inventions

MDPI

Article

Method of Forming Various Configurations of Telecommunication System Using Moving Target Defense

Anatoly V. Ryapukhin *, Evgeny O. Karpukhin and Ivan O. Zhuikov

Moscow Aviation Institute (National Research University), Volokolamskoe Highway 4, 125993 Moscow, Russia
* Correspondence: ryapukhin.a.v@mail.ru

Abstract: The purpose of this paper is to improve the effectiveness of the Moving Target Defense (MTD)-based protection method, which reduces the problem of using traditional network protection tools due to the static nature of network services and configurations. Options for solving the problems of choosing an adequate time interval for activating the technology of MTD and its application in networks are given. A new approach is proposed, which consists in creating a set of system configurations and changing it when an attack is detected and determined. The design implementation was tested on a network model using software defined networks (SDN). The advantages of the proposed method are highlighted, among which the most significant are: simple operation scheme, ability to deploy the system without the use of software-defined networks and absence of violations of security policies within the system.

Keywords: cybersecurity; moving target defense (MTD); networks; software defined networks (SDN); HoneyPot

Citation: Ryapukhin, A.V.; Karpukhin, E.O.; Zhuikov, I.O. Method of Forming Various Configurations of Telecommunication System Using Moving Target Defense. *Inventions* **2022**, *7*, 83. https://doi.org/10.3390/inventions7030083

Academic Editor: Konstantinos G. Arvanitis

Received: 8 August 2022
Accepted: 12 September 2022
Published: 16 September 2022

1. Introduction

The main security problem of traditional networks is the fact that the defender always has to come second to the attacker. An attacker can carefully plan their actions in advance by conducting network reconnaissance and preparing the necessary tools. To solve this fundamental problem, MTD was developed, the main idea of which is to make the elements of the network move, thus reducing the effectiveness of intelligence for an attacker. The problem with using MTD is updating and maintaining predefined security policies.

The object of study of this work is the process of ensuring security in software-defined networks using MTD. The subject of the study is the very principles of decision-making and methods for setting the time interval for their adoption.

The purpose of the study is to improve the effectiveness of the protection system for telecommunication networks that use MTD, taking into account the problems of managing security policies.

In accordance with the goal, the following tasks were solved in the work:

1. Review of the technology of MTD and software-defined networks has been carried out, and current approaches to ensuring information security with their application have been considered;
2. Model of a software-defined network has been developed for further testing of the proposed approach;
3. Design implementation of the method for the formation of various configurations of a telecommunications system using MTD has been developed;
4. Developed design implementation of the method was tested on a previously prepared model of a software-defined network.

2. Theoretical Basis

This section provides a comprehensive description of the concepts used in this paper. MTD is considered in depth, and the main strategies and approaches used with it in

telecommunication systems are described. The architecture of software-defined networks and how they are related to MTD is also described. The problematic situation of security policy conflict is considered. A review of existing approaches to the solution is made, and a hypothesis for an accessible and effective solution to the problem is formulated.

2.1. MTD in Telecommunication Networks

MTD is designed to solve the problem of traditional network defenses that do not take into account the inherent advantage of attackers due to the static nature of network services and configurations. Time is good for a cybercriminal: he can conduct network reconnaissance and then carefully plan his attack. MTD solves this problem with two approaches: dynamic reconfiguration of the protected system or response to an emerging threat and taking countermeasures, which leads to an increase in information entropy for the attacker. The use of this technology significantly reduces the advantage of intelligence over time, which the attacker has in relation to traditional defense mechanisms [1]. Thus, the information collected by the attacker in the exploration phase will become obsolete during the attack if the defender has switched to a new configuration during this time. The most complete description of this technology can be found in [2,3].

When building a system using MTD, there are three main problems to solve [4]:

1. Choice of system elements that should be given dynamism;
2. Establishing a time interval when it is necessary to take any action;
3. Implementation of decision-making principles in the system.

Strategies using MTD in telecommunications systems can be roughly divided into three categories:

1. MTD at the network level, which changes the way it operates. For example, using IP-hopping technique, in which IP address changes periodically, or using random port numbers, fake hosts, etc.;
2. MTD at the host level is directed to its changes, for example, to periodically change the configuration or name;
3. MTD at the application level changes their types, versions, randomizes the address space layout randomization and source code with compilation processes.

Thus, MTD system, $\sum$, is the ordered set consisting of (σ, G, P), where σ is the configurable system, G is the set of operational and security goals, and P is the security policies. Then σ in turn is the set of (S, A, τ), where $S = (s_1, s_2, \ldots, s_n)$ is the set of system states in which it can be, $A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is the set of actions to take, and $\tau : S \times A \rightarrow S$ is the system state transition function. The system state s is the unique assignment of the value z from the configuration parameter type P to the configuration parameter π. The type of the configuration parameter P denotes the range of possible values that the configuration parameter π can take. The configuration parameter π can take on a value based on its configuration type P to define configuration details. An example host P configuration is shown in the diagram in Figure 1.

2.2. SDN

SDN is a network architecture that separates the control plane and data plane of a network device. While the data plane is still on the device, the control plane is transferred to the centralized controller. The control plane and the data plane interact through software interfaces, which allows consistent and comprehensive management of the entire network, regardless of the underlying technology used in the network. Visualization of the network architecture is shown in Figure 2.

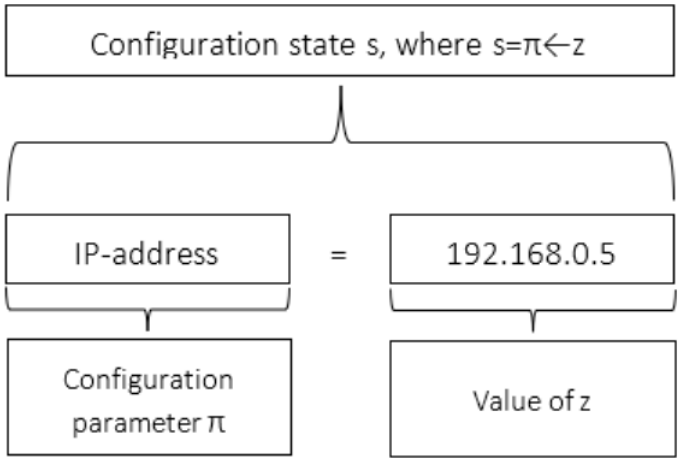

Figure 1. Host P configuration example.

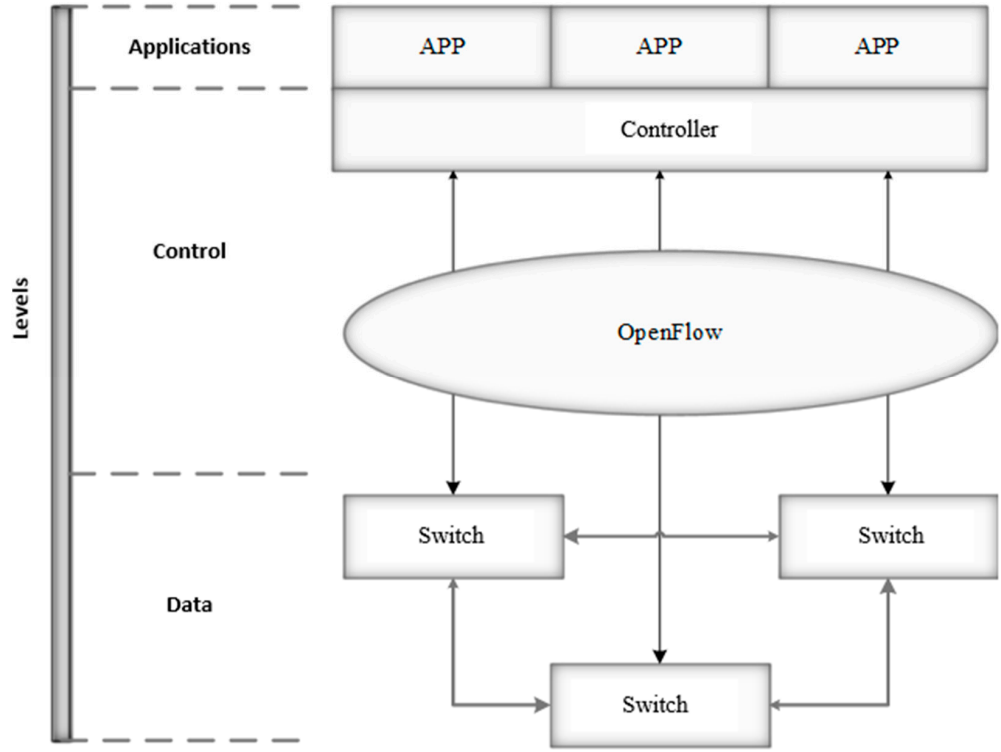

Figure 2. SDN architecture (dotted lines mean the separation of planes of different levels; lines with arrows show the directions of interaction between network components; APP—application).

2.3. Security Policy Conflict

For clarity and subsequent detailing of the proposed solution to the problem posed, we describe the scenario for applying the technology of MTD.

In the network shown schematically in Figure 3, upon detecting an attack directed from host A to host P, MTD technology takes retaliatory measures, which consist of performing the following actions:

1. New web server (host SP) is created that handles the load of host P;
2. IP address of host P is transferred to the controlled Honeypot network [5,6] and assigned to host H.

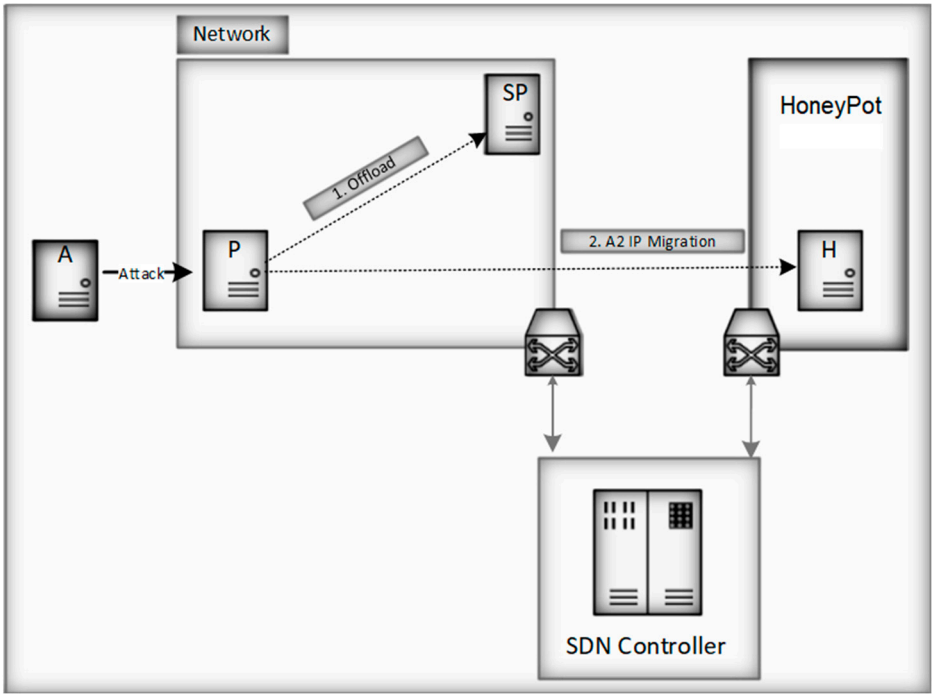

Figure 3. Security policy conflict model (A, P, SP, H) are names of hosts; numbers show the sequence of actions).

To analyze and minimize possible damage from an attacker, the HoneyPot network allows all incoming traffic but does not allow outgoing traffic to other sections of the network. As a result of the performed actions, new rules are entered into the forwarding table:

1. Allow all incoming traffic to IP addresses that originally belonged to host P but now belong to host H;
2. Change address for incoming packets from host P to host SP;
3. Stop all outgoing traffic from IP address that previously belonged to host P, but now belongs to host H, to the rest of the data center;
4. Allow traffic from port 443 to host SP.

This creates a conflict between the original policy allowing only port 443 to IP address of host P and the new set of rules that allow all incoming traffic to the IP address of host H [7].

In addition, in [4], researchers come to similar conclusions, that is, the problem of conflict resolution exists, and it needs special attention.

2.4. Problem Area

To solve the problem of setting a time interval when it is necessary to take any action, there are two approaches: to constantly switch between different states of the system or to use a certain feature to move between configurations (variable approach). When constantly switching between system states, the constant t is used, which determines the time interval when it is necessary to take any action. One example of the practical implementation of this approach would be the use of a virtual IP that changes at a fixed time interval in a predefined or randomized manner [8]. With a variable approach, protection can be implemented in three ways:

1. The first way is to react to events. An example of such an event could be the detection of an attack, the unavailability of a server or a channel. An example of work using this approach is [9], where the authors take action after they detect suspicious incoming packets;

2. The second way is more intellectual. With it, a predetermined time constant t is not known, but instead a time interval from 0 to tmax is determined, and the defenders need to determine the time value from the interval $0 \leq t \leq tmax$. For example, in [10], the idea of the state of the system is used to find the optimal value of time t, which determines when the administrator should take active steps to change the system configuration;
3. The third way is a hybrid of constant and variable approaches. The combination of these methods at the time of writing is poorly understood, but the example of [11], where the authors used constant randomization of IP addresses in conjunction with the creation of false nodes in the network when an attack was detected, and their positive result suggests that this concept is worth studying further.

To solve the problem of implementing the principles of decision making in the system, defenders have two approaches:

1. To think through actions based only on the current state of the system. For example, in [12], M. Thompson et al. propose a method that consists in rotating operating systems and thereby improves system security;
2. To take into account the future development of events, predicting the actions of the attacker and the defender many steps ahead. R. Colbaugh and K. Glass [13] use the co-evolutionary relation between attackers and defenders to develop methods for predicting and countering attacks, as well as to limit the extent to which adversaries can learn about defense strategies.

Due to the new realities in the Russian Federation, all the above approaches cannot be used before passing the analysis and certification. With the current increased demand for import substitution of smart information security systems, it is necessary to develop new promising approaches to improve the effectiveness of the security process in telecommunication networks. When developing such approaches, it is necessary to take into account the problem of emerging security policy conflicts.

3. Methodology

In this section, the hypothesis is formed, and the tool is considered, which will be further used to test the design implementation. A network model of a complex with software-defined networks is being built.

3.1. Various System State Configurations

The hypothesis on which further work is based on the fact that it is possible to think over a set of optimal system states S for each class of attacks. For example, in the case of a direct attack on a host, it is beneficial to perform the actions described in "Attack Scenarios" or otherwise to keep the security inside the system and neutralize the attack. Thus, the system will apply actions from some pre-designed set of decisions A in a form randomized for the attacker but predetermined for the defenders. Accordingly, for each set of solutions, their own sets of rules for the internal functioning of the system P will be prescribed in order to avoid violation of security policies and all the negative consequences that come from this, which include the loss of legal traffic within the system and the emergence of new vulnerabilities.

Returning to the previously described model of the attacker, when implementing the approach described earlier, from the point of view of the defender, the situation will look like this:

1. The system was in its standard configuration S_0;
2. The operator or intrusion detection system detected an attack on one of the hosts;
3. Depending on the class of a certain attack, MTD decided to perform one of the options for pre-defined optimal actions α_1;
4. Along with the actions taken, the policy of the internal functioning of P system has also changed;

5. Attack is neutralized. The system is now operating with the new configuration s_1.

As a result, all incoming traffic was allowed, but outgoing traffic to the address of the host now located in HoneyPot network was denied; the address of the attacked host was changed to a new one, and traffic from port 443 was allowed for it. There are no violations of security policies; the load from the host is processed on another web server, and the attacker's actions are neutralized and can be analyzed.

3.2. Graphical Network Simulator-3 (GNS3)

The approach proposed earlier will be tested using GNS3 tool [14]. The choice in favor of this network emulator was made, guided by the following principles:

1. The tool should demonstrate realistic network behavior;
2. To conduct security tests, it is necessary to be able to use real network security or introduce testing tools into the network;
3. It is necessary to be able to flexibly configure all the necessary parameters and network elements.

Network simulators OMNET++ [15] or ns-3 [16] cannot be taken into account due to inconsistency with the second point. GNS3, which is widely used to create, design and test a network in a virtual environment, offers an easy way to design and create networks of any size without the need for hardware and meets all the requirements put forward. GNS3 is highly flexible and can handle most network tasks. It supports various types of virtualized devices and can be easily administered using a graphical interface. A graphical user interface (GUI) may be installed remotely from the real environment, which may run on a different computing platform and thus may use, for example, cloud computing resources.

The above properties allow creating a network topology model without resorting to hardware implementation in order to test the proposed approach.

3.3. Network Model of a Complex with Software-Defined Networks

In the previously described GNS3, a software-defined network model was built. Hosts actively participating in the trial were installed with ParrotOs [17], which is a Linux distribution based on Debian with a focus on computer security. It is designed for introduction testing, vulnerability assessment, mitigation and anonymous web browsing. This choice was made due to the many built-in introduction tools, system and network monitoring tools. To further test the previously proposed method, we need to consider three states of this model.

In the first state, shown in Figure 4, the system is at rest. There is one working host: Host_P and HoneyPot, necessary for the initial confrontation with a potential attacker. They are located on different subnets and are connected to a single SDN controller using switches as intermediate devices. It is worth noting that on this and following images of the model there are designations e0, e1, f0/0, f1/0 and f0/1, indicating the connection interface numbers; they are added to simplify the configuration and work with the model and not important in the context of this work.

The next step is the appearance of an attacker. The mechanism for the appearance of a malicious participant in the network process is beyond the scope of this work. At the current moment, the MTD system has not reacted to this change in any way. The state of the model is shown in Figure 5. The attacker launches an attack on host P, for example by performing DoS attack. The attack class is not important in this trial, since it does not affect the demonstration of the general concept presented.

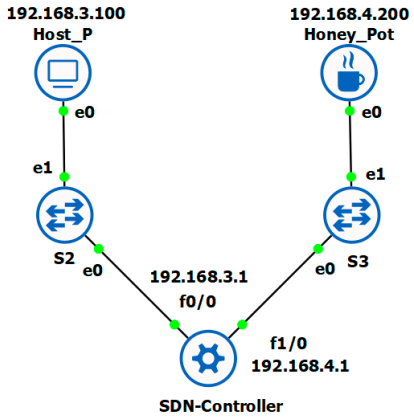

Figure 4. Network model in normal state.

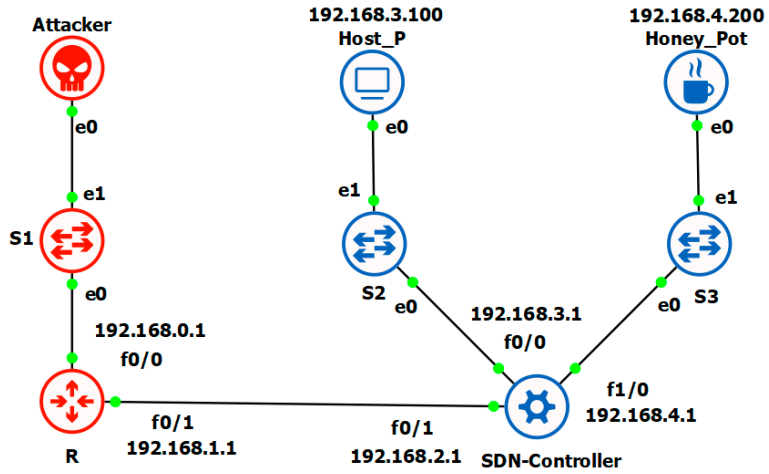

Figure 5. Network model with attacker.

As a result of the attack, the resources of the host P become overloaded, and it cannot continue to work. The third and final stage is the system's predetermined response to an attack. These actions are described in detail earlier and in our model will look as it is shown in the Figure 6.

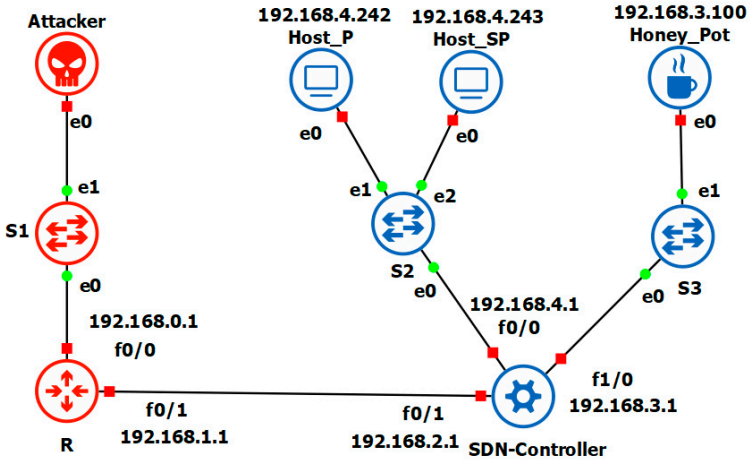

Figure 6. Network model after attack response.

As we can see from the Figure 6; MTD responded in a predetermined format by creating an additional host on the network to take over the load of the attacked host; and

the IP address of the attacked host was assigned to HoneyPot for further analysis of the attacker's actions. At the same time, since this set of actions was thought out in advance, the new rules were automatically applied, and the legitimate participants in the network process did not lose anything.

4. Results and Discussion

This section summarizes the testing of the proposed approach. Conclusions are formulated about the applicability of software-defined networks. Prospects for the development of the proposal disclosed above are noted.

4.1. Result of Testing the Design Implementation

After the attack began, the host stopped performing its tasks; however, when implementing the method of various configurations and executing a predetermined response to such a threat, the system was reconfigured and continued to operate in normal mode, and the attacker's actions were neutralized using HoneyPot.

To assess the attacker's impact on the host, hping3 utility [18] was used to create and send custom ICMP/UDP/TCP packets. Using the limited resources of the device on which the model was launched in the emulator, it turned out to create an average load of ≈ 1950 Kb/s. After performing the described actions and transferring the payload to another host, the effect of the attack on it was naturally absent. Only legitimate network members continued to interact with it and, accordingly, the traffic was not zero. The result of this experiment is shown in Figure 7.

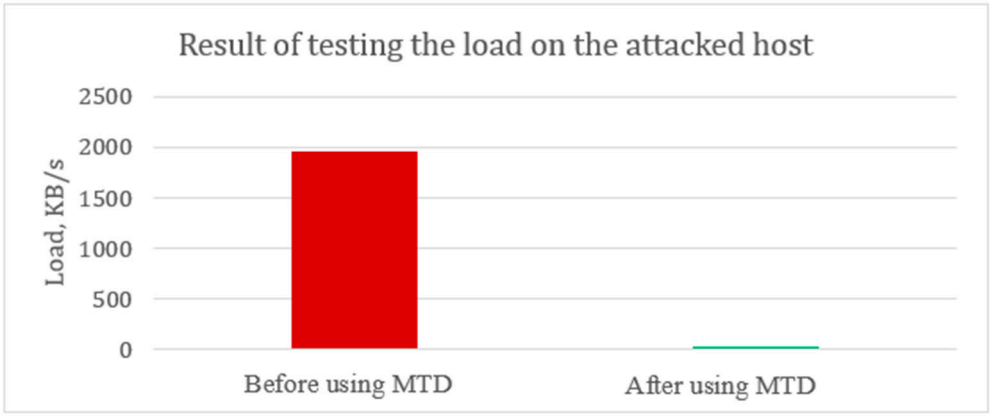

Figure 7. Result of testing the load on the attacked host (red color means there is a significant load on the attacked host; green color means no load on the attacked host).

4.2. Assessment of Applicability and Development Prospects

Software-defined networks are excellent for implementing the ideas of MTD [19–21] due to their flexibility, the ability to control the flow of data in the network [22] and the availability of open programmable interfaces such as OpenFlow [23]. It is on this environment that the main emphasis will be placed in the further study of the effectiveness of the proposed approach.

An alternative approach, which consists in using middleboxes [24], i.e., devices for performing network functions, also has a number of advantages, including ease of implementation, use and management. At the time of writing, much less attention was paid to it in research in the context of using it to implement MTD systems due to its disadvantages: high probability of misconfiguration and overloads, leading to an increase in the cost of using and maintaining these devices [25]. The described model of pre-created system states in theory solves the problem of middlebox configuration and, therefore, eliminates the main negative feature of this approach.

Thus, at the time of writing, it is not clear which environment will be optimally suited for implementation, and therefore, in the future, research should be directed to identifying the most appropriate implementation options.

5. Conclusions

This work is devoted to the study of existing approaches to solving two of the three main problems in creating a network structure using MTD. Some of the most appropriate solutions to these problems have been described. An attacker model was built, which was subsequently used to test the design implementation of the proposed solution.

A method is proposed that consists in creating a set of solutions for each class of attacks. When detecting and determining the type of attack, MTD selects one of the prepared options and executes it, while it also contains new security policies within the system to avoid violating them. The advantages of this method are its simple scheme of work, the potential for deployment of the system without the use of software-defined networks and the absence of violations of security policies within the system. The hypothetical application of this method in ensuring information security in other areas, for example, in protecting web resources is also described.

The approach described above was implemented on a network model using a GNS3 emulator with software-defined networks. In addition, an alternative approach using middleboxes was considered, which had not received much attention previously due to a number of shortcomings, which the described approach can level.

Further research will be aimed at developing the proposed approach, compiling responses to the main classes of attacks and analyzing their effectiveness. It will also be analyzed to improve it with the addition of a hybrid system for changing the state of the system, in which mobility is applied to several elements of the system at once.

Author Contributions: Conceptualization, A.V.R., E.O.K. and I.O.Z.; methodology, A.V.R.; software, I.O.Z.; validation, E.O.K.; formal analysis, E.O.K.; investigation, A.V.R.; resources, E.O.K. and I.O.Z.; data curation, A.V.R. and E.O.K.; writing—original draft preparation E.O.K. and I.O.Z.; writing—review and editing, A.V.R., E.O.K. and I.O.Z.; visualization, E.O.K.; supervision, A.V.R.; project administration, A.V.R.; funding acquisition, I.O.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lei, C.; Zhang, H.Q.; Tan, J.L.; Zhang, Y.C.; Liu, X.H. Moving Target Defense Techniques: A Survey. *Sec. Commun. Netw.* **2018**, *2018*, 1–25. [CrossRef]
2. Jajodia, S.; Ghosh, A.K.; Swarup, V.; Wang, C.; Wang, X.S. *Moving Target. Defense. Creating Asymmetric Uncertainty for Cyber Threats*; Springer: London, UK, 2011.
3. Jajodia, S.; Ghosh, A.K.; Subrahmanian, V.S.; Swarup, V.; Wang, C.; Wang, X.S. *Moving Target. Defense II. Application of Game Theory and Adversarial Modeling*; Springer: London, UK, 2013.
4. Sengupta, S.; Chowdhary, A.; Sabur, A.; Alshamrani, A.; Huang, D.; Kambhampati, S. A survey of moving target defenses for network security. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1909–1941. [CrossRef]
5. Pouget, F.; Dacier, M. Honeypot-based forensics. In Proceedings of the AusCERT Asia Pacific Information Technology Security Conference, Brisbane, Australia, 23–27 May 2004.
6. Oosterhof, M. Cowrie Honeypot. 2014. Available online: https://github.com/micheloosterhof/cowrie (accessed on 1 August 2022).
7. Chowdhary, A.; Pisharody, S.; Huang, D. SDN based scalable MTD solution in cloud network. In Proceedings of the 2016 ACM Workshop on Moving Target Defense, Vienna, Austria, 24 October 2016; pp. 27–36.

8. Al-Shaer, E.; Duan, Q.; Jafarian, J.H. Random host mutation for moving target defense. In Proceedings of the International Conference on Security and Privacy in Communication Systems, Berlin, Germany, 3 September 2012; pp. 310–327.
9. Zhao, Z.; Liu, F.; Gong, D. An SDN-based fingerprint hopping method to prevent fingerprinting attacks. *Sec. Commun. Netw.* **2017**, *2017*, 1–12. [CrossRef]
10. Lei, C.; Ma, D.H.; Zhang, H.Q. Optimal strategy selection for moving target defense based on Markov game. *IEEE Access* **2017**, *5*, 156–169. [CrossRef]
11. Clark, A.; Sun, K.; Bushnell, L.; Poovendran, R. A game-theoretic approach to IP address randomization in decoy-based cyber defense. In Proceedings of the International Conference on Decision and Game Theory for Security, London, UK, 4 November 2015; pp. 3–21.
12. Thompson, M.; Evans, N.; Kisekka, V. Multiple OS rotational environment an implemented moving target defense. In Proceedings of the 2014 7th International Symposium on Resilient Control Systems (ISRCS), Denver, CO, USA, 19–21 August 2014; pp. 1–6.
13. Colbaugh, R.; Glass, K. Predictability-oriented defense against adaptive adversaries. In Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul, Korea, 14–17 October 2012; pp. 2721–2727.
14. The Software that Empowers Network Professionals. Available online: https://gns3.com (accessed on 1 August 2022).
15. OMNeT++ 6.0. Available online: https://omnetpp.org (accessed on 1 August 2022).
16. NS-3 Network Simulator. Available online: https://www.nsnam.org (accessed on 1 August 2022).
17. Parrot Security Os. Available online: https://www.parrotsec.org (accessed on 1 August 2022).
18. Hping3. Available online: https://www.kali.org/tools/hping3 (accessed on 1 August 2022).
19. Jafarian, J.H.; Al-Shaer, E.; Duan, Q. Openflow random host mutation: Transparent moving target defense using software defined networking. In Proceedings of the First Workshop on Hot Topics in Software Defined Networks, Helsinki, Finland, 13 August 2012; pp. 127–132.
20. Chowdhary, A.; Pisharody, S.; Alshamrani, A.; Huang, D. Dynamic game based security framework in SDN-enabled cloud networking environments. In Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, Scottsdale, AZ, USA, 24 March 2017; pp. 53–58.
21. Debroy, S.; Calyam, P.; Nguyen, M.; Stage, A.; Georgiev, V. Frequency-minimal moving target defense using software-defined networking. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), Kauai, HI, USA, 15–18 February 2016; pp. 1–6.
22. Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2014**, *103*, 14–76. [CrossRef]
23. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comp. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
24. Carpenter, B. Middleboxes: Taxonomy and Issues. Available online: https://datatracker.ietf.org/doc/html/rfc3234 (accessed on 1 August 2022).
25. Sherry, J.; Ratnasamy, S. A Survey of Enterprise Middlebox Deployments. Technical Report No. UCB/EECS-2012-24. 2012. Available online: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-24.html (accessed on 1 August 2022).