IRSTI 76.13.23 https://doi.org/10.58805/kazutb.v.4.21-198

SECURITY ISSUES OF CONTAINERIZATION OF MICROSERVICES

S.U.Aralbayev[1], G.Z.Ziyatbekova[1,2*], P.Kisala[3]
[1]Al-Farabi Kazakh National University, Almaty, Kazakhstan,
[2]RSE Institute of Information and Computational Technologies MSHE RK CS,
Almaty, Kazakhstan,
[3] Lublin Technical University, Poland
e-mail: ziyatbekova@mail.ru

Microservices architecture has become known for its scalability and flexibility in recent years. Containerization of microservices using technologies such as Docker and Kubernetes has increased the efficiency of application deployment. However, this technological change has raised questions about the impact of containerization on security. This paper discusses various security aspects in the context of microservices containerization, including security, reliability, and data integrity. We explore challenges, best practices, and emerging trends affecting the security of containerized microservices.

Keywords: Docker, Kubernetes, containerization, microservices.

ПРОБЛЕМЫ БЕЗОПАСНОСТИ КОНТЕЙНЕРИЗАЦИИ МИКРОСЕРВИСОВ

С.У. Аралбаев[1], Г.З. Зиятбекова[1,2*], P. Kisala[3]
[1]Казахский национальный университет имени аль-Фараби, Алматы, Казахстан,
[2]Институт информационных и вычислительных технологий КН МНВО РК,
Алматы, Казахстан,
[3]Люблинский технический университет, Польша,
e-mail: ziyatbekova@mail.ru

Архитектура микросервисов в последние годы стала известна своей масштабируемостью и гибкостью. Контейнеризация микросервисов с использованием таких технологий, как Docker и Kubernetes, повысила эффективность развертывания приложений. Однако это технологическое изменение вызвало вопросы о влиянии контейнеризации на безопасность. В этой статье рассматриваются различные аспекты безопасности в контексте контейнеризации микросервисов, включая безопасность, надежность и целостность данных. Мы исследуем проблемы, передовой опыт и новые тенденции, влияющие на безопасность микросервисов контейнеров.

Ключевые слова: Docker, Kubernetes, контейнеризация, микросервисы.

МИКРОСЕРВИСТЕРДІ КОНТЕЙНЕРЛЕУДІҢ ҚАУІПСІЗДІК МӘСЕЛЕЛЕРІ

С.У. Аралбаев[1], Г.З. Зиятбекова[1,2*], P. Kisala[3]
[1]әл-Фараби атындағы Қазақ ұлттық университеті, Алматы, Қазақстан,
[2]Қазақстан Республикасы Ғылым және жоғары білім министрлігі Ақпараттық және есептеуіш технологиялар институты, Алматы, Қазақстан,
[3]Люблин техникалық университеті, Польша
e-mail: ziyatbekova@mail.ru

Микросервис архитектурасы соңғы жылдары өзінің ауқымдылығы мен икемділігімен танымал болды. Docker және Kubernetes сияқты технологияларды қолдана отырып, микросервистерді контейнерлеу қосымшаларды орналастырудың тиімділігі артты. Алайда, бұл технологиялық өзгеріс контейнерлеудің қауіпсіздікке әсері туралы сұрақтар туғызды. Бұл мақалада микросервистерді контейнерлеу контекстінде қауіпсіздіктің әртүрлі аспектілері, соның ішінде қауіпсіздік, сенімділік және деректердің тұтастығы қарастырылады. Біз контейнерлік микросервистердің қауіпсіздігіне әсер ететін мәселелерді, ең жақсы тәжірибелерді және жаңа тенденцияларды зерттейміз.

Түйін сөздер: Docker, Kubernetes, контейнерлеу, микросервистер.

Introduction. Microservices have become the dominant architectural paradigm for building and deploying modern applications. They allow complex systems to be developed as a collection of small, loosely coupled services. Microservices offer many advantages, including scalability, maintainability, and rapid development [1].

The use of containers has grown dramatically in recent years. This is because containers have many advantages. Containerization is a technology that allows you to package and run an application Together with its dependencies and configurations in a lightweight and isolated environment called a container. As the Docker announcement says, with them you can ”do it once, and you can do it anywhere.” That is, by isolating the application from the rest of the machine it's running on, we can integrate the application and all its dependencies into a package. This makes it possible to do isolation in parallel - several different containers that don't interfere with each other. Containerization of microservices, often facilitated by Docker containers and managed with Kubernetes, has further simplified the deployment process. However, as microservices and containers become more prevalent, security issues arise.

Materials and methods. The security issues of containerization of microservices are extensive and complex. Therefore, we will consider all possible issues from the microservice code installed in the container to the parties that have access to the containers. We will use the following approaches to assess security in containerized microservices:

- Identification of vulnerabilities of microservices in containers;
- Risk assessment;
- Exploring ways to realize the risks (threats);
- Find ways to minimize the consequences of threats (mitigation).

Results and discussion. A microservice is an independent, self-contained resource created as a separate executable file or process. A microservice communicates with other microservices through standard, lightweight interprocess communications such as the Hypertext Transfer Protocol (HTTP). What is the meaning of the term ”autonomous resource”, let's focus on it. That is, a single microservice performs only one function. For example, a microservice is only concerned with registering orders coming into the system from customers. Even the order data is not sent to the consumer. But a microservice can launch another microservice that performs this function. As we have seen, microservice based applications are very different from monolithic applications. This is because in a monolithic application, all the services in the application are assembled into one large executable file [2].

Let's look at the above concepts with a simple example. Web application of a calculator. As can be seen from Figure 1, although in a monolithic application the operations in the calculator (addition, subtraction, multiplication, division) are written as separate functions, they operate in only one process.

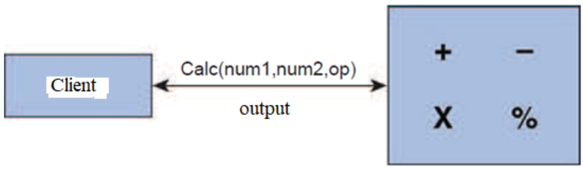


Fig. 1 - Calculator application built on a monolithic architecture

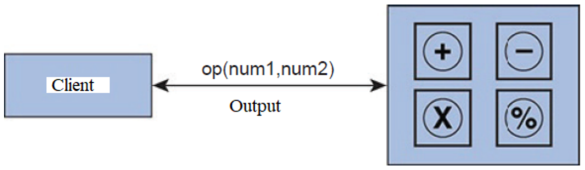


Fig. 2 - Calculator application based on microservices architecture

The application shown in the above example is too simple to fully describe microservices. However, if any error occurs in any of the services of this application or if we want to add a new service to the application, we will have to terminate the application, modify its code and restart it. Now, let's look at the Figure 2 web application structure based on microservices architecture.

Before containers, dependency ownership was a nightmare for developers who needed different versions of the same packages for both applications. We realized that the easiest way to solve this problem was to run applications on separate machines. This container technology, based on satisfying a single request, created the ability to run multiple applications on a single server, isolating dependencies. It is very important to realize that the Container runs in the kernel of the operating system and is isolated by the operating system tools, not by the hardware capabilities of the computer, such as a virtual machine. We can see this in Figure 3.

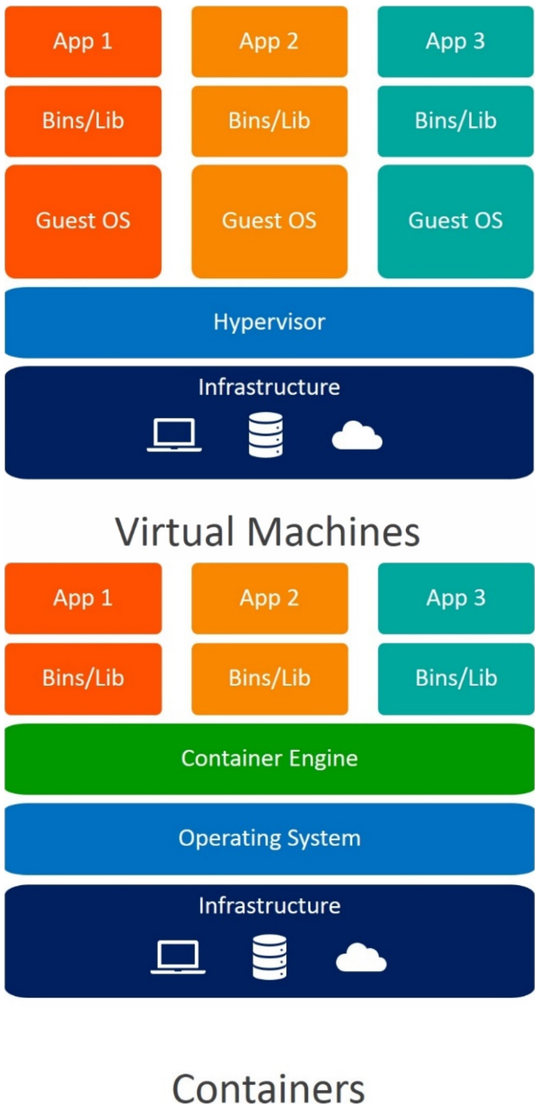
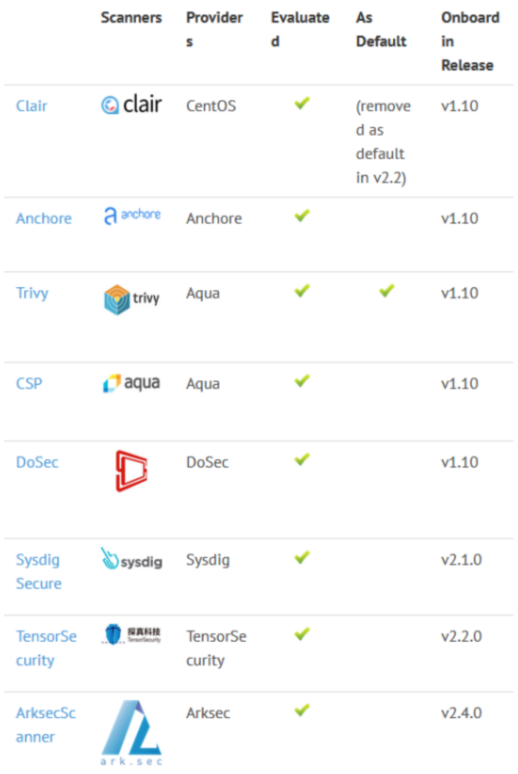


Fig. 3- Structure of containerization and virtualization



	Scanners	Providers	Evaluated	As Default	Onboard in Release
Clair	clair	CentOS	✓	(removed as default in v2.2)	v1.10
Anchore	anchore	Anchore	✓		v1.10
Trivy	trivy	Aqua	✓	✓	v1.10
CSP	aqua	Aqua	✓		v1.10
DoSec		DoSec	✓		v1.10
Sysdig Secure	sysdig	Sysdig	✓		v2.1.0
TensorSecurity	探真科技	TensorSecurity	✓		v2.2.0
ArksecScanner	ark.sec	Arksec	✓		v2.4.0

Fig. 4 - Ranking of scanner tools

The next logical step we should take after deploying microservices in containers is to distribute containerized applications to a cluster of servers. Thanks to coordination tools like Kubernetes, this process is automated to the point where you don't need to manually deploy applications to specific machines, you just tell the coordination tool which containers to run and it will find the right machine for each one.

From a security perspective, a containerized environment is similar to a normal hosting environment.

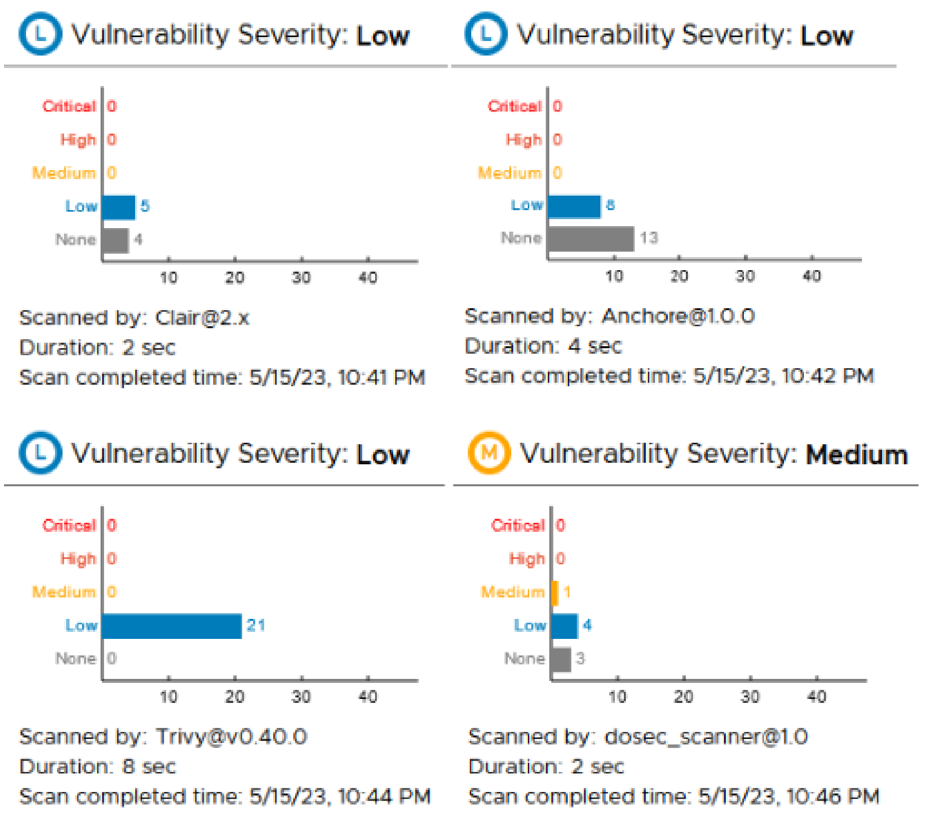
Attackers try to steal data, change system behavior, or use other people's computing resources to mine cryptocurrency. When moved to containers, none of these things change. However, containers significantly change the way applications work, leading to a different set of security risks.

Risks in containers and their mitigation:

- A risk is a potential problem and its consequences.
- A threat is a pathway for a risk to materialize.
- Mitigation is countermeasures that can prevent a threat or reduce the likelihood of its successful realization [3].

Currently, there are many scanning tools that detect container vulnerabilities and assess risks (Figure 4).

We tested the placed and configured container for security using Trivy and Anchore tools. The result is shown in Figure 5.



Scanned by: Clair@2.x
Duration: 2 sec
Scan completed time: 5/15/23, 10:41 PM

Scanned by: Anchore@1.0.0
Duration: 4 sec
Scan completed time: 5/15/23, 10:42 PM

Scanned by: Trivy@v0.40.0
Duration: 8 sec
Scan completed time: 5/15/23, 10:44 PM

Scanned by: dosec_scanner@1.0
Duration: 2 sec
Scan completed time: 5/15/23, 10:46 PM

Fig. 5 - Result of container scanning

Risks vary from organization to organization. For example, for a bank that holds customer money, the main threat is theft of money. The main headache for an online store is fraudulent transactions. For example, a user running a personal blog may fear that someone will hack into their account and start imitating themselves and writing obscene comments. In different countries the legislation on personal information protection has its own peculiarities, so the risks of leakage of personal data of users also differ - in many countries the risks have the reputation of ”only”, and in Europe the General Data Protection Regulation (GDPR) allows to fine up to 4% of the total revenue of the company.

Because risks vary widely, the relative importance of potential risks, as well as the appropriate set of mitigation tools, varies considerably. Risk management is based on the process of systematizing risks, listing potential threats, prioritizing them, and choosing how to minimize their consequences.

Threat modeling is the process of recognizing and counting potential threats to a system. By planning the analysis of its components and potential attack vectors, a threat model helps identify the most vulnerable areas of a system to attack.

There is no single comprehensive threat model; it all depends on the risks of the particular environment, organization, and applications being run. But you can list some potential hazards of containerization that are common to many, even all.

What vulnerabilities exist and what danger do they pose?

Conventionally, vulnerabilities in docker images can be divided into:

- OS Vulnerabilities-vulnerabilities in the main images and system packages included in this image;
- Dependencies-vulnerabilities in third-party dependencies;
- Software Vulnerabilities-vulnerabilities in application code running in containers;
- Dockerfile-dangerous instructions for building images.

Using a vulnerable docker image can pose a serious threat to the security and stability of an organization's IT infrastructure and applications:

Infrastructure loop security breach: a vulnerable image can be an access point for attackers to gain unauthorized access to other containers or even to the host system, other internal organization resources, sensitive data.

Malware infection: An attacker can drop malware on a corrupted image that will infect an organization's infrastructure or applications. This can lead to data loss or disruption of critical services;

General instability: using a vulnerable image can lead to system performance issues.

Analyzing and systematizing the risks within the containerization technology, we get the same result as in Table 1.

Table 1 - Systematized representation of hazards in containerization technology

Images	Image Registry	Orchestration	Containers	Host OS
Use of unreliable images	Unsecured connection	No restriction on administrative access	Vulnerabilities in the working environment	The OS kernel is common to all containers
Software Vulnerabilities	Using outdated images with vulnerabilities	Login without authorization	Unlimited network access	Vulnerabilities of OS components
Configuration errors	Insufficient level of authentication and authorization	Lack of isolation and inspection of traffic between containers	Secure customized working environment	Incorrect user access rights
	Depending on the importance of the data, containers of different levels are not placed on hosts	Vulnerability of containerized applications	The file system is accessible via containers	
	Orchestrator configuration errors	Presence of unplanned containers in the runtime environment		

Build a risk model using the above data:

Where:

- External attackers attempting to gain access to an externally hosted system (external attackers);
- internal attackers gaining access to a specific part of the extended system;
- how malicious internal actors, such as developers and administrators, have some level of extended login

credentials;

- inadvertent internal actors, which can cause problems through inadvertence;
- application processes are not the people who have access to specific software on the system [4].

If we look at these vectors in more detail:

Vulnerable code. The life cycle of an application begins when a developer writes his code. It, and its dependencies, may include flaws (vulnerabilities). There are lists of thousands of known vulnerabilities that (if present in the application) can be exploited by attackers. This must be done regularly, as vulnerabilities are constantly being discovered in existing code. The analysis process should also detect containers with outdated software that needs to be updated with security patches. In addition, there are analyzers that can detect malware embedded in the image (Figure 6).

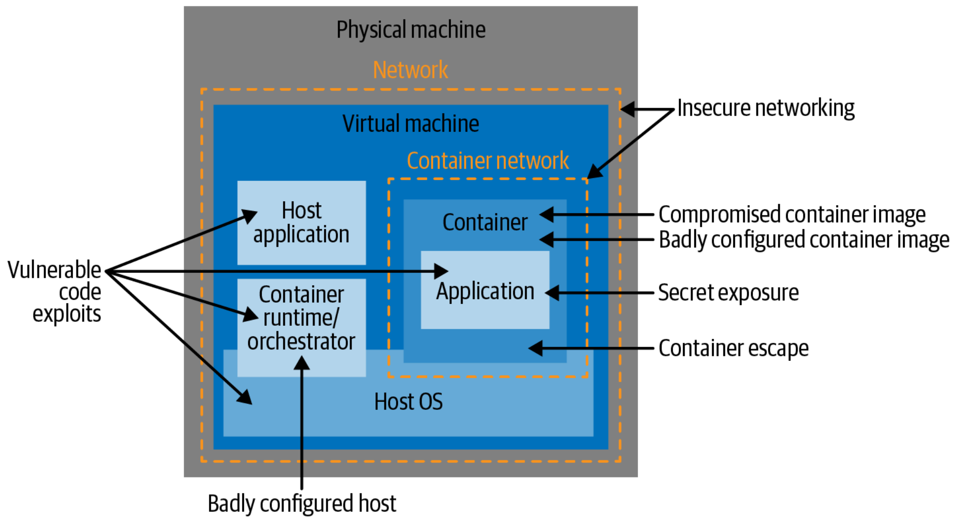


Fig. 6 - Vectors of attack on the container

Poorly customized container images. Written code is embedded in the container image. The configuration of the container image build provides many opportunities to create vulnerabilities that open the way for further attacks on a running container. These include executing the container as a superuser, resulting in the superuser being granted more authority than necessary.

Attacks on the build system. If an attacker can change the container image structure or affect it in any way, they can inject malicious code that is then activated in the production environment. In addition, the ability to gain a foothold within the assembly environment is a platform for malware to further infiltrate the production environment.

Supply Chain Attacks. The collected container image is stored in a registry that will be extracted before launch. How do you ensure that the extracted image matches the one previously entered into the registry? Could pests have made changes to it? Anyone who can replace the image or modify it in the space between build and deployment can execute any code on the extended system.

Poorly tuned containers. you can run a container with settings that result in unnecessary and sometimes unintended permissions. When downloading YAML configuration files from the Internet, do not run them without making sure there are no safe settings!

Vulnerable hosts. Containers are executed on host computers, so the code running on them needs to be checked for vulnerabilities (e.g., tracking down older versions of coordination mechanism components with known vulnerabilities). To reduce the attack surface, it makes sense to minimize the size of the software running on each host. In addition, the hosts should be properly configured according to the security guidelines [5].

Conclusions. Containerization of microservices is a powerful way to create scalable and flexible applications. However, it brings new challenges to the forefront, especially from a security perspective.

This paper examines the impact of microservices containerization from different perspectives on security and provides recommendations to address these challenges. As technology advances, organizations need to prioritize security to fully leverage containerized microservices.

References

1. Jung, Kwang wook, Chao, Yang-Ki, Tak, Yong-Jin. Containers and orchestration of numerical ocean model for computational reproducibility and portability in public and private clouds: Application of ROMS 3.6. Simulation Modelling Practice and Theory. - 2021. - 109 p.

2. Parminder Singh Kocher. Microservices and Containers. Addison-Wesley Professional.- 2013. - pp. 990-998.

3. Liz Rice. Container Security: Fundamental Technology Concepts that Protect Containerized Applications. O'Reilly Media.- 2020. - 198 p.

4. Muthanna. Distributed intelligent communication network architecture for unmanned vehicles // Electrosvyaz. - 2020. No 7. - pp. 29-34.

5. Ermolenko D., Kilicheva K., Khakimov A., Muthanna A.: Exploring a Model Network for Orchestation IoT Services Based on Kubernetes // Telecom IT. - 2020. - Vol. 8. - Iss. 4. - pp. 69-82 (in Russian).

Information about the authors

Aralbayev Serikbolsin Usenbayevich - graduate student at Al-Farabi Kazakh National University;

serikbolsynaralbayev@gmail.com;

Ziyatbekova Gulzat Ziyatbekkyzy - PhD, Acting Associate Professor NAO Al-Farabi Kazakh National University; Senior Researcher at the RSE Institute of Information and Computational Technologies of the National Academy of Sciences of the Republic of Kazakhstan; ziyatbekova@mail.ru;

Piotr Kisala - PhD, Associate Professor Lublin Technical University, Poland; p.kisala@pollub.pl

Сведения об авторах

Аралбаев Серикболсин Усенбаевич - магистрант НАО Казахского национального университета имени аль-Фараби, serikbolsynaralbayev@gmail.com;

Зиятбекова Гулзат Зиятбеккызы - PhD, и.о. доцента НАО Казахского национального университета имени аль-Фараби; старший научный сотрудник Института Информационных и вычислительных технологий КН МНВО РК, ziyatbekova@mail.ru;

Piotr Kisala - PhD, доцент Люблинского технического университета, Польша; p.kisala@pollub.pl