J Inf Process Syst, Vol.17, No.4, pp.675~689, August 2021
https://doi.org/10.3745/JIPS.03.0164

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

High Rate Denial-of-Service Attack Detection System for Cloud Environment Using Flume and Spark

Janitza Punto Gutierrez* and Kilhung Lee*

Abstract
Nowadays, cloud computing is being adopted for more organizations. However, since cloud computing has a virtualized, volatile, scalable and multi-tenancy distributed nature, it is challenging task to perform attack detection in the cloud following conventional processes. This work proposes a solution which aims to collect web server logs by using Flume and filter them through Spark Streaming in order to only consider suspicious data or data related to denial-of-service attacks and reduce the data that will be stored in Hadoop Distributed File System for posterior analysis with the frequent pattern (FP)-Growth algorithm. With the proposed system, we can address some of the difficulties in security for cloud environment, facilitating the data collection, reducing detection time and consequently enabling an almost real-time attack detection.

Keywords
Denial-of-Service, FP-Growth Pre-filtering, HDFS Spark Streaming, Web Log

1. Introduction

The appearance of new technologies has facilitated the occurrence of criminal behavior in the network. Criminals have bigger chances of abusing new technologies with different purposes than the one they originally had. As a result, people have found and developed complex and innovative methods to perform traditional crimes using those technologies. Several of current systems are becoming vulnerable, not only as a target but also as a tool for criminal activities [1].

Additionally, the rising access to the Internet and massive use of social networking sites (i.e., Facebook) and in conjunction with mobile technology has completely transformed our daily life, due to the new opportunities and facilities created by them and the always present digital information and data [1]. Consequently, the amount of data flowing in the network is increasing. For this reason, the detection of any threat to the network security is becoming more difficult [2]. For example, a traditional intrusion detection system (IDS) working in cloud have low detection accuracy, high false positive rate and high running time [3]. The changing nature of the cloud makes it even more difficult to implement an efficient attack detection system.

Since cloud computing has become a top research topic in the information technology (IT) area and is viewed as a paradigm modification when providing computing resources through the network [4], various companies and organizations are anticipated to adapt their IT infrastructure and systems to the cloud.

Manuscript received March 27, 2019; first revision November 4, 2019; accepted April 14, 2020.
Corresponding Author: Kilhung Lee (khlee@seoultech.ac.kr)
* Dept of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul, Korea (jnpunto.92@gmail.com, khlee@seoultech.ac.kr)

This change can be beneficial in the economic aspect since it enables them to reorganize the spending on technology infrastructure and capital cost [5]. However, as shown in [4], there are some issues related to cloud computing reliability and high availability. Load balancing, energy efficiency, data/service availability and security are some of the critical problems presented by cloud business models. Mainly, security can be pointed as one of the causes of evading the adoption of cloud services. As a specific example, the security risk related to denial-of-service (DoS) attacks represents the main impediment of the rapid and complete adaption to the cloud [5].

DoS attacks can appear in many forms and may have different targets (e.g., application, web services) [5]. According to [6], in the last decade, many servers in a cloud environment were victim of a denial of service attack, where a big number of abused systems sent countless requests to one or more web servers. This traffic is heavy and is originated in bursts, increasing in a short time. Even though there exist numerous advanced approaches to detect DoS attacks in cloud, these proposals still lack a good detection accuracy since they do not contemplate the characteristics of cloud environment [5]. In this paper, a DoS attack detection system for cloud environment which uses Hadoop was proposed. In the following content, the related work was reviewed in Section 2. Section 3 contains cloud computing concepts and security challenges, while Section 4 talks about DoS attacks. Section 5 explains about Hadoop environment and some of its components. After that, Section 6 describes in detail the proposed solution. Finally, Section 7 shows results of the performed tests.

2. Related Work

Recent research is focused on attack detection in cloud environment (Table 1). Among them, we can mention the work [7], which designed an immune-based intrusion detection system that were deployed in virtual machines (VM) of cloud computing. The purpose of this proposal was keeping the user-level applications safe for them in client VM. The proposed solution took sequences of commands that compose programs, conceptualizes them as antigens (similarly to the biological immune system), synthesize and use environmental information of client VM to form danger signals, and finally implements an intrusion detection system. As a part of the detection phase, the solution includes a mechanism that monitors information in order to control the intrusion detection program, by doing this it is possible to guarantees the authenticity of the test data. After performing experiments, the results indicated that the solution does not use many resources of the virtual machine and has good detection performance.

Another direction some studies are following is the DoS attack detection in order to protect the systems. The proposed model in [6] aims to detect distributed denial-of-service (DDoS) attacks and discriminate them from the flash events, by looking for any anomaly in the traffic pattern. They emphasized that, if packets in traffic are analyzed individually, they seem to be non-malicious. Though, in case packets at different locations are associated, it is possible to find an anomaly. Because DDoS attacks are created by humans, the study of packets can show some patterns, such as change of the common composition and order of packets in non-malicious environments. Based on this, in [6] they proposed a model that finds the initial number of source IP addresses from which the traffic comes (Source Entropy) and then finds the grade of disorder in the traffic coming from the same network (Traffic Entropy). Researchers did simulations by sending packets that could be from an attacker or a normal user and detecting the moment when the traffic starts increasing. At that moment the model starts to calculate the current source entropy and traffic entropy, so it is possible to know the deviation compared to the initial values and it is possible to determine if there is a DDoS attack or it is just a flash event. In order to measure the efficiency of the solution, the detection rate, classification rate and false positive rate were utilized.

Additionally, if we talk about solutions for DoS attack detection in cloud environment, in [5] they considered DoS attacks targeting VM in a cloud, defined as the event where one or more VMs consume all the physical resources, provoking that the hypervisor cannot maintain more VMs. To address those attacks, they proposed a framework that offers a flexible detection by using the support vector machine (SVM) learning technique. This technique employs a nonlinear mapping to transform the original data into data with many dimensions, with the purpose of creating a hyperplane that efficiently splits the training tuples according to their classes. Following this concept, the proposed framework is composed of a hypervisor gathering features for the training phase of the SVM classifier so it can differentiate the ordinary activity from a DoS attack. Moreover, the proposed model enables VMs to frequently indicate their current metrics to the hypervisor, who can find a relation between those values and the actual resources load and discriminate a normal high load from a DoS attack. Also, this mechanism permits that the hypervisor recognizes the affected VMs trying to consume more resources. This is possible because the hypervisor can estimate the resources load in the affected VMs according to the announced metrics. That estimation can be compared with the real-time resources load and, if the estimation is higher than a certain value or it is out of a certain range of values, there is a higher chance that the analyzed VM has been affected.

Table 1. Reviewed literature

Study	Main remarks and contributions
Study of danger-theory-based intrusion detection technology in virtual machines of cloud computing environment [7]	– Attack detection solution for cloud environment with good detection performance – Applied immune-based intrusion detection system that were deployed in virtual machines of cloud computing – Included mechanism to monitor information for control of the intrusion detection program
Detection of spoofed and non-spoofed DDoS attacks and discriminating them from flash crowds [6]	– DoS attack detection system by discriminating them from non-malicious flash events, based on anomaly detection in traffic pattern – Initial values of source and traffic entropy were calculated and compared to the real time values in order to find deviation – Detection rate, classification rate and false positive rate were used as efficiency metrics
An SVM-based framework for detecting DoS attacks in virtualized clouds under changing environment [5]	– Solution for DoS attack detection with virtual machines as targets in cloud environment – Applied the SVM classifier by using features gathered by the hypervisor to distinguish normal activity and DoS attack – Resources metrics from virtual machines are known by the hypervisor, making it possible to calculate estimated used resources at any moment and compare those values to the real time used resources in order to identify affected virtual machines
Distributed intrusion detection system for cloud environments based on data mining techniques [3]	– Hadoop-based security solution for systems adapting cloud computing – Implemented an intrusion detection system by collecting traffic data with Hadoop and MapReduce, employing Naïve Bayes model to identify normal and abnormal traffic, and using classifiers based on Random Forest to identify the type of attack
A state-of-the-art survey of malware detection approaches using data mining techniques [8]	– Survey of malware detection systems based on data mining techniques – Comparison of systems based on classification approaches, data analysis methods and accuracy
Network intrusion detection in big dataset using Spark [2]	– Classification-based intrusion detection scheme – Applied feature reduction algorithms and supervised data mining techniques such as Naïve Bayes and Random Forest – Employed Spark to decrease processing time
A multimodal fusion based framework to reinforce IDS for securing big data environment using Spark [9]	– Malicious content detection model – Applied a multi-modal fusion of different classifiers, increasing accuracy and detection rate and decreasing false alarm rate – Used MapReduce and Spark to reduce processing time

In recent literature, we can find too some works aiming to offer security solutions with the use of Hadoop environment. Among the researches using Hadoop, we can mention [3] which emphasize the security issues that appear in systems transitioning to cloud computing, so an IDS using distributed machine learning techniques was proposed. The system intercepts incoming network traffic and stores it for formatting, cleaning and normalizing with Hadoop and MapReduce. The result is classified into normal or abnormal traffic by employing the naïve Bayes model. Finally, the type of attack is determined with the help of an ensemble of learning classifiers based on Random Forest, permitting to take any mitigation action. In fact, in recent researches the utilization of malware detection systems based on data mining techniques have augmented, even machine learning is employed to identify malicious files. In [8], an organized and meticulous survey of malware detection systems based on data mining techniques was shown. Those systems were compared considering factors including classification approaches, data analysis methods and accuracy.

The study [2] is other work implemented with Hadoop environment. In this case, Spark was used for implementation in order to speed up the processing. This work proposed a framework that uses feature reduction algorithms in order to decrease the less important features of the data and apply supervised data mining techniques, such as naïve Bayes and Random Forest algorithms, to create a classification-based intrusion detection scheme. Similarly, the authors [9] proposed a model to detect any malicious content by using a multi-modal fusion that combines the results of different classifiers, which permits to increase accuracy and detection rate while decreasing false alarm rate. In this work, Hadoop, MapReduce, and Spark were included to guarantee fast processing in a parallel computational environment and reduce processing time.

3. Cloud Computing

According to the National Institute of Standards and Technology (NIST), cloud computing can be defined as a new computing model that enables network access in an ubiquitous, on-demand as pay-per-use service, accessing to a shared and location-independent resource pool of configurable computing resources such as servers, applications, storage and services, which can be fast provisioned and released with almost no management effort or service provider interaction with rapid elasticity [10–12]. Currently there are three main models to supply cloud services [11,12]: software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS). However, in the process of offering these services, matters of information and privacy security are arising, becoming important reasons that stop the adaption of cloud computing [7]. Since cloud computing has a virtualized, volatile, scalable, multi-tenancy and multi-jurisdictional distributed nature, it is challenging task to perform attack detection in the cloud following conventional processes, while keeping the reliability, confidentiality and integrity of data [13,14]. Furthermore, the collection and analysis of data stored in the cloud environment are also difficult processes. In consequence, these complications highlight the need of new methods, since the existing traditional ones are not applicable to the cloud environment. The NIST published a list of 65 challenges associated with the cloud computing security. According to this list, the most common problems are associated to the distributed nature of the resources in the cloud and the enormous number of users accessing to them. Moreover, stopping a real device to get some data for attack detection is not possible and copying any data of that machine may disturb the privacy of other users. The previous challenges can be resumed in the following list and are shown in Fig. 1:

- Physical access: In cloud environment it is not possible to power off devices and connect directly to them in order to analyze or get digital evidence [13,15].
- Live forensics: Evidence may be lost if the system is powered down, so it is important to find a way to perform investigation while the system is still running [11].
- Evidence collection, considered a challenge because there are diverse digital media types and diverse log formats that are not standardized [15,16], the evidences can be at any data center so finding location is time-consuming [11], and the integrity of log files is hard to validate [17].
- Volume of data, this increases the chances of losing important evidence data and the required time to perform investigation and tool capacity is limited too [13].

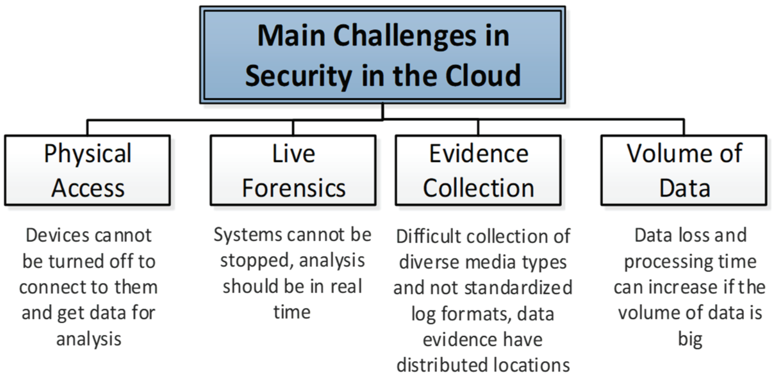
Main Challenges in Security in the Cloud
Physical Access
Live Forensics
Evidence Collection
Volume of Data
Devices cannot be turned off to connect to them and get data for analysis
Systems cannot be stopped, analysis should be in real time
Difficult collection of diverse media types and not standardized log formats, data evidence have distributed locations
Data loss and processing time can increase if the volume of data is big

Fig. 1. Main challenges in security in the cloud.

4. Denial-of-Service Attack

DoS attacks aim to prevent access of real non-malicious users to a specific computing resource [18]. Recently, this attack has escalated to a distributed scheme where the attacker abuse a group of hosts, becoming a big security threat. Generally, DoS attacks consume most of the resources of the target and the network bandwidth [19]. The classification of DoS attacks can be based on the protocol level at which the attack is directed. We can mention the following [18,20,21]:

1) Network/transport-level DDoS flooding attacks: In these attacks, network/transport protocols are employed as tool to flood the victim. Usually the attacker sends packets using TCP, UDP, ICMP and DNS protocols.
2) Application-level DDoS flooding attacks: These attack typically consume less bandwidth and, because they are similar to the non-malicious traffic, they are difficult to recognize. Also, these attacks exploit explicit characteristics of application protocols such as HTTP, DNS, and SIP.

Finally, according to the type of malicious traffic related to a DDoS attack, we can describe the next types [19]:

1) High-rate DDoS attack traffic: This attack involves a big traffic sent to the target, which is similar to a flash crowd or the scenario when a big amount of legitimate users access to a system.

2) Low-rate DDoS attack traffic: This attack does not include big traffic so it looks like a common group of legitimate users accessing to a system. Because of this, it is difficult to detect and mitigate on time.

When these DoS attacks are directed to cloud environment, the access to services and resources are impossible for the user [20]. In case of IaaS model, users can easily ask for more storage or computing power through web interfaces but they will not be able to receive the requested resources in case of a DoS attack [22]. Similarly, in case of PaaS, the offered operating systems, runtime systems and web servers would become inaccessible, while users would report errors in the software provided by the SaaS service model. The variety of the explained delivery models increases the vulnerability of cloud computing to DoS attacks, compared to other platforms [23]. For this research, we focus on high rate DoS attacks exploiting HTTP protocol to target web servers in the cloud. The application layer DoS type of attacks are problematic and difficult to mitigate when flooding the target system with requests that look innocuous. They can destabilize the cloud services by sending HTTP flood packets to distress a web server in the cloud [20].

5. Hadoop Environment

The Apache Hadoop is a project that develops open-source software and aims to offer reliable, scalable and distribute computing of big data sets in clusters of machines by employing simple programming models. Each of those machines offer computation and storage and, even if they can have failures, Hadoop library ensures high availability by detecting and handling failures at the application layer [24].

Among the modules included in the project, Hadoop Distributed File System (HDFS) can be mentioned. It is mainly known as a fault-tolerant distributed file system that enables access with high throughput while running on commodity hardware and managing large data sets. HDFS let applications to run near the data location, instead of moving huge data set to the computation, minimizing network congestion [25]. When a file is stored in HDFS, it is divided in one or many blocks, which are replicated and stored in DataNodes for fault tolerance. The mapping of blocks and their locations is controlled by the NameNode [25]. The components of HDFS are shown in Fig. 2.

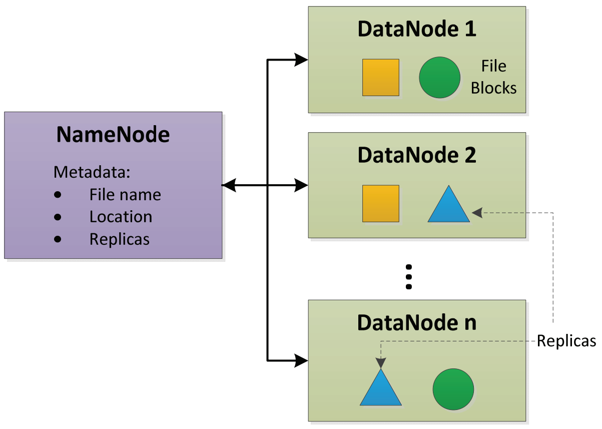

Fig. 2. Hadoop Distributed File System (HDFS) architecture.

Hadoop also includes a cluster computing engine called Spark, which offers a programming model able to create a variety of applications including machine learning, stream processing, and graph computation, by using APIs in Java, Scala, Python, and R. Specifically about Spark Streaming tool, it permits to process live data streams that can come from sources such as Kafka or Flume, and store the results in file systems or databases [26]. As mentioned, Flume can be integrated to Hadoop and permits to collect, aggregate and move big size of data from many diverse sources. Mainly, Flume is used with log data, however it can be also utilized to get network traffic data, social media data and email messages. With Flume, the data goes from Source to Channel and then to Sink by running a process called Flume agent, which allows to move the data from an external source to an external destination [27].

6. Proposed Solution

The proposed solution consists of a filtering scheme for web server logs that takes data related to a HTTP flooding DoS attack, specifically request flooding attack, and stores it in HDFS to be analyzed with an algorithm for DoS attack detection. In this research we used the FP-Growth algorithm with the purpose of confirming the existence of an attack and determining other characteristics of the attack, such as origin IP address or target's IP address. The FP-Growth algorithm is a method that mines frequent patterns ("FP"). The algorithm starts with frequency calculation of items included in a group of transactions. Those values are compared, and the frequent items are obtained. Then the algorithm employs a suffix tree, called FP-tree, in order to encode the transactions but without creating a list of candidates and without increasing the consumed processing power. Finally, the algorithm gets the frequent groups of items (frequent patterns) from the FP-tree based on the threshold for minimum support and calculating the support of each itemset composed of the frequent items [28].

The process performed by the FP-Growth algorithm can be considered as an important phase when looking for associations, correlations, partial periodicity and emerging patterns [29]. The version of FP-Growth algorithm that is implemented in the MLlib library included in Spark was used.

The proposed solution is shown in Fig. 3 and was composed of:

a) Flume: Collects web server access logs.
b) Spark: Possess two main functions:
   - ✓ Using Spark Streaming, receives and filters data from Flume to find any suspicious request and stores them in HDFS.
   - ✓ Apply the FP-Growth algorithm to analyze suspicious data stored in HDFS.
c) HDFS: Stores logs previously generated in normal state and suspicious logs found in real-time analysis.

Before the implementation of the system, it is necessary to collect web server access logs in a normal state (without attacks). The purpose of this preliminary step is to have a baseline that can be compared with the real time collected logs and permit the detection of any suspicious log. This collection should be done during an entire year since the number of requests to the web server can change according to the hour of the day or season of the year. During this preliminary step, the number of requests done to the server per second are calculated and the values are stored in HDFS with the format "[Time hh:mm:ss]|[number of requests]" and the name of the file will indicate the corresponding date. All this processing can be done offline by Spark.

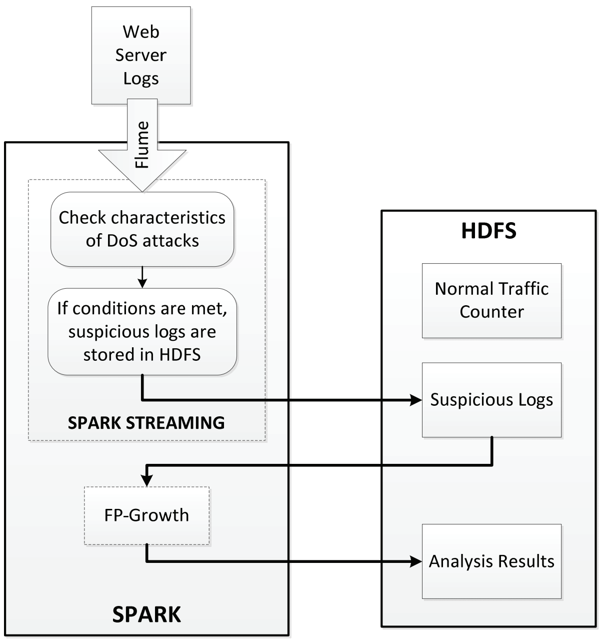

Fig. 3. Proposed solution scheme.

In the first step of the solution, Flume is used for web server logs collection. It is required to modify the configuration file and indicate the type of source. Among the types of sources included in Flume, the source TailDir is the most suitable to keep the logs secure even if Flume stops. Also, in the configuration file the sink is specified as an Avro Sink, where the log events are transformed into Avro events and sent to a specified hostname and port. This is to facilitate the integration with Spark Streaming.

In the next step, the logs coming from Flume are analyzed with Spark Streaming to filter only the suspicious ones and store them in HDFS. To be considered as a possible DoS attack, there are three conditions that should be checked. The implementation of this filter was done by using python and including Spark Streaming package in the program. The main steps of this filter are:

a) Read the file containing information about the number of requests per second in a normal state, which corresponds to the same date of the real time analysis. For example, if the logs are being collected and analyzed on November 17th 2018, the normal number of requests should be read from the file which name includes November and 17. The year does not matter since the year of the retrieved file corresponds to the time when that information was collected.
b) Identify the fields included in the web server access log by using a Regular Expression (RegEx).
c) Calculate the number of requests per second and compare it to the normal number of requests that was read in step a). If the number of requests per second coming in real time is higher than the number in a normal state in at least a specified threshold t_{time}, then the log is suspicious and continue to step f). Otherwise, continue to the next step.
d) Calculate the number of requests per second that has a server response with the same size. If this number is higher than a specified threshold t_{size}, the log is considered as suspicious and continue to step f). Otherwise, continue to the next step.
e) Calculate the number of requests per second containing an empty agent. If this number is higher than a specified threshold t_{agent}, the log is considered as suspicious and continue to the next final

step. Otherwise, the logs are not stored in HDFS.

f) The suspicious logs are stored in HDFS in a directory which name includes the complete date and time of the corresponding logs, but each log is saved with a different format according to the filter condition that matched:
   - ✓ If the condition was about the number of packets per second, the format is "[Time]||[Number of requests per second]||[Number of requests in a normal state]||[Complete web server log]".
   - ✓ If the condition was about the number of server responses with same size, the format is "[Time]||[Size of server response]||[Number of requests with server response of same size]||[Complete web server log]".
   - ✓ If the condition was about the agent, the format is "[Time]||[Agent]||[Number of requests with empty agent]||[Complete web server log]".

The conditions of the filter are similar to the ones uses in [30]. About the thresholds used in the filter conditions, the recommendations in [31] were followed since those specifications are used in products that aim to detect DDoS attacks. First, the maximum number of requests per second was calculated by analyzing logs in normal state. After that, according to [31], that value should be multiplied by a number between 1 and 5 to set the maximum number of accepted requests. However, in the proposed solution the threshold is defined as the difference between the number of requests in real time and the number of requests in normal state. Based on this, the threshold t_{time} is defined as 1.5 × (max_req) – (max_req) = 0.5 × (max_req). Additionally, the thresholds t_{size} and t_{agent} are defined as 1.5 × (max_req).

As a third step of the process, the web server access logs that were found as suspicious of being part of a DoS attack are retrieved from HDFS and analyzed by Spark. This last step was also implemented by using python and employed the FP-Growth algorithm already included in Spark library called mllib. The process of this program starts with reading only the directories including logs from the last 30 minutes, which is possible to identify since the directory name includes the date and time when the logs were generated. Again, a RegEx is used to identify the fields of the web server access log. Moreover, the FP-Growth algorithm analyzes transactions including group of items, so the log fields are considered as the items and each log is a transaction. The considered fields were the IP address from where the request came, user id, HTTP method, requested resource, response code and server response size. Based on this, the read logs are analyzed in periods of 5 minutes to find the most frequent item sets composed of the mentioned log fields.

However, as explained before, the FP-Growth algorithm can find frequent item sets that are a subset of the original transaction. For example, if the original transaction is [147.115.0.1, root, GET, /index.php, 200, 154], the algorithm can give as a result the subsets [147.115.0.1, /index.php, 200, 154], [147.115.0.1, root, GET] or even subsets with one field [147.115.0.1] with their corresponding frequencies. For this reason, the Python file that implements this final step considers the item sets that have more than four 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 only the first third part of them are taken as result and are stored in HDFS under directories with names including the date and time. By doing this, characteristics of the attack, such as attacker IP address and used protocol, can be identified. It is important to mention that the mentioned period of 30 and 5 minutes can be modified by the cloud administrator. Still, if those periods get longer, the results of the analysis can lose accuracy.

7. Test and Results

The tests performed were done in two main scenarios: (1) using the proposed solution (with filter) and (2) analyzing all the logs (without filter). The process of the second scenario is presented in Fig. 4. This process contains same components as the proposed solution but with the main difference being the steps performed with Spark Streaming after the log collection. In this case, there is no filtering and only the log format is changed to log fields separated by a specified character (e.g., a vertical bar "|") in order to facilitate the use of the FP-Growth algorithm which has lists of items as input. Consequently, without any filtering, the complete logs are stored in HDFS and analyzed with the FP-Growth algorithm.

Tests of both scenarios were performed with two sets of access log, one with normal log without DoS attack (normal log) found in SecRepo.com (Samples of Security Related Data) and other with DoS attacks (attack log) found in log-sharing.dreamhosters.com (Public Security Log Sharing Site). All the testing was performed in a Docker container with Hortonworks Data Platform (HDP) 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 volume of monitoring data is big and requires long time to processing it. The main purpose of this paper is reducing the processing load, required memory usage by adopting a pre-filter mechanism without loss of detection performance. And if possible, we find a mechanism for real time analysis and detection capability. So, we selected the metrics as memory usage, CPU usage, and processing time, which were collected from Apache Ambari that is included in the HDP Sandbox. For comparative purposes, the process of the proposed solution was divided in two main phases that were run sequentially:

1) Storage Phase: Includes log collection, filtering and suspicious logs storage
2) FP-Growth Phase: Includes FP-Growth algorithm analysis and results storage

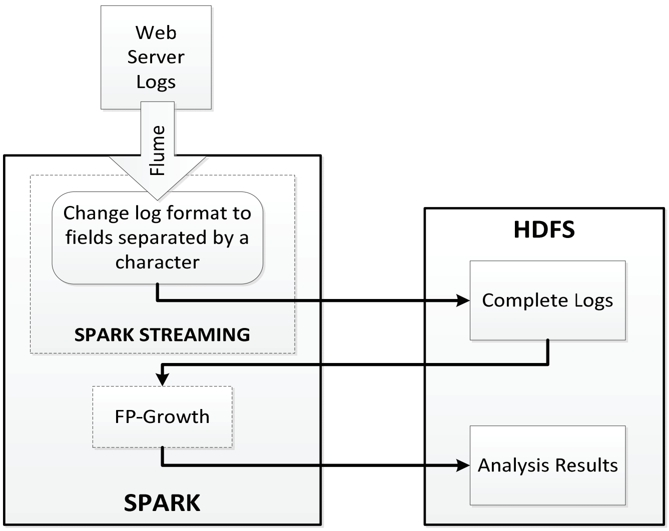

Fig. 4. Test scenario without filtering.

As shown in Table 2, in case of the tests with normal log, we found that the proposed system did not find any suspicious log so the analysis with FP-Growth algorithm was not performed. This resulted in a total time of 30 seconds to collect the logs and apply the filter, using 15 GB of memory and 78% of CPU. On the other hand, if we analyzed the logs without a previous filtering, the additional memory of 13.7 GB was used. Also the CPU usage was higher with an additional 41%. However, the time was much higher because the FP-Growth was analysis took 276 seconds.

Table 2. 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 (s)	
		Storage	FP-Growth	Storage	FP-Growth	Storage	FP-Growth
Normal log	With filter	15.0	-	78	-	30	-
	Without filter	14.8	13.7	62	41	10	276
Attack log	With Filter	16.2	15.2	75	53	30	78
	Without Filter	16.7	15.1	54	45	30	210

In case of the tests with attack log, the filtering got some suspicious logs which were stored in HDFS and analyzed with the FP-Growth algorithm, so the total time was 108 seconds, with memory usage of 16.2 GB and CPU usage of 75% when storing the suspicious data and memory usage of 15.2 GB and CPU usage of 53% when analyzing it. However, when no previous filtering was done, the total time was 240 seconds. The memory usage was similar to the case with filtering: 16.7 GB when storing data and 15.1 GB when analyzing it; while CPU usage was 54% when storing and 45% when analyzing with the FP-Growth algorithm.

As additional tests, the process was run with both phases simultaneously performed and logs generated in real time with DoS attack and without any attack. This means the spark streaming is constantly reading the access logs and filtering them, while the python file that applies the FP-Growth algorithm is constantly checking if suspicious logs were stored in HDFS. The used environment was the same but a web server was implemented by using Xampp for Linux version 7.3.0. In order to get the information about the browsing agent, the log format configuration was set to "combined."

In case of normal logs (without any attack), we used the curl command in the form of "for i in 'seq 1 20'; do curl http://IP_ADDRESS:PORT; done" to send requests to the web server. On the other hand, in case of attack logs (with DoS attack), we used a DoS tool called HULK, which can attack a web server by creating large amount of traffic.

The results of these tests are shown in Table 3. In this case, the metrics also include the memory usage and the CPU usage; however, it was necessary to include the time to get FP-Growth algorithm results since that analysis confirms if there is any DoS attack and gives information about it, so it can be used as a metric to compare the performance of the proposed solution.

Table 3. 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 (s)
Normal log	With filter	16.3	76.5	-
	Without filter	17.1	68.2	-
Attack log	With Filter	18.3	76.5	60
	Without Filter	17.7	70.4	120

With the normal logs, the filtering process did not find any suspicious log so the FP-Growth algorithm was not used, resulting in a memory usage of 16.3 GB and CPU usage of 76.5%. Without filtering the logs, all were stored in HDFS but no attack was found after applying the FP-Growth algorithm so it was not possible to determine the time that the analysis took. In this test, the memory usage was 17.1 GB and the CPU usage was 68.2%, similarly to the test with filtering.

With the attack logs, the filtering process found suspicious logs and the FP-Growth algorithm results were gotten after 60 seconds. In this test, the memory usage was 18.3 GB and the CPU usage was 76.5%. Without any filtering, the resources used were similar too, but higher than the test of normal logs without filtering. The memory usage was 17.7 GB and the CPU usage was 70.4%. However, the FP-Growth analysis results were gotten after 120 seconds. This shows that it is possible to know the existence of a DoS attack in a shorter period when applying a simple filter to the logs.

Finally, in order to know the change of the processing time according to the log size or the number of analyzed requests, additional tests with different size logs were performed. In case of the normal log, the sizes for small, medium, and large logs were 806 requests in 16 minutes, 1,817 requests in 1 hour, and 2,862 requests in 1 hours, respectively. In case of the attack log, the sizes for small, medium, and large logs were 5,696 requests in 4 minutes, 9,297 requests in 12 minutes, and 15,062 requests in 17 minutes, respectively. Both sets of logs were taken from the original sources mentioned at the beginning of Section 7. Also, the tests were done with and without filter by running the storage and the FP-Growth phase sequentially, so the considered processing time is the addition of each phase processing time.

Fig. 5 shows the results when analyzing a normal log. We can clearly notice the advantage of using a filter, since the difference between the processing times of with-filter case and without-filter case is big even with small logs and it keeps increasing when the log becomes larger. Even in the scenario of a DoS attack, we can have a smaller processing time when applying a filter to the logs, as shown in Fig. 6. Also with the attack log, the difference between processing times of the with-filter case and without-filter case has a tendency to increase when log size becomes bigger.

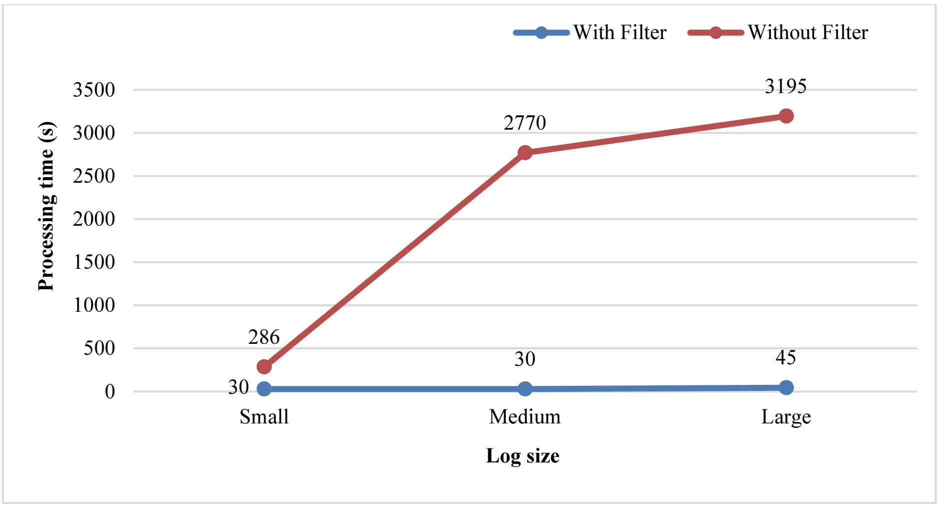

Fig. 5. Log size vs. processing time for a normal log.

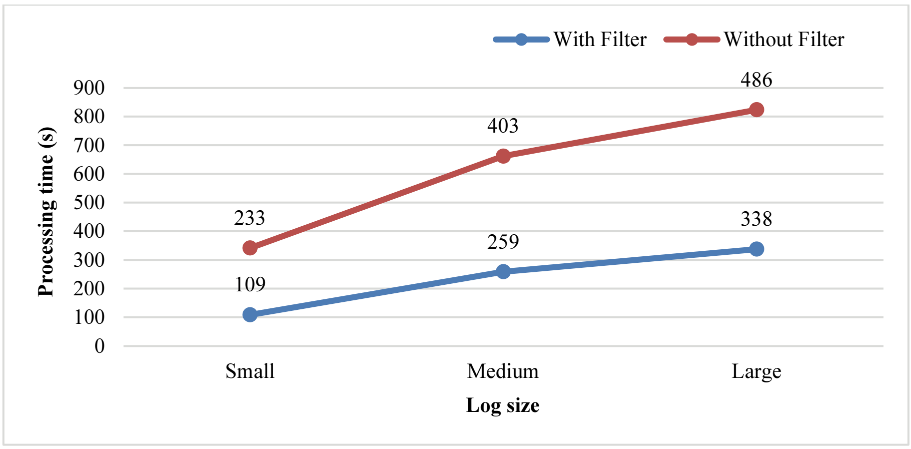

Fig. 6. Log size vs. processing time for an attack log.

8. Conclusion

With the proposed system, we can address some of the difficulties in security for cloud environment. The utilization of Flume facilitates the data collection since it works with different types of data and have mechanisms to ensure the data is not lost during the collection. Moreover, the use of Spark Streaming to implement a data filter by checking simple conditions (e.g., number of requests) permits to reduce the time of DoS attack detection. This is because the FP-Growth algorithm is applied only to suspicious data instead of all web server logs. Additionally, the reduction of processing time enables an almost real-time attack detection. Finally, the use of the Hadoop environment is suitable for processing a big volume of data, which is typical characteristic in cloud environment.

Still, we consider as a future work the establishment of a model to estimate appropriate values of the thresholds used in the conditions of the filter. Another important consideration is the definition of a period to update the conditions used in the filter. This is because new attacks can appear each day and the characteristics can vary, so the filter should be adapted to detect them properly. Additionally, it is important to perform test with variations in the environment, such as using more nodes in the Hadoop environment or using other algorithms different from the FP-Growth algorithm, in order to compare the performance of the proposed solution. Finally, we aim to extend the solution to be possible to analyze traffic packets and identify also Slow Rate DoS attacks, such as Slow Message Body, Slow Post and Slow Header DoS attacks.

Acknowledgement

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

References

[1] 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.

[2] P. Dahiya and D. K. Srivastava, "Network intrusion detection in big dataset using spark," *Procedia Computer Science*, vol. 132, pp. 253-262, 2018.

[3] 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.

[4] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: a reference roadmap," *Human-centric Computing and Information Sciences*, vol. 8, article no. 20, 2018. https://doi.org/10.1186/s13673-018-0143-8

[5] A. Abusitta, M. Bellaiche, and M. Dagenais, "An SVM-based framework for detecting DoS attacks in virtualized clouds under changing environment," *Journal of Cloud Computing*, vol. 7, article no. 9, 2018.

[6] J. Gera and B. P. Battula, "Detection of spoofed and non-spoofed DDoS attacks and discriminating them from flash crowds," *EURASIP Journal on Information Security*, vol. 2018, article no. 9, 2018. https://doi.org/10.1186/s13635-018-0079-6

[7] R. Zhang and X. Xiao, "Study of danger-theory-based intrusion detection technology in virtual machines of cloud computing environment," *Journal of Information Processing Systems*, vol. 14, no. 1, pp. 239-251, 2018.

[8] A. Souri and R. Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques," *Human-centric Computing and Information Sciences*, vol. 8, article no. 3, 2018. https://doi.org/10.1186/s13673-018-0125-x

[9] G. Donkal and G. K, Verma, "multimodal fusion based framework to reinforce IDS for securing big data environment using spark," *Journal of Information Security and Applications*, vol. 43, 1-11, 2018.

[10] K. K. R. Choo, C. Esposito, and A. Castiglione, "Evidence and forensics in the cloud: challenges and future research directions," *IEEE Cloud Computing*, vol. 4, no. 3, pp. 14-19, 2017.

[11] S. A. Hussain, M. Fatima, A. Saeed, I. Raza, and R. K. Shahzad, "Multilevel classification of security concerns in cloud computing," *Applied Computing and Informatics*, vol. 13, no. 1, pp. 57-65, 2017.

[12] 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, MA, 2016, pp. 1-6.

[13] S. Nanda and R. A. Hansen, "Forensics as a service: three-tier architecture for cloud based forensic analysis," in *Proceedings of 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC)*, Fuzhou, China, 2016, pp. 178-183.

[14] J. H. Park, S. H. Na, J. Y. Park, E. N. Huh, C. W. Lee, and H. C. Kim, "A study on cloud forensics and challenges in SaaS application environment," in *Proceedings of 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Sydney, Australia, 2016, pp. 734-740.

[15] 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, Malawi, 2015, pp. 1-9.

[16] S. Zawoad and R. Hasan, "Trustworthy digital forensics in the cloud," *Computer*, vol. 49, no. 3, pp. 78-81, 2016.

[17] 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, Spain, 2017, pp. 143-149.

[18] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046-2069, 2013.
[19] 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.
[20] O. Osanaiye, K. K. 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.
[21] M. T. Manavi, “Defense mechanisms against distributed denial of service attacks: a survey,” *Computers & Electrical Engineering*, vol. 72, pp. 26-38, 2018.
[22] T. C. Vance, N. Merati, C. Yang, and M. Yuan, “Cloud computing for ocean and atmospheric science,” in *Proceedings of 2016 MTS/IEEE Conference in Monterey (OCEAN)*, Monterey, CA, 2016, pp. 1-4.
[23] M. A. Khan, “A survey of security issues for cloud computing,” *Journal of Network and Computer Applications*, vol. 71, pp. 11-29, 2016.
[24] Apache Software Foundation, “Flume User Guide,” 2021 [Online]. Available: https://flume.apache.org/FlumeUserGuide.html.
[25] Apache Software Foundation, “Apache Hadoop Project,” 2021 [Online]. Available: https://hadoop.apache.org/.
[26] Apache Software Foundation, “HDFS Architecture Guide,” 2018 [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.
[27] Apache Software Foundation, “Spark Streaming Programming Guide,” 2021 [Online]. Available: https://spark.apache.org/docs/latest/streaming-programming-guide.html.
[28] Apache Software Foundation, “Frequent Pattern Mining: RDD-based API,” 2018 [Online]. Available: https://spark.apache.org/docs/2.3.0/mllib-frequent-pattern-mining.html.
[29] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” *ACM SIGMOD Record*, vol. 29, no. 2, pp. 1-12, 2000.
[30] K. Sornalakshmi, “Detection of DoS attack and zero day threat with SIEM,” in *Proceedings of 2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 2017, pp. 1-7.
[31] Fortinet, “FortiDDoS: Protection Profile Settings,” 2019 [Online]. Available: https://help.fortinet.com/fddos/4-3-0/FortiDDoS/Managing_thresholds.htm.

Janitza Punto Gutierrez **https://orcid.org/0000-0003-3518-3858**

She 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. She is a software engineer and her research interests include cloud computing, big data and deep learning.

Kilhung Lee **https://orcid.org/0000-0001-5406-9888**

He 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 and wireless sensor networks, high speed network and management, distributed computing and web service.