GroddDroid
A Gorilla for Triggering Malicious Behavior

10th Int. Conference on Malicious and Unwanted Software
October 20-23rd 2015

A. Abraham, R. Andriatsimandefitra, A. Brunelat
Jean-François Lalande, Valérie Viet Triem Tong

CominLabs

Inria
INVENTEURS DU MONDE NUMÉRIQUE

INSA
INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
CENTRE VAL DE LOIRE

CentraleSupélec

Executing Android Malware is necessary

- *To observe their behavior*
- *To understand them*
- *To test the robustness of a protection*

➢ But malware do not run on demand

- **starts immediately !**
- **starts when the phone is unlocked**
- **starts after a reboot**
- **sleeps a week long**
- **detects emulators**
- **waits for a message of their master**
- **…**

Existing approaches: Monkey, PuppetDroid and A³E

- ***The Monkey*** *hits randomly the graphical interface and can be combined with a random sequence of events (SMS, phone call, reboot…)*
- ***PuppetDroid*** *Re-execute recorded interactions and reproduces the typical UI-interaction of a potential victim of the malware.*
- ***Android Automatic App Explorer (A³E)*** *extracts GUI elements and generates related events handlers to mimicking a real user.*

BUT

- *Remain inefficient to trigger delayed attack or commanded by a remote server*
- *Is not able to recreate the same scenario twice (Monkey)*

Our proposition: identification and forcing of malicious behaviors

GRODDDROID

GroddDroid:

1. Identifies suspicious part of the bytecode
2. Plays with the app as Grodd
3. Forces the malicious code if needed
4. Is freely available on http://kharon.gforge.inria.fr/

1st Step: *Suspicious code targeting*

For each method in the bytecode computes a risk score.
The more the method uses sensitive APIs, the higher is the score.

Example of scoring :

Android.telephony.SmsManager +50

Android.telephony.TelephonyManager +20

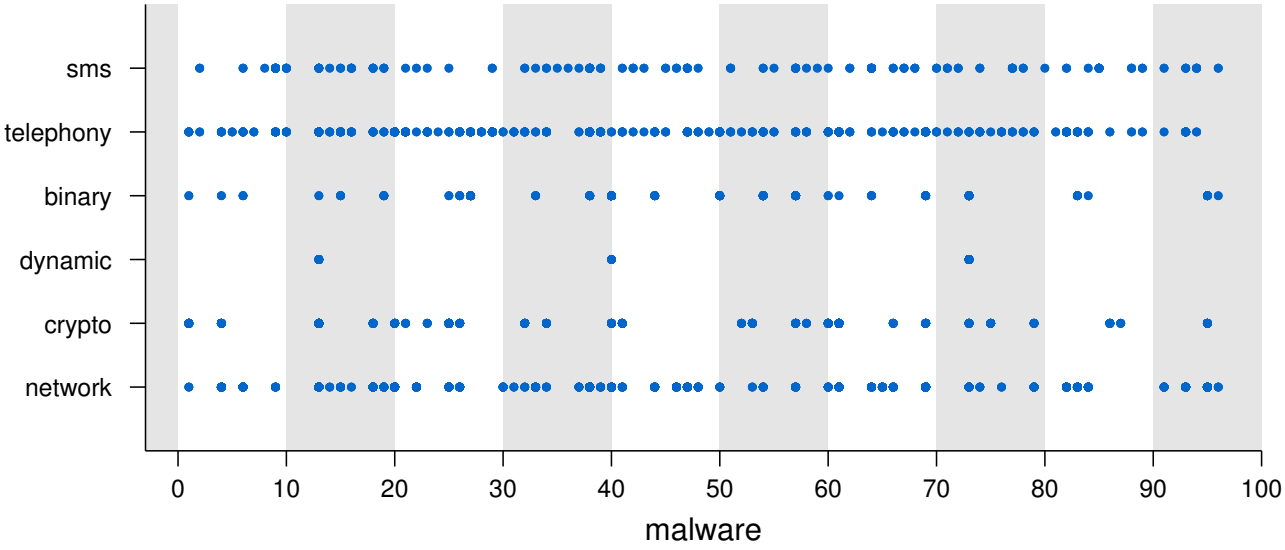
Java.lang.Process +10

Java.net.UrlConnection +3

GRODDROID

Is our scoring function well-adapted to malware ?

We verify that the pointed out APIs are really used by malware (on a dataset of 100 malware)



GRODDROID

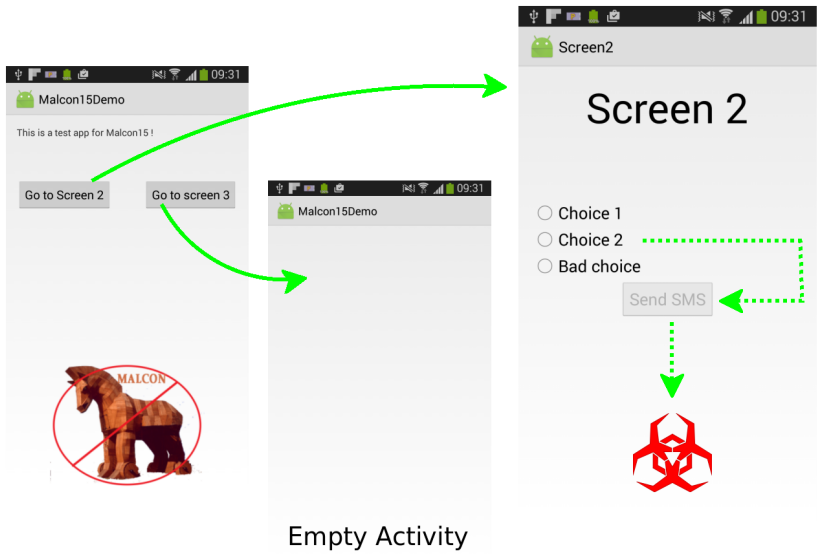
Did we succeed to target malicious bytecode ?

We have test the scoring function on a dataset of well studied malware : Kharon15

Malware	High score	Malicious or not ?	Most scored method
BadNews	80	✓	gathers user's information
Cajino	200	✓	Sends SMS
DroidKungFu	50	✓	Run a binary exploit
MobiDash	147	WRONG	gathers user's information for legitimate use
SaveME	100	✓	Sends SMS
SimpleLocker	-	CRASH	
WipeLocker	150	✓	Sends SMS

GRODDROID

2nd Step: *Running the app as a Gorilla*



① Collects graphical elements
② Explores the app by clicking on the buttons
③ Can go back
④ Can launch the app again
⑤ Detects loops
⑥ Until he has explored all the different activities

GRODDROID

Is our Gorilla better than Monkey and A³E ?

We compare the code coverage on 100 tested malware

- Always better than A³E (cannot handle properly recent Android apps)
- Slightly better than Monkey
- 23 crashs
- Serves a reference execution path

Stimulating the GUI is not sufficient

GroddDroid forces the execution path direct to the most ranked method.

GRODDDROID

3 step: Running the app as GroddDroid

GroddDroid

① modifies the bytecode and cancels the conditional jumps that could drive away from the malicious code

② Recomputes an execution path reachable by a gorilla to execute the most scored unit of code

Let us have an example

GRODDDROID

3 step: Running the app as GroddDroid

① modifies the bytecode and cancels the conditional jumps that could drive away from the malicious code

The source code of a simple protection

```
if (isOnEmulator()) return; // Branch 1
else
manager = SmsManager.getDefault(); // Branch 2
```

GRODDDROID

3 step: Running the app as GroddDroid

① modifies the bytecode and cancels the conditional jumps that could drive away from the malicious code

Same code in Jimple (intermediate representation of bytecode)

```
$z0 = staticinvoke <DummyClass: boolean isOnEmulator()>();
if $z0 != 0 goto label3;
return; // Branch 1
label3: // Branch 2
$r6 = staticinvoke <SmsManager: SmsManager
getDefault()>();
```

GRODDDROID

3 step: Running the app as GroddDroid

① modifies the bytecode and cancels the conditional jumps that could drive away from the malicious code

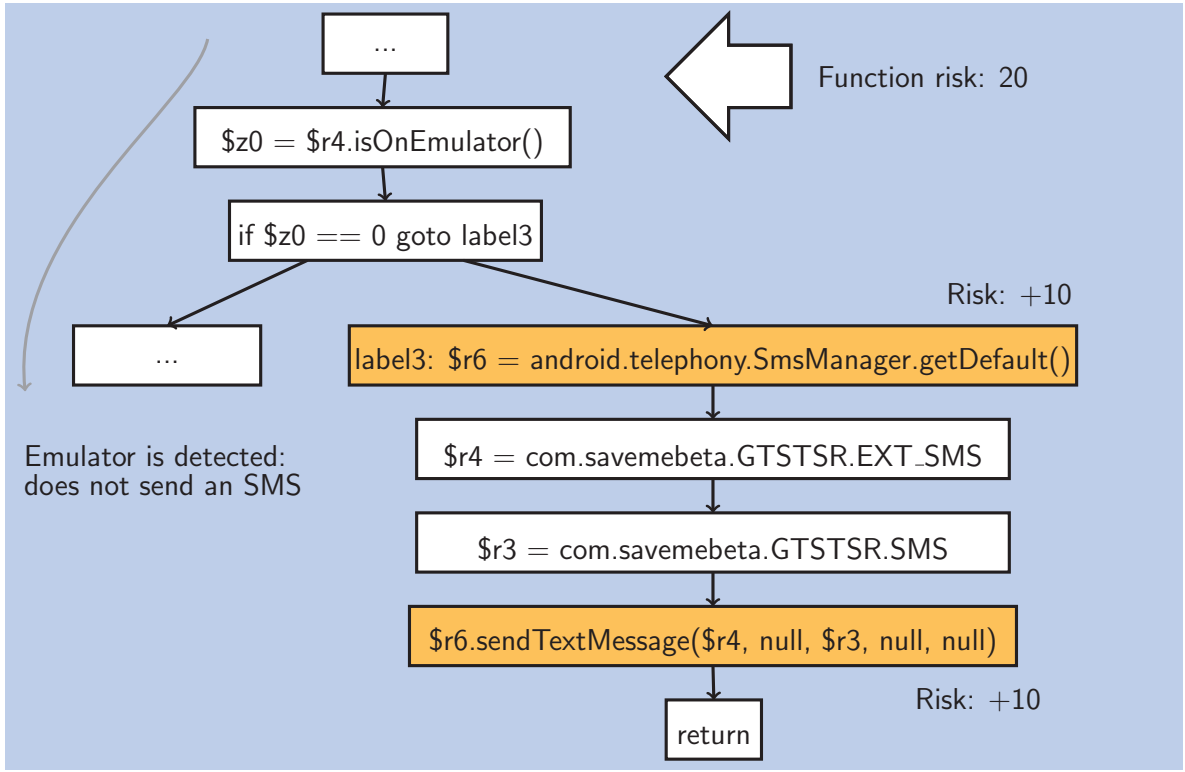
Forced code: the conditional jump is modified

```
$z0 = staticinvoke <DummyClass: boolean isOnEmulator()>();
goto label3 ; // Forced branch 2
return; // Branch 1 is now unreachable
label3: // Branch 2 is always executed
$r6 = staticinvoke <SmsManager: SmsManager
getDefault()>();
```

GRODDDROID

3 step: Running the app as GroddDroid

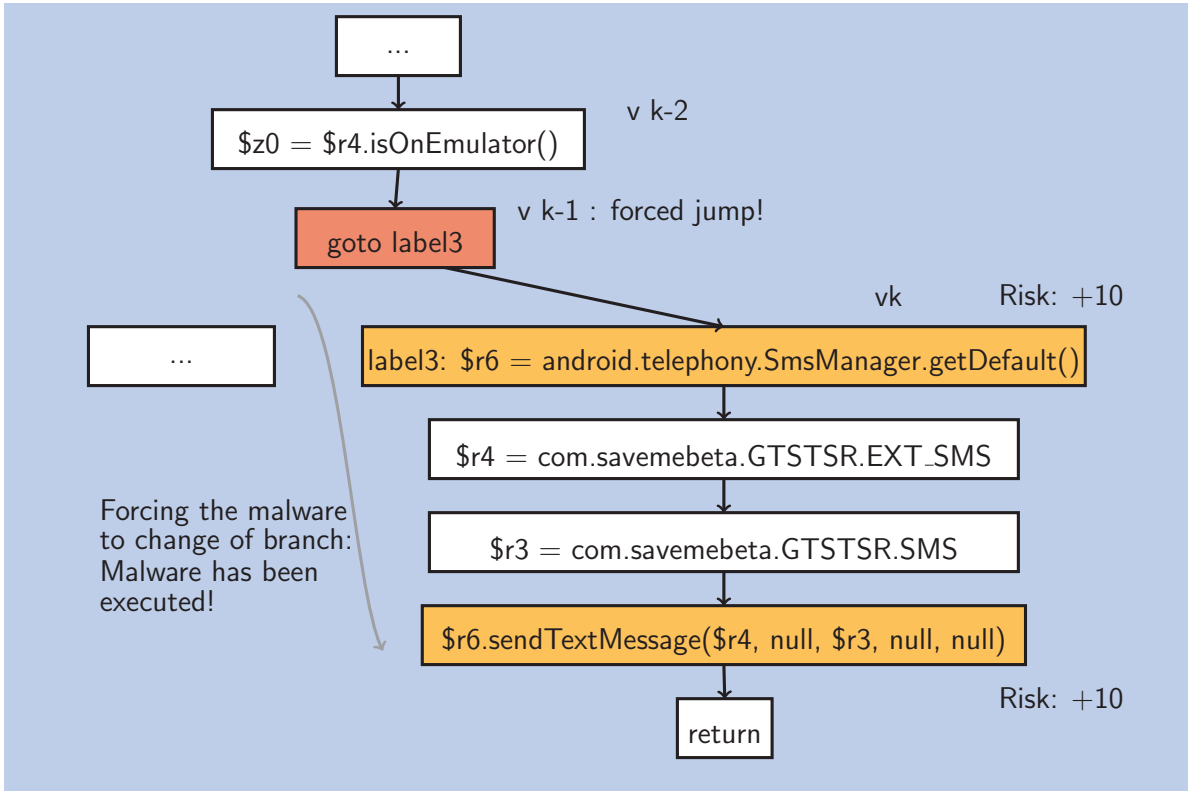
② *The process is repeated to exhibit an execution path*



GRODDDROID

3 step: Running the app as GroddDroid

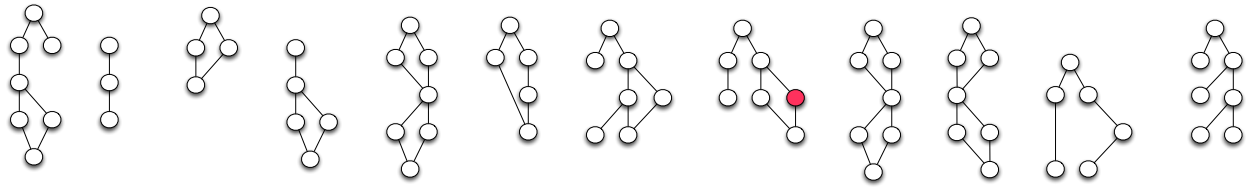
② *The process is repeated to exhibit an execution path*



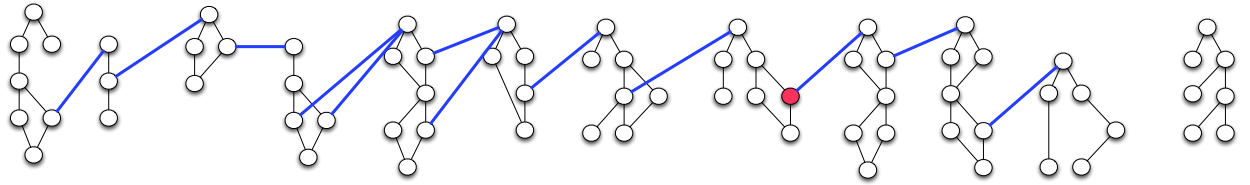
GRODDDROID

Reconstructing an execution path

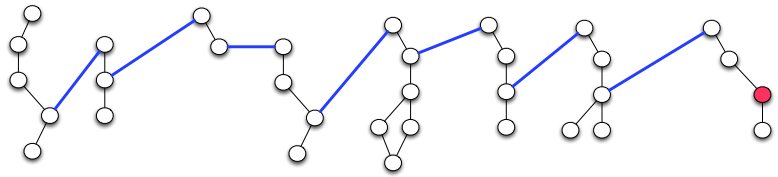
1. *Computation of Control Flow Graph of each methods in the bytecode*



2. *Method calls & intents connect the graphs*

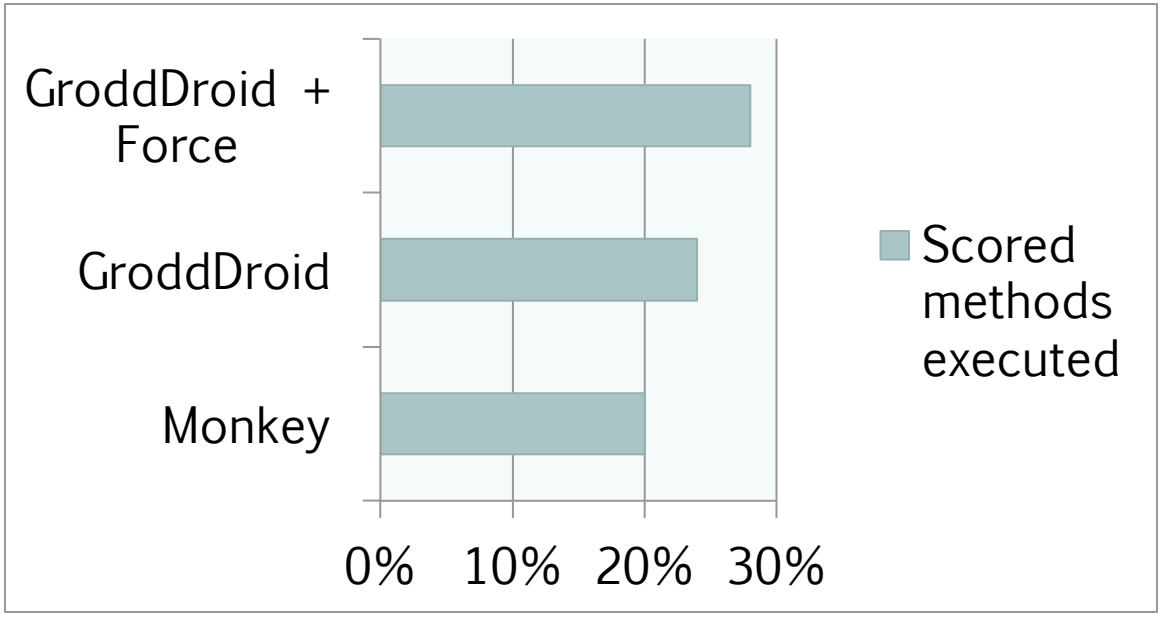


3. *A forced execution path is a shortest path from the target to an activity*
4. *Conditions along the path are all forced*



Is GroddDroid able to force malware to execute ?

We recompile the modified bytecode and run it. We test this procedure on a dataset of 100 malware.



GRODDDROID

Summary and Conclusion

GroddDroid provides a solution to defeat protections of Android malware using:

- ✓ *Automatic identification of malicious code*
- ✓ *Intelligent exploration of activities*
- ✓ *A forcing of the most scored unit of code*

Can certainly be improved

- in taking into account other GUI elements
- in forcing more than one execution path

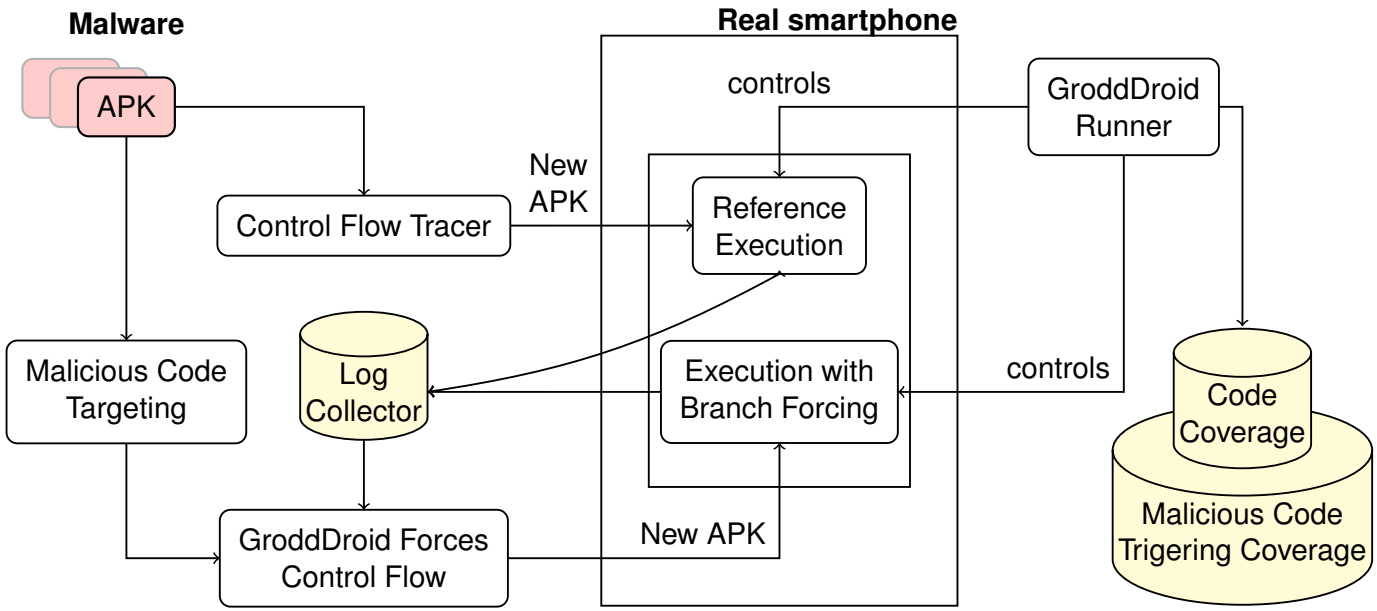
GRODDDROID

Visit our web site
and use GroddDroid
http://kharon.gforge.inria.fr/

Feel free to contact us
jean-francois.lalande@insa-cvl.fr
valerie.viettriemtong@centralesupelec.fr

GRODDDROID

Overview of the GroddDroid framework



GRODDDROID

Bibliography

- ***Android Automatic App Explorer (A³E)***

Targeted and depth-first exploration for systematic testing of android apps. . A.Tanzirul, J. Neamtiu

Proceedings of the 2013 ACM SIGPLAN international conference on Object oriented programming systems languages & applications

- ***PuppetDroid***

PuppetDroid : a remote execution environment and UI exerciser for Android malware analysis. A. Gianazza,F. Maggi, A. Fattori, L. Cavallaro, S. Zanero

Avail. on arxiv http://s2lab.isg.rhul.ac.uk/papers/files/arxiv2014.pdf