CERTIFIGATE

Front Door Access
to Pwning hundreds of Millions of Androids

Avi Bashan

Ohad Bobrov

Check Point
SOFTWARE TECHNOLOGIES LTD.

AGENDA

- Mobile Threats and Research Motivation
- Mobile Remote Support Tool Overview
- Pwning Mobile Remote Support Tool
- Conclusions
- Q & A

ABOUT US

OHAD BOBROV

- Decade of experience researching and working in the mobile security space
- Former Co Founder & CTO @Lacoon Mobile Security
- Mobile Threat Prevention Area Manager @Check Point
- Presented in BH SP, InfoSec, etc

AVI BASHAN

- Security researcher for over a decade in the PC and mobile areas
- Technical Leader @Check Point
- Former CISO & Security Researcher @Lacoon

MAJOR CONTRIBUTORS

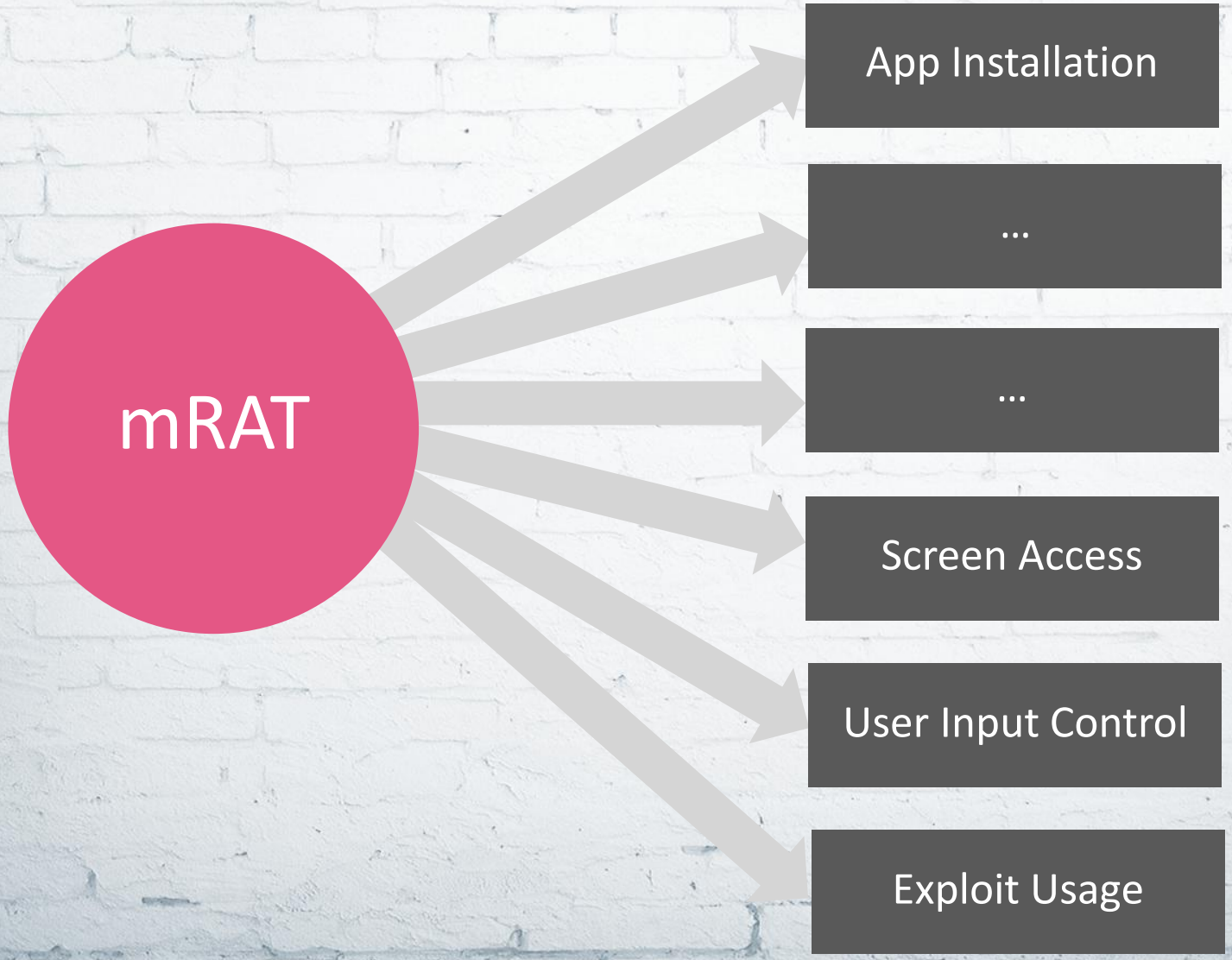
- Pavel Berengoltz
- Daniel Brodie
- Andrey Polkovnichenko
- Denis Voznyuk

MOBILE REMOTE ACCESS TROJAN (mRAT)

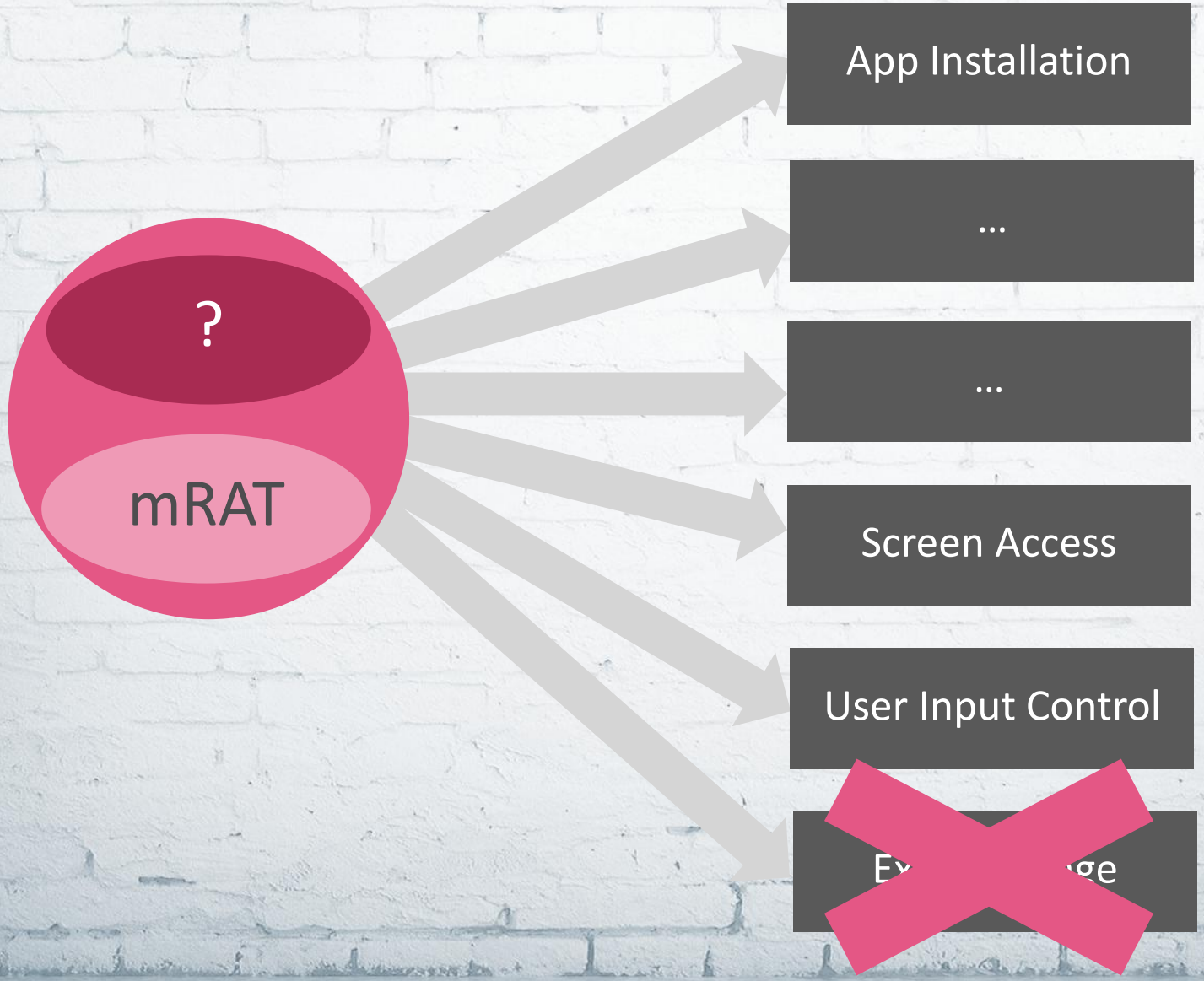
- Used by malicious threat actors
- Provides unauthorized and stealth access to mobile devices
- Known mRATs

mRAT CAPABILITY ANALYSIS

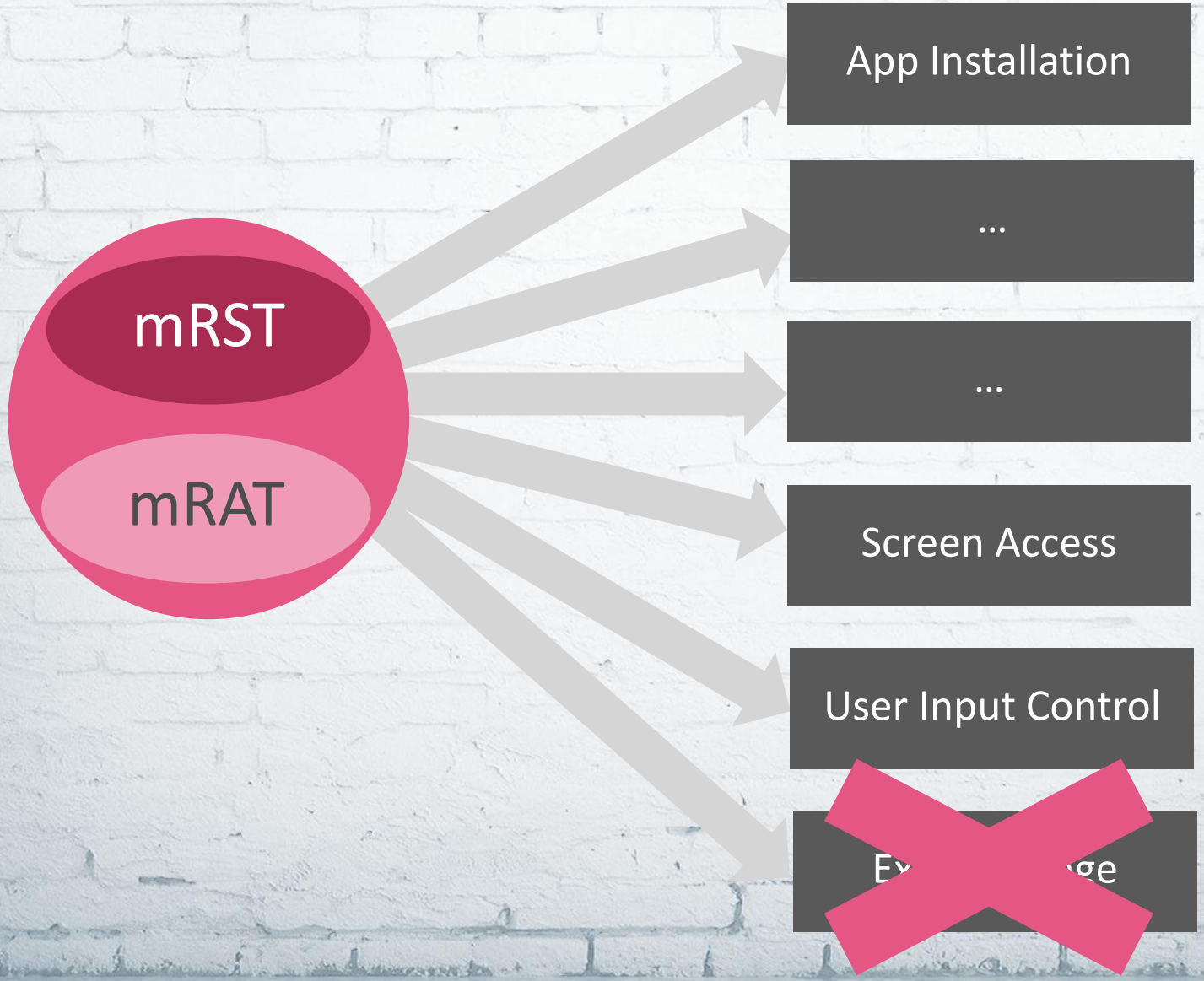
mRAT CAPABILITY ANALYSIS



mRAT CAPABILITY ANALYSIS



mRAT CAPABILITY ANALYSIS



MOBILE REMOTE SUPPORT TOOLS (mRST)

- IT Departments
- Used by Mobile Carriers
- Device Manufacturers

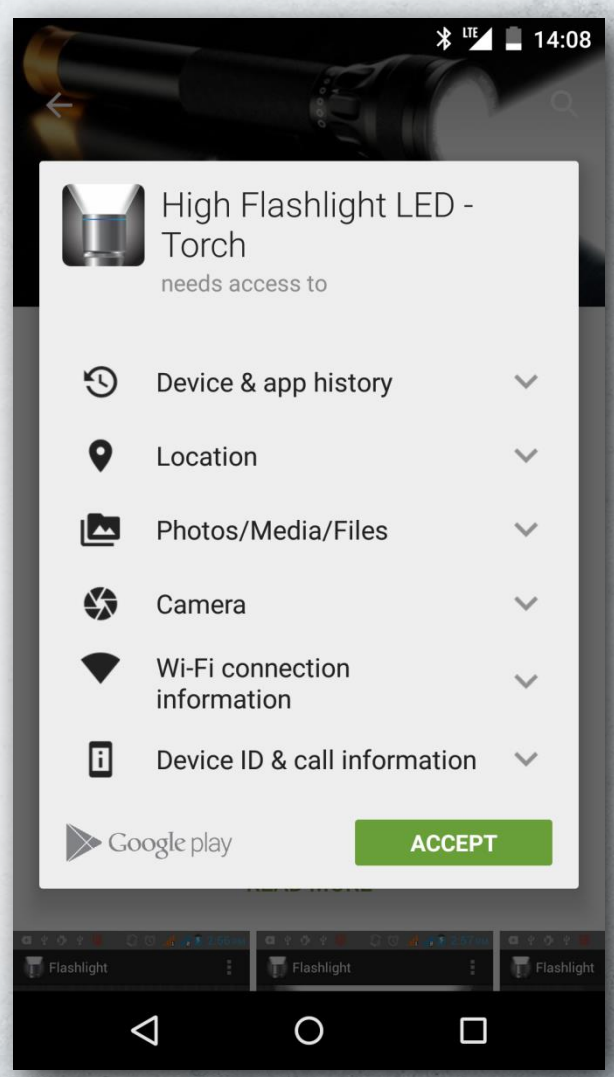
Main Players

MOBILE REMOTE SUPPORT
Tools Overview

ANDROID PERMISSION MODEL 101

ANDROID IS A MODERN OS

- Sandboxing features
- Permission based access
  - Must be obtained to access a resource
  - User can view upon app installation
  - ‘Take it or leave it’ approach



ANDROID PERMISSION MODEL 101

SOME PERMISSIONS are considered “privileged”

Permissions	Action
INSTALL_PACKAGES	App installation
READ_FRAME_BUFFER ACCESS_SURFACE_FLINGER	Screen access
INJECT_EVENTS	User Input Control

GRANTED ONLY TO PRIVILEGED SYSTEMS APPS

- ROM Pre-installed apps located under **`/system/priv-app`**
- Apps signed with the OEM’s certificate

mRST PERMISSIONS

- Access Internet
- Get device network info
- Query installed app list
- Access to device storage

Install apps

Capture screen

User input control

PRIVILEGED PERMISSIONS

ANDROID CUSTOMIZATION CHAIN

AOSP → OEMs → Carriers

LogMeIn

RSUPPORT

TeamViewer

CITRIX®

mRST ARCHITECTURE

MAIN APP

- Signed by mRST developer
- Regular permissions
- Network connection
- User interface

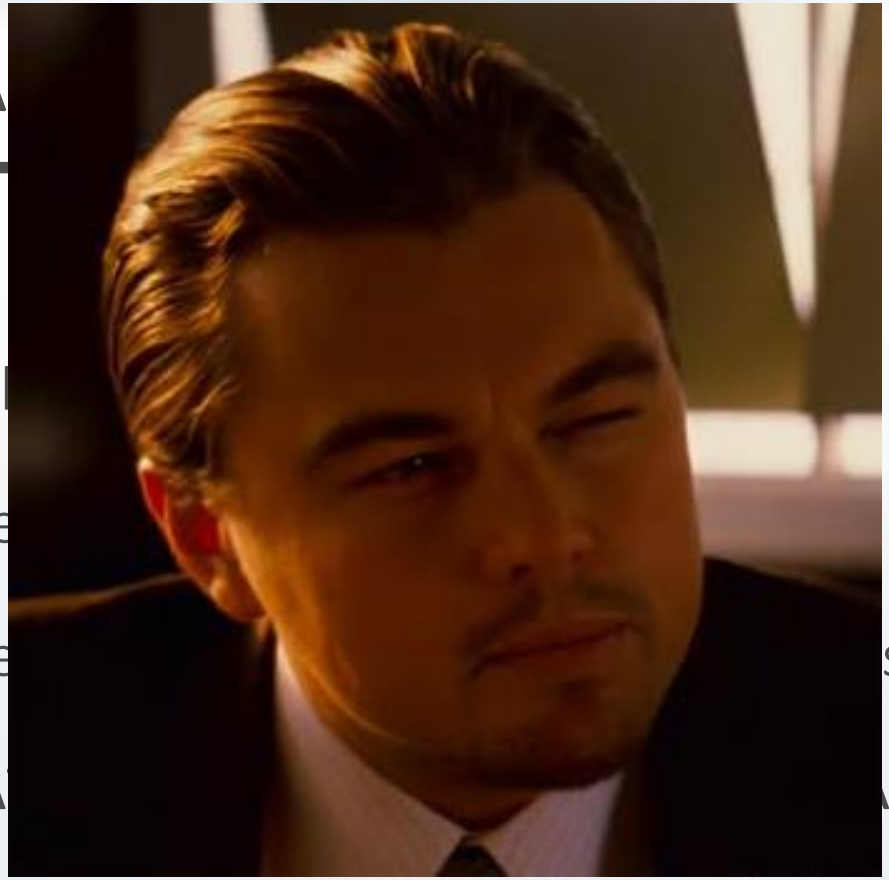
Binder

Verification Mechanism?

PLUGIN

- Signed by OEM
- privileged permissions
- Exported service
- No user interaction

WHA
WIT



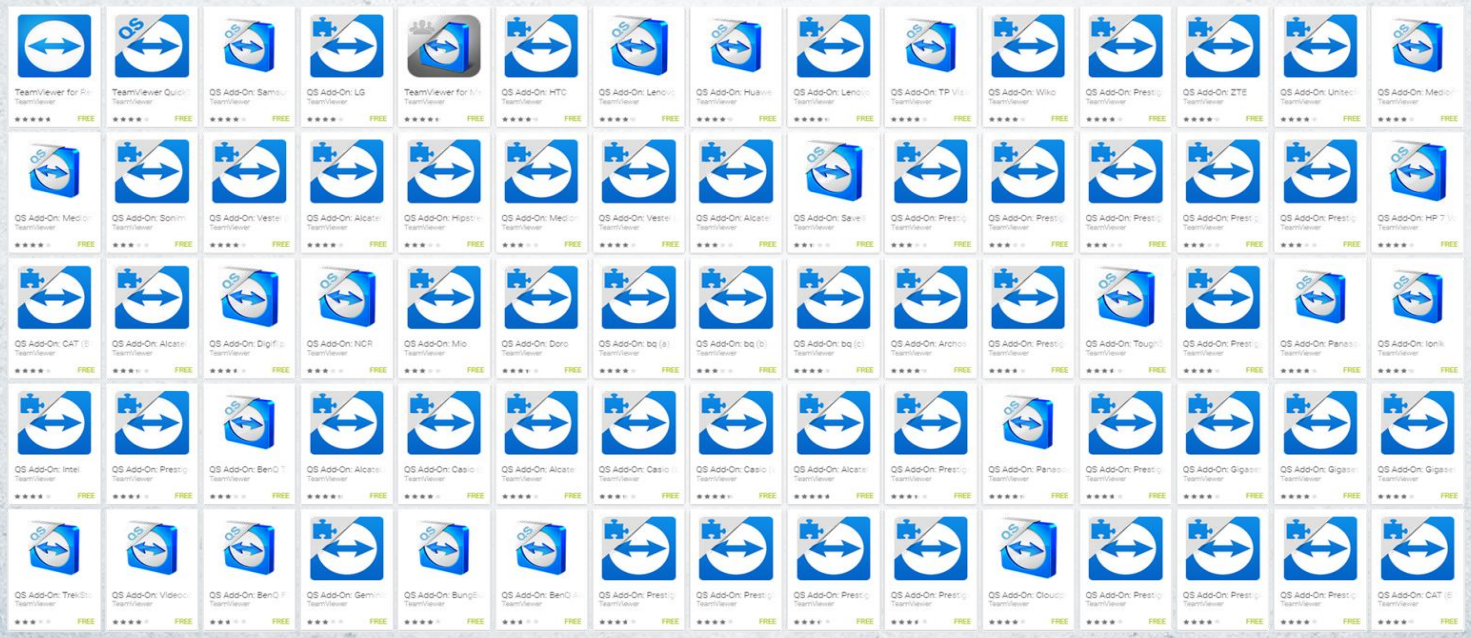
- Signed
- Obtaine
- Designe s
- VALIDA CH VENDOR!

WHAT DID WE FIND?

TEAM VIEWER OVERVIEW

LG htc HUAWEI

lenovo SAMSUNG



TEAM VIEWER'S PLUGIN

- App connects to plugin over Binder
- Plugin needs to verify connection to TeamViewer's main app
- Plugin compares the connecting app's certificate serial number to a hardcoded serial number

WHERE'S WALDO?

```
.method static constructor <clinit>()V
        .registers 2
00000000  new-instance        v0, BigInteger
00000004  const-string        v1, "1287658381"
00000008  invoke-direct       BigInteger-><init>(String)V, v0, v1
0000000E  sput-object         v0, TVAddonService->serialNum_v:BigInteger
00000012  return-void
.end method

.method private checkCallerCertSerialMatch__v(String)Z
        .registers 4
00000000  invoke-virtual      TVAddonService->getApplicationContext()Context, p0
00000006  move-result-object  v0
00000008  invoke-static       certMgr_v->return_caller_cert__v(String, Context)X509Certificate, p1
0000000E  move-result-object  v0
00000010  invoke-virtual      X509Certificate->getSerialNumber()BigInteger, v0
00000016  move-result-object  v0
00000018  sget-object         v1, TVAddonService->serialNum_v:BigInteger
0000001C  invoke-virtual      BigInteger->equals(Object)Z, v0, v1
00000022  move-result         v0
00000024  if-eqz              v0, :2C
:28
00000028  const/4             v0, 1
:2A
0000002A  return              v0
:2C
0000002C  const-string        v0, "TVAddonService"
00000030  const-string        v1, "checkSignature(): serial mismatch - onBind will fail"
00000034  invoke-static       Logging->a(String, String)V, v0, v1
0000003A  const/4             v0, 0
0000003C  goto                :2A
.end method
```

RFC 2459

Internet X.509 Public Key Infrastructure

4.1.2.2 **Serial number**

The serial number is an integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate)

ANDROID APPS SIGNATURE

- Who signs applications on Android?
- Where do they get the certificate?

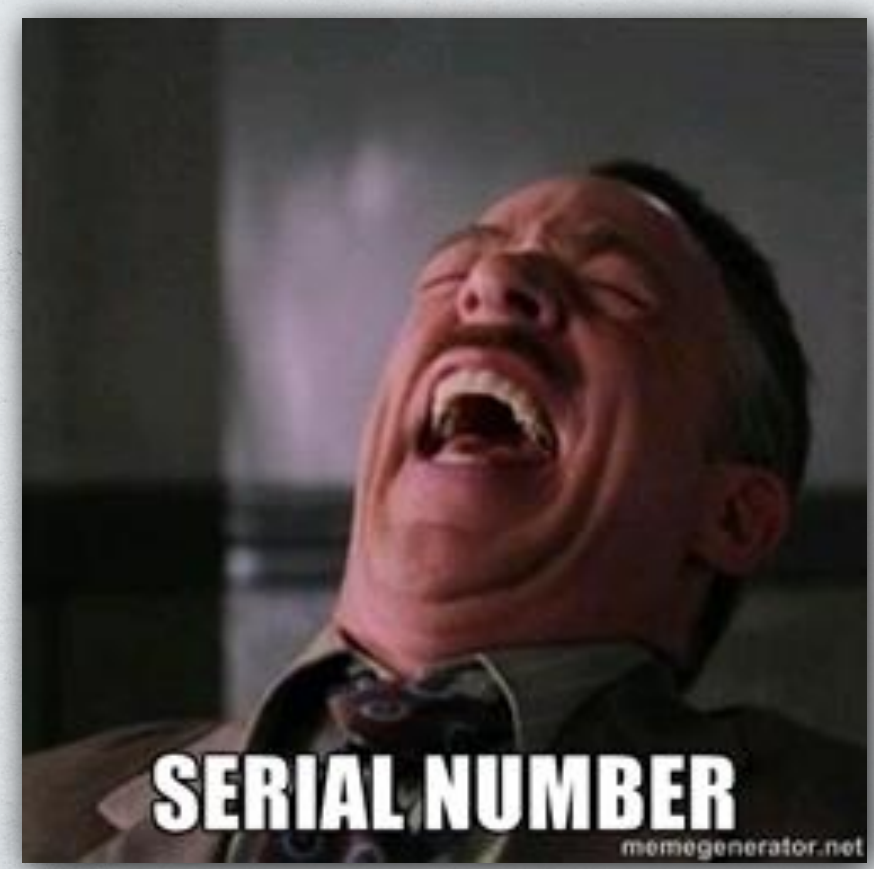
Signing Your Applications

Android requires that all apps be digitally signed with a certificate before they can be installed. Android uses this certificate to identify the author of an app, and the certificate does not need to be signed by a certificate authority. Android apps often use self-signed certificates. The app developer holds the certificate's private key.

- So..

```
avi@avi-laptop /tmp>
openssl req -x509 -nodes -newkey rsa:1024 -keyout evil_key.key
 -out evil_cert.cer -set_serial 1287658381
Generating a 1024 bit RSA private key
.....................................................++++++
.....................................................
.....................................................
.....................................++++++
unable to write 'random state'
writing new private key to 'evil_key.key'
-----
You are about to be asked to enter information that will be in
corporated
```

Pwned!


SERIAL NUMBER
memegenerator.net

DEMO TIME!

RSUPPORT OVERVIEW

Samsung & LG ship the plugin pre-installed

- LG G4, G3, G2 and G Pro 2
- Samsung Galaxy S5 and S4 (Some ROMs)
- And more!

RSupport CODE OVERVIEW

The plugin compares the connecting app's certificate hash code to a hardcoded hash code

```
.method private a(I)Z
          .registers 10
          .param p1, ""
00000000  const/4                 v7, 1
          .prologue
00000002  const/4                 v2, -1
00000004  invoke-virtual          i->getApplicationContext()Context, p0
0000000A  move-result-object      v0
0000000C  invoke-virtual          Context->getPackageManager()PackageManager, v0
00000012  move-result-object      v3
00000014  const/4                 v1, 0
:16
00000016  invoke-virtual          i->getApplicationContext()Context, p0
0000001C  move-result-object      v0
0000001E  invoke-virtual          Context->getPackageName()String, v0
00000024  move-result-object      v1
00000026  const/16                v0, 0x0040
0000002A  invoke-virtual          PackageManager->getPackageInfo(String, I)PackageInfo, v3, v1, v0
00000030  move-result-object      v0
00000032  iget-object             v0, v0, PackageInfo->signatures:[Signature
00000036  const/4                 v4, 0
00000038  aget-object             v0, v0, v4
0000003C  invoke-virtual          Signature->hashCode()I, v0
:42
00000042  move-result             v0
00000044  move                    v2, v0
```

Get the certificate hashCode

RSupport CODE OVERVIEW (Cont.)

```
:E0
000000E0  move-result             v5
000000E2  if-eqz                  v5, :100
:E6
000000E6  const                   v5, 0x0300C78B
000000EC  if-eq                   v4, v5, :FE
:F0
000000F0  const                   v5, 0xE951DACD
000000F6  if-eq                   v4, v5, :FE
:FA
000000FA  if-ne                   v4, v2, :60
:FE
000000FE  return                  v7
:100
00000100  const                   v5, 0x39E0A536
00000106  if-eq                   v4, v5, :FE
:10A
0000010A  if-ne                   v4, v2, :60
:10E
0000010E  goto                    :FE
:110
00000110  move-exception          v0
00000112  const-string            v2, "rsperm"
00000116  new-instance            v3, StringBuilder
0000011A  const-string            v4, "ex4: "
0000011E  invoke-direct           StringBuilder-><init>(String)V, v3, v4
00000124  invoke-virtual          StringBuilder->append(String)StringBuilder, v3, v1
0000012A  move-result-object      v1
0000012C  const-string            v3, ", "
00000130  invoke-virtual          StringBuilder->append(String)StringBuilder, v1, v3
00000136  move-result-object      v1
00000138  invoke-virtual          Exception->toString()String, v0
0000013E  move-result-object      v3
00000140  invoke-virtual          StringBuilder->append(String)StringBuilder, v1, v3
00000146  move-result-object      v1
00000148  invoke-virtual          StringBuilder->toString()String, v1
0000014E  move-result-object      v1
00000150  invoke-static           Log->e(String, String)I, v2, v1
00000156  invoke-virtual          Exception->printStackTrace()V, v0
0000015C  goto                    :68
          .catch Exception {:16 .. :42} :74
          .catch Exception {:46 .. :60} :110
          .catch Exception {:C2 .. :E0} :110
.end method
```

Try to compare it to a few hash codes,
if it's equal - continue

HASHCODE?

- But wait, what is the Signature's hashCode?

MD5? **SHA1**? **SHA256**? **CRC32**???

Android is open source,
so we can just see it's implementation

HASHCODE!

```
@Override
public int hashCode() {
    if (mHaveHashCode) {
        return mHashCode;
    }
    mHashCode = Arrays.hashCode(mSignature);
    mHaveHashCode = true;
    return mHashCode;
}
```

Executes the Arrays.hashCode function on the certificate

```
public static int hashCode(byte[] array) {
    if (array == null) {
        return 0;
    }
    int hashCode = 1;
    for (byte element : array) {
        // the hash code value for byte value is its integer value
        hashCode = 31 * hashCode + element;
    }
    return hashCode;
}
```

32-bit signed integer

Only 2^{32}

$\tilde{=}$ 4 Bilion

Possibilities!


HASHCODE
memegenerator.net

WHAT ELSE?

- We found multiple vulnerable plugins
- We didn't check them all
  Left as an exercise for the reader
- Verification flaw is not limited to mRSTs

mRST PLUGIN
ANOTHER ANGLE

- Found a problem in one of the vendor's main app
- Allowed us to manipulate the main app logic, in order to take control of the OEM signed plugin

COMMUNITAKE VULNERABILITY

Main app allows changing settings by SMS

One of the commands can modify the subdomain of the CnC server
<xxx>.communitake.com

The subdomain can be altered without requiring authentication

The app does not sanitize the subdomain properly
Enables the addition of the '/' character to the subdomain

COMMUNITAKE VULNERABILITY (CONT.)

- An attacker can send a command which changes the CnC server to a malicious CNC server
- Enabling them to take full control of the device with a single SMS message without user intervention!

DEMO TIME!

VULNERABILITIES DISCLOSURE TIMELINE

MID APRIL

Reported to Vendors, OEMs, Google

MID APRIL – MAY

Got responses from most of the vendors, which started to work on resolving the issues

MAY – JUNE

New version of the plugins were uploaded to the Play Store

AUGUST

Still waiting for some vendors responses..

CONCLUSION

Android's eco-system is flawed

- Google moved the responsibility to the OEMs
- No way to patch it

Hundred of millions of Android devices are vulnerable

SO WHAT SHOULD I DO?

- Check if you device is on the list of vulnerable OEMs
  Can be found in our blog post
- Check if you have one of the plugins installed
  Remove it (If you can)

A LAYERED MOBILE SECURITY APPROACH

VULNERABILITY ASSESSMENT

- System, OEM and 3rd party apps, and plugins
- Continues monitoring

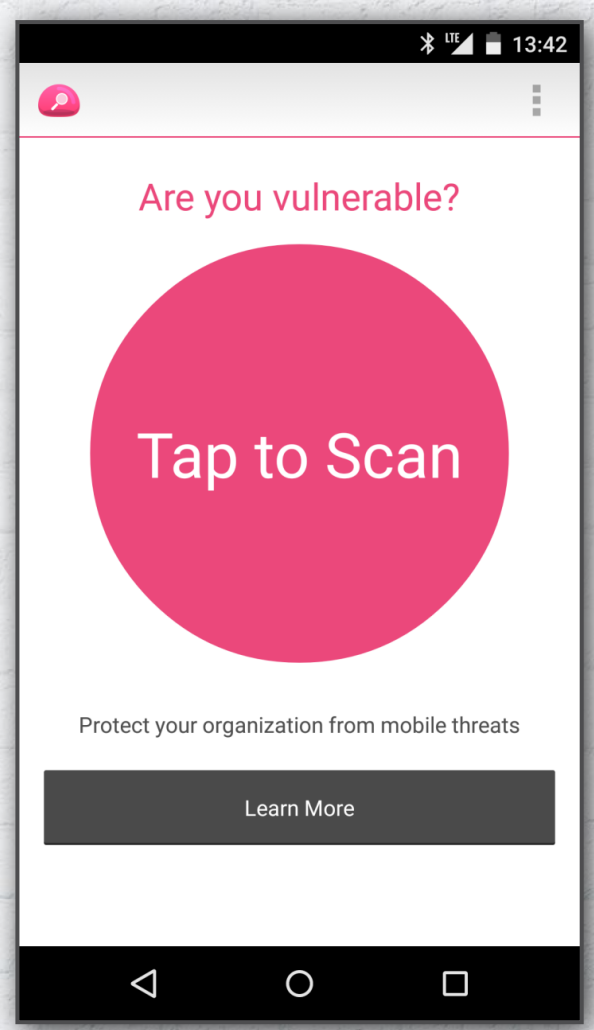
THREAT DETECTION

- Horizontal escalation from 3rd party apps

RISK MITIGATION

- Alert user to remove vulnerable plugins
- Track patching progress

CERTIFI-GATE SCANNER



Google Play

QUESTIONS?