Toward Software-Defined Moving Target Defense for Secure Service Deployment Enhanced with a User-Defined Orchestration

Ki-Wan Kang
Department of Computer and Information Security, Sejong University
Seoul, Republic of Korea
kkwan0226@gmail.com

Ki-Woong Park*
Department of Computer and Information Security, Sejong University
Seoul, Republic of Korea
woongbak@sejong.ac.kr

ABSTRACT

In recent years, cloud native computing, which involves the deployment of scalable applications enhanced with containers, microservices, and serverless functions, has been actively studied to maximize its efficiency, flexibility, and economic feasibility. In this regard, studies on the security of the cloud native computing environment have been conducted. Among various studies on the security of these systems, moving target defense (MTD), which is an area of research that blocks various security threats in advance by actively changing the main properties of the protected target to deceive attackers, has been actively studied and developed. However, cloud native computing is highly dynamic; it is difficult to apply MTD technologies that actively change static system properties. Therefore, a software-defined MTD framework was designed for easier application of MTD technology to the cloud native environment. In this study, the user-defined adaptability of the software-defined MTD framework was implemented, and it was verified that the properties of the target service were changed according to previously defined mutation properties.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

moving target defense, software-defined, cloud native computing, user-defined adaptability

ACM Reference Format:
Ki-Wan Kang and Ki-Woong Park. 2020. Toward Software-Defined Moving Target Defense for Secure Service Deployment Enhanced with a User-Defined Orchestration. In *2020 ACM International Conference on Intelligent Computing and its Emerging Applications (ACM ICEA '20), December 12–15, 2020, GangWon, Republic of Korea.* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.1145/3440943.3444725

*Corresponding author

ACM ICEA '20, December 12–15, 2020, GangWon, Republic of Korea

ACM ISBN 978-1-4503-8304-2/20/12.
https://doi.org/10.1145/3440943.3444725

1 INTRODUCTION

In recent years, cloud native computing, which combines container technologies and microservice architectures in a cloud computing structure, has been recognized as an excellent technology to maximize the efficiency, flexibility, and economic feasibility of computing processes [7, 19]. As such, studies on the security of cloud native computing environments have been actively conducted [8, 9, 20]. Because cloud native computing maintains the initial system properties, attackers may be given enough time to obtain information required to analyze the vulnerability of the target system. To mislead the attacker, research and development of moving target defense (MTD) systems are being actively conducted, which is a method that proactively blocks various security threats by actively changing the main properties (e.g., network, platform, runtime environment, software, and data) of the protected system [12]. In particular, among various MTD technologies, network-based MTD technology is known to be very effective because it can reverse the attack–defense asymmetry by making the first reconnaissance step of the attack difficult. As a result, this technology is emerging as a very effective defense technology [23].

Cloud native computing can provide high efficiency compared to the existing cloud by combining container technology and microservice architecture [14, 21]. However, it is difficult to apply MTD technology to a highly dynamic cloud-native environment as it changes the main properties of the system in which a target is operating. Furthermore, there is a limit to the applications of existing MTD technologies due to the interactions between independent microservices in the cloud native environment.

Therefore, in this study, the requirements of a software-defined MTD framework were derived to facilitate the application of existing and future MTD technologies. First, this framework should easily accommodate and consider current and future MTD studies. Second, the administrator must be able to define the mutation elements of the system to be protected, and they must be able to actively redefine the mutation elements according to the situation. Third, the software-defined MTD technology should not be dependent on a specific system but must be compatible with and possess the ability to be used in various systems. Based on the derived requirements, a software-defined MTD framework consisting of an MTD repository for a deployable MTD, a mutation master module for user-defined adaptability, mutation connector module for an interoperable MTD, and a designed mutation agent module.

Among the software-defined MTD frameworks designed using an effective defense technique (a network-based MTD technology), the accuracy of user-defined adaptability was verified. A dashboard was developed for verification, and the Apache service was used for

the accuracy verification of user-defined adaptability through the dashboard. As a result of this verification, it was confirmed that the properties of the service change according to the defined mutation elements, network class, and hopping cycle. Furthermore, it was confirmed that the request was not made to the existing IP and ports after the mutation.

The remainder of this paper is structured as follows. Section 2 summarizes the background knowledge and presents the limitations of existing studies through the analysis of MTD-related studies. Section 3 describes the derivation of the requirements for and design of a software-defined MTD framework. Section 4 verifies the accuracy of the proposed framework in terms of user-defined adaptability. Finally, in Section 5, the conclusions of this study and future research directions are presented.

2 BACKGROUND AND RELATED WORK

In this section, background knowledge on the cloud native environment and MTD technology to which the software-defined MTD framework is applied is explained, and the existing research and development MTD technology is examined. Thus, the limitations of existing studies are presented.

2.1 Background

2.1.1 Cloud Native Computing. In recent years, cloud native computing, which combines container technologies and microservice architectures in a cloud computing structure, has been recognized as an excellent technology to maximize the efficiency, flexibility, and economic feasibility of cloud computing processes [13]. Compared to virtual machines (VMs), containers require fewer system resources to execute services, demonstrate faster startup times, and provide excellent input/output (I/O) throughput [16]. At the same time, by packaging the application code and the property of dependency together, container management platforms (e.g., Docker [16] and Kubernetes [3]) can provide a consistent environment for application development, testing, and production [4]. Because of these properties, containers are not dependent on any particular environment and can be operated in various environments. Microservice architectures can make the target system very efficient by reducing unnecessary resource use because the entire application does not have to be scaled out for specific and load-intensive services. In particular, because resources can be allocated and scaled out according to the properties of a service, efficient resource use is possible [2].

2.1.2 MTD. MTD technology prevents various security threats in advance by actively changing the main properties of the protected target system (such as the network, platform, runtime environment, software, and data) to reverse the attack–defense asymmetry [22]. In 2009, the Networking and Information Technology Research and Development (NITRD) program explicitly emphasized the concept of MTD because it can utilize existing resources to increase effectiveness and efficiency [10]. Since this time, the MTD research community has been formed and has attracted attention due to the approach it utilizes, as well as its advantages [5]. The objective of the MTD research community is to increase the uncertainty and complexity of the system for the system attacker such that the chances of target identification (e.g., vulnerable system elements) are reduced, as well as to increase expenses for initiating attacks or scans (e.g., reconnaissance attacks). The MTD can be classified according to the system properties, as shown in Figure 1 [17].

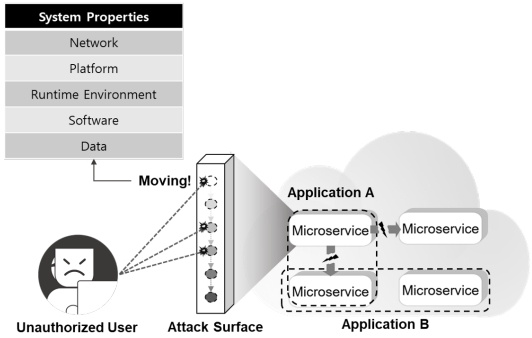

Figure 1: Apply MTD to cloud native environment

2.2 Related Work

2.2.1 Random Port and Address Hopping (RPAH). Luo et al. [15] proposed a mechanism that constantly changes the IP addresses and communication ports for unpredictability.

2.2.2 Random Host Mutation (RHM). Al-Shaer et al. [1] proposed a mechanism that assigns a virtual IP address using low-frequency mutation (LFM), which actively changes the range of the virtual IP addresses for the protected target system. High-frequency mutation (HFM) is also utilized, which allocates IP addresses within the virtual IP address range specified by the LFM.

2.2.3 Decoy-Based MTD. Clark et al. [6] introduced an approach involving the use of numerous decoy systems to prevent attackers from targeting the actual system. They also proposed a method to change the network addresses of the decoy systems along with the system network.

2.2.4 Host IDEntify (HIDE) Anonymization. Jafarian et al. [11] proposed an additional honeypot cloud operation to the RHM model proposed by Al-Shaer et al. [6]. A mechanism was proposed to prevent attackers from identifying the network properties of the target system. The honeypot cloud and the attacker's target system are located in different networks, and if suspicious traffic is found through the MTG in the external network, the proposed system redirects the corresponding traffic to the honeypot cloud.

2.2.5 Ghost MTD (gMTD). Park et al. [18] proposed a protocol mutation mechanism using a previously shared one-time bit sequence (OTBS). Only users who are aware of the protocol variation pattern can communicate with the service module of the server system. Other user messages are redirected to the decoy-hole module to mislead the attacker's successful system penetration.

Among various MTD technologies, network-based MTD technology is one of the most effective defense methods. Analyzing the existing network-based MTD technologies reveals limitations when these technologies are applied to actual systems. First, it is necessary to implement additional functions and introduce systems for each MTD technology. With the development of MTD technology,

each time a newly researched and developed MTD technology is applied, the implementation of the additional functions and introduction of a new system are very inefficient. Second, it is difficult to apply MTD technology to a system because of the dynamics of the cloud native environment. In the case of existing MTD technologies, only the interactions between the user and the server system is considered; the interactions between independent microservices is not considered.

3 SOFTWARE-DEFINED MTD FRAMEWORK

In this section, the software-defined MTD framework requirements are derived, and the software-defined MTD framework is designed based on the derived requirements.

3.1 Derived Requirements

3.1.1 Deployability of MTD. The existing and future MTD studies and technologies should be easily accommodated and applied to the cloud native computing environment through the use of software-defined MTD.

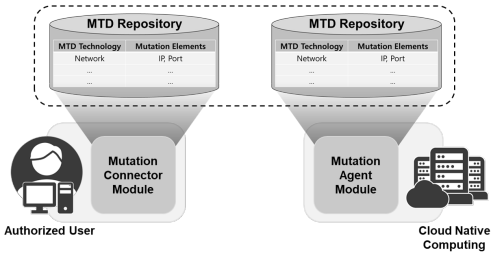

Figure 2: Deployment guarantee through MTD Repository

3.1.2 User-Defined Adaptability. The administrator must be able to define the mutation elements of the system to be protected, and they must be able to actively redefine the mutation elements according to the situation. In addition, the user can define a valid range for the defined mutation element, and the mutation element must be continuously changed according to a certain period or a randomly defined period.

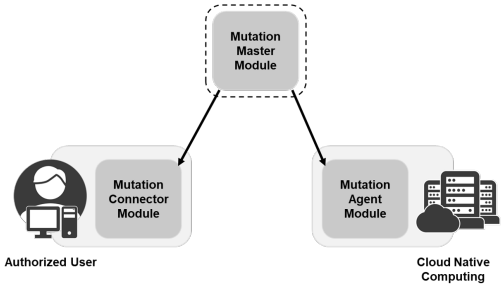

Figure 3: Module management through Mutation Master

3.1.3 Interoperability of MTD. The MTD technology defined by software is not dependent on a specific system and must be compatible with and able to be used in various systems. The system should be able to apply and manage various MTD technologies defined by the software.

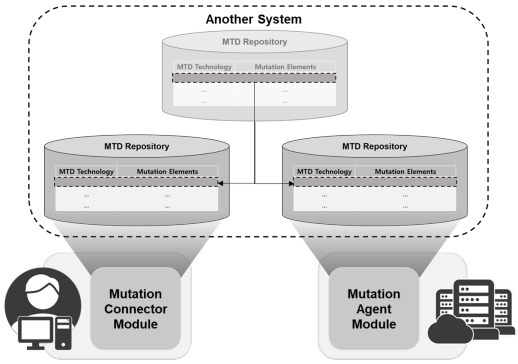

Figure 4: MTD technology developed in a specific system can be used in various systems

3.2 Software-Defined MTD Framework Design

3.2.1 Design for Ensuring MTD Deployment. The framework should be designed such that existing and future MTD technologies can be easily accommodated and applied in cloud native computing systems. To achieve this, the MTD technology that will be applied to a module performing mutation on various mutation elements is software-defined and stored in the MTD repository

3.2.2 Design for Ensuring User-Defined Adaptability. The manager should be able to define the applied MTD technology, effective range for the mutation elements, and mutation cycle. To achieve this, a dashboard was developed through which the MTD mechanism to be applied was defined via a mutation master module, and the effective range and mutation cycle for mutation elements were defined. Furthermore, the MTD technology defined in two modules between the service user and the protected target system (mutation connector module and mutation agent module), the effective range for the mutation elements, and the mutation cycle are transmitted to enable continuous communication.

3.2.3 Design for Ensuring Interoperability. The software-defined MTD framework should not be dependent on a specific system and should be compatible with different types of systems. To this end, two modules (a mutation connector module and a mutation agent module) are deployed between the service user and the protected target system. As a result, the software-defined MTD framework applies and manages the MTD technology stored in the MTD repository existing in each module.

4 IMPLEMENTATION AND EXPERIMENT

In this section, the accuracy of the user-defined adaptability of the proposed software-defined MTD framework is verified. For verification, a dashboard is developed and an experiment is conducted.

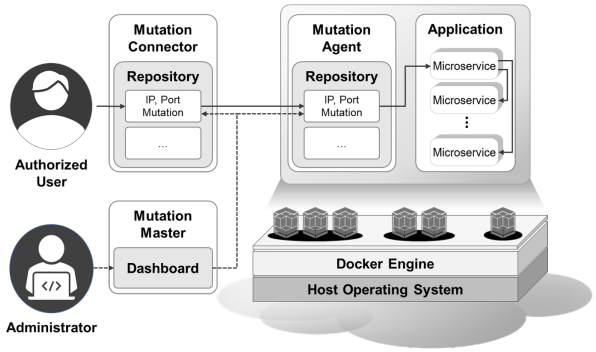

Figure 5: An example of software-defined MTD framework

4.1 Implementation

The accuracy of the proposed MTD framework in terms of user-defined adaptability was verified through network-based MTD technology. For verification, a dashboard was developed, and the features of this dashboard are:

- To define mutation properties (including the mutation elements, network class, and hopping cycle) of existing network-based MTD technologies
- To evaluate the situation and status of currently operating services
- To evaluate the results of applying the MTD and the corresponding logs through a terminal

Figure 6: Dashboard of mutation master module

4.2 Experiment

4.2.1 Experiment Environment. Experiments were conducted using a server equipped with an Intel (R) Core ™ i7-8700 and 32GB RAM. For this experiment, a virtual machine consisting of 2 vCPUs, 4.0 GB RAM, and Ubuntu 18.04 LTS was created.

4.2.2 Verification of User-Defined Adaptability. Using the developed dashboard, the accuracy of the user-defined adaptability of the Apache service was verified. Through the dashboard, after inputting Seed for IP and Port mutation of Apache service, network class selection and mutation cycle were specified. The results of the experiment confirmed that the properties of the service changed according to the defined mutation elements, network class, and hopping cycle. Moreover, it was confirmed that the request was not made to the existing IP and ports after the mutation.

5 CONCLUSION

In recent years, cloud native computing, which combines container technologies and microservice architectures in a cloud computing structure, has been recognized as an excellent technology to maximize the efficiency, flexibility, and economic feasibility of computing processes. However, as the dynamics of cloud native computing are increased due to the container technologies and microservice architectures, it is difficult to apply MTD technologies as an active defense mechanism. Therefore, this paper presented a software-defined MTD framework to easily apply existing and future MTD technologies to cloud native computing environments.

The proposed software-defined MTD framework is composed of an MTD repository, mutation agent module, mutation connector module, and mutation master module.

The MTD repository stores existing and future MTD technologies. The mutation agent module and mutation connector module are located between a user and a service to accommodate the MTD technologies targeting various system properties. They enable the mutation of various system properties without being dependent on a specific environment. The mutation master module enables the system to be deployed within a valid range of system properties, and it can actively change the system properties by considering limited resources.

Among various MTD technologies, network-based MTD technology is one of the most effective defense methods. The accuracy of user-defined adaptability was verified using a network-based MTD technology. A dashboard was developed for verification, and accuracy verification of the user-defined adaptability of the Apache services was performed using the dashboard. The results of the experiment confirmed that the properties of the service changed according to the defined mutation elements, network class, and hopping cycle. Furthermore, it was confirmed that the request was not made to the existing IP and ports after the mutation.

Future studies should be conducted concerning the implementation of the proposed software-defined MTD framework, measurement of overhead (which occurs when the framework is applied to a cloud native computing environment), and accuracy verification of this framework.

ACKNOWLEDGMENTS

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grants funded by the Korean government (MSIT) (No. 2018-0-00420 and No. 2019-0-00426) and supported by National Research Foundation of Korea (NRF) grants funded by the Korean government (NRF2020R1A2C4002737)

REFERENCES

[1] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. 2012. Random host mutation for moving target defense. In *International Conference on Security and Privacy in Communication Systems*. Springer, 310–327.

[2] Nuha Alshuqayran, Nour Ali, and Roger Evans. 2016. A systematic mapping study in microservice architecture. In *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE, 44–51.

[3] David Bernstein. 2014. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing* 1, 3 (2014), 81–84.

[4] Eric A Brewer. 2015. Kubernetes and the path to cloud native. In *Proceedings of the sixth ACM symposium on cloud computing*. 167–167.

[5] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J Moore, Dong Seong Kim, Hyuk Lim, and Frederica F Nelson. 2020. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 709–745.

[6] Andrew Clark, Kun Sun, and Radha Poovendran. 2013. Effectiveness of IP address randomization in decoy-based moving target defense. In *52nd IEEE Conference on Decision and Control*. IEEE, 678–685.

[7] Dennis Gannon, Roger Barga, and Neel Sundaresan. 2017. Cloud-native applications. *IEEE Cloud Computing* 4, 5 (2017), 16–21.

[8] Xing Gao, Zhongshu Gu, Mehmet Kayaalp, Dimitrios Pendarakis, and Haining Wang. 2017. ContainerLeaks: Emerging security threats of information leakages in container clouds. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 237–248.

[9] Somya Garg and Satvik Garg. 2019. Automated cloud infrastructure, continuous integration and continuous delivery using docker with robust container security. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 467–470.

[10] AK Ghosh, D Pendarakis, and WH Sanders. 2009. Moving target defense co-chair's report-National Cyber Leap Year Summit 2009. *Tech. Rep., Federal Networking and Information Technology Research and Development (NITRD) Program* (2009).

[11] Jafar Haadi Jafarian, Amirreza Niakanlahiji, Ehab Al-Shaer, and Qi Duan. 2016. Multi-dimensional host identity anonymization for defeating skilled attackers. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense*. 47–58.

[12] Sushil Jajodia, Anup K Ghosh, Vipin Swarup, Cliff Wang, and X Sean Wang. 2011. *Moving target defense: creating asymmetric uncertainty for cyber threats*. Vol. 54. Springer Science & Business Media.

[13] Nane Kratzke and Peter-Christian Quint. 2017. Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software* 126 (2017), 1–16.

[14] David S Linthicum. 2017. Cloud-native applications and cloud migration: The good, the bad, and the points between. *IEEE Cloud Computing* 4, 5 (2017), 12–14.

[15] Yue-Bin Luo, Bao-Sheng Wang, Xiao-Feng Wang, Xiao-Feng Hu, Gui-Lin Cai, and Hao Sun. 2015. RPAH: Random port and address hopping for thwarting internal and external adversaries. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. IEEE, 263–270.

[16] Dirk Merkel. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux journal* 2014, 239 (2014), 2.

[17] Hamed Okhravi, Thomas Hobson, David Bigelow, and William Streilein. 2013. Finding focus in the blur of moving-target techniques. *IEEE Security & Privacy* 12, 2 (2013), 16–26.

[18] Jun-Gyu Park, Yangjae Lee, Ki-Wan Kang, Sang-Hoon Lee, and Ki-Woong Park. 2020. Ghost-MTD: Moving Target Defense via Protocol Mutation for Mission-Critical Cloud Systems. *Energies* 13, 8 (2020), 1883.

[19] Zhiming Shen, Zhen Sun, Gur-Eyal Sela, Eugene Bagdasaryan, Christina Delimitrou, Robbert Van Renesse, and Hakim Weatherspoon. 2019. X-containers: Breaking down barriers to improve performance and isolation of cloud-native containers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 121–135.

[20] Byungchul Tak, Canturk Isci, Sastry Duri, Nilton Bila, Shripad Nadgowda, and James Doran. 2017. Understanding security implications of using containers in the cloud. In *2017 {USENIX} Annual Technical Conference ({USENIX}{ATC} 17)*. 313–319.

[21] Kennedy A Torkura, Muhammad IH Sukmana, and Christoph Meinel. 2017. Integrating continuous security assessments in microservices and cloud native applications. In *Proceedings of the10th International Conference on Utility and Cloud Computing*. 171–180.

[22] Jianjun Zheng and Akbar Siami Namin. 2019. A survey on the moving target defense strategies: An architectural perspective. *Journal of Computer Science and Technology* 34, 1 (2019), 207–233.

[23] Yuyang Zhou, Guang Cheng, Shanqing Jiang, Ying Hu, Yuyu Zhao, and Zihan Chen. 2019. A cost-effective shuffling method against DDoS attacks using Moving Target Defense. In *Proceedings of the 6th ACM Workshop on Moving Target Defense*. 57–66.