12. Безопасность Android-приложений. Permissions (разрешения) в Android. Безопасность данных (шифрование, хранение ключей).

Цели лекции:

1. Ознакомить с основами безопасности Android-приложений.

2. Изучить систему разрешений в Android.

3. Рассмотреть подходы к шифрованию данных.

4. Изучить методы хранения ключей.

5. Проанализировать примеры уязвимостей и защиты данных.

Лекция посвящена безопасности Android-приложений, в которой рассматриваются ключевые аспекты, связанные с защитой данных и управлением разрешениями. В ходе лекции студенты узнают о системе разрешений в Android, ее значении для безопасности приложений и правилах запроса у пользователей. Также будет обсуждено шифрование данных, методы их защиты, а также хранение ключей для обеспечения конфиденциальности. Приведенные примеры и практические рекомендации помогут студентам лучше понять, как реализовать безопасность в своих приложениях и минимизировать риски утечек данных.

**Введение**

С каждым годом мобильные приложения становятся все более распространёнными и важными в нашей повседневной жизни. Android, как одна из самых популярных мобильных платформ, предоставляет разработчикам широкие возможности для создания разнообразных приложений, однако, это также связано с рядом вызовов в области безопасности. Безопасность приложений критически важна, поскольку утечки данных и уязвимости могут привести к серьёзным последствиям для пользователей и разработчиков. В этой лекции мы подробно рассмотрим основные аспекты безопасности Android-приложений, включая управление разрешениями, шифрование данных и методы безопасного хранения ключей.

1. Основы безопасности Android-приложений

Безопасность в разработке программного обеспечения является неотъемлемой частью жизненного цикла приложения. В Android, как и в других платформах, безопасность можно рассматривать через несколько основных призмов:

- Конфиденциальность: Пользователи ожидают, что их личная информация будет защищена от несанкционированного доступа. Это включает в себя не только личные данные, такие как имена и адреса, но и более чувствительную информацию, такую как финансовые данные и данные о местоположении.

- Целостность: Данные, хранящиеся и обрабатываемые приложением, должны оставаться неизменными, если они не были специально модифицированы авторизованным образом. Вредоносные приложения могут попытаться изменить данные пользователя или настроить их неправильно.

- Доступность: Приложение должно быть доступно для законных пользователей и работать должным образом. Успешные атаки могут привести к сбоям в работе приложения или полной его недоступности.

- Доверие: Пользователи доверяют приложениям, которые обеспечивают высокую безопасность. Низкий уровень безопасности может повредить репутации разработчика и привести к потерям пользователей.

2. Система разрешений в Android

Система разрешений в Android — это ключевой компонент обеспечения безопасности, который контролирует доступ приложений к конфиденциальным данным и ресурсам устройства. Начиная с версии Android 6.0 (Marshmallow), система разрешений была усовершенствована, и теперь разработчики должны запрашивать разрешения у пользователей динамически.

2.1 Типы разрешений

Разрешения в Android делятся на несколько категорий:

- Обычные разрешения: Эти разрешения не затрагивают конфиденциальные данные и обычно предоставляются автоматически. Например, разрешение на доступ к интернету.

- Опасные разрешения: Эти разрешения требуют явного согласия пользователя, так как они могут затрагивать личные данные или функции устройства. Например, доступ к камере, контактам, геолокации и т.д.

2.2 Запрос разрешений

С динамическим запросом разрешений приложение должно запрашивать доступ к ресурсам в момент, когда это действительно необходимо. Например, если приложение запрашивает доступ к камере, оно должно сделать это в момент, когда пользователь пытается сделать снимок.

Пример кода для запроса разрешения на доступ к камере:

```java

if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)

!= PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.CAMERA}, REQUEST\_CAMERA);

}

```

2.3 Обработка разрешений

Необходимо правильно обрабатывать случаи, когда пользователь отказывает в предоставлении разрешения. Важно информировать пользователей о причинах запроса разрешений и о том, как это влияет на функциональность приложения. Если разрешение не было предоставлено, разработчики должны быть готовы ограничить функциональность приложения или завершить его работу.

Пример обработки результата запроса разрешения:

```java

@Override

public void onRequestPermissionsResult(int requestCode,

String permissions[], int[] grantResults) {

switch (requestCode) {

case REQUEST\_CAMERA: {

if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

// Разрешение предоставлено

} else {

// Разрешение отклонено

}

return;

}

}

}

```

3. Защита данных: шифрование и хранение ключей

Шифрование данных — это ключевая стратегия обеспечения безопасности информации. Шифрование позволяет защитить данные, преобразуя их в неразборчивый вид, который может быть расшифрован только с использованием правильного ключа.

3.1 Шифрование данных

Существует несколько подходов к шифрованию данных:

- Симметричное шифрование: Использует один и тот же ключ для шифрования и расшифровки данных. Это обеспечивает быструю обработку, но требует надежного способа хранения ключа. Примеры алгоритмов: AES (Advanced Encryption Standard).

- Асимметричное шифрование: Использует пару ключей — открытый и закрытый. Открытый ключ используется для шифрования данных, а закрытый — для их расшифровки. Пример: RSA (Rivest-Shamir-Adleman).

Для шифрования данных в Android разработчики могут использовать классы `Cipher` и `SecretKeySpec`. Пример использования симметричного шифрования с AES:

```java

SecretKeySpec secretKey = new SecretKeySpec(key.getBytes("UTF-8"), "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT\_MODE, secretKey);

byte[] encryptedData = cipher.doFinal(data.getBytes("UTF-8"));

```

3.2 Хранение ключей

Безопасное хранение криптографических ключей является критически важным аспектом защиты данных. Разработчики должны учитывать следующие методы:

- Keystore API: Android предоставляет `KeyStore` для безопасного хранения ключей. Он использует аппаратные средства устройства для защиты ключей и предотвращения их несанкционированного доступа. Ключи, хранящиеся в `KeyStore`, могут быть использованы для шифрования и расшифровки данных, и даже если устройство будет скомпрометировано, ключи останутся защищёнными.

Пример использования `KeyStore` для создания и хранения симметричного ключа:

```java

KeyGenerator keyGenerator = KeyGenerator.getInstance(KeyProperties.KEY\_ALGORITHM\_AES);

keyGenerator.init(new KeyGenParameterSpec.Builder("my\_key",

KeyProperties.PURPOSE\_ENCRYPT | KeyProperties.PURPOSE\_DECRYPT)

.setBlockModes(KeyProperties.BLOCK\_MODE\_CBC)

.setEncryptionPaddings(KeyProperties.ENCRYPTION\_PADDING\_PKCS7)

.build());

SecretKey secretKey = keyGenerator.generateKey();

```

- SharedPreferences: Не рекомендуется использовать для хранения ключей. Если это необходимо, важно шифровать данные перед сохранением, чтобы предотвратить несанкционированный доступ.

3.3 Пример шифрования и хранения данных

При хранении конфиденциальных данных, таких как пароли или токены доступа, рекомендуется следовать следующему процессу:

1. Шифровать данные с использованием симметричного шифрования (например, AES).

2. Хранить зашифрованные данные в `SharedPreferences` или базе данных.

3. Хранить ключ шифрования в `KeyStore` для безопасного доступа.

Пример кода для шифрования и хранения пароля:

```java

String password = "user\_password";

byte[] encryptedPassword = cipher.doFinal(password.getBytes("UTF-8"));

SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs", MODE\_PRIVATE);

sharedPreferences.edit().putString("encryptedPassword", Base64.encodeToString(encryptedPassword, Base64.DEFAULT)).apply();

```

4. Примеры уязвимостей и защита данных

Несмотря на наличие инструментов и методов для обеспечения безопасности, разработчики должны быть внимательны к потенциальным уязвимостям. Некоторые распространенные уязвимости включают:

- Отсутствие шифрования данных: Если данные не шифруются, они могут быть легко украдены или подменены. Это может произойти как при передаче данных через интернет, так и при их хранении на устройстве.

- Слабые пароли: Использование простых паролей или отсутствие их использования может привести к взлому учетных записей пользователей. Рекомендуется использовать хэширование паролей (например, с использованием bcrypt или PBKDF2) перед их хранением.

- Неправильная обработка разрешений: Если приложение запрашивает избыточные разрешения, это может вызвать недоверие у пользователей и снизить вероятность его установки. Разработчики должны запрашивать только те разрешения, которые необходимы для работы приложения.

- Уязвимости в сторонних библиотеках: Использование библиотек с известными уязвимостями может привести к рискам. Необходимо следить за обновлениями библиотек и использовать только проверенные.

5. Практические рекомендации по безопасности

Обновление приложений: Регулярно обновляйте приложения для исправления уязвимостей и добавления новых функций безопасности. Убедитесь, что пользователи знают о доступных обновлениях.

Аудит безопасности: Проводите регулярные проверки безопасности своих приложений с использованием статических и динамических анализаторов кода. Это поможет выявить потенциальные уязвимости и недочеты в коде.

Обучение пользователей: Обучайте пользователей правильным практикам безопасности, таким как использование сложных паролей, осмотрительность при установке приложений и предоставлении разрешений.

Шифрование при передаче данных: Используйте HTTPS для всех передач данных между клиентом и сервером. Это защитит данные от перехвата.

Тестирование на проникновение: Проводите тестирование на проникновение, чтобы выявить уязвимости и слабые места в приложении.

Логирование и мониторинг: Ведите логи действий приложения, чтобы отслеживать подозрительную активность и реагировать на инциденты безопасности.В заключение, безопасность Android-приложений — это комплексный и многоуровневый процесс, который включает в себя управление разрешениями, шифрование данных и безопасное хранение ключей. Разработчики должны осознавать важность этих аспектов и активно работать над тем, чтобы создавать надежные и безопасные приложения, которые защищают данные пользователей. Важно постоянно следить за новыми угрозами и адаптировать свои приложения к изменяющимся условиям безопасности, чтобы обеспечить высокое доверие и удовлетворенность пользователей. Инвестирование времени и ресурсов в безопасность приложений — это не только защита интересов пользователей, но и основа долгосрочного успеха разработчика на рынке мобильных приложений.

Видеоматериал для дополнительного обучения лекции:

<https://youtu.be/t7GiWN2muEE?feature=shared>