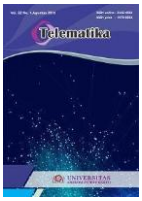

Available online at :
http://ejournal.amikompurwokerto.ac.id/index.php/telematika/

Telematika

Accreditated SINTA "2" Kemenristek/BRIN, No. 85/M/KPT/2020

Comparative Analysis of UFW and CSF Using the SEPER Framework

Arif Kurniawan[1], Muhamad Yusuf[2,*], Agung Budi Prasetio[3]

[1,2,3] Information Technology Study Program, Faculty of Computer Science, Institut Teknologi Tangerang Selatan, Indonesia

ARTICLE INFO

History of the article:
Received September 1, 2025
Revised October 5, 2025
Accepted January 27, 2026

Keywords:
Firewall
Linux Security
UFW
CSF
Performance Evaluation

Correspondence:

E-mail: yusuf@itts.ac.id

ABSTRACT

This study presents a comparative evaluation of two widely used Linux-based firewall solutions, Uncomplicated Firewall (UFW) and ConfigServer Security & Firewall (CSF), using the SEPER framework, which encompasses Security, Performance, Effectiveness, and Reliability dimensions. While previous studies have examined Linux firewall configurations individually, systematic comparisons that apply a structured evaluation framework such as SEPER remain limited. The experiments were conducted on Ubuntu Server using an intra-host virtualized environment consisting of multiple virtual machines. Network performance was evaluated using throughput and latency measurements, while security effectiveness was assessed through port scanning, SSH brute-force simulations, and mild SYN flood scenarios. System reliability was analyzed based on CPU and memory utilization. The results indicate that UFW and CSF exhibit comparable network performance, with throughput differences remaining below 5%, suggesting no statistically significant performance advantage for either firewall. UFW demonstrates slightly lower resource overhead, whereas CSF provides stronger automated brute-force mitigation through its integrated Login Failure Daemon (LFD), at the cost of modestly higher resource usage. Mild SYN flood tests produced similar outcomes across all configurations, largely influenced by Linux kernel-level mechanisms. Overall, this study highlights a trade-off between resource efficiency and advanced security automation. By applying the SEPER framework, the findings provide balanced and practical guidance for Linux administrators in selecting firewall solutions based on deployment priorities rather than isolated performance metrics.

1. INTRODUCTION

Network security is a fundamental aspect of modern information systems. One of the main layers of defense is the firewall, which controls data traffic and prevents unauthorized access (Ernawati et al., 2022; ISO, 2022; Wool, 2009). In practice, threats such as Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks, including SYN Flood or LOIC-based attacks, which are sent continuously, remain a serious problem because they can cripple Linux servers in a short time. Research by Hasani et al. (2024) shows that SYN Flood attacks using Hping3 and LOIC can increase CPU usage by almost 100%, thus emphasizing the need for an effective firewall mitigation mechanism. Similar findings were demonstrated by Arman & Rachmat (2023), who emphasized the role of IPTables firewalls in preventing LOIC attacks while hiding server information from port scanning results. Furthermore, firewall management best

practices emphasize the importance of regular audits and rule reviews to maintain security consistency (Lamdakkar et al., 2024).

In Linux environments, there are various firewall options available, including Uncomplicated Firewall (UFW) and ConfigServer Security & Firewall (CSF). Both are popular due to their relative ease of use, but they have different characteristics. UFW is known for its lightweight design with simple configuration syntax, making it suitable for novice administrators or small servers (Canonical Ltd., 2025; Lamdakkar et al., 2024; Wool, 2009). In contrast, CSF offers more comprehensive security features, including a Login Failure Daemon (LFD) to detect brute-force attacks, protection against port flooding, and integration with popular hosting panels such as cPanel and DirectAdmin (ConfigServer Services, n.d.).

Several previous studies have examined the effectiveness of Linux firewalls, with a focus on configuration techniques, packet filtering efficiency, or resistance to specific attack scenarios. Ariyadi et al. (2023), highlighted the importance of firewall configuration in preventing SSH brute-force attacks, while Salopek & Mikuc (2023); and Niemann et al. (2015), demonstrated that increasingly complex firewall rules can negatively impact network throughput. Other studies evaluated firewall effectiveness in specific application or attack contexts, such as DoS protection in cloud services (Alfazry et al., 2024). Experimental comparisons focusing on throughput and latency have also been reported (Rafamantanantsoa & Rabetafika, 2018). However, most of these studies evaluate performance and security aspects in isolation, without applying a unified evaluation framework that simultaneously considers efficiency, effectiveness, and system reliability.

In recent years, the widespread adoption of low-cost cloud servers and virtual private servers has increased the exposure of Linux systems to external threats. As a result, system administrators are often faced with a practical dilemma when selecting firewall solutions: choosing between lightweight and easy-to-manage tools such as UFW, and more feature-rich solutions such as CSF that provide automated intrusion detection and response (Ariyadi et al., 2023). Despite this growing relevance, systematic comparative studies that address this trade-off in a structured manner remain limited.

To address this gap, this study employs the SEPER framework, which evaluates systems across Security, Performance, Effectiveness, and Reliability dimensions. Unlike ad hoc benchmarking approaches commonly used in previous firewall studies (Ruambo et al., 2025; Rafamantanantsoa & Rabetafika, 2018), SEPER provides a structured methodology that enables balanced interpretation of performance efficiency and security behavior. By applying this framework to UFW and CSF under identical experimental conditions, this study aims to provide clearer guidance for firewall selection based on operational priorities rather than isolated performance metrics.

To date, comparative studies that simultaneously evaluate performance, resource usage, and reliability of UFW and CSF using standardized evaluation methodologies remain limited. Although benchmarking frameworks such as RFC 9411 provide general guidance for network security device evaluation (Balarajah et al., 2023), their application to host-based Linux firewalls—particularly UFW and CSF—has not been widely reported. Therefore, this study aims to conduct a comprehensive evaluation of UFW and CSF firewalls using the SEPER (Security, Performance, Effectiveness, Reliability) framework (Longueira-

Romero et al., 2021; Skybakmoen & Conrad, 2018; Kramer, 2008), providing empirical insights into throughput, latency, resource utilization, and response to common attack scenarios.

2. RESEARCH METHODS

This study uses a laboratory experimental approach with the SEPER framework, the stages of which are shown in figure 1. Research Flow. The experiment begins with the preparation of the test environment, followed by testing (Baseline, UFW, CSF), and the final stage is data analysis.

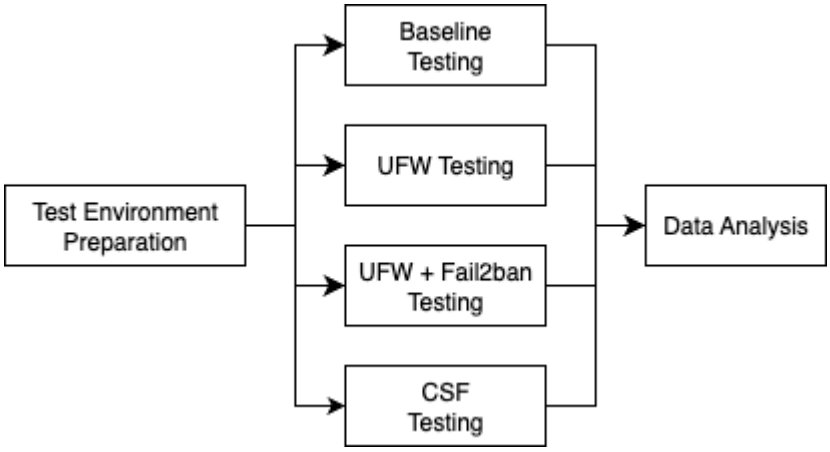

Figure 1. Research Flow

3.1. Test Environment Preparation

This research environment was built using three Ubuntu-based virtual machines (VMs) (Purwoko & Hilal, 2019), running version 24.04 LTS, each with different functionalities. All VMs resided on a single hypervisor and were connected via a virtual switch/bridge (intra-host), rather than a physical 1 GbE boundary. The first machine served as the system under test, serving as the test object. On this server, the UFW and CSF firewalls were alternately activated to compare their performance. The second machine served as an attack generator, used to conduct port scanning, SSH brute-force attacks, and flooding. A mild DoS scenario was modeled to resemble LOIC traffic, using rate-limiting techniques and logging on HTTP connections, based on iptables practices. The attack rate was intentionally limited (approximately 200 packets per second) to represent controlled stress conditions rather than a large-scale distributed attack, allowing firewall-level behavior and resource utilization to be observed without being dominated by kernel-level saturation effects. This approach has been shown to help stabilize the host while reducing fingerprinting from port scanners (Arman & Rachmat, 2023). The third machine served as both a client and a monitoring machine, using Ubuntu with Prometheus and Grafana installed to send normal traffic (load tests) while recording network performance metrics and server resource consumption.

It is important to note that this experimental setup utilizes an intra-host virtualized environment, where all virtual machines operate on the same physical host. As a result, measured throughput values reflect virtual bus performance rather than physical network bandwidth. While this configuration does not represent real-world wide-area or physical network conditions, it provides a controlled and reproducible environment for relative performance comparison between firewall configurations under identical system constraints.

All test tools are installed in this environment, including nmap, hydra, hping3, iperf3, wrk, htop, and pidstat. The topology design is visualized in figure 2. Test Topology, where all VMs are connected through a virtual switch/bridge. With this architecture, attack traffic and normal traffic can be run in parallel, in accordance with the benchmarking methodology recommended by RFC 9411 (Balarajah et al., 2023).

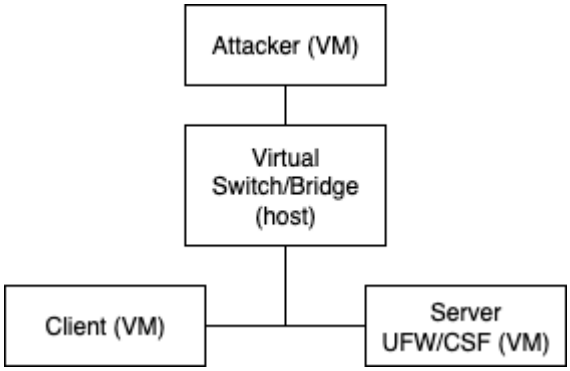

Figure 2. Test Topology

In this experiment, the independent variables used were the firewall types, namely UFW and CSF. The dependent variables covered four evaluation aspects: security effectiveness, measured by time-to-block, detection rate, and false positive probability (Ruambo et al., 2025; Singh et al., 2024); performance, evaluated by TCP/UDP throughput, latency (p50, p75, p90, p99), and requests per second (RPS) in the HTTP test (Ubicloud, 2024; Lencse & Shima, 2023); resource usage, which included CPU, RAM, and I/O usage (Kurek et al., 2024); and reliability, which reflected service stability under flood conditions and consistency of results across three to five iterations (Salopek & Mikuc, 2023). This study adopts a descriptive evaluation approach; therefore, inferential statistical tests were not applied, and results are interpreted based on relative performance trends rather than statistical significance.

To maintain consistency of results, several control variables were applied, such as similar hardware specifications (CPU, RAM) between the server and client, use of the same operating system (Ubuntu LTS), equivalent firewall rules (opening ports 22, 80, and 443), and identical attack types and test loads on both firewalls. The experiment was run on a virtual server with 4 vCPUs, 4 GB of RAM, and 20 GB of storage. The operating system used was Ubuntu 24.04 LTS (Noble) with the Linux kernel 6.8.0-78-generic. The firewalls tested consisted of UFW 0.36.2 (iptables/nftables frontend) and CSF 14.24 (ConfigServer Security & Firewall with Login Failure Daemon). The test environment consisted of three virtual machines (VMs): a client for load testing and monitoring, a server as the test object, and an attacker with the same operating system for attack simulation. All VMs were connected via a switch/bridge with a one-hop distance. As a baseline, initial measurements were performed without a firewall, then compared with the test results using UFW and CSF. For firewall rules, both UFW and CSF used open ports 22, 222, 80, 9100, and 5201 for TCP and port 5201 for UDP.

3.2. Testing

The initial testing phase involved a baseline test, running the server without a firewall to obtain baseline performance and security values for comparison. Once baseline values were obtained, testing continued with the UFW configuration by enabling standard rules (Canonical Ltd., 2025), namely the default deny incoming and allow for SSH, HTTP, and HTTPS services. Efficient firewall rule design on

Ubuntu systems is crucial for minimizing overhead while maintaining security, as emphasized in the study by Ma & Zhao (2022). For brute-force scenarios, UFW was also integrated with fail2ban to simulate an automatic blocking mechanism for repeated login attempts. All security and performance scenarios were then executed, and the test results were recorded.

After UFW testing was completed, the firewall was disabled and replaced with CSF using equivalent rules. CSF was tested with the Login Failure Daemon (LFD) active to model automatic detection and blocking scenarios for brute-force and flood attacks. The testing process was performed using the same procedures as UFW to ensure fair comparison of the results. Overall, this experiment covered five main categories, as summarized in table 1. Test Parameters and Tools, which cover aspects of security, performance, and resource usage, along with the test equipment used for each parameter.

Table 1. Test Parameters and Tools

Category	Tool	Purpose	Key Parameters
Network Performance	Iperf3 (v3.16)	Throughput TCP/UDP & latency with 4 streams, 60 seconds duration, 500 Mbps target	-p 4 -t 60
HTTP Benchmark	Wrk (v4.20)	RPS & latency (p50, p75, p90, p99) with 12 threads, 500 connections, 30 seconds	-t 12 -c500 -d30s
Security	Nmap (v7.94)	Open port detection and filtering with Full TCP scan with -T4 option	-T4 -sS -p-
Security	Hydra (v9.5)	SSH brute-force test with password dictionary	-l testuser -P rockyou.txt
Flood Attacks	Hping3 (v3.0)	SYN flood simulation with 200 packets per second (pps), 30 seconds duration	-S -p 80 -i u500
Resource Monitoring	Prometheus Grafana	CPU and RAM monitoring with Sampling every 1 second, duration 5 minutes	1s interval

Testing tools and parameters were based on standardized configurations and are described as follows. Network port discovery was performed with nmap (v7.94) using a full TCP scan (-T4 -sS -sF -sN -sX -p-) to enumerate open/filtered ports and service responses. The command used in the experiments was: nmap -T4 -sS -p- <IP_TARGET>

SSH brute-force simulations were executed with Hydra (v9.5) to evaluate automatic blocking behavior, using a targeted user and a wordlist with 8 parallel threads for short runs: timeout 30s hydra -l testuser -P /usr/share/wordlists/rockyou.txt -t 8 ssh://<IP_TARGET> This setup allowed measurement of the firewall's time-to-block and login attempt counts.

SYN flood tests (mild DoS) were generated with hping3 (v3.0) by sending SYN packets at approximately 200 packets per second to port 80 for 30 seconds to observe firewall and kernel behavior under connection flood conditions: timeout 30s hping3 -S -p 80 --rand-source -i u5000 <IP_TARGET>

Throughput and network performance were measured using iPerf3 (v3.16) with four parallel TCP streams for 60 seconds: iperf3 -c <IP_TARGET> -P 4 -t 60

Application-level HTTP load was simulated using wrk (v4.20) for 30 seconds with 12 threads and 500 concurrent connections to measure request-per-second (RPS) rate and response latency: wrk -t12 -c500 -d30s "http://<IP_TARGET>/"

System-level metrics (CPU, memory, and logging activity) were collected with Prometheus and Grafana at 1-second sampling intervals for 5 minutes to correlate resource usage with each test scenario. Additional environment details include Fail2ban v1.0.2 and CSF v14.24 used in the experiments. All scenarios (Baseline, UFW, UFW + Fail2ban, CSF) were repeated three times and the reported results present the average and standard deviation across runs to improve reliability and reproducibility. Measurement results are reported as an average ±. For example, TCP throughput was 500 Mbps. To maintain consistency, non-essential services were disabled and the CPU governor was set to performance mode.

3.3. Data Analysis

This stage compares the UFW and CSF results for each test parameter. The test scenario flow can be seen in figure 3. Test Scenario Flow

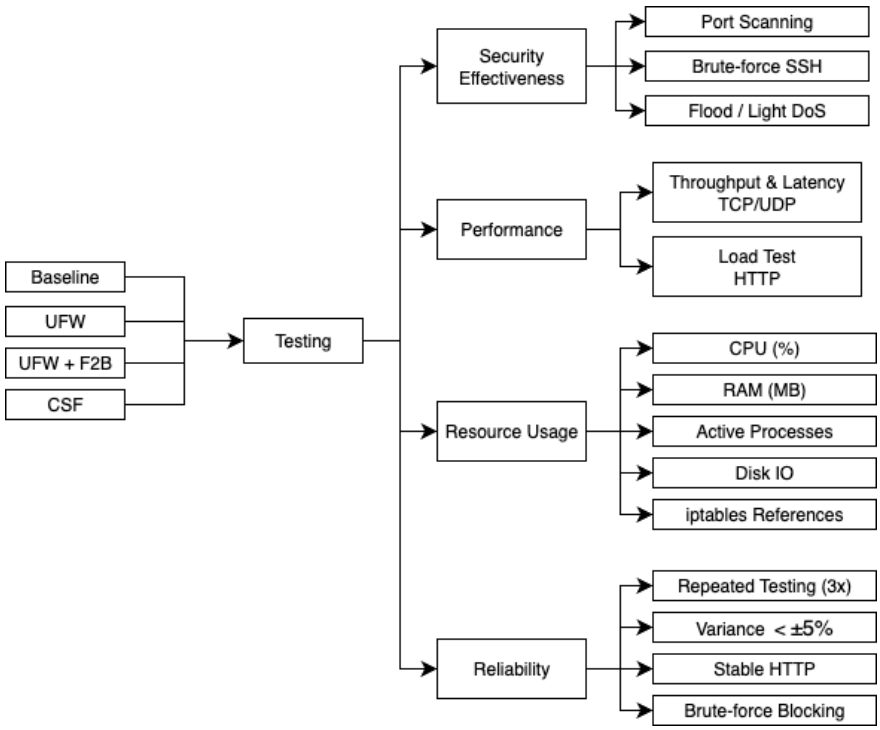

Figure 3. Test Scenario Flow

Security effectiveness testing was conducted across three scenarios to evaluate the firewall's capability in mitigating common network threats. The first scenario examined port scanning resistance using Nmap with SYN, FIN, NULL, and Xmas scan techniques. This test aimed to assess how the firewall responds to port scan attempts by observing the number of ports detected as open or filtered, as well as the average response time (RTT) or latency per scan. Unauthorized ports were expected to be identified as filtered.

The second scenario focused on brute-force SSH mitigation, where attacks were simulated using Hydra against port 22 through multiple username and password combinations. This evaluation measured

the firewall's ability to detect and block brute-force attempts, with metrics including time-to-block, number of attempts before blocking, and false positive rate (Ruambo et al., 2025; Singh et al., 2024). CSF demonstrated more automated brute-force mitigation due to its integrated Login Failure Daemon (LFD), whereas UFW required additional integration, such as Fail2ban, to achieve comparable behavior.

The third scenario evaluated resilience against mild flooding or Denial-of-Service (DoS) attacks at Layers 3 and 4. A controlled SYN flood attack of approximately 200 packets per second was generated using hping3 to measure the firewall's ability to maintain service availability. Observed parameters included malicious packet loss rate, server availability, and CPU utilization spikes (Salopek & Mikuc, 2023). Similar methodologies have been applied in prior studies, where SYN flood attacks generated with hping3 were shown to consume up to 100% of server CPU resources in the absence of firewall protection (Hasani et al., 2024). Additionally, Arman and Rachmat (2023) demonstrated that appropriate firewall configurations can prevent LOIC attacks, thereby maintaining CPU stability and reducing the risk of information leakage through port scanning.

Performance testing was conducted under two primary scenarios. The first assessed throughput and latency using iperf3 to measure network performance while the firewall was active. Testing parameters included TCP streams with 1, 4, and 16 parallel connections, alongside UDP transmissions at transfer rates of 10 Mbps and 100 Mbps. Observed metrics comprised average throughput, packet loss, and round-trip time (RTT) as key performance indicators. The second scenario involved HTTP load testing using wrk or ApacheBench directed at an Nginx server to evaluate the firewall's impact on application-level performance. Concurrency levels of 1, 16, 64, and 256 were applied over a one-minute duration, with metrics including latency percentiles (p50, p75, p90, and p99), requests per second (RPS), and error rates such as timeouts and 5xx responses (Lamdakkar et al., 2024; Ubicloud, 2024).

Resource utilization was continuously monitored using Prometheus, Grafana, and system logs. Recorded parameters included CPU usage, memory consumption, the number of active firewall-related processes, and disk I/O during intensive logging activities. These measurements were further compared with optimization approaches discussed in prior studies on iptables and flowtables (Ubicloud, 2024; Kurek et al., 2024).

Reliability was evaluated by repeating each test scenario at least three times. Results were considered reliable when variations remained within ±5% of the mean, a threshold commonly adopted in experimental system evaluation research (Salopek & Mikuc, 2023). Additional reliability assessment focused on HTTP service stability during DoS simulations and the consistency of the firewall's brute-force blocking mechanism (Salopek & Mikuc, 2023; Ruambo et al., 2025).

The experimental procedure was executed in sequential stages. Baseline measurements were first obtained from a server operating without firewall protection to establish reference performance and security values. Subsequently, the UFW firewall was enabled using standard rules, followed by replacement with CSF configured under default settings. Performance testing was then conducted to measure TCP/UDP throughput and HTTP latency, after which effectiveness testing was performed through port scanning and SYN flood scenarios. Resource utilization was monitored under both idle and load conditions to evaluate CPU and memory behavior. The final stage involved documenting reliability aspects by observing fluctuations in resource usage and service stability over a 30-minute testing period.

3. RESULTS AND DISCUSSION

3.1. Port Scan Test Results

Tests were conducted using nmap using various scanning methods on the server under three conditions: when the server was running without a firewall (baseline), when the UFW firewall was enabled, and when the CSF firewall was enabled.

Table 2. Port Scan Test Results

Firewall	Scan Method	Open Ports Detected	Filtered Ports	Average Latency (ms)	Note
Baseline	SYN Scan	9 (22, 80, 2222, 4330, 5201, 9100, 44321, 44322, 44323)	-	0.0223 ms	All ports are clearly visible
UFW	SYN Scan	5 (22, 80, 2222, 5201, 9100)	-	0.0580 ms	5 ports appear as open, 0 ports are detected as filtered
UFW + F2B	SYN Scan	5 (22, 80, 2222, 5201, 9100)	-	0.0580 ms	5 ports appear as open, 0 ports are detected as filtered
CSF	SYN Scan	4 (22, 80, 2222, 5201)	-	0.0627 ms	Similar to UFW, port 22 remains open according to the default configuration
Baseline	Xmas Scan	0	9 (22, 80, 2222, 4330, 5201, 9100, 43321-44323)	0.0227 ms	All ports appear as open|filtered, none are completely hidden
UFW	Xmas Scan	0	-	0.1500 ms (timeout)	All ports successfully hidden (timeout)
UFW + F2B	Xmas Scan	0	-	0.1500 ms (timeout)	All ports successfully hidden (timeout)
CSF	Xmas Scan	0	-	0.1867 ms (timeout)	All ports successfully hidden (timeout)

In table 2. Port Scan Test Results, all open ports (22, 80, 2222, 4330, 5201, 9100, 44321, 44322, 44323) were clearly visible during the port scan. With UFW and CSF, unauthorized ports were successfully hidden and marked as filtered. CSF detected fewer open ports compared to UFW under the same configuration, indicating stricter default filtering behavior. In the Xmas scan method, both UFW and CSF were able to hide all open ports and even timed out, allowing nmap to pass through the host, indicating fairly effective filtering. In terms of average latency, both UFW and CSF added approximately ±0.01 ms during the SYN Scan and 0.367 ms during the Xmas Scan. However, these values are still within reasonable limits and do not significantly impact service performance.

3.2. SSH Brute-force Test Results

This test uses Hydra to perform an SSH brute-force attack on port 22 with 100 login attempts per second.

Table 3. SSH Brute-force Test Results

Firewall	Blocking time (seconds)	Number of attempts before blocking	Max retry	Note
Baseline	N/A	100+	N/A	The server accepts all brute-force attacks without protection.
UFW	N/A	100+	N/A	Same as baseline, requires fail2ban.
UFW+F2B	3	13	3	Fail2ban automatically blocks IP addresses with the default configuration.
CSF	1	8	5	The default LFD automatically blocks IP addresses; with the default configuration, threshold tuning is required.

In table 3 SSH Brute-force Test Results, the baseline scenario of a server without a firewall accepts all brute-force attempts without any blocking (blocking time: N/A). With UFW, the attack is only blocked when Fail2ban is added. In contrast, CSF with LFD is able to detect brute-force patterns faster, blocking within 1 second with ±100 attempts. These results confirm that CSF is more effective for brute-force mitigation, while UFW requires additional integration to be comparable. This finding is in line with the research of Singh et al. (2024) who showed the importance of fast detection of SSH brute-force attacks in a real environment and is also consistent with Ariyadi et al. (2023) who studied the implementation of iptables for brute-force attack mitigation on the SSH protocol in Ubuntu. For practical cases in a hosting environment, CSF demonstrates faster and more automated brute-force mitigation compared to UFW. This test does not measure the false positive rate directly, because it does not simulate a legitimate login process. False positive evaluation requires additional scenarios with valid inbound traffic that were not included in this study.

3.3. Mild DoS Attack Test Results

The test was conducted using hping3 to conduct a mild DoS attack based on a SYN flood with an intensity of approximately 200 packets per second (pps). The goal was to observe how the firewall responded and its impact on service availability.

Table 4. Mild DoS Test Results

Firewall	CPU Usage	Memory Usage	Packet Transmitted	Dropped Packet
Baseline	0.9 %	441 MB	11930	100 %
UFW	1.2 %	452 MB	11908	100 %
UFW+F2B	1.3 %	477 MB	11932	100 %

Firewall	CPU Usage	Memory Usage	Packet Transmitted	Dropped Packet
CSF	1.8 %	487 MB	11914	100 %

Table 4 shows that in a mild DoS (mild SYN flood, 200 pps for 30 s), the baseline results show that 100% of packets are dropped even without a firewall. This is not an effect of the firewall, but rather a built-in Linux kernel mechanism (e.g., SYN backlog and SYN cookies) that automatically throttles connections. Therefore, in mild DoS, all scenarios appear equal. This is consistent with the hybrid DoS mitigation approach studied by Salopek & Mikuc (2023), where a combination of software filtering can suppress most flood traffic. It also aligns with research by Tambunan & Neyman (2024), which emphasizes the importance of a Linux firewall as an initial layer of protection in preventing the impact of DoS attacks.

3.4. Resource Monitoring Results

The test also recorded server resource usage (CPU & RAM) when the firewall was active, in idle mode (without an attack), and during a mild DoS. In table 5 Resource Consumption Comparison, CPU usage remained low across all scenarios, with baseline values of approximately 0.9%, UFW around 1.0–1.2%, and CSF up to 1.8% under load. This is relatively low because the packet size sent is only 200 pps (u5000) over 60 seconds. Memory consumption is also low, at 441 MB in the baseline state, 452 MB in UFW, 477 MB in UFW+Fail2ban, and 487 MB in CSF. However, CSF is slightly heavier than UFW due to additional features such as the Login Failure Daemon (LFD), automatic logging, and integration. Both firewalls are still able to maintain service stability, despite the differences in CPU consumption.

Table 5. Resource Consumption Comparison

Usage	Baseline	UFW	UFW+F2B	CSF
CPU Idle	0.9 %	0.9 %	1 %	1 %
CPU Usage DoS	1.1 %	1.2 %	1.2 %	1.8 %
Memory Usage Idle	260 MB	260 MB	277 MB	280 MB
Memory Usage DoS	441 MB	452 MB	477 MB	487 MB

3.5. Comparison of UFW vs. CSF Effectiveness

After conducting port scans, SSH brute-force attacks, mild DoS attacks, and resource monitoring, both UFW and CSF proved effective against basic attacks. Their advantages and disadvantages are shown in table 6. Comparison of UFW vs. CSF Effectiveness.

Table 6. Comparison of UFW vs. CSF Effectiveness

Firewall	Advantages	Disadvantages
UFW (Uncomplicated Firewall)	- Lightweight and easy to use. - Direct integration with Ubuntu (default). - Suitable for individual users or small servers.	- Limited features in basic configuration. - Lacks a log-based automatic blocking system or LFD (Login Failure Daemon).

CSF (ConfigServer Security & Firewall)	- Provides advanced security features (e.g., automatic blocking of brute-force attacks, integration with cPanel/WHM). - More detailed logging, aiding forensic analysis. - Suitable for multi-user hosting or servers.	- Heavier than UFW (requires additional resources). - More complex initial configuration.

UFW and CSF significantly reduced the number of malicious packets while maintaining service stability. In this study, baseline values without a firewall were used as control values to assess the firewall's impact on performance and security. The UFW+Fail2ban configuration was presented as a supplementary measure to demonstrate that UFW can be enhanced with external modules. However, the primary comparison focuses on UFW and CSF because they are both commonly used Linux firewalls and are available by default, allowing for direct testing under standard conditions. Therefore, UFW+Fail2ban is not considered a direct competitor to CSF, but rather an additional variant to complement the analysis.

3.6. Network Performance Results

The results of the throughput test using iperf3 are shown in table 7 the iperf3 test results indicate that UFW and CSF are able to maintain relatively high data transfer rates, with only a small difference. In the TCP throughput test, UFW achieved an average of 59000 Mbps, UFW+Fail2ban 61000 Mbps, and CSF approximately 60000 Mbps. The difference is only about 0.5%, so it can be considered insignificant. The high throughput (>50000 Mbps) occurs because the tests are conducted between VMs on a single host, so they are not limited by the physical 1 GbE NIC, but rather by the hypervisor's memory/CPU. For UDP throughput, both are identical at 500 Mbps, with an average latency (jitter) of 0.004 ms for UFW and 0.0043 ms for UFW+Fail2ban and CSF. These results indicate that neither firewall introduces significant overhead to network performance, although UFW+Fail2ban and CSF tend to add slightly more jitter than UFW due to the additional filtering process.

Table 7. The iperf3 test results

Firewall	TCP Throughput	Retransmit	UDP Throughput	UDP Loss	UDP Jitter
Baseline	57000 Mbps	6725	500 Mbps	0.0057 %	0.0047 ms
UFW	59000 Mbps	6950	500 Mbps	0.0061 %	0.0030 ms
UFW+F2B	61000 Mbps	14337	500 Mbps	0.0093 %	0.0043 ms
CSF	60000 Mbps	8979	500 Mbps	0.0103 %	0.0043 ms

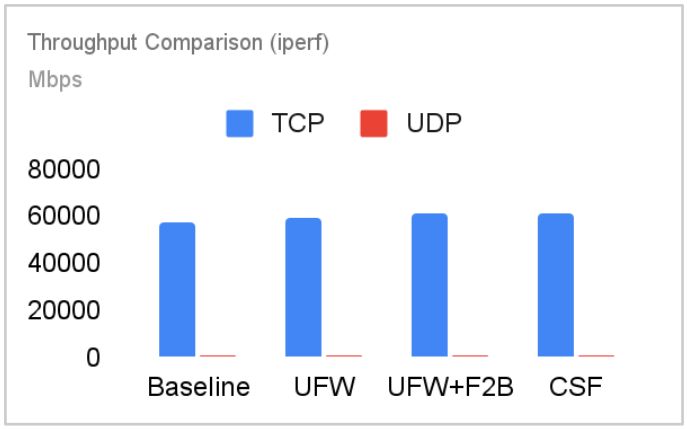

Figure 4. Throughput comparison (iperf3)

Figure 4 throughput comparison (iperf3) shows the average throughput comparison between UFW and CSF on TCP and UDP tests. The observed throughput differences between UFW and CSF remain below 5% across all test scenarios and are therefore considered insignificant within this experimental context. This indicates that both firewalls provide comparable network performance under identical conditions. Additional background processes in CSF, such as log monitoring and intrusion detection via the LFD daemon, may contribute to minor variations without significantly affecting overall throughput. While these differences are negligible in the tested environment, they may become more relevant in high-traffic production systems with sustained workloads and complex filtering rules.

Table 8. Wrk Test Results

Firewall	RPS	p50 Latency	p75 Latency	p90 Latency	p99 Latency
Baseline	164499	0.224 ms	14.81 ms	30.22 ms	44.72 ms
UFW	164372	0.225 ms	14.90 ms	30.50 ms	44.81 ms
UFW+F2B	163524	0.226 ms	14.83 ms	30.39 ms	44.52 ms
CSF	163555	0.226 ms	14.79 ms	30.37 ms	44.50 ms

HTTP load testing using wrk in Table 8. Wrk Test Results show small differences in latency metrics (p50, p75, p90, p99) and Requests Per Second (RPS) as shown in table 8 UFW is able to achieve 164,372 RPS with a latency of p50 0.225 ms, p75 14.9 ms, p90 30.50 ms, and p99 44.81 ms. While CSF recorded 163,555 RPS with a latency of p50 0.226 ms, p75 14.79 ms, p90 30.37 ms, and p99 44.50 ms. The difference is very small (±0.1 ms) and is not significant. Overall, the firewall overhead at the software level is relatively small, so UFW and CSF are still able to maintain stable web application performance.

3.7. Resource Utilization Results

Table 9. Resource Usage Results

Firewall	CPU Idle	CPU Load	RAM Idle	RAM Load
Baseline	0.9 %	0.9 %	260 MB	472 MB
UFW	0.9 %	1.0 %	260 MB	484 MB

UFW + F2B	1 %	1.5 %	277 MB	499 MB
CSF	1 %	1.5 %	280 MB	490 MB

From table 9 resource Usage Results, at idle Baseline H0.9% CPU & 260 MB RAM; UFW H0.9-1.0% CPU & 260 MB RAM; UFW+Fail2ban H1-1.5% CPU & 277 MB RAM; CSF H1-1.5% CPU & 280 MB RAM. CSF requires additional memory allocation due to the presence of the always-on LFD daemon process. At load, RAM increases moderately (472-499 MB) with a small increase in CPU and the CSF pattern is slightly higher due to the always-on LFD daemon. These results indicate that UFW operates with lower resource overhead under the tested conditions, while CSF requires additional CPU and memory due to continuous log monitoring and the Login Failure Daemon (LFD). This reflects a trade-off between resource efficiency and enhanced security automation. Minor variations between resource usage values across tables are attributed to differences in measurement intervals and workload timing, as monitoring was conducted continuously with 1-second sampling resolution.

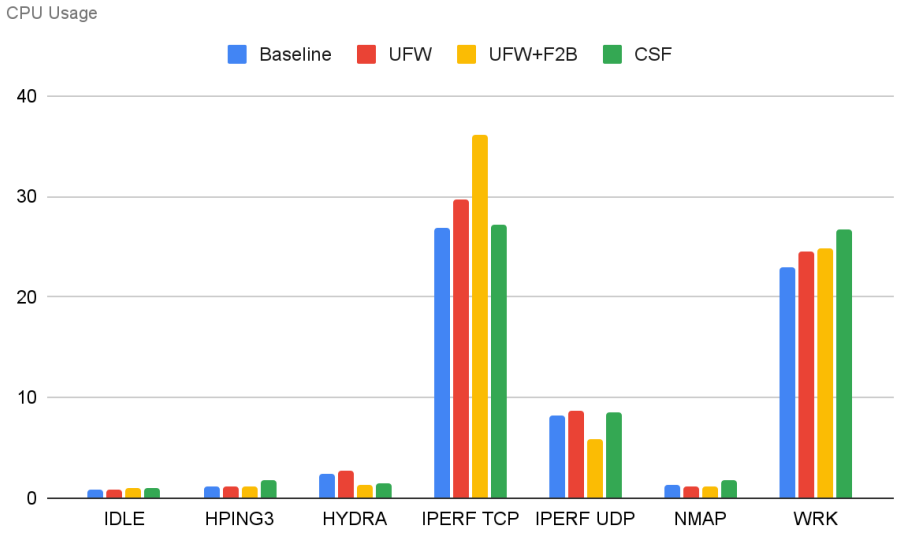

Figure 5. CPU Usage

Figure 5 CPU Usage for UFW vs. CSF shows the difference in CPU consumption. While the difference is relatively small, CSF's tendency to use slightly more CPU consistently appears under load conditions.

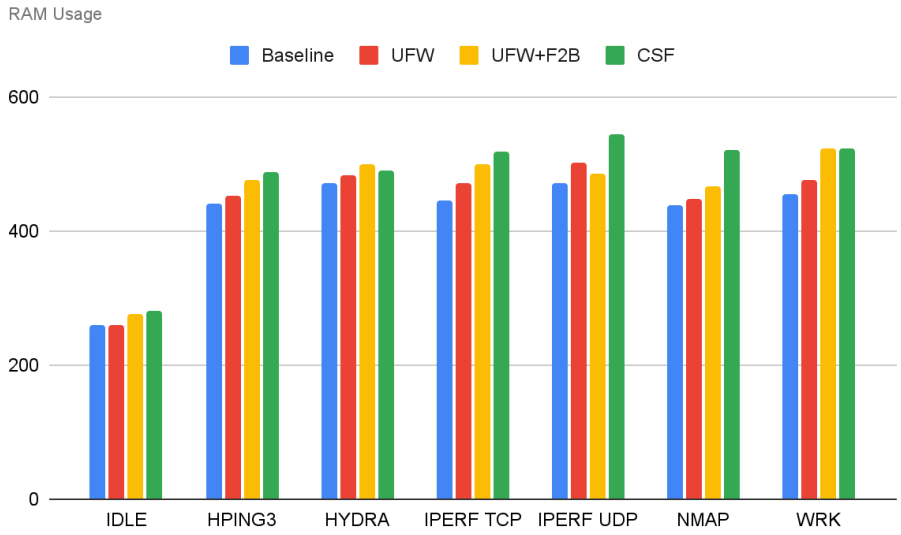

Figure 6. RAM usage

Figure 6 RAM usage for UFW vs. CSF shows that UFW uses memory more efficiently, while CSF requires stable additional allocations, both during idle and load periods. This difference can be attributed to CSF's automatic attack detection and logging mechanisms.

4. CONCLUSIONS AND RECOMMENDATIONS

The test results show fundamental differences between UFW and CSF in terms of performance, effectiveness, reliability, and security. Overall, UFW and CSF exhibit comparable performance under the tested conditions, with UFW demonstrating lower resource overhead due to its lightweight design, while CSF excels in comprehensive security features and automated attack detection through the Login Failure Daemon (LFD).

In terms of performance, iPerf3 tests between VMs on a single host yielded an average TCP throughput of approximately 57000–61000 Mbps. UFW recorded approximately 59100 Mbps, while CSF achieved 60700 Mbps with slightly higher jitter. For the HTTP load test (wrk). The latency difference between the two firewall was relatively small (p75 H14.8-14.9 ms; p99 H44-45 ms), so neither firewall introduced significant overhead to web application services. This high throughput is due to the tests being conducted within the host (via a virtual switch), rather than at the physical limits of a 1 GbE NIC.

In terms of effectiveness, both UFW and CSF successfully hide unauthorized ports from Nmap scans (filtered/timed). CSF outperforms SSH brute-force mitigation, being able to block in ±1 second (H8 attempts) thanks to the built-in LFD, while UFW requires Fail2ban integration (H3 seconds, 13 attempts). All firewalls, even standard ones, discard light SYN flood packets (~200 pps), as the linux TCP/IP stack has limited connection backlogs, making the firewall's role more visible in logging and notification. This finding aligns with research by Hasani et al. (2024) who emphasized the effectiveness of Linux firewalls in reducing the impact of SYN Flood attacks by up to 75% and Arman & Rachmat (2023) who highlighted the role of IPTables in resisting LOIC attacks and preventing information leakage through port scanning. Thus, despite their different approaches, both UFW and CSF remain consistently effective against basic

attacks. From a reliability perspective, UFW proved more efficient, with an average CPU idle time of 0.9–1.0% and DoS of 1.2%, RAM idle time of 260 MB and DoS of 452 MB. CSF was slightly more demanding, with CPU idle time of 1% and DoS of 1.8%, RAM idle time of 280 MB and DoS of 487 MB, due to the additional daemons running for logging and monitoring. Although CSF is more resource-intensive, its automatic logging mechanism makes the system more stable for long-term operation. These results are consistent with a study by Kurek et al., (2024) which emphasized the importance of stable resource consumption in firewall operation

Based on theoretical analysis and practical testing of UFW (Uncomplicated Firewall) and CSF (ConfigServer Security & Firewall) on Ubuntu Server, it can be concluded that both are equally effective against basic attacks such as port scanning and mild DoS attacks, with relatively comparable success rates. UFW demonstrated better resource efficiency under the tested conditions due to its lightweight design, while CSF provides stronger automated security features at the cost of higher resource usage. In contrast, CSF excels in advanced security features, such as automatic log-based brute-force blocking, alerts, integration with cPanel/WHM, and more detailed logging (ConfigServer Services, n.d.).

This finding is consistent with studies by Singh et al. (2024) and Skybakmoen & Conrad (2018) who emphasized the importance of quickly detecting SSH brute-force attacks, and Salopek & Mikuc (2023) who demonstrated the effectiveness of filtering software in mitigating traffic flooding. From a best practice perspective, regular audits of firewall rules are crucial as part of server hardening (Lamdakkar et al., 2024). This trade-off aligns with the SEPER framework, where UFW performs better, while CSF offers stronger effectiveness and security (Kramer, 2008). Therefore, firewall selection largely depends on user needs. For small servers or individual use, UFW is more appropriate due to its simplicity and efficiency. However, for hosting servers or organizations with many users, CSF is more suitable due to its more comprehensive security features. Overall, UFW is more suitable when efficiency and simplicity are prioritized, while CSF is better suited for environments that require proactive security automation despite higher resource usage.

For further research, the scope can be expanded by adding comparisons to other firewalls such as pure iptables, nftables, or enterprise solutions like pfSense or Fortinet, to gain a broader picture of the performance and effectiveness of various platforms.

Testing is also recommended to include more complex attack scenarios, such as large-scale DDoS with packet rates >100 kpps, SQL injection, RDP brute-force attacks, and layer 7 application attacks, to more closely represent real-world conditions. Furthermore, testing on real traffic with high loads on a production network will demonstrate firewall performance in a more realistic environment than intra-host VMs.

For practitioners or server administrators, firewall selection should be tailored to operational needs. UFW is recommended for lightweight servers that require efficiency and simple configuration, while CSF is more suitable when additional security features, monitoring capabilities, and management are needed in a multi-user environment. Furthermore, implementing the defense-in-depth principle is also important, for example by combining a firewall with an IDS/IPS (Snort, Suricata) or a WAF (Web Application Firewall), to achieve stronger, layered protection. Further research could also explore integrating firewalls with

modern orchestrations (Docker, Kubernetes) and configuration automation, so that firewalls are not only secure but also scalable across cloud and multi-tenant infrastructures.

REFERENCES

Alfazry, M. R., Fadilah, F., Putra, A. P., & Setiawan, A. (2024). Perlindungan Keamanan Website NextCloud: Mengatasi Serangan DoS dengan Konfigurasi Firewall pada Ubuntu. *Journal of Internet and Software Engineering*, *1*(3), 11-11.

Arman, M., & Rachmat, N. (2023). Penanggulangan Serangan LOIC Terhadap Web Server. *Techno. com*, *22*(3).

Ariyadi, T., Pohan, M. R., Hadi, M. K., & Widodo, A. A. (2023). Implementasi firewall pada protokol SSH Linux Ubuntu menggunakan iptables. In *Prosiding Seminar Riset Mahasiswa* (Vol. 1, No. 1, pp. 170-175).

Balarajah, B., Rossenhoevel, C., & Monkman, B. (2023). RFC 9411: Benchmarking Methodology for Network Security Device Performance.

Canonical Ltd. (2025). Firewall. In *Ubuntu Community Help Wiki.* Retrieved from https://help.ubuntu.com/community/Firewall

ConfigServer Services. (n.d.). *ConfigServer Security & Firewall (CSF).* Retrieved from https://configserver.com/configserver-security-and-firewall

Ernawati, R., Ruslianto, I., & Bahri, S. (2022). Implementasi metode port knocking pada sistem keamanan server ubuntu virtual berbasis web monitoring. *Coding: Jurnal Komputer dan Aplikasi*, *10*(01), 158-169.

Hasani, F. R., Sardjono, S., & Alamsyah, R. Y. R. (2024). Pencegahan Serangan DDOS Syn Flood Terhadap Web Server. In *Seminar Nasional Penelitian (SEMNAS CORISINDO 2024)* (pp. 124-130).

International Organization for Standardization. (2022). ISO/IEC 27001:2022 — Information security, cybersecurity and privacy protection—Information security management systems—Requirements. Geneva, Switzerland: ISO.

Kramer, W. T. (2008). Holistic Evaluation of Lightweight Operating Systems using the PERCU Method.

Kurek, T., Niemiec, M., & Lason, A. (2024). Performance evaluation of a firewall service based on virtualized IncludeOS unikernels. *Scientific Reports*, *14*(1), 557.

Lamdakkar, O., Ameur, I., Eleyatt, M. M., Carlier, F., & Ait Ibourek, L. (2024). Toward a modern secure network based on next-generation firewalls: recommendations and best practices. *Procedia Computer Science*, *238*, 1029-1035.

Lencse, G., & Shima, K. (2023). Optimizing the performance of the iptables stateful NAT44 solution. *Infocommunications Journal*, *15*(1), 55-63.

Longueira-Romero, Á., Iglesias, R., Gonzalez, D., & Garitano, I. (2021). How to quantify the security level of embedded systems? a taxonomy of security metrics. *arXiv preprint arXiv:2112.05475.*

Ma, L., & Zhao, D. (2022, September). Research on Setting of Two Firewall Rules Based on Ubuntu Linux System. In *2022 International Conference on Computer Network, Electronic and Automation (ICCNEA)* (pp. 178-182). IEEE.

Niemann, R., Pfingst, U., & Göbel, R. (2015). Performance evaluation of Netfilter: a study on the performance loss when using Netfilter as a firewall. *arXiv preprint arXiv:1502.05487.*

Purwoko, M., & Hilal, H. (2019). Analisis Penerapan Firewall Nftables Sebagai Sistem Keamanan Server Pada Mesin Virtualisasi. *InComTech: Jurnal Telekomunikasi dan Komputer*, *9*(1), 1-22.

Rafamantanantsoa, F., & Rabetafika, H. L. (2018). Performance Comparison and Simulink Model of Firewall Free BSD and Linux. *Communications and Network*, *10*(04), 180.

Ruambo, F. A., Masanga, E. E., Lufyagila, B., Ateya, A. A., Abd El-Latif, A. A., Almousa, M., & Abd-El-Atty, B. (2025). Brute-force attack mitigation on remote access services via software-defined perimeter. *Scientific Reports*, *15*(1), 18599.

Salopek, D., & Mikuc, M. (2023). Enhancing mitigation of volumetric ddos attacks: A hybrid fpga/software filtering datapath. *Sensors*, *23*(17), 7636.

Singh, S. K., Gautam, S., Cartier, C., Patil, S., & Ricci, R. (2024). Where The Wild Things Are: Brute-Force SSH Attacks In The Wild And How To Stop Them. In 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24) (pp. 1731-1750).

Skybakmoen, T., & Conrad, C. (2018). Next generation firewall comparative report. *NSS Labs, Fort Collins, Colorado, Tech. Rep.*

Tambunan, M. R. H., & Neyman, S. N. (2024). Implementasi Firewall pada Linux untuk Pencegahan Dari Serangan DoS. *Journal of Technology and System Information*, *1*(4), 10-10.

Ubicloud. (2024, March). Linux flowtables improve latency ~7.5%. *Ubicloud Blog.* Retrieved from https://www.ubicloud.com/blog/improving-network-performance-with-linux-flowtables

Voronkov, A., Martucci, L. A., & Lindskog, S. (2019). System administrators prefer command line interfaces, don't they? an exploratory study of firewall interfaces. In *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)* (pp. 259-271).

Wool, A. (2009). Firewall configuration errors revisited. *arXiv preprint arXiv:0911.1240.*