Rootkit Design

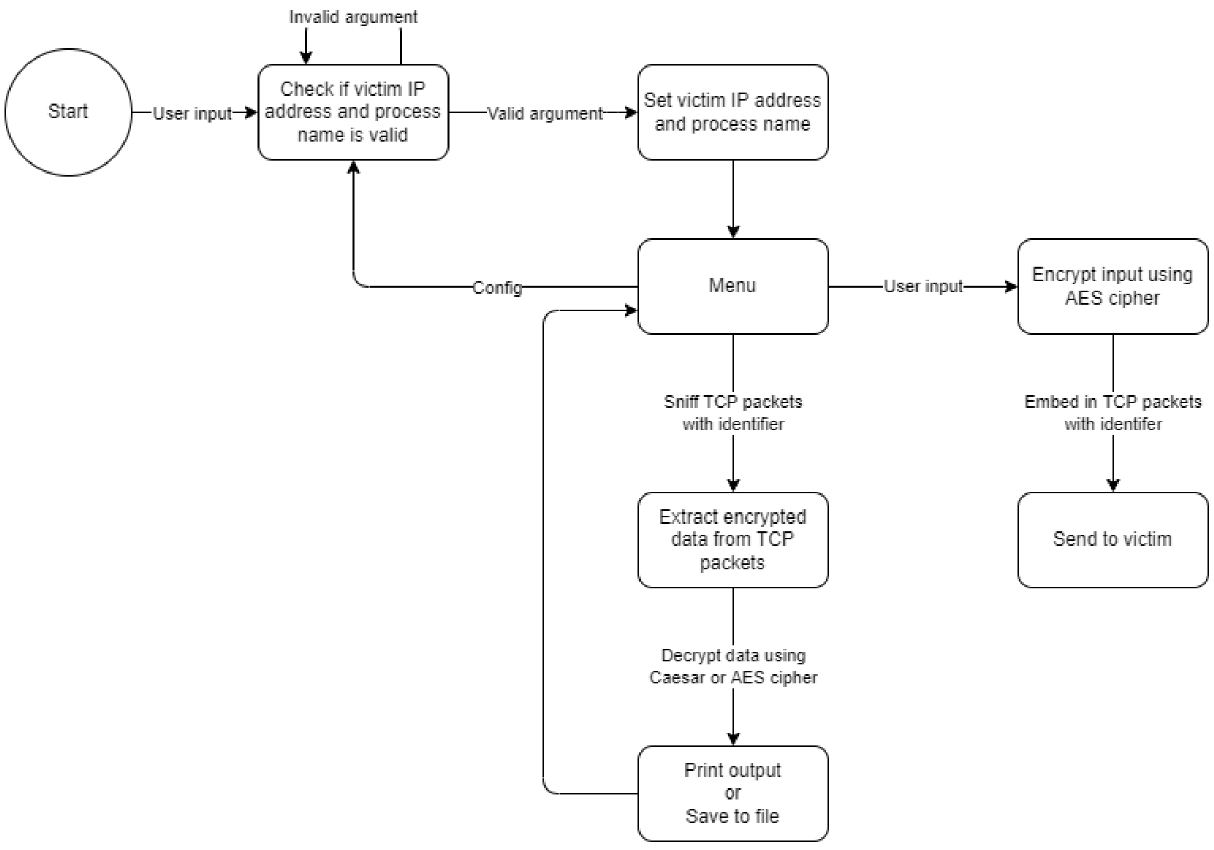
Eunsaem Lee

December 6th, 2022

Table of Contents

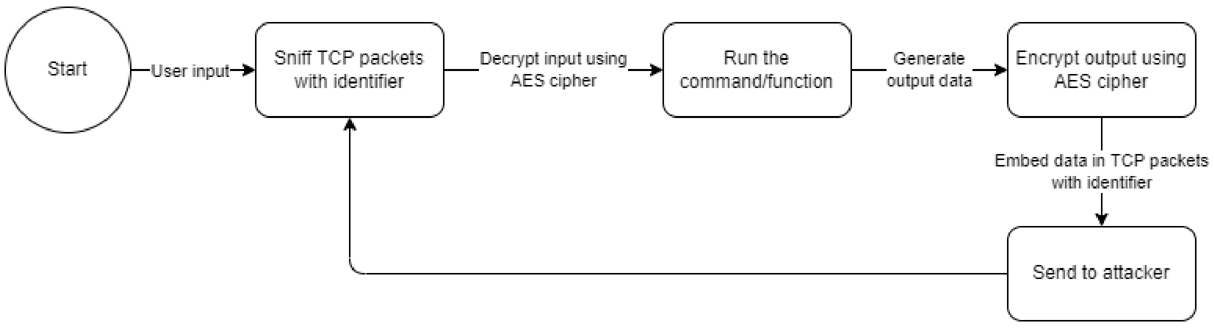
Finite State Machine

attacker.py



Invalid argument
Start
User input
Check if victim IP address and process name is valid
Valid argument
Set victim IP address and process name
Config
Menu
User input
Encrypt input using AES cipher
Sniff TCP packets with identifier
Embed in TCP packets with identifer
Extract encrypted data from TCP packets
Send to victim
Decrypt data using Caesar or AES cipher
Print output or Save to file

victim.py



Start
User input
Sniff TCP packets with identifier
Decrypt input using AES cipher
Run the command/function
Generate output data
Encrypt output using AES cipher
Embed data in TCP packets with identifier
Send to attacker

Functions

main (attacker.py)

Purpose

Command line prompt with a helper function that displays the application's switches. Determines whether the application is running on root and sniffs TCP packets with the correct identifier. If not, prints an error message.

Parameters

The state object to initialize.

Return

Type	Next State
Success	`KEYLOGGER_SNIFF, TCONFIRM_SNIFF, FCONFIRM_SNIFF, DCONFIRM_SNIFF, DIR_SNIFF, or SHELL_SCRIPT`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Check if the victim IP address is valid
     If not, print an ERROR message and re-ask for the victim
IP address.

Generate and set the port number to a random integer number
between 1025 and 65535.

Check if the process name is valid
     If it is, set the process name to the input.
     If not, set the process name to default "abc".

Check if the application is running on root
     If not, print an ERROR message and exit the program.

Create a TCP packet with an identifier, encrypt the menu
command input using AES cipher with an encryption key and salt
value, and send it to the victim.
```

Based on the selection, the program should:

1. start the keylogger
2. stop the keylogger
3. transfer a file from the victim to the attacker
4. start watching a file for changes. When the file changes, transfer it to the attacker.
5. stop watching a file
6. start watching a directory for changes. When a file is created or modified in the directory or its subdirectories, transfer it to the attacker.
7. stop watching a directory
8. run shell script
9. change process name
10. exit

keylogger_sniff (attacker.py)

Purpose

Receives the keylogging data from the victim by parsing packet object from sniffing the network traffic for TCP packet(s) with the correct identifier. It extracts and decrypts the embedded data in the source port using Caesar cipher with a set key value and appends the data to the log files.

Parameters

Packet object sniffed with Scapy.

Return

Type	Next State
Success	`RECV_KEYLOGGER, CHECK_KEYLOGGER`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the keylogger sniffer by
the identifier.
     If not, ignore packet.

Decrypt the extracted keyboard input using Caesar cipher and a
set key value.

Create and write the encrypted and decrypted keyboard input
from the victim into log files.
```

tconfirm_sniff (attacker.py)

Purpose

Checks if the specified file path exists on the victim machine.

Parameters

File path input from the attacker.

Return

Type	Next State
Success	`TRANSFER_SNIFF`
Failure	`MENU`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the tconfirm sniff.
     If not, ignore packet.

If the source port is 7000,
     the file exists on the victim machine.

If the source port is 8000,
     the file does NOT exist on the victim machine.

If the file exists,
     begin sniffing for TCP packets with the correct
identifier.

If the file does NOT exist,
     redirect back to the menu.
```

transfer_sniff (attacker.py)

Purpose

Receives the transfer file data from the victim by parsing packet object from sniffing the network traffic for TCP packet(s) with the correct identifier. It extracts and decrypts the embedded data using AES cipher with encryption key and salt value and creates the specified file.

Parameters

Packet object sniffed with Scapy.

Return

Type	Next State
Success	`MENU`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the transfer sniff.
     If not, ignore packet.

Decrypt the extracted transfer file data using AES cipher with
an encryption key and salt value.

Create the received transfer file in the local set directory.
```

fconfirm_sniff (attacker.py)

Purpose

Checks if the specified file path exists on the victim machine.

Parameters

File path input from the attacker.

Return

Type	Next State
Success	`FILE_SNIFF`
Failure	`MENU`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the fconfirm sniff.
     If not, ignore packet.

If the source port is 7000,
     the file exists on the victim machine.

If the source port is 8000,
     the file does NOT exist on the victim machine.

If the file exists,
     begin sniffing for TCP packets with the correct
identifier.

If the file does NOT exist,
     redirect back to the menu.
```

file_sniff (attacker.py)

Purpose

Receives the modified file data from the victim by parsing packet object from sniffing the network traffic for TCP packet(s) with the correct identifier. It extracts and decrypts the embedded data using AES cipher with encryption key and salt value and creates the specified file.

Parameters

Packet object sniffed with Scapy.

Return

Type	Next State
Success	`MENU`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the file sniff.
     If not, ignore packet.

Decrypt the extracted modified file data using AES cipher with
an encryption key and salt value.

Create the received modified file in the local set directory.
```

dconfirm_sniff (attacker.py)

Purpose

Checks if the specified directory path exists on the victim machine.

Parameters

Directory path input from the attacker.

Return

Type	Next State
Success	`DIR_SNIFF`
Failure	`MENU`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the dconfirm sniff.
     If not, ignore packet.

If the source port is 7000,
     the directory exists on the victim machine.

If the source port is 8000,
     the directory does NOT exist on the victim machine.

If the directory exists,
     begin sniffing for TCP packets with the correct
identifier.

If the file does NOT exist,
     redirect back to the menu.
```

dir_sniff (attacker.py)

Purpose

Receives the created or modified file data from the victim by parsing packet object from sniffing the network traffic for TCP packet(s) with the correct identifier. It extracts and decrypts the embedded data using AES cipher with encryption key and salt value and creates the specified file.

Parameters

Packet object sniffed with Scapy.

Return

Type	Next State
Success	`MENU`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the dir sniff.
     If not, ignore packet.

Decrypt the extracted created or modified file data using AES
cipher with an encryption key and salt value.

Create the received created or modified file in the local set
directory.
```

shell_script (attacker.py)

Purpose

Receives the shell script output data from the victim by parsing packet object from sniffing the network traffic for TCP packet(s) with the correct identifier. It extracts and decrypts the embedded data using AES cipher with encryption key and salt value and prints the shell script output as a string.

Parameters

Packet object sniffed with Scapy.

Return

Type	Next State
Success	`RECV_CMD, CHECK_CMD`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the shell script.
     If not, ignore packet.

Decrypt the extracted created or modified file data using AES
cipher with an encryption key and salt value.

Print out the shell script output data as a string.
```

main (victim.py)

Purpose

Parses packet object from sniffing the network traffic, decrypts the extracted command line input using AES cipher with encryption key and salt value and carries out the specified function. If there is an output generated, encrypts it using AES, embeds it into a TCP packet with an identifier, and sends it to the attacker.

Parameters

The state object to initialize.

Return

Type	Next State
Success	`RECV_MENU, CHECK_MENU`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Sniff TCP packets using Scapy with the correct identifier.

Check that the packets are meant for the backdoor.
     If not, ignore packet.
```

keylogger_listener (victim.py)

Purpose

Reads from the keyboard, encrypts the data using Caesar cipher and set key value, embeds the encrypted data into TCP packets, and sends the packets over a covert channel to the attacker.

Parameters

Attacker IP address and Caesar cipher key value.

Return

Type	Next State
Success	`LISTENER`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Listen for keyboard input.

If a special key is pressed,
     pass.

If a normal key is pressed,
     encrypt the character using Caesar cipher according to
the set key value.

Create a TCP packet with an E flag.

Embed the encrypted data into the packet's source port using
Scapy.

Randomize the destination port.

Send the packet.
```

transfer_file (victim.py)

Purpose

Reads from the specified file, encrypts the data using AES cipher with encryption key and salt value, embeds the encrypted data into TCP packets, and sends the packets over a covert channel to the attacker.

Parameters

Attacker IP address and file path input from the attacker.

Return

Type	Next State
Success	`RECV_MENU, CHECK_MENU`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Open and read the specified file data as bytes.

Create a TCP packet with SEQ = 5555, Flags = C, and port
number = 7000.

Embed the encrypted data into the packet using Scapy

Randomize the destination port.

Send the packet.

At the end of the file, create a TCP packet with a SEQ = 5555,
Flags = C, and port number = 8080.

Randomize the destination port.

Send the packet.
```

file_listener (victim.py)

Purpose

Listens for modifications of the specified file using Watchdog. If there is a change, encrypts and embeds the log data and specified file data into TCP packets with identifiers and sends it to the attacker.

Parameters

Attacker IP address and AES encryption key and salt value.

Return

Type	Next State
Success	`LISTENER`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Listen for file modifications.

If the file is modified,
     encrypt the log data using AES cipher with encryption key and salt value.

If the file is not modified,
     keep listening.

Create a TCP packet with SEQ = 7777, Flags = A, and source port = 7000.

Embed the encrypted log data into the packet using Scapy.

Randomize the destination port.

Send the packet.

Create TCP packets with SEQ = 7777, Flags = A, and source port = 8000.

Embed the encrypted modified file data into the packets using Scapy.

Randomize the destination port.
```

Send the packets.

At the end of the file, create a TCP packet with SEQ = 7777, Flags = A, and port number = 8080.

Randomize the destination port.

Send the packet.

dir_listener (victim.py)

Purpose

Listens for creations or modifications of the specified directory using Watchdog. If there is a change, encrypts and embeds the specified file data into TCP packets with identifiers and sends it to the attacker.

Parameters

Attacker IP address and AES encryption key and salt value.

Return

Type	Next State
Success	`LISTENER`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Listen for directory modifications.

If a file is created or modified within the directory,
     encrypt the specified file data using AES cipher with
encryption key and salt value.

If the directory is not modified,
     keep listening.

Create a TCP packet with SEQ = 9999, Flags = U, and source
port = 7000.

Embed the filename into the packet using Scapy.

Randomize the destination port.

Send the packet.

Create a TCP packet with SEQ = 9999, Flags = U, and source
port = 8000.

Embed the specified file data into the packet using Scapy.

Randomize the destination port
```

Send the packet.

At the end of the file, create a TCP packet with SEQ = 9999,
Flags = U, and port number = 8080.

Randomize the destination port.

Send the packet.

shell_script (victim.py)

Purpose

Parses packet object from sniffing the network traffic, decrypts the extracted shell script command using AES cipher with encryption key and salt value, and runs the command. If there is a command output, encrypts it using AES cipher, embeds it into a TCP packet with an identifier, and sends it to the attacker.

Parameters

Packet object sniffed with Scapy.

Return

Type	Next State
Success	`RECV_CMD, CHECK_CMD`
Failure	`ERROR`

Pseudocode

```
If any errors occur
     return ERROR

Check that the packets are meant for the shell script by the
identifier.

Decrypt the extracted shell script command.

Pipe the command to a shell to execute it and receive the
output
     If not, print an ERROR message.

Encrypt the command output.

Create a TCP packet with an identifier, embed the encrypted
data into it, and send it to the attacker.
```