APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

Port Knock Method and RST Cookies on the UbuntuOS Linux Server Security System

Darmono1, Amikom University Purwokerto, darmono9296@gmail.com

Jumadi Mabe Parenreng2, State University of Makassar, jparenreng@unm.ac.id

ABSTRACT

The increase in online-based services makes a security system for server computers increasingly needed. A server computer is a device that must always be available to be accessed anytime and anywhere. Some of the security systems needed for server computers include security for ssh port access for remote server access needs and a security system to protect servers from Denial of Service (DoS) attacks which can make the server down and completely inaccessible. In this study, a security system is proposed for a server computer with the Linux UbuntuOS operating system on a port 22 secure shell(ssh) access system using the port knock method and a security system to prevent Denial of Service (DoS) attacks on the server computers using the RST Cookies method. The simulation results from the port 22 Secure Shell (SSH) access experiment for the server computer can work well where port 22 Secure Shell(SSH) will remain closed and cannot be accessed carelessly except by accessing several ports first according to predefined port knocking rules. , the implementation of a security system with the RST Cookies method works very well to prevent Denial of Service (DoS) attacks and can still keep the server accessible with a good response time of under 1 ms.

Keywords *: Security, Linux UbuntuOS, Port Knock, RST Cookies*

1. INTRODUCTION

Commented [at1]: minimum 10 pages

Security is an important aspect of an information system. Unfortunately, this security issue often receives little attention from owners and managers of information systems. According to GJ Simons, information security is how we can prevent fraud or, at least, detect fraud in an information-based system, where the information itself has no physical meaning [1].

Information technology security refers to efforts to secure information technology infrastructure from disturbances in the form of prohibited access and unauthorized network utilization [2]. One of the most important parts of an information system infrastructure is the server. The security aspect is an important factor to pay attention to on a server computer because various external attacks are often launched by exploiting weaknesses in the server. Securing the server computer is as important as securing the website or web application itself and the surrounding network. As in research conducted by Ramli Ahmad et al entitled "Use of

Backpropagation Methods in Artificial Neural Networks for Intrusion Detection Systems" [3]. In this research, researchers designed a security network for an Intrusion Detection System (IDS) using the Backpropagation method in artificial neural networks. From this research, it was found that this security system is very useful in helping detect abnormal activity in a network, which is very helpful in terms of cyber security. When creating a system that involves a server, sometimes there is negligence in adding a security system for the server, which can result in losses if illegal actions occur on the server computer. If we have a secure web application and an unsafe server computer or vice versa, then this is still very risky to the system[4]. There are several things that need to be considered as the basic basis for securing a server computer, including removing unnecessary services, making good server access rules, separating development/testing/production environment settings, deleting modules that are unused or do not meet your needs. install several required security patches and diligently scan the services running on the server computer to ensure all services are running properly and as needed [5].

In this research, it is proposed to create an additional security system for a server computer with the aim of securing the Secure Shell (SSH) port, which is the main gateway for accessing a server computer. Apart from that, the security system to keep the server stable and running well also applies a security system with RST Cookies as a technique to prevent large-scale attacks which aim to make the server computer go down or cannot be accessed normally.

2. METHODOLOGY

In carrying out this research, there are several stages that will be passed. The research stages that will be carried out during the research process are as follows.

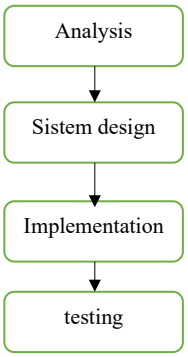


Figure 1. Research stages

Figure 1 shows the stages carried out in this research starting from analyzing research needs, creating security system designs and simulations, implementing security methodologies and testing the results of implementing these methods [6].

1. Analysis

The analysis phase is essential in understanding and evaluating the overall system by breaking it down into smaller, more manageable components. System analysis allows developers to examine the structure of the system in detail, identifying potential flaws, bottlenecks, or inefficiencies [7]. By thoroughly investigating each part, developers can pinpoint areas that need improvement or redesign. This process enables them to plan more effectively for future development, ensuring that all elements of the system work together cohesively. System analysis also plays a vital role in recognizing risks and obstacles that could impact the overall functionality and success of the project [8].

In addition to system analysis, requirements analysis is equally important. It focuses on determining the specific needs the system must meet to achieve its objectives. This involves defining both the software and hardware requirements necessary for the system's operation. Through this analysis, developers establish the system's desired outcomes, ensuring that all components are in place to meet user expectations. This phase also includes identifying performance benchmarks and any constraints related to the system's implementation, providing a clear roadmap for the development team to follow [9].

2. System Design

System design is the next critical stage where the theoretical aspects of the system begin to take shape [8]. This phase involves defining the functional and technical requirements based on the previous analysis. It is a detailed process that requires designers to create plans and models that will guide the development and implementation [8]. Through diagrams, flowcharts, and blueprints, the system design process maps out how different components—both hardware and software—will interact. This design serves as a guide to ensure all elements are compatible and work efficiently as a unified system.

Additionally, system design also addresses configuration and architecture. Designers consider how different system components should be organized to achieve optimal performance [10]. Whether it is software modules, databases, or physical hardware, every part must be arranged in a way that supports the system's overall functionality. This step is vital in ensuring that the system not only works efficiently but is also scalable, flexible, and easy to maintain. Good system design lays the groundwork for successful implementation, helping developers avoid complications later in the project [11].

3. Implementation

Once the system design is finalized, the implementation phase brings the plan into reality. At this stage, developers begin the process of coding, configuring, and setting up the actual components of the system [12]. For example, when creating a security system for a server, this is the point where security protocols are configured, and hardware and software components are installed. The implementation stage marks the point where the abstract design turns into an operational system that can perform real-world tasks. Developers must follow the design specifications closely to ensure the system functions as intended [13].

During the implementation phase, it is critical to test individual components to ensure they integrate smoothly. Developers may encounter challenges or unforeseen issues that require

adjustments [14]. This iterative process ensures that the final system is free of major bugs or vulnerabilities. In the case of a server security system, the implementation process would involve setting up specific configurations, such as access controls, firewalls, and encryption methods, all of which are crucial for safeguarding sensitive data and maintaining system integrity [15].

4. Testing

The testing phase is the final step in the system development lifecycle, where the functionality and reliability of the implemented system are thoroughly examined. In this stage, the newly built system is subjected to a series of tests designed to uncover any issues or vulnerabilities [16]. These tests ensure that the system meets its performance benchmarks and that all components are working as expected. Testing is particularly important for security systems, as it verifies whether the system is adequately protected against potential threats, such as unauthorized access or cyber-attacks. Developers perform a variety of tests, including stress testing, security testing, and performance evaluations [17]

In the specific case of a server security system, the testing phase focuses on evaluating the system's response to threats. For example, root access might be protected using the port knocking method, which helps prevent unauthorized access to sensitive system areas. Additionally, the system's defense against Denial of Service (DoS) attacks would be tested using methods like RST Cookies [18]. This phase ensures that the server is secure and capable of withstanding real-world threats. Any issues discovered during testing are addressed and resolved to ensure the system operates smoothly once deployed in a live environment [19].

The method used is an experimental approach or direct trials and direct improvements to see the performance of the server security system for root access using the port knocking method and the Denial of Service (DoS) attack security system using the RST Cookies method on a server computer with the Linux UbuntuOS version 18 operating system.

3.1. Port Knock Method

The Port Knocking method is used to secure SSH port access. Moreover, the foundational principles of Port Knocking are well articulated in the work of , who describe it as a method for communicating through closed ports, thereby dynamically altering firewall rules to permit access only to authorized users [19]. In addition to its application in SSH security, the Port Knocking method has been shown to be versatile across various platforms and services. For example, discuss how Port Knocking can effectively block unwanted access by keeping all ports closed until the correct sequence is executed, thus providing a robust security framework [20]. With this technique, port 22 which is used to access the root server will be closed and protected using a firewall first. During the port knocking authentication process, all involved ports remain closed, and the server does not send any acknowledgment packets to the client. This "silent authentication" method ensures that the port remains protected until the correct sequence is executed, thereby preventing unauthorized access [21]. Port 22 can be opened only by using a port knocking pattern on several ports first. If the pattern used is correct, the firewall will open access to that port, but if the pattern used is wrong, the firewall will block the access attempt. If we configure port Knocking access to port 2, this port will open when we make a request to port 1000

200 3000 in that order, once we complete the sequence correctly the firewall will open. With this we add another level of security for several types of connections to our server. To gain access to port 22, the client/sysadmin can type the port using Nmap, Telnet, or a tool for this purpose.

```
[options]
        logfile = /var/log/knockd.log

[openSSH]
        sequence = 5040,6010,6500
        seq_timeout = 30
        tcpflags = syn
        Start_command = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 22 -j ACCEPT

[closeSSH]
        sequence = 4040,5050,8080
        seq_timeout = 30
        command = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags = syn
```

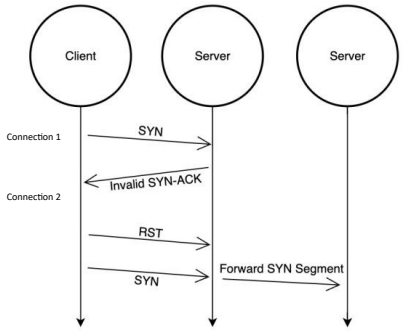
Gambar 2. Konfigurasi aturan port knocking

Figure 2. is the rule configuration for the port knocking method. In this configuration, rules have been added for which ports must be accessed first before accessing SSH port 22. The pattern used to open access is port 5040, 6010, 6500 and the pattern used to close access is port 4040, 5050, 8080. .

3.2. RST Cookies Method

The RST Cookies security method is an essential tool in mitigating Denial of Service (DoS) attacks, particularly when dealing with large-scale, resource-draining threats. Although the primary focus may be on more destructive variants such as permanent DoS (PDoS) attacks, the insights into the mechanics of general DoS attacks demonstrate the critical need for robust security implementations like RST Cookies [22]. This method plays a key role in protecting servers from being overwhelmed by malicious traffic that seeks to render a system inoperable by exhausting its resources.

The RST Cookies method works by utilizing a clever handshake mechanism that helps to differentiate between legitimate and illegitimate connection requests. When a client attempts to establish a connection, the server deliberately responds with an invalid SYN-ACK packet [23]. This invalid response is designed to confuse any illegitimate or malicious client, causing it to send an RST (Reset) packet back to the server, indicating that the connection attempt has failed. If the connection is legitimate, the server will recognize the valid RST packet and proceed to log the client, allowing the connection to be established. By using this method, the server can effectively filter out malicious or fake connection attempts that are typically associated with DoS attacks. The working process for this security method is depicted in the accompanying chart.



Saving a log of remote server ssh port access attempts looks as follows.

Figure 4. Log file for remote server access using

the port knocking method

Figure 3. RST Cookies workflow

Figure 3 shows how the RST Cookies system works in securing the server. In the illustration, it can be seen that the server deliberately sends an invalid SYN-ACK in response to the primary connection from a particular client. This will cause the client to generate an RST parcel, signaling that something went wrong with the connection request. If this connection is accepted, the server acknowledges that the request is genuine then logs the client, and accepts the connection association with that client.

3. RESULTS AND DISCUSSION

3.1 Try Server Root Access

In the experiment, root server access on the SSH port was carried out by calling the port array specified in the security rules that were created. The response from the server. Figure 4 shows the access attempt log stored on the server. The log file shows how the server responds to open SSH port access to the server when remote access is attempted. By implementing this port knocking method, it can be a security system to make the server keep SSH port 22 closed so that it cannot be accessed carelessly by other parties. The server will only open access to the SSH port if the port knocking process in a predetermined order is carried out correctly [24].

3.2 Denial of Service (DoS) Attack Experiment In the Denial of Service (DoS) attack experiment, the following experimental parameters were used.

Table 1. DoS attack parameters

Parameter percobaan DoS	
Port	80
Protocol	TCP
Jumlah Thread	1.000.000

Table 1 provides a detailed overview of the parameters used in simulating a Denial of Service (DoS) attack on a server that employs the RST Cookie method as its security measure. The simulation targets the TCP protocol, specifically focusing on the http port, which is commonly used for web traffic. The attack simulation involves an attempt to overwhelm the server by sending an enormous number of threads, amounting to 1 million threads, in an effort to disrupt normal server operations. This large-scale simulation is designed to test the robustness and effectiveness of the RST Cookie method in defending against such an attack.

The experimental results from the simulation are presented in a comprehensive manner, demonstrating the server's response under the stress of the simulated DoS attack. Key performance metrics, such as the server's ability to maintain availability, response times, and overall stability, are evaluated. These results highlight the effectiveness of the RST Cookie method in mitigating the impact of a DoS attack, providing valuable insights into

ACSSE
APPLIED COMPUTER SCIENCE AND SOFTWARE ENGINEERING
E-ISSN : 0000000000

the security performance of the system when subjected to extreme conditions.

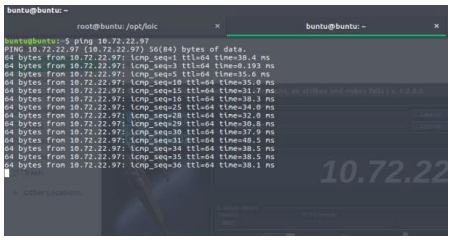


Figure 5. Server response when an attack is carried out before security

Figure 5 illustrates the server's response behavior when it is subjected to a Denial of Service (DoS) attack while being accessed. Under these conditions, the server faces a high volume of malicious traffic intended to overwhelm its resources and disrupt normal operation. Despite the attack, the server manages to provide an average response time of over 30 milliseconds (ms), which indicates that the server remains functional, albeit with a slight delay in processing requests due to the high traffic load.

After implementing the RST Cookies security method, the server demonstrates an improved and more stable response when accessed. The RST Cookies method helps mitigate the effects of the DoS attack by filtering out illegitimate connection attempts and only accepting genuine client requests. This security mechanism ensures that the server can continue to serve legitimate users even during an ongoing attack, maintaining a reasonable response time and preventing the server from becoming fully incapacitated. The overall effectiveness of the RST Cookies method in maintaining server availability and performance is evident in these results.

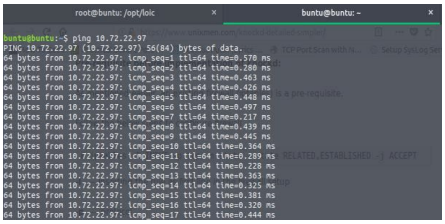


Figure 6. Server response when an attack is carried out after security

Figure 6 illustrates the server's response when it is accessed under the condition of a Denial of Service (DoS) attack, showcasing the effectiveness of the RST Cookies security method. In this scenario, the server is able to maintain a fast and stable response time, consistently delivering responses in under 1 millisecond (ms), even while under attack. This demonstrates the efficiency of the RST Cookies method in preventing the DoS attack from overwhelming the server, allowing it to handle legitimate requests without a noticeable performance decline.

The comparison between the server's response times with and without the RST Cookies

security method highlights a significant difference in performance. Without any security measures, the server's response time during the DoS attack rises above 30 ms, indicating that it is struggling to cope with the malicious traffic. However, with the RST Cookies method in place, the server is able to preserve its resources, reject illegitimate requests, and provide a rapid response to genuine users. This stark contrast underscores the importance of implementing security mechanisms to safeguard the server against DoS attacks and ensure continued, efficient service delivery [25].

4. CONCLUSION AND RECOMMENDATION

Based on the results of the design and implementation of the security system on a server running Linux UbuntuOS version 18, the application of the port knocking method to secure port 22 for Secure Shell (SSH) access proved highly effective. Initially, the server keeps port 22 closed, preventing unauthorized remote access. The server only opens port 22 if the user attempting to access it first interacts with a series of predefined ports in a specific sequence, as dictated by the port knocking configuration. This method adds an additional layer of security by ensuring that only those with knowledge of the correct port sequence can gain access, thus safeguarding the SSH service from unauthorized entry.

In addition to port knocking, the implementation of the RST Cookies method further enhanced the server's security, particularly in defending against large-scale Denial of Service (DoS) attacks. This method ensures that the server remains stable and fully operational even when subjected to substantial attack efforts aimed at exhausting its resources. By utilizing RST Cookies, the server can efficiently manage and mitigate such attacks, maintaining accessibility and preventing downtime. This dual approach—securing SSH access with port knocking and defending against DoS attacks with RST Cookies—has proven to be a robust and reliable solution for maintaining the server's security and stability.

REFERENCES

Commented [at2]: minimum 25 references

[1] G. J. Simson and G. Spafford, *Practical UNIX & Internet Security*. O'Reilly & Associates, Inc, 1996.

[2] W. Stallings, *Network and Internetwork Security*. Prentice Hall, 1995.

[3] R. Ahmad, B. A. Candra, and A. M. Nur, "Use of Backpropagation Methods in Artificial Neural Networks for Intrusion Detection Systems," *Infotek J. Informatics Technol.*, vol. 3, no. 2, 2020.

[4] Suhartini and et al., "Implementation of a Client Server-Based Online Examination Application Case Study at SMA Negeri 3 Selong," *Infotek J. Informatics Technol.*, vol. 5, no. 1, 2022.

[5] B. Rahardjo, *Information Systems Security: Security on the Internet*. National Infrastructure Information Seminar, ITB, 1997.

[6] I. Riadi, R. Umar, I. Busthomi, and A. W. Muhammad, "Block-Hash of Blockchain Framework Against Man-in-the-Middle Attacks," *Regist. J. Ilm. Teknol. Sist. Inf.*, vol. 8, no. 1, p. 1, 2021, doi: 10.26594/register.v8i1.2190.

[7] E. Reed, J. D. Harwood, J. L. A. Jackson, and T. Gebreyesus, "'Leveling Up' for Change: Application of a Multilevel System Capacity Change Model to Evaluate U.S. Department of Defense Global Health Engagement Activities," *New Dir. Eval.*, vol. 2021, no. 170, pp. 39–50, 2021, doi: 10.1002/ev.20458.

[8] J. A. Wilson and G. Morrisroe, "Systems Analysis and Design," pp. 241–279, 2005, doi: 10.1201/9781420055948.pt2.

[9] S. Ramakrishnan, "System Analysis and Design," *J. Inf. Technol. Softw. Eng.*, vol. 02, no. 05, 2012, doi: 10.4172/2165-7866.s8-e001.

[10] Z. Hu, "Construction and Application of Product Optimisation Design Model Driven by User Requirements," *Sci. Rep.*, vol. 14, no. 1, 2024, doi: 10.1038/s41598-024-67406-x.

[11] A. J. Lattanze, "Architecting Software Intensive Systems," 2008, doi: 10.1201/9781420045703.

[12] T. N. Nguyen, E. V Munson, J. Boyland, and C. Thao, "Architectural Software Configuration Management in Molhado," 2010, doi: 10.1109/icsm.2004.1357815.

[13] C. Haas, S. Münz, J. Wilke, and A. Hergenröder, "Evaluating Energy-Efficiency of Hardware-Based Security Mechanisms," 2013, doi: 10.1109/percomw.2013.6529559.

[14] P. Daca, T. A. Henzinger, W. Krenn, and D. Ničković, "Compositional Specifications for Ioco Testing," 2014, doi: 10.1109/icst.2014.50.

[15] C. Mawardi, D. Kuswoyo, and N. Falah, "Implementation of a Cyberpanel-Based Partial Cloud Server as a Prevention of Security Information Management System (SIMS) Encryption," 2022, doi: 10.4108/eai.16-11-2022.2326064.

[16] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner, "Model-Based Security Testing: A Taxonomy and Systematic Classification," *Softw. Test. Verif. Reliab.*, vol. 26, no. 2, pp. 119–148, 2015, doi: 10.1002/stvr.1580.

[17] T. Vieira and C. Serrão, "Web Security in the Finance Sector," 2016, doi: 10.1109/icitst.2016.7856707.

[18] Mursyidah *et al.*, "Analysis and Implementation of the Port Knocking Method Using Firewall-Based Mikrotik RouterOS," *Iop Conf. Ser. Mater. Sci. Eng.*, vol. 536, no. 1, p. 12129, 2019, doi: 10.1088/1757-899x/536/1/012129.

[19] W. R. Koch and A. Bestavros, "PROVIDE: Hiding From Automated Network Scans With Proofs of Identity," 2016, doi: 10.1109/hotweb.2016.20.

[20] M. Idhom, H. E. Wahanani, and A. Fauzi, "Network Security Applications Using the Port Knocking Method," *J. Phys. Conf. Ser.*, vol. 1569, no. 2, p. 22046, 2020, doi: 10.1088/1742-6596/1569/2/022046.

[21] M. Shiraz, L. Boroumand, A. Gani, and S. Khan, "An Improved Port Knocking Authentication Framework for Mobile Cloud Computing," *Malaysian J. Comput. Sci.*, vol. 32, no. 4, pp. 269–283, 2019, doi: 10.22452/mjcs.vol32no4.2.

[22] S. Abaimov, "Understanding and Classifying Permanent Denial-of-Service Attacks," *J. Cybersecurity Priv.*, vol. 4, no. 2, pp. 324–339, 2024, doi: 10.3390/jcp4020016.

[23] M. A. J. Hidayat and H. M. Putra, "Sistem Keamanan Server Linux CentOS Dengan Metode Port Knock Dan RST Cookies," *Infotek J. Inform. Dan Teknol.*, vol. 6, no. 2, pp. 411–420, 2023, doi: 10.29408/jit.v6i2.17500.

[24] H. Husain, "Implementation of Port Knocking With Telegram Notifications to Protect Against Scanner Vulnerabilities," *Matrik J. Manaj. Tek. Inform. Dan Rekayasa Komput.*, vol. 23, no. 1, pp. 215–228, 2023, doi: 10.30812/matrik.v23i1.3459.

[25] M. Durairaj and A. Persia, "ANM to Perceive and Thwart Denial of Service Attack in WLAN," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 6, pp. 59–66, 2015, doi: 10.5815/ijcnis.2015.06.07.