Proactive Detection for Countermeasures on Port Scanning based Attacks

E.S. Sagatov*, S. Mayhoub†, A.M. Sukhov*, F. Esposito‡, P. Calyam§

* HSE University, Tallinskaya, 34, Moscow, 123458, Russia

†Samara National Research University, Moskovskoe sh. 34, Samara, 443086, Russia

‡ Computer Science Department, Saint Louis University, 3450 Lindell Blvd, St. Louis, MO, 63103, USA

§ University of Missouri-Columbia, 221 Naka Hall, Columbia, MO 65211, USA

***Abstract*—Defending a cyber asset from a targeted attack based on port scanning is a challenging task because attackers exploit protocol behavior essential for productive use of applications. For instance, TCP or UDP ports opened for applications such as file transfer or video can be exploited to launch denial of service attacks. There is a need for proactive methods that can detect port scanning based attacks at their initial stage so that counter-measures can be initiated before the attack impact is disruptive on a cyber asset. This paper presents methods to counteract the initial stages of network attacks involving TCP and UDP port scanning. Our methods analyze outgoing traffic to identify ICMP 3.3 and TCP RST response packets that indicate the beginning of an attack launch. We specifically describe two countermeasures based on software-defined networking controller (at the network level) and Linux utility (at the host level) modules we developed. To validate the effectiveness of our proactive detection based methodology, we set up a testbed with a scheme of a polygon and conducted experiments related to distortion of the port status of attacks. Our results demonstrate that our approach is effective and the accuracy of determining open TCP ports did not exceed 15%, and it did not reach 2% for the remaining ports (closed TCP, UDP of any type).**

***Keywords*—SDN, port scaning attack, intrusion detection, IDS**

I. Introduction

Large-scale interventions in network infrastructure have become the norm in the modern world. Increasingly, threats about network attacks are occurring at the interstate level. Modern network attacks [1] are complex, consist of several stages, and are capable of causing damage at a high-scale.

However, many types of network-based attacks that involve some type of recognition of the target start the same way. At the first stage, the reconnaissance of the network connections of a targeted information system is carried out to collect as much information as possible in order to devise a robust attack. It includes port scanning in the early steps [2]. Simultaneously, attackers look for open ports; according to their numbers and configuration, they can infer the operating system used and the types of software installed. Based on this data, attackers choose their attack models. By completing the reconnaissance stage, system administrators can significantly reduce the risks [3]. To this aim, this paper is devoted to the design and implementation of countermeasures for port scanning.

Snort [4] is designed to protect large networks, hence it is not intended for personal end-point protection, for example at the level of a smartphone device. To obtain the necessary information, attackers must test all ports, with both TCP and UDP packets. By capturing responses from closed ports, it is possible to form simple qualification attributes of the initial stage of an attack called “port scanning”. As opposed to prior works that analyze incoming traffic [4], [5], the novelty of our approach involves analysis of outgoing traffic i.e., packets sent in response to attack related requests.

The search for such qualifications [6] is one of the goals of this study. Once these qualifications are formulated, we can begin to develop measures (e.g., software, policies) that will counteract attacks at the intelligence gathering stage. Qualification attributes for recognizing port scans can be formulated as simple statistical rules for all outgoing traffic. Currently, there are two main technologies for blocking traffic from an IP address. These are firewall rules (iptables) and SDN. Our software solution to secure against port scanning attacks has two versions. The first version is the software-defined networking (SDN) module [7] [8] that acts as a counter measure at the network level. The second implementation is a direct-action utility for Linux operating systems i.e., it acts as a countermeasure at the host level. The overall protection solution can be unified and is based on the found qualification attributes. Our solution approach involves checking the health of the system to be carried out experimentally.

II. Related Work and Qualification Attributes of the Initial Phase of a Network Attack

Our approach has several features; in this section, we highlight the work that most closely relates to our design. The work in [9] presents a statistical cross-relation to detect network scans and determine its sources. The proposed approach uses TCP RST packets to detect sequential TCP scans and ICMP type 3 packets (port unavailability) to detect serial UDP scans. This approach is closest to what we have proposed, where we focus on the organization of protection technologies. The authors in [10] detail different types of port scans and discuss features of different scans in order to suitably analyze the response packets. A mathematical model is constructed using this data that can help in anomaly detection events related to port scans. Their related algorithm is found to be useful for detecting port scans and allows identification of port scanning as well as the source. In addition, their algorithm identifies the IP address of the attacker performing the scan.

In the field of network security, intrusion detection systems such as e.g., Snort have been developed to protect network assets [11]. The rule-based method in [12] is effective to detect real-time port scan attacks, proactive techniques have been found to be more appropriate to cope with modern attacks [13]. In [5], a universal SDN module is described, which involves countering DDoS attacks and port scanning.

Reliable and straightforward protection is also required in terms of user security and privacy relating to smart/networked consumer electronics devices. Such vulnerable devices are identified by using a serial port scan. In [14], the authors focused on the problem of detecting port scans within home networks. They proposed a SDN firewall platform that can detect scans. In [15], the authors conduct the first systematic study of the use of open ports on a mobile platform and outlined their security implications. To achieve this goal, OPAnalyzer, a static analysis tool, was developed and implemented. In a subsequent vulnerability analysis, the authors found that almost half of the applications are not protected and can be remotely used.

An attacker, as a rule, begins an invasion by searching for open ports, as well as examining responses to various types of ICMP requests. As a rule, ports are tied to Internet services in a standard way, obtaining a list of open ports allows us to assume what software is installed on this resource. Due to this, an attacker can make a list of possible vulnerabilities for applications installed on the attacked server.

Qualification attributes can be sought for both attacking packets and response packets to attacking requests. We consider the appearance of response packets of certain types as qualifying attributes. The header of such packets contains information about the sources of the attack. For scanning UDP ports, such a qualification attribute is a response containing an ICMP packet of type 3 code 3. If such a response appears several times and is directed to the same external IP address, all traffic from this IP address should be blocked.

During any scanning of closed ports by TCP packets, the qualification attribute of scanning is the packet with the RST flag from the attacked server. When such a packet reappears, traffic from the recipient address should be blocked.

The qualification attributes for identifying an attacking server can be formulated as two points:

(a) The protected server must send at least three ICMP type 3 code three or TCP packets with the RST flag to the address of the attacking server, and
(b) The attack must be exposed to 2 different TCP or UDP ports.

If both of these conditions are met, then the traffic from attacking IP address must be blocked for a short time.

The critical point is accessing the second port, after which all traffic from this address will be blocked. The number of attacking addresses in everyday use can be estimated from our data collected from trap servers [16]. This value can reach 10 requests per second, or 3000 requests in 5 minutes. Modern smartphones can easily cope with this task.

III. Security Software and Testing Procedure

Based on qualification attributes, algorithms for counteracting network intrusions of various types have been developed. These algorithms form the basis of the developed software in this work. Different categories of consumers require software of varying performance. The highest performance is achieved using SDN technology. Therefore, the first implementation of our defense mechanism is available as an SDN module.

At the other extreme of consumers are network users with smartphones, laptops, and other smart devices. They do not need high-performance defense mechanisms [17]. The main requirement for personal protective software is the low demand for computing resources. These include processor consumption and RAM. Therefore, the second implementation is made in the form of a simple utility that runs on Linux operating systems.

If the corresponding bits indicate that the outgoing packet is a TCP packet with the RST flag or an ICMP packet of type 3 with code 3, then the destination address and source port are extracted from this packet, and unique counters increase the value by one. The first counter shows the number of packets sent to a fixed IP address, the second counter shows how many ports were requested from this address. After the first counter reaches three, and the second two, the iptables utility stops receiving any incoming packets from this address for 5 minutes (300 seconds).

Our development is not limited only to such a Linux utility that blocks port scanning. In the following, we present a specialized module for the Floodlight SDN controller that we developed. This module is designed to block traffic from IP addresses from which ports are scanned. The module code is written in Java. The bloking rules are based on the qualification attributes formulated in Section II. The created rules are uploaded to the switch with FlowMod messages.

To conduct our tests on the developed protective mechanisms, a specialized polygon training ground was built. This polygon features a segment of a local area network connected to the Internet, in which a specialized server is installed. It is thus able to provide virtual machines on public IP addresses.

During testing, an *nmap* server located outside the local network was used to scan TCP and UDP ports of the protected resource. Each scan is done twice. During the first test, the defense mechanisms are disabled; the second test was carried out with the protection.

Tests were conducted separately for the Linux utility and the SDN module. The first group of experiments involves testing the ports of a particular server running OC Linux on which the developed utility was installed. The *nmap* scanning utility was launched from an IP address from the global network.

A particular server is installed as an object of protection, on which 100 TCP and UDP ports are open. The decision about which ports to open was made based on data from trap servers [16]. The selection criterion is based on popularity, that is, those TCP and UDP ports were selected, to which there was the largest number of calls.

Snort is an open-source, network-based intrusion prevention (IPS), and detection system (IDS). It can analyze traffic in IP networks in real-time. *Snort* has a built-in *sfPortscan* software module that can detect port scans of the protected system based on information about server connection requests and responses to these requests.

We also tested antivirus software developed by the Kaspersky Lab. We have chosen the most widespread information security product in Russia. This tool is also highly prevalent in the Middle East, Africa, and South America. Most importantly, the Kaspersky company regularly releases reports describing the main threats and statistics on their use, which are often cited in research on network security. Kaspersky Total Security version 20.0.14.1085 is relevant at the time of the experiment, with updated threat databases via the Internet. It detects port scanning attempts and prohibits further interaction with the attacking computer. All possible protections, network protection, and all protocols possible for the program are activated in the settings.

IV. Performance Evaluation

A. UDP Port Scan Results

UDP scanning is performed by sending a UDP packet without a load on each scanned machine's port. Based on the response, or lack thereof, the port is assigned to one of four states. Receiving any UDP response from the scanned port means that the port is open. Receiving an ICMP packet of type 3.3 means that the UDP port is closed. It should be noted that the lack of response means either an undefined state or an open port.

Scanning is limited to two states (closed, filtered) if the security utility is disabled (see Table I). Enabling the security utility limits the receipt of any responses from the scanned server. If no response is received, then *nmap* classifies this state as undefined (filtered). When protection is enabled, the vast majority of ports (65165 out of 65536) are recognized as undefined.

TABLE I
Utility Test Results

Test mode	UDP			TCP		
	open	closed	filtered	open	closed	filtered
No security utility	0	65 435	101	99	65 436	1
Security utility activated	0	1 333	64 203	14	1262	64 260

The data in Table I indicates that the developed utility radically distorts the data on the status of TCP and UDP ports. In the case of UDP ports, it is almost impossible to determine which ports are open. We can say for sure that 2% of the ports are closed. The developed utility makes it possible to correctly determine an available port's status for only 14% of TCP ports. At the same time, the status of the closed port is confirmed only for 1.9% of scanned TCP ports.

The graph from Fig. 1 shows the comparative dynamics of scanning closed UDP ports for all three security mechanisms (Kaspersky, Linux utility, and SDN module). In this graph, the abscissa represents the number of UDP ports scanned. The ordinate axis shows the number of closed UDP ports whose status is determined correctly.

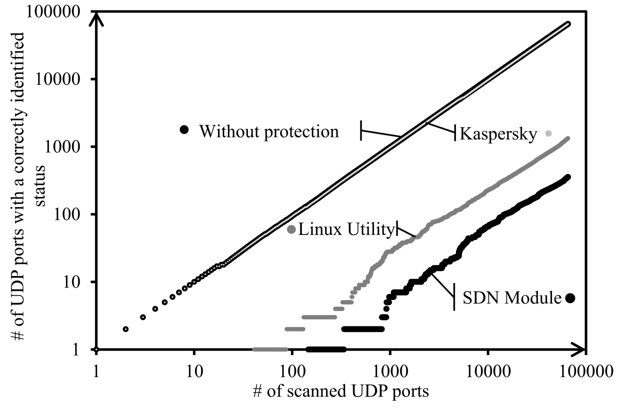

Fig. 1. Results of Scanning Closed UDP Ports (less is better). The Abscissa is the Number of Scanned Ports, the Ordinate is the Number of Ports with a correctly defined status

The graph shows that Kaspersky Total Security protects a small number of the most critical ports from scanning, while the security tools we developed allow us to repeatedly reduce the accuracy of determining the protected server's configuration.

B. TCP Port Scan Results

TCP scanning begins by sending a TCP packet with the SYN flag. In response, *nmap* receives SYN+ACK and then terminates the connection using RST. Based on the response packet or its absence, a conclusion is made about the port's status. It can be in one of three states (open-closed-filtered). Receiving a response in a TCP packet with SYN/ACK flags from the scanned port means that the port is open. Receiving a TCP response with RST flags from the scanned port implies that the port is closed. The absence of a response or the receipt of an ICMP packet of any type means that the TCP port is in an undefined state (filtered). Test results for TCP ports with Linux utility turned on and off can be found in the Table I.

Figures 2 and 3 show the comparative dynamics of scanning open and closed TCP ports for three security mechanisms (Kaspersky, Linux utility, and SDN module).

In the graph from Fig. 2, the abscissa axis shows the number of TCP ports scanned. The number of closed TCP ports is delayed on the ordinate axis, whose status is determined correctly. We can see from Fig. 2 that the Kaspersky software protects against scanning a small number of the most important TCP ports, as in UDP testing.

In the graph from Fig. 3, the abscissa axis shows the number of TCP ports scanned. The number of open TCP ports

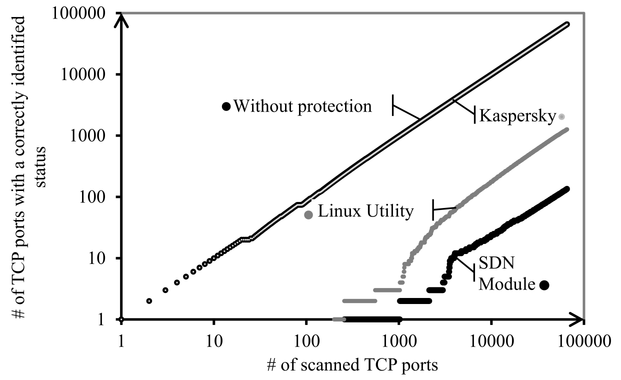

Fig. 2. Closed TCP Ports Scan Result (less is better). The Abscissa is the Number of Scanned Ports, the Ordinate is the Number of Ports with a correctly defined status

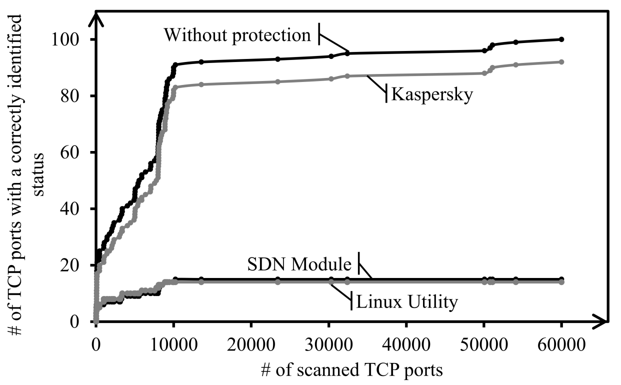

Fig. 3. Results of scanning open TCP ports (less is better). The Abscissa is the Number of Scanned Ports, the Ordinate is the Number of Ports with a correctly defined status

is delayed on the ordinate axis, whose status is determined correctly during the scanning process. We can see from the graph that the effectiveness of our defense mechanisms for TCP ports is lower than the same for UDP ports. However, both the Linux utility and the SDN module significantly reduce the number of TCP ports with a correctly defined status compared to Kaspersky. The corresponding test results of the SDN module are shown in Table II.

TABLE II
Test Results of the SDN Module

Test mode	UDP			TCP		
	open	closed	filtered	open	closed	filtered
No security utility	0	65 435	101	99	65 436	1
SDN module activated	0	357	65 171	15	1262	64 260

Scanning with the module turned off made it possible to determine whether a TCP or UDP port is open or closed with high accuracy. During testing, only one error was detected for both types of ports. The activation of the protective SDN module allowed to reduce the scanning accuracy by many times. So, the accuracy of determining open TCP ports was 15%. For closed UDP ports, the status is correctly determined only in 0.5% of cases. For closed TCP ports, this accuracy is even lower at 1.9%. Note that the SDN module's efficiency is three to four times higher than that of the Linux utility. Specifically, it allows us to accurately set the status for the number of ports, which is three to four times less than with protection using the Linux utility.

C. Comparison of Developed Defense Mechanisms with Snort

To understand how effectively our developed defense mechanisms work, we conducted additional comparative testing with *Snort*'s *sfPortscan* module. Three defense mechanisms were consistently tested during this experiment:

- Linux utility developed by us
- our SDN module
- Snort (with sfPortscan module)

Each defense mechanism allowed us to find the number of ports that were scanned. Thus, it was possible to evaluate the accuracy of attack detection for any of the three defense mechanisms listed above. Data on the number of detected ports that are scanned is shown in Figs. 4 and 5.

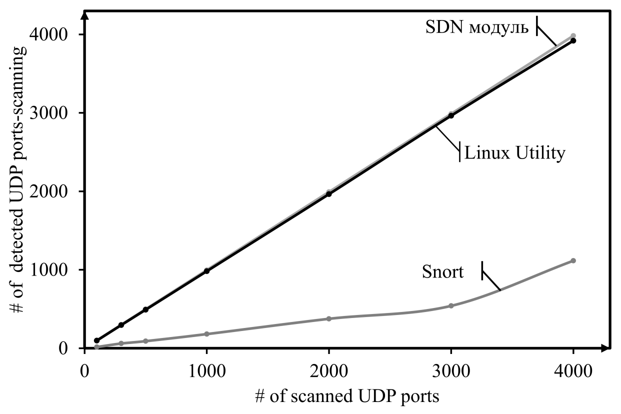

Fig. 4. Results on the Number of attacked UDP ports, the status of which was determined correctly (more is better)

The abscissa in Fig. 4 shows the number of UDP ports (N) that have been scanned. This number is set by *nmap* settings. The ordinate shows the number of ports (S), scanning of which detected the defense mechanism. Similarly, the abscissa in Fig. 5. shows the number of TCP ports (N) that have been scanned. This number is set by *nmap* settings. The ordinate shows the number of ports (S), scanning of which detected the defense mechanism. We can see from these graphs that the protective mechanisms that we have proposed are more effective i.e., they determine most of the tested ports compared to that of Snort.

V. Conclusion

In this paper, a novel method was presented to counteract the initial stage of any network attack. At this stage, the TCP and UDP ports are scanned, and it is determined which of them are open. This information allows attackers to collect general information about the attacked server, the type of operating system, installed software, remote control features, etc.

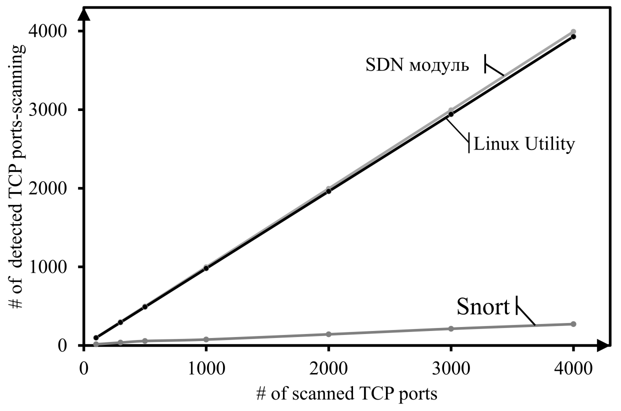

Fig. 5. Results on the Number of attacked TCP ports, the status of which was determined correctly (more is better)

To develop algorithms that determine the IP addresses from which the scan is being conducted, qualification attributes were formulated by studying outgoing traffic from the scanned servers. When scanning UDP ports, closed ports were found to send ICMP packets of type 3 code 3. A distinctive feature of TCP port scanning is the response in TCP with RST flag packets. The number of such response packets is considered separately for each IP address. If the number of packets of any of these types reaches three and two different ports are scanned, traffic from this IP address will be blocked for 5 minutes.

These qualification attributes were the basis for our developed algorithms for specialized software systems. Two different software products were developed using the algorithms. One of these products is implemented as a Linux utility. The other is an SDN module. We remark that the Linux utility is not demanding on computing resources, but it copes with protecting a separate information server, user workstation, or mobile device. Separately, we highlight the possibility of installing our utility on smart edge devices. As a rule, these devices are based on Advanced RISC Machine (ARM) technologies, so the installation of full-fledged protective systems is rather difficult.

In order to understand how effectively the developed defense mechanisms work, we conducted additional comparative testing with four security mechanisms: a Linux utility developed by us, our SDN module, Kaspersky Total Security and Snort with the sfPortscan module. During testing, each defense mechanism allowed us to find the number of ports that were scanned. Thus, attack detection accuracy was estimated for the four defense mechanisms listed above. The counteraction mechanisms developed by us were shown to be superior to traditional defense mechanisms that involve: Kaspersky Total Security and Snort (with the sfPortscan module).

When using the Linux utility, the accuracy of determining open TCP ports did not exceed 14%, and for other ports (closed TCP, UDP of any type) it did not reach 2%. When using the SDN module, the accuracy of determining open TCP ports did not exceed 15%, the number of closed TCP ports is correctly determined in less than 2% of cases. For UDP ports, the accuracy slightly exceeded 0.5%.

Acknowledgment

This study has been supported by the Russian Foundation for Basic Research (RFBR): grant 20-37-90002. The publication was prepared within the framework of the Academic Fund Program at the National Research University Higher School of Economics (HSE) in 2021 (grant № 21-04-030).

References

[1] P. Mell, "Understanding intrusion detection systems," in *IS Management Handbook*. Auerbach Publications, 2003, pp. 409–418.
[2] A. Sridharan, T. Ye, and S. Bhattacharyya, "Connectionless port scan detection on the backbone," in *2006 IEEE International Performance Computing and Communications Conference*. IEEE, 2006, p. 10.
[3] J. Gadge and A. A. Patil, "Port scan detection," in *2008 16th IEEE International Conference on Networks*. IEEE, 2008, pp. 1–6.
[4] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.
[5] C. Birkinshaw, E. Rouka, and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks," *Journal of Network and Computer Applications*, vol. 136, pp. 71–85, 2019.
[6] B. Morin and H. Debar, "Correlation of intrusion symptoms: an application of chronicles," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 94–112.
[7] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2015.
[8] P. Calyam, S. Rajagopalan, A. Selvadhurai, S. Mohan, A. Venkataraman, A. Berryman, and R. Ramnath, "Leveraging openflow for resource placement of virtual desktop cloud applications," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 311–319.
[9] M. Anbar, A. Manasrah, and S. Manickam, "Statistical cross-relation approach for detecting tcp and udp random and sequential network scanning (scans)," *International Journal of Computer Mathematics*, vol. 89, no. 15, pp. 1952–1969, 2012.
[10] E. V. Ananin, A. V. Nikishova, and I. S. Kozhevnikova, "Port scanning detection based on anomalies," in *2017 Dynamics of Systems, Mechanisms and Machines (Dynamics)*. IEEE, 2017, pp. 1–5.
[11] C. Joshi and U. K. Singh, "Information security risks management framework–a step towards mitigating security risks in university network," *Journal of Information Security and Applications*, vol. 35, pp. 128–137, 2017.
[12] S. K. Patel and A. Sonker, "Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort," *International Journal of Future Generation Communication and Networking*, vol. 9, no. 6, pp. 339–350, 2016.
[13] S. Marchal, J. François, R. State, and T. Engel, "Phishstorm: Detecting phishing with streaming analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.
[14] S. Shirali-Shahreza and Y. Ganjali, "Protecting home user devices with an sdn-based firewall," *IEEE Transactions on Consumer Electronics*, vol. 64, no. 1, pp. 92–100, 2018.
[15] Y. J. Jia, Q. A. Chen, Y. Lin, C. Kong, and Z. M. Mao, "Open doors for bob and mallory: Open port usage in android apps and security implications," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2017, pp. 190–203.
[16] E. S. Sagatov, D. Shkirdov, and A. M. Sukhov, "Analysis of network threats based on data from server-traps," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2019, pp. 1–5.
[17] W. M. Fitzgerald, U. Neville, and S. N. Foley, "Mason: Mobile autonomic security for network access controls," *Journal of Information Security and Applications*, vol. 18, no. 1, pp. 14–29, 2013.