EASEAndroid: Automatic Policy Analysis and Refinement for Security Enhanced Android via Large-Scale Semi-Supervised Learning

Ruowen Wang[1,2], William Enck[2], Douglas Reeves[2], Xinwen Zhang[1], Peng Ning[1,2], Dingbang Xu[1], Wu Zhou[1], Ahmed M. Azab[1]
[1]Samsung KNOX R&D, Samsung Research America
[2]North Carolina State University

SAMSUNG RESEARCH AMERICA

Security Enhanced Android

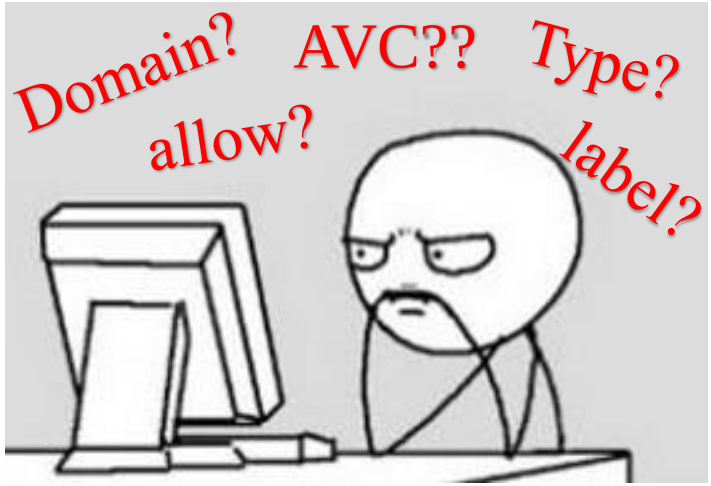
“Multiple vulnerabilities have been prevented since we introduced SELinux”

--Android Official Blog, October 28, 2014

“SEAndroid prevents first exploit against commercial phone”

--http://securityblog.org/2013/04/30/SE-Android-and-the-motochopper-exploit/

The Core of SEAndroid: Policy



"Vendors don't know how to write policies"

--@pof "Defeat SEAndroid" at Defcon 2013

Policy Language

- Security labels <=> Concrete Subjects/Objects

```
app_data_file <=> /data/data/.*
```

- Allow rules grant benign operations

```
allow appdomain app_data_file:file
{read write execute}
```

- Neverallow rules define privilege escalation

```
neverallow untrusted_app init:file
{read} (Compile-Time)
```

SEAndroid Policy Challenges

- Require Complete Redesign of Policy
  - Android is different from traditional Linux
- Require Policy Analysts to Have Both
  - Domain Knowledge (Allow Benign Accesses)
  - Security Expertise (Prevent Malicious Accesses)
- Require Continuous **Refinements**
  - New Android releases
  - New attacks

How to Refine?
Analyze Audit Log

- Audit Log
  - Log access events not matched with `allow` rules
  - Analysts parse the logs to refine policy
- Information in one access event
  - Security labels of the denied access
  - Syscall Subject Info (e.g. process)
  - Syscall Object Info (e.g. file path)
- We model as 6-tuple access pattern
  - **<sbj, sbj_label, perm, tclass, obj, obj_label>**

Real-World Challenges

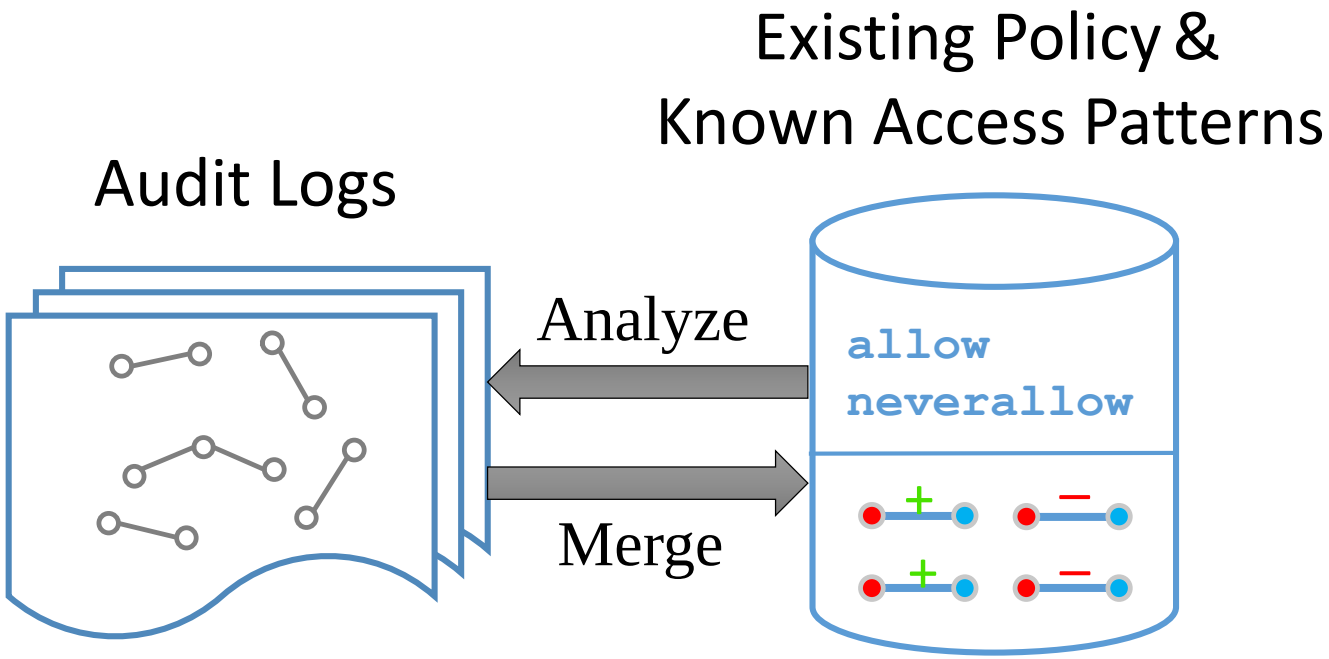
- Millions of such audit logs
- Unknown new benign & malicious access patterns mixed together
- Continuous efforts due to Android updates and emerging new attacks

EASEAndroid

- Elastic Analytics of SEAndroid
- Features:
  1. Analyze audit logs in a large scale
  2. Classify new benign & malicious access patterns
  3. Propose new security labels and rules as policy refinements
- Key insight:
  - Model policy refinement as semi-supervised learning

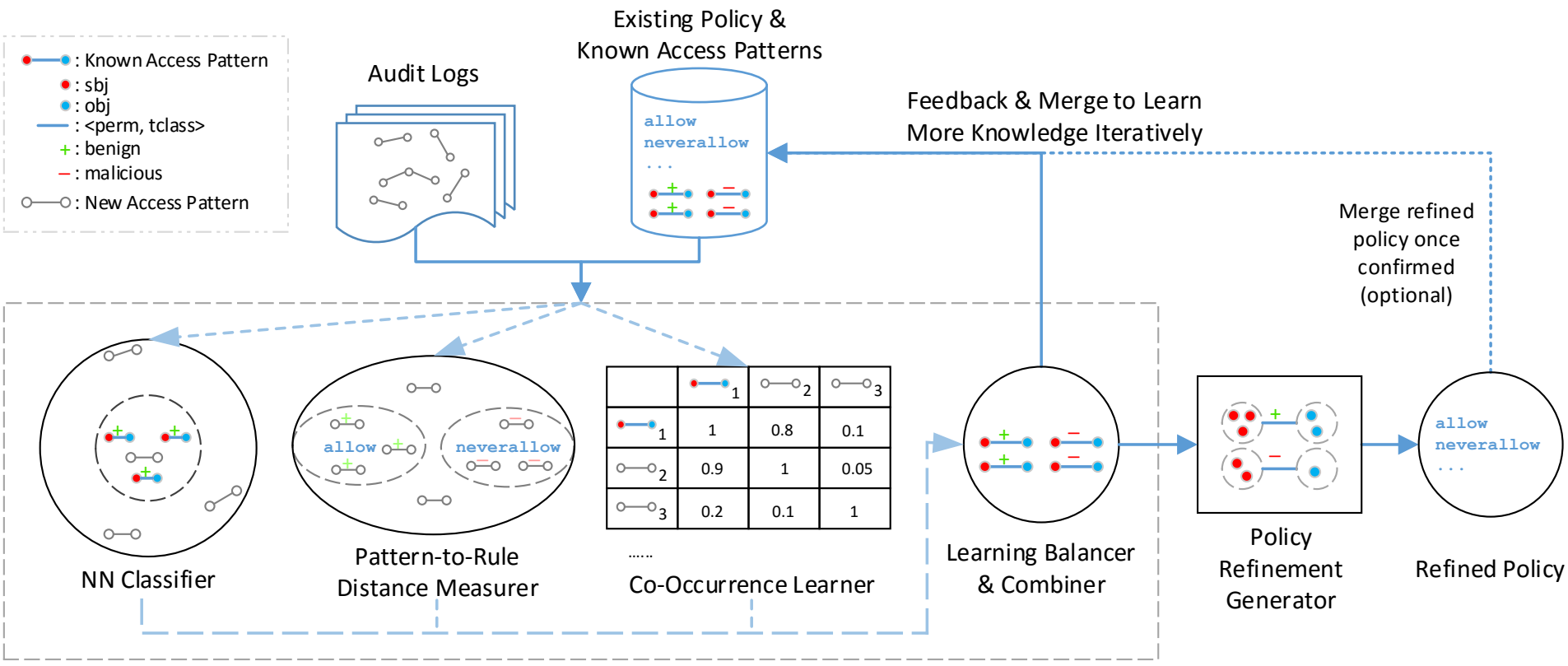
: Known Access Pattern
: sbj
: obj
: <perm, tclass>
+ : benign
– : malicious
: New Access Pattern

Key Insight



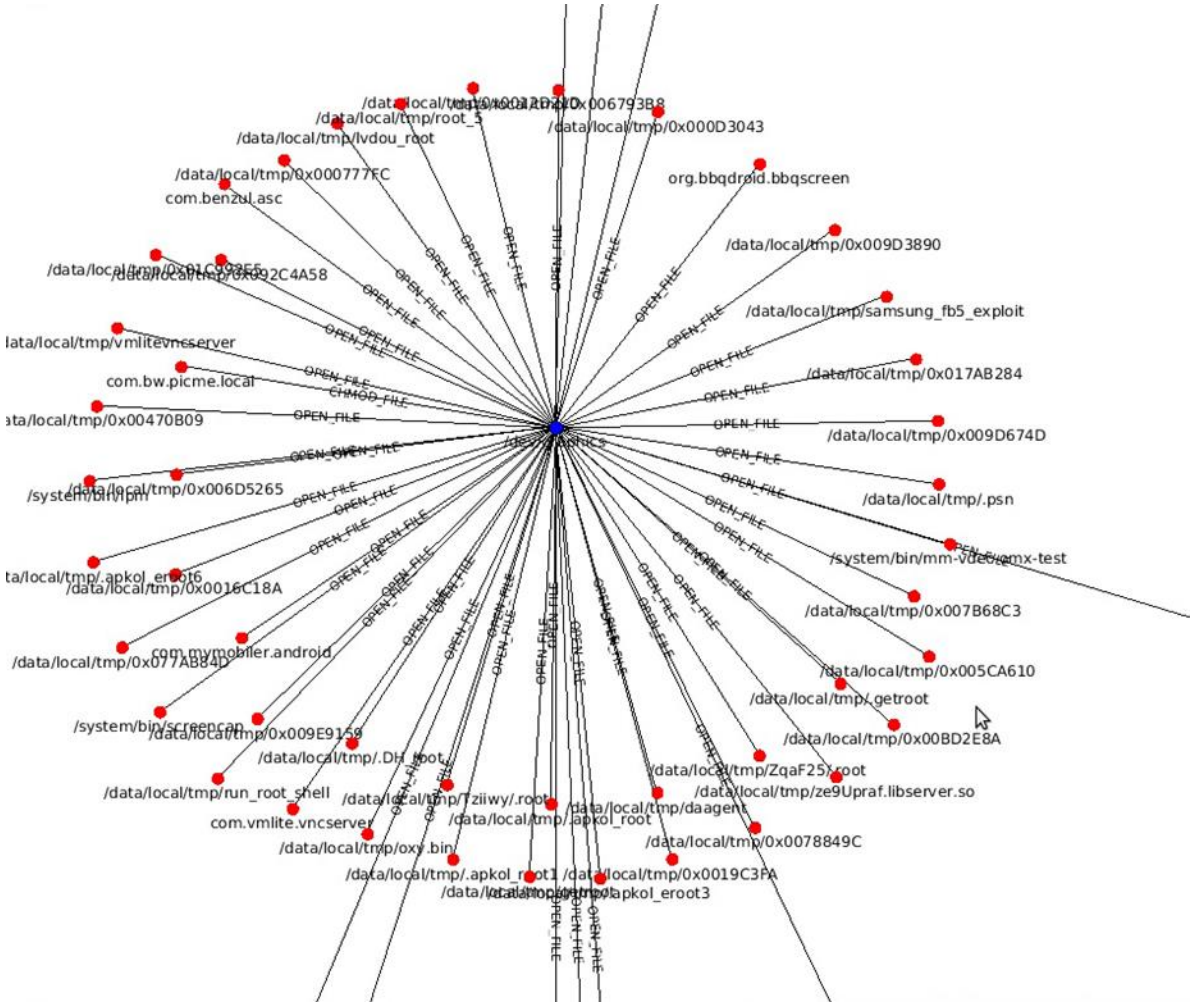
Learning Unknown based on Semantic Correlations
Known <=> Unknown
(Semi-Supervised Learning)

EASEAndroid Architecture



Nearest-Neighbor (NN) Classifier

- Observation
  - Known sbjs perform new access patterns
    - Android apps/binaries update with new features
  - New sbjs perform known access patterns
    - Exploit kits share malicious access patterns
- NN Classifier identifies connections between
  - Known subjects <=> New access patterns
  - New subjects <=> Known access patterns



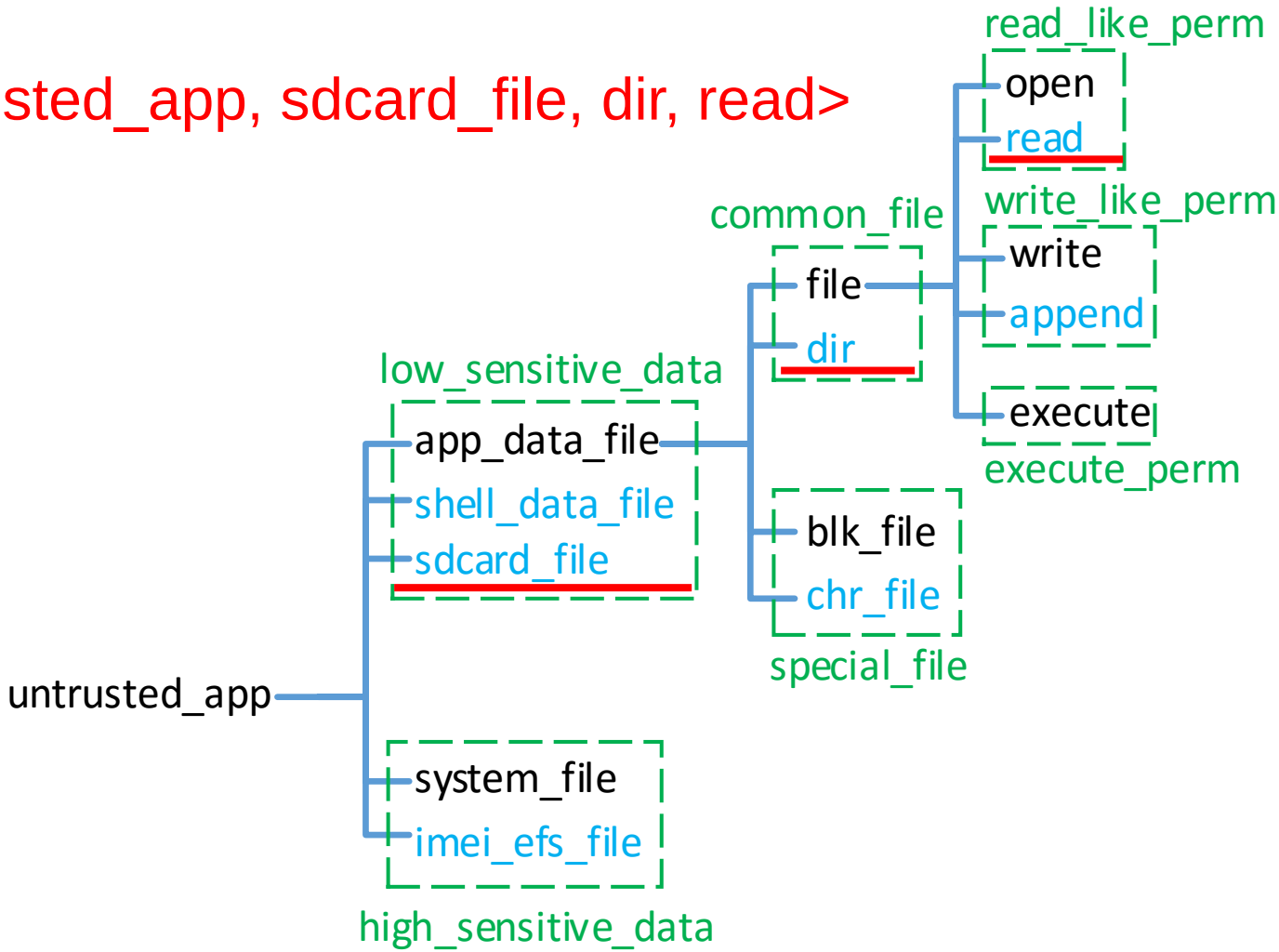
Known & unknown exploit kits share the same `/dev/graphics/fb` exploit

Pattern-to-Rule (P2R) Distance Measurer

- Observation
  - New access patterns close to existing incomplete rules are the missing parts of those rules
- Decision-Tree-based Approach
  - Classified as benign if closest to `allow`
  - Classified as malicious if closest to `neverallow`
  - Remain unclassified if far from both sides

Decision-Tree-Based P2R

<untrusted_app, sdcard_file, dir, read>



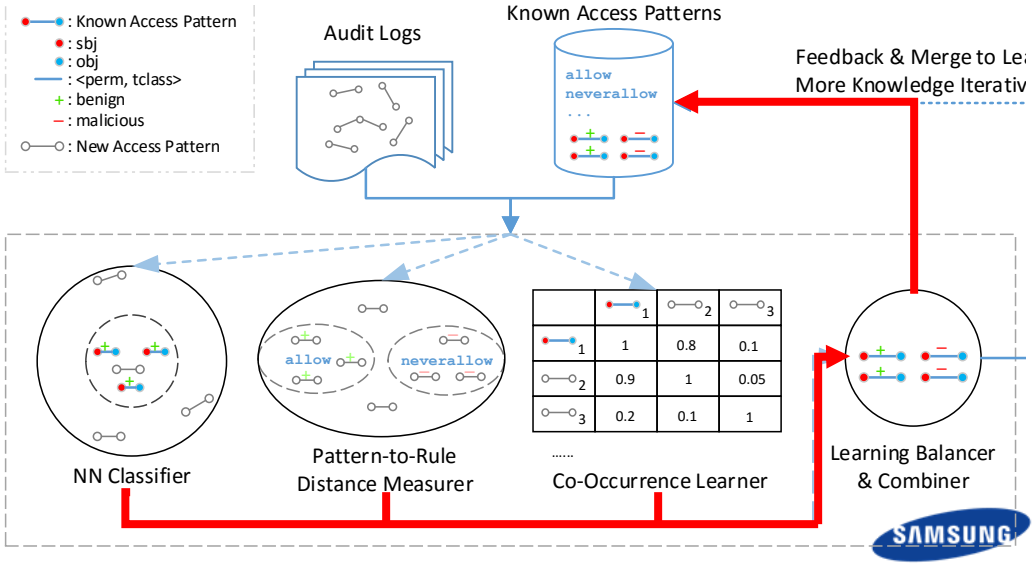
Co-Occurrence Learner

- Observation
  - A functionality or an attack often involve a series of access patterns captured together
- Co-Occurrence Learner
  - Infer new access patterns based on known access patterns if they co-occur together

				……
	1	0.8	0.1	
	0.9	1	0.05	
	0.2	0.1	1	
……				

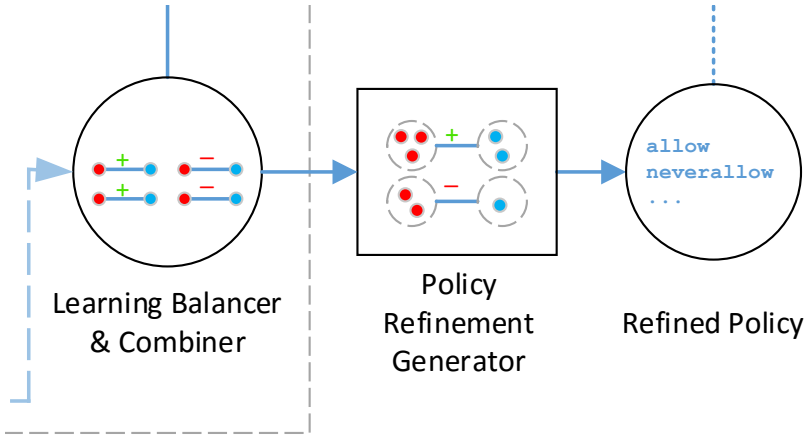
Learning Balancer & Combiner

- Manage thresholds of each learner
- Combine results to expand knowledge base
- Balance precision and coverage
  - Automated Mode (high precision)
  - Semi-Automated Mode (high coverage)



Policy Refinement Generator

- Suggest new security labels and rules
- Group sbjs/objs together based on existing coarse-grained labels
- Infer fine-grained labels and encode into rules



Implementation

- 8-node Hadoop Cluster, 256 GB Memory.
- Distributed SQL on HDFS.
- SLOC: 10K Java + 5K Cloudera Impala

Evaluation

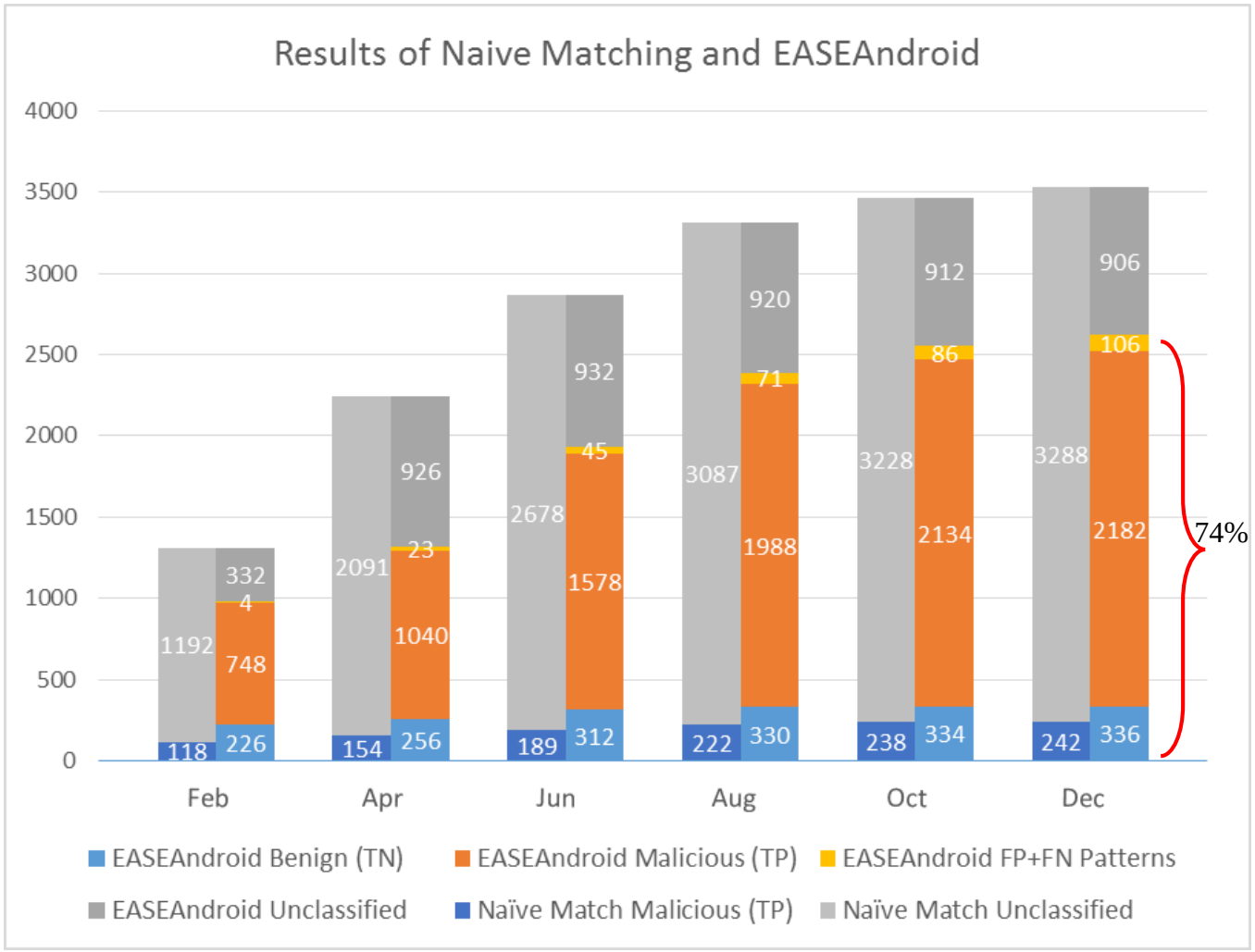
- **RQ1:** How is the coverage and precision of EASEAndroid? What are the effects of different threshold settings?
- **RQ2:** How is a policy refinement generated? What is the difference compared to human-written policy?
- **RQ3:** What kinds of new malicious patterns are discovered by EASEAndroid?

Evaluation

- Audit Log Dataset
  - 1.3M logs from real-world Samsung devices with Android 4.3 over 2014 (collected anonymously with user consent)
  - 145K unique access events and generalized into 3530 access patterns
- Initial Knowledge
  - An early version policy, 9 confirmed exploit kits
- Ground Truth
  - A later version of human-refined policy
  - Consult with experienced policy analysts

Evaluation

Coverage & Precision



Evaluation
Different Thresholds (Coverage)

NN Classifier Threshold	Pattern-to-Rule Threshold	Co-Occurrence Threshold	Classified Malicious (TP+FP)	Classified Benign (TN+FN)	Remain Unclassified
$\sigma = 55\%$	Dist ≤ 2	$c_{ij} > 0.55$	77.2%	14.0%	8.8%
$\sigma = 65\%$	Dist ≤ 1	$c_{ij} > 0.65$	70.0%	11.8%	18.2%
$\sigma = 75\%$	Dist ≤ 1	$c_{ij} > 0.75$	65.7%	10.9%	23.4%
$\sigma = 85\%$	Dist ≤ 0	$c_{ij} > 0.85$	63.9%	10.5%	25.7%
$\sigma = 95\%$	Dist ≤ 0	$c_{ij} > 0.95$	53.1%	9.2%	37.7%

Evaluation
Different Thresholds (Precision)

NN Classifier Threshold	Pattern-to-Rule Threshold	Co-Occurrence Threshold	True Malicious (TP)	False Malicious (FP)	True Benign (TN)	False Benign (FN)
$\sigma = 55\%$	Dist ≤ 2	$c_{ij} > 0.55$	62.96%	37.04%	58.65%	41.35%
$\sigma = 65\%$	Dist ≤ 1	$c_{ij} > 0.65$	88.73%	11.27%	71.35%	28.65%
$\sigma = 75\%$	Dist ≤ 1	$c_{ij} > 0.75$	91.35%	8.65%	88.92%	11.08%
$\sigma = 85\%$	Dist ≤ 0	$c_{ij} > 0.85$	96.81%	3.19%	90.81%	9.19%
$\sigma = 95\%$	Dist ≤ 0	$c_{ij} > 0.95$	97.27%	2.73%	100.00%	0.00%

Evaluation
Refinement Example

<surfaceflinger, {open…}, file, /data/misc/zoneinfo/*, system_data_file>
<dhcpcd, {open…}, file, /data/misc/zoneinfo/*, system_data_file >
<pppd, {open…}, file, /data/misc/zoneinfo/*, system_data_file >
<vendor_daemon, {open…}, file, /data/misc/zoneinfo/*, system_data_file >

=>

```
/data/misc/zoneinfo/* ⇔ u:object_r:zoneinfo_file:s0
attribute access_zoneinfo_domain;
typeattribute surfaceflinger access_zoneinfo_domain;
.....(same for other 3 system daemons)
allow access_zoneinfo_domain zoneinfo_file:file {open
read};
```

Evaluation
Comparison with Human-written Policy

- 336 benign access patterns, 51 policy rules
- All rules semantically match human rules
- EASEAndroid: Fine-grained + Evidence
  - `allow access_zoneinfo_domain zoneinfo_file:file {open read};`
- Human: Coarse-grained + Macro
  - `allow system_domain zoneinfo_data_file:file rw_file_perms;`

Evaluation

Malicious Access Patterns by EASEAndroid

- {read,write} files in /dev/graphics, /dev/block, /dev/exynos-mem, /dev/mem
- {dac_override,chown,fsetid} capability in /data/data, /data/local, /data/misc, /data/system, /sdcard
- {create,write,unlink} files in /system/app, /system/bin, /system/xbin, /system/etc
- {read,write} files in /sys/block, /sys/devices, /sys/fs, /sys/kernel
- {read,write} files in /proc/sys, /proc/pid/environ|exe|mem
- {kill,sys_admin,sys_ptrace,sys_chroot,setuid,setgid} capability
- {transition,dyntransition} process
- {connectto} unix sockets of privileged daemons directly

Evaluation
Attacks against SEAndroid

- Manipulate files under `/sys/fs/selinux`
- Inject `allow` rules into policy
- Transition to privileged domain
- Modify `struct cred` by exploiting kernel

Limitations

- Information missed by audit logs
  - High-level semantics in Android framework
- Countermeasure against EASEAndroid
  - Data poisoning attacks
- Unclassified access patterns
  - Human can interact with EASEAndroid by adding extra knowledge

Conclusion

- SEAndroid policy development and refinement is challenging
- We propose EASEAndroid, an analytic system to refine the policy based on semi-supervised learning
- Big Data/Machine Learning is promising for security policy development & management

Thank you for your time!

Q & A