LEVERAGING ANDROID PERMISSIONS

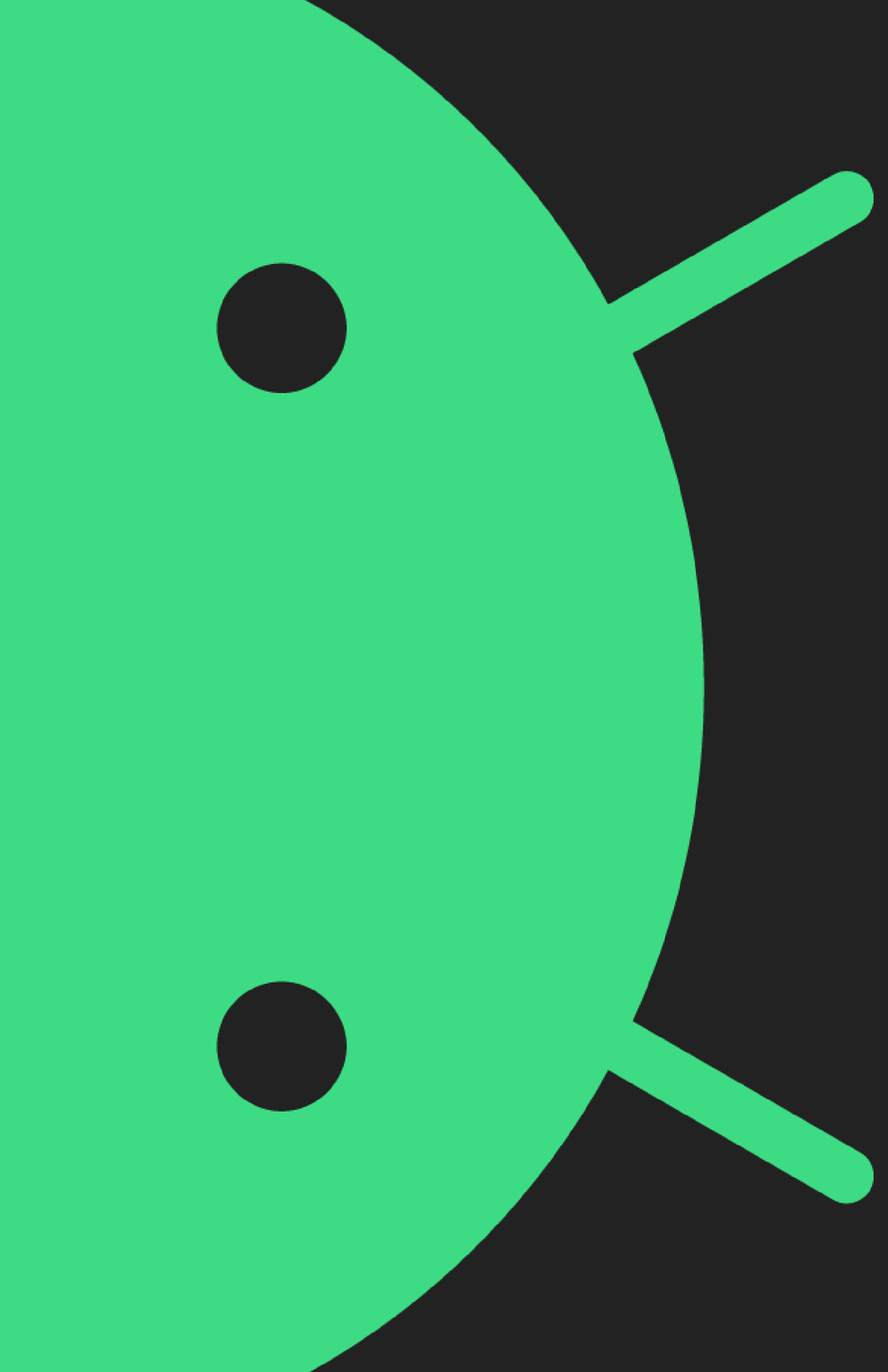
A SOLVER APPROACH

Jérémy Breton

About Me

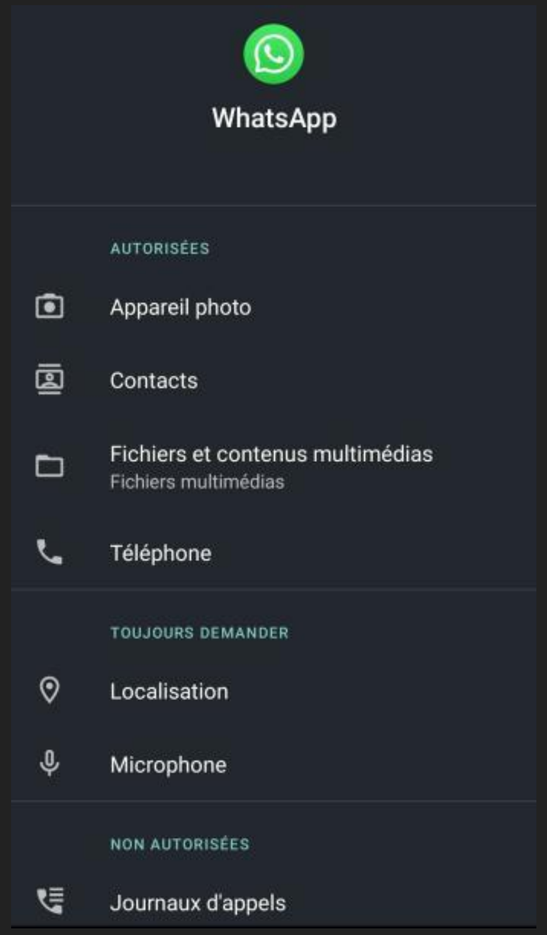
- Jérémy Breton (ghizmo)
- ISEP (Institut Supérieur d'Electronique de Paris)
- 2021: Erasmus @ IST (Institut Superior Técnico – Lisbon)
- 2022: Internship @ Thalium

SUMMARY

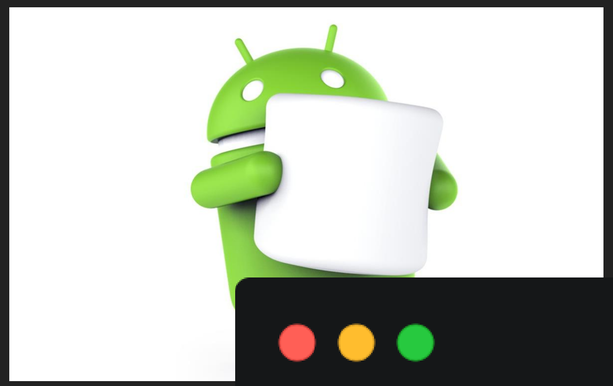


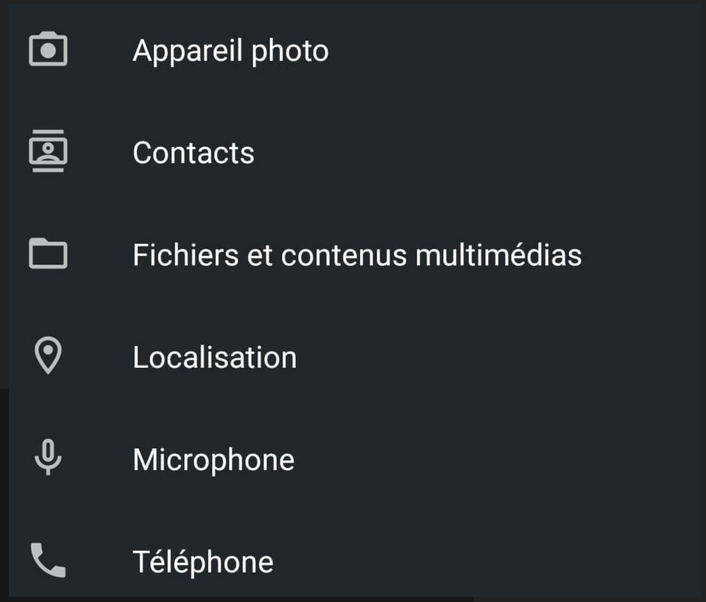
INTRODUCTION

INTRODUCTION

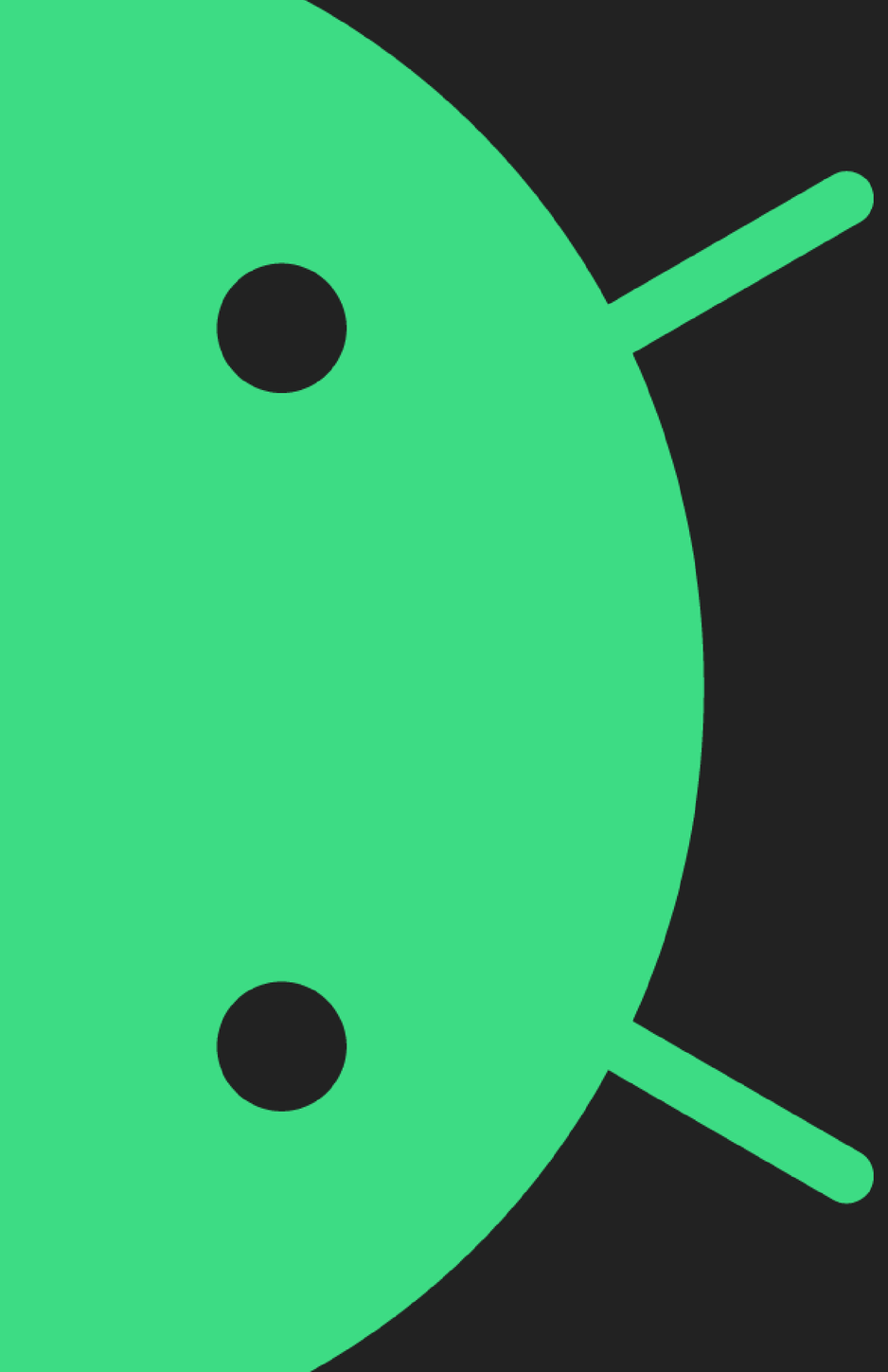


INTRODUCTION





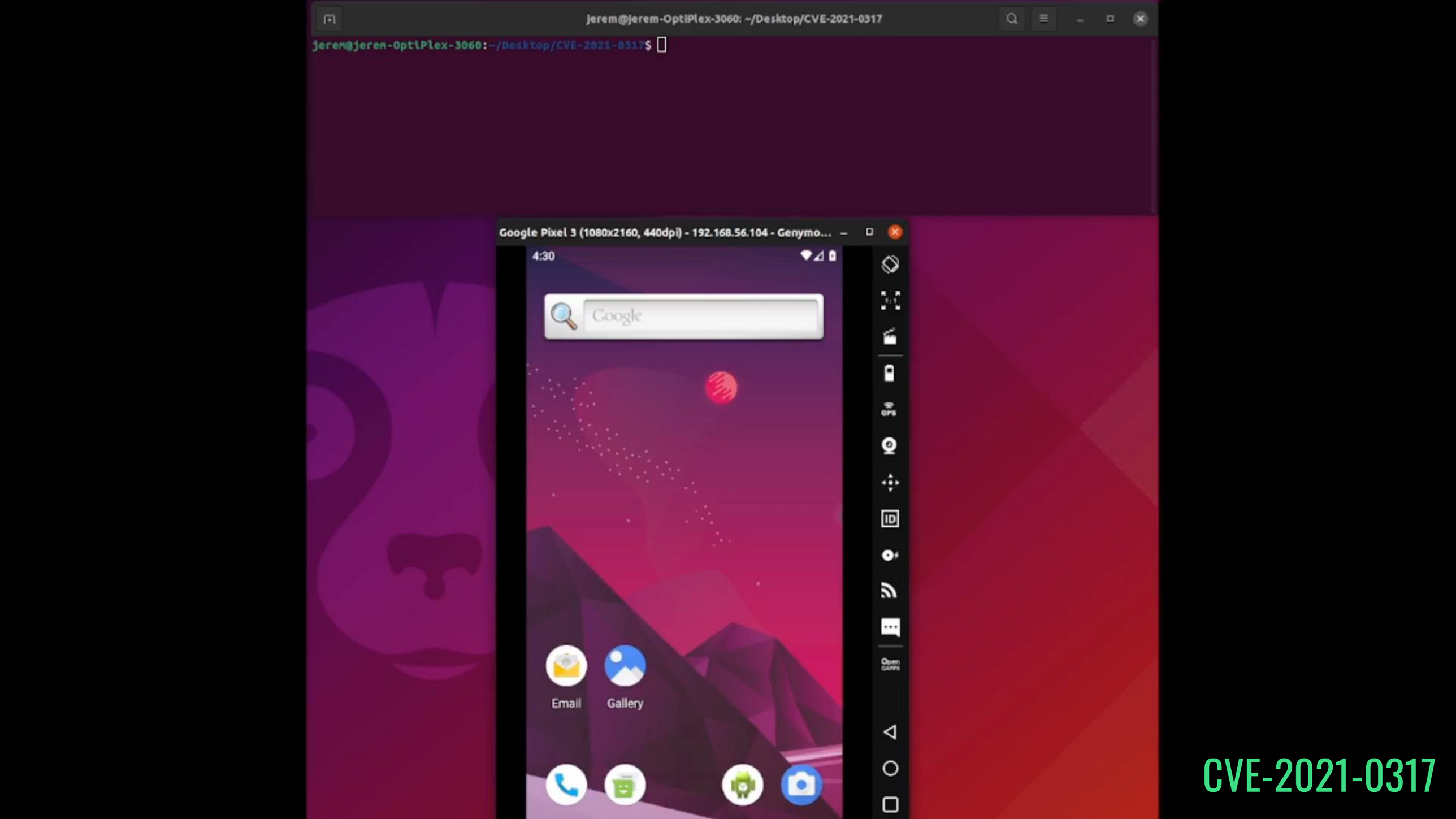

```
<permission android:name="com.test.cp"
android:protectionLevel="dangerous"
android:permissionGroup="android.permission-group.PHONE">
</permission>
```



STATE OF THE ART

STATE OF THE ART

- Few researches
- 4 CVES in 2020-2021 found by fuzzing (Severity: High)
    - Android Custom Permissions Demystified: From Privilege Escalation to Design Shortcomings


jerem@jerem-OptiPlex-3060: ~/Desktop/CVE-2021-0317
jerem@jerem-OptiPlex-3060:~/Desktop/CVE-2021-0317$
Google Pixel 3 (1080x2160, 440dpi) - 192.168.56.104 - Genymo...
4:30
Google
Email
Gallery
CVE-2021-0317

CVE-2021-0317

Attack Case:

- App def and request normal custom: com.test.cp
- App updated:

```
<permission android:name="com.test.cp"
android:protectionLevel="dangerous"
android:permissionGroup="android.permission-group.PHONE" />

<uses-permission android:name="com.test.cp" />
<uses-permission android:name="android.permission-group.PHONE" />
```

1) Install App
2) Update App
3) Reboot Phone

-> CALL_PHONE granted without user consent

CVE-2021-0317

How it works:

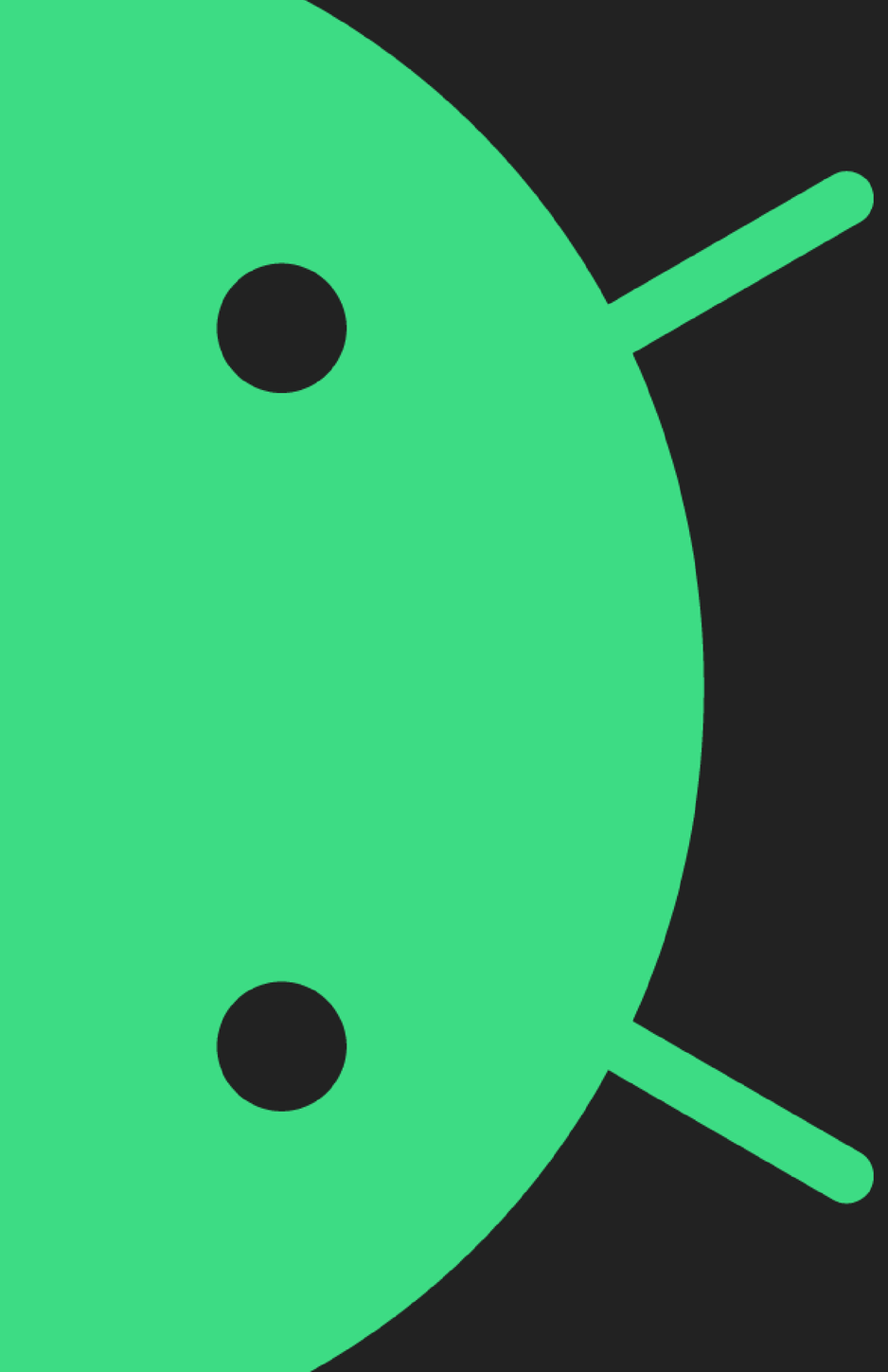
- App installation may update custom permissions
- If try to update normal/signature to dangerous
  System keep old protection level (to prevent upgrade attack)

-> If find a way to refresh permissions granting status
= Privilege Escalation !

OS init -> PMS scan APKs in app folders

Later, custom perms' protection levels will be updated according to the scan

-> Perms defined recorded by the system will be updated !



SOLVER

SOLVEUR

TÉCNICO LISBOA

Algorithms for Computational Logic

- Problem: Is propositional formula ϕ with n variables satisfiable?

Example:

$$(x \vee y) \wedge (x \vee \neg y)$$

Example:

$$(x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

Example:

$$(x) \wedge (\neg x \vee y) \wedge (\neg y \vee \neg z) \wedge (x \vee y \vee z) \wedge (\neg y \vee z) \wedge (\neg x \vee \neg y)$$

- There is at least one number in each entry:

$$\bigwedge_{x=1}^{9} \bigwedge_{y=1}^{9} \bigvee_{z=1}^{9} s_{xyz}$$

- Each number appears at most once in each row:

$$\bigwedge_{y=1}^{9} \bigwedge_{z=1}^{9} \bigwedge_{x=1}^{8} \bigwedge_{i=x+1}^{9} (\neg s_{xyz} \vee \neg s_{iyz})$$

- Each number appears at most once in each column:

$$\bigwedge_{x=1}^{9} \bigwedge_{z=1}^{9} \bigwedge_{y=1}^{8} \bigwedge_{i=y+1}^{9} (\neg s_{xyz} \vee \neg s_{xiz})$$

- Each number appears at most once in each 3x3 sub-grid:

$$\bigwedge_{z=1}^{9} \bigwedge_{i=0}^{2} \bigwedge_{j=0}^{2} \bigwedge_{x=1}^{3} \bigwedge_{y=1}^{3} \bigwedge_{k=y+1}^{3} (\neg s_{(3i+x)(3j+y)z} \vee \neg s_{(3i+x)(3j+k)z})$$

$$\bigwedge_{z=1}^{9} \bigwedge_{i=0}^{2} \bigwedge_{j=0}^{2} \bigwedge_{x=1}^{3} \bigwedge_{y=1}^{3} \bigwedge_{k=x+1}^{3} \bigwedge_{l=1}^{3} (\neg s_{(3i+x)(3j+y)z} \vee \neg s_{(3i+k)(3j+l)z})$$

SOLVEUR

```
innocent(Suspect) :- motif(Suspect), not coupable(Suspect).
```

```
motif(toto).
motif(patrick).
coupable(toto).
```

```
jerem@jerem-OptiPlex-3060:~/Desktop$ clingo crime.lp
clingo version 5.4.1
Reading from crime.lp
Solving...
Answer: 1
motif(toto) motif(patrick) coupable(toto) innocent(patrick)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.000s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

Solveur

MODEL:

- System or Custom
- Levels : Normal / Dangerous
- Dangerous permissions can be grouped
- First to define
- Actions : install / update / reboot …
- …

```
%------------ PARAMETERS ------------%

% NB APPS SYST GENERATED
#const appsyst = 2.
#const permsyst = appsyst.
% same amount of apps, each apps define his own perms/groups.

% NB APPS USER GENERATED
#const apps = 3.
#const perms = 2. % must be >0
#const groups = perms.
#const manifests = 3.

% NB STEPS
#const steps = 7.

%-------------------------------------%
```

```
%---------- User ----------%
% app(idApp)
1 { app(A): A=apps+1..a+appsyst}.


%-- PERMS --%

% Permissions may change over actions

% permManifest(idPerm, idGrp, pl, syst)
% perm in manifest
steps*perms { permManifest(Perm,Group,Level,Step):
    Perm=permsyst+1..perms+permsyst, Group=0..groups+permsyst, Level=1..2, Step=1..steps }.


% Normal perms cannot be grouped
:- permManifest(Perm,Group,1,Step), Group!=0.

% perms cannot have the same id at the same S
:- permManifest(Perm1, Group1, Level1, Step2), permManifest(Perm1, Group2, Level2, Step2), Group1 != Group2.
:- permManifest(Perm1, Group1, Level1, Step2), permManifest(Perm1, Group2, Level2, Step2), Level1 != level2.
```

LEVERAGING ANDROID PERMISSIONS: A SOLVER APPROACH

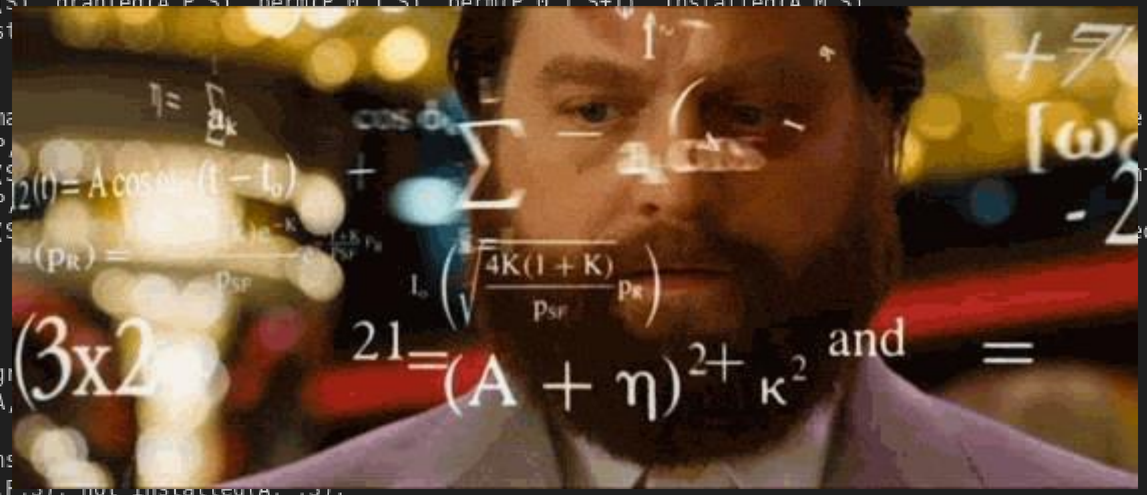
```
%-- MANIFEST --%

% all possibilities of manifest
{ manifest(Manifest,Use,Define):
    Manifest=appsyst+1..appsyst+manifests, Use=1..perm+permsyst, Define=permsyst+1..perm+permsyst }.

% manifest that doesn't define any perms
{ manifest(Manifest,Use):
    Manifest=appsyst+manfiests+1..appsyst+manifest+manifest, Use=1..perm+permsyst }.

% all posibilities of perms definitions and utilizations
{use(Use,Perm) : Use=1..perm+permsyst , Perm=1..perm+permsyst}. % U to use different perms in same time
{defineP(Define,Perm) :
    Define=permsyst..perm+permsyst, Perm=permsyst+1..permsyst+perm}. % D to define different perms in same time

% if defineP, then we must have a perm
:- defineP(Define,Perm), not permManifest(Perm,_,_,_).

% if manifest(Manifest,Use,Define), then we must have defineP(Define,_) and use(Use,_)
:- manifest(Manifest,Use,Define), not defineP(Define,_).
:- manifest(Manifest,Use,Def), not use(Use,_).

:- manifest(Manifest,Use1), manifest(Manifest,Use2), Use1 != Use2.
:- manifest(Manifest1,Use), manifest(Manifest2,Use), Manifest1 != Manifest2.
```

```
%-------------- Actions ---------------%

action(1..steps).


% generate all possibilities of actions
1 { install(A,M,S) : app(A), manifest(M,_,_) ;
    install(A,M,S) : app(A), manifest(M,_) ;
    uninstall(A,S) : app(A) ;
    run(A,S) : app(A) ;
    stop(A,S) : app(A) ;
    grant(A,P,S) : app(A),permManifest(P,_,_,S) ;
    grant(A,P,S) : app(A),permSyst(P,_,_) ;
    grantAuto(A,P,S) : app(A), permManifest(P,_,_,S) ;
    grantAuto(A,P,S) : app(A), permSyst(P,_,_) ;
    grantOneTime(A,P,S) : app(A), permManifest(P,_,_,S);
    grantOneTime(A,P,S) : app(A), permSyst(P,_,_) ;
    reboot(S) ;
    update(A,M,S) : app(A), manifest(M,_,_) ;
    update(A,M,S) : app(A), manifest(M,_)
} 1 :- action(S).
```

```
% if install, then installed at S+1
installed(A,M,S+1) :- install(A,M,S).

% if not installed, we cannot have the run action
:- run(A,S), not installed(A,_,S).

% if it's installed, we cannot install it
:- install(A,_,S), installed(A,M,S).
```

LEVERAGING ANDROID PER

```
%---------- GRANT ----------%

% a perm can be granted to an app, if this app is installed and the perm is in use in his manifest,
% and this perm is defined
granted(A,P,S+1) :- grant(A,P,S), installed(A,M,S), manifest(M,U,D), use(U,P), defPerm(_,P,S).
granted(A,P,S+1) :- grant(A,P,S), installed(A,M,S), manifest(M,U), use(U,P), defPerm(_,P,S).


% the perm is still granted if we dont update a dangerous perm, if we dont reboot, and if we dont uninstall
granted(A,P,S+1) :-
    not updateDangerousPerm(P,S), not revokeGrant(A,P,S), not uninstall(A,S),
    granted(A,P,S), installed(A,M,S), manifest(M,U,D), use(U,P), defPerm(_,P,S+1), S<=s.
granted(A,P,S+1) :-
    not updateDangerousPerm(P,S), not revokeGrant(A,P,S), not uninstall(A,S),
    granted(A,P,S), installed(A,M,S), manifest(M,U), use(U,P), defPerm(_,P,S+1), S<=s.


% when we reboot, if the perm is updated from normal to dangerous we revoke the grant
revokeGrant(A,P,S) :- reboot(S), perm(P,G,1,S), perm(P,G2,2,S+1), granted(A,P,S).




% the perm is still granted if we reboot and the perm is still normal
% (if after reboot the perm becomes dangerous its not more granted)
granted(A,P,S+1) :-
    reboot(S), granted(A,P,S), perm(P,0,1,S), perm(P,0,1,S+1), installed(A,M,S),
    manifest(M,U,D), use(U,P), defPerm(_,P,S).
granted(A,P,S+1) :-
    reboot(S), granted(A,P,S), perm(P,0,1,S), perm(P,0,1,S+1), installed(A,M,S),
    manifest(M,U), use(U,P), defPerm(_,P,S).


% package ma
granted(A,P,
    reboot(S
granted(A,P,
    reboot(S


% if it's gr
not grant(A,

% if not ins
:- grant(A,P,S), not installed(A,_,S).
```

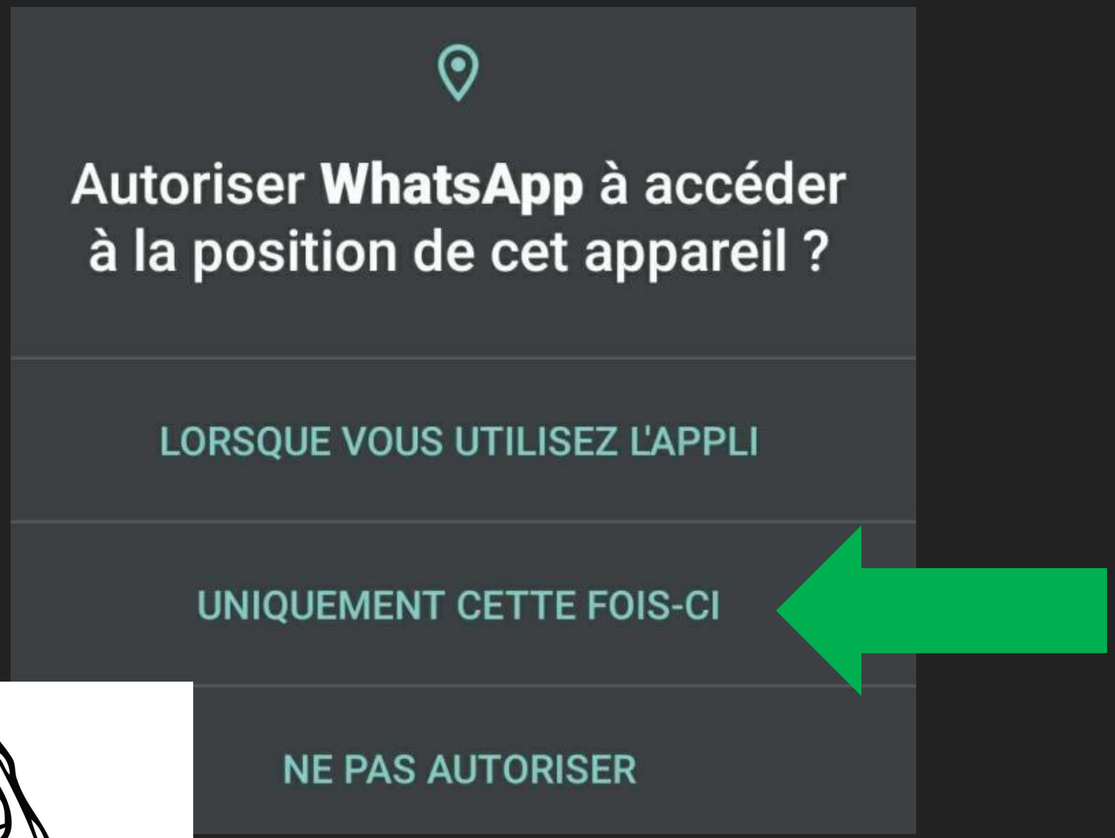


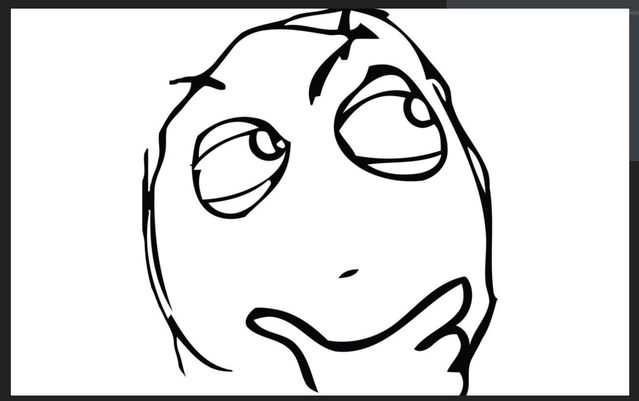
Recherche de vulnérabilités

```
% we cannot use grant action
:- grant(_,_,_).

% the system perm is granted to the user's app at the end
:- not granted(A,P,S), A=as+1, P=as, S=s+1.
```

LEVERAGING ANDROID PERMISSIONS: A SOLVER APPROACH

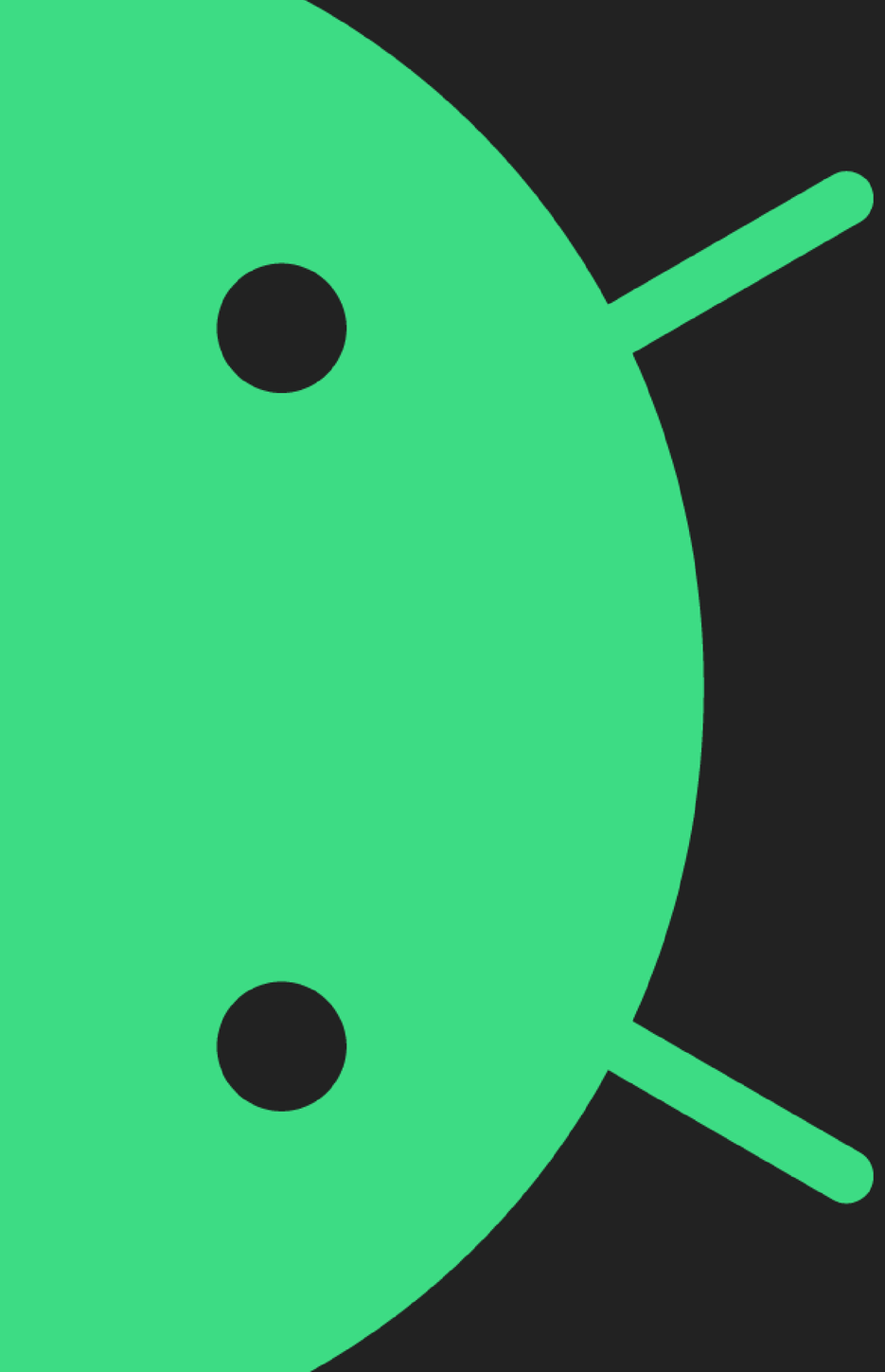
```
❯ clingo cve-2021-0317.lp
clingo version 5.4.1
Reading from cve-2021-0317.lp
Solving...
Answer: 1
action(1) action(2) action(3) action(4) action(5) manifSyst(1,1) installed(1,1,1) installed(1,1,2) installed(1,1,3) i
nstalled(1,1,4) installed(1,1,5) permSyst(1,1,2) installed(1,1,6) defPerm(1,1,1) defPerm(1,1,2) defPerm(1,1,3) defPer
m(1,1,4) defPerm(1,1,5) defPerm(1,1,6) permManifest(2,1,2,3) permManifest(2,1,2,4) permManifest(2,1,2,5) appSyst(1) m
anifest(2,1,2) permManifest(2,0,1,1) permManifest(2,0,1,2) installed(2,2,2) install(2,2,1) installed(2,2,3) installed
(2,2,4) installed(2,2,5) installed(2,2,6) update(2,2,2) usrP(2,2) defPerm(2,2,2) defPerm(2,2,3) defPerm(2,2,4) defPer
m(2,2,5) defPerm(2,2,6) app(2) reboot(3) perm(2,0,1,3) perm(2,1,2,4) perm(2,1,2,5) perm(2,1,2,6) perm(2,0,1,2) update
NormalToDangerousPerm(2,2) use(1,1) use(1,2) use(2,2) granted(2,1,6) granted(2,2,6) granted(2,2,5) granted(2,2,4) gra
nted(2,2,3) granted(2,2,2) running(2,5) run(2,4) running(2,6) grantAuto(2,1,5) newPermManifest(2,3) grantAutoOK(2,1,5
) grantAutoOK(2,1,4)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.052s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.046s
```


Autoriser WhatsApp à accéder à la position de cet appareil ?
LORSQUE VOUS UTILISEZ L'APPLI
UNIQUEMENT CETTE FOIS-CI
NE PAS AUTORISER

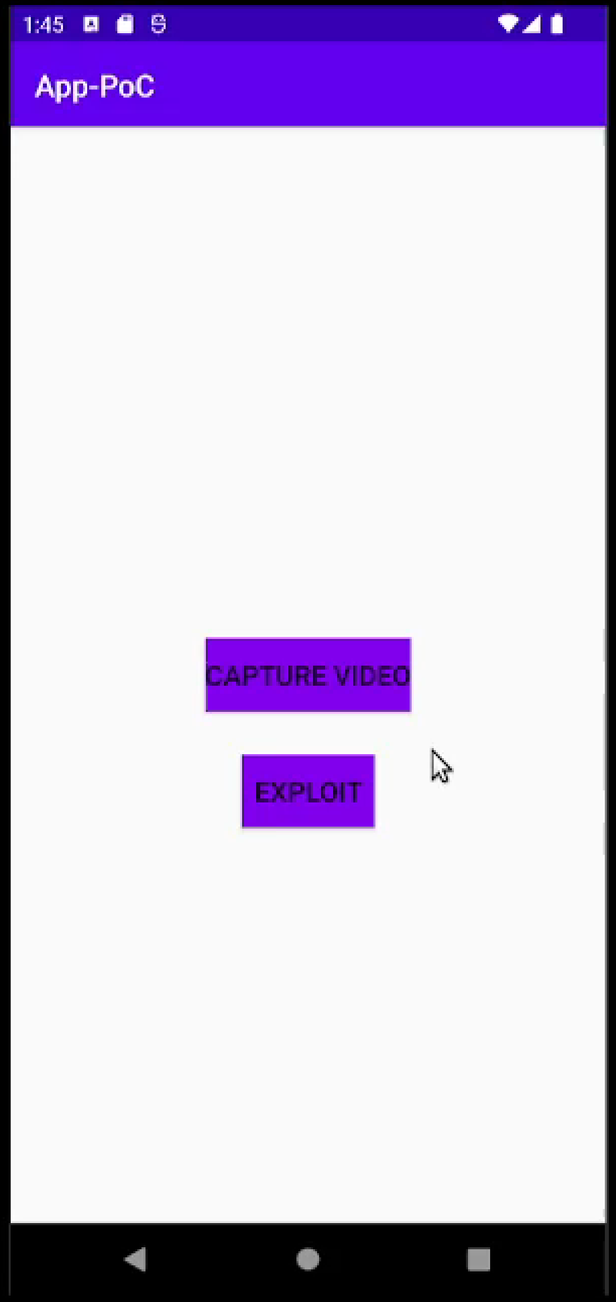


SOLVEUR

```
$ clingo oathime.lp
clingo version 5.4.1
Reading from oathime.lp
Solving...
Answer: 1
appSyst(1) manifSyst(1,1) permSyst(1,1,2)
app(2) manifest(3,1,2) use(1,1) use(1,2) usrP(2,2)
perm(2,1,2,1) ... perm(2,1,2,8)
install(2,3,1) run(2,2) grantOneTime(2,1,3) grantAuto(2,2,4) stop(2,5) run(2,6) grantAuto(2,1,7)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.007s (Solving: 0.00s 1st Mode
CPU Time     : 0.007s
```



VULNERABILITY

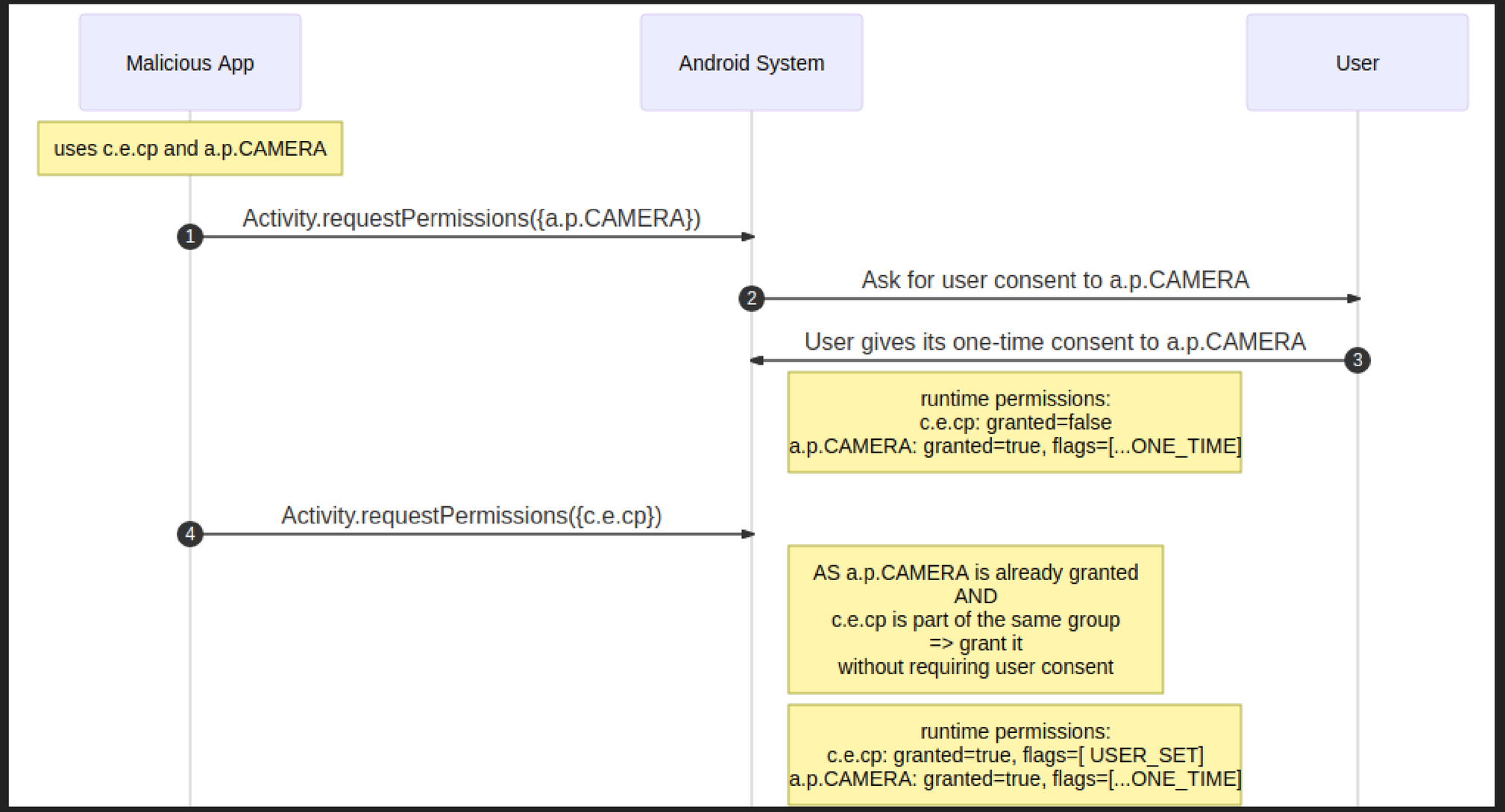


CVE-2023-20947

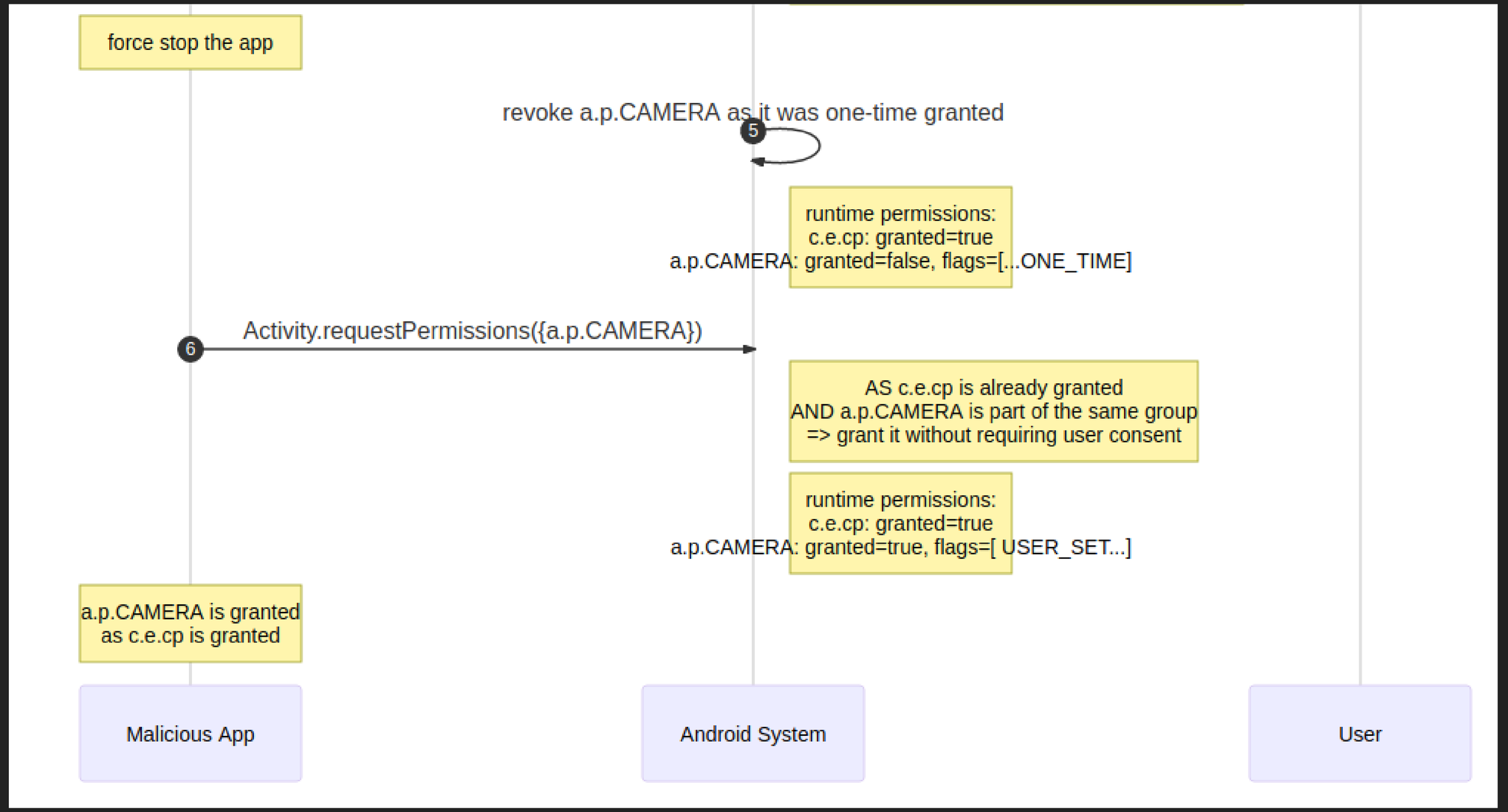
VULNERABILITY

```
<permission android:name="com.example.cp"
    android:protectionLevel="dangerous"
    android:permissionGroup="android.permission-group.CAMERA" />

<uses-permission android:name="com.example.cp"/>
<uses-permission android:name="android.permission.CAMERA" />
```

LEVERAGING ANDROID PERMISSIONS: A SOLVER APPROACH



LEVERAGING ANDROID PERMISSIONS: A SOLVER APPROACH



LEVERAGING ANDROID PERMISSIONS: A SOLVER APPROACH

Hello,

The Android security team has conducted a … sed on our published severity assessment matrix (1) it was rated as High severity. Thi … am for remediation, and we're targeting a fix for releas… … le. We ask for your c…

Thank you

Android Security Team

(1) Severity Matrix: https://source.android.… …erity

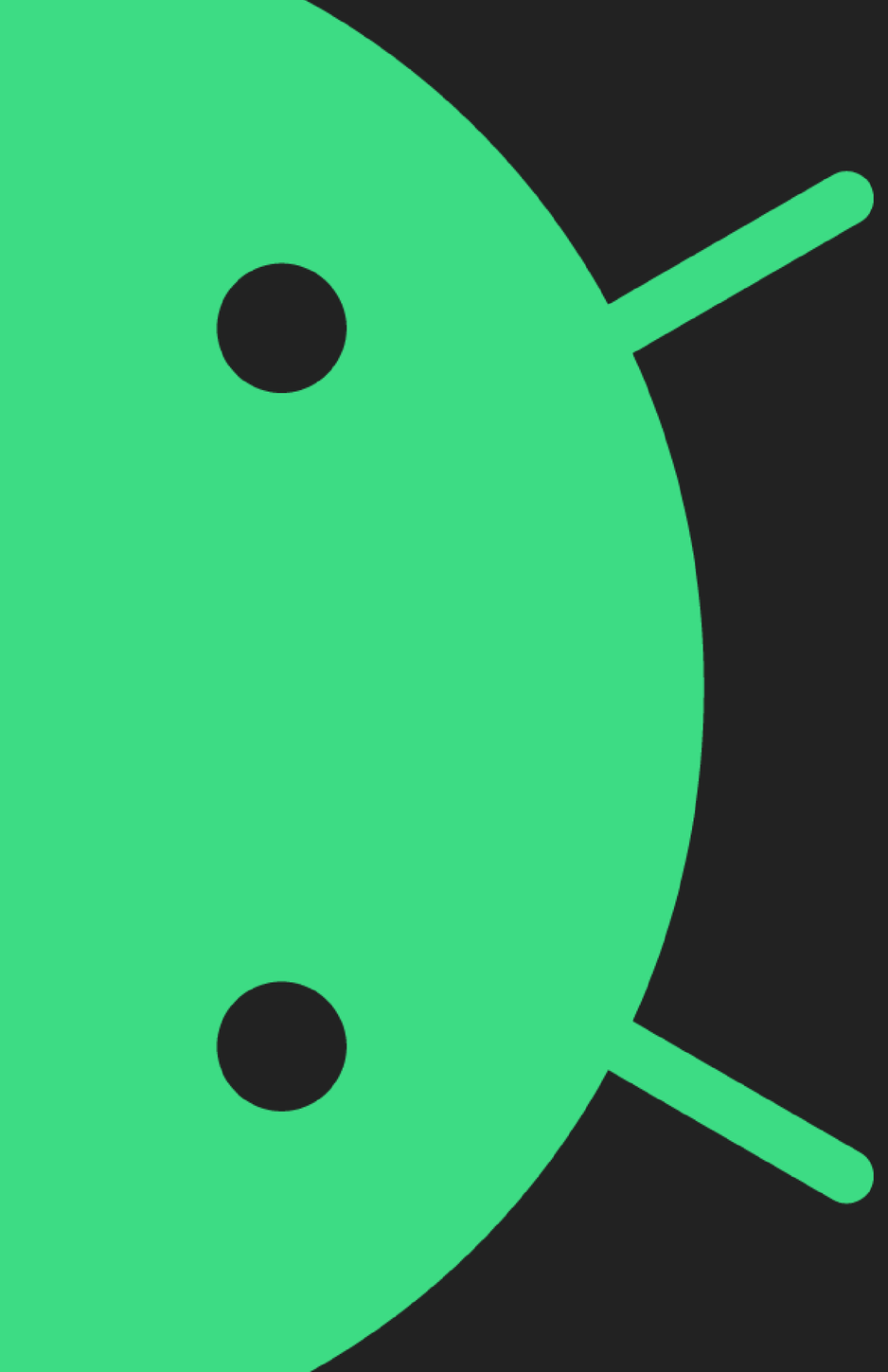
ASR Severity: <none> → High

ASR Seve… → High

Your CVE ID is CVE-2023-20947.

Thanks,
Android Security Team

CVE-2023-20947	A-237405974	EoP	High	12, 12L, 13

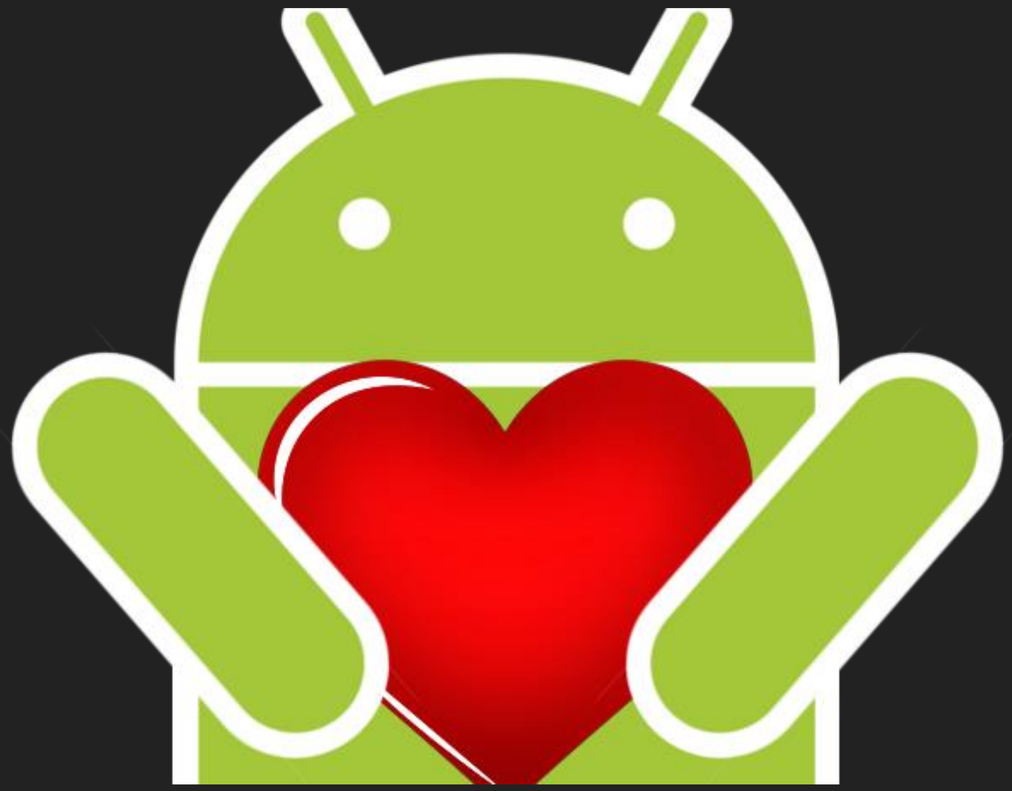


CONCLUSION

CONCLUSION

- A field not much explored
- Tool to help research
- Can be applied to other systems
- Limitations

THANKS!



QUESTIONS?