14.09.2023

Курс:

Практическая работа к уроку № Lesson_7

--

Усиление защиты от XSS при помощи Content Security Policy

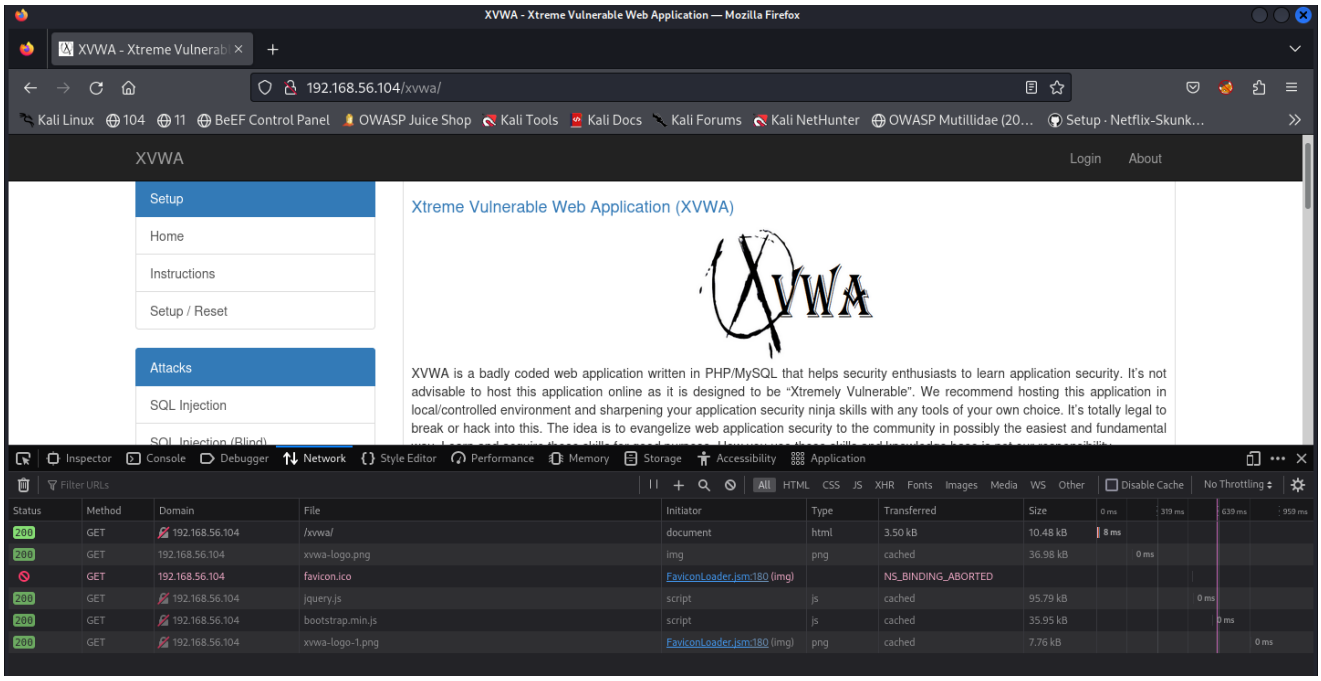
Задание_1:

Выполнить максимально полную защиту сервиса XVWA средствами CSP.

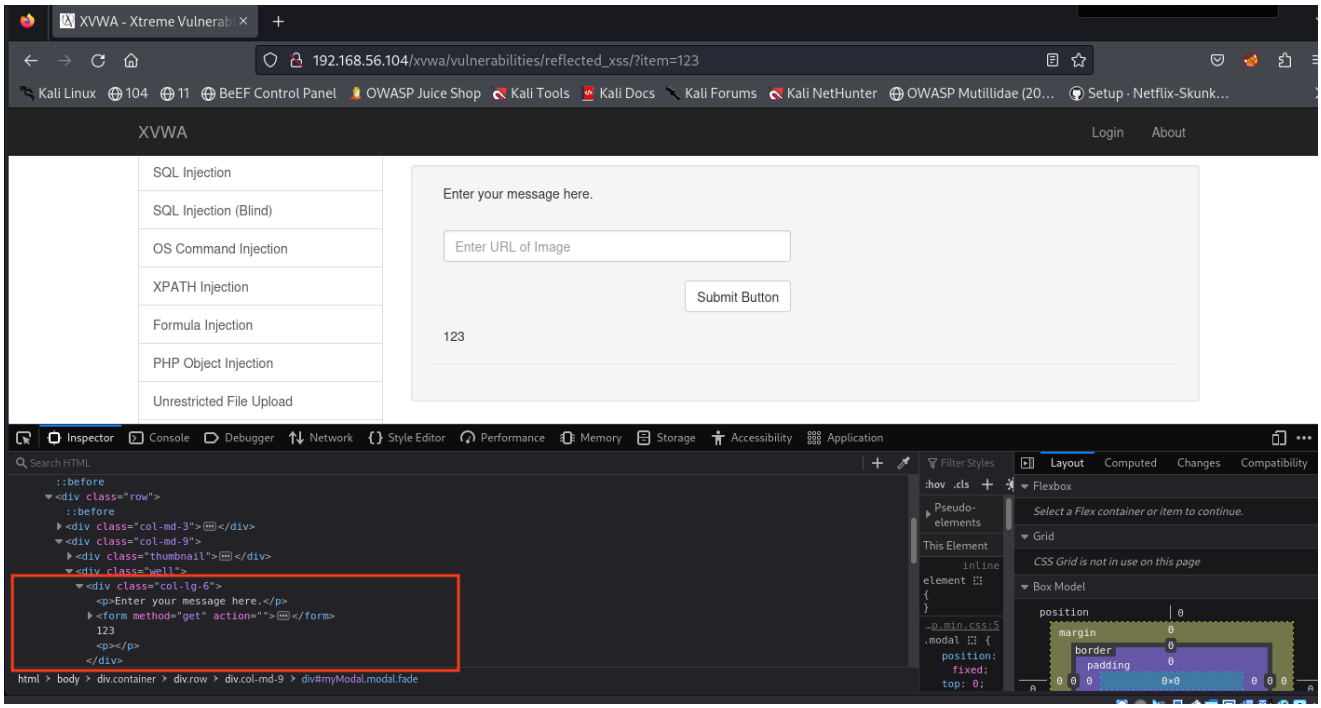
Настройка CSP

1. Настройка напрямую на сервере либо
2. XVWA -> Firefox Inspector

Заголовков нет, мешать ничего не должно

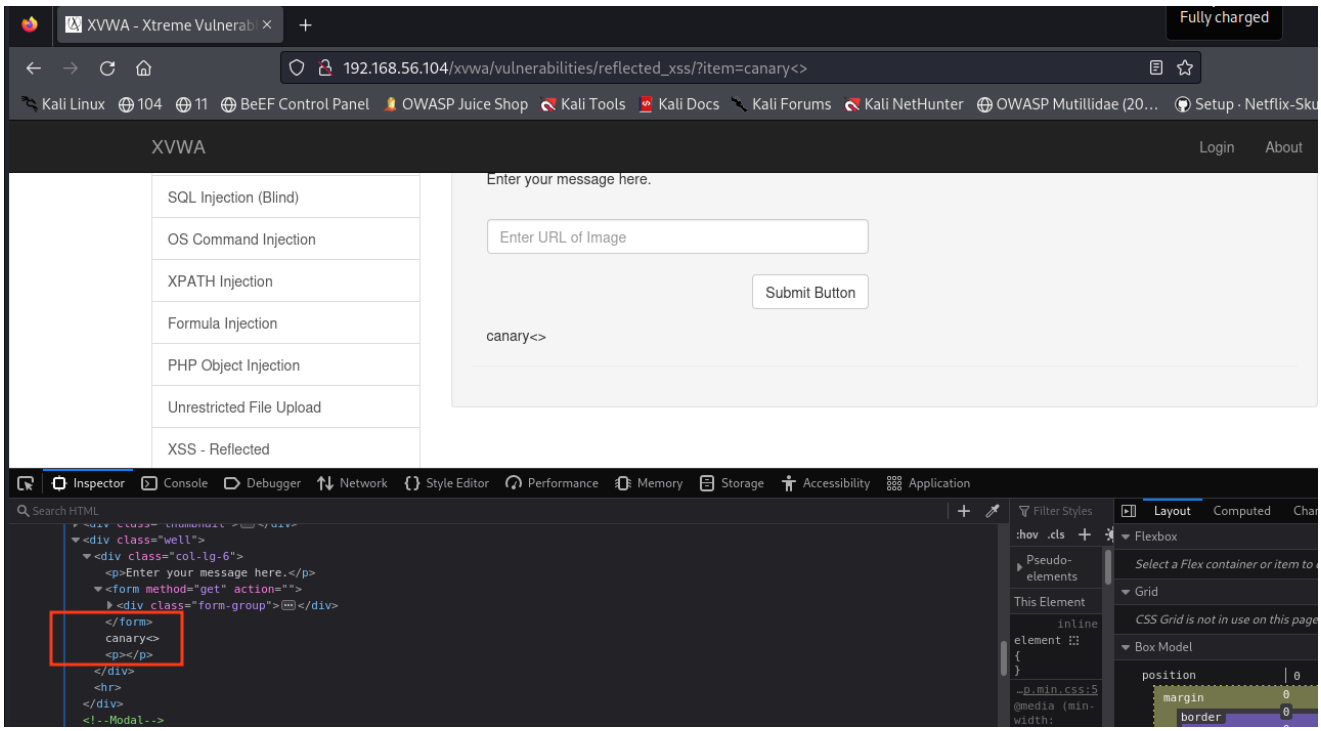


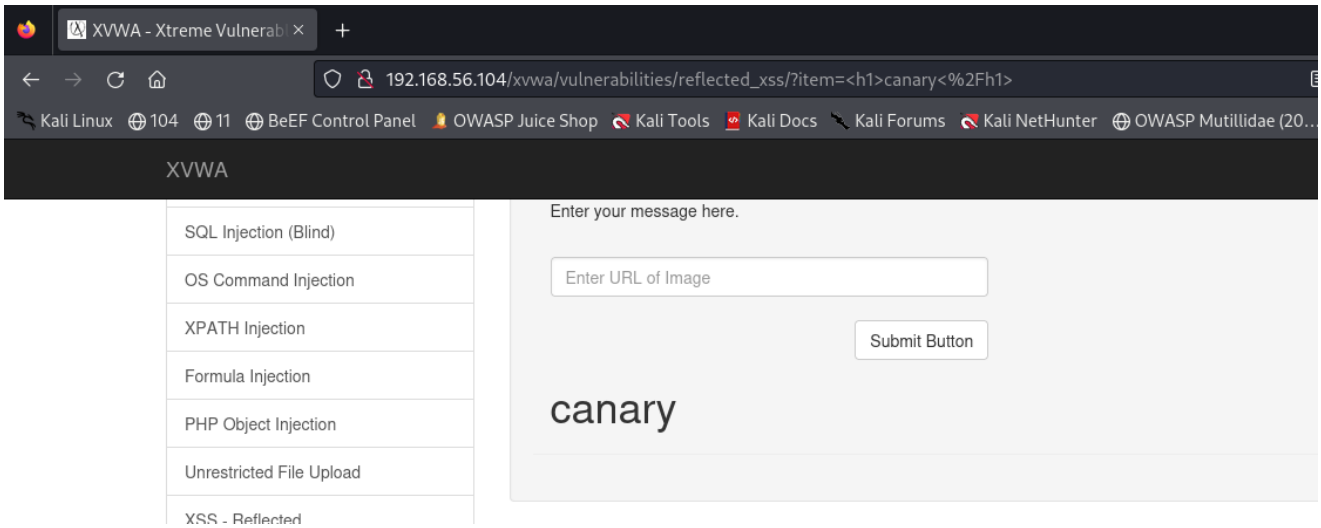
- Проверим на *XSS Reflected*

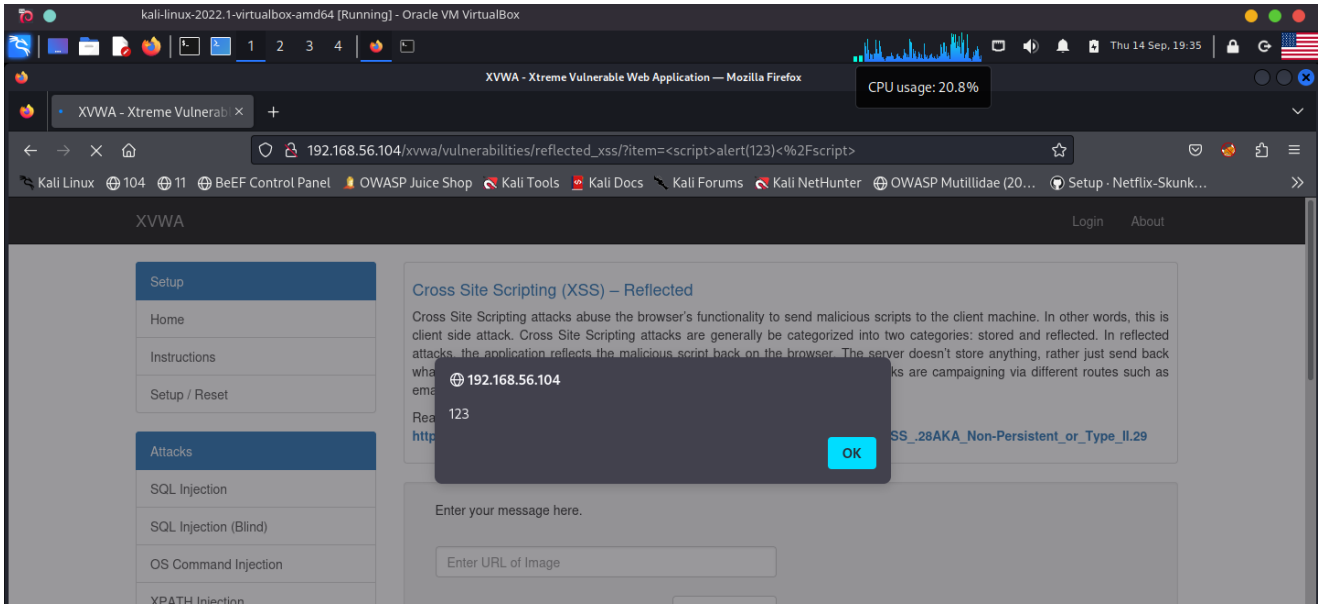


УЯ чистый HTML, пробуем проверить на фильтр через вектор атаки в виде тегов:

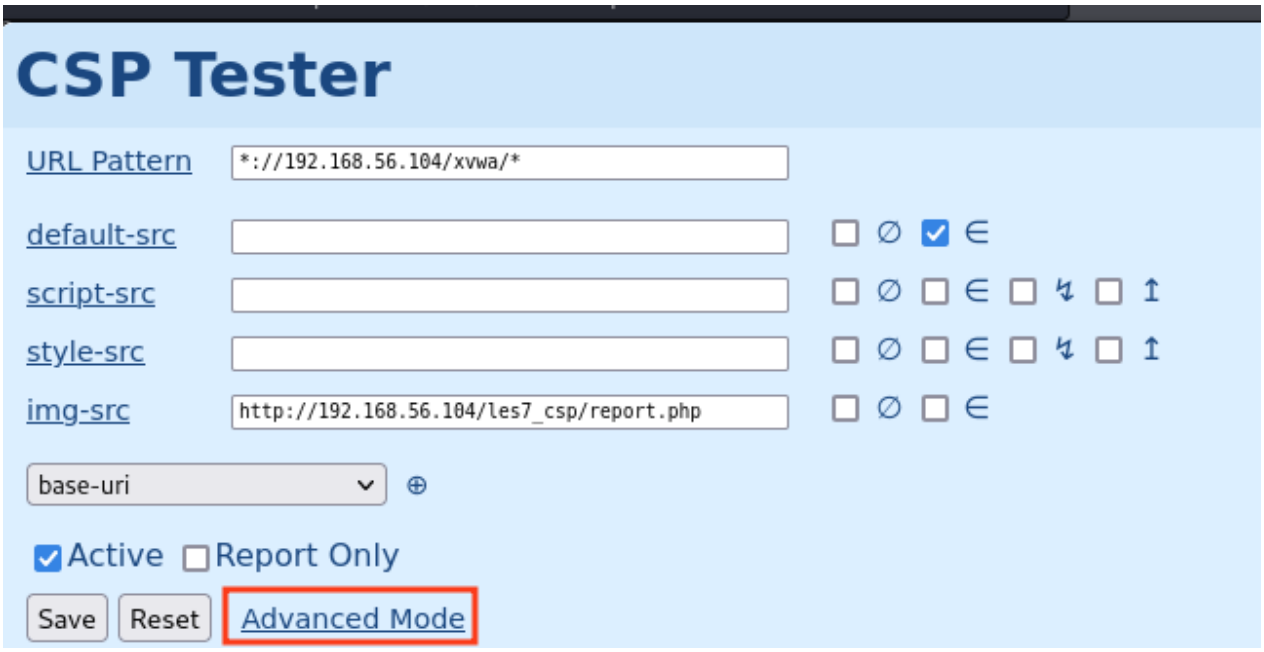
```
canary<>
<h1>canary</h1>
<script>alert(123)</script>
```

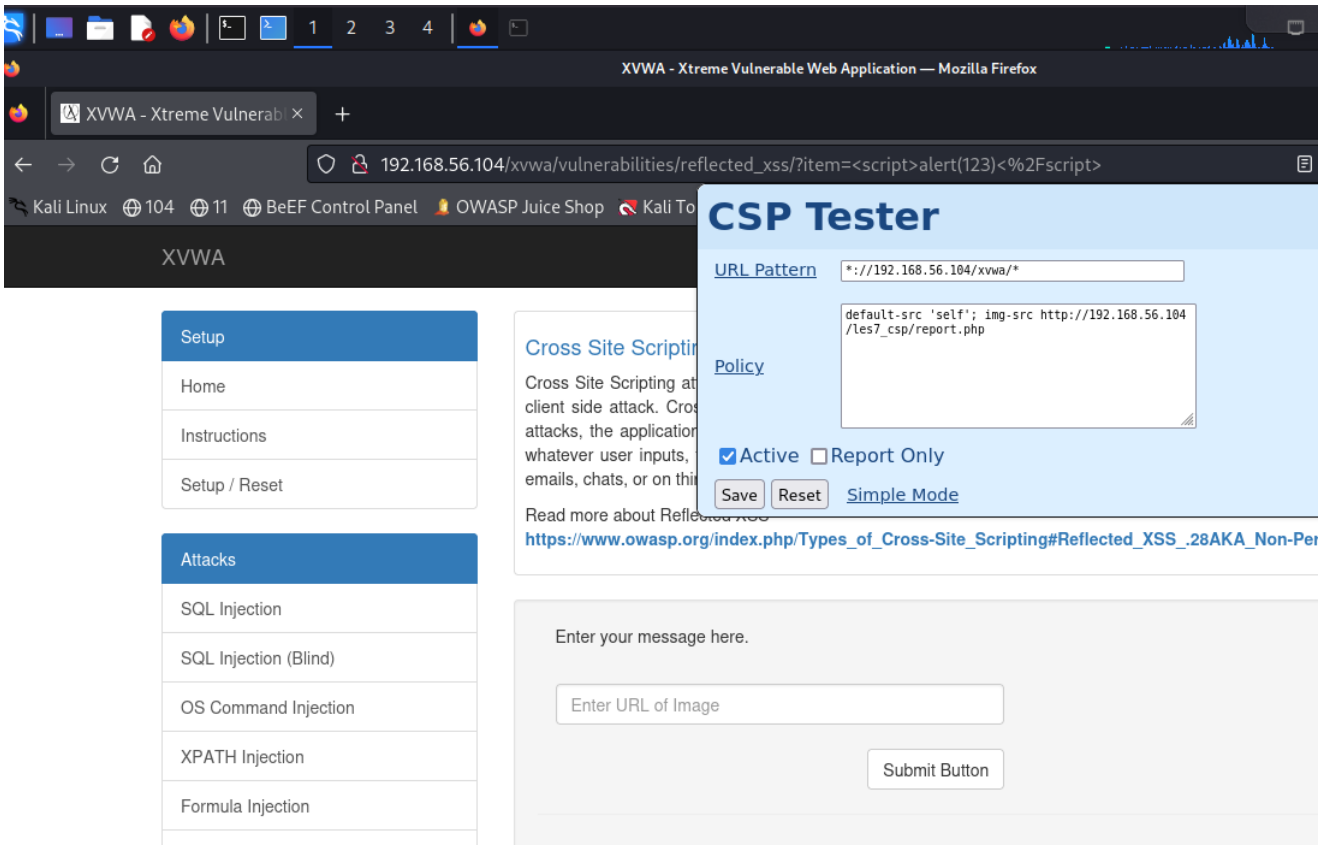




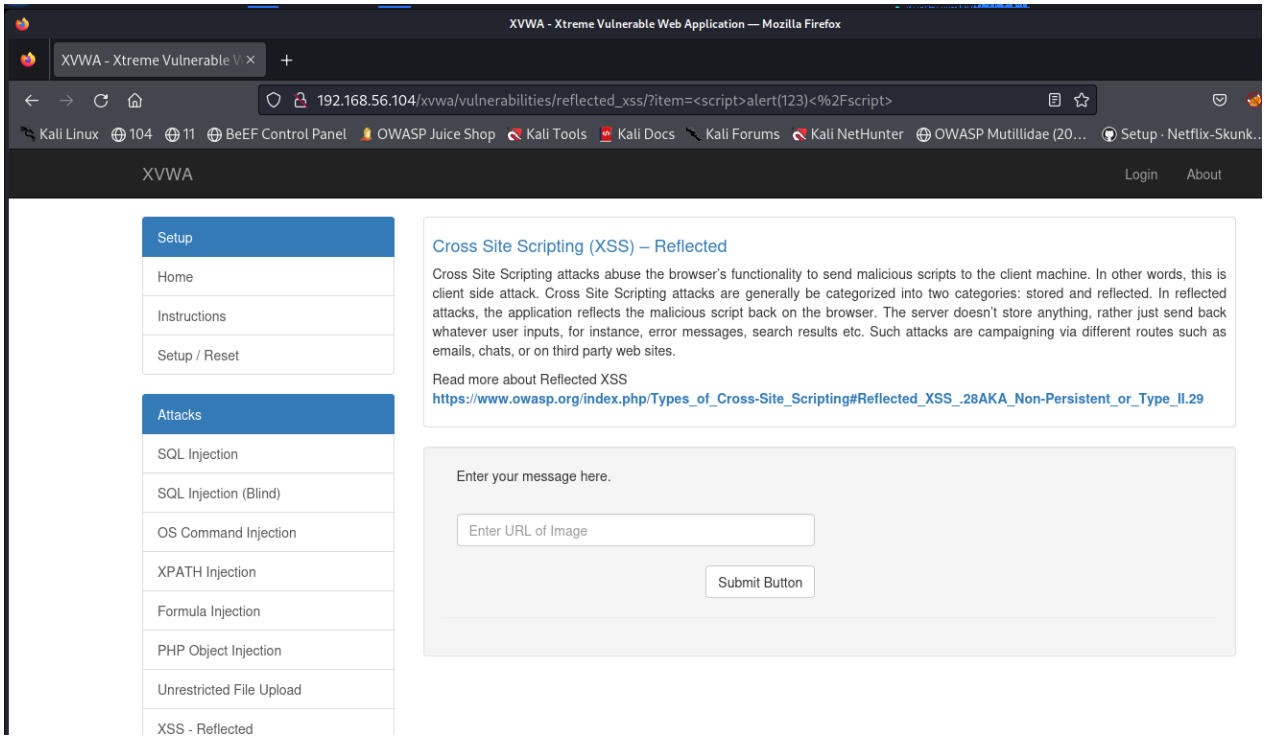


- Настраиваем через *CSP tester* (Firefox Extension):





- Проверяем, сработало:



- Копируем на сервер, *CSP*

```
default-src 'self'; img-src http://192.168.56.104/les7_csp/report.php
```

... доделываю

Задание_2:

Выполнить максимально полную защиту сервиса bWAPP средствами CSP.

... доделываю

Задание_3:

Составить план защиты сервиса Mutillidae, в рамках которого можно внедрить CSP, не нарушив при этом работу сервиса. Указать, какие дополнительные средства можно внедрить для защиты от XSS.

... доделываю

Задание_4:

(*) Реализовать защиту сервиса Mutillidae согласно разработанному выше плану.

... доделываю

Выводы:

Content Security Policy – это дополнительный уровень безопасности для защиты данных в случае обхода злоумышленником Same-origin policy. Например, если у злоумышленника есть возможность добавить скрипт на страницу с помощью XSS, то с точки зрения жертвы произойдет обход SOP и атака реализуется.

Можно выделить следующие этапы создания политики CSP:

1. Определить, какие сценарии надо разрешить. Нужно иметь в виду, что с точки зрения дальнейшего развертывания CSP гораздо проще писать приложение так, чтобы все сценарии, которые используются в приложениях, были внешними (т.е. хранились на сервере, отличном от сервера для сайта). Это расширит границы использования директивы script-src.
2. Настроить политику в тестовом режиме. Для этого можно использовать CSP Tester или CSP Mitigator. Важно понимать, что при настройке CSP внешний вид сайта может поменяться. Поэтому этот риск надо заранее закладывать при проектировании сайта (или приложения). Многие хостинг-провайдеры позволяют использовать свои настройки для CSP, что также нужно учитывать при тестировании и настройке политики.
3. Протестировать, какие могут быть способы обхода политики, и внедрить для них дополнительные меры защиты. Может возникнуть ситуация, при которой какие-то директивы CSP будут сильно нарушать работу сайта и внедрить их не получится. Чтобы снизить риск от последствий таких ситуаций, рекомендуется внедрять дополнительные средства защиты.
4. Настроить понятную систему оповещения о срабатывании CSP. В первую очередь это будет полезно для контроля системы защиты. Чем серьезнее будет проект, тем выше роль в нем системы, которая позволит оперативно реагировать на риски, связанные со срабатыванием правил CSP.
5. Внедрить политику на сервер и оценить ее влияние на работу ресурса. Важно помнить, что при использовании фреймворков наподобие Jquery, AngularJS и т.п. всегда будут риски, связанные с обходом CSP. И всегда следует помнить о том, что CSP – один из последних рубежей обороны от XSS.

Ссылки / дополнительные материалы:

https://habr.com/company/kidsreview/blog/224797/ – подмена (встраивание) спам-ссылок на страницы.
https://report-uri.com/home/hash – генератор хэш для тегов `<script> и <style>`.
https://content-security-policy.com/ – описание основных директив для CSP.
https://report-uri.com/home/generate – генератор правил для CSP.
https://www.blackhat.com/docs/us-17/thursday/us-17-Lekies-Dont-Trust-The-DOM-Bypassing-XSS-Mitigations-Via-Script-Gadgets.pdf – обход CSP при помощи Script Gadjet (AngularJS и т.п.).
https://www.owasp.org/images/3/32/OWASP_BeNeLux-Day_2017_Bypassing_XSS_mitigations_via_script_gadgets_Sebastian_Lekies.pdf – обход XSS-фильтров при помощи Script Gadjets.
https://blog.zsec.uk/csrf-2018/ – практическая реализация CSRF.
https://medium.com/@henslejoseph/php-and-jwt-tutorial-make-a-two-factor-authentication-system-7264962b8fcc – реализация JWT в PHP с использованием Zend framework.
https://www.owasp.org/index.php/JSON_Web_Token_(JWT)_Cheat_Sheet_for_Java#Issues – ошибки в использовании JWT и их решение на примере Java.
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
https://developer.mozilla.org/ru/docs/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C/safe
https://www.owasp.org/index.php/Testing_for_CSRF_(OTG-SESS-005)#Summary
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
https://www.securitylab.ru/analytics/292473.php
https://www.acunetix.com/blog/articles/csrf-xss-brothers-arms/
https://www.acunetix.com/blog/articles/cross-site-request-forgery/
https://nvisium.com/blog/2014/02/14/using-burp-intruder-to-test-csrf.html
https://www.acunetix.com/websitesecurity/csrf-attacks/
https://jwt.io/
https://habr.com/post/340146/
https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec.
https://coderwall.com/p/8wrxfw/goodbye-php-sessions-hello-json-web-tokens.

> Вся информация в данной работе представлена исключительно в ознакомительных целях! Любое использование на практике без согласования тестирования подпадает под действие УК РФ.

- https://gb.ru

Выполнил: AndreiM