Overview

Title	Category	Points	Flag
Welcome	Misc	10	HKCTF{WELCOME}
Welcome again	Misc	25	HKCTF{Welcome!_25607a}
Connect	Misc	50	HKCTF{Zmlyc3RfbmNfMTIzMDE4Cg}
Bit Key	Misc	50	HKCTF{a_s3cr3t_k3y}
Simple Forensics	Forensics	50	HKCTF{grep_and_you_will_find_me}
Simple Forensics 2	Forensics	75	HKCTF{sTrIngS_sAVeS_Time_4c987dcwxq}
Good Image	Forensics	50	HKCTF{Open2019_b07209aJ}
Client is not secure	Web	50	HKCTF{client_code_can_be_seen_735a829c2}
Admin Access	Web	125	HKCTF{Cookie_should_be_examined_45a812ea}
Only Web? No	Web	300	HKCTF{2B1AS1_82n8d^hs}

Protect	Crypto	100	HKCTF{Th1s_Is_@_L0ng_P@ssw0rd}
Simple Crypto	Crypto	75	HKCTF{MessageEncryptedUsingVigenereCipher}
Simple Crypto 2	Crypto	100	HKCTF{message_encrypted_using_rot47}
RSA 1	Crypto	300	HKCTF{rs@_crypt0_us3_10801193}
RSA 2	Crypto	450	HKCTF{w@tch_y0ur_rs@_c@r3fu11y_87027243203}
Buffer Overflow	Pwn	100	HKCTF{S1mpl3_Buff3r_0v3rfl0w_016dc68c}
Simple APK	Reverse Engineering	100	HKCTF{pwyorowkw3}
Secret in Android App 1	Reverse Engineering	125	HKCTF{Android_RE_is_simple}

Misc 10: Welcome

Description

Welcome to HKCTF Open. Please get the flag from the file.

File: Welcome.png

Solution:

Download and open the Welcome.png.

Oh look, there's the flag!

Flag: HKCTF{WELCOME}

Misc 25: Welcome again

Description

The flag is 484b4354467b57656c636f6d65215f3235363037617d.

Solution:

The message seems like it is in HEX format.

Convert the message "484b4354467b57656c636f6d65215f3235363037617d" to ascii format

Ok, that's looks like a flag.

Flag: HKCTF{Welcome!_25607a}

Misc 50: Connect

Description

Can you get the flag from 47.56.165.70 at port 22222?

Solution:

Ok, let's try to connect 47.56.168.70:22222 by netcat

Input Command: netcat 47.56.165.70 22222

```
root@kali:~# netcat 47.56.165.70 22222
HKCTF{Zmlyc3RfbmNfMTIzMDE4Cg}root@kali:~#
```

It's been listening all the time?, we get "HKCTF{Zmlyc3RfbmNfMTIzMDE4Cg}"

Flag: HKCTF{Zmlyc3RfbmNfMTIzMDE4Cg}

Forensics 50: Simple Forensics

Description

Can you find the flag in this file?

File: file

Solution:

Download the file.

Grep string that sticks with "HKCTF".

Command: grep HKCTF file

```
run$ grep HKCTF file
HKCTF{grep_and_you_will_find_me}
run$
```

Flag: HKCTF{grep_and_you_will_find_me}

Forensics 75: Simple Forensics 2

Description

Can you find the flag in this file?

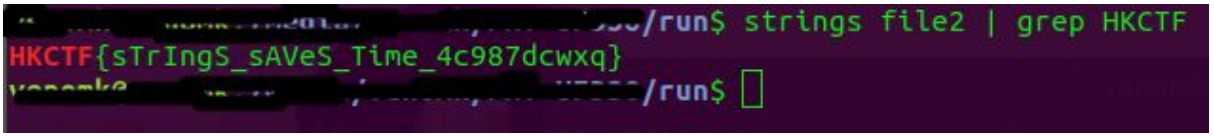
File: file

Solution:

Download the file.

Use Strings Command and grep with “HKCTF” again.

Command: Strings file | HKCTF file



Flag: HKCTF{sTrIngS_sAVeS_Time_4c987dcwxq}

Forensics 50: Good Image

Description

Can you help us find the flag in this image?

File: CTFOpen2019.png

Solution:

Download the image and see.

That's a Great image, let's try to throw it in Text Editor.

```
<rdf:Description rdf:about=''
 xmlns:dc='http://purl.org/dc/elements/1.1/'>
 <dc:creator>
  <rdf:Seq>
   <rdf:li>HKCTF{Open2019_b07209aJ}</rdf:li>
  </rdf:Seq>
 </dc:creator>
</rdf:Description>

<rdf:Description rdf:about=''
 xmlns:exif='http://ns.adobe.com/exif/1.0/'>
 <exif:PixelXDimension>476</exif:PixelXDimension>
 <exif:PixelYDimension>439</exif:PixelYDimension>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
```

Great the flag is in the Meta data.

Flag: HKCTF{Open2019_b07209aJ}

Web 50: Client is not secure

Description

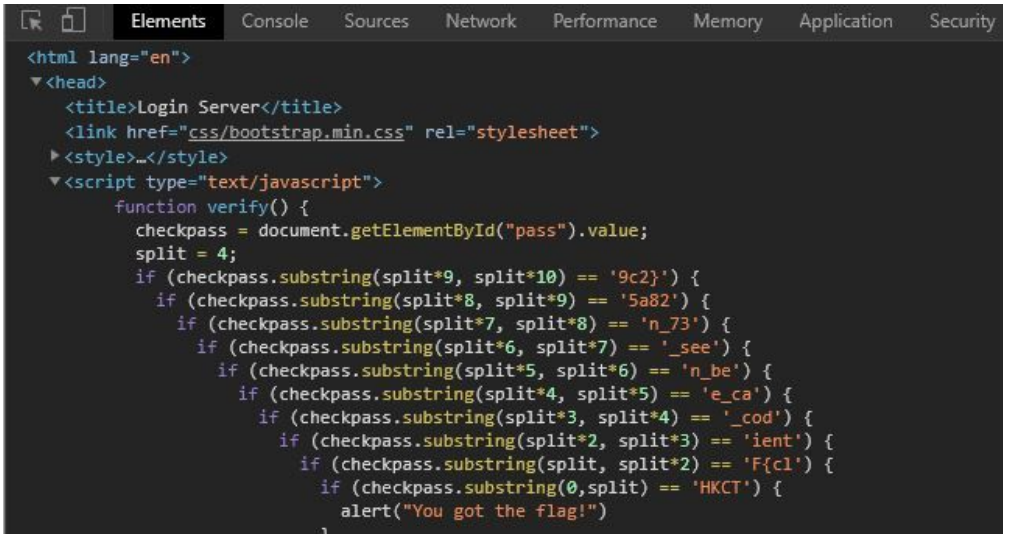
Can you help me to find my password? http://47.56.165.70:25241/login.php

Solution:

Enter the link, and I need a password?

Ok, let's go through the sources, JavaScript file.

This line seems to be comparing the string for the password.



This format seems to be a flag,let's just append the string according to the splits counts. The result will be "HKCTF{client_code_can_be_seen_735a829c2}".

I will not attempt to login but submit the flag.

Flag: HKCTF{client_code_can_be_seen_735a829c2}

Web 125: Admin Access

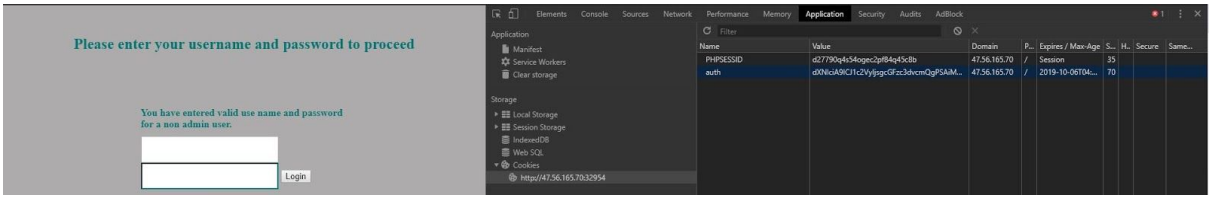
Description

I got a non-admin account (username is user, password is 1234) for this website: http://47.56.165.70:32954/login.php. But I need an admin user. How should I do?

Solution:

Enter the link, so I got the (username is user, password is 1234), login with information given.

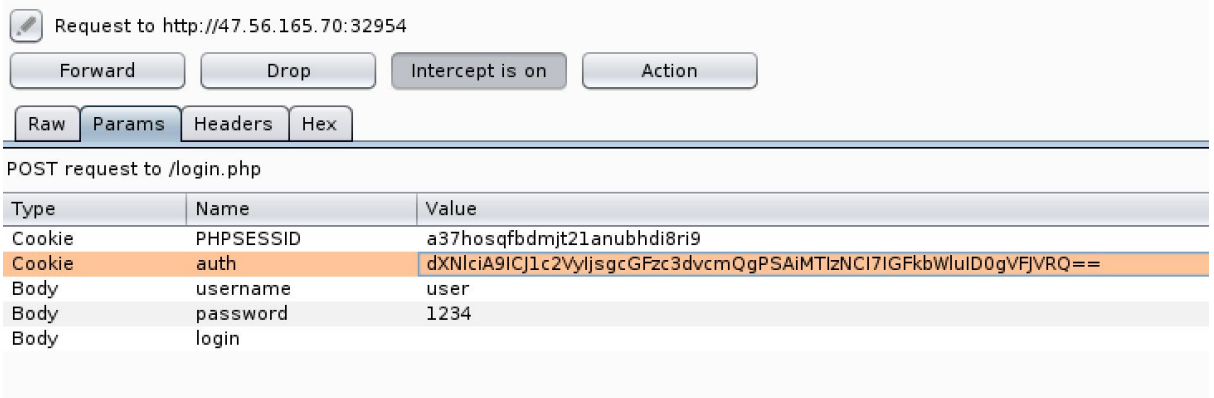
I am required an admin account, let's check the cookie.

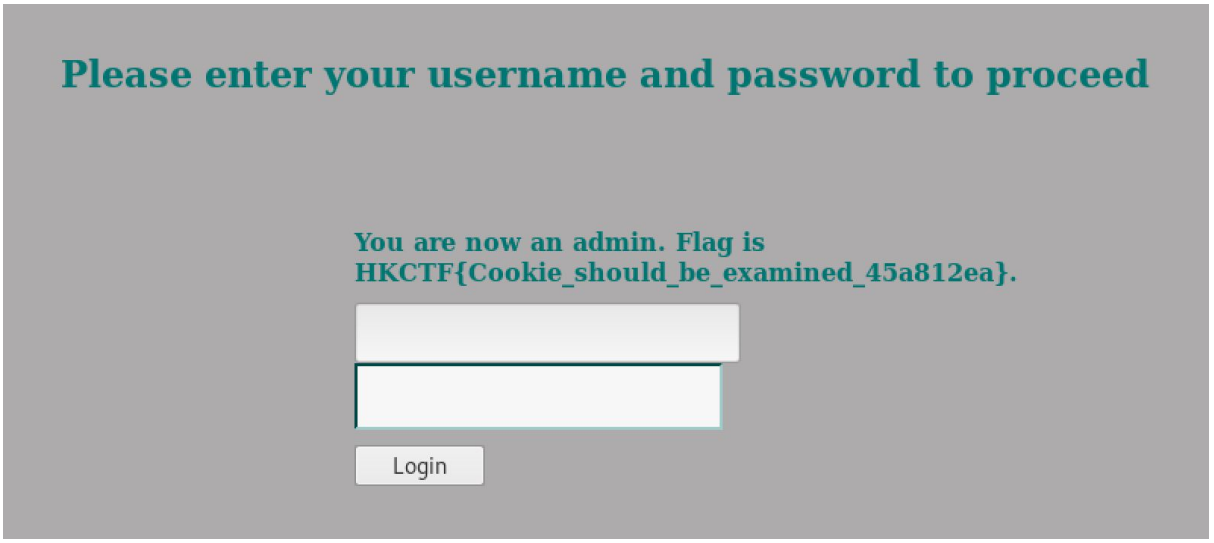


“auth” seems to be in base64 format.

```
root@kali:~# echo -n "dXNlciA9ICJ1c2VyIjsgcGFzc3dvcmQgPSAiMTIzNCI7IGFkbWluID0gRkFMU0U%3D" | base64 -d
user = "user"; password = "1234"; admin = FALSEbase64: invalid input
```

Something I just input (username = user, password = 1234), “admin = FALSE” seems fun. Lets try to change it to TRUE and change it back base64 format and send the cookie.





Flag: HKCTF{Cookie_should_be_examined_45a812ea}

Web 300: Only Web? No

Description:

Do you want to read some Book?

Please access it on http://13.251.58.69:8004

Solution:

Enter the website, "`it is not the spoon that bends`", I love Matrix!

Nothing special here. There is /common directory, so there is uncommon directory? Let's try /flag directory


```
HKCTF{2B1AS1_82n8d^hs}
```

Great here is the flag.

Flag: HKCTF{2B1AS1_82n8d^hs}

Crypto 75: Simple Crypto

Description:

I got a message from a friend, ARKLN{EekwcxiXxgprwbwlMsariMmzorcklKaxzej} with the key of thisisasecretkey. Can you use the table to help me to decrypt it?

File: table.txt

Solution:

So I got Cipher " ARKLN{EekwcxiXxgprwbwlMsariMmzorcklKaxzej}" and secret key "thisisasecretkey" and a shifting table.

A **Vigenère cipher**, let's just keep shift it with the table. The result is "HKCTF{MessageEncryptedUsingVigenereCipher}"

Flag: HKCTF{MessageEncryptedUsingVigenereCipher}

Crypto 100: Simple Crypto 2

Description:

I got another message from a friend, wzr%uL>6DD28606?4CJAE650FD:?80C@EcfN. Can you help me to decrypt it this time?

Solution:

The Cipher is "wzr%uL>6DD28606?4CJAE650FD:?80C@EcfN"

Involving numbers and special characters, so this is ROT47 cipher.

Decipher it with ROT47, the result is "HKCTF{message_encrypted_using_rot47}"

Flag: HKCTF{message_encrypted_using_rot47}

Crypto 100: Protect

Description:

The message is encrypted with a password with the provided script! Can you get us the password?

File: protect.py

message

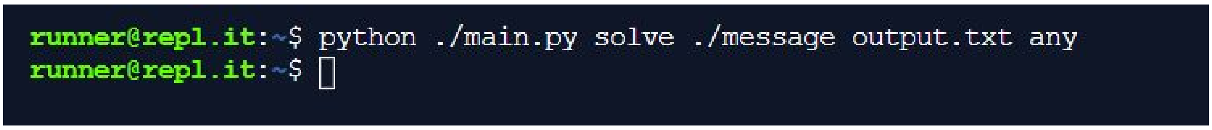
Solution:

Edit the python file: add solve function and edit main function

```
def solve(input_data):
    result = ""
    for key in range(0,255):
        result += process(input_data,key)
        result += "  solve "
    return result
```

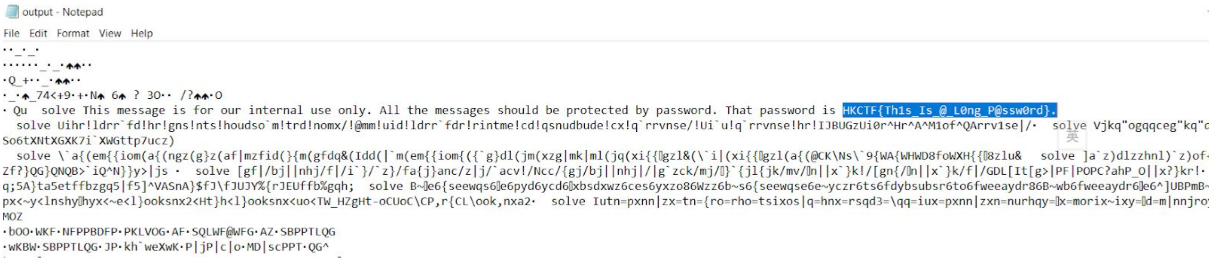
```
def main():
    if len(sys.argv) < 5:
        usage()

    input_data = open(sys.argv[2], 'r').read()
    result_data = ""

    if sys.argv[1] == "encrypt":
        result_data = encrypt(input_data, sys.argv[4])
    elif sys.argv[1] == "decrypt":
        result_data = decrypt(input_data, sys.argv[4])
    elif sys.argv[1] == "solve":
        result_data = solve(input_data)
    else:
        usage()

    out_file = open(sys.argv[3], 'w')
    out_file.write(result_data)
    out_file.close()

main()
```

Run and generate output



```
runner@repl.it:~$ python ./main.py solve ./message output.txt any
runner@repl.it:~$ 
```

Open the output file and get the flag



```
••_•_•
••••••_•_•♠♠••
•Q_+••_•♠♠••
•_•♠_74<+9•+•N♠ 6♠ ? 3O•• /?♠♠•O
• Qu  solve This message is for our internal use only. All the messages should be protected by password. That password is HKCTF{Th1s_Is_@_L0ng_P@ssw0rd}.
  solve Uihr!ldrr`fd!hr!gns!nts!houdso`m!trd!nomx/!@mm!uid!ldrr`fdr!rintme!cd!qsnudbude!cx!q`rrvnse/!Ui`u!q`rrvnse!hr!IJBUGzUi0r^Hr^A^M1of^QArrv1se|/•  solve Vjkq"ogqqceg"kq"
So6tXNtXGXK7i`XWGttp7ucz)
  solve \`a{(em{{iom(a{(ngz(g}z(af|mzfid(}{m(gfdq&(Idd(|`m(em{{iom{({`g}dl(jm(xzg|mk|ml(jq(xi{{▯gzl&(\`i|(xi{{▯gzl(a{(@CK\Ns\`9{WA{WHWD8foWXH{{▯8zlu&   solve ]a`z)dlzzhnl)`z)of
Zf?}QG}QNQB>`iQ^N}}y>|js •  solve [gf|/bj||nhj/f|/i`}/`z}/fa{j}anc/z|j/`acv!/Ncc/{gj/bj||nhj|/|g`zck/mj/▯}`{jl{jk/mv/▯n||x`}k!/[gn{/▯n||x`}k/f|/GDL[It[g>|PF|POPC?ahP_O||x?}kr!•
q;5A}ta5etffbzgq5|f5]^VASnA}$fJ\fJUJY%{rJEUffb%gqh;  solve B~▯e6{seewqs6▯e6pyd6ycd6▯xbsdxwz6ces6yxzo86Wzz6b~s6{seewqse6e~yczr6ts6fdybsubsr6to6fweeaydr86B~wb6fweeaydr6▯e6^]UBPmB~
px<~y<lnshy▯hyx<~e<l}ooksnx2<Ht}h<l}ooksnx<uo<TW_HZgHt-oCUoC\CP,r{CL\ook,nxa2•  solve Iutn=pxnn|zx=tn={ro=rho=tsixos|q=hnx=rsqd3=\qq=iux=pxnn|zxn=nurhqy=▯x=morix~ixy=▯d=m|nnjro
MOZ
•bOO•WKF•NFPPBDFP•PKLVOG•AF•SQLWF@WFG•AZ•SBPPTLQG
•wKBW•SBPPTLQG•JP•kh`weXwK•P|jP|c|o•MD|scPPT•QG^
```

Flag : HKCTF{Th1s_Is_@_L0ng_P@ssw0rd}

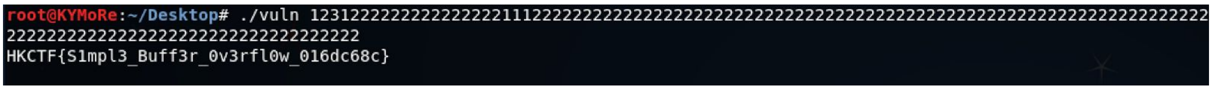
Pwn 100: Buffer Overflow

Description

Can you overflow the right buffer in this program, vuln, to get the flag? You can read the source (has the same functionality as) of the program, vuln.c, if you want to.

Solution:

Just input a long argument that is longer than the buffer to cause buffer flow, then the program will lead us to the flag.



```
root@KYMoRe:~/Desktop# ./vuln 12312222222222222221112222222222222222222222222222222222222222222222222222222222222222222222222222
22222222222222222222222222222222222
HKCTF{S1mpl3_Buff3r_0v3rfl0w_016dc68c}
```

Flag: HKCTF{S1mpl3_Buff3r_0v3rfl0w_016dc68c}

Reverse Engineering 100: Simple APK

Description

Find the flag in the APK

Solution:

Just open it with notepad and search the flag

Flag: HKCTF{pwyorowkw3}

Crypto 300: RSA 1

Description

A customer suspected that his message are cracked by somebody. Can you confirm the issue by decrypting the message below?

c:
1267512865767235284579628962679981517661651162006242932307504395229069157817213250932198187

n:
1441341319160614646189772775947458689224268167522603816085381857689639120119532883988453931

e: 65537

Solution:

https://www.alpertron.com.ar/ECM.HTM

Use the above website to do the factorization

Press the **Help** button to get help about this application. Press it again to return to the factorization. Keyboard users can press CTRL+ENTER to start factorization. This is the WebAssembly version.

- 1 441341 319160 614646 189772 775947 458689 224268 167522 603816 085381 857689 639120 119532 883988 453931 (91 digits) = 1 334556 412516 773775 580925 026039 868529 270309 (43 digits) × 1 080015 281214 272920 245486 385411 085443 503979 257359 (49 digits)

Get the flag via RSACTFTOOL.

```
root@KYMoRe:~/RsaCtfTool# python RsaCtfTool.py -p 1334556412516773775580925026039868529270309 -q 1080015281
2142729202454863854110854435039792573 59 -e 65537 --uncipher 1267512865767235284579628962679981517661651162 0
0624293230750439522906915781721325093219818 7
[+] Clear text : HKCTF{rs@ crypt0 us3 10801193}
```

Flag: HKCTF{rs@_crypt0_us3_10801193}

Crypto 450: RSA 2

Description

The customer fixed the previous issue. Can you decrypt the message this time?

n :
12301141972724292960585948437971278722411942786812218502841442603874721196772812668708222319195958380012403093044253399755799762549531260814883719682766538294441114283781632163571070754847369804586838545770200114861944189730393681376431146673015470955622822572616394605670811484761576817673309001

c :
52127047932811110668013864133349571790867805534855543297048613670965313472772560575182413718393507843165300390227330555558736664829280206143700900677133430280286900207970264306716622456672164011284966405111997950821662174402552515880677459669874493401227056820044471828999856535032257525489909965248714298022

e :
20370827750732677953101194500404700852089173301382884082478321647291201786559551992537091540692087873762090234342322985115231826746732803397154738501467143140654322623572067396058860233911575260540468106570172920030157403146163826401598627492164762337650828047117823273414399019740348998847585859920303350373

Solution:

Get the flag via RSACTFTOOL directly.

```
root@KYMoRe:~/RsaCtfTool# python RsaCtfTool.py -n 12301141972724292960585948437971278722411942786812218502841442603874721196772812668708222319195958380012403093044253399755799762549531260814883719682766538294441114283781632163571070754847307015584514936980458683854577020011486194418973039368137643114667301547095562282257261639460567081148476157681767330901 -e 20370827750732677953101194500404700852089173301382884082478321647291201786559551992537091540692087873762090234342322985115231826746732803397154738501467143140654322623572067396058860233911575260540468106570172920030157403146163826401598627492164762337650828047117823273414399019740348998847585859920303350373 --uncipher 52127047932811110668013864133349571790867805534855543297048613670965313472772560575182413718393507843165300390227330555558736664829280206143700900677133430280286900207970264306716622456672164011284966405111997950821662174402552515880677459669874493401227056820044471828999856535032257525489909965248714298022
[+] Clear text : HKCTF{w@tch_y0ur_rs@_c@r3fu11y_87027243203}
```

Flag: HKCTF{w@tch_y0ur_rs@_c@r3fu11y_87027243203}

Misc 50: Bit Key

Description

Can you find the password to gain access from the script?

Solution:

Code

```
string =['{','}','[',']','\','|',';',':','"',',','<','>','.','/','?','0','1','2','3','4','5','6','7','8',
while(1):
    jiami = (int)(input("Input: "))
    for i in range(0,(len(string)-1)):
        g = ( (((ord(string[i]) << 5) | (ord(string[i]) >> 3)) ^ 123) & 255 )
        if (g == jiami):
            print("related char")
            print(string[i])
```

Input the number from python code [114, 18, 19, 241, 179, 20, 87, 144, 21, 29, 23, 53, 29, 245, 144, 22, 29, 84, 212] step by step and get the original character.

```
PS C:\Users\Kin\Desktop> python test123.py
Input: 114
related char
H
Input: 18
related char
K
Input: 19
related char
C
Input: 241
related char
T
Input: 179
related char
F
Input: 20
related char
{
Input: 87
related char
a
```

And combine all the characters to the flag: HKCTF{a_s3cr3t_k3y}

Fflag: HKCTF{a_s3cr3t_k3y}

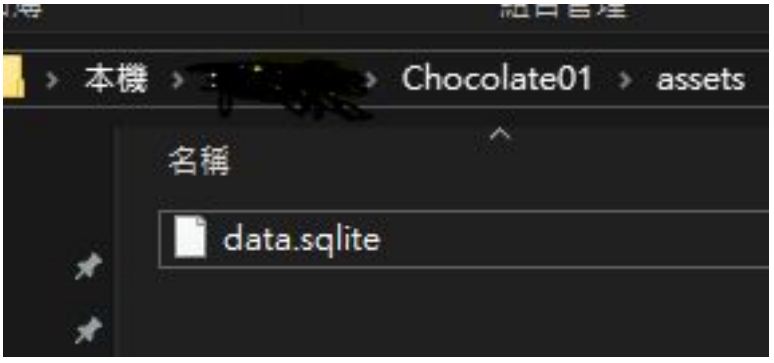
Reverse Engineering 125: Secret in Android App 1

Description

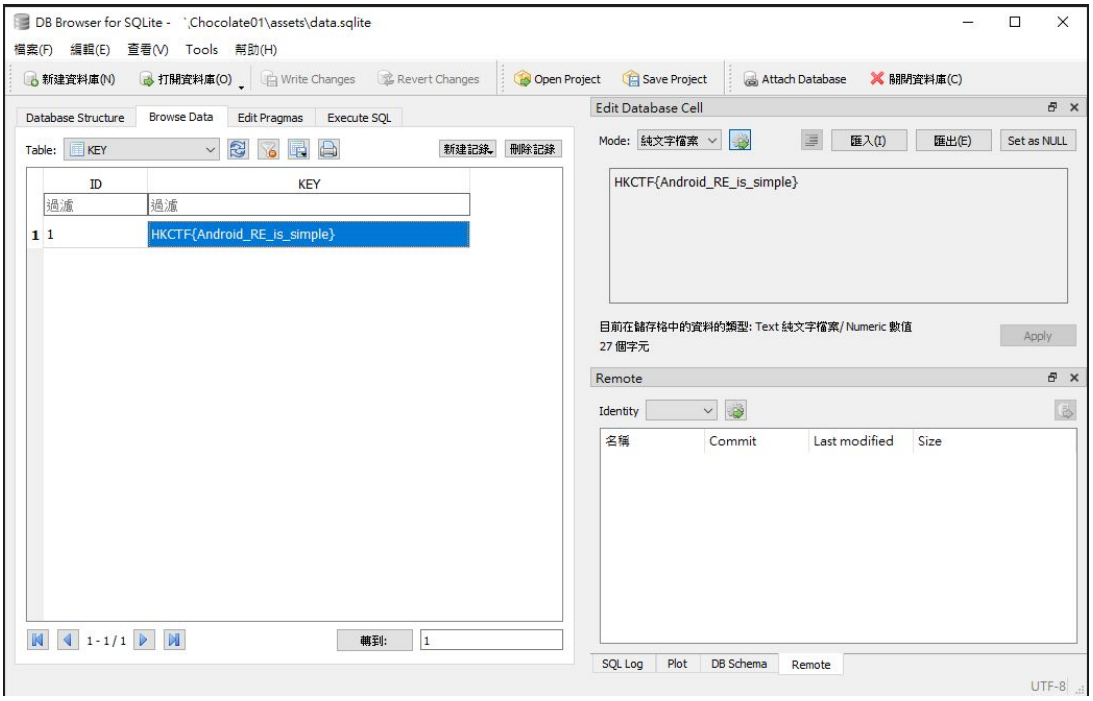
Something interesting is stored in the Android app. But I can't find it because I don't know the PIN. Can you help me?

Solution:

Extract the apk file and open assets folder



Use the Sqlite tool to view the sql file from assets and get the flag



Flag: HKCTF{Anroid_RE_is_simple}