Enterprise Service Bus Construction in SOA Architecture for SIEM Implementation in Critical Information Infrastructure

Sergiy Gnatyuk[1], Rat Berdibayev[2], Viktoriia Sydorenko[1], Artem Polozhentsev[1], and Myroslav Ryabyy[1]

[1] *National Aviation University, 1 Liubomyra Huzara ave., Kyiv, 03058, Ukraine*
[2] *Almaty University of Power Engineering and Telecommunication, 126/1 Baytursynuli str., Almaty, 050013, Kazakhstan*

Abstract
The number of cyber threats in ICT is increasing and the development of new security oriented instrumental tools is very important and relevant scientific task. Security incident and event management (SIEM) systems are category of such tools, directed on log analysis and incident management to prevent negative consequences minimize damage of cyber threats for end user. In the previous works authors have analyzed existed SIEM systems and database types for them as well as created new architecture of cloud-based SIEM. Next step of this research project is enterprise service bus architecture justification. The paper defines the place of distributed data bus in the concept of service oriented architecture, identifies the functions and benefits. Also authors analyzed most popular up-to-date enterprise service bus solutions and provides recommendations in context of developed SIEM implementation in the critical infrastructure. Besides, the data sheet for SIEM in critical infrastructure was formed and proposed in this paper.

Keywords
SIEM, incident management, ESB, cyber threat, cloud-based architecture, SOA.

1. Introduction

Nowadays, the number of cyber threats is increasing, this is due to the development of new information-communication technologies (ICT) and an insufficiently good level of testing of the developed software and physical software, as well as the lack of maintenance and support for outdated software and server software [1, 2]. There are various vulnerabilities in protocols, software, as well as the architecture of electronic equipment, which affects the cyber security level [3]. Many various instruments were developed to solve mentioned problems and mitigate threats. One of them is Security Information and Event Management (SIEM) [4], that was created to prevent the future consequences of the exploitation of vulnerabilities by undesirable persons, as well as to minimize damage for the end user [6].

2. Analysis of Modern Approaches and Problem Statement

In the paper [6] the basic concept of cloud-based SIEM architecture (Fig. 1) for different sectors of critical infrastructure was proposed. This scheme can also be integrated to real ICT infrastructures with existed SIEM (proposed by various vendors [5]) and other incident management instrumental tools. The main structural units of proposed SIEM are following:

- Horizontal Databases.
- Blocks of Analytics and Monitoring.
- Cloud Storage.
- Encryptor.
- Message Broker.
- Sources (System 1 – System N).

CPITS-2022: Cybersecurity Providing in Information and Telecommunication Systems, October 13, 2022, Kyiv, Ukraine
EMAIL: s.gnatyuk@nau.edu.ua (S. Gnatyuk); r.berdybaev@aues.kz (R. Berdibayev); v.sydorenko@ukr.net (V. Sydorenko); artem.polozhencev@gmail.com (A. Polozhentsev); m.riabyi@nau.edu.ua (M. Ryabyy)
ORCID: 0000-0003-4992-0564 (S. Gnatyuk); 0000-0002-8341-9645 (R. Berdibayev); 0000-0002-5910-0837 (V. Sydorenko); 0000-0003-0139-0752 (A. Polozhentsev); 0000-0002-9651-9135 (M. Ryabyy)

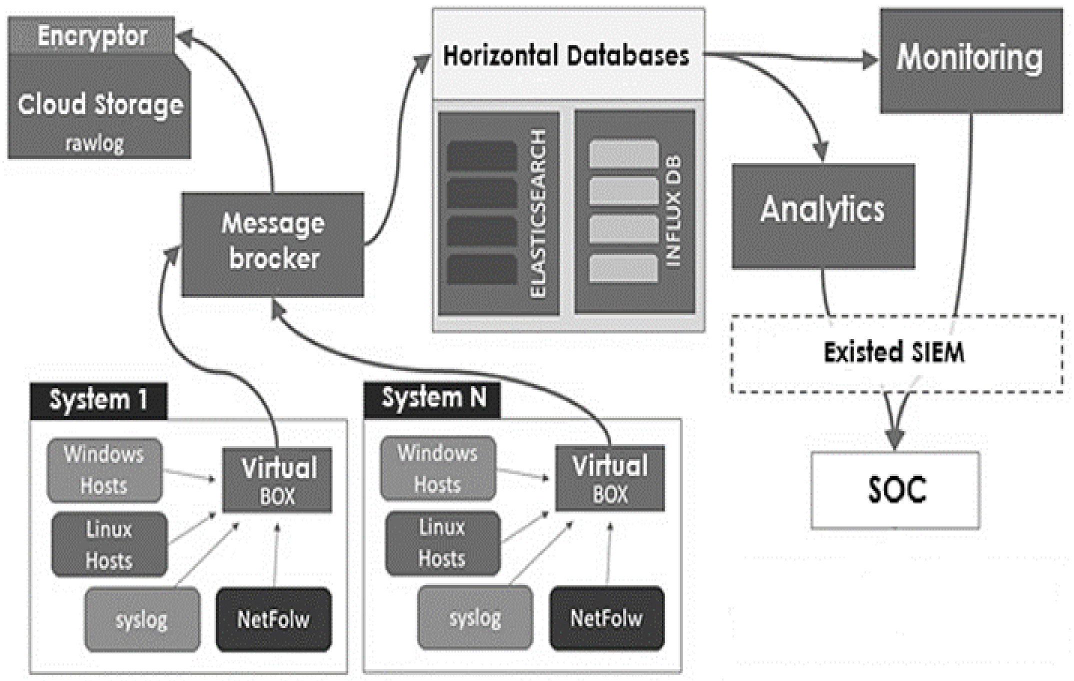

Figure 1: Proposed cloud-based SIEM architecture concept

One of the most important unit of this system is Encryptor, which creates single block Cloud Storage by providing confidentiality of the non-processed data after its gathering using syslog, NetFlow etc. Besides, the Virtual Box sends gathered and encrypted data Horizontal Databases via Message Broker. If there is no connection with Message Broker, the temporary data storage provides in Cloud Storage.

In [5] the analysis of up-to-date SIEM systems was carried out; in the paper [6] basic concept of cloud-based SIEM was developed; in the work [7] authors analyzed DB types in context of SIEM implementation. Next step is enterprise service bus justification. It will be the main objective of this research paper.

3. ESB Implementation in SOA Information Infrastructure

Service Oriented Architecture (SOA) provides a way to allow multiple usage of software components through service interfaces (Fig. 2) [8]. Such interfaces use common communication standards so they can be quickly integrated into new applications without the need for in-depth integration each time.

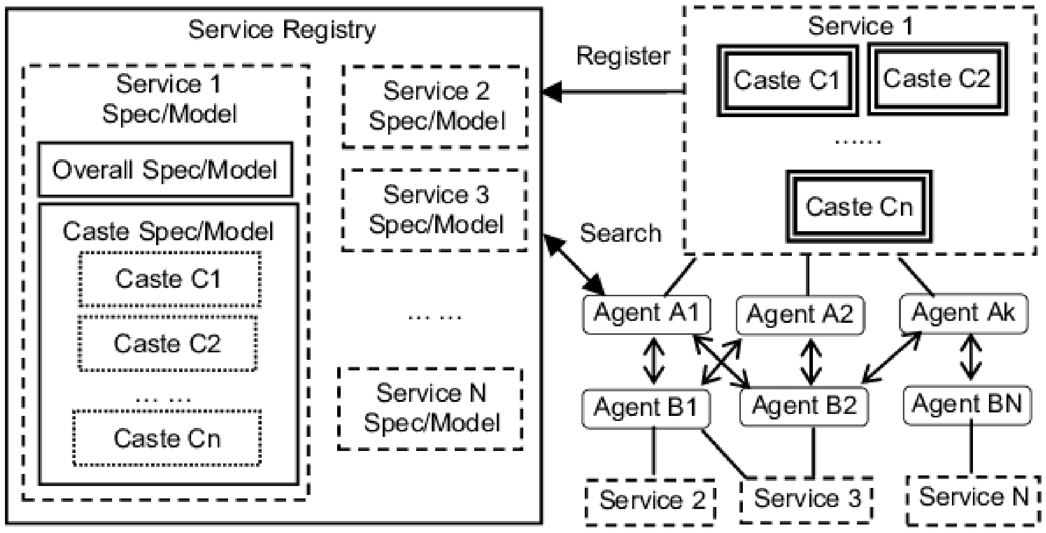

Figure 2: Scheme of the SOA construction

The SOA service contains the code and data integrations necessary to perform a particular business function [9]. Service interfaces are weakly interacted, meaning that they can be used even with minimal knowledge on how the integration is performed. Services are accessed via standard network protocols, such as SOAP/ HTTP or JSON/HTTP, which send read/modify data requests. Services are published in such a way that they allow developers to find and reuse them to build new applications quickly. These services can be created from the ground but are often made by exporting functions from existing systems as interfaces.

In SOA, services can interact with each other regardless of the type of service. This means that a particular service may be platform- or protocol-specific, but SOA allows such services to interact and exchange data. This data is exchanged through a distributed data bus or Enterprise Service Bus (ESB) [10], which forms the basis of any SOA architecture. Thus, the ESB is a template (Fig. 3) in which a centralized component integrates with core systems and then accesses these integrations as service interfaces.

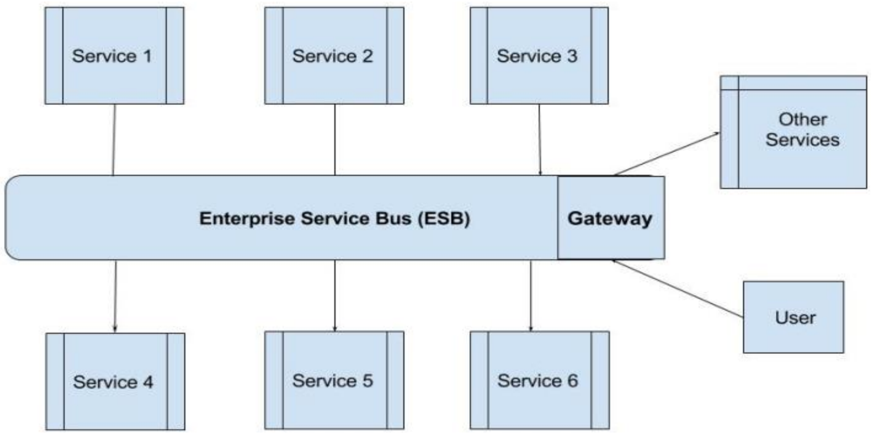

Figure 3: The ESB bus layout

It provides data model conversions, strong interaction, routing, and even multiple query creation, combining these functions in a single service interface that can be repeatedly used by new applications. Typically, the ESB template is implemented with a specially designed integration execution environment and tools that are well suited to perform the above functions as efficiently as possible [11].

In principle, SOA can be implemented without ESB (Fig. 4), but application owners would have to find a unique way to provide access to interfaces which is a very time-consuming task (even with multiple interfaces), and which will also make future maintenance very difficult. For example, ESB can be implemented by using JMS servers and XML/XSD as a means of transferring data between different services. Thus, different services will register or connect to these JMS servers and exchange data in XML format. Typically, the SOA suite comes with so-called adapters that help convert messages to and from a format the service and XML understands.

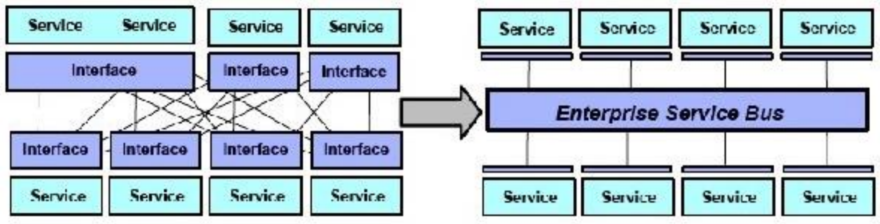

Figure 4: SOA Comparison with and without ESB

Consider a stock trading system as an example. Messages from the stock exchange come in using the FIX protocol. It’s possible to create an application that expects JSON. To make both systems working, SOA will be used—the FIX adapter will convert the FIX message into XML, then that xml will be passed to the JSON adapter via ESB, which is then converted into JSON as required by your endpoint system. In Fig. 5 is an example of a JBoss ESB implementation [9].

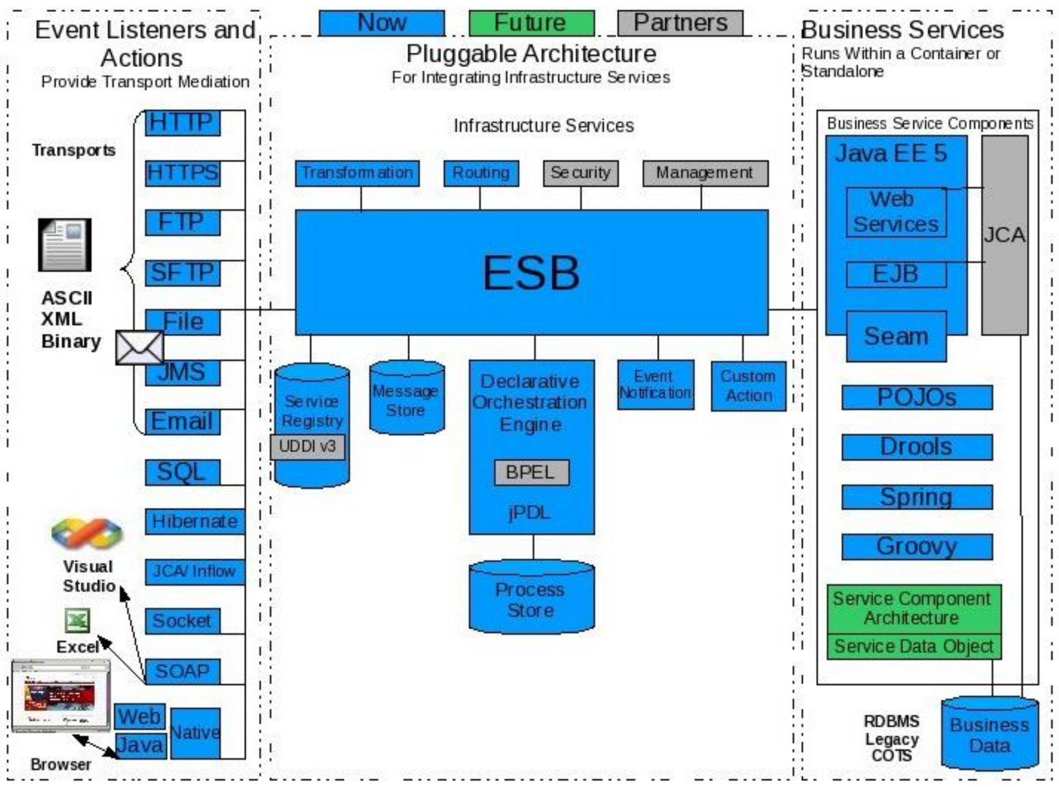

Figure 5: JBoss ESB implementation example

4. ESB Comparative Analysis

Below let's consider how ESB components are implemented and used in the solutions that are most often offered on the Kazakhstan market (Talend [12], Mule [13], WSO2 [14], Red Hat Fuse).

Table 1
ESB comparative analysis

No.	ESB / Criteria	Talend	Mule	WSO2	Red Hat Fuse
1	Studio	+	+	+	±
2	Message Broker Support	JMS 1.1, Microsoft MQ 3.0, JBoss Messaging 1.4.4, IBM MQ 8.0, Apache ActiveMQ 5.13.2	Anypoint MQ, IBM MQ, Apache Kafka, JMS 1.0.2, 1.1, 2.0 support	Amazon SQS, JMS support, Apache Kafka	Apache ActiveMQ, Apache Kafka, AWS MQ, RabbitMQ, JMS support
3	Logging	statistics on the execution of tasks and components, errors, warnings and exceptions at the task level, data flow within tasks; logging in Elastic, Apache Log4j, Apache Commons Logging, Trace Logs	logging within each integration created in Mule: errors and events mandatory for logging by integration logic; logging starting, stopping, deploying, and disconnecting Mule services and integrations	Apache Log4j-based logging via the Apache Commons Logging library. System and component events are logged separately	Apache Log4j-based logging via Apache Commons Logging library, SLF4J, java.util.logging, Elastic
4	Monitoring	+	+	+	+

Developers also add Apache Kafka (plus Kafka Connect) and RabbitMQ message brokers to the list, but these two solutions are not ESB, and it is not reasonable to consider them within the scope of this analysis. As criteria let's choose the basic functional components of data buses: whether the studio is, Message Broker Support, the way of logging and monitoring.

5. Features of Modern ESB

The data bus is a set of software that acts as a single hub for the exchange of messages between information systems and applications. The service bus allows for easy configuration of message paths, stores the history of messages, and records the path of each message. The basic principles of it are below:

1. Any upgrade of an item inevitably requires a large-scale reworking of the integrations. For example, an Oracle database update is released, and all the integrations related to it have to be reworked.
2. Event logging in each of the integrations is implemented differently (if implemented). If data is lost or comes in with errors, it will be difficult to trace the time and cause of the error.
3. Each new element of the system requires a significant investment in Point-to-Point integration. For example, to add new trading platforms, it will be necessary to integrate them with an online store, CRM, WMS, ERP, PIM, etc.
4. Business intelligence is becoming more complex: data is stored in different sources, in different formats, or duplicated. Combining them into a convenient tool for making management decisions is a difficult task.
5. As the infrastructure increases, the time and resources required to maintain it increase accordingly. Also, the reserve of resources to improve its work decreases.
6. ESB-bus combines a few functions, which a "star" topology distributes across integrations or not implements.
7. ESB collects information from other systems: either related to the company's IT infrastructure or external. The information is received in the form and formats in which it is contained in the source system.
8. Within the ESB, the data are converted into the required formats for transmitting to other systems.
9. The logic of routes and conversions is set by the operator: the source of information, the purpose of conversion and the place of acceptance.
10. Logs are saved in the message broker. If errors or losses occur, it will be possible to determine the cause of the failure without having to repeat the incident. Accordingly, errors can be corrected, and data restored quickly and easily.

6. Practical Implementation of ESB for Effective Integration of SIEM in Critical Information Infrastructure

The developed platform [6, 15] applies a security event correlation mechanism that allows the platform to be used as a dedicated and fully functional SIEM system. The platform provides correlation for normalized logs/events, searches/queries for threat analysis and sources of vulnerability factor information and produces risk-aware alarms. The primary objective is to collect as many events as possible from the organization's infrastructure [16].

During the digital transformation, companies (regardless of their size) use multiple information systems. Frequently, they operate overlapping data arrays. ESB is designed to integrate different information systems. The data exchange takes place via ESB, using various protocols and formats, allowing to avoid modifications of the systems being integrated. The use of ESB for SIEM-class systems is aimed at the balanced distribution of the load on services and security of data exchange.

Consider how services can communicate directly with each other. To retrieve data from an application, it is necessary to go through a complex multi-level chain of operations. There can be several to dozens or hundreds of such services. The continuous exchange of messages between systems can create a heavy load. On the user side, it will lead to long latency times and constant application failures (Fig. 6). It is also worth noting that if one of the systems needs to be updated, changed, or distributed to other departments, this will inevitably affect all other services.

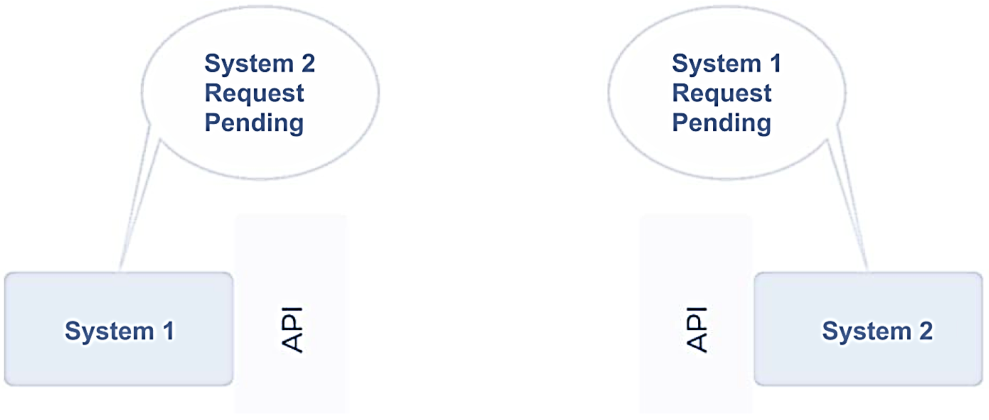

Figure 6: Interaction of services without a bus

Using ESB for SIEM-class systems completely changes the organization of processes in a company. Namely, applications no longer need to communicate directly with each other, instead, each of them interacts only with the integration platform. This instantly eliminates the need for a huge number of accesses methods, as many interfaces as there are services will be needed. If changes need to be made to one of the systems, it will not affect the other corporate applications. The ESB will single-handedly take on all these tasks (Fig. 7). Thus, this approach, unlike traditional point-to-point architecture (where services interact directly with each other), has more flexibility. Integration scenarios can be modified with minimal developer intervention. The benefits of the solution are as follows: makes it easier to integrate applications by implementing ESB for SIEM-class systems saves time and resources, improves the functioning of services, and enhances the organization's efficiency and security.

To collect information (events) the system uses its agents installed in monitored subsystems, as well as standard existing mechanisms for collecting events (syslog, snmp, etc.). For network control, it can be used as a collector of NetFlow statistics received from network equipment. It can also be used to analyze network traffic either by mirroring traffic from network equipment or by sending traffic through itself.

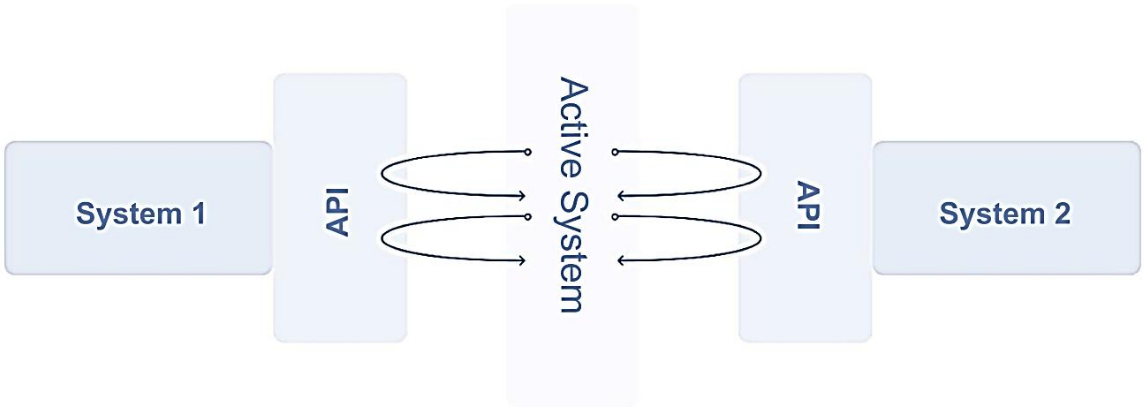

Figure 7: Service interaction with the bus

The system sends events over an encrypted channel to the message broker. If there is no connection to the message broker, it ensures temporary storage of data, minimizing the risks of losing critical information. Multiple brokers can be installed in a monitored system. Some important concepts:

- Message broker should be understood as a special software which ensures the guaranteed delivery of messages from multiple sources to multiple recipients. This is an electronic queue for messages.
- Repository is a special storage of unprocessed records in encrypted form. An important part for collecting legally relevant evidence for incident investigations.

Horizontally extensible databases are a distinctive architectural advantage of the developed platform. The system uses distributed databases of different types to solve metrics control (monitoring) and event control (SIEM) tasks in parallel [17]:

- High speed of processing large streams of information.
- Minimal delays in data processing.
- Minimal delays for building analytical reports and queries.
- High fault tolerance.
- Storage expandability by adding nodes without database downtime.

A monitoring module is comprehensive software for controlling metrics. Usually, such tasks are called "Monitoring" and are real-time tracking of quantitative metrics of systems. When metrics enter risk zones, the module creates a security violation event. This module has an interactive graphical interface.

The analytics module is comprehensive event analysis software that performs normalization, correlation, and event analysis. It also finds dependencies, defines an event as an incident, and informs other systems. Also, it has an interactive graphical interface.

The service-oriented architecture, of which the developed platform is a part, integrates all APIs, which ensures end-to-end integration. An API is a so-called set of rules and conditions for programs to communicate with each other: input and output data, and types of operations. The use of an API significantly simplifies interaction: it ties together the capabilities of different services, forming interfaces that are accessible to different users (Fig. 8).

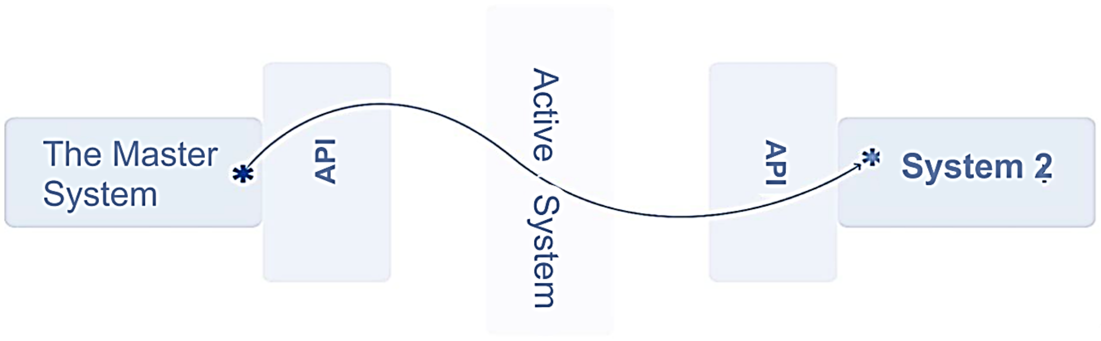

Figure 8. SOA service to bus interaction

Microservice architecture differs from the traditional ESB approach for SIEM-class systems because its functionality is organized into small services, each of them is responsible for a separate task, is supported by one team, and can work in isolation from the others. There is no centralized base with this approach. Each service has its repository of information. The ESB for SIEM-class systems, however, serves only as a transport, being, in essence, just a message broker. Interaction between the user and the platform services is also performed via API [18].

Taking to account [5–7] SIEM data sheet well be following (Table 2).

Additional requirements are following:

- The Supplier ensures the installation of SIEM software on physical servers and/or the Customer's virtualization platform.
- The supplier provides the Customer with a calculation of the resource requirements for installing SIEM. The calculation is made by the Supplier based on the performance requirements specified in this specification (clauses 11.12 in Table 2).
- The customer provides the allocation of resources in accordance with the specified calculation, installation and basic configuration of operating systems (including configuration of the disk subsystem and network interfaces) for installing SIEM. The Supplier shall provide the Customer with an operating system distribution (on media or in the form of a download link) and, if necessary, a license for operating systems.
- The customer provides access from servers to the Internet during the installation and configuration of SIEM.
- For the entire duration of the SIEM operation, the customer provides constant access from the SIEM servers to the licensing server to control the license.

The Supplier, within 10 (ten) calendar days, completes the installation and configuration of the SIEM system.

Table 2
Data sheet information security and event management system

N	Name	Description
		Information security event monitoring system and real-time incident detection (SIEM)
1	Special purpose	The IS event monitoring system and real-time incident detection (hereinafter referred to as SIEM) is designed to monitor and analyze IS events and must: • Carry out centralized collection, storage and processing of events of system logs (logs), as well as network flows from various systems of the Customer's infrastructure. • Identify important events and IS incidents in the total mass of data, which should allow the Customer's IS specialists (hereinafter referred to as the Customer's personnel) to concentrate on the most serious incidents and respond to them in a timely manner. • Inform the Customer's personnel about identified information security incidents by sending messages to e-mail.
2	Centralized Management	SIEM should provide centralized management of all its components and functionality through a single graphical Web interface.
3	Data visualization (Dashboards)	SIEM should: • Allow the creation of graphical panels (dashboards) using any events, with automatic updating at a given interval. • Support the creation of new graphical panels or modification of existing ones using a "wizard," by method that does not require the use of programming languages. • Allow the saving of graphical panels for collective use. The graphic panels of the module must support various types of data presentation: tables, pie and line charts, etc. The graphic panels of the module should function automatically, without the need for regular maintenance by the operator. • Display graphical panels via WEB. • Interface.
4	API support	SIEM should: have an open software interface API for possibility integration with other Modules.
5	Support for authentication and authorization	SIEM should support the following methods to provide user authentication and authorization: • Local user base. • Active Directory. • LDAP. • Tokens (for API access).
6	Update	The SIEM must support the ability to automatically and/or manually update as new versions are released.
7	Fault Tolerance	The SIEM database must be able to support a cluster organization in the amount of at least two nodes (node)
8	Scaling	SIEM should: ▪ Provide horizontal scaling by adding hardware and, if necessary, purchasing additional licenses for SIEM in accordance with the current (at the time of scaling) licensing policy. ▪ Have an event storage component (database) in which the following functions are implemented: ● Scaling without a fixed limit on the volume of event storage (adding additional equipment if necessary). ● Fault-tolerant implementation.

9	Collecting and filtering events	SIEM should: • Support standard methods for collecting event logs: Syslog, Raw/Plaintext, GELF, CEF, file event logs (using agents for Linux/Windows). • Provide analysis of events in real time. • Provide filtering, as well as display through the user interface of an event in real time, where the user can immediately apply filters. • Be able to save search criteria for quick access in the future. • Support search by events using the query language (if you use your own query language, it should be described in the documentation). • Provide the user with the opportunity to independently connect event sources that are not supported by default or systems of their own design. • Support data transmission from sources to the control system via a secure channel (if there is support for secure transmission in the protocol). • Support centralized management of agents through the SIEM interface (for agents of the Beats family).
10	Account Management Requirements	SIEM should: • Support a role-based management model with a predefined set of roles. • Be able to create and use User Groups (Teams). • Have a token management system for authorization in the API.
Performance Requirements		
11	Events per second requi-rements (EPS)	• Average daily—no more than 200 EPS • Maximum in the busiest hour—no more than 400 EPS
12	Requirements for informa-tion stored in the database	The daily amount of information stored in the event database is no more than 4 GB per day. The storage period for events in the database and/or SIEM archive is at least 3 years.

7. Conclusions

The basis of any SOA architecture is the ESB, the main advantages of which are a wide range of connectors and scalability of the solution; flexible data routing; guaranteed delivery of information messages; organization of a secure transmission channel; centralized management; ability to monitor and diagnose the state of transmission; possibility of integration with third-party message queues.

The analysis of modern ESB solutions has shown that each of the products has its features, which form the basis of their fields of usage. If a company wants to use free versions of the product, Fuse would be the most suitable option (but it would be necessary to consult developers for significant revisions). Talend or Mule are good options in the early stages of a company's development. WSO2 has the best balance of functionality and ease of calculating the cost of the license.

To implement SIEM in a critical information infrastructure, a SOA-based distributed data bus is required. The platform uses distributed databases of different types to solve metrics and event control tasks in parallel. This increases parameters by an order of magnitude, providing processing speed of large flows of information; minimal delays for data processing; minimal delays for analytical reports and queries; high fault tolerance; storage extensibility by simply adding nodes without database downtime. The use of API significantly simplifies interaction: it ties together the capabilities of different services, forming interfaces available to different users.

After concept and DB development, the ESB was justified as well as data sheet for SIEM in critical infrastructure was formed and proposed in this paper.

8. Acknowledgment

This work is carried out within the framework of research grant #AP06851243 “Methods, models and tools for security events and incidents management for detecting and preventing cyber-attacks on critical infrastructures of digital economics” (2020–2022), funded by the Ministry of Digital Development, Innovation and Aerospace Industry of the Republic of Kazakhstan.

9. References

[1] V. Grechaninov, et al., Decentralized Access Demarcation System Construction in Situational Center Network, in Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3188, no. 2, 2022, pp. 197–206.

[2] V. Grechaninov, et al., Formation of Dependability and Cyber Protection Model in Information Systems of Situational Center, in Emerging Technology Trends on the Smart Industry and the Internet of Things, vol. 3149, 2022, pp. 107–117.

[3] V. Buriachok, V. Sokolov, P. Skladannyi, Security rating metrics for distributed wireless systems, in 8th International Conference on “Mathematics. Information Technologies. Education:” Modern Machine Learning Technologies and Data Science (MoMLeT and DS), vol. 2386, 2019, pp. 222–233.

[4] A. Skendžić, B. Kovačić, B. Balon, Management and Monitoring Security Events in a Business Organization—SIEM system, in 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), 2022, pp. 1203–1208, doi: 10.23919/mipro55190. 2022.9803428.

[5] S. Gnatyuk, et al., Modern SIEM Analysis and Critical Requirements Definition in the Context of Information Warfare, in CEUR Workshop Proceedings, 2021, vol. 3188, pp. 149–166.

[6] R. Berdibayev, et al., A Concept of the Architecture and Creation for SIEM System in Critical Infrastructure, Studies in Systems, Decision and Control, vol. 346, 2021, pp. 221–242.

[7] S. Gnatyuk, et al., Modern Types of Databases for SIEM System Development, CEUR Workshop Proceedings, vol. 3187, 2021, pp. 127–138.

[8] Z. Jin, H. Zhu, A Framework for Agent-Based Service-Oriented Modelling, 2008 IEEE International Symposium on Service-Oriented System Engineering, 2008, pp. 160–165, doi: 10.1109/SOSE.2008.15.

[9] W. Li, Design and Implementation of Software Testing Platform for SOA-Based System, in IEEE 6th Int. Conf. on Comp. and Commun. Syst., 2021, pp. 1094–1098, doi: 10.1109/icccs52626.2021.9449221.

[10] ESB (Enterprise Service Bus), https://www.ibm.com/cloud/learn/esb.

[11] P. Dai, Design and implementation of ESB based on SOA in power system, in 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011, pp. 519–522, doi: 10.1109/drpt.2011.5993946.

[12] J. Sreemathy, et al., Data Integration in ETL Using TALEND, in 6th International Conference on Advanced Computing and Communication Systems, 2020, pp. 1444–1448, doi: 10.1109/ICACCS48705.2020. 9074186.

[13] X. Mkhwanazi, H. Le, E. Blake, Clustering between Data Mules for Better Message Delivery, in 26th Int. Conf. on Advanced Information Networking and Applications Workshops, 2012, pp. 209–214.

[14] I. Kumara, C. Gamage, Towards Reusing ESB Services in Different ESB Architectures, in IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010, pp. 25–30, doi: 10.1109/compsacw.2010.15.

[15] S. Gnatyuk, et al., Cloud-Based Cyber Incidents Response System and Software Tools, Communications in Computer and Information Science, vol. 1486, 2021, pp. 169–184.

[16] T. Laue, et al., A SIEM Architecture for Multidimensional Anomaly Detection, in 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2021, pp. 136–142, doi: 10.1109/IDAACS53288.2021.9660903.

[17] P. Asef, et al., SIEMS: A Secure Intelligent Energy Management System for Industrial IoT applications, in IEEE Transactions on Industrial Informatics, doi: 10.1109/tii.2022. 3165890.

[18] M. Orsós, et al., Log Collection and SIEM for 5G SOC, in IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2022, pp. 147–152, doi: 10.1109/sami54271.2022.9780759.