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/ijatcse55952020.pdf
https://doi.org/10.30534/ijatcse/2020/55952020

Constructing Pattern of Mirai Botnets Attack using Graph Theory Approach

Raihana Syahirah Abdullah[1], Amirulnaim Nazri, Warusia Yasin[2], Faizal M.A[3] , Wan Nur Fatihah Wan Mohd Zaki
[1]Dr, Center for Advanced Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka (UTeM), 76100 Durian Tunggal, Melaka, Malaysia
raihana.syahirah@utem.edu.my
[2]Dr, Center for Advanced Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka (UTeM), 76100 Durian Tunggal, Melaka, Malaysia
s.m.warusia@utem.edu.my
[3] Associate Professor, Center for Advanced Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka (UTeM), 76100 Durian Tunggal, Melaka, Malaysia
faizal.abdollah@utem.edu.my
(Corresponding author: Raihana Syahirah Abdullah: raihana.syahirah@utem.edu.my)

ABSTRACT

Botnets is a community of Mirai botnets-infected computers to perform various attacks. Hackers were using vulnerabilities in the Mirai botnets to create bots that can execute malicious activities. A sandbox experimentation was conducted to identify Mirai botnets behaviors and distinguish Mirai botnets activities. The Mirai botnets behaviours were analyzed in order to construct the Mirai botnets attack pattern. The Mirai attack pattern is constructing using graph theory approach. The results reveal that the Mirai botnets can be identified by their behaviours by examining their attack activities. The experimental results also indicate that the graph theory approach can be identify and differentiate benign and botnets traffic efficiently. Therefore, this research proposes a graph theory approach in constructing the Mirai botnets attack pattern through its behaviors.

Key words: IoT Botnets, Mirai, Graph Theory.

1. INTRODUCTION

Computer security is now the biggest problem due to rising security threats. Any user using the IoT devices or the IoT environment platform must be aware the existing of Mirai botnets attack. The Internet of Things (IoT) gadgets are constantly multiplying, as indicated by [1]. The huge number of risky IoT gadgets makes them low-hanging focuses to make huge scale botnets for assailants. The expansion of IoT Mirai botnet and particularly IoT botnets is legitimately identified with the multiplication of IoT gadgets.

While, botnets are a standout amongst the most dominant parts of a cutting edge digital criminal's weapons store of assault strategies due to their boundless size and capacity [2]. [3] expressed that the first recognized by the whitewashed security research bunch Mirai botnetMustDie, and its numerous variations and imitators filled in as the vehicle for the absolute most dominant DDoS assaults ever.

Nowadays, Mirai botnets is spread very rapidly within the network and any platform. Due to this issue, sometimes it is difficult to identify the behaviour of Mirai botnets. The interest of cybercriminals in IoT devices continues to growth. Based on Figure 1, the number of Mirai botnet samples attacking smart devices is increasing three times as many as in all of 2017. And there were ten times as many in 2018. For the years ahead, that does not bode well.

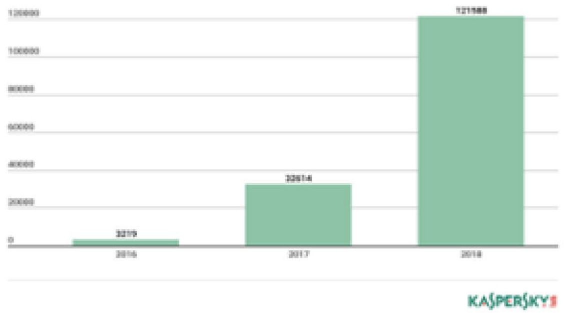
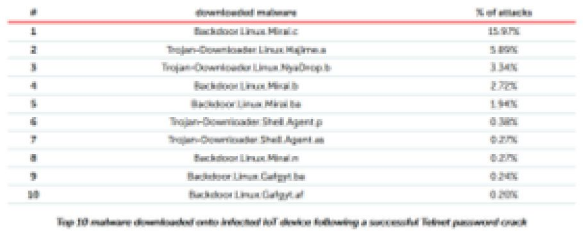


Figure 1: Number of Mirai botnet samples for IoT devices in Kaspersky Lab’s collection, 2016-2018

When it came to download Mirai botnets to IoT devices, one of the Mirai family 20.9% was the preferred option for cyber criminals. The Mirai botnets, made up of IoT gadgets and engaged with DDoS assaults whose scale broke every single imaginable record, causing disavowal of administration all through a district, was broadly secured by the broad communications.



#	downloaded malware	% of attacks
1	Backdoor.Linux.Mirai.c	15.97%
2	Trojan-Downloader.Linux.Hajime.a	5.89%
3	Trojan-Downloader.Linux.NyaDrop.b	3.34%
4	Backdoor.Linux.Mirai.b	2.72%
5	Backdoor.Linux.Mirai.ba	1.94%
6	Trojan-Downloader.Shell.Agent.p	0.38%
7	Trojan-Downloader.Shell.Agent.as	0.27%
8	Backdoor.Linux.Mirai.n	0.27%
9	Backdoor.Linux.Gafgyt.ba	0.24%
10	Backdoor.Linux.Gafgyt.af	0.20%

Top 10 malware downloaded onto infected IoT device following a successful Telnet password crack

Figure 2: Top 10 Mirai botnet downloaded onto infected IoT device following a successful Telnet password crack.

In the field of botnets discovery and botnets distinguishing proof, which is dynamic methodologies and aloof methodologies, two sorts of strategies can be perceived. So as to mimic in conducting a genuine bot, dynamic methodologies include defanging bots or composing specific bots. Detached methodologies are those that can't be identified using any and all means, as there is no progression of data back to the botnet administrator. Dynamic methodologies incorporate a wide range of strategies that illuminate the botnets administrator of the perception straightforwardly or by implication.

The analyst arranged botnets discovery and following strategies into three classes: honeypots, characterization of traffic applications and examination of detached irregularities [4]. The developed Honeypot server application is capable of analysing data in real time as it has been combined with IDSs to provide effective detection level capabilities (Baykara & Das, 2018).

The Mirai botnets can be detected early in the field of IoT by knowing their taxonomy and behaviours. So, this research will focuses on sandbox as a method of analyze for Mirai botnets behaviour. While, to detect Mirai botnets, it is necessary to analyze the Mirai botnets attack pattern by knowing its behaviors. It is possible to specify the attack pattern by using of graph theory approach.

This section started with the briefly explanation of Internet of Things (IoT). The remainder of this paper is presented as follows: Section II discusses related studies and Section III explain the methodology used for this paper. Section IV presents some analyses of the results. Section V concludes the paper and suggests future work directions.

2. RELATED WORK

The Mirai botnet caught worldwide attention in 2016, consisting of streak-breaking DDoS attacks on Krebs, OVH, and Dyn. The botnet, which in the closed-circuit targeted TV cameras, routers and DVRs, generated volumes of traffic above 1Tbps. This botnet has taken down service providers and cloud scrubbers infrastructure with ten predefined attack vectors. Some of the vectors are GRE flooding and water torture attacks. Based on what the author mentions, Mirai IoT botnet is a worm-like Mirai botnet family that infected and corralled IoT devices into a DDoS botnet at the end of 2016. This botnet starts flooding with HTTP and uses IoT device network level attacks as a target. When the botnet infects a device and wipes it out and claims the gadget as its own, Mirai searches for other malware on that device [3].

To detect malware, there are many types of detection. They are a system for firewall, honeypot and detection of intrusion [5]. Every detection can secure host or network, or both, from any intrusion. There is many type of anomaly detection which is Flow-based methods, Graph-based methods, Clustering and Classification [6]. There are group that can be categorized, static graphs and dynamic graphs for detecting anomalies [7].

An undirected graph element, sometimes referred to as a connected element, is a subgraph in which any two vertices are connected by paths to each other and no additional vertices in the super graph are connected to each other. For example, there are three components in the graph shown in the illustration. An incident-free vertex is itself a component. A graph connected by itself has exactly one component, which consists of the entire graph.

A dynamic malware analyser proposed by [8] against virtual machine-conscious Mirai botnet is capable of bypassing anti-VM detection techniques to detect malware by monitoring its behaviour in a virtual environment. The method proposed may bypass detection techniques and identify the technique used by malware as well. It is capable of monitoring system resources like connections, files, processes, and services. It also generates information about changes in the registry of Windows to the analyst. The detection ratio tested using the malware set of Pahadus is successful and 92% accuracy of malware samples is detected.

In addition, by using behavioural sequential patterns as a method of malware detection [9], a dynamic malware detection system based on mining the API sequences and iterative patterns extracted from an executable trace of API calls was proposed. The framework can examine and detect malicious behaviour as well as introduce in this field the concept of iterative pattern mining.

In AccessMiner, which uses system-centered models for malware protection, a large-scale collection is carried out to study the diversity of system calls [10]. Data analysis demonstrates that straightforward identifier of malware by utilizing an elective location model that describes the general collaborations between benevolent projects and the OS. Considerate projects that entrance OS assets, for example, records and library passages are investigated by framework driven methodology models and result in not very many or even zero false positive Mirai botnet discovery.

The method discussed by [7] in malware investigation and identification instruments examinations understood malware and favorable projects and looks at the outcomes got. In the investigation, 100 examples of malware and 100 favorable projects were gathered from different sources and dissected under different variants of Windows machines. The test outcomes demonstrated that malware can barely be distinguished by utilizing just one instrument. By consolidating static and dynamic investigation apparatuses, precision and identification rate can be expanded.

HOLMES as figured out by Jha, [11], similar to the Behaviour-based Detection Model, also analyses files and registry using another model. It exhibits a programmed system for ideally removing discriminative determinations that decide a program class particularly. Because of probabilistic testing of determination space, the proposed method depends on chart mining and idea examination, scales to enormous classes of projects. The proposed HOLMES can blend biased details that unequivocally recognize programs, keeping up a recognition rate of 86% on new, obscure malware with zero false positive rates.

The comparison of related works on Windows Registry under detection malware illustrated in Appendix A: Detection of Malware on Windows Registry illustrates that dissimilar detection designs are used in the same location of malware.

Then, the comparison of related works on detection of malware using graph theory are illustrated in Appendix B: Detection of Malware using Graph Theory. According to [12], a researcher use graph-based detection method that focus on dependency behaviour graph. Dependency graph used to identify a node that uses the name of API call for each node. Dynamic taint analysis technique has been used in this research to find the relations between system calls. A researcher also proposed an algorithm to extract the common behaviour graph by using graph matching algorithm. The result from this approach is different for each malware family. For overall, the result is high true positive which is more than 80% true positive rate and less than 20% false positive rate.

Based on the research by [13], dependency graphs (ScD-graphs) has been used to determine whether a program is malicious or benign a researcher also propose the similarity metric for detection process. In this research, 2631 malware samples pre-classified into 48 families of malware have been tested. An experiment also testing on unknown sample file. A result from this method, it achieved 94.7% true positive and 13.1% false positive.

A graph-based malware activity detection are proposed by [14] by achieving robustness against evasion techniques. This method used to detect infected clients and malicious domain names in DNS traffic. Four sets of DNS traffic that captured from ISP networks have been used to do an experiment for this method. As a result, the method achieved high true positive which is more than 84% of TPR and less than 0.26% FPR.

According to [15], the proposed method is classification method based on the maximal common subgraph detection. A graph is consulted by capturing system call in sandbox environment. The method has been tested on 300 malware samples that consist of 6 families of malware. A result from this method shows that TPR is high and FPR is low.

The method that proposed by [16] is using function-call graph to compute similarity between two binaries. Next, a researcher proposed a graph matching method also to compute similarity between two binaries. For this research, it used malware samples that provide by VX Heavens. The result shows that high TPR and low FPR.

3. METHODOLOGY

This research is utilizing the methodology demonstrated to be a cascade model for this venture. The Figure 3 shows all of this research's methodology that have four phases. First, the phase of data collection and information gathering. Next, the phase of data extraction and analysis. The final phase is to construct the graph theory. In this chapter, the description of each phase will be discussed further.

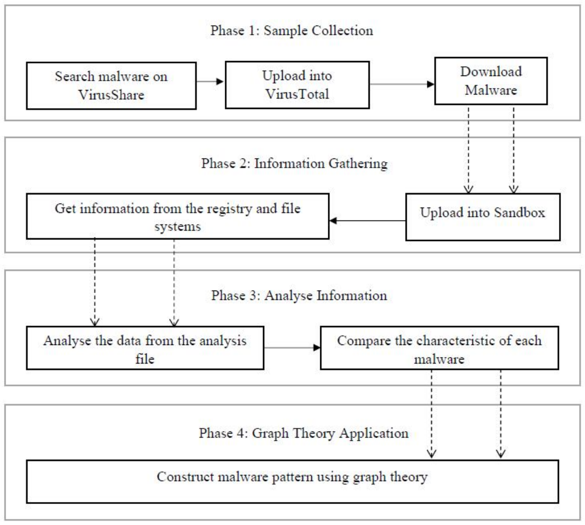

Phase 1: Sample Collection
Search malware on VirusShare
Upload into VirusTotal
Download Malware
Phase 2: Information Gathering
Get information from the registry and file systems
Upload into Sandbox
Phase 3: Analyse Information
Analyse the data from the analysis file
Compare the characteristic of each malware
Phase 4: Graph Theory Application
Construct malware pattern using graph theory

Figure 3: Research Methodology.

Figure 4 showed the Cuckoo sandbox's architecture design. Cuckoo sandbox is act as centralized access that handles the analysis and samples Mirai botnets samples execution for initialize the Phase 1 and Phase 2. Each one of the analysis is started neither in a physical or virtual machine that is fresh and isolated. Cuckoo's main components can be categorized in two which are the host machine and the number of guest machines. The host play as an important role in running the core sandbox's components that handles the entire process of analysis, where the guests in another hand are the isolated environments where the samples of Mirai botnets are executed and analyzed in a safe manner.

In the sample collection stages, at first, open the web browser's VirusShare to select samples of Mirai botnet. Reading the description provided will select the appropriate IoT Botnet. In ensuring that Mirai botnets is not outdated and can be analyzed, the latest Mirai botnet must be considered. Then, insert the Mirai botnet sample MD5 or SHA value into VirusTotal to check the variant validation.

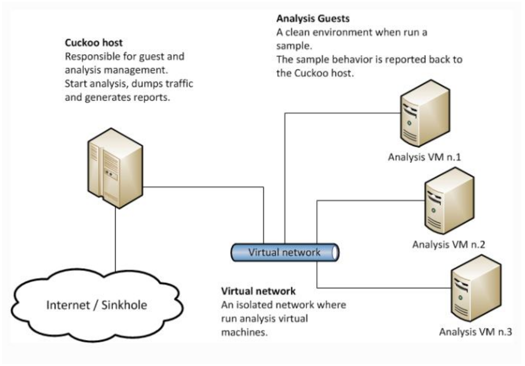

Analysis Guests
A clean environment when run a sample.
The sample behavior is reported back to the Cuckoo host.
Cuckoo host
Responsible for guest and analysis management.
Start analysis, dumps traffic and generates reports.
Analysis VM n.1
Analysis VM n.2
Virtual network
Internet / Sinkhole
Virtual network
An isolated network where run analysis virtual machines.
Analysis VM n.3

Figure 4: Architecture Design ("Cuckoo Sandbox," 2018).

In the information gathering stages, the phase of data extraction is applied. For this part, this research use the sandbox to detect each Mirai botnet's activity and behavior. Cuckoo Sandbox is used for this project to analyze the Mirai botnet. After download the Mirai botnet samples, upload the chosen samples into Cuckoo Sandbox. It will display the report after submitting the samples. The Cuckoo report provides a summary of Mirai botnet execution information. We can extract the information on the basis of the summarized report.

During the analyse information phase, the report produce from sandbox will be analysed. We will analyse each of the features yield such as registry. In each of the features, we will look up the several files which are read, modified and deleted. The last stages is graph theory application. After analysing the Mirai botnet sample, the graph is constructing. The graph is construct according to the API call of the Mirai botnet produced in the report. Then, the graph created is compared to another graph of each Mirai botnets. After comparing the graph, we will extract the pattern of the graph and find the similarities of each Mirai botnets sample.

4. RESULTS AND DISCUSSION

This section defines the results and analyses for Mirai botnets evaluation from the experiment. Analysis is essential to understand the Mirai botnet sconduct to detect Mirai botnets using a graph theory approach. The md5 sample of Mirai botnets are validate using VirusTotal in order to validate the sample. To run the study, the complete sample is 20 sample, which must be in the same Mirai botnet variant. In this graph theory approach, the research uses the directed graph method. A directed graph or digraph is a set of vertices and a collection of directed edges that connect a pair of vertices ordered by each. It's a pointing edge from the pair's first vertex and points to the pair's second vertex.

Mirai botnets attackers use several techniques of obfuscation such as domain generation, domain fluxing and more. To create it even more difficult for anti-virus products to detect their infrastructure. Some Mirai botnets leverages existing innocent server's safety vulnerabilities to distribute the real malicious code. This usually makes Mirai botnets detection difficult for anti-virus products when they are hosted on legitimate servers.

After run the sample, the report produced by cuckoo has several analytical reports. The report to be analysed in this research is the behaviour analysis. Then, there was plenty of process and detection sources that were born with the Mirai botnets in the behaviour analysis report. So, cmd.exe and svchost.exe (WBEM) are the process to take their registry as sources of detection. It is the deleted registry done by the process based on Table 1.

The Windows API informally WinAPI is the key set of application programming interfaces (APIs) accessible in the operating systems of Microsoft Windows. Collectively, the name Windows API relates to several distinct platform applications often referred to by their own names (e.g. Win32 API); see the section on variants. Nearly all Windows programs communicate with the Windows API. A tiny amount such as programs launched early in the Windows startup process use the Native API on the Windows NT operating system line. The API called in process cmd.exe and svchost.exe listed in Table 2, and the number indicates as their id that will be used in the representation of graphs.

Table 1: Deleted Registry.

Process	Deleted Registry
cmd.exe	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Setting\ZoneMap\ProxyBypass HKEY_LOCAL_MACHINE\SOFTWARE\Wow643Node\Microsoft\Windows\CurrentVersion\Internet Setting\ZoneMap\ProxyBypass HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Setting\ZoneMap\IntranetName HKEY_LOCAL_MACHINE\SOFTWARE\Wow643Node\Microsoft\Windows\CurrentVersion\Internet Setting\ZoneMap\IntranetName
svchost.exe	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WBEM\CIMOM\LastServicesStart

Table 2: List of API.

API	Function
NtOpenKey	Opens an existing registry key.
NtQueryValueKeyEx	Returns a value entry for a registry key.
RegCloseKey	Closes a handle to the specified registry key.
RegDeleteValue	Removes a named value from the specified registry key
RegEnumKeyEx	retrieves information about one subkey each time it is called.
RegEnumKey	Retrieves the name of one subkey each time it is called.
RegOpenKeyEx	Opens the specified registry key.
RegQueryInfoKey	Retrieves information about the specified registry key.
RegQueryValueEx	Retrieves the data associated with the default or unnamed value of a specified registry key.
RegSetValue	Sets the data for the default or unnamed value of a specified registry key.
NtQueryKey	Provides information about the class of a registry key, and the number and sizes of its subkeys(user mode).
NtQueryMultipleValueKey	Retrieves values for the specified multiple-value key.
RegCreateKeyEx	Creates the specified registry key. If the key already exists, the function opens it.
RegNotifyChangeKeyValue	Notifies the caller about changes to the attributes or contents of a specified registry key.
NtDeletekey	Deletes an open key from the registry (user mode).
NtEnumeratekey	Returns information about a subkey of an open registry key(user mode).
RegDeleteKey	Deletes a subkey and its values.

The graph displays the full cycle of the method that is cmd.exe and svchost.exe referring to the behaviours analysis report (registry) from the start of the API to the end. Figure 5 is the full cycle for cmd.exe.

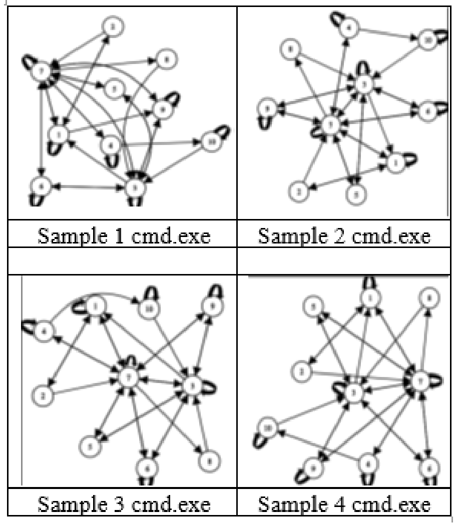
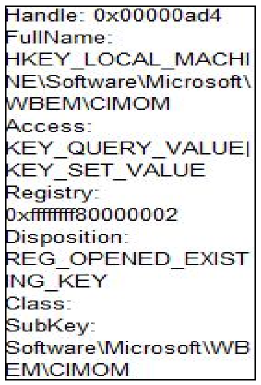


Figure 5: Full cycle for cmd.exe

The deleted registry cycle is extracted from full cycle of the process which is cmd.exe and svchost.exe. The cycle of deleted registry is depending on handle's value as Figure 6.



```
Handle: 0x00000ad4
FullName:
HKEY_LOCAL_MACHI
NE\Software\Microsoft\
WBEM\CIMOM
Access:
KEY_QUERY_VALUE|
KEY_SET_VALUE
Registry:
0xffffffff80000002
Disposition:
REG_OPENED_EXIST
ING_KEY
Class:
SubKey:
Software\Microsoft\WB
EM\CIMOM
```

Figure 6: Handle from behavior analysis report (Registry).

For example, with several APIs, RegOpenKeyEx, RegCreateKeyEx and NtOpenKey, the deleted registry cycle can start and end with RegCloseKey. Based on Figure 7 form part of the cmd.exe cycle which is the particular cycle of the deleted registry Mirai botnets behaviours.

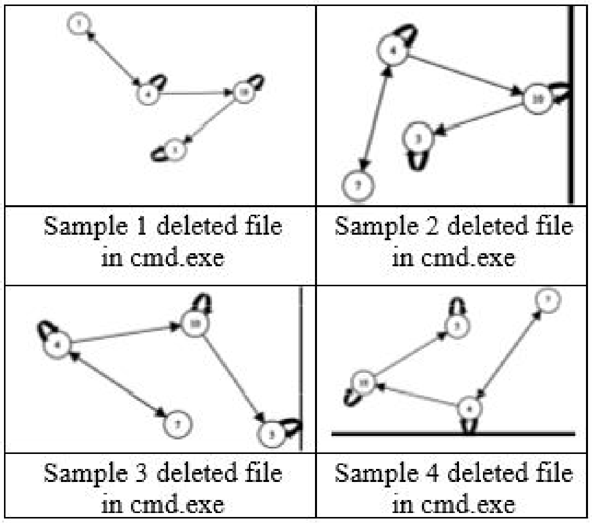

Sample 1 deleted file in cmd.exe
Sample 2 deleted file in cmd.exe
Sample 3 deleted file in cmd.exe
Sample 4 deleted file in cmd.exe

Figure 7: Deleted file in cmd.exe.

In addition, svchost.exe's full cycle is based on Figure 8. While Figure 9 form part of the svchost.exe cycle, which is also the specific cycle of the deleted registry Mirai botnets behaviours. The number at the edges is represented as the cycle sequence.

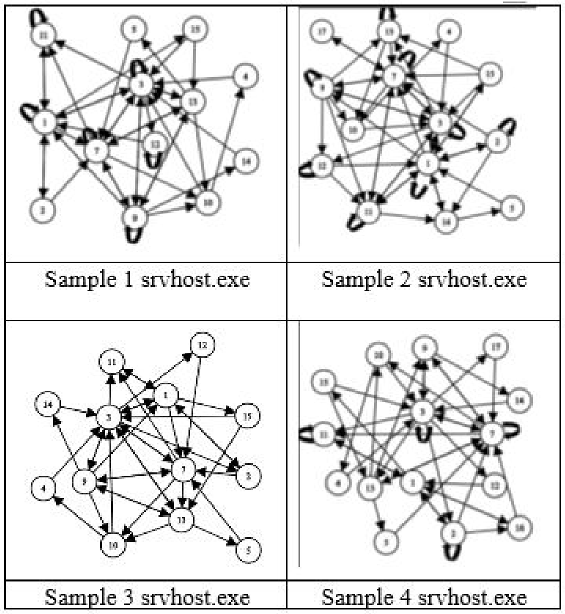


Figure 8: Full cycle of srvhost.exe

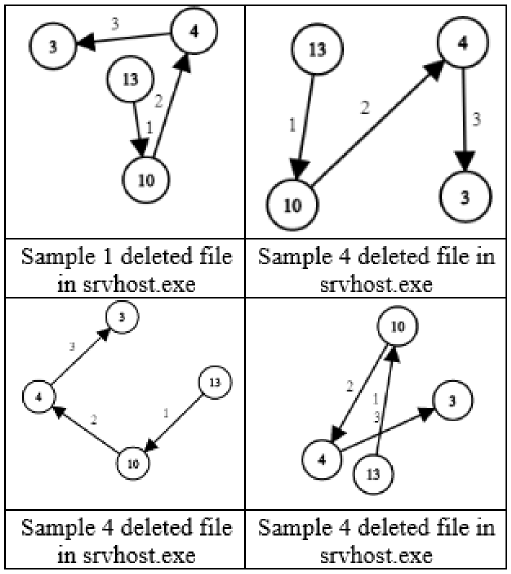

Sample 1 deleted file in srvhost.exe
Sample 4 deleted file in srvhost.exe
Sample 4 deleted file in srvhost.exe
Sample 4 deleted file in srvhost.exe

Figure 9: Deleted file in srvhost.exe

The deleted registry cycle always begins with either RegOpenKeyEx or RegCreateKeyEx and ends with RegCloseKey. As shown in Table 3, all possible connection of APIs in the cycle of deleted registry. The edges are depicted as the next activity or API to be called by the Mirai botnets.

Table 3: Relation between API.

API	Explanation
RegOpenKeyEx → RegQueryValueEx	Open the specified registry key then it will retrieve the data associated with the default or unnamed value of a specified registry key
RegOpenKeyEx → RegDeleteValue	Open the specified registry key then it will remove a named valued from the specified registry key
RegCreateKeyEx → RegSetValueEx	Create the specified registry key, but if the key already exists, the function opens it then it will set the data for the default or unnamed value of the specified registry key.
RegDeleteValue →RegDeleteValue	Remove a named valued from the specified registry key but if fail, it will try another named value to be remove.
RegDeleteSet → RegSetValueEx	Remove a named valued from the specified registry key then it will set the data for the default or unnamed value of the specified registry key.
RegSetValueEx →	Set the data for the default or

API	Explanation
RegSetValueEx	unnamed value of the specified registry key then it will set another data.
RegSetValueEx → RegCloseKey	Set the data for the default or unnamed value of the specified registry key then it will close a handle to the specified registry key
RegCloseKey → RegCloseKey	Closes a handle to the designated registry key, it also closes another handle if there is a handle present that needs to be closed.

As the goal of this studies to achieve the pattern of Mirai botnets behaviours and also to achieve the resemblance of each Mirai botnets sample. Based on the study conducted, the sample of Mirai botnet shas its own behaviours but also has similarity. The cycle of cmd.exe from start to finish is the same with another sample. In other words, the behaviours of Mirai botnets on the cycle of cmd.exe will be the same for API call (refer to Table 2) and registry path as Figure 10.

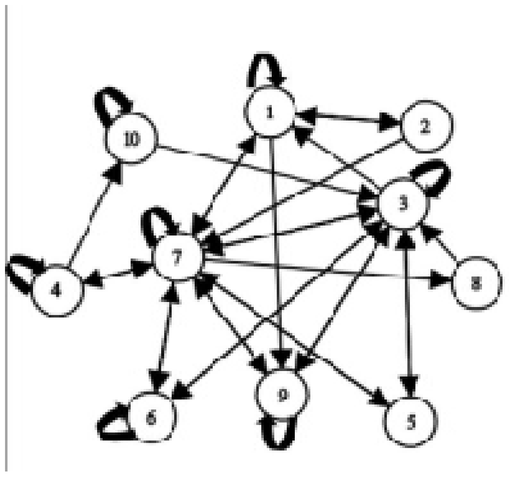


Figure 10: Mirai botnet behaviors pattern in cmd.exe

On the other hand, svchost.exe has a distinctive sequence from sample to sample, but svchost.exe has its pattern for the deleted registry cycle. The API consist in the cycle is RegCreateKeyEx, RegSetKeyEx, RegDeleteKeyEx and RegCloseKey. The pattern of sequence Mirai botnets behaviours as shown in Figure 11.

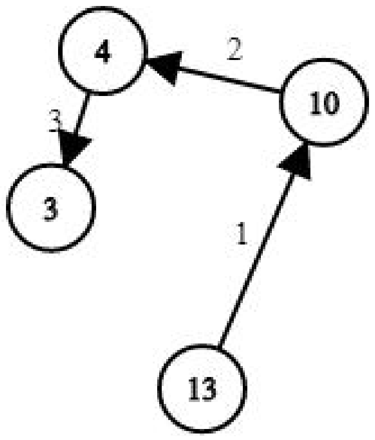


Figure 11: Pattern of deleted registry behavior cycle in svchost.exe

The result of this research shows the pattern of Mirai botnets behaviour pattern using a graph theory approach. In this research, the analysis uses the subject of graph-modeling to identify patterns of Mirai botnets behaviors based on samples from virusshare. The research creates a graph that identifies the behaviors of the Mirai botnets. The outcome of the research shows that the Mirai botnets has its unique behavior, the resemblance still appears somehow. The Mirai botnets tend to delete the registry using the same technique or process. The solution is also instrumental in studying the infrastructure of some Mirai botnets and revealing widespread patterns among Mirai botnets. Thus, these results show the accuracy and efficiency of the technique with data about real-world Mirai botnets classification.

Therefore, the significant output of this study is the development of a clustering model to evaluate and detect Mirai botnets by using graph theory approach strategy to identify Mirai botnets behaviour based on behaviour analysis report produced by cuckoo sandbox. The research's contributions is extract the similarity of Mirai botnets behaviour by formulating registry information. Then, the Mirai botnets behaviour pattern is visualized using graph theory approach.

5. CONCLUSION

This paper presented results of Mirai botnets attack pattern using graph theory approach. These findings enhance our understanding of Mirai botnets behaviors and their attack activities. This research has found that generally it provides excellent advantage to the society by offering security awareness regarding malware detection. Future research should therefore concentrate on the investigation of different Mirai botnets types. More research is required on determine the characteristics for better evaluation of Mirai botnets. Further research also might explore to use powerful software and hardware efficiency to optimize Mirai botnets analysis.

ACKNOWLEDGEMENT

This work has been supported by research grant GLUAR/APNIC/2018/FTMK-CACT/A00018. Thank you to research group of Information Security Forensics and Computer Networking (INFORSNET), Center for Advanced Computing Technology (C-ACT), Fakulti Teknologi Maklumat dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka (UTeM).

REFERENCES

1. Oliveri, A., & Lauria, F. (2019). Sagishi: an undercover software agent for infiltrating IoT botnets. Network Security, 2019(1), 9–14. https://doi.org/10.1016/S1353-4858(19)30009-1
2. Singh, K., Guntuku, S. C., Thakur, A., & Hota, C. (2014). Big Data Analytics framework for Peer-to-Peer Botnet detection using Random Forests. Information Sciences, 278, 488–497. https://doi.org/10.1016/j.ins.2014.03.066
3. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Arbor, A., Bursztein, E., … Zhou, Y. (2017). Understanding the Mirai Botnet This paper is included in the Proceedings of the Understanding the Mirai Botnet.
4. Chen, C. M., & Lin, H. C. (2015). Detecting botnet by anomalous traffic. Journal of Information Security and Applications, 21, 42–51. https://doi.org/10.1016/j.jisa.2014.05.002
5. Kaur, Tejvir Malhotra, Vimmi Singh, D. (2014). Comparison of network security tools-Firewall, Intrusion Detection System and Honeypot. International Journal of Enhanced Research in Science Technology & Engineering, 3(2), 200–204.
6. Chowdhury, S., Khanzadeh, M., Akula, R., Zhang, F., Zhang, S., Medal, H., … Bian, L. (2017). Botnet detection using graph-based feature clustering. Journal of Big Data. https://doi.org/10.1186/s40537-017-0074-7
7. Amin, M. S., Chiam, Y. K., & Varathan, K. D. (2019). Identification of significant features and data mining techniques in predicting heart disease. Telematics and Informatics, 36(November 2018), 82–93. https://doi.org/10.1016/j.tele.2018.11.007
8. Pektaş & Tankut, (2019). Internet of Things: A survey on machine learning-based intrusion detection approaches. Computer Networks, 151, 147–157.
9. Ahmadi, Sami, Rahimi, & Yadegari, (2018). Classification and interaction in random forests. Proceedings of the National Academy of Sciences. https://doi.org/10.1073/pnas.1800256115
10. Lanzi, Balzarotti, Kruegel, Christodorescu, & Kirda, (2018). New Iot Botnet Torii Uses Six Methods for Persistence, Has No Clear Purpose. Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012. https://doi.org/10.1109/FIT.2012.53
11. Fredrikson, Christodoresu, Sailer, & Yan (2017). A K-nearest neighbor classifier for ship route prediction. OCEANS 2017 - Aberdeen.
12. Ding, Xia, Chen, & Li (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502
13. Nikolopoulos & Polenakis, (2017). Machine Learning and Images for Mirai botnet Detection and Classification. https://doi.org/10.1145/3139367.3139400
14. Lee & Lee (2017). A comparative study of static, dynamic and hybrid analysis techniques for android Mirai botnet detection. International Journal of Engineering Development and Research.
15. Park, Reeves, Mulukutla, & Sundaravel (2017). Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering. https://doi.org/10.1155/2017/9324035
16. Shang, Zheng, Xu, Xu, & Zhang (2010) Research on the architecture of Internet of Things. ICACTE 2010 - 2010 3rd International Conference on Advanced Computer Theory and Engineering, Proceedings. https://doi.org/10.1109/ICACTE.2010.