Received: 3 July 2019 | Revised: 7 November 2019 | Accepted: 14 November 2019

DOI: 10.1002/spy2.106

RESEARCH ARTICLE

WILEY

BackStreamDB: A stream processing engine for backbone traffic monitoring with anomaly detection

Elias P. Duarte Jr[1,2] | **Carmem Hara**[1,2] | **Pedro Torres Jr**[1,2] | **Christian Gomes**[1,2]

[1]Department Informatics, Federal University of Parana (UFPR), Curitiba, Brazil

[2]Brazilian Academic Network (RNP), PoP-PR, Brazil

Correspondence
Elias P. Duarte Jr, Department Informatics, Federal University of Parana (UFPR), Brazil.
Email: elias@inf.ufpr.br

Funding information
CNPq—The Brazilian Research Council, 311451/2016-0

Abstract

BackStreamDB is distributed traffic monitoring system based on a stream processing engine (SPE) designed to monitor the traffic of wide area backbones. BackStreamDB provides arbitrary metrics about the traffic in real time, taking into account the backbone as a whole. The system was developed for and successfully deployed on the Brazilian National Academic Network (RNP). In this work, we describe the functionality for the detection of traffic anomalies. A large number of Internet attacks are continuously reported, and several types of attacks result in anomalous traffic. In the proposed strategy for anomaly detection, the traffic is sampled by monitors that are distributed across the backbone, which are accessed and processed by the SPE. BackStreamDB was extended with stream processing modules for computing traffic entropy and principal component analysis, which are the employed to detect traffic anomalies. Experimental results are reported which were obtained to validate the effectiveness of the proposed strategy for different types of attacks.

KEYWORDS

anomaly detection, entropy, principal component analysis (PCA), stream processing engine (SPE), traffic monitoring

1 | INTRODUCTION

BackStreamDB is a distributed monitoring system for wide area backbones that integrates a stream processing engine (SPE) and a flow monitoring tool within a framework designed for large-scale backbones. SPEs[1] were proposed to provide the same basic features found in traditional database management systems (DBMSs), but operations are executed in real time on continuous data streams. In the case of BackStreamDB, the data stream is the network traffic itself. With this system, the network administrator can use a high level SQL-like language to issue arbitrary queries about the network traffic. Query results are provided in real time, and are executed during a specifiable time frame. It is possible to monitor the traffic between specific endpoints, a single segment, a set of segments, or even the whole backbone.

BackStreamDB allows backbone traffic to be captured with different levels of granularity, according to the user's need. Allowing the selection of monitoring metrics as a query is specified is flexible and convenient, as new information can be obtained on the fly by just executing a different query. In addition, arbitrary measurements can be obtained without storing any traffic logs, which is important as the amount of traffic in these networks is huge. Furthermore, it is possible to extend the system by implementing new operators that manipulate traffic in real time. New operators are easily incorporated as new modules.

In Reference 2, we focus on the ability of BackStreamDB to process arbitrary queries about the backbone traffic in real time. In the present work, we describe the features for detecting traffic anomalies. A very large number of attacks are reported daily in the Internet, and most of them have an impact on the traffic. In computer networks, traffic anomalies usually cause unpredictable behavior, often having a negative impact on the availability of services and the network itself. Several different types of traffic anomalies have been reported, and they are usually classified according to the way they affect the traffic.[3] The traffic anomaly detection approach we implemented is based on a method originally proposed by Crovella et al.[4] This method employs statistical analysis of the traffic based on entropy and principal component analysis (PCA) and allows the detection of traffic anomalies considering the network as a whole.

Among the significant types of events that lead to traffic anomalies, some of the most important include denial of service (DoS), flash crowds, port scans, and worms. DoS[5] is usually a type of attack in which one or more hosts try to turn a particular host or service unavailable so that the attacked entity cannot process legitimate requests. Flash crowds cause a sudden increase on the number of accesses to a given service. Port scans[6] probe a range of ports of a given host in order to discover vulnerabilities. Worms[7] usually try to explore vulnerabilities by trying to communicate with several hosts using a small set of ports.

Given the variety of events that may cause traffic anomalies, it is hard to attain a perfect solution to detect and classify general anomalies in computer networks, in particular the Internet. Several existing approaches aim at a specific type of anomaly. The objective of this work is to show that BackStreamDB is capable of detecting different types of traffic anomalies in real time. Another objective of this work is to describe the extension of BackStreamDB with new operator modules, and investigate the performance of running complex statistical operators on network traffic. We describe the implementation of both entropy and PCA operators, which are the basis for detecting anomalies. We validate the proposed traffic anomaly detection system using synthetic traffic including three different well-known types of anomalies: DoS, port scans, and worms. The system proved to effectively detect all those types of anomalous traffic.

The rest of this paper is organized as follows. In the next section, we present an overview of related work. Next, in Section 3, the BackStreamDB monitoring system is described. The traffic anomaly detection operators are described in Section 4. Experimental results are reported in Section 5. Finally, Section 6 concludes the work.

2 | RELATED WORK

SPEs were first proposed motivated by the requirements of a new class of applications, characterized by the continuous generation of data. They were designed to provide the same functionality as DBMSs, but applied to continuous data flows or streams. The main feature of these systems is their ability to provide results in real time, without requiring data to be locally stored. This is particularly useful for network monitoring. The main difference between DBMSs and SPEs is related to how data and queries are handled.[1] DBMSs work with static data and dynamic queries, while SPEs work with dynamic data and static queries. That is, traditional databases apply different queries on the same set of data; on the other hand, SPEs apply the same queries on streaming (dynamic) data.

Several SPEs have been proposed. Borealis[8] is a second generation distributed SPE, which extends and modifies some of the functionalities of the first generation centralized SPE Aurora[9] and multinode SPE Medusa.[10] Other prototypes have been developed in the context of TelegraphCQ[11] and STREAM[12] projects. Gigascope[13] is a system, which uses an SPE tailored for high speed network monitoring. Although reported results are promising, Gigascope is a proprietary (AT&T's) commercial product.

Several applications have been developed on top of SPEs. The authors of Reference 14 describe a case study using TelegraphCQ SPE.[11] This case study involved a functionality analysis to determine whether the SPE can be used to provide the same metrics of T-RAT, a tool developed for analyzing TCP packet traces. Other case studies include the use of Borealis SPE in a multiuser game,[15] and in a sensor network.[16] MaD-WiSe[17] is also a distributed monitoring system for managing data streams generated by wireless sensor networks. These previous works either implement a specific application, or compare their functionality and performance with other tools. In contrast, BackStreamDB[2] is a general purpose and flexible network management system that has been developed on top of the Borealis SPE. Some of the main features of BackStreamDB include the following: it allows data gathering from multiple data sources and features distributed processing at multiple nodes. It is also based on an architecture with separate modules for data acquisition and query result treatment, and it is able to process data in Netflow format, considering the whole backbone. Details on BackStreamDB are presented in Section 3.

The other field of which we describe related work is traffic anomaly detection.[18] Anomaly detection usually consists of two phases: the learning phase and a testing phase. In the first phase, a profile of normal traffic is defined, in the second phase, the learning profile is applied to new monitored flows. The central premise of anomaly detection is that intrusive activities are a subset of anomalous activities.

Statistical methods for anomaly detection observe traffic features and generate profiles to represent the behavior. The profile typically includes measures such as how frequently the feature is observed (including the distributions observed). The profile can also include simple metrics such as CPU usage. In general, two profiles are stored for each object: the current profile and a history profile. As events are processed in the network, the intrusion detection system updates the current profile and periodically compares the current profile with the history profile. If the evaluation results show a difference for some feature greater than a certain threshold, the system generates an alert. An advantage of statistical models is that they do not need prior knowledge of security flaws and/or attacks. As a result, such systems are capable of detecting new attacks, without interference. But there are also disadvantages. Attackers with some experience can train the system to accept abnormal behavior as being normal.

In the area of anomaly detection, the authors of Reference 3 propose heuristics based on rules to distinguish specific types of anomalies in samples of the volume of traffic, but no evaluation with real data was reported. In Reference 4, the authors suggest that the reason for the limited success of these two attempts to detect anomalies, is due to the fact they are based on metrics that rely on traffic volume, and then they argue that volume in itself does not provide enough information to distinguish several anomalies. They show that anomalies can be classified into distinct categories in a systematic way.

Although it is possible to say that machine learning has the same objective of the statistical methods, machine learning is more, as it allows the system to change its behavior, so that it can improve its performance based on previous results. Thus, machine learning has been increasingly applied to anomaly detection with the purpose of building self-configuring systems.

Network traffic monitoring and analysis is considered from the point of view of big data analytics in Reference 19. The Big-DAMA framework is presented, which can store and process both structured and unstructured data from heterogeneous sources, with both stream and batch processing capabilities. The authors claim that as network data is multidimensional, machine learning can effectively allow the detection and classification of network attacks and anomalies. Big-DAMA implements several algorithms for anomaly detection and other security-related tasks using supervised and unsupervised machine learning (ML) models. Experimental results obtained from running the system with network measurements collected at the WIDE backbone network allow the authors to conclude that Big-DAMA is effective for detecting a set of well-known attacks.

Yet another work that applies machine learning to network traffic processing is Reference 20, in which the authors employ unsupervised machine learning algorithms coupled with real-time streaming and analytics to detect and mitigate distributed denial of service (DDoS) attacks. Experimental results are presented for real DDoS traces.

In Reference 21, a framework with expert system functionality, which also aims at detecting and mitigating DDoS attacks. Traffic of a service is first aggregated based on common IP address prefixes, and attacks are detected as the aggregated traffic deviates from what is considered to be regular. Upon an attack detection, traffic from suspicious sources is discarded. The elastic and parallel-distributed SPE StreamCloud was employed to characterize and detect anomalies in real time. An empirical evaluation shows the effectiveness of the system.

A related strategy is presented in Reference 22 that employs rule mining to detect anomalous flows in a large set of flows. The strategy actually uses meta-data from histogram-based detectors to identify suspicious flows, and then apply association rule mining to find and summarize anomalous flows. Results are presented which were executed with data from a backbone network, and show that the strategy effectively finds the anomalous flows, generating a small number of false positives.

Another strategy that relies on artificial intelligence is Reference 23, which proposes a virtualized network function (VNF) that implements a distributed stream processing technique for real-time threat detection. The proposed VNF presents auto-scaling properties and is empirically shown to be able to process up to 5 million messages per second. Another work that uses NFV technology to implement network monitoring architecture is presented in Reference 24. A set of monitoring agents is distributed across the network implemented as VNFs. The management plane enables users to process, analyze and visualize customized network monitoring data. An empirical evaluation shows that the proposed is able to detect traffic anomalies in real time.

TABLE 1 Summary of related work

Reference	Summary
[14]	Employs the TelegraphCQ SPE for the analysis of TCP flows
[16,17]	Employs the Borealis SPE on sensor networks
[3]	Traffic anomaly detection: heuristics
[25]	Traffic anomaly detection: analysis of traffic volume based on origin–destination flows
[4]	Traffic anomaly detection: network as a whole, method based on entropy and PCA
[19,20]	Traffic anomaly detection based on machine learning
[21]	Traffic anomaly detection: expert system to detect DDoS attacks
[22]	Traffic anomaly detection: rule mining
[23,24]	Traffic anomaly detection using NFV technology
[26]	Employs SPE for intrusion detection
This work	BackStreamDB detects diverse traffic anomalies with entropy and PCA operators

Several anomaly detection efforts have focused on traffic that passes on a single link. However, in Reference 4, the authors propose to detect anomalies in the network as a whole, and the work described in Reference 25, which examines the traffic volume in terms origin-destination flows.

Intrusion detection is related to anomaly detection, but intrusion detection methods are more effective at the network edge, where it is feasible to collect and analyze packets in depth. The system presented in Reference 26 is focused on real time intrusion detection. The system employs a complex event processing (CEP) engine, which runs on an auto-scaling SPE. The idea is to allow self-adaptation to new attacks and to be able to optimize CEP rules, which is done using particle swarm optimization and bisection algorithms. Experimental results confirm that the system is effective in terms of intrusion detection and efficient given the auto-scaling features.

In Table 1, we give a synthesis of all related work. SPE's have been applied to sensor networks,[16,17] the analysis of TCP flows,[14] network intrusion detection.[26] Multiple, diverse strategies for traffic anomaly detection have been proposed: machine learning,[19,20] expert systems,[21] rule mining,[22] NFV technology,[23,24] besides the strategy implemented in the present work which is the usage of entropy and PCA to detect traffic anomalies by monitoring the network traffic as a whole.[4] It is possible to say that the major contribution of the present work is to employ an SPE extended with the operators to compute traffic entropy and PCA of the network as a whole in real time. We show that the strategy is not only effective and efficient, but also that it is general enough to detect diverse types of attacks—DoS, port scan, worm.

3 | BACKSTREAMDB: A SPE-BASED TRAFFIC MONITORING SYSTEM

BackStreamDB is a distributed monitoring system that allows a network administrator to issue arbitrary queries to obtain network traffic information from a multi-AS backbone. Different granularity is permitted as monitored objects may range from individual segments to the backbone as a whole. Data is obtained from multiple flow data sources that are geographically distributed across the network, and traffic information is obtained and processed in a distributed fashion in real time. This strategy is scalable, as it is possible to accommodate increasingly larger traffic loads by changing the system configuration to distribute data to other existing nodes in the network.

BackStreamDB has been implemented on the Borealis SPE.[8] Borealis was chosen because of its distributed nature, which enables a set of SPEs to be deployed across the network. Each Borealis SPE node receives as input a data stream, and processes its records continuously. In this work, the input stream is provided by BackStreamDB, which collects NetFlow* data. Queries on these streams may involve several operations, such as filtering, aggregation, and correlation. Operations can be pipelined in a way that the output of one is forward as the input to the next until the final result is produced. An example of a query is shown in Figure 1. This example determines the number of octets sent to each IP address to a port number smaller than 1024. In the diagram, each box corresponds to an operation. First, the union operator is applied to a set of input streams in order to generate a single stream as input to the filter operator. The output of the second operator is composed of the records with destination port (dstPort) smaller than 1024, which are then grouped by destination IP

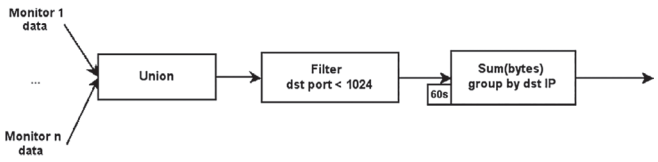


FIGURE 1 Example of a query diagram

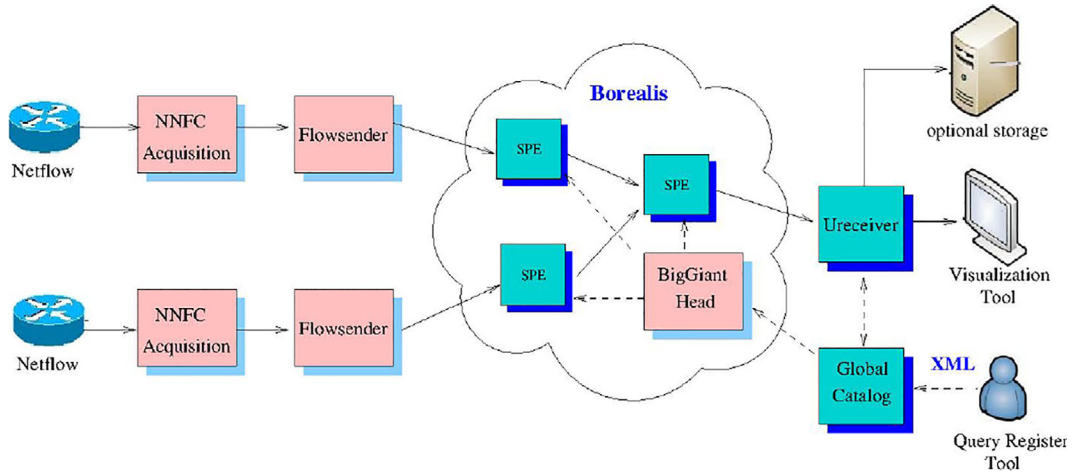


FIGURE 2 BackStreamDB architecture

(dstIP). For each group an output is generated every 60 seconds with the sum of octets. The input for the Borealis SPE consists of XML files with the query specification and the structure of the input and output records.

Borealis has a set of predefined operators, such as union, filter and group by as illustrated in Figure 1. However, an important feature is that new operators can be created and be used for expressing queries combining new and predefined operators seamlessly. This feature allowed us to create two new operators, entropy and PCA for detecting network anomalies, as described in Section 4.

The overall architecture of the BackStreamDB is shown in Figure 2. Borealis SPE nodes are deployed for processing queries through a component called BigGiantHead. The BigGiantHead is responsible for continuously listening to query invocation requests and for sending control data to SPE nodes, which consists of assignment tasks and flow information. BackStreamDB has four other components: acquisition module, flowsender, *universal receiver* (ureceiver), and global catalog, that are described below. Acquisition modules are in charge of receiving data flow, while flowsender is responsible for converting them to Borealis conformant format and forwarding the flow to one or more SPE nodes for query processing. Currently, BackStreamDB processes Netflow data. The acquisition module obtains data using the *New Netflow Collector* (*NNFC*).[†] NNFC is a tool for capturing and storing Netflow data sent by a router. A NNFC plugin was developed to allow communication using IPC (internet process communication) with flowsender. In short, the acquisition module and flowsender are responsible for the interface between data sources and SPEs. A ureceiver (universal receiver) is responsible for the interface between SPEs and visualization tools. Query results can also be stored for historical purposes.

In the standard Borealis distribution, it is necessary to develop a new receiver application for each distinct query result format. This is because Borealis outputs query results in binary format, and the receiver is responsible for decoding these values into typed output fields. We have changed this approach by coding the ureceiver with the capability to *infer* the query output format based on the query definition. As a result, the system does not have to be recompiled when new query results are defined, as in the standard Borealis distribution. When invoked, ureceiver waits for a connection from a Borealis SPE, and when new query results arrive, they are output in either text or graphical form by a visualization tool. That are being processed by SPEs are stored at the fourth component of the system: the global catalog. For each query, the catalog maintains the query definition and information specifying the SPE nodes, which are executing the query. We employ the same language adopted by Borealis, in which queries are expressed in an XML document, containing input, output, and query definitions. BackStreamDB's query register tool reads the XML document, stores the information in the global catalog, and communicates with BigGiantHead through the network in order to deploy the execution of queries on different nodes.

Query results can be either accessed in real time by a network administrator with visualization tools, or can also be stored if required. Since the system does not log flow data, but only query results that have been individually specified by the administrator, BackStreamDB can drastically reduce the storage cost. Query definitions are fed to the system by a query register tool in a high-level query language, which makes queries easy to maintain and reuse.

There is a large spectrum of possible system configurations, ranging from a fully distributed system in which each module is assigned to a distinct node, to a centralized system, in which a single node runs all modules. Ideally, when data sources are geographically distributed, both an acquisition module, flowsender, and an SPE node should be deployed close to the source. In this way, the source data can be locally filtered by the SPE node, reducing the volume of data to be transmitted among SPE nodes and the ureceiver.

4 | USING ENTROPY AND PCA TO DETECT ANOMALIES

In this section, we describe the implementation of the entropy and PCA operators in BackStreamDB for the detection of traffic anomalies. Before describing the implementation, we give a brief overview of entropy and PCA.

4.1 | Entropy and PCA

Entropy is a measure of the uncertainty of a random variable.[27] In the particular case of traffic monitoring, entropy captures in a single value the probabilities of changes occurring to certain traffic features. By computing the entropy, one extracts the properties of traffic feature distributions so that it is possible to then detect and classify anomalies.

Let E_i be an event and p_i the probability of the occurrence of this event. Consider n events $E_1 \dots E_n$ with probabilities $p_1 \dots p_n$ the summation of these probabilities is equal to 1. The rationale behind the concept of entropy is the fact that the occurrence of events with low probabilities result in more information for the observer. Shannon[28] proposed a logarithmic function to express this concept which he called $h(p_i)$ which decreases as p_i increases. $h(p_i)$ is defined as follows:

$$h(p_i) = \log_2 \left(\frac{1}{p_i} \right) \tag{1}$$

$h(p_i)$ varies from ∞ to 0 as p_i varies from 0 to 1. This function reflects the idea that the lower the probability of an event, the greater is the amount of information obtained when the event occurs.

The entropy is computed as the weighted average of the N values of $h(p_i)$ each weighted by the probability of its occurrence:

$$\left(\frac{1}{p_i} \right) H = \sum_{i=1}^{n} p_i h(p_i) \tag{2}$$

PCA[29] is a statistical technique that receives as input high-dimensional data, and using the dependencies between the variables represents the same data in more tractable format using less dimensions. PCA is considered to be a simple and robust way to reduce the number of dimensions to make the data simpler to work with.

PCA transforms a random vector $x \in R^m$ into another vector $y \in R^n$, $n \leq m$, projecting x on the n orthogonal directions with greater variance—these are called the principal components. In general, most data variance can be explained by a reduced number of components, it is thus possible to discard the remaining components without losing too much information.

4.2 | Implementation of traffic anomaly detection in BackStreamDB

We implemented the entropy operator and the PCA operator to allow BackStreamDB to detect anomalies. The entropy operator computes the entropy of four traffic features for the input stream during a certain time frame. The four features considered are: source and destination addresses (srcIP, dstIP) and ports (srcPort, dstPort). The entropy operator uses the feature data to generate a tuple which consists of the four entropy values and the corresponding timestamp, which

FIGURE 3 Data flow between the operators

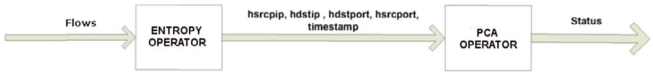
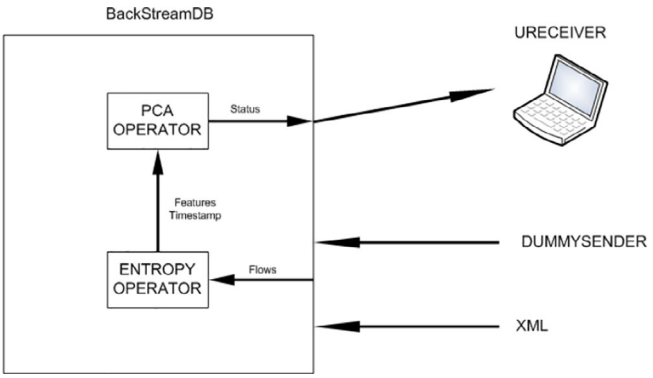


FIGURE 4 Experimental environment



is generated at the end of the time interval. These tuples are then sent as input to the PCA operator that infers whether there are traffic anomalies in the observed traffic, as described below.

The data flow is shown in Figure 3. The PCA operator creates a matrix based on the four traffic features (srcIp, destIP, srcPort, and dstPort). Each matrix row consists of the tuples received from the entropy operator, computed for the corresponding feature for an OD pair during a certain time interval, in our case 5 minutes. Four traffic matrices are built each for a traffic feature. However, it is important to be able to detect changes across the four traffic features, and also taking into account the whole set of OD flows, which can together represent an attack. In order to do this, a method based on multivariate statistics is proposed,[25] that "unfolds" the four traffic feature matrices into a single large matrix, on which anomalies are detected as described next.

The entropy and PCA operators were implemented taking into account a single OD pair. In this way, there are four vectors representing the computed entropy for each time interval, one vector for each feature.

The PCA operator normalizes the columns of the matrix by dividing each element by the sum of all elements of the corresponding column. In this way, the sum of all elements of a column is equal to 1. The PCA operator then computes the eigenvectors—which correspond to the principal components—and the eigenvalues, which define their magnitude. Then the components that represents what we call normal data, which covers 85% of the variance, and the component that corresponds to the remaining residual data. Let $\widehat{y}$ correspond to the normal data component and $\widetilde{y}$ represent the residual data component. A traffic anomaly can be inferred using a function on the size of $\widetilde{y}$, which is given by $\|\widetilde{y}\|^2$. Unusually large values of $\|\widetilde{y}\|^2$ that are greater than a threshold δ_α^2 correspond to anomalies. The threshold δ_α^2 is computed as a function of α: the desired false alarm rate.[25]

5 | EXPERIMENTAL RESULTS

In this section, we present experimental results obtained with the use of BackStreamDB for network wide traffic anomaly detection. Two operators were implemented: the first for computing the traffic entropy and the other for PCA. Two sets of experiments were implemented. The goal of the first set is to determine the effectiveness of the tool for detecting anomalies, and the goal of the second set is to evaluate the efficiency. We have deployed BackStreamDB with a single SPE node executing the query as shown in Figure 4, which applies the entropy and PCA operators on all traffic records of the input stream. The figure shows a BackStreamDB processing node, the XML file that is given as input, which specifies both the query and the operators it requires—in this case, the only operators required are the entropy and PCA operators. In the experiments, we employed a tool called `dummysender` for generating synthetic traffic. This approach makes it easier to inject anomalies and determine whether the tool is able to correctly identify those anomalies. We must note however that in a real setting the input flow can be originated from any network tool such as NetFlow, sFlow, or IPFIX. The results reported in the following sections were collected with all BackStreamDB modules and the `dummysender` running on a virtual machine with 512 MB of RAM and a 2.66 GHz Intel Xeon processor with Debian Linux kernel version 2.6.26.

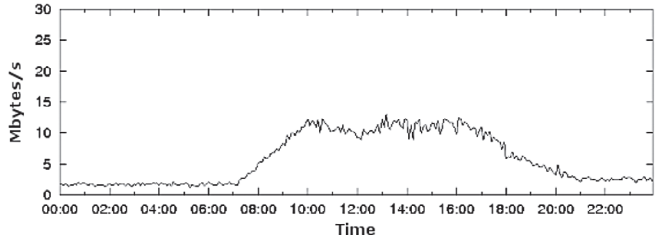


FIGURE 5 Traffic volume in megabytes per second

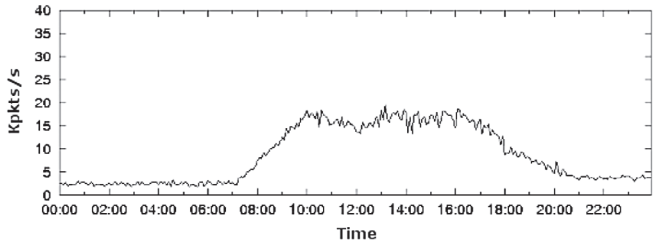


FIGURE 6 Traffic volume in packets per second

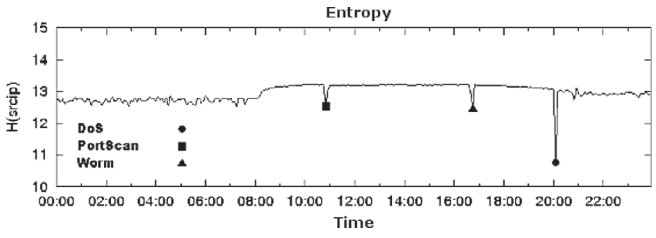


FIGURE 7 Entropy computed for the source address

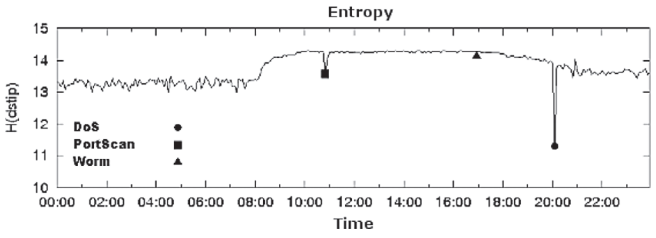


FIGURE 8 Entropy computed for the destination IP address

5.1 | Experiments

In order to validate the ability of the tool to detect traffic anomalies, synthetic traffic was generated simulating 24 hours of network traffic with the insertion of three well-known anomalies: port scan, worm, and DoS. We used three different values for the false positive rate α: $\alpha = 0.999$ (actually meaning one false positive per 1000) $\alpha = 0.995$ (5 false positives per thousand) $\alpha = 0.70$ (30% of results are false positives). For each value of α, we evaluated the effectiveness of the detection of anomalies as well as the amount of false positives generated.

The results are shown in Figures 5 and 6. Figure 5 represents the volume of traffic in megabytes per second, while Figure 6 reports the volume in packets per second. Figures 7-10 represent the entropy computed for the traffic generated for the period of one day. Figure 11 represents the projection of the vector $\|\widetilde{y}\|^2$ as given by the PCA method using the thresholds based on the three different values of α.

As can be seen in Figures 7-10, some anomalies have a deep influence on the corresponding entropy values. The most pronounced values occurred because the anomalies changed the distribution of source or destination IP addresses, source or destination ports. The worm, for example, produces expressive changes on the entropy values computed for the source IP address and destination port, but show no significant change for the entropy of the destination IP address.

As can be seen in Figure 11, there are four outstanding peaks in the generated graphic. The first peak represents a false positive detected only for $\alpha = 0.70$ (a 30% rate of false positives). The second peak represents a port scan attack detected for all thresholds, and the third peak represents a worm, which was not detected with the most restrictive threshold $\alpha = 0.999$ (which allows 1 in a thousand false positive rate), and the fourth peak was detected with all the thresholds. Based on

FIGURE 9 Entropy computed for the source port

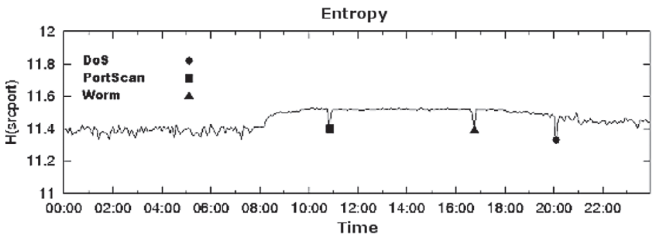


FIGURE 10 Entropy computed for the destination port

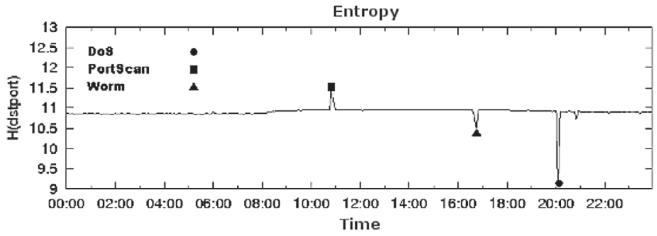
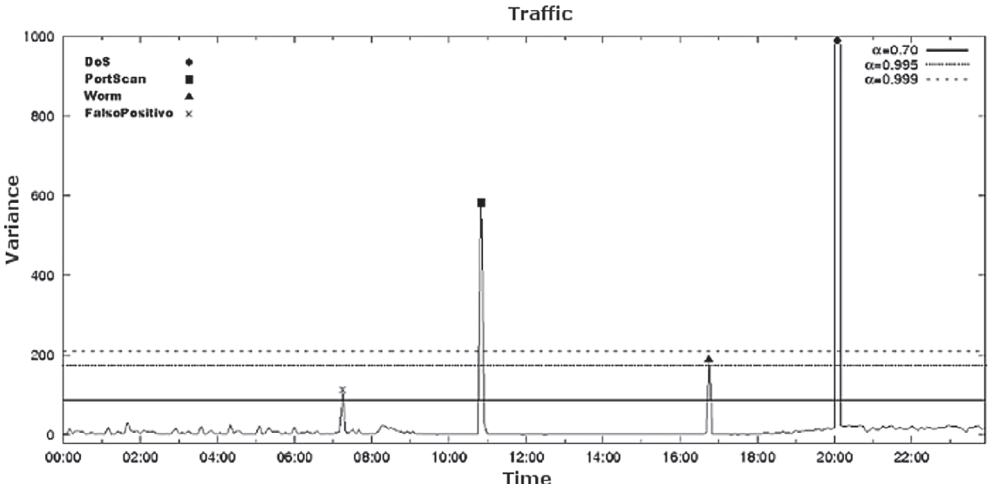


FIGURE 11 PCA application: the $\|\tilde{y}\|^2$ vector



these results, it can be concluded that a low value of α (0.70) is very sensitive to small changes on the traffic features, but it can also mistakenly interpret normal flow as anomalous. On the other hand, a high value of α (0.999) presents the results when detecting anomalies that cause significant changes on the traffic features; although it hardly commits mistakes by considering normal flow to be anomalous, it does fail to detect certain anomalies. Using an intermediate value for α (0.995), it was possible to adjust the anomaly detection and false positive rates, so that all injected anomalies were detected and no false positives were generated.

5.2 | Performance evaluation

In this set of experiments, we evaluated the performance of the tool. We first measured CPU and memory usage as the volume of traffic is varied. The same time window of 5 minutes was employed, but the number of flows per second was increased at subsequent time windows. In this case, it was expected that the entropy operator would reach up to 100% CPU usage and then packets would start to be dropped. However, what happened and is shown in Figures 12 and 13 is that with an input of 4000 flows per second the CPU usage was only up to 50%. In terms of memory consumption, we increased the number of flows until they started to be dropped as the 512 MB got full at nearly 4000 flows per second. Note that if we decrease the window size and keep the flow rate constant, the number of flows that have to be kept in main memory will also decrease.

Since traffic volume can be very high, and the PCA method is computationally intensive, we measured the time interval from the instant the PCA operator received the input flow to the instant the output is produced. The results are shown

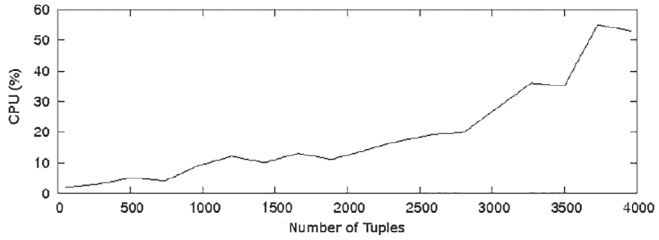


FIGURE 12 CPU usage

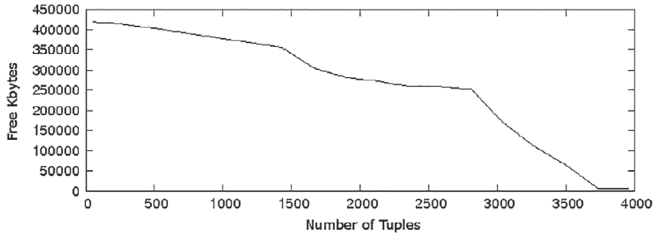


FIGURE 13 Memory usage

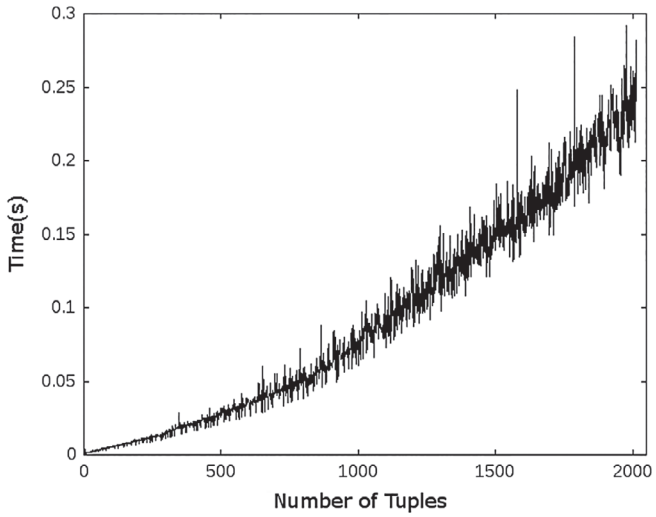


FIGURE 14 Performance of the PCA operator

in Figure 14. Data was generated each 5 minutes for a whole week. As shown in the figure, for 2016 tuples the PCA operator still required less than 0.3 seconds to complete its task, which confirms the feasibility of BackStreamDB to detect anomalies with complex statistical methods.

6 | CONCLUSIONS

In this work, we presented BackStreamDB a general-purpose backbone monitoring system based on a SPE that can detect traffic anomalies in real time. BackStreamDB has been deployed on the Brazilian National Academic Network (RNP). Operators were implemented for computing both the traffic entropy and PCA, which characterize the flow based on four traffic features: source address, destination address, source port, and destination port. The anomaly detection module uses the entropy of these features and the normalization applied by the PCA operator to identify the occurrence of anomalies. Experimental results are presented which were obtained with synthetic traffic. In order to validate the detector we have injected three different types of anomalies. Results show that our strategy has effectively recognized the anomalies. Future work includes investigating system under different types of attacks, as well as comparing the performance of the statistical methods with operators based on machine learning for real-time anomaly detection.

ACKNOWLEDGMENTS

We would like to thank Fabricio Dyck and Emanuel Paul Filho for their work on the implementation of the BackStreamDB modules for traffic anomaly detection. We would also like to thank the Brazilian National Research and Education Network (RNP—www.rnp.br) for the project that supported the early development of BackStreamDB: the Working Group GT-BackStreamDB, Phases 1 and 2 (2008-2010). This work was partially supported by CNPq—The Brazilian Research Council grant 311451/2016-0.

ENDNOTES

*http://www.cisco.com/web/go/netflow
†http://sourceforge.net/projects/nnfc

ORCID

Elias P. Duarte Jr https://orcid.org/0000-0002-8916-3302

REFERENCES

1. Garofalakis M, Gehrke J, Rastogi R. *Data Stream Management: Processing High-Speed Data Streams*. Berlin: Springer; 2016.
2. Lyra C, Hara CS, Duarte EP Jr. BackStreamDB: a distributed system for backbone traffic monitoring providing arbitrary measurements in real-time. *Passive and Active Measurement*. Berlin: Springer; 2012:42-52.
3. Kim M-S, Kong H-J, Hong S-C, Chung S-H, Hong JW. A flow-based method for abnormal network traffic detection. *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*. Vol 1. Picataway, NJ: IEEE; 2004:599-612.
4. Lakhina A, Crovella M, Diot C. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Comput Commun Rev*. 2005;35(4):217-228.
5. Mahjabin T, Xiao Y, Sun G, Jiang W. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *Int J Distrib Sens Netw*. 2017;13(12):1–33.
6. Bhuyan MH, Bhattacharyya D, Kalita J. Surveying port scans and their detection methodologies. *Comput J*. 2011;54(10):1565-1581.
7. Qing S, Wen W. A survey and trends on internet worms. *Comput Secur*. 2005;24(4):334-346.
8. Abadi DJ, Ahmad Y, Balazinska M, et al. The design of the borealis stream processing engine. *Proc. of the Conf. on Innovative Data Systems Research*. Asilomar, CA: CIDR; 2005:277-289.
9. Carney D, Cetintemel U, Cherniack M, et al. Monitoring streams: a new class of data management applications. *Proc. of the Int. Conf. on Very Large DataBases*. Hong Kong, China: VLDB; 2002:215-226.
10. Cherniack M, Balakrishnan H, Balazinska M, et al. Scalable distributed stream processing. *Proc. of the Conf. on Innovative Data Systems Research*. Asilomar, CA: CIDR; 2003.
11. Chandrasekaran S, Cooper O, Deshpande A, et al. TelegraphCQ: continuous dataflow processing for an uncertain world. *Proc. of the Conf. on Innovative Data Systems Research*. Asilomar, CA: CIDR; 2003.
12. Arasu A, Babcock B, Babu S, et al. STREAM: the stanford data stream management system. *IEEE Data Eng Bull*. 2003;26(1):19-26.
13. Cranor C, Johnson T, Spataschek O. Gigascope: a stream database for network applications. *Proc. of the ACM SIGMOD Int. Conf. on Management of Data Conference*. Asilomar, CA: CIDR; 2003:647-651.
14. Plagemann T, Goebel V, Bergamini A, Tolu G, Urvoy-Keller G, Biersack EW. Using data stream management systems for traffic analysis—a case study. *Proc. of the Int. Workshop on Passive and Active Network Measurement*. Berlin: Springer; 2004:215-226.
15. Ahmad Y, Berg B, Cetintemel U, et al. Distributed operation in the borealis stream processing engine. *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*. New York, NY: ACM; 2005:882-884.
16. Abadi DJ, Lindner W, Madden S, Schuler J. An integration framework for sensor networks and data stream management systems. *Proceedings of the Thirtieth International Conference on Very Large Data Bases*. Vol 30. Toronto, Canada: VLDB Endowment; 2004:1361-1364.
17. Amato G, Chessa S, Vairo C. MaD-WiSe: a distributed stream management system for wireless sensor networks. *Softw Pract Exp*. 2010;40(5):431-451.
18. Fernandes G, Rodrigues JJPC, Carvalho LF, Al-Muhtadi JF, Proença ML. A comprehensive survey on network anomaly detection. *Telecommun Syst*. 2019;70(3):447-489.
19. Casas P, Soro F, Vanerio J, Settanni G, D'Alconzo A. Network security and anomaly detection with Big-DAMA, a big data analytics framework. *The 6th IEEE International Conference on Cloud Networking (CloudNet)*. Piscataway, NJ: IEEE; 2017:1-7.
20. Villalobos JJ, Rodero I, Parashar M. An unsupervised approach for online detection and mitigation of high-rate DDoS attacks based on an in-memory distributed graph using streaming data and analytics. *The 4th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*. Piscataway, NJ: IEEE; 2017:103-112.
21. Gulisano V, Callau-Zori M, Fu Z, Jimenez-Peris R, Papatriantafilou M, Patino-Martinez M. STONE: a streaming DDoS defense framework. *Expert Syst Appl*. 2015;42(24):9620-9633.
22. Brauckhoff D, Dimitropoulos X, Wagner A, Salamatian K. Anomaly extraction in backbone networks using association rules. *IEEE/ACM Trans Netw (TON)*. 2012;20(6):1788-1799.
23. Lopez MA, Lobato AGP, Duarte OCMB, Pujolle G. An evaluation of a virtual network function for real-time threat detection using stream processing. *The 4th International Conference on Mobile and Secure Services (MobiSecServ)*. Piscataway, NJ: IEEE; 2018:1-5.

24. Pavlidis A, Sotiropoulos G, Giotis K, Kalogeras D, Maglaris V. NFV-compliant traffic monitoring and anomaly detection based on dispersed vantage points in shared network infrastructures. *The 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. Piscataway, NJ: IEEE; 2018:197-201.
25. Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Comput Commun Rev*. 2004;34(4):219-230.
26. Loganathan G, Samarabandu J, Wang X. Real-time intrusion detection in network traffic using adaptive and auto-scaling stream processor. *2018 IEEE Global Communications Conference (GLOBECOM)*. Piscataway, NJ: IEEE; 2018:1-6.
27. De Luca A, Termini S. A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory. *Inf Control*. 1972;20(4):301-312.
28. Shannon CE. *A Mathematical Theory of Communication. The Bell System Technical Journal*. Vol 27. Murray Hill, NJ: Bell Labs; 1948:379-423.
29. Ait-Sahalia Y, Xiu D. Principal component analysis of high-frequency data. *J Am Stat Assoc*. 2019;114(525):287-303.

How to cite this article: Duarte EP, Hara C, Torres P, Gomes C. BackStreamDB: A stream processing engine for backbone traffic monitoring with anomaly detection. *Security and Privacy*. 2020;3:e106. https://doi.org/10.1002/spy2.106