Building Scalable Distributed Intrusion Detection Systems Based on the MapReduce Framework

Marcelo D. Holtz, Bernardo M. David & Rafael Timóteo de Sousa Júnior

Abstract— **Intrusion detection systems (IDS) are essential components in a secure network environment, allowing for early detection of malicious activities and attacks. By employing information provided by an IDS it is possible to apply appropriate countermeasures and mitigate attacks that would otherwise seriously undermine network security. However, current high volumes of network traffic overwhelm most IDS techniques, requiring new approaches that are able to handle huge quantities of traffic during analysis while still maintaining high throughput. We propose an architecture for distributed Network Intrusion Detection Systems where comprehensive data analysis is executed in a cloud computing environment. Network traffic, operating system logs and general application data are collected from various sensors in different places in the network, comprising networking equipment, servers and user workstations. The data collected from different sources is aggregated, processed and compared using the Map-Reduce framework, analysing event correlations which may indicate intrusion attempts and malicious activities. The proposed architecture is able to efficiently handle large volumes of collected data and consequent high processing loads, seamlessly scaling to enterprise network environments. Also, differently from previous IDS models, it capable of detecting complex attacks through the correlation of information obtained from different sources, identifying patterns which may not be apparent in centralized traffic captures or single host log analysis. Besides an architecture description, we present feasibility results based on experiments performed on real cluster and cloud infrastructure.**

I. Introduction

Intrusion Detection Systems (IDS) are important mechanisms which play a key role in network security and self-defending networks. Such systems perform automatic detection of intrusion attempts and malicious activities in a network through the analysis of traffic captures and collected data in general. Such data is aggregated, analysed and compared to a set of *rules* in order to identify attack *signatures*, which are traffic patterns present in captured traffic or security logs that are generated by specific types of attacks. In the process of identifying attacks and malicious activities, an IDS parses large quantities of data searching for patterns which match the rules stored in its signature database. Such procedure demands high processing power and data storage access velocities in order to be executed efficiently in large networks.

The first issue in designing intrusion detection systems lies in efficient and comprehensive collection of data that comprises activities in different network portions in order to obtain a general view of network activity. An IDS should reach high collection throughput while collecting as much data as possible. Once enough data is gathered, it is necessary to rapidly analyse it and determine whether any attacks or malicious activities are present, which is the main issue that impacts IDS performance. Usually attack detection requires processing collected data through pattern matching algorithms in order to determine whether any of the patterns contained in the signature database are present. Hence, if the quantity of collected data is excessively massive, IDS performance is seriously affected.

It is clear that overall Internet traffic has been growing exponentially over the past few years and it is projected to maintain such growth [1], which is illustrated in Figure 1. Such large traffic volumes generate unforeseen analysis datasets which overwhelm most of current IDS pattern matching approaches, specially considering that most of this approaches are based on centralized collection and processing of data. Centralized non-parallel approaches to IDS data analysis and attack detection are constrained by resource limitations of individual computer systems, which lack the storage and processing resources for handling the necessary quantity of data. Moreover, current systems are based on very limited data collection methods that mainly consider only network traffic data, excluding other important sources such as operating system logs.

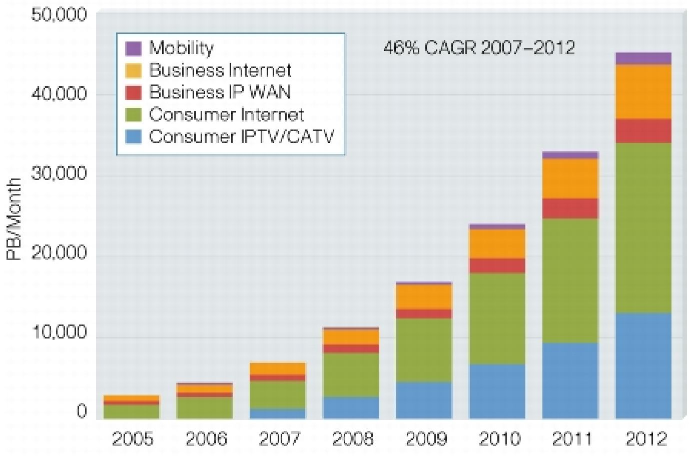

Fig. 1. Internet traffic growth since 2005 and projections for 2012 [1]

Most of current IDS architectures for infra-structured networks in current literature are based on centralized data collection and processing by certain nodes and areas in the network.This approach does not effectively result in a thorough

Manuscrito recebido em 30 de novembro de 2010; revisado em 1 de abril de 2011.

Marcelo D. Holtz (holtz@redes.unb.br), Bernardo M. David (bernardo.david@redes.unb.br) e Rafael Timóteo de Sousa Júnior (desousa@redes.unb.br) pertencem ao Departmento de Engenharia Elétrica, Universidade de Brasilia.

view of localized malicious network activity in scenarios where adversaries connect through different network access media, such as wireless access points, virtual private networks (VPNs) and spare cabled ethernet connections. Being focused on the sole analysis of central traffic capture, the centralized intrusion detection systems are incapable of detecting complex attacks which generate patterns both in network traffic and in application and operating system logs in multiple network nodes. Furthermore, with rapidly growing network activity, classical IDS rule parsing and data analysis mechanisms are overwhelmed by the sheer volume of network traffic and data collected, specially in large enterprise networks. Such systems are not able to efficiently process this volume of data or to scale as the network grows.

A. *Related Works*

In order to address these issues, new parallel network data analysis approaches have been proposed but until now there are few concrete proposals of a fully distributed intrusion detection system architecture with distributed data storage and processing. There are several interesting mobile agent based approaches which transfer data storage and processing to the actual hosts being analysed by setting up mobile agents at each host and dynamically balancing storage and processing loads [2] [3]. Another approach proposed in [4] builds on mobile agents to collect data at different network hosts and perform data mining in order tyo detect attacks and malicious activities, thus obtaining nice scalability and a general view of network activity while adaptively distributing load accross several hosts. However, even though these architectures solve the issues related to centralized data storage and processing, in such a loosely coupled and dynamical scenario, various security issues arise, such as authenticating the information exchanged between agents and individual corruption of agents by malicious applications, such as viruses and trojan horses. Solving these security issues in uncontrolled environments such as large scale networks is not trivial, thus reducing the applicability of such solutions in real world systems.

In the realm of distributed network data analysis methods, the approach of [5] to analysing internet traffic on MapReduce clouds stands as a milestone in designing efficient distributed IDS systems. The results in [5] show that it is possible to efficiently obtain statistical data from very large raw traffic capture datasets. The first approaches to analysing IDS datasets on MapReduce based infrastructures were proposed independently in [6] and [7], and consist in transfering data storage and processing to the cloud in order to obtain better efficiency, scalability and throughput. The approach of [7] differs in that it also proposes an architecture for distributed data collection which provides comprehensive data on overall network activities across different individual systems.

Other interesting approaches are based on fast pattern matching by custom FPGA designs [8] [9], which are able to process several Gigabits of data per second, but still do not solve storage issues. Also, these methods are considerably more expensive and less scalable than software based techniques, since they require purposely built hardware.

B. *Our Contributions*

We propose a novel distributed network intrusion detection system architecture which decentralizes both data collection and processing, thus achieving better scalability, faster data analysis and better event detection probability. The proposed architecture enjoys the following characteristics:

- Distributed data collection from multiple sources (*e.g.* network traffic and operating system security logs) in multiple network areas.
- Scalable storage in a distributed filesystem infrastructure.
- Scalable distributed data processing in cloud environments through the MapReduce framework.
- Implementable from widely deployed open source software tools.

Data collected from various sources located at different places in the network is correlated in order to detect complex attacks which may not be apparent in the analysis of network traffic. The proposed architecture is based on distributed data analysis through the MapReduce framework in a cloud computing environment with a distributed filesystem to rapidly parse collected data. It is potentially capable of detecting potential attack signatures with very high throughput while delivering network management statistics. This architecture scales to analyse the sheer quantity of data collected in today's growing enterprise networks while being able to detect complex malicious activities. In order to prove the feasibility of our approach we show that the MapReduce framework and distributed filesystems are able to achieve the expected data storage and processing performance through simulations conducted using the Hadoop project implementation of MapReduce algorithms. We also analyse the performance of the distributed filesystem HDFS for such an application, showing that it is possible to store and retrieve large quantities of data with the necessary speed .

C. *Organization*

In Section II, we introduce the basic concepts and techniques of classical intrusion detection systems. In Section II-A, we discuss distributed intrusion detection systems, listing the main characteristics, advantages and caveats of such systems. In Section III, we describe the MapReduce framework used for distributed data processing in cloud environments, also commenting on its implementation by the Hadoop project and its various components. In Section IV, we introduce our architecture for cloud based distributed intrusion detection systems and discuss its details. In Section V, we describe the experiments conducted in order to attest the feasibility of our proposed approach and the data obtained from these experiments. Finally, in Section VI, we summarize our results and conclude with promising directions for further research.

II. Intrusion Detection Systems

In this section, we introduce the concept of intrusion detection systems and the several approaches to building such systems. We further classical works and techniques that relate to the system proposed in this paper.

Intrusion detection systems (IDS) automatically monitor events occurring in a computer system or network in order to detect malicious activities or security policy violations. IDSs issue security alerts when an intrusion or suspect activity is detected through the analysis of different aspects of collected data (*e.g.* packet capture files and system logs). Classical intrusion detection systems are based on a set of attack signatures and filtering rules which model the network activity generated by known attacks and intrusion attempts [10], [11]. There are also efforts towards the development of IDSs based on machine learning techniques (such as neural networks) and data mining which automatically identify and incorporate new attack signatures [12].

Intrusion detection systems detect malicious activities through basically two approaches: anomaly detection and signature detection [12], [13].In traffic anomaly detection, first a standard traffic pattern statistical profile is established and then it is compared current traffic in order to detect any deviation from the expected normal behavior. In signature detection (which has been discussed before), network traffic is compared with attack signatures stored in a database in order to detect- specific attacks [14]. Anomaly detection is capable of identifying attacks that were not previously observed but this kind of technique is always subject to a high rate of false positives.

Intrusion detection systems are classified in mainly two groups Network Intrusion Detection Systems (NIDS), which are based on data collected directly from the network, and Host Intrusion Detection Systems (HIDS), which are based on data collected from individual hosts. HIDSs are composed basically by software agents which analyse application and operating system logs, filesystem activities, local databases and other local data sources, reliably identifying local intrusion attempts. Such systems are not affected by switched network environments (which segment traffic flows) and is effective in environments where network packets are encrypted (thwarting usual traffic analysis techniques) [15]. However, they demand high processing power overloading the nodes' resources and may be affected by denial-of-service attacks.

Network intrusion detection systems identify attacks through the analysis of network traffic capturd at the network border, thus containing traffic flowing to and from all internal hosts. This kind of IDS is capable of processing packet captures containing traffic from several nodes with little or no network overload [16]. It is secure against internal and external attacks as it functions invisibly in the network, simply capturing packets in promiscuous mode. In face of the growing volume of network traffic and high transmission rates, software based NIDSs present performance issues, not being able analyse all the captured packets rapidly enough. Some hardware based NIDSs offer the necessary analysis throughput [8] but the cost of such systems is too high in relation to software based alternatives. The generic structure of a network intrusion detection system is illustrated in Figure 2.

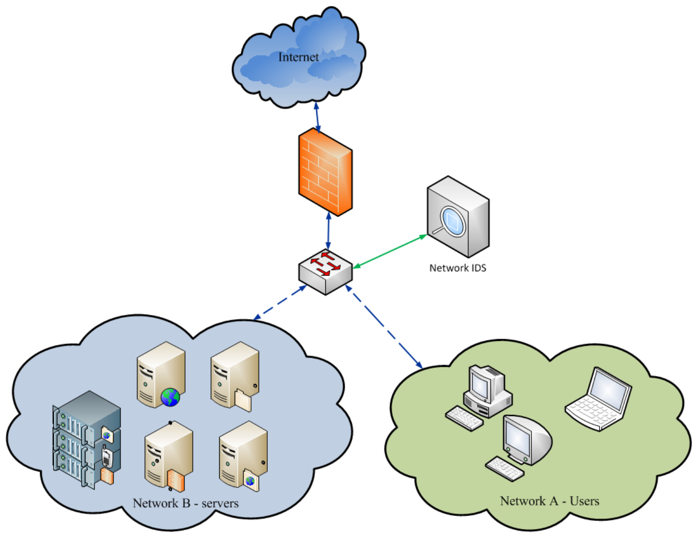

Fig. 2. The Structure of a Network Intrusion Detection System

A. *Distributed Data Collection and Correlation*

Current networking environments are becoming increasingly heterogeneous and complex, incorporating several access media and network access which contribute to the decentralization and segregation of network traffic and activities. Moreover, in large networks, different areas are usually segmented from each other for security and organization reasons. This separation occurs in different layers, depending on the method utilized (*e.g.* IEEE 802.1q VLANS or Routing) and offering different segregation levels . The heterogeneous and decentralized nature of current networks results in significant portions of traffic and activities being restricted only to certain areas of the network (specially when different access media is used) and never reaching central or border nodes.

In face of this situation, classic network intrusion detection systems do not efficiently identify attacks in large heterogeneous networks due to their inherent centralized nature. While NIDSs are commonly placed in central or border regions of the network (*e.g.* next to gateways, servers or firewalls), malicious activities which occur inside the network and are restricted to a specific region may not generate traffic reaching the NIDS nodes. In such scenarios, it is unlikely that the central NIDS nodes would detect all insider and outsider attacks and intrusion attempts directed to the network. Furthermore, large scale networks frequently provide several access points and gateways, requiring NIDS systems to be deployed in each of them in order to guarantee thorough monitoring.

In order to effectively capture representative data of network activities it is necessary to collect and analyse IDS data in a distributed manner, ensuring that malicious activities occurring in different layers of isolated network regions are detected. The concept of distributed intrusion detection systems (DIDS) was first proposed in [17], where a system composed of distributed sensors and a centralized analysis system is introduced. This DIDS basically consists in collecting data from heterogeneous sources located in different areas of the network, aggregating this data in a centralizer host referred to as *director* and finally analysing it locally using standard intrusion detection algorithms. It is important to notice that the proposed DIDS collects not only traffic but also audit traces from different nodes in the network, making it possible to identify subtle and complex attacks through data fusion and event correlation techniques. The general structure of a distributed intrusion detection system is depicted in Figure 3.

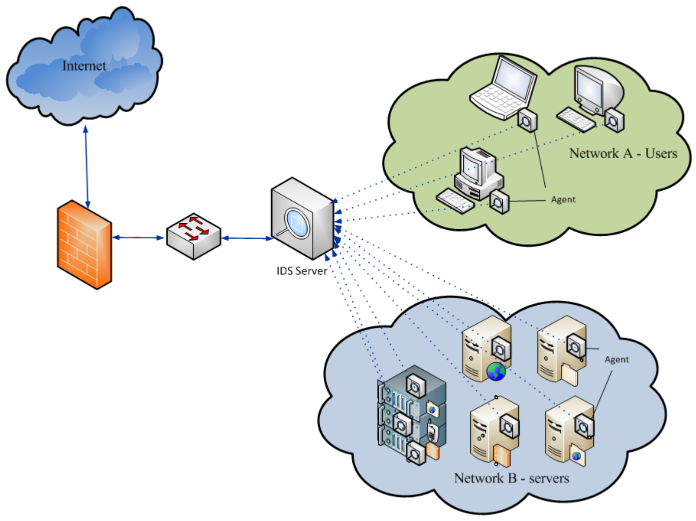

Fig. 3. The Structure of a Distributed Intrusion Detection System

While such system provides a nice solution for moderately large network setups with moderate traffic throughputs, in high throughput settings its performance can be seriously affected. However, it provides the basic structure that an efficient distributed intrusion detection system should follow, only failing in the centralized storage and processing of information. A similar approach is presented in [18], where classic DIDS techniques are coupled with cooperative intrusion detecting agents that analyse the strategy utilized by attackers. Once again, this approach solves the distributed data collection issues but does not address issues regarding data storage and processing in traffic intensive environments.

Recent research on attacks detection in distributed honeypots and honeynets indicate that it is feasible to implement distributed data collection architectures spanning a large number of heterogeneous nodes located in different networks [19]. An intelligent distributed intrusion detection system based on honeynets for data collection and mobile agents for distributed data processing was proposed in [20]. This DIDS is capable of processing a large quantity of logs through workload distribution but attack detection is restricted to the honeynet area. Even though these experiments were conducted on honeypot infrastructures, they provide nice evidence regarding the feasibility of building such distributed network data collection and analysis systems for large scale network environments.

III. Cloud Computing and The Map-Reduce Framework

In this section we discuss the basic concepts of cloud computing and the MapReduce framework, introducing the architecture of the MapReduce implementation provided by the Hadoop project.

The constant and rapid growth in the volume of data and communications in current networks and information systems raises the need for efficient scalable data storage and processing techniques. In order to address this issue, a new paradigm commonly called *cloud computing* was introduced. The main characteristic of this paradigm is the decentralization of data processing and storage through clustered environments that seamlessly scale to fit the constantly increasing demand, offering high performance and achieving efficient response times.

Usually a cloud computing environment is composed by a data processing cluster coupled with a storage area network to provide easily scalable resources. The users and applications access this environment seamlessly and transparently through standard API frontends, eliminating the need for knowledge of the specific network infrastructure and underlying services. Such architectures enable fast development and provisioning of large scale applications that handle a high number of requests or large quantities of data. Applications running on a cloud environment are able to scale transparently and automatically by simply adding more cloud resources, which are handled by the corresponding API. The overall structure of a generic cloud computing environment is illustrated in Figure 4.

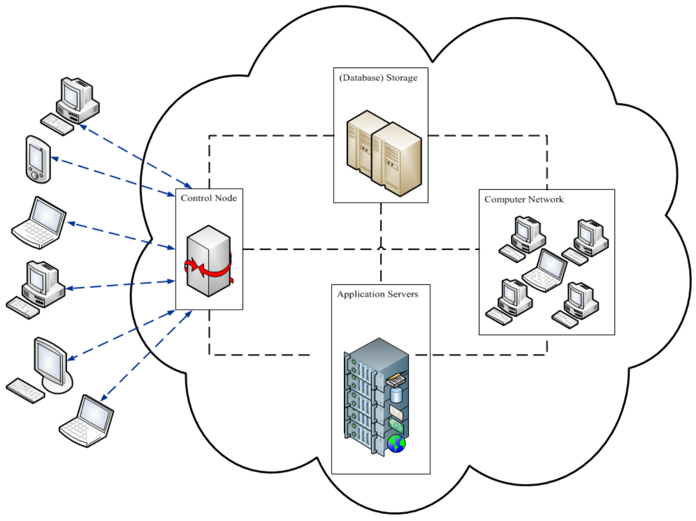

Fig. 4. The Structure of a Cloud Computing Environment

A cloud application can be easily developed by utilizing the proper frontend API calls to perform the desired actions on the cloud environment (*e.g.* store and retrieve files or sort data). After an application is functional, it can be transparently scaled by adding more cloud storage or processing resources. Hence, cloud applications can potentially scale to handle any quantity of data. These characteristics make such platforms ideal for constructing efficient and scalable distributed intrusion detection data analysis systems.

Among the several approaches to cloud computing based solutions there is the MapReduce framework [21], which was first introduced by Google and can be employed for scalable large dataset processing. This framework comprises both data storage solutions (distributed filesystems and databases), management tools and a full distributed data processing environment (including support for different programming languages). Different types of processing intensive applications may take advantage of this framework to perform common tasks over large volumes of data. For example, this framework has already been successfully applied by large companies to perform the following tasks: search engine indexing, statistical data analysis, business intelligence and document indexing.

The main concept of Map and Reduce was introduced in classical functional programming languages (*e.g.* LISP) and consists in splitting computing tasks over large inputs into several steps whose output is then combined into the final output. A MapReduce application takes as input a series of key/value pairs and outputs other key/value pairs, relying on basically two functions to perform the actual computing, namely: Map and Reduce. The Map function takes as input a pair and splits it into a set of transitory key/value pairs. After the input is processed and mapped into transitory key/value pairs, the Map function hands its output to the Reduce function. Upon receiving a given key ”X” and a set of values related to this key, the Reduce function aggregates the transitory results and process the data in order to obtain the final expected output.

This approach makes applications extremely flexible, since MapReduce allows for dynamic configuration of the quantity of maps and reduces. In other words, it is possible to configure the granularity of the data splitting process, thus adjusting the size of transitory key/value pairs that are distributed among the processing nodes in the cloud. The main issue in designing MapReduce applications lies in writing optimal map and reduce functions and determining the optimal number of maps for the specific application and the environment where it is executed. This choice is important in order to optimize storage resources access and the utilization of each cloud node’s processing resources.

A. The MapReduce Architecture

A typical MapReduce cloud environment consists in master node, worker nodes and a distributed filesystem access through the environment. The master node acts as a frontend to the cloud environment, receiving jobs and distributing computing loads and the necessary data among the *worker* nodes. The worker nodes are common compute nodes, which actually perform the desired computation on the provided data and send their outputs back to the master node. In this paper, we focus on the MapReduce implementation of the Hadoop project, which already includes the MapReduce processing environment, a distributed filesystem (Hadoop File system), several API spanning many languages and accessory management services. A Hadoop environment mainly consists in the same components depicted in Figure 4. In order to coordinate the interaction between the master node, the several worker nodes, the distributed filesystem and other components that may exist in the cloud, Hadoop provides a set of backgorund control services that run on each node, providing the master node with information about all the worker nodes and the other cloud components.

1) The Hadoop File System: Before analysing the details of the actual MapReduce data processing in a Hadoop environment, it is necessary to first consider the structure of its distributed file system. Hadoop File System (HDFS) consists in volumes spread among the compute nodes in a Hadoop cloud, thus leveraging each individual node’s data storage resources to build big volumes. Besides the clear advantage of simply relying on the storage resources that are already commissioned on the cluster, this architectural characteristic brings another important benefit. Since the data is already located in the compute clusters, it is not necessary to fetch it from a separate storage area network when its needed during computation. This reduces network loads and setup times before a given application can be processed.

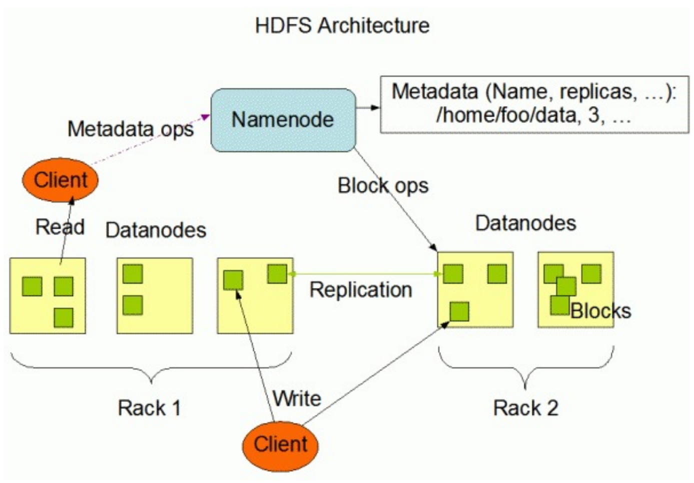

Fig. 5. The Architecture of the HDFS Filesystem [22]

A HDFS environment is mainly composed by NameNodes and DataNodes, which are loosely equivalent to master nodes and worker nodes from a MapReduce perspective. When a file is stored in an HDFS volume, it is divided into several *slices*, that are subsequently spread across different nodes that together provide physical storage resources for the volume. Due to their similarity to the master and worker nodes in a MapReduce environment, the NameNode and DataNodes in a Hadoop setting are usually associated to the master node and the worker nodes, respectively. However, it is possible to set up an HDFS environment with more than one NameNode for backup and high availability support.

The NameNode is responsible for maintaining a central catalogue of the directory structures and files stored in each volume, which contains pointers to the location (*i.e.* DataNode address) of the slices of each file that are spread across the volume. It also handles all volume operations, such as read and write, acting as a frontend to the volume. The DataNodes are responsible for actually storing data, keeping slices of the files stored in the volume, providing access to the data that is stored locally and providing control information about the status of this data. The NameNode polls DataNodes in order to obtain information about data integrity and performs data slice replication tasks to distribute different slices of files across different DataNode, providing protection against failures of individual nodes. The general architecture of an HDFS environment is depicted in Figure 5.

2) The Hadoop MapReduce Architecture: In a Hadoop environment the MapReduce framework is strongly coupled to the HDFS filesystem by several control services in order to form an scalable and resilient cloud infrastructure. The master node is mainly composed by the JobTracker and NameNode services. JobTracker is a service determines the worker nodes that have access to the necessary data and that are available to receive tasks. Based on the information collected from individual worker nodes, it allocates tasks to proper worker nodes. Jobtracke also collects information on each running and scheduled task from the worker nodes, which it uses to allocate nodes and provide user with statistical and mangerial information on the overall cloud status. NameNode simply acts as a NameNode for the Hadoop File System distributed filesystem. Client MapReduce applications access the NameNode service in order to perform tasks such as add, copy, move or delete on files stored in HDFS volumes.

The rest of the cluster worker nodes run other two services that compose the processing environment and data storage architecture, namely: TaskTracker and DataNode. TaskTracker receives tasks (*i.e.* Map, Reduce and Shuffle) from the JobTracker in the master node and reports back to the master node the status of currently running and scheduled tasks. It is has a set of virtual *slots*, which determine the number of tasks that it may accept and process. DataNode acts as a HDFS DataNode and manages local data storage of the slices allocated to each worker node by the master node.

When the master node first receives a job, JobTracker queries the DataNode in each worker node to determine which node already has the necessary data stored locally and then allocates a slot with the TaskTracker running in the given node. If it does not find a node that already has the data, it allocate the first empty slot it finds in any node. Usually, a file stored in a HDFS volume is replicated to several worker nodes through their respective DataNodes. When the TaskTracker receives a new job (consisting of the MapReduce task to be run and the description of the necessary dataset), it spawns a child process to execute it and then reports back on its status to the JobTracker. The structure and processes of a MapReduce job are represented in Figure 6.

IV. The Distributed IDS Architecture

In this section, we introduce the proposed architecture for scalable distributed intrusion detection systems, discussing in the deatils each of its components.

The goal of this distributed intrusion detection system architecture is two solve two main issues: comprehensive data collection accross different network areas and scalable efficient processing and analysis of the resulting dataset. In order to achieve this, we combine previous distributed data collection approaches with cloud computing techniques, achieving a hybrid system where data collection is performed by software

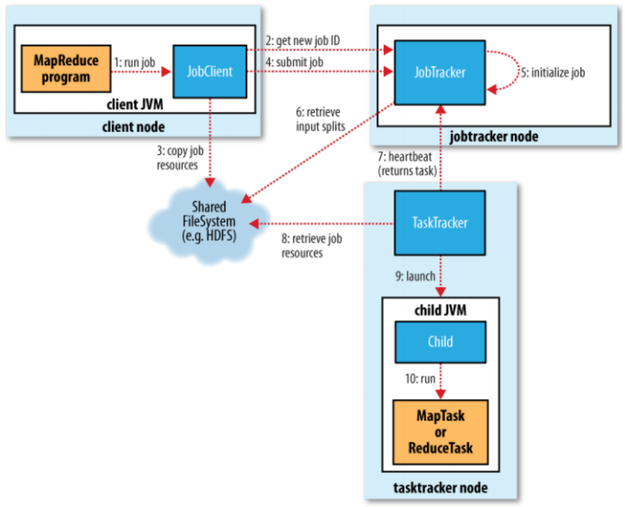

Fig. 6. The structure of a MapReduce job

sensor agents installed in different host in the network and data processing is carried out at a MapReduce cloud that aggregates and analyses collected data. The cloud based attack detection application is able to correlate diverse collected data (such as traffic captures and logs) in order to detect complex attacks and it may easily scale through the addition of more worker nodes, being easily adaptable to different networks.

The proposed distributed IDS architecture is composed by mainly three parts: the sensor agents, the cloud based data storage and processing infrastructure and a web visualization interface. In this architecture, information flows from the data collection sensor agents installed in different network areas to the central MapReduce cloud. Differently from other approaches, instead of being processed by a centralized system, the collected data is then analysed by a cloud application that leverages the resources of the worker nodes in the cloud, scaling transparently as network traffic (and consequently the analysis dataset) grows. The information obtained from the analysis dataset (*i.e.* intrusion alerts) is then conveniently displayed in a web based interface for visualization. The proposed architecture is depicted in Figure 7.

This architecture provides an efficient topology for distributed data collection and storage, tasks that are of great importance in handling large quantities of distributed data (such as logs in the distributed intrusion detection system). It builds on the fact that the Hadoop implementation of the MapReduce framework provides both a distributed filesystem for data storage and a data processing environment. The logs and traffic collected from the different systems spread in the network are aggregated in distributed file system volumes, which may be subsequently accessed by MapReduce applications running on the cloud infrastructure. Notice that this increases the storage capacity of the data processing environment while decreasing data access times, since the data is already in the cluster.

A. Sensor Agents

In order to thoroughly capture network activity in different network segments, our architecture employs several sensor

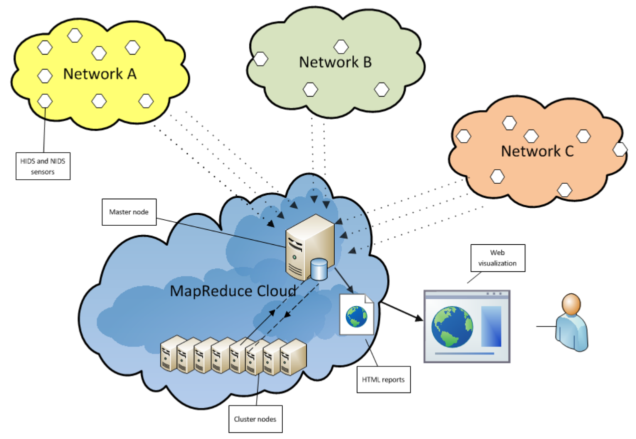

Fig. 7. The Distributed IDS Architecture

agents are placed in different network regions [23]. Applying this distributed collection strategy, the DIDS is also able to correlate activities in different network portions to obtain general overview of network status. The sensor agents are multi-platform applications installed in heterogeneous network nodes located in isolated regions. These agents collect relevant information captured and generated by their host nodes and send it to the master node in the cloud environment, which centralizes data collection. The sensors mainly collect traffic captures and regular IDS logs generated in host systems (possibly by local HIDS solutions). This procedure delivers consistent traffic capture data which thoroughly represents network activity as it contains packets captured in different isolated network regions.

The sensor agents also collect audit data and security logs generated by the host operating system (*e.g.* authentication logs). Correlating this information with traffic captures and regular IDS logs, the intrusion detection model and the analysis system placed in the cloud infrastructure identify and confirm attacks that generate patterns in different layers. As an example, a doorknock attack, where a malicious user tries to login to several nodes using common combinations of usernames and passwords, generates a typical traffic pattern and also causes the security log files to register the failed login attempts. The distributed IDS would then by able to detect such an attack by correlating traffic capture data with security logs collected from the affected nodes.

In order to guarantee the authenticity and confidentiality of the data exchanged between the sensor agents and the master node in the cloud, standard cryptographic techniques can be applied for establishing secure authenticated channels. The simplest alternative is to deploy a Public Key Infrastructure (PKI) across the monitored network and use digital certificates to sign and encrypt the data exchanged by sensor agents. However, if a sensor agent is compromised, it is necessary to revogate its certificate, which is still an open problem. Nevertheless, notice that certificate revocation would not have a great effect if single sensor agents are compromised, since the central DIDS application correlates information from many different sensors and the injection of false data in only one of the sensors would not have a significant impact in the final analysis.

B. Cloud Infrastructure

The cloud infrastructure is a simply a Hadoop environment that aggregate data received from the individual sensors and process it through attack detection algorithms. This is an heterogeneous environment composed of different computers with different resources and architectures. In theory, any platform capable of running a Hadoop MapReduce framework implementation can be used in the cloud infrastructure to process IDS data. Since the only requirement for running Hadoop is an up to date Java Virtual Machine (JVM), this flexibility makes it viable to use legacy equipment for IDS log analysis, reducing the costs of implementing such system.

The hosts in the MapReduce cloud are also part of a distributed filesystem where the data collected by the sensor agents is stored during analysis. The cloud's master node receives the data and stores it in the distributed filesystem where it is accessed and modified in the analysis process. The distributed filesystem seamlessly scales together with the cloud infrastructure providing enough storage space to large quantities of logs without requiring special storage devices. Moreover, filesystem access speed is improved by distributing data among the cloud nodes.

Several intrusion detection algorithms, data analysis, sensor fusion and event correlation models are intended to run as MapReduce jobs on the cloud infrastructure, which provides scalable performance for increasingly large volumes of data processing tasks. Information such as network flows (obtainable from packet capture files) is efficiently processed in a MapReduce grid, yielding fast results even in settings with sheer quantities of logs [5]. This system can also be used to calculate statistical data regarding network activities and monitored nodes security.

C. Web Visualization Interface

After the collected data is processed in the cloud, the intrusion detection models issue alerts regarding detected ongoing malicious activities. It is also possible to extract statistical information from the collected data, yielding results which require different visualization methods. However, the mapReduce framework provides text only files containing the desired results.

In order to provide efficient and adequate visualization of the results obtained in the data analysis process, the result files generated are parsed and relevant information is shown in a web visualization interface. This enables the system to flexibly handle different intrusion detection model outputs and statistical data by simply adding a new module to the visualization interface.

V. Experimental Results

A series of experimental simulations were performed in order to prove the feasibility of the proposed distributed intrusion detection system. In this section we discuss the results obtained, which attest the feasibility of our approach.

In this experiments we analyse the data storage and processing performance of a Hadoop environment deployed on a typical cluster. The cloud infrastructure used for the simulation was composed by one master node and five slave nodes. Each node has a Intel Core 2 Duo 2.66 GHz cpu, 4 GB DDR667 RAM, 300 GB hard disk and a 10/100 Mpbs ethernet network interface. The nodes are connected to a dedicated 10/100 ethernet switch and the Hadoop framework was deployed on machines running a standard setup of CentOS as underlying operating system.

The Hadoop implementation of the MapReduce framework was used together the HDFS distributed filesystem. In order to illustrate the performance of HDFS and MapReduce, two important operations for the proposed DIDS were simulated, namely filesystem input/output and data sorting. Those operations were implemented in the Pig scripting language, which is part of the Hadoop project and automatically creates the necessary map and reduce functions upon receiving a description of the desired operations, thus decreasing development time and complexity.

The first experiment consists in creating new files of varying sizes containing random data on the master node and then moving them to a HDFS volume in order to measure filesystem throughput and performance. Figure 8 shows the time taken to write a file varying from 1 Megabytes to 250 Megabytes to the distributed filesystem. These file sizes were chosen as to represent the quantity of data collected by sensor during one minute. Notice that collecting 250 Megabytes per minute during one day would yield approximately 351 Gigabytes, which is certainly more than many networks generate in terms of logs and IDS data. This experiment shows that write times increase linearly with the file sizes. It is clear that the system would scale to a large quantity of collected data. Moreover, these times can be significantly lowered if a higher throughput network fabric (such as 10 Gbps Ethernet) is deployed or if more nodes are added to the cluster.

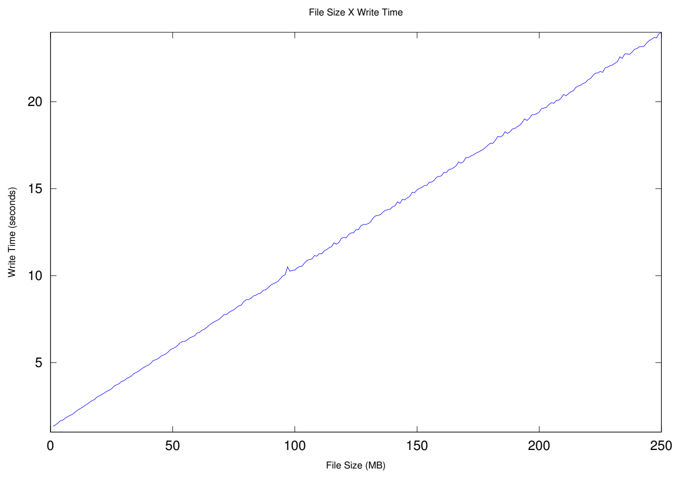

Fig. 8. File system write performance: File size x Time

The second experiment consists in sorting random data varying from 1 megabyte to 250 megabytes using the QuickSort algorithm. In this experiment the same files created in the first experiment are used, which is interesting since the entropy of random files is considerably higher than the entropy of real network logs and traffic, representing a worst case for sorting or pattern matching algorithms. In the tested file size range the sorting times oscillated between 33.4 and 35.6 seconds due to natural oscillations in network performance. The sorting time is essentially the same for data in this range due to the synchronization and communication overhead between the cloud nodes. It is important to notice that there is a constant overhead, which is naturally expected since the nodes that compose the volume have to receive and process synchronization messages before the actual data transfers. This overhead should be accounted for in the design of the DIDS. Even in the case of single megabyte data sorting it is necessary to prepare the nodes and the distributed filesystem to run the required sorting task, which takes a constant amount of time. The time elapsed in the sorting task itself is insignificant when compared to the synchronization overhead, showing that this solution nicely scales to sheer volumes of data.

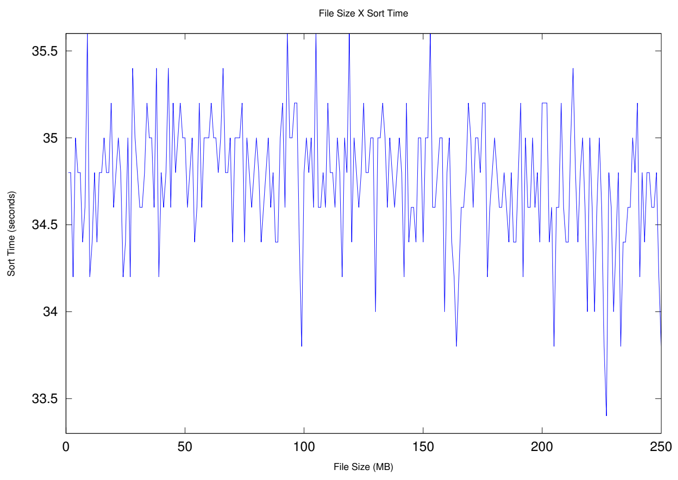

Fig. 9. Data processing performance

Both of the experiments conducted with the cloud infrastructure intended as the central component of the proposed architecture show that it is feasible to store and process massive quantities of data in a Hadoop environment. Both the file creation and transfer process and the sorting algorithm can be efficiently performed on such data volumes on the cloud infrastructure. Those operations are the core of the proposed DIDS, since it constantly collects and classifies information. Hence, these experiments show that the proposed distributed intrusion detection system architecture is feasible for large network environments.

VI. Conclusion

Current intrusion detection systems do not properly handle the sheer amount of traffic and data transmitted in large scale networks. Furthermore, the heterogeneous and decentralized nature of current networks causes certain network regions to be isolated from the network's core, where most of the data used in current NIDSs is captured. We propose an efficient and scalable distributed intrusion detection system based on the MapReduce framework which is capable of handling large volumes of logs and seamlessly scale to handle network growth. Moreover, the proposed DIDS captures data and logs in different regions of the network, efficiently detecting internal and external attacks which occur in isolated network regions. While previous research provide results which attest the feasibility and efficiency of analysing NIDS logs, we present simulation results show that data collection and analysis may be performed in time intervals small enough to provide almost real-time results. As a future work further investigation on intrusion detection algorithms based on the MapReduce framework is to be conducted. Also, a full implementation of sensor agents and MapReduce based analysis algorithms is to be developed and tested.

REFERENCES

[1] Cisco, "Approaching the zettabyte era," 2008, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537 /ns705/ns827/white_paper_c11-481374_ns827_Networking_Solutions _White_Paper.html.

[2] L. Guangchun, L. Xianliang, L. Jiong, and Z. Jun, "Madids: a novel distributed ids based on mobile agent," *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 46–53, January 2003.

[3] J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Computer Security Applications Conference, 1998, Proceedings., 14th Annual*, dec 1998, pp. 13 –24.

[4] W. Lee and S. J. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th conference on USENIX Security Symposium - Volume 7*. Berkeley, CA, USA: USENIX Association, 1998, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267549.1267555

[5] Y. Lee, W. Kang, and H. Son, "An internet traffic analysis method with mapreduce," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, April 2010, pp. 357 –361.

[6] S.-F. Yang, W.-Y. Chen, and Y.-T. Wang, "Icas: An inter-vm ids log cloud analysis system," in *Proceedings of the International Conference on Cloud Computing and Intelligence Systems - CCIS 2011*, 2011.

[7] M. D. Holtz, B. M. David, L. Peotta, and R. T. de Sousa Junior, "An architecture for distributed network intrusion detection based on the map-reduce framework," in *Proceedings of the International Workshop on Telecommunications - IWT 2011*, C. A. Ynoguti and M. C. de Paiva, Eds., 2011, pp. 106–110.

[8] L. Bu and J. A. Chandy, "Fpga based network intrusion detection using content addressable memories," *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, vol. 0, pp. 316–317, 2004.

[9] I. Sourdis and D. Pnevmatikatos, "Fast, large-scale string match for a 10gbps fpga-based network intrusion," *FPL*, vol. 2003, pp. 880–889, 2003.

[10] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Tech. Rep., 2000.

[11] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999.

[12] A. Ghosh and A. Schwartzbard, "A study in using neural networks for anomaly and misuse detection," in *Proceedings of the 8th conference on USENIX Security Symposium-Volume 8*. USENIX Association, 1999, p. 12.

[13] D. Denning, "An intrusion-detection model," *Software Engineering, IEEE Transactions on*, no. 2, pp. 222–232, 2006.

[14] P. Porras and P. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," in *Proceedings of the 20th National Information Systems Security Conference*. Citeseer, 1997, pp. 353–365.

[15] R. Bace, P. Mell, BOOZ-ALLEN, and H. I. M. VA, "NIST special publication on intrusion detection systems," 2001.

[16] B. Mukherjee, L. Heberlein, and K. Levitt, "Network intrusion detection," *Network, IEEE*, vol. 8, no. 3, pp. 26–41, 2002.

[17] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "Internet besieged," D. E. Denning and P. J. Denning, Eds. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1998, ch. DIDS (distributed intrusion detection system - motivation, architecture, and an early prototype, pp. 211–227. [Online]. Available: http://portal.acm.org/citation.cfm?id=275737.275751

[18] M.-Y. Huang, R. J. Jasper, and T. M. Wicks, "A large scale distributed intrusion detection framework based on attack strategy analysis," *Computer Networks*, vol. 31, no. 23-24, pp. 2465 – 2475, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128699001140

[19] E. Alata, M. Dacier, Y. Deswarte, M. Kaniche, K. Kortchinsky, V. Nicomette, V. H. Pham, and F. Pouget, "Collection and analysis of attack data based on honeypots deployed on the internet," in *Quality of Protection*, ser. Advances in Information Security, D. Gollmann, F. Massacci, and A. Yautsiukhin, Eds. Springer US, 2006, vol. 23, pp. 79–91.

[20] L. Tian, "Design and implementation of a distributed intelligent network intrusion detection system," *Electrical and Control Engineering, International Conference on*, vol. 0, pp. 683–686, 2010.

[21] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008.

[22] D. Borthakur, "Hdfs architecture guide," http://hadoop.apache.org/common/docs/current/hdfs_design.html.

[23] T. Bass, "Intrusion detection systems and multisensor data fusion," *Commun. ACM*, vol. 43, pp. 99–105, April 2000. [Online]. Available: http://doi.acm.org/10.1145/332051.332079

.

Rafael Timóteo de Sousa Junior received his B.E. degree in electrical engineering from the University of Paraíba, Paraíba, Brasil, in 1984, the MSc degree from Ecole Supérieure D'electricité Supelec, France, in 1985 and the PhD degree from Université de Rennes, France, in 1988. He is currently a professor in the Department of Electrical Engineering in the University of Brasília. His professional experience includes technological consulting for private organizations and the Brazilian Federal Government. Professor de Sousa is currently the leader of the LATITUDE and CORTECS research groups at the Networking Laboratory of the aforementioned institution. His research interests include Mobile Ad-hoc Networks, secure routing, computer forensics, business intelligence and computational trust.

Marcelo Dias Holtz received his B.E. degree in network engineering from the University of Brasília, Brasília, Brazil, in 2009. He is a member of the LATITUDE and CORTECS research groups at the aforementioned institution and is now pursuing a MSc in electrical engineering at the University of Brasília, focusing on distributed cloud based security data analysis systems. Before that, Holtz worked on network management and security methods, also have having extensive experience in consulting and joint projects with the Brazilian government. His current research interests include cloud computing, network management and security.

Bernardo Machado David is a member of the Cryptography & Information Theory group at the Electrical Engineering Department - University of Brasília. Bernardo has worked on several joint projects between the University of Brasília and Brazilian governmental organizations as a consultant in IT security and infrastructure. He has authored and co-authored scientific papers on security in ad-hoc networks, cloud computing environments, online banking systems and multi-party computation. He is currently pursuing a B.E. degree in network engineering and doing research in theoretical cryptography and network security, focusing on secure multi-party computation, post quantum cryptography and cloud security.