Write'ups KerberINT 2017

Begin here

Welcome to the jungle

CTF{w3lc0me_t0_the_jungl3}

Connect to IRC

https://chat.rezosup.org/ Connect at #CTF2017

CTF{f4ceb00k_c_3st_0utd4ted}

Toolbox

```
aurelien@aurelien:~$ ssh toolbox@157.159.40.161
toolbox@157.159.40.161's password:
Votre dossier temporaire est: /tmp/tmp.DEBvriuKxv
Notez le quelque part si vous voulez le réutiliser plus tard
toolbox@ctfgate:/tmp/tmp.DEBvriuKxv$ ls
flag.txt
toolbox@ctfgate:/tmp/tmp.DEBvriuKxv$ cat flag.txt
CTF{v13ns_t4ter_m3s_gr0s_out1ls}
toolbox@ctfgate:/tmp/tmp.DEBvriuKxv$ exit
exit
Connection to 157.159.40.161 closed.
```

NEWS

Challenge | 16 Solves

Update

10

Android Malware has been updated !
The challenge was not possible according to the creator... we are sorry.

Here is an easy flag for you :
CTF{you_dumb_creator}

"Seul Link Peut Vaincre Ganon" team, you get 100 points because you solved an impossible challenge, or because you find a nice method to solve it.

Key

SUBMIT

Web

Nothing to Hide

https://manwefm.hosting.minet.net/

Inspecter :
Ctrl+Maj+i ou Ctrl+u

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>KerberINT 2016 - Move Along</title>
        <link rel="stylesheet" type="text/css" href="./main.css">
    </head>
    <body>
        <img src="nothing.jpg">
<!--

CTF{g00d_boy}
-->

</body></html>
```

Web

Spotlight

https://manwefm.hosting.minet.net/

```
console.log("DEBUG: CTF{5tup1d_d3v5_w1th_th31r_l095}");
console.log("DEBUG: Thank you IceCTF2016 !");

console.log("DEBUG: Loading up helper functions...");
console.log("DEBUG:     * getMousePos(canvas, evt)");
function getMousePos(canvas, evt) {
    var rect = canvas.getBoundingClientRect();
    return {
        x:  evt.clientX - rect.left,
        y:  evt.clientY - rect.top
    };
}
```

Inspecter :
Ctrl+Maj+i ou Ctrl+u

```
:"./spotlight.js"><
```

Web

Index

https://manwefm.hosting.minet.net/INDEX/

https://manwefm.hosting.minet.net/

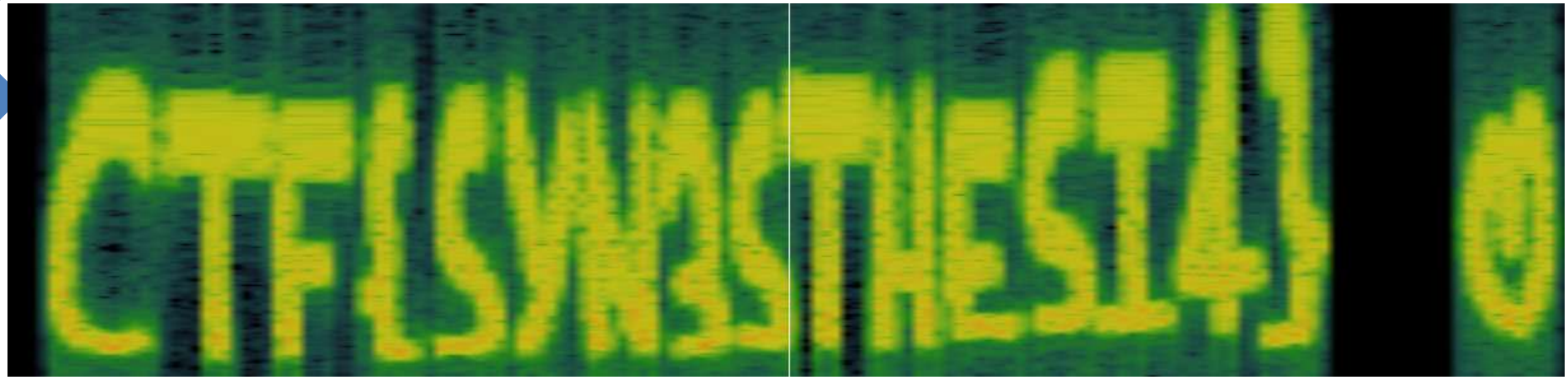
https://manwefm.hosting.minet.net/INDEX/.index.html

CTF{That_was_not_that_hard_right?}

Stegano

```
aurelien@aurelien:~/Documents/dossiers_windows/Telecom_SudParis/KerberIN
T/CTF1/6 Stegano$ sonic-visualiser message_from_space.wav
```

Layer → Add Spectrogramm → All Channel Mixed (Shift + G)



Forensic

A walk in the zoo

https://doc.ubuntu-fr.org/photorec

```
aurelien@aurelien:~/Documents/dossiers_windows/Telecom_SudParis/KerberINT/CTF1/1
 Forensic/1 A walk in the zoo$ photorec USBKEY_JEANMICH_PERSO.img
```

```
Select a media (use Arrow keys, then press Enter):
>Disk USBKEY_JEANMICH_PERSO.img - 63 MB / 61 MiB (RO)
>[Proceed ]  [  Sudo  ]  [  Quit  ]
```

```
     Partition                  Start        End    Size in sectors
      Unknown                  0   0  1  1007   0 60      124928 [Whole disk]
>   P FAT16                    0   0  1  1007   0 60      124928 [NO NAME]
```

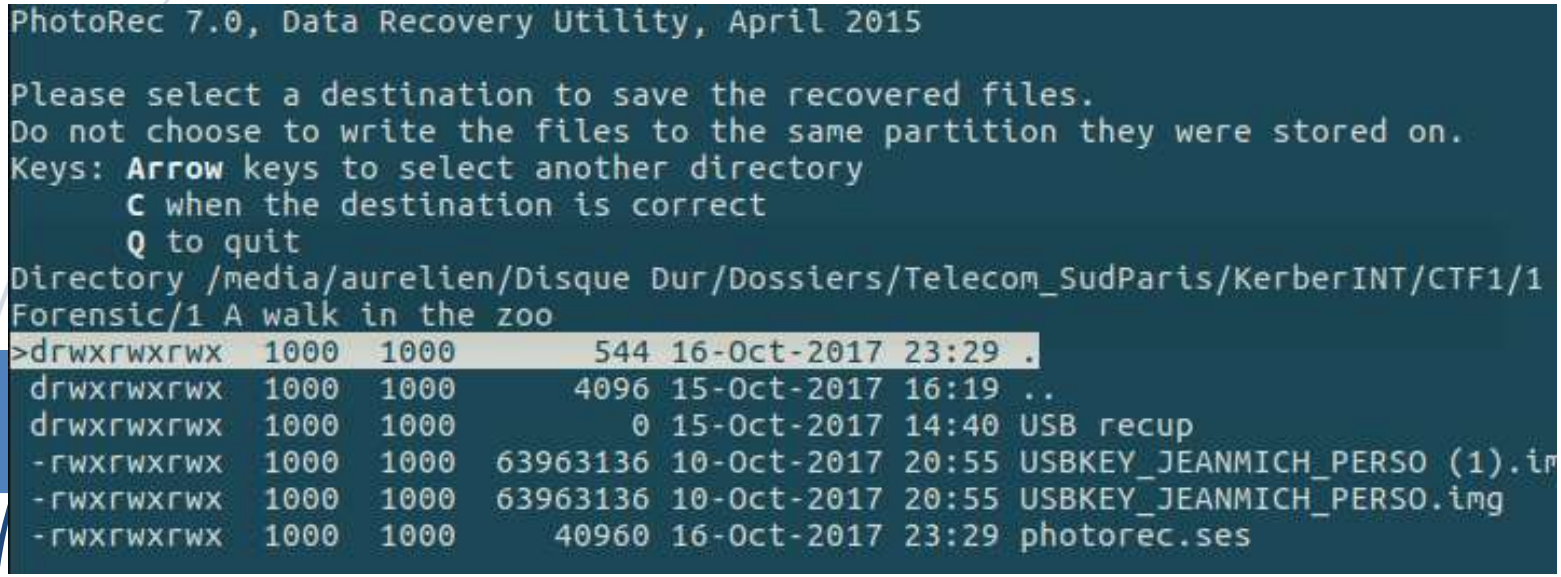
```
 [ ext2/ext3 ] ext2/ext3/ext4 filesystem
>[ Other     ] FAT/NTFS/HFS+/ReiserFS/...
```

```
Please choose if all space need to be analysed:
 [   Free    ] Scan for files from FAT16 unallocated space only
>[   Whole   ] Extract files from whole partition
```

Forensic

A walk in the zoo

https://doc.ubuntu-fr.org/photorec



Forensic

A walk in the zoo

https://doc.ubuntu-fr.org/photorec

Forensic

http://dabeaz.blogspot.fr/2010/08/decoding-superboard-ii-cassette-audio.html

Biiiiip bip biip bip

```
#! /bin/bash
python3 kcs_decode.py Tape.wav >> resultat.txt
nb_ligne=$(wc -l resultat.txt | cut -d ' ' -f1)
iteration=`expr $nb_ligne - 1`
for i in `seq 4 $iteration`;do
        echo -n $(sed -n ""$i"p" resultat.txt | cut -c1 | grep -v ^M)
done
rm resultat.txt
echo " "
```

```
aurelien@aurelien:~/Documents/dossiers_windows/Telecom_SudParis/Kerber
INT/CTF1/1 Forensic/2 Biiiiip bip biip bip$ ./flag.sh
CTF{r1ck_4stl3y_r3memb3r_f0r3ver}
```

```
aurelien@aurelien:~/Documents/dossiers_windows/Telecom_SudPa
ris/KerberINT/CTF1/1 Forensic/2 Biiiiip bip biip bip$ python
3 kcs_decode.py Tape.wav
NEVER GONNA GIVE YOU UP


CWe're no strangers to love
TYou know the rules and so do I
FA full commitment's what I'm thinking of
{You wouldn't get this from any other guy

rI just want to tell you how I'm feeling
1Gotta make you understand

cNever gonna give you up, never gonna let you down
kNever gonna run around and desert you
_Never gonna make you cry, never gonna say goodbye
4Never gonna tell a lie and hurt you

sWe've known each other for so long
tYour heart's been aching but you're too shy to say it
lInside we both know what's been going on
3We know the game and we're gonna play it

yAnd if you ask me how I'm feeling
_Don't tell me you're too blind to see

rnNever gonna give you up, never gonna let you down
3Never gonna run around and desert you
mNever gonna make you cry, never gonna say goodbye
eNever gonna tell a lie and hurt you

mNever gonna give you up, never gonna let you down
bNever gonna run around and desert you
3Never gonna make you cry, never gonna say goodbye
rNever gonna tell a lie and hurt you

_We've known each other for so long
fYour heart's been aching but you're too shy to say it
0Inside we both know what's been going on
rWe know the game and we're gonna play it

3I just want to tell you how I'm feeling
vGotta make you understand

eNever gonna give you up, never gonna let you down
rNever gonna run around and desert you
}Never gonna make you cry, never gonna say goodbye
Never gonna tell a lie and hurt you
```

Forensic

Memory Dump

```
aurelien@aurelien:~/Documents/dossiers_windows/Telecom_SudParis/KerberINT/CTF1/1 Forensic/3 Memory  dump$ bless memorydump.dmp
```

memorydump.dmp

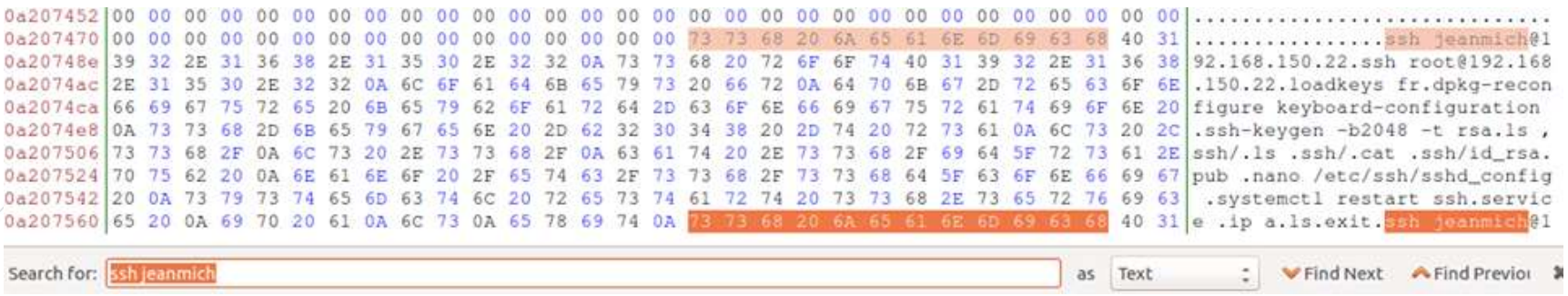
```
074ba468 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....................
074ba47c 00 00 00 00 2D 2D 2D 2D 2D 42 45 47 49 4E 20 52 53 41 20 50 ....-----BEGIN RSA P
074ba490 52 49 56 41 54 45 20 4B 45 59 2D 2D 2D 2D 2D 0A 4D 49 49 45 RIVATE KEY-----.MIIE
074ba4a4 6F 77 49 42 41 41 4B 43 41 51 45 41 78 52 6E 76 49 72 6F 63 owIBAAKCAQEAxRnvIroc
074ba4b8 54 2B 4A 73 6D 54 37 6B 34 37 34 33 77 56 6C 78 73 76 7A 4C T+JsmT7k4743wVlxsvzL
074ba4cc 57 31 63 74 64 32 4B 5A 78 51 4C 6F 62 45 39 6E 58 6F 72 58 W1ctd2KZxQLobE9nXorX
074ba4e0 0A 71 6A 2B 72 6F 64 39 55 35 2F 62 6B 47 70 38 66 4A 61 71 .qj+rod9U5/bkGp8fJaq
074ba4f4 51 53 6C 78 57 6B 79 6E 72 52 64 7A 77 55 74 4E 44 78 6E 74 QSlxWkynrRdzwUtNDxnt
074ba508 63 38 78 64 7A 4A 6D 58 56 2B 43 32 77 4B 4B 31 46 6A 6B 33 c8xdzJmXV+C2wKK1Fjk3
074ba51c 39 50 47 44 59 0A 2B 49 4D 5A 58 32 45 59 2B 6D 31 39 51 6D 9PGDY.+IMZX2EY+m19Qm
074ba530 7A 4A 65 30 45 66 41 64 79 31 47 72 65 4B 54 74 4B 65 37 48 zJe0EfAdy1GreKTtKe7H
074ba544 63 77 6E 52 4F 61 79 52 70 46 53 37 53 31 47 38 41 49 34 50 cwnROayRpFS7S1G8AI4P
074ba558 53 64 46 39 4C 38 59 68 4E 48 0A 43 76 64 54 63 37 74 74 6A SdF9L8YhNH.CvdTc7ttj
074ba56c 68 50 30 6A 66 53 4A 7A 49 69 32 6B 76 63 31 4A 46 63 4D 54 hP0jfSJzIi2kvc1JFcMT
074ba580 6A 49 36 56 71 75 57 38 63 50 54 2F 46 77 50 47 76 67 6E 7A jI6VquW8cPT/FwPGvgnz
074ba594 69 2B 41 2F 7A 69 31 59 79 2B 6F 57 4A 63 59 0A 76 53 75 5A i+A/zilYy+oWJcY.vSuZ
074ba5a8 54 71 53 36 31 43 76 76 4F 4B 49 54 61 53 65 33 68 61 2B 2F TqS61CvvOKITaSe3ha+/
074ba5bc 39 71 77 35 61 62 33 63 58 58 64 37 41 5A 76 7A 48 74 2B 68 9qw5ab3cXXd7AZvzHt+h
074ba5d0 4E 4D 46 59 79 54 62 4B 76 51 42 46 72 43 50 59 32 6F 59 32 NMFYyTbKvQBFrCPY2oY2
074ba5e4 0A 2F 73 4B 78 4C 78 48 52 57 4A 57 49 69 33 34 64 34 38 70 ./sKxLxHRWJWIi34d48p
074ba5f8 59 5A 63 6F 6D 6D 51 61 38 55 74 47 4B 72 51 61 4C 44 77 49 YZcommQa8UtGKrQaLDwI
074ba60c 44 41 51 41 42 41 6F 49 42 41 44 4B 62 58 51 34 69 6B 51 42 DAQABAoIBADKbXQ4ikQB
074ba620 6C 31 35 4A 6C 0A 56 71 6D 75 2F 6D 76 54 68 6D 44 35 6A 56 l15Jl.Vqmu/mvThmD5jV
074ba634 45 76 69 67 6E 38 4E 42 55 6B 6B 50 36 32 53 49 6B 36 78 76 Evign8NBUkkP62SIk6xv
074ba648 4C 38 6C 42 6E 6B 32 4D 30 53 56 44 45 4E 71 42 61 6F 75 35 L8lBnk2M0SVDENqBaou5
074ba65c 47 4D 4D 65 4A 70 51 66 4A 58 0A 4C 78 46 6C 45 79 4C 55 4B GMMeJpQfJX.LxFlEyLUK
074ba670 2B 4A 79 6F 2B 73 34 2F 48 35 54 5A 4D 61 63 37 70 34 43 49 +Jyo+s4/H5TZMac7p4CI
074ba684 6B 49 74 62 36 67 6E 4B 51 32 58 52 61 32 50 44 54 6C 50 36 kItb6gnKQ2XRa2PDTlP6
074ba698 30 2B 4E 51 65 50 64 67 6C 61 51 78 35 39 44 0A 62 76 43 79 0+NQePdglaQx59D.bvCy
074ba6ac 44 62 5A 58 41 55 67 66 53 61 6D 4E 68 57 37 52 6D 4C 59 42 DbZXAUgfSamNhW7RmLYB
074ba6c0 75 4A 41 54 39 55 32 68 52 52 79 6B 65 69 30 2F 53 31 35 2B uJAT9U2hRRykei0/S15+
074ba6d4 6F 32 4F 70 53 74 31 36 6A 34 70 33 76 6F 77 53 48 71 34 2B o2OpSt16j4p3vowSHq4+
074ba6e8 0A 62 50 37 57 64 35 53 58 4D 4B 62 70 57 71 34 33 77 77 41 .bP7Wd5SXMKbpWq43wwA
074ba6fc 4D 4E 32 53 71 37 53 33 51 6B 38 46 52 6C 57 53 68 49 45 6C MN2Sq7S3Qk8FRlWShIEl
074ba710 38 6D 64 69 4A 31 69 57 61 64 57 61 76 58 59 6F 34 44 6D 4B 8mdiJ1iWadWavXYo4DmK
074ba724 77 54 73 65 52 0A 4C 49 72 61 6C 62 6A 74 64 4C 55 63 55 4A wTseR.LIralbjtdLUcUJ
074ba738 32 51 6F 2B 64 6E 79 76 76 69 64 6A 49 75 6D 33 63 36 42 39 2Qo+dnyvvidjIum3c6B9
074ba74c 50 74 37 4A 36 79 42 64 68 51 4F 6F 6E 55 49 63 62 33 44 59 Pt7J6yBdhQOonUIcb3DY
```

Search for: BEGIN RSA as Text Find Next Find Previou

```
aurelien@aurelien:~/Documents/dossiers_windows/Telecom_SudParis/KerberINT/CTF1/1 Forensic/3 Memory  dump$ cat rsa_key
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAxRnvIrocT+JsmT7k4743wVlxsvzLW1ctd2KZxQLobE9nXorX
qj+rod9U5/bkGp8fJaqQSlxWkynrRdzwUtNDxntc8xdzJmXV+C2wKK1Fjk39PGDY
+IMZX2EY+m19QmzJe0EfAdy1GreKTtKe7HcwnROayRpFS7S1G8AI4PSdF9L8YhNH
CvdTc7ttjhP0jfSJzIi2kvc1JFcMTjI6VquW8cPT/FwPGvgnzi+A/zi1Yy+oWJcY
vSuZTqS61CvvOKITaSe3ha+/9qw5ab3cXXd7AZvzHt+hNMFYyTbKvQBFrCPY2oY2
/sKxLxHRWJWIi34d48pYZcommQa8UtGKrQaLDwIDAQABAoIBADKbXQ4ikQBl15Jl
Vqmu/mvThmD5jVEvign8NBUkkP62SIk6xvL8lBnk2M0SVDENqBaou5GMMeJpQfJX
LxFlEyLUK+Jyo+s4/H5TZMac7p4CIkItb6gnKQ2XRa2PDTlP60+NQePdglaQx59D
bvCyDbZXAUgfSamNhW7RmLYBuJAT9U2hRRykei0/S15+o2OpSt16j4p3vowSHq4+
bP7Wd5SXMKbpWq43wwAMN2Sq7S3Qk8FRlWShIEl8mdiJ1iWadWavXYo4DmKwTseR
LIralbjtdLUcUJ2Qo+dnyvvidjIum3c6B9Pt7J6yBdhQOonUIcb3DY4aT6K7lm/e
4k+uQGECgYEA9ctwmTbRaIJSQWuTtYcK7U+FMj6rKLAWMDOuoFZlGoaK94+MKm1O
7wjsrAV/ZVPhTgxSOwmscSpBliQQJ0SnWuKyV8joxHE+FS+xvykuMuEVo9PNVL0S
8FBILPcbBeXiQ4dsohlMbU3Mz2EPDnQMK8GRH6AG1QyAz9EsbDiPTnECgYEAzUju
P8ZHjtA5AwfcnCkOisCxWQP+v3Xukq1PFJavg8A9TLsKYXeZp6aGtWDl4DaH0jqK
Orq3HSYINM/wt01414Njjiz/z71/R7jaj8M3bWhr71EkMbF9PuSO2w75nnRRiFWt
vxl/FKmfnKd0a6sNirHT5BdLf/cPluw2MYobMX8CgYAalGBdwSbOBWG9v1NeyvasiO/fahV6NeiAV+ZnYlBWmPwIH1bJ874YbT+iromDPJndpclypu4l4qrrAIVaqbrrQh6FDa3WKGTNR7YR9PMQ/3Qmni3THwsgP54+do5Oifh40vk1YkT4gPRH7LgDGRuD08/MurJicZKwZm9GWKtukQKBgQCx0TqOg3H4guTfey/dFLND/SWhtZQ0xDKjWvvlGaFuvlZa4KldQ/wU/GozMnvse0+rCoox1QpIdWkmVfhszu6dv3G89b2gdMSLm9O2M6MPYiQcaGyxnsyebdxQY+hVGOyG67BUmAE8JLHdQC/YE5LnUKH6MTN05MzcLj/DKZ3kgQKBgHu7tXlw9S7NCDOlJPPhGgdxCg6TZj5xOvc4Orpuj+AaTwXGPv4H+plYDeQx6Zj2vM9a7JHUKL7RXusxn77FgDJ4SPeHW0Nm8NQDidC/F5s5mcWCx7OfeBKWBxsFsZCR5Nw1I15lBCnzUzGuh/dFHBvxnHcgc+VZh4d3T5otYWuW
-----END RSA PRIVATE KEY-----
```

Forensic

Memory Dump




```
pierrick@HP-Pavilion-TS-11:~/Téléchargements$ chmod 600 key_rsa
pierrick@HP-Pavilion-TS-11:~/Téléchargements$ ssh -i key_rsa jeanmich@157.159.40.161
Linux ctfgate 4.9.0-3-amd64 #1 SMP Debian 4.9.30-2+deb9u3 (2017-08-06) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Oct 17 14:17:37 2017 from 157.159.40.161
jeanmich@ctfgate:~$ ls
flag.txt
jeanmich@ctfgate:~$ cat flag.txt
CTF{je4nm1ch_h4s_4_g00d_m3m0ry}
```

Crypto

Basic encoding

https://conv.darkbyte.ru/

Q1RGe2l0c2U2NF8= → BASE 64 to Text → **CTF{b4se64_**

34 6e 64 5f 68 33
78 61 64 65 63 69
6d 34 6c 7d → Hexadécimal to Text → **4nd_h3xadecim4l}**

CTF{b4se64_4nd_h3xadecim4l}

Crypto

1+3+3=7

https://www.wikiwand.com/fr/Leet_speak

13m07d3p4553357r34l173 → Leet speak → **lemotdepasseestrealite**

CTF{realite}

Crypto

Easy Cipher

https://conv.darkbyte.ru/

00110101 00110001 00100000 00110011 00110001 00100000 00110101
00110010 00100000 00110100 00110111 00100000 00110110 00110101
00100000 00110011 00110010 00100000 00110011 00110100 00100000
00110111 00110111 00100000 00110101 00111000 00100000 00110011
00110011 00100000 00110100 00111001 00100000 00110011 00110000
00100000 00110101 01100001 00100000 00110011 00110010 00100000
00110101 00110110 00100000 00110110 00110110 00100000 00110101
01100001 00100000 00110100 00110100 00100000 00110100 01100101
00100000 00110110 00110110 00100000 00110110 00110010 00100000
00110101 00110100 00100000 00110100 00110010 00100000 00110111
00110101 00100000 00110101 00111000 00100000 00110011 00110010
00100000 00110101 00110110 00100000 00110111 00110101 00100000
00110101 00111001 00100000 00110111 01100001 00100000 00110100
00110010 00100000 00110110 01100010 00100000 00110100 01100101
00100000 00110100 00110111 00100000 00110110 00110011 00100000
00110111 01100001 00100000 00110110 00110110 00100000 00110101
00110001 00100000 00110011 01100100

Binary to Text

51 31 52 47 65 32 34 77 58 33
49 30 5a 32 56 66 5a 44 4e 66
62 54 42 75 58 32 56 75 59 7a
42 6b 4e 47 63 7a 66 51 3d

Hex to Text

Base64 to Text

Q1RGe24wX3I0Z2VfZDN
fbTBuX2VuYzBkNGczfQ=

CTF{n0_r4ge_d3_m0n_enc0d4g3}

Crypto

@/bin/bash

http://rumkin.com/tools/cipher/atbash.php

fm kvf kofh wfi jf'fm hrnkov xlwv wv xvhzi nzrh kzh hr wfi klfi zfgzmg vg ov kzhh vhg hfyhgrgfgrlm_hfxph

un peu plus dur qu'un simple code de cesar mais pas si dur pour autant et le pass est substitution_sucks

CTF{substitution_sucks}

Crypto

```
#! /bin/bash

secret=$(cat /dev/urandom | tr -dc 'a-z' | fold -w 5 | head -n 1)
flag=CTF\{$secret\}
echo $flag
echo -n $flag | md5sum
```

612eb3c0d9879006cb2beef6fa3c8cd2

Hashing

```
ubuntu@ubuntu:/media/ubuntu/Disque Dur/Dossiers/Telecom_SudParis/Kerb
/CTF1/2 Crypto/5 Hashing$ echo -n "CTF{tcvlb}" | md5sum
612eb3c0d9879006cb2beef6fa3c8cd2  -
ubuntu@ubuntu:/media/ubuntu/Disque Dur/Dossiers/Telecom_SudParis/Kerb
erINT/CTF1/2 Crypto/5 Hashing$ cat message.txt
"612eb3c0d9879006cb2beef6fa3c8cd2  -"
```

```
#! /bin/bash

msg_hash="$1"
ALPHA="a b c d e f g h i j k l m n o p q r s t u v w x y z"
for carac1 in $ALPHA; do
        for carac2 in $ALPHA; do
                for carac3 in $ALPHA; do
                        for carac4 in $ALPHA; do
                                for carac5 in $ALPHA; do
                                        secret="$carac1$carac2$carac3$carac4$carac5"
                                        flag=CTF\{$secret\}
                                        resultat=$(echo -n $flag | md5sum)
                                        echo "$flag"
                                        if [ "$resultat" = $msg_hash ]; then
                                                echo "$flag"
                                                break 5
                                        fi
                                done
                        done
                done
        done
done
```

Programming

Find the flag

ubuntu@ubuntu: /media/ubuntu/Disque Dur/Dossiers/Telecom_SudParis/KerberINT/CTF1

```
#! /bin/bash
flag=$1
i=$2
echo "$flag$i" | nc 157.159.40.161 3333 | head -n 4 | tail -n 1
```

CTF{p3tit_4_pet1t_l_0ise4u_f41t_s0n_nid}

```
#! /bin/bash
alphabet="a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 _ { }"
while true; do
        for i in $alphabet; do
                reponse=$(exec ./reponse_netcat.sh "$flag" "$i")
                if [ "$reponse" = "Continue comme ça" ]; then
                        echo -n "$i"
                        break
                elif [ "$reponse" = "Bravo tu viens de trouver le flag !"]; then
                        echo -n "$i"
                        break 2
                fi
        done
done
echo " "
```

Programming

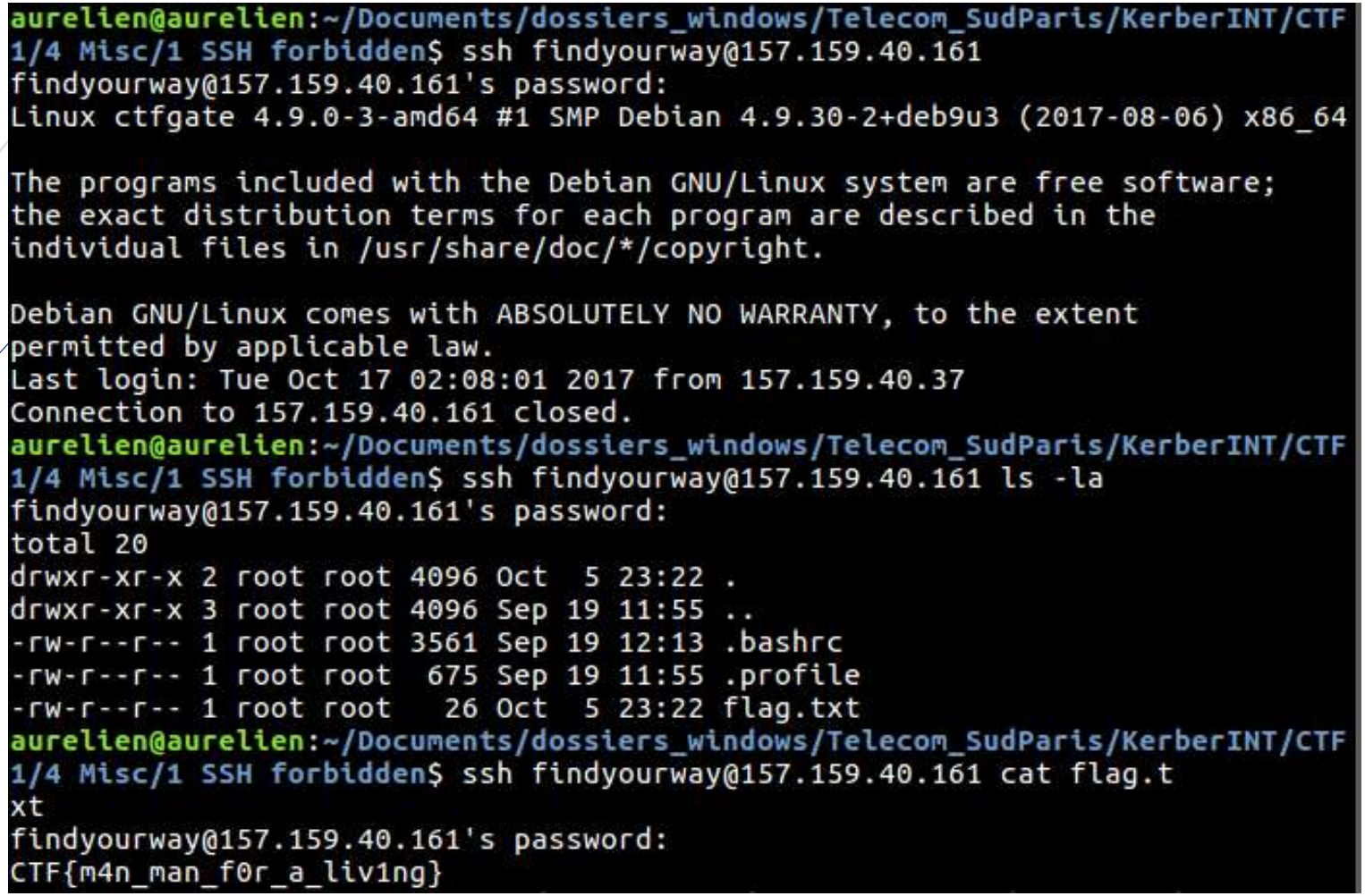
Find the flag

CTF{p3tit_4_pet1t_l_0ise4u_f41t_s0n_nid}

```
import socket
import time

hostname="localhost"
port=3333
ALPHA = '_}{0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
flag = 'CTF'
notFinished = True


s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((hostname, port))
s.recv(999)

while(notFinished):
    for i in range(len(ALPHA)):
        s.send(flag+ALPHA[i])
        time.sleep(0.05)
        data = s.recv(100)
        #print(str(i)+ " " +data)
        if "Continue" in data:
            flag = flag + ALPHA[i]
            print(flag)
            if ALPHA[i] == "}":
                notFinished = False
            break

print("Finished, flag is " + flag)
```

Misc

SSH Forbidden



Programming

Placed Under Surveillance

http://dialabc.com/sound/detect/

Recording.wav → DTMF TONES →

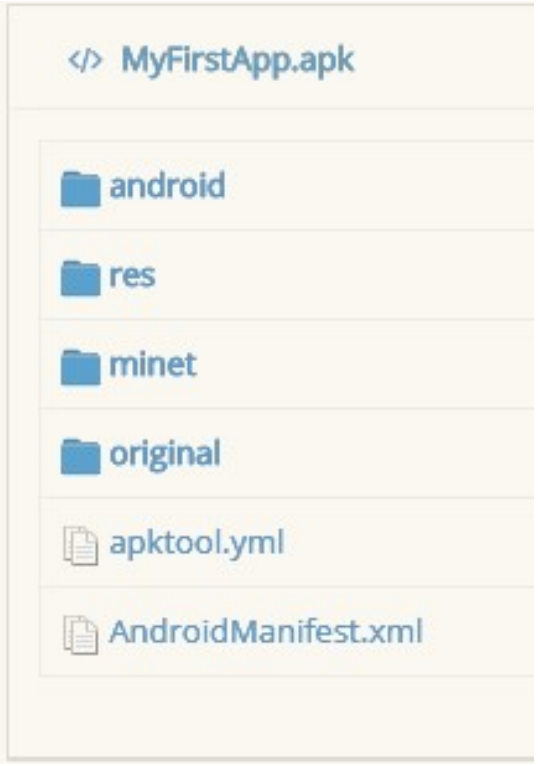
551165066488662266662233286666445522262446466

CTF{k1mj0ngunbombeatomik2main}

Misc

AndroidMyFirstApp

http://www.javadecompilers.com/apk



Misc

AndroidMyFirstApp

http://www.javadecompilers.com/apk

```
package minet.net.quickly;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;

public class Flag extends AppCompatActivity {
    static final int REP_DELAY = 50;
    static final String flag = "Q1RGe25pYW5fbmlhbl9uaWFuX25pYV9uaWFfbmlhbmFhYWFhYSF9Cg==";
    private Activity myActivity = this;
    private Handler repeatUpdateHandler = new Handler();

    class RptUpdater implements Runnable {
        RptUpdater() {
        }

        public void run() {
            Flag.this.startActivity(new Intent(Flag.this.myActivity, Troll.class));
        }
    }

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView((int) C0194R.layout.activity_flag);
        this.repeatUpdateHandler.postDelayed(new RptUpdater(), 100);
    }
}
```

Q1RGe25pYW5fbmlhbl9
uaWFuX25pYV9uaWFfb
mlhbmFhYWFhYSF9Cg==

Base64 to Text

**CTF{nian_nian_nian_n
ia_nia_nianaaaaaa!}**

Crypto

Vig Baby

```
#!/usr/bin/python
#import hashlib
#import sys

from itertools import cycle

ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_{}'

def encrypt(key, plaintext):
    """Encrypt the string and return the ciphertext"""
    pairs = zip(plaintext, cycle(key))
    result = ''

    for pair in pairs:
        (x,y)=pair
        print(x,y)
        total = ALPHA.index(x) + ALPHA.index(y)
        result += ALPHA[total % len(ALPHA)]

    return result

"""
key=sys.argv[1]
plaintext=sys.argv[2]+key
cipher=encrypt(key,plaintext)
print(cipher)
"""
```

```
>>> zip("01234",cycle("abc"))
[('0', 'a'), ('1', 'b'), ('2', 'c'), ('3', 'a'), ('4', 'b')]
```

Crypto

Vig Baby

```
ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_{}'
texte="TKlc_OLxUOhJMGbiVa0mhjHifvG6JiHDJZnhJ7ULikhxLra"

""" Calcul du debut de la cle dechiffrant CTF{ au debut """
def deb_cle(texte):
    debut=""
    msg="CTF{"
    extrait=texte[0:4]
    for loop in range(len(extrait)):
        lettre=extrait[loop]
        for lettre_cle in ALPHA:
            indice=(ALPHA.index(lettre)-ALPHA.index(lettre_cle))%len(ALPHA)
            if ALPHA[indice]==msg[loop]:
                debut+=lettre_cle
                break
    return debut
```

T	K	l	c
?	?	?	?
C	T	F	{

Crypto

Vig Baby

```
ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_{}'
texte="TKlc_OLxUOhJMGbiVa0mhjHifvG6JiHDJZnhJ7ULikhxLra"

""" Calcul du debut de la cle dechiffrant CTF{ au debut """
def deb_cle(texte):
    debut=""
    msg="CTF{"
    extrait=texte[0:4]
    for loop in range(len(extrait)):
        lettre=extrait[loop]
        for lettre_cle in ALPHA:
            indice=(ALPHA.index(lettre)-ALPHA.index(lettre_cle))%len(ALPHA)
            if ALPHA[indice]==msg[loop]:
                debut+=lettre_cle
                break
    return debut
```

T	K	l	c
R	4	g	e
C	T	F	{

Crypto

Vig Baby

```
""" Enregistrement Pairs key """
def stock_pairs_key(texte):
    rang=0
    debut=deb_cle(texte)
    key=[]
    for loop in range(len(debut)):
        key.append(debut[loop])
    key.append("#"+str(rang+1))
    L_selection=[]
    while len(key)<len(texte)-len(key):
        pairs_key=zip(texte,cycle(key))[len(texte)-len(key):]
        L_selection.append(pairs_key)
        rang+=1
        key.append("#"+str(rang+1))
    return L_selection
```

```
texte="TKlc_OLxUOhJMGbiVa0mhjHifvG6JiHDJZnhJ7ULikhxLra"
```

```
[('h', 'g'), ('x', 'e'), ('L', '#1'), ('r', 'R'), ('a', '4')]
[('k', '#2'), ('h', 'R'), ('x', '4'), ('L', 'g'), ('r', 'e'), ('a', '#1')]
[('i', '#2'), ('k', '#3'), ('h', 'R'), ('x', '4'), ('L', 'g'), ('r', 'e'), ('a', '#1')]
[('L', '#4'), ('i', 'R'), ('k', '4'), ('h', 'g'), ('x', 'e'), ('L', '#1'), ('r', '#2'), ('a', '#3')]
[('U', 'g'), ('L', 'e'), ('i', '#1'), ('k', '#2'), ('h', '#3'), ('x', '#4'), ('L', '#5'), ('r', 'R'), ('a', '4')]
```

Crypto

Vig Baby

```
""" Enregistrement combinaison clés : cyclées et ordonnées """
def stock_key(texte):
    L_selection=stock_pairs_key(texte)
    liste=[]
    for loop in range(len(L_selection)):
        resultat=[]
        for i in range(len(L_selection[loop])):
            (lettre,lettre_cle)=L_selection[loop][i]
            resultat.append(lettre_cle)
        liste.append(resultat)
    return liste

def stock_key_ordonnee(texte):
    import copy
    L_key_ordonnee=[]
    L_selection=stock_pairs_key(texte)
    debut=deb_cle(texte)
    key=[]
    for loop in range(len(debut)):
        key.append(debut[loop])
    for i in range(len(L_selection)):
        key.append("#"+str(i+1))
        X=copy.deepcopy(key)
        L_key_ordonnee.append(X)
    return L_key_ordonnee
```

```
['g', 'e', '#1', 'R', '4']
['#2', 'R', '4', 'g', 'e', '#1']
['#2', '#3', 'R', '4', 'g', 'e', '#1']
['#4', 'R', '4', 'g', 'e', '#1', '#2', '#3']
['g', 'e', '#1', '#2', '#3', '#4', '#5', 'R', '4']
```

```
['R', '4', 'g', 'e', '#1']
['R', '4', 'g', 'e', '#1', '#2']
['R', '4', 'g', 'e', '#1', '#2', '#3']
['R', '4', 'g', 'e', '#1', '#2', '#3', '#4']
['R', '4', 'g', 'e', '#1', '#2', '#3', '#4', '#5']
```

Crypto

Vig Baby

```
zip(texte,cycle(["R","4","g","e","#1","#2","#3","#4","#5"]))
```

```
[('T', 'R'), ('K', '4'), ('l', 'g'), ('c', 'e'), ('_', '#1'), ('0', '#2'), ('L', '#3'), ('x', '#4'), ('U', '#5'),
('0', 'R'), ('h', '4'), ('J', 'g'), ('M', 'e'), ('G', '#1'), ('b', '#2'), ('i', '#3'), ('V', '#4'), ('a', '#5'),
('0', 'R'), ('m', '4'), ('h', 'g'), ('j', 'e'), ('H', '#1'), ('i', '#2'), ('f', '#3'), ('v', '#4'), ('G', '#5'),
('6', 'R'), ('J', '4'), ('i', 'g'), ('H', 'e'), ('D', '#1'), ('J', '#2'), ('Z', '#3'), ('n', '#4'), ('h', '#5'),
('J', 'R'), ('7', '4'), ('U', 'g'), ('L', 'e'), ('i', '#1'), ('k', '#2'), ('h', '#3'), ('x', '#4'), ('L', '#5'),
('r', 'R'), ('a', '4')]
```

U	L	i	k	h	x	L	r	a
g	e	#1	#2	#3	#4	#5	R	4
R	4	g	e	#1	#2	#3	#4	#5

U	L	i	k	h	x	L	r	a
g	e	C	G	f	r	T	R	4
R	4	g	e	C	G	f	r	t

Crypto

Vig Baby

```
""" Cle definies """
def key_totale(texte):
    L_key_possible=[]
    L_key=stock_key(texte)
    L_key_ordonnee=stock_key_ordonnee(texte)
    for loop in range(len(L_key)):
        key=L_key[loop]
        key_ordonnee=L_key_ordonnee[loop]
        key_texte=zip(texte,cycle(key_ordonnee))[len(texte)-len(key):]
        finish=False
        while finish==False:
            for i in range(len(key)):
                (lettre,lettre_cle)=key_texte[i]
                resultat=key_ordonnee[i]
                if "#" in lettre_cle and "#" not in resultat:
                    indice=ALPHA.index(lettre)-ALPHA.index(resultat)
                    ajoute=ALPHA[indice]
                    key[i]=ajoute
                    key_texte[i]=(key_texte[i][0],ajoute)
                    emplacement=key_ordonnee.index(lettre_cle)
                    key_ordonnee[emplacement]=ajoute
                if "#" not in lettre_cle and "#" in resultat:
                    indice=ALPHA.index(lettre)-ALPHA.index(lettre_cle)
                    ajoute=ALPHA[indice]
                    emplacement=key.index(resultat)
                    key[emplacement]=ajoute
                    key_texte[emplacement]=(key_texte[emplacement][0],ajoute)
                    key_ordonnee[i]=ajoute
            compteur=0
            for i in range(len(key)):
                test1=("#" in key[i] and "#" not in key_ordonnee[i])
                test2=("#" not in key[i] and "#" in key_ordonnee[i])
                if test1 or test2:
                    compteur+=1
            if compteur==0:
                finish=True
        L_key_possible.append(key_ordonnee)
    return L_key_possible
```

Crypto

Vig Baby

```
""" Dechiffrage vigenere simple """
def decrypt_with_key(key,texte):
    decrypt_message=""
    pairs = zip(texte, cycle(key))
    for pair in pairs:
        (lettre,lettre_cle)=pair
        indice=(ALPHA.index(lettre)-ALPHA.index(lettre_cle)) % len(ALPHA)
        lettre_decode=ALPHA[indice]
        decrypt_message+=lettre_decode
    return decrypt_message

def phase_finale(texte):
    L_key_possible=key_totale(texte)
    L_flag_key=[]
    for loop in range(len(L_key_possible)):
        key=L_key_possible[loop]
        message=decrypt_with_key(key,texte)
        plaintext=message[:len(texte)-len(key)]
        key_msg=message[len(texte)-len(key):]
        chaine_caractere=""
        print message
        for i in range(len(key)):
            chaine_caractere+=key[i]
        if "}" in plaintext[len(plaintext)-1] and key_msg==chaine_caractere:
            L_flag_key.append([plaintext,key_msg])
    return L_flag_key


def solution(texte):
    [[flag,key]]=phase_finale(texte)
    print "FLAG = "+flag
    print "KEY = "+key


#solution(texte)

KEY = "R4gePasTuY3sPresqu3707"
FLAG = "CTF{v1gen3re_c_3st_sup3r}"
```

Crypto

Vig Baby

T	K	I	c	_	O	L	x	U	O	h	J	M
R	4	g	e	#1	#2	#3	#4	#5	#6	#7	#8	#9
C	T	F	{	?	?	?	?	?	?	?	?	?

G	b	i	V	a	0	m	h	j	H	i	f
#10	#11	#12	#13	#14	#15	#16	#17	#18	R	4	g
?	?	?	?	?	?	?	?	?	?	?	?

v	G	6	J	i	H	D	J	Z	n	h
e	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
R	4	g	e	#1	#2	#3	#4	#5	#6	#7

J	7	U	L	i	k	h	x	L	r	a
#11	#12	#13	#14	#15	#16	#17	#18	R	4	g
#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18

Crypto

Vig Baby

T	K	I	c	_	O	L	x	U	O	h	J	M
R	4	g	e	#1	#2	#3	#4	#5	#6	#7	#8	#9
C	T	F	{	?	?	?	?	?	?	?	?	?

G	b	i	V	a	0	m	h	j	H	i	f
#10	#11	#12	#13	#14	#15	#16	#17	#18	R	4	g
?	?	?	?	?	?	?	?	?	?	?	?

v	G	6	J	i	H	D	J	Z	n	h
e	P	a	s	T	u	Y	3	s	P	r
R	4	g	e	P	a	s	T	u	Y	3

J	7	U	L	i	k	h	x	L	r	a
e	s	q	u	3	7	0	7	R	4	g
P	r	e	s	q	u	3	7	0	7	e

Crypto

Vig Baby

T	K	I	c	_	O	L	x	U	O	h	J	M
R	4	g	e	P	a	s	T	u	Y	3	s	P
C	T	F	{	?	?	?	?	?	?	?	?	?

G	b	i	V	a	0	m	h	j	H	i	f
r	e	s	q	u	3	7	0	7	R	4	g
?	?	?	?	?	?	?	?	?	?	?	?

v	G	6	J	i	H	D	J	Z	n	h
e	P	a	s	T	u	Y	3	s	P	r
R	4	g	e	P	a	s	T	u	Y	3

J	7	U	L	i	k	h	x	L	r	a
e	s	q	u	3	7	0	7	R	4	g
P	r	e	s	q	u	3	7	0	7	e

Crypto

Vig Baby

T	K	I	c	_	O	L	x	U	O	h	J	M
R	4	g	e	P	a	s	T	u	Y	3	s	P
C	**T**	**F**	**{**	**v**	**1**	**g**	**3**	**n**	**3**	**r**	**e**	**_**

G	b	i	V	a	0	m	h	j	H	i	f
r	e	s	q	u	3	7	0	7	R	4	g
c	**_**	**3**	**s**	**t**	**_**	**s**	**u**	**p**	**3**	**r**	**}**

v	G	6	J	i	H	D	J	Z	n	h
e	P	a	s	T	u	Y	3	s	P	r
R	4	g	e	P	a	s	T	u	Y	3

J	7	U	L	i	k	h	x	L	r	a
e	s	q	u	3	7	0	7	R	4	g
P	r	e	s	q	u	3	7	0	7	e

Crypto

Good Ol' Crypto

```
#!/usr/bin/python
#import hashlib
#import sys

from itertools import cycle

ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_{}'

def encrypt(key, plaintext):
    """Encrypt the string and return the ciphertext"""
    pairs = zip(plaintext, cycle(key))
    result = ''
    for pair in pairs:
        (x,y)=pair
        #print(x,y)
        total = ALPHA.index(x) + ALPHA.index(y)
        result += ALPHA[total % 26]
    return result.lower()

"""
key=sys.argv[1]
plaintext=sys.argv[2]+key
cipher=encrypt(key,plaintext)
print(cipher)
"""
```

Crypto

Good Ol' Crypto

```
""" Dechiffrage en connaissance de la cle """
def possibilite(lettre,lettre_cle):
    indice_key=ALPHA.index(lettre_cle)
    L_possible=[]
    indice=ALPHA.index(lettre)
    indicek=(indice-indice_key)%26
    L_indice=[indicek+(26*i) for i in range(3)]
    for k in range(len(L_indice)):
        if L_indice[k]<len(ALPHA) and L_indice[k]>=0:
            L_possible.append(ALPHA[L_indice[k]])
    return L_possible
```

CTF{j_4i_p3rdu_la_cle_s0us_l3_c4nape}

FLAG

```
['C', 'c', '2']
['T', 't']
['F', 'f', '5']
['L', 'l', '{']
['J', 'j', '9']
['K', 'k', '_']
['E', 'e', '4']
['I', 'i', '8']
['K', 'k', '_']
['P', 'p']
['D', 'd', '3']
['R', 'r']
['D', 'd', '3']
['U', 'u']
['K', 'k', '_']
['L', 'l', '{']
['A', 'a', '0']
['K', 'k', '_']
['C', 'c', '2']
['L', 'l', '{']
['E', 'e', '4']
['K', 'k', '_']
['S', 's']
['A', 'a', '0']
['U', 'u']
['S', 's']
['K', 'k', '_']
['L', 'l', '{']
['D', 'd', '3']
['K', 'k', '_']
['C', 'c', '2']
['E', 'e', '4']
['N', 'n']
['A', 'a', '0']
['P', 'p']
['E', 'e', '4']
['M', 'm', '}']
```

KEY

```
['V', 'v']
['E', 'e', '4']
['C', 'c', '2']
['H', 'h', '7']
['D', 'd', '3']
['R', 'r']
['C', 'c', '2']
['H', 'h', '7']
['E', 'e', '4']
['R', 'r']
['L', 'l', '{']
['E', 'e', '4']
['C', 'c', '2']
['L', 'l', '{']
['E', 'e', '4']
['F', 'f', '5']
['C', 'c', '2']
['H', 'h', '7']
['E', 'e', '4']
['Z', 'z']
['M', 'm', '}']
['D', 'd', '3']
['M', 'm', '}']
['E', 'e', '4']
['S', 's']
['O', 'o']
['U', 'u']
['S', 's']
['L', 'l', '{']
['D', 'd', '3']
['C', 'c', '2']
['A', 'a', '0']
['N', 'n']
['E', 'e', '4']
['P', 'p']
['E', 'e', '4']
```

Forensic

Android Malware

tar –xvf malware.tgz
mount data.img /mnt/data/

/mnt/data/data/com.android.wannacry/files/Mzg1N2Nj/MGJhOGZj/N2ZiZjIz/Application2Cryptage.apk

http://www.javadecompilers.com/apk

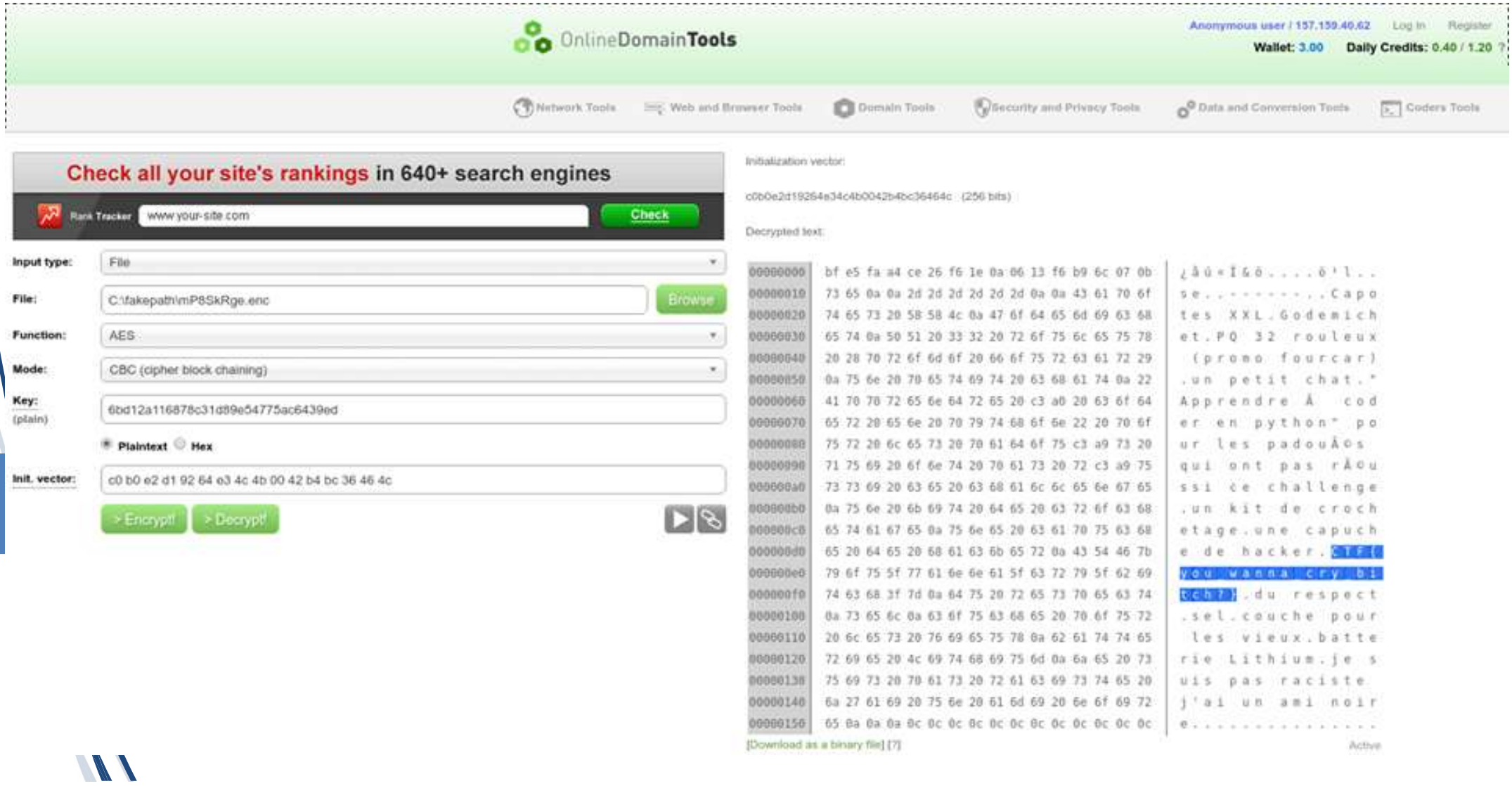
Forensic

Android Malware

```java
package com.android.wannacry;

import java.security.NoSuchAlgorithmException;
import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class WannaCry {
    private String SecretKey = "6bd12a116878c31d89e54775ac6439ed";
    private Cipher cipher;
    private String iv = "248951a7ab4bf545";
    private IvParameterSpec ivspec;
    private SecretKeySpec keyspec;

    public WannaCry() {
        try {
            this.ivspec = new IvParameterSpec(this.iv.getBytes());
            this.keyspec = new SecretKeySpec(this.SecretKey.getBytes(),
            this.cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        } catch (NoSuchPaddingException e2) {
            e2.printStackTrace();
        }
    }

    public byte[] encrypt(String text) throws Exception {
```

Application2Cryptage.apk, fichier /com/android/wannacry/WannaCry.java

Forensic Android Malware



CONCLUSION

http://www.france-ioi.org/

https://www.newbiecontest.org/

http://overthewire.org/wargames/

https://www.root-me.org/