Рассмотрим основные уязвимости, которым подвержены веб-приложения, какие действия приводят к их появлению, а также способы предотвращения их возникновения.

1. **Нарушение контроля доступа**

Это набор уязвимостей, при которых система плохо контролирует уровни доступа к информации или к своей функциональности. Из-за этого злоумышленники могут пользоваться функциями, к которым не должны иметь доступа. Такая уязвимость также может привести к разглашению или утрате конфиденциальной информации, взлому учетных записей пользователей и нарушению целостности данных.

Как предотвратить:

* Проектируйте контроль доступа на основе принципа наименьших привилегий. Пользователи должны иметь только те права, которые необходимы для выполнения их задач.
* Проводите аутентификацию и авторизацию на всех уровнях приложения — и на серверной, и на клиентской стороне.
* Регулярно проводите тестирование и аудит контроля доступа.

Чего делать не стоит:

* Полагаться только на скрытие ссылок или кнопок в пользовательском интерфейсе для ограничения доступа. Это не предотвратит доступ к закрытой функциональности по прямым запросам.
* Доверять пользовательским входным данным при авторизации. Всегда следует проводить проверку на сервере.
* Оставлять прежней политику контроля доступа при изменении требований и бизнес-логики приложения.

1. **Недостатки криптографии**

Это уязвимости, связанные с неправильной настройкой и использованием криптографических методов для защиты данных. К ним относят недостаточную длину ключей, ненадежные условия их хранения, использование устаревших алгоритмов и другие ошибки в криптографической реализации. Слабая криптография — как картонный сейф: делает данные уязвимыми для атакующих, но создает иллюзию защищенности.

Как предотвратить:

* Обновляйте и пересматривайте криптографические методы и ключи с учетом последних рекомендаций и стандартов.
* Храните криптографические ключи в надежном месте. Избегайте их хранения вместе с кодом приложения или в открытом виде.

Чего делать не стоит:

* Использовать устаревшие или слабые алгоритмы шифрования.
* Реализовывать собственные криптографические методы, если вы не являетесь экспертом в этой области.
* Хранить криптографические ключи в открытом виде или внутри кода приложения.

1. **Инъекции**

Это пользовательский ввод с вредоносным кодом. Чаще всего инъекции включают SQL-запросы и команды на языке оболочки операционной системы. Они позволяют злоумышленникам внедрять свой вредоносный код на сервер и выполнять его. Результат — потеря данных, кража данных или повреждение системы.

Как предотвратить:

* Используйте параметризованные запросы или ORM (object-relational mapping) для работы с базой данных.
* Валидируйте и фильтруйте входные данные. Принимайте только допустимые символы и структуры данных.
* Применяйте принцип наименьших привилегий: ограничивайте права доступа к базе данных необходимыми.
* Используйте LIMIT и другие элементы управления SQL в запросах для предотвращения массового раскрытия записей в случае SQL-инъекции.

Чего делать не стоит:

* Конкатенировать и вставлять непроверенные данные пользователя напрямую в SQL-запросы, команды операционной системы или другие исполняемые на сервере контексты.
* Надеяться на то, что фильтрация одного типа данных предотвратит инъекции. Злоумышленники могут использовать разные методы атак.
* Хранить конфиденциальные данные в чистом тексте без шифрования в базе данных.

1. **Небезопасный дизайн архитектуры приложения**

Уязвимости этой категории возникают потому, что сама логика работы приложения может позволять использовать существующие функции для взлома.

Как предотвратить:

* Продумывайте аспекты безопасности на ранних этапах проектирования приложения.
* Оценивайте потенциальные угрозы и риски на этапе проектирования и разрабатывайте меры их предотвращения.
* Обязательно моделируйте угрозы для критической аутентификации, контроля доступа, бизнес-логики и ключевых потоков в приложении.
* Ограничивайте количество ресурсов на сервере, которое выделяется на одного пользователя и на одну сессию.

Чего делать не стоит:

* Полагаться только на обеспечение безопасности на уровне кода. Безопасный дизайн важен для создания надежной системы в целом.
* Разрабатывать систему, не учитывая возможные атаки на нее.

1. **Небезопасная конфигурация**

Эта уязвимость представляет собой ситуацию, когда настройки приложения, сервера, базы данных или других компонентов системы не являются безопасными. К этой группе уязвимостей относят ненадежные или отсутствующие настройки аутентификации, авторизации и доступа.

Как предотвратить:

* Проводите безопасную настройку всех компонентов приложения и инфраструктуры, следуя рекомендациям и стандартам безопасности.
* Продумайте и поддерживайте политику настройки доступов.
* Отключайте или удаляйте ненужные функции и службы на сервере, чтобы сократить возможный спектр атак.
* Реализуйте автоматизированный процесс проверки эффективности конфигураций и настроек во всех средах.
* Регулярно проверяйте настройки на наличие уязвимостей.

Чего делать не стоит:

* Оставлять дефолтные пароли, настройки или ключи. Обязательно меняйте их на уникальные и сложные.
* Оставлять включенными даже те функции и службы, что кажутся избыточными.
* Полагаться только на документацию по установке. Проверяйте и дорабатывайте настройки с учетом текущих требований безопасности.

1. **Использование уязвимых или устаревших компонентов**

К этому типу уязвимостей относят случаи, когда веб-приложение использует сторонние фреймворки, библиотеки, плагины или другие компоненты, которые имеют выявленные дефекты безопасности. У злоумышленников даже есть автоматизированные инструменты, которые помогают находить непропатченные или неправильно сконфигурированные системы.

Как предотвратить:

* Регулярно обновляйте используемые компоненты. Следите за выпуском обновлений и исправлений, касающихся безопасности компонентов.
* Удаляйте неиспользуемые зависимости, ненужные функции, компоненты и файлы.
* Используйте источники, которые предоставляют информацию о безопасности компонентов: OWASP Dependency-Check, Retire.js и другие.

Чего делать не стоит:

* Использовать устаревшие компоненты без обновления.
* Игнорировать предупреждения о безопасности, которые касаются используемых компонентов.

1. **Ошибки идентификации и аутентификации**

К этой категории уязвимостей относят слабые пароли, недостаточную проверку подлинности, неэффективные системы учета сеансов, незащищенные способы восстановления паролей (например, подходы на основе знаний, когда человек должен ответить на секретный вопрос), отсутствие многофакторной авторизации, раскрытие идентификатора сессии в URL.

Как предотвратить:

* Используйте сильные механизмы аутентификации, такие как двухфакторная аутентификация.
* Требуйте от пользователей создавать пароли с высокой устойчивостью к взлому, включающие в себя не только буквы, но и другие символы.
* Не раскрывайте идентификаторы сессии в URL-адресе.
* Блокируйте аккаунты после определенного количества неудачных попыток входа.

Чего делать не стоит:

* Разрешать пользователям использовать слабые пароли или пароли по умолчанию.
* Хранить пароли пользователей в открытом виде в базе данных. Храните хеши паролей с солью.

1. **Нарушения целостности программного обеспечения и данных**

К этой группе уязвимостей относят случаи, когда после обновления приложение или оборудование начинает работать неправильно. Например, роутер после обновления прошивки не требует пароля для подключения или сбрасывает его до заводского.

Как предотвратить:

* Используйте проверку целостности данных, используя хеши и цифровые подписи для обнаружения несанкционированных изменений.
* Ограничивайте доступ и возможность изменения данных для неавторизованных пользователей.
* Внедряйте подробные журналы и мониторинг подозрительной активности, чтобы сохранять информацию о совершаемых пользователями действиях.

Чего делать не стоит:

* Хранить критически важные данные в открытом виде или без требования авторизации.
* Выдавать всем пользователям полные права на изменение данных. Всегда используйте принцип наименьших привилегий, выдавая только действительно необходимые права.
* Игнорировать предупреждения системы мониторинга о подозрительной активности. Реагируйте на них своевременно.

1. **Ошибки логирования и мониторинга безопасности**

Это уязвимости, при которых система неправильно регистрирует аномальные события, касающиеся безопасности. К ним также относят отсутствие или неправильную настройку механизмов логирования и отсутствие уведомлений о подозрительных событиях.

Если в системе отсутствует мониторинг безопасности, то атаки злоумышленников могут остаться незамеченными. Это снижает вероятность быстрой реакции на возникающие инциденты, обнаружение угроз и определение их источников.

Как предотвратить:

* Внедрите механизмы логирования для регистрации важных событий, таких как попытки аутентификации, изменения в конфигурации и доступе к чувствительным данным.
* Установите систему мониторинга, которая анализирует логи на наличие подозрительной активности и уведомляет вас об инцидентах.
* Определите четкие процедуры реагирования на инциденты и оповещения и обязательно расскажите о них всей команде.

Чего делать не стоит:

* Оставлять логирование без внимания. Регулярно анализируйте логи для выявления аномальных событий.
* Обходиться без мониторинга. Обязательно убедитесь, что система мониторинга активна и правильно настроена.
* Использовать только автоматические уведомления о состоянии системы. Регулярно вручную проверяйте состояние системы и логов.

1. **Подделка запросов на стороне сервера**

Это тип уязвимости, при котором злоумышленник заставляет сервер отправлять запросы к внутренним ресурсам или внешним сайтам. Наличие данной уязвимости в веб-приложениях часто используется злоумышленниками для обнаружения и атаки внутренних ресурсов, к которым они обычно не имеют доступа извне.

Как предотвратить:

* Ограничивайте или фильтруйте пользовательский ввод, который используется для формирования запросов.
* Используйте белый список (whitelist) разрешенных адресов, на которые сервер может отправлять запросы.
* Ограничьте и контролируйте доступ сервера к внутренним ресурсам.

Чего делать не стоит:

* Доверять непроверенным или неконтролируемым URL-адресам, переданным пользователем.
* Открывать доступ сервера к внутренним ресурсам без проверки.
* Использовать пользовательский ввод напрямую для формирования запросов на стороне сервера.

Не стоит забывать, что способа абсолютной защиты от хакерских атак пока не существует. Поэтому очень важно регулярно отслеживать работу веб-приложения и своевременно предотвращать возникновение ошибок.

**СПИСОК ИСТОЧНИКОВ:**

OWASP Top 10: самые распространенные уязвимости веб-приложений // Skillbox Media : [сайт]. — URL: https://skillbox.ru/media/code/owasp-top-10-samye-rasprostranyennye-uyazvimosti-vebprilozheniy/ (дата обращения: 03.12.2023).

OWASP Top Ten // OWASP Foundation : [сайт]. — URL: https://owasp.org/www-project-top-ten/ (дата обращения: 03.12.2023).