Cloud processing of security events for VANET distributed sensors

I P Bolodurina¹, Y A Ushakov¹, D I Parfenov¹, J Cui²

¹Orenburg State University, Orenburg, Pobedy str., 13, 460018, Russia
²Anhui University, Anhui, Hefei, 12487, China

E-mail unpk@mail.ru

Abstract. Machine-to-machine communication networks, which are related to human activities, are today one of the fastest growing IT sectors. The most vulnerable in terms of security is the Internet of Things and VANET segments. When devices interact, various types of targeted attacks are possible, as well as failures or random dangerous events. The handling and response to security events, in this case, must be fast. However, the specificity of VANET networks does not allow the use of classical intrusion prevention systems due to the constant movement between connection points, for example, in LTE or 5G networks. The article is devoted to the possibility of quickly deploying a cluster of security events analysis based on containerization on existing virtual machines that are closest to the event source. The analysis of the scaling speed, as well as the performance limits of event processing processes on various configurations of the data analysis cluster.

Keywords: cluster, Hadoop, Spark, big data, security.

1. Introduction

Around the world, the number of smart devices is increasing every year. The interaction of such network devices generates significant amounts of data transmitted. With the increase in the number of devices the volume of transmitted data the complexity of control over end devices grows. As a consequence, the process of administering such heterogeneous networks requires more and more resources to monitor, identify, and prevent possible destructive information impacts on them. In this regard, the vehicular ad-hoc networks (VANETs) are the most vulnerable.

In VANET networks, data is transmitted both between vehicles (Vehicle To Vehicle, V2V) and between vehicles and base stations of the 5G network (Vehicle To Road), when vehicles move in their coverage area. The main purposes of such networks are road safety, assisted driving, distribution of alerts and warnings, navigation, provision of integrated road services. In this way, VANETs carry critical information that directly affects road safety [1]. Therefore, the development of methods to ensure confidentiality, integrity, and availability of data, as well as the design of means of protecting the data transmission medium in VANET networks is an important area of research.

The amount of data that needs to be processed and analyzed grows exponentially with the number of connected devices. This is especially true for data for security and attack detection. At the present stage
of technology development, tools have appeared that allow storing, processing, and analyzing large amounts of data, and approaches to analysis have also begun to change.

The most popular big data analytics solutions are clusters and cloud structures [2]. Existing solutions for creating cloud systems, such as cloud resource management systems OpenNebula, OpenStack, CloudStack, Doku, etc. are universal, they do not take into account the specifics of working with big data - the possibility of parallelizing the network (alternative routing and switching), a large amount of memory consumed and calculations per cycle of analysis. This leads to the impossibility of deploying the structures responsible for processing big data in an arbitrary cloud.

Providers provide out-of-the-box cloud toolkits in BigData-as-a-Service format. In most cases, this is a Hadoop / Spark cluster with storage on HDFS / Hive / Gluster, NoSQL services like Redis / MongoDB [3].

There are several problems associated with the use of distributed computing for processing big data. Consumers often lack computing resources for big data processing. It is also necessary to ensure the planning of processes and flows serving individual parts of the algorithm or data structure, to plan the placement and transfer of data between nodes with isolated networks. Also, there is a need to predict the amount of computation, execution time, consumed processor, and memory resources when executing distributed applications and their subtasks. To efficiently use the resources of dedicated virtual machines, it is necessary to ensure the selection of the optimal pool of virtual machines for the tasks to be solved for processing big data. Also, there is the problem of isolating the work and deploying distributed applications in multi-user environments of public computing service providers - it is necessary to ensure the security of the calculations performed and the data being processed in an environment where tasks of different users can run on the same virtual or physical machine.

2. Review of big data processing methods

The most flexible solution when creating a cloud infrastructure is to use a platform that supports IaaS technology.

Cloud systems can plan computations at the level of virtual machines. With the advent of the era of cheap servers, the Apache Hadoop project began to develop, which made it possible to build clusters for storage and processing of data. But with the proliferation of cloud computing and the increase in data volumes to hundreds of terabytes, Hadoop's lack of flexibility in dealing with random sampling has led to the increased popularity of alternative methods. Apache Spark can use many components of the Hadoop cluster, but at the same time, it works much more efficiently with the analysis of arbitrary data samples, especially in distributed storage [4]. Also, a direct analysis of big data is available in neural network frameworks, which have also begun to integrate with Spark [5]. There is also a native Python Dask solution for distributed analysis, but the flexibility of the Spark cluster and full compatibility with the existing storage system, the ability to work with different schedules, and worker nodes makes it more versatile.

Not so long ago, elements of computing on GPU accelerators were added to common cluster platforms. Hadoop and Spark began to be used in various projects. An analyst does not have to write a new Hadoop program every time. Some extensions allow you to formulate queries in a simpler language, for example, Apache, Hive (queries in SQL language). Spark has a flexible programming model that allows you to store data in RAM for multiple processing [6].

For network operation, Hadoop uses the Master-Slave model, and there is work to optimize data transmission using software-defined networks [7]. For correct operation, it is necessary to have direct access from Slave to Master, which is not always possible to organize immediately.

Software-defined networks are actively used to solve various problems arising in cloud platforms: migration of virtual machines between different data centers [8], data flow control in parallel DBMS, data flow control in frameworks for the development of distributed scientific applications (Hadoop MapReduce, Hadoop -OFE, Orchestra, Coflow, OpenMPI), building locally managed MAC addresses for virtualizing the physical infrastructure of the data center [9], power management of network devices in the data center [10], etc.
Algorithms for routing data flows in software-defined networks when choosing paths do not take into account the need to provide QoS parameters for the laid and previously installed data flows [11]. The solution to this problem is important since requires proactive routing of data flows between nodes (containers, virtual machines) executing individual processes of neural network applications.

The main advantage of containerization technology compared to virtualization is the reduction of overhead. To organize the operation of full-fledged virtual machines, full emulation of the computer hardware by the hypervisor (for example, KVM, Xen, Hyper-V) is required, each virtual machine includes a complete copy of the guest OS. In the case of using container technology (for example, Linux-VServer, OpenVZ, LXC, Docker), there is a single common kernel of the computer OS, on top of which lightweight containers with processes of running programs run, there is no need to emulate the hardware - the physical means of the server are used [12].

The most important advantage of containers is the ease of deployment of applications and services, taking into account all the dependent components, files, and libraries [13]. Also, containers are launched much faster compared to virtual machines, because there is no need to load the OS, you just need to run the necessary applications [14].

As part of this work, it is planned to launch big data cloud applications in a set of containers mapped to the physical nodes of the data center.

3. Security events data collection and analysis scheme

The main requirements for the selection of tools are the ability to work in distributed systems and support for streaming data processing. Since a large amount of heterogeneous data is generated in real-time, which needs to be brought to a common form, processed and analyzed, the selected tools must support the ability to work with streaming data. The sources of data flows are authentication systems, active network equipment, IDS / ISP, server event logs, antiviruses, vulnerability scanners, and other information security protection and control systems.

The distributed security data processing system consists of two dependent parts:

- receiving and recording data;
- data processing and response.

Since event processing is required all the time, the delay between the arrival of data in storage and processing must meet the requirements for these events. To save history and restore the history of events, to create a description of incidents, and identify threats with a long pre/aftereffect, it is necessary to store a big data for a long time. The data entry scheme is shown in Figure 1.

![Security event data collection scheme](image)

Instead of Apache Kafka, you can also use Spark Streaming, which, according to tests, works more stable under high loads, but has its characteristics. The Hadoop NameNode is located where the node can be accessed from the outside; the node can be used for running internal Hadoop clients and for
external Hadoop services. To interact with external consumers directly, Hadoop uses the forwarded HDFS and WebHDFS ports, as well as the mode of collecting cluster addresses via DNS.

Reading data consists of the following steps (Figure 2):

- Request to the application, which will be accompanied by data reading
- Reverse proxying of the request to the container.
- Request for DataNode addresses with the required information
- Return the list of addresses.

Addressing the required address, reading information.

This architecture works transparently through the Swarm or K8n cluster overlay network and transmits data encrypted using configured IPSEC mechanisms.

![Figure 2. Container interaction scheme.](image)

To use external processing systems you need to run Master and several Slaves. Their connection will be carried out according to the specified Master service name through the DNS of Docker.

Spark Streaming, an extension of the Apache Spark API, splits streams into small batches of time intervals. This mechanism is called DStream. DStream micro-packages contain resilient distributed datasets (RDDs). RDD sets support transformation and action operations. A feature of transformation operations is deferred execution: Spark remembers which transformation and on what data should be performed, and the result is returned when the action is called. DataSets can be used in place of RDDs. When using a DataSet, before performing the transformations, the optimizer is triggered to determine a more efficient way to obtain the result. Working with it occurs mainly using SQL queries [15].

The disadvantages of Spark include variable load on the network during data loading and processing, which makes it necessary to set limits on the input stream density. Another disadvantage of Spark is the lack of the ability to recover clusters from failures. To solve these problems, the Apache Kafka framework will additionally be used, which is capable of creating real-time data pipelines and streaming applications. Replicated and persistent storage reduces the risk of data loss.

4. Organization of data transfer and storage

Raw data enters the system from network devices, antivirus, firewalls, firewalls, etc. To receive data from these systems, there is Kafka Connect, a tool for scalable and reliable streaming of data between Apache Kafka and other systems. Kafka Connect can accept entire databases or collect metrics from all application servers into Kafka partitions, making data available for low latency streaming processing. Since the sources are different, a different Kafka section is created for each.

Events are handled by the Spark Streaming application. The hierarchical structure of events is determined by the way the event is received and its features. There are three large classes of events: events received using SNMP, events received using Syslog and other events. Each class can be subdivided into smaller subclasses; subclasses can be subdivided as well.

It becomes necessary to draw up rules according to which the correlation analysis will be carried out. In the system under development, it is planned to use only two types: templates and queries. A template
is a set of rules with some fields that are filled with data from an event. The main difference between a request and a template is working with data streams. They allow you to combine, group, filter, deduplicate, sort, run events through sliding windows. The template and queries can specify the actions that the system should perform when the data matches the template. The simplest response is to alert the administrator of suspicious activity.

The data processing architecture is shown in Figure 3. Information from sources that do not cover SNMP and Syslog are collected using Kafka and fed to the data collection server. All received raw data is stored in the database, from where it is retrieved by the Spark application. It converts, filters, and aggregates events, and then performs correlation analysis. Correlation rules are taken from a separate store. All obtained results are saved in a separate database. If incidents are detected, a message is sent to the response module, which notifies the administrator.

Since data processing also includes notification of detected patterns and critical events, the diagram shows some of the modules for interacting with the interface.

5. **Experiment with handling events in a container-based cluster**

A Docker-based cluster was created for the experiment. Virtual machines in a private cloud based on OpenNebula are used as nodes; the nodes are connected by a 10Gbps network, and have common network storage with a separate connection to each node. Also, each node has local storage that is used to store data. To generate events, Metasploit attack generation, the method of mirroring traffic through SPAN, detection on IPS Suricata, and redirection to the cluster are used (Figure 4).

To test the scaling speed of a Docker-based cluster, all nodes use a dedicated repository located on the central node. This repository is used to download images to launch cluster members and is used to speed up container deployment. Also, the nodes have the usual variants of the cluster elements (Spark, Hadoop, Kafka).

The experiment consists of measuring the speed of automatic scaling of the cluster, as well as comparing the speed of Docker mode and the standard model of the cluster. In the absence of events, the Spark cluster can be reduced to one working container, but when events arrive, it is possible to scale up to the required number if the nodes with the operating system are already running.
The start times for a different number of Spark worker containers are shown in Figure 5. The time is estimated from receiving a start command to registering with the Spark Master control process. Up to 4 Spark worker processes will run on one physical node.

As you can see from the graph, the launch speed depends on the number of working containers launched but is within acceptable limits.

To estimate the recording rate and the rate of event processing, let's start generating events, as well as processing them in the cluster. Since latency is important in handling security events, the scheduler latency and event latency will be measured. As processing, aggregation, and grouping of events are used, which refers to the time window. This method simultaneously loads both the storage system and the kernel. Figure 6 shows the processing latency for a maximum number of containers and an inbound stream of events with the exponential time between events at a rate of 850 events/s.

As a result of measurements, it can be seen that the failure of individual processes does not affect the overall processing process, increasing the processing time for individual jobs due to the redistribution of the job and waiting for a response from the old process. To estimate the maximum rate of event processing, let's measure the processing acceleration at various values of the number of worker processes.
Graph on Figure 7 shows non-linear acceleration, which is limited by the speed of concurrent storage access.

6. Conclusion
As a result of the research, a system for rapid deployment of a security event processing cluster was created. The use of containerization allowed us to create a system of fast scaling. The measurement result showed the efficiency of this approach, and measurements of its performance and delays allow us to conclude that it applies to the analysis of a security event.

Acknowledgments
The study was carried out with the financial support of the RFBR in the framework of scientific projects No. 18-07-01446, No. 20-57-53019, as well as a grant from the President of the Russian Federation Federations for state support of young Russian scientists - candidates of sciences (MK-860.2019.9) and by the grant of the President of the Russian Federation for state support of leading scientific schools of the Russian Federation (NSh-2502.2020.9).

References
[1] Cooper C, Franklin D, Ros M, Safaei F & Abolhasan M 2016 A comparative survey of VANET clustering techniques IEEE Communications Surveys & Tutorials 19(1) 657-81
[2] Choi TM, Wallace SW & Wang Y. 2018 Big data analytics in operations management Production and Operations Management 27(10) 1868-83
[3] Bhattacharjee A, Barve Y, Khare S, Bao S, Kang Z, Gokhale A & Damiano T 2019 Stratum: A bigdata-as-a-service for lifecycle management of iot analytics applications 2019 IEEE International Conference on Big Data 1607-12
[4] Erraissi A, Belangour A & Tragha A 2017 Digging into Hadoop-based big data architectures International Journal of Computer Science Issues 14(6) 52-9
[5] Johnsrini Venkatesan N, Nam CS & Shin DR 2019 Deep learning frameworks on apache spark: a review IETE Technical Review 36(2) 164-77
[6] Mavridis I & Karatza H 2017 Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark Journal of Systems and Software 125 133-51
[7] Zhao S and Medhi D 2017 Application-Aware Network Design for Hadoop MapReduce Optimization Using Software-Defined Networking IEEE Transactions on Network and Service Management 14(4) 804-16
[8] Boughzala B, Ben Ali R, Lemay M, Lemieux Y & Cherkaoui O 2011 OpenFlow supporting inter-domain virtual machine migration Eighth International Conference on Wireless and Optical Communications Networks 1-7
[9] Matias J, Jacob E, Sanchez D & Demchenko Y 2011 An OpenFlow Based Network Virtualization Framework for the Cloud IEEE Third International Conference on Cloud Computing Technology and Science 672-8
[10] Heller B, Seetharaman S, Mahadevan P, Yiakoumis Y, Sharma P, Banerjee S & McKeown N 2010 ElasticTree: Saving Energy in Data Center Networks Proceedings of the 7th USENIX conference on Networked systems design and implementation 1–16
[11] Ibanez G, Naous J, Rojas E, Rivera D, Schuymer BD & Dietz T 2011 A Small Data Center Network of ARP-Path Bridges made of Openflow Switches The 36th IEEE Conference on Local Computer Networks
[12] Zeng H, Wang B, Deng W & Zhang W 2017 Measurement and evaluation for docker container networking 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery 105-8
[13] Zheng C & Thain D 2015 Integrating Containers into Workflows: A Case Study Using Makeflow, Work Queue, and Docker VTDC ’15: Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing 31-8
[14] Huber N, von Quast M, Hauck M & Kounev S 2011 Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments CLOSER 563-73.

[15] Nabi Z 2016 DStreams: Real-Time RDDs Pro Spark Streaming 29-50.