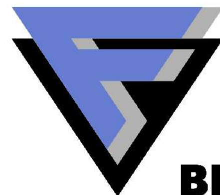
Digital Genome Mapping
Advanced Binary Analysis

Ero Carrera

Gergely Erdélyi



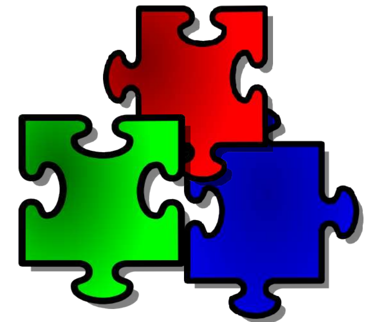
THE MAP

- Introduction
- IDA Pro + Python
- IDAPython Demo
- Introducing graphs
- Creating graphs
- Using graphs

- YOU ARE HERE

Challenges in Security Research

- Overwhelming number of samples to check
- Size of code to be analysed
- Repetitive analysis of similar samples
- Difficulties of classification
- Use of non-intuitive tools



IDA Pro

- Full name is Interactive DisAssembler Pro
- Advanced disassembler
- Supports more than 30 processors
- Supports dozens of executable file types
- Has library recognition (FLIRT)
- Extensible by scripts and plugins

F-SECURE®

Extending IDA Pro

- Using IDC
  - C-like script language
  - Dynamically typed
  - Lacks higher level construct and data types
- IDA plugins
  - Written in C/C++
  - Compiled and linked into binary code
  - Loaded as DLLs

```
#include <idc.idc>

static main ()
{
	auto baseaddr;
	auto key;
	auto c;

	baseaddr = ScreenEA();
	key = 0xff;

	while (1)  {
		c = Byte(baseaddr);

		if (c == 0)  {
			break;
		}

		PatchByte(baseaddr, c ^ key);

		baseaddr++;
	}
}
```

Extension Language Requirements

- Easy to learn, remember and read
- Supports easy refactoring
- User error tolerant
- Comes with a complete environment
- Provides complex data structures
- Open source
- Free (as in Free beer and Freedom)

Python

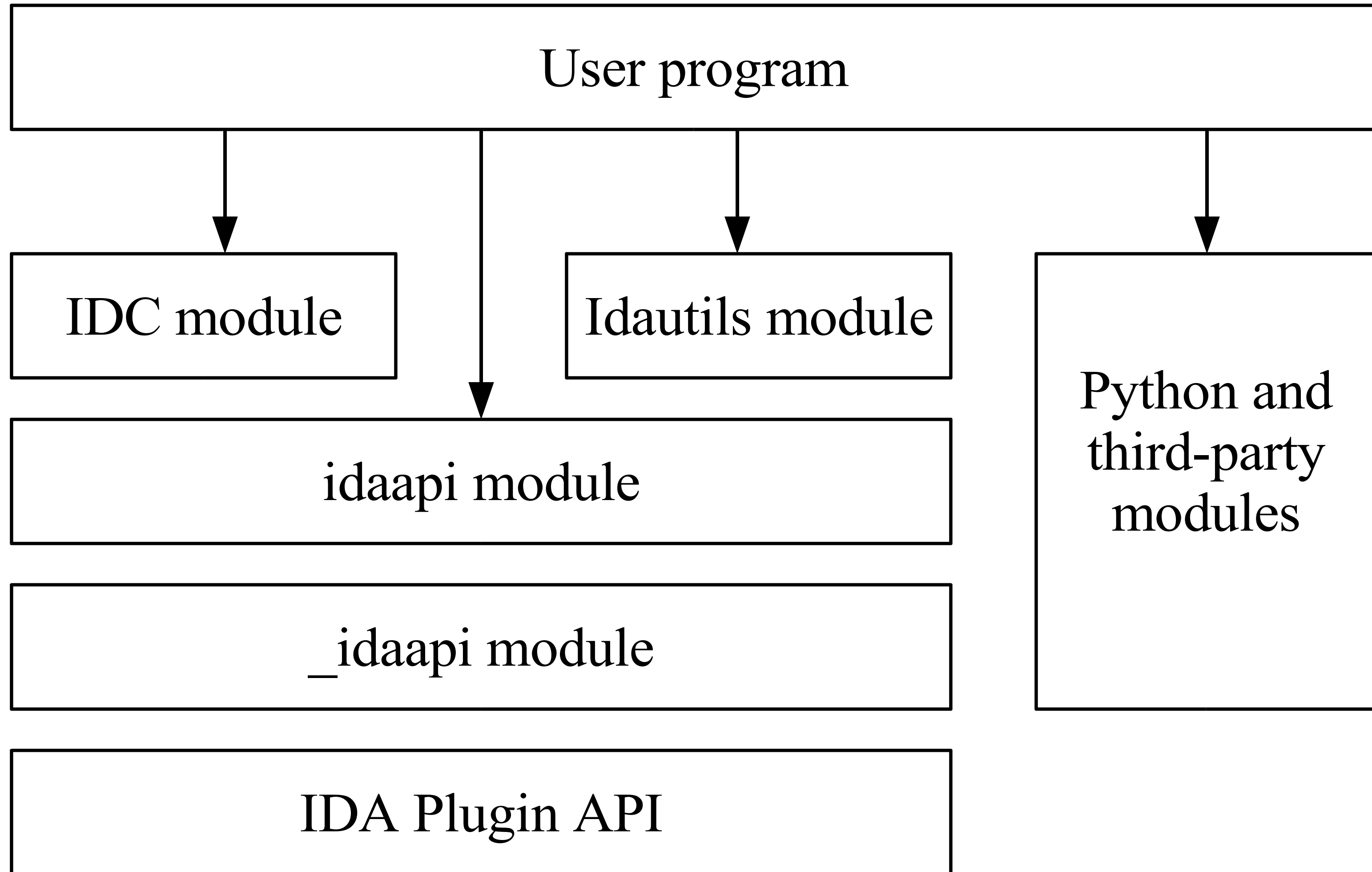
- Interpreted
- Interactive
- Object-oriented
- Dynamically typed
- Supports modules
- Has classes and exceptions
- Very high level data types
  - lists
  - tuples
  - hashes (dictionaries)
- Extensive libraries
  - OS support
  - Threading
  - Data processing
  - Networking
- Runs on...
  - Linux, *NIX
  - Windows, OS/2
  - Amiga, etc

...and a lot more

IDA + Python = IDAPython

- IDAPython is a plugin for IDA Pro
- Provides access for most of IDA Plugin API
  - Functions wrapped with SWIG
  - IDA Plugin API is provided in the module '_idaapi'
- Python libraries and extensions can be utilized
- IDAPython has an IDC compatiblity module
- The 'idautils' module provides abstract utility functions

Structure of IDAPython

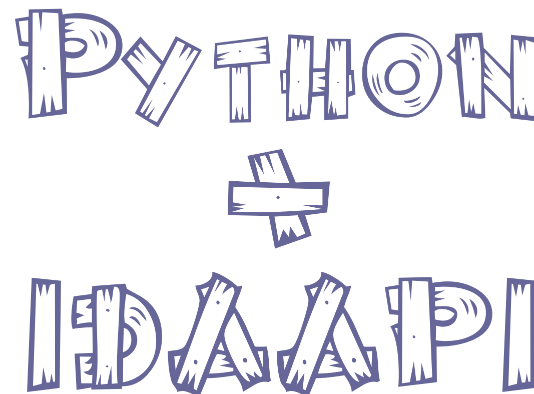


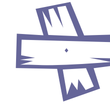
F-SECURE®

Plain IDC

```
#include <idc.idc>

static main()
{
    auto ea, func, ref;

    // Get current ea
    ea = ScreenEA();

    // Loop from start to end in the current segment
    for (func=SegStart(ea);
         func != BADADDR && func < SegEnd(ea);
         func=NextFunction(func))
    {
        // If the current address is function process it
        if (GetFunctionFlags(func) != -1)
        {
            Message("Function %s at 0x%x\n", GetFunctionName(func), func);

            // Find all code references to func
            for (ref=RfirstB(func); ref != BADADDR; ref=RnextB(func, ref))
            {
                Message("  called from %s(0x%x)\n", GetFunctionName(ref), ref);
            }
         }
    }
}
```

```
from idaapi import *

# Get current ea
ea = get_screen_ea()

# Get segment class
seg = getseg(ea)

# Loop from segment start to end
func = get_func(seg.startEA)

while func != None and func.startEA < seg.endEA:
    funcea = func.startEA
    print "Function %s at 0x%x" % (GetFunctionName(funcea), funcea)

    ref = get_first_cref_to(funcea)

    while ref != BADADDR:
        print "  called from %s(0x%x)" % (get_func_name(ref), ref)
        ref = get_next_cref_to(funcea, ref)

        func = get_next_func(funcea)
```



Python

IDAUtils

```
from idautils import *

# Get current ea
ea = ScreenEA()

# Loop from start to end in the current segment
for funcea in Functions(SegStart(ea), SegEnd(ea)):
    print "Function %s at 0x%x" %
          (GetFunctionName(funcea), funcea)

    # Find all code references to funcea
    for ref in CodeRefsTo(funcea, 1):
        print "  called from %s(0x%x)" %
              (GetFunctionName(ref), ref)
```

THE MAP

- Introduction
- IDA Pro + Python
- IDAPython Demo
- Introducing graphs
- Creating graphs
- Using graphs

● YOU ARE HERE

Availability

IDAPython distribution site:

http://www.d-dome.net/idapython/

THE MAP

- Introduction
- IDA Pro + Python
- IDAPython Demo
- Introducing graphs
- Creating graphs
- Using graphs

- YOU ARE HERE

Introducing Graphs

- Visual output
  - Graphs provide ways of visualizing complex structures.
  - Human brain is good at capturing patterns, computer has to be taught
- Analysis playground
  - Algorithms can be applied to graphs

Mimail.A

F-SECURE®

Mimail.C

F-SECURE®

Error creating window

sub_402C7F
.
.
..
\n%s%s

sub_401250

sub_402E09

sub_40491F
darkprofits.com
darkprofits.net
www.

j_GetModuleHandleA

sub_401381

sub_405BCC

j_strchr

sub_4012ED

sub_40331
{9BA05972-F6A8-11CF-A4
Line %d:
Line %d:
LINE %d
Line %d:
LINE %d
Line %d:
LINE %d
Line %d:
LINE %d
Line %d:
LINE %d
Line %d:
LINE %d
Line %d:
LINE %d

sub_40130C

sub_401E58
HELO localhost
HELO localhost
MAIL FROM:<%s>
RCPT TO:<%s>
DATA
DATA
From: %s
To: %s
Reply-To: %s
.
.
QUIT
QUIT

sub_40126A

sub_401401

sub_403FF7
Domain: '%s'
MX: '%s'
Lookup failed
hostent() error: %d
HELO localhost
MAIL FROM:<%s>
RCPT TO:<%s>
DATA
DATA
%s %d
From: %s
To: <%s>
Reply-To: <%s>
Subject: %s
.
.
QUIT
QUIT
Failed to connect: '%s'

j_strlen

sub_402987

sub_403042

SoftwareMicrosoftWindowsCurrentVersionExplorerShell Folders

j_strncmp

sub_401469
ab
rb

sub_40171A

sub_401B81
rb
%.8X%.8X

sub_401A58
------%s

X-Mailer: The Bat! (v1.62)
X-Priority: 1 (High)
Subject: Re[2]: our private photos
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=%s
--%s
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

EnumValueA

sub_402EC6
Read error
Memory allocation error, file %s, size %d
Failed to open '%s'

sub_4056B4

sub_4022F2
250
220
354
221

sub_402C3E
ab
%s

sub_40541D

sub_4058AC

j_FileTimeToDosDateTime

j_GetSystemTimeAsFileTime

j__filelength

j__fileno

j_fclose

j_fopen

j_fread

j_free

j_fwrite

j_malloc

j_GetTickCount

j_GetFileSize

sub_403163

j_sprintf

Hello Dear!,

Finally i've found possibility to right u, my lovely gi
All our photos which i've made at the beach (even when u're w
photos are great! This evening i'll come and we'll make the

THE MAP

- Introduction
- IDA Pro + Python
- IDAPython Demo
- Introducing graphs
- Creating graphs
- Using graphs

● YOU ARE HERE

Creating Graphs

- What can be show in a graph?
  - Just *Nodes* and *Edges*
- How do we represent a virus in a graph?
  - Viruses are like any other program
  - Programs are generally composed of functions
  - Functions are individual parts, used as building blocks
  - We could show the relationships between them

Infrastructure

In order to represent them we created the following software infrastructure:

- Data exporting (From IDA DB into an XML format)
- Data loader (Parses the data into easily manipulable objects)
- Methods of outputting such objects and their relations as graphs

Exporting data

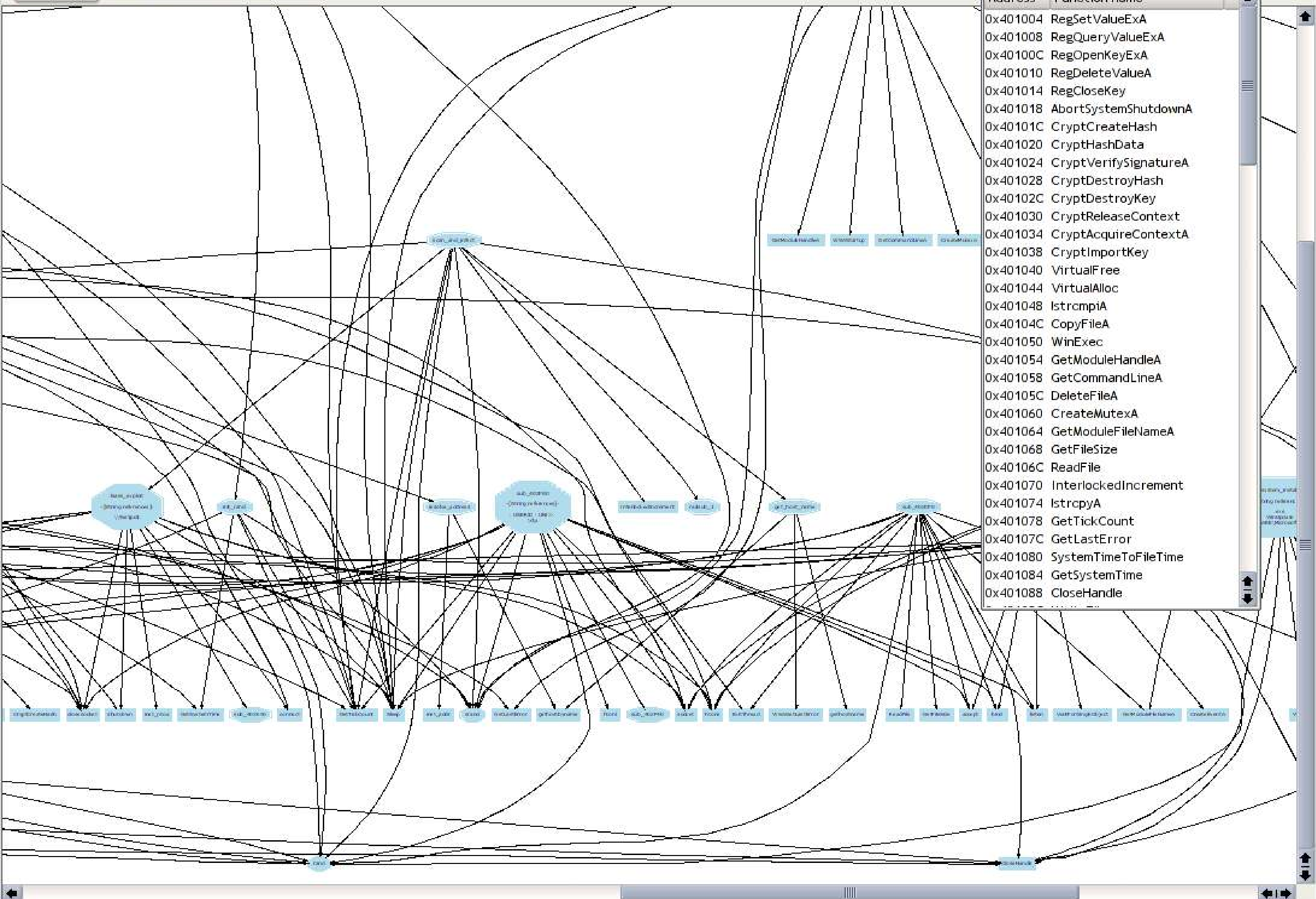
An IDC script exports information from IDA into XML:

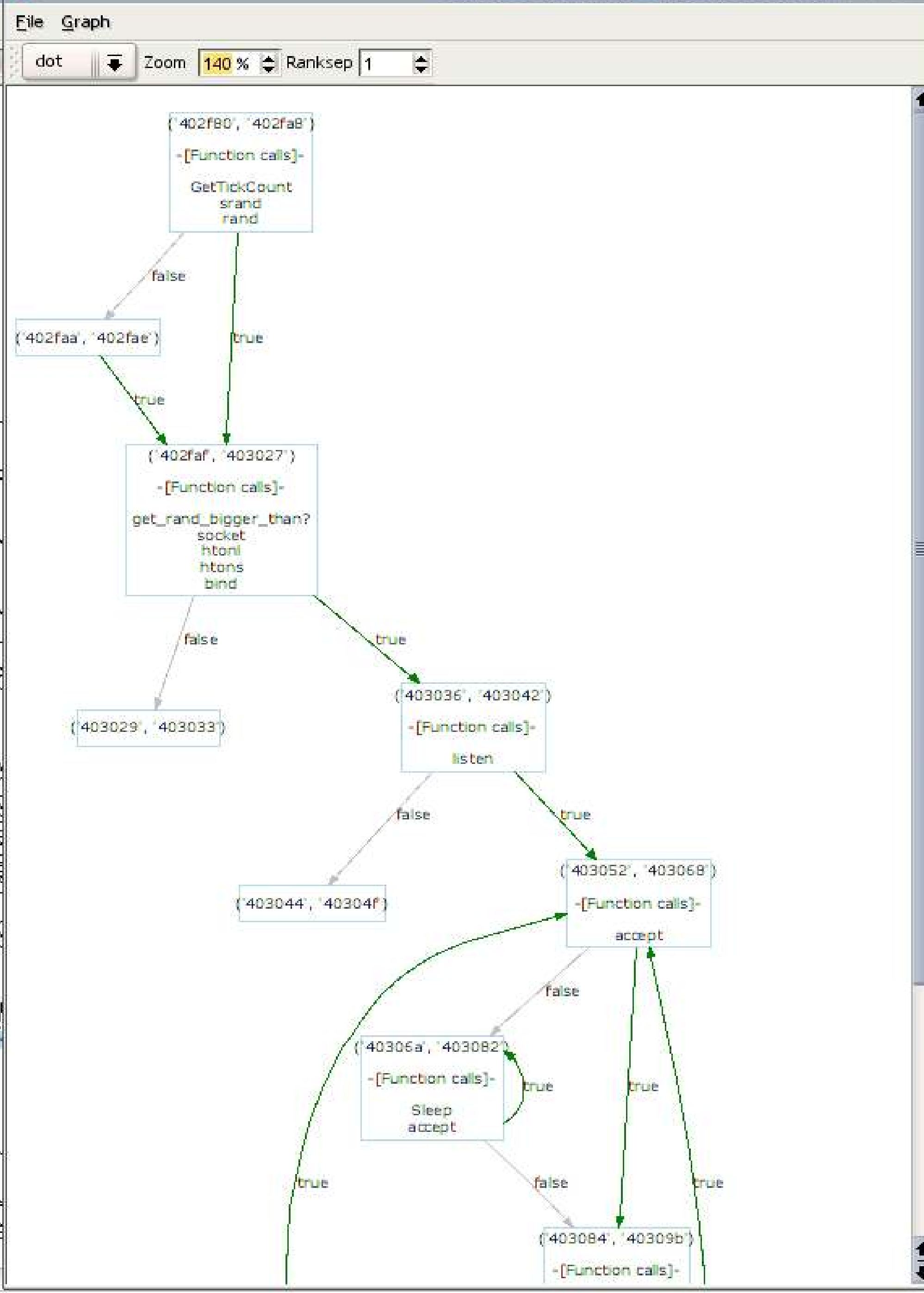
- Every defined function and instruction
- The full binary representation
- Mnemonic and operands as shown by IDA
- All the defined referenced strings and names
- All code references
- Code flags: whether the code is flow
- Miscellaneous flags: whether a function belongs to a known library or is a thunk

```
<function name="" is_library="true" is_thunk="true">

<i at="401000" is_code="true"><d>31c0</d><mnem>xor</mnem><o
n="0" t="1">eax</o><o n="1" t="1">eax</o><ref_data
from="40102a"/></i>

[...]

<i at="40100e" is_code="true"
is_flow="true"><d>740f</d><mnem>jz</mnem><o n="0"
t="7">4198431</o></i>

[...]

<i at="40101a" is_code="true"
is_flow="true"><d>b803000000</d><mnem>mov</mnem><o n="0"
t="1">eax</o><o n="1" t="5">3</o></i>

<i at="40101f" is_code="true"
is_flow="true"><d>c3</d><mnem>retn</mnem><ref
from="40100e"/></i>
```

Data loader

- REML Python loader
  - Provides an interface to the data in a binary, such as functions; allowing to handle them as objects
- Graph handling tools


IDA RevEngML Broser [korgo-a.unp]
File
Graph
Windows
dot
Zoom
100 %
Ranksep
1
reveng_browser.py
Address
Function name
0x401004 RegSetValueExA
0x401008 RegQueryValueExA
0x40100C RegOpenKeyExA
0x401010 RegDeleteValueA
0x401014 RegCloseKey
0x401018 AbortSystemShutdownA
0x40101C CryptCreateHash
0x401020 CryptHashData
0x401024 CryptVerifySignatureA
0x401028 CryptDestroyHash
0x40102C CryptDestroyKey
0x401030 CryptReleaseContext
0x401034 CryptAcquireContextA
0x401038 CryptImportKey
0x401040 VirtualFree
0x401044 VirtualAlloc
0x401048 lstrcmpiA
0x40104C CopyFileA
0x401050 WinExec
0x401054 GetModuleHandleA
0x401058 GetCommandLineA
0x40105C DeleteFileA
0x401060 CreateMutexA
0x401064 GetModuleFileNameA
0x401068 GetFileSize
0x40106C ReadFile
0x401070 InterlockedIncrement
0x401074 lstrcpyA
0x401078 GetTickCount
0x40107C GetLastError
0x401080 SystemTimeToFileTime
0x401084 GetSystemTime
0x401088 CloseHandle



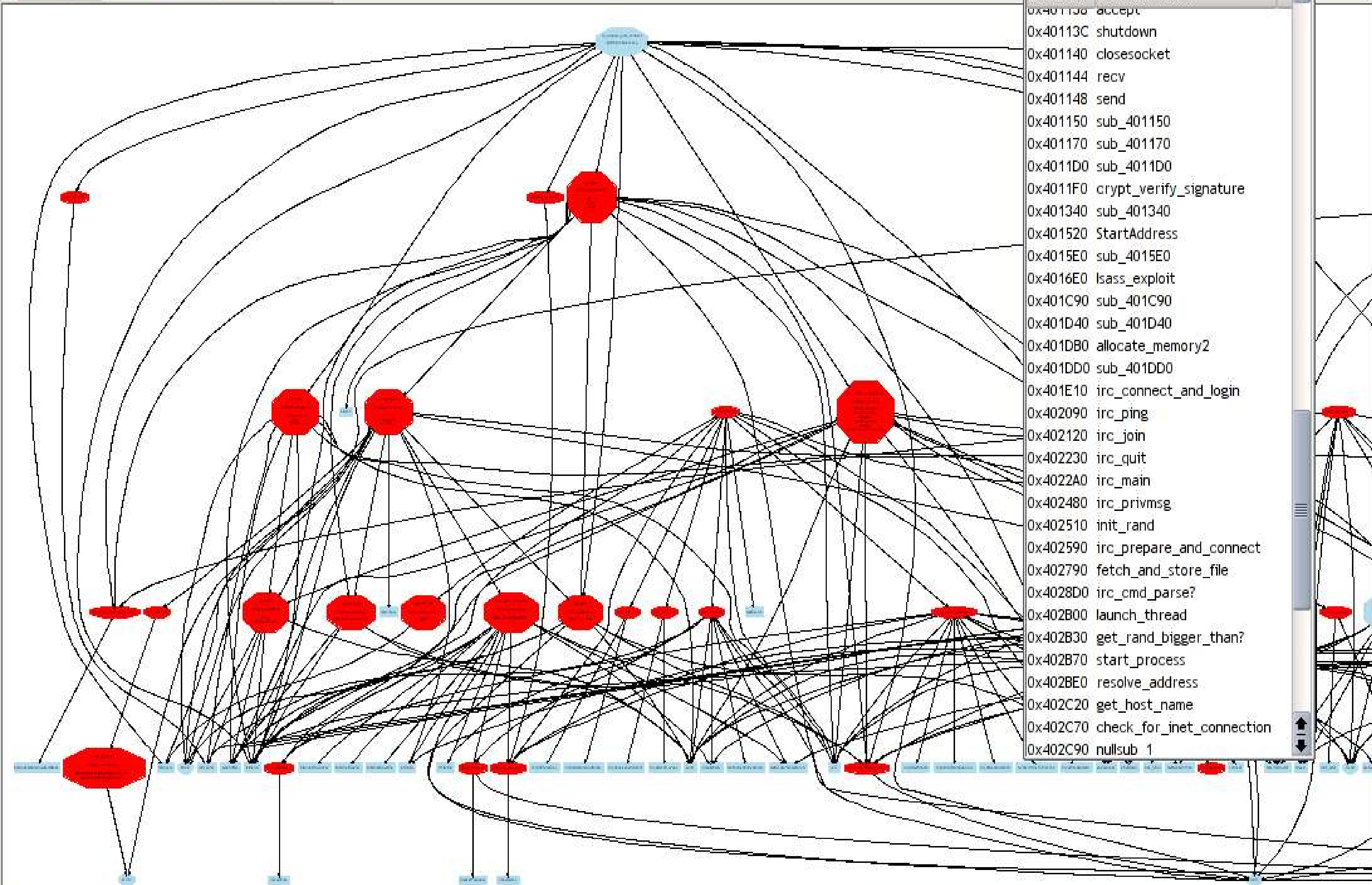
File Graph Windows

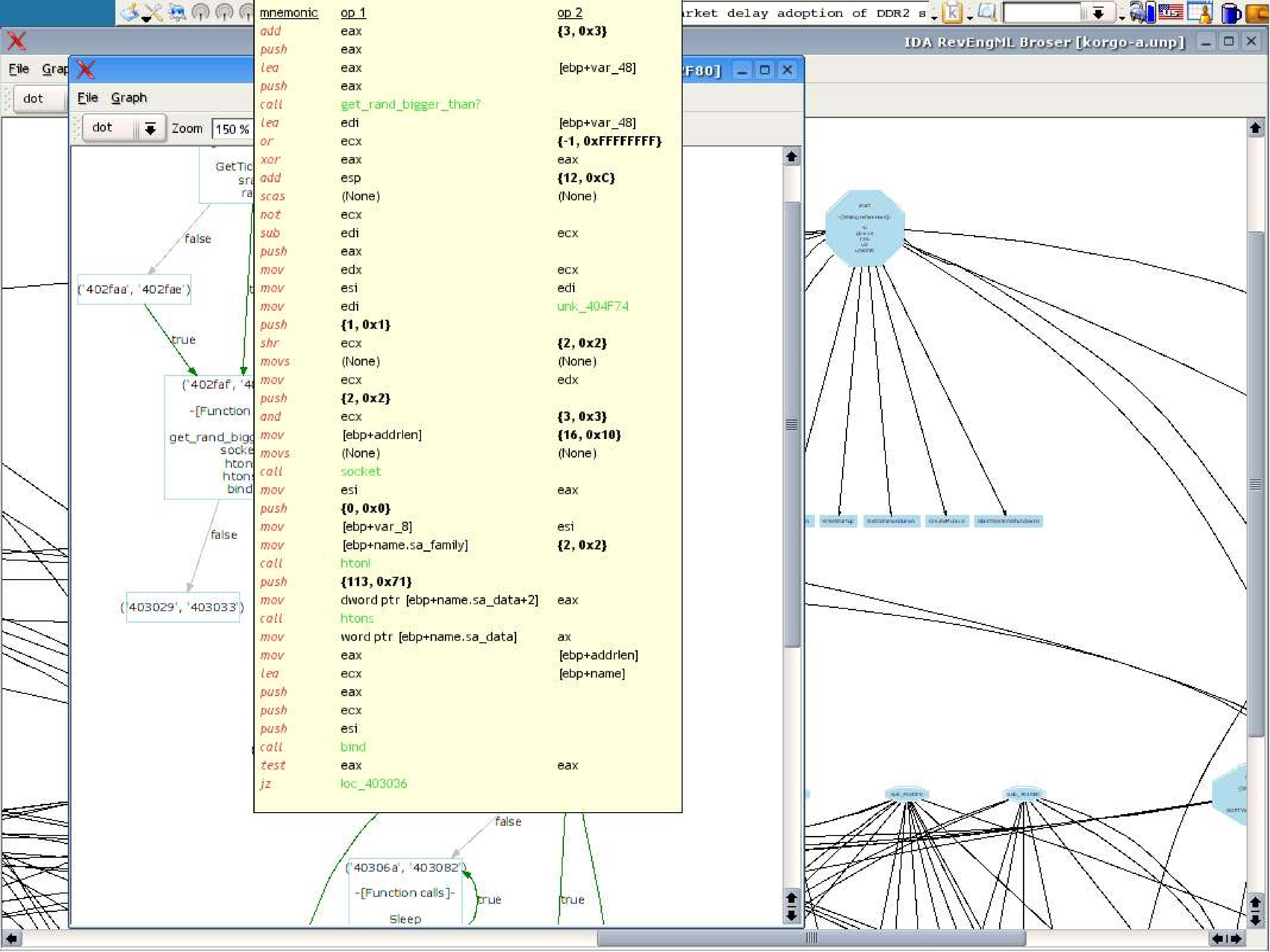
dot Zoom 100 %

IDA RevEngML Broser [korgo-a.unp]

reveng_browser.py

Address	Function name
0x401004	RegSetValueExA
0x401008	RegQueryValueExA
0x40100C	RegOpenKeyExA
0x401010	RegDeleteValueA
0x401014	RegCloseKey
0x401018	AbortSystemShutdownA
0x40101C	CryptCreateHash
0x401020	CryptHashData
0x401024	CryptVerifySignatureA
0x401028	CryptDestroyHash
0x40102C	CryptDestroyKey
0x401030	CryptReleaseContext
0x401034	CryptAcquireContextA
0x401038	CryptImportKey
0x401040	VirtualFree
0x401044	VirtualAlloc
0x401048	lstrcmpiA
0x40104C	CopyFileA
0x401050	WinExec
0x401054	GetModuleHandleA
0x401058	GetCommandLineA
0x40105C	DeleteFileA
0x401060	CreateMutexA
0x401064	GetModuleFileNameA
0x401068	GetFileSize
0x40106C	ReadFile
0x401070	InterlockedIncrement
0x401074	lstrcpyA
0x401078	GetTickCount
0x40107C	GetLastError
0x401080	SystemTimeToFileTime
0x401084	GetSystemTime
0x401088	CloseHandle


IDA RevEngML Broser [korgo-a.unp]
File Graph Windows
dot
Zoom 60 %
Ranksep 1
reveng_browser.py
Address Function name
0x40113C shutdown
0x401140 closesocket
0x401144 recv
0x401148 send
0x401150 sub_401150
0x401170 sub_401170
0x4011D0 sub_4011D0
0x4011F0 crypt_verify_signature
0x401340 sub_401340
0x401520 StartAddress
0x4015E0 sub_4015E0
0x4016E0 lsass_exploit
0x401C90 sub_401C90
0x401D40 sub_401D40
0x401DB0 allocate_memory2
0x401DD0 sub_401DD0
0x401E10 irc_connect_and_login
0x402090 irc_ping
0x402120 irc_join
0x402230 irc_quit
0x4022A0 irc_main
0x402480 irc_privmsg
0x402510 init_rand
0x402590 irc_prepare_and_connect
0x402790 fetch_and_store_file
0x4028D0 irc_cmd_parse?
0x402B00 launch_thread
0x402B30 get_rand_bigger_than?
0x402B70 start_process
0x402BE0 resolve_address
0x402C20 get_host_name
0x402C70 check_for_inet_connection
0x402C90 nullsub_1



IDA RevEngML Broser [korgo-a.unp]
File
Graph
dot
Zoom
150 %
mnemonic op 1 op 2
add eax {3, 0x3}
push eax
lea eax [ebp+var_48]
push eax
call get_rand_bigger_than?
lea edi [ebp+var_48]
or ecx {-1, 0xFFFFFFFF}
xor eax eax
add esp {12, 0xC}
scas (None) (None)
not ecx
sub edi ecx
push eax
mov edx ecx
mov esi edi
mov edi unk_404F74
push {1, 0x1}
shr ecx {2, 0x2}
movs (None) (None)
mov ecx edx
push {2, 0x2}
and ecx {3, 0x3}
mov [ebp+addrlen] {16, 0x10}
movs (None) (None)
call socket
mov esi eax
push {0, 0x0}
mov [ebp+var_8] esi
mov [ebp+name.sa_family] {2, 0x2}
call htonl
push {113, 0x71}
mov dword ptr [ebp+name.sa_data+2] eax
call htons
mov word ptr [ebp+name.sa_data] ax
mov eax [ebp+addrlen]
lea ecx [ebp+name]
push eax
push ecx
push esi
call bind
test eax eax
jz loc_403036
false
('402faa', '402fae')
true
('402faf', '4
-[Function
get_rand_bigg
false
('403029', '403033')
('40306a', '403082')
-[Function calls]-
Sleep
true

THE MAP

- Introduction
- IDA Pro + Python
- IDAPython Demo
- Introducing graphs
- Creating graphs
- Using graphs

● YOU ARE HERE

Comparison

Our work usually faces the need to identify and classify malware into families.

Members of a family, must share some similarities. Those become quite evident in graphs of their call trees.

Binary comparison, the algorithm

We want to map functions from one binary to another:

- We use the Windows API as a ground set for matching
- We match the API callers and iteratively ascend the graph

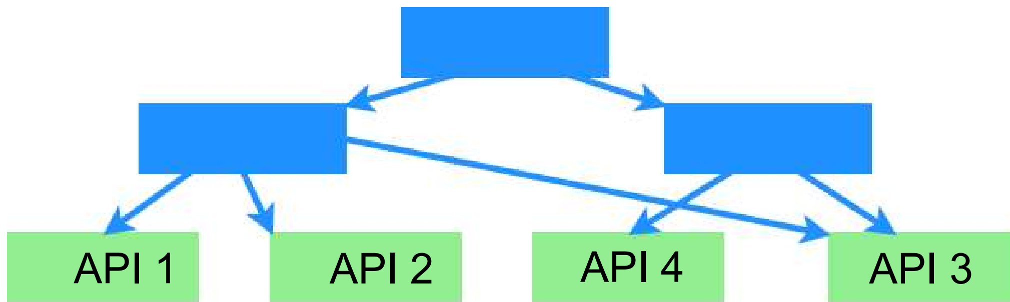
Signatures

For deciding whether two functions match, a signature must to agree, we use two approaches

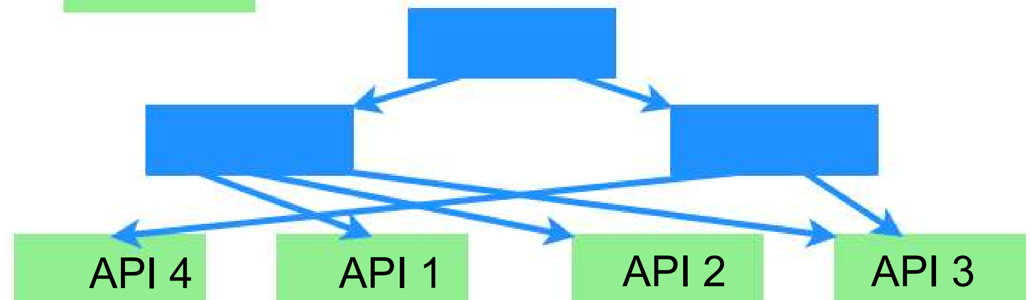
- Call Tree Signatures. Giving an idea of the connectivity of a function within the malware body
- CFG signatures. Relying only on the internal structure of a function

Call Tree Signatures

Call Tree Signatures

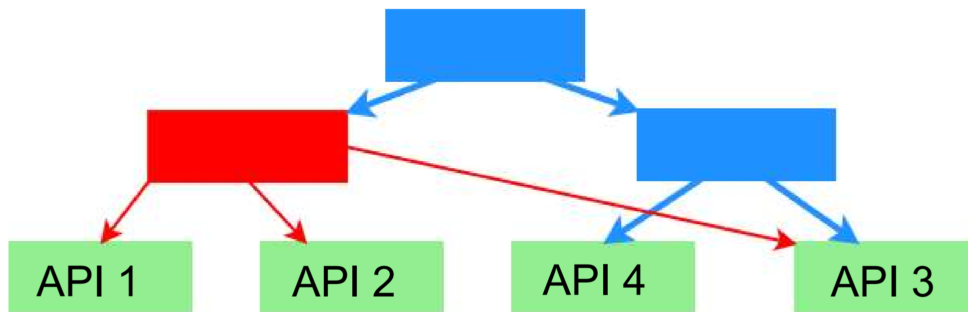


Sample A

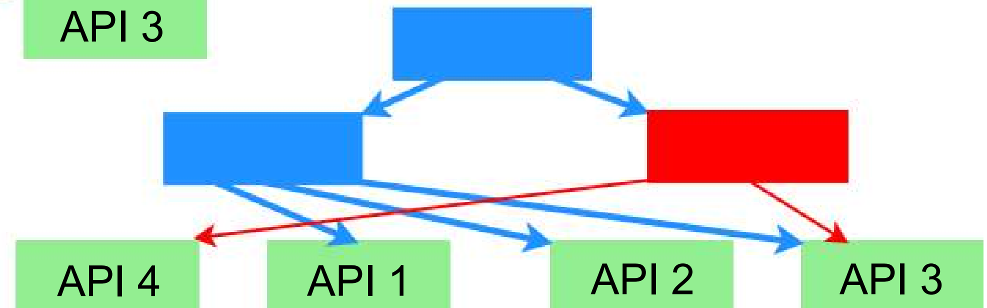


Sample B

Call Tree Signatures

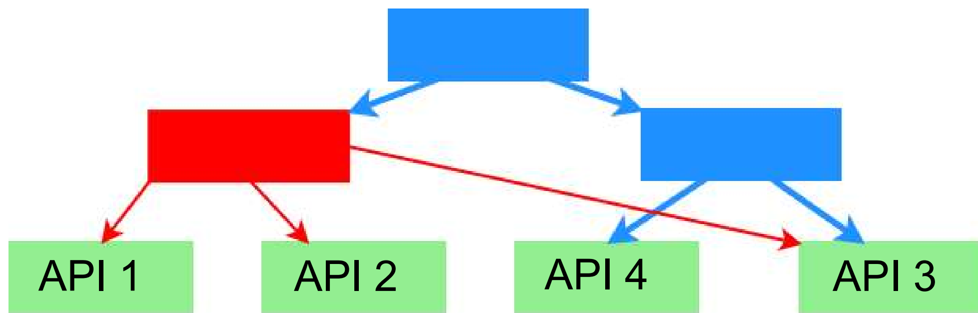


Sample A

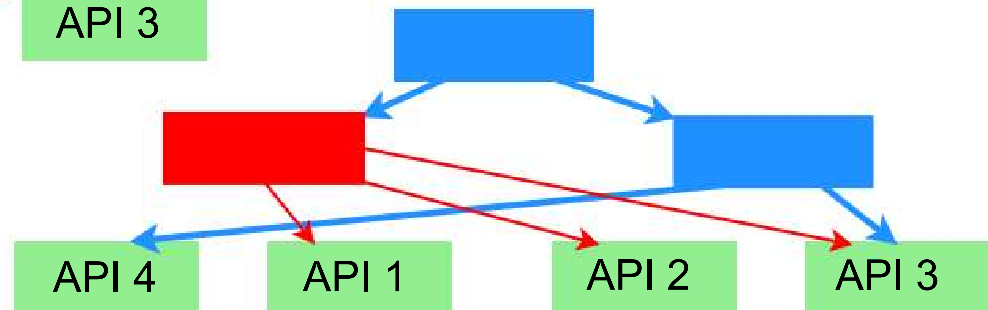


Sample B

Call Tree Signatures



Sample A

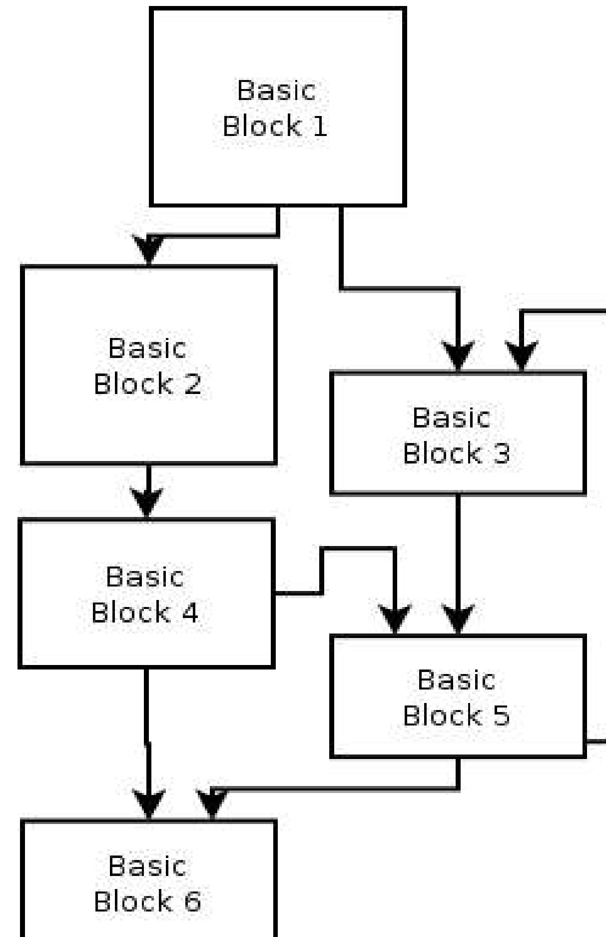


Sample B

Control Flow Graph Signatures

We number the basic blocks forming a function. The signature is the sorted list of edges linking the blocks.

(1,2) (1,3)
(2,4)
(3,5)
(4,5) (4,6)
(5,3) (5,6)



F-SECURE®

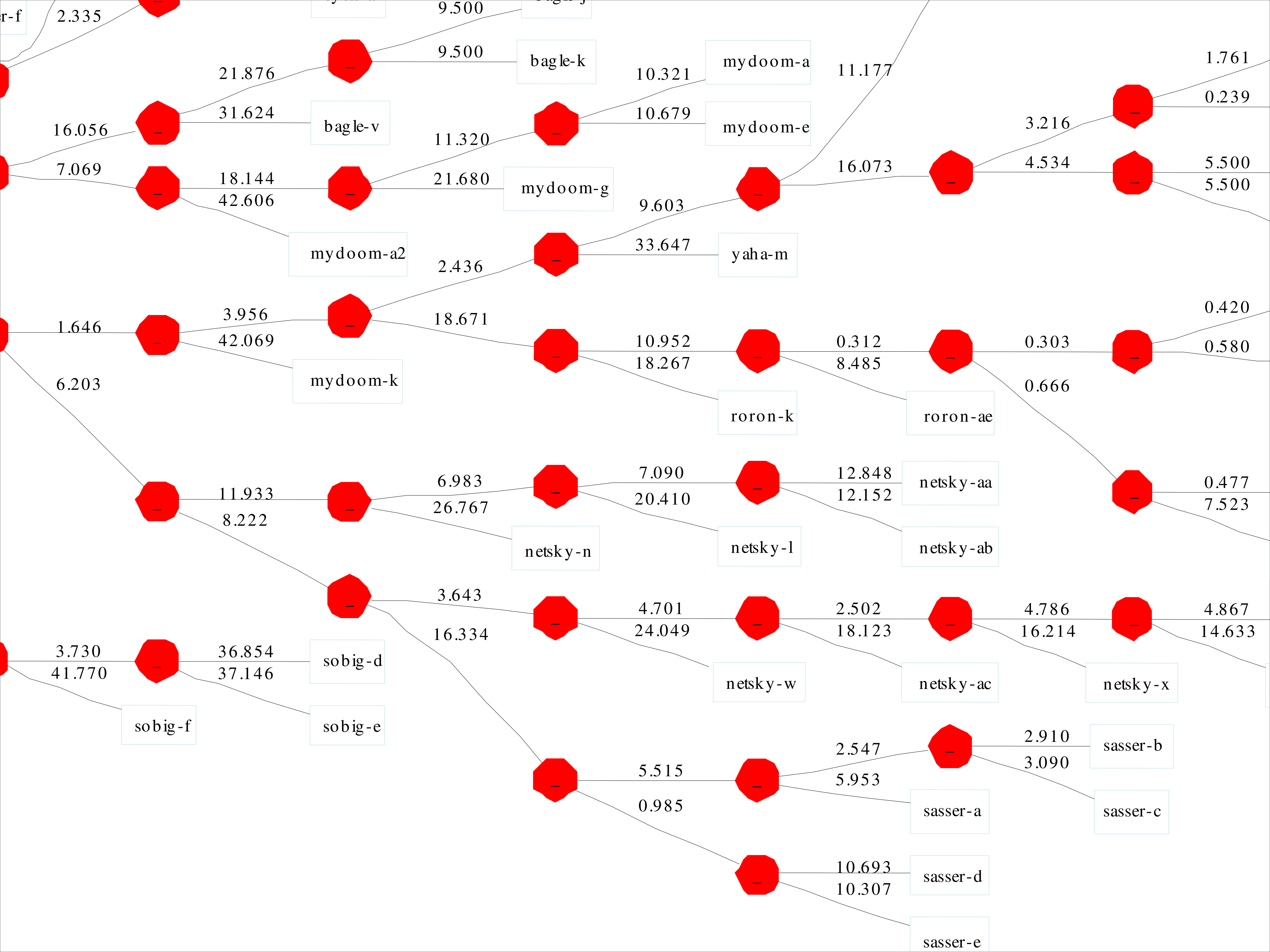
Possible Uses

A measure of similarity gives us ways to:

- Classify: Phylogenetics based on similarity matrices
- Identify work by same authors or groups

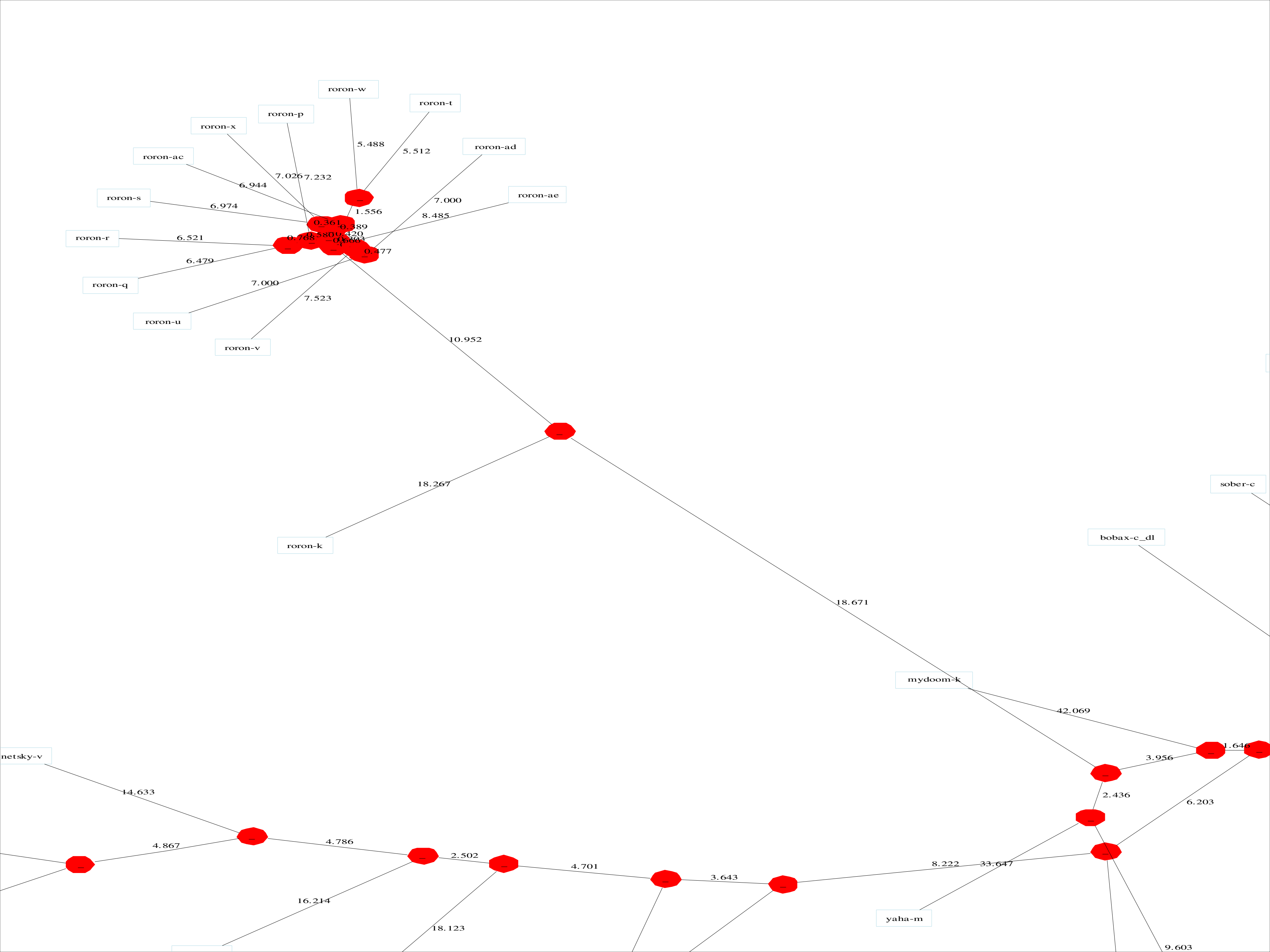
Phylogenetics (From Wikipedia, the free encyclopedia.): *Phylogenetics is the taxonomical classification of organisms based on how closely they are related in terms of evolutionary differences*

mimail-a
4.966
5.034 mimail-b
1.184
2.500 mimail-e
0.000
6.316 mimail-d
2.500 mimail-f
2.389
0.000
2.500 mimail-g
8.611 mimail-c
2.907
2.500 mimail-h
4.996
0.593
4.830 mimail-l
3.897
5.170
3.911
mimail-m
6.603
2.617 mimail-i
6.310
2.383
19.299 mimail-s
mimail-j
8.117
38.430 bagle-i
38.570 bagle-m
0.262
28.196 mimail-q
klez-a
18.849
10.532
0.238
38.500 bagle-w
7.226
38.500
16.100 bugbear-e
10.468 klez-b
46.964 Cmjspy.y
46.274 bagle-ac
bagle-x
15.900
bugbear-g
8.028
3.635
47.023 sober-a
3.795
klez-c
6.000
46.977 sober-c
sober-d
23.972 bugbear-b
1.180
1.875
21.755
33.000
8.365
6.000 klez-d
1.205
4.375
45.245 cycle-a
bagle-j
klez-e
5.036
33.000 sober-f
11.679
2.335
9.500
44.321
4.964 klez-f
9.500 bagle-k
mydoom-a
1.761
0.137
sober-b
21.876
10.321
11.177
0.239
6.500 klez-g
5.512 roron-t
0.957
31.624 bagle-v
10.679 mydoom-e
16.056
3.216
6.500
5.488
11.320
1.556
klez-j
roron-w
0.508
7.069
4.534
5.500 klez-h
18.144
16.073
21.680 mydoom-g
5.500
6.944 roron-ac
0.502
42.606
9.603
klez-i
mydoom-a2
0.389
33.647 yaha-m
0.344
1.607
4.113
2.436
6.974 roron-s
4.055
0.361
0.420
7.026
3.956
18.671
roron-x
7.162
1.646
42.069
0.000 bobax-a-dr
10.952
0.312
0.303
46.062
0.580
0.768
6.479
mydoom-k
8.485
18.267
7.232
roron-q
6.203
0.666
6.521
bobax-c_dl
roron-k
roron-ae
roron-p
roron-r
12.848 netsky-aa
7.090
6.983
0.477
7.000 roron-ad
11.933
20.410
12.152
26.767
8.222
7.523
7.000
netsky-n
netsky-l
netsky-ab
roron-v
roron-u
3.643
4.701
2.502
4.786
4.867
8.809 netsky-s
16.334
24.049
18.123
16.214
14.633
3.730
36.854 sobig-d
9.191
41.770
37.146
netsky-w
netsky-ac
netsky-x
netsky-v
netsky-u
sobig-f
sobig-e
2.910 sasser-b
2.547
5.515
3.090
5.953
sasser-a
sasser-c
0.985
10.693 sasser-d
10.307
sasser-e

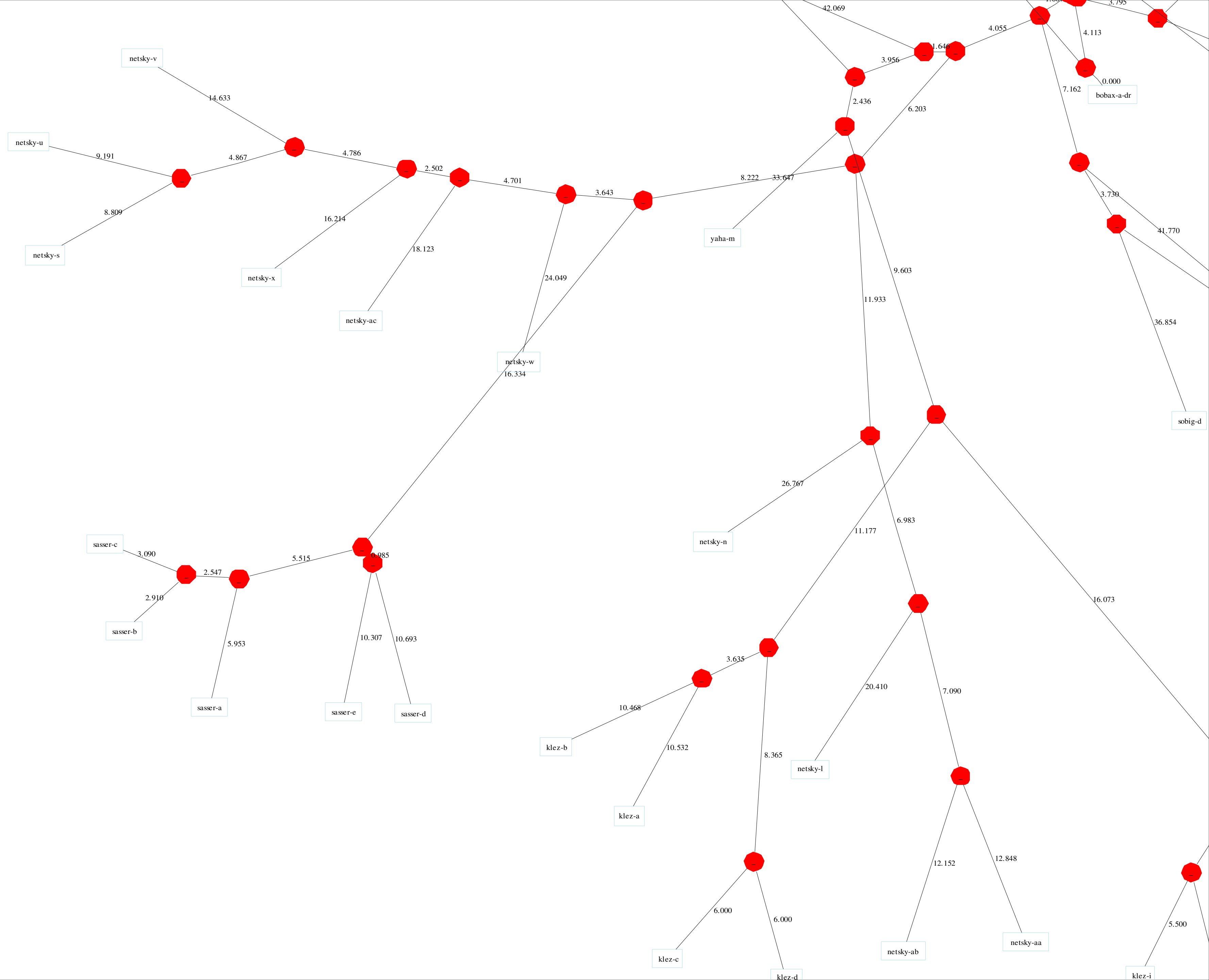


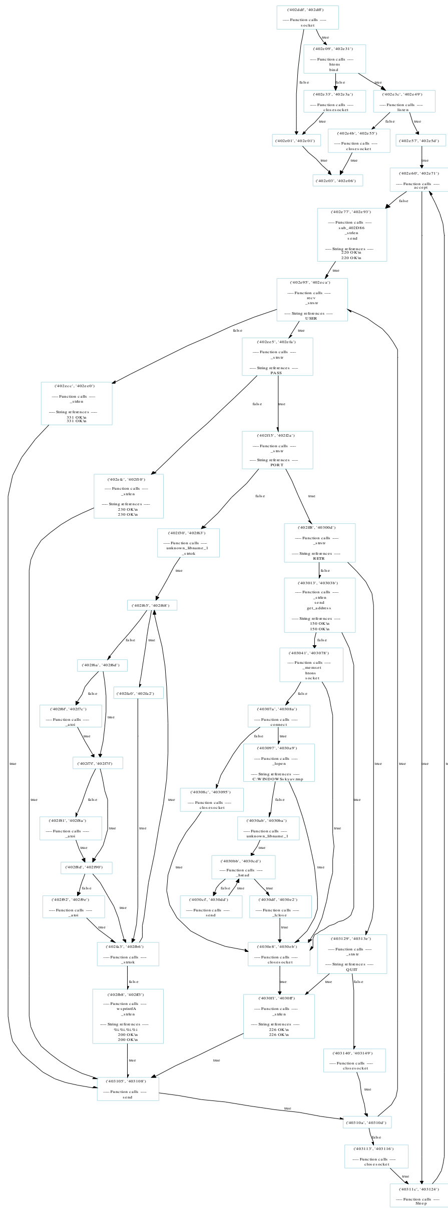
2.335
9.500
9.500
bagle-k
21.876
31.624
bagle-v
16.056
10.321
mydoom-a
10.679
mydoom-e
11.320
11.177
1.761
0.239
3.216
7.069
18.144
42.606
21.680
mydoom-g
16.073
4.534
5.500
5.500
mydoom-a2
9.603
33.647
yaha-m
2.436
1.646
3.956
42.069
mydoom-k
18.671
10.952
18.267
0.312
8.485
0.303
0.420
0.580
0.666
roron-k
roron-ae
6.203
6.983
26.767
7.090
20.410
12.848
12.152
netsky-aa
netsky-ab
netsky-l
netsky-n
11.933
8.222
0.477
7.523
3.643
16.334
4.701
24.049
2.502
18.123
4.786
16.214
4.867
14.633
netsky-w
netsky-ac
netsky-x
3.730
41.770
36.854
37.146
sobig-d
sobig-e
sobig-f
5.515
0.985
2.547
5.953
2.910
3.090
sasser-b
sasser-c
sasser-a
10.693
10.307
sasser-d
sasser-e

F-SECURE®

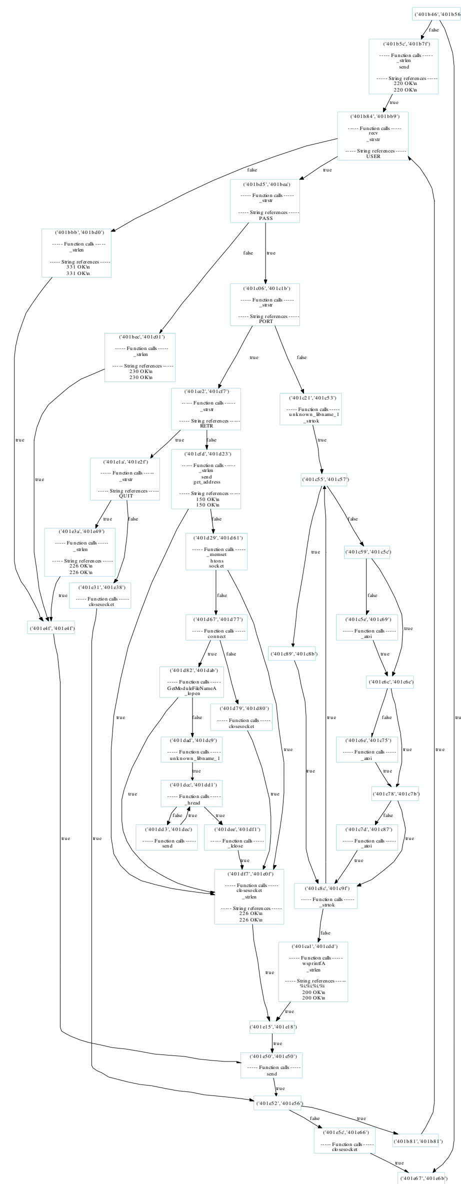


roron-w
roron-t
roron-p
roron-x
roron-ad
roron-ac
roron-ae
roron-s
roron-r
roron-q
roron-u
roron-v
5.488
5.512
7.026
7.232
6.944
7.000
6.974
8.485
1.556
0.361
0.389
0.580
0.420
0.768
6.521
0.477
6.479
7.000
7.523
10.952
18.267
roron-k
18.671
sober-c
bobax-c_dl
mydoom-k
42.069
1.646
3.956
2.436
6.203
netsky-v
14.633
4.867
4.786
2.502
4.701
3.643
8.222
33.647
16.214
18.123
yaha-m
9.603


netsky-v
14.633
netsky-u
9.191
4.867
netsky-s
8.809
4.786
2.502
4.701
3.643
8.222
33.647
16.214
netsky-x
18.123
netsky-ac
24.049
netsky-w
16.334
yaha-m
42.069
3.956
1.646
4.055
2.436
6.203
4.113
0.000
bobax-a-dr
7.162
3.795
3.730
41.770
36.854
sobig-d
9.603
11.933
26.767
netsky-n
6.983
11.177
16.073
sasser-c
3.090
2.547
5.515
0.985
sasser-b
2.910
5.953
sasser-a
10.307
10.693
sasser-e
sasser-d
3.635
10.468
klez-b
10.532
klez-a
8.365
20.410
netsky-l
7.090
12.152
12.848
netsky-ab
netsky-aa
6.000
6.000
klez-c
klez-d
5.500
klez-i



Control Flow Graph for NetSky.V FTP function

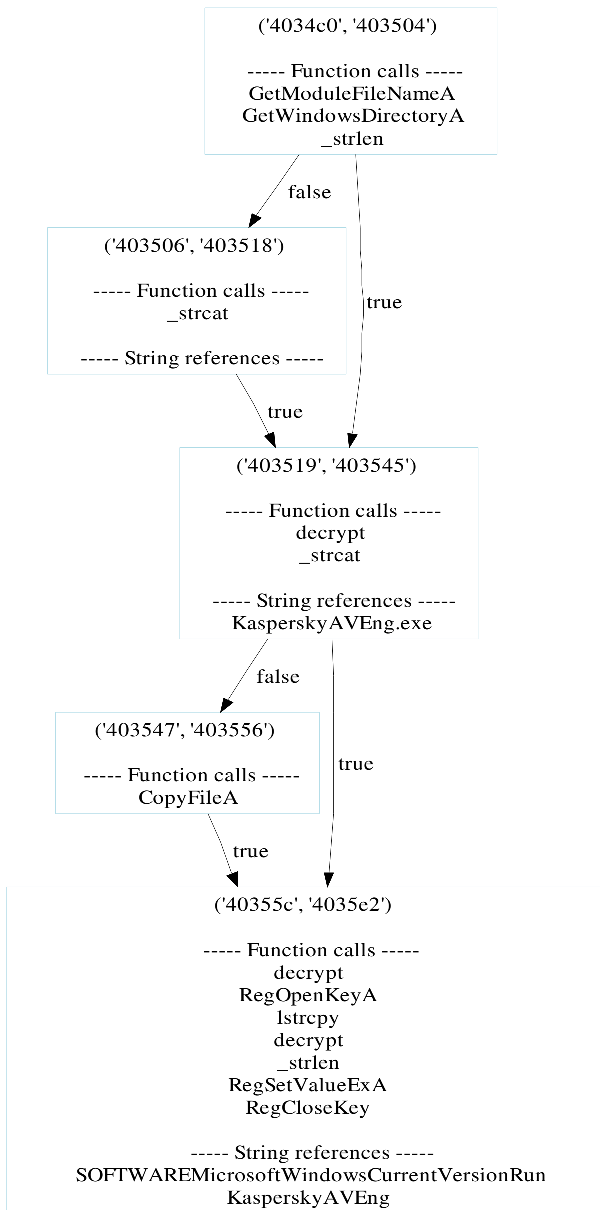


Control Flow Graph for Sasser.D FTP function

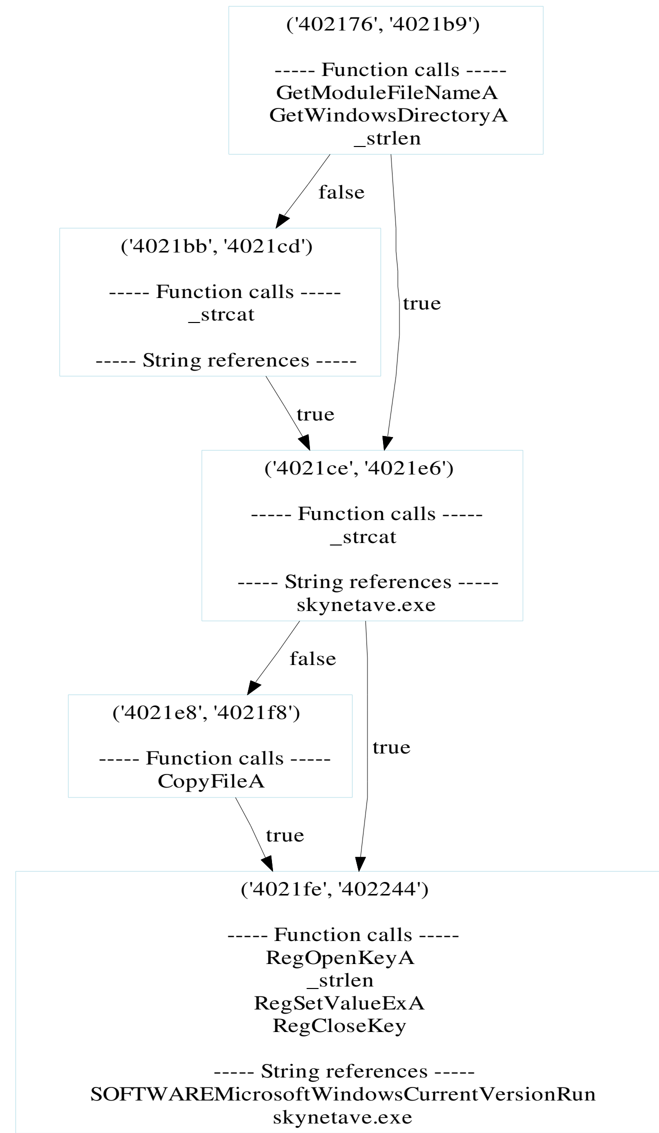
Control Flow Graph for NetSky.V FTP function

Control Flow Graph for Sasser.D FTP function

F-SECURE®



Control Flow Graph for NetSky.V Install function



Control Flow Graph for Sasser.D Install function

Conclusions

The authors believe approaches such as the one shown have an immense potential

The advantages are patent: compare and classify malware in an automatic manner

Fertile ground of possibilities for taking such methods further

Closing

IDAPython distribution site:

http://www.d-dome.net/idapython/

Questions?

Gergő

Ero