СПОСОБЫ АТАК ЧЕРЕЗ WEB-БРАУЗЕРЫ

В данном докладе будет перечислены основные виды атак на веб-ресурсы, а также показаны на практике некоторые способы защиты для нейтрализации этих атак. Будут рассмотрены несколько из самых распространенных браузеров, некоторые аспекты их безопасности.

На сегодняшний день существует гигантское количество разнообразных веб-сайтов, а также веб-приложений. Ими пользуется подавляющий процент населения планеты. Современный человек не мыслит своей жизни без смартфона, планшета или компьютера. Но без подключения к Интернету эти девайсы достаточно бесполезны.

Получать информацию, делиться своей, продавать товары – все это удобно делать через упомянутые выше веб-сайты и веб-приложения.

Что бы использовать их, пользователь использует браузер – это специальное прикладное программное обеспечение, которое применяется для взаимодействия с информационными веб-ресурсами через гаджеты которые подключены к сети Интернет.

Независимо от того, какой браузер выбрал пользователь, механизм работы каждого из них будет одинаковым.

1. Пользователь запускает браузер на своем устройстве и вводит адрес

необходимого сайта в адресную строку.

1. Браузер начинает поиск сервера. Поиск выполняется по ip-адресу, уникального для каждого веб-ресурса. Браузер связывается с DNS-сервером по полученным данным (строке, которую ввел пользователь). Например когда пользователь вводит google.com, то это соответствует ip-адресу 216.58.207.110.
2. Далее выполняется соединение с сервером с помощью протокола TCP/IP (которой отвечает за передачу данных в сети).
3. После установки соединения, на сервер отправляется запрос HTTP, который запрашивает информацию для отображения пользователю на экране его устройства (используются GET и POST-запросы).
4. После этого сервер выполняет запрос и отправляет браузеру интерфейс, который видит пользователь, а также различные скрипты и данные о файлах cookie.
5. Браузер принимает ответ и рендерит страницу которую уже непосредственно видит пользователь.

Кроме основной функции, которая описана по шагам выше, браузер подразумевает выполнение других, не менее важных задач. Приведем некоторые из них:

1. Дает возможность скачивать на телефон или компьютер различные файлы разного типа – видео, аудио, различная графическая информация и различные текстовые форматы
2. Позволяет сохранять всю вашу активность – все сайты, которые вы посетили отображаются в истории браузера.
3. Как можно догадаться, пользователь имеет возможность сохранить какие-то посещенные страницы в закладки, это позволит сэкономить время – пользователю не надо искать сайт в поисковых системах или же вбивать в адресную строку название сайта.
4. Сохранение различной информации (например использование куки). Куки (cookie) - это переменная, которая хранится на стороне клиента, то есть в браузере пользователя. Каждый раз, когда пользователь пытается открыть веб-страницу в браузере, он посылает куки. Взаимодействие с куки возможно организовать с использованием JavaScript.

Примеры куки: куки с пользовательским логином.

Когда посетитель заходит на сайт впервые, ему часто предлагается представиться – пользователь вводит в форму свой логин, далее, в случае если пользователь ещё раз посещает данный сайт, то он может получить персонализированное приветствие – все из-за того, что будет отображаться занесенный в куки при первом посещении веб-ресурса логин.

1. Браузер поддерживает различные дополнения – от различных тем оформления браузера до различных виджетов. Можно поставить какие-то расширения, которые будут автоматически переводить иностранные слова, которые встречаются на открытом пользователем сайте.

Взлом сайтов и веб-приложений является наиболее частым методом кибератаки на различные компании, а также на различных частных лиц. Взломанные сайты используются для различных целей злоумышленников, от распространения на взломанном ресурсе заведомо ложной информации, до кражи личных данных (например из базы данных организации).

Браузер является отличным инструментом для проведения атаки на веб-приложения или сайт.

Как было сказано выше, существует большое количество разных браузеров, кроме дизайна, организации меню и так далее.

Зачастую пользователи отдают предпочтение тому или иному браузеру как раз из-за красивого интерфейса, скорости работы или возможности установить то или иное расширение.

Но достаточно часто забывается один из главных критериев – безопасность браузера, а ведь это очень важно, ведь с использованием браузера мы ведем столько дел – от оплаты каки-то услуг до использование почты, где часто содержится конфиденциальная информация. Правильно выбранный браузер может уменьшить риск той или иной атаки на пользователя.

Рассмотрим несколько браузеров:

1. Apple Safari

В браузере есть очень полезная функция защищенного просмотра. В данном режиме браузер не ведет учет истории посещаемых ресурсов, загружаемого программного обеспечения. Не выполняется сохранения поисковых запросов пользователя, данных веб-форм и куки.

Существует функция, которая блокирует всплывающие окна.

В сам браузер встроены технологии блокировки различных вредоносных сайтов, технологии противодействия атакам XSS.

2. Googel Chrome

Является одним из самых популярных браузеров, миллионы людей используют его каждый день.

Существует защита от фишинговых сайтов, сайтов, которые несут в себе вредоносные программы. В Google Chrome существует технология обеспечения непрерывности HTTPS-соединения и защиты его от компрометации, защита от XSS-атак.

В данном браузере есть защита от компрометации HTTPS-соединений (в отличии от Apple Safari, который был описан выше).

3. Opera

Среди проанализированных, данный браузер является самым плохим в плане безопасности.

В данном браузере, в отличии от большинства существующих решений нет защиты от XSS-атак.

Также нет фильтра вредоносных программ.

4. Mozilla FireFox

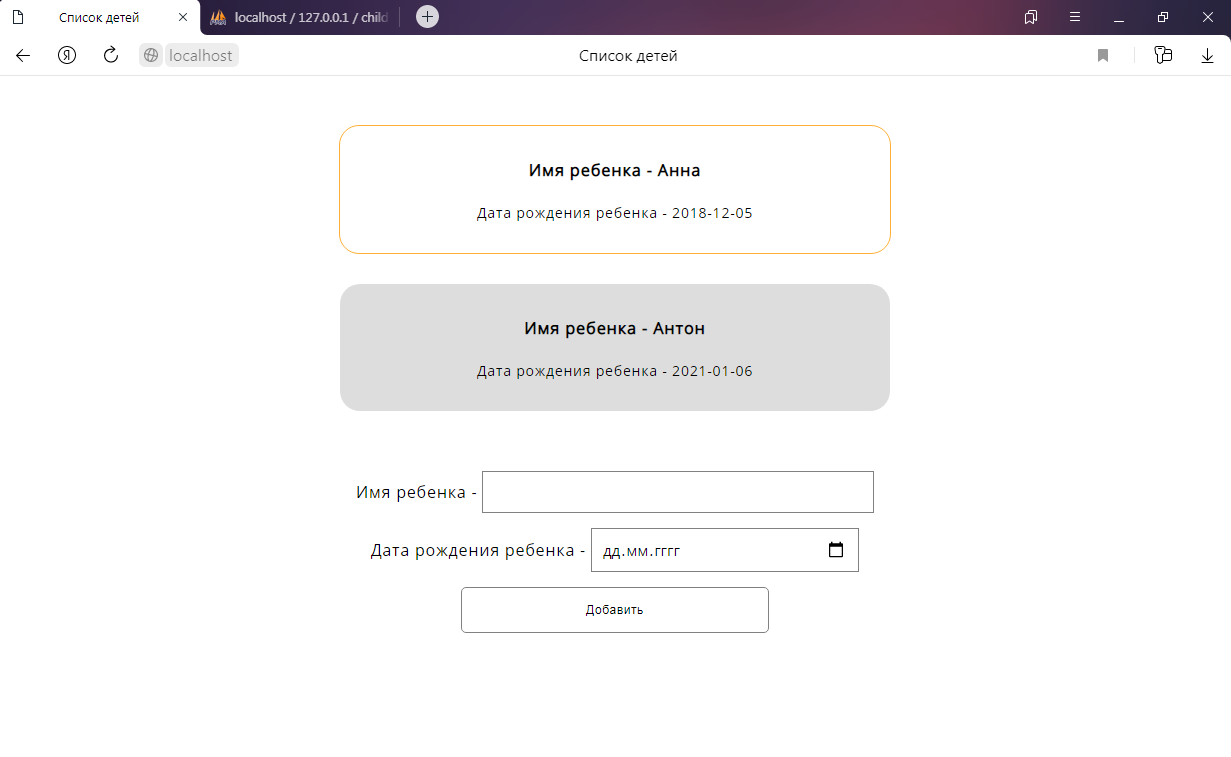
Разработчики данного решения уделили особое внимание вопросам безопасности своего браузера. Есть поддержка EV-сертификатов, защита от XSS-атак, есть возможность интеграции с родительским контролем, который установлен в операционной системе Windows.

Также стоит сказать, что разработчики поощряют разработку различных расширений пользователями браузера. С использованием набора таких расширений безопасность браузера повышается.

Перейдем непосредственно к более подробному рассмотрению атак через веб-браузеры и способах защиты от них.

Предположим, что у нас есть сайт, который ведет учет детей – есть список детей с и именами и датой рождения. Функции пользователя на сайте состоят в возможности просмотра списка детей и возможности добавления ребенка на сайт (все данные сайта хранятся в базе данных на сервере).

На рисунке 1 представлен главная (и единственная) страница сайта. На ней есть форма добавления ребенка – пользователю необходимо ввести имя и дату рождения, после чего нажать на кнопку «Добавить» - и с помощью POST-запроса и sql команды INSERT данные пользователя будут добавлены в базу данных.



*Рис. 1.* – Главная страница сайта

Именно на данном этапе нам нужно защитить базу данных и все приложение в целом от потенциальных кибер-атак.

Одними из наиболее часто встречающихся угроз являются SQL-инъекции.

В ходе проведения SQL-инъекции осуществляется попытка изменения SQL-выражения, которое приложение отправляет базе данных. Заключается она в вставке SQL-команд в формы ввода данных на информационном ресурсе, которые доступны для заполнения пользователями веб-ресурса (в нашем примере это форма добавления ребенка).

Например, пользователь может ввести такой SQL-запрос в поле формы: SELECT table\_name FROM information\_schema.tables; - появится список всех таблиц в базе данных.

После того, как он узнал список таблиц, он может удалить таблицу child таблицы введя в поле формы команду DROP TABLE child;.

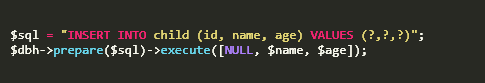
Можно сделать вывод, что требуется любым адекватным способом фильтровать входные данные. Нельзя допустить отправку опасного запроса в базу данных или вывода данных с визуальными ошибками.

Один из вариантов защититься от данного вида атак – подготовленные запросы.

Выполнение подготовленного запроса начинается с формирования шаблона будущего запроса – это стандартное выражение, написанное на языке *SQL*, но в нем не указываются данные, которые передал пользователь на форму. Далее в базу данных (в нашем случае это *MySQL*) передаются значения для шаблона, который был сформирован ранее.   
Первый этап данного процесса называется подготовкой, а второй — выражением. Подготовленный запрос можно выполнять несколько раз, передавая туда разные значения.

Значения привязанных к запросу переменных сервер экранирует автоматически. Привязанные переменные отправляются на сервер отдельно от запроса и не могут влиять на него. Сервер использует эти значения непосредственно в момент выполнения, уже после того, как был обработан шаблон выражения. Привязанные параметры не нуждаются в экранировании, так как они никогда не подставляются непосредственно в строку запроса.

Для нашего сайта такой запрос представлен на рисунке 2.



*Рис. 2* – Подготовленный запрос

Сайты и веб-приложения часто разрабатываются с использованием различных веб-фреймворков. Данные фреймворки в большинстве случаев гарантируют, что пользовательские данные, которые будут передаваться в базу данных сайта будут экранированы (например Django или Symfony).

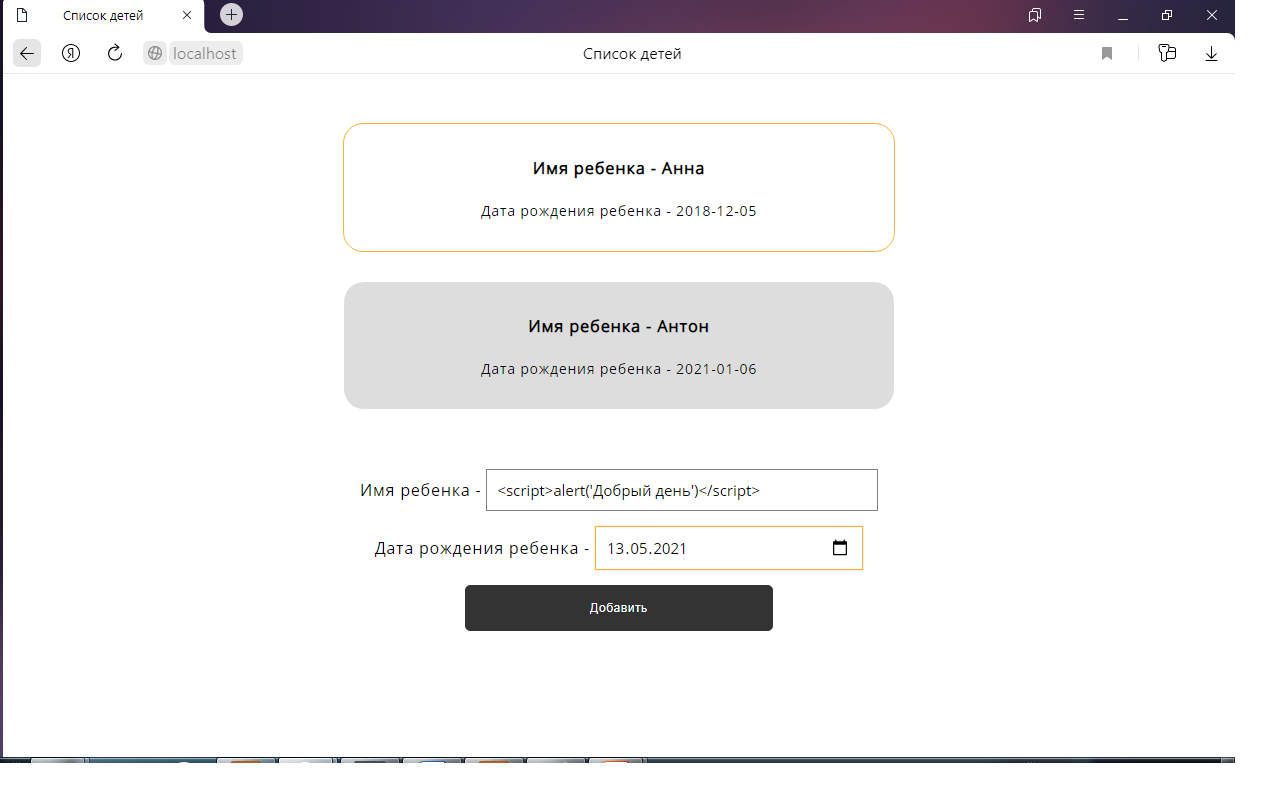
Также, на каждом из создаваемых сайтов и приложений разработчику необходимо предусмотреть использование экранирования, которое сможет предотвратить XSS-атаки.

XSS-атака – данный вид атаки предусматривает выполнение очень вредоносного скрипта написанного на языке javascript*,* в браузерах посетителей, которые зашли на атакованный веб-ресурс.

Один из вариантов реализации этой атаки – это передача вредоносного кода на сервер. Данный код будет срабатывать когда другие пользователи будут заходит на сайт. Например злоумышленник может организовать перенаправление с атакованного ресурса на другой, например на какой-то рекламный сайт. Пользователь, ничего не подозревая, будет заходить на атакованный портал, а его будет сразу же перекидывать на совершенно другой, прописанный злоумышленником сайт.

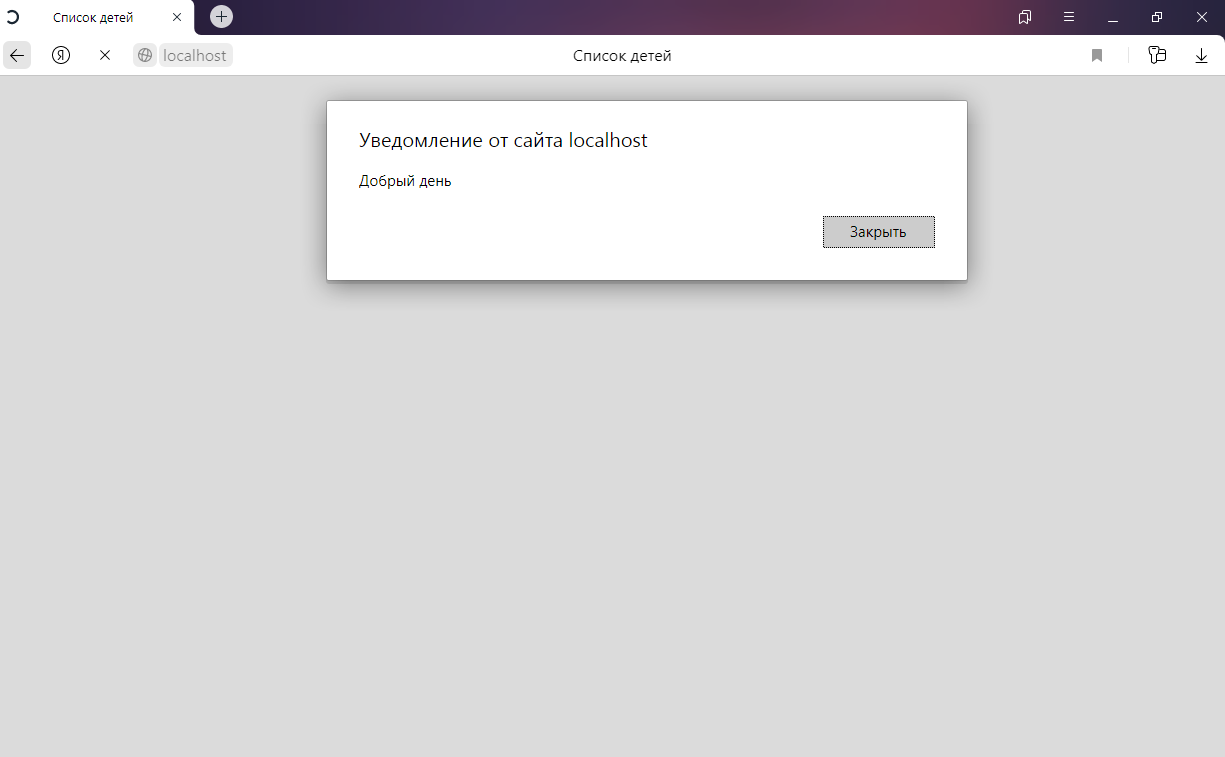
Но как же защититься от данного вида атак?

Вернемся к нашему примеру, предположим, что пользователь введет в форму выражение как на картинке 3.



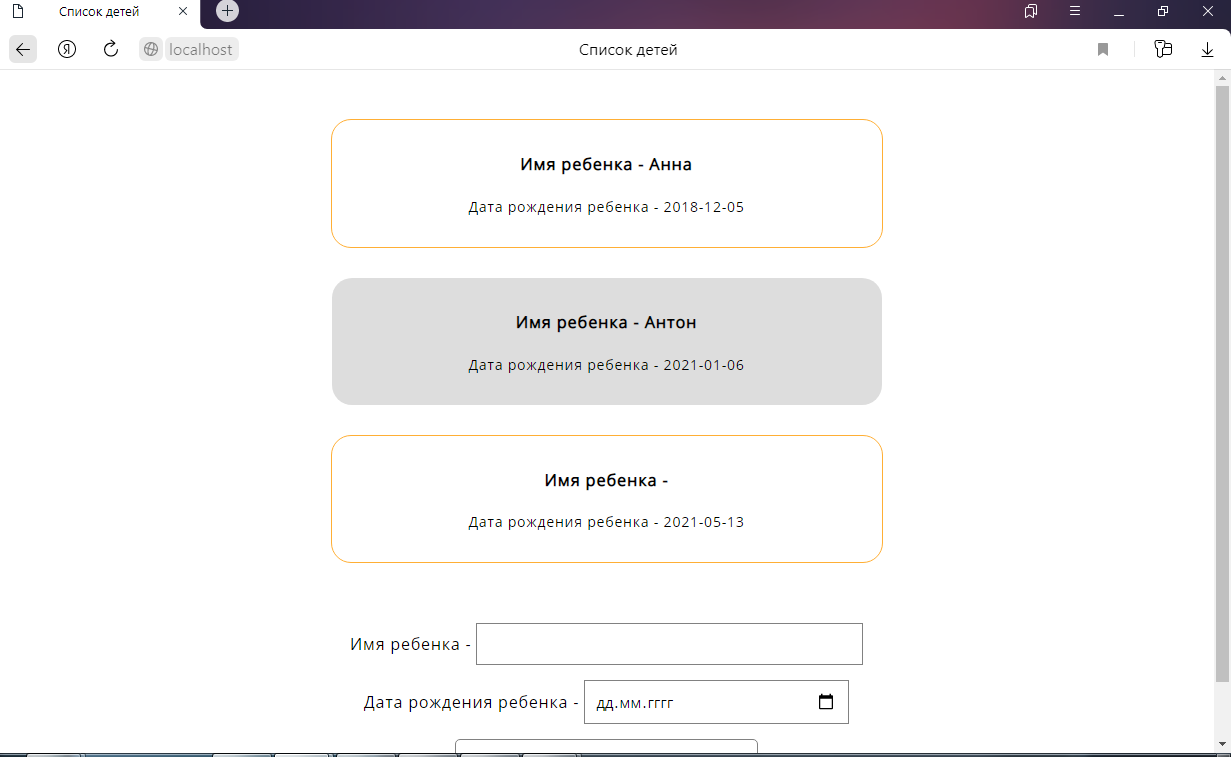
*Рис. 3.* – Запрос с кодом JS

Результат – скрипт *JS* тут же выполнится (представлено на рисунке 4).



*Рис. 4.* – Результат после нажатия кнопки «Добавить»

Также в бд и на странице будет результат как на рисунке 5.



*Рис. 5* – Данные которые ввел пользователь

И запрос будет исполняться при каждой загрузке страницы.

Код, который написан и допускает данную атаку, на рисунке 6. Код, с использованием экранирования (команды *htmlspecialchars*) на рисунке 7.

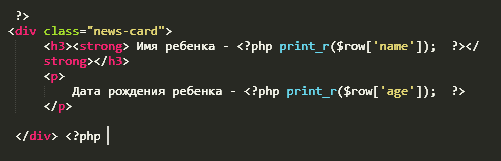
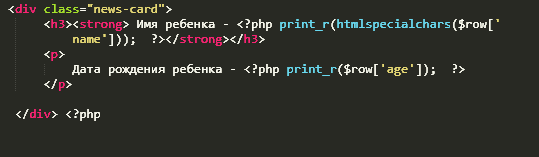


Рис.6. – Код, допускающий вывод скрипта JS на странице

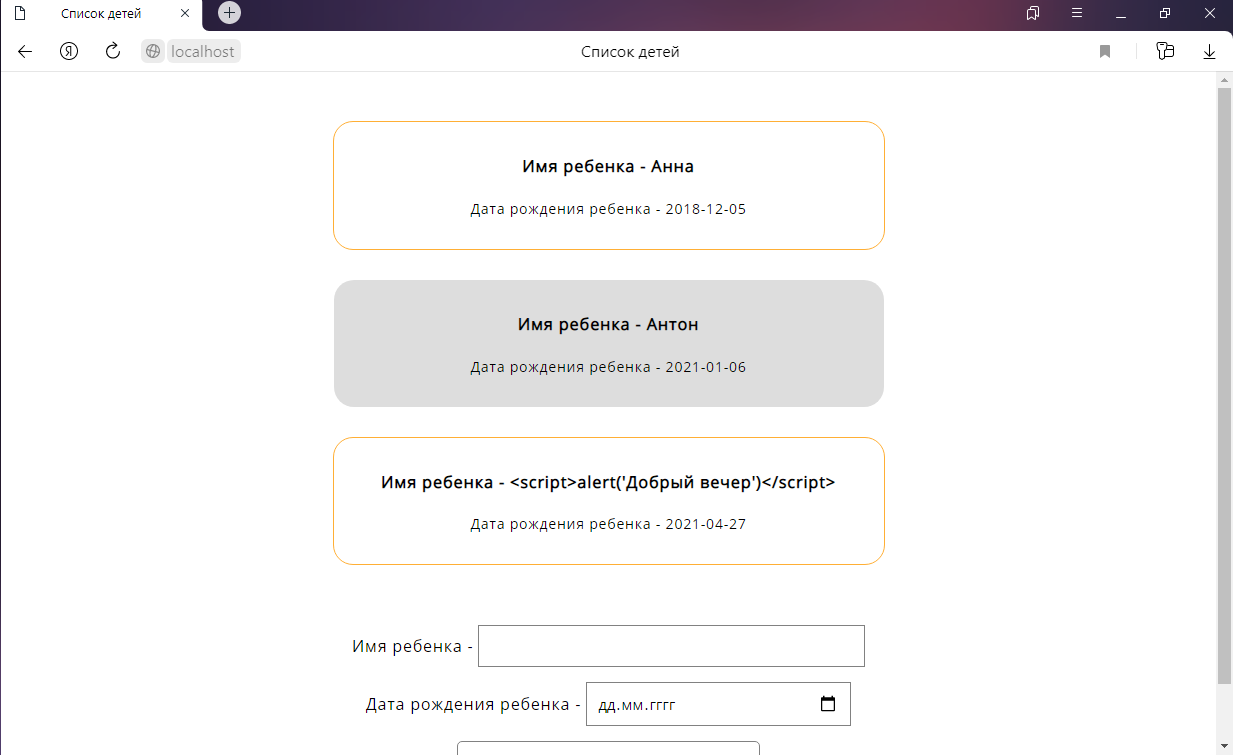
Код с использованием экранирования:



*Рис. 7.* – Код, недопускающий вывод скрипта JS на странице

В результате, на странице будет просто запись команды, которую ввел пользователь, исполнятся она не будет и не причинит никакого вреда. Это представлено на рисунке 8.

Функция *htmlspecialchars()* позволяет преобразовать *html*-теги в специальные сущности. Данная функция используется, главным образом, для преобразования данных, получаемых от пользователя.



*Рис. 8.* – Страница, в коде которой используется htmlspecialchars

Также рассмотрим XPath-инъекцию.

Данный вид атаки становится возможным если веб-сайт использует данные пользователя для создания запроса XPath для данных XML.

Атака организуется подобно SQL-инъекции, только преступники отправляют запрос, который будет взаимодействовать не с базой данных, а с структурой XML.

После успешного получения структуры данных, злоумышленник получает и доступ к этим данных.

Способы борьбы с данным видом атак – это валидация полученных от пользователя данных, а также экранирование специальных символов.

Также часто встречается Dos-атака на сайты.

Это непосредственно хакерская атака, которая несет в себе цель сделать недоступным посещение веб-ресурса для обычных посетителей.

Как было сказано выше, каждый веб-ресурс размещен на сервере, куда приходят запросы от пользователей, которые заходят на портал. У каждого сервера есть какой-то лимит информации, которую он способен обработать за единицу времени. Данный лимит находится в прямой зависимости от емкости диска, параметров трафика, или же, если сайт размещен на хостинг-провайдера, то от условий подписки на услуги этой компании.

Если на данных сервер в одну и ту же единицу времени приходит большое количество запросов, то он сначала начинает тормозить, а при превышении лимита перестает корректно работать.

Защита от данного вида атак предусматривает:

* **Создание PHP-скрипта, который будет анализировать IP-адреса, с которых поступают запросы. В случае, если с одного IP-адреса приходят запросы в промежутки времени, нетипичные для человека, то пользователю с данным IP-адресом блокируется доступ к сайту.**
* **Использование специального сервиса для очистки от спамного трафика. К доменному имени ресурса добавляется специальный DNS-сервер. В обязанности этого сервера и входит фильтрация запросы (запросы сначала перенаправляются на этот сервис, безопасные пакеты потом снова перенаправляются на ресурс, а подозрительные пакеты подлежат блокировке).**
* **Использование файла .htaccess. Если на ресурсе присутствует данный файл, то можно ограничить доступ к сайту по определенным ограничениям по IP-адресам и определенным признакам в запросе от лица, которое посылает запрос, желая получить доступ на сайт.**

Рассмотрим ещё одну атаку, достаточно часто встречающуюся в мире браузеров. Это CSRF-атака.

Данный вид атаки подразумевает выполнение преступником различных действий из под учетных данных другого пользователя сети Интернет (без уведомления его об этом и получения его согласия на данные действия).

Для того что бы лучше понять данный вид атаки, необходимо привести пример.

Скажем, у нас есть злоумышленник Иван, который узнал, что какой-то конкретный ресурс позволяет пользователям, прошедшим авторизацию (а перед этим регистрацию, что естественно), отправлять деньги из своего личного кабинета другому пользователю на банковские реквезиты. Данный перевод выполняется, используя запрос HTTP-запрос POST, который включает в себя логин пользователя и сумму денег.

Иван может создать форму, которая включает в себя его банковские реквизиты и сумму денег в форме скрытых полей.(скажем, если у html-тега input поставить type – hidden, то эти элементы формы не будут заметны и пользователь не сможет их заметить).

Данную форму Иван может отправить по электронной почте какому-то пользователю вышеописанного ресурса, замаскировав её под какое-нибудь безобидное уведомление (например кнопку формы, которая инициирует запрос POST Иван может замаскировать под ссылку, которую его ничего не подозревающая жертва может открыть).

Если пользователь переходит по этой ссылке, и он авторизован на вышеописанном ресурсе (например, с использованием куки), то на сервер ресурса будет отправлен запрос, содержащий сведения о банковских данных Ивана и сумме, которую надо перевести на данный счет, а также содержащий файлы куки на стороне клиента жертвы, которые браузер связал с сайтом (добавление связанных файлов куки сайта в запросы является нормальным поведением браузера). Сервер проверит файлы куки и использует их, чтобы определить, вошёл ли пользователь в систему и имеет ли разрешение на совершение транзакции. В результате авторизованная на сайте жертва Ивана совершает транзакцию из своего личного кабинета на ресурсе и посылает деньги Ивану.

Также часто встречающейся атакой является LFI (Local file include).

Данный вид атак несет очень высокую угрозу для сайта, который атакуют, уязвимость позволяет удаленному пользователю получить доступ с помощью специально сформированного запроса к произвольным файлам на сервере в том числе содержащую конфиденциальную информацию. Возможность использования локальных системных файлов сервера. LFI возникает в случае, когда проверка входящих данных и параметров в коде сайта отсутствует или недостаточна.

Возникает эта проблема из-за грубейших ошибок разработки сайта, отсутствия фильтрации передающих параметров, а также некорректной настройки сервера.

Последним типом атак, рассмотренных в данной работе, будет Clickjacking.

В этой атаке злоумышленник перехватывает клики, предназначенные для видимого сайта верхнего уровня, и направляет их на скрытую ниже страницу. Этот метод можно использовать, например, для отображения законного сайта банка, но захвата учётных данных для входа в невидимый <iframe>, контролируемый злоумышленником.

Эту атаку также можно использовать для того, чтобы заставить пользователя нажать кнопку на видимом сайте, но при этом на самом деле невольно нажимать совершенно другую кнопку. В качестве защиты ресурс может предотвратить встраивание себя в iframe на другом сайте, установив соответствующие заголовки HTTP, это нужно реализовать на этапе разработки сайта.

1. 9 популярных типов атак на веб-приложения. [Электронный ресурс] / Сайт по изучению веб-разработки – Режим доступа - https://www.lordlikely.com/bezopasnost/9-populjarnyh-tipov-atak-na-veb-prilozhenija/ - Дата доступа: 26.10.2021

2. XPath injection. [Электронный ресурс] / Сайт по изучению веб-безопасности – Режим доступа - https://insafety.org/xpath.php - Дата доступа: 26.10.2021

3. Бондарев.В. В. Введение в информационную безопасность автоматизированных систем : учебное пособие / В. В. Бондарев. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2016. — 250 с.

4. Петров С.В. Информационная безопасность. : учебное пособие / С.В. Петров И.П. Слинькова,В.В. Гафнер - АРТА, Москва 2012. 296 c

5. Глинская, Е.В. Информационная безопасность конструкций ЭВМ и систем: учебное пособие / Е.В. Глинская, Н.В. Чичварин. - Москва: Инфра-М, 2018. - 160 c.

6. Гафнер, В.В. Информационная безопасность: Учебное пособие / В.В. Гафнер. – Ростов-на-Дону: Феникс, 2010. - 324 c.

7. Мельников, Д.А. Информационная безопасность открытых систем: учебник / Д.А. Мельников. - Москва: Флинта, 2013. - 448 c.

8. Чипига, А.Ф. Информационная безопасность автоматизированных систем / А.Ф. Чипига. - Москва: Гелиос АРВ, 2010. - 336 c.

9. Партыка, Т.Л. Информационная безопасность: Учебное пособие / Т.Л. Партыка, И.И. Попов. - Москва: Форум, 2012. - 432 c.

10. Семененко, В.А. Информационная безопасность / В.А. Семененко. - Москва: МГИУ, 2011. - 277 c.