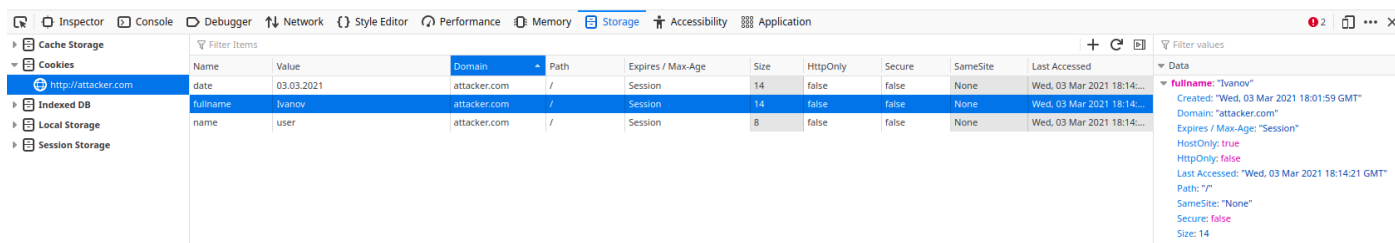
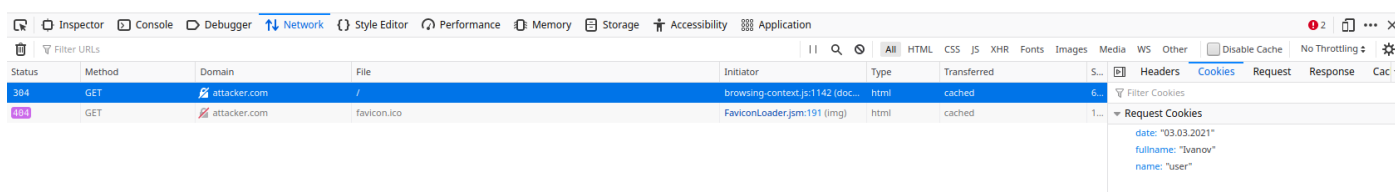
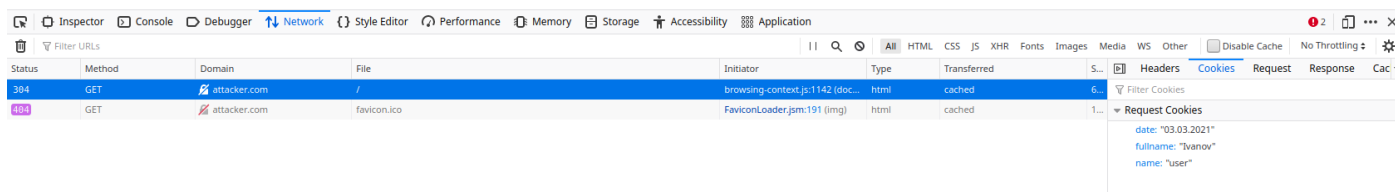
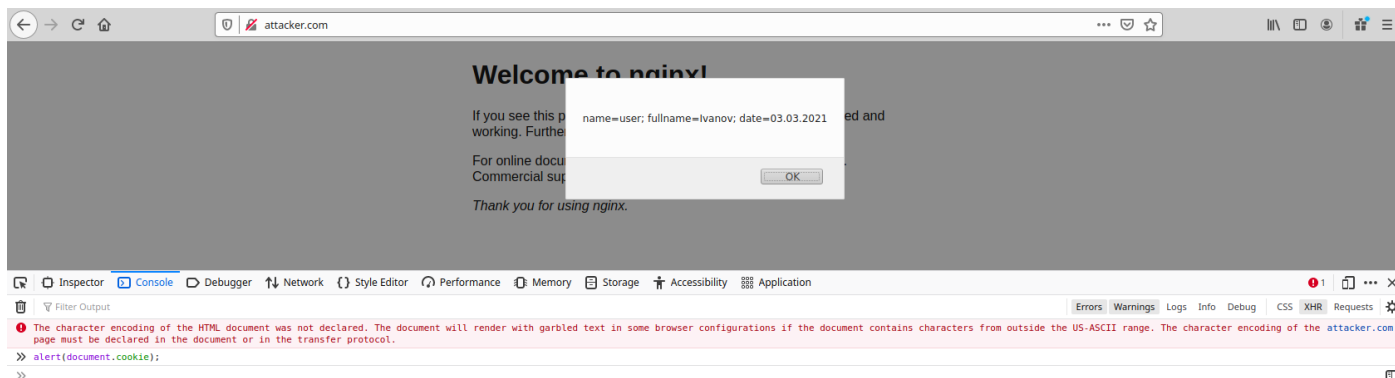


1. Это задание выполняется на домене **attacker.com**. Прочитать куки домена **attacker.com** и вывести их. Попробовать прочитать и вывести куки домена **victim.com**.

Чтобы прочитать и вывести куки домена **attacker.com** используем команду `alert(document.cookie)`; в консоли браузера:



The screenshot shows the Chrome DevTools Storage tab. On the left, the 'Cookies' section is expanded, showing a list of cookies for the domain 'http://attacker.com'. The main table displays the following cookies:

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
date	03.03.2021	attacker.com	/	Session	14	false	false	None	Wed, 03 Mar 2021 18:14:...
fullname	Ivanov	attacker.com	/	Session	14	false	false	None	Wed, 03 Mar 2021 18:14:...
name	user	attacker.com	/	Session	8	false	false	None	Wed, 03 Mar 2021 18:14:...

The right sidebar shows the details for the selected 'name' cookie:

- name:** "user"
- Created:** "Wed, 03 Mar 2021 18:00:25 GMT"
- Domain:** "attacker.com"
- Expires / Max-Age:** "Session"
- HostOnly:** true
- HttpOnly:** false
- Last Accessed:** "Wed, 03 Mar 2021 18:14:21 GMT"
- Path:** "/"
- SameSite:** "None"
- Secure:** false
- Size:** 8

По умолчанию куки доступны лишь тому домену, который его установил. Куки, которые установил сайт **victim.com**, не будут доступны сайтом **attacker.com**. Нет способа сделать куки доступным на другом домене 2-го уровня, так что **attacker.com** никогда не получит куки, установленное сайтом **victim.com**

Но сайт *attacker.com* получит куки, сайта *sub.attacker.com*, если настроить установку кук для доменов. Опция *domain* позволяет нам разрешить доступ к куки для поддоменов, добавив конфигурационный файл *nginx* "*Domain=sub.attacker.com*":

```
server {
    listen 80;
    server_name attacker.com;
    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    location / {
        add_header "Set-Cookie" "Domain=victim.com";
        add_header "Set-Cookie" "Domain=sub.attacker.com";
        try_files $uri $uri/ =404;
    }
}
```

Проверяем:

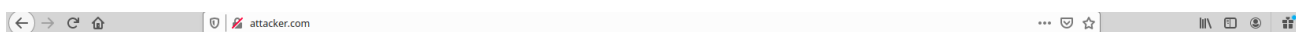
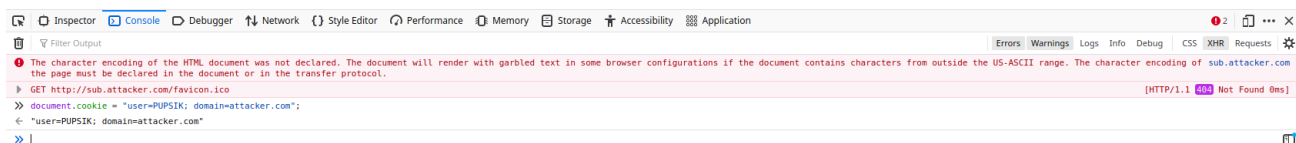


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

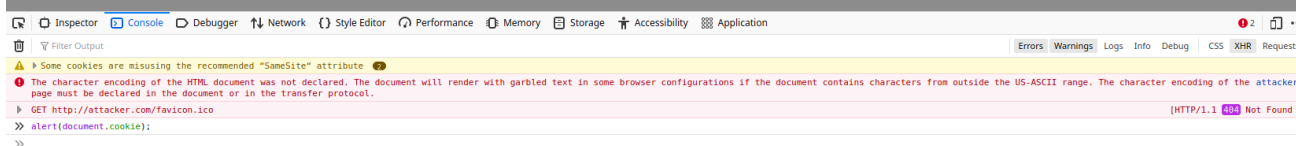


Welcome to nainyl

If you see this page working. Further co

For online documents, visit www.pearsoncmg.com
Commercial support: csupport@pearsoncmg.com

Thank you for using nginx.



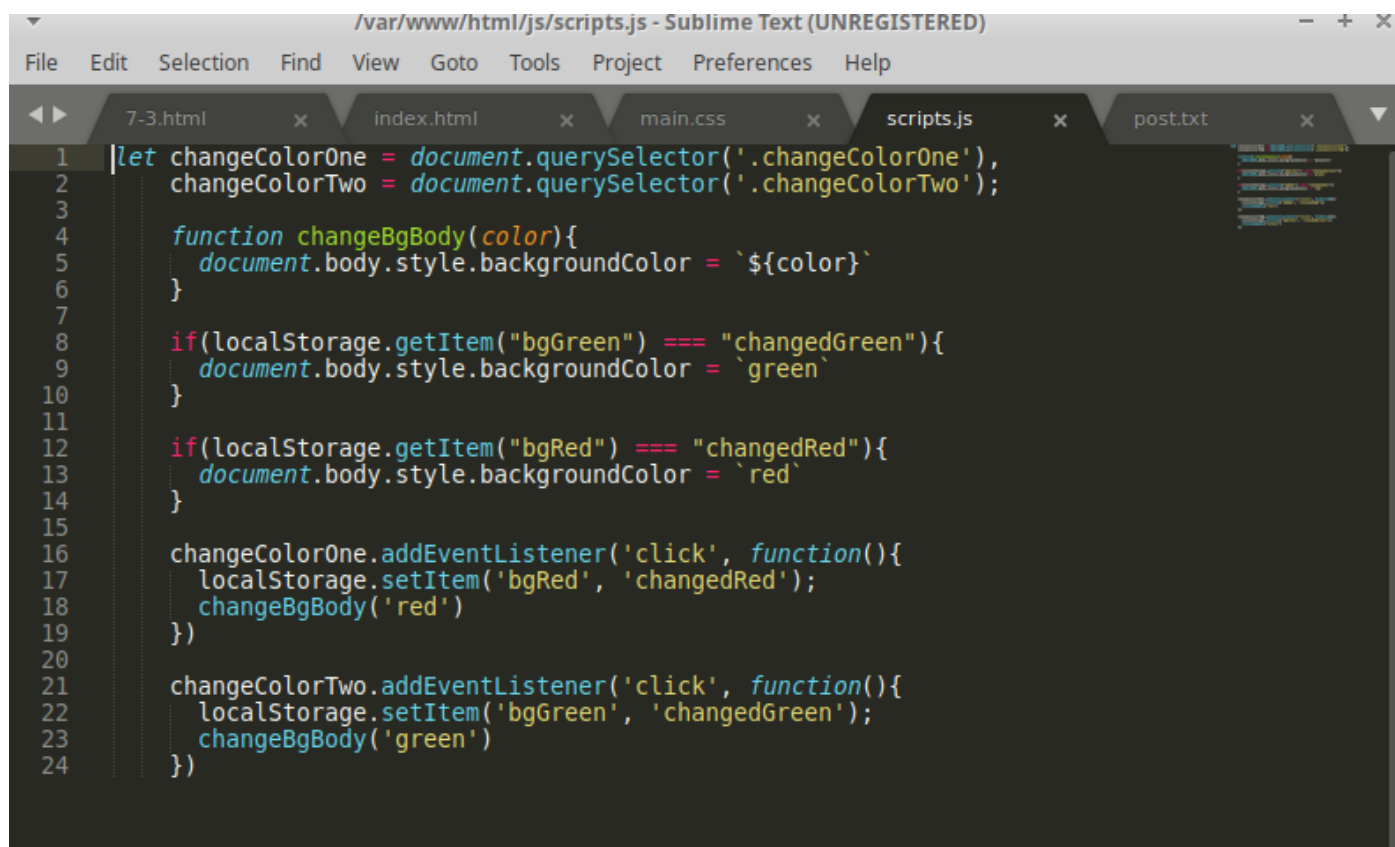
домене *attacker.com*. мы смогли прочитать и вывести куки домена *sub.attacker.com*

2. Дан сайт, который при нажатии на кнопку меняет цвет фона. Дописать, чтобы при открытии сайта JS обращался в web storage за цветом фона и восстанавливает его.

```
<body>
<script>
  function changeBodyColor(color) {
    document.body.style.backgroundColor = color;
  }
</script>
<button onclick="changeBodyColor('red')">Make it hell!</button>
<button onclick="changeBodyColor('green')">Make it grass!</button>
</body>
```

Чтобы записать данные в хранилище web storage, применим следующий способ `Storage.setItem()`

Чтобы получить данные из хранилища web storage, применим следующий способ `Storage.getItem()`

A screenshot of a Sublime Text editor window titled "/var/www/html/js/scripts.js - Sublime Text (UNREGISTERED)". The editor shows a JavaScript file named "scripts.js" with the following code:

```
1 let changeColorOne = document.querySelector('.changeColorOne'),
2   changeColorTwo = document.querySelector('.changeColorTwo');
3
4 function changeBgBody(color){
5   document.body.style.backgroundColor = `${color}`
6 }
7
8 if(localStorage.getItem("bgGreen") === "changedGreen"){
9   document.body.style.backgroundColor = `green`
10 }
11
12 if(localStorage.getItem("bgRed") === "changedRed"){
13   document.body.style.backgroundColor = `red`
14 }
15
16 changeColorOne.addEventListener('click', function(){
17   localStorage.setItem('bgRed', 'changedRed');
18   changeBgBody('red')
19 })
20
21 changeColorTwo.addEventListener('click', function(){
22   localStorage.setItem('bgGreen', 'changedGreen');
23   changeBgBody('green')
24 })
```

The code defines two event listeners for buttons with classes ".changeColorOne" and ".changeColorTwo". When the first button is clicked, it sets a localStorage item "bgRed" to "changedRed" and calls the "changeBgBody" function with "red". When the second button is clicked, it sets "bgGreen" to "changedGreen" and calls "changeBgBody" with "green". The "changeBgBody" function updates the document body's background color based on the passed color parameter. The code also includes checks for existing localStorage items to restore the background color when the page is loaded.

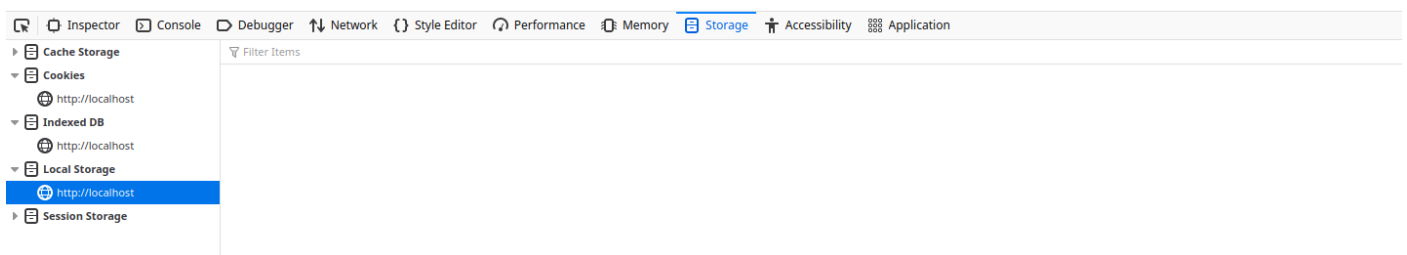
```

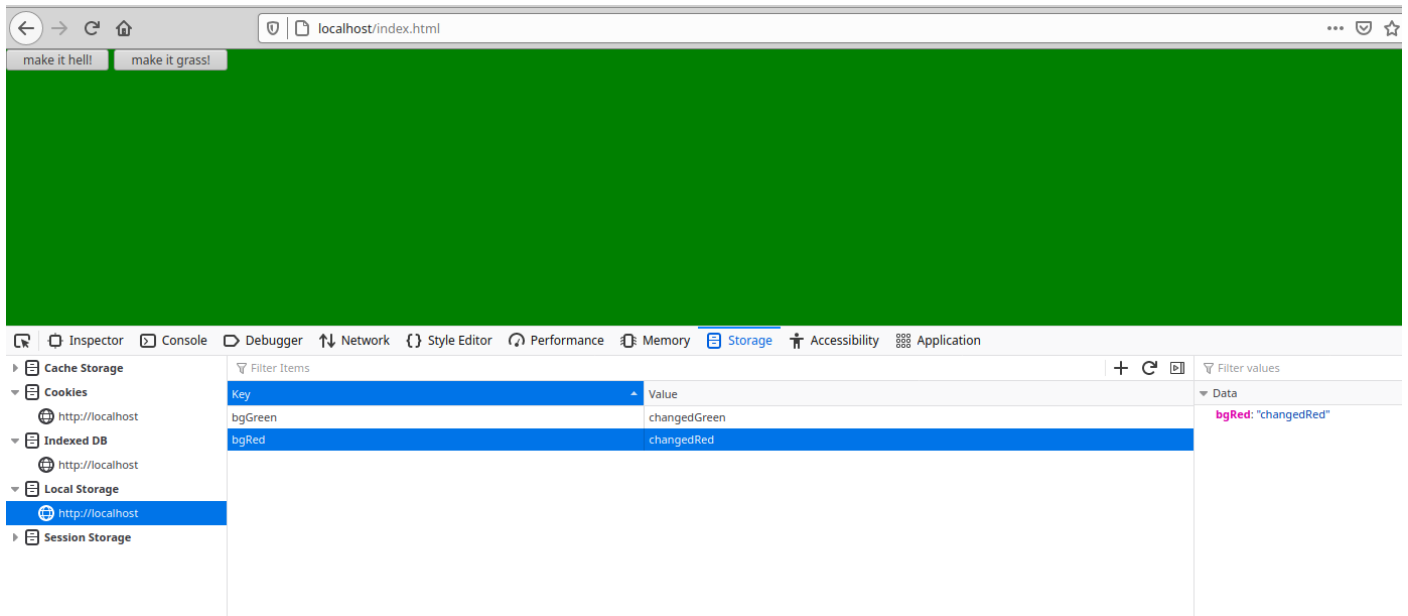
/var/www/html/index.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

7-3.html x index.html x main.css x scripts.js x post.txt x
1 <!DOCTYPE html>
2 <html lang="ru">
3
4 <head>
5
6 <meta charset="utf-8">
7 <!-- <base href="/" -->
8
9 <title>Lesson-1</title>
10 <meta name="description" content="">
11
12 <meta http-equiv="X-UA-Compatible" content="IE=edge">
13 <meta name="viewport" content="width=device-width, initial-scale=1,
14 maximum-scale=1">
15
16 <!-- Custom Browsers Color Start -->
17 <meta name="theme-color" content="#da9a17">
18 <!-- Custom Browsers Color End -->
19
20 <link rel="stylesheet" href="css/main.css">
21
22 </head>
23
24 <body>
25
26 <button class="changeColorOne">make it hell!</button>
27 <button class="changeColorTwo">make it grass!</button>
28 <script src="js/scripts.js"></script>
29
30 </body>
31 </html>

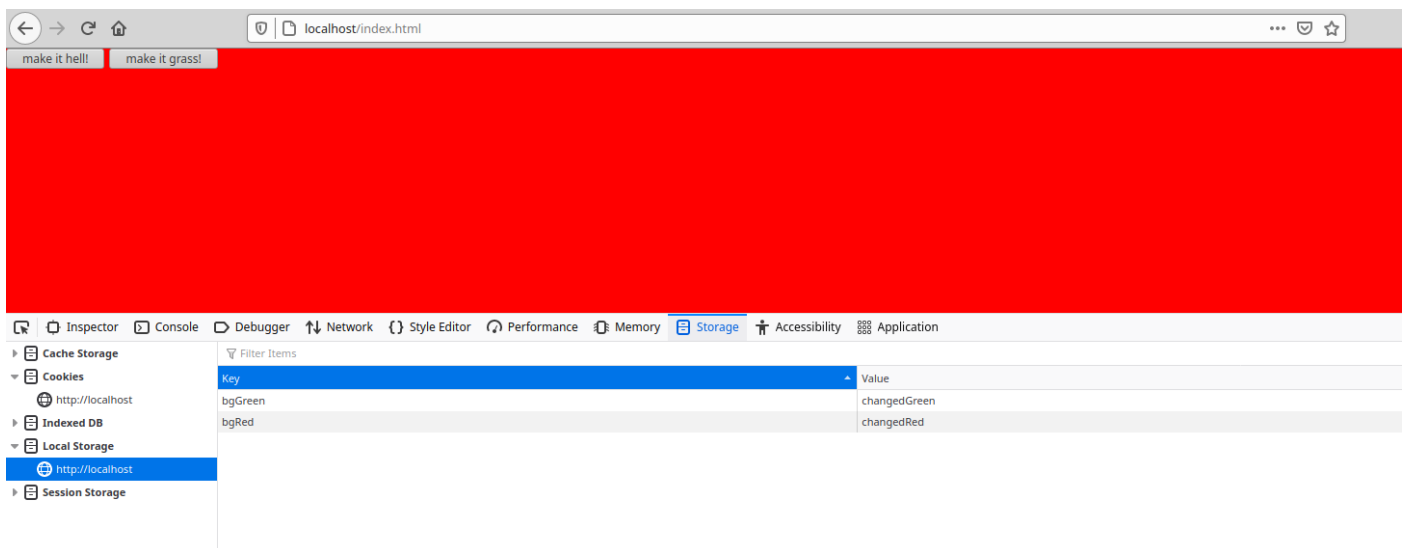
```

Проверяем в браузере:





Данные записаны в Local Storage, далее закрываем и открываем сайт:



Отлично, при открытии сайта JS обращается в web storage за цветом фона и восстанавливает его.

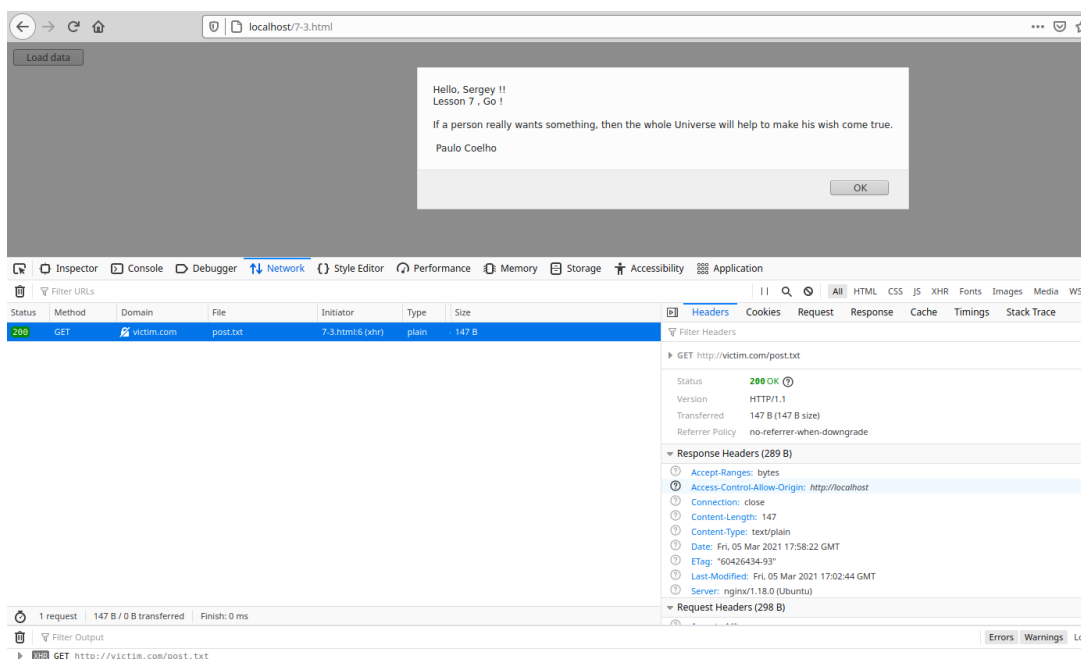
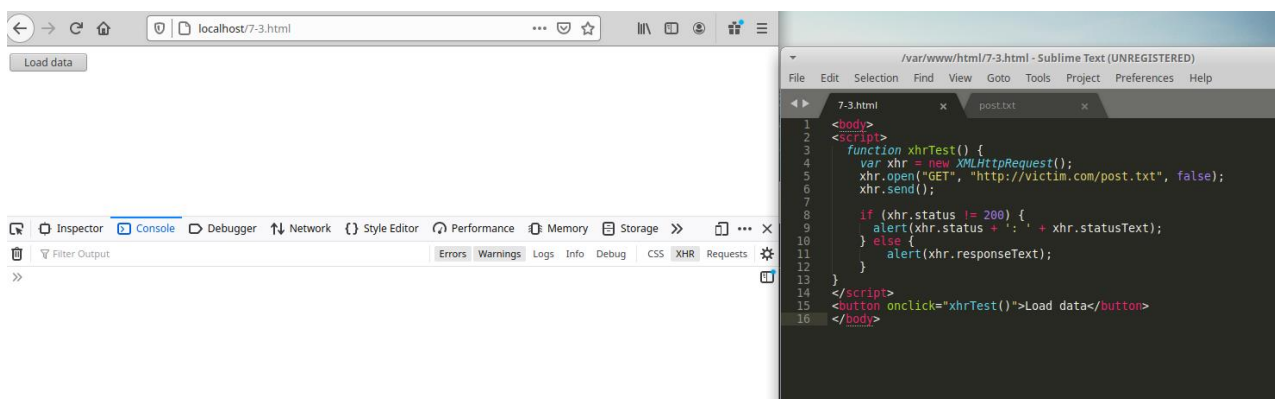
Задание 3. (*) Самостоятельно настроить CORS на <http://victim.com>. Разрешить <http://localhost> с помощью CORS делать запросы к <http://victim.com>.

Чтобы разрешить <http://localhost> делать запросы к <http://victim.com> и читать ответы, нужно настроить CORS в nginx, добавим add_header Access-Control-Allow-Origin <http://localhost>;

```
server_name localhost;

location / {
    # First attempt to serve request as file, then
    add_header Access-Control-Allow-Origin "http://localhost";
    #add_header 'Access-Control-Allow-Methods' "GET, POST, OPTIONS, DELETE, PUT";
```

ПОП больше не будет запрещать получать ресурсы с <http://victim.com> с помощью запросов отправленных с <http://localhost>

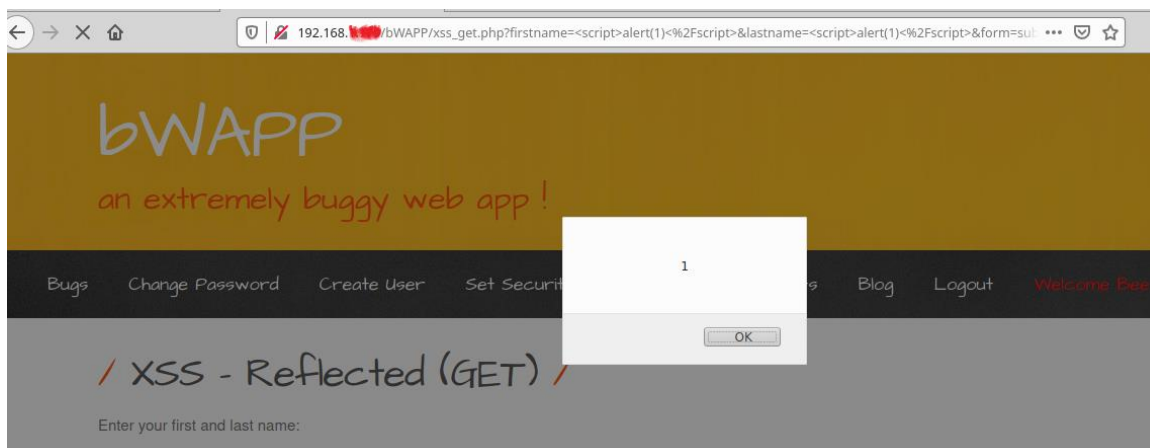


Отлично, получен ресурс post.txt от <http://victim.com>

Задание 4. (*) Решить как можно больше XSS на уровне low в bWAPP.

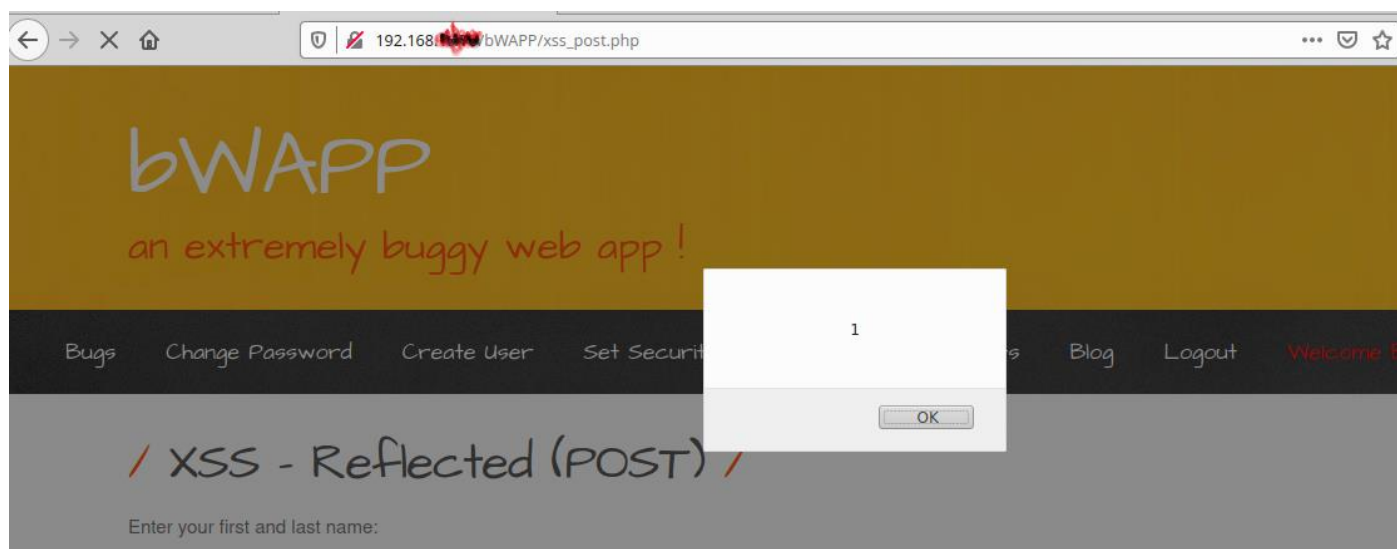
Cross-site-Scripting-Reflected (GET) :

поставим полезную нагрузку в оба поля `<script>alert(1)</script>`



Cross-site-Scripting-Reflected (POST) :

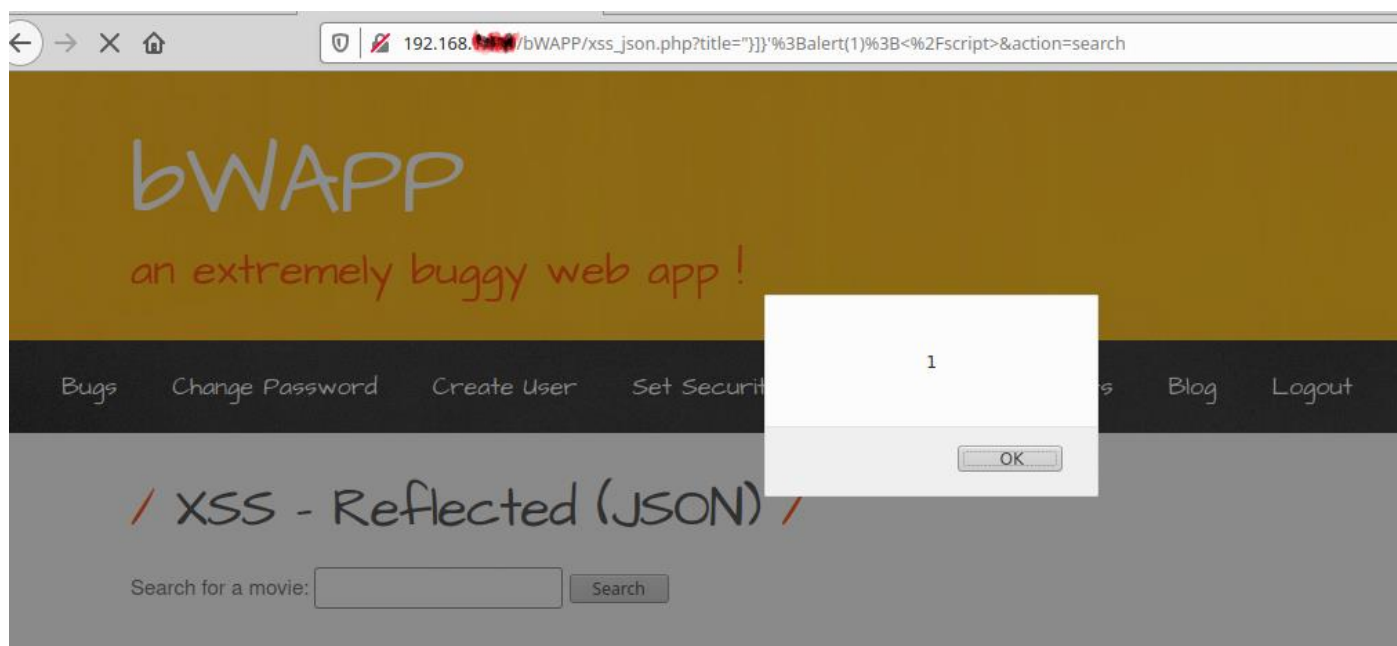
поместим полезную нагрузку в оба поля `<script>alert(1)</script>`



Cross-site-Scripting-Reflected (JSON) :

Ответ веб-приложения будет распечатан JS в теге JS, его можно обойти, закрыть текущий оператор JS и ввести новую строку вредоносного кода.

поместим полезную нагрузку в поле `"}}';alert(1);</script>`





Cross-site-Scripting-Reflected (EVAL) :

пместим полезную нагрузку напрямую, просто поместим `iot` на url-адрес `at date=payload` at address bar `alert(1)`

