CPSC 526 Winter 2024
Assignment 4

Oscar campos
UCID: 30057153

Task 1 –ncbot.py

ncbot.py is a Python script designed to run a networked command bot that connects to a server via TCP, listens for authenticated commands, and performs actions based on those commands. This bot can execute a variety of tasks such as cheeking status of the bots, simulating attacks on specified hosts and ports, moving to a new server, and shutting down gracefully.

Features

- Server Connection: Automatically connects to a specified server and port, retrying indefinitely on failure.
- Command Authentication: Verifies commands using a nonce and a secret for message authentication, ensuring that commands are accepted only from authenticated sources.
- Command Execution: Executes commands including status, shutdown, move, and attack, with specific actions for each.
- Attack Simulation: Can simulate an attack on a specified host and port, sending a predefined message.
- Server Mobility: Capable of moving to a new server by disconnecting from the current server and connecting to a new specified server.
- Graceful Shutdown: Can shut down the bot gracefully, closing all connections properly.

Usage

To use ncbot.py, you need Python 3 installed on your system. Run the script from the command line, providing the target server's hostname and port, the bot's nickname, and a secret for command authentication:

Example

- ./ncbot.py <hostname:port> <nick> <secret>
    - ./ncbot.py localhost:6667 myBot secretKey123

Commands

- status: Displays the status of the bot, including its nickname and the number of commands processed.
- shutdown: Instructs the bot to shut down gracefully.
- move <hostname>:<port>: Commands the bot to move to a new server specified by the hostname and port.
- attack <hostname>:<port>: Simulates an attack on the specified hostname and port.

Developing

The ncbot.py script is structured into functions handling different aspects of the bot's operation, from network communication to command processing. A lot of the developing was following the instruction from the assignment description. Key functions include:

- parse_command_line_arguments(): Parses and validates command-line arguments.
- connect_to_server(): Manages the connection to the server, with reconnection logic.
- send_data() and recv_data(): Functions for sending and receiving data over a socket.
- listen_for_commands(): Listens for and processes incoming commands from the server.
- execute_command(): Executes the command based on its type (status, shutdown, move, attack).
- Handling KeyboardInterrupt: The main loop of the script includes handling for KeyboardInterrupt (Ctrl+C), allowing for the graceful termination of the bot in case of an interruption signal from the user.

Task 2 – nccontroller.py

nccontroller.py is a Python script designed to act as a controller for managing bots connected to a server. It sends authenticated commands to bots and processes their responses. This controller can issue the following commands including querying bot status, initiating simulated attacks, instructing bots to shut down, and commanding bots to move to a different server.

Features

- Command Sending: Sends commands to bots connected to a specified server, using a secret for command authentication.
- Response Processing: Receives and processes responses from bots, providing feedback on the execution of issued commands.
- Flexible Command Support: Supports a variety of commands including status, shutdown, move <hostname>:<port>, and attack <hostname>:<port>.
- Interactive Command Input: Allows users to input commands interactively through the command line.

Usage

To use nccontroller.py, you will need Python 3 installed on your system. Run the script from the command line, providing the target server's hostname and port, and a secret for command authentication:

Example:

- ./nccontroller.py <hostname:port> <secret>
    - ./nccontroller.py localhost:6667 mySecretKey

Commands

- status: Queries the status of all connected bots.
- shutdown: Instructs all connected bots to shut down gracefully.
- move <hostname>:<port>: Commands all bots to disconnect and move to a new server specified by the hostname and port.
- attack <hostname>:<port>: Initiates a simulated attack by all bots on the specified hostname and port.

Command Line Interface

- cmd>: The script provides a prompt (cmd>) where you can type commands to be sent to bots.
- quit: Type quit at the cmd> prompt to exit the controller.

Developing

The nccontroller.py script includes several key functions that were help develop by the task description in the assignment pdf. Its functions include:

- parse_arguments(): Parses command line arguments to extract the server hostname, port, and secret.
- compute_mac(): Computes the message authentication code (MAC) for a given nonce and secret.
- send_data() and recv_data(): Functions for sending and receiving data over a socket.
- send_command(): Formats and sends a command along with its MAC to the server.
- receive_responses(): Receives responses from bots within a specified timeout period.
- process_responses(): Processes and prints the responses received from bots based on the issued command.
- Handling KeyboardInterrupt:The script includes handling for KeyboardInterrupt (Ctrl+C), allowing for graceful termination if interrupted by the user.

Task 3 – ircbot.py

ircbot.py is a Python script designed to run a bot that connects to an IRC (Internet Relay Chat) server, listens for authenticated commands, and executes actions based on those commands. It can perform a variety of tasks such as simulating attacks on specified hosts and ports, moving to a new server, reporting its status, and shutting down gracefully.

Features

- Automatic Server Connection: Connects to a specified IRC server and joins a channel, retrying indefinitely on failure.
- Command Authentication: Ensures commands are authentic by verifying a nonce and secret-based MAC (Message Authentication Code).
- Command Execution: Responds to commands like status, shutdown, move, and attack with specific actions for each.
- Attack Simulation: Capable of simulating an attack on a given hostname and port, reporting the outcome back to the IRC channel.
- Server Mobility: Can move to a new server as instructed by commands received over IRC.
- Graceful Shutdown: Supports shutting down the bot through an IRC command, closing connections properly.

Usage

To use ircbot.py, ensure Python 3 is installed on your system. Run the script from the command line, providing the IRC server's hostname and port, the channel to join, and the secret for command authentication:

Example

- ./ircbot.py <server:port> <channel> <secret>
    - ./ircbot.py localhost:6667 "#myChannel" mySecret

Ensure to use quotes around parameters that contain special characters or spaces.

Commands

This bot supports several commands issued over the IRC channel, which must be authenticated using a nonce and secret-based MAC:

- status: Reports the bot's current status, including its nickname and the number of commands processed.
- shutdown: Instructs the bot to shut down gracefully, closing its connection to the IRC server.

- move <hostname>:<port>: Commands the bot to disconnect and move to a new server specified by hostname and port.
- attack <hostname>:<port>: Initiates a simulated attack on the specified hostname and port.

<u>Developing</u>

The nccontroller.py script includes several key functions that were help develop by the task description in the assignment pdf and the ncbot.py which has a similar functionality other than connecting server appropriately. Its functions include:

- generate_random_nickname(): Generates a random nickname for the bot.
- connect_to_irc_server(): Manages the bot's connection to the IRC server, including reconnecting if disconnected.
- authenticate_command(): Authenticates incoming commands based on the nonce and secret.
- listen_for_commands(): Listens for and processes commands from the IRC channel.
- process_command(): Executes the received command.
    - send_status(), shutdown_bot(), perform_attack(), move_to_new_server(): Perform actions based on the command received.
- Handling KeyboardInterrupt:To handle unexpected exits, the script gracefully shuts down the bot upon receiving a KeyboardInterrupt signal (Ctrl+C).