Malware Incident Handling and Analysis Workflow

Doaa Wael
Computer Emergency Readiness Team,
National Telecommunications Regulatory
Authority ,
Nile University,
Cairo, Egypt
d.wael@nileu.edu.eg

Marianne A. Azer
National Telecommunication Institute, NTI
Nile University,
Cairo, Egypt
mazer@nu.edu.eg

***Abstract*—Malware attacks are amongst the most common security threats. Not only malware incidents are rapidly increasing, but also the attack methodologies are getting more complicated. This raises the importance of being prepared with malware incident handling and analysis plan and keeping it up-to-date. In this paper, recent research in malware analysis approaches is presented. In addition, a malware incident handling and analysis workflow is proposed.**

***Keywords* — Attacks, incident handling, malware analysis workflow, malware forensics.**

I. INTRODUCTION

With the increasing use of internet and the dependency on online services, the security concerns have increased. Unfortunately, the use of security defense methodologies and appliances cannot totally prevent security incidents. Therefore, proactive and reactive actions should be taken. Malware, short for malicious software, is any software code intends to gain unauthorized access to victim computer systems, disrupt their operations, access sensitive information, or display advertising[1]. There are different types of malware such as viruses, worms, Trojans, backdoors, keyloggers, rootkits and bootkits.

There is a large numbers of new malware samples. They have reached 390,000 registered samples every day. According to [2] Malware is considered as one of the greatest external threats to most systems. This is due to its widespread damage effects and the needed efforts for recovery. Malware creation rate broke all records in 2014. More than 75 million new malware over the year before were recorded. This is 2.5 times the number of malware specimens detected in 2013, when the number of new variants created reached 30 million. Cyber-attacks on large organizations were reported, and the year 2014 was considered as one of the worst years in computer security. According to [4] 2,315,931 ransomware attacks have been blocked by Kaspersky Lab products between April 2015 and April 2016 which is an increase of 17.7% than previous year. Also number of cryptors increased from 131,111 in 2014-15 to 718,536 in 2015-16.

Through malware analysis, capabilities of malicious software can be examined in order to gain a better knowledge of the nature of security incidents. Infection can be prevented through the understanding of the process of malware infection, its effect and defensive measures. In addition, malware analysis can be useful in improving the Software Development Life Cycle (SDLC) [5]. This is done by including misuse cases derived from malware analysis.

In this paper, a survey on malware analysis recent research advancements is presented, and a malware's incident response and analysis workflow is proposed. The description of each phase's best practices and tools is also provided. The proposed malware analysis workflow can be used to handle malware incidents and go through malware analysis. The remainder of this paper is organized as follows. Section II presents the background and the state of art of malware analysis. A malware incident handling and analysis workflow is proposed in section III. Finally the conclusions and future work are presented in section IV

II. BACKGROUND

Malware incidents have several indicators; examples of basic indicators were presented in [6]. These indicators include suspicious web pages created on the web server, data transmission using unusual protocols, huge data compressed files being transmitted. This is in addition to unlikely connections between machines using native operating system's networking features and so on.

In this section, we present the work that has been done in the literature for malware detection. Malware analysis techniques can be classified into manual and automated analysis. Section A and section B present the manual and automated approaches respectively. Section C focuses on malware forensics.

A. Manual Malware Analysis

Manual malware analysis has two phases, behavior analysis and code analysis, as it is shown in Figure 1.

In behavior analysis, the malicious files are executed in an isolated monitored environment to detect the changes and interactions that have been done by malware in system processes, files and registries. Beside this, the tested machine network traffic is dumped to analyze inbound and outbound traffic.

Code analysis is the second phase in manual malware analysis. In this phase, the malware code analysis helps to gain more understanding of the malware functions, way of spreading removal techniques and systems hardening requirements. Code analysis can be performed using two approaches, static and dynamic analysis.

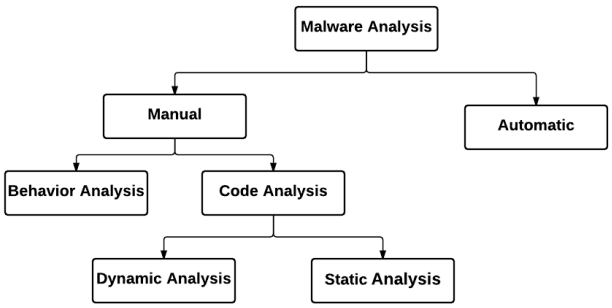

Figure1: Malware analysis approaches

1. *Static Malware Analysis:*

Static malware analysis is performed by analyzing malicious file code without executing it. It can be used to examine all possible execution paths and variable values not just those invoked during execution. Different techniques can be used to analyze the assembly code. Examples of these techniques are control flow analysis, data flow and alias analysis, symbolic evaluation, program slicing. This is in addition to reverse engineering of assembly code and security-specific analysis explained in [7]. A comparison between the most popular tools used in static analysis was provided in [7].

Static analysis suffers when dealing with polymorphism, encrypted and obfuscated malware samples. In [8], the authors have summarized anti-static analysis techniques like packers, code obfuscation, information hiding.

In [9], a new tool called Codescanner was proposed. Different types of files like (binary files, pictures, pdfs, or excerpts from memory) can be inputted to Codescanner. It detects the malware, and hidden code in files and memory, or identifies malware with anti-disassembly techniques.

2. Dynamic Malware Analysis

Dynamic malware analysis is performed by executing the malicious code's file instruction by instruction. It can be used to reveal much information about malicious code during running. Moreover, it gives the ability to modify the code and run it again. Dynamic analysis is faster and provides more information about malicious files than static analysis and can deal with encrypted and obfuscated malware samples. There are many anti-debugging techniques like detection of debugger, detection of virtual machine, detection of monitoring tools and other anti-tracking methods explained in [8].

Both static and dynamic analyses have advantages in some areas and limitations in other areas. Therefore, in order to get the complete view of malware code and have more understanding, they need to be used in a hybrid approach.

B. Automatic Malware Detection and Analysis

Due to the time consuming procedure of manual malware analysis, the need to automate the process of malware analysis has emerged to speed up the process, however they have their own limitations. For example, in automated analysis systems like sandboxes the analysis results contain a large volume of information about changes that have been made on the system during malware execution. The changes are either caused by malware or by the operating system. In addition, some malware have different features which need customization in analysis tools and analysis environment. This can't be done in the most of public sandboxes which provide general implementations. Moreover, some malwares start their activity in a specific timing after execution, so they need to vary the period of monitoring, which also does not exist in most public sandboxes.

A study of the analysis and classification of malware samples automatically was presented in [10]. The authors developed automatic analysis system by extending Cuckoo sandbox. They used distributed virtual environment for fast analysis

In order to automate the process of identifying malicious files, the authors in [11] automated the behavior analysis process by using formal method in defining the malicious behavior in form of group of tasks. In addition, a mapping from the behavior's primary tasks into patterns of system calls was presented.

In [12], the authors used cuckoo sandbox to analyze a number of samples and compare results with those of manual malware analysis of the same samples. It was found that cuckoo sandbox produces similar results in a suitable amount of time.

An automated framework for malware investigation was proposed in [13]. It is based on the integration between malware analysis and data mining clustering technique by grouping similar behavior analyzed samples in order to make investigations easy decrease the efforts and improve the present static/manual approach of analysis.

Based on data mining using extracted API calls, the authors in [14], proposed a method to detect malware. It helps in exploring the behavior of the program. An

algorithm of feature selection based on fisher score to select unique APIs was suggested to help in clustering files into malicious or benign.

In [15], the authors provided a Malicious Code Automated Run-Time Analysis framework MCARTA. It can be used to identify the highly suspicious files that should be in depth analyzed using reverse engineering tools. The new framework speeds the malware analysis process. It takes few minutes to analyze a large group of doubtful files and outputs a small group of highly suspicious files to be analyzed using reverse engineering tools.

iPanda malware analysis tool was introduced in [16]. The authors claimed that this tool has different static and dynamic malware analysis techniques which can help in evading techniques detection. It does a comprehensive analysis in malware samples and generates the analysis reports. However, the accuracy is one of the main limitations of this tool. In addition, it has limitations in advanced evading techniques and the malware sample network behavior mining.

The automatic malware analysis and the use of sandbox and machine learning in automatic analysis were discussed in [17]. Moreover the authors proposed other malware detection approach which can be used to achieve an accuracy rate more than 90%, in faster manner.

C. Malware Forensics

In the majority of malware incidents, victims detect the infection of their machine after the execution of an unknown file which deletes itself after execution, or when the original malicious files do not exist anymore for any other reason. Afterwards, analysts start to deal with malware forensics.

In [5], the authors stated that organizations need to prepare a framework for incident response and forensics investigations. The framework helps organizations in incident handling, evidence collection and analysis, incident containment and recovery. They also discussed the best practice to respond to security incidents in order to help enterprises to get rid of active malware.

A method to check if the process, which maintains suspicious connection, is malicious or not was suggested in [18]. It extracts the suspicious process image which maintains the suspicious connection. It then executes the same executable in clean virtual machine and extracts the second process image. Afterwards, it compares between the two images to find if there is any injected code.

In [19], the authors used LiveDetector & LiveSearch tools kit to forensically analyze malware attacks in Windows system.

The information in the memory data structures was used in [20] for malware detection. In order to ensure the precision, multiple memory data structures were used at the same time. Experimental results have shown that this technique has reached 98% detection rate.

III PROPOSED MALWARE INCIDENT HANDLING AND ANALYSIS WORKFLOW

As mentioned earlier, malware analysis is a time consuming processes and needs high skills. Therefore, in order to speed up and improve the malware analysis process, organized and structured steps are needed to help analysts in improving the cooperation between the team members.

Challenges related to the malware analysis procedure based on cooperation were identified in [21]. A malware analysis workflow was proposed to overcome these challenges. The workflow speeds up in-depth analysis of malware by having a parallel analysis process using the efforts of multiple analysts, Scrum was used as a process management framework. It has several features and is suitable for malware analysis as it is lightweight and flexible. Scrum provides an iterative progress with incremental results for each analyst. Therefore, if any task has been accomplished, all the analysts will know and will not spend more time or effort on it anymore.

In this section, we propose a malware analysis workflow. It is based on the workflow proposed in [21]. The authors in [21] assumed that the starting point for the analysis is an extraction from the malicious file, and mentioned that the digital forensics is out of their scope. However our proposed scheme's main objective is to cover both of malware analysis and malware incidents handling phase. The flow chart of the proposed workflow is illustrated in Figure 1.

In the following, two cases will be discussed. In the first, the analysts receive the malicious file for analysis, for example malicious spam mail attached file. In the second case, the victims have already run malicious code on their machine, so the analysts will have to create a memory image from the victim's machine and work on it.

Malware analysis is usually performed on an isolated network. The analysis can take place on virtual machines to have the chance to run multiple operating systems simultaneously. Using virtual machines, small networks can be simulated within one machine. In addition, the snapshot feature helps in preserving the system's state.

A template is prepared to be used as a baseline for analysis. This template contains operating systems which can be used to analyze different types of malwares. It also contains typical software products like web browsers, document processor, monitoring and analysis tools.

If the malware detects that it is running inside a virtual machine, the analysts transfer the analysis into physical machines and the state restoring can be handled using different tools. For example, for computer imaging, the open source FOG [22] can be used. It supports Windows, Mac OS, Linux, and even multi-boot setups.

Free linux toolkit that can be used to detect malicious software reverse engineering is called REMnux [23]. It simulates different network services using different installed tools, such as "FakeDNS"[24], "fakeMail"[25] and "inetsim"[26]. As mentioned in [21], the analysts'

machines are connected with an analysis server to enable collaboration. They host documentation and task management system.

As shown in Figure 2, the workflow starts by incident reporting. Based on the type of incident, the analyst can move on the workflow. For example, if the malicious file exists, the analyst will go through initialization, classification, preliminary analysis, in depth analysis, and evidence providence. The output in that case would be the recommendation of long term mitigation plans. If the reported incident is related to a suspicious machine, or if indicators of compromise appear on machine where the exact malicious file hasn't been revealed yet, the analyst goes through the following phases: Initialization, identification, investigation, in depth analysis and providing evidence and long term mitigation concepts phases. In the following, the phases are explained in details.

1. Initialization

In this phase, the analysts set the environment to start new analysis. A plan for the team of analysts is created, and the available incident information is discussed. Moreover communication channels are defined and analysis constraints created. In addition, the level of case confidentiality and the overall analysis goals is defined.

2. Classification

In this phase, the analysts classify the received file and detect whether it is a malicious one, or it is just a false positive, and the file is benign. This phase has the objective of reducing the time wasted in analyzing a wrong file, as malware analysis is a time consuming activity. The first step in this phase is creating malware file hash value. Then analysts can use online tools like Virus Total [27] and Jotti [28] to guess the malware's identity and have a hint on the malware family and behavior if it was known malware. The case confidentiality must be taken in consideration. Automatic malware detection techniques can be used in this phase to classify the suspicious files using techniques like data mining.

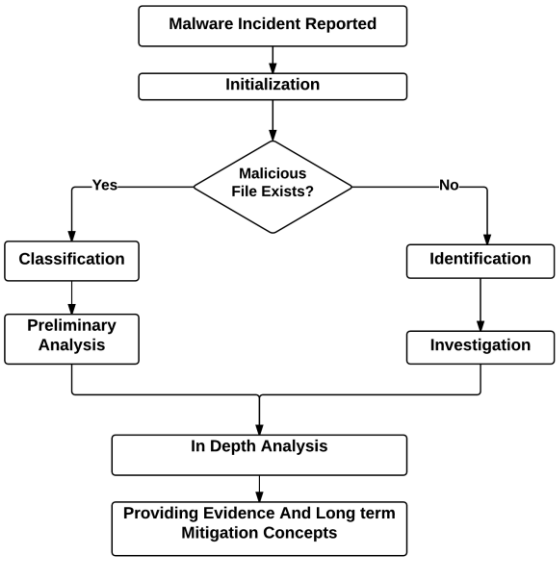

Figure2: Proposed malware incident handling and analysis workflow

3. Preliminary Analysis

In this phase, the analysts start analyzing the malicious file by exploring the file's PE header, identifying packing type, extracting strings from malicious file and going through both of file behavior analysis and memory analysis as follows:

- Explore the malicious file PE header using tools like CFF Explorer and PE studio.
- Calculate the file entropy, identify if the malicious file is packed and identify the packing type using tool like PEid.
- Explore strings in the malicious file using tools like Strings or Bintext which may reveal some information like file names, domains and packing type but this information must not be fully trusted, as it may be inserted to deceive the analysts.
- Analyze the malicious file's behavior, by executing the file on an isolated virtual machine and monitoring its behavior using tools Caputer-BAT, Regshot, WiTS, Process Explorer, Process Monitor, Autoruns, Wireshark and TCPView.
- Perform memory analysis because memory is an important place to find more information about the malicious file. Even if with the malware is packed, obfuscated or encrypted, at runtime the packed file is unpacked and reveals original file code on the memory. Memory analysis has the advantage of dealing with memory contents directly without involving the operating system. Therefore, memory analysis adds great help when dealing with rootkits as rootkits have the ability to hide their behavior from monitoring and detection tools. Memory analysis can be done by executing the malicious file on the isolated machine then dumping the memory using tools like FTK Imager then using tools like volatility, Rekall or Totalrecall to investigate the memory

image file. This will be discussed in details later in the investigations phase.

- Analyzing through automatic malware analysis, systems like sandbox may be helpful in this phase. There are free online analysis sandboxes like Anubis, threat expert and Cuckoo
- Unpacking the malicious file if it was packed. Unpacking can be performed using unpacking scripts or tools depending on the packing type identified earlier. If it doesn't work the analyst should unpack it manually with an understanding of the packing concept

4. *In Depth Analysis*

In this phase, the analysts analyze the code of the malicious file using static and dynamic analysis in order to identify the inner functionality of the malware. Because reverse engineering is time-consuming, pre-processing steps for reducing time and effort were provided in [21]. Static analysis can be done using IDA Pro Disassembler as it has a good code graphing feature. Dynamic analysis can be performed by tools like IDA Pro Debugger, OllyDbg or Windbg usually used in kernel debugging.

5. *Providing Evidence and Long Term Mitigation Methodology*

In this phase, the malware analysts provide a full documentation with all analysis results. Mitigation Methodology is also included in this document. Mitigation methodology contains detection, tracking and countermeasure steps. The full understanding of malware behavior through previous phases helps in finding indicators of compromise, and creating signatures using tools like YARA [29].

6. *Identification*

In this phase, the infected machine is examined. In order to find the source of infection, the following steps are followed.

- Check all log files from network devices like firewalls, IDS/IPS, A/V, proxy and email servers.
- Search for known bad sites (blacklisted websites), connectivity information in order to identify the malware behavior [5].
- Use tools like Internet Evidence Finder to search memory for chat, web mail and "In private" browsing activities.
- Take memory image from the victim's machine to work on, which can be done by various tools like FTK Imager.
- Identify rogue processes by analyzing the memory image contents using tools like Volatility Framework[30] with the following plugins:
  - pslist: which uses the linked list which maintained by the kernel to show the running processes.
  - pscan: which scans the memory for EPROCESS data structure to show the running processes, therefore it has the ability to find the hidden processes. By comparing the output from the two plugins the most suspicious processes which tried to hide itself from detection can be found. This also can be done using pstotal plugin. Beside this, other parameters can be used in order to find the suspicious processes like [31]:

Process name: To make sure it is correctly spelled and matches the system's context.
Process path: To identify whether it is running from system, user or temp folder.
Parent process: To figure out which process created the suspicious process.
Process command line: To find out if the executable matches the image's name
Process starting time: In order to know the process starting time, and whether it started during the system's or later.

Other tools like Redline can also be used to find all previous findings through graphical user interface. It creates malware risk index number for each process, which helps in identifying the malicious process. Malware risk index feature determines the index based on behavior rule list and verification of digital signature [31].

7. *Investigation*

In this phase, analysts continue to explore all objects related to the malicious process like DLLs, handles, files, registry entries, connections and sockets using the following plugins:

- Dlllist: To print a list of loaded dll files for each process and it can be dumped using dlldump.
- Files: To print a list of open files for each process. It can be dumped using dumpfiles.
- Handles: To display the open handles in a process.
- Hivescan: To scan the physical memory for CMHIVE objects (registry hives).
- Hivelist: To print list of registry hives and finally hivedump to prints out a hive.
- Strings: To search for strings in the malicious process address space.
- Svcscan: To scan memory for windows service information.
- Memdump: To dump malicious processes memory.
- Connections: To print list of open connections
- Connscan2: To print list of open connections and connections that were closed prior to the memory image being acquired.
- Sockets: To print a list of open sockets.
- Sockscan: To print a list of open sockets and ports those were listening on but are not any longer.

The previous plugins are also used to search for any other suspicious network activity, which are not made by the

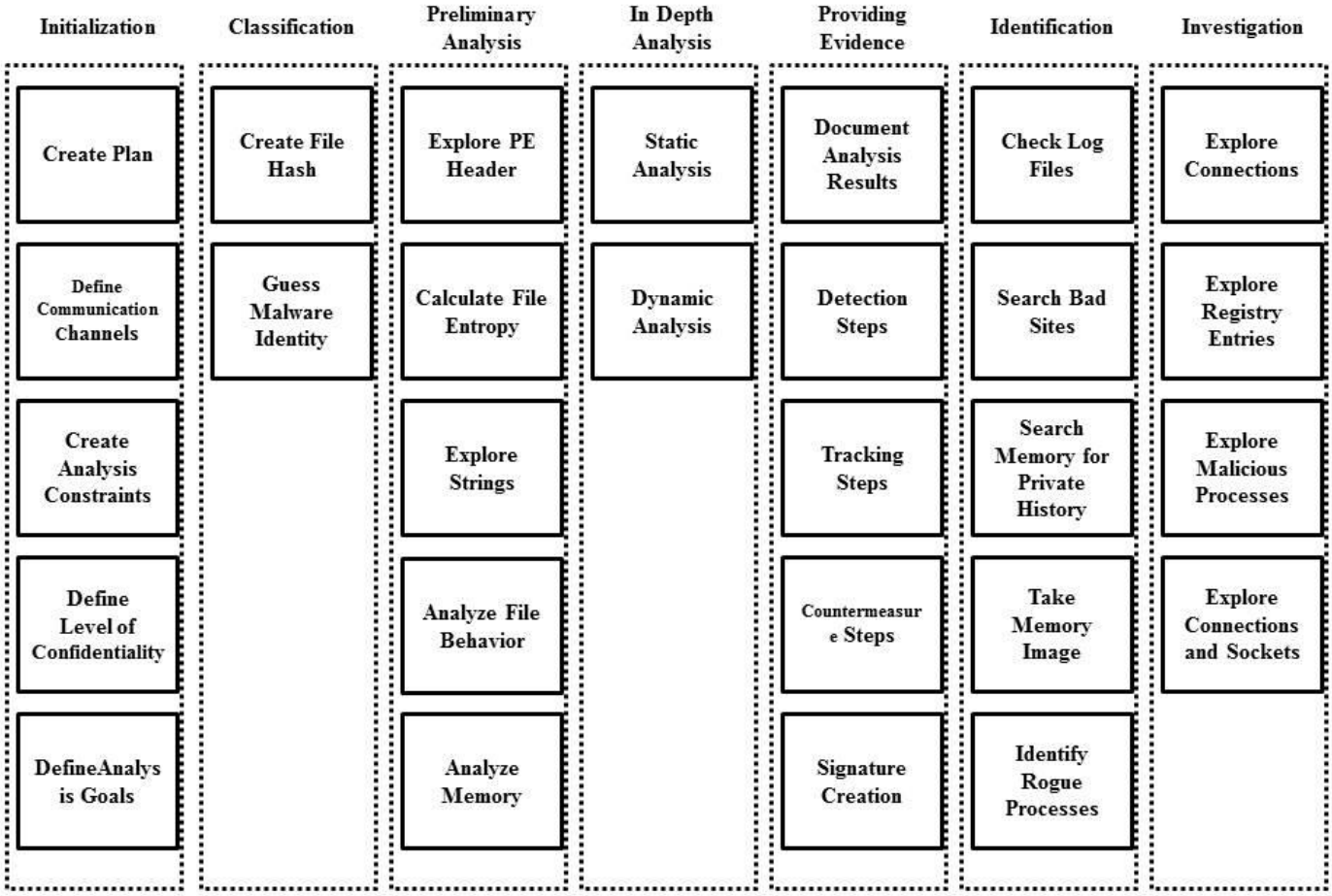

Figure 3: Building Blocks of the proposed malware incident handling and analysis work flow

suspicious process. For example, the existence of connection in suspicious ports, known blacklisted domains or IP addresses, suspicious network behavior from system behavior and so on.

Using volatility plugin procexedump, the malicious process can be dumped and pushed to in depth analysis phase in order to analyze it statically using disassembler like IDA Pro.

Afterwards, analysts search for any injected code in other processes in the system. This can be done using malfind plugin. It dumps suspicious memory segments. This is followed by searching for unlinked dlls using ldrmodules plugin. Finally, the following plugins can be used to detect if there are any residual rootkits in the system.

- **Psxview:** To find hidden processes using seven different process listing plugins
- **Modscan:** To provide a list of drivers, their location and size. It can be dumped using moddump.
- **Apihooks:** To detect two types of hooking (inline and import address table function)
- **Ssdt:** To list hooked functions in the system's service descriptor table.

Another useful tool is TotalRecall [32] which is based on volatility. It can be used to automate the previous volatility plugins. It also enables automatic investigation of dumped items with yara, Cymru, and clamav if using investigation (*-i*) option. Moreover it has option timeline (*-t*) which run plugins (Userassist-Shellbags-Shimcache- Iehistory -Mftparser- Timeliner -Evtlogs)which helping to create timeline.

Figure 3 depicts the main building blocks of the proposed malware analysis workflow

IV CONCLUSIONS AND FUTURE WORK

There is a rapid increase in malware numbers with more complicated and advanced samples. Malware authors always invent and use new techniques and methodologies to avoid detection by security defense tools and antiviruses. Therefore, there is a must to increase the researches in malware detection and analysis field in order to keep up with the development in malware creation. In addition, it is

important to be highly prepared with structured steps to deal with malware infection incidents in order to decrease the losses. New incident information can be used in improving malware detection and analysis development.

This paper presented the state-of-the-art related to malware analysis. A survey on manual malware analysis phases, recent researches in behavior, dynamic, static malware analysis and automatic malware detection and analysis was presented. Moreover, a workflow that aims to handle malware incidents was proposed. The different phases were explained in details.

In the future the performance of this workflow will be tested on a real network environment, and malware detection results, and the workflow's performance will be compared with other workflows that were proposed in the literature.

REFERENCES

[1] Israa G. SEissa;Jamaludin Ibrahim; Nor-Zaiasron Yahaya," Cyberterrorism Definition Patterns and MitigationStrategies: A Literature Review", International Journal of Science and Research (IJSR), Volume 6 Issue 1, January 2017

[2] Steven Strandlund Hansen ; Thor Mark Tampus Larsen ; Matija Stevanovic ; Jens Myrup Pedersen ,"An approach for detection and family classification of malware based on behavioral analysis", Computing, Networking and Communications (ICNC), 2016 International Conference , 15-18 Feb. 2016

[3] http://www.pandasecurity.com/mediacenter/src/uploads/2015/02/Pandalabs2014-DEF2-en.pdf

[4] "IT threat evolution in Q2 2016Overview", https://securelist.com/files/2016/08/Kaspersky_Q2_malware_report_ENG.pdf

[5] Mead, N.R.; Morales, J.A., "Using malware analysis to improve security requirements on future systems," Evolving Security and Privacy Requirements Engineering (ESPRE), 2014 IEEE 1st Workshop, 25-25 Aug. 2014

[6] Harsch, A.; Idler, S.; Thurner, S., "Assuming a State of Compromise: A Best Practise Approach for SMEs on Incident Response Management," IT Security Incident Management & IT Forensics (IMF), 2014 Eighth International Conference , 12-14 May 2014

[7] Kaiping Liu; Hee Beng Kuan Tan; Xu Chen, "Binary Code Analysis," Computer , August 2013

[8] Yuxin Gao; Zexin Lu; Yuqing Luo, "Survey on malware anti-analysis," Intelligent Control and Information Processing (ICICIP), 2014 Fifth International Conference , 18-20 Aug. 2014

[9] Zwanger, V.; Gerhards-Padilla, E.; Meier, M., "Codescanner: Detecting (Hidden) x86/x64 code in arbitrary files," Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference, 28-30 Oct. 2014

[10] Pirscoveanu, R.S.; Hansen, S.S.; Larsen, T.M.T.; Stevanovic, M.; Pedersen, J.M.; Czech, A., "Analysis of Malware behavior: Type classification using machine learning," in Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2015 International Conference, 8-9 June 2015

[11] Dornhackl, H.; Kadletz, K.; Luh, R.; Tavolato, P., "Defining Malicious Behavior," Availability, Reliability and Security (ARES), 2014 Ninth International Conference, 8-12 Sept. 2014

[12] Vasilescu, M.; Gheorghe, L.; Tapus, N., "Practical malware analysis based on sandboxing," RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, 2014 , 11-13 Sept. 2014

[13] Edem, E.I.; Benzaid, C.; Al-Nemrat, A.; Watters, P., "Analysis of Malware Behaviour: Using Data Mining Clustering Techniques to Support Forensics Investigation," Cybercrime and Trustworthy Computing Conference (CTC), 2014 Fifth , 24-25 Nov. 2014

[14] Uppal, D.; Sinha, R.; Mehra, V.; Jain, V., "Exploring Behavioral Aspects of API Calls for Malware Identification and Categorization," Computational Intelligence and Communication Networks (CICN), 2014 International Conference, 14-16 Nov. 2014

[15] Nolan, R.A.; Chen, P.P., "MCARTA: A Malicious Code Automated Run-Time Analysis framework," Homeland Security (HST), 2012 IEEE Conference on Technologies ,13-15 Nov. 2012

[16] Peidai Xie; Xicheng Lu; Jinshu Su; Yongjun Wang; Meijian Li, "iPanda: A comprehensive malware analysis tool," Information Networking (ICOIN), 2013 International Conference, 28-30 Jan. 2013.

[17] Borges de Andrade, C.A.; Gomes de Mello, C.; Duarte, J.C., "Malware Automatic Analysis," Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress, 8-11 Sept. 2013

[18] Yamamoto, T.; Kawauchi, K.; Sakurai, S., "Proposal of a Method Detecting Malicious Processes," Advanced Information Networking and Applications Workshops (WAINA), 2014 28th International Conference, 13-16 May 2014

[19] Chih-Pai Chang; Chun-Te Chen; Tsung-Hui Lu; I-Long Lin; Chang, J.; Chen-Cheng Lin, "Study on constructing malware attack forensic procedure of digital evidence," System Science and Engineering (ICSSE), 2013 International Conference, 4-6 July 2013.

[20] Aghaeikheirabady, M.; Farshchi, S.M.R.; Shirazi, H., "A new approach to malware detection by comparative analysis of data structures in a memory image," Technology, Communication and Knowledge (ICTCK), 2014 International Congress, 26-27 Nov. 2014

[21] Plohmann, D.; Eschweiler, S.; Gerhards-Padilla, E., "Patterns of a cooperative malware analysis workflow," Cyber Conflict (CyCon), 2013 5th International Conference, 4-7 June 2013

[22] https://fogproject.org/

[23] https://remnux.org/

[24] http://code.activestate.com/recipes/491264-mini-fake-dns-server

[25] http://sourceforge.net/projects/fakemail/

[26] http://www.inetsim.org/

[27] https://www.virustotal.com/

[28] https://virusscan.jotti.org/

[29] http://plusvic.github.io/yara/

[30] https://code.google.com/p/volatility/

[31] Advanced Digital Forensics and Incident Response, SANS FOR508

[32] https://github.com/sketchymoose/TotalRecall