Signature-Based Detection of Behavioural Malware Features with Windows API Calls

Simon Jansen

Master Thesis Defence Presentation

Examiner: Prof. Dr.-Ing. Felix Freiling
Advisor: Dr. Christian Gorecki

20th July 2020, Erlangen

Outline

1 Motivation

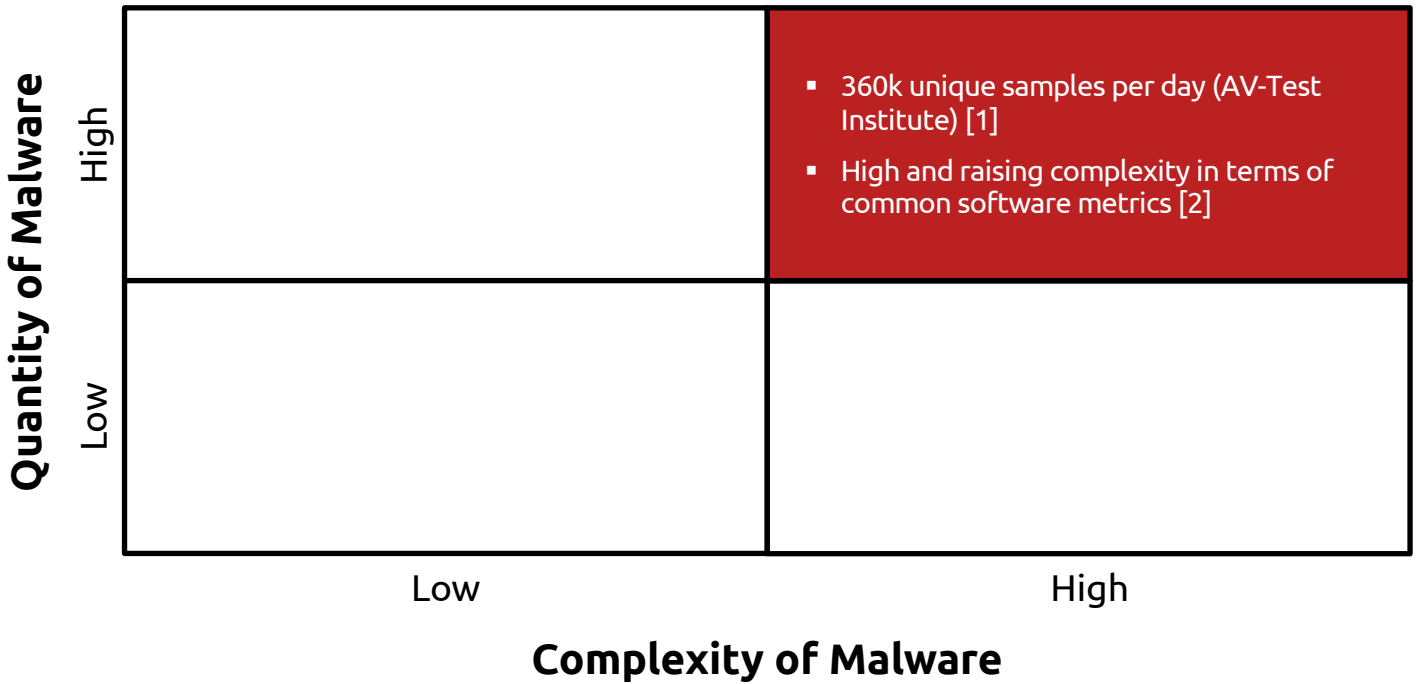
2 Windows API and VMRay Analyzer Fundamentals

3 dynmx Detection Approach

4 Evaluation Results

5 Conclusion and Future Work

Motivation



[1] https://www.av-test.org/en/statistics/malware/, [2] Calleja, Tapiador and Caballero. A Look into 30 Years of Malware Development from a Software Metrics Perspective, 2016.

Motivation

Research Question

Missing Extendability

Malware sandboxes are not extendable and flexible in terms of detected behaviour

Missing Behaviour Signatures

No practically relevant YARA equivalent for behaviour

API Call Usage

Malware extensively uses API calls to interact with the operating system

> *From the viewpoint of a **domain expert**, to what extent can the **signature-based detection of malicious runtime behaviour** based on **API function calls** which is derived from the **dynamic analysis of malware in sandboxes** contribute to the **improvement of the malware analysis process**?*

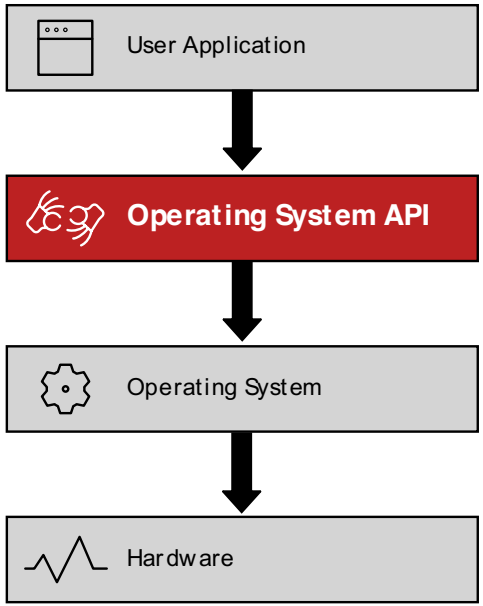
Contributions

- Introduction of a sandbox-agnostic domain specific language (DSL) to define known malicious behaviour on the API call level
- Development of a suitable algorithm to detect behaviour signatures in raw unprocessed information provided by the VMRay Analyzer sandbox
- Deduction of resource usage based on characteristic API call usage (based on work of Lanzi et al. on Access Activity Models)
- Introduction of a generic function log format to store function logs in a harmonised manner
- Prototype implementation and evaluation of the detection approach based on real-world malware samples

Fundamentals

Windows API

- Application programming interface (API) of Microsoft Windows
- Categorisation
  - WinAPI (userland API calls)
  - Native API (system calls)
  - Kernel (kernel routines)
- Comprehensive and complex (~4,800 WinAPI functions, ~400 system calls)



- Malware uses large set of API function calls (1,000 function logs contained ~2,400 different API calls)

VMRay Analyzer Sandbox

- Commercial Virtual Machine Introspection (VMI) based malware sandbox
- Functionality based on CXPInspector [1]
- Monitors malware activity from the hypervisor
- Provides analysis report and raw source information of the analysis process

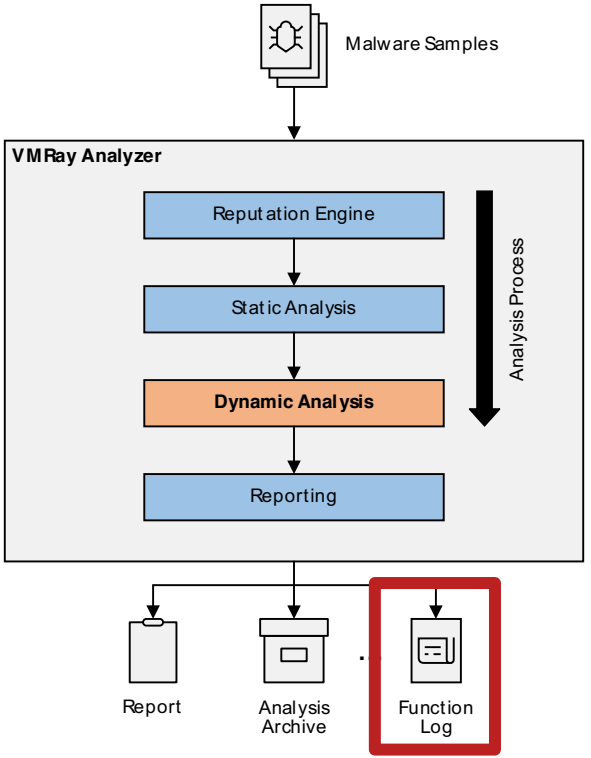
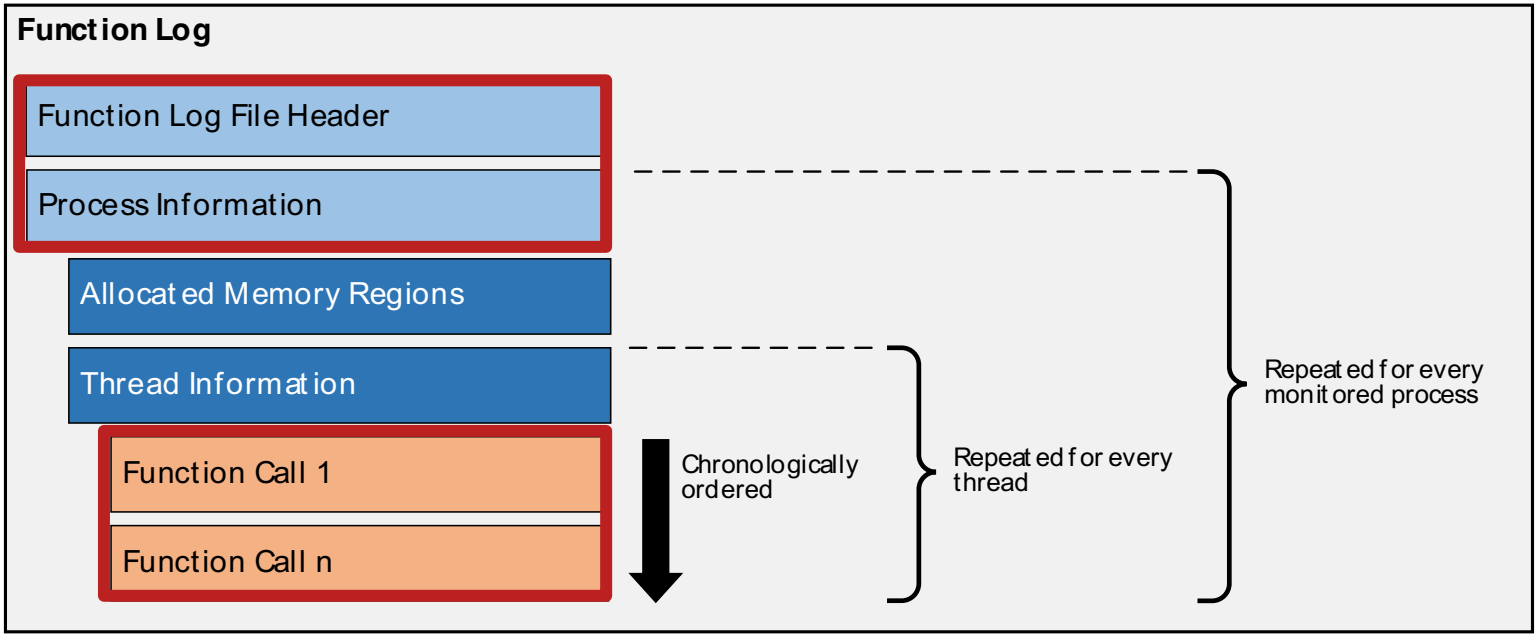


Figure based on https://www.vmray.com/wp-content/uploads/2019/03/Data-Sheet-VMRay-Analyzer.pdf

[1] C. Willems, R. Hund, T. Holz. https://www.syssec.ruhr-uni-bochum.de/media/emma/veroeffentlichungen/2012/11/26/TR-HGI-2012-002.pdf, 2012.

VMRay Analyzer Sandbox

Semantic Function Log Structure



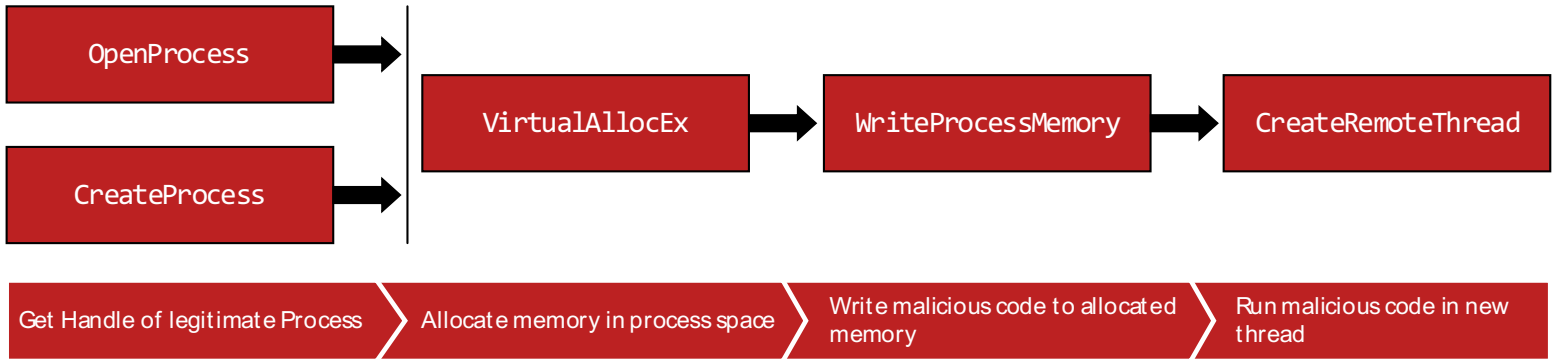
⚠ Only API calls directly called by the user application are traced in the function log!

dynmx Detection Approach

Malware Feature Classification

- ***Generic*** Malware Features
  - No tied to a specific malware family or species
  - Used across a wide variety of malware
  - Valuable resource for technique descriptions: MITRE ATT&CK®

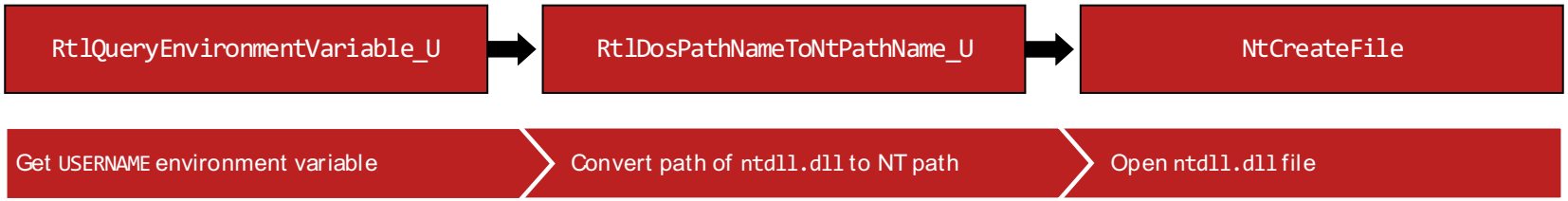
Direct code injection (MITRE ATT&CK® T1055)



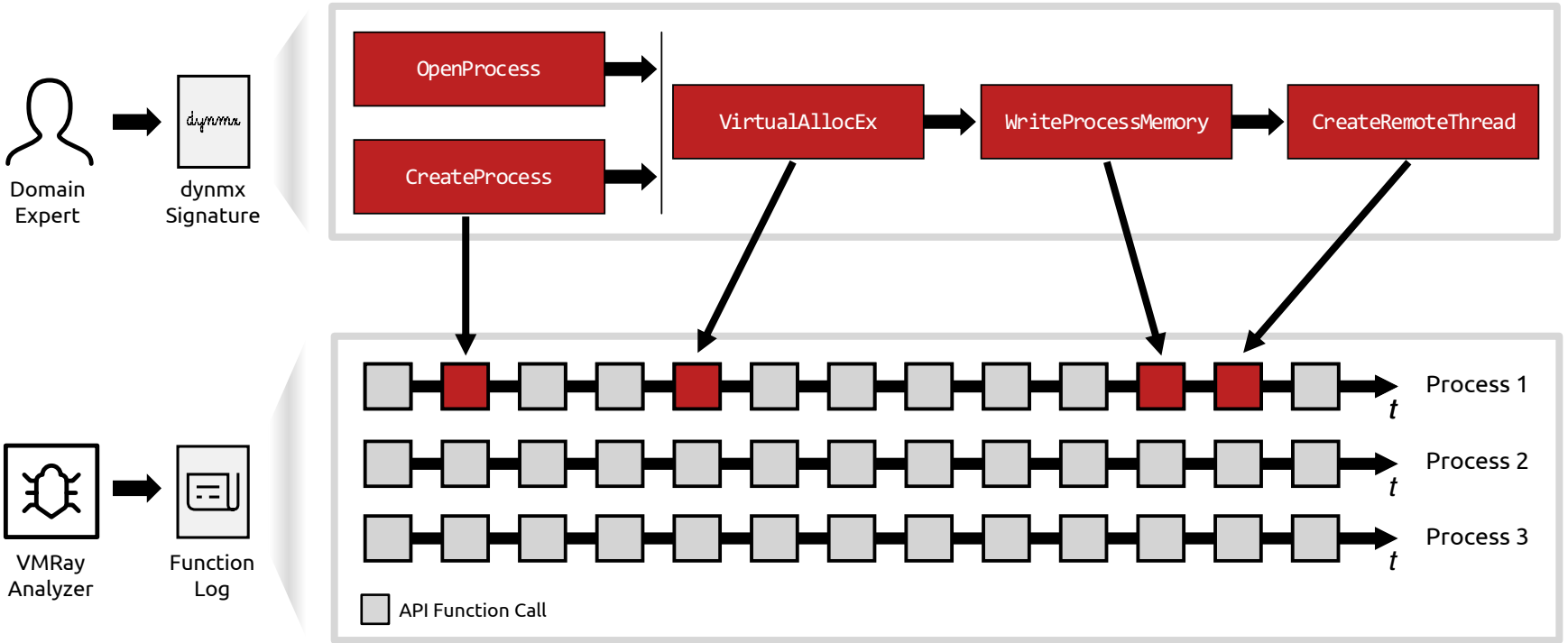
Malware Feature Classification (cont.)

- ***Specific*** Malware Features
  - Closely related to malware families
  - Used to characterise a certain malware family

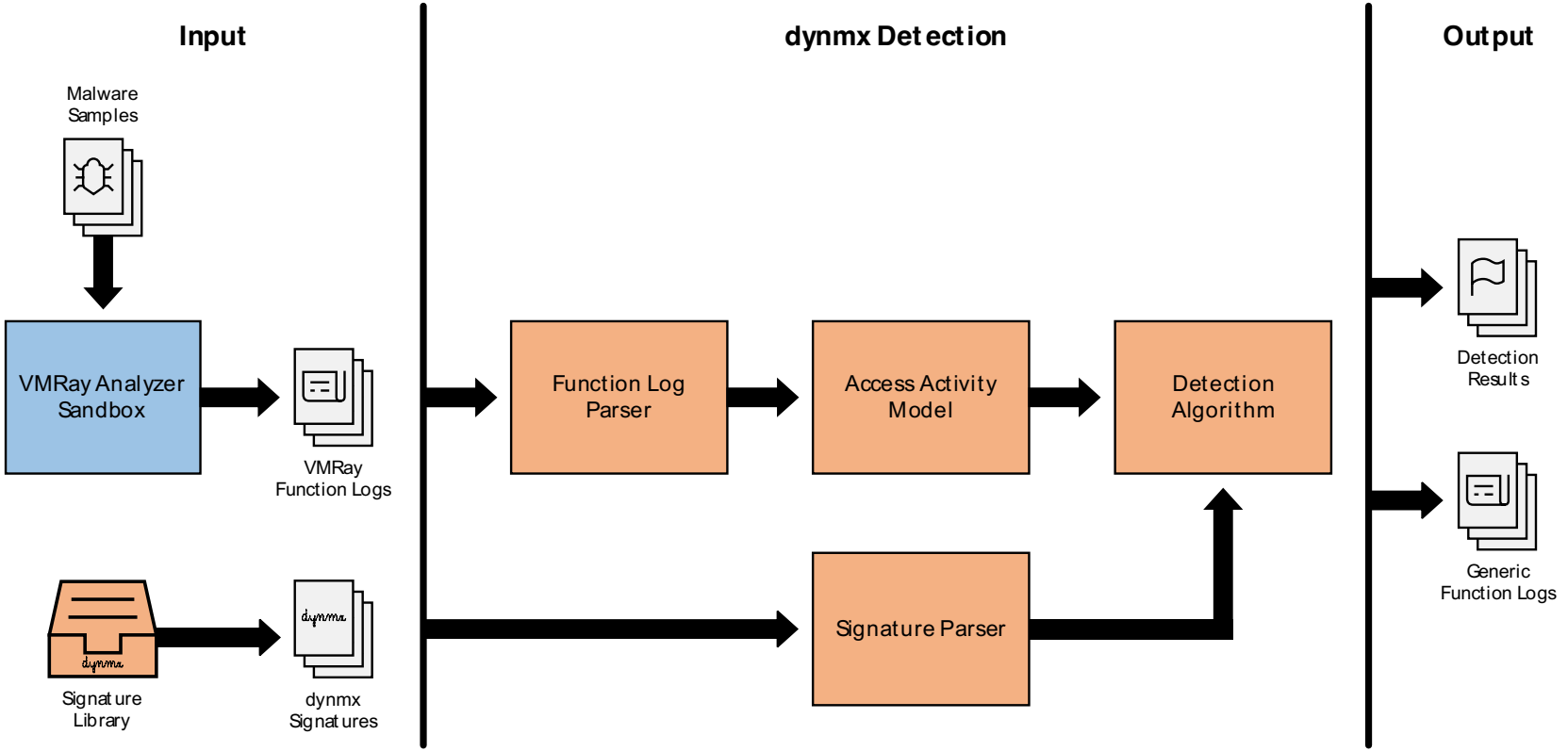
Formbook Malware Family



Example Detection



dynmx Detection Process

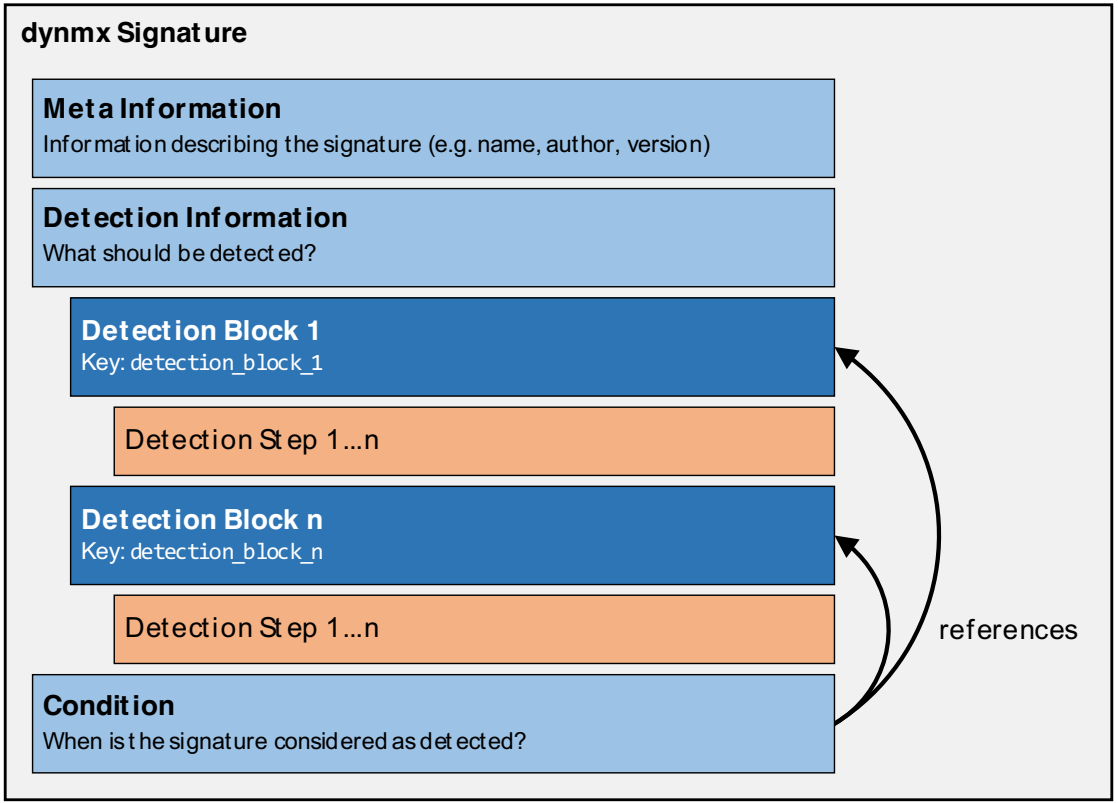


Signature DSL

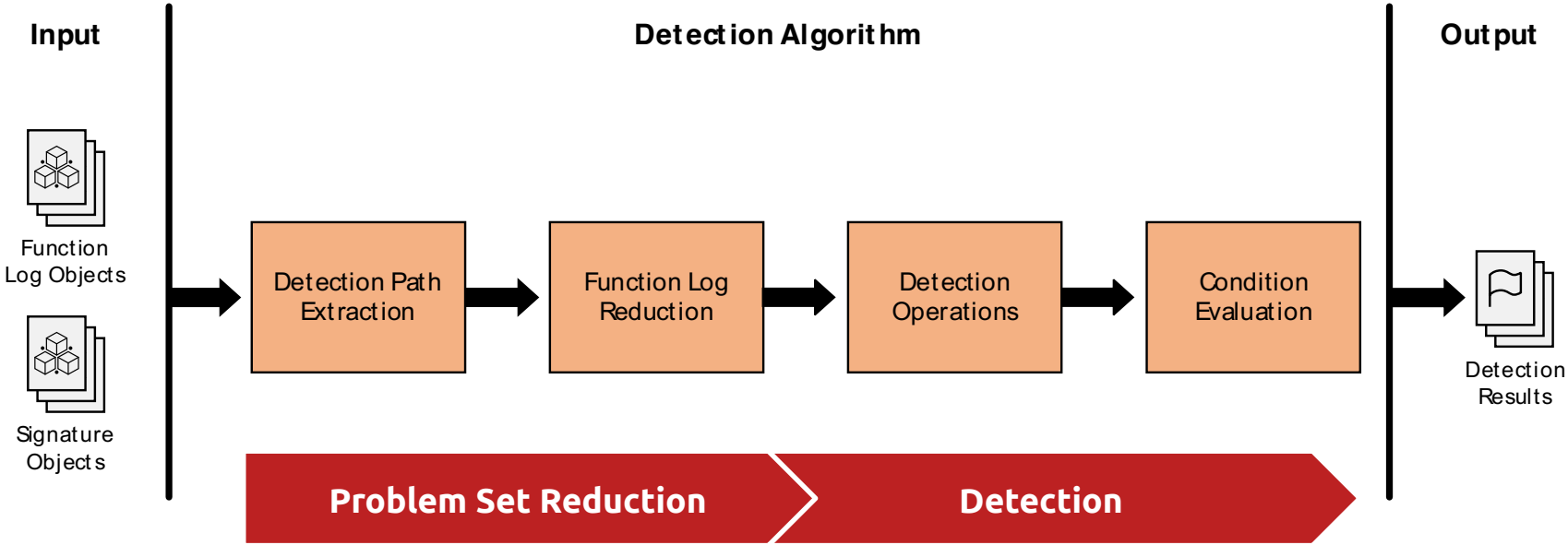
- YAML-based syntax
- Designed to define characteristic API call and resource usage in a sandbox-agnostic manner
- Basic signature structure adopted from YARA
- Features
  - As precise definition of API call or resource as needed
  - Usage of variables (typically to keep track of handles)
  - Condition (binary expression)
  - Two distinct detection types (`simple` and `sequence`)

Signature DSL

Basic Signature Structure

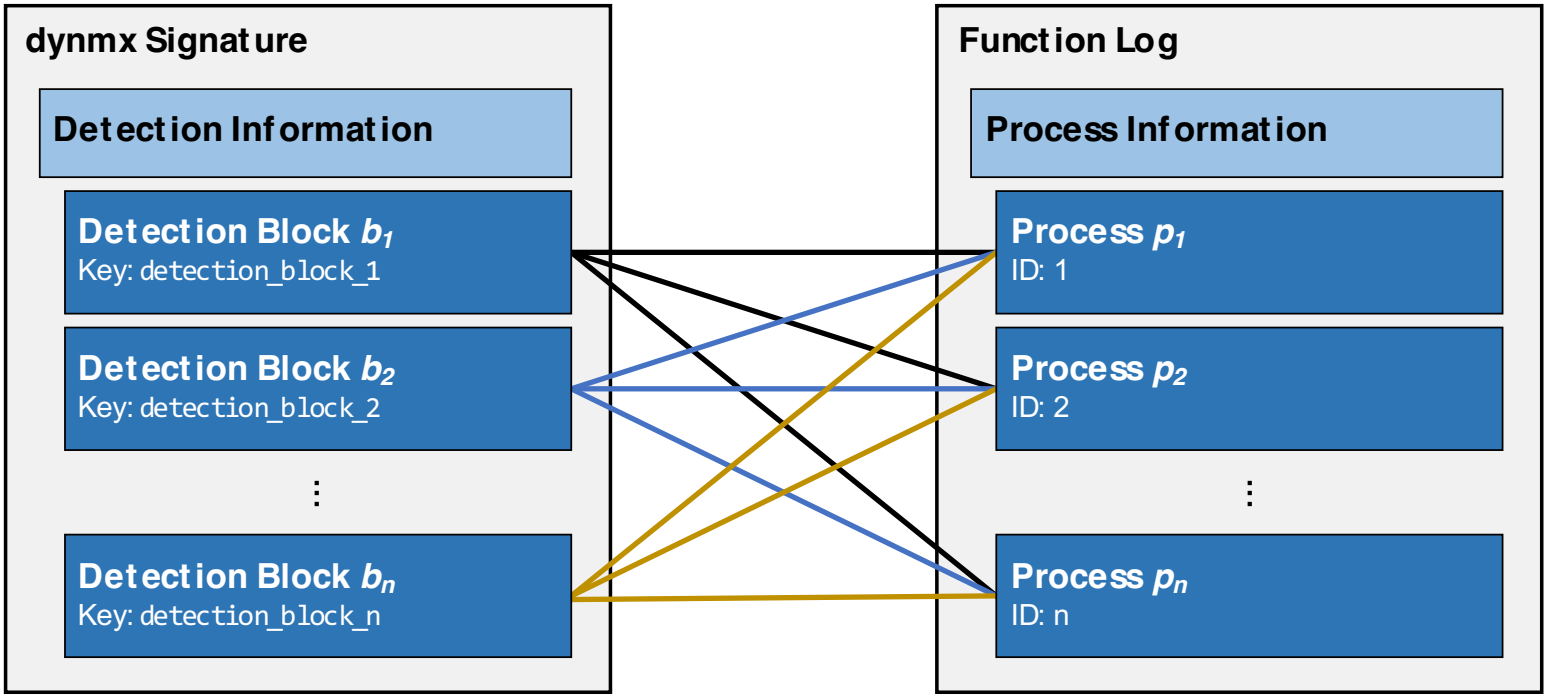


Detection Algorithm



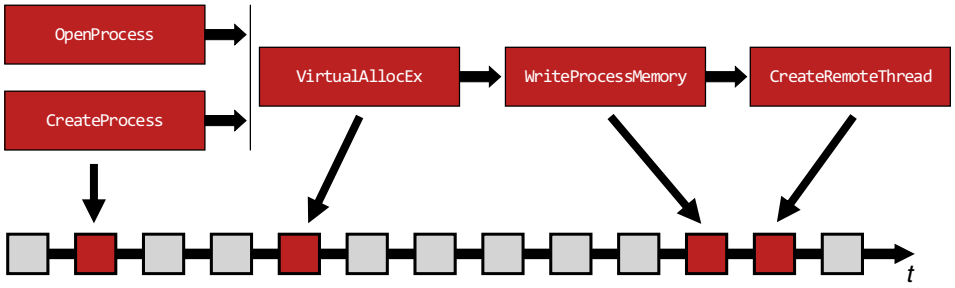
Detection Algorithm

Mode of Operation



Detection Algorithm

LCS Calculation



$$X = \{x_1, ..., x_m\} \text{ (reduced API call sequence)}$$
$$Y = \{y_1, ..., y_n\} \text{ (detection path)}$$

$$LCS_{API}(X_i, Y_j) = \begin{cases} \emptyset, & \text{if } X = \emptyset \text{ or } Y = \emptyset, \\ LCS_{API}(X_{i-1}, Y_{j-1}) + \text{API call} & \text{if } \texttt{detect}(x_i, y_j), \\ max(LCS_{API}(X_i, Y_{j-1}), LCS_{API}(X_{i-1}, Y_j)) & \text{if not } \texttt{detect}(x_i, y_j) \end{cases}$$

X \ Y	∅	y_1	y_2	y_3
∅	∅	∅	∅	∅
x_1	∅	0	0	0
x_2	∅	0	0	0
x_3	∅	***1***	1	1
x_4	∅	1	***2***	2
x_5	∅	1	2	***3***
x_6	∅	1	2	3

dynmx Demo

Evaluation Results

Evaluation Data Sets

Malicious Data Set

- 31,898 function logs of real-world malware
- Analysis in on-premise VMRay Analyzer instance of Deutsche Telekom Cyber Defense Center
- ~74 % of function logs unkown in terms of malware family
- ~18% of function logs origin from Emotet malware samples
- Sources:
  - Internal DTAG tools (malware database, MCE Tool)
  - Malpedia

Benign Data Set

- 1,421 function logs of real-world benign software
- Analysis in on-premise VMRay Analyzer instance of Deutsche Telekom Cyber Defense Center
- Sources:
  - Windows 10 x64 executables
  - Portable applications
  - Forensic tools
  - Benign documents uploaded to VirusTotal

Generic Malware Features

Generic Malware Feature	Detection Discrepancy dynmx vs. VMRay	False Positive dynmx Detections
Process Hollowing (T1093)	**100.00 %**	0 %
DLL Injection (T1055)	**100.00 %**	0 %
Direct Code Injection (T1055)	**100.00 %**	0 %
IAT Hooking (T1179)	**100.00 %**	**100.00 %**
Windows Registry Run Keys (T1060)	**3.84 %**	0 %
Winlogon DLLs (T1004)	**100.00 %**	0 %
AppInit DLLs (T1103)	**100.00 %**	0 %
Service Installation (T1050)	**0.01 %**	**0.0001 %**
Scheduled Task (T1053)	**4.96 %**	0 %

dynmx detected for all considered malware features more malware samples than VMRay Analyzer

Specific Malware Features

Malware Family	Precision *P*	Recall *R*	F_1 Score	Additionally detected
DarkComet	**1.0**	0.9919	0.9959	**1,941**
Emotet	**1.0**	0.9953	0.9976	**655**
Formbook	**1.0**	0.9587	0.9789	**377**
Remcos	**1.0**	0.9984	0.9992	**2,163**

Detection of malware families based on behavioural characteristics with zero false-positive rate and well coverage

Conclusion

Conclusion

The signature DSL allows domain experts to define malware behaviour precisely and enables them to share signatures collaboratively.

Signatures can be detected automatically with the help of the introduced detection algorithm which reduces recurring analysis tasks.

The dynmx detection approach has the potential to improve the efficiency of the malware analysis process in general.

Limitations

- Process-centric detection → malware features implemented with several processes working together cannot be detected
- Dependent on the function log quality
- Malware features need to be specific on the API call level
- Complex function logs can lead to long runtimes

Future Work

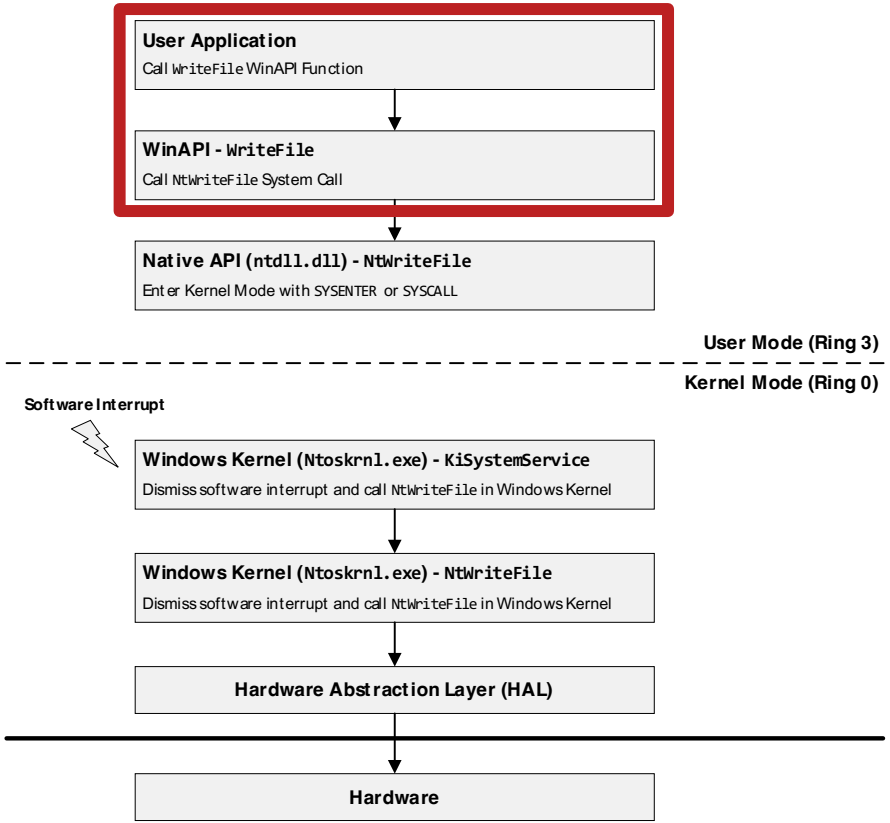
- Improvement of detection algorithm in terms of long runtimes for complex function logs
- Integration of further sandboxes
- Detection of malware features in memory images
- Malware Classification

Simon Jansen
<0x534a@mailbox.org>

Backup

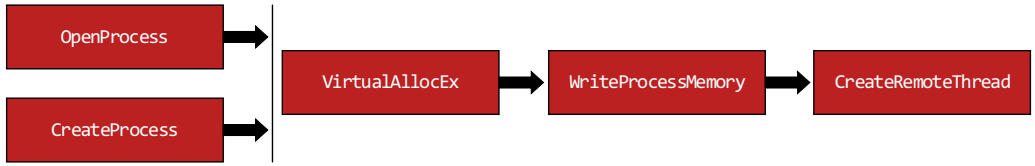
Windows API Call Cascade

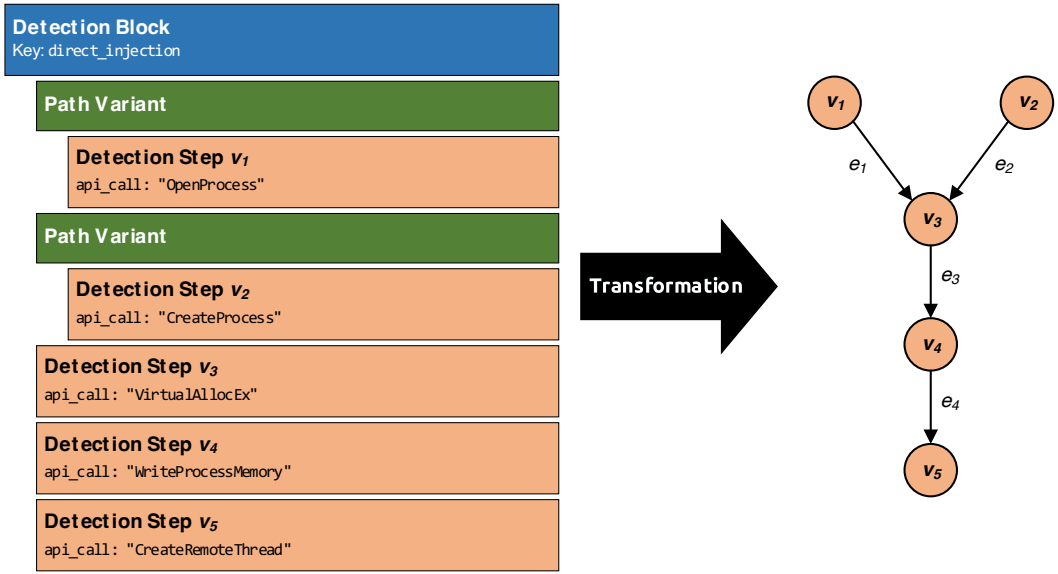
Traced API Calls



Signature DSL

Detection Graph Transformation





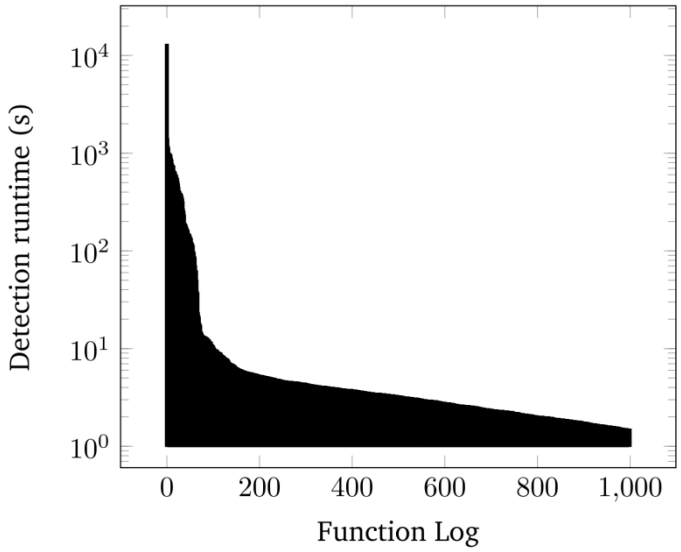
Performance Evaluation

Runtimes Malicious Data Set

Generic Malware Feature	Feature Type	Overall Runtime
Process Hollowing (T1093)	Generic	~44.42min
DLL Injection (T1055)	**Generic**	**~236.77min**
Direct Code Injection (T1055)	Generic	~43.35min
IAT Hooking	Generic	~51.75min
Windows Registry Run Keys (T1060)	Generic	~107.2min
Winlogon DLLs (T1004)	Generic	~70.15min
AppInit DLLs (T1103)	Generic	~83.17min
Service Installation (T1050)	Generic	~139.82min
Scheduled Task (T1053)	Generic	~89.23min
DarkComet	Specific	~43.48min
Emotet	Specific	~75.85min
Formbook	**Specific**	**~42.4min**
Remcos	Specific	~79.03min

Performance Evaluation

Detection Run Times of DLL Injection Feature



Memory Consumption