



Version 1.0

Beagle Technical Document



Multi-Scan Session

Session	Status
http://testphp.vulnweb.com/ vulnerable web application #1	Idle
http://demo.testfire.net/ vulnerable web application #2	Idle
http://hackazon.webscantest.com/ vulnerable web application #3	Idle

Host	URL
testphp.vulnweb.com	/
testphp.vulnweb.com	/index.p
testphp.vulnweb.com	/userinf
testphp.vulnweb.com	/Mod_Rev
testphp.vulnweb.com	/cart.ph
testphp.vulnweb.com	/privacy
testphp.vulnweb.com	/categor
testphp.vulnweb.com	/hpp/
testphp.vulnweb.com	/guestbo
testphp.vulnweb.com	/disclai

i Unlike the other web application security scanners, Beagle can manage more than one scan session at the same time in one instance of the application.

Switching between sessions is just one click away.

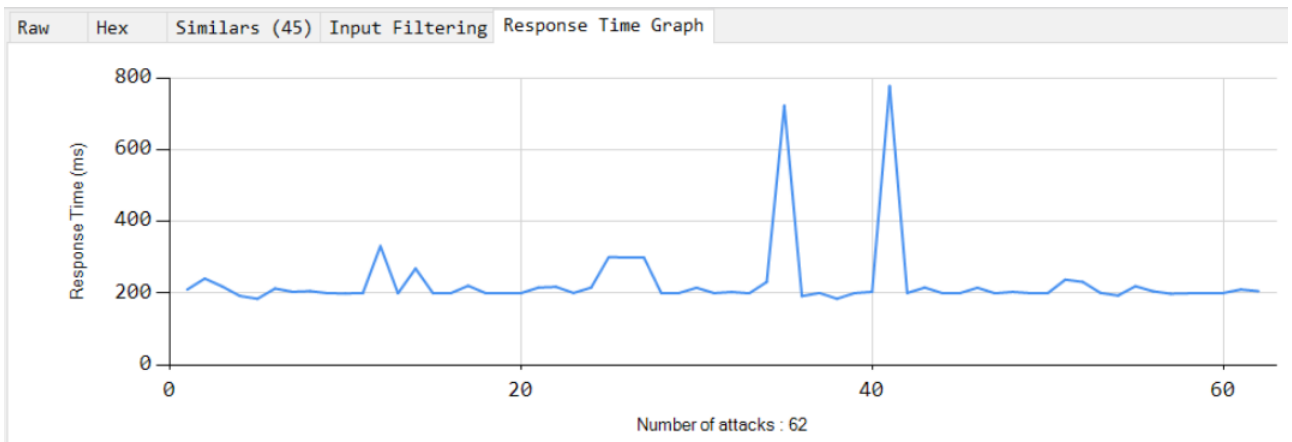


Intuitive Sitemap

Host	Path	Method	Avg Response Time	#Params	Attack Duration	
testphp.vulnweb.com	/pictures/wp-config.bak	GET	110 ms	0	00:00:03	
testphp.vulnweb.com	/cart.php	POST	178 ms	P2	00:00:35	
testphp.vulnweb.com	/comment.php	GET	91 ms	0	00:00:02	
testphp.vulnweb.com	/medias/	GET	122 ms	0	00:00:12	
testphp.vulnweb.com	/medias/img/	GET	128 ms	0	00:00:14	
testphp.vulnweb.com	/sendcommand.php	POST	317 ms	P2	00:00:33	
testphp.vulnweb.com	/sendcommand.php	GET	117 ms	0	00:00:03	
testphp.vulnweb.com	/AJAX/artists.php	GET	96 ms	0	00:00:02	
testphp.vulnweb.com	/AJAX/infoartist.php?id=1	GET	142 ms	G1	00:00:02	
testphp.vulnweb.com	/AJAX/categories.php	GET	124 ms	0	00:00:03	
testphp.vulnweb.com	/AJAX/infocateg.php?id=1	GET	146 ms	G1	00:00:02	
testphp.vulnweb.com	/AJAX/titles.php	GET	117 ms	0	00:00:03	
testphp.vulnweb.com	/AJAX/infotitle.php	POST	252 ms	P1	00:00:18	

i Each of the crawled pages gets added into the sitemap and it is easy to track down the attack progress. Sitemap gives you the most important details about pages throughout the crawling & attacking. You can see;

- HTTP method (GET | POST | PUT | DELETE etc.)
- Total number of parameters to attack
- **Average response time** which changes during attacking. This will give you a clue about there might be a problem with loading page under heavy load.
- Attack duration
- If there is an **issue found** for the page.





Injection Configuration

i Sometimes you will need to restrict the scan. For example you only need to;

- 1- Detect specific type(s) of vulnerability (e.g, SQL injection)
- 2- Attack only an injection point (e.g, POST parameters)
- 3- Filter parameter names

Beagle can restrict scans based on 3 combinations above.

Injection Controller
✕

Default state is **disabled**
[Invert rules](#)
[Reset to default](#)

Engine	Injection Type	Name	Enabled	
SQL Injection	QueryString	<any>	<input checked="" type="checkbox"/> Yes	Remove
HTML Injection	QueryString	<any>	<input checked="" type="checkbox"/> Yes	Remove
HTML Injection	Post	<any>	<input checked="" type="checkbox"/> Yes	Remove
Local File Inclusion	QueryString	path	<input checked="" type="checkbox"/> Yes	Remove
Open Redirection	QueryString	url	<input checked="" type="checkbox"/> Yes	Remove
Command Injection	Post	cmd	<input checked="" type="checkbox"/> Yes	Remove

Apply

Conflicts / Hints

Result

Engine	QueryString	Post	Header	Cookie	Url	
Backup Files	[]	[]	[]	[]	[]	
Open Redirection	[O]	[]	[]	[]	[]	
Command Injection	[]	[O]	[]	[]	[]	
Header Injection	[]	[]	[]	[]	[]	
Local File Inclusion	[O]	[]	[]	[]	[]	
Code Evaluation	[]	[]	[]	[]	[]	
SQL Injection	[X]	[]	[]	[]	[]	
HTML Injection	[X]	[X]	[]	[]	[]	



Context-Aware Analyze

i When a user controllable input is reflected without being encoded in HTML, it might lead to a security issue in a certain level.

Beagle detects these types of vulnerabilities by analyzing context not brute-force attacks. In first step, it identifies the context by analyzing the reflection and then generates a payload based on this context without breaking the javascript context.

Raw	Hex	Similar (0)	Input Filtering	Response Time Graph
<	>	[space]	<scRipt>	' + " ` * / - () : & = @ , ;
=====				
p	X	- - X	- - - - -	- - - - -
HTML injection				
=====				
p	/* script context */			
	<script>			
	y="fooINJECTION_HEREbar";			
	</script>			
	/* script context */			
	<script>			
	y="foo"+xss(0xDEADC0DE)+"bar";			
	</script>			



Boolean SQL Injection

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

Injection Point: artist

Type: QueryString

Payload: 1 AND 1=1

Exploit

Exploit SQLi ✕

Database : MySQL

SELECT @@VERSION Execute Query

5.1.73-0ubuntu0.10.04.1

Request

GET /artists.php?artist=1%20AND%201%3d1 HTTP/1.1

Command Injection

Url: [redacted]

Injection Point: [redacted]

Type: Post

Payload: -1&&

Exploit

Request

POST [redacted]
 Cache-Control: [redacted]
 Accept-Charset: [redacted]
 Accept-Language: [redacted]
 Accept-Encoding: [redacted]
 Accept: text/x[redacted]
 User-Agent: Mo[redacted]
 Referer: http: [redacted]

Remote Code Exploiter - □ ✕

Operating System : Windows

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0		0	24 K
System	4		0	300 K
smss.exe	272		0	1,104 K
csrss.exe	348		0	5,064 K
wininit.exe	400		0	4,348 K
csrss.exe	408		1	3,844 K
winlogon.exe	436		1	4,156 K
services.exe	496		0	8,984 K
lsass.exe	504		0	11,708 K
lsm.exe	512		0	5,420 K
svchost.exe	608		0	8,908 K
nvsvc.exe	672		0	6,632 K
nvwm164.exe	696		0	3,956 K
nvSCPAPISvr.exe	720		0	5,624 K
svchost.exe	764		0	7,136 K
LogonUI.exe	844		1	14,184 K
svchost.exe	852		0	12,048 K
svchost.exe	904		0	37,008 K
svchost.exe	960		0	13,048 K

Command



i This feature is only available in Professional Edition

Beagle scripting engine allows you to generate payloads on the fly while performing manual assessments. Its integrated tools, internal data sources and 30+ commands make payload generation easier to carry out in HTTP requests.

You can work with as much different data sources as possible and combine them.

The screenshot displays the Beagle Script Development Environment. The main editor shows a script with the following code:

```
/*
Beagle Script Editor
*/
state Init
  our $payloads = array(string);
  call(GeneratePayload);
  exit();
end state

state GeneratePayload
  $user = load("users");
  while(next(user)) {
    $payload = concat($name, ":", $pass);
    push($payloads, $payload);
  }
end state
```

The `load("users")` and `concat($name, ":", $pass)` lines are highlighted in yellow. A red arrow points from the `load("users")` line to the `Data Sources` window, which shows a table of users:

name	pass
john	123
jane	456
foo	bar

Another red arrow points from the `concat($name, ":", $pass)` line to the `Array Viewer` window, which displays the resulting payload array:

```
Variable : payloads
Value:
- ROOT
  0 : john:123
  1 : jane:456
  2 : foo:bar
```

The `Locals` window shows the following variables:

Name	Value
PRODUCT_NAME	Beagle
PRODUCT_VERSION	1.0
payloads	Item Count : 3

At the bottom, a status bar indicates: `Compiled successfully in (5,0859 ms)`