EKO-PARTY 2009

Binary Diffing

by Nicolás A. Economou

What is binary diffing ?

- Compare byte a byte 2 binary files

- Binary File Examples:
  - .exe 

  - .wav
  - .ppt
  - .jpg

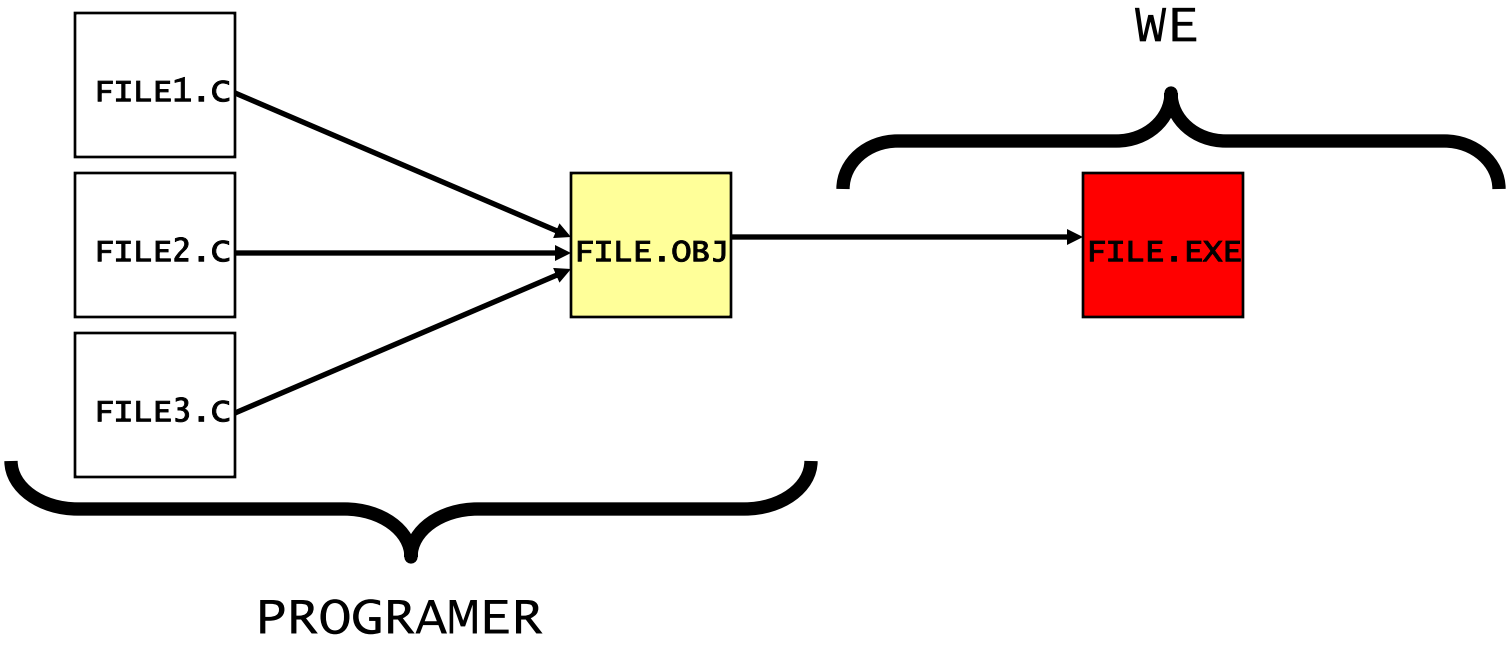
What is executable binary diffing ?

- Compare functions between 2 executable binary files.

- It's necessary to use heuristics (compare byte to byte doesn't work !)

Why we need to compare binary files ?

– Because we don't have the source code ☹



What is the use of that ?

- **Looking for coincidences**:
  - Code Theft Detection
- **Looking for differences**:
  - Security Patch Detection
  - Added/Removed Code Detection
- **Misc**:
  - Virus/Worms Mutation Detection, Firmware Updates

“C” Code Example

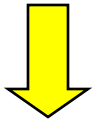
```
void check_arguments ( int argc , char *argv [] )
{
  if ( argc == 3 )
  {
    printf ( "arguments ok\n" );
  }
}
```

“asm x86” Code Example

```
.text:00401015 check_arguments proc near               ; CODE XREF: _main+9↑p
.text:00401015
.text:00401015 arg_0           = dword ptr  8
.text:00401015
.text:00401015                 push    ebp
.text:00401016                 mov     ebp, esp
.text:00401018                 cmp     [ebp+arg_0], 3
.text:0040101C                 jnz     short loc_401029
.text:0040101E                 push    offset format   ; "arguments ok\n"
.text:00401023                 call    _printf
.text:00401028                 pop     ecx
.text:00401029
.text:00401029 loc_401029:                             ; CODE XREF: check_arguments+7↑j
.text:00401029                 pop     ebp
.text:0040102A                 retn
.text:0040102A check_arguments endp
```

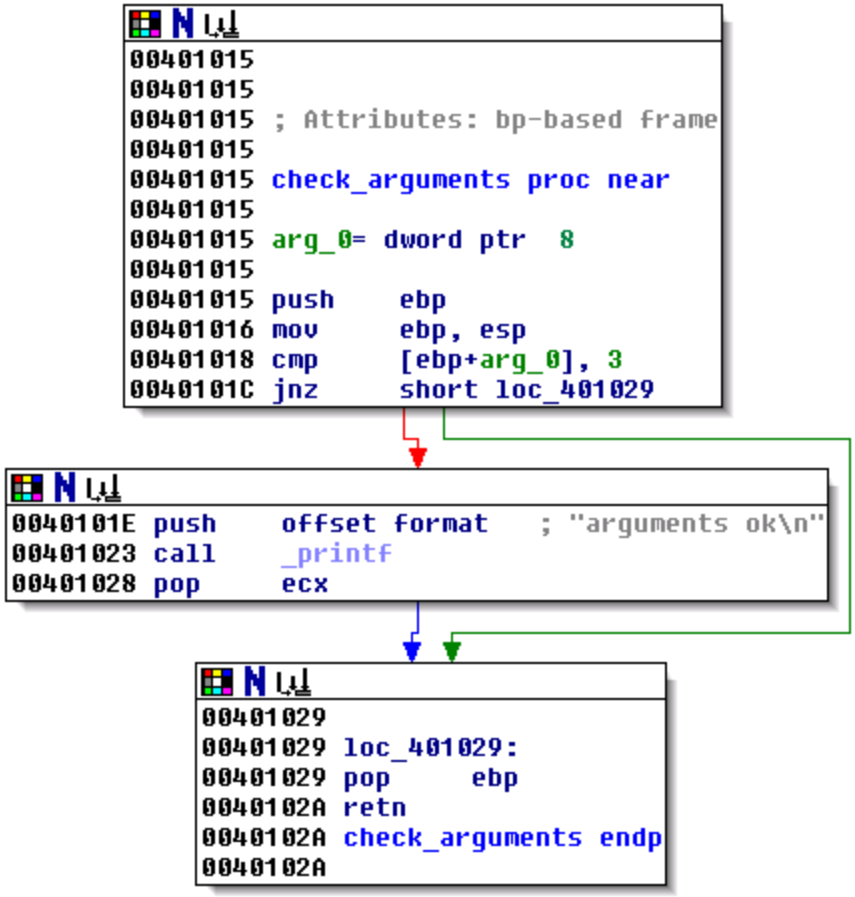
“binary x86” Code Example

check_parameters function

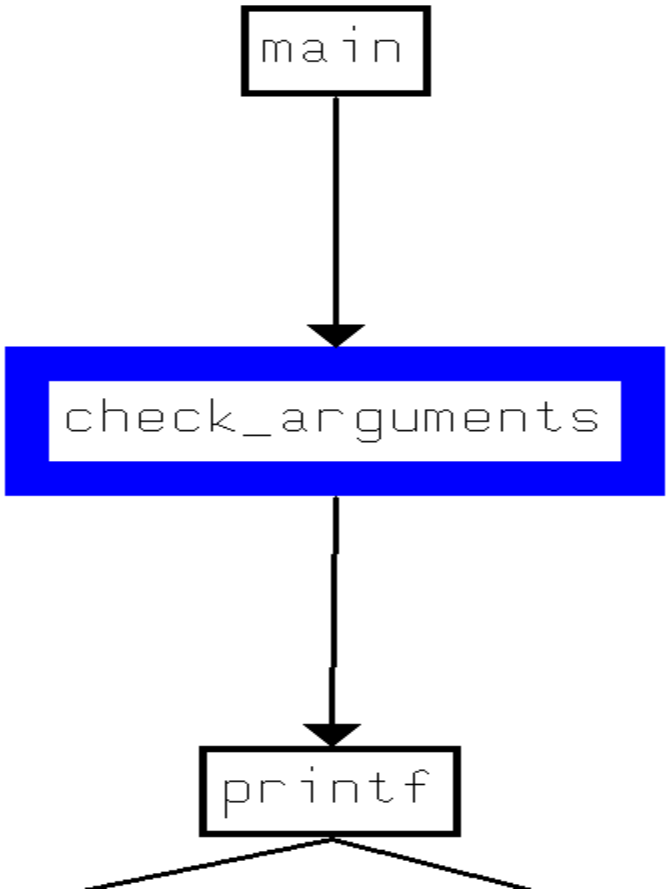


```
.text:00401000  55 8B EC FF 75 08 FF 75  08 E8 07 00 00 00 00 C4  Uï8 u■ u■F■...â-
.text:00401010  00 00 00 5D C3 55 8B EC  83 7D 08 03 75 0B 68 84  ■3+]+Uï8â}■■u■hä
.text:00401020  A0 40 00 E8 58 28 00 00  59 5D C3 90 EB 10 66 62  á@.FX(..Y]+Éd■fb
.text:00401030  3A 43 2B 2B 48 4F 4F 4B  90 E9 2C A1 40 00 A1 1F  :C++HOOKÉT,í@.í■
.text:00401040  A1 40 00 C1 E0 02 A3 23  A1 40 00 52 6A 00 E8 C1  í@.-a■ú#í@.Rj.F-
.text:00401050  88 00 00 8B D0 E8 76 10  00 00 5A E8 0C 04 00 00  ê..ï-Fv■..ZF■■..
.text:00401060  E8 6F 10 00 00 6A 00 E8  80 1C 00 00 59 68 C8 A0  Fo■..j.FÇ■..Yh+á
.text:00401070  40 00 6A 00 E8 9B 88 00  00 A3 27 A1 40 00 6A 00  @.j.F¢ê..ú'í@.j.
.text:00401080  E9 AF 6A 00 00 E9 AE 1C  00 00 33 C0 A0 11 A1 40  T»j..T«■..3+á■í@
.text:00401090  00 C3 A1 27 A1 40 00 C3  60 BB 00 50 B0 BC 53 68  .+í'í@.+`+.P¦+Sh
.text:004010A0  AD 0B 00 00 C3 B9 9C 00  00 00 0B C9 74 4D 83 3D  ¡■..+¦£...■+tMâ=
.text:004010B0  1F A1 40 00 00 73 0A B8  FE 00 00 00 E8 D7 FF FF  ■í@..s■+¦...F+
.text:004010C0  FF B9 9C 00 00 00 51 6A  08 E8 58 88 00 00 50 E8    ¦£...Qj■FXê..PF
.text:004010D0  7C 88 00 00 0B C0 75 0A  B8 FD 00 00 00 E8 B6 FF  |ê..■+u■+²...F¦
```

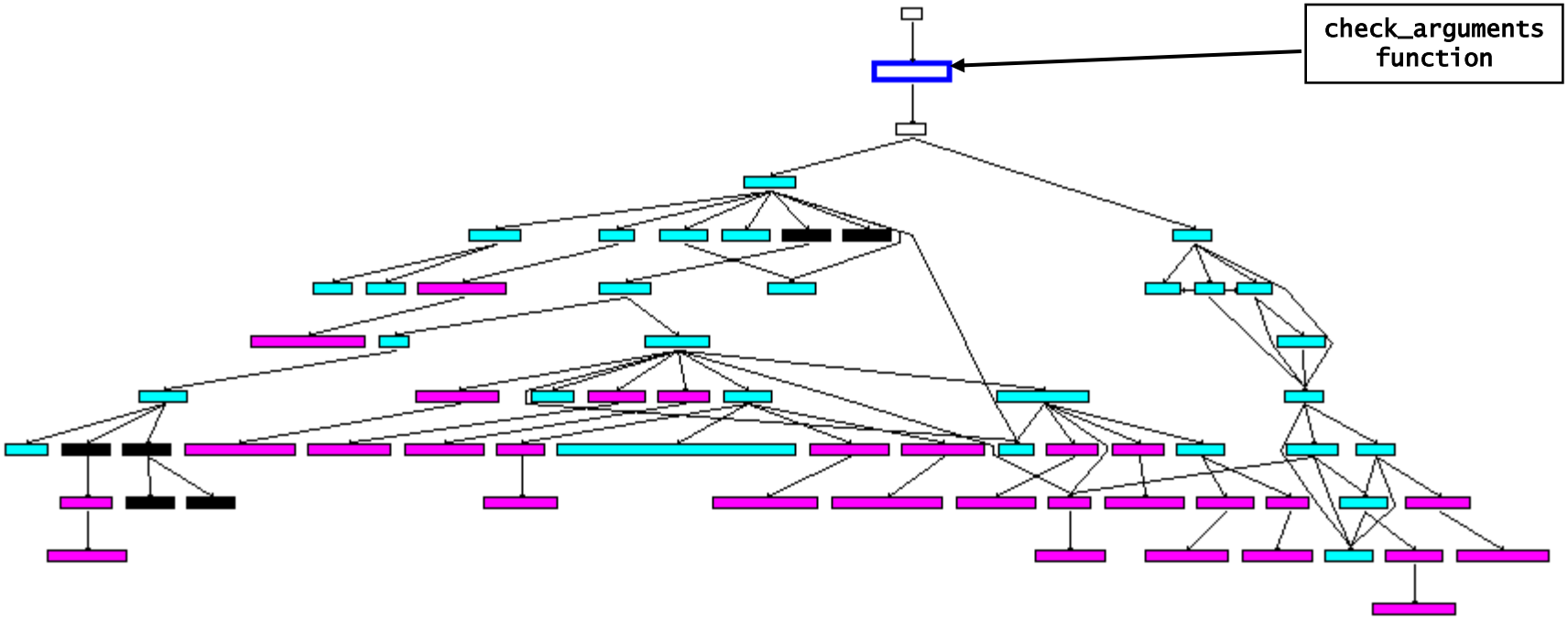
“function graph” example



“partial call graph” example



“complete call graph” example



test1.exe (vulnerable application)

```
void file_reader ( FILE *f , char buffer [ 256 ] )
{
  int len;
```

-2.147.483.648 to 2.147.483.647

```
/* Read the len */
  fread ( &len , sizeof ( int ) , 1 , f );

/* Check the len */
  if ( len <= 256 )
```

SECURITY PROBLEM

```
  {
  /* Read the data */
    fread ( buffer , len , 1 , f );
  }
}
```


0 to 4.294.967.295

EXAMPLE (signed/unsigned)

LEN = -1

LEN = 4.294.967.295

test2.exe (the patched application)

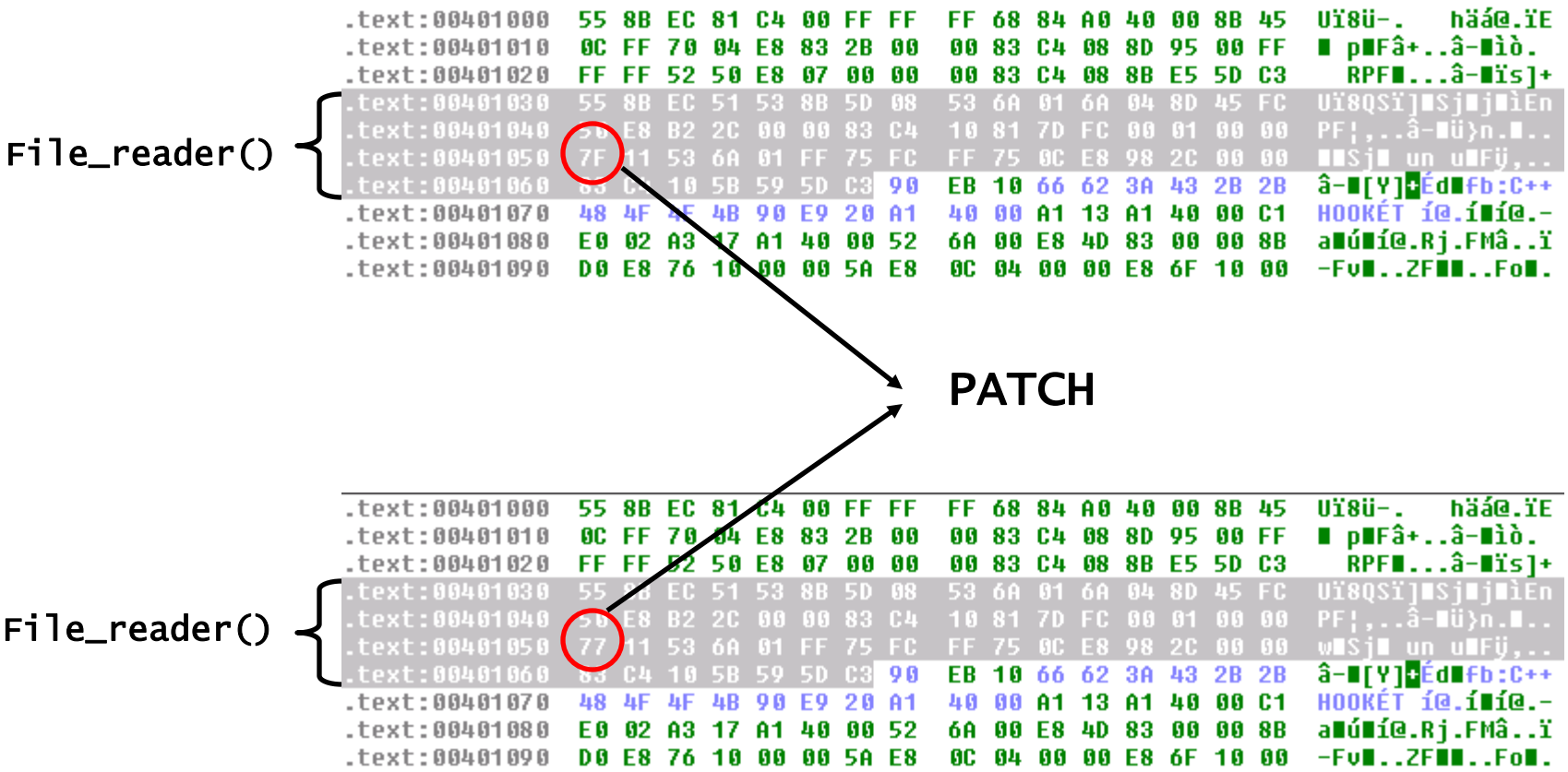
```
void file_reader ( FILE *f , char buffer [ 256 ] )
{
  unsigned int len;
```

SECURITY PATCH

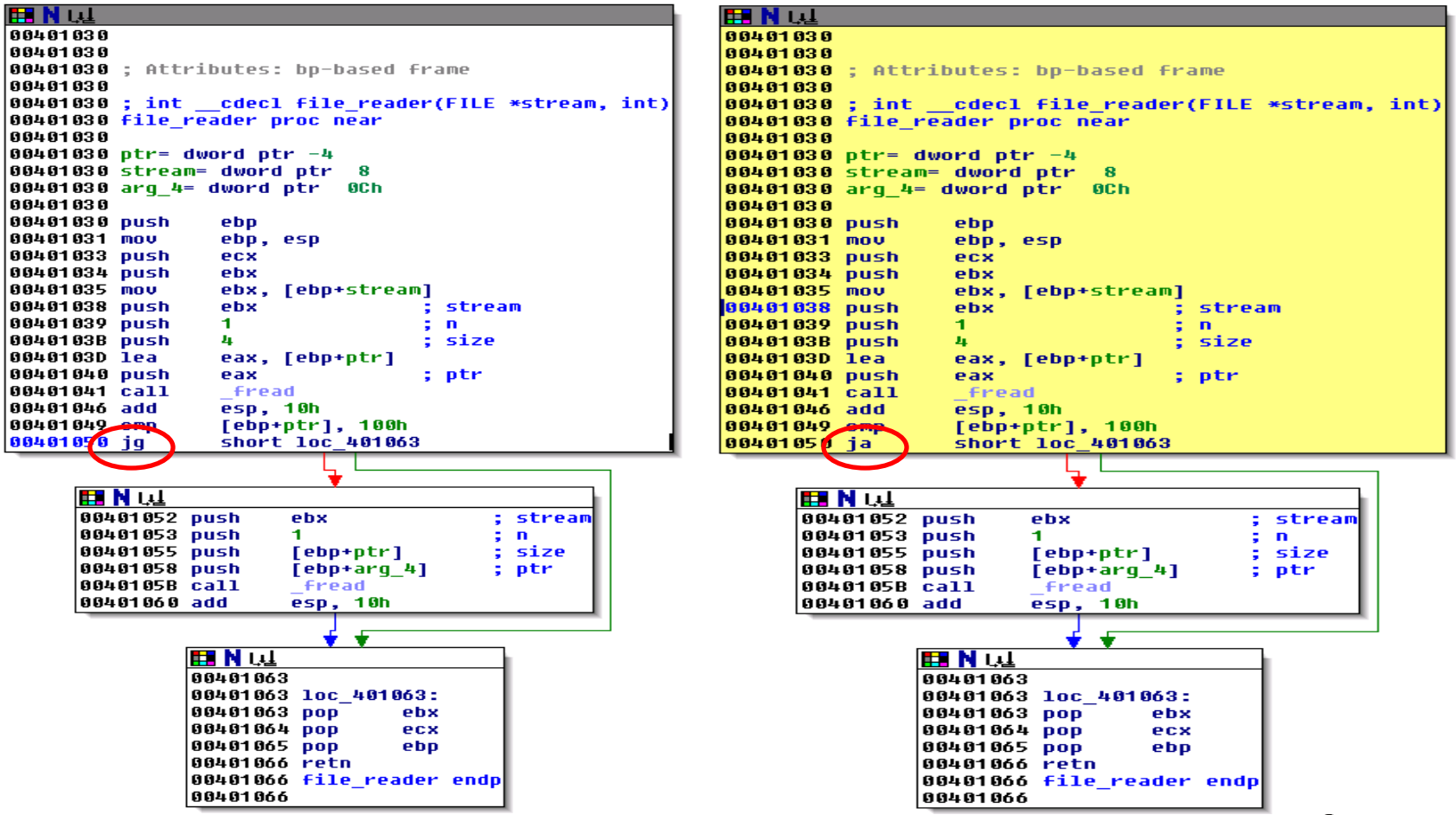
```
/* Read the len */
  fread ( &len , sizeof ( unsigned int ) , 1 , f );

/* Check the len */
  if ( len <= 256 )
  {
  /* Read the data */
    fread ( buffer , len , 1 , f );
  }
}
```

Comparing binary diferences



Comparing function graph differences



Simple Diff Demo (1 + 1 = 2)

Diffing “test1.exe” vs “test2.exe”

- Using Bindiff v2 (commercial)
- Using Patchdiff v2.0.6 (free)
- Using Darumgrim 2 v1.0 (free)
- Using Turbodiff 1.01b release 1 (free)

Story

Once upon a time ...

Story

An exploit writer

Story

And a boss ...

Story

One day his boss said

Story

Hey look at this !

- **Vulnerability on test1.exe**
- **CVE-9999-9999**
- **Patch Available from here**

Story

Do the exploit !

Story

No problem the exploit writer said

Story

I will find the vulnerability with bindiff ...

Story

Mmm it's rare ...

Story

I couldn't find the vulnerability ...

Story

I going to use another differ ...

Patchdiff

Story

WTF !!!!!

Story

It's my last chance said the EW

Story

Please Darumgrim 2 help me !

Story

Story

One week later ...

Story

His boss asked him

...

Story

Are you working very hard in the exploit ?

Story

Yes, of course said the exploit writer

Story

While he was looking to Angeline in his screen

Story

Is there another differ said the EW?

Story


Google™
Argentina
binary differ
Búsqueda avanzada
Preferencias
Herramientas del idioma
Buscar con Google
Voy a tener suerte
Buscar en: la Web páginas en español páginas de Argentina
Hacer de Google mi página de inicio
Programas de publicidad - Soluciones Empresariales - Todo acerca de Google - Google.com in English
©2009 - Privacidad

Story

Story

http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=tool&name=turbodiff

Story

Using turbodiff ...

Story

Binary diffing problems

- Functions matching:
  - test1.main () → test2.main ()
  - **?** test1.sub_401030 () → test2.???

- Searching differences between matched functions:
  - **?** test1.file_reader () != test2.file_reader ()

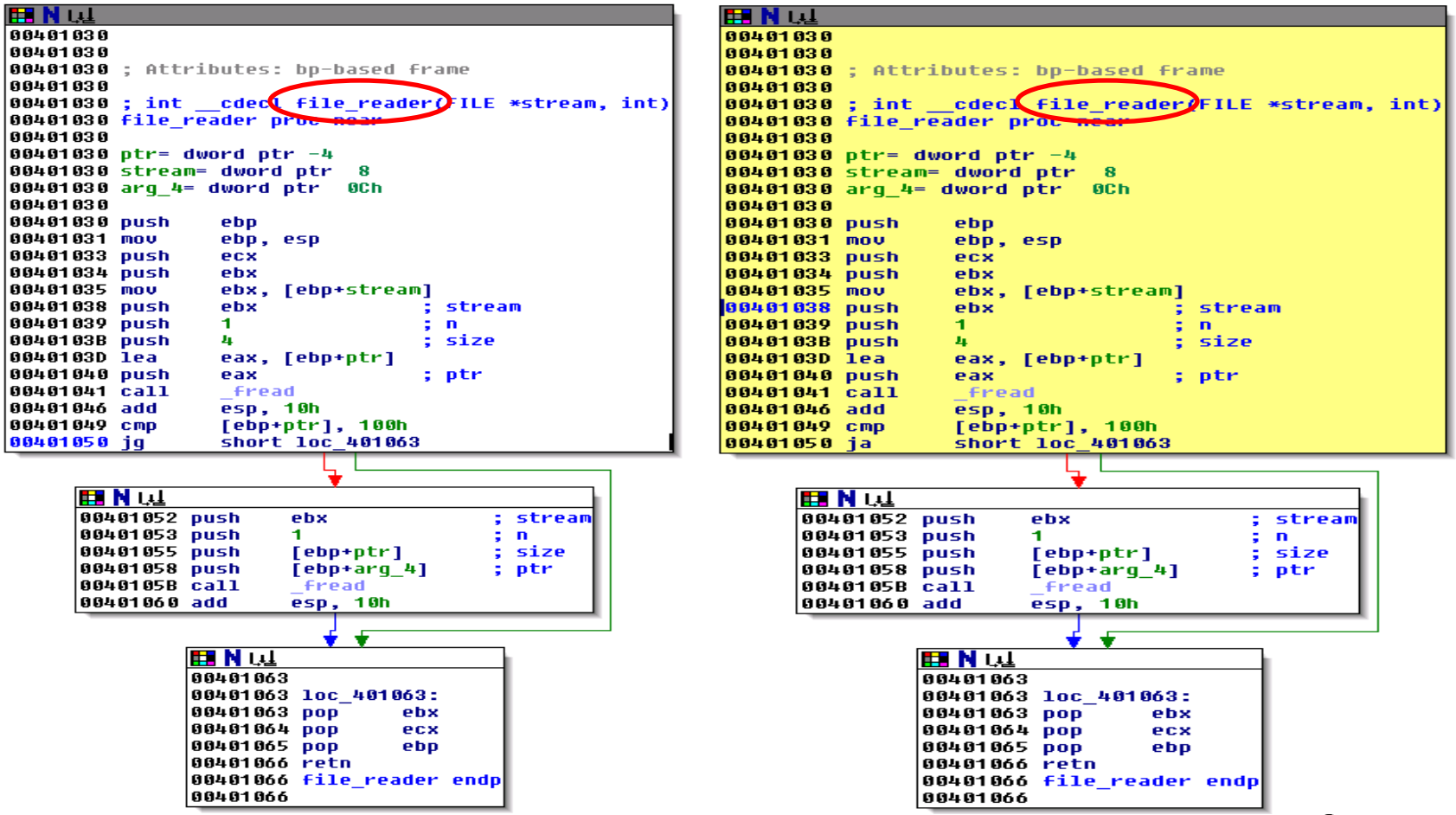
Functions matching problems

- Uncertainty
- Many heuristics are required for a correct match
- A good handling of probabilities is required (common sense !)

The simplest Heuristic

- Use symbols if they exist
  - High probability of correct matches
  - Matching via function names
    - » Mangled names (C): “_main ()”
    - » Demangled names (C++): “file::read ()”

Matching functions by name



Function list (example)

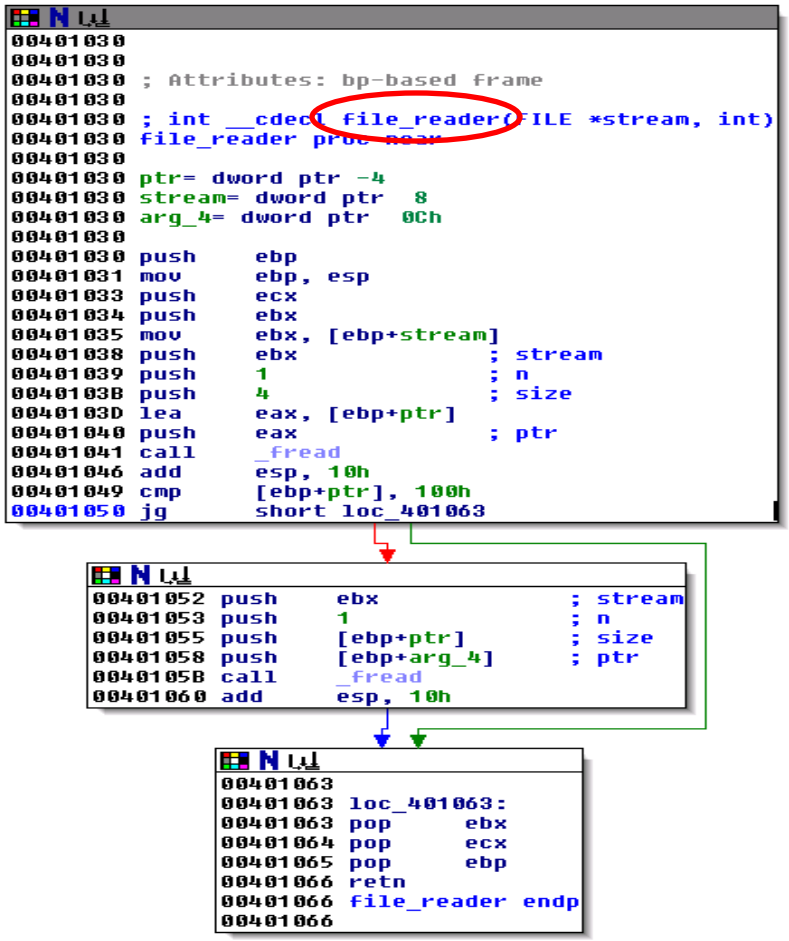
- **test.1.exe**

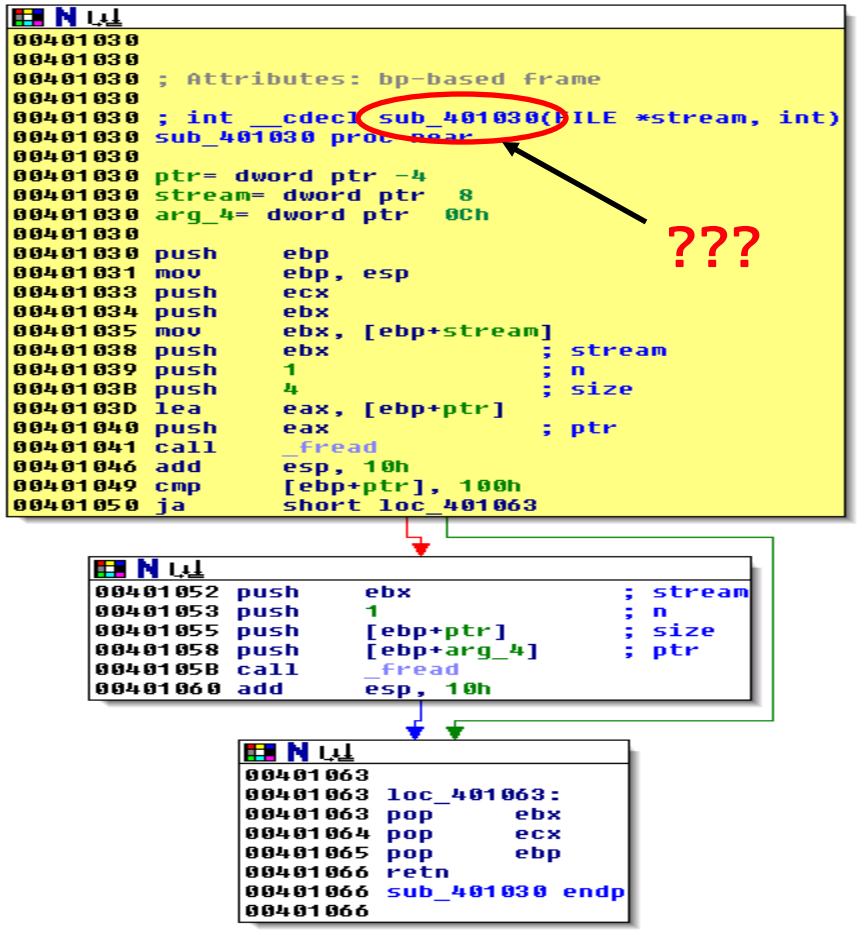
- 00401000 _main
- **00401030 file_reader**
- 00401068 start
- 004010c1 __GetExceptDLLinfo
- 004011b8 _calloc
- 004011e4 __rtl_close
- 004011f4 __close
- 00401204 @_virt_reserve

test2.exe

- 00401000 _main
- **00401030 file_reader**
- 00401068 start
- 004010c1 __GetExceptDLLinfo
- 004011b8 _calloc
- 004011e4 __rtl_close
- 004011f4 __close
- 00401204 @_virt_reserve

What if we don't have symbols ?

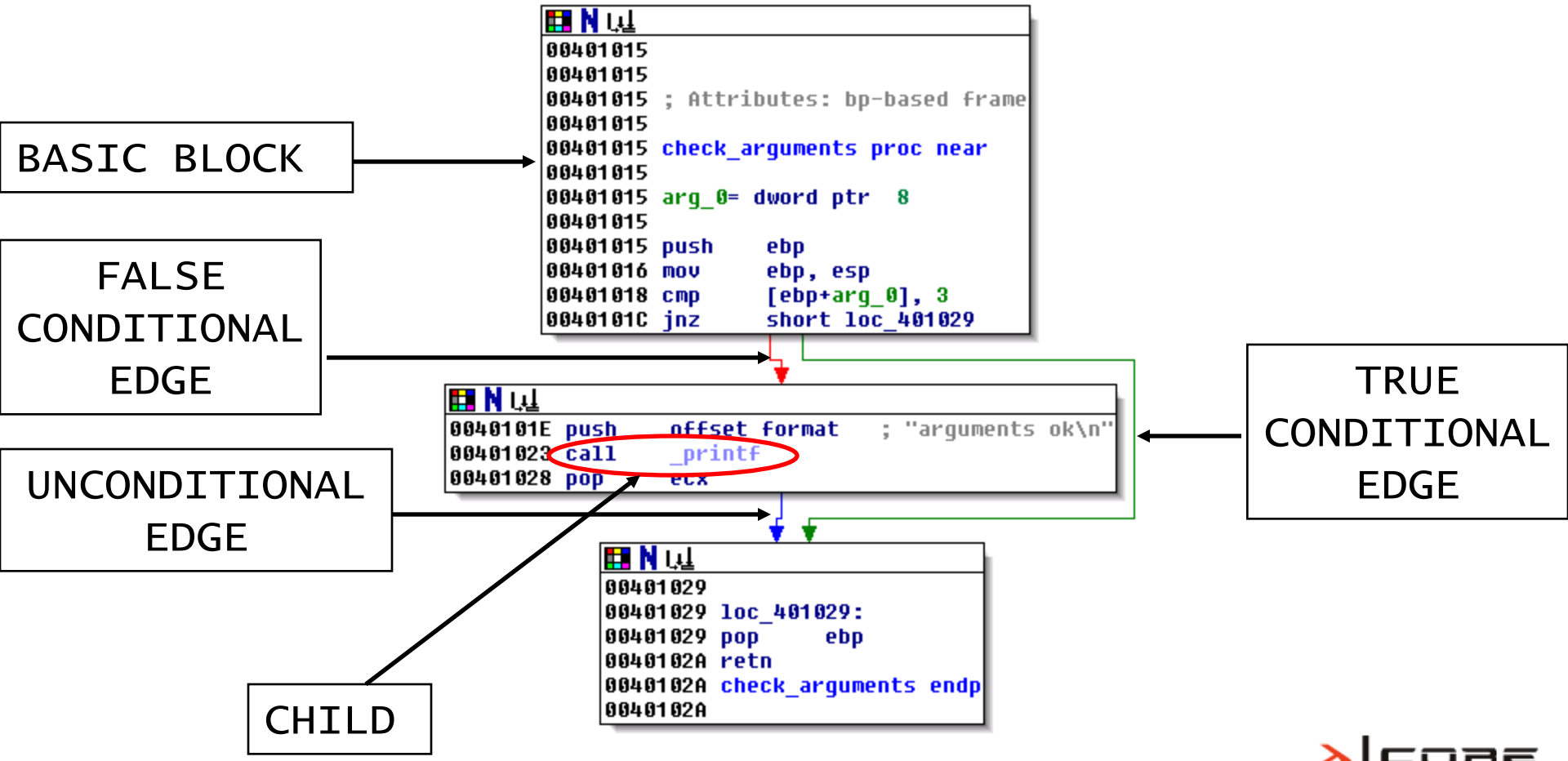




Function list without symbols

test.1.exe	**test2.exe**
00401000 **_main**	- 00401000 **sub_401000**
00401030 file_reader	**- 00401030 sub_401030**
00401068 **start**	- 00401068 **sub_401068**
004010c1 **__GetExceptDLLinfo**	- 004010c1 **sub_4010c1**
004011b8 **_calloc**	- 004011b8 **sub_4011b8**
004011e4 **__rtl_close**	- 004011e4 **sub_4011e4**
004011f4 **__close**	- 004011f4 **sub_4011f4**
00401204 **@_virt_reserve**	- 00401204 **sub_401204**

Entering the graph world ...



Essential function graph characteristics

- A function is made of:
  - Basic Blocks
    - » Have Parents (calls from functions)
    - » Have Children (calls to functions)
  - Edges
    - » Conditionals (TRUE/FALSE)
    - » Unconditionals (GOTOs)

check_argument () characteristics

- 3 basic blocks
  - 2 without children
  - 1 child
- 3 edges
  - 1 true
  - 1 false
  - 1 unconditional

Matrix representation

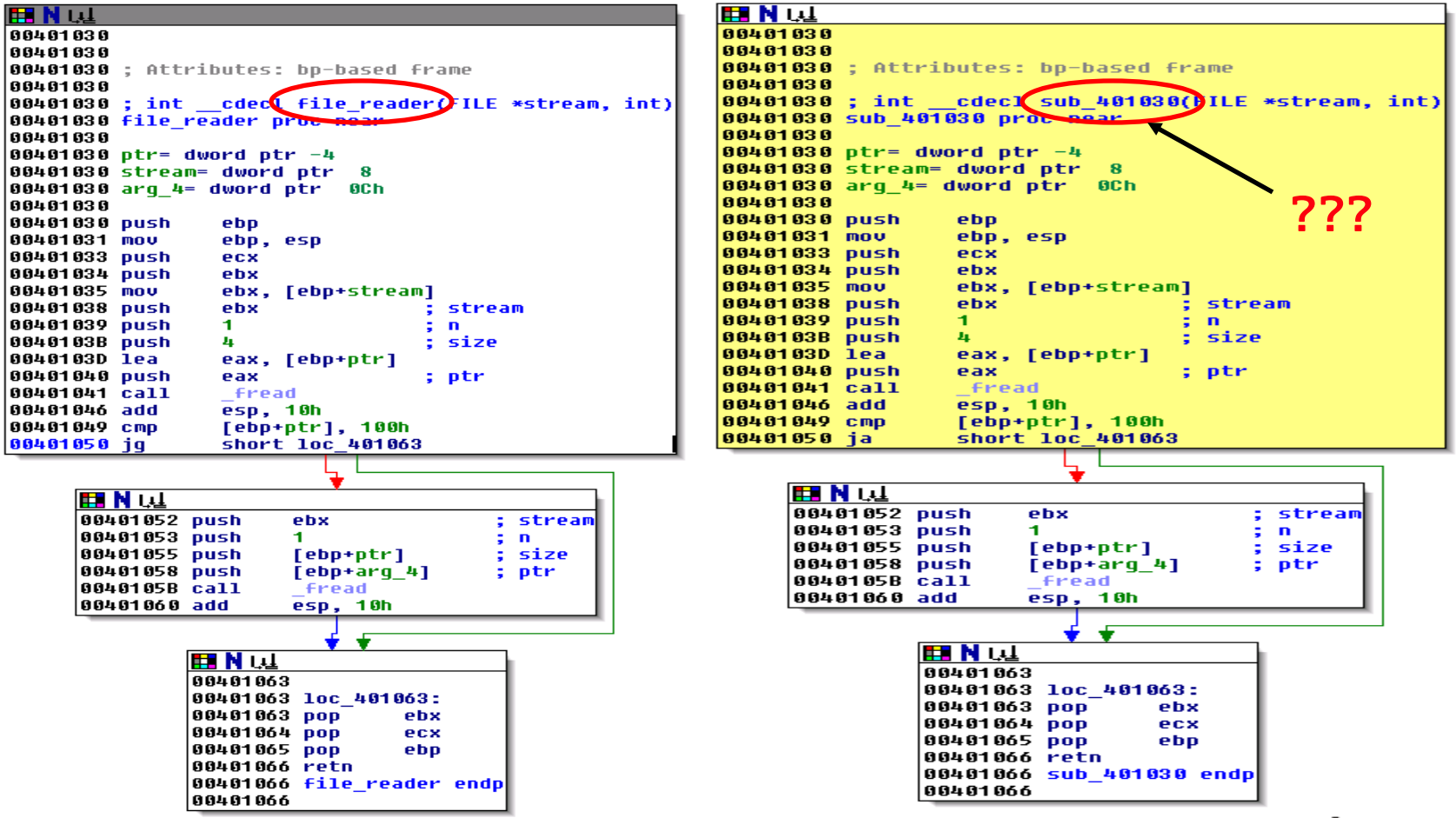
- A = 0x401015 B = 0x40101e C = 0x401029
- 1 = green edge 2 = red edge 3 = blue edge

*	A	B	C
A	-	1	2
B	-	-	3
C	-	-	-

CONNECTIONS

A → B
A → C
B → C

The Function Graph Comparation Heuristic



test1.file_reader vs. test2.file_reader

- **A = 0x401030 B = 0x401052 C = 0x401063**
- **1 = green edge 2 = red edge 3 = blue edge**

} `test1.file_reader()`

- **A = 0x401030 B = 0x401052 C = 0x401063**
- **1 = green edge 2 = red edge 3 = blue edge**

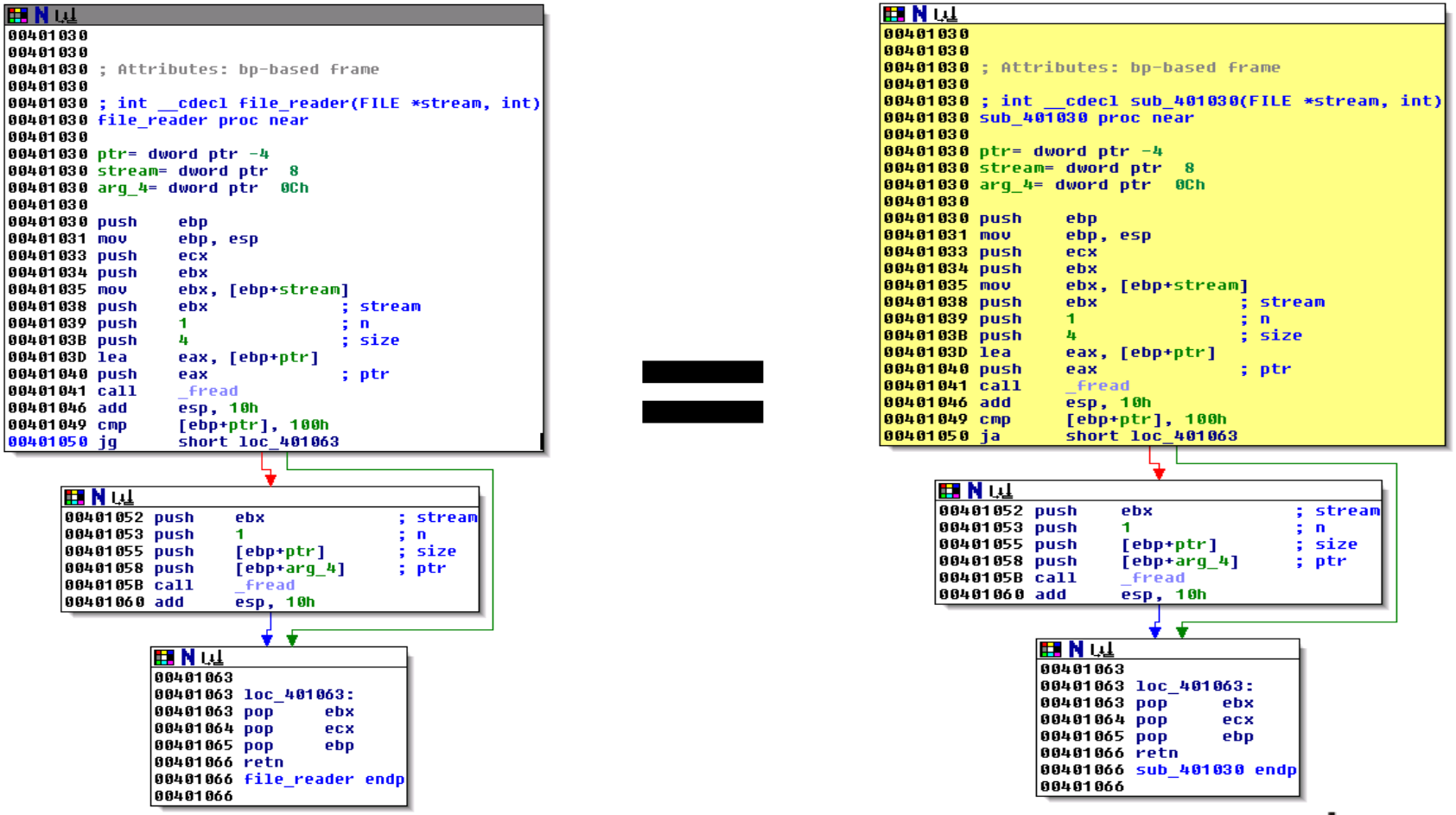
} `test2.file_reader()`

*	A	B	C
A	-	1	2
B	-	-	3
C	-	-	-

VS

*	A	B	C
A	-	1	2
B	-	-	3
C	-	-	-

The same graph



We add code

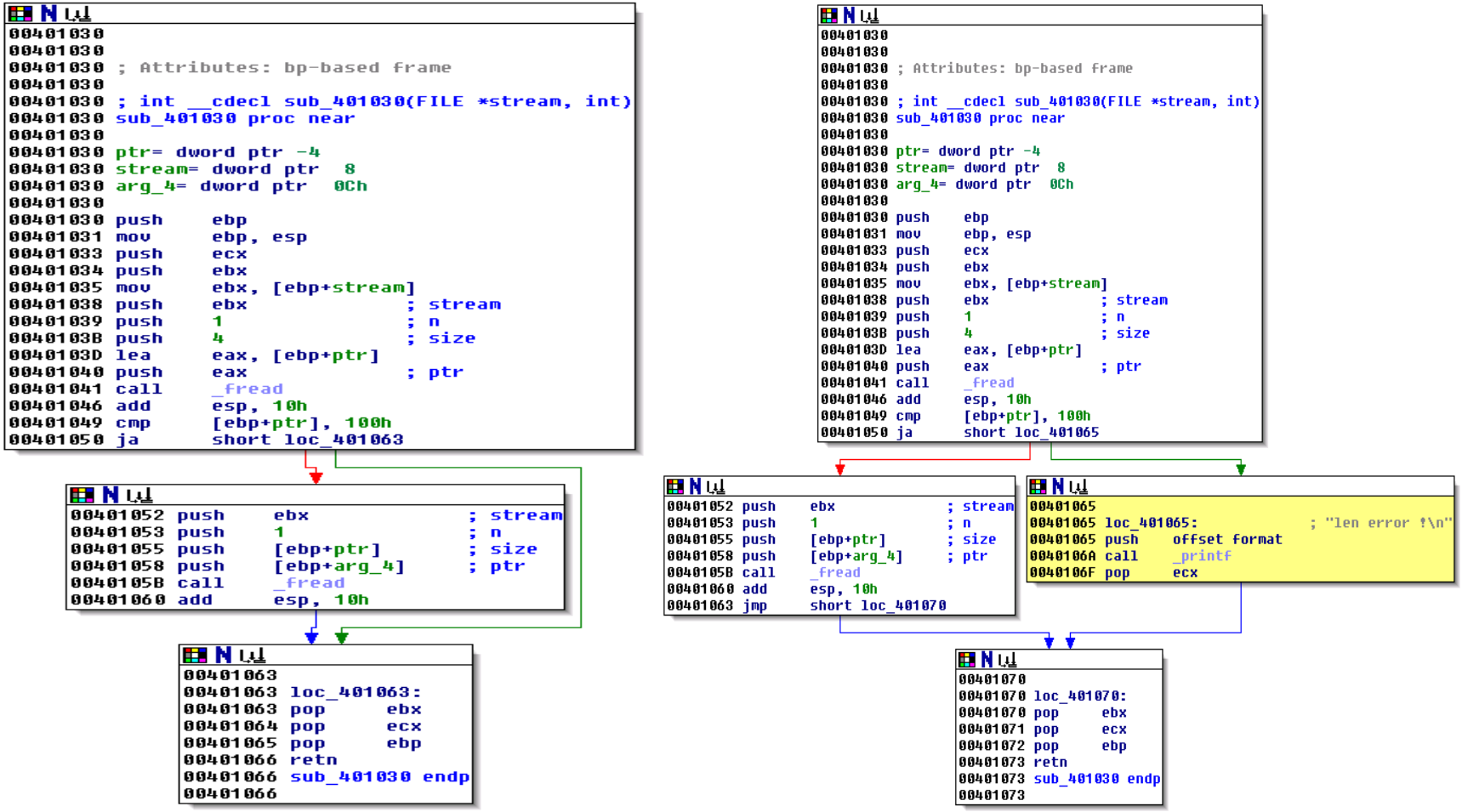
```
test2.file_reader ()

/* Read the len */
if ( len <= 256 )
{
/* Read the data */
  fread ( buffer , len , 1 , f );
}
```

```
test3.file_reader ()

/* Read the len */
if ( len <= 256 )
{
/* Read the data */
  fread ( buffer , len , 1 , f );
}
else
{
  printf ( "len error !\n" );
}
```

test2. file_reader vs test3. file_reader



test2.file_reader vs. test3.file_reader

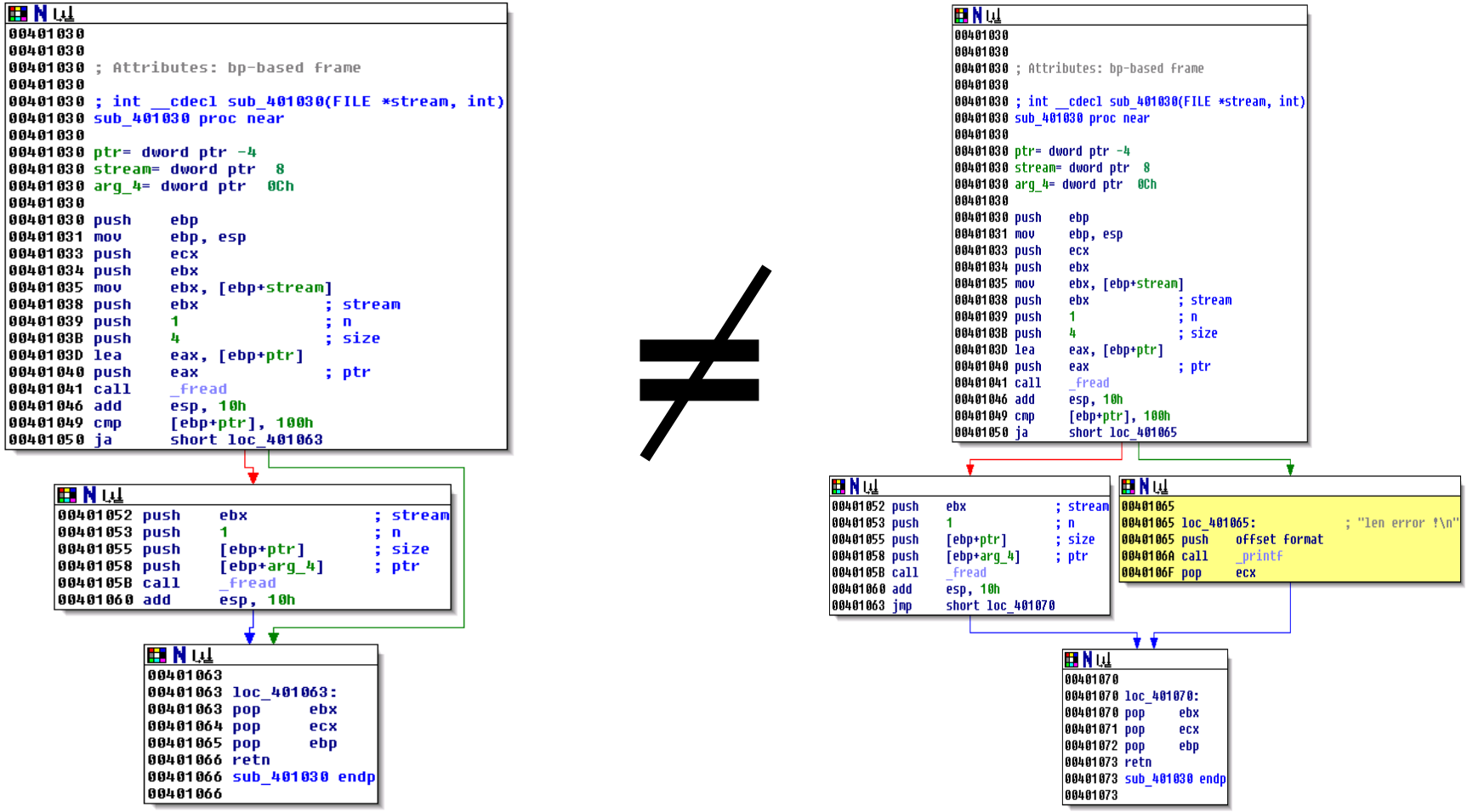
- **test2.exe**
- **A = 0x401030 B = 0x401052 C = 0x401063**
- **test3.exe**
- **A = 0x401030 B = 0x401065 C = 0x401052 D = 0x401070**

*	A	B	C
A	-	1	2
B	-	-	3
C	-	-	-

VS

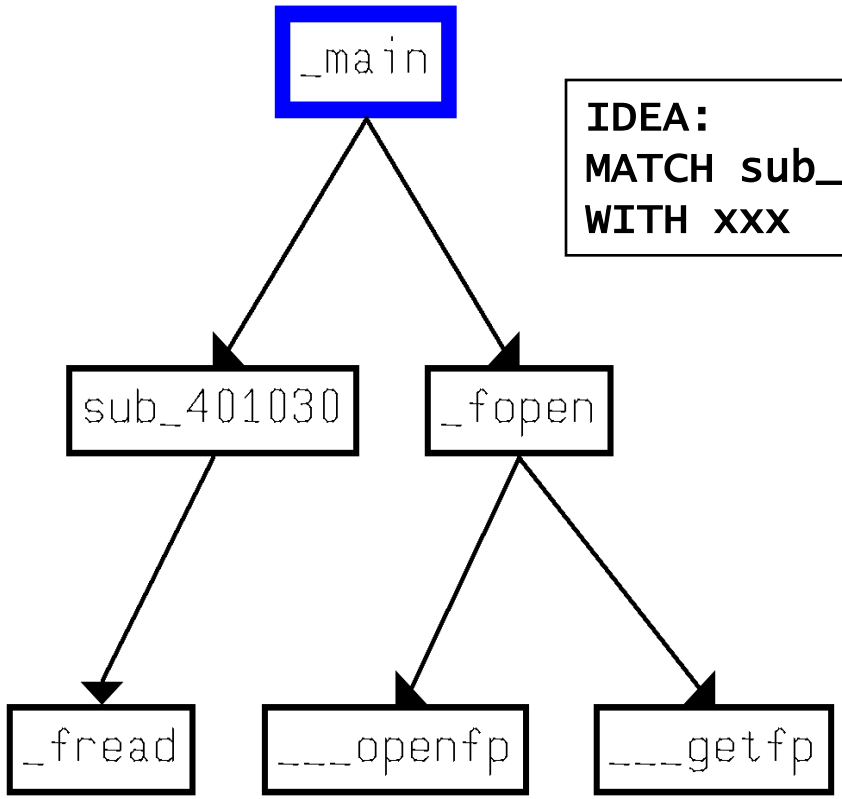
	A	B	C	D
A	-	1	2	-
B	-	-	-	3
C	-	-	-	3
D	-	-	-	-

Different Graph



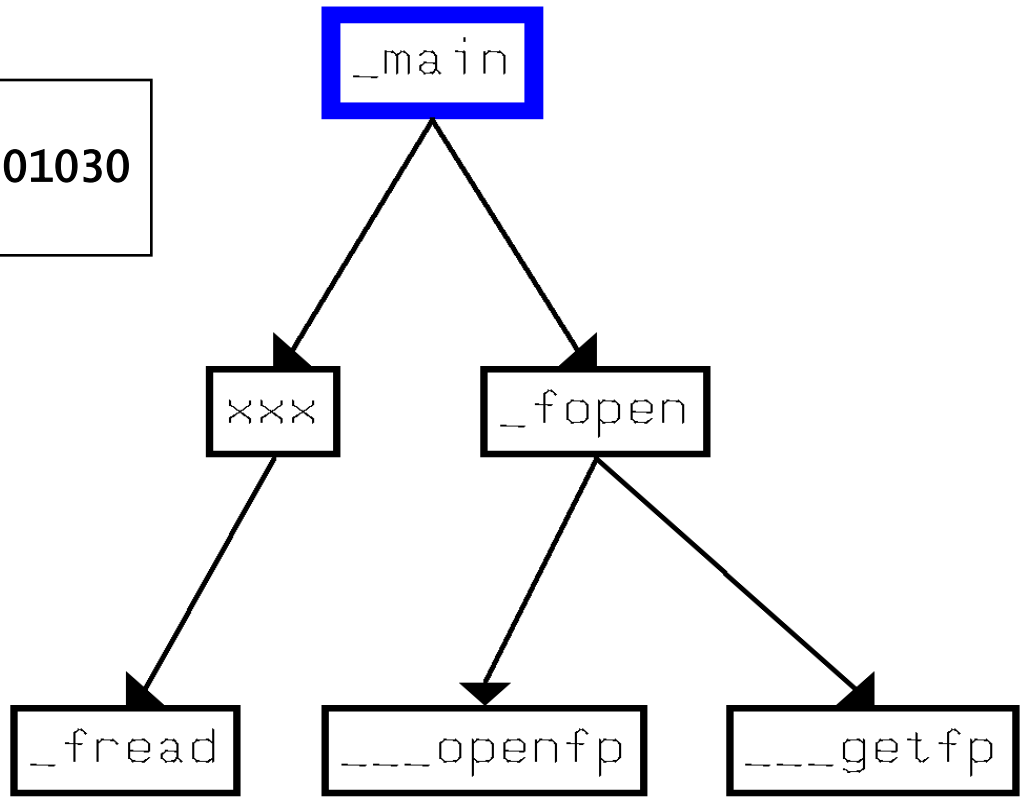
The call graph heuristic

- **test2.exe**



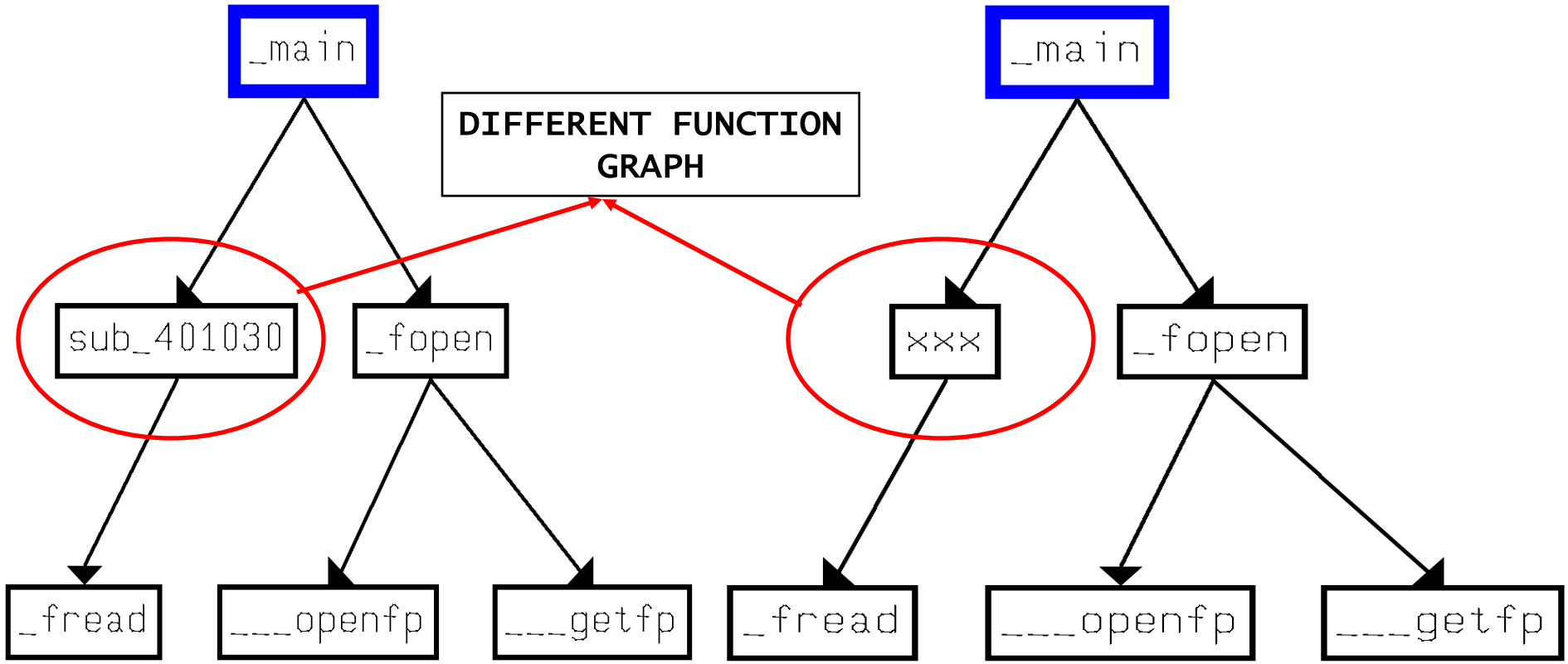
IDEA:
MATCH sub_401030
WITH xxx

test3.exe



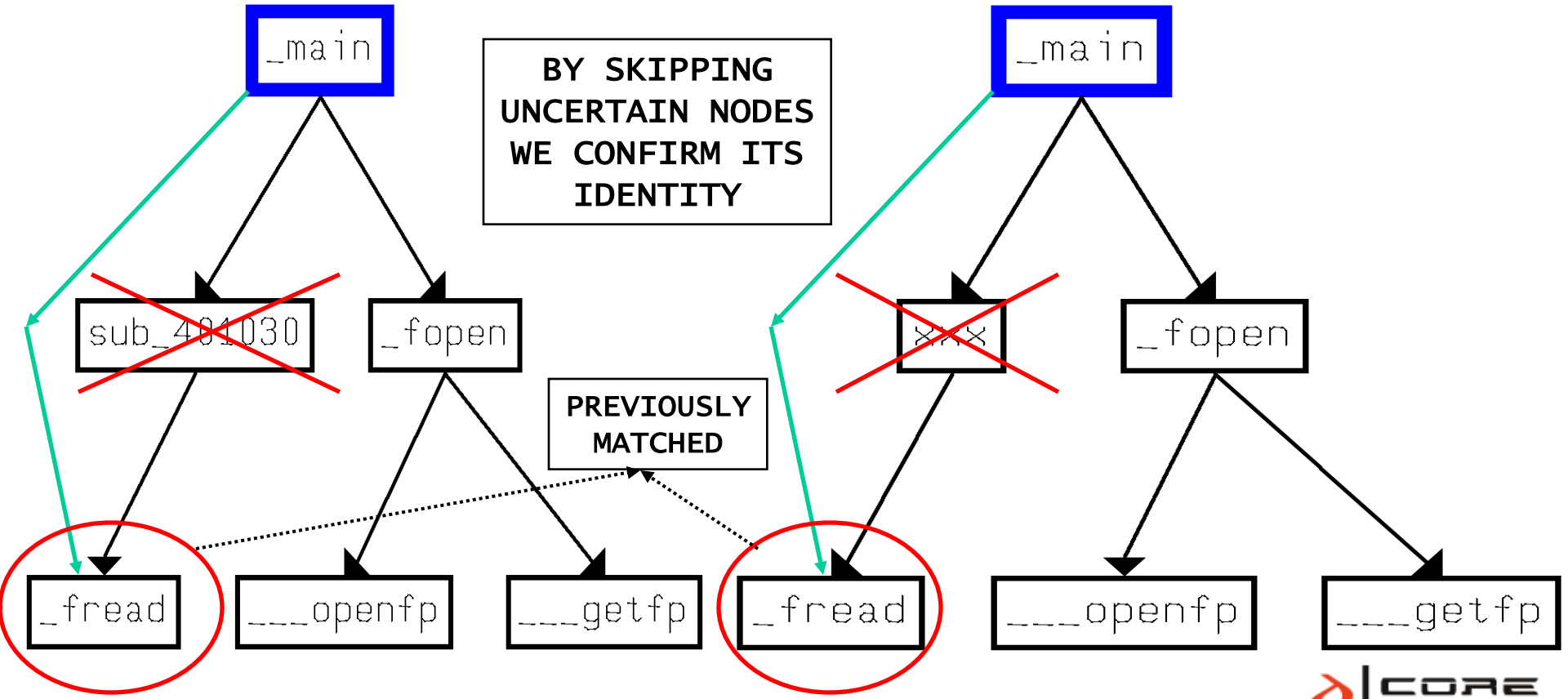
A place in the world

- **test2.exe** **test3.exe**



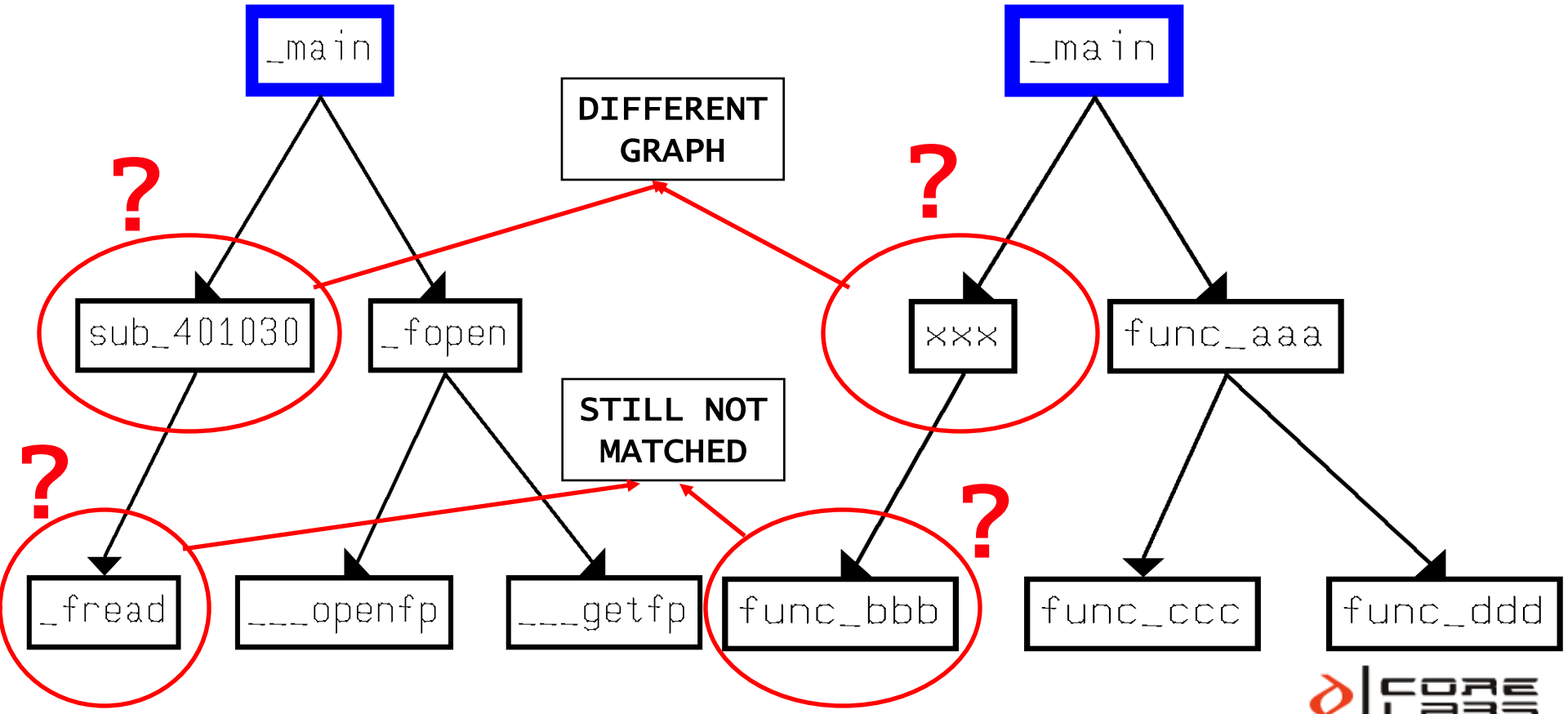
Exploring the call graph

- **test2.exe** **test3.exe**

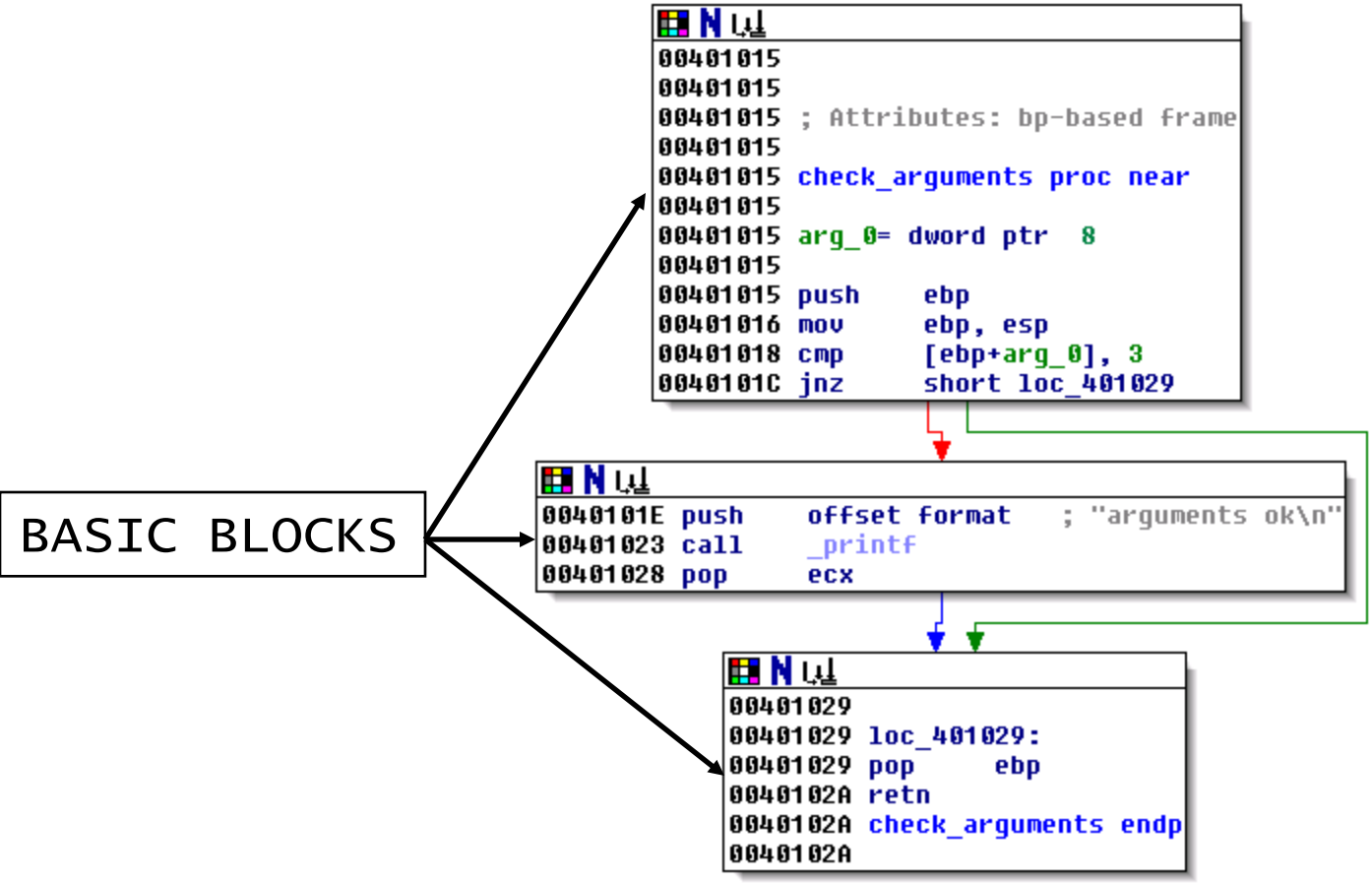


Uncertainty

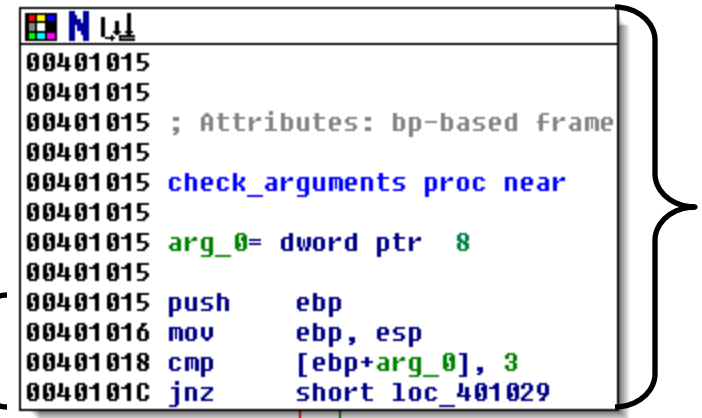
- **test2.exe** **test3.exe**



Taking information from ...



Checksumming basic blocks



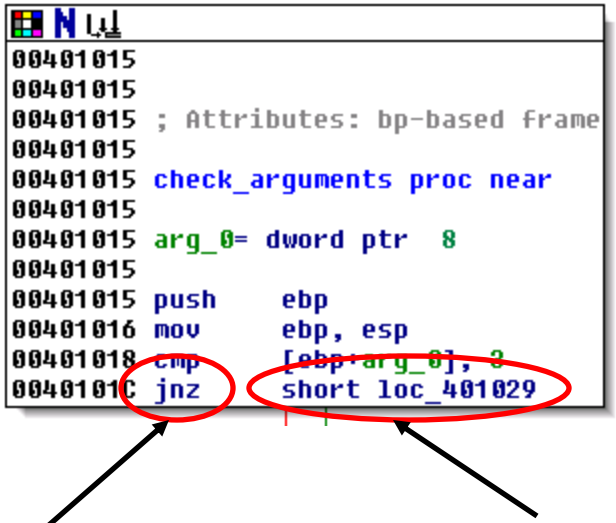
CHECKSUM = 0x429

- 4 INSTRUCTIONS
- 9 BYTES
- 0 CHILDREN
- 0 GLOBAL REFERENCES
- 0 INPUTS
- 2 OUTPUTS

FALSE CONDITION

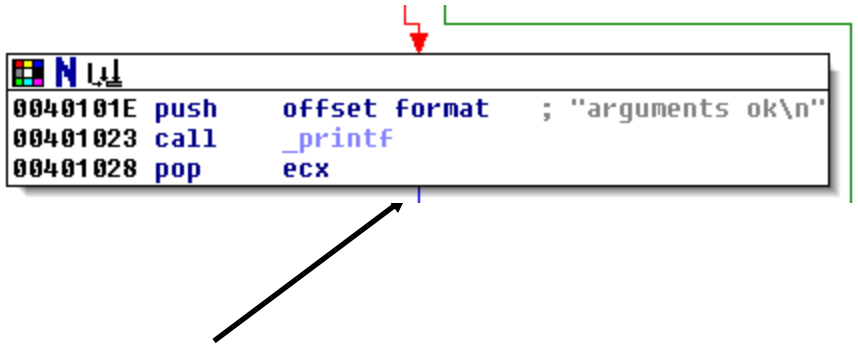
TRUE CONDITION

Be careful with this!



Checksumming basic blocks

- 3 INSTRUCTIONS
- 11 BYTES
- 1 CHILD
- 1 GLOBAL REFER.
- 1 INPUT
- 1 OUTPUT



UNCONDITIONAL

CHECKSUM = 0x110059

Comparing function checksums

CHK 0x100ACD

```
00401030
00401030
00401030 ; Attributes: bp-based frame
00401030
00401030 ; int __cdecl file_reader(FILE *stream, int)
00401030 file_reader proc near
00401030
00401030 ptr= dword ptr -4
00401030 stream= dword ptr  8
00401030 arg_4= dword ptr  0Ch
00401030
00401030 push    ebp
00401031 mov     ebp, esp
00401033 push    ecx
00401034 push    ebx
00401035 mov     ebx, [ebp+stream]
00401038 push    ebx             ; stream
00401039 push    1               ; n
0040103B push    4               ; size
0040103D lea     eax, [ebp+ptr]
00401040 push    eax             ; ptr
00401041 call    _fread
00401046 add     esp, 10h
00401049 cmp     [ebp+ptr], 100h
00401050 jg      short loc_401063
```

CHK 0x100605

```
00401052 push    ebx             ; stream
00401053 push    1               ; n
00401055 push    [ebp+ptr]       ; size
00401058 push    [ebp+arg_4]     ; ptr
0040105B call    _fread
00401060 add     esp, 10h
```

CHK 0x1D4

```
00401063
00401063 loc_401063:
00401063 pop     ebx
00401064 pop     ecx
00401065 pop     ebp
00401066 retn
00401066 file_reader endp
00401066
```

CHK 0x100AC7

```
00401030
00401030
00401030 ; Attributes: bp-based frame
00401030
00401030 ; int __cdecl file_reader(FILE *stream, int)
00401030 file_reader proc near
00401030
00401030 ptr= dword ptr -4
00401030 stream= dword ptr  8
00401030 arg_4= dword ptr  0Ch
00401030
00401030 push    ebp
00401031 mov     ebp, esp
00401033 push    ecx
00401034 push    ebx
00401035 mov     ebx, [ebp+stream]
00401038 push    ebx             ; stream
00401039 push    1               ; n
0040103B push    4               ; size
0040103D lea     eax, [ebp+ptr]
00401040 push    eax             ; ptr
00401041 call    _fread
00401046 add     esp, 10h
00401049 cmp     [ebp+ptr], 100h
00401050 ja      short loc_401063
```

CHK 0x100605

```
00401052 push    ebx             ; stream
00401053 push    1               ; n
00401055 push    [ebp+ptr]       ; size
00401058 push    [ebp+arg_4]     ; ptr
0040105B call    _fread
00401060 add     esp, 10h
```

CHK 0x1D4

```
00401063
00401063 loc_401063:
00401063 pop     ebx
00401064 pop     ecx
00401065 pop     ebp
00401066 retn
00401066 file_reader endp
00401066
```

Problem: Dependence of architecture

x86 CHK1 != CHK2 !!! **ARM**

```
00401015
00401015
00401015 ; Attributes: bp-based frame
00401015
00401015 check_arguments proc near
00401015
00401015 arg_0= dword ptr  8
00401015
00401015 push    ebp
00401016 mov     ebp, esp
00401018 cmp     [ebp+arg_0], 3
0040101C jnz     short loc_401029
```

```
0040101E push    offset format   ; "arguments ok\n"
00401023 call    _printf
00401028 pop     ecx
```

```
00401029
00401029 loc_401029:
00401029 pop     ebp
0040102A retn
0040102A check_arguments endp
0040102A
```

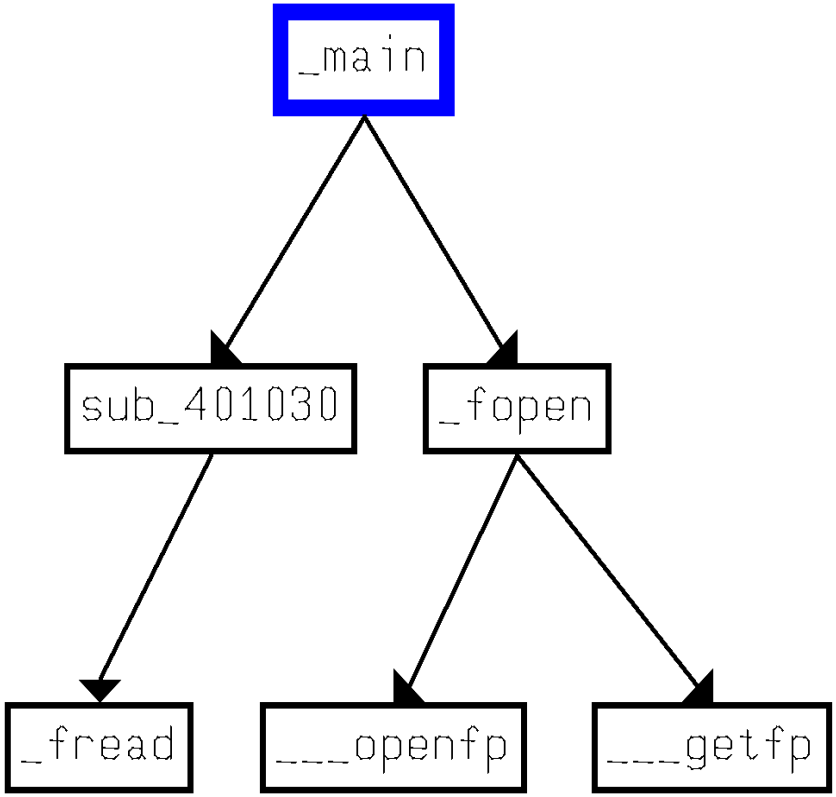
```
00001F2C
00001F2C
00001F2C
00001F2C EXPORT _check_arguments
00001F2C _check_arguments
00001F2C
00001F2C var_18= -0x18
00001F2C var_14= -0x14
00001F2C var_10= -0x10
00001F2C var_C= -0xC
00001F2C var_8= -8
00001F2C var_4= -4
00001F2C
00001F2C SUB      SP, SP, #8
00001F30 STR      LR, [SP,#8+var_4]
00001F34 STR      R7, [SP+8+var_8]
00001F38 MOV      R7, SP
00001F3C SUB      SP, SP, #8
00001F40 STR      R0, [R7,#0x10+var_14]
00001F44 STR      R1, [R7,#0x10+var_18]
00001F48 LDR      R3, [R7,#0x10+var_14]
00001F4C CMP      R3, #3
00001F50 BNE      loc_1F60
```

```
00001F54 LDR      R3, =(aArgumentsOk - 0x1F60)
00001F58 ADD      R0, PC, R3        ; "arguments ok"
00001F5C BL       _printf
```

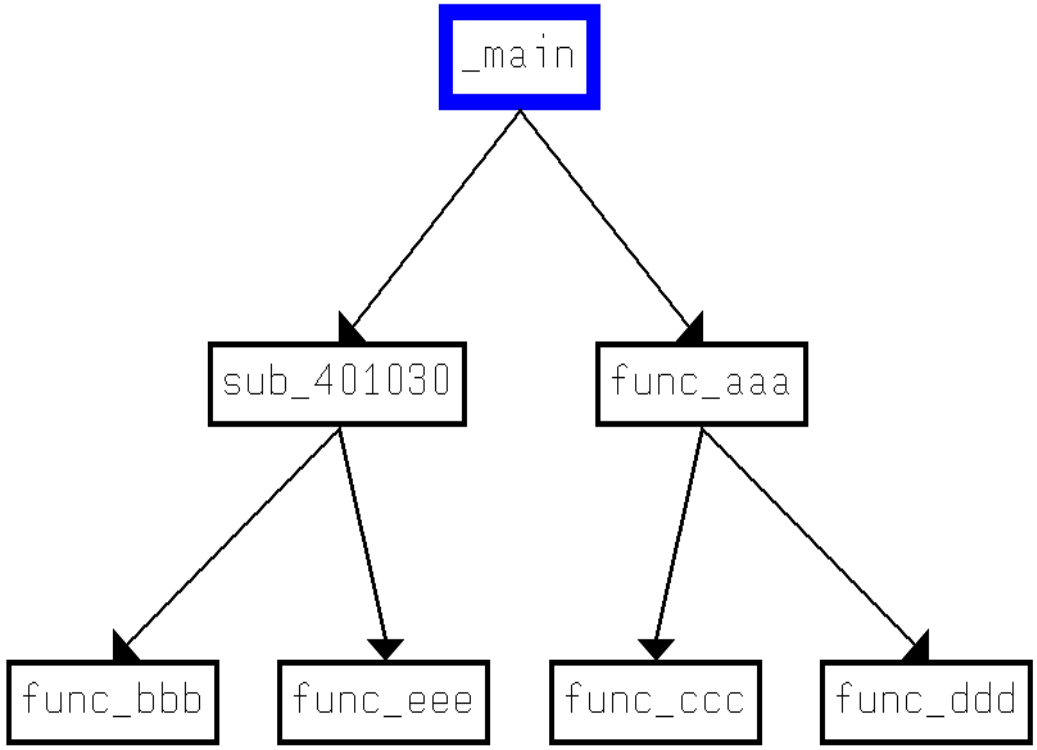
```
00001F60
00001F60 loc_1F60
00001F60 MOV      SP, R7
00001F64 LDR      R7, [SP+0x10+var_10]
00001F68 LDR      LR, [SP,#0x10+var_C]
00001F6C ADD      SP, SP, #8
00001F70 BX       LR
00001F70 ; End of function _check_arguments
00001F70
```

Edges order

- **test2.exe**

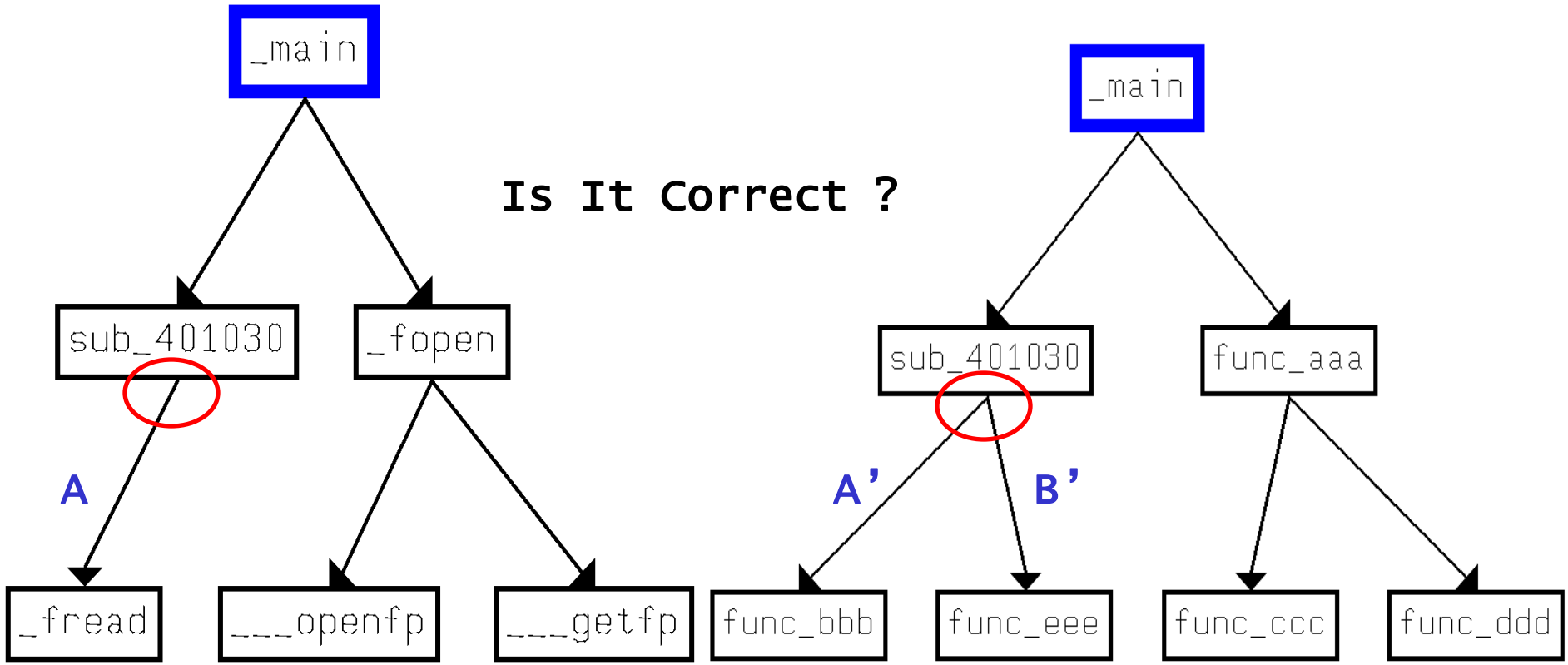


test3.exe



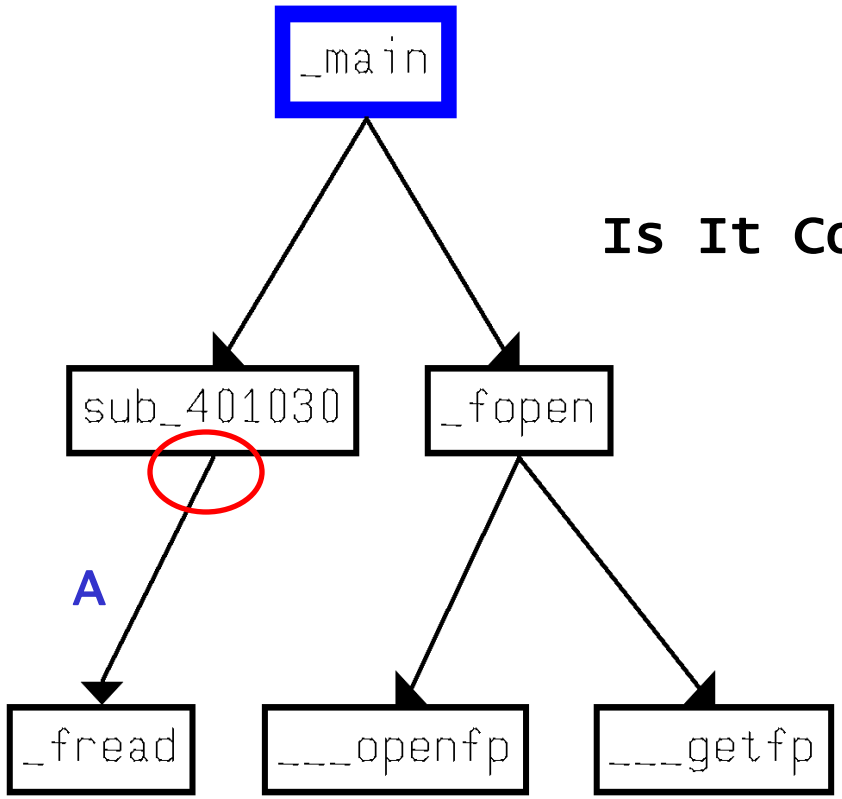
Edges order 1

- **test2.exe** **test3.exe**



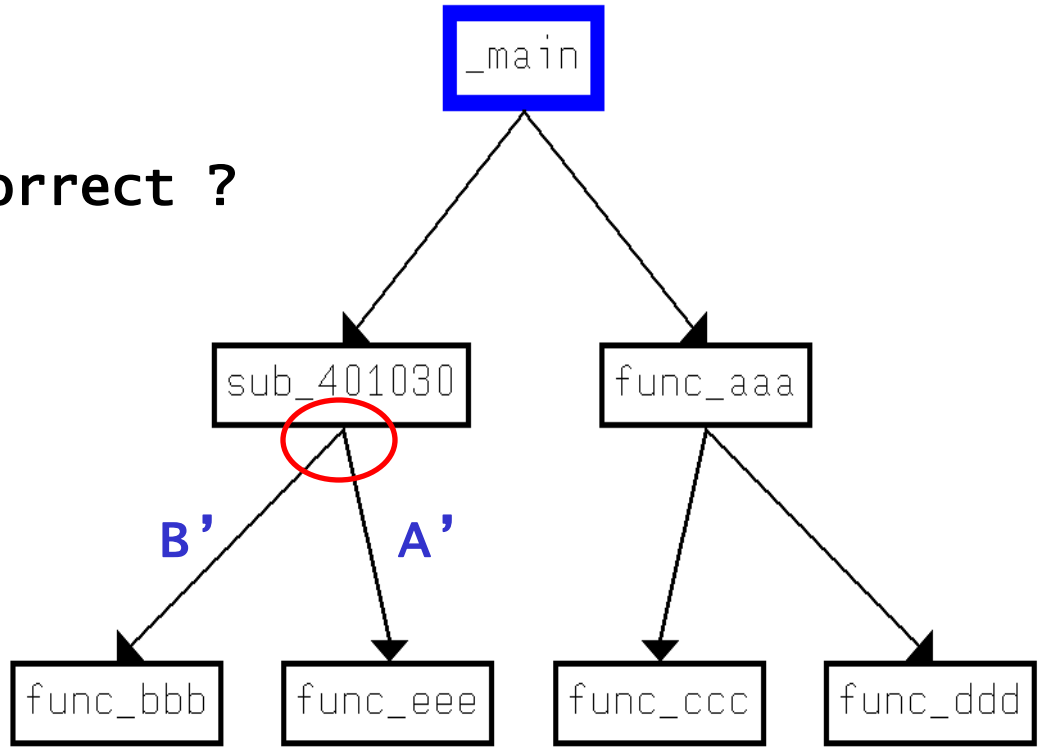
Edges order 2

- **test2.exe**

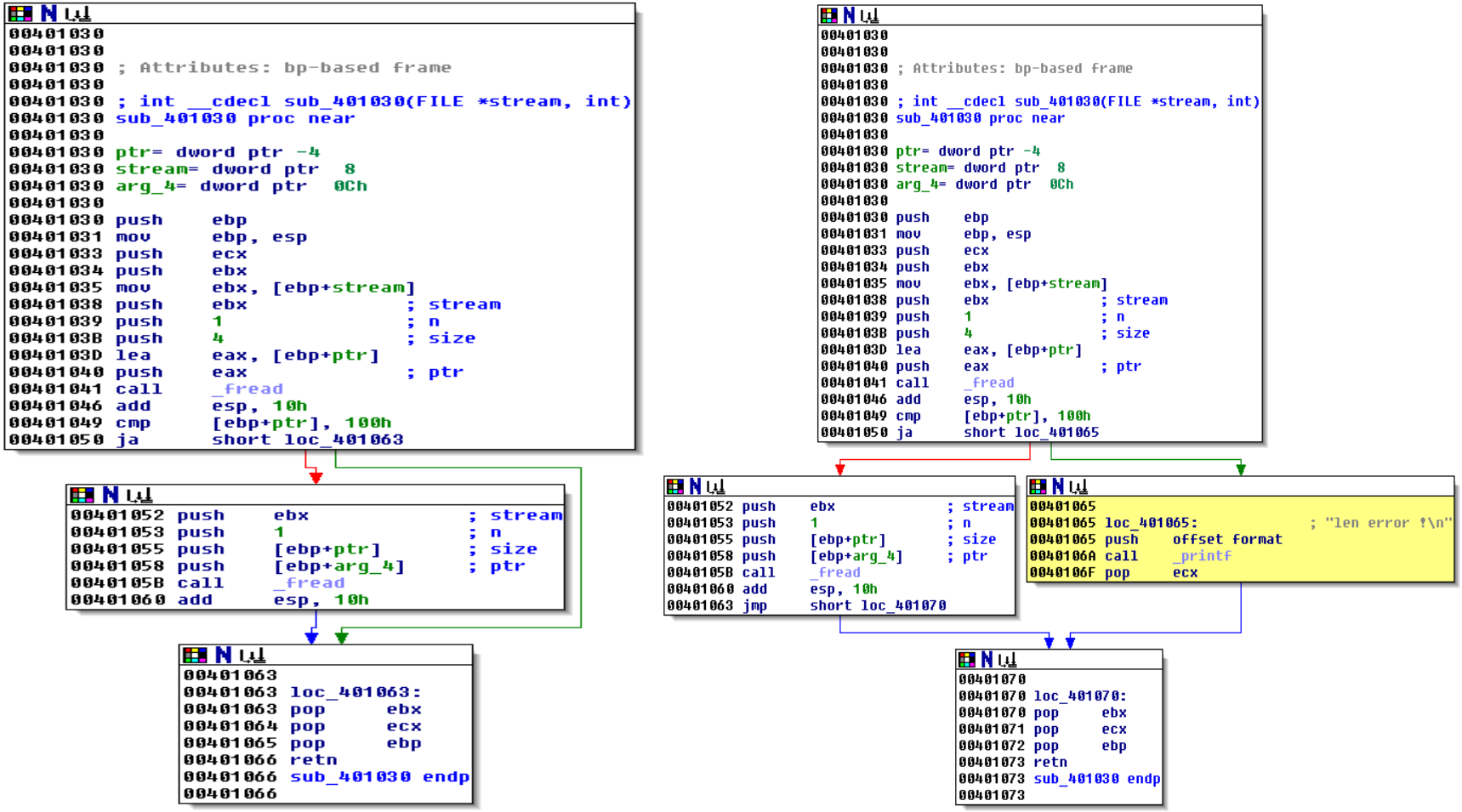


Is It Correct ?

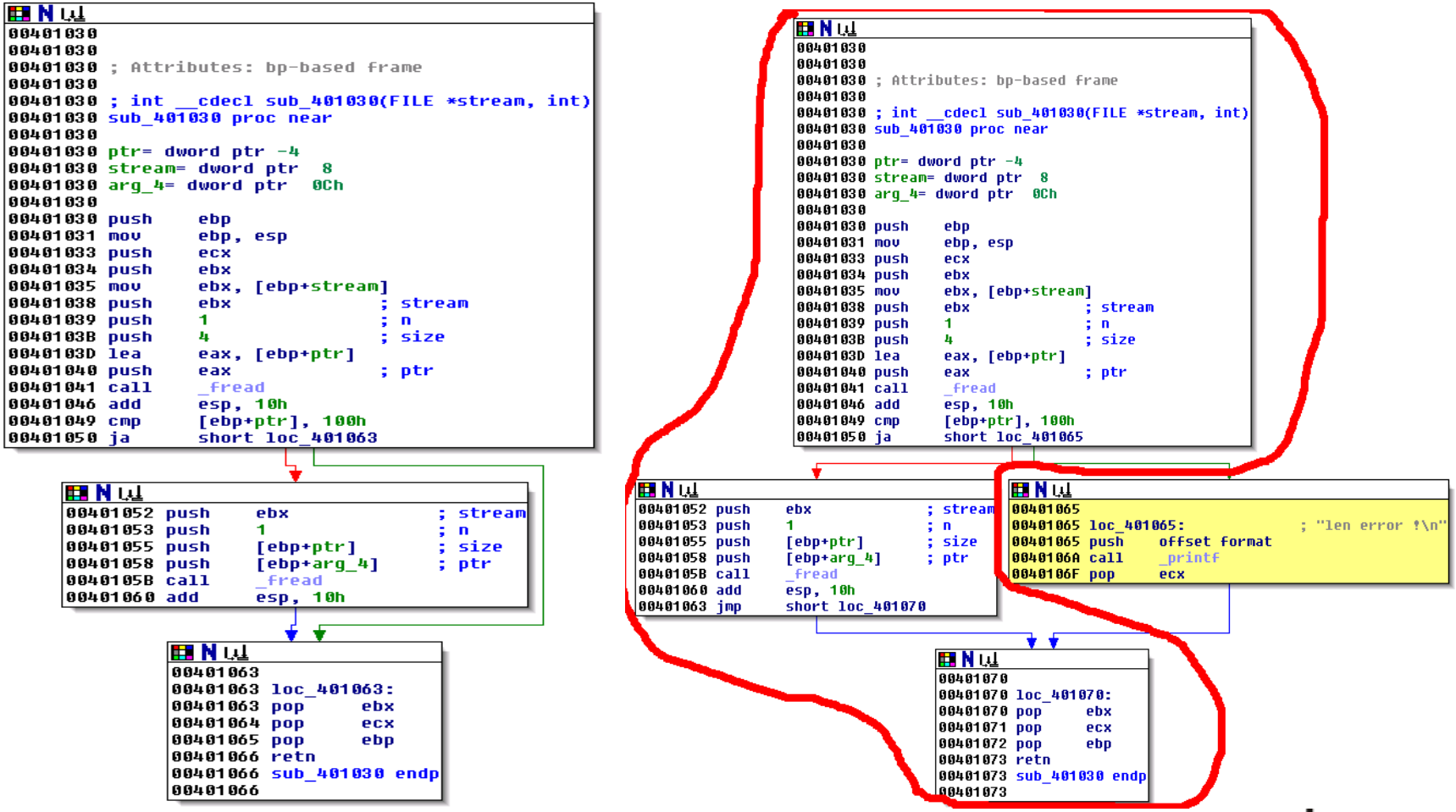
test3.exe



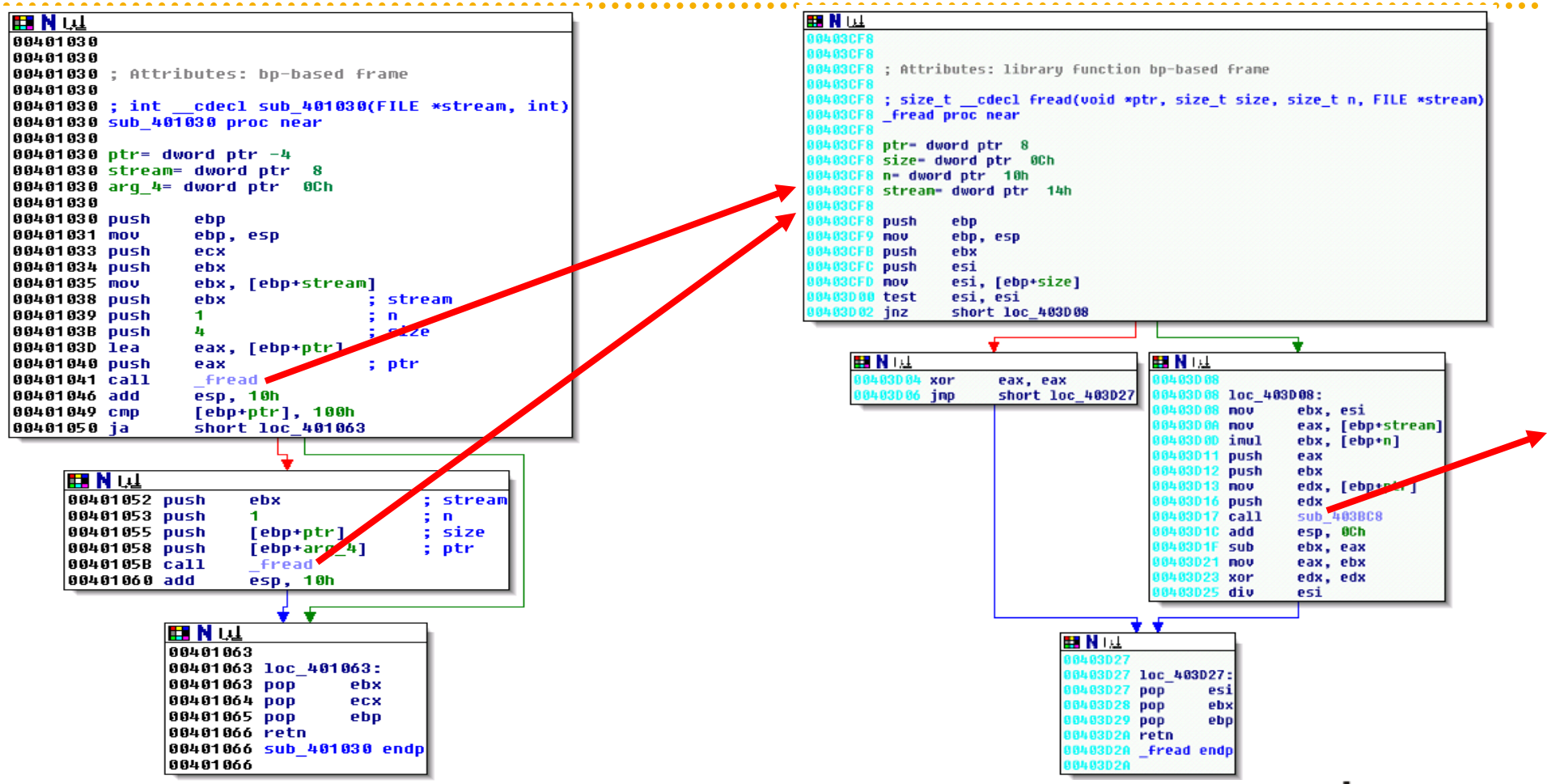
NODE 1 vs NODE 2



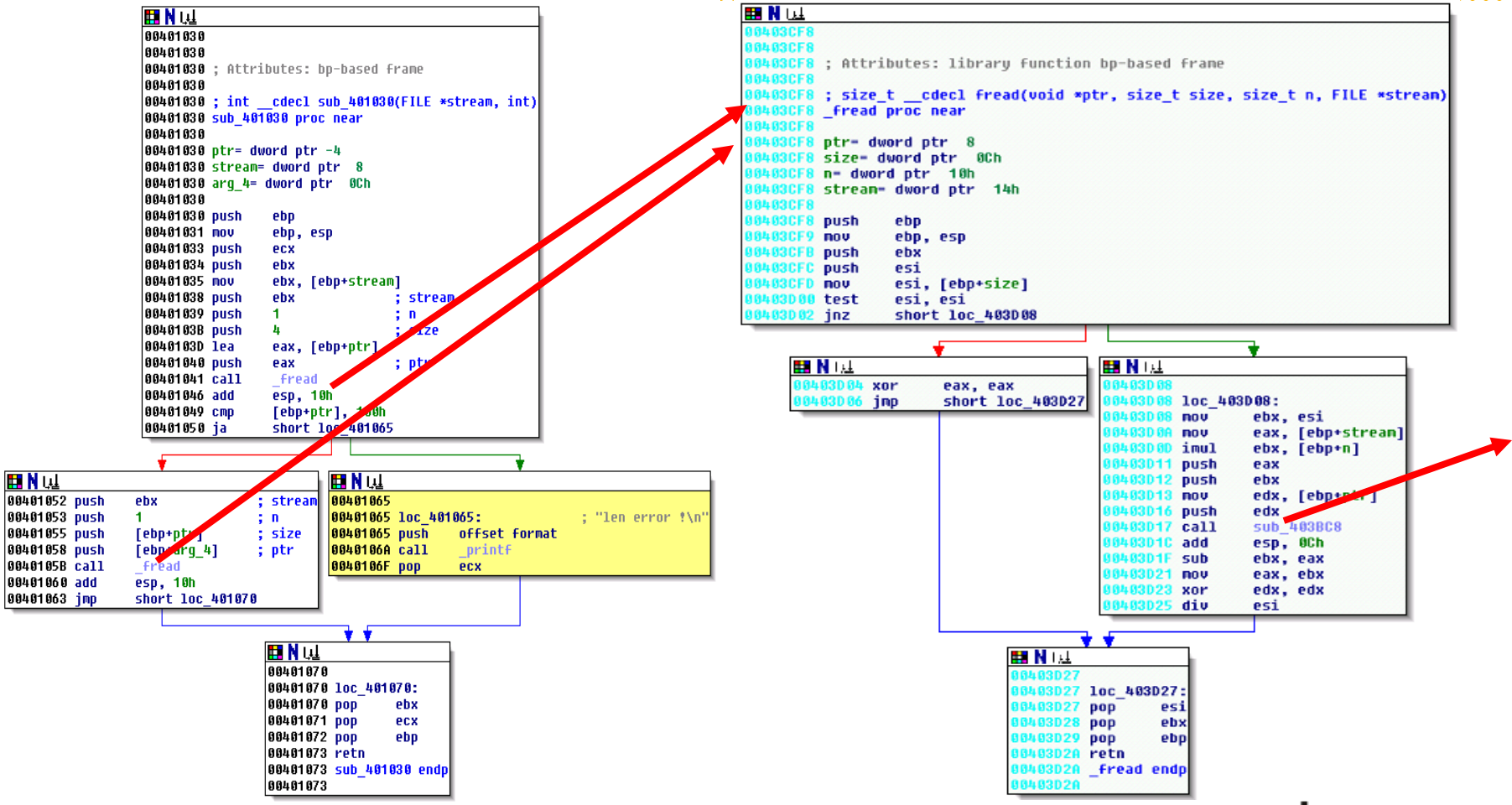
Reliable subgraph



Following the call graph 1 ... →→

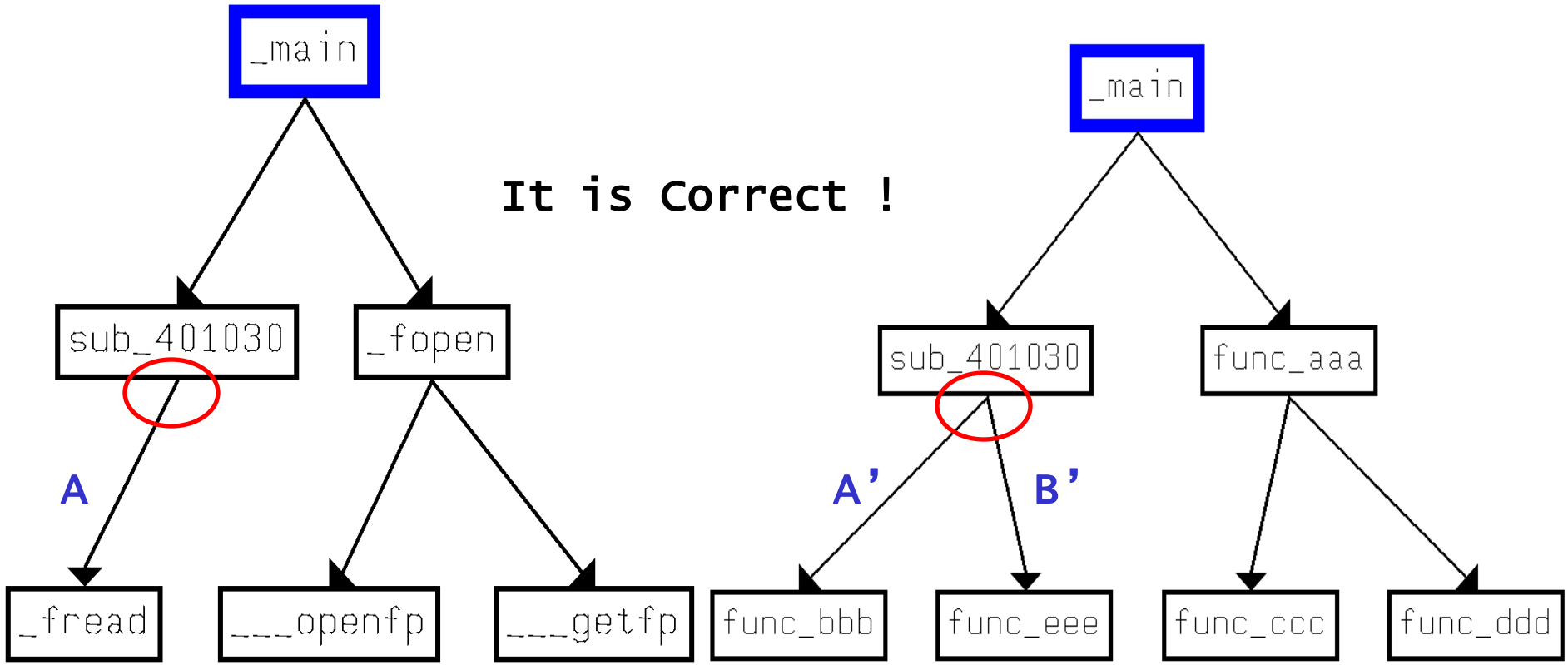


Following the call graph 2 ... →→



The Correct Edges Order

- **test2.exe** **test3.exe**

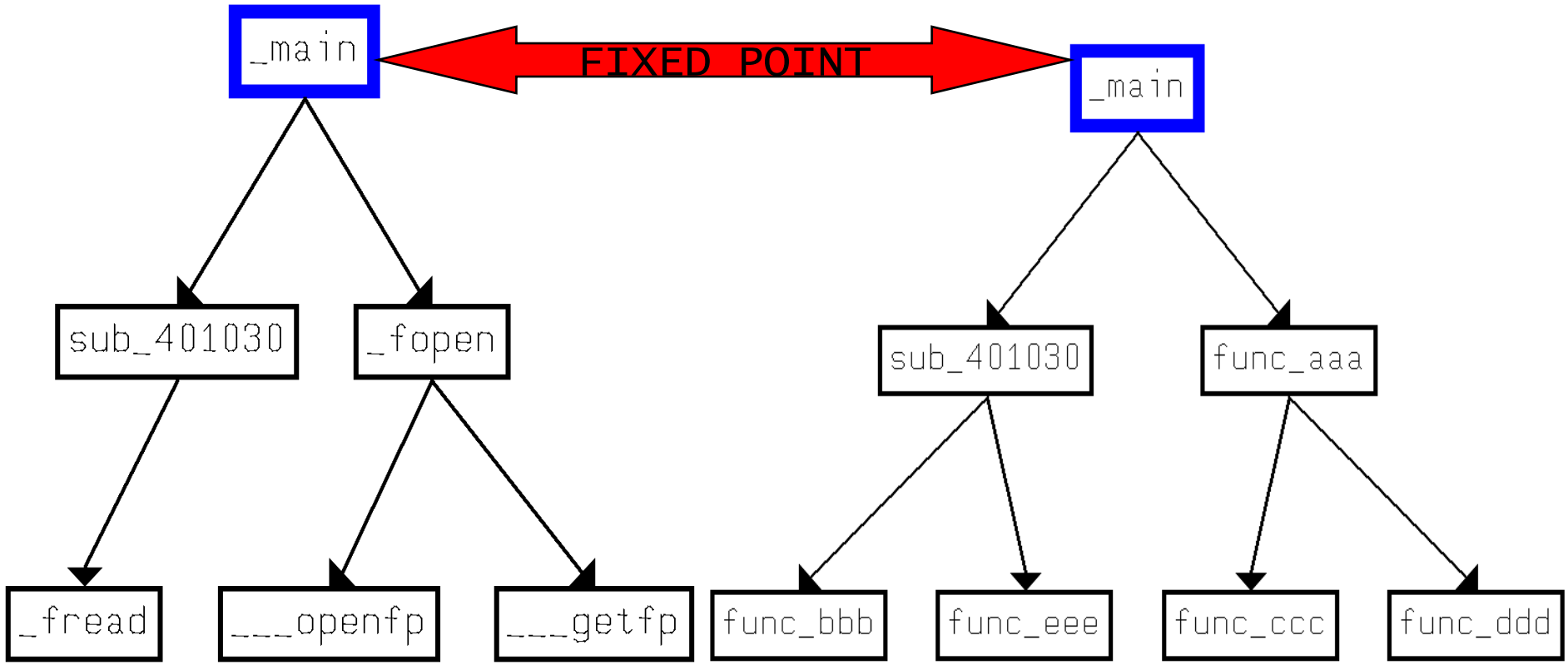


Fixed Points

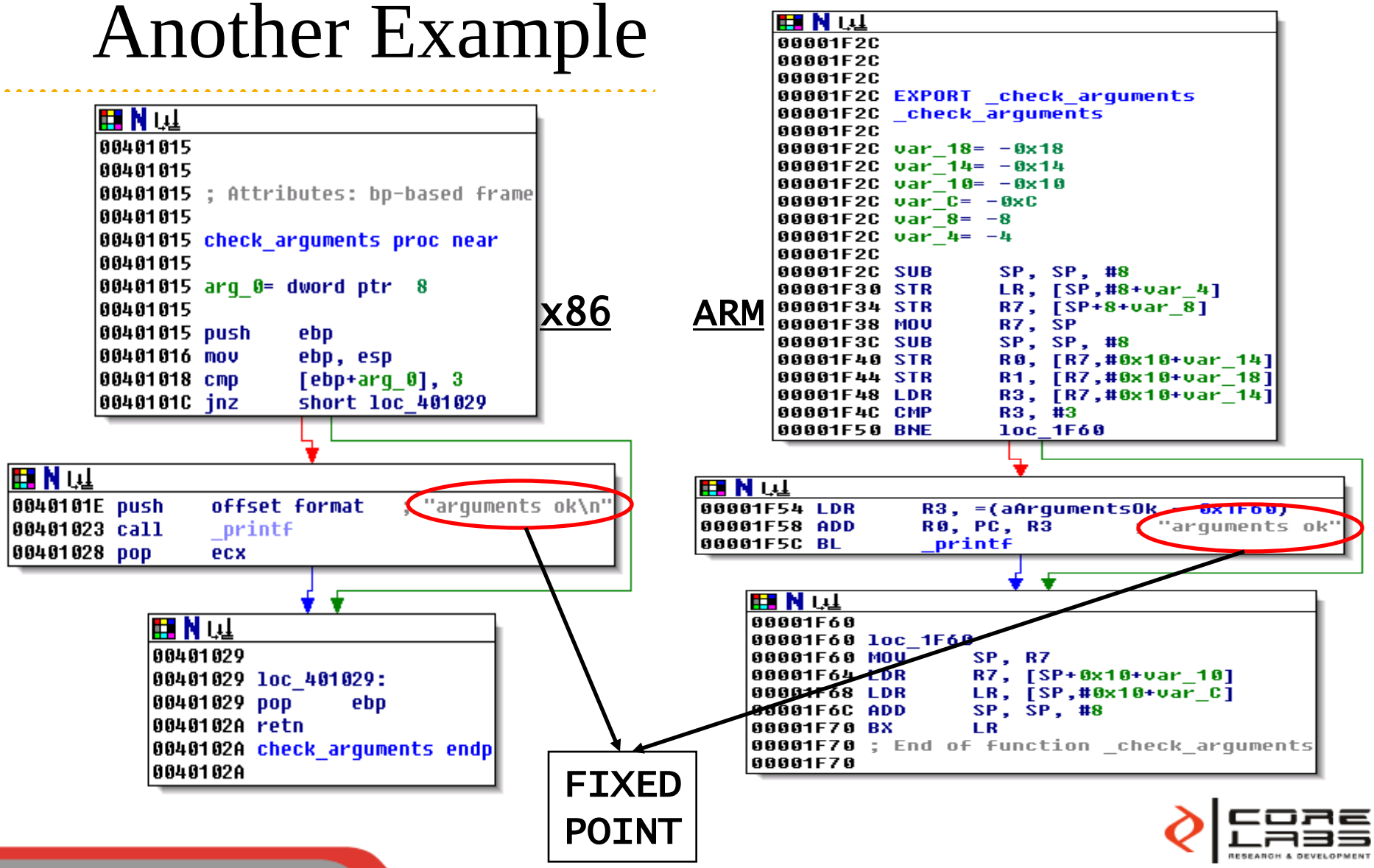
- Reliable points from which we can start to match functions.
- It is the first thing to do.
- If we dont’t have Fixed Points the function matching is more difficult.

Fixed Point Example

- **test2.exe** **test3.exe**



Another Example



Other info we can use...

- **Imported function names (eg.** LoadLibrary **)**
- **Strings (eg.** “arguments ok”**)**
- **Vtables**
- **Global variables**
- **Constants**
- **Etc ...**

Problems

- **Shared Basic Blocks between different functions**
- **Reverted Conditions (** reordered basic blocks TRUE→FALSE, FALSE→ TRUE **)**
- **JUMPs added (**reordered basic blocks**)**
- **Different registers used to represent the same variable**

Detecting Possible Code Theft

- Problems:
  - The differ has to deal with:
    - **Independent Function Positions**
    - **Partial Matches**
    - **Different Compilers**
    - **Different Architectures**

Detecting Possible Code Theft

- Heuristics:
  - Looking for matches
    - **Functions graph (flow chart)**
    - **Partial calls graph ()**
    - **Strings (Very useful)**
  - An expert has to confirm it !!!

The Turbodiff Project

- Researched by Nicolás A. Economou
- Independent Investigation
- Developed on C++
- More than 7300 lines of code (ver 1.01b r1)
- Architecture Independent Diffing
- Oriented to detect changes (for now ...)
- It works best with binaries compiled by the same compiler (for now ...)

Benchmarks

Benchmarks

- **TURBODIFF:**
  - VERSION 1.01 beta release 1
- **MACHINE:**
  - AMD ATHLON 2800+ 1.8 GHz
  - 1 GB RAM
- **SOME TESTS:**
  - TEST1.EXE/TEST2.EXE (PRESENTATION)
  - VMM.SYS (MS09-033)
  - SRV.SYS (MS08-063)
  - EXCEL.EXE (MS09-021)

Benchmarks

- test1.exe/test2.exe
  - 338 functions vs 338 functions

- Results:
  - Elapsed time: 1.5 seconds
  - Match: 335 identical, 1 changed, 2-2 unmatched

Benchmarks

- vmm.sys (Virtual PC) (MS09-033)
  - 552 functions vs 554 functions

- Results:
  - Elapsed time: 2.5 seconds
  - Match: 436 identical, 103 changed, 13-15 unmatched

Benchmarks

- srv.sys (Windows SMB) (MS08-063)
  - 766 functions vs 766 functions

- Results:
  - Elapsed time: 5 seconds
  - Match: 667 identical, 97 changed, 2-2 unmatched

Benchmarks

- excel.exe (MS09-021)
  - 21539 functions vs 21334 functions

- Results:
  - Elapsed time: 210 seconds
  - Match: 20766 identical, 359 changed, 414-209 unmatched.

DEMO: MS09-038

Microsoft Security Bulletin MS09-038 - Critical

Vulnerabilities in Windows Media File Processing Could Allow Remote Code Execution (971557)

Published: August 11, 2009

Vulnerability Information

- ⊞ **Severity Ratings and Vulnerability Identifiers**
- ⊞ **Malformed AVI Header Vulnerability - CVE-2009-1545**
- ⊞ **AVI Integer Overflow Vulnerability - CVE-2009-1546**

Immunity Challenge

- **EkoParty Reverse && GO challenge**
  - http://www.immunityinc.com/contest-es.html
- **Find a bug in a XML Parser**

Questions ?