VB 2022

Combating control flow flattening in .NET malware

Georgy Kucherin
Kaspersky GReAT
Junior Security Researcher
Keybase: gkucherin

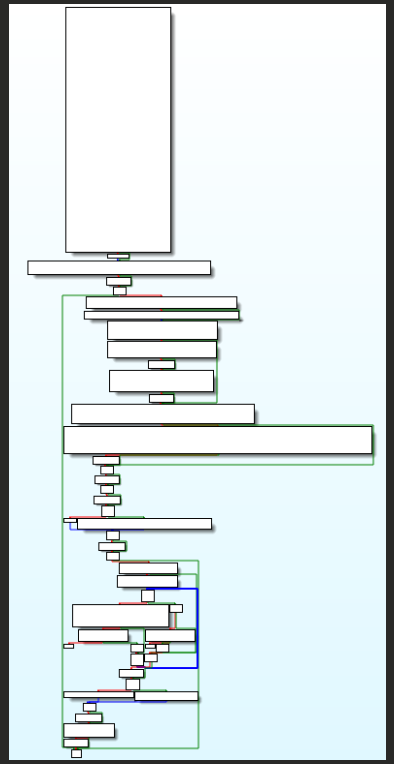
kaspersky

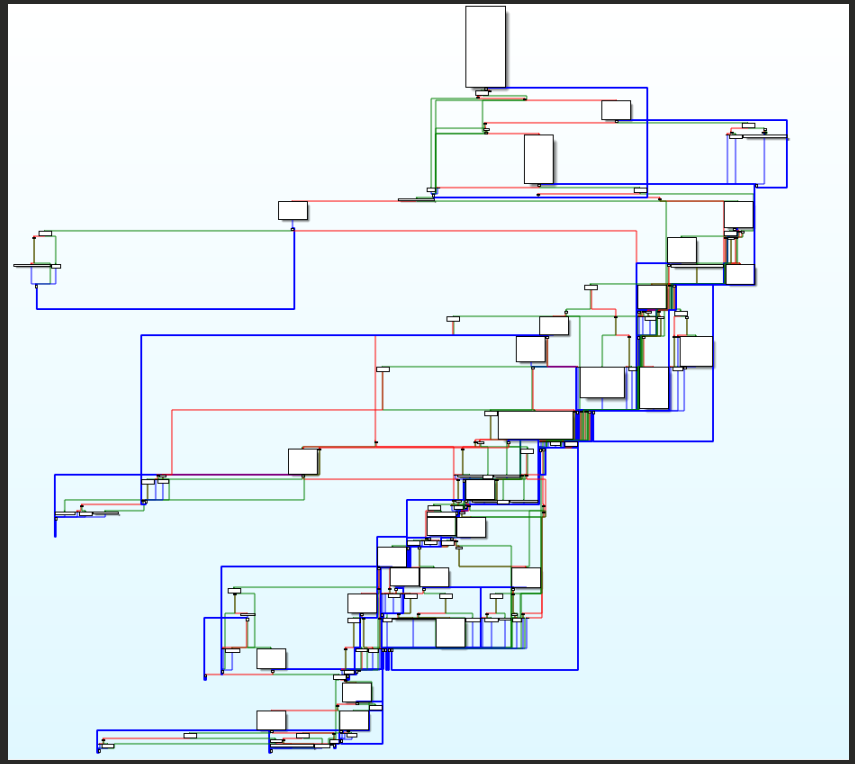
Obfuscation is widespread in malware

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

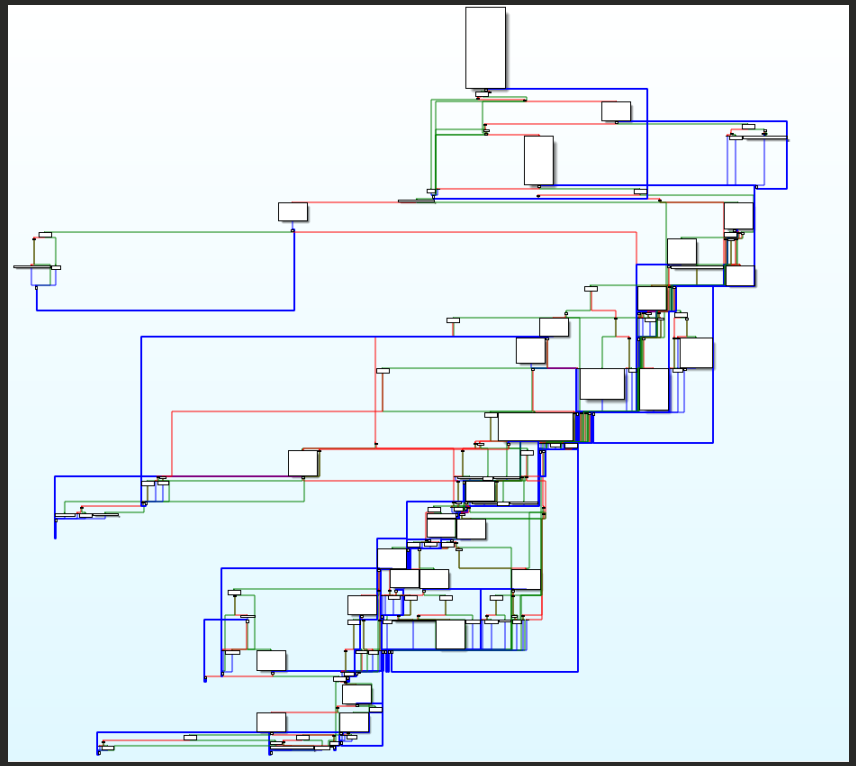
Control flow flattening technique

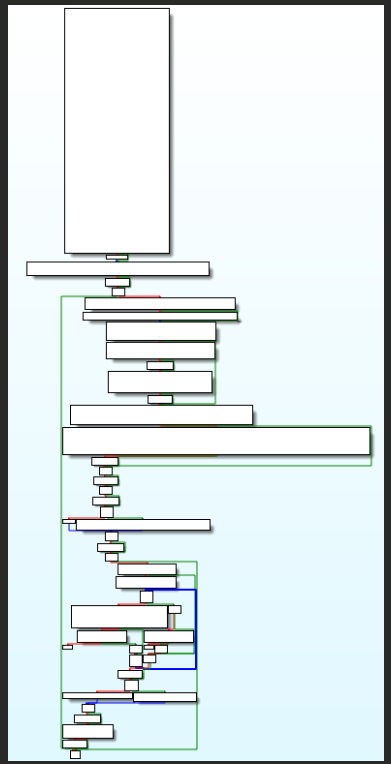
Is considered to be one of the hardest-to-crack techniques

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!

```
if (num2 <= 542)
{
    if (num2 != 219)
    {
        if (num2 != 494)
        {
            if (num2 == 542)
            {
                _9f6717951b3535fb甸橵ि嘍썩曆瓶纇릙엥숬拐?抹?臬?椦誥?茊浏릝蠶経헬렛쭁갿츎昑.RtlAdjustPrivilege(9UL, true, false, ref flag);
            }
        }
        else
        {
            FieldDirection fieldDirection = (FieldDirection)obj2;
        }
    }
}
else if (num2 != 544)
{
    if (num2 != 559)
    {
        if (num2 == 581)
        {
            _9f6717951b3535fb甸橵ि嘍썩曆瓶纇릙엥숬拐?抹?臬?椦誥?茊浏릝蠶経헬렛쭁갿츎昑.RtlAdjustPrivilege(18UL, true, false, ref flag);
        }
    }
    else
    {
        _9f6717951b3535fb甸橵ि嘍썩曆瓶纇릙엥숬拐?抹?臬?椦誥?茊浏릝蠶経헬렛쭁갿츎昑.RtlAdjustPrivilege(17UL, true, false, ref flag);
    }
}
else
{
    _9f6717951b3535fb甸橵ि嘍썩曆瓶纇릙엥숬拐?抹?臬?椦誥?茊浏릝蠶経헬렛쭁갿츎昑.RtlAdjustPrivilege(19UL, true, false, ref flag);
}
j++;
}
goto IL_2A1;
```

4 lines turned into 317 lines
Code is hard to read

```
object[] array = new object[]
{
    new int[] {90,2089875171,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875154,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875047,90,1318285745,10,90,886806518,10,180}
};
while (i < 5)
{
    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;
}
```

Taking a close look at the obfuscation pattern

```
object[] array = new object[]
{
    new int[] {90,2089875171,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875154,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875047,90,1318285745,10,90,886806518,10,180}
};
while (i < 5)
{
    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;
}
```

Initialization of an array of integer arrays

```
object[] array = new object[]
{
    new int[] {90,2089875171,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875154,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875047,90,1318285745,10,90,886806518,10,180}
};
while (i < 5)
{
    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;
}
```

Loop over the array of integer arrays

```
object[] array = new object[]
{
    new int[] {90,2089875171,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875154,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875047,90,1318285745,10,90,886806518,10,180}
};
while (i < 5)
{
    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;
}
```

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

```
object[] array = new object[]
{
    new int[] {90,2089875171,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721120,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,-986721052,90,1218371032,20,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875154,90,1318285745,10,90,886806518,10,180},
    new int[] {90,2089875047,90,1318285745,10,90,886806518,10,180}
};
while (i < 5)
{
    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;
}
```

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)

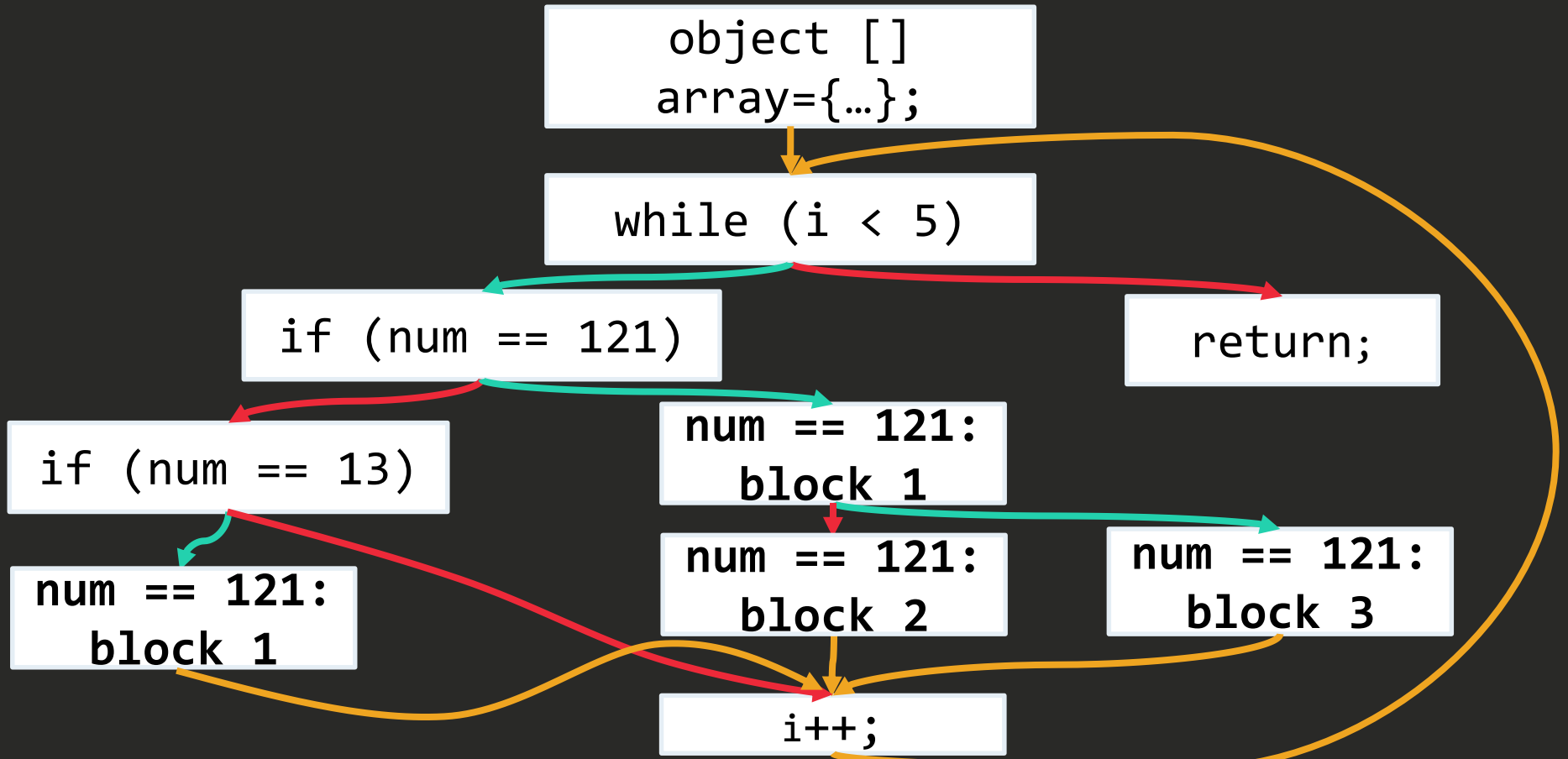
Extracting lines

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

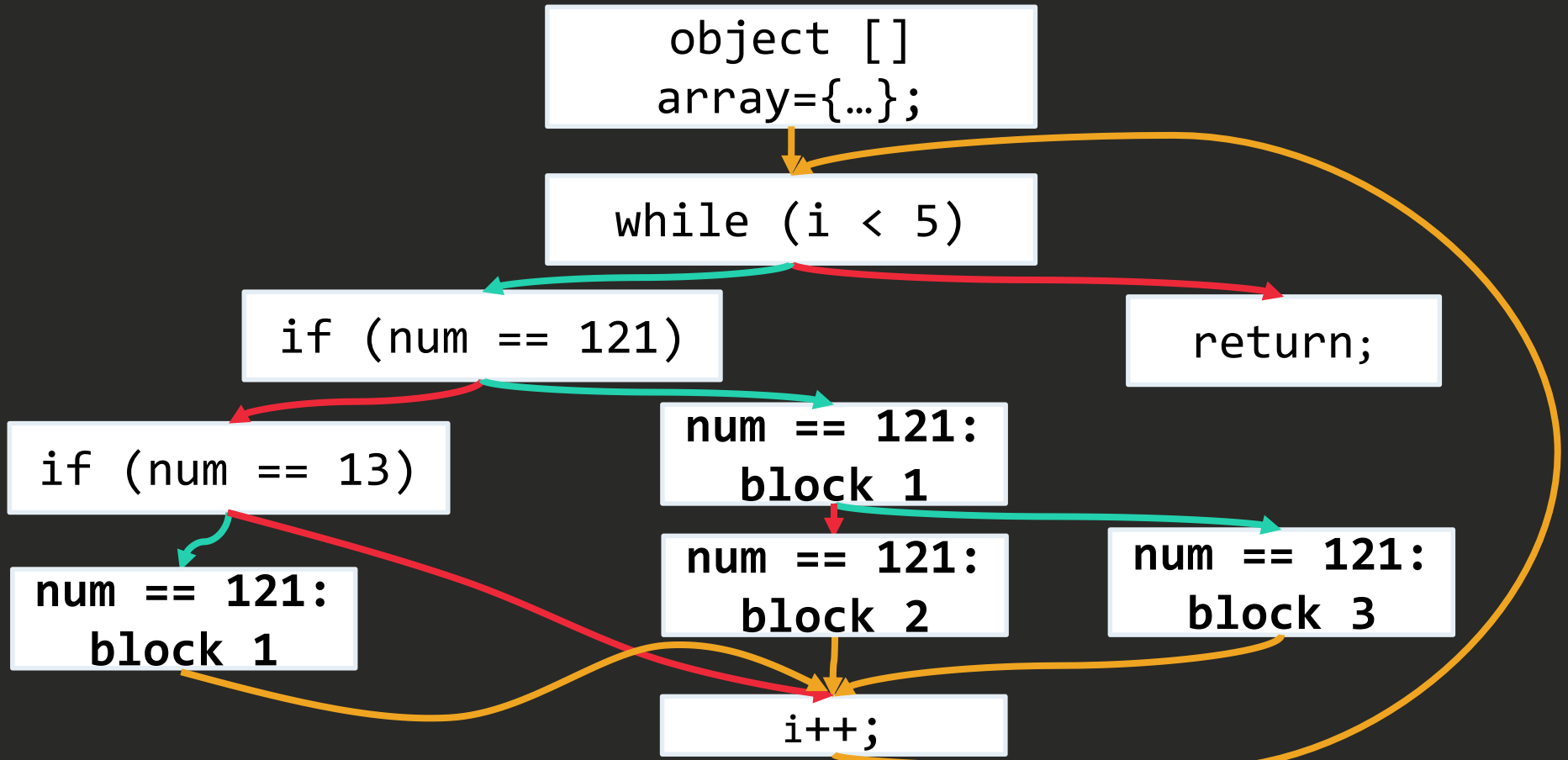
```
int num = X1BFovEdxME9D.HBkAc6Pu((int[])array[i], 0, 0);
if (num == 13)
{
    goto IL_29A;
}
if (num == 121)
{
    <code omitted>
}
IL_2A1:
i++;
continue;
IL_29A:
RequestCachingSection requestCachingSection = (RequestCachingSection)obj;
goto IL_2A1;
```

Handling goto statements would be problematic

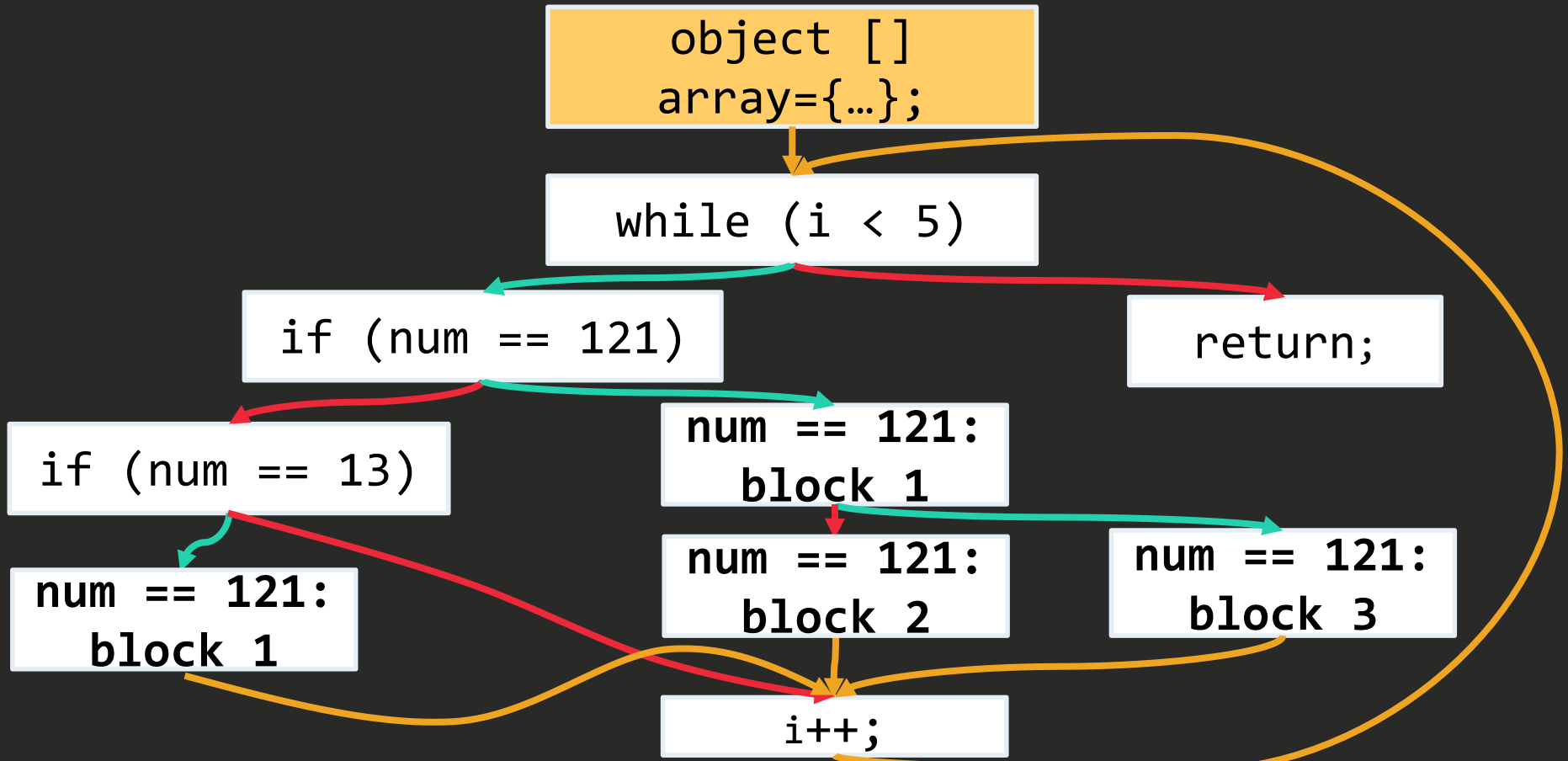
A better way of extracting code



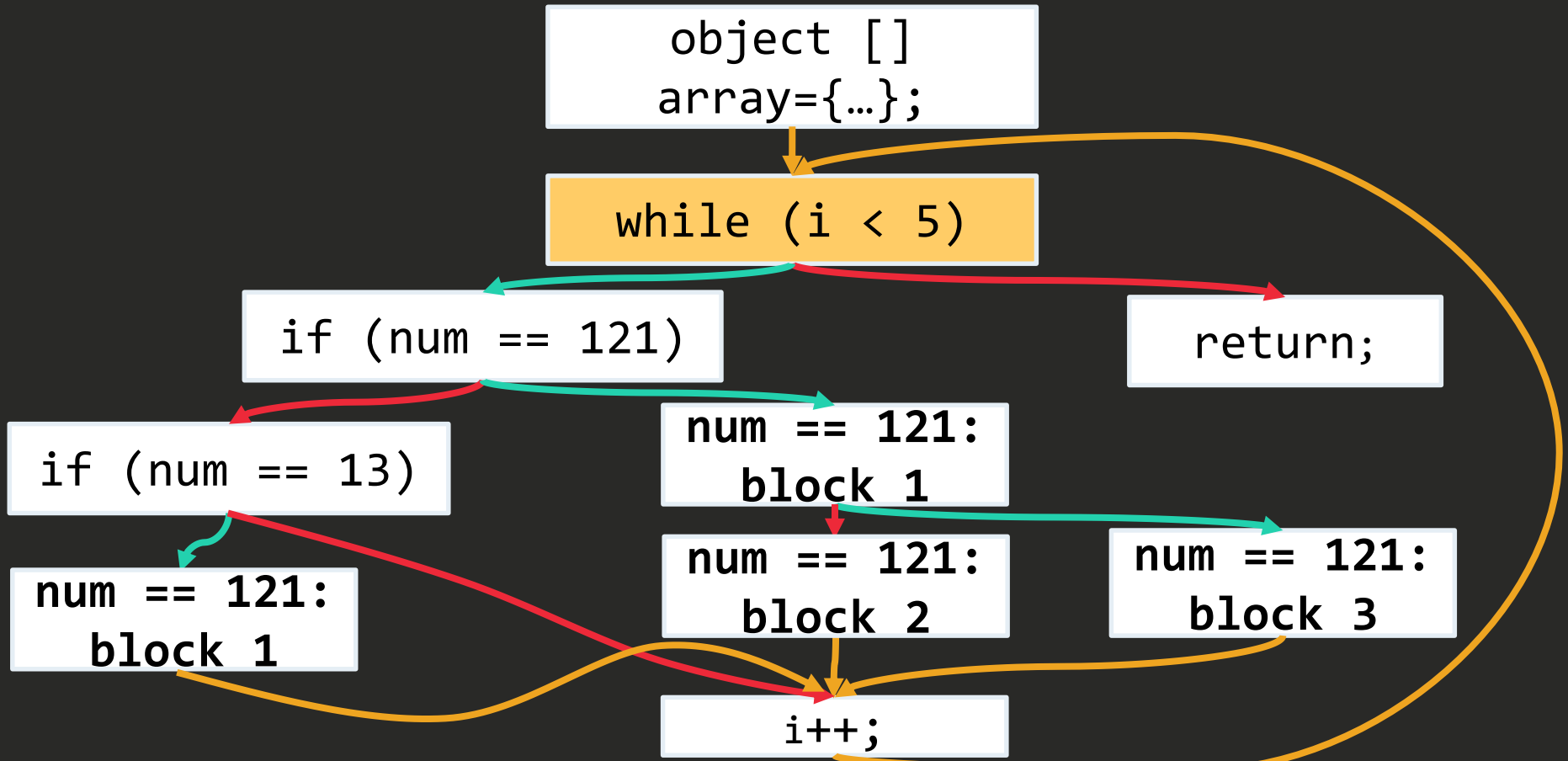
Goal: mark all blocks with bold text



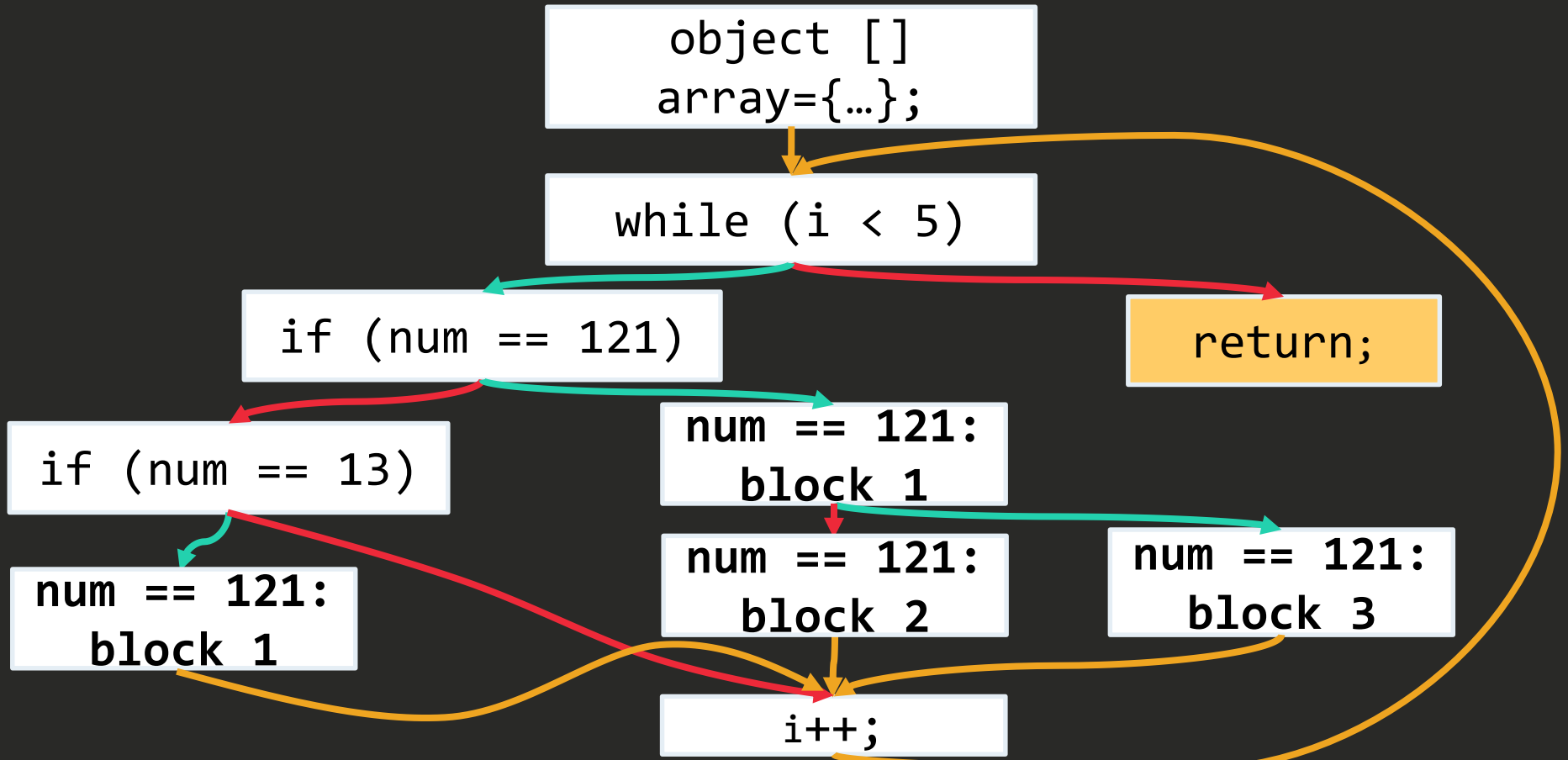
Traverse the graph until we reach an if



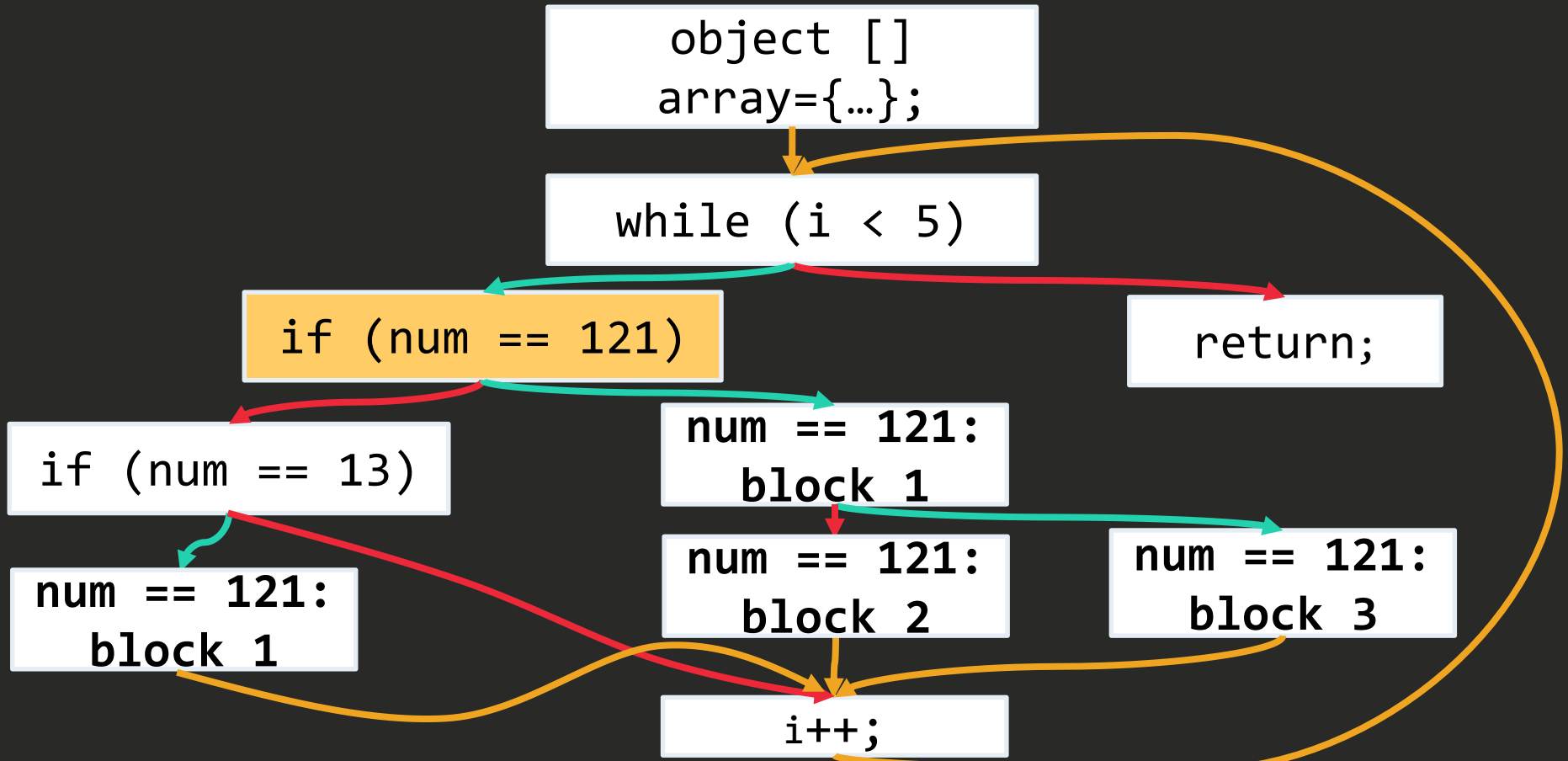
Traverse the graph until we reach an if



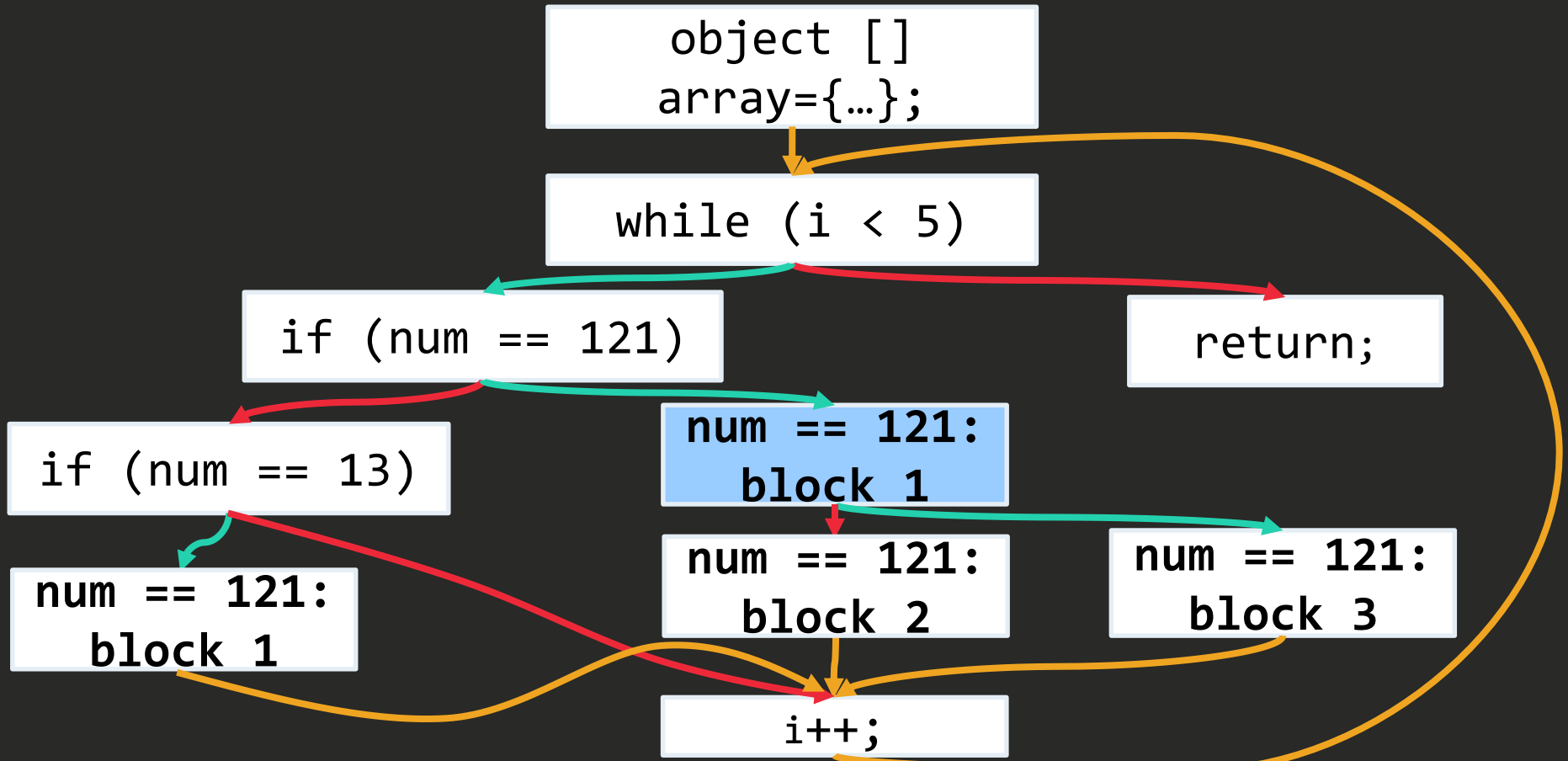
Traverse the graph until we reach an if



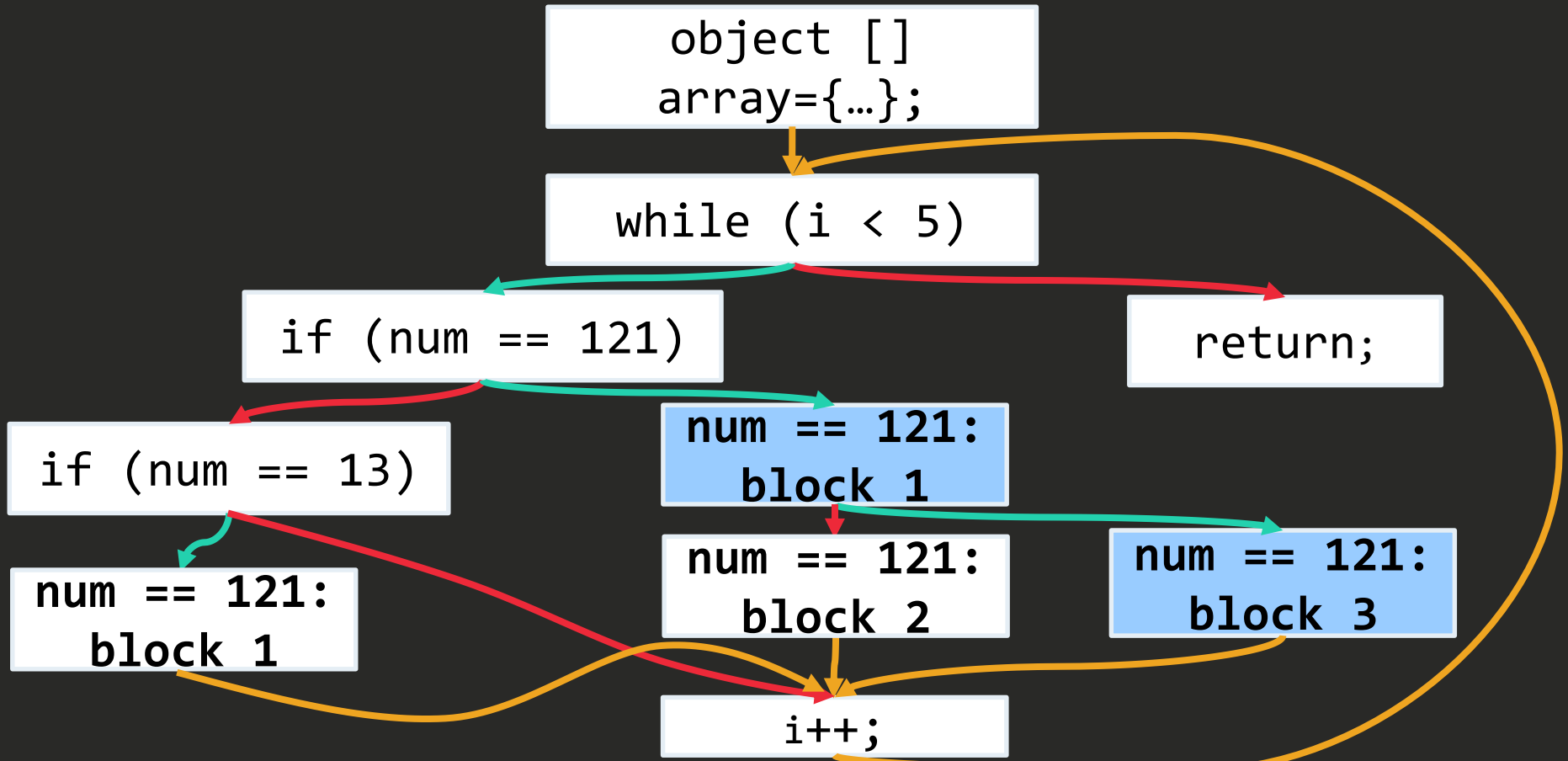
We have reached an if statement



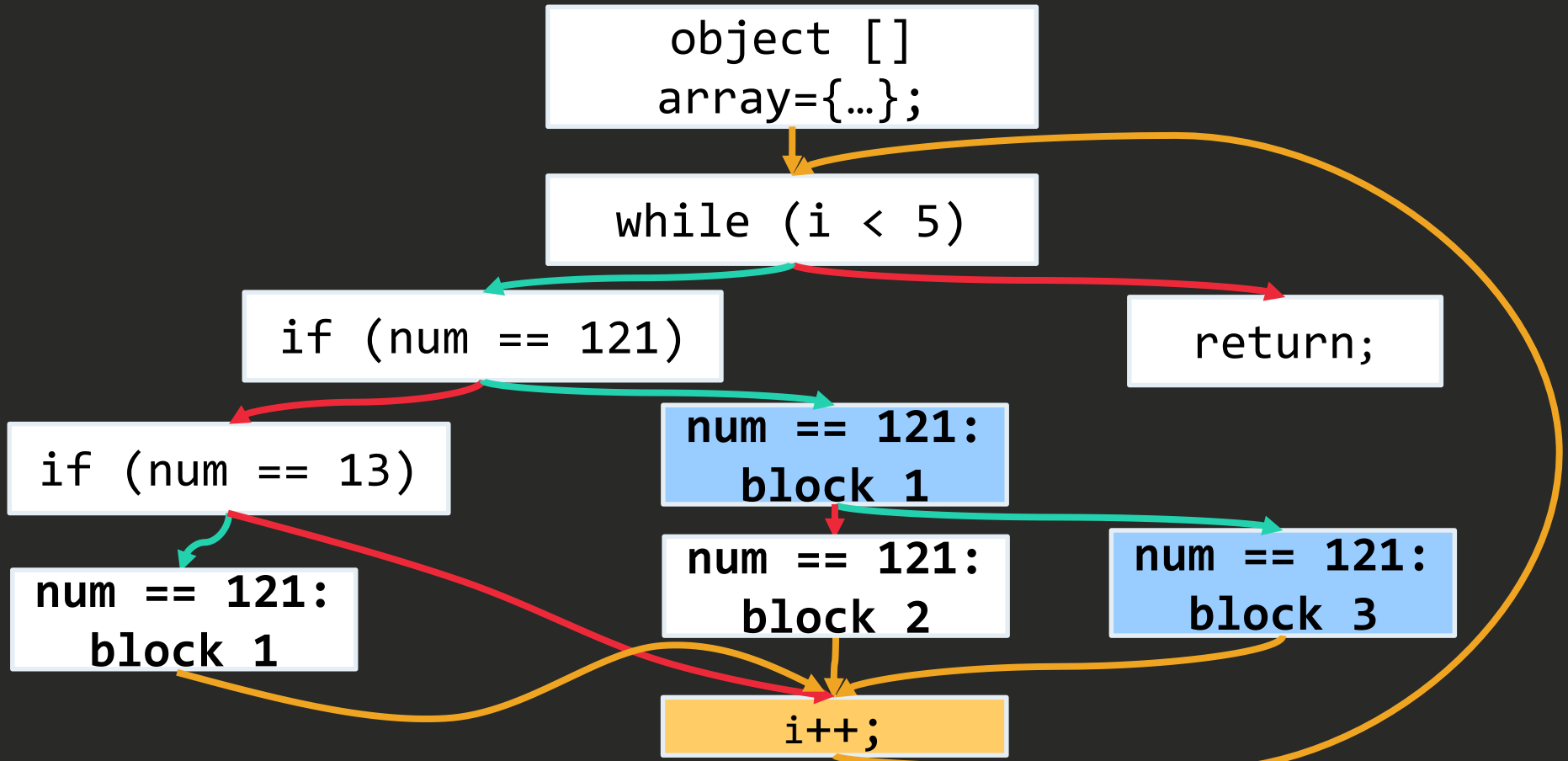
Mark all blocks branching out of the if



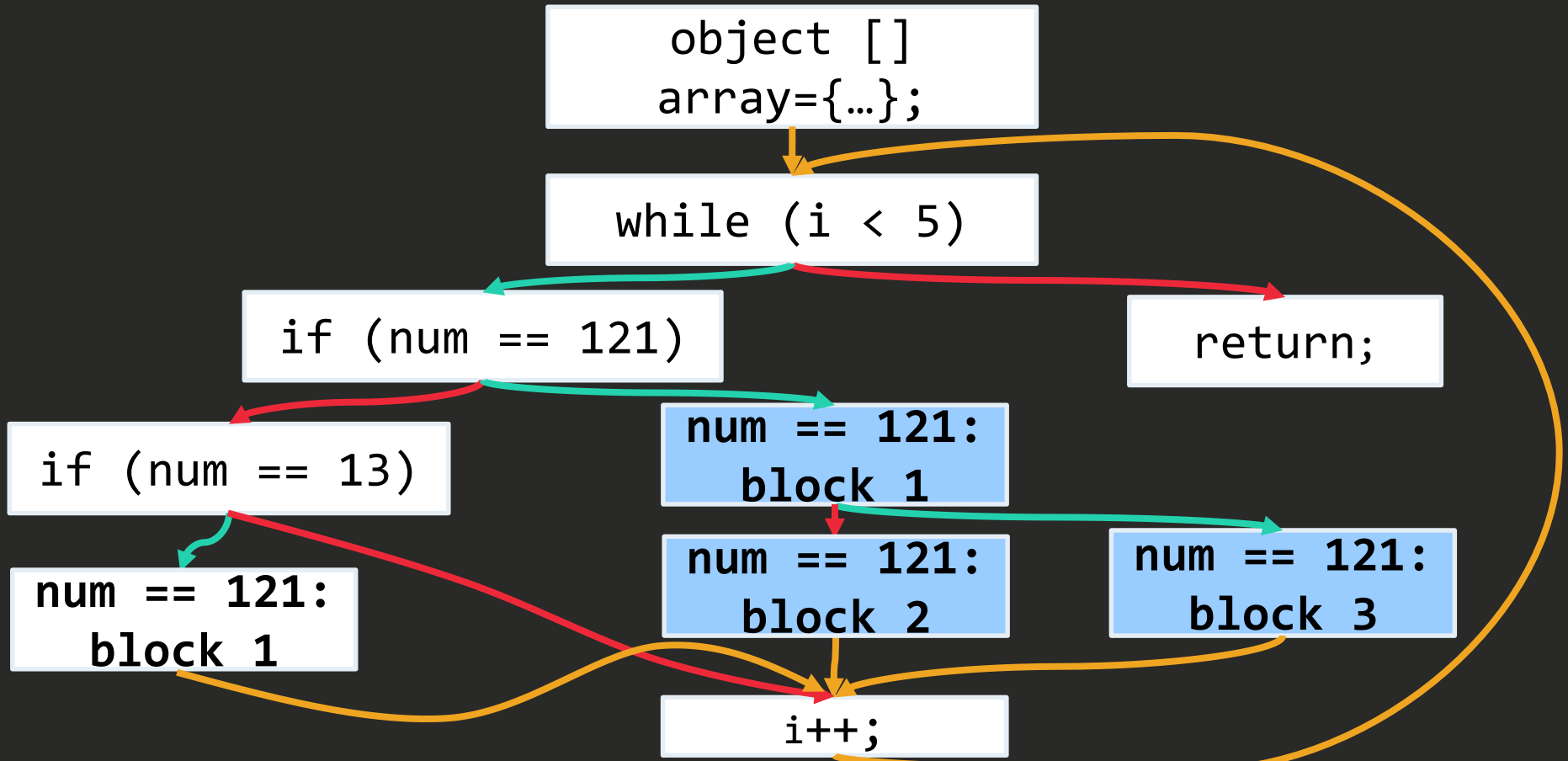
Mark all blocks branching out of the if



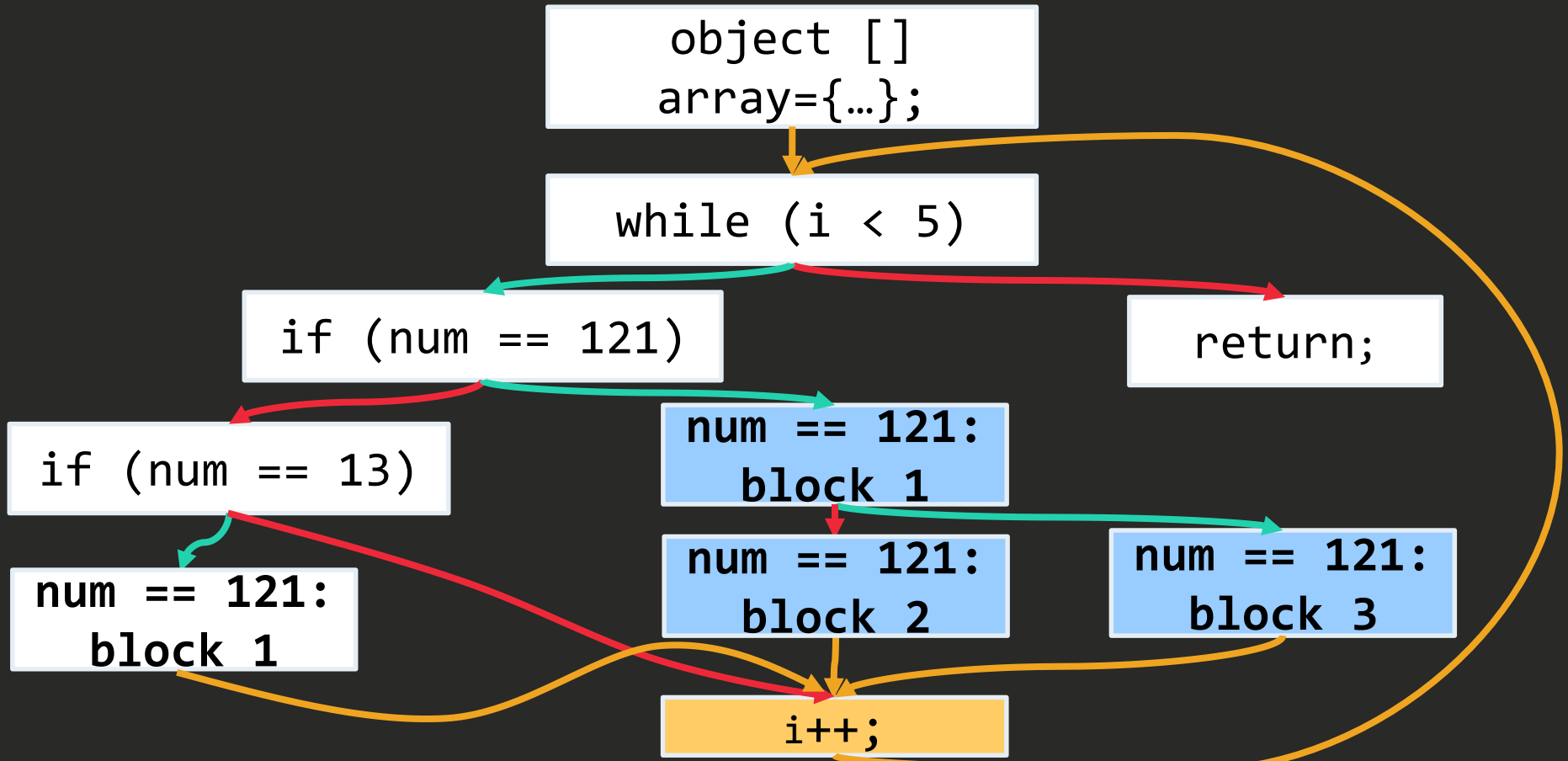
Stop when we reach the i++ block



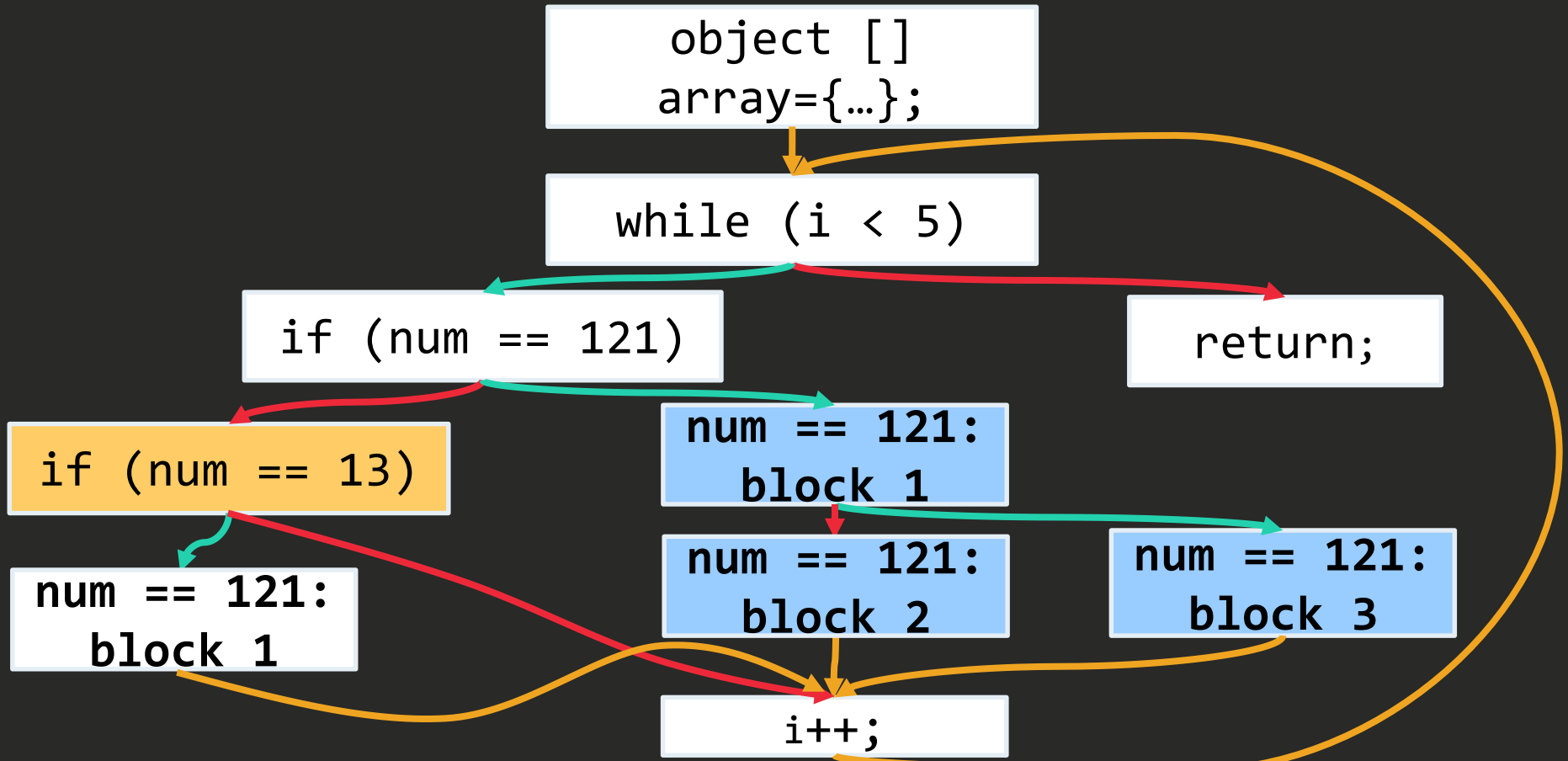
Mark all blocks branching out of the if



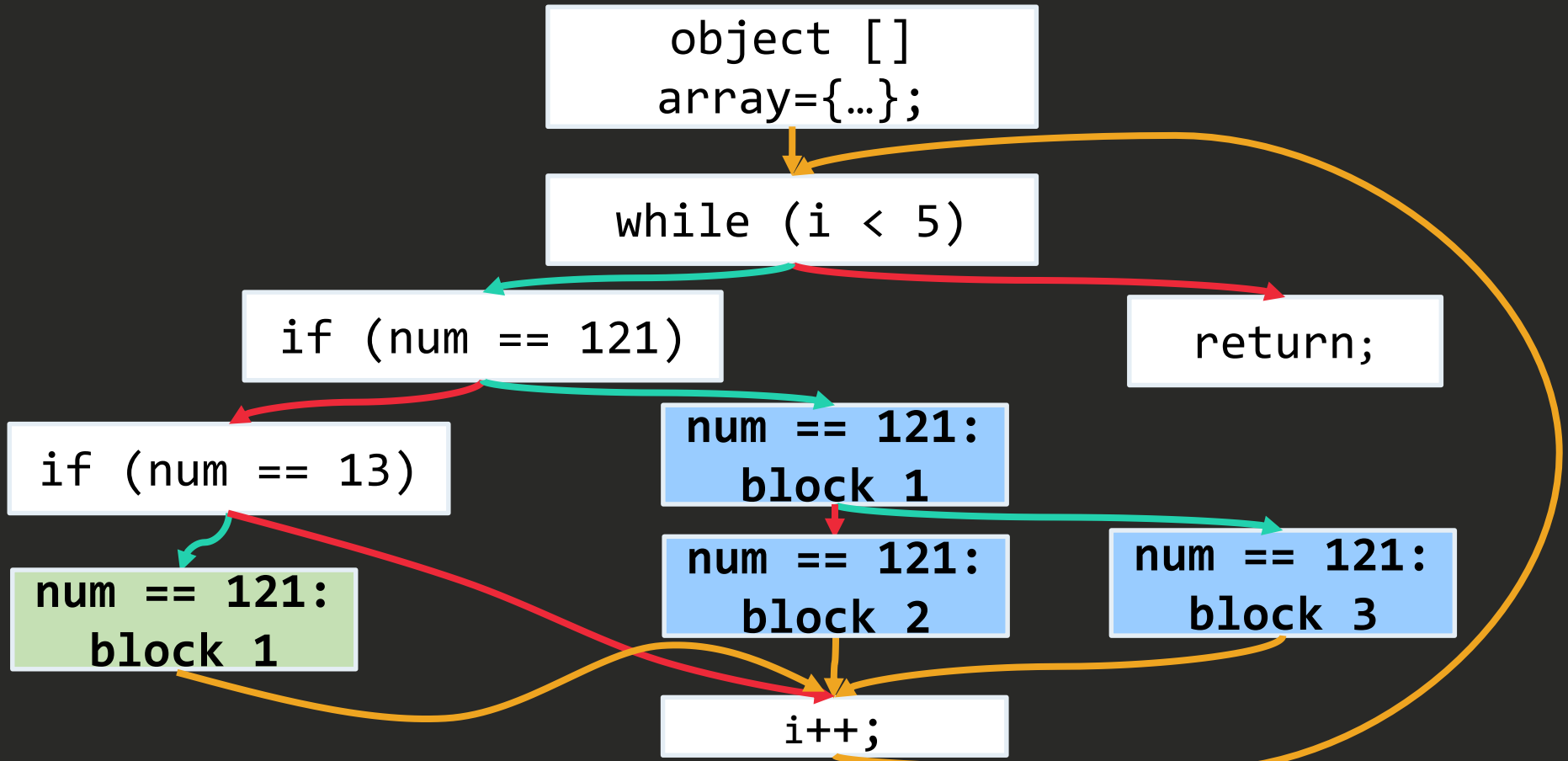
Stop when we reach the i++ block



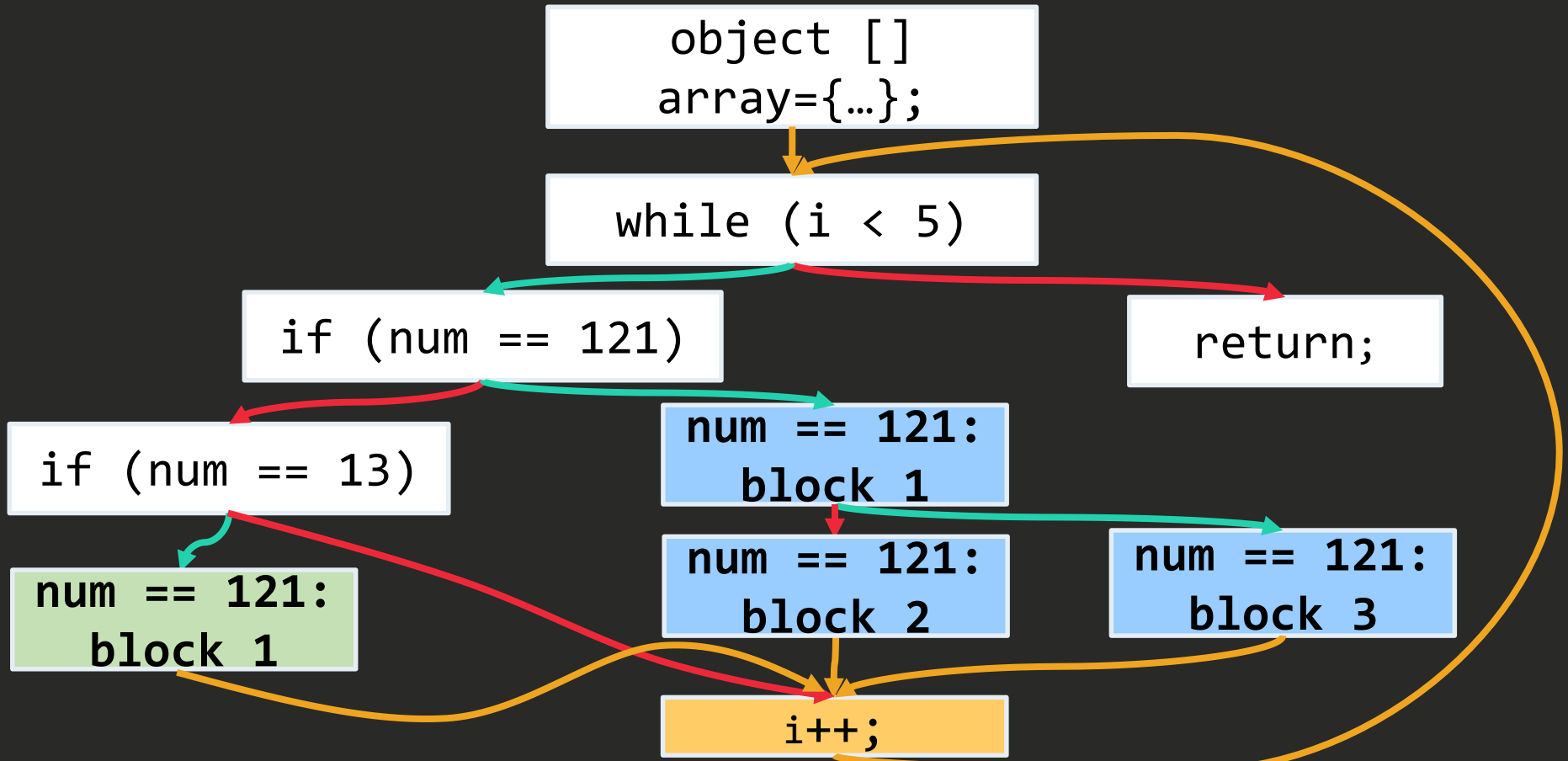
The same for num == 13



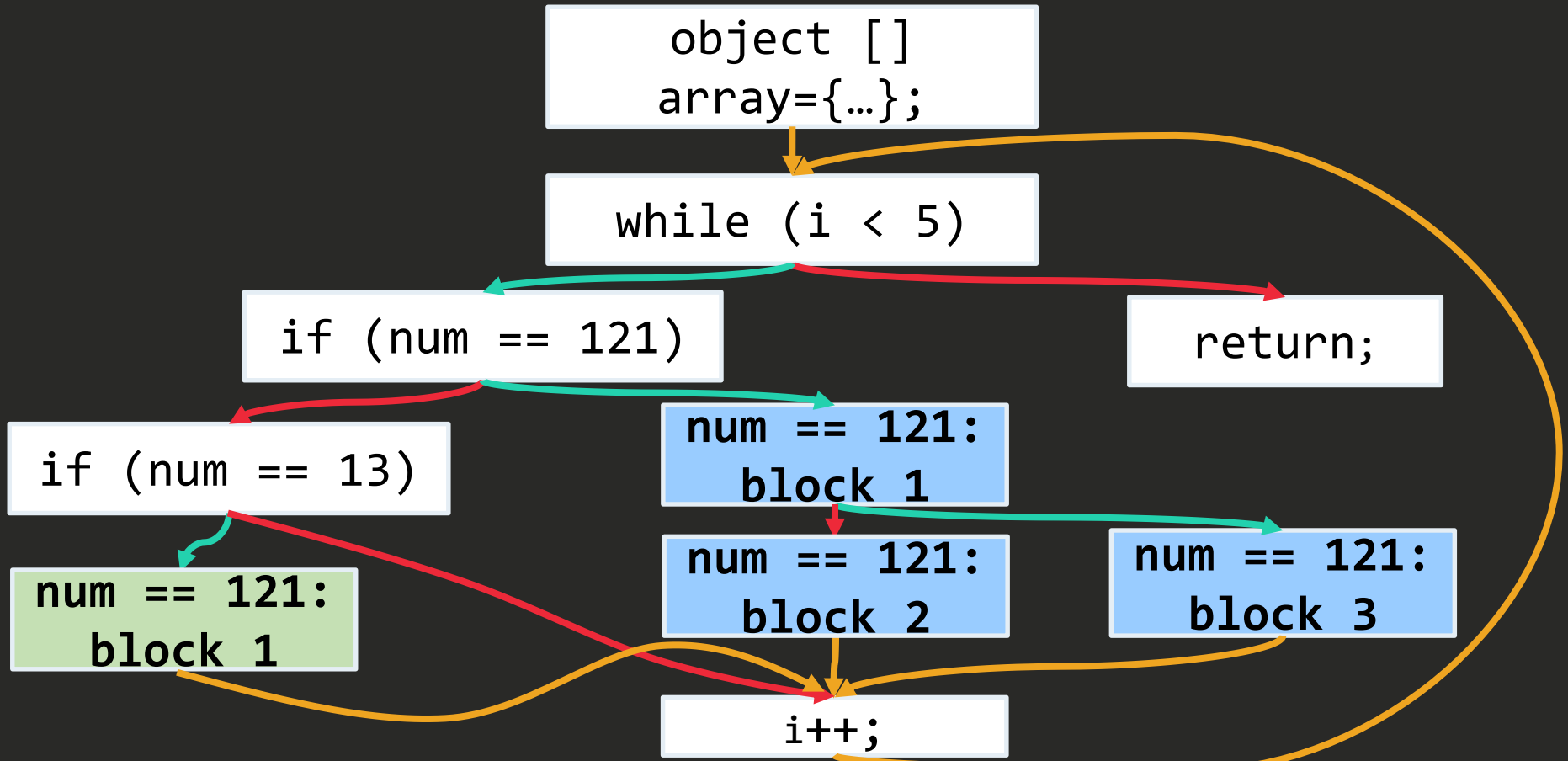
The same for num == 13



The same for num == 13

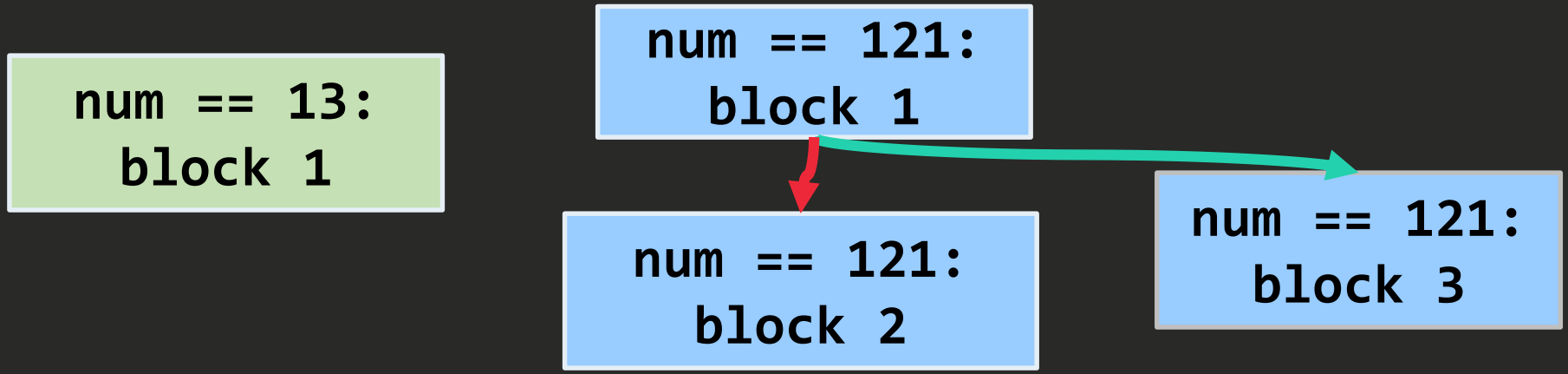


Done!


object []
array={…};
while (i < 5)
if (num == 121)
return;
if (num == 13)
num == 121:
block 1
num == 121:
block 1
num == 121:
block 2
num == 121:
block 3
i++;

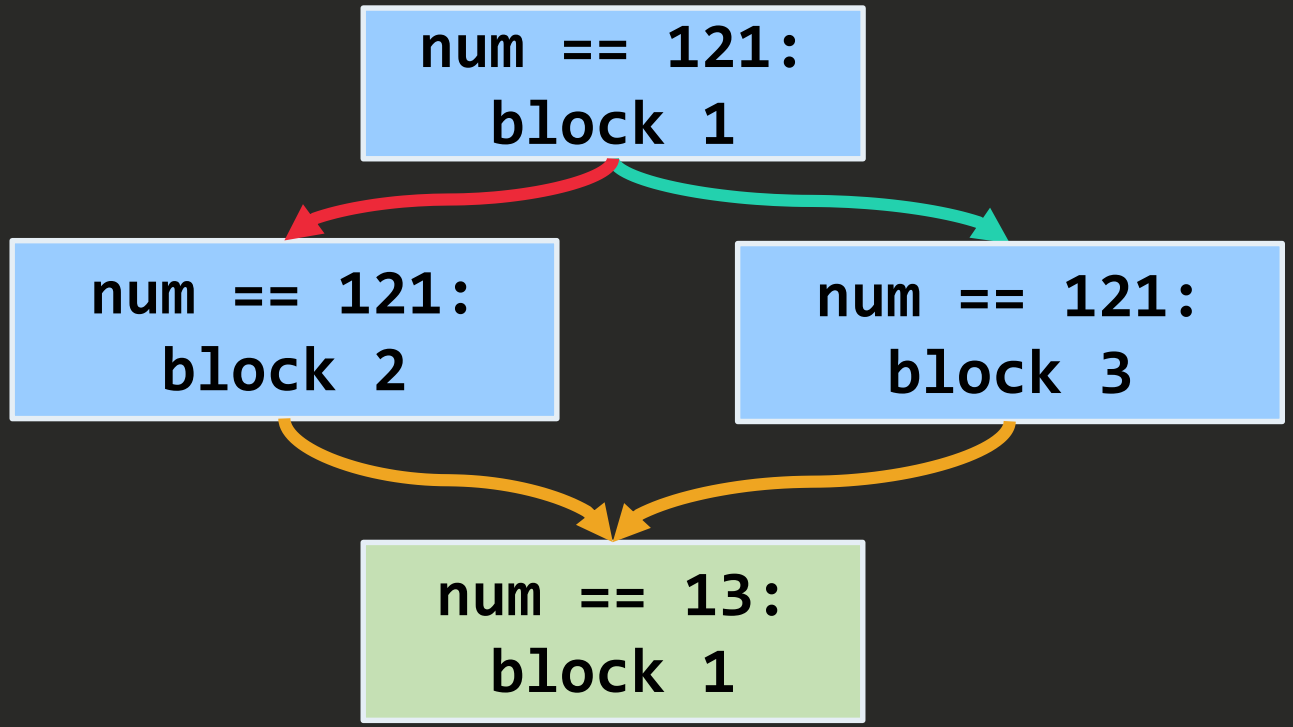
Connect colored blocks to unflatten

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



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

wtfsck 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.

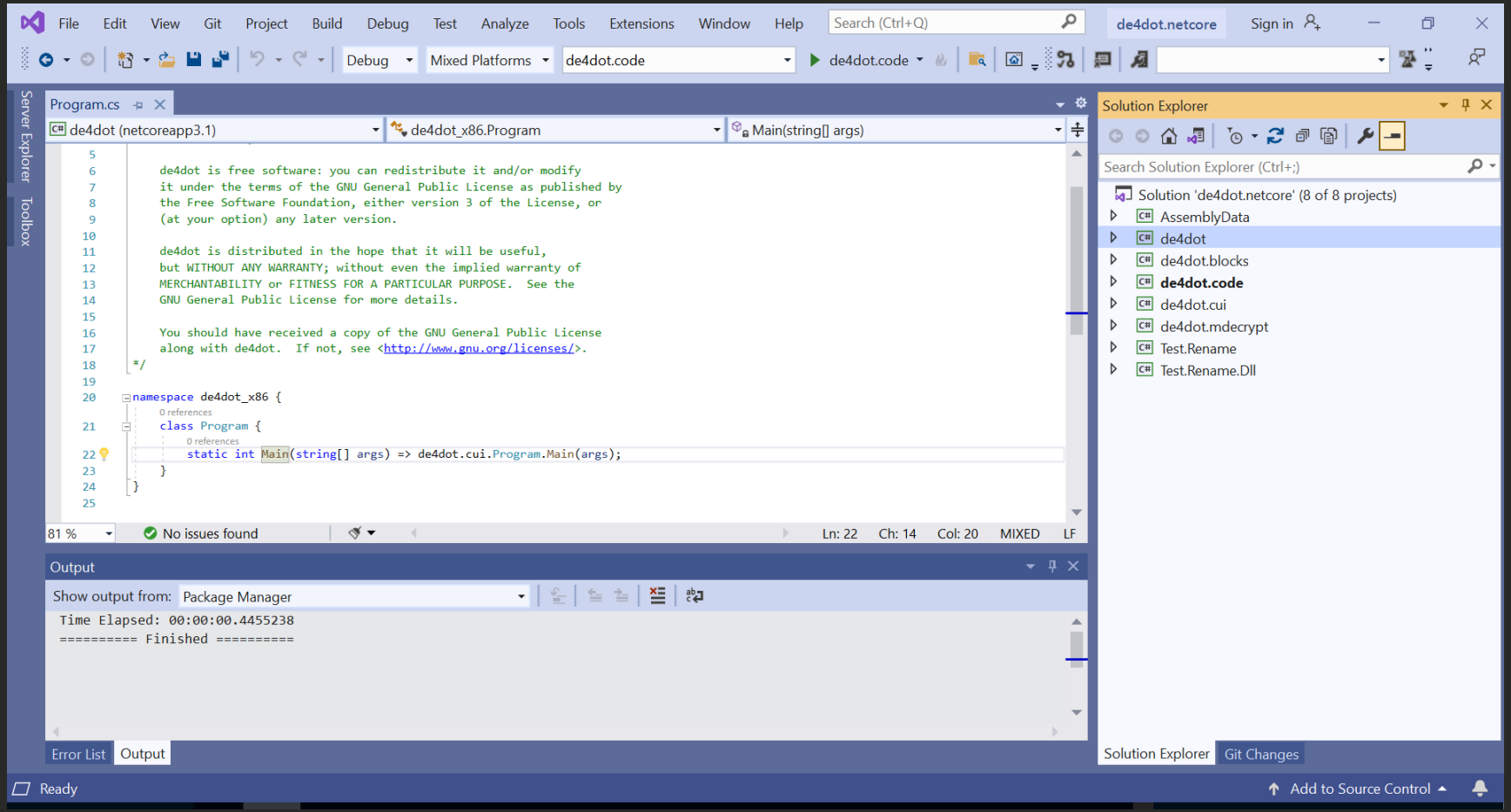
Readme
GPL-3.0 license
6k stars
504 watching
2.4k forks

Used by 13

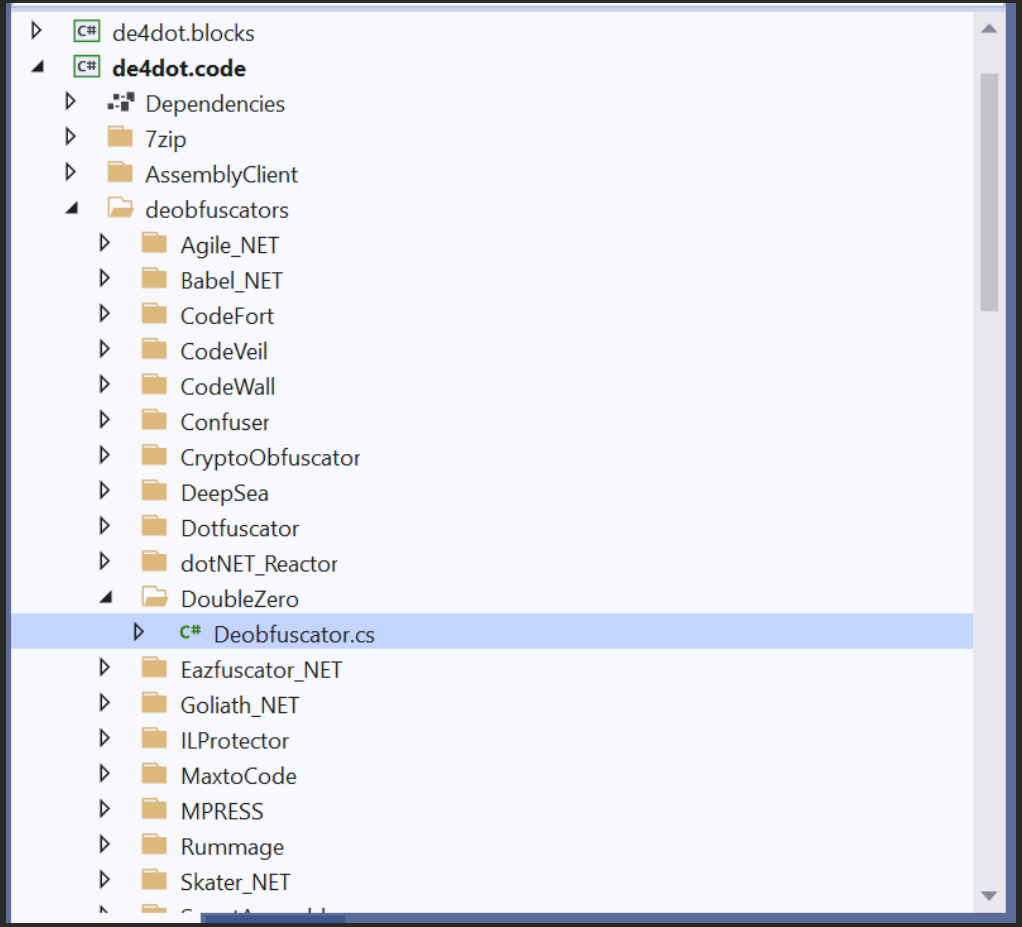
+ 5

Contributors 20

Loading the source code into Visual Studio



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 {
        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
              1 reference
29          public DeobfuscatorInfo()
30              : base(DEFAULT_REGEX) {
31          }
32
              3 references
33          public override string Name => THE_NAME;
              7 references
34          public override string Type => THE_TYPE;
35
              3 references
36          public override IDeobfuscator CreateDeobfuscator() =>
37              new Deobfuscator(new Deobfuscator.Options {
38                  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 (!^[oO01l]{4,}$&!^(get_|set_|add_|remove_|_)?[x_][a-f0-9]{16,}$&^[\u2E80-\u9
FFFa-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"
```

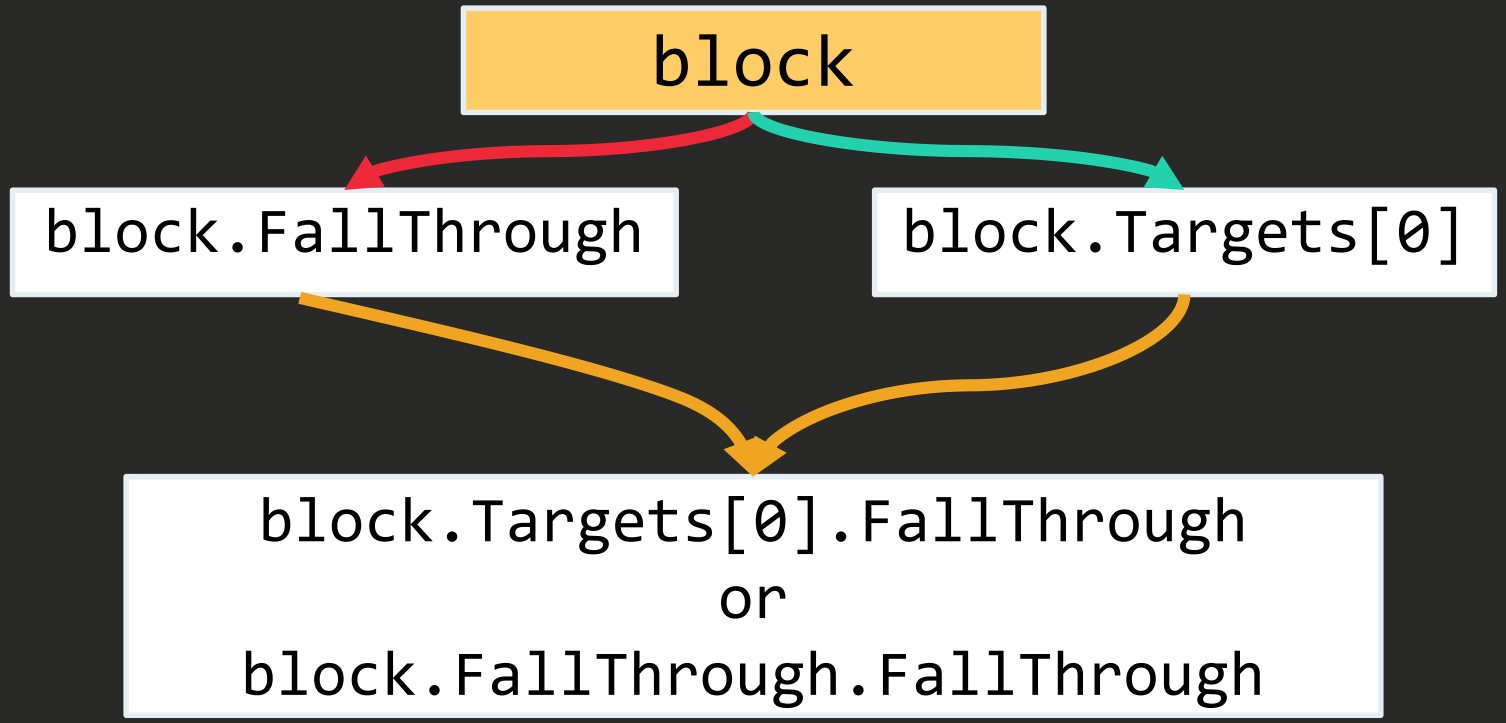
Extracting integer arrays from the array

```
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