COMP 8505 Assignment 2

Linux Backdoor

Xinghua Wei & Tao Yuan

A00978597 & A00952007

# Introduction

In this assignment, I will implement a Linux backdoor. This backdoor will allow a user to remote control, execute a command, and receive the command result.

# Constraints

There are few constraints I need to follow:

* The backdoor must be hidden. So, users can hardly identify it when looking at the process table
* The backdoor should only receive and send packets that are specified.
* The backdoor must interpret commands sent to it and execute them, and send the results back.
* The backdoor commands sent to it and results back should be encrypted

# Design

## Tools

To satisfy all the constraints with Python, the following tools will be used:

* Python 3.8
* Scapy
  + Scapy has a sniff function that is great for getting packets. Sniff has an argument prn that allows users to pass a function that executes with each packet sniffed. Also, unlike traditional TCP\UDP connections, sniff does not need to specify buffer size, which gives me the flexibility to control the packets.
* Setproctitle
  + The setproctitle module allows a process to change its title as displayed by system tools such as ps. Therefore, anyone who is looking at the process table can hardly identify the backdoor process.
* Subprocess
  + Subprocess module allows users to spawn new processes, connect to their input, output and error pipes
  + Popen function takes arguments to set up the new process so the parent can communicate with it via pipes.
  + Stdout and Stderr will be pipes to open and will be where the command executes and where the results store
* Crypto
  + Crypto is a python cryptography toolkit
  + AES encryption will be used in this assignment

## Detail Design

### Client (Attacker)

On the client-side, first users need to gather a few pieces of information:

* Target IP address(Destination IP address)
* Target port (Destination port and source port)
  + I will use destination port 8000 and source port 8505 as default
  + UDP will be used as the default
* Attacker IP address( Source IP address)
* The backdoor title which camouflages the backdoor
* The command to send to execute on the victim machine

Once users have all the information, the program will start processing this information and send it to a victim machine.

1. Concatenate the command and process title to one packet
2. Encrypt the packet with AES and UTF-8
3. Send the packet so the victim machine can sniff the packet
4. Sniff the packet in the victim machine using a filter. So, only the information we want would be sniffed.
5. Load the sniffed raw packets and decrypt them into readable information

Diagram

Description automatically generated

### Server(Victim)

On the server-side, the default protocol, destination port and the source port will be set in this assignment.

* Protocol: UDP
* Destination port: 8000
* Source Port: 8505

Once the program starts, the following actions will be done:

1. Load the packet with sniff
2. Parse and decrypt the packet to retrieve the command and the title
3. Use subprocess and Popen functions to create pips to execute the command
4. Camouflage the process with the title
5. Read results from stdout and stderr
6. Encrypt the result with AES
7. Send packet so the attacker machine can sniff the packet

Diagram

Description automatically generated

### Cipher

For cipher, I will use AES with CFB 8 mode (8-bit cipher feedback mode). The following information is required:

* Key: fixed data block size of 16 bytes
* IV: Initialization Vector is used by several modes to randomize the encryption and produce distinct ciphertexts even if the same plaintext is encrypted multiple times. For CFB mode, it must be 16 bytes long.

Encryption:

1. Encode the key and the initialization vector with UTF-8 because AES in Crypto cannot take a string.
2. Encrypt the message and return the IV and the message

Decryption:

1. Encode the key
2. Parse the message that the former 16 bytes will be the IV and the rest will be the message
3. Decrypt the message with the key and the IV
4. Decode the message again with UTF-8 because it is still in bytes format

Diagram

Description automatically generated

# Test

## Test Case 1 – The backdoor title (Camouflage the process)

Test case one will demonstrate camouflaging the process title to the title users specified.

* Destination IP: 10.0.0.33
* Source IP: 10.0.0.123
* Camouflaged title: admin
* Command: ls

### Attacker

Graphical user interface, application

Description automatically generated

Users specify destination IP, source IP, process title and commands to send.

Text

Description automatically generated

The result is back to the attack machine with lists of files in the backdoor file directory.

A picture containing graphical user interface

Description automatically generated



The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted.

Text

Description automatically generated

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

### Victim

Text

Description automatically generated

On the victim machine, I printed out the information just to make sure everything is working. So, it shows the command and the title received, and the encrypted message that will be sent back. Because the encoded cipher could be very long, I only print the first 120 bytes.

A picture containing meter, street

Description automatically generated

This image shows that the program successfully camouflaged the process title to the received title “admin.”

This test means the function to camouflage the process is a success.

## Test Case 2 – Encryption

Test case one will demonstrate encryption data transmission, including encrypted command and encrypted result.

* Destination IP: 10.0.0.33
* Source IP: 10.0.0.123
* Camouflaged title: root
* Command: uname -a
* Encryption

### Attacker

Graphical user interface, application

Description automatically generated

Users specify destination IP, source IP, process title and commands to send.



The result is back to the attack machine with the system information of the victim machine.

Graphical user interface

Description automatically generated



The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted.

The encrypted also printed out in the console.



The data was encrypted using AES in bytes and IV combined with it.

Text

Description automatically generated

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

### Victim

Text

Description automatically generated

On the victim machine, I printed out the information to make sure everything is working. So, it shows the command and the title received, and the encrypted message sent back. Because the encoded cipher could be very long, I only print the first 120 bytes.

This test means the data encryption function is a success.

## Test Case 3 – List files

Test case one will demonstrate list files in the current directory remotely to the victim machine.

* Destination IP: 10.0.0.33
* Source IP: 10.0.0.123
* Camouflaged title: admin
* Command: ls

### Attacker

Graphical user interface, application

Description automatically generated

Users specify destination IP, source IP, process title and list files “ls” commands to send.

Text

Description automatically generated

The result is back to the attack machine with lists of files in the backdoor file directory.

A picture containing graphical user interface

Description automatically generated



The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted.

Text

Description automatically generated

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

### Victim

Text

Description automatically generated

It shows the command and the title received, and the encrypted message that will be sent back. Because the encoded cipher could be very long, I only print the first 120 bytes.

This test means the list file function is a success.

## Test Case 4 – Ifconfig

Test case one will demonstrate Ifconfig command in the current directory remotely to the victim machine.

* Destination IP: 10.0.0.33
* Source IP: 10.0.0.123
* Camouflaged title: sad
* Command: ifconfig

### Attacker

Graphical user interface, text, application

Description automatically generated

Users specify destination IP, source IP, process title and Ifconfig commands to send.

Text

Description automatically generated

The result is back to the attack machine with Ifconfig of the victim machine. It shows victim machine IP address, network card names, IPv6 address, flags and etc.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

The Wireshark capture shows that the attacker sends the command to the victim and get the result. The result and the command are encrypted. Because the Ifconfig result is large, so the encryption will be extensive too.

Text

Description automatically generated

Also, the print out in the console shows the encrypted message being sent and the decrypted result.

### Victim

Text

Description automatically generated

It shows the command and the title received, and the encrypted message that will be sent back. Because the encoded cipher could be very long, I only print the first 120 bytes. The following is the complete encrypted data.

This test means the Ifconfig command is a success.

## Test Case 5 – Create, Execute and Remove an Executable

Test case one will demonstrate list files in the current directory remotely to the victim machine.

* Destination IP: 10.0.0.33
* Source IP: 10.0.0.123
* Camouflaged title: sad
* Command: echo / bash / less / rm

### Attacker

#### Create Executable

Users specify destination IP, source IP, process title and Ifconfig commands to send. First, I will send “echo echo hello > hi.sh” to create a executable bash file.

Graphical user interface, text, application

Description automatically generated

The console shows the plaintext command and encrypted command.



#### Validate The Executable

Next, I need to check if the file has been created. I use “ls” command to list all files in that directory.

Graphical user interface, text

Description automatically generated

The result shows that the hi.sh has been created.

A picture containing text

Description automatically generated

The console shows the plaintext command and encrypted command.



Then I want to use less command to check the content of the executable.

A screenshot of a cell phone

Description automatically generated

The result shows that the executable is what I expected.



The console shows the plaintext command and encrypted command.



#### Execute The Executable

Then I need to run the hi.sh remotely. I used bash hi.sh command.

Graphical user interface, text, application

Description automatically generated

The result shows that it echoed hello in the console.



The console shows the plaintext command and encrypted command.



#### Remove The Executable

Now, after executing it, I want to remove it from the victim machine. I use rm hi.sh command.

A screenshot of a cell phone

Description automatically generated

The console shows the plaintext command and encrypted command.



Finally, I use ls command to check if the hi.sh was deleted or not.

A screenshot of a cell phone

Description automatically generated

The result shows that hi.sh was removed.

A picture containing table

Description automatically generated

In total, I used six commands for all the steps:

1. Echo echo hello > hi.sh
2. Ls
3. Less hi.sh
4. Bash hi.sh
5. Rm hi.sh
6. Ls

Each step will generate 2 UDP requests. So in total, there will be 12 UDP requests. (ignore the ICMP bad requests.)

Table

Description automatically generated

Wireshark also shows that each command and transmission is encrypted.

Graphical user interface, text, application

Description automatically generated

### Victim

In total, I used six commands for all the steps, so the victim machine will react six times.

#### Echo echo hello > hi.sh

Text

Description automatically generated

It shows the command and the title received, and the encrypted message that will send back.

#### Ls

Text

Description automatically generated

It shows the list file command, and the title received, and the encrypted message that will send back.

#### Less hi.sh

Text

Description automatically generated

It shows the “less hi.sh” command, the title received, and the encrypted message sent back.

#### Bash hi.sh

Text

Description automatically generated

It shows the “bash hi.sh” command, the title received, and the encrypted message sent back.

#### Rm hi.sh

Text

Description automatically generated

It shows the remove command, the title received, and the encrypted message sent back.

#### Ls

Text

Description automatically generated

It shows the list file command, the title received, and the encrypted message sent back.

This test means the function to create, execute and remove an executable is a success.