

---

# Semantic Binary Exploration

Speeding up malware analysis

Laura Guevara and Daniel Plohmann

[laura.guevara@fkie.fraunhofer.de](mailto:laura.guevara@fkie.fraunhofer.de)

[daniel.plohmann@fkie.fraunhofer.de](mailto:daniel.plohmann@fkie.fraunhofer.de)

Dec. 3<sup>rd</sup> 2014, Nancy



# Outline for this Talk

- Motivation
- Preface: Behaviour Analysis
- Semantics Exploration
  - Malware Semantics
  - Methodology
  - Algorithm
- Demo
  - Introduction to IDAScope
  - Semantic Explorer vs. Citadel
- Conclusion

# Motivation

## Why build a scanner for semantic exploration

- Experiences of daily work:
  - „What are the capabilities of this unknown executable?“
- Observation: Different malware samples share many common aspects of malicious functionality
  - Evolution of version within one family: minor modification or changed appearance through compiler fragmentation
  - Authors seem to have copy-cat mentality regarding snippets available on the Internet

# Static Analysis

- Decoupling analysis from the malware's execution time
  - Access all of the code (also „dormant“ parts)
  - Allows exploration and documentation at the same time
  
- Automated tool
  - Explore the control flow graph of executable Windows memory images
  - Support the analyst during static analysis
  - Guidance to specific regions of interest

# Our Approach

- We examine sequences of calls to API functions in malware instances and try to infer the user-level functionality connected to them.

# Malware Behaviour Patterns

---

# Malware Features

- Equivalent features reappear from one malware variant to another
  - Shaped differently in code
  - But with a predictable occurrence of used API functions
- Abstracting the interaction of malware with the Operating System
  - Syntax vs. Semantics
- Platform dependent
  - Windows Subsystem DLLs

Windows API

# Windows Application Programming Interface

---



# Windows API

- Formerly called Win32 API
- Specifies a collection of services needed during runtime
- Usually loaded before or during the actual execution
- A common way to analyze the behaviour of programs is by inspecting its calls to API functions

## Abstracting Behaviour

# Malware Semantics

---

# Semantics

## „Behaviour Profile“

- Assign meaning to the set of common malware operations
  - Copying or deleting files for hidden persistence
  - Injecting into processes for more control or concealment
  - Communicating over the network, etc.
- These are usually implemented using calls to a specific collection of Windows API subroutines.

# Malware Semantics

# Example

---

# Process Injection

## Step 1. Iteration over Processes



## Step 2. Process injection



Overview

# Semantic Binary Exploration

---

# Import Table Directory

- Our tool requires availability of API information (e.g. restored import tables) since resolved call destinations are cataloged and examined

# Unpacked Binaries

- Applicable to files in Windows Portable Executable (PE) format and shellcode



# Methodology

- Collection of Malware Behaviour
- Definition of malware semantics
- Exploration:
  - Extraction of flow components
  - Call Graph construction
  - Matching of specifications
  - Semantic Traces:
  - Cross-evaluation with HCA

# Semantic Explorer

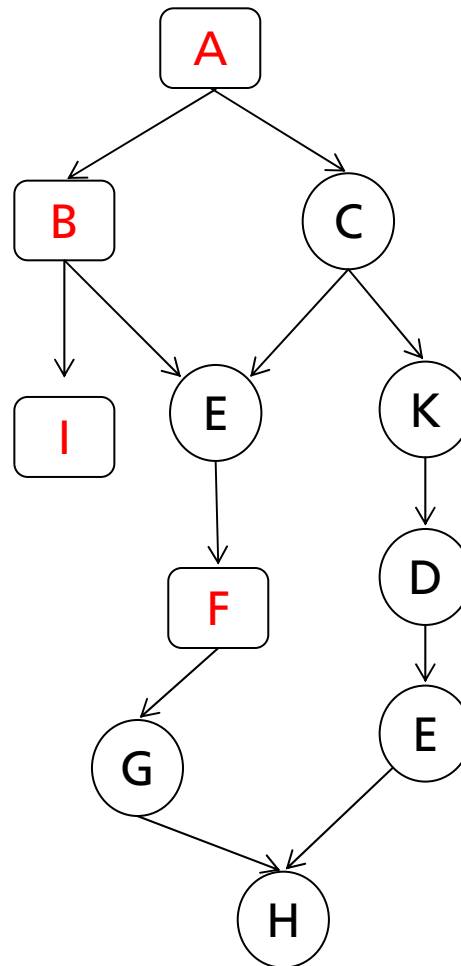
# Argument Parsing

- Backtrace reference of registers
- String Parsing
  - Based on Alexander Hanel's work

Difficulty:

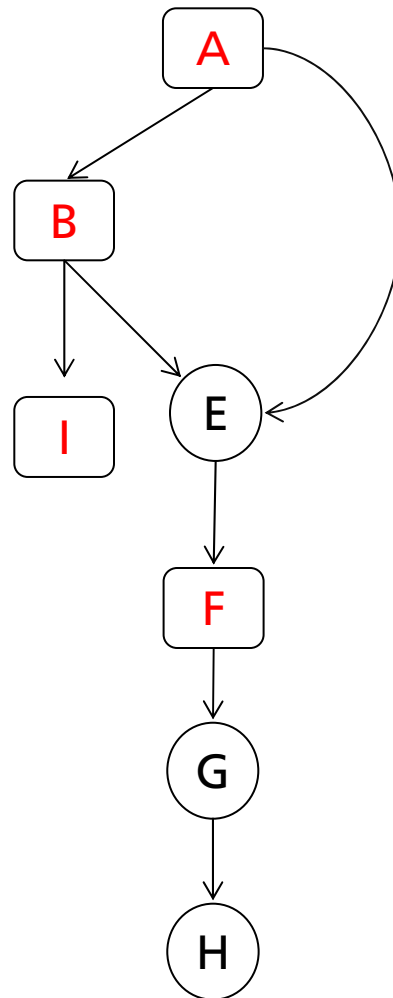
- Address every possible scenario w.r.t how the data is moved
  - Cross-reference
  - Function return values

# Data Reduction Strategies



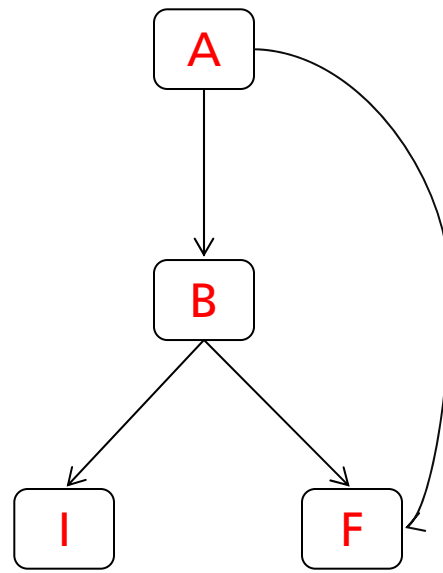
{ A, B, I, F }

# Data Reduction Strategies



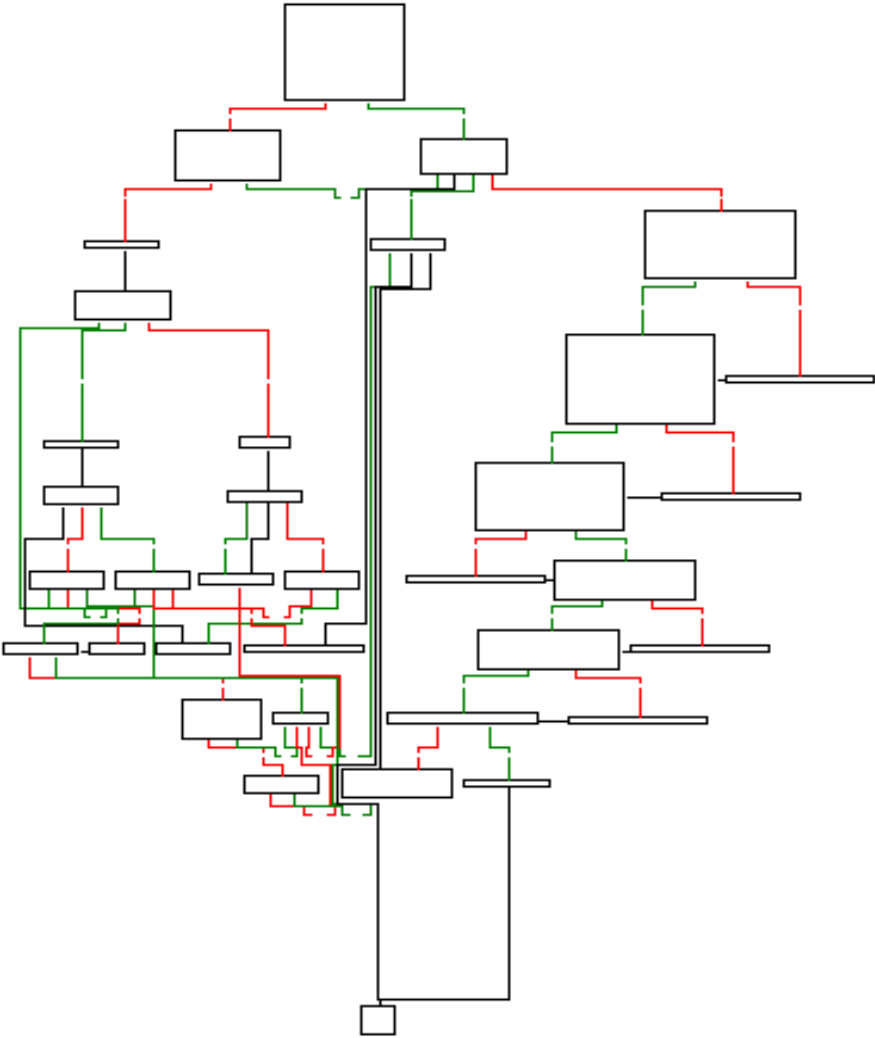
{ A, B, I, F }

# Data Reduction Strategies

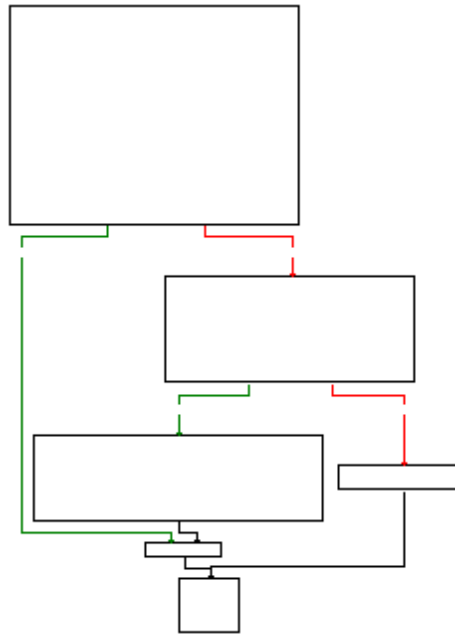


{ A, B, I, F }

# Data Reduction Strategies

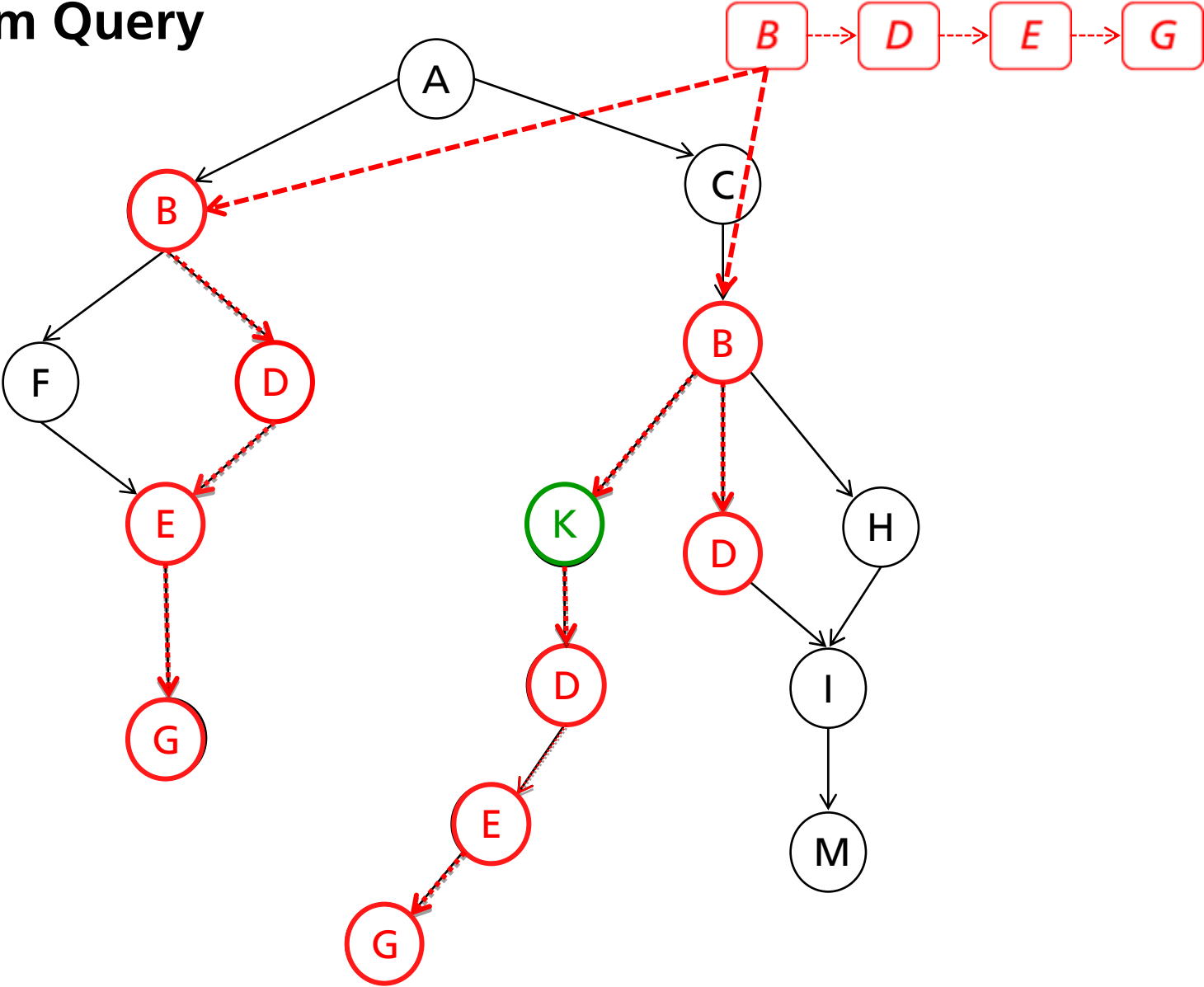


# Data Reduction Strategies

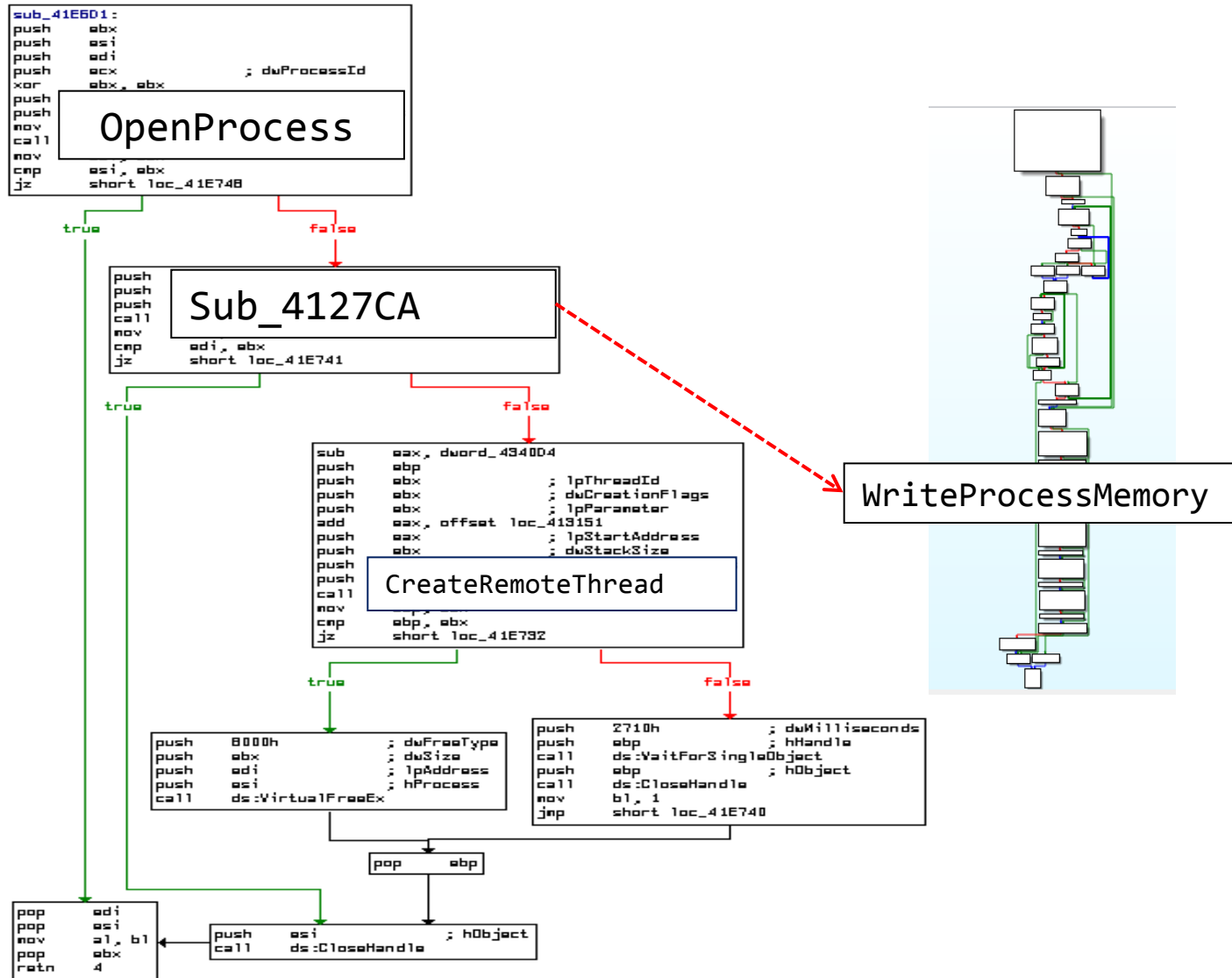




# N-Gram Query



# Nested Functionality and Control Flow Integrity



The following slides reflect the content of the

# Demo

---

# IDAscope

## An IDA Pro extension to aid malware reverse engineering


- Motivated by the current typical workflow of working with IDA Pro.
  - Repeat: „Identify relevant parts of the binary; tear apart; document findings.“
- Common tasks:
  - Work corner pieces: strings, API calls, signature hits, ...
  - Reoccurring need for looking up things in MSDN (switch windows...)
  - C&C communication schemes are of high interest!
  - Find and understand cryptographic routines used.
- Idea:
  - Provide automation/integration of „helpers“ that assist with regularly performed tasks.

Hex-Rays Home > Plug-In Contest

## Hex-Rays

### Plug-In Contest 2012: Hall Of Fame

Contests 2012 2011 2010 2009



This year the plugin contest gathered five contestants. But as you know, there can only be ~~one~~, well, two winners!

Based on the plugin's functionality, robustness, usefulness, ease of use and documentation, we declare the following winners:

1. Aaron Portnoy, of Exodus Intelligence, with the [IDA Toolbag plugin](#)
2. Daniel Plohmann, of the Fraunhofer FKIE, with the [IDAscope plugin](#)

Congratulations to both! We are pleased with the improved plugin quality and complexity.

Below is the list of all submissions in no particular order. All contest entries are interesting and useful:

# IDAscope

## Seamless integration

- IDAScope is directly integrated as widget in the IDA interface

The screenshot displays the IDA Pro interface with the IDAScope widget integrated into the right-hand pane. The main window shows the assembly code for function `sub_4248A2`, which is a standard Windows API call wrapper. The code includes variable declarations for `lpMsg` and stack frame setup. A call instruction `call sub_412966` is highlighted, and the IDAScope widget on the right shows the semantic analysis results for this call, indicating that no semantics were matched.

```
; Attributes: bp-based frame
; int_stdcall sub_4248A2(MSG *lpMsg)
sub_4248A2 proc near

var_374= dword ptr -374h
pwszBuff= word ptr -370h
String1= word ptr -35Ch
var_344= byte ptr -344h
KeyState= byte ptr -308h
var_208= byte ptr -208h
lpMsg= dword ptr 8

push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF8h
sub     esp, 374h
push    ebx
mov     ebx, [ebp+lpMsg]
push    esi
push    edi
test   ebx, ebx
jz      loc_424A08

call   sub_412966
test   al, al
jz      loc_424A08

mov     eax, [ebx+4]
cmp    eax, 201h
jnz    loc_424A97

loc_424A08:

```

IDA View-A: `call sub_412966`  
test al, al  
jz loc\_424A08

IDA View-A: `mov eax, [ebx+4]`  
cmp eax, 201h  
jnz loc\_424A97

simplyFIRE.IDAScope v1.2

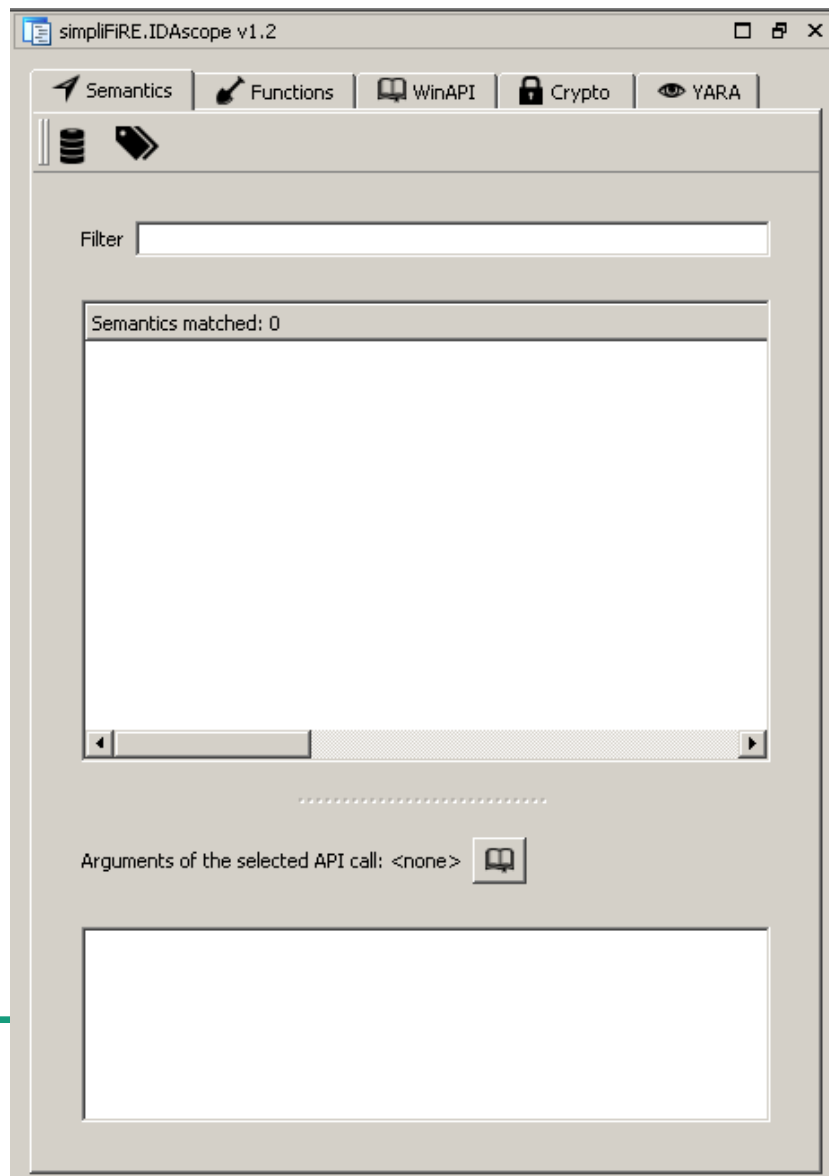
Semantics matched: 0

Arguments of the selected API call: <none>

# IDAscope

... has grown since ist original release (+YARA and Semantic Explorer)

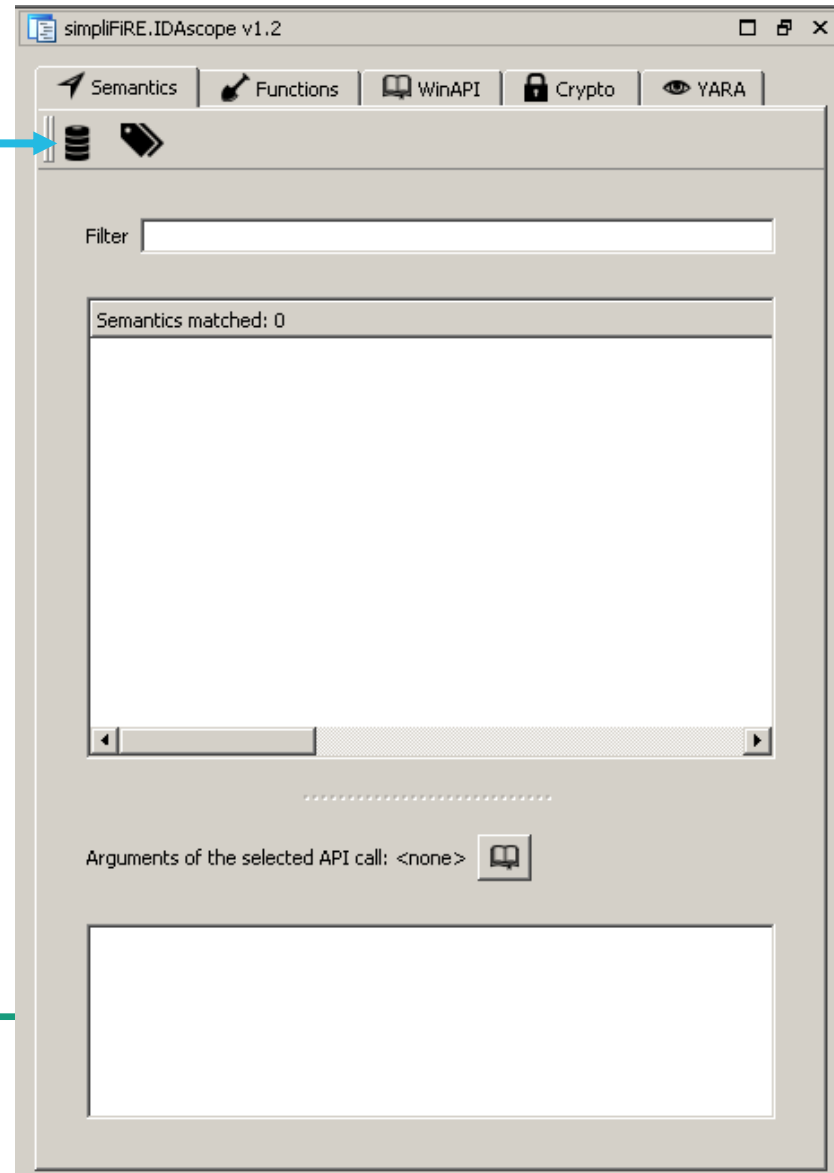
- Tabs for different functions:
  - Semantic Explorer
    - as presented in this talk
  - Function Inspection
    - predecessor for the work presented in this talk
  - MSDN Browsing (WinAPI)
    - seamless lookup of function signatures, enums, ...
  - Cryptography Scanner
    - heuristically over instruction type frequency (arithmetic & logic vs. other)
    - signatures for common algorithms
  - YARA Scanner
    - shows incomplete matches, which is useful when writing signatures



# IDAscope

## Semantic Explorer usage examples (scanning)

- Clicking the DB symbol will initiate semantic matching



# IDAscope

## Semantic Explorer usage examples (result display)

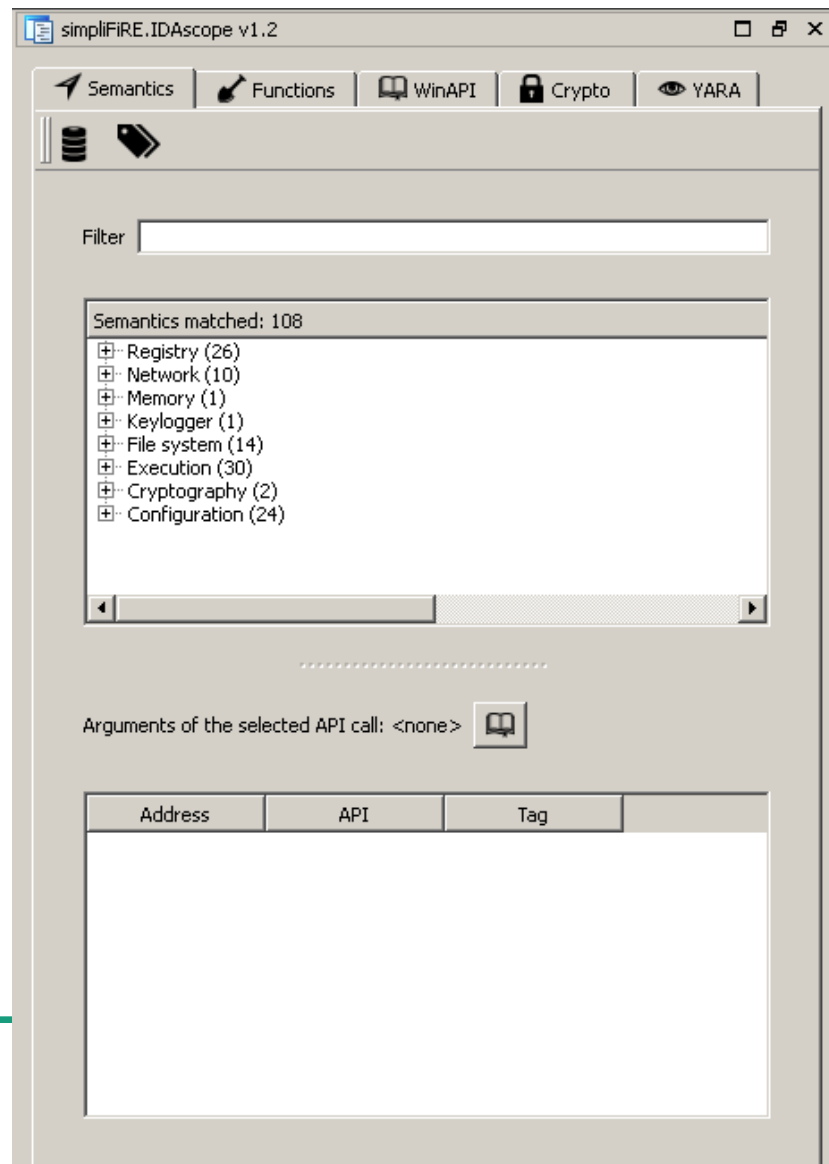
- In this example, we have 108 occurrences of matched semantics
- Organized in thematic groups
- These can be freely defined in the config file
- Scanning takes around 20 seconds (sample with 750+ functions)

```
Output window
Building data structures...
Calculating control flow... done.
Pruning flow graph... done.
completed after 5.50 seconds.

Matching Semantics...

Full analysis completed in 16.14 seconds.
```

Python





# IDAscope

## Semantic Explorer usage examples (expanded results)

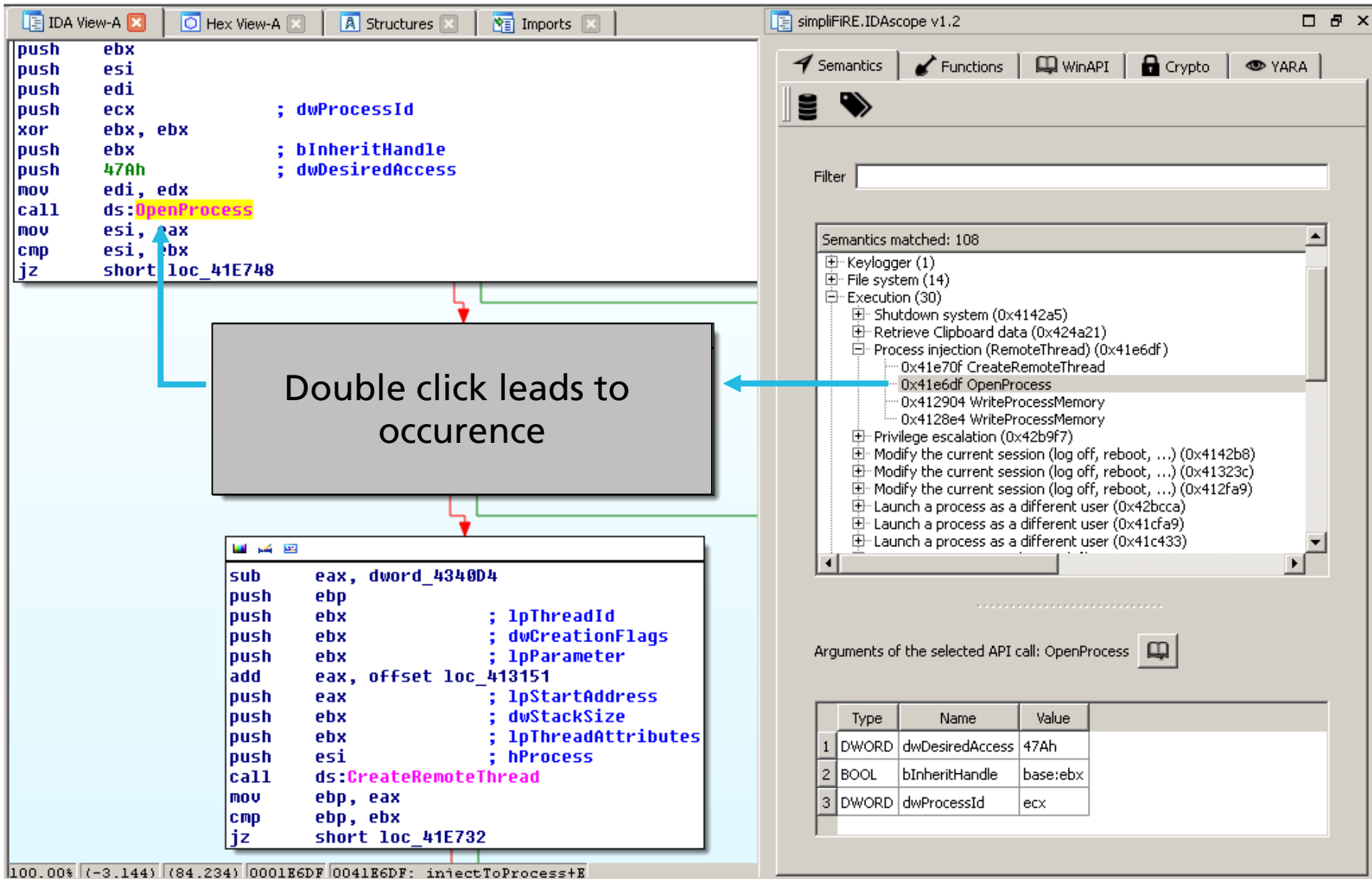
- Opening one of the groups shows all the semantics contained in the group
- Opening a semantic match shows the suspicious API call sequence
- Selecting an API call shows its arguments

The screenshot shows the IDA Scope Semantic Explorer interface. The top toolbar includes buttons for Semantics, Functions, WinAPI, Crypto, and YARA. A filter box is present above the main list. The main list displays 'Semantics matched: 108' and is expanded to show a tree structure. The 'Execution (30)' group is expanded, and the 'OpenProcess' entry is selected. Below the list, the 'Arguments of the selected API call: OpenProcess' section is visible, containing a table with the following data:

|   | Type  | Name            | Value    |
|---|-------|-----------------|----------|
| 1 | DWORD | dwDesiredAccess | 47Ah     |
| 2 | BOOL  | bInheritHandle  | base:ebx |
| 3 | DWORD | dwProcessId     | ecx      |

# IDAscope

## Semantic Explorer usage examples (interactive link to IDA navigation)



Double click leads to occurrence

```
push ebx
push esi
push edi
push ecx ; dwProcessId
xor ebx, ebx
push ebx ; bInheritHandle
push 47Ah ; dwDesiredAccess
mov edi, edx
call ds:OpenProcess
mov esi, eax
cmp esi, ebx
jz short loc_41E748
```

```
sub eax, dword_4340D4
push ebp
push ebx ; lpThreadId
push ebx ; dwCreationFlags
push ebx ; lpParameter
add eax, offset loc_413151
push eax ; lpStartAddress
push ebx ; dwStackSize
push ebx ; lpThreadAttributes
push esi ; hProcess
call ds:CreateRemoteThread
mov ebp, eax
cmp ebp, ebx
jz short loc_41E732
```

Semantics matched: 108

- Keylogger (1)
- File system (14)
- Execution (30)
  - Shutdown system (0x4142a5)
  - Retrieve Clipboard data (0x424a21)
  - Process injection (RemoteThread) (0x41e6df)
    - 0x41e70f CreateRemoteThread
    - 0x41e6df OpenProcess
    - 0x412904 WriteProcessMemory
    - 0x4128e4 WriteProcessMemory
  - Privilege escalation (0x42b9f7)
  - Modify the current session (log off, reboot, ...) (0x4142b8)
  - Modify the current session (log off, reboot, ...) (0x41323c)
  - Modify the current session (log off, reboot, ...) (0x412fa9)
  - Launch a process as a different user (0x42bcc)
  - Launch a process as a different user (0x41cfa9)
  - Launch a process as a different user (0x41c433)

Arguments of the selected API call: OpenProcess

|   | Type  | Name            | Value    |
|---|-------|-----------------|----------|
| 1 | DWORD | dwDesiredAccess | 47Ah     |
| 2 | BOOL  | bInheritHandle  | base:ebx |
| 3 | DWORD | dwProcessId     | ecx      |

# IDAscope

## Semantic Explorer integration with other tabs (jump to API info)

- Clicking the “book” button opens the respective API information in WinAPI view (MSDN entry)

The screenshot shows the simpliFIRE.IDAScope v1.2 interface. The top navigation bar includes tabs for Semantics, Functions, WinAPI, Crypto, and YARA. The main area displays a list of semantics matched for the filter 'process'. The list is categorized into Memory (1), Keylogger (1), File system (14), and Execution (30). The Execution category is expanded to show 'Process injection (RemoteThread) (0x41e6df)', which includes sub-items like CreateRemoteThread, OpenProcess, WriteProcessMemory, and AdjustTokenPrivileges. A blue arrow points from the 'Process injection' entry to the 'OpenProcess' entry in the 'Arguments of the selected API' section. This section shows the arguments for the OpenProcess API, including the API name and a book icon for more information.

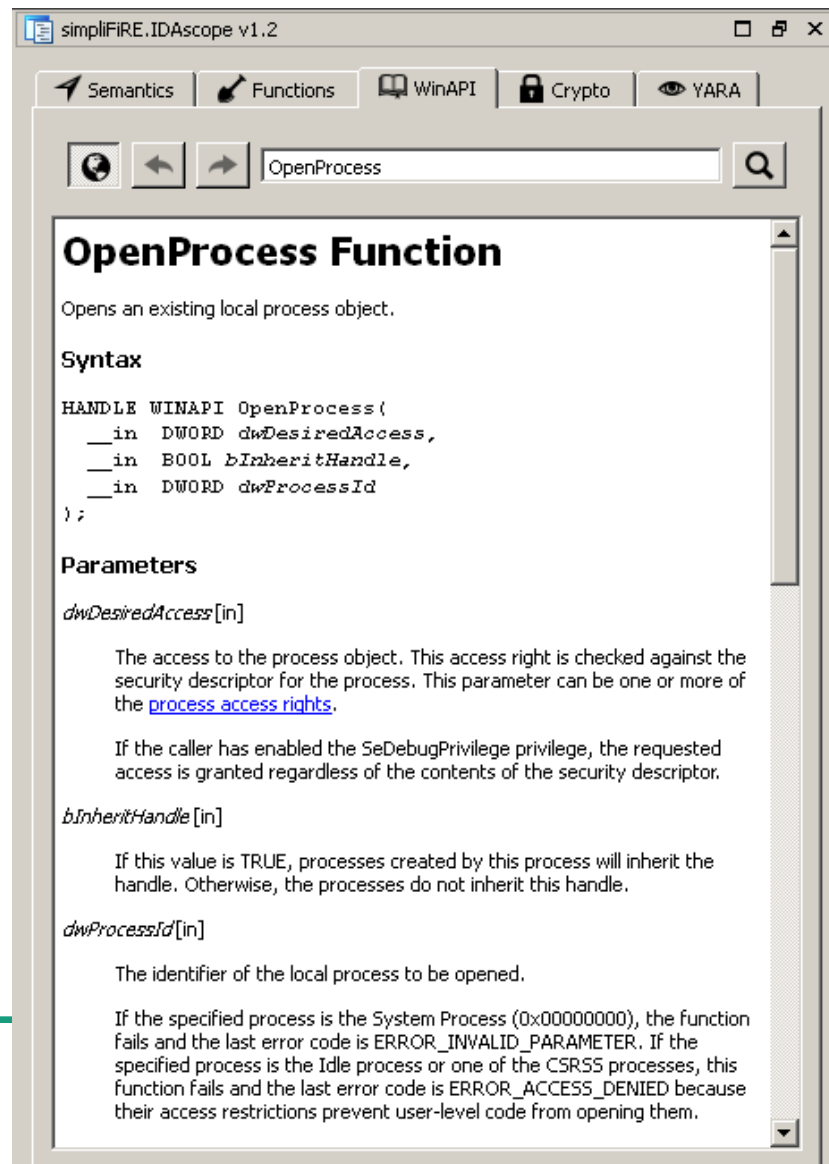
Arguments of the selected API: OpenProcess

|   | Type  | Name            | Value    |
|---|-------|-----------------|----------|
| 1 | DWORD | dwDesiredAccess | 47Ah     |
| 2 | BOOL  | bInheritHandle  | base:ebx |
| 3 | DWORD | dwProcessId     | ecx      |

# IDAscope

## Semantic Explorer integration with other tabs

- Clicking the “book” button opens the respective API information in WinAPI view (MSDN entry)



# IDAscope & Semantic Explorer

## Limitations & Outlook

- Semantic Explorer code release
  - Currently tied to IDA Pro -> support other frameworks (radare, ...?)
- Improvements to graph exploration / referencing
  - Duplicate reduction
  - Fix occasional recursions
- Improvements to backtracking / dataflow analysis
  - Infer more calculated / constant arguments
  - Resolving more enums
- Expansion of set of semantic signatures
  - Contributions welcome! :)
  - Adoption of MITRE MAEC standard?
- Export / rendering of results
  
- IDAscope repository:
  - [https://bitbucket.org/daniel\\_plohmann/simplifire.idascope](https://bitbucket.org/daniel_plohmann/simplifire.idascope)