#IBRAHIMXSS TOOL INTRODUCTION

Welcome to the **#IBRAHIMXSS** Tool, a groundbreaking innovation designed to revolutionize the way you approach web application security testing. This tool is meticulously crafted to detect and exploit **Cross-Site Scripting (XSS)** vulnerabilities with unparalleled accuracy and efficiency.

About the #IBRAHIMXSS Tool

The **#IBRAHIMXSS** Tool operates through a powerful API, providing seamless integration and ease of use for security professionals. After purchasing the tool from our official store **https://ibrahimxss.store/**, you will receive a complete package that includes everything you need to get started:

1. The #IBRAHIMXSS Tool: The core application designed for XSS vulnerability detection.
2. Requirements and Dependencies: All necessary libraries and dependencies to ensure smooth operation.
3. Payloads: A comprehensive set of payloads tailored for various XSS attack vectors.
4. Google Chrome: The preferred web browser for running the tool.
5. ChromeDriver: A necessary component to control Chrome via the **#IBRAHIMXSS** Tool.
6. After you Download file from **ibrahimxss.store** you will have these files on your machine: for Linux [*chromedriver google-chrome-stable_114.0.5735.90-1_amd64.deb, payloads.txt, request.txt, urls.txt and xss-checker], for Windows: [ChromeSetup.exe, chromedriver.exe, payloads.txt, request.txt, urls.txt and xss-checker.exe]*

Setting Up the Environment

To ensure optimal performance and functionality, you will need to set up your environment correctly. Follow these steps to get started:

1. Download the Tool: After your purchase, download the tool and all associated files from the provided link.
2. Install Google Chrome: Ensure that Google Chrome is installed on your system. The tool relies on Chrome for its scanning and exploitation capabilities.
3. Install ChromeDriver: Download and install the appropriate version of ChromeDriver to enable automated interactions with Google Chrome.
4. Install Dependencies: Use the provided package or installation script to install all necessary dependencies and libraries.
5. Load Payloads & URLs: Import the provided payloads into the tool. These payloads are specifically designed to maximize the detection of XSS vulnerabilities and also same for URLs.

By following these steps, you will be ready to use the #IBRAHIMXSS Tool to its full potential.

Installation for Windows

1. **Extract Files:**
   - After downloading the package, extract all the files into a dedicated folder on your system.
   - **Disable Windows Defender**: To avoid potential issues with running the **#IBRAHIMXSS Tool**, it is recommended to temporarily disable Windows Defender.
2. **Install Google Chrome:**
   - If you already have Google Chrome installed on your computer, please uninstall it.
   - Navigate to the folder where you extracted the files and run ChromeSetup.exe.
   - Ensure you install version 126.0.6478.127.
3. **Login to Google Account:**
   - Once Google Chrome is installed, open it and log in to your Gmail account.

   - Make sure to save your password in Chrome settings for seamless integration.

4. **Navigate to Folder:**
   - Open Command Prompt (CMD) as **Administrator**.
   - Navigate to the folder where you extracted the files. In this folder, you should see the following files: *chromedriver.exe, ChromeSetup.exe, request.txt, urls.txt, payloads.txt, and xss-checker.exe.*

5. **Fix Permission Error:**
   - If you encounter the error Error happened while checking for alert: Access to the path 'C:\Program Files\Google\Chrome\Application\screenshot\2024_7_3_12_5_3_488.png ' is denied, follow one of the two fixes below:

Fix 1: Command Prompt Method:

   - Open Command Prompt as **Administrator** and enter the following commands:

takeown /f "C:\Program Files\Google\Chrome\Application" /r /d y

icacls "C:\Program Files\Google\Chrome\Application" /grant ***YourSystemUsername****:F /*

- After running these commands, you will have successfully added **permission** to that folder. The **#IBRAHIMXSS** Tool needs to create folders for **screenshots**, and some folders require Administrator permissions.

Fix 2: Change Permissions via File Explorer:

- Navigate to C:\Program Files\Google\Chrome\Application.
- Right-click on the folder and select "Properties".
- Go to the "Security" tab and click "Edit".
- Select your user account and check the "**Full control**" checkbox under "**Allow**".
- Click "Apply" and then "OK".

Run #IBRAHIMXSS Tool:

- To start the **#IBRAHIMXSS** Tool on Windows, use the following command in Command Prompt (CMD) via Administrator:

```
xss-checker.exe --urls urls.txt --payloads payloads.txt --shuffle --threads 15
```

Troubleshooting Common Errors

1. Access to the path 'CrashpadMetrics-active.pma' is denied:

If this error occurs, open another Command Prompt as **Administrator** and enter the following command to kill all Google Chrome instances:

```
taskkill /F /IM chrome.exe /T
```

This is necessary because in order to run the **#IBRAHIMXSS Tool**, all Google Chrome instances must be cleared.

Installation for Linux

To ensure that the #IBRAHIMXSS Tool functions correctly, follow these detailed installation steps.

1. System Update

First, update your system with the following commands:

```
sudo apt update

sudo apt upgrade

sudo apt dist-upgrade
```

To ensure optimal performance and prevent potential issues, please make sure that you have at least 50 GB of free storage space available on your disk. This will help maintain system efficiency and provide sufficient room for updates, new applications, and other necessary files for running #IBRAHIMXSS Tool.

2. Google Chrome Installation

Ensure that Google Chrome is not already installed. If it is, remove it using the following commands:

```
sudo apt-get purge google-chrome-stable

sudo apt-get autoremove --purge google-chrome-stable

sudo rm -rf /etc/opt/chrome

sudo rm -rf /opt/google/chrome

sudo rm -rf /usr/lib64/google-chrome

sudo rm -rf /usr/share/man/man1/google-chrome.1

sudo rm -rf /usr/share/doc/google-chrome-stable

sudo rm -rf ~/.config/google-chrome

sudo rm -rf /usr/bin/google-chrome

sudo apt-get clean
```

Then, install Google Chrome again with these commands:

```
sudo dpkg -i google-chrome-stable_114.0.5735.90-1_amd64.deb

sudo apt update
```

If you are a **non-root** user, the setup for Google Chrome is **complete, but you need first to open Google Chrome and login to your Google account**. However, if you are a **root** user, you need to enable the **--no-sandbox** option:

3. Google Chrome Setup for Root Users

Edit the google-chrome.desktop file:

sudo leafpad /usr/share/applications/google-chrome.desktop

Scroll down to the bottom and modify the line to include **--no-sandbox**:

Exec=/usr/bin/google-chrome-stable ***--no-sandbox***

Edit the google-chrome script:

sudo leafpad /opt/google/chrome/google-chrome

Modify the last exec line to include --no-sandbox:

exec -a "$0" "$HERE/chrome" "$@" ***--no-sandbox***

Always remember, when you upgrade your system, Google Chrome will be updated automatically. To prevent this and ensure your **#IBRAHIMXSS** Tool continues to work, hold Google Chrome updates with this command:

sudo apt-mark hold google-chrome-stable

After successfully removing and installing the current version of Google Chrome (version 114.0.5735.90-1_amd64), verify the installation by typing:

google-chrome --version

If you are a root user, open Google Chrome from the terminal to ensure the

--no-sandbox option is enabled:

google-chrome

Now you must login to your Gmail account and save passwords into your Google Chrome Settings account. And with this you have successfully configured Google Chrome for #IBRAHIMXSS Tool for Linux. Please keep in mind that first you need to complete these steps for #IBRAHIMXSS Tool to make it work.

4. Setting Up Permissions

After setting up your system and Google Chrome, grant all permissions to the

xss-checker and **chromedriver** files with the following commands:

```
chmod +x xss-checker
```

```
chmod +x chromedriver
```

5. Running the #IBRAHIMXSS Tool

Run the xss-checker **without** the **sudo** option:

```
./xss-checker
```

You will be prompted to enter an **API key**. Depending on the plan you purchased from ibrahimxss.store, you will have those features enabled in your plan. The API key will be sent to the email address you used to purchase and create an account on the website.

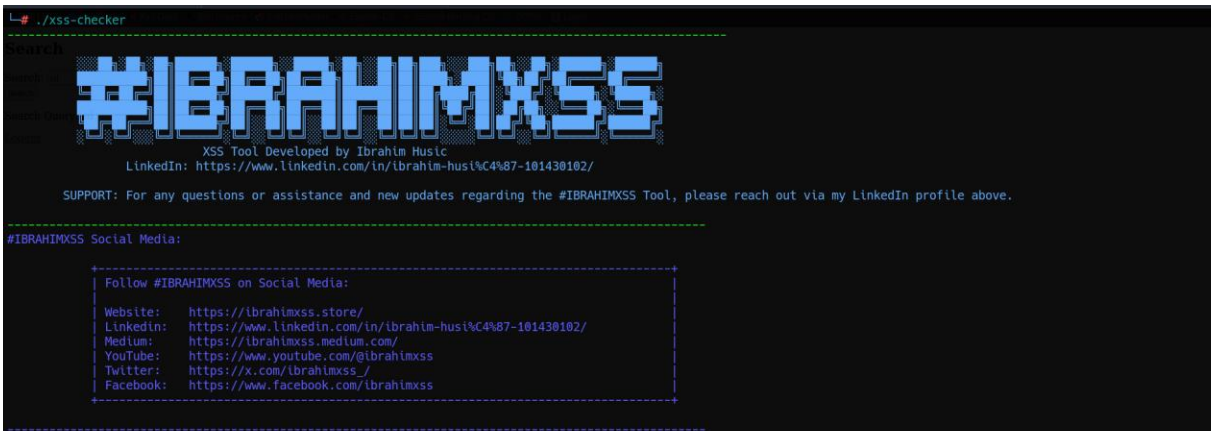
IMPORTANT: Ensure you have access to the email used for purchasing the plan, as the API key will be sent to that email.

IMPORTANT: Once you enter the API key, it will be saved into the system, and you will not be prompted again for your safety to enter it. If anyone finds your API key and tries to use it on another machine or run two instances at once, you will receive a **permanent ban**, and your API key will be **invalidated**. This means your purchased plan will automatically **expire** and be **banned**. Therefore, after you enter your API key, do not save it anywhere or try to enter it on another machine.

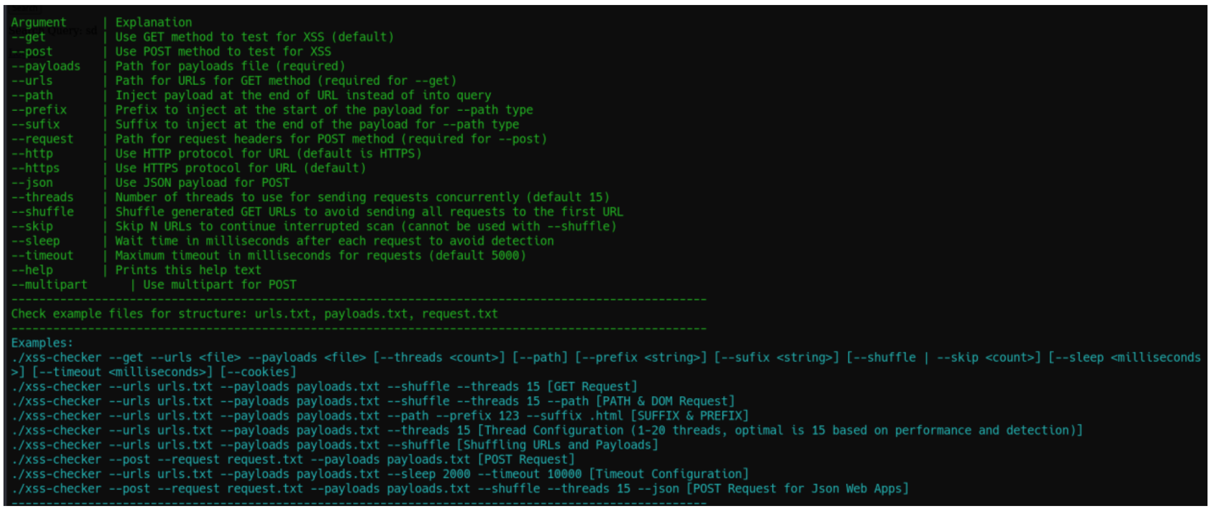
API access will be provided after purchasing a plan within a minimum of **6 hours and a maximum of 12 hours.** Sometimes the process is faster and can take just a few minutes. However, please note that users who buy a plan will have **access** to the **#IBRAHIMXSS tool,** but API access will be delivered to the email address used for purchasing and registering an account within the stated time frame.

After entering your API key, the **#IBRAHIMXSS** Tool will start, displaying a banner and examples on how to begin scanning for XSS vulnerabilities.

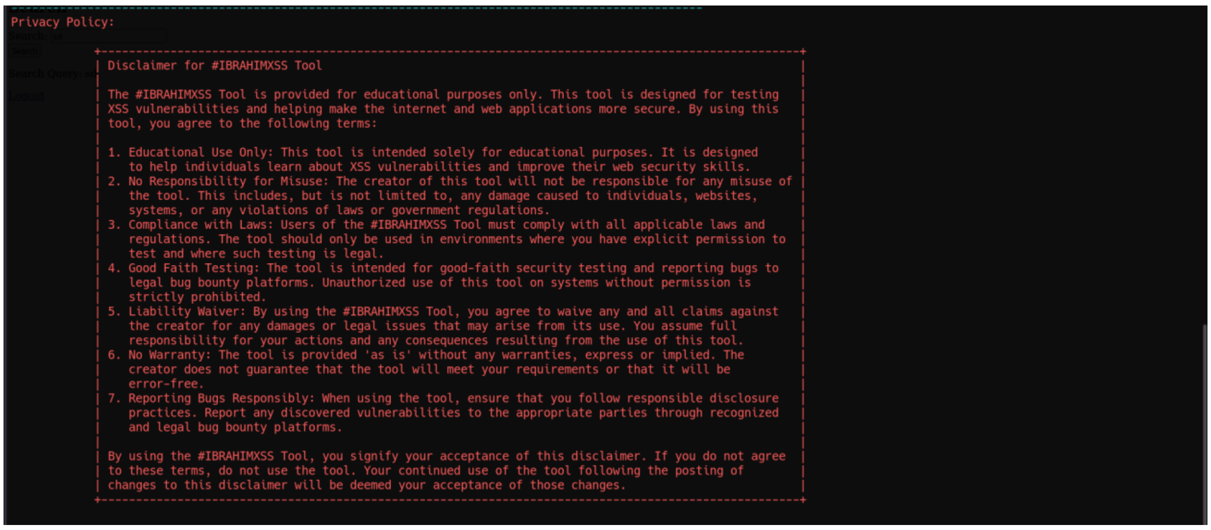
With these steps, your **#IBRAHIMXSS** Tool setup will be complete, allowing you to harness its powerful capabilities for web application security testing.



```
└# ./xss-checker
------------------------------------------------------------------------------------------------
#IBRAHIMXSS
                  XSS Tool Developed by Ibrahim Husic
           LinkedIn: https://www.linkedin.com/in/ibrahim-husi%C4%87-101430102/

        SUPPORT: For any questions or assistance and new updates regarding the #IBRAHIMXSS Tool, please reach out via my LinkedIn profile above.

---------------------------------------------------------------------------------------------
#IBRAHIMXSS Social Media:

            +-----------------------------------------------------------------------------+
            | Follow #IBRAHIMXSS on Social Media:                                         |
            |                                                                             |
            | Website:    https://ibrahimxss.store/                                       |
            | Linkedin:   https://www.linkedin.com/in/ibrahim-husi%C4%87-101430102/       |
            | Medium:     https://ibrahimxss.medium.com/                                  |
            | YouTube:    https://www.youtube.com/@ibrahimxss                             |
            | Twitter:    https://x.com/ibrahimxss_/                                      |
            | Facebook:   https://www.facebook.com/ibrahimxss                             |
            +-----------------------------------------------------------------------------+

---------------------------------------------------------------------------------------------
```



```
Argument      | Explanation
--get         | Use GET method to test for XSS (default)
--post        | Use POST method to test for XSS
--payloads    | Path for payloads file (required)
--urls        | Path for URLs for GET method (required for --get)
--path        | Inject payload at the end of URL instead of into query
--prefix      | Prefix to inject at the start of the payload for --path type
--sufix       | Suffix to inject at the end of the payload for --path type
--request     | Path for request headers for POST method (required for --post)
--http        | Use HTTP protocol for URL (default is HTTPS)
--https       | Use HTTPS protocol for URL (default)
--json        | Use JSON payload for POST
--threads     | Number of threads to use for sending requests concurrently (default 15)
--shuffle     | Shuffle generated GET URLs to avoid sending all requests to the first URL
--skip        | Skip N URLs to continue interrupted scan (cannot be used with --shuffle)
--sleep       | Wait time in milliseconds after each request to avoid detection
--timeout     | Maximum timeout in milliseconds for requests (default 5000)
--help        | Prints this help text
--multipart       | Use multipart for POST
---------------------------------------------------------------------------------------------
Check example files for structure: urls.txt, payloads.txt, request.txt
---------------------------------------------------------------------------------------------
Examples:
./xss-checker --get --urls <file> --payloads <file> [--threads <count>] [--path] [--prefix <string>] [--sufix <string>] [--shuffle | --skip <count>] [--sleep <milliseconds
>] [--timeout <milliseconds>] [--cookies]
./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 [GET Request]
./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 --path [PATH & DOM Request]
./xss-checker --urls urls.txt --payloads payloads.txt --path --prefix 123 --suffix .html [SUFFIX & PREFIX]
./xss-checker --urls urls.txt --payloads payloads.txt --threads 15 [Thread Configuration (1-20 threads, optimal is 15 based on performance and detection)]
./xss-checker --urls urls.txt --payloads payloads.txt --shuffle [Shuffling URLs and Payloads]
./xss-checker --post --request request.txt --payloads payloads.txt [POST Request]
./xss-checker --urls urls.txt --payloads payloads.txt --sleep 2000 --timeout 10000 [Timeout Configuration]
./xss-checker --post --request request.txt --payloads payloads.txt --shuffle --threads 15 --json [POST Request for Json Web Apps]
---------------------------------------------------------------------------------------------
```



```
---------------------------------------------------------------------------------------------
Privacy Policy:

           +-------------------------------------------------------------------------------------------+
           | Disclaimer for #IBRAHIMXSS Tool                                                           |
           |                                                                                           |
           | The #IBRAHIMXSS Tool is provided for educational purposes only. This tool is designed for testing |
           | XSS vulnerabilities and helping make the internet and web applications more secure. By using this |
           | tool, you agree to the following terms:                                                   |
           |                                                                                           |
           | 1. Educational Use Only: This tool is intended solely for educational purposes. It is designed |
           |    to help individuals learn about XSS vulnerabilities and improve their web security skills. |
           | 2. No Responsibility for Misuse: The creator of this tool will not be responsible for any misuse of |
           |    the tool. This includes, but is not limited to, any damage caused to individuals, websites, |
           |    systems, or any violations of laws or government regulations.                          |
           | 3. Compliance with Laws: Users of the #IBRAHIMXSS Tool must comply with all applicable laws and |
           |    regulations. The tool should only be used in environments where you have explicit permission to |
           |    test and where such testing is legal.                                                  |
           | 4. Good Faith Testing: The tool is intended for good-faith security testing and reporting bugs to |
           |    legal bug bounty platforms. Unauthorized use of this tool on systems without permission is |
           |    strictly prohibited.                                                                   |
           | 5. Liability Waiver: By using the #IBRAHIMXSS Tool, you agree to waive any and all claims against |
           |    the creator for any damages or legal issues that may arise from its use. You assume full |
           |    responsibility for your actions and any consequences resulting from the use of this tool. |
           | 6. No Warranty: The tool is provided 'as is' without any warranties, express or implied. The |
           |    creator does not guarantee that the tool will meet your requirements or that it will be |
           |    error-free.                                                                            |
           | 7. Reporting Bugs Responsibly: When using the tool, ensure that you follow responsible disclosure |
           |    practices. Report any discovered vulnerabilities to the appropriate parties through recognized |
           |    and legal bug bounty platforms.                                                        |
           |                                                                                           |
           | By using the #IBRAHIMXSS Tool, you signify your acceptance of this disclaimer. If you do not agree |
           | to these terms, do not use the tool. Your continued use of the tool following the posting of |
           | changes to this disclaimer will be deemed your acceptance of those changes.               |
           +-------------------------------------------------------------------------------------------+
```

How the #IBRAHIMXSS Tool Works

The **#IBRAHIMXSS** Tool is designed to thoroughly test web applications for Cross-Site Scripting (XSS) vulnerabilities by injecting payloads into various parts of a web request. Here's a detailed explanation of how the tool operates:

Payload Injection

The tool will inject payloads into all possible parameters and paths if no specific location is specified. For example:

https://ibrahimxss.store.com/?ref=homepage&campaign=spring_sale&user_id=12345&source=email

Payload Injection:

The tool will send payloads to the following parameters:

- ref=
- campaign=
- user_id=
- source=

All these parameters will be tested in GET requests to identify potential XSS vulnerabilities.

Path-Based XSS Testing

When using the --path option, the tool will inject payloads into all paths of the URL. For example:

Example URL:

https://ibrahimxss.store.com/homepage/spring_sale/12345/email

Payload Injection:

The tool will inject payloads sequentially into each path segment:

- First into homepage
- Then into spring_sale
- Followed by 12345
- Finally into email

While injecting payloads into each segment, the tool maintains the previous paths unchanged.

Specific Injection Location

If you specify a desired injection location, the tool will only inject payloads into that specified part of the URL. For example:

Example URL:

https://ibrahimxss.store.com/homepage/spring_sale/12345/{payload}

Payload Injection:

In this case, the tool will only inject payloads into the specified {payload} location, which is the email path in this example.

Testing POST Requests

To test for XSS vulnerabilities in POST requests, follow these steps:

1. Save the POST request into a .txt file using Burp Suite's "Copy to file" option or any other method which can copy whole POST request in right format.
2. In the saved file, replace the desired injection location with **{payload}** where you want to test for XSS.

JSON Web Applications

For Json-based web applications, the procedure is similar to the POST request method. Additionally, include the **--json** command to ensure proper handling of Json payloads.

By utilizing these methods, the **#IBRAHIMXSS** Tool can effectively and efficiently test for XSS vulnerabilities across various parts of a web application, ensuring comprehensive coverage and detection.

Command Line Arguments for #IBRAHIMXSS Tool

The **#IBRAHIMXSS** Tool supports a variety of command line arguments that allow you to customize the way it scans for XSS vulnerabilities. Here is a detailed explanation of each argument and how to use them:

Arguments and Explanations

- --get: Use the GET method to test for XSS vulnerabilities. This is the default method.
- --post: Use the POST method to test for XSS vulnerabilities specifically for Content-Type: application/x-www-form-urlencoded.
- --payloads <path>: Specifies the path to the file containing payloads. This argument is required.
- --urls <path>: Specifies the path to the file containing URLs to be tested using the GET method. This argument is required when using --get.
- --path: Injects the payload at the end of the URL instead of into the query parameters.
- --prefix <string>: Adds a prefix to the start of the payload for --path type injections.
- --suffix <string>: Adds a suffix to the end of the payload for --path type injections.
- --request <path>: Specifies the path to the file containing request headers for the POST method. This argument is required when using --post.
- --http: Uses the HTTP protocol for URLs. The default protocol is HTTPS.
- --https: Uses the HTTPS protocol for URLs. This is the default setting.
- --json: Uses JSON payloads for POST requests.
- --threads <count>: Specifies the number of threads to use for sending requests concurrently. The default is 15.
- --shuffle: Shuffles the generated GET URLs to avoid sending all requests to the first URL.
- --skip <count>: Skips the first N URLs to continue an interrupted scan. This option cannot be used with --shuffle.
- --sleep <milliseconds>: Sets a wait time in milliseconds after each request to avoid detection.
- --timeout <milliseconds>: Sets the maximum timeout in milliseconds for requests. The default is 5000.

- --help: Prints the help text with descriptions of all available arguments.
- --multipart: Uses multipart encoding for POST requests when there is more than one Content-Disposition: form-data; part in enctype="multipart/form-data".

Example Files

- urls.txt: Contains the list of URLs to be tested.
- payloads.txt: Contains the list of payloads to be injected.
- request.txt: Contains the request headers and body for POST requests.

Examples:

./xss-checker --get --urls <file> --payloads <file> [--threads <count>] [--path] [--prefix <string>] [--suffix <string>] [--shuffle | --skip <count>] [--sleep <milliseconds>] [--timeout <milliseconds>] [--cookies]

./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 [GET Request]

./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 --path [PATH & DOM Request]

./xss-checker --urls urls.txt --payloads payloads.txt --path --prefix 123 --suffix .html [SUFFIX & PREFIX]

./xss-checker --urls urls.txt --payloads payloads.txt --threads 15 [Thread Configuration (1-20 threads, optimal is 15 based on performance and detection)]

./xss-checker --urls urls.txt --payloads payloads.txt --shuffle [Shuffling URLs and Payloads]

./xss-checker --post --request request.txt --payloads payloads.txt [POST Request for Content-Type: application/x-www-form-urlencoded]

./xss-checker --post --request request.txt --payloads payloads.txt --multipart   [POST Request for multipart body handler] enctype="multipart/form-data"

./xss-checker --urls urls.txt --payloads payloads.txt --sleep 2000 --timeout 10000 [Timeout Configuration]

./xss-checker --post --request request.txt --payloads payloads.txt --shuffle --threads 15 --json [POST Request for Json Web Apps]

Basic GET Request:

```
./xss-checker --get --urls urls.txt --payloads payloads.txt --threads 15
```

GET Request with Path Injection:

```
./xss-checker --urls urls.txt --payloads payloads.txt --path --threads 15
```

GET Request with Path Injection and Prefix/Suffix:

```
./xss-checker --urls urls.txt --payloads payloads.txt --path --prefix 123 --suffix .html
```

Thread Configuration:

```
./xss-checker --urls urls.txt --payloads payloads.txt --threads 15
```

Shuffling URLs and Payloads:

```
./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15
```

POST Request for URL-Encoded Content-Type:

```
./xss-checker --post --request request.txt --payloads payloads.txt
```

POST Request with Multipart Body:

```
./xss-checker --post --request request.txt --payloads payloads.txt --multipart
```

Timeout Configuration:

```
./xss-checker --urls urls.txt --payloads payloads.txt --sleep 2000 --timeout 10000
```

POST Request for JSON Web Apps:

```
./xss-checker --post --request request.txt --payloads payloads.txt --json --shuffle --
threads 15
```

GET Request: Support for Cookies & Authenticated WebApps

The ability to test **authenticated** web applications using cookies in GET requests is available only in the **GOLDEN** and **BUSINESS** plans. These plans include detailed instructions on how to set up and use **GET** requests for authenticated web applications.

Important Note:

- **GOLDEN and BUSINESS Plans**: Support for testing authenticated web applications using cookies is included. This allows you to scan XSS vulnerabilities on both public and authenticated URLs.

- **BASIC and PRO Plans**: These plans support scanning XSS vulnerabilities only on **publicly** accessible URLs. They do **not include** support for authenticated web applications using cookies in GET requests.

POST Request: Support for Cookies & Authenticated WebApps

Support for testing authenticated web applications using cookies in **POST** requests is **available** in all **four** plans (BASIC, PRO, GOLDEN, and BUSINESS). When you save a POST request into a .txt file, it already includes all necessary headers, including cookies, enabling you to test for XSS vulnerabilities in authenticated web applications. But support for authenticated WebApps for **GET** requests is only available in GOLDEN and BUSINESS Plans.

Additional Information on Using the #IBRAHIMXSS Tool

Using the --threads Option

The **--threads** option allows you to specify the number of threads to use for sending requests concurrently. The optimal number of threads is between 1 and 15 per request. Using more threads can potentially slow down your network and PC, and may result in fewer confirmed alerts due to the high load. While you can use more than 15 threads, it is recommended to stick with 15 for balanced performance and efficiency, and **–**threads **must** be entered in every scanning process in **#IBRAHIMXSS** Tool.

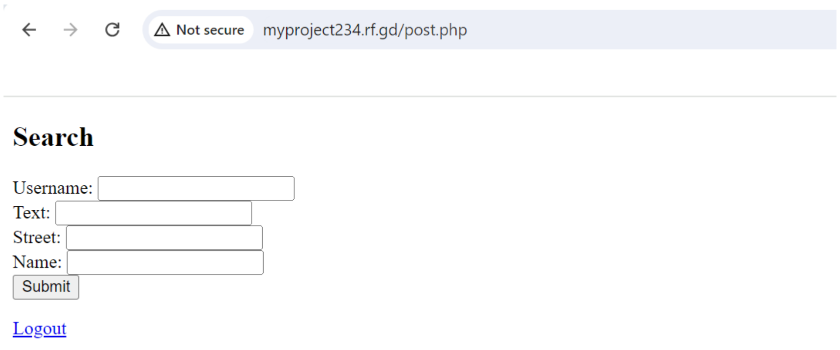
Handling Multipart POST Requests

For POST requests with multipart body handlers, which have more than one **Content-Disposition: form-data**;, you need to include the **--multipart argument**. This ensures that the tool can properly handle such POST requests.

POST Request for Multipart Body Handler (enctype="multipart/form-data"):

http://myproject234.rf.gd/login.php **admin:admin123**

After login Visit url: http://myproject234.rf.gd/post.php
Note: Don't forget to include an argument for –http



Search

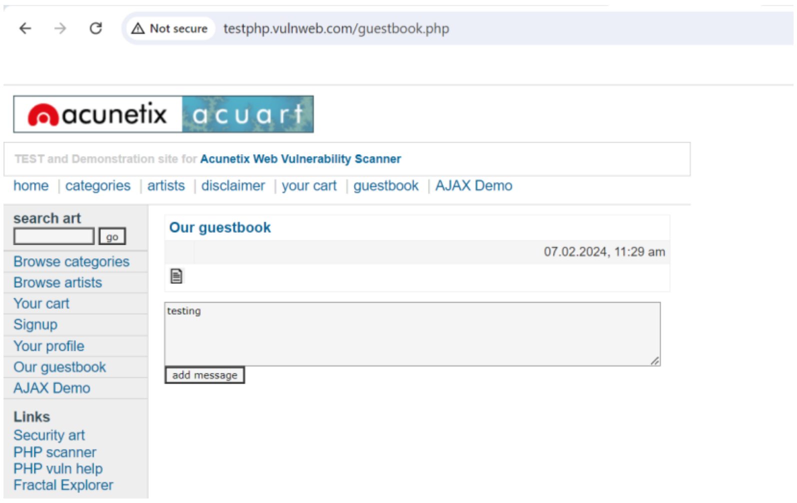
Username:
Text:
Street:
Name:
Submit

Logout

This URL is a POST example of a form that uses multipart encoding, often required when files are being uploaded through the form. To handle this type of request, include the **--multipart argument**. This will allow the tool to manage multiple Content-Disposition: form-data; fields effectively. When saving the POST request to a .txt file, make sure all relevant headers and body content are captured accurately and to put {payload} in desired place in this case in Text field.

POST Request for Content-Type: application/x-www-form-urlencoded:

http://testphp.vulnweb.com/guestbook.php



This URL represents a typical web form submission that uses application/x-www-form-urlencoded content type. It is an ideal example for testing standard POST requests.

When saving such a request to a .txt file, ensure all necessary headers, including cookies, are included to properly test authenticated web applications.

Basic Commands for Using the #IBRAHIMXSS Tool

1. GET Request

To perform a scan using a GET request, use the following command:

./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15

Example URL to place in urls.txt for testing:

http://testphp.vulnweb.com/listproducts.php?artist=1

This command will start the process of scanning based on GET requests.

2. Path Request

For scanning based on path requests, use the --path option:

./xss-checker --urls urls.txt --payloads payloads.txt --shuffle --threads 15 --path

This command will start injecting XSS payloads in every web path. You can also specify the injection point by placing {payload} in the desired path.

3. POST Request

To perform a scan using a POST request, use the following command:

./xss-checker --post --request request.txt --payloads payloads.txt

./xss-checker --post --request request.txt --payloads payloads.txt –multipart (for multripart body)

./xss-checker --post --request request.txt --payloads payloads.txt **–http** (for HTTP websites)

Note: Save a POST request into request.txt and put {payload} in the desired place within the request.

4. Json Web Apps

For JSON-based web applications, include the **--json** option:

./xss-checker --post --request request.txt --payloads payloads.txt --json

Additional Information

All additional information, including visual samples and detailed explanations with reports, can be found and read at the following URL:

https://ibrahimxss.medium.com/unlocking-the-future-of-web-security-with-the-ibrahimxss-tool-a33843cdc259

Farewell Summary for #IBRAHIMXSS Tool Users

To all the students who have purchased the **#IBRAHIMXSS Tool**, I wish you the best of luck in testing URLs and securing our world from XSS attacks. Your efforts in learning and applying these skills are crucial in making the internet a safer place.

For the bug bounty hunters, may this tool help you uncover vulnerabilities and earn well-deserved rewards. Your work not only benefits you financially but also strengthens the security of numerous applications and platforms.

To cyber security companies, I hope this tool enhances your ability to secure your clients' systems, making them more reliable and trustworthy. May it also help you provide quick and comprehensive reports on XSS vulnerabilities, adding value to your services.

To everyone using the **#IBRAHIMXSS** Tool, thank you for your dedication to improving web security. Your contributions, whether big or small, are vital in creating a safer online environment for all.

A Special Request

If you find this tool valuable and use it to discover XSS vulnerabilities on domains or bug bounty platforms, please tag me on LinkedIn or X. I would be thrilled to see your successes and, as a token of appreciation, I will offer a discount on your next month's purchase for those who share positive feedback about the tool on social media and use it for good.

Stay Connected

Feel free to reach me on my **LinkedIn profile** **or** **X profile** for any questions and for tagging me.

Thank you for choosing the **#IBRAHIMXSS Tool**.

Together, let's make the web a safer place!

#IBRAHIMXSS