Aggregation of Elastic Stack Instruments for Collecting, Storing and Processing of Security Information and Events

Igor Kotenko
St.Petersburg Institute for Informatics and Automation
of the Russian Academy of Sciences (SPIIRAS),
39 14th Liniya, St.Petersburg 199178 Russia
ivkote@comsec.spb.ru
St. Petersburg National Research University of Information
Technologies, Mechanics and Optics,
49, Kronverkskiy prospekt, Saint-Petersburg, Russia

Artem Kuleshov and Igor Ushakov
The Bonch-Bruevich Saint - Petersburg State University of
Telecommunications (SPbSUT),
22 (b.1) pr. Bolshevikov, St.Petersburg 193232 Russia
{gart9515, ushakovia}@gmail.com

***Abstract*—The paper suggests an approach to construction of the system for collecting, storing and processing of data and security events on the basis of aggregation of instruments provided by Elastic Stack. Basing on the analysis of the monitoring and incident management tasks for computer security and comparative analysis of existing technologies and architectural solutions the technical requirements for such systems are identified, and on their basis the architecture of the proposed solution is formed. The paper describes the developed system for data collecting, storing and analyzing for various components of information security systems. Results of experiments with the developed prototype are presented.**

Keywords—Big Data; security information and event management; SIEM systems; Elastic Stack.

I. INTRODUCTION

Technological progress does not stand still, and information security systems are being developed and evolve with it. The systems for security information and event management (SIEM) are no exception. Previously, the functionality of the classic SIEM solutions for large and medium sized companies more or less satisfied existing requirements. However, nowadays new mechanisms and functions are required that are able to timely and adequately identify, process and analyze current information flows and security events and to manage incidents for a much larger number of devices with significantly increased amounts of information and speed of information flows [1-4]. The problem is that modern SIEM systems are insufficiently adapted to the timely processing of large amounts of security information and events needed to assess the current state, perform incident management and develop countermeasures.

In this paper we set the task of developing the architecture and implementation of a research prototype of the system for collection, storing and processing of security information and events based on big data technology, as the basis for a new generation SIEM system, as well as the preliminary analysis of the functioning parameters of this system. Specific of the task we set is the ability to apply big data technologies for monitoring and to select the most productive architecture.

The main contribution of the paper is in the integration of a bunch of existing open source packages to create a complete security monitoring system. The solution being proposed differs from existing ones in integration of Elastic stack, Nginx, and Docker software for collecting, storing and processing large amounts of data in order to analyze security information and events. This solution is aimed at providing high-performance message processing with regard to possible overloads and further expansion of the system. One rationale for the work is that existing commercial solutions are too expensive for SMEs to adopt. The paper has the following structure. In *section II* a concise analysis of relevant works and the most representative SIEM products is given, their advantages and disadvantages are discussed. *Section III* identifies the technical requirements for a next generation system for collecting, storing and processing of security information and events, and on their basis a general architecture of the proposed solution is formed, and noting the characteristics of the products, available on the market, the selection of the instruments for implementation is carried out. *Section IV* describes the proposed approach to the implementation of the system for collection, storing and processing of security information and events that meet the specified requirements, and its implementation is described. *Section V* presents the results of the experiments and comparison of our prototype with several others architectures from research papers. In *conclusion*, the findings are shown and directions for future research are identified.

II. RELEVANT SOLUTIONS

Let us select two areas for review of relevant solutions as research works and software implementations of commercial products and open source ones.

Aiming at objective and clear understanding of existing architectures of SIEM systems, let us first consider some research papers. [5] presents a generalized architecture of

978-1-5386-0435-9/17/$31.00 ©2017 IEEE

SIEM systems of the new generation, which can be divided into levels of network, data, events and applications. The following main system components are outlined: collector, universal translator of events, highly reliable data bus, scalable event processor, repository, system for decision making and response, component of the attack modeling and security analysis, predictive security analyzer and visualization system.

[6] identifies three architectural levels to build SIEM systems: data analysis, data management and data collection. This approach allows to analyze the complexity of processing and the number of events to handle at each level. It is concluded that the most loaded and computing power consuming level is data collection.

Research, described in [7], were aimed at analyzing the central component of any SIEM system, which is the data storage system. The benefits of storing information in the hybrid repository are presented. The suggested approach to storage provides a convenient and reliable data sharing across heterogeneous storage systems and can provide the performance high enough.

Systems that are capable of working with big data are integrated into growing number of different products, and SIEM products are no exception. Representative of such a system is Hadoop [8]. It works on the principle of batch processing, when direct analysis does not depend on data flow. The platform provides instruments for processing structured and unstructured large files. The component of resource management MapReduce [9] responds for this functionality. The basic idea of MapReduce is to distribute the tasks using a large number of nodes, organized in a cluster.

[10] presents the result of implementation of this approach (speed was up to 30,000 MB of processed data per second). Such performance is possible using a large number of resources to calculate, in particular, there were used 1800 nodes, each of which included two Intel Xeon 2GHz CPUs and 4Gb RAM. In the referenced paper the technology of batch processing was discussed, but for processing the data in the stream they use stream processing, allowing to track messages in real time.

This task is performed, for example, by Apache Storm [11]. Cisco has provided an analysis of this approach [12], capable of handling over a million packages per second on a single node system.

Gartner [13] highlights some of the most advanced SIEM systems, including HP ArcSight, IBM Qradar (commercial) and AlienVault OSSIM (open source).

SIEM HP ArcSight [14] has modules for event monitoring, behavioral analysis, system of rules for security events processing. Due to the closed source code it is difficult to add new features. HP Arcsight uses its own product CORR as DBMS. In the system by default the ability to store events coming in with no particular pattern or mask is not implemented, i.e. to add new information an additional intervention into the system is required. The system implements centralized data storage. For geographically distributed organizations it is required to transmit all events to the central database, where all processing is performed. The ArcSight ESM system core is licensed by volume of logs per day. In addition to the core, you must license a variety of settings and options, for example, the number of users, development of own connectors, the number of event sources (considered separately for source types), modules of compliance, log management, etc. which means that ArcSight is focused on large corporations.

IBM SIEM Qradar [15] integrates into a single, unified solution for information management and security event management system logs, anomaly detection, configuration management and vulnerability remediation. This system allows to reveal attributes of most critical incidents. Qradar uses the advantages of unified architecture for the analysis of system logs, data flows, vulnerabilities, user data and information resources. However, this approach leads to a technically challenging process of increasing the resources used by the system. The acquisition cost of IBM Qradar SIEM is composed of many factors of distribution and configuration of the system. The price of the license depends on the processing capabilities, measured in events per second. High price does not allow to use this solution for small and medium-sized enterprises.

OSSIM (Open Source Security Information Management) [16] is a SIEM system based on open source code from the company AlientVault. OSSIM implements functions of collection, analysis and correlation of events, as well as intrusion detection. It includes the host intrusion detection system (HIDS), network intrusion detection system (NIDS), intrusion detection system for wireless networks (WIDS), the components of network nodes monitoring, network anomaly analysis, vulnerability scanner, system for the exchange of information about threats between users, a set of plug-ins for parsing and correlation of syslog records with a variety of external devices and services. The main disadvantage of these solutions is the limited functionality for the aggregation of received messages.

As the SIEM products developed gradually, with the advent of the concept of big data into the information security domain most of them were forced to introduce additional set of methods of Hadoop (from Apache Software Foundation) to work with large flow of information. Today, the situation has not changed, and Hadoop is used as the main solution. It is worth considering that the Hadoop project was developed with the aim of building a software infrastructure for distributed computing. The module YARN, issued in autumn of 2013, made this technology universal for data processing, is considered as a great development for the project, however, in our opinion, is redundant for solving problems of collection of indexing and data distribution.

III. Requirements, generic architecture and the choice of instruments for implementation

Basing on the results of the analysis of relevant works, the present paper aims at development of the generic architecture and a research prototype of the system for collecting, storing and processing of data and security events based on big data technology, as the basis for a next generation SIEM system.

It is assumed that the decision must comply with the complex of requirements. Let us enumerate the basic requirements: (1) ensuring robust collection of information from many different sources; (2) elasticity of system, that is providing of optimal (rational) distribution of load and low dependence of the performance of the components at change of individual components; (3) possibility of efficient integration of own algorithms into the analytics subsystem for further development of the SIEM system; (4) minimization of time to deployment and configuration of infrastructure, services, and other system tasks; (5) implementation of efficient mechanisms to ensure fault tolerance and high availability at the software level; (6) support for horizontal scaling, that is the ability to divide the system into individual components and their distribution into separate physical machines; (7) prevalence of development tools of the market system and the active user community.

The system architecture should be built on microservices that will provide system virtualization, fault tolerance and fast horizontal scaling. It will also allow to automate and simplify deployment. For efficient integration of algorithms of analysis it is required to support the developed API from popular programming languages such as Java, C++ or Python. There should not be hard linking of the prototype to a particular source event, i.e. the reception of information must be accessed through the mask or universal pattern. It should be possible to use multiple network streams or similar components for load balancing, to avoid overload on the server side. The main component of the architecture of the SIEM system is a set of tools performing the functions of collecting, storing, processing of security information and events and analytic capabilities that jointly represent the system of monitoring and incident management [3, 4].

The generalized theoretical architecture of the developing SIEM system has four levels: network, data, events and analytic processing. *Network level* consists of aggregation agents that are presented on different nodes and have a function of aggregation and sending information from different sources such as: systems logs, security logs from firewall systems, sensors, etc. Main components of the *data level* includes functions of indexing and preparation for data storage. Due to the initial separation and load balancing, the indexing could be fulfilled on multiple nodes in parallel. This helps to achieve elastics and horizontal scalability. Before saving, messages are indexed and processed. It helps the search and correlation engines do not use much compute power that helps to process a lot of information for a short period of time. *Events level* consists of two components: (1) Analysis and evaluation of events for data aggregation and structuring; (2) Real-time monitoring controlling the main parameters of information security. Separation of the data and events levels helps to create backup storages and components that work with it, and to make complex parallel tasks of aggregation, storage and preliminary analysis of received messages. Eventually that helps to create fault tolerance infrastructure. *Level of analytic processing* includes functions of security analysis, anomaly detection and countermeasure generation. This level could be installed on separate physical server. The main idea of the described architecture is to provide complex scenarios to manage security incidents. System core includes the functions of big data aggregation and storage, and could store the data for future forensic analysis.

On the contemporary market there are many commercial and open source solutions for the collection, storing and processing of security information and events.

One of the most popular solutions of this kind is the commercial software product Splunk Enterprise [17]. This product is an analytical platform for collecting and analyzing machine data with automatic load balancing. It has the ability to increase productivity by adding typical servers, that provides horizontal scalability and fault tolerance of the system. Splunk has a documented API and SDKs for popular programming languages. This framework handles data in any format, including dynamic data from software applications, application servers, web servers, operating systems, and many other sources. In addition, Splunk is a commercial product with closed source code, which limits and slows down the development of the API and makes scaling of the system paid.

Another common commercial solution is ManageEngine EventLog Analyzer [18] with similar advantages and disadvantages. The product EventLog Analyzer supports load balancing using the internal service. The algorithms, methods and platform core are not available at the software level, and the entire setup and operation is done via an API that evolves depending on the needs of the consumer, which reduces the flexibility of the system.

In 2014, Cisco Systems has published the source code of the solution OpenSOC [19]. It is a solution to create the centers of cyber threats monitoring, based on the Apache Big Data open source. This source code has become the basis for the Metron Apache project [20]. This solution has the architecture that was initially focused on processing large arrays of data from multiple distributed information systems. Currently, the Apache project Metron is not actively developed by open community; the statistics from Google search and GitHub data [21] on adding code to the project show weak interest from potential users. The development of the project is now performed almost solely by the company Hortonworks, which offers a commercial implementation of the solution. Unfortunately, the tasks adapted architecture of the Apache Metron has a poorly implemented system of inverse scaling, and to build the simplest system it is required to deploy a large number of system components and resource-intensive services, most of which will not be used. In terms of the average resources it is sometimes impossible to provide loading of data from sources, for which this solution was designed, while the components of the solutions require a lot of manpower to deploy and configure. The project Apache Metron deserves attention in case of necessity to solve the tasks aimed at maintenance of high-loaded and distributed information systems.

One of the most efficient of open source software, specializing in solving specified problems, is Graylog [22]. This is a free centralized system for collecting, storing and

analyzing information. This system uses the functions of Elasticsearch. It is worth considering that, despite the frequent updating Graylog and developed user community, the integration of the latest versions of Elasticsearch into the project requires a lot of time. Proof of this is that in 2017 the latest version of Graylog 2.2.1 only works with Elasticsearch version 2.4.4 [23], which is obsolete.

Due to partial inability to satisfy all the requirements formulated above in the present work we suggested the following approach. Basing on the results of analysis of the available industry tools we selected the open source software complex Stack Elastic [24], as it is suitable to meet the previously established requirements. Besides, it is free and popular solution, managed to prove its efficiency in a large number of implementations worldwide. The main software components are: Elasticsearch, as a search and analytical system; Logstash, as a software pipeline for data processing; Kibana, as a tool for visualization and navigation through the system; Beats, as a set of programs required for collection and transportation of system logs and files. Software stack, consisting of Elasticsearch, Logstash and Kibana (named ELK hereinafter) was specifically designed to solve the problems of collecting, storing and processing of system logs. It should be noted that the decision on the software stack ELK is a mature product, and on its base there was implemented the commercial solution Elastic X-Pack that combines the above technology with the addition of auxiliary analytical capabilities. Let us consider the basic components of Elastic Stack, which are used in the project.

Elasticsearch is a distributed search and analytical system core that supports the architectural style REST API and sending data via JSON. It centralizes incoming data and supports clustered architecture, allowing the system to scale out. It is a transparent and reliable system and has a failure detection system, which in its turn ensures high availability. Elasticsearch kernel performs real-time search over large volumes of heterogeneous data structures (documents). Document is the basic unit of information that can be indexed. Documents are specified in JSON format. The system has well developed API, and the list of supported languages for interoperability includes Java, Python, C++ and others.

Logstash is a software pipeline for data processing, it simultaneously collects data from many different sources, primarily processes them and sends them to the storage subsystem. It has built-in parser, allowing to normalize heterogeneous data, to determine the geographical coordinates by IP, to process information from various sources regardless of format and structure.

Kibana is a software component that implements visualization and navigation in the Elastic Stack. It presents data as customizable interactive dashboard in real time and implements a large number of built-in custom widgets (histograms, graphs, maps, and other standard tools). It has well developed API.

Beats is a set of programs - collectors of data with low requirements on resources that are installed on client devices to collect system logs and files. There is a wide choice of collectors, and the ability to write your own collector. Filebeat transmits to the server the information from dynamically updating system logs and files that contain text information. Winlogbeat is used for similar actions with the Windows system logs. Packetbeat is a network packet analyzer, which transits information about network activity between the application servers. It captures network traffic, decodes the protocols, and retrieves the required data. Metricbeat periodically collects metrics of operating systems that characterize, for example, CPU and memory usage, number of packets transmitted in the network, the state of the system, and services running on the server.

IV. Architecture and Implementation of the Prototype

The implemented prototype has the architecture shown in Fig. 1. Data collected using Winlogbeat and Metricbeat as an XML document is sent via Internet to the server where the ELK server is deployed. Data is taken by a Logstash instance that is configured to collect data, which serves as a collector. Then a special instance of Logstash indexes incoming information from collectors and sends the data to Elasticsearch for subsequent storing and processing.

All the analytical processing is performed by the software component Kibana. To realize the function of viewing the Kibana representation by the domain name in the Internet, you need an Nginx server, where the information from output port of the Kibana is proxied to the 80th port. The received user commands via the API are transmitted from Kibana to Elasticsearch, where they are processed and receive the resulting data.

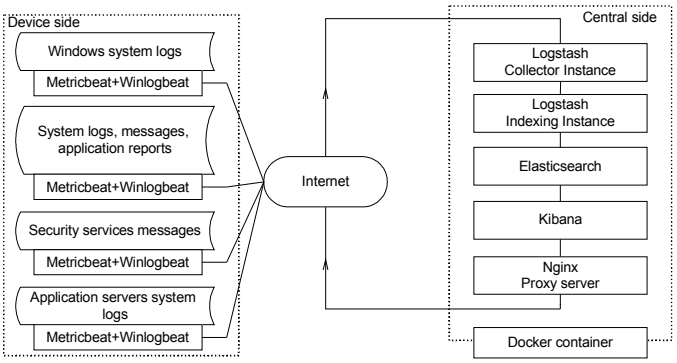

Fig. 1. The architecture of the prototype

The architecture of the prototype can be conditionally divided into the following components: (1) the subsystem for sending data from client devices; (2) the subsystem for pipelining and data delivery; (3) the fault tolerance and load balancing mechanism; (4) the subsystem of search and analytical core, combined with the storage subsystem; (5) the visualization subsystem.

Let us consider the first four subsystems.

The prototype is developed with the goal to be the basis of the SIEM system, therefore it is necessary to provide in the prototype the broad coverage of available information for analysis. Currently, it supports event collection of syslog

protocol, Windows events logs, telemetry hardware, OS and services, as well as information about the flow of network traffic from netflow/sflow with Filebeat, Winlogbeat, Metricbeat, and Packetbeat, respectively. In the future, if it is needed to send specific data, it is possible to write own Beat-collectors on the basis of the presented library Libbeat and well developed API.

For processing and delivery of data in the Elastic Stack the Logstash implementation is used, shown in Fig. 2.

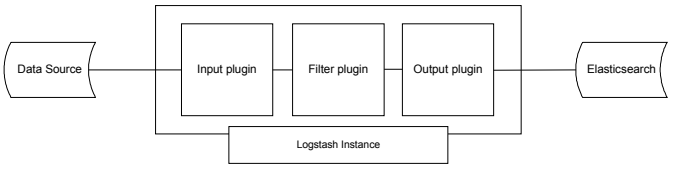

Fig. 2. The scheme of Logstash functioning

It is divided into three major functional blocks, implemented in the form of three extensions (plugins): *Input*; *Filter*; *Output*. In the functional block *Input plugin* the specific event source is specified, which is read by Logstash pipeline. In the prototype, these sources are beats. It accepts documents in JSON format that contain data from system logs, system metrics, the information from the protocols and other available data in accordance with the selected collector. The block *Filter plugin* performs intermediate processing of an event. This allows you to structure data by extracting only necessary information, such as date, time, IP address, error code etc. and storing them in data structures, to send data further to output plugin for onward transmission to Elasticsearch. The filter used is selected depending on the characteristics of the event. Note that this unit is resources consuming, so the Filter plugin is actively using parallel computing. In the *Output plugin* the further route for processing of documents in JSON format is specified. This is the final phase of the pipeline's functioning. In the prototype data is transferred to the subsystem of analysts and storing in Elasticsearch.

The architecture of the pipeline for data processing and delivery in the prototype is shown in Fig. 3.

In case of excess of the rate of incoming events over processing speed the Logstash implementation begins to drop events. To prevent loss it was proposed the use of message broker Redis as a buffer (it is possible to use the industry standard solutions such as Redis, Kafka or RabbitMQ).

In the implemented prototype the subsystem of pipeline processing on the basis of Logstash was divided into two separate program services:

(1) the service for data receiving from the event sources beats, and further sending to the buffer of the massage broker;

(2) the service for reception data from the buffer, further processing and sending to Elasticsearch.

The usage of multiple object instances of Logstash with the division of functional roles allows you to use load balancing between the data source and the Logstash cluster.

To avoid the impossibility of entering data of a specific type, when an instance of Logstash of this type is not available, we use a specially configured Logstash pipeline supporting a plurality of input plugin modules. For example, if the subsystem had only one instance of the object with Logstash file input plugin, then when it fails it would be impossible to take data from Filebeat. Increasing the number of input plugin modules allows you to scale horizontally, while separated parallel receiving pipelines increase system reliability and eliminate single point of failure.

Elasticsearch module output plugin is also configured for automatic load balancing using a multitude of nodes in the Elasticsearch cluster. If one of the nodes fails, the data stream is not interrupted, which eliminates single point of failure. This ensures high availability of the cluster and route traffic to active nodes in the cluster.

The use of multiple instances of Logstash objects with the division of functional roles allows you to use load balancing between the data source and cluster Logstash.

To avoid the impossibility of entering data of a specific type, when an instance of Logstash of this type is not available, we use the specially configured Logstash pipeline that supports plurality of input plugin modules. For example, if the subsystem had only one instance of the object with Logstash file input plugin, when it fails it would be impossible to take data from Filebeat. Increasing the number of input

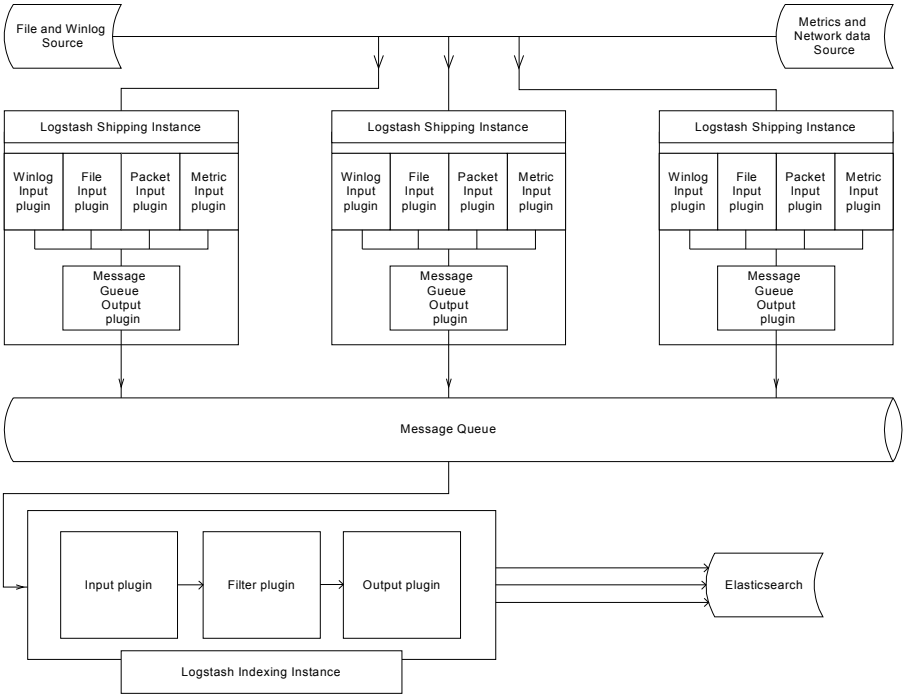

Fig. 3. The architecture of the pipeline for data processing and delivery

plugin modules allows you to scale horizontally, while separated parallel receiving pipelines increase system reliability and eliminate single point of failure.

In the prototype the Elasticsearch cluster was deployed. In the terminology of Elastic Stack, the cluster is a set of nodes (servers) that hold all the information and provide the ability to index and search across all nodes. In the terminology of ELK the set of structures-documents that have any similar characteristics, is called as an *index*. For convenience, the indexes are divided into types – documents with common fields. Potentially the index can grow to large sizes that exceed the physical capabilities of the node. This index is therefore divided into several parts called *shards*. This allows you to distribute data across multiple nodes, and to distribute and parallelize operations with the shards, which increases performance and throughput. To prevent failures and ensure fault tolerance mechanism, Elasticsearch allows you to make copies of shards of the index which are called *replicas*. The shard and its replica are never placed on the same node.

In the prototype the system elements are deployed in Docker containers that allow you to automate the deployment and management. To accelerate development and testing, as well as to easily upgrade to the next version of the services in the future, we selected an microservices approach and deployment in the environment of container virtualization.

This separated the subsystem into modules, making it easier to upgrade individual services and test their compatibility with each other. At the same time, containers solve problems of reliability and backup of infrastructure services, enable engineers to focus on the logic of the solution and abstract from infrastructure problems.

V. THE RESULTS OF THE EXPERIMENTS

With the purpose of validation of the proposed solutions, a series of experiments was conducted. The test bench for the study had the following characteristics: 16 GB of ddr3 technology RAM; 4-nuclear processor with frequency of 1.9 GHz; Elasticsearch of version 5.1.1, Kibana 5.1.1, Logstash 5.1.1, Winlogbeat 5.1.1, Nginx 1.10.2, Java(TM) SE Runtime Environment (build 1.8.0_73-b02); operating system CentOS 7 Linux 3.10.0-327.36.3.el7.x86_64. One of the important criteria of operation of the monitoring system is the system throughput, i.e. the number of packets from the device processed in a certain period of time. When a packet arrives at the ELK server port, Elasticsearch takes it and starts to index information. After indexing it saves the received data in the database and updates the API. The update occurs not immediately, but in the time interval set in the parameters as refresh_interval, which by default is 1 sec. The update of the API requires processing power, so if you load a large number of packages, at possibility of delay of the monitoring, the interval is been increased, which increases the speed of indexing. All experiments were carried out with refresh interval equal 1 sec. For the experiment to the server via WLAN with maximum throughput 10 Mbps there was connected the computer from which different libraries of pre-prepared system logs were sent. During the experiment the library without additional information about the source was sent, i.e. one message. The average indexing and processing amounted to 13,000 packets per second with an average CPU utilization of 65%. Fig. 4 shows the dependence of the CPU time as a percentage of time. Fig. 5 shows the dependence of the number of packets processed from time. However, for SIEM systems the errors tests are not very helpful. In order the system can track, for example, from which node the message arrived, which process was the sender, and what is the time of the incident, etc., messages are packaged into JSON packets, containing the additional information. During the second experiment a library of JSON packets with additional information was sent. As a result, the size of the packets increased relatively to the first experiment and on average was equal to 330 bytes. This had an impact on the processing speed, which on average was equal to 4000 packets per second and CPU utilization averaged 40% (Fig. 6 and Fig. 7).

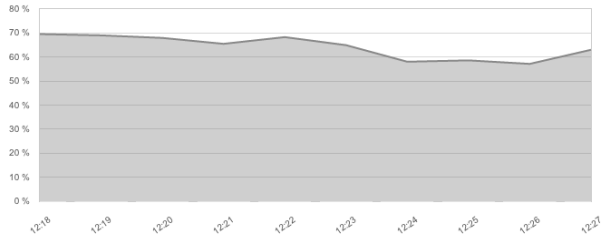

Fig. 4. CPU utilization while processing packets without additional metrics

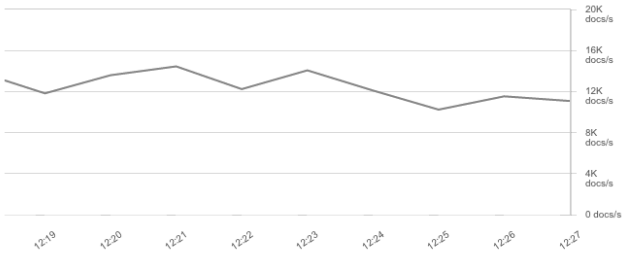

Fig. 5. Speed of packet processing without additional metrics

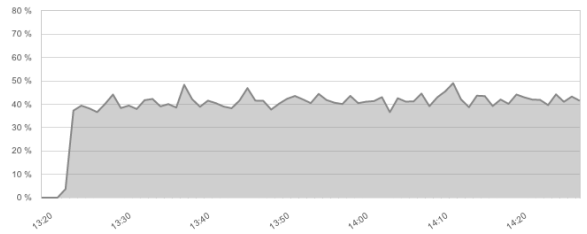

Fig. 6. CPU load when processing JSON packets with extra metrics

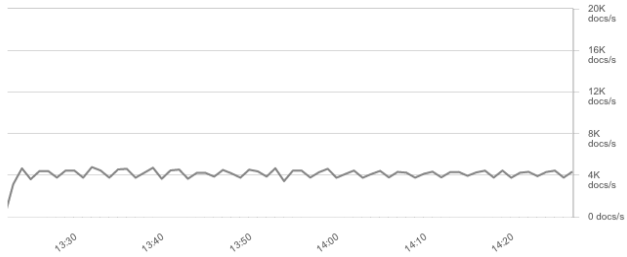

Fig. 7. Speed of packet processing without additional metrics

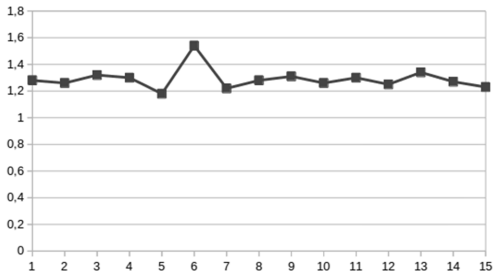

Fig. 8. Time of search of nodes by the phrase with tests' numbers

Another primary mechanism for SIEM systems is analytics, for which the ability to quickly extract the necessary information from the database is critical. To check the time of the search, the system Elasticsearch was experimentally sent requests for load of indexed and saved messages (*nodes*) containing the word of 5 letters. From the obtained results, presented in Fig. 8 (where the Y-axis specifies the time in ms, and the X axis is the number of tests), we revealed that the average search time of nodes by the phrase is 1.28 ms.

The experiments demonstrated that the computational power of the system is sufficient to increase the processing speed, but the speed increase was not happening.

The reason for this is the bandwidth limitations of the network. However, the set goal of processing and presenting a large number of packets in real time is achieved. Given the fact that Elastic Stack has support for netflow protocol, the analysis of network protocols in real time is possible.

The demonstrated in the experiments large potential of the system at low consumption of computing power allows to obtain a significant gain in comparison with analogues and provides an opportunity for further research with large volumes of incoming security information and events.

Let us consider several solutions that were created by researchers and compare them with our prototype. Table I represents the comparison of several systems.

TABLE I. COMPARISON PARAMETERS OF THE SYSTEMS

Considered system	**Volume if input data**	**Number of servers/nodes**	**Processing time**	**Method of data processing**	**Main task of the developed prototype**
Massive Distributed and Parallel Log Analysis [26]	From 100 to 500 MB	Up to 8 slave-servers	450 seconds for 500 MB of traffic processing of 8 servers	Stream data processing. Data from logs loads to one server storage	Development of the architecture for processing shared logs information
VSS Monitoring. Leveraging a Big Data Model in the Network Monitoring Domain [27]	More than 3,5 Exabyte	Not presented	Speed of traffic from 100 Mb/s to 100 Gb/s	Ability to data processing based on batch or stream mechanisms	An ability to split network analytic from storage system. An ability to get more effectiveness with integration of different data, equipment, infrastructure of big data
Toward a Standard Benchmark for Computer Security Research. WINE [28]	More than 100 TB. Including: up to 10 million URL-reputation domains; up to 2,5 millions of e-mail accounts; 130 million of antivirus telemetry	240000 sensors though the world	Not presented	Ability to data processing based on batch or stream mechanisms. Hadoop/Spark support	An ability to data process, gathering them from millions of hosts worldwide, using key security fields
Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats [29]	More than 74 GB about 144 million events	One physical server with 16 cores	1500 sec. using 15 cores	Batch data processing mechanism	The developed scheme of attack detection has to integrate all information security incidents that are collected by the organization
RF (Random Forest) [25]	500 GB	10 nodes, joined in the cluster. There are 500 trees used for algorithm	Average time - 517,8 s	Classification is based on votes, where each tree verify that each object is confront to each class	Algorithm of machine learning, that is able to process the data with big number of signs classes
Spark-MLRF [25]	500 GB	10 nodes, joined in the cluster. There are 500 trees used for algorithm	Average time -186,2 s	Based on Apache Spark Mllib	Method of parallel work of the Random Forest algorithm
PRF (Parallel Random Forest) [25]	500 GB	10 nodes, joined in the cluster. There are 500 trees used for algorithm	Average time is 101,3 sec	Based on the platform Apache Spark	Hybrid method of parallel optimization of the Random Forest algorithm
Developed system based on Elastic Stack instruments	1 GB of stream data indexing. For data processing we used library with 2,5 million rows (768 MB)	1 physical server: 16 GB of RAM, 4 cores processor, 1.9 GHz, throughput 10 Mb/s	Average time – 794 ms to transmit 330 byes of JSON packets	Stream and indexing data processing	Distributed search and analytic core have to search a big amount of different types of data

Let us consider parameters we used in the table.

1. *Volume of input data*. There were represented results of tests and experiments in analyzed papers where authors presented information about test bed prototypes, using different volume of data [25-29]. Comparing with papers [25-29] our prototype uses 1 GB of data processed in stream way. This is a little bit more that in paper [25], but much more lower than in other papers [26-29]. In general this is enough for confirmation that our system is workable and also we could make a conclusion of satisfaction of the requirements for processing and analyzing data in real time.

2. *Number of servers/nodes used in the prototypes*. Most of the papers we considered use parallel and load balancing mechanisms to split the data in multiple streams [25-28]. This

parameter determines the ability of a system to load balance traffic between different nodes in a network. In our prototype we used one physical server with characteristics that are presented in the table 1, however, different components of the system were located on different virtual machines.

3. *Processing time*. This parameter shows the speed of indexing and analyzing time of the data, loaded in the system. There is not possible to make a comparison between papers [25-29] using this parameter, because we have no ability to make experiment on one platform. However, taking into consideration a power and time of data processing that we got in our experiment results, we could suppose that Elastic Stack is one of the most productive solutions in area of big data.

4. *Method of data processing*. This parameter describes type of data processing [25-29]. According to the Table 1 we could see that most of the researches use stream method of data processing [25-28]. Our prototype uses stream data processing mechanisms also, that is a key parameter for building new generation SIEM-systems.

5. *Main task of the developed prototype*. Each research paper that is considered in Table 1 solves a specific challenge. All the prototypes we have considered in this paper are able to proceed the huge amount of stream data for fast detection of information security incidents. The goal of the prototype we developed is to solve the challenge of creation a shared search and analytic core control system and management of the incidents that is able to search information through huge volumes of different data types.

VI. Conclusion

The paper presented the architecture and the prototype of the system for the collecting, storing and processing of security information and events based on big data technologies. To solve the problem the analysis of relevant papers and modern SIEM products and solutions, implementing the collecting, storing and analysis of system events and telemetry data was performed. Based on this the generalized architecture of the monitoring system that meets the requirements was suggested. We selected the solution on Elastic Stack and implemented a prototype system for research purposes basing on it. We conducted several experiments demonstrating the performance of the developed system. The ability of this solution to collect, store, structure and analyze data of any type with high performance, flexibility and extensibility of a software system provides wide opportunities for further product development in the area of security, monitoring and control of information systems. Future research and development will be focused on further improvement of the system architecture, the study of the interaction of the components with each other for security information and events processing, as well as on analysis and experimental evaluation of the functioning of the system for different security information and events streams.

VII. Acknowledgment

The work is performed by the grant of RSF #15-11-30029 in SPIIRAS.

References

[1] Big Data Analytics for Security Intelligence. Cloud Security Alliance. September 2013. pp.1-12.

[2] R. Zuech, T.M. Khoshgoftaar, R. Wald, “Intrusion detection and Big Heterogeneous Data: a Survey”, *in Journal of Big Data*, Springer, December 2015. pp.1-42.

[3] I.V. Kotenko and I.B. Saenko, “Creating New Generation Cybersecurity Monitoring and Management Systems”, *Herald of the Russian Academy of Sciences*, vol.84, no.6, 2014, pp.993-1001.

[4] I. Kotenko, O. Polubelova, and I. Saenko, “Data Repository for Security Information and Event Management in Service Infrastructures”, *in SECRYPT 2012 - Proc. of the International Conference on Security and Cryptography*, 2012, pp. 308-313.

[5] I. Kotenko and A. Chechulin, “Common Framework for Attack Modeling and Security Evaluation in SIEM Systems”, *in 2012 IEEE Intern. Conference on Green Computing and Communications*, 2012, pp. 94-101.

[6] I. Kotenko, A. Chechulin, and E. Novikova, “Attack Modelling and Security Evaluation for Security Information and Event Management”, *in SECRYPT 2012. Intern. Conference on Security and Cryptography,* 2012, pp. 391-394.

[7] I. Kotenko, O. Polubelova, and I. Saenko, “The Ontological Approach for SIEM Data Repository Implementation”, *in 2012 IEEE International Conference on Green Computing and Communications,* 2012, pp. 761-766.

[8] ApacheHadoop 2.7.2, Web: http://hadoop.apache.org/docs/ current/.

[9] J. Dean and S. Ghemawat, *MapReduce: Simplified Data Processing on Large clusters*, Google Inc., 2004. pp.1-13.

[10] K. Shim, “MapReduce algorithms for big data analysis”, *Databases in Networked Information Systems. Lecture Notes in Computer Science*, vol.7813, 2013, pp.44-48.

[11] Apache Storm, Web: http://storm.apache.org/.

[12] O. Santos, *Network Security with NetFlow and IPFIX: Big Data Analytics for Information Security*, Cisco Press, 2015, 320 p.

[13] K.M. Kavanagh, O. Rochford, T. Bussa, *2016 Magic Quadrant for SIEM. Gartner*, 10 August 2016.

[14] (2017, Jun.) HPE Security ArcSight ESM, Web: https://saas.hpe.com/en-us/software/siem-security-information-event-management.

[15] (2017, Jun.) IBM Security QRadar SIEM, Web: http://www-03.ibm.com/software/products/en/qradar-siem.

[16] (2017, Jun.) Alienvault OSSIM. [Online] Available: https:// www.alienvault.com/products/ossim.

[17] (2017, Jun.) Splunk Enterprise. [Online] Available: https://www.splunk.com/en_us/products/splunk-enterprise.html.

[18] (2017, Jun.) ManageEngine EventLog Analyzer. [Online] Available: https://www.manageengine.com/products/eventlog/.

[19] (2017, Jun.) Cisco Systems OpenSOC. [Online] Available: https://github.com/OpenSOC/.

[20] (2017, Jun.) Apache Metron, Web: http://metron.incubator.apache.org/.

[21] (2017, Jun.) GitHub Apache Metron. [Online] Available: https://github.com/apache/incubator-metron/pulls.

[22] (2017, Jun.) Graylog. [Online] Available: https://www.graylog.org/.

[23] (2017, Jun.) Documentation Graylog. [Online] Available: http://docs.graylog.org/en/2.2/pages/configuration/elasticsearch.html.

[24] (2017, Jun.) Elastic Stack. [Online] Available: https://www.elastic.co/.

[25] J. Chen, Z. Tang, K. Bilal, S. Yu, C. Weng, and K. Li, “A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment”, *IEEE Transactions on Parallel and Distributed Systems*, vol.28, Issue 4, 2017, pp.919-933.

[26] X. Shu, J. Smiy, D. Yao, H. Lin, “Massive Distributed and Parallel Log Analysis For Organizational Security”, *in IEEE Globecom Workshops*, December 2013, pp.194-199.

[27] Leveraging a Big Data Model in the Network Monitoring Domain. White Paper. VSS Monitoring. 2014.

[28] T. Dumitras, D. Shou, “Toward a Standard Benchmark for Computer Security Research: the Worldwide Intelligence Network Environment (WINE)”, *in BADGERS’11*, 2011, pp.89-96.

[29] P. Giura, W. Wang, “Using Large Scale Distributed Computing to Unveil Advanced Persistent Threats”, *Science Journal*, vol.1, no.3, 2013, pp.93-105.