Unmasking the Godfather

Reverse Engineering the Latest Android Banking Trojan

Laurie Kirk

whoami

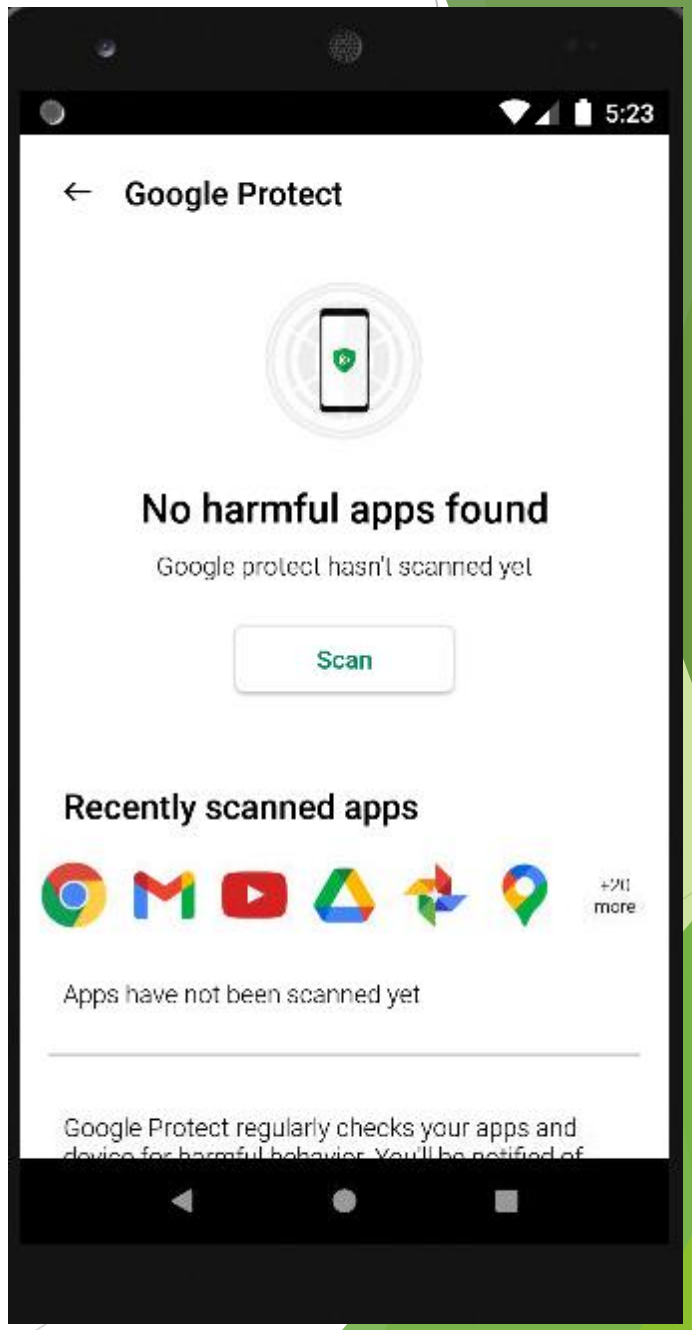
- Laurie Kirk
- Reverse Engineer at Microsoft
- Specialize in cross-platform malware with a focus on mobile malware
- Run YouTube channel @lauriewired
- Representing myself as an individual security researcher today (not representing Microsoft)

@lauriewired

Analysis Materials

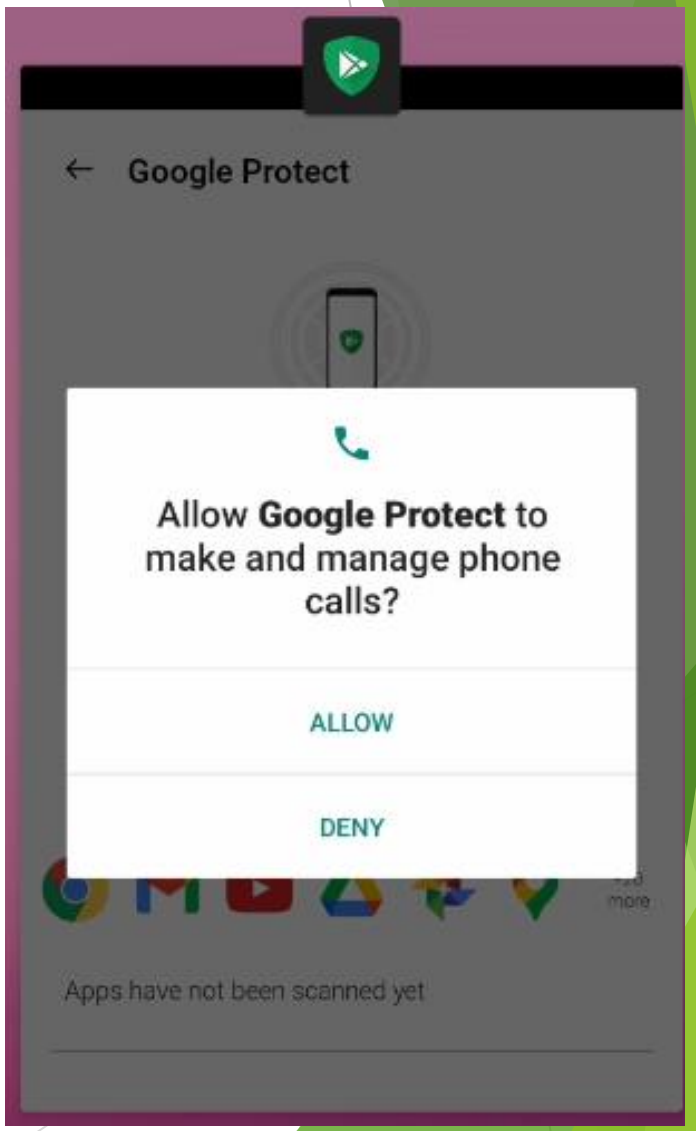
- LaurieWired StrangeLoop Github Repo
  - https://github.com/LaurieWired/StrangeLoop
- SHA256: a14aad1265eb307fbe71a3a5f6e688408ce153ff19838b3c5229f26ee3ece5dd

This application promising to protect you…



is actually going to steal your banking credentials.


"SECURITY" APPLICATION NEEDS TO MAKE PHONE CALLS
SEEMS LEGIT


Google Protect
Allow Google Protect to make and manage phone calls?
ALLOW
DENY
Apps have not been scanned yet

The Eternal Struggle Against Banking Trojans

- Plaguing Android users since 2011
- Billions of downloads from Google Play Store
  - Prevalent families: Godfather, Anubis, Cerberus, SharkBot
- Masquerade as legitimate applications

The Origin of The Godfather

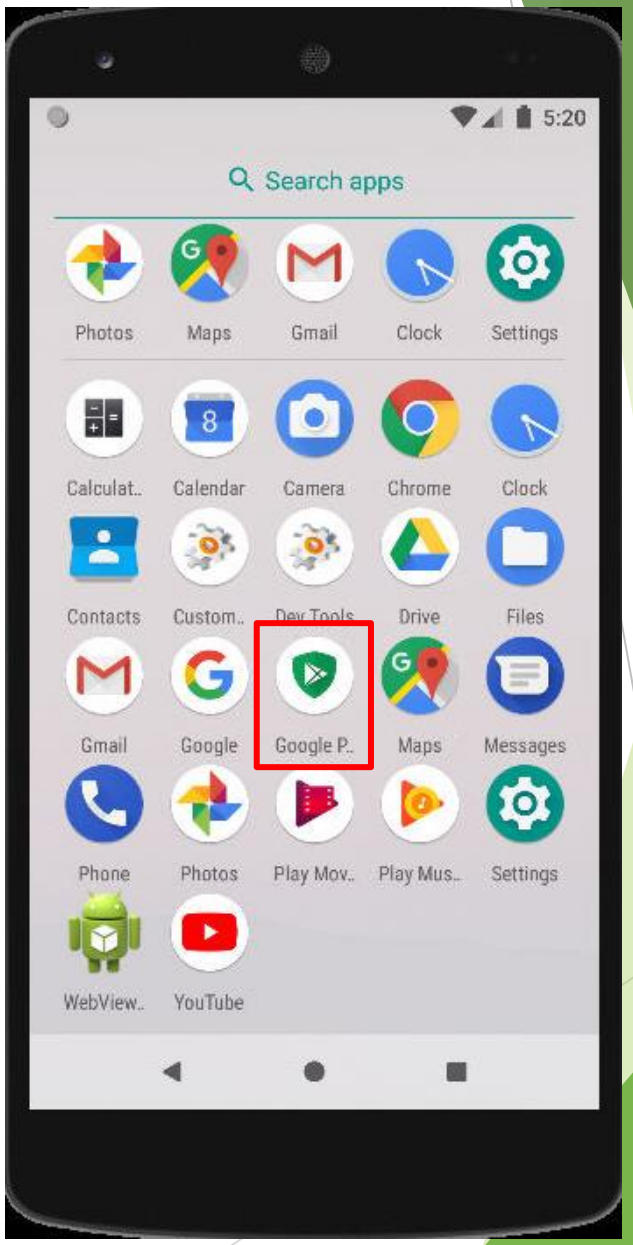
- More than 10 million downloads from Google Play Store
- Targets over 400 financial institutions across 16 countries
- First seen in 2021 and still used today
- Codebase is derived from notorious Anubis malware

If banking trojans have been around so long, why are they still effective?

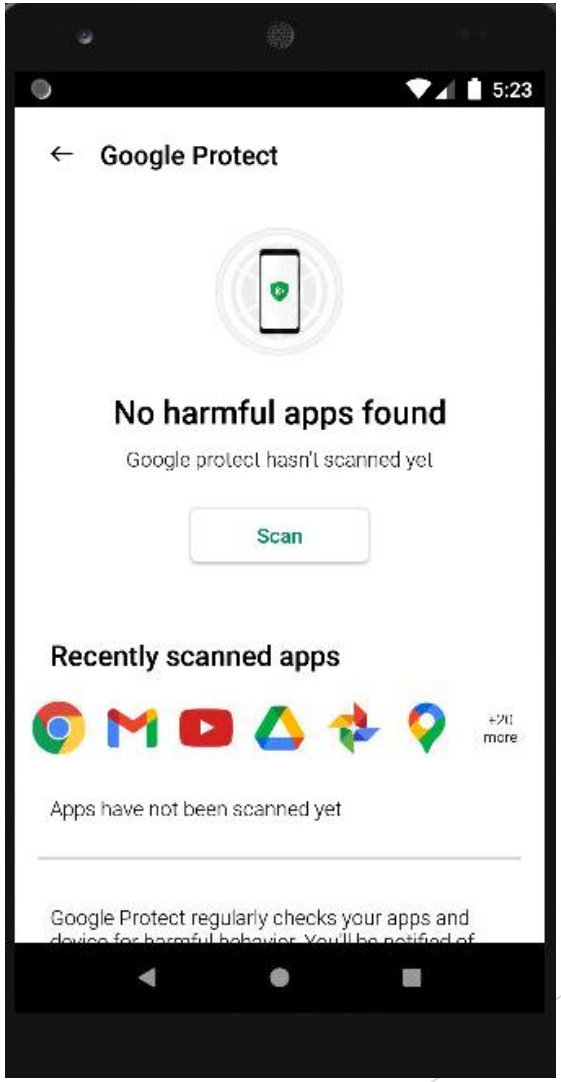
Let’s dive into The Godfather to find out!

Google Protect Icon

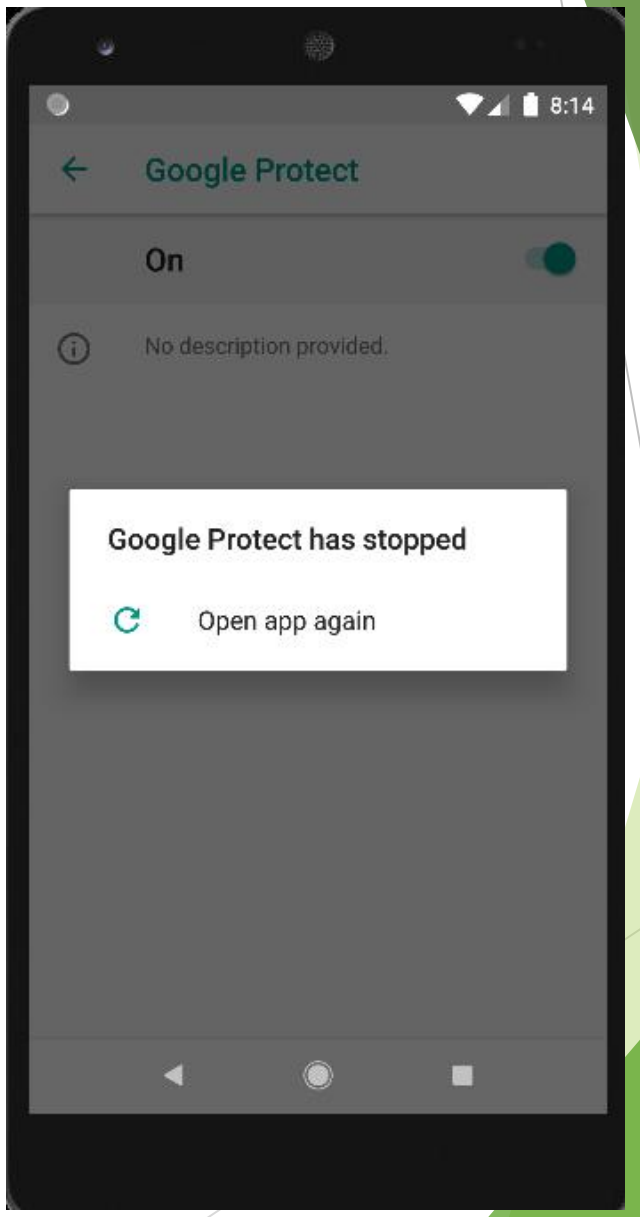
- Google Protect is a legitimate application
- Scans device for harmful behavior



Google Protect Activity

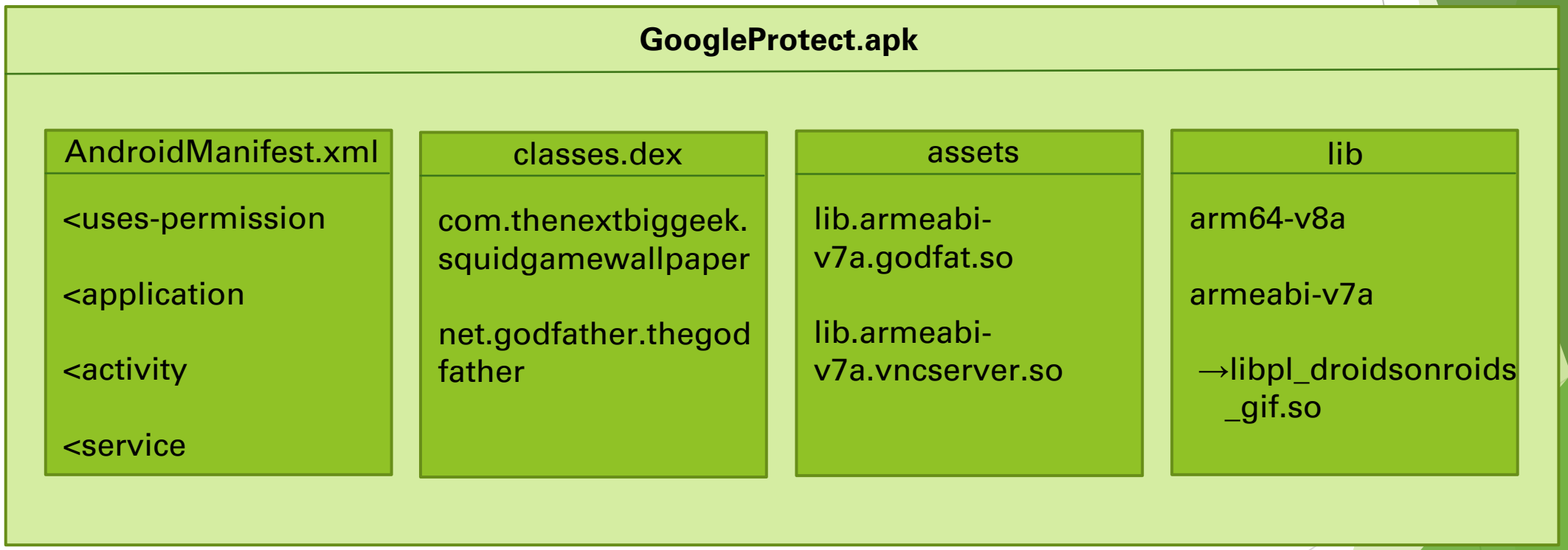


Running the app causes a crash



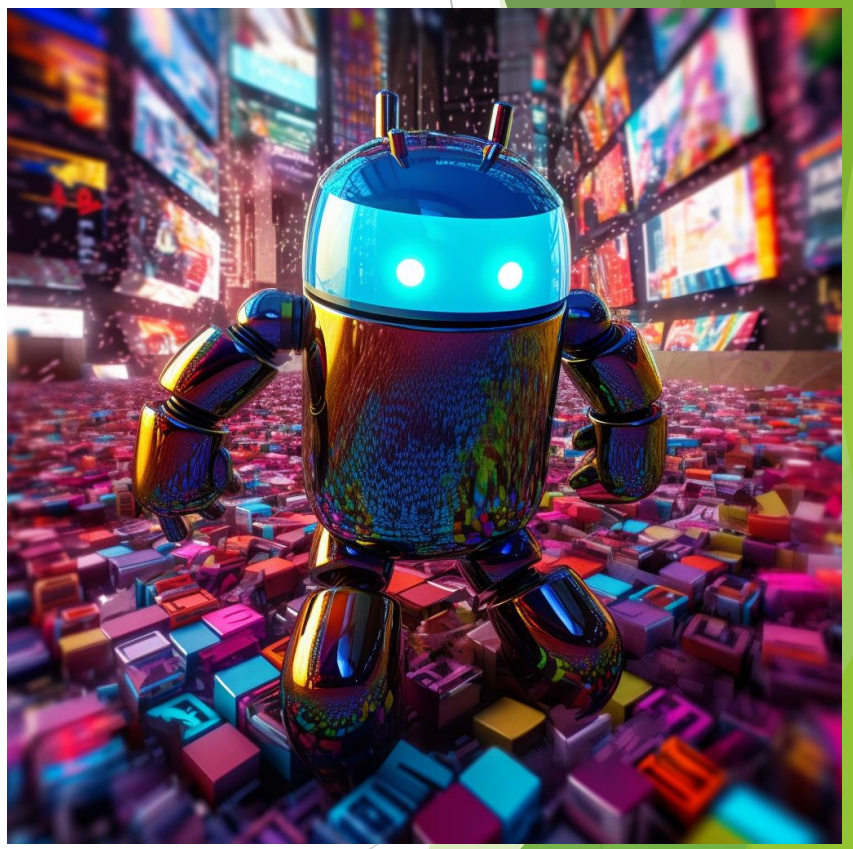
We’re going to have to look at the code.

High-Level Application Structure



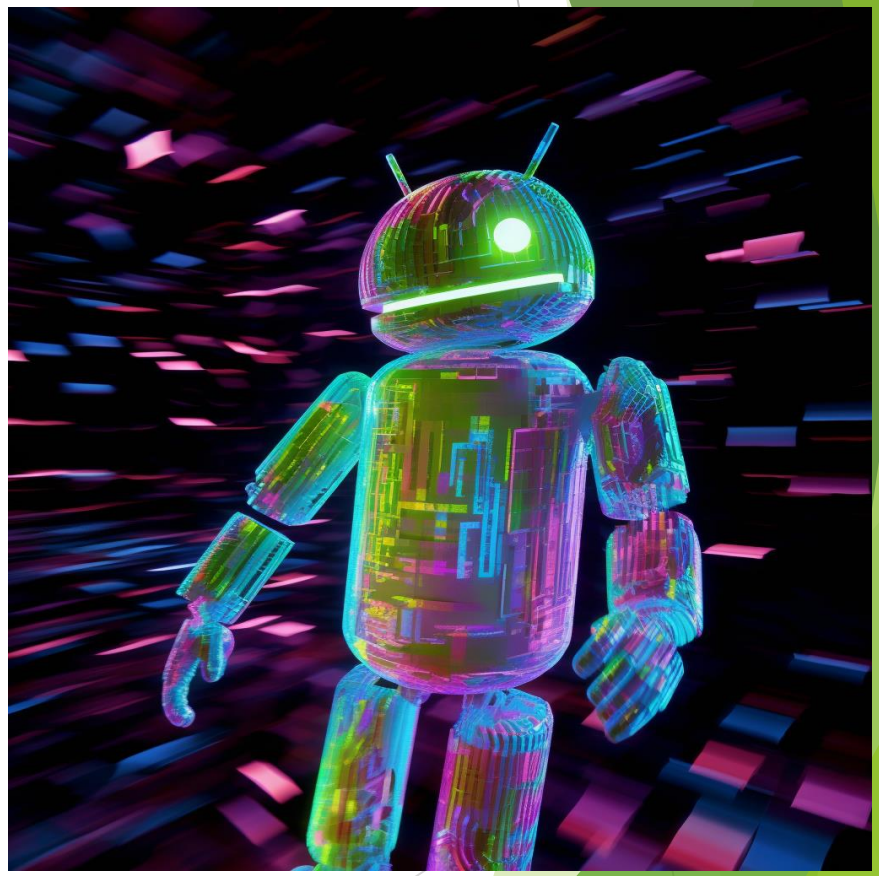
Important Android Components

- Defined in the AndroidManifest.xml
  - Components can run simultaneously in the foreground or background
- Activities
  - User interacts with activities
  - Main foreground components



Android Services and Receivers

- Services
  - Code executes in the background
- Receivers
  - Waits for a certain event to run



Hands On:
Finding the Entrypoint

Why is this code difficult to read?

Obfuscation Techniques

First of all, what is obfuscation?

- Obfuscation obscures app data and functionality
- Common among all platforms
- Offensive and defensive motivations for obfuscation
- Essential for Android
  - Decompiled into pretty Java code

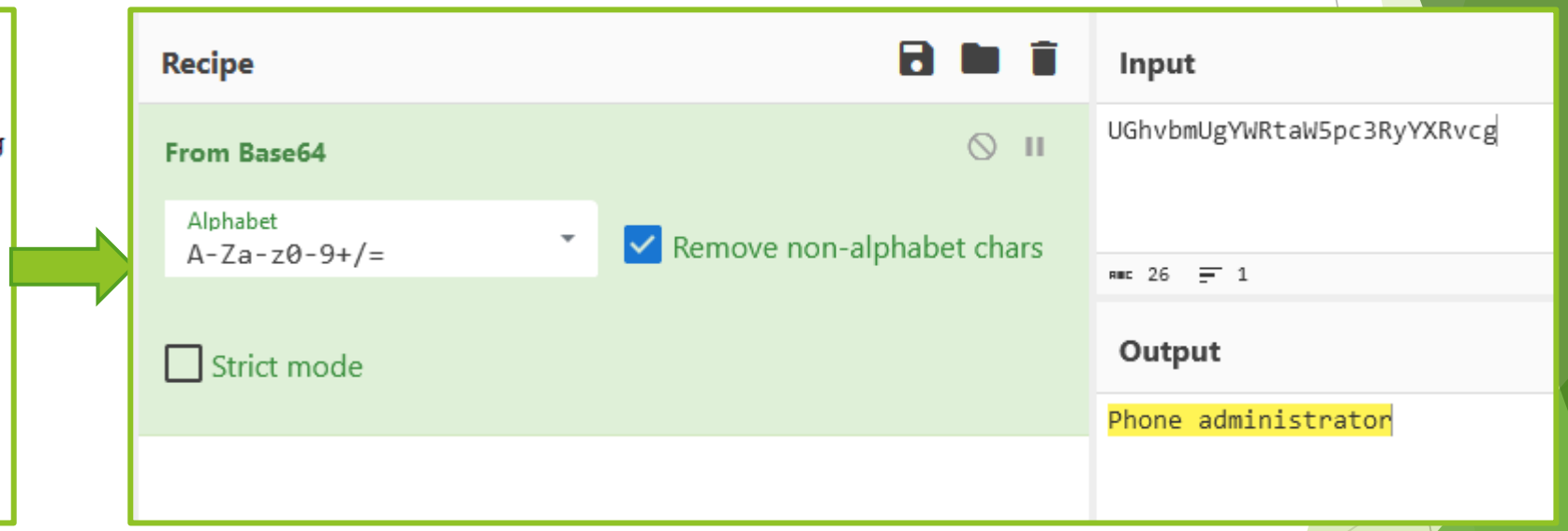
Junk Code Insertions

- Uncalled methods
- Pad application with nonsense
- Empty if-statements
- Special character strings

```
@Override // android.app.Service
public void onTaskRemoved(Intent intent) {
    if ((8 + 17) % 17 <= 0) {
    }
    String str = "";
    while (true) {
        switch ((str.hashCode() ^ 978) ^ 491991272) {
            case -867391267:
                super.onTaskRemoved(intent);
                str = "";
                break;
            case 424828093:
                return;
            case 644549326:
                str = "";
                break;
            case 1358770060:
                str = "";
                break;
        }
    }
}
```

Decoding Strings with Cyberchef

```
TWVzc2FnZXM
ZGV2aWNlIGFkbWluIGFwcA
bm90aWZpY2F0aW9ucw
UGhvbmUgYWRtaW5pc3RyYXRvcg
U3RhcnQgbm93
U3RhcnQgbm93
U3RhcnQgbm93
ZGV2aWNlIGFkbWlu
ZGV2aWNlIGFkbWlu
VXNlIHNlcnZpY2U
asadadad
asadadad
QXBwZWFyIG9uIHRvcA
b3ZlciBvdGhlciBhcHBz
```



Decoded Strings

Base64 Decoded English Value
Enable accessibility for protection to take effect
System Files Cannot be Removed!
Please activate for updates to be active
device admin app
Phone administrator
Use service
over other apps

Now we've found the malicious code, but it's wrapped in anti-emulation.

Anti-Emulation

- Avoids executing on Android emulators
  - Prevent reverse engineering
- Heuristic device checks

Device Characteristic Checks

Fingerprint	Generic
Model	Emulator, Android SDK built for x86
Brand	generic_x86
Device	vbox86p
Manufacturer	Genymotion, unkown
Hardware	Goldfish

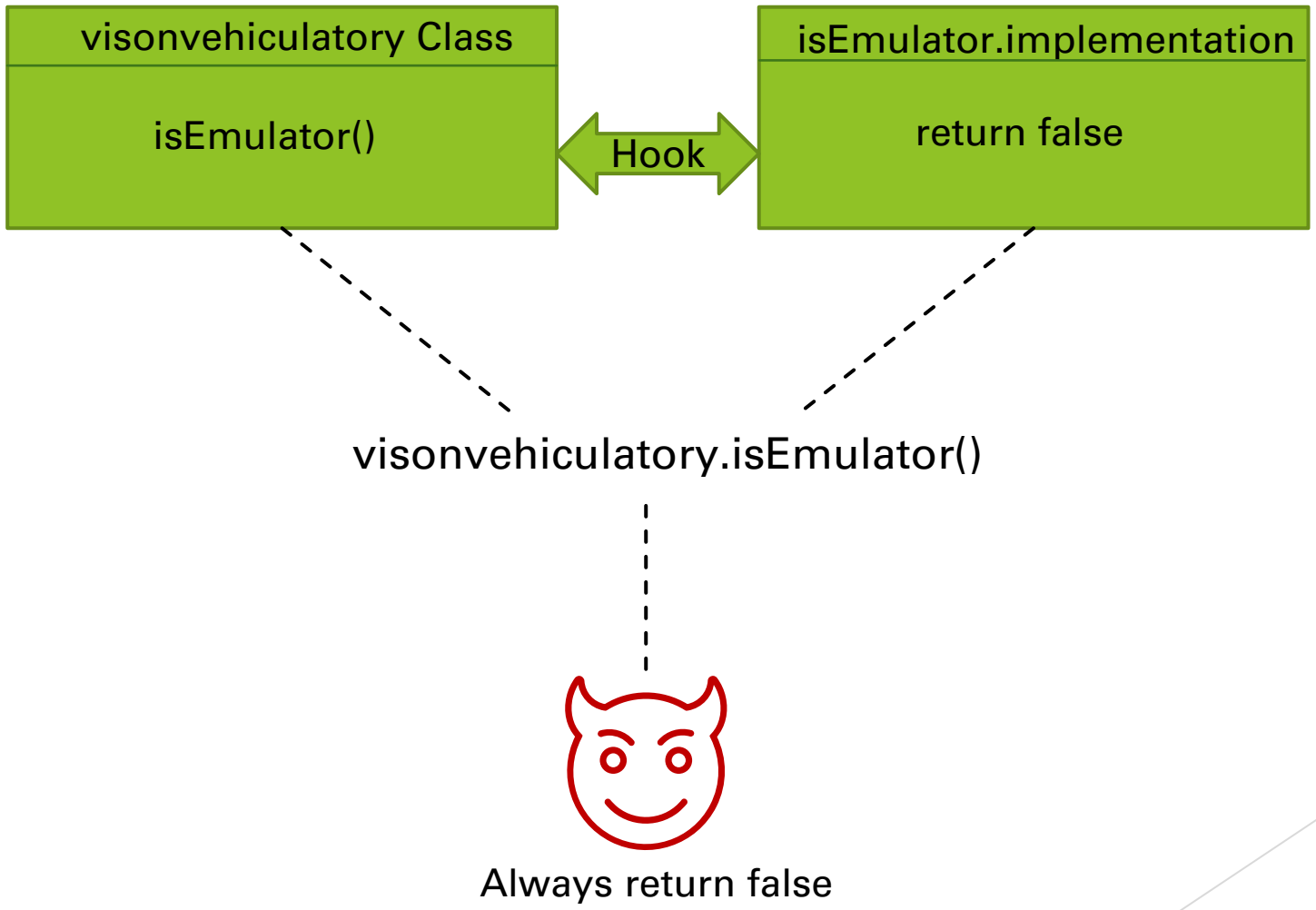
If isEmulator returns true, the device hangs

```
String locate = Resources.getSystem().getConfiguratic
if (ArrayUtils.contains(this.mw_countriesExcludeLis
    finish();
} else if (this.mw_mainWorkClass.isEmulator()) {
} else {
    if (this.mw_mainWorkClass.PRead(this, "key") ==
```

Defeating Anti-Emulation with Hooking

- Frida is a multi-platform code instrumentation toolkit
- Write new method functionality during runtime

Defeating Anti-Emulation with Frida



Demo:
Using Frida to Defeat Anti-Emulation

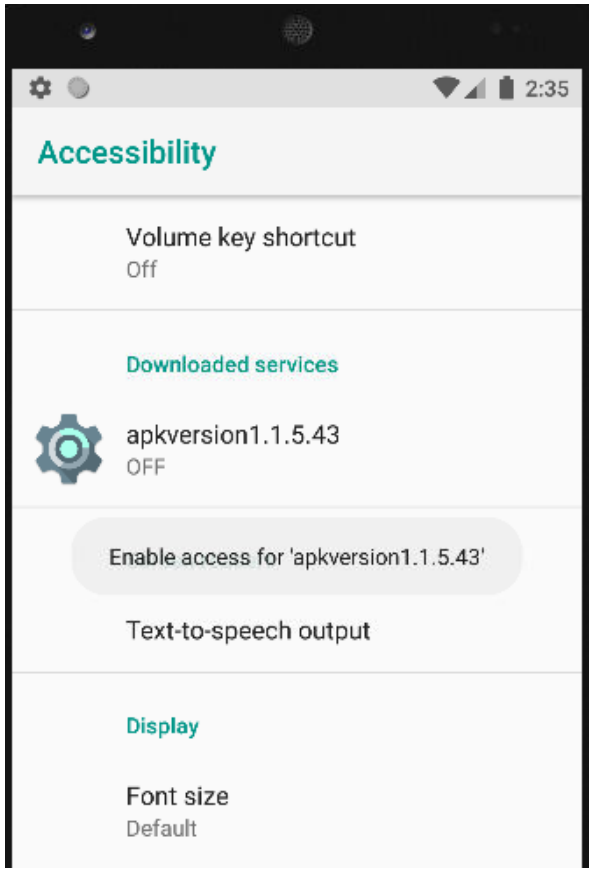
Or you could just run an ARM emulator… lol

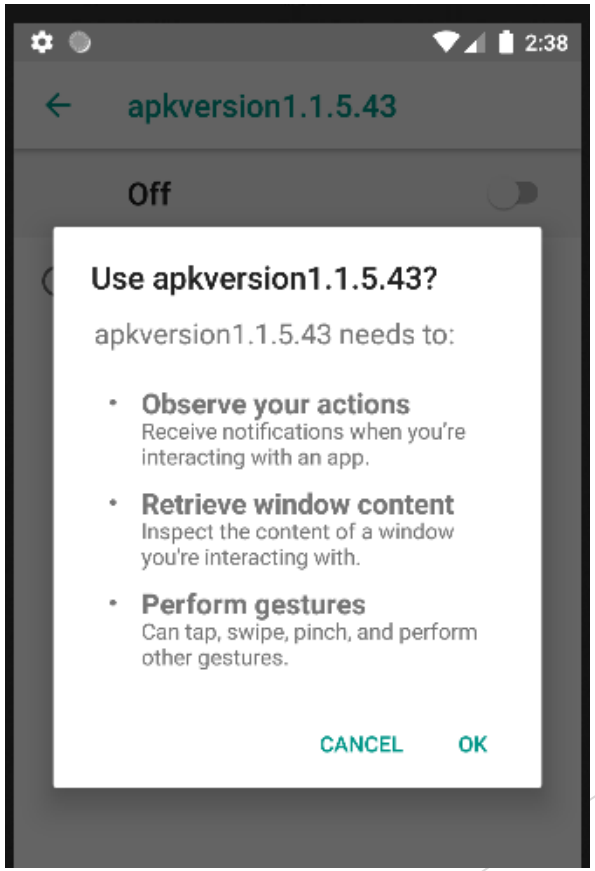
Why did they keep spamming accessibility requests though?

Accessibility Features

- Legitimate Android feature
  - Provides additional functionality for vision, audio, and mobility needs
- Allows an app to perform extra device manipulation
- Does not require user approval

All Godfather Variants Spam Accessibility





Summary of Accessibility Attempts

- Shared among all Godfather variants
- Repeated popup in the center of the screen
- Alarm triggered until accessibility enabled
- Constantly brings user back to settings page

It seems like they want us to enable accessibility settings.

We need to keep digging into the code to find out why.

Hands On:
Analyzing the “Godfather” Module

We finally know why they were so pushy about accessibility!

HTML Phishing Pages

Check foreground application

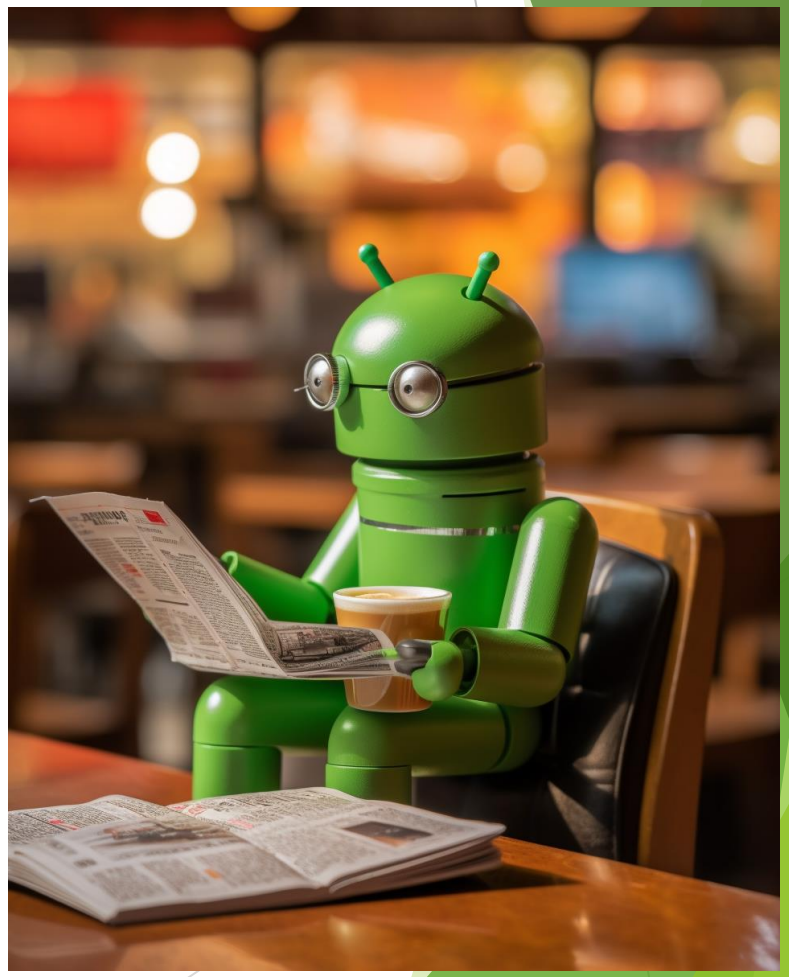
Create new WebView client class

Load HTML from malicious URL

Overlay fake webpage on top of legitimate app

Victims Enter Sensitive Data into Fake Pages

- Abuse accessibility to capture screen data
- Use regular expressions to search for patterns of interest
  - Pins, passwords



Parsing Pins with Regular Expressions

```
Pattern mPattern = Pattern.compile("^([0-9•]{1,16})$");
Matcher matcher = mPattern.matcher(text);
AccessibilityNodeInfo pin_field = mw_findDataInAccessibilityNode(rootNode, "pinEntry");
if (pin_field != null && matcher.find()) {
    if (!text.replace("•", "").isEmpty() && text.length() >= 4) {
        return "PIN_GOOD:" + text;
    }
    return "PIN_PART:" + text;
}
return "PASSWORD:" + text;
```


ENABLES ACCESSIBILITY TO
SHUT OFF THE ANNOYING NOTIFICATIONS.
BANK ACCOUNT NOW SHOWING NEGATIVE

Posting Data to URL

- Gathers device data and recorded malicious events
- Stores encrypted command and control server
- Base64 encodes event data
  - POSTs data to the C2 server

Screen Recording

- Records screen data
  - Using built-in Android MediaRecorder class
- Saves to MP4 file
- Uploads file to C2 server

Full Godfather Commands and Capabilities

Command String	Action
startUSSD	Call phone (USSD)
startApp	Start specified app on the device
startforward	Forward calls on the device
openbrowser	Open specified URL in default browser
killbot	Open the settings for the current app
startPush	Start the WebView activity with a malicious URL
startsocks5	Open socket connection
open (array)	VNC session, keylogger, video recorder, screen locker

Summarize Our Findings

Obfuscation Used by the Godfather

- Meaningless identifiers
- String / class encryption
- Junk code insertions
- Anti-emulation checks
- Native code

Config with SharedPreferences

- Hides strings by using a key-value pair to hold the config
- Allows custom behavior per infected device
  - Stores malicious URL, whether accessibility enabled, keylogger active
  - Allows device characteristic checking during runtime

Avoids Execution for Certain Countries

Code	Country
RU	Russia
AZ	Azerbaijan
AM	Armenia
BY	Belarus
KZ	Kazakhstan
KG	Kyrgyzstan
MD	Moldova
UZ	Uzbekistan
TJ	Tajikistan

Components

Services

- Runs malicious Godfather service
- Receives remote commands

Receivers

- Awaits notification of Accessibility permissions granted

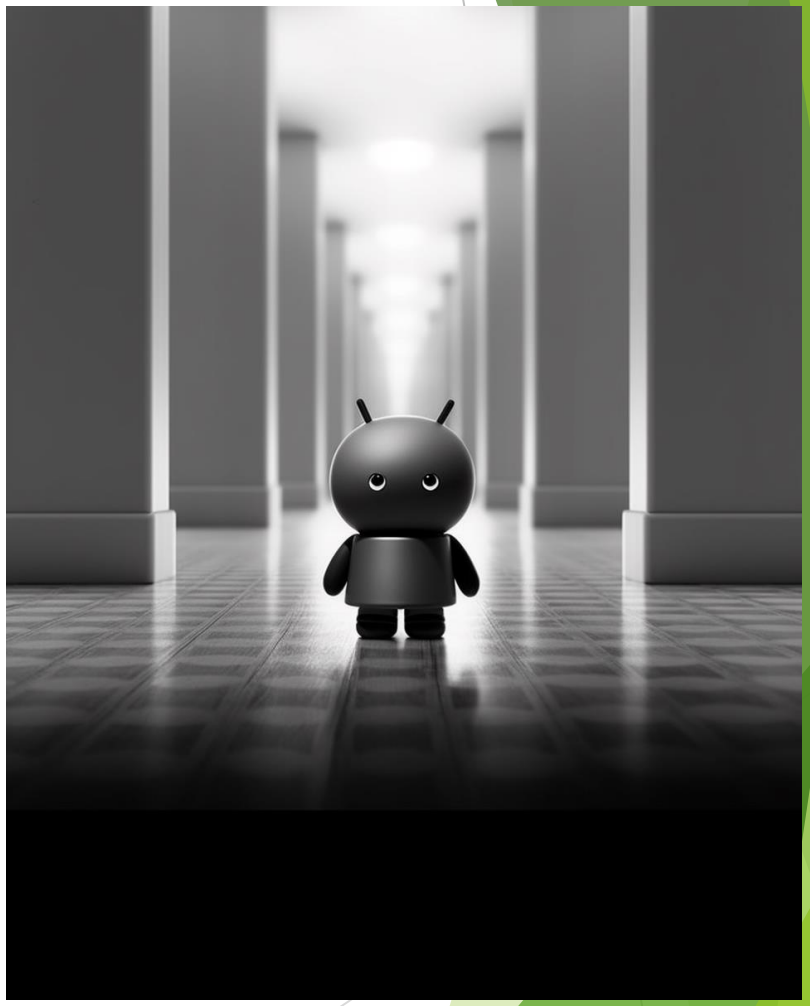
Activities

- Trojanized Google Protect interface
- Fake WebView pages

Android Banking Trojans In the Wild

Targets

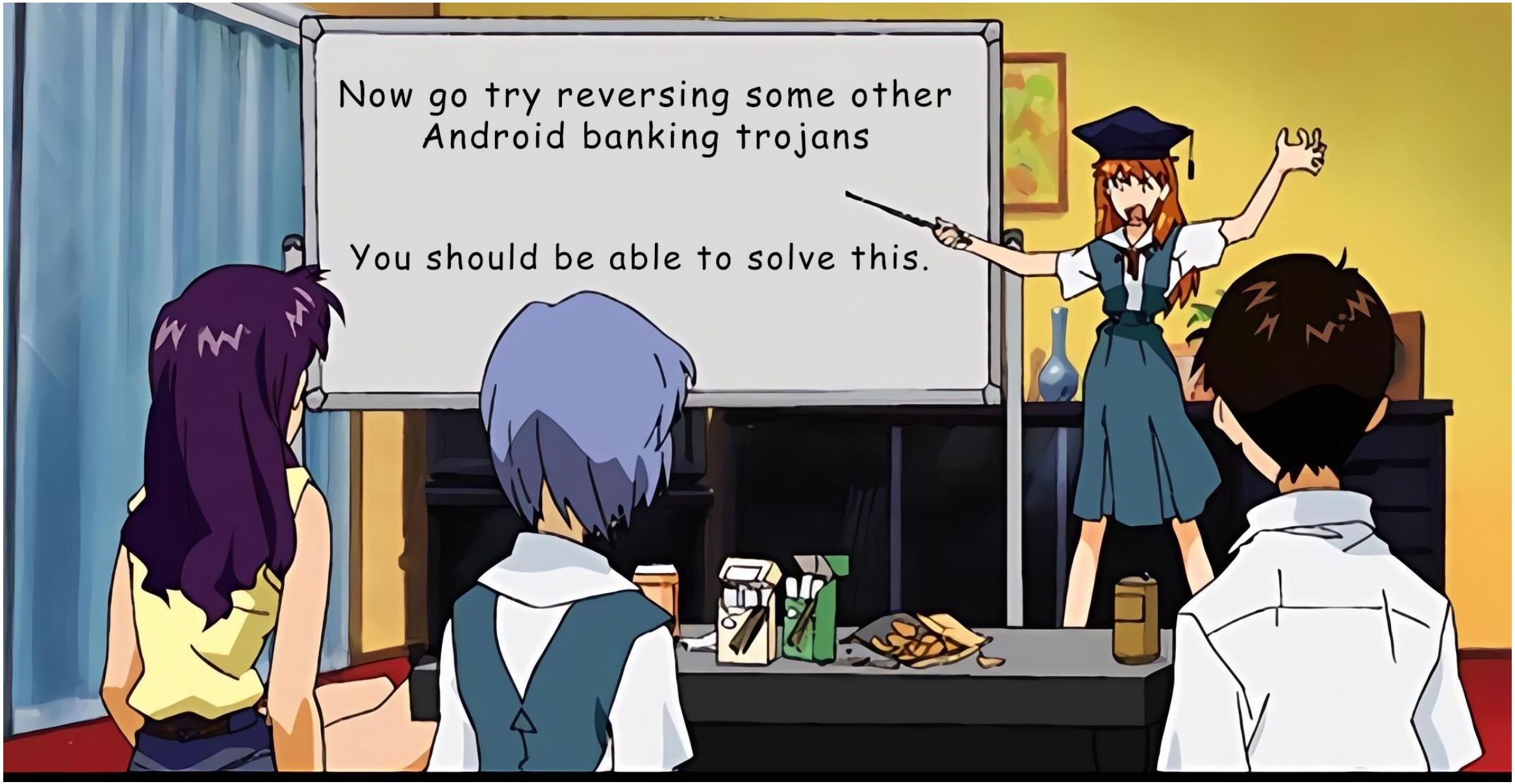
- Financial applications
- Authenticators and OTP generators
- Cryptocurrency apps



Common Capabilities

- Abuse accessibility services
- Create fake HTML overlays to steal credentials
- Spy on infected device screens and SMS messages
- Perform commands from command-and-control (C2) server
- Intercept 2FA one-time-passwords (OTPs)

That seems familiar. Didn’t we already reverse engineer that?


Now go try reversing some other Android banking trojans
You should be able to solve this.

Thank you!

Bonus Section

Godfather IOCs

- Marked up JADX file: https://github.com/LaurieWired/StrangeLoop
- SHA256: a14aad1265eb307fbe71a3a5f6e688408ce153ff19838b3c5229f26ee3ece5dd
- Another Godfather Example
  - SHA256: 0b72c22517fdefd4cf0466d8d4c634ca73b7667d378be688efe131af4ac3aed8

Other Banker IOCs

- Cerberus
  - https://bazaar.abuse.ch/sample/c81234b6ceb3572c6d862a9313e019b98efd83165d8c085bd3e74971c66763bb/
- Anubis
  - https://bazaar.abuse.ch/sample/731c0da8d74adbb557a0abd4ec2aa6c61e09d429560d76549881f08e564b27cd/
- Sharkbot
  - https://bazaar.abuse.ch/sample/71c78101f7792fe879a082e323fed89c5e4a43132d01d3f79ed02afd8db45497/

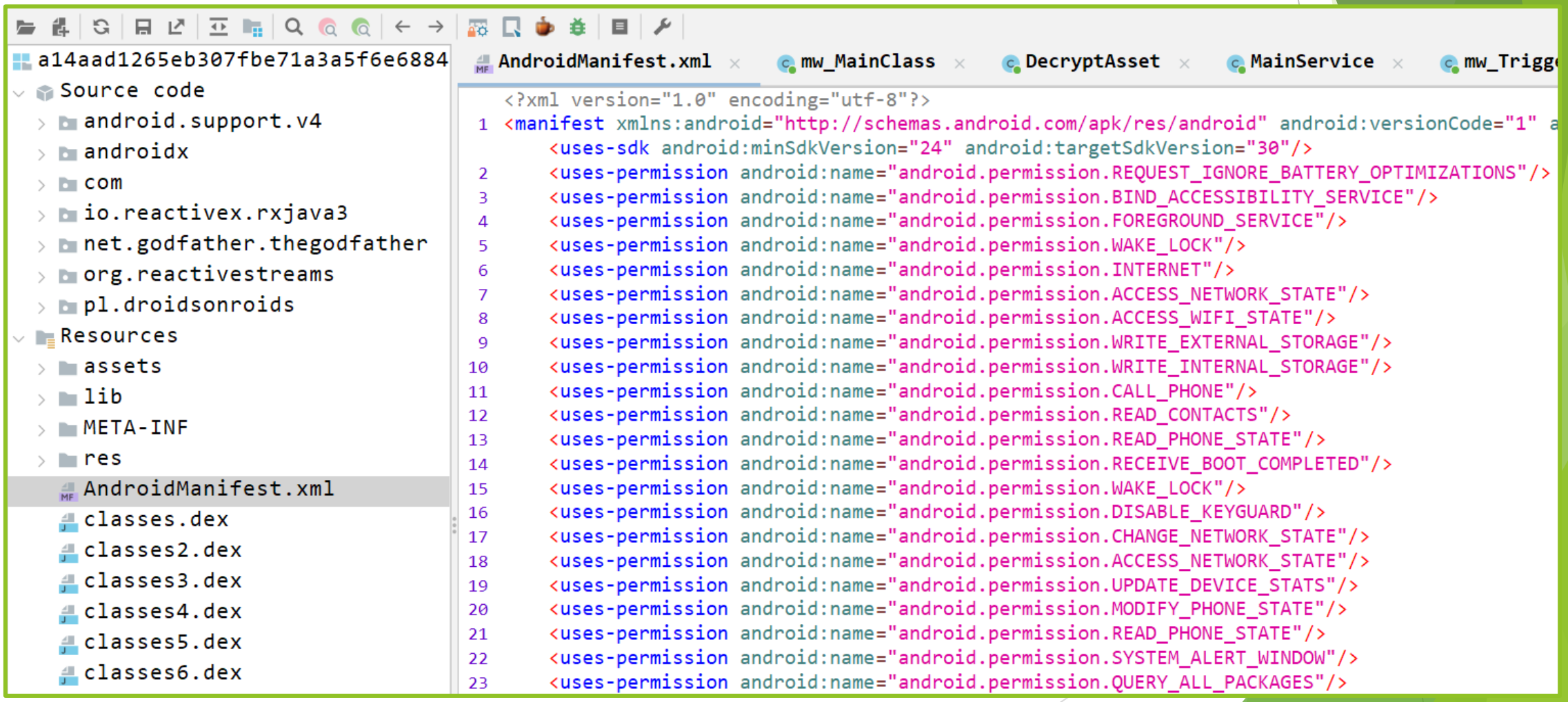
Android Analysis Tools

- JADX: Java decompiler / disassembler for Android
  - https://github.com/skylot/jadx
- Ghidra: C / C++ decompiler / disassembler
  - https://ghidra-sre.org/
- Docker-android: emulator for Android
  - https://github.com/budtmo/docker-android
- Recaf: Up-and-coming Java bytecode editor
  - https://github.com/Col-E/Recaf

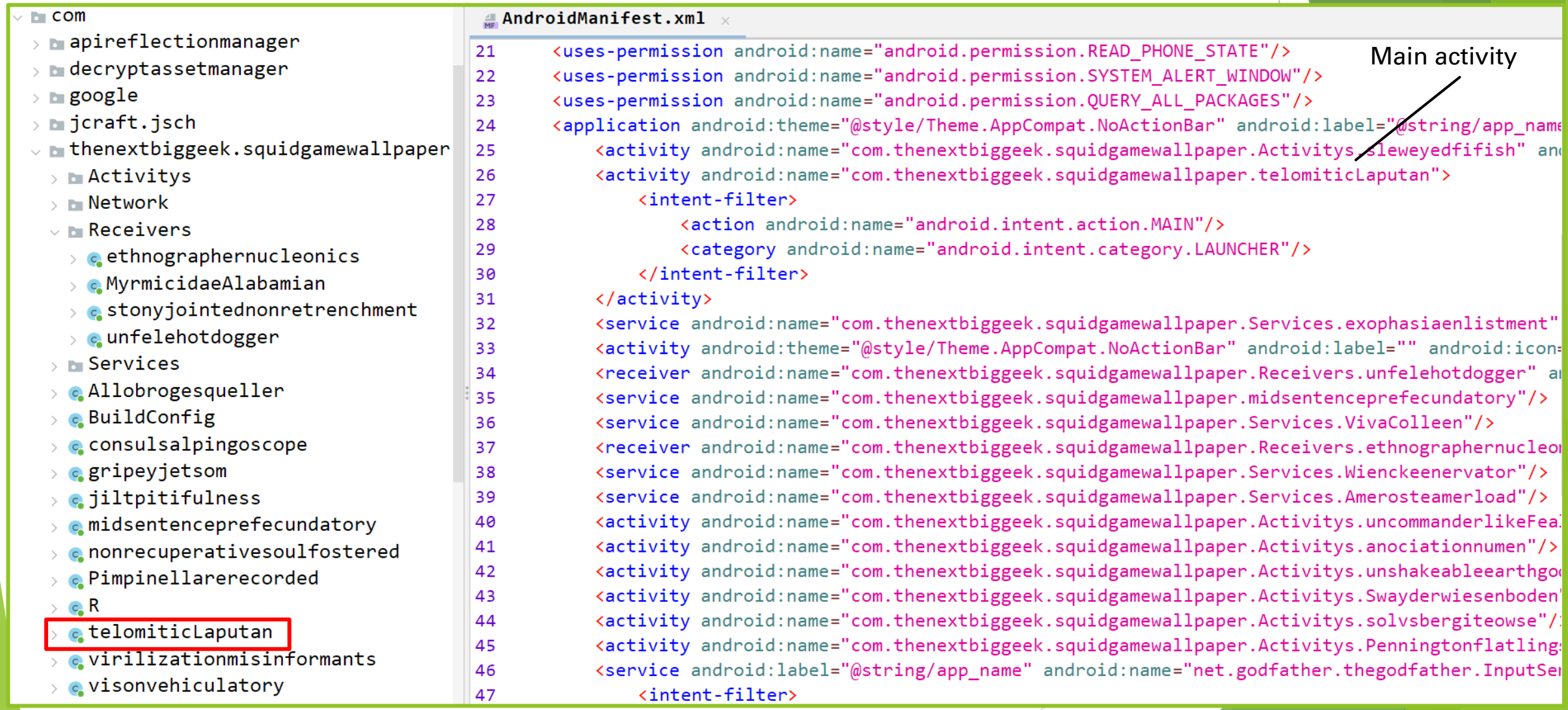
Other Resources

- Full Anubis banker analysis (in progress)
  - https://www.youtube.com/watch?v=Vs9Z3NDnVT8
- Hooking Android methods with Frida
  - https://www.youtube.com/watch?v=RJXsvAjZl9U
- Running an Android ARM emulator
  - https://www.youtube.com/watch?v=fTT5hxiMv6I

Permissions

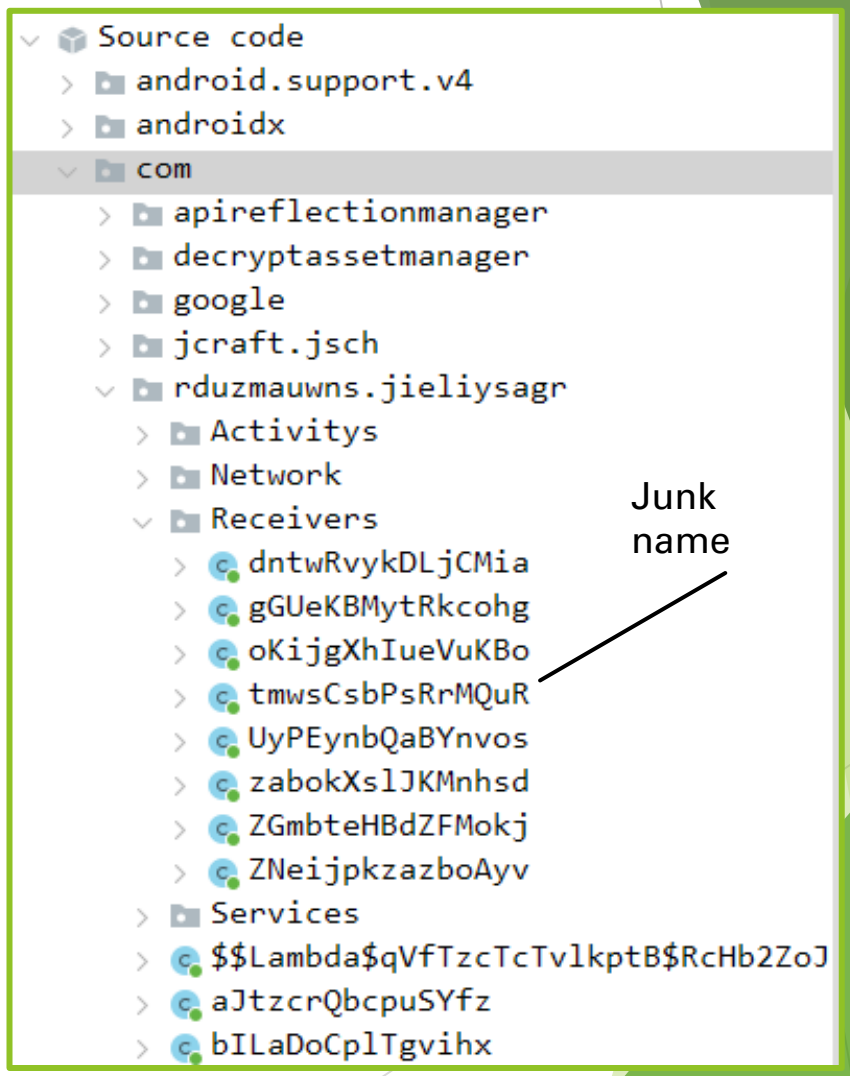


Deconstructing the Manifest



Identifier Renaming

- Rename classes, methods, and variables
- Change to meaningless names
- By default, Android apps include original developer names



Custom Frida JavaScript

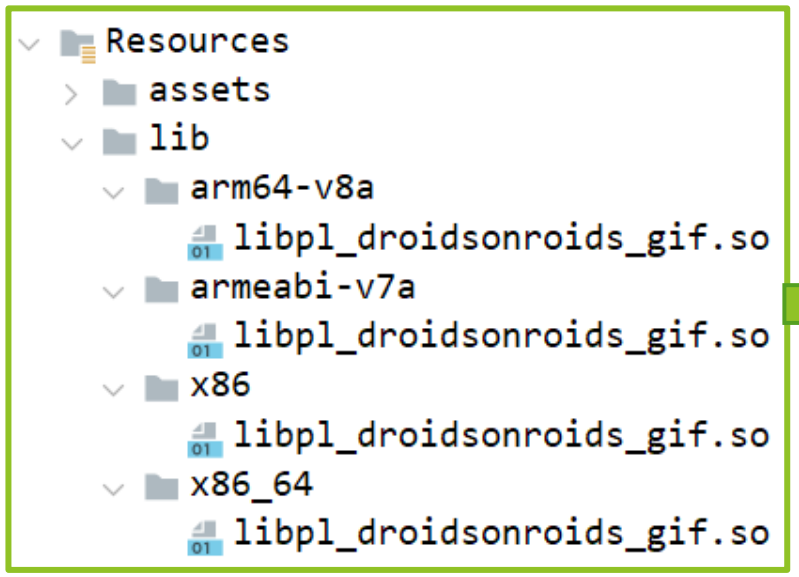
Class to hook

New functionality

```
Java.perform(() => {
  const antiEmClass = Java.use('com.thenextbiggeek.squidgamewallpaper.visonvehiculatory');

  antiEmClass.isEmulator.implementation = function () {
    send('Hooking anti-em method. Always return false...');
    return false;
  };
});
```

Benign Native Binary

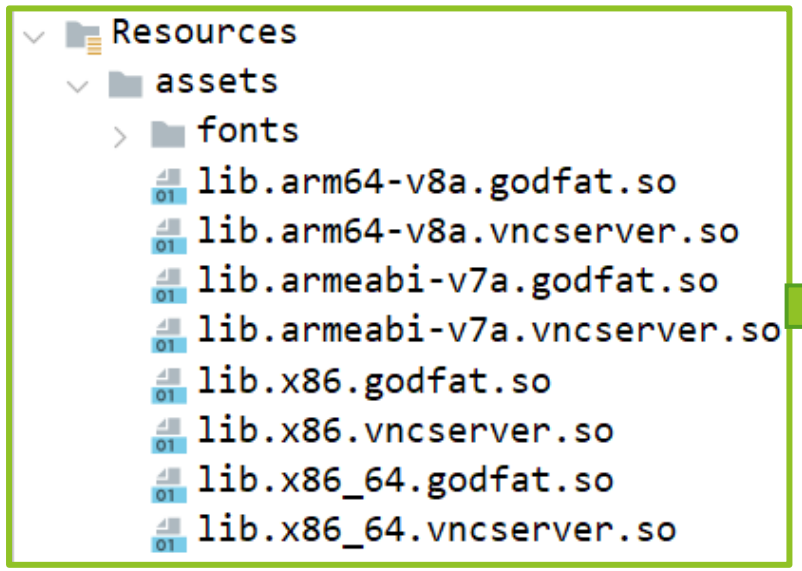


Executable and Linkable Format

libpl_droidsonroids_gif.so

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	7F	45	4C	46	01	01	01	00	00	00	00	00	00	00	00	00	.ELF.....
00000010	03	00	28	00	01	00	00	00	00	00	00	00	34	00	00	00	..(.....4...
00000020	54	A2	00	00	00	02	00	05	34	00	20	00	08	00	28	00	T¢....4. ...(.
00000030	1B	00	1A	00	06	00	00	00	34	00	00	00	34	00	00	00	....4...4...
00000040	34	00	00	00	00	01	00	00	00	01	00	00	04	00	00	00	4.....
00000050	04	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	58	93	00	00	58	93	00	00	05	00	00	00	....X"..X".....
00000070	00	10	00	00	01	00	00	00	F8	9C	00	00	F8	AC	00	00	....øœ..ø¬..
00000080	F8	AC	00	00	18	03	00	00	25	03	00	00	06	00	00	00	ø¬....%.....
00000090	00	10	00	00	02	00	00	00	70	9D	00	00	70	AD	00	00	....p...p...
000000A0	70	AD	00	00	28	01	00	00	28	01	00	00	06	00	00	00	p...(...(.....
000000B0	04	00	00	00	04	00	00	00	34	01	00	00	34	01	00	00	....4...4...
000000C0	34	01	00	00	BC	00	00	00	BC	00	00	00	04	00	00	00	4...¼...¼.....
000000D0	04	00	00	00	51	E5	74	64	00	00	00	00	00	00	00	00	....Qåtd.....
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	06	00	00	00	.....
000000F0	10	00	00	00	01	00	00	70	28	80	00	00	28	80	00	00	.....p(€..(€..

Malicious Encrypted Native Binaries



Encrypted bytes

lib.armeabi-v7a.godfat.so

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	15	75	C4	6A	E5	89	7E	33	F3	FF	DF	4F	01	B7	C9	11	.uÄjå‰~3óÿßO.·É.
00000010	CF	26	E4	0D	18	29	3A	85	9D	96	E6	8C	95	A6	23	92	Ï&ä..):….–æŒ•¦#’
00000020	CF	9A	9F	73	70	C1	1D	F9	10	7B	3F	7E	C5	51	32	E9	ÏšŸspÁ.ù.{?~ÅQ2é
00000030	D4	E0	04	40	DB	24	A8	19	5E	A2	FC	20	C7	91	C3	05	Ôà.@Û$¨.^¢ü Ç‘Ã.
00000040	A8	9D	AE	0C	90	96	DC	CA	A6	5A	EE	8B	45	D6	68	AE	¨.®..–ÜÊ¦Zî‹EÖh®
00000050	7A	D0	37	42	98	C8	9A	EA	76	9A	FD	C6	3E	EC	FB	1E	zÐ7B˜ÈšêvšýÆ>ìû.
00000060	B7	5F	2D	68	91	78	B7	4A	30	B6	72	A8	10	AE	D6	B4	·_-h‘x·J0¶r¨.®Ö´
00000070	FE	E5	B6	96	D2	44	3A	5A	5B	2B	1B	79	73	54	31	A4	þå¶–ÒD:Z[+.ysT1¤
00000080	F2	FC	D8	9C	52	B0	F5	51	99	4C	4C	B9	9A	F7	D8	14	òüØœR°õQ™LL¹š÷Ø.
00000090	98	29	3D	45	0C	F3	E4	41	5C	4D	07	3A	CA	C5	79	3F	˜)=E.óäA\M.:ÊÅy?
000000A0	0A	77	55	D1	DB	6A	E8	45	C1	AE	5F	2B	89	EA	26	44	.wUÑÛjèEÁ®_+‰ê&D
000000B0	C7	BB	24	28	A2	0E	FA	C6	37	86	55	D3	01	88	DE	7F	Ç»$(¢.úÆ7†UÓ.ˆÞ.
000000C0	CC	CD	35	EF	07	59	68	F6	40	12	B5	3F	3B	F1	CC	47	ÌÍ5ï.Yhö@.µ?;ñÌG

Native References in Java

```
private native boolean vncConnectReverse(String host, int port);

private native int vncGetFramebufferHeight();

private native int vncGetFramebufferWidth();

private native boolean vncNewFramebuffer(int width, int height);

private native boolean vncStartServer(int width, int height, int port, String desktopname, String password);

private native boolean vncStopServer();

private native boolean vncUpdateFramebuffer(ByteBuffer buf);

static {
    if ((23 + 6) % 6 <= 0) {
    }
    DecryptAsset.loadEncryptedLibrary(MainService.class, "vncserver");
    DecryptAsset.loadEncryptedLibrary(MainService.class, "godfat");
}
```

System.load()