WIC3004 COMPUTER PENETRATION

You Are What You Do: Hunting Stealthy Malware via Data Provenance Analysis

NDSS 2020

NAME	: CHAN JIA LIANG
MATRIC NUMBER	: WIC180008 | 17128679/1
LECTURER NAME	: DR. MUHAMMAD REZA BIN Z'ABA
SESSION	: SEMESTER 2, 2020/2021

Table of Contents

Introduction 3
The PROVDETECTOR 6
  Overview: How It Works? 7
  Step 1: Data collection and storage 8
  Step 2: Provenance Graph Building 9
  Step 3: Representation Extraction 10
  Step 4: Embedding 11
  Step 4: Embedding 12
  Step 5: Anomaly Detection 13
  13
Conclusion 15

Introduction

Nowadays, traditional malwares are getting harder and harder to penetrate through computer devices for example they will get recognised and halted by the Windows Defender in Windows 10 as the community started to share the signature of virus among their common database and as well the involvement of machine learning approaches in uncovering the intrusion of malwares.

However, the cybercriminals do evolve their malware as well and now it's time to introduce to the stealthy malware. Instead of causing your PC slow or destroying your files and systems directly, most of them acts silently as backdoors to keep monitoring the victim or make use of victims PC as part of the botnet in attacking.

So, what's stealthy malware? We could say stealthy malware is the evolution from attackers so that traditional detection tools could not easily detect them because there is no malicious payload file available in memory disk to be scanned. Thus, stealthy malware will not store or embed its malicious payload together with targeted file anymore, instead stealthy malware tried to hide its own identity by impersonating among the benign behaviours of trusted application such as Command Prompt (cmd.exe) and Certification Authority Utility (certutil.exe).

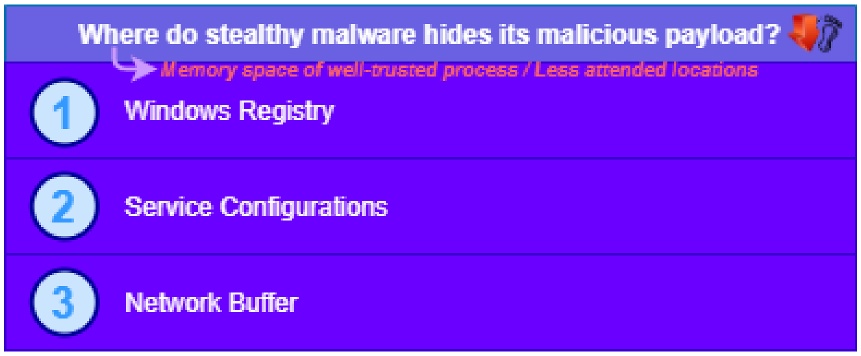
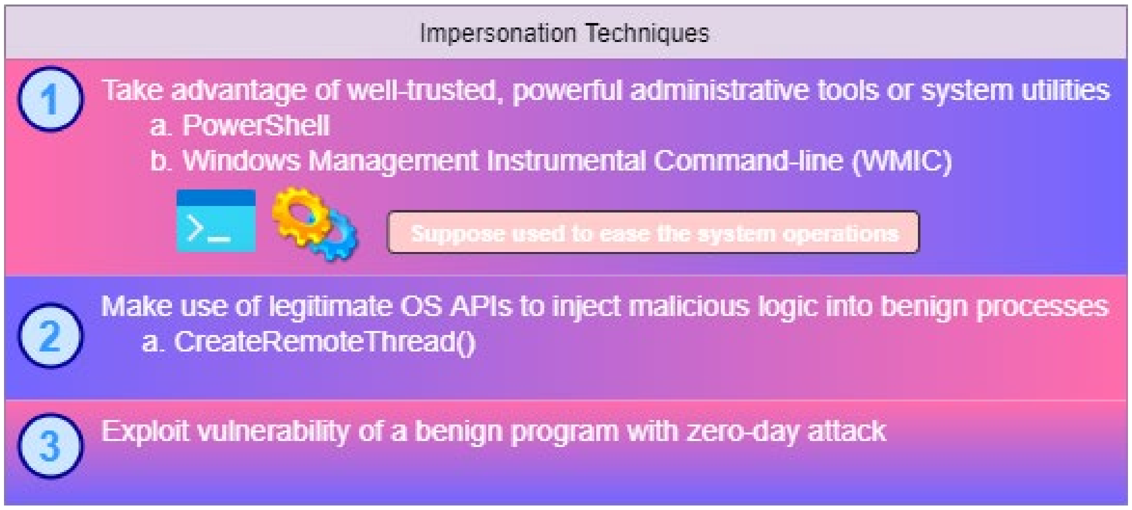

Where do stealthy malware hides its malicious payload?
Memory space of well-trusted process / Less attended locations
1 Windows Registry
2 Service Configurations
3 Network Buffer

Figure 1: Stealthy Malware Hideouts

With various impersonation techniques, stealthy malware is able to stay longer on the victims device and increase the chances to encrypt all victims file to get paid the ransom, and as well fileless mining to make use of botnet devices in gaining cryptocurrency. For example, WannaRen ransomware and PowerShellMiner is two well-known fileless malware respectively for the ransom and crypto mining. According to the researcher, these attacks are ten times more likely to succeed.


Impersonation Techniques

1. Take advantage of well-trusted, powerful administrative tools or system utilities
   a. PowerShell
   b. Windows Management Instrumental Command-line (WMIC)

   Suppose used to ease the system operations
2. Make use of legitimate OS APIs to inject malicious logic into benign processes
   a. CreateRemoteThread()
3. Exploit vulnerability of a benign program with zero-day attack

Figure 2: Three Types of Impersonation Techniques

Existing virus detection models which are based on file signatures and hashes are not able to detect modified unknown malware. Meanwhile, to catch such kind of computer virus , we should monitor their behaviour and processes instead. Hence, stealthy malwares can still be captured with provenance monitoring whereby gaining where the data comes from and where it goes.

With subsequent analysis and monitoring on a process instance, we can tell whether it's malicious or just normal behaviour especially when the malware is living off the land. For example, Microsoft Word which called PowerShell to run some script and open up socket connections to outsiders would consider as a very suspicious behaviour that legitimate application would not carry out such kind of processes.

PROVDETECTOR uses a novel selection technique based on this intuition to identify potentially harmful components in a process's OS-level provenance data. The system then uses a neural embedding and machine learning process to recognise any behaviour that deviates significantly from typical, common process.

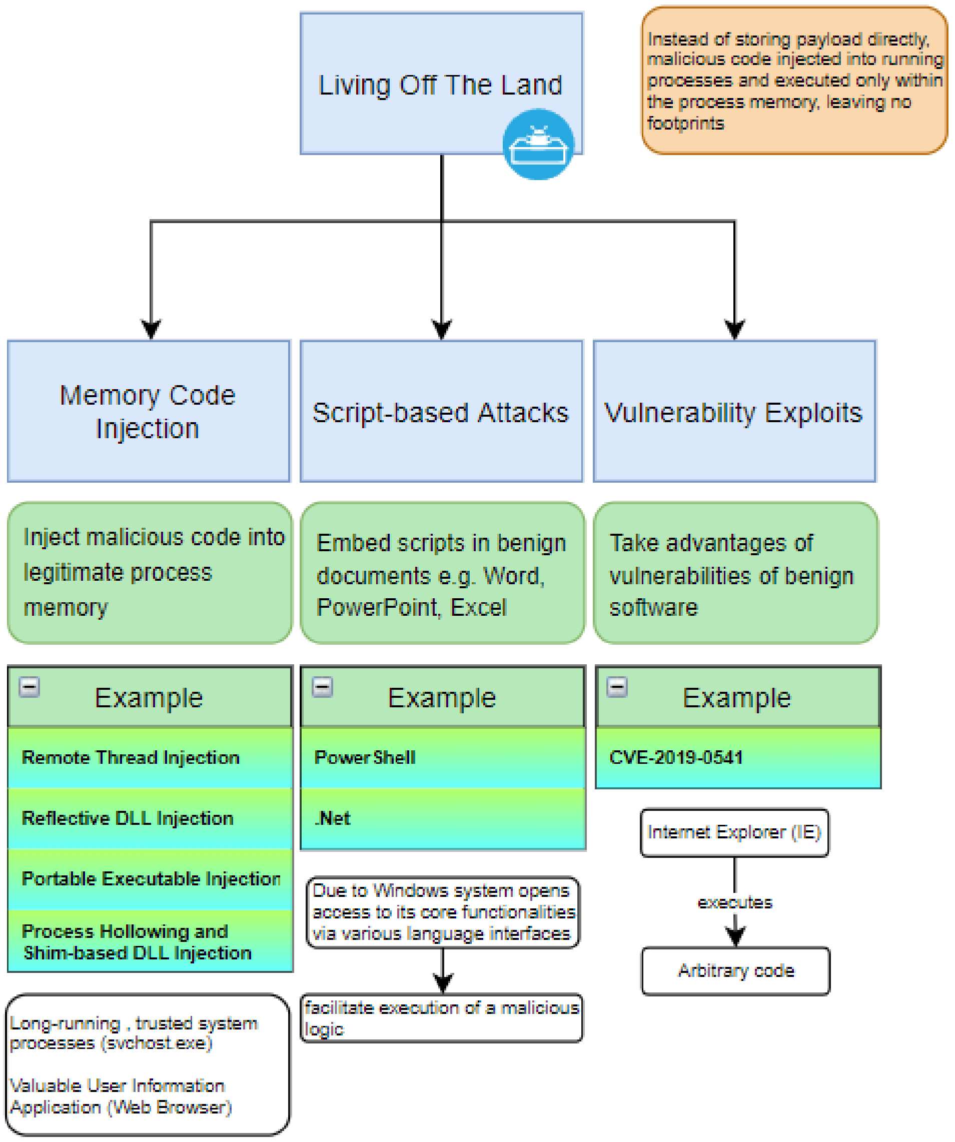

Living Off The Land
Instead of storing payload directly, malicious code injected into running processes and executed only within the process memory, leaving no footprints
Memory Code Injection
Script-based Attacks
Vulnerability Exploits
Inject malicious code into legitimate process memory
Embed scripts in benign documents e.g. Word, PowerPoint, Excel
Take advantages of vulnerabilities of benign software
Example
Remote Thread Injection
Reflective DLL Injection
Portable Executable Injection
Process Hollowing and Shim-based DLL Injection
Example
Power Shell
.Net
Example
CVE-2019-0541
Internet Explorer (IE)
executes
Arbitrary code
Due to Windows system opens access to its core functionalities via various language interfaces
facilitate execution of a malicious logic
Long-running , trusted system processes (svchost.exe)
Valuable User Information Application (Web Browser)

Figure 3: Explanation of Living Off The Land

Dear readers, you could refer to the Figure 1 and as well the second paragraph of the paper located at page 3 that describes an example of real-world Dynamic Data Exchange (DDE) script-based attack whereby no physical file has been stored in the disk while the malicious logic is being fetched from online and ran through the PowerShell which results in forming a backdoor that connect to Command and Control server (C&C). By then, the victims by then are fully under the attackers' control.

The PROVDETECTOR

A lightweight kernel-level provenance tracking and as well focuses on dealing with a wide field of impersonation techniques. With audit logs which comes along with OS-level provenance tracking, we could always find the fishy component of any mischievous processes.

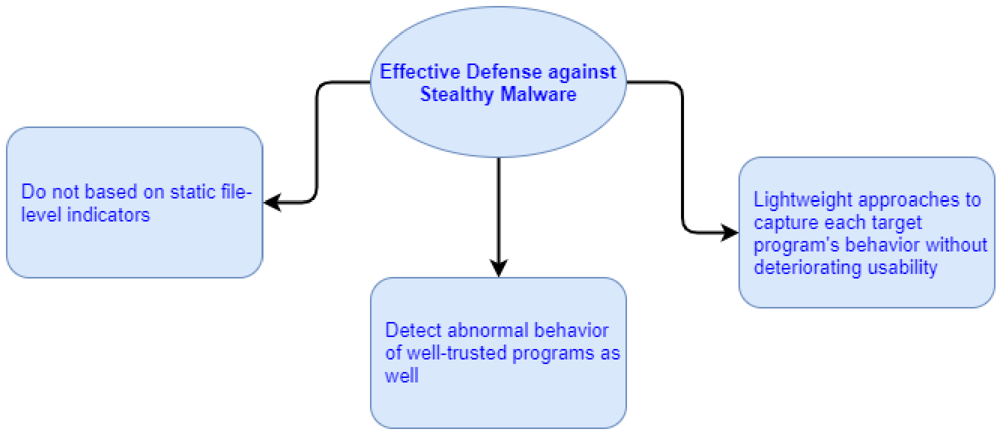

Effective Defense against Stealthy Malware
Do not based on static file-level indicators
Detect abnormal behavior of well-trusted programs as well
Lightweight approaches to capture each target program's behavior without deteriorating usability

Figure 4: Three Criteria of Effective Defence against Stealthy Malware

The researcher has taken these three measures into considerations when they design their system architecture in making sure to produce a stealthy malware detector which is effective and, in the meantime, not too excessive required a large amount of computing power.

With their proposed system, ProvDetector operates in a way that first by capturing dynamic behaviours of target processes and automatically detect for abnormalities from time-to-time as shown in Figure 5.

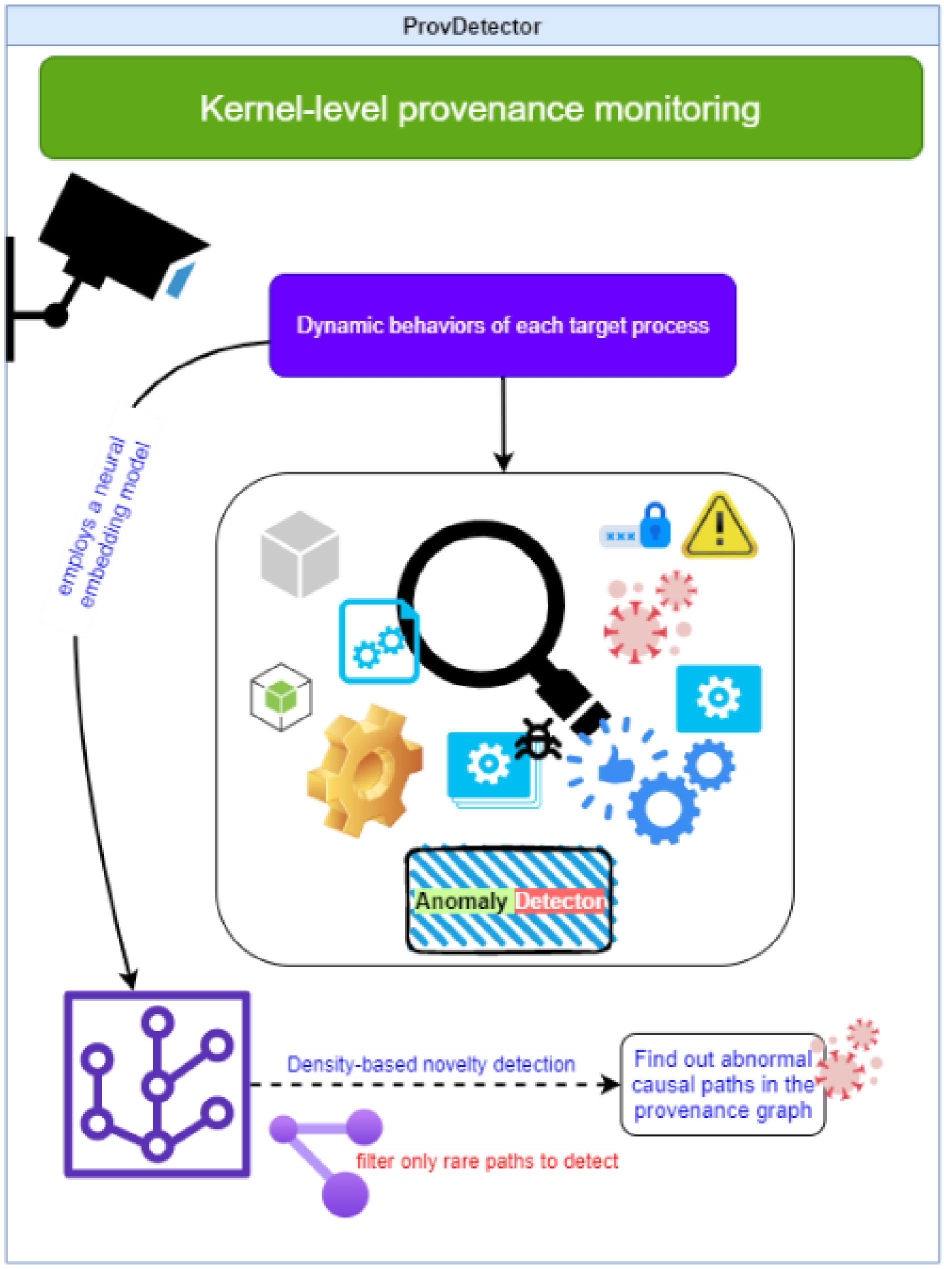

ProvDetector
Kernel-level provenance monitoring
Dynamic behaviors of each target process
employs a neural embedding model
Anomaly Detector
Density-based novelty detection
Find out abnormal causal paths in the provenance graph
filter only rare paths to detect

Figure 5: Operations of ProvDetector

Overview: How It Works?

There's basically five steps in ProvDetector and we are going to break them down one stage by one stage in the upcoming section. First thing first, the system starts with data collection. Second, build a provenance graph with the collected processes from first stage. Third, perform a feature extraction to gain only non-duplicate kind of common workloads which decrease the model training overhead and faster performance in real-world detection. Fourth, prepare the extracted features in a general, structured form of data before fitting in for model training. Fifth, train the model for every targeted programs respectively and anomaly detection is ready to up and run.

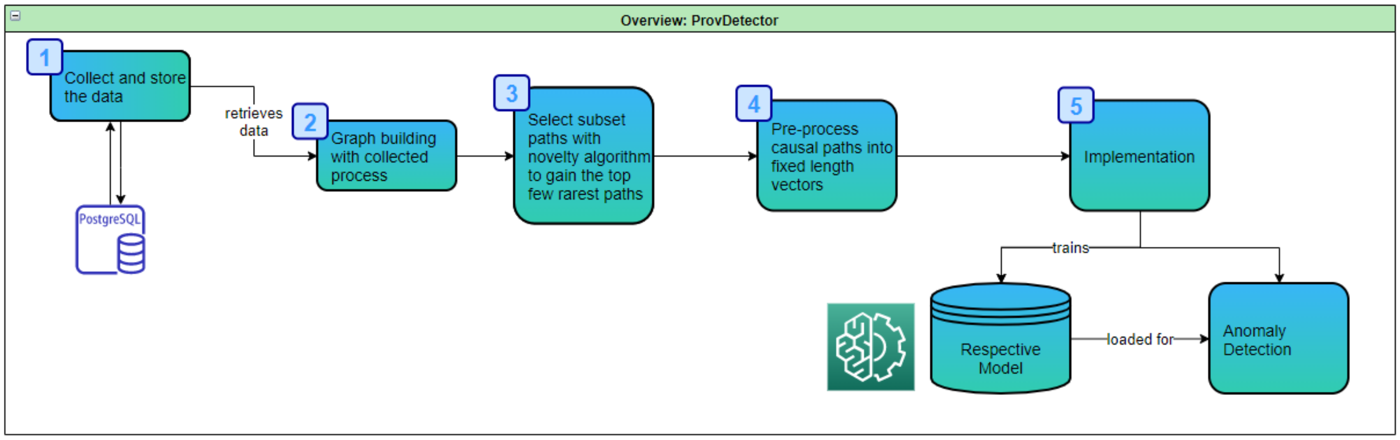

Overview: ProvDetector
1
Collect and store the data
retrieves data
PostgreSQL
2
Graph building with collected process
3
Select subset paths with novelty algorithm to gain the top few rarest paths
4
Pre-process causal paths into fixed length vectors
5
Implementation
trains
Respective Model
loaded for
Anomaly Detection

Figure 6: Overview of ProvDetector

Step 1: Data collection and storage

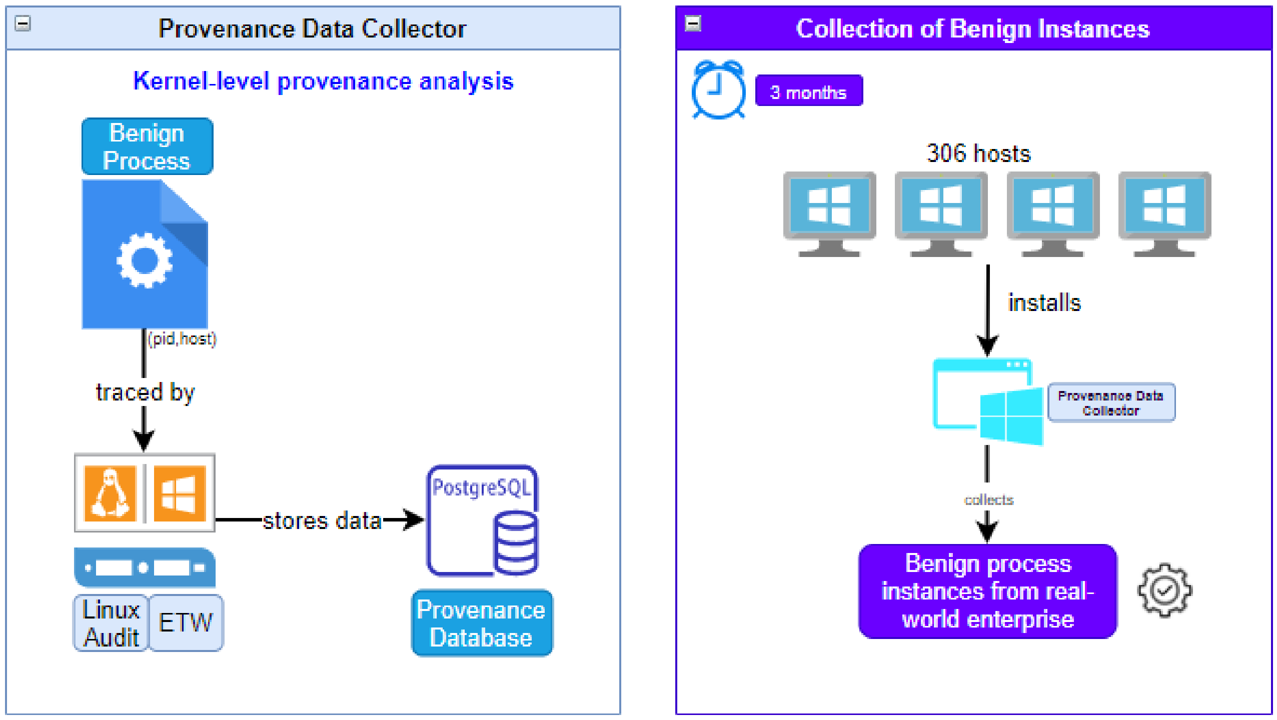
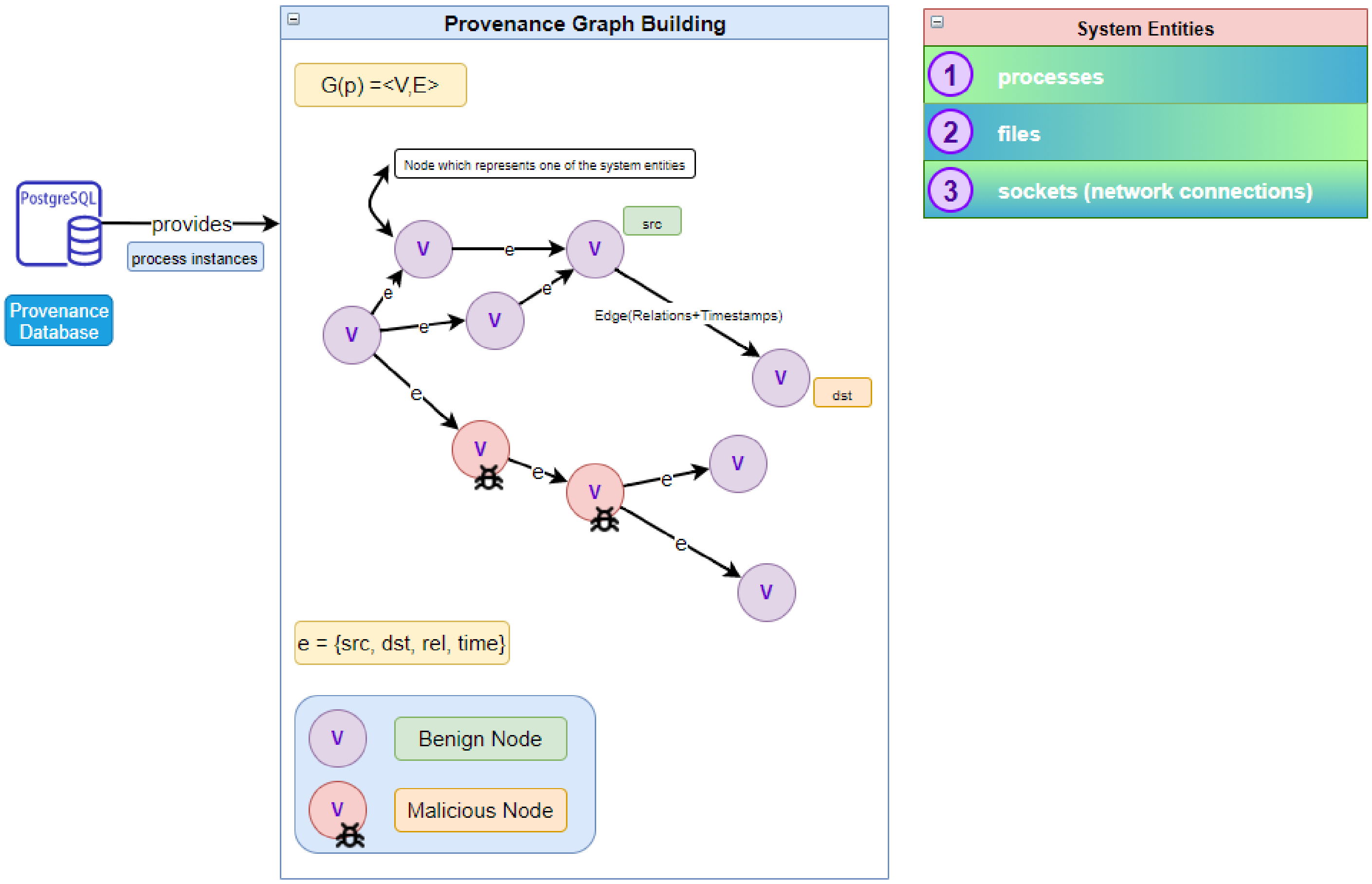

Provenance Data Collector
Kernel-level provenance analysis
Benign Process
(pid,host)
traced by
stores data
PostgreSQL
Linux Audit
ETW
Provenance Database
Collection of Benign Instances
3 months
306 hosts
installs
Provenance Data Collector
collects
Benign process instances from real-world enterprise

Figure 7: Data Collection and Storage of ProvDetector

The very first step is to collect raw data and store it in the database. So, we can analyse the data and find a way to pre-process the data to train the model for malware detection afterwards. With OS kernel either in Linux or Window, both supports the operation of collecting data for provenance analysis with a fair overhead. In Linux, we can make use of the Linux Audit framework meanwhile we can implement Windows Event Tracking (ETW) framework on Windows operating system. With ETW, it provides a mechanism to trace and log event that are raised by user-mode applications and kernel-mode drivers. In this ProvDetector system, the researcher suggests to train the model only with benign instances and hence they have collected their required data for engaged 23 programs from an enterprise with a total of 306 devices running over 3 months duration.

Step 2: Provenance Graph Building



PostgreSQL
provides
process instances
Provenance Database
Provenance Graph Building
G(p) =<V,E>
Node which represents one of the system entities
src
dst
Edge(Relations+Timestamps)
e
V
e = {src, dst, rel, time}
Benign Node
Malicious Node
System Entities
1 processes
2 files
3 sockets (network connections)

Figure 8: Provenance Graph Building of ProvDetector

First, a process instance of a program is the process created in one execution of the program. It's like an example of process running by a program at a duration. Basically, the researcher concluded that there are three type of system entities and we can consider them as nodes which is represented as vertices in graph representation. From one process instance to another, there are relations together with timestamp recorded which is represented as edges in graph form. In this stage, the system would build a provenance graph which clearly record down all the origin and destination of each process instances that have been created in a specific program which acts as raw fundamental data to be passed into stage 3 for extraction.

Step 3: Representation Extraction

Before we deep dive into the working principles, let us first understand what's causal paths. In general, a causal path record down the relationships of entities among the previous and the subsequent components. For example, a text file is previously written by Microsoft Word and going to be read by Outlook for email attachment.

For further information on causal paths, readers are encouraged to view the section C of the original paper located at page 5.

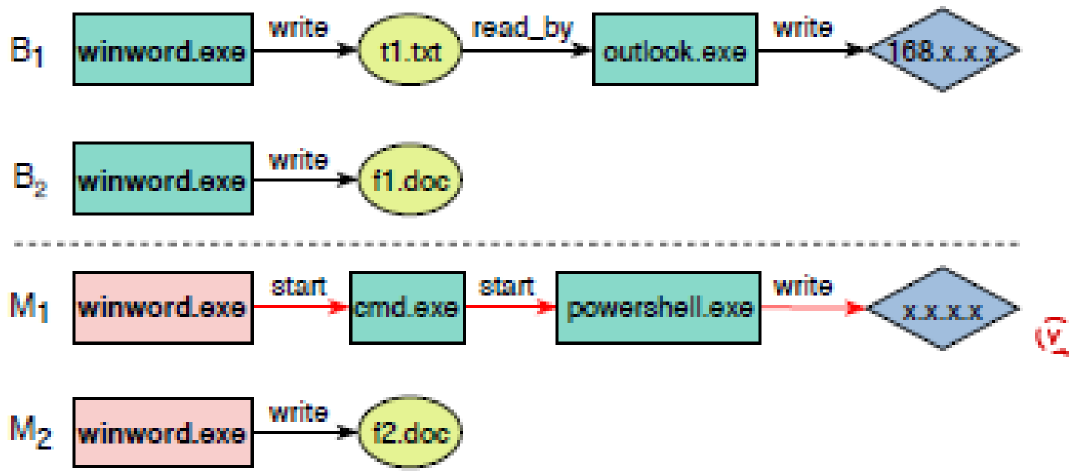


Fig. 4: Example causal paths from the provenance graphs in Figure 2. We concretize the * with file names.

Figure 9: Example of causal paths ***[Retrieved from the original paper]***

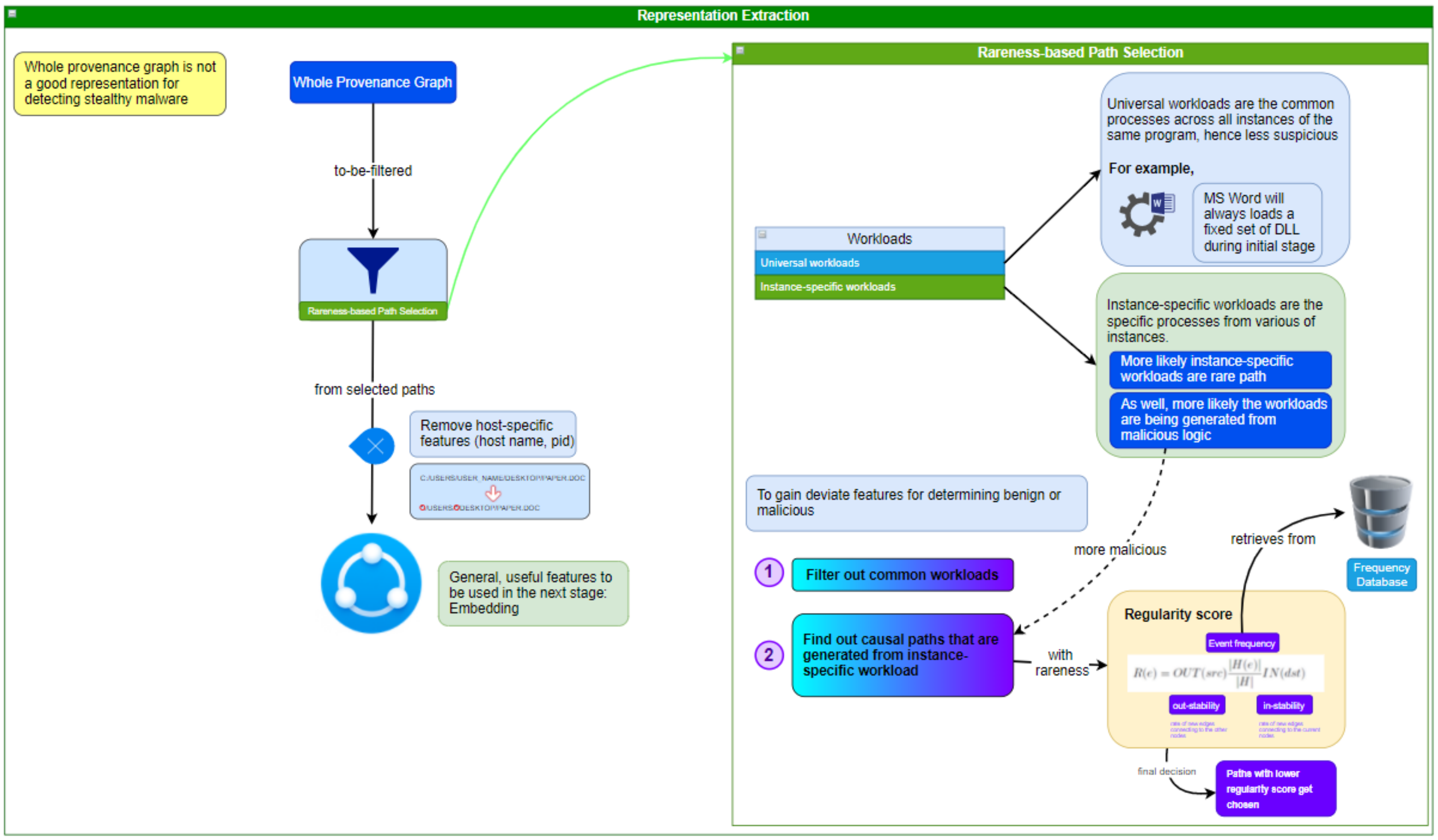
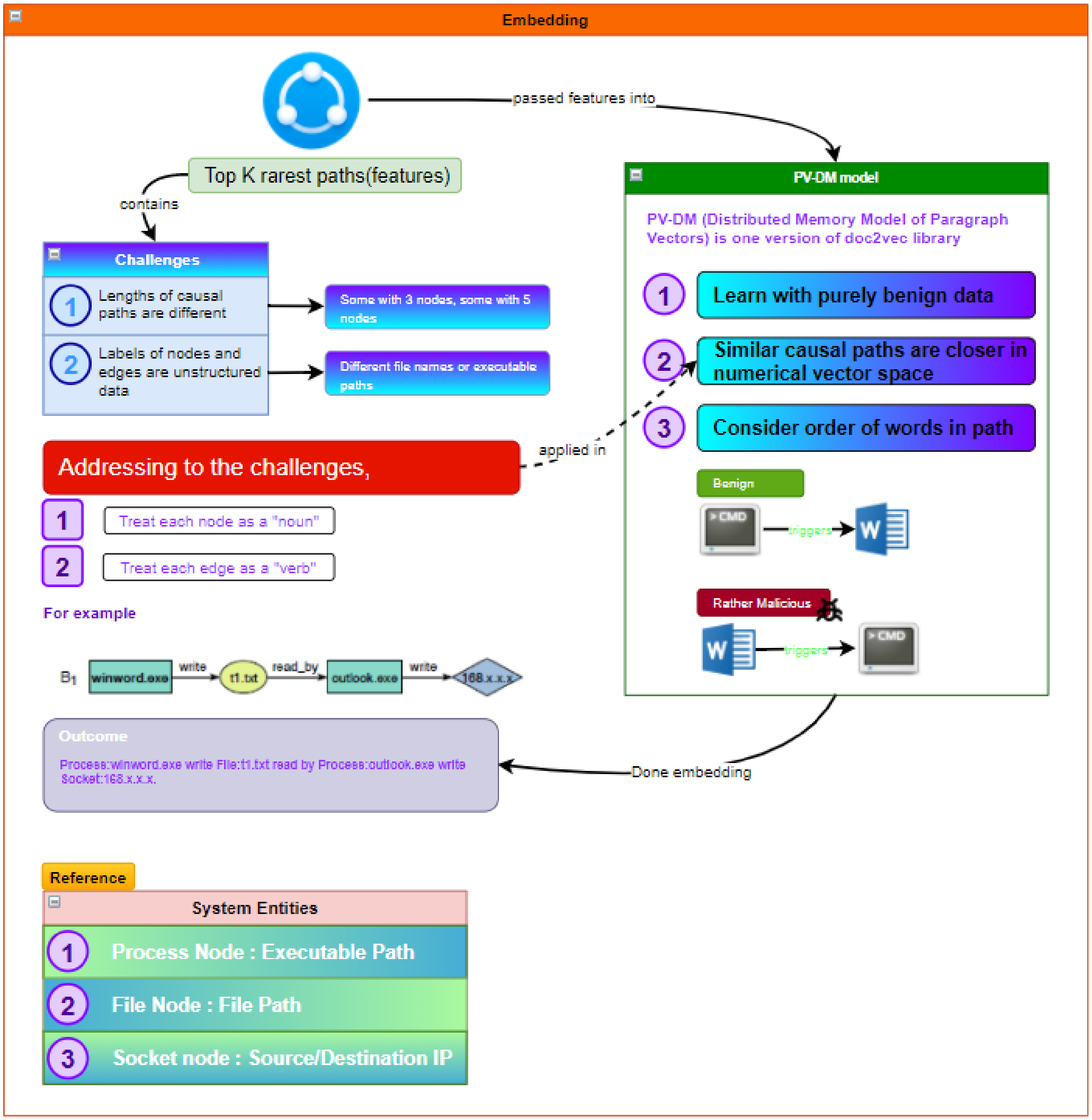

Representation Extraction
Whole provenance graph is not a good representation for detecting stealthy malware
Whole Provenance Graph
to-be-filtered
Rareness-based Path Selection
from selected paths
Remove host-specific features (host name, pid)
C:/USERS/USER_NAME/DESKTOP/PAPER.DOC
/USERS//DESKTOP/PAPER.DOC
General, useful features to be used in the next stage: Embedding
Rareness-based Path Selection
Universal workloads are the common processes across all instances of the same program, hence less suspicious
For example,
MS Word will always loads a fixed set of DLL during initial stage
Workloads
Universal workloads
Instance-specific workloads
Instance-specific workloads are the specific processes from various of instances.
More likely instance-specific workloads are rare path
As well, more likely the workloads are being generated from malicious logic
To gain deviate features for determining benign or malicious
more malicious
retrieves from
Frequency Database
1
Filter out common workloads
2
Find out causal paths that are generated from instance-specific workload
with rareness
Regularity score
Event frequency
$R(e) = OUT(src)\frac{|H(e)|}{|H|}IN(dst)$
out-stability
in-stability
rate of new edges connecting to the other nodes.
rate of new edges connecting to the current nodes.
final decision
Paths with lower regularity score get chosen

Figure 10: Representation Extraction of ProvDetector

Basically, there are a lot of common, universal workloads that are going to be run by a certain program and all these universal workloads are getting recorded in the provenanve graph. In stage 3, we are like performing data cleaning to gain only the instance-specific paths which we mentioned as features to be emdedded in next stage for model training. This is utmost important and smart step proposed by the researcher. With the representation extraction, they did save times and computing piower in running the anomaly detector in real-life. As a result, the ProvDetector outperforms other similar project in detecting stealthy malware in term of precision and speed.

Hence, they are able to extract out specific paths of benign behaviours that are non-duplicate operations according to its rareness. The researcher team have proposed an algorithm in calculting regularity score whereby the lower the regularity score, the more rare an event has happened before or either connected to related others instances. Meanwhile, malicious behaviours are mostly considered as rare path as well. **With the rare benign behaviors as training dataset**, the model can precisely detect malicious behaviours which try to blend among all the good processes.

Step 4: Embedding



Embedding
passed features into
Top K rarest paths(features)
contains
Challenges
1 Lengths of causal paths are different
Some with 3 nodes, some with 5 nodes
2 Labels of nodes and edges are unstructured data
Different file names or executable paths
Addressing to the challenges,
applied in
1 Treat each node as a "noun"
2 Treat each edge as a "verb"
For example
B1 winword.exe write t1.txt read_by outlook.exe write 168.x.x.x
Outcome
Process:winword.exe write File:t1.txt read by Process:outlook.exe write Socket:168.x.x.x.
PV-DM model
PV-DM (Distributed Memory Model of Paragraph Vectors) is one version of doc2vec library
1 Learn with purely benign data
2 Similar causal paths are closer in numerical vector space
3 Consider order of words in path
Benign
CMD
triggers
Rather Malicious
triggers
CMD
Done embedding
Reference
System Entities
1 Process Node : Executable Path
2 File Node : File Path
3 Socket node : Source/Destination IP

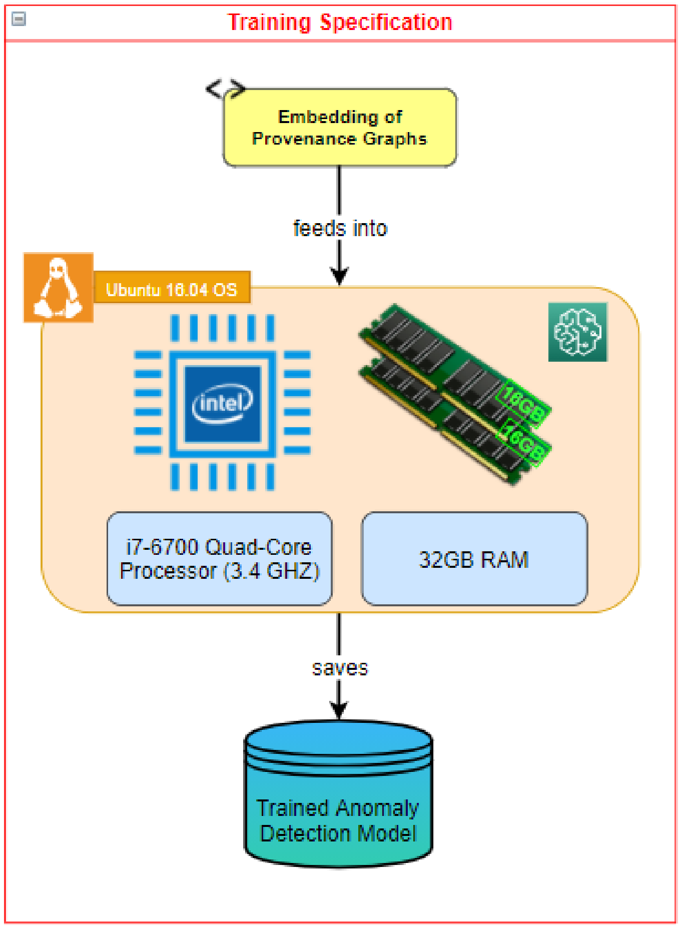
Figure 11: Embedding of ProvDetector

In this stage, the main purpose is to address to the two challenges encountered by the researcher. The ProvDetector team has spent a lot of time to find suitable embedding model to convert selected causal paths into numerical vectors space which is in a structured form for model training afterwards. With a series of evaluation, the team choose to use the PV-DM (Distributed Memory Model of Paragraph Vectors) model by *doc2vec* with obtaining the best performance for the paper's use case among all neural embedding models. One of the advantages of PV-DM model would be it takes in order of words for account.

For further information on PV-DM model, readers could refer to the Appendix of the paper at page 16.

Step 5: Anomaly Detection

The training specifications for the anomaly detection model are as below:



Training Specification
Embedding of Provenance Graphs
feeds into
Ubuntu 16.04 OS
intel
16GB
16GB
i7-6700 Quad-Core Processor (3.4 GHZ)
32GB RAM
saves
Trained Anomaly Detection Model

Figure 12: Training Specification of ProvDetector

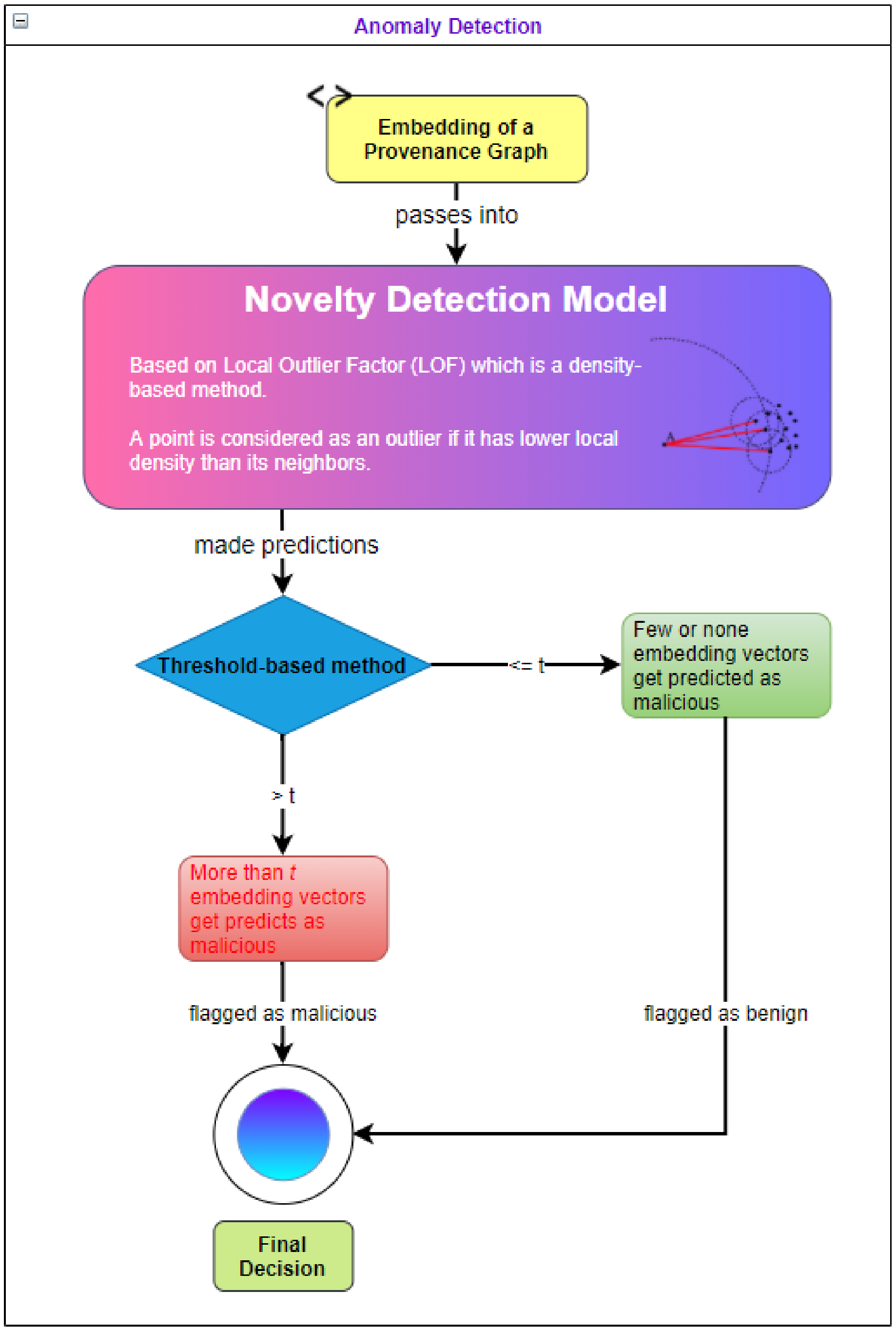

Anomaly Detection
Embedding of a Provenance Graph
passes into
Novelty Detection Model
Based on Local Outlier Factor (LOF) which is a density-based method.
A point is considered as an outlier if it has lower local density than its neighbors.
made predictions
Threshold-based method
<= t
Few or none embedding vectors get predicted as malicious
> t
More than t embedding vectors get predicts as malicious
flagged as malicious
flagged as benign
Final Decision

Figure 13: Anomaly Detection of ProvDetector

In the final stage, ProvDetector can predict and flag a provenance graph of current running program as malicious or benign with the novelty detection model which is based on Local Outlier Factir (LOF) with the control of threshold that can be configured manually.

Conclusion

From the experiment evaluation, I can conclude that the proposed defensive system to detect stealthy malware is practical and effective. According to the researcher, the PROVDETECTOR has been implemented on 306 hosts of an enterprise for over 3 months and evaluate with 1150 stealthy impersonation attacks and 1150 benign program instances in total. On average it achieves 0.959, 0.991 and 0.974 of precision, recall and F1-score respectively which represents that the proposed architecture is really useful and the hypothesis and intuition from researchers have been proven.

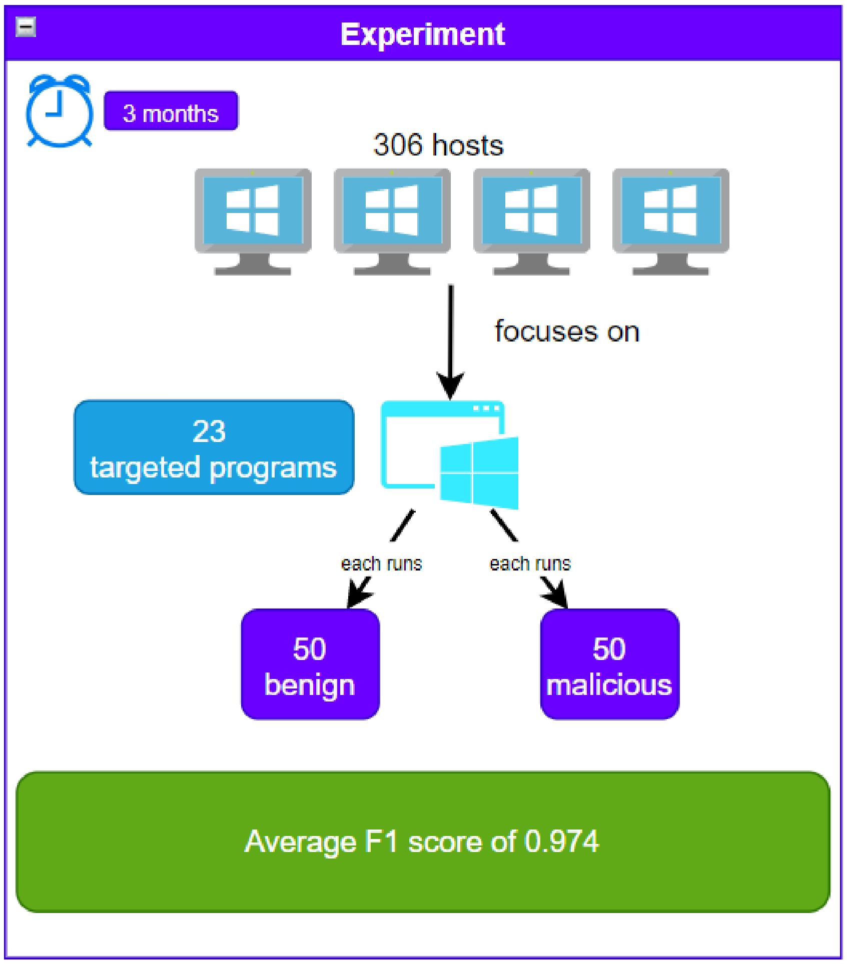


Figure 14: Experiment and Evaluation of ProvDetector

The article is considered easy-to-be-understand on the overall figure of how the system works to differentiate between benign and malicious behaviour among all the process instances of a program. However, the embedding step of the proposed malware defence system is considered the core technical value to me. Since there's no source codes implementation provided, it's a bit hard to understand for the embedding part which make use of PV-DM neural embedding model. This paper does require readers to have at least intermediate knowledge in Machine Learning and graph theory to be able to fully understand how the ProvDetector works. Hence, readers are suggested to have some findings on Local Outlier Factor algorithm and neural embedding model beforehand. In a nutshell, this paper is considered quite amazing and knowledgeable to me in recognizing stealthy malware and provenance analysis approaches in detecting them.

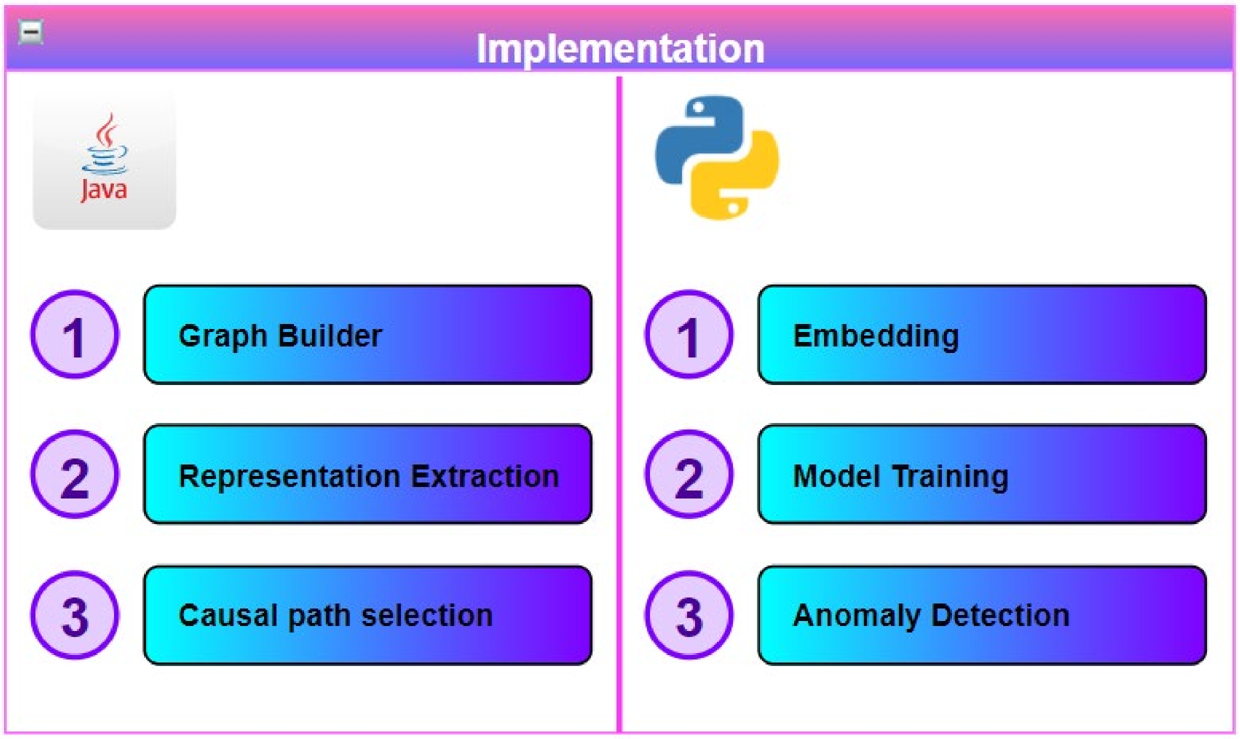
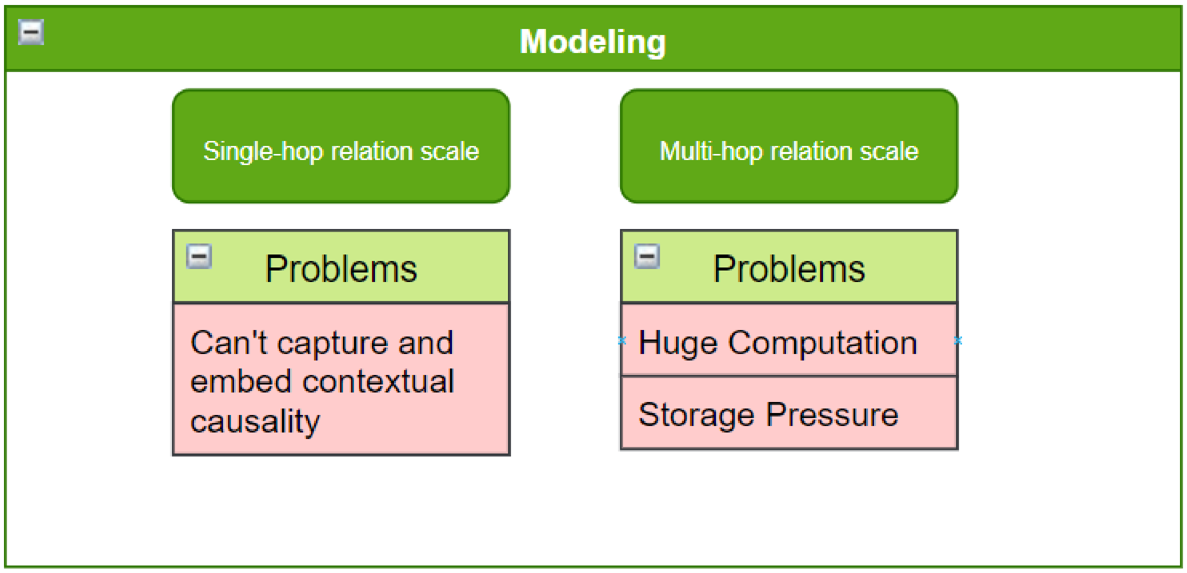


Figure 15: Components Development of ProvDetector based on Java and Python

Although the PROVDETECTOR focuses on 23 common programs and trained their respective model for detection, one of the advantages of the system is once the data is collected and model has been trained after gone through a series of process such as features extraction and embedding, it is considered one-off effort and there's no need to retrain the model every time running the stealthy malware detection on that particular program unless we want to improve the accuracy of the trained model. In another word, the trained model has always been loaded and no need re-train on the spot.

The other benefit of ProvDetector is the ability to detect previous unseen attacks (zero-day attack) as the system only trained with benign data and since it's multi-hop relation scale which is considered another advantage of the system.



The provenance data collection is considered reasonable amount of overhead compared to heavy-weight dynamic analyses although ProvDetector is based on multi-hop relation scale. Nevertheless, one-hop approach is not looking back on previous steps or subsequent following steps to link up all of them to have an overall big picture in finding malicious logic. In some attacks, every step in the attack looks normal by itself towards single-hop relation-based malware detection system.

Now, let's explore the disadvantages of the ProvDetector. First, ProvDetector cannot deal with attacks performed using implicit flows which sabotaged and modified the system audit logs. Also, ProvDetector would require time in building provenance graph for a new encountered program before it can perform anomaly detection and it's one of its flaws which is cannot perform real-time detection on stealthy malware yet. Then, one of the limitations of ProvDetector is not able to detect the traditional malware or worms. However, this is not the focus of the system.

For the future work, the researchers team planned to implement Natural Language Processing (NLP) algorithm in neural embedding and aids in the feature extraction as well for the PV-DM model.

In my opinion, I think the researcher can work together with current market HIDS or antivirus provider to integrate their great system into antivirus program as one of the features to provide fully protection to all of the PC users in knocking down all kind of computer viruses. Hooray, the solution is able to provide a safer environment for all targeted, hurt Windows victims so far.

Thank you.