

VB 2022

Combating control flow flattening in .NET malware

kaspersky

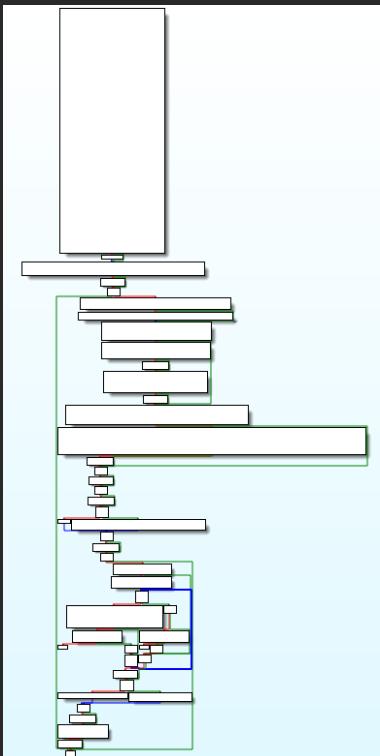
Georgy Kucherin
Kaspersky GReAT
Junior Security
Researcher
Keybase: gkucherin

Obfuscation is widespread in malware

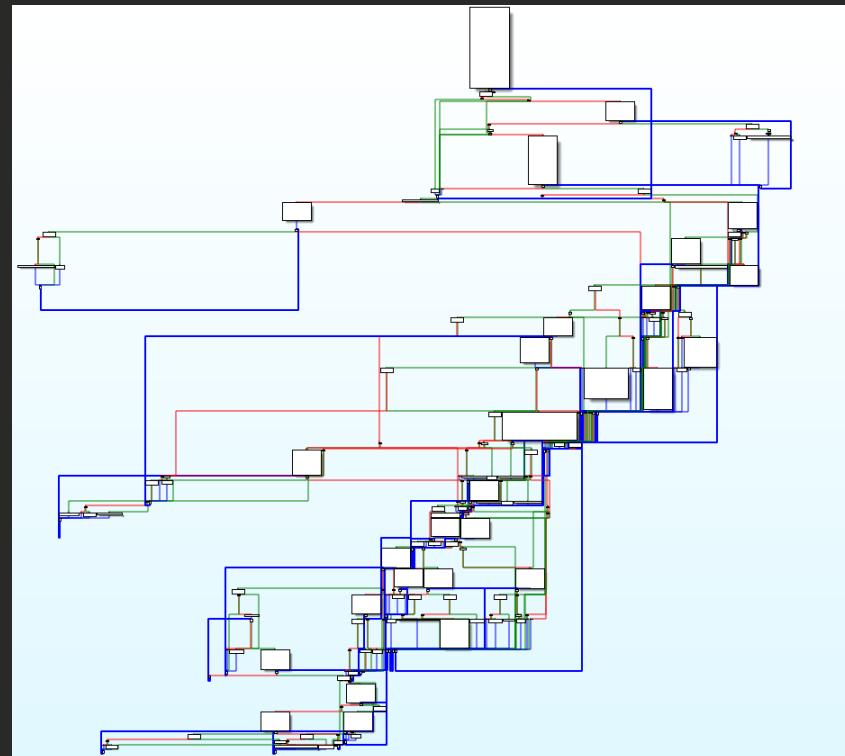
Malware developers obfuscate their code to extend reverse engineering time

Sometimes developers include custom obfuscation, making code even harder to reverse engineer

DoubleZero .NET wiper



Original code



Obfuscated code

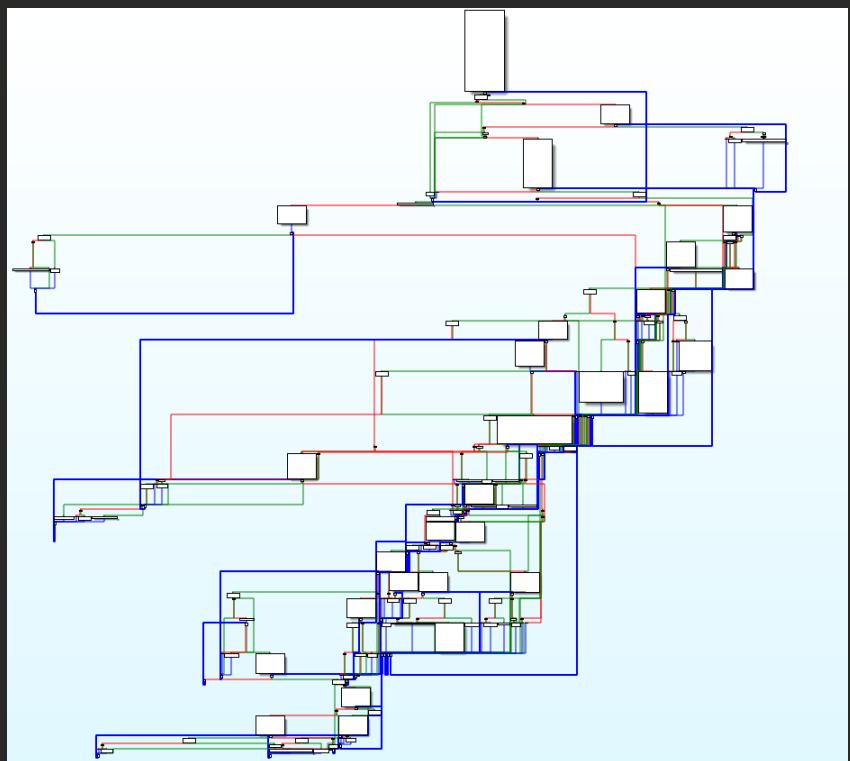
Control flow flattening technique

Is considered to be
one of the hardest-
to-crack techniques

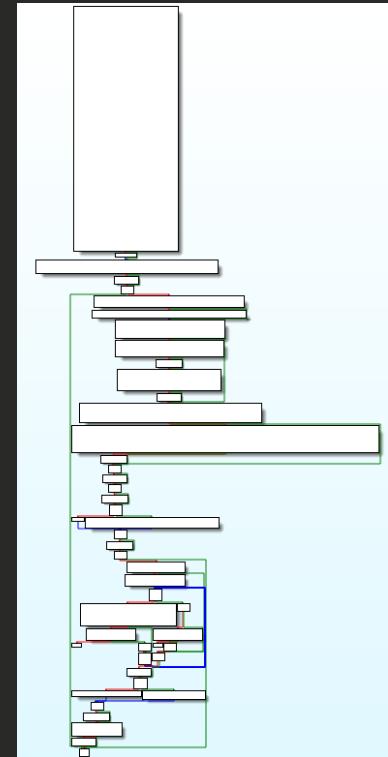
Has been widely
researched for
C/C++ binaries

Sparse information about
C# binaries
(DoubleZero's
programming language)

Our goal



Obfuscated code



Original code

Let's look at the code!

```
264
265
266     if (num2 <= 542)
267     {
268         if (num2 != 219)
269         {
270             if (num2 != 494)
271             {
272                 if (num2 == 542)
273                 {
274                     _9f6717951b3535fb甸歟喽𠮣厯瓶類弱 엔 솔拐囉採囉梟𠮣榜誥囉叢浏囉蠻經헬製喽칼盍昤.RtlAdjustPrivilege(9UL, true, false, ref flag);
275                 }
276             }
277             FieldDirection fieldDirection = (FieldDirection)obj2;
278         }
279     }
280 }
281 else if (num2 != 544)
282 {
283     if (num2 != 559)
284     {
285         if (num2 == 581)
286         {
287             _9f6717951b3535fb甸歟喽𠮣厯瓶類弱 엔 솔拐囉採囉梟𠮣榜誥囉叢浏囉蠻經헬製喽칼盍昤.RtlAdjustPrivilege(18UL, true, false, ref flag);
288         }
289     }
290     else
291     {
292         _9f6717951b3535fb甸歟喽𠮣厯瓶類弱 엔 솔拐囉採囉梟𠮣榜誥囉叢浏囉蠻經헬製喽칼盍昤.RtlAdjustPrivilege(17UL, true, false, ref flag);
293     }
294 }
295 else
296 {
297     _9f6717951b3535fb甸歟喽𠮣厯瓶類弱 엔 솔拐囉採囉梟𠮣榜誥囉叢浏囉蠻經헬製喽칼盍昤.RtlAdjustPrivilege(19UL, true, false, ref flag);
298 }
299 j++;
300 }
301 goto IL_2A1;
```

4 lines turned into 317 lines Code is hard to read

```
1 object[] array = new object[]
2 {
3     new int[] { 90, 2089875171, 90, 1318285745, 10, 90, 886806518, 10, 180 },
4     new int[] { 90, -986721120, 90, 1218371032, 20, 90, 1318285745, 10, 90, 886806518, 10, 180 },
5     new int[] { 90, -986721052, 90, 1218371032, 20, 90, 1318285745, 10, 90, 886806518, 10, 180 },
6     new int[] { 90, 2089875154, 90, 1318285745, 10, 90, 886806518, 10, 180 },
7     new int[] { 90, 2089875047, 90, 1318285745, 10, 90, 886806518, 10, 180 }
8 };
9 while (i < 5)
10 {
11     int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
12     if (num == 13)
13     {
14         goto IL_29A;
15     }
16     if (num == 121)
17     {
18         <code omitted>
19     }
20     IL_2A1:
21     i++;
22     continue;
23     IL_29A:
24     RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
25     goto IL_2A1;
26 }
```

Taking a close look
at the obfuscation
pattern

```
1 object[] array = new object[]
2 {
3     new int[] { 90,2089875171,90,1318285745,10,90,886806518,10,180 },
4     new int[] { 90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
5     new int[] { 90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
6     new int[] { 90,2089875154,90,1318285745,10,90,886806518,10,180 },
7     new int[] { 90,2089875047,90,1318285745,10,90,886806518,10,180 }
8 };
9 while (i < 5)
10 {
11     int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
12     if (num == 13)
13     {
14         goto IL_29A;
15     }
16     if (num == 121)
17     {
18         <code omitted>
19     }
20     IL_2A1:
21     i++;
22     continue;
23     IL_29A:
24     RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
25     goto IL_2A1;
26 }
```

Initialization of an array of integer arrays

```
1 object[] array = new object[]
2 {
3     new int[] { 90,2089875171,90,1318285745,10,90,886806518,10,180 },
4     new int[] { 90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
5     new int[] { 90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
6     new int[] { 90,2089875154,90,1318285745,10,90,886806518,10,180 },
7     new int[] { 90,2089875047,90,1318285745,10,90,886806518,10,180 }
8 };
9 while (i < 5)
10 {
11     int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
12     if (num == 13)
13     {
14         goto IL_29A;
15     }
16     if (num == 121)
17     {
18         <code omitted>
19     }
20     IL_2A1:
21     i++;
22     continue;
23     IL_29A:
24     RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
25     goto IL_2A1;
26 }
```

Loop over the
array of
integer arrays

```
1 object[] array = new object[]
2 {
3     new int[] { 90,2089875171,90,1318285745,10,90,886806518,10,180 },
4     new int[] { 90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
5     new int[] { 90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
6     new int[] { 90,2089875154,90,1318285745,10,90,886806518,10,180 },
7     new int[] { 90,2089875047,90,1318285745,10,90,886806518,10,180 }
8 };
9 while (i < 5)
10 {
11     int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
12     if (num == 13)
13     {
14         goto IL_29A;
15     }
16     if (num == 121)
17     {
18         <code omitted>
19     }
20     IL_2A1:
21     i++;
22     continue;
23     IL_29A:
24     RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
25     goto IL_2A1;
26 }
```

**Integer decryption
function**
**(input: integer array,
output: integer)**

```
1 object[] array = new object[]
2 {
3     new int[] { 90,2089875171,90,1318285745,10,90,886806518,10,180 },
4     new int[] { 90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
5     new int[] { 90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180 },
6     new int[] { 90,2089875154,90,1318285745,10,90,886806518,10,180 },
7     new int[] { 90,2089875047,90,1318285745,10,90,886806518,10,180 }
8 };
9 while (i < 5)
10 {
11     int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
12     if (num == 13)
13     {
14         goto IL_29A;
15     }
16     if (num == 121)
17     {
18         <code omitted>
19     }
20     IL_2A1:
21     i++;
22     continue;
23     IL_29A:
24     RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
25     goto IL_2A1;
26 }
```

We take one of the two branches depending on the integer decrypted from array[i]

Observations

Control flow flattening rearranges code lines in a random order

During execution, the program uses an array of integer arrays to execute lines in the correct order

How to perform unflattening?

1. Determine the correct execution order from the array of integer arrays.
2. Find all lines of code that correspond to each execution order number.
3. Rearrange code in accordance with the execution order.

Determining the execution order

```
object[] array = new object[]  
{  
    new int[]  
    {90,2089875171,90,1318285745,10,90,886806518,10,180},  
    ...  
};
```

We can use the two array types as a pattern
(as we will see later, this pattern is efficient enough)

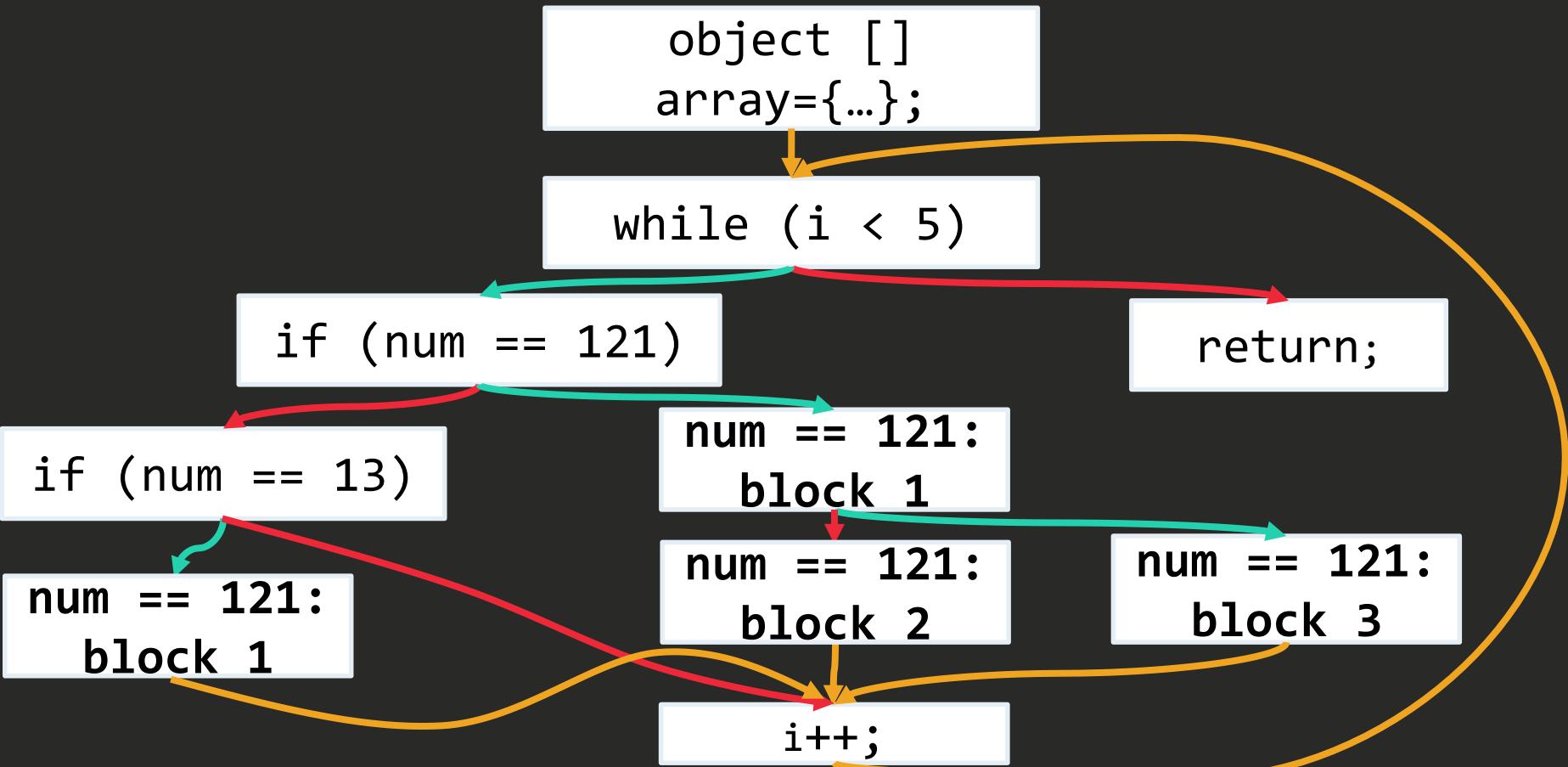
Extracting lines

Can we take the decompiled code and extract lines inside if statements?

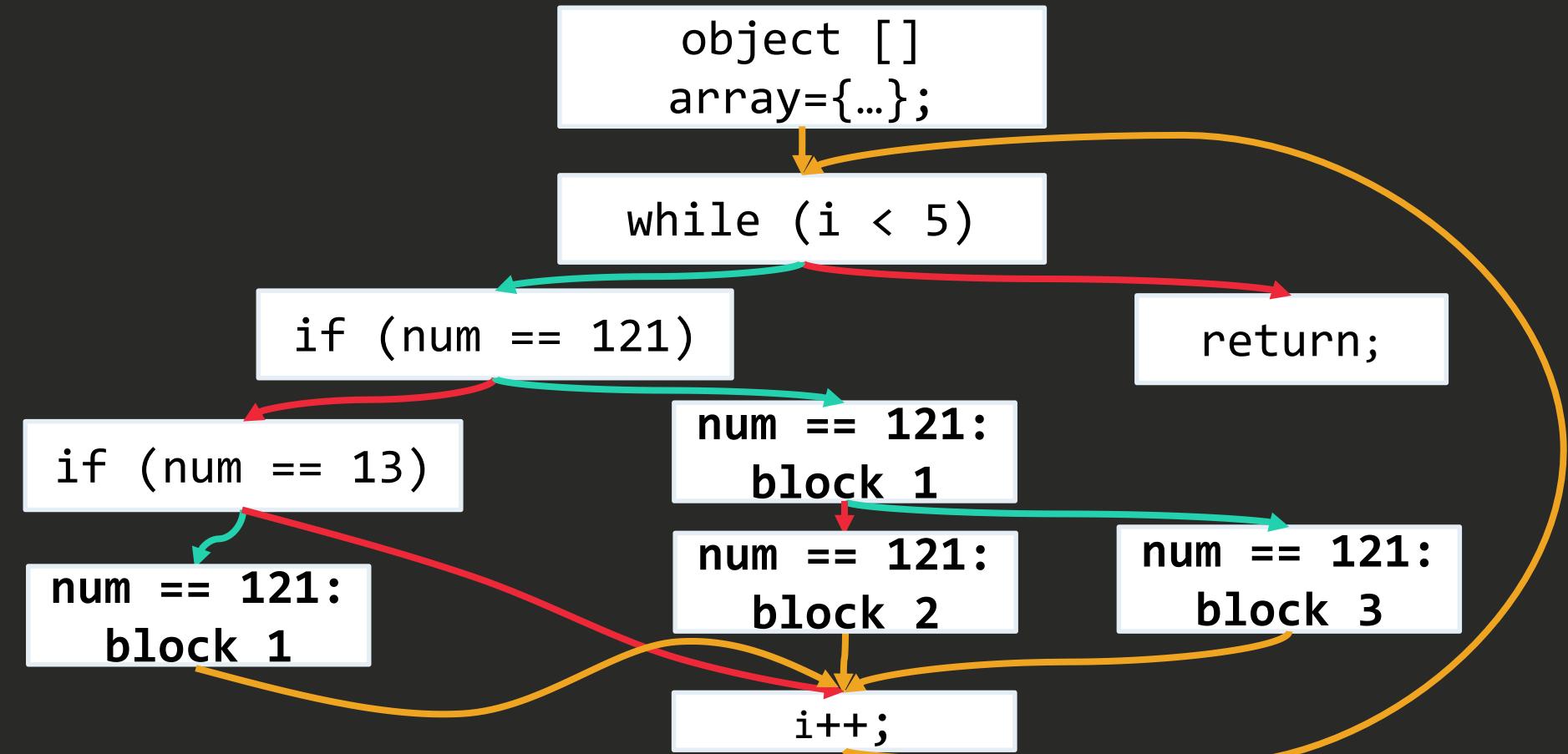
```
int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
if (num == 13)
{
    goto IL_29A;
}
if (num == 121)
{
    <code omitted>
}
IL_2A1:
i++;
continue;
IL_29A:
RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
goto IL_2A1;
```

Handling goto statements
would be problematic

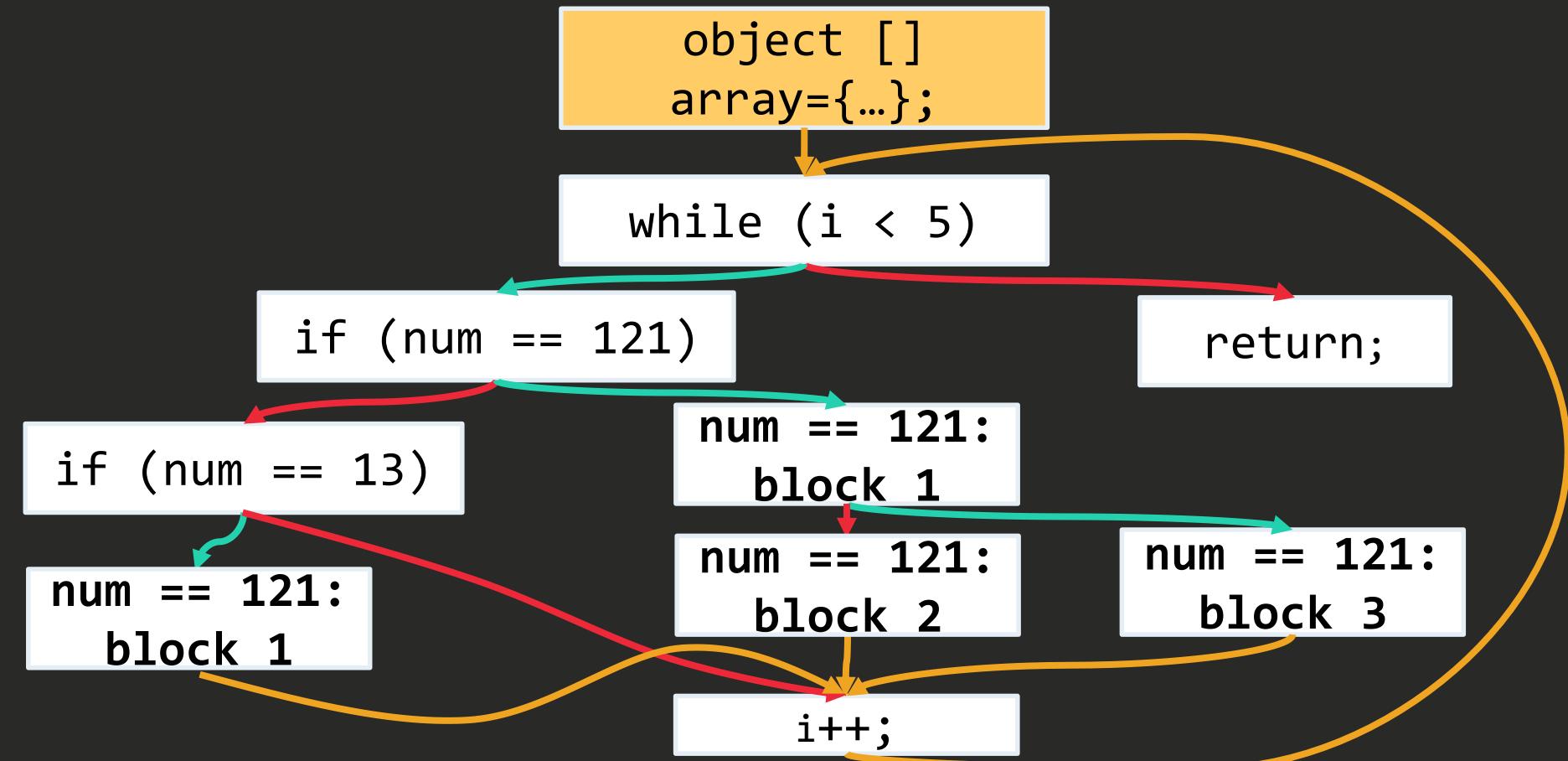
A better way of extracting code



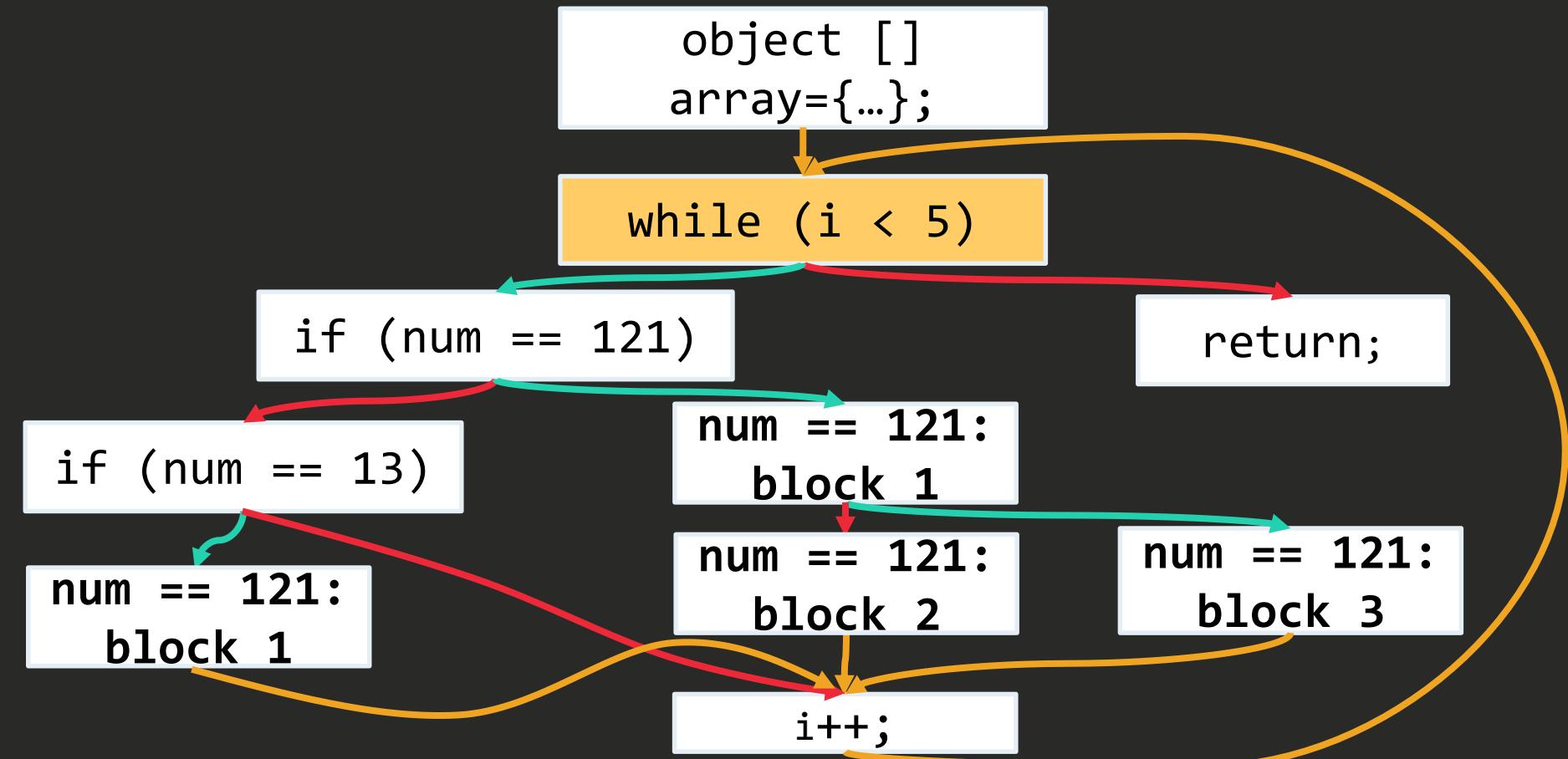
Goal: mark all blocks with bold text



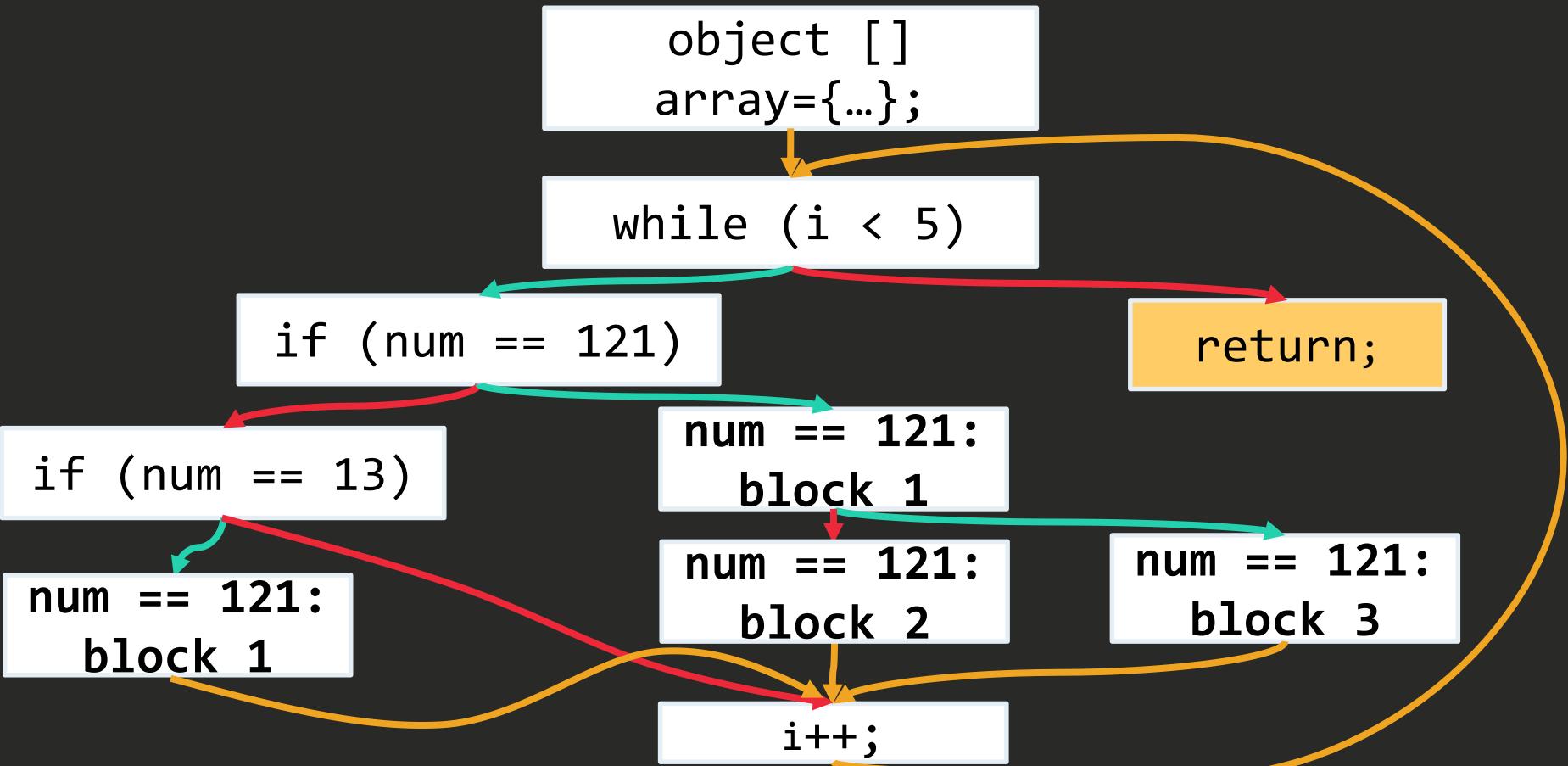
Traverse the graph until we reach an if



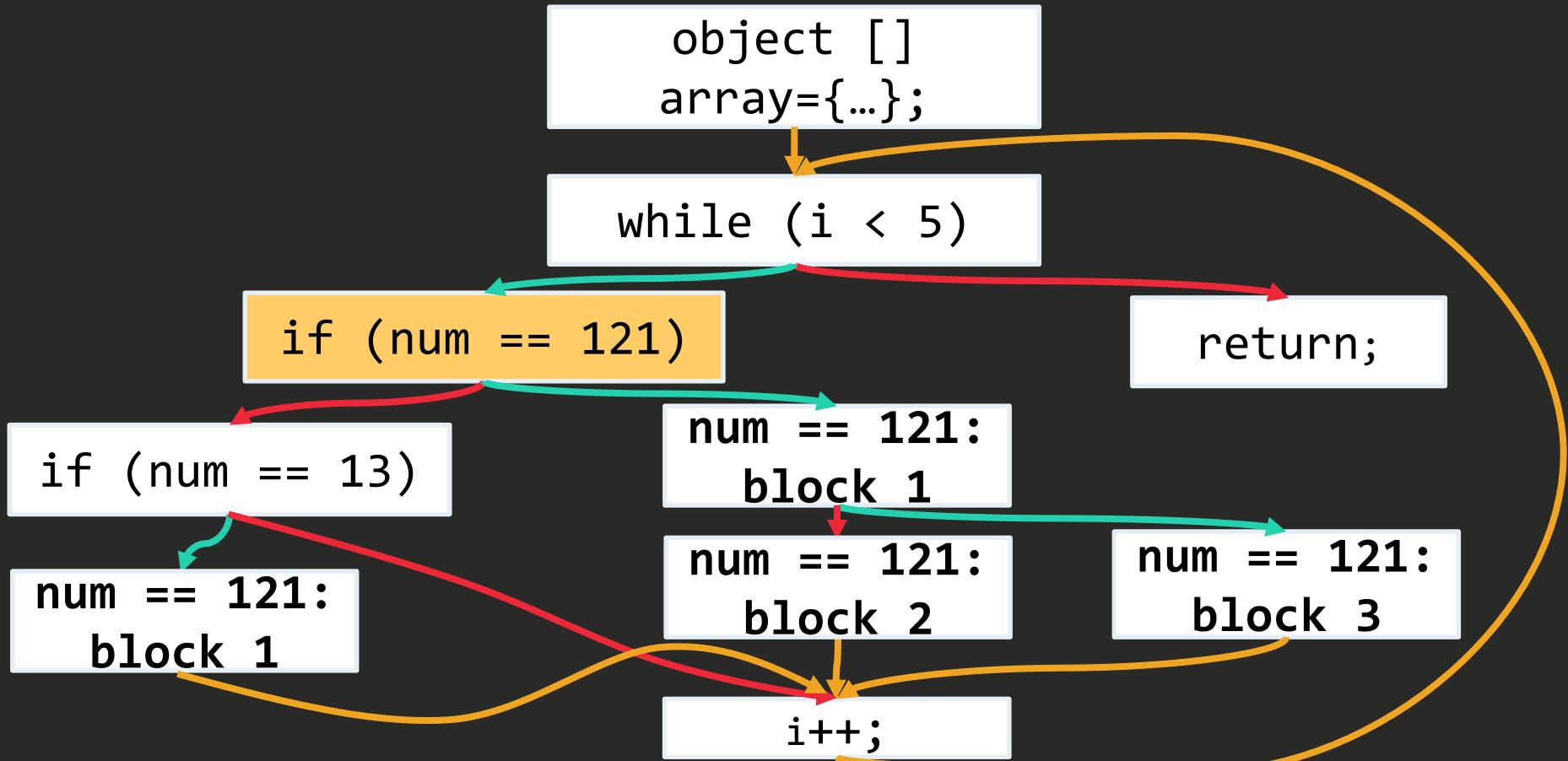
Traverse the graph until we reach an if



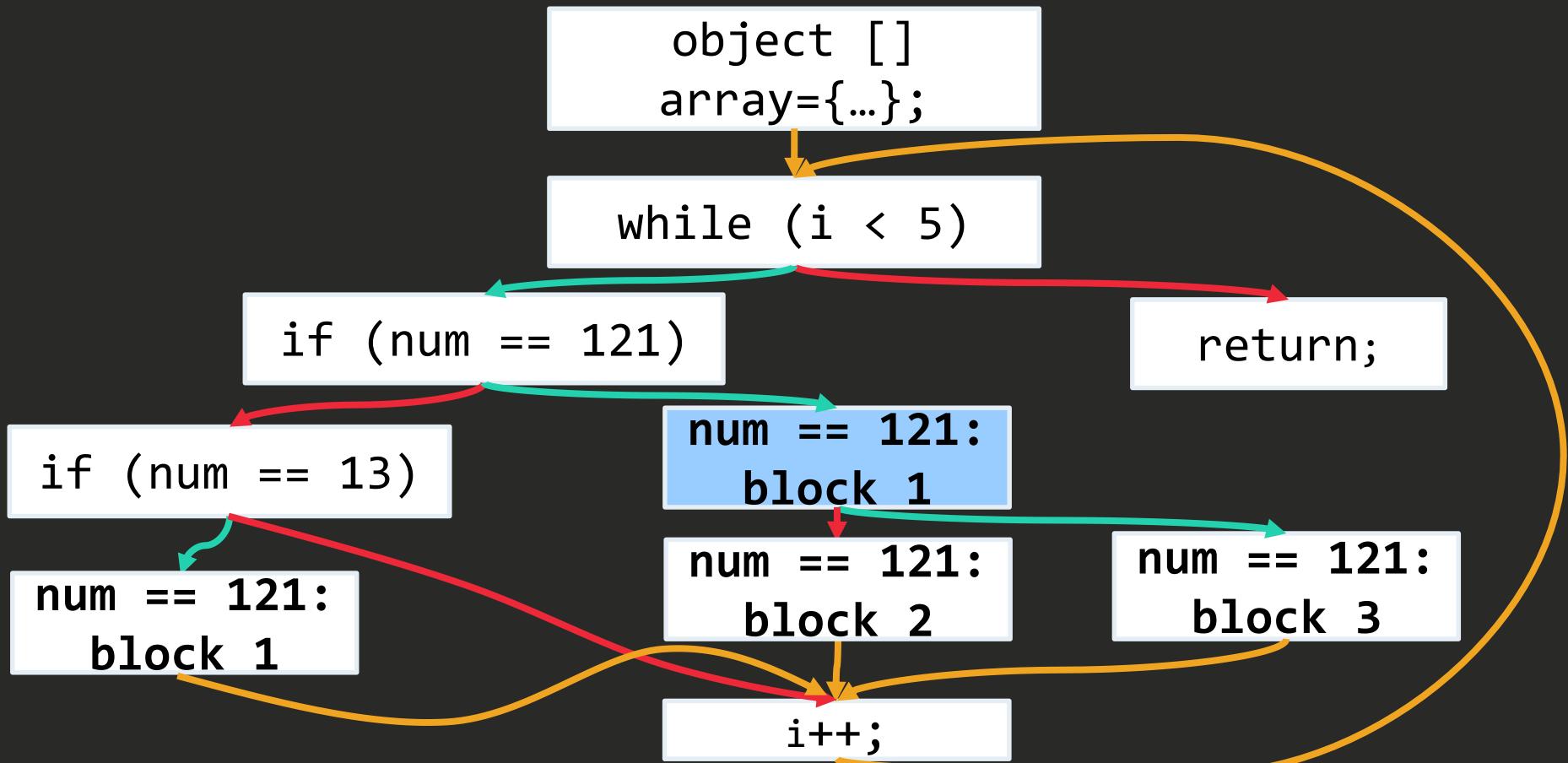
Traverse the graph until we reach an if



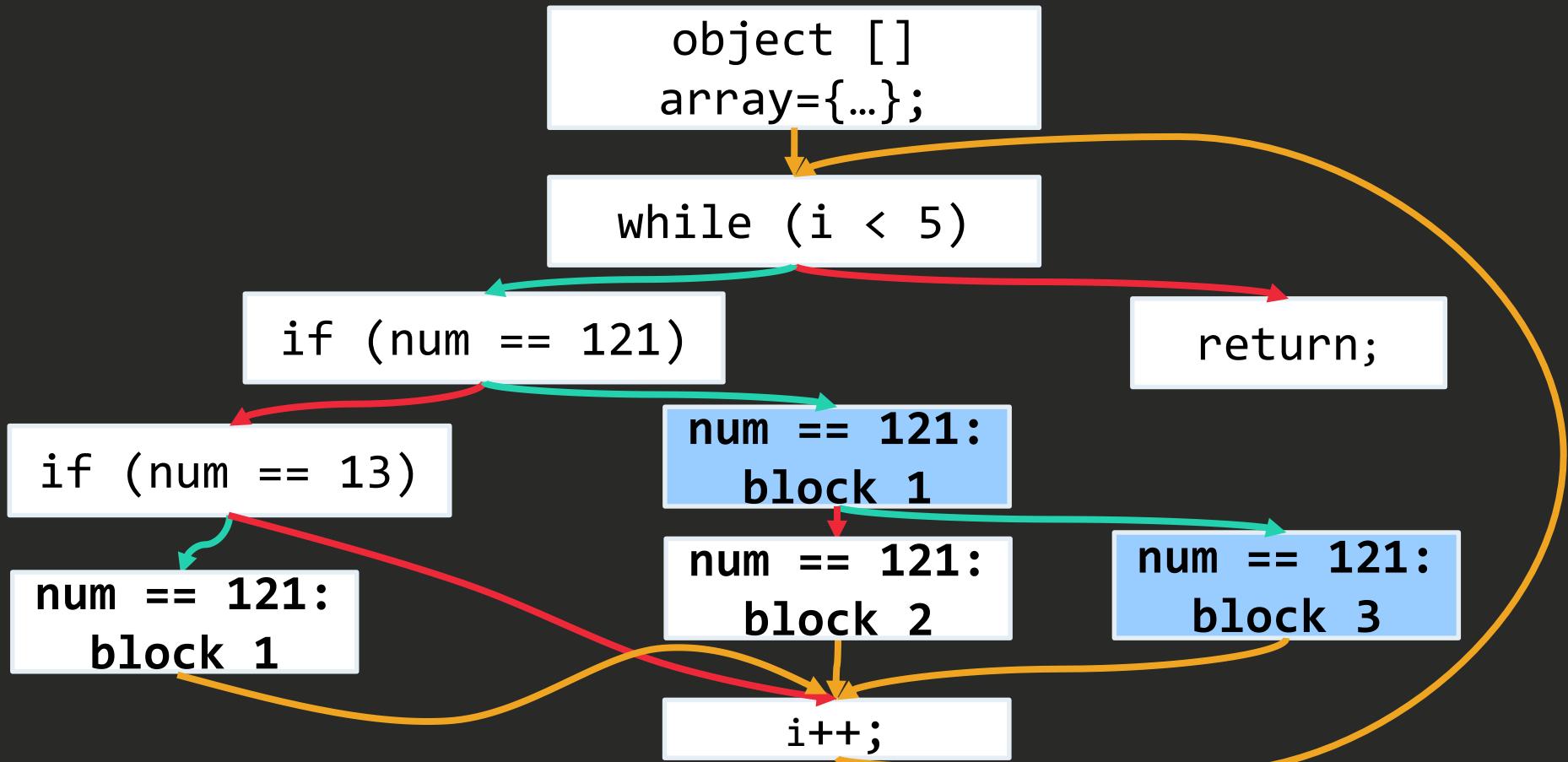
We have reached an if statement



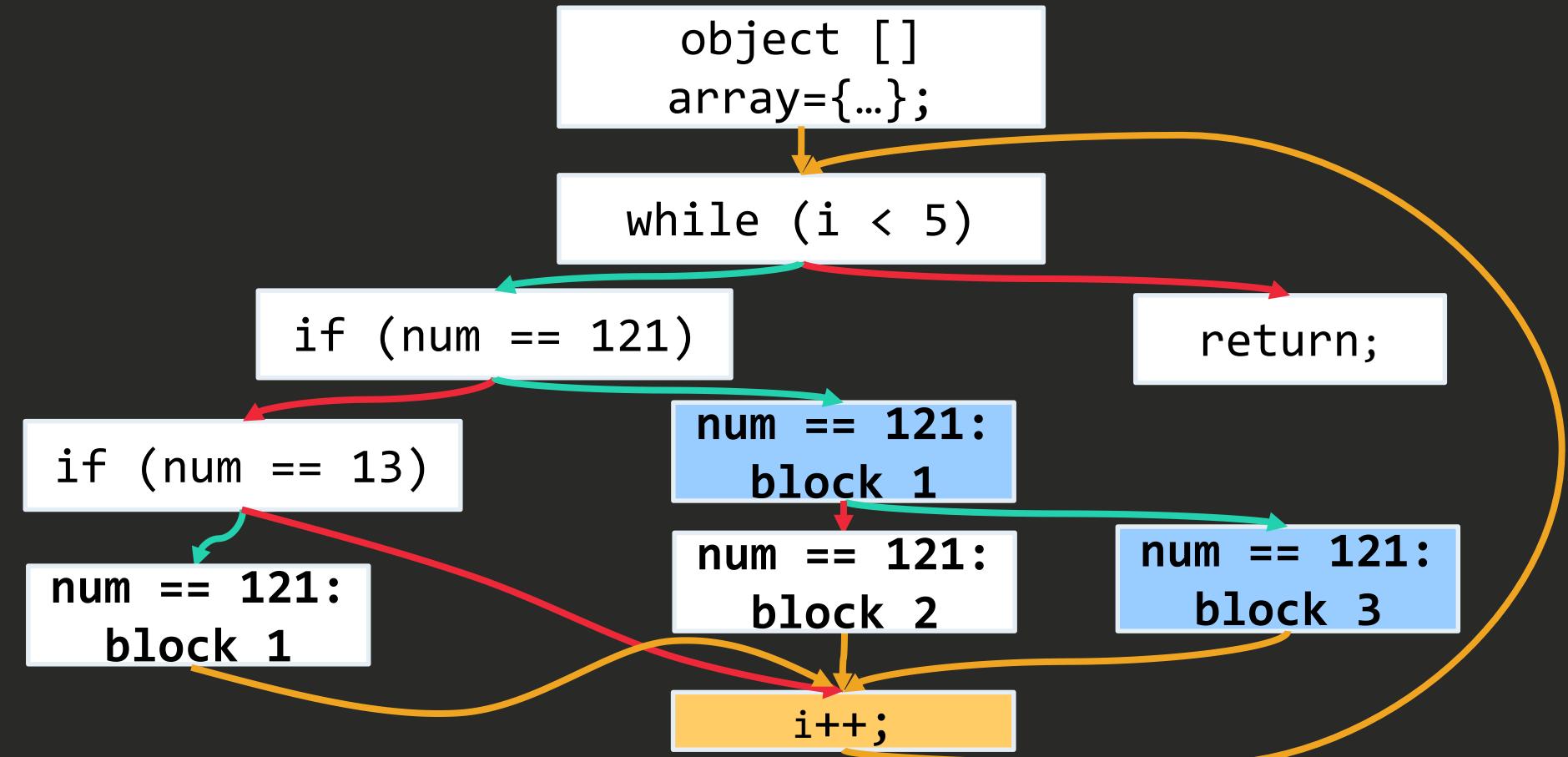
Mark all blocks branching out of the if



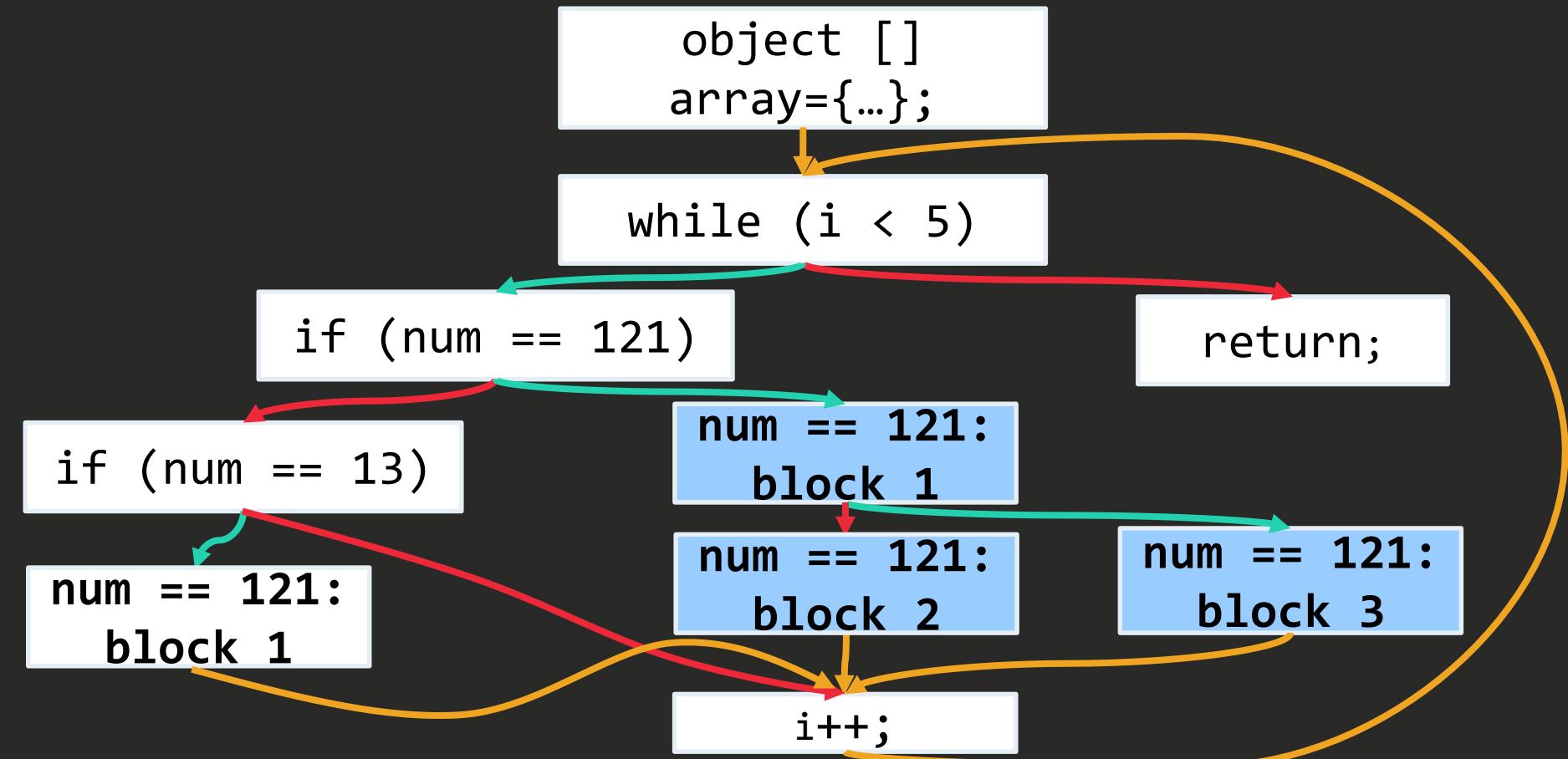
Mark all blocks branching out of the if



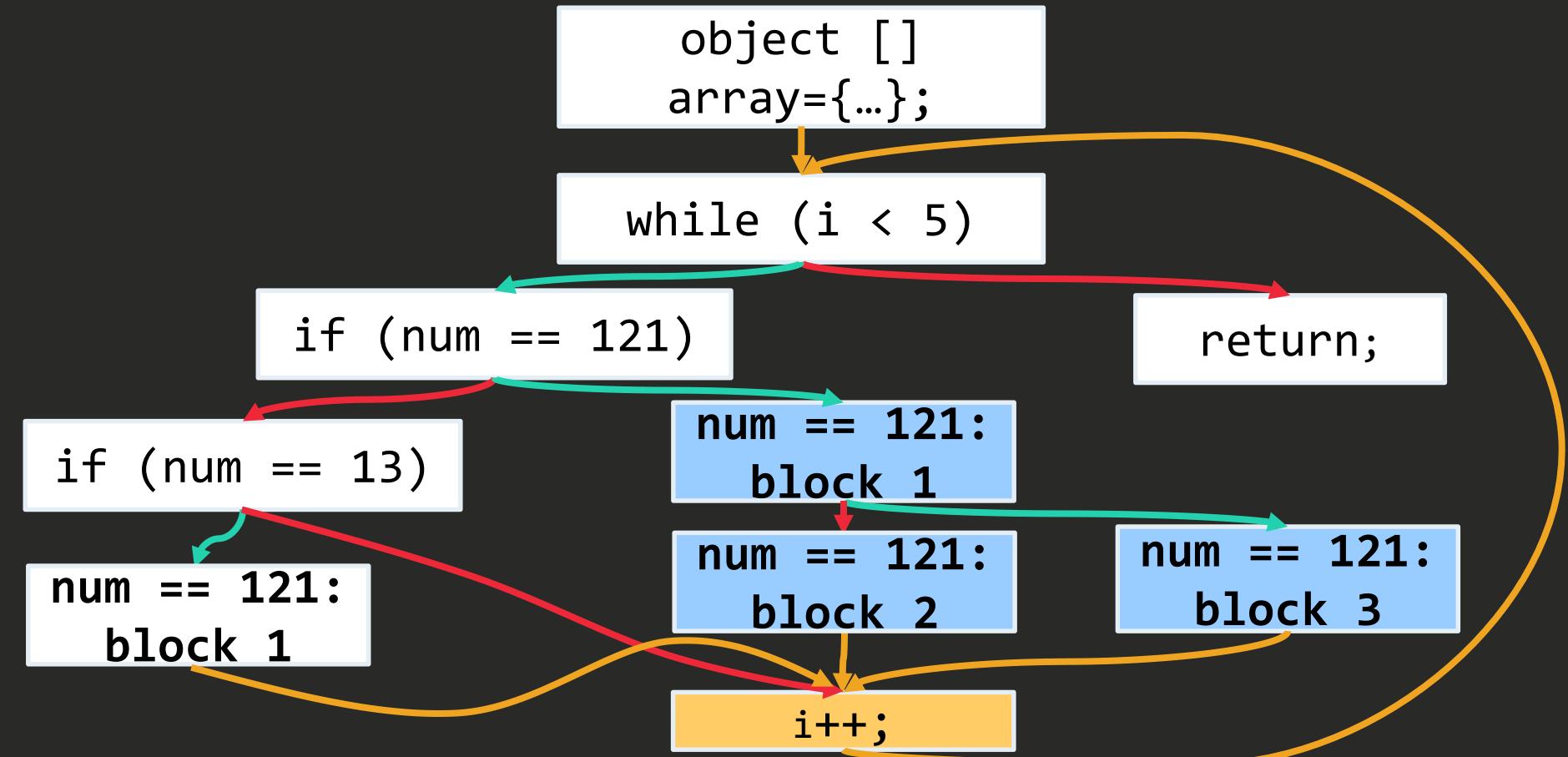
Stop when we reach the `i++` block



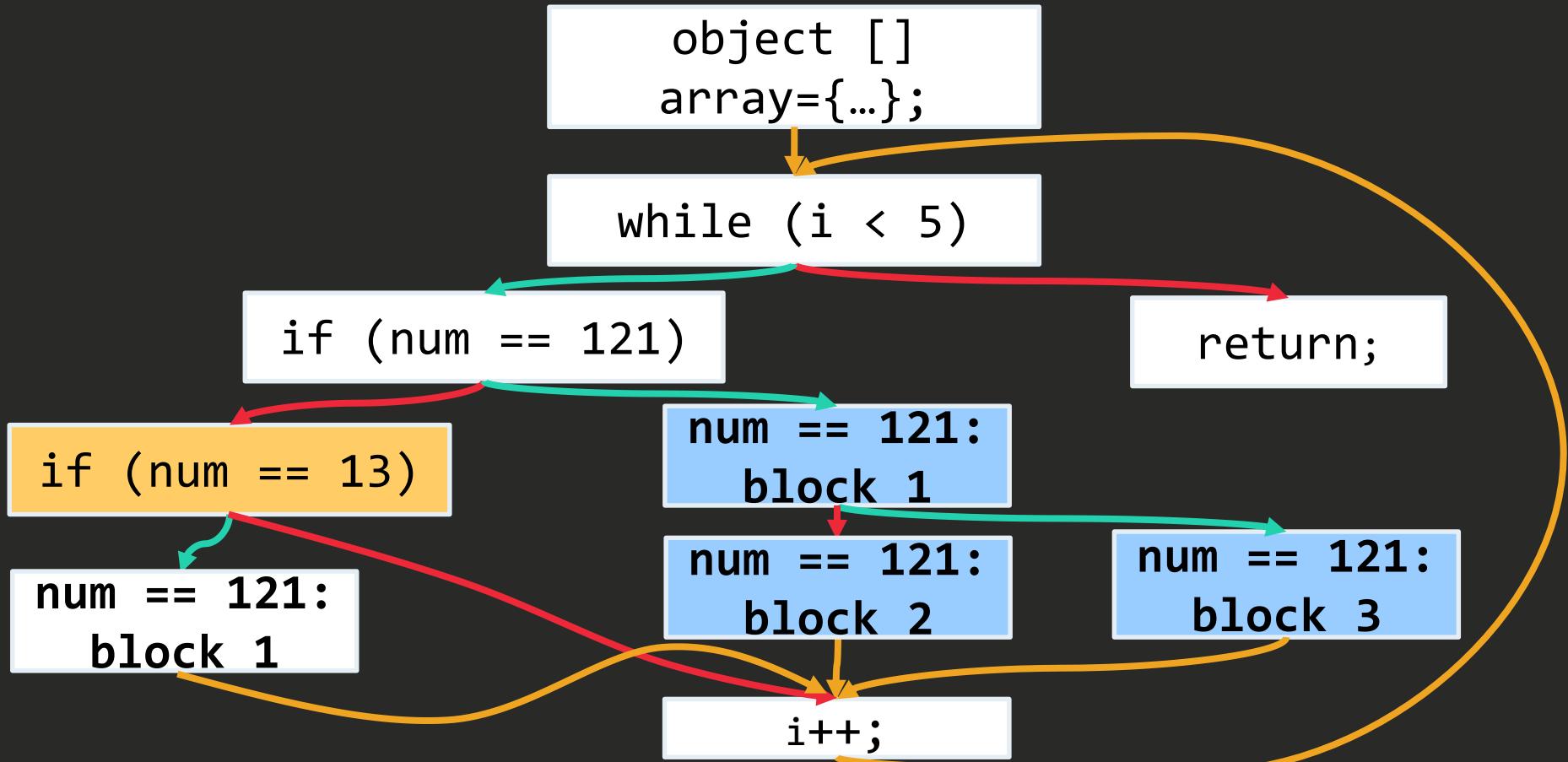
Mark all blocks branching out of the if



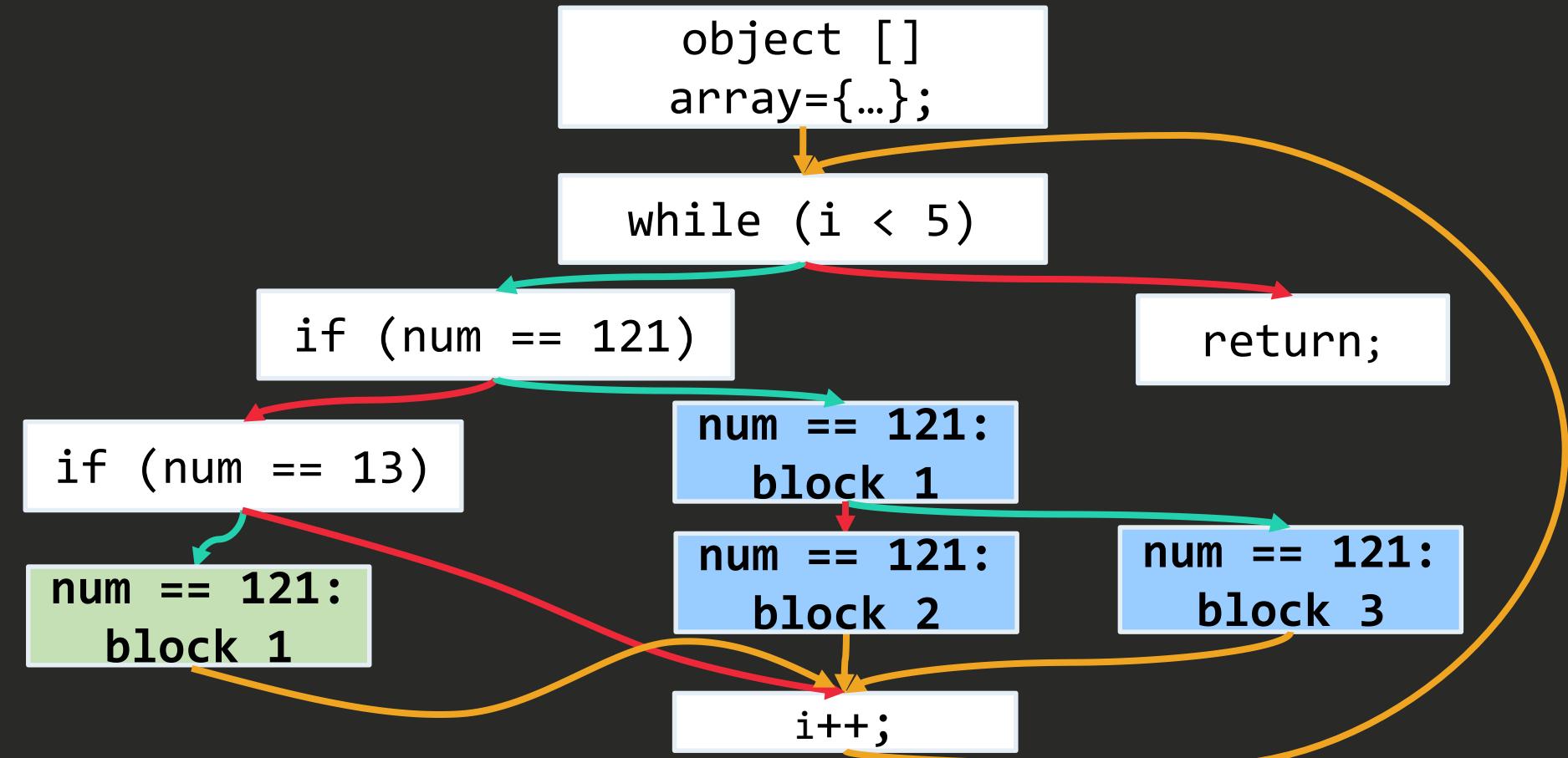
Stop when we reach the `i++` block



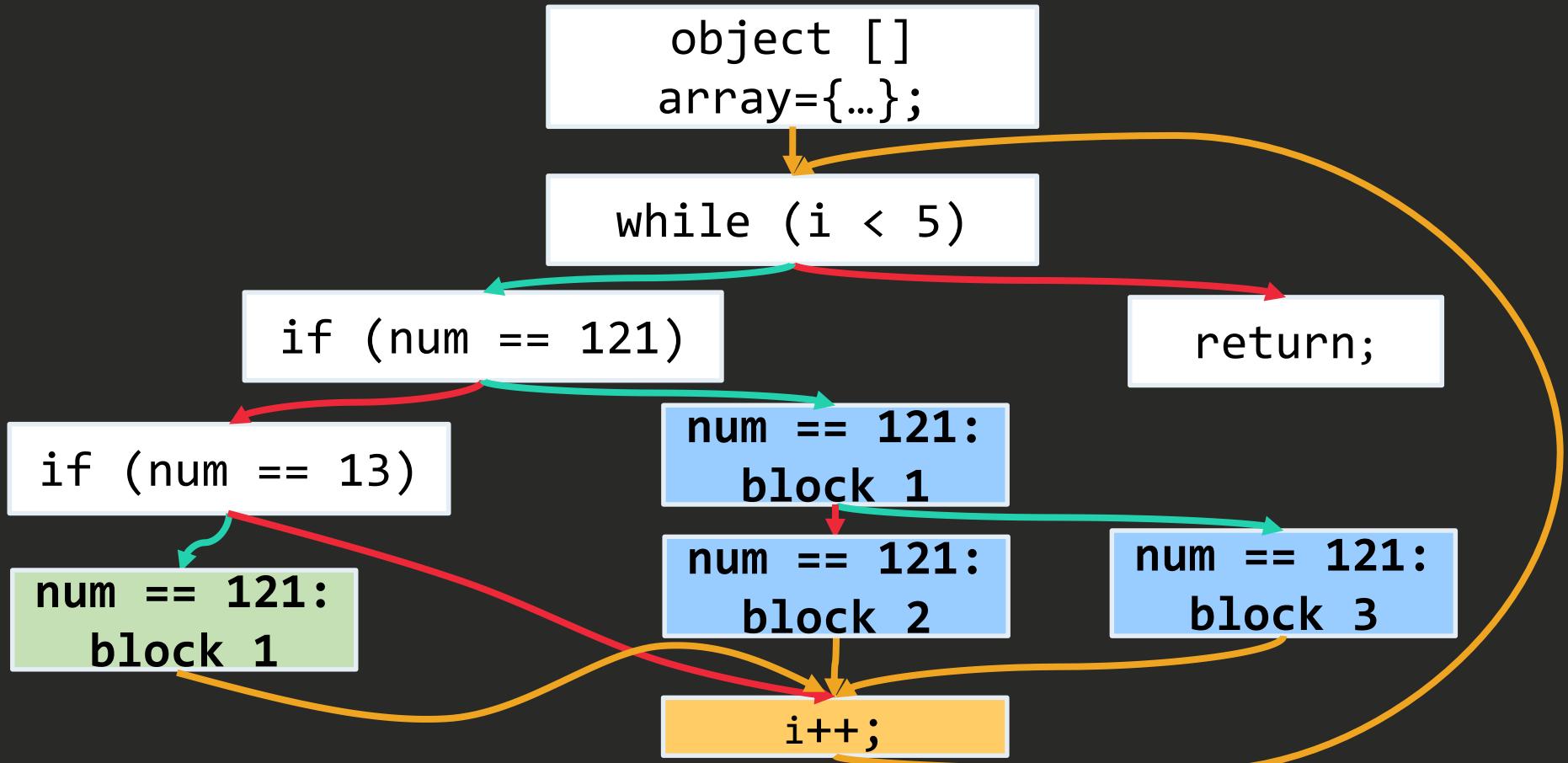
The same for num == 13



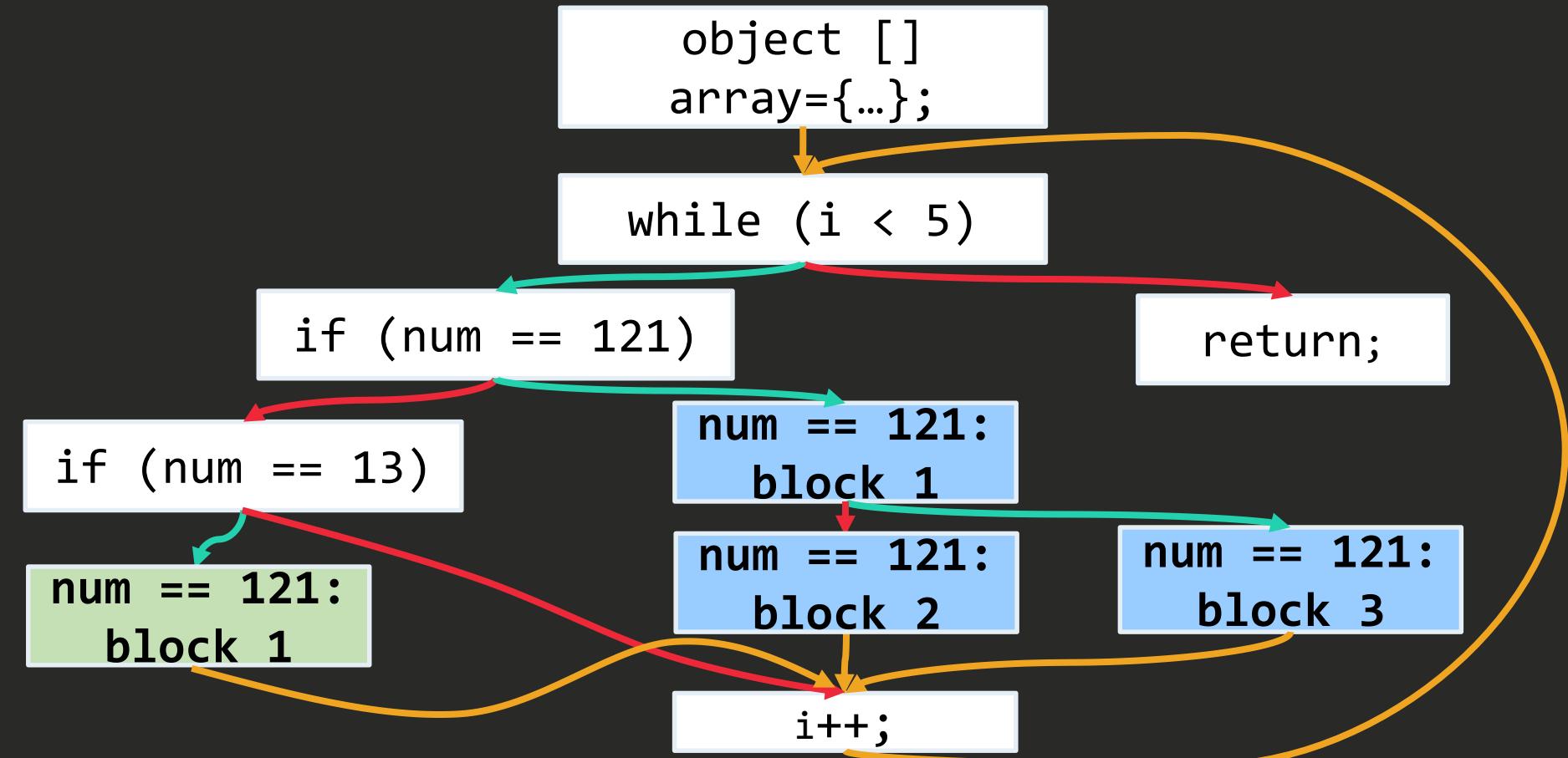
The same for num == 13



The same for num == 13



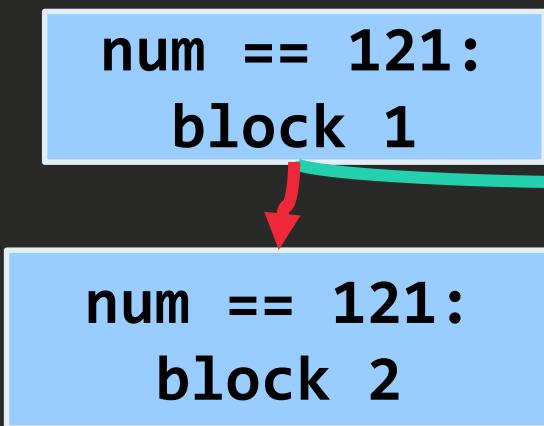
Done!



Connect colored blocks to unflatten

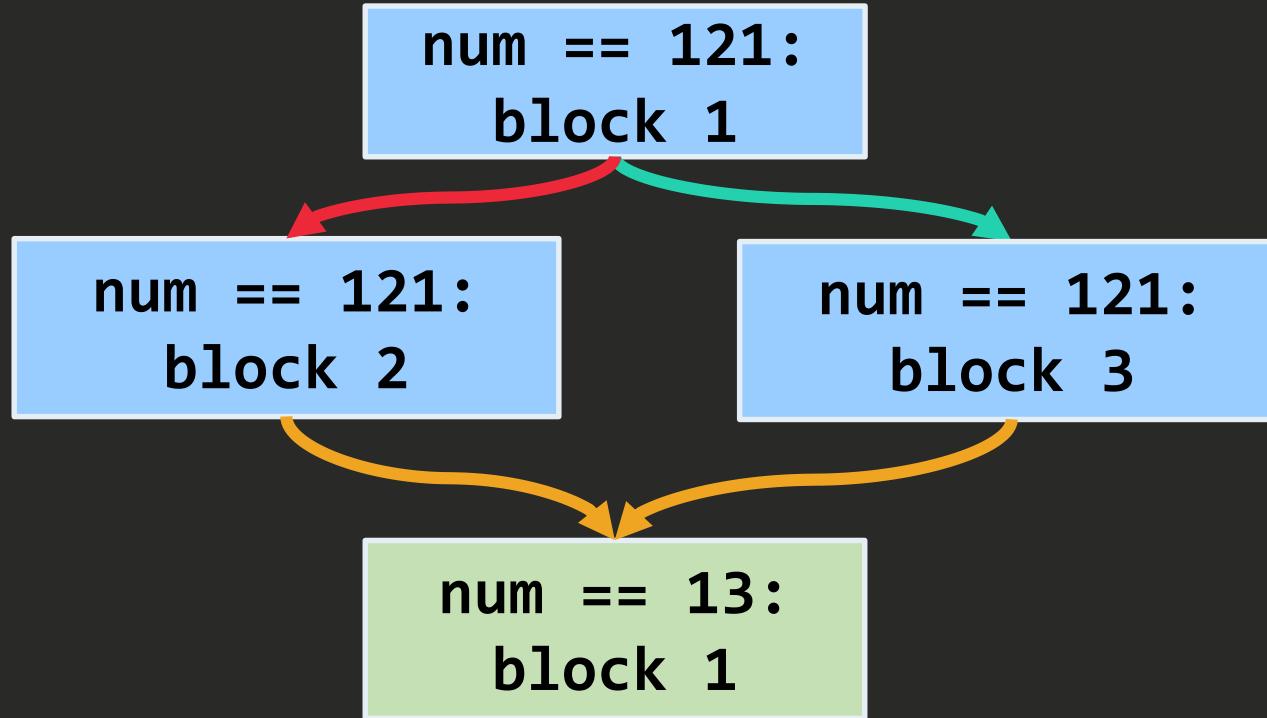
Let's suppose the execution order is {121, 13}

```
num == 13:  
block 1
```



Connect colored blocks to unflatten

Let's suppose the execution order is {121, 13}



Automating the unflattening

- We will use the **de4dot** deobfuscation framework to automate our actions
- We are not just going to launch de4dot, we are going to extend its code
- De4dot's code already implements all functionalities that we need

Downloading de4dot's source code

[de4dot / de4dot](#) Public archive

[Watch](#) 504 [Fork](#) 2.4k [Star](#) 6k

[Code](#) [Pull requests](#) 1 [Actions](#) [Wiki](#) [Security](#) [Insights](#)

[master](#) [1 branch](#) [0 tags](#) [Go to file](#) [Code](#)

wtfscck	Update build.yml	✓ b7d5728 on 29 Aug 2020	2,090 commits
.github/workflows	Update build.yml	2 years ago	
AssemblyData	Add net45 tfm	3 years ago	
AssemblyServer-CLR20-x64	Remove BOM	3 years ago	
AssemblyServer-CLR20	Remove BOM	3 years ago	
AssemblyServer-CLR40-x64	Remove BOM	3 years ago	
AssemblyServer-CLR40	Remove BOM	3 years ago	
AssemblyServer-x64	Remove BOM	3 years ago	
AssemblyServer	Remove BOM	3 years ago	
Test.Rename.Dll	Remove BOM	3 years ago	
Test.Rename	Remove BOM	3 years ago	

About

.NET deobfuscator and unpacker.

[Readme](#) [GPL-3.0 license](#) [6k stars](#) [504 watching](#) [2.4k forks](#)

Used by 13

 + 5

Contributors 20



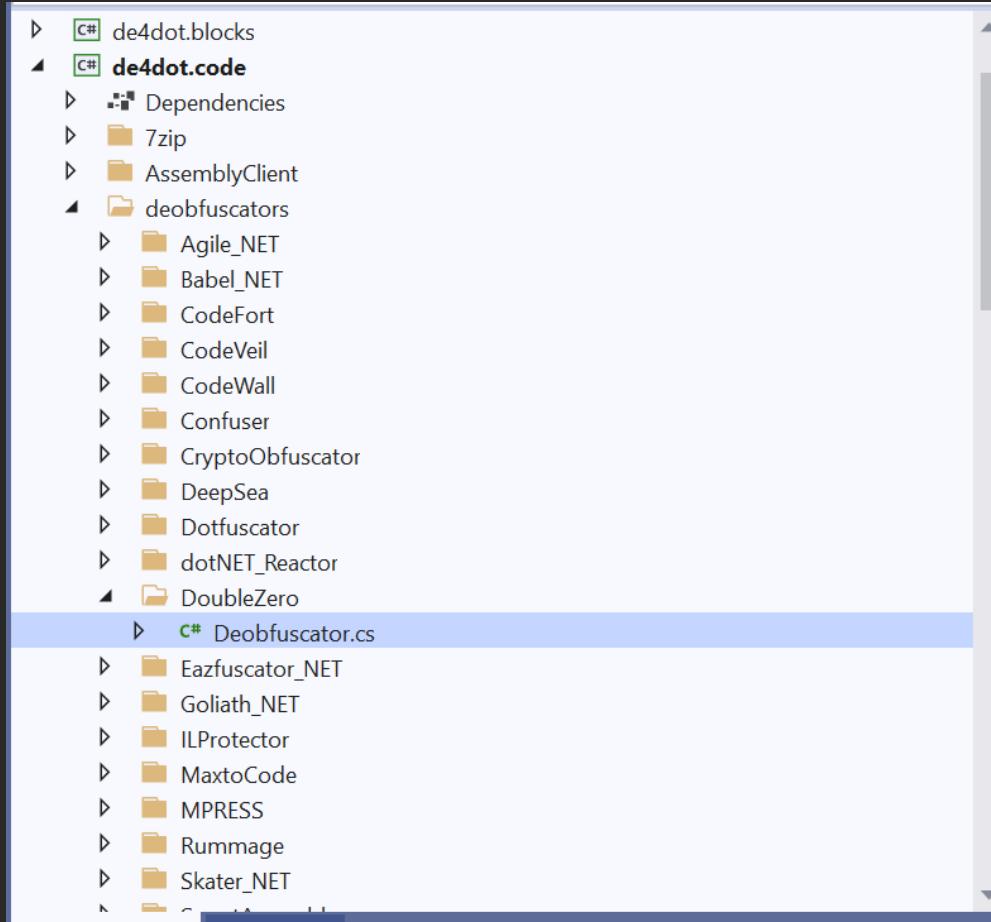
Loading the source code into Visual Studio

The screenshot shows the Visual Studio IDE interface with the following details:

- File Bar:** File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help.
- Search Bar:** Search (Ctrl+Q).
- Solution Explorer:** Shows the solution 'de4dot.netcore' containing 8 projects: AssemblyData, de4dot, de4dot.blocks, de4dot.code (selected), de4dot.cui, de4dot.mdecrypt, Test.Rename, and Test.Rename.DLL.
- Code Editor:** The 'Program.cs' file is open, displaying the following C# code:

```
5  de4dot is free software: you can redistribute it and/or modify
6  it under the terms of the GNU General Public License as published by
7  the Free Software Foundation, either version 3 of the License, or
8  (at your option) any later version.
9
10 de4dot is distributed in the hope that it will be useful,
11 but WITHOUT ANY WARRANTY; without even the implied warranty of
12 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 GNU General Public License for more details.
14
15 You should have received a copy of the GNU General Public License
16 along with de4dot. If not, see <http://www.gnu.org/licenses/>.
17 */
18
19 namespace de4dot_x86 {
20     0 references
21     class Program {
22         0 references
23         static int Main(string[] args) => de4dot.cui.Program.Main(args);
24     }
25 }
```
- Status Bar:** 81%, No issues found, Ln: 22, Ch: 14, Col: 20, MIXED, LF.
- Output Tab:** Shows output from Package Manager: Time Elapsed: 00:00:00.4455238, ====== Finished ======.
- Bottom Navigation:** Error List, Output (selected), Solution Explorer, Git Changes, Add to Source Control, Ready.

Creating a new deobfuscator



Adding template code for the deobfuscator

```
19
20  using System.Collections.Generic;
21  using de4dot.blocks.cflow;
22
23  namespace de4dot.code.deobfuscators.DoubleZero {
24      5 references
24      public class DeobfuscatorInfo : DeobfuscatorInfoBase {
25          public const string THE_NAME = "DoubleZero Obfuscator"; // Obfuscator name
26          public const string THE_TYPE = "dblz"; // Obfuscator short name
27          const string DEFAULT_REGEX = @"(^.*|([a-zA-Z_<{$][a-zA-Z_0-9<>}$].`-]*$)";
28
29      1 reference
29      public DeobfuscatorInfo()
30          : base(DEFAULT_REGEX) {
31      }
32
33      3 references
33      public override string Name => THE_NAME;
34      7 references
34      public override string Type => THE_TYPE;
35
36      3 references
36      public override IDeobfuscator CreateDeobfuscator() =>
37          new Deobfuscator(new Deobfuscator.Options {
37              RenameResourcesInCode = false,
```

Launching de4dot after adding the template

```
--sn-name REGEX  Valid name regex pattern (!^[_a-zA-Z0-9]{1,2}$&^[\u2E80-\u9FFFa-zA-Z_<{$}[\u2E80-\u9FFFa-zA-Z_0-9<>{}$.^-]*$)  
--sn-inline BOOL  Inline short methods (True)  
--sn-remove-inlined BOOL  
                    Remove inlined methods (True)  
--sn-ns1 BOOL     Clear namespace if there's only one class in it (True)  
--sn-rsrc BOOL    Restore resource names (True)
```

Type xc (Xenocode)

```
--xc-name REGEX  Valid name regex pattern (!^[_o001l]{4,}$&!^(get_|set_|add_|remove_|_)?[_x_][a-f0-9]{16,}$&^[\u2E80-\u9FFFa-zA-Z_<{$}[\u2E80-\u9FFFa-zA-Z_0-9<>{}$.^-]*$)
```

Type dblz (DoubleZero Obfuscator)

```
--dblz-name REGEX  
                    Valid name regex pattern ((^.*|(^[_a-zA-Z_<{$}][_a-zA-Z_0-9<>{}$.^-]*$))
```

String decrypter types

none	Don't decrypt strings
default	Use default string decrypter type (usually static)
static	Use static string decrypter if available
delegate	Use a delegate to call the real string decrypter
emulate	Call real string decrypter and emulate certain instructions

Multiple regexes can be used if separated by '&'

Adding a block deobfuscator

A block deobfuscator allows to traverse and modify control flow graphs

```
public interface IBlocksDeobfuscator {  
    bool ExecuteIfNotModified { get; }  
    void DeobfuscateBegin(Blocks blocks);  
    bool Deobfuscate(List<Block> allBlocks);  
}
```

Finding the execution order array

```
object[] array = new object[]  
{  
    new int[]  
    {90,2089875171,90,1318285745,10,90,886806518,10,180},  
    ...  
};
```

Finding the execution order array

```
object[] array = new object[]  
{  
    ...  
};
```

```
ldc.i4.s <number of members in the object [] array>  
newarr [mscorlib]System.Object
```

Finding the execution order array

```
object[] array = new object[]  
{  
    ...  
};
```

```
ldc.i4.s <number of members in the object [] array>  
newarr [mscorlib]System.Object
```

```
instr.OpCode == OpCodes.Newarr &&  
instr.Operand.ToString() == "System.Object"
```

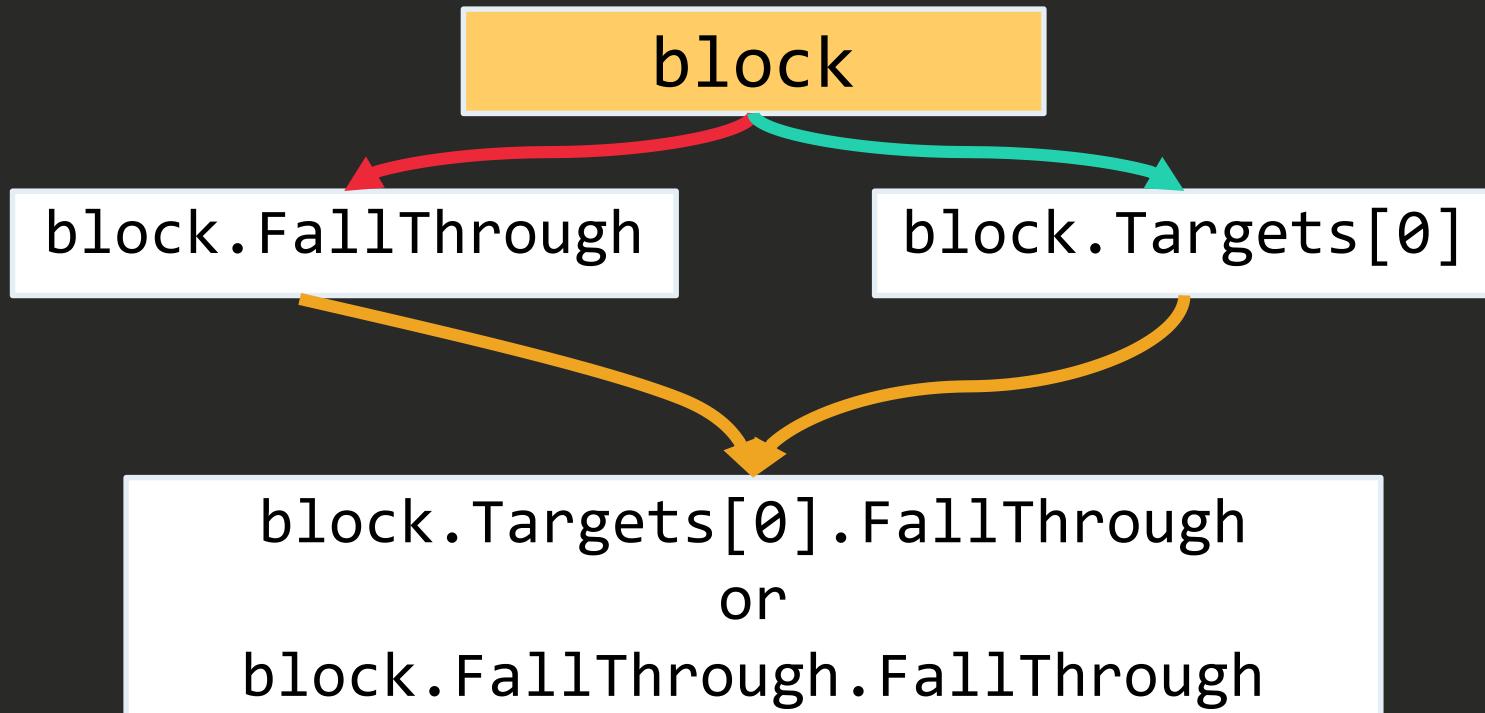
Extracting integer arrays from the array

```
new int[] {  
90,2089875171,90,1318285745,10,90,886806518,10,180  
}
```

```
ldtoken field valuetype <int [] array value>  
call InitializeArray  
stelem.ref
```

```
byte [] arr = instruction.Operand.InitialValue;
```

Traversing the control flow graph



Reconnecting blocks

You can perform reconnection in a single line of code!

Connection through an unconditional jump:

```
sourceBlock.SetNewFallthrough(targetBlock);
```

Through a conditional jump (true branch):

```
sourceBlock.SetNewTarget(i, targetBlock);
```

Conclusions

- The discussed unflattening strategy can be applied to any programming language
- It has turned out to be extremely convenient to implement it with de4dot
- Surprisingly, the unflattener code is only about 250 lines long!

Thank you!

The deobfuscator code is available at
<https://github.com/gkucherin/de4dot>

Please feel free to ask me questions!

Email: georgy.kucherin@gmail.com
Keybase: [gkucherin](#)