ПРОБЛЕМЫ БЕЗОПАСНОСТИ СЕРВЕРНОЙ ЧАСТИ ANDROID ПРИЛОЖЕНИЙ И ПУТИ ИХ РЕШЕНИЯ

В. И. Синица

Белорусский государственный университет, г. Минск;
Vladsinitsa23@gmail.com
науч. рук. – Е. А. Чудовская, канд. физ.-мат. наук, доц.

В последнее время значительно возросло число успешных хакерских атак на пользовательские приложения. В результате таких атак похищаются не только огромные денежные средства, но и пользовательские данные. Это связано с тем, что современные приложения имеют массу уязвимостей разной степени риска: уязвимости высокого риска были обнаружены в 38 % мобильных приложений для iOS и в 43 % приложений для Android [1]. Большинство проблем с безопасностью встречается на обеих платформах. Небезопасное хранение данных – самая распространенная проблема, встречающаяся в 76 % мобильных приложений. Под угрозой находятся пароли, финансовая информация, личные данные и переписка. Большинство случаев вызвано слабостью механизмов безопасности (74 % и 57 % для приложений iOS и Android соответственно и 42 % для серверных компонентов). В данной работе будет описан ряд основных уязвимостей безопасности клиент-серверных приложений и предложены варианты их предотвращения.

Ключевые слова: уязвимости Android приложений, уязвимости сервера, JWT, авторизация, аутентификация, угрозы безопасности клиент-серверных приложений, SQL-инъекция, XSS.

ВВЕДЕНИЕ

Спрос на продукцию мобильных приложений растет с каждым годом. Во всем мире число загрузок в App Store и Google Play по итогам 2020 года превысило 130 миллиардов скачиваний. При этом сохраняется тенденция увеличения хакерских атак с целью хищения пользовательских данных: информация о банковских картах, пользовательской конфиденциальной информации и паролей [1].

И несмотря на это, многие разработчики уделяют пристальное внимание дизайну программного обеспечения для предоставления бесперебойной и удобной работы, при этом зачастую забывая о качественной реализации защиты. А пользователи с удовольствием устанавливают мобильные приложения и предоставляют личную информацию, но редко задумываются о последствиях для безопасности.

Поэтому на плечи разработчиков в первую очередь ложится ответственность за данные, которые предоставляет клиент, качественную их транзакцию и хранение. Рассмотрим, какие угрозы особенно опасны в последнее время и каким образом их можно избежать.

РЕАЛИЗАЦИЯ ЗАЩИТЫ ОТ ИНЪЕКЦИЙ

Недостатки реализации защиты от инъекций возникают из-за классической неспособности отфильтровать ненадежный ввод. Это может произойти, когда передаются нефильтрованные данные на SQL-сервер (SQL-инъекция) [1], в браузер (XSS) [1], на LDAP-сервер (LDAP-инъекция) или в другое место. Проблема состоит в том, что злоумышленник может вводить определенные команды, которые приводят к потере данных, захвату браузеров клиентов и ко множеству других проблем, связанных с нарушениями конфиденциальности, целостности и доступности данных.

Защита от инъекций – это вопрос правильной фильтрации входных данных и анализа того, можно ли доверять входным данным. Но нужно понимать, что организация качественной проверки и фильтрации данных приводит как к временным, так и функциональным затратам. А доверие к входным данным само по себе является достаточно опасным решением.

Межсайтовый скриптинг (XSS) – это довольно распространенное нарушение. Злоумышленник передает веб-приложению теги JavaScript при вводе. При загрузке страницы скрипт запускается и, например, может использоваться для отправки пользовательских файлов злоумышленнику.

Один из часто используемых для решения методов – использование регулярных выражений для поиска и удаления HTML-тегов или иных символов.

ОТСУТСТВИЕ КОНТРОЛЯ ДОСТУПА НА СЕРВЕРЕ

Отсутствие контроля доступа на сервере – это проблема авторизации. В этом случае при вызове функции на сервере не производится надлежащая авторизация. Часто разработчики полагаются на тот факт, что пользовательский интерфейс генерируется на стороне сервера и функции, которые не предоставляются сервером, якобы не могут быть доступны для клиента. На самом деле это не так и злоумышленник всегда может подделать запросы к функциям и методам, которые не предполагают их использование со стороны клиента. Поэтому на стороне сервера всегда должна выполняться качественная авторизация и производиться разделение доступа ко всем функциям приложения.

НЕКАЧЕСТВЕННАЯ АУТЕНТИФИКАЦИЯ

Построение качественной аутентификации является первоочередной задачей каждого разработчика при создании безопасного приложения. Существует множество проблем, которые могут возникнуть при слабой

или вовсе неработающей аутентификации, перечислять их можно до бесконечности. Существует ряд типичных ошибок у разработчиков:

- URL-адрес может содержать идентификатор сеанса и передать его кому-то другому в заголовке.
- Незашифрованные пароли как при хранении, так и при передаче.
- Идентификаторы сеанса могут быть предсказуемыми, поэтому получение доступа тривиально.
- Возможна фиксация сеанса, его захват, неправильное выполнение таймаутов и многое другое.

Решение перечисленных проблем. Разработки качественной аутентификации и авторизации с помощью технологии JWT

Решения перечисленных проблем во многом уже реализованы в популярных фреймворках. Взять к примеру защиту от инъекций: поскольку фильтрацию довольно сложно сделать правильно (например, с криптографией), лучшим решением будет как раз-таки полагаться на функции фильтрации фреймворков: они доказали свою эффективность и тщательно изучаются.

Современные же решения для аутентификации и авторизации внедрили концепцию токенов в свои протоколы. Токены – это специально созданные фрагменты данных, которые несут в себе достаточно информации, чтобы либо авторизовать пользователя для выполнения действия, либо позволить клиенту получить дополнительную информацию о процессе. Доступна ли эта информация для чтения или анализа клиенту определяется реализацией. Одним из наиболее распространенных типов токенов на данный момент является JWT [2].

JWT (JSON Web Token) определяет способ, в котором может быть представлена некоторая общая информация, относящаяся к процессу аутентификации или авторизации. JWT становятся действительно полезными в сочетании с другими спецификациями, такими как JSON Web Signature (JWS) и JSON Web Encryption (JWE). Вместе эти спецификации предоставляют не только всю информацию, обычно необходимую для токена авторизации, но также средства для проверки содержимого токена, чтобы его нельзя было подделать (JWS), и способ шифрования информации, чтобы она оставалась непрозрачной клиенту.

ТИПЫ ТОКЕНОВ

Существует два типа токенов: Access token (токен доступа) и Refresh token (токен обновления). Зачастую в приложениях используют только

токены доступа, из-за более простой реализации и использования. Но хорошим тоном считается реализовывать оба.

Токены доступа несут в себе необходимую информацию для прямого доступа к ресурсу. Другими словами, когда клиент передает такой токен серверу, управляющему ресурсом, этот сервер может прочитать информацию, содержащуюся в токене, и сам решить, авторизован пользователь или нет (никаких проверок на сервере авторизации не требуется). Это одна из причин, по которой токены должны быть подписаны (например, с использованием JWS). Токены доступа обычно имеют короткий срок действия [3].

Токены обновления несут информацию, необходимую для получения нового токена доступа. Другими словами, всякий раз, когда требуется Access token для доступа к определенному ресурсу, клиент может использовать токен обновления для его получения, который выдается сервером аутентификации. Общие варианты использования включают получение новых токенов доступа после истечения срока действия старых или получение доступа к новому ресурсу в первый раз. Токены обновления также могут истекать, но они довольно долговечны. К Refresh token обычно предъявляются строгие требования к хранению, чтобы гарантировать отсутствие утечки [3].

ЗАКЛЮЧЕНИЕ

В заключение хотелось бы отметить важность построения мощной и качественной безопасности в приложениях. В этом во многом помогают технологии, реализованные в фреймворках. Особое внимание стоит уделять аутентификации и авторизации. Современным и лучшим решением для этого является использование сразу нескольких токенов: доступа и обновления. Такой способ повышает безопасность и позволяет сократить время ожидания от сервера.

Библиографические ссылки

1. Уязвимости и угрозы в мобильных приложениях [Электронный ресурс]. URL: https://www.ptsecurity.com/ww-en/analytics/mobile-application security-threats-and-vulnerabilities-2019 (дата обращения: 10.05.2021).
2. Введение в веб-токены JSON [Электронный ресурс]. URL: https://jwt.io/introduction (дата обращения: 10.05.2021).
3. Типы токенов и их использование [Электронный ресурс]. URL: https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them (дата обращения: 10.05.2021).