DEFCON 32 - DEMO LABS

ZIP IT UP SNEAK IT IN

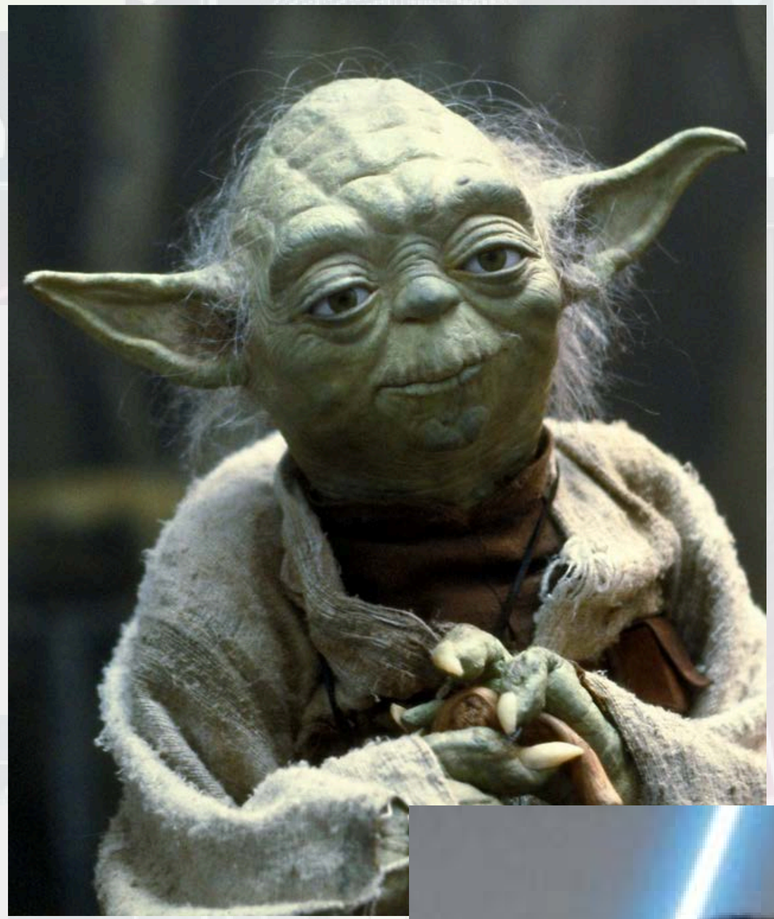
apkInspector

Secura
A BUREAU VERITAS COMPANY

WHO ARE WE ?

Leonidas Vasileiadis
Senior Security Specialist

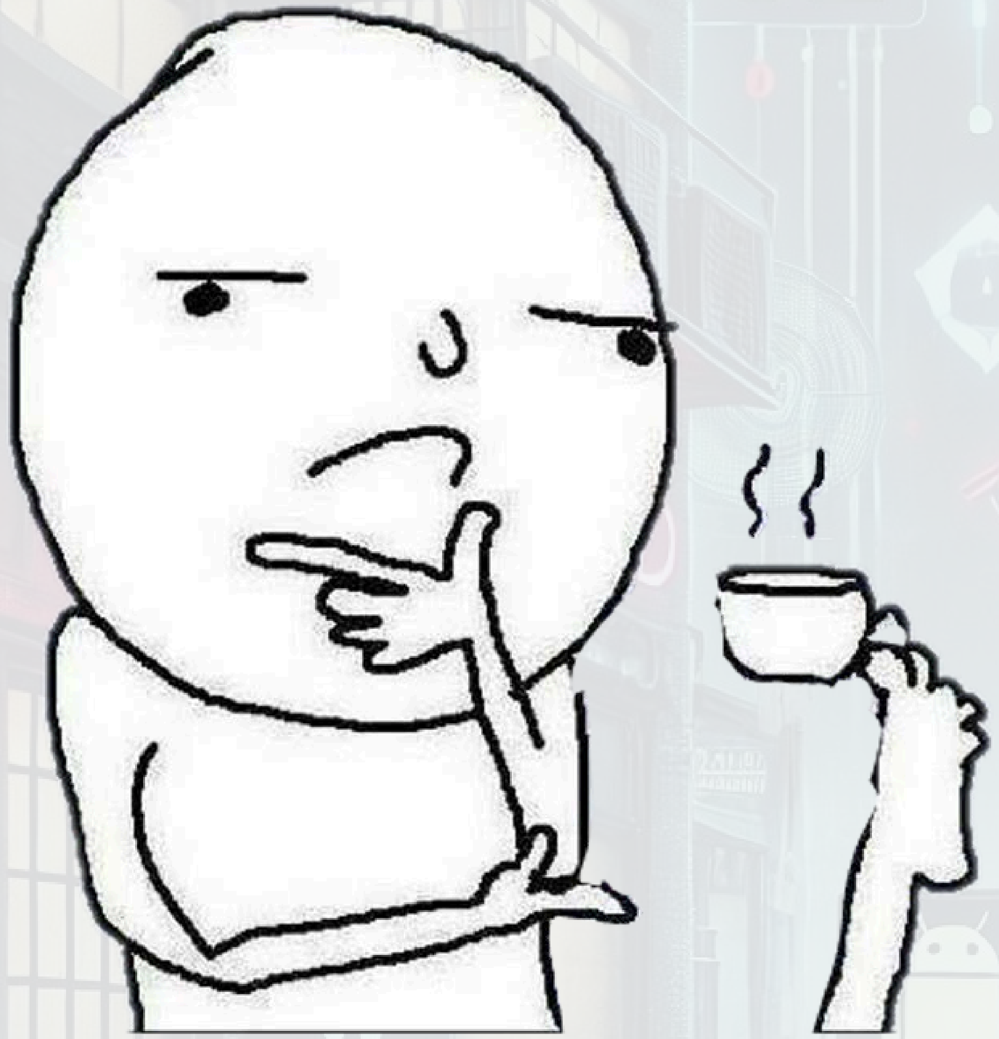
Kaloyan Velikov
Senior Security Specialist

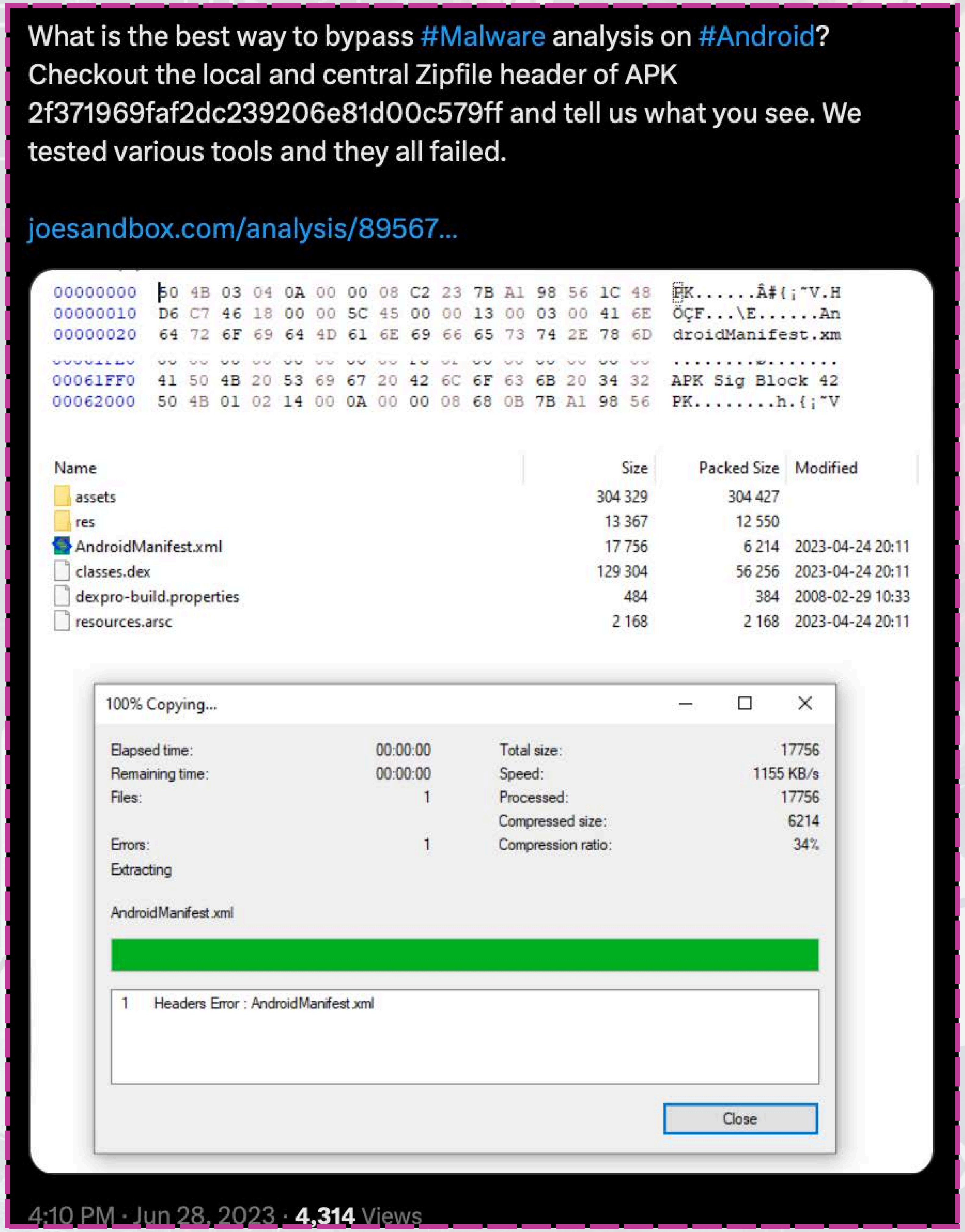


Secura
A BUREAU VERITAS COMPANY

BUREAU VERITAS 1828
BUREAU
VERITAS

HOW IT STARTED





What is the best way to bypass #Malware analysis on #Android? Checkout the local and central Zipfile header of APK 2f371969faf2dc239206e81d00c579ff and tell us what you see. We tested various tools and they all failed.

joesandbox.com/analysis/89567...

```
00000000  50 4B 03 04 0A 00 00 08 C2 23 7B A1 98 56 1C 48  PK......Â#{¡˜V.H
00000010  D6 C7 46 18 00 00 5C 45 00 00 13 00 03 00 41 6E  ÖÇF...\E......An
00000020  64 72 6F 69 64 4D 61 6E 69 66 65 73 74 2E 78 6D  droidManifest.xm
[illegible]
00061FF0  41 50 4B 20 53 69 67 20 42 6C 6F 63 6B 20 34 32  APK Sig Block 42
00062000  50 4B 01 02 14 00 0A 00 00 08 68 0B 7B A1 98 56  PK........h.{¡˜V
```

Name	Size	Packed Size	Modified
assets	304 329	304 427	
res	13 367	12 550	
AndroidManifest.xml	17 756	6 214	2023-04-24 20:11
classes.dex	129 304	56 256	2023-04-24 20:11
dexpro-build.properties	484	384	2008-02-29 10:33
resources.arsc	2 168	2 168	2023-04-24 20:11

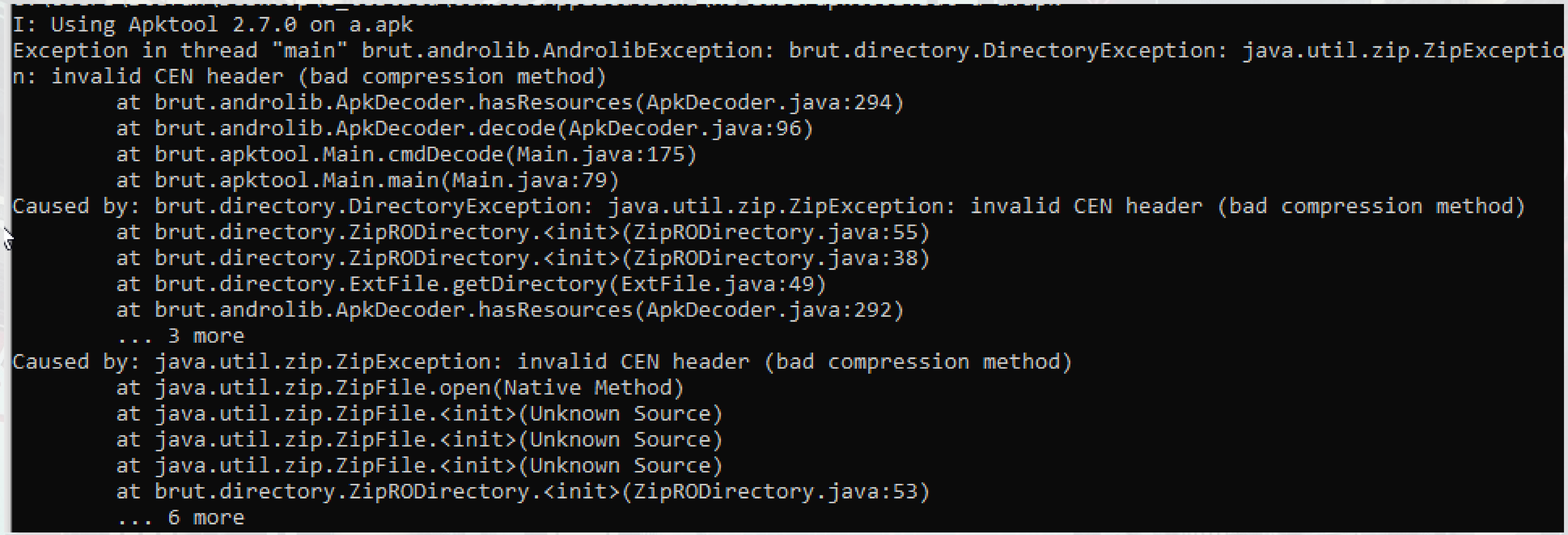
100% Copying...

```
Elapsed time:     00:00:00     Total size:          17756
Remaining time:   00:00:00     Speed:           1155 KB/s
Files:                   1     Processed:           17756
                               Compressed size:      6214
Errors:                  1     Compression ratio:     34%
Extracting

AndroidManifest.xml

1   Headers Error : AndroidManifest.xml
```

Close

4:10 PM · Jun 28, 2023 · 4,314 Views

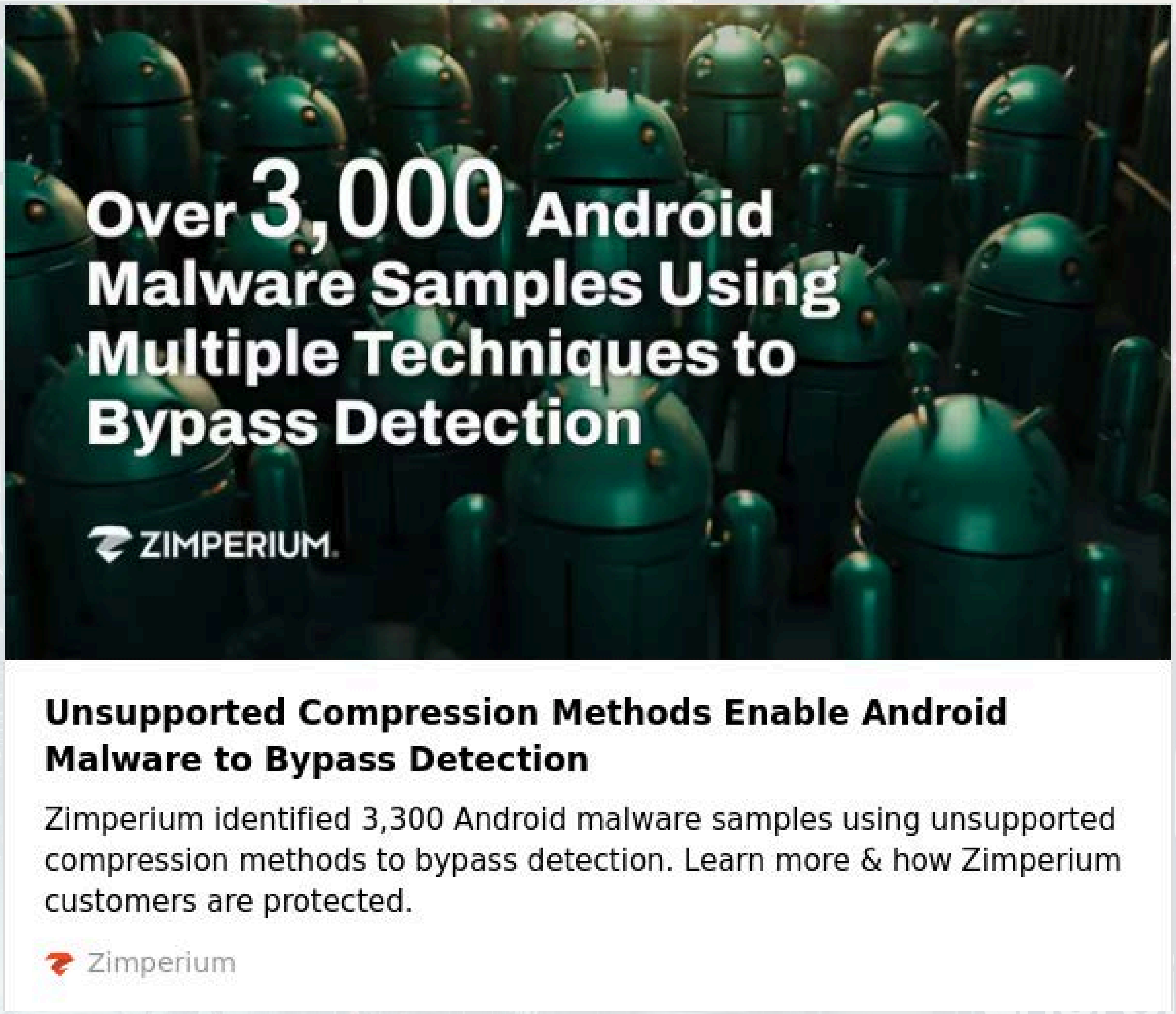
- Consistent errors found in known tools.
- Installable on Android devices.



```
I: Using Apktool 2.7.0 on a.apk
Exception in thread "main" brut.androlib.AndrolibException: brut.directory.DirectoryException: java.util.zip.ZipExceptio
n: invalid CEN header (bad compression method)
        at brut.androlib.ApkDecoder.hasResources(ApkDecoder.java:294)
        at brut.androlib.ApkDecoder.decode(ApkDecoder.java:96)
        at brut.apktool.Main.cmdDecode(Main.java:175)
        at brut.apktool.Main.main(Main.java:79)
Caused by: brut.directory.DirectoryException: java.util.zip.ZipException: invalid CEN header (bad compression method)
        at brut.directory.ZipRODirectory.<init>(ZipRODirectory.java:55)
        at brut.directory.ZipRODirectory.<init>(ZipRODirectory.java:38)
        at brut.directory.ExtFile.getDirectory(ExtFile.java:49)
        at brut.androlib.ApkDecoder.hasResources(ApkDecoder.java:292)
        ... 3 more
Caused by: java.util.zip.ZipException: invalid CEN header (bad compression method)
        at java.util.zip.ZipFile.open(Native Method)
        at java.util.zip.ZipFile.<init>(Unknown Source)
        at java.util.zip.ZipFile.<init>(Unknown Source)
        at java.util.zip.ZipFile.<init>(Unknown Source)
        at brut.directory.ZipRODirectory.<init>(ZipRODirectory.java:53)
        ... 6 more
```

STATS

In August 2023, over 3,000 Android apps were found to have methods to evade detection according to a Zimperium article.

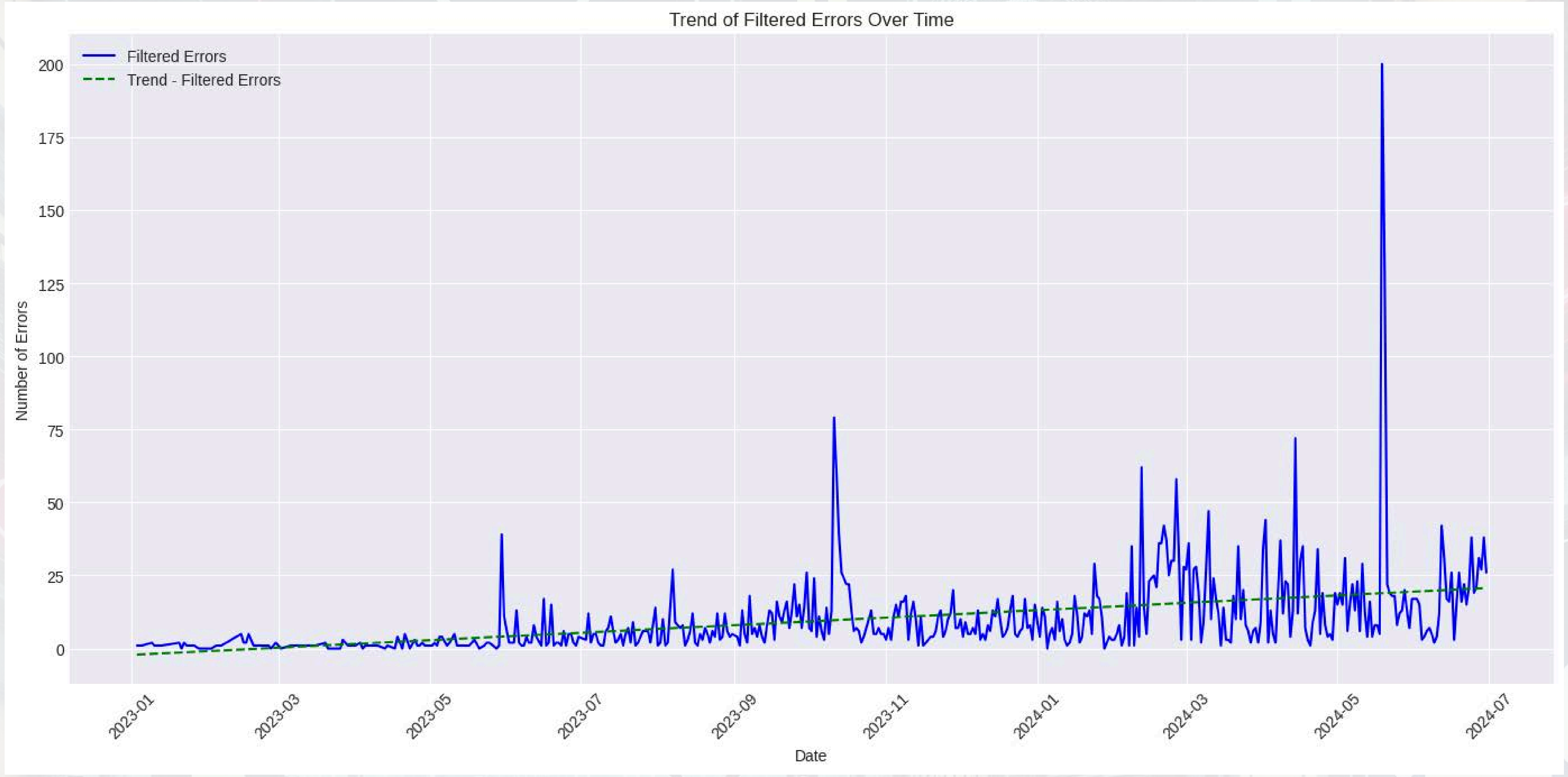

Over 3,000 Android Malware Samples Using Multiple Techniques to Bypass Detection
ZIMPERIUM.

Unsupported Compression Methods Enable Android Malware to Bypass Detection

Zimperium identified 3,300 Android malware samples using unsupported compression methods to bypass detection. Learn more & how Zimperium customers are protected.

Zimperium

Timeframe: Jan 2023 till Jul 2024



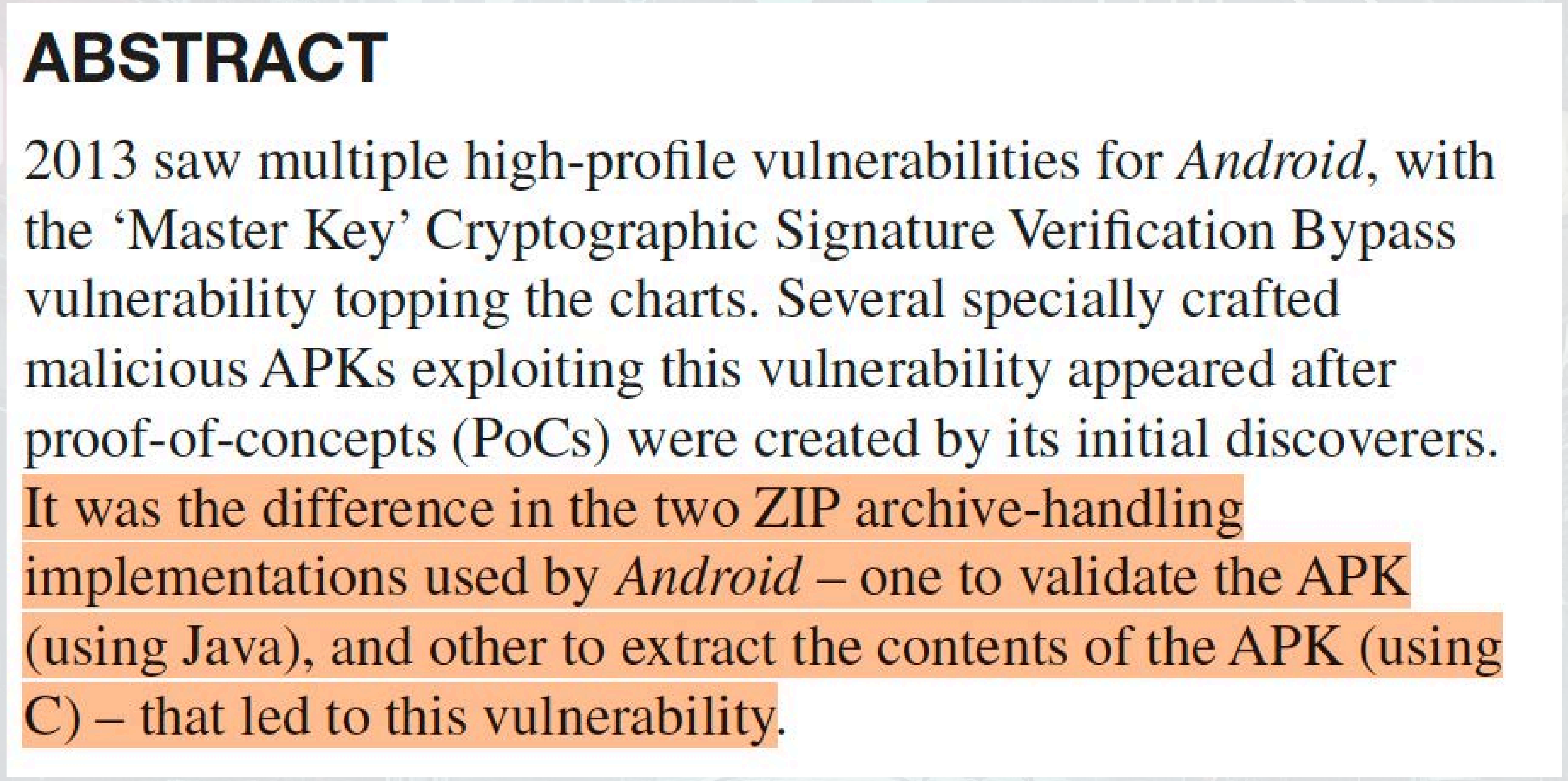
*Data based on filtered APK sample reports from Tria.ge

WHAT WENT WRONG?

DEFINE THE PROBLEM

Is this something new?

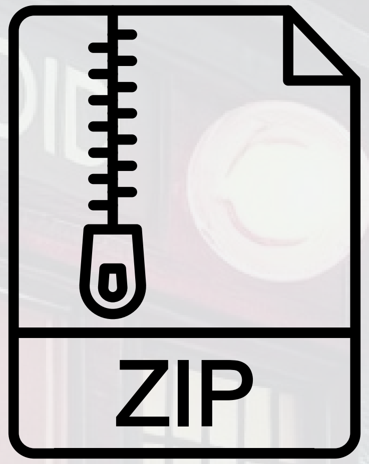
Research Paper of Gregory R. Panakkal:
Leaving our zip undone:how to abuse zip to deliver malware apps*



ABSTRACT

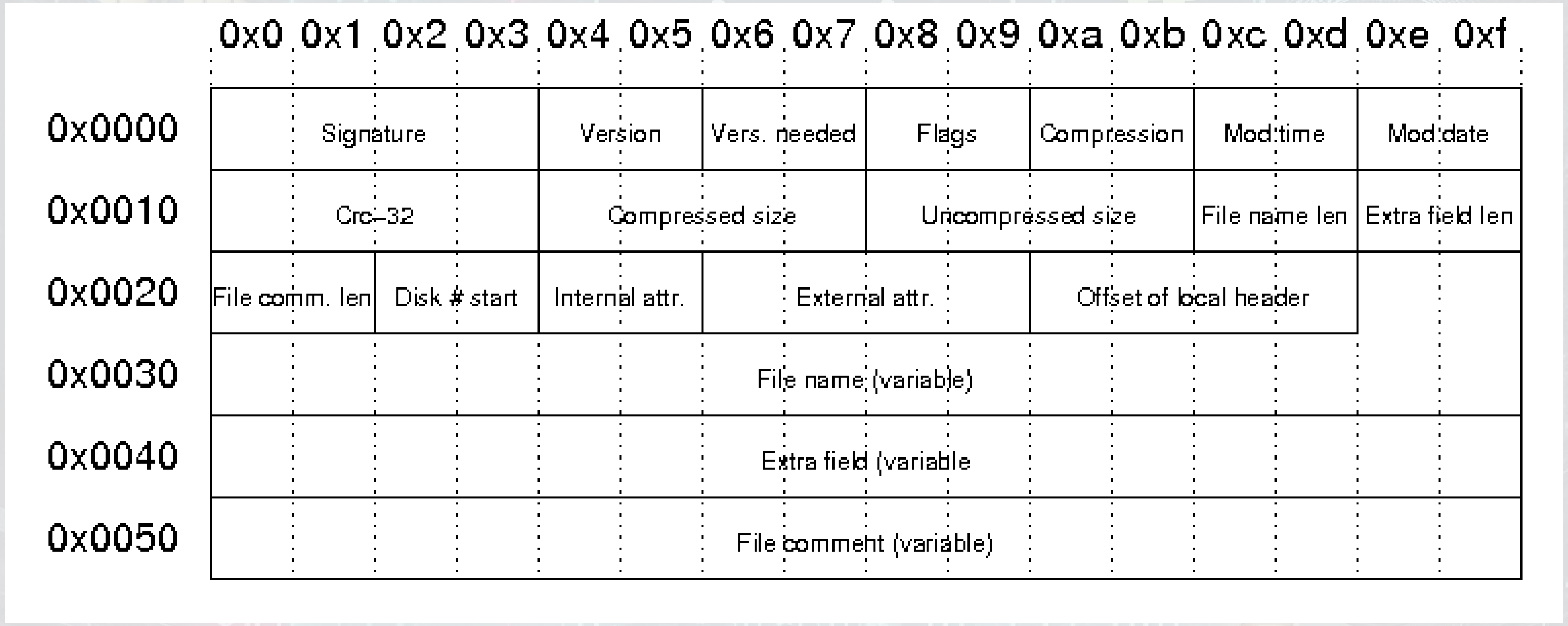
2013 saw multiple high-profile vulnerabilities for *Android*, with the 'Master Key' Cryptographic Signature Verification Bypass vulnerability topping the charts. Several specially crafted malicious APKs exploiting this vulnerability appeared after proof-of-concepts (PoCs) were created by its initial discoverers. It was the difference in the two ZIP archive-handling implementations used by *Android* – one to validate the APK (using Java), and other to extract the contents of the APK (using C) – that led to this vulnerability.

*https://www.virusbulletin.com/uploads/pdf/conference/vb2014/VB2014-Panakkal.pdf

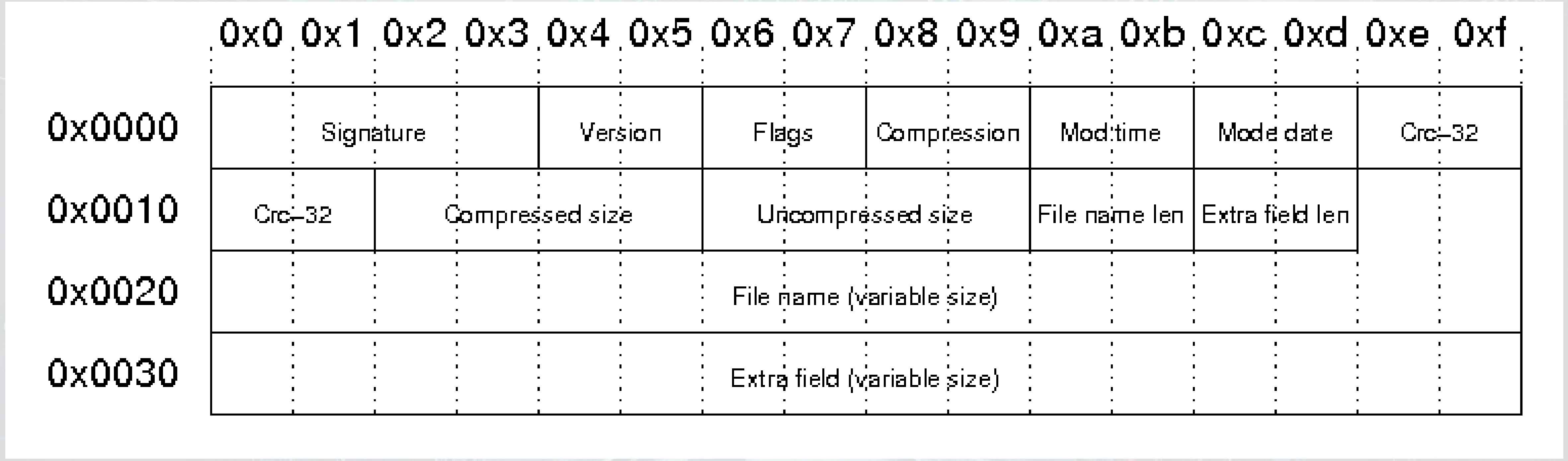


1. Tampered compression methods
2. Zipentry with empty filename
3. Spoofing the Type Identifier
4. Tampered stringCount value
5. Strings surpassing maximum length
6. Invalid data between elements
7. Unexpected attribute size
8. Unexpected attribute names or values
9. Zero size header for namespace end nodes

Tampered compression methods & Zipentry with empty filename

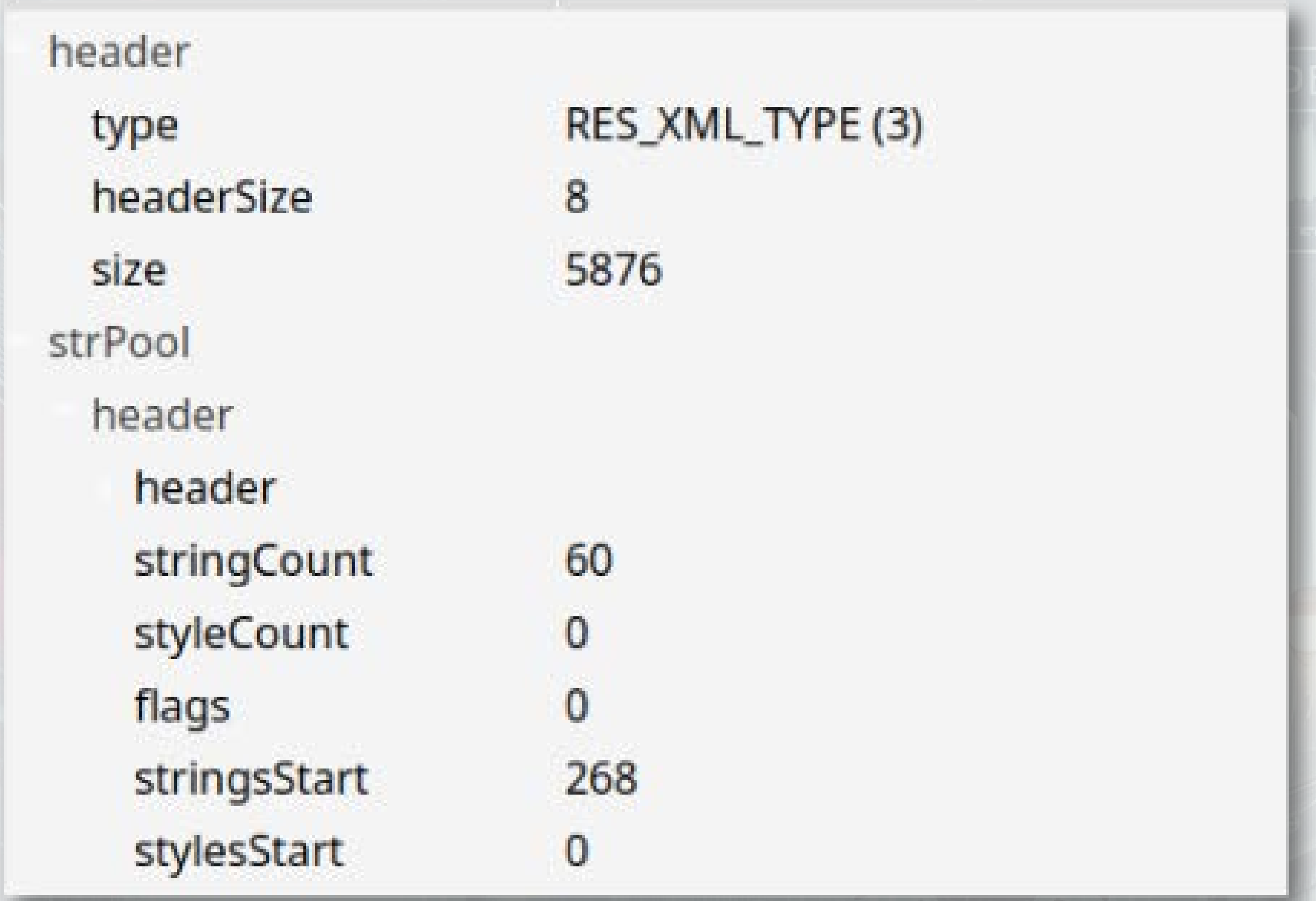


	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x0000	Signature				Version		Vers. needed		Flags		Compression		Mod time		Mod date	
0x0010	Crc-32				Compressed size				Uncompressed size				File name len		Extra field len	
0x0020	File comm. len		Disk # start		Internal attr.		External attr.				Offset of local header					
0x0030	File name (variable)															
0x0040	Extra field (variable															
0x0050	File comment (variable)															



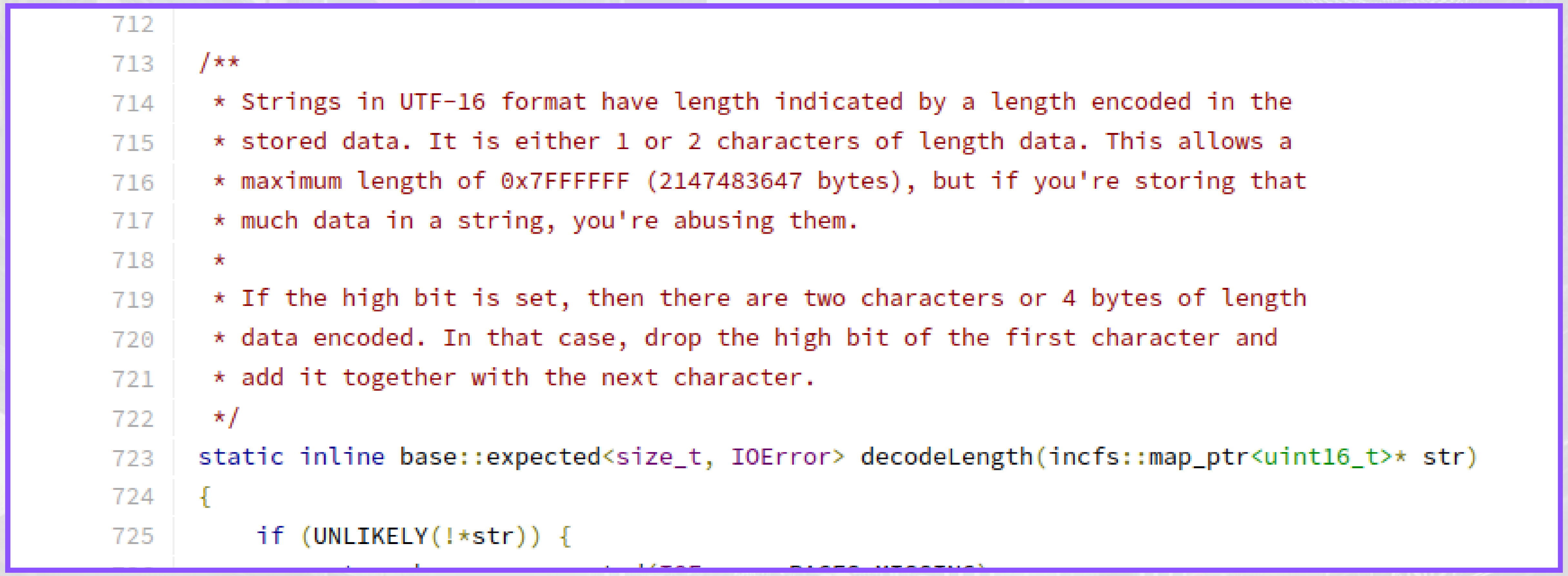
	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xa	0xb	0xc	0xd	0xe	0xf
0x0000	Signature				Version		Flags		Compression		Mod time		Mode date		Crc-32	
0x0010	Crc-32		Compressed size				Uncompressed size				File name len		Extra field len			
0x0020	File name (variable size)															
0x0030	Extra field (variable size)															

Spoofing the Type Identifier & Tampered 'stringCount' value



```
header
  type              RES_XML_TYPE (3)
  headerSize        8
  size              5876
strPool
  header
    header
    stringCount     60
    styleCount      0
    flags           0
    stringsStart    268
    stylesStart     0
```

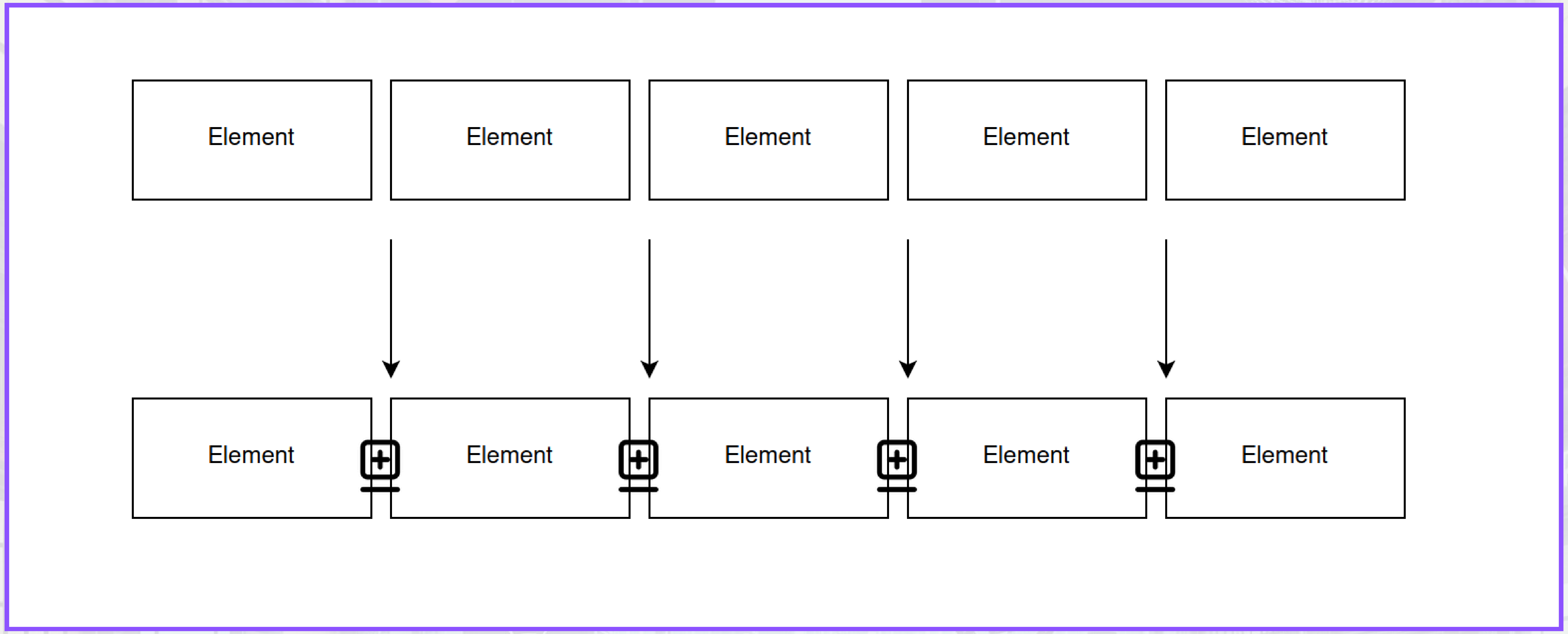
Strings surpassing maximum length



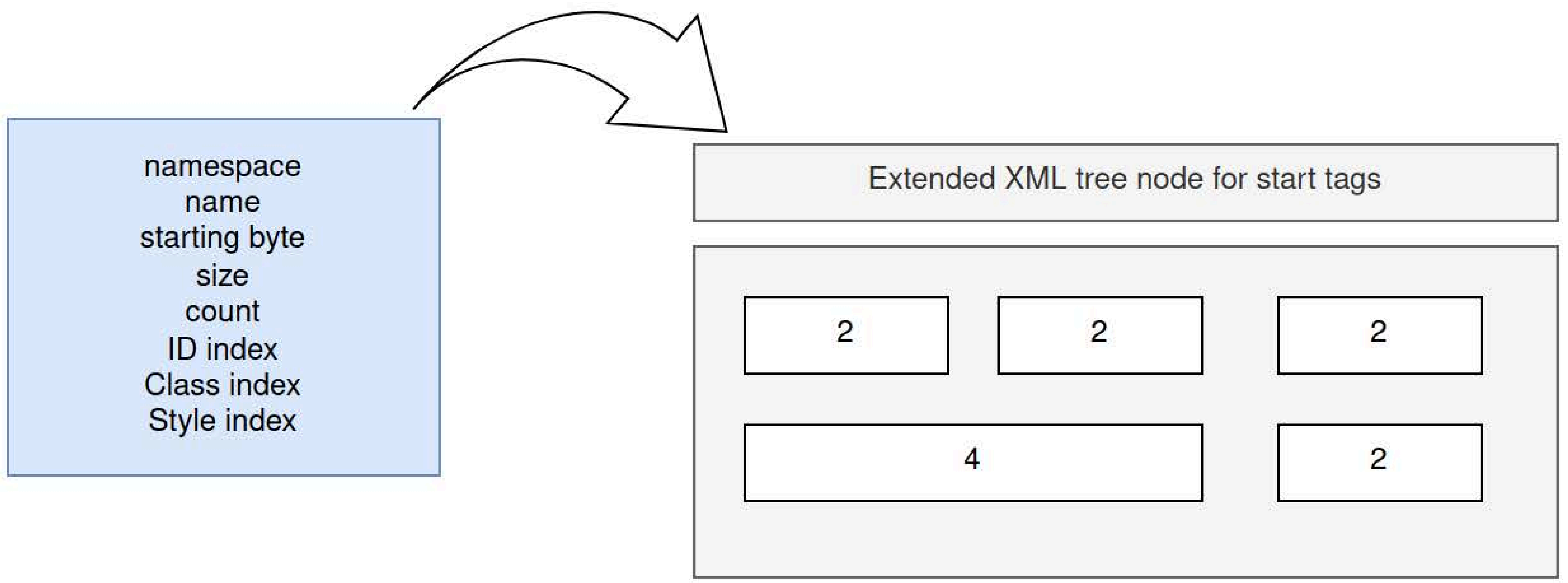
```
/**
 * Strings in UTF-16 format have length indicated by a length encoded in the
 * stored data. It is either 1 or 2 characters of length data. This allows a
 * maximum length of 0x7FFFFFF (2147483647 bytes), but if you're storing that
 * much data in a string, you're abusing them.
 *
 * If the high bit is set, then there are two characters or 4 bytes of length
 * data encoded. In that case, drop the high bit of the first character and
 * add it together with the next character.
 */
static inline base::expected<size_t, IOError> decodeLength(incfs::map_ptr<uint16_t>* str)
{
    if (UNLIKELY(!*str)) {
```

https://android.googlesource.com/platform/frameworks/base/+/refs/heads/android14-release/libs/androidfw/ResourceTypes.cpp

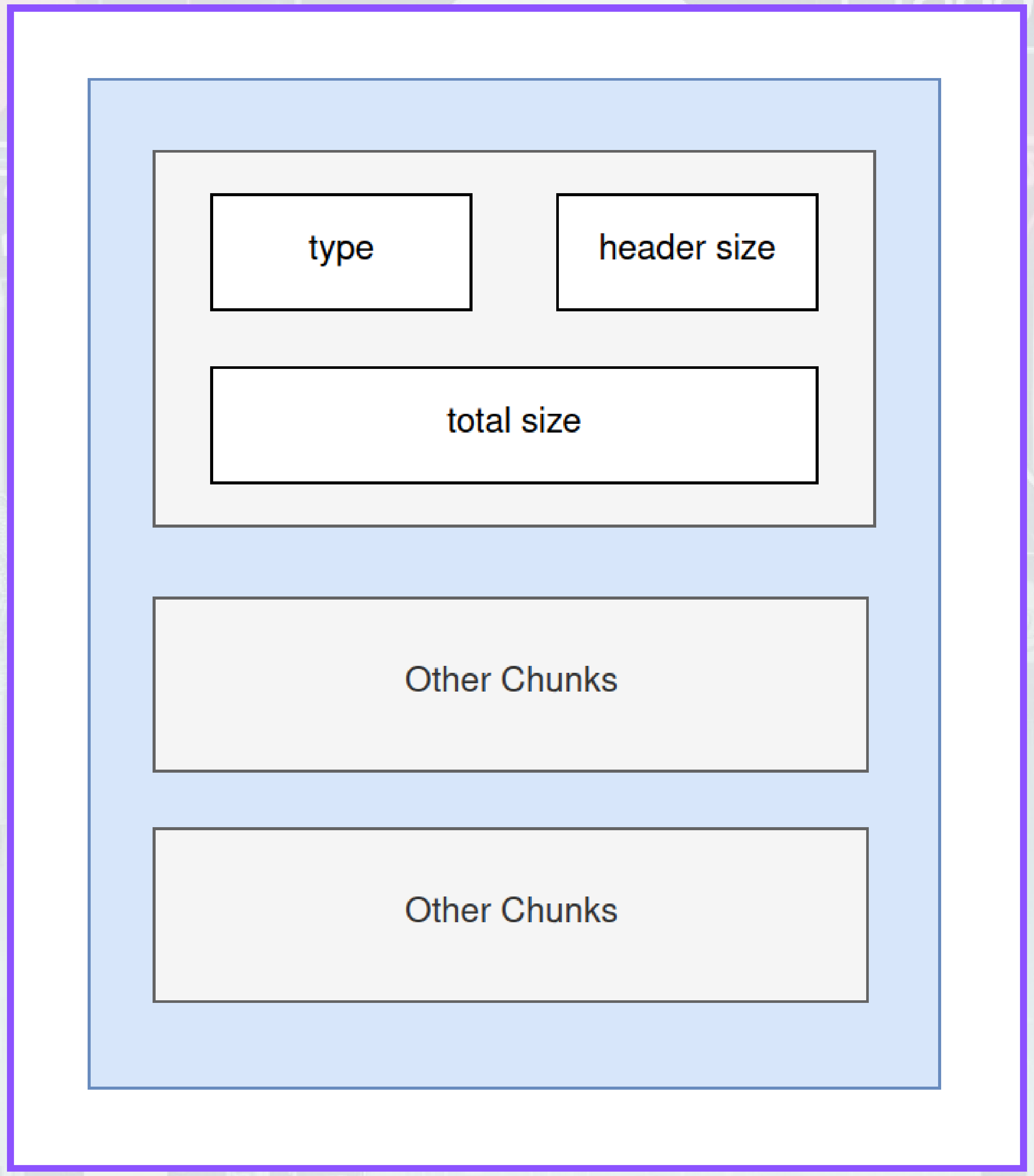
Invalid data between elements



Unexpected attribute size & Unexpected attribute names or values



Zero size header for namespace end nodes



DEFCON 32 - DEMO LABS

MALWARE APK SAMPLES

HOW DID THE ORIGINAL MALWARE BEHAVE?

ORIGINAL MALWARE PACKAGE: **WYIJA.UTYKUVR.UWPEXGH**

EST. DATE: **06-2023**

MD5: **2F371969FAF2DC239206E81D00C579FF**

SHA-1: **0AD5289C6B7A438E3970149B183E74B89F534109**

SHA-256: **B3561BF581721C84FD92501E2D0886B284E8FA8E7DC193E41AB300A063DFE5F3**

So there is no tool to tackle these cases?

DEFCON 32 - DEMO LABS

APKINSPECTOR IN ACTION

AVAILABLE ON PYPI

CLI & LIBRARY

APACHE 2.0 LICENSE

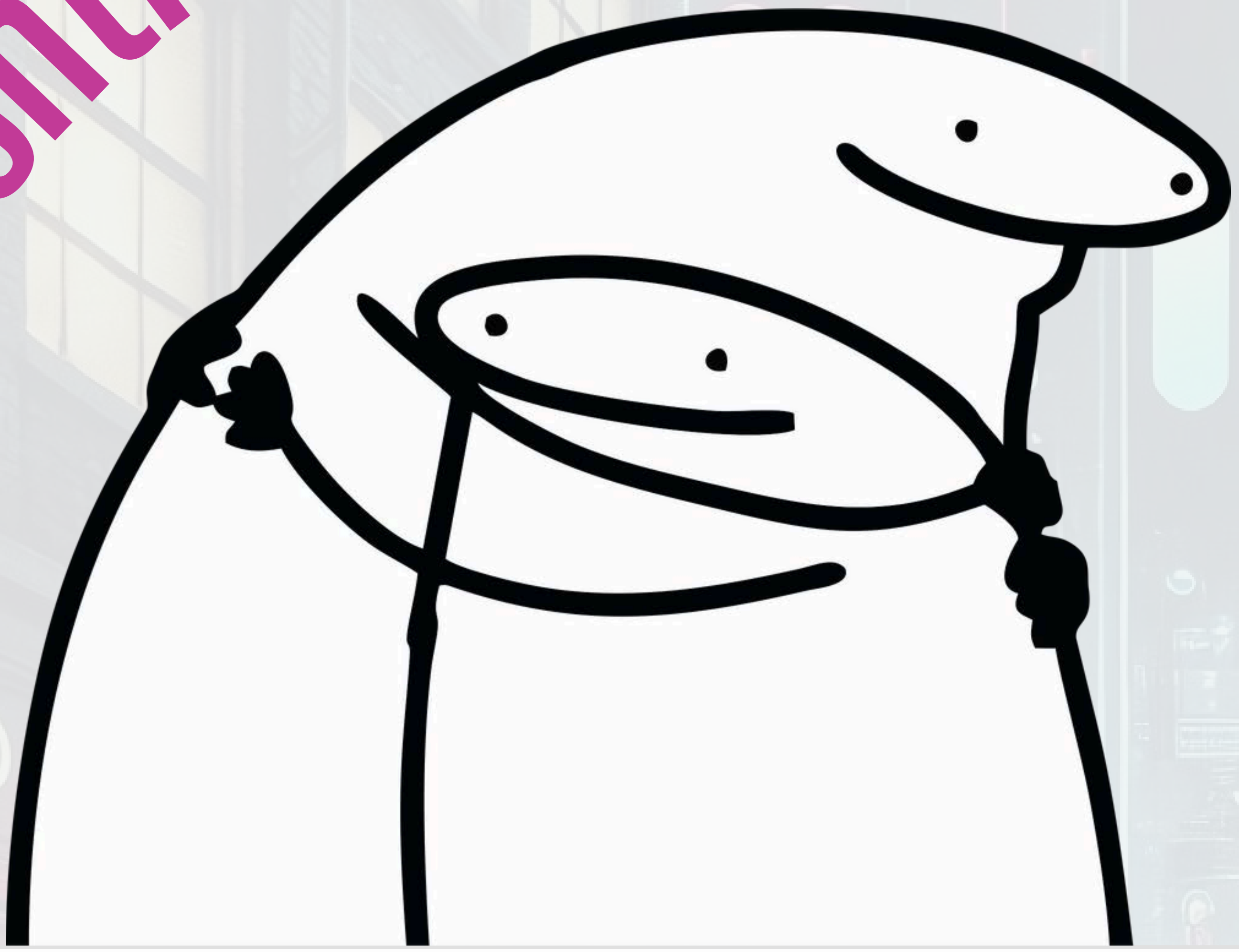
NO DEPENDENCIES

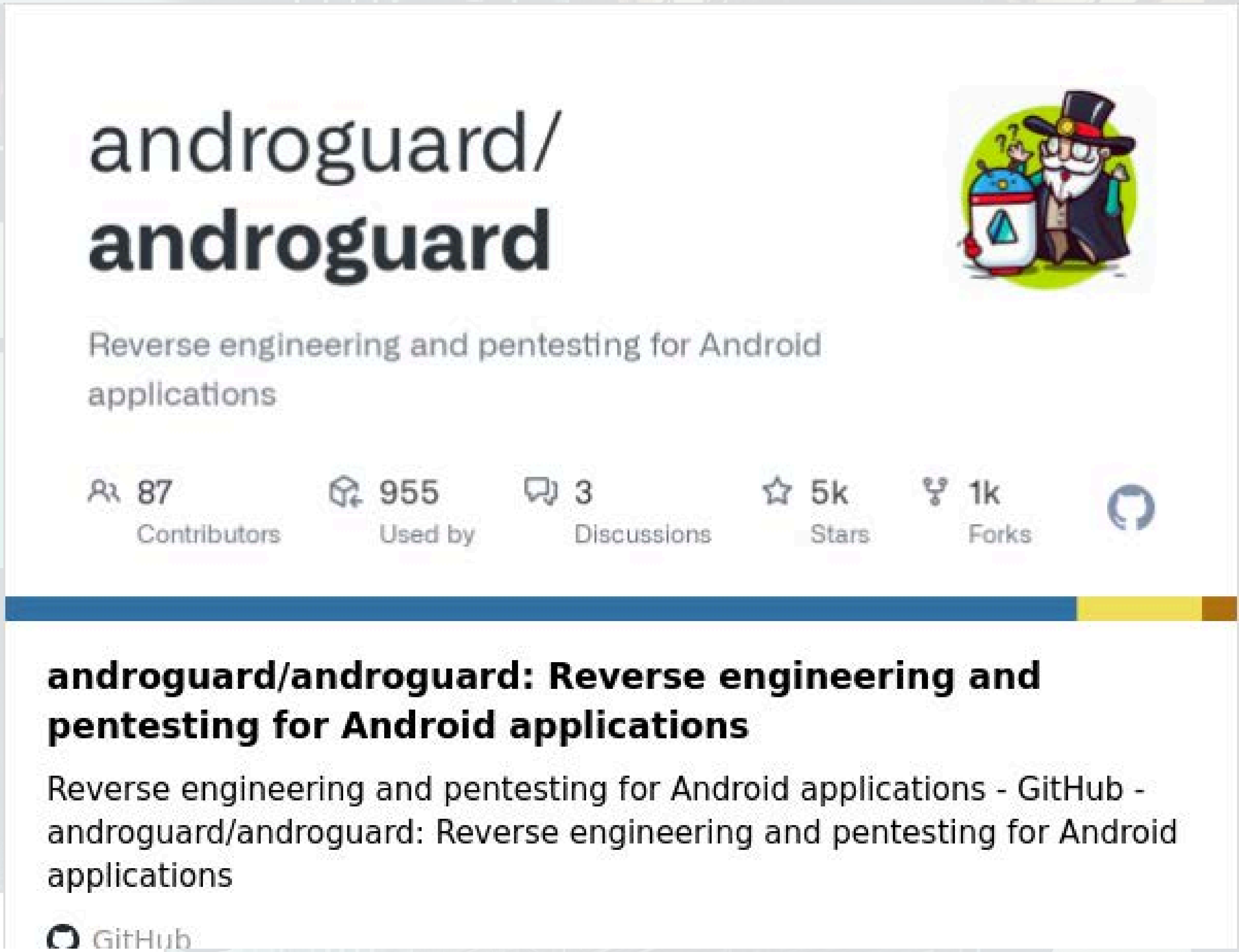
erev0s/apkInspector: apkInspector is a tool designed to provide detailed insights into the zip structure of APK...

apkInspector is a tool designed to provide detailed insights into the zip structure of APK files, offering the capability to extract content and decode the AndroidManifest.xml file. - erev0s/apkIns...

GitHub

We contribute to...





androguard/
androguard

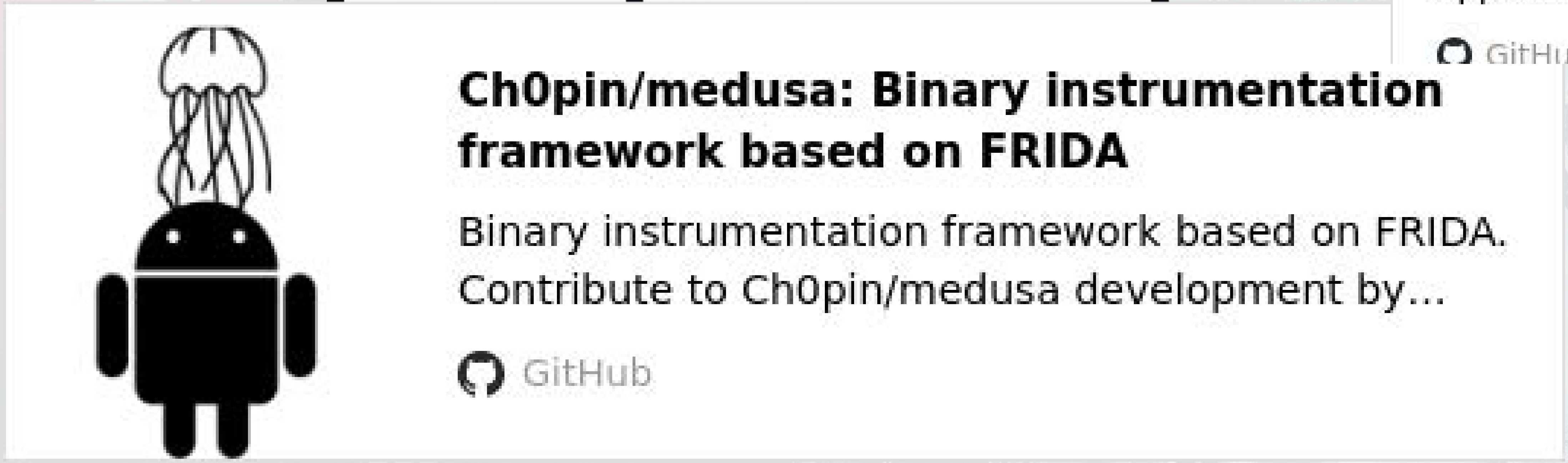
Reverse engineering and pentesting for Android applications

87	955	3	5k	1k
Contributors	Used by	Discussions	Stars	Forks

androguard/androguard: Reverse engineering and pentesting for Android applications

Reverse engineering and pentesting for Android applications - GitHub - androguard/androguard: Reverse engineering and pentesting for Android applications

GitHub



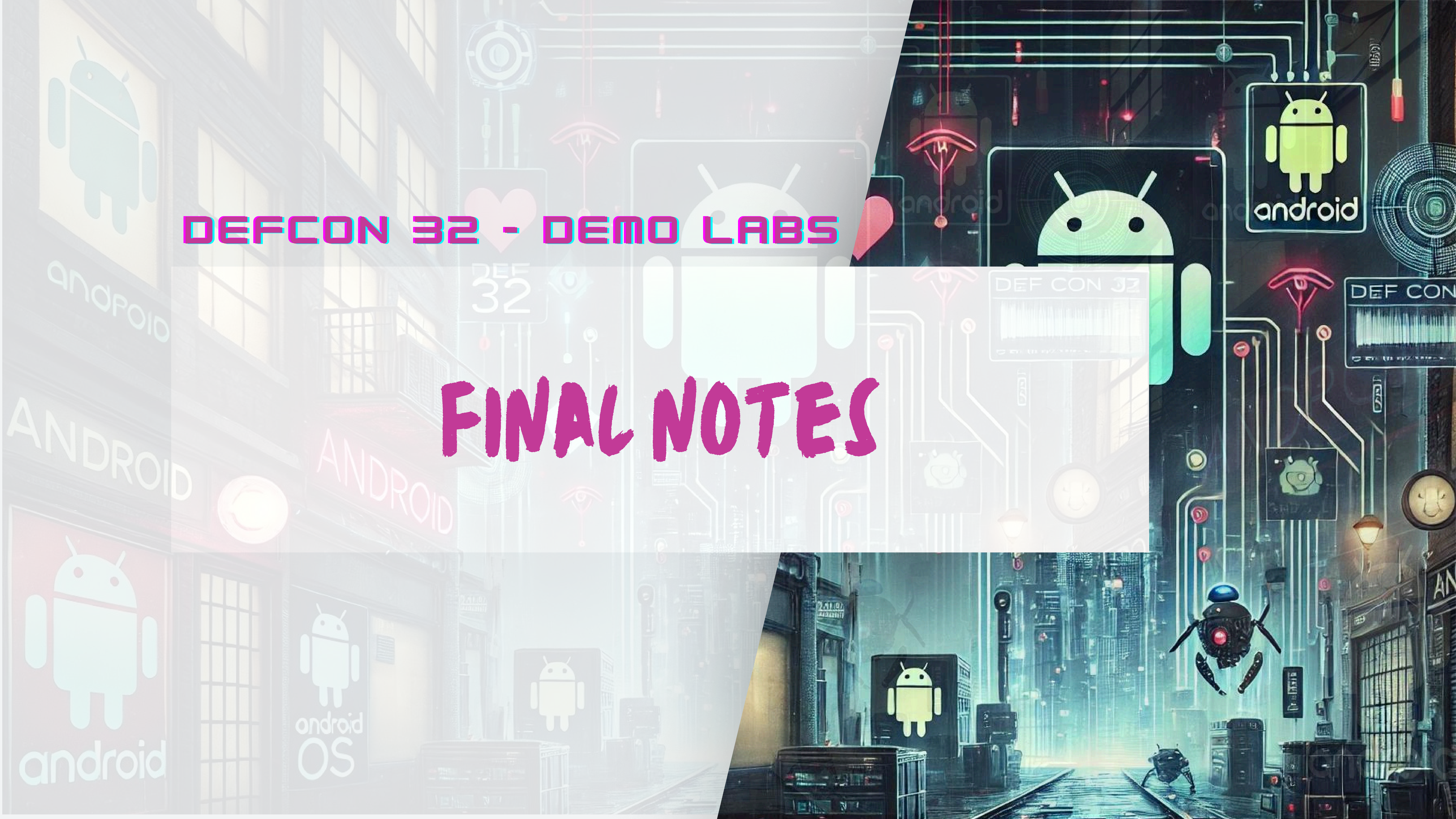
Ch0pin/medusa: Binary instrumentation framework based on FRIDA

Binary instrumentation framework based on FRIDA. Contribute to Ch0pin/medusa development by...

GitHub

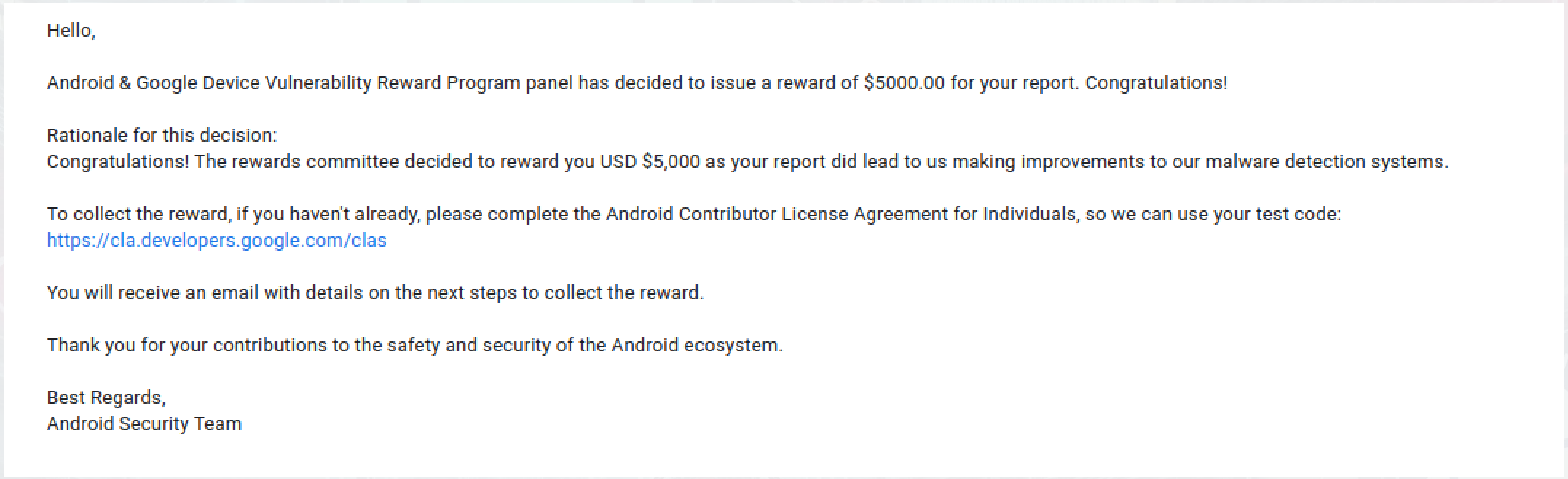
DEFCON 32 - DEMO LABS

FINAL NOTES



FINAL NOTES

Google rewarded us!



Hello,

Android & Google Device Vulnerability Reward Program panel has decided to issue a reward of $5000.00 for your report. Congratulations!

Rationale for this decision:
Congratulations! The rewards committee decided to reward you USD $5,000 as your report did lead to us making improvements to our malware detection systems.

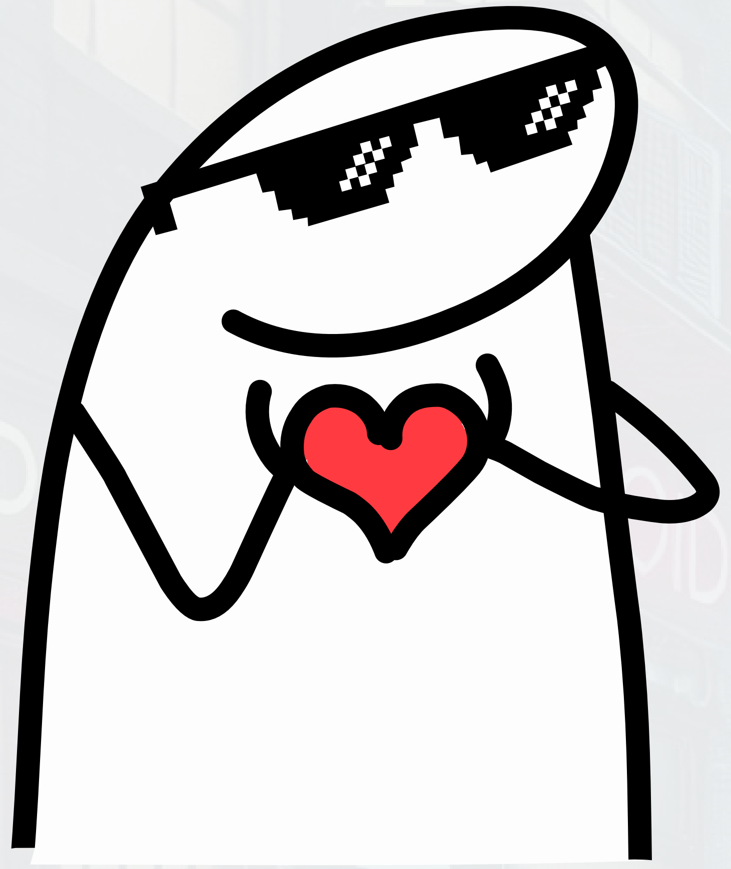
To collect the reward, if you haven't already, please complete the Android Contributor License Agreement for Individuals, so we can use your test code:
https://cla.developers.google.com/clas

You will receive an email with details on the next steps to collect the reward.

Thank you for your contributions to the safety and security of the Android ecosystem.

Best Regards,
Android Security Team

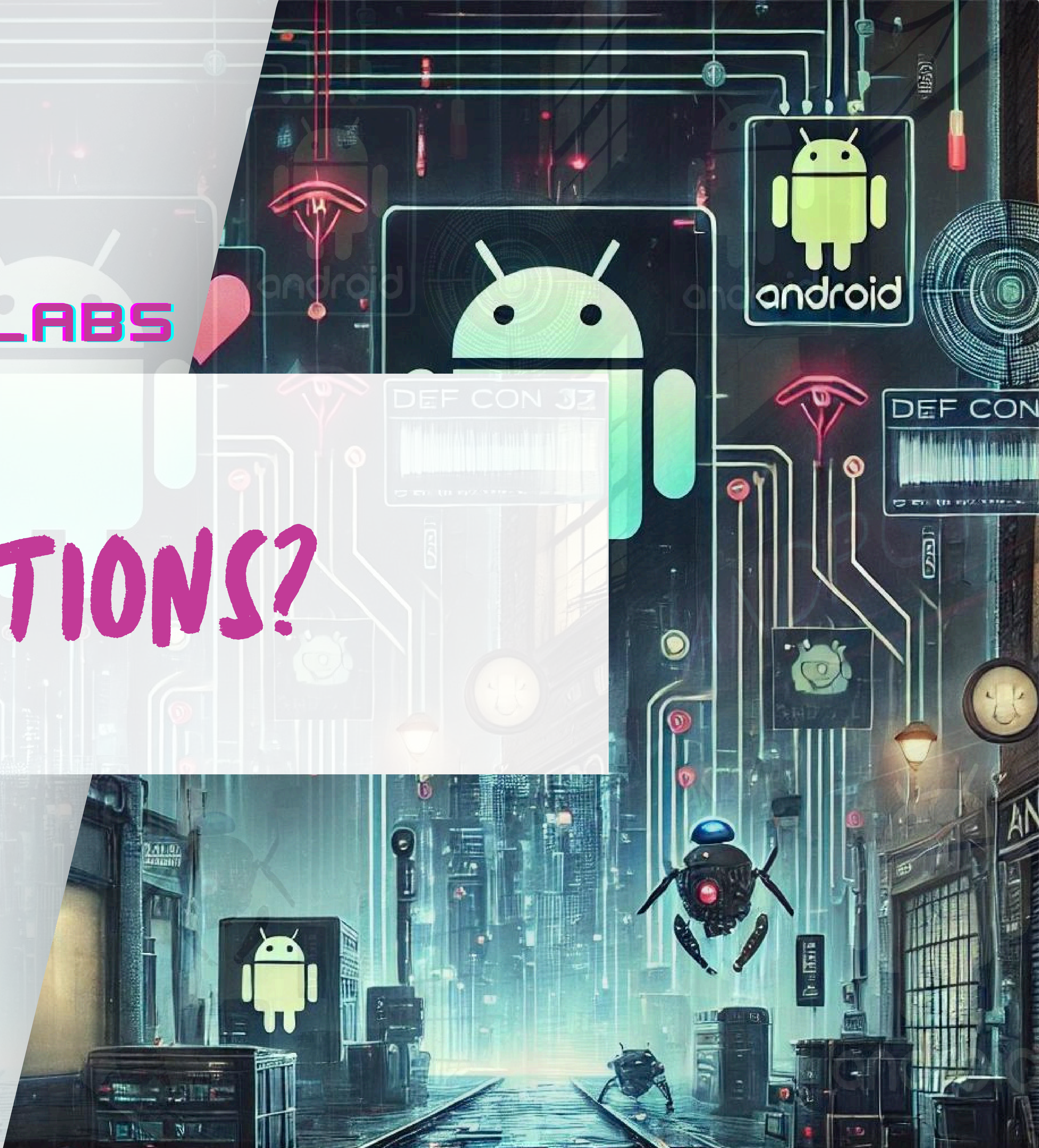
FINAL NOTES



- SHARE IDEAS
- REPORT ISSUES
- MAKE IT KNOWN

DEFCON 32 - DEMO LABS

ANY QUESTIONS?



LinkedIn

https://www.linkedin.com/in/anon/

https://www.linkedin.com/in/kaloyan-velikov/

OTHER PROJECT TO CHECK OUT

VAmPI is a vulnerable API made with Flask and it includes vulnerabilities from the OWASP top 10 vulnerabilities for APIs.