0-Knowledge Fuzzing

Vincenzo Iozzo

vincenzo.iozzo@zynamics.com

Black Hat
BRIEFINGS AND TRAINING

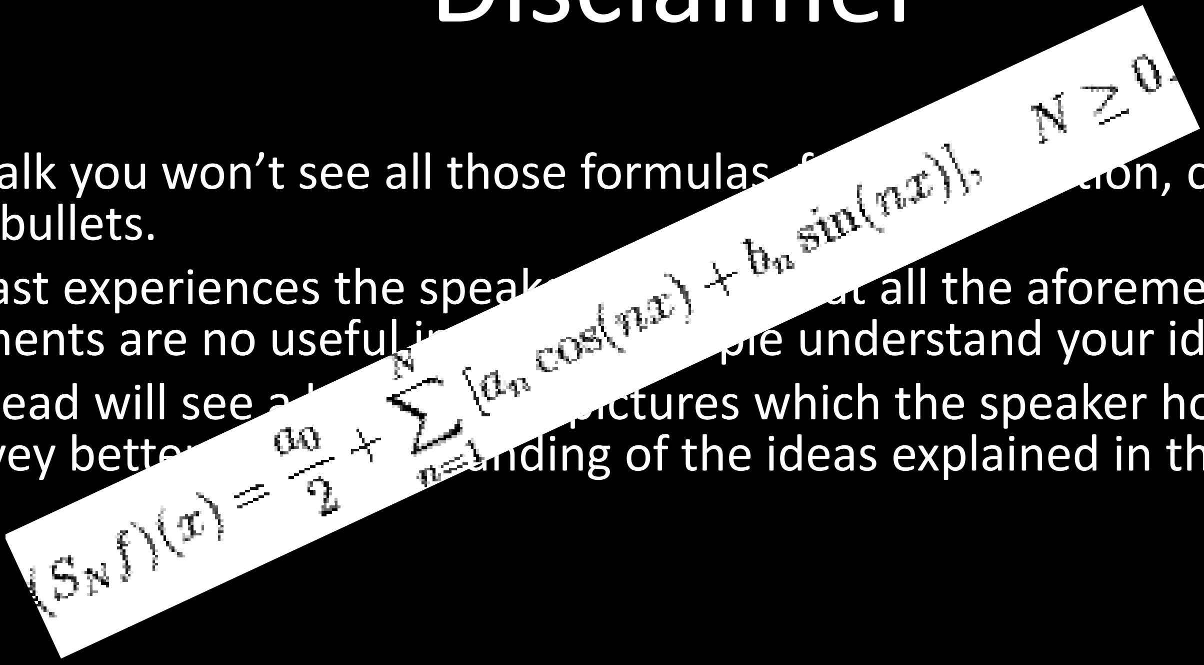
zynamics
www.zynamics.com

Disclaimer

In this talk you won’t see all those formulas, f tion, code snippets and bullets.

From past experiences the speak t all the aforementioned elements are no useful i ple understand your idea.

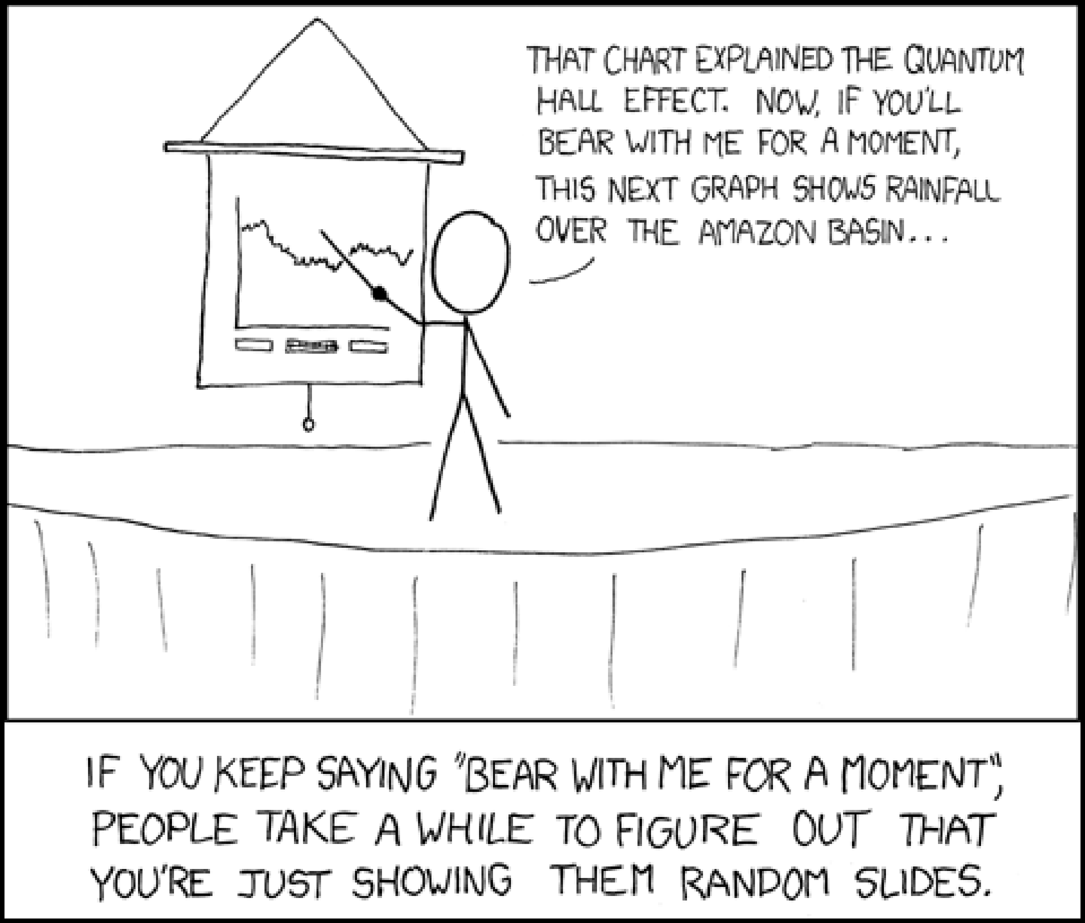
You instead will see a pictures which the speaker hopes will convey bette nding of the ideas explained in the talk



$$(S_N f)(x) = \frac{a_0}{2} + \sum_{n=1}^{N} [a_n \cos(nx) + b_n \sin(nx)], \quad N \geq 0.$$

You don’t want slides like this, do you?

Motivations


THAT CHART EXPLAINED THE QUANTUM HALL EFFECT. NOW, IF YOU'LL BEAR WITH ME FOR A MOMENT, THIS NEXT GRAPH SHOWS RAINFALL OVER THE AMAZON BASIN...
IF YOU KEEP SAYING "BEAR WITH ME FOR A MOMENT", PEOPLE TAKE A WHILE TO FIGURE OUT THAT YOU'RE JUST SHOWING THEM RANDOM SLIDES.

Questions!

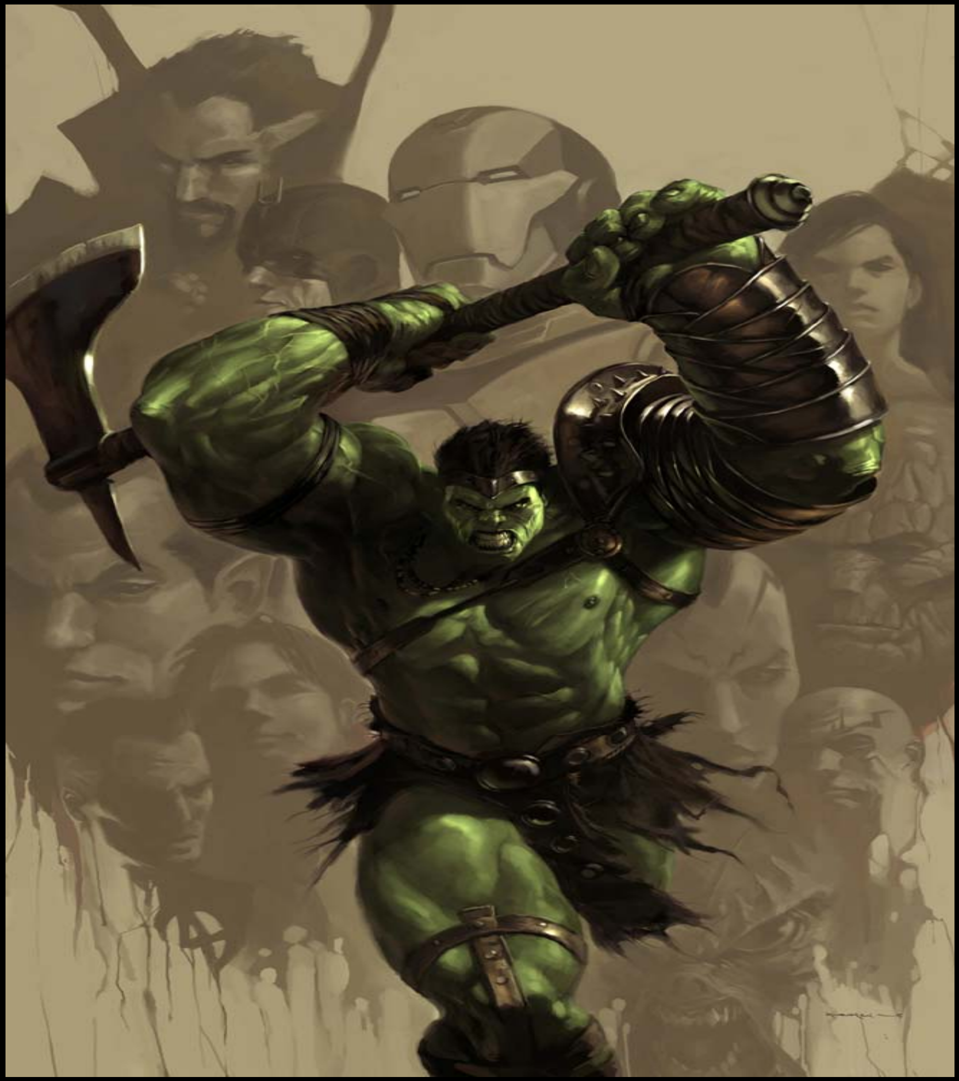
Fuzzing

How it used to be

]

How it is today
(aka the reason of this talk)

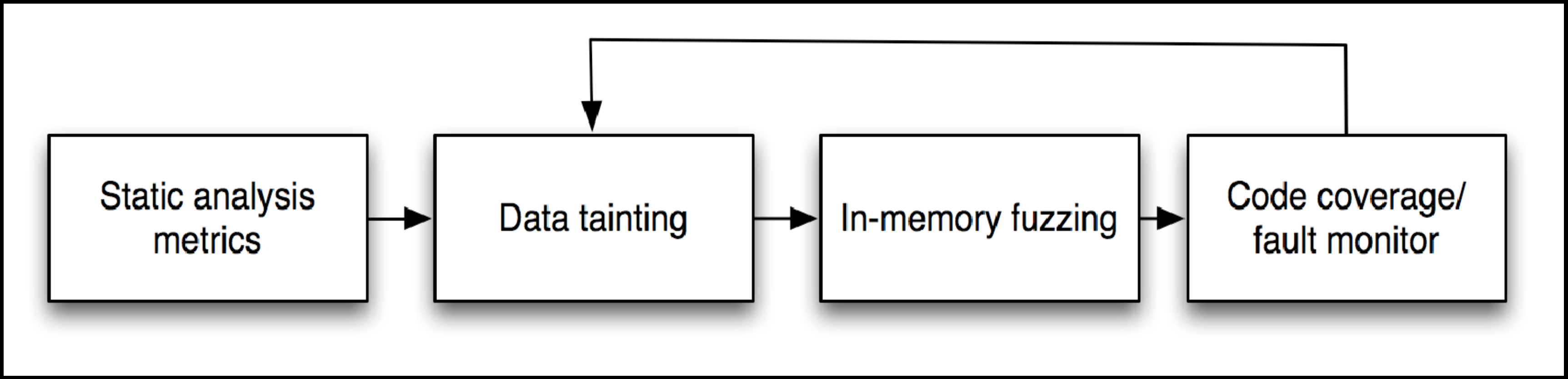
Dumb fuzzing



Smart Fuzzing

Evolutionary Based Fuzzing

The idea

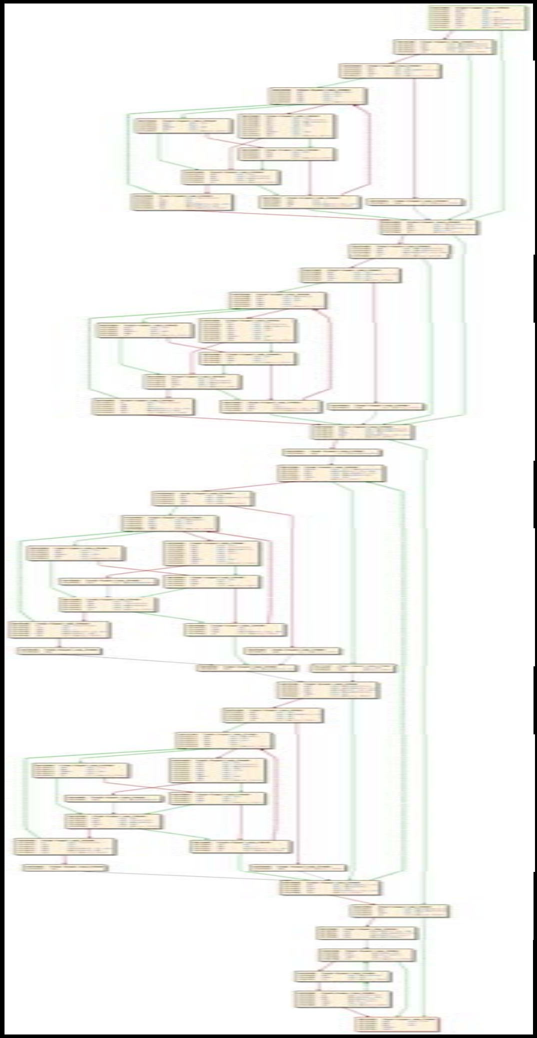

Static analysis metrics
Data tainting
In-memory fuzzing
Code coverage/ fault monitor

The surface

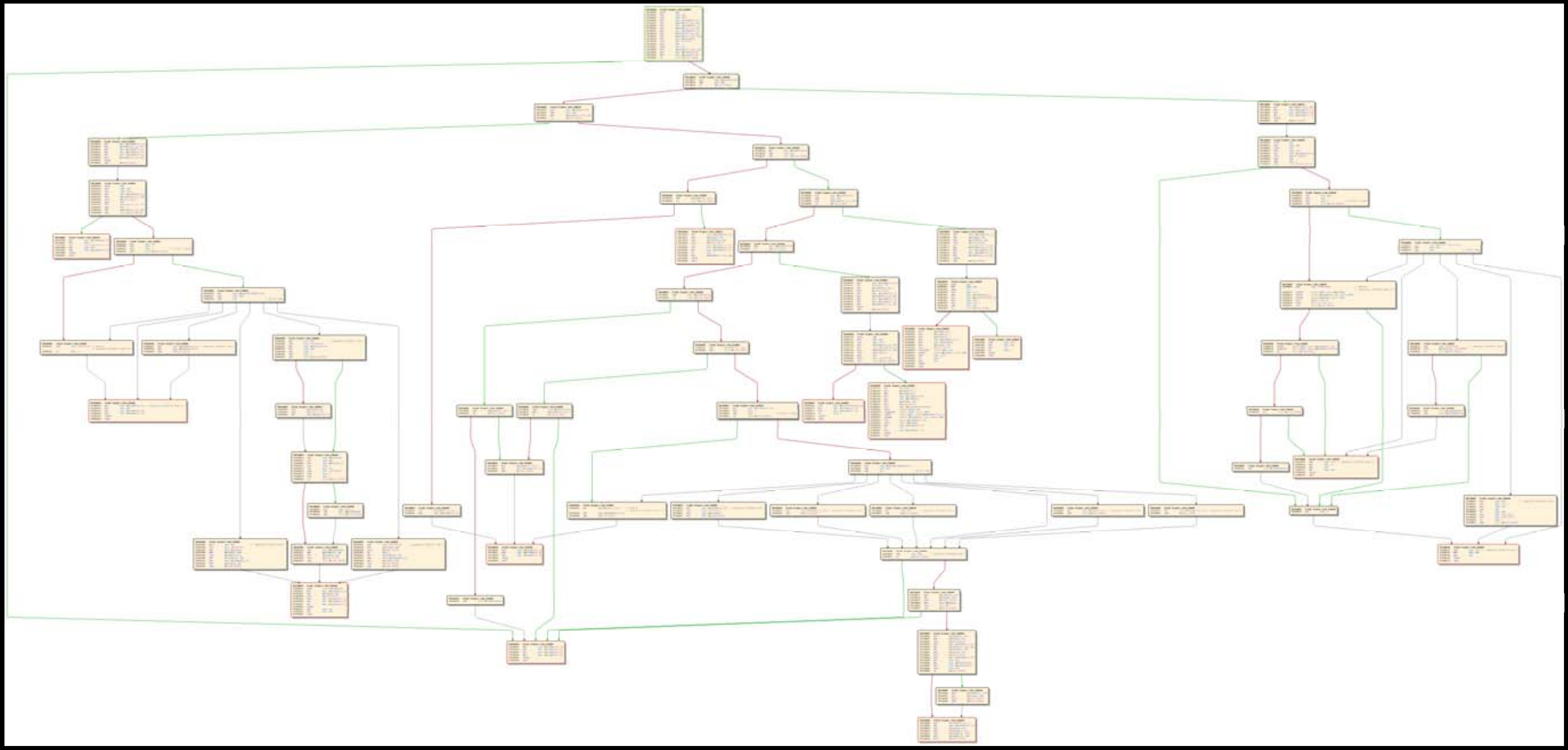
We need a filter

Cyclomatic complexity

This one

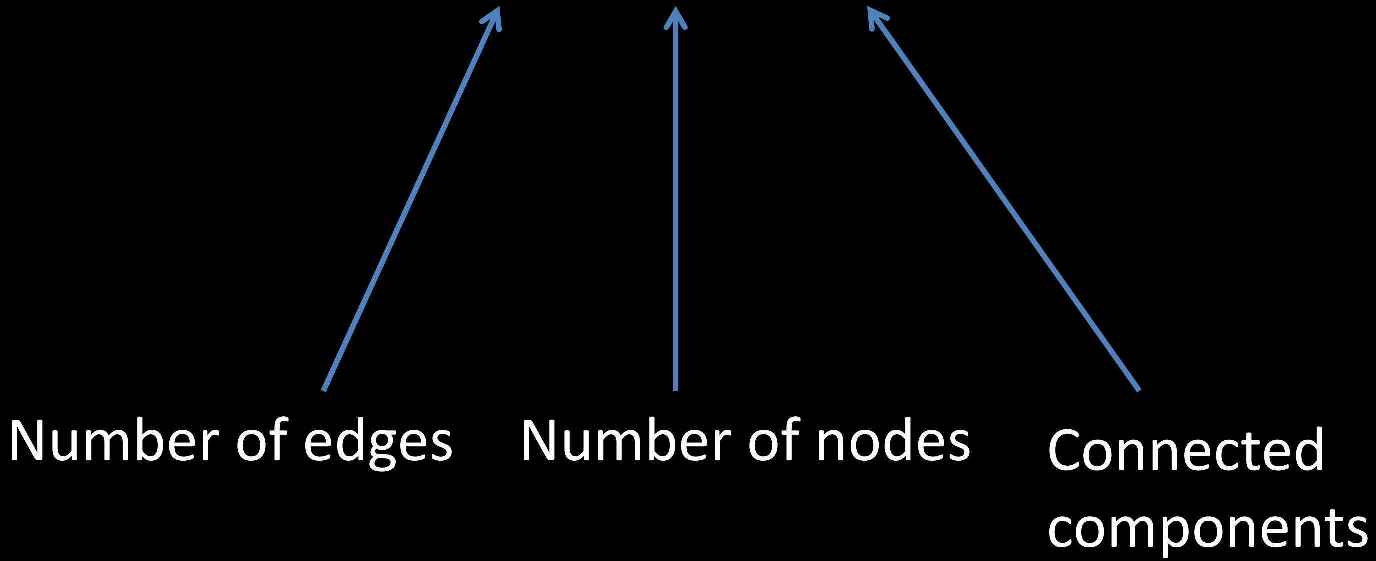


Not this one



Original formula

$$M = E - N + 2P$$



Number of edges

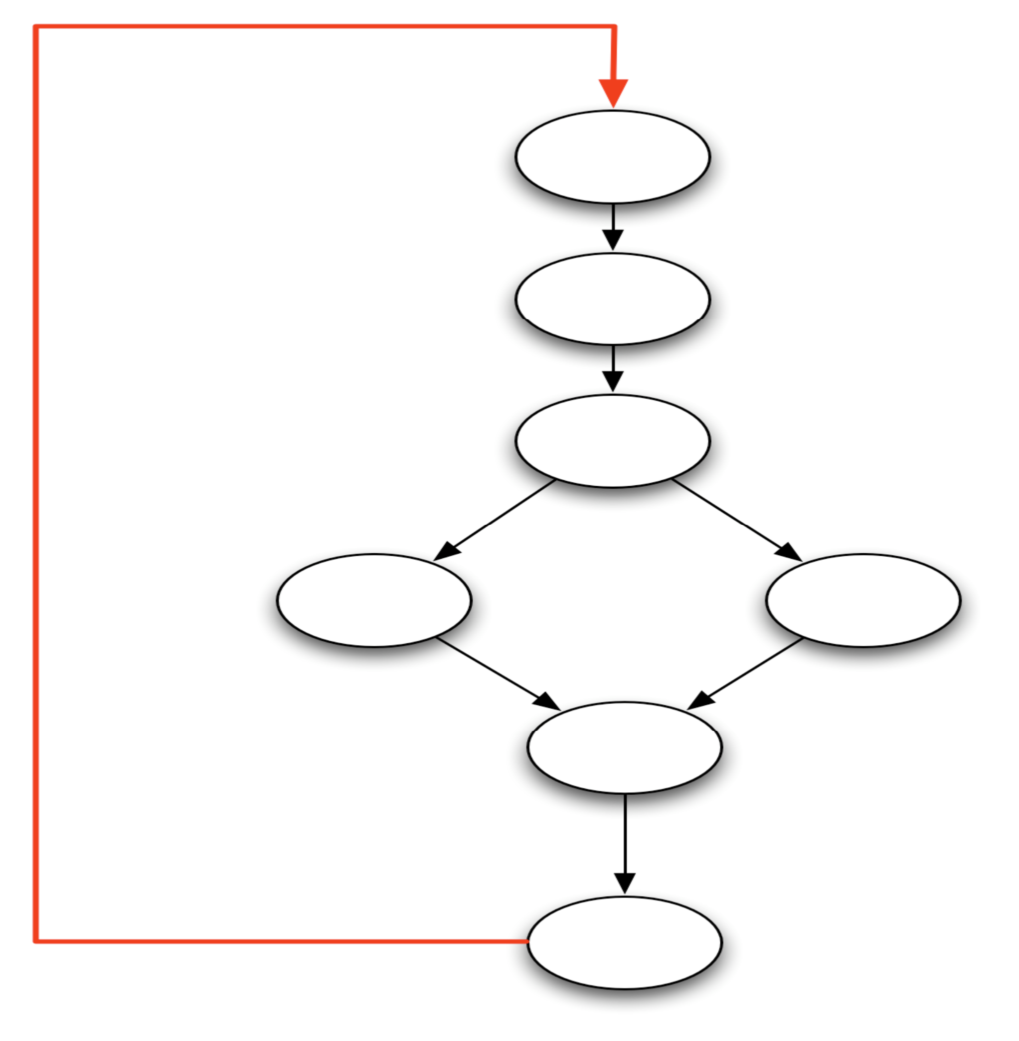
Number of nodes

Connected components

Why? Cyclomatic number

$$M = E - N + P$$

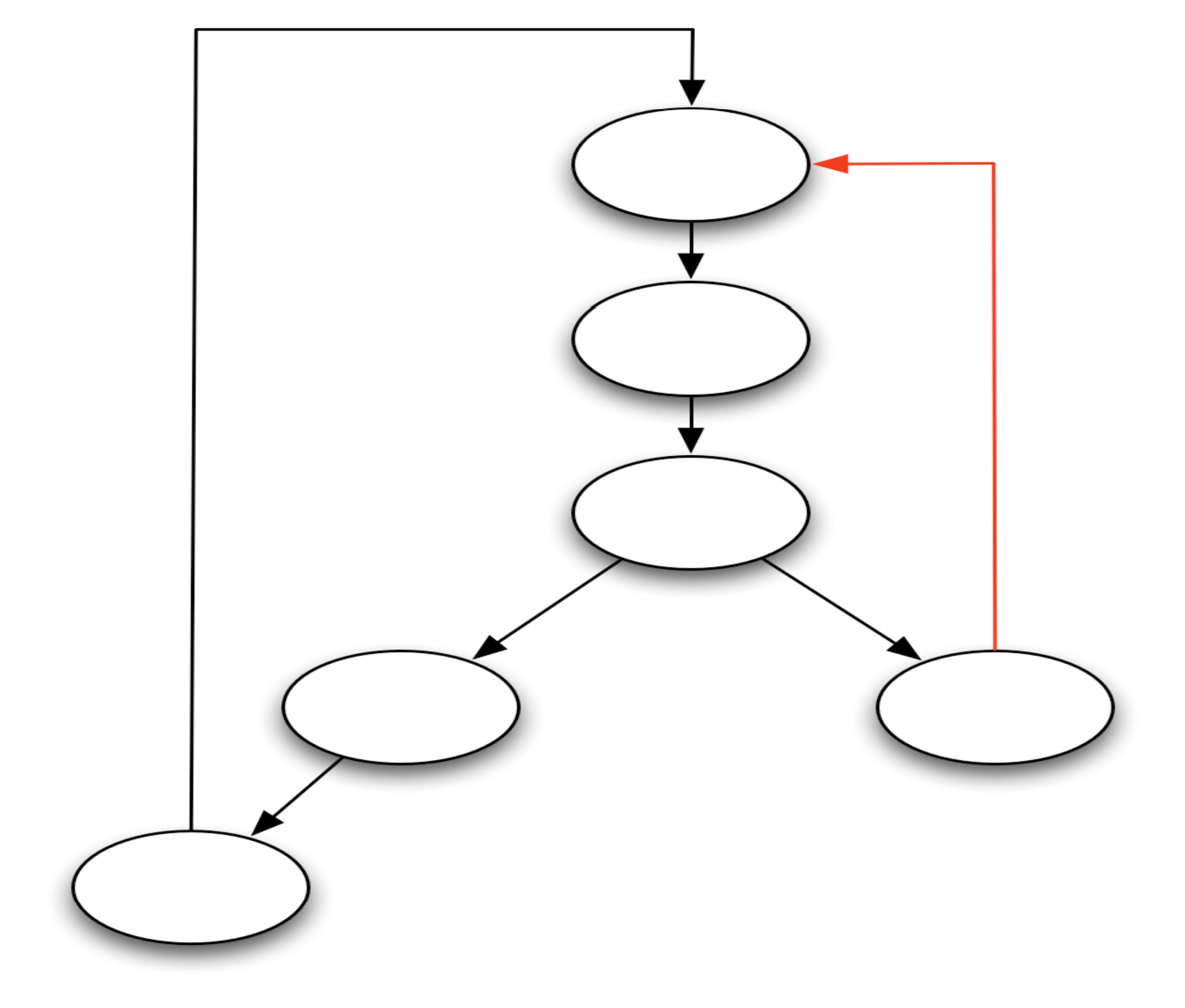
Simplify



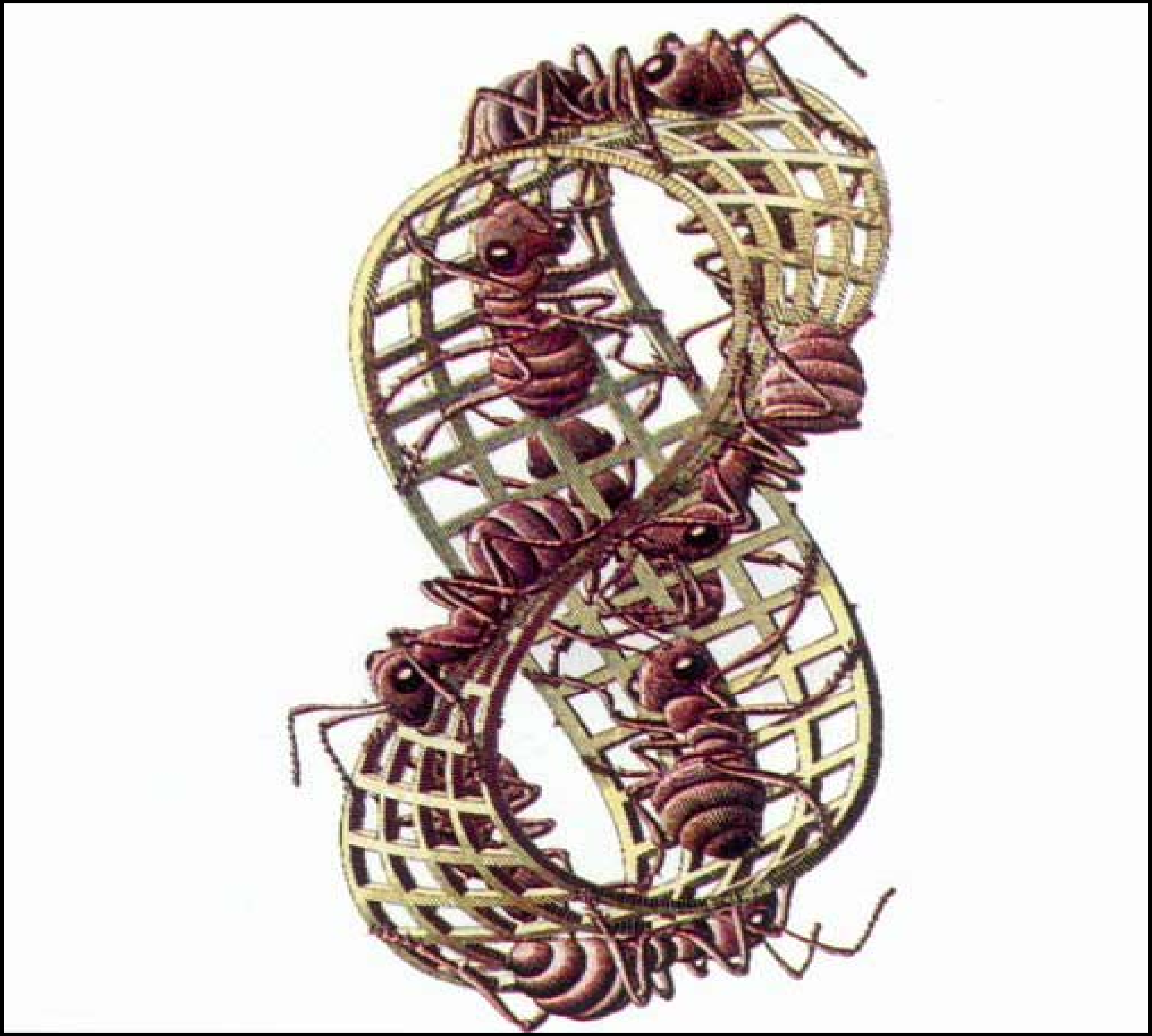
Formula

$$M = E - N + 2$$

Problem

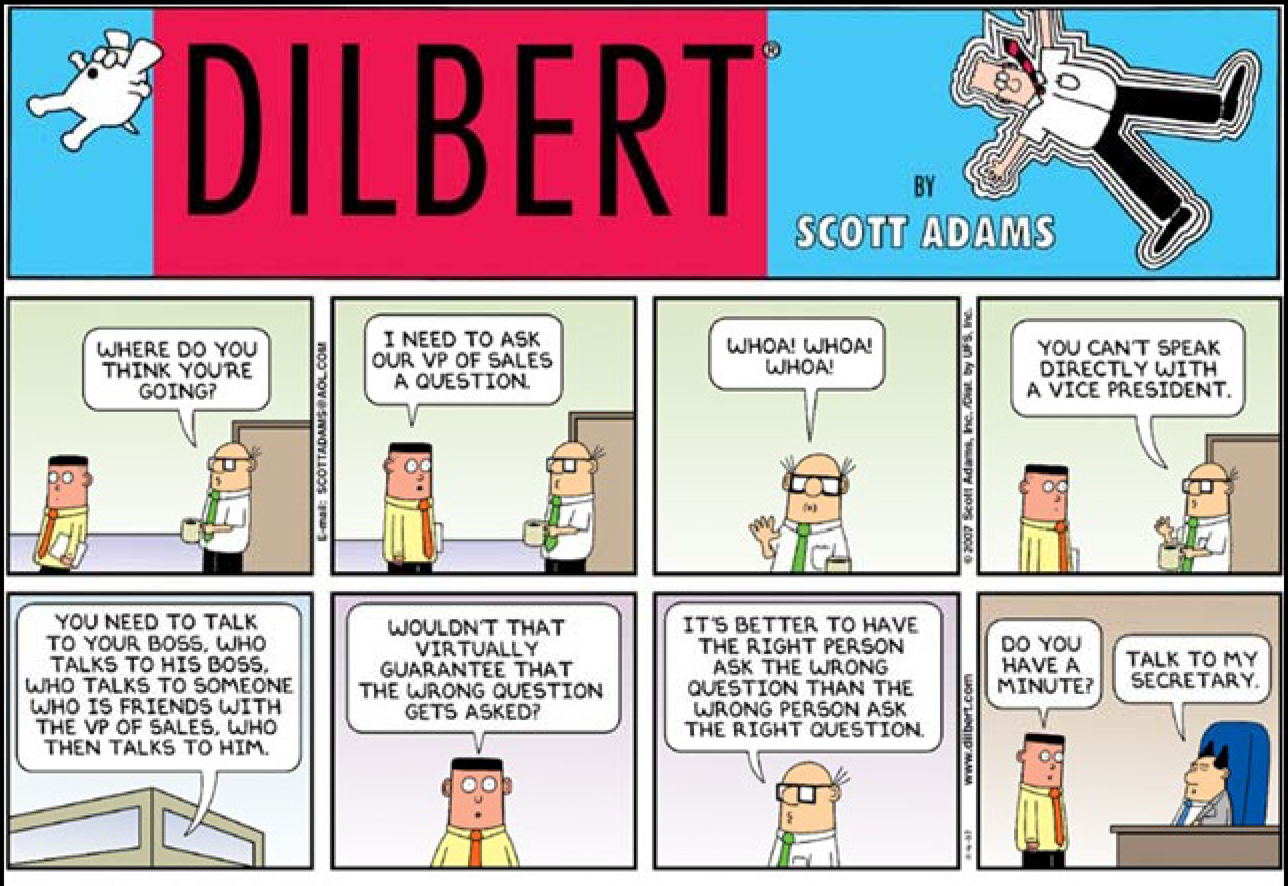


Loop detection



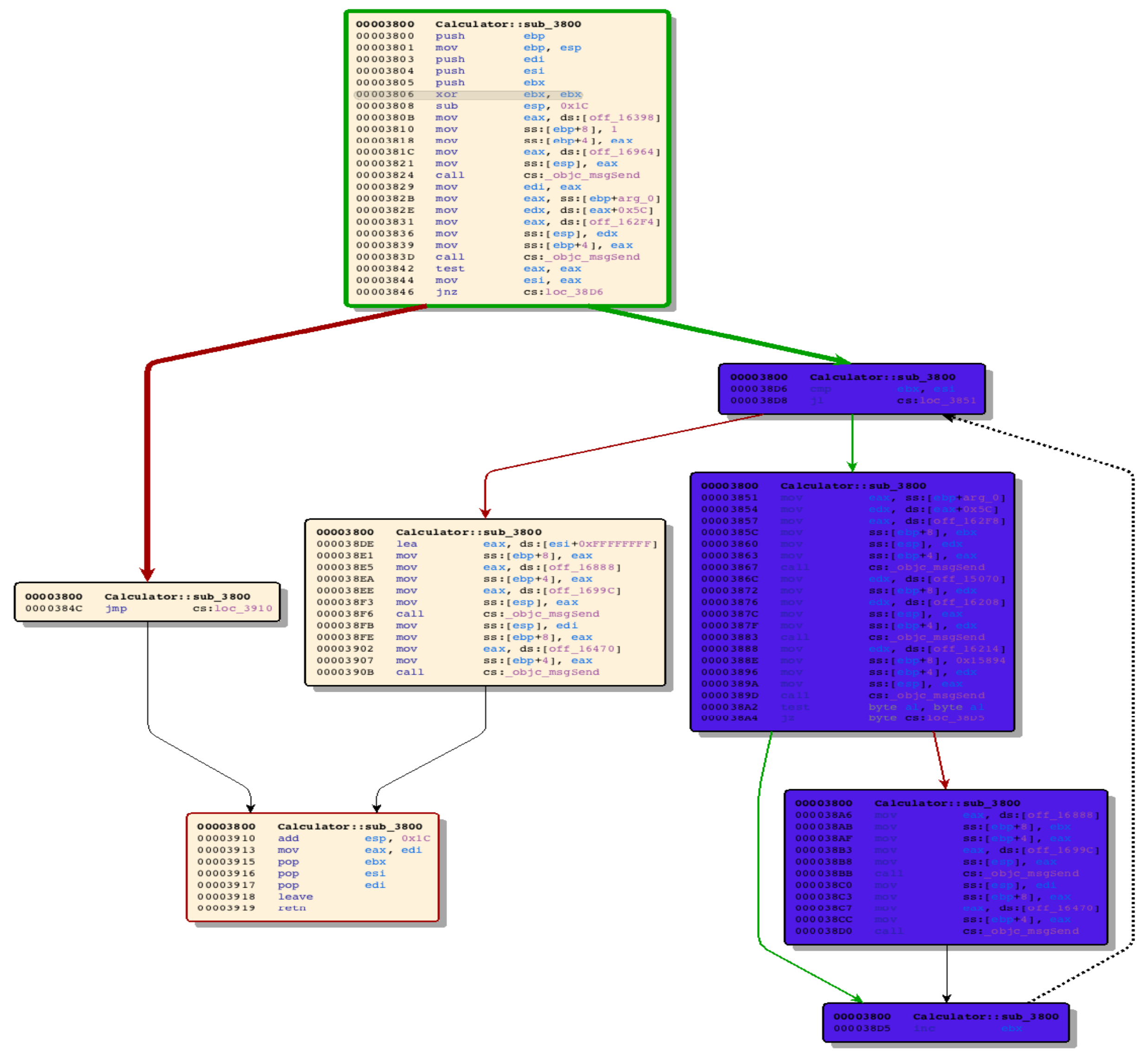
Dominator tree

Dominators


DILBERT®
BY SCOTT ADAMS
WHERE DO YOU THINK YOU'RE GOING?
E-mail: SCOTTADAMS@AOL.COM
I NEED TO ASK OUR VP OF SALES A QUESTION.
WHOA! WHOA! WHOA!
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
YOU CAN'T SPEAK DIRECTLY WITH A VICE PRESIDENT.
YOU NEED TO TALK TO YOUR BOSS, WHO TALKS TO HIS BOSS, WHO TALKS TO SOMEONE WHO IS FRIENDS WITH THE VP OF SALES, WHO THEN TALKS TO HIM.
WOULDN'T THAT VIRTUALLY GUARANTEE THAT THE WRONG QUESTION GETS ASKED?
IT'S BETTER TO HAVE THE RIGHT PERSON ASK THE WRONG QUESTION THAN THE WRONG PERSON ASK THE RIGHT QUESTION.
www.dilbert.com
DO YOU HAVE A MINUTE?
TALK TO MY SECRETARY.

© Scott Adams, Inc./Dist. by UFS, Inc.

Function



```
00003800   Calculator::sub_3800
00003800   push        ebp
00003801   mov         ebp, esp
00003803   push        edi
00003804   push        esi
00003805   push        ebx
00003806   xor         ebx, ebx
00003808   sub         esp, 0x1C
0000380B   mov         eax, ds:[off_16398]
00003810   mov         ss:[ebp+8], 1
00003818   mov         ss:[ebp+4], eax
0000381C   mov         eax, ds:[off_16964]
00003821   mov         ss:[esp], eax
00003824   call        cs:_objc_msgSend
00003829   mov         edi, eax
0000382B   mov         eax, ss:[ebp+arg_0]
0000382E   mov         edx, ds:[eax+0x5C]
00003831   mov         eax, ds:[off_162F4]
00003836   mov         ss:[esp], edx
00003839   mov         ss:[ebp+4], eax
0000383D   call        cs:_objc_msgSend
00003842   test        eax, eax
00003844   mov         esi, eax
00003846   jnz         cs:loc_38D6
```

```
00003800   Calculator::sub_3800
000038D6   cmp            ebx, esi
000038D8   jl             cs:loc_3851
```

```
00003800   Calculator::sub_3800
0000384C   jmp         cs:loc_3910
```

```
00003800   Calculator::sub_3800
000038DE   lea         eax, ds:[esi+0xFFFFFFFF]
000038E1   mov         ss:[ebp+8], eax
000038E5   mov         eax, ds:[off_16888]
000038EA   mov         ss:[ebp+4], eax
000038EE   mov         eax, ds:[off_1699C]
000038F3   mov         ss:[esp], eax
000038F6   call        cs:_objc_msgSend
000038FB   mov         ss:[esp], edi
000038FE   mov         ss:[ebp+8], eax
00003902   mov         eax, ds:[off_16470]
00003907   mov         ss:[ebp+4], eax
0000390B   call        cs:_objc_msgSend
```

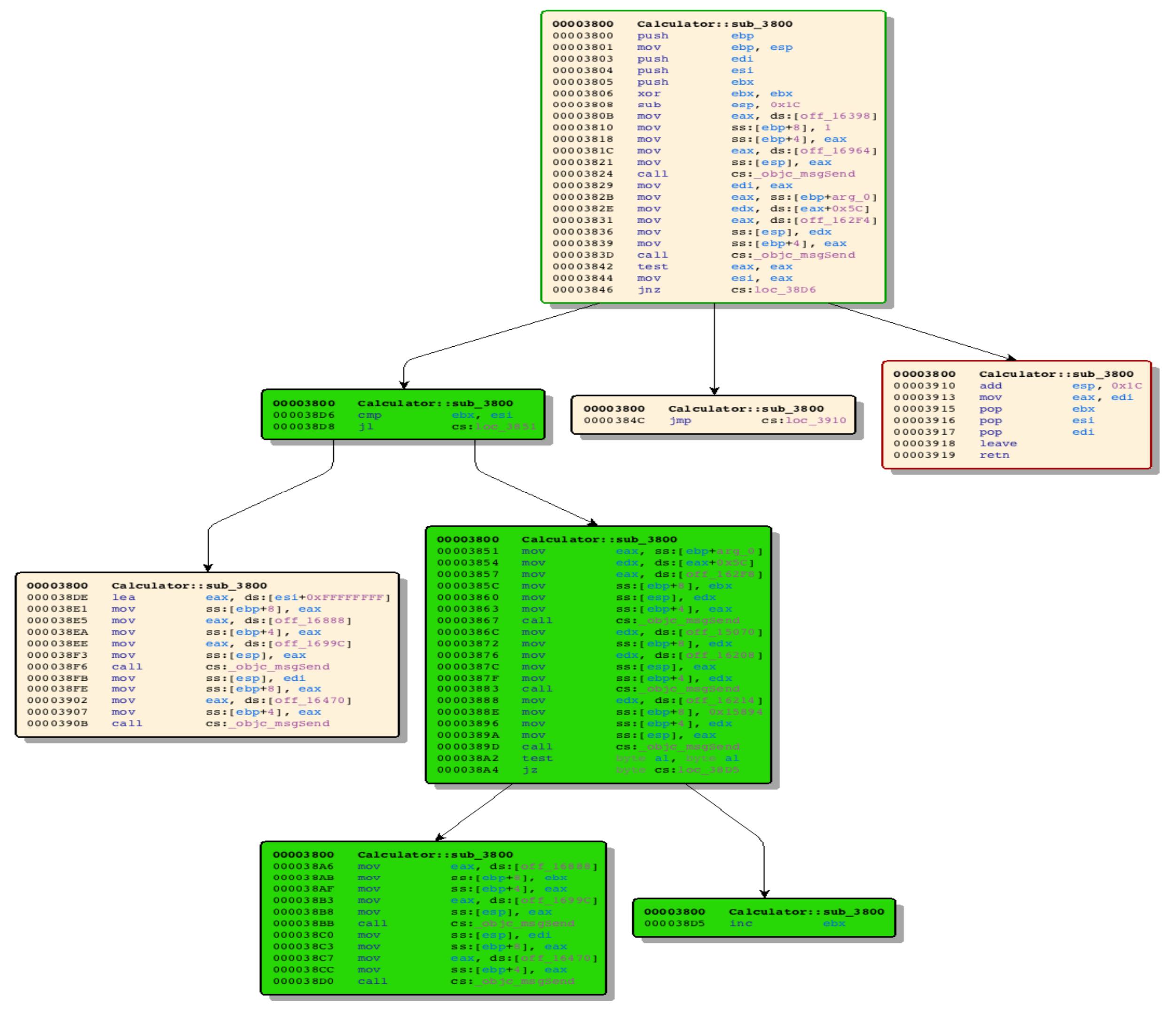
```
00003800   Calculator::sub_3800
00003851   mov         eax, ss:[ebp+arg_0]
00003854   mov         edx, ds:[eax+0x5C]
00003857   mov         eax, ds:[off_162F8]
0000385C   mov         ss:[ebp+8], ebx
00003860   mov         ss:[esp], edx
00003863   mov         ss:[ebp+4], eax
00003867   call        cs:_objc_msgSend
0000386C   mov         edx, ds:[off_15070]
00003872   mov         ss:[ebp+8], edx
00003876   mov         edx, ds:[off_16208]
0000387C   mov         ss:[esp], eax
0000387F   mov         ss:[ebp+4], edx
00003883   call        cs:_objc_msgSend
00003888   mov         edx, ds:[off_16214]
0000388E   mov         ss:[ebp+8], 0x15894
00003896   mov         ss:[ebp+4], edx
0000389A   mov         ss:[esp], eax
0000389D   call        cs:_objc_msgSend
000038A2   test        byte al, byte al
000038A4   jz          byte cs:loc_38D5
```

```
00003800   Calculator::sub_3800
00003910   add         esp, 0x1C
00003913   mov         eax, edi
00003915   pop         ebx
00003916   pop         esi
00003917   pop         edi
00003918   leave
00003919   retn
```

```
00003800   Calculator::sub_3800
000038A6   mov         eax, ds:[off_16888]
000038AB   mov         ss:[ebp+8], ebx
000038AF   mov         ss:[ebp+4], eax
000038B3   mov         eax, ds:[off_1699C]
000038B8   mov         ss:[esp], eax
000038BB   call        cs:_objc_msgSend
000038C0   mov         ss:[esp], edi
000038C3   mov         ss:[ebp+8], eax
000038C7   mov         eax, ds:[off_16470]
000038CC   mov         ss:[ebp+4], eax
000038D0   call        cs:_objc_msgSend
```

```
00003800   Calculator::sub_3800
000038D5   inc            ebx
```

Dominator tree



```
00003800   Calculator::sub_3800
00003800   push        ebp
00003801   mov         ebp, esp
00003803   push        edi
00003804   push        esi
00003805   push        ebx
00003806   xor         ebx, ebx
00003808   sub         esp, 0x1C
0000380B   mov         eax, ds:[off_16398]
00003810   mov         ss:[ebp+8], 1
00003818   mov         ss:[ebp+4], eax
0000381C   mov         eax, ds:[off_16964]
00003821   mov         ss:[esp], eax
00003824   call        cs:_objc_msgSend
00003829   mov         edi, eax
0000382B   mov         eax, ss:[ebp+arg_0]
0000382E   mov         edx, ds:[eax+0x5C]
00003831   mov         eax, ds:[off_162F4]
00003836   mov         ss:[esp], edx
00003839   mov         ss:[ebp+4], eax
0000383D   call        cs:_objc_msgSend
00003842   test        eax, eax
00003844   mov         esi, eax
00003846   jnz         cs:loc_38D6
```

```
00003800   Calculator::sub_3800
000038D6   cmp         ebx, esi
000038D8   jl          cs:loc_3851
```

```
00003800   Calculator::sub_3800
0000384C   jmp         cs:loc_3910
```

```
00003800   Calculator::sub_3800
00003910   add         esp, 0x1C
00003913   mov         eax, edi
00003915   pop         ebx
00003916   pop         esi
00003917   pop         edi
00003918   leave
00003919   retn
```

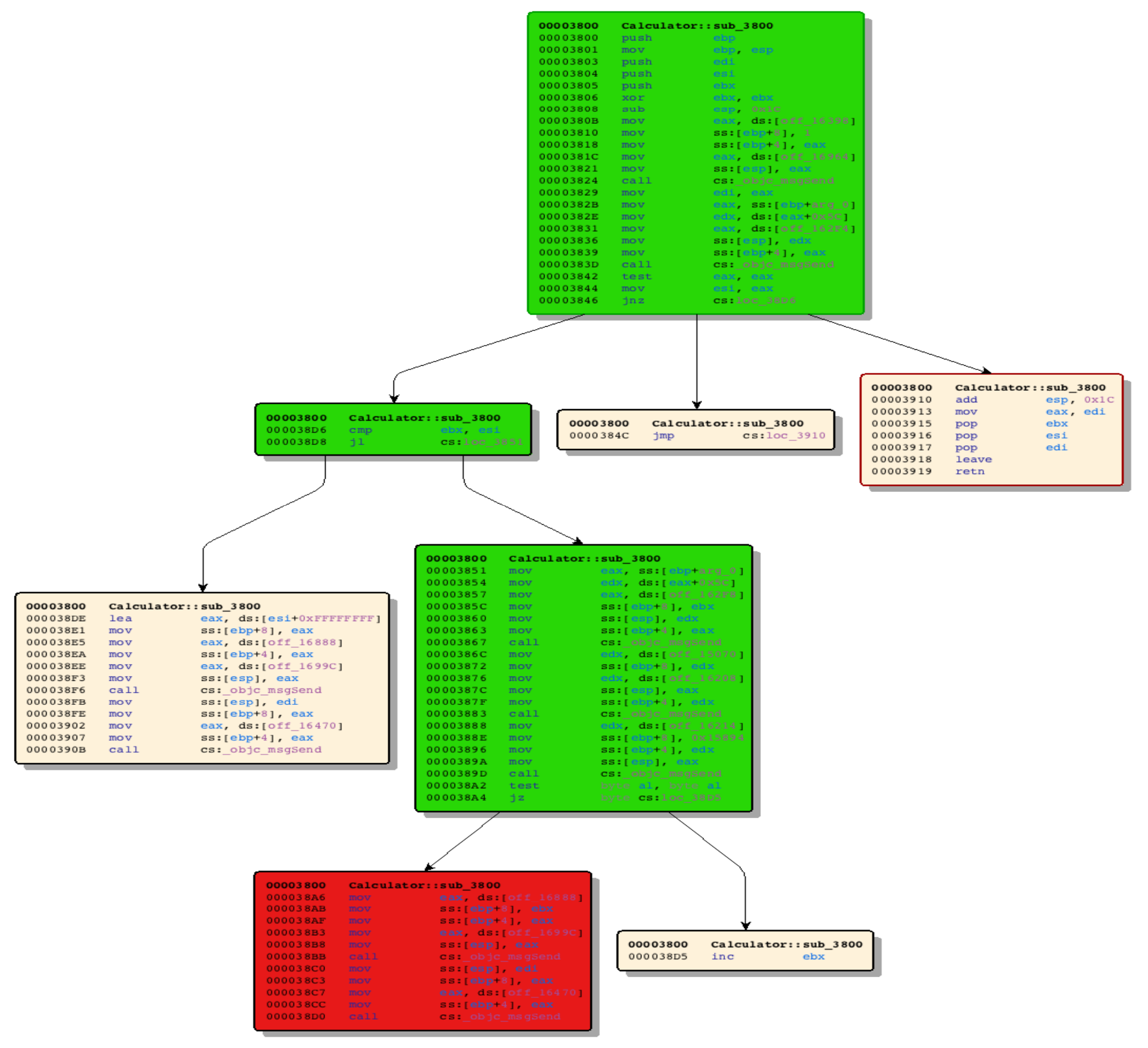
```
00003800   Calculator::sub_3800
000038DE   lea         eax, ds:[esi+0xFFFFFFFF]
000038E1   mov         ss:[ebp+8], eax
000038E5   mov         eax, ds:[off_16888]
000038EA   mov         ss:[ebp+4], eax
000038EE   mov         eax, ds:[off_1699C]
000038F3   mov         ss:[esp], eax
000038F6   call        cs:_objc_msgSend
000038FB   mov         ss:[esp], edi
000038FE   mov         ss:[ebp+8], eax
00003902   mov         eax, ds:[off_16470]
00003907   mov         ss:[ebp+4], eax
0000390B   call        cs:_objc_msgSend
```

```
00003800   Calculator::sub_3800
00003851   mov         eax, ss:[ebp+arg_0]
00003854   mov         edx, ds:[eax+0x5C]
00003857   mov         eax, ds:[off_162F8]
0000385C   mov         ss:[ebp+8], ebx
00003860   mov         ss:[esp], edx
00003863   mov         ss:[ebp+4], eax
00003867   call        cs:_objc_msgSend
0000386C   mov         edx, ds:[off_15070]
00003872   mov         ss:[ebp+8], edx
00003876   mov         edx, ds:[off_16208]
0000387C   mov         ss:[esp], eax
0000387F   mov         ss:[ebp+4], edx
00003883   call        cs:_objc_msgSend
00003888   mov         edx, ds:[off_16214]
0000388E   mov         ss:[ebp+8], 0x15894
00003896   mov         ss:[ebp+4], edx
0000389A   mov         ss:[esp], eax
0000389D   call        cs:_objc_msgSend
000038A2   test        byte al, byte al
000038A4   jz          byte cs:loc_38D5
```

```
00003800   Calculator::sub_3800
000038A6   mov         eax, ds:[off_16888]
000038AB   mov         ss:[ebp+8], ebx
000038AF   mov         ss:[ebp+4], eax
000038B3   mov         eax, ds:[off_1699C]
000038B8   mov         ss:[esp], eax
000038BB   call        cs:_objc_msgSend
000038C0   mov         ss:[esp], edi
000038C3   mov         ss:[ebp+8], eax
000038C7   mov         eax, ds:[off_16470]
000038CC   mov         ss:[ebp+4], eax
000038D0   call        cs:_objc_msgSend
```

```
00003800   Calculator::sub_3800
000038D5   inc         ebx
```

Dominators



```
00003800   Calculator::sub_3800
00003800   push        ebp
00003801   mov         ebp, esp
00003803   push        edi
00003804   push        esi
00003805   push        ebx
00003806   xor         ebx, ebx
00003808   sub         esp, 0x1C
0000380B   mov         eax, ds:[off_16398]
00003810   mov         ss:[ebp+8], 1
00003818   mov         ss:[ebp+4], eax
0000381C   mov         eax, ds:[off_16964]
00003821   mov         ss:[esp], eax
00003824   call        cs:_objc_msgSend
00003829   mov         edi, eax
0000382B   mov         eax, ss:[ebp+arg_0]
0000382E   mov         edx, ds:[eax+0x5C]
00003831   mov         eax, ds:[off_162F4]
00003836   mov         ss:[esp], edx
00003839   mov         ss:[ebp+4], eax
0000383D   call        cs:_objc_msgSend
00003842   test        eax, eax
00003844   mov         esi, eax
00003846   jnz         cs:loc_38D6
```

```
00003800   Calculator::sub_3800
000038D6   cmp         ebx, esi
000038D8   jl          cs:loc_3851
```

```
00003800   Calculator::sub_3800
0000384C   jmp         cs:loc_3910
```

```
00003800   Calculator::sub_3800
00003910   add         esp, 0x1C
00003913   mov         eax, edi
00003915   pop         ebx
00003916   pop         esi
00003917   pop         edi
00003918   leave
00003919   retn
```

```
00003800   Calculator::sub_3800
000038DE   lea         eax, ds:[esi+0xFFFFFFFF]
000038E1   mov         ss:[ebp+8], eax
000038E5   mov         eax, ds:[off_16888]
000038EA   mov         ss:[ebp+4], eax
000038EE   mov         eax, ds:[off_1699C]
000038F3   mov         ss:[esp], eax
000038F6   call        cs:_objc_msgSend
000038FB   mov         ss:[esp], edi
000038FE   mov         ss:[ebp+8], eax
00003902   mov         eax, ds:[off_16470]
00003907   mov         ss:[ebp+4], eax
0000390B   call        cs:_objc_msgSend
```

```
00003800   Calculator::sub_3800
00003851   mov         eax, ss:[ebp+arg_0]
00003854   mov         edx, ds:[eax+0x5C]
00003857   mov         eax, ds:[off_162F8]
0000385C   mov         ss:[ebp+8], ebx
00003860   mov         ss:[esp], edx
00003863   mov         ss:[ebp+4], eax
00003867   call        cs:_objc_msgSend
0000386C   mov         edx, ds:[off_15070]
00003872   mov         ss:[ebp+8], edx
00003876   mov         edx, ds:[off_16208]
0000387C   mov         ss:[esp], eax
0000387F   mov         ss:[ebp+4], edx
00003883   call        cs:_objc_msgSend
00003888   mov         edx, ds:[off_16214]
0000388E   mov         ss:[ebp+8], 0x15894
00003896   mov         ss:[ebp+4], edx
0000389A   mov         ss:[esp], eax
0000389D   call        cs:_objc_msgSend
000038A2   test        byte al, byte al
000038A4   jz          byte cs:loc_38D5
```

```
00003800   Calculator::sub_3800
000038A6   mov         eax, ds:[off_16888]
000038AB   mov         ss:[ebp+8], ebx
000038AF   mov         ss:[ebp+4], eax
000038B3   mov         eax, ds:[off_1699C]
000038B8   mov         ss:[esp], eax
000038BB   call        cs:_objc_msgSend
000038C0   mov         ss:[esp], edi
000038C3   mov         ss:[ebp+8], eax
000038C7   mov         eax, ds:[off_16470]
000038CC   mov         ss:[ebp+4], eax
000038D0   call        cs:_objc_msgSend
```

```
00003800   Calculator::sub_3800
000038D5   inc         ebx
```

Implicit loops

REIL

Is that enough?

I DON'T THINK I HAVE YOUR FULL ATTENTION.
www.dilbert.com
scottadams@aol.com
IT'S ASOK'S TURN TO LISTEN. IF YOU SAY ANYTHING USEFUL, HE'LL SEND US AN INSTANT MESSAGE.
11-3-08 © 2008 Scott Adams, Inc./Dist. by UFS, Inc.
HE'S ASLEEP.
HE'S EMPLOYING HEURISTICS.
zzz

Not enough

Of course not, more heuristics needed

```
void *safe_strcpy(void *old_dest, void *src, int size){

    void *dst = realloc(old_dest, size +1);
    strncpy(dst, src, size);
    return dst;
}
```

Add your own

For static analysis we use zynamics www.zynamics.com BinNavi graph visualization

DEMO

Questions!

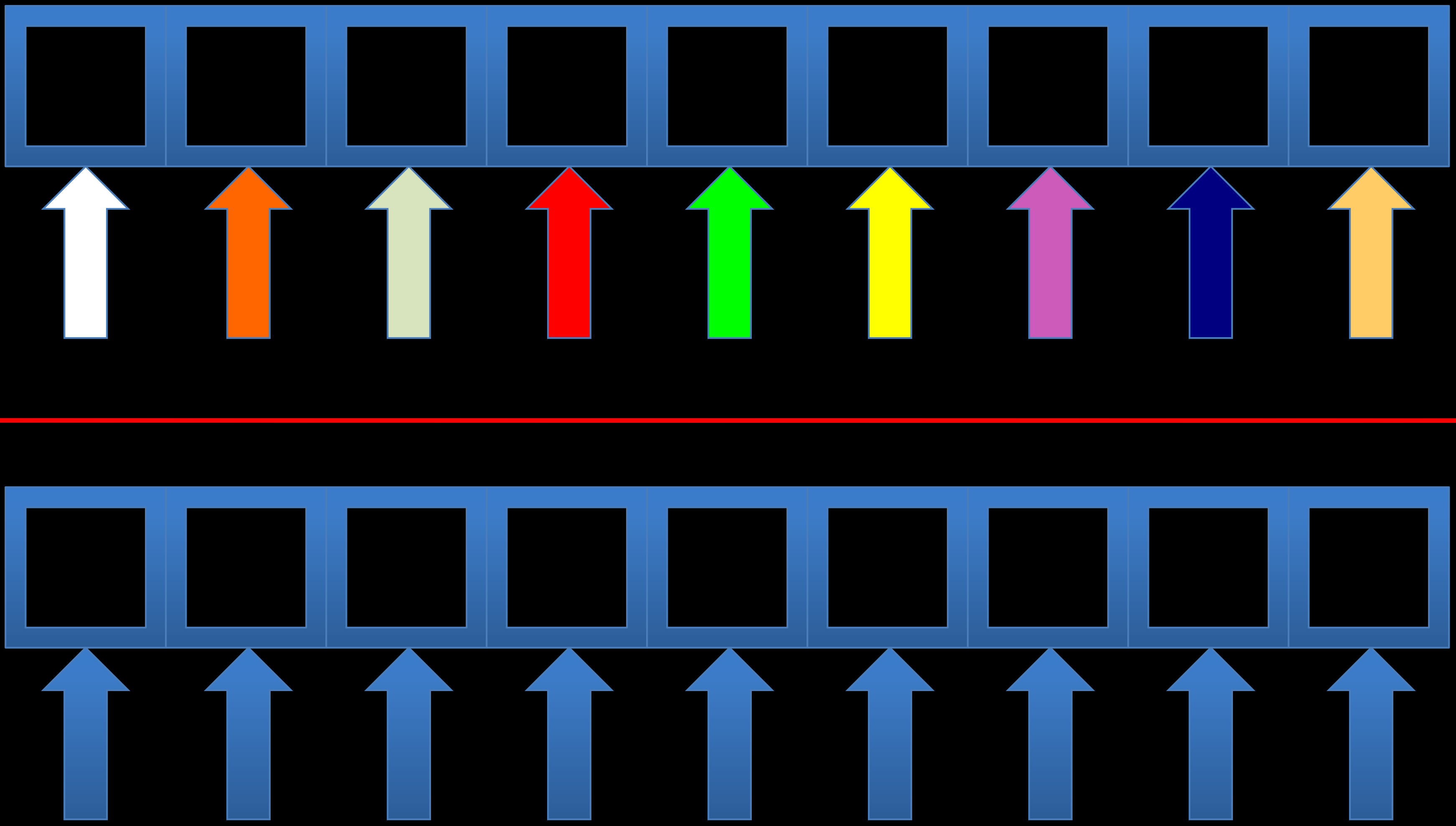
Data Tainting

Dytan

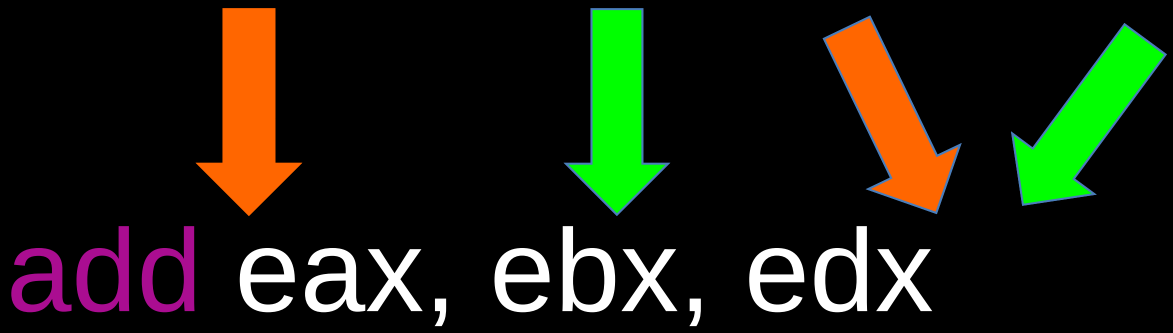
PIN

Taint sources

Markings granularity



Propagation



add eax, ebx, edx

Output

Registers

Memory locations

DEMO

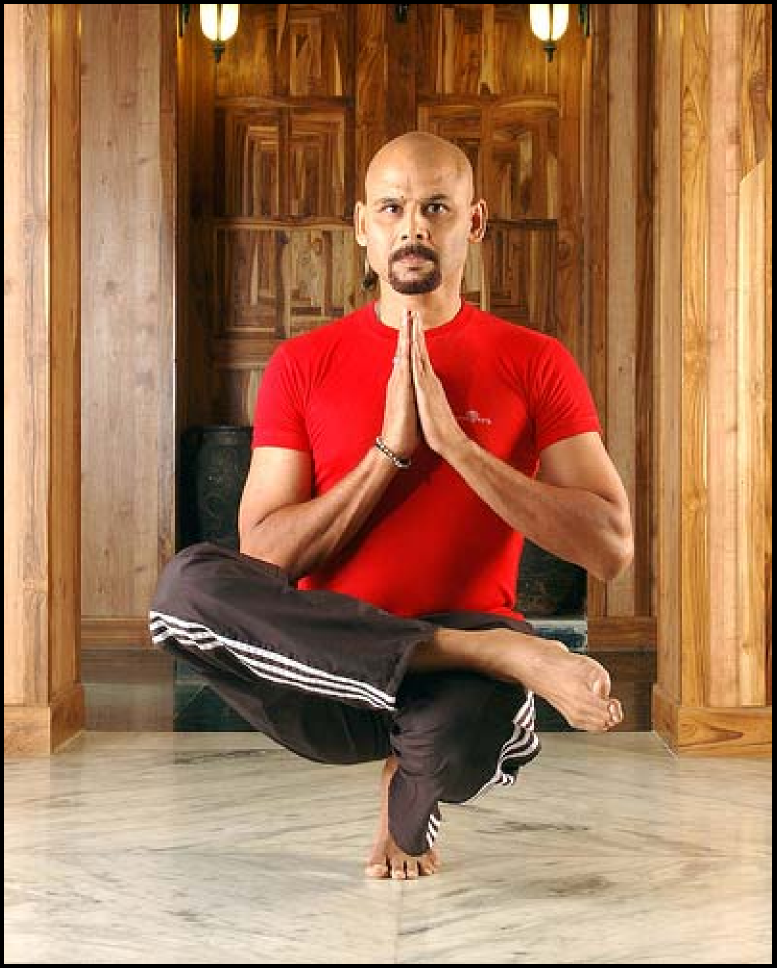
Questions!

In-memory fuzzing

Why?

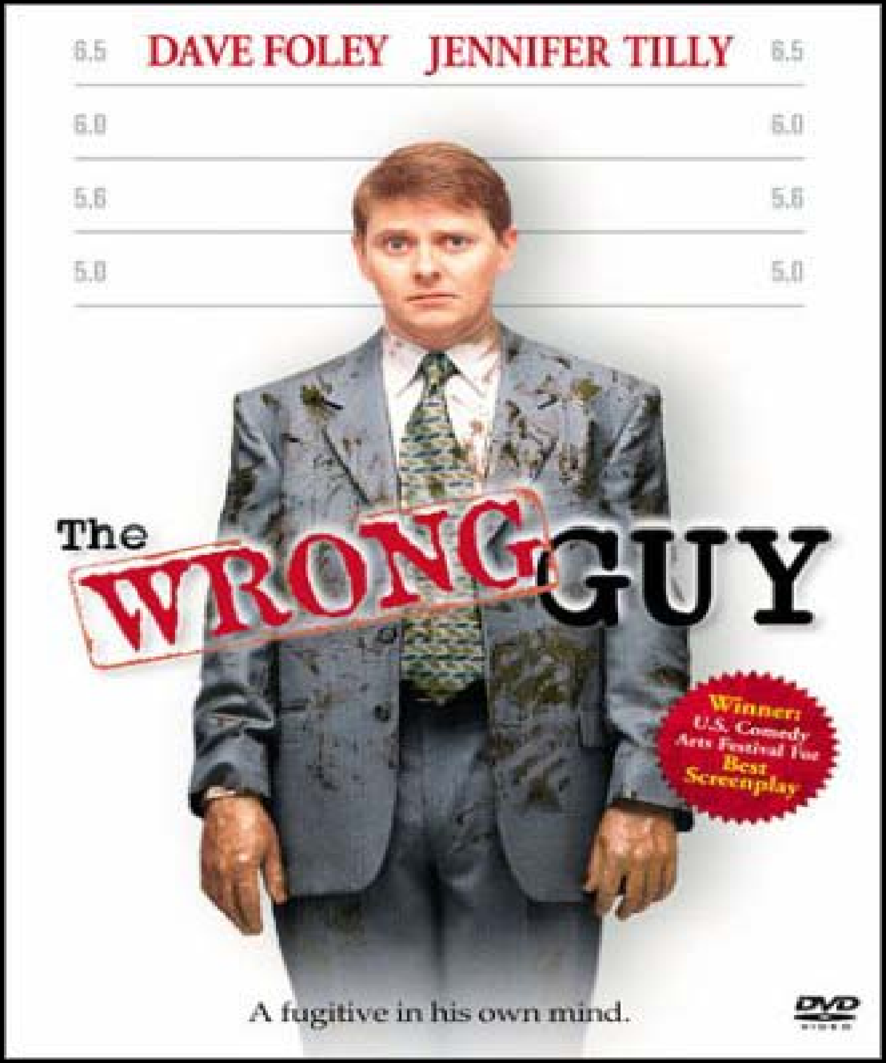
Problems

Expertise and patience



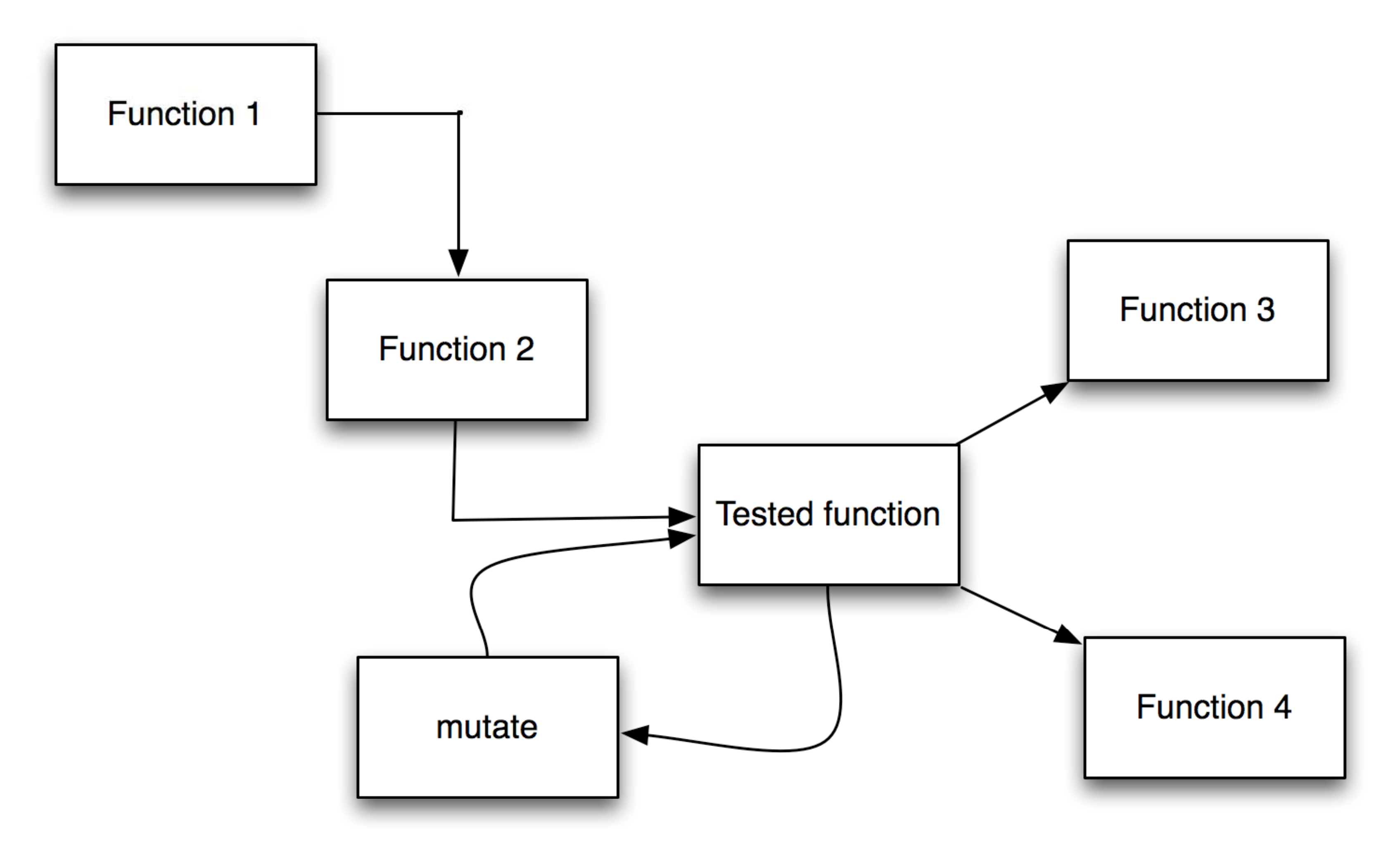
Memory instability

False positives

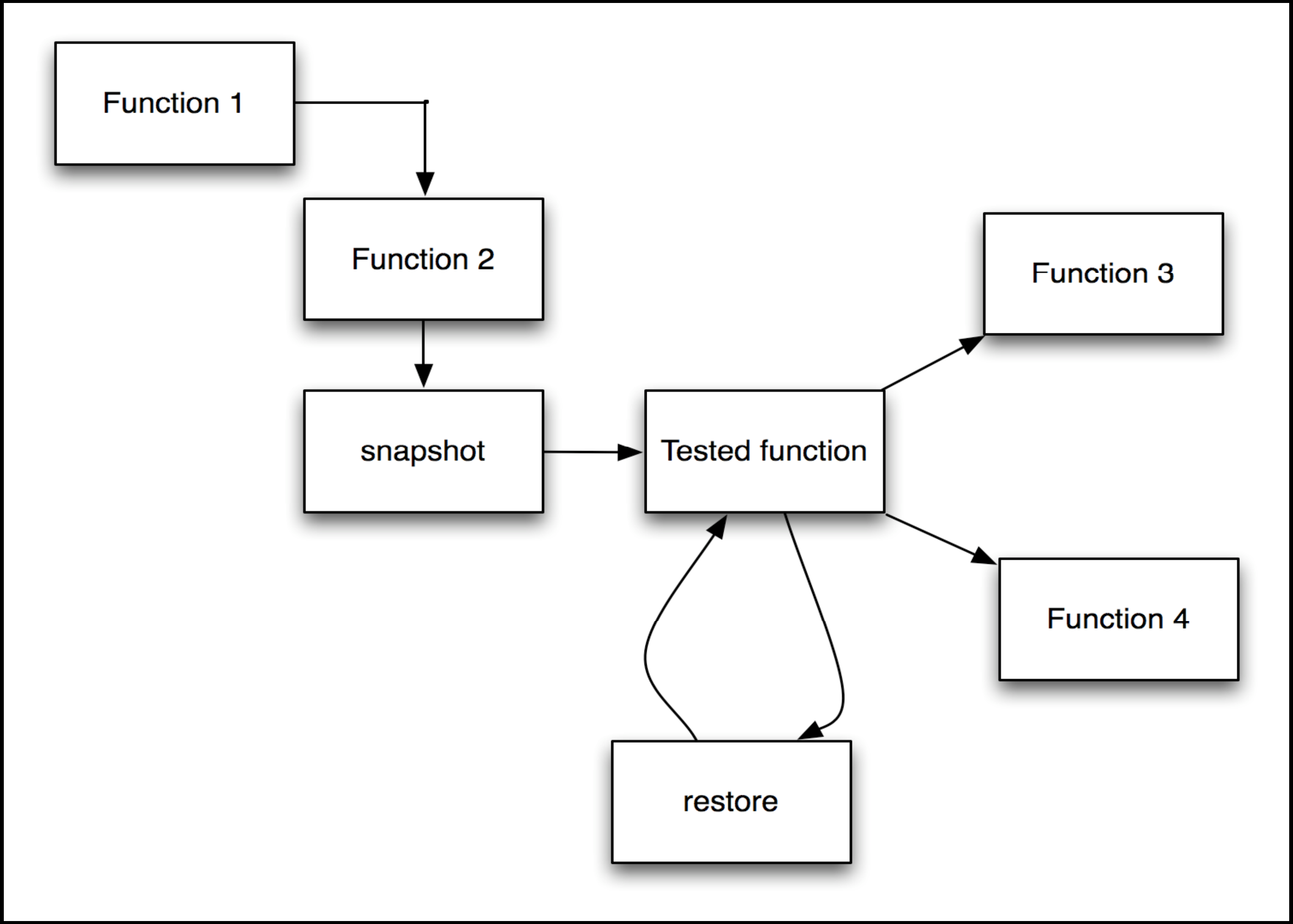


False negatives

Mutation loop insertion



Snapshot mutation restoration

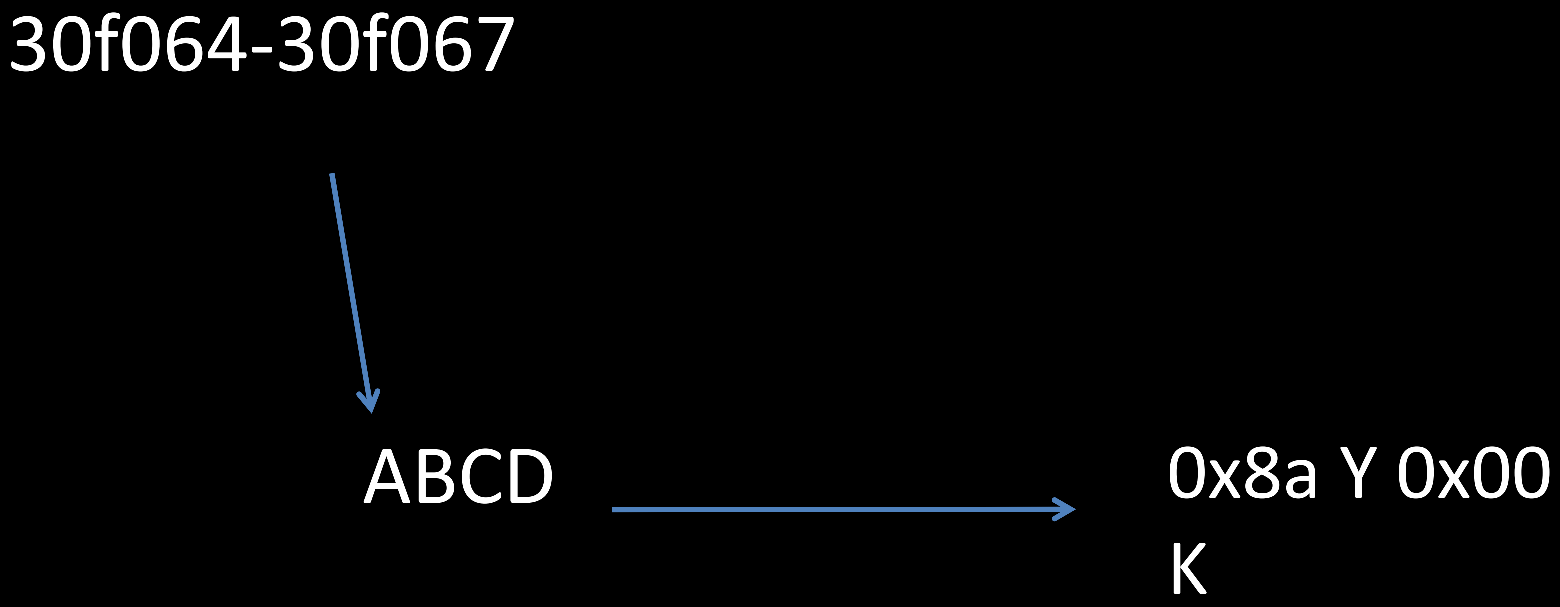


What do we do?

- Hook image
- Hook functions
- Hook instructions

First approach

For instance…



30f064-30f067

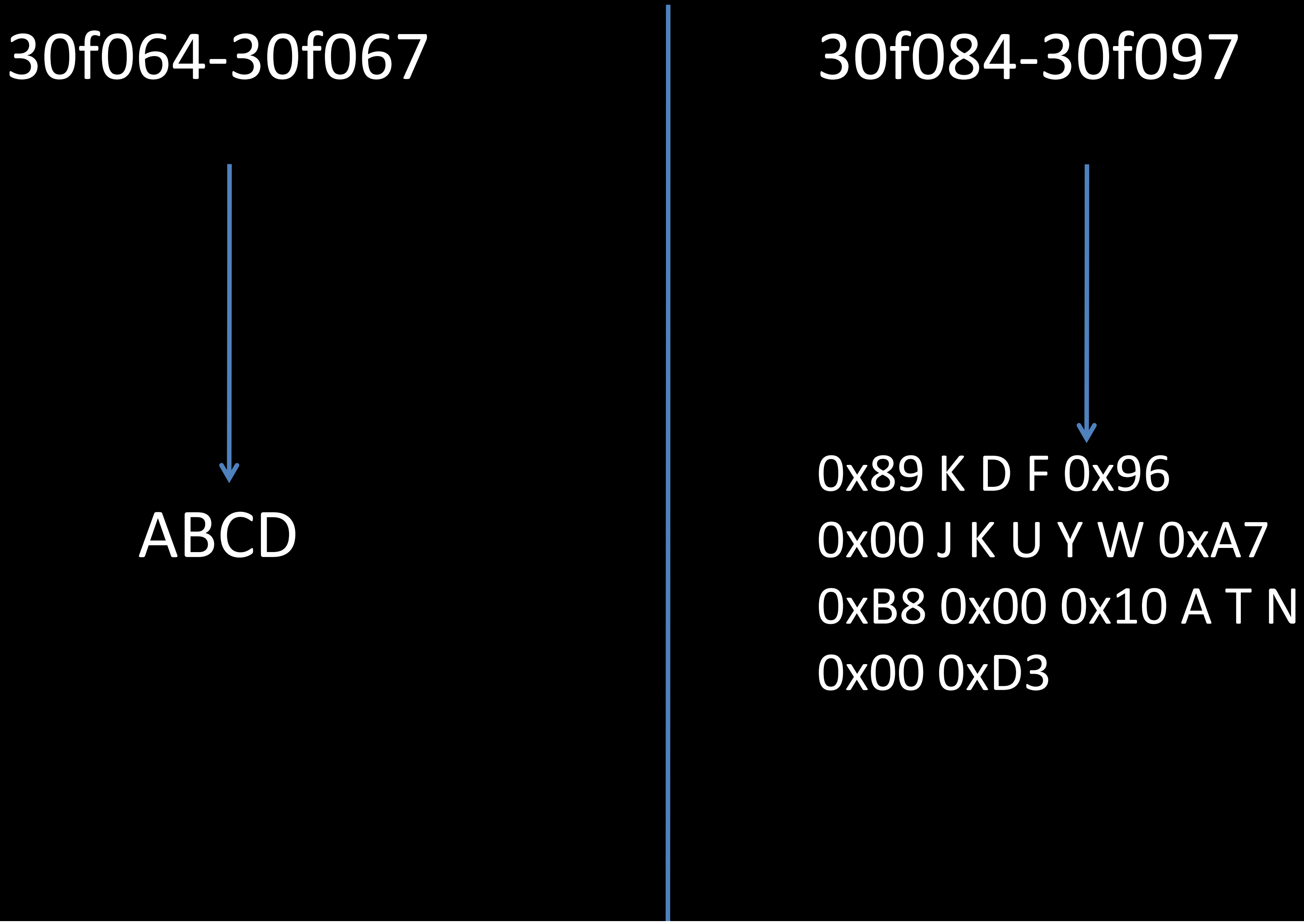
ABCD

0x8a Y 0x00

K

Second approach

Example



30f064-30f067

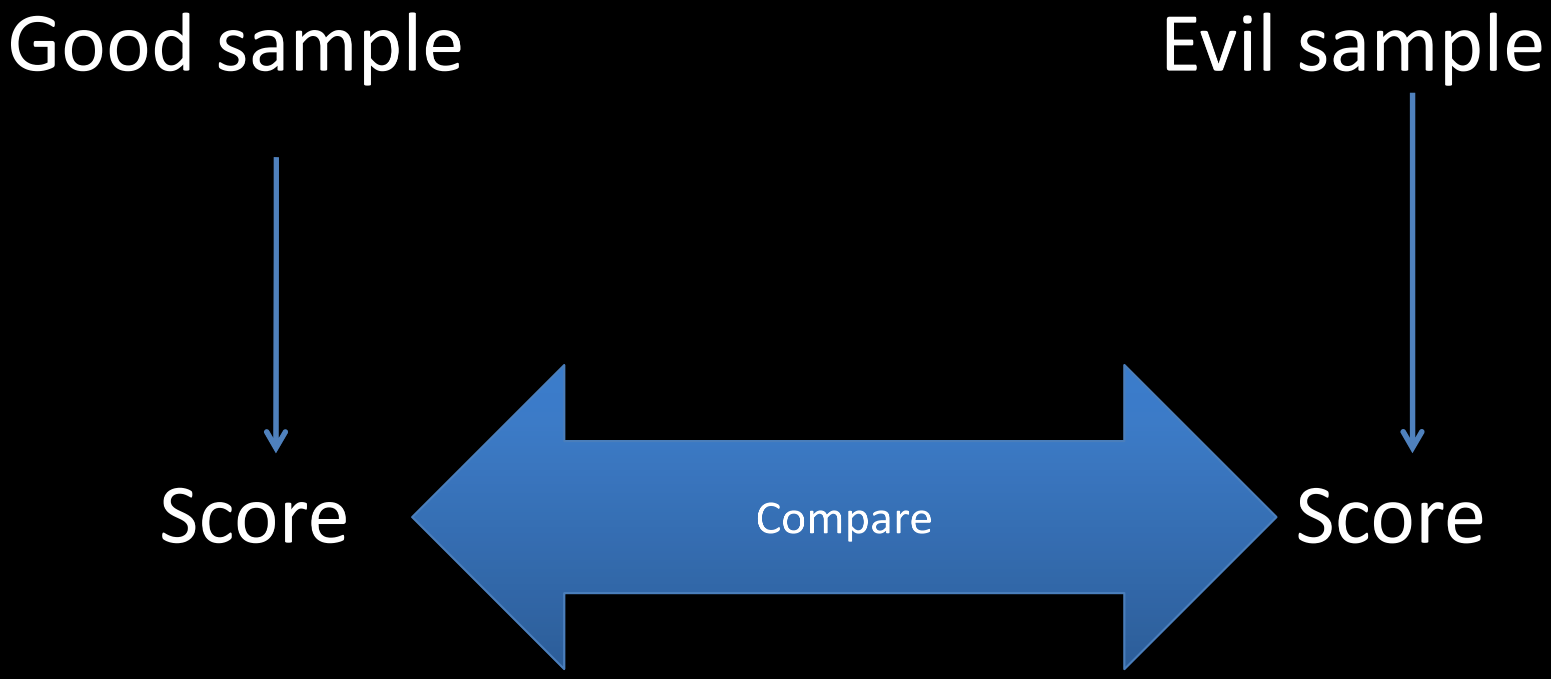
ABCD

30f084-30f097

0x89 K D F 0x96
0x00 J K U Y W 0xA7
0xB8 0x00 0x10 A T N
0x00 0xD3

Code coverage

How?



Good sample

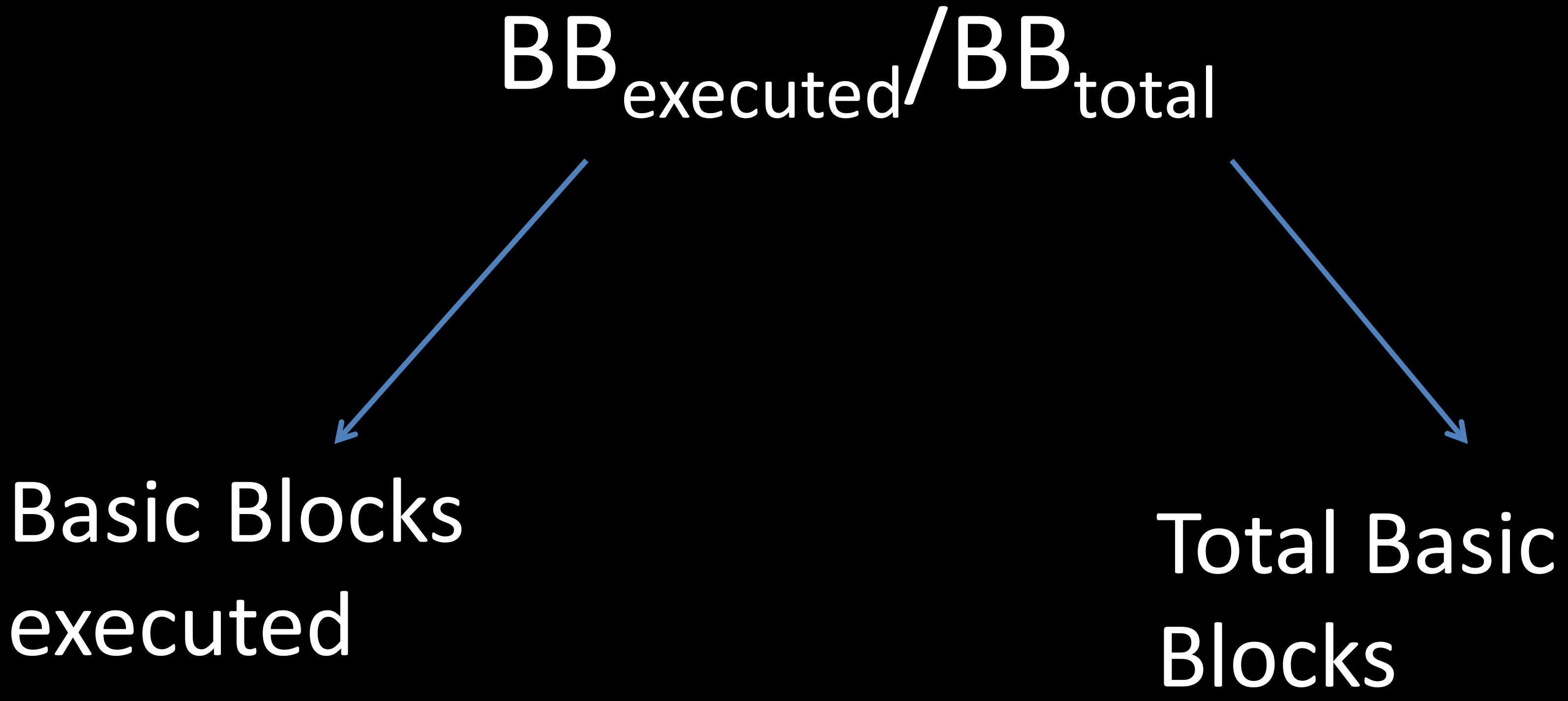
Score

Compare

Evil sample

Score

Score

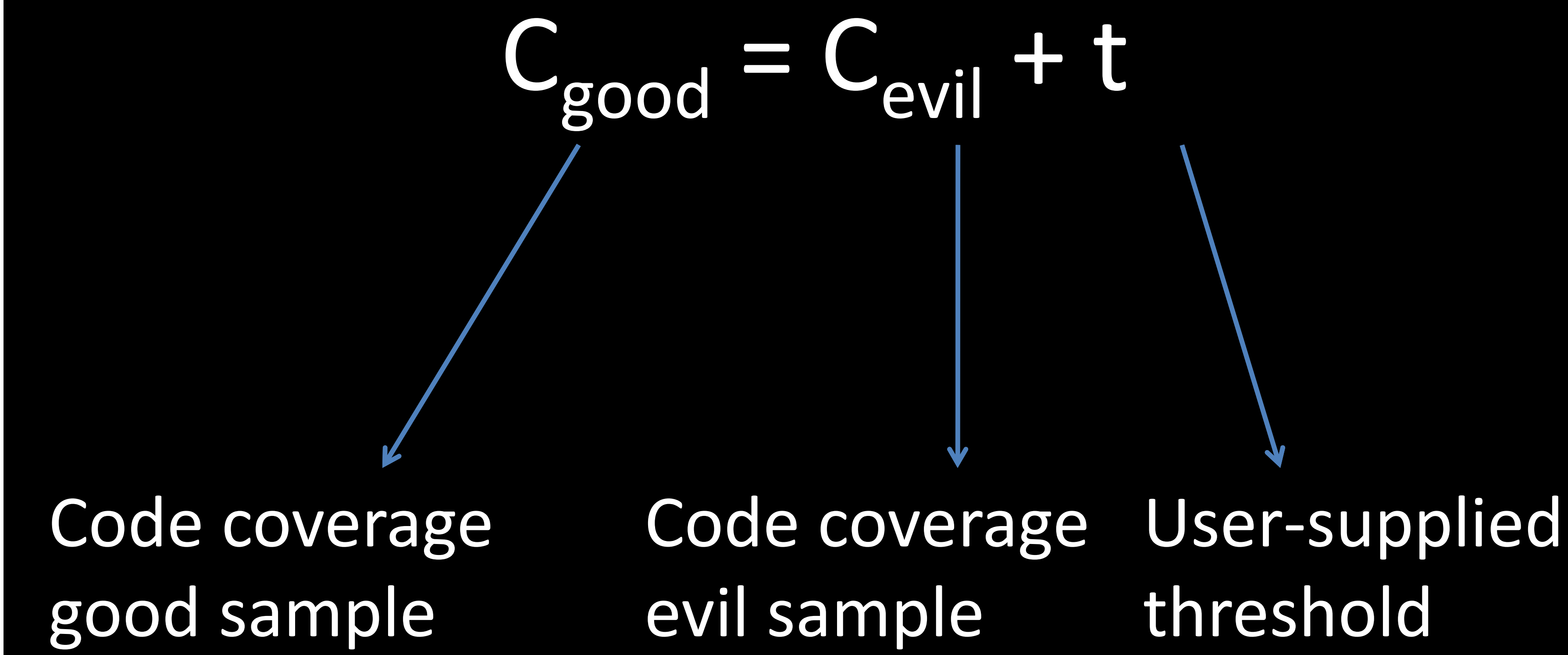


$BB_{executed}/BB_{total}$

Basic Blocks executed

Total Basic Blocks

Halting



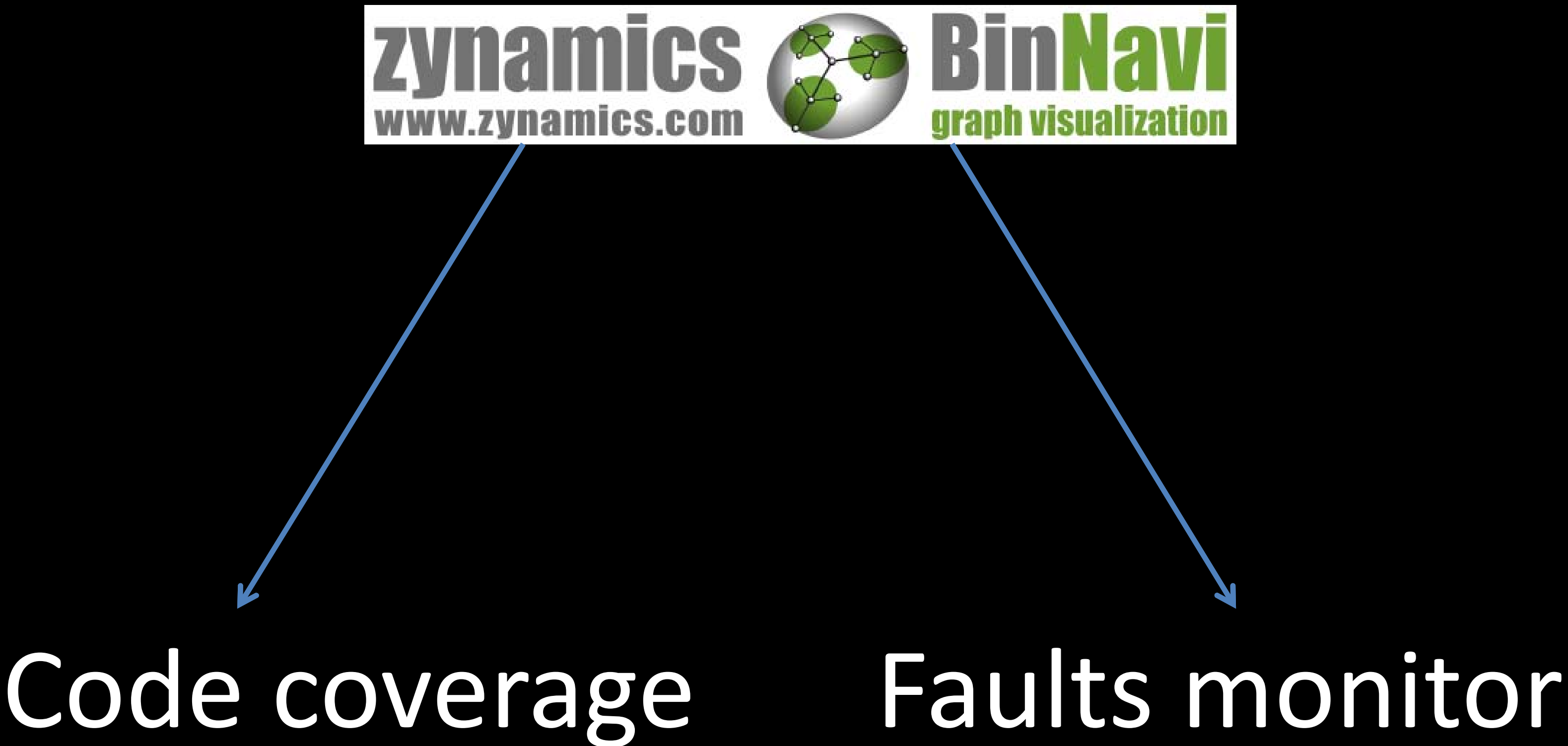
$$C_{good} = C_{evil} + t$$

Code coverage good sample

Code coverage evil sample

User-supplied threshold

What do we use?



Code coverage

Faults monitor

DEMO

Future – A reasoner

Thanks

Questions!

More Info

viozzo.wordpress.com

@_snagg

vincenzo.iozzo@zynamics.com