Актуальность и практическая значимость
В связи с интенсивным развитием информационных технологий появляются и новые риски информационной безопасности. Ввиду возрастающей доступности интернета, веб-сайты заводят даже пользователи, далекие от понятия информационных технологий и уж тем более, защиты информации. Как следствие, огромное количество интернет-ресурсов и веб-сервисов подвержены уязвимостям, наличие которых может привести к серьезным неблагоприятным последствиям. Чтобы сократить количество преступлений в области информационной безопасности, необходимо своевременно выявлять наличие уязвимостей для дальнейшего их устранения. Для решения этой задачи нужен сервис, автоматически выполняющий аудит веб-приложения на наличие уязвимостей, простой в использовании и понятный пользователям с любым уровнем подготовки.

Обзор существующих решений

Рассмотрим конкретные решения:

Акунетикс — система с графическим интерфейсом и онлайн версией. Очень дорогая. Для использования даже пробной версии необходимо выполнить неимоверное количество действий, начиная от регистрации, включающей получение пин-кода по телефону, и заканчивая подтверждением прав на исследование конкретного веб-сайта. Простому пользователю в этом разобраться крайне сложно.

Arachni, SkipFish и W3AF — не имеют графического интерфейса, что сразу делает их сложными в использовании для пользователей с начальным уровнем подготовки.

OWASP ZAP — бесплатное средство с открытым исходным кодом. Не имеет онлайн-версии. Чтобы начать аудит веб-сайта от пользователя требуется скачать и установить программу а также указать объект аудита.

Burp Suite — платный, сложный, нет онлайн-версии.

Вега — требует установки и выбора параметров сканирования. Нет онлайн версии.

Проанализировав рынок существующих решений я задался целью:

Цель
Целью данной работы является разработка автоматизированной системы аудита веб-приложений на наличие уязвимостей, которая проста в использовании для пользователей с начальным уровнем подготовки, имеет онлайн-версию, открытый исходный код и требует минимального количества действий от пользователя.

Схема работы системы
Схема работы системы изображена на idef0 диаграмме. Функция, которую выполняет система — аудит веб-приложения. На вход система получает указанный пользователем объект аудита. На выходе — отчет о проведенном аудите и рекомендации по повышению уровня безопасности. Механизмы — движок Node JS и инструменты операционной системы Linux. Управлением служат рекомендации OWASP.
Подробнее о рекомендациях OWASP: Open Web Application Security Project — это открытый проект обеспечения безопасности веб-приложений. Сообщество OWASP включает в себя корпорации, образовательные организации и частных лиц. Сообщество работает над созданием статей, учебных пособий, документации, инструментов и технологий, находящихся в свободном доступе.
OWASP Топ-10 является признанной методологией оценки уязвимостей веб-приложений во всем мире.

Функции разрабатываемой системы
Сбор данных о веб-приложении

Проверка на соответствие рекомендациям OWASP
Анализ на наличие уязвимостей
Составление отчета
Формирование рекомендаций по повышению уровня безопасности
Подробнее:

Сбор данных
Читать по слайду

Проверка на соответствие рекомендациям OWASP
Система проверяет веб-ресурс на соответствие следующим правилам OWASP:
Наличие http-заголовков, повышающих уровень безопасности сайта, факт установки времени жизни сессии, отсутствие факта использования внешних объектов, отсутствие факта кэширования страниц, содержащих информацию конфиденциального характера.

Анализ веб-приложения на наличие уязвимостей
Для анализа на наличие уязвимостей используются как уже готовые, и зарекомендовавшие себя средства, такие, как SQLmap, gobuster, wpscan (все с открытым кодом), так и методы, разработанные специально для данного проекта и реализованные с использованием nodeJS и оболочки shell.

Клиентская часть
Система имеет удобный графический интерфейс, представляющий собой поле для ввода урл-адреса объекта аудита и набора кнопок, характеризующих функции, доступные для выполнения. Работают они независимо друг от друга в целях экономии времени ожидания результата. После выбора той или иной функции, пользователь получает **отчет** о проведенном аудите и **рекомендации** по повышению уровня безопасности.
Кнопки соответственно — сбор информации о ресурсе, проверка на предмет следования рекомендациям OWASP, проверка наличия XSS уязвимости, проверка наличия SQL-инъекций, проверка возможности прямого доступа к объектам в обход авторизации, проверка использования компонентов с известными уязвимостями, проверка наличия небезопасных перенаправлений.

Серверная часть
На слайде представлена структура системы. На программной платформе nodeJS реализован сервер, принимающий запросы клиента, роутер, вызывающий тот или иной обработчик, и скрипт, содержащий обработчики запросов, который содержит в себе большую часть логики. Также используются команды оболочки linux. Процесс работы системы логируется, результат можно увидеть в консоли или лог-файле.

Анализ наличия XSS-уязвимости
Пользователь указывает урл сайта и нажимает на кнопку XCC - идет запрос на сервер с указанием обработчика и урлом в значении пост-параметра. Сервер передает полученный запрос в роутер, который вызывает нужный обработчик.
Обработчик делает запрос на указанный сайт и получает ответ в формате JSON (содержит заголовки, мета данные и тело ответа). Тело ответа (которое нас и интересует) в формате plain/text. Для дальнейшей работы с ним используется npm-модуль jsdom для преобразования text в DOM (Document Object Model), чтобы дать возможность javascript обращаться непосредственно к объектам DOM-модели. Далее скрипт обращается к объекту forms и создает json объект, содержащий в себе адреса обработчиков форм, методы отправки форм и их параметры. Зная эти параметры, скрипт делает запрос к обработчику формы с параметрами, имеющими в значении обычную строку. Ответ преобразуется тем же образом,

что и ранее и количество скриптов, содержащихся в странице ответа записывается в переменную. После этого скрипт начинает делать запросы, в значении параметров имеющие векторы xcc-атак. Ответы преобразуются в дом-модель и подсчитывается количество скриптов, содержащихся в теле ответа. Если это число больше числа, записанного в переменную, то атака прошла успешно и вредоносный запрос встроил скрипт в ДОМ-модель страницы ответа.

Демонстрация работы сервиса
На слайде можно увидеть результат анализа одного из поддоменов сайта ИТМО, уязвимого к XSS. На экран выводится информация об обработчиках форм, методах их отправки и параметрах. Далее следует отчет о наличии или отсутствии уязвимости. Затем идут ссылки со встроенными векторами атак, успешность которых можно проверить вручную. В конце идут рекомендации OWASP по устранению уязвимости. В окне справа — результат перехода по одной из ссылок - в страницу встроен скрипт, выводящий на экран сообщение ”XSS”. Владельцу сайта было сообщено о наличии уязвимости.

Слайд **Выводы**

- Разработана система, проводящая аудит веб-приложений на соответствие рекомендациям OWASP и наличие уязвимостей.
- Система проста в использовании даже для неопытных пользователей.
- Есть онлайн-версия (с обрезанным функционалом ввиду бесплатного тарифа на хостинг nodeJS-приложений)
- Открытый исходный код

Возможные ответы на возможные вопросы:

Аудит информационной безопасности — независимая оценка текущего состояния системы информационной безопасности, устанавливающая уровень ее соответствия определенным критериям, и предоставление результатов в виде рекомендаций.
Заголовки

- Заголовок **HSTS (HTTP Strict Transport Security)** гарантирует, что весь обмен данными из браузера осуществляется по протоколу HTTPS
- Заголовок **X-Frame-Options** позволяет снизить уязвимость вашего сайта для кликджекинг-атак. Этот заголовок служит инструкцией для браузера не загружать вашу страницу в frame/iframe.
- **X-Content-Type-Options** - Можно предотвратить атаки с использованием подмены MIME типов, добавив этот заголовок ответа HTTP. Содержит инструкции по определению типа файла и не допускает сниффинг контента.
- Чтобы предотвратить XSS-атаки, кликджекинг, внедрение кода, можно добавить заголовок ответа **Content Security Policy (CSP)**

Unsafe redirects — Анализ наличия мета-тега http-equiv, анализ get и post параметров.