МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Кафедра *технологий обработки и защиты информации*

Тема реферата

«Уязвимости мобильных приложений»

10.03.01 Информационная безопасность Безопасность компьютерных систем

Выполнила _____ Ю.П. Горбунова, 4 курс, д/о

Руководитель _____ М.А.Дрюченко, к.т.н., доцент

2023

Содержание

Введение

Разработка мобильных приложений в тренде, ее технологии непрерывно развиваются. Большинство современных решений имеют клиент-серверную архитектуру. Клиент работает под управлением мобильной операционной системы; чаще всего это Android или iOS. Клиентская часть загружается на устройство из так называемого магазина приложений — специализированной площадки, где разработчики размещают свои системы. С точки зрения обычного пользователя, установленная на смартфон программа — это и есть мобильное приложение, ведь именно с ней он взаимодействует напрямую: совершает покупки, оплачивает счета, просматривает почту.

Мобильные устройства прочно вошли в нашу жизнь: мессенджеры, банкинг, бизнес-приложения, личные кабинеты сотовых операторов — при современном ритме жизни мы используем эти приложения практически ежедневно. Общее число пользователей мобильных банковских приложений приближается к двум миллиардам, что составляет порядка 40% всего взрослого населения. Мобильным банком пользуется каждый третий (34%) россиянин старше 18 лет.

Разработчики уделяют большое внимание дизайну программных продуктов для наших гаджетов, стараясь сделать их максимально удобными. Люди охотно устанавливают мобильные приложения и регистрируются в них, но мало кто из рядовых пользователей задумывается о безопасности данных.

В данном реферате будут рассмотрены уязвимости мобильных приложений, а также рассмотрены некоторые методы решения этих проблем.

Реферат составлен на основе данных полученных из Open Web Application Security Project (OWASP).

OWASP – открытый проект по обеспечению безопасности приложений, все материалы которого доступны бесплатно на веб-сайте некоммерческой организации OWASP Foundation. Поставляемые материалы включают документацию, мероприятия, форумы, проекты, инструменты и видео, такие как OWASP Top 10, веб-протоколы OWASP CLASP и OWASP ZAP, а также сканер веб-приложений с открытым исходным кодом.

OWASP важен для организаций, потому что его рекомендации высоко ценятся

проверяющими безопасность корпоративных систем аудиторами. Рекомендации проекта по защите приложений необходимо включить в жизненный цикл разработки программного обеспечения и использовать для формирования политик безопасности.

OWASP Mobile TOP 10 — одна из основных методологий тестирования приложений на уязвимости.

1. **M1 - Обход архитектурных ограничений**

Эта категория охватывает неправильное использование функции платформы или отказ от использования средств управления безопасностью платформы. Любой открытый вызов API может служить здесь вектором атаки.

Существует несколько способов, которыми мобильные приложения могут испытывать этот риск.

1. Нарушение опубликованных руководящих принципов. Все платформы имеют рекомендации по разработке для обеспечения безопасности. Если приложение противоречит рекомендациям производителя, оно будет подвержено этому риску. Например, существуют рекомендации по использованию связки ключей iOS или по защите экспортируемых сервисов на Android. Приложения, которые не соответствуют этим рекомендациям, будут подвергаться этому риску.
2. Нарушение конвенции или общепринятой практики. Не все рекомендации кодифицированы в руководстве производителя. В некоторых случаях де-факто существуют лучшие практики, которые распространены в мобильных приложениях.
3. Непреднамеренное неправильное использование. Некоторые приложения намерены делать правильные вещи, но на самом деле получают некоторую часть реализации неправильно. Это может быть простая ошибка, например, установка неправильного флага на вызов API, или это может быть неправильное понимание того, как работает защита.

Сбои в моделях разрешений платформы подпадают под эту категорию. Например, если приложение запрашивает слишком много разрешений или неправильные разрешения, это лучше всего классифицировать здесь.

Предотвращение:

Безопасные методы кодирования и настройки должны использоваться на стороне сервера мобильного приложения.

2. **M2 - Небезопасное хранение данных**

Небезопасное хранение данных, а также непреднамеренные утечки данных также попадают в первую десятку OWASP Mobile. Инструменты тестирования на проникновение мобильных приложений помогают выявить такие жалобы.

В программе часть отладочной информации может отображаться в логах системы. В

этом случае сторонняя программа, которая имеет права READ_LOGS (например, logcat или pidcat), может получить доступ к чувствительной информации, тем самым нарушая ее конфиденциальность. Также могут быть затронуты файлы манифестов и журналов, хранилище файлов cookie или облачная синхронизация.

Небезопасные уязвимости хранилища данных возникают, когда команды разработчиков предполагают, что пользователи или вредоносные программы не будут иметь доступа к файловой системе мобильного устройства и последующей конфиденциальной информации в хранилищах данных на устройстве. Файловые системы легко доступны. Организации должны ожидать, что злоумышленник или вредоносная программа проверят конфиденциальные хранилища данных. Следует избегать использования плохих библиотек шифрования. Рутирование или джейлбрейк мобильного устройства обходит любые средства защиты шифрования. Когда данные не защищены должным образом, специализированные инструменты — это все, что нужно для просмотра данных приложения.

Это может привести к потере данных, в лучшем случае для одного пользователя и в худшем случае для многих пользователей. Это также может привести к следующим техническим последствиям: извлечение конфиденциальной информации приложения с помощью мобильного вредоносного ПО, модифицированных приложений или криминалистических инструментов. Характер воздействия на бизнес в значительной степени зависит от характера украденной информации. Небезопасные данные могут привести к следующим последствиям для бизнеса:

- Кража личных данных;
- Нарушение конфиденциальности;
- Мошенничество;
- Репутационный ущерб;
- Нарушение внешней политики (PCI); или
- Материальные потери.

Рекомендации по закрытию уязвимости:

Важно моделировать угрозы для мобильного приложения, ОС, платформ и платформ, чтобы понять информационные активы, обрабатываемые приложением, и то, как API обрабатывают эти ресурсы. Важно посмотреть, как они обрабатывают

следующие типы функций:

- кэширование URL (как запроса, так и ответа);
- Кэширование нажатия клавиши клавиатуры;
- Кэширование буфера копирования/вставки;
- Фоновая подготовка приложений;
- Промежуточные данные
- Хранение данных HTML5;
- Объекты cookie браузера;
- Аналитические данные, отправленные 3-м лицам.

3. M3 - Небезопасная передача данных

При разработке мобильного приложения обмен данными обычно осуществляется клиент-серверным способом. Когда передают свои данные, все должно проходить через сеть оператора мобильного устройства и Интернет. Злоумышленники могут использовать уязвимости для перехвата конфиденциальных данных во время их перемещения по проводам. Существуют следующие угрозы:

- Преступник, который совместно использует вашу локальную сеть (скомпрометированный или контролируемый Wi-Fi);
- Операторы или сетевые устройства (маршрутизаторы, вышки сотовой связи, прокси и т.д.);
- Вредоносное ПО на вашем мобильном устройстве.

Здесь помогут основные инструменты тестирования на проникновение мобильных приложений. Они помогут обнаружить неисправную связь между приложениями и серверами или мобильными устройствами. Самой большой проблемой является передача конфиденциальных данных с одного устройства на другое. Это может быть шифрование, пароли, данные учетной записи или личная информация о пользователе. Если необходимые меры безопасности отсутствуют на этом этапе, хакерам легко получить доступ к данным.

Этот недостаток раскрывает данные отдельного пользователя и может привести к краже учетной записи. Если злоумышленник перехватывает учетную запись администратора, весь сайт может быть раскрыт. Плохая настройка SSL также может способствовать фишинговым атакам и атакам MITM.

Этот риск включает в себя все коммуникационные технологии, которые может использовать мобильное устройство: TCP / IP, WiFi, Bluetooth / Bluetooth-LE, NFC, аудио, инфракрасное, GSM, 3G, SMS.

Рекомендации:

- Использование надежных стандартных наборов шифров с соответствующей длиной ключей.
- Использование сертификатов, подписанных доверенным поставщиком ЦС.
- Никогда не разрешайте самозаверяющие сертификаты и рассмотрите возможность закрепления сертификатов для приложений, заботящихся о безопасности.
- Всегда требуется проверка цепочки SSL.
- Устанавливайте безопасное соединение только после проверки подлинности сервера конечной точки с помощью доверенных сертификатов в цепочке ключей.
- Предупреждайте пользователей через пользовательский интерфейс, если мобильное приложение обнаруживает недопустимый сертификат.
- Не отправляйте конфиденциальные данные по альтернативным каналам (например, SMS, MMS или уведомления).
- Если возможно, примените отдельный уровень шифрования к любым конфиденциальным данным, прежде чем они будут переданы в канал SSL. В случае обнаружения будущих уязвимостей в реализации SSL зашифрованные данные обеспечат вторичную защиту от нарушения конфиденциальности.

4. M4 - Небезопасная аутентификация

Как только злоумышленник понимает, насколько уязвима схема аутентификации, он подделывает или обходит аутентификацию, отправляя запросы на обслуживание на внутренний сервер мобильного приложения и обходя любое прямое взаимодействие с мобильным приложением. Этот процесс отправки обычно выполняется через мобильное вредоносное ПО на устройстве или в ботнетах, принадлежащих злоумышленнику.

Существует множество различных способов, которыми мобильное приложение может пострадать от небезопасной аутентификации:

- Если мобильное приложение может анонимно выполнить запрос серверной службы API без предоставления маркера доступа, это приложение страдает от небезопасной проверки подлинности;
- Если мобильное приложение хранит какие-либо пароли или общие секреты локально на устройстве, оно, скорее всего, страдает от небезопасной аутентификации;
- Если мобильное приложение использует слабую политику паролей для упрощения ввода пароля, оно страдает от небезопасной аутентификации; или
- Если мобильное приложение использует такую функцию, как TouchID, оно страдает от небезопасной аутентификации.

Избегайте следующих небезопасных шаблонов проектирования проверки подлинности мобильных приложений:

- При переносе веб-приложения в его мобильный эквивалент требования к проверке подлинности мобильных приложений должны совпадать с требованиями компонента веб-приложения. Поэтому не должно быть возможности аутентификации с меньшими факторами аутентификации, чем веб-браузер;
- Локальная проверка подлинности пользователя может привести к уязвимостям обхода на стороне клиента. Если приложение хранит данные локально, процедуру проверки подлинности можно обойти на взломанных устройствах с помощью манипуляций во время выполнения или модификации двоичного файла. Если существует настоятельная бизнес-потребность в автономной аутентификации;
- По возможности убедитесь, что все запросы проверки подлинности выполняются на стороне сервера. После успешной аутентификации данные приложения будут загружены на мобильное устройство. Это гарантирует, что данные приложения будут доступны только после успешной проверки подлинности;
- Если требуется хранение данных на стороне клиента, данные должны быть зашифрованы с использованием ключа шифрования, надежно полученного из учетных данных пользователя. Это гарантирует, что сохраненные данные приложения будут доступны только после успешного ввода правильных учетных данных. Существуют дополнительные риски того, что данные будут расшифрованы с помощью двоичных атак;
- Функция постоянной аутентификации (Remember Me), реализованная в

мобильных приложениях, никогда не должна хранить пароль пользователя на устройстве;

- В идеале мобильные приложения должны использовать токен аутентификации для конкретного устройства, который может быть отозван пользователем в мобильном приложении. Это гарантирует, что приложение сможет смягчить несанкционированный доступ с украденного /потерянного устройства;
- Не используйте поддельные значения для проверки подлинности пользователя. Это включает в себя идентификаторы устройств или геолокацию;
- Постоянная аутентификация в мобильных приложениях должна быть реализована как согласие и не должна быть включена по умолчанию;
- По возможности не разрешайте пользователям предоставлять 4-значные PIN-коды для паролей аутентификации.

5. M5 - Слабая криптостойкость

Небезопасное использование криптографии можно наблюдать в большинстве приложений. Это почти всегда одна из двух проблем: фундаментально несовершенный процесс, стоящий за механизмами шифрования, или реализация слабого алгоритма.

Если ключи шифрования хранятся некорректно, злоумышленники могут получить к ним доступ через уязвимости чтения произвольных файлов в приложениях. Также если применяется один ключ шифрования для всех клиентов, реально скопировать его и затем расшифровать данные, хранящиеся в приложении. В перспективе это может привести к разным последствиям, вплоть до полной потери аккаунта. Использование устаревших алгоритмов опасно тем, что может стать причиной компрометации пользовательских данных.

Эта уязвимость может иметь ряд различных последствий для бизнеса. Как правило, неработающая криптография приводит к следующему:

- Нарушения конфиденциальности;
- Кража информации;
- Кража кода;
- Кража интеллектуальной собственности.

Многие совершают ошибку, используя правильный алгоритм шифрования, но реализуя свой собственный протокол для его использования. Некоторые примеры

проблем здесь включают в себя:

- Включение ключей в тот же каталог, который можно прочитать злоумышленнику, что и зашифрованное содержимое;
- Предоставление злоумышленнику доступа к ключам;
- Избегайте использования жестко закодированных ключей в вашем двоичном файле; и
- Ключи могут быть перехвачены с помощью двоичных атак.

Всегда используйте современные алгоритмы, которые считаются сильными сообществом безопасности, и по возможности используйте современные API шифрования на своей мобильной платформе. Двоичные атаки могут привести к тому, что злоумышленник определит общие библиотеки, которые вы использовали, вместе с любыми жестко закодированными ключами в двоичном файле.

Многие криптографические алгоритмы и протоколы не следует использовать, поскольку было показано, что они имеют значительные недостатки или иным образом недостаточны для современных требований безопасности. К ним относятся:

- RC2;
- MD4;
- MD5;
- SHA1.

Рекомендуется использовать руководство по шифрованию NIST.

6. M6 – Небезопасная авторизация

В отличие от аутентификации, авторизация имеет дело с проверкой идентифицированного лица. Он проверяет наличие необходимых разрешений для выполнения определенных действий.

Как только злоумышленник понимает, насколько уязвима схема авторизации, он входит в приложение как законный пользователь. Они успешно передают элемент управления проверкой подлинности. После прохождения проверки подлинности они обычно принудительно переходят к уязвимой конечной точке для выполнения административных функций. Этот процесс отправки обычно выполняется через мобильное вредоносное ПО на устройстве.

Чтобы проверить плохие схемы авторизации, тестировщики могут выполнять

двоичные атаки на мобильное приложение и пытаться выполнить привилегированную функциональность, которая должна исполняться только с пользователем с более высокими привилегиями, пока мобильное приложение находится в «автономном» режиме. Кроме того, тестировщики должны попытаться выполнить любую привилегированную функциональность с использованием маркера сеанса с низкими привилегиями в соответствующих запросах POST/GET для конфиденциальной функциональности к внутреннему серверу. Плохие или отсутствующие схемы авторизации позволяют злоумышленнику выполнять функциональные возможности, которые они не должны иметь права использовать аутентифицированного, но с более низкими привилегиями пользователя мобильного приложения. Требования к авторизации более уязвимы при принятии решений об авторизации на мобильном устройстве, а не через удаленный сервер. Это может быть требованием из-за мобильных требований автономного использования.

Существует несколько простых правил, которым нужно следовать при попытке определить, страдает ли мобильная конечная точка от небезопасной авторизации:

- Наличие уязвимостей небезопасной прямой ссылки на объекты (IDOR) — если отображается уязвимость IDOR, код, скорее всего, не выполняет действительную проверку авторизации; и
- Скрытые конечные точки — как правило, разработчики не выполняют проверку авторизации для скрытых функций серверной части, поскольку они предполагают, что скрытая функциональность будет видна только кому-то в нужной роли;
- Передача ролей или разрешений пользователя — если мобильное приложение передает роли или разрешения пользователя серверной системе в рамках запроса, оно страдает от небезопасной авторизации.

7. M7 - Низкое качество кода

Все уязвимости, связанные с ошибками на уровне кода, могут предоставить злоумышленникам путь внутрь. Основной риск заключается в необходимости внесения локализованных изменений в код. В частности, небезопасное использование API или небезопасные языковые конструкции являются распространенными проблемами, которые необходимо исправить непосредственно на уровне кода.

Злоумышленники могут передавать ненадежные входные данные вызовам методов, выполняемым в мобильном коде. Эти типы проблем не обязательно являются проблемами безопасности сами по себе, но приводят к уязвимостям безопасности. Проблемы с плохим качеством кода обычно используются с помощью вредоносных программ или фишинговых атак.

Злоумышленник обычно использует уязвимости этой категории, предоставляя жертве тщательно продуманные входные данные. Эти входные данные передаются в код, который находится в мобильном устройстве, где происходит эксплуатация. Типичные типы атак будут использовать утечки памяти и переполнение буфера.

Как правило, проблем с качеством кода можно избежать, выполнив следующие действия:

- Поддерживать согласованные шаблоны кодирования, с которыми согласны все в организации;
- Пишите код, который легко читается и хорошо документируется;
- При использовании буферов всегда проверяйте, что длина любых входящих данных буфера не будет превышать длину целевого буфера;
- С помощью автоматизации выявлять переполнения буфера и утечки памяти с помощью сторонних инструментов статического анализа; и
- Отдавайте приоритет решению проблем переполнения буфера и утечек памяти по сравнению с другими проблемами «качества кода».

8. M8 – Подделка кода

С технической точки зрения любой код на мобильном устройстве уязвим для подделки. Это связано с тем, что мобильный код выполняется в чужой среде. Он больше не находится под контролем вашей организации. Поэтому существует множество способов изменить его по желанию.

Как правило, злоумышленник делает следующие действия, чтобы воспользоваться этой категорией:

- Внесение прямых двоичных изменений в основной двоичный файл пакета приложения;
- Внесение прямых двоичных изменений в ресурсы в пакете приложения;
- Перенаправление или замена системных API для перехвата и выполнения

вредоносного внешнего кода.

Технически весь мобильный код уязвим для подделки кода. Мобильный код выполняется в среде, которая не находится под контролем организации, производящей код. В то же время существует множество различных способов изменения среды, в которой выполняется этот код. Эти изменения позволяют злоумышленнику возиться с кодом и изменять его по своему желанию.

Мобильное приложение должно иметь возможность обнаруживать во время выполнения, что код был добавлен или изменен по сравнению с тем, что он знает о своей целостности во время компиляции. Приложение должно быть в состоянии соответствующим образом реагировать во время выполнения на нарушение целостности кода.

9. M9 – Реверс-инжиниринг

Злоумышленник обычно загружает целевое приложение из магазина приложений и анализирует его в своей локальной среде, используя набор различных инструментов.

Злоумышленник должен выполнить анализ окончательного двоичного файла ядра, чтобы определить его исходную строковую таблицу, исходный код, библиотеки, алгоритмы и ресурсы, внедренные в приложение. Злоумышленники будут использовать относительно доступные и хорошо понятные инструменты, такие как IDA Pro, Hopper, otool, strings и другие двоичные инструменты проверки из среды злоумышленника.

Как правило, весь мобильный код подвержен обратному проектированию. Некоторые приложения более восприимчивы, чем другие. Код, написанный на языках / фреймворках, которые обеспечивают динамический самоанализ во время выполнения (Java, .NET, Objective C, Swift), особенно подвержен риску обратного проектирования. Обнаружение восприимчивости к обратному проектированию довольно просто.

Злоумышленник может использовать обратный инжиниринг для достижения любого из следующих целей:

- Раскрытие информации о внутренних серверах;
- Выявление криптографических констант и шифров;
- Кража интеллектуальной собственности;

- Выполнять атаки на серверные системы.

Чтобы предотвратить эффективное обратное проектирование, необходимо использовать инструмент обфускации. На рынке есть много бесплатных и коммерческих обфускаторов. И наоборот, на рынке существует множество различных деобфускаторов. Чтобы измерить эффективность любого инструмента обфускации, который вы выберете, попробуйте деобфускировать код с помощью таких инструментов, как IDA Pro и Hopper.

Хороший обфускатор будет обладать следующими способностями:

- Какие методы/сегменты кода запутывать;
- Настройте степень запутывания, чтобы сбалансировать влияние на производительность;
- Выдерживают деобфускацию от таких инструментов, как IDA Pro и Hopper;
- Обфускация строковых таблиц, а также методов.

10. M10 - Скрытый функционал

Как правило, злоумышленник стремится понять посторонние функции в мобильном приложении, чтобы обнаружить скрытые функции в серверных системах. Злоумышленник обычно использует посторонние функции непосредственно из своих собственных систем без какого-либо участия конечных пользователей**.** Они будут изучать файлы журналов, файлы конфигурации и, возможно, сам двоичный файл, чтобы обнаружить любые скрытые переключатели или тестовый код, который был оставлен разработчиками. Они будут использовать эти коммутаторы и скрытые функции в серверной системе для выполнения атаки.

Существует высокая вероятность того, что любое данное мобильное приложение содержит посторонние функции, которые не предоставляются непосредственно пользователю через интерфейс. Большая часть этого дополнительного кода является доброкачественной по своей природе и не даст злоумышленнику никакого дополнительного понимания возможностей серверной части. Однако некоторые посторонние функции могут быть очень полезны злоумышленнику. Функциональные возможности, предоставляющие сведения, связанные с серверными тестовыми, демонстрационными, промежуточными средами или средами UAT, не должны включаться в рабочую сборку. Кроме того, административные конечные точки API или неофициальные конечные точки не

должны включаться в окончательные рабочие сборки. Обнаружение посторонних функций может быть сложным. Автоматизированные инструменты статического и динамического анализа могут подбирать низко висящие плоды (логарифмические выписки). Тем не менее, некоторые бэкдоры трудно обнаружить в автоматизированных средствах. Таким образом, всегда лучше предотвратить эти вещи с помощью ручного обзора кода.

Часто разработчики включают скрытые функции бэкдора или другие внутренние средства управления безопасностью разработки, которые не предназначены для выпуска в рабочую среду.

Лучший способ предотвратить эту уязвимость — выполнить ручную проверку безопасного кода с помощью защитных полей или экспертов по предмету, наиболее хорошо знакомых с этим кодом. Они должны делать следующее:

- Изучить параметры конфигурации приложения, чтобы обнаружить скрытые переключатели;
- Убедитесь, что весь тестовый код не включен в окончательную рабочую сборку приложения;
- Изучить все конечные точки API, к которым обращается мобильное приложение, чтобы убедиться, что эти конечные точки хорошо документированы и общедоступны;
- Изучить все операторы журнала, чтобы убедиться, что в журналы не записывается ничего чрезмерно описательного о серверной части.

Вывод

Как мы видим, злоумышленники могут атаковать мобильные приложения и устройства по многим направлениям. При разработке приложения необходимо проверять возможность осуществления каждого из описанных сценариев атак. Угрозы и уязвимости необходимо учитывать во время разработки, а некоторые необходимые меры защиты — принимать еще на стадии проектирования. В особенности это касается финансовых приложений, личных кабинетов, мессенджеров, а также прочих приложений, работающих с персональными данными, банковскими картами, личной или рабочей перепиской и другой подобной информацией. Хорошей рекомендацией для разработчиков будет внедрение практики безопасной разработки (secure software development lifecycle, SSDL) и регулярного анализа защищенности приложения. Такие меры не только помогут своевременно выявить потенциальные угрозы, но и повысят уровень знаний разработчиков в вопросах безопасности и, как следствие, уровень защищенности разрабатываемых приложений.

В то же время, хотя на разработчиках лежит значительная ответственность по обеспечению безопасности приложений, добиться приемлемого уровня защищенности на мобильных устройствах возможно только принимая комплексные меры.

Приведенный список уязвимостей мобильных приложений может послужить отправной точкой для организаций, разработчиков, специалистов по безопасности и пользователей, которые только начинают решать соответствующие проблемы. Здесь были приведены краткие определения для каждого вида уязвимости, способы их предотвращения.

Список использованных источников

1 - Электронный ресурс - https://mas.owasp.org/MAS_checklist/.

2 - Электронный ресурс - https://vc.ru/s/1153268-informacionnye-tehnologii/395772-short-list-samyh-populyarnyh-uyazvimostey-v-mobilnyh-prilozheniyah.

3 - Электронный ресурс - https://owasp.org/www-project-mobile-top-10/.

4 - Электронный ресурс - https://detoxtechnologies.com/top-10-attacks-and-vulnerabilities-of-owasp-mobile/.