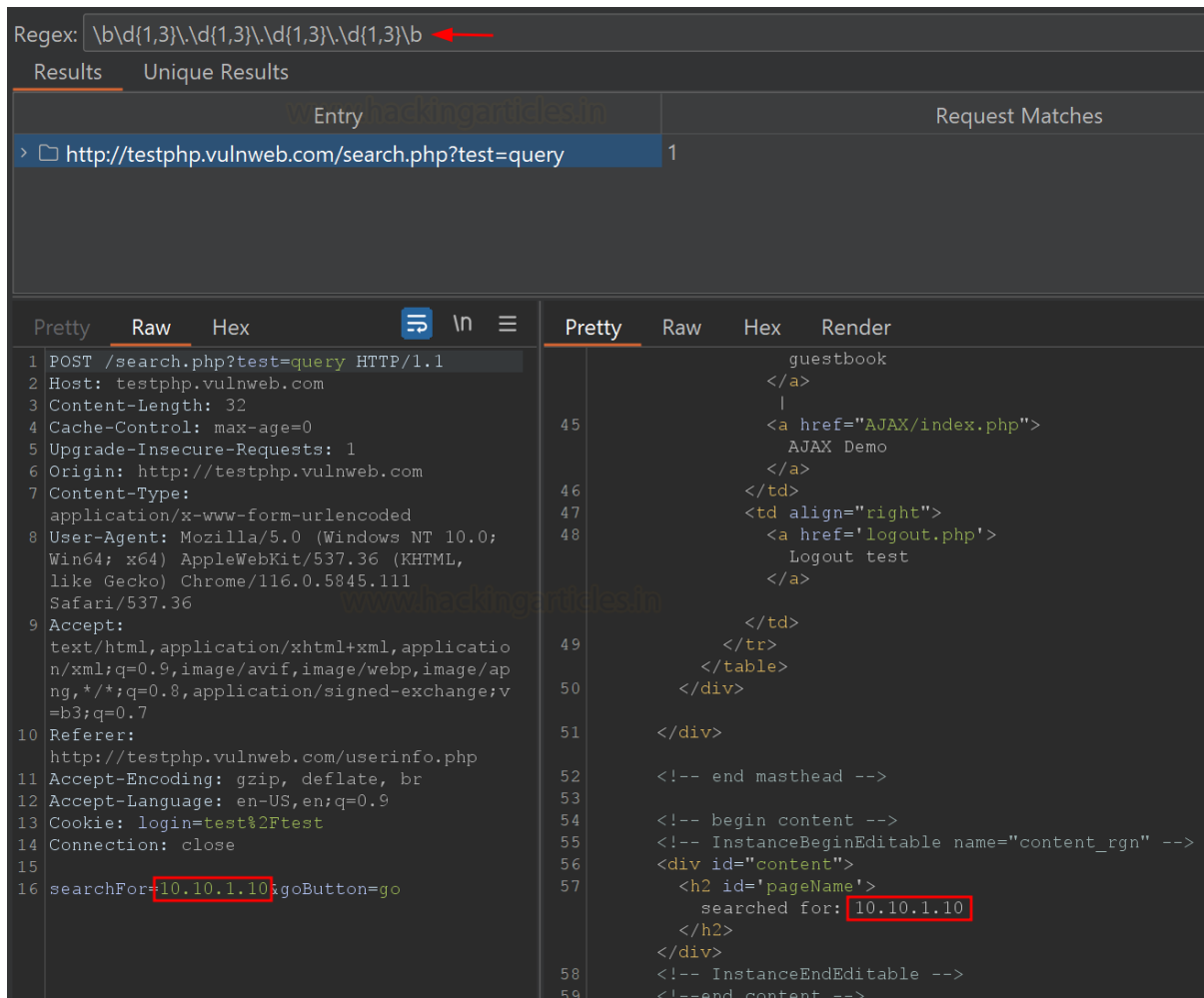
Value	Total Matches
A@r	1
email@email.com	1
raj@hackingarticles.in	1

The email `raj@hackingarticles.in` is highlighted with a red box.

Similarly, Let's check for IP Addresses,

Regex Exp:

`\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b`



The screenshot shows a web browser's developer tools with the 'Network' tab selected. A POST request to `http://testphp.vulnweb.com/search.php?test=query` is highlighted. The request body is visible in the 'Raw' tab, showing a search query for the IP address `10.10.1.10`. The response is an HTML page with a search result for `10.10.1.10`.

It is evident that a POST request is being sent through the IP address 10.10.1.10.

In the same way, you can check for other important information like if you want to find the web traffic contains any FTP, HTTP, WWW.

Regex:

`\b(ftp|www|http)[^\s]+`



Export Data Feature

Burp Logger's data export feature is a valuable tool for web security professionals. It allows you to save, analyze, and share the captured data efficiently, making it an essential tool for documenting findings, performing in-depth analysis, and collaborating with others in the field of web security.

Why Export Data Feature is Helpful?

- **Data Preservation:** Exporting data from the Logger++ allows you to save a record of your testing session. This is essential for documentation and analysis.
- **External Analysis:** By exporting data, you can use external tools or software to perform in-depth analysis, generate reports, or share findings with team members.
- **Archiving Evidence:** It helps in preserving evidence of potential vulnerabilities or security issues discovered during testing, which is crucial for audits and compliance.
- **Collaboration:** Exported data can be easily shared with colleagues or experts for collaborative analysis, making it an asset in team-based security testing.
- **Customization:** Depending on the export format chosen, you can tailor the exported data to meet specific reporting or analysis requirements.

Supported Formats:

- **Base64 JSON Format:** Base64-encoded data is often used to include binary data within a JSON structure.
- **JSON Format:** JSON is a lightweight data-interchange format used for structured data.
- **CSV Format:** CSV files are widely supported and can be opened in spreadsheet software like Microsoft Excel or Google Sheets.
- **HAR Format:** HTTP Archive (HAR) format is used for capturing and storing the performance-related data. The HAR format contains detailed information about HTTP requests and responses.

View Logs Filter Library Grep Values Options About Help

Filter:

#	Method	Host	Title	Path	New Cookies	Inferred Type
1	GET	http://testphp.vuln...	Home of Acu...	/		HTML
2	GET	http://testphp.vuln...		/images/logo.gif		IMAGE_GIF
3	GET	http://testphp.vuln...	picture categ...	/categories.php		HTML
4	GET	http://testphp.vuln...	artists	/artists.php		HTML
5	GET	http://testphp.vuln...	you cart	/cart.php		HTML
6	GET	http://testphp.vuln...		/userinfo.php		PLAIN_TEXT
7	GET	http://testphp.vuln...	login page	/login.php		HTML
8	GET	http://testphp.vuln...	login page	/login.php		HTML
9	GET	http://testphp.vuln...	guestbook	/guestbook.php		HTML
10	GET	http://testphp.vuln...		/images/remark.gif		IMAGE_GIF
11	GET	http://testphp.vuln...	ajax test	/AJAX/index.php		HTML
12	GET	http://testphp.vuln...	login page	/login.php		HTML
13	POST	http://testphp.vuln...	user info	/userinfo.php	[login=test%2Ft...	HTML
14	POST	https://passwordleakcheck-pa.googleapis.com/v1...		/lookupSingle		SCRIPT
15		https://passwordleakcheck-pa.googleapis.com/v1...		.php		HTML
16				.p		HTML

Use Request.Method Value As LogFilter
Set Request.Method Value as Color Filter
Remove from scope
Export as...
Crawl from here
Do an active scan
Do a passive scan
Send to Repeater
Send to Intruder
Send to Comparer
Remove Item

Export 1 entry as JSON
Export 1 entry as HAR
Export 1 entry as Base64 (JSON Formatted) >
Export 1 entry as CSV

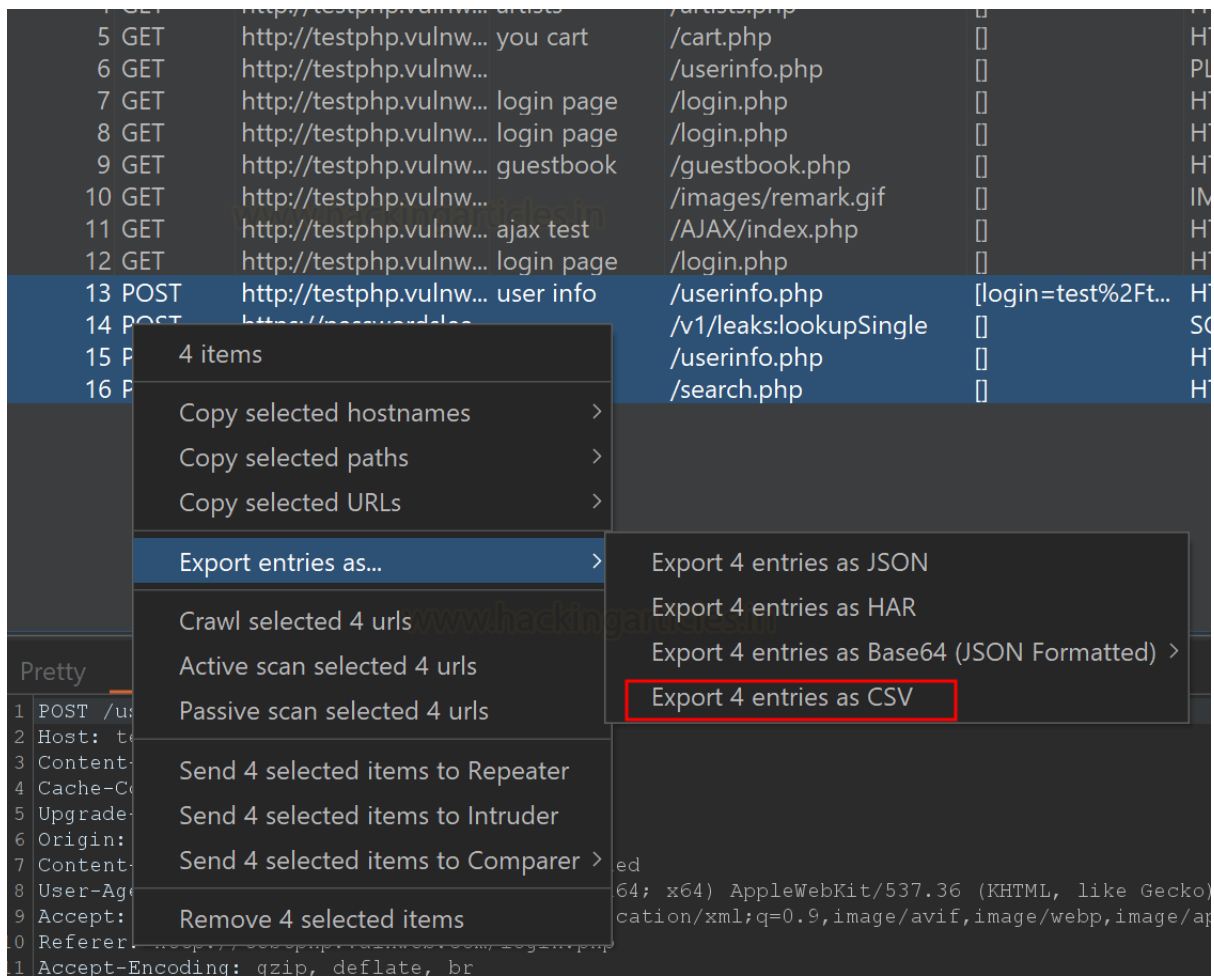
Pretty

1 POST /v...
2 Host: f...
3 Content...
4 X-Goog...
5 Content...
6 Sec-Fet...
7 Sec-Fet...
8 Sec-Fet...
9 User-Ag...
10 Accept...
11 Accept-Language: en-US,en;q=0.9
12 Connection: close

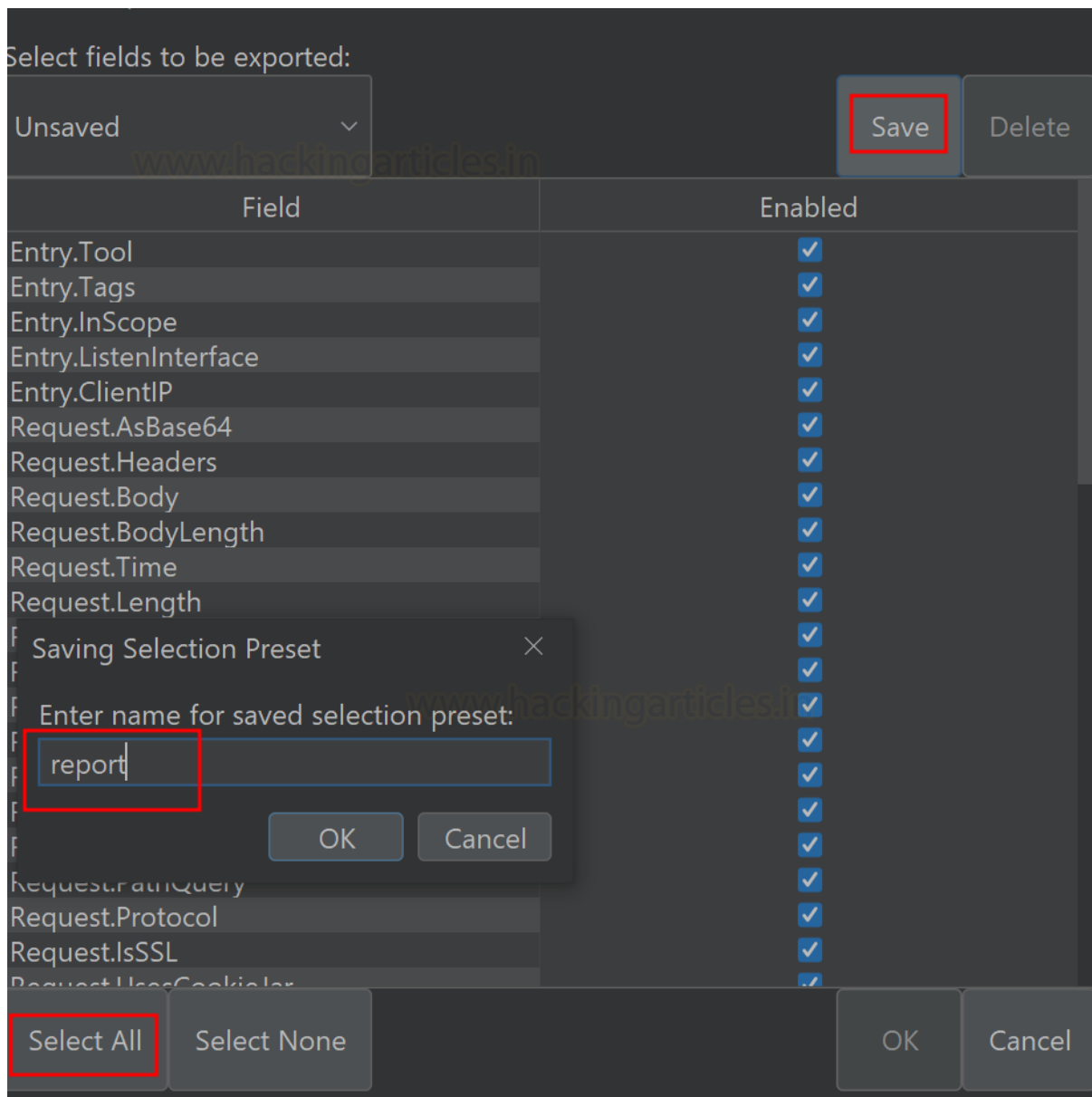
WebKit/537.36 (KHTML, like Gecko) Chrome/116

For Example, suppose you want to export all POST requests for further analysis.

Select the associate requests > **right click** > choose **Export entries as** > **Export as CSV**



Now **Select All** > **Choose Save** > Enter the name and click on Ok.



Save the result to your system offline. You can examine the CSV file; it contains all of the values that you chose to save.

You may select the only required values to store based on your needs.



File Home Insert Page Layout Formulas Data Review View Help														
Clipboard Font Alignment Number Styles Cells Comments Share														
A1														
	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1														
2	tes	POST	/userinfo. null	/userinfo. http	FALSE	No		testphp.v	http://tes	80	applicatio	testphp.v	php	http://
3	/pa	POST	/v1/leaks: null	/v1/leaks: https	TRUE	No		password:https://pa	443	applicatio	passwordsleakcheck-pa.goc			
4	tes	POST	/userinfo. null	/userinfo. http	FALSE	Yes		testphp.v	http://tes	80	applicatio	testphp.v	php	http://
5	tes	POST	/search.pl test=query	/search.pl http	FALSE	Yes		testphp.v	http://tes	80	applicatio	testphp.v	php	http://
6														
7														
8														
9														
10														
11														
12														

Conclusion

Whether you're doing thorough security checks or searching for hidden issues on your web applications, Burp Suite gives you a set of extensions to help you as a penetration tester. These tools boost the efficiency and effectiveness of your security assessments on web applications.

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- <https://www.hackingarticles.in/burpsuite-for-pentester-authorize/>
- <https://www.hackingarticles.in/burpsuite-for-pentester-logger/>
- <https://www.jython.org/download.html>
- <https://github.com/lwierzbicki/RegexFinder/blob/main/burp.regex.tsv>