Final Project
Comp 8505
Design

Aadi Bisht
December 5, 2023

Table of Contents

Finite State Machine

State Transition Table

commander.py

From State	To State	Action
Start	parse_args	Command line invocation
parse_args	instantiate_covertTCP	Correct arguments
instantiate_covertTCP	send_port_knock_sequence	covertTCP instance initialized
send_port_knock_sequence	display_encryption_key	port knock sequence sent
display_encryption_key	make_ip_based_dirs	encryption key generated
make_ip_based_dirs	instantiate_watcher	directories created
instantiate_watcher	display_menu	Watcher initialized
display_menu	wait_for_command	menu displayed
wait_for_command	command_received	command received
command_received	send_command	command received
	command = 3 Transfer Keylog File	
send_command	wait_for_signal	if command is 3 (Transfer Keylog File)
wait_for_signal	keylogger_running_error	if signal received is 1
wait_for_signal	keylog_does_not_exist_error	if signal received is 2
wait_for_signal	receive_data	if signal received is 0
receive_data	wait_for_command	data received
	command = 4 Transfer File To	
send_command	get_file_name	if command is 4 (Transfer File To)

get_file_name	check_file_exists	input received
check_file_exists	send_signal	if exists send 1 and 0 if not exists
send_signal	wait_for_command	if file does not exist
send_signal	send_file	if file exists
send_file	wait_for_command	file sent
	command = 5 Transfer File From	
send_command	get_file_name	if command is 5 (Transfer File From)
get_file_name	send_file_name	input received
send_file_name	wait_for_signal	file name sent
wait_for_signal	file_does_not_exist_error	if signal received is 0
file_does_not_exist_error	wait_for_command	error printed
wait_for_signal	receive_data	if signal received is 1
receive_data	wait_for_command	data received
	command = 6 Run Program	
send_command	get_program_name	if command is 6 (Run Program)
get_program_name	send_program_name	program/command line args received
send_program_name	wait_for_response	program name sent
wait_for_response	print_error	response is 0
print_error	wait_for_command	error printed
wait_for_response	print_response	response is not 0
print_response	wait_for_command	response printed
	command = 7 Watch File	
send_command	get_file_name	if command is 7 (Watch File)
get_file_name	send_file_name	input received
send_file_name	wait_for_signal	file name sent
wait_for_signal	file_does_not_exist_error	if signal received is 0
file_does_not_exist_error	wait_for_command	error printed

wait_for_signal	check_watcher_status	if signal received is 1
check_watcher_status	start_watcher_process	if watcher is not running currently
start_watcher_process	receive_data	watcher process started
start_watcher_process	wait_for_command	watcher process running
check_watcher_status	print_watcher_running_error	if watcher is currently running
print_watcher_running_error	wait_for_command	error printed
	command = 8 Watch Directory	
send_command	get_dir_name	if command is 8 (Watch Directory)
get_dir_name	send_dir_name	input received
send_dir_name	wait_for_signal	directory name sent
wait_for_signal	dir_does_not_exist_error	if signal is 0
dir_does_not_exist_error	wait_for_command	error printed
wait_for_signal	check_watcher_status	if signal received is 1
check_watcher_status	start_watcher_process	if watcher is not running currently
start_watcher_process	receive_data	watcher process started
start_watcher_process	wait_for_command	watcher process running
check_watcher_status	print_watcher_running_error	if watcher is currently running
print_watcher_running_error	wait_for_command	error printed
	command = 9 Stop Watching File	
send_command	check_watcher_status	if command is 9 (Stop Watching File)
check_watcher_status	print_watcher_error	if watcher is not running or is watching a directory currently
print_watcher_error	wait_for_command	error printed

check_watcher_status	stop_watcher_process	if watcher is running and watching a file currently
stop_watcher_process	wait_for_command	watcher process stopped
	command = 10 Stop Watching Directory	
send_command	check_watcher_status	if command is 9 (Stop Watching File)
check_watcher_status	print_watcher_error	if watcher is not running or is watching a file currently
print_watcher_error	wait_for_command	error printed
check_watcher_status	stop_watcher_process	if watcher is running and watching a directory currently
stop_watcher_process	wait_for_command	watcher process stopped
	command = 11 or 12 Disconnect or Uninstall	
send_command	print_disconnecting_message	if command is 11 or 12 (Disconnect or Uninstall)
print_disconnecting_message	Exit	message printed

victim.py

From State	To State	Action
Start	parse_args	Command line invocation
parse_args	change_program_name	Correct arguments
change_program_name	instantiate_keylogger	program name obfuscated
instantiate_keylogger	instantiate_watcher	keylogger instantiated
instantiate_watcher	listen_for_port_knock	watcher instantiated
listen_for_port_knock	instantiate_covertTCP	correct sequence received
instantiate_covertTCP	get_encryption_key	covertTCP instantiated
get_encryption_key	wait_for_command	input received
wait_for_command	command_processor	command received
	command = 1 Start Keylogger	
command_processor	start_keylogger	if command received is 1 (Start Keylogger)
start_keylogger	create_keylog_file	creating keylogger dependencies
create_keylog_file	find_event_file_path	keylog.txt created
find_event_file_path	start_keylogger_process	event file found
start_keylogger_process	wait_for_event	process started
wait_for_event	check_stop_flag	event generated
check_stop_flag	stop_process	stop flag is set
check_stop_flag	parse_event	stop flag is not set
parse_event	write_to_log_file	events are parsed
write_to_log_file	wait_for_event	events are logged to keylog.txt
	command = 2 Stop Keylogger	
command_processor	check_keylogger_status	if command received is 2 (Stop Keylogger)
check_keylogger_status	keylogger_not_running	if keylogger is not running
keylogger_not_running	wait_for_command	error printed

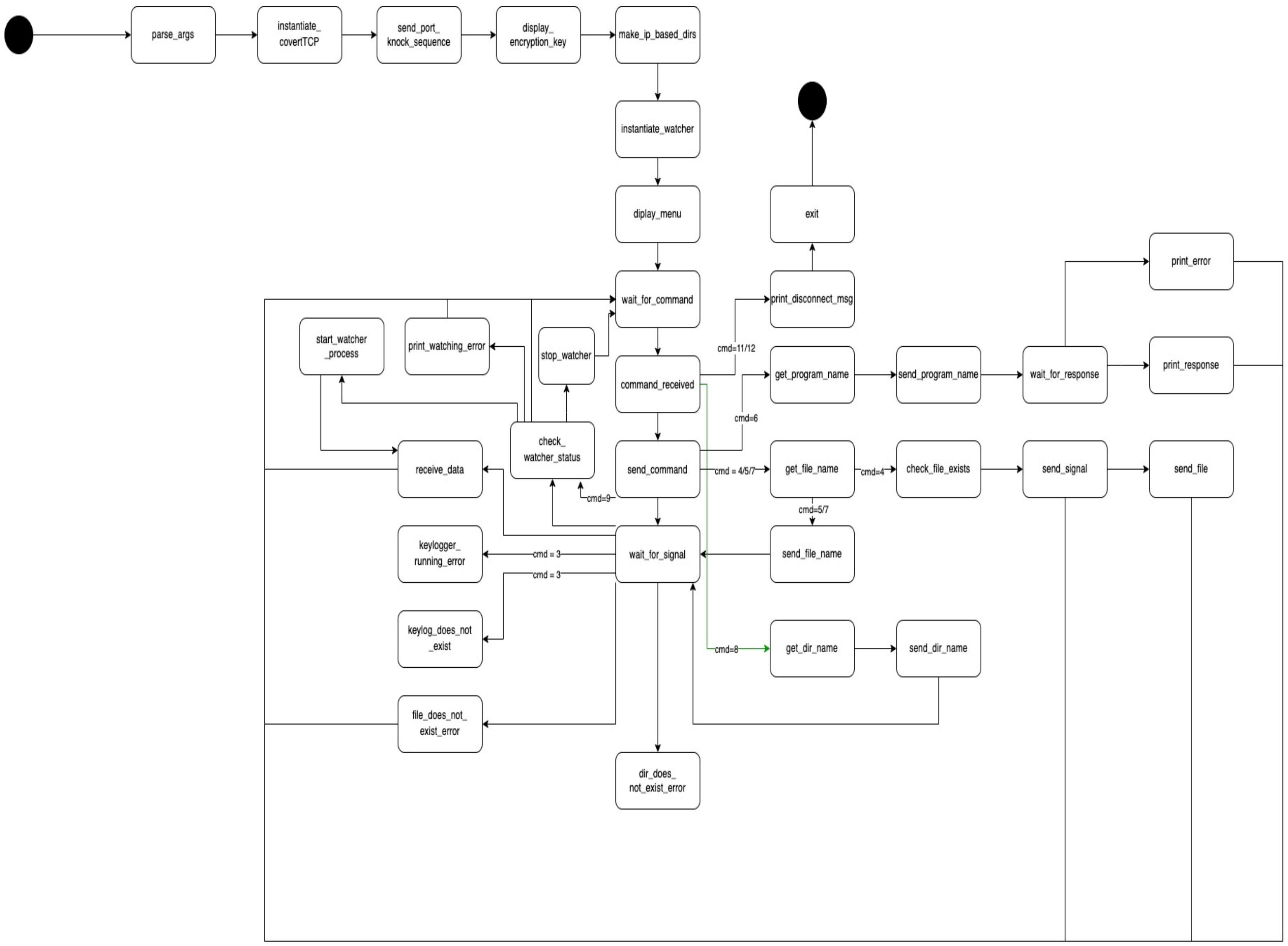
check_keylogger_status	call_stop_keylogger	if keylogger is running
call_stop_keylogger	set_stop_flag	
set_stop_flag	toggle_keylogger_status	stop flag is set
toggle_keylogger_status	wait_for_command	keylogger status set to False
	command = 3 Transfer Keylog File	
command_processor	check_keylogger_status	if command received is 3 (Transfer Keylog File)
check_keylogger_status	keylogger_running_error	if keylogger is running
keylogger_running_error	send_signal_1	error printed
send_signal_1	wait_for_command	signal_sent
check_keylogger_status	send_signal_0	if keylogger is not running
send_signal_0	send_keylog_file	signal sent
send_keylog_file	remove_keylog_file	file sent
remove_keylog_file	wait_for_command	keylog.txt removed from the system
	command = 4 Transfer File To	
command_processor	wait_for_signal	if command received is 4 (Transfer File To)
wait_for_signal	file_does_not_exist_error	if signal is 0
file_does_not_exist_error	wait_for_command	error printed
wait_for_signal	receive_data	if signal is 1
receive_data	wait_for_command	data received
	command = 5 Transfer File From	
command_processor	wait_for_file_name	if command received is 5 (Transfer File From)
wait_for_file_name	check_file_exists	filename received
check_file_exists	send_signal_0	if file does not exist
send_signal_0	print_error_message	signal sent
print_error_message	wait_for_command	error printed
check_file_exists	send_signal_1	if file exists
send_signal_1	send_file_data	signal sent
send_file_data	wait for command	file data sent

	command = 6 Run Program	
command_processor	receive_program	if command received is 6 (Run Program)
receive_program	run_program_in_shell	program received
run_program_in_shell	print_error	program is run in shell and an error is generated
print_error	send_response_0	error is printed
send_response_0	wait_for_command	response is sent
run_program_in_shell	capture_output	program is run in shell without an error
capture_output	send_output	output is captured
send_output	wait_for_command	output is sent
	command = 7 Watch File	
command_processor	wait_for_file_name	if command received is 7 (Watch File)
wait_for_file_name	check_file_exists	file name is received
check_file_exists	check_watcher_status	-
check_watcher_status	send_signal_0	if file does not exist or if watcher is running
send_signal_0	print_watcher_error	signal is sent
check_watcher_status	send_signal_1	if file exists and watcher is not running
send_signal_1	start_watching_file	signal is sent
start_watching_file	send_initial_file	watcher process is started
send_initial_file	wait_for_event	initial file is sent
wait_for_event	send_updated_file	event is generated
	command = 8 Watch Directory	
command_processor	wait_for_dir_name	if command received is 8 (Watch Directory)
wait_for_dir_name	check_dir_exists	dir name received
check_dir_exists	check_watcher_status	-
check_watcher_status	send_signal_0	if dir does not exist or if watcher is running
send_signal_0	print_watcher_error	signal is sent

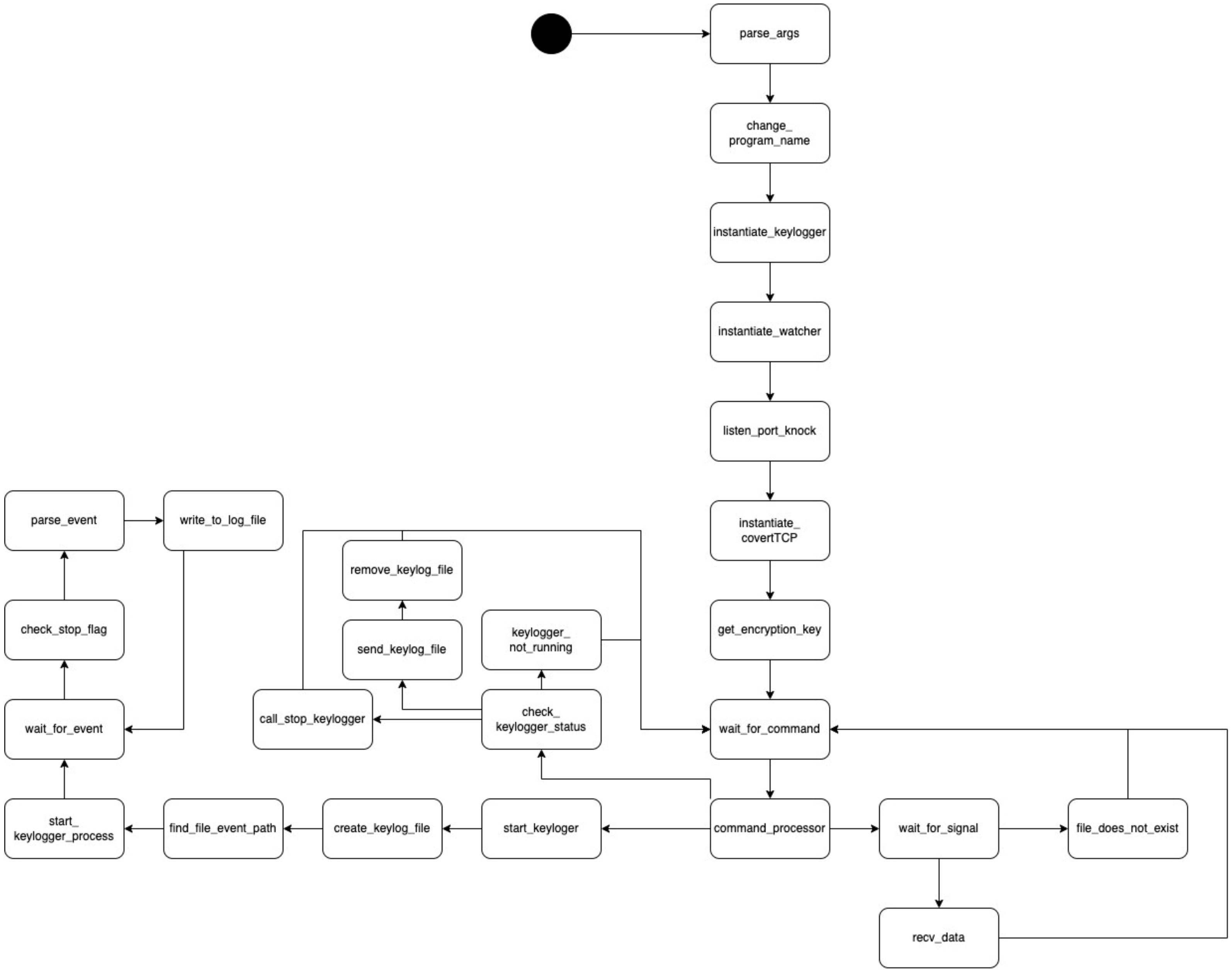
check_watcher_status	send_signal_1	if dir exists and watcher is not running
send_signal_1	start_watching_dir	signal is sent
start_watching_dir	send_initial_dir	watcher process is started
send_initial_dir	wait_for_event	initial dir is sent
wait_for_event	send_updated_dir	event is generated
	command = 9 Stop Watching File	
command_processor	check_watcher_status	if command received is 9 (Stop Watching File)
check_watcher_status	not_running_error	if watcher is not running
check_watcher_status	call_watcher_stop_watching	if watcher is running
call_watcher_stop_watching	terminate_watcher_process	-
	command = 10 Stop Watching Directory	
command_processor	check_watcher_status	if command received is 10 (Stop Watching Directory)
check_watcher_status	not_running_error	if watcher is not running
check_watcher_status	call_watcher_stop_watching	if watcher is running
call_watcher_stop_watching	terminate_watcher_process	-
	command = 11 Disconnect	
command_processor	print_message	if command received is 11 (Disconnect)
print_message	listen_for_port_knock	message is printed
	command = 12 Uninstall	
command_processor	print_message	if command received is 12 (Uninstall)
print_message	remove_files	message is printed
remove_files	Exit	files are removed

State Transition Diagram

commander.py



victim.py


parse_args
change_program_name
instantiate_keylogger
instantiate_watcher
listen_port_knock
instantiate_covertTCP
get_encryption_key
wait_for_command
command_processor
wait_for_signal
file_does_not_exist
recv_data
start_keyloger
create_keylog_file
find_file_event_path
start_keylogger_process
wait_for_event
check_stop_flag
parse_event
write_to_log_file
check_keylogger_status
keylogger_not_running
send_keylog_file
remove_keylog_file
call_stop_keylogger

Functions: Commander.py

handle_victim

Purpose

This function provides a menu for interaction.

Parameters

covert: a CovertTCP object

Return

None

Pseudocode

```
Call the make_dir function
Create a Watcher instance called watcher_instance
Start an infinite loop:
    Display a menu of options to the user
    Read an integer “choice” from the user input
    Send data with covert
    If the choice is equal to 3:
        Receive a signal "sig" from the victim using covert
        If sig is equal to 1:
                Print "[BAD COMMAND] Keylogger should be
        stopped before transferring keylog.txt"
                Continue to the next iteration
        ElseIf sig is equal to 2:
                Print "[FILE ERROR] keylog.txt does not exist."
                Continue to the next iteration
            Receive data from the victim using covert
    ElseIf "choice" is equal to 4:
        Read a file name from user input
        Check if file exists
        If the file does not exist:
            Print "[ERROR: File does not exist] wrong file path"
            Send a signal ‘0’ to the victim
            Continue to the next iteration
        Send a signal ‘1’ to the victim
        Send the file to the victim
    ElseIf "choice" is equal to 5:
        Read a file name from user input
        Send the file name to victim
        Receive a signal from the victim
        If signal is 0
            Print that the file does not exist
            Continue to the next iteration
        Receive the file’s data
    ElseIf "choice" is equal to 6:
        Read a command from user input
        Send the command to victim
        Receive the response from the victim
        Print the response
    ElseIf "choice" is equal to 7:
        Read a file name from user input
        Send the file name to victim
        Receive a signal from the victim
        If signal is 0
            Print that the file does not exist
            Continue to the next iteration
        If the watcher is not running
            Create and start a watcher process
        Else
```

```
            Print the watcher’s status
    ElseIf "choice" is equal to 8:
        Read a directory name from user input
        Send the directory name to victim
        Receive a signal from the victim
        If signal is 0
            Print that the directory does not exist
            Continue to the next iteration
        If the watcher is not running
            Create and start a watcher process
        Else
            Print the watcher’s status

    ElseIf "choice" is equal to 9:
        If watcher is running and watcher is watching a file:
            Stop the watcher process
        Else
            Print the watcher process status

    ElseIf "choice" is equal to 10:
        If watcher is running and watcher is watching a directory:
            Stop the watcher process
        Else
            Print the watcher process status

    If the choice is equal to 11:
        Print a disconnection message
        Break out of the infinite loop
```

```
If the choice is equal to 12:
    Print a disconnection message
    Break out of the infinite loop
```

make_dir

Purpose

This function creates a directory with the given IP address.

Parameters

ip: a string

Return

Returns the path of the created directory as a string

Pseudocode

```
initialize a variable 'directory' to 'downloads/' + ip.

Use os.makedirs(directory, exist_ok=True) to create the directory if
it doesn't exist.

Return the 'directory' string.
```

Functions: watcher.py

start_watching

Purpose

Start watching a file or directory using a separate process.

Parameters

covert_instance: an instance of CovertTCP object
path: the path of the file or directory to watch

Return

None

Pseudocode

```
Set self.__status to True
If self.__is_file:
        Create a new multiprocessing process:
            Target is self.watch_file
            Arguments are covert_inst and path
        Print "[WATCHER] File Watching on {path}"
        Start the watcher_process
        Call self.toggle_file
        Set self.__child to watcher_process


    ElseIf self.__is_dir:
        Create a new multiprocessing process:
            Target is self.watch_file
            Arguments are covert_inst and path
        Print "[WATCHER] Directory Watching on {path}"
        Start the watcher_process
        Call self.toggle_dir
        Set self.__child to watcher_process
```

watch_file

Purpose

watches the file and directory for changes

Parameters

self: the instance of the class
covert: an instance of CovertTCP
file_name: the name of the file to watch

Return

None

Pseudocode

```
Set acceptable_events to ["IN_MOVE_SELF", "IN_MODIFY", "IN_MOVED_TO",
"IN_MOVED_FROM", "IN_CREATE"]
    Create an instance i of inotify.adapters.Inotify()
    Add a watch for file_name in i


    If not watching_dir_or_file():
        Set covert.file_name to file_name
        Send data with is_victim=False and event="IN_MODIFY"
    Else:
        Set covert.file_name to file_name
        Set covert.is_dir to True
        Send data with is_victim=False and event="IN_MODIFY"
        Set covert.file_name to None
        Set covert.is_dir to False


        For each entry in os.scandir(file_name):
            If entry is a file:
                Set covert.file_name to file_name + '/' + entry.name
                Send data with is_victim=False and event="IN_MODIFY"
                Set covert.file_name to None
            Else if entry is a directory:
                Set covert.file_name to file_name + '/' + entry.name
                Set covert.is_dir to True
                Send data with is_victim=False and event="IN_MODIFY"
                Set covert.file_name to None
                Set covert.is_dir to False

    For each event in i.event_gen(yield_nones=False):
        Extract (_, type_names, path, filename) from the event
        If ".part" or ".kate-swp" in filename:
            Continue to the next iteration
        If type_names[0] is in acceptable_events:
            If watching_dir_or_file():
                If "IN_ISDIR" in type_names:
                    Set covert.is_dir to True
                Else:
                    Set covert.is_dir to False
                Set covert.file_name to path + '/' + filename
                Send data with is_victim=False and event=type_names[0]
                Set covert.file_name to None
                Set covert.is_dir to False
            Else if not watching_dir_or_file():
                Send data with is_victim=False and event=type_names[0]
```

Functions: Victim.py

Main

Purpose

the main entry point for the program

Parameters

None

Return

None

Pseudocode

```
Create a command-line argument parser object using
argparse
Add an arguments '-p'
Parse the command-line arguments
Call the change_program_name function
Start an infinite loop:
    Accept a connection conn by calling receive_conn
    Print a message that a connection has been
    established
    Start a nested infinte loop:
        Receive a command from the conn socket
        Call the command_processor function to process
        the received command
        Check if the received command is 9 or 0:
            close the socket and break out of the
            nested loop
```

receive_conn

Purpose

Listen and accept the incoming connection

Parameters

port: an int

Return

A socket object with connection to commander

Pseudocode

```
Store a self_ip of victim’s (self) ip
Use with to create and open a socket object as sock:
    Put socket option to SO_REUSEADDR to 1
    Bind the socket to port and self_ip
    Listen for incoming connection
    Return the socket given by accept function
```

start_keylogger

Purpose

Start the keylogger on victim’s machine.

Parameters

Conn: a socket object with connection to commander

Return

None

Pseudocode

```
Call the function create_log_file to create the
keylog.txt file

Get the event file of the keyboard located in
/proc/bus/input/devices by calling get_event_path
function

Instantiate a stop_flag, a Multiprocessing Event object

Create a keylogger_process, a Multiprocessing Process,
which runs the read_event

Start the keylogger_process

Call the function stop_keylogger with stop_flag and
conn as arguments

Wait for keylogger_process
```

read_event

Purpose

Read the event file associated with the system's keyboard

Parameters

eventpath: path to the event file associated with the system's keyboard

stop_flag: a Multiprocessing Event object

Return

None

Pseudocode

```
Instantiate 2 boolean variables shift_key_pressed and
capslock_pressed

Make a subprocess called process which uses Popen to
run the linux command evtest and get the stdout and
stderr

Start a loop until stop_flag is set:

    Instantiate a line object which takes stdout from
process

    Send the line to process_line function and store
    the returned value in key_value

    If key_value exists:

        The first returned value is code

        The second returned value is value

        If code is 42 or 54 and the value is 1:

            Set shift_key_pressed to True

        Else if the code is 42 or 54 and value is 0

            Set shift_key_pressed to False
```

```
Else if the code is 58 and the value is 1:
    Toggle the capslock_pressed value

Else:
    Call the manage_shift_and_caps with
    arguments of shift_key_pressed and
    capslock_pressed
```

manage_shift_and_caps

Purpose

Logs the key pressed according to the shift key and caps key

Parameters

shift_key_pressed: a Boolean representing state of shift key

capslock_pressed: a Boolean representing state of shift key

code: an int of the key pressed

Return

None

Pseudocode

```
If shift key and capslock are pressed:
    Log the special character if it exists
     Or log the character as non-capitalised
Else if the shift key is pressed but the capslock is not pressed:
    Log the special character if it exists
     Or log the character as capitalised
Else if the shift key is not pressed but the capslock is pressed:
    log the character as capitalised
else if both keys are not pressed:
    log the character as non-capitalised
```

get_event_path

Purpose

Gets the event file of the keyboard from proc/bus/input/devices folder

Parameters

None

Return

Infile_path: a string representing the path to the event file of the keyboard

Pseudocode

```
Open the proc/bus/input/devices file
    Read all the lines
    Use regular expressions to get all the lines with
    “Handlers|EV=”
    Get the line with “EV=120013”
    Go to the line present above the EV line and get
    the event number
    Return the "/dev/input/" + event number found
```

stop_keylogger

Purpose

Stop the keylogger on victim’s machine.

Parameters

Conn: a socket object with connection to commander

stop_flag: a Multiprocessing Event object

Return

None

Pseudocode

```
Wait for a command from the commander
Start an infinite while loop:
    If command received is 2:
        Set the stop_flag and break
    Else if the command is 9 or 3 or 0:
        Print error as the keylogger is running
    Else:
        Print unrecognized command
    Receive another command from the commander
```