UDC 004.056

Kostiantyn Minkov,
Viktor Selivanov, Artem Volokyta

PROTECTION SYSTEM OF MICROSERVICE SYSTEMS

Костянтин Міньков,
Віктор Селіванов, Артем Волокита

СИСТЕМА ЗАХИСТУ МІКРОСЕРВІСНИХ СИСТЕМ

The article provides an overview of the security system for microservices. As methods used by this system, tools such as HTTPS, JWT, OAuth2, RM, TOTP were considered. The implementation is performed using the Java programming language.

Key words: microservice systems, security systems, distributed systems, authentication, role model.

Fig.: 1. Tabl. 0. Bibl.: 12.

У статті виконується огляд системи захисту для мікросервісних систем. У якості методів, якими оперує дана система, були розглянуті засоби, такі як HTTPS, JWT, OAuth2, RM, TOTP. Реалізація виконана за допомогою мови програмування Java.

Ключові слова: мікросервісні системи, системи захисту, розподілені системи, автентифікація, рольова модель.

Рис.: 1. Табл. 0. Бібл.: 12.

Relevance of research topic. Due to the rapid introduction of microservice systems among various e-commerce companies (Netflix, Amazon, Hailo), there is a need for the means for their effective protection.

Analysis of recent research and publications. Despite the large number of works and studies devoted to the systems of protection, the protection of the microsystems itself isn't quite detailed in the literature.

Identification of unexplored parts of the general problem. Integration of known methods of protection into a coherent system. Creating a system of protection focused on microservices.

Setting objectives. The purpose of this work is to systematize known methods of protection and their consolidation to create a unified system in the context of microservice systems.

The statement of basic materials. Given the rapid development and globalization of modern business, there has been a demand for the development and

support of large systems for automating business processes in large companies, thus making their digital transformation. Such systems are most often implemented in the form of monolithic systems [1]. Over time, such systems grow more and more and can become difficult to maintain, deploy and develop by a large team. The introduction of small changes often leeds to the need to re-plan the entire application. With the development of cloud technologies, the issue is about scaling such systems, since monolithic applications usually offer a large number of services, some of which are used more often than others. This leeds to constant financial overheads for the maintenance of such software products [2].

Microservices have become the solution to the above problems. The microservice pattern suggests dividing system A into a plurality of small services μS (μS_1, μS_2, μS_n), each offering a subset of services S (S_1, S_2, S_x) provided by program A. Each microservice is developed and tested by the team of developers μT_i. Each microservice is developed using the independent code bases and the μT_i team, which is also responsible for deploying, scaling, operating and upgrading the micro-service on IaaS / PaaS solutions in a cloud environment. This approach allows to simplify the scaling, deployment and making necessary changes to individual parts of the system [3].

However, it should be noted that the microservice technology is not without some disadvantages. Such an approach complicates the architecture of the system, there is a complex network model of interaction between its components, considering that the number of services can reach several hundreds (Figure 1). Delay in the network, fault tolerance, message transformation, network reliability, asynchronousity, versioning of different subsystems, changes in loads within a particular version of applications - common issues in such systems [4].

The security challenge caused by the complexity of the network is the ever-increasing difficulty in monitoring, auditing and analyzing the functioning of the entire program. Since microservices are often deployed in a cloud environment that the application owners do not control, it is difficult for them to imagine the overall view of the entire application. Thus, attackers can use this complexity to launch attacks on applications. Another security issue is related to the trust among distributed microservices. An individual microservice can be compromised and controlled by an attacker. For example, an attacker can take advantage of the vulnerability in the microservice facing the public user and increase his privileges on the virtual machine on which the microservice operates. As a result, individual microservices can become unreliable [5]. Also, there is a question of authentication of individual system services among themselves.

Since, as noted above, the system is distributed, and communication is most often performed using the REST architectural style, one of the first methods to help secure the system is to configure the HTTPS protocol accross of the system [7]. It should be noted that the use of keys generated in certification center is not necessary for all system services. This should only be done for client-facing applications. For

internal services there is enough to have self-signed certificate, which must be added to the keystore on the virtual machines of other microservices.

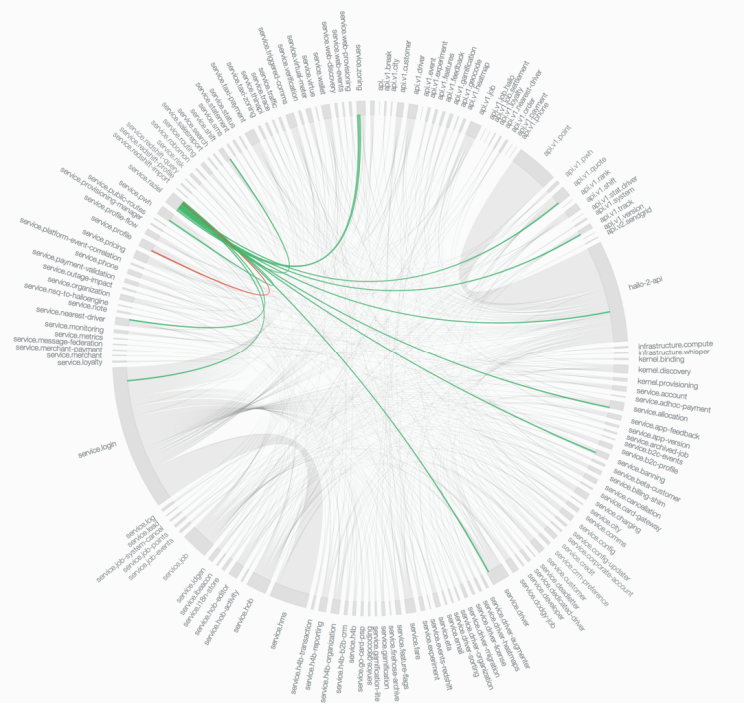

Fig. 1. Scheme of Heptio microsystem [6]

One of the standard ways to protect the system is to introduce the role model for users. This model simplifies the setup of checks in the system and builds a clear access model that fits well into the hierarchical model of the company's departments. Mathematically, such a model is described using formula 1 [8].

$$permissions(s) = \bigcup_{r \in roles(s)} \{p_i | (p_i, r) \in PA\} \quad (1)$$

OAuth2 is an open standard that allows third-party applications to have limited access to the HTTP service on behalf of the owner of the resource.

This standard solves the problem of the need to store a password in the client system, since compromising a third-party program will compromise the end-user password and all data protected by this password. Instead of using the resource owner credentials to access secure resources, the client receives an access token, a string that denotes a specific area, lifetime, and other attributes of access. Access Tokens are issued to third-party clients by the authorization server with the permission of the owner of the resource. The client uses a token to access secure resources hosted on the resource server [9].

Oauth2 introduces a clear limitation of the rights and duration of access to a secured resource by the client, since the token is temporary and contains only a set of specific permissions.

As an add-on to OAuth2, another one is used - JWT. In the standard

implementation of OAuth2, a random string is used as a token. It has some disadvantages, namely:

1. At each request to one of the protected microservices, an additional request to the SSO is required to confirm the validity of the token, which increases the load on the network and introduces an additional delay to the response time.

2. The token does not have any information about the user or the authentication system.

To resolve this issue, a RFC 7519 standard was introduced, which defines a JSON token containing a payload (user, custom role, SSO, time before disabling the token, etc.), and a digital signature generated with the private key of the authentication system based on the body of the request using RSA or ECDSA. Thus, any service having a public key can check the token without any additional queries and get information about the user directly from the token [10].

A known problem in developing security systems is the storage of user passwords in a relational database. Storing passwords in the usual way is equivalent to writing them on a digital paper. If an attacker gets access to the database and steals a password table, then he will be able to access each user's account.

The safest way to store a password is to hash it.

The problem with the SHA family of algorithms is that they were designed in such a way as to have a higher computing speed. Rapid calculations mean faster brute-force attacks. An example of adaptive functions that can compensate increasing computer power is the bcrypt algorithm. Also, the algorithm is resistant to attack using rainbow tables, because it uses a random tape (salt) that is added to the hash of the password while storing.

One of the standards that helps protect a user from losing a password is the two-factor authentication of the TOTP protocol.

One-time passwords are often better than stronger authentication forms, such as public key infrastructure (PKI) or biometric data, since this method does not require the installation of any client software on a user's computer [11].

The HOTP algorithm is based on the HMAC-SHA-1 algorithm and is adapted to increase the counter value size representing the message in the HMAC calculation. TOTP is a time version of this algorithm, where the value T, derived from the time and time step, replaces the counter C in the HOTP calculation (2) [12].

$$TOTP\ (K,T)\ =\ Truncate(HMAC-SHA-1\ (K,T)) \qquad (2)$$

As we combine all methods described above we will receive a security system which meets the goals defined for this work.

Conclusions. The result of the design and development is a system that includes the methods in the main part, and can be used to protect the micro-service systems.

References

1. Matt C. Digital Transformation Strategies / C. Matt, T. Hess, A. Benlian. // Business & Information Systems Engineering. – 2015. – №57. – С. 339–343.

2. Newman S. Building Microservices / Sam Newman. – Sebastopol, CA: O'Reilly Media, Inc., 2015. – 472 с.

3. Evaluating the monolithic and the microservice architecturepattern to deployweb applications in the cloud [Електронний ресурс] / V.Mario, O. Garcés, H. Castro,S. Gil.–2015.–Режим доступу до ресурсу: https://www.researchgate.net/publication/304317852_Evaluating_the_monolithic_and_the_microservice_architecture_pattern_to_deploy_web_applications_in_the_cloud.

4. Wootton B. Microservices-Not A Free Lunch! [Електронний ресурс]/Benjamin Wootton.–2014.–Режим доступу до ресурсу: http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html.

5. Y. Sun, S. Nanda and T. Jaeger,"Security-as-a-Service for Microservices-Based Cloud Applications," 2015 IEEE 7th International Conference on CloudComputing Technology and Science (CloudCom), Vancouver, BC, 2015, pp. 50-57.

6. Mirko Novakovic. Introducing Dynamic Focus for Application Performance Management [Електронний ресурс] / Mirko Novakovic. – 2017. – Режим доступу до ресурсу: https://www.instana.com/blog/introducing-dynamic-focus-application-performance-management/.

7. E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3 / E. Rescorla. – Mozilla: IETF, 2018. – 160 с.

8. В. Л. Цирлов. Основы информационной безопасности автоматизированных систем / В.Л. Цирлов., 2008. – 119 с.

9. D. Hardt, Ed. The OAuth 2.0 Authorization Framework / D. Hardt, Ed.., 2012. – 76 с.

10. Sebastián E. Peyrott. The JWT Handbook / Sebastián E. Peyrott., 2017. – 85 с.

11. TOTP: Time-Based One-Time Password Algorithm / D. M'Raihi, S. Machani, M. Pei та ін.]., 2011. – 16 с.

12. HOTP: An HMAC-Based One-Time Password Algorithm / D. M'Raihi, M. Bellare, F. Hoornaert та ін., 2005. - 37 с.

Autors

Minkov Kostiantyn – student of Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: k.minkov-2017@kpi.ua

Міньков Костянтин Павлович – студент кафедри обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Viktor Selivanov – associate professor, Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

Селіванов Віктор Левович – доцент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

Volokyta Artem – associate professor, Department of Computer Engineering, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute".

E-mail: artem.volokita@kpi.ua

Волокита Артем Миколайович – доцент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського».

РОЗШИРЕНА АНОТАЦІЯ

К. П. Міньков,
В. Л.Селіванов, А. М. Волокита

СИСТЕМА ЗАХИСТУ МІКРОСЕРВІСНИХ СИСТЕМ

Актуальність теми дослідження. Завдяки швидкому впровадженню систем мікросервісу між різними компаніями електронної комерції (Netflix, Amazon, Hailo) виникла потреба в засобах їх ефективного захисту.

Аналіз останніх досліджень і публікацій. Незважаючи на велику кількість робіт і досліджень, присвячених системам захисту, захист самих мікросистем досить докладно описана в літературі.

Виявлення недосліджених частин загальної проблеми. Інтеграція відомих методів захисту в когерентну систему. Створення системи захисту, орієнтованої на мікросервіси.

Цілі дослідження. Метою даної роботи є систематизація відомих методів захисту та їх консолідації для створення єдиної системи в контексті мікросистемних систем.

Викладення основного матеріалу. Визначено причини появи та розвитку систем мікросервісу, їх переваги та базові уразливості. В якості методів безпеки розглядаються такі стандарти, як HTTPS, JWT, OAuth2, RM, TOTP. Результатом є система, реалізована в Java відповідно до стандартів програмування.

Висновки. Результатом розробки та розробки є система, що включає в себе основні методи і може бути використана для захисту систем мікросервісу.