Obfuscation Resilient Search Through Executable Classification

Fang-Hsiang Su*, Jonathan Bell§, Gail Kaiser*, **Baishakhi Ray***
*Columbia University, §George Mason University

Problem: Obfuscation Resilient Search

?

?

Macneto

Why does it matter?

Android apps are usually obfuscated

- Decrease executable size
- Reduce disallowed reuse such as plagiarism
- Hide the true intent of the executable: malware

Why does it matter?

A security analyst wants to review the application

- A malware analyst receives an unknown malware
- Checks if such malware is a va[...]ng malware

Search Problem

Popular Obfuscation Techniques

➢ **Lexical transformation**

- Replace identifier names
- Anonymize programs/executables

➢ **Control transformation**

- Change control flows

➢ **Data transformation**

- Encrypt/decrypt data, e.g., strings
- Might insert helper methods changing program structures

Obfuscation Example

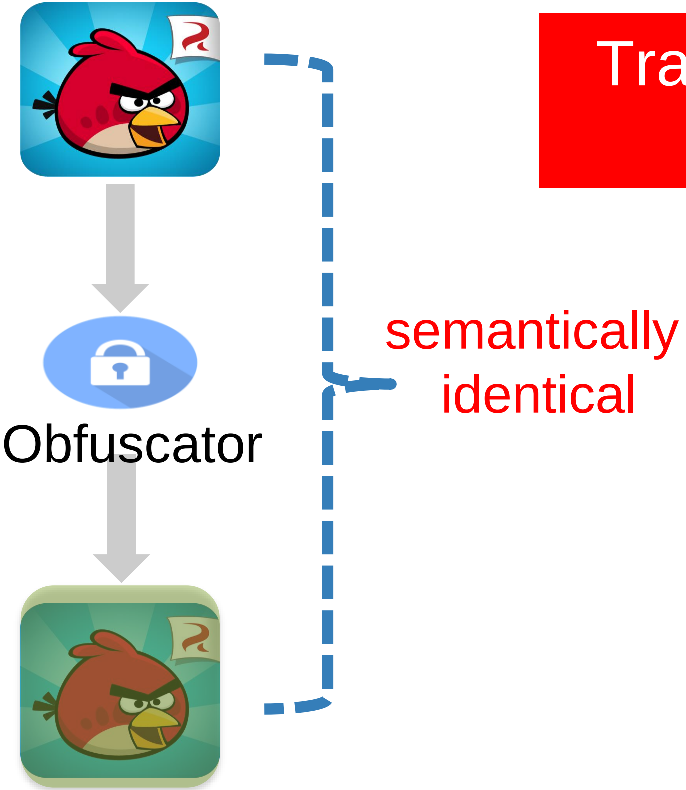
```
int fib(int n) {
  int a, b, c;
  a = 1;
  b = 1;
  if (n <= 1) return
  1;
  for (; n > 1; n--) {
    c = a + b;
    a = b;
    b = c;
  }
  return c;
  }
```

```
int f1(int r0) {
  int r1, r2, r3;
  r1 = 1;
  r2 = 1;
  if (r0 > 1) goto L22;
  return 1;
  L22: if (r0 <= 1)
  goto L23;
  r3 = (r1 + r2);
  r1 = r2;
  r2 = r3;
  r0--;
  goto L22;
  L23: return r3;
  }
```

Search to Deobfuscate

- Recover identifier names
- Classify programs/executables:
  - Given an unknown executable, what are other relevant executables?
  - Malware family identification
- Detect plagiarism
- Support analyst to discover semantic clusters among programs

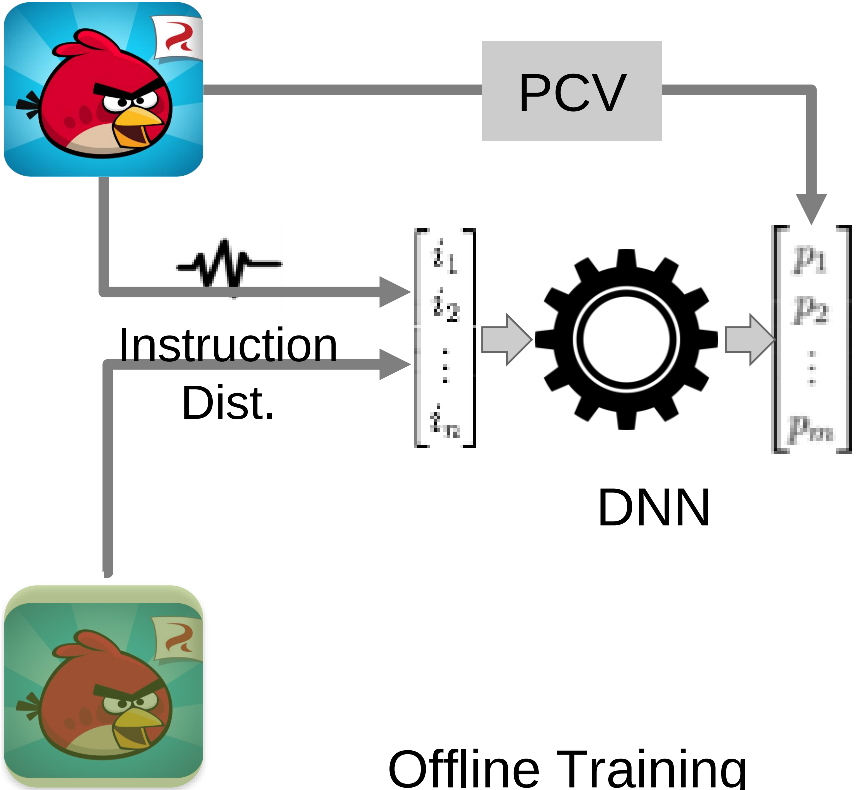
Macneto: Obfuscation Resilient Search



Train a DNN that can capture the semantic similarity

Represented by
Principal Component Vector (PCV)

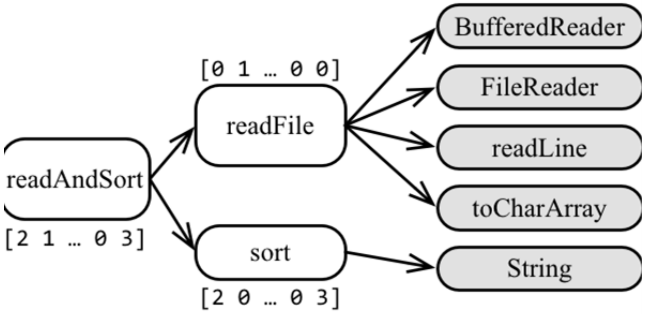
Macneto: Obfuscation Resilient Search



Offline Training

Macneto: Instruction Distribution

- A semantic proxy of application executables
- Use data flow analysis to collect potential methods
- InstructionDistribution(A) = Sum(InstructionDistribution(Method))



Macneto: PCA on Executables

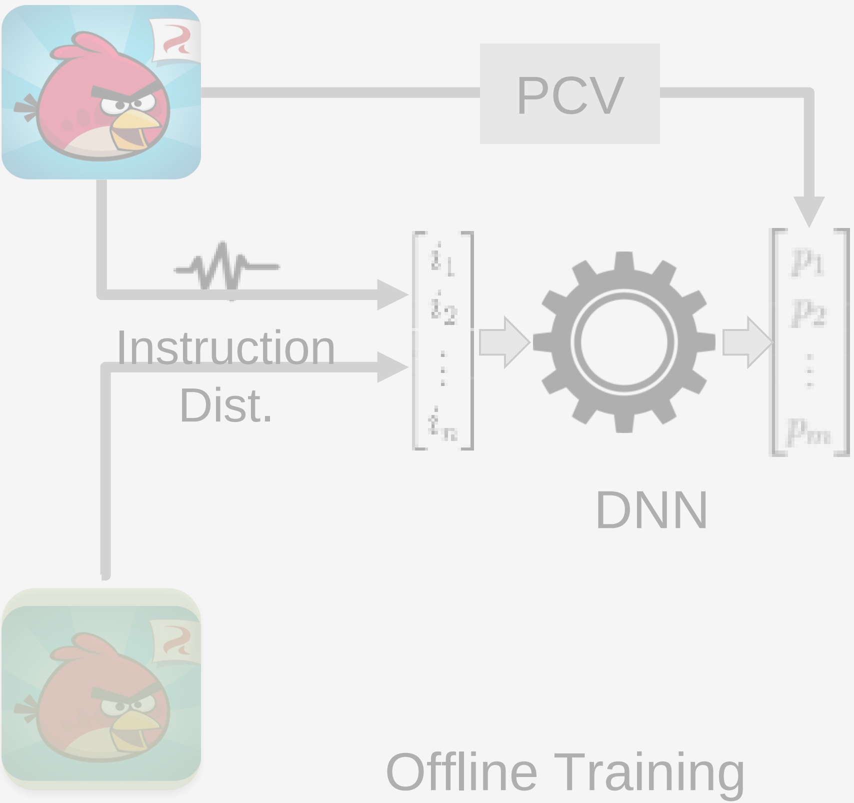
- PCA on instruction distribution
- Select important dimensions
- Reduce dimensions
  - 252 features (instruction types) ->32 dimensions
- Decrease search time

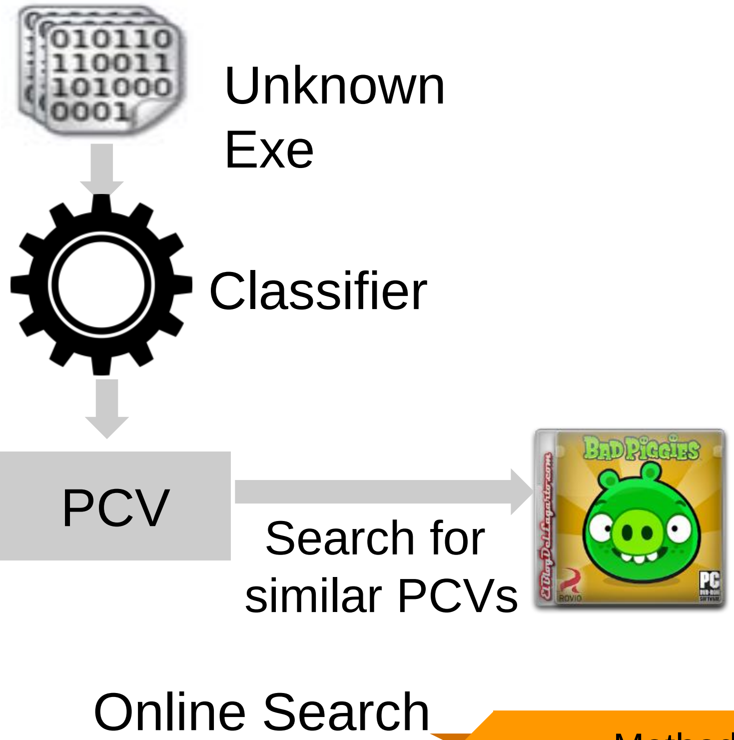
Table 1. Macneto's instruction set.

Opcode	Description
xaload	Load a primitive/object x from an array
xastore	Store a primitive/object x to an array
arraylength	Retrieve the length of an array.
xadd	Add two primitives of type x on the stack.
xsub	Subtract two primitives of type x on the stack.
xmul	Multiply two primitives of type x on the stack.
xdiv	Divide two primitives of type x on the stack.
xrem	Compute the remainder of two primitives x on the stack
xneg	Negate a primitive of type x on the stack.
xshift	Shift a primitive x (type integer/long) on the stack.
xand	Bitwise-and two primitives of type x (integer/long) on the stack.
xor	Bitwise-or two primitives of type x (integer/long) on the stack.
x_xor	Bitwise-xor two primitives x (integer/long) on the stack.
iinc	Increment an integer on the stack.
xcomp	Compare two primitives of type x on the stack
ifXXX	Represent all conditional jumps.
xswitch	Jump to a branch based on stack index.
android_apis	The APIs offered by the Android framework

Macneto: Obfuscation Resilient Search

Fork me on Github





Research Questions

- **RQ1: How precisely can Macneto retrieve relevant executables?**
  - Executable Search

- **RQ2: Given an unknown executable, can Macneto infer meaningful (human readable) keywords?**
  - Executable Understanding

Evaluation Settings

- **1,500+ Android apps from FDroid repository**

- **Systematically obfuscate apps by Allatori**
  - Anonymize apps
  - Change control flows
  - Encrypt data by inserting helper methods

- **Systems to evaluate**
  - Macneto
  - PCA: Using only PCA without deep learning to search
  - Naive: Using instruction distribution to search

Evaluation Metrics

- Given an obfuscated executable A’ as a query
- Mean Reciprocal Ranking: Multiplicative inverse of rank of A
- Top@K: if the rank of A is equal or better than Kth position. K= {1, 5, 10}
- Ex: A is returned by a search system with rank 2nd
  - MRR = ½
  - Top@1 = false, Top@5 = true

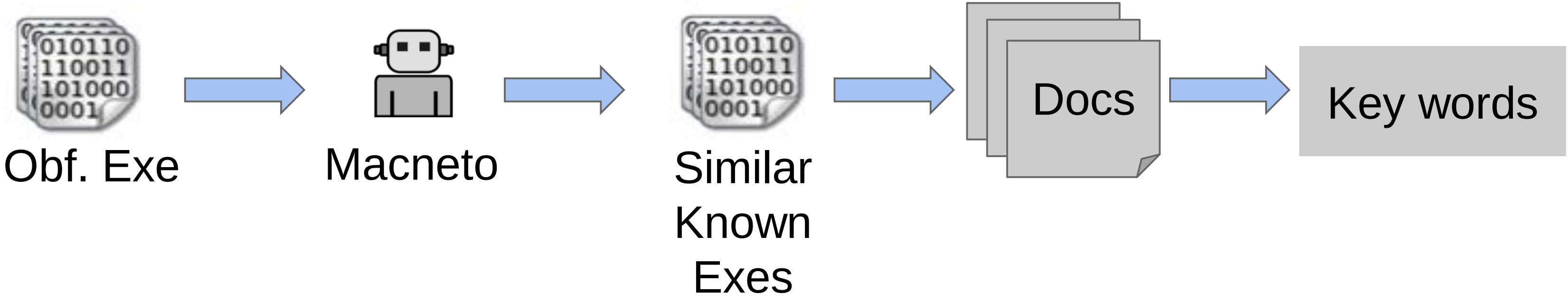
Result: Executable Search

- K-fold (8-fold) analysis: Each executable will be tested
- Here we present avg. values for 8 experiments
- Training APK: 1359, Testing APK: 200

	Training Time (s)	Query Time (s)	Top@1	MRR	Boost@1
Macneto	2845.7	24.09	0.80	0.86	17.76%
PCA	0.0354	20.13	0.74	0.82	8.32%
Naive	N/A	65.09	0.68	0.78	0.00%

Result: Executable Understanding

- Input: An unknown executable without human description
- Output: Key human words
- Find neighbors⇒ Leverage their descriptions (documents)



Result: Executable Understanding

- **net.bierbaumer.otp_authenticator**
  - Real description: “...two-factor authentication...scan the QR code...”
  - Macneto said: “security” and “QR”

Out of 20 test APKs, at least one meaningful keyword provided by:
- Macneto :14
- naïve approach: 7
- PCA: 4

Threat of Validity

- While we believe the generalizability of Macneto, only examine a single obfuscator
- Two executables may have different semantics. After adding noise by obfuscators, they may become more similar.
- DNN Hyper parameter tuning: more obfuscators, more layers

Future Work

- Larger scale experiments
  - More executables
  - More obfuscators
  - More types of instructions

- Other proxies to represent executable semantics
  - Auto-encoders

Conclusion

- **Goal**: precisely search for relevant executables, when the query is obfuscated
- **Macneto = Data flow analysis + PCA + Deep learning**
- Up to 84% search precision
- Potential to infer human keywords given unknown executables

https://github.com/Programming-Systems-Lab/macneto_release

Obfuscation Resilient Search Through Executable Classification

Fang-Hsiang Su*, Jonathan Bell§, Gail Kaiser*, **Baishakhi Ray***
*Columbia University, §George Mason University

Macneto: Learning

- Insight: Both original and obfuscated application exectuables share the same semantics⇒ same labels/classifications
- Input, ID(A_ori), ID(A_obf): Instruction distributions
- Output, PCV(A_ori): Principal Component Vector of original app
- Deep learning minimizes

$$J(\Theta) = \sum_{A_j \in T} \|PCV(A_j) - l(\theta^{(3)} \cdot g(\theta^{(2)} \cdot f(\theta^{(1)} \cdot A_j)))\|^2$$
$$+\|PCV(A_j) - l(\theta^{(3)} \cdot g(\theta^{(2)} \cdot f(\theta^{(1)} \cdot A_j^{ob})))\|^2$$

Macneto : Code (Executable) Search

- Given an unknown executable, the classifier predicts its PCV
- Using this PCV to search for the most similar application in the existing codebase
- This similar application can be the original version of this unknown executable, even it is obfuscated
- Understand executables by inferring human words