Rogue Behavior Detection

Tackling binaries while they‘re on the ground

G DATA ADVANCED ANALYTICS

Marion Marschalek

Malware Researcher
G DATA Advanced Analytics

marion.marschalek@gdata-adan.de
@pinkflawd

G DATA | ADVANCED ANALYTICS

Malware Analysts

And their issues...

Malware analysis and its issues

The average malicious binary is not interesting

- *Repetitive code*
- *Repetitive techniques*
- *Self-taught developers*
- *Limited interests*

Wouldn‘t it be neat to see at one glance roughly what a binary is about?

Limitations of contemporary automated malware analysis

Static

Obfuscation

Self-modifying code

Byte code and virtual machines

Dynamic API loading

Asynchronous code

Object oriented code

Dynamic

Sandbox detection

Missing dependencies/components

Need for interaction

Time based evasion

Missing input values

Multiple execution paths

Incompatibilities

Multiple execution paths

Common sandboxes are fairly limited in their analysis capabilities of multi-purpose malware

In almost all cases they are totally useless for analyzing benign binaries

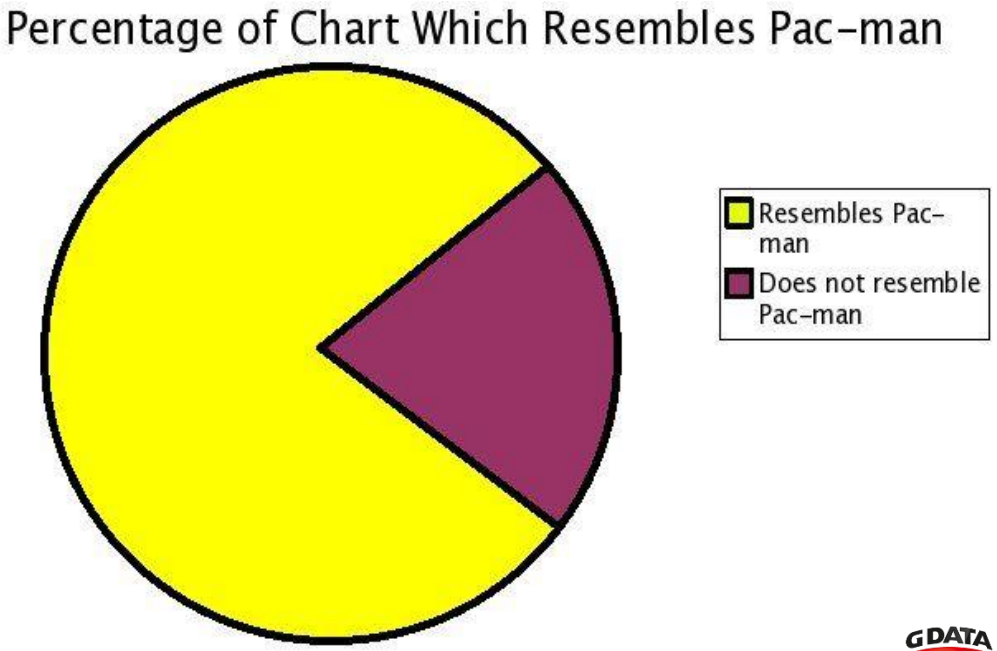
G DATA | ADVANCED ANALYTICS

Wicked plan...

Look at all areas of a binary
- API calls
- Strings
- Structure

Graphs

Radare2



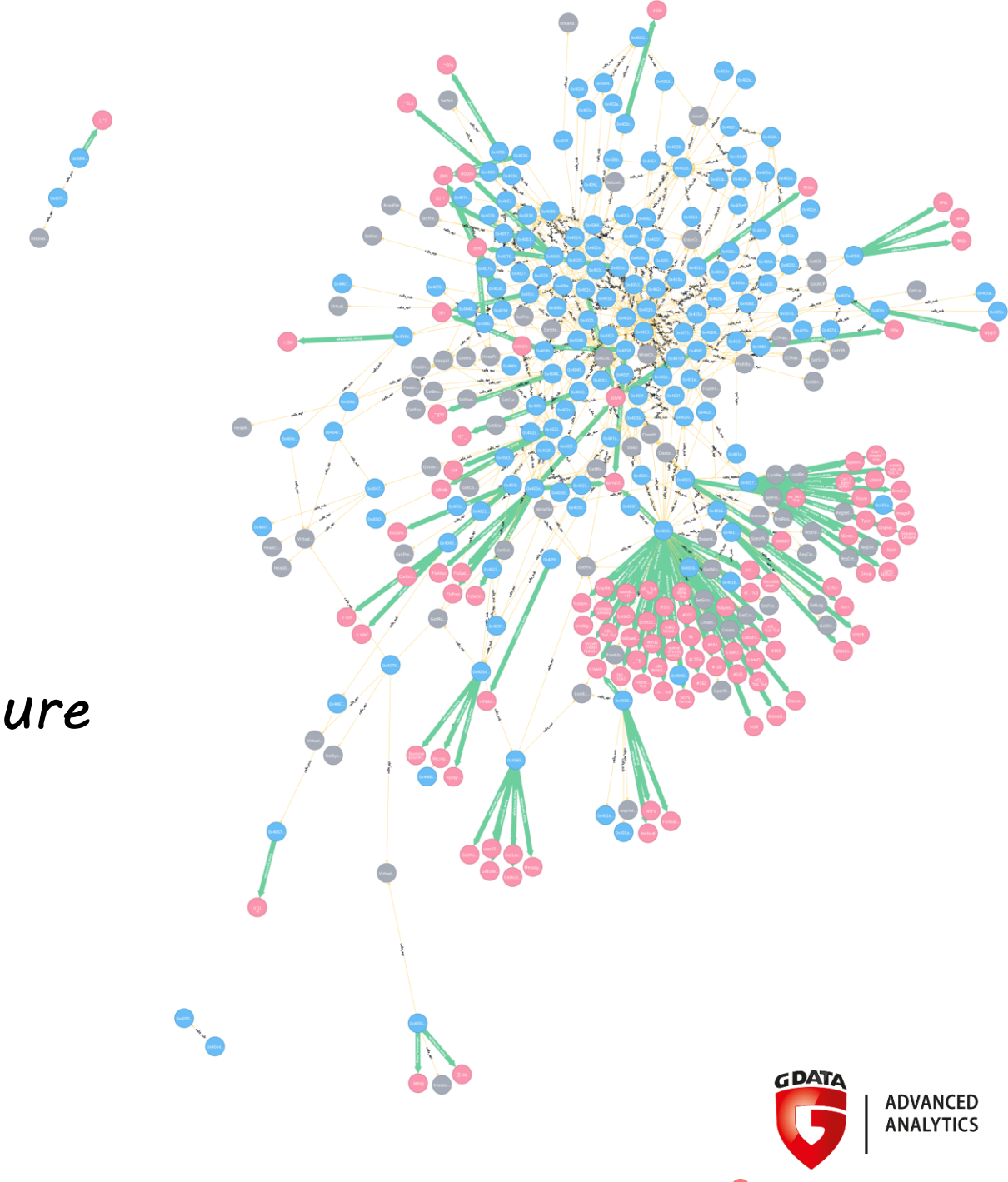
Graphs

Binaries naturally are graphs,
and graphs of graphs,
and graphs of.. You get it

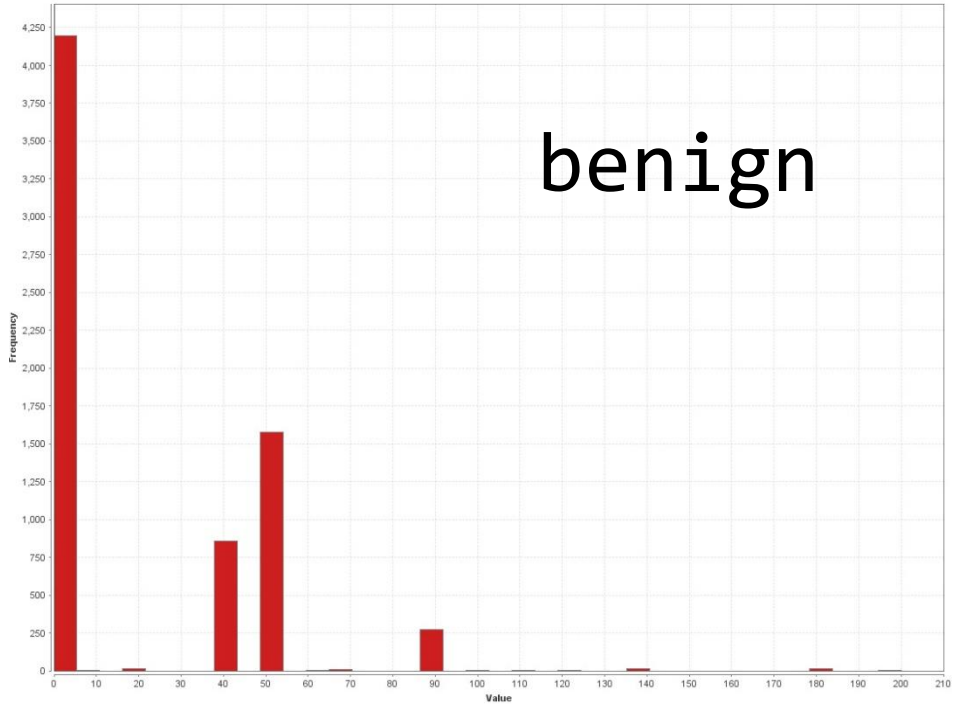
Great for visualisation

Also pretty amazing data structure

Strings, APIs, what not

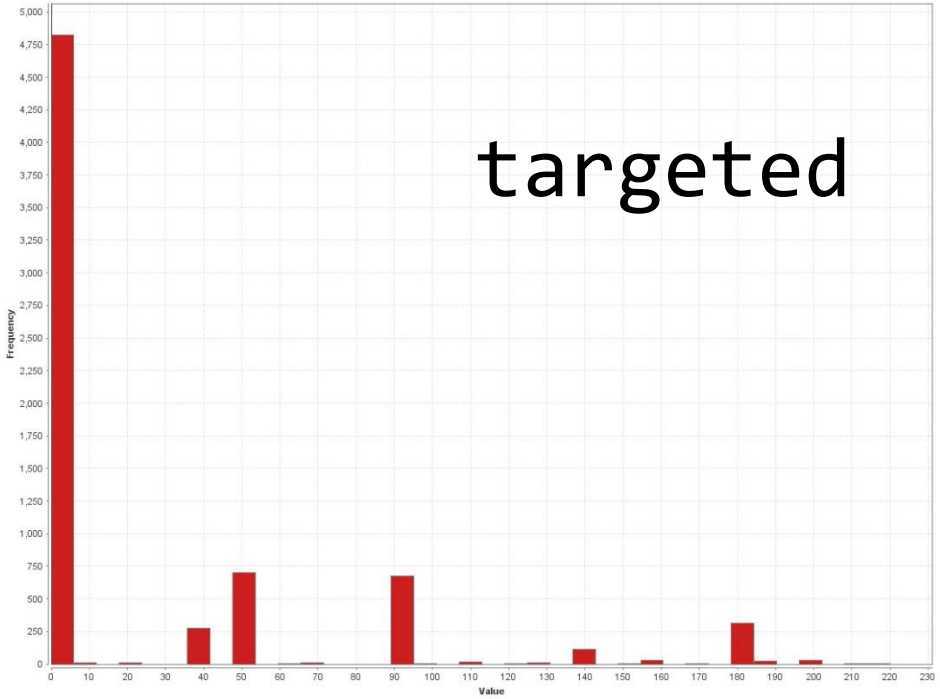


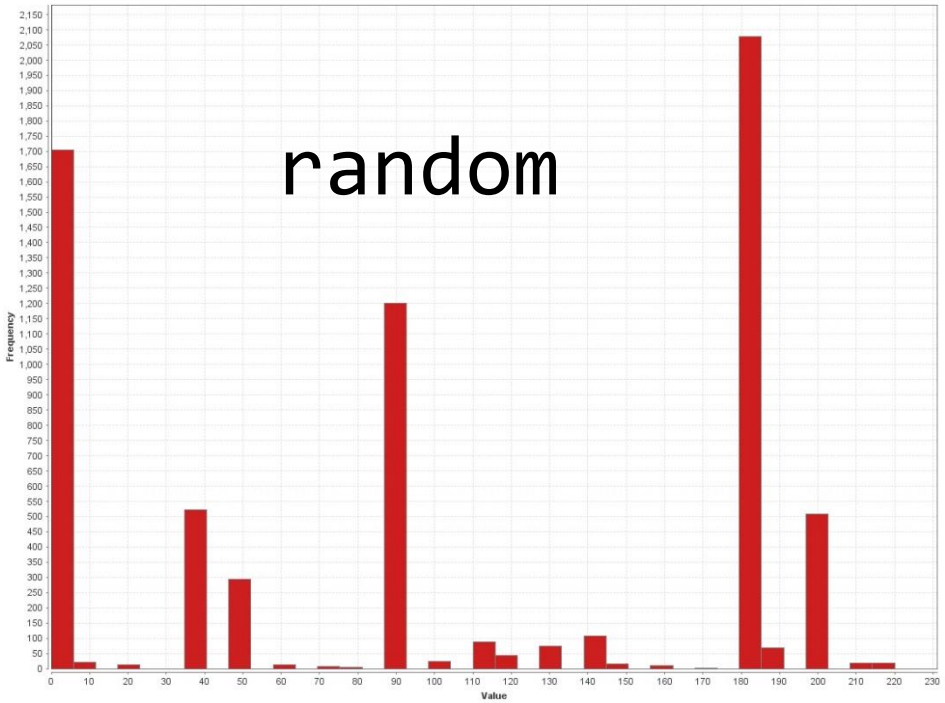
G DATA | ADVANCED ANALYTICS



Indicators for packers

EP section name abnormal
EP section entropy too high/low
Use of TLS sections
API calls / KB ratio
Section count too low
Imphash missing





G DATA | ADVANCED ANALYTICS


WHY?
G DATA
ADVANCED
ANALYTICS

Help in static analysis

Persisting of analysis results

Small to medium scale sample sets

Tool thats easy to handle and extendable

Metrics

Creative indicator extraction

No big data

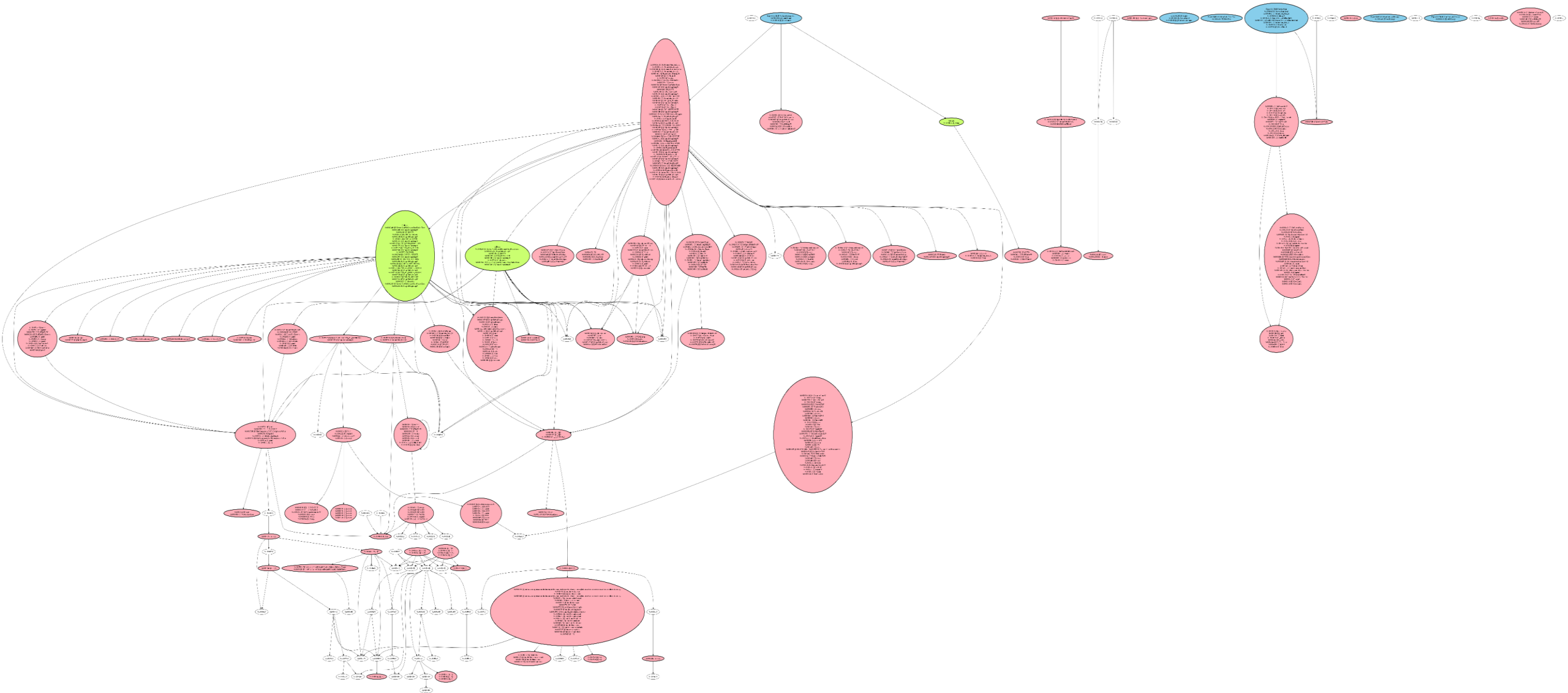
No clustering

For sure no machine learning

No binary diffing

No serious math

No software licenses ^^



G DATA | ADVANCED ANALYTICS

So yeah.. I used radare2

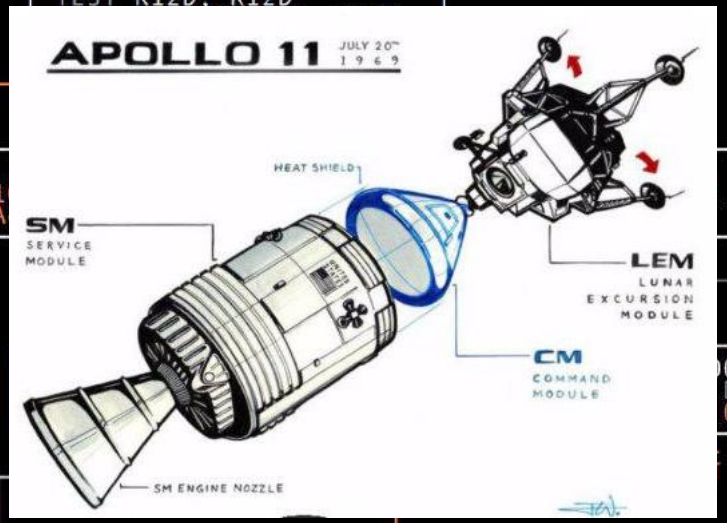
Radare2 accessed through r2pipe, scripted from Python

Available for free
Disassemble (and assemble for) many different architectures
Debug with local native and remote debuggers (gdb, rap, webui, r2pipe, winedbg, windbg)
Run on Linux, *BSD, Windows, OSX, Android, iOS, Solaris and Haiku
Perform forensics on filesystems and data carving
Be scripted in Python, Javascript, Go and more
Support collaborative analysis using the embedded webserver
Visualize data structures of several file types
Patch programs to uncover new features or fix vulnerabilities
Use powerful analysis capabilities to speed up reversing
Aid in software exploitation

```
kevin@kevin-VirtualBox:~/radare2$ r2 /mnt/hgfs/projects/testmalware/banito.bin
 -- Microloft Visual Radare.NET 2008. Now OOXML Powered!
```

```
0x100001a0e ;[As]
TEST R12D, R12D
```

```
0x100001a2e ;[Ad]
LEA  RAX, sym.func.1
JMP  0x100001A47 ;[A
```



```
0x100001a40 ;[Al]
RAX, sym.func.10000368c
```

```
0x100001a25 ;[An]
LEA  RAX, sym.func.10000375e
JMP  0x100001A47 ;[Ac]
```

```
0x100001a6b ;[Af]
CALL sym.func.100001b58 ;[Ah]
MOVZX EDI, BYTE [0x100005518]
CALL sym.imp.exit ;[Ai]
```

With splendid reasoning

Scalable

Scriptable

GUI-free

Great support

Quick bug fixes

Can analyze entire binaries

Provides

- functions and cross references
- symbols
- strings
- basic PE information

```
kevin@kevin-VirtualBox:~/radare2$ r2 /mnt/hgfs/projects/testmalware/banito.bin
 -- I script in C, because I can.
```

Color me rainbow ^^

```
; arg int arg_1h @ ebp+0x1
; arg int arg_3h @ ebp+0x3
; arg int arg_8h @ ebp+0x8
; arg int arg_ch @ ebp+0xc
; arg int arg_10h @ ebp+0x10
; arg int arg_14h @ ebp+0x14
; var int local_0h @ ebp-0x0
; var int local_2h @ ebp-0x2
; var int local_4h @ ebp-0x4
; var int local_8h @ ebp-0x8
; var int local_ch @ ebp-0xc
; var int local_10h @ ebp-0x10
; var int local_14h @ ebp-0x14
; var int local_18h @ ebp-0x18
; var int local_1ch @ ebp-0x1c
; var int local_20h @ ebp-0x20
; CALL XREF from 0x773d4664 (sym.COMCTL32.dll_Ordinal_156)
; CALL XREF from 0x773d460a (sym.COMCTL32.dll_CreateMRUListW)
0x773d42f6      8bff            mov edi, edi
0x773d42f8      55              push ebp
0x773d42f9      8bec            mov ebp, esp
0x773d42fb      83ec20          sub esp, 0x20
0x773d42fe      53              push ebx
0x773d42ff      8b5d08          mov ebx, dword [ebp + arg_8h] ; [0x8:4]=4
0x773d4302      8b4314          mov eax, dword [ebx + 0x14] ; [0x14:4]=0
0x773d4305      8b4b0c          mov ecx, dword [ebx + 0xc]  ; [0xc:4]=0xffff
0x773d4308      8b5310          mov edx, dword [ebx + 0x10] ; [0x10:4]=184
0x773d430b      56              push esi
0x773d430c      57              push edi
0x773d430d      8b7b04          mov edi, dword [ebx + 4]    ; [0x4:4]=3
0x773d4310      33f6            xor esi, esi
0x773d4312      85c0            test eax, eax
0x773d4314      8975f8          mov dword [ebp - local_8h], esi
0x773d4317      897de8          mov dword [ebp - local_18h], edi
0x773d431a      8945ec          mov dword [ebp - local_14h], eax
,=< 0x773d431d      7521            jne 0x773d4340
|   0x773d431f      8b4308          mov eax, dword [ebx + 8]    ; [0x8:4]=4
|   0x773d4322      a801            test al, 1                  ; "Z.  @ 0x1
```

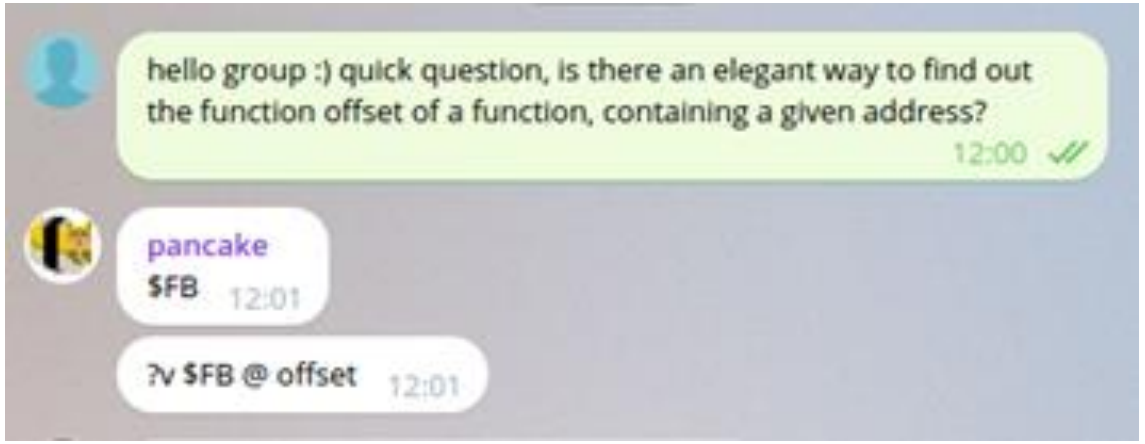
R2handle = r2pipe.open(<file>)
R2handle.cmd(<cmd>)
Watch magic

```
kevin@kevin-VirtualBox:~/radare2$ r2 /mnt/hgfs/projects/testmalware/banito.bin
 -- See you in shell
```

aaa – analyze the target binary
afr @ [address] – recursively analyze function at [address]
iS – get information about file sections
iij – get import table in JSON format
axtj @@ sym.* - get cross references on found symbols in JSON
axtj @ [address] – get cross references for [address]
pd 300 @ [address] – disassemble 300 instructions at [address]
pd -30 @ [address] – disassemble backwards 30 instructions at [address]
pdf @ [address] – disassemble function at [address], after e.g. aaa command
izzj – get strings out of entire binary in JSON
iz – get strings out of code section
iEj – get exports of a library
?v $FB @ [address] – get function which contains [address]
aflj – get list of functions with supporting information in JSON

r2 command cheat sheet

Many thanks to pancake, maijin & friends <3



G DATA ADVANCED ANALYTICS

Graphity

Python project built on
- radare2 / r2pipe
- NetworkX
- pyplot
- pefile
- Neo4j

Be published soonish at
https://github.com/pinkflawd

graphity
graphityOut
graphityFunc
graphityUtil

GDATA ADVANCED ANALYTICS

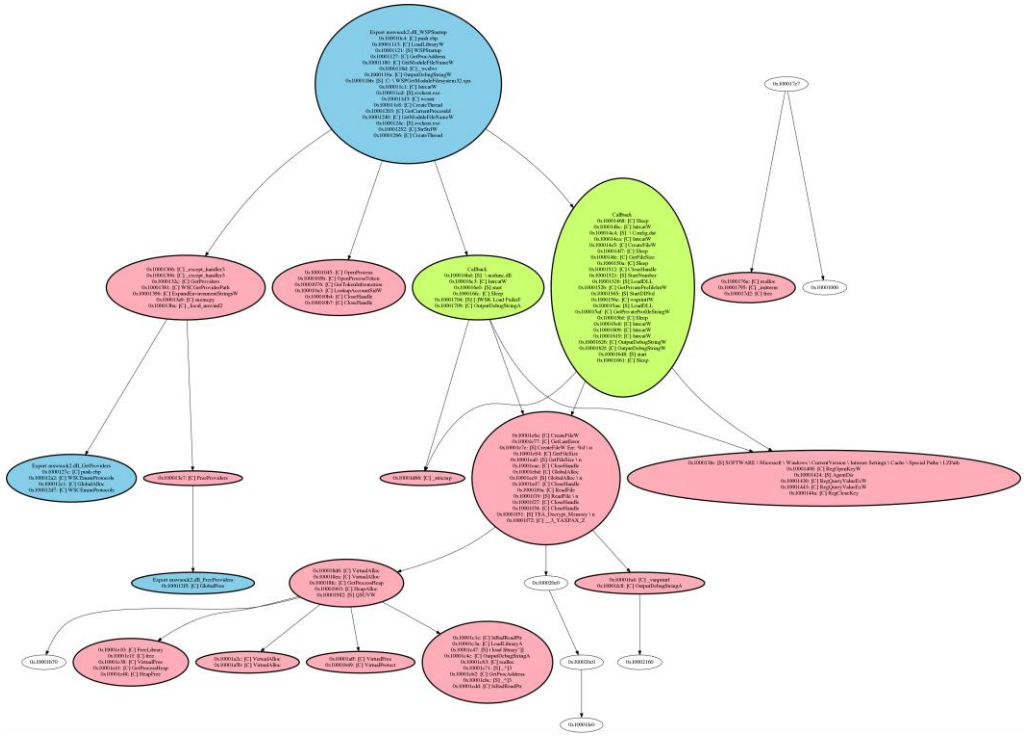
Function call graphs

- Function cross references within code section
  - References to function offsets
  - References to code w/o function
  - Outside executable section(s)

Nodes: functions

=> Offset, size, calling convention

Edges: calls, handler functions



A binary art project :)

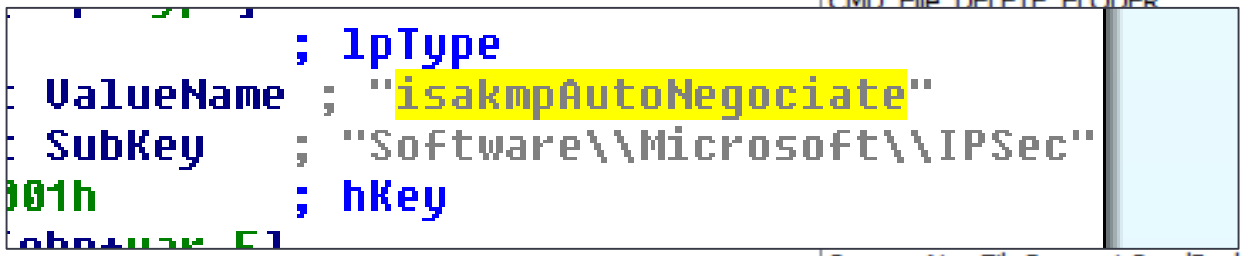
Strings

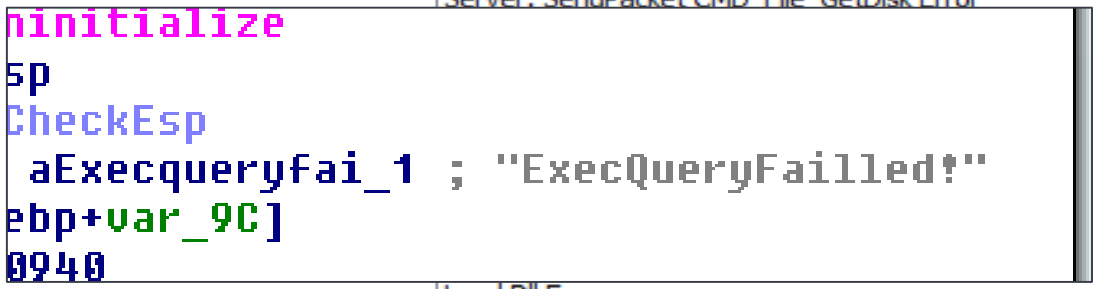
String parsing
Evaluation: ASCII, cross references

String list detection
string length + alingment
string following w/o cross reference

Fitting strings into the graph

Whats the information one can gain from strings?





ADVANCED ANALYTICS

APIs

Cross references on symbols

Indirect calls
- parsing for mov/lea
- disassembling further
- call and jmp considered xref

Thunk pruning

Dynamic loading

```
[0x004344b6]> axt @@ sym.*
data 0x40e552 mov ebp, dword [sym.imp.KERNEL32.dll_LoadLibraryA] in fcn.00402db0
data 0x40e558 mov ebx, dword [sym.imp.KERNEL32.dll_GetProcAddress] in fcn.00402db0
call 0x4345de call dword [sym.imp.KERNEL32.dll_GetModuleHandleA] in entry0
data 0x4345de call dword [sym.imp.KERNEL32.dll_GetModuleHandleA] in entry0
call 0x4345ba call dword [sym.imp.KERNEL32.dll_GetStartupInfoA] in entry0
data 0x4345ba call dword [sym.imp.KERNEL32.dll_GetStartupInfoA] in entry0
call 0x401c3f call dword [sym.imp.GDI32.dll_RealizePalette] in fcn.00401040
data 0x401c3f call dword [sym.imp.GDI32.dll_RealizePalette] in fcn.00401040
call 0x401b5b call dword [sym.imp.GDI32.dll_CreateDIBSection] in fcn.00401040
call 0x401bd6 call dword [sym.imp.GDI32.dll_CreateDIBSection] in fcn.00401040
data 0x401b5b call dword [sym.imp.GDI32.dll_CreateDIBSection] in fcn.00401040
data 0x401bd6 call dword [sym.imp.GDI32.dll_CreateDIBSection] in fcn.00401040
call 0x401b6b call dword [sym.imp.GDI32.dll_IntersectClipRect] in fcn.00401040
data 0x401b6b call dword [sym.imp.GDI32.dll_IntersectClipRect] in fcn.00401040
call 0x401c5d call dword [sym.imp.GDI32.dll_CreateRectRgn] in fcn.00401040
data 0x401c5d call dword [sym.imp.GDI32.dll_CreateRectRgn] in fcn.00401040
call 0x401c4f call dword [sym.imp.GDI32.dll_GetBkMode] in fcn.00401040
data 0x401c4f call dword [sym.imp.GDI32.dll_GetBkMode] in fcn.00401040
call 0x401c47 call dword [sym.imp.GDI32.dll_CreateCompatibleDC] in fcn.00401040
data 0x401c47 call dword [sym.imp.GDI32.dll_CreateCompatibleDC] in fcn.00401040
data 0x401c2d mov esi, dword [sym.imp.GDI32.dll_SetPaletteEntries] in fcn.00401040
call 0x401c27 call dword [sym.imp.GDI32.dll_GetClipBox] in fcn.00401040
```

Callbacks / Handler Functions

„Top-down“

Disassemble upwards
Check the push instructions for function cross references
Add edge and tag
Currently only CreateThread and SetWindowsHookEx, because context

„Bottom-up“

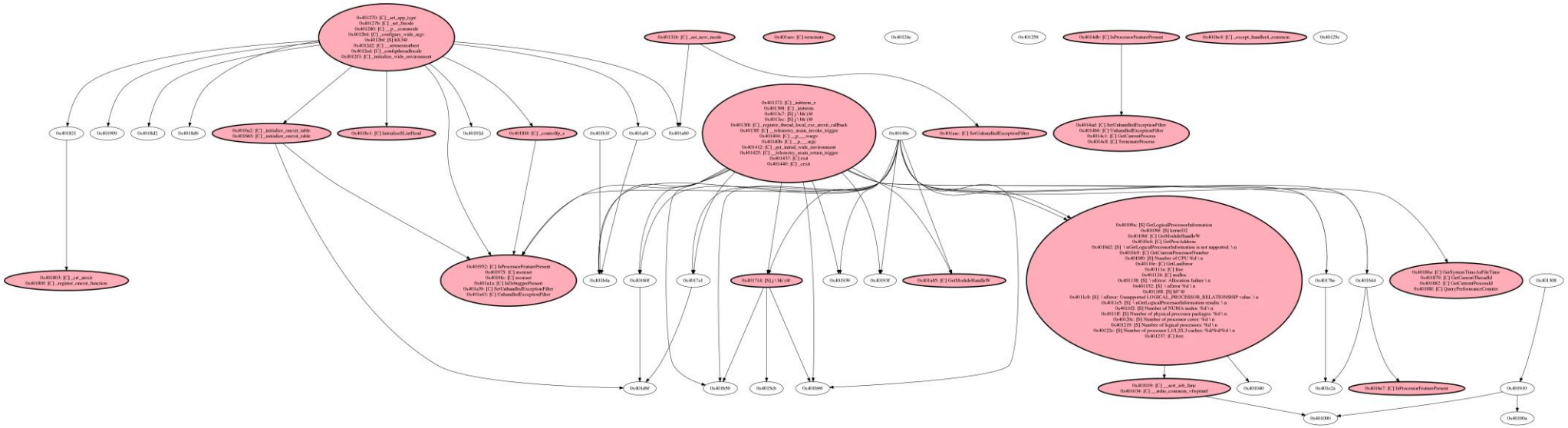
Sweep for nodes without inbound edges
Check for cross references within functions
Add edge and tag

ADVANCED ANALYTICS

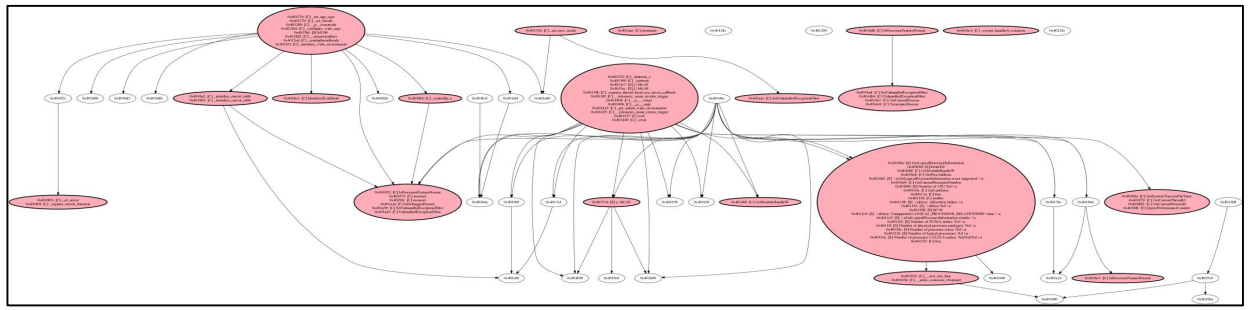
Compiler settings & optimizer magic

Graphing objectives:

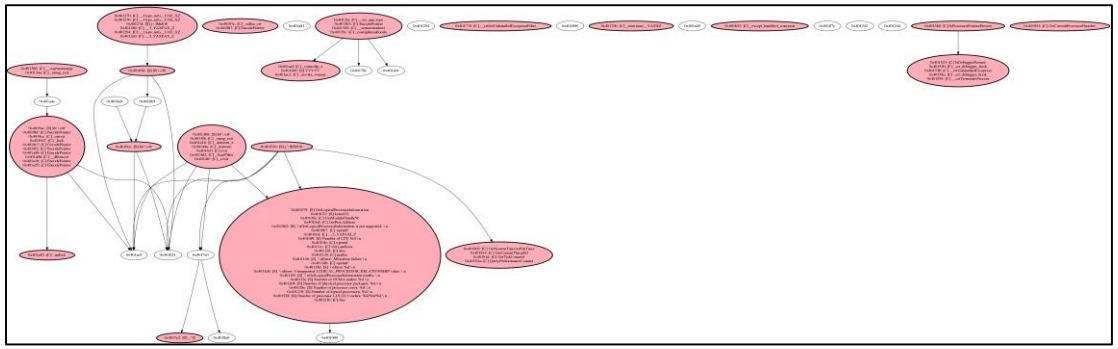
- *as little data as possible*
- *with as much information as possible*



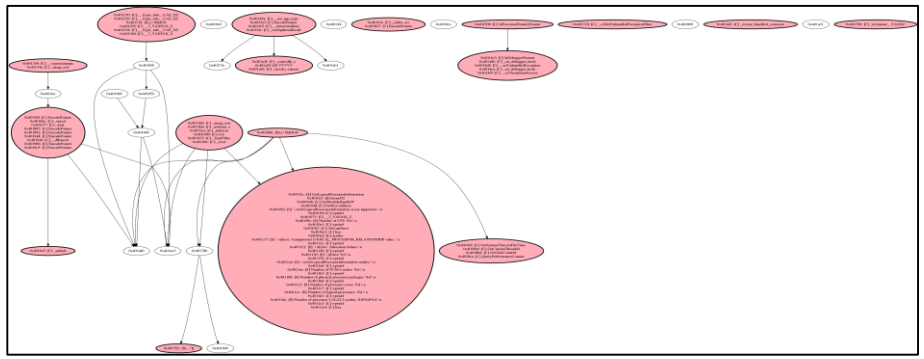
```
kevin@kevin-VirtualBox:~/radare2$ r2 /mnt/hgfs/projects/testmalware/banito.bin
 -- Using radare2 to generate intelligence ...
```



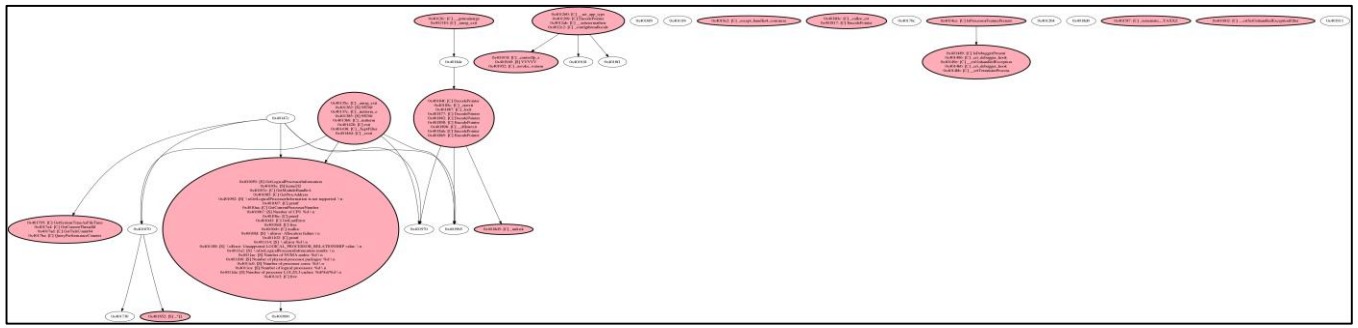
Default



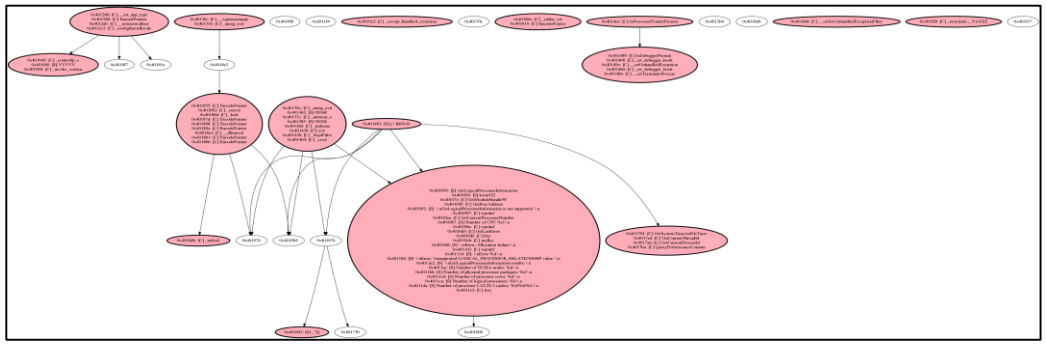
FullyOptimized



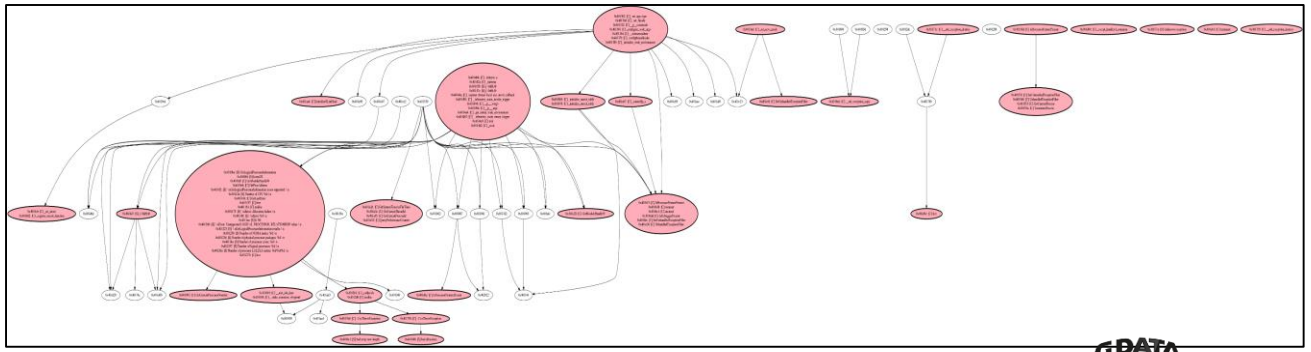
SizeOptimized



vc110



vc120



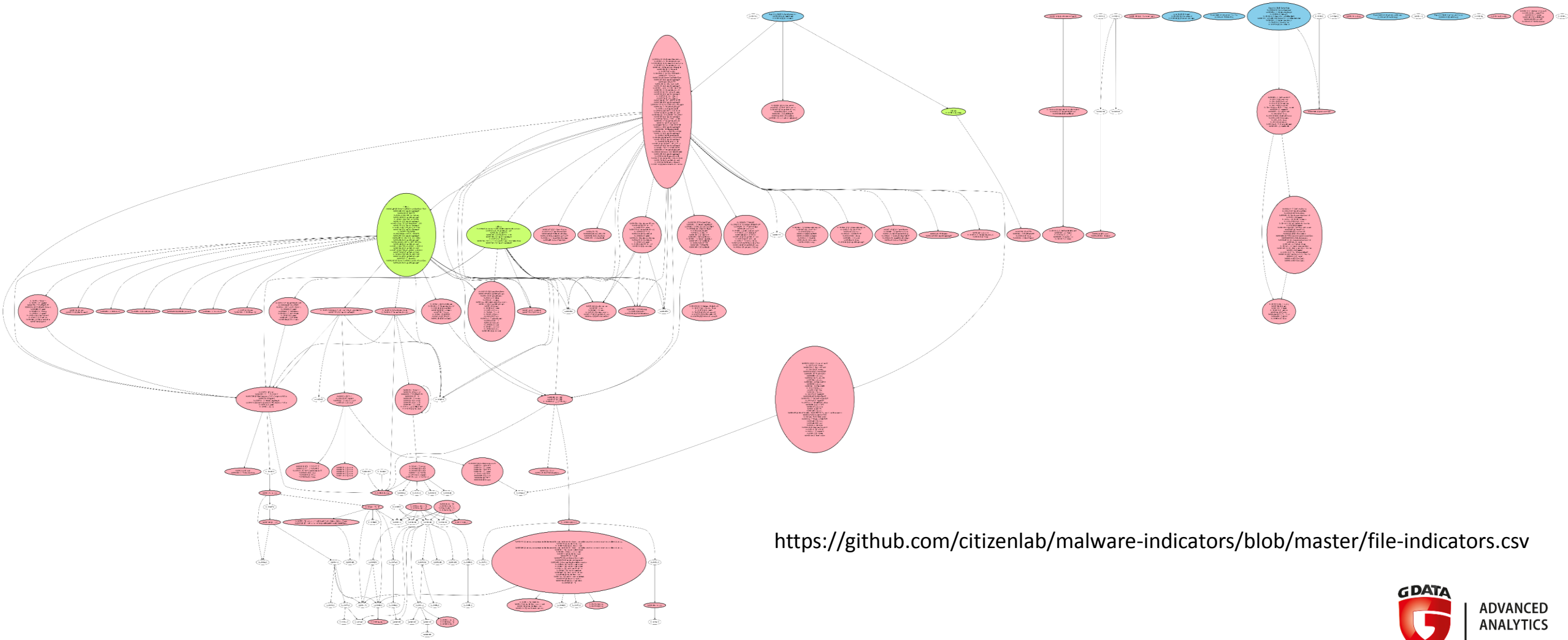
vc150Dbg

GDATA | ADVANCED ANALYTICS

STUFF

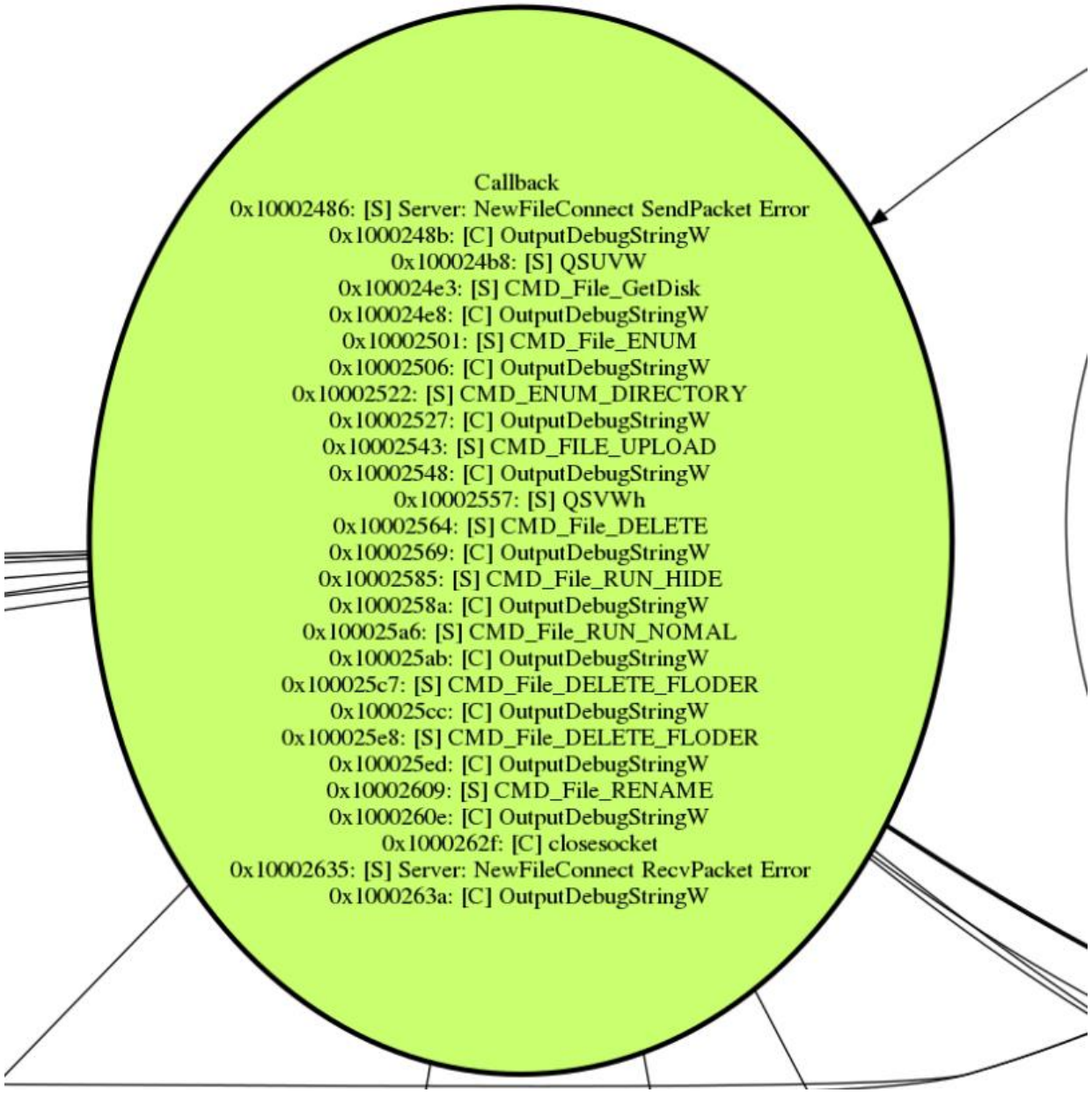
Visualization
Behavior
Metrics
GraphDB

Backdoor: Win32/Redsip.A



https://github.com/citizenlab/malware-indicators/blob/master/file-indicators.csv

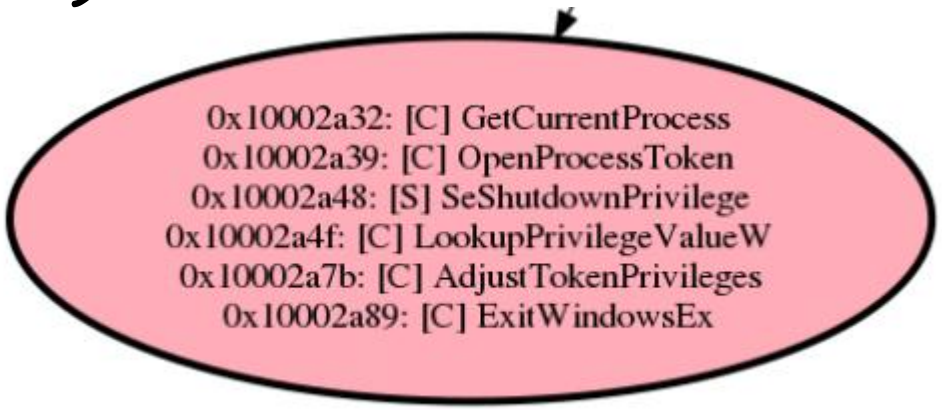
G DATA | ADVANCED ANALYTICS



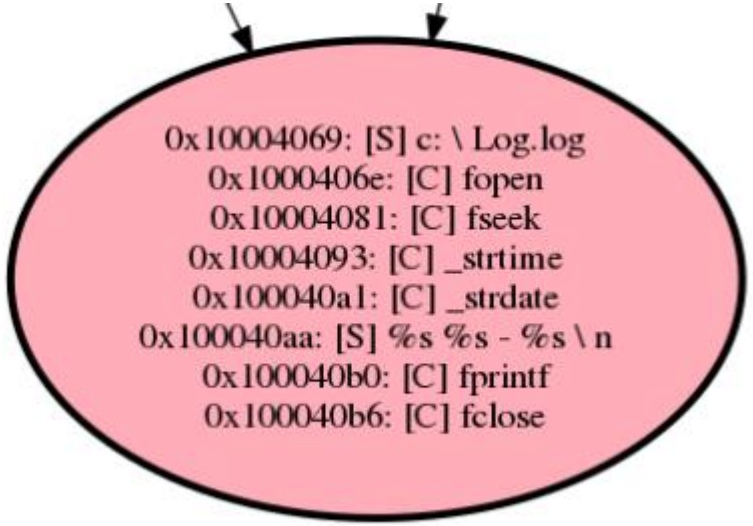
Thread handler function
C&C command parsing

System information gathering

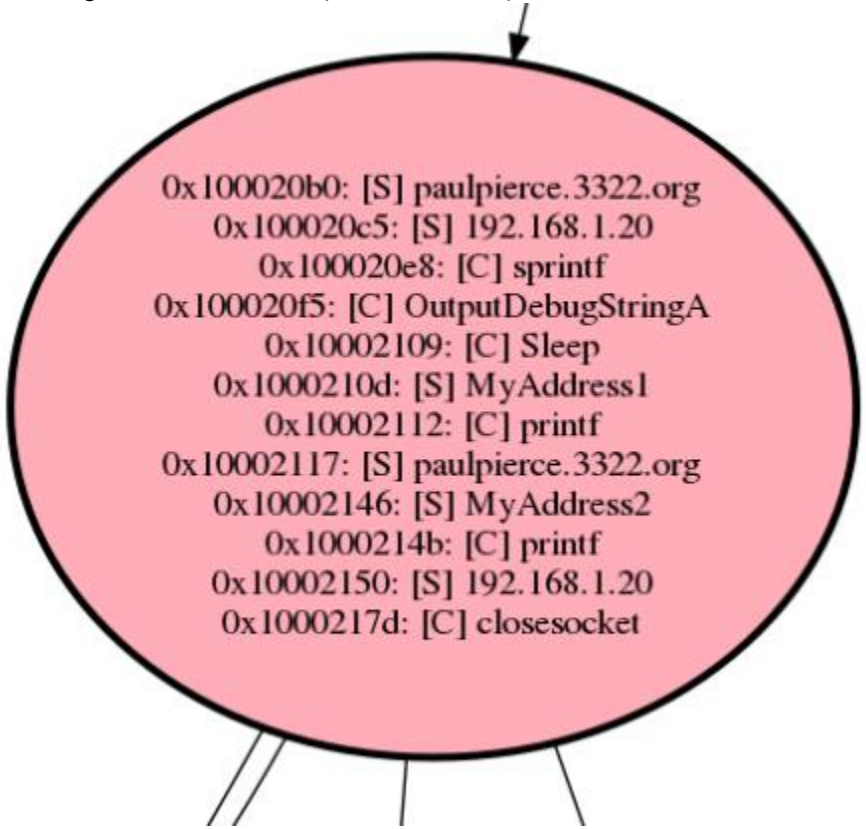
System shutdown feature

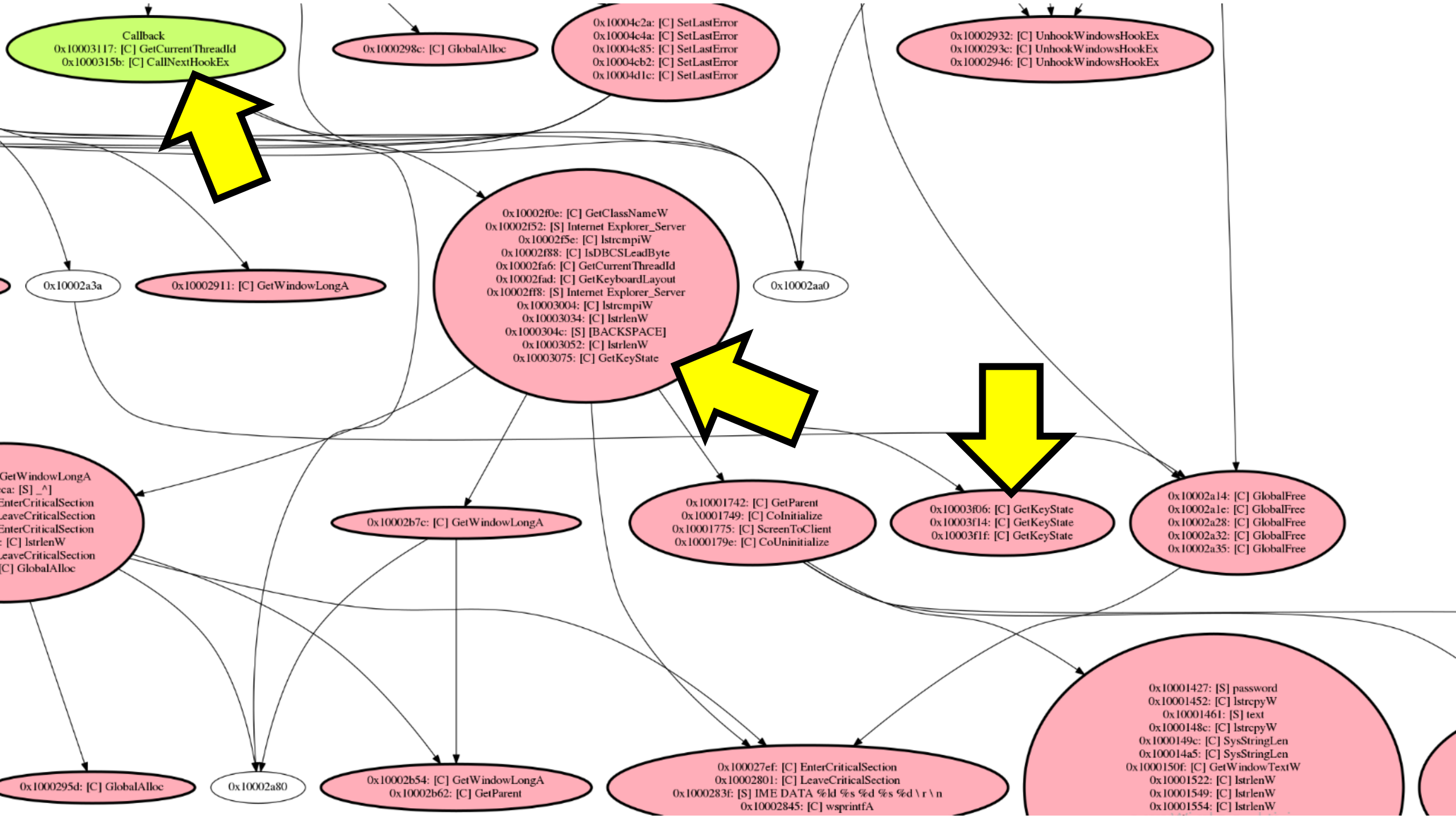


Writing to logfile

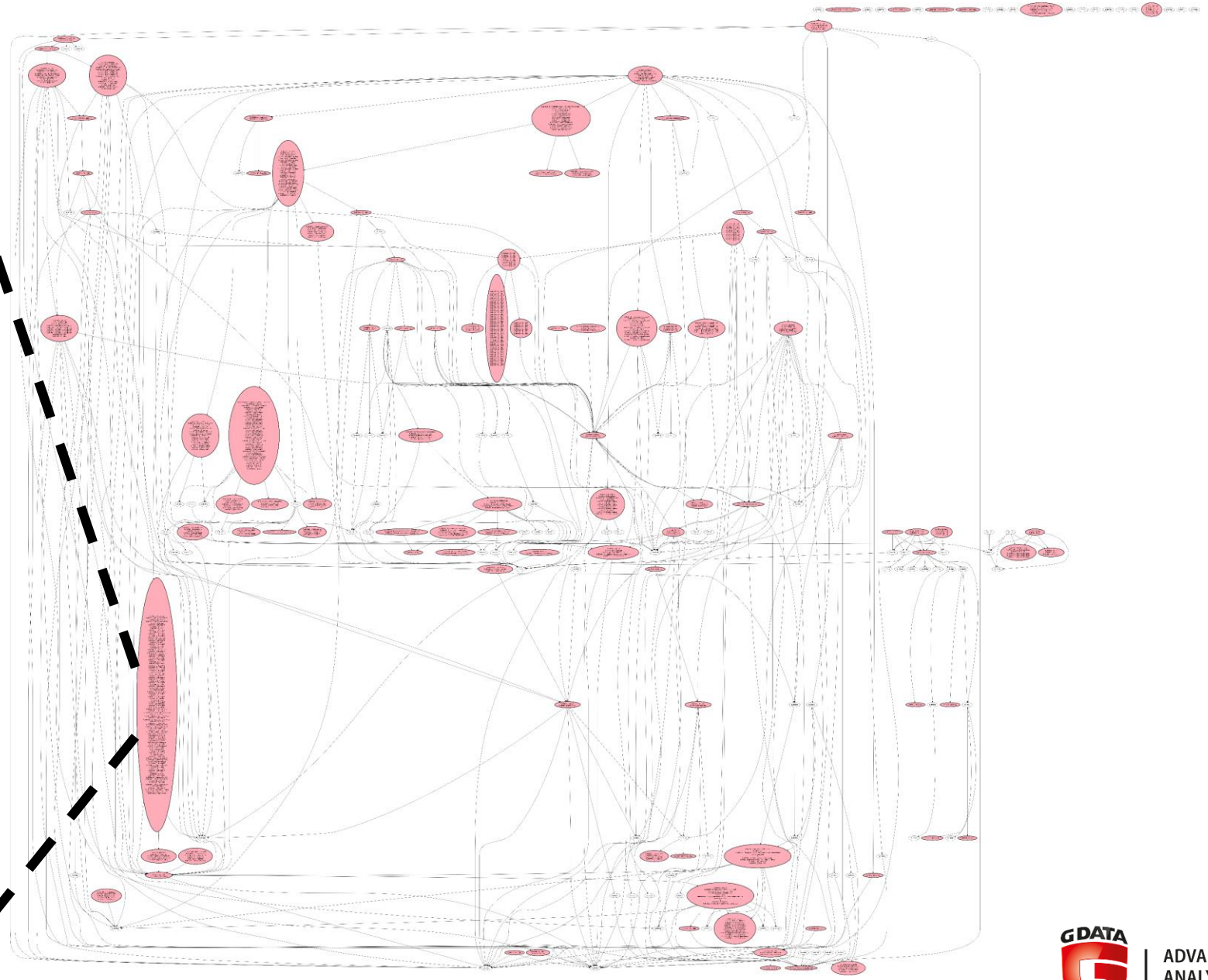


Who's Paul Pierce?!

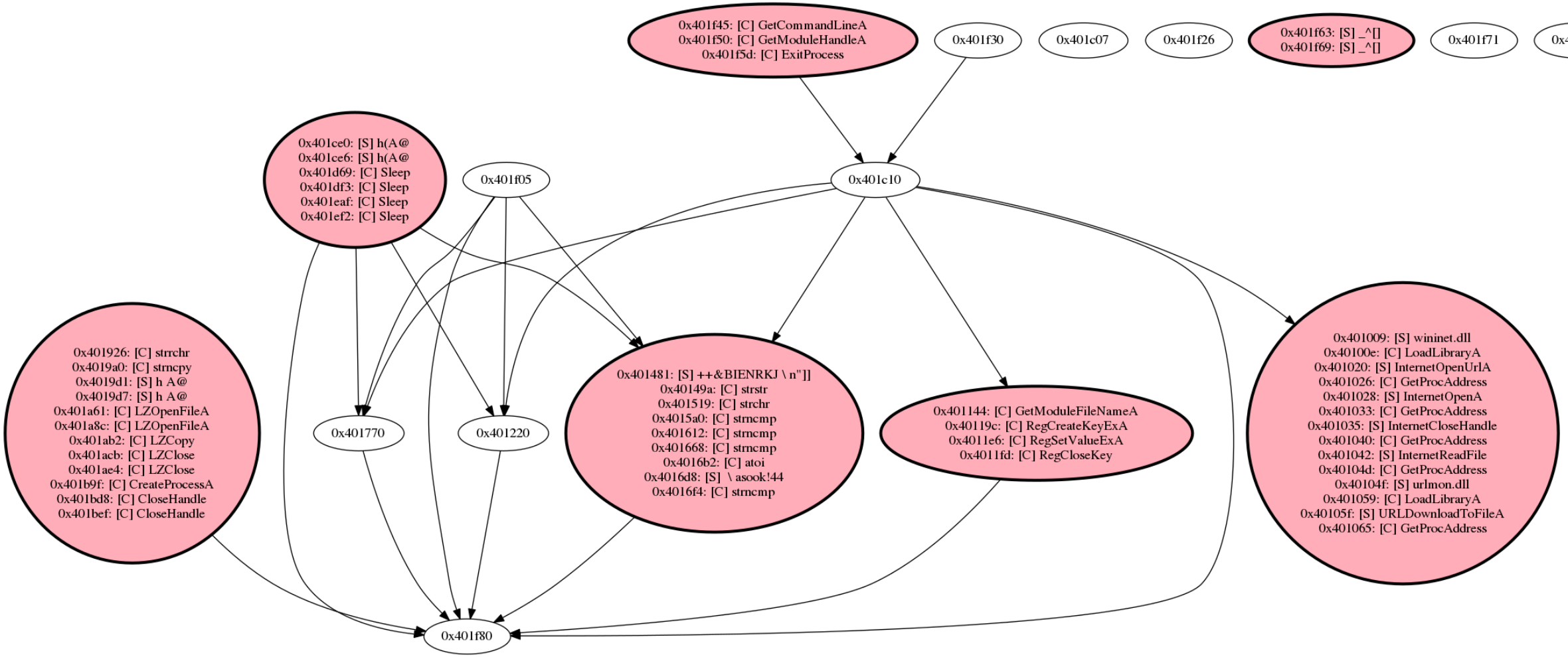



Callback
0x10003117: [C] GetCurrentThreadId
0x1000315b: [C] CallNextHookEx
0x1000298c: [C] GlobalAlloc
0x10004c2a: [C] SetLastError
0x10004c4a: [C] SetLastError
0x10004c85: [C] SetLastError
0x10004cb2: [C] SetLastError
0x10004d1c: [C] SetLastError
0x10002932: [C] UnhookWindowsHookEx
0x1000293c: [C] UnhookWindowsHookEx
0x10002946: [C] UnhookWindowsHookEx
0x10002f0e: [C] GetClassNameW
0x10002f52: [S] Internet Explorer_Server
0x10002f5e: [C] lstrcmpiW
0x10002f88: [C] IsDBCSLeadByte
0x10002fa6: [C] GetCurrentThreadId
0x10002fad: [C] GetKeyboardLayout
0x10002ff8: [S] Internet Explorer_Server
0x10003004: [C] lstrcmpiW
0x10003034: [C] lstrlenW
0x1000304c: [S] [BACKSPACE]
0x10003052: [C] lstrlenW
0x10003075: [C] GetKeyState
0x10002a3a
0x10002911: [C] GetWindowLongA
0x10002aa0
GetWindowLongA
ca: [S] _^]
EnterCriticalSection
LeaveCriticalSection
EnterCriticalSection
: [C] lstrlenW
LeaveCriticalSection
[C] GlobalAlloc
0x10002b7c: [C] GetWindowLongA
0x10001742: [C] GetParent
0x10001749: [C] CoInitialize
0x10001775: [C] ScreenToClient
0x1000179e: [C] CoUninitialize
0x10003f06: [C] GetKeyState
0x10003f14: [C] GetKeyState
0x10003f1f: [C] GetKeyState
0x10002a14: [C] GlobalFree
0x10002a1c: [C] GlobalFree
0x10002a28: [C] GlobalFree
0x10002a32: [C] GlobalFree
0x10002a35: [C] GlobalFree
0x1000295d: [C] GlobalAlloc
0x10002a80
0x10002b54: [C] GetWindowLongA
0x10002b62: [C] GetParent
0x100027ef: [C] EnterCriticalSection
0x10002801: [C] LeaveCriticalSection
0x1000283f: [S] IME DATA %ld %s %d %s %d \ r \ n
0x10002845: [C] wsprintfA
0x10001427: [S] password
0x10001452: [C] lstrcpyW
0x10001461: [S] text
0x1000148c: [C] lstrcpyW
0x1000149c: [C] SysStringLen
0x100014a5: [C] SysStringLen
0x1000150f: [C] GetWindowTextW
0x10001522: [C] lstrlenW
0x10001549: [C] lstrlenW
0x10001554: [C] lstrlenW

0x401941: [C] lstrcmpiW
0x401964: [S] ESET
0x40196e: [C] lstrcmpiW
0x40197a: [S] \ eset
0x4019a5: [S] nod32
0x4019af: [S] KasperskyLab
0x4019b9: [C] lstrcmpiW
0x4019c5: [S] \ KasperskyLab
0x4019fa: [S] JiangMin
0x401a04: [C] lstrcmpiW
0x401a27: [S] AhnLab
0x401a31: [C] lstrcmpiW
0x401a3d: [S] \ ahnlab
0x401a72: [S] Filseclab
0x401a7c: [C] lstrcmpiW
0x401a88: [S] \ Filseclab
0x401abd: [S] micropoint
0x401ac7: [C] lstrcmpiW
0x401aea: [S] MPAV
0x401af4: [C] lstrcmpiW
0x401b17: [S] 360safe
0x401b21: [C] lstrcmpiW
0x401b2d: [S] \ 360safe
0x401b62: [S] Norton
0x401b6c: [C] lstrcmpiW
0x401b8f: [S] G Data
0x401b99: [C] lstrcmpiW
0x401ba5: [S] \ G Data
0x401bd0: [S] G data
0x401bda: [S] Panda Security



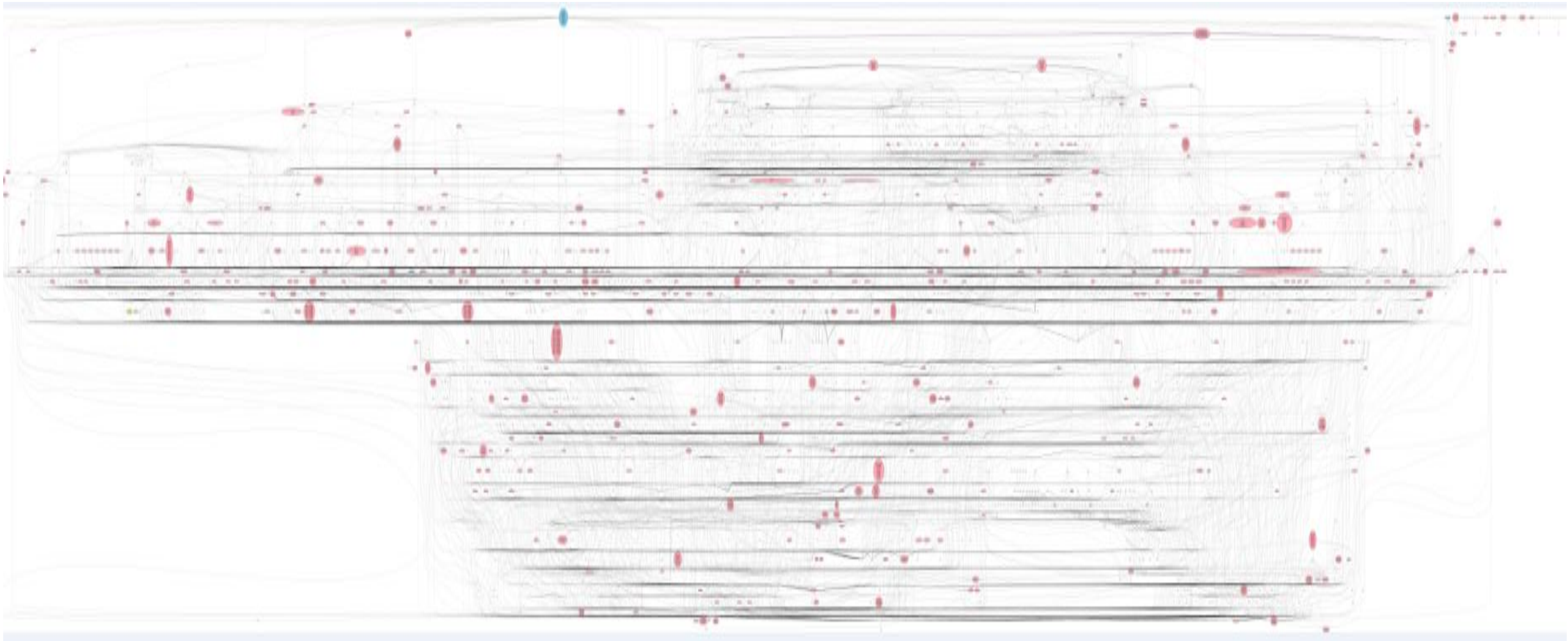
G DATA | ADVANCED ANALYTICS



Suspiciously low on strings
Rich in APIs for download-and-execute-binary ops

G DATA | ADVANCED ANALYTICS

So..



['0x100018fc', u'GetProcessHeap']
['0x10001903', u'HeapAlloc']
['0x10001982', 'QSUVW']
0x10001dc0
['0x10001e10', u'FreeLibrary']
['0x10001e1f', u'free']
['0x10001e38', u'VirtualFree']
['0x10001e41', u'GetProcessHeap']
['0x10001e48', u'HeapFree']
0x100019e0
['0x10001a2c', u'VirtualAlloc']
['0x10001a5b', u'VirtualAlloc']
0x100010c4
mswsock2.dll_WSPStartup
['0x100010c4', u'push ebp']
['0x10001113', u'LoadLibraryW']
['0x10001121', 'WSPStartup']
['0x10001127', u'GetProcAddress']
['0x10001180', u'GetModuleFileNameW']
['0x1000118d', u'_wcslwr']
['0x1000119a', u'OutputDebugStringW']
['0x100011bb', ' C: \\\\ WSPGetModuleFilesystem32.xps']
['0x100011c1', u'lstrcatW']
['0x100011cd', 'svchost.exe']
['0x100011d3', u'wcsstr']
['0x100011e8', u'CreateThread']
['0x10001203', u'GetCurrentProcessId']
['0x10001240', u'GetModuleFileNameW']
['0x1000124c', 'svchost.exe']
['0x10001252', u'StrStrIW']
['0x10001266', u'CreateThread']
0x1000100b
['0x10001045', u'OpenProcess']
['0x1000105b', u'OpenProcessToken']
['0x10001076', u'GetTokenInformation']
['0x100010a3', u'LookupAccountSidW']
['0x100010b4', u'CloseHandle']
['0x100010b7', u'CloseHandle']
0x10001d10
['0x10001d66', u'_stricmp']
0x100012fc
['0x10001306', u'_except_handler3']
['0x10001306', u'_except_handler3']
['0x1000132c', u'GetProviders']

Rogue behavior detection

API call gadgets

„pattern matching“ of APIs

Iterate nodes

Iterate neighbors

If feasible, further iterations

Problems:

- indirect function calls
- bigger call gadgets lower hit chances
- human analyst to draw final conclusions

```
funcDict = {
    'DRIVERCOMM': ['DeviceIoControl'],
    'CREATESTARTSERVICE': ['OpenSCManager', 'CreateService', 'OpenService', 'StartService'],
    'CREATETHREAD': ['CreateThread'],
    'PROCESSITER': ['CreateToolhelp32Snapshot', 'Process32First', 'Process32Next'],
    'APILOADING': ['LoadLibrary', 'GetProcAddress'],
    'WRITEFILE': ['CreateFile', 'WriteFile'],
    'READFILE': ['CreateFile', 'ReadFile'],
    'WINHOOK': ['SetWindowsHookEx'],
    'DRIVESITER': ['GetLogicalDriveStrings', 'GetDriveType'],
    'FILEITER': ['FindFirstFile', 'FindNextFile', 'FindClose'],
    'REGSETVAL': ['RegOpenKey', 'RegSetValue'],
    'REGQUERY': ['RegOpenKey', 'RegQueryValue'],
    'DUMPRSRC': ['FindResource', 'LoadResource', 'CreateFile', 'WriteFile'],
    'LOADRSRC': ['FindResource', 'LoadResource', 'LockResource'],
    'WSASEND': ['WSAStartup', 'gethostbyname', 'send'],
    'RECV': ['recv', 'send'],
    'RETROINJECTION': ['GetCurrentProcess', 'CreatePipe', 'DuplicateHandle'],
    'WINEXEC': ['WinExec'],
    'SHELLEXEC': ['ShellExecute'],
    'CREATEPROC': ['CreateProcess'],
    'WINDOW': ['CreateWindow', 'RegisterClass', 'DispatchMessage'],
    'EXITSYSTEM': ['ExitWindows'],
    'TEMPFILEWRITE': ['GetTempFileName', 'CreateFile', 'WriteFile'],
    'REMTHREAD': ['CreateThread', 'WriteProcessMemory', 'ReadProcessMemory', 'ResumeThread'],
    'FPRINT': ['fopen', 'fprintf', 'fclose'],
    'UPDATERESOURCE': ['BeginUpdateResource', 'UpdateResource', 'EndUpdateResource'],
    'SCREENSHOT': ['CreateCompatibleDC', 'GetDeviceCaps', 'CreateCompatibleBitmap', 'BitBlt'],
    'CRYPT': ['CryptAcquireContext', 'CryptGenKey', 'CryptEncrypt']
}
```

Backdoor: Win32/Redsip.A

```
71  For INET found {'recv': '0x10003a40', 'send': '0x10003a40'}
72  For CREATETHREAD found {'CreateThread': '0x10002010'}
73  For CREATETHREAD found {'CreateThread': '0x10003080'}
74  For CREATETHREAD found {'CreateThread': '0x10001bb0'}
75  For CREATETHREAD found {'CreateThread': '0x100034f0'}
76  For CREATETHREAD found {'CreateThread': '0x10002030'}
77  For CREATEPROC found {'CreateProcess': '0x10001cd0'}
78  For READFILE found {'CreateFile': '0x10003dc0', 'ReadFile': '0x10003dc0'}
79  For READFILE found {'CreateFile': '0x10002ce0', 'ReadFile': '0x10002ce0'}
80  For EXITSYSTEM found {'ExitWindows': '0x10002a20'}
81  For EXITSYSTEM found {'ExitWindows': '0x10002aa0'}
82  For REGQUERY found {'RegQueryValue': '0x10001790', 'RegOpenKey': '0x10001790'}
83  For SHELLEXEC found {'ShellExecute': '0x10002960'}
84  For SHELLEXEC found {'ShellExecute': '0x10002930'}
85  For APILOADING found {'GetProcAddress': '0x10003f40', 'LoadLibrary': '0x10003f40'}
86  For APILOADING found {'GetProcAddress': '0x10002e40', 'LoadLibrary': '0x10002e40'}
87  For APILOADING found {'GetProcAddress': '0x10001cd0', 'LoadLibrary': '0x10001cd0'}
88  For APILOADING found {'GetProcAddress': '0x10001be0', 'LoadLibrary': '0x10001be0'}
89  For FILEITER found {'FindNextFile': '0x100027b0', 'FindClose': '0x100027b0', 'FindFirstFile': '0x100027b0'}
90  For FILEITER found {'FindNextFile': '0x10001290', 'FindClose': '0x10001290', 'FindFirstFile': '0x10001290'}
91  For WRITEFILE found {'WriteFile': '0x10002f90', 'CreateFile': '0x10002f90'}
92  For WRITEFILE found {'WriteFile': '0x10002db0', 'CreateFile': '0x10002db0'}
93  * 2016-10-13 23:02:58.973503 Scan finished
```

G DATA | ADVANCED ANALYTICS

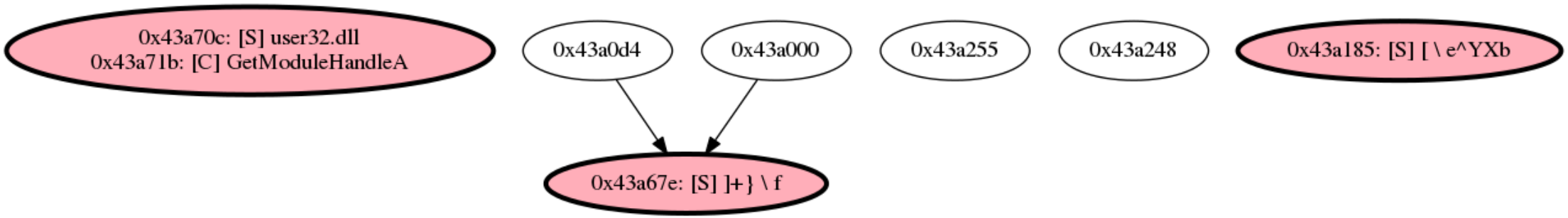
Random Dropper

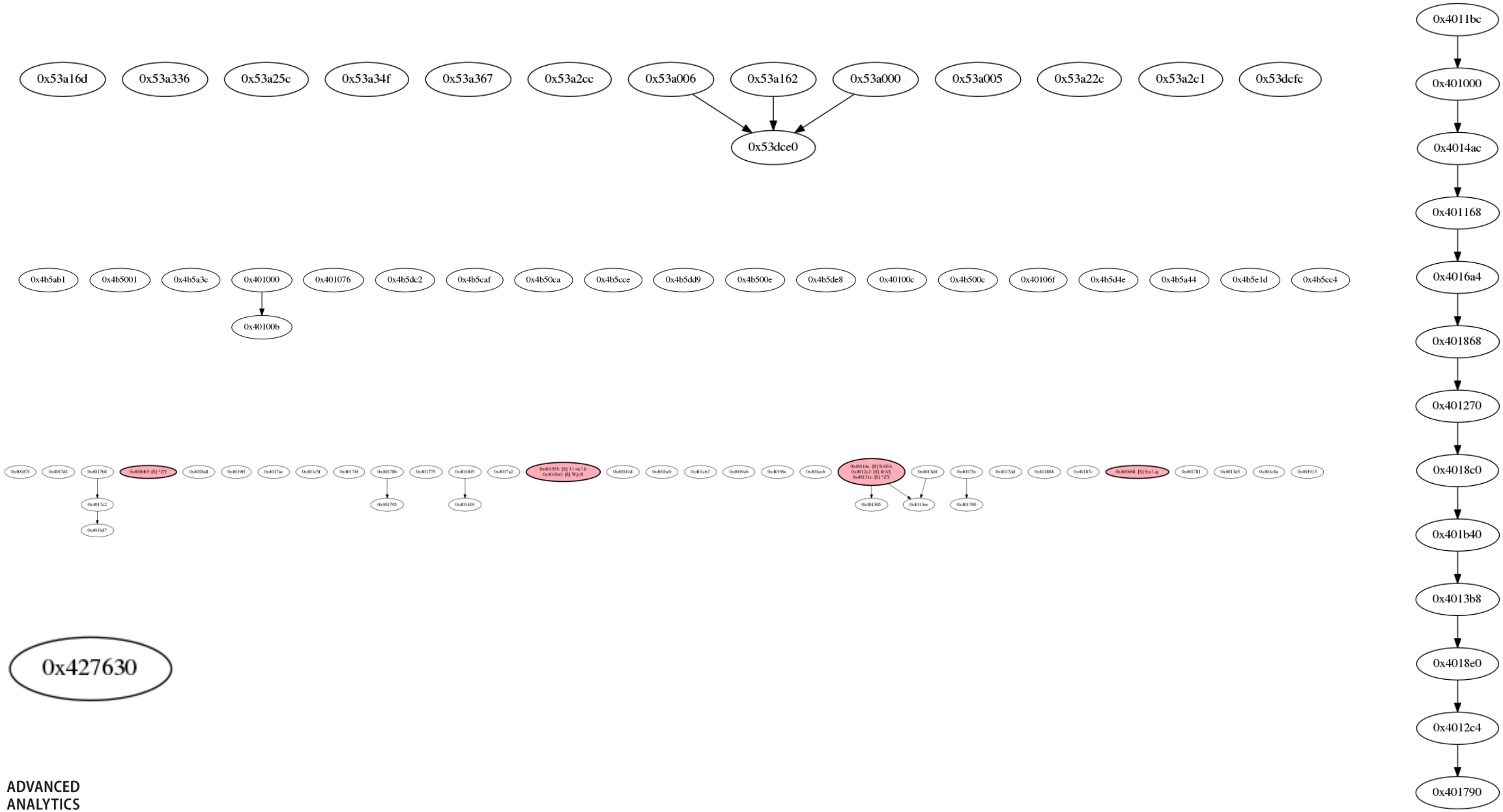
```
For REGSETVAL found {'RegOpenKey': '0x4011c0', 'RegSetValue': '0x4011e4'}
For CREATEPROC found {'CreateProcess': '0x401000'}
For CREATEPROC found {'CreateProcess': '0x4013c5'}
For READFILE found {'CreateFile': '0x401618', 'ReadFile': '0x401618'}
For APILOADING found {'GetProcAddress': '0x4053a5', 'LoadLibrary': '0x4053a5'}
For WRITEFILE found {'WriteFile': '0x401618', 'CreateFile': '0x401618'}
For WINEXEC found {'WinExec': '0x401618'}
```

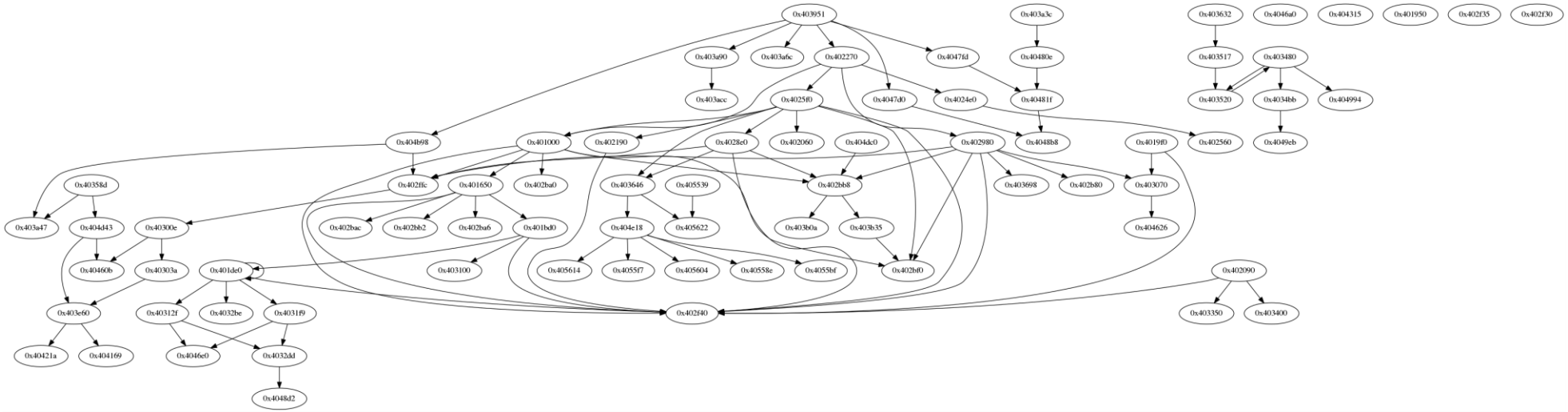
Win32/Banito

```
For APILOADING found {'GetProcAddress': '0x402db0', 'LoadLibrary': '0x402db0'}
```

Packed / obfuscated binaries







Some binaries got _something_ to hide

G DATA | ADVANCED ANALYTICS

Why metrics?

Measuring things is fun

Lack of metrics for sophistication

Lack of metrics for complexity

IOCs suck

- they ain‘t no metrics that aren‘t cheaply tricked

Little ability to measure suspiciousness

Little ability to masure benign-ness

Backdoor: Win32/Redsip.A
Random Info

```
.
General graph info:
SAMPLE c3f8690087a454fa45e8975fd0b8b0b76aba554f540d7c2c98d3e15512268b52
Type: PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Size: 71168
MD5: a372c78309a2a521ac4d6899d0ef2369
Name:
Type: DiGraph
Number of nodes: 126
Number of edges: 151
Average in degree:   1.1984
Average out degree:   1.1984
.
```

G DATA | ADVANCED ANALYTICS

Graph Measurement

```
.
Graph measurement data:
   157 Total functions detected with 'aflj'
   344 Count of references to local functions
     1 Count of references to data section, global variables
     0 Count of references to unrecognized locations
   238 Total API refs found via symbol xref check
     0 Count APIs w/o function xref
   180 Total referenced Strings
     0 Count of dangling strings (w/o function reference)
   438 Count of strings w/o any reference
.
```

Numbers: simplified representation, allow for distance measurement, help finding outliers and anomalies

Fat node detection

Also called spaghetti code metric

```
.
Out degree centrality, count calls, count strings:
0x10003080 0.184000 23 29
0x100023a0 0.152000 13 14
0x10002210 0.080000 4 4
0x10007dd0 0.064000 0 0
0x10005ea0 0.048000 0 21
0x10004550 0.048000 0 1
0x10001410 0.040000 3 3
0x10002060 0.032000 6 6
0x10002710 0.032000 1 1
0x10002b30 0.032000 1 1
0x100044c0 0.032000 0 0
0x100040e0 0.024000 0 1
0x10001200 0.024000 3 1
0x10003750 0.024000 5 3
0x100041b0 0.024000 0 1
0x10003890 0.024000 1 2
0x10007fe0 0.024000 0 0
0x10002f90 0.016000 9 4
0x10005290 0.016000 0 0
0x10001ee0 0.016000 12 3
.
```

interesting

awkward

interesting

GDATA | ADVANCED ANALYTICS

Math, FTW

Useful for graph complexity evaluation

```
.
Average degree connectivity per degree k:
  0 0.000000
  1 0.169811
  2 1.739130
  3 1.481481
  4 0.875000
  5 2.200000
  6 1.333333
  7 2.071429
  8 1.750000
 10 0.700000
 11 4.272727
 20 3.250000
 24 3.958333
.
```

```
.
Loose nodes 23 of total 126, thats 18.253968%
.
```

```
.
ExecSize FunctionCount ApiCount StringCount
37888 157 238 180
Per-Kilobyte ratio
4.14379222973 6.2816722973 4.75084459459
.
```

```
.
Histogram of out degree centrality:
0.0 0.0005 0.001 0.0015 0.002 0.004 0.006 0.008 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.2 0.3 0.4 0.5
79 0 0 0 0 0 25 0 9 8 7 2 0 1 1 0 0 2 0 0 0
.
```

GDATA | ADVANCED ANALYTICS

Moarrr metrics to come

Library usage

API usage variance

Global variables

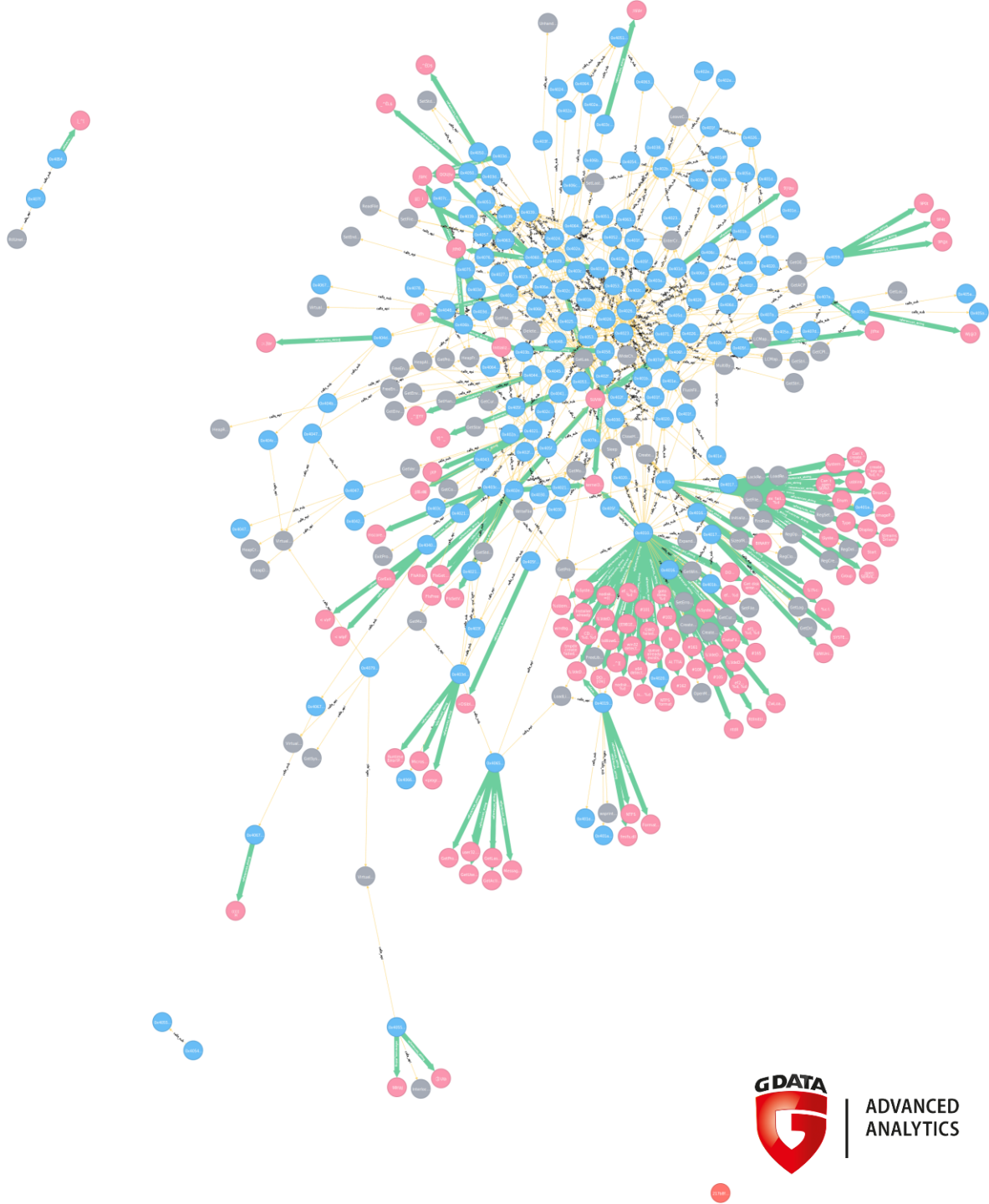
Data cross references

Neo4j

Graph database with nice documentation and tutorials and a python connector

Chosen backend (for now)

Got visualization (again)



Now what

Tool still far from being ready for use in production

Works great with dynamically linked Win32 C binaries

Works somewhat with statically linked and/or Win64 binaries

Produces funny results for C++, Delphi and such things

Barely ever crashes ;) ;)

Thank You ☺

Good Papers

„Jackdaw: Towards Automated Reverse Engineering of Large Datasets of Binaries“, Polino, Scorti, Maggi, Zanero
https://iseclab.org/media/uploads/zotero/Polino%20et%20al_2015_Jackdaw.pdf

„Distributing the Reconstruction of High-Level Intermediate Representation for Large Scale Malware Analysis“, Matrosov, Rodionov, Barbosa, Branco
https://github.com/REhints/BlackHat_2015/blob/master/slides_BHUS_2015.pdf

„Automated Reverse Engineering“, Halvar Flake
http://www.blackhat.com/presentations/win-usa-04/bh-win-04-flake.pdf