A Cloud-Based Energy Efficient Hosting Model for Malware Detection Framework

Qublai K. Ali Mirza
Faculty of Business, Computing, & Applied Sciences
University of Gloucestershire
Cheltenham, United Kingdom
Email: qalimirza@glos.ac.uk

Irfan Awan
School of Electrical Engineering & Computer Science
University of Bradford
Bradford, United Kingdom
Email: i.u.awan@bradford.ac.uk

Muhammad Younas
School of Engineering, Computing, & Mathematics
Oxford Brookes University
Oxford, United Kingdom
Email: m.younas@brookes.ac.uk

***Abstract*—Protection strategy against malware attacks, whether in enterprise networks or on personal computers is generally divided into two main aspects; the detection rate and the resource consumption. Having a better detection rate is not the solution unless it is complimented by energy efficiency with respect to its CPU resource consumption of the host machine. This paper presents a cloud-based energy efficient hosting model for an intelligent malware detection framework, which amalgamates different components of Amazon's cloud services to develop a unique and dynamically scalable hosting model. The hosting model is then evaluated separately for its CPU utilization and in comparison with the conventional security software. The experimental results demonstrate significant energy efficiency in terms of CPU utilization by the hosting model.**

1. Introduction

A general perception about computer security is that it only requires a solution that detects an attack and eradicates it [1]. In fact, in the current age of technology, securing an entire network or even an individual computer from a malware attack has become significantly resource intensive [2]. If the emphasis is to acquire licenses and tools for advance network protection and monitoring then undoubtedly security of such network can be enhanced [3]. However, such sophisticated tools require a significant amount of dedicated resources, such as; hosting servers, trained personnel, network bandwidth, etc. [4]. Similar to the enterprise networks, a general user experience the similar threat, which can only be managed by acquiring a periodic subscription of an antivirus [5].

There is a known tradeoff when both security and energy or resource efficiency are of primary concern and it is crucial that how the impact of this tradeoff can be appropriately managed and minimized [6]. In [7], detection and prevention abilities of antiviruses were thoroughly evaluated, which highlighted the significantly low detection percentage of mainstream antiviruses. However, along with the deficiency of malware detection abilities, such security software consume a substantial amount of host machines' resources while operating in a scan mode [8]. Operating systems allocate more resources to the high priority tasks or software. Given that antiviruses, while running in a scan mode have the highest priority, operating system is left with fewer resources that makes the overall system more vulnerable during that period. This implies that along with having a low detection ratio, antiviruses actually make the host system more vulnerable against different external threats while operational [9]. This paper presents a cloud-based distributed and energy efficient hosting model for an intelligent malware detection framework, which is built on our previous research [7]. The hosting model presented and evaluated in this paper is a unique amalgamation of different services of Amazon's cloud infrastructure, specifically designed for the aforementioned framework.

The main contribution of this paper is to design, develop, and evaluate an energy efficient hosting model by using Amazon's cloud platform, for an intelligent malware detection framework, which is proposed and discussed in an earlier research. The excess CPU resource consumption identified in the evaluation of mainstream antiviruses, sets the scale to identify how resource efficient is the proposed hosting model.

The remainder of the paper is structured as follows: Section two of this paper presents an evaluation of conventional antiviruses with respect to their CPU resource consumption while operating in scan mode. Section three presents the design and architecture of the hosting model. Section four presents the performance evaluation of the model along with the discussion on different aspects of evaluation. Finally, section five concludes the discussion.

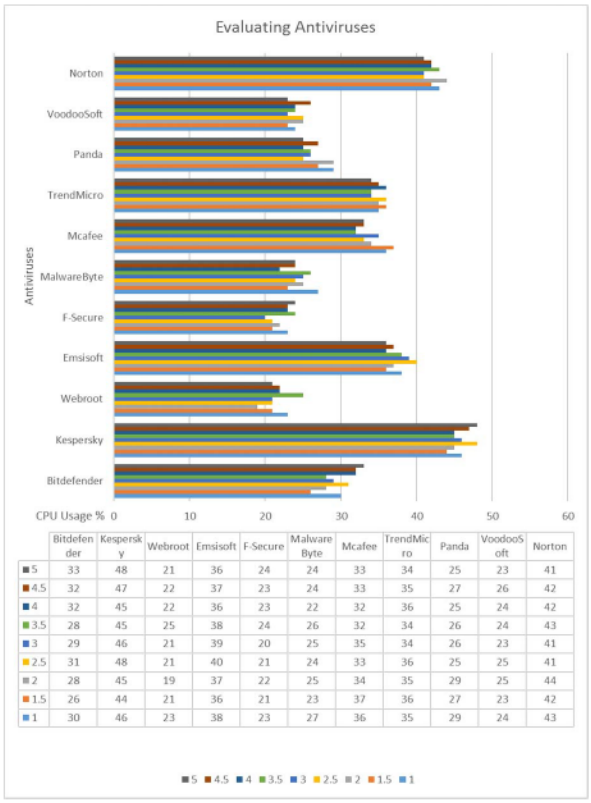
2. Evaluation of Conventional Antiviruses

This section presents an evaluation of mainstream conventional antiviruses with respect to their CPU utilization along with their impact on the host system. Majority of the mainstream conventional antiviruses are host-based, which mainly implies that the entire processing required to identify a malicious file will be based on the host machine. This include, generation of signatures, storage, matching process, and other resource intensive tasks on the host machine.

There are a few mainstream antiviruses that claim to be cloud-based, however, a lot of their resource intensive tasks are executed in the client's end. For the purpose of this evaluation, eleven mostly used antiviruses were selected and a repository of clean and malicious files were used for the experiment. A clean installation of Windows 7 was used as a host environment on eleven virtual machines with similar hardware and network configurations.

The experiment, which was performed on concurrent eleven computers lasted roughly for five hours and performance were recorded after every thirty minutes. The primary reason for evaluating the performance after a thirty minute interval was to check whether the security software was starting the scan with a higher resource signature and then consuming less resources later in the process. Fig. 1 presents the outcome of the experiments, which illustrate some astonishing results for all the antiviruses that were evaluated. The results illustrated in Fig. 1, present the performance for every thirty minute interval for each antivirus. As per the results, the resource consumption of each antivirus after a thirty minute interval was inconsistent throughout the entire five hours of experiment. Another significant aspect was the evaluation result of the high-profile antiviruses, such as; Norton, Kaspersky, and Bitdefender. Both Norton and Kaspersky mostly stayed in mid to high 40% of CPU consumption throughout the experiment cycles, however, Bitdefender started with 30% went down to mid-twenties and climbed back up to 33% towards the end. Furthermore, the antiviruses, such as; VoodooSoft, Panda, MalwareByte, F-Secure, Emsisoft, Webroot did consume between 20-30%. However, consuming a quarter of the CPU resources still is a significantly higher rate while detection significantly lower malware samples, as discussed in [7].

The experiments, which evaluated the mainstream antiviruses confirm the initial hypothesis along providing a range for lower and upper ranges of resource consumption, which will be used as benchmarking parameters for the hosting model proposed in this research. The proposed model can only be considered successful if the overall resource consumption is lower than the lowest CPU consumption in Fig. 1.



	Bitdefender	Kespersky	Webroot	Emsisoft	F-Secure	Malware Byte	Mcafee	TrendMicro	Panda	VoodooSoft	Norton
5	33	48	21	36	24	24	33	34	25	23	41
4.5	32	47	22	37	23	24	33	35	27	26	42
4	32	45	22	36	23	22	32	36	25	24	42
3.5	28	45	25	38	24	26	32	34	26	24	43
3	29	46	21	39	20	25	35	34	26	23	41
2.5	31	48	21	40	21	24	33	36	25	25	41
2	28	45	19	37	22	25	34	35	29	25	44
1.5	26	44	21	36	21	23	37	36	27	23	42
1	30	46	23	38	23	27	36	35	29	24	43

Figure 1. Antivirus Evaluation Chart

3. The Hosting Model

This section presents the design and architectural details of the hosting model along with the Amazon's cloud components used to host the hosting model. The energy efficiency of the hosting model is dependent on the components used and the extent of overall scalability of the system. Along with hosting the malware detection framework, the hosting model is required to have a dynamic behavior to enable runtime scalability of the overall security system.

Before discussing the design and architectural details of the hosting model, it is pivotal to understand the overall requirements of the malware detection framework. There are two aspects of the malware detection framework, which complete its functionality. The first is the training and testing aspect and the second is the real-time malware detection, both these aspects have some common modules.

3.1. Training and Testing Aspect

The training and testing aspect of the malware detection framework is primarily for training and testing the machine learning algorithms, which facilitate the real-time malware detection once they are trained.

3.1.1. Elastic File System. As discussed in [7], there are three separate repositories required in the framework. To train the test the machine learning algorithms, the malware detection framework uses a large repository of clean and malicious files. Therefore, there is a requirement of two repositories for both clean and malicious files. The malware detection framework initially analyzes both types of files and stores them in the analysis repository, which means that there is another repository required to store the analysis files. This means that a large and scalable storage is required for this purpose, which has the ability to dynamically scale whenever is required by the framework. Elastic File System (EFS) was chosen to host these repositories because of its dynamic scalability along with its seamless integration with EC2 (Elastic Compute Cloud) and other cloud-based components. One of the requirements of these repositories is the constant updating of files, which means there is always

a fluctuation in storage use. EFS can dynamically scale up or down, which also significantly reduces the hosting cost.

3.1.2. Elastic Compute Cloud. The analysis module is powered by a customized tool, which statically analyzes a large dataset of clean and malicious files simultaneously and generates a thorough report of each file. It also integrates an external API to make an unbiased decision on the legitimacy of each file analyzed. Therefore, it requires a hosting mechanism, which not only has the potential to perform intensive analysis, it should also be able to scale up or down its computational power whenever required. EC2 not only offers both of these features, it can be dynamically replicated if the framework requires significant scalability.

EC2 is also used to host the classification module, which hosts a unique combination of machine learning algorithms that are trained and tested using the clean and malicious files in the repositories. EC2 hosts two concurrent instances of the analysis module; one for clean and one for malicious and a single instance of classification module.

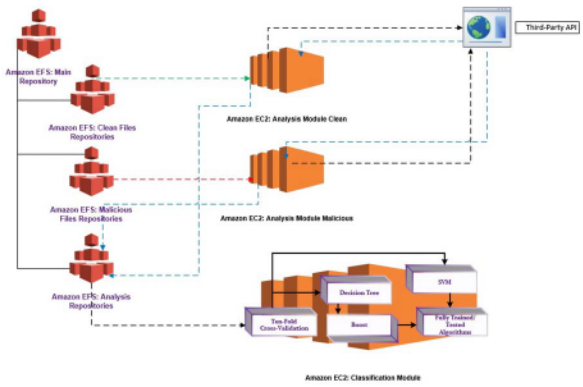


Figure 2. Training and Testing Aspect Architecture

Fig. 2 presents the architectural diagram for the training and testing aspect hosted on the Amazon's cloud components. It presents the three sub-repositories; clean, malicious, and analysis reports. The "analysis module clean" instance, hosted on EC2, retrieves the files from the clean repository, depicted with a green dotted line and the analysis reports are stored in the analysis repository. Similar operation is performed by the second instance of the analysis module "malicious", which retrieves the malicious files and stores the analysis reports in the analysis repository. Both the instances of the analysis modules integrate an external API to confirm the results and fetch additional information about the legitimacy of each file before the analysis report is generated. The analysis reports are then fetched by the classification module, which are then used to train and test the machine learning algorithms that power the entire classification module to facilitate the second aspect of the framework, which is real-time detection.

3.2. Real-Time Detection Aspect

The real-time detection aspect of the malware detection framework is based on the training and testing performed in the training and testing aspect and primarily performs the real-time detection. The following cloud components are used in the real-time detection aspect:

3.2.1. Amazon Queues. As mentioned earlier, the primary objective of this aspect is real-time malware detection, which means that it determines the legitimacy of the files sent in real-time by different clients. The primary requirement to facilitate this operation seamlessly is to use a mechanism which avoids any deadlocks and smoothly responds to each query in the most efficient manner. This also requires the hosting mechanism to ensure that each request is dealt with at most once and at least once, which eradicates the risk of system going into an infinite loop of processing similar requests. To facilitate these requirements, Amazon's SQS (Simple Queuing Service) was used. SQS is a specifically built to handle requests in queues and its completely managed by Amazon. Along with other cloud services mentioned above SQS also has elastic characteristics, which enable the queues to dynamically scale-up or scale-down based on the request traffic. It also enable different components of a cloud application to be connected together to send/receive/store requests or messages without loss of any message. It also has the capability to handle requests with embedded commands, which enabling components to trigger multiple functionalities through a single request. SQS FIFO (First-in-First-out) queues specifically ensure the sequence or in which request was received and delivered.

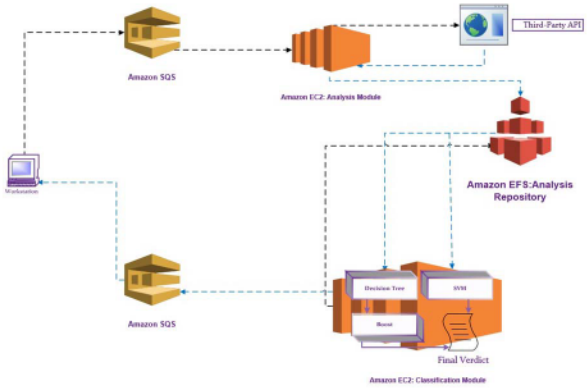


Figure 3. Real-Time Detection Architecture

SQS or queue is the only component that is not present in the training and testing aspect. As illustrated in Fig. 3, when the request is sent by the remote client, there can be hundreds of clients or more, it is received by the queues that manages the requests as FIFO, it is then sent to the available instance of the analysis module. In the real-time detection aspect there is only one type of analysis module because rather than just analyzing and storing the analysis

reports, this module identifies the legitimacy of the analyzed file and makes the decision and store the comprehensive analysis in the repository, which is used by the classification module to either confirm or reject the decision made by the analysis module based on the training of the machine learning algorithms.

Every message, which is forwarded by the queue is in XML format and contains embedded information for both the tasks; analysis and classification. The real-time detection aspect of the architecture has two queues for both request and response, as illustrated in Fig. 3. Once the decision about a file is confirmed by both the modules, it is then sent to the remote client through the response queue. Both the request and response messages are received and sent using the FIFO principle. This ensures that there is no unnecessary delays or deadlocks in the overall operations of the system.

The XML request presented in Fig. 4, show the embedded message in a hierarchy, which illustrate the priority of the jobs. Additionally, for each module to execute the relevant request, each module can only execute the task with the relevant keyword within the task = "" tag of the message.

```
<queue quid="fdssgrell921hhgbzmurtaaqp">
<job id="1" task="analysis">

    <file type="source" bucket="rep_clean">AccessDatabaseEngine_x64.exe</file>
    <file type="destination" bucket="rep_analysis">AccessDatabaseEngine_x64.JSONK</file>

    <cli>analyze -exe -JSON _SOURCE_ _DESTINATION_</cli>
    <retry_remain>3</retry_remain>

        <job id="2" task="classificatoin">

            <file type="source" bucket="rep_analysis">AccessDatabaseEngine_x64.JSON</file>
            <file type="destination" queue="quid">message.xml</file>

            <cli>classif -JSON -XML _SOURCE_ _DESTINATION_</cli>


</job>
</queue>
```

Figure 4. Request Message

The response message contains the job it was initially assigned by the request queue to identify the actual client, request was received from. The quid mentioned in Fig. 4 and Fig. 5 is actually based on the unique id that actually assigned to the remote client that sent the request along with the unique job id assigned at the time of request.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This document is a sample demonstrating result queue messages
content. -->
<!-- job_result is the root node for the document -->
<!-- The processed date time uses the date format yyyy-MM-dd HH:mm:ss. -->
<job_result processed_date_time="2017-06-28 01:01:01Z">
<!-- queue_stage_id represents at which stage the job is. -->
<queue_stage_id>1</queue_stage_id>
<!-- result_status indicates the result when an EC2
analysis & classification module attempted to process the queue task.
Values are: FAIL | ERROR | SUCCESS -->
<result_status>SUCCESS</result_status>
<!-- This is the unique hash identifier for this job. -->
<quid>fdssgrell921hhgbzmurtaaqp</quid>
<!-- An important tag to hold verdict of the analysis.
 Will be either CLEAN FILE or MALICIOUS FILE.
 The body is wrapped in a CDATA section -->
<message>CLEAN FILE</message>
</job_result>
```

Figure 5. Response Message

Fig. 5 also presents the status of the job along with the message from the framework hosted on the cloud-based hosting mechanism, which tells the client agent whether the file is clean of malicious.

4. Performance Evaluation

After the successful deployment of the malware detection framework, based on the details presented in the earlier section, the hosting model was evaluated against different parameters. This section presents a critical evaluation of the cloud-based hosting model while hosting the malware detection framework in full operational mode.

As mentioned in the earlier section, the malware detection framework has two main aspects, one for training and testing, and the other for real-time detection. This evaluation process evaluated both these aspects separately to accurately present a breakdown of the performance. It also compares the performance of the hosting model with the performance of conventional antiviruses discussed earlier in this paper. The entire evaluation process lasted for seventy-two hours and number of benign and malicious files were 75000 and 100000 respectively. The performance graphs presented and discussed in the next section were generated using Amazon's EC2 performance monitoring tool.

4.1. Training and Testing Aspect

The architecture of the hosting model for the training and testing aspect is specifically designed to handle the its operational needs, which mainly revolve around the comprehensive static analysis of a large dataset of clean and malicious files along with training and testing of three different machine learning algorithms.

Both instances of the analysis module were separately evaluated. Fig. 6 presents the CPU utilization details of the clean instance of the analysis module. It illustrates that during the first fifteen minutes of execution the analysis module consumed around 18% of the CPU power and then immediately came down to under 1%. There were few spikes in the CPU utilization during the entire evaluation period and on one instance the utilization went to nearly 15% and after a couple of fluctuations in the thirty-minute period it again came back to under 1%. It can be assumed that the reason behind few spikes in CPU utilization was because of comparatively larger files in the dataset

Evaluation of the second instance of analysis module, which hosted malicious analysis mechanism for malicious files also showed some interesting results.

As illustrated in Fig. 7, the malicious instance of the analysis module started with around 24% of CPU consumption but straightaway came down to under 1%. However, the CPU consumption of the malicious instance of the analysis module slightly consumed higher than the clean instance. The reason was the type of malicious files used in the dataset had higher level of obfuscation, which required the analysis module to first remove the layers of obfuscation to actually analyze the file. The process for removing layers of obfuscation consumed more resources.

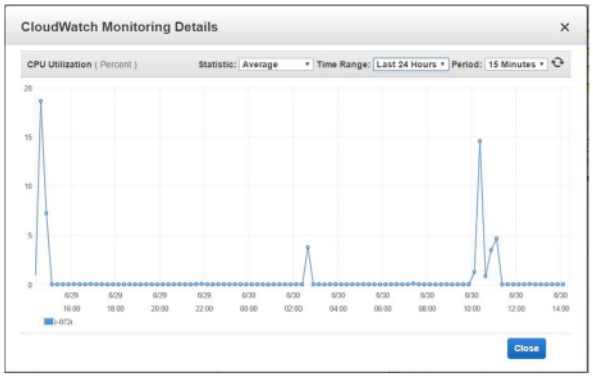


Figure 6. CPU Utilization - Analysis Module Clean

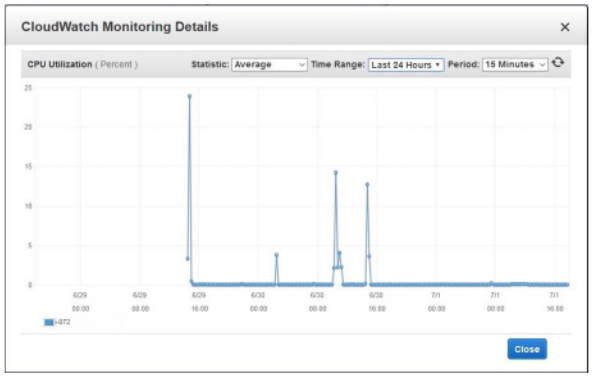


Figure 7. CPU Utilization - Analysis Module Malicious

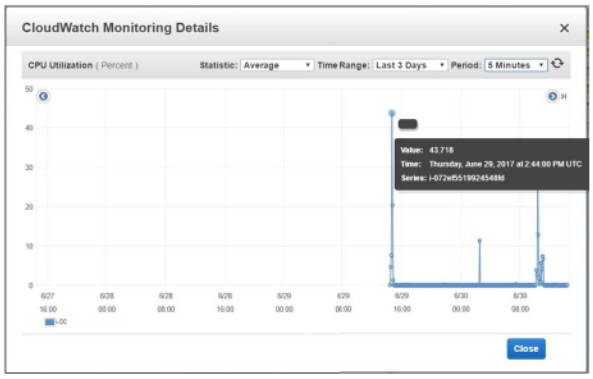


Figure 8. CPU Utilization - Classification Module

After both the instances of the analysis module were evaluated, classification module was evaluated. Fig. 8 illustrates the CPU utilization of the classification module. The graph illustrated in Figure 8 starts later than the previous two graphs and the reason is that the entire framework was triggered together and the classification module was waiting for both the analysis module to finish their analysis and populate the repositories with the reports. Once completed, the analysis reports were then used for training and testing, which is when the classification module started to show activity in the CPU utilization.

The classification module started with nearly 44% of the CPU consumption and immediately when down to under 1%. There were few spikes during the evaluation period, which suggested the testing phase of the algorithms. At the end of this evaluation phase, the training and testing aspect is finished execution and it is ready to facilitate the second aspect of the framework, which is real-time detection.

4.2. Real-Time Detection Aspect

As mentioned earlier, this aspect primarily focusses on the real-time detection. The evaluation of this aspect is different from the first aspect, it deals with the requests coming through the request queues. Additionally, for this evaluation, multiple remote clients continuously sent a combination of clean and malicious files, which were 93200 in total. The experiment lasted for twenty-four hours.

Fig. 9 presents the CPU utilization for the real-time detection aspect of analysis and classification module. As illustrated in Figure 9, orange line represent the analysis module and blue line represents the classification module. Analysis module started off with a 2% CPU usage while classification module started with 1% of CPU utilization, which basically represent that analysis module starts before the classification module in the overall framework operations. It can be seen in the graph that throughout the evaluation period, the CPU utilization for both the modules fluctuated simultaneously. The simultaneous fluctuation represent the analysis and classification cycle, which starts with analysis and ends at classification. Throughout the process, both the module utilized less than 2.5%, which represent that the framework and the cloud-based hosting model demonstrated optimum performance with the increasing cycles.

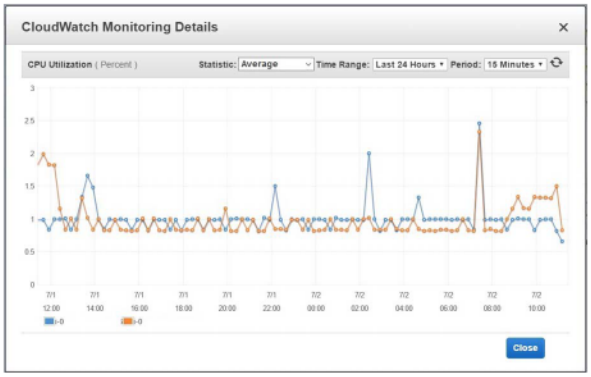


Figure 9. CPU Utilization, Real-Time Detection Aspect - Analysis & Classification Module

Additionally, another important aspect of this scenario is the unsupervised operation. The entire evaluation session lasted for more than ninety hours, during this period the framework and the hosting model handled more than 2.4 Million messages that contained very complex and multi-layered obfuscated files. This complete evaluation process not only shows the energy efficiency of the hosting model, it also shows the resilience against different type of files and distributed request management in a real-time environment.

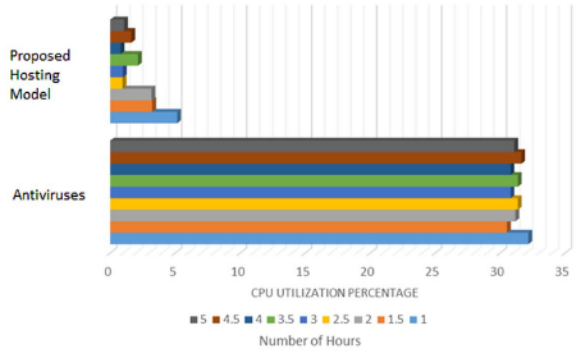


Figure 10. CPU Utilization Comparison: Antiviruses & Hosted Framework (Real-Time Detection Aspect)

The graph presented in Fig. 1, which illustrate the dataset gathered by evaluating the CPU resource consumption of antiviruses, is used to evaluate the proposed hosting model and to identify its efficiency in CPU resource consumption. The dataset of evaluating the antiviruses was acquired as a result of the initial experiment within this research, which suggested the significant weakness in the mainstream antiviruses, along with their low rate of malware detection, as discussed in [7]. Fig. 10 presents the comparison chart, which compares the hosted real-time detection aspect of the framework with the mainstream antiviruses' CPU utilization. Only the real-time detection aspect's CPU utilization was compared with the antiviruses because only this aspect performs the real-time detection, which makes it relevant to compare it with antiviruses. The graph illustrated in Fig. 10 present the average CPU consumption of eleven antiviruses listed in Fig. 1 and the CPU utilization of the second aspect operational on the cloud-based hosting model. The maximum CPU utilization of the real-time detection was 5%, however, the combine average of antiviruses was still 30%. Moreover, the CPU consumption of the second aspect of the model went down after each cycle and during the last cycle CPU utilization of the hosted second aspect was less than 1%, which makes a significant difference between the two.

Cloud platform has been used quite frequently in the recent past to enhance the level of computer/network security against malware attacks. However, the majority of the research and solutions available in this domain are focussed towards enhancing the level of security for the solutions implemented or hosted within the cloud platform. The architecture proposed in this research primarily uses the cloud platform for the purpose of building a specialized energy efficient hosting model for an intelligent malware detection framework. It is not quite possible to compare the proposed model and the outcomes of this model with a similar research in this area, as there is no research that covers the issue of CPU resource consumption of anti-malware solutions by proposing, implementing, and evaluating a dynamic cloud-based hosting model. The only possible comparison is between the CPU resource consumption of the mainstream antiviruses and the proposed hosting model, which is thoroughly discussed in this paper.

5. Conclusion

Malware detection framework discussed in [7] required a hosting model, which not only has flexibility, it should also be energy efficient in CPU utilization. This research initially evaluated the conventional mainstream antiviruses and set benchmarking numbers for the cloud-based hosting model. The said hosting model not only successfully hosted the detection framework, it also demonstrated optimum performance, which was significantly better than the conventional antiviruses. The analysis and classification module did consume between 40-45% but that was only for a very small interval. Whereas, the overall performance of the hosting mode along with the malware detection framework was extremely energy efficient. The hosting model and the malware detection framework can be further enhanced in future to cater the needs of enterprise networks by enhancing the client module to work as a network service and integrate the mechanism of intrusion detection, which will be supported by the cloud-based engine.

References

[1] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *The 5th Conference on Information and Knowledge Technology*, May 2013, pp. 113–120.

[2] A. Bisong, Syed, and M. Rahman, "An Overview of the Security Concerns in Enterprise Cloud Computing," *International Journal of Network Security & Its Applications*, vol. 3, no. 1, pp. 30–45, Jan. 2011, arXiv: 1101.5613. [Online]. Available: http://arxiv.org/abs/1101.5613

[3] J. Kołodziej, S. U. Khan, L. Wang, M. Kisiel-Dorohinicki, S. A. Madani, E. Niewiadomska-Szynkiewicz, A. Y. Zomaya, and C.-Z. Xu, "Security, energy, and performance-aware resource allocation mechanisms for computational grids," *Future Generation Computer Systems*, vol. 31, pp. 77–92, Feb. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12001823

[4] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *Journal in Computer Virology*, vol. 7, no. 4, pp. 247–258, Nov. 2011. [Online]. Available: http://link.springer.com/10.1007/s11416-011-0152-x

[5] G. Vasiliadis, M. Polychronakis, and S. Ioannidis, "GPU-assisted malware," *International Journal of Information Security*, vol. 14, no. 3, pp. 289–297, Jun. 2015. [Online]. Available: https://link.springer.com/article/10.1007/s10207-014-0262-9

[6] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds. Springer International Publishing, Sep. 2013, no. 127, pp. 86–103.

[7] Q. K. Ali Mirza, I. Awan, and M. Younas, "CloudIntell: An intelligent malware detection system," *Future Generation Computer Systems*, Jul. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17314929

[8] J. Oberheide, E. Cooke, and F. Jahanian, "CloudAV: N-Version Antivirus in the Network Cloud," University of Michigan, Tech. Rep., 2008.

[9] R. McMillan, "Is Antivirus Software a Waste of Money?" Mar. 2012. [Online]. Available: http://www.wired.com/2012/03/antivirus/