Reverse engineering with determination

Carl Svensson

June 1, 2017

Securityfest 2017

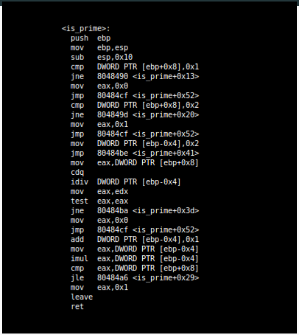
About me

- Carl Svensson, 26
- MSc in Computer Science, KTH
- Head of Security, Kry
- CTF-player, HackingForSoju
- ✉ calle.svensson@zeta-two.com
- 🐦 @zetatwo
- 🌐 https://zeta-two.com

Reverse engineering in 30 seconds?

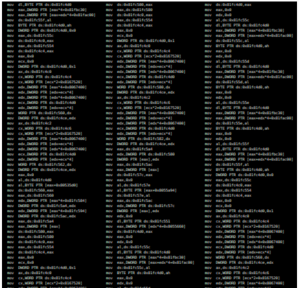
- Take stuff, e.g. software, apart
- Understand how it works
- Many possible goals
  - Malware, what does it do?
  - Audit closed source components
- Static analysis
  - + The whole picture
  - - A lot to read, slow
- Dynamic analysis
  - + See data flow
  - - Not all paths taken

Enter the M/o/Vfuscator



```
<is_prime>:
  push  ebp
  mov   ebp,esp
  sub   esp,0x10
  cmp   DWORD PTR [ebp+0x8],0x1
  jne   8048490 <is_prime+0x13>
  mov   eax,0x0
  jmp   80484cf <is_prime+0x52>
  cmp   DWORD PTR [ebp+0x8],0x2
  jne   804849d <is_prime+0x20>
  mov   eax,0x1
  jmp   80484cf <is_prime+0x52>
  mov   DWORD PTR [ebp-0x4],0x2
  jmp   80484be <is_prime+0x41>
  mov   eax,DWORD PTR [ebp+0x8]
  cdq
  idiv  DWORD PTR [ebp-0x4]
  mov   eax,edx
  test  eax,eax
  jne   80484ba <is_prime+0x3d>
  mov   eax,0x0
  jmp   80484cf <is_prime+0x52>
  add   DWORD PTR [ebp-0x4],0x1
  mov   eax,DWORD PTR [ebp-0x4]
  imul  eax,DWORD PTR [ebp-0x4]
  cmp   eax,DWORD PTR [ebp+0x8]
  jle   80484a6 <is_prime+0x29>
  mov   eax,0x1
  leave
  ret
```

- x86 MOV is turing complete
- 1 instruction compiler
  - Chris Domas
- No branches
- Dummy data



```
mov  dl,BYTE PTR ds:0x81fc4d0
mov  eax,DWORD PTR [eax*4+0x81fbc30]
mov  eax,DWORD PTR [eax+edx*4+0x81fac00]
mov  ds:0x81fc55f,al
mov  BYTE PTR ds:0x81fc4d0,ah
mov  DWORD PTR ds:0x81fc4d0,0x0
mov  eax,ds:0x81fc55c
mov  ds:0x81fc4c0,eax
mov  eax,ds:0x81fc554
mov  ds:0x81fc4c4,eax
mov  eax,0x0
mov  ecx,0x0
mov  DWORD PTR ds:0x81fc4d0,0x1
mov  ax,ds:0x81fc4c0
mov  cx,WORD PTR ds:0x81fc4c4
mov  cx,WORD PTR [ecx*2+0x8167520]
mov  edx,DWORD PTR [eax*4+0x8067400]
mov  edx,DWORD PTR [edx+ecx*4]
mov  edx,DWORD PTR [edx*4+0x8067400]
mov  ecx,DWORD PTR ds:0x81fc4d0
mov  edx,DWORD PTR [edx+ecx*4]
mov  WORD PTR ds:0x81fc560,dx
mov  DWORD PTR ds:0x81fc4ce,edx
mov  ax,ds:0x81fc4c2
mov  cx,WORD PTR ds:0x81fc4c6
mov  cx,WORD PTR [ecx*2+0x8167520]
mov  edx,DWORD PTR [eax*4+0x8067400]
mov  edx,DWORD PTR [edx+ecx*4]
mov  edx,DWORD PTR [edx*4+0x8067400]
mov  ecx,DWORD PTR ds:0x81fc4d0
mov  edx,DWORD PTR [edx+ecx*4]
mov  WORD PTR ds:0x81fc562,dx
mov  DWORD PTR ds:0x81fc4ce,edx
mov  eax,0x0
mov  al,ds:0x81fc4d0
mov  al,BYTE PTR [eax+0x80535d0]
mov  ds:0x81fc560,eax
mov  eax,ds:0x81fc560
mov  edx,DWORD PTR [eax*4+0x81fc584]
mov  DWORD PTR ds:0x81fc5a4,edx
mov  edx,DWORD PTR [eax*4+0x81fc594]
mov  DWORD PTR ds:0x81fc5ac,edx
mov  eax,ds:0x81fc5a4
mov  eax,DWORD PTR [eax]
mov  ds:0x81fc580,eax
mov  eax,ds:0x81fc580
mov  ds:0x81fc4c0,eax
mov  eax,ds:0x81fc554
mov  ds:0x81fc4c4,eax
mov  eax,0x0
mov  ecx,0x0
mov  DWORD PTR ds:0x81fc4d0,0x1
mov  ax,ds:0x81fc4c0
mov  cx,WORD PTR ds:0x81fc4c4
mov  cx,WORD PTR [ecx*2+0x8167520]

mov  ds:0x81fc580,eax
mov  eax,ds:0x81fc580
mov  ds:0x81fc4c0,eax
mov  eax,ds:0x81fc554
mov  ds:0x81fc4c4,eax
mov  eax,0x0
mov  ecx,0x0
mov  DWORD PTR ds:0x81fc4d0,0x1
mov  ax,ds:0x81fc4c0
mov  cx,WORD PTR ds:0x81fc4c4
mov  cx,WORD PTR [ecx*2+0x8167520]
mov  edx,DWORD PTR [eax*4+0x8067400]
mov  edx,DWORD PTR [edx+ecx*4]
mov  edx,DWORD PTR [edx*4+0x8067400]
mov  ecx,DWORD PTR ds:0x81fc4d0
mov  edx,DWORD PTR [edx+ecx*4]
mov  WORD PTR ds:0x81fc580,dx
mov  DWORD PTR ds:0x81fc4ce,edx
mov  ax,ds:0x81fc4c2
mov  cx,WORD PTR ds:0x81fc4c6
mov  cx,WORD PTR [ecx*2+0x8167520]
mov  edx,DWORD PTR [eax*4+0x8067400]
mov  edx,DWORD PTR [edx+ecx*4]
mov  edx,DWORD PTR [edx*4+0x8067400]
mov  ecx,DWORD PTR ds:0x81fc4d0
mov  edx,DWORD PTR [edx+ecx*4]
mov  WORD PTR ds:0x81fc582,dx
mov  DWORD PTR ds:0x81fc4ce,edx
mov  eax,ds:0x81fc5a4
mov  edx,DWORD PTR ds:0x81fc580
mov  DWORD PTR [eax],edx
mov  eax,ds:0x81fc5ac
mov  eax,DWORD PTR [eax]
mov  ds:0x81fc57c,eax
mov  eax,0x0
mov  al,ds:0x81fc57e
mov  al,BYTE PTR [eax+0x8055a94]
mov  ds:0x81fc57e,al
mov  eax,ds:0x81fc5ac
mov  edx,DWORD PTR ds:0x81fc57c
mov  DWORD PTR [eax],edx
mov  edx,0x0
mov  dl,BYTE PTR ds:0x81fc551
mov  eax,DWORD PTR [edx*4+0x8055660]
mov  ds:0x81fc4d0,eax
mov  eax,0x0
mov  edx,0x0
mov  al,ds:0x81fc55c
mov  dl,BYTE PTR ds:0x81fc4d0
mov  eax,DWORD PTR [eax*4+0x81fbc30]
mov  eax,DWORD PTR [eax+edx*4+0x81fac00]
mov  ds:0x81fc55c,al
mov  BYTE PTR ds:0x81fc4d0,ah
mov  eax,0x0
mov  edx,0x0

mov  ds:0x81fc4d0,eax
mov  eax,0x0
mov  edx,0x0
mov  al,ds:0x81fc55c
mov  dl,BYTE PTR ds:0x81fc4d0
mov  eax,DWORD PTR [eax*4+0x81fbc30]
mov  eax,DWORD PTR [eax+edx*4+0x81fac00]
mov  ds:0x81fc55c,al
mov  BYTE PTR ds:0x81fc4d0,ah
mov  eax,0x0
mov  edx,0x0
mov  al,ds:0x81fc55d
mov  dl,BYTE PTR ds:0x81fc4d0
mov  eax,DWORD PTR [eax*4+0x81fbc30]
mov  eax,DWORD PTR [eax+edx*4+0x81fac00]
mov  ds:0x81fc55d,al
mov  BYTE PTR ds:0x81fc4d0,ah
mov  eax,0x0
mov  edx,0x0
mov  al,ds:0x81fc55e
mov  dl,BYTE PTR ds:0x81fc4d0
mov  eax,DWORD PTR [eax*4+0x81fbc30]
mov  eax,DWORD PTR [eax+edx*4+0x81fac00]
mov  ds:0x81fc55e,al
mov  BYTE PTR ds:0x81fc4d0,ah
mov  eax,0x0
mov  edx,0x0
mov  al,ds:0x81fc55f
mov  dl,BYTE PTR ds:0x81fc4d0
mov  eax,DWORD PTR [eax*4+0x81fbc30]
mov  eax,DWORD PTR [eax+edx*4+0x81fac00]
mov  ds:0x81fc55f,al
mov  BYTE PTR ds:0x81fc4d0,ah
mov  DWORD PTR ds:0x81fc4d0,0x0
mov  eax,ds:0x81fc55c
mov  ds:0x81fc4c0,eax
mov  eax,ds:0x81fc554
mov  ds:0x81fc4c4,eax
mov  eax,0x0
mov  ecx,0x0
mov  DWORD PTR ds:0x81fc4d0,0x1
mov  ax,ds:0x81fc4c0
mov  cx,WORD PTR ds:0x81fc4c4
mov  cx,WORD PTR [ecx*2+0x8167520]
mov  edx,DWORD PTR [eax*4+0x8067400]
mov  edx,DWORD PTR [edx+ecx*4]
mov  edx,DWORD PTR [edx*4+0x8067400]
mov  ecx,DWORD PTR ds:0x81fc4d0
mov  edx,DWORD PTR [edx+ecx*4]
mov  WORD PTR ds:0x81fc560,dx
mov  DWORD PTR ds:0x81fc4ce,edx
mov  ax,ds:0x81fc4c2
mov  cx,WORD PTR ds:0x81fc4c6
mov  cx,WORD PTR [ecx*2+0x8167520]
mov  edx,DWORD PTR [eax*4+0x8067400]
```

Crack Me #1

- Find the password
- Dead ends
    - Name functions
    - Study branches
    - Trace execution
    - Count instructions

Intermission: What is a program?

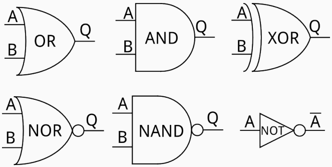
- Takes input
- Perform calculations
- Affects the world
    - Output
    - Affect memory
    - OS API calls

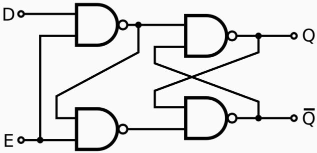
Crack Me #1, take 2

- Memory trace
- Diff
- Scripting glue
- Victory!

Electronics 101

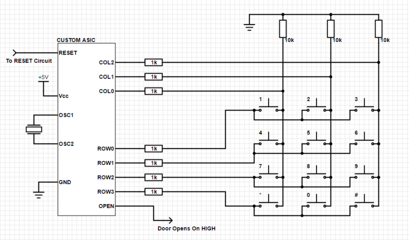
- Boolean circuits
- Gates
  - AND, OR, NOT, NAND, XOR
  - D-latch, Flip-Flip
- Functional completeness

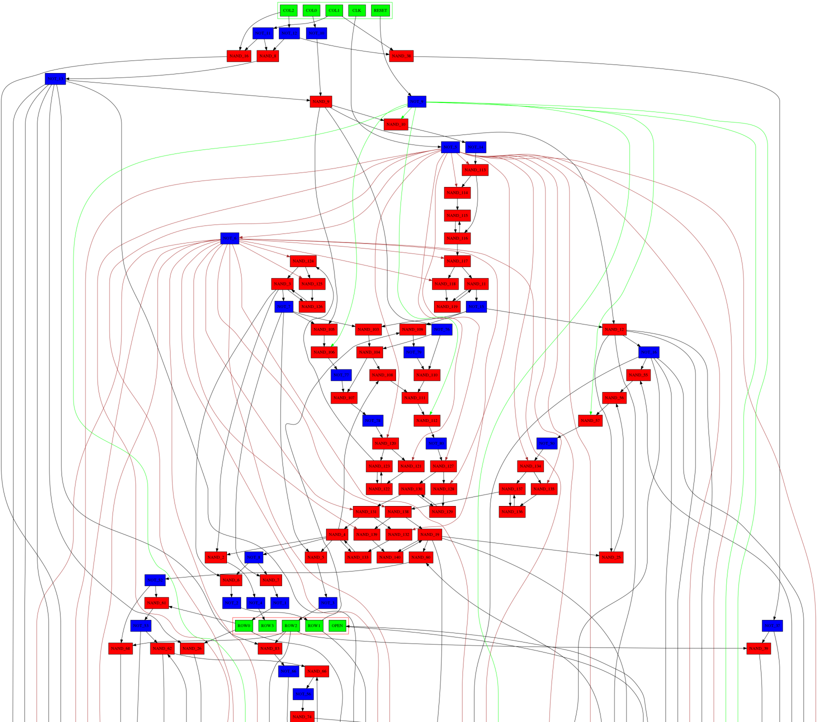


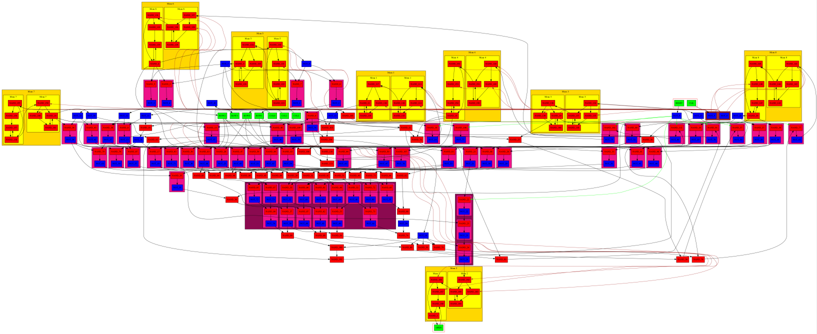


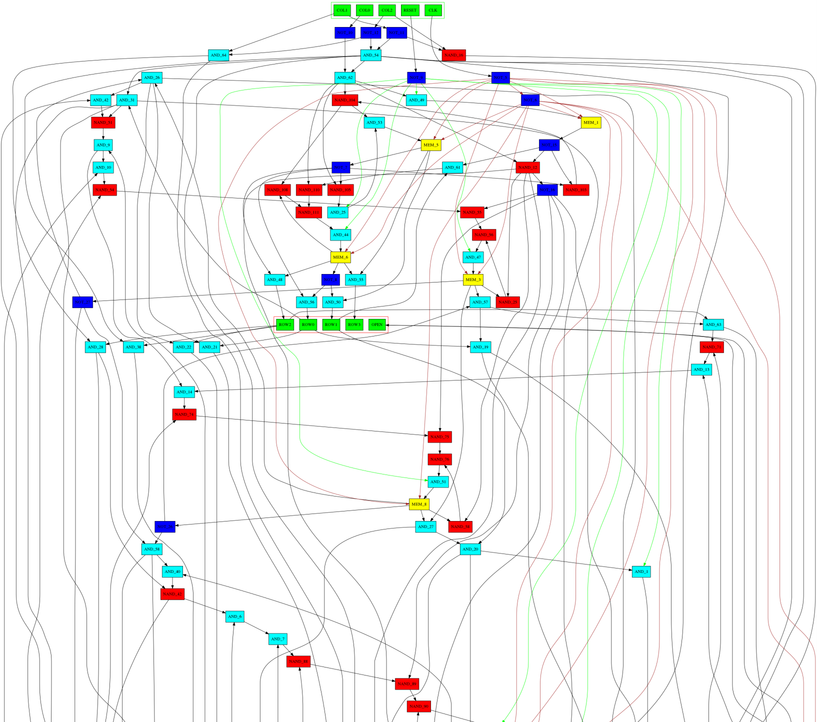
Crack Me #2

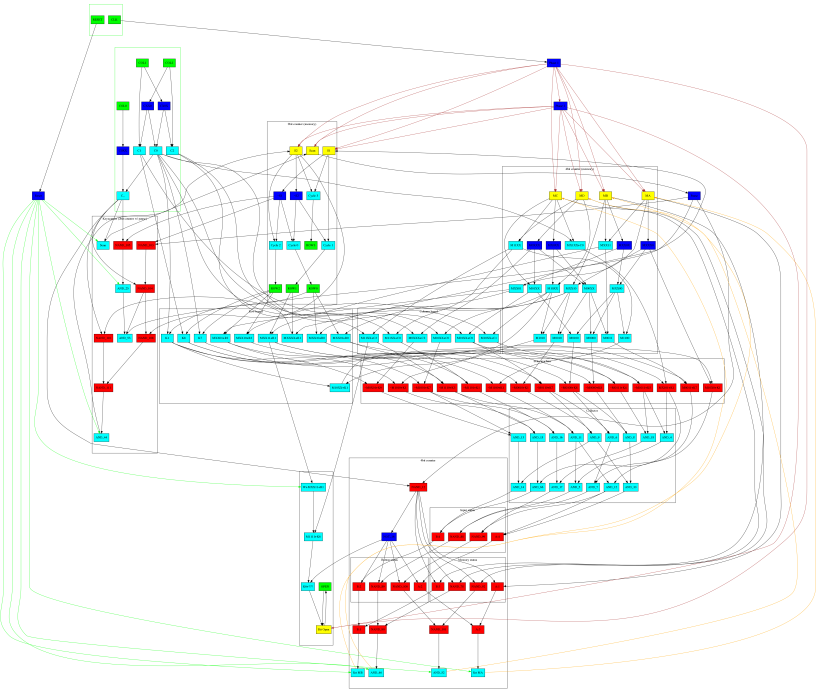
- Keypad, IC & lock
- Single ”instruction”
- Easy way: dynamic
- My way: static
    - Find subgraph
    - Replace, abstract, iterate
    - Naming




COL2
COL0
COL1
CLK
RESET
NOT_11
NOT_12
NOT_10
NAND_16
NAND_8
NAND_38
NOT_13
NAND_9
NOT_9
NAND_10
NOT_5
NOT_14
NAND_113
NAND_114
NAND_115
NAND_116
NOT_6
NAND_124
NAND_117
NAND_3
NAND_125
NAND_118
NAND_11
NOT_7
NAND_126
NAND_119
NOT_15
NAND_105
NAND_103
NAND_109
NOT_76
NAND_12
NAND_106
NAND_104
NOT_79
NOT_16
NOT_77
NAND_108
NAND_110
NAND_55
NAND_107
NAND_111
NAND_56
NOT_78
NAND_112
NAND_57
NAND_120
NOT_80
NOT_50
NAND_123
NAND_121
NAND_127
NAND_134
NAND_122
NAND_130
NAND_128
NAND_137
NAND_135
NAND_131
NAND_138
NAND_129
NAND_136
NAND_4
NAND_139
NAND_132
NAND_19
NAND_2
NOT_8
NAND_5
NAND_133
NAND_140
NAND_60
NAND_25
NOT_52
NAND_6
NAND_7
NAND_61
NOT_2
NOT_4
NOT_1
NOT_3
NOT_53
ROW0
ROW3
ROW2
ROW1
OPEN
NOT_37
NAND_68
NAND_62
NAND_26
NAND_83
NAND_39
NOT_66
NAND_66
NOT_56
NAND_74


Mem 6
Mem 5
Mem 1
Mem 4
Mem 0
Mem 7
Mem 3
Mem 2
ROW2
ROW3
ROW0
ROW1
COL0
COL1
COL2
RESET
CLK
OPEN


COL1
COL0
COL2
RESET
CLK
NOT_10
NOT_12
NOT_11
AND_64
AND_54
NAND_16
AND_26
AND_62
NOT_9
NOT_5
AND_42
AND_31
NAND_104
AND_49
NOT_6
NAND_51
AND_53
MEM_1
AND_9
MEM_5
NOT_15
AND_10
NOT_7
AND_61
NAND_12
NAND_54
NAND_108
NAND_110
NAND_105
NOT_16
NAND_103
NAND_111
AND_25
NAND_55
NAND_56
AND_44
AND_47
MEM_6
MEM_3
AND_48
NOT_8
AND_55
NOT_27
AND_56
AND_50
AND_57
NAND_25
ROW2
ROW0
ROW1
ROW3
OPEN
AND_63
AND_28
AND_38
AND_22
AND_21
AND_19
NAND_21
AND_13
AND_14
NAND_74
NAND_75
NAND_76
AND_51
MEM_8
NOT_26
AND_27
NAND_58
AND_58
AND_20
AND_40
AND_1
NAND_42
AND_6
AND_7
NAND_88
NAND_89
NAND_90


RESET
CLK
COL1
COL2
COL0
CXX0
CX0X
C0XX
C1
C0
C2
C_
Reset
Phase 1
Phase 2
2bit counter (memory)
S2
Scan
S1
~S1
~S2
Cycle 3
Cycle 2
Cycle 0
ROW3
Cycle 1
ROW2
ROW1
ROW0
4bit counter (memory)
MC
MD
MB
MA
Pause
M11XX
M0XXX
MX0XX
MX1XX+C0
MXX11
MXX0X
MXXX0
MXX01
M01XX
M10XX
MXX10
M00XX
MXX00
M1010
M0010
M0100
M0000
M0011
M1100
Keyscanner (2bit counter w/ pause)
Scan
NAND_105
NAND_103
AND_25
NAND_104
NAND_110
AND_53
NAND_108
NAND_111
AND_44
Row based
K1
K8
K7
MXX01+R2
MXX10+R2
MXX11+R1
MXXXX+R1
MXX10+R0
MXX01+R0
M10XX+K1
Column based
M11XX+C2
M11XX+C0
M0XXX+C2
M10XX+C0
M01XX+C0
M10XX+C1
State machine
~M0X01+K9
~M1010+K1
~M1001+K7
~M1110+K3
~M1101+K1
~M1100+K1
~M0010+K1
~M0110+K7
~M0100+K8
~M0000+K8
~M0111+K4
~M1011+K5
MX101+K1
M0011+K7
M10X0+K1
Collector
AND_13
AND_15
AND_16
AND_11
AND_9
AND_6
AND_8
AND_18
AND_4
AND_14
AND_66
AND_17
AND_5
AND_7
AND_12
AND_10
W+MXX11+R1
M1111+K6
K6+???
OPEN
Bit Open
4bit counter
NAND_12
NOT_16
Input status
B:4
NAND_88
NAND_99
A:4
Button status
B:2
NAND_89
NAND_100
A:2
Memory status
B:3
NAND_78
NAND_92
A:1
B:1
NAND_90
NAND_101
A:3
Set MB
AND_46
AND_52
Set MA

Keep practicing, stay sharp

- Like going to the gym
- Push your limits
- Have fun!
- LabyREnth, http://labyrenth.com/ 10/6-23/7
    - $23,000 USD total prizes

Thanks for listening!