Reversing & Stack Based Buffer Overflows

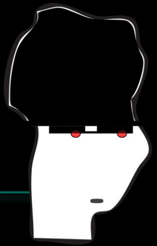
Seguridad Ofensiva

by Joshep Cortez S.
OLAPIC - FAMAF - IUA

¿Qué hace este programa?

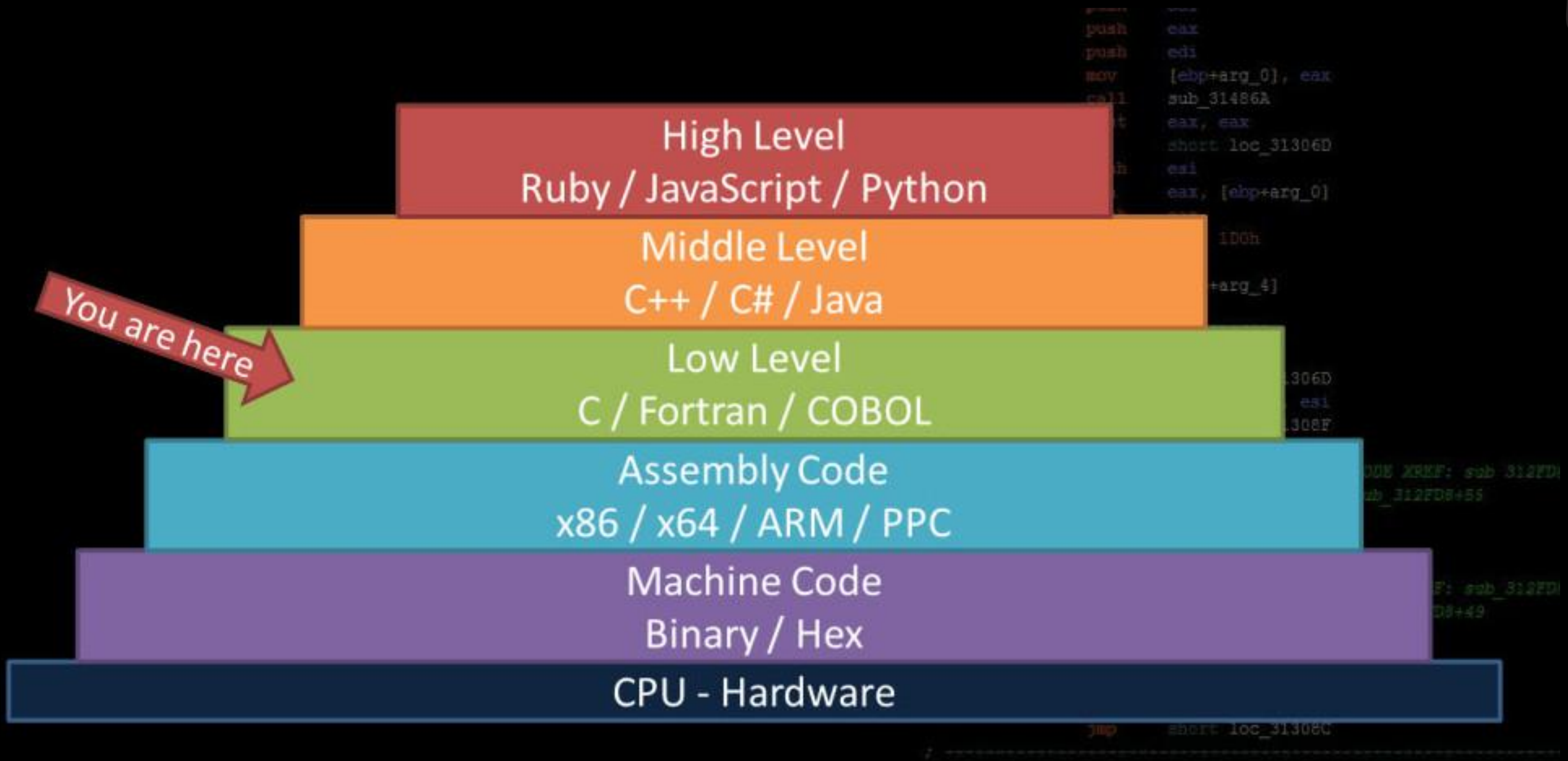
```
int main() {
   int cookie;
   char buf[80];

   gets(buf); //Lee hasta el primer ...
   if (cookie == 0x41424344)
      printf("Ganaste!\n");
}
```

¿Se puede ganar?

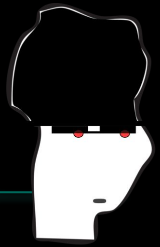


```
int main() {
    int cookie;
    char buf[80];

    gets(buf); //Lee hasta el primer ...
    if (cookie == 0x41424344)
        printf("Ganaste!\n");
}
```

Lenguajes

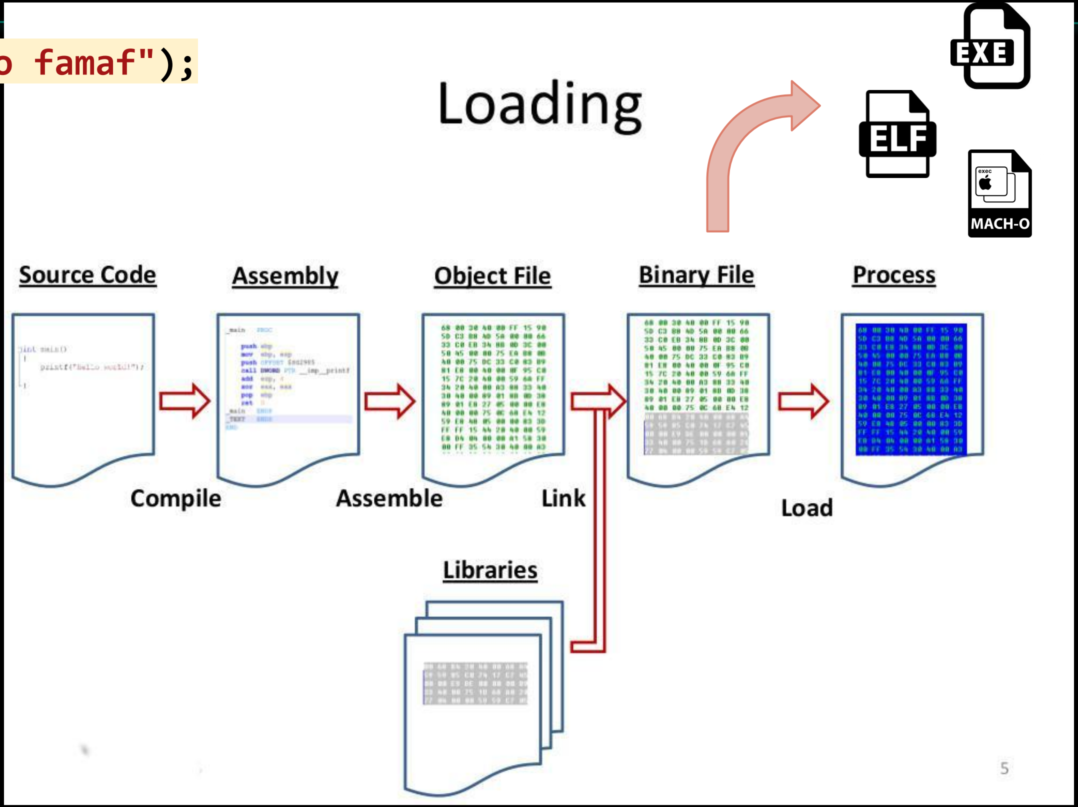

High Level
Ruby / JavaScript / Python
Middle Level
C++ / C# / Java
Low Level
C / Fortran / COBOL
Assembly Code
x86 / x64 / ARM / PPC
Machine Code
Binary / Hex
CPU - Hardware
You are here

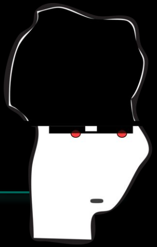
Hoy vamos a hablar



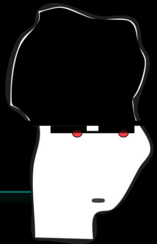
- Arquitectura Básica: Memoria / Regs
- Ensamblador: Instrucciones / Syscall
- Reversing
- Debugging

```c
int main(void) {
    printf ("hello famaf");
    return 0;
}
```





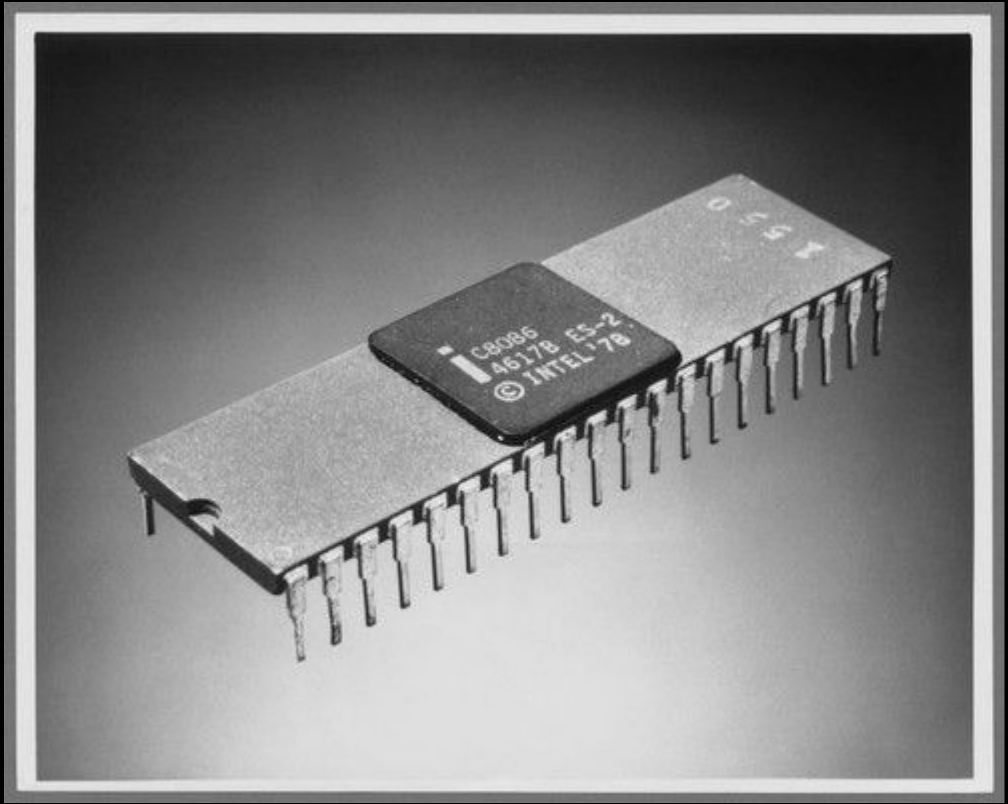
Arquitecturas: Hay un montón



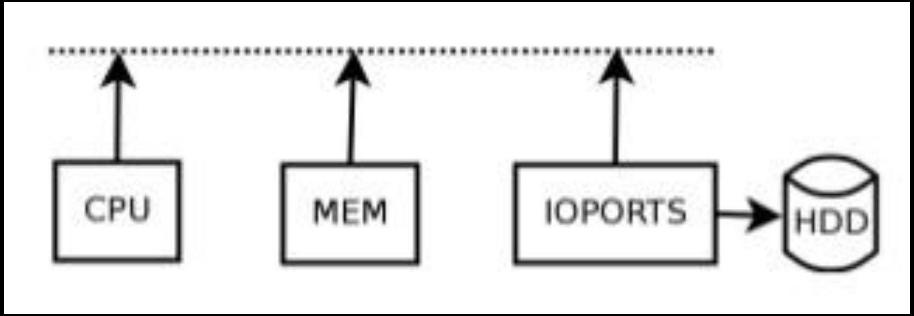
- x86 -   IA-32
- x86-64
- ARM (celulares/tablets/*duinos/Rasp*)
- SPARC (Sun)
- Power[1-9]/PC (PS3/Xbox)
- MIPS (PS2)
- aarch64

Today(): micro-arch

Arquitectura: x86

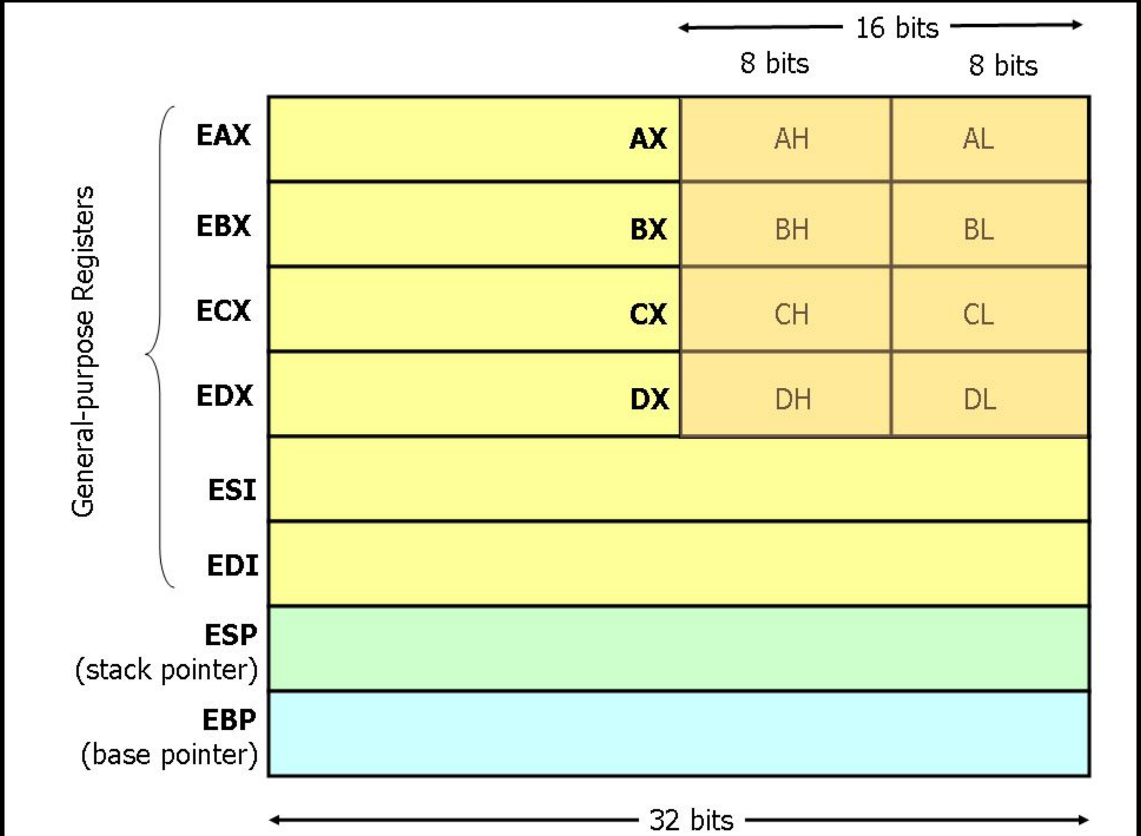


Repasando: Arquitectura Básica



```
cat /proc/cpuinfo
cat /proc/meminfo
```

Arquitectura x86: Registros



Arquitectura x86: Memoria

La memoria se direcciona por byte(8 bits)

El bit menos significativo a la derecha

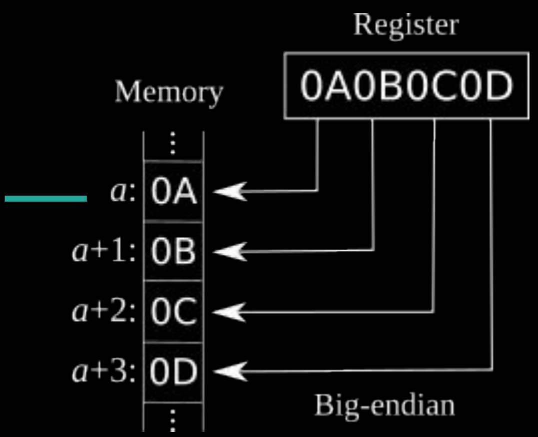
Hay instrucciones que acceden a más de un byte a la vez.

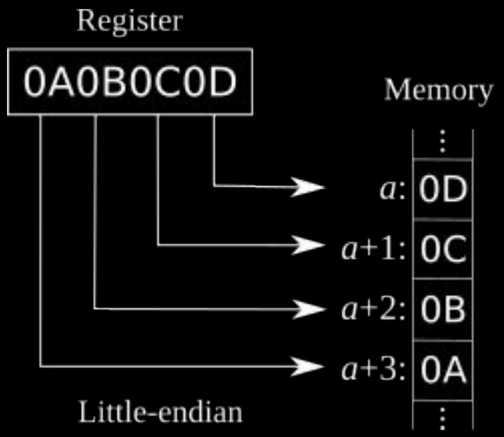
Arquitectura x86: Memoria

- Un WORD ocupa 2 bytes (16 bits). De 0 a 64K (-1)
- Un DWORD ocupa 4 bytes (32 bits). De 0 a 4G (-1)
- En Little Endian está como "al revés"
- Intel es LE

Arquitectura x86: Endianess

"Política" de lectura/escritura.

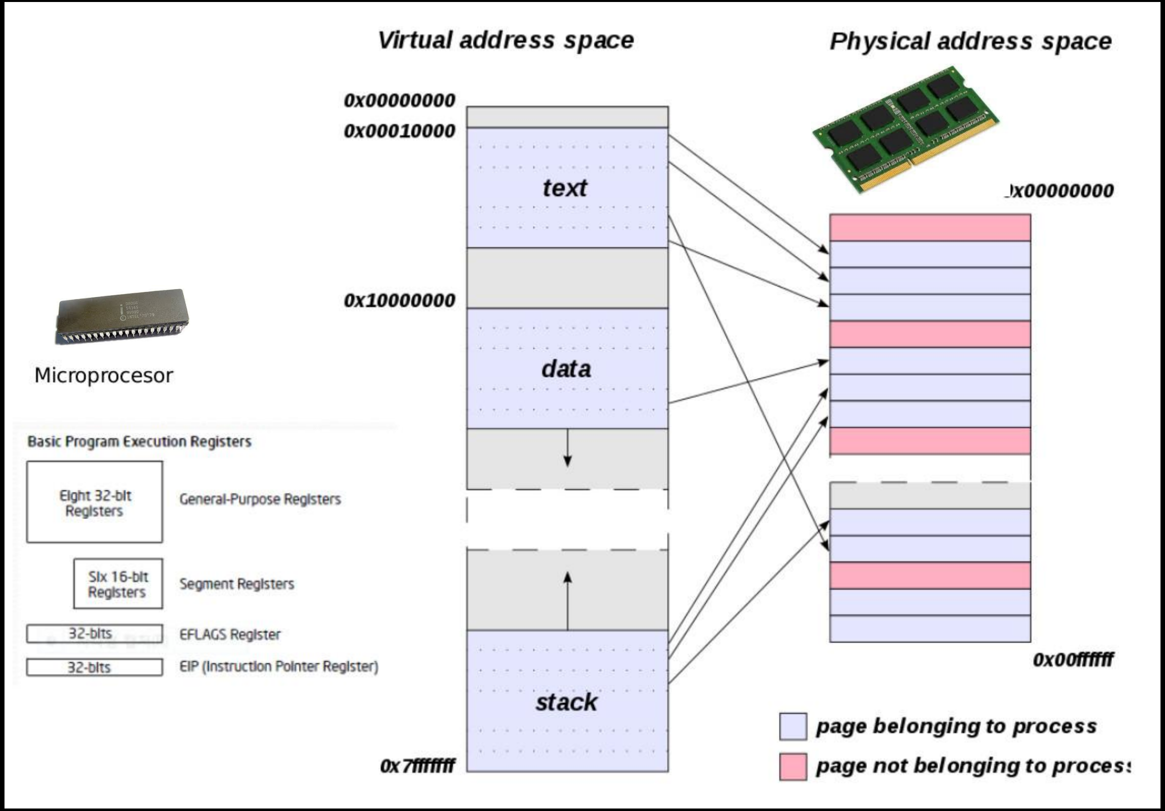




Arquitectura `x86: Bin`

- El código de un programa se termina convirtiendo en binario crudo.
- An executable such as an .EXE, ELF, MachO or other code containers that run on a machine
- Other names: program, application, service (sometimes)
- `gcc -E , -S , -c , -o`

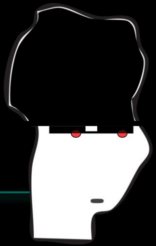
Arquitectura x86: Ejecución



Arquitectura x86:

Definiciones y datos a tener en cuenta.

- static vars, buffers,
- data types, sizes
- dynamic vars
- overflow: desbordamiento



Arquitectura x86: ASM

Instrucciones - Mnemónicos:

```
LABEL: OPCODE arg1, arg2, arg3
```

- LABEL un identificador seguido de ":"
- OPCODE nombre que representa una instrucci\'on
- ARGUMENTOS que pueden ser inmediatos o referencias a memoria

Ejemplo:

```
    func1:    MOV EAX, EDX
```

Arquitectura x86: Sintaxis ASM

- ATT

  Ej: `mov %eax, %edx`

- Intel

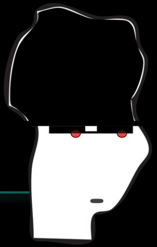
  Ej: `mov edx, eax`

```
(gdb) set disassembly-flavor intel
```

Arquitectura x86: ASM

- Acceso a memoria: `MOV, MOVS, MOVB, XCHG, ...`
  Ej: `mov %eax, %edx`
- Aritméticas: `ADD, SUB, MUL, NEG, INC, DEC, ...`
  Ej: `inc %edx`
- Lógicas: AND, OR, XOR, NOT, ...
  Ej: `xor %edx, %edx`
- Comparaciones: AND, OR, XOR, NOT, ...
  Ej: `cmp %eax, 8(%ebp)`
- Saltos: JMP, JGE, JE, JG, JLE, ...
  Ej: `je $0x12345678`

x86: Llamadas a función



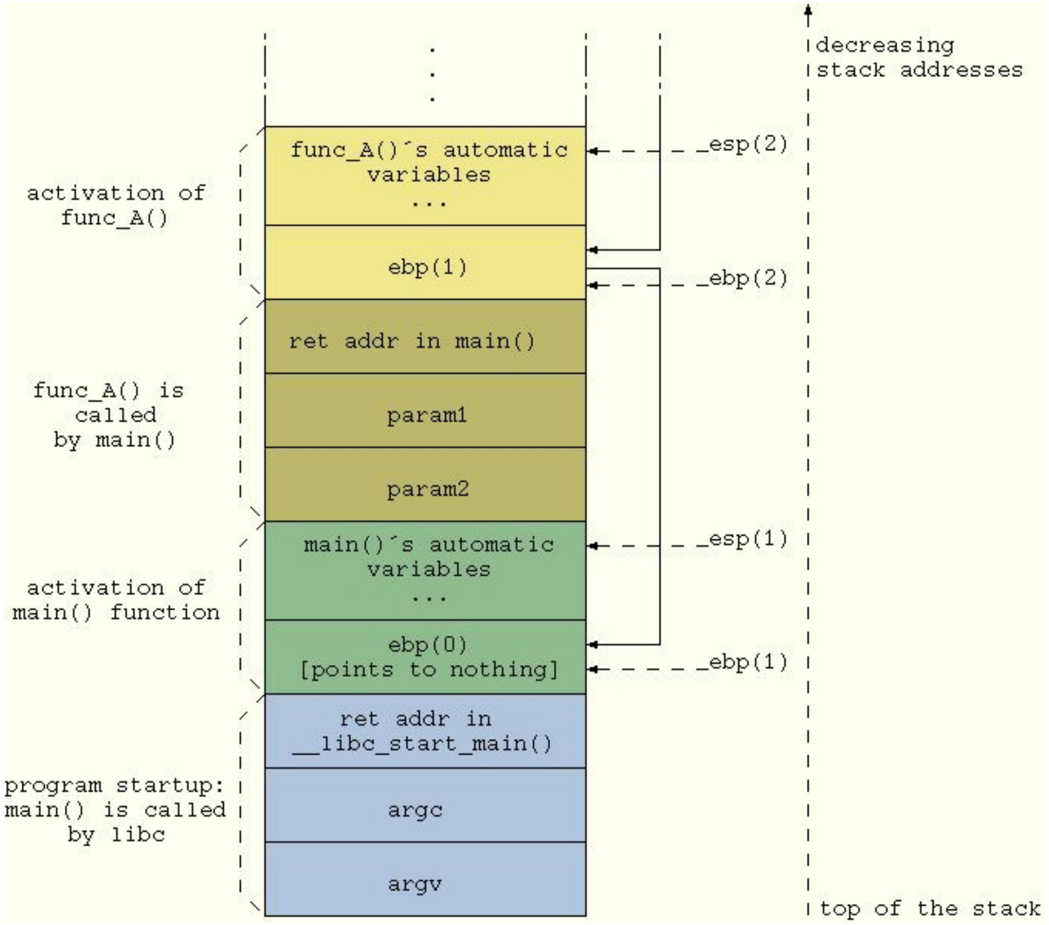
`CALL`: Pushes the offset of the next instruction onto the stack and branches to the target address, which contains the first instruction of the called procedure...

`RET`: Returns from a procedure previously entered by a CALL near instruction. This form of the RET instruction returns to a calling procedure within the current code segment…

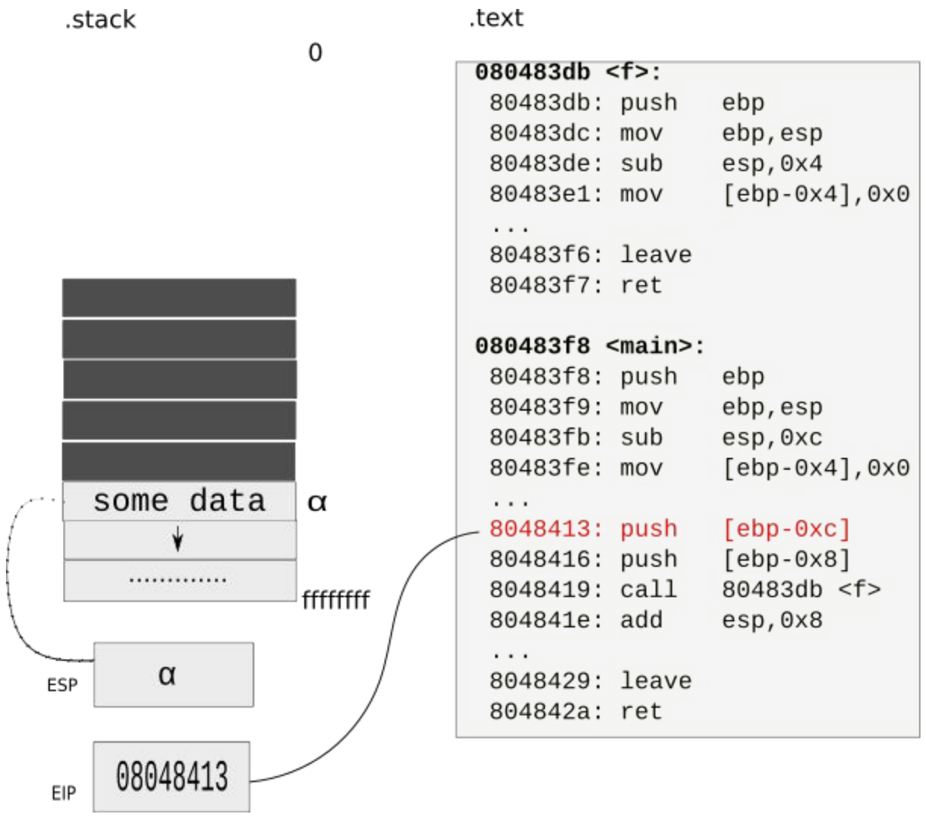
Formal Spec

x86: f(x,y)

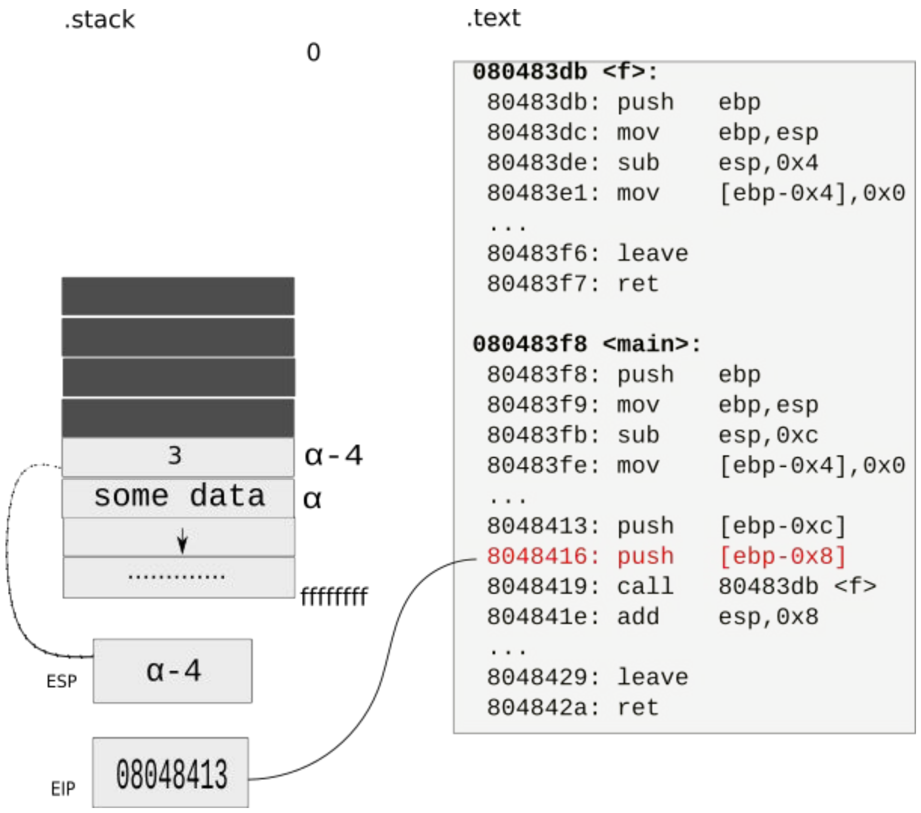
```
int func_A(int a, int b){
     int s = 0;
     s = a + b;
     return s;
}

int main(void){
     int r = 0, x = 2, y = 3;
     r = f(x,y);
     return 0;
}
```


decreasing stack addresses
func_A()´s automatic variables ...
_esp(2)
activation of func_A()
ebp(1)
_ebp(2)
ret addr in main()
func_A() is called by main()
param1
param2
main()´s automatic variables ...
_esp(1)
activation of main() function
ebp(0) [points to nothing]
_ebp(1)
ret addr in __libc_start_main()
program startup: main() is called by libc
argc
argv
top of the stack

x86:

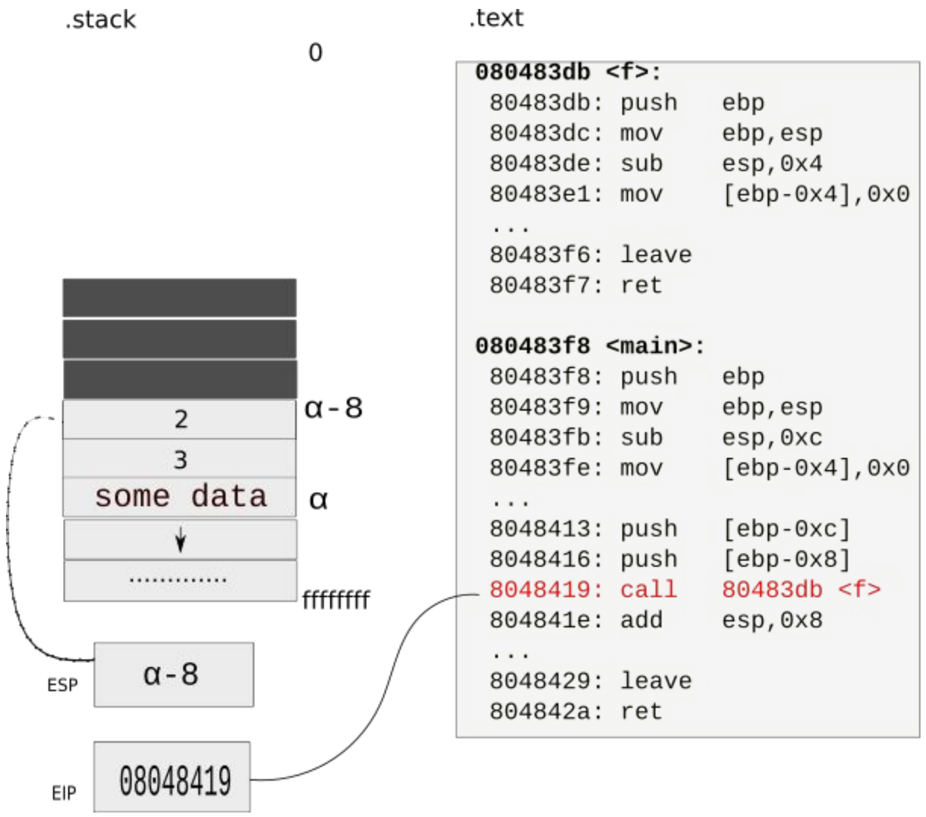


```
080483db <f>:
 80483db: push   ebp
 80483dc: mov    ebp,esp
 80483de: sub    esp,0x4
 80483e1: mov    [ebp-0x4],0x0
 ...
 80483f6: leave
 80483f7: ret

080483f8 <main>:
 80483f8: push   ebp
 80483f9: mov    ebp,esp
 80483fb: sub    esp,0xc
 80483fe: mov    [ebp-0x4],0x0
 ...
 8048413: push   [ebp-0xc]
 8048416: push   [ebp-0x8]
 8048419: call   80483db <f>
 804841e: add    esp,0x8
 ...
 8048429: leave
 804842a: ret
```

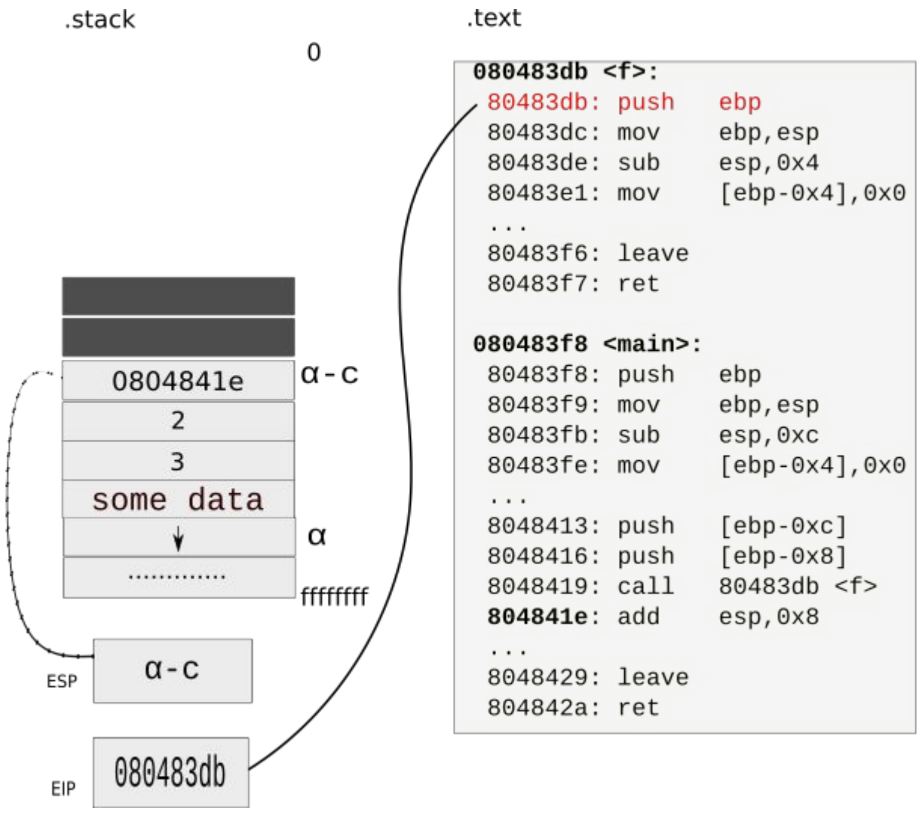


```
080483db <f>:
 80483db: push   ebp
 80483dc: mov    ebp,esp
 80483de: sub    esp,0x4
 80483e1: mov    [ebp-0x4],0x0
 ...
 80483f6: leave
 80483f7: ret

080483f8 <main>:
 80483f8: push   ebp
 80483f9: mov    ebp,esp
 80483fb: sub    esp,0xc
 80483fe: mov    [ebp-0x4],0x0
 ...
 8048413: push   [ebp-0xc]
 8048416: push   [ebp-0x8]
 8048419: call   80483db <f>
 804841e: add    esp,0x8
 ...
 8048429: leave
 804842a: ret
```

x86:

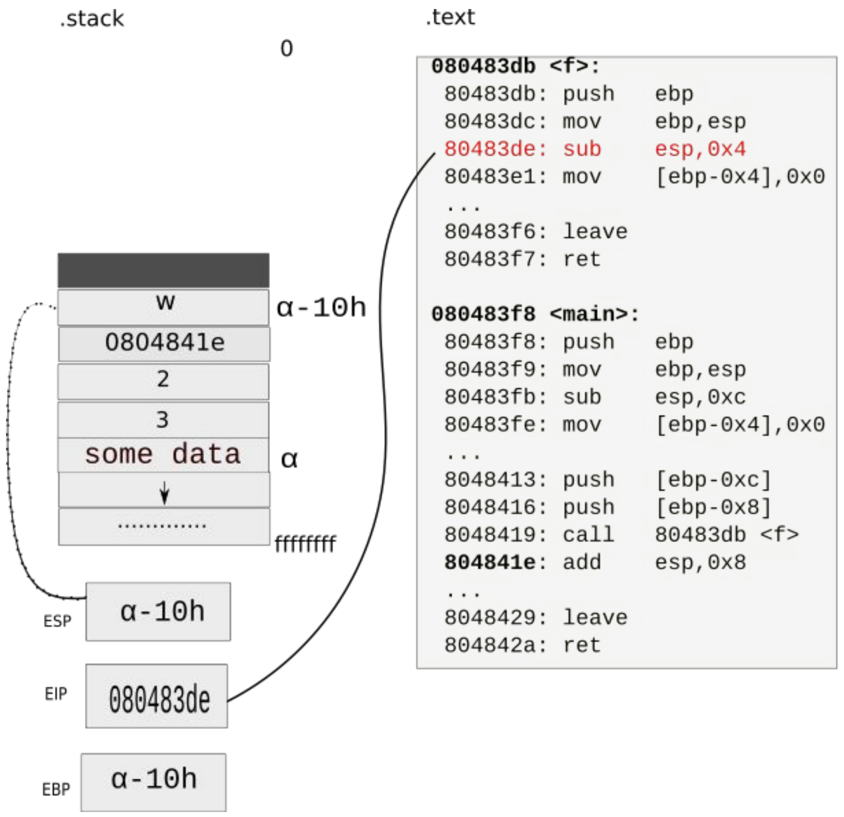


```
080483db <f>:
 80483db: push   ebp
 80483dc: mov    ebp,esp
 80483de: sub    esp,0x4
 80483e1: mov    [ebp-0x4],0x0
 ...
 80483f6: leave
 80483f7: ret

080483f8 <main>:
 80483f8: push   ebp
 80483f9: mov    ebp,esp
 80483fb: sub    esp,0xc
 80483fe: mov    [ebp-0x4],0x0
 ...
 8048413: push   [ebp-0xc]
 8048416: push   [ebp-0x8]
 8048419: call   80483db <f>
 804841e: add    esp,0x8
 ...
 8048429: leave
 804842a: ret
```

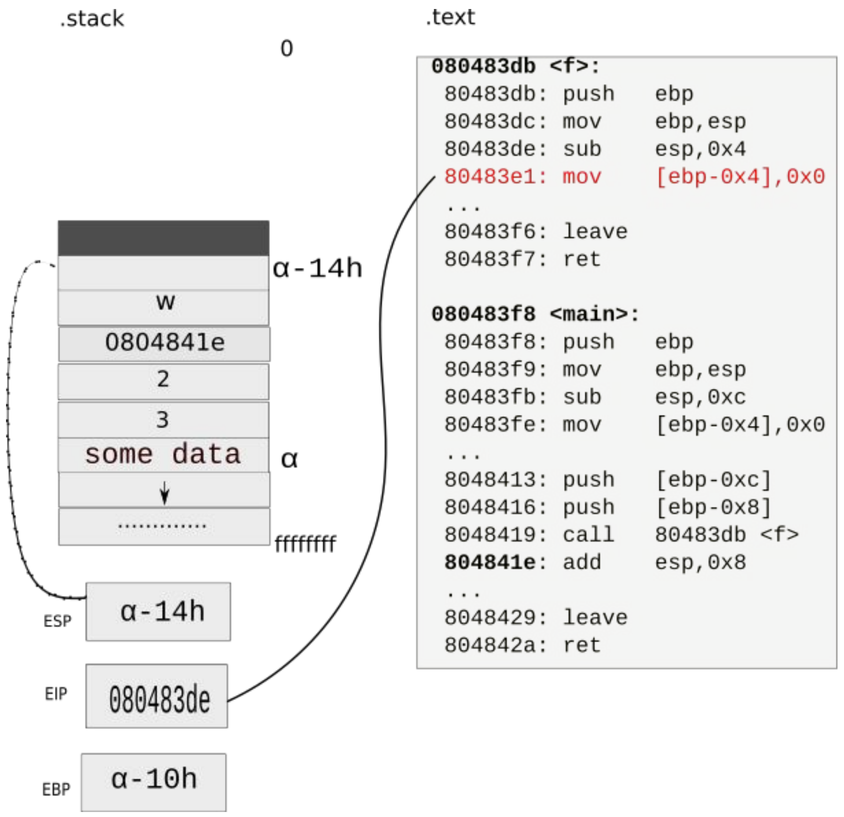


```
080483db <f>:
 80483db: push   ebp
 80483dc: mov    ebp,esp
 80483de: sub    esp,0x4
 80483e1: mov    [ebp-0x4],0x0
 ...
 80483f6: leave
 80483f7: ret

080483f8 <main>:
 80483f8: push   ebp
 80483f9: mov    ebp,esp
 80483fb: sub    esp,0xc
 80483fe: mov    [ebp-0x4],0x0
 ...
 8048413: push   [ebp-0xc]
 8048416: push   [ebp-0x8]
 8048419: call   80483db <f>
 804841e: add    esp,0x8
 ...
 8048429: leave
 804842a: ret
```

x86:



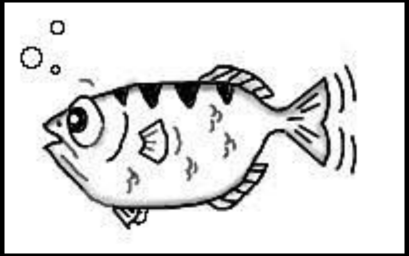
```
080483db <f>:
 80483db: push   ebp
 80483dc: mov    ebp,esp
 80483de: sub    esp,0x4
 80483e1: mov    [ebp-0x4],0x0
 ...
 80483f6: leave
 80483f7: ret

080483f8 <main>:
 80483f8: push   ebp
 80483f9: mov    ebp,esp
 80483fb: sub    esp,0xc
 80483fe: mov    [ebp-0x4],0x0
 ...
 8048413: push   [ebp-0xc]
 8048416: push   [ebp-0x8]
 8048419: call   80483db <f>
 804841e: add    esp,0x8
 ...
 8048429: leave
 804842a: ret
```

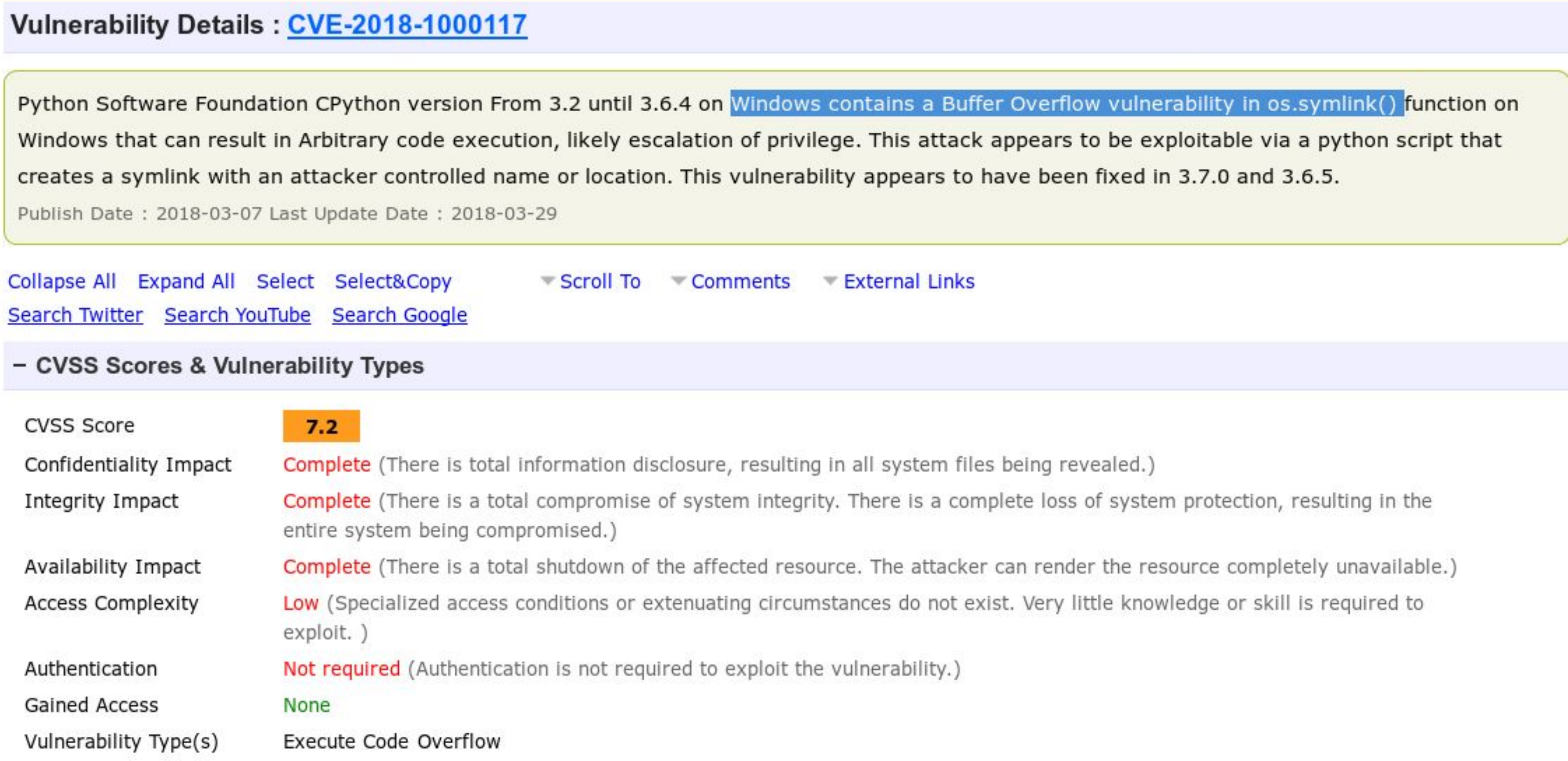


```
080483db <f>:
 80483db: push   ebp
 80483dc: mov    ebp,esp
 80483de: sub    esp,0x4
 80483e1: mov    [ebp-0x4],0x0
 ...
 80483f6: leave
 80483f7: ret

080483f8 <main>:
 80483f8: push   ebp
 80483f9: mov    ebp,esp
 80483fb: sub    esp,0xc
 80483fe: mov    [ebp-0x4],0x0
 ...
 8048413: push   [ebp-0xc]
 8048416: push   [ebp-0x8]
 8048419: call   80483db <f>
 804841e: add    esp,0x8
 ...
 8048429: leave
 804842a: ret
```

x86:

DEMO





Vulnerability Details : CVE-2018-1000117

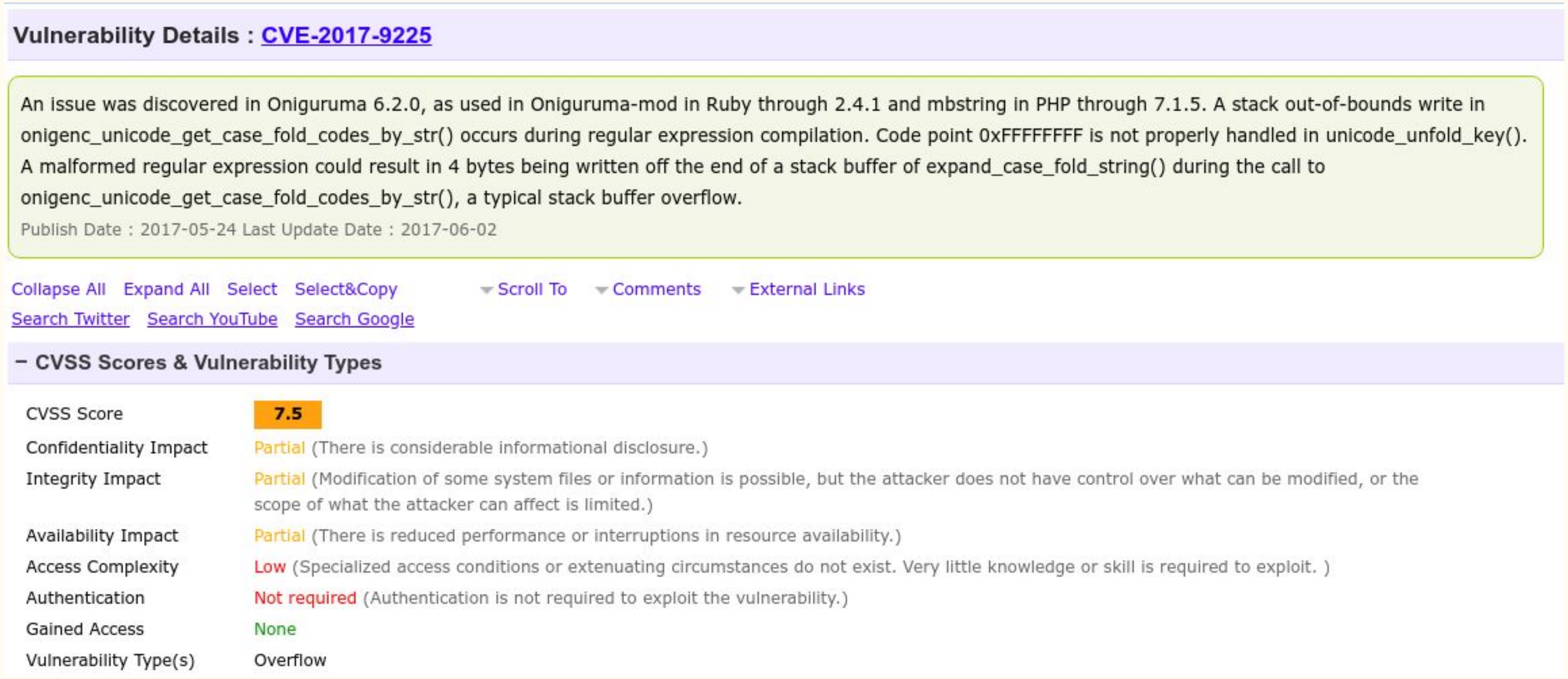
Python Software Foundation CPython version From 3.2 until 3.6.4 on Windows contains a Buffer Overflow vulnerability in os.symlink() function on Windows that can result in Arbitrary code execution, likely escalation of privilege. This attack appears to be exploitable via a python script that creates a symlink with an attacker controlled name or location. This vulnerability appears to have been fixed in 3.7.0 and 3.6.5.

Publish Date : 2018-03-07 Last Update Date : 2018-03-29

– CVSS Scores & Vulnerability Types

CVSS Score	7.2
Confidentiality Impact	Complete (There is total information disclosure, resulting in all system files being revealed.)
Integrity Impact	Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code Overflow

[Python] https://www.cvedetails.com/cve/CVE-2018-1000117/



Vulnerability Details : CVE-2017-9225

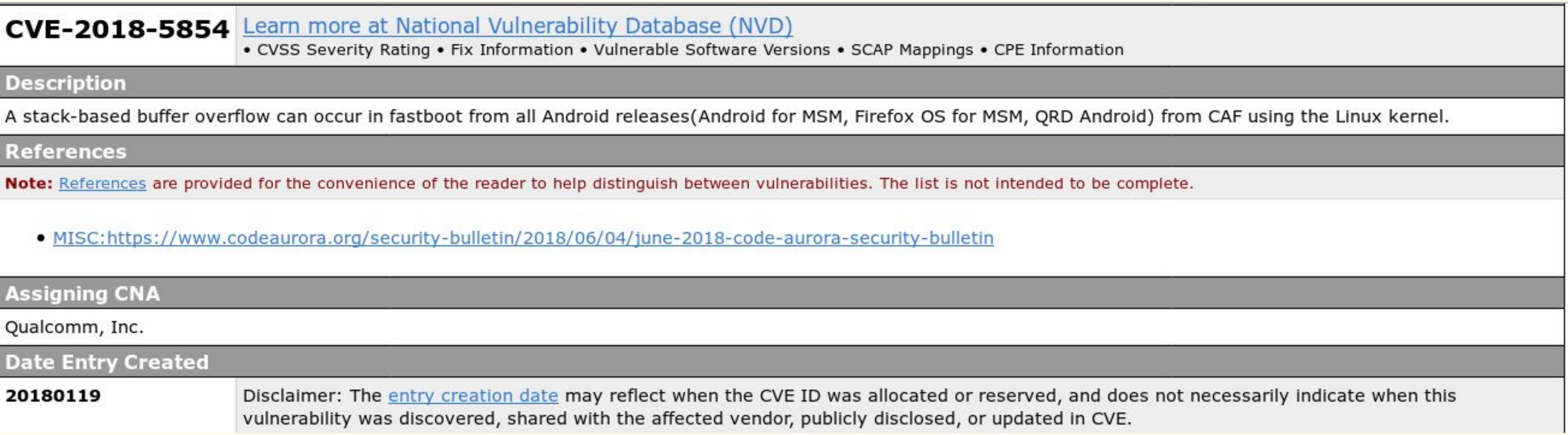
An issue was discovered in Oniguruma 6.2.0, as used in Oniguruma-mod in Ruby through 2.4.1 and mbstring in PHP through 7.1.5. A stack out-of-bounds write in onigenc_unicode_get_case_fold_codes_by_str() occurs during regular expression compilation. Code point 0xFFFFFFFF is not properly handled in unicode_unfold_key(). A malformed regular expression could result in 4 bytes being written off the end of a stack buffer of expand_case_fold_string() during the call to onigenc_unicode_get_case_fold_codes_by_str(), a typical stack buffer overflow.

Publish Date : 2017-05-24 Last Update Date : 2017-06-02

− CVSS Scores & Vulnerability Types

CVSS Score	7.5
Confidentiality Impact	Partial (There is considerable informational disclosure.)
Integrity Impact	Partial (Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.)
Availability Impact	Partial (There is reduced performance or interruptions in resource availability.)
Access Complexity	Low (Specialized access conditions or extenuating circumstances do not exist. Very little knowledge or skill is required to exploit.)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Overflow

[Ruby] https://www.cvedetails.com/cve/CVE-2017-9225/


CVE-2018-5854 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A stack-based buffer overflow can occur in fastboot from all Android releases(Android for MSM, Firefox OS for MSM, QRD Android) from CAF using the Linux kernel.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- MISC:https://www.codeaurora.org/security-bulletin/2018/06/04/june-2018-code-aurora-security-bulletin

Assigning CNA

Qualcomm, Inc.

Date Entry Created

20180119	Disclaimer: The entry creation date may reflect when the CVE ID was allocated or reserved, and does not necessarily indicate when this vulnerability was discovered, shared with the affected vendor, publicly disclosed, or updated in CVE.

[Android]

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-5854

[Firefox]

https://www.mozilla.org/en-US/security/advisories/mfsa2018-29/

¿Y cuando el programador hace todo bien?

CVE-2016-5296 (Cairo)

A **heap-buffer-overflow** in Cairo when processing SVG content **caused by compiler optimization**, resulting in a potentially exploitable crash.

This vulnerability affects Thunderbird < 45.5, Firefox ESR < 45.5, and Firefox < 50.

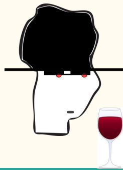
CVE-2018-2844 (Virtualbox)

https://www.voidsecurity.in/2018/08/from- **compiler-optimization**-to-code.html

(CVE-2018-0946) (Edge)

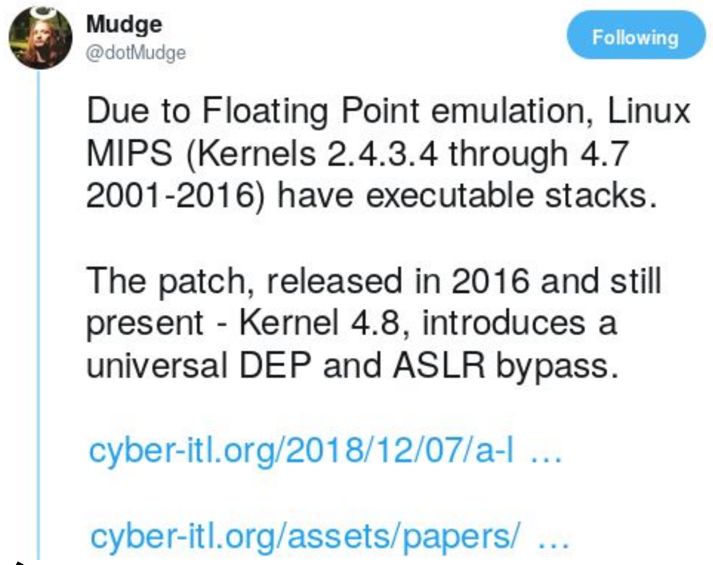
https://www.fortinet.com/blog/threat-research/an-analysis-of-the-use-after-free-bug-in-microsoft-edge-chakra-engine.html

This **use-after-free** bug occurs when the Chakra Engine tries to execute the **optimized function** code **generated by** the **just-in-time (JIT) compiler**, which has already been freed when closing the related context.



IoT

Stack Buffer Overflow - CVE-2018-16595 (high severity): Sony Bravia
This is a memory corruption vulnerability that results from insufficient size checking of user input. With a long enough HTTP POST request sent to the corresponding URL, the application will crash.
https://www.sony.co.uk/electronics/support/articles/00201041



Mudge
@dotMudge
Following

Due to Floating Point emulation, Linux MIPS (Kernels 2.4.3.4 through 4.7 2001-2016) have executable stacks.

The patch, released in 2016 and still present - Kernel 4.8, introduces a universal DEP and ASLR bypass.

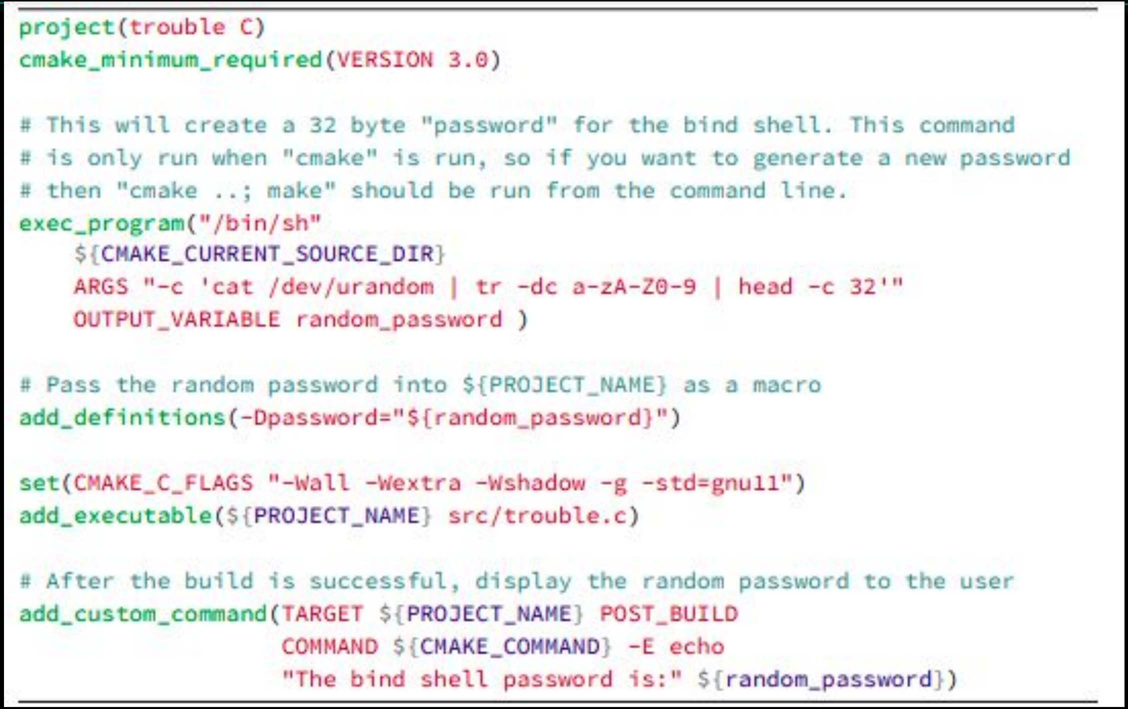
cyber-itl.org/2018/12/07/a-l …

cyber-itl.org/assets/papers/ …

CVE-2018-4249 DoS Exec Code Overflow: **Apple TV** …….
An issue was discovered in certain Apple products. iOS before 11.4 is affected. macOS before 10.13.5 is affected. tvOS before 11.4 is affected. **watchOS** before 4.3.1 is affected. The issue involves pktmnglr_ipfilter_input in com.apple.packet-mangler in the "Kernel" component. It allows attackers to execute arbitrary code in a privileged context or cause a denial of service (integer overflow and stack-based buffer overflow) via a crafted app.

CVE-2018-3938 Exec Code Overflow
An exploitable stack-based buffer overflow vulnerability exists in the 802dot1xclientcert.cgi functionality of **Sony IPELA E Series Camera** G5 firmware 1.87.00. A specially crafted POST can cause a stack-based buffer overflow, resulting in remote code execution. An attacker can send a malicious POST request to trigger this vulnerability.

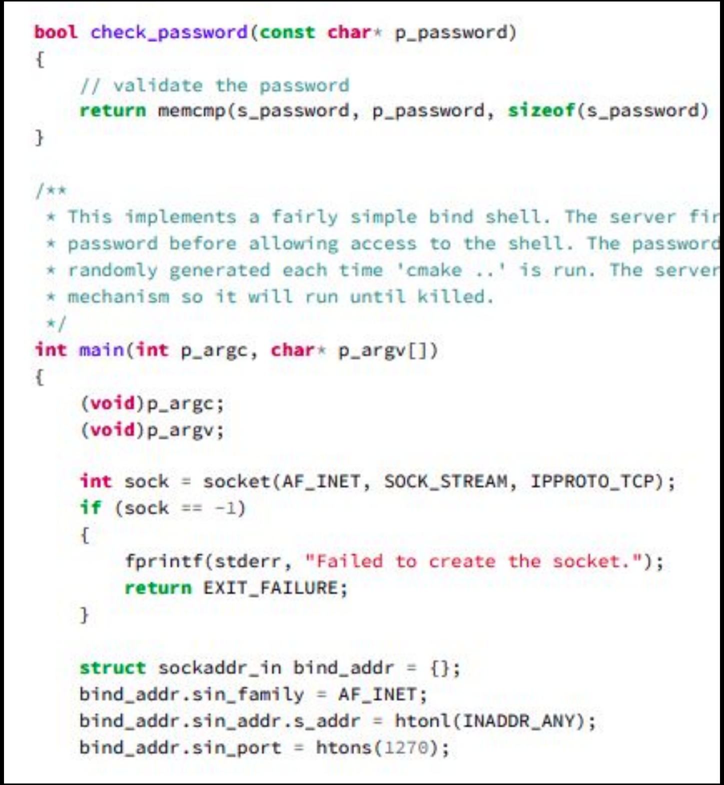
Reversing



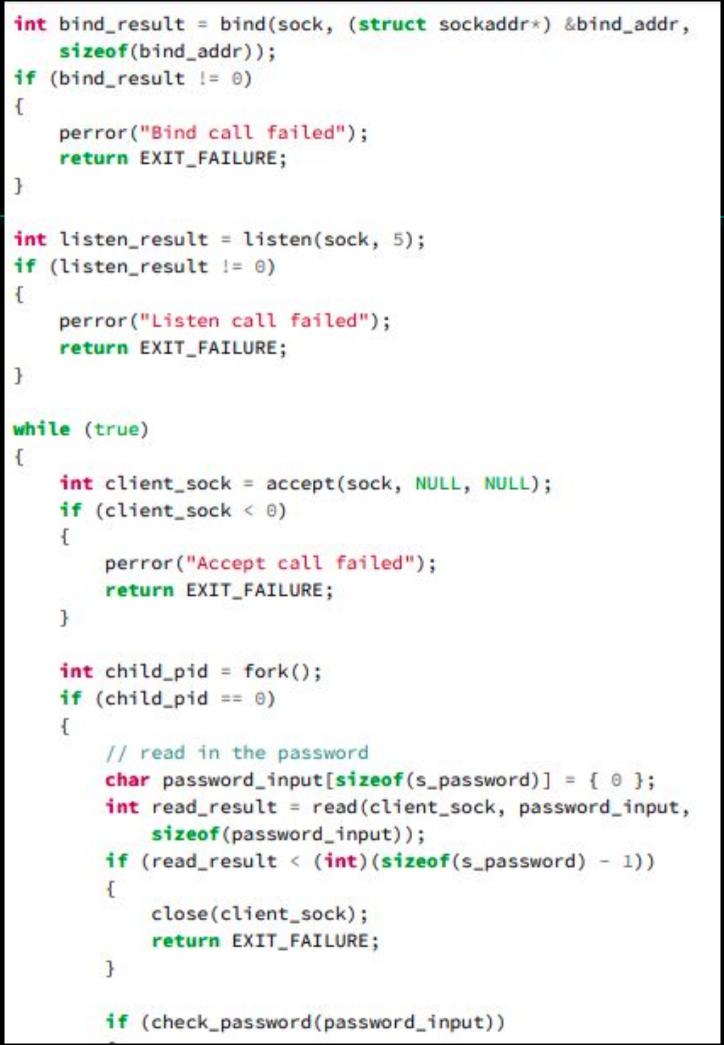
```
project(trouble C)
cmake_minimum_required(VERSION 3.0)

# This will create a 32 byte "password" for the bind shell. This command
# is only run when "cmake" is run, so if you want to generate a new password
# then "cmake ..; make" should be run from the command line.
exec_program("/bin/sh"
    ${CMAKE_CURRENT_SOURCE_DIR}
    ARGS "-c 'cat /dev/urandom | tr -dc a-zA-Z0-9 | head -c 32'"
    OUTPUT_VARIABLE random_password )

# Pass the random password into ${PROJECT_NAME} as a macro
add_definitions(-Dpassword="${random_password}")

set(CMAKE_C_FLAGS "-Wall -Wextra -Wshadow -g -std=gnu11")
add_executable(${PROJECT_NAME} src/trouble.c)

# After the build is successful, display the random password to the user
add_custom_command(TARGET ${PROJECT_NAME} POST_BUILD
                   COMMAND ${CMAKE_COMMAND} -E echo
                   "The bind shell password is:" ${random_password})
```

https://github.com/antire-book/antire_book.

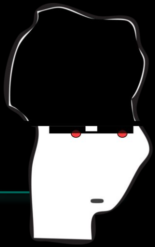
Reversing



```
bool check_password(const char* p_password)
{
    // validate the password
    return memcmp(s_password, p_password, sizeof(s_password)
}

/**
 * This implements a fairly simple bind shell. The server fir
 * password before allowing access to the shell. The password
 * randomly generated each time 'cmake ..' is run. The server
 * mechanism so it will run until killed.
 */
int main(int p_argc, char* p_argv[])
{
    (void)p_argc;
    (void)p_argv;

    int sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if (sock == -1)
    {
        fprintf(stderr, "Failed to create the socket.");
        return EXIT_FAILURE;
    }

    struct sockaddr_in bind_addr = {};
    bind_addr.sin_family = AF_INET;
    bind_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    bind_addr.sin_port = htons(1270);
```

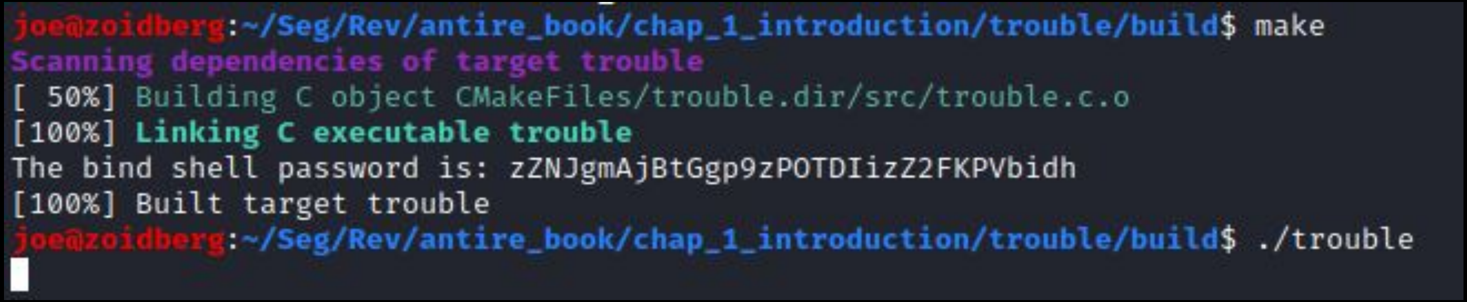


```
int bind_result = bind(sock, (struct sockaddr*) &bind_addr,
    sizeof(bind_addr));
if (bind_result != 0)
{
    perror("Bind call failed");
    return EXIT_FAILURE;
}

int listen_result = listen(sock, 5);
if (listen_result != 0)
{
    perror("Listen call failed");
    return EXIT_FAILURE;
}

while (true)
{
    int client_sock = accept(sock, NULL, NULL);
    if (client_sock < 0)
    {
        perror("Accept call failed");
        return EXIT_FAILURE;
    }

    int child_pid = fork();
    if (child_pid == 0)
    {
        // read in the password
        char password_input[sizeof(s_password)] = { 0 };
        int read_result = read(client_sock, password_input,
            sizeof(password_input));
        if (read_result < (int)(sizeof(s_password) - 1))
        {
            close(client_sock);
            return EXIT_FAILURE;
        }

        if (check_password(password_input))
```

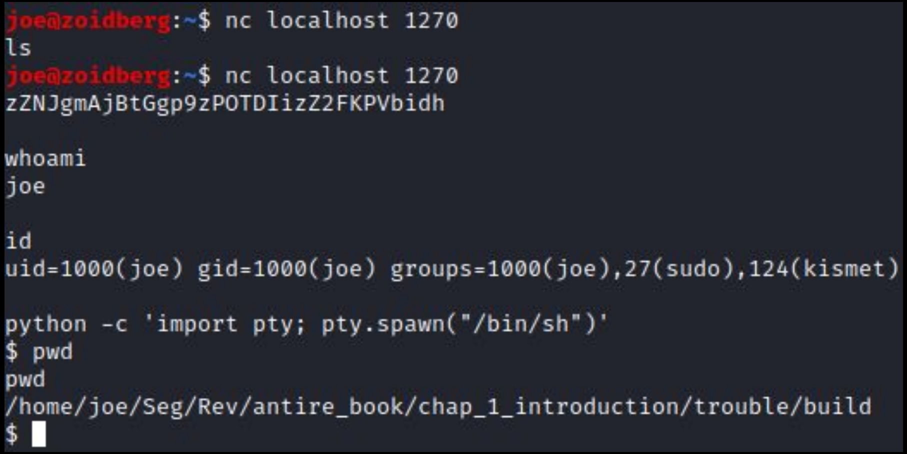


...

Reversing

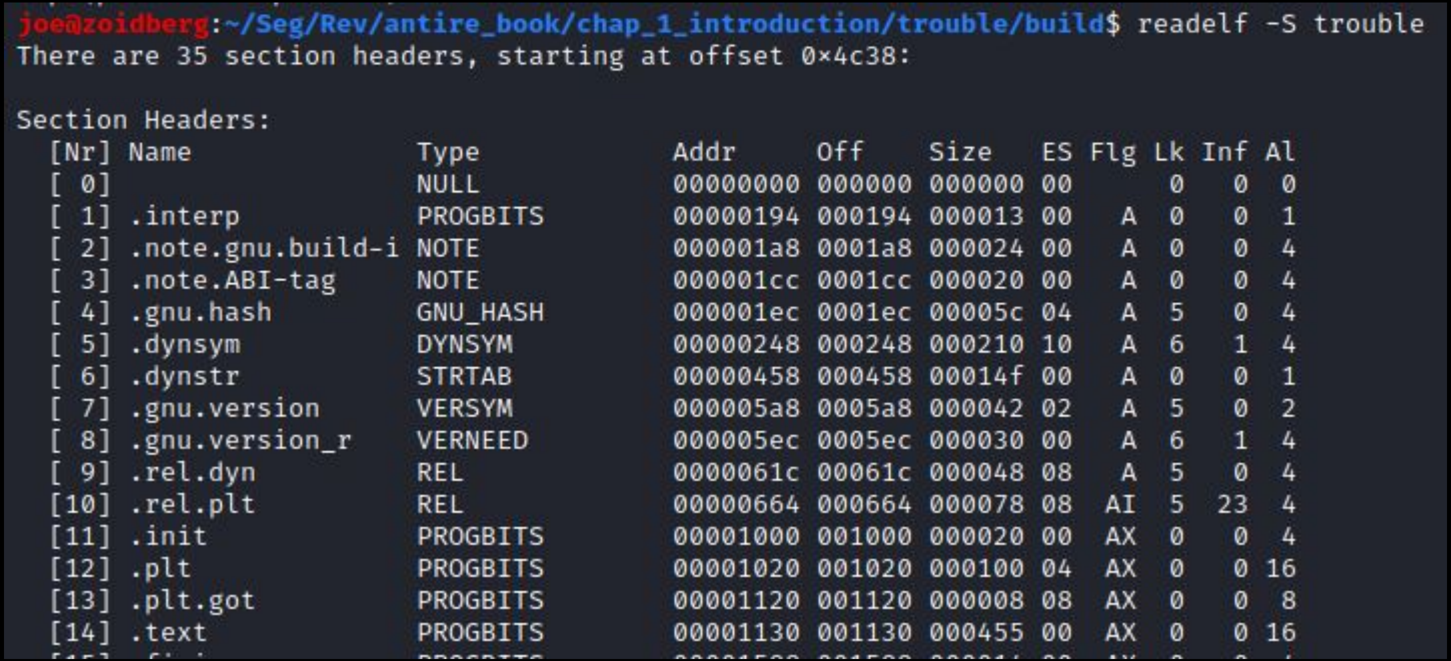


```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ make
Scanning dependencies of target trouble
[ 50%] Building C object CMakeFiles/trouble.dir/src/trouble.c.o
[100%] Linking C executable trouble
The bind shell password is: zZNJgmAjBtGgp9zPOTDIizZ2FKPVbidh
[100%] Built target trouble
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ ./trouble
```

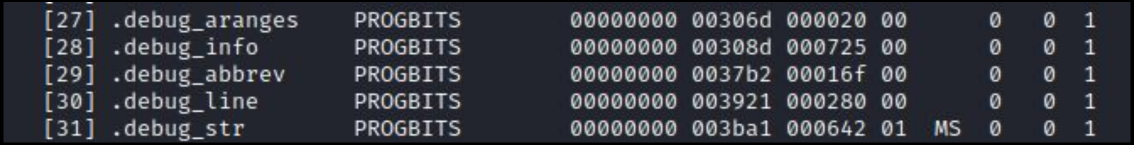


```
joe@zoidberg:~$ nc localhost 1270
ls
joe@zoidberg:~$ nc localhost 1270
zZNJgmAjBtGgp9zPOTDIizZ2FKPVbidh

whoami
joe

id
uid=1000(joe) gid=1000(joe) groups=1000(joe),27(sudo),124(kismet)

python -c 'import pty; pty.spawn("/bin/sh")'
$ pwd
pwd
/home/joe/Seg/Rev/antire_book/chap_1_introduction/trouble/build
$ 
```

Reversing

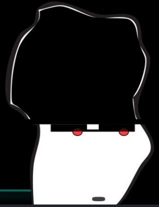


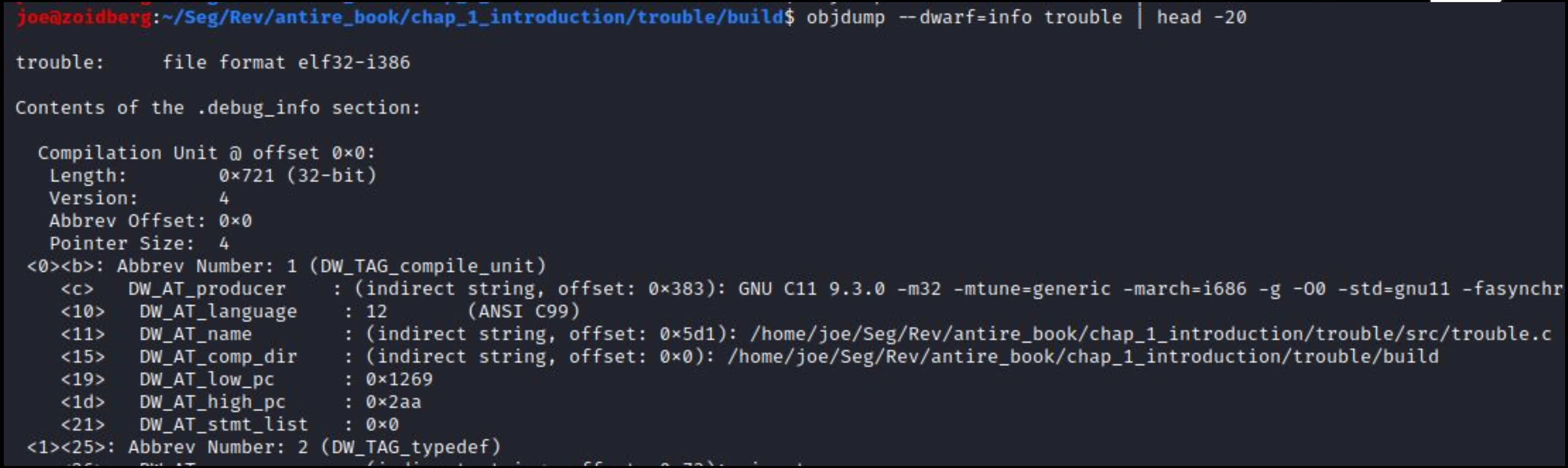
```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ readelf -S trouble
There are 35 section headers, starting at offset 0x4c38:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        00000194 000194 000013 00   A  0   0  1
  [ 2] .note.gnu.build-i NOTE            000001a8 0001a8 000024 00   A  0   0  4
  [ 3] .note.ABI-tag     NOTE            000001cc 0001cc 000020 00   A  0   0  4
  [ 4] .gnu.hash         GNU_HASH        000001ec 0001ec 00005c 04   A  5   0  4
  [ 5] .dynsym           DYNSYM          00000248 000248 000210 10   A  6   1  4
  [ 6] .dynstr           STRTAB          00000458 000458 00014f 00   A  0   0  1
  [ 7] .gnu.version      VERSYM          000005a8 0005a8 000042 02   A  5   0  2
  [ 8] .gnu.version_r    VERNEED         000005ec 0005ec 000030 00   A  6   1  4
  [ 9] .rel.dyn          REL             0000061c 00061c 000048 08   A  5   0  4
  [10] .rel.plt          REL             00000664 000664 000078 08  AI  5  23  4
  [11] .init             PROGBITS        00001000 001000 000020 00  AX  0   0  4
  [12] .plt              PROGBITS        00001020 001020 000100 04  AX  0   0 16
  [13] .plt.got          PROGBITS        00001120 001120 000008 08  AX  0   0  8
  [14] .text             PROGBITS        00001130 001130 000455 00  AX  0   0 16
```



```
  [27] .debug_aranges    PROGBITS        00000000 00306d 000020 00      0   0  1
  [28] .debug_info       PROGBITS        00000000 00308d 000725 00      0   0  1
  [29] .debug_abbrev     PROGBITS        00000000 0037b2 00016f 00      0   0  1
  [30] .debug_line       PROGBITS        00000000 003921 000280 00      0   0  1
  [31] .debug_str        PROGBITS        00000000 003ba1 000642 01  MS  0   0  1
```

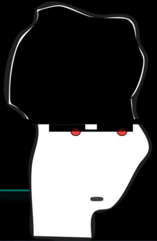
Reversing

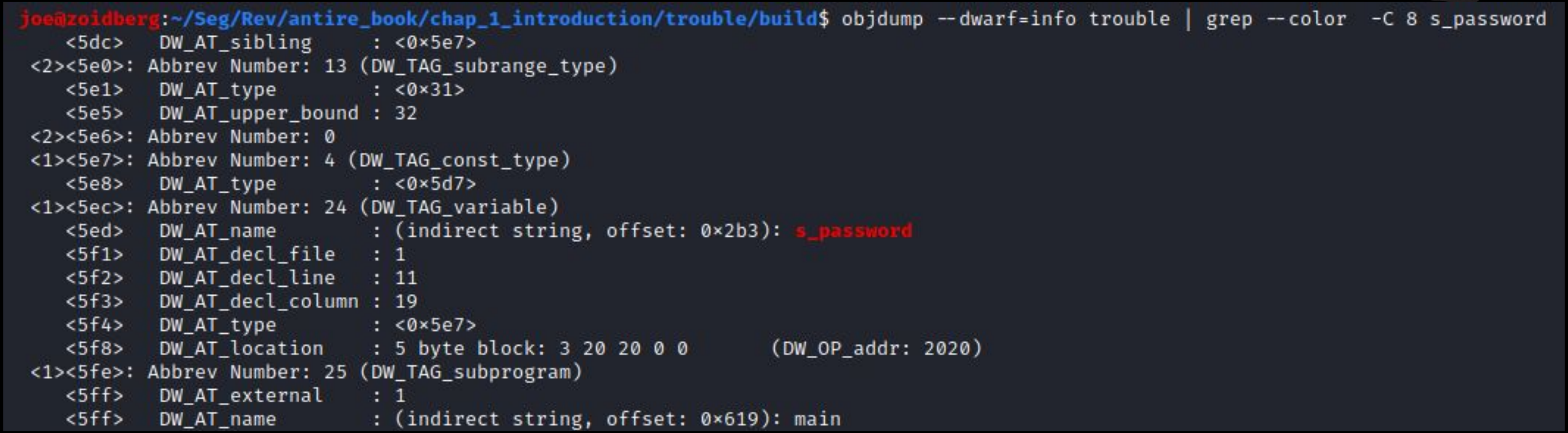




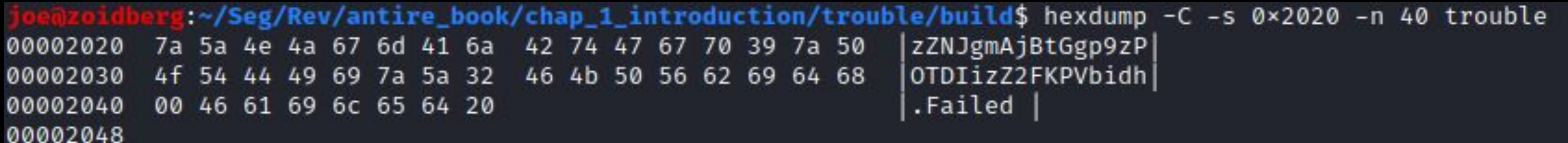
```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ objdump --dwarf=info trouble | head -20

trouble:     file format elf32-i386

Contents of the .debug_info section:

  Compilation Unit @ offset 0x0:
   Length:        0x721 (32-bit)
   Version:       4
   Abbrev Offset: 0x0
   Pointer Size:  4
 <0><b>: Abbrev Number: 1 (DW_TAG_compile_unit)
    <c>   DW_AT_producer    : (indirect string, offset: 0x383): GNU C11 9.3.0 -m32 -mtune=generic -march=i686 -g -O0 -std=gnu11 -fasynchr
    <10>   DW_AT_language    : 12       (ANSI C99)
    <11>   DW_AT_name        : (indirect string, offset: 0x5d1): /home/joe/Seg/Rev/antire_book/chap_1_introduction/trouble/src/trouble.c
    <15>   DW_AT_comp_dir    : (indirect string, offset: 0x0): /home/joe/Seg/Rev/antire_book/chap_1_introduction/trouble/build
    <19>   DW_AT_low_pc      : 0x1269
    <1d>   DW_AT_high_pc     : 0x2aa
    <21>   DW_AT_stmt_list   : 0x0
 <1><25>: Abbrev Number: 2 (DW_TAG_typedef)
```

Reversing



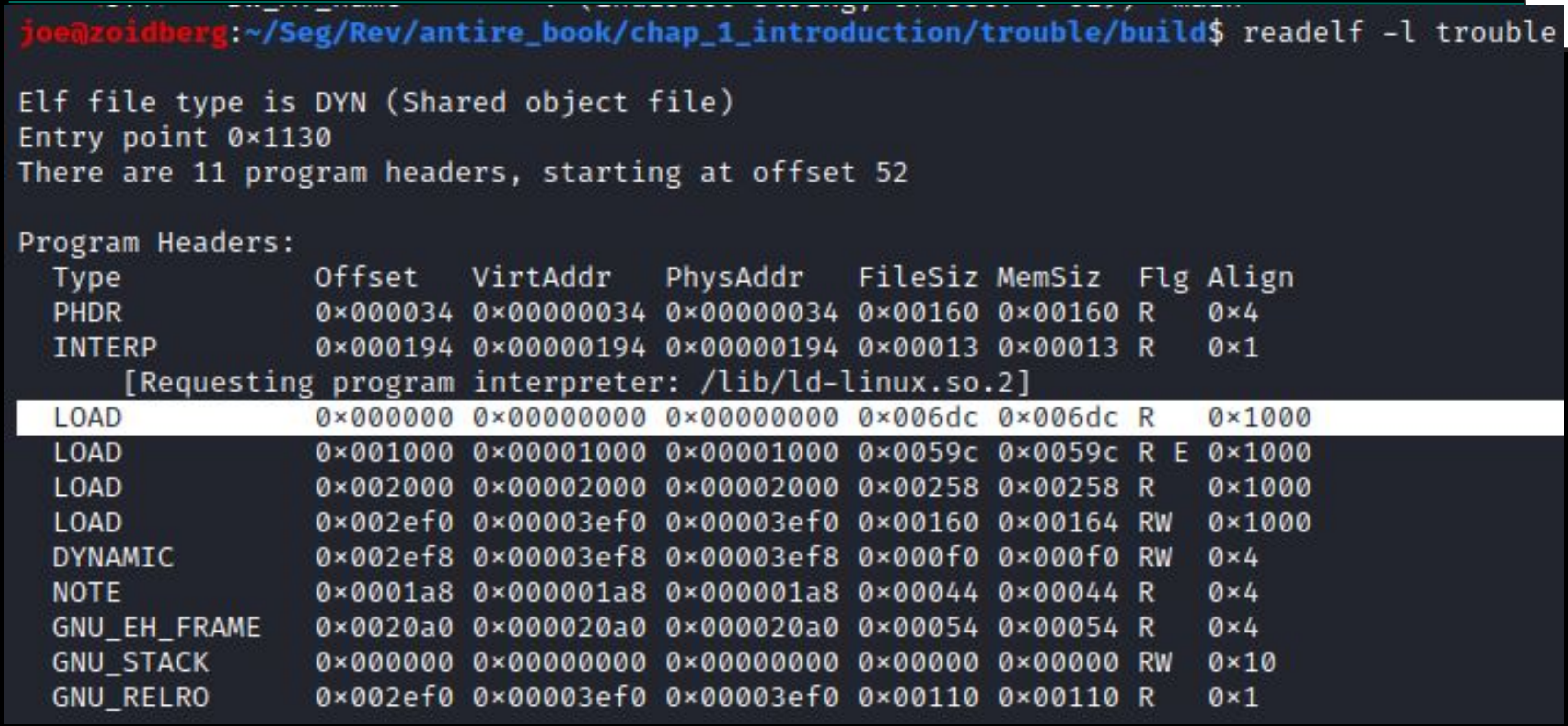


```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ objdump --dwarf=info trouble | grep --color  -C 8 s_password
    <5dc>   DW_AT_sibling     : <0x5e7>
 <2><5e0>: Abbrev Number: 13 (DW_TAG_subrange_type)
    <5e1>   DW_AT_type        : <0x31>
    <5e5>   DW_AT_upper_bound : 32
 <2><5e6>: Abbrev Number: 0
 <1><5e7>: Abbrev Number: 4 (DW_TAG_const_type)
    <5e8>   DW_AT_type        : <0x5d7>
 <1><5ec>: Abbrev Number: 24 (DW_TAG_variable)
    <5ed>   DW_AT_name        : (indirect string, offset: 0x2b3): s_password
    <5f1>   DW_AT_decl_file   : 1
    <5f2>   DW_AT_decl_line   : 11
    <5f3>   DW_AT_decl_column : 19
    <5f4>   DW_AT_type        : <0x5e7>
    <5f8>   DW_AT_location    : 5 byte block: 3 20 20 0 0       (DW_OP_addr: 2020)
 <1><5fe>: Abbrev Number: 25 (DW_TAG_subprogram)
    <5ff>   DW_AT_external    : 1
    <5ff>   DW_AT_name        : (indirect string, offset: 0x619): main
```



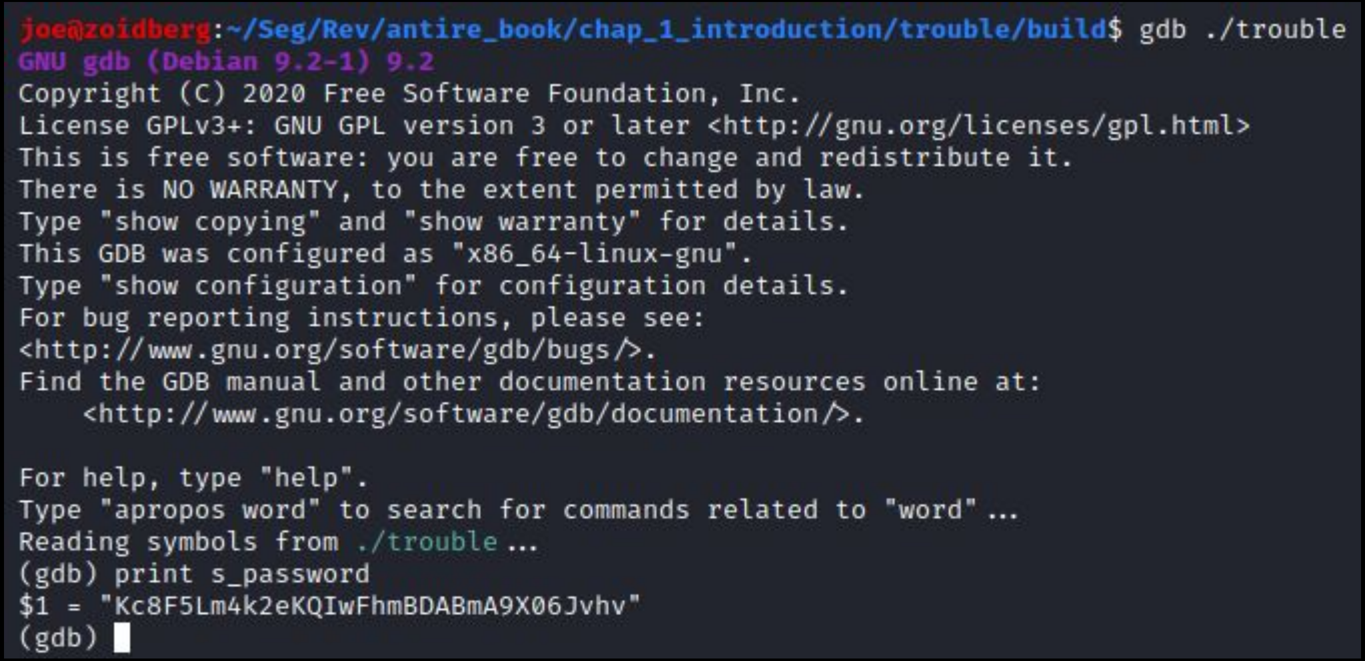
```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ hexdump -C -s 0x2020 -n 40 trouble
00002020  7a 5a 4e 4a 67 6d 41 6a  42 74 47 67 70 39 7a 50  |zZNJgmAjBtGgp9zP|
00002030  4f 54 44 49 69 7a 5a 32  46 4b 50 56 62 69 64 68  |OTDIizZ2FKPVbidh|
00002040  00 46 61 69 6c 65 64 20                           |.Failed |
00002048
```

Reversing



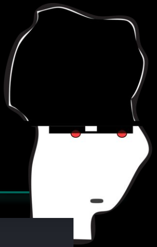
```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ readelf -l trouble

Elf file type is DYN (Shared object file)
Entry point 0x1130
There are 11 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x00000034 0x00000034 0x00160 0x00160 R   0x4
  INTERP         0x000194 0x00000194 0x00000194 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x00000000 0x00000000 0x006dc 0x006dc R   0x1000
  LOAD           0x001000 0x00001000 0x00001000 0x0059c 0x0059c R E 0x1000
  LOAD           0x002000 0x00002000 0x00002000 0x00258 0x00258 R   0x1000
  LOAD           0x002ef0 0x00003ef0 0x00003ef0 0x00160 0x00164 RW  0x1000
  DYNAMIC        0x002ef8 0x00003ef8 0x00003ef8 0x000f0 0x000f0 RW  0x4
  NOTE           0x0001a8 0x000001a8 0x000001a8 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x0020a0 0x000020a0 0x000020a0 0x00054 0x00054 R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RW  0x10
  GNU_RELRO      0x002ef0 0x00003ef0 0x00003ef0 0x00110 0x00110 R   0x1
```

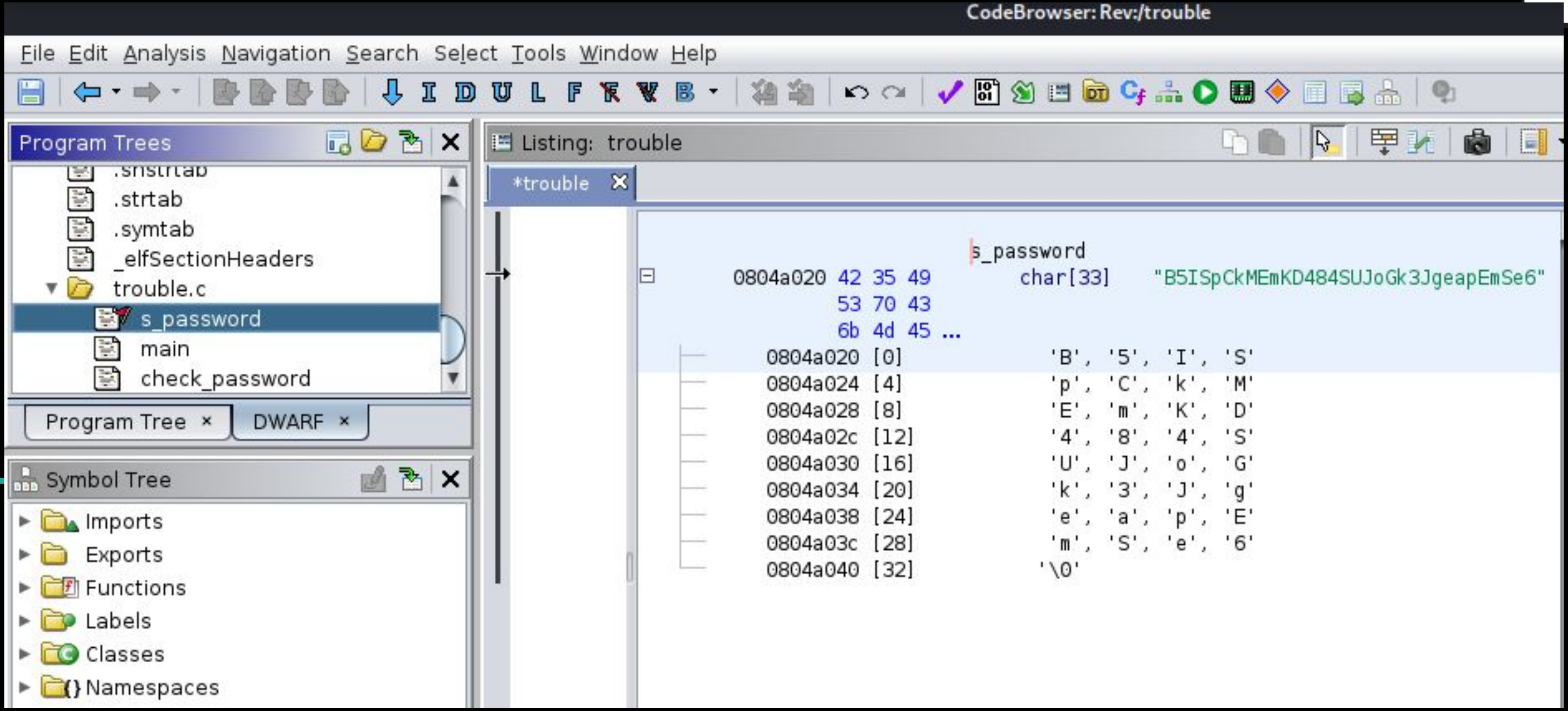
Reversing (con gdb)



```
joe@zoidberg:~/Seg/Rev/antire_book/chap_1_introduction/trouble/build$ gdb ./trouble
GNU gdb (Debian 9.2-1) 9.2
Copyright (C) 2020 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:
<http://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 ./trouble...
(gdb) print s_password
$1 = "Kc8F5Lm4k2eKQIwFhmBDABmA9X06Jvhv"
(gdb) 
```

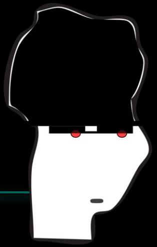
Reversing (con Ghidra o IDA)

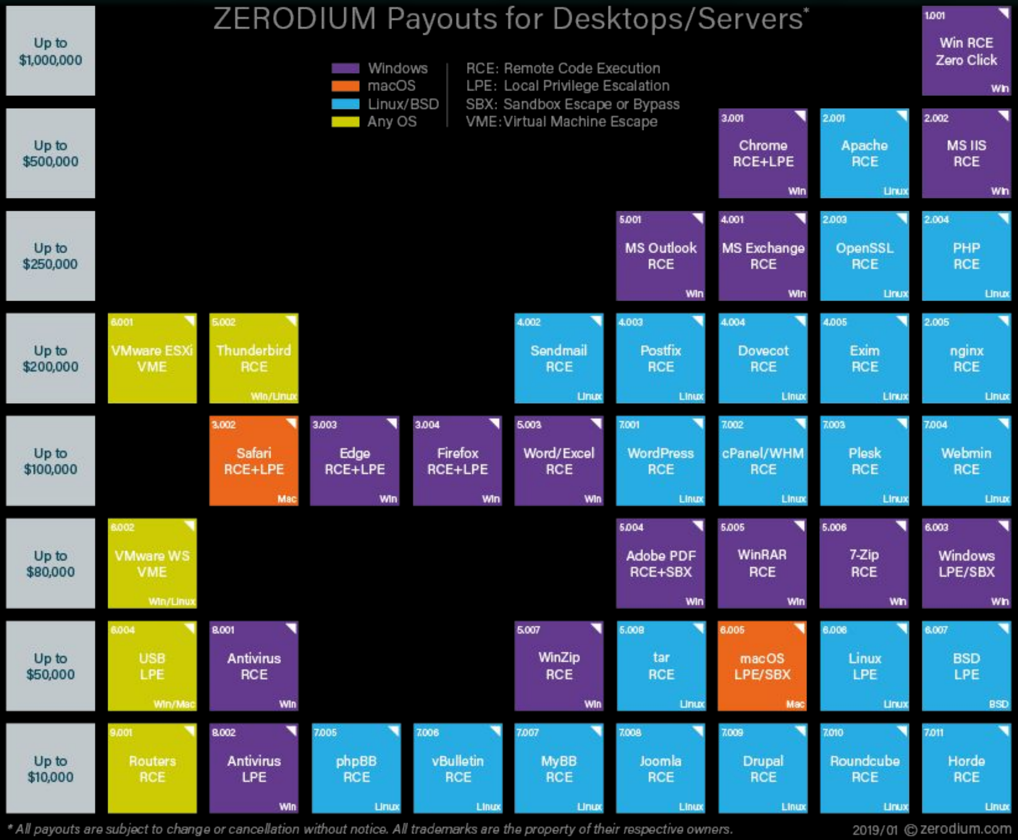




```
                         s_password
0804a020 42 35 49          char[33]    "B5ISpCkMEmKD484SUJoGk3JgeapEmSe6"
         53 70 43
         6b 4d 45 ...
   0804a020 [0]                 'B', '5', 'I', 'S'
   0804a024 [4]                 'p', 'C', 'k', 'M'
   0804a028 [8]                 'E', 'm', 'K', 'D'
   0804a02c [12]                '4', '8', '4', 'S'
   0804a030 [16]                'U', 'J', 'o', 'G'
   0804a034 [20]                'k', '3', 'J', 'g'
   0804a038 [24]                'e', 'a', 'p', 'E'
   0804a03c [28]                'm', 'S', 'e', '6'
   0804a040 [32]               '\0'
```

The Market for An 0day





ZERODIUM Payouts for Desktops/Servers*

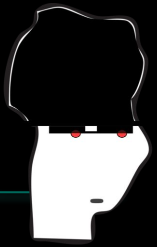
🟪 Windows | RCE: Remote Code Execution
🟧 macOS | LPE: Local Privilege Escalation
🟦 Linux/BSD | SBX: Sandbox Escape or Bypass
🟨 Any OS | VME: Virtual Machine Escape

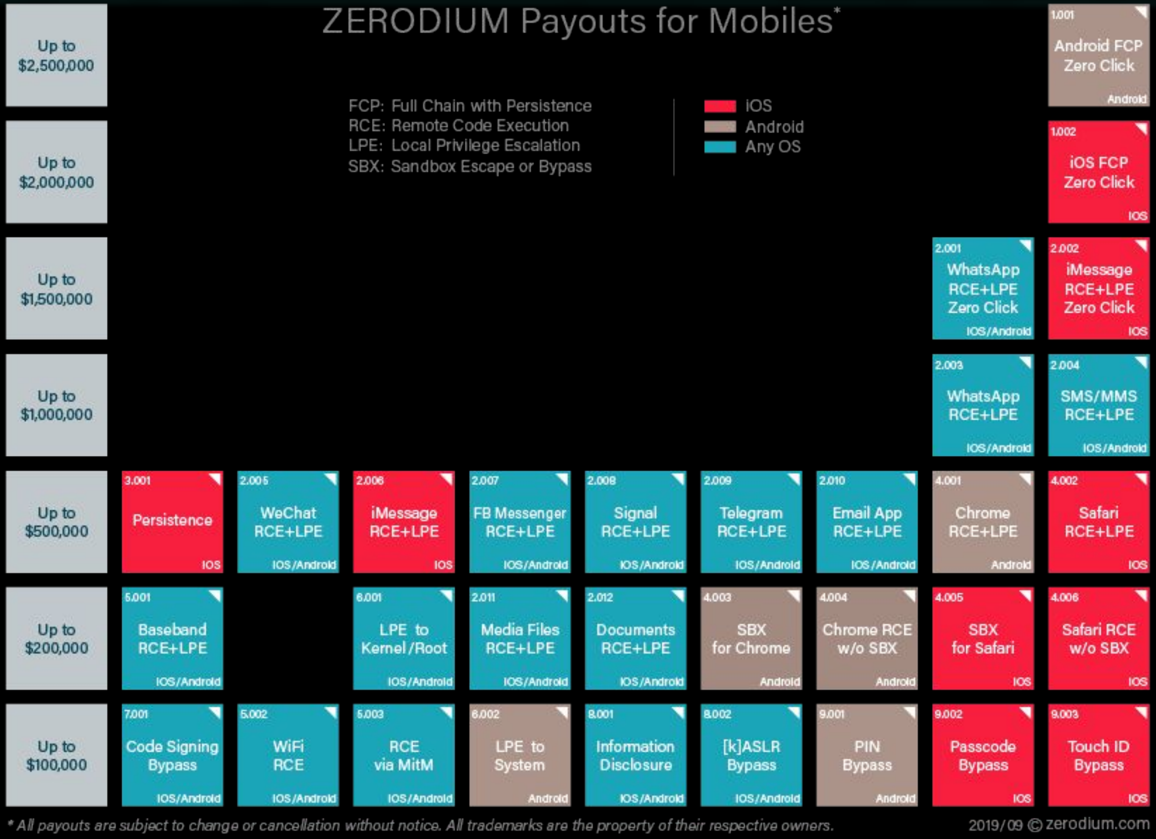
Up to $1,000,000									1.001 Win RCE Zero Click Win
Up to $500,000							3.001 Chrome RCE+LPE Win	2.001 Apache RCE Linux	2.002 MS IIS RCE Win
Up to $250,000						5.001 MS Outlook RCE Win	4.001 MS Exchange RCE Win	2.003 OpenSSL RCE Linux	2.004 PHP RCE Linux
Up to $200,000	6.001 VMware ESXi VME	5.002 Thunderbird RCE Win/Linux			4.002 Sendmail RCE Linux	4.003 Postfix RCE Linux	4.004 Dovecot RCE Linux	4.005 Exim RCE Linux	2.005 nginx RCE Linux
Up to $100,000		3.002 Safari RCE+LPE Mac	3.003 Edge RCE+LPE Win	3.004 Firefox RCE+LPE Win	5.003 Word/Excel RCE Win	7.001 WordPress RCE Linux	7.002 cPanel/WHM RCE Linux	7.003 Plesk RCE Linux	7.004 Webmin RCE Linux
Up to $80,000	6.002 VMware WS VME Win/Linux					5.004 Adobe PDF RCE+SBX Win	5.005 WinRAR RCE Win	5.006 7-Zip RCE Win	6.003 Windows LPE/SBX Win
Up to $50,000	6.004 USB LPE Win/Mac	8.001 Antivirus RCE Win			5.007 WinZip RCE Win	5.008 tar RCE Linux	6.005 macOS LPE/SBX Mac	6.006 Linux LPE Linux	6.007 BSD LPE BSD
Up to $10,000	9.001 Routers RCE	8.002 Antivirus LPE Win	7.005 phpBB RCE Linux	7.006 vBulletin RCE Linux	7.007 MyBB RCE Linux	7.008 Joomla RCE Linux	7.009 Drupal RCE Linux	7.010 Roundcube RCE Linux	7.011 Horde RCE Linux

* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/01 © zerodium.com

The Market for An 0day





ZERODIUM Payouts for Mobiles*

FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

🟥 iOS
▨ Android
🟦 Any OS

Up to $2,500,000									1.001 Android FCP Zero Click (Android)
Up to $2,000,000									1.002 iOS FCP Zero Click (iOS)
Up to $1,500,000								2.001 WhatsApp RCE+LPE Zero Click (iOS/Android)	2.002 iMessage RCE+LPE Zero Click (iOS)
Up to $1,000,000								2.003 WhatsApp RCE+LPE (iOS/Android)	2.004 SMS/MMS RCE+LPE (iOS/Android)
Up to $500,000	3.001 Persistence (iOS)	2.005 WeChat RCE+LPE (iOS/Android)	2.006 iMessage RCE+LPE (iOS)	2.007 FB Messenger RCE+LPE (iOS/Android)	2.008 Signal RCE+LPE (iOS/Android)	2.009 Telegram RCE+LPE (iOS/Android)	2.010 Email App RCE+LPE (iOS/Android)	4.001 Chrome RCE+LPE (Android)	4.002 Safari RCE+LPE (iOS)
Up to $200,000	5.001 Baseband RCE+LPE (iOS/Android)		6.001 LPE to Kernel/Root (iOS/Android)	2.011 Media Files RCE+LPE (iOS/Android)	2.012 Documents RCE+LPE (iOS/Android)	4.003 SBX for Chrome (Android)	4.004 Chrome RCE w/o SBX (Android)	4.005 SBX for Safari (iOS)	4.006 Safari RCE w/o SBX (iOS)
Up to $100,000	7.001 Code Signing Bypass (iOS/Android)	5.002 WiFi RCE (iOS/Android)	5.003 RCE via MitM (iOS/Android)	6.002 LPE to System (Android)	8.001 Information Disclosure (iOS/Android)	8.002 [k]ASLR Bypass (iOS/Android)	9.001 PIN Bypass (Android)	9.002 Passcode Bypass (iOS)	9.003 Touch ID Bypass (iOS)

* All payouts are subject to change or cancellation without notice. All trademarks are the property of their respective owners.

2019/09 © zerodium.com

Fuente: Zerodium