**[Слайд 1]**

Добрый день, уважаемые члены комиссии. Меня зовут Кудин Артемий Юрьевич, я студент группы ОБС 21-2. Подготовил димпломную работу, на тему “Анализ современных угроз информационной безопасности на веб-сайтах и способы их предотвращения”.

**[Слайд 2]**

Целью работы является проанализировать современные угрозы безопасности на веб-сайтах и выявить способы их предотвращения.

**[Слайд 3]**

В соответствии с целью, были поставлены следующие задачи:

* Создать среду для тестирования методов защиты от атак;
* Провести атаки на тестовую среду;
* Обеспечить защиту тестовой среды от атак.

**[Слайд 4]**

**Межсайтовый скриптинг**

Межсайтовый скриптинг (XSS или *Cross-Site Scripting* ) - это тип атаки на веб-страницу, заключающийся во внедрении в веб-страницу вредоносного кода, который будет выполнен на устройстве пользователя при открытии им этой страницы

**[Слайд 5]**

Рассмотрим некоторые способы предотвращения XSS атак:

* **Экранирование и кодирование вывода данных**;
* **Использование HTTP заголовков Content Security Policy (CSP);**
* **Валидация и фильтрация ввода данных;**

И другие

**[Слайд 6]**

**SQLI**

SQL Инъекции (SQLI) - это тип атаки на веб-приложения, при которой злоумышленник внедряет SQL-запросы в поля ввода или другие механизмы общения с базой данных.

**[Слайд 7]**

Способы предотвращения SQL Инъекций, включают в себя способы предотвращения JavaScript XSS, а также:

* **Применение принципа минимальных привилегий;**
* **Использование параметризованных запросов;**

И другие

**[Слайд 8]**

CSRF (Cross-Site Request Forgery) — это тип атаки, при котором злоумышленник заставляет пользователя выполнить нежелательный запрос на доверенном сайте, где тот уже авторизован. В результате сайт может обработать этот запрос от имени пользователя, не подозревая, что инициатором был атакующий.

**[Слайд 9]**

Рассмотрим некоторые способы предотвращения CSRF атак:

* **Использование CSRF токена**;
* **Использование SameSite - Strict**
* **Проверка заголовка Authorization вместо cookie для авторизации**

**[Слайд 10]**

**Используемые технологии для разработки тестовой среды**

Начнем с технологий, используемых на стороне клиента.

**HTML;**

**CSS**;

**JavaScript;**

**[Слайд 11]**

Для серверной части проекта, было принято решение использовать следующие технологии:

**MySQL;**

**PHP;**

**[Слайд 12]**

**Применение XSS и защита от атаки.**

Для работы с XSS была разработана веб-страница, которая имитирует блок комментариев в интернете.

На странице присутствует поле ввода, через которое пользователь может ввести текст и сделать запись в базе, а также список комментариев каждый из которых представляет одну запись в базе данных.

**[Слайд 13]**

Рассмотрим примеры внедрения вредоносного кода в базу данных. Пользователь вводит в поле ввода HTML тэг <img/> с ссылкой на картинку, и отправляет эти данные на сервер.

После отправки запроса на сервер, картинку можно увидеть в последнем комментарии на странице.

**[Слайд 14]**

Чтобы избавиться от данной уязвимости, было принято решение, использовать PHP функцию “htmlspecialchars”, которая экранирует данные, превращая специальные символы в HTML сущности. При переходе на страницу, можно наблюдать результат экранирования.

**[Слайд 15]**

Для работы с SQL Инъекциями была разработана веб-страница, которая имитирует форму авторизации.

**[Слайд 16]**

После того как пользователь вводит данные в форму, его перенаправляет на страницу с сообщением о статусе авторизации: “Ошибка!” либо “Вход успешно выполнен”.

**[Слайд 17]**

Рассмотрим ситуацию, когда злоумышленник с помощью SQL инъекции может успешно авторизоваться.

Чтобы злоумышленнику добиться работы его введённого вредоносного кода, ему достаточно вписать в поле ввода строчную кавычку, которая позволит коду после нее выполниться как часть запроса.

Данное выражение означает, что после закрытой строки, благодаря кавычке, выполняется оператор OR, код после которого, в итоге позволит злоумышленнику войти в систему. После того, как злоумышленник отправит данные, его перенаправит на страницу, где он увидит сообщение о том, что вход выполнен успешно.

**[Слайд 18]**

Чтобы защититься от взлома, было принято решение использовать параметризованный запрос. Такой подход предотвращает выполнение вредоносных SQL-инструкций, подставляемых через пользовательский ввод.

**[Слайд 19]**

Для тестирования CSRF атаки была создана страница профиля пользователя. При входе на страницу пользователь видит свой логин, и может ввести новый логин или новый пароль. Также на сайт внедрен функционал авторизации. При авторизации пользователь получает уникальный токен, который помещается в базу данных, и с помощью которого в будущем идентифицируется.

**[Слайд 20]**

Также, для демонстрации CSRF, была создана страница, на которой расположена кнопка, по нажатию на которую происходит отправка формы на адрес сервера. В этой форме есть уже заполненные поля “login” и “password”.

В момент отправки формы происходит запрос на сервер от лица жертвы, благодаря чему к запросу прикрепляются ее cookie.

Сервер проверяет принадлежность токена пользователю и обновляет его данные на те, что указал злоумышленник в своей форме.

**[Слайд 21]**

Так как все запросы проходят в рамках одного домена, для предотвращения поддельных запросов был добавлен атрибут SameSite со значением Strict, что делает cookie недоступными при междоменных запросах, тем самым защищая от CSRF.

**[Слайд 22]**

**Заключение**

Подводя итоги, были рассмотрены актуальные уязвимости веб-безопасности, и методы их предотвращения. Также, была проведена практическая работа с уязвимостями внедрения вредоносного кода на веб-страницы. Была разработана тестовая среда, на которой были протестированы XSS, CSRF, SQLI атаки и способы их предотвращения.

По итогам работы, все следующие задачи были выполнены:

**[Слайд 23]**

Спасибо за внимание! Готов выслушать ваши вопросы.