Yet Another Way to Collect Android Malicious Behaviour

INFOSEK@Nova Gorica
December 2, 2016
Speaker: Chengyu Zheng
Authors: C. Zheng, M. D. Preda, J. Granjal,
S. Zanero, and F. Maggi

PaLSIT

About Me

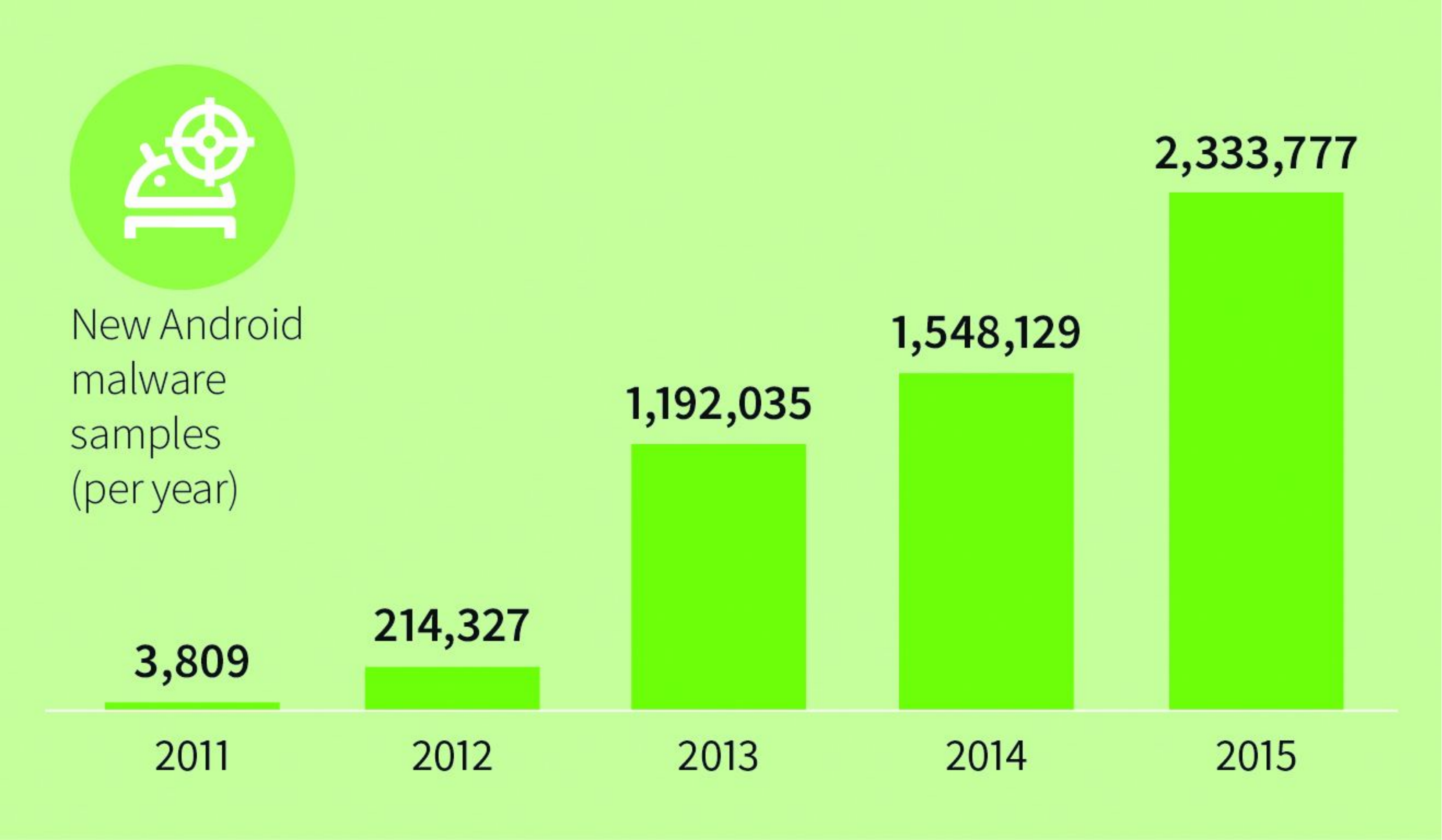
Name: Chengyu Zheng

Position: 2nd year phd student

Research area: Security in Mobile Environment

Location: NECSTLab at Politecnico di Milano

Android Malware Trend (G data)



PALSIT

Automated App Analysis

Static analysis

❑ parse the application binary code

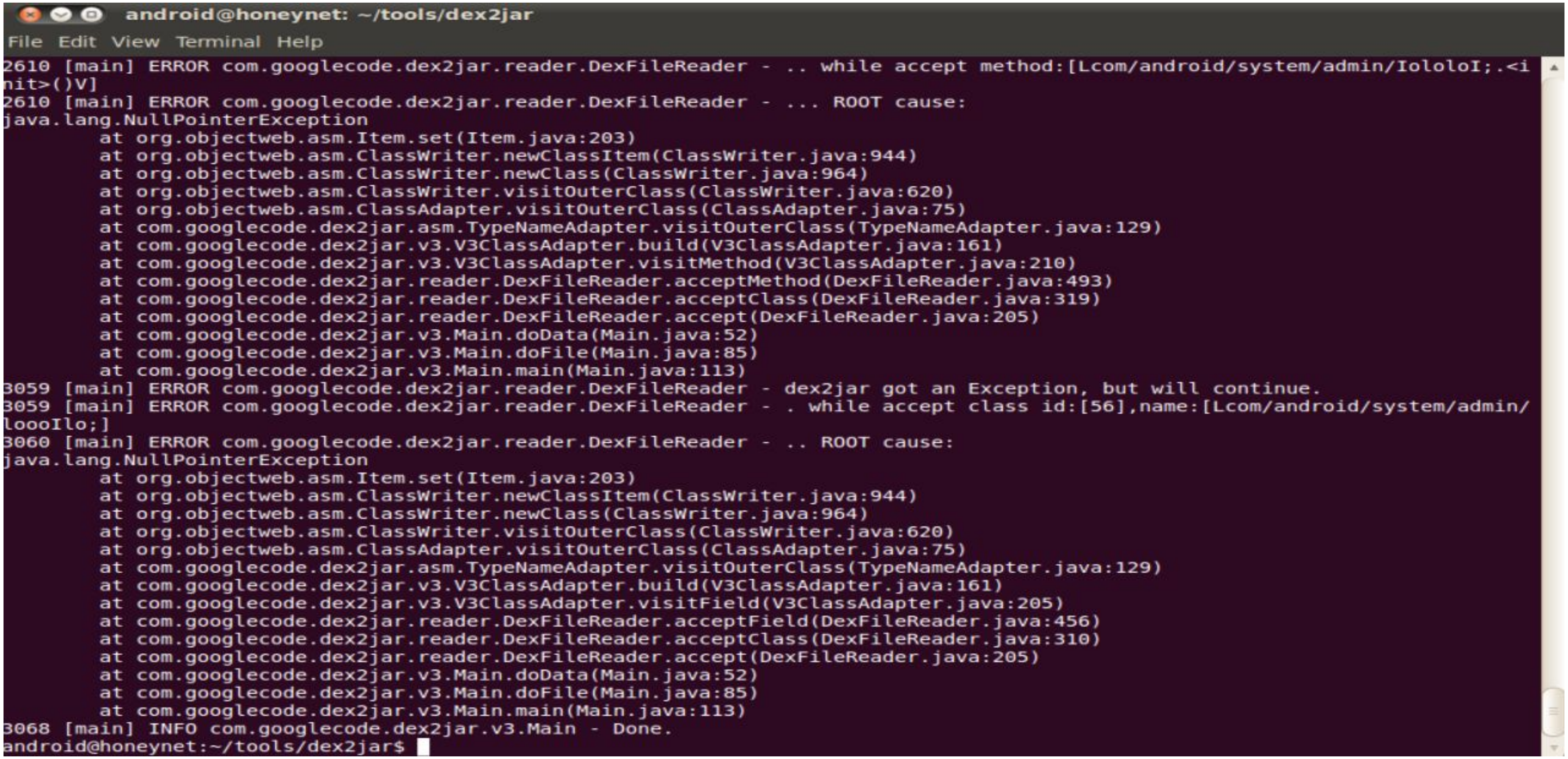
  ❑ - obfuscation, encryption, packing

Dynamic analysis

❑ observe the runtime behavior of an app

  ❑ + obfuscation, encryption, packing

Static Analysis Example



android@honeynet: ~/tools/dex2jar
File Edit View Terminal Help

```
2610 [main] ERROR com.googlecode.dex2jar.reader.DexFileReader - .. while accept method:[Lcom/android/system/admin/IololoI;.<i
nit>()V]
2610 [main] ERROR com.googlecode.dex2jar.reader.DexFileReader - ... ROOT cause:
java.lang.NullPointerException
        at org.objectweb.asm.Item.set(Item.java:203)
        at org.objectweb.asm.ClassWriter.newClassItem(ClassWriter.java:944)
        at org.objectweb.asm.ClassWriter.newClass(ClassWriter.java:964)
        at org.objectweb.asm.ClassWriter.visitOuterClass(ClassWriter.java:620)
        at org.objectweb.asm.ClassAdapter.visitOuterClass(ClassAdapter.java:75)
        at com.googlecode.dex2jar.asm.TypeNameAdapter.visitOuterClass(TypeNameAdapter.java:129)
        at com.googlecode.dex2jar.v3.V3ClassAdapter.build(V3ClassAdapter.java:161)
        at com.googlecode.dex2jar.v3.V3ClassAdapter.visitMethod(V3ClassAdapter.java:210)
        at com.googlecode.dex2jar.reader.DexFileReader.acceptMethod(DexFileReader.java:493)
        at com.googlecode.dex2jar.reader.DexFileReader.acceptClass(DexFileReader.java:319)
        at com.googlecode.dex2jar.reader.DexFileReader.accept(DexFileReader.java:205)
        at com.googlecode.dex2jar.v3.Main.doData(Main.java:52)
        at com.googlecode.dex2jar.v3.Main.doFile(Main.java:85)
        at com.googlecode.dex2jar.v3.Main.main(Main.java:113)
3059 [main] ERROR com.googlecode.dex2jar.reader.DexFileReader - dex2jar got an Exception, but will continue.
3059 [main] ERROR com.googlecode.dex2jar.reader.DexFileReader - . while accept class id:[56],name:[Lcom/android/system/admin/
loooIlo;]
3060 [main] ERROR com.googlecode.dex2jar.reader.DexFileReader - .. ROOT cause:
java.lang.NullPointerException
        at org.objectweb.asm.Item.set(Item.java:203)
        at org.objectweb.asm.ClassWriter.newClassItem(ClassWriter.java:944)
        at org.objectweb.asm.ClassWriter.newClass(ClassWriter.java:964)
        at org.objectweb.asm.ClassWriter.visitOuterClass(ClassWriter.java:620)
        at org.objectweb.asm.ClassAdapter.visitOuterClass(ClassAdapter.java:75)
        at com.googlecode.dex2jar.asm.TypeNameAdapter.visitOuterClass(TypeNameAdapter.java:129)
        at com.googlecode.dex2jar.v3.V3ClassAdapter.build(V3ClassAdapter.java:161)
        at com.googlecode.dex2jar.v3.V3ClassAdapter.visitField(V3ClassAdapter.java:205)
        at com.googlecode.dex2jar.reader.DexFileReader.acceptField(DexFileReader.java:456)
        at com.googlecode.dex2jar.reader.DexFileReader.acceptClass(DexFileReader.java:310)
        at com.googlecode.dex2jar.reader.DexFileReader.accept(DexFileReader.java:205)
        at com.googlecode.dex2jar.v3.Main.doData(Main.java:52)
        at com.googlecode.dex2jar.v3.Main.doFile(Main.java:85)
        at com.googlecode.dex2jar.v3.Main.main(Main.java:113)
3068 [main] INFO com.googlecode.dex2jar.v3.Main - Done.
android@honeynet:~/tools/dex2jar$
```

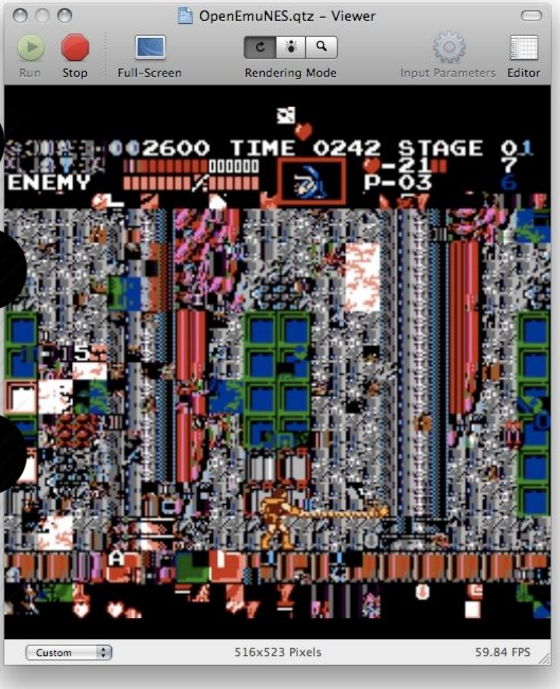
Dynamic Analysis Example (1)

Method: gdb

Countermeasure: Anti-Debug Technique

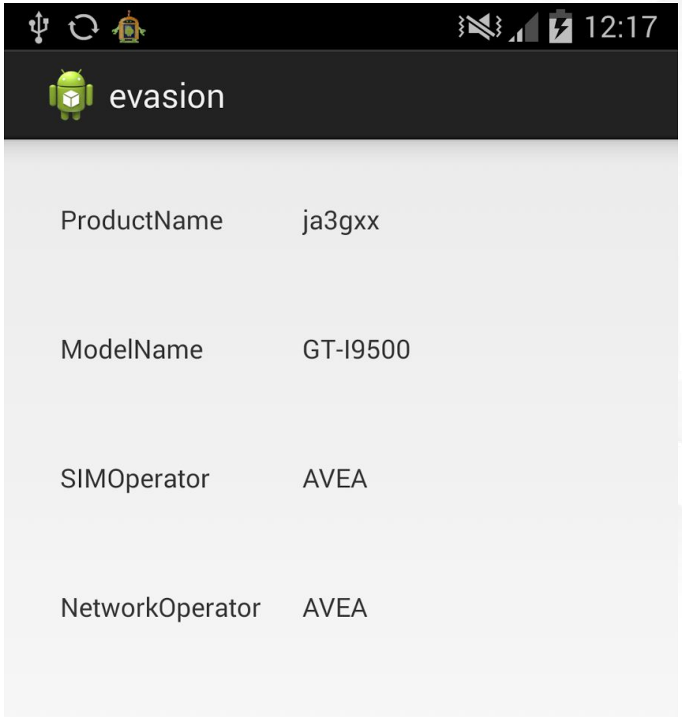
Emulator Glitches

≠



Android Emulator Gliches

Real Device

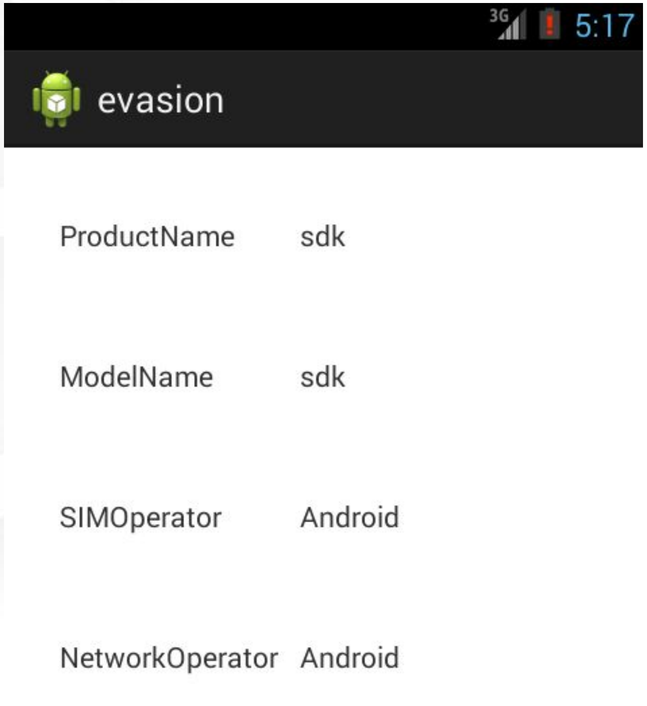


12:17

evasion

ProductName	ja3gxx
ModelName	GT-I9500
SIMOperator	AVEA
NetworkOperator	AVEA

Emulator



5:17

evasion

ProductName	sdk
ModelName	sdk
SIMOperator	Android
NetworkOperator	Android

PaLSIT

Dynamic Analysis example (2)

Analyzer: Google Bouncer / Emulator

Countermeasure: Fingerprinting

Dynamic Analysis example (3)

Analyzer: OpenST/Hardware-Based

Countermeasure: Timing Attack

Goal

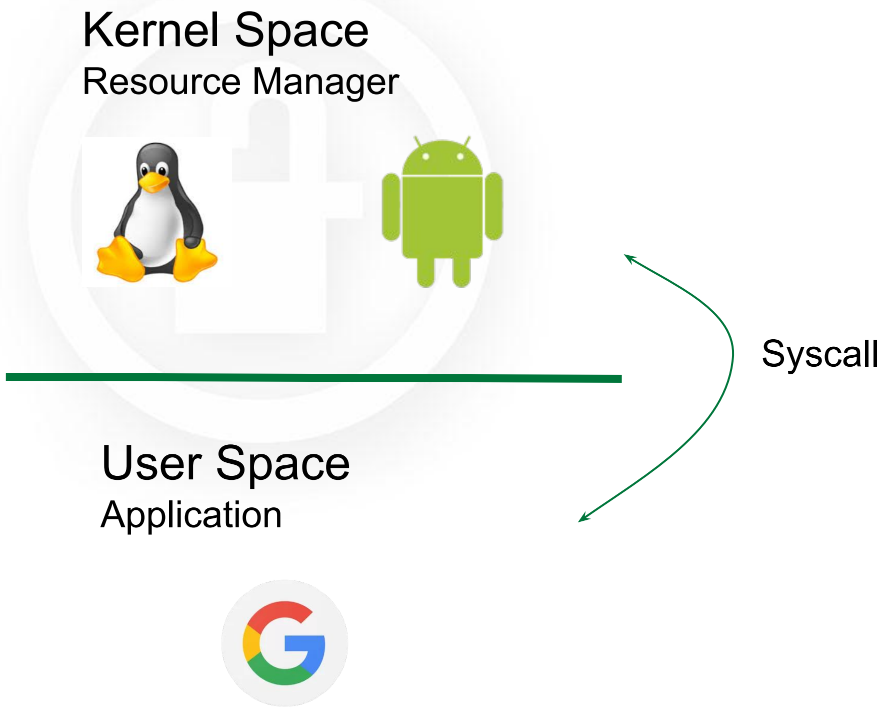
Low artifacts

Highly Transparent

Capture program interaction:

- ❑ operating system procedures
- ❑ network-level events
- ❑ content of memory

How system calls work



Kernel Space

Resource Manager

Syscall

User Space

Application

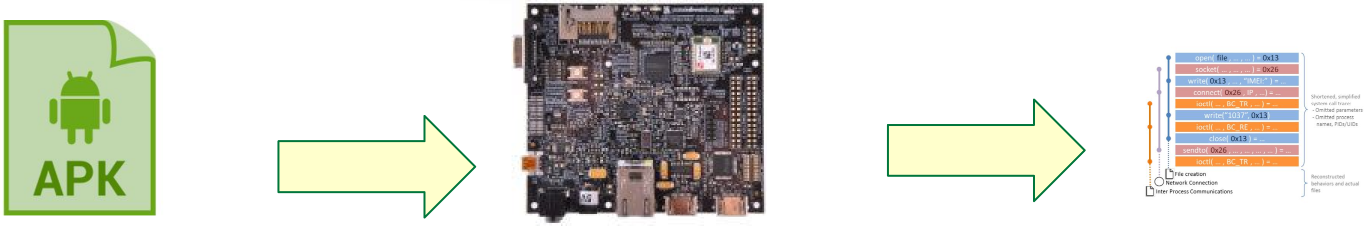
OpenST

External Tracing

- ❑ Instrumentation the target system calls with the breakpoint
- ❑ High overhead
- ❑ High transparent

In-Kernel Tracing

- ❑ Dynamically instrument the kernel at runtime
- ❑ Low overhead
- ❑ Low transparency



```
open( file , ... , ... ) = 0x13
socket( ... , ... , ... ) = 0x26
write( 0x13 , ... , "IMEI:" ) = ...
connect( 0x26 , IP , ...) = ...
ioctl( ... , BC_TR , ... ) = ...
write("1037", 0x13)
ioctl( ... , BC_RE , ... ) = ...
close( 0x13 ) = ...
sendto( 0x26 , ... , ... , ... , ... ) = ...
ioctl( ... , BC_TR , ... ) = ...
```

File creation
Network Connection
Inter Process Communications
Shortened, simplified system call trace:
- Omitted parameters
- Omitted process names, PIDs/UIDs
Reconstructed behaviors and actual files

PaLSIT

External Tracing

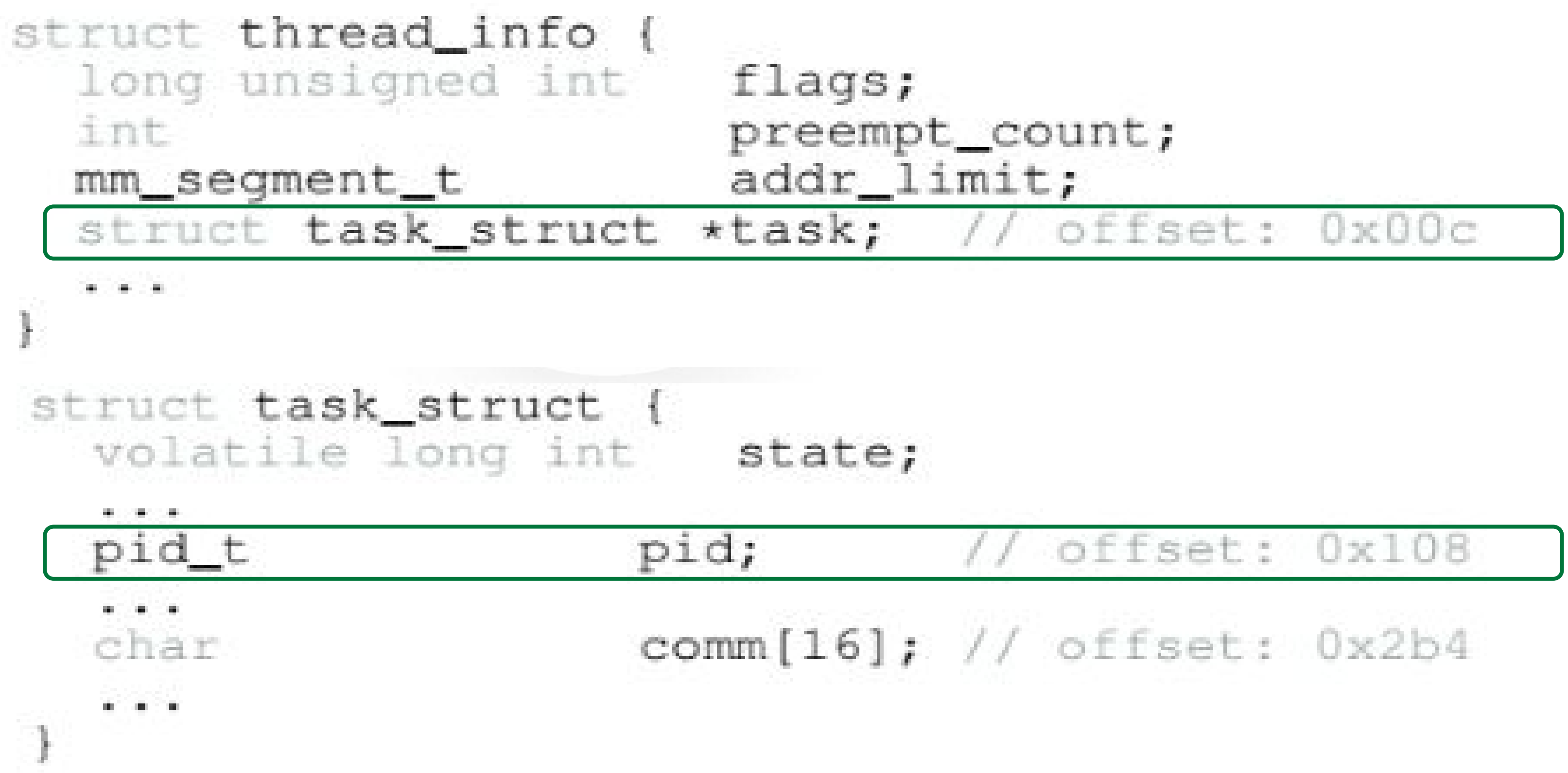
Use JTAG Interface

- ❑ intercept system calls
- ❑ reconstructing its arguments
- ❑ PID of the caller process

Open(AAA,BBB)

Challenge 1: Find PID

Reconstruct kernel data structure



```
struct thread_info {
    long unsigned int        flags;
    int                      preempt_count;
    mm_segment_t             addr_limit;
    struct task_struct *task;   // offset: 0x00c
    ...
}

struct task_struct {
    volatile long int        state;
    ...
    pid_t                 pid;         // offset: 0x108
    ...
    char                  comm[16]; // offset: 0x2b4
    ...
}
```

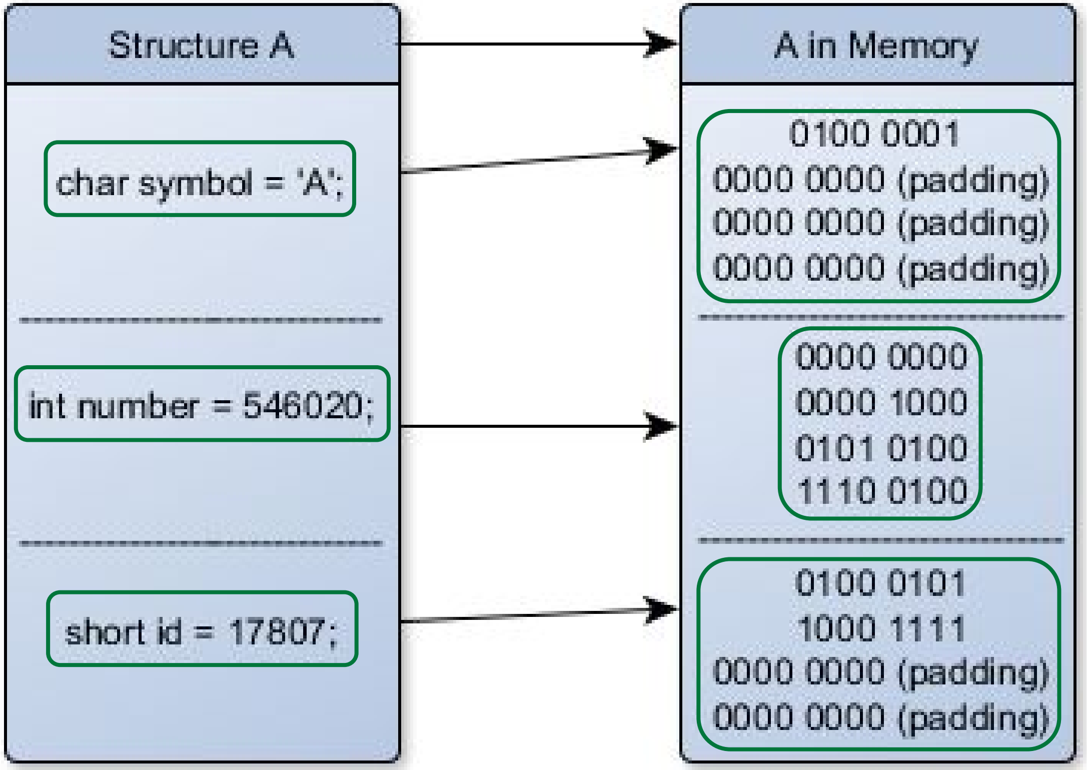
Challenge 2: Reconstruct System Calls

Every architecture have their convention about the location to store the system call arguments.

In Android:

- ❑ Simple argument type are stored in registers
- ❑ Complex arguments type are store in memory

Reconstruct System Calls (2)



Structure A

char symbol = 'A';

int number = 546020;

short id = 17807;

A in Memory

0100 0001
0000 0000 (padding)
0000 0000 (padding)
0000 0000 (padding)

0000 0000
0000 1000
0101 0100
1110 0100

0100 0101
1000 1111
0000 0000 (padding)
0000 0000 (padding)

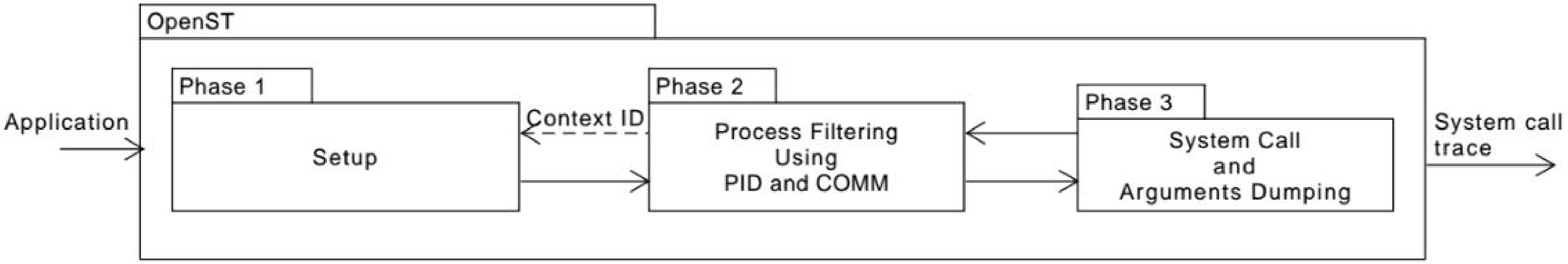
Reconstruct System Calls (3)

Parse the kernel image

❑ Collect data about size and offset

❑ Generate the introspection code

The Architecture



In-Kernel Tracing

Use JTAG Interface

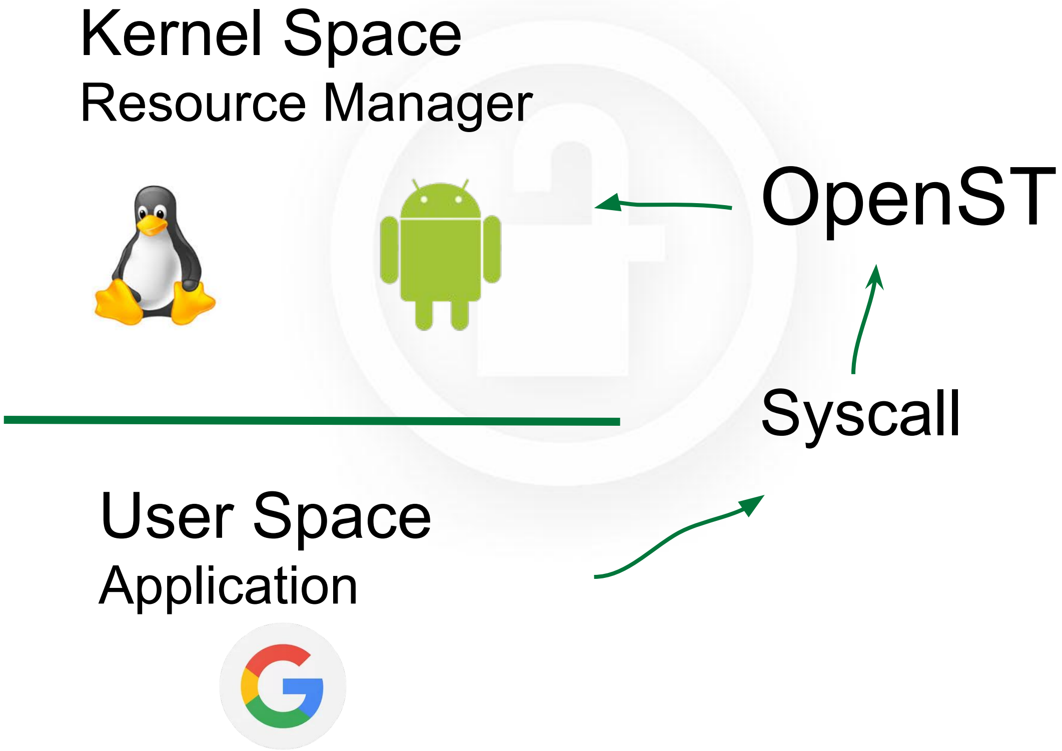
❑ Hot patch the kernel

❑ Trace the system calls and arguments inside the kernel

Patching The Kernel

- ❑ Allocate memory with execution privileges
- ❑ Write that the introspection code
- ❑ Hijack the execution flow (Hooking)

Hooking



Kernel Space

Resource Manager

OpenST

Syscall

User Space

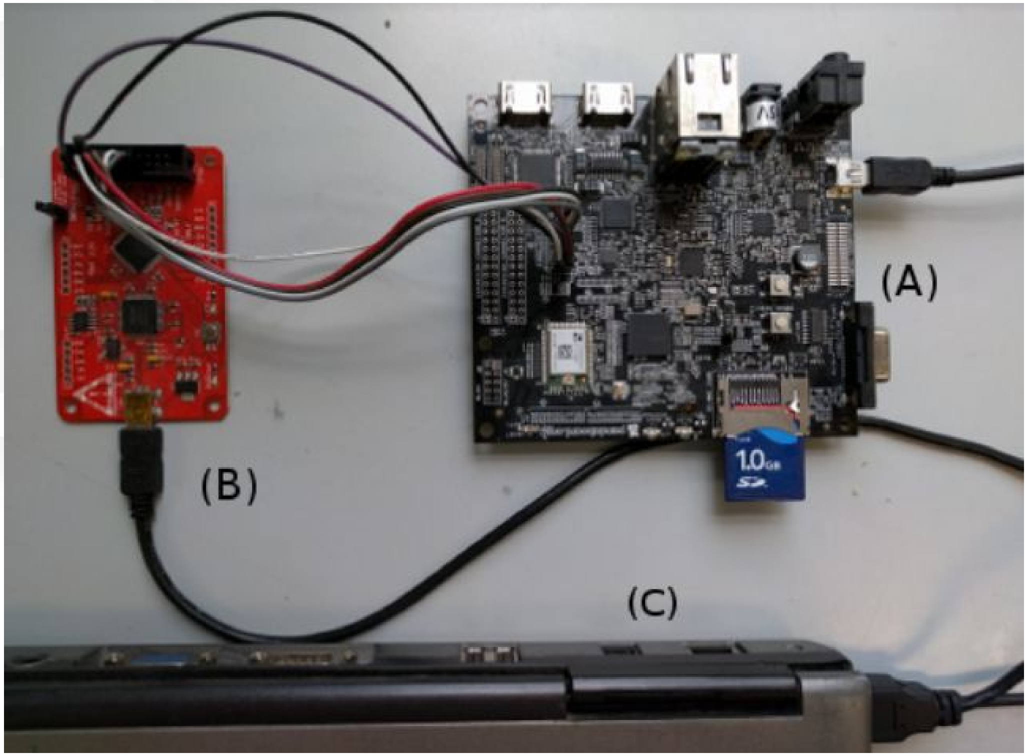
Application

Physical Implementation

A. Dev aBoard

B. JTAG Dev

C. PC



Experiment

Evasion

❑ Android.HeHe (6 variants)

❑ Android.Pincer.A

Performance

❑ Micro Benchmark

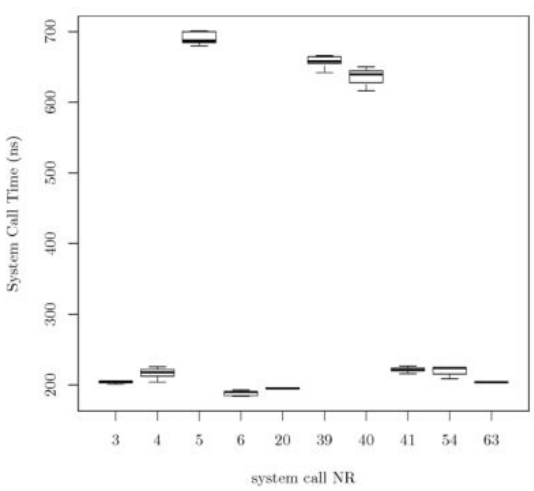
❑ Macro Benchmark

Evasiveness Comparison

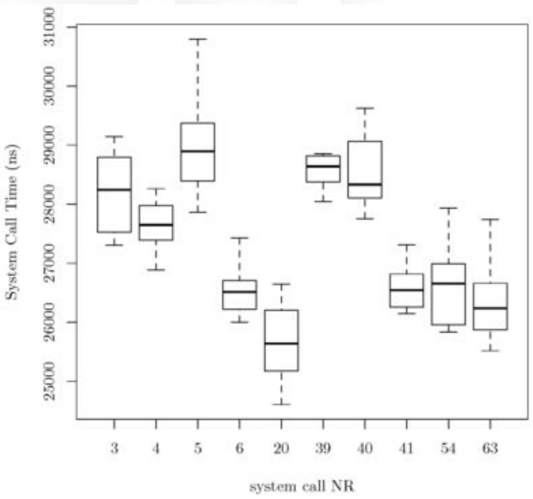
Sample	Emulator (file ops)	OpenST In-Kernel (file ops)	OpenST External (file ops)
Android.HeHe.1	3	475	468
Android.Pincer.A	3	334	334

Micro Benckmark

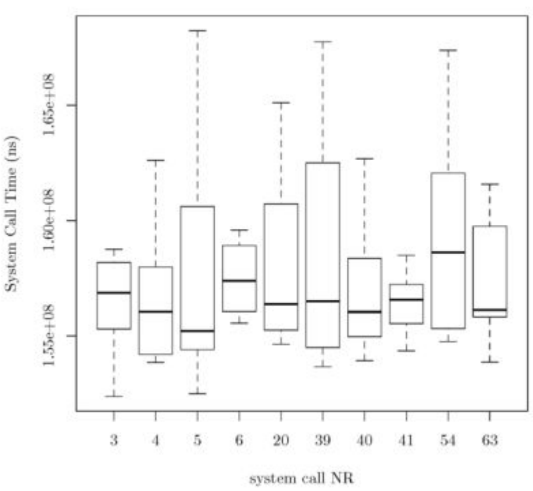
Vanilla



Internal Tracing

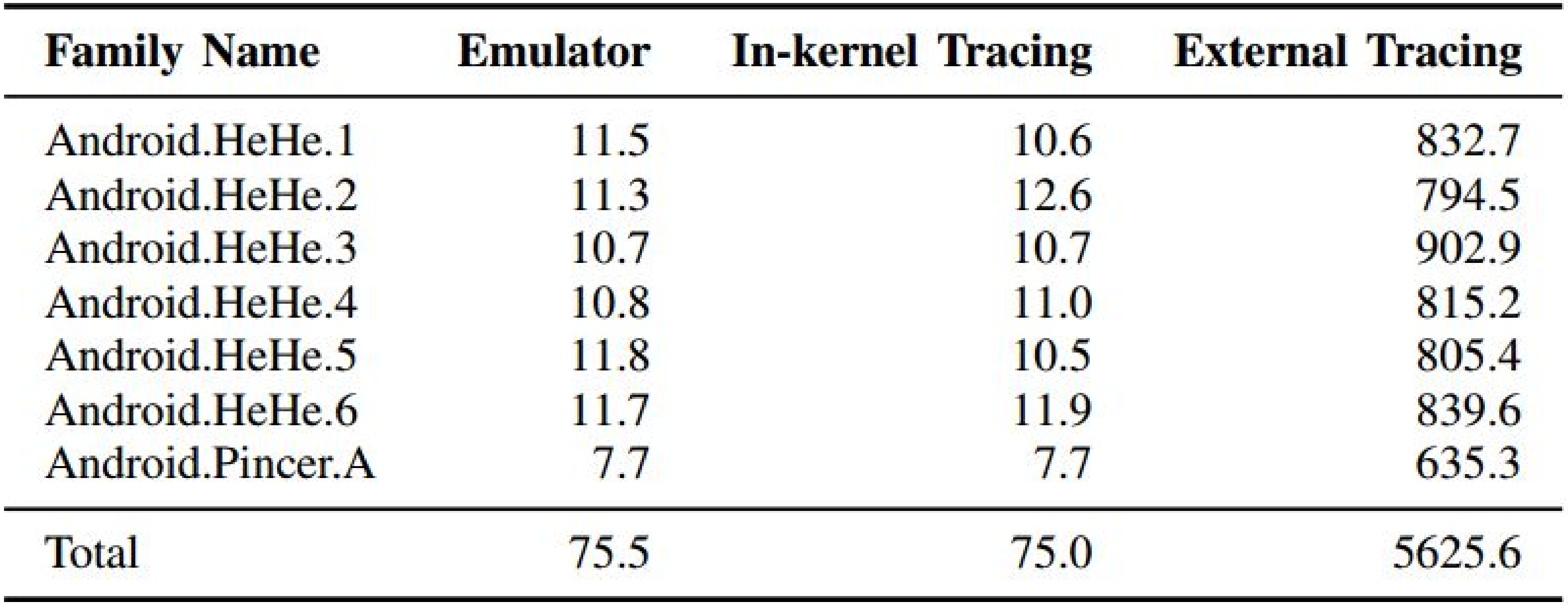


External Tracing



0.2-0.7us x100 25-31us x10,000 155-165ms

Macro Benckmark

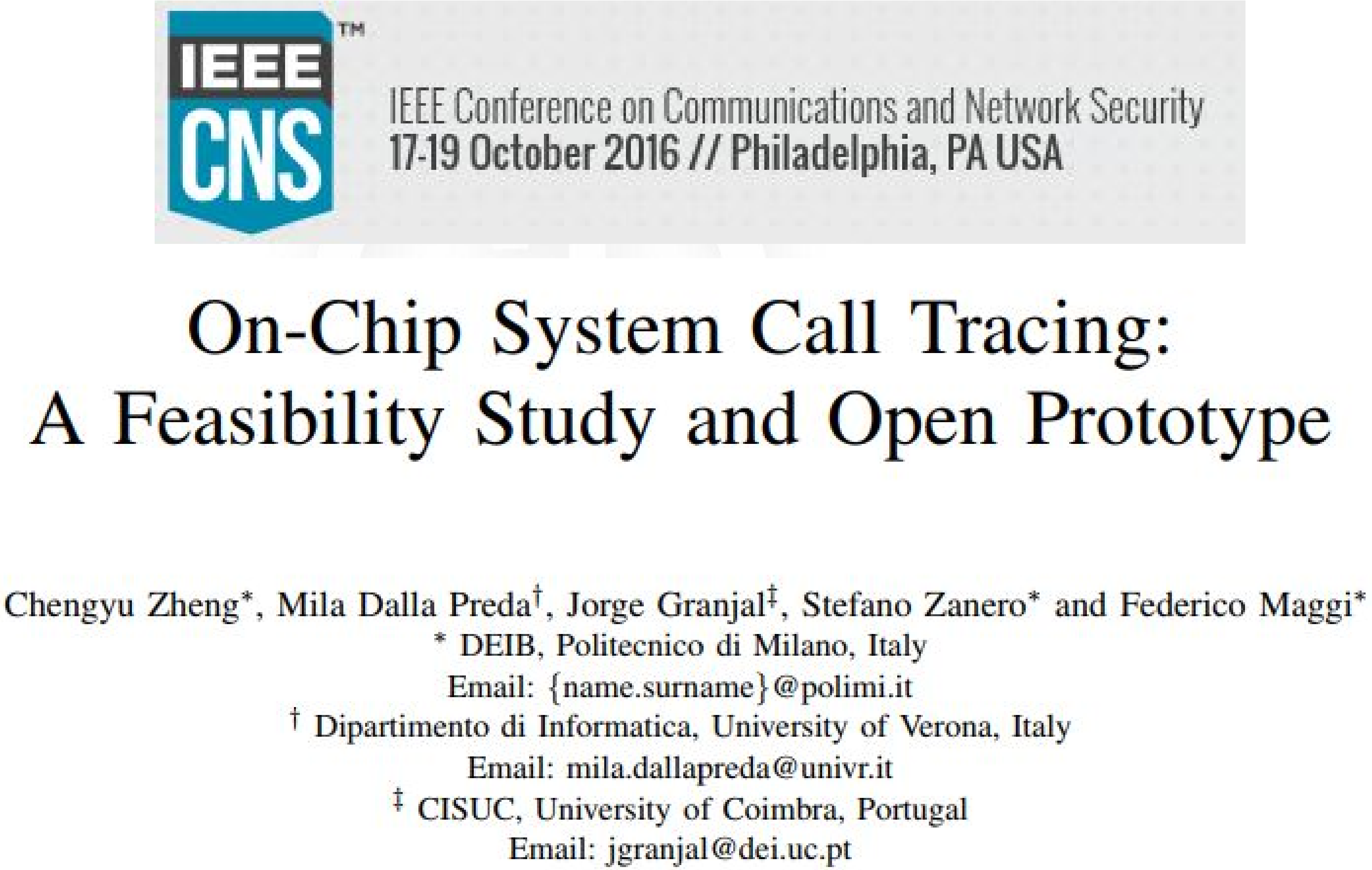


Family Name	Emulator	In-kernel Tracing	External Tracing
Android.HeHe.1	11.5	10.6	832.7
Android.HeHe.2	11.3	12.6	794.5
Android.HeHe.3	10.7	10.7	902.9
Android.HeHe.4	10.8	11.0	815.2
Android.HeHe.5	11.8	10.5	805.4
Android.HeHe.6	11.7	11.9	839.6
Android.Pincer.A	7.7	7.7	635.3
Total	75.5	75.0	5625.6

Future Work

- Use USB emulated digitizer in order to have better code coverage
- Use USB emulated storage to efficiently snapshot

References



IEEE CNS

IEEE Conference on Communications and Network Security
17-19 October 2016 // Philadelphia, PA USA

On-Chip System Call Tracing:
A Feasibility Study and Open Prototype

Chengyu Zheng*, Mila Dalla Preda†, Jorge Granjal‡, Stefano Zanero* and Federico Maggi*
* DEIB, Politecnico di Milano, Italy
Email: {name.surname}@polimi.it
† Dipartimento di Informatica, University of Verona, Italy
Email: mila.dallapreda@univr.it
‡ CISUC, University of Coimbra, Portugal
Email: jgranjal@dei.uc.pt

Conclusion

- Increasing number of malware has forces the security community to use automated analysis tools
- Malware in Android started simple without active measure against analyses
- Evolved with measure against static analyses
- Evolved again to include anti-emulator techniques
- Following this trend we propose OpenST as an hardware based a dynamic analysis tool.