Software Protection via Obfuscation Techniques

Ciprian LUCACI

Outline

Outline

- **Context** – Software Protection

Outline

- **Context** – Software Protection
- **Obfuscation** – Techniques and Tools

Outline

- **Context** – Software Protection
- **Obfuscation** – Techniques and Tools
- **Virtualization Obfuscation** – *VOT4CS* Tool Design

Outline

- **Context** – Software Protection
- **Obfuscation** – Techniques and Tools
- **Virtualization Obfuscation** – *VOT4CS* Tool Design
- **Evaluation** – Selection Criteria

Outline

- **Context** – Software Protection

- **Obfuscation** – Techniques and Tools

- **Virtualization Obfuscation** – *VOT4CS* Tool Design

- **Evaluation** – Selection Criteria

- **Conclusion** – To Obfuscate Or Not To Obfuscate

Context: **Software protection**

- **Threat Models**
  1. Malicious Code – Software security
  2. Malicious Host – Software protection

Context: **Software protection**

- **Threat Models**
  1. Malicious Code – Software security
  2. Malicious Host – Software protection

- **Attack Scenario**
  - Man-At-The-End (MATE)

Context: **Software protection**

- **Threat Models**
  1. Malicious Code – Software security
  2. Malicious Host – Software protection

- **Attack Scenario**
  - Man-At-The-End (MATE)

- **Case Study**
  - Industry partners: Jungheinrich GmbH, .NET C#, Technische Universität München

Context: **Software protection**

- **Means** [Collberg1998]

Context: **Software protection**

- **Means** [Collberg1998]
  1. Legal

Context: **Software protection**

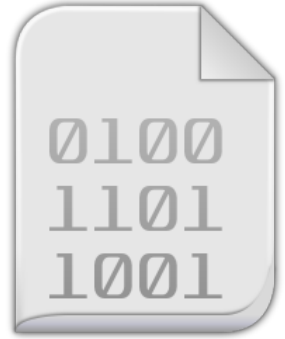
- **Means** [Collberg1998]
  1. Legal
  2. Remote Services

Context: **Software protection**

- **Means** [Collberg1998]
  1. Legal
  2. Remote Services
  3. Encryption

Context: **Software protection**

- **Means** [Collberg1998]
  1. Legal
  2. Remote Services
  3. Encryption
  4. Machine / Native code

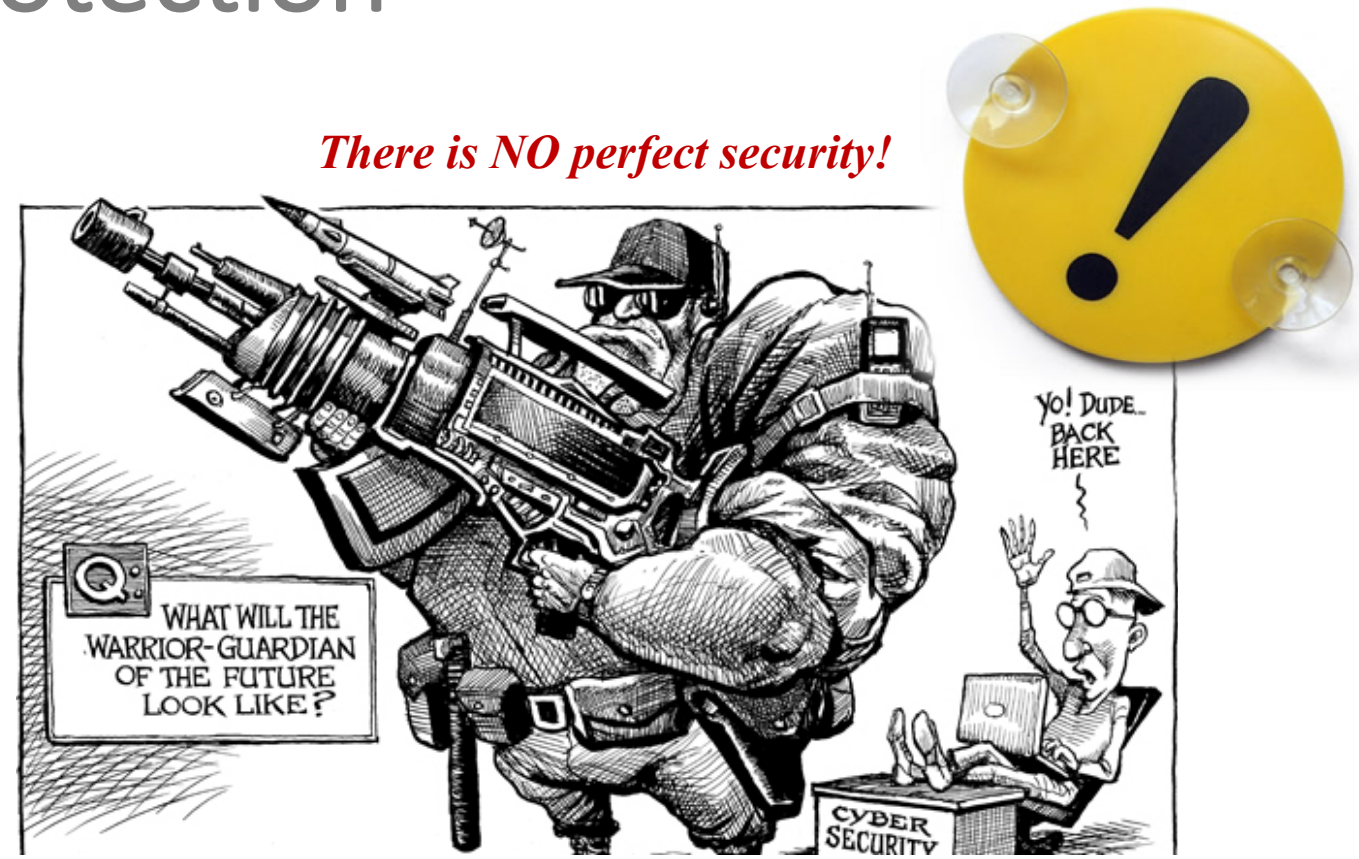


Context: **Software protection**

- **Means** [Collberg1998]
  1. Legal
  2. Remote Services
  3. Encryption
  4. Machine / Native code
  5. **Obfuscation**

Context: **Software protection**

- **Means** [Collberg1998]
  1. Legal
  2. Remote Services
  3. Encryption
  4. Machine / Native code
  5. **Obfuscation**



- **Obfuscation**

  First layer of *defense against intelligent tampering*

  Prevent understanding and reuse of intellectual property

Context: Protection via Obfuscation

- **Techniques** [classes of transformations]

Context: Protection via Obfuscation

- **Techniques** [classes of transformations]
  - **Lexical transformations**
    - Renaming of variables, methods
    - Assembly renaming and merging
    - Metadata Reduction

Before

Disassembler

```
private static void Main(string[] args)
{
    DateTime now = DateTime.Now.AddMonths(1);
    Unrenamed_Method_From_This_Assembly();
    MessageBox.Show("Program expires on " + now.ToLongDateString());
    Application.Exit();
}
```

After

Disassembler

```
private static void Main(string[] args)
{
    DateTime a = L.B(ref M.B(), 1);
    K.B();
    H.B(I.B(B.B(1), J.B(ref a)));
    G.B();
}
```

Context: Protection via Obfuscation

- **Techniques** [classes of transformations]
  - **Lexical transformations**
    - Renaming of variables, methods
    - Assembly renaming and merging
    - Metadata Reduction
  - **Data transformations**
    - String encoding
    - Array merging
    - Constants encoding
    - Value and Array Encryption

```
char *decode(char *s, size_t len) {
  for (int i = 0; i < len; i++)
    s[i] ^= 0x15;
  return s;
}

int main(int argc, char *argv[]) {
  struct hostent *addr =
       gethostbyname(decode("}aaef/::lz`a`;wp:qDb!b,BrMvD", 28));
  return 0;
}
```

Context: Protection via Obfuscation

- **Techniques** [classes of transformations]
  - **Lexical transformations**
    - Renaming of variables, methods
    - Assembly renaming and merging
    - Metadata Reduction
  - **Data transformations**
    - String encoding
    - Array merging
    - Constants encoding
    - Value and Array encoding
  - **Preventive transformations**
    - Code encryption

Code not Encrypted

```
public void LoadFile(string path)
{
   FileStream stream = new FileStream(path, FileMode.Open, FileAccess.Read);
   try
   {
      byte[] buffer = new byte[stream.Length];
      stream.Read(buffer, 0, (int) stream.Length);
      this.SetData(buffer);
   }
   finally
   {
      if (stream != null)
      {
         stream.Close();
      }
   }
}
```

Code Encrypted

```
public void LoadFile(string path)
{
   □.□("□", new object[] { this, path });
}
```

Context: Protection via Obfuscation

- **Techniques** [classes of transformations]
  - **Lexical transformations**
    - Renaming of variables, methods
    - Assembly renaming and merging
    - Metadata Reduction
  - **Data transformations**
    - String encoding
    - Array merging
    - Constants encoding
    - Value and Array Encryption
  - **Preventive transformations**
    - Code encryption
  - **Control flow transformations**
    - Control flow flattening
    - Hide External and Internal Calls

Control Flow not Obfuscated

```
private TreeNode<T> Pair(TreeNode<T> p, TreeNode<T> q)
{
    TreeNode<T> result;
    if (this._comparer.Compare(p.Value, q.Value) < 0)
    {
        p.Right = q.Left;
        q.Left = p;
        p.Parent = q;
        result = q;
    }
    else
    {
        q.Right = p.Left;
        p.Left = q;
        q.Parent = p;
        result = p;
    }
    return result;
}
```

Control Flow Obfuscated

```
private 0<0> 0(0<0> 0, 0<0> 0)
{
    0<0> result;
    if (((this.0.Compare(0.0, 0.0) < 0.0(1)) ? 1 : 0) != 0.0(1))
    {
        while (true)
        {
            IL_67:
            int num = 0.0(26);
            int num2 = -2;
            while (true)
            {
                num2 ^= 72;
                switch (num2 + 76)
                {
                case 0:
                    goto IL_67;
                case 1:
                    switch (num + 73)
                    {
                    case 0:
                        0.0 = 0;
                        num = -6;
                        goto IL_41;
                    case 1:
                        0.0 = 0.0;
                        num = 0.0(18);
                        goto IL_41;
```

Context: Protection via Obfuscation

- **Techniques** [classes of transformations]
  - **Lexical transformations**
    - Renaming of variables, methods
    - Assembly renaming and merging
    - Metadata Reduction
  - **Data transformations**
    - String encoding
    - Array merging
    - Constants encoding
    - Value and Array Encryption
  - **Preventive transformations**
    - Code encryption
  - **Control flow transformations**
    - Control flow flattening
    - Hide External and Internal Calls

- **Equivalence**
  - observable behavior

State of the Art **Obfuscation tools**

	C# obfuscators	Price	Code Virtualization
1.	Agile.NET	$795	Yes
2.	Crypto Obfuscator	$149–$4469	Yes
3.	Eazfuscator.NET	$399	Yes
4.	babelfor.NET	$150-$1500	Yes
5.	ConfuserEx	Free (opensource)	No
6.	Dotfuscator	~ $1600	No

State of the Art **Obfuscation tools**

	C# obfuscators	Price	Code Virtualization
1.	Agile.NET	$795	Yes
2.	Crypto Obfuscator	$149–$4469	Yes
3.	Eazfuscator.NET	$399	Yes
4.	babelfor.NET	$150-$1500	Yes
5.	ConfuserEx	Free (opensource)	No
6.	Dotfuscator	~ $1600	No

	C# decompilers	Price
1.	dotPeek	Free
2.	ILSpy	
3.	JustDecompile	
4.	ILDasm	

State of the Art **Obfuscation tools**

	C# obfuscators	Price	Code Virtualization
1.	Agile.NET	$795	Yes
2.	Crypto Obfuscator	$149–$4469	Yes
3.	Eazfuscator.NET	$399	Yes
4.	babelfor.NET	$150-$1500	Yes
5.	ConfuserEx	Free (opensource)	No
6.	Dotfuscator	~ $1600	No

	C# decompilers	Price
1.	dotPeek	Free
2.	ILSpy	
3.	JustDecompile	
4.	ILDasm	

	C# tracers	Price
1.	dotTrace	~$300
2.	Intel Pin (binary)	Free

Project: VOT4CS - Virtualization Obfuscation for C#

Goals

- design and implement virtualization obfuscator for C#
  - an open source alternative to commercial obfuscators
  - no free obfuscation tool with virtualization as a feature
  - available @ https://github.com/tum-i22/vot4cs

Project: VOT4CS - Virtualization Obfuscation for C#

Goals

- design and implement virtualization obfuscator for C#
  - an open source alternative to commercial obfuscators
  - no free obfuscation tool with virtualization as a feature
  - available @ https://github.com/tum-i22/vot4cs
- perform a case-study on a real-world software solution
  - performance evaluation
  - security evaluation

Background: Virtualization Obfuscation

- Input: Program P
- Generate a random new language
- Translate P to the new language as P`
- Synthetize an interpreter to translate the new instructions

Background: Virtualization Obfuscation

- Input: Program P
- Generate a random new language
- Translate P to the new language as P`
- Synthetize an interpreter to translate the new instructions

Background: Virtualization Obfuscation

- Input: Program P
- Generate a random new language
- Translate P to the new language as P`
- Synthetize an interpreter to translate the new instructions

Background: Virtualization Obfuscation

- Input: Program P
- Generate a random new language
- Translate P to the new language as P`
- Synthetize an interpreter to translate the new instructions

Usage

- Software Diversification
- Mitigate MATE attacks
- Mitigate Static and Dynamic analysis

Background: Virtualization Obfuscation

```
1   private int addition()
2   {
3       int a = 2;
4       int b = 3;
5       int sum = b + a;
6
7       Console.WriteLine(sum);
8       sum += sum + a - b;
9
10      return sum;
11  }
```

Background: Virtualization Obfuscation

```
1   private int addition()
2   {
3       int a = 2;
4       int b = 3;
5       int sum = b + a;
6
7       Console.WriteLine(sum);
8       sum += sum + a - b;
9
10      return sum;
11  }
```

...

...

- Operations
  - assignment
  - addition
  - subtraction
  - method invocation
  - return

Background: Virtualization Obfuscation

```
1   private int addition()
2   {
3       int a = 2;
4       int b = 3;
5       int sum = b + a;
6
7       Console.WriteLine(sum);
8       sum += sum + a - b;
9
10      return sum;
11  }
```

- Operations
  - assignment
  - addition
  - subtraction
  - method invocation
  - return

```
1   //code sequence
2   int[] code = { 150, 3, 0, 150, 4, 1, ..., 100, 0, 1, 6,..., 200, 300, 400, 5, ...}
3   object[] data = { a, b, sum, 2, 3, ...} //Program Variables
4   int vpc = 0; //Virtual Program Counter
5
6   while(true){
7       switch(code[vpc++]){
8           case 150: //assignment
9
10              ...
11
12          case 100: //addition
13
14              ...
15
16              break;
17          case 200: //substraction
18              ...
19          case 300: //comparison
20              ...
21          case 400: //method invocation
22              Console.WriteLine(data[code[vpc++]]);
23              break;
24          ...
25      }
26  }
```

Background: Virtualization Obfuscation

```
1   private int addition()
2   {
3       int a = 2;
4       int b = 3;
5       int sum = b + a;
6
7       Console.WriteLine(sum);
8       sum += sum + a - b;
9
10      return sum;
11  }
```

- Operations
  - assignment
  - addition
  - subtraction
  - method invocation
  - return

```
1   //code sequence
2   int[] code = { 150, 3, 0, 150, 4, 1, ..., 100, 0, 1, 6,..., 200, 300, 400, 5, ...}
3   object[] data = { a, b, sum, 2, 3, ...} //Program Variables
4   int vpc = 0; //Virtual Program Counter
5
6   while(true){
7       switch(code[vpc++]){
8           case 150: //assignment
9
10              ...
11
12          case 100: //addition
13
14              ...
15
16              break;
17          case 200: //substraction
18              ...
19          case 300: //comparison
20              ...
21          case 400: //method invocation
22              Console.WriteLine(data[code[vpc++]]);
23              break;
24          ...
25      }
26  }
```

Background: Virtualization Obfuscation

```
1   private int addition()
2   {
3       int a = 2;
4       int b = 3;
5       int sum = b + a;
6
7       Console.WriteLine(sum);
8       sum += sum + a - b;
9
10      return sum;
11  }
```

- Operations
  - assignment
  - addition
  - subtraction
  - method invocation
  - return

```
1   //code sequence
2   int[] code = { 150, 3, 0, 150, 4, 1, ..., 100, 0, 1, 6,..., 200, 300, 400, 5, ...}
3   object[] data = { a, b, sum, 2, 3, ...} //Program Variables
4   int vpc = 0; //Virtual Program Counter
5
6   while(true){
7       switch(code[vpc++]){
8           case 150: //assignment
9
10              ...
11
12          case 100: //addition
13
14              ...
15
16              break;
17          case 200: //substraction
18              ...
19          case 300: //comparison
20              ...
21          case 400: //method invocation
22              Console.WriteLine(data[code[vpc++]]);
23              break;
24          ...
25      }
26  }
```

Background: Virtualization Obfuscation

```
1   private int addition()
2   {
3       int a = 2;
4       int b = 3;
5       int sum = b + a;
6
7       Console.WriteLine(sum);
8       sum += sum + a - b;
9
10      return sum;
11  }
```

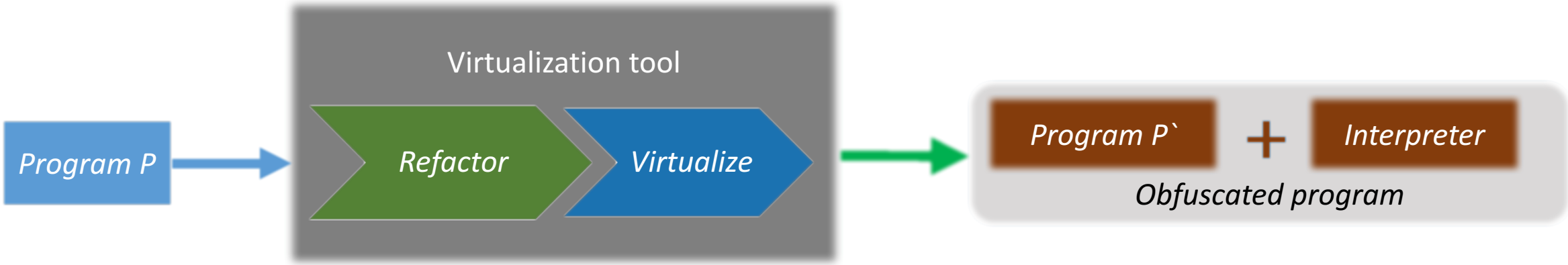
- Operations
  - assignment
  - addition
  - subtraction
  - method invocation
  - return

```
1   //code sequence
2   int[] code = { 150, 3, 0, 150, 4, 1, ..., 100, 0, 1, 6, ..., 200, 300, 400, 5, ...}
3   object[] data = { a, b, sum, 2, 3, ...} //Program variables
4   int vpc = 0; //Virtual Program Counter
5
6   while(true){
7       switch(code[vpc++]){
8           case 150: //assignment
9
10              ...
11
12          case 100: //addition
13              var op1 = data[code[vpc++]]; // op1 = x;
14              var op2 = data[code[vpc++]]; // op2 = y;
15              data[code[vpc++]] = op1 + op2; // data[6] = x + y;
16              break;
17          case 200: //substraction
18              ...
19          case 300: //comparison
20              ...
21          case 400: //method invocation
22              Console.WriteLine(data[code[vpc++]]);
23              break;
24          ...
25      }
26  }
```

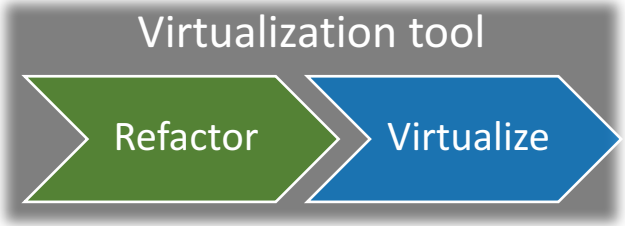
Design and Implementation

Virtualization Obfuscation Algorithm

- Refactoring Transformations
- Virtualization Transformations

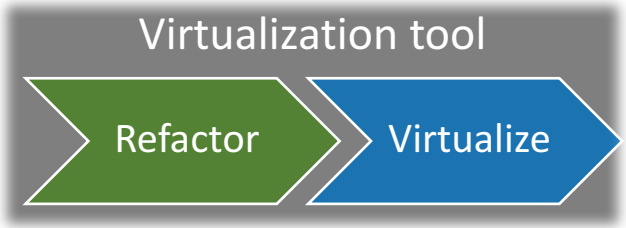


Design and Implementation:
Algorithm – Refactoring transformations



- Arithmetic simplification
  - +, -, *, /, ++, --, %

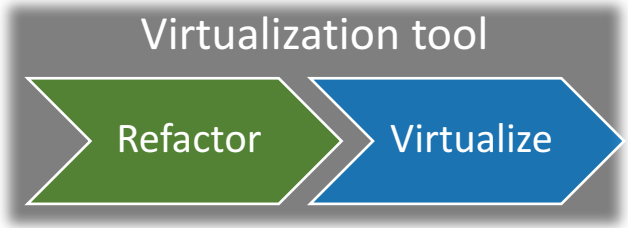
Design and Implementation:
Algorithm – Refactoring transformations



- Arithmetic simplification
  - +, -, *, /, ++, --, %

```
1  sum += sum + a - b;
2
3  int addTemp_0 = sum + a;
4  int addTemp_1 = addTemp_0 - b;
5  sum = sum + addTemp_1;
```

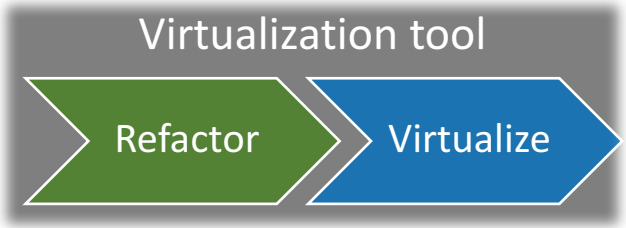
Design and Implementation:
Algorithm – Refactoring transformations



- Arithmetic simplification
  - +, -, *, /, ++, --, %

```
1  sum += sum + a - b;
2
3  int addTemp_0 = sum + a;
4  int addTemp_1 = addTemp_0 - b;
5  sum = sum + addTemp_1;
```

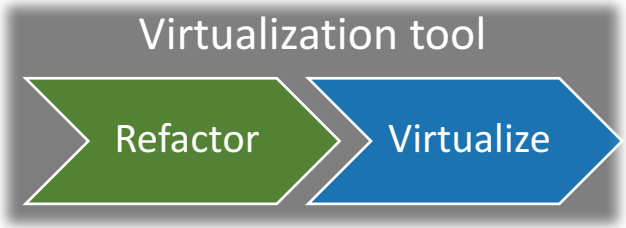
Design and Implementation:
Algorithm – Refactoring transformations



- Arithmetic simplification
  - +, -, *, /, ++, --, %
- Invocation simplification
  - obj.m1().m2().m3()
  - obj.member1.member13

```
1   sum += sum + a - b;
2
3   int addTemp_0 = sum + a;
4   int addTemp_1 = addTemp_0 - b;
5   sum = sum + addTemp_1;
```

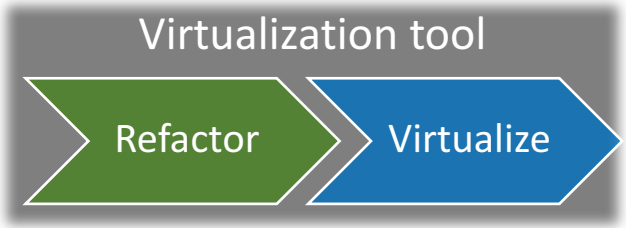
Design and Implementation:
Algorithm – Refactoring transformations




```
1   sum += sum + a - b;
2
3   int addTemp_0 = sum + a;
4   int addTemp_1 = addTemp_0 - b;
5   sum = sum + addTemp_1;
```

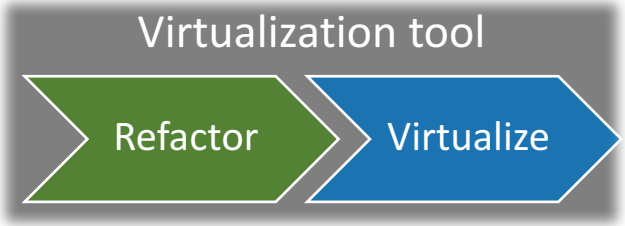
- Arithmetic simplification
  - +, -, *, /, ++, --, %
- Invocation simplification
  - obj.m1().m2().m3()
  - obj.member1.member13
- Comparison simplification
  - >, <, >=, <=
- For, ForEach, DoWhile conversion to While
- Switch conversion to chained If
- Local variables initialization
- Composed assignment
- Conditional expression
- Try/Catch statement
- …

Design and Implementation:
Algorithm – Virtualization transformations



Step 1. ***Select method body to virtualize.***

Design and Implementation:
Algorithm – Virtualization transformations



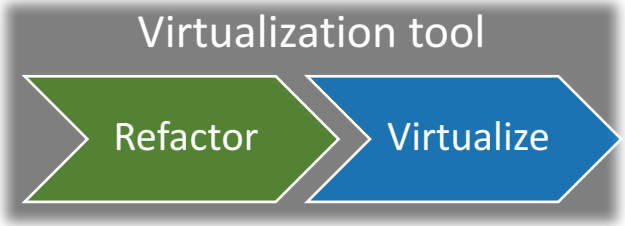
Step 1. ***Select method body to virtualize.***

Step 2. ***Virtualize constants, local arguments, parameters.***

- *generate DATA array*

```
object[] data = { a, b, sum, 2, 3, ...} //Program Variables
int vpc = 0; //Virtual Program Counter
```

Design and Implementation:
Algorithm – Virtualization transformations



Step 1. ***Select method body to virtualize.***

Step 2. ***Virtualize constants, local arguments, parameters.***

- generate DATA array

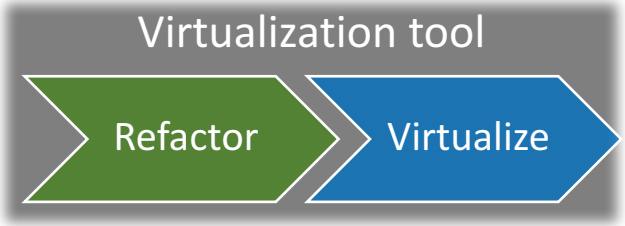
```
object[] data = { a, b, sum, 2, 3, ...} //Program Variables
int vpc = 0; //Virtual Program Counter
```

Step 3. ***Process method's body statements.***

- generate CODE array

```
//code sequence
int[] code = { 150, 3, 0, 150, 4, 1, ..., 100, 0, 1, 6,..., 200, 300, 400, 5, ...}
```

Design and Implementation:
Algorithm – Virtualization transformations



Step 1. ***Select method body to virtualize.***

Step 2. ***Virtualize constants, local arguments, parameters.***

- generate DATA array

```
object[] data = { a, b, sum, 2, 3, ...} //Program Variables
int vpc = 0; //Virtual Program Counter
```

Step 3. ***Process method's body statements.***

- generate CODE array

```
//code sequence
int[] code = { 150, 3, 0, 150, 4, 1, ..., 100, 0, 1, 6,..., 200, 300, 400, 5, ...}
```

Step 4. ***For compound structures go to previous step and virtualize the statement's body.***

Design and Implementation:
Algorithm – Virtualization transformations

int[] code

- **Virtual Operation**
  - Operation Key
  - List<Operand>
  - Prefix Size
  - Postfix Size
  - Offset
  - Frequency

Design and Implementation:
Algorithm – Virtualization transformations

int[] code

- **Virtual Operation**
  - Operation Key
  - List<Operand>
  - Prefix Size
  - Postfix Size
  - Offset
  - Frequency

```
1   code[1151]=3448; //fake-ExpressionStatement_2_3448_-19
2   code[1156]=3425; //fake-ExpressionStatement_2_3425_-14
3   code[1163]=342; //fake-ExpressionStatement_2_342_-7
4   code[1169]=1958; //fake-ExpressionStatement_2_1958_-1
5   code[1170]=9919; //ExpressionStatement_2 # ExpressionStatement_17
6   code[1172]=3083; //sum
7   code[1174]=3083; //sum
8   code[1176]=3418; //fake-ExpressionStatement_2_3418_6
9   code[1190]=2482; //invocationTemp_7
10  code[1199]=3471; //fake-ExpressionStatement_2_3471_29
```

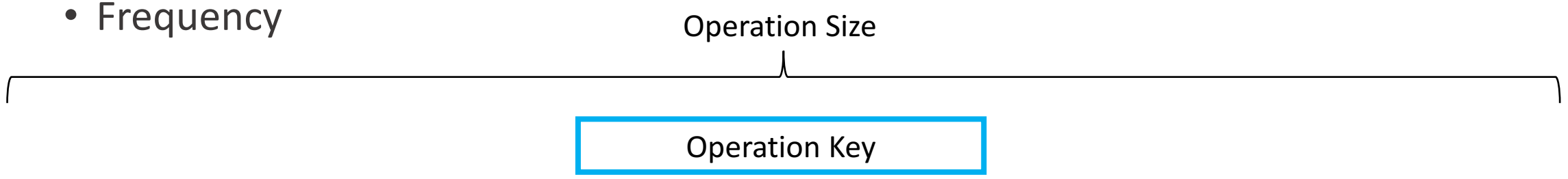
Operation Size

Design and Implementation:
Algorithm – Virtualization transformations

int[] code

- **Virtual Operation**
  - Operation Key
  - List<Operand>
  - Prefix Size
  - Postfix Size
  - Offset
  - Frequency

```
1   code[1151]=3448; //fake-ExpressionStatement_2_3448_-19
2   code[1156]=3425; //fake-ExpressionStatement_2_3425_-14
3   code[1163]=342; //fake-ExpressionStatement_2_342_-7
4   code[1169]=1958; //fake-ExpressionStatement_2_1958_-1
5   code[1170]=9919; //ExpressionStatement_2 # ExpressionStatement_17
6   code[1172]=3083; //sum
7   code[1174]=3083; //sum
8   code[1176]=3418; //fake-ExpressionStatement_2_3418_6
9   code[1190]=2482; //invocationTemp_7
10  code[1199]=3471; //fake-ExpressionStatement_2_3471_29
```

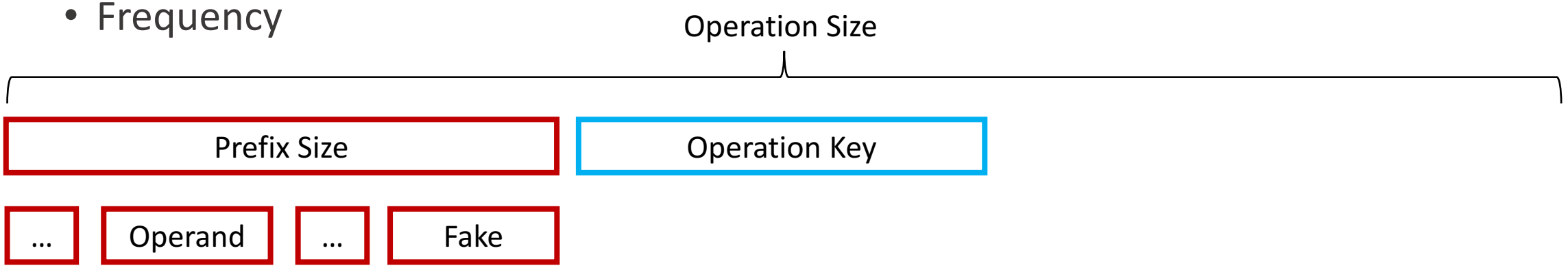


Design and Implementation:
Algorithm – Virtualization transformations

int[] code

- **Virtual Operation**
  - Operation Key
  - List<Operand>
  - Prefix Size
  - Postfix Size
  - Offset
  - Frequency

```
1   code[1151]=3448; //fake-ExpressionStatement_2_3448_-19
2   code[1156]=3425; //fake-ExpressionStatement_2_3425_-14
3   code[1163]=342; //fake-ExpressionStatement_2_342_-7
4   code[1169]=1958; //fake-ExpressionStatement_2_1958_-1
5   code[1170]=9919; //ExpressionStatement_2 # ExpressionStatement_17
6   code[1172]=3083; //sum
7   code[1174]=3083; //sum
8   code[1176]=3418; //fake-ExpressionStatement_2_3418_6
9   code[1190]=2482; //invocationTemp_7
10  code[1199]=3471; //fake-ExpressionStatement_2_3471_29
```

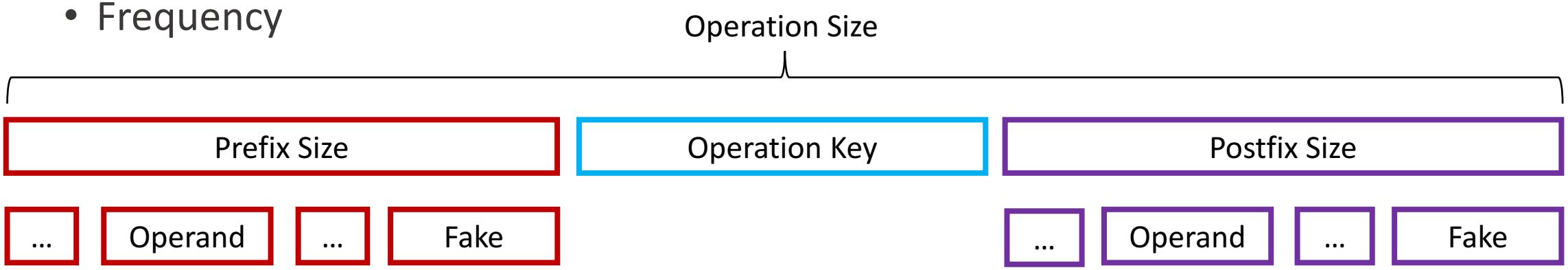


Design and Implementation:
Algorithm – Virtualization transformations

int[] code

- **Virtual Operation**
  - Operation Key
  - List<Operand>
  - Prefix Size
  - Postfix Size
  - Offset
  - Frequency

```
1    code[1151]=3448; //fake-ExpressionStatement_2_3448_-19
2    code[1156]=3425; //fake-ExpressionStatement_2_3425_-14
3    code[1163]=342; //fake-ExpressionStatement_2_342_-7
4    code[1169]=1958; //fake-ExpressionStatement_2_1958_-1
5    code[1170]=9919; //ExpressionStatement_2 # ExpressionStatement_17
6    code[1172]=3083; //sum
7    code[1174]=3083; //sum
8    code[1176]=3418; //fake-ExpressionStatement_2_3418_6
9    code[1190]=2482; //invocationTemp_7
10   code[1199]=3471; //fake-ExpressionStatement_2_3471_29
```



Design and Implementation:
Obfuscation settings

- **Interpreter**
  - Inside Method
  - Class Instance
  - Class Static

```
1   [Obfuscation(Exclude = false, Feature = "virtualization; class; readable")]
2   private string ForDoubleBreak_0(int a, int b)
3   {
4       //Virtualization variables
5       int[] code = new int[3000];
6       object[] data = new object[1500];
7       int vpc = 0;
8
9       //Data init
10      data[0]=a; //a
11      //...
12
13      //Code init
14      code[0]=3822; //
15      //...
16
17      return (string)InstanceInterpreterVirtualization(vpc, data, code);
18  }
```

Design and Implementation:
Obfuscation settings

- **Interpreter**
  - Inside Method
  - Class Instance
  - Class Static

```
1   [Obfuscation(Exclude = false, Feature = "virtualization; class; readable")]
2   private string ForDoubleBreak_0(int a, int b)
3   {
4       //Virtualization variables
5       int[] code = new int[3000];
6       object[] data = new object[1500];
7       int vpc = 0;
8
9       //Data init
10      data[0]=a; //a
11      //...
12
13      //Code init
14      code[0]=3822; //
15      //...
16
17      return (string)InstanceInterpreterVirtualization(vpc, data, code);
18  }
```

Design and Implementation:
Obfuscation settings

- **Interpreter**
  - Inside Method
  - Class Instance
  - Class Static

```
1   [Obfuscation(Exclude = false, Feature = "virtualization; class; readable")]
2   private string ForDoubleBreak_0(int a, int b)
3   {
4       //Virtualization variables
5       int[] code = new int[3000];
6       object[] data = new object[1500];
7       int vpc = 0;
8
9       //Data init
10      data[0]=a; //a
11      //...
12
13      //Code init
14      code[0]=3822; //
15      //...
16
17      return (string)InstanceInterpreterVirtualization(vpc, data, code);
18  }
```

Design and Implementation:
Obfuscation settings

- **Interpreter**
  - Inside Method
  - Class Instance
  - Class Static

- **Refactoring**
  - Max operands
  - Max invocations

Design and Implementation:
Obfuscation settings

- **Interpreter**
  - Inside Method
  - Class Instance
  - Class Static
- **Operation**
  - Prefix size
  - Postfix size
  - Junk code
- **Refactoring**
  - Max operands
  - Max invocations

Design and Implementation:
Obfuscation settings

- **Interpreter**
  - Inside Method
  - Class Instance
  - Class Static
- **Refactoring**
  - Max operands
  - Max invocations
- **Operation**
  - Prefix size
  - Postfix size
  - Junk code
- **Switch**
  - Default section
  - Operation size

Design and Implementation:
Obfuscation settings

- **Randomization points**

```
1   switch(code[vpc])
2   {
3       case 4774: //frequency 1 WhileStatementSyntax_23
4           data[code[vpc+(-14)]]=(bool)data[code[vpc+(-13)]]?(int)data[code[vpc+(-6)]]:(int)data[code[vpc+(-9)]];
5           vpc+=(int)data[code[vpc+(-14)]];
6           break;
7       case 7385: //frequency 4 ExpressionStatement_0
8           data[code[vpc+(23)]]= (string)data[code[vpc+(-18)]]+ (int)data[code[vpc+(11)]];
9           vpc+=61;
10          break;
11      case 9872: //frequency 2 ExpressionStatement_21
12          data[code[vpc+(-4)]]= ReturnArg_Array((int)data[code[vpc+(-16)]]);
13          vpc+=52;
14          break;
15      case 7635: //frequency 2 ExpressionStatement_18
16          data[code[vpc+(9)]]= data[code[vpc+(25)]];
17          vpc+=63;
18          break;
19      default: //frequency 10 ExpressionStatement_2
20          data[code[vpc+(2)]]= (string)data[code[vpc+(4)]]+ (string)data[code[vpc+(20)]];
21          vpc+=69;
22          break;
23  }
```

Design and Implementation:
Obfuscation settings

- **Randomization points**
  - Operation key

```
1   switch(code[vpc])
2   {
3       case 4774: //frequency 1 WhileStatementSyntax_23
4           data[code[vpc+(-14)]]=(bool)data[code[vpc+(-13)]]?(int)data[code[vpc+(-6)]]:(int)data[code[vpc+(-9)]];
5           vpc+=(int)data[code[vpc+(-14)]];
6           break;
7       case 7385: //frequency 4 ExpressionStatement_0
8           data[code[vpc+(23)]]= (string)data[code[vpc+(-18)]]+ (int)data[code[vpc+(11)]];
9           vpc+=61;
10          break;
11      case 9872: //frequency 2 ExpressionStatement_21
12          data[code[vpc+(-4)]]= ReturnArg_Array((int)data[code[vpc+(-16)]]);
13          vpc+=52;
14          break;
15      case 7635: //frequency 2 ExpressionStatement_18
16          data[code[vpc+(9)]]= data[code[vpc+(25)]];
17          vpc+=63;
18          break;
19      default: //frequency 10 ExpressionStatement_2
20          data[code[vpc+(2)]]= (string)data[code[vpc+(4)]]+ (string)data[code[vpc+(20)]];
21          vpc+=69;
22          break;
23  }
```

Design and Implementation:
Obfuscation settings

- **Randomization points**
  - Operation key
  - Operands position

```
1  switch(code[vpc])
2  {
3      case 4774: //frequency 1 WhileStatementSyntax 23
4          data[code[vpc+(-14)]]=(bool)data[code[vpc+(-13)]]?(int)data[code[vpc+(-6)]]:(int)data[code[vpc+(-9)]];
5          vpc+=(int)data[code[vpc+(-14)]];
6          break;
7      case 7385: //frequency 4 ExpressionStatement_0
8          data[code[vpc+(23)]]= (string)data[code[vpc+(-18)]]+ (int)data[code[vpc+(11)]];
9          vpc+=61;
10         break;
11     case 9872: //frequency 2 ExpressionStatement_21
12         data[code[vpc+(-4)]]= ReturnArg_Array((int)data[code[vpc+(-16)]]);
13         vpc+=52;
14         break;
15     case 7635: //frequency 2 ExpressionStatement_18
16         data[code[vpc+(9)]]= data[code[vpc+(25)]];
17         vpc+=63;
18         break;
19     default: //frequency 10 ExpressionStatement_2
20         data[code[vpc+(2)]]= (string)data[code[vpc+(4)]]+ (string)data[code[vpc+(20)]];
21         vpc+=69;
22         break;
23 }
```

Design and Implementation:
Obfuscation settings

- **Randomization points**
  - Operation key
  - Operands position
  - Switch sections

```
1   switch(code[vpc])
2   {
3       case 4774: //frequency 1 WhileStatementSyntax_23
4           data[code[vpc+(-14)]]=(bool)data[code[vpc+(-13)]]?(int)data[code[vpc+(-6)]]:(int)data[code[vpc+(-9)]];
5           vpc+=(int)data[code[vpc+(-14)]];
6           break;
7       case 7385: //frequency 4 ExpressionStatement_0
8           data[code[vpc+(23)]]= (string)data[code[vpc+(-18)]]+ (int)data[code[vpc+(11)]];
9           vpc+=61;
10          break;
11      case 9872: //frequency 2 ExpressionStatement_21
12          data[code[vpc+(-4)]]= ReturnArg_Array((int)data[code[vpc+(-16)]]);
13          vpc+=52;
14          break;
15      case 7635: //frequency 2 ExpressionStatement_18
16          data[code[vpc+(9)]]= data[code[vpc+(25)]];
17          vpc+=63;
18          break;
19      default: //frequency 10 ExpressionStatement_2
20          data[code[vpc+(2)]]= (string)data[code[vpc+(4)]]+ (string)data[code[vpc+(20)]];
21          vpc+=69;
22          break;
23  }
```

Design and Implementation:
Obfuscation settings

- **Randomization points**
  - Operation key
  - Operands position
  - Switch sections
  - Array initialization
  - VPC initialization

```
code[438]=3590;    data[2038]=-979;
    data[2495]=67;
    data[3724]=max;
code[827]=3148;code[1068]=3590;code[479]=3382;code[859]=3590;code[915]=
code[344]=3147;code[390]=2070;code[1192]=9352;code[1145]=3382;
code[404]=121;code[750]=2495;    data[2686]=-886;
code[355]=2070;code[638]=7133;code[757]=347;    data[1529]=899;
code[687]=3590;code[854]=1208;code[1079]=2038;code[525]=2686;
code[766]=7383;code[117]=8297;    data[2542]=false;
code[962]=9352;    data[1271]=key;
code[1055]=3148;    data[121]=2; data[1855]=67;
code[386]=3590;code[793]=914;code[586]=9352;code[519]=7383;code[345]=75
    data[2174]=false;
    data[917]=551; data[569]=-1;
code[1112]=2897;    data[3369]=533;
    data[347]=303; data[914]=false;
```

Design and Implementation:
Obfuscation settings

- **Randomization points**
  - Operation key
  - Operands position
  - Switch sections
  - Array initialization
  - VPC initialization
- **Switch**:
  - Default section

```
1  switch(code[vpc])
2  {
3      case 4774: //frequency 1 WhileStatementSyntax_23
4          data[code[vpc+(-14)]]=(bool)data[code[vpc+(-13)]]?(int)data[code[vpc+(-6)]]:(int)data[code[vpc+(-9)]];
5          vpc+=(int)data[code[vpc+(-14)]];
6          break;
7      case 7385: //frequency 4 ExpressionStatement_0
8          data[code[vpc+(23)]]= (string)data[code[vpc+(-18)]]+ (int)data[code[vpc+(11)]];
9          vpc+=61;
10         break;
11     case 9872: //frequency 2 ExpressionStatement_21
12         data[code[vpc+(-4)]]= ReturnArg_Array((int)data[code[vpc+(-16)]]);
13         vpc+=52;
14         break;
15     case 7635: //frequency 2 ExpressionStatement_18
16         data[code[vpc+(9)]]= data[code[vpc+(25)]];
17         vpc+=63;
18         break;
19     default: //frequency 10 ExpressionStatement_2
20         data[code[vpc+(2)]]= (string)data[code[vpc+(4)]]+ (string)data[code[vpc+(20)]];
21         vpc+=69;
22         break;
23 }
```

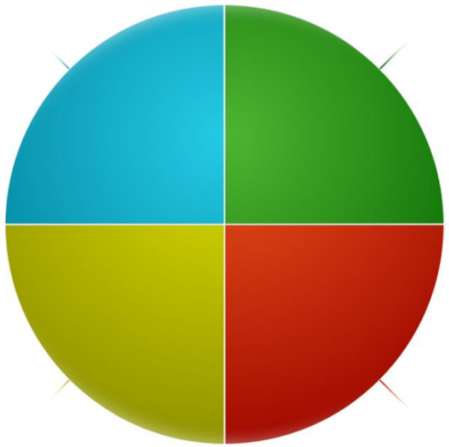
Design and Implementation:
Obfuscation settings

- **Randomization points**
  - Operation key
  - Operands position
  - Switch sections
  - Array initialization
  - VPC initialization
- **Switch**:
  - Default section
  - Operation size

```
1  switch(code[vpc])
2  {
3      case 4774: //frequency 1 WhileStatementSyntax_23
4          data[code[vpc+(-14)]]=(bool)data[code[vpc+(-13)]]?(int)data[code[vpc+(-6)]]:(int)data[code[vpc+(-9)]];
5          vpc+=(int)data[code[vpc+(-14)]];
6          break;
7      case 7385: //frequency 4 ExpressionStatement_0
8          data[code[vpc+(23)]]= (string)data[code[vpc+(-18)]]+ (int)data[code[vpc+(11)]];
9          vpc+=61;
10         break;
11     case 9872: //frequency 2 ExpressionStatement_21
12         data[code[vpc+(-4)]]= ReturnArg_Array((int)data[code[vpc+(-16)]]);
13         vpc+=52;
14         break;
15     case 7635: //frequency 2 ExpressionStatement_18
16         data[code[vpc+(9)]]= data[code[vpc+(25)]];
17         vpc+=63;
18         break;
19     default: //frequency 10 ExpressionStatement_2
20         data[code[vpc+(2)]]= (string)data[code[vpc+(4)]]+ (string)data[code[vpc+(20)]];
21         vpc+=69;
22         break;
23 }
```

Evaluation

- **Framework** [Collberg1998]
  - **Potency**: To what degree is a **human** reader confused?
  - **Resilience**: How well are automatic deobfuscation **attacks** resisted?
  - **Cost**: How much time/space **overhead** is added?
  - **Stealth**: How well does obfuscated code **blend** in the original code?



Evaluation - Potency

To what degree is a human reader confused?

Evaluation - Potency

To what degree is a human reader confused?

- **Layout transformations**
  - Remove comments
  - Variable lose names

Evaluation - Potency

To what degree is a human reader confused?

- **Layout transformations**
  - Remove comments
  - Variable lose names
- **Control flow obfuscation**
  - Logic hidden in CODE array

Evaluation - Potency

To what degree is a human reader confused?

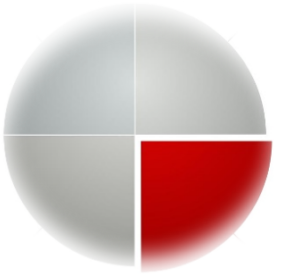
- **Layout transformations**
  - Remove comments
  - Variable lose names
- **Control flow obfuscation**
  - Logic hidden in CODE array
- **Method “overloading”**
  - Class / Instance interpreter
  - Same interpreter for multiple methods

Evaluation - Potency

To what degree is a human reader confused?

- **Layout transformations**
  - Remove comments
  - Variable lose names
- **Control flow obfuscation**
  - Logic hidden in CODE array
- **Method “overloading”**
  - Class / Instance interpreter
  - Same interpreter for multiple methods
- **Data transformations**
  - Variables and constants are stored in DATA array

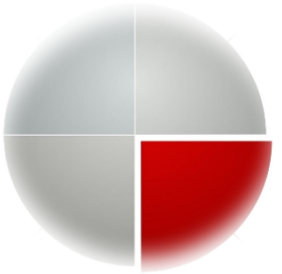
Evaluation - Resilience



To what degree is an automatic tool confused?

Obfuscation Goal	protect intellectual property
Attacker Goal	extract source code or control flow graph
Attacker Model	man-at-the-end (MATE)

Evaluation - Resilience



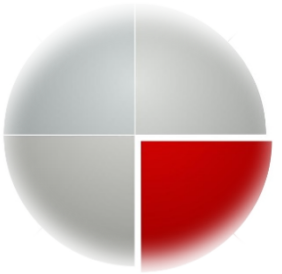
To what degree is an automatic tool confused?

Obfuscation Goal	protect intellectual property
Attacker Goal	extract source code or control flow graph
Attacker Model	man-at-the-end (MATE)

Virtualization Obfuscation analysis

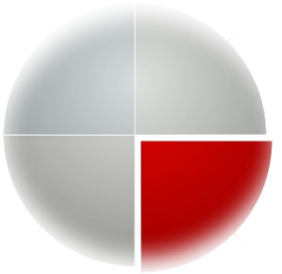
	Paper	Level	Automation	Type	Result	Limitation	Risk	Applicable
1	[Rolles2009]	binary	no / partial	**static manual analysis**	**extract original code**	time consuming not scalable	high	**yes**
2	[Sharif2009]	binary	yes	dynamic analysis	control flow graph	strong assumptions	high	no
3	[Coogan2011]	binary	yes	**dynamic automatic analysis**	approximation of original code **significant trace**	difficult to convert equations to CFG	high	**yes**
4	[Kinder2012]	binary	yes	static analysis	approximated data values	code of the emulator analyzed	low	no
5	[Yadegari2015]	binary	yes	**dynamic analysis** trace analysis	**control flow graph**	large input space trace simplification	high	**yes**

Evaluation - Resilience



- **Manual static analysis**: [Rolles2009]
  - Revert Interpreter

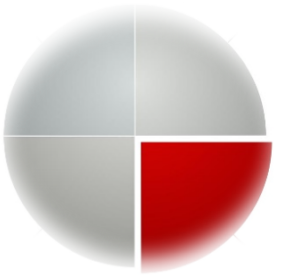
Evaluation - Resilience



- **Manual static analysis**: [Rolles2009]
  - Revert Interpreter

- **Automatic dynamic analysis**

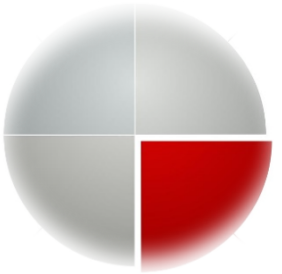
Evaluation - Resilience



- **Manual static analysis**: [Rolles2009]
  - Revert Interpreter

- **Automatic dynamic analysis**
  - Step 1: Mark method to trace
    - ConfuserEx, CIL instructions
    - Code injection

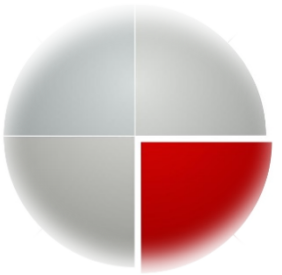
Evaluation - Resilience



- **Manual static analysis**: [Rolles2009]
  - Revert Interpreter

- **Automatic dynamic analysis**
  - Step 1: Mark method to trace
    - ConfuserEx, CIL instructions
    - Code injection
  - Step 2: Trace instructions
    - Data Dump
    - Runtime trace

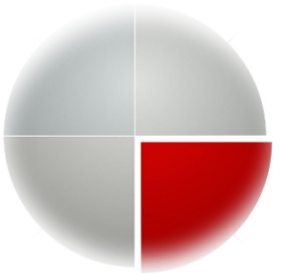
Evaluation - Resilience



- **Manual static analysis**: [Rolles2009]
  - Revert Interpreter

- **Automatic dynamic analysis**
  - Step 1: Mark method to trace
    - ConfuserEx, CIL instructions
    - Code injection
  - Step 2: Trace instructions
    - Data Dump
    - Runtime trace
  - Step 3: Simplify traces
    - Python scripts

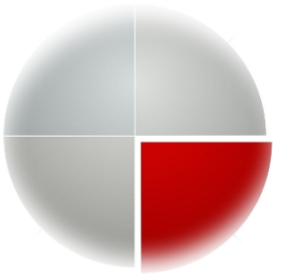
Evaluation - Resilience



- **Manual static analysis**: [Rolles2009]
  - Revert Interpreter

- **Automatic dynamic analysis**
  - Step 1: Mark method to trace
    - ConfuserEx, CIL instructions
    - Code injection
  - Step 2: Trace instructions
    - Data Dump
    - Runtime trace
  - Step 3: Simplify traces
    - Python scripts
  - Step 4: Compare traces

Evaluation - Resilience

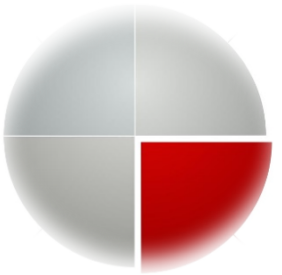


- Step 3: Simplify traces

Simplified

Obfuscated

Evaluation - Resilience



- Step 3: Simplify traces

```
string sum = "" + 3 + 4 + "";
```

Simplified

Obfuscated

Evaluation - Resilience

- Step 3: Simplify traces

```
#p#1#vpc#44
#p#2#data#System.Object[]
#1#0##7697
96#IL_01BD# ldarg data
97#IL_01C1# ldarg code
98#IL_01C5# ldarg vpc
99#IL_01C9# ldc.i4 -17
100#IL_01CE# add
101#IL_01CF# ldelem.i4
102#IL_01D0# ldarg data
103#IL_01D4# ldarg code
104#IL_01D8# ldarg vpc
105#IL_01DC# ldc.i4 14
106#IL_01E1# add
107#IL_01E2# ldelem.i4
108#IL_01E3# ldelem.ref
109#IL_01E4# castclass System.String
110#IL_01E9# ldarg data
111#IL_01ED# ldarg code
112#IL_01F1# ldarg vpc
113#IL_01F5# ldc.i4 3
114#IL_01FA# add
115#IL_01FB# ldelem.i4
116#IL_01FC# ldelem.ref
117#IL_01FD# unbox.any System.Int32
118#IL_0202# box System.Int32
119#IL_0207# call System.String
System.String::Concat(System.Object,System.Object)
120#IL_020C# stelem.ref
```

Obfuscated

```
string sum = "" + 3 + 4 + "";
```

```
1  ldloc var_3410
2  ldloc var_3245
3  call System.String System.String::Concat
4  stloc var_2165
```

Simplified

Evaluation - Resilience

- Step 3: Simplify traces

```
#p#1#vpc#44
#p#2#data#System.Object[]
#1#0##7697
96#IL_01BD# ldarg data
97#IL_01C1# ldarg code
98#IL_01C5# ldarg vpc
99#IL_01C9# ldc.i4 -17
100#IL_01CE# add
101#IL_01CF# ldelem.i4
102#IL_01D0# ldarg data
103#IL_01D4# ldarg code
104#IL_01D8# ldarg vpc
105#IL_01DC# ldc.i4 14
106#IL_01E1# add
107#IL_01E2# ldelem.i4
108#IL_01E3# ldelem.ref
109#IL_01E4# castclass System.String
110#IL_01E9# ldarg data
111#IL_01ED# ldarg code
112#IL_01F1# ldarg vpc
113#IL_01F5# ldc.i4 3
114#IL_01FA# add
115#IL_01FB# ldelem.i4
116#IL_01FC# ldelem.ref
117#IL_01FD# unbox.any System.Int32
118#IL_0202# box System.Int32
119#IL_0207# call System.String
System.String::Concat(System.Object,System.Object)
120#IL_020C# stelem.ref
```

Obfuscated

```
string sum = "" + 3 + 4 + "";
```

```
1  ldloc var_3410
2  ldloc var_3245
3  call System.String System.String::Concat
4  stloc var_2165
```

Simplified

Evaluation - Resilience

- Step 3: Simplify traces

```
#p#1#vpc#44
#p#2#data#System.Object[]
#1#0##7697
96#IL_01BD# ldarg data
97#IL_01C1# ldarg code
98#IL_01C5# ldarg vpc
99#IL_01C9# ldc.i4 -17
100#IL_01CE# add
101#IL_01CF# ldelem.i4
102#IL_01D0# ldarg data
103#IL_01D4# ldarg code
104#IL_01D8# ldarg vpc
105#IL_01DC# ldc.i4 14
106#IL_01E1# add
107#IL_01E2# ldelem.i4
108#IL_01E3# ldelem.ref
109#IL_01E4# castclass System.String
110#IL_01E9# ldarg data
111#IL_01ED# ldarg code
112#IL_01F1# ldarg vpc
113#IL_01F5# ldc.i4 3
114#IL_01FA# add
115#IL_01FB# ldelem.i4
116#IL_01FC# ldelem.ref
117#IL_01FD# unbox.any System.Int32
118#IL_0202# box System.Int32
119#IL_0207# call System.String
System.String::Concat(System.Object,System.Object)
120#IL_020C# stelem.ref
```

Obfuscated

```
string sum = "" + 3 + 4 + "";
```

```
1  ldloc var_3410
2  ldloc var_3245
3  call System.String System.String::Concat
4  stloc var_2165
```

Simplified

Evaluation - Resilience

- Step 3: Simplify traces

```
#p#1#vpc#44
#p#2#data#System.Object[]
#1#0##7697
96#IL_01BD# ldarg data
97#IL_01C1# ldarg code
98#IL_01C5# ldarg vpc
99#IL_01C9# ldc.i4 -17
100#IL_01CE# add
101#IL_01CF# ldelem.i4
102#IL_01D0# ldarg data
103#IL_01D4# ldarg code
104#IL_01D8# ldarg vpc
105#IL_01DC# ldc.i4 14
106#IL_01E1# add
107#IL_01E2# ldelem.i4
108#IL_01E3# ldelem.ref
109#IL_01E4# castclass System.String
110#IL_01E9# ldarg data
111#IL_01ED# ldarg code
112#IL_01F1# ldarg vpc
113#IL_01F5# ldc.i4 3
114#IL_01FA# add
115#IL_01FB# ldelem.i4
116#IL_01FC# ldelem.ref
117#IL_01FD# unbox.any System.Int32
118#IL_0202# box System.Int32
119#IL_0207# call System.String
System.String::Concat(System.Object,System.Object)
120#IL_020C# stelem.ref
```

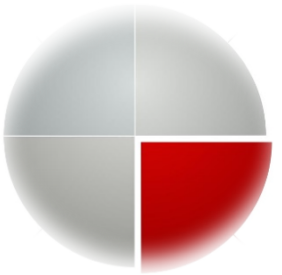
Obfuscated

```
string sum = "" + 3 + 4 + "";
```

```
1  ldloc var_3410
2  ldloc var_3245
3  call System.String System.String::Concat
4  stloc var_2165
```

Simplified

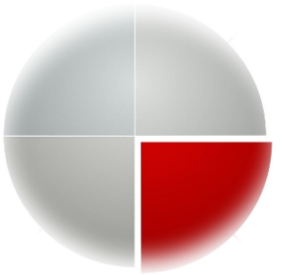
Evaluation - Resilience



- Step 3: Simplify traces

Loop iterations		Original	Refactored					Obfuscated				
	Refactoring settings		op2 in1	op3 in1	op4 in1	op4 in4	op4 in5	op2 in1	op3 in1	op4 in1	op4 in4	op4 in5
Execution trace	5	289	284	259	325	313	313	2498	2030	2092	1824	1687
	25	1149	1104	999	1285	1233	1233	9538	7450	7972	7224	6587
	125	5449	5204	4699	6085	5833	5833	44738	34550	37372	34224	31087
Simplified trace	5	280	275	250	316	304	304	295	234	331	307	289
	25	1120	1075	970	1256	1204	1204	1095	814	1231	1167	1069
	125	5320	5075	4570	5956	5704	5704	5095	3714	5731	5467	4969

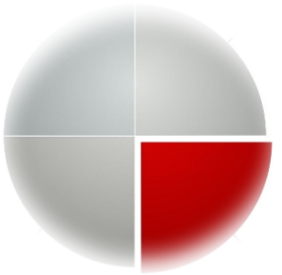
Evaluation - Resilience



- Step 3: Simplify traces

Loop iterations		Original	Refactored					Obfuscated				
	Refactoring settings		op2 in1	op3 in1	op4 in1	op4 in4	op4 in5	op2 in1	op3 in1	op4 in1	op4 in4	op4 in5
Execution trace	5	289	284	259	325	313	313	2498	2030	2092	1824	1687
	25	1149	1104	999	1285	1233	1233	9538	7450	7972	7224	6587
	125	5449	5204	4699	6085	5833	5833	44738	34550	37372	34224	31087
Simplified trace	5	280	275	250	316	304	304	295	234	331	307	289
	25	1120	1075	970	1256	1204	1204	1095	814	1231	1167	1069
	125	5320	5075	4570	5956	5704	5704	5095	3714	5731	5467	4969

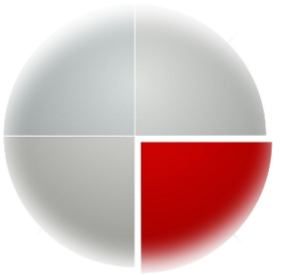
Evaluation - Resilience



- Step 3: Simplify traces

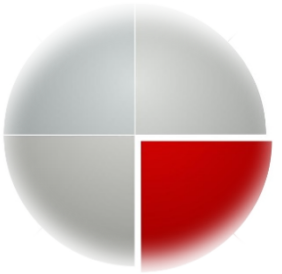
Loop iterations		Original	Refactored					Obfuscated				
	Refactoring settings		op2 in1	op3 in1	op4 in1	op4 in4	op4 in5	op2 in1	op3 in1	op4 in1	op4 in4	op4 in5
Execution trace	5	289	284	259	325	313	313	2498	2030	2092	1824	1687
	25	1149	1104	999	1285	1233	1233	9538	7450	7972	7224	6587
	125	5449	5204	4699	6085	5833	5833	44738	34550	37372	34224	31087
Simplified trace	5	280	275	250	316	304	304	295	234	331	307	289
	25	1120	1075	970	1256	1204	1204	1095	814	1231	1167	1069
	125	5320	5075	4570	5956	5704	5704	5095	3714	5731	5467	4969

Evaluation - Resilience



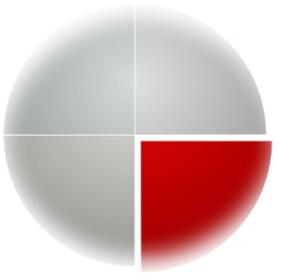
- Step 4: Compare traces

Evaluation - Resilience



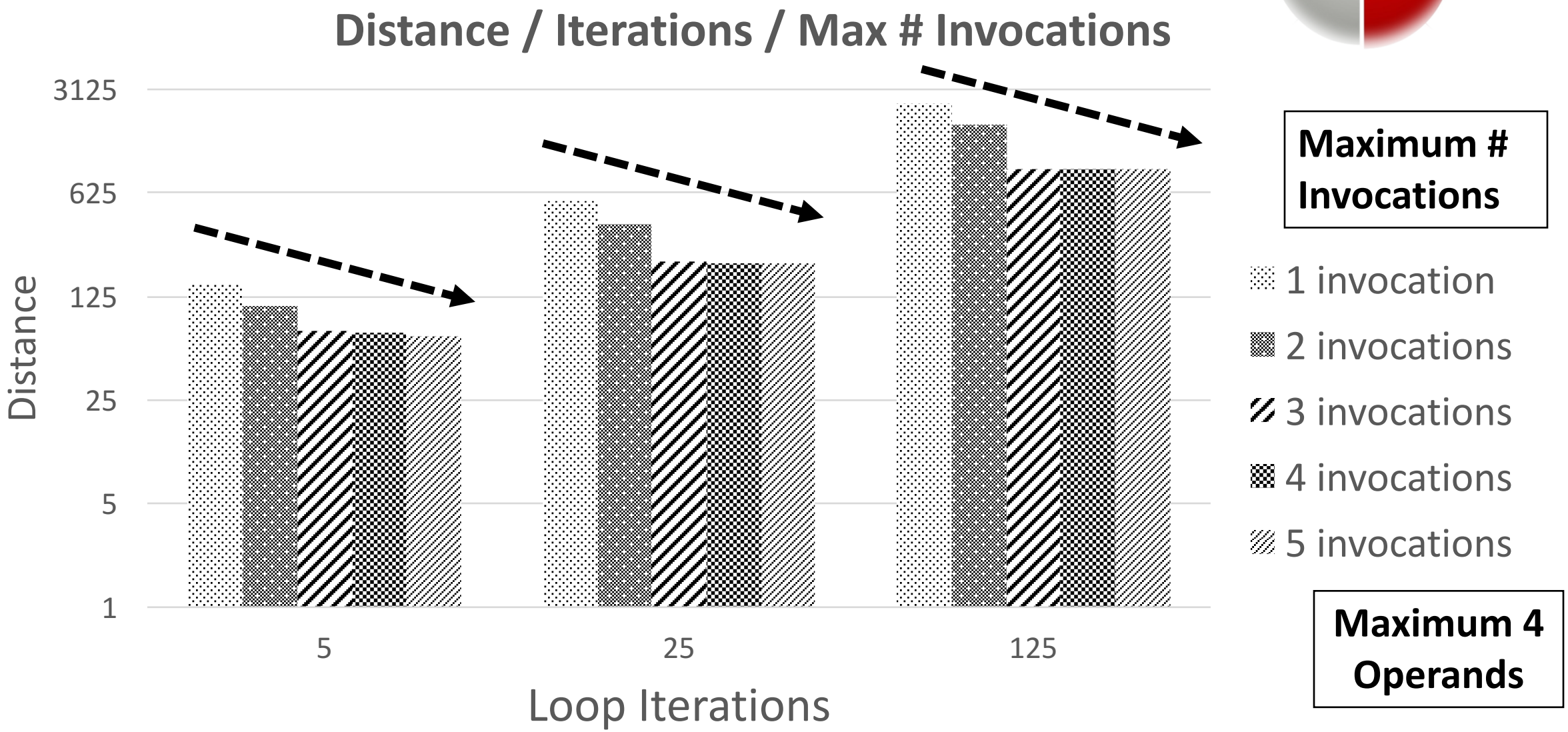
- Step 4: Compare traces
  - Levenshtein distance (edit distance)
    - Character = generic instruction, e.g. load, store, method call
    - Compare: original simplified trace with obfuscated simplified trace

Evaluation - Resilience

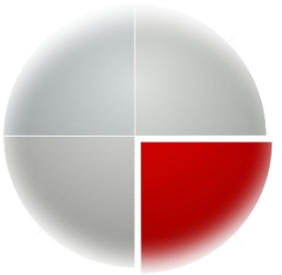


- Step 4: Compare traces
  - Levenshtein distance (edit distance)
    - Character = generic instruction, e.g. load, store, method call
    - Compare: original simplified trace with obfuscated simplified trace
  - Different obfuscation settings
    - Max number of operands
    - Max number of invocations

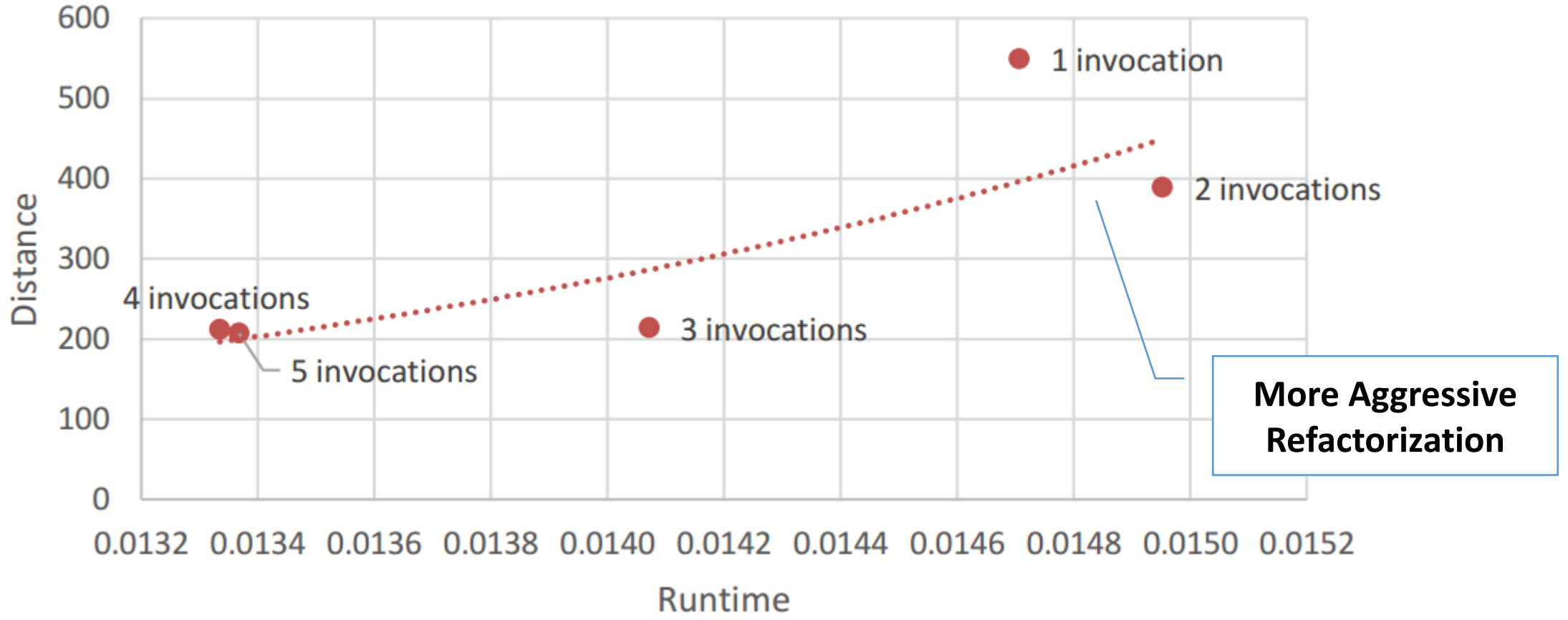
Evaluation - Resilience



Evaluation - Resilience



Resilience to Settings correlation



Evaluation - Cost

How much overhead is added?

- **Runtime overhead**

Evaluation - Cost

How much overhead is added?

- **Runtime overhead**
  - Quick Sort: Iterative and Recursive
  - Binary Search: Iterative and Recursive
  - GitHub Opensource Project: ResourceLib
  - Jungheinrich Components: Startup time

Evaluation - Cost

How much overhead is added?

- **Runtime overhead**
  - Quick Sort: Iterative and Recursive
  - Binary Search: Iterative and Recursive
  - GitHub Opensource Project: ResourceLib
  - Jungheinrich Components: Startup time

- **Size**
  - Managed code assembly on the disk

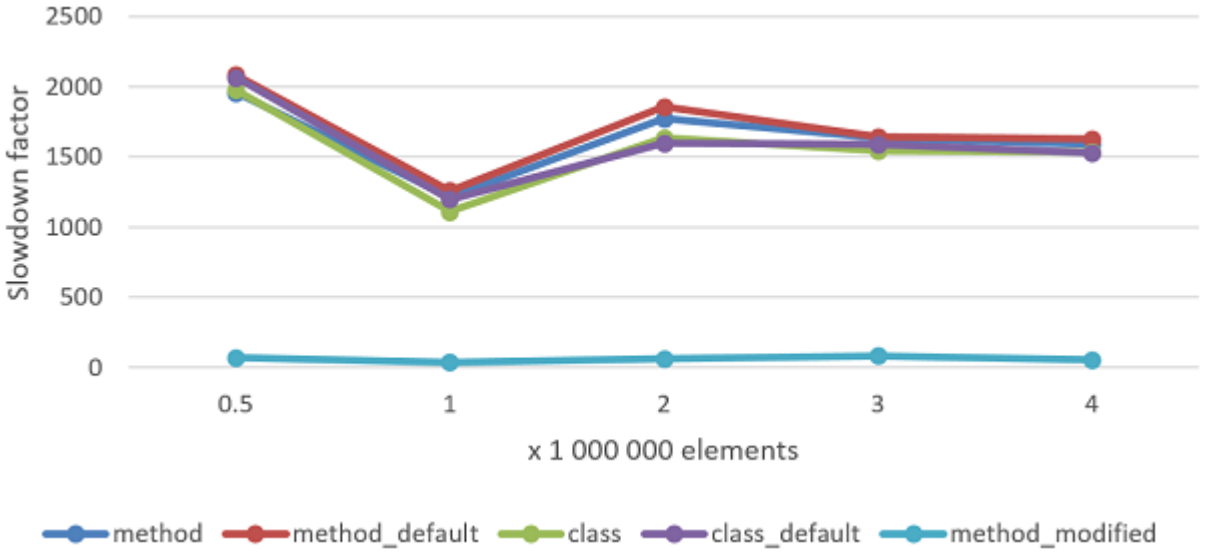
Evaluation - Cost

- **Binary Search** runtime performance – 4.000.000 elements

Runtime (seconds)	Obfuscation	original	method	method_default	class	class_default	method_modified
Iterative	Mean	0.0001038	0.1647730	0.1689987	0.1594647	0.1590038	0.0058794
	Std. dev.	0.0000412	0.0055362	0.0050380	0.0039784	0.0030870	0.0007934
	Slowdown	**-**	**1587.35**	**1628.06**	**1536.21**	**1531.77**	**56.64**
Recursive	Mean	0.0002117	3.9098402	3.9363272	3.7410884	3.8718838	0.1496612
	Std. dev.	0.0000961	0.0405461	0.0480120	0.0589763	0.0970117	0.0031326
	Slowdown	**-**	**18465.64**	**18590.73**	**17668.65**	**18286.37**	**706.83**

Evaluation - Cost

- **Binary Search** runtime performance



Binary Search iterative - slowdown trend

Binary Search recursive - slowdown trend

Evaluation - Cost

- Open Source Project: **ResourceLib C# File Resource Management Library**
  - 46 unit tests
  - 8 obfuscated methods in 4 different classes out of >100 methods, >40 classes
  - >1300 calls to obfuscated methods
  - 25 times evaluated
  - runtime performance

Runtime (seconds)	Original	Obfuscated
Min	0.876	1.415
Max	1.019	1.517
Mean	**0.945**	**1.466**
Std. dev.	0.031	0.026

+0.5

https://github.com/dblock/resourcelib

Evaluation - Cost

- Jungheinrich Components: Startup time

Evaluation - Cost

- Jungheinrich Components: Startup time

Logging Component – 4 methods obfuscated

Runtime (seconds)	Original	Obfuscated
Min	1.0892148	1.0983814
Max	1.4061824	1.4809621
Mean	**1.1928786**	**1.2407594**
Std. dev.	0.099945	0.126084

+0.05

Evaluation - Cost

- Size

	Original	Obfuscated
Jungheinrich ExitMainGui	12,800	17,920
Jungheinrich Logging	67,072	72,192
ResourceLib	75,264	145,920
Quick Sort iterative	479,232	491,520
Quick Sort recursive	483,328	491,520
Binary Search iterative	483,328	491,520
Binary Search recursive	483,328	491,520
	Size (bytes)	

Evaluation - Cost

- Size

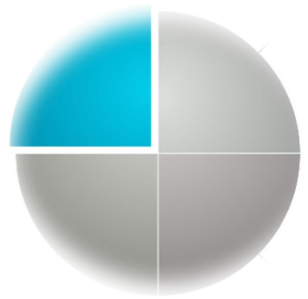
	Original	Obfuscated
Jungheinrich ExitMainGui	12,800	17,920
Jungheinrich Logging	67,072	72,192
ResourceLib	75,264	145,920
Quick Sort iterative	479,232	491,520
Quick Sort recursive	483,328	491,520
Binary Search iterative	483,328	491,520
Binary Search recursive	483,328	491,520
	Size (bytes)	

Evaluation - Cost

- Size

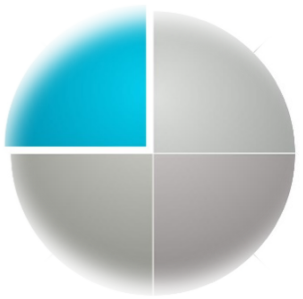
	Original	Obfuscated
Jungheinrich ExitMainGui	12,800	17,920
Jungheinrich Logging	67,072	72,192
ResourceLib	75,264	145,920
Quick Sort iterative	479,232	491,520
Quick Sort recursive	483,328	491,520
Binary Search iterative	483,328	491,520
Binary Search recursive	483,328	491,520
	Size (bytes)	

Evaluation - Stealth



How well does obfuscated code blend in the original code?

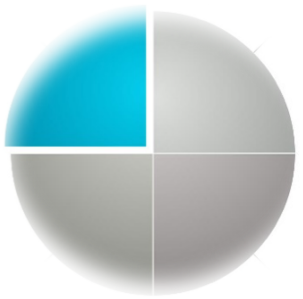
Evaluation - Stealth



How well does obfuscated code blend in the original code?

- **Method Signatures maintained**
  - No broken dependencies

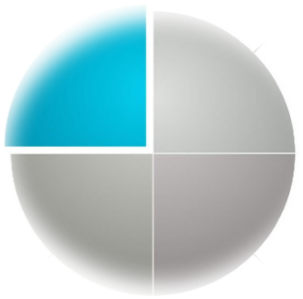
Evaluation - Stealth



How well does obfuscated code blend in the original code?

- **Method Signatures maintained**
  - No broken dependencies
- **Side effects maintained**
  - Fields
  - Lambda expressions

Evaluation - Stealth



How well does obfuscated code blend in the original code?

- **Method Signatures maintained**
  - No broken dependencies
- **Side effects maintained**
  - Fields
  - Lambda expressions
- **Annotations for non-intrusive tagging**
  - Use annotations to mark which methods to obfuscate

Conclusion – To Obfuscate or Not To Obfuscate?

Conclusion – To Obfuscate or Not To Obfuscate?

Software Protection

- there is **NO** silver bullet!!!
- always consider the **ENTIRE** system
- use **OWASP** testing guide
- consider concurrency issues

Conclusion – To Obfuscate or Not To Obfuscate?

Software Protection

- there is **NO** silver bullet!!!
- always consider the **ENTIRE** system
- use **OWASP** testing guide
- consider concurrency issues

Evaluation

- do you ***really*** need to obfuscate everything?
- consider ***worst case*** scenario
- use *potency, resilience, cost,* and *stealth* as criteria

Conclusion – To Obfuscate or Not To Obfuscate?

Software Protection

- there is **NO** silver bullet!!!
- always consider the **ENTIRE** system
- use **OWASP** testing guide
- consider concurrency issues

Evaluation

- do you ***really*** need to obfuscate everything?
- consider ***worst case*** scenario
- use *potency, resilience, cost,* and *stealth* as criteria

Reverse Engineering

- reconstruct source code from simplified traces
- think as the ***other side***

Conclusion – To Obfuscate or Not To Obfuscate?

Software Protection

- there is **NO** silver bullet!!!
- always consider the **ENTIRE** system
- use **OWASP** testing guide
- consider concurrency issues

Little protection is always better than no protection

Evaluation

- do you ***really*** need to obfuscate everything?
- consider ***worst case*** scenario
- use *potency, resilience, cost,* and *stealth* as criteria

Reverse Engineering

- reconstruct source code from simplified traces
- think as the ***other side***

Thank you for your attention!
Any questions?

References

- [Rolles2009] Unpacking Virtualization Obfuscators
- [Sharif2009] Automatic Reverse Engineering of Malware Emulators
- [Coogan2011] Deobfuscation of Virtualization-Obfuscated Software
- [Kinder2012] Towards Static Analysis of Virtualization-Obfuscated Binaries
- [Yadegari2015] A Generic Approach to Automatic Deobfuscation of Executable Code
- [Cazalas2014] Probing the Limits of Virtualized Software Protection
- [Anckaert2006] Proteus: Virtualization for Diversified Tamper-Resistance
- [Cohen1993] Operating system protection through program evolution
- [Collberg1998] Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs
- Microsoft Roslyn Project: https://github.com/dotnet/roslyn

Obfuscation tools

	C++	Java	Python	Javascript
1.	cxx-obfus	Jfuscator	pyminifier	garble
2.	StarForce C++	Allatori	pyobfuscate	Oxyry
3.	Obfuscator-LLVM	Dash-O-Pro		
4.	Obfusion	GuardIT for Java		
5.	Cryptanium Code Protection	Proguard		
6.		Stringer		
7.		yGuard		

Further Reading

- http://www.excelsior-usa.com/articles/java-obfuscators.html
- https://www.owasp.org/index.php/Bytecode_obfuscation
- http://www.foundbit.com/en/resources/languages/cpp/advanced/articles/cpp-code-obfusc.html