REpsych
: psychological warfare in reverse engineering

{ def con 2015 // domas }
This serves no purpose

Warning
Taking something apart …
… to figure out how it works

With software…
- Interfacing
- Documentation
- Obsolescence
- Bug fixing
- Academic

Reverse Engineering?
Taking something apart …
  … to figure out how it works

With software…

- Military/commercial espionage
- Unauthorized duplication
- Security analysis
- Vulnerability analysis
- Malware analysis

Reverse Engineering?
Whenever we write something awesome…

- Video game
- Encryption algorithm
- Malware
- 0-Day
- RAT

... someone, at some point, is going to …

- Capture it
- Dissect it
- Reverse it

Reverse Engineering?
If you don’t want your work destroyed ...

... it pays to plan ahead

Anti-RE
 Encryption
 Obfuscation
 Anti-debugging

Anti-RE
Reverse Engineering.

```bash
objdump -d -Mintel a.out
```
mov DWORD PTR [rbp-0x8], 0x0
push 600004
call printf
pop eax
add DWORD PTR [rbp-0x8], 0x1
cmp DWORD PTR [rbp-0x8], 0x100
jle 4004f2 <main+0xb>
mov is Turing-complete

Stephen Dolan

mov destination, source
Any code we write …
… can be written as a set of movs instead
… and nothing else
Really?
That’d be tough to reverse engineer, wouldn’t it?

Turing Complete?
4004e9:  mov    DWORD PTR [rbp-0x8],0x0
4004f2:  push   600004
4004f8:  call   printf
4004fa:  pop    eax
4004fc:  add    DWORD PTR [rbp-0x8],0x1
400500:  cmp    DWORD PTR [rbp-0x8],0x100
400507:  jle    4004f2 <main+0xb>
80515bc: mov    eax, ds:0x835d81a
80515c1: mov    ebx, DWORD PTR [eax+0x835d6fc]
80515c7: mov    edx, DWORD PTR ds:0x835d7da
80515cd: mov    eax, 0x0
80515d2: mov    al, BYTE PTR [ebx+edx*1]
80515d5: mov    al, BYTE PTR [eax+0x835dc7e]
80515db: mov    BYTE PTR [ebx+edx*1], al
80515de: mov    eax, ds:0x835d81a
80515e3: mov    ebx, DWORD PTR [eax+0x835d6fc]
80515e9: mov    edx, DWORD PTR ds:0x835d7da
80515ef: mov    eax, 0x0
80515f4: mov    al, BYTE PTR [ebx+edx*1]
The M/o/Vfuscator

- mov-only C Compiler
  - [https://github.com/xoreaxeaxeax](https://github.com/xoreaxeaxeax)
- First single instruction C compiler!
Crackmes

The M/o/Vfuscator
How would an experienced reverse engineer approach this?
mov byte [eax+0x80e17bc],0x0
mov ax,[0x80a0451]
mov eax,0x0
mov [0x80a044d],eax
mov [0x80a0451],eax
mov eax,[eax+0x80a05a6]
mov eax,[0x80a044d]
mov eax,[eax+edx]
mov eax,[0x80a0556]
mov edx,[eax+0x80c0bba]
mov [ebx],edx
mov eax,[eax+0x80a051e]
mov edx,[eax+0x80a058e]
mov eax,[eax+0x80a044d]
mov eax,[0x80a0556]
mov edx,0x0
mov [dword 0x80a0451],edx
mov dl,[eax+0x80e0bbc]
mov edx,0x0
mov eax,[0x80a0516]
mov eax,[0x80a044d]
mov eax,[0x80a056e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a058e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a051e]
mov edx,0x0
mov eax,[0x80a0556]
mov edx,0x0
mov [0x80a044d],eax
mov eax,[eax+0x80a0556]
Anti-RE

- Code doesn’t have to be hard to reverse
- Just need to make the reverser give up

Realization
Demoralization

Break down the reverser

Psychological Warfare
How else can we make a reverser quit?

Psychological Warfare
Sending messages...
..cantor.dust..
Visualize data patterns
Default: entropy distribution

cantor.dust..
Send a message?

- Run a message through an inverse Hilbert transform
- Rebuild program to match desired entropy

..cantor.dust..
Top o' the mornin' to ya!
Strings?

Sending messages
These are horrible…

No one will ever see the message
And if they do, they won’t care
Need something better…

Sending messages
IDA
Control flow graphs...
sub_100000360 proc near
mov5xd r8, dword ptr [rcx+3Ch]
xor r9d, r9d
mov r10, rdx
add r8, rcx
movzx eax, word ptr [r8+14h]
movzx r11d, word ptr [r8+6h]
lea rcx, [rax+r8+18h]
test r11d, r11d
jz short loc_10000039F

loc_100000381:
mov edx, [rcx+0Ch]
cmp r10, rdx
jb short loc_100000393

mov eax, [rcx+8]
add eax, edx
cmp r10, rax
jb short loc_1000003A2

loc_100000393:
inc r9d
add rcx, 28h
cmp r9d, r11d
jb short loc_100000391

loc_1000003A2:
mov rax, rcx
ret
sub_100000360 endp

loc_10000039F:
xor eax, eax
ret

IDA...
Hopper...
Radare...
We’ll look at IDA
But the algorithm will work on anything
If you stare at these control graphs long enough... 
... they almost start to look like things

Idea...
Could we send a message through a CFG?
Reverse engineer IDA?
   Yep!

Drawing with CFGs
Draw horizontal lines:

Switch

“Orphan” jumps

jmp a
jmp a
jmp a
jmp a
jmp a
jmp a
jmp a
jmp a
jmp a

a:

Idea 1
Draw vertical lines:

Non-branching code

nop
nop
nop
nop
nop
nop
nop

Idea 1
Combining the two
Etch-a-sketch, in IDA!

Idea 1
top:
jmp left
jmp top_end
... ; repeat
jmp right_side
top_end:
jmp $

left_side:
nop
... ; repeat
jmp bottom_left

right_side:
nop
... ; repeat
jmp bottom_right

bottom:
nottom_left:
jmp bottom_end
... ; repeat
bottom_right:

bottom_end:
ret
Observation

IDA tries to align blocks in a given row
top:
jmp left
jmp top_end
... ; repeat
jmp right_side
top_end:
jmp $

left_side:
jmp $+2
... ; repeat
jmp bottom_left

right_side:
jmp $+2
... ; repeat
jmp bottom_right

bottom:
botton_left:
jmp bottom_end
... ; repeat
bottom_right:

bottom_end:
ret
Observation

IDA tries to keep rows/columns together
- But minimize branching distance
• Hour of tinkering
• Couldn’t make it work
• Try something else

Separating the columns
We have some control over how rows are arranged
  Depends on nodes between
IDA has all the control over columns
  Can rearrange parent nodes and branches to keep columns close together

R.I.P. Idea 1
Force IDA to keep things in order
- Tie nodes together as tightly as possible
- Prevent rearranging

Idea 2
A node
A tightly woven CFG
x:

a0: je b1  
a1: je b2  
a2: je b3  
a3:  
b0: je c1  
b1: je c2  
b2: je c3  
b3:  
c0: je d1  
c1: je d2  
c2: je d3  
c3:  
d0: jmp F  
d1: jmp F  
d2: jmp F  
d3: jmp F  
F:
%macro column 3-4 "nonempty"
    %assign r 0
    %assign c %1
    %rep %2-1
        %assign nr r+1
        %assign nc c+1
        e_%+r%+_%+c:
        %ifidn %4, "empty"
        %else
            je e_%+nr%+_%+nc
        %endif
        %assign r r+1
    %endrep
    e_%+r%+_%+c: jmp %3
%endmacro
Idea 2, continued

- "Weave" the CFG together
- Turn "pixel" off by removing node?
A tightly woven CFG
A tightly woven CFG
A tightly woven CFG, II
; row, column, width, height, done
%macro diag 5
  %assign r %1
  %assign c %2
  %assign width %3
  %assign height %4

  %rep 256 ; max size
    %assign nr r+1
    %assign nc c+1

    e_%+r%+_%+c:
    %if nr >= height
      %elif nc >= width
        je e_%+nr%+_%+nc
        %exitrep
    %else
      %endif
    %endif
  %endrep

%endmacro

%if c == 0
  jmp e_%+nr%+_%+nc
  %exitrep
%else
  %endif

%endif

%assign r r+1
%assign c c-1
%if r>=width
  jmp %5
%exitrep
%endif

%endrep

%endmacro
We still can’t remove a node

R.I.P. Idea 2
Leave all nodes
Fill with code if “on”
Leave empty if “off”

Idea 3
- “Empty” pixel still needs 2 lines
- Increase contrast by reducing impact of those 2
- Reduce impact by increasing height
- Increase height by increasing width
- vfmaddsub132ps xmm0, xmm1, xmmword ptr cs:[edi+esi*4+8068860h]

Enhance contrast
Almost there

- Insert always on column
Add a junk code generator

Almost there
movzx eax, bh
movzx ecx, dh
dec ecx
xor ebx, ecx
lea ebx, [ebp+1*4]
mov eax, 3526025642
or eax, 188401817
mov ah, 4
lea eax, [ecx+4*edx]
test edx, eax
mov cl, 2
add ebx, ecx
shr eax, 21
movzx ecx, dl
movzx ecx, dl
add ebx, ecx
shr eax, 25
mov ah, 4
test edx, eax
shr ecx, 19
movzx eax, bh
or eax, 2742937504
mov ah, 4
and edx, eax
Almost there
REpsych Toolchain

Generates assembly ...
... to form images through CFGs
(Demo)
Reverser is forced to sit and stare at whatever message you embed
Use it to your advantage, crush their soul

Psychological Warfare
Spoiler alert: You fail.
This gets worse and worse, until you give up or die.
(Draw an assembly selfie)
Stego
More ideas
QR
\[ a.k.a. \text{the ultimate CTF problem} \]

More ideas
Creepiest malware ever
- Scans your hard disk
- Rewrites itself to match your personal images
- (Demo)

More ideas
- 14 lines of assembly
- 328 lines of preprocessor macros
REpysch
M/o/Vfuscator 2.0
x86 0-day POC
Etc.

Feedback?

domas

@xoreaxeaxeax
xoreaxeaxeax@gmail.com