A STUDY PROPOSING IMPROVEMENTS TO OVERLAY NETWORK ARCHITECTURE TO MITIGATE THE IMPACT OF DISTRIBUTED DENIAL-OF-SERVICE ATTACKS ON WEBSITE SYSTEMS

Vu Thi Phuong*, Nguyen Van Vu, Dang Thi Hien
Nam Dinh University of Technology Education, Nam Dinh, Vietnam

ARTICLE INFORMATION

Journal: *Vinh University Journal of Science Natural Science, Engineering and Technology*
p-ISSN*: 3030-4563*
e-ISSN: *3030-4180*

Volume: 53
Issue: 4A
****Correspondence:***
vtphuongspktnd@gmail.com

Received: *13 August 2024*
Accepted: *24 September 2024*
Published: *20 December 2024*

Citation:
Vu Thi Phuong, Nguyen Van Vu, Dang Thi Hien (2024). A study proposing improvements to overlay network architecture to mitigate the impact of distributed denial-of-service attacks on website systems
Vinh Uni. J. Sci.
Vol. 53 (4A), pp. 15-26
doi: 10.56824/vujs.2024a080a

ABSTRACT

Denial-of-service (DoS) attacks on website systems have become increasingly sophisticated. These attacks are evolving at a pace and complexity that outstrips the capabilities of current defensive measures. This paper presents an effective method to enhance the resilience against distributed denial-of-service (DDoS) attacks by implementing an overlay network architecture to protect website systems. This architecture employs a group of Secure Overlay Access Points (SOAP) to authenticate and distinguish legitimate users from malicious attackers, ensuring that valid requests are routed to hidden nodes within the overlay network via Secure Sockets Layer (SSL) connections. These hidden nodes then forward user requests through a filtering region before delivering them to the target server. The study has introduced enhancements to the existing overlay network architecture to promptly detect scenarios where a network node becomes a source of attack and automatically reroute queries to prevent potential damage. After conducting experimental simulations of attack scenarios, the improved architecture has been tested and yielded auspicious and reliable results compared to the original architecture. This approach contributes to mitigating and reducing the impact of DDoS attacks on modern websites.

Keywords: Cyber attacks; denial of service attacks; distributed denial of service attacks; website; overlay network.

1. Introduction

Distributed denial-of-service (DDoS) attacks have seriously threatened contemporary website systems. By overloading servers with a large volume of requests from multiple sources, DDoS attacks can disrupt or completely halt website operations. This affects user experience, erodes customer trust, and results in significant economic damage for businesses. DDoS attacks are becoming increasingly sophisticated, with attackers leveraging botnets to execute large-scale, hard-to-detect attacks. These attacks can cause prolonged disruptions, weaken security systems, and sometimes be part of broader campaigns targeting organizations. Given the increasing

reliance of businesses on online operations, preventing and mitigating the impact of DDoS attacks has become a top priority for safeguarding the stability and security of website systems.

Various methods have been proposed to combat denial-of-service attacks, including packet filtering to prevent source address spoofing, attack redirection, reverse traffic flows to the network, and isolation to differentiate between client and server traffic [1], [2]. While each of these solutions offers valuable techniques to address aspects of denial-of-service attacks, they can only protect individual facets of the problem.

This study presents a comprehensive approach to mitigating distributed denial-of-service (DDoS) attacks based on implementing overlay network architecture to protect targets from attacker infiltration. Building on this architecture, proposals such as SOS and WebSOS have been developed [3], [4], [5], [6], [7], [8]. The SOS architecture employs an overlay network to ensure that only authenticated queries reach the target server using secret nodes. WebSOS extends the SOS architecture with enhanced security mechanisms, including CAPTCHA checks, Proxylet connections, SSL, and X.509 authentication. To address scenarios where nodes in the overlay network are compromised, we propose enhancements to detect and adjust queries to prevent attacks automatically.

2. Overlay network architectures SOS and WebSOS

Both the SOS and WebSOS architectures employ a crucial technique known as structured routing, utilizing distributed hash tables (DHT) with the Chord protocol. The Chord protocol is a distributed lookup protocol designed to perform a single operation: mapping a key to a node within the network. This process involves mapping the key to a node via a consistent hashing function. Specifically, the key is hashed to produce a hash value, which is then mapped to a particular node in the network. This approach simplifies the storage and retrieval of data, as each key is associated with a unit of data, and key/data pairs are stored at the node corresponding to the hashed key. Chord effectively solves various issues, including load balancing, distribution, flexibility, and scalability. Additionally, Chord performs well in dynamic network environments where nodes frequently join or leave the network.

2.1. SOS Architecture

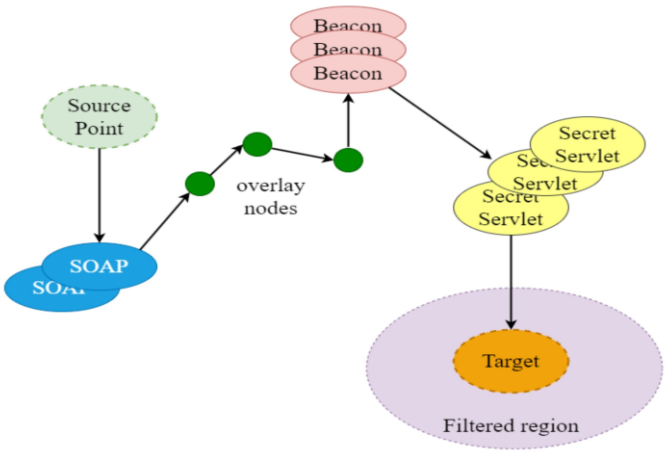

Figure 1: *Basic architecture of SOS*

In this architecture (see Figure 1) [2], client requests from the source point are directed into an overlay layer through a SOAP - Secure Overlay Access Point node. Due to the characteristics of the SOS architecture, this node is responsible for verifying the validity of the user through an authentication mechanism, such as login. Once the user is successfully authenticated, the request is forwarded through the overlay network. This overlay network functions as a distributed firewall, built on the Chord protocol with structured routing techniques and utilizing distributed hash tables (DHT).

2.2. WebSOS Architecture

Regarding the overlay network structure, WebSOS inherits from the SOS model, as illustrated in Figure 1. WebSOS builds upon the principles of SOS [3] to develop an architecture designed to mitigate denial-of-service attacks, ensuring connectivity to the target server even under attack conditions. WebSOS enhances the SOS concept by implementing a CAPTCHA system to differentiate legitimate users from automated bots (Autobots). User requests are transmitted through the overlay network via a web proxy, and clients are authenticated using SSL/TLS protocols without requiring changes to the existing network infrastructure.

2.3. Application of SOS and WebSOS architectures

The paper's main idea is to apply the SOS architecture to build an overlay layer around the target server to prevent attackers from accessing and attacking the server while allowing only authenticated users to connect to it. The paper combines the use of WebSOS to protect websites against denial-of-service attacks. This architecture is implemented through crucial activities such as authenticating legitimate users via a Graphic Turing Test, establishing an SSL connection through a proxy applet over the overlay network to a Servlet, and forwarding the request from the Servlet through a filtering zone to the target server.

3. Improvement results and discussion

The experiment was conducted to establish a website protected by the WebSOS overlay network. The objective of the experiment was to implement the WebSOS solution as described and simultaneously evaluate the latency of client requests when using the WebSOS overlay network compared to direct connections to the target server [4], [5], [6], [7], [8], [9], [10], [11].

3.1. Experimental environment

The WebSOS overlay network architecture is implemented on a network of virtual machines running CentOS 5, with 3.0 GHz processors and 1GB of RAM. The system comprises three main modules:

- CAPTCHA Module: Installed on the Xampp WebServer.
- Secure Tunnel Proxylet Module: Developed in Java.
- Communication Control Module and Overlay Network Module (Chord): Developed in Java and C, respectively.

The website to be protected is hosted on a machine with the Xampp WebServer installed.

3.2. WebSOS architecture setup

Compared to the proposed WebSOS, the experimental architecture incorporates several distinct mechanisms. Servlets are manually configured via command-line mode rather than receiving notifications from the server. To join the WebSOS overlay network, each machine executes commands within the Communication Control Module and the Overlay Network Module. When a node joins the overlay network and acts as a Servlet, it declares a file containing the IP addresses of the target servers it serves. If a node declares the corresponding file empty, it functions as a SOAP, Beacon, or a standard overlay node. For nodes serving as SOAP, two additional modules are installed: the CAPTCHA module for user authentication and the Secure Tunnel Proxylet module to allow users to download and run the proxy applet in their browsers. Machines functioning as target servers only need XAMPP installed and upload some HTML files to create a test website accessible through the overlay network.

3.3. Proposed improvements

3.3.1. Issues with WebSOS overlay network

To study and enhance the WebSOS architecture, we assume a scenario where an attacker compromises one or more nodes in the WebSOS overlay network. From these compromised nodes, the attacker could perform one of the following three types of attacks:

Data Integrity Attack: A data integrity attack can occur on the request channel by dropping packets or disrupting the established communication channel. When a compromised node drops packets on the request channel, users may experience difficulties connecting to the server. Suppose the compromised node attacks data integrity on the established communication channel. In that case, such attacks can be detected through enhanced solutions or user-side applications, such as detecting discrepancies in the returned data or through authentication mechanisms. Upon detecting such attacks, the system can redirect requests to another SOAP node to maintain service continuity and integrity.

Packet Dropping Attack: A packet-dropping attack aims to obstruct the connection setup, preventing users from connecting to the server through the compromised node. Upon closer analysis, it is evident that the attacker can drop packets transmitted between legitimate users and the server within an established communication channel. Like data integrity attacks, this attack can be detected through enhanced solutions or user-side applications, such as noticing disrupted connections or low application throughput.

Packet Flooding Attack: A compromised node can conduct a flooding attack against the target server by sending a high volume of packets to the Servlet, overwhelming it with excessive traffic.

3.3.2. Proposed improvements

We will focus on two types of attacks, packet-dropping attacks, and propose a solution by establishing a mechanism to detect these attacks. Subsequently, the user's proxylet will be configured to switch SOAP, enabling connection to the server via an alternative routing path unaffected by the compromised node.

The attack detection mechanism periodically sends a probe packet to the target server. In the WebSOS architecture, the user's proxylet sends periodic probe packets to the

server. Suppose the server responds to the probe packet in a manner that does not conform to predefined criteria. In that case, we can conclude that a node along the routing path has been compromised and is conducting a packet-dropping attack. The proxylet automatically switches SOAP to connect to the server through an alternative routing path. This mechanism operates transparently to the user without requiring the user to authenticate through the new SOAP.

Additionally, attackers may selectively block legitimate requests while allowing probe packets and probe responses to pass through the compromised node. To address this, the mechanism will automatically switch to another proxy if a certain number of requests do not receive responses from the server. This ensures that users can establish a regular connection without routing through the compromised node.

The proposed improvement can be illustrated in pseudocode as follows:

- Proposed improvements implemented at the proxy applet:

Set the variable for unanswered probes: probe = 0;
Set the variable for failed connections: numD = 0;
Set the variable for successful connections: numS = 0;
Set the variable for connection failure check: drop = false;
(*) If probe > 3, change SOAP for the client and reset the variables to default values.
If numD >= 3, change SOAP for the client and reset the variables to default values.
Send request data.
Send probeRequest after a random delay and increment probe by 1.
Check if drop == true, increment numD by 1.
Set drop = true.
If the number of successful connections numS > 7, set numS = 0 and numD = 0.
If a response is received, increment numS by 1 and reset drop = false.
If the response data is probeResponse, set probe = 0 and numD = 0.
Repeat (*)

- Proposed improvements implemented at the destination server:

If a request is received as probeRequest, process it and send back probeResponse.

According to the pseudocode, the proxylet running on the client monitors the process of sending requests to the server. Suppose three out of 10 requests fail to receive a response. In that case, the proxylet will interpret this as an indication of a packet drop attack. It will automatically change the Secure Overlay Access Point (SOAP) to connect to the server. Additionally, after a random interval, the proxylet on the client will send a probeRequest packet to the destination server. If the client does not receive a valid probeResponse, the value of numD will be incremented. When numD reaches 3, the proxylet will change the SOAP for the client and reset numD to 0, continuing the monitoring process. Conversely, if a valid probeResponse is received, the proxylet will register that no packet drop attack is occurring, reset the variables, and restart the process from the beginning.

3.3.3. Experimental implementation

To implement this proposal, we modified the proxylet's operational mechanism to send probe packets to the server periodically and await responses. If the responses are

inaccurate or absent, a pre-defined variable will increment until it reaches a predetermined threshold. When this variable reaches the threshold, the proxylet will automatically switch to a different Secure Overlay Access Point (SOAP) to ensure continued access to the service for users. Furthermore, if user requests do not receive a response from the server, this variable will also increment towards the threshold. When the proxylet switches to another SOAP, the variable will be reset to 0. Upon experimentation with the system, we observed that this mechanism performs effectively. Instances of no response from probe packets or the cancellation of server responses, even when probe and probe response packets are maintained while other packets are dropped, were successfully detected and managed by changing the SOAP. A list of available SOAPs is maintained at each SOAP, and the proxylet reads and stores this list in an array. Upon detecting a packet drop attack, the proxylet utilizes this array to switch to an alternative SOAP.

a. Test scenarios

To implement the proposal and validate the results of the proposed mechanism, we first developed two experimental scenarios as follows:

- ***Scenario 1:*** Assume a client connects to a SOAP. After completing user authentication through a CAPTCHA test, the user downloads and runs a proxy applet to connect to the SOAP. The SOAP will forward the user's request through the WebSOS overlay network to the Servlet and from the Servlet to the destination server. During the transmission from SOAP to Servlet, it is possible that a node in the overlay network has been compromised and is conducting a packet-dropping attack. This node will continue to forward the user's requests to the destination server and listen for responses from the server. However, it will stop writing data to the client in the outbound stream, leading to the dropping of all packets from the server to the user. For legitimate users, the delay in receiving packets may result in prolonged web page loading, loss of connection, or low application throughput. Based on these symptoms, packet dropping can be detected by sending and receiving probeRequest and probeResponse packets, and corrective measures can be taken. This measure involves configuring the proxy applet on the client machine to switch to another SOAP automatically. We executed this scenario with both the original program and the improved program to assess the impact of packet-dropping attacks on the original program and evaluate the improved mechanism's ability to detect and address packet-dropping attacks from compromised nodes effectively.

- ***Scenario 2:*** Similar to the first scenario, however, in the second scenario, the attack node does not drop all packets. We assume that the attacker is sophisticated enough to detect and handle probeRequest and probeResponse packets, regardless of how well they are concealed within the outbound data stream. Alternatively, the attacker may drop many packets from the server, allowing only a few packets to pass through to deceive users into believing there is still a connection to the server, albeit at a plodding speed. In this case, probeRequest and probeResponse packets might not be dropped, or the dropping of three consecutive probeResponse packets might be avoided. Consequently, in the second scenario, the proposed mechanism that relies on probeRequest and probeResponse packets for attack detection becomes ineffective. In this situation, an alternative method must be employed to detect large-scale packet dropping, identify the presence of a malicious node and perform SOAP switching for the client.

b. Experimental results

-For the original program: When executing the test scenario with the original WebSOS program, the observed phenomenon is that the request packets are generally sent on the client side. However, no response packets are received, leading the browser to continue waiting for the response packets. The webpage displays the message "*Waiting for http://...."* but fails to load the result page. After 20 seconds (as set by the ***setSoTimeout*** configuration in the program code) without receiving a response, the browser displays the error message "*Internet Explorer cannot display the webpage*."

Figure 2: *Experimental scenario executed with the original program*

The inevitable result when executing the scenario with the original WebSOS program is that the browser on the client side does not receive a response from the destination server (see Figure 2). This indicates that legitimate users cannot connect to the destination server when a node in the routing path from the client to the destination server is compromised and conducts a packet-dropping attack. In the second scenario, due to the majority of packets being dropped, users are also nearly unable to connect to the server.

- With the improved program:

+ Scenario 1: When executing the first test scenario with the improved program, the initial behavior was similar to the original program, with the browser not receiving a response from the server and displaying the message "Waiting for http://...." with no further progress. However, after approximately 14 seconds, the website began to load normally, and users could successfully access the site. Subsequent queries usually proceeded without the extended page loading time experienced initially.

This improvement is attributed to the proposed mechanism, where each time a request is sent to the server, a probeRequest packet is periodically sent every 3 seconds (the interval between probeRequest transmissions can be adjusted). If three consecutive probeRequests do not receive a response, the proxy applet automatically switches to a different SOAP, altering the routing path from the client to the destination server. The attack's impact is mitigated once the path no longer passes through the compromised node. Concurrently, the proxy applet will resend the request via the new SOAP and receive a response, allowing the webpage to load successfully after the probeResponse timeout without a reply. With the new routing path, subsequent queries usually proceed, eliminating the issue of no response from the browser.

It is important to note that this scenario represents the worst case: a new user encounters a malicious node in their routing path. In better scenarios, where a user accessing the website encounters a compromised node in the routing path, probeRequests and probeResponses are still sent but may be dropped. After approximately 12 seconds (the fourth probeRequest transmission), the proxy applet will detect the malicious node and change the SOAP. During this period, if the user does not navigate to another page on the website (e.g., is reading an article), they may not perceive the SOAP change. When the user navigates to a different page, there is no significant delay, and they do not experience any impact from the SOAP change.

+Scenario 2: In the second test scenario with the improved program, users encountered issues connecting to the website or experienced a significant loss of content due to many packets being dropped. In this scenario, it is assumed that the probeRequest and probeResponse mechanism was ineffective because the attacking node did not drop three consecutive probe packets or the attacker was sophisticated enough to allow probe packets to pass through despite attempts to conceal them. To address this issue, a solution was proposed to count the number of request packets that do not receive a response. According to this solution, if up to 3 out of 10 request packets do not receive a response (this ratio can be adjusted according to the network), the proxy applet will recognize this as a sign of a packet-dropping attack. It will switch to a different SOAP for the client.

In this scenario, users may need help because the request counting mechanism could require them to make up to three requests without successfully loading the webpage. However, from the fourth request onward, users can generally access the website as the proxy applet has switched the SOAP, thereby avoiding the packet-dropping attack from the malicious node.

In better cases, where the user's initial request to the website receives a response and subsequent webpage components are automatically fetched by the browser, the attack would only cause the user to perceive that the website is missing many components. In such cases, the proxy applet will detect the malicious node and switch SOAP, allowing the user to access the website usually and have a satisfactory experience when accessing subsequent pages.

3.3.4. Performance evaluation of the improved program

To evaluate the performance of the improved program compared to the original program, we compared the access times of both programs to various addresses. Specifically, by measuring the average access times to several websites, we obtained the results presented in Table 1.

Table 1: *Performance comparison of the improved program versus the original program*

Address	Direct access	Original version	Improved version
http://ddoswebsite/main/index.htm	0.52	1.46	1.31
http:// ddoswebsite /main/test.htm	0.93	2.99	3.25
http://www.google.com	1.61	2.52	2.41

The comparison chart of query times, as shown in Figure 3, illustrates the performance of both programs running with a covering network architecture consisting of 3 nodes.

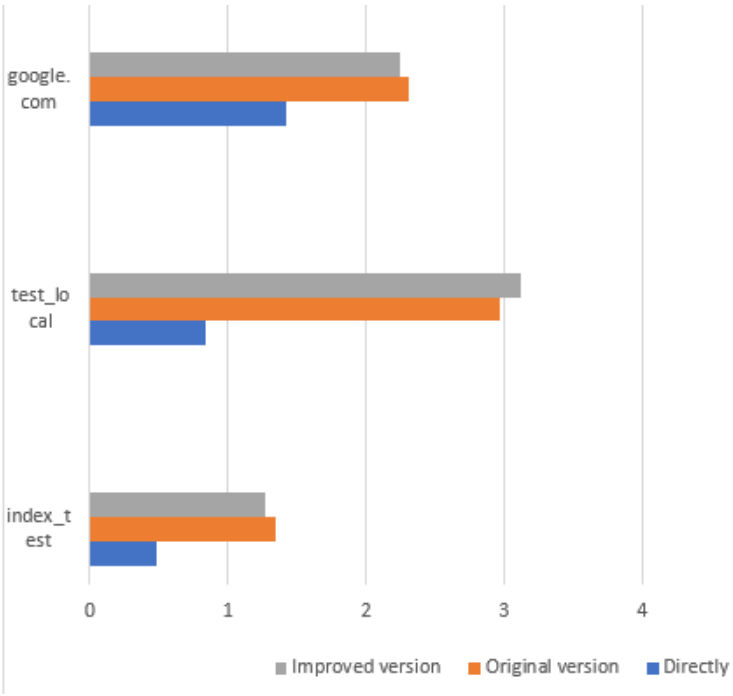

Figure 3: *Average query time for various programs across several websites*

To evaluate the performance of the improved program compared to the original program, we compared the access times under attack scenarios described in scenarios 1 and 2. The results are presented in Table 2.

Table 2: *Comparison of access time under attack for scenarios 1 and 2*

Address	Direct access	Original version (Both scenarios)	Improved version (Scenario 1) first access	Improved version (Scenario 2) from the fourth attempt
index.htm_local	0.51	No Connection	14.72	2.55
test.htm_local	0.92	No Connection	15.33	3.79
www.google.com	1.58	No Connection	16.27	3.43

The chart comparing the average query times of the programs for various websites under scenarios 1 and 2 is shown in Figure 4. For the original version, the result was always a failure to connect. The overlay network architecture consists of 3 nodes. In scenario 1, the detection mechanism to switch SOAP took 12 seconds. In scenario 2, queries succeeded only after the fourth attempt.

The measurements demonstrate the limitations of the original architecture, as 100% of the tests failed to connect when a node in the overlay network was compromised and attacked. The improved mechanism shows promising results, indicating that it is a viable solution for deployment.

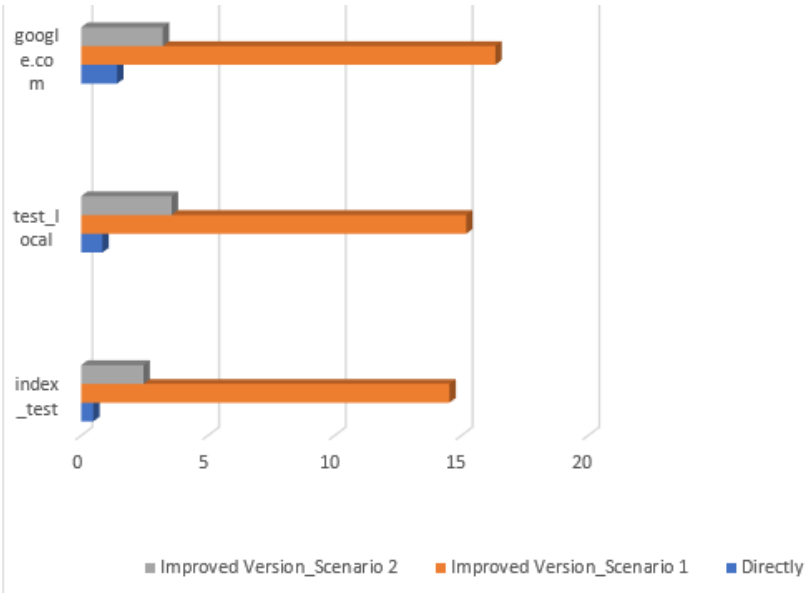

Figure 4: *Comparison of average query time for programs accessing various websites under scenarios 1 and 2*

4. Conclusion

The research and experimental evaluation yielded the following key findings:

- **Development of the WebSOS Architecture:** The WebSOS architecture was developed with strong capabilities to defend against denial-of-service (DoS) attacks. The system allows direct user access to the website, activating its defense mechanisms only when a DoS attack is detected. These mechanisms include robust IP filtering, secure user authentication, and the concealment of sensitive servlets to safeguard the target server.

- **Performance Evaluation and Scalability:** Experiments indicated that the system's latency is within acceptable limits, with a high potential for deployment and scalability in public web servers catering to large user bases.

- **Enhancements and Effectiveness:** The system has been improved to handle scenarios where one or more nodes in the overlay network are compromised, executing packet-dropping or packet integrity attacks. These enhancements have proven effective, operating transparently to users and maintaining a seamless experience even during an attack

Disadvantages and Limitations:

- **Latency:** The system experiences high latency due to user requests needing to pass through multiple intermediary stages within the overlay network.

- **Compromised Nodes:** The system does not fully address cases where a node in the overlay network is compromised and becomes an attacker's agent.

- **Direct Attacks:** Attackers may bypass the overlay network and directly target the server through the filtering zone, potentially overwhelming it with packet floods and rendering it ineffective.

These findings provide valuable insights into the WebSOS architecture's capacity to combat DoS attacks, while also highlighting areas for further enhancement to improve system efficiency and security.

REFERENCES

[1] D. Rubenstein, "SOS: An architecture for mitigating DDoS attacks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 176-188, 2014. DOI: 10.1109/JSAC.2003.818807

[2] A. Stavrou, "WebSOS: An overlay-based system for protecting web servers from denial of service attacks," *Computer Networks*, vol. 48, no. 5, pp. 781-807, 2015. DOI: 10.1016/j.comnet.2005.01.005

[3] S. S. Chowriwar and M. N. Sambhe, "Mitigating denial-of-service attacks using secure," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 8, no. 9, pp. 479-483, 2014. DOI: 10.14445/22315381/IJETT-V8P284

[4] E. Cambiaso, G. Papaleo, G. Chiola, and M. Aiello, "Slow DoS attacks: Definition and categorisation," *International Journal of Trust Management in Computing and Communications*, vol. 1, no. 3, pp. 300-319, 2013. DOI: 10.1504/IJTMCC.2013.056440

[5] T. Hirakawa, K. Ogura, B. B. Bista, and T. Takata, "A defense method against distributed slow HTTP DoS attack," In *Proc. Int. Conf. on Network-Based Information Systems (NBiS)*, pp. 152-158, 2016. DOI: 10.1109/NBiS.2016.58

[6] N. Tripathi, N. Hubballi, and Y. Singh, "How secure are web servers? An empirical study of slow HTTP DoS attacks and detection," In *Proc. Int. Conf. on Availability, Reliability and Security (ARES)*, pp. 454-463, 2016. DOI: 10.1109/ARES.2016.20

[7] S. Tayama and H. Tanaka, "Analysis of slow read DoS attack and communication environment," In *Proc. Int. Conf. on Information and Communication Technologies (ICMWT)*, pp. 350-359, 2017. DOI: 10.1007/978-981-10-5281-1_38

[8] S. Naseer and Y. Saleem, "Enhanced network intrusion detection using deep convolutional neural networks," *KSII Transactions on Internet and Information Systems*, vol. 12, no. 10, pp. 5159-5178, 2018. DOI: 10.3837/tiis.2018.10.028

[9] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," In *Proc. IEEE Int. Conf. on Big Data and Smart Computing (BigComp)*, pp. 313-316, 2017. DOI: 10.1109/BIGCOMP.2017.7881684

[10] E. Alhajjar, "Adversarial machine learning in network intrusion detection systems," *Expert Systems with Applications*, vol. 186, pp. 1-10, 2021. DOI: 10.1016/j.eswa.2021.115782

[11] K. Salah, "Enhanced EdoS-Shield for mitigating EdoS attacks originating from spoofed IP address," In *Proc. 11th Int. Conf. on Trust, Security and Privacy in Computing and Communications*, IEEE, pp. 1167-1174, 2012. DOI: 10.1109/TrustCom.2012.146

TÓM TẮT

NGHIÊN CỨU ĐỀ XUẤT CẢI TIẾN KIẾN TRÚC MẠNG BAO PHỦ ĐỂ GIẢM TÁC HẠI DO TẤN CÔNG TỪ CHỐI DỊCH VỤ PHÂN TÁN ĐỐI VỚI HỆ THỐNG WEBSITE

Vũ Thị Phương, Nguyễn Văn Vũ, Đặng Thị Hiền
Trường Đại học Sư phạm Kỹ thuật Nam Định, Nam Định, Việt Nam
Ngày nhận bài 13/08/2024, ngày nhận đăng 24/9/2024

Các cuộc tấn công từ chối dịch vụ (DoS) vào hệ thống website hiện nay đang phát triển với tính chất ngày càng tinh vi. Những cuộc tấn công này diễn ra nhanh chóng và phức tạp hơn nhiều so với các phương pháp phòng thủ hiện có. Bài báo này trình bày một phương pháp hiệu quả nhằm nâng cao khả năng phòng chống tấn công từ chối dịch vụ phân tán (DDoS) bằng cách áp dụng kiến trúc mạng bao phủ để bảo vệ hệ thống website. Trong kiến trúc này, một nhóm các điểm truy cập bảo mật (Secure Overlay Access Points, SOAP) được sử dụng để xác thực và nhận diện người dùng hợp pháp, phân biệt họ với các phần mềm độc hại của kẻ tấn công. Các yêu cầu hợp lệ sẽ được gửi đến các nút bí mật trong mạng bao phủ thông qua kết nối SSL (Secure Sockets Layer). Những nút bí mật này sau đó sẽ chuyển tiếp yêu cầu của người dùng qua một lớp lọc trước khi truyền đến máy chủ đích. Nghiên cứu đã tiến hành cải tiến kiến trúc mạng bao phủ hiện có để phát hiện kịp thời khi các nút trong mạng trở thành nguồn tấn công, đồng thời tự động chuyển hướng truy vấn cần thiết nhằm tránh các thiệt hại không mong muốn. Sau khi mô phỏng các kịch bản tấn công, kiến trúc cải tiến đã được kiểm tra và cho thấy kết quả rất khả quan, đáng tin cậy hơn so với kiến trúc ban đầu. Điều này sẽ giúp giảm thiểu các tác hại của các cuộc tấn công từ chối dịch vụ phân tán đối với các hệ thống website hiện nay.

Từ khóa: Tấn công mạng; tấn công từ chối dịch vụ; tấn công từ chối dịch vụ phân tán; website; mạng bao phủ.