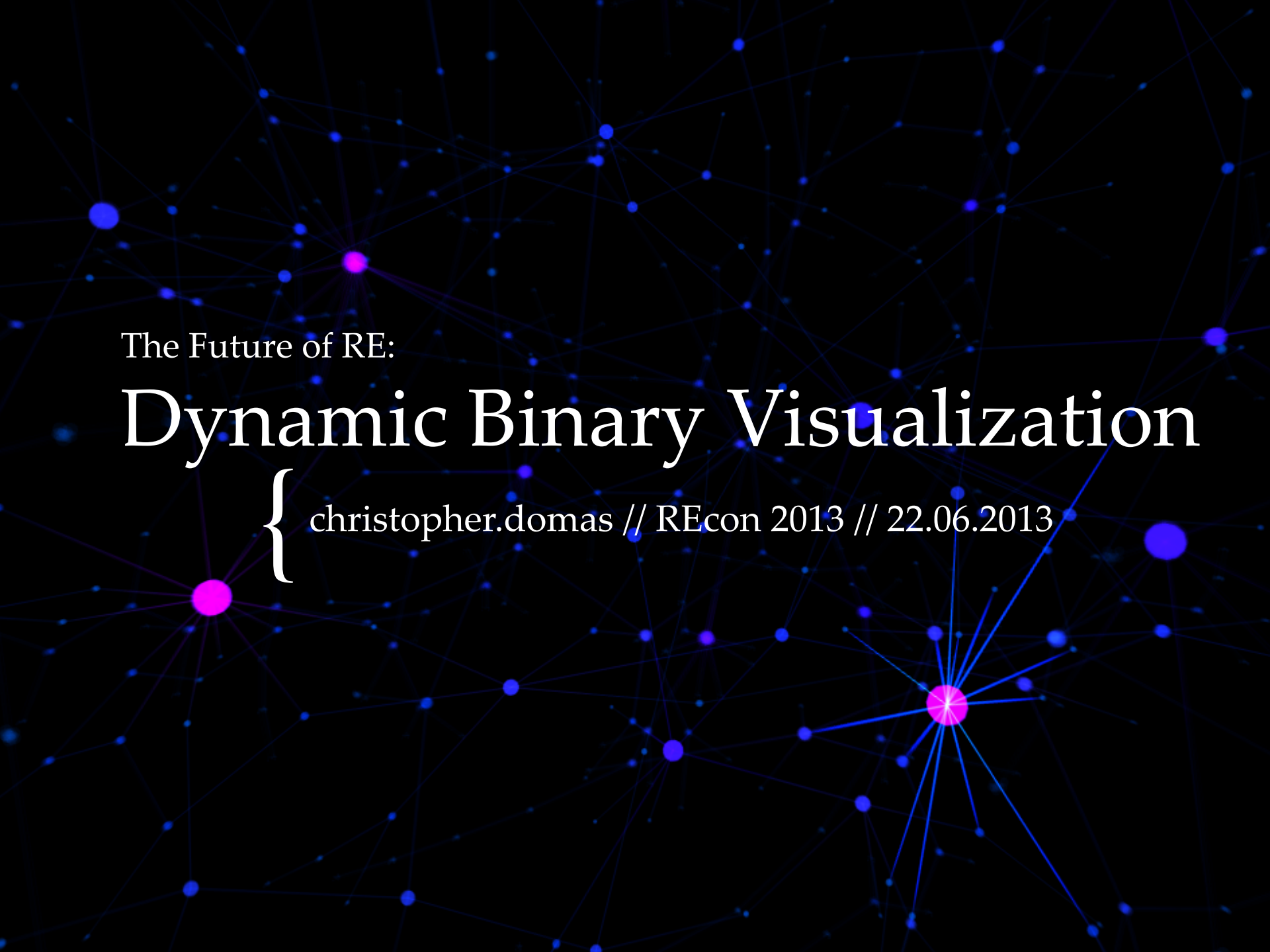


The Future of RE:

Dynamic Binary Visualization

{ christopher.domas // REcon 2013 // 22.06.2013

Myself

- Myself
  - Chris Domas
  - Embedded Systems Engineer
  - Cyber Innovation Unit
  - National Security Division
  - The Battelle Memorial Institute
- Battelle
  - World's largest non-profit R&D organization
  - Manages leading national laboratories
  - Awesome

- Information Analysis

Reverse Engineering

- What IS this?

In other words

- How we conceptualize binary information
- How we use our conceptualization for analysis

The ever-present issues

```
66 52 66 8B D0 FA B4 1B D0 C0
66 D1 C8 E6 74 E4 75 F6 C4 80
74 08 86 C4 E6 74 E4 75 86 C4
8A E6 66 5A C3 66 52 66 8B D0
86 C4 B4 1B D0 C0 66 D1 C8 FA
66 50 8A E0 8A C2 86 C4 E6 74
86 C4 E6 75 66 58 F6 C4 80 74
06 86 C4 E6 74 E4 75 66 8B C2
66 5A C3 E6 72 E4 73 C3 66 50
86 C4 E6 72 86 C4 E6 73 66 58
C3 66 9C 66 50 66 51 FA 66 8B
CB 66 33 DB 8A C1 E8 87 FF FF
FF 02 D8 80 D7 00 FE C1 38 E9
76 EE 66 59 66 58 66 9D C3 66
50 66 53 80 E4 80 B3 10 0A DC
```

- Flexible
- Exact
- Complex

```
push    dx
mov     dx, ax
cli
mov     ah, 1Bh
rol     al, 1
ror     ax, 1
out     74h, al
in      al, 75h
test    ah, 80h
jz      $+0x0A
xchg    al, ah
out     74h, al
in      al, 75h
xchg    al, ah
mov     ah, dh
```

- Rigid
- Rules
- Succinct

The RE Dichotomy

```
xor     edx, edx        ; int
lea     rcx, [rsp+78h]  ; void *
mov     [rsp+arg_68], ebx
lea     r8d, [rdx+60h]  ; size_t
call    memset
xor     ecx, ecx        ; hWnd
call    cs:GetDC
mov     rdi, rax
cmp     rax, rbx
jz      loc_1000015A4
lea     rsi, stru_1000101A0
mov     edx, 5Ah        ; int
mov     rcx, rax        ; HDC
mov     [rsp+arg_68], 68h
mov     [rsp+78h], rbp
mov     [rsp+88h], rsi
call    cs:GetDeviceCaps
mov     ecx, cs:nNumber
mov     r8d, 2D0h
mov     edx, eax
call    cs:MulDiv
mov     rdx, rdi        ; hDC
xor     ecx, ecx        ; hWnd
neg     eax
mov     dword ptr [rsp+94h], 1000041h
mov     eax, 2000h
mov     [rsp+0C8h], ax
call    cs:ReleaseDC
call    cs:ChooseFontW
cmp     eax, ebx
jz      loc_1000015A4
mov     rcx, cs:hCursor ; hCursor
call    cs:SetCursor
mov     rcx, rsi        ; LOGFONTW *
call    cs:CreateFontIndirectW
mov     rdi
cmp     rax, rbx
jz      short loc_10000158C
mov     rcx, cs:wParam  ; HGDIOBJ
call    cs:DeleteObject
mov     rcx, cs:qword_1000100A0 ; hWnd
mov     r9, r14         ; lParam
mov     r8, rdi         ; wParam
mov     edx, 30h        ; Msg
mov     cs:wParam, rdi
```

- What does this do?
- Where's the vulnerability?
- Reverse engineering as an *art*

Investigating the concept

- Let's fix this.
  - Bridge the analysis gap
  - Change RE from an ART to a SCIENCE

Reverse Engineering

- Visual RE
  - Address the conceptualization
- Statistical Exploration
  - Address the analysis

How?

Visual RE

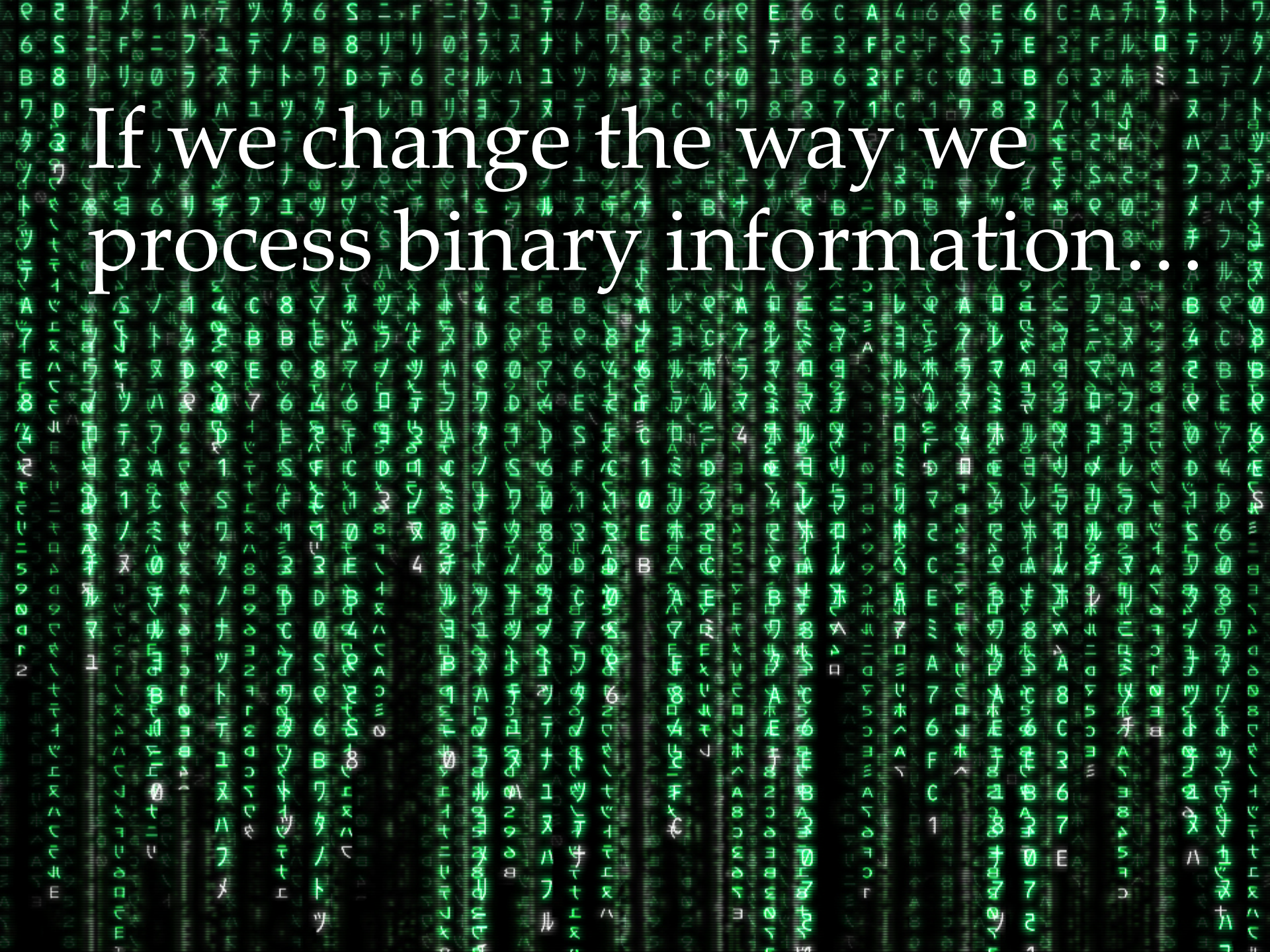
- Idea:
  - Take a computationally difficult task
  - Translate it to a problem our brains do naturally

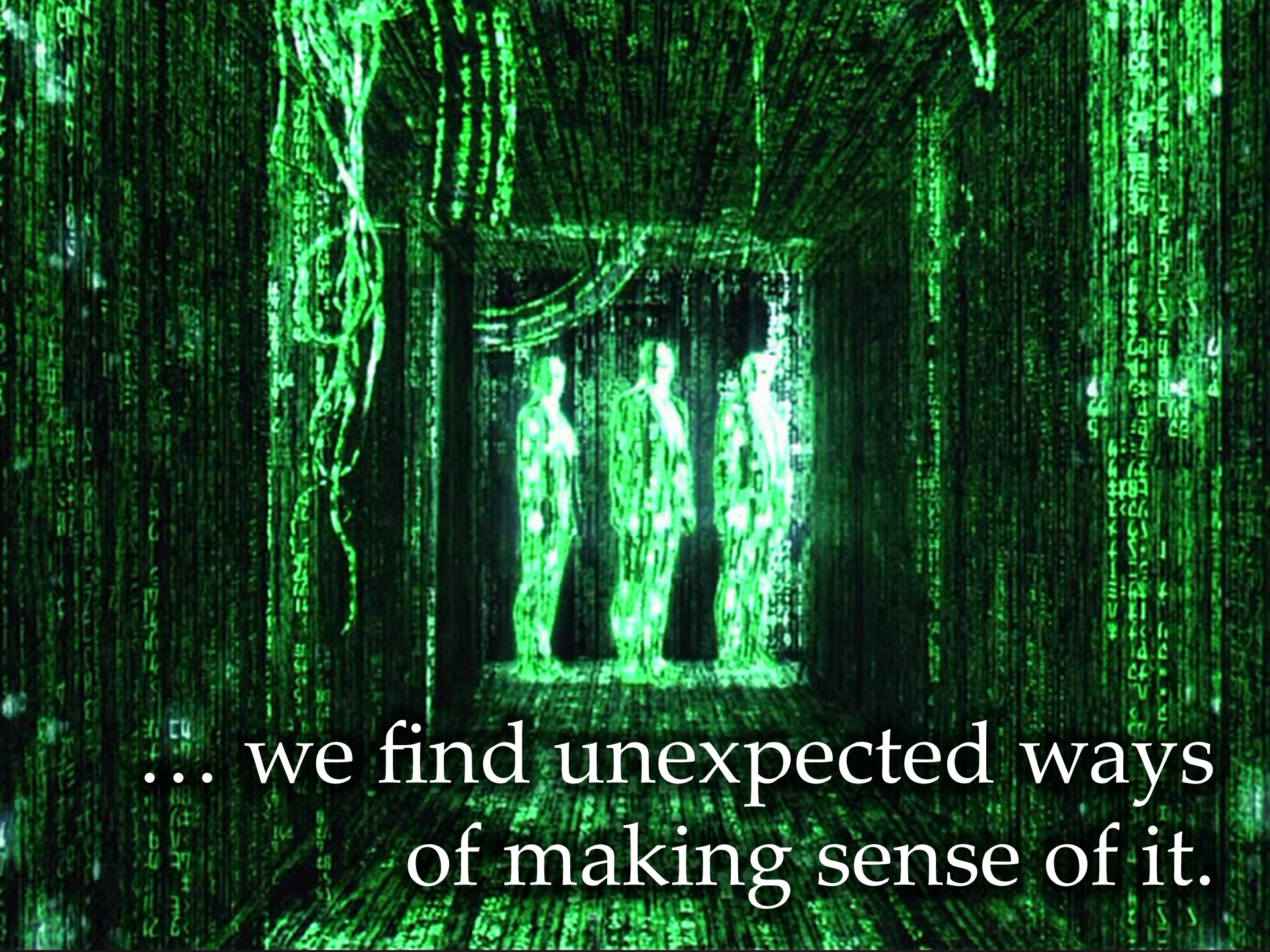
- In other words
  - how to traverse a hundred thousand pages of

```
3E 2E 0F 84 AE 33 00 00 8B F3 EB 00 4C 8B AC 24 90
B9 04 01 00 00 48 8D 15 53 C9 00 00 4D 8B C4 4C 2B
3B C3 74 0D 66 89 02 48 83 C2 02 48 83 E9 01 75 DD
FF 89 1D E0 BD 00 00 C7 05 0A BE 00 00 01 00 00 00
BE 00 00 FF 15 22 83 00 00 83 3D E7 BD 00 00 02 0F
00 00 00 48 8B 0D 25 BE 00 00 FF 15 E7 82 00 00 3B
48 8B 0D 04 BE 00 00 FF 15 D6 82 00 00 4C 8B C0 BA
00 45 33 C9 41 8D 51 0B 45 8D 41 01 48 8B 0D D4 BD
00 00 FF 15 F3 80 00 00 48 8B 0D B4 BD 00 00 FF 15
00 00 00 EB 00 48 8B 8C 24 10 03 00 00 48 33 CC E8
40 49 8B E3 41 5F 41 5E 41 5D 41 5C 5F C3 48 8D 54
94 24 A0 03 00 00 48 8B D8 48 83 F8 FF 0F 84 D4 12
```

in 20 seconds

…what?


If we change the way we process binary information…



… we find unexpected ways of making sense of it.

- obfuscated file headers
- no headers
- multiple binaries embedded in a single blob
- unique instruction sets
- proprietary data formats
- overwhelming complexity
- steganography
- memory dumps
- rapid RE
- triage
- firmware
- forensics

Why bother?

- Our best RE tools are completely dependent on known structure
  - Information is evolving faster than our tools can keep up
- Gate's Law
  - Software is getting slower more rapidly than hardware becomes faster
  - The amount of information we need to analyze is growing exponentially

Why bother?

- Greg Conti
  - United States Military Academy
  - Black Hat 2010
- Aldo Cortesi
  - Nullcube
  - corte.si

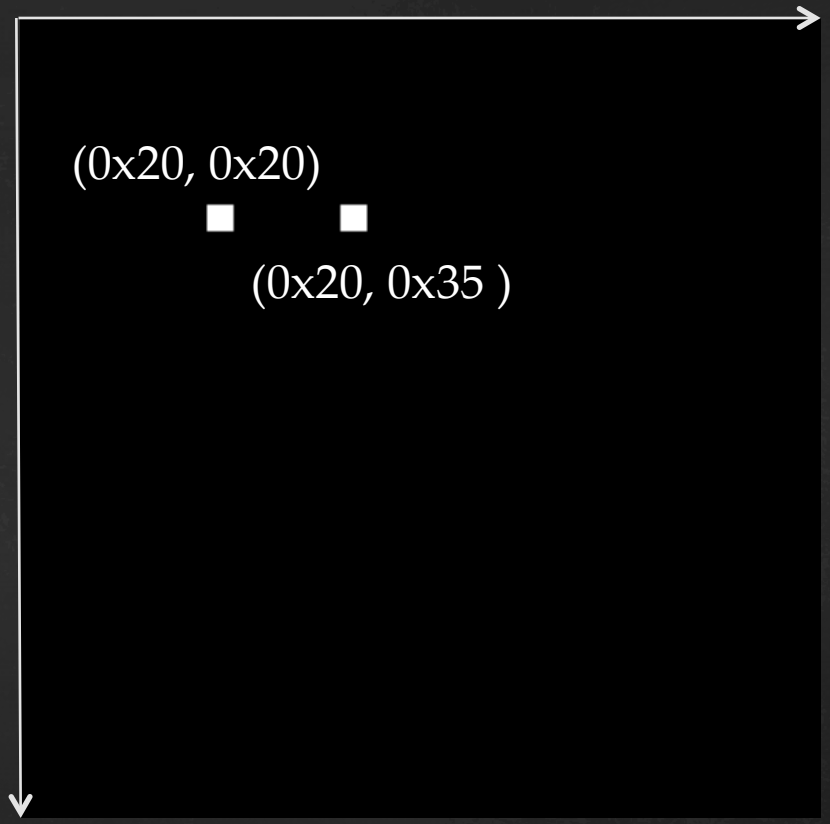
Some interesting ideas

Digraphs (conti)

- Idea:
  - Even in unstructured data, there is structure
  - Sequential bytes have an implicit relationship

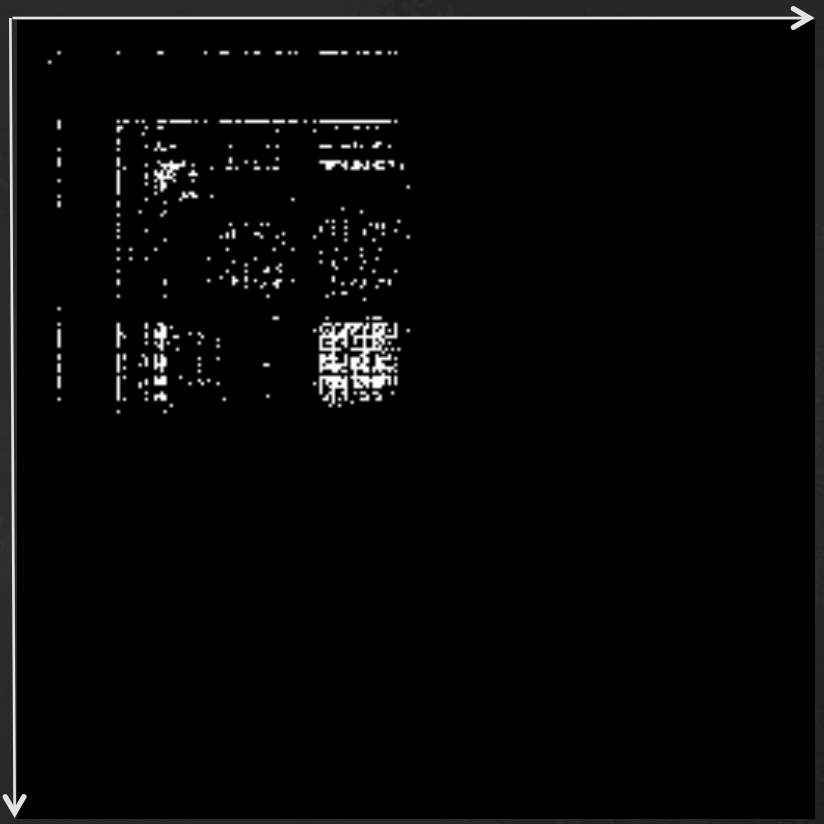
```
recon
re          (72,65)
 ec         (65,72)
  co        (63,6F)
   on       (6F,6E)
```

Digraphs (conti)



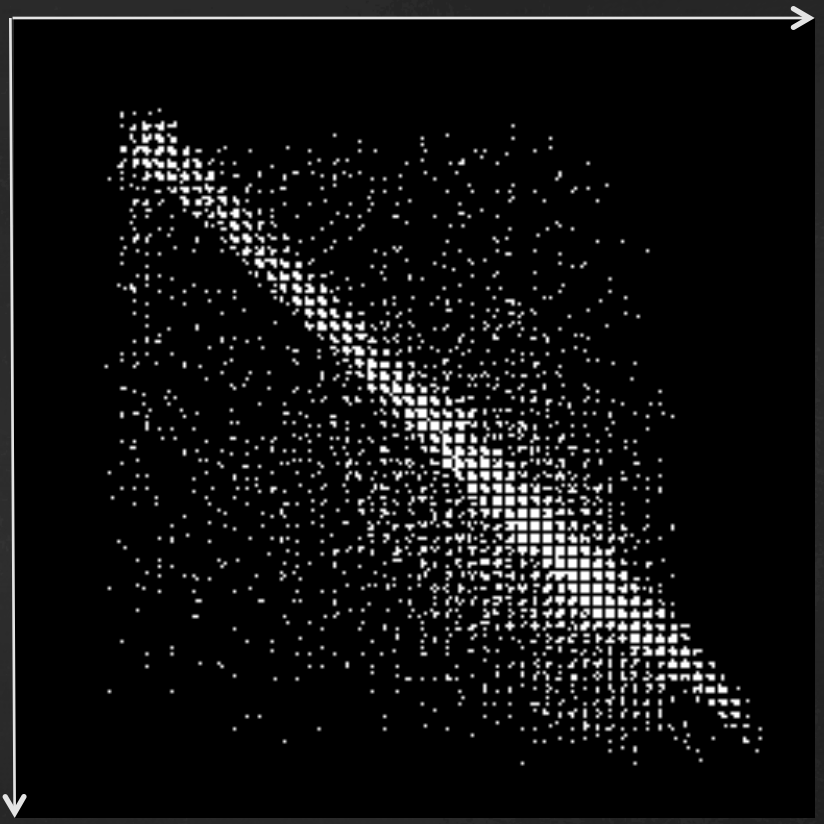
Digraphs (conti)

- ASCII



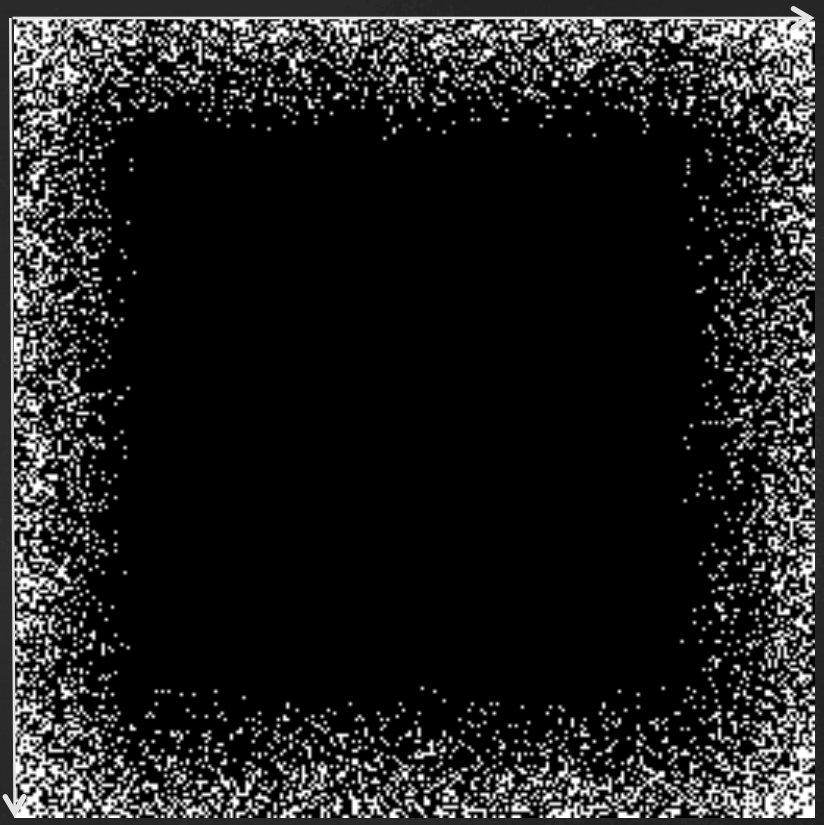
Digraphs (conti)

- Image



Digraphs (conti)

- Audio



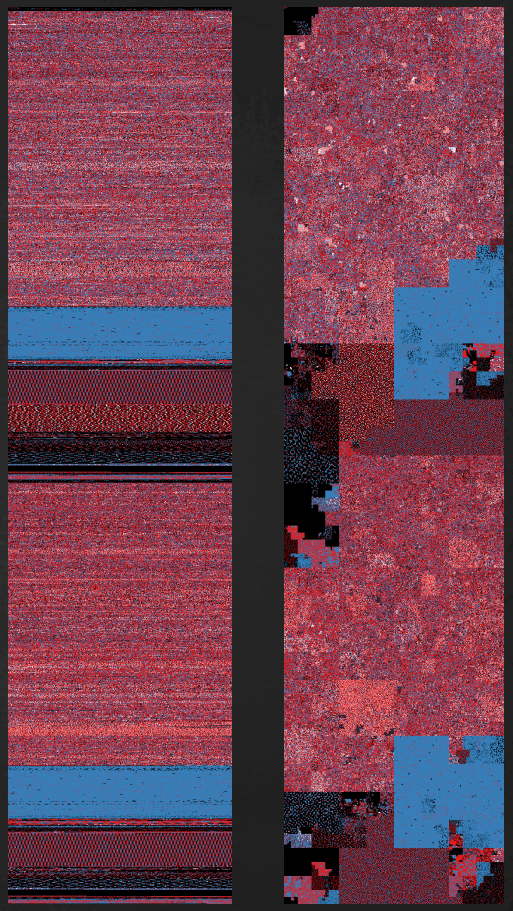
Digraphs (conti)

- Introducing the Hilbert Curve

- Continuous fractal space filling curve
- Maps 1D space to an alternate dimensional space
- *Preserves Locality*
- Unexpected utility in computer science

Hilbert Translation (cortesi)

- Traditional byte plot “Zig-Zag” order on left
- Hilbert byte plot on right
- Shaded by Byte Class



Hilbert Translation (cortesi)

- Build on these concepts
- Goal
  - understand data
  - *independent of format*
    - Unknown code
    - Format deviations
    - Proprietary structure
  - a priori

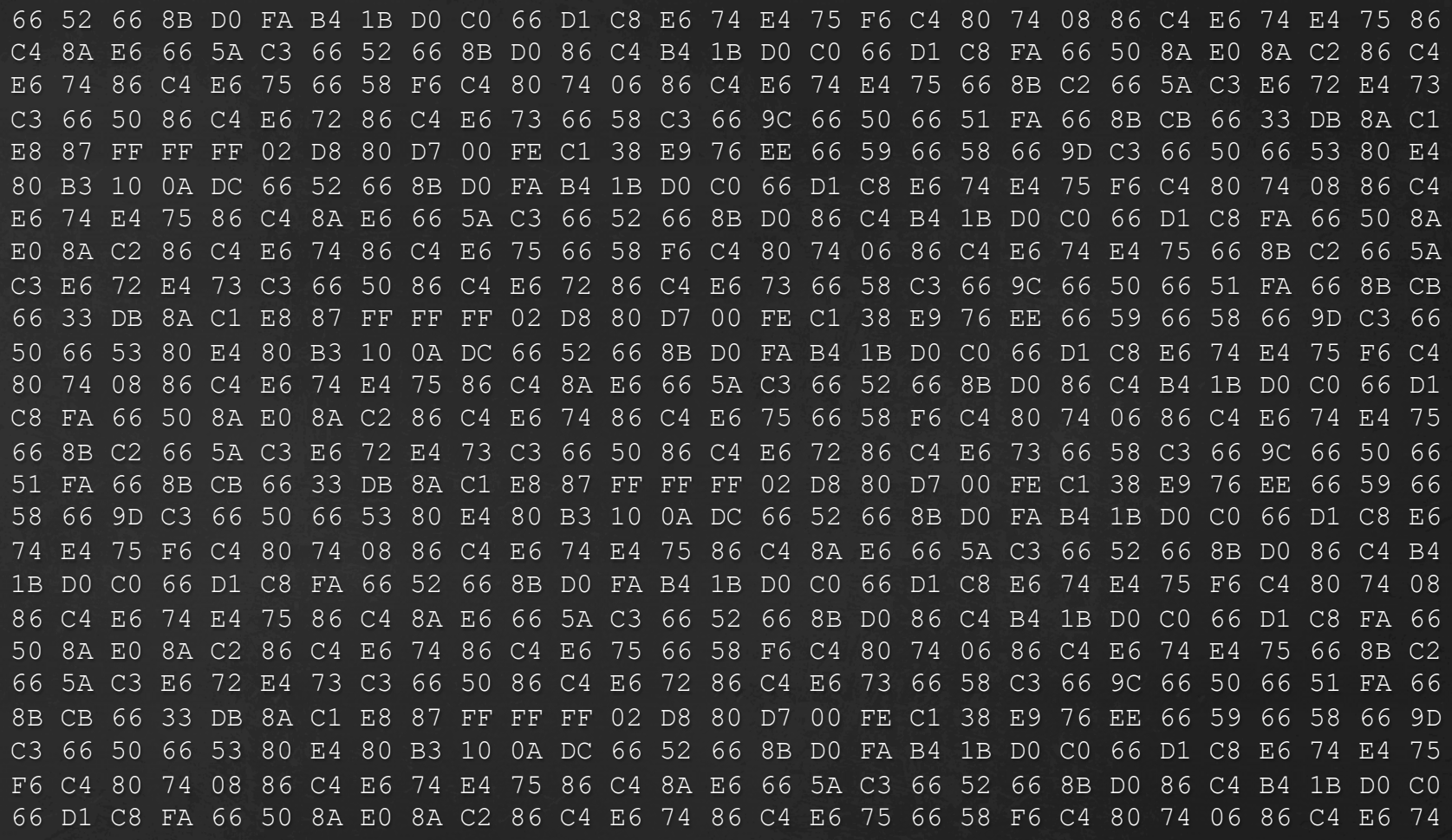
Our software

- ..cantor.dust..
- Named after this guy →
- Illustrating concepts
  - Doesn't need to be this software

Introducing ..cantor.dust..

- An embarrassing mistake

..cantor.dust.. // background



..cantor.dust.. // background

- An embarrassing mistake
- Spare time

..cantor.dust.. // background

- Interface
- Patterns

..cantor.dust.. // demo

- Visualizations
  - Tuple systems
    - Why
    - Time/Space
    - Coordinate Systems
    - Fun

..cantor.dust.. // demo

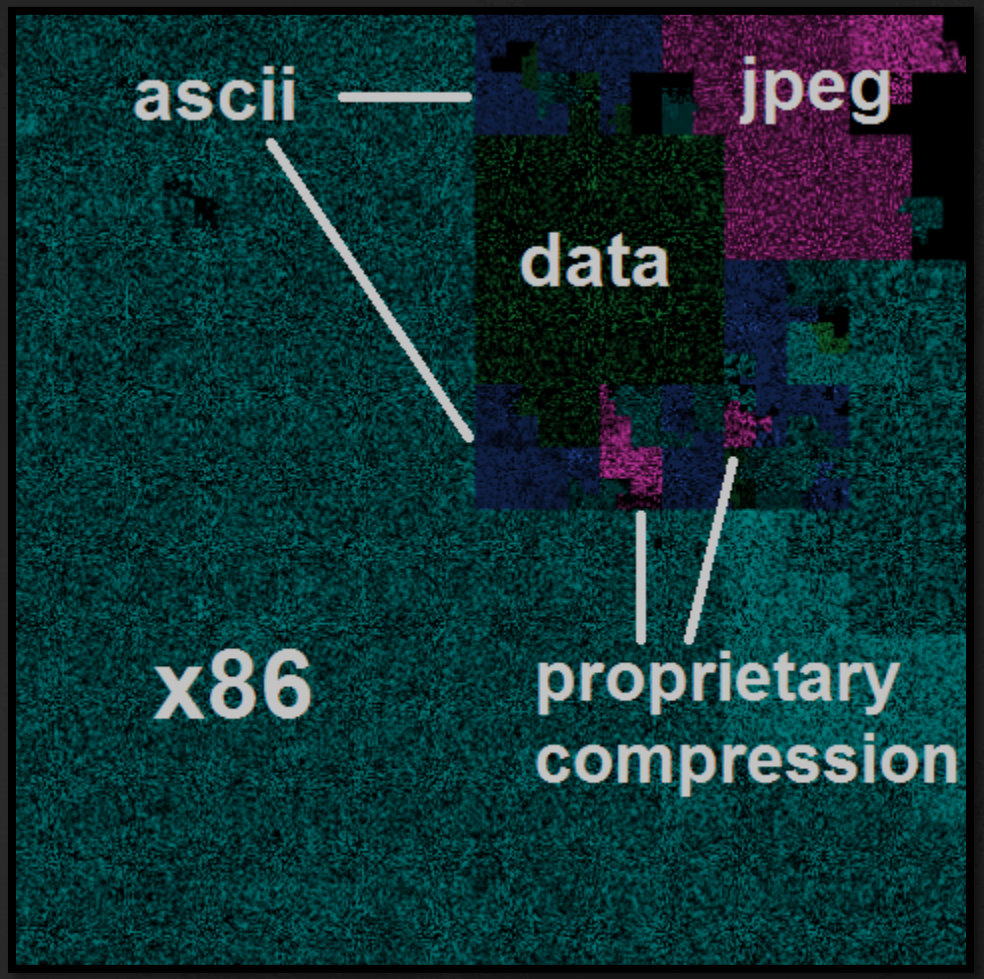
- Visualizations
  - Metric map
    - Entropy
      - Uses
        - Encryption Keys
        - Packing
        - Obfuscation
      - Examples
        - Malware
        - Key check

..cantor.dust.. // demo

- Notepad.exe dissection

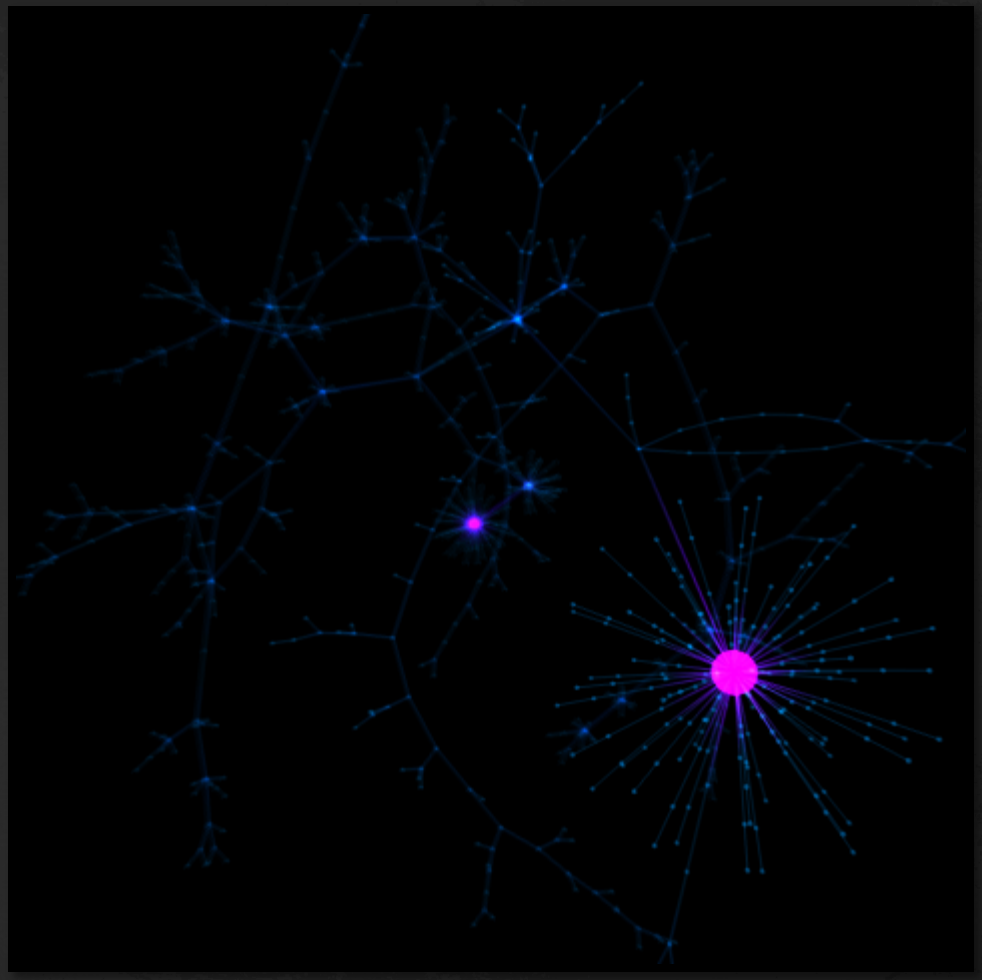
..cantor.dust.. // demo

- Binary region identification through statistical analysis
  - Naïve Bayes classification of n-gram models
  - Identify regions of an object based on supplied templates
  - Uses:
    - Custom/proprietary formats
    - Unique Instruction Sets
    - Data/code features



..cantor.dust.. // classificaton

- Conventional parsing
  - Recursive descent
  - Linear sweep
- A new approach?
  - “Probabilistic parsing”
  - Identify structure based on statistical *patterns*, not grammar definitions



..cantor.dust.. // parsing

- “Attacking Intel BIOS”
  - Invisible Things Lab
  - Rafal Wojtczuk and Alexander Tereshkin
  - Black Hat 2009

..cantor.dust.. // case study

- Goal:
  - Bypass EFI signing protection

..cantor.dust.. // case study

- Background:
  - EFI is a BIOS replacement
  - Update images can be signed and checked before they are flashed

..cantor.dust.. // case study

- Background
  - Update image “envelope”
    - Signed module
    - Signed module
    - Signed module
    - Unsigned module

..cantor.dust.. // case study

- Background
  - Update image “envelope”
    - Signed module
    - Signed module
    - Signed module
    - Unsigned module ← Attack Vector!

..cantor.dust.. // case study

- Why have an unsigned module?
  - Boot Splash Logo
    - Can be changed by the OEMs

..cantor.dust.. // case study

- Exploit!
  - Goal: Unsigned Code Execution
    - Can't update code, can update bitmap
  - Vulnerability identified in bitmap parser for splash screen
    - In EFI template code used by most IBVs
  - Flash the bitmap with an invalid image
    - Overflow with width & height
  - Cause an overflow, get execution

..cantor.dust.. // case study

- Observations
  - No one is really “EFI”
    - There’s just “more” or “less” EFI for now
    - This means
      - Less structure
      - More headache

..cantor.dust.. // case study

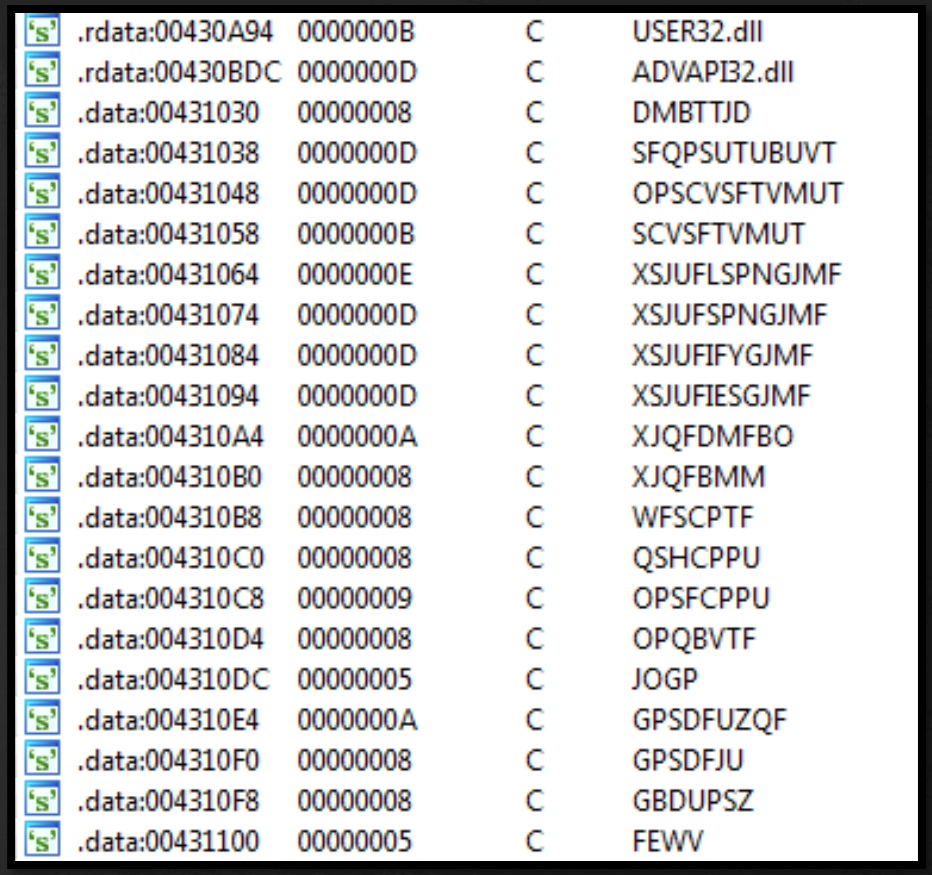
- ITL covered the exploit
- Let's figure out everything else

..cantor.dust.. // case study

- Choose a victim
  - Targeting my junk laptop
  - Don't care if I brick it
- Grab the update executable from vendor's website

case study // step 1

- Quick RE to extract the firmware image
  - Ultra-secret secure flags have been encrypted by adding 1 to each character
  - We're looking for "WRITEROMFILE"
  - a.k.a. "XSJUFSPNGJMF"



.rdata:00430A94	0000000B	C	USER32.dll
.rdata:00430BDC	0000000D	C	ADVAPI32.dll
.data:00431030	00000008	C	DMBTTJD
.data:00431038	0000000D	C	SFQPSUTUBUVT
.data:00431048	0000000D	C	OPSCVSFTVMUT
.data:00431058	0000000B	C	SCVSFTVMUT
.data:00431064	0000000E	C	XSJUFLSPNGJMF
.data:00431074	0000000D	C	XSJUFSPNGJMF
.data:00431084	0000000D	C	XSJUFIFYGJMF
.data:00431094	0000000D	C	XSJUFIESGJMF
.data:004310A4	0000000A	C	XJQFDMFBO
.data:004310B0	00000008	C	XJQFBMM
.data:004310B8	00000008	C	WFSCPTF
.data:004310C0	00000008	C	QSHCPPU
.data:004310C8	00000009	C	OPSFCPPU
.data:004310D4	00000008	C	OPQBVTF
.data:004310DC	00000005	C	JOGP
.data:004310E4	0000000A	C	GPSDFUZQF
.data:004310F0	00000008	C	GPSDFJU
.data:004310F8	00000008	C	GBDUPSZ
.data:00431100	00000005	C	FEWV

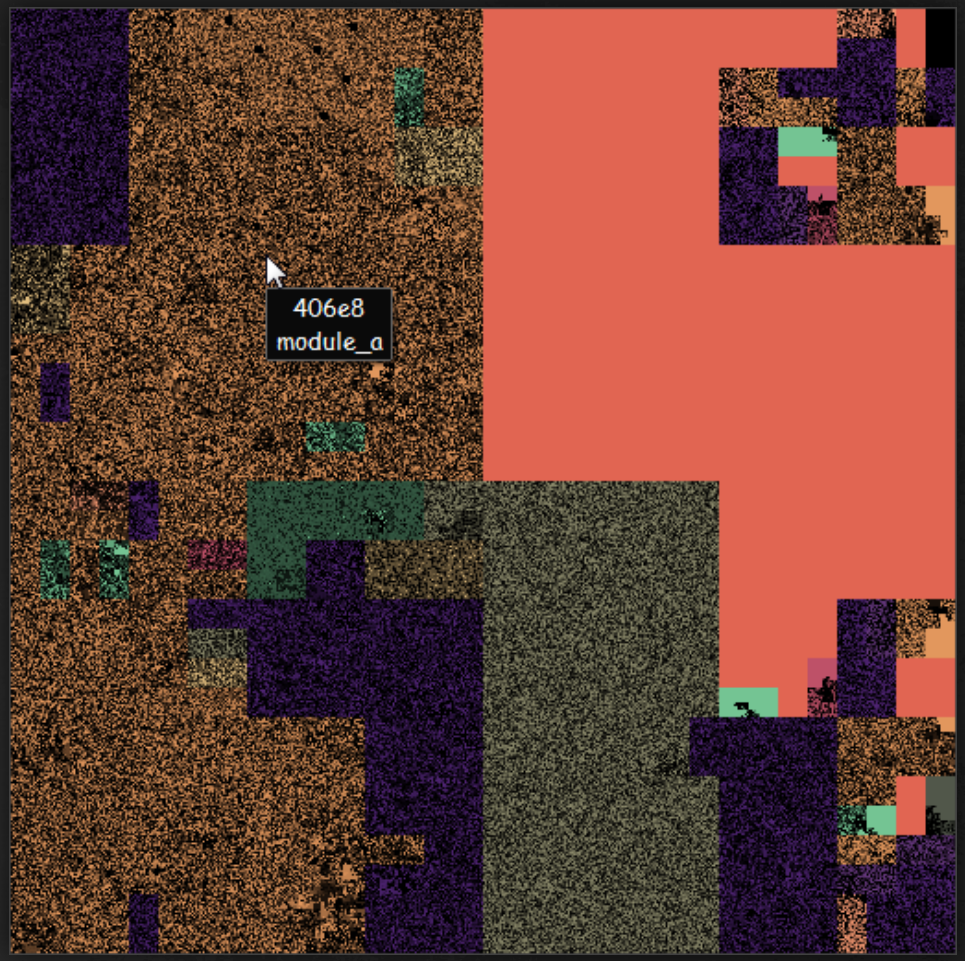
case study // step 2

- RE image
  - Find a custom decompression routine
  - But only one module
  - Where are the others?

```
decomp_module proc near
        push    edi
        add     edx, esi

loc_FFFF90D7:

        xor     ecx, ecx
        cmp     esi, edx
        jnb     short loc_FFFF9140
        lodsb
        test    al, 0Fh
        jnz     short loc_FFFF9108
        cmp     al, 0E0h ; 'a'
        jz      short loc_FFFF90FC
        cmp     al, 0F0h ; '='
        jz      short loc_FFFF90F4
        shr     al, 4
        mov     cl, al
        inc     ecx

loc_FFFF90F0:
        rep movsb
        jmp     short loc_FFFF90D7
```

case study // step 3

- Use ..cantor.dust..
  - Crop out the known module
  - Use as a template for statistical analysis
  - Identify regions of the binary image with patterns that match the template



case study // step 4

- Decompress the located modules
  - Use Chris Eagle's x86emu
  - No need to understand the algorithm

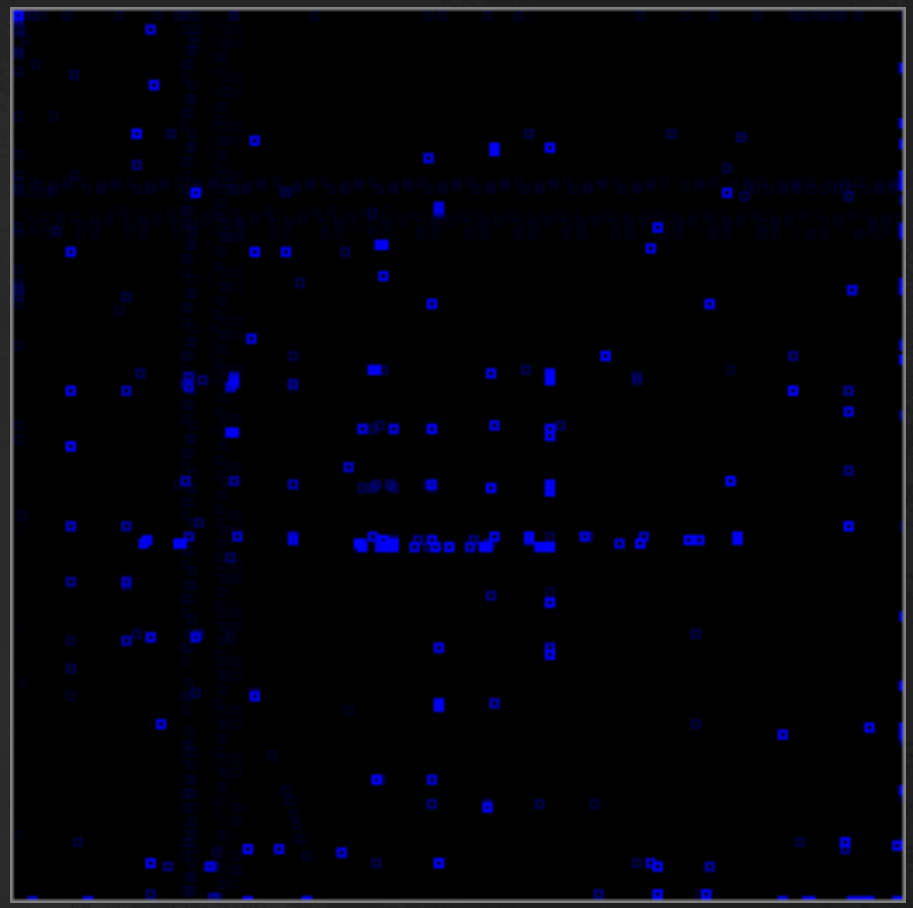
case study // step 5

- Modules contain proprietary headers

```
root@deltaleph-ubuntu:~/modules# file *
mod_01: data
mod_02: data
mod_03: data
mod_04: DOS executable (COM)
mod_07: data
mod_09: 8086 relocatable (Microsoft)
mod_0c: data
mod_0d: data
mod_0e: data
mod_0f: data
mod_10: data
mod_11: data
mod_13: data
mod_14: data
mod_16: data
mod_19: data
mod_1a: data
mod_25: data
mod_3c: data
mod_3f: data
mod_40: data
mod_41: data
mod_42: data
mod_43: data
mod_45: data
mod_48: data
mod_49: data
```

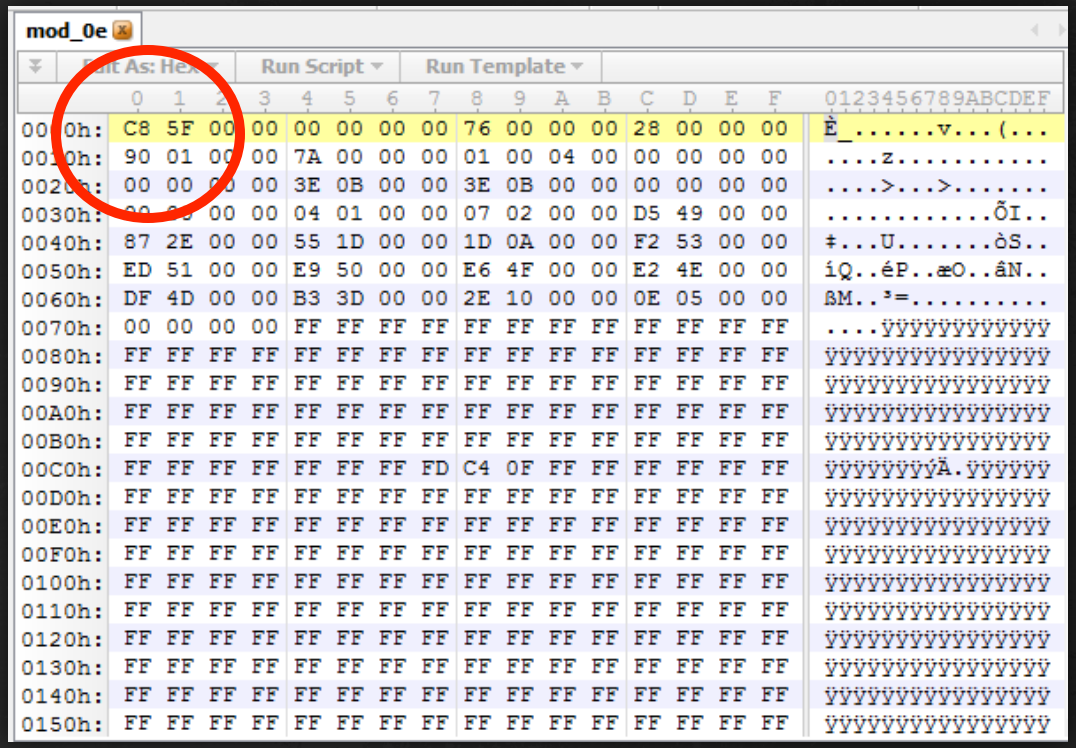
case study // step 6

- How do we find the splash screen?
  - Too time consuming at a binary level!
  - Use visual abstractions to locate the “bitmap data” fingerprint



case study // step 6

- Examine in a hex editor – follows the bitmap format, minus the ‘BM’ signature
- Easy to find width/ height



case study // step 7

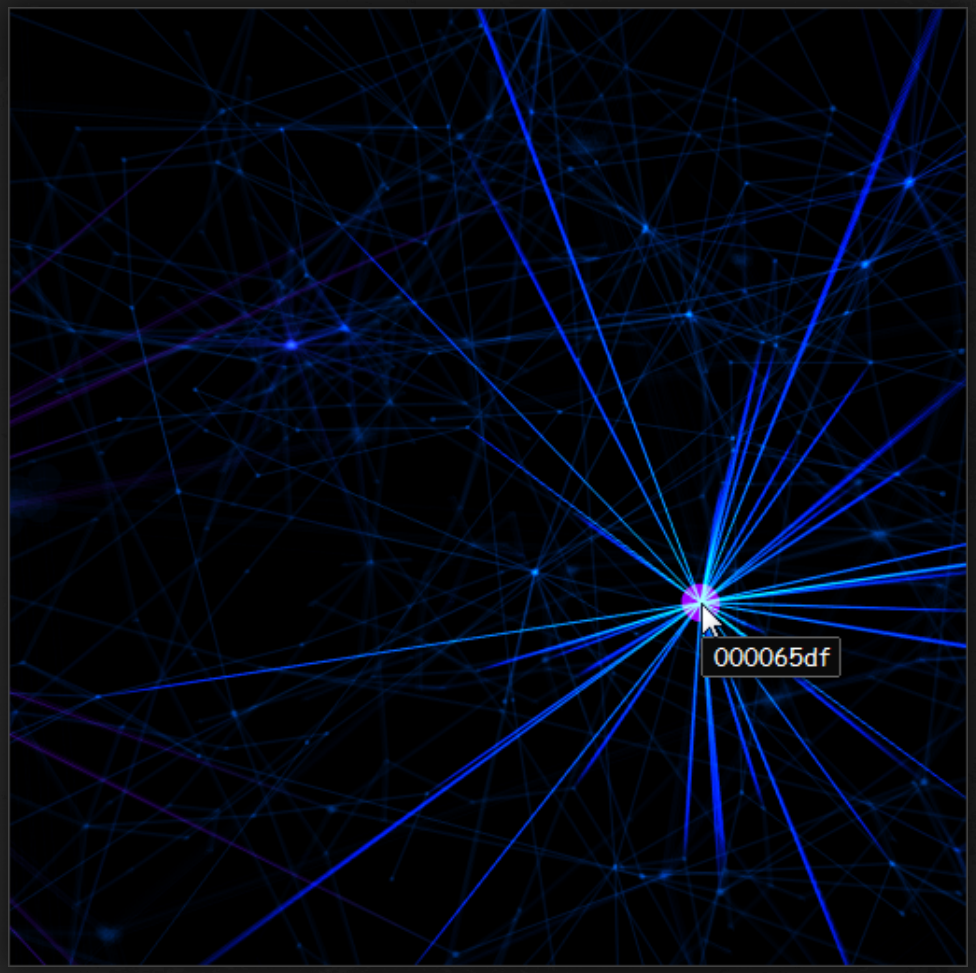
- Repackage the splash screen with modified information

case study // step 8

- Try to flash
- Image is not accepted
  - Modules are signed
  - Bitmap is not
  - “Envelope” is checksum’d
  - Need to fix the checksum
- Drop first module into IDA
  - There’s no known structure to this module
  - IDA tells us nothing

case study // step 9

- Use probabilistic parsing to identify key functions
- Find debug_printf
- Find calls to debug_printf
- Isolate CRC checksum routine
  - Identify CRC algorithm
  - Locate checksum position in image



case study // step 10

- Repackage the image with proper checksum
- Success!

case study // step X

- Without ..cantor.dust..?
  - Took me 37 hours

case study // results

- Hope I've shown…

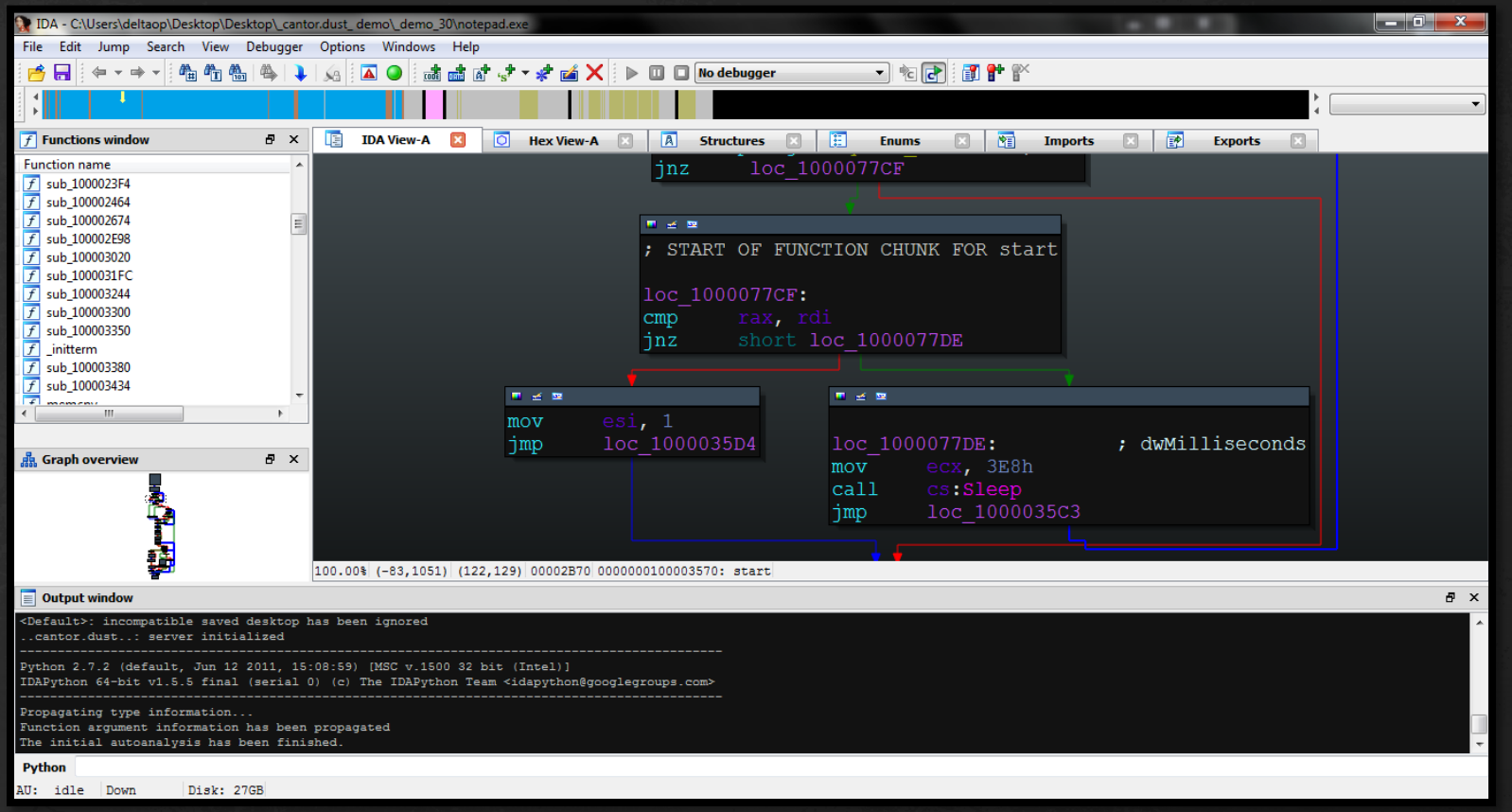
case study // results

- Useful:
  - User interface

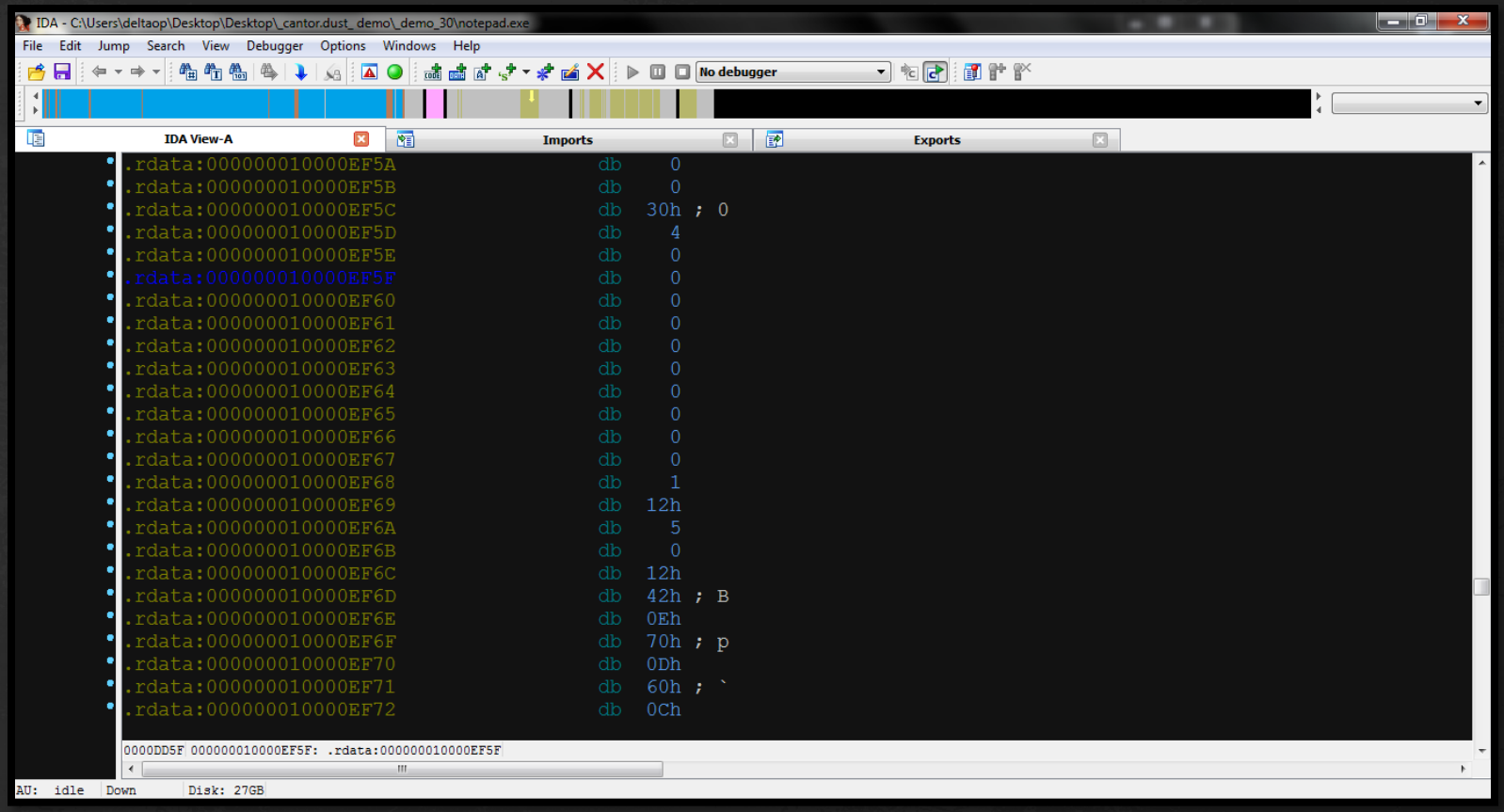
..cantor.dust.. // future

- Useful:
  - Complete integration

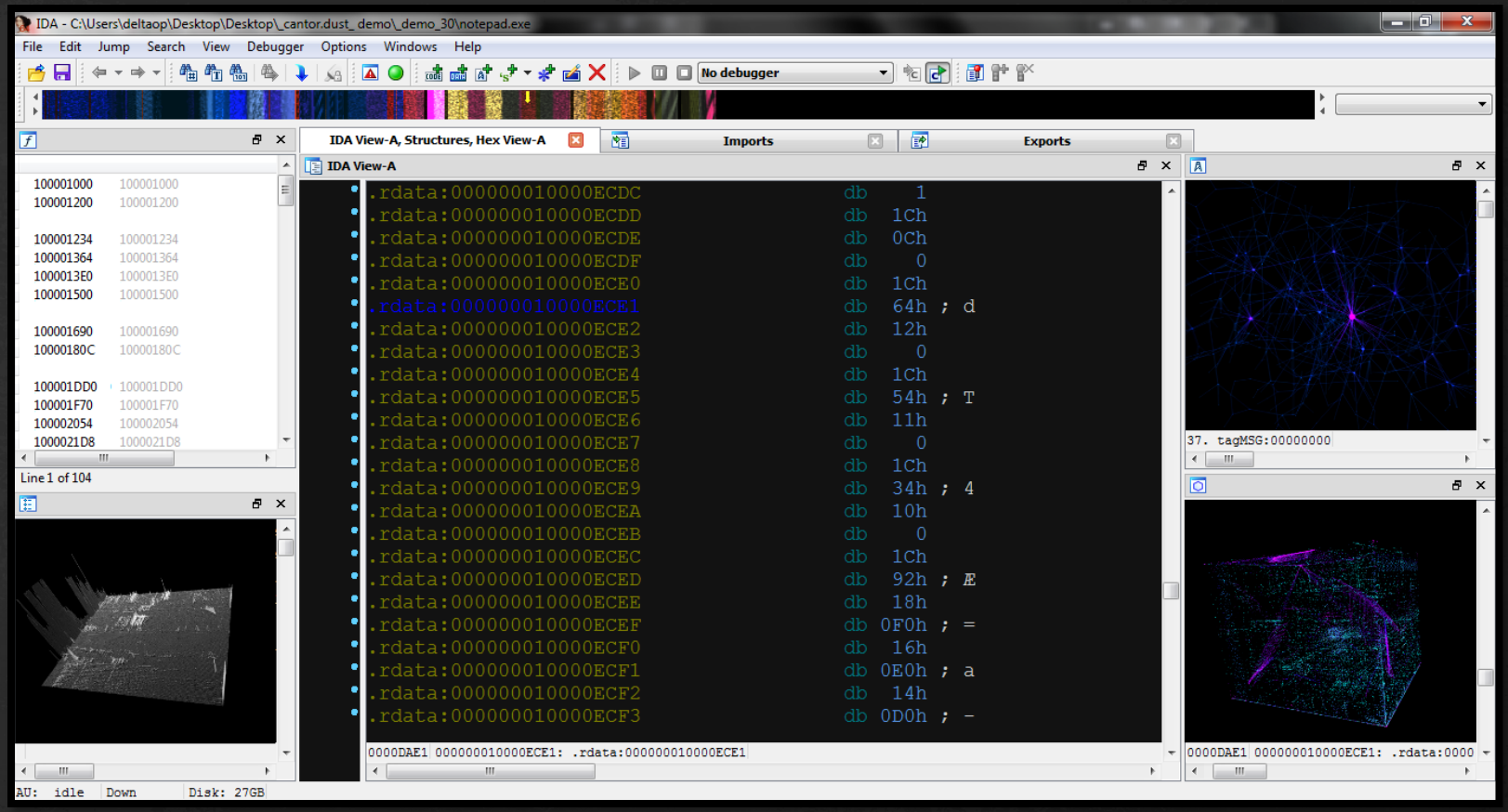
..cantor.dust.. // future



..cantor.dust.. // future



..cantor.dust.. // future

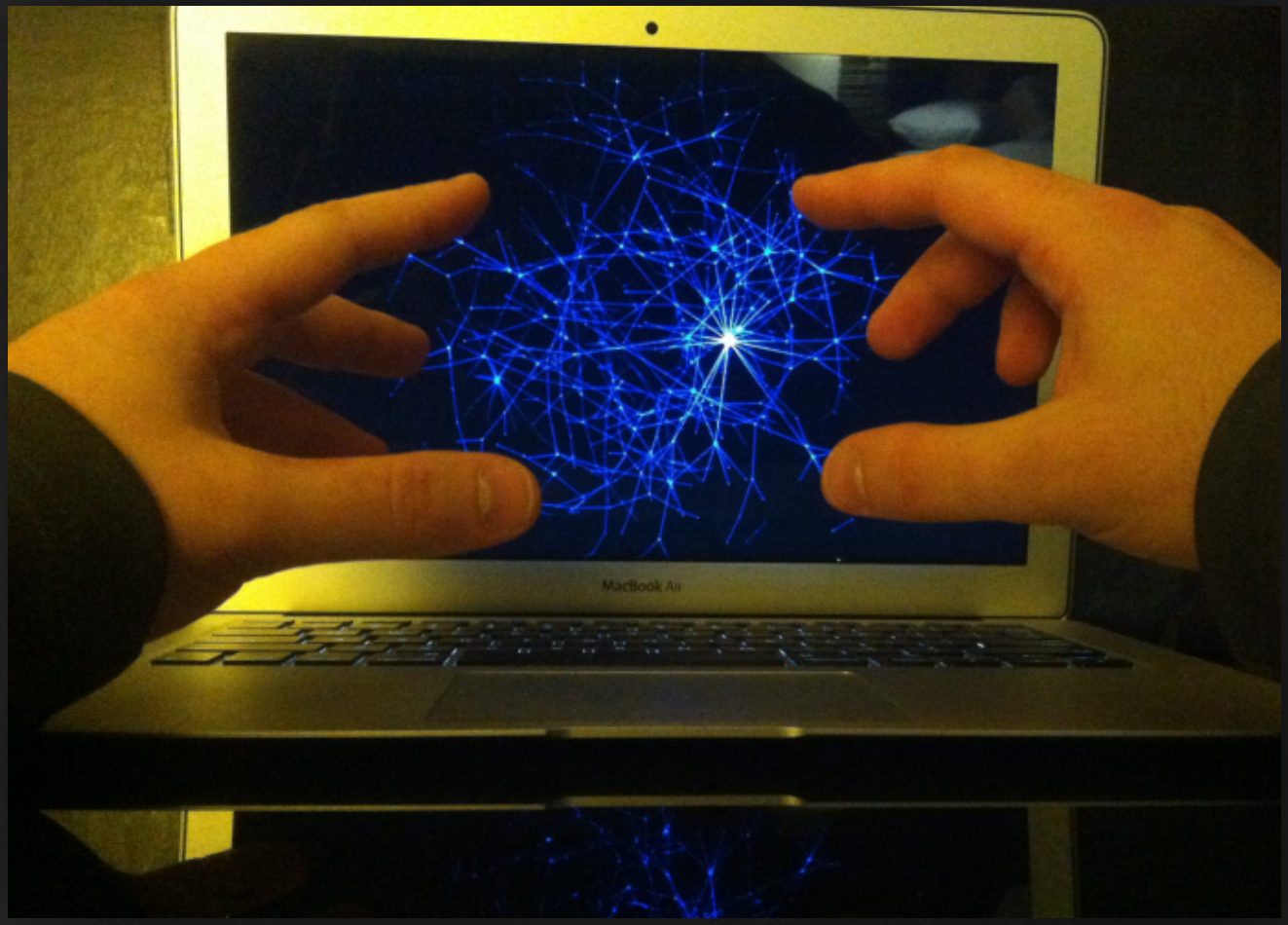


..cantor.dust.. // future

- Pointless:
  - Hands free

..cantor.dust.. // future

..cantor.dust.. // future

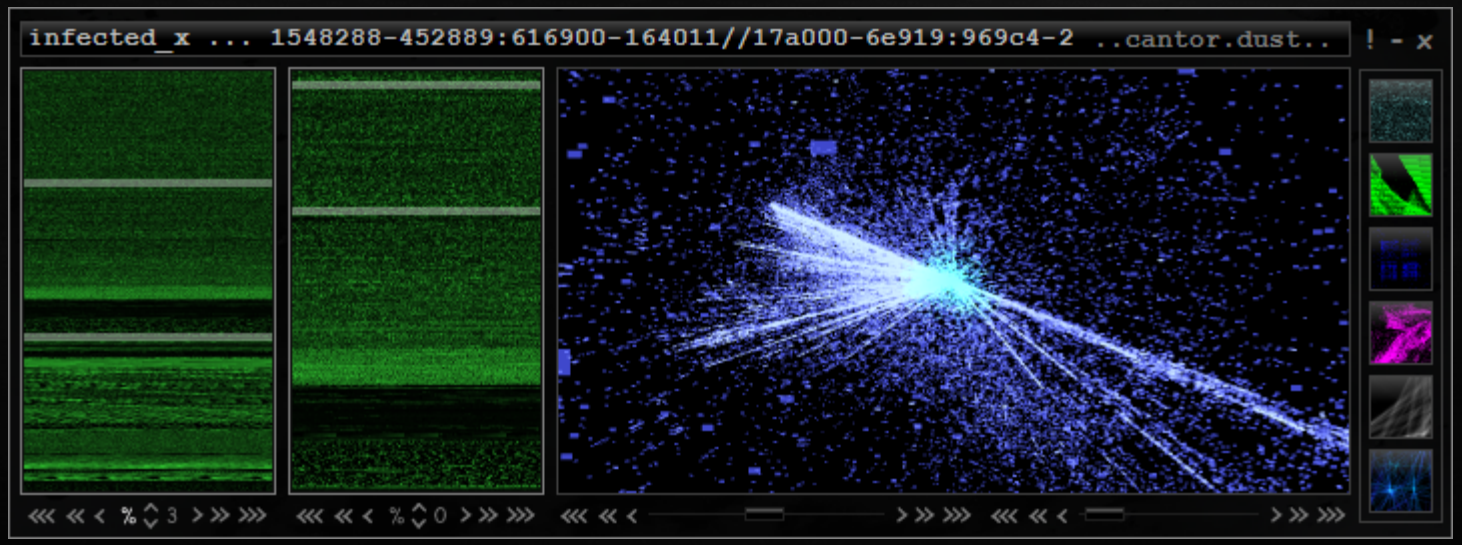


..cantor.dust.. // future

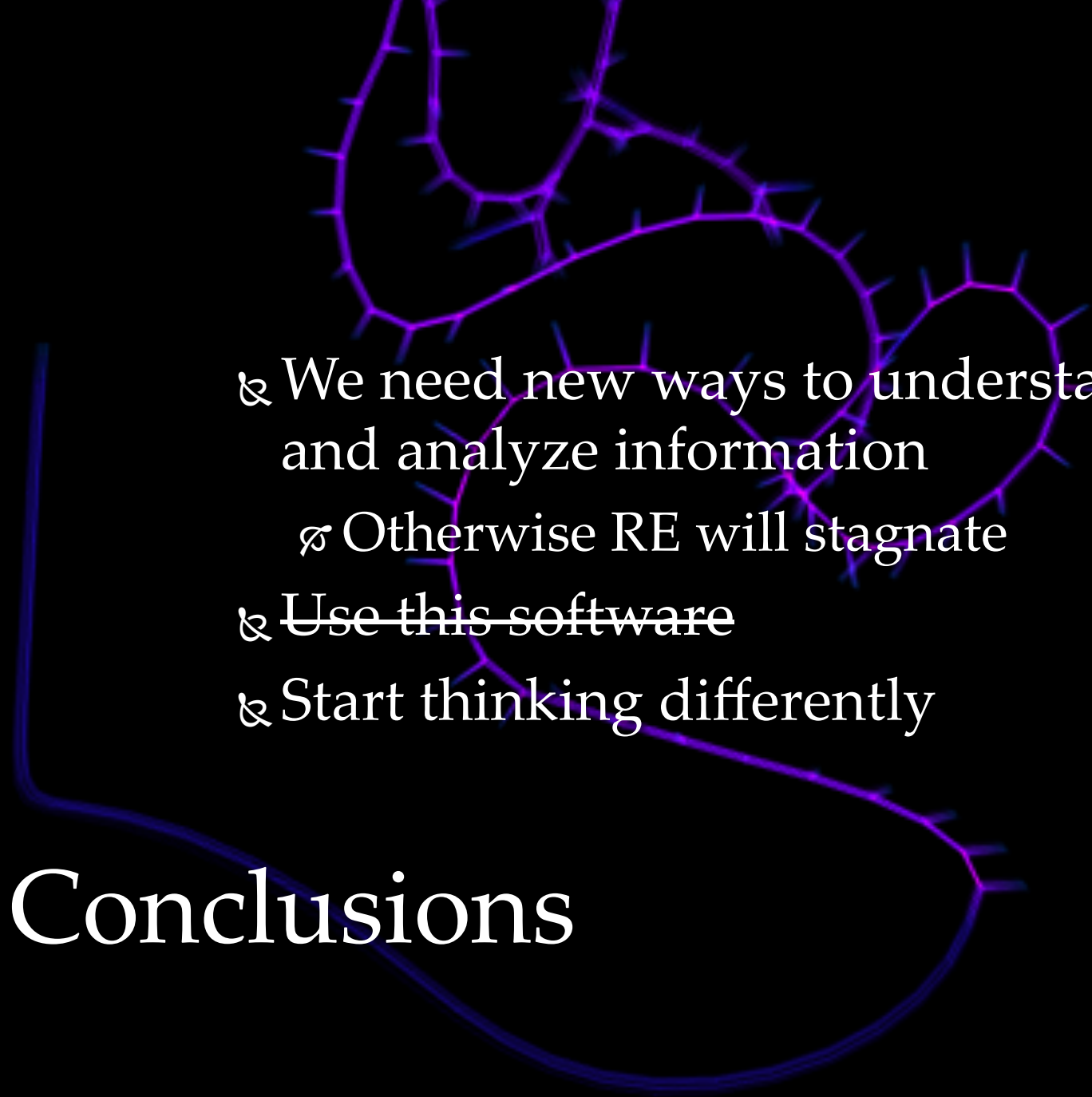
..cantor.dust.. // future

..minority.report..

- We need new ways to understand and analyze information
  - Otherwise RE will stagnate



Conclusions

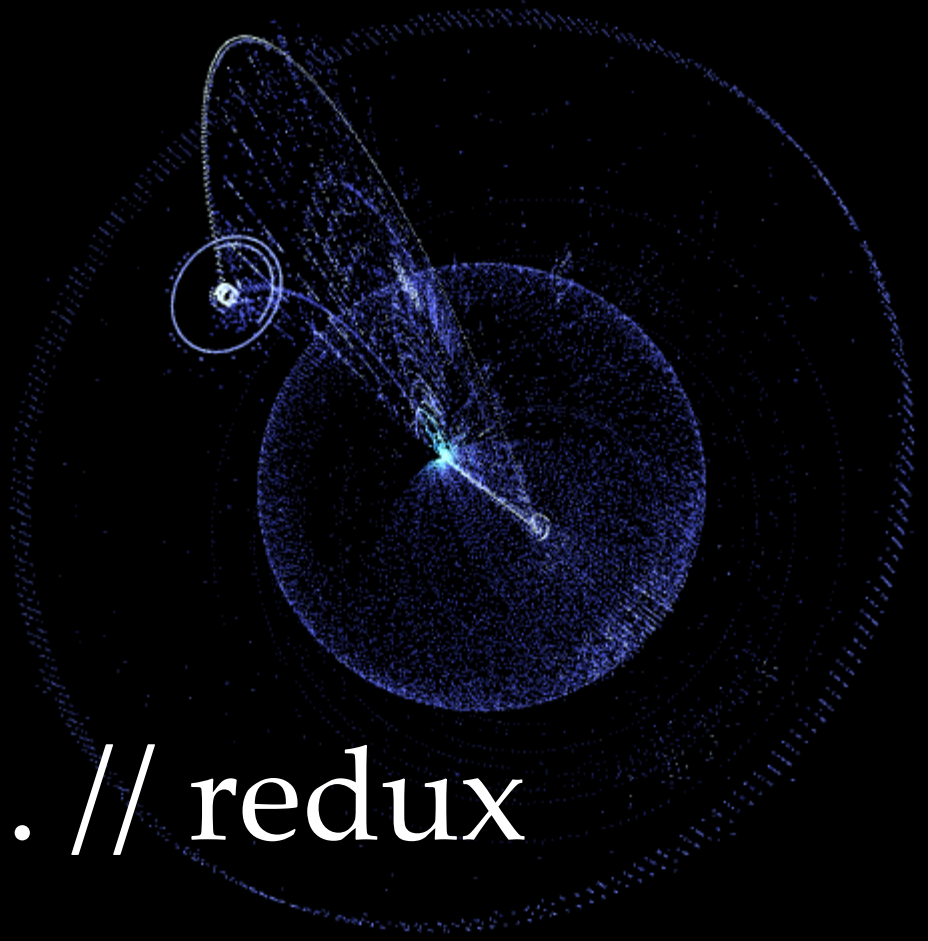


Conclusions

- We need new ways to understand and analyze information
  - Otherwise RE will stagnate
- ~~Use this software~~
- Start thinking differently

- Getting a copy
  - sites.google.com/site/xxcantorxdustxx/home
  - the.delta.axiom@gmail.com
  - hammera@battelle.org
- We've just gotten started

Thank you



..cantor.dust.. // redux