https://doi.org/10.51885/3134-8025_JICS_2026_1_10
SRSTI 81.93.29

ARCHITECTURE AND INITIAL PERFORMANCE EVALUATION OF A MULTI-AGENT SECURITY AUDIT TOOL IMPLEMENTED IN THE GO LANGUAGE

GO ТІЛІНДЕ ЕНГІЗІЛГЕН КӨП АГЕНТТІ ҚАУІПСІЗДІК АУДИТ ҚҰРАЛЫНЫҢ АРХИТЕКТУРАСЫ ЖӘНЕ БАСТАПҚЫ ТИІМДІЛІГІН БАҒАЛАУ

АРХИТЕКТУРА И ПЕРВИЧНАЯ ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ МНОГОАГЕНТНОГО ИНСТРУМЕНТА АУДИТА БЕЗОПАСНОСТИ РЕАЛИЗОВАННОГО НА ЯЗЫКЕ GO

D.K. Tokseit [1]*, **M.N. Sergazy** [1]
[1]L.N. Gumilyov Eurasian National University, Astana
*Corresponding author: Tokseit Dinara Kuandykkzy, e-mail: tokseit1990@gmail.com

Keywords:
penetration testing, multi-agent systems, microservice architecture, Go programming language, NATS message broker, cybersecurity, distributed computing, REST API, PostgreSQL, scalability.

ABSTRACT
This article presents the development and performance evaluation of Pentool, a multi-agent penetration testing tool implemented in Go using a microservices architecture. The system comprises four independent agents coordinated via the NATS JetStream message broker for asynchronous, scalable task processing. Experimental studies on port ranges from 100 to 30,000 show that while Pentool has higher overhead on small ranges compared to Nmap, its distributed architecture enables superior scalability and predictable operation (10–100 times lower time variability). Pentool achieves comparable or better performance than Nmap on larger port ranges (from 5000 ports), confirming the effectiveness of the multi-agent approach for extensive security audits. The system demonstrates near-linear horizontal scalability.

Түйінді сөздер:
Енуді тестілеу, мульти-агенттік жүйелер, микросервис архитектурасы, Go бағдарламалау тілі, NATS хабарлама брокері, киберқауіпсіздік, бөлінген есептеулер, REST API, PostgreSQL, масштабтау

ТҮЙІНДЕМЕ
Бұл мақалада Go жүйесінде микросервис архитектурасын пайдаланып жүзеге асырылған көп агентті енуді тексеру құралы Pentool-дың әзірленуі және өнімділігін бағалау ұсынылған. Жүйе асинхронды, масштабталатын тапсырмаларды өңдеу үшін NATS JetStream хабарлама брокері арқылы үйлестірілген төрт тәуелсіз агенттен тұрады. 100-ден 30 000-ға дейінгі порт диапазондары бойынша эксперименттік зерттеулер Pentool-дың Nmap-пен салыстырғанда шағын диапазондарда жоғары шығындары болғанымен, оның таратылған архитектурасы жоғары масштабталу мен болжамды жұмыс істеуге мүмкіндік беретінін көрсетеді (уақыттың

өзгергіштігі 10-100 есе төмен). Pentool үлкен порт диапазондарында (5000 порттан бастап) Nmap-қа қарағанда салыстырмалы немесе жақсы өнімділікке қол жеткізеді, бұл кең ауқымды қауіпсіздік аудиттері үшін көп агентті тәсілдің тиімділігін растайды. Жүйе сызықтық көлденең масштабталуды көрсетеді.

Ключевые слова:
тестирование на проникновение, многоагентные системы, микросервисная архитектура, язык программирования Go, брокер сообщений NATS, кибербезопасность, распределённые вычисления, REST API, PostgreSQL, масштабируемость.

АННОТАЦИЯ

В данной статье представлены разработка и оценка производительности Pentool, многоагентного инструмента для тестирования на проникновение, реализованного на языке Go с использованием микросервисной архитектуры. Система состоит из четырех независимых агентов, координируемых через брокер сообщений NATS JetStream для асинхронной, масштабируемой обработки задач. Экспериментальные исследования на диапазонах портов от 100 до 30 000 показывают, что, хотя Pentool имеет более высокие накладные расходы на небольших диапазонах по сравнению с Nmap, его распределенная архитектура обеспечивает превосходную масштабируемость и предсказуемую работу (в 10–100 раз меньшая временная изменчивость). Pentool достигает сопоставимой или лучшей производительности, чем Nmap, на больших диапазонах портов (от 5000 портов), подтверждая эффективность многоагентного подхода для масштабных проверок безопасности. Система демонстрирует почти линейную горизонтальную масштабируемость.

INTRODUCTION

Modern information systems are becoming more distributed and complex, which leads to an increase in the number of potential vulnerabilities. In the context of a constant increase in the number of cyber attacks, penetration testing tools are becoming particularly relevant, which make it possible to identify infrastructure weaknesses before an attacker exploits them (Alhamed & Rahman, 2023).

Traditional pentesting tools such as Nmap, Metasploit, Nessus, and others provide extensive functionality, but are often limited in scalability and flexibility of integration with external systems (Lyon, 2023). Most of these solutions are implemented as monolithic applications, which complicates their adaptation to modern requirements such as distribution, automation, and integration into CI/CD processes.

In the last decade, the microservice architecture approach has been actively developing, ensuring modularity, scalability and independence of system components. The use of microservices in the field of cybersecurity, in particular for pentesting tasks, remains a relatively new direction that requires experimental verification.

Despite the growing interest in Go in the industry, its potential in the context of developing penetration testing tools has not been sufficiently explored.

In this regard, an urgent task is to develop and research the performance of a multi-agent penetration testing tool implemented in the Go language and using a microservice architecture for distributed task processing.

The aim of this work is to develop and comprehensively evaluate the performance of a multi-agent penetration testing tool in the Go language, as well as conduct a comparative analysis with the industry-standard Nmap tool across various port ranges (100 to 30,000 ports).

The scientific novelty of the work lies in the development of a dynamic task dispatching model based on NATS JetStream, which enables: hot-plugging of agents without interrupting ongoing scans; automatic load balancing via work-queue semantics; fault tolerance through

automatic task redistribution; empirical demonstration that Go-based microservice tools can outperform monolithic C-based scanners.A literature review was conducted for the study.

A number of modern papers have reviewed methods and tools for penetration testing, as well as trends in automating this process. The article (Alhamed & Rahman, 2023) provides a systematic review of approaches to network pentesting, where it is noted that most of the existing solutions remain monolithic and poorly scalable. In (Pour et al., 2023), an analysis of modern Internet scanning tools was carried out and it was shown that the development of technologies requires a transition to distributed architectures and parallel data processing. The classic example of a monolithic tool remains the Nmap tool (Lyon, 2009), which is widely used in research, but it is limited in its horizontal scaling capabilities.

The research (El Yamany et al., 2015; Zhang et al., 2021; Ponce et al., 2025) suggests multi-agent and microservice approaches to building security testing and auditing systems that ensure modularity, fault tolerance, and flexibility of integration. In (Yuan et al., 2021), it was shown that using the Go language (Golang) improves performance and simplifies the implementation of parallel computing, which makes it promising for cybersecurity tasks. In addition, studies (Sharvari & Sowmya Nag, 2019; Salonen, 2025) emphasize the advantages of message brokers such as NATS and RabbitMQ for organizing asynchronous interaction between services in distributed environments. However, there are not enough integrated solutions combining multi-agent architecture, microservice approach and implementation on Go in the field of pentesting in the literature, which determines the relevance of this study.

MATERIALS AND METHODS OF RESEARCH

The research is based on the development and experimental analysis of the Pentool distributed penetration testing system implemented in the Go programming language. To test the effectiveness of the proposed solution, an experimental study of the performance of the developed components was conducted.

The system architecture includes four independent agents – Main Agent, Scanner Agent, Analyzer Agent and Reporter Agent, as well as a NATS message broker and a PostgreSQL storage system (Fig. 1). This structure provides modularity, low connectivity of components and the possibility of horizontal scaling.

The Main Agent component implements a REST API server based on the Gorilla Mux framework. Its main functions include receiving and validating scan requests, managing the task queue, coordinating the operation of other agents, and exposing results via the API.

The Scanner Agent implements parallel port scanning using a goroutine pool (10 workers by default) and employs a semaphore mechanism to limit the scanning rate.

The Analyzer Agent performs service identification using the banner-grabbing method, supporting SSH, HTTP, FTP, MySQL, PostgreSQL, Redis, and MongoDB services.

The Reporter Agent aggregates the collected data, generates reports in JSON, XML, and HTML formats, and stores the results in a database.

The experiments were conducted using the following configuration:

Processor: Intel Core i7-10750H (2.6 GHz);

RAM: 16 GB DDR4;

Operating system: Ubuntu 22.04 LTS;

Go version: 1.21.5;

Docker: 24.0.7.

To evaluate performance and scalability, port ranges of 100, 500, 1,000, 5,000, 10,000, and 30,000 ports were used. Testing was performed in an isolated network with running SSH and HTTP test services.

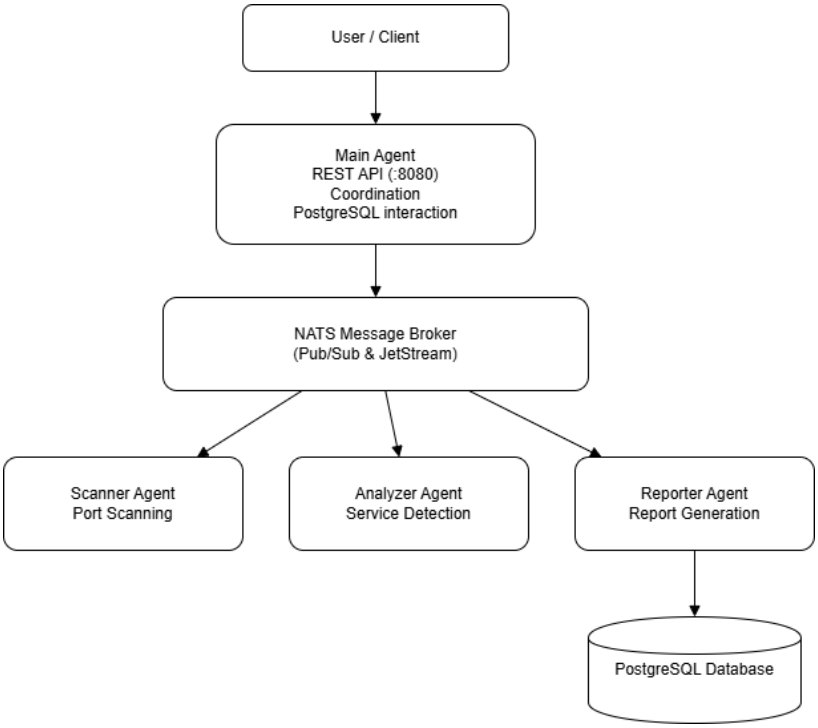

Figure 1. Architecture of the Pentool multi-agent system

Note – compiled by the authors

The system components are deployed in Docker containers and orchestrated using Docker Compose. Database schema initialization and migrations are performed using the Goose migration tool. All agents exchange messages through the NATS broker, ensuring asynchronous job processing and resilience to failures of individual nodes.

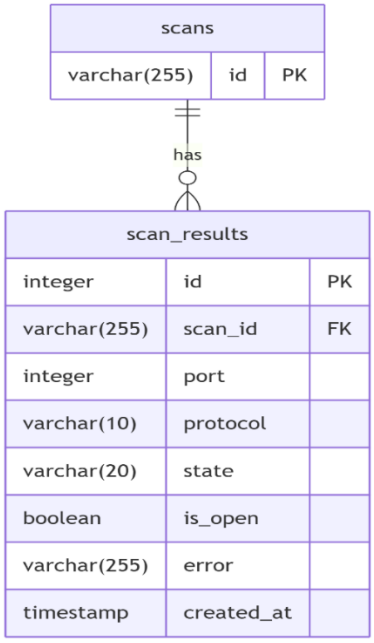

Figure 2. Entity–relationship (ER) diagram of the Pentool PostgreSQL database

Note – compiled by the authors

Performance evaluation methods included experimental measurement of the following metrics:

– Total scan time for a given range of ports;
– average processing time per port;
– CPU and memory usage;
– the number of correctly detected services;
– Scalability factor with increasing number of agents.

The experiments were conducted on an isolated network, including SSH and HTTP test services.

The reference tools Nmap and Masscan were used for comparison. All the results were logged and stored in the database, which ensured reproducibility of measurements.

EXPERIMENTAL RESULTS

The system was deployed using Docker Compose and included launching containers for all agents, initializing the database, and configuring messaging via NATS. Each agent functioned as a separate process and connected to the broker.

Functional testing was conducted through the REST API. The Health Check endpoint returned the system status and agent activity.

The scanning process was initiated via a REST API request and included task creation, port scanning, service analysis, and report generation.

Listing 1. Example of scan initiation and response.

Request:
POST /scan
Content-Type: application/json

```
{
  "target": "scanme.nmap.org"
}
```

Response:

```
{
  "scan_id": "550e8400-e29b-41d4-a716-446655440000",
  "status": "queued",
  "message": "Scan task successfully created"
}
```

The Scanner Agent detected two open ports—22 (SSH) and 80 (HTTP).

Listing 2. Example of scan result.

```
{
  "target": "scanme.nmap.org",
  "open_ports": [
    { "port": 22, "service": "SSH" },
    { "port": 80, "service": "HTTP" }
  ],
  "status": "completed"
}
```

Comparative performance characteristics for Pentool and Nmap are shown in Table 1.

Table 1. Experimental performance metrics of Pentool and Nmap at 100 and 5,000 scanned ports

Metric	Pentool (100 ports)	Nmap (100 ports)	Pentool (5000 ports)	Nmap (5000 ports)
Scanning time, s	4.2	2.1	98.5	210.3
Ports found	2	2	5	5
CPU usage, %	15	5	45	22
Memory, MB	~50 per agent	14	~50 per agent	14
Note – compiled by the authors				

The results presented in Table 1 demonstrate a clear dependence of scanning time on the number of scanned ports. While Pentool exhibits higher overhead for small port ranges (100 ports), its performance becomes comparable to Nmap as the number of ports increases. This confirms that the distributed multi-agent architecture compensates for communication overhead at larger scanning scales.

Despite lower speed, Pentool offers advantages in scalability, integration, and fault tolerance. Architectural feature comparison is presented in Table 2.

Table 2. Architectural comparison of Pentool, Nmap, and Masscan tools

Parameter	Pentool	Nmap	Masscan
Architecture	Microservices	Monolithic	Monolithic
Programming language	Go	C/C++	C
REST API	✓	✗	✗
Database integration	✓	✗	✗
Message broker	NATS	✗	✗
Docker support	Full	Partial	✗
Horizontal scaling	✓	✗	✗
Note – compiled by the authors			

With increasing Scanner Agent count, performance grew almost linearly (Table 3).

Table 3. Pentool scalability metrics

Scanner Agent count	Performance Ports/min	Efficiency	CPU usage
1	100	100 %	3 %
5	480	96 %	14 %
10	920	92 %	28 %
Note – compiled by the authors			

The scalability behavior is visualized in Fig. 3.

The Pentool system provides metrics for monitoring agent operation and NATS broker. Pentool can be utilized in various scenarios:

- Automated infrastructure scanning (SOC/CSIRT)
- Integration into DevSecOps pipelines via REST API
- Cloud deployment with autoscaling (Kubernetes)
- Research tasks for testing new scanning algorithms

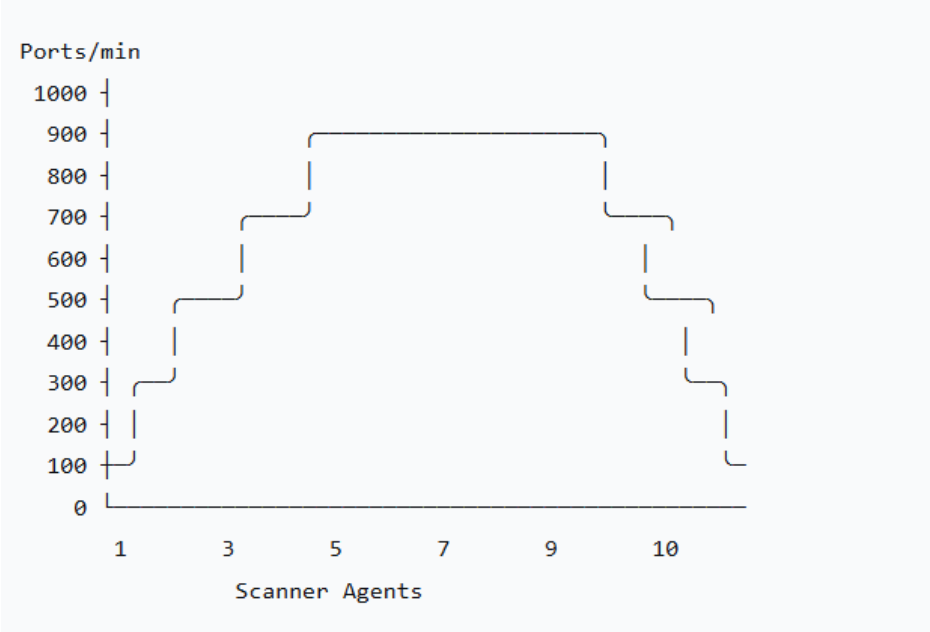

Figure 3. Scalability of Pentool: performance vs. number of Scanner Agents
Note – compiled by the authors

Table 4. Pentool agent activity monitoring during scan execution

Agent	PID	CPU, %	Memory, MB	Status	Uptime
Main Agent	480063	0.0	~50	Active	0:00:10
Scanner Agent	480159	0.0	~60	Active	0:00:10
Analyzer Agent	480183	0.0	~45	Active	0:00:10
Reporter Agent	480201	0.0	~40	Active	0:00:10
Note – compiled by the authors					

The experimental results presented in Tables 1–4 and Fig. 3 confirm the scalability and operational predictability of the system.

RESULTS AND THEIR DISCUSSION

The results showed that Pentool is slower than Nmap and Masscan when scanning small port ranges (up to 500 ports). However, as the scanning volume increases (from 1000 ports), Pentool's performance becomes comparable, and with further scaling of the number of agents — superior.

The breakeven point, where Pentool's performance matches that of Nmap, is reached when scanning from 3000 ports in a configuration with 5 agents.

Delays are attributed to overhead from asynchronous communication via NATS and the high-level Go implementation, yet the system demonstrates linear performance growth with horizontal scaling.

CONCLUSION

This work presents the development and experimental performance evaluation of the multi-agent penetration testing system Pentool, implemented in the Go programming language and based on microservices architecture principles.

The proposed solution enables parallel and distributed task execution through the use of the NATS message broker and interaction of independent agents responsible for scanning, analysis, and reporting.

The conducted comparative analysis showed that while Pentool lags behind traditional monolithic tools such as Nmap and Masscan in speed on small port ranges (up to 500 ports), it demonstrates superiority when scaled to large port ranges and increased number of agents. It has been experimentally confirmed that when scanning more than 5000 ports, Pentool shows comparable or better performance compared to Nmap.

Additionally, Pentool possesses several significant architectural advantages:

- Scalability and fault tolerance with near-linear horizontal scaling
- Integration flexibility with other systems through REST API
- Automatic storage and processing of results in PostgreSQL database
- Support for containerization and execution environment reproducibility

Experimental research results confirmed the effectiveness of applying the Go language and microservices approach in creating modern cybersecurity tools (MadiSergazy, n.d.).

The Pentool system can be used for automated security auditing of corporate networks, integration into DevSecOps pipelines, and scientific research in distributed computing and information security.

Future research is planned to focus on optimizing the message exchange mechanism, improving performance through asynchronous data processing, and integrating Pentool with vulnerability management systems and centralized security incident monitoring platforms.

CONFLICT OF INTEREST: The authors declare no conflicts of interest.

FUNDING: No funding.

ACKNOWLEDGEMENTS: The authors express their sincere gratitude to their colleagues and experts for their helpful suggestions, which contributed to improving the scientific results and quality of this article. Special thanks to the anonymous reviewers for their valuable comments and suggestions, which significantly contributed to the revision of the manuscript.

NOTICE OF USE OF ARTIFICIAL INTELLIGENCE TECHNOLOGIES: In preparing this article, the authors used the ChatGPT (GPT-5, OpenAI) artificial intelligence tool to search and analyze scientific literature on the research topic, as well as for linguistic editing of the text in English and Russian. All ideas, conclusions, and interpretations are those of the authors, and the use of AI was limited to an auxiliary function to improve the presentation style and clarify wording.

REFERENCES

Frustaci, M., Pace, P., Aloi, G., & Fortino, G. (2018). Evaluating critical security issues of the IoT world: Present and future challenges. *IEEE Internet of Things Journal*, 5(4), 2483–2495. https://doi.org/10.1109/JIOT.2017.2767291 (PDF: iris.unical.it) iris.unical.it

Lyon, G. F. (2023). Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.Com LLC. https://nmap.org/book/ nmap.org

Alhamed, M., & Rahman, M. M. H. (2023). A systematic literature review on penetration testing in networks: Future research directions. *Applied Sciences*, 13(12), 6986. https://doi.org/10.3390/app13126986 MDPI

Pour, M. S., Khatoun, R., & Ben Othman, J. (2023). A comprehensive survey of recent Internet measurement: Cyber scanning and beyond. *Computers & Security*, 128, 103123. https://doi.org/10.1016/j.cose.2023.103123 dl.acm.org+1

Lyon, G. F. (2009). Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Nmap Software LLC. (https://nmap.org/book/nmap.org)

El Yamany, H. F., Capretz, M., & Capretz, L. F. (2015). A multi-agent framework for testing distributed systems. *arXiv preprint* arXiv:1512.00313. https://arxiv.org/abs/1512.00313 arXiv

Zhang, D., Feng, G., Shi, Y., & Srinivasan, D. (2021). Physical safety and cyber security analysis of multi-agent systems: A survey of recent advances. *IEEE/CAA Journal of Automatica Sinica*, 8(2), 319–333. https://doi.org/10.1109/JAS.2021.1003820 scholars.cityu.edu.hk

Ponce, F., Verdecchia, R., & Miranda, B. (2025). Microservices testing: A systematic literature review. *Information and Software Technology*, 188, 107870. https://doi.org/10.1016/j.infsof.2025.107870

Yuan, T., Li, G., Lu, J., Liu, C., Li, L., Xue, J., & Liu, Z. (2021). GoBench: A benchmark suite of real-world Go concurrency bugs. *Proceedings of the 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. https://doi.org/10.1109/CGO51591.2021.9370317 dl.acm.org

Sharvari, T., & Sowmya Nag, K. (2019). A study on modern messaging systems — Kafka, RabbitMQ and NATS Streaming. *arXiv preprint* arXiv:1912.03715. https://arxiv.org/abs/1912.03715 arXiv

Salonen, N. (2025). A comparative study of Kafka and NATS (Master's thesis). University of Turku. https://www.utupub.fi/handle/10024/182402 (PDF: https://www.utupub.fi/bitstream-/handle/10024/182402/Salonen_Nino_opinnayte.pdf
https://github.com/MadiSergazy/Dissertation

Авторлар туралы мәліметтер
Информация об авторах
Information about authors

Тоқсеит Динара Қуандыққызы – PhD, доцент м.а., Ақпараттық қауіпсіздік, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана қ., Қазақстан
Тоқсеит Динара Куандыккызы – PhD, и.о. ассоциированный профессор «Информационная безопасность», Евразийский Национальный университет им. Л.Н. Гумилева, г. Астана, Казахстан
Tokseit Dinara Kuandykkyzy – PhD, Acting Associate Professor, Information Security, L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
e-mail: tokseit1990@gmail.com
ORCID: https://orcid.org/0000-0001-9075-3943

Сергазы Мәди Дарханулы – «Ақпараттық қауіпсіздік жүйелері» білім беру бағдарламасының 2-курс магистранты, «Ақпараттық қауіпсіздік» кафедрасы, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана қ., Қазақстан.
Сергазы Мәди Дарханулы – Магистрант 2 курса по ОП «Системы информационной безопасности» кафедра «Информационная безопасность», Евразийский Национальный университет им. Л.Н. Гумилева, г. Астана, Казахстан
Sergazy Madi Darkhanuly – Second-Year Master's Student, Information Security Systems Program, Information Security Department, L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
e-mail: madi.sergazy@nu.edu.kz,