# ЗАЩИТА ВЕБ-СЕРВИСОВ НА ОСНОВЕ ТЕХНОЛОГИИ WS-SECURITY

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Михайлов А.С.*

*Саломатин С.Б. – к.т.н., доцент*

Безопасность веб-приложений находится в первой десятке трендов и угроз информационной безопасности уже свыше 10 лет. Тем не менее специализированных средств защиты веб-приложений довольно мало, по большей части эту задачу возлагают (или надеются что она будет решена) на разработчиков. Это и использование различных фреймворков, средств санации, очистки данных, нормализации и многого другого. Со временем усложнялись веб-приложения, серверная инфраструктура и взаимодействие, код становился все более объемным и громоздким — это намного увеличило т.н. "поверхность атаки".

**Базы данных**

Храните данные идентификации пользователей и конфиденциальные данные (токены, адреса электронной почты, платежные реквизиты) в зашифрованном виде.

Если база данных поддерживает шифрование хранящихся данных (например, AWS Aurora), подключите его для защиты данных на диске. Убедитесь, что все резервные копии также хранятся в зашифрованном виде.

Используйте наименьший уровень привилегий для доступа к учетным записям пользователей в базе данных. Не используйте учётную запись root базы данных.

Предотвращайте SQL-инъекции, используя исключительно подготовленные SQL-запросы. Например: если вы используете NPM, не используйте npm-mysql, используйте npm-mysql2, который поддерживает подготовленные выражения.

**Разработка**

Убедитесь, что все компоненты приложения проверены на наличие уязвимостей для каждой версии, переданной в продакшн. Сюда входят O/S, библиотеки и пакеты. Проверка должна быть автоматизирована в процессе CI-CD (CI — continuous integration — непрерывная интеграция, CD — continuous delivery — постоянная поставка, прим. перев.).

С одинаковой бдительностью относитесь как к безопасности среды разработки, так и к безопасности сервера. Создавайте программное обеспечение в защищенной изолированной среде разработки.

**Идентификация**

Убедитесь, что все пароли хэшируются с использованием соответствующей криптографической функции, например, bcrypt. Никогда не пишите собственную функцию хеширования и корректно инициализируйте используемую криптографическую библиотеку случайными данными.

Реализуйте простые, но адекватные правила паролей, которые побуждают пользователей вводить длинные уникальные пароли.

Во всех сервисах используйте многофакторную аутентификацию для входа в систему.

**Защита от DDoS-атак**

Убедитесь, что DDoS-атаки на ваши API не навредят сайту. Как минимум, защитите «узкие» места API, такие как процедуры генерации логина и токена.

Обеспечьте разумные ограничения по размеру и структуре предоставляемых пользователем данных и запросов.

Смягчайте DDoS-атаки с помощью глобального сервиса с кэширующим прокси, например, CloudFlare. Он включается, когда вы находитесь под DDOS-атакой, а в обычном режиме функционирует как DNS lookup.

**Веб-трафик**

Используйте TLS для всего сайта, а не только для форм входа и ответов. Никогда не используйте TLS только для формы входа.

Куки должны быть «безопасными» (secure) и httpOnly, а область видимости должна определятся атрибутами path и domain.

**API**

Убедитесь, что в API нет общедоступных ресурсов.

Убедитесь, что при использовании ваших API пользователи полностью идентифицированы и авторизованы.

**Облачная конфигурация**

Убедитесь, что все сервисы имеют минимальное количество открытых портов. В то время как принцип «безопасность через неясность» (security through obscurity) не обеспечивает полной защиты, использование нестандартных портов немного усложнит жизнь злоумышленникам.

Размещайте базу бекэнда на частных VPC, которые не видны в публичной сети. Будьте очень осторожны при настройке групп безопасности AWS и пиринговых VPC — можно непреднамеренно сделать службы публичными.

**Инфраструктура**

Убедитесь, что апгрейды делаются без простоя, а ПО обновляется автоматически.

Не используйте SSH в службах, кроме разве что разовой диагностики. Регулярное использование SSH, как правило, означает, что вы не автоматизировали все как надо.

**Эксплуатация**

Выключите неиспользуемые службы и серверы. Самый безопасный сервер — это выключенный сервер.

**Тестирование**

Проводите аудит и проекта, и готовой реализации.

Список использованных источников:

1. Michael O’Brien https://simplesecurity.sensedeep.com/web-developer-security-checklist-f2e4f43c9c56