Abstract of CTF event (codefest iit-BHU)

Sunday, February 28, 2016 8:30 PM

Web Basics

First of all I open the source code and go to script part there I saw that there are term window.location(" ") and directory was written in between the " " marks. I had to open that page on url by directory transversal attack .

Hey,, now You will get the flag :)

Flag: 2e746ccb4882d3a7ae21

Launch codes

I used tool called autospy to get open img file and get the flag.txt .

Flag :3f5a57462f77faa850439989bf773df9

Login details

After analyzing the file clearly I find word "BM" then I convert the file in .bmp and then on opening the image I got the flag

Flag : paint:rules

Looking Back in Life

First of all I tried to search on wiki about loopback then I thought that 127.0.0.1 is flag but soon, I find that ip6 localhost is ::1 ..and that was the flag

Flag: ::1

Guess Me

After open the source of image in editor I find text as "I like chocolate", so I guessed the flag.. Chocolate

Flag: chocolate

BrightEdge Needs Help

Convert the file format in .zip and then open the file and I find key.txt.

Flag: It_WaS_CF'16

Weak Authentication

I find cookie named access and then edit the cookie

As True

Then I accessed the page and get the key :)

Flag: d55f03db07878b51b

Ghost Recon

Open the pcap in wireshark , there was a lot of tcp stream and there i got flag in one of tcp stream in plain text

Flag: vim_is_the_best

whiteout

Use the Stagsolve Tool and then I get The flag

Flag: forensicsis_fun

USB modem

I was searching on google then I found it..

I received a pcap file containing USB Request Blocks (URB s) with no other information. A quick look at the exchanged frames with Wireshark revealed that most of the data was sent to the host from a specific device (26.3, HID device from “bInterfaceClass”, keyboard from “bInterfaceProtocol” from the official documentation) on an interrupt endpoint.

The first idea was of course: is the key typed on the keyboard? Every interrupt packet from the 26.3 device was carrying a keycode, and all these packets had the same URB id: 0xffff88003b7d8fc0. Exploring packets structure made it easy to localize these keycodes: the offset 0x42 of these interrupt packets. I just had to script keycodes extracting using a correspondance table, then!

I created a Python script using the dpkt library to parse the pcap file and extract the keycodes:

```python
import binascii
import dpkt
import struct
import sys
# Start the pcap file parsing
f = open(sys.argv[1], 'rb')
pcap = dpkt.pcap.Reader(f)
# Create a partial mapping from keycodes to ASCII chars
keys = {}
keys.update({
    i + 0x4: chr(i + ord('a'))
    for i in range(26)
})
keys.update({
    i + 0x1e: chr(i + ord('1'))
    for i in range(9)
})
keys[0x27] = '0'
```

```
keys.update({
    0x28: '\n',
    0x2c: ' ',
    0x2d: '-',
0x2e: '+',
    0x2f: '[',
    0x30: ']',
})
# Then iterate over each USB frame
for ts, buf in pcap:
    # We are interested only in packets that has the expected URB id, and
    # packets carrying keycodes embed exactly 8 bytes.
    urb_id = ''.join(reversed(buf[:8]))
    if binascii.hexlify(urb_id) != 'ffff88003b7d8fc0':
        continue
    data_length, = struct.unpack('<I', buf[0x24:0x28])
    if data_length != 8:
        continue
    key_code = ord(buf[0x42])
    if not key_code:
        continue
    sys.stdout.write(keys[key_code])




```

The output of this script was the following “keyboard stream”:

```
rxterm -geometry 12x1+0+0
echo k
rxterm -geometry 12x1+75+0
echo e
rxterm -geometry 12x1+150+0
echo y
rxterm -geometry 12x1+225+0
echo [
rxterm -geometry 12x1+300+0
echo c
rxterm -geometry 12x1+375+0
echo 4
rxterm -geometry 12x1+450+0
echo 8
rxterm -geometry 12x1+525+0
echo b
rxterm -geometry 12x1+600+0
echo a
rxterm -geometry 12x1+675+0
echo 9
rxterm -geometry 12x1+0+40
echo 9
rxterm -geometry 12x1+75+40
echo 3
rxterm -geometry 12x1+150+40
echo d
rxterm -geometry 12x1+225+40
echo 3
rxterm -geometry 12x1+300+40
echo 5
rxterm -geometry 12x1+450+40
echo c
rxterm -geometry 12x1+375+40
echo 3
rxterm -geometry 12x1+525+40
echo a
rxterm -geometry 12x1+600+40
echo ]
```

Alright, the indented result should be to display the key re-ordering first the characters with terminal positions. I then had just to format a script to actually open multiple terms in the same time at the right place and containing the associated character:

```
python2 extract_keyboard.py dongle.pcap |
    sed 's/rxterm \(.*\)/xterm \1 -e "\\/g' |
    sed 's/echo \(.*\)/echo -n \1; read" \&/g' > display_key.sh
```

And finally, running the display_key.sh script gave us the key: key[c48ba993d353ca]

cryptograph

Use the basic conversion as ascii,hex,decimal,and oct,

Convert given binary into decimal and then convert all

Decimal into ascii.

indentation matters

When I was opening the page then I was getting blank page but after sometime I find that this is not blank page it is containing many whitespaces and then I run it on ideone and I get the flag ..really it was very interesting Problem .

Easy RSA

After a lot of futile search I came to know about weiner attack on rsa

https://github.com/pablocelayes/rsa-wiener-attack

I got some idea with this link and do the step which is given in it and got private exponent ,I use openssl rsa utilities to decypt the given file and got the flag

looks can be deceiving

After watching the image I came to know first one is gnu and second is Dennis ritchie and third one is bhu ..

So I guessed that passphrase is gnuunixbhu

And then use the tool steghide use fallowing command

$ steghide info filename

Then enter the passphrase and get a embedded flag.txt file

$ steghide extract -sf filename

Then open the file flag.txt and get flag.

crack me

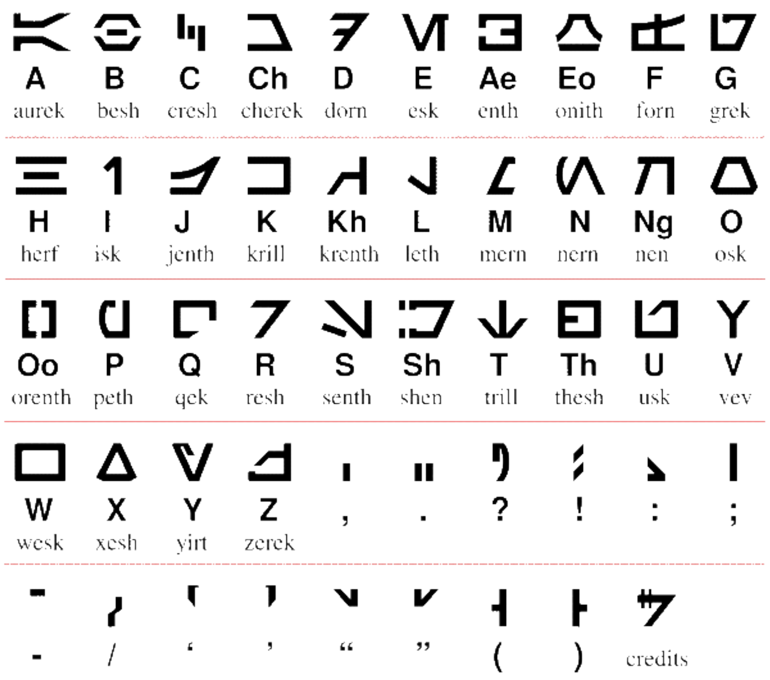
I used the tool snowman as well as Ida pro to sort this problem out. I tried for hours to understand the assembly , figure out some switch case in the code taken out by the snowman and after doing a lot of attempt and making it possible to get password ,after consuming a lots of hour I got this password as well as flag.

Fight club

A mp3 file was given .So first I open this on an editor and saw all the string and after analyzing the all the strings I found the very valuable flag :)

Back to the Future 1

This is the aurebesh language ..i search on google then I find this image



A aurek B besh C cresh Ch cherek D dorn E esk Ae enth Eo onith F forn G grek
H herf I isk J jenth K krill Kh krenth L leth M mern N nern Ng nen O osk
Oo orenth P peth Q qek R resh S senth Sh shen T trill Th thesh U usk V vev
W wesk X xesh Y yirt Z zerek , . ? ! : ;
- / ‘ ’ “ ” ( ) credits

Then I get the flag by decoding the given text using this image..

Flag : CodefestRessurects

Old School

A encrypted message was given .As according to problem name I guessed that it can be old (basic)encryption so convert the symbols of periodic table to decimal and decimal to ascii and I will get the flag

Flag : CODEFeSTCAPTUrEthEFLaG16

Don't trust the user submitted files

Actually you can upload any file that ffmpeg can read and ends with `.gif and ffmpeg will happily convert it for you. ffmpeg probes the files for certain magic values to detect the filetype.`

The first thing I noticed was that they use a custom compiled version of ffmpeg:

```
FFMPEG = "ffmpeg-2.8.2-64bit-static/ffmpeg"
```

So I noticed the `concat demuxer at first. It takes a textfile as input. This file contains script like instructions for concatenating multiple files: e.g.`

```
ffconcat version 1.0
file '/home/temp/key.txt'
```

Unfortunately the concat demuxer doesn't allow 'unsafe' files in the input file

The applehttp or hls demuxer for HTTP Live Streaming playlists. I searched for an example of this file format and copy pastedthis example from apple. Maybe ffmpeg can also open file:// URLs. So I just gave it a try, with the following content for akash.gif:

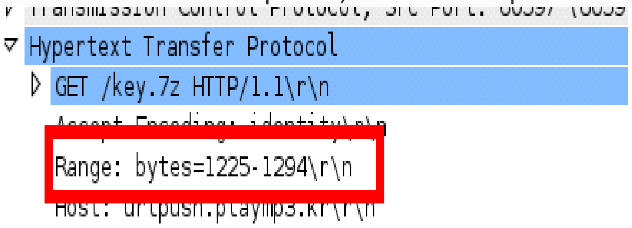
```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:1
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
file:///home/temp/key.txt
#EXT-X-ENDLIST
```

```
Flag: 2e81333e94ec763
```

Key Lost in the Wires

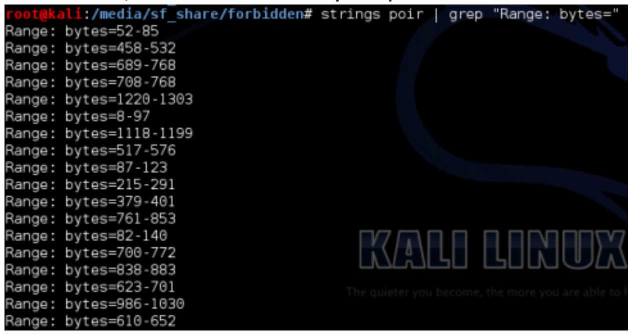
Same as above Is was searching something on google then I found this ..

This challenge provides us a file named poir. It is a pcap-ng file. Wireshark shows us more than 10k packages, mostly HTTP traffic. These packages are from the transfer of a file named “key.7z”. But this is not only one transfer. This file was send in 500 different pieces, which overlaps and in wrong order.



```
▽ Hypertext Transfer Protocol
  ▷ GET /key.7z HTTP/1.1\r\n
    Accept-Encoding: identity\r\n
    Range: bytes=1225-1294\r\n
    Host: urlpush.playmp3.kr\r\n
```

I use the “Export Objects”-Option from Wireshark to save all 500 pieces. For the next step i have to think about how I can get all files in the right order. I could extract the “Range: bytes=”-strings of the poir-file and save those to a new file. If I do this, I can sort the key.7z pieces based on the order of the range strings. Ok, lets do this.



```
root@kali:/media/sf_share/forbidden# strings poir | grep "Range: bytes="
Range: bytes=52-85
Range: bytes=458-532
Range: bytes=689-768
Range: bytes=708-768
Range: bytes=1220-1303
Range: bytes=8-97
Range: bytes=1118-1199
Range: bytes=517-576
Range: bytes=87-123
Range: bytes=215-291
Range: bytes=379-401
Range: bytes=761-853
Range: bytes=82-140
Range: bytes=700-772
Range: bytes=838-883
Range: bytes=623-701
Range: bytes=986-1030
Range: bytes=610-652
```

I write a little python-script which sorts the files and combines them to one file in the one step.

```
#!/usr/bin/env python
f=open("sort.txt","r")
target=open("res.7z","wb")
c=0
d={}
for l in f:
s= l.split("=")[1]
ss=s.split("-")[0]
d[int(ss)]=c
c+=1

for key,val in d.items():
name="key(%d).7z" % val
```

```
print key,val
target.seek(key)
with open(name,"rb") as src:
buf=src.read(1)
while buf:
target.write(buf)
buf=src.read(1)
target.close()
f.close()
```

Now I can extract the files of this archive. There is only one file named “key.png”. It is a completely white picture.

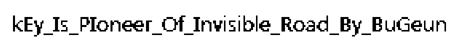
I take a close look and see that not all pixels are completely white. I increase the visibility of these pixels.

```
#!/usr/bin/env python
import Image

image = Image.open('key.png')
new = Image.new('RGB',(image.size))
for x in range(image.size[0]):
for y in range(image.size[1]):
pixel = image.getpixel((x,y))
if pixel == (255,255,255):
new.putpixel((x,y),pixel)
else:
new.putpixel((x,y),(0,0,0))

new.save('key2.png','PNG')
```

The result shows a png with the key.


kEy_Is_PIoneer_Of_Invisible_Road_By_BuGeun

Then I tried Pioneer_Of_Invisible_Road_By_BuGeun

That was my key..