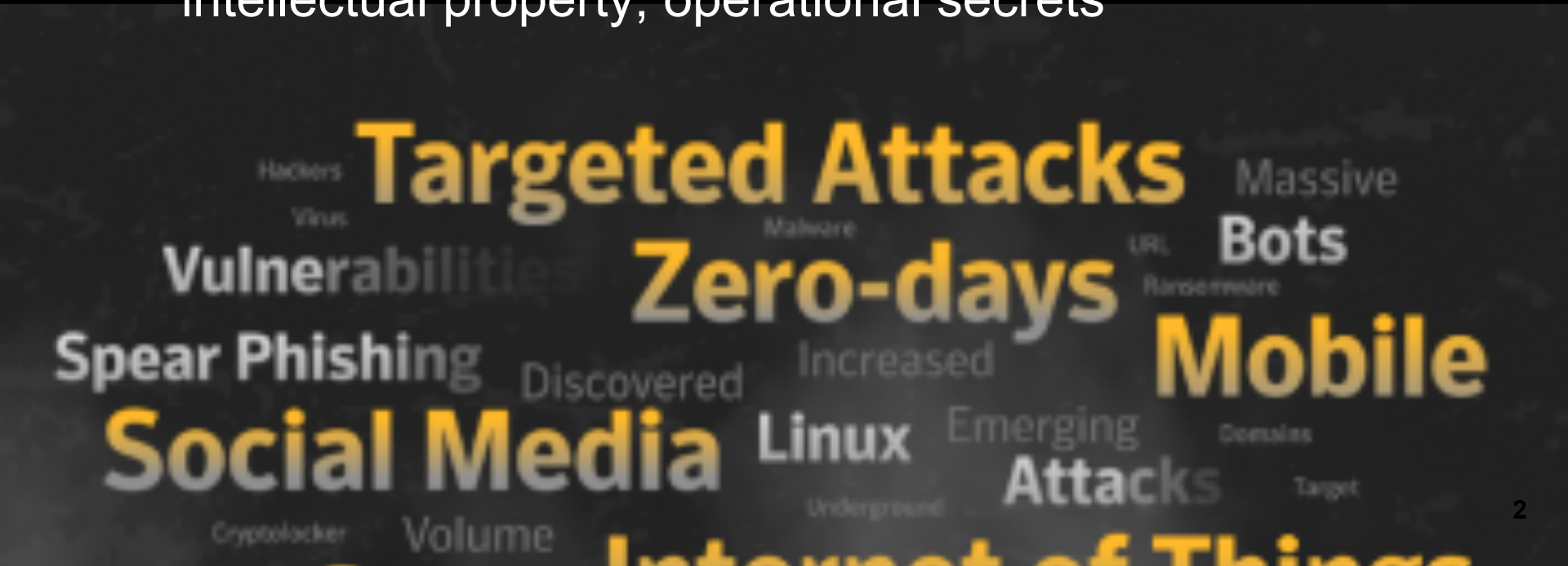
Guilt by Association:

Large Scale Malware Detection by Mining File-relation Graphs

Acar Tamersoy	Georgia Tech
Kevin Roundy	Symantec
Polo Chau	Georgia Tech

Malware Attacks

- Protection against novel malware attacks is becoming more important
- Malware attacks are causing great damage
  - **Consumers**: monetary & emotional costs
  - **Government & businesses**: financial cost, losses of intellectual property, operational secrets

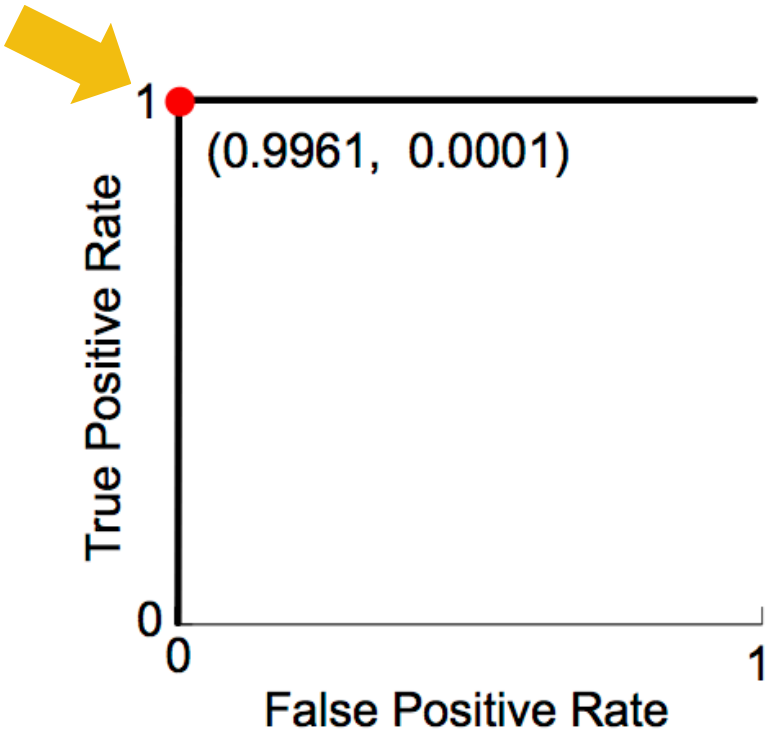


Goals of Protection

- Limit malware's window of effectiveness
- Obtain low **false positive rate** (FP)
  - labeling a benign file as malicious can have devastating consequences (e.g., system files)

We achieve 99.61% true positive rate (TP) at **0.01% FP rate**

In literature, results are often reported at 1% FP rate **(100x)**



Our Approach: AESOP Algorithm

- Current techniques for malware detection do not consider **file-to-file relationships**
- Relationships we exploit
  - Multiple files used by same software
  - Files used by malware to talk to command-and-control server

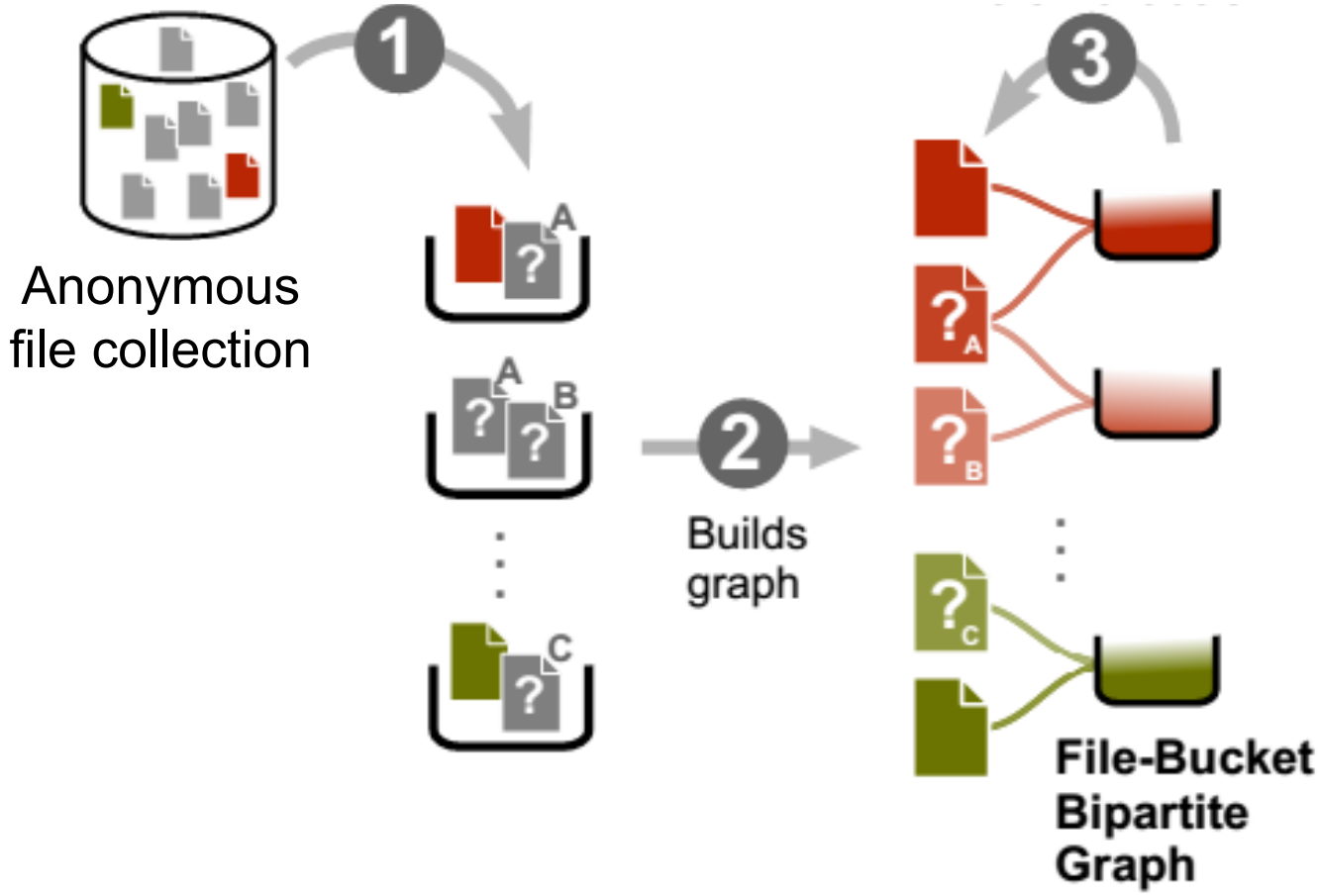
Our Intuition:

If we can identify related files, we can label unknown files using **guilt by association**

AESOP Technology Overview

Cluster related files into buckets using **locality-sensitive hashing (LSH)**

Propagate goodness scores to unlabeled files using **belief propagation (BP)**



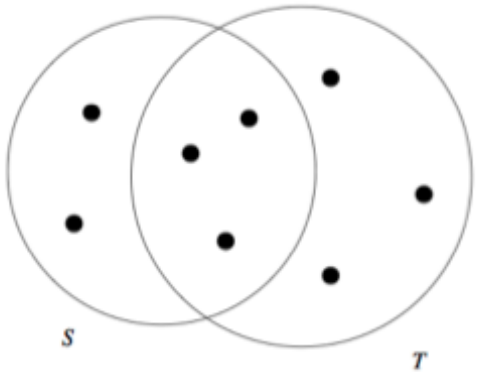
Step 1:

Cluster related files into buckets *based on how frequently they co-occur*

Finding Related Files via Co-occurrence

- Let M_{f1} and M_{f2} be the sets of the machines files f_1 and f_2 appear on
- We quantify co-occurrence based on the overlap between M_{f1} and M_{f2} and use Jaccard similarity

$$Jaccard(f_1, f_2) = \frac{|M_{f_1} \cap M_{f_2}|}{|M_{f_1} \cup M_{f_2}|}$$



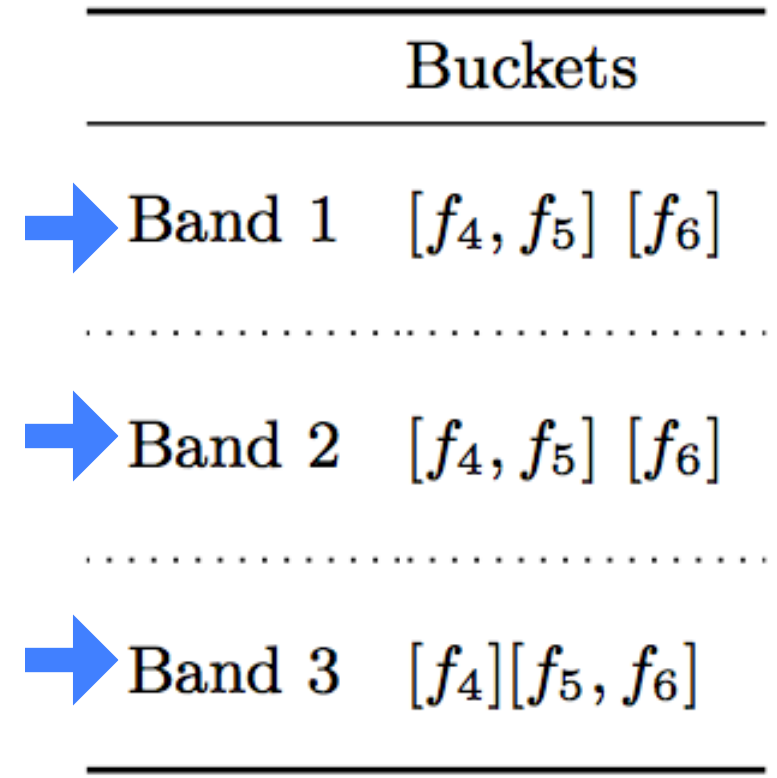
- $Jaccard(f_1, f_2) > 0.5 \rightarrow f_1$ and f_2 highly related

Challenges in Clustering Files

- Naïve approach
  - consider every pair of files (trillions)
  - compute the Jaccard similarity between files
  - **Does not scale**
- Solution
  - Use **MinHashing** to compute Jaccard similarities quickly and approximately
  - Use **Locality-sensitive Hashing (LSH)** to increase approximation accuracy

Locality-sensitive Hashing (LSH) with MinHashing

- Main ideas:
  - LSH puts related files (high Jaccard similarity) into same bucket with high probability
  - To increase probability, LSH uses multiple hash functions
- Can find related files with only **one pass** over the dataset



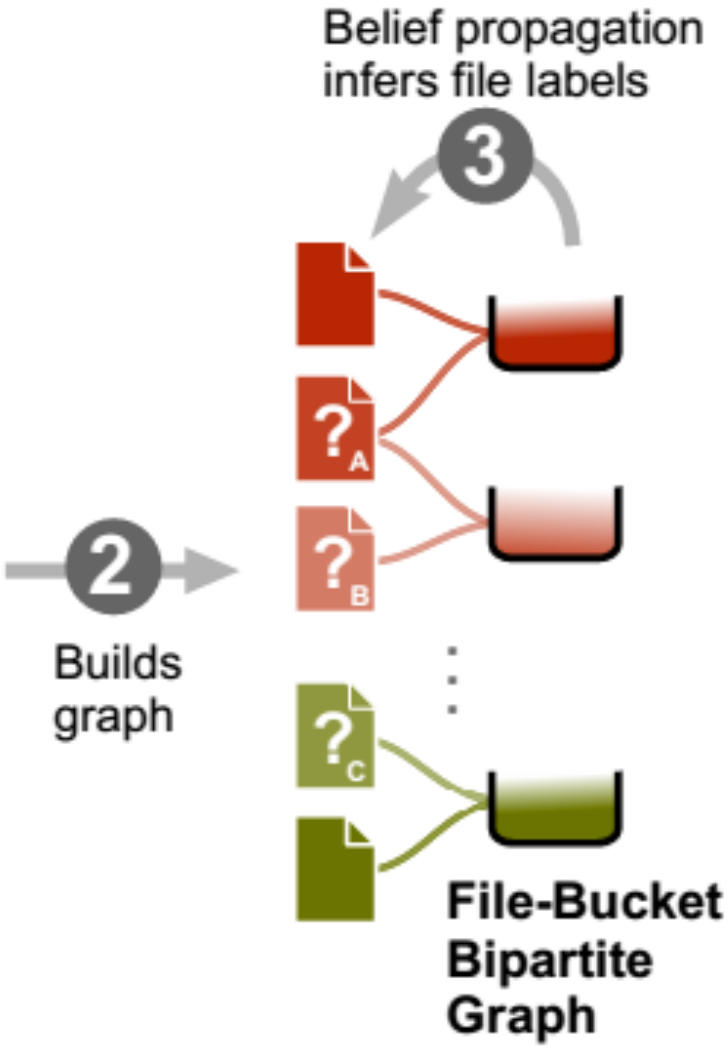
Step 2:

Construct a **file-bucket graph**

- Buckets become **middlemen** connecting related files
- Files are “clustered”

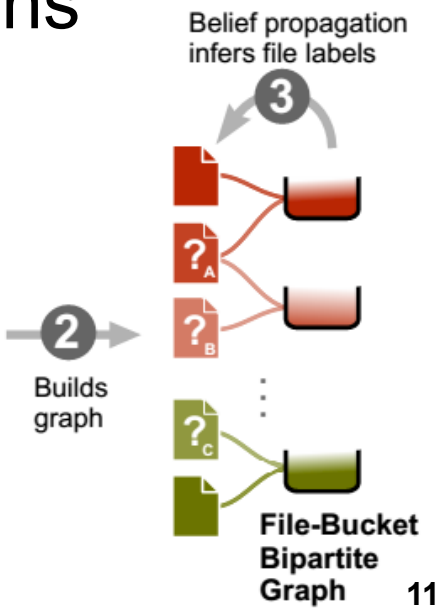
Step 3:

Propagate goodness scores to unlabeled files

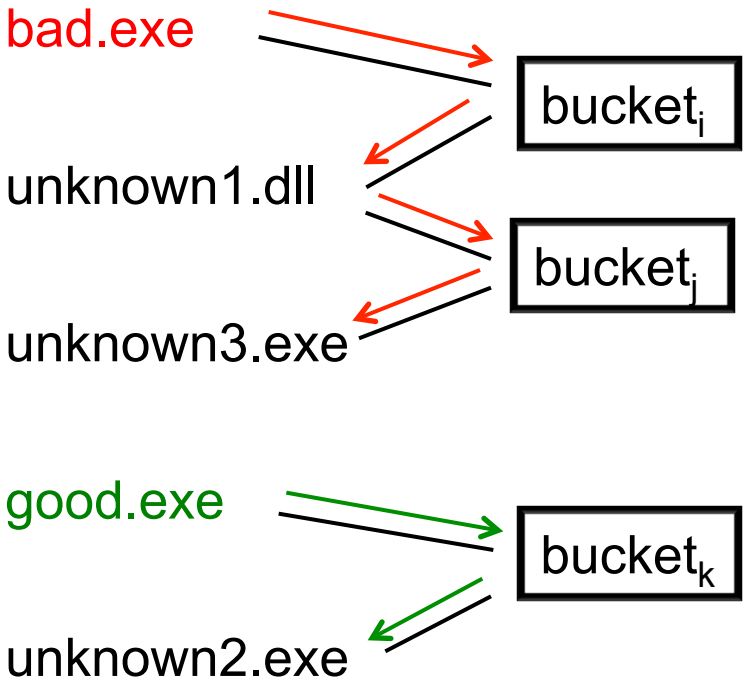


Belief Propagation (BP)

- Solves inference problems over large graphs in many domains
  - computer vision, image processing, error correcting code, etc.
- Infers a node's state from some prior knowledge about the node and its neighbors' opinions



Labeling Files using Belief Propagation



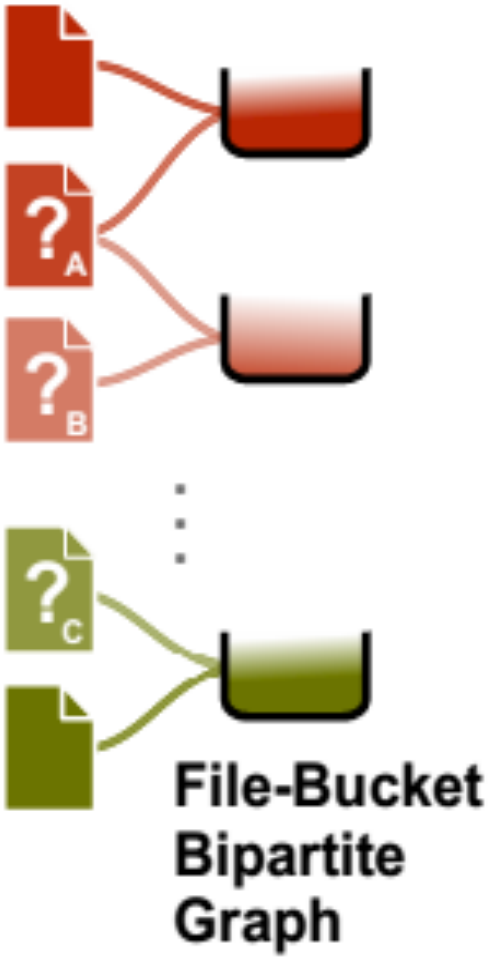
Final Beliefs/ Goodness

bad.exe	0.01
unknown1.dll	0.18
unknow3.exe	0.29
good.exe	0.99
unknown2.exe	0.81

Main Idea: **"Guilt by Association"**

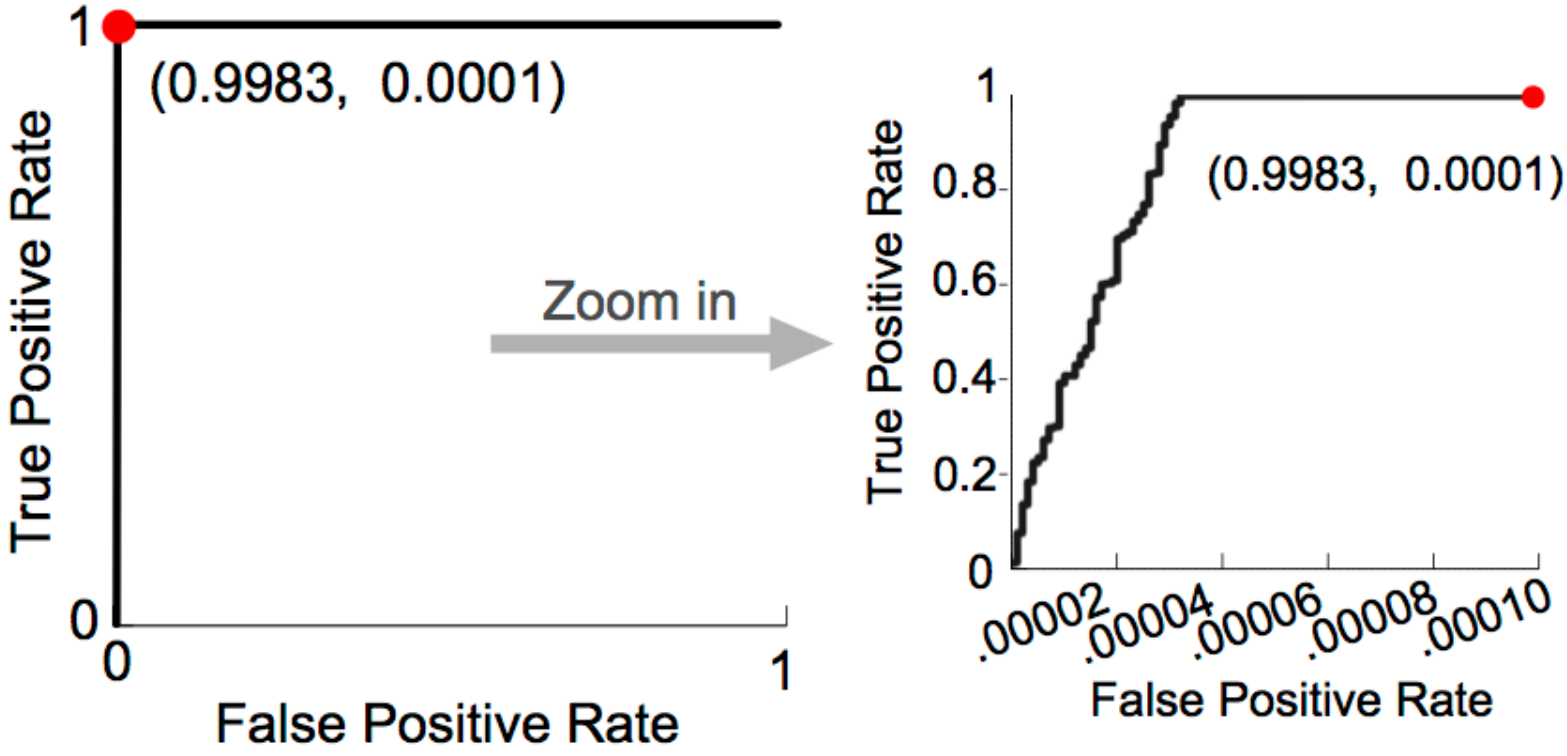
Experiments

- File-bucket graph contains
  - 6M nodes and 19M edges
    - 1.6M good files
    - 47K bad files
    - 1M unlabeled files



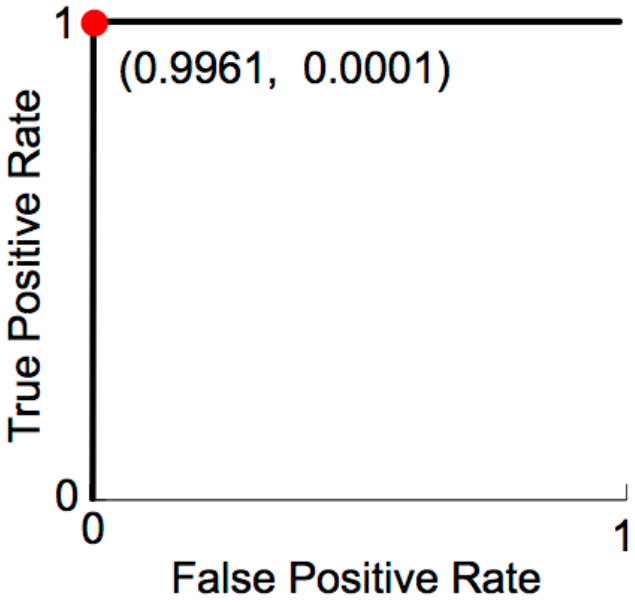
Evaluation using 10-fold Cross Validation

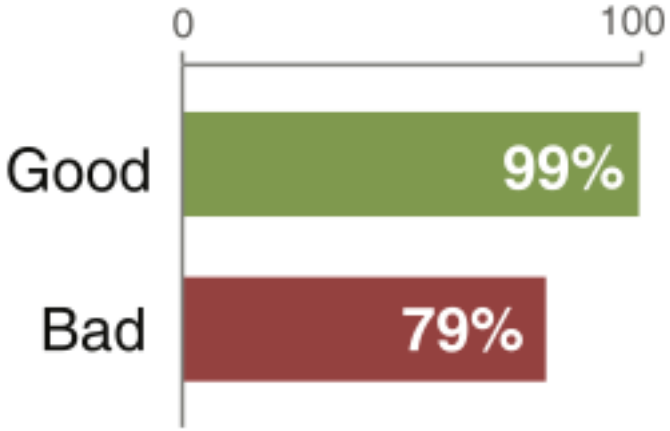
- Labeled files in the fold under consideration are treated as unlabeled



Evaluation in the "Wild"

- Tested on **initially unlabeled** files that became labeled after **3 months**
  - 17K unlabeled to labeled good files
  - 774 unlabeled to labeled bad files



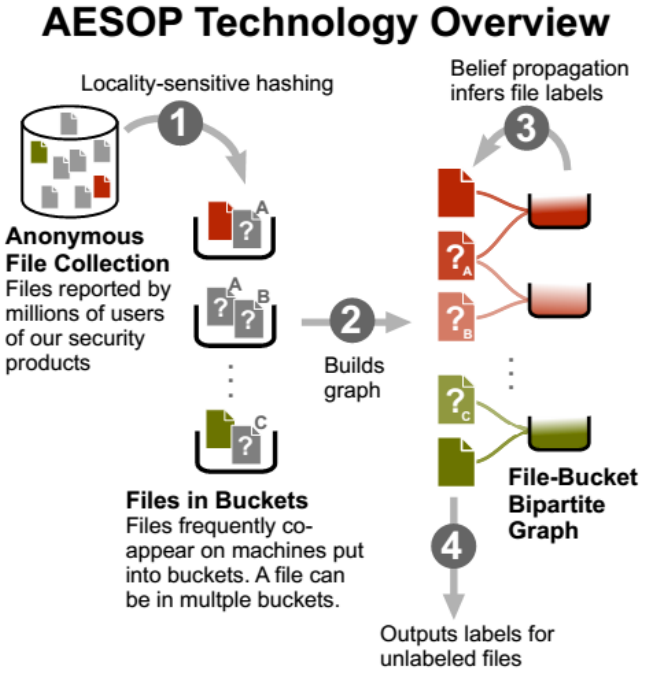


% of files AESOP detects **at least 1 week** ahead of current technology

Conclusions & Deployment

Thanks!

- AESOP achieves **99.61%** TP rate at **0.01%** FP rate
  - **LSH** to cluster related files
  - **BP** to infer file goodness
- Patent pending with plans for deployment



Acar Tamersoy (tamersoy@gatech.edu)
PhD student, Georgia Tech
Symantec Fellow, 2014

Georgia Tech