Generic and Static Detection of Mobile Malware Using Machine Learning

Minh Tran

https://www.linkedin.com/in/minhtq/

Agenda

- Introduction
- Background
- Architecture
- Results
- Conclusions

FORTINET

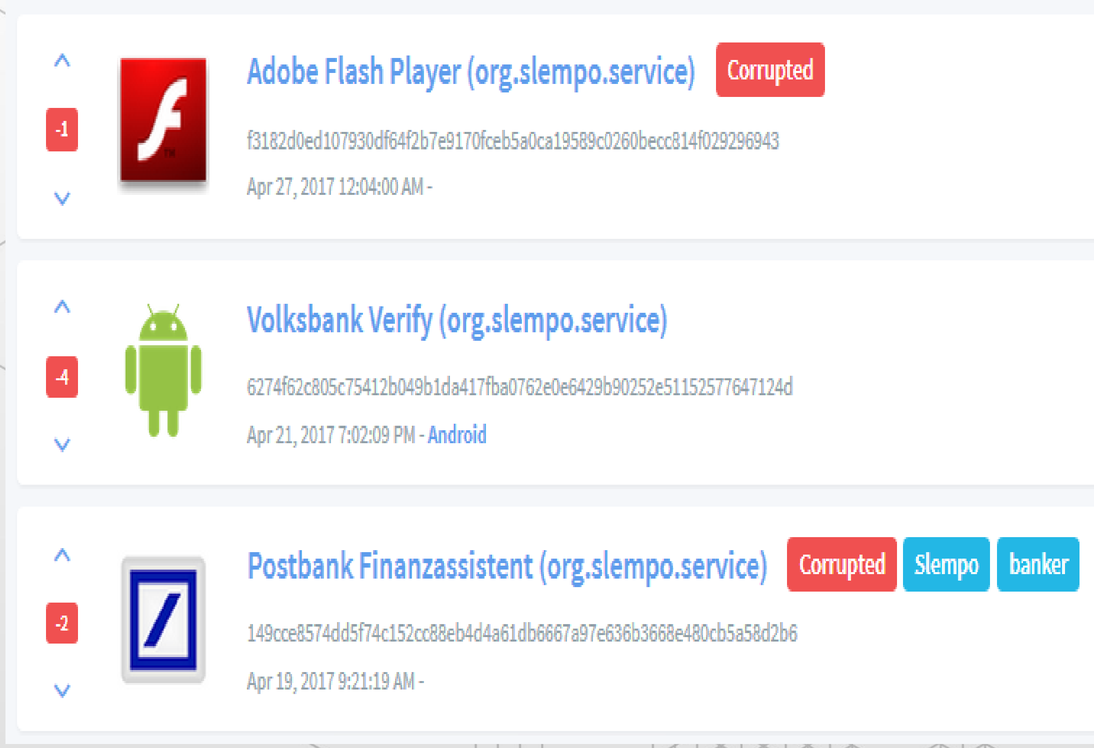
Introduction

- Sr. Security Researcher @ Fortinet (FortiGuard)
  - Sr. Malware Research Engineer @ Palo Alto Networks
- 13+ years of experience
- PhD Candidate @ North Carolina State University
  - Master of Science - 2011
- #56 of Microsoft's Top 100 Security Researchers
  - https://blogs.technet.microsoft.com/msrc/2018/08/08/microsofts-top-100-security-researchers-black-hat-2018-edition/

- *Opinions are my own

Motivating Example

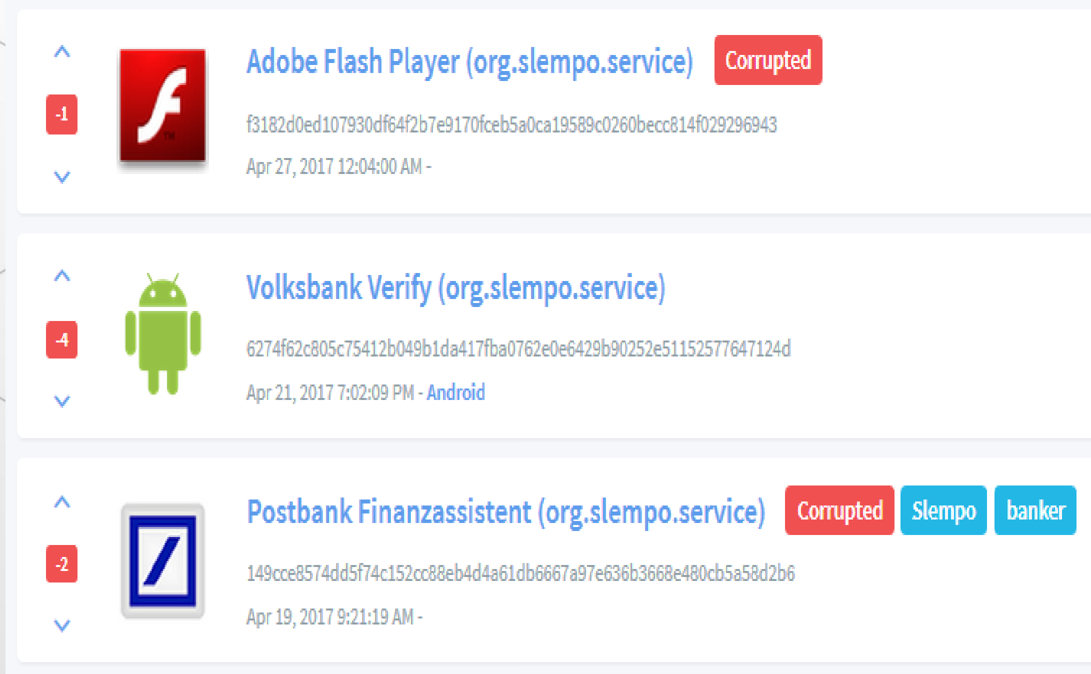
- Marcher!
- Social engineering attacks
- Corrupted



Why Signature-based and Behavior-based Malware Detection Are Still not Sufficient?

- Not resilient against variations.
- Malware samples can be corrupted

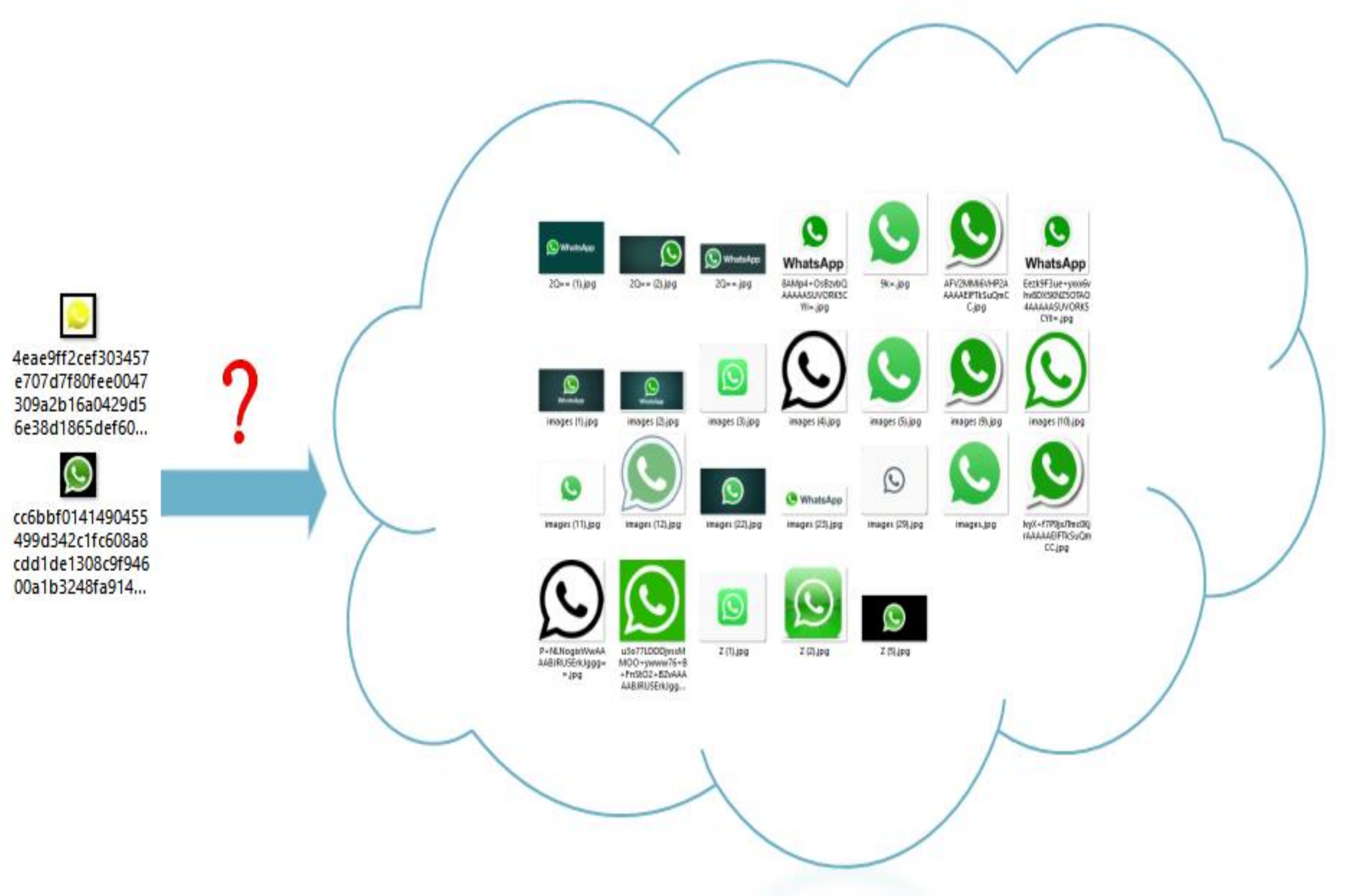
- Rooms for improvement!



Key Insights

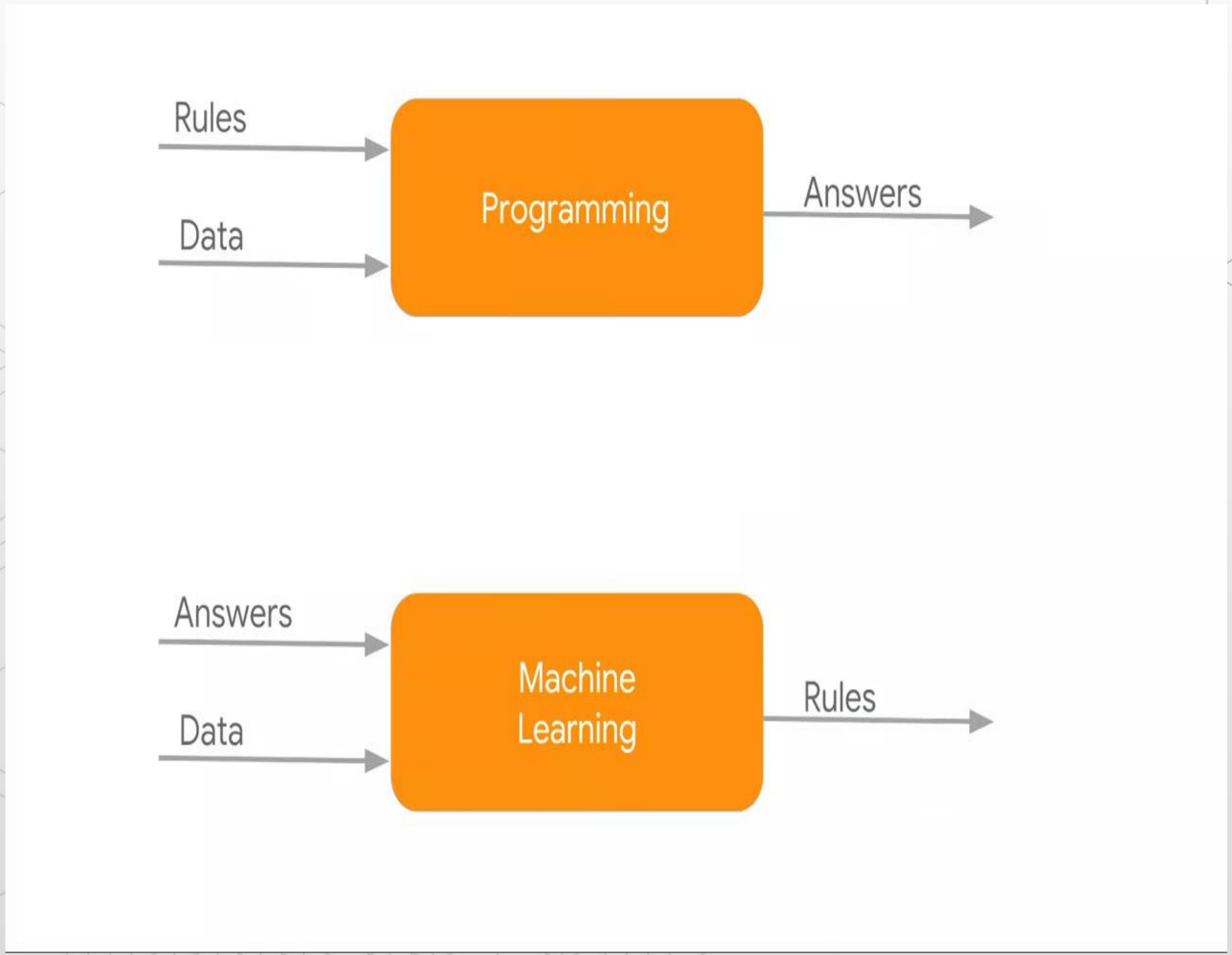
- *Legit:* ***com.symantec.mobilesecurity***
- vs
- Marcher: **etcqlnzwauf.hflivryhdnjb**
- Key Insight 1: obfuscation.
  - Use your enemy's strength against them!

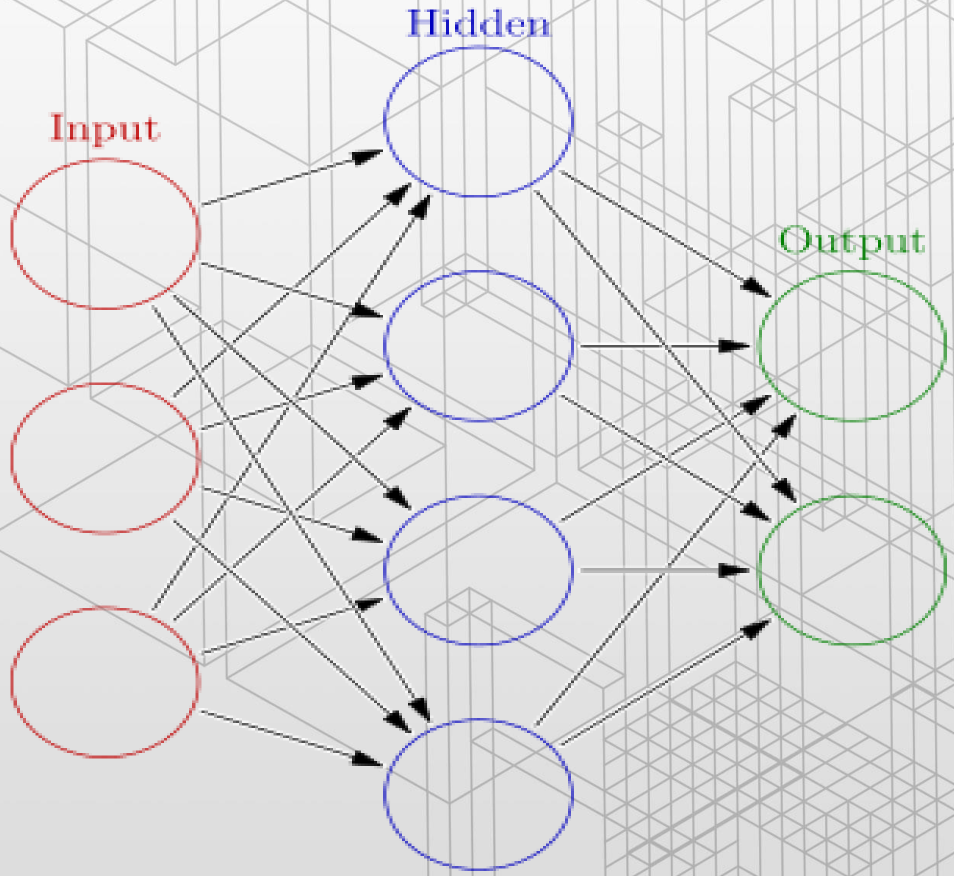
- Social engineering attacks



- A benign app should NOT do both at the same time!
  - Popular apps

Machine Learning To The Rescue



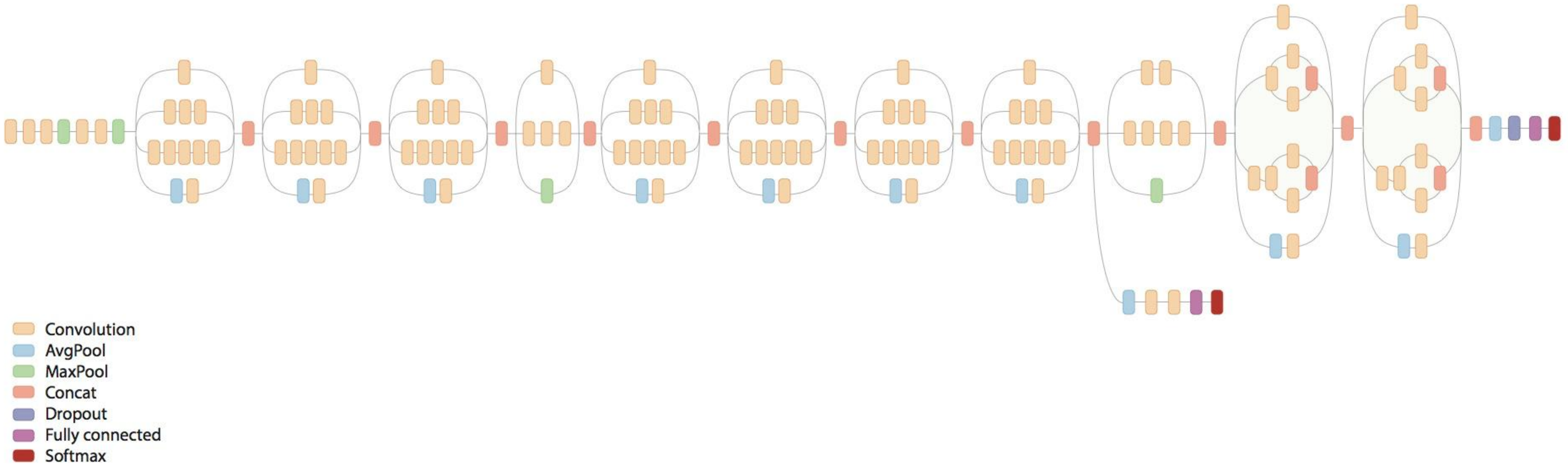


Machine Learning To The Rescue

- Classify package names:
  - N-gram

- Classify images:
  - Neural Networks

TensorFlow

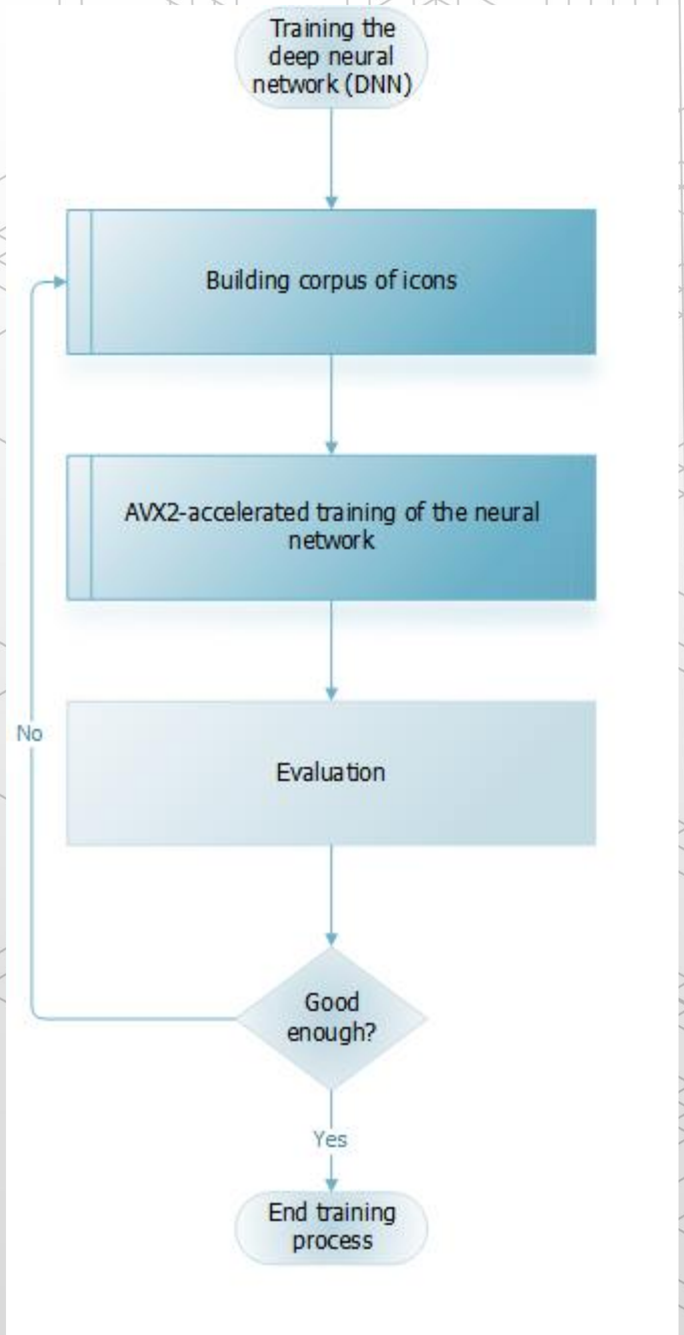


- PoC: Inception-v4:
  - 43 layers (deep learning!)
  - Lower computational cost (vs e.g. VGGNet)

**Credit belongs to the respective owners

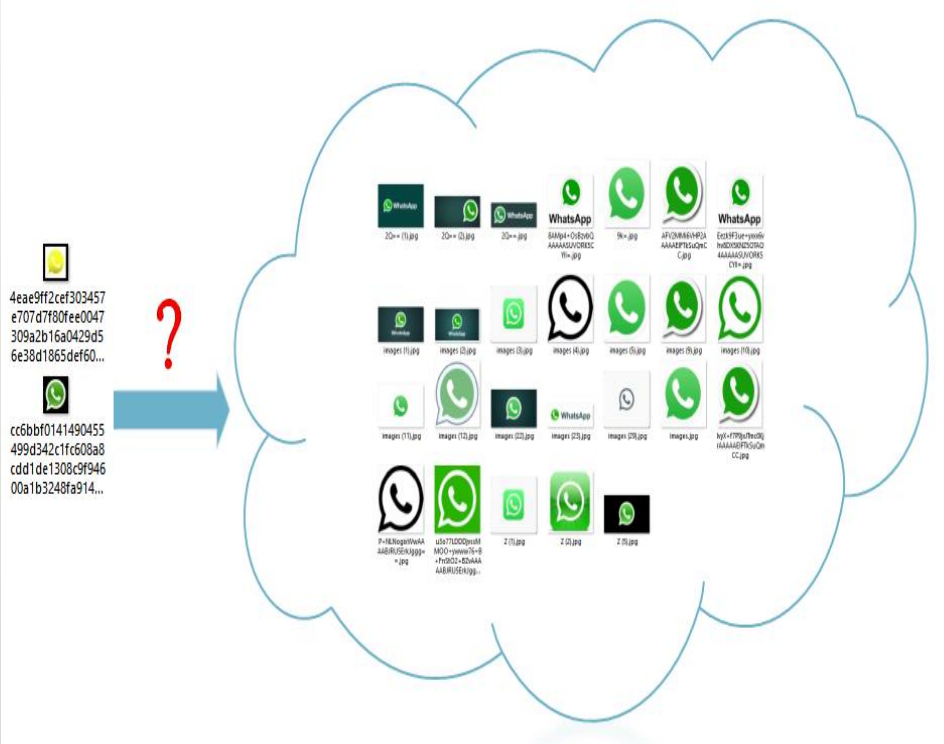
Workflow to Train the DNN

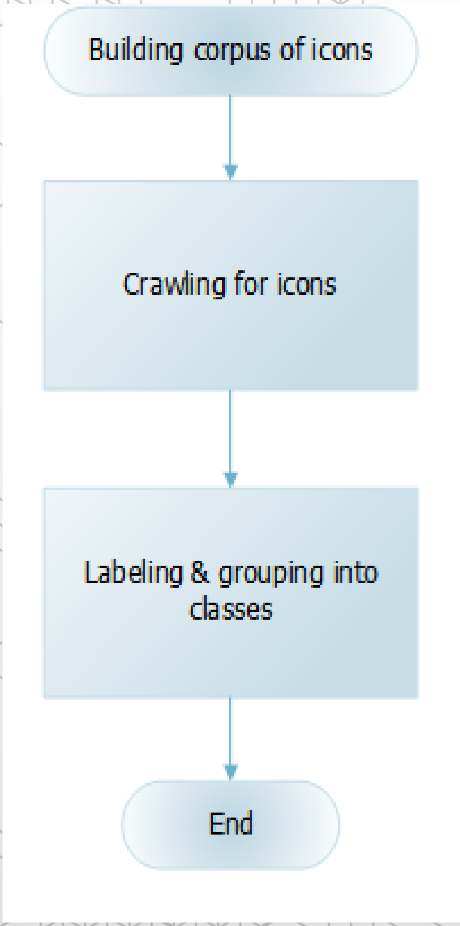
- Building corpus of icons
- Training of the neural network
  - Produce model files
- Evaluation using the test corpus



Building Corpus of Icons

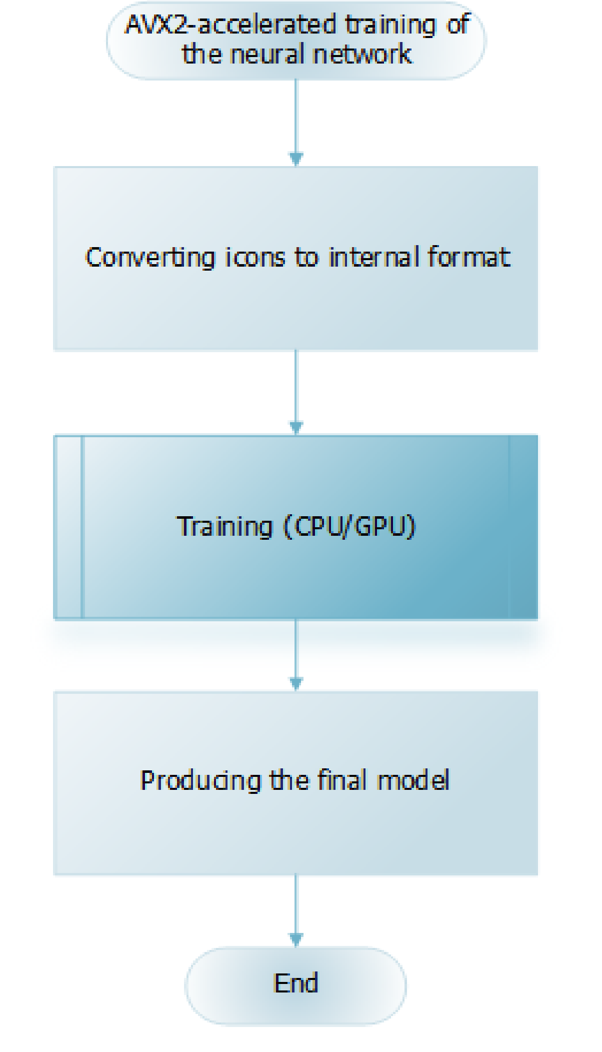
- Crawling for icons of legitimate apps (e.g. WhatsApp) using Google Images search
- Labeling & grouping into classes. One class corresponds to one app.





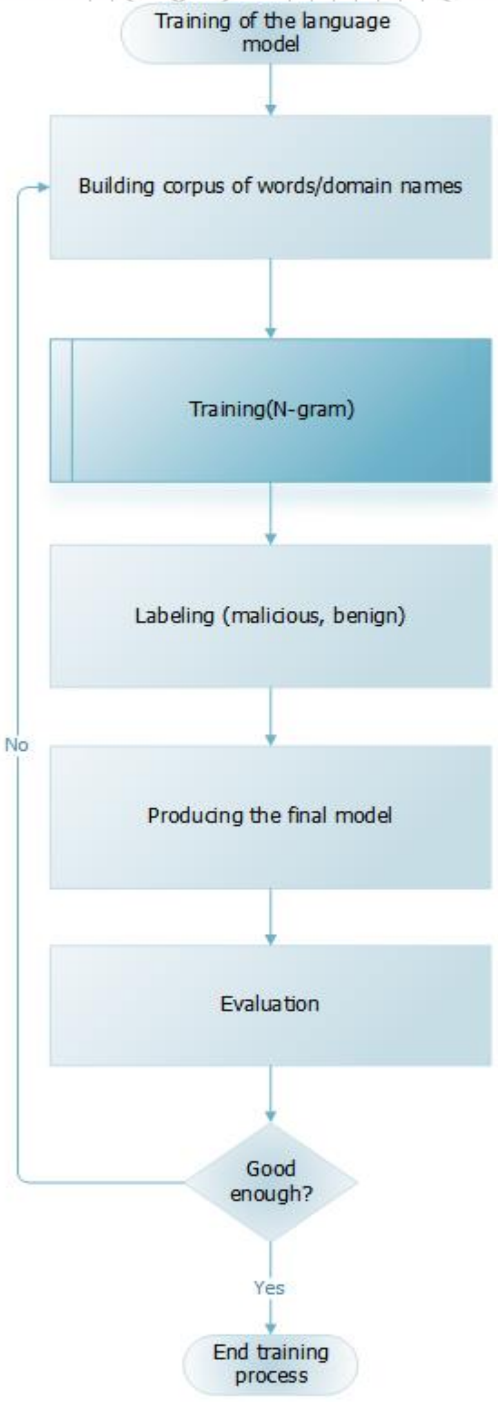
Training of the Neural Network Model

- Converting icons to internal format
- Training the neural network for n steps (e.g. n = 3000)
- Producing the final model (i.e. model files with the optimal weights & biases for neurons)
- Evaluation based on testing corpus



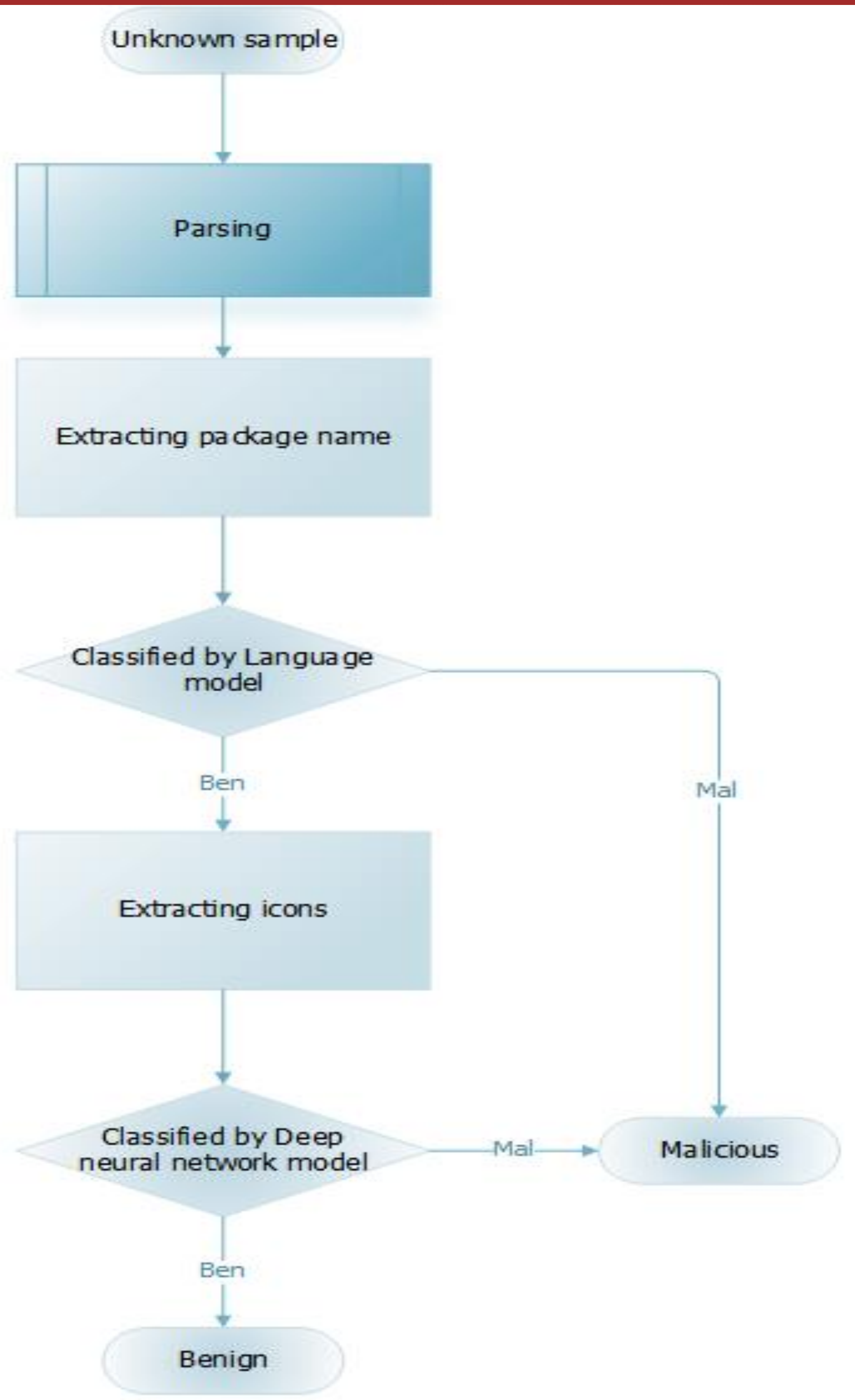
Training of the Language Model

- Building corpus of words/domain names/package names (e.g. Alexa, Majestic Million)
- Training (N-gram with n is a customizable length parameter e.g. n = 2)
- Labeling (malicious, benign) based on ground truths (from existing malware collections)
- Producing the final model
- Evaluation based on testing corpus



Workflow to Classify Samples

- Parsing packages
- Extracting package name and feed into the Language Model
- Extracting icons and feed into the Neural Network Model



Results

- Test set: 306847 samples
- 2gram total detection: right 271133 vs wrong 35714 = 11.64%
  FN 88.13% FP 11.87%
- 3gram total detection: right 277024 vs wrong 29823 = 9.72%
  FN 78.88% FP 21.12%
- 4gram total detection: right 274412 vs wrong 32435 = 10.57%
  FN 84.69% FP 15.31%

Results

- Our system classifies all Android malware, but especially effective against social engineering malware who masquerade as legitimate apps
- Our system has better coverage: many samples can be corrupted and our system still works because fundamentally speaking it is static analysis whereas solutions based on dynamic analysis fail.
- Our system has better performance: it is faster than dynamic analysis because no execution in sandbox is required
- Effectively speaking, detection rate is 99.928%.

Conclusions

- ML is valuable to malware detection
- Future research
  - Increasing the quality and the quantity of the data set: different languages etc
  - Improving training performance: distributed training etc

Questions