Design Document

for

Packet-Sniffing Backdoor

<Assignment 2>

Version 1.0

Prepared by Wesley Yue

October 15, 2012

Table of Contents

1. Introduction

1.1 Abstract

This assignment puts the basic components of a packet sniffing backdoor and the libpcap library together to implement a complete and working Linux backdoor. For this assignment, Ruby was used instead of C to put these pieces together. The PacketFu package is the main component to get the packet sniffing backdoor to work. This library is a mid-level packet manipulation library for Ruby. However, using the PacketFu package alone will not run if the missing packages are not installed. For Ruby, PacketFu, pcaprub, rspec, sdoc, jruby-openssl, daemons, and trollop will need to be installed. As for Linux, Ruby*, pcap-dev*, ruby-dev*, libpcap*, and libpcap-dev* needs to be installed. The PacketFu package is one piece part of the Linux backdoor, but there are other pieces that were used to implement a complete and working Linux backdoor.

One of the other pieces would be the SHA1 encryption, which was used to encrypt the TCP payload into non-human readable form. Moreover, another piece would be the backdoor camouflaging itself and running in daemon mode. However, if one decides to run in daemon mode, one cannot specify arguments for the server. As a result, the code to run it daemon mode has been commented out. More importantly, a backdoor is supposed to be able to read the client's command and authenticate them, which this Linux backdoor is capable of. Other than that, all the testing results are in the WY_Testing_Assignment2 document. For the testing results, both the server and client have the netfilter of "iptables –A INPUT –j DROP".

2. Design

2.1 Pseudo-Code

server.rb pseudo-code

```
Define global variables;
def set_process_name name method
        Check if OS is Linux
        Change process name
end
```

```
def sniff(iface) method
        Capture packet with filter
        While loop
                For each captured packet do
                        Parse it into a structure
                        If the captured packet is TCP and IP
                                Read TCP/IP header contents
                                Store the contents that is needed into packet_info variable
                                begin
                                        Decrypt
                                rescue
                                        Packet_info is null
                                end
                                If Packet_info is not null
                                        Encrypt the TCP payload data
                                        Use scan method
                                end
                        end

                end
        end
end

def scan method
        Read the parsed structure
        Send raw packets
end
```

client.rb pseudo-code

```
def scan method
        Read the parsed structure
        Send raw packets
end
```

```
def sniff(iface) method
        Capture packet with filter
        For each captured packet do
                Parse it into a structure
                If the captured packet is TCP and IP
                        Read TCP/IP header contents
                        Decrypt the TCP payload
                        Puts the decrypted TCP payload
                end
        end
end
```

2.2 Command-Line Parameters

Server Command Line Parameters

-iface <interface> - The interface card to sniff packets from. (Default: em1)
-process <name> - The camouflaged name for the process. (Default: [kworker/1:1])
-daddr <ipdaddr> - The client's IP address. (Default: 192.168.0.22)
-dport <dst port> - IP dst port you want data to go to. (Default: 4040)

Client Command Line Parameters

-iface <interface> - The interface card to sniff packets from. (Default: em1)
-cmd <command> - The command line to send to the backdoor to execute. (Default: ls)
-daddr <ipdaddr> - The server's IP address. (Default: 192.168.0.21)
-dport <dst port> - IP dst port you want data to go to. (Default: 4040)

3. Summary

The server.rb and client.rb has put the basic components of a packet sniffing backdoor and the libpcap library together to implement a complete and working Linux backdoor. However, there are several other pieces that were used to put together the Linux backdoor. Those pieces are SHA1 encryption, camouflaging itself and running in daemon mode, executing commands and sending results back, and authenticating the client. Moreover, the scripts were tested and both the server and client have the netfilter of “iptables –A INPUT –j DROP”.

The accompanying DVD will include the complete source code, and pcaps files for confirmation of design and evidence of test results. Also, the DVD will include this and testing document.

4. Appendix

/documents/ directory:

WY_DesignDocument_Assignment2.docx	(docx version of the design document)
WY_DesignDocument_Assignment2.pdf	(pdf version of the design document)
WY_Testing_Assignment2.docx	(docx version of the testing file for the backdoor)
WY_Testing_Assignment2.pdf	(pdf version of the testing file for the backdoor)

/pcap/ directory:

Test_Case_1_Authenticate.pcap	(test case #1 Wireshark packet capture file)
Test_Case_2_EncryptTcp.pcap	(test case #2 Wireshark packet capture file)
Test_Case_3_Spoofed.pcap	(test case #3 Wireshark packet capture file)
Test_Case_4_InvalidCmd.pcap	(test case #4 Wireshark packet capture file)

/src/ directory:

client.rb	(Ruby client's source file for this assignment)
server.rb	(Ruby server's source file for this assignment)