Virmon: A Virtualization-Based Automated Dynamic Malware Analysis System

Huseyin Tirli, Abdurrahman Pektas, Yliès Falcone, and Nadia Erdogan

Abstract— **Nowadays, security companies are faced tens of thousands new malware samples every single day. Analysis of these malicious threats in an automated fashion and in a controlled environment are still the valid choice. Even though there are highly skilled dynamic analysis systems, they are weak against malwares targeting new operating systems.**

In this work, we present Virmon (Virus Monitor), a powerful system for dynamic analysis of Windows malwares. It is designed as an automated and virtualization based system. Virmon collects the behavior activities of analyzed samples in low kernel level. It can also use all Windows operating systems as an analysis environment and successfully analyze malwares targeting latest versions. We implement the system to be adaptable to the new operating systems. To demonstrate our effectiveness, we test the system with a real malware sample observed in June, 2013 and obtained results encourage us.

Index Terms— **behavior analysis, dynamic analysis, malware detection, virtualization**

I. Introduction

MALICIOUS software (malware), an executable used or created by an attacker to attain access to or steal sensitive information from a computer system, has become a major threat against computer systems and even for smart phones. According to [1], Kaspersky claims that they faced with nearly 200,000 new malware samples during the first half of 2013.

The malwares and 0-day exploits (exploits for unpatched vulnerabilities) are becoming a commercial industry. Previously, one could write malware for fun and prominence, now for money, espionage and ideological purposes. Furthermore, some states develop malwares in order to achieve their objectives. For example, *Stuxnet*, one of the most known sophisticated malware (used for advanced persistent treat – APT attacks) is developed to attack Iran’s nuclear facilities and cost an estimated *US$1 million* to create [2]. These information help us to understand how big and vast the malware development research is.

Manuscript received July 15, 2013.

H. T. is with TÜBİTAK BİLGEM Cyber Security Institute, 41470 Gebze/Kocaeli TURKEY (e-mail: huseyin.tirli@ tubitak.gov.tr).

A. P. is with TÜBİTAK BİLGEM Cyber Security Institute, 41470 Gebze/Kocaeli TURKEY (e-mail: abdurrahman.pektas@ tubitak.gov.tr).

Y. F. is with University of Grenoble I LIG, 38400 Saint Martin d’Hères FRANCE (e-mail: ylies.falcone@ujf-grenoble.fr).

N. E. is with Istanbul Technical University Faculty of Computer and Informatics 34469 Maslak/İstanbul TURKEY (e-mail: nerdogan@itu.edu.tr).

Due to the ineffectiveness of anti-viruses on previously unknown malwares (whose signature have not been created yet), many researchers have introduced several techniques to overcome limitation of anti-virus solutions [3-7]. These techniques can be divided into two categories:

- *Static analysis* means examining a file without running on the system. Because of the obfuscation techniques such as polymorphism, metamorphism, compression and encryption, it can be difficult and time consuming to statically analyze malwares.
- *Dynamic analysis* means observing activities of a suspicious file by executing it in a controlled environment so as to understand its purpose.

To successfully detect and take appropriate measures, one must be able to analyze them in a reproducible and controlled environment. There are highly skilled dynamic analysis systems revealed by malware researchers. For instance, *Anubis* formerly TTAnalyze [8], is an emulation based dynamic malware analysis system. It performs analysis operations on Windows XP OS running on Qemu emulator. During analysis, it monitors *Win32* and *native API* functions with their parameters through Qemu. Since the system runs on an emulator, analysis time may be longer and this situation can be realized by malwares.

Another tool, *Capture-BAT* [9] can monitor system changes at kernel level. It can be considered that Capture-BAT has a big disadvantage, since it is not an automated malware analysis system. *Cuckoo* [10], an open source analysis system, uses virtualization technology like Virmon. The user space API hooking technique stands behind its methodology. It can track some Win32 API functions with their parameters. Owing to the fact that Cuckoo runs at user level, it can easily be detected by malwares.

To the best of our knowledge, today’s dynamic analysis systems generally employ old versions of windows OS as their analysis environment. Nevertheless, the majority of the computer end users prefer to use the latest ones. It seems that they cannot analyze malwares targeting latest windows OS appropriately. Therefore, it is a necessity that next generation malware analysis solutions should cover latest 64-bit OSs and be adapted easily to future OS versions.

With the increase of dynamic malware analysis popularities, malware authors try to collect private information about analysis systems in order to prevent them detect their malwares. One of the most important private information is *public IP addresses* of malware analysis systems. If the public IP address of the environment in which malware executed is a known address by malware authors, malwares change their behaviors or refuse to run on these systems. For instance, *AV Tracker* [11] is a web platform publishing public IP addresses of well-known analysis systems. By using these technique, analysis systems can be evaded easily by malwares.

In this work, we propose Virmon, a scalable automated anti-malware system designed to address the above challenges. Our contributions can be described as follows:

- *Portability*: Our system can use all versions of windows OS including windows 8 64-bit. When a new windows version is released, it can be simply adjusted to use new versions.
- *Scalable & distributed analysis environment*: The system capacity can be easily increased by adding new hardware. By using sensor – analysis machine matching mechanism, the network traffic of analysis machines is distributed to different network locations. Thus, it gives an opportunity that, our analysis system is not easily detectable.

The rest of the paper is organized as follows: We start by discussing our proposed dynamic malware analysis system. Then, we follow that with implementation details of the system components. In section 4, we evaluate our system with a real-world malware sample thereby showing the effectiveness of the system. Finally, we conclude the paper with a discussion of the limitations and future works in section 5.

II. System overview

One of the most preferred methods for behavior based dynamic malware analysis is *API hooking*. With API hooking, the call made by an application to a function can be redirected to a custom defined function. When the parameters of this function are analyzed, the intention of call (behavior) can be understood.

There are two types of API hooking: *user* and *kernel* space API hooking. Generally in user space API hooking, the Win32 API and Windows native API functions are targeted. However, malware authors try to find new methods to evade API hooking. Since the applied processes are not transparent to malware, the obtained results can be misleading.

Kernel space API hooking, can be implemented with injection of codes to the kernel. As it is known, malware authors misuse the modifiable property of Windows kernel. Additionally, kernel changes can cause fatal errors and prevent proper working of some applications. Therefore, Microsoft created a kernel protection mechanism (*patchguard*) with Vista 64 bit OS to prevent this type of misusage [12]. This mechanism provides code integrity of kernel and causes a blue screen of death (*BSOD*) when a change attempt occurs. Microsoft encourages the user of Windows to use callback mechanism supplied by OS.

Windows callback mechanism provides the delivery of necessary information needed by drivers when the certain conditions are met [13]. In order to be notified of events (*process, registry* and *file*), kernel space component registers itself to OS. When the desired events occur, OS calls the function of the driver determined at registration time. Owing to the fact that patching the kernel of new OS's is too hard, it is unreasonable to try to patch kernel code for behavior monitoring. Thus, we prefer to use callback mechanism presented by the OS.

Our approach to construct a dynamic analysis environment consists of two main components:

i. *Analysis machine components*: It is responsible of reporting host-based activities such as process, registry and file system activities created by analyzed file.
ii. *Network components*: Network components of Virmon is responsible of reporting network activities of an analyzed file and consists of the following modules:

- The sensors at different locations that forward network traffic of analysis machines to the command and control (C&C) servers.
- Virtualization environment which contains the analysis machines.
- An application server managing whole analysis processes.
- IDS that generates alarms, extracts files and HTTP requests from network traffic.
- A DNS server replying and logging DNS requests.
- An NTP server that used to synchronize all machines in the system.
- A database storing all events captured during analysis.

III. System architecture

In this section, we provide the details of Virmon dynamic analysis system. Firstly, we give information about how we collect behavior activities of analyzed suspicious files from operating system (OS). Next, we discuss the network topology of the system where we automatically analyze files to gather their network activities.

A. Analysis Machine Components

On each machine that analyzes malicious software, there are two main components are activated: application working in user space and managing analysis process, and drivers working in kernel space. The application running in user space can be considered as a bridge between OS and application server. The file taken from application server is executed by user space component. The process id information that is obtained by executing the file is sent to the kernel space component. Kernel space component watches three main activities:

1. Process activity
2. Registry activity
3. File system activity

Activities of the process owned by that executable and (if any) processes created by this process are monitored by kernel space component. In the following part, we describe what and how to monitor each type of activities listed above.

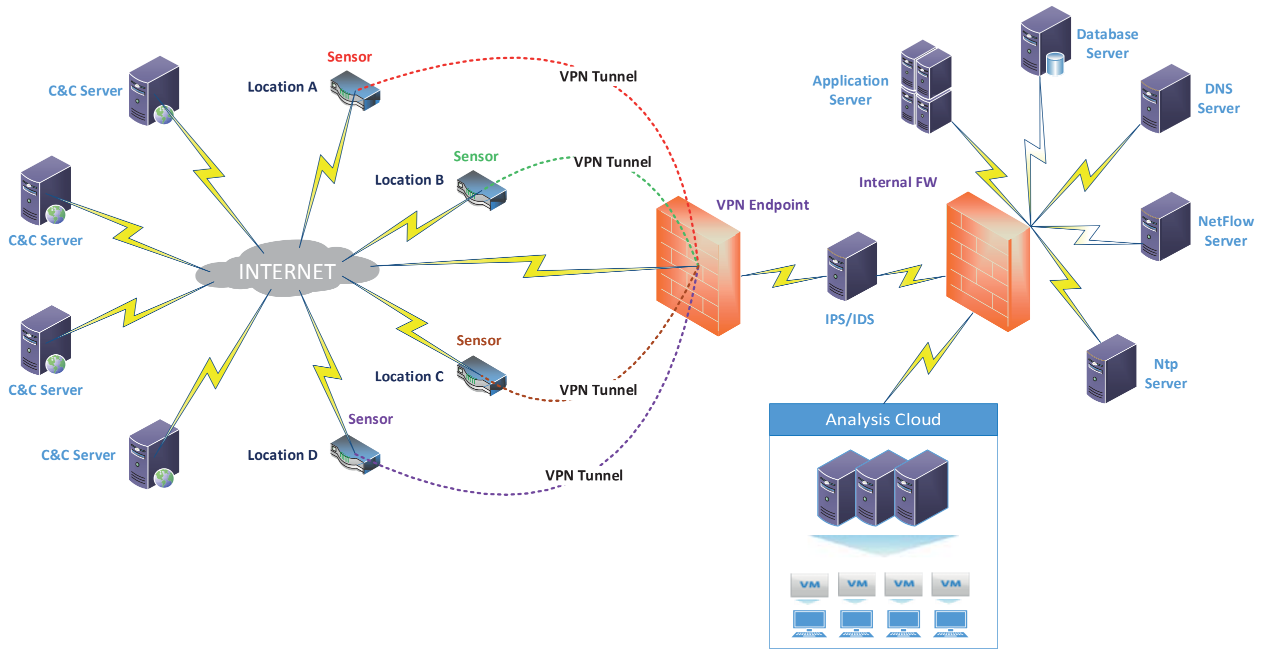

Fig. 1. The topology of Virmon including all the local and remote components of malware analysis system.

1) Process Monitoring

One of the activities needed by malware researchers is process activity of analyzed malware samples. Process monitoring part of the kernel space component observes *process creation/deletion* and *remote thread creation* events. When an application tries to create a new process, OS delivers the information about the process starting the create operation (parent process) and the process to be created to the process monitor. In a similar manner, before a new thread created, OS notifies thread information to the process monitor. If currently working process is one of the processes that is tracked by process monitor and created thread does not belong to current process, this event also recorded as a remote thread creation. In these event types, notification operation happens in the context of the thread creating the new process or thread [14] [15]. On the other hand, in delete process event, notification routine runs in the context of the last thread to exit from the process [14]. Therefore, it is too hard to understand who killed the process. Due to this reason, we only record the deletion events of processes tracked by the process monitor.

2) Registry Monitoring

Windows Registry is a hierarchical database used to store configuration information of applications, hardwares and users [16]. In order to understand what changes are made by the analyzed file on the system, monitoring the registry can be a useful method. Majority of the malwares use windows registry so as to be persistent. They register themselves to the registry to be executed automatically at startup. Also some of them uses the registry to prevent users from calling the task manager, defender etc.

Windows callback mechanism can notify the drivers when events such as read, write and delete on the registry occurs. There are two types of registry notifications: *preoperation* and *postoperation*. By monitoring each types of events, it can be determined whether the operation is successfull or not. *OpenKey*, *CreateKey*, *DeleteKey*, *SetValueKey*, *DeleteValueKey*, *QueryValueKey* and *EnumerateKey* registry operations are watched by registry monitor.

3) File System Monitoring

File system monitor is used to track all file system events (*read/write/delete*) on the system. Monitoring the file system is extremely crucial, since lots of malwares copy themselves to a known location in the OS. Also some of them downloads new payloads or updates itself periodically. Like registry monitor, pre and post IO operation notifications delivered by file system filter manager. The read, write and delete events caused by all tracked processes are recorded by the file monitor.

B. Network Components

Malwares often need to connect C&C servers to send confidential information collected from compromised machines or to take commands from them. That is why analysis of the network activities of malwares is an inevitable requirement for malware analysts. In the proposed system we use different network solutions, such as VLAN, VPN, IPS, and firewall to monitor network activities of suspicious files. We are inspired by our previous work [17] and performed to collect malware samples via high interaction honeypots [18].

The *sensor*, a kind of mini-PC, is placed at any location on the internet in a plug-and-play fashion and is used to match its public IP address to private IP address of an analysis machine. Its main objective is to forward traffic of analysis machines to the Internet. To do this end, each sensor creates a secure channel (*VPN*) to our data center (as depicted in *Figure-1*).

With the use of VPN connection, each analysis machine seems to be directly connected to the Internet. Therefore, IP layer information (e.g. source, destination IP and TTL) of all network packets originating from analysis machine are updated on the sensors to make them appear as if those packets were sent by sensors. *Figure-2* illustrates logical network topology of analysis machines.

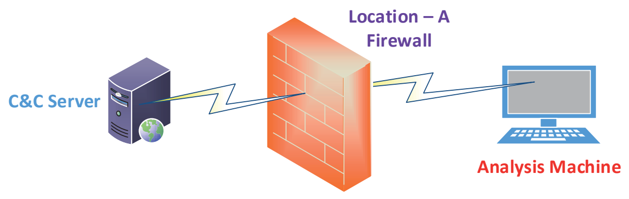

Fig. 2. The logical topology of the system. Each analysis machine is seen by C&C servers as if it is working behind the firewall of network that includes our corresponding sensors.

We assign statically separate *C class* networks to each analysis machines on the *VPN Endpoint*. When a suspicious file analyzed in our system tries to make a connection to remote host, VPN endpoint identifies the related sensor and forwards its network traffic to that sensor.

In addition, to improve transparency of the system, we define separate VLANs for each analysis machines on the internal firewall and virtualization server. So, we reduce the size of broadcast domain, as well as prevent analysis machines from seeing each other.

1) DNS-Server

The Domain Name System is a dynamic database service that translates internet domains and host names to IP addresses. It is a critical service for internet access since internet relies on IP addresses and without the DNS service it would be an obligation for users to know all IP addresses of servers accessed. Similarly, malwares often use domain names to access C&C servers which are changed frequently by attackers to be undetected (fast flux) [19].

To monitor DNS requests made by analysis machines we manually set the DNS server of each machine to our DNS server located in the data center. Meanwhile, in order to prevent malwares to resolve domain name by using public DNS server, (e.g. 8.8.8.8 is Google's well-known public DNS server) we redirect all DNS queries to internal DNS server by using our internal firewall. In this way, all DNS requests made by malwares can be answered and logged by our DNS server. DNS server logs requested domain, time and IP of analysis machine. DNS log file is parsed and the extracted information is sent to database at runtime.

2) IPS/IDS (IPDS):

Intrusion prevention system (*IPS*) is a network security solution monitoring network traffic and system activities. We need an IPS to prevent likely networks attacks caused by suspicious files analyzed in the system.

Due to the following reasons, analysis of web pages requested and files downloaded by malware has become a necessity for the malware analyzer:

1. Nowadays great deal of companies allow their personnel to make only HTTP/HTTPS, FTP etc. connections because of security reasons. Malwares have adapted themselves to use HTTP/HTTPS protocols to communicate with their C&C servers.

2. As it is known, a special type of malware named dropper downloads another malware. Thus, if an analyzed file downloads extra contents via internet, these contents also should be examined.

Suricata [20], an open source IPS solution, can prevent malicious attacks such as distributed denial of service (DDoS), port scanning and shellcodes; it can also extracts files and HTTP requests from live network traffic. Therefore, we have decided to use Suricata as the IPS component of our dynamic analysis system.

Suricata writes HTTP requests to a log file. HTTP log file is parsed at predefined regular intervals and the extracted information is stored to a database. Meanwhile, if the files extracted by Suricata have not already been dissected beforehand, they are enqueued into application server's priority queue.

3) Netflow Server

Netflow server operates to obtain summary information about network traffic of the analysis machines. It records source and destination hosts, the protocol used, ports, flow duration and number of the bytes transferred. To extract these information, all network traffic made by analysis machines is duplicated by internal Firewall and sent to the netflow server. When the server obtains brief information about the flow, it stores them to a database at regular time intervals similar to other servers.

4) NTP-Server

Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems. During analysis, we collect information from different analysis servers. The time information of each type of data must be the same on all servers. To do this end, we deploy NTP server so as to synchronize the time of each server in the system.

5) Application Server

It is responsible of the management of the total malware analysis processes. Application server determines the analysis machine on which a submitted file will be analyzed. It collects the activities from all analysis system components. Then, it reformats the collected data and stores them in a database. Furthermore, application server returns the analysis machines to the clean state after the analysis operations are completed.

IV. Evaluation

In this section, we demonstrate how our dynamic malware analysis system works. Firstly, we explain the communication mechanism between the application server and analysis machines. Then, we show results of a recently observed malware's analysis.

Figure-3 visualizes the communication process between application server and an analysis machine. The process can be described as follows:

1. Analysis application sends a request that specifies the start of new analysis process to the application server.
2. Application server pops a file from its priority queue and sends it to the analysis machine which initiates a new analysis request.
3. Analysis application injects codes to the running *explorer.exe* process. This process executes the file coming from application server in *suspend mode*. If the file is executable, explorer.exe writes id (PID) of newly created process to a shared memory. Analysis application reads PID info from shared memory. Then it sends a message that includes this info to the driver and asks it to record events

initiated by this process.

4. Analysis application waits until the analyzed process exits or a timeout (3 minutes). Then it sends a message to the driver to stop the recording events.
5. Driver stops the recording and writes events to the event file.
6. Analysis application sends the event file to the application server for parse operation.
7. Application server parses the event file and stores them in the database. Than it returns the analysis machine to the clean state.

In order to show the capabilities of our dynamic malware analysis system, we analyze a dropper which was detected on July, 2013 [21]. Dropper is a type of malware created by malware authors to download and install new malwares (e.g. virus, backdoor) on the compromised systems. It can also include the malwares in itself.

The analysis operation is done automatically on a Windows 8 Ultimate 64-bit OS. The steps described above is completed and the activities of malware sample are examined. The important events that have happened on the system is displayed in Table-1.

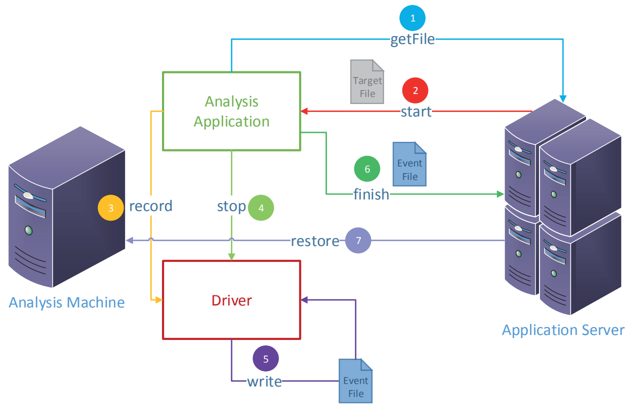

Fig. 3. The communication mechanism between application server and an analysis machine. It demonstrates the main information flow of analysis.

Since the number of events gathered from analysis system for this malware is too high (10000+), only the important events selected manually are included in the table. The majority of the activities are system dll files and registry accesses. If the table is examined, the intention of malware sample can be understood easily.

TABLE I
IMPORTANT ACTIVITIES OF ANALYZED SAMPLE

Time	Event	Process	Detail
2013-07-09 09:50:42.048	Process Create	C:\Windows\explorer.exe	C:\Users\virmon\Desktop\file.exe (Analyzed File)
2013-07-09 09:50:45.018	Process Create	C:\Users\virmon\Desktop\file.exe (Analyzed File)	C:\Users\virmon\Desktop\file.exe
2013-07-09 09:50:45.172	Process Terminate	C:\Users\virmon\Desktop\file.exe (Analyzed File)	-
2013-07-09 09:50:47.906	DNS Query	C:\Users\virmon\Desktop\file.exe	sunelec-kk.com
2013-07-09 09:50:50.393	HTTP Request	C:\Users\virmon\Desktop\file.exe	/tmp/r1.php on sunelec-kk.com (5 times in 24 seconds)
2013-07-09 09:51:15.324	DNS Query	C:\Users\virmon\Desktop\file.exe	japmotors.net
2013-07-09 09:51.15.686	HTTP Request	C:\Users\virmon\Desktop\file.exe	/tmp/r1.php on japmotors.net
2013-07-09 09:51:16.664	DNS Query	C:\Users\virmon\Desktop\file.exe	www.piazzabrothers.com
2013-07-09 09:51:16.876	HTTP Request	C:\Users\virmon\Desktop\file.exe	/tmp/file1.exe on www.piazzabrothers.com
2013-07-09 09:51:17.246	HTTP Request	C:\Users\virmon\Desktop\file.exe	/tmp/file2.exe on www.piazzabrothers.com
2013-07-09 09:51:17.936	DNS Query	C:\Users\virmon\Desktop\file.exe	fondear.es
2013-07-09 09:51:18.776	HTTP Request	C:\Users\virmon\Desktop\file.exe	/tmp/file1.exe on fondear.es
2013-07-09 09:51:18.991	HTTP Request	C:\Users\virmon\Desktop\file.exe	/tmp/file2.exe on fondear.es
2013-07-09 09:51:19.677	File Write	C:\Users\virmon\Desktop\file.exe	C:\Users\virmon\AppData\Local\Temp\990203.bat
2013-07-09 09:51:20.443	Process Create	C:\Users\virmon\Desktop\file.exe	C:\Windows\SysWOW64\cmd.exe
2013-07-09 09:51:20.490	Process Create	C:\Windows\SysWOW64\cmd.exe (cmd /c C:\Users\virmon\AppData\Local \Temp\990203.bat C:\Users\virmon\Desktop\file.exe)	C:\Windows\System32\conhost.exe
2013-07-09 09:51:20.490	Process Terminate	C:\Users\virmon\Desktop\file.exe	-
2013-07-09 09:51:22.115	File Delete	C:\Windows\SysWOW64\cmd.exe	C:\Users\virmon\Desktop\file.exe
2013-07-09 09:51:22.568	File Delete	C:\Windows\SysWOW64\cmd.exe	C:\Users\virmon\AppData\Local\Temp\990203.bat
2013-07-09 09:51:22.662	Process Terminate	C:\Windows\SysWOW64\cmd.exe	-
2013-07-09 09:51:22.678	Process Terminate	C:\Windows\System32\conhost.exe	-

When the sample is executed, firstly the process created by explorer.exe creates a new process having the same name in its directory. This technique, named as *process hollowing* [22], is often used by malwares recently to hide themselves. Then, the process created by explorer.exe terminates itself. The file.exe process created by using hollowing technique sends HTTP requests to predefined four distinct compromised servers respectively to download new malwares. Since the malwares targeted by file.exe are deleted from servers before the analysis operation, it cannot access to malwares and changes its behavior. The dropper creates a script named as *990203.bat* and executes this script. By this execution, malware terminates itself and the script removes malware's footprints. As it is seen on Table-1, our dynamic malware analysis system successfully collects the behavior activities of the analyzed sample.

V. CONCLUSION

In this paper, we present a virtualization-based automated dynamic malware analysis system. We have developed a kernel driver which can monitor state changes during the malware analysis. Virmon currently supports all Windows OSs and we can easily adapt the system to use new OSs. The distributed architecture of the system can be extended easily. Sensor – analysis machine matching mechanism can help us to hide the analysis system against the malwares using IP based detection techniques.

Currently, we are not fully capable to analyze malwares which are sensitive to virtualized environments because of the fact that our system based on virtualization technology. Also the system can only analyze executable files. Additionally, the network activities of the analysis machines are gathered system-wide. Therefore, we are planning to improve our system to be capable of analyzing virtualized environment sensitive malwares, collect network events on low kernel level and also increase the number of file types that can be analyzed on Virmon. Meanwhile we are working on automated classification of malwares and encouraging ourselves to integrate the classification system into Virmon.

REFERENCES

[1] "Kaspersky Lab Press Release 2012", Internet: http://newsroom.kaspersky.eu/fileadmin/user_upload/en/Downloads/PDFs/101212_Press_Release_2012_by_numbers.pdf, [Jul 10, 2013].
[2] "Estimated Cost of Stuxnet", Internet: http://www.thenational.ae/business/industry-insights/technology/former-cia-chief-speaks-out-on-iran-stuxnet-attack, [Jul 14, 2013].
[3] A. Moser, C. Kruegel, and E. Kırda, "Exploring multiple execution paths for malware analysis," in Proceedings of the IEEE Symposium on Security and Privacy, 2007, pp. 231-245.
[4] H. Agrawal, L. Bahler, J. Micallef, S. Snyder, and A. Virodov, "Detection of global, metamorphic malware variants using control and data flow analysis," in Military Communications Conference (MILCOM), 2012, pp. 1-6.
[5] H. Huang, G. Acampora, V. Loia, C. Lee, and H. Kao, "Applying FML and Fuzzy Ontologies to Malware Behavioral Analysis," in Proceedings of the IEEE Symposium on Fuzzy Systems (FUZZ), 2011, pp. 2018-2025.
[6] G. Tahan, L. Rokach, and Y. Shahar, "Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features," in The Journal of Machine Learning Research, vol. 13, 2012, pp. 949-979.
[7] R. Rolles, "Unpacking Virtualization Obfuscators," in Proceedings of the 3rd USENIX conference on Offensive technologies, 2009, pp.1-10.
[8] U. Bayer, C. Kruegel, E. Kirda, "TTAnalyze: A tool for analyzing malware," Internet: http://www.iseclab.org/papers/ttanalyze.pdf, [Jun. 21, 2013].
[9] C. Seiferta, R. Steensona, I. Welcha, and P. Komisarczuka, B. Endicott-Popovskyb, "Capture – A behavioral analysis tool for applications and documents," Digital Investigation: The International Journal of Digital Forensics & Incident Response, vol. 4, 2007, pp. 23-30.
[10] "Cuckoo Sandbox", Internet: http://www.cuckoosandbox.org/, [Jun. 20, 2013].
[11] "AV Tracker", Internet: http://www.avtracker.info/, [Jul. 5, 2013].
[12] "Kernel Patch Protection: Frequently Asked Questions." Internet: http://msdn.microsoft.com/en-us/library/windows/hardware/gg487353.aspx, Jan. 22, 2007 [Jun. 18, 2013].
[13] "Callback Objects." Internet: http://msdn.microsoft.com/en-us/library/windows/hardware/ff540718(v=vs.85).aspx, [Jun. 18, 2013].
[14] "PsSetCreateProcessNotifyRoutineEx routine." Internet: http://msdn.microsoft.com/en-us/library/windows/hardware/ff559953(v=vs.85).aspx, [Jun. 20, 2013].
[15] "PsSetCreateThreadNotifyRoutine routine." Internet: http://msdn.microsoft.com/en-us/library/windows/hardware/ff559954(v=vs.85).aspx, [Jun. 20, 2013].
[16] "What is the registry?" Internet: http://windows.microsoft.com/en-id/windows-vista/what-is-the-registry, [Jun. 21, 2013].
[17] N. E. Siseci, B. Emre, and H. Tirli, "Case Study: Malicious Activity in the Turkish Network," Internet: http://www.syssec-project.eu/media/page-media/3/syssec-d5.3-TurkishNetworkCaseStudy.pdf [Jun. 23, 2013]
[18] A. Caglayan, M. Toothaker, D. Drapaeau, D. Burke, and G. Eaton, "Behavioral Patterns of Fast Flux Service Networks," in in Proceedings of 43rd Hawaii International Conference on System Sciences (HICSS), 2010, pp. 1-9.
[19] J. Nazario, and T. Holz, "As the Net Churns: Fast-Flux Botnet Observations," in Proceedings of 3rd International Conference on Malicious and Unwanted Software, 2008, pp. 24-31.
[20] "Suricata", Internet: http://suricata-ids.org/, [Jul. 1, 2013].
[21] "Antivirus scan for b0d36b9d9a6866a70b13b71772a0de84 at 2013-06-26 16:01:34 UTC - VirusTotal" Internet: https://www.virustotal.com/tr/file/c3833351f751eedcd13e23cff488d66dc9e45aab0b86a4c4b6e3481389bf5e29/analysis/, [Jul. 10, 2013].
[22] M. H. Ligh, S. Adair, B. Hartstein, and M. Richard. Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code. Indianapolis, USA: Wiley Publishing, Inc, 2011, pp. 596-599.