

Extending and Evaluating a Control Flow Obfuscation Technique for JVM Applications Utilizing `invokedynamic` with Native Bootstrapping

Bachelor's Thesis

Andre Blanke

Juli 16, 2022

Abstract

TODO

Contents

1	Motivation	2
2	Background	2
2.1	Obfuscation	2
2.2	Java Native Interface	2
2.3	<code>invokedynamic</code>	2
2.4	Proposed Technique	2
3	Implementation	2
3.1	Obfuscation Process	2
3.2	Limitations	3
4	Evaluation	3
4.1	Performance Overhead	3
4.2	Bytecode Size Inflation	3
4.3	Obfuscation Level	3
5	Future Work	3
6	Conclusion	3
7	References	3

1 Motivation

2 Background

2.1 Obfuscation

2.2 Java Native Interface

2.3 invokedynamic

2.4 Proposed Technique

3 Implementation

3.1 Obfuscation Process

```
1 .class public HelloWorld
2 .super java/lang/Object
3
4 .method public static main : ([Ljava/lang/String;)V
5     .stack 2
6     .locals 1
7     getstatic java/lang/System out Ljava/io/PrintStream;
8     ldc "Hello, world!"
9     invokevirtual java/io/PrintStream println (Ljava/lang/Object;)V
10    return
11 .end method
```

```
1 .class public HelloWorld
2 .super java/lang/Object
3
4 .method public static main : ([Ljava/lang/String;)V
5     .stack 2
6     .locals 1
7     invokestatic HelloWorld out ()Ljava/io/PrintStream;
8     ldc "Hello, world!"
9     invokevirtual java/io/PrintStream println (Ljava/lang/Object;)V
10    return
11 .end method
12
13 .method private static synthetic out : ()Ljava/io/PrintStream;
14     .stack 1
15     .locals 0
16     getstatic java/lang/System out Ljava/io/PrintStream;
17     areturn
18 .end method
```

3.2 Limitations

4 Evaluation

4.1 Performance Overhead

4.2 Bytecode Size Inflation

4.3 Obfuscation Level

4.3.1 Ease of Recognition

4.3.2 Attack Resilience

5 Future Work

6 Conclusion

7 References