ISSN 2278-3091
Volume 9, No.5, September - October 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse198952020.pdf
https://doi.org/10.30534/ijatcse/2020/198952020

Detecting behavior of malware using MITRE ATT&CK

Vu Ngoc Son[1], Tisenko Victor Nikolaevich[2],Le Duong Anh Tuan[3], Nguyen Tung Lam[4], Pham Thi Thuong[5], Dong Xuan Anh[6]
[1,3,4,5,6]Information Assurance dept. FPT University, Hanoi, Vietnam, sonvnse04460@fpt.edu.vn , TuanLDAHE130230@fpt.edu.vn, LamNTHE130587@fpt.edu.vn, ThuongPTSE05856@fpt.edu.vn, AnhDXSE06086@fpt.edu.vn
[2]Department Quality Systems, Peter the Great St. Petersburg Polytechnic University, Russia, St.Petersburg, Polytechnicheskaya, 29, v_tisenko@mail.ru

ABSTRACT

The trend of cyber-attacks through end-users is currently widely used by attackers. One of them is the attack form by spreading malware on users' computers to steal data or escalate the privilege higher. The attack technique by spreading malware is a dangerous attack method that difficult to detect and prevent. Therefore, the task of detecting and alerting users or the system about signs of malware is very necessary today. The current studies and proposals on malware detection are usually based on two main methods: using signs, and analyzing abnormal behavior based on machine learning or deep learning machine techniques. In this paper, we propose a method of detecting malware on users' computers using the Event ID profile analysis technique. Event IDs that are signs and behaviors of malware are tracked and collected on the operating system kernel of the workstation. The difference between our research with other published methods is that we collect malware behavior directly on processes on the operating system kernel, not through virtualized systems. Therefore, even though the malware uses techniques to conceal itself, their behavior when executed is recorded by the operating system kernel. Based on those processes, we use Event ID analysis techniques to conclude the existence of malware in the system.

Key words: machine learning, malicious URLs, TF- IDF.

1. INTRODUCTION

The document [1] presented the concept, characteristics, and classification of malware. In the study [2], the authors statistic and explain why spreading malware is becoming the trend of current cyber-attack types. Therefore, the problem of detecting malware in the system has become a problem that needs to be concerned today. The study [1] listed two main current methods used to detect malware that are detection techniques based on signs and based on machine learning. However, one common characteristic of these methods is using the method of extracting behaviors and signs of malware based on sample datasets. These datasets are built based on virtualization tools or network monitoring and static analysis tools. For virtualization tools, studies often use the Sandbox tool [3] to execute and extract malware behavior. The disadvantage of Sandbox tools is that they only record behaviors in a certain time, so they will not be able to fully statistic malware behavior. For datasets collected during static analysis, only detect anomalies when malware has spread and connected to steal data. Therefore, these traditional approaches are always overtaken by malware. In addition, the tendency of malware spreading attacks is the attack on the user in order to escalate privileges into the system. From the above reasons, in this paper, we propose a method to detect malware on Workstation. Our proposed method will directly monitor and detect malware on Workstation based on Events collected on the operating system kernel using sign set and behavior analysis.

2. RELATED WORKS

Currently, the theoretical research related to malware detection on Workstations is very limited. Concerning the problem of detecting malware on a workstation, besides some antivirus software, there are some Endpoint Detection & Response (EDR) support products. The EDR product has the function of detecting and tracking anomalous incidents on Enduser in order to give out the incident response scenarios. According to the publication [4], there are some solutions and products of EDR as follows: Trend Micro EDR Apex One product [5] has the ability to automatically detect and prevent many threats on the endpoint as possible, without any manual user intervention. Apex One also detect and prevent exploitation of vulnerabilities in the operating system before threats gain access to the endpoint with virtual patches that are constantly updated with artificial intelligence from Trend Micro's Zero Day Initiative. Similarly, the Palo Alto Networks Traps product [6] of Palo Alto prevents threats on endpoints, coordinates with cloud security and network security to prevent cyber-attacks. Traps prevent the execution of the malicious executables, DLLs, and Office files with many methods of preventing, reducing attack surfaces, and increasing the accuracy of malware blocking. This approach prevents known and unknown malware from infecting the endpoint by combining some methods: WildFire threat

intelligence; Local analysis via machine learning; WildFire inspection and analysis; Granular child process protection; and Periodic scanning for dormant malware. Kaspersky EDR [7] can continuously monitor and analyze anomalies, suspicious processes on employee workstations, and response to threats in both manual mode and passive mode. In addition, Kaspersky EDR enables the control of incidents on endpoints of the network, detecting malware and unrecognizable unauthorized behaviors at the level of network protection, and responding quickly to them. There are also a number of other solutions such as VMware Carbon Black EDR [8], Falcon [9], Malwarebytes Endpoint Detection and Response [10].

3. PROPOSING DETECTION MODEL

3.1. Model architecture

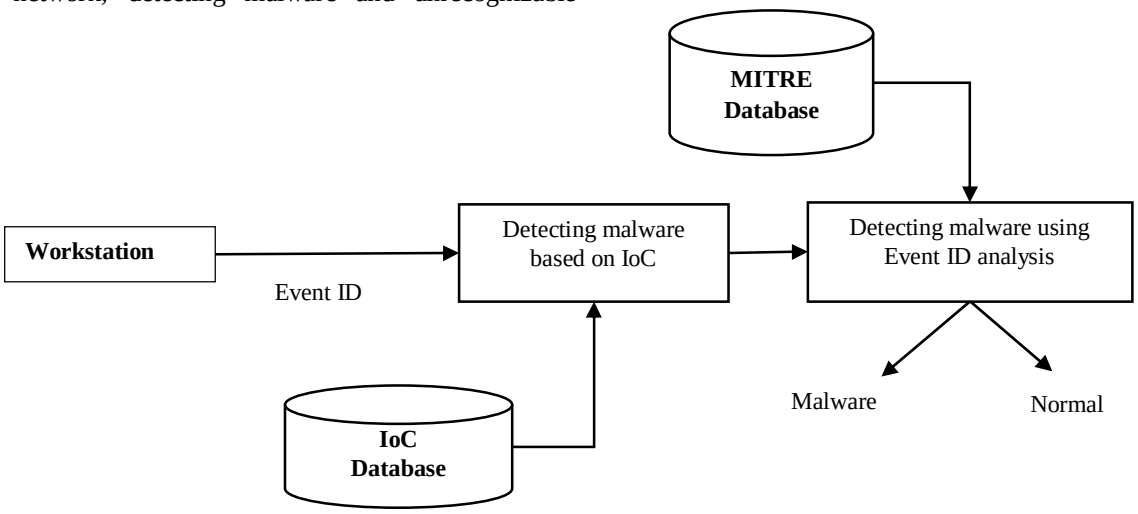


Fig. 1. Proposed model of malware detection on the workstation

Figure 1 shows our proposed model of malware detection on the workstation. The main components in our proposed model include:

- **Workstation** is the user's computer that needs to monitor. Accordingly, in this paper, to collect processes on the operating system kernel of a user's machine, we need a tool to collect, process, and transfer processes to the analysis center. Thus, each user using different operating systems will have different collection and processing tools. We will collect processes on the Windows and Ubuntu operating system kernels by using the Sysmon tool [11].
- **Event ID** is process collected by the Sysmon tool on the operating system kernel. According to the document [11], the Sysmon tool collects a total of 21 processes from the operating system kernel. Table 1 below describes the 21 processes that were collectedCrawling data: This module collects the URL data from different sources.

Table 1. List of processes collected on the operating system kernel by Sysmon tool

No.	Name and notation of Event	Description
1	Event ID 1: Process creation	This event shows additional information about a newly created process. The full command line returns the execution process context. Besides, this event has the ProcessGUID and HashType field.
2	Event ID 2: A process changed a file creation time	This event helps to track the real creation time of a file. When a process modified explicitly a file creation time, this event is registered. For example, to make a backdoor look normal, attackers may change its file creation time.
3	Event ID 3: Network connection	This event contains logs of TCP/UDP connections. By using ProcessId and ProcessGUID fields, a connection and a process are linked together.
4	Event ID 4: Sysmon service state changed	This event logs the Sysmon service state (started or stopped).
5	Event ID 5: Process terminated	When a process terminates, this event logs UtcTime, ProcessGuid, and ProcessId of the process.
6	Event ID 6: Driver loaded	This event records information about a driver being loaded, hashes and signature.
7	Event ID 7: Image loaded	This event records the module loading process, Hashes, and signature.
8	Event ID 8: CreateRemoteThread	When a process creates a thread in another process to inject code and conceal, the event logs the source and target process and information on the code that will be run in the new threads.

9	Event ID 9: RawAccessRead	This event logs the source process and target device when a process reads from the drive using the \\.\ denotation. Malware often uses this technique to filter data of locked files and avoid auditing tools.
10	Event ID 10: ProcessAccess	This event is recorded when a process opens another process. This event is used for detecting malicious tools that read process memory data. However, enabling it can generate significant amounts of logging.
11	Event ID 11: FileCreate	This event logs when a file is created or overwritten. This event is used in autostart location monitoring.
12	Event ID 12: RegistryEvent (Object create and delete)	This event maps to operations that create and delete the Registry key and value.
13	Event ID 13: RegistryEvent (Value Set)	This event records the Registry values (with type DWORD and QWORD) in order to identify Registry value modifications.
14	Event ID 14: RegistryEvent (Key and Value Rename)	This event records the new name of the renamed key and value.
15	Event ID 15: FileCreateStreamHash	When a named file stream is created, this event logs the hash of contents of both unnamed and named streams.
16	Event ID 17: PipeEvent (Pipe Created)	When a named pipe is created, this event is generated. This pipe is often used by Malware.
17	Event ID 18: PipeEvent (Pipe Connected)	When between a client and a server has a named pipe connection, this event logs.
18	Event ID 19: WmiEvent (WmiEventFilter activity detected)	This event records the filter name, expression, the namespace of WMI when a WMI event filter is registered.
19	Event ID 20: WmiEvent (WmiEventConsumer activity detected)	This event records the registration of WMI clients.
20	Event ID 21: WmiEvent(WmiEventConsumerToFilter activity detected)	When a client uses a filter, this event logs the name and filter path of the client.
21	Event ID 22: DNSEvent (DNS query)	This event is created when a DNS query is executed by a process, whether the result is successful or fails, cached or not. This event is available for win 8 and later.
22	Event ID 255: Error	This event is created when a system overloads, a task is not performed, or having a bug in the Sysmon.

- **IoC database** is a database about Indicators of compromise (IoC) of known malware including IP blacklist, malicious URL, C&C server, Virus signatures, MD5 hashes, botnet command, etc.
- **Detecting malware based on IoC**: This is a function block that is responsible for comparing each collected Event ID with the malware's IoC database. The result of the Event ID comparison show which Event is malicious.
- **MITRE database** is a database that we built on the collected malware samples. Accordingly, we will experiment with malware samples in a virtualized environment to obtain Event ID groups as well as behaviors of each the Event ID groups.
- **Detecting malware using Event ID analysis:** Based on the Event profiles we built as well as the weights of each Event in the Event profile, we will evaluate these Event profiles using the Event ID analysis technique.

  From Figure, 1 we propose the operating procedure of the system as follows:

- **Step 1:** Collecting and processing Event ID on Workstation. As shown in Section 3.1, the malware detection system is responsible for detecting and monitoring the signs and behaviors of malware based on the processes that they recognized in the operating system. To perform the task of collecting and extracting these processes, we will install and configure 2 main tools: Sysmon and Linux Auditing System [12] corresponding to 2 current popular operating systems as Windows and Linux. These tools collect the processes logged by the operating system and transfer them to the processing and monitoring center.
- **Step 2:** Detecting malware based on Event ID using the IoC database. After the processes are collected and transferred to the processing and monitoring center, the system will check the Event IDs based on the IoC database of previously collected malware. If an Event ID is found in the IoC database, the system will immediately alert the user to the existing malware. If no anomalies are found in the Event ID,

the system will transfer monitoring of this Event ID to the next step.

- **Step 3:** Detecting malware based on Event profile analysis. At this step, to check whether the newly born Event ID is an abnormal event or not, we perform the following 2 main stages:
  - **Stage 1:** Building behavior profiles of Events. Accordingly, the Events generated from the operating system will be checked and evaluated to know whether an Event is related to the processes that were previously collected or not. If true, the Event will continue to be appended to the behavior profile of the previous Events. If false, the Event will be built into a new Event behavior profile.
  - **Stage 2:** Detecting malware based on analyzing the behavior of processes using the MITRE attack. Accordingly, based on Event profiles collected in stage 1, we will evaluate each Event profile to conclude about the signs of malware in the system. In order to accomplish this task, in this paper, we will use the ruleset of the MITRE attack [13]. Details of the process of building rules and methods of detecting abnormal behaviors of malware will be described in section 3.2 of the paper.

3.2. Malware detection method using MITRE attack

3.2.1. Introduction to MITRE attack

In fact, malware will have many different ways and processes to hide and bypass surveillance systems. In recent studies, our team often focuses and extracts the properties and behavior of the malware based on data collected from the sandbox. However, we realize that collecting malicious behavior will not be able to guarantee all of their behavior is fully documented. Also, quick detection time is not guaranteed. Therefore, in this project, we will not directly extract anomalies of malware based on data analysis and evaluation from the sandbox. Instead, we use MITRE ATT&CK to define and behave malicious behavior.

MITRE ATT&CK is a knowledge base of attack tactics and techniques gathered on observations of actual attacks. Each attack tactic represents a target or a certain stage in the attack (How to escalate privileges, how to fix malware on the victim's system). And attack techniques, which describe methods to achieve those goals. MITRE ATT&CK currently has 11 popular attack tactics including Initial Access, Execution, Persistence, Privilege Escalation, Defense Evasion, Credential Access, Discovery, Lateral Movement, Collection, Exfiltration, Impact [13]. Table 2 below lists tactics of the MITRE attack.

Table 2. List of all MITRE ATT&CK Tatic

No.	Attack tactic	Description
1	Initial Access	This is the first step for a hacker to get into your network.
2	Execution	After access to your network, attackers will run malware to gain more data on your system.
3	Persistence	This step is usually used to maintain the control of the hacker after access to the information system. Such as a backdoor.
4	Privilege Escalation	Hackers try to gain more permission for deeper access to sensitive data.
5	Defense Evasion	Attackers avoid being detected.
6	Credential Access	Attackers are trying to steal login credentials such as username and password.
7	Discovery	The adversary is trying to understand network elements. Such as how many computers are in the network?
8	Lateral Movement	The adversary is trying to move through your environment such as, from this computer to another computer.
9	Collection	In this tactic, the attacker is trying to gather data that they are interested in.
10	Command and Control	The adversary is trying to remote control a hacked computer system.
11	Exfiltration	All techniques of this tactic are dangerous, the attacker is trying to steal the data.
12	Impact	Attackers are trying to make data corruption. Such as delete or modify valid data and even destroy all systems.

3.2.2. The principle of building the ruleset based on MITRE

Table 3 below lists some main components of a rule that is built from MITRE attack.

a) The structure and components of the ruleset

Table 3. Some components of the ruleset

Elements	Type	Description
Title	String	A brief title for the rule that should contain what the rules is supposed to detect
Description	String	Short description of malicious rules and practices that can be detected
References	Array	Reference to the source derived from the rule. These could be blog articles, technical articles, presentations, or even tweets.
Status	any	Declare the state of the rule
Log Source	record	This section describes the log data that is applied to detection. It describes the log source, platform, application, and type required when detected.
Detection	record	A set of search-identifiers that represent searches on log data
Condition	any	The condition is the most complex part of the specification and will be subject to change over time and arising requirements. In the first release, it will support the following expressions.
Fields	array	A list of log fields that could be interesting in further analysis of the event and should be displayed to the analyst.
Level	any	The level field contains one of four string values. It describes the criticality of a triggered rule. While low and medium level events have an informative character, events with high and critical levels should lead to immediate reviews by security analysts.
Tags	array	A rule can be classified by tags

a) ***Definition of components of the ruleset***

1. Title

A brief title for the rule that should contain what the rules is supposed to detect

2. **Description**

Short description of malicious rules and practices that can be detected

3. References

Reference to the source derived from the rule. These could be blog articles, technical articles, presentations, or even tweets.

4. Status

Declare the state of the rule:

- stable: the rule is considered stable and can be used in production systems or control panels.
- test: a quasi-stable rule might require some tweaking.
- experimental: a test rule can lead to false or noisy results, but can also identify events.

5. Log Source

This section describes the log data that is applied to detection. It describes the log source, platform, application, and type required when detected. It includes three properties that are automatically evaluated by the converter and some optional elements.

- **category** - examples: firewall, web, antivirus. The "category" value is used to select all log files that are written by a certain product group, such as a firewall or web server logs. Automatic conversions use the keyword as a selector for multiple metrics.
- **product** - examples: windows, apache, checkpoint fw1. The "product" value is used to select all log outputs of a certain product, e.g. all Windows Eventlog types including "Security", "System", "Application" and the new log types like "AppLocker" and "Windows Defender".
- **service** - examples: sshd, applocker. Use the "service" value to select only a subset of the product's log, such as "sshd" on Linux or the "Security" event log on Windows systems.
- **Definition**: The "definition" can be used to describe the log source, including some information about the granularity of the log or the configurations that must be applied. It is not evaluated automatically by the converter but gives helpful advice to the reader on how to configure the source to deliver the necessary events used in detection.

6. Detection

A set of search-identifiers that represent searches on log data:

- ***Search-Identifier:*** A definition that can consist of two different data structures - lists and maps
- ***General:***

  - All values are treated as case-insensitive strings
  - Can use wildcard characters '*' and '?' in strings

- Wildcards can be escaped with \, e.g. *. If some wildcard after a backslash should be searched, the backslash has to be escaped: *.
- Regular expressions are case-sensitive by default

- ***Lists:*** The list contains strings that apply to the full log message, and are reasonably linked with 'OR', for example:

- Match on 'EvilService' **OR** 'svchost.exe -n evil':

Detection:
Keywords:
-EvilService
- svchost.exe -n evil

- ***Maps:*** Maps (or dictionaries) consist of key/value pairs, in which the key is a field in the log data and the value of a string or integer value. Lists of maps are joined with a logical 'OR'. All elements of a map are joined with a logical 'AND', for example:

Matches on **EventLog** 'Security' **AND** (**EventID** 'X' **OR EventID** 'Y'):

Detection:
Selection:
EventLog: Security
EventID:
- X
-Y
condition: selection

Matches on **EventLog** 'Security' **AND EventID** 'X' **AND TicketOptions** 0x40810000 **AND TicketEncryption** 0x17:

Detection:
Selection:
-EventLog: Security
EventID: X
TicketOptions: '0x40810000'
TicketEncryption: '0x17'
condition: selection

- ***Special Field Values:*** There are special field values that can be used.

- An empty value is defined with ''
- A null value is defined with null
- OBSOLETE: An arbitrary value except null or null cannot be specified with non-null anymore

- ***Value Modifiers:*** Values contained in a rule can be modified with value modifiers. The value modifier is added after the field name with the pipe character | as a separator and can also be strung, for example, the field name | mod1 | mod2: value. The value modifier is applied in the order given the value.

- ***Currently Available Modifiers***

- endswith: The value is expected at the end of the field's content (replaces e.g. '*\cmd.exe')
- startswith: The value is expected at the beginning of the field's content. (replaces e.g. 'adm*'

7. Condition

The condition is the most complex part of the specification and will be subject to change over time and arising requirements. In the first release, it will support the following expressions.

- Logical **AND/OR**

keywords1 or keywords2

- 1/all of search-identifier. Same as just 'keywords' if keywords are defined in a list. X may be:

- 1 (logical or across alternatives)
- All (logical and across alternatives)

Example: '**all of the keywords**' mean that all items of the list keywords must appear, instead of the default behavior of any of the listed items.

- 1/all of them: Logical OR (1 of them) or AND (all of them) across all defined search identifiers. The search identifiers themselves are logically linked with their default behavior for maps (AND) and lists (OR). Example: 1 of them means that one of the defined search identifiers must appear.

- 1/all of search-identifier-pattern: Same as *1/all of them*, but restricted to matching search identifiers. Matching is done with * wildcards (any number of characters) at arbitrary positions in the pattern. Examples:**1of selection* and keywords**

any of selection* and not filters

- Negation with 'not'

keywords and not filters

- Brackets

selection1 and (keywords1 or keywords2)

- Operator Precedence (least to most binding)

- or
- and
- not
- x of search-identifier

If multiple conditions are given, they are logically linked with OR.

8. Fields

A list of log fields that could be interesting in further analysis of the event and should be displayed to the analyst.

9. Level

The level field contains one of four string values. It describes the criticality of a triggered rule. While low and medium level events have an informative character, events with high and critical levels should lead to immediate reviews by security analysts.

- **low**: Interesting event but rarely an incident. Low events are relevant in high numbers or combined with others. A security analyst has to review the events and identify anomalies or suspicious indicators. Use them in a dashboard panel, e.g. in the form of a chart.
- **medium**: The relevant event that should be reviewed manually on a more frequent basis. A security analyst has to review the events and identify anomalies or suspicious indicators. List the events in a dashboard panel for manual review.
- **high**: The relevant event that should trigger an internal alert and requires a prompt review.
- **critical**: The highly relevant event indicates an incident. We recommend critical events for immediate response actions and external notifications (E-Mail, Ticket).

10. Tags

A rule can be classified by tags. Tags should generally conform to the following syntax:

- Character set: lower case, underscore and hyphen
- There are no spaces

Cards have a namespace, dots are used as separators. For example. attack.t1234 refers to technique 1234 in the namespace attack; Namespaces can also be nested.

b) **The architecture of the ruleset**

```
" Field name            Data type
title:
      type:          str
      length:
         min: 1
         max: 256
description:          str
references:
      type:          arr
      contents:             str
status:
      type:          any
      of:
         - type:             str
           value: stable
         - type:             str
           value: testing
         - type:          str
           value: experimental
logsource:
      type:          rec
      optional:
         category:          str
         product:       str
         service:       str
         definition:          str
```

```
detection:
      type:          rec
      required:
         condition:
            type:          any
            of:
               - type:          str
               - type:          arr
                 contents:       str
fields:
      type:          arr
      contents:             str
level:
      type:          any
      of:
         - type:             str
           value: low
         - type:             str
           value: medium
         - type:             str
           value: high
         - type:             str
           value: critical
tags:
      type:          arr
      contents:             str
```

3.2.3. Principle of detecting abnormal behavior of malware using MITRE attack

To detect malware based on its abnormal behavior by using the MITRE attack, we will focus on building its behavior profiles and then compare and analyze these behavior profiles. The principle of detecting malware by the MITRE attack technique is presented in algorithm 1 below.

Algorithm 1: detection_operator

```
Function detection_operator():
   selections = {}
   For each key, value is in detection field:
      if key is not ‘condition filed’
         result = Check selection_operator is True or False
                                  Put the result into a dictionary
selections[key] = result
      else:
         Get condition
   For each key, value in selections:
      if key in condition:
         Alter key in condition is value of each selection
      Return condition
```

The selection_operator function will be responsible for checking whether the selection is True or False and returning the result to check against the condition, the content inside the selection is a series of conditions according to the rule if dictionaries - condition AND, If the list - condition OR.

Algorithm 2: selection_operator

```
Function selection_operator():
            If selection is lits:
              list_selection = []
       For each element inside the selection:
          result = selection_operator (element)
   //Recursively call function A again until get the result
from the leaf          Add result to list_selection
          Return check or_operator(result)
    If selection is dictionaries:      list_selection = []
       For each element inside the selection:
          If key is 'condition field':
                    continue
                If value is list
                    result is return of check_many(key,
value)
                    Add result to list_selection
                else:
                    result is return of check_sigle(key,
value)
              Add result to list_selection
          Return check and_operator(result )
```

This function executes in case a key has a list of many different values, in order to be able to check, you need to take each value inside and check with each value in the field of the log message, the check_many function will get Each value entered into the check_single function waits for the return result, as mentioned in the form of a key with multiple values, the or_operator function will perform the test with the OR condition.

Algorithm 3: selection_operator

```
Function check_many(key, values)
                        List_value_check = []
                        For each value inside the values:
                            result is return of
                    check_sigle(key, value)
                            Add result to
                    List_value_check
                        Return check or_operator(result)
```

The check_single function references each value in the rule in turn with the reference value having the same key as the field from the message, each key has a way to compare the value from the log message like contains, endswith, startswith or nothing, the test and return results are done in the simple_operator function.

Algorithm 4: simple_operator function

```
Function check_sigle(key, value)
   elements = number return by split key with "|"
            field = element[0]
   If elements == 1:
      Return simple_operator(message[field], value, "=")
   condition = elements[1]
   Return simple_operator(message[field], value, condition)
Function simple_operator(left, right, operator)
   If operator is "=":
      If operator is String:
          if appears "*" at the beginning and end:
             simple_operator(left, right[1:-1], "contains")
          if appears "*" at the end:
             simple_operator(left, right[1:], "startswith")
          if appears "*" at the beginning:
             simple_operator(left, right[1:], "endswith")
   If left is None:
      Return False
   If operator is "endswith":
      Return left endswith right
   if operator is "startswith":
      Return left startswith right
   if operator is "contains":
      Right in Left
```

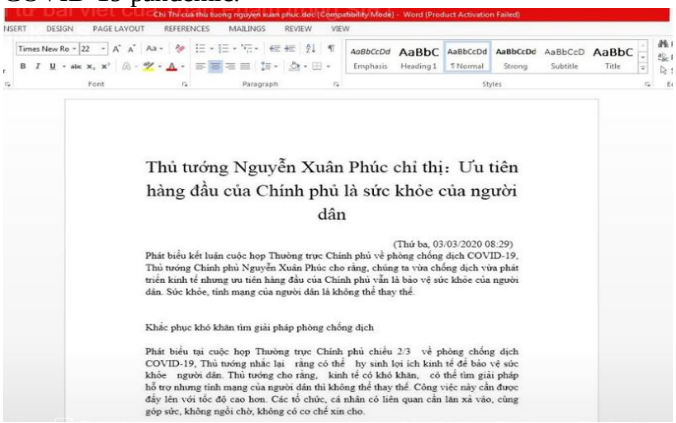
4. Experiments and evaluations

4.1. Experimental ruleset

In this paper, we collect samples of malware in [14]. Here we obtained about 4,847 behavior profiles of different types of malware including Emotet, Remcos, Lokibot, Njrat, etc. Combining the behavior of the above malware types with MITRE attack, we built and implemented this rule. At the same time, we collected more ruleset also built according to a strategy of the MITRE attack that is Sigma [15]. Accordingly, the rules at Sigma provide 68 different rulesets:

4.2. *Example of malware detection results*

Now recently, the COVID-19 pandemic has been complicated in many countries, some hacker groups have taken advantage of this situation to launch and conduct targeted cyber-attack campaigns on agencies and organizations in the world, including Vietnam. Specifically, in recent days, hackers have spread malware via email with an attached word file with the title "Chi Thi of Thu tuong nguyen xuan phuc.lnk" which disguised the Prime Minister's announcement about the COVID-19 pandemic.


Thủ tướng Nguyễn Xuân Phúc chỉ thị: Ưu tiên hàng đầu của Chính phủ là sức khỏe của người dân

(Thứ ba, 03/03/2020 08:29)

Phát biểu kết luận cuộc họp Thường trực Chính phủ về phòng chống dịch COVID-19, Thủ tướng Chính phủ Nguyễn Xuân Phúc cho rằng, chúng ta vừa chống dịch vừa phát triển kinh tế nhưng ưu tiên hàng đầu của Chính phủ vẫn là bảo vệ sức khỏe của người dân. Sức khỏe, tính mạng của người dân là không thể thay thế.

Khắc phục khó khăn tìm giải pháp phòng chống dịch

Phát biểu tại cuộc họp Thường trực Chính phủ chiều 2/3 về phòng chống dịch COVID-19, Thủ tướng nhắc lại rằng có thể hy sinh lợi ích kinh tế để bảo vệ sức khỏe người dân. Thủ tướng cho rằng, kinh tế có khó khăn, có thể tìm giải pháp hỗ trợ nhưng tính mạng của người dân thì không thể thay thế. Công việc này cần được đẩy lên với tốc độ cao hơn. Các tổ chức, cá nhân có liên quan cần xả vào, cùng góp sức, không ngồi chờ, không có cơ chế xin cho.

Figure 2. Contents of the malicious file

Table 4. Details of the malware used

Malware file	bbbeb1a937274825b0434414fa2d9ec629ba8 46b1e3e33a59c613b54d375e4d2.rar
Hash	60C89B54029442C5E131F01FF08F84C9

File format	After extracting the rar file obtained the file "Chi Thi cua thu tuong nguyen xuan phuc.lnk"
URL to download malware	https://app.any.run/tasks/dd877b4d-8b36-48c0-af07-ce37fd9fee7b/

After conducting a test run of this malware, we obtained the progress tree of the malware as shown in Figure 3.

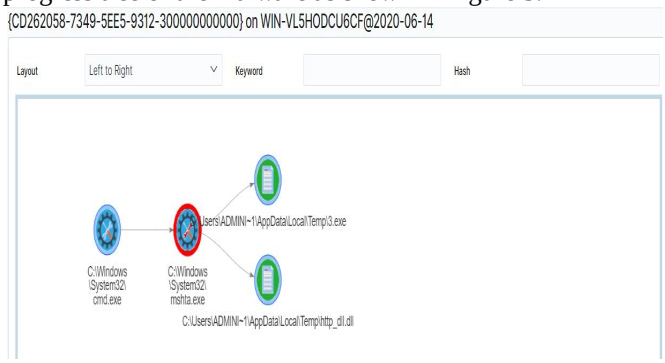


Figure 3. Behavior profile of malware

When checking the details of this event, we notice that after running the malware file, the malware created 2 new files including 3.exe and http_dll.dll at the path C\users\Admin~1\AppData\Local\Temp\ as shown in Figure 4.

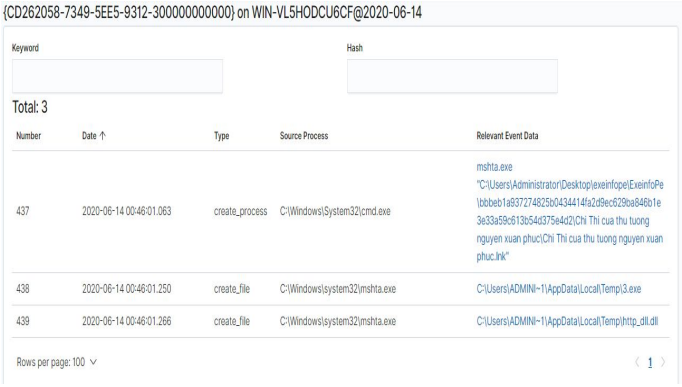


Figure 4. Information about the malware process that creates 02 files

When checking the details of the 3.exe file, we found that the file 3.exe created a registry and called unsecapp.exe file as shown in Figure 5.

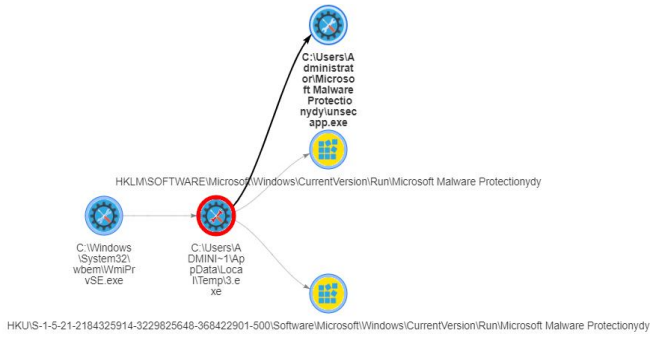


Figure 5. Details for the file 3.exe

The details about the event of the 3.exe file show clearly the file's Hash code as well as the parent process that called this file.

Figure 6. Information about the event of 3.exe file

Thus, through collecting and monitoring information about Event IDs from Sysmon, we succeeded in building behavior profiles as well as detecting anomalies of malware based on obtained behaviors. Obviously, without using this behavior analysis technique and Miter attack, the system is very difficult to detect the signs of malware.

5 CONCLUSION AND FUTURE DIRECTION

Detecting and classifying malware has been a current very necessary task. Meanwhile, problems related to detecting malware using behavior analysis techniques and machine learning algorithms have yielded good results. However, this approach also has the disadvantage that is requiring large amounts of data to train. In this paper, based on malware behaviors and the strategy of the MITRE attack, we have partially solved the disadvantages of the malware detection approach using machine learning. The Malware detection approach based on the MITRE attack that proposed in our research has great coverage and is effective when they are able to detect malware in real-time, as soon as the malware starts spreading. In the future, we will conduct research and propose additional malware detection methods based on processes using graph analysis techniques.

REFERENCES

[1] Alireza Souri, Rahil Hosseini. **A state□of□the□art survey of malware detection approaches using data mining techniques**(2018) 8:3. pp 1-22. https://doi.org/10.1186/s13673-018-0125-x.

[2] YANFANG YE, TAO LI, DONALD ADJEROH, S. SITHARAMA IYENGAR. **A survey on malware detection using data mining techniques**. *ACM Comput. Surv.* Vol, 50, No 3, Article 41 (June 2017), 40 pages. DOI: http://dx.doi.org/10.1145/3073559.

[3] IMPORTANT INFORMATION REGARDING SANDBOXIE VERSIONS. https://www.sandboxie.com/. [Last accessed 26 August 2020]

[4] Endpoint Detection and Response Solutions Market- https://www.gartner.com/reviews/market/endpoint-detection-and-response-solutions. [Last accessed 26 August 2020]

[5] Endpoint Security with Apex One Endpoint security redefined. https://www.trendmicro.com/en_us/business/products/user-protection/sps/endpoint.html [Last accessed 26 August 2020]

[6] Palo Alto Networks Traps Endpoint (EDR). https://paloaltofirewalls.co.uk/palo-alto-traps-endpoint/ [Last accessed 26 August 2020]
[7] Kaspersky Endpoint Detection and Response- https://www.kaspersky.com/enterprise-security/endpoint-detection-response-edr [Last accessed 26 August 2020]
[8] VMware Carbon Black EDR - https://www.carbonblack.com/products/edr/ [Last August 26 February 2020]
[9] Falcon Insight: EDR -https: //www.crowdstrike .com / endpoint-security-products / falcon-insight-endpoint-detection-response /[Last accessed 26 August 2020]
[10] https://www.malwarebytes.com/business/endpointdetectionresponse/ [Last accessed 26 August 2020]
[11] Sysmon v10.42. https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon [Last accessed 26 August 2020]
[12] Auditd Linux Tutorial. https://linuxhint.com/auditd_linux_tutorial/. [Last accessed 26 August 2020]
[13] ATT&CK for Industrial Control Systems. https://attack.mitre.org/.[Last accessed 26 August 2020]
[14] Malware hunting with live access to the heart of an incident. https://app.any.run/ [Last accessed 26 August 2020].
[15] Neo23x0/sigma. https://github.com/Neo23x0/sigma/blob/master/tools/README.md [Last accessed 26 August 2020]