Internet of Things (IoT) and Engineering Applications (2021) 6: 1-9 **DOI: 10.23977/iotea.2021.060101**
Clausius Scientific Press, Canada **ISSN 2371-8617**

Structured and Unstructured Log Analysis as a Methods to Detect DDoS Attacks in SDN networks

Nazar Peleh[1,a], Stanislav Zhuravel[1,b], Olha Shpur[1,c*] and Olha Rybytska[2, d]

[1] Department of Telecommunication, Lviv Polytechnic National University S. Bandery str., 12, Lviv, Ukraine
[2]Department of Mathematics, Lviv Polytechnic National University S. Bandery str., 12, Lviv, Ukraine
a. van_plus_k@ukr.net, b. zhuravelstas@gmail.com, c. olha.m.shpur@lpnu.ua,
d. olha.m.rybytska@lpnu.ua
**corresponding author – Olha Shpur*

Keywords: data analysis, log analysis, structured data, unstructured data, DDoS attacks, SDN.

Abstract: In this paper, we proposed a method for detecting DDoS attacks in SDN networks. Since the SDN controller contains information about the network and can create rules for its proper functioning, we propose to configure the SDN controller to detect a possible DDoS attack by examining the session information based on information from logs and flow tables. The information from the logs will be transmitted to the Log Analysis Subsystem, where two independent analysis processes will be started. To achieve this goal, we divide session information into normal and abnormal using the entropy method. If traffic deviations are detected, which will indicate a DDoS attack, the Log Analysis Subsystem will transmit the information to the SDN controller, which will create a rule to block the harmful connection. To identify these connections, we suggest using the Kulbak-Labler approach to detect anomalies during the session so that the SDN controller can block the IP addresses suspected of a harmful connection.

1. Introduction

A threat of unauthorized access influences demands on network infrastructure. Current security management in networks strongly relays on rules management and prioritization of the traffic flows, those flows in modern environments do not satisfy newly emerged requirements. A solution to new demands could be a system that analyses potential problems and perform safeguards actions or could notify system administrators about the emerging problems. That kind of a system would have a small amount of time to track malicious changes to the system state. Those kinds of tasks could be solved by implementing log analysis algorithms in core network infrastructure, this approach not only enables as to track the systems changes but also predict them.

One way of tracking and preventing malicious changes to the system state could be control over session time user spent in the system and analyzing the logs of the system produced over last time span. Such kind of analysis should be done at the core network level where it is possible to log all incoming packets before them will be redirected to the Black Hole by the network firewall. There are a wide range of investigations have been already in field of web services. Investigations of Hu Q., Lin, D and Tang B., [1] suggests giving each log entry a weight factor. This weight factor is determined based on number of unique requests produced by the user during the determined period of time. Events would be classified into normal (if the weight is average) or abnormal (if a value have a deviation).

Max Landauer et al. [2] detecting abnormal activities by grouping logs into clusters based on their similarity. The cluster created from logs in particular time span then the overlap metric is calculated which describes a probability of transition from one cluster to another.

Having machine learning can solve the monitoring problems and protect web services, particularity in the scope of software-controlled data centers. The authors of [3] taking focus on deep machine learning, they work on the Restricted Boltzmann Machine (RBM) method as a result they created a framework with SDN management and IDS structure. The research has been conducted on the basis of Tensorflow and KDD99 as an input set. The suggested algorithms presented 94% accuracy. Alternative variation of IDS is offered in [4]. Investigators simulate protected network by presenting identifiers of attacks types and parallel neural cross-training context.

Realization of IDS on machine learning and software control was introduced in [5] and [6], the authors [6] suggested substitute approach for the operation of intelligent transport SDN networks.

Wang and Lin suggested an instrument for detection and safeguarding from DDoS attacks in [1]. They proposed the method that splits Opneflow and sFlow controls for abnormality discovery. As a result, deployment of such protection is a rather difficult task. A more accurate method of detecting attacks is proposed in [7].

Yang et al. offered a method which is based on the value of the entropy between the flow data and the average value of the entropy of the flow. Even though the information entropy is more accurate for detecting abnormalities, it still needs to be combined with other technologies in determining the threshold and multi-element weight distribution. Said et al. [8] investigated that based on the analysis of the characteristics of each TCP / UDP / ICMP protocols and utilization of ANN training it is possible to detect DDoS attacks, to achieve that the method should distinguish among the packet protocols. However, this is quite difficult, as it is necessary to constantly analyze all packet headers.

On the other hand, the use of statistical approaches to anomaly detection, are not out. The authors of paper [9] proposed the SOM algorithm to detect DDoS-attacks, which works by extracting flow statistics in a certain time gap. But, the drawback of this method is that the behavior of the attack is not timely and inaccurate. In [10] the authors offered the mechanism for detecting DDoS attacks based on the analysis of anomalous characteristics of the source and destination IP address. However, the method does not take into account and does not adjust a certain threshold of IP address irregularity values.

In this paper, we proposed our approach to detecting anomalies based on log analysis. The threat detection approach is a parallel analysis of log files based on structured and unstructured log analysis. This will allow monitoring of both existing threats and forecasting attacks on the system as a whole.

2. Concepts of Detection of DDoS Attacks in SDN Networks Using Log Analysis

2.1.General concepts

Nowadays, it is difficult to find a system that works with web services without software regulation. SDN controller does management of data transferring and collection of statistical information. When it comes to web services, the main goal of the SDN controller is to pass the request to the appropriate application, so the controller is responsible for the security part of the interaction.

Web services are most exposed to DDoS attacks. Failure of some web services could lead to data loss or, worst case, to corruption of the system since modern web applications usually consist of a few services. Nowadays, there's no widespread instrument to deal with DDOS attacks. To cope with DDoS attacks, there are two main tasks to be performed:

1. An attack should be recognized as soon as possible.
2. Traffic should be classified into normal and abnormal.

Having an id of the user that created abnormal spikes in traffic it is possible to track down an IP of that user and block him.

The key idea is to teach the SDN controller to recognize attacks by means of information about flow state, the time of the session, and its source. We propose to get this data by means of the utilization of log analysis.

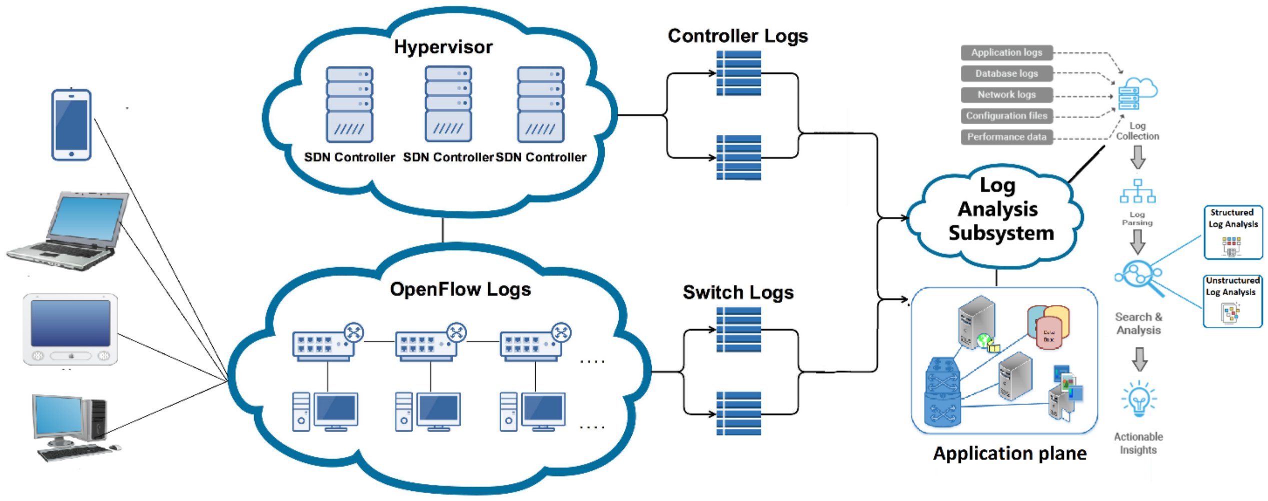


Figure 1: SDN architecture with DDoS attacks discovery.

We propose examining log files generated as a result of the user's activity. There are two possible ways to deal with data from the log files – structured and unstructured.

Structured data is highly organized, predefined and formatted to a set structure before being saved in data storage whereas unstructured data is not predefined by the data model, it might be human-generated, or machine-generated in a textual or a non-textual format.

Structured data could be universally understood, gives a possibility of utilization of various data tools, could be easily digested by various data programs and machine leering algorithms, and require much less space to be stored. Disadvantages include storage inflexibility (if there is a requirement to change the data structure the whole data set needs to be updated) and limited use cases (almost all the time data could be utilized only for the intended purposes, which causes some inflexibility).

To solve the issue with identifying abnormal traffic, we offer the utilization of the Kulbak-Leibler approach to spot flow abnormalities over the period of the session.

Unstructured data include wider use cases, flexible formatting, and easy storage. There is also a possibility to derive highly valuable insights from it which could not be obtained from the structured data sources. Disadvantages include difficulties in data preparation and analysis and requirement of data specific tools.

In our proposed system architecture depicted in Figure 1, all log events produced by a controller or a set of controllers are collected by a log management agent and periodically pushed to the log management and analysis platform deployed in the cloud. The same agent monitors the performance of the server running the SDN controller[s] and periodically pushes data to the log analysis subsystem. Similar to the SDN Control Plane, all logs produced by a single machine or multiple machines running network components in the infrastructure layer are collected by the log analysis subsystem and pushed to the log management platform for real-time analysis via TCP connections. Network administrators are now able to view, correlate and analyze SDN network logs in real-time or refer to historical data. To assure the reliability of the monitoring data, TCP should be used to transport log messages to the analysis platform. The log analysis subsystem on the basis of the received service information starts two independent processes of the log analysis: structured and unstructured.

In the next chapters, we will clarify approaches of structured and unstructured data analysis to achieve the system's goal.

2.2.Principles of Structured Log Analysis

Since most of modern firewalls function on the network core we could log all the requests sent by the users at this point since then all of them will be redirected to the black hole. After splitting the log file, it is possible to retrieve all needed information. Consequently, the created an analyzer that will function before the firewall.

Constantly parsing the log file could be costly task to perform. To overcome this problem, we will utilize a database and move to it all logs. This will boost the speed of search for the specific log entries. There are class of databases that holds all the data in RAM even though it is costly such approach boosts productivity tremendously. Utilization of such kind of a database would be the best option since time is very crucial when it comes to detection of DDoS attacks.

Date, IP address and port number (socket), amount of time the user spent connected to the service are the main properties of log entry. We propose the algorithm (shown in figure 2) that leverages those properties and is able to identify the malicious client).

Analytics subsystem perform a batch processing of all log entries recorded in last 24 hours from the moment of detection of suspicions activity. To get more precise result we separate the traffic from users that usually generate a log of it. To know whether the traffic is usual and does not diverge from the other users' traffic, we calculate the average number of requests among all the users except the spikes in users' traffic. M23 imply an average traffic from all the users (average number of requests). Next step to determine whether the average number of requests diverge from spikes in generated traffic. To achieve that, we calculate a correction factor. This will help to determine whether it is a DDoS attack or just normal spikes in users' traffic.

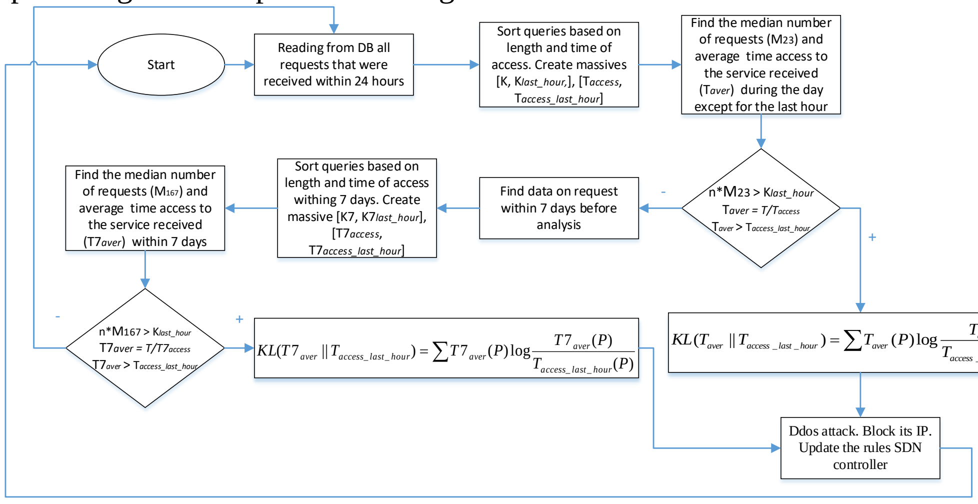
Alternative significant property is the time the user spent connected to the survival (session time). Based on it, we will train the system, to archive that we calculate the entropy of Kulbak-Labler. We will compare the session time with the average value from specific IP addresses, which were sorted as a result of the flow shown in figure 2. To calculate the KL divergence, which determines how much one probability distribution diverges from another one, so we compute the entropy for both distributions to determine the divergence. Appropriate formulas are presented by equations 1 and 2.

$$KL(T_{aver} \parallel T_{access\_last\_hour}) = \sum T_{aver}(P) \log \frac{T_{aver}(P)}{T_{access\_last\_hour}(P)} \tag{1}$$

$$H(T_{aver}) = \log T_{aver}(P) - KL \tag{2}$$

In our case, users should be segregated into groups, if the entropy of both distributions and their divergence are huge values then we could say that malicious user has been detected and give the appropriate information to controller to block that. If it is infeasible to detect an attack, we will take into account information about the user's activity from the last 7 days. As a result, the SDN controller will block the domains from where DDoS has been perfumed.

The proposed algorithm is presented in figure 2.



Start
Reading from DB all requests that were received within 24 hours
Sort queries based on length and time of access. Create massives [K, K_{last_hour}], [T_{access}, $T_{access\_last\_hour}$]
Find the median number of requests (M_{23}) and average time access to the service received (T_{aver}) during the day except for the last hour
$n*M_{23} > K_{last\_hour}$ $T_{aver} = T/T_{access}$ $T_{aver} > T_{access\_last\_hour}$
-
+
Find data on request within 7 days before analysis
Sort queries based on length and time of access withing 7 days. Create massive [K7, $K7_{last\_hour}$], [$T7_{access}$, $T7_{access\_last\_hour}$]
Find the median number of requests (M_{167}) and average time access to the service received ($T7_{aver}$) within 7 days
$n*M_{167} > K_{last\_hour}$ $T7_{aver} = T/T7_{access}$ $T7_{aver} > T_{access\_last\_hour}$
-
+
$KL(T7_{aver} \parallel T_{access\_last\_hour}) = \sum T7_{aver}(P) \log \frac{T7_{aver}(P)}{T_{access\_last\_hour}(P)}$
$KL(T_{aver} \parallel T_{access\_last\_hour}) = \sum T_{aver}(P) \log \frac{T_{aver}(P)}{T_{access\_last\_hour}(P)}$
Ddos attack. Block its IP. Update the rules SDN controller

Figure 2: The algorithm that recognizes the DDoS attack.

2.3.Principles of Unstructured Log Analysis

Even though we have the possibility to analyze structured data, it is worth paying attention to other data that the user sends to the server. For example, to detect a possible malicious action we could examine logs and payload of messages that users send to the server and calculate entropy based on the last time span to compare it with historical data.

The data source is going to be the same log files that the used in the previous section, the difference is that here we will be dealing with the rest of the log file data we haven't pay attention to in the previous section. Since the previous approach works on the assumption of structured log entries in the file the rest of the file is not taken to account at all. The other problem with the structured approach is that it could not take into account entries generated in the wrong format (even though connotating all needed data), those entries could be generated in the case of a software bag or simply could come from another part of a heterogeneous system.

The technique takes into account heterogeneous log sources with a universal analysis style.

Let' s say that our system is composed of K_i $(1 \le i \le S)$, services with each having a diverse log structure since log sections can be produced by a diverse range of frameworks, databases, middleware, etc. As one of the steps, we could filter out log entries with representing users performing operations in the system. For example, by looking for the user id in the log files we could find appropriate log entries which belong to the user. The figure 3 shows a computation topology that could be used to implement further described technic. One could use any kind of log oriented streaming frameworks and windowing technique to implement this topology, for instance, Apache Kafka could be utilized.

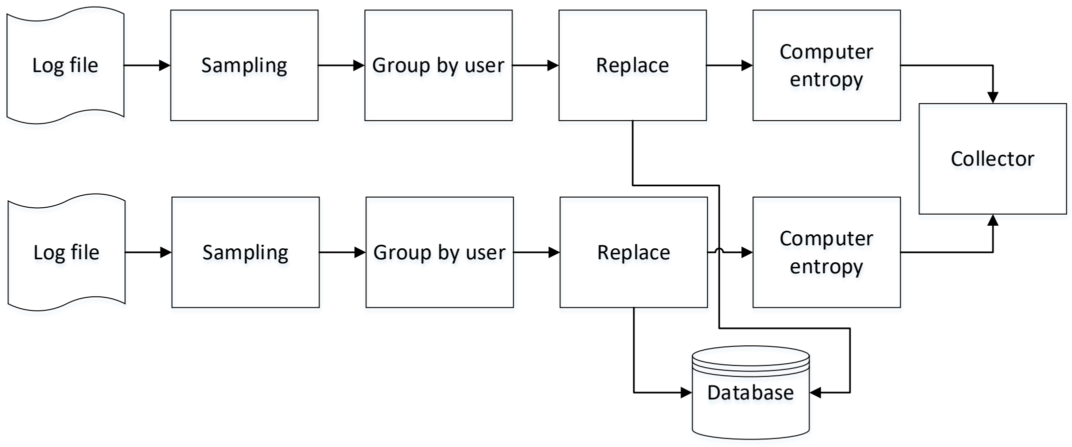


Figure 3: Method computation topology.

The proposed method is a scheduled process that takes a set of log entries from the file and computes the entropy of the entries generated during the last time span. Periodic task includes two vital steps terms/counts extraction and weighting.

Having log entries created in the last T_i terms extraction step employs a few steps to the log entries, the phase takes all the terms that compose the entries (a term is a few characters separated by whitespace), then counts a number of occurrences of those terms in chank generated in the last time span, whereas the next step computes the entropy based on the counts of the terms. Entropy is a numeric score that denotes the importance of the log entries, entries are more relevant if they have hinger score of entropy. If a term has never happened before in the log file or if it's has been emitted to many times in comparison to the historical timeline then the entropy of such term would be higher than usual. The collector is a task that obtains recent entropy values for all log files in the system. In the end, we could use Kulbak-Labler entropy to detect potential DDoS attacks.

Figure 5 could signify log entries structure in the system. For an instance, the term in figure 4 is a slice of the log entry that denote timestamp value, while other values could represent very specific values in some domain. Terms extraction employs some data tuning steps to terms, before entropy calculation will take place.

$$[``yyyy`/`MM`/`dd`:`HH`:`mm`:`ss`]$$

Figure 4: Term structure that represent timestamp.

Data preparation phase is very crucial for the correct end result of entropy calculation so there is a strong need for this phase. Log entries that have been generated under even regular system behavior could be filled with very specific rarely occurring terms (i.e., timestamps, value of variables, process and unrelated identifiers. Actually, a usual log entry contains constant and variable values.

The log entries like in figure x, share a common consistent structure, we could represent this structure by removing variable values from the log entries: “MESSEGE: router: KM message = * user = * “.

```
“MESSEGE: router: KM message = 0x312121 user = JY5_UID”
“MESSEGE: router: KM message = 0x317677 user = KJ7_UID”
```

Figure 5: Example of log entries which share common structure.

Constant terms do not change whereas variable fields tend to change very often so their values are likely to occur very rarely. This could alter the weighting phase, whose objective is to give larger entropy scores to seldom occurring terms created by suspicious system activity like DDoS attacks.

Having a bunch of log entries, data preparation phase goal to lessen variability of the terms created by the system. This goal could be archived by utilization diverse tasks based on a variety of practices from the information domain. For instance, terms exclusion removes unusual non-alphanumeric characters (e.g., +, “, and &). More critical is that each log entry is checked against a substitutions dictionary. The log entry is substituted with the equivalent pattern, if the pattern is present in the dictionary, the record is left as it is, otherwise. When all the entries have been organized, the terms extraction phase tokenizes the items and calculates the count for each term across the log file.

We suggest to use entropy of Kulbak-Labler as a numeric value that could reveal a potential DDoS attack that occurred in the last time span. To calculate the measure, we propose to build a matrix where a column of the matrix contains the term counts produced during the last time span, the leftmost column would contain the term counts gained from the period of time T where the system was under regular system activity. The term counts of the rightmost column are obtained in real time, while the other columns could be set offline. The leftmost column is constant, whereas the rightmost is constantly overwritten. At this point we could use those data to calculate Kulbak-Labler between the distribution of terms regarding last timestamp and historical data, using formula 3.

$$KL(K_{regular} \| T_{last\_time\_span}) = \sum T_{regular}(P) \log \frac{T_{regular}(P)}{T_{last\_time\_span}(P)} \quad (3)$$

Spikes in Kubak-Laber entropy could signalize malicious user activity, so we could notify system administrator to block the appropriate user and at the same time, we could start the process of analysis of structured data from the previous section, an algorithm detailed in figure 6.

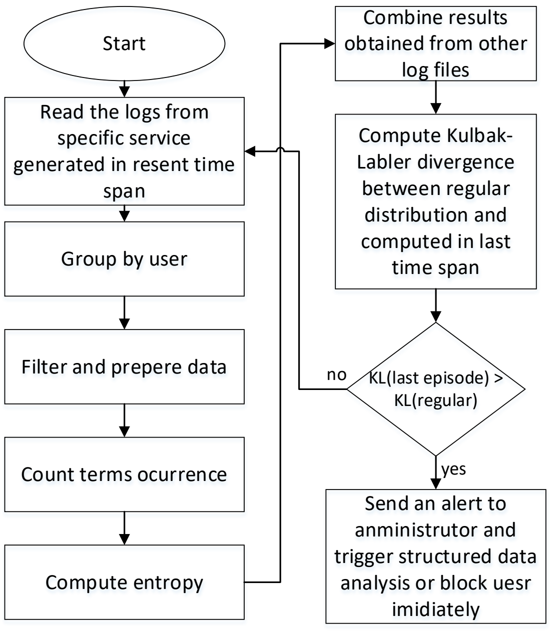
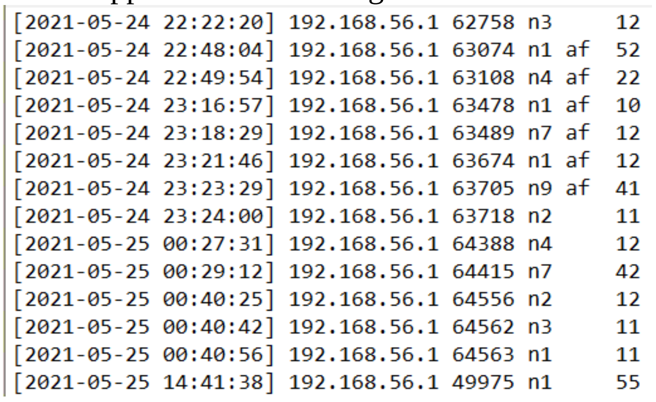

Start
Read the logs from specific service generated in resent time span
Group by user
Filter and prepere data
Count terms ocurrence
Compute entropy
Combine results obtained from other log files
Compute Kulbak-Labler divergence between regular distribution and computed in last time span
KL(last episode) > KL(regular)
no
yes
Send an alert to anministrutor and trigger structured data analysis or block uesr imidiately

Figure 6: Algorithm to recognize DDoS attack using unstructured data analysis.

3. Web Applications Security Monitoring Using Log Analysis Subsystem

For investigation of efficiency of the developed system for detecting DDoS attacks we performed simulation using Java and Python programming languages. The socket module of the Python programming language is used to establish a connection between clients and the server. This module provides a user-friendly and consistent interface for API Berkeley sockets. Log data from the file generates data according to the normal distribution law. Before starting the web server, random data is generated and written to a log file. Data in file is not consistent over time, so they are further sorted before recording to the database.

For correct work150 000 entries are written into the eventLogs.txt file. For them, the IP address and time are generated according to the Gaussian distribution law and in case of error it is recorded. The contents of the log file will appear as shown in figure 7.

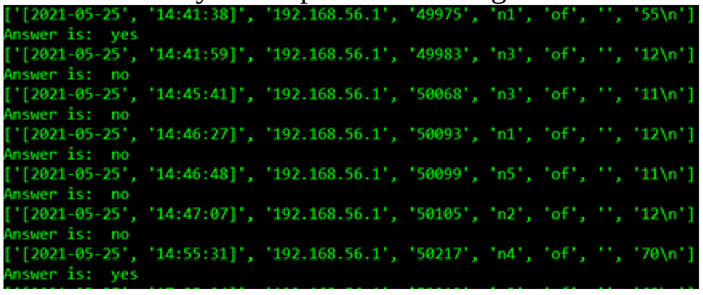


```
[2021-05-24 22:22:20] 192.168.56.1 62758 n3     12
[2021-05-24 22:48:04] 192.168.56.1 63074 n1 af  52
[2021-05-24 22:49:54] 192.168.56.1 63108 n4 af  22
[2021-05-24 23:16:57] 192.168.56.1 63478 n1 af  10
[2021-05-24 23:18:29] 192.168.56.1 63489 n7 af  12
[2021-05-24 23:21:46] 192.168.56.1 63674 n1 af  12
[2021-05-24 23:23:29] 192.168.56.1 63705 n9 af  41
[2021-05-24 23:24:00] 192.168.56.1 63718 n2     11
[2021-05-25 00:27:31] 192.168.56.1 64388 n4     12
[2021-05-25 00:29:12] 192.168.56.1 64415 n7     42
[2021-05-25 00:40:25] 192.168.56.1 64556 n2     12
[2021-05-25 00:40:42] 192.168.56.1 64562 n3     11
[2021-05-25 00:40:56] 192.168.56.1 64563 n1     11
[2021-05-25 14:41:38] 192.168.56.1 49975 n1     55
```

Figure 7: Generated log file.

In case of structured data analysis, the system parses the log and converts log entries from each connection to the model, which preform a check whether user is malicious. If the client is spotted, his ID is saved into the Check.txt file.

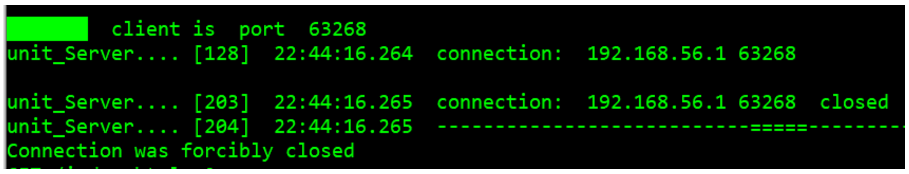
At the same time unstructured data analysis takes place in accordance to algorithm presented in figure 6, as a result of this analysis the user could be placed directly to Check.txt file or an additional check could be performed by means of structured data analysis. The results of detecting atypical traffic as a result of such analysis are presented in Figure 8.



```
['[2021-05-25', '14:41:38]', '192.168.56.1', '49975', 'n1', 'of', '', '55\n']
Answer is:  yes
['[2021-05-25', '14:41:59]', '192.168.56.1', '49983', 'n3', 'of', '', '12\n']
Answer is:  no
['[2021-05-25', '14:45:41]', '192.168.56.1', '50068', 'n3', 'of', '', '11\n']
Answer is:  no
['[2021-05-25', '14:46:27]', '192.168.56.1', '50093', 'n1', 'of', '', '12\n']
Answer is:  no
['[2021-05-25', '14:46:48]', '192.168.56.1', '50099', 'n5', 'of', '', '11\n']
Answer is:  no
['[2021-05-25', '14:47:07]', '192.168.56.1', '50105', 'n2', 'of', '', '12\n']
Answer is:  no
['[2021-05-25', '14:55:31]', '192.168.56.1', '50217', 'n4', 'of', '', '70\n']
Answer is:  yes
```

Figure 8: The analysis of the log and identify malicious clients.

When the user establishes connection to the service, the service checks his ID with the previously spotted malicious users which are stored in the Check.txt file. If they equal, the controller blocks the user.



```
[redacted]   client is  port  63268
unit_Server.... [128]  22:44:16.264  connection:  192.168.56.1 63268

unit_Server.... [203]  22:44:16.265  connection:  192.168.56.1 63268  closed
unit_Server.... [204]  22:44:16.265  -------------------------====---------
Connection was forcibly closed
```

Figure 9: Blocking by the controller of the malicious client.

4. Conclusions

In this paper, we proceed to study the availability of web services in SDN networks and spotting DDoS attacks by means of structured and unstructured log analysis. In this paper, we offer to teach the SDN controller to spot DDoS attacks utilizing information about the session state, its time, and source, extracting data from logs generated by the services. To archive this, it is essential to classify the traffic into abnormal and normal. In the case of structured data analysis, we propose the utilization of Kulbak-Leibler approach to spot anomalies over the user's session. We also propose unstructured data analysis from system logs and user's generated data to spot malicious users by means of Kulbak-Leibler approach in real-time. After successful and timely detection, we are able to form the rules to block him.

References

[1] Bawany, N., Shamsi, J. & Salah, K., 2017. DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions. Arabian Journal for Science and Engineering, Volume 42, p. 425–441.

[2] Braga, R., Mota, E. & Passito, A., 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow. Denver, s.n., p. 408–415.

[3] Dawoud, A., Shahristani, S. & Raun, C., 2018. Deep learning and software-defined networks: Towards secure IoT architecture. Internet of Things, Volume 3, p. 82–89.

[4] Giura, P. & Wang, W., 2012. Using large scale distributed computing to unveil advanced persistent threats. Science J, Volume 1, p. 93–105.

[5] Gonçalves, D., Bota, J. & Correia, M., 2015. Big data analytics for detecting host misbehavior in large logs. In: Trustcom/BigDataSE/ISPA, 2015 IEEE. s.l.:s.n., p. 238–245.

[6] Hu, Q., Tang, B. & Lin, D., 2017. Anomalous user activity detection in enterprise multi-source logs. New Orleans, LA, USA, s.n., p. 797–804.

[7] Klymash, M., Peleh, N., Shpur, O. & Hladun, S., 2020. Monitoring of Web Service Availability in Distributed Infocommunication Systems. Lviv-Slavske; Ukraine, s.n., p. 723–728.

[8] Landauer, M. et al., 2018. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. Computers & Security, Volume 79, p. 94–116.

[9] Lin, H. & Wang, P., 2016. Implementation of an SDN-based security defense mechanism against DDoS attacks. Pennsylvania, s.n.

[10] Raj, A., Truong-Huu, T., Mohan, P. & Gurusamy, M., 2019. Crossfire Attack Detection using Deep Learning in Software Defined ITS Networks. Kuala, s.n.

[11] Saied, A., Overill, R. & Radzik, T., 2016. Detection of known and unknown DDoS attacks using Artificial Neural Networks. Neurocomputing, Volume 172, p. 385–393.

[12] Shu, X., Smiy, J., Yao, D. & Lin, H., 2013. Massive distributed and parallel log analysis for organizational security. In: 2013 IEEE Globecom Workshops (GC Wkshps). s.l.:s.n., p. 194–199.

[13] Smith, R., Zincir-Heywood, A., Heywood, M. & Jacobs, J., 2016. Initiating a Moving Target Network Defense with a Real-time Neuro-evolutionary Detector. New York, New York, USA, s.n., p. 1095–1102.

[14] Ten, C., Manimaran, G. & Liu, C., 2010. Cybersecurity for critical infrastructures: Attack and defense modeling. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, July, Volume 40, p. 853–865.

[15] Wang, X., Chen, M., Xing, C. & Zhang, T., 2016. Defending DDoS attacks in software-defined networking based on legitimate source and destination IP address database. IEICE Transaction on Information and Systems, Volume 99, p. 850–859.

[16] Yang, J., Wang, X. & Liu, L., 2016. Based on traffic and IP entropy characteristics of DDoS attack detection method. Application Research of Computers, Volume 33, p. 1145–1149.