Università
Ca’Foscari
Venezia

Remote Debugging Detection in Android

Graduand Mirco Venerba
Matriculation Number 872653

Supervisor Prof. Paolo Falcarin

Master’s Degree programme in Computer Science

Academic Year 2022-2023

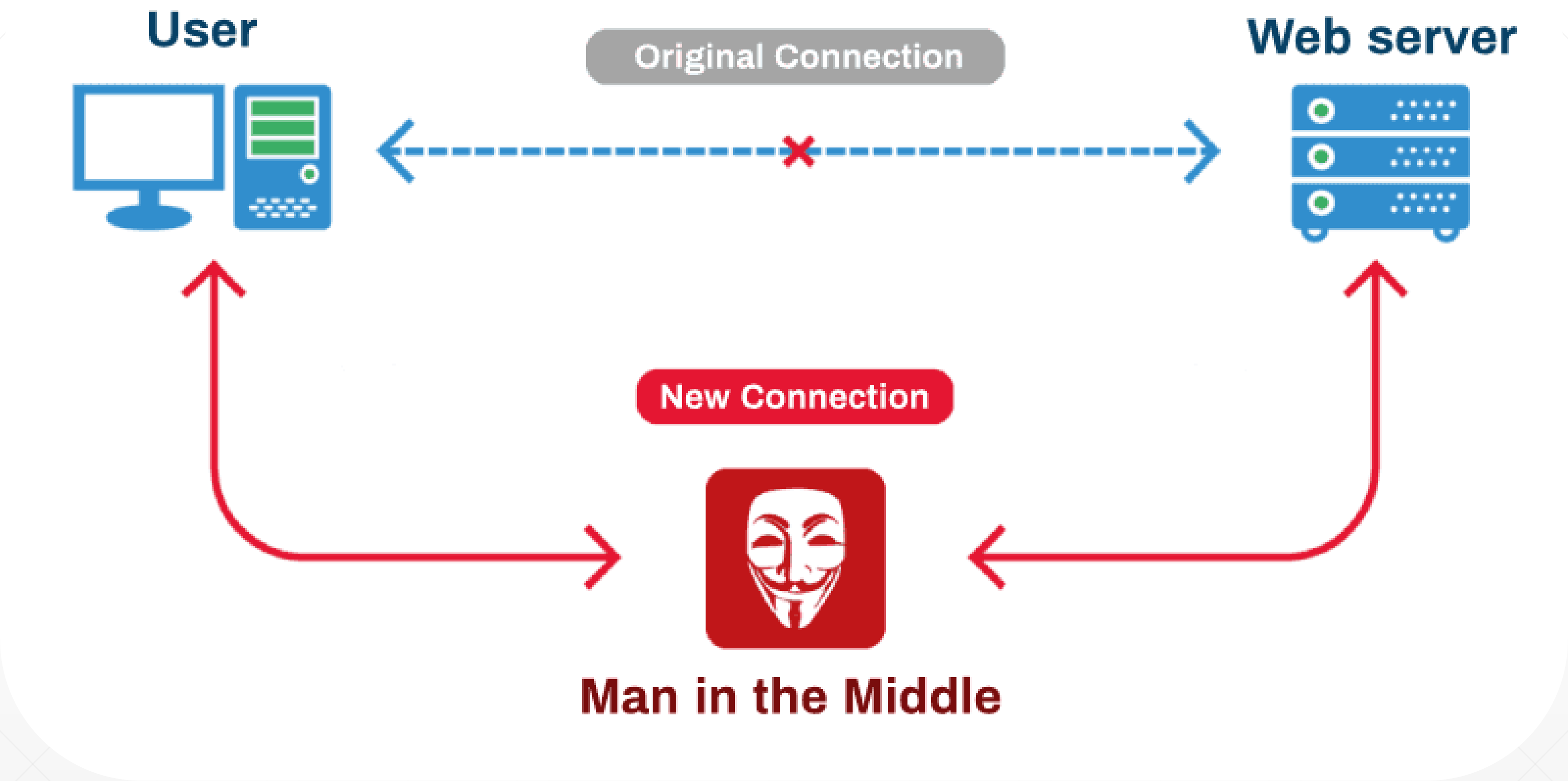
What is the problem?

Store sensitive information

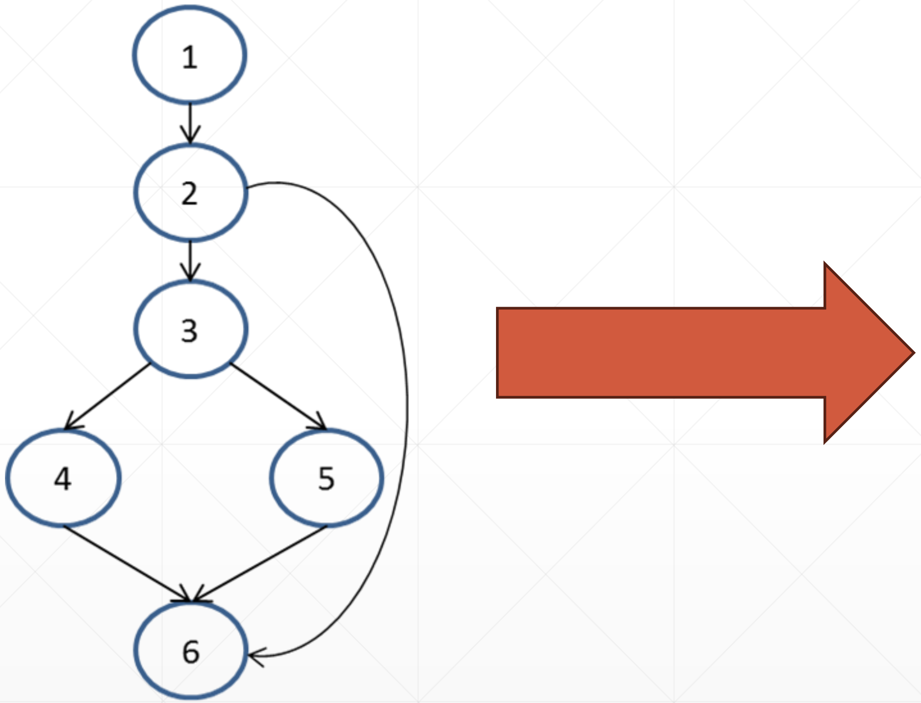
Steal sensitive information

How can this happen?

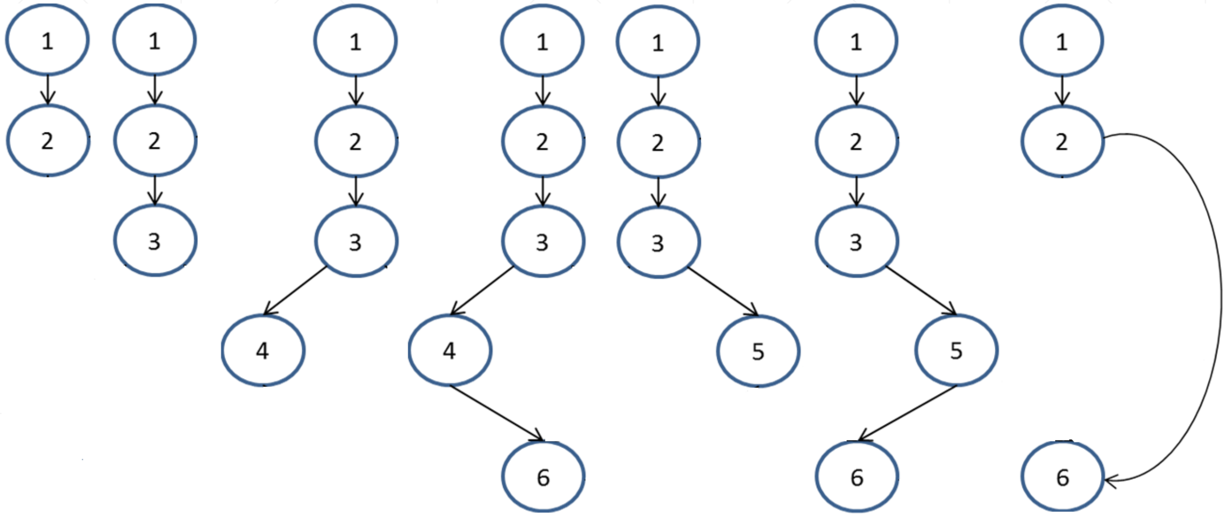
Man-At-The-End (MATE) Attacks



Incremental executions attacks



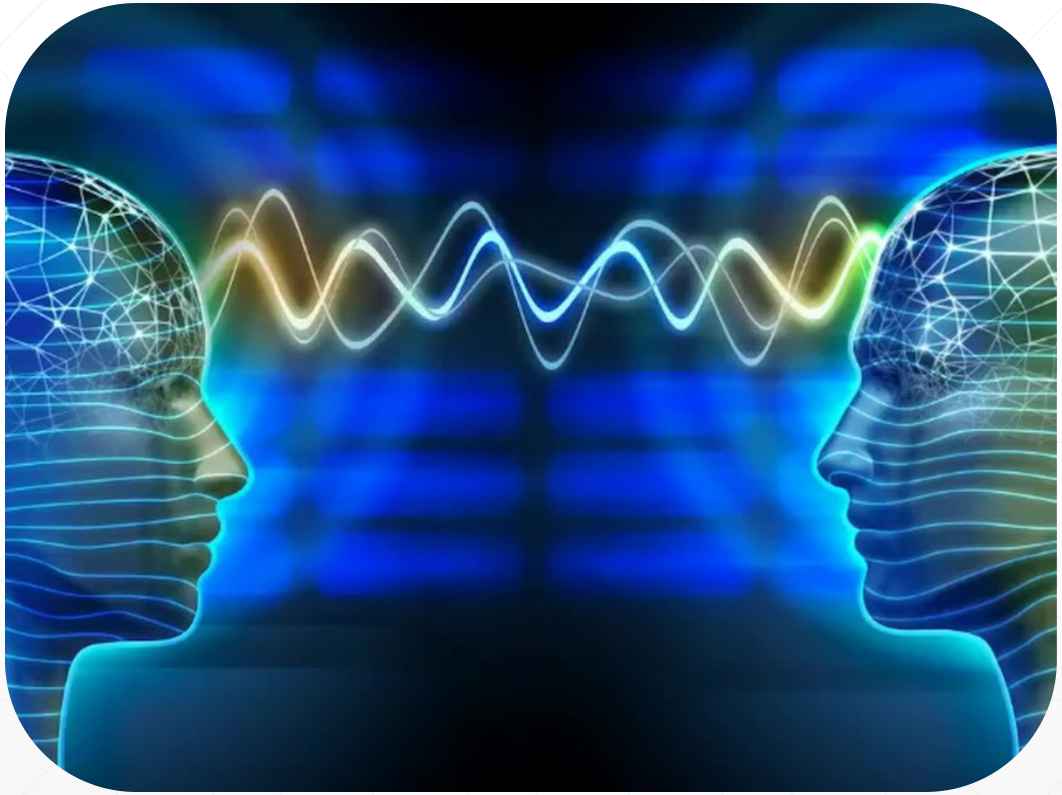
Target program



Incremental executions

What is a possible solution?

Know the attacker in advance



How does an attacker think?

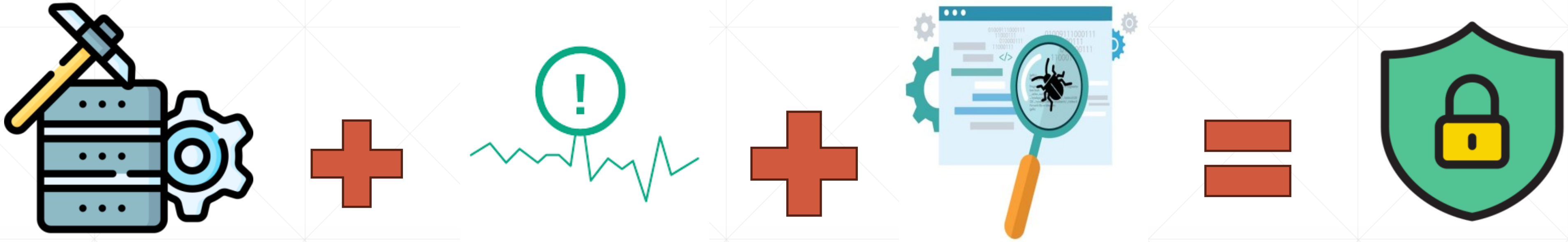
What could the attacker do?

↓

Prevent the attack

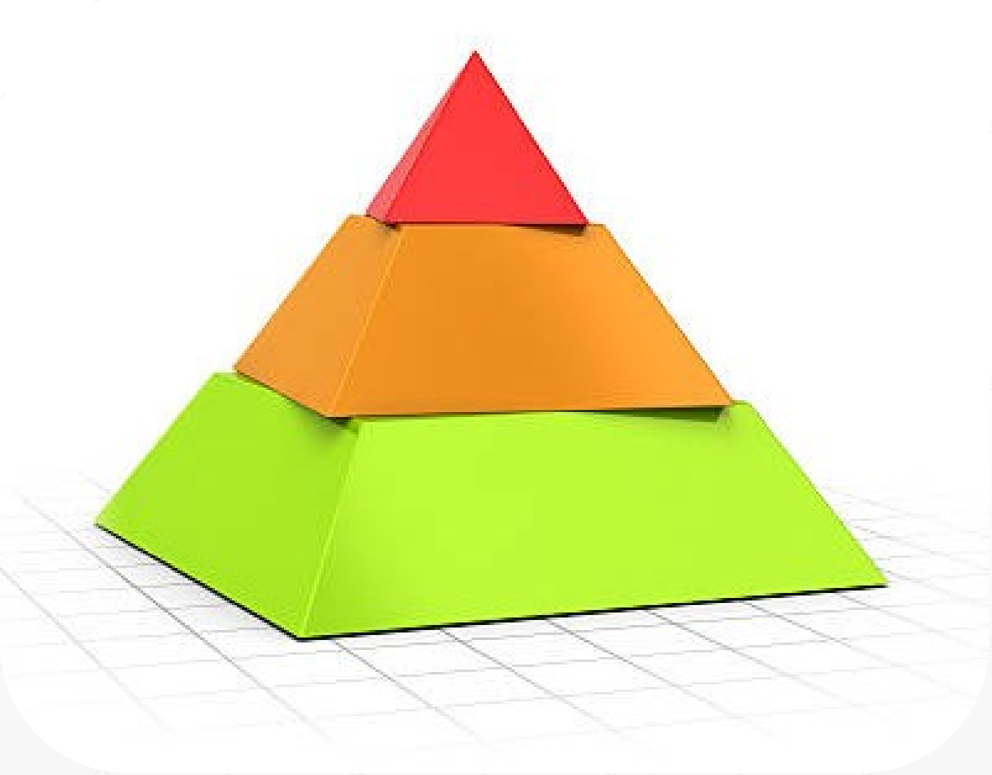
How is it possible?

Components of the process



Data mining + Anomaly detection + Debugger detection = Security level

What types of data are useful?



High level information:
connected real debuggers

Medium level information:
sensors data, sensors alerts, settings, recharge type, debuggable applications, application state in lifecycle

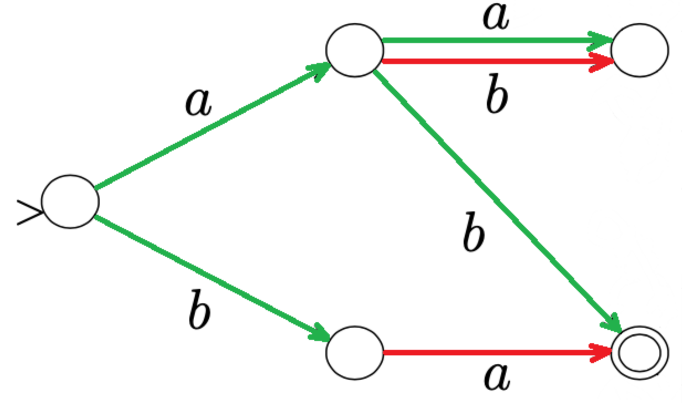
Low level information:
duration of syscall, sequences of syscall, incremental executions path

Why exactly these information?

Some suspicious cases

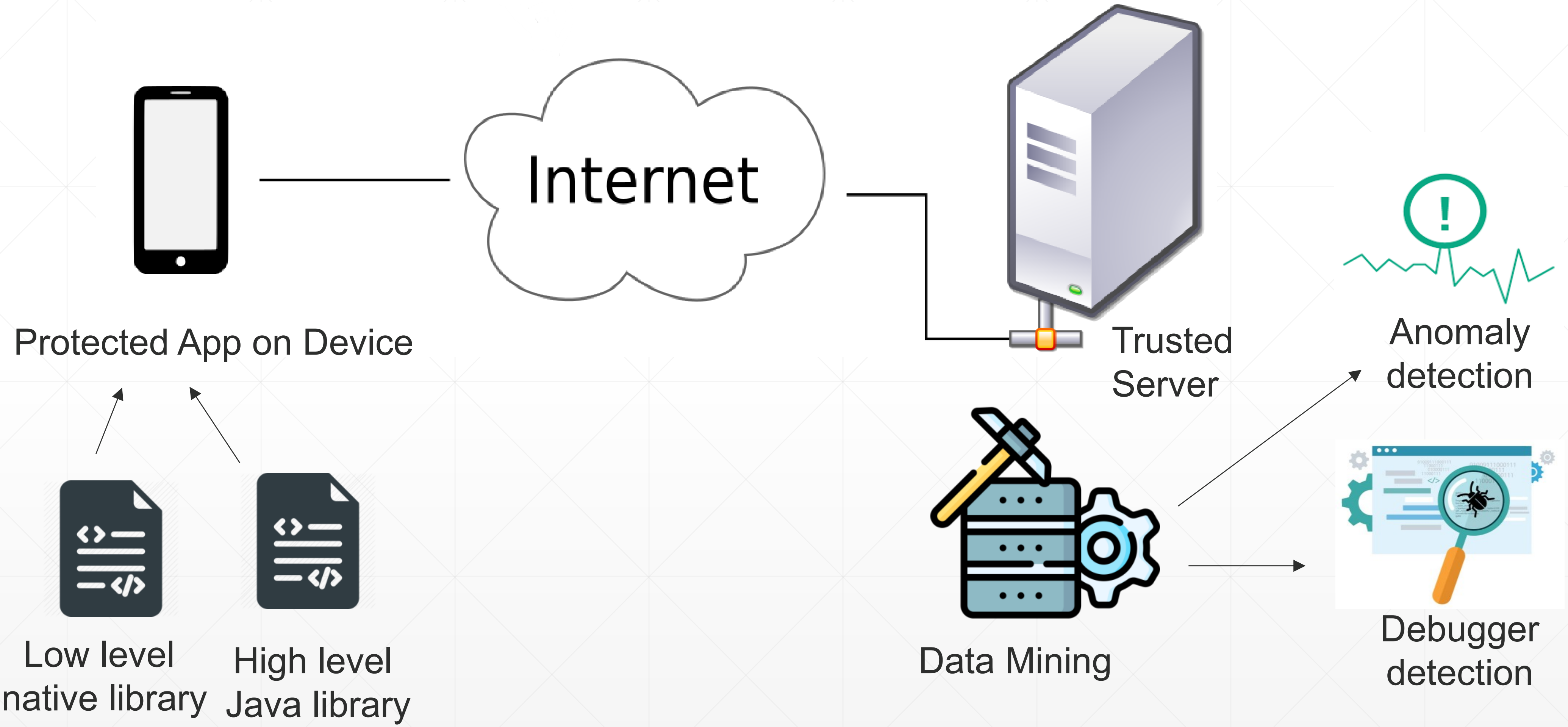
Stationary device

USB connection

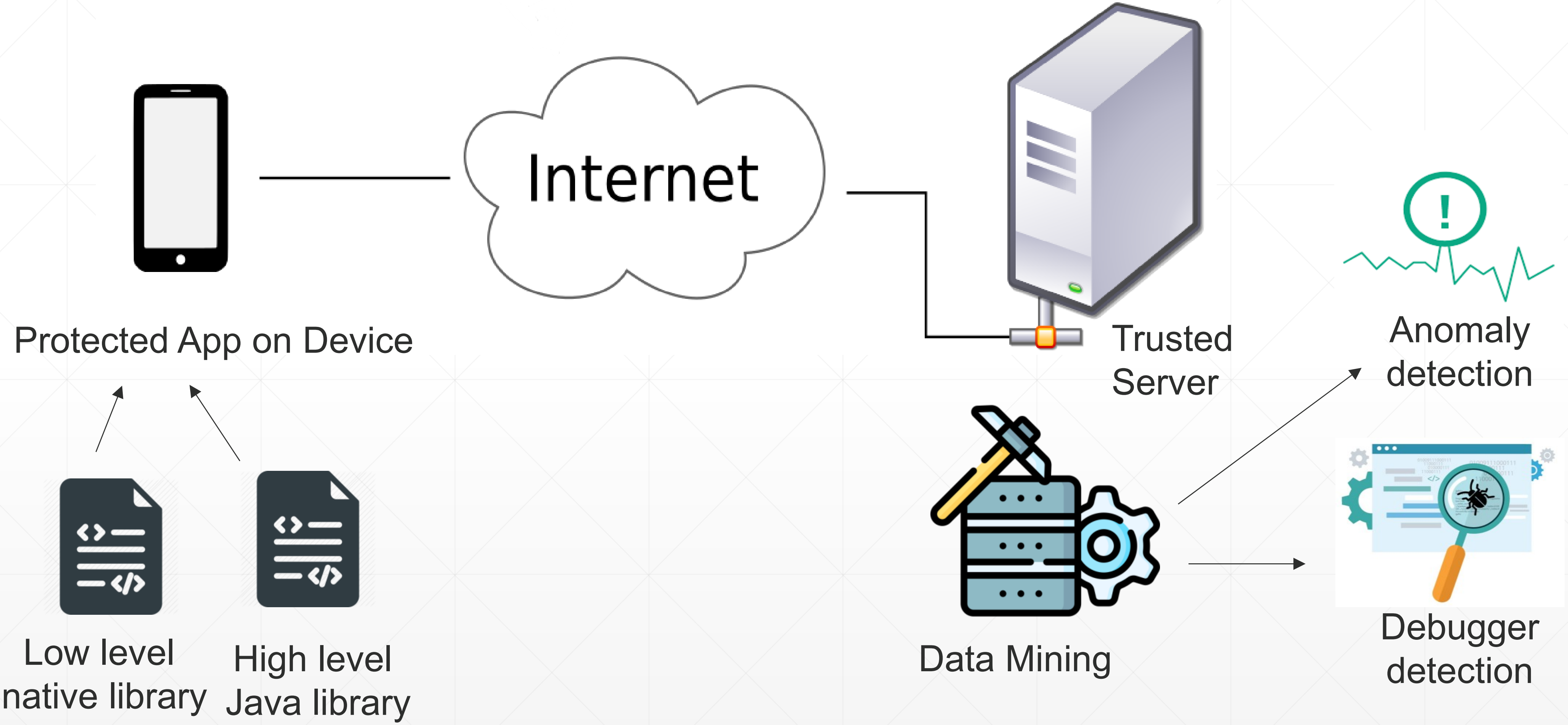


Wrong syscall sequences

Software architecture


Internet
Protected App on Device
Trusted Server
Anomaly detection
Low level native library
High level Java library
Data Mining
Debugger detection

Architecture description



Components description

Server (Python): sensors alerts, statistics of system calls, instruction sequences, incremental execution paths, calculation of device alert level

Low level library (C++): system calls, sequences of syscall, incremental executions path, thesis project of another classmate

High level library (JAVA): debuggers, sensors data, settings, recharge type, debuggable applications

Types of execution logs

C++ library logs

Syscall started

```
------------------ SYSCALL ENTRY START ------------------
Notification origin: it.mircovenerba.audiorecorder
PID: 14711 SPID: 14711
Timestamp: 1673561963897680
Syscall = ioctl (29)
Stack unwinding = {
                PC 0x000079d54b34d8 … - startRecording(..)
                PC 0x000079d546f2f4 … - talkWithDriver(..)
                PC 0x000079ca76e7ac … - main(..) ……}
Parameters = {
                0x000000000000003c
                0x00000000c0306201
                0x0000007fcfedfb68…}
Registers = {
                PC: 0x00000079d54b34d8
                SP: 0x0000007fcfedfa50
                RET: 0x000000000000003c…}
------------------ SYSCALL ENTRY STOP ------------------
```

Syscall finished

```
------------------ SYSCALL EXIT START ------------------
PID: 18480
SPID: 18480
Timestamp: 1684069975351329
Return value: 0000000000000000
------------------ SYSCALL EXIT STOP ------------------
```

Types of extracted data for each syscall

- timing
- system call number
- stack trace
- parameters
- cpu registers

Java library logs – first part

Debuggable applications	Recharge type	Developer options
Debuggable flag enabled in the manifest file	Check when the recharge type is USB	Check if they are enabled
LIST OF DEBUGGABLE APPLICATIONS Application Name: AudioRecorder Application Name: Progetto Android	LIST OF CHARGING RECORDS Charging record Is charging: True Usb charging: True Ac charging: False Start timestamp: 1684070992389 Finish timestamp: 17893546284049 Charging record Is charging: False Usb charging: False Ac charging: False Start timestamp: 17893546284050	LIST OF DEVELOPER OPTIONS RECORDS Developer options record Developer options: True Android debug bridge: True Start timestamp: 1684070992362

Java library logs – second part

Debuggers	App lifecycle	Sensors data
Check when a debugger is attached	Check when the application is running	Check if each position is valid or not
LIST OF ALL DEBUGGERS Debugger Name: GDB debugger Found: False Debugger Name: JDWP debugger Found: True Timestamp: 1684070992389	LIST OF ALL LIFECYCLE RECORDS Lifecycle record On resume: True On pause: False Start timestamp: 1684071051553 Finish timestamp: 1891736453297 Lifecycle record On resume: False On pause: True Start timestamp: 1891736453298	LIST OF ALL SENSOR NUMBER RECORDS Sensor record Azimuth value: 0 Pitch value: 2 Roll value: 0 Start timestamp: 1684070994605 Finish timestamp: 1684070994606 Sensor record Azimuth value: 0 Pitch value: 2 Roll value: 359 Start timestamp: 1684070994607

Web server logs – first part

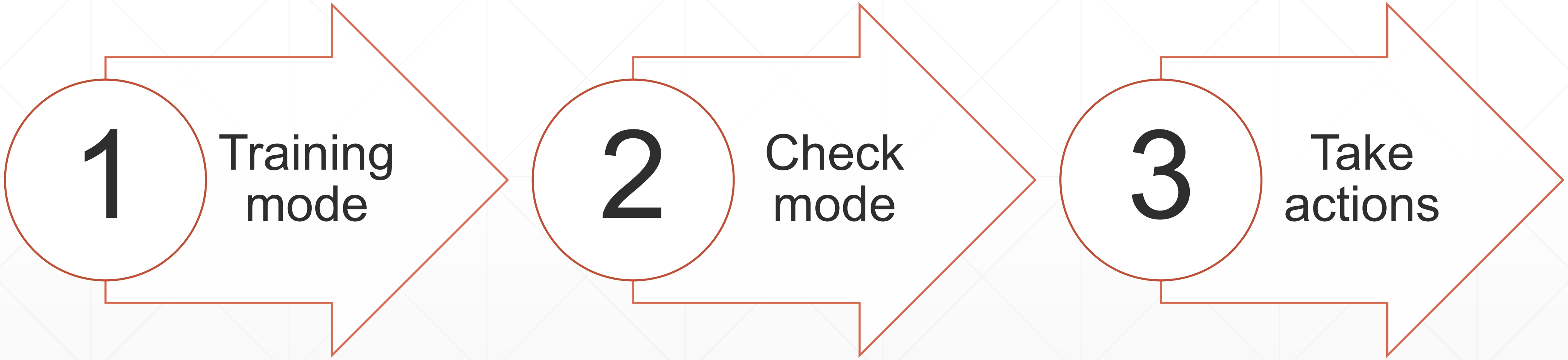
Instructions analysis	Instructions sequence	Simple syscalls sequences
Count instructions with longer duration		(pid, spid)
ANALYSIS OF INSTRUCTIONS Instruction Name: clock_gettime Minimum duration: 182 Maximum duration: 5436 Average duration: 829.66 Number measurements: 78 List measurements: [1854, 857, 801...] Instruction Name: close Minimum duration: 353 Maximum duration: 353 Average duration: 353 Number measurements: 1 List measurements: [353]	LIST OF INSTRUCTIONS Instruction Name: clock_gettime Pid: 20403 Spid: 20403 Status: Finished Start: 1684070932532139 Finish: 1684070932533993 Duration: 1854 Return Value: 0 Instruction Name: clock_gettime Pid: 20403 Spid: 20403 Status: Not finished Start: 1684070932533994	LIST OF SYSCALLS SEQUENCES Sequence (20403, 20403) -> ['clock_gettime', 'clock_gettime', 'gettimeofday', 'write', 'sendto', 'recvfrom', 'clock_gettime', 'clock_gettime', 'gettimeofday', 'read',….] Sequence (20403, 20404) -> ['getuid', 'epoll_pwait', 'clock_gettime', 'clock_gettime', 'gettimeofday', 'read', 'clock_gettime', 'read', 'getuid', 'epoll_pwait',…]

Web server logs – second part

Possible next instruction	Incremental execution path	Alert level device
Check if the sequence is valid or not	Check how actual path is % equal of last one	Alert level from 0 to 10, 9 / 10 device blocked
LIST OF POSSIBLE INSTRUCTIONS Instruction Name: clock_gettime Next: [clock_gettime, epoll_pwait, ..] Instruction Name: epoll_pwait Next: [fcntl, gettimeofday, getuid, ioctl, …]	LIST OF FOUND SUBSEQUENCES SUBSEQUENCE Portion of code: 8.928571428571429% Subsequence: ['recvfrom', 'recvfrom', 'clock_gettime', 'epoll_pwait', ...] SUBSEQUENCE Portion of code: 12.5% Subsequence: ['futex', 'clock_gettime', 'clock_gettime', 'futex', 'clock_gettime', ...]	Device Ip address: 192.168.1.3 Security level: 7 Good things: C++ library started Debugger not found Subsequences not found Bad things: Debuggable applications Developer options Charging type USB Instruction more duration many times Sequence not secure many times Sensor alerts many times Stationary device many times

Validation and testing

Process stages



Web server – Training mode

Model = all the possible combinations that identify normal, or rather ideal, behavior of an Android application

Instruction level
clock_gettime has longer duration: 612, now is mapped
epoll_pwait has longer duration: 1345, now is mapped
recvfrom has longer duration: 438, now is mapped
recvfrom has new minimum duration: 177 vs 438->438
clock_gettime has new minimum duration: 237 vs 612->612
clock_gettime has new maximum duration: 879 vs 237->612
gettimeofday has longer duration: 176, now is mapped

Sequence level
Insert recvfrom -> clock_gettime sequence? [yes/no] yes
Insert clock_gettime -> clock_gettime sequence? [yes/no] yes
Insert clock_gettime -> gettimeofday sequence? [yes/no] yes
Insert gettimeofday -> write sequence? [yes/no] yes
Insert write -> clock_gettime sequence? [yes/no] no
Insert gettimeofday -> read sequence? [yes/no] no
Insert read -> clock_gettime sequence? [yes/no] yes

Web server – Check mode

Found debuggable application
Security level of device = 1

Found developer options enabled
Security level of device = 2

Device stationary

Bad position

Found invalid sequence: recvfrom -> epoll_pwait

socket has longer duration: 6230 vs 213->4937

Bad position

Found USB charging type
Security level of device = 3

clock_gettime has longer duration: 16228 vs 176->14562

gettimeofday has longer duration: 21512 vs 145->7662

Longer duration many times
Security level of device = 4

Device stationary

Found subsequence

Found invalid sequence: getuid -> gettimeofday

Found invalid sequence: ioctl -> epoll_ctl

Insecure sequence many times
Security level of device = 5

Bad position

Device stationary

Bad position

Bad position many times
Security level of device = 6

openat has longer duration: 6707 vs 354->5938

socket has longer duration: 5544 vs 213->4937

Device stationary

Device stationary many times
Security level of device = 7

Found invalid sequence: futex -> write

ioctl has longer duration: 9449 vs 225->3346

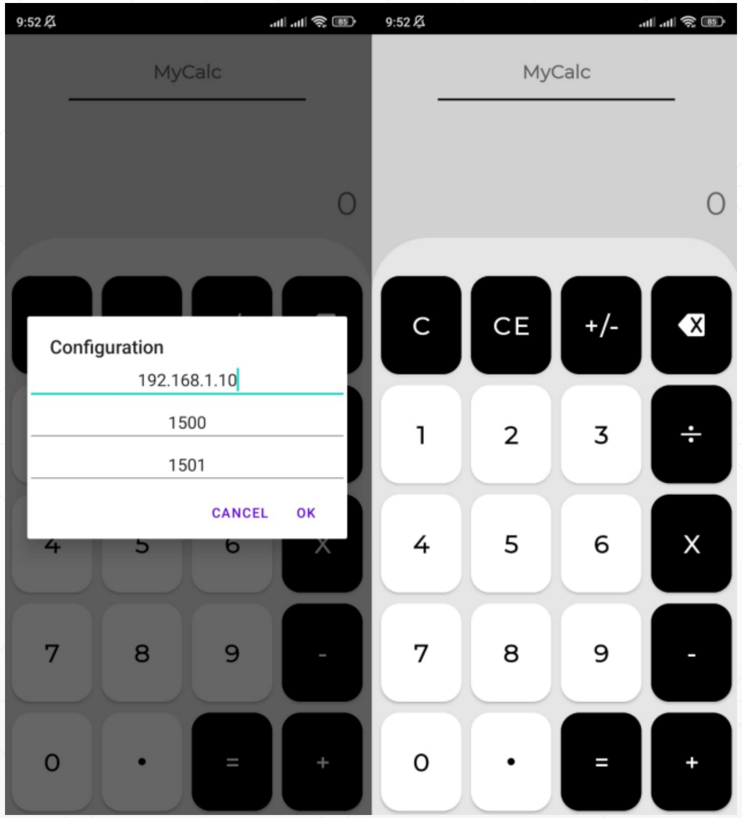
A jdwp debugger is found
Security level of device = 10, device blocked

Examples of next actions

In relation of the **alert level of the device**

- keep the connection enabled for safe device
- keep warning devices always monitored
- put in a blacklist all blocked devices
- possibly block the connection for less secure devices

Test applications



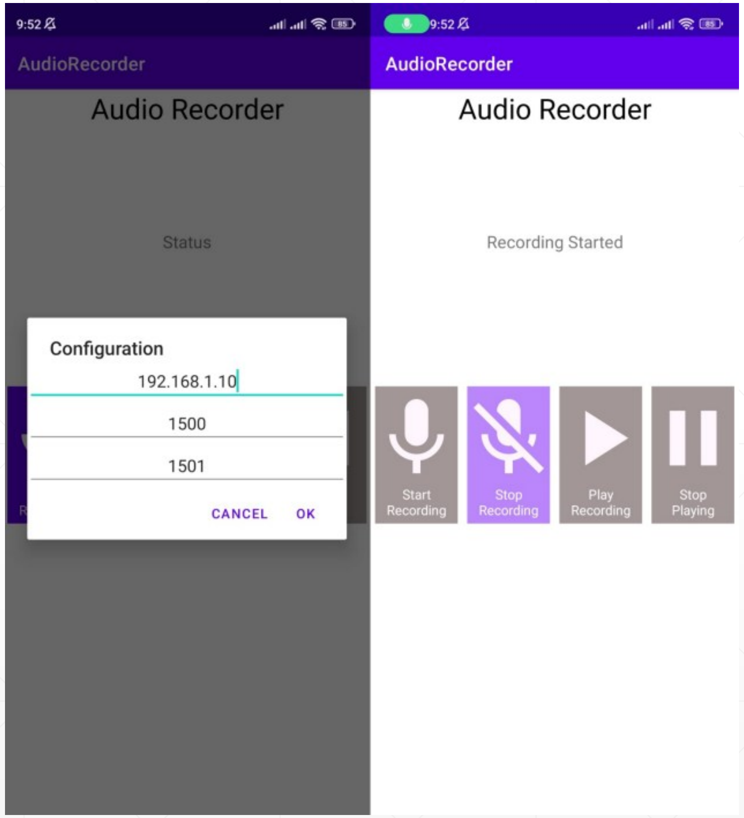
Configuration

192.168.1.10

1500

1501

Calculator



Configuration

192.168.1.10

1500

1501

Audio recorder

Future developments

Some future developments

- Unique identifier of the device
- Reprojection all functionality in the same and native package
- In-depth study of settings file to understand which is the most suitable combination for all configuration variables
- Improvement of the training phase using machine learning algorithms
- Implementation of 1 to n model in the web server

Thanks for the attention

Related work

- MATE attacks
  - Obfuscation, e.g. the modification of the code / flow of execution
  - Device attestation, e.g. device properties verified and not counterfeit
  - Dynamic code renewability, e.g. a tool chain capable to modify code
  - Self debugging, e.g. connected debugger to occupy the unique free position
- Anomaly detection
  - Static analysis (ProfileDroid), e.g. code that acquire and communicate sensitive data
  - Dynamic analysis (TaintDroid), e.g. various levels of abstraction
  - Download external resources (DroidTrace), e.g. check resources from a remote server
  - Sandboxing (Aurasium), e.g. application inside a bubble