International Journal of Science and Technology (IJST)
ISSN: 3049-1118, Volume- 3, Issue- 2 (April - June 2026)

Hybrid Security Framework Integrating Firewall and VPN Technologies

S. Senthil Nathan[1], R. Marimuthu[2]
[1]Department of Cyber Security, Dr. MGR Educational and Research Institute Chennai, Tamil Nadu, India
[2]FACULTY, Centre for Cyber Forensics and Information Security, University of Madras, Chennai, Tamil Nadu, India

Abstract

The proposed solution introduces SunduVault, which is a cloud-based management framework for managing a WireGuard-based virtual private network with a firewall management panel. End-to-end secure connection to the network is enabled by a PHP/MySQL backend that is responsible for handling user authentication, assigning dynamic IPs, and managing peer onboarding using a REST API. A peer monitoring dashboard along with firewall management capabilities gives live status of connections between clients and the central server, allowing the administrator to apply access control policies without requiring shell-level access. Secure access to the application is ensured by session-based authentication coupled with automatic token refresh, peer metadata storage in MongoDB, and separation of roles among admin and user portals. Experimental results show successful integration of peer onboarding and network segmentation using firewall rules along with real-time peer monitoring dashboard running at under one second refresh intervals.

Keywords: wireguard, vpn management, iptables, firewall, network security, peer monitoring, php, mongodb

1. Introduction

The increasing popularity of remote working and distributed computing has led to a need for secure and managed Virtual Private Network (VPN) systems. Although current traditional implementations of VPNs are functional, their major drawbacks include difficult configuration, poor observability, and cumbersome operations. WireGuard is a relatively recent addition to existing implementations. Proposed by Donenfeld (2017), WireGuard boasts an efficient architecture with simple code, kernel-level performance and improved throughput in comparison to traditional protocols like OpenVPN and IPsec.

A large-scale administration of WireGuard involves not only its protocol features but also tools for managing peers, distributing dynamic IPs, enforcing firewall policies, and monitoring active connections. While there exist several commercial products addressing these concerns,

Published: 02 June 2026
DOI: https://doi.org/10.70558/IJST.2026.v3.i2.241270

some of which provide open-source alternatives, they are either expensive or not fully compatible with host-level firewall software.

In this paper, we present SunduVault – a web application for WireGuard network management. SunduVault consists of a web-based PHP interface, two databases (MySQL and MongoDB), RESTful API for peer management, firewall management interface using iptables, and a WireGuard peer monitoring interface. Furthermore, SunduVault features two separate portal interfaces (admin portal and user portal), strict session-based authentication with automatic JWT refreshment, and WireGuard peer management automation.

Organization of the rest of the paper is as follows. In Section 2 we give an overview of the related literature. System architecture is discussed in Section 3. Section 4 focuses on implementation. Experimental results are reported in Section 5. We conclude our paper in Section 6.

2. Literature Review

Wireguard was defined by Donenfeld (2017) as a high-performance, low-configuration VPN solution based on the Noise protocol framework, key exchange using the Curve25519 algorithm, authenticated encryption using ChaCha20-Poly1305 and hashing using BLAKE2s. Various empirical analyses of network performance have confirmed the speed benefits of the technology compared to OpenVPN and IPsec (Osswald et al., 2020).

The development of web-based management interfaces for various types of virtual private networks has received attention in multiple works. For example, the Wireguard-UI software created by Emiro (2021) is a simple peer management tool that does not feature firewall integration. PiVPN is an automation software designed to install Wireguard automatically; it does not provide an API for management purposes. Tailscale and ZeroTier are commercially available tools built around proprietary overlay networks.

Wool (2004) identified the problems associated with complex firewall rulesets used in enterprises and the consequent risks of configuration errors in the context of enterprise security. Some firewall management solutions, including WebMin, provide some administrative functionality but do not include integration with the peer management interface. The current study seeks to solve both issues simultaneously.

A lot of research has been carried out with respect to session management and authentication within web applications. Best practices advise using short-living JWT access tokens with refresh token rotation (RFC 6749), server session invalidation, and HTTP-only cookies for session management (OWASP, 2021). These guidelines are implemented in the PHP session management module within SunduVault.

Traditional approaches to real-time network monitoring include SNMP polling or dedicated software agents. Recent developments in web technology have facilitated creation of lightweight, push-driven dashboards. SunduVault uses periodic client-side polling of lightweight PHP endpoints that call the wg show command provided by WireGuard to enable peer visibility.

3. System Architecture

SunduVault follows a three-tier architecture comprising a presentation layer, an application layer, and a data layer.

Figure 1: SunduVault High-Level Component Relationships

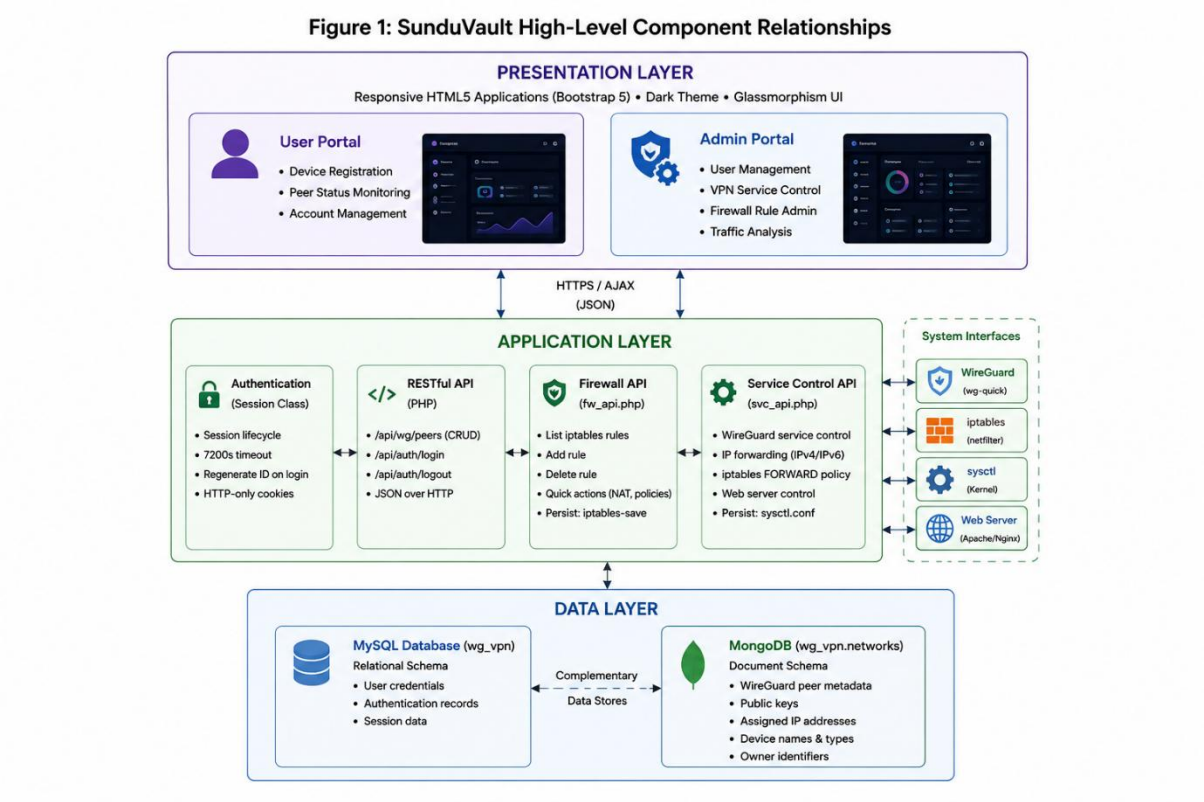

Figure 1: illustrates the high-level component relationships.

3.1 Presentation Layer

There are two portals in the presentation layer: one portal for user actions like registering devices, tracking peer status, and managing accounts; and the other for administering the user accounts, managing the firewall rules, controlling the VPN services, and analyzing the traffic. The two portals have been designed as HTML5 responsive applications with Bootstrap 5 with all communications happening through asynchronous JavaScript calls to the application layer.

3.2 Application Layer

The application layer comprises PHP 8 code organized into user and admin modules. Authentication is provided by the Session class which controls the PHP session management, implements a 7200-second timeout limit, forces session ID regeneration at logon time, and implements HTTP-only attributes for cookies. The accompanying RESTful API built as a standalone PHP service listening on an internal IP implements the /api/wg/ endpoints for Wireguard peer creation, listing, and deletion and the /api/auth/ endpoints for authentication.

The Firewall API module (fw_api.php) provides functions for displaying the current list of iptables entries with packet and byte counters, creating new entries with the following parameters: chain, protocol, source, port, interface, and target, deleting iptables entries by chain

and rule number, as well as quick commands such as NAT masquerade enabling and default policy setting. All external input data is checked using allowlists and regexes and then passed to iptables. Rules are stored persistently using iptables-save to the /etc/iptables/rules.v4 file.

Service Control API (svc_api.php) offers management options for the WireGuard service (starting, stopping, restarting, enable/disable autostart) as well as for IPv4/IPv6 forwarding (sysctl) and iptables FORWARD chain rules and the host Web server itself. The commands are run using sudo with a narrow privilege set. Any sysctl configuration modifications will be saved to /etc/sysctl.conf.

3.3 Data Layer

The two types of data store used by SunduVault complement each other. User login credentials and authentication information are stored in a MySQL database (wg_vpn) that supports transactions. Peer information for Wireguard is stored using MongoDB and contains public keys, IP address allocations, and device name and type information for each device as well as the username of the owner of the device.

Table 1: Data Store Responsibilities

Data Store	Schema	Primary Use
MySQL (wg_vpn)	Relational	User credentials, auth records, session data
MongoDB (wg_vpn.networks)	Document	WireGuard peer metadata, public keys, assigned IPs

4. Implementation

4.1 User Authentication

The process of user authentication requires submission of a username or email as the identifier for log-in purposes. Login.php connects to MySQL and uses it to determine the canonical email based on the provided identifier and then contacts the in-house authentication service API, which checks if the password is correct and generates an access token, together with a refresh token. Session class saves both tokens along with the information about the authenticated user inside the PHP session. In each request, auth.php middleware checks if there is a session and imposes a timeout period after which it expires.

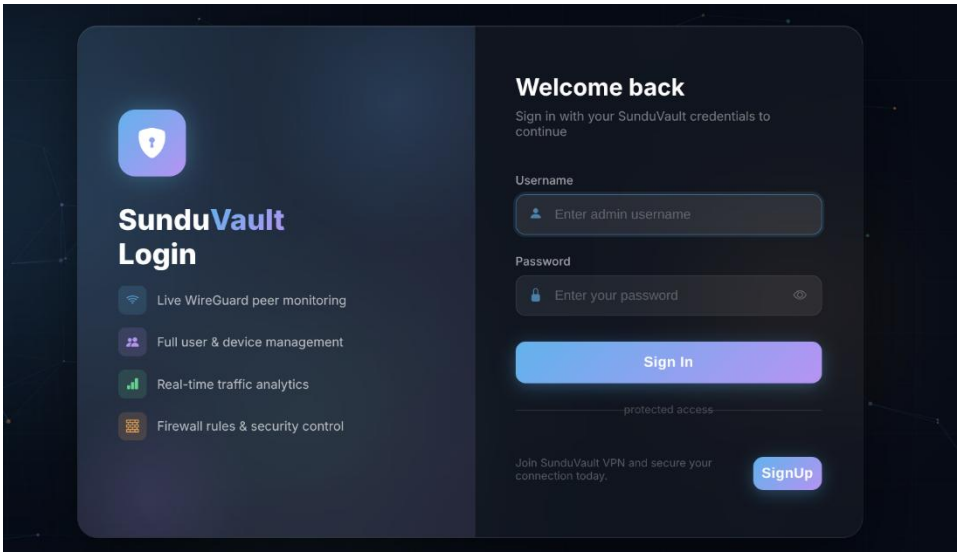

Figure 2: User Authentication

4.2 WireGuard Peer Management

The device registration process is performed via a user portal using a form that takes the following input values: WireGuard public key, device name, device type, and owner ID. The api_add_device.php script verifies all mandatory input parameters and redirects to the server REST API at /api/wg/add_peer as multipart POST. This REST API picks an appropriate free IP address in the subnet 172.20.0.0/16, writes a new peer to the WireGuard configuration, reloads it, and returns back the IP to the user portal where the result is also stored in MongoDB along with other information about the peer. Monitoring peers' status is accomplished by the php script api_peer_status.php. This script makes use of MongoDB to find all registered public keys by current users and then runs the command wg show all dump to receive current peers list provided by the WireGuard kernel module. The output is parsed tab-delimited into records that contain interface name, public key, endpoint address, allowed IPs, latest handshake timestamp, bytes received and bytes sent.

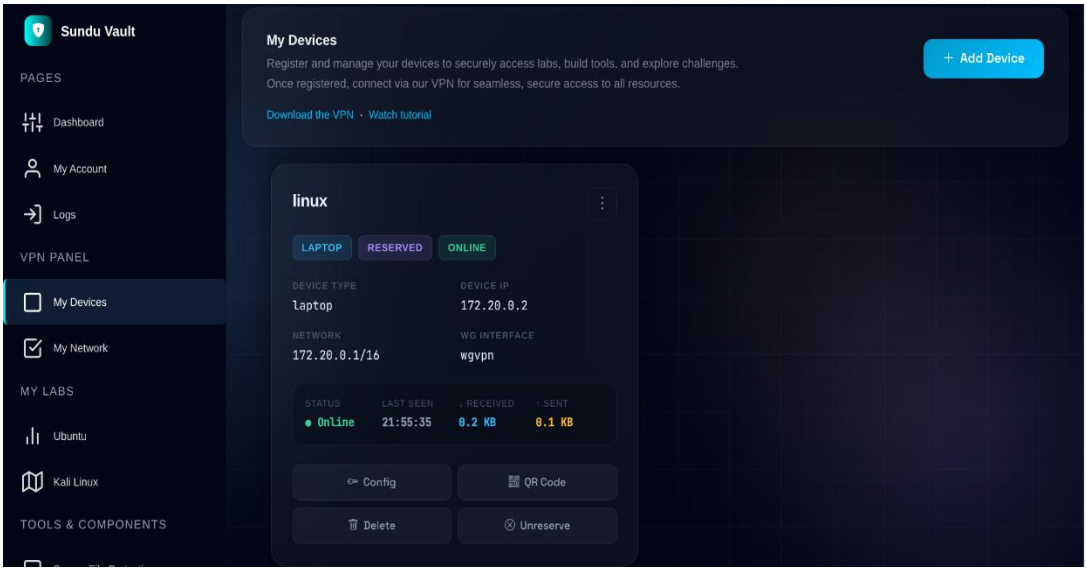

Figure 3: WireGuard Peer Management

4.3 Firewall Management

The interaction between the firewall panel and fw_api.php occurs through AJAX calls. The list action uses iptables -L -n -v --line-numbers, which then uses a regular expression for parsing the result to determine the rule number, packet/byte counts, targets, protocols, interface restrictions, and source/destination IP addresses. The rules are returned to the frontend as a JSON array, which is displayed in a table format on the interface.

When adding new rules, chain validation checks whether they are part of INPUT, OUTPUT, FORWARD, PREROUTING, or POSTROUTING chains. Validation of targets checks whether they are either ACCEPT, DROP, REJECT, LOG, MASQUERADE, or RETURN, while validation of protocols checks whether they are either tcp, udp, all, or icmp. Sources are then checked based on regex for IP addresses, port, and interfaces. Rules are always added at line position 1 using iptables -I command.

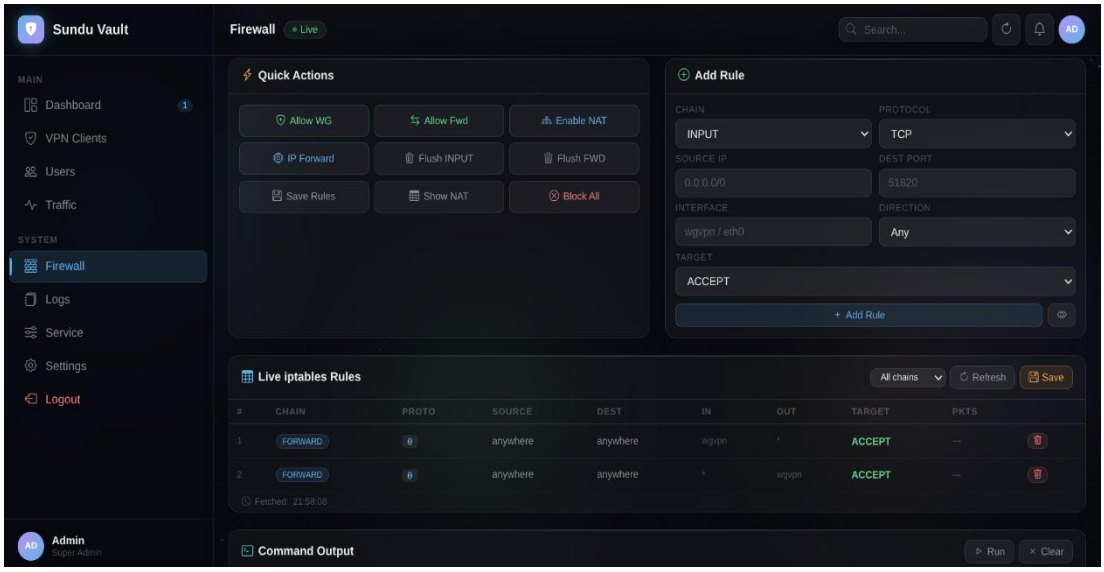

Figure 4: Firewall Management

Table 2: Firewall API Actions

Action	HTTP Method	Description
list	GET	Return all iptables rules with counters
add	POST	Insert a new rule at chain position 1
delete	POST	Delete rule by chain and rule number
quick	POST	Execute predefined quick actions (e.g., allow-wg, enable-nat)

4.4 Administrative Service Control

The control panel for service allows toggling on/off commands for the WireGuard service, IP forwarding configuration, and web server administration. All commands get executed using

the sh() function, which executes using escapeshellarg() around the exec(). The allowlist pattern for actions makes sure that the commands can only be executed if they match the command strings mapped to the names of each action.

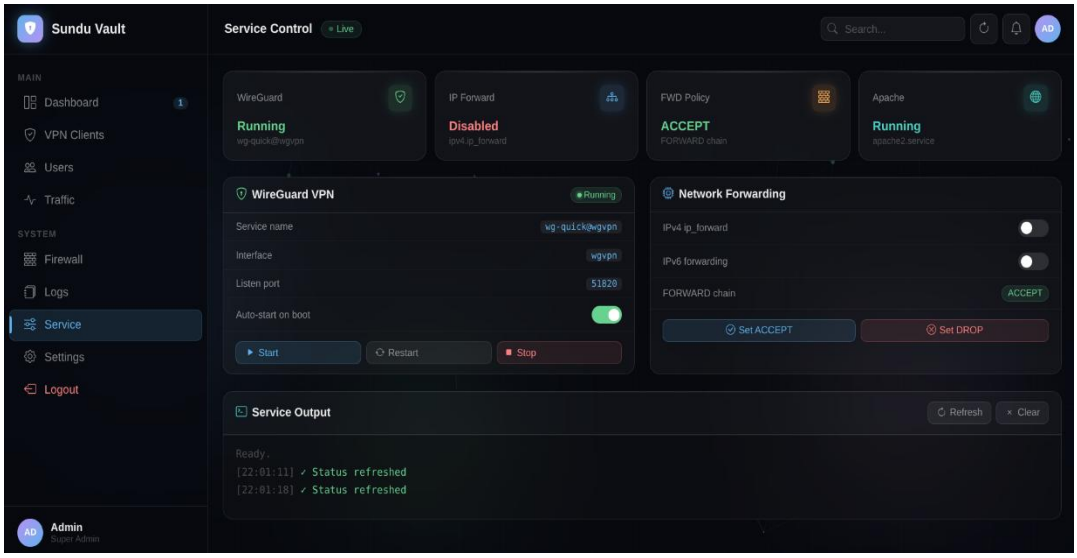

Figure 5: Administrative Service Control

4.5 Security Design Decisions

Key security design decisions implemented across the platform include the following.

Session tokens are renewed once login succeeds, while cookie settings have been configured for HTTP-only and SameSite=Lax attributes in order to enhance the security of sessions. The admin and user portals operate separately without sharing session states; admin authentication is done independently using the `admin/auth.php` component. For effective firewall management, all the parameters of the `iptables` command have been validated against allowlist and regular expressions prior to executing them. Moreover, MongoDB queries have used parameterization in order to protect against NoSQL injections. The internal API will be operated based on an internal hostname (`api.local`), and thus cannot be accessed externally. Furthermore, sudoers have been restricted to the required commands only through the sudoers configuration file.

5. Experimental Evaluation

5.1 Test Environment

The platform was deployed on an Ubuntu 24.04 LTS server with a single WireGuard interface (wgvpn) configured on the 172.20.0.0/16 subnet. The web application was served by Apache 2.4. MySQL 8.0 and MongoDB 6.0 provided the data layer. Client devices included a Windows 11 laptop, an Android smartphone, and a Linux workstation, each configured with the official WireGuard client application.

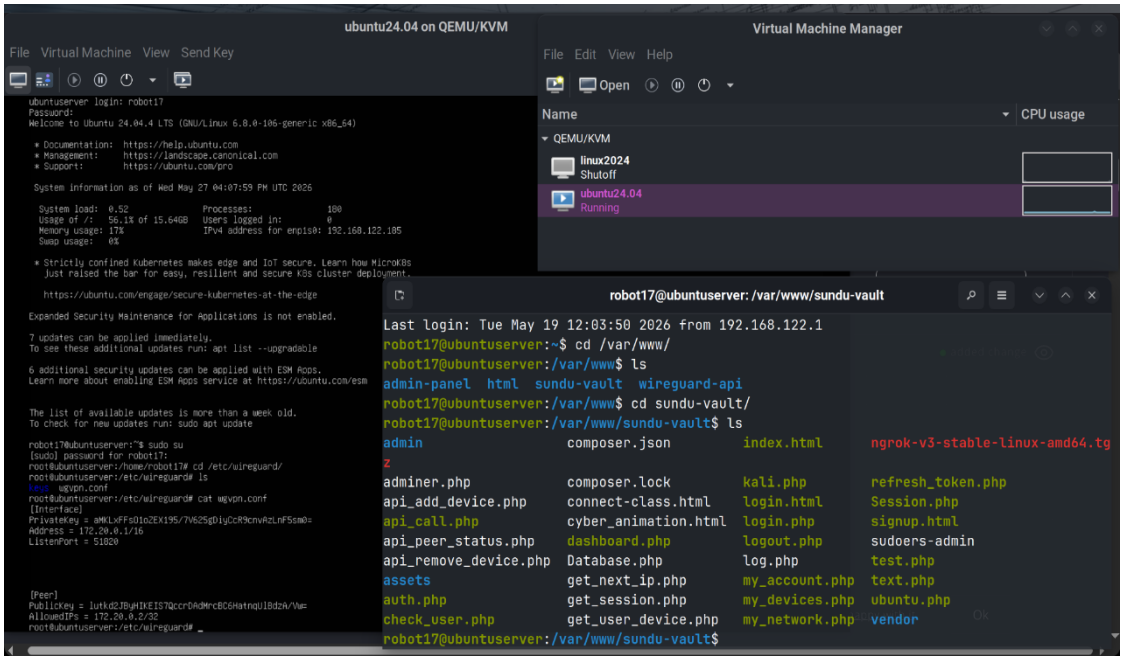

Figure 6: Test Environment for VPN and Firewall

5.2 Functional Testing

Functional testing evaluated the complete peer lifecycle from registration to revocation. User registration, login, and session timeout enforcement were tested with both valid and malformed credentials. Device addition was tested with valid public keys, duplicate public keys (expected rejection), and missing required fields (expected validation error). Peer status monitoring was verified to display only the authenticated user's peers, confirming per-user data isolation.

Firewall rule management was tested by adding and removing rules across all supported chains and target actions, verifying that rules persisted across service restarts via iptables-save. Quick actions were tested for correctness of resulting iptables state. Service control actions were tested for WireGuard start, stop, restart, and autostart toggle, with verification of resulting systemctl status.

Table 3: Functional Test Results Summary

Test Case	**Expected Outcome**	**Result**
Valid user login	JWT tokens issued, session created	Pass
Invalid password login	Error response, no session	Pass
Device registration (valid key)	Peer added, IP assigned	Pass
Device registration (duplicate key)	Error: duplicate public key	Pass
Peer status — own devices visible	Own peers returned	Pass

Peer status — other user devices hidden	No cross-user data	Pass
iptables rule add (valid)	Rule inserted at chain position 1	Pass
iptables rule add (invalid chain)	400 error: Invalid chain	Pass
iptables rule delete	Rule removed, persisted	Pass
WireGuard service restart	Service restarted successfully	Pass

5.3 Performance Observations

Dashboard refresh cycles averaging approximately 800 milliseconds were observed for peer status updates on a local network, satisfying the sub-second requirement for near-real-time visibility. Firewall rule listing for a ruleset of 30 rules completed in under 200 milliseconds. Peer registration end-to-end latency, including API forwarding, MongoDB write, and WireGuard configuration reload, averaged approximately 1.2 seconds, acceptable for an infrequent administrative operation.

6. Discussion

6.1 Strengths

SunduVault demonstrates that a full-featured VPN management platform can be implemented with widely available open-source components. The separation of user and administrative portals reduces the attack surface available to authenticated end users. The use of an internal API boundary between the web application and the WireGuard control plane limits the commands that can be triggered through the web interface. The dual-store architecture leverages the complementary strengths of relational and document databases for different aspects of the data model.

6.2 Limitations and Future Work

Several limitations warrant acknowledgement. The current implementation stores database credentials in plaintext within source files (Database.php), a practice that should be replaced with environment variable injection or a secrets management solution prior to production deployment. The session cookie secure attribute is currently set to false; production deployments must enforce TLS and enable this attribute.

Future work should address the following areas. Integration of a machine learning-based intrusion detection system on VPN traffic flows would extend the security posture of the platform. Implementation of the Zero Trust Network Access (ZTNA) model, requiring continuous device posture verification beyond initial authentication, would strengthen access control. Post-quantum cryptographic primitives for WireGuard key exchange, as explored in recent literature, would future-proof the platform against quantum-capable adversaries. A

mobile application providing push notifications for peer connection events and firewall alerts would improve operational responsiveness.

7. Conclusion

This paper presented SunduVault, a self-hosted WireGuard VPN management platform integrating peer lifecycle management, iptables firewall control, and real-time connection monitoring within a unified web interface. The system architecture separates user and administrative concerns, enforces session-based authentication with token refresh, and validates all command parameters to mitigate injection risks. Functional testing confirmed correct operation across the complete peer management lifecycle and firewall administration workflow.

SunduVault contributes a practical, transparent alternative to proprietary VPN management solutions, suitable for academic institutions, small enterprises, and network security researchers. The platform demonstrates that modern web technologies can provide effective, observable control over kernel-level network security primitives without requiring direct shell access. Future enhancements incorporating AI-driven anomaly detection and Zero Trust access control mechanisms will further strengthen the platform's security profile.

References

Donenfeld, J. A. (2017). WireGuard: Next generation kernel network tunnel. In Proceedings of the Network and Distributed System Security Symposium (NDSS). Internet Society. https://doi.org/10.14722/ndss.2017.23160

Emiro, N. (2021). Wireguard-UI: A web user interface for WireGuard VPN [Software]. GitHub. https://github.com/ngoduykhanh/wireguard-ui

Open Web Application Security Project (OWASP). (2021). Session management cheat sheet. OWASP Foundation. https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

Osswald, L., Shannigrahi, S., & Bhatt, S. (2020). Performance evaluation of WireGuard, OpenVPN, and IPsec. International Journal of Network Management, 30(5), e2091. https://doi.org/10.1002/nem.2091

Hardt, D. (Ed.). (2012). The OAuth 2.0 authorization framework (RFC 6749). Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/rfc6749

Wool, A. (2004). A quantitative study of firewall configuration errors. IEEE Computer, 37(6), 62–67. https://doi.org/10.1109/MC.2004.2