Pak. j. life soc. Sci. (2025), 23(2): 249-257 E-ISSN: 2221-7630;P-ISSN: 1727-4915

Pakistan Journal of Life and Social Sciences

www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2025-23.2.00162

RESEARCH ARTICLE

# Development of a Methodology for the Safe Operation of Containerization Technology

Elena Revyakina[1], Andrey Gazizov[2]

[1,2]Don State Technical University, Rostov-on-Don, Russia

***ARTICLE INFO***

Received: May 4, 2025

Accepted: July 15, 2025

***Keywords***

Information security
Containerization Technology
Crypto Containers
Information Systems
Secure Network Infrastructure

***Corresponding Author:**

elena.a.revyakina@gmail.com

**ABSTRACT**

The article examines the need to develop algorithms for exploiting containerization technologies aimed at ensuring the stability and security of software applications during their interaction with information technology objects. This is relevant in the context of the constant growth of threats and requirements for the protection of information resources. The study proposes to develop a mechanism for implementing containerization in information systems, as well as to create software for monitoring the status of containers, controlling their launch, analyzing their operation and searching for vulnerabilities, which will increase the level of security and efficiency of using container technologies. In this study, the method of exploiting containerization technology is understood as the use of software scripts for loading and unloading critical files of the information system into crypto containers. This allows increasing the level of data protection and simplifying resource management.

## INTRODUCTION

Organizations, engaged in development and operation of software tools of information systems, actively use virtualization and containerization technologies. Containerization, as a method of packaging applications and their dependencies in isolated environments, plays a key role in increasing the efficiency of software development, deployment and management. Cryptocontainers, in particular, ensure the protection of confidential data by encryption, which makes them an important tool in ensuring information security. However, despite the advantages of containerization, such as acceleration of development processes, saving resources and increasing flexibility, the protection methods used for physical and virtual servers are not always suitable for containers. This creates the need to develop specialized approaches to ensuring security.

Modernapproaches to building a secure network infrastructure involve not only high efficiency of means informational security, but also their availability, flexibility, and responsiveness to potential threats. In this study, the method of exploiting containerization technology is understood as the use of software scripts to download and unload critical files of an information system into cryptocontainers. This allows increasing the level of data protection and simplifying resource management. In the process of exploiting information systems, the task of integrating methods and means of protection, as well as operational management of information security events, arises. One of the key problems is choosing the optimal strategy for implementing containerization into an existing security system. This is due to the lack of practical experience in using such technologies, as well as the complexity of their combination with traditional protection methods. As a result, at the stage of active exploitation of systems, the problem of resource protection becomes more acute, which requires the development of new approaches and solutions (Vyugov et al., 2025).

Thus, there is a contradiction between the growing requirements for the security and stability of the functioning of information systems and the possibility of integrating new methods, such as containerization, into existing security systems. This defines the research problem, which consists in finding effective ways to use containerization to ensure information security in accordance with modern requirements.

## MATERIALS AND METHODS

Assessing the level of information security in the context of information systems with implemented containerization technology is a multifaceted task that requires taking into account the specifics of this technology. Containerization, as a method of virtualization at the operating system level, offers significant advantages in terms of flexibility, scalability, and efficiency of using software and computing resources. However, the above-mentioned advantages are accompanied by new challenges in the field of information security, which require clarification in the approach to information security (Boydadaev, 2024).

The main problem with assessing information security in containerized systems is that such systems have a slightly different architecture compared to traditional information systems. Containers use a common kernel of the host operating system, which ensures high performance and resource savings, but at the same time creates potential vulnerabilities. Kernel-level vulnerabilities can lead to the compromise of all containers running on a single host, which makes host security critical. In addition, containerized systems are often built on a microservice architecture, which involves multiple network interactions between containers. This aspect can increase the number of potential approaches to implementing an attack and requires a well-developed approach to protecting network interactions.

To increase the protection of inter-network interaction, it is necessary to use a technology such as a firewall, which can effectively minimize unwanted interactions with the external environment, which means resources outside the containerized system. This step may require the expenditure of additional resources, but ultimately provides greater security, compared to classic information systems due to the high degree of isolation of processes and network interaction. Another important problem is the lack of unified standards and methodologies for assessing the security of containerized systems. As a result, there is a need for regular vulnerability scanning and container configuration analysis with subsequent analysis of the results, taking into account the specifics of each container image and the information system as a whole.

One of the key issues in the context of containerized systems security is the widespread use of container images from open repositories such as Docker Hub. These repositories contain a huge number of ready-made images that can be easily used to deploy applications. However, many of these images contain vulnerabilities or misconfigured security settings. Research shows that a significant portion of the images available in open repositories contain outdated versions of software that may be vulnerable to known attacks. For example, in 2020, a study by Palo Alto Networks found that more than 30% of images in Docker Hub contained critical vulnerabilities (Vikhlyaev, 2024). This poses serious security risks, since the use of such images can lead to a compromise of the entire system. In addition, many developers do not pay enough attention to checking images for malicious code. Images containing hidden cryptocurrency miners or other forms of malware have been found in open repositories. This highlights the need to implement rigorous image verification processes before and during use in production environments. Containerized systems are often the target of attacks that exploit known vulnerabilities. In recent years, several serious vulnerabilities related to container technologies have been recorded. The problem of assessing information security in containerized systems is complex and requires taking into account many factors, including the specifics of the container architecture, the use of images from open repositories, and the presence of known vulnerabilities. The widespread use of container technologies and their integration into modern information systems make this problem especially relevant. To solve it, it is necessary to develop specialized tools that will effectively assess and improve the security level of containerized systems. This includes the implementation of automated testing processes, regular component updates, and the use of modern protection mechanisms.

### 2.1. Theoretical aspects of information security in containerized information systems

Containerization technology is an operating system-level virtualization method that isolates applications and their dependencies in separate containers. Unlike classic virtual machines, which emulate hardware and require a separate operating system for each instance, containers use a common kernel of the host operating system, making them more lightweight and resource efficient (Fung et al., 2024).

Containerization plays an important role in ensuring increased information security due to several key features. First, containers provide process isolation, which helps minimize the risks of data leakage and unauthorized access. Each container runs in its own isolated space, which prevents other containers or host processes from interfering with its operation. This is especially important in conditions where several applications belonging to different users or organizations can run on one host. Second, containerization helps improve security by standardizing and automating the processes of deploying and managing applications. Containers are created based on images that contain all the necessary dependencies and settings. This helps minimize the risks associated with configuration errors or lack of necessary updates. In addition, the use of orchestration tools such as Kubernetes allows you to automate the processes of scaling, updating, and restoring containers, which reduces the likelihood of vulnerabilities due to human error. Third, containerization helps improve security by simplifying the processes of testing and implementing updates. Since containers are isolated from each other, updates and changes in one container do not affect the operation of others.

Containerization technology is the process of isolating software applications and their dependencies into lightweight, self-contained, and reproducible containers that can run independently of the underlying operating system (Amaliev et al., 2024). Containerization enables applications to be portable, reproducible, and independent of infrastructure constraints.

The fundamental principle of containerization is the use of virtualization at the operating system level. Unlike classic virtualization, in which each virtual environment (virtual machine) is allocated a separate copy of the operating system, containers use a single kernel of the host system, which makes them much lighter in terms of resource consumption. Containers provide process isolation, allowing each of them to function in a separate environment that contains only the necessary libraries (Shaidullin et al., 2025).

The container environment operates on the copy-on-write (COW) file system, which reduces the amount of disk space occupied by dividing the base layers between containers. This mechanism significantly speeds up container startup and reduces the load on the system. Thanks to this approach, containers can be launched almost instantly, and they are updated by replacing only the changed layers, which minimizes downtime and reduces the amount of data transferred. Thus, containerization provides effective management software environments, increasing the mobility, performance and security of applications. At the same time, it is important to consider the potential risks associated with the operation of this technology, as well as to use the best practices for managing container infrastructure to minimize vulnerabilities and ensure reliable protection of information systems.

Despite the many advantages of containerization, its use is associated with certain risks that can pose a threat to the security of information systems. Container infrastructure includes many elements, such as the operating system kernel, container images, networks, image registries, and orchestration mechanisms, each of which can serve as a potential attack point. Let's consider the main threats and risks associated with the use of containerization technology, as well as possible methods for minimizing them.

Container isolation is achieved through namespaces and cgroups mechanisms in the operating system kernel. However, the level of isolation provided by containers is significantly inferior to full-fledged virtualization, which makes attacks on the host system kernel level possible

Container images are the foundation for deploying containers, but when used with unverified, outdated, or in correct collected images may introduce vulnerabilities into the system. The main problems associated with container images are:

use of outdated components - many container images include libraries and packages that contain known vulnerabilities;

Malicious images – uncontrolled use of third-party images from open registries can lead to the download of malicious code; Protection methods:

use of official and verified images from trusted repositories (e.g. Docker Hub, Google Container Registry, AWS Elastic Container Registry);– Regular updating of images and dependencies.

## 2.3 Benefits of using containerization technology in information security

Modern software systems require effective methods of ensuring information security, given the dynamic development of threats and the need to protect data at all levels of architecture applications. In this context, containerization becomes not only a technology that improves the process of deploying and managing software environments, but also an important tool in ensuring the security of applications and data.

One of the key benefits of containerization is the ability to run applications in isolated environments. This isolation ensures that processes in one container cannot interact with processes an other container, unless this is provided for by security settings. Containers use namespaces and cgroups mechanisms, which allow you to control access to host system resources. Namespaces create logically separated instances of system resources, such as file systems, network, process identifiers (PIDs), and interprocess communication (IPC). This prevents one container from interfering with the processes of another (Usoltsev et al., 2024). Cgroups, in turn, allow you to set limits on the use of processor time, RAM, and other system resources, preventing attacks based on resource exhaustion. Thus, container isolation provides a high level of protection, preventing unauthorized access to data and reducing the likelihood of attacks spreading between applications running on the same host system.

Containers contain only the minimum required set of dependencies and executables, which significantly reduces the potential attack surface. Traditional server and cloud infrastructures often include numerous software libraries and services, many of which may contain vulnerabilities. Containers, in turn, provide the ability to include only those components that are necessary for a specific application to work. A minimalist approach to container configuration allows you to exclude unnecessary services, ports, and dependencies, which reduces the likelihood of exploitation by attackers. In addition, container images can be built using the "least privilege" principle, excluding administrative accounts from them and granting each container only the necessary access rights.

Containerization enables dynamic scaling of applications depending on the load. Orchestration tools such as Kubernetes, Docker Swarm, and OpenShift allow you to quickly increase or decrease the number of running containers, adapting to changing operational conditions. The flexibility of containers is also evident in their ability to work in various environments, including cloud platforms, on-premises servers, and hybrid infrastructures. This means that companies do not need to completely rebuild their IT infrastructure when migrating to container technologies, and security remains high regardless of the deployment environment.

Within the frame work concepts DevSecOps containerization allows you to integrate security mechanisms across all stages of the application lifecycle. Automated processes for scanning container images for vulnerabilities (for example, using tools like Trivy, Clair, Snyk, and Anchore) allow you to identify potential threats already at the development stage (Marker et al., 2024). Additionally, automated monitoring and analysis tools such as Aqua Security and Sysdig Secure provide the ability to monitor container operations in real time. They monitor the behavior of running containers, analyze network connections, and prevent unauthorized access attempts.

If an individual container is compromised, it can be quickly replaced with a new instance without affecting the overall infrastructure. This property of container technologies is especially useful in the event of cyberattacks, when it is important to minimize service downtime and ensure their uninterrupted operation. Thanks to the Copy-On-Write (COW) file system, container environments support instant deployment of updates and rollbacks to previous versions of the application, which makes recovery from an incident almost instantaneous. This approach is especially important for critical services whose operation must be continuous.

Many modern security tools, such as Falco, Sysdig, and StackRox, provide continuous analysis of container behavior, detecting anomalous activity in real time. These solutions use machine learning and process behavior analysis techniques to detect suspicious patterns, such as execution of unknown binaries, attempts to escalate privileges, or changes to system configurations (Vered, 2024). Using such tools allows administrators to quickly respond to threats, preventing potential attacks and reducing the likelihood of exploitation of vulnerabilities in a container environment.

Container technologies provide advanced network security controls that effectively prevent cross-container attacks and provide robust protection against unauthorized access to data. One of the key tools in this area is container orchestrators such as Kubernetes, which support network security policies (Network Policies). These policies allow administrators to clearly define allowed and prohibited interactions between containers, which contributes to a more secure and manageable environment.

With these capabilities, you can set up network microsegmentation by separating applications into isolated network zones. Microsegmentation allows you to limit interactions between different system components, which significantly reduces the risks of horizontal attack propagation in the event of a compromise of one of the infrastructure elements. For example, if an intruder gains access to one container, their ability to attack other system components will be strictly limited thanks to pre-configured rules. Thus, container technologies not only provide effective application isolation, but also offer powerful tools for increasing the level of network security. Using microsegmentation and network security policies allows you to minimize potential threats and create a more attack-resistant infrastructure.

## RESULTS AND DISCUSSION

### Algorithmization of container exploitation methods to improve the security of software application operation

Development of exploitation algorithms technologies containerization, aimed at ensuring the stability and security of software applications in the process of their interaction with information technology objects. This is especially important in the context of the constant growth of threats and requirements for the protection of information resources. The study proposes to develop mechanisms for implementing containerization in information systems, as well as to create a Telegram bot for monitoring the status of containers, controlling their launch, analyzing their operation and searching for vulnerabilities, which will increase the level of security and efficiency in using container technologies

The creation of a bot for monitoring and managing Docker containers is due to the need to simplify and automate the processes of administering containerized applications. Such a tool provides a convenient interface for interacting with containers, allowing administrators to quickly respond to changes and maintain stable operation of the system.

Telegram bot for managing Docker containers is a modern software solution created to simplify interaction with container infrastructure. This tool is aimed at optimizing routine processes related to monitoring, management and analysis To containers. The main idea is to provide users with an intuitive interface that works within the popular Telegram messenger, which minimizes the complexity of setup and training.

**The product under development is view Telegram bot stands out from existing analogues [] thanks to the following key features:**

1. The program does not require installation of additional software or complex configuration. All work is done through Telegram, which reduces the entry threshold for new users;

2. Real-time availability, notifications about the status of containers and the ability to manage them are available at any time and from any device with access to the messenger;

3. Integration with everyday tools. Telegram is one of the most popular messengers in the world, which is actively used in the professional environment, which makes our product as accessible and familiar to the audience as possible;

4. Focus on small and medium businesses; most existing solutions are aimed at large corporations, while our bot is ideal for startups, small DevOps teams and freelancers

**The main functions of the program include:**

Monitoring of container statuses, automatic notification of container starts and stops allows time identify and troubleshoot problems while minimizing service downtime;

State query and state management, the ability to query the current status of containers, as well as manually start or stop them via a bot provides flexibility and control over the environment;

Working with logs, access to container logs via a bot allows you to quickly analyze the causes of possible errors or unstable operation of applications;

Backup, integration of the backup mechanism guarantees the safety of data and provides the ability to restore it in case of unforeseen situations;

Component updates, regular reports on the availability of updates, the ability to request them and manually launch them through a bot help maintain the current and secure state of all components.

So the bot unites functionality for monitoring, managing and updating containers in a single interface, reducing administrative costs and increasing the efficiency of working with containerized infrastructure. This is important because the use of vulnerable images or improperly configured containers can lead to the compromise of the entire system. Scanning allows you to identify such problems at an early stage, which allows you to eliminate them before deploying to the final system, minimizing security risks. It is also necessary if the content has not been updated for a long time, this allows you to make the necessary decision either to update the internal configuration or other measures if there is a known incompatibility of certain versions of applications in the container environment or in the container-host system (Kazaryan et al,. 2024).

Creating a Telegram bot for managing Docker containers requires careful planning and adherence to the sequence of production stages. The first stage involves defining the bot structure, identifying key modules and their interaction. The main focus is on: - integration with the Docker API to perform operations with containers (e.g. monitoring, starting, stopping, updating and analyzing logs). Next, integration with the Telegram API is performed to create a convenient management interface via chat. This includes setting up user communication with the bot, processing commands and sending notifications; - implementing a system for differentiating access rights. To improve security, user roles with different levels of authority are designed, which is important for enterprises using a common infrastructure. At this stage, performance and scalability requirements are also taken into account so that the bot can operate stably under increased load. The next stage involves writing the code to implement all the planned bot functions.

**Important aspects of implementation:**

Connection to Docker: setting up interaction with the Docker Engine to perform basic operations (for example, getting a list of containers, managing their states, and viewing logs);

Setting notifications: creating a system for automatically sending notifications when the status of containers changes or failures occur;

Development of mechanisms for creating and storing backup copies of containers.;

Adding functions for checking for updates and applying them automatically or manually;

Designing an intuitive interface for working via Telegram, taking into account convenience and the minimum number of steps to complete tasks.

After the functionality is implemented, testing is carried out to ensure the reliability and stability of the bot:

Checking all declared capabilities of the bot for compliance with the technical specifications;

Modeling increased load to test the stability of the system under a large number of requests;

Security testing: checking for vulnerabilities, including assessing the reliability of the access rights control system;

Bug fixing: identified bugs are recorded and retested to ensure full functionality.

The program's start menu is an interactive panel for interacting with the bot. Among the available functionality, there is also the ability to subscribe and unsubscribe from receiving notifications about changes in container statuses and receiving a regular report on possible updates to container components.

**Below in Figure 1 is presented example implementation of obtaining container statuses**

```
async    def    get_container_status(update:    Update,    context:
ContextTypes.DEFAULT_TYPE):
    client = docker.from_env()
    query = update.callback_query
    try:
        containers = client.containers.list(all=True)
        if containers:
            status_message = "Контейнеры:\n"
            for container in containers:
                status_message    +=    f"{container.name}    (ID:
{container.short_id}) - Статус: {container.status}\n"
        else:
            status_message = "Нет контейнеров."
    except docker.errors.DockerException as e:
        logger.error(f"Docker error: {e}")
        status_message = f"Ошибка доступа к Docker: {e}"
    await query.edit_message_text(status_message)
    await send_start_button(update)
```

**Figure 1 - Implementation of receiving container statuses**

Let's look at some functions more details of this code. The get_container_status function connects to Docker via the docker.from_env() client, gets a list of all containers (including stopped ones) and generates a message with their statuses (name, ID and state). If no containers are found, a corresponding notification is displayed. In case of a Docker error (for example, a daemon is unavailable), it is logged and an error message is sent to the user. After that, the container status is sent to Telegram using query.edit_message_text, and then the send_start_button function is called, which adds a button to return to the main menu

Operational receiving Container logging is critical for monitoring and troubleshooting applications and infrastructure. Logs provide detailed information about events occurring inside container, such as startup, shutdown, errors, warnings, and other system messages. For example, in the given fragment of the logs of the container new–zabbix–db, you can see that MariaDB has started successfully, but there are warnings about the impossibility of using some functions (for example, io_uring_queue_init() failed with errno 1), which may indicate problems with the configuration or environment

In this example MariaDB logs show that the database has successfully initialized the buffer pool and completed startup, but there are warnings about the impossibility of using some functions (for example, liburing disabled). This can be useful for administrators to understand which functions are unavailable and how this may affect performance. Promptly obtaining such logs allows you to quickly respond to problems, minimize downtime and ensure stable operation of the system.

Function prompt_container_selection connects to Docker, gets a list of all containers, and prompts the user to select a container to receive logs from via interactive buttons in Telegram. If there are no containers or a Docker error occurs, the user is sent a corresponding message. After selecting a container in handle_log_request, its name is saved in context.user_data, and the user is prompted to enter the number of log lines. In handle_log_lines, the entered number of lines is checked, and logs are requested from the selected container using the Docker API. Logs are split into parts (20 lines each) and sent to the user in Telegram to avoid message length restrictions.

If arise errors (such as invalid input or Docker issues), the user is notified and send_start_button is always called at the end to return to the main menu. This allows for convenient querying and viewing of container logs.

Functional programs allows you to generate a report on available updates. This report is useful because it informs administrators about available updates to packages (e.g. apt, base-files, libssl3), including critical security updates to help prevent exploitation of vulnerabilities.

Function prompt_container_selection_for_action allows the user to select a container to start or stop. First, it checks if the user is authorized (their name must be in the AUTHORIZED_USERS list). If access is allowed, the bot gets a list of all containers and creates interactive buttons with their names. The user selects a container, and depending on the selected action (start or stop), the handle_container_action function is called.

By using functions handle_container_action bot checks user authorization and performs the requested action (start or stop a container) via the Docker API. If the container is not found or a Docker error occurs, the user receives a corresponding notification. If successful, the bot reports that the request has been completed and offers to enter /start to return to the main menu. This provides convenient container management via Telegram with restricted access for unauthorized users. The get_container_list function executes the docker ps –a command to get a list of all containers (including stopped ones) and returns their names (Korochentsev et al., 2021). The show_container_list function then creates interactive buttons for each container, allowing the user to select a container to scan for vulnerabilities. If there are no containers, the user receives a corresponding notification.

This Telegram bot also implements a system for delimiting the rights to execute critical functions, such as starting and stopping containers, as well as launching updates, since this functionality can potentially paralyze the operation of an application, of which this or that container may be a component.

## CONCLUSION

Telegramm-bot for management Docker– containers is a modern software solution created to simplify interaction with container infrastructure. This tool is aimed at optimizing routine processes related to monitoring, managing and analyzing containers. The main idea is to provide users with an intuitive interface that works within the popular Telegram messenger, which minimizes the complexity of setup and training. Telegram functionality– the bot includes the following key features, presented below:

The ability to find out the status of containers at any time, obtain information about running instances, and evaluate their current performance;

The program automatically sends messages about any changes in the state of containers, including their start, stop, or the occurrence of errors. This helps users to always be aware of what is happening and responds to critical events in real time;

The program provides the ability to manually start and stop containers, which is especially useful for performing point operations or in emergency situations;

Performing a full backup of containers and their contents ensures the safety of data and facilitates the recovery process in case of failures;

The program provides convenient access to container logs on request, which greatly simplifies problem diagnostics and performance analysis.

Access to container logs by request allows you to define permission levels for different users, which increases security and prevents unauthorized interference in the system.

The developed methodology for the safe operation of containerization technology is a solution for convenient and efficient management of container infrastructure based on Docker technology. Developed as a result of the conducted research program. The product simplifies routine tasks and increases the productivity of information security specialists, DevOps engineers and system administrators.

## REFERENCES

Amaliev HS, Yakubovich SA. Analysis of cryptocontainer detection during user action reconstruction. Science Bulletin 2024;2.12(81):1316–1320.

Boydadaev MN. Comparative analysis of the performance of network plugins of the KUBERNETES orchestrator. Universum: Technical Sciences 2024;1.11(128):38–47.

Fung WQ, Bogatyrev VA, Karmanovsky NS, Le VH. Evaluation of probabilistic-temporal characteristics of a computer system with container virtualization. Scientific and Technical Bulletin of Information Technologies, Mechanics and Optics 2024;24(2):249–255.

Kazaryan MM, Cherckesova LR, Revyakina EA, Safaryan OV, Porksheyan VV. Analysis of the possibilities of conducting attacks on the functions of transferring control of the operating system console using active reconnaissance methods. High-tech in Space Research of the Earth 2024;16(3):18–29.

Korochentsev DA, Cherkesova LV, Revyakina EA. Import-substituting technologies for ensuring information security and data protection. Rostov-on-Don: Don State Technical University; 2021. 334 p.

Marker VA, Kochegurova EA. Application of containerization technologies for assembly and delivery of a multiservice web application. In: Youth and Modern Information Technologies: Proceedings of the XXI International Scientific and Practical Conference of Students, Graduate Students and Young Scientists, April 15–18, 2024, Tomsk. Tomsk Polytechnic University; 2024.

Shaidullin DT, Ivanov FYu, Khakulova AS. Containerization in development: modern technologies, advantages and key approaches. IT & Transport 2025.

Usoltsev DA, Boykov SYu. Containerization technologies: DOCKER and KUBERNETES in 2024.

Vered S, Scheinman V, Kuznetsov M, Kotov A. Improving the efficiency of software development processes: container technologies. Software Systems and Computational Methods 2024;4:151–161.

Vikhlyaev DR. Deploying a container using Docker Desktop on the Windows operating system. Postulate 2024;Dec 12.

Vyugov SG, Kozachok AV. A modern approach to monitoring the security of container environments. In: Grigorov NI, Nikitina EYu, Rabchevsky AN, Chernikov AV, editors. 2025. p. 25.