Metamorphic Malware Behavior Analysis using Sequential Pattern

M. Saqib Nawaz[1], P. Fournier-Viger[1], M. Zohaib Nawaz[2], Guoting Chen[1], Youxi Wu[3]

[1]Harbin Institute of Technology (Shenzhen), China

[2]University of Sargodha, Pakistan

[3]Hebei University of Technology, Tianjin, China

MLiSE @ ECML PKDD 2021

17 September, 2021

Introduction

Malware: a malicious software that gets installed on a computing device and perform unwanted tasks.

Types of malware

5 Ransomware
4 Adware
Trojan 6
3 Spyware
Worms 7
2 Cryptojacking
Rootkits 8
1 Malvertising
Backdoors 9

Some recent statistics:

- 87% increase in malware infections in last ten years.
- Malware attacks in 2019 have costed the average US company an avg. of $2.4 million per year.
- Appx. 77 Million new malicious activities were detected in Q1 2021 (McAfee Report).

- As malware can cause enormous loss and adverse effects, its analysis and detection is an important research topic in cybersecurity.

Introduction

- Anti-malware software that provides protection against malware mainly use signature-based method for malware detection. However,
  - Extracting signature is a tedious and time consuming activity and requires funds and expertise.
  - These method looks for already known malicious patterns.
  - To overcome aforementioned drawbacks, data mining and machine learning techniques are now used.
- **In this study, we focus on Windows-based malware.**

Introduction

- On Windows OS, a malware needs to use some of the OS's services.

- The entire set of requests made by malware to get OS services through API calls creates malicious behavior.

- Sequential pattern mining (SPM) is well suited to analyze sequences of API calls made by malware.

SPM: discover meaningful and hidden knowledge in large sequential datasets.

Introduction

Contribution

An SPM-based approach is presented for the analysis of API calls sequences made by malware.

Basic idea

1. First convert the API calls sequences into a corpus that is suitable for learning, where each API call is converted into an integer.
2. SPM techniques are then used on the corpus to find:
   - Frequent API calls and their patterns,
   - Sequential rules between API calls patterns
   - Maximal and closed frequent API calls

Dataset

- The benchmark dataset [1] contains Windows API calls of various metamorphic malware.

- Appx. 20,000 malware were collected from GitHub.
- Checking the behavior of malware using Cuckoo Sandbox Environment.
- The dataset contains all Windows API call requests made by the malware on Windows 7 OS.

- The hash values of malware were searched with the VirusTotal service.
- Each malware's families was identified by using each analysis result that was obtained with VirusTotal.

1. Ö. Çatak, A. F. Yazi, O. Elezaj, and J. Ahmed. Deep learning based sequential model for malware analysis using Windows exe API calls. PeerJ Computer Science, 6:e285, 2020

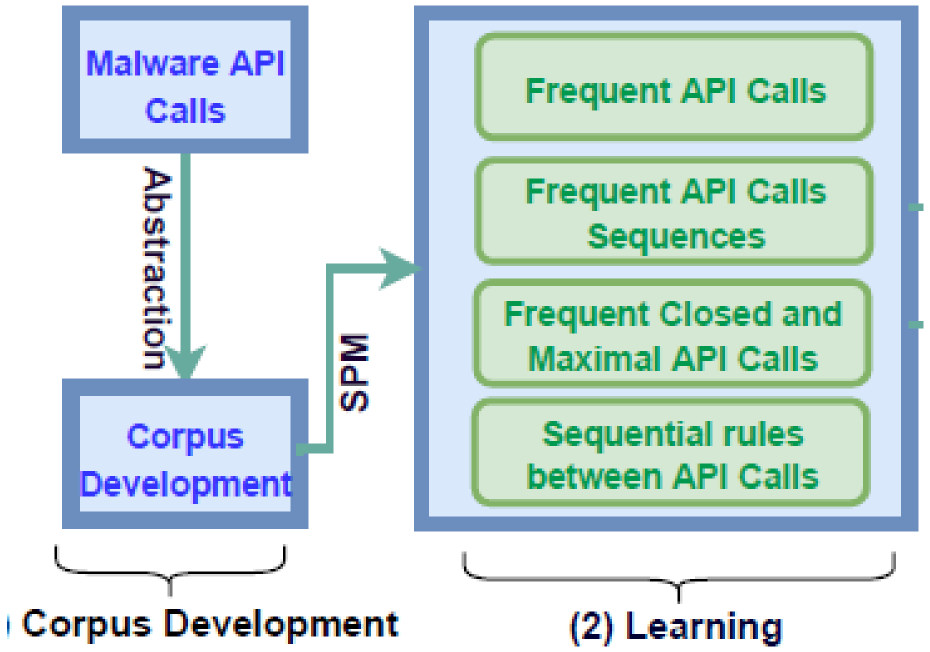
Dataset

The final dataset contains 8 different leading malware families

Distribution of malicious software according to their families

Malware Family	Samples	Description
Adware	379	Hides on your device and serves you advertisements
Backdoor	1001	bypass the system security mechanism undetectably to access a computer or its data
Downloader	1001	share the primary functionality of downloading content
Dropper	891	Surreptitiously carries viruses, backdoors and other malicious software so they can be executed on the compromised machine
Virus	1001	spread from host to host and has the ability to replicate itself
Worms	1001	spreads copies of itself from computer to computer
Trojan	1001	misleads users of its true intent
Spyware	832	enables a user to obtain covert information about another’s computer activities
Total	7107	

Proposed Methodology

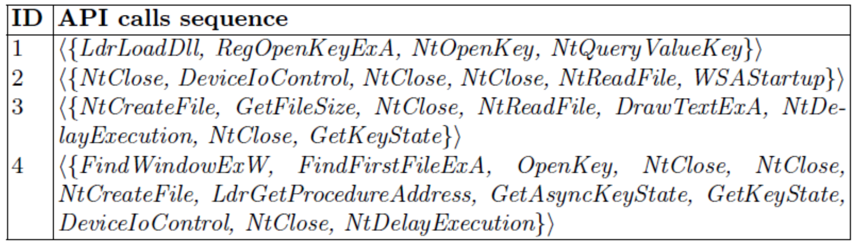

Malware API Calls
Abstraction
Corpus Development
SPM
Frequent API Calls
Frequent API Calls Sequences
Frequent Closed and Maximal API Calls
Sequential rules between API Calls
Corpus Development
(2) Learning

Proposed SPM-based approach

Corpus Development

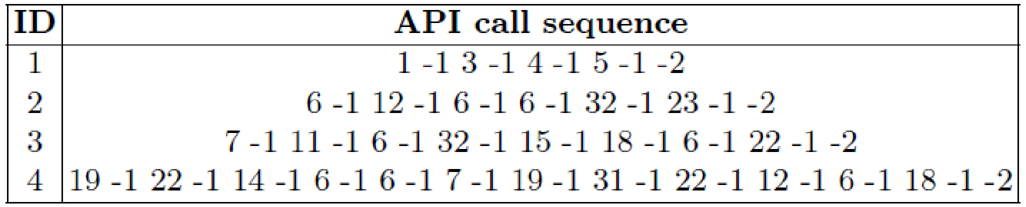
- Converting different malware API calls dataset into a suitable format

A sample of an API calls corpus



ID	API calls sequence
1	⟨{*LdrLoadDll, RegOpenKeyExA, NtOpenKey, NtQueryValueKey*}⟩
2	⟨{*NtClose, DeviceIoControl, NtClose, NtClose, NtReadFile, WSAStartup*}⟩
3	⟨{*NtCreateFile, GetFileSize, NtClose, NtReadFile, DrawTextExA, NtDelayExecution, NtClose, GetKeyState*}⟩
4	⟨{*FindWindowExW, FindFirstFileExA, OpenKey, NtClose, NtClose, NtCreateFile, LdrGetProcedureAddress, GetAsyncKeyState, GetKeyState, DeviceIoControl, NtClose, NtDelayExecution*}⟩

Conversion of API calls sequences into integer sequences



ID	API call sequence
1	1 -1 3 -1 4 -1 5 -1 -2
2	6 -1 12 -1 6 -1 6 -1 32 -1 23 -1 -2
3	7 -1 11 -1 6 -1 32 -1 15 -1 18 -1 6 -1 22 -1 -2
4	19 -1 22 -1 14 -1 6 -1 6 -1 7 -1 19 -1 31 -1 22 -1 12 -1 6 -1 18 -1 -2

Learning through SPM

- Various SPM algorithms are used:
  - **Apriori**: Discover frequent API calls
  - **TKS**: Discover Top-k most frequent sequential API Calls patterns
  - **CM-SPAM**: Discover frequent sequential API calls patterns
  - **ERMiner**: Discover sequential rules between frequent API calls
  - **CloFast**: Discover closed sequential API calls patterns
  - **VMSP:** Discover maximal sequential API calls patterns

- SPMF[1] data mining library, implements more than 180 data mining algorithms, is used to analyze the API calls corpus.

[1] http://www.philippe-fournier-viger.com/spmf/

Results

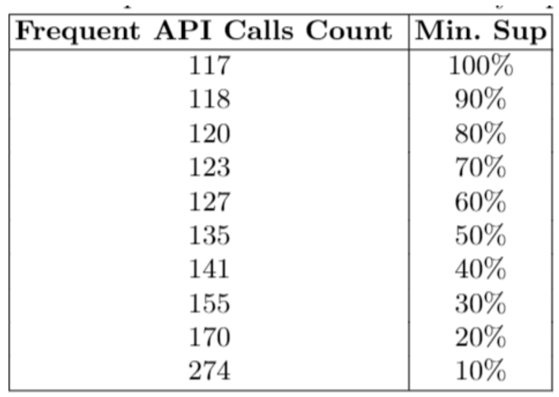
Stats about the dataset

Malware Type	Samples	API calls	Avg. Sequence Length
Adware	379	212	6867
Backdoor	1001	227	11293
Downloader	1001	232	6522
Dropper	891	226	16008
Virus	1001	241	18370
Worms	1001	236	33614
Trojan	1001	255	13818
Spyware	832	229	46951

- The first **1000 sequences contain 254 distinct API calls**
- On average, each sequence contains **approximately 11,502 API calls**.

Apriori Results

Frequent API Calls extracted by Apriori



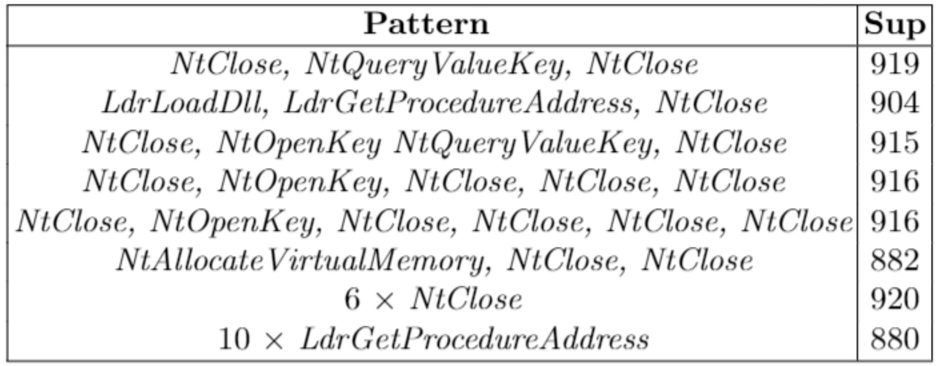
Frequent API Calls Count	Min. Sup
117	100%
118	90%
120	80%
123	70%
127	60%
135	50%
141	40%
155	30%
170	20%
274	10%

- The top five frequent API calls in 1000 sequences were:
  - *GetAsyncKeyState* (1,103,492)
  - *GetKeyState* (1,103,492)
  - *DeviceIoControl* (800,088)
  - *NtClose* (680,206)
  - *NtDelayExecution* (573,625)

- Frequent API calls sets obtained with Apriori are uninteresting, because:
  1. They are unordered.
  2. Apriori fails to discover important sequential patterns and also ignore the sequential relationship between API calls.

TKS Results

Frequent API Calls extracted with TKS



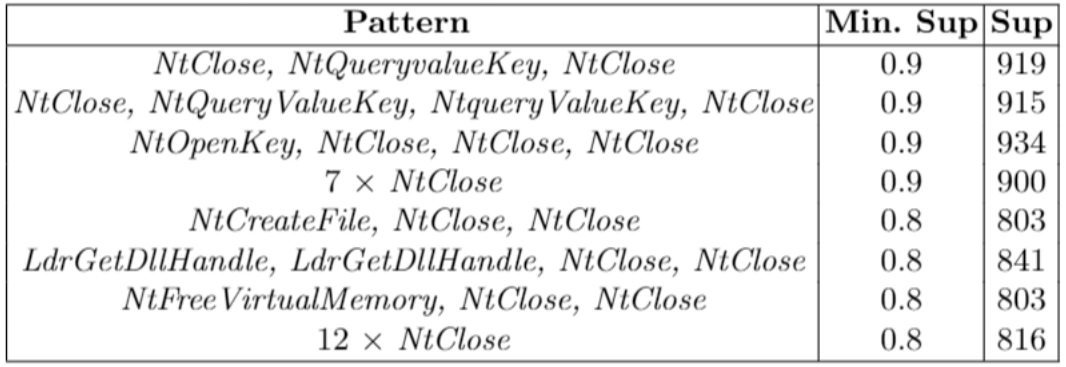
Pattern	Sup
NtClose, NtQueryValueKey, NtClose	919
LdrLoadDll, LdrGetProcedureAddress, NtClose	904
NtClose, NtOpenKey NtQueryValueKey, NtClose	915
NtClose, NtOpenKey, NtClose, NtClose, NtClose	916
NtClose, NtOpenKey, NtClose, NtClose, NtClose, NtClose	916
NtAllocateVirtualMemory, NtClose, NtClose	882
6 × *NtClose*	920
10 × *LdrGetProcedureAddress*	880

Sup indicates the occurrence count of each pattern in the corpus.

$$\textbf{Sup} = \frac{\text{number of sequences where the pattern occurs}}{\text{total number of sequences in the database}}$$

CM-SPAM Results

Frequent API Calls extracted with CM-SPAM

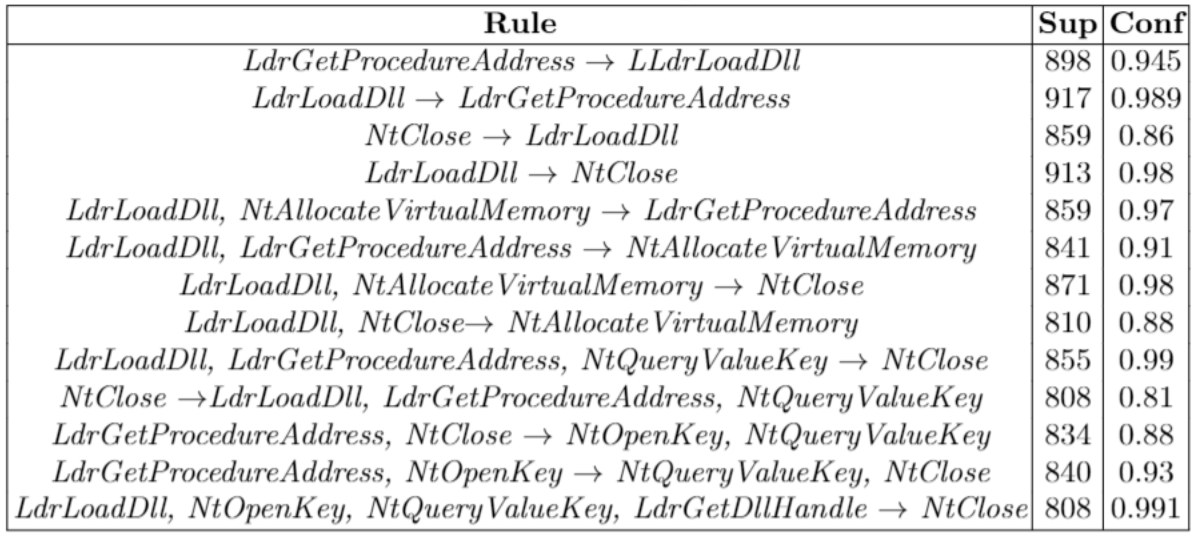


Pattern	Min. Sup	Sup
NtClose, NtQueryvalueKey, NtClose	0.9	919
NtClose, NtQueryValueKey, NtqueryValueKey, NtClose	0.9	915
NtOpenKey, NtClose, NtClose, NtClose	0.9	934
7 × *NtClose*	0.9	900
NtCreateFile, NtClose, NtClose	0.8	803
LdrGetDllHandle, LdrGetDllHandle, NtClose, NtClose	0.8	841
NtFreeVirtualMemory, NtClose, NtClose	0.8	803
12 × *NtClose*	0.8	816

- Unlike TKS, the CM-SPAM algorithm offers the ***minsup*** threshold.
- Discovered patterns with the **CM-SPAM algorithm are almost similar to the** results obtained with the **TKS algorithm**.

ER-Miner Results

Discovered sequential rules

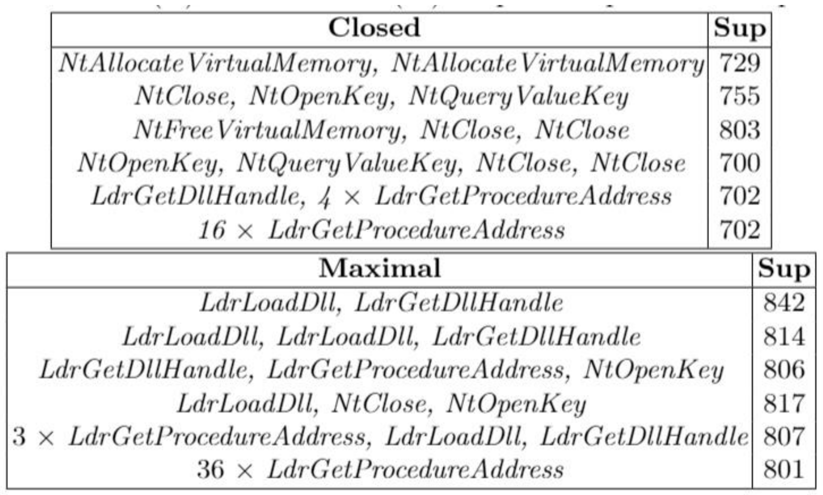


Rule	Sup	Conf
LdrGetProcedureAddress → *LLdrLoadDll*	898	0.945
LdrLoadDll → *LdrGetProcedureAddress*	917	0.989
NtClose → *LdrLoadDll*	859	0.86
LdrLoadDll → *NtClose*	913	0.98
LdrLoadDll, NtAllocateVirtualMemory → *LdrGetProcedureAddress*	859	0.97
LdrLoadDll, LdrGetProcedureAddress → *NtAllocateVirtualMemory*	841	0.91
LdrLoadDll, NtAllocateVirtualMemory → *NtClose*	871	0.98
LdrLoadDll, NtClose→ *NtAllocateVirtualMemory*	810	0.88
LdrLoadDll, LdrGetProcedureAddress, NtQueryValueKey → *NtClose*	855	0.99
NtClose →*LdrLoadDll, LdrGetProcedureAddress, NtQueryValueKey*	808	0.81
LdrGetProcedureAddress, NtClose → *NtOpenKey, NtQueryValueKey*	834	0.88
LdrGetProcedureAddress, NtOpenKey → *NtQueryValueKey, NtClose*	840	0.93
LdrLoadDll, NtOpenKey, NtQueryValueKey, LdrGetDllHandle → *NtClose*	808	0.991

- First rule: **94.5% (confidence) of the time**, the API *callLdrLoadDll* is followed after *LdrGetProcedureAddress* and this **rule has occurred 898 (support) times** in the corpus.

CloFast and VMSP Results

Closed (C) and Maximal (M) frequent sequential API patterns



Closed	Sup
NtAllocateVirtualMemory, NtAllocateVirtualMemory	729
NtClose, NtOpenKey, NtQueryValueKey	755
NtFreeVirtualMemory, NtClose, NtClose	803
NtOpenKey, NtQueryValueKey, NtClose, NtClose	700
LdrGetDllHandle, 4 × LdrGetProcedureAddress	702
16 × LdrGetProcedureAddress	702

Maximal	Sup
LdrLoadDll, LdrGetDllHandle	842
LdrLoadDll, LdrLoadDll, LdrGetDllHandle	814
LdrGetDllHandle, LdrGetProcedureAddress, NtOpenKey	806
LdrLoadDll, NtClose, NtOpenKey	817
3 × LdrGetProcedureAddress, LdrLoadDll, LdrGetDllHandle	807
36 × LdrGetProcedureAddress	801

- In maximal sequential API patterns, the **max gap** is set to1.
- The closed sequential patterns **provide a lossless representation of all sequential patterns** and they represent the largest subsequences common to sets of sequences.
- The maximal API calls patterns are **not loseless** and is always not larger than the set of closed sequential patterns and all sequential patterns.

Conclusion

Summary

- All the algorithms worked efficiently on the corpus
- Results indicated that the total number of API calls in each (abstraction simplicity) has a direct correlation on the efficiency of SPM algorithms.

Future Work

- Predicting the next API calls in a sequence using prediction models offered by SPMF.
- Using contrast set mining on the API calls of various malware to discover emerging (or contrasting) trends that show a clear and useful difference (or contrast) between various malware behavior.
- Taking the frequent patterns obtained with SPM algorithms as features for the development of malware samples classification and detection.

THANKS!