Journal of Multimedia Information System VOL. 7, NO. 2, June 2020 (pp. 125-136): ISSN 2383-7632 (Online)
http://doi.org/10.33851/JMIS.2020.7.2.125

An Attack-based Filtering Scheme for Slow Rate Denial-of-Service Attack Detection in Cloud Environment

Janitza Nicole Punto Gutierrez[1], Kilhung Lee[1*]

Abstract

Nowadays, cloud computing is becoming more popular among companies. However, the characteristics of cloud computing such as a virtualized environment, constantly changing, possible to modify easily and multi-tenancy with a distributed nature, it is difficult to perform attack detection with traditional tools. This work proposes a solution which aims to collect traffic packets data by using Flume and filter them with Spark Streaming so it is possible to only consider suspicious data related to HTTP Slow Rate Denial-of-Service attacks and reduce the data that will be stored in Hadoop Distributed File System for analysis with the FP-Growth algorithm. With the proposed system, we also aim to address the difficulties in attack detection in cloud environment, facilitating the data collection, reducing detection time and enabling an almost real-time attack detection.

Key Words: Cloud computing, Denial-of-Service Attack, Slow Rate DoS Attack.

I. INTRODUCTION

Recently, there is a need to detect fast a network attack, especially in environments such as the cloud. Among the issues of cloud computing, the cloud is vulnerable to Denial of Service (DoS) attacks, which is the worst enemy for service availability [1]. In 2016, the Cloud Security Alliance performed a survey [2] consulting industry experts about their considerations and experience about security issues in cloud computing. According to the results, a list of 12 security concerns are the ones that worry more to the organizations and researchers, which included Denial of Service attacks. As a fact, Distributed Denial of Service (DDoS) attacks are growing each year. According to statistics published in the VeriSign Distributed Denial of Service Trends Report for Q1 2018, DDoS activity increased 53% compared to Q4 2017. Also, the 26% of mitigations in Verisign clients were to protect IT services and Cloud SaaS services [3]. Based on these statistics, DoS attacks in cloud can affect greatly to businesses. For this reason, DoS attacks should be detected fast [2]. In addition, it is important to mention that Cloud Computing services and APIs are mainly provisioned through HTTP protocol. This makes easier the access to services and resources in cloud and reduces costs. Yet, this results in cloud services being constant target of attacks that take advantage of vulnerabilities in the HTTP protocol such as HTTP DoS attacks [4].

In the reviewed solutions, we can see that there are many solutions that aim to detect DoS attack. Even if many solutions were proposed to detect these attacks, still they lack a good performance and high effectiveness. The fast detection represents a challenge as different kinds of DoS attacks are appearing and causing big damage in target systems [1] [24]. Among the reviewed solutions, researches such as [6], [7], [8], [15], [19], and [14], are solutions that implicate high complexity and it is not possible to include simple mechanisms that would prevent the analysis of all the data in real time in order to detect a DoS attack, meaning that it is difficult to reduce the DoS attack detection time. In case of the solutions in [9], [10], [11], [20], and [22], they applied machine learning algorithms that were trained with normal datasets and attack datasets. In those cases, instead of analyzing all real time data with the trained models, those solutions can apply methods to only consider suspicious

Manuscript received May 27, 2020; Revised June 15, 2020; Accepted June 17, 2020. (ID No. JMIS-20M-05-015)
Corresponding Author (*): Kilhung Lee, 232 Gongneung-ro, Nowon-gu, Seoul, 01811, Korea, 02-970-6704, khlee@seoultech.ac.kr.
[1]Dept. of Computer Science and Engineering, Seoul National University of Science and Technology, Korea, jnpunto.92@gmail.com, khlee@seoultech.ac.kr

data in the analysis with the trained models, reducing detection time without affecting the efficiency. Specifically, about Slow Rate DoS attacks, the proposed systems in [12], [13], and [14] aimed to detect those attacks but they were demonstrated to be not efficient in all scenarios but only under specific attack conditions. Talking about Apache Hadoop, the solutions in [16], [17], [18], [19], [20], [21], [22], and [23] take advantage of the ability of this tool to manage big datasets, which is generally a main requirement in case of DoS attacks and in an environment like the cloud. As an additional advantage of Apache Hadoop shown in [20], [21], and [23], it includes libraries with already implemented useful tools such as machine learning algorithms.

Based on the previously described situation, this research proposes an attack-based filtering scheme for HTTP Slow Rate DoS attack detection in cloud environment, while addressing challenges in attack detection for cloud by facilitating the collection of distributed attack evidence, enabling a real time detection and managing adequately the big volume of data.

II. RELATED WORK

In this section will include a review of previously researches. First, some works for DoS attack detection, the research in [9] presented a method that analyzes a list of traffic packet parameters and uses Naïve Bayes classification algorithm. On top of that, Information gain algorithm was applied to decrease the number of parameters considered by the algorithm. Additionally, the authors in [10] proposed the use of a back propagation neural network that was trained with data containing the CPU usage, Frame length and packet rate. Additionally, solutions for DoS attack detection in cloud were reviewed. The work in [14] proposed a solution for protecting the virtual machines (VM) in cloud by employing an Intrusion Detection System (IDS) located in the physical host of the VMs. The IDS was integrated of a packet sniffer, a feature extractor and a Support Vector Machine classifier (SVM). The results showed good accuracy except for the Slowloris attack. About Low Rate DoS attack detection, the work [12] studied the efficiency of spectral analysis technique to detect those attacks, because Low Rate DoS attacks have energy focused in lower frequencies. However, it was concluded that the efficiency of the method depends on the period in which the requests are sent. On the other hand, the research presented in [13] applied the Hilbert-Huang Transform (HHT) to the traffic data and then calculated the Pearson Correlation between the obtained HHT spectrum and the base HTT spectrum. The authors concluded the solution has a good performance only when the attacker sends traffic with a period which value is in a certain interval.

About the use of Hadoop for DoS attack detection, we can mention [19]. The proposed solution aimed to detect DoS attack with spoofed IP address. The collected network traffic was stored in HDFS and checked with MapReduce to know the authenticity of the source IP address. Additionally, a non-parametric CUSUM-based decision algorithm was used to confirm the attack. The experimental results showed that this solution was efficient in finding SYN, HTTP and DNS flooding attacks with IP spoofing. Another work in this category is [20]. Three classification algorithms were evaluated considering the volume of analyzed traffic, accuracy and delay of each one. Then, based on that, the fuzzy logic created rules that dictate the order in which those algorithms should be utilized with Apache Spark to identify an attack. Finally, works using Hadoop for security in cloud were reviewed. In [21], they proposed collecting cloud application data, using a sniffing module to only get useful information and store it in HDFS as Hive tables to apply queries and get features for machine learning models. In [22], an IDS system for cloud environment was proposed. The system depurated data with Hadoop MapReduce, identified suspicious traffic and determined the attack type by using the Random Forest algorithm. Finally, the work [23] proposed a system deployed in Apache Spark, which uses anomaly detection-based and signature-based IDS sequentially in order to detect DDoS attacks.

III. CLOUD COMPUTING

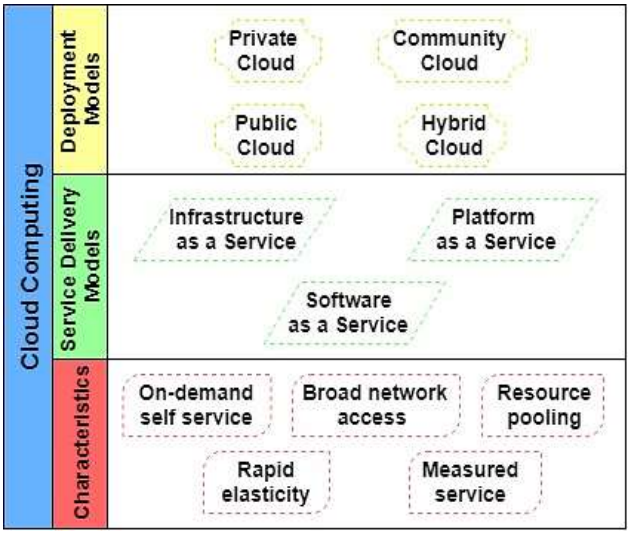

Fig. 1. Cloud Computing characteristics, service delivery model and deployment models.

According to the National Institute of Standards and Technology (NIST) [25], Cloud computing can be

described as a new computing model that permits accessing through network in an ubiquitous way while offering a group of configurable computing resources that can be shared among clients, such as servers, applications, storage and services, and according to the user's demand, so it is possible to be provisioned quickly with small intervention from the service provider. Following the previous definition, the NIST explained five characteristics of cloud computing: On-demand self-service, Broad network access, Resource pooling, Rapid elasticity, and Measured service. Additional to the definition of cloud computing, NIST described three main cloud service delivery models: Software as a Service (SaaS), Cloud Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Other concepts included in NIST document are the four cloud deployment models, which are defined according to who can use the cloud resources: Private cloud, Community cloud, Public cloud, and Hybrid.

Also, many security issues exist in cloud computing. The most common security concerns in cloud environment can be divided into four categories [26]: a) Related to the security of physical and software infrastructure of cloud service provider; b) Related to the integrity, confidentiality and privacy of data in cloud; c) Related to user authorization and authentication; and d) Related to the compliance of cloud providers with regulation. All these security problems can make cloud computing a vulnerable system. A common attack in the cloud is the Denial of Service attack, which is a threat that disturbs the data or service availability. Generally, these kinds of attacks are performed at the network/transport level or the application level. It is common that the malicious user begins to send huge quantity of requests to a service, so the cloud will provide more and more computational power or any required capability to keep the service up until exhausting all the resources [27] [28].

Any malicious behavior performed in the cloud is challenging to detect and examine because of dynamism, scalability, and virtualized nature of the cloud [29]. In that complex environment, the collection and analysis of data stored in the cloud environment are difficult. Furthermore, privacy is becoming a big worry for cloud users. Referring to a list of 65 challenges related to cloud computing security that was published by NIST, the most typical difficulties are linked to the distributed nature of the cloud components and the high quantity of users who access to the cloud services. Under such conditions, any method selected by the cloud provider with the aim of protect the system represents a vital part when collecting and analyzing information [30]. The following list summarizes the main challenges in attack detection for cloud: Physical access, Live detection, Evidence collection, and Volume of data [26] [31][32] [33] [34].

IV. DENIAL-OF-SERVICE (DoS) ATTACKS

Denial of Service (DoS) attacks can be described as the type of attack that aims to prevent genuine users from using a network resource. DoS attacks are commonly done by continuously directing a big quantity of traffic or requests to the victim [35]. These attacks also affect the network bandwidth availability by corrupting network traffic. Additionally, it has become challenging to differentiate attack traffic from non-malicious user traffic due to the likeness between each other [36]. During a DoS attack there are two types of traffic: High-rate traffic, which looks like a scenario when many users access at the same time to a system but without malicious intentions (flash crowd); and Slow-rate traffic, which looks like authentic traffic. The most common classification of DoS attacks is based on the protocol level at which the attack works:

a) Network/transport-level DoS attack: The attacker utilizes protocols in the network and transport layers to affect the target system. Usually performed with TCP, UDP, ICMP and DNS protocols [35] [5].
b) Application-level DoS attacks: In this kind of attack, the consumption of bandwidth is lower, but they are more difficult to recognize compared to the network/transport level attacks because they don't have suspicious characteristics and look more like benign traffic. These attacks leverage specific features of application protocols such as HTTP, DNS, and SIP [35].

Between the Application-level attacks, we can mention the HTTP flooding attacks, which includes four categories that are described below:

a) Slow Header attack: Also known as Slowloris attack due to the name of the tool used to perform this attack. In this case, partial HTTP requests (with incomplete headers) are sent to the target server, which constantly grow but never are closed. The result is all the available sockets being used and the web server being unreachable.
b) HTTP fragmentation attack: This attack consists on establishing a HTTP connection with the target web server, dividing a non-malicious packet in fragments with smaller sizes and directing them slowly to the target. By doing this, the attacker can keep multiple connections open for a long period. Usually, the interval between each sent packet is defined according to the server time out.
c) Slowpost attack: It is like the previous attack but in this case the attacker indicates the content-length

value in the first packet and then slowly sends HTTP post commands with small data. The web server usually waits until the value indicated in the content-length field is reached.

d) Slowreading attack: The attacker reads the server's response slowly by indicating a smaller receive window-size than the victim web server's send buffer. In this case, the TCP protocol keeps open connections even when there is no data transferred so the attacker can obligate the victim to retain many connections open.

V. APACHE HADOOP

Apache Hadoop [37] is an open-source project with the purpose of offering software that can be used as a reliable, scalable and distribute computing tool. This project includes the Apache Hadoop software library, which can be understood as a framework that permits distributed processing of big data sets running in groups of computers, based on simple programming models. Those computers have computation and storage as available resources; however, since the machines can have failures, the Hadoop software library is able to detect and handle them, guaranteeing high availability. This project also includes other components which will be described since they were useful tools in the process of developing the solution presented in this research.

Hadoop Distributed File System (HDFS) [38] is a distributed file system and has as a main attribute the possibility of accessing to it with high throughput. HDFS was thought as a system able to function on average computers and hardware components. The main characteristics of HDFS are fault tolerant, large files management, and computation closer to data. HDFS possess an architecture based on master and slave roles. Usually, an HDFS cluster contains one NameNode and various DataNodes. A NameNode is the component that plays the master role by controlling the file system namespace, monitoring the file access and keeping other metadata. It is also in charge of operations such as opening, closing and renaming files or directories. On the other hand, a DataNode is limited to control the data contained in its own machine. Following the slave role, it attends any user request to read or write data. Like traditional file systems, there is a namespace that permits the storage of user data in files. However, even if the user sees the file under one directory, the stored file is divided in blocks which are saved in a group of DataNodes. The mapping between blocks and DataNodes is managed by the NameNode. In this scenario, the creation or deletion of a data block is done by the DataNodes according to the NameNode requirements. Also, the important security features of HDFS are re-replication and data integrity.

In Hadoop environment there is other important component named Apache Spark [39]. It can be described as a quick engine for computation of stored information in Hadoop, which can be used to build application such as Extract-Transform-Load models, machine learning, stream processing, and graph computation. Usually, Spark is deployed in a cluster of machines which makes it fast and useful for many purposes. It is considered a group of applications running independently on a group of computers. Each application is controlled and managed by a SparkContext object, which should be created in the main program of the application and works as a driver program and coordinates with a cluster manager, like Apache Mesos or Hadoop Yarn. By doing this, the application controlled by the SparkContext can get some computation power and memory. After the assignation of resources, the cluster manager transfers the application code to the executors so the SparkContext can entrust tasks to them. Another characteristic that should be highlighted is the availability of high-level APIs written in Java, Scala, Python and R, which permit the development of applications in Spark. Among the supported tools, we can mention Spark SQL, MLlib, GraphX and Spark Streaming. Spark Streaming is a tool considered an extension of the Spark core that permits the processing of data streams being generated in real time. This tool can be extended, allows high throughput and is resistant to system faults. The data streams can come from diverse sources including Kafka, Flume and general TCP sockets.

Apache Flume [40] can be described as a distributed system that collects, aggregates and transfers big amount of data from numerous diverse sources to a centralized storage. It can integrate with HDFS and keep the main concepts of Hadoop such as reliability and availability. Commonly, Apache Flume is used as a tool for log data collection and aggregation. However, Flume can also work with different types of data like network traffic data, data from social media and email messages. Flume manage all the processes of collection, aggregation and transference of data from origin to destination by using a process called Agent. Also, the data flow being transferred is represented in many Events, each one of them carrying a byte payload and other optional attributes. Inside the agent, there are three main components: Source, Channel and Sink. Among the characteristics of Apache Flume, it is important to mention the high reliability that possess. Another important feature of Apache Flume is the recoverability in case of failure.

Journal of Multimedia Information System VOL. 7, NO. 2, June 2020 (pp. 125-136): ISSN 2383-7632 (Online)
http://doi.org/10.33851/JMIS.2020.7.2.125

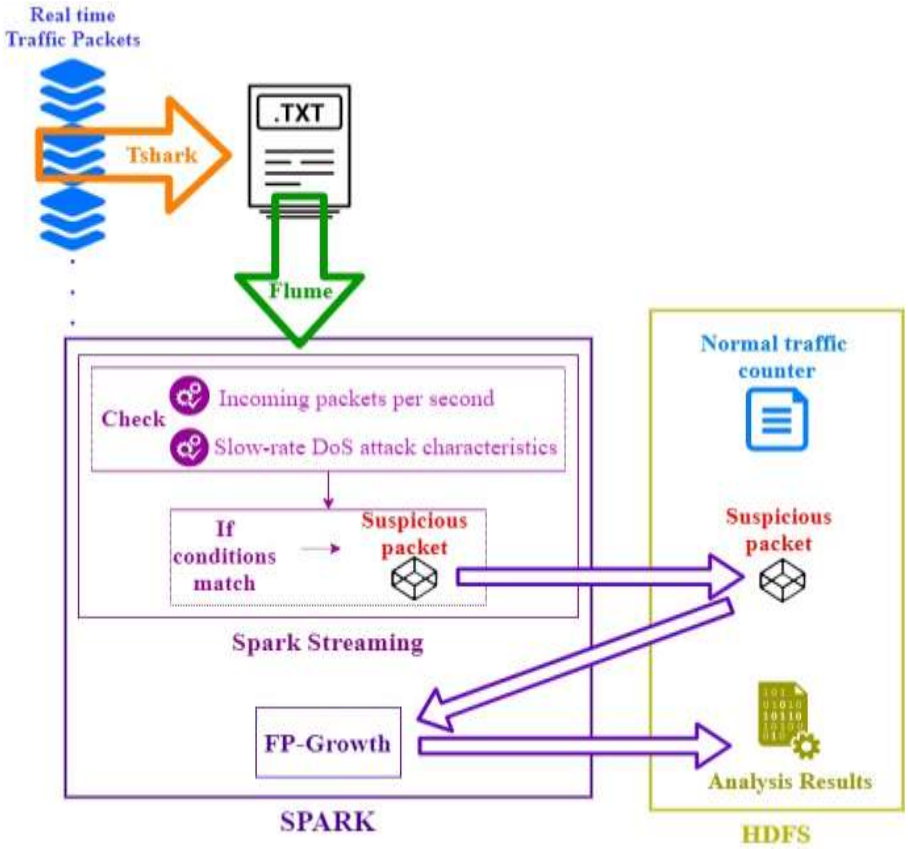

Fig. 2 The proposed solution flow.

VI. PROPOSED SOLUTION

The proposed solution is composed of the process of collection and filtering traffic packets data in order to take in consideration only the packets related to a request flooding HTTP DoS attack and the packets related to a Slow Request/Response HTTP DoS attack. These Slow Rate DoS attacks become more effective if the attackers make use of reflectors or botnets that replicate the same behavior of each attack, resulting in the exhaustion of server resources. For this reason, it is important to mention that the proposed solution can identify the beginning of these attacks, so it permits a rapid detection and helps the cloud administrator to take measures in order to mitigate the attack. Additionally, another important point is that in this case the implementation also permits finding possible flooding attacks that uses other protocols such as TCP, UDP or ICMP.

First, it is required as a preliminary step the collection of traffic packets to obtain traffic packets data in a normal state (without attacks), which will be used to make a comparison with the real time traffic. In this case, the data collection should be done during a year since there are fluctuations in the network according to the season. Moreover, the number of traffic packets per second is calculated and stored in HDFS Spark can be used for this preliminary step. After that, Flume will collect text files that contain the traffic packets data generated in real time. The Flume configuration file in this process is using the source TailDir and the sink Avro Sink. The mentioned text files are the result of running tshark and getting the packets fields of date and time, source IP address, source port, destination IP address, destination port, protocol, frame length, HTTP method, window size, and other packet information (field col.Info), considering only packets coming to the web server.

After the collection with Flume, the traffic packet data is analyzed with Spark Streaming to apply a filter, get data only from suspicious traffic packets and store it in HDFS. The filtering process was implemented with python and considers the conditions explained in section 2.4. The steps are explained below:

a) Read the file containing information about the number of packets per second in a normal state
b) Splits the traffic packet information in order to get the packet fields (date and time, IP address, etc.)
c) Calculate the number of packets per second and checks if at least one of the following conditions matches:
   - ✓ The number of packets per second coming in real time is higher than the number in a normal state in at least a specified threshold t_{time}. It is important to

mention that this first condition permits finding packets suspicious to be part of a request flooding DoS attack.

- ✓ The protocol is HTTP, the method is POST and the frame size is less than 1000 bytes. This is considered as a threshold t_{size} that can be modified by the cloud administrator.
- ✓ The protocol is TCP, the window size is 0 and the packet information (col.Info) includes the text "ZeroWindow".
- ✓ The protocol is HTTP but the packet information (col.Info) indicates the message "non-HTTP" since the HTTP headers are not complete.

in detail would be t_{time} = 1.5 * (max_req) – (max_req) = 0.5 * (max_req).

Finally, after the filtering process, the suspicious packets data is read from HDFS and analyzed by Spark. The implementation of this last step was done with python and the FP-Growth algorithm was employed. Directories including packet data from the last 30 minutes are retrieved from HDFS and, since the FP-Growth algorithm analysis is centered in transactions composed of a group of items, the packet fields are considered as the items of a transaction. However, only the following fields were in this final analysis: source IP address, source port, destination IP address, destination port, protocol, frame length, HTTP

Table 1. Tested Scenarios and Results in terms of Memory Usage, CPU Usage and Time with the process phases run sequentially.

Scenario		**Memory Usage (GB)**		**CPU Usage (%)**		**Time (seconds)**		**# of FP-Growth Results**
		Storage	FP-Growth	Storage	FP-Growth	Storage	FP-Growth	
Normal log	With Filter	9.9	9.9	73	53	17	166	68
	Without Filter	9.9	10.7	52	63	9	189	648
Attack log	With Filter	11.6	11.8	69	48	26	392	19063 (102 false positives)
	Without Filter	11.1	12.6	28	61	8	474	20392 (987 false positives)

Table 2. Tested Scenarios and Results in terms of Memory Usage, CPU Usage and Time with the process phases are run simultaneously.

Scenario		**Memory Usage (GB)**	**CPU Usage (%)**	**Time to get FP-Growth analysis results (seconds)**	**# of Stored Logs**	**# of FP-Growth Results**
Normal log	With Filter	12.9	79	79	31 (from 90872 in 6 minutes)	25
	Without Filter	12.1	61	68	8699 (in 2 minutes)	45
Attack log	With Filter	12.5	72	80	1055 (from 15607)	180 (2 false positives)
	Without Filter	12.5	64	80	11893	343 (16 false positives)

d) The results of the filtering step are stored in HDFS.

About the threshold used in this process, the definition of t_{time} is: 0.5 * (max_packets). Following the recommendations in [41], the maximum number of requests per second calculated in normal state should be multiplied by a number between 1 and 5 to set the maximum number of packets. But, in our system the threshold is calculated as the difference between the number of requests in real time and the number of requests in normal state. The calculation

method, window size and the packet information from col.Info. Based on this, the data is analyzed in periods of 5 minutes to find the most frequent item sets including the mentioned packet fields.

Still, it is important to consider the FP-Growth algorithm issue about the results that can contain sizes different from the original number of packet fields. For this reason, the python file that implements this final process only results in item sets that have more than 5 fields and have a

frequency higher than 1. Finally, since the number of results can be large even after doing the mentioned consideration, the results are sorted by frequency in decreasing order and, since there are two types of attacks identified (flooding and slow rate attacks), the results with high frequency are included in the final results to be stored in HDFS, same as the results with a frequency between 2 and 30. This second condition comes from the definition of slow rate DoS attack that mentions that the number of packets is small, so in this case 30 packets during 5 minutes may be understood as 1 packet received every 10 seconds that is a slow rate.

VII. TESTS AND RESULTS

All the testing was performed in a Docker container with Hortonworks Data Platform Sandbox version 2.6.5.0, which includes Spark 2.3.0, Flume 1.5.2 and Python 2. The container was deployed in a host machine running the operating system Linux with 64 GB of RAM memory and a CPU Intel i3 7350K 4.20 GHZ. The monitored metrics were Memory Usage, CPU usage and Processing Time. For comparative purposes, the process of the proposed solution was divided in two main phases:

a) Storage Phase: Includes log collection, filtering and suspicious logs storage.
b) FP-Growth Phase: Includes FP-Growth algorithm analysis and results storage

Tests of scenarios with and without the filtering scheme were performed with traffic packet data generated by using Tshark. In case of the normal traffic, it was composed of incoming packets to the docker container. In case of the attack traffic, it was generated by using the tool Slowloris for Slow Header DoS attack, a python file sending requests that simulate Slow Body DoS attack and the tool SlowHTTPTest for Slow Read DoS attacks. Additionally, the command ping was used to add traffic.

According to [42], there are some performance evaluation parameters that are important since they show how proficient is the proposed solution and it is possible to compare it with others. In this work, we will consider the false positive ratio, which is calculated as the ratio of legitimate packets classified as malicious and the total of legitimate packets; the performance of the system, which is related to the resources used to run the system (memory, CPU usage, time); and implementation complexity.

The first test was running both phases (Storage and FP-Growth) sequentially. The number of total packets in normal traffic were 15318 in 8 minutes. On the other hand, the number of total packets during attacks were 55388 for 16 minutes. The results of this first test are shown in the Table 1. In case of the normal traffic, the memory usage when using a filter and without filter are the same during the storage, while the memory is higher when not using a filter during the FP-Growth phase. About the CPU usage, it is higher when using a filter during storage, but it is lower when using a filter during FP-Growth phase. About the time, the use of the filter permits to process the data in a shorter period of 183, compared to the 198 seconds without any filter. Also, with filter the number of FP-Growth results are 68, which is much smaller compared to the 648 results in case of not using filter. Even if all the results represent false positives, the use of a filter permits obtaining less false positives. In case of the attack traffic, during the storage phase the memory usage with filter is higher in 0.5 GB compared to the scenario without filter, while during the FP-Growth phase it was the opposite since the memory usage with filter was less in 0.8 GB compared to the scenario without filter. About CPU usage, the situation is similar: during the Storage phase it is higher when using a filter but lower during the FP-Growth phase. About the time, when using a filter, the time is 64 seconds less compared to the scenario without filter. Additionally, the difference between the numbers of FP-Growth results in both scenarios is about 1300, but the use of a filter obtains 102 false positives from 19063 results, which is less compared to the 987 from 20392 results in the scenario without filter. If we compare to the false positive rate of [43], which is in the range of 0.15 and 0.30, our system has a much lower false positive rate. The number of legitimate packets identified as malicious can be a maximum of 102 and the legitimate ones are 45000, so the false positive rate is 0.0023, showing higher accuracy. Still, this is in the ideal situation of only having slow rate attacks and no other attacks with similar characteristics.

As second tests, the process was run with both phases (Storage and FP-Growth) simultaneously performed and traffic packet data created in real time with Slow Request/Response HTTP DoS attack and without any attack. In case of the normal traffic, with the filter the number of generated packets was 90872 in 6 minutes, while without filter the number was 8699 in 2 minutes. In case of the attack traffic, with the filter the number of generated packets was 15607 in 2 minutes, while without filter the number was 11893 in 2 minutes. The results of these tests are shown in Table 2. In this case, the metrics also include the Memory Usage, the CPU Usage and the time to get the first FP-Growth algorithm results. With the normal traffic, the scenario with filter uses more memory and CPU with a 12.9 GB and 79% respectively, compared to the scenario without filter which uses 12.1 GB of memory and 61% of CPU. About the time, the scenario with filter takes 11 seconds more compared to scenario without filter. This can

be explained because the packet flow in each scenario was captured by using the same command tshark; however, since the moments were different, it is a probability that the number of incoming packets decreased when the filter was not used. In the scenario with filter, 31 packets were found suspicious and the FP-Growth results were 25 records; while, in the scenario without filter all the captured packets were stored in HDFS and the FP-Growth results were 45. This shows that, even if the traffic load was higher during the use of the filter and the FP-Growth results don't correspond to an attack, the detection is more precise compared to the scenario without filter. With the attack traffic, the memory usage for both scenarios is 12.5 GB, but the CPU usage in the scenario with filter is higher in 8% compared to the scenario without filter. With the filter, 1055 packets were found suspicious from the total of 15607. About the time, in both cases it takes 80 seconds to get the FP-Growth algorithm results. However, with filter the number of FP-Growth results is 180 including only 2 false positive, while without filter the number is 343 results including 16 false positives. This emphasizes the increased precision when using the filter.

Finally, tests with different number of generated packets were performed to analyze the relation between the processing time and the traffic load. In case of the normal traffic, the sizes for small, medium and large traffic load contained 15318 packets in 8 minutes, 31988 packets in 17 minutes and 44951 in 24 minutes respectively. In case of the attack traffic, the sizes for small, medium and large traffic load were 24043 packets in 9 minutes, 55388 packets in 16 minutes and 84152 packets in 16 minutes respectively. The tests were performed with and without filter, and the storage and the FP-Growth phases were run sequentially, so the considered processing time is the addition of each phase processing time. In the Fig. 3 the graph shows the results of tests with normal traffic. It shows that the difference in processing times in the scenarios with and without filter is not big, but it increases as the traffic load increases. On the other hand, the Fig. 4 shows the results of test with attack traffic. Even if the difference in processing times is small with little traffic load, the difference increases faster when the traffic load increases, compared to the case of normal traffic. This graph demonstrates that the filter is more beneficial in case of an attack since it decreases the time of detection.

About the complexity of the proposed solution, we can see that, as explained in the previous section, simple conditions were applied to identify suspicious packets and then to classify them as malicious or not. For this reason, we can affirm that our proposed solution has low complexity, compared to reviewed works such as [6], [7], [8], [14], [15], and [19].

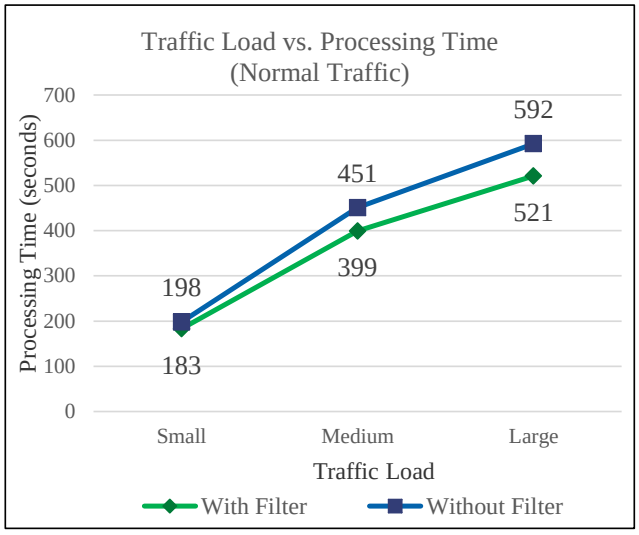

Fig. 3 Traffic Load vs. Processing Time for Normal Traffic.

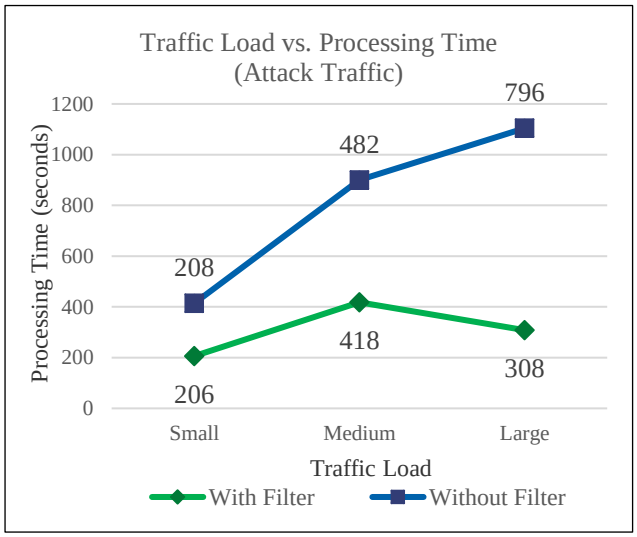

Fig. 4 Traffic Load vs. Processing Time for Attack Traffic.

VIII. CONCLUSION

In this research, an attack-based filtering scheme was proposed as a component for HTTP Slow Rate DoS attack detection in cloud environment by using Apache Hadoop components such as Apache Flume, Apache Spark and Hadoop Distributed File System. To evaluate the performance of the scheme, a complete attack detection system was implemented and tests with filter and without filter were performed. The results of testing the system in a real scenario simulation, generating data in real time and analyzing as created, showed that by using the filtering scheme the efficiency increased in 12 % when detecting Slow Rate DoS attacks by analyzing traffic packets data, due to the elimination of traffic with normal behavior preventing the DoS detection algorithm to consider them as

an attack.

In addition, the use of Apache Flume as a collection tool facilitated the distributed data collection in cloud environment, preventing data loss which was confirmed in the scenario without filtering when all the traffic packets data was stored in HDFS. The data loss prevention was possible due to the configuration offered by Apache Flume, which permits continuing with the collection even if Flume stops. Another important remark is that the reduction in time thanks to the filtering component permits the creation of almost real time detection system for HTTP Slow Rate DoS attacks, while not affecting negatively the efficiency. This means too that the detection can be done without affecting the normal functioning of the cloud since it is not necessary to stop it to perform any analysis.

Finally, it was confirmed that the use of Apache Hadoop permits the analysis of big datasets such as traffic packets data that is generated during a HTTP Slow Rate DoS attack, which was possible by applying its main characteristic of performing distributed processing, even during heavy attacks such as the generated during the tests with high traffic load.

Acknowledgement

This study was supported by the Research Program funded by the SeoulTech (Seoul National University of Science and Technology).

References

[1] T. Tamanna, T. Fatema, and R. Saha, "SDN, A research on SDN assets and tools to defense DDoS attack in cloud computing environment," in *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Chennai, pp. 1670-1674, 2017.

[2] Cloud Security Alliance, "The Treacherous 12-Top Threats to Cloud Computing + Industry Insights"; https://downloads.cloudsecurityalliance.org/assets/research/top-threats/treacherous-12-top-threats.pdf, 2017.

[3] VeriSign Distributed Denial of Service Trends Report (Q1 2018), https://www.a10networks.com/sites/default/files/A10-TPS-EB-Verisign_Distributed_Denial_of_Service_Trends_Report.pdf, 2018.

[4] M. Idhammad, K. Afdel, and M. Belouch, "Detection System of HTTP DDoS Attacks in a Cloud Environment Based on Information Theoretic Entropy and Random Forest," Security and Communication Networks, vol. 2018, Article ID 1263123, 13 pages, 2018.

[5] O. Osanaiye, Kim-Kwang R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147-165, 2016.

[6] P. Sharma, R. Sharma, E. S. Pilli, and A. K. Mishra, "A Detection Algorithm for DoS Attack in the Cloud Environment," in *Proceedings of the 8th Annual ACM India Conference (Compute '15)*, New York, pp. 107-110, 2015.

[7] V. Shah, and A. K. Aggarwal, "Heterogeneous Fusion of IDS Alerts for Detecting DOS Attacks," in *Proceedings of International Conference on Computing Communication Control and Automation*, Pune, pp. 153-158, 2015.

[8] N Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Denial of Service Attack Detection using Multivariate Correlation Analysis," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16)*, New York, 2016.

[9] N. A. Singh, K. J. Singh, and T. De, "Distributed denial of service attack detection using Naive Bayes Classifier through Info Gain Feature Selection," in *Proceedings of the International Conference on Informatics and Analytics (ICIA-16)*, New York, 2016.

[10] M. Khandelwal, D. K. Gupta, and P. Bhale, "DoS attack detection technique using back propagation neural network," in *Proceedings of International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Jaipur, pp. 1064-1068, 2016.

[11] L. Gao, Y. Li, L. Zhang, F. Lin, and M. Ma, "Research on Detection and Defense Mechanisms of DoS Attacks Based on BP Neural Network and Game Theory," *IEEE Access*, vol. 7, pp. 43018-43030, 2019.

[12] J. Brynielsson, and R. Sharma, "Detectability of low-rate HTTP server DoS attacks using spectral analysis," in *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Paris, pp. 954-961 2015.

[13] X. Wu, D. Tang, L. Tang, J. Man, S. Zhan, and Q. Liu, "A Low-Rate DoS Attack Detection Method Based on Hilbert Spectrum and Correlation," in *Proceedings of IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation*

(SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, pp. 1358-1363, 2018.

[14] R. Kumar, S. P. Lal, and A. Sharma, "Detecting Denial of Service Attacks in the Cloud," in *Proceedings of IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, Auckland, pp. 309-316, 2016.

[15] A. Ahmad, M. N. Kama, O. M. Yusop, N. A. A. Bakar, and N. B. Idris, "Cloud denial of service detection by dendritic cell mechanism," in *Proceedings of 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, Penang, pp. 179-184, 2018.

[16] S. S. Vernekar, and A. Buchade, "MapReduce based log file analysis for system threats and problem identification," in *Proceedings of the 3rd IEEE International Advance Computing Conference (IACC)*, Ghaziabad, pp. 831-835, 2013.

[17] J. Therdphapiyanak, and K. Piromsopa. "Applying Hadoop for log analysis toward distributed IDS," in *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication (ICUIMC '13)*, New York, 2013.

[18] M. A. Latib, S. A. Ismail, O. M. Yusop, P. Magalingam, and A. Azmi, "Analysing Log Files for Web Intrusion Investigation Using Hadoop," in *Proceedings of the 7th International Conference on Software and Information Engineering (ICSIE '18)*, New York, pp. 12-21, 2018.

[19] J. Zhang, P. Liu, J. He, and Y. Zhang, "A Hadoop Based Analysis and Detection Model for IP Spoofing Typed DDoS Attack," in *Proceedings of 2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, pp. 1976-1983, 2016.

[20] A. Alsirhani, S. Sampalli, and P. Bodorik, "DDoS Attack Detection System: Utilizing Classification Algorithms with Apache Spark," in *Proceedings of the 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, pp. 1-7, 2018.

[21] R. More, A. Unakal, V. Kulkarni, and R. H. Goudar, "Real time threat detection system in cloud using big data analytics," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, pp. 1262-1264, 2017.

[22] M. Idhammad, K. Afdel, and M. Belouch, "Distributed Intrusion Detection System for Cloud Environments based on Data Mining techniques," *Procedia Computer Science*, vol. 127, pp. 35-41, 2018.

[23] S. Alzahrani, and L. Hong, "Detection of Distributed Denial of Service (DDoS) Attacks Using Artificial Intelligence on Cloud," in *Proceedings of 2018 IEEE World Congress on Services (SERVICES)*, San Francisco, pp. 35-36, 2018.

[24] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425-441, Feb. 2017.

[25] P. Mell, and T. Grance, "The NIST definition of cloud computing," in *National Institute of Standards and Technology*, Gaithersburg, pp. 1–7, 2011.

[26] E. Morioka and M. S. Sharbaf, "Digital forensics research on cloud computing: An investigation of cloud forensics solutions," in *Proceedings of 2016 IEEE Symposium on Technologies for Homeland Security (HST)*, Waltham, pp. 1-6, 2016.

[27] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud security: Emerging threats and current solutions," *Computers & Electrical Engineering*, vol. 59, pp. 126-140, 2017.

[28] S. Basu et al., "Cloud computing security challenges & solutions-A survey," in *Proceedings of 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, pp. 347-356, 2018.

[29] A. Odebade, T. Welsh, S. Mthunzi and E. Benkhelifa, "Mitigating anti-forensics in the Cloud via resource-based privacy preserving activity attribution," in *Proceedings of 2017 Fourth International Conference on Software Defined Systems (SDS)*, Valencia, pp. 143-149. 2017.

[30] H. Arshad, A. B. Jantan, and O. I. Abiodun, "Digital Forensics: Review of Issues in Scientific Validation of Digital Evidence," *Journal of Information Processing Systems*, vol. 14, no. 2, pp. 346-376, 2018.

[31] K. K. R. Choo, C. Esposito and A. Castiglione, "Evidence and Forensics in the Cloud: Challenges and Future Research Directions," in *Proceedings of IEEE Cloud Computing*, vol. 4, no. 3, pp. 14-19, 2017.

[32] G. Sibiya, H. S. Venter and T. Fogwill, "Digital forensics in the Cloud: The state of the art," in *Proceedings of 2015 IST-Africa Conference*, Lilongwe, pp. 1-9, 2015.

[33] S. Zawoad and R. Hasan, "Trustworthy Digital Forensics in the Cloud," *Computer*, vol. 49, no. 3, pp. 78-81, Mar. 2016.

[34] S. Nanda and R. A. Hansen, "Forensics as a Service: Three-Tier Architecture for Cloud Based Forensic

Analysis," in *Proceedings of the 15th International Symposium on Parallel and Distributed Computing (ISPDC)*, Fuzhou, pp. 178-183, 2016.
[35] S. T. Zargar, J. Joshi and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," in *Proceedings of IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046-2069, 2013.
[36] M. H. Bhuyan, D.K. Bhattacharyya, and J.K. Kalita, "An empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection," *Pattern Recognition Letters*, vol. 51, pp. 1-7, 2015.
[37] Apache Hadoop, https://hadoop.apache.org/, 2018.
[38] Hadoop Distributed File System, https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html, 2018.
[39] Spark Streaming Programming Guide, https://spark.apache.org/docs/latest/streaming-programming-guide.html, 2018.
[40] Flume User Guide, https://flume.apache.org/FlumeUserGuide.html, 2018.
[41] Fortinet FortiDDoS: Protection Profile Settings, https://help.fortinet.com/fddos/4-3-0/FortiDDoS/Managing_thresholds.htm, 2019.
[42] B.B. Gupta, and O.P. Badve, "Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a Cloud computing environment," *Neural Computing and Applications*, vol. 28, Apr. 2016.
[43] K. Bhushana, and B. B. Gupta, "Hypothesis Test for Low-rate DDoS Attack Detection in Cloud Computing Environment," *Procedia Computer Science*, vol. 132, pp. 947-955, May 2018.

Authors

Janitza Punto Gutierrez received B.S. degree in Telecommunications Engineering in the Pontifical Catholic University of Peru in 2014. In August 2019, she graduated from Seoul National University of Science and Technology and received master's degree in computer science and Engineering. Her research interests include Cloud Computing, Big Data and Deep Learning.

Kilhung Lee is a Professor at the Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, Korea. He received the M.SC. in Networks in 1991 and Ph.D. degree in Networks from Yonsei University, Seoul, Korea. During 1991-1995, he was working at the Research Center of LG Information and Communication. His research interests are Ad-hoc Networks, Wireless Sensor Networks, High Speed Network, Network Management, Distributed Computing and Web Service.