ARBOR®
NETWORKS

Tasty Malware Analysis with T.A.C.O.

Bringing Cuckoo Metadata into IDA Pro

Jason Jones

Me

- Security Architect for Arbor Networks
  - Security Research Analyst with Arbor ASERT for 3.5 yrs prior
- Previously spoken at
  - BlackHat / Ruxcon / AusCERT / REcon
- Research Interests
  - Automating reverse engineering
  - Graph theory / database applications for RE / security
  - Botnet monitoring

ARBOR® NETWORKS

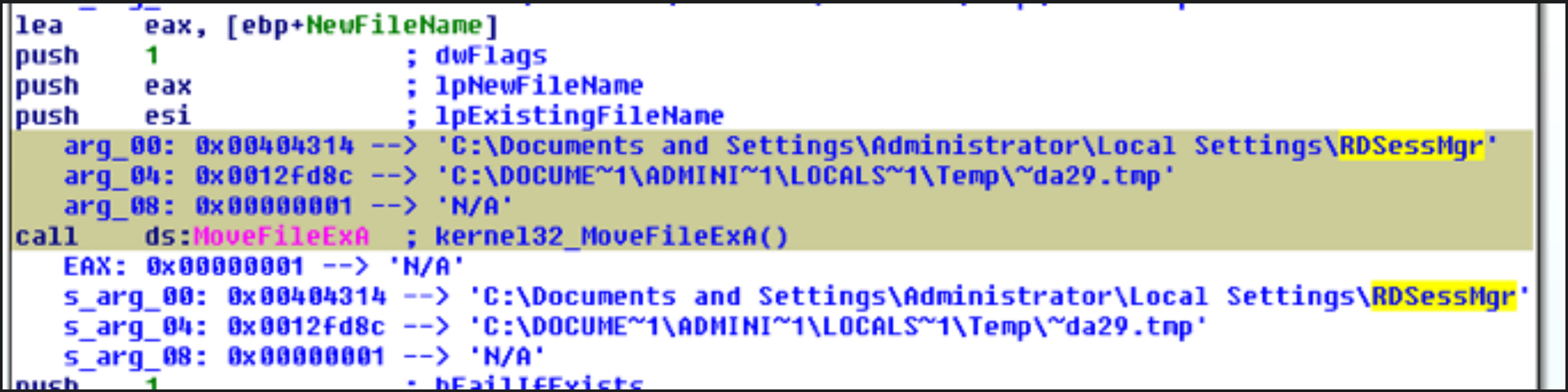
Similar Work

Similar Work

- Nothing (that I know of) uses Cuckoo as it's mechanism for propagating data into an IDB
- Inspired by similar work from many authors
- UI takes inspiration from IDAScope by Daniel Plohmann (@push_pnx)
    - Excellent plugin, in my toolbox

funcap

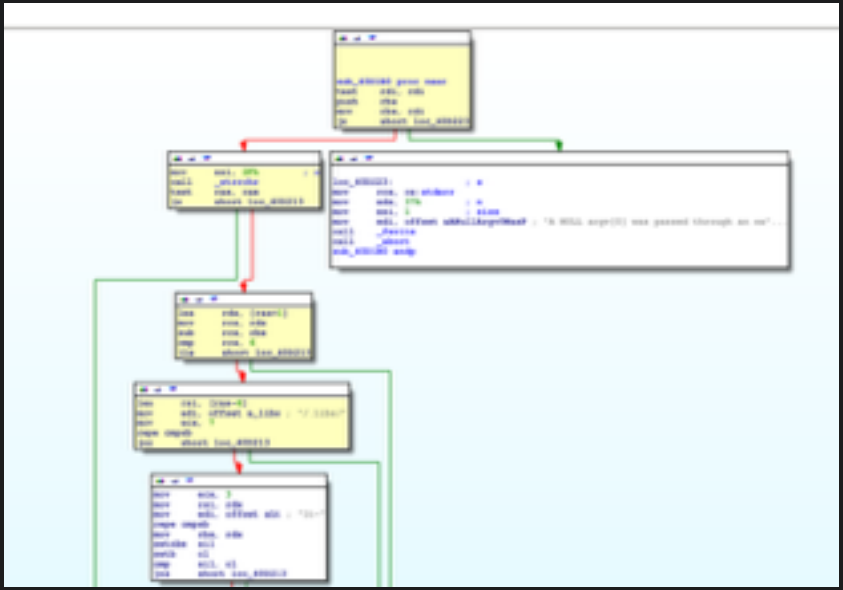
- https://github.com/deresz/funcap
- IDA Pro script to add some useful runtime info to static analysis.

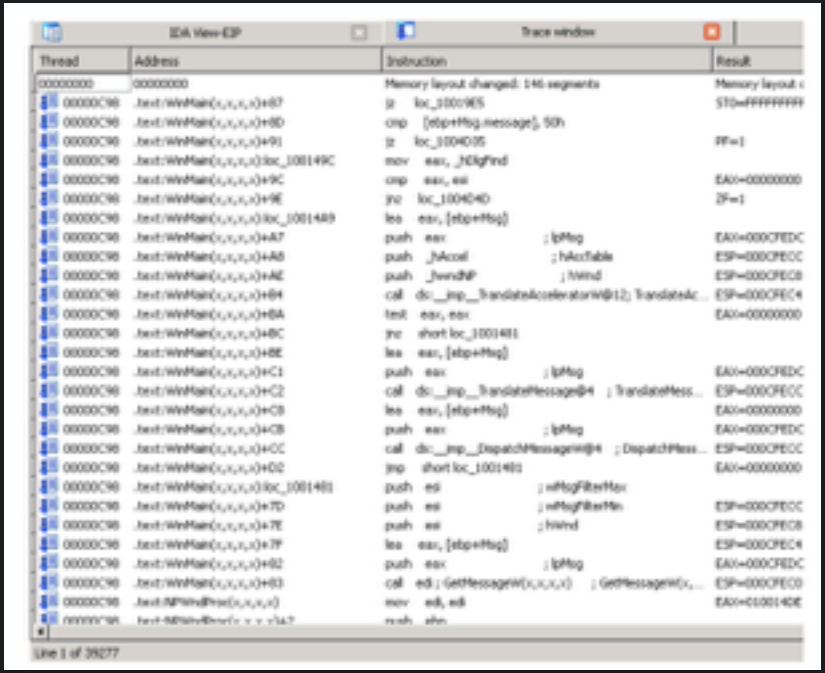


```
lea     eax, [ebp+NewFileName]
push    1                    ; dwFlags
push    eax                  ; lpNewFileName
push    esi                  ; lpExistingFileName
    arg_00: 0x00404314 --> 'C:\Documents and Settings\Administrator\Local Settings\RDSessMgr'
    arg_04: 0x0012fd8c --> 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\~da29.tmp'
    arg_08: 0x00000001 --> 'N/A'
call    ds:MoveFileExA       ; kernel32_MoveFileExA()
    EAX: 0x00000001 --> 'N/A'
    s_arg_00: 0x00404314 --> 'C:\Documents and Settings\Administrator\Local Settings\RDSessMgr'
    s_arg_04: 0x0012fd8c --> 'C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\~da29.tmp'
    s_arg_08: 0x00000001 --> 'N/A'
```

IDA Pro Pintracer

- Maintained by Hex-Rays
- Highlights executed instructions
- Can also track registers





```
IDA View-EIP                                  Trace window
Thread    Address                               Instruction                                          Result
00000000  00000000                              Memory layout changed: 146 segments                  Memory layout c
00000C98  .text:WinMain(x,x,x,x)+87             jz    loc_100C9E5                                    STD=FFFFFFFF
00000C98  .text:WinMain(x,x,x,x)+8D             cmp   [ebp+Msg.message], 50h
00000C98  .text:WinMain(x,x,x,x)+91             jz    loc_1004D35                                    PF=1
00000C98  .text:WinMain(x,x,x,x):loc_100149C    mov   eax, _hDlgFind
00000C98  .text:WinMain(x,x,x,x)+9C             cmp   eax, esi                                       EAX=00000000
00000C98  .text:WinMain(x,x,x,x)+9E             jnz   loc_100404D                                    ZF=1
00000C98  .text:WinMain(x,x,x,x):loc_10014A9    lea   eax, [ebp+Msg]
00000C98  .text:WinMain(x,x,x,x)+A7             push  eax                     ; lpMsg                EAX=000CFEDC
00000C98  .text:WinMain(x,x,x,x)+A8             push  _hAccel                 ; hAccTable            ESP=000CFECC
00000C98  .text:WinMain(x,x,x,x)+AE             push  _hwndNP                 ; hWnd                 ESP=000CFEC8
00000C98  .text:WinMain(x,x,x,x)+B4             call  ds:__imp__TranslateAcceleratorW@12; TranslateAc...  ESP=000CFEC4
00000C98  .text:WinMain(x,x,x,x)+BA             test  eax, eax                                       EAX=00000000
00000C98  .text:WinMain(x,x,x,x)+BC             jnz   short loc_1001481
00000C98  .text:WinMain(x,x,x,x)+BE             lea   eax, [ebp+Msg]
00000C98  .text:WinMain(x,x,x,x)+C1             push  eax                     ; lpMsg                EAX=000CFEDC
00000C98  .text:WinMain(x,x,x,x)+C2             call  ds:__imp__TranslateMessage@4   ; TranslateMess...  ESP=000CFECC
00000C98  .text:WinMain(x,x,x,x)+C8             lea   eax, [ebp+Msg]                                 EAX=00000000
00000C98  .text:WinMain(x,x,x,x)+CB             push  eax                     ; lpMsg                EAX=000CFEDC
00000C98  .text:WinMain(x,x,x,x)+CC             call  ds:__imp__DispatchMessageW@4   ; DispatchMess...   ESP=000CFECC
00000C98  .text:WinMain(x,x,x,x)+D2             jmp   short loc_1001481                              EAX=00000000
00000C98  .text:WinMain(x,x,x,x):loc_1001481    push  esi                     ; wMsgFilterMax
00000C98  .text:WinMain(x,x,x,x)+7D             push  esi                     ; wMsgFilterMin        ESP=000CFECC
00000C98  .text:WinMain(x,x,x,x)+7E             push  esi                     ; hWnd                 ESP=000CFEC8
00000C98  .text:WinMain(x,x,x,x)+7F             lea   eax, [ebp+Msg]                                 ESP=000CFEC4
00000C98  .text:WinMain(x,x,x,x)+82             push  eax                     ; lpMsg                EAX=000CFEDC
00000C98  .text:WinMain(x,x,x,x)+83             call  edi ; GetMessageW(x,x,x,x)   ; GetMessageW(x,...   ESP=000CFEC0
00000C98  .text:NPWndProc(x,x,x,x)              mov   edi, edi                                       EAX=0100140E
00000C98  .text:NPWndProc(x,x,x,x)+2            push  ebp
Line 1 of 39277
```

Joe Sandbox

- Commercial product from Joe Security
- Can produce execution graphs
- Claims to have similar plugin
- Never used personally
- Seeing that they were using API traces gave inspiration to look into doing similar with Cuckoo
- Opted to not attempt to find code so my plugin would be "clean"

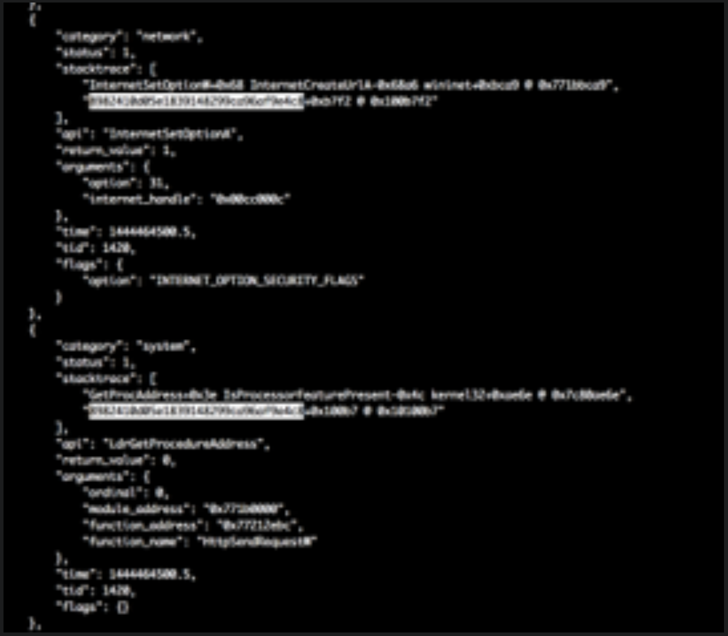
Background Material

Malware Analysis Challenges

- Packers / Crypters
- Self-Modifying Code
- Process / DLL Injection
- DLL Side-loading

Cuckoo Sandbox

- Popular open-source and free sandbox
- Injects monitor DLL into malicious process, logs API calls
- Cuckoo 2.0 currently in RC stage
  - HTTPS Decryption
  - Debug stacktrace available



```
},
{
    "category": "network",
    "status": 1,
    "stacktrace": [
        "InternetSetOptionW+0x68 InternetCreateUrlA-0x68a6 wininet+0xbca9 @ 0x771bbca9",
        "[illegible]+0xb7f2 @ 0x100b7f2"
    ],
    "api": "InternetSetOptionA",
    "return_value": 1,
    "arguments": {
        "option": 31,
        "internet_handle": "0x00cc000c"
    },
    "time": 1444464500.5,
    "tid": 1420,
    "flags": {
        "option": "INTERNET_OPTION_SECURITY_FLAGS"
    }
},
{
    "category": "system",
    "status": 1,
    "stacktrace": [
        "GetProcAddress+0x3e IsProcessorFeaturePresent-0x4c kernel32+0xae6e @ 0x7c80ae6e",
        "[illegible]+0x100b7 @ 0x10100b7"
    ],
    "api": "LdrGetProcedureAddress",
    "return_value": 0,
    "arguments": {
        "ordinal": 0,
        "module_address": "0x771b0000",
        "function_address": "0x77212ebc",
        "function_name": "HttpSendRequestW"
    },
    "time": 1444464500.5,
    "tid": 1420,
    "flags": {}
},
```

ARBOR NETWORKS

Memory Dumping

- Using the debug stacktrace in Cuckoo 2.0 can
  - Build a list of executed addresses
  - Use procdump to get base executable dumped
  - Attempt to retrieve memory pages containing addresses from the ramdump
  - Also use malfind from Volatility to locate other pages possibly undetected
- Using extra memory regions can then append extra sections onto base executable dump
  - Appending segment in IDA is non-ideal, IDA auto-analysis falls down in a few places

TACO

TACO Overview

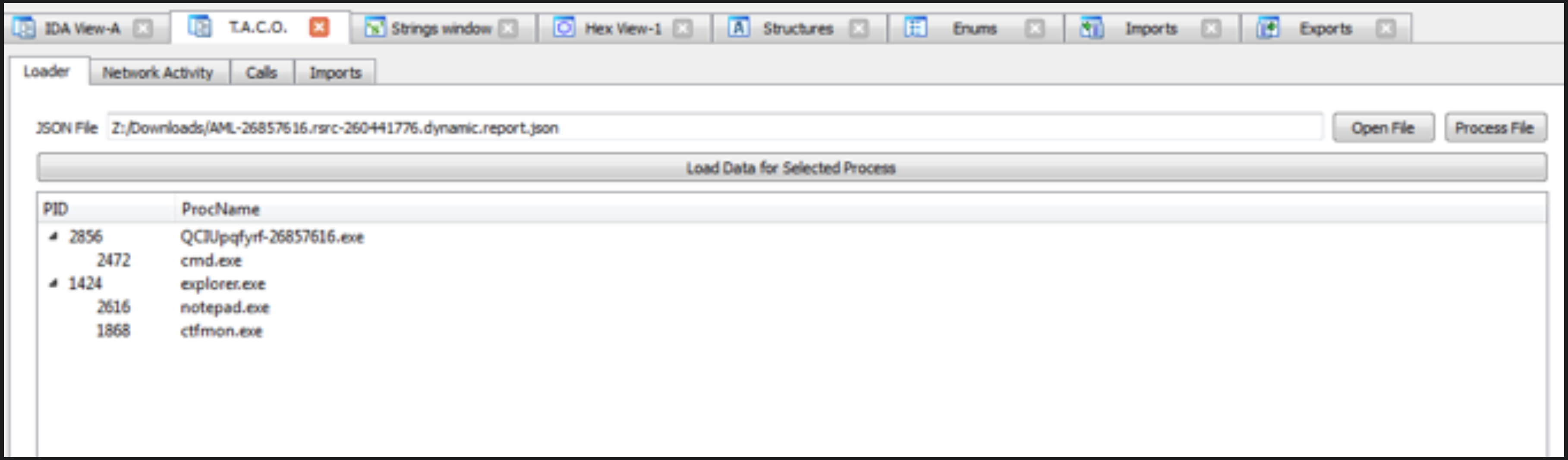
- Started out as dynamically generated IDAPython scripts
    - Clunky, prevented from doing "cool" things
    - Dynamically generating "clean" IDAPython is hard
- Some features incompatible with Cuckoo 1.2 due to lack of call metadata
- Cuckoo-Modified and current Cuckoo 2.0-dev branch supported supported for markup
- Idea sprung out of Joe Security's posts about execution graphs and seeing they imported analysis info into IDA
- Prior usage of tools like funcap and IDA's pintracer

TACO

- Consists of Cuckoo-based tabs for showing:
  - Processes
  - API Calls
  - Signatures
  - Imports
- Also includes other IDAPython scripts I have developed
  - Byte / Stack String viewer
  - "Interesting" XOR locator
  - Switch Jump / Case statement viewer

Loader Tab

- Main location to show a process tree and allow for specific processes to be inspected

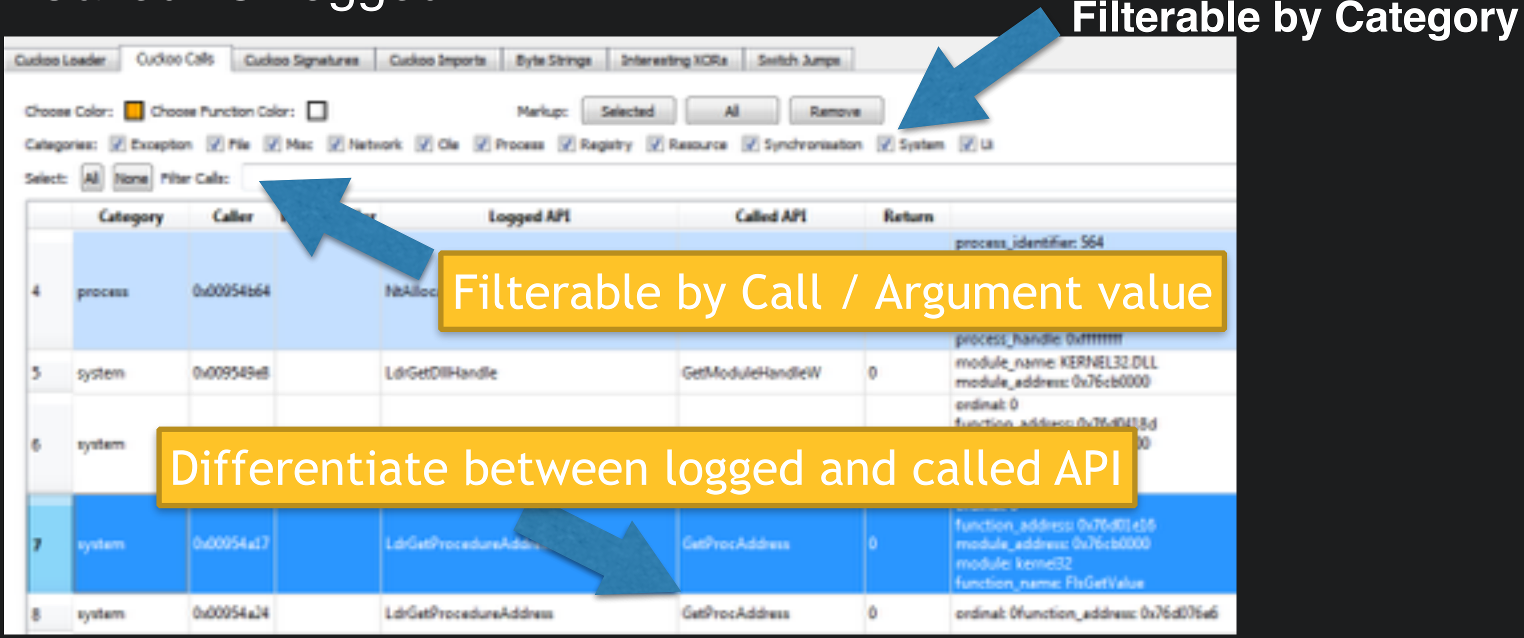


JSON File Z:/Downloads/AML-26857616.rsrc-260441776.dynamic.report.json

```
PID         ProcName
◢ 2856      QCIUpqfyrf-26857616.exe
    2472    cmd.exe
◢ 1424      explorer.exe
    2616    notepad.exe
    1868    ctfmon.exe
```

API Calls Tab

- Reproduction of Cuckoo's Output
- Filterable / Searchable / Clickable
- Detect Called vs Logged API



Choose Color: 🟧 Choose Function Color: ☐ Markup:

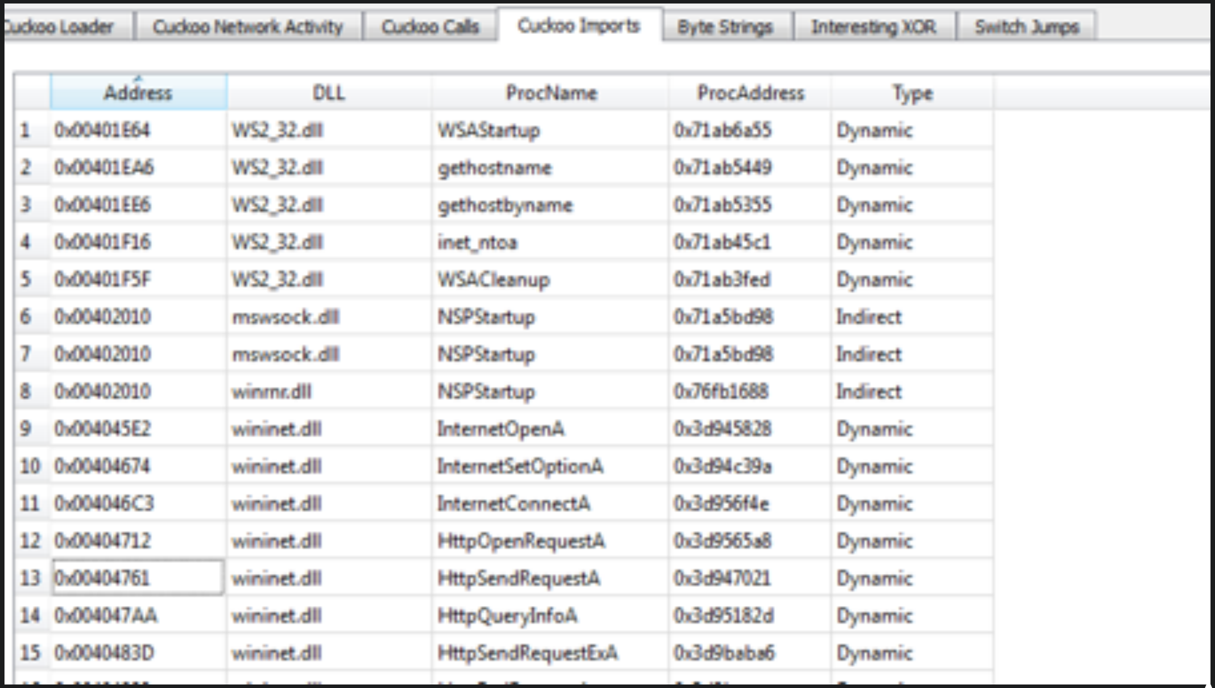
Categories: ✓ Exception ✓ File ✓ Misc ✓ Network ✓ Ole ✓ Process ✓ Registry ✓ Resource ✓ Synchronisation ✓ System ✓ UI

Select: Filter Calls:

	Category	Caller	[illegible]	Logged API	Called API	Return	
4	process	0x00954b64		NtAlloc[illegible]	[illegible]	[illegible]	process_identifier: 564 [illegible] process_handle: 0xffffffff
5	system	0x009549e8		LdrGetDllHandle	GetModuleHandleW	0	module_name: KERNEL32.DLL module_address: 0x76cb0000
6	system	[illegible]		[illegible]	[illegible]	[illegible]	ordinal: 0 function_address: 0x76d0418d [illegible]
7	system	0x00954a17		LdrGetProcedureAddr[illegible]	GetProcAddress	0	[illegible] function_address: 0x76d01e16 module_address: 0x76cb0000 module: kernel32 function_name: FlsGetValue
8	system	0x00954a24		LdrGetProcedureAddress	GetProcAddress	0	ordinal: 0function_address: 0x76d076a6

Imports Tab

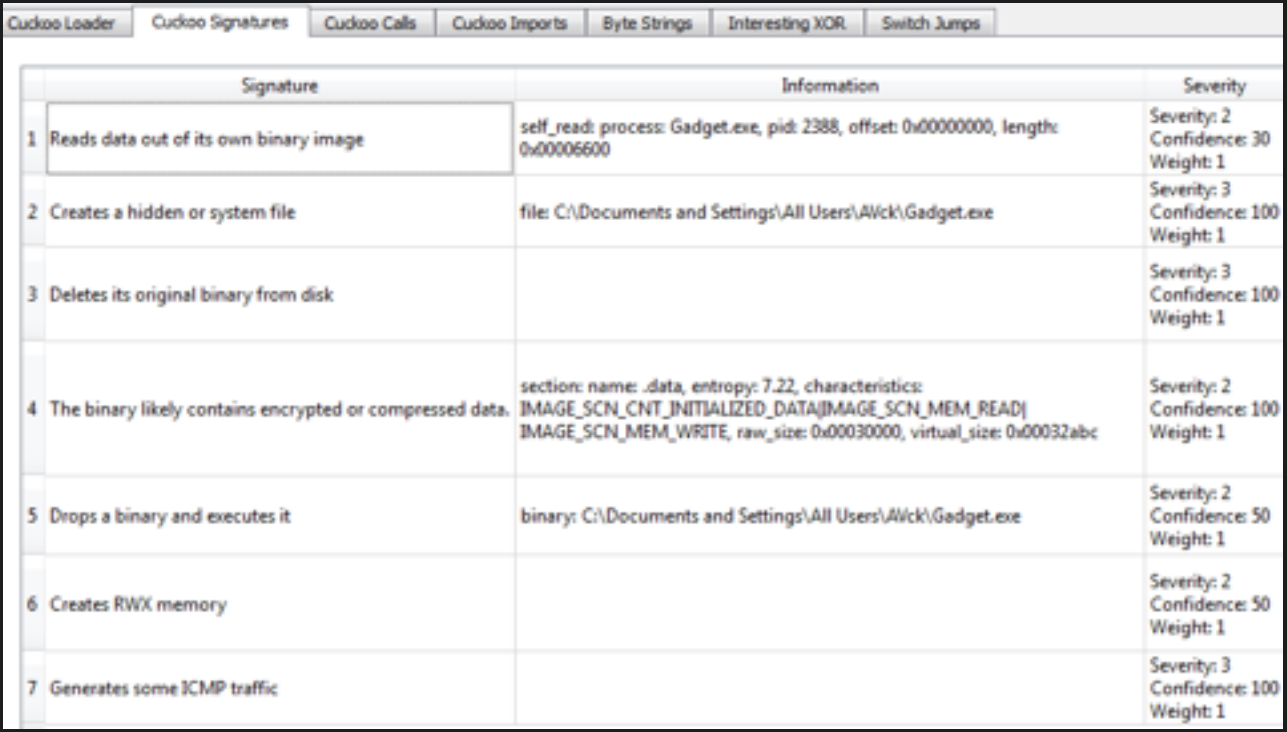
- Tries to detect dynamic imports via direct / indirect calls
- Can rename addresses of detected imports



	Address	DLL	ProcName	ProcAddress	Type
1	0x00401E64	WS2_32.dll	WSAStartup	0x71ab6a55	Dynamic
2	0x00401EA6	WS2_32.dll	gethostname	0x71ab5449	Dynamic
3	0x00401EE6	WS2_32.dll	gethostbyname	0x71ab5355	Dynamic
4	0x00401F16	WS2_32.dll	inet_ntoa	0x71ab45c1	Dynamic
5	0x00401F5F	WS2_32.dll	WSACleanup	0x71ab3fed	Dynamic
6	0x00402010	mswsock.dll	NSPStartup	0x71a5bd98	Indirect
7	0x00402010	mswsock.dll	NSPStartup	0x71a5bd98	Indirect
8	0x00402010	winrnr.dll	NSPStartup	0x76fb1688	Indirect
9	0x004045E2	wininet.dll	InternetOpenA	0x3d945828	Dynamic
10	0x00404674	wininet.dll	InternetSetOptionA	0x3d94c39a	Dynamic
11	0x004046C3	wininet.dll	InternetConnectA	0x3d956f4e	Dynamic
12	0x00404712	wininet.dll	HttpOpenRequestA	0x3d9565a8	Dynamic
13	0x00404761	wininet.dll	HttpSendRequestA	0x3d947021	Dynamic
14	0x004047AA	wininet.dll	HttpQueryInfoA	0x3d95182d	Dynamic
15	0x0040483D	wininet.dll	HttpSendRequestExA	0x3d9baba6	Dynamic

Signatures Tab

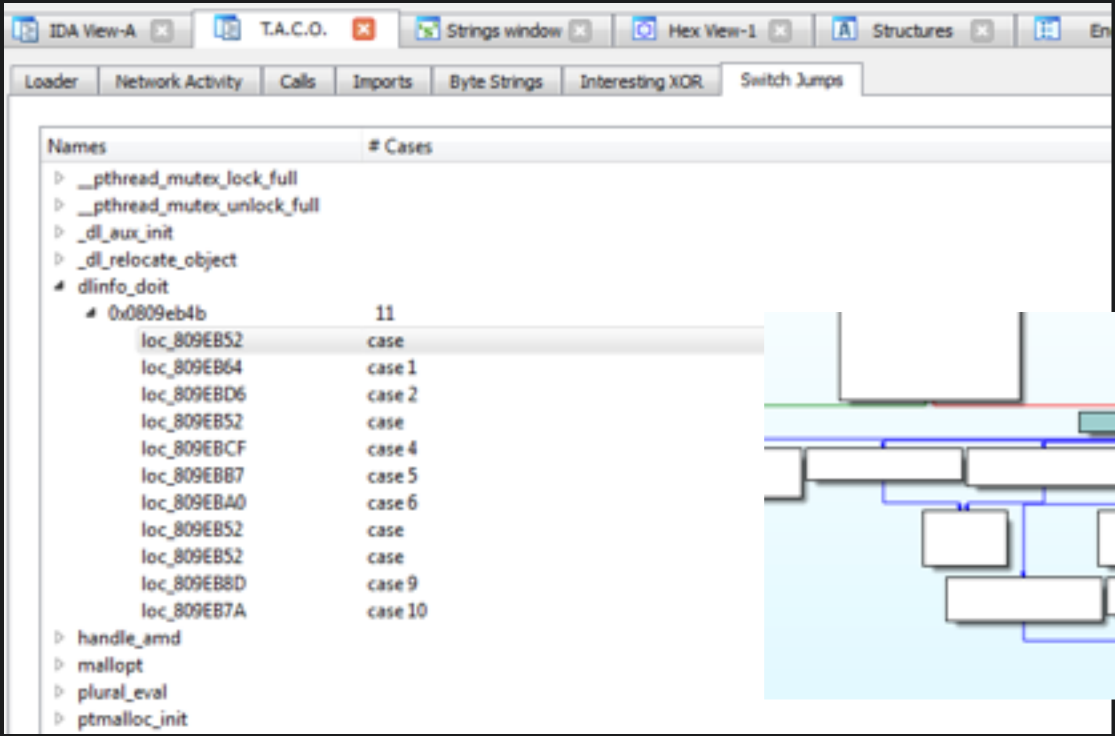
- Simple Display of Cuckoo Triggered Signatures



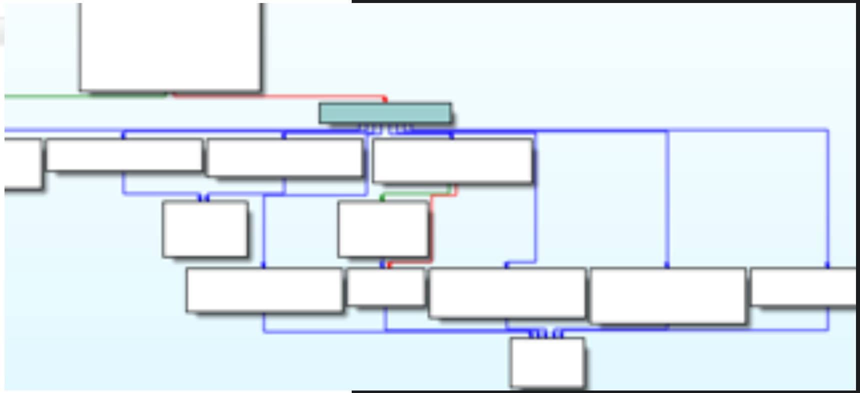
	Signature	Information	Severity
1	Reads data out of its own binary image	self_read: process: Gadget.exe, pid: 2388, offset: 0x00000000, length: 0x00006600	Severity: 2 Confidence: 30 Weight: 1
2	Creates a hidden or system file	file: C:\Documents and Settings\All Users\AVck\Gadget.exe	Severity: 3 Confidence: 100 Weight: 1
3	Deletes its original binary from disk		Severity: 3 Confidence: 100 Weight: 1
4	The binary likely contains encrypted or compressed data.	section: name: .data, entropy: 7.22, characteristics: IMAGE_SCN_CNT_INITIALIZED_DATA|IMAGE_SCN_MEM_READ| IMAGE_SCN_MEM_WRITE, raw_size: 0x00030000, virtual_size: 0x00032abc	Severity: 2 Confidence: 100 Weight: 1
5	Drops a binary and executes it	binary: C:\Documents and Settings\All Users\AVck\Gadget.exe	Severity: 2 Confidence: 50 Weight: 1
6	Creates RWX memory		Severity: 2 Confidence: 50 Weight: 1
7	Generates some ICMP traffic		Severity: 3 Confidence: 100 Weight: 1

Switch Jump Viewer

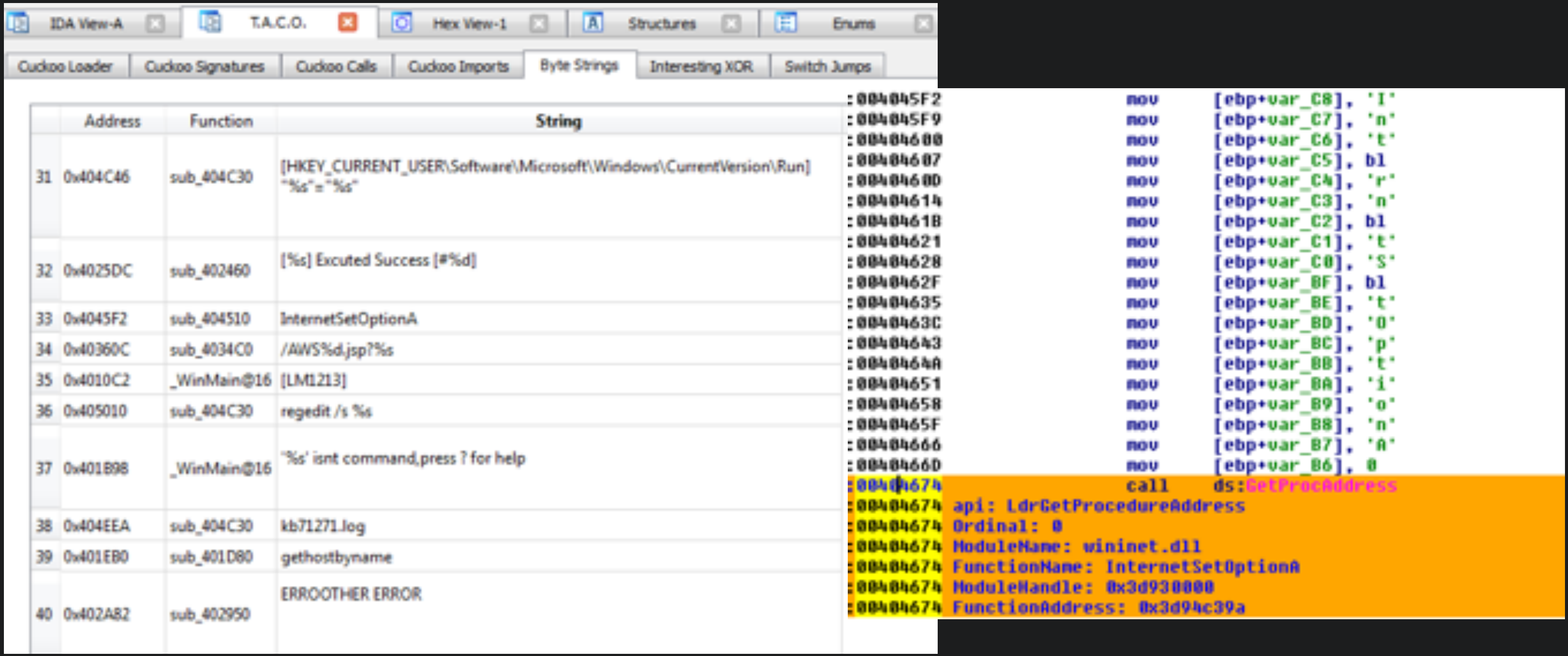
- Switch jumps in malware can indicate config or cmd parsing



```
Names                          # Cases
▷ __pthread_mutex_lock_full
▷ __pthread_mutex_unlock_full
▷ _dl_aux_init
▷ _dl_relocate_object
◢ dlinfo_doit
    ◢ 0x0809eb4b               11
        loc_809EB52            case
        loc_809EB64            case 1
        loc_809EBD6            case 2
        loc_809EB52            case
        loc_809EBCF            case 4
        loc_809EBB7            case 5
        loc_809EBA0            case 6
        loc_809EB52            case
        loc_809EB52            case
        loc_809EB8D            case 9
        loc_809EB7A            case 10
▷ handle_cmd
▷ mallopt
▷ plural_eval
▷ ptmalloc_init
```



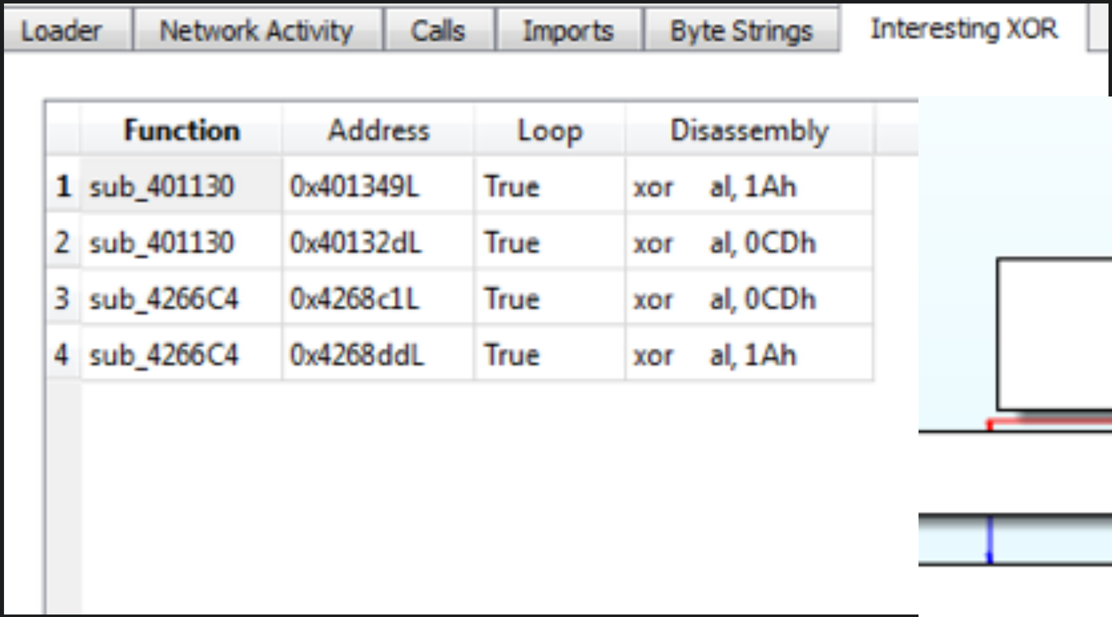
Stack String Locator



	Address	Function	String
31	0x404C46	sub_404C30	[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run] "%s"="%s"
32	0x4025DC	sub_402460	[%s] Excuted Success [#%d]
33	0x4045F2	sub_404510	InternetSetOptionA
34	0x40360C	sub_4034C0	/AWS%d.jsp?%s
35	0x4010C2	_WinMain@16	[LM1213]
36	0x405010	sub_404C30	regedit /s %s
37	0x401B98	_WinMain@16	'%s' isnt command,press ? for help
38	0x404EEA	sub_404C30	kb71271.log
39	0x401EB0	sub_401D80	gethostbyname
40	0x402A82	sub_402950	ERROOTHER ERROR

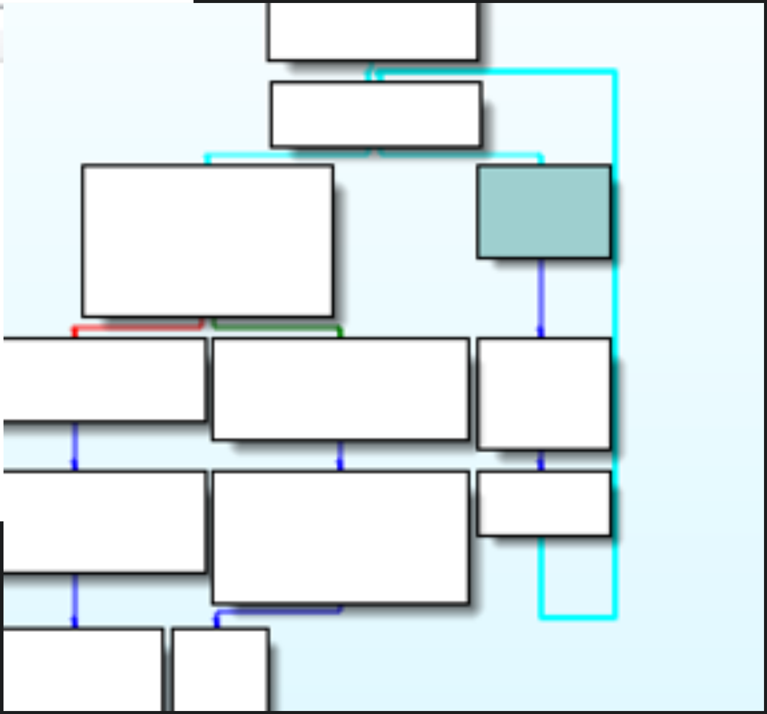
```
:004045F2                mov     [ebp+var_C8], 'I'
:004045F9                mov     [ebp+var_C7], 'n'
:00404600                mov     [ebp+var_C6], 't'
:00404607                mov     [ebp+var_C5], bl
:0040460D                mov     [ebp+var_C4], 'r'
:00404614                mov     [ebp+var_C3], 'n'
:0040461B                mov     [ebp+var_C2], bl
:00404621                mov     [ebp+var_C1], 't'
:00404628                mov     [ebp+var_C0], 'S'
:0040462F                mov     [ebp+var_BF], bl
:00404635                mov     [ebp+var_BE], 't'
:0040463C                mov     [ebp+var_BD], 'O'
:00404643                mov     [ebp+var_BC], 'p'
:0040464A                mov     [ebp+var_BB], 't'
:00404651                mov     [ebp+var_BA], 'i'
:00404658                mov     [ebp+var_B9], 'o'
:0040465F                mov     [ebp+var_B8], 'n'
:00404666                mov     [ebp+var_B7], 'A'
:0040466D                mov     [ebp+var_B6], 0
:00404674                call    ds:GetProcAddress
:00404674 api: LdrGetProcedureAddress
:00404674 Ordinal: 0
:00404674 ModuleName: wininet.dll
:00404674 FunctionName: InternetSetOptionA
:00404674 ModuleHandle: 0x3d930000
:00404674 FunctionAddress: 0x3d94c39a
```

"Interesting" XOR Tab



Loader | Network Activity | Calls | Imports | Byte Strings | Interesting XOR

	Function	Address	Loop	Disassembly
1	sub_401130	0x401349L	True	xor al, 1Ah
2	sub_401130	0x40132dL	True	xor al, 0CDh
3	sub_4266C4	0x4268c1L	True	xor al, 0CDh
4	sub_4266C4	0x4268ddL	True	xor al, 1Ah



DEMO

Fin

- https://github.com/arbor-jjones/idataco