Design and Implementation of a Reverse Proxy Architecture for Cyber Attack Mitigation on Private Cloud: A Case Study

Marco Ariano Kristyanto
Department of Informatics
University of Surabaya
Surabaya, Indonesia
marcokristyanto@staff.ubaya.ac.id

***Abstract*—Web applications are vital to the operation of institutions and organizations. Numerous firms employ a singular monolithic server architecture to host their essential web-based applications. This technique may lower management costs, but it also increases the risk of a single point of failure. This paper presents a multi-VM isolation architecture integrated with a Defense-in-Depth strategy, utilizing open-source tools such as Nginx reverse proxy, Suricata, Fail2ban, and iptables to address this challenge. This stratified security methodology guarantees enhanced access to web services while preserving cost-effectiveness for small to medium-sized enterprises. Validation via penetration testing confirms that the architecture can withstand DoS attacks and effectively operate five separate web applications.**

Keywords—Nginx, Open Source Security, Defense in Depth, Reverse proxy, Private Cloud.

I. Introduction

In today's digital era, web applications play a vital role in supporting the operational activities of organizations and institutions. To simplify management, many organizations consolidate critical web-based applications into a single, monolithic server architecture. While this can help reduce management costs, it can significantly increase security risks. For example, when a server is attacked through DDoS, brute force, or other vulnerabilities, this can lead to a single point of failure that can paralyze all digital services, disrupting the organization's operations. To address these risks and problems, it is necessary to implement a layered security strategy (Defense in Depth) appropriately and measurably.

This study is motivated by a case study of the failure of a monolithic-based server architecture on a production server running several web-based applications simultaneously. The incident highlights the system's vulnerability to cyberattacks, resulting in both operating system and hardware failures, and demonstrates that an architecture lacking a properly configured defense layer is insufficient to counter the growing complexity of cyber threats.

Various approaches include migrating to commercial cloud services, employing paid web application firewall services, or leveraging a Content Delivery Network (CDN). However, this solution also requires significant technical and financial resources. Therefore, it cannot always be applied to organizations or institutions with limited financial budgets that still manage their architecture privately. Small and medium-sized enterprises face challenges in implementing commercial security services, as these solutions often exceed their financial capabilities and require outsourcing critical data flows to third-party suppliers. This raises issues with budgetary constraints and data sovereignty.

Therefore, a study of the implementation of a practical, affordable, replicable, and server-based security solution is needed. This study proposes a new server architecture based on multi-VMs, implementing a security strategy that incorporates built-in firewalls, iptables, Fail2ban, and open-source IDS as an additional layer of defense. By separating it into multiple virtual machines, the server workload can be balanced, and the risk of a single point of failure can be minimized. Furthermore, the mitigation of cybersecurity threats can be enhanced by controlling data traffic on proxies and firewalls.

This study aims to design and evaluate a distributed architecture with a layered defense for private cloud environments. Focusing on the effectiveness of handling cyberattacks and improving service management in multi-VM architectures. The novelty of the research lies in the practical combination of reverse proxy implementation with multi-VM isolation equipped with *open-source security layers*. The results of this study are expected to serve as a design reference for small and medium-sized organizations that require affordable layered defense solutions.

The remainder of the papers are as follows. Section I will provide an overview of the research background. Section II will discuss related works and literature review that were used in this research. Section III will discuss the architecture proposed in this research and explain its components. Section IV will discuss the results and implementation of this architecture, particularly in the context of handling penetration testing. Section V will discuss the conclusion and future works.

II. Related Work

A. Nginx as Reverse Proxy

Nginx is widely used as a reverse proxy, particularly in modern web architectures. For example, it is commonly implemented as a load balancer, and comparisons with other load balancing algorithms have shown that Nginx can be more efficient. [1]. Other research on the use of nginx as a load-balancing tool on cluster servers was also implemented by [2]. Using Nginx as a web server backend combined with a round-robin algorithm, the results showed that the algorithm works well with Nginx as a reverse proxy in the website architecture.

In addition to load balancing, the Nginx reverse proxy is also used to improve security. For example, combining Nginx with the NAXSI web application demonstrated significant performance improvement when handling several penetration testing experiments on the site, and also reduced CPU usage on the web server [3].

Other studies have also described how Nginx can be used to counter DDoS attacks by filtering the data traffic to the server [4].

B. *Firewall Technologies*

Firewalls are an essential component in cyber defense. Firewalls act as gatekeepers for data traffic entering a system or server. They can be configured according to the system owner's policies. [5] And regulate data traffic between the Internet and intranet networks [6].

In addition to static rules, firewalls can also be configured using dynamic rules. As research conducted by [7] Indicates that dynamic firewalls are more robust in protecting and securing networks and servers. In securing computer networks and the internet, firewalls can be combined with other components, such as IDS, for example, in securing big data infrastructures. [8] And with a reverse proxy to secure the web server [9].

Firewalls can also be enhanced with artificial intelligence, specifically through machine learning algorithms to improve their detection capabilities. Studies have shown that a web firewall combined with AI can detect various threats, including XSS attacks and SQL injection attacks. [10].

C. *Intrusion Detection System (IDS)*

An Intrusion Detection System (IDS) serves as the eyes and ears of cybersecurity, working proactively to monitor network activity and identify the occurrence of attacks. IDS (Intrusion Detection System) is often a signature-based system and is used to secure computer architectures [11] as well as cloud infrastructures [12].

IDS can be integrated with private cloud architectures such as OpenStack. Implementation results have shown that IDS can also improve the security of OpenStack [13].

D. *Gap Analysis*

The integration of Reverse Proxies with other security techniques has also been explored in several advanced studies to enhance the security of web-based applications. For example, one study combined a web application firewall, Fail2ban, an nginx reverse proxy, and PostgreSQL databases to secure the Company's web-based applications, demonstrating that the proposed solutions effectively protect user applications.[14]. Another study about the integration of Nginx reverse proxy architecture, DNS Bind services to improve the access of institutional web-based applications [15].

A review of previous studies reveals existing gaps. Most research has focused on using Nginx as a load balancer or standalone web application firewall, but has not explored its role in isolating a multi-VM cloud architecture on a limited-scale infrastructure. Additionally, the integration of Nginx with other security solutions, such as Fail2ban, Suricata, and iptables, for implementing a Defense-in-Depth strategy remains underexplored. Addressing this gap is essential for small and medium-sized organizations that require affordable yet practical solutions to secure their private cloud environments.

In addition to academic research, many businesses use Content Delivery Networks (CDNs) and cloud-based Web Application Firewalls (WAFs) to protect against DDoS and intrusion threats. These services offer strong protection and use large-scale infrastructure, but they cost a lot of money and require businesses to send critical data traffic to third-party suppliers. This kind of dependence could raise concerns about data sovereignty, making it impossible for small and medium-sized businesses with limited financial resources. This study, on the other hand, uses open-source technologies in a private cloud environment. This allows institutions to maintain complete control over their data and infrastructure while achieving security results comparable to those of commercial products.

This gap presents an opportunity for research focused on designing and implementing Nginx reverse proxy architectures in multi-VM-based private cloud architectures, serving not only as a load balancer or standalone WAF but also as part of an integral layered security system. Therefore, this study aims to fill this gap by presenting practical, affordable, and replicable solutions, particularly for small and medium-sized organizations.

III. System Architecture

This section contains information about the system architecture that was implemented and tested in this study. And the methodology used in the study

A. *Overall System Architecture*

The system architecture to be tested in this study is presented in Fig. 1.`

As illustrated in Fig. 1, the overall flow for the proposed architecture begins with the incoming traffic from the internet, which is passed to the reverse proxy machine. The Nginx reverse proxy is the primary security gateway, as seen in Fig. 1. It filters traffic coming in from the Internet. After that, the reverse proxy forwards requests to application VMs: e.g., the primary web server, application VM 1, and application VM 2. This keeps things separate and provides an additional layer of security against potential threats. Requests from the internet first go through the Nginx reverse proxy. It demonstrates how reverse proxies and web-based applications function on a private cloud based on Proxmox. We chose Proxmox because it is open-source and can be easily set up to meet the user's specific needs. You may also set up Suricata and Fail2ban to watch traffic coming in from the public IP.To maintain confidentiality and protect the production environment, the Public IP Address and domain shown in the pictures have been intentionally obfuscated.

B. *Nginx Reverse Proxy VM Configurations*

For VM, nginx servers use the Ubuntu LTS 24.04 Operating System, which then installs several supporting applications as follows:

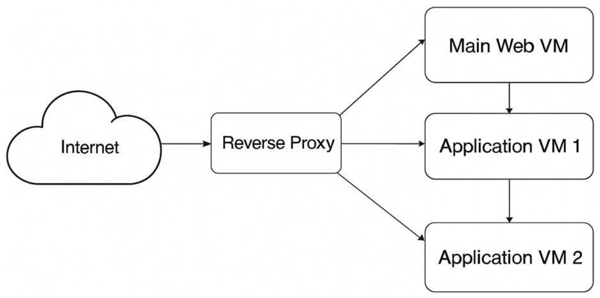


Fig. 1. The suggested system architecture for the private cloud environment

- Nginx
- Fail2ban
- Suricata IDS

The architecture was deployed using the latest stable releases of Nginx, Suricata, and Fail2ban. Additionally, adjustments were made to the rule settings on the built-in Linux firewall, ufw, as well as iptables. So the architecture of the reverse proxy VM is as follows.

Suricata is configured by monitoring the data traffic that enters the public IP address. In Fig. 2, the Public IP address used is 203.114.0.0/16, through which Suricata will monitor incoming data traffic. The address is installed on Ethernet interface one on the Reverse Proxy VM, while Ethernet 2 will be populated with a Private IP address that has a subnet of 192.168.x.x/24. Nginx reverse proxy is configured according to the needs of each application.

C. *Security Configurations*

For Suricata configurations, use the default Suricata rules that are updated using the default command, which is updated periodically based on the standard ET Open rules. Then, for Fail2ban, use the following configuration snippet on Fig. 3.

As shown in Fig. 3, an IP address is banned for 3600 seconds if it fails authentication 10 times within a 600-second window. This rule helps mitigate brute-force attempts by automatically blocking suspicious sources.. Additionally, UFW and iptables are used to restrict open ports to only 80 (HTTP) and 443 (HTTPS), thereby minimizing the attack surface.

D. *Back-end web Application*

Each back-end VM runs LAMP (Linux, Apache, MySQL, and PHP) and PhpMyAdmin for database Management. Applications are isolated on separate VMs to reduce interdependency and prevent single points of failure.

IV. EXPERIMENTAL RESULT

This section presents the results of implementing the designed architecture. A more detailed explanation of the actions taken is provided in Section III, including instructions on how to install each component and the penetration testing that is conducted. The validation test of the components of each web-based application has also been performed, and it has shown that they continue to run normally, just as they would in a monolithic application. Due to access limitations and security policies on the organizational network, detailed Suricata and Fail2ban log files could not be retrieved for inclusion in this paper. However, the system's response was validated through penetration testing, as described in this section.

A. *Penetration testing*

After the installation and preparation of each component in section III. So the next stage is to conduct penetration testing on the proposed architecture. The penetration testing undertaken in this study is as follows:

- Port Scanning test: nmap
- ICMP flood test: hping3
- SYN TCP flood test: hping3
- Directory scanning test: dirb

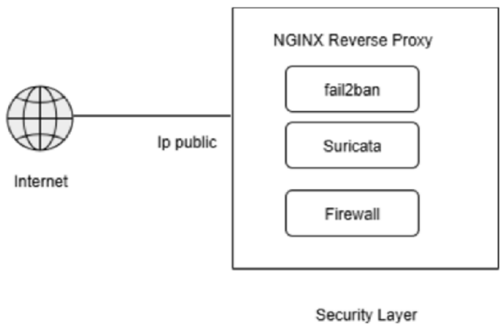
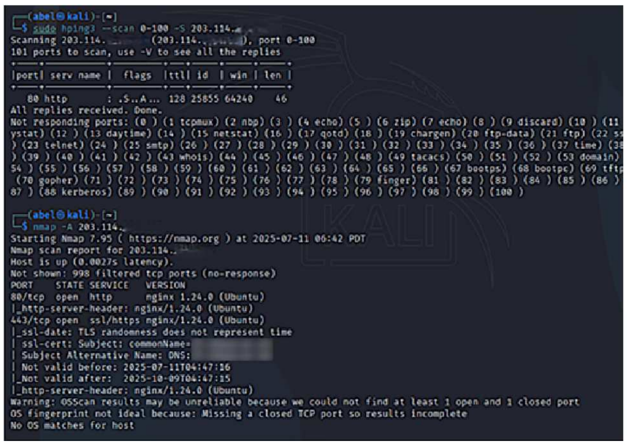


Fig. 2. Architecture for VM Reverse Proxy

```
[DEFAULT]
bantime = 3600
findtime = 600
maxretry = 10
backend = auto
```

Fig. 3. Example Configuration for Fail2Ban Jail Rule



```
┌──(abel㉿kali)-[~]
└─$ sudo hping3 --scan 0-100 -S 203.114.[redacted]
Scanning 203.114.[redacted] (203.114.[redacted]), port 0-100
101 ports to scan, use -V to see all the replies
+----+-----------+---------+---+-----+-----+-----+
|port| serv name |  flags  |ttl| id  | win | len |
+----+-----------+---------+---+-----+-----+-----+
   80 http       : .S..A... 128 25855 64240    46
All replies received. Done.
Not responding ports: (0 ) (1 tcpmux) (2 nbp) (3 ) (4 echo) (5 ) (6 zip) (7 echo) (8 ) (9 discard) (10 ) (11
ystat) (12 ) (13 daytime) (14 ) (15 netstat) (16 ) (17 qotd) (18 ) (19 chargen) (20 ftp-data) (21 ftp) (22 ss
) (23 telnet) (24 ) (25 smtp) (26 ) (27 ) (28 ) (29 ) (30 ) (31 ) (32 ) (33 ) (34 ) (35 ) (36 ) (37 time) (38
) (39 ) (40 ) (41 ) (42 ) (43 whois) (44 ) (45 ) (46 ) (47 ) (48 ) (49 tacacs) (50 ) (51 ) (52 ) (53 domain)
54 ) (55 ) (56 ) (57 ) (58 ) (59 ) (60 ) (61 ) (62 ) (63 ) (64 ) (65 ) (66 ) (67 bootps) (68 bootpc) (69 tftp
 (70 gopher) (71 ) (72 ) (73 ) (74 ) (75 ) (76 ) (77 ) (78 ) (79 finger) (81 ) (82 ) (83 ) (84 ) (85 ) (86 )
87 ) (88 kerberos) (89 ) (90 ) (91 ) (92 ) (93 ) (94 ) (95 ) (96 ) (97 ) (98 ) (99 ) (100 )

┌──(abel㉿kali)-[~]
└─$ nmap -A 203.114.[redacted]
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-11 06:42 PDT
Nmap scan report for 203.114.[redacted]
Host is up (0.0027s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT    STATE SERVICE   VERSION
80/tcp  open  http      nginx 1.24.0 (Ubuntu)
|_http-server-header: nginx/1.24.0 (Ubuntu)
443/tcp open  ssl/https nginx/1.24.0 (Ubuntu)
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=[redacted]
| Subject Alternative Name: DNS:[redacted]
| Not valid before: 2025-07-11T04:47:16
|_Not valid after:  2025-10-09T04:47:15
|_http-server-header: nginx/1.24.0 (Ubuntu)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host
```

Fig. 4. Result of port scanning attempt. The UFW responds, only ports 80 and 443 are open to the public.

The first penetration testing test was to do port scanning from Kali Linux using the nmap command. The results of nmap scanning are as follows:

Fig. 4 shows that the scanning experiment identified only 80 open ports, which were detected during the port scanning experiment. While other ports can't be detected, this suggests that the rules of the pre-configured UFW firewall are only opening ports 80 and 443, which are accessible and functioning properly.

The settings related to ports that are opened only 80 and 443 are indicated by using the following command:

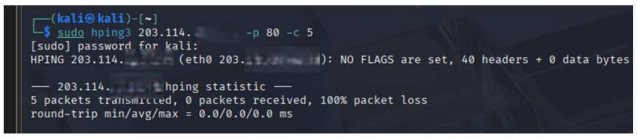
- Sudo ufw allow 80
- Sudo ufw allow 443

According to the results of penetration testing, it was demonstrated that the other ports were not exposed when nmap port scanning was performed. In addition to port scanning, the following test is to be carried out: directory scanning.

The results of directory scanning found that some of the contents of folders from the web application can be detected with the dirb command. To overcome this, a slight modification was made to the Nginx reverse proxy by adding

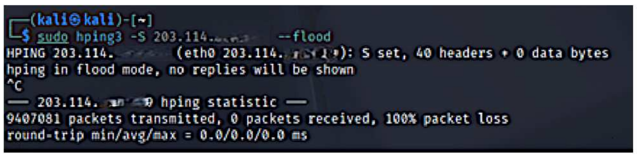
```
server {
    listen 80;
    server_name example.com; # for domain name

    location / {
        autoindex off; # |
        proxy_pass http://backend_servers;
        # ... other configuration...
    }
}
```

Fig. 5. Modification of Nginx Proxy Server to turn off the autoindex for prevent the directory scanning



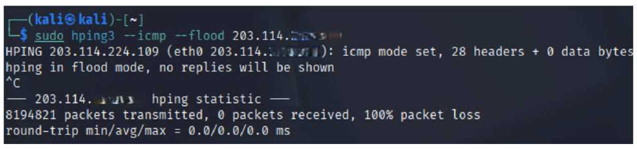
```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 203.114.[redacted] -p 80 -c 5
[sudo] password for kali:
HPING 203.114.[redacted] (eth0 203.[redacted]): NO FLAGS are set, 40 headers + 0 data bytes

── 203.114.[redacted] hping statistic ──
5 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Fig. 6. Result of TCP Null scan using HPing3, which indicates the reverse proxy architecture blocked all the incoming traffic.



```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 -S 203.114.[redacted] --flood
HPING 203.114.[redacted] (eth0 203.114.[redacted]): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
── 203.114.[redacted] hping statistic ──
9407081 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Fig. 7. Result of HPing3 SYN Flood, which indicates the reverse proxy architecture can handle the TCP DDOS attack



```
┌──(kali㉿kali)-[~]
└─$ sudo hping3 --icmp --flood 203.114.[redacted]
HPING 203.114.224.109 (eth0 203.114.[redacted]): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
── 203.114.[redacted] hping statistic ──
8194821 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Fig. 8. Result of ICMP Flood penetration testing, which indicates the architecture

an 'autoindex off' option. The results of these modifications are shown in Fig. 5.

Fig. 5 shows that the implementation of autoindex=off aims to prevent users from browsing directories that lack index files. Although there has been no rescan after this addition, autoindex=off is a practical implementation for securing web servers.

B. *ICMP and TCP DDOS Mitigation*

Resistance tests against DOS attacks were also conducted in this study. The test was performed using Hping3. Here is a test ICMP flood using the HPing3 tools. Command *sudo hping3 <destination IP address> -p 80 -c 5.* The experiment yielded a 0% success rate. This means that the VM Proxy rejects the formation of a SYN connection; the test results are shown in Fig. 6.

The results of the TCP Null scan using HPING3 on port 80 showed that the server received no packets. This demonstrates that iptables and Reverse proxy rules effectively handle DoS attacks.

Then, the second test was carried out by sending a TCP SYN flood to port 80, and the final result showed that out of the 9.4 million packets sent to the server, none of them were successful. This means that the combination of reverse proxies and iptables is effective in mitigating SYN flood attempts. This is referred to in Fig. 7.

The test for an ICMP flood was performed using hping3 with the *command sudo hping3 –icmp –flood 203.104.0.0/16.* The results of this DDOS experiment are shown in Fig. 8. The results of this experiment show that of the 8.1 million submitted, all of them were 100% failed. This means that the iptables rules set in Chapter 3 are working as expected. This referred in Fig. 8.

C. *Research Limitations*

It is important to note that the validation test conducted in this study has certain limitations. This section outlines the limitations of the research tests conducted.

In this study, what has been tested so far is how the architecture survives several penetration testing experiments, such as:

- DDOS attack mitigation
- Port scanning.
- Directory scanning

Additionally, several test scenarios have not been conducted, including the ability to handle incoming data traffic. Then, how this architecture survives large-scale attacks becomes a space for further research to answer this problem. While this study validated the deployment of five applications, the scalability of the proposed architectures to handle large numbers of applications (10,20, or 100 applications) has not yet been tested. This will be addressed in future works, along with the architectures that handle large-scale attacks and VM workloads.

Although detailed logs could not be exported this time, the test confirmed that the proposed architecture is effective in blocking unauthorized access and mitigating common network threats. Due to organizational security policies, the detailed log of Suricata and Fail2ban cannot be included in this study. Future research will address this limitation by designing controlled test environments that enable secure log collection and analysis.

In addition, performance benchmarking (CPU utilization, memory consumption, throughput, and latency) was not conducted in this research, as controlled high-intensity testing could not be undertaken due to organizational restrictions in production mode. This remains an essential limitation, which will be addressed in future research using isolated test environments.

V. Conclusion And Future Works

A. *Conclusion*

This research successfully designed and implemented a reverse proxy architecture combined with layered security, executed on a limited-scale cloud environment. The proposed architectural design was validated with five different applications. This architecture can withstand penetration testing attempts, specifically DDoS TCP attack (blocking 9.4 million packets sent to the server) and ICMP flood experiments (blocking 8.1 million packets). Additionally, access to each web-based application can run smoothly, just like in a monolithic architecture. These results validate the effectiveness of the *Defense in Depth approach* through the integration of Nginx *reverse proxies* with open-source *security solutions* (Fail2Ban, Suricata, *iptables*) as a practical and affordable solution for organizations with *limited-scale* private cloud infrastructure.

B. *Future Works*

This research still has considerable room for further development, particularly in measuring nginx's performance in load balancing access to each web-based application. Additionally, it is necessary to measure the performance of Reverse Proxy VMs to determine how the VM can withstand larger-scale cyberattack attempts. Future work will also provide a detailed log-based analysis of Suricata and fail2ban, and performance benchmarking will be conducted to evaluate resource usage under both normal and attack conditions. Scalability test with larger application sets (10,20, or 100 web applications), and integration with firewalls, such as OPNSense or other firewalls, to assess their performance and the robustness of the architecture.

ACKNOWLEDGEMENT

The authors acknowledge the use of generative AI language models (ChatGPT, GPT-4 model) to assist in refining and editing the language of this manuscript. All ideas, results, and interpretations are solely those of the authors.

REFERENCES

[1] K. I. Nikishin, "Load Balancer of Data in a Distributed Network via Nginx Proxy Server," *Proc. Southwest State Univ.*, vol. 26, no. 3, pp. 98–111, Feb. 2023, doi: 10.21869/22231560-2022-26-3-98-111.

[2] J. Yu, J. Jiang, and W. Ye, "Design and implementation of adaptive dynamic load balancing strategy based on server cluster," *Other Conf.*, p. 90, Jul. 2024, doi: 10.1117/12.3031078.

[3] M. Innuddin, P. Irfan, and R. Hammad, "Improving the Security of an Nginx Web Server with NAXSI as a Web Application Firewall," *J. Apl. Teknol. Inf. dan Manaj.*, vol. 4, no. 2, pp. 148–156, Oct. 2023, doi: 10.31102/JATIM.V4I2.2310.

[4] V. B. A. Pardosi, "Cost-Effective DDoS Mitigation: Leveraging Nginx Reverse Proxy for Enhanced Server Protection," *J. Teknol. Inf. dan Pendidik.*, vol. 17, no. 2, pp. 371–382, Dec. 2024, doi: 10.24036/JTIP.V17I2.845.

[5] L. Ceragioli, P. Degano, and L. Galletta, "Can my firewall system enforce this policy?," *Comput. Secur.*, vol. 117, p. 102683, Jun. 2022, doi: 10.1016/J.COSE.2022.102683.

[6] "Firewall Technology's Importance in the Fight against the Internet Security," *Am. J. Multidiscip. Res. Africa*, Nov. 2021, doi: 10.58314/099098.

[7] S. Ahmadi, "Adaptive Cybersecurity: Dynamically Retrainable Firewalls for Real-Time Network Protection," Jan. 2025, Accessed: Jul. 17, 2025. [Online]. Available: https://arxiv.org/pdf/2501.09033

[8] Q. Zhang, Z. Luo, and J. Zhang, "Analysis of the Application of Firewall and Intrusion Detection Technology in Network Security in the Era of Big Data," *2023 2nd Int. Jt. Conf. Inf. Commun. Eng.*, pp. 167–171, 2023, doi: 10.1109/JCICE59059.2023.00042.

[9] D. Arnaldy and T. S. Hati, "Performance Analysis of Reverse Proxy and Web Application Firewall with Telegram Bot as Attack Notification on Web Server," *2020 3rd Int. Conf. Comput. Informatics Eng. IC2IE 2020*, pp. 455–459, Sep. 2020, doi: 10.1109/IC2IE50715.2020.9274592.

[10] Y. Nikam, S. Ware, T. Patil, H. Waghmare, S. Dedgaonkar, and P. Futane, "Ai-Based Web Application Firewall," in *IEEE International Conference on "Computational, Communication and Information Technology", ICCCIT 2025*, Institute of Electrical and Electronics Engineers Inc., 2025, pp. 136–141. doi: 10.1109/ICCCIT62592.2025.10928035.

[11] K. Jiang and H. Zheng, "Design and Implementation of A Machine Learning Enhanced Web Honeypot System," *Proc. - 2020 13th Int. Congr. Image Signal Process. Biomed. Eng. Informatics, CISP-BMEI 2020*, pp. 957–961, 2020, doi: 10.1109/CISP-BMEI51763.2020.9263640.

[12] H. Gajjar and Z. Malek, "A Survey of Intrusion Detection System (IDS) using Openstack Private Cloud," *2020 Fourth World Conf. Smart Trends Syst. Secur. Sustain.*, pp. 162–168, Jul. 2020, doi: 10.1109/WORLDS450073.2020.9210313.

[13] R. G. Roshan, V. S. Salanke, and S. Nagasundari, "Leveraging OpenStack-based Private Cloud for Intrusion Detection Using Deep Learning Techniques," *2024 First Int. Conf. Women Comput.*, 2024, doi: 10.1109/INCOWOCO64194.2024.10863595.

[14] R. Sime, N. Sezgin, and F. Ağgün, "An Integrated Web Security Application: Integration Of Nginx Reverse Proxy, Fail2ban, Waf, Postgresql and Laravel," *Balk. J. Electr. Comput. Eng.*, vol. 13, no. 1, pp. 106–111, Mar. 2025, doi: 10.17694/BAJECE.1547456.

[15] L. Zhe, D. Gengsheng, Z. Jingjing, and D. Wei, "Research and Implementation of Dual-stack Web Service Architecture Based on Intelligent DNS and Reverse Proxy Technology," *2020 12th Int. Conf. Adv. Infocomm Technol. ICAIT 2020*, pp. 68–73, Nov. 2020, doi: 10.1109/ICAIT51223.2020.9315501.