“Gen Z P'Batching Game” Official Writeup

27th January 2026

Prepared By: {CYNX}`Team

Title: Gen Z P'Batching Game

Description: I tried so hard to break this game, but I could not because of the ununderstandable strings. idk what is happening with this generation!! Maybe I’m just generation X, OR my RE skills are already getting rusty and I couldn’t even read the instructions :(

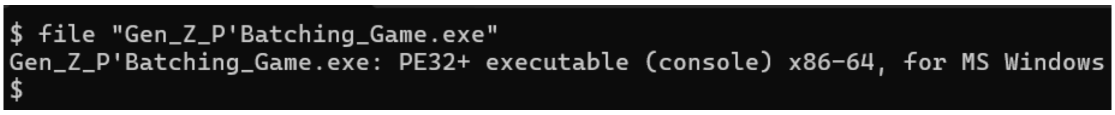
Flag: flag{P3tch!nG_s_Kinds_C00oO0OOL_IG}

Difficulty: Easy

Writeup classification: Official

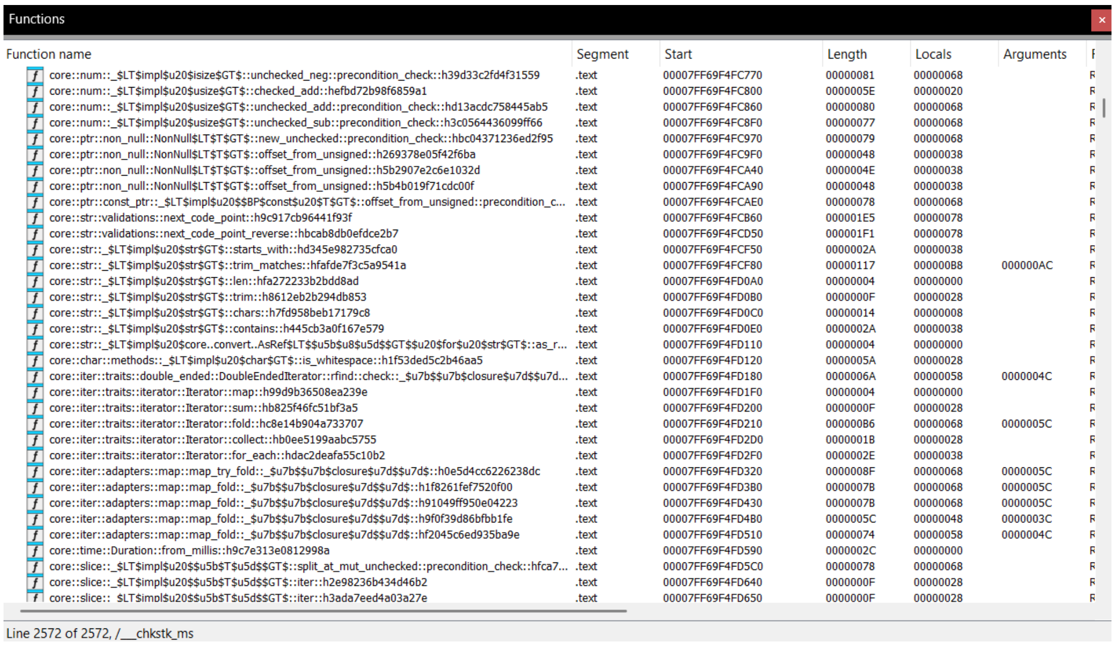
Note: Although the challenge itself is straightforward and primarily involves simple binary patching, its difficulty is slightly increased by the use of the Rust programming language. Rust binaries typically include a large number of packages and functions, which can overwhelm the player during analysis. This makes it harder to quickly identify the relevant function to target, and it also complicates copying and pasting code snippets for AI assisted analysis. As a result, the challenge feels more confusing than difficult, however, the overall difficulty classification remains **Easy**

Solve:



```
$ file "Gen_Z_P'Batching_Game.exe"
Gen_Z_P'Batching_Game.exe: PE32+ executable (console) x86-64, for MS Windows
$
```

The provided file is a PE32+ executable for the Microsoft Windows

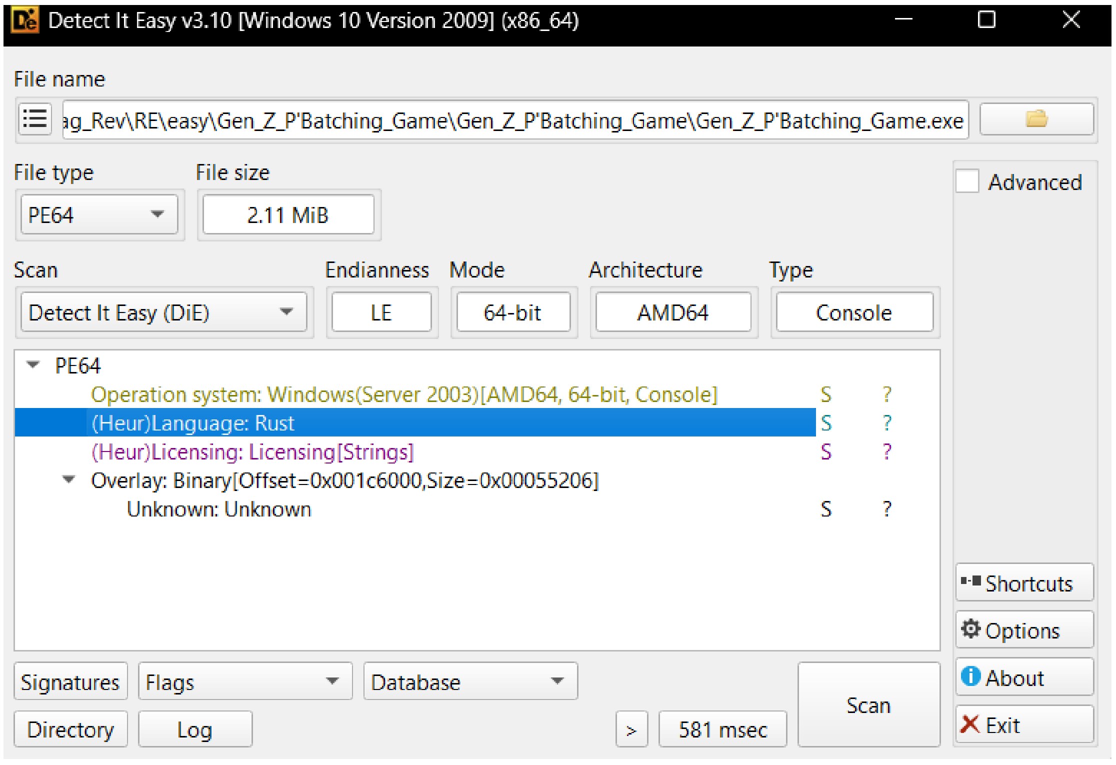


Functions

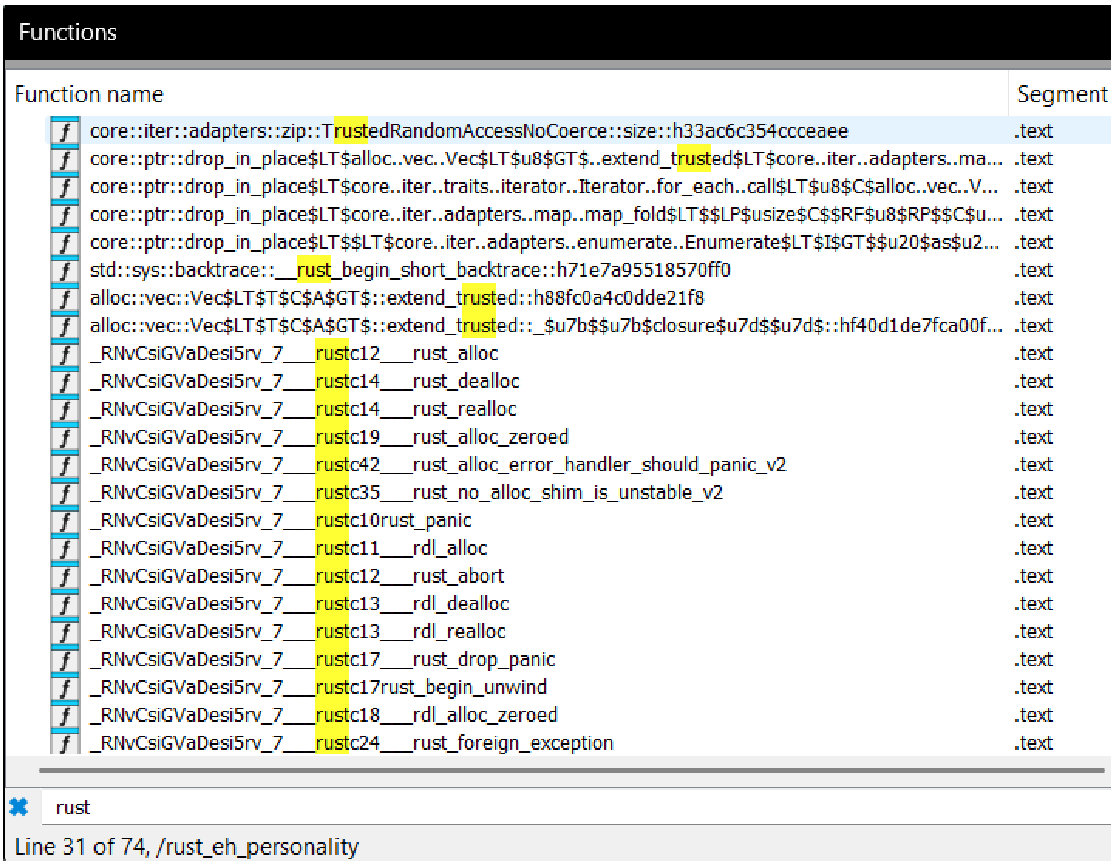
Function name	Segment	Start	Length	Locals	Arguments	F
core::num::_$LT$impl$u20$isize$GT$::unchecked_neg::precondition_check::h39d33c2fd4f31559	.text	00007FF69F4FC770	00000081	00000068		R
core::num::_$LT$impl$u20$usize$GT$::checked_add::hefbd72b98f6859a1	.text	00007FF69F4FC800	0000005E	00000020		R
core::num::_$LT$impl$u20$usize$GT$::unchecked_add::precondition_check::hd13acdc758445ab5	.text	00007FF69F4FC860	00000080	00000068		R
core::num::_$LT$impl$u20$usize$GT$::unchecked_sub::precondition_check::h3c0564436099ff66	.text	00007FF69F4FC8F0	00000077	00000068		R
core::ptr::non_null::NonNull$LT$T$GT$::new_unchecked::precondition_check::hbc04371236ed2f95	.text	00007FF69F4FC970	00000079	00000068		R
core::ptr::non_null::NonNull$LT$T$GT$::offset_from_unsigned::h269378e05f42f6ba	.text	00007FF69F4FC9F0	00000048	00000038		R
core::ptr::non_null::NonNull$LT$T$GT$::offset_from_unsigned::h5b2907e2c6e1032d	.text	00007FF69F4FCA40	0000004E	00000038		R
core::ptr::non_null::NonNull$LT$T$GT$::offset_from_unsigned::h5b4b019f71cdc00f	.text	00007FF69F4FCA90	00000048	00000038		R
core::ptr::const_ptr::_LTimpl$u20$$BP$const$u20TGT$::offset_from_unsigned::precondition_c...	.text	00007FF69F4FCAE0	00000078	00000068		R
core::str::validations::next_code_point::h9c917cb96441f93f	.text	00007FF69F4FCB60	000001E5	00000078		R
core::str::validations::next_code_point_reverse::hbcab8db0efdce2b7	.text	00007FF69F4FCD50	000001F1	00000078		R
core::str::_$LT$impl$u20$str$GT$::starts_with::hd345e982735cfca0	.text	00007FF69F4FCF50	0000002A	00000038		R
core::str::_$LT$impl$u20$str$GT$::trim_matches::hfafde7f3c5a9541a	.text	00007FF69F4FCF80	00000117	000000B8	000000AC	R
core::str::_$LT$impl$u20$str$GT$::len::hfa272233b2bdd8ad	.text	00007FF69F4FD0A0	00000004	00000000		R
core::str::_$LT$impl$u20$str$GT$::trim::h8612eb2b294db853	.text	00007FF69F4FD0B0	0000000F	00000028		R
core::str::_$LT$impl$u20$str$GT$::chars::h7fd958beb17179c8	.text	00007FF69F4FD0C0	00000014	00000008		R
core::str::_$LT$impl$u20$str$GT$::contains::h445cb3a0f167e579	.text	00007FF69F4FD0E0	0000002A	00000038		R
core::str::_LTimpl$u20$core..convert..AsRef$LT$$u5b$u8$u5d$$GT$$u20foru20strGT$::as_r...	.text	00007FF69F4FD110	00000004	00000000		R
core::char::methods::_$LT$impl$u20$char$GT$::is_whitespace::h1f53ded5c2b46aa5	.text	00007FF69F4FD120	0000005A	00000028		R
core::iter::traits::double_ended::DoubleEndedIterator::rfind::check::_$u7b$$u7b$closure$u7d$$u7d...	.text	00007FF69F4FD180	0000006A	00000058	0000004C	R
core::iter::traits::iterator::Iterator::map::h99d9b36508ea239e	.text	00007FF69F4FD1F0	00000004	00000000		R
core::iter::traits::iterator::Iterator::sum::hb825f46fc51bf3a5	.text	00007FF69F4FD200	0000000F	00000028		R
core::iter::traits::iterator::Iterator::fold::hc8e14b904a733707	.text	00007FF69F4FD210	000000B6	00000068	0000005C	R
core::iter::traits::iterator::Iterator::collect::hb0ee5199aabc5755	.text	00007FF69F4FD2D0	0000001B	00000028		R
core::iter::traits::iterator::Iterator::for_each::hdac2deafa55c10b2	.text	00007FF69F4FD2F0	0000002E	00000038		R
core::iter::adapters::map::map_try_fold::_$u7b$$u7b$closure$u7d$$u7d$::h0e5d4cc6226238dc	.text	00007FF69F4FD320	0000008F	00000068	0000005C	R
core::iter::adapters::map::map_fold::_$u7b$$u7b$closure$u7d$$u7d$::h1f8261fef7520f00	.text	00007FF69F4FD3B0	0000007B	00000068	0000005C	R
core::iter::adapters::map::map_fold::_$u7b$$u7b$closure$u7d$$u7d$::h91049ff950e04223	.text	00007FF69F4FD430	0000007B	00000068	0000005C	R
core::iter::adapters::map::map_fold::_$u7b$$u7b$closure$u7d$$u7d$::h9f0f39d86bfbb1fe	.text	00007FF69F4FD4B0	0000005C	00000048	0000003C	R
core::iter::adapters::map::map_fold::_$u7b$$u7b$closure$u7d$$u7d$::hf2045c6ed935ba9e	.text	00007FF69F4FD510	00000074	00000058	0000004C	R
core::time::Duration::from_millis::h9c7e313e0812998a	.text	00007FF69F4FD590	0000002C	00000000		R
core::slice::_LTimpl$u20$$u5bTu5d$$GT$::split_at_mut_unchecked::precondition_check::hfca7...	.text	00007FF69F4FD5C0	00000078	00000068		R
core::slice::_LTimpl$u20$$u5bTu5d$$GT$::iter::h2e98236b434d46b2	.text	00007FF69F4FD640	0000000F	00000028		R
core::slice::_LTimpl$u20$$u5bTu5d$$GT$::iter::h3ada7eed4a03a27e	.text	00007FF69F4FD650	0000000F	00000028		R

Line 2572 of 2572, /__chkstk_ms

After opening the executable in IDA and navigating to the function list, it becomes clear that the binary contains a large number of functions. Without a clear methodology to narrow down the scope, analyzing these functions can be time-consuming and inefficient.



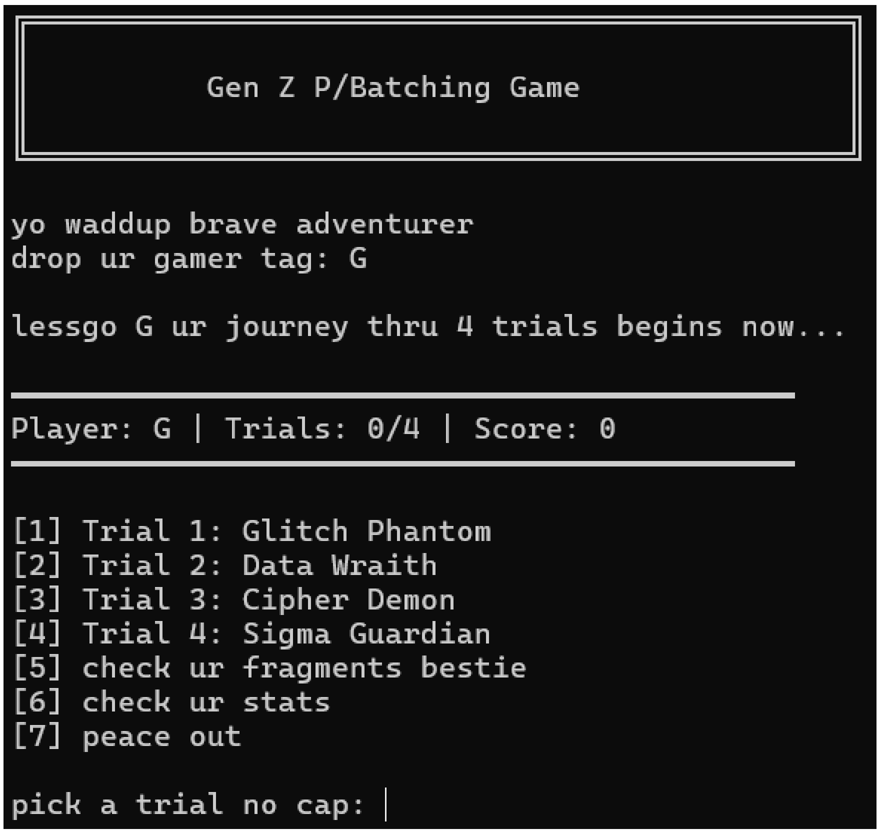
```
Detect It Easy v3.10 [Windows 10 Version 2009] (x86_64)
File name: ag_Rev\RE\easy\Gen_Z_P'Batching_Game\Gen_Z_P'Batching_Game\Gen_Z_P'Batching_Game.exe
File type: PE64    File size: 2.11 MiB
Scan: Detect It Easy (DiE)   Endianness: LE   Mode: 64-bit   Architecture: AMD64   Type: Console

PE64
    Operation system: Windows(Server 2003)[AMD64, 64-bit, Console]      S   ?
    (Heur)Language: Rust                                                S   ?
    (Heur)Licensing: Licensing[Strings]                                 S   ?
    Overlay: Binary[Offset=0x001c6000,Size=0x00055206]
        Unknown: Unknown                                                S   ?

581 msec
```



Functions

```
Function name                                                                                     Segment
core::iter::adapters::zip::TrustedRandomAccessNoCoerce::size::h33ac6c354ccceaee                   .text
core::ptr::drop_in_place$LT$alloc..vec..Vec$LT$u8$GT$..extend_trusted$LT$core..iter..adapters..ma... .text
core::ptr::drop_in_place$LT$core..iter..traits..iterator..Iterator..for_each..call$LT$u8$C$alloc..vec..V... .text
core::ptr::drop_in_place$LT$core..iter..adapters..map..map_fold$LT$$LP$usize$C$$RF$u8$RP$$C$u... .text
core::ptr::drop_in_place$LT$$LT$core..iter..adapters..enumerate..Enumerate$LT$I$GT$$u20$as$u2... .text
std::sys::backtrace::__rust_begin_short_backtrace::h71e7a95518570ff0                              .text
alloc::vec::Vec$LT$T$C$A$GT$::extend_trusted::h88fc0a4c0dde21f8                                   .text
alloc::vec::Vec$LT$T$C$A$GT$::extend_trusted::_$u7b$$u7b$closure$u7d$$u7d$::hf40d1de7fca00f...   .text
_RNvCsiGVaDesi5rv_7___rustc12___rust_alloc                                                        .text
_RNvCsiGVaDesi5rv_7___rustc14___rust_dealloc                                                      .text
_RNvCsiGVaDesi5rv_7___rustc14___rust_realloc                                                      .text
_RNvCsiGVaDesi5rv_7___rustc19___rust_alloc_zeroed                                                 .text
_RNvCsiGVaDesi5rv_7___rustc42___rust_alloc_error_handler_should_panic_v2                          .text
_RNvCsiGVaDesi5rv_7___rustc35___rust_no_alloc_shim_is_unstable_v2                                 .text
_RNvCsiGVaDesi5rv_7___rustc10rust_panic                                                           .text
_RNvCsiGVaDesi5rv_7___rustc11___rdl_alloc                                                         .text
_RNvCsiGVaDesi5rv_7___rustc12___rust_abort                                                        .text
_RNvCsiGVaDesi5rv_7___rustc13___rdl_dealloc                                                       .text
_RNvCsiGVaDesi5rv_7___rustc13___rdl_realloc                                                       .text
_RNvCsiGVaDesi5rv_7___rustc17___rust_drop_panic                                                   .text
_RNvCsiGVaDesi5rv_7___rustc17rust_begin_unwind                                                    .text
_RNvCsiGVaDesi5rv_7___rustc18___rdl_alloc_zeroed                                                  .text
_RNvCsiGVaDesi5rv_7___rustc24___rust_foreign_exception                                            .text

rust
Line 31 of 74, /rust_eh_personality
```

Use the **Detect It Easy** tool to identify the programming language used to develop the executable file, and the language used is **Rust**



```
Gen Z P/Batching Game

yo waddup brave adventurer
drop ur gamer tag: G

lessgo G ur journey thru 4 trials begins now...

Player: G | Trials: 0/4 | Score: 0

[1] Trial 1: Glitch Phantom
[2] Trial 2: Data Wraith
[3] Trial 3: Cipher Demon
[4] Trial 4: Sigma Guardian
[5] check ur fragments bestie
[6] check ur stats
[7] peace out

pick a trial no cap:
```

As a logical first step in analyzing the executable, it is important to understand how the program runs and how its internal logic works. Based on this analysis, we observe that the game consists of four trials:

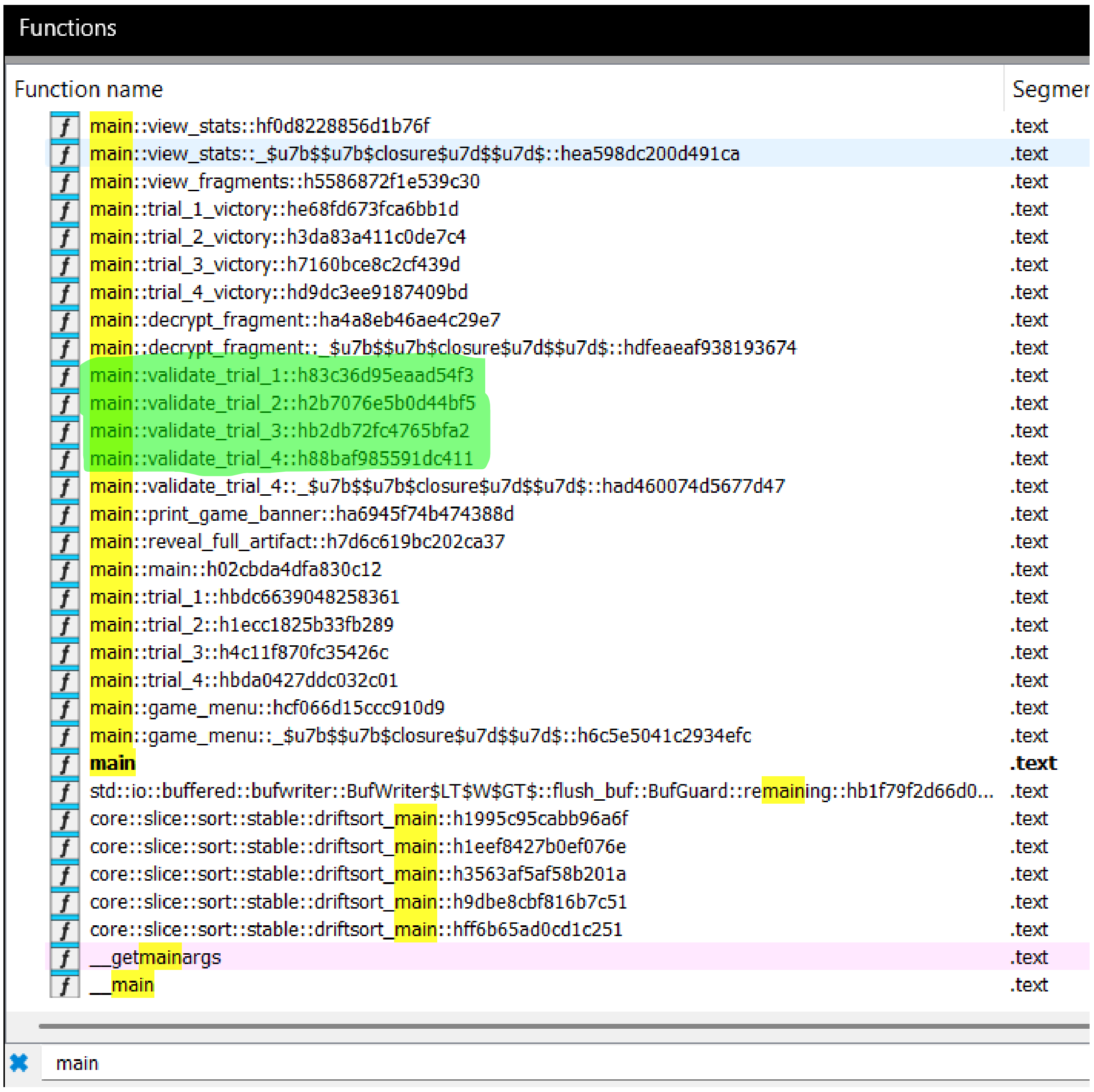
- **Trial 1 (Glitch Phantom):** Prompts the user for an 8-character code. Regardless of the input, the validation always fails.
- **Trial 2 (Data Wraith):** Remains locked until Trial 1 is completed. It asks for a code starting with "HACK", but all inputs are rejected.
- **Trial 3 (Cipher Demon):** Unlocked only after completing Trial 2. It requires a code containing "X0X", yet it always fails.
- **Trial 4 (Sigma Guardian):** Unlocked after Trial 3. It asks for a code whose characters sum to 500, but validation still fails.

Key Observation

No valid input exists, even inputs that clearly satisfy the stated conditions are rejected (validation error) This indicates that the challenge is not about discovering the correct passwords, but rather about **patching the binary to bypass the validation checks**.

Goal

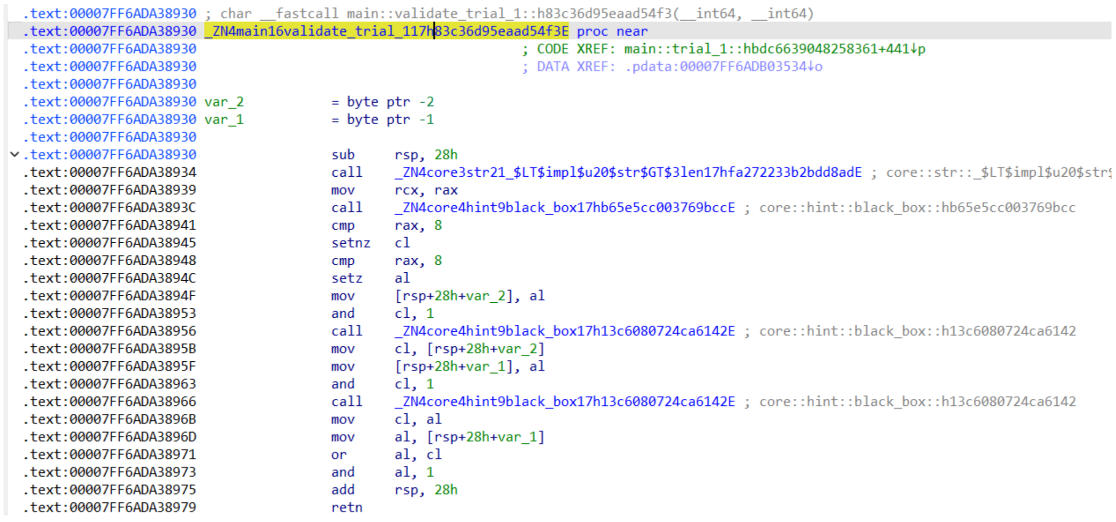
The objective is to patch the executable so that all four trials pass successfully, revealing flag fragments that combine to form the final flag, the challenge name **"P/Batching"** serves as a hint that binary patching is required.



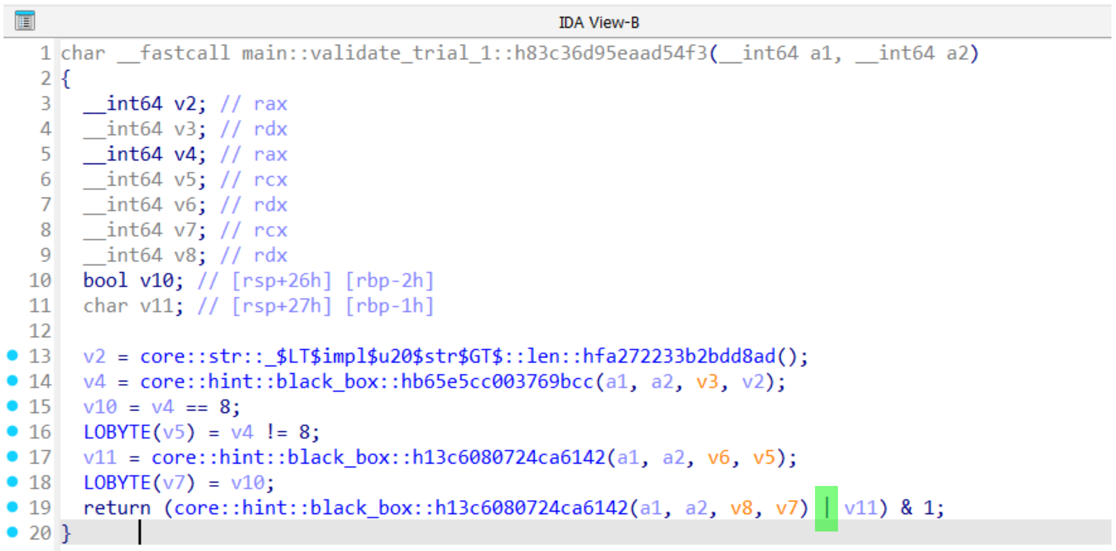
```
Functions

Function name                                                                          Segmer
main::view_stats::hf0d8228856d1b76f                                                    .text
main::view_stats::_$u7b$$u7b$closure$u7d$$u7d$::hea598dc200d491ca                      .text
main::view_fragments::h5586872f1e539c30                                                .text
main::trial_1_victory::he68fd673fca6bb1d                                               .text
main::trial_2_victory::h3da83a411c0de7c4                                               .text
main::trial_3_victory::h7160bce8c2cf439d                                               .text
main::trial_4_victory::hd9dc3ee9187409bd                                               .text
main::decrypt_fragment::ha4a8eb46ae4c29e7                                              .text
main::decrypt_fragment::_$u7b$$u7b$closure$u7d$$u7d$::hdfeaeaf938193674                .text
main::validate_trial_1::h83c36d95eaad54f3                                              .text
main::validate_trial_2::h2b7076e5b0d44bf5                                              .text
main::validate_trial_3::hb2db72fc4765bfa2                                              .text
main::validate_trial_4::h88baf985591dc411                                              .text
main::validate_trial_4::_$u7b$$u7b$closure$u7d$$u7d$::had460074d5677d47                .text
main::print_game_banner::ha6945f74b474388d                                             .text
main::reveal_full_artifact::h7d6c619bc202ca37                                          .text
main::main::h02cbda4dfa830c12                                                          .text
main::trial_1::hbdc6639048258361                                                       .text
main::trial_2::h1ecc1825b33fb289                                                       .text
main::trial_3::h4c11f870fc35426c                                                       .text
main::trial_4::hbda0427ddc032c01                                                       .text
main::game_menu::hcf066d15ccc910d9                                                     .text
main::game_menu::_$u7b$$u7b$closure$u7d$$u7d$::h6c5e5041c2934efc                       .text
main                                                                                   .text
std::io::buffered::bufwriter::BufWriter$LT$W$GT$::flush_buf::BufGuard::remaining::hb1f79f2d66d0...  .text
core::slice::sort::stable::driftsort_main::h1995c95cabb96a6f                           .text
core::slice::sort::stable::driftsort_main::h1eef8427b0ef076e                           .text
core::slice::sort::stable::driftsort_main::h3563af5af58b201a                           .text
core::slice::sort::stable::driftsort_main::h9dbe8cbf816b7c51                           .text
core::slice::sort::stable::driftsort_main::hff6b65ad0cd1c251                           .text
__getmainargs                                                                          .text
__main                                                                                 .text

main
```

From the list of functions, we can identify the validation logic for each trial, implemented as four separate functions

Trial 1:



```
.text:00007FF6ADA38930 ; char __fastcall main::validate_trial_1::h83c36d95eaad54f3(__int64, __int64)
.text:00007FF6ADA38930 _ZN4main16validate_trial_117h83c36d95eaad54f3E proc near
.text:00007FF6ADA38930                                         ; CODE XREF: main::trial_1::hbdc6639048258361+441↓p
.text:00007FF6ADA38930                                         ; DATA XREF: .pdata:00007FF6ADB03534↓o
.text:00007FF6ADA38930
.text:00007FF6ADA38930 var_2           = byte ptr -2
.text:00007FF6ADA38930 var_1           = byte ptr -1
.text:00007FF6ADA38930
.text:00007FF6ADA38930                 sub     rsp, 28h
.text:00007FF6ADA38934                 call    _ZN4core3str21_$LT$impl$u20$str$GT$3len17hfa272233b2bdd8adE ; core::str::_$LT$impl$u20$str$
.text:00007FF6ADA38939                 mov     rcx, rax
.text:00007FF6ADA3893C                 call    _ZN4core4hint9black_box17hb65e5cc003769bccE ; core::hint::black_box::hb65e5cc003769bcc
.text:00007FF6ADA38941                 cmp     rax, 8
.text:00007FF6ADA38945                 setnz   cl
.text:00007FF6ADA38948                 cmp     rax, 8
.text:00007FF6ADA3894C                 setz    al
.text:00007FF6ADA3894F                 mov     [rsp+28h+var_2], al
.text:00007FF6ADA38953                 and     cl, 1
.text:00007FF6ADA38956                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF6ADA3895B                 mov     cl, [rsp+28h+var_2]
.text:00007FF6ADA3895F                 mov     [rsp+28h+var_1], al
.text:00007FF6ADA38963                 and     cl, 1
.text:00007FF6ADA38966                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF6ADA3896B                 mov     cl, al
.text:00007FF6ADA3896D                 mov     al, [rsp+28h+var_1]
.text:00007FF6ADA38971                 or      al, cl
.text:00007FF6ADA38973                 and     al, 1
.text:00007FF6ADA38975                 add     rsp, 28h
.text:00007FF6ADA38979                 retn
```



```
char __fastcall main::validate_trial_1::h83c36d95eaad54f3(__int64 a1, __int64 a2)
{
  __int64 v2; // rax
  __int64 v3; // rdx
  __int64 v4; // rax
  __int64 v5; // rcx
  __int64 v6; // rdx
  __int64 v7; // rcx
  __int64 v8; // rdx
  bool v10; // [rsp+26h] [rbp-2h]
  char v11; // [rsp+27h] [rbp-1h]

  v2 = core::str::_$LT$impl$u20$str$GT$::len::hfa272233b2bdd8ad();
  v4 = core::hint::black_box::hb65e5cc003769bcc(a1, a2, v3, v2);
  v10 = v4 == 8;
  LOBYTE(v5) = v4 != 8;
  v11 = core::hint::black_box::h13c6080724ca6142(a1, a2, v6, v5);
  LOBYTE(v7) = v10;
  return (core::hint::black_box::h13c6080724ca6142(a1, a2, v8, v7) | v11) & 1;
}
```

v4 = black_box(code.len());	// Get string length
v10 = v4 == 8;	// right = (length == 8)
LOBYTE(v5) = v4 != 8;	// wrong = (length != 8)
v11 = black_box(wrong);	// Store wrong result
LOBYTE(v7) = v10;	// Prepare right for black_box
return (black_box(right) | v11) & 1;	// ALWAYS returns 1

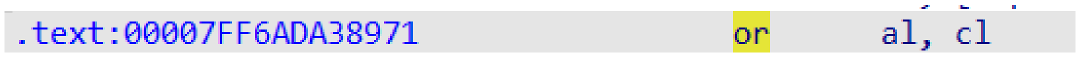
The Problem:

Input Length	(v11) wrong	(v10) right	Wrong/Right	Result
7 chars	1	0	**1**	FAIL
8 chars	0	1	**1**	FAIL
9 chars	1	0	**1**	FAIL

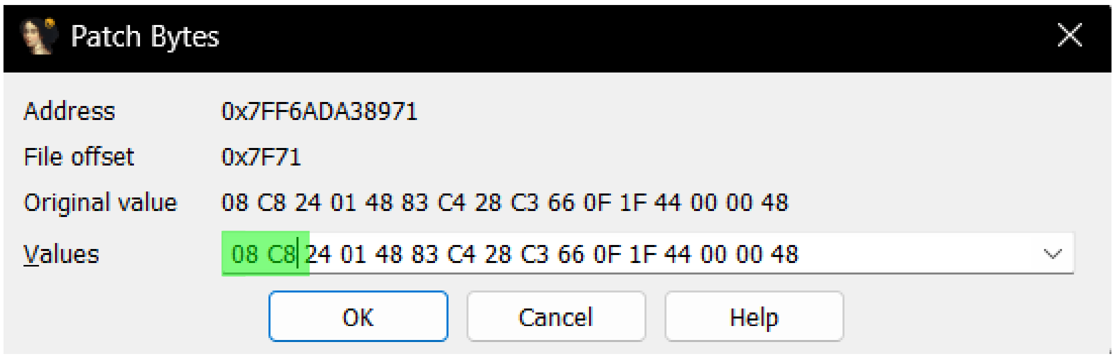
The OR of opposite conditions is ALWAYS true

Patch Point:

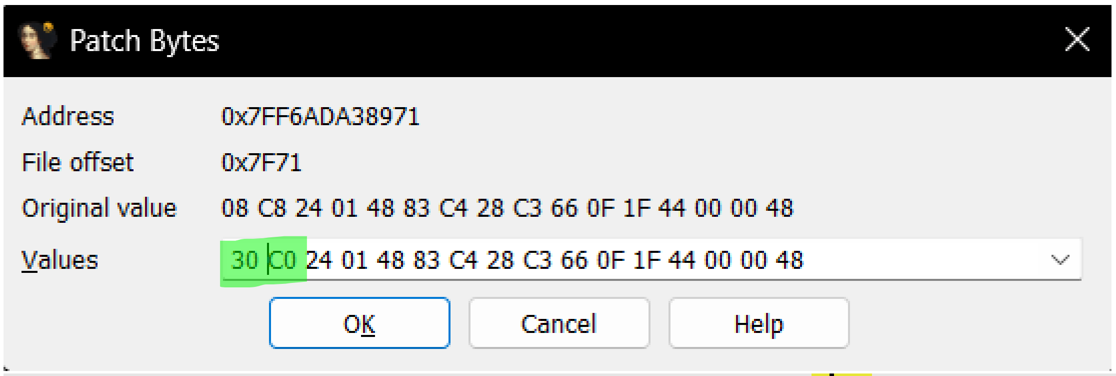
Target: The final | operation in the return statement



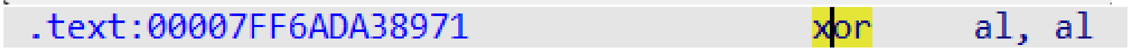
```
.text:00007FF6ADA38971                  or      al, cl
```



```
Address          0x7FF6ADA38971
File offset      0x7F71
Original value   08 C8 24 01 48 83 C4 28 C3 66 0F 1F 44 00 00 48
Values           08 C8 24 01 48 83 C4 28 C3 66 0F 1F 44 00 00 48
```

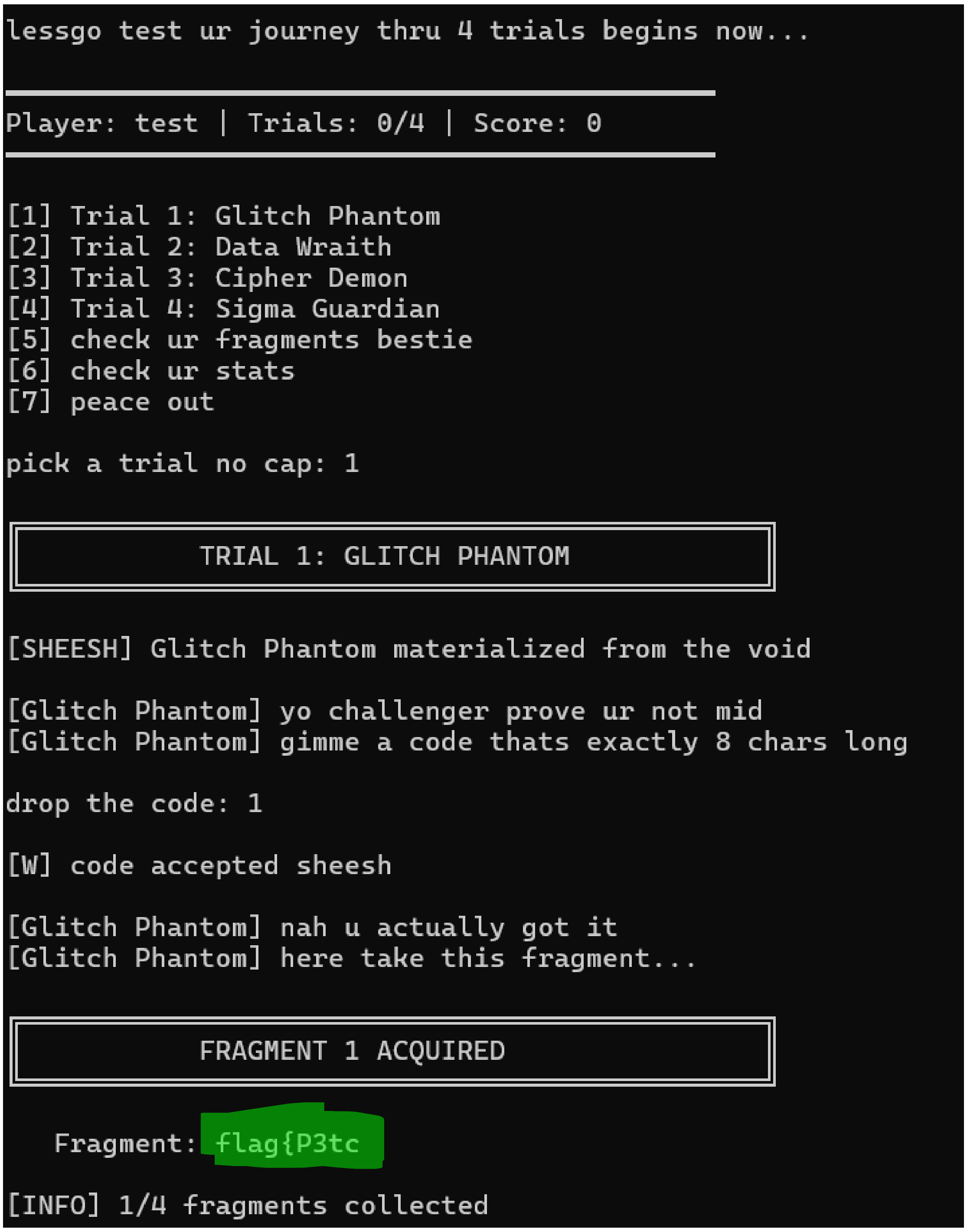


```
Address          0x7FF6ADA38971
File offset      0x7F71
Original value   08 C8 24 01 48 83 C4 28 C3 66 0F 1F 44 00 00 48
Values           30 C0 24 01 48 83 C4 28 C3 66 0F 1F 44 00 00 48
```



```
.text:00007FF6ADA38971                  xor     al, al
```

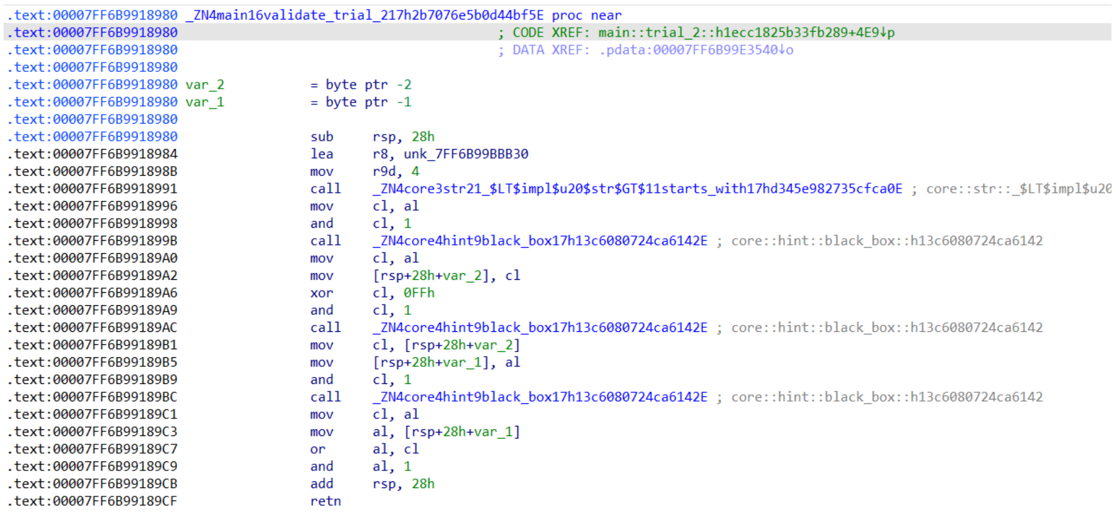
Change the **OR** instruction to **XOR** by patching the bytes from 08 C8 to 30 C0



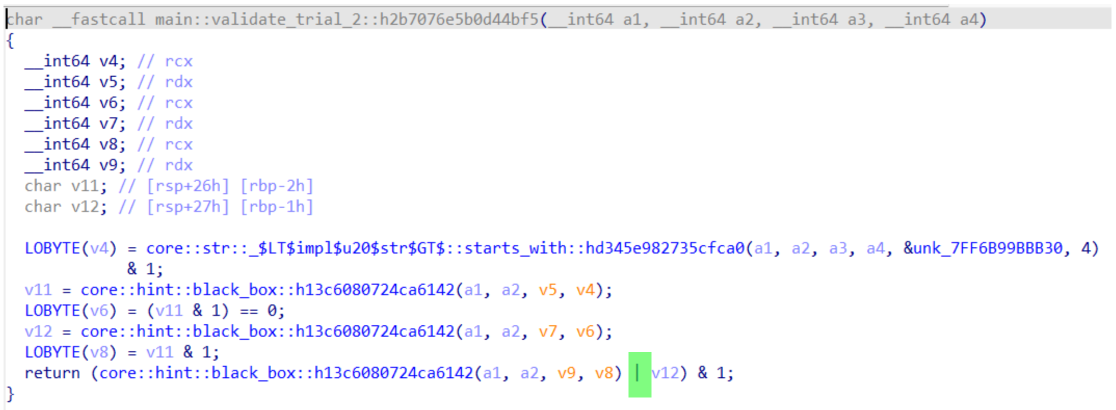
```
lessgo test ur journey thru 4 trials begins now...

════════════════════════════════════════════
Player: test | Trials: 0/4 | Score: 0
════════════════════════════════════════════

[1] Trial 1: Glitch Phantom
[2] Trial 2: Data Wraith
[3] Trial 3: Cipher Demon
[4] Trial 4: Sigma Guardian
[5] check ur fragments bestie
[6] check ur stats
[7] peace out

pick a trial no cap: 1

╔══════════════════════════════════════════════╗
║            TRIAL 1: GLITCH PHANTOM           ║
╚══════════════════════════════════════════════╝

[SHEESH] Glitch Phantom materialized from the void

[Glitch Phantom] yo challenger prove ur not mid
[Glitch Phantom] gimme a code thats exactly 8 chars long

drop the code: 1

[W] code accepted sheesh

[Glitch Phantom] nah u actually got it
[Glitch Phantom] here take this fragment...

╔══════════════════════════════════════════════╗
║            FRAGMENT 1 ACQUIRED               ║
╚══════════════════════════════════════════════╝

   Fragment: flag{P3tc

[INFO] 1/4 fragments collected
```

Run the exe and test the first trial and get the 1st fragment of the flag

Trial 2:



```
.text:00007FF6B9918980 _ZN4main16validate_trial_217h2b7076e5b0d44bf5E proc near
.text:00007FF6B9918980                                         ; CODE XREF: main::trial_2::h1ecc1825b33fb289+4E9↓p
.text:00007FF6B9918980                                         ; DATA XREF: .pdata:00007FF6B99E3540↓o
.text:00007FF6B9918980
.text:00007FF6B9918980 var_2           = byte ptr -2
.text:00007FF6B9918980 var_1           = byte ptr -1
.text:00007FF6B9918980
.text:00007FF6B9918980                 sub     rsp, 28h
.text:00007FF6B9918984                 lea     r8, unk_7FF6B99BBB30
.text:00007FF6B991898B                 mov     r9d, 4
.text:00007FF6B9918991                 call    _ZN4core3str21_$LT$impl$u20$str$GT$11starts_with17hd345e982735cfca0E ; core::str::_$LT$impl$u2
.text:00007FF6B9918996                 mov     cl, al
.text:00007FF6B9918998                 and     cl, 1
.text:00007FF6B991899B                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF6B99189A0                 mov     cl, al
.text:00007FF6B99189A2                 mov     [rsp+28h+var_2], cl
.text:00007FF6B99189A6                 xor     cl, 0FFh
.text:00007FF6B99189A9                 and     cl, 1
.text:00007FF6B99189AC                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF6B99189B1                 mov     cl, [rsp+28h+var_2]
.text:00007FF6B99189B5                 mov     [rsp+28h+var_1], al
.text:00007FF6B99189B9                 and     cl, 1
.text:00007FF6B99189BC                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF6B99189C1                 mov     cl, al
.text:00007FF6B99189C3                 mov     al, [rsp+28h+var_1]
.text:00007FF6B99189C7                 or      al, cl
.text:00007FF6B99189C9                 and     al, 1
.text:00007FF6B99189CB                 add     rsp, 28h
.text:00007FF6B99189CF                 retn
```



```
char __fastcall main::validate_trial_2::h2b7076e5b0d44bf5(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 v4; // rcx
  __int64 v5; // rdx
  __int64 v6; // rcx
  __int64 v7; // rdx
  __int64 v8; // rcx
  __int64 v9; // rdx
  char v11; // [rsp+26h] [rbp-2h]
  char v12; // [rsp+27h] [rbp-1h]

  LOBYTE(v4) = core::str::_$LT$impl$u20$str$GT$::starts_with::hd345e982735cfca0(a1, a2, a3, a4, &unk_7FF6B99BBB30, 4)
             & 1;
  v11 = core::hint::black_box::h13c6080724ca6142(a1, a2, v5, v4);
  LOBYTE(v6) = (v11 & 1) == 0;
  v12 = core::hint::black_box::h13c6080724ca6142(a1, a2, v7, v6);
  LOBYTE(v8) = v11 & 1;
  return (core::hint::black_box::h13c6080724ca6142(a1, a2, v9, v8) | v12) & 1;
}
```

v4 = starts_with(code, "HACK", 4) & 1;	// Check if string starts with "HACK"
v11 = black_box(v4);	// v11 = starts_with result (1 if starts with HACK)
LOBYTE(v6) = (v11 & 1) == 0;	// wrong = !starts_with (1 if DOESN'T start with HACK)
v12 = black_box(wrong);	// Store wrong result
LOBYTE(v8) = v11 & 1;	// right = starts_with (1 if DOES start with HACK)
return (black_box(right) | v12) & 1;	// ALWAYS returns 1

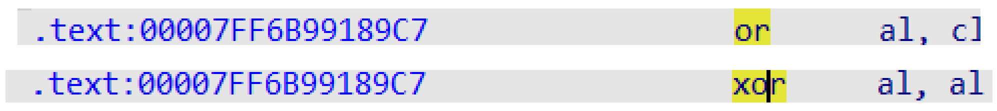
The Problem:

Input	Start with	(v12) wrong	(v8) right	wrong | right	Result
"HACK123"	1	0	1	**1**	FAIL
"hello"	0	1	0	**1**	FAIL
"HACKme"	1	0	1	**1**	FAIL

The function returns starts_with | !starts_with **which is ALWAYS 1**

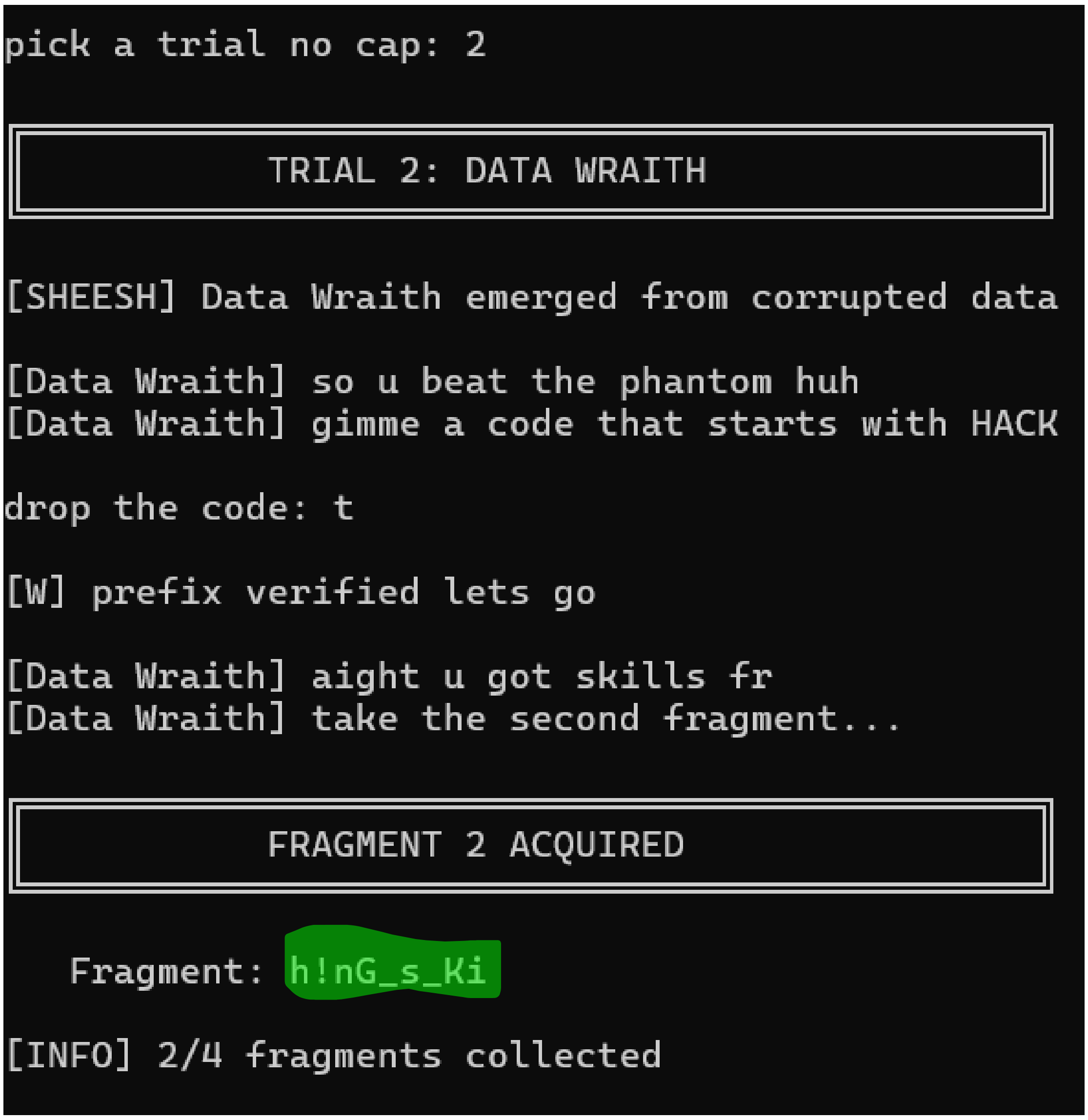
Patch Point:

Target: The final | operation in the return statement



```
.text:00007FF6B99189C7                    or          al, cl
.text:00007FF6B99189C7                    xor         al, al
```

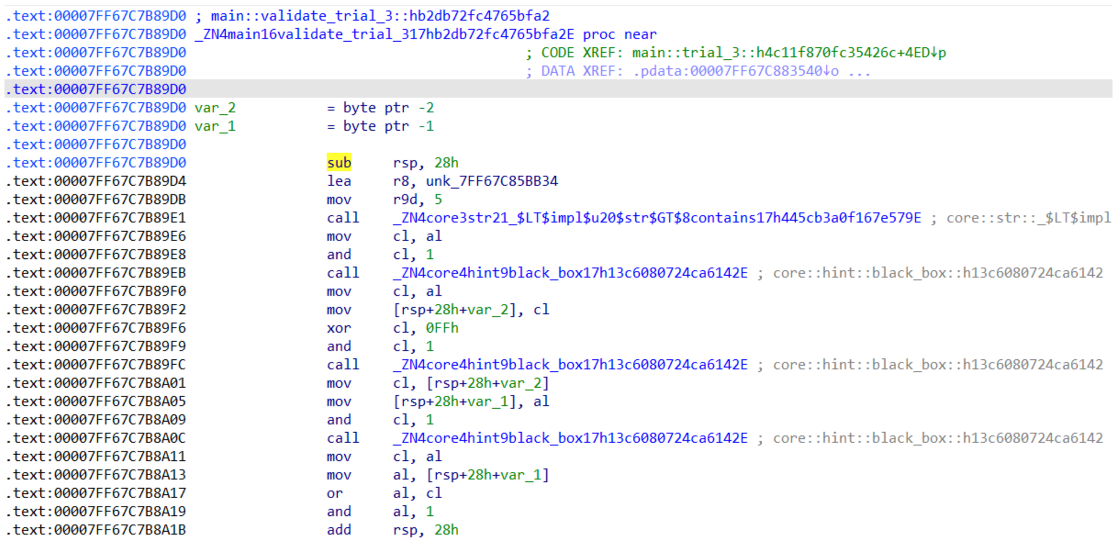
Change the **OR** instruction to **XOR** by patching the bytes from 08 C8 to 30 C0



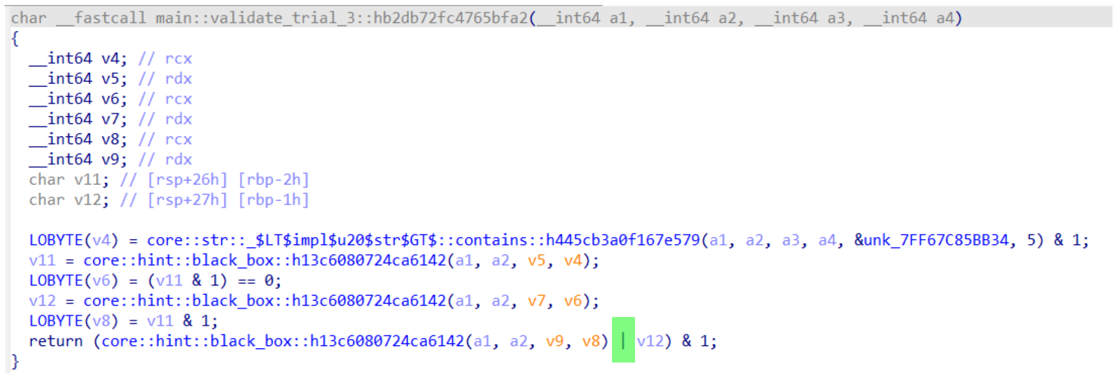
```
pick a trial no cap: 2

╔══════════════════════════════════════════════╗
║            TRIAL 2: DATA WRAITH              ║
╚══════════════════════════════════════════════╝

[SHEESH] Data Wraith emerged from corrupted data

[Data Wraith] so u beat the phantom huh
[Data Wraith] gimme a code that starts with HACK

drop the code: t

[W] prefix verified lets go

[Data Wraith] aight u got skills fr
[Data Wraith] take the second fragment...

╔══════════════════════════════════════════════╗
║            FRAGMENT 2 ACQUIRED               ║
╚══════════════════════════════════════════════╝

   Fragment: h!nG_s_Ki

[INFO] 2/4 fragments collected
```

Run the exe and test the first trial and get the 2nd fragment of the flag

Trial 3:



```
.text:00007FF67C7B89D0 ; main::validate_trial_3::hb2db72fc4765bfa2
.text:00007FF67C7B89D0 _ZN4main16validate_trial_317hb2db72fc4765bfa2E proc near
.text:00007FF67C7B89D0                                         ; CODE XREF: main::trial_3::h4c11f870fc35426c+4ED↓p
.text:00007FF67C7B89D0                                         ; DATA XREF: .pdata:00007FF67C883540↓o ...
.text:00007FF67C7B89D0
.text:00007FF67C7B89D0 var_2           = byte ptr -2
.text:00007FF67C7B89D0 var_1           = byte ptr -1
.text:00007FF67C7B89D0
.text:00007FF67C7B89D0                 sub     rsp, 28h
.text:00007FF67C7B89D4                 lea     r8, unk_7FF67C85BB34
.text:00007FF67C7B89DB                 mov     r9d, 5
.text:00007FF67C7B89E1                 call    _ZN4core3str21_$LT$impl$u20$str$GT$8contains17h445cb3a0f167e579E ; core::str::_$LT$impl
.text:00007FF67C7B89E6                 mov     cl, al
.text:00007FF67C7B89E8                 and     cl, 1
.text:00007FF67C7B89EB                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF67C7B89F0                 mov     cl, al
.text:00007FF67C7B89F2                 mov     [rsp+28h+var_2], cl
.text:00007FF67C7B89F6                 xor     cl, 0FFh
.text:00007FF67C7B89F9                 and     cl, 1
.text:00007FF67C7B89FC                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF67C7B8A01                 mov     cl, [rsp+28h+var_2]
.text:00007FF67C7B8A05                 mov     [rsp+28h+var_1], al
.text:00007FF67C7B8A09                 and     cl, 1
.text:00007FF67C7B8A0C                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF67C7B8A11                 mov     cl, al
.text:00007FF67C7B8A13                 mov     al, [rsp+28h+var_1]
.text:00007FF67C7B8A17                 or      al, cl
.text:00007FF67C7B8A19                 and     al, 1
.text:00007FF67C7B8A1B                 add     rsp, 28h
```



```
char __fastcall main::validate_trial_3::hb2db72fc4765bfa2(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 v4; // rcx
  __int64 v5; // rdx
  __int64 v6; // rcx
  __int64 v7; // rdx
  __int64 v8; // rcx
  __int64 v9; // rdx
  char v11; // [rsp+26h] [rbp-2h]
  char v12; // [rsp+27h] [rbp-1h]

  LOBYTE(v4) = core::str::_$LT$impl$u20$str$GT$::contains::h445cb3a0f167e579(a1, a2, a3, a4, &unk_7FF67C85BB34, 5) & 1;
  v11 = core::hint::black_box::h13c6080724ca6142(a1, a2, v5, v4);
  LOBYTE(v6) = (v11 & 1) == 0;
  v12 = core::hint::black_box::h13c6080724ca6142(a1, a2, v7, v6);
  LOBYTE(v8) = v11 & 1;
  return (core::hint::black_box::h13c6080724ca6142(a1, a2, v9, v8) | v12) & 1;
}
```

v4 = contains(code, "_X0X_", 5) & 1;	// Check if string contains "_X0X_" (5 characters)
v11 = black_box(v4);	// v11 = contains result (1 if contains _X0X_)
LOBYTE(v6) = (v11 & 1) == 0;	// wrong = !contains (1 if DOESN'T contain _X0X_)
v12 = black_box(wrong);	// Store wrong result
LOBYTE(v8) = v11 & 1;	// right = contains (1 if DOES contain _X0X_)
return (black_box(right) | v12) & 1;	// ALWAYS returns 1

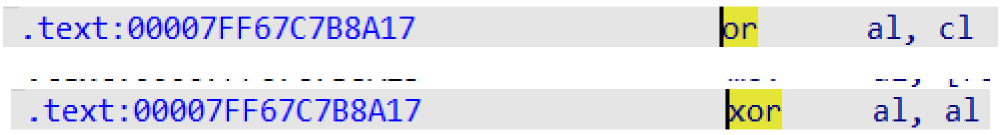
The Problem:

Input	contains	wrong (v12)	right (v8)	wrong | right	Result
"test_X0X_pass"	1	0	1	**1**	FAIL
"hello"	0	1	0	**1**	FAIL
"*X0X*"	1	0	1	**1**	FAIL

The function returns contains | !contains **which is ALWAYS 1**

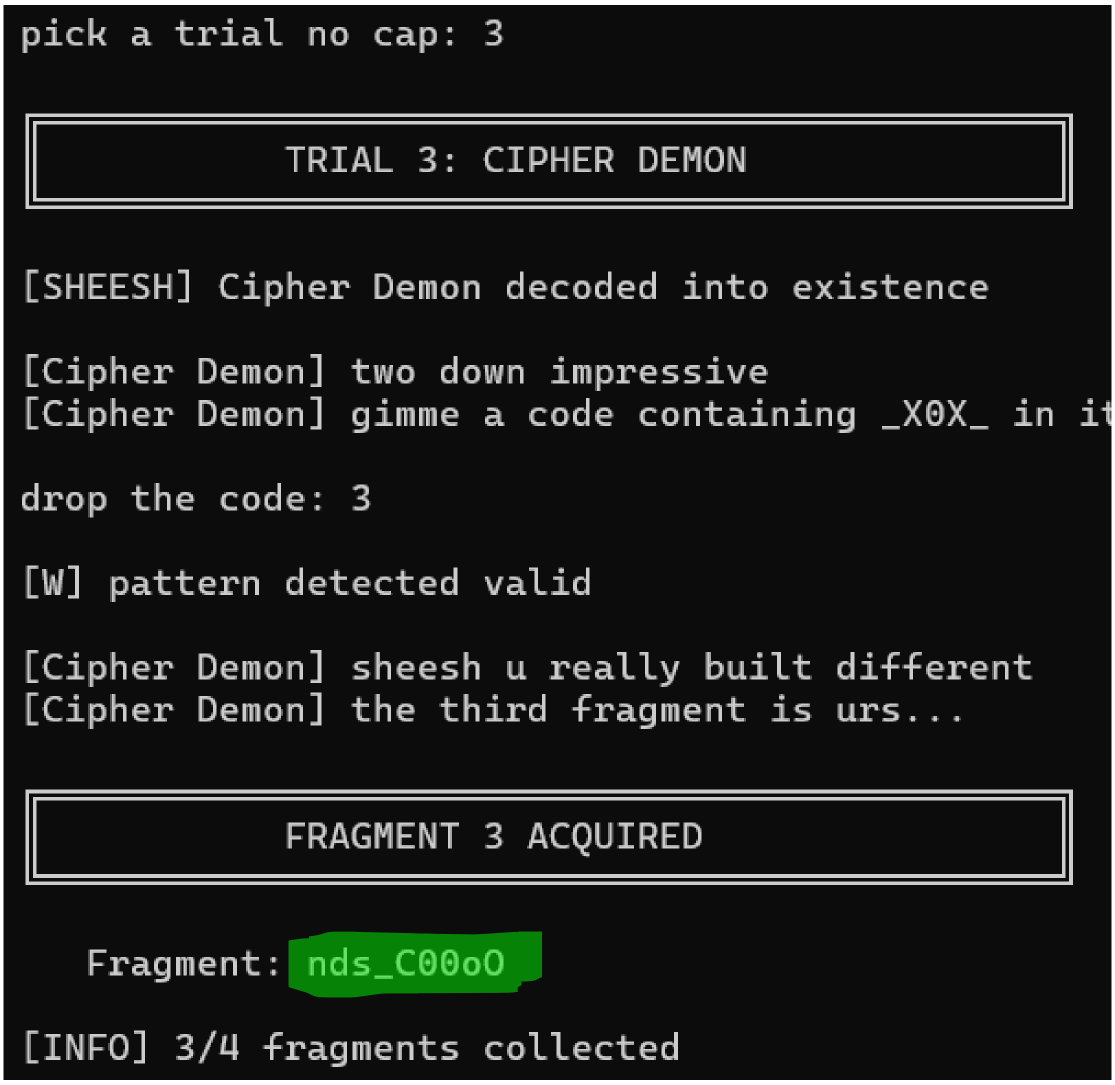
Patch Point:

Target: The final | operation in the return statement



```
.text:00007FF67C7B8A17                 or      al, cl
.text:00007FF67C7B8A17                 xor     al, al
```

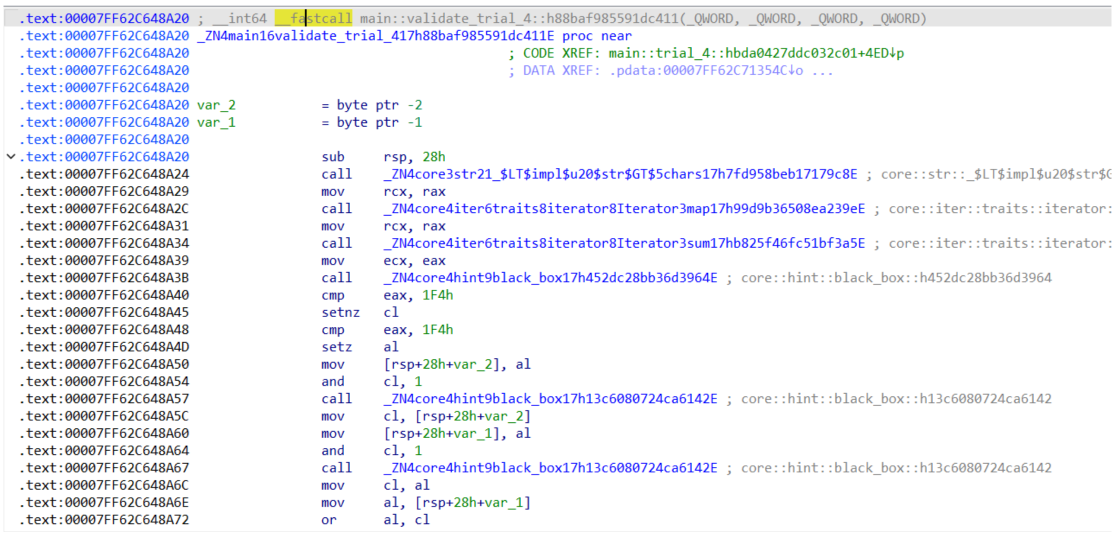
Change the **OR** instruction to **XOR** by patching the bytes from 08 C8 to 30 C0



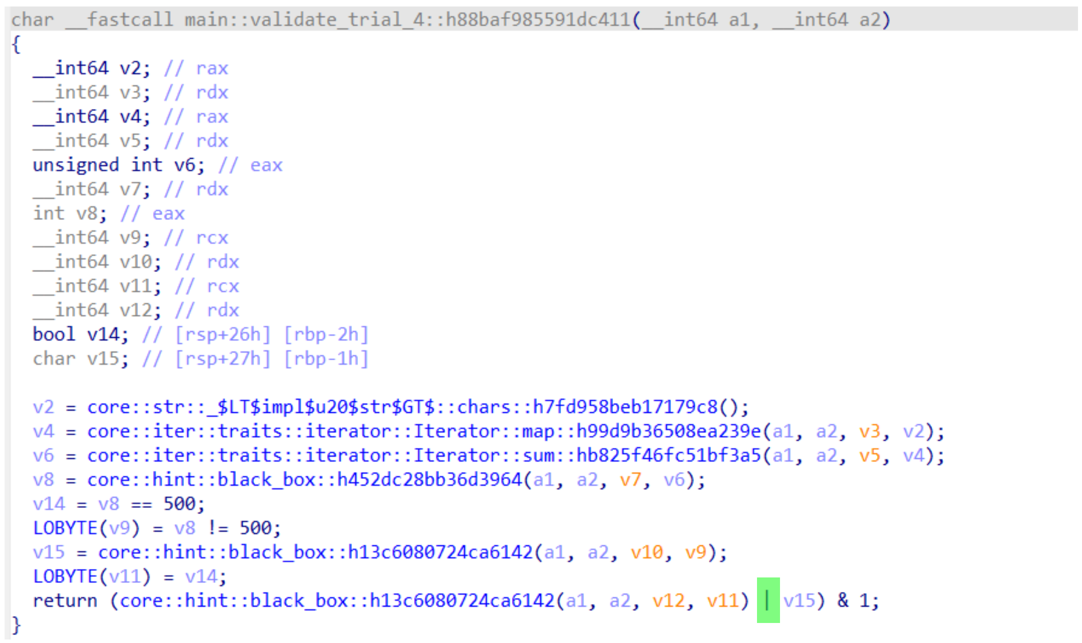
```
pick a trial no cap: 3

╔══════════════════════════════════════════════╗
║           TRIAL 3: CIPHER DEMON              ║
╚══════════════════════════════════════════════╝

[SHEESH] Cipher Demon decoded into existence

[Cipher Demon] two down impressive
[Cipher Demon] gimme a code containing _X0X_ in it

drop the code: 3

[W] pattern detected valid

[Cipher Demon] sheesh u really built different
[Cipher Demon] the third fragment is urs...

╔══════════════════════════════════════════════╗
║           FRAGMENT 3 ACQUIRED                ║
╚══════════════════════════════════════════════╝

   Fragment: nds_C00oO

[INFO] 3/4 fragments collected
```

Run the exe and test the first trial and get the 3rd fragment of the flag

Trial 4:



```
.text:00007FF62C648A20 ; __int64 __fastcall main::validate_trial_4::h88baf985591dc411(_QWORD, _QWORD, _QWORD, _QWORD)
.text:00007FF62C648A20 _ZN4main16validate_trial_417h88baf985591dc411E proc near
.text:00007FF62C648A20                                         ; CODE XREF: main::trial_4::hbda0427ddc032c01+4ED↓p
.text:00007FF62C648A20                                         ; DATA XREF: .pdata:00007FF62C71354C↓o ...
.text:00007FF62C648A20
.text:00007FF62C648A20 var_2           = byte ptr -2
.text:00007FF62C648A20 var_1           = byte ptr -1
.text:00007FF62C648A20
.text:00007FF62C648A20                 sub     rsp, 28h
.text:00007FF62C648A24                 call    _ZN4core3str21_$LT$impl$u20$str$GT$5chars17h7fd958beb17179c8E ; core::str::_$LT$impl$u20$str$G
.text:00007FF62C648A29                 mov     rcx, rax
.text:00007FF62C648A2C                 call    _ZN4core4iter6traits8iterator8Iterator3map17h99d9b36508ea239eE ; core::iter::traits::iterator:
.text:00007FF62C648A31                 mov     rcx, rax
.text:00007FF62C648A34                 call    _ZN4core4iter6traits8iterator8Iterator3sum17hb825f46fc51bf3a5E ; core::iter::traits::iterator:
.text:00007FF62C648A39                 mov     ecx, eax
.text:00007FF62C648A3B                 call    _ZN4core4hint9black_box17h452dc28bb36d3964E ; core::hint::black_box::h452dc28bb36d3964
.text:00007FF62C648A40                 cmp     eax, 1F4h
.text:00007FF62C648A45                 setnz   cl
.text:00007FF62C648A48                 cmp     eax, 1F4h
.text:00007FF62C648A4D                 setz    al
.text:00007FF62C648A50                 mov     [rsp+28h+var_2], al
.text:00007FF62C648A54                 and     cl, 1
.text:00007FF62C648A57                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF62C648A5C                 mov     cl, [rsp+28h+var_2]
.text:00007FF62C648A60                 mov     [rsp+28h+var_1], al
.text:00007FF62C648A64                 and     cl, 1
.text:00007FF62C648A67                 call    _ZN4core4hint9black_box17h13c6080724ca6142E ; core::hint::black_box::h13c6080724ca6142
.text:00007FF62C648A6C                 mov     cl, al
.text:00007FF62C648A6E                 mov     al, [rsp+28h+var_1]
.text:00007FF62C648A72                 or      al, cl
```



```
char __fastcall main::validate_trial_4::h88baf985591dc411(__int64 a1, __int64 a2)
{
  __int64 v2; // rax
  __int64 v3; // rdx
  __int64 v4; // rax
  __int64 v5; // rdx
  unsigned int v6; // eax
  __int64 v7; // rdx
  int v8; // eax
  __int64 v9; // rcx
  __int64 v10; // rdx
  __int64 v11; // rcx
  __int64 v12; // rdx
  bool v14; // [rsp+26h] [rbp-2h]
  char v15; // [rsp+27h] [rbp-1h]

  v2 = core::str::_$LT$impl$u20$str$GT$::chars::h7fd958beb17179c8();
  v4 = core::iter::traits::iterator::Iterator::map::h99d9b36508ea239e(a1, a2, v3, v2);
  v6 = core::iter::traits::iterator::Iterator::sum::hb825f46fc51bf3a5(a1, a2, v5, v4);
  v8 = core::hint::black_box::h452dc28bb36d3964(a1, a2, v7, v6);
  v14 = v8 == 500;
  LOBYTE(v9) = v8 != 500;
  v15 = core::hint::black_box::h13c6080724ca6142(a1, a2, v10, v9);
  LOBYTE(v11) = v14;
  return (core::hint::black_box::h13c6080724ca6142(a1, a2, v12, v11) | v15) & 1;
}
```

v2 = code.chars();	// Get character iterator and sum all ASCII values
v4 = iterator.map(|c| c as u32);	
v6 = iterator.sum();	// Sum of all character ASCII values
v8 = black_box(v6);	// v8 = power level
v14 = v8 == 500;	// right = (power == 500)
LOBYTE(v9) = v8 != 500;	// wrong = (power != 500)
v15 = black_box(wrong);	// Store wrong result
LOBYTE(v11) = v14;	// right = (power == 500)
return (black_box(right) | v15) & 1;	// ALWAYS returns 1

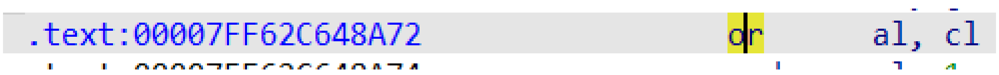
The Problem:

Input	Power Sum	wrong(v15)	right (v14)	wrong | right	Result
"ddddd" (100×5=500)	500	0	1	**1**	FAIL
"AAAA"	260	1	0	**1**	FAIL
"~~~~"	504	1	0	**1**	FAIL

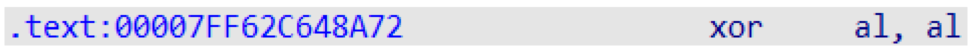
The function returns (power == 500) | (power != 500) **which is ALWAYS 1**

Patch Point:

Target: The final | operation in the return statement

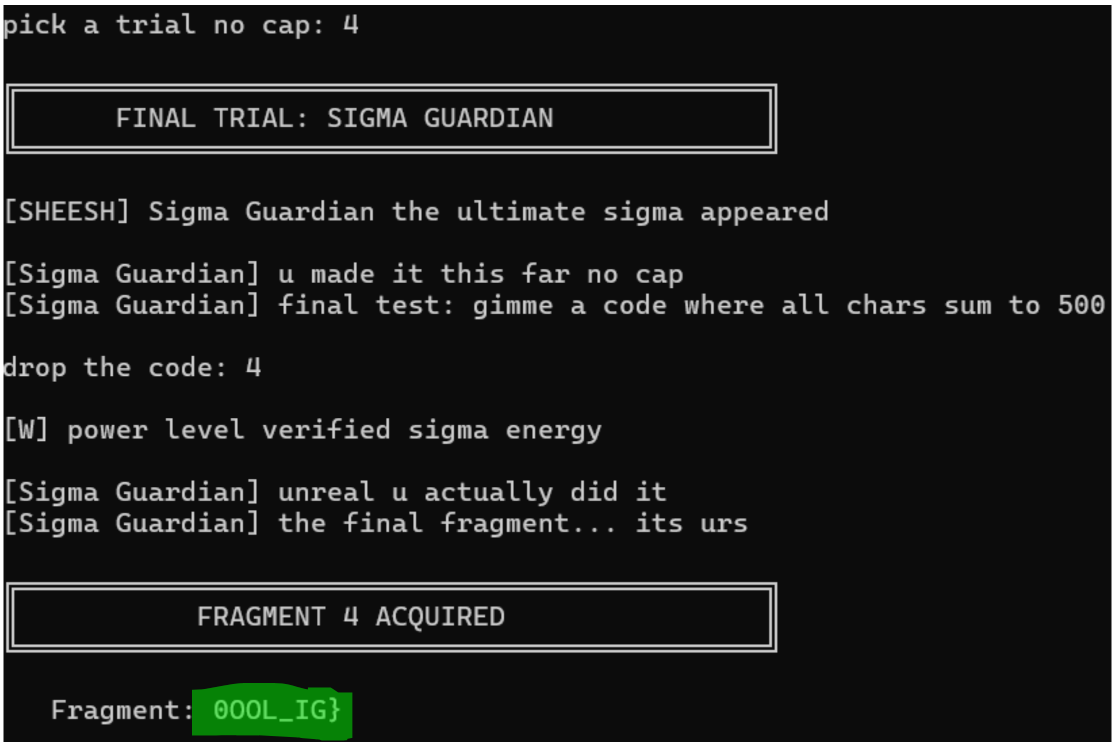


```
.text:00007FF62C648A72                  or      al, cl
```



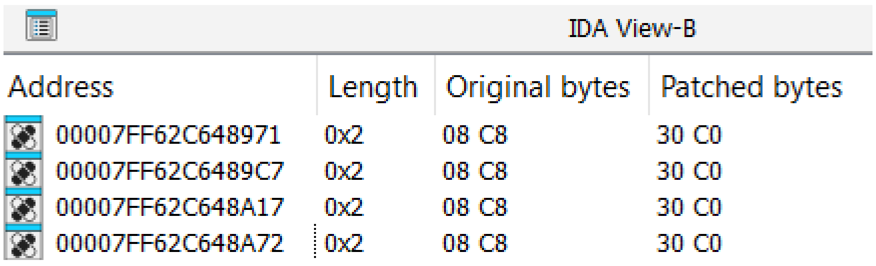
```
.text:00007FF62C648A72                  xor     al, al
```

Change the **OR** instruction to **XOR** by patching the bytes from 08 C8 to 30 C0



```
pick a trial no cap: 4

 ╔════════════════════════════════════════════════╗
 ║       FINAL TRIAL: SIGMA GUARDIAN              ║
 ╚════════════════════════════════════════════════╝

[SHEESH] Sigma Guardian the ultimate sigma appeared

[Sigma Guardian] u made it this far no cap
[Sigma Guardian] final test: gimme a code where all chars sum to 500

drop the code: 4

[W] power level verified sigma energy

[Sigma Guardian] unreal u actually did it
[Sigma Guardian] the final fragment... its urs

 ╔════════════════════════════════════════════════╗
 ║            FRAGMENT 4 ACQUIRED                 ║
 ╚════════════════════════════════════════════════╝

   Fragment: 0OOL_IG}
```

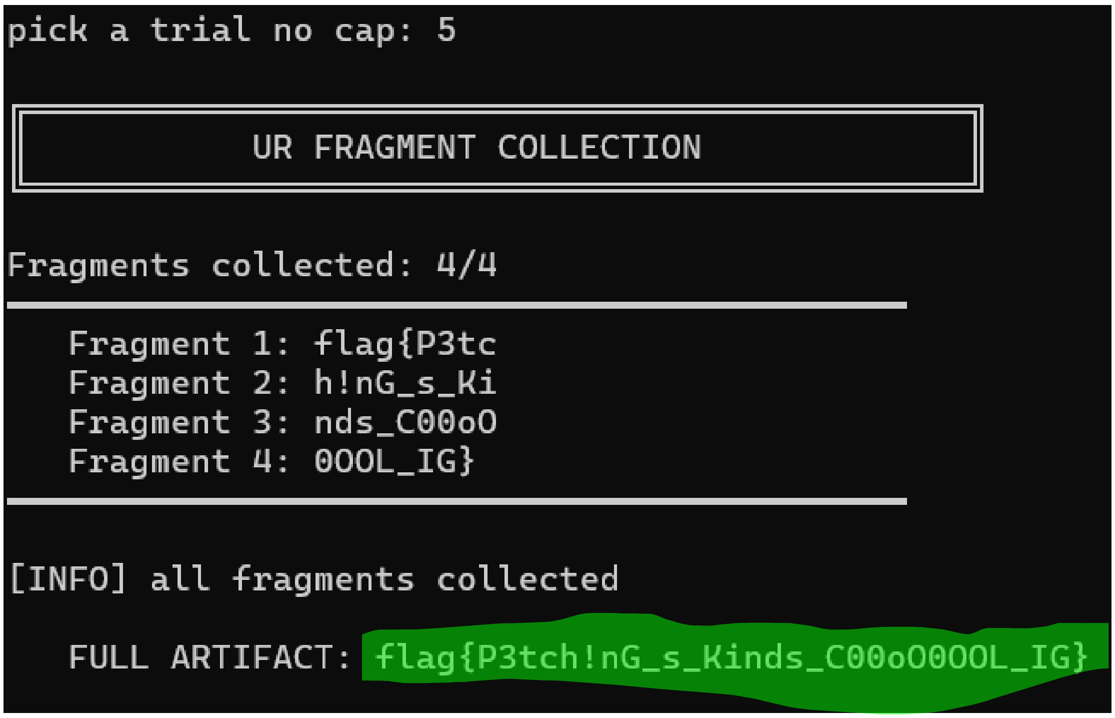
Run the exe and test the first trial and get the 4th fragment of the flag



IDA View-B

Address	Length	Original bytes	Patched bytes
00007FF62C648971	0x2	08 C8	30 C0
00007FF62C6489C7	0x2	08 C8	30 C0
00007FF62C648A17	0x2	08 C8	30 C0
00007FF62C648A72	0x2	08 C8	30 C0

The four patched bytes



```
pick a trial no cap: 5

╔════════════════════════════════════════════════╗
║          UR FRAGMENT COLLECTION                ║
╚════════════════════════════════════════════════╝

Fragments collected: 4/4
════════════════════════════════════════════
   Fragment 1: flag{P3tc
   Fragment 2: h!nG_s_Ki
   Fragment 3: nds_C00oO
   Fragment 4: 0OOL_IG}
════════════════════════════════════════════

[INFO] all fragments collected

   FULL ARTIFACT: flag{P3tch!nG_s_Kinds_C00oO0OOL_IG}
```