14 лекция.

Тема: Настройки безопасности, разрешения, шифрование данных при разработке и использовании мобильных приложений.

План:

1. **Защита мобильного приложения**
2. **Защита мобильного приложения**
3. **Дополнительные рекомендации**

При разработке мобильного приложения следует учитывать, что данные, которыми оперирует это приложение, могут представлять определенный интерес для третьих лиц. Степень ценности этих данных варьируется в широких пределах, тем не менее, даже наиболее простая приватная информация, например, пароль входа в приложение, требует проработки ее защиты. Особенно это важно в свете распространения мобильных приложений на все сферы электронных услуг, включая финансовые, банковские операции, хранение и передачу личных данных и так далее. Всем интересующимся — добро пожаловать под кат.

Все нижесказанное — исключительно мой опыт, безусловно, данные могут быть неточными, поэтому буду благодарен за любые поправки и дополнения к статье. Я не нашел исчерпывающих статей в сети на подобную тематику, которые бы собирали всю нужную (хотя бы базовую) информацию в одном месте, поэтому решил обобщить свой опыт в этой области на текущий момент времени.

1. Защита мобильного приложения

Основные виды атак на мобильное приложение:

- Декомпиляция файла приложения (.ipa-файлы для Apple iOS и .apk-файлы для Google Android) и разбор локально сохраненных данных. Защита этого, наиболее важного в настоящее время, уровня целиком лежит на плечах мобильного разработчика.
- Перехват данных, передаваемых по сети (MITM-атаки). Большинство мобильных приложений являются клиент-серверными, следовательно, постоянно передают и принимают большие объемы информации. И хотя современная мобильная и веб-разработка активно завершают переход на HTTPS-протокол общения, тем не менее, не стоит полагаться на единственный рубеж защиты в виде защищенного канала связи.

- Рутование устройства и атака на приложение и применяемые в нем алгоритмы через внешние отладочные инструменты.

Перечень основных уязвимостей приложений

Рассмотрим уязвимости общего характера, без привязки к конкретной платформе. Здесь и далее используется аббревиатура КВД — критически важные данные пользователей. К КВД относятся любые данные, которые не должны быть доступны третьей стороне, это касается как персональных данных пользователя (дата рождения, адрес проживания, личная переписка), так и его приватных данных (пароли, данные кредитных карт, номера банковских счетов, номера заказов и так далее).

Перечень основных уязвимостей следующий:

- Использование незащищенных локальных хранилищ.
- **Опасность:** Очень высокая.
- **Комментарий:** Встречается повсеместно, выражается в хранении КВД в незащищенных или слабо защищенных локальных хранилищах, специфических для конкретной платформы. Вскрытие третьей стороной — элементарное, и, как правило, не требуется наличие специальных навыков у атакующего.
- **Защита:** Хранить КВД можно только в защищенных хранилищах платформы.

- Хранение КВД в коде.
- **Опасность:** Высокая.
- **Комментарий:** Уязвимость касается хранения КВД внутри кода (в статических константных строках, в ресурсах приложения и т.п.). Яркие примеры: хранение соли для пароля (password salt) в константе или макросе, которая применяется по всему коду для шифрования паролей; хранение приватного ключа для асимметричных алгоритмов; хранение паролей и логинов для серверных узлов или баз данных. Легко вскрывается третьей стороной при наличии базовых навыков декомпиляции.
- **Защита:** Не хранить никакие КВД в коде или ресурсах приложения.

- Применение алгоритмов с хранением приватного ключа.

    - **Опасность:** Высокая.
    - **Комментарий:** Уязвимость актуальна в случае, если приватная информация алгоритма (приватный ключ) вынужденно сохраняется в коде или ресурсах мобильного приложения (чаще всего так и бывает). Легко вскрывается методом декомпиляции.
    - **Защита:** В мобильной разработке желательно применять только современные симметричные алгоритмы с генерируемым случайным одноразовым ключом, обладающие высокой стойкостью с взлому методом грубой силы, либо выводить асимметричный приватный ключ за пределы приложения, либо персонализировать этот ключ (как пример — приватным ключом может выступать пользовательский код входа, сохраненный в зашифрованном виде в защищенном хранилище операционной системы).

- Использование асимметричного алгоритма с приватным ключом, известным серверу.

    - **Опасность:** Зависит от степени защищенности сервера.
    - **Комментарий:** Уязвимость носит двойной характер. Хранение приватного ключа допускает возможность расшифровки пользовательских данных на стороне сервера. Во-первых, это некорректно с точки зрения безопасности (если сервер будет взломан — атакующий также получит доступ к приватным данным пользователей), а во-вторых, это нарушает приватность персональных данных. Пользователь всегда должен быть уверен, что его персональная информация не известна никому, кроме него самого (только если он явно не дал разрешение на ее публикацию). Часто приложения позиционируют себя как защищенные, но на деле таковыми не являются, так как содержат внутри себя средства для расшифровки персональной информации.
    - **Защита:** Без явной необходимости и явного разрешения пользователя (чаще всего через лицензионное соглашение) ни приложение, ни сервер не должны иметь никакой возможности расшифровать приватные данные пользователя. Простейший пример — пароль пользователя должен уходить на сервер уже в

виде хеша, и проверяться должен хеш, а не исходный пароль (серверу абсолютно незачем знать пользовательский пароль; если же пользователь его забыл — для такой ситуации существует давно отлаженный механизм восстановления пароля, в том числе с двухфакторной авторизацией клиента для повышенной безопасности процедуры восстановления).

- Использование самописных алгоритмов шифрования и защиты.

  - **Опасность:** Средняя.
  - **Комментарий:** Это прямое нарушение принципа Керкгоффса. Выражается в попытке разработчика изобрести "свой личный, не известный никому, а поэтому супер-защищенный алгоритм шифрования". Любое отклонение от существующих, многократно проверенных и изученных, математически доказанных алгоритмов шифрования в 99% случаев оборачивается быстрым взломом подобной "защиты". Требует наличия средне-высоких навыков у атакующего.
  - **Защита:** Следует подбирать подходящий алгоритм только из отлаженных и актуальных общеизвестных криптографических алгоритмов.

- Передача КВД во внешнюю среду в открытом виде.
  - **Опасность:** Средняя.
  - **Комментарий:** Выражается в передаче КВД без применения шифрования по любому доступному каналу связи с внешней средой, будь то передача данных стороннему приложению или передача в сеть. Может быть вскрыто опосредованно путем вскрытия не приложения, а его хранилища, или целевого приложения. Взлом требователен к наличию навыков у атакующего, при условии, что хранилище является защищенным.
  - **Защита:** Любые КВД перед выходом за пределы приложения должны быть зашифрованы. Локальные хранилища платформы ***не являются*** областью приложения, они тоже должны получать на вход только зашифрованные данные.

- Игнорирование факта наличия рутованных или зараженных устройств.

    - **Опасность:** Средняя.
    - **Комментарий:** Рутованные устройства — это девайсы, где выполнена модификация для получения прав суперпользователя на любые операции, изначально запрещенные производителем операционной системы. Выполняется пользователем на своем устройстве самостоятельно, и не обязательно добровольно (клиент может быть не в курсе, что устройство взломано). Установка приложения на рутованный девайс нивелирует все штатные средства защиты операционной системы.
    - **Защита:** Если это технически возможно для платформы — то желательно запрещать работу приложения, если удалось понять, что запуск производится на рутованном устройстве, или хотя бы предупреждать об этом пользователя (спасибо за дополнение DjPhoeniX).

- Хранение КВД в защищенных хранилищах, но в открытом виде.

    - **Опасность:** Средняя.
    - **Комментарий:** Разработчики зачастую склонны сохранять КВД в защищенные системные хранилища без дополнительной защиты, поскольку системные механизмы хорошо сопротивляются взлому. Однако уровень их стойкости падает до минимума в случае, если устройство рутованное.
    - **Защита:** КВД не должны использоваться в приложении без дополнительного шифрования. Как только надобность в "открытых" КВД отпала — они немедленно должны быть либо зашифрованы, либо уничтожены.

- Перевод части функционала во встроенные веб-движки.

    - **Опасность:** Средняя.
    - **Комментарий:** Чаще всего выглядит как передача КВД во встроенный браузер, где загружается внешняя веб-страница, выполняющая свою часть функционала. Уровень защиты в этом случае резко снижается, особенно для рутованных устройств.

- **Защита:** Не использовать встроенный браузер и встроенный веб-движок в операциях с КВД. На крайний случай — шифровать КВД перед передачей.

- Реверсивная инженерия алгоритмов, представляющих интеллектуальную ценность.

  - **Опасность:** Низкая, зависит от ценности алгоритма.
  - **Комментарий:** Если при разработке приложения внутри компании используются некие собственные алгоритмы, которые могут представлять высокую ценность для потенциальных конкурентов или взломщиков, то эти алгоритмы должны быть защищены от постороннего доступа.
  - **Защита:** Автоматическая или ручная обфускация кода.

Специфика разработки мобильных приложений

Есть несколько общих для всех мобильных платформ моментов, которые следует соблюдать при разработке.

Защита пользовательским кодом

- Если приложение защищено пользовательским паролем (PIN-кодом, сканом отпечатка пальца, графическим паролем и т.д.), то при уходе приложения в фон ("сворачивании") оно должно немедленно отображать окно ввода этого защитного кода, перекрывая собой весь экран приложения. Это исключает возможность для злоумышленника получить приватную информацию в случае кражи устройства, пока приложение все еще запущено и находится в спящем режиме.
- Любой пользовательский код должен иметь ограниченное количество попыток ввода (например, 5 раз), затем, в случае неудачи, приложение должно автоматически разлогиниваться (или и вовсе блокироваться, зависит от конкретного приложения).
- В настоящее время при использовании цифровых кодов строго рекомендуется использовать ограничение на длину кода в минимум 6 цифр (больше можно, меньше — нельзя).

2. Функционирование клиент-серверного приложения

- Для клиент-серверных приложений очень полезно применять сессионный механизм с ограниченным временем жизни сессии. Это позволит избежать "простаивания" приложения в незащищенном режиме, если пользователь просто забыл закрыть его и оставил устройство в свободном доступе. Следует учитывать, что срок действия сессии и ее идентификатор относятся к КВД, со всеми вытекающими отсюда последствиями. Одним из удачных примеров реализации подобного механизма является получение абсолютного значения времени с сервера после прохождения процедуры авторизации пользователя (дата и время должны показывать, когда именно сессия станет неактивной). Дату и время окончания действия сессии не следует генерировать на устройстве, это снижает безопасность и гибкость приложения.
- Клиент-серверное приложение не должно осуществлять изменение КВД в локальном режиме. Любое действие, требующее изменения КВД, должно проходить синхронизацию с сервером. Исключение из этого правила составляет только пользовательский код входа, задаваемый лично пользователем и сохраненный в защищенном локальном хранилище.

Работа с датами

- При оперировании важными для работы приложения датами, вроде времени уничтожения сессии, не следует опираться на относительное время. То есть, передаваемые с сервера данные не должны содержать дату в виде "плюс N секунд/часов/дней от текущего момента". В силу наличия потенциально высоких задержек в передаче данных по сети от мобильного приложения к серверу и обратно, подобный способ синхронизации будет обладать слишком большой погрешностью. Кроме того, атакующий (или просто недобросовестный пользователь) может попросту сменить локальный пояс на устройстве, нарушив таким образом логику работы ограничительных механизмов приложения. Всегда нужно передавать только абсолютное значение времени.
- Абсолютные значения следует передавать с применением универсальных способов обмена подобной информацией, без привязки к часовому поясу конкретного пользовательского устройства. Чаще всего, оптимальным вариантом является поведение приложения, при котором данные отображаются пользователю в его локальном часовом

поясе, но их хранение и передача осуществляется в формате, не привязанном к тайм-зоне. Подходящими форматами для дат и времени являются либо универсальный UNIX timestamp, сохраненный в переменной 64-битного целого знакового типа (UNIX timestamp — это количество секунд, прошедшее с 1 января 1970 года), либо, на крайний случай, строка в полном формате ISO-8601 с нулевой тайм-зоной. Предпочтителен именно UNIX timestamp, он позволяет избежать потенциальных ошибок и проблем с конвертацией строк в дату и обратно на разных мобильных платформах.

3. Дополнительные рекомендации

- Приложение не должно отображать приватную пользовательскую информацию большими, яркими, хорошо читаемыми шрифтами, без явной на то необходимости и без отдельного запроса пользователя, чтобы исключить возможность чтения этих данных издали с экрана устройства.
- Не стоит слепо доверять библиотекам с открытым исходным кодом, которые предлагают некую защиту приватным данным пользователей. Исключение составляют библиотеки, проверенные временем и используемые в крупных проектах корпораций (например, встроенное шифрование в открытом движке базы данных Realm). Штатных механизмов защиты операционной системы и общедоступных проверенных криптографических алгоритмов в подавляющем большинстве случаев будет более, чем достаточно.
- Абсолютно недопустимо использовать криптографические библиотеки с закрытым исходным кодом (даже если они платные). В таких решениях вы никак не сможете проверить, насколько эффективна данная библиотека, а также насколько "честная" у нее защита (нет ли там механизма backdoor, или не отсылаются ли "защищенные" данные какой-то третьей стороне).
- В релизных сборках приложений должно быть отключено логгирование данных в системную консоль и незащищенные файлы. Специфические логи для разработчиков могут присутствовать, но желательно в зашифрованном виде, во избежание доступа третьих лиц к закрытой служебной информации, которая может присутствовать в логах.