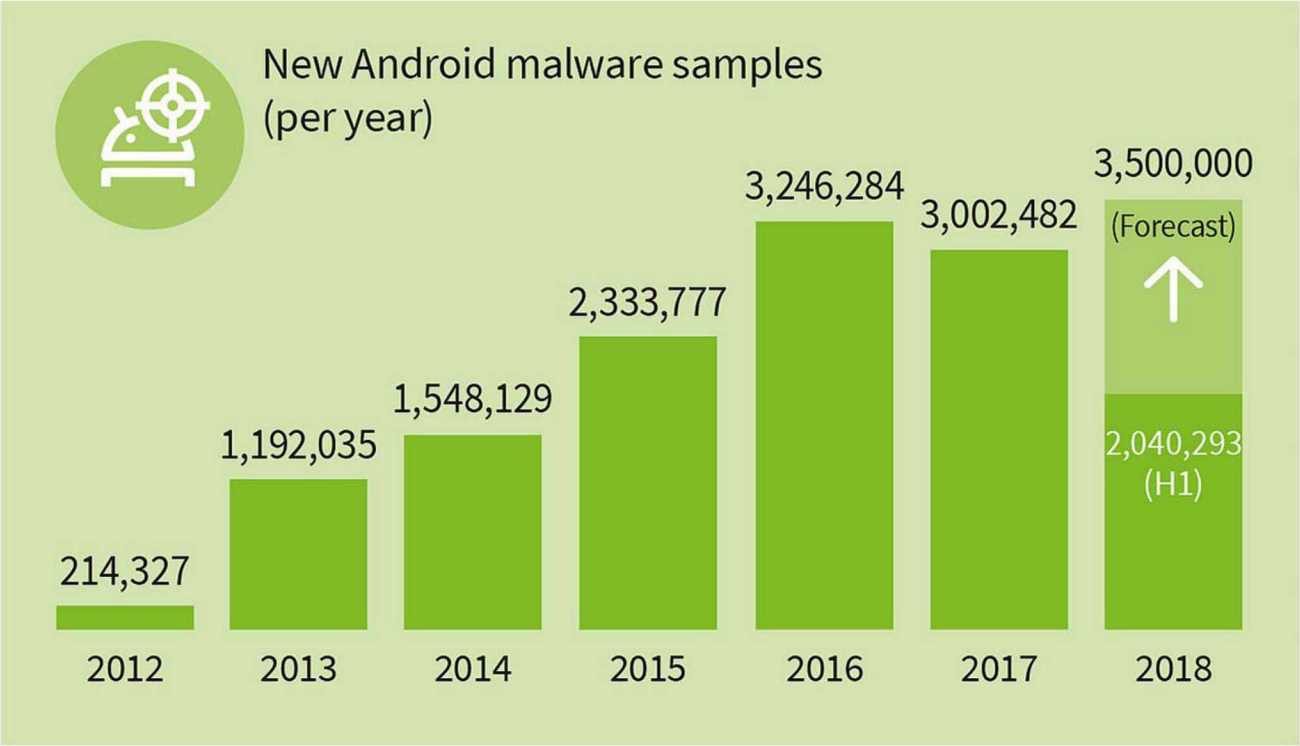
SAND2018-13921C

LoopMC: Using Loops for Malware Classification Resilient to Feature-aware Perturbations

Aravind Machiry, Nilo Redini, Eric Gustafson, Yanick Fratantonio, Yung Ryn Choe, Christopher Kruegel, and Giovanni Vigna

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Android Malware is a problem (still!??)



Used from: https://www.gdata.fr/news/2018/08/30994-menaces-sur-android-etat-des-lieux-pour-le-1er-semestre-2018

Loops are interesting

- Runtime-optimization techniques focus on loops.

- Previous work CLAPP (FSE 2015) from Yanick et.al shows that loops can encompass interesting behaviors about the apps.

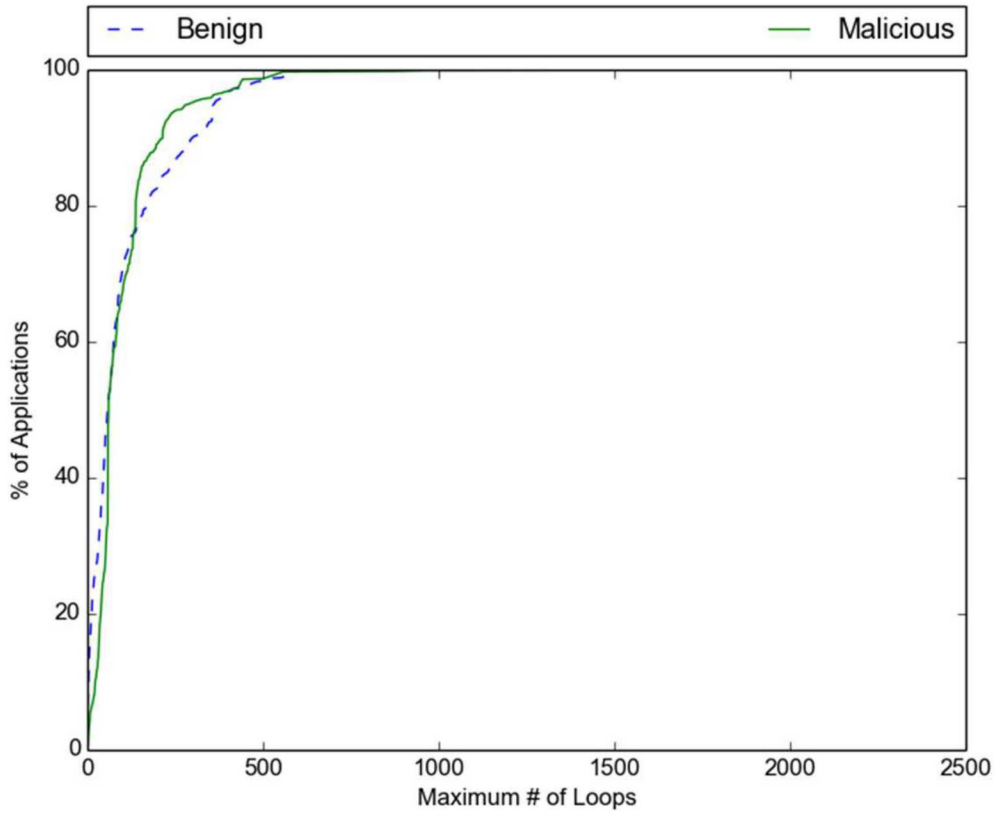
Loops are interesting

- Runtime-optimization techniques focus on loops.

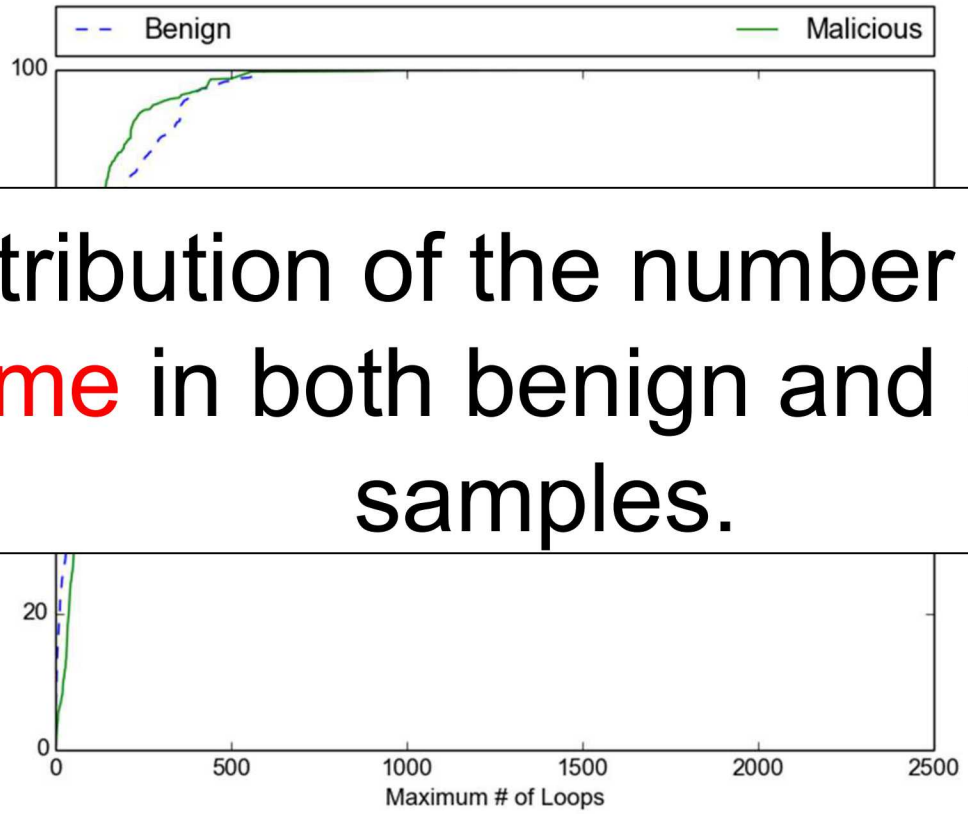
- Previous work CLAPP (FSE 2015) from Yanick et.al shows that loops can encompass interesting behaviors about the apps.

- ***Is the behavior captured by loops enough to detect malware?***

Are we lucky?


Benign
Malicious
100
80
60
40
20
0
% of Applications
0
500
1000
1500
2000
2500
Maximum # of Loops

Are we lucky? NO



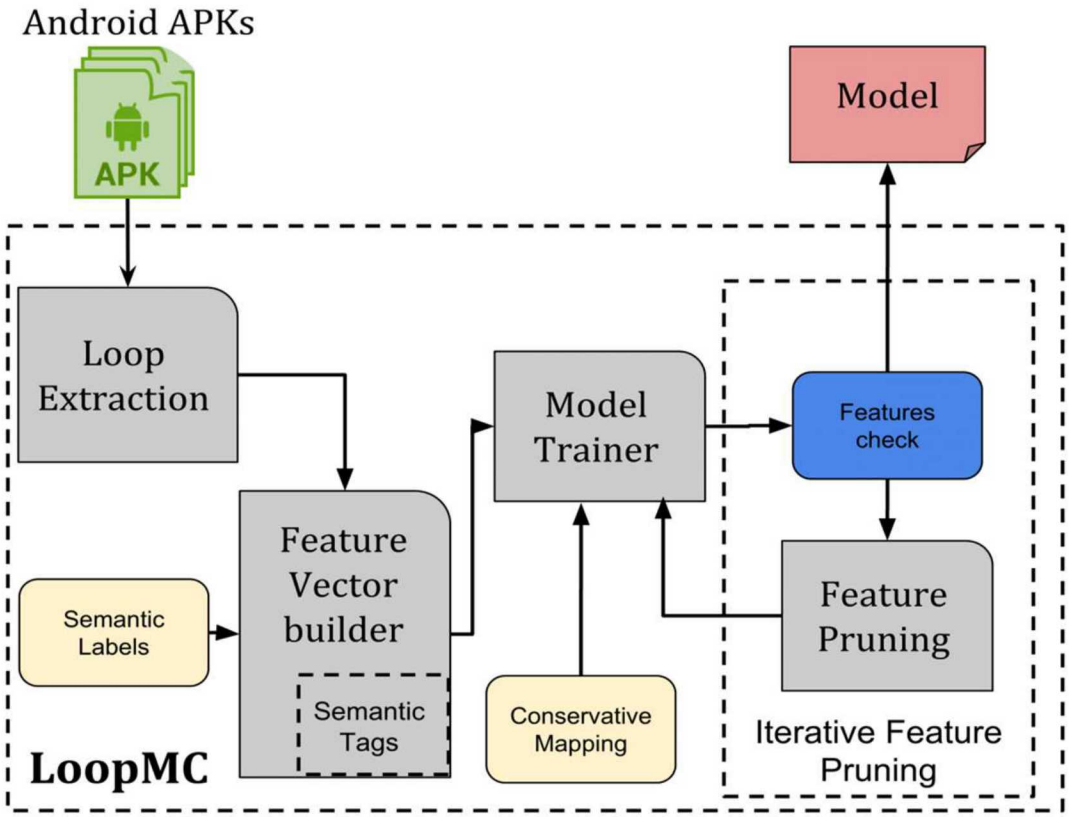
The distribution of the number of loops is the same in both benign and malicious samples.

LoopMC

- Using Loops for Malware Classification.

- Conservative mapping provides resilience against feature-unaware (or blind) perturbations.

LoopMC: Overview



LoopMC: Semantic Labels

- Semantic label to each of the Android API methods (Manually!!).

- Semantic labels capture the end-result of a method on the system.

- `java.io.FileOutputStream!!write` has the **same** label as `android.media.ExifInterface!!saveAttributes` i.e., **ioWrite**

LoopMC: Semantic Tags

- Set of Semantic Labels.

- Set of labels of all the API methods ***"reachable"*** from within a loop.

- Class Hierarchy Analysis (CHA) to handle dynamic dispatch.

LoopMC: Semantic Tag Example

```
do {
    obj = ite.next();
    if(!obj.exists()) {
      break;
    }
    if(obj.isFile()) {...}
} while(ite.hasNext())
```

LoopMC: Semantic Tag Example

```
do {
                iterator
    obj = ite.next();

    if(!obj.exists()) {   GenericFileOp

      break;

    }
                          GenericFileOp
    if(obj.isFile()) {...}

} while(ite.hasNext())   iterator
```

LoopMC: Semantic Tag Example

```
do {
                iterator
    obj = ite.next();

    if(!obj.exists()) {   GenericFileOp

      break;

    }
                          GenericFileOp
    if(obj.isFile()) {...}

} while(ite.hasNext())  iterator
```

Semantic Tag: {iterator, GenericFileOp}

LoopMC: Semantic Tag Example

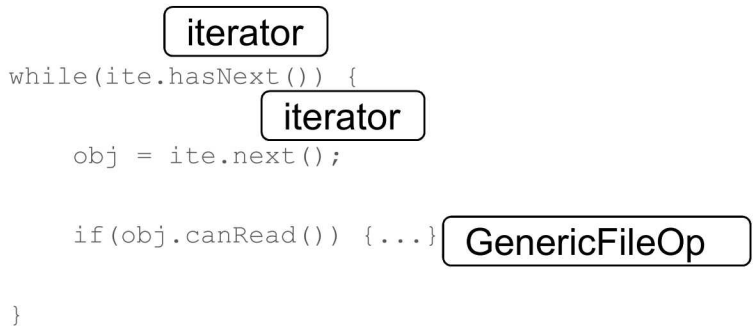
```
while(ite.hasNext()) {

    obj = ite.next();

    if(obj.canRead()) {...}

}
```

LoopMC: Semantic Tag Example



LoopMC: Semantic Tag Example

iterator

```
while(ite.hasNext()) {
```

iterator

```
    obj = ite.next();

    if(obj.canRead()) {...}
```

GenericFileOp

```
}
```

Semantic Tag:{iterator, GenericFileOp}

LoopMC: Semantic Tag Examples

```
do {
    obj = ite.next();
    if(!obj.exists()) {
      break;
    }
    if(obj.isFile()) {...}
} while(ite.hasNext())
```

iterator

GenericFileOp

GenericFileOp

iterator

Semantic Tag: {iterator, GenericFileOp}

```
while(ite.hasNext()) {
    obj = ite.next();
    if(obj.canRead()) {...}
}
```

iterator

GenericFileOp

Semantic Tag:{iterator, GenericFileOp}

LoopMC: Feature Vector

- Number of loops of each known semantic tag.

	SemTag1	SemTag2	SemTag3	...	SemTagN-1	SemTagN
app1	12	1	0	..	32	19
app2	1	0	31	...	1	0
...	...	...	...	...	...	...
appX	0	11	12	...	4	2

LoopMC: Model Training

- Decision trees: Random forest.
- Known to perform well for a large feature space, without much tuning.
- Accuracy: 99.3% (Genome) and 99.1% (VirusTotal)

LoopMC: Iterative Feature Pruning

- Many semantic tags, not all of them are important.
- Iteratively remove tags that have zero importance (based on the underlying model).
- Useful for conservative mapping.

LoopMC: Feature Vector

- Number of loops of each **known** semantic tag.

	SemTag1	SemTag2	SemTag3	...	SemTagN-1	SemTagN
app1	12	1	0	..	32	19
app2	1	0	31	...	1	0
...	...	...	...	...	...	...
appX	0	11	12	...	4	2

LoopMC: Conservative Mapping

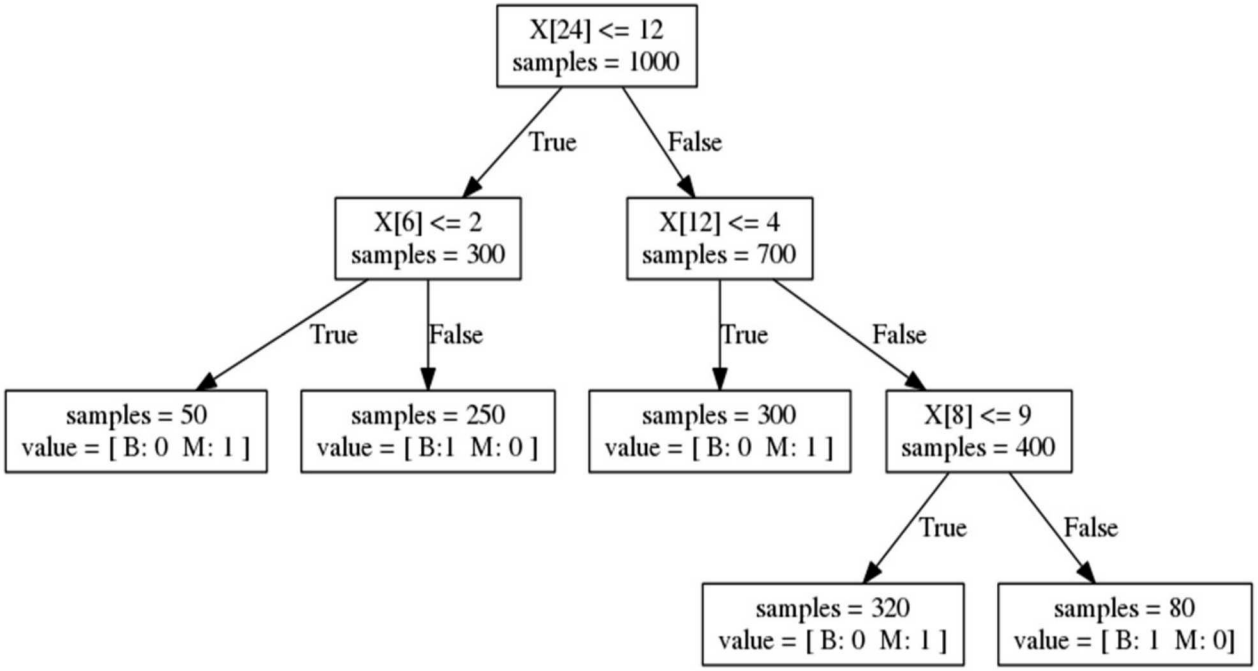
- Technique to handle unseen semantic tags.

- Map an unseen semantic tag to the closest known semantic tag.

- Resolving ties: **Higher Malware Importance (MI)**.

LoopMC: Malware Importance (MI)

- Value indicating a semantic tag ability to classify an app as malware.



LoopMC: Malware Importance (MI)

- Value indicating a semantic tag ability to classify an app as malware.

$$MI(semantic\_tag) = \begin{cases} \dfrac{(MS(tc)-MS(fc))}{N_{root}} & \textit{If } MS(tc) > MS(fc) \\ 0 & \textit{otherwise} \end{cases}$$

MS(tc) => Malware samples having the semantic tag value higher or equal to that of the threshold.

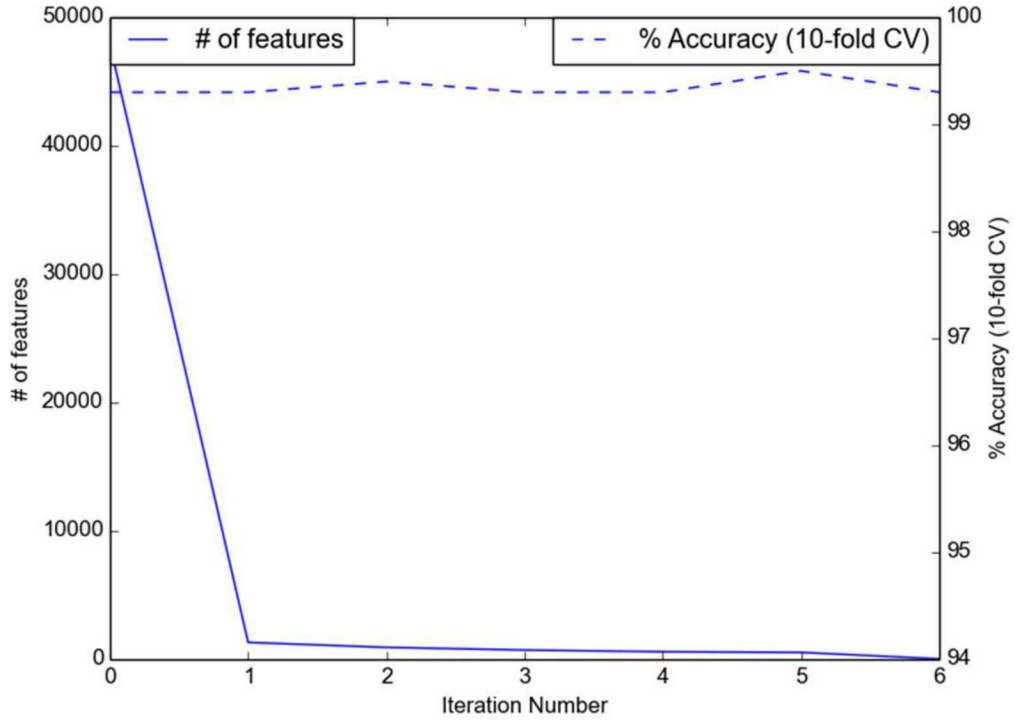
MS(fc) => Malware samples having the semantic tag value lower than the threshold.

LoopMC: Evaluation

- Dataset: Malware Genome and VirusShare.

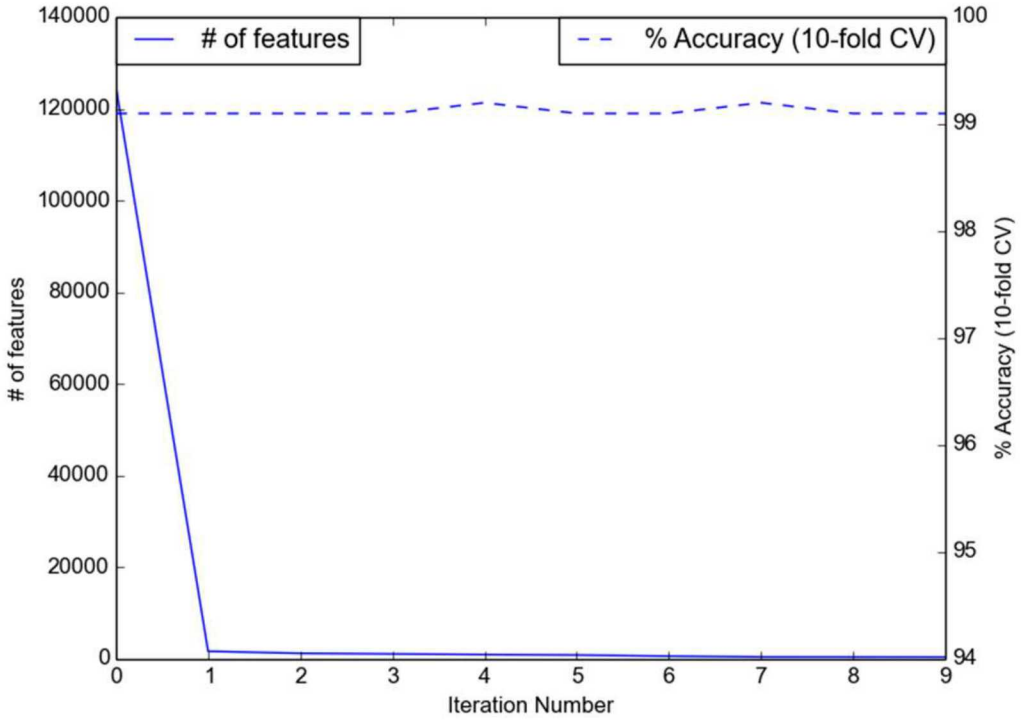
System	Malware Genome	VirusShare
LoopMC	**99.3%**	**99.1%**
DroidAPIMiner	99%	97.4%
Drebin	94%	-

LoopMC: Iterative Feature Pruning on Malware Genome



Accuracy remains almost constant as we prune the features.

LoopMC: Iterative Feature Pruning on VirusShare



Accuracy remains almost constant as we prune the features.

LoopMC: Are semantic labels the secret sauce?

- Using semantic tags for DroidAPIMiner resulted in accuracy of only **53.37% (almost random)**

- Using APIs *reachable* only from loops for DroidAPIMiner resulted in 96.15% accuracy.

LoopMC: Are semantic labels the secret sauce?

- Using semantic tags for DroidAPIMiner resulted in accuracy of only **53.37%** **(almost random)**

Semantic tags alone doesn't help. Looks like it is the combination of loops and semantic tags that helps.

Feature-unaware perturbations

- Evasion attempts unaware of the details of the underlying model.
    - CFG Obfuscation: Spurious and Infeasible loops
    - Reflection

CFG Obfuscation

- We used the ADAM tool to obfuscate all the apps.

Dataset	True Positive	False Positive
Malware Genome	**99.99%**	**2.01%**
VirusShare	**99.79%**	**5.45%**

Reflection

- Replaced all API calls with reflection: LoopMC classified all the samples as malware but DroidAPIMiner classified them as benign.

- Replaced only SMS related API calls with reflection:

System	Accuracy (on VirusShare)
LoopMC	92.47% (⇓**4.63%**)
DroidAPIMiner	83.54% (⇓**13.86%**)

Hardened feature vector

- Hard to affect the feature vector (without knowing the exact semantic tags).

- Brute-force techniques are impractical.

Limitations

- Static analysis evasion.
- Interactions with Android framework not modelled (yet!!).
- No loops malware.
- Splitting and Stitching the loops.

Conclusions

- Android malware detection using semantics of loops.
- Semantic labelling.
- Resilient to blind perturbations.
- Will be available at: https://github.com/ucsb-seclab/LoopMC

secLab

Thank You