Detect and Prevention of Malware Documents in Cloud Computing Infrastructures

MULAGAPATI SRI VANI SYAMALA [#1] & L.SOWJANYA [#2] & D.D.D.SURIBABU [#3]
[#1] M.Sc Student, Master of Computer Science, D.N.R. College, P.G.Courses & Research Center, Bhimavaram, AP, India.
[#2] Assistant Professor, Master of Computer Science, D.N.R. College, P.G.Courses & Research Center, Bhimavaram, AP, India.
[#3] Head & Associate Professor, Dept of CSE, D.N.R. College of Engineering, Bhimavaram, AP, India.

Received: January 13, 2019 Accepted: February 20, 2019

ABSTRACT: *In the current days cloud computing plays a very crucial role in almost all aspects of real time environments like MNC companies, Many Industries, schools, hospitals, software companies and so on. Generally there are various types of services in the cloud computing which differ mainly with one another.One among the best of all services is Infra Structure as a Service in which infrastructure plays a vital role in forming the cloud environment. Almost cloud services are prominent within the private, public and commercial domains. All the domains are in a critical nature regarding the storage of files or documents; therefore, security and resilience are increasingly important aspects. In order to remain resilient, a cloud needs to possess the ability to react not only to known threats, but also to new challenges that target cloud infrastructures. In this paper we introduce and discuss an online cloud anomaly detection approach, comprising dedicated detection components of our cloud resilience architecture. Here we try to find out those files which are almost malware type, which can cause attack to the cloud infra structure as a service. If these type of files are found the cloud server should identify them very easily and then it should be blocked by not allowing the users to download the files from that cloud server.By conducting various experiments on our proposed protocol, we finally came to a conclusion that our proposed approach is best in identifying the malware files in the cloud and block the files not to download from the cloud server for the end users.*

Key Words*: Malware Documents, Cloud Infra Structure, Diseases, Commercial Domains.*

I. INTRODUCTION

In recent days a lot of users try to show their attention towards the cloud for their data storage as well as retrieving of data to and from the cloud server. As the data is been increasing day by day almost all the companies are unable to store their valuable data on their own individual devices, so in this situation they opt for a new data storage area known as Cloud Data Storage [1], [2]. Generally, cloud service providers allow the users to access their services for a low economical and ascendable marginal cost compared with primitive data storage services. Generally, the data which is stored in the cloud server is mainly used for sharing within the users of same group or between the users of different group with a valid authentication. Some of the best cloud data storage services are as follows: Google Drive, DriveHq Server, DropBox and ICloud. As these all are the best among various types of cloud service providers in which the data can be stored either in public cloud or private cloud, sometimes can be stored in both combine known as Hybrid Cloud.

Figure.1. Denotes the Architecture of Various Cloud Service Providers and Their Applications

From the figure 1, we can clearly find out that there are various cloud service providers that are available in the real-time environment that are used for storing various applications like word documents, pdf, excel and many more files. If you look at the above figure you can find out the various cloud service providers like Zip Cloud, Just Cloud, BOX, Google Drive, DROP BOX and a lot more. Of all these we are using DRIVEHQ.COM as the storage medium for storing the uploaded files in this proposed application.

As we all know that there are many applications of cloud computing, such as data sharing for remote systems [3], [4], [5], [6], data storage from a remote system to a centralized location [7], [8], [9], big data management systems [10], medical information system etc. All the cloud users try to access cloud-based applications or cloud server through a web browser to store or access the data to and from the cloud server. This is the main cause to launch the malware documents inside the cloud server from an a un-authorized users. Here lot of users illegally try to inject malware contents inside the cloud server and try to create problem for the cloud servers.So in order to overcome this problem we came with this proposed paper.

II. Related Work

In this section we mainly try to discuss about the various type of cloud services that are been used in the real time. Now let us look about them in detail:

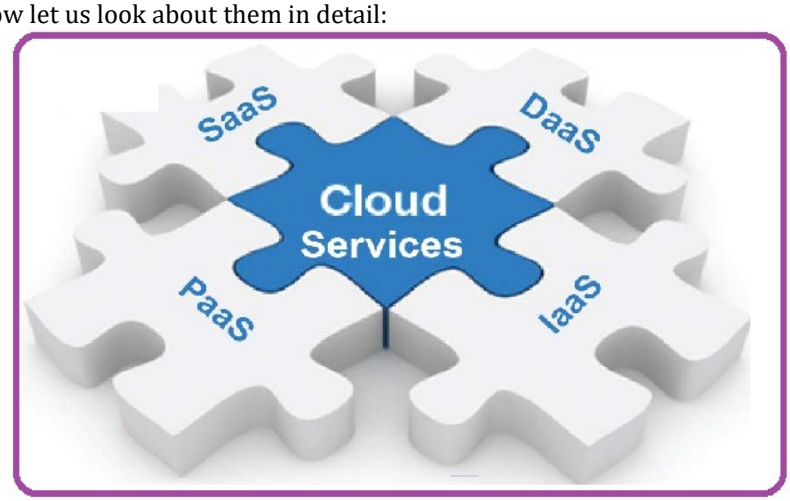

Figure.2. Denotes the Various Cloud Services that are Available in Real Time Cloud

From the above figure 2, we can clearly find out that there are four different services available and one among them is DaaS which is the main service that what we are using now for providing security for the current application that and prove that this service also gives the best security for the data which is stored inside the cloud memory locations [7]. Now let us discuss about each and every service in detail as follows:

A. IaaS (Infrastructure as a Service)
B. PaaS (Platform as a Service)
C. SaaS (Software as a Service)
D. DaaS (Data /Data Base as a Service)

A. IaaS (Infrastructure as a Service)

This is the first service out of various services that are available in the cloud. This service mainly deals with application level and it is basically used to set the infra-structure for the users[5]. This service is mainly used to create infrastructure for the set of PCs that are linked in an area. The persons who come under this service is IT Professionals, this is clearly shown in the figure 2.

B. PaaS (Platform as a Service)

The second important service in the cloud computing is Platform as a Service, where this is mainly used for customization of cloud server. Here in this service we try to set the platform for the users, where the developer comes under this service[6]. Here the cloud server customizes which type of platforms is needed for their company usage is seen in this service.

C. SaaS (Software as a Service)

The third service one among the best services in cloud computing is Software as a Service, where this is mainly used for a consumer to use the cloud service provider's applications running on a cloud IaaS[7]. Generally business end-users come under this service where all the software's that are required for running the cloud are processed in this service.

D. DaaS (Data/Database as a Service)

This is the last one among the set of cloud services that was launched and included in various cloud client services is DaaS, which is clearly seen in above figure 2. This DaaS service is used mainly for storing the data in the form of encrypted manner [10]. As this is having various advantages compared with other cloud client services, it has a small limitation like the data which is stored in this DaaS is not stored in the encrypted manner which is stored in the plain manner. Also one more limitation in this service is if any intruder or attacker try to inject any malicious contents inside this layer,no cloud servers is able to identify the malware files and block them immediately .All are almost trying to ignore such malware content despite of identifying and blocking them.So in this proposed paper we are going to identify them and block them at the cloud server level .If this file is detected by the cloud server,then the file can be blocked by the server not to be accessed or downloaded by the end users.

III. Proposed One Class Support Vector Machine Algorithm for Identifying the Malware Documents Inside the Cloud Server

In this section we try to identify the One Class SVM algorithm for identifying the malware contents inside the cloud and try to block the malware contents not to be downloaded into the application.

One class Support Vector Machine proposed by Scholkopf et al is an unsupervised learning algorithm for outlier detection. It is used for training with the collected data and then performs classification[8]. The one-class SVM formulation handles cases using unlabelled data. The main goal is to produce a decision function that is able to return a class vector y given an input matrix x based on the distribution of a training dataset. The class y contains two class where one outcome is the known class, which is the normal VM behaviour, and the other is the novel class, which represents any testing instances that are unknown to the classifier[9].

Procedure for Malware Detection

Step 1: Creation of SVM Data set in Window OS using.

Step 2: Collection of Sample malware documents from Google and Place it in a Separate List to Classify and check the performance of our Proposed Application.

Step 3: Try to inject the Malware Files into the Application

Step 4: Collection of Application Status after the Malware Document injection.

Step 5: Training and testing of workload dataset using one class Support Vector Machine algorithm.

Sample SVM Which we Used in this Current Application

```
Malware Set =
{
if (fname.toLowerCase().endsWith(".ade")
                                        || fname.toLowerCase().endsWith(".adp")

                                        || fname.toLowerCase().endsWith(".bas")
                                        || fname.toLowerCase().endsWith(".bat")
                                        || fname.toLowerCase().endsWith(".chm")
                                        || fname.toLowerCase().endsWith(".cmd")
                                        || fname.toLowerCase().endsWith(".com")
                                        || fname.toLowerCase().endsWith(".crt")
                                        || fname.toLowerCase().endsWith(".dll")
                                        || fname.toLowerCase().endsWith(".do*")
                                        || fname.toLowerCase().endsWith(".hlp")
                                        || fname.toLowerCase().endsWith(".hta")
                                        || fname.toLowerCase().endsWith(".inf")
                                        || fname.toLowerCase().endsWith(".ins")

                                        || fname.toLowerCase().endsWith(".js")
                                        || fname.toLowerCase().endsWith(".jse")
                                        || fname.toLowerCase().endsWith(".lnk")
                                        || fname.toLowerCase().endsWith(".md*")
                                        || fname.toLowerCase().endsWith(".msc")
                                        || fname.toLowerCase().endsWith(".msi")
                                        || fname.toLowerCase().endsWith(".mst")
```

```
                                        || fname.toLowerCase().endsWith(".ocx")
                                        || fname.toLowerCase().endsWith(".pcd")
                                        || fname.toLowerCase().endsWith(".pif")
                                        || fname.toLowerCase().endsWith(".pot")
                                        || fname.toLowerCase().endsWith(".reg")
                                        || fname.toLowerCase().endsWith(".scr")
                                        || fname.toLowerCase().endsWith(".sct")
                                        || fname.toLowerCase().endsWith(".shb")
                                        || fname.toLowerCase().endsWith(".shs")
                                        || fname.toLowerCase().endsWith(".sys")
                                        ||
fname.toLowerCase().endsWith(".url")||s7.toLowerCase().endsWith(".ade")
                                        || s7.toLowerCase().endsWith(".adp")
                                        || s7.toLowerCase().endsWith(".bas")
                                        || s7.toLowerCase().endsWith(".bat")
                                        || s7.toLowerCase().endsWith(".chm")
                                        || s7.toLowerCase().endsWith(".cmd")
                                        || s7.toLowerCase().endsWith(".com")
                                        || s7.toLowerCase().endsWith(".crt")
                                        || s7.toLowerCase().endsWith(".dll")
                                        || s7.toLowerCase().endsWith(".do*")
                                        || s7.toLowerCase().endsWith(".hlp")
                                        || s7.toLowerCase().endsWith(".hta")
                                        || s7.toLowerCase().endsWith(".inf")
                                        || s7.toLowerCase().endsWith(".ins")

                                        || s7.toLowerCase().endsWith(".js")
                                        || s7.toLowerCase().endsWith(".jse")
                                        || s7.toLowerCase().endsWith(".lnk")
                                        || s7.toLowerCase().endsWith(".md*")
                                        || s7.toLowerCase().endsWith(".msc")
                                        || s7.toLowerCase().endsWith(".msi")
                                        || s7.toLowerCase().endsWith(".mst")
                                        || s7.toLowerCase().endsWith(".ocx")
                                        || s7.toLowerCase().endsWith(".pcd")
                                        || s7.toLowerCase().endsWith(".pif")
                                        || s7.toLowerCase().endsWith(".pot")
                                        || s7.toLowerCase().endsWith(".reg")
                                        || s7.toLowerCase().endsWith(".scr")
                                        || s7.toLowerCase().endsWith(".sct")
                                        || s7.toLowerCase().endsWith(".shb")
                                        || s7.toLowerCase().endsWith(".shs")
                                        || s7.toLowerCase().endsWith(".sys")
                                        || s7.toLowerCase().endsWith(".url")
                                                || b.endsWith("00001111111")
                                                || b.endsWith("01111100000")
                                                || b.startsWith("111110000")
                                                || b.startsWith("01111100000")


}
```

From the above set we can clearly identify that those files which contain the text with 1's and 0's or files which contain with some extensions are identified as malware and they are blocked by the system end.

IV. Implementation Modules

Implementation is the stage where the theoretical design is converted into programmatically manner. In this stage we will divide the application into a number of modules and then coded for deployment. We have

implemented the proposed concept on Java programming language with JEE as the chosen language in order to show the performance this proposed novel IPath protocol. The front end of the application takes JSP, HTML and Java Beans and as a Back-End Data base we took My-SQL Server. The application is divided mainly into following 3 modules. They are as follows:

1) Service Provider Module

In this module, the service provider will initially try to register into the application by substituting all his basic details and then he try to login with his valid login credentials.Once he gets login into the account,he has the following functions like he can upload a set of documents into the cloud server from his account.He can view the set of files that are already available in the cloud server.He can see the alerts which was send to him from the cloud server if the uploaded files contain any malware content.Then immediately he may try to remove those malware contents from the cloud server or he try to avoid the malware contents not to be uploaded in future inside the cloud server.

2) **Cloud Server Module**

The cloud server is one who can login with his pre-defined username and password and once he gets login, then he can enter into his account and try to see all the files which are available in its server location. He can view list of service providers who got registered and he can also activate or de-activate the service providers. He can view the list of end users who got registered for participating into cloud data access. He can also see the list of malicious files which are available in the cloud server. He also has many other functions like ,sending alerts for the end users regarding the malicious files. He can send keys for decrypting the data and view the files by authorized users. He can block the malicious files not to be downloaded by the end users and a lot more.

3) End User Module

He is the one who gets register initially into the application and waits for the cloud server approval. Once the cloud server approve the end user ,then he/she can login into the system with all his basic credentails.Once he get login into his account he has the facility to search all the files that are available in the cloud server. So if he wants to download any file from the server he needs to send request to the service provider to download the data in a plain text manner. So once the file request is send to the service provider that request will be processed by the cloud server and in turn send keys for the end user for downloading the file in a plain manner. If the requested file contains no malware content ,then only the end user can download the file in a plain text manner and in turn he can download the same in his PC.If the same requested file contains any malware content ,then the file can't be downloaded by the end user even he got the decryption key from the cloud server.

V.Conclusion

In this paper, we for the first time have designed a novel method to find out those files which are almost malware type, which can cause attack to the cloud infra structure as a service. If these type of files are found the cloud server should identify them very easily and then it should be blocked by not allowing the users to download the files from that cloud server.By conducting various experiments on our proposed protocol, we finally came to a conclusion that our proposed approach is best in identifying the malware files in the cloud and block the files not to download from the cloud server for the end users.

VI. References

1. A.M. Cohen and W.R. Hersh, and R.T. Bhupatiraju, "Feature Generation, Feature Selection, Classifiers, and Conceptual Drift for Biomedical Document Triage," Proc. 13th Text Retrieval Conf.(TREC), 2004.
2. L. Kaufman, "Data security in the world of cloud computing," Security Privacy, IEEE, vol. 7, no. 4, pp. 61–64, July 2009.
3. M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, and D. Zamboni, "Cloud security is not (just) virtualization security: A short paper," in Proceedings of the 2009 ACM Workshop on Cloud Computing Security, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 97–102. [Online]. Available: http://doi.acm.org/10.1145/1655008.1655022
4. R. Bunescu and R. Mooney, "A Shortest Path Dependency Kernel for Relation Extraction," Proc. Conf. Human Language Technology and Empirical Methods in Natural Language Processing (HLT/ EMNLP), pp. 724-731, 2005.
5. R. Bunescu, R. Mooney, Y. Weiss, B. Scho¨ lkopf, and J. Platt, "Subsequence Kernels for Relation Extraction," Advances in Neural Information Processing Systems, vol. 18, pp. 171-178, 2006.
6. M. Craven, "Learning to Extract Relations from Medline," Proc. Assoc. for the Advancement of Artificial Intelligence, 1999.

7. A. Marnerides, C. James, A. Schaeffer, S. Sait, A. Mauthe, and H. Murthy, "Multi-level network resilience: Traffic analysis, anomaly detection and simulation," ICTACT Journal on Communication Technology, Special Issue on Next Generation Wireless Networks and Applications, vol. 2, pp. 345–356, June 2011.
8. J. P. G. Sterbenz, D. Hutchison, E. K. C¸ etinkaya, A. Jabbar, J. P. Rohrer, M. Sch¨ oller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," Comput. Netw., vol. 54, no. 8, pp. 1245–1265, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet. 2010.03.005
9. A. K. Marnerides, M. R. Watson, N. Shirazi, A. Mauthe, and D. Hutchison, "Malware analysis in cloud computing: Network and system characteristics," IEEE Globecom 2013, 2013.
10. M. R. Watson, N. Shirazi, A. K. Marnerides, A. Mauthe, and D. Hutchison, "Towards a distributed, self-organizing approach to malware detection in cloud computing," 7th IFIP/IFISC IWSOS, 2013.