DexLego: Reassembleable Bytecode Extraction for Aiding Static Analysis

Zhenyu Ning and Fengwei Zhang

COMPASS Lab
Wayne State University

Outline

WAYNE STATE UNIVERSITY

- Introduction
- System Overview
- Implementation
- Evaluation
- Conclusions

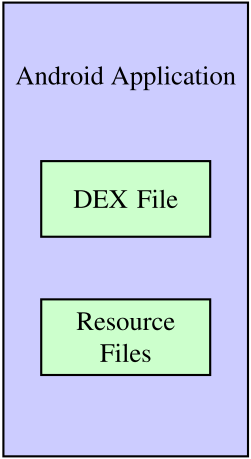
Outline

WAYNE STATE UNIVERSITY

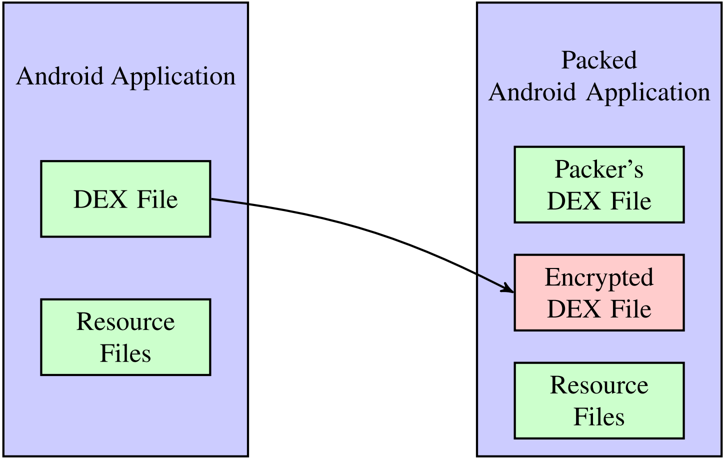
- Introduction
- System Overview
- Implementation
- Evaluation
- Conclusions

Why still doing static analysis?

Packing Technique



Packing Technique



Packing Technique

WAYNE STATE UNIVERSITY

- Previous Android Unpacking Systems: DexHunter [1], AppSpear [2]
  - Assuming a clear transition between the packer's code and the original code.
  - Using method-level collection to collect code.

Self-modifying Code

```java
public void example() {
    helper();
    Log.d("Hello World!");
}
```

Self-modifying Code

WAYNE STATE UNIVERSITY

```java
public void example() {
    helper();
    Log.d("Hello World!");
}
```

```java
public void example() {
    helper();
    Log.d("Password is 1234!")
}
```

- Android allows the applications modify its bytecode at runtime.
- These changes cannot be detected by current static analysis tools.

Reflection

WAYNE STATE UNIVERSITY

```java
public class Main extends Activity {
   protected void onCreate(Bundle savedInstanceState) {
     //...
     // 55 is the index of method "reflectiveLeak" in the method array of class "Main"
     Method reflectiveLeakMethod = getClass().getMethods()[55];
     reflectiveLeakMethod.invoke(this, "sensitive data");
   }

   public void reflectiveLeak(String data) {
     // leak data
     SmsManager.getDefault().sendTextMessage("800-123-456", null, data, null, null);
   }
}
```

Dynamic Code Loading

- Additional DEX files can be downloaded from cloud at runtime.
- Malicious activities in these DEX files are ignored by current static analysis tools.

Why not using dynamic analysis?

Challenges of Dynamic Analysis

- Implicit Flows
- Performance Overhead vs Accuracy

Our Goal

Wayne State University

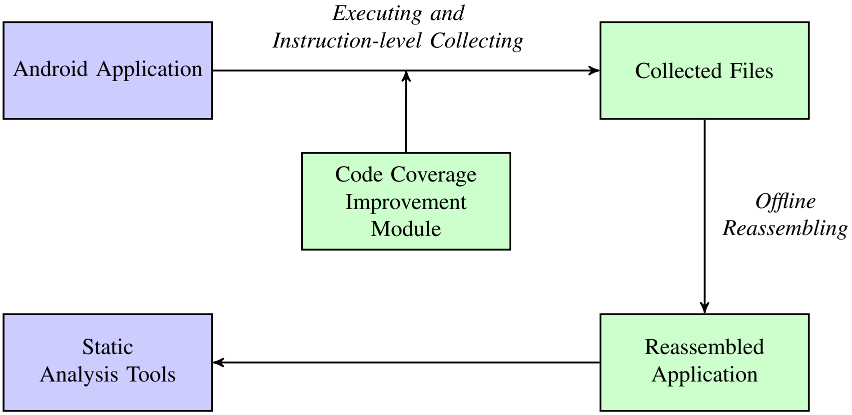
DexLego: Instruction-level collecting and offline reassembling

- Use dynamic approach to collect executed instruction.
- Improve the current static analysis via offline reassembling.

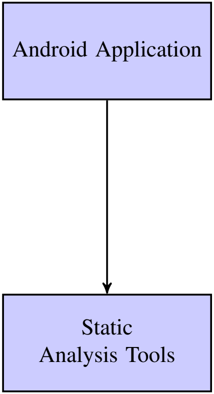
Outline

WAYNE STATE UNIVERSITY

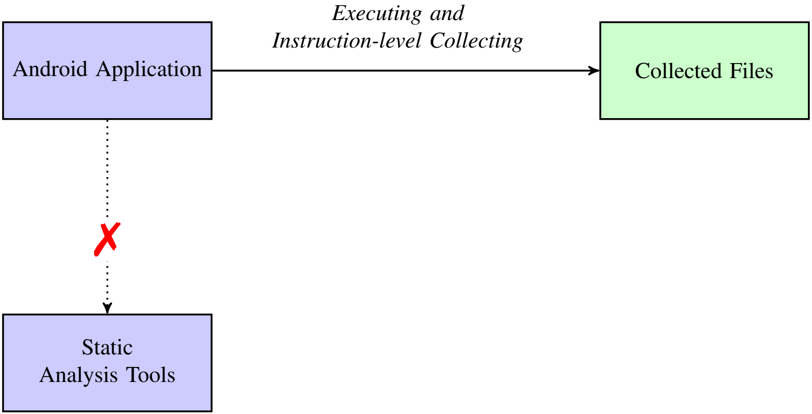
System Overview



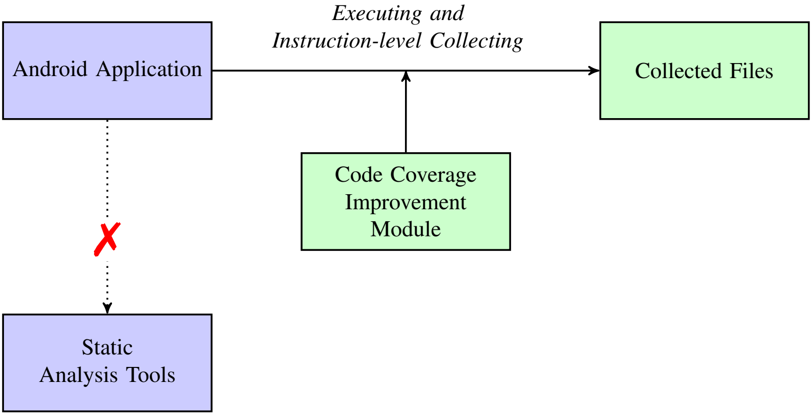
System Overview



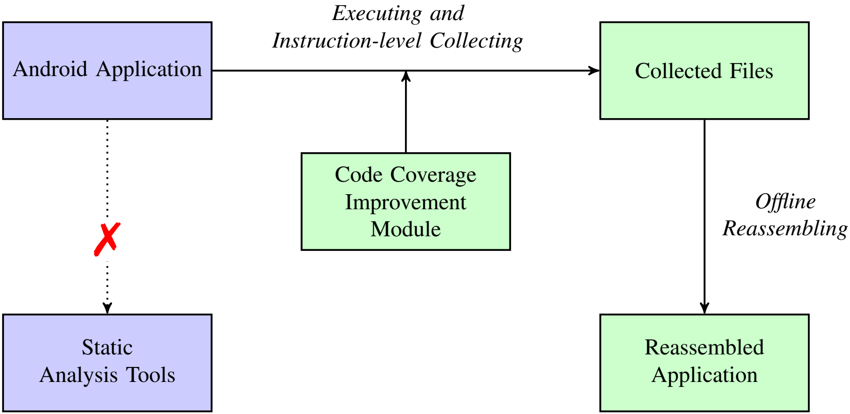
System Overview



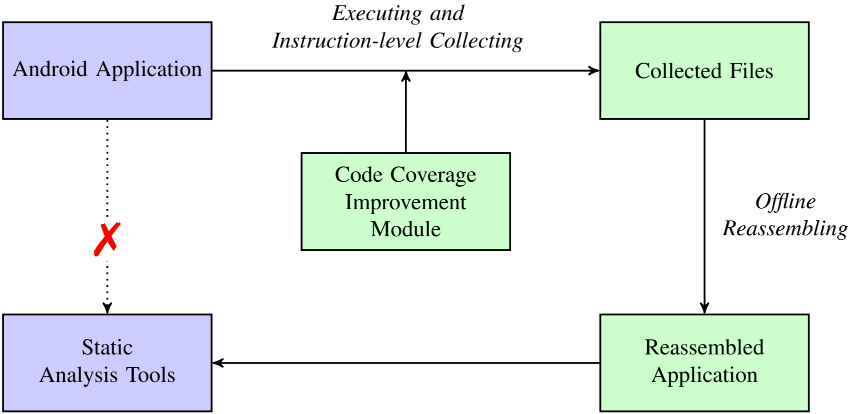
System Overview



System Overview



System Overview



Outline

WAYNE STATE UNIVERSITY

- Introduction
- System Overview
- Implementation
- Evaluation
- Conclusions

Simply list the executing instructions one-by-one?

Instruction-level Collection

WAYNE STATE UNIVERSITY

```java
public void example() {
    int i;
    for (i = 0; i < 10000; ++i) {
        Log.d("Hello World!");
    }
}
```

Instruction-level Collection

```
public void example() {
    int i;
    for (i = 0; i < 10000; ++i) {
        Log.d("Hello World!");
    }
}
```

```
public void example() {
    Log.d("Hello World!");
    Log.d("Hello World!");
    Log.d("Hello World!");
    Log.d("Hello World!");
    Log.d("Hello World!");
    Log.d("Hello World!");
    Log.d("Hello World!");
    Log.d("Hello World!");
    ...
}
```

Instruction-level Collection

WAYNE STATE UNIVERSITY

- Record both the index and the content of each instruction.
- Instruction with same index and content will not be repeatedly collected.

How about self-modifying code?

Instruction-level Collection

- **Collection Tree**
  - A independent tree for each execution of each method.
  - Each node indicates a piece of changed code.

Instruction-level Collection

```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         benign();
13         helper();
14     }
15 }
```

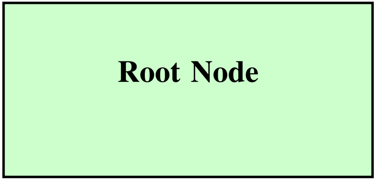
Instruction-level Collection

```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         benign();
13         helper();
14     }
15 }
```

```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         malicious();
13         helper();
14     }
15 }
```

Instruction-level Collection

```java
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         benign();
13         helper();
14     }
15 }
```



Instruction-level Collection

i = 0

```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         benign();
13         helper();
14     }
15 }
```

Root Node

for (int i = 0; i < 2; ++i) {

}

Instruction-level Collection

WAYNE STATE UNIVERSITY

i = 0

```
1   // No malicious activities
2   public void benign() {}
3
4   // Leak data
5   public void malicious() {}
6
7   // Modify line 12 to
        "malicious()" at runtime
8   public void helper() {}
9
10  public void execute() {
11      for (int i = 0; i < 2; ++i) {
12          benign();
13          helper();
14      }
15  }
```

Root Node

```
for (int i = 0; i < 2; ++i) {
        benign();

}
```

Instruction-level Collection

WAYNE STATE UNIVERSITY

i = 0

```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         benign();
13         helper();
14     }
15 }
```

Root Node

```
for (int i = 0; i < 2; ++i) {
    benign();
    helper();
}
```

Instruction-level Collection

WAYNE STATE UNIVERSITY

i = 1

```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         malicious();
13         helper();
14     }
15 }
```

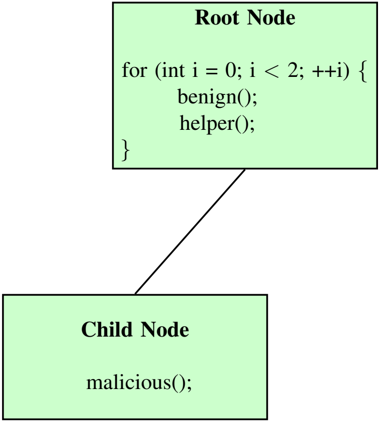
Root Node

```
for (int i = 0; i < 2; ++i) {
        benign();
        helper();
}
```

Instruction-level Collection

i = 1

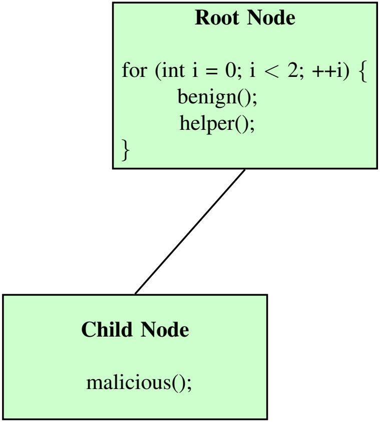
```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         malicious();
13         helper();
14     }
15 }
```



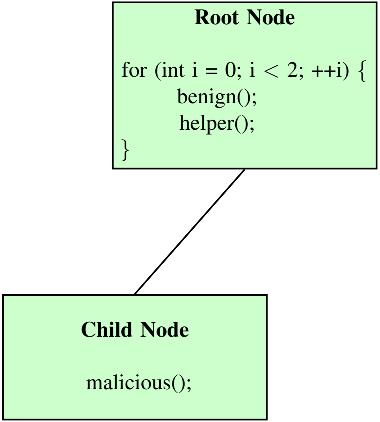
Instruction-level Collection

i = 1

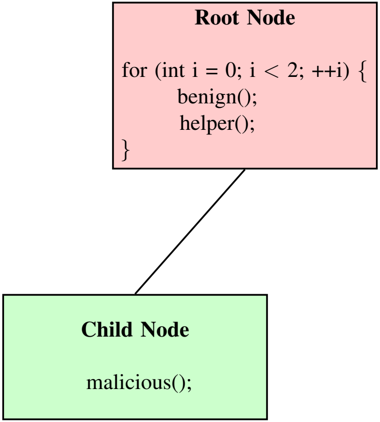
```
1  // No malicious activities
2  public void benign() {}
3
4  // Leak data
5  public void malicious() {}
6
7  // Modify line 12 to
       "malicious()" at runtime
8  public void helper() {}
9
10 public void execute() {
11     for (int i = 0; i < 2; ++i) {
12         malicious();
13         helper();
14     }
15 }
```



Bytecode Reassembling

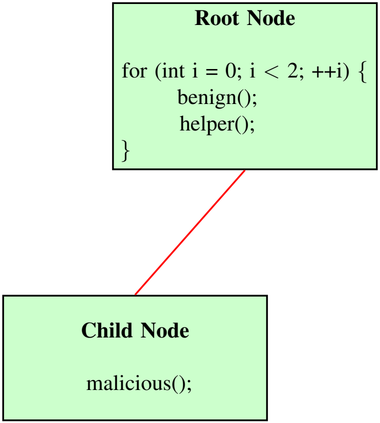


Bytecode Reassembling




```java
public void execute() {
    for (int i = 0; i < 2; ++i) {
        benign();
        helper();
    }
}
```

Bytecode Reassembling

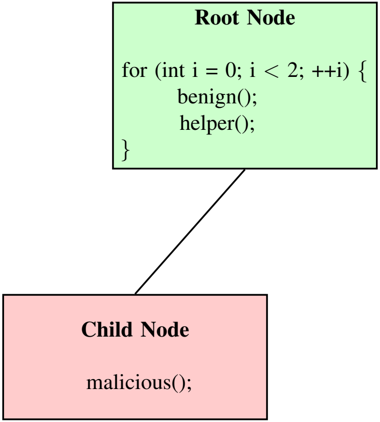



```
public void execute() {
    for (int i = 0; i < 2; ++i) {
        if (RANDOM_VALUE) {
            benign();
        } else {

        }
        helper();
    }
}
```

Bytecode Reassembling

WAYNE STATE UNIVERSITY




```
public void execute() {
    for (int i = 0; i < 2; ++i) {
        if (RANDOM_VALUE) {
            benign();
        } else {
            malicious();
        }
        helper();
    }
}
```

Implementation

WAYNE STATE UNIVERSITY

- Replace the reflective calls with the direct calls during bytecode collection.
- Use force execution to improve the code coverage.

Outline

- Introduction
- System Overview
- Implementation
- Evaluation
- Conclusions

Evaluation

WAYNE STATE UNIVERSITY

- Testbed Specification
  - LG Nexus 5X
  - A dual-core 1.8 GHZ Cortex-A57 cluster and a quad-core 1.4 GHZ Cortex-A53 cluster
  - Android 6.0 and TWRP Recovery

Can we correctly reconstruct the behavior of apps?

Evaluation

Table: Test Result of Different Packers.

Applications	HTMLViewer	Calculator	Calendar	Contacts
# of Instructions	217	2,507	78,598	103,602
360 [3]	✓	✓	✓	✓
Alibaba [4]	✓	✓	✓	✓
Tencent [5]	✓	✓	✓	✓
Baidu [6]	✓	✓	✓	✓
Bangcle [7]	✓	✓	✓	✓

WAYNE STATE UNIVERSITY

How is DexLego comparing with other tools?

Evaluation

- 134 samples from DroidBench [8].
- 3 static analysis tools: FlowDroid [9], DroidSafe [10], HornDroid [11].
- 2 unpacking tools: DexHunter [1], AppSpear [2].

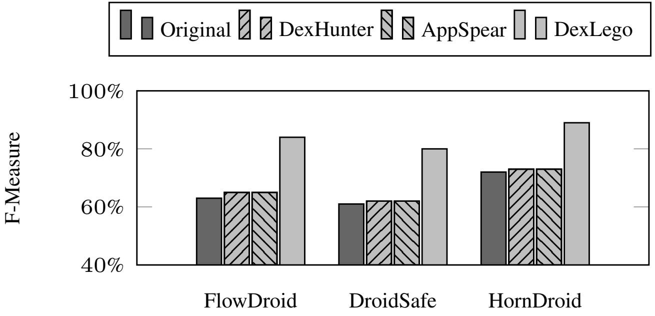


Figure: F-Measure of DroidBench samples.

Can DexLego work with real-world packed apps?

Evaluation

Table: Analysis Result of Packed Real-world Applications.

Package Name	Version	# of Installs	Original	Reassembled
com.lenovo.anyshare	3.6.68	100 million	0	4
com.moji.mjweather	6.0102.02	1 million	0	5
com.rongcai.show	3.4.9	100 thousand	0	3
com.wawoo.snipershootwar	2.6	10 million	0	4
com.wawoo.gunshootwar	2.6	10 million	0	5
com.alex.lookwifipassword	2.9.6	100 thousand	0	2
com.gome.eshopnew	4.3.5	15.63 million	0	3
com.szzc.ucar.pilot	3.4.0	3.59 million	0	5
com.pingan.pabank.activity	2.6.9	7.9 million	0	14

How about code coverage?

Evaluation

Table: Samples from F-Droid [12].

Package Name	Version	# of Instructions
be.ppareit.swiftp	2.14.2	8,812
fr.gaulupeau.apps.InThePoche	2.0.0b1	29,231
org.gnucash.android	2.1.7	56,565
org.liberty.android.fantastischmemopro	10.9.993	57,575
com.fastaccess.github	2.1.0	93,913

Table: Code Coverage with F-Droid Applications.

	Class	Method	Line	Branch	Instruction
Sapienz [13]	44%	37%	32%	20%	32%
Sapienz + DexLego	87%	88%	82%	78%	82%

Performance overhead?

Evaluation

- 7.5x, 1.4x, and 2.3x overhead on Java score, native score, and overall score, respectively.

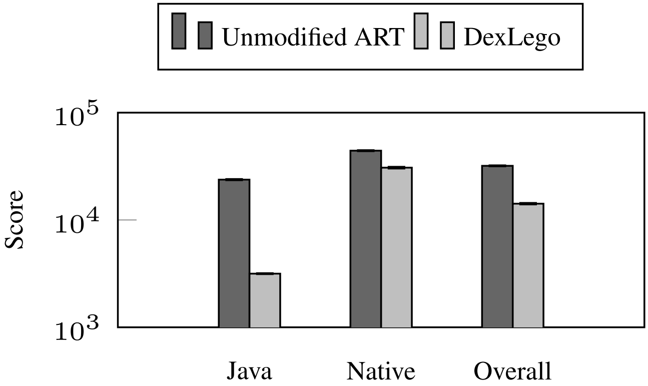


Figure: Performance Measured by CF-Bench [14].

Outline

- Introduction
- System Overview
- Implementation
- Evaluation
- Conclusions

Conclusions

- DexLego leverages instruction-level bytecode collecting and offline reassembling to aid existing static analysis tools.

- It helps to overcome the weakness of static analysis and increases the analysis accuracy with reasonable performance overhead.

References I

WAYNE STATE UNIVERSITY

[1] Y. Zhang, X. Luo, and H. Yin, "DexHunter: Toward extracting hidden code from packed Android applications," in *Proceedings of the 20th European Symposium on Research in Computer Security (ESORICS'15)*., 2015.

[2] W. Yang, Y. Zhang, J. Li, J. Shu, B. Li, W. Hu, and D. Gu, "AppSpear: Bytecode decrypting and DEX reassembling for packed Android malware," in *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions and Defenses (RAID'15)*, 2015.

[3] Qihoo 360 Inc., "360Protector," http://jiagu.360.cn/protection, 2014.

[4] Alibaba Inc., "AliProtector," http://jaq.alibaba.com/, 2014.

[5] Tencent Inc., "TencentProtector," http://legu.qcloud.com/, 2014.

[6] Baidu Inc., "BaiduProtector," http://app.baidu.com/jiagu/, 2014.

[7] Bangcle Ltd., "BangcleProtector," https://www.bangcle.com/, 2013.

[8] EC SPRIDE Secure Software Engineering Group, "DroidBench," https://github.com/secure-software-engineering/DroidBench, 2013.

[9] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "FlowDroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'14)*, 2014.

[10] M. I. Gordon, D. Kim, J. H. Perkins, L. Gilham, N. Nguyen, and M. C. Rinard, "Information flow analysis of Android applications in DroidSafe," in *Proceedings of the 22nd Network and Distributed System Security Symposium (NDSS'15)*, 2015.

[11] S. Calzavara, I. Grishchenko, and M. Maffei, "HornDroid: Practical and sound static analysis of Android applications by SMT solving," in *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P'16)*, 2016.

References II

[12] F-Droid, "F-Droid," https://f-droid.org/, 2011.

[13] K. Mao, M. Harman, and Y. Jia, "Sapienz: Multi-objective automated testing for Android applications," in *Proceedings of the 25th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'16)*, 2016.

[14] Chainfire, "CF-Bench," https://play.google.com/store/apps/details?id=eu.chainfire.cfbench, 2013.

Thank you!

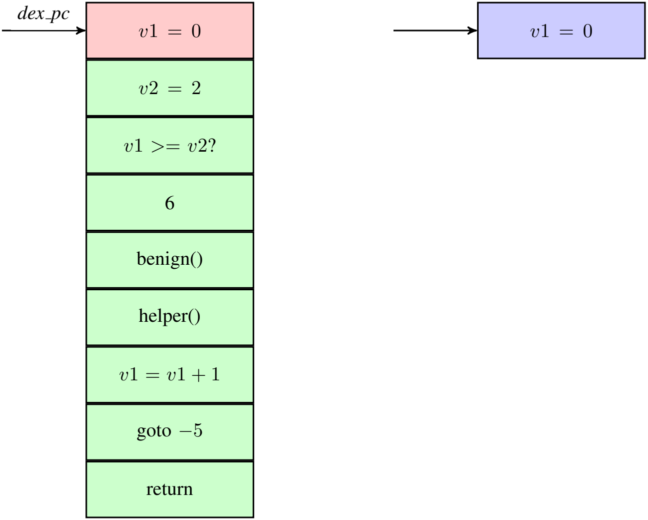
WAYNE STATE UNIVERSITY

Questions?

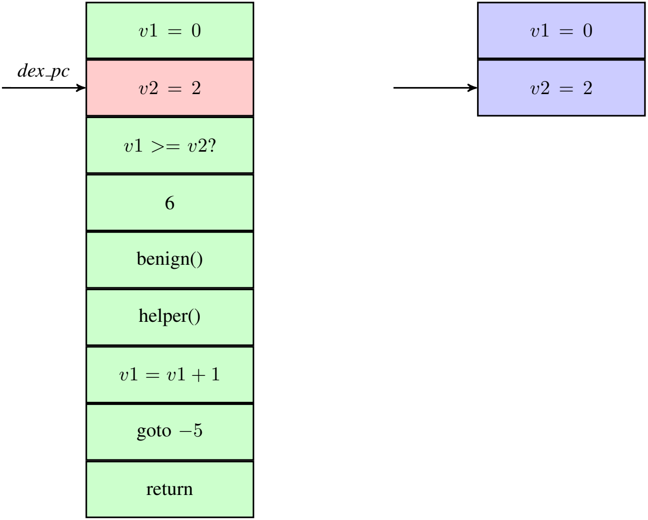
zhenyu.ning@wayne.edu

`http://compass.cs.wayne.edu`

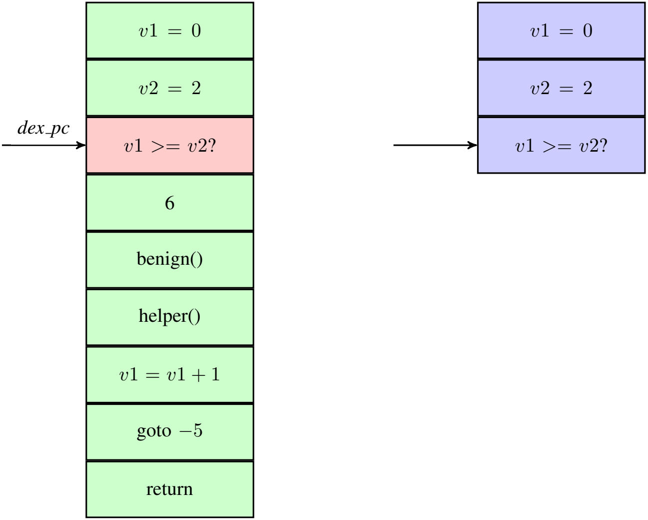
Branch Instruction



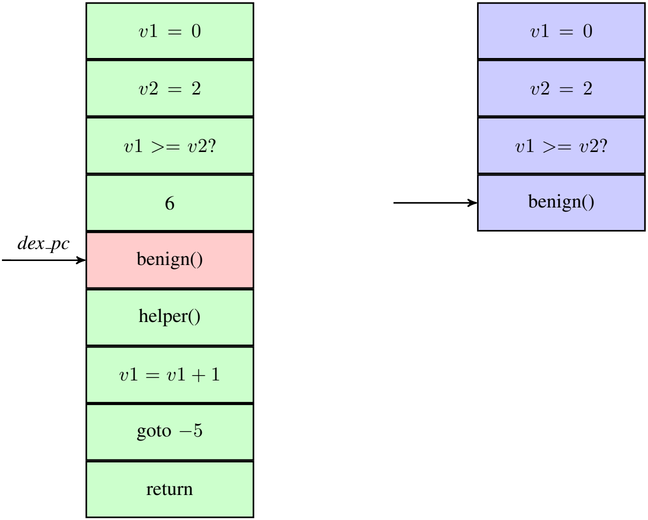
Branch Instruction



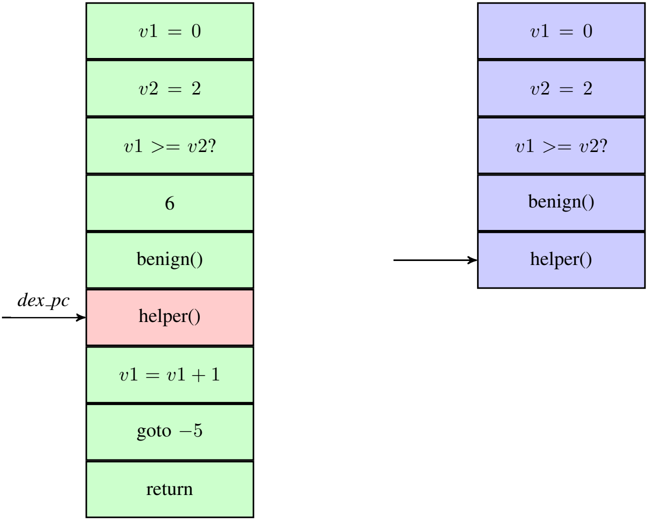
Branch Instruction



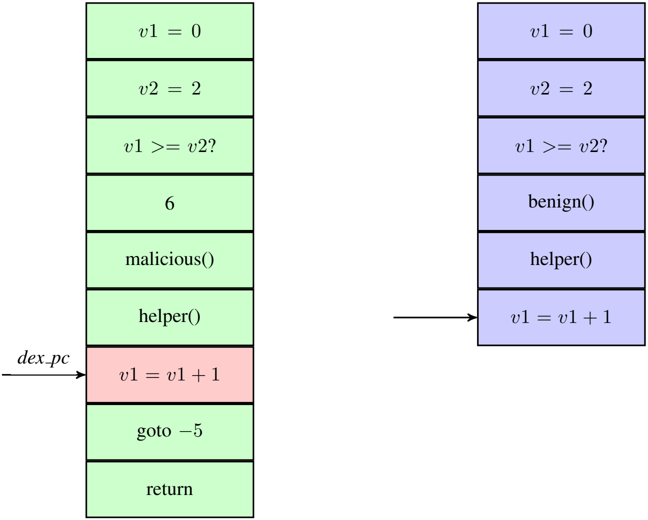
Branch Instruction



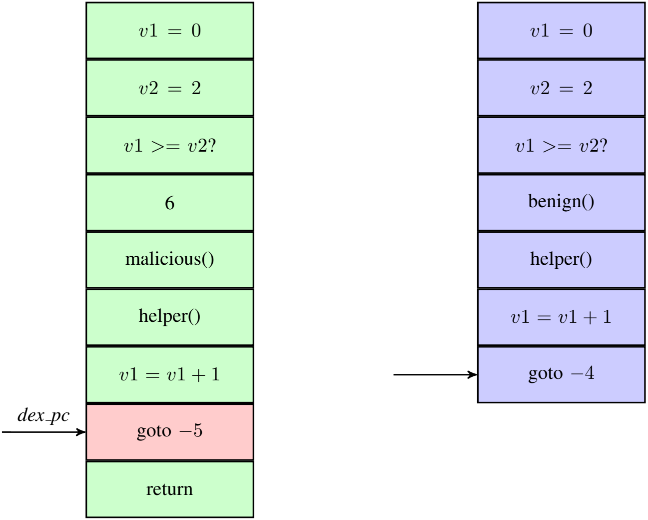
Branch Instruction



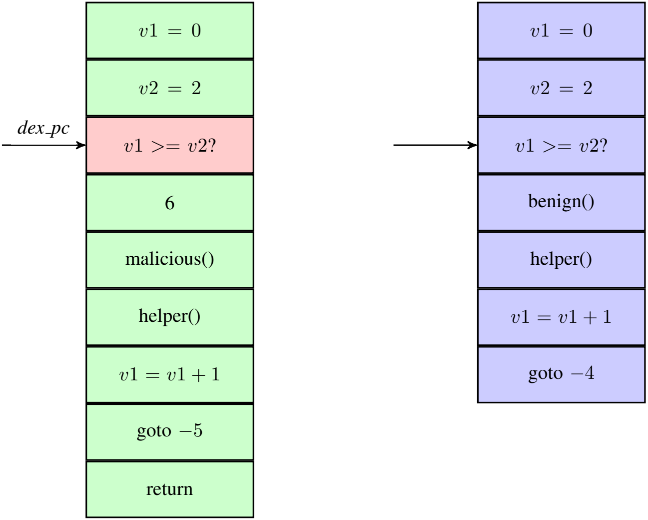
Branch Instruction



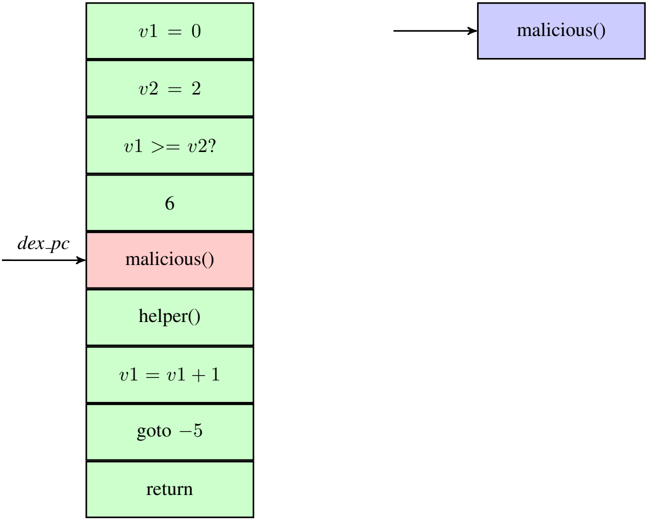
Branch Instruction



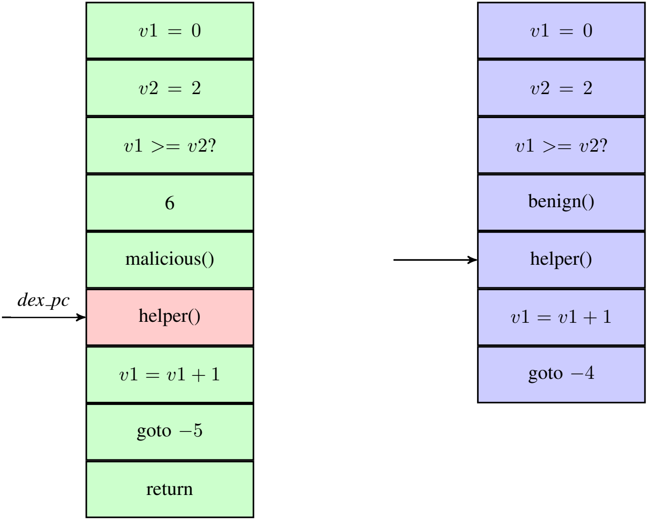
Branch Instruction



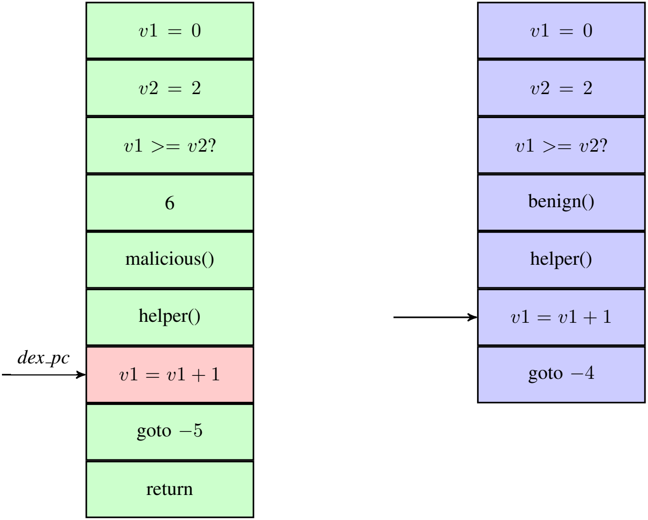
Branch Instruction



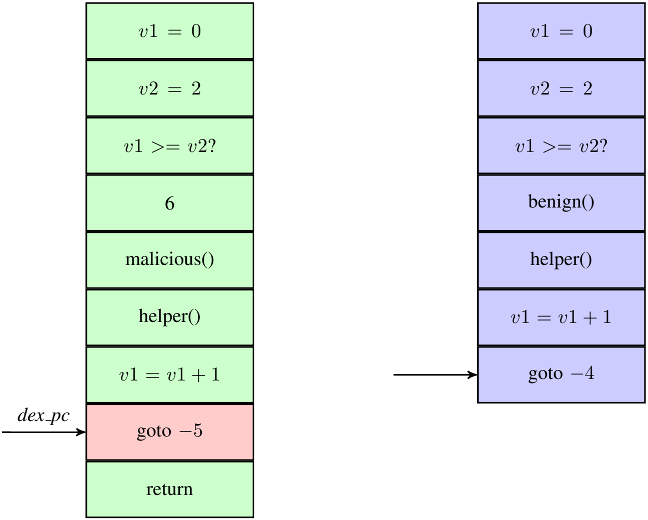
Branch Instruction



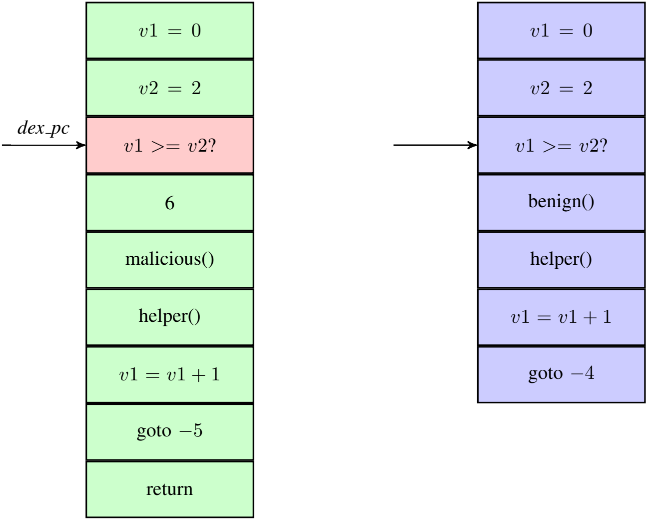
Branch Instruction



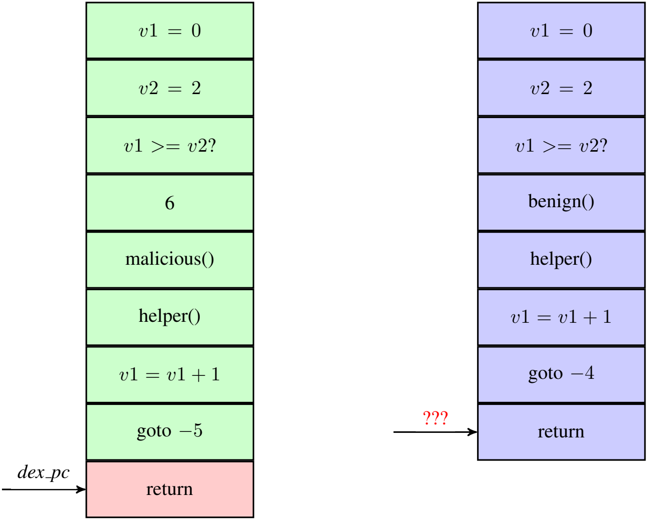
Branch Instruction



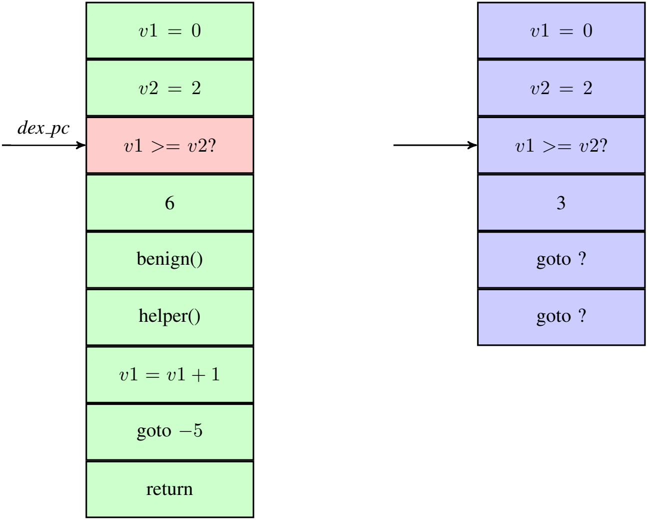
Branch Instruction



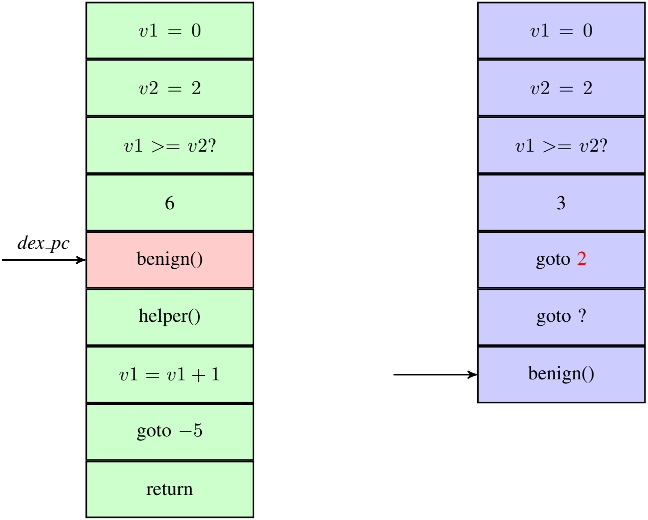
Branch Instruction



Branch Instruction



Branch Instruction



Branch Instruction