**Final Report: Remote Shell**

**Submitted to:**

Inst. Yinqian Zhang

**Created by:**

CodeRobot

Yihang Du

Mike Zhang

Chenhui Pan

CSE 2431

The Ohio State University

Columbus, OH

30 November 2016

**Introduction**

The purpose of this project is to design a remote shell using C on Linux operating system. The functionality of this program enable remote user to execute commands from his own computer to manipulate other computers across the network. This project is complete by three members in the team CodeRobot. There are three major components in this project, including network transmission, malicious activity and automatic startup of the malware. Each team member is responsible for each component.

**Description**

The remote shell is command line computer program to execute commands as remote user on the target computer and transmit data information via network. Network communication, malicious activity and automatic startup are three essential parts in this project.

The network part is to transfer data using TCP protocol. On the client side, the user executes commands on terminal and send commands to server. Once the server receives the commands, it executes and collects information and send information back to client. To achieve this functionality, the library <sys/socket.h> is used to do socket programming in this project.

Malicious activity is another critical part in the remote shell. It includes all malicious function that user decides to do on target computer. “ls” is a Linux terminal command to list all the files in the current directory. “rm filename” is used to delete a file. “chmod” is used to change the accessibility of file. The file could be read or write by local user only, but when we change the mode of file, the file could be accessed by other users to read and write.

Automatic component is a bonus part for this project. The program initializes under the root directory, and the second time when the target computer reboots, the program would automatically start in the background. The user could not detect the program activity by simply calling “ps” on the terminal, which lowers the probability of being detected.

**Evaluation**

This project involves three main parts to complete: network communication, malicious functions and automatic startup.

For the network part, the difficult part for this project is to record the result from the terminal, transmit it from server to client, and then display it on the client side terminal. It requires to use popen() and pipe() functions in standard C library to achieve. For each command, there is a delay for transmission via network and the team needs to figure out how to speed up the whole process. For the file transfer, the team struggled with how to identified the file name and continuous file transfer. After each file transfer the application is able to execute other commands given by client side.

For malicious functions, once the team figures out how to call the system() in a correct way, the server will execute the commands as expected. Two of the difficult commands to figure out is ls and cd, it doesn't works directly from system() system call. The team is able to find a way to implement it.

For the automatic startup, the hardest part is how to locate the absolute path for the remote shell. When system start from shutdown and rebooting, the Linux system execute different files in different locations, it is quite difficult to find the way to make program run both from startup and reboot. The team member needs to ensure that each time when the program runs at the background, the program has the root access for the whole system.

**Discussion**

In this project, the team debates on what format of transmission would be the best way to transmit data between client and server. The major way to pass command is passing string via network. The commands are passed using a char array. On the server side, once the server receives the command, it parses the commands and calls system calls. After executing the command, the team debate the way to send data back to client. The first way is to record all data into a txt file and pass the file back to client. This method may take a long time and generated errors. The second way is to record the information in a string buffer array and send it back to the terminal of client. Compared to the first option, the second one is much complicated. It requires to record the data from server terminal, send it back and display the result on the client terminal. The scope of this method is beyond the course since it uses pipe() and popen() function in C.

The issue occurs when the team use remote terminal to test the project. In the local, the remote shell works well for all commands. However, when team switches the server to Virtual Pirate Server (VPS)terminal, the ‘ls’ commands become malfunctioned. It fails to transmit all the file names from VPS to local terminal. If the user executes ‘ls’ commands for consecutive time, the whole data transmit from remote terminal becomes chaos. In the README file, the team is able to indicate how to test the malware in VPS and locally, it will help to avoid the problem.

**Conclusion**

Overall, this project is challenging and time consuming. It requires the team to be familiar with socket programming, execute system calls, and script file to complete the optional component. t requires the team to be familiar with socket programming, execute system calls, and script file to complete the optional component. The hardest part for this project is the transmission part using socket programming. Students need to be familiar with the socket program and pipe function to fully achieve the functionality of this remote shell.

**Contribution**

Mike Zhang: design functions for malicious activities on the server side, use popen() to transfer the result from server side to client side, edit introduction and description part in the final report.

Yihang Du: Write three scripts to make the malware be able to auto start when system is booting or rebooting, the target computer will not get any form of output after the computer is rebooting. Also, find a VPS for demo and testing, and write a readme for the program.

Chenhui Pan: Build the TCP connection for the server and clients. Make sure client can send commands to the server, in addition, the client can request the files and received the files from the server.