Big Data Meets Cloud Computing Integration using the Importance of Virtualization

Prof. Manjunath R. Associate professor
Of Computer science Department and
City engineering college, affiliated to VTU
Karnataka, Bangalore, India

Student Akshatha A. and Sneha R of mtech in
Computer science Department and
City engineering college, affiliated to VTU
Karnataka, Bangalore, India

***Abstract*— Network traffic is a rich source of information for security monitoring. However the increasing volume of data to treat raises issues, rendering holistic analysis of network traffic difficult. In this paper we propose a solution to cope with the tremendous amount of data to analyse for security monitoring perspectives. We introduce an architecture dedicated to security monitoring of local enterprise networks. The application domain of such a system is mainly network intrusion detection and prevention, but can be used as well for forensic analysis. This architecture integrates two systems, one dedicated to scalable distributed data storage and management and the other dedicated to data exploitation. DNS data, Net Flow records, HTTP traffic and honeypot data are mined and correlated in a distributed system that leverages state of the art big data solution. Data correlation schemes are proposed and their performance are evaluated against several well-known big data framework including Hadoop and Spark. Cloud computing is a modern technology that increase application potentialities in terms of functioning, elastic resource management and collaborative execution approach. The central part of cloud computing is virtualization which enables industry or academic IT resources through on- demand allocation dynamically. The resources have different forms such as network, server, storage, application and client. This paper focus as on how virtualization helps to improve elasticity of the resources using cloud computing environment In Big Data.**

Keywords*— *Big data, Cloud computing and Virtualization

I. INTRODUCTION

Cloud computing refers to a collaborative IT (Information Technology) environment, which is planned with the intention of measurable and remotely purveying scalable IT resources for effective and efficient utilization. National Institute of Standards and Technology (NIST) has given a definition for Cloud computing which says that ―Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (eg., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction‖. Five essential characteristics of cloud computing listed by NIST are on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. Mobile cloud computing is the computing which refers to anytime, anywhere accessibility to applications and data through internet using mobile devices. Traditional computing resources are stored in an individual device and accessed by an authenticated user. In Cloud computing, resource are stored in centralized manner and accessed on demand basis. In recent days, mobile devices and subsequent mobile computing become an imperative component in cloud computing. Internet made the possibilities of accessing applications and data from anywhere at any time. According to Juniper research , the mobile users and enterprise market for mobile cloud based applications worth are expected to increase to $9.5 billion by 2014. Aepona describes that MCC (Mobile Cloud Computing) as a new paradigm for mobile applications whereby the data processing and storage are moved from the mobile devices to powerful and centralized computing platforms located in clouds. These centralized applications are then accessed over the wireless connection based on a thin native client or web browser on the mobile devices.

The detection and prevention of network intrusions is recurrent security problem. It has been studied for over thirty years with the first concept of Intrusion Detection System (IDS) being proposed in 1987 . However it remains an open research topic due to the constant evolution of types of data to analyse. In addition, the adaptation of attackers' techniques to cope with new means of protection and firewall policy makes it a continuously evolving field. Moreover, it raises new challenging issues related to identifying relevant features for intrusion detection, as well the means of processing the increasing volume of heterogeneous security data produced by a network. The operations of Network Intrusion Detection Systems (NIDS) rely on network traffic analysis, where Snort] and Suricata are typical examples. Network traffic from several protocols (HTTP, SIP, DNS, etc.) is inspected to find anomalies. These anomalies are defined by rules that rely on either signatures or anomalous traffic behaviour. If such anomalies are observed, the system either raises an alert (IDS) or stops the communication (IPS). Current IDSs analyse several protocols and data and events observed by them are correlated by SIEM (Security Information and Event Management) in order to detect intrusions. One shortcoming is that current solutions realizing in-depth packet analysis are not scalable and adaptable to big network producing high quantity of data.

The contribution of this paper are:

• We introduce a new intrusion detection architecture that correlates several data sources (HTTP, DNS, IP flow, etc.),

• We propose a solution for processing and storing data coming from different data storage system in a single facility,
• We present data correlation schemes useful for security monitoring and evaluate these against several state of the art distributed computing system including Hadoop and Spark.

II. VIRTUALIZATION FOR CLOUD

Virtualization technology diverts the human's perspective for utilizing IT resources from physical to logical. The goal of virtualization is to collaboratively utilize the IT resources such as storage, processor and network to maximum level and to reduce the cost of IT resources which can be achieved by combining multiple idle resources into shared pools and creating different virtual machines to perform various tasks simultaneously. The resources can be allocated or altered dynamically. User should be conscious of basic techniques such as emulation, hypervisor, full, para and hardware assisted virtualization while using virtualization in cloud computing environment.
Emulation: It is a virtualization technique which converts the behaviour of the computer hardware to a software program and lies in the operating system layer which lies on the hardware. Emulation provides enormous flexibility to guest operating system but the speed of translation process is low compared to hypervisor and requires a high configuration of hardware resources to run the software.
Virtual Machine Monitor or Hypervisor: A software layer that can monitor and virtualize the resources of a host machine conferring to the user requirements. It is an intermediate layer between operating system and hardware. Basically, hypervisor is classified as native and hosted. The native based hypervisor runs directly on the hardware whereas host based hypervisor runs on the host operating system. The software layer creates virtual resources such as CPU, memory, storage and drivers.
Para Virtualization: This technique provides special hyper calls that substitutes the instruction set architecture of host machine. It relates communication between hypervisor and guest operating system to improve efficiency and performance. Accessing resources in para virtualization is better than the full virtualization model since all resources must be emulated in full virtualization model. The drawback of this technique is to modify the kernel of guest operating system using hyper calls. This model is only suitable with open source operating systems.
Full Virtualization: Hypervisor creates isolated environment
Between the guest or virtual server and the host or server hardware. Operating systems directly access the hardware controllers and its peripheral devices without cognizant of virtualized environment and requirement modifications.

III. VIRTUALIZATION TYPES

There are three major types of virtualization such as Server virtualization, Client virtualization and Storage virtualization. The architecture and categorization of virtualization techniques.

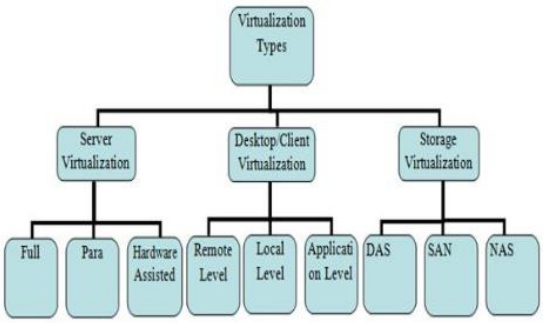

FIG 1: TYPES OF VIRTUALIZATION

Server Virtualization: In server virtualization, single server performs the task of multiple servers by portioning out the resources of an individual server across multi-environment. The hypervisor layer allows for hosting multiple applications and operating systems locally or remotely. The advantages of virtualization include cost savings, lower capital expenses, high availability and efficient use of resources.
Client Virtualization: This client virtualization technology makes the system administrator to virtually monitor and update the client machines like workstation desktop, laptop and mobile devices. It improves the client machines management and enhances the security to defend from hackers and cybercriminals. There are three types of client virtualization . First, remote or server hosted virtualization which is hosted on a server machine and operated by the client across a network. Second, local or client hosted virtualization in which the secured and virtualized operating environment runs on local machine. Third, application virtualization that provides multiple ways to run an application which is not in traditional manner. In this technique an isolated virtualized environment or partitioning technique is used to run an application.
Storage Virtualization: It creates the abstraction of logical storage from physical storage. Three kinds of data storage are used in virtualization, they are DAS (Direct Attached Storage), NAS (Network Attached Storage) and SAN (Storage Area Network). DAS is the conventional method of data storage where storage drives are directly attached to server machine. NAS is the shared storage mechanism which connects through network. The NAS is used for file sharing, device sharing and backup storing among machines. SAN is a storage device that are shared with different server over a high accelerate network. Hypervisor is the software package that controls working access to the physical hardware of host machine. There are two kinds of hypervisor models as hosted and bare metal / native. Hosted hypervisor instance operates on top of the host operating system whereas bare metal based hypervisor operates directly on the hardware of host machine. Fig 2 shows the comparison between traditional, bare metal and hosted models.

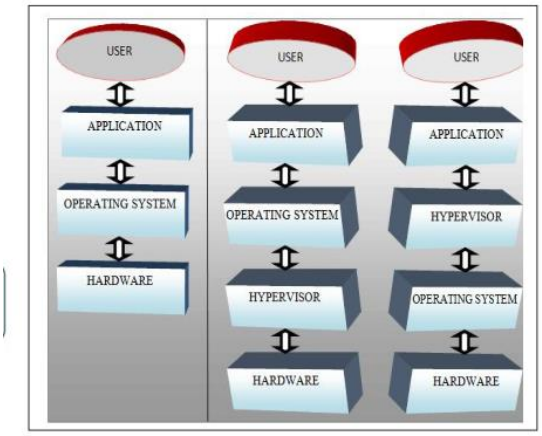

FIG 2 TRADITIONAL MODEL VS BARE MODEL VS HOSTED MODEL

Majority of obstacles arises in the acceptance and development of virtualization and cloud computing are concerned to the basic management aspects such as data leakage, virtualization security threats, data remanense issue, privacy and elastic resource management.

IV. DATA TORAGE AND PROCESSING

We briefly introduced in pevious section that the data exploited by our system is captured at different points and stored in various data storage systems (see Figure 1). This data storage heterogeneity is due to specific requirements for each kind of data (passive DNS, Dionaea, NetFlow). During experiments our system was deployed in a 200 employees company having activities related to electronic payment.

To give an idea of the amount of data collected per time period and justify the storage choices here are some values measured during the testing phase:

• Dionaea honeypot: around 1,000 connection attempts from 15 different hosts per day.

• NetFlow: average of 9,600,000 flows per day with an export eve Minute (approximately 450 Megabytes).

• DNS: around 13 million DNS replies per day (approximately 1.5 Gigabyte). Dionaea honeypot, with few daily connections initiated by few attackers does not have high requirements. In addition over the observation period some IP addresses seemed to be redundantly connecting over time. A basic SQLite database is chosen to store the logged information (IP, port, protocol, uploaded payload, etc.). Consultation of the SQLite database is fast enough for this small quantity of data. It is worth noting that even with a larger network to monitor this amount of data would not vary a lot. Having a network of 500 machines or 10,000 would not impact the quantity of data logged by a single honeypot machine deployed in the network. Storing NetFlow records is more challenging especially if we need the approach to be scalable. NetFlow are exported using nfdump every minutes in order to have an almost real time view of the communications. NetFlow records are stored in nfcapd format binary files on several distributed servers. The use of several servers is not mandatory for our example but the quantity of flows exported by a router grows with the size of the network it serves. This choice for Netflow Storage ensures to meet storage scalability require ments for any network size. However it raises some issues for data treatment as nfcapd files are binary files. Big data framework such as Hadoop have an input format for MapReduce tasks that is usually text based. Even though Hadoop supports building sequence file format for binary input/output, nfcapd files would have to be first converted in a HDFS-specific (Hadoop Distributed File System) sequence file before being uploaded. This process implies a high computational over-head which is time consuming and not acceptable for real-time security monitoring. The alternative is to develop a new API that directly reads NetFlow data in the native nfcapd format. Such solution is proposed in through a binary input format for reading packet and NetFlow records concurrently in HDFS. This solution outperforms the previously cited one and is proved fast, providing a throughput of 14 Gbps in a 200-node testbed according to experiments performed in . As for NetFlow, DNS monitoring produces a massive amount of data as seen in our measurements. In addition this volume of data grows with the number of users/machines making DNS queries, i.e. the size of the network. Contrarily to NetFlow, all data does not need to be stored for DNS monitoring and only partial information is extracted from DNS packets in order to avoid information redundancy and save storage space. Typically, stored information consists in all possible DNS resource records for a domain name, the TTL of each, some flags, the timestamp for first seen and last seen, etc. All DNS packets do not need to be saved, hence we chose to store DNS related information in a database.

To meet data storage and availability requirements, DNS data is extracted from packets and stored in an Apache Cassandra database. Cassandra is a distributed database solution for data storage that exhibits high performance in data access. It is a decentralized database allowing to store Terabytes of data. In addition, Apache Cassandra integrates Hadoop management since version 0.6 ensuring easy interfacing with state of the art solution for big data processing. This implementation ensures our architecture to fit to larger network than the one we performed tests on.

V. P ERFORMANCE ANALYSIS

We presented in previous sections an architecture for large scale monitoring. We described data extraction and storage as well as theoretical correlation scheme and their applications. In this section we test the proposed correlation schemes against

Several big data management systems in order to find the most suitable for such applications. Two well-known open source big data frameworks are assessed, the popular Apache Hadoop ecosystem and the Spark project from AMP Lap of Berkeley University. We focus on performance comparisons of five components of these two frameworks namely Hadoop, Pig, Hive, Shark and Spark.

VI. BIG DATA TOOLS PRESENTATION

We briefly present here the big data tools we used for the performance assessment of our architecture. For a more detailed description of these tools

• Hadoop is a distributed batch processing framework to process and to analyse large scale datasets. It consists of two primary components, which are HDFS (Hadoop File System) and MapReduce data processing model [20]. Hadoop employs a master/slave architecture to manage a cluster.

• Hive is an open source data warehouse infrastructure running on top of Hadoop. It proposes a high level programming language that abstract the implementation of MapReduce jobs to give an user-friendly interface to Hadoop. Commands are expressed in the form of SQ Like Queries, thanks to a language called HiveQL

• Pig is also a high level distributed programming model built on top of Hadoop. The main difference between Hive and Pig is on their purpose. Hive is appropriate for database users while Pig targets experienced programmers who are not used to write declarative SQL query.

• Spark: Like Hadoop does, Spark proposes a distribute data processing solution for data-intensive applications with the difference that data to process is stored in-memory. Spark has been proved up to 100 times faster than Hadoop for specific tasks like iterative jobs.

• Shark is a sub-project of Spark that implements Hadoop's Hive on top of Spark such that it is fully compatible with Hive.

Experiments and Results The five big data solutions are tested in four different scenarios relevant for the computation of the metrics introduced in Section II-B. Experiments were conducted on a cluster of eight machines (one master node and seven slave nodes). Each machine runs a 12.04.4 x86 Ubuntu operating system on an Intel(R) Core(TM) 2 Duo with 4GB of RAM. The versions of experimental frameworks are Hadoop-1.2.1, Hive-0.9.0, Pig0.11.1 Spark-0.6.1 and Shark-0.2.1. The Spark framework was assigned with 2 GB of memory per node i.e. 14 GB of working memory in total. The dataset used consists in 767 MB of network traffic. All the scenarios were run ten times for each of the five framework.

The four scenarios are the followings:

• Scenario 1: find packets that match a given source IP address and a given source port.

• Scenario 2: find packets containing a given substring in their payload.

• Scenario 3: count the number of destination IP per source IP and order the result.

• Scenario 4: join two sets according to a common key i.e. the source IP addresses Scenario 1 corresponds to finding information according to two IP addresses that is given.

VII. CONCLUSION AND FUTURE WORK

In this paper we introduced a new scalable architecture for protecting from and detecting network intrusions virtualization techniques, virtualization types, hypervisor techniques and challenges incloud computing system to reduce IT costs and effective utilization of cloud resources such as rapid elastic provisioning of virtual machines, elastic application programming model. In addition, the virtualization techniques get universal support when users consider elastic resource management issues and security issues before moving into cloud. In future, we aim to develop new policies, framework and techniques to maintain elastic resources and data availability, as a result, the performances of cloud services could steps into next higher level.. This system collects and stores in a distributive manner honeypot data, DNS data, HTTP traffic and IP-flow records. Several correlation schemes relying on this data are introduced and their application, ranging from intrusion detection to forensic analysis, are listed. Five state of the art big data frameworks that can fit for such an architecture are evaluated in four scenarios of data correlation relevant for security monitoring. Out of this performance analysis Spark and Shark appear to be the best performers in all scenarios and thus the best suited to implement the solution. Even though our architecture computes score with few delay, it still use off-line analysis tool with Hadoop and Shark. Future work will consist in implementing the same system with on-line analysis big data framework such as Spark Streaming or 0Storm. This paper discussed various This study paper discussed various issues pertaining to cloud services which can be used to design strong frame work for effective elastic resource management in cloud

REFERENCES

[1] J. P. Anderson, "Computer security threat monitoring and surveillance," Fort Washington, Pennsylvania, Tech. Rep., 1980.

[2] D. E. Denning, "An intrusion-detection model," IEEE Transactions in Software Engineering, vol. 13, no. 2, Feb. 1987.

[3] M. Roesch, "Snort - lightweight intrusion detection for networks," in Proceedings of the 13th USENIX conference on System administration, ser. LISA '99, 1999, pp. 229–238.

[4] V. Paxson, "Bro: a system for detecting network intruders in real-time," in Proceedings of the 7th conference on USENIX Security Symposium Volume 7, ser. SSYM'98, 1998.

[5] "Suricata, open source ids/ips/nsm engine." [Online]. Available: http://www.suricata-ids.org

[6] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster, "Building a dynamic reputation system for DNS," in Proceedings of the 19th Usenix Security Symposium, 2010.

[7] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive DNS analysis," in Proceedings of NDSS, 2011.

[8] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An overview of ip flow-based intrusion detection," Communications Surveys Tutorials, IEEE, vol. 12, no. 3, pp. 343–356, Third 2010.

[9] J. François, S. Wang, R. State, and T. Engel, "Bottrack: Tracking botnets using netflow and pagerank," in Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I, ser. NETWORKING'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 1–14.

[10] C. Kreibich and J. Crowcroft, "Honeycomb: creating intrusion detection signatures using honeypots," SIGCOMM Comput. Commun. Rev.,vol. 34, no. 1, pp. 51–56, Jan. 2004

[11] F. Weimer, "Passive dns replication," in Proceedings of the 17th Annual FIRST Conference on Computer Security Incident Handling, 2005.

[12] R. Edmonds, "ISC Passive DNS Architecture," Internet Systems Consortium, Inc., Tech. Rep., 2012. [Online]. Available: https: //security.isc.org/Passive DNS/passive-dns-architecture.pdf

[13] A. Lakshman and P. Malik, "Cassandra: structured storage system on a p2p network," in Proceedings of the 28th ACM symposium on Principles of distributed computing, ser. PODC '09. New York, NY, USA: ACM, 2009, pp. 5–5.
[14] "Cisco netflow." [Online]. Available: http://www.cisco.com/web/go/netflow
[15] "Dionaea, catches bugs." [Online]. Available: http://dionaea.carnivore.it/
[16] S. Marchal, J. Franc¸ois, C. Wagner, R. State, A. Dulaunoy, T. Engel,
[17] P. Mell, T. Grance, ―The NIST Definition of Cloud Computing‖, National Institute of Standards and Technology, Information Technology Laboratory, Technical Report Version 15, 2009.
[18] S. Perez, ―Mobile cloud computing: $9.5 billion by 2014‖, http://exoplanet.eu/catalog.php, 2010.
[19] White Paper, ―Mobile Cloud Computing Solution Brief,‖ AEPONA, November 2010.