

Lesson 0: What is CTF and who is flagbot?

flagbot (CTF@VIS)

20. January 2024



Table of Contents

Introduction to CTFs
The Different Categories

About Us

Practical Example

What to do?

What is CTF?

- ▶ Stands for Capture the Flag
- ▶ Sport consisting of solving hacking challenges
- ▶ Usually done in a team
- ▶ Flag as proof of solving a challenge
 - ▶ Example flag: `flagbot{sh0uld_h4ve_g0ne_f0r_the_he4d}`
- ▶ Challenges grouped into five major categories: pwn, reversing, web, crypto and misc



pwn challenges

- Usually a small application
- Get remote code execution by finding a vulnerability
- Often need to use a disassembler / decompiler

```
__unwind ( // _m_personality_v0
push r13
push r14
lea rsi, _m_55A6AB74953 ; "\0*****\0*****\0*****\0"
push r13
push r12
lea rdi, _RST4cout@GLIBCXX_3.4
push rbp
push rbp
sub rbp, 140h
lea r12, [rbp+arg_10]
mov [rsp+arg_18], 0
mov [rsp+arg_20], 0
mov [rsp+anonymouse_0], 0
mov [rsp+anonymouse_1], 0
lea rax, [r12+10h]
mov [rsp+anonymouse_2], 0
mov [rsp+arg_38], rax

try {
call _RST1@std::char_traits<char>::basic_ostream_of_885_PwC ; std::operator<<std::char_traits<char>>(std::basic_ostream<char, std::
lea rdi, _RST4cout@GLIBCXX_3.4
mov edx, 1Ah
call _RST8@std::ostream_insert<std::char_traits<char>@std::basic_ostream_of_885_PwC>_1 ; std::_ostream_insert<char, std::char_traits<char>
lea rdi, _RST4cout@GLIBCXX_3.4
mov edx, 1Ah ; 'A'
call _RST10@std::ostream_insert<std::char_traits<char>@std::basic_ostream_of_885_PwC>_1 ; std::_ostream_insert<char, std::char_traits<char>
lea rdi, _RST2lose7nsEnti ; "> Enclose the entire regular expression..."

void *v39; // [rsp+1D8h] [rbp+60h]
int64 v40; // [rsp+1E0h] [rbp+68h]
Char v41; // [rsp+1E8h] [rbp+70h]
_int128 v42; // [rsp+1F8h] [rbp+80h]
Char v43; // [rsp+208h] [rbp+90h]
NFA v44; // [rsp+218h] [rbp+a0h]
NFA nfa; // [rsp+248h] [rbp+d0h]
string v46; // [rsp+278h] [rbp+100h]

v33 = 0LL;
v34 = 0LL;
v35 = 0LL;
v37 = 0LL;
v38 = 0;
v36 = &v38;
std::operator<<std::char_traits<char>>(&std::cout, "\n*****\t*****\n", envp);
std::_ostream_insert<char, std::char_traits<char>>(&std::cout, "\nREGEX FORMAT : \n", 17LL);
std::_ostream_insert<char, std::char_traits<char>>(&std::cout,
"> Enclose every 'concatenation' and 'or' section by parentheses \n",
65LL);
std::_ostream_insert<char, std::char_traits<char>>(&std::cout,
"> Enclose the entire regular expression with parentheses \n",
58LL);
std::operator<<std::char_traits<char>>(&std::cout, "> Use square brackets as a shortcut to 'or' ranges.\n\n", v33);
std::operator<<std::char_traits<char>>(&std::cout, "For example : (a(b|c)[a-c]*) \n", v44);
while ( 1 )

vagrant@ubuntu-bionic:~/CTF/google/regex$ ./regex

*****

REGEX FORMAT :
> Enclose every 'concatenation' and 'or' section by parentheses
> Enclose the entire regular expression with parentheses
> Use square brackets as a shortcut to 'or' ranges.

For example : (a(b|c)[a-c]*)

Enter a regular expression in the above mentioned format, or QUIT to quit.

(a*)
Your regex was saved as regex #0.
Enter REGEX to create another regex, or TEST to test your regexes.
TEST
Which regex do you want to test? (zero-indexed)
0
Enter a string to match against the regex, or QUIT to quit and enter more regexes.
```


reversing challenges

- ▶ Very similar to pwn challenges
- ▶ Figure out, what the program **exactly** does
- ▶ Examples are:
 - ▶ Key generator
 - ▶ Heavily obfuscated binary

```
text:080482E7      public master_loop
text:080482E7      master_loop:
text:080482E7      mov     esp, off_83FD250 ; DATA XREF: .data:sa_loopio
text:080482E7      ; start0
text:080482ED      mov     eax, toggle_execution
text:080482F2      mov     eax, sel_on[eax*4]
text:080482F9      mov     dword ptr [eax], 1
text:080482FF      mov     toggle_execution, 0
text:08048309      mov     eax, sesp
text:0804830E      mov     edx, 4
text:08048313      mov     alu_x, eax
text:08048318      mov     alu_y, edx
text:0804831E      mov     ecx, 0
text:08048323      mov     alu_c, 0
text:08048328      mov     ax, word ptr alu_x
text:08048332      mov     cx, word ptr alu_y
text:08048338      mov     edx, alu_add16[eax*4]
text:0804833F      mov     edx, [edx+ecx*4]
text:08048346      mov     cx, word ptr alu_c+2
text:08048350      mov     edx, alu_add16[edx*4]
text:08048357      mov     edx, [edx+ecx*4]
text:0804835A      mov     word ptr alu_s, dx
text:08048361      mov     alu_c, edx
text:08048367      mov     ax, word ptr alu_x+2
text:0804836D      mov     cx, word ptr alu_y+2
text:08048374      mov     edx, alu_add16[eax*4]
text:0804837B      mov     edx, [edx+ecx*4]
text:0804837E      mov     cx, word ptr alu_c+2
text:08048385      mov     edx, alu_add16[edx*4]
text:0804838C      mov     edx, [edx+ecx*4]
text:0804838F      mov     word ptr alu_s+2, dx
text:08048396      mov     alu_c, edx
text:0804839C      mov     eax, alu_s
text:080483A1      mov     eax, eax
text:080483A3      mov     stack_temp, eax
text:080483A8      mov     eax, Offset off_83FD250
text:080483AD      mov     edx, on
text:080483B3      mov     data_p, eax
text:080483B8      mov     eax, sel_data[edx*4]
text:080483BF      mov     edx, off_83FD250
text:080483C5      mov     edx, [edx-200068h]
text:080483CB      mov     [eax], edx
text:080483CD      mov     eax, off_83FD250
text:080483D2      mov     edx, on
text:080483D8      mov     data_p, eax
text:080483DD      mov     eax, sel_data[edx*4]
text:080483E4      mov     edx, stack_temp
text:080483EA      mov     [eax], edx
text:080483EC      mov     eax, sesp
text:080483F1      mov     eax, [eax]
text:080483F3      mov     eax, eax
text:080483F5      mov     stack_temp, eax
text:080483FA      mov     eax, Offset off_83FD250
text:080483FF      mov     edx, on
text:08048405      mov     data_p, eax
text:0804840A      mov     eax, sel_data[edx*4]
text:08048411      mov     edx, off_83FD250
text:08048417      mov     edx, [edx-200068h]
text:0804841D      mov     [eax], edx
```


web challenges

- ▶ Focuses on web services, languages and frameworks
- ▶ Often a “blackbox”
- ▶ Could even have to exploit language bugs
- ▶ Examples are:
 - ▶ SQL Injection
 - ▶ XSS
 - ▶ Badly implemented authentication



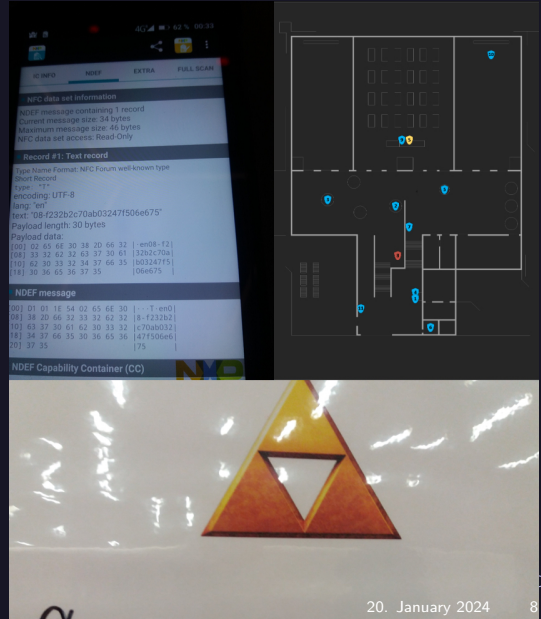
crypto challenges

- ▶ Short for cryptography
- ▶ Find weaknesses in specific implementation of existing cryptography protocol
- ▶ Analyze published research for specific cryptography protocol
- ▶ Examples are:
 - ▶ Using 3 as the exponent in RSA
 - ▶ Not initializing the IV



misc challenges

- ▶ No real focus
- ▶ Often includes steganography and some cryptography
- ▶ Examples are:
 - ▶ Hidden data inside an image
 - ▶ Scavenger hunt
 - ▶ Reading unknown symbols



Who is flagbot?

- ▶ We are a VIS committee
- ▶ Meet every Monday for Training
- ▶ Participate (sometimes) in cool CTF events over the Weekends
- ▶ Currently ranked decently in Switzerland



Who is organizers

- ▶ Joint team between polyglots (EPFL), cr0wn (UK) and secret.club
- ▶ Team up together for larger events
- ▶ Last year ranked #6 worldwide, 2022 we were #1
- ▶ Multiple big wins, such as best European team at DEFCON and #1 at Tencent CTF 2021



Some Impressions



DEFCON Finals in Las Vegas

Some Impressions



Google CTF Hackcelers8



Some Impressions



Google CTF Hackceler8



Some Impressions



SECCON Finals



Some Impressions



Meetup for big events



Some Impressions



Meetup for big events



- ▶ Subscribe at <https://lists.vis.ethz.ch/postorius/lists/ctf.lists.vis.ethz.ch/>
- ▶ We will send E-Mails there, whenever we participate in a CTF event
- ▶ Can be used to ask questions, but please keep them **spoiler free!**
- ▶ In general used for announcements



Some Practical Example - Hackceler8

- ▶ Hackceler8 == Google CTF Finale
- ▶ Special type of MISC challenge, mostly game-hacking
- ▶ We had to "reverse engineer" a game and find ways to cheat or exploit bugs



- ▶ Beginner CTF
- ▶ Ideal way to introduce people to CTF
- ▶ Register here: play.picoctf.org

