Spark-Based Anomaly Detection: the Case of Port and Net Scan

Antonia Affinito*, Alessio Botta*+, Luigi Gallo*, Mauro Garofalo*, Giorgio Ventre*+
*Department of Electrical Engineering and Information Technology
University of Napoli Federico II, Napoli, Italy
+NM2 SRL, Napoli, Italy
{antonia.affinito, luigi.gallo10}@studenti.unina.it, {alessio.botta, mauro.garofalo, giorgio.ventre}@unina.it

Abstract—Today, due to the high number of attacks and of anomalous events in network traffic, the network anomaly detection has become an important research area. In fact it is necessary to detect all behaviors which do not comply with a well-defined notion of a normal behavior in order to avoid further harms. The two most spread network anomalies related to network security are port and net scan, activities performed by a malicious host to find and examine potential victims. In this work, a novel approach for detecting port and net scan using Big Data Analytics framework is presented. The approach works at flow-level and has been conceived to detect such anomalous events on high-speed networks in a short time.

In accordance with this approach, an algorithm has been created able to detect IP addresses that generate port and net scanning activities, and suited for the execution on Apache Spark framework. The paper describes the approach and the algorithm proposed and then presents an experimental analysis of its performance, containing also a comparison with MAWIlab archive. The execution time of the algorithm has also been experimentally evaluated, running Apache Spark on a private Cloud and proved to be very short even on large traffic traces. Moreover, the results of this comparison show that the algorithm is highly efficient in terms of Precision and Recall for port and net scan detection. Anomalies not detected by the gold standard are also detected by our approach.

Index Terms—Anomaly Detector, Port Scan, Net Scan, Apache Spark, MAWI and MAWI Lab, Precision, Recall.

I. INTRODUCTION

In the last years the Internet has become a fundamental tool for technological and scientific innovation, e.g. the web is visited every day by billions of people for a variety of reasons. The very nature of the Internet and more generally of the network, which allows people to connect independently, makes them prone to human errors or malicious voluntary attacks. It is therefore very important to detect these events, which can be dangerous both for the network and the targets. Therefore, an efficient and effective anomaly detection system able to continuously monitor the traffic and to raise an alert if it finds a trend in the data different from the normal is necessary.

The immoderate use of the Internet has led to significant increase in the amount of traffic that crosses the network every day, so the amount of data that such a detector has to analyze is higher and higher, especially in high-speed networks. Network traffic can be analyzed at several layers of the protocol stack: packet, flow, etc. In the first case, every single packet is analyzed, which is nowadays a very intensive computational task: packets can flow at rates up to tens or hundreds of millions per second in modern networks, which leaves a few nanoseconds of time to process a single packet. At flow-level, instead, packets belonging to the same TCP or UDP communication (e.g. all packets related to an HTTP communication from and to a single host and a web server) are grouped together, and a summary of such group of packets is retained, discarding the single packets. These summaries can now be provided directly by network devices such as switches and routers and standard protocols have been defined for this aim (e.g. Internet Protocol Flow Information Export or IPFIX [1]). Therefore, working at flow level seems the most promising approach for high-speed networks. However, even if traffic is analyzed at the flow-level, the detection of anomalies in high-speed networks still require enormous computational power or data reduction as flow records in these networks represent a huge quantity of data.

In this article, we present an approach to detect anomalies in high-speed networks working at flow-level. We concentrate on the two most spread network anomalies related to network security, i.e. port and network (or simply net) scan. In the former case, a malicious host (the attacker) probe another host (the victim) on a number of TCP/UDP ports to find active and vulnerable services. In the latter case, the attacker scans a group of victim hosts on a single or a small number of TCP/UDP ports. These activities generally represent the first step an attacker makes towards compromising a system, and they are also associated with worm and botnet activities.

Port and net scans generate a very specific pattern in network traffic, which seems to be easy to detect. However, its detection is far from being simple in current, high-speed networks. The main reason is related to the huge amount of data to be analyzed, which prevents many approaches from being effective in real time. To cope with this problem, we propose an approach for net and port scan detection based on Big Data Analytics. In particular, we use the Apache Spark framework (simply Spark in the following), similar to Hadoop Map/Reduce, which uses a cluster of computers to perform a distributed computation, but much faster, also because it works completely in memory. We implement a simple port and net scan detection algorithm in Spark and test it by using several real traces. Comparing our detection results with the available literature, we show that our approach is more
accurate than the available gold standard in terms both of recall and precision. We also show that in many cases, our approach can detect more anomalies than the gold standard. Finally, we also present results in terms of detection time, showing that we achieve a processing rate up to 1/16 of the trace duration, which makes the approachable to run in real time in high-speed networks.

II. STATE OF THE ART

Anomalies of Network and Port Scan are treated for several years in different research fields in the scientific literature. Generally, they are examined considering that a source of a scanning activities shows a very high number of outgoing connection. This hypothesis has been also made by Zhao et al., who present a new algorithm that provides accurate and efficient solutions to the problem of scan detection. To cope with high connection speeds (10-40 Gb), this algorithm samples the flows based on standard hash and sampled traffic is further filtered by a data streaming module. The latter makes it possible to obtain a much higher sampling rate and a better precision than the one achievable with hash-based flow sampling.

The main goal of such the method is the same as the one proposed in this work, but, unlike the latter, it also includes the sampling of flows to obtain a reduction of the dataset. To overcome this problem, the approach proposed in this work is based on the processing of the entire traffic through specific frameworks, such as Apache Spark. Kim et al. describes a scanning activity in terms of traffic models. Particularly, like in this work, the traffic is analyzed at flow-level, but the scanning anomalies are detected by examination of changes in network traffic models.

Chan et al. propose two machine learning methods, useful for the constructing models for detecting network anomalies detection starting from a past behavior. In addition, the approach described by Wagner et al. is based on probabilistic measurement of entropy that is used to indicate regularity in traffic of network flows. Unlike this work, in which the anomalies detection is based on heuristics, traffic models used in the three last works are more sensitive to changes in the type of traffic and network in which such detection is made. Finally, MAWI Lab team proposed a system to detect attacks or anomalous events, using a combination of four detectors based on different theoretical backgrounds to identify anomalous, suspicious, or benign traffic.

III. APACHE SPARK

Apache Spark is a platform for fast and efficient distributed processing of Big Data which has almost completely substituted Hadoop. In fact, it is very fast both in storage and in data processing because of supports in-memory processing that allows querying data directly in main memory. For this reason, Apache Spark is 100 times faster than Apache Hadoop that requires the use of hard disk.

Apache Spark allows to work in two different ways: Batch and Streaming. In Batch mode processing only starts after all data has been stored while in Streaming mode data are processed in real time. An efficient anomalies detector should analyze traffic and detect suspicious behaviors in the shortest time possible and so it would be advantageous to use streaming modality. In this work both batch mode and streaming modes were used, where switching between the two is very simple.

Apache Spark allows data storage in three different types of structure: Resilient Distributed Dataset (RDD), Dataframe, and Dataset. In this work the data frame structure is used: it is conceptually equivalent to a table in a relational database, a distributed collection of data organized in named columns. On this types of tables, you can make SQL queries using SQL commands, whose results are still a data frame.

Apache Spark supports different programming languages such as Java, Python, and Scala. In this work Scala was used for the two main reasons below:

- Apache Spark is built on Scala and so if there are errors in the code or the source code does not have the expected result it is easier to debug;
- Scala is about 10 times faster than others, such as Python, in analyzing and processing data thanks to the presence of the Java Virtual Machine.

IV. USED DATA

The proposed approach has been tested using real traffic traces from the Dataset MAWI (Measurement and Analysis of the WIDE Internet): an archive of traces of real traffic provided by the MAWI Working Group. Traces are captured since 2007 and therefore it constitutes a rich dataset that includes different applications and network conditions, including the presence of various known anomalies with global or local impact, periods of congestion, and network reconfiguration.

Traffic traces are captured inside the backbone WIDE network, on a transoceanic link that connects Japan and United States of America. Each trace is characterized by traffic captured every day from 14:00 to 14:15 in different locations inside WIDE. Traces of 24 and 48 hours are also occasionally collected. A 15-minute trace, characterized by anonymous IP addresses and without payload, usually contains 300k-500k unique IP addresses and various type of anomalies. The traces we consider are captured at Samplepoint-F, a link working at 1 Gbps with an average load of 150 Mbps.

MAWI group also presented the MAWI Lab project, an approach to the identification of network anomalies and provides archives of anomalies present in MAWI traces. This database helps researchers to measure the detection rate of the anomaly detector realized. In fact, results of this detector can be accurately compared with the labels provided by
MAWILab uses a combination of four anomaly detectors based on different theoretical backgrounds: Principal Component Analysis (PCA), Gamma distribution, Kullback Leibler (KL) divergence, and Hough transformation. Such detectors only work on the IP header [13].

The results of this detector are combined to classify the anomalies in four types of records, as reported below [14]:

- **Anomalous**: assigned to all abnormal traffic and should be identified by any efficient anomaly detector, according to the authors.
- **Suspicious**: assigned to all traffic that is probably anomalous but not clearly identified by our method.
- **Notice**: assigned to all traffic that is not identified anomalous but has been reported by at least one anomaly detector. According to the authors, this traffic should not be identified by any anomaly detector, but they do not label it as benign in order to trace all the alarms reported by the combined detectors.
- **Benign**: all the rest of the traffic where no detector has labeled it as abnormal.

For each trace of MAWI archive, MAWILab provides the corresponding analysis in two files, Anomalous and Notice, in which there is all information on anomalies found. After detecting the anomalies as reported above, MAWILab uses a heuristic to assign a label related to the type of anomaly. Possible labels are represented in tree-based taxonomy, where the root is a generic event and nodes contain anomaly label.

In this work, we use MAWILab archive as a Gold Standard [15] to validate our method as we will see in VI-B.

V. THE PROPOSED APPROACH

The algorithm proposed in this paper for port and net scan detection is described below. Fig. 2 shows the flow chart of the algorithm. First of all, the input data is processed in such a way that the entire trace is divided into time intervals, denominated slices, of arbitrary duration (e.g. 30 seconds). Afterward, two SQL queries are executed on the couples (source IP address, slice) and (destination IP address, slice) so as to obtain the number of source and destination flows respectively in each slice. The results obtained from these two queries are merged through a Full Outer Join to include all the rows of both tables, whether or not there are corresponding values. In this way, all the couples (IP address, slice) that present same values in both tables are grouped together with the ones that are present in the first but not in the second and viceversa [16].

After that, the ratio between generated and received flows in each slice and from each IP address is calculated and it is compared with a Threshold value. In this way, only the IP for which the amount of received flows is much larger than the ones generated are considered. The ratio which satisfies the following condition are considered:

\[
\frac{\#\text{generated flows}}{\#\text{received flows}} > \text{Threshold} \quad (1)
\]

An IP address shall be considered sender or receiver of scanning activities if the ratio satisfies the inequality. As we will see in VI, this simple algorithm is an anomaly detector based on the threshold value that detects port and net scan anomalies with high precision and in a very short execution time.

Although, this algorithm cannot detect other types of anomalies or more sophisticated attacks by design. It is a particular approach to net and port scan detection that does not require traffic sampling or modeling. It is also an example of how the Big Data Analytics framework can solve long-term problems in this important field of research.

VI. VALIDATION

The execution of the Anomaly Detection algorithm, described in Sec. V, provides in output the list of IP addresses sources of anomalous behavior. For validation purposes, these addresses are compared with the ones in two files proposed by the gold standard MAWILab: Anomalous.xml and Notice.xml. The latter ones are characterized by a list of source and destination IP addresses that present an anomalous or suspicious behavior, with a related anomaly label.

Three validation cases will be analyzed in the following: comparison with MAWILab (Sec. VI-A), comparison with filtered MAWILab (Sec. VI-B), and deeper investigations on false positives (Sec. VI-C).

A confusion matrix is also generated from the comparison between the results obtained by the algorithm and the ones provided by MAWILab. Such a matrix, also named error matrix, allows to describe the performance of a classifier on a set of test data for which real values are known. The elements of Confusion Matrix are the following:

- **True Positive**: addresses IP that are considered anomalous by both algorithm and MAWILab.
- **True Negative**: addresses IP that are not considered anomalous either by the algorithm or by MAWILab.
- **False Positive**: addresses IP that are considered anomalous by the algorithm but not by MAWILab.
- **False Negative**: addresses IP that are considered anomalous by MAWILab but not by the algorithm.

To evaluate the efficiency of the anomalies detector implemented, two metrics are used: Recall, also defined as True Positive Rate, and Precision. The first is a measure of completeness and is defined as follows:

\[
\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (2)
\]

Precision is a measure of Correctness and is defined as follows:

\[
\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (3)
\]

Different traces have been analyzed. The results reported in the following refer to eight traces, characterized by a duration of 15 minutes, collected in January 2018. Similar results have been obtained on the other traces. Different threshold values are considered in this analysis, three are reported are reported...
A. Comparison with MAWILab

Let us start analyzing the results obtained comparing the IP addresses resulted anomalous by the algorithm implemented with the ones provided by MAWILab in the files Anomalous.xml and Notice.xml. Tables I and II contain the mean and variance values of the Recall and Precision, respectively, for the eight traces analyzed. Recall decreases both with a threshold value and the amount of IP addresses IP, i.e. switching from anomalous to notice. Precision, instead, increases both with threshold value and the amount of IP addresses considered. Both Recall and Precision values are not satisfying. Performing a manual inspection on False Negative, many IP addresses associated with anomalies that the implemented algorithm is not able to detect because it was not designed to detect them have been noted. That is why we performed the following analysis.

| Threshold | Anomalous.xml | Notice.xml | Total    |
|-----------|---------------|------------|----------|
| 50        | 0.293±0.009   | 0.209±0.038| 0.162±0.001|
| 100       | 0.266±0.008   | 0.097±0.001| 0.131±0.001|
| 200       | 0.211±0.004   | 0.174±0.049| 0.237±0.064|

**TABLE I:** First Case: Mean and Variance of Recall

| Threshold | Anomalous.xml | Notice.xml | Total    |
|-----------|---------------|------------|----------|
| 50        | 0.124±0.005   | 0.230±0.007| 0.350±0.013|
| 100       | 0.229±0.015   | 0.351±0.017| 0.575±0.027|
| 200       | 0.339±0.030   | 0.381±0.029| 0.714±0.026|

**TABLE II:** First Case: Mean and Variance of Precision

B. Comparison with filtered MAWILab

In this section, we analyze results obtained performing a filtering on MAWILab results: all anomalies that are not part of scanning activities have been removed from their results (e.g. normal events, Denial of Services, Distributed Denial of Services). Moreover, since in the analyzed traces come all the protocols that are not TCP or UDP we have filtered from MAWILab all the anomalies that are not relevant to these two protocols and also anomalies based on ICMP because packets of this protocol are not contained in the analyzed traces. Table III and Tab. IV contain the mean and the variance values of the Recall and Precision obtained. In this case, Recall has higher values than the previous case (Sec. VI-B), while for the Precision there are not significant changes. For this reason, a manual analysis of False Positive was carried out to verify whether the algorithm generates false alarms or MAWILab does not efficiently detect these types of anomalies. This manual inspection revealed that many False Positive was really a source of the scanning activity and, therefore, MAWILab was unable to detect them. For this reason, we consider MAWILab as Gold Standard and not as Ground-truth. This drove us to perform the analysis reported in the following.
Anomalous.xml  0.103 ± 0.005  0.455 ± 0.009  0.508 ± 0.008
Notice.xml  0.608 ± 0.017  0.376 ± 0.009  0.442 ± 0.007

Total  0.711 ± 0.012  0.831 ± 0.018  0.453 ± 0.002

TABLE III: Second Case: Mean and Variance of Recall

Threshold | Anomalous.xml | Notice.xml | Total
--- | --- | --- | ---
50 | 0.103 ± 0.004 | 0.178 ± 0.006 | 0.278 ± 0.012
100 | 0.202 ± 0.013 | 0.302 ± 0.016 | 0.498 ± 0.027
200 | 0.321 ± 0.027 | 0.341 ± 0.023 | 0.665 ± 0.026

TABLE IV: Second Case: Mean and Variance of Precision

C. Deeper investigations on false positives

We have implemented three post-processing rules in order to complement MAWILab result for all the cases in which a source of the scanning activity was not detected by MAWILab. In this way, the True Positive are reintegrated of False Positive into True Positive. The rules are reported the following:

1) An IP address is a generator of **Net Scan** if it generates at least 20 flows towards different IPs of the same subnet.
2) An IP address is a generator of **Port Scan** if it contacts another IP on more than 10 ports.
3) An IP address is a generator of **Net Scan and Port Scan**, if it generates at least 20 flows per slice towards different IPs on known ports.

In the third case, these rules were applied to the results of the second case.

Fig. 3 shows the trend of the Recall (left) and the Precision (right) in the last two cases analyzed, for three threshold values: 50, 100, 200. So, the line reported in red color represents the comparison with filtered MAWILab, while the one in blue color represents the comparison with filtered MAWILab plus the rules reported above. Both in the second and third cases, by increasing the threshold value, the Recall increase and the Precision decrease showing higher values in the third case. The Precision in the last case has values close or equal to the ideal value, while the Recall has values close to the ideal value especially with a threshold value of 50.

D. Speed analysis

After a performance analysis in terms of Recall and Precision seen above, an analysis of execution times of the algorithm was carried out.

Traffic traces with duration of 24 hours have been taken from MAWI Dataset and provide in input to the algorithm. Different configurations of virtual machines have been instantiated on a private cloud in order to realize the typical scenarios of Apache Spark: a single master machine and different worker instances:

- **First Configuration**: A machine with 32Gb of Ram and 4 cores.
- **Second Configuration**: Two machines with 32Gb of Ram and 4 cores.
- **Third Configuration**: A machine with 64Gb of Ram and 8 cores.

Fig. 4 shows the box plot of the execution times related to the three configurations: the x-axis represents the three configurations, instead the y-axis represents the ratio between the execution time and the duration of the trace. In the first configuration, the median execution time is about 1/14 of trace duration. Instead, in the second and in the third configuration, that represent respectively more machines and more resources, the execution time is shorter than the first one. In addition, there is no particular improvement between the second and the third configuration, instead of the first one where a larger variance is observed.

![Box Plot of execution times related to three configurations](image)

**Fig. 4: Box Plot of execution times related to three configurations**

It is worth noting that MAWILab does not provide the labels for 24-hours traces and, therefore, it is no possible compare that with the results of the algorithm. For this reason, IP addresses resulting anomalous from our algorithm have been assigned directly the three rules where satisfied. Tab. V reports the results obtained. It can be noted that the amount of IP addresses responsible to port scan label is decisively high.

| Hours               | Threshold | Anomalous IP | True Positive | False Positive(%) |
|---------------------|-----------|--------------|---------------|-------------------|
| From 00:00 To 01:45 | 50        | 4539         | 98.39%        | 1.61%             |
|                     | 100       | 1453         | 99.11%        | 0.90%             |
|                     | 200       | 112          | 99.56%        | 0.44%             |
| From 02:00 To 03:45 | 50        | 4672         | 92.89%        | 7.11%             |
|                     | 100       | 1684         | 81.14%        | 18.86%            |
|                     | 200       | 773          | 71.70%        | 28.30%            |
| From 04:00 To 05:45 | 50        | 4165         | 92.34%        | 7.66%             |
|                     | 100       | 1613         | 82.63%        | 17.37%            |
|                     | 200       | 621          | 99.56%        | 0.44%             |
| From 06:00 To 07:45 | 50        | 2214         | 95.80%        | 4.20%             |
|                     | 100       | 1243         | 97.53%        | 2.47%             |
|                     | 200       | 710          | 99.56%        | 0.44%             |
| From 08:00 To 09:45 | 50        | 5484         | 97.39%        | 2.61%             |
|                     | 100       | 2834         | 98.45%        | 1.55%             |
|                     | 200       | 1480         | 99.93%        | 0.07%             |
| From 10:00 To 11:45 | 50        | 3566         | 96.94%        | 3.06%             |
|                     | 100       | 1825         | 98.58%        | 1.42%             |
|                     | 200       | 968          | 99.00%        | 0.37%             |

TABLE V: Label assignment- trace of 24 hours

VII. Conclusion

The problem of network anomaly detection is treated in a different context in literature and it is in continuous evolution.
Many proposals have been presented for this important research problem, but still, today is difficult to detect efficiently and with higher precision these kinds of anomalies. In the area of network security, two well known malicious activities are port and net scans. In this work, a novel flow-level algorithm for detecting port and net scans has been presented. To cope with the large size of the data to be examined, typical in high-speed networks, Big Data Analytics framework and Apache Spark, in particular, was used. Taking advantage of this framework, the algorithm presents short execution times. In fact, they vary between 1/12 and 1/16 of the duration of the trace. Real traffic traces have been used to validate the algorithm.

Our results have been compared with a Gold Standard, i.e. MAWI Lab, to evaluate Recall and Precision. Results have shown that the algorithm achieves high performance in terms of Recall and Precision and, in particular, through further analysis it was found that the algorithm is more effective in detecting malicious scanning activity than MAWI Lab.

REFERENCES

[1] J. Quittek, T. Zseby, B. Claise, S. Zander, “RFC3917: Requirements for IP Flow Information Export (IPFIX),” https://tools.ietf.org/html/rfc3917, 2004.

[2] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, “An Overview of IP Flow-Based Intrusion Detection,” IEEE Communications Surveys Tutorials, vol. 12, no. 3, pp. 343–356, Third 2010.

[3] Q. Zhao, J. Xu, and A. Kumar, “Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation,” IEEE Journal on Selected Areas in Communications, vol. 24, no. 10, pp. 1840–1852, Oct 2006.

[4] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, “A flow-based method for abnormal network traffic detection,” in 2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507), vol. 1, April 2004, pp. 599–612 Vol.1.

[5] M. V. M. Philip K. Chan and M. H. Arshad, “Learning rules and clusters for anomaly detection in network traffic,” https://pdfs.semanticscholar.org/2058/571b2e2b8a32560ab96bb7caf266499288c.pdf, July 2018.

[6] A. Wagner and B. Plattner, “Entropy based worm and anomaly detection in fast ip networks,” in 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05), June 2005, pp. 172–177.

[7] “Welcome to apache™ hadoop®!” https://hadoop.apache.org/

[8] T. R. S. Abdul Ghaffar Shoro, “View of big data analysis: Apache spark perspective,” vol. 15, 2015.

[9] “spark streaming - spark 2.3.0 documentation.”

[10] D. V. K. J. V. K. Tarun Kumawat, Pradeep Kumar Sharma, “Implementation of spark cluster technique with scala,” November 2012.

[11] R. Fontugne, P. Abry, K. Fukuda, D. Veitch, K. Cho, P. Borgnat, and H. Wendt, “Scaling in internet traffic: A 14 year and 3 day longitudinal study, with multiscale analyses and random projections,” IEEE/ACM Transactions on Networking, vol. 25, no. 4, pp. 2152–2165, Aug 2017.

[12] P. Borgnat, G. Dewaele, K. Fukuda, P. Abry, and K. Cho, “Seven years and one day: Sketching the evolution of internet traffic,” in IEEE INFOCOM 2009, April 2009, pp. 711–719.

[13] J. Mazel, R. Fontugne, and K. Fukuda, “A taxonomy of anomalies in backbone network traffic,” in 2014 International Wireless Communications and Mobile Computing Conference (IWCMC), Aug 2014, pp. 30–36.

[14] “Mawilab - documentation,” http://www.fukuda-lab.org/mawilab/documentation.html#admcl

[15] N. Radboud University Nijmegen Medical Centre, Nijmegen, “The gold standard: not a golden standard,” https://www.ncbi.nlm.nih.gov/pmc/articles/PMC557893/, August 2018.

[16] “Using Outer Join,” https://technet.microsoft.com/it-it/library/ms187518(v=sql.105).aspx, July 2018.