SpotCheck:
On-Device Anomaly Detection for Android

Mark Vella & Christian Colombo
L-Università ta' Malta

SINCONF 2020

This project has received financial support from the European Union Horizon 2020 Program under grant agreement no. 832735

LOCARD

Overview

Problem:

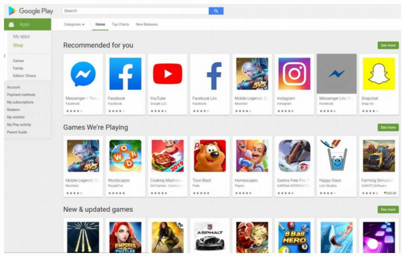
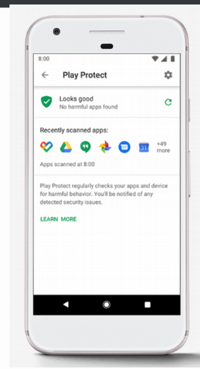
Mobile devices are increasingly targeted by malware, posing privacy and financial threats. App store and on-device scanning are however limited mainly due to signature-based detection.

A novelty detection layer is needed.

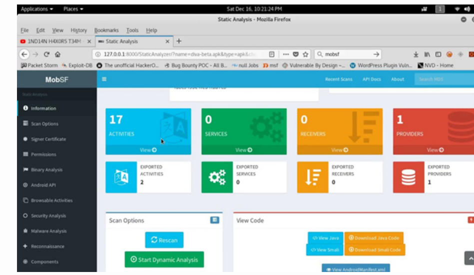
Contributions:

1) Re-purposing of Kernel Principal Component Analysis (KPCA) and Variational Autoencoders (VAE), as used for network anomaly detection (AD), for Android AD
2) A novel process memory dump approach, from which to derive app behavior, as compared to a system-call-trace baseline
3) Openly available datasets capturing benign/malicious app behaviour for both representations

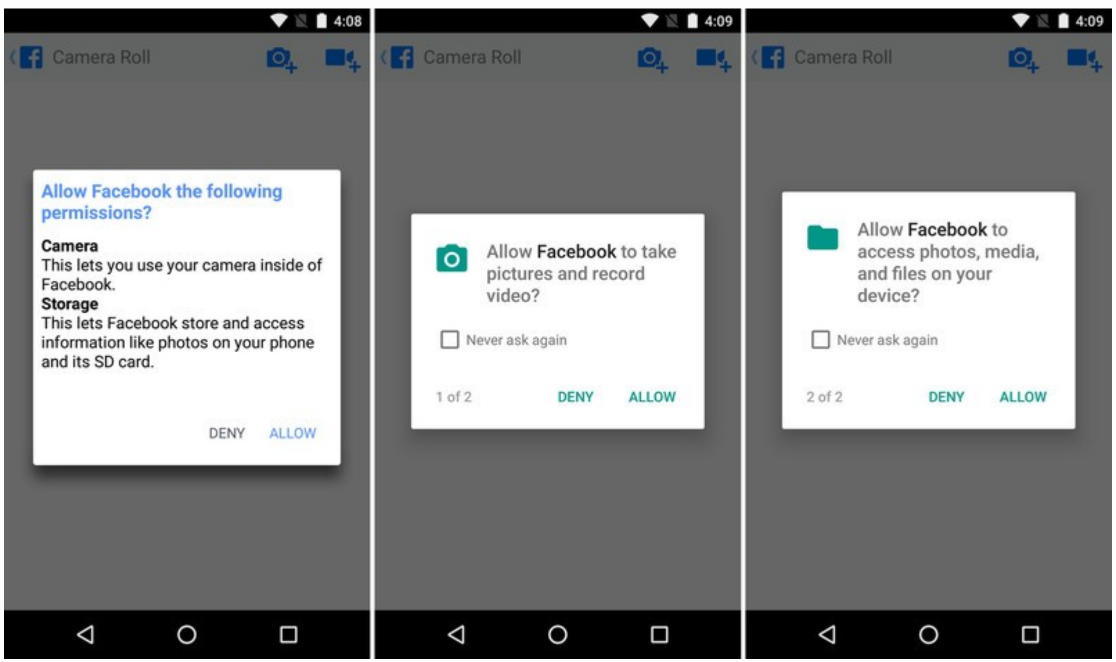
State-of-the-art

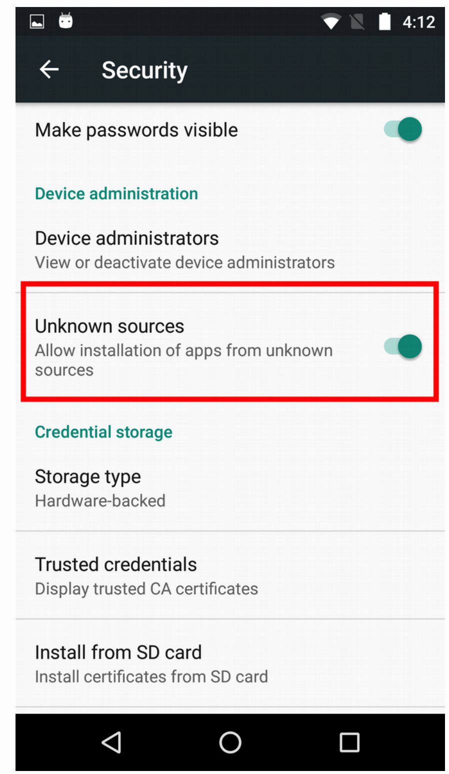
App store & on-device protection

Static & dynamic analysis



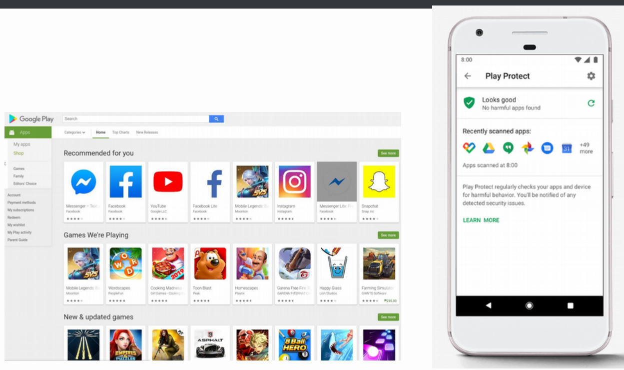
Stringent permission granting



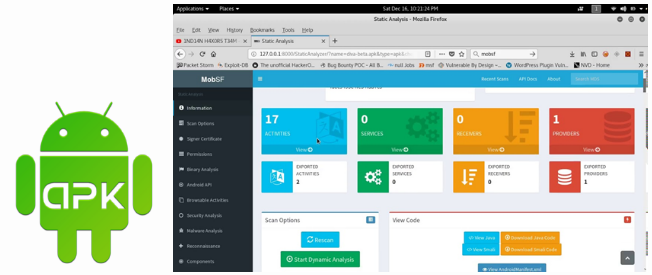
Discourage unknown sources

LOCARD

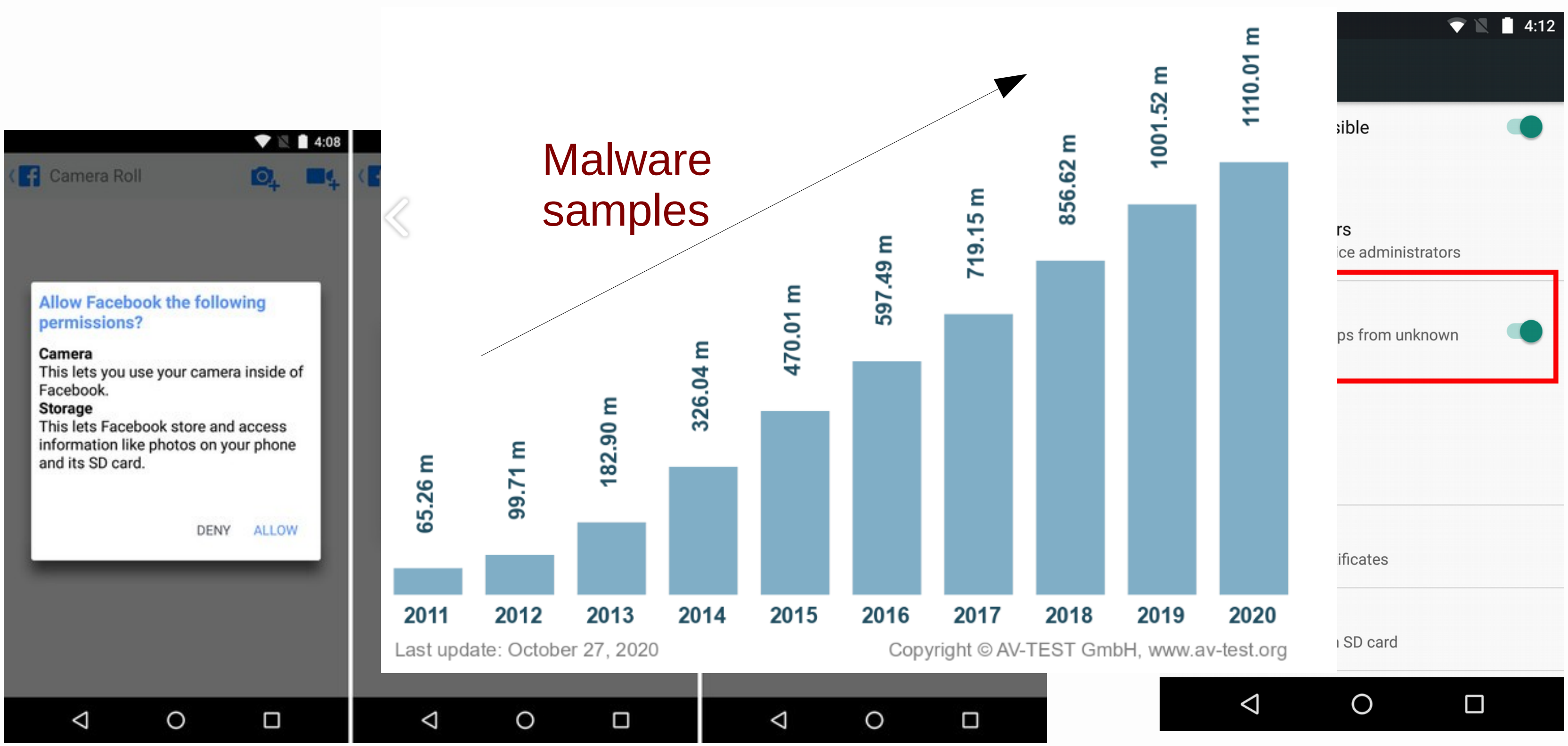
State-of-the-art



App store & on-device protection



Static & dynamic analysis

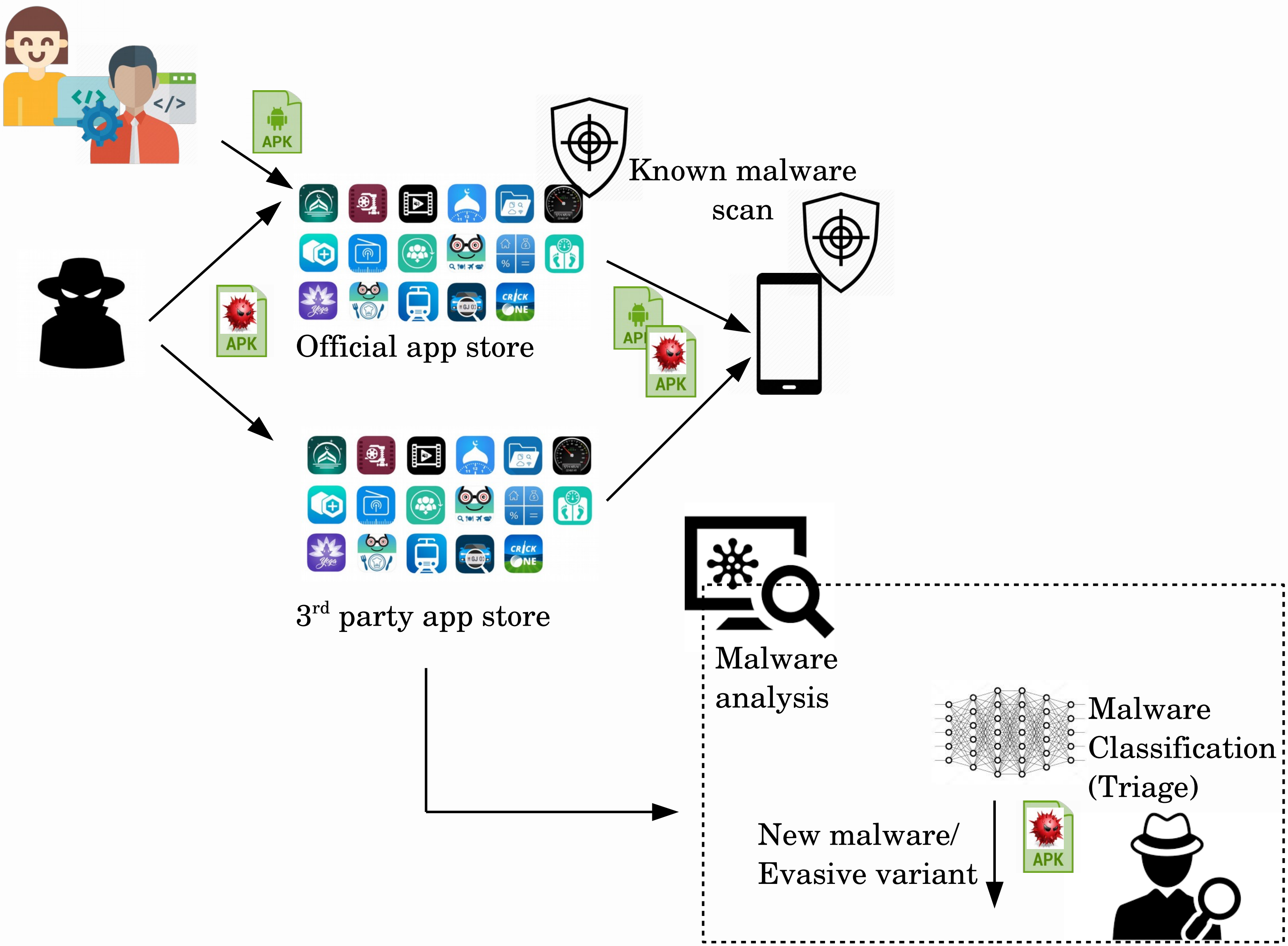


Stringent permission granting

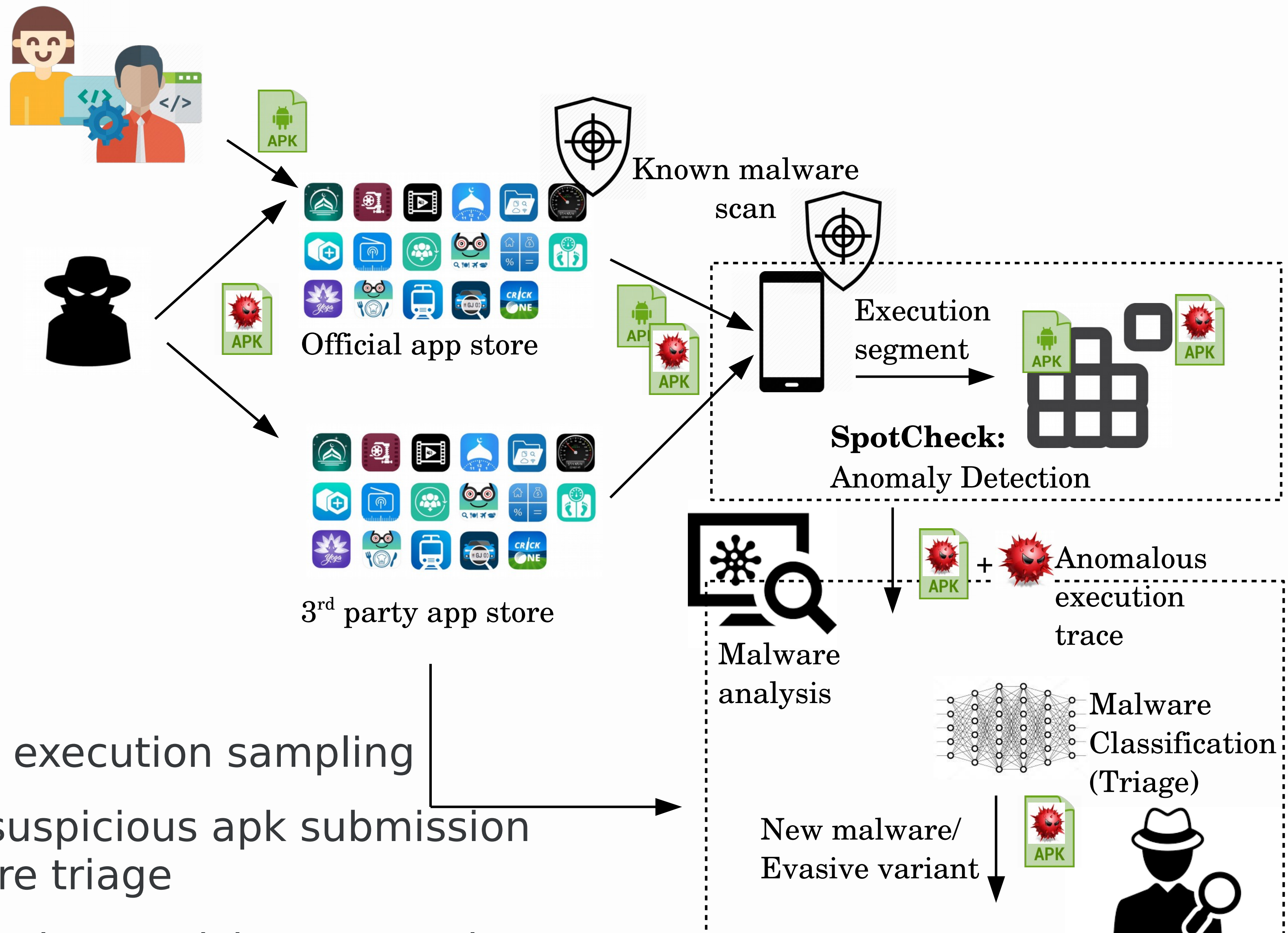
Discourage unknown sources

LOCARD

SpotCheck


APK
Known malware
scan
Official app store
APK
3rd party app store
Malware
analysis
Malware
Classification
(Triage)
New malware/
Evasive variant
APK

SpotCheck



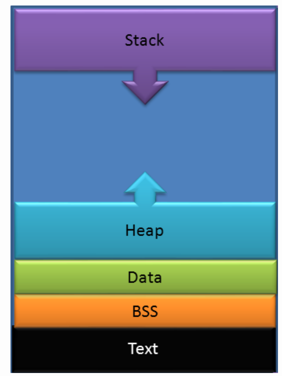
Approach:

- On-device execution sampling
- Prioritize suspicious apk submission for malware triage
- Along with the suspicious execution trace

App behavior representation i/ii


```
function_1()
function_2()
function_3()
...
function_n()
```

(Function) Call trace



Process memory

(exec side-effects)

- Process memory approach:
  - Less invasive but represents only the residue of execution – time-critical

App behavior representation ii/ii

- Call trace
  - Linux system call histogram
  - Successfully used for malware classification
  - In-line hooking on non-rooted devices is possible

$$x \stackrel{def}{=} < accept, access, bind, chdir, ..., writev >$$

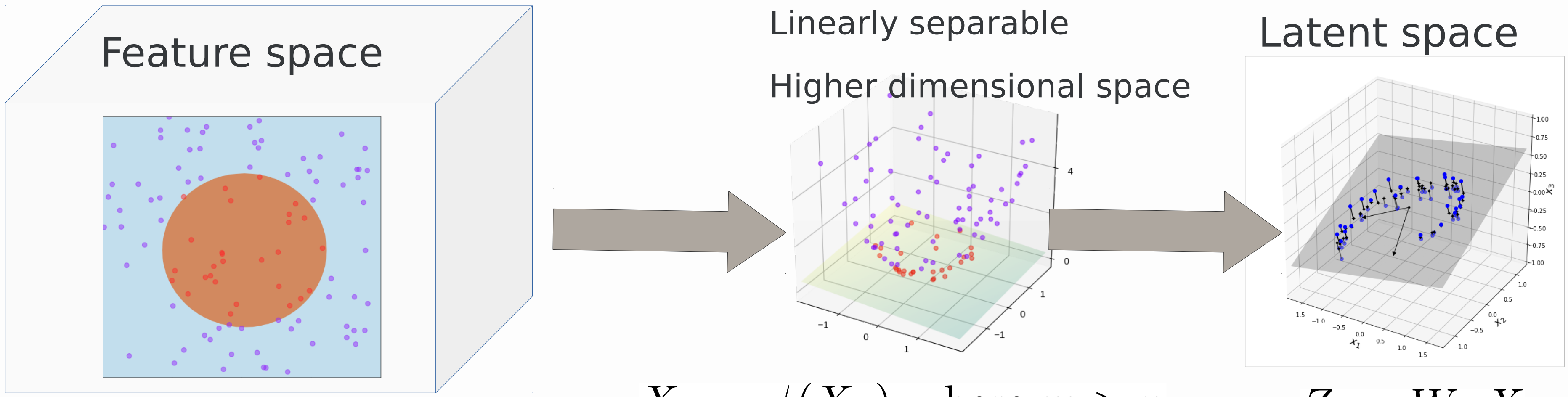
- Process memory dump
  - `android.content.Context.getSystemService()` manager class histogram
  - HPROF – with `android.os.Debug.dumpHprofData()`
  - `ArtMethod→data_` patching possible On non-rooted devices is possible

$$x \stackrel{def}{=} < AccessibilityManager, AccountManager, ..., WindowManager >$$

- Normalization

$$\hat{x} \stackrel{def}{=} < a_i/||x||_1, ..., a_n/||x||_1 > \qquad (||\hat{x}||_1 = 1)$$

KPCA-based AD



$$X_m = \phi(X_n), \text{where } m > n \qquad Z_2 = W_2.X$$

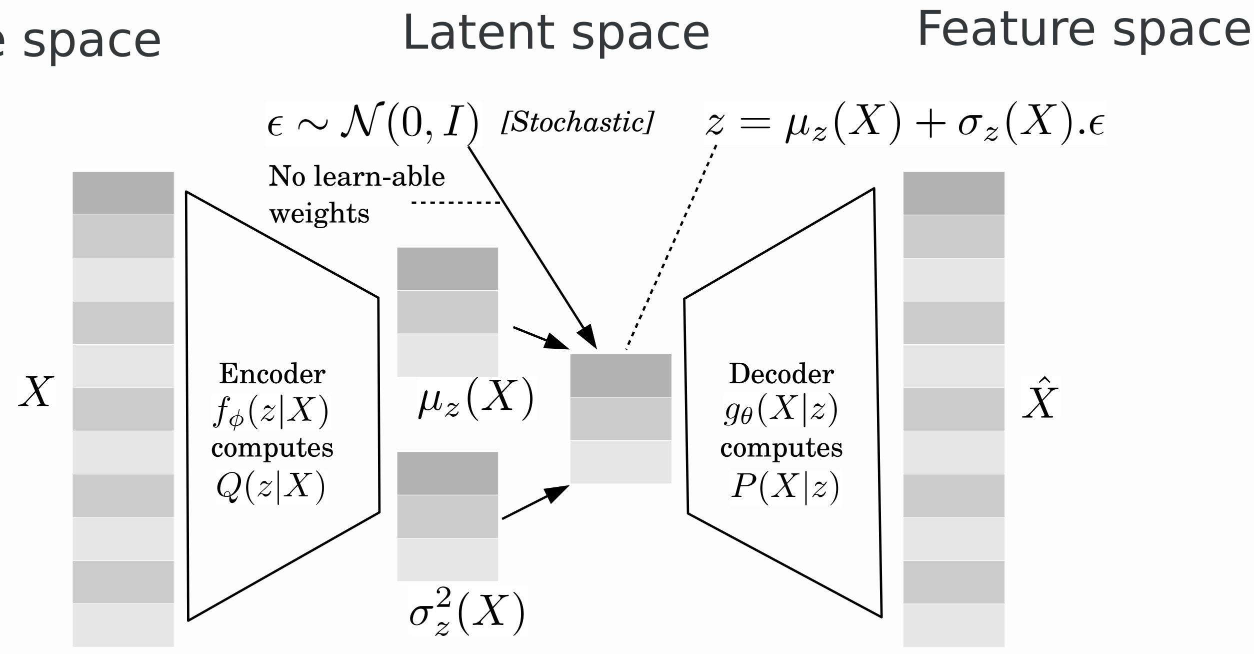
$$X_n$$

$$\phi(X^T).\phi(X) = k(X) \overset{eigendecomp.}{\rightarrow} W.\lambda.W^{-1}$$

$$\text{where } k(x,y) \overset{def}{=} e^{-\gamma||x-y||^2}, \text{where } \gamma = \frac{1}{2\sigma^2} > 0$$

- Premise for AD
  - For learned: γ, W_2
  - The lossy inverse transform $X_n = Z_2.W_2^T$ minimizes reconstruction error only in the case datapoints are from the same distribution of X
  - Returns a higher reconstruction error otherwise

VAE-based AD



Loss: $\mathcal{D}_{KL}(Q(z|x^{(i)})||\mathcal{N}(0, I)) - \mathbb{E}_{Q(z|x^{(i)})}[\log P(x^{(i)}|z)]$

for $x^{(i)} \in X$

- Premise for AD
  - For learned ϕ, θ: $\hat{x}^{(i)}$ is similar to $x^{(i)}$ but only if $x^{(i)}$ is derived from $P(X)$
  - Similarity defined in terms of a reconstruction probability

$$P(x^{(i)}) \leftarrow \frac{1}{L}\sum_{l=1}^{L} P(x^{(i,l)}; \mu_{\hat{x}^{(i,l)}}, \sigma^2_{\hat{x}^{(i,l)}})$$

Dataset

Google Play

VIRUSTOTAL

2K + 1K apps

https://github.com/mmarrkv/spotcheck_ds

FRIDA

MAT

Exerciser Monkey

dumpsys

calllog

HPROF

CSV

LOCARD

Results i/ii

Dataset / AUC ROC	KPCA	VAE*
Android AD (calllog)	0.708	0.694
Android AD (HPROF)	0.69	0.712
NSL-KDD (DoS)	0.59	0.795
NSL-KDD (Probe)	0.821	0.944
NSL-KDD (R2L)	0.712	0.777
NSL-KDD (U2R)	0.712	0.782

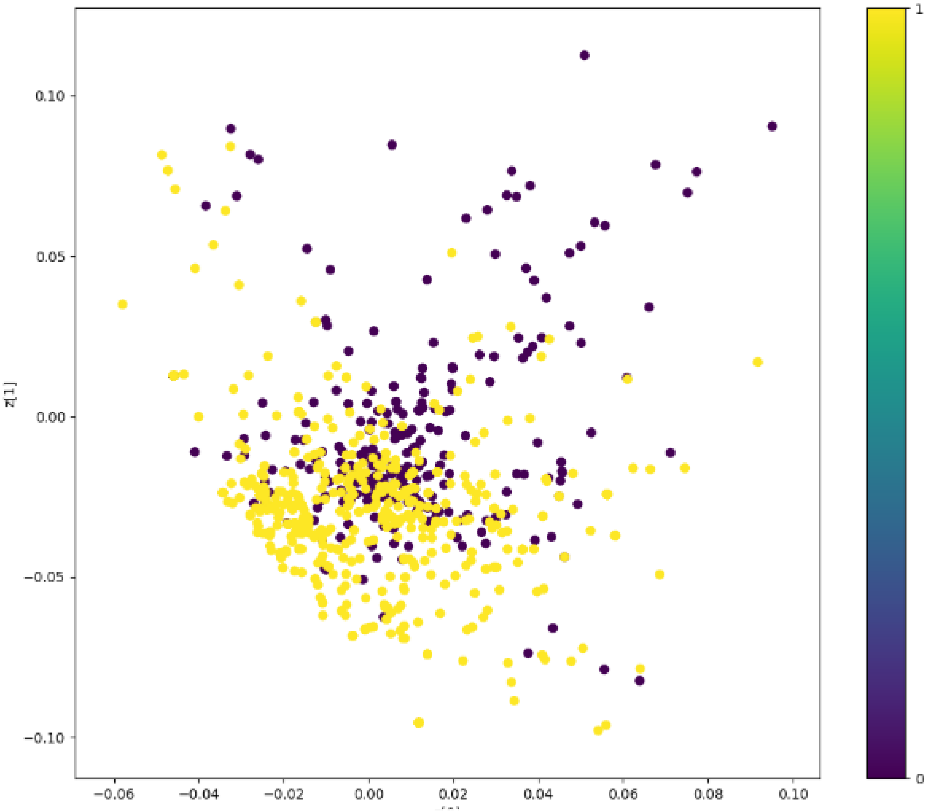
- Successful re-purposing from network AD (An & Cho, 2015)
  - Note: Probe is particularly noisy on the network level
- KPCA-HPROF
  - F1/recall/pres – **0.88/0.97/0.8**
  - *Note 1*: 0.2 imprecision results in benign apps being sent for malware triage, rather than apps being immediately flagged as malicious
  - *Note 2*: 0.03 non-recalled malware could in reality be offset by considering multiple execution samples in a multi-device deployment setting

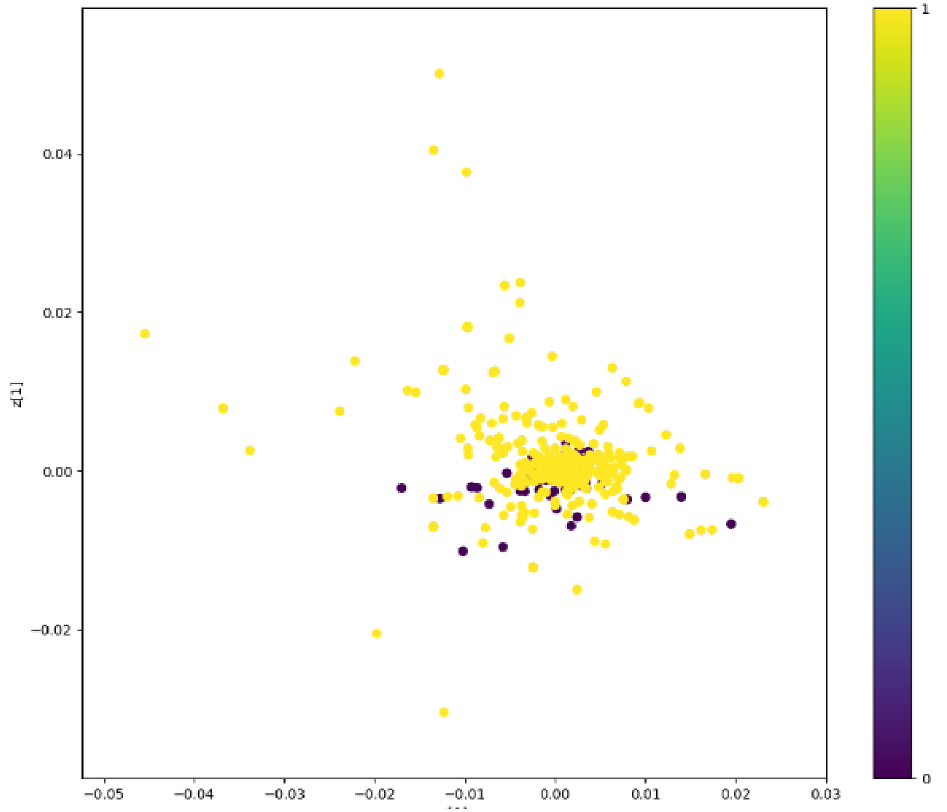
 * Android AD topology: 50-25-2/$NLL_{Gaussian}$

LOCARD

Results ii/ii

- Digging deeper into Android AD using HPROF
  - Latent spaces KPCA vs VAE





Conclusion and Next steps

- We have shown that KPCA & VAE can work for Android AD
- The process memory approach is promising, and which in turn is conducive to practical implementation
- Planned experimental improvements
  - App behavior representation: timely memory dumps
    - A meet-in-the middle with sys call traces
  - AD modeling
    - VAE – Supervised learning: a loss function that pushes the latent distribution away from labeled anomalies
- Closing the loop
  - Generate anomalous execution traces for malware sandbox triage to use
    - Static app re-writing to mark decision points close to entry point, and handler code
    - Direct sandbox execution accordingly

Q&A

Mark.Vella@um.edu.mt