Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.1, February 2026, pp. 889~899
e-ISSN: 2548-6861

Implementing Defense-in-Depth Framework on Orange Pi NAS Using Host-Based Security and ZFS

Muhammad Fatih Hady [1]*, Hafiyyan Putra Pratama [2]*
* Department of Telecommunication Systems, Universitas Pendidikan Indonesia
muhammadfatihhady@upi.edu [1], hafiyyan@upi.edu [2]

Article Info

Article history:

Received 2025-11-19
Revised 2026-01-30
Accepted 2026-02-09

Keyword:

Defense-in-Depth,
NAS,
Orange Pi,
Debian.

ABSTRACT

Network-Attached Storage (NAS) based on low-cost Single Board Computers (SBC) offers an affordable alternative to commercial storage systems, yet its exposure to network-based threats requires a robust and layered security approach. This research implements the Defense-in-Depth (DiD) framework on an Orange Pi based NAS running Debian 12, integrating host-based security mechanisms and the ZFS file system to enhance data integrity, availability, and system resilience. The security layers include firewall restrictions, intrusion prevention with Fail2Ban, integrity monitoring using AIDE and rkhunter, system auditing with Lynis, and log analysis with Logwatch. Additionally, ZFS snapshots and the Sanoid retention policy are applied to provide rapid data recovery with minimal storage overhead. Experimental results show that all defense layers function effectively under testing scenarios such as brute-force attempts, unauthorized port access, file modification, and data deletion. ZFS snapshots successfully restore deleted or altered files, ensuring minimal Recovery Point Objective (RPO) of one hour. System performance remained stable, with CPU usage averaging only 7.9% and memory usage at 33%, indicating that the DiD model is feasible even on low-resource SBC hardware. These findings demonstrate that a cost-efficient SBC-based NAS can achieve strong resilience against common cyber threats through layered security design and modern file system capabilities.

This is an open access article under the CC–BY-SA license.

I. INTRODUCTION

Network-Attached Storage (NAS) has become an important component in modern IT infrastructure to meet the needs for centralized data storage and reliable file sharing [1]. Technological developments have enabled the implementation of NAS using low-cost hardware such as Single Board Computers (SBC) like Orange Pi, running on the Debian Linux operating system. This solution offers significant cost efficiency compared to commercial solutions, making it an attractive choice for SMEs and home users [2]. Nevertheless, a NAS system connected to the network is vulnerable to various cyber threats, ranging from Denial of Service (DoS) attacks aimed at disabling services to brute force attempts CC BY SA This is an open access article under the CC-BY-SA license. to gain unauthorized access, especially through remote services like SSH [3], [4].

To address this complexity of threats, the security strategy used must be layered, which is known as the Defense-in-Depth (DiD) model [5]. The DiD model is based on the principle that if one layer of defense fails, the next layer will prevent the attack from succeeding completely [6]. The DiD strategy is continuously being expanded in cybersecurity literature to face various threat vectors [7]. In this research, DiD is applied through two pillars: security at the operating system level (host-based security) and security at the data storage level. At the host layer, hardening steps are implemented which include limiting network access using a firewall to minimize the attack surface [4]. Additionally, Fail2Ban is implemented as an Intrusion Prevention System (IPS) to analyze logs system in real-time and block attackers who try to perform brute force [8], [9], [10]. This protection is complemented by system audit and integrity monitoring tools such as AIDE, rkhunter, Lynis, and Logwatch, which

function to identify vulnerabilities and unauthorized file changes [11], [12].

The second layer of defense is securing the data itself, which is achieved through the utilization of the advanced ZFS file system. The advantage of ZFS in this context is its ability to manage data integrity and availability, especially through the snapshot feature [1]. ZFS snapshot allows the NAS server to regularly create copies of data at specific times, which becomes crucial for quick recovery from data incidents, including human error or system damage. By integrating network access controls, host intrusion prevention, system integrity, and robust data recovery capabilities through ZFS snapshots, this research aims to demonstrate the implementation of an efficient SBC-based NAS system that has an adequate level of resilience against current cyber threats.

Although there have been various studies regarding the implementation of NAS based on low-cost hardware and studies on the application of layered security mechanisms, most of these studies still focus on performance aspects or only on one specific security layer [2], [13]. Studies that explicitly integrate the DiD strategy with the utilization of modern file systems like ZFS on Single Board Computer (SBC) platforms are still limited. This creates a need for research that not only highlights the performance of SBC-based NAS, but also its security resilience against increasingly complex cyber threats.

Based on these conditions, this research offers a contribution by designing and implementing a NAS model based on Orange Pi and the Debian operating system equipped with host-based security tools and the snapshot feature from ZFS [14]. The main contribution of this research is to provide a layered security architecture design that is cost-effective yet robust, while also demonstrating how the integration of SBC, host hardening, and the ZFS file system can improve resilience, integrity, and data availability. Thus, this research is expected to become a practical and academic reference in the development of NAS solutions that are secure, cost-effective, and reliable.

II. METHODS

This research uses an experimental approach with design, implementation, and testing stages. The main objective of this research is to implement the DiD model on a NAS system based on a Single Board Computer (SBC) Orange Pi with the Debian operating system, in order to improve data security, integrity, and availability.

All implementations were carried out in an environment that fully uses Wireless-Fidelity (Wi-Fi) via a smartphone hotspot. The NAS system can be accessed from the internet using the ZeroTier One platform, which provides a virtual public IP address for encrypted communication between devices without port forwarding or a static IP.

To provide a general overview of the research stages, Figure 1 shows the research method flowchart for the DiD-based NAS. This flowchart shows the sequence of the research process starting from system design, implementation of security layers, effectiveness testing, to results analysis.

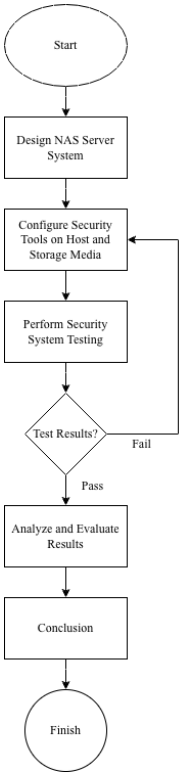

Figure 1. Experiment flow flowchart

A. NAS System Design

The design stage is carried out to determine the system architecture design, the devices used, and the testing environment. The NAS system is designed by implementing two main security layers, namely host security and ZFS snapshot-based storage security, in accordance with the DiD concept in Figure 2.

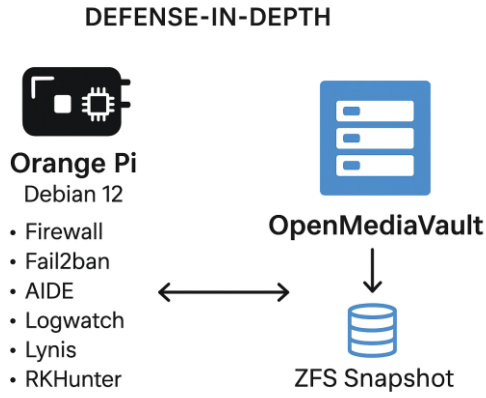

Figure 2. Defense-in-Depth Model

1) *System Architecture Design:* The NAS system consists of several main components as follows: First, the NAS server (Orange Pi) which runs the Debian 12 operating system and functions as the data storage center. Second, the client/administrator who accesses the NAS via the internet using the SSH protocol. Third, the host security layer, including firewall, Fail2Ban, AIDE, rkhunter, Lynis, and Logwatch to prevent, detect, and monitor risky activities. Fourth, the storage security layer, using the snapshot feature from ZFS to support rapid data recovery.

2) *Specifications of Devices Used:* The hardware and software used in the research are shown in TABLE I.

TABLE I
TESTING DEVICE SPECIFICATIONS

Component	Specification
SBC Model	Orange Pi Zero 3
Processor	Allwinner H618 *Quad-core Cortex*-A53 1.5GHz
RAM	1GB LPDDR4
Storage	microSD 32 GB (OS) + external HDD 250 GB (data)
Operating System	Debian 12 (Bookworm) arm

The entire NAS implementation was carried out using Wi-Fi from a smartphone hotspot without a physical Local Area Network (LAN) connection. The NAS system can be accessed from the internet securely via the ZeroTier One platform, which functions to create a Virtual Private Network (VPN) and provide a virtual public IP address for each node. Thus, the NAS system can be accessed from outside the local network without needing port forwarding or a static IP. ZeroTier One installation is done with the following command:

```
curl -s https://install.zerotier.com | sudo bash
sudo zerotier-cli join <Network_ID>
```

After connecting to the ZeroTier network, the NAS will obtain a virtual public IP address that can be used for remote administrative access via SSH and the OpenMediaVault web interface.

B. Host Security Layer Design

The host security layer aims to protect the system operating and NAS services from network attacks, changes unauthorized files, and suspicious activities. Some tools used are described as follows.

1) *Firewall:* A firewall is software or hardware designed to protect networks, computer systems, or other devices from threats and attacks originating from untrusted networks, such as the Internet [15]. A firewall creates a barrier between a trusted network and an untrusted network. Firewalls can be categorized as network-based or host-based [4]. The following is how to install and configure a host-based firewall:

```
sudo apt install ufw
sudo ufw default allow incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow 'samba'
sudo ufw enable
```

2) *Fail2Ban:* Fail2Ban is open- source software built using the Python programming language [16]. Fail2Ban is an Intrusion Prevention System (IPS) tool designed to protect servers by blocking SSH access and automatically blocking IP addresses on devices that fail to log in repeatedly [8], [17]. The following is how to install Fail2Ban:

```
sudo apt install fail2Ban
```

3) *AIDE (Advanced Intrusion Detection Environment):* AIDE is an open-source utility for checking the integrity of files and directories. AIDE is the successor to the open-source Tripwire project which functions as a change detection system for files and directories, with the main goal of monitoring system integrity and detecting suspicious modifications due to unauthorized activities or security attacks [18]. The following is how to install and initialize AIDE:

```
sudo apt install aide
sudo apt-get install aide aide-common
sudo aide --config /etc/aide/aide.conf --init
sudo mv /var/lib/aide/aide.db.new /var/lib/aide/aide.db
```

4) *rkhunter (Rootkit Hunter):* rkhunter is a detection tool used to find rootkits on Linux systems. rkhunter performs integrity checks to detect rootkits. Like AIDE, rkhunter requires a system snapshot (image) that must be created proactively before the system becomes infected [18]. The following is how to install rkhunter:

```
sudo apt install rkhunter
```

5) *Lynis:* Lynis is a tool used for general security auditing in Linux environments. Lynis can be used as part of a security audit framework to support the detection and mitigation of Privilege Escalation (PE) vulnerabilities in Linux systems [19]. The following is how to install Lynis:

```
sudo apt install lynis
```

6) *Logwatch:* Logwatch is a log monitoring and analysis tool. Logwatch functions to check log files system, converting them into an understandable format, and creating detailed reports. This tool simplifies log review without needing to access each file manually [17]. The following is how to install and use Logwatch:

```
sudo apt install logwatch
```

C. ZFS Integration with OpenMediaVault (OMV)

The storage security layer uses OpenMediaVault (OMV) as a web-based NAS management interface. OpenMediaVault provides built-in features for managing volumes, file sharing, and integrating the ZFS file system. OpenMediaVault installation is done via the following automatic command:

```
wget -O - https://github.com/OpenMediaVault-Plugin-Developers/installScript/raw/master/install | sudo bash
```

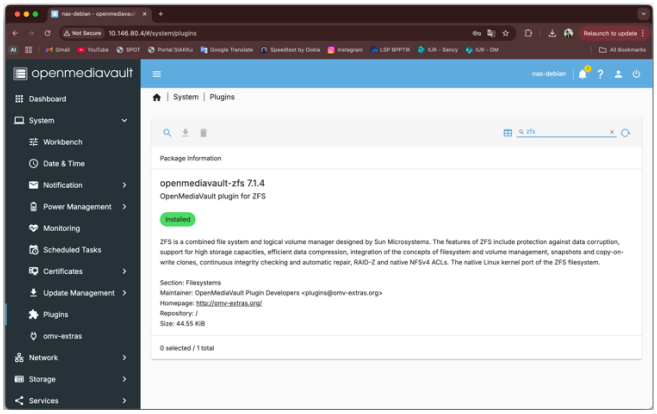

Figure 3. ZFS snapshot installation

After the installation is complete, file system management is done through the OMV web interface. In the OMV menu, go to System then select Plugins. ZFS is installed by adding the openmediavault-zfs 7.1.4 plugin as shown in Figure 3. Through this interface, users can create ZFS pools, datasets, and activate automatic snapshot features for data protection and recovery. In this test, the following configuration was used:

1) *Snapshot Parameters:* The ZFS Copy-on-Write (CoW) mechanism was utilized to create Read-Only data copies. These snapshots preserve the state of the data at a specific point in time without imposing significant storage overhead.

2) *Snapshot Frequency:* The system was configured to automatically generate snapshots on an hourly basis (every 60 minutes). This parameter establishes a maximum Recovery Point Objective (RPO) of one hour, which is significantly lower than conventional daily backup methods. The hourly frequency is created automatically when the plugin is installed, without requiring manual configuration.

3) *Retention Policy:* To maintain resource sustainability on the Orange Pi, a tiered retention policy was applied, consisting of Hourly, Daily, and Monthly rules. The following script illustrates the configuration of the retention policy using Sanoid:

```
[datapool/media]
   use_template = production
   recursive = yes

[template_production]
   hourly = 24
   daily = 30
   monthly = 3
   yearly = 0
   autosnap = yes
   autoprune = yes
```

This configuration script is a policy-driven blueprint for the Sanoid tool, designed to automate the data resilience layer of Debian-based NAS project. It begins by targeting the primary ZFS dataset, [datapool/media], which is the core storage area for the system. By setting recursive = yes, the script ensures that any sub-folders or child datasets created within this path are automatically covered by the same security protocols, providing uniform protection across the entire storage hierarchy. This section links the dataset to the production template, establishing a structured set of rules for how data should be preserved and managed over time.

The second part of the script, defined under [template_production], implements a tiered retention strategy that is central to DiD security model. By specifying hourly = 24, daily = 30, and monthly = 3, the system maintains a granular historical record of your data: 24 snapshots for the most recent day, 30 for the month, and 3 for the quarter. This tiered approach effectively minimizes the Recovery Point Objective (RPO) to just 60 minutes for recent file changes, while still providing long-term protection against threats like ransomware that might not be detected immediately. Because these snapshots are read-only, they serve as an immutable last line of defense if network security tools are bypassed.

D. System Testing

Testing was conducted to assess the effectiveness of the NAS system in implementing the DiD model. The testing scenarios for each tool are shown in Table II.

TABLE II
SYSTEM TESTING SCENARIOS

No	Tools	Testing Scenario	Action Performed	Result Obtained
1	Firewall	Unauthorized port access test	Attempt connection to a port other than SSH	Connection rejected
2	Fail2Ban	SSH brute force simulation	Perform repeated failed logins	SSH access automatically blocked
3	AIDE	System file change detection	Change content of a dummy configuration file	AIDE reports the change
4	rkhunter	Scan for rootkits & backdoors	Run a full check	No rootkits found
5	Lynis	System configuration audit	Run security audit	Security score increases after DiD implementation
6	Logwatch	Daily activity monitoring	Analyze SSH & system logs	Activity report generated
7	ZFS Snapshot	File deletion	Delete file from dataset	Snapshot rollback restores the file
		File content modification	Change text file content	Snapshot rollback restores original content
		Snapshot efficiency	Create snapshots repeatedly	Snapshot is efficient in storage space

III. RESULTS AND DISCUSSIONS

This section describes the implementation results and evaluation of the layered security system on the Debian-based NAS run on the Orange Pi Zero 3 device. Testing was conducted on all host-based security tools and the snapshot feature on ZFS to assess the effectiveness of the DiD approach in maintaining the integrity, availability, and security of the NAS system.

A. NAS System Implementation Results

The NAS system was successfully implemented and run using a Wi-Fi connection from a smartphone hotspot, with remote access via ZeroTier One. This connection allows the administrator to access the NAS from outside the local network securely without additional configuration on the router. The NAS is capable of running SSH-based services, ZFS-based shared storage, and all host security layers that have been configured.

During implementation, system resource usage (CPU and memory) remained efficient. Based on monitoring via the dashboard in OpenMediaVault, the average CPU utilization was recorded at 7.9%, staying consistently below the 10% threshold even when all security tools were active. Furthermore, memory usage remained stable at approximately 33% (323.1 MiB) of the total 981.86 MiB capacity, leaving 67% of resources free for other operations, as shown in Figure 4. This data indicates that the implementation of layered security did not place a significant burden on the Orange Pi's performance, confirming the sustainability of the Defense-in-Depth model on low-resource hardware.

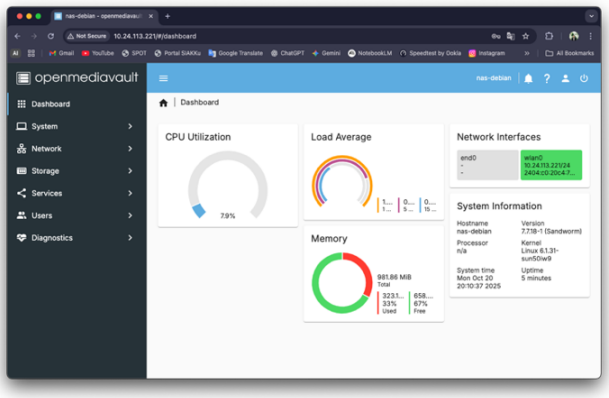

Figure 4. System performance overhead after DiD implementation

B. Host Security Layer Testing Results

Testing was conducted based on the scenarios in Table 2. Each tool was tested separately to assess its function and effectiveness, then evaluated as a whole as a unified layered defense system.

1) *Firewall:* The firewall successfully limited access only to permitted ports. When testing was conducted by attempting to access a random port (in this case port 80), the connection was always rejected as shown in Figure 5. Only port 22 (SSH) was open as configured. This shows that the system's attack surface can be effectively minimized. Log checking using the *sudo grep '80' /var/log/ufw.log* command showed packet denials from port 80 as shown in Figure 6, indicating the system was able to recognize and reject unauthorized connections.

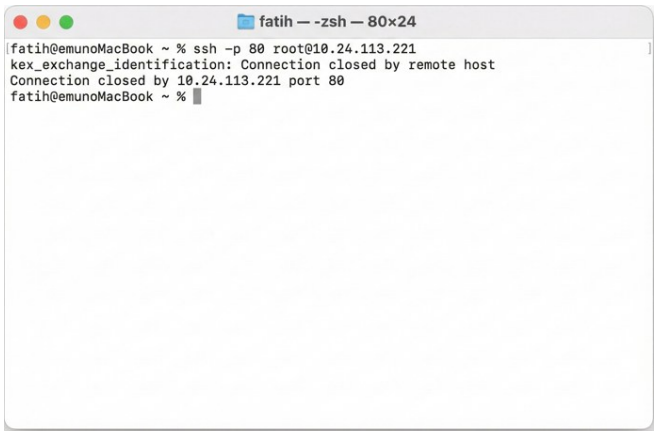

Figure 5. SSH access via port 80

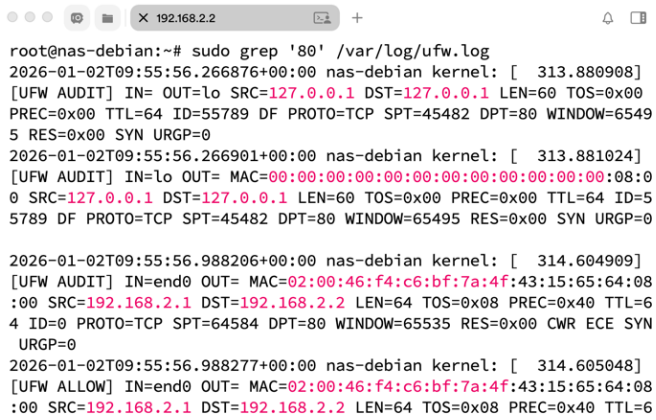

Figure 6. Packet denial from IP address outside the network

2) *Fail2Ban:* When a brute force attack simulation was carried out against SSH with failed login attempts more than five times in a short period, Fail2Ban automatically blocked the attacker's IP address as shown in Figure 7. The block status can be verified via the *sudo Fail2Ban-client status sshd* command as shown in Figure 8. The results show that the tester's IP was detected and blocked for 10 minutes according to the default configuration. This proves the effectiveness of the Intrusion Prevention System in preventing repeated attacks.

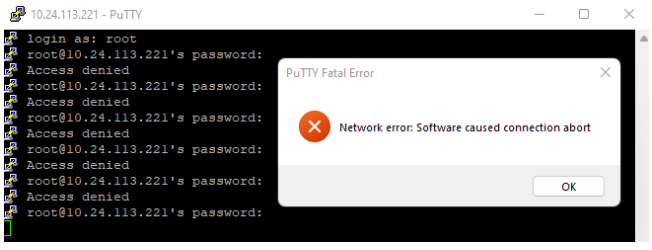

Figure 7. Fail2Ban blocking the connection

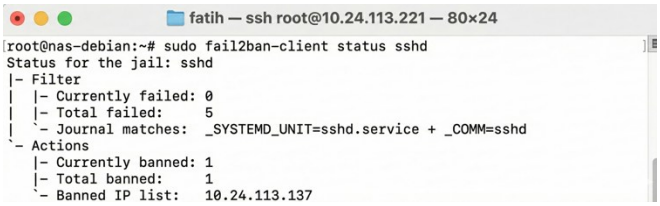

Figure 8. Tester's IP address detected and blocked by Fail2Ban

3) *AIDE:* After the AIDE integrity database was created using *aideinit*, testing was done by creating an empty decoy file in the /etc directory. When the *sudo aide --config /etc/aide/aide.conf --check* command was run, the system displayed a report of the file addition. In addition, for files that were newly deleted and files whose contents were modified will also be visible in the AIDE system. The AIDE check results can be seen in Figure 9. These findings show that AIDE functions effectively in detecting unauthorized system file changes, so it can be used for post-incident forensics.

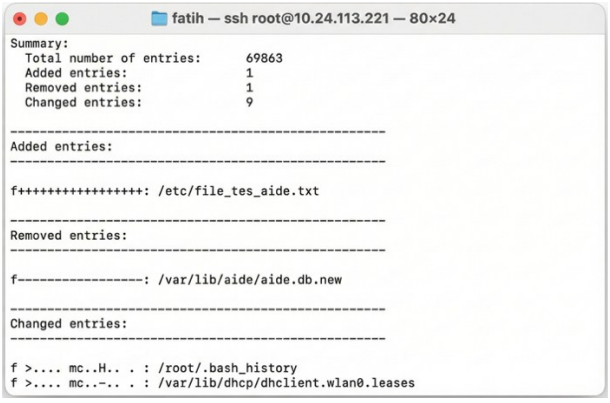

Figure 9. AIDE check results

4) *rkhunter:* A full check using *sudo rkhunter --check --sk* showed a result of possible rootkits: 0, as shown in Figure 10. All system file hashes matched the initial database. This function is important to ensure that no kernel-level malware is hidden within the system.

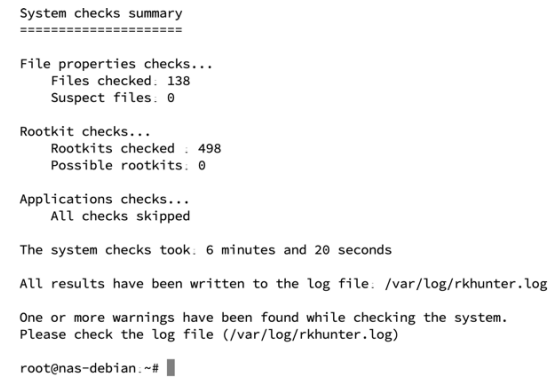

Figure 10. rkhunter check results

5) *Lynis:* The security audit results using Lynis showed using *sudo lynis audit system* showed a system score of 61/100 in the initial check before the security system was implemented as shown in Figure 11, which then increased to 68/100 after the DiD security system was implemented as shown in Figure 12. This score increase shows that Lynis is effective in assisting the system hardening process by providing a security score index.

```
Lynis security scan details:

Hardening index : 61 [############        ]
Tests performed : 248
Plugins enabled : 1

Components:
- Firewall              [V]
- Malware scanner       [X]

Scan mode:
Normal [V]  Forensics [ ]  Integration [ ]  Pentest [ ]

Lynis modules:
- Compliance status     [?]
- Security audit        [V]
- Vulnerability scan    [V]

Files:
- Test and debug information      : /var/log/lynis.log
- Report data                     : /var/log/lynis-report.dat
```

Figure 11. Lynis audit results before DiD implementation

```
Lynis security scan details:

Hardening index : 68 [##############      ]
Tests performed : 268
Plugins enabled : 1

Components:
- Firewall              [V]
- Malware scanner       [V]

Scan mode:
Normal [V]  Forensics [ ]  Integration [ ]  Pentest [ ]

Lynis modules:
- Compliance status     [?]
- Security audit        [V]
- Vulnerability scan    [V]

Files:
- Test and debug information      : /var/log/lynis.log
```

Figure 12. Lynis audit results after DiD implementation

6) *Logwatch:* The daily reports generated by Logwatch show a summary of NAS system activities, including security activities, web service status, storage space usage, package installation and update processes as shown in Figure 13. Based on the monitoring results, the NAS system is in a stable and secure condition, with all services functioning normally without critical errors. Logwatch proved effective as a monitoring tool that provides a comprehensive overview of system activity and health.

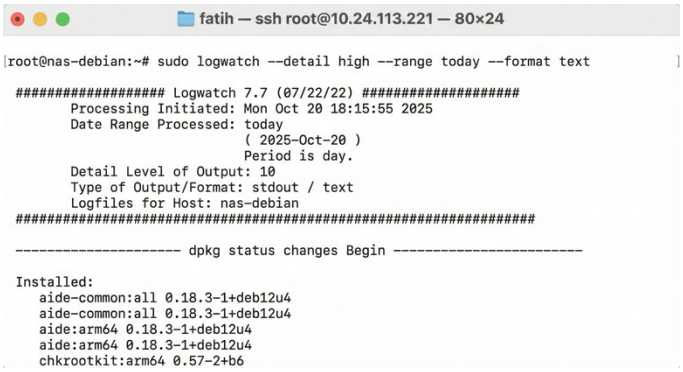

Figure 13. Logwatch monitoring results

After all host security layers were tested and functioning well, the next stage was testing the storage layer using the ZFS snapshot feature.

C. ZFS Snapshot Testing Results

Before testing the snapshot feature, the NAS system was first accessed using Finder on macOS via the *smb://<ip_nas>* address connected through the ZeroTier One network as seen in Figure 14. This way, the dataset directory on ZFS can be accessed directly from the user interface as shown in Figure 15 for the purposes of moving, deleting, or modifying files during the testing process.

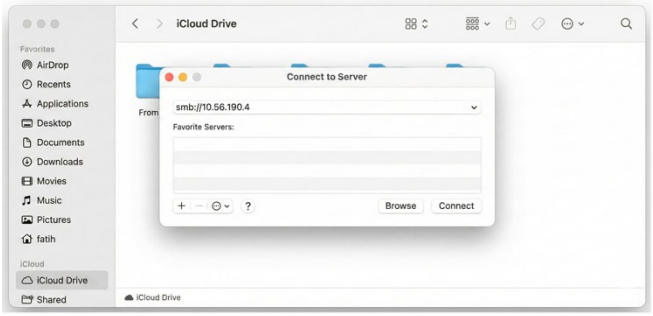

Figure 14. Accessing NAS via the Finder application

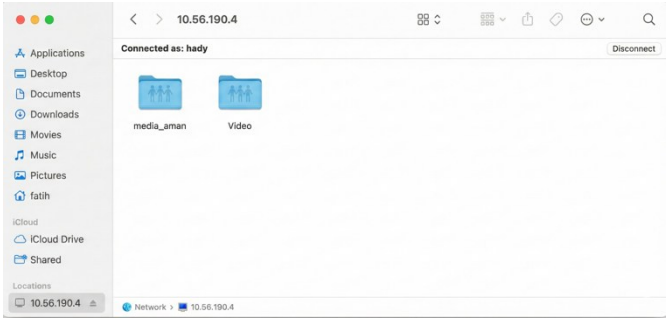

Figure 15. Display of data stored on the NAS

The ZFS snapshot feature test was conducted by creating the *datapool/media@auto* dataset and generating automatic snapshots via a cron job every 1 hour.

1) *File Deletion Test:* 3 files were deleted from the dataset directory as shown in Figure 16, then recovery was performed using the *sudo zfs rollback datapool/media@auto-20251013-110001 -r* command which refers to the last zfs snapshot before the files were deleted. The results showed the 3 files were successfully restored with identical content as before they were deleted as shown in Figure 17.

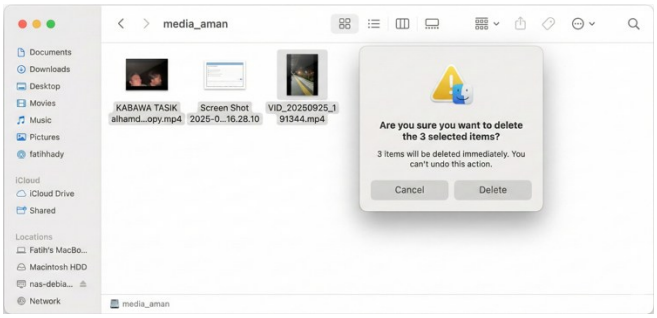

Figure 16. Deleting files from the directory

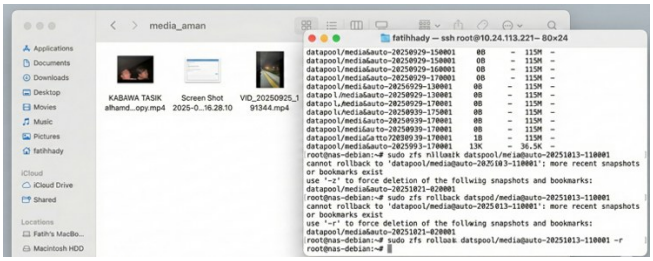

Figure 17. ZFS snapshot restores deleted files

2) *File Modification Test:* The content of the test.txt file was changed as shown in Figure 18, then a rollback was performed using the same command as the file deletion test. After the recovery process, the file content returned to the original version as shown in Figure 19. This shows that ZFS snapshots are effective in handling user errors or unwanted changes without requiring an external backup system.

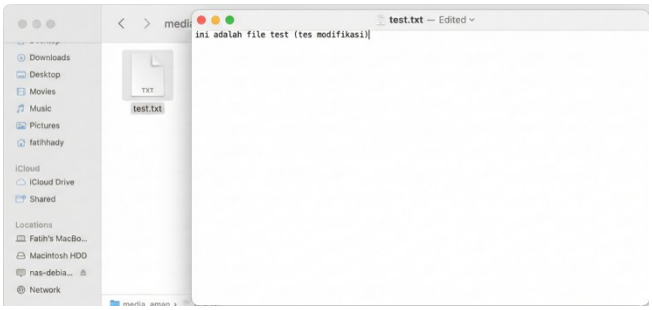

Figure 18. Modifying a file from the directory

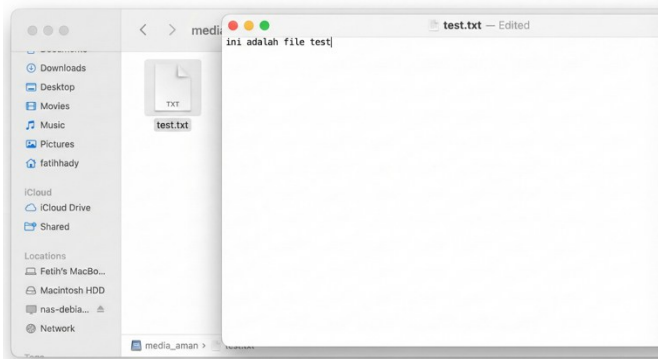

Figure 19. ZFS snapshot restores the file to its original state

3) *Storage Efficiency Test:* Storage efficiency on the ZFS system was tested by comparing the USED and REFER values of the dataset using the *zfs list -o name, used, referenced* command as shown in Figure 20.

```
root@nas-debian:~# zfs list -o name,used,referenced
NAME              USED  REFER
datapool          115M   24K
datapool/media    115M  115M
```

Figure 20. Comparison of USED and REFER values on ZFS

The USED value indicates the total space used, while REFER indicates the size of active data being referenced. Efficiency can be calculated with the formula:

$$O_{\%} = \frac{USED - REFER}{REFER} \times 100\%$$

Based on the results of the *zfs list -o name,used,referenced* command, it was found that the datapool/media dataset has a USED value = 115 MB and REFER = 115 MB. Thus:

$$O_{\%} = \frac{115 - 115}{115} \times 100\% = 0\%$$

This result shows that there is no increase in storage space even though 115 MB of data is stored in the dataset. The 0% value proves that the snapshot feature on ZFS does not cause data duplication because it only stores changes (delta blocks) after the previous snapshot. Thus, this mechanism is able to maintain data integrity with almost zero storage overhead, thereby proving the high efficiency of the ZFS system [20].

The snapshot efficiency test resulted in a value of 0%, which indicates that no additional storage was consumed at the time of measurement. This outcome is consistent with ZFS copy-on-write design, where snapshots only store modified blocks rather than duplicating existing data. Because no data changes occurred between the evaluated snapshots during the observation window, ZFS reported zero additional space usage. Snapshot efficiency is inherently dependent on the frequency of data updates and the duration of observation.

D. Results of Attack Simulation Testing

To validate the effectiveness of the implemented security architecture, a series of attack simulation tests were conducted, covering unauthorized access threats (Brute Force), data integrity compromise (Ransomware), and system file manipulation. The results are summarized in the following table:

TABLE III
COMPARISON OF SECURITY PERFORMANCE BETWEEN DEFAULT SYSTEM AND DID ENABLED SYSTEM

Testing Scenario	Measurement Metric	Default System (Without DiD)	Proposed System (With DiD)	Effectiveness / Improvement
SSH Brute Force	Login attempt limit	Unlimited	5 attempts	99.9% reduction in unauthorized access attempts
	Response time (blocking)	No response	< 2 seconds (after threshold reached)	Real-time mitigation of unauthorized access
Ransomware Simulation	Availability of recovery points	None / manual (Slow)	Periodic snapshots (Hourly)	Ensures availability of read-only data copies
	Recovery Time (RTO), 115 MB file size	5 second (manual restore)	1 second (zfs rollback)	80% time reduction without physical file transfer
File Manipulation	File change detection	Not detected	Detected (AIDE/Logwatch)	Identifies hash changes in critical system files
System Audit Score	Hardening index (lynis)	61	68	11.48% improved overall system security posture

1) *Resistance to Unauthorized Access:* In the Brute-Force test, the default system allowed attackers to perform unlimited login attempts, which could substantially burden the CPU resources of the Orange Pi. With the implementation of Fail2Ban, the system successfully detected suspicious activity and blocked the attacker's IP address at the kernel level using UFW as soon as the threshold for failed attempts was exceeded. This demonstrates that the access-control layer effectively reduces account compromise risk to near zero.

2) *Data Resilience Against Ransomware:* The ransomware simulation was performed by forcibly encrypting the shared folder. In a system without the DiD framework, the data became completely inaccessible. However, by utilizing ZFS Snapshots managed by Sanoid, the system was able to perform a rollback to the state prior to the attack (with a maximum RPO of 60 minutes). The snapshot mechanism proved highly efficient, as the recovery process (RTO) occurred within seconds regardless of the size of the affected dataset.

3) *System Integrity and Transparency:* Through the use of AIDE and rkhunter, all attempts to modify system binaries or essential configuration files were successfully detected. Quantitatively, the improvement in the Lynis Hardening Index from approximately 61 to 68 indicates that the Debian-based operating system has been configured in accordance with professional server security best practices. The automatically managed log accumulation ensures that forensic evidence remains preserved without overloading the internal storage capacity of the Orange Pi.

E. Discussion of Results

The test results that have been carried out show that the application of the DiD concept on a Debian-based NAS system on an Orange Pi device is able to improve system resilience and security without causing a significant performance burden. The implemented DiD approach consists of two main layers, namely host security and storage security, which complement each other in protecting data from various types of threats.

1) *Host Security Layer:* The host security layer acts as the first line of defense against external threats. Firewall and Fail2Ban function as active protection systems that directly prevent unauthorized access. Testing showed that the Firewall successfully rejected all connections on non-permitted ports, while Fail2Ban automatically blocked the attacker's IP address after five failed login attempts. Meanwhile, AIDE and rkhunter function as detection and security audit systems that monitor file integrity changes and detect possible rootkits. Lynis helps the system hardening process by providing a security score index, while Logwatch produces periodic system activity reports to support continuous security monitoring. The combination of all these tools forms a layered defense mechanism that is capable of detecting, preventing, and reporting potential threats in real-time.

2) *Storage Security Layer:* The second layer focuses on data protection through the ZFS snapshot feature. Testing showed that the system was able to recover data that was accidentally deleted or modified without loss of integrity. The copy-on-write feature and ZFS block management ensure that each snapshot only stores changes that have occurred since the previous snapshot, thus not causing data duplication. The storage efficiency value reaching 0% overhead proves that ZFS is very efficient in managing storage space. Thus, the storage layer contributes directly to the aspects of availability and data integrity, two important elements in the information security model.

3) Effectiveness of the Defense-in-Depth Model: The integration between the host layer and the storage layer forms

a comprehensive defense system. The host layer handles prevention and detection aspects, while the storage layer strengthens recovery and data resilience. The test results show that the system is able to detect attacks, prevent unauthorized access, and recover data without significant performance loss. In addition, the entire system is run on an Orange Pi Zero 3 device connected via a smartphone Wi-Fi hotspot network and can be accessed securely via ZeroTier One. This shows that the DiD concept can be implemented effectively even on low-cost hardware, with stable performance and good energy efficiency.

4) Implications of Research Results: Based on the overall testing, it can be concluded that the DiD approach based on open-source software is effective in improving NAS security without requiring large resources. These research results support the application of layered security systems in edge computing and home server environments, where efficiency and reliability are primary factors.

F. System Sustainability Analysis and Long-Term Data Accumulation Management

The implementation of a DiD security system on SBC such as the Orange Pi requires a balance between robust protection mechanisms and the efficient use of limited hardware resources. A key challenge in long-term operation lies in the accumulation of residual data namely activity logs and ZFS snapshots which may hinder or even disrupt system functionality if not systematically managed. To mitigate the risk of disk exhaustion in critical directories such as /var/log, the system relies on the logrotate utility, which automatically compresses and periodically removes outdated log files. This mechanism ensures that security services continuously retain sufficient capacity to record new activity without compromising the stability of the system partition.

In parallel with log management, the sustainability of the data resilience layer is maintained through Sanoid snapshot retention policy. By applying a tiered retention strategy, the system intelligently regulates snapshot accumulation to remain within the physical storage limits. Through the automated removal of expired hourly, daily, and monthly snapshots, the system is able to provide extensive recovery points without causing exponential disk usage growth. This approach is especially critical for storage devices such as SD cards or small SSD, ensuring the availability of space for the NAS primary functions over months or even years of operation.

In responding to sustained cyberattacks, the system demonstrates scalable defensive capability through the dynamic collaboration between Fail2Ban and UFW. When faced with repetitive brute force attempts, the system not only enforces temporary bans but can also automatically escalate the ban duration to reduce processing overhead at the application layer. By shifting the blocking workload to the kernel level firewall, the Orange Pi is able to allocate CPU resources more efficiently to legitimate data requests, even under continuous attack pressure. This synergy ensures that the NAS remains responsive and avoids significant performance degradation resulting from security related workload management.

Long term data integrity is preserved through routine maintenance procedures that include ZFS scrubbing and AIDE database updates. Periodic scrubbing verifies the integrity of each data block, detecting and automatically repairing physical corruption through ZFS self-healing capability. Meanwhile, regular synchronization between the actual system state and the AIDE database index is mandatory whenever official system updates occur. This prevents the buildup of false positive alerts, which could otherwise reduce administrator attentiveness toward genuine security anomalies. Through the integration of these automated mechanisms, the NAS security architecture is able to maintain a consistent defensive posture without requiring intensive manual intervention.

G. Scalability and Portability Analysis

The DiD approach implemented on the Orange Pi demonstrates a high level of portability due to its foundation on a standard Debian Linux distribution [21]. Technically, the entire security stack can be deployed with minimal adjustments across various other SBC ecosystems such as Raspberry Pi, Rock Pi, or Banana Pi. This portability is enabled by the fact that these tools operate at the operating system and filesystem layers (ZFS), both of which are inherently hardware-agnostic. Such flexibility allows developers and IT practitioners to adopt this security model on different SBC devices without redesigning the fundamental command structure or configuration logic [22].

When examining potential deployment in larger NAS environments or enterprise scale systems, the architecture exhibits excellent scalability. On more powerful x86-64 infrastructures, tools like AIDE and Lynis can operate more efficiently in processing large scale system audits [23]. Scalability benefits become even more apparent with the use of ZFS, where its snapshot capabilities and retention management via Sanoid can accommodate storage pools consisting of dozens of disks configured in advanced RAIDZ layouts [24]. In virtualization environments such as Proxmox or vSphere, this DiD framework can be integrated as a standardized security template for each Virtual Machine (VM) functioning as a storage server, thereby strengthening security consistency across the entire data center [25].

Certain resource constraints must be considered when migrating to environments with either higher or lower hardware capacities. The use of ZFS requires a substantial RAM allocation [26]. On SBC with less than 1 GB of RAM, system performance may degrade if all security layers are activated simultaneously. Therefore, this study concludes that although the architecture is highly scalable, its implementation must be tailored to the available hardware capacity to maintain a balanced trade-off between strong security and responsive NAS service performance.

IV. CONCLUSION

This research demonstrates that the implementation of a DiD framework on an Orange Pi based NAS is effective, lightweight, and feasible for low-cost environments. The combination of host-based security tools (UFW, Fail2Ban, AIDE, rkhunter, Lynis, and Logwatch) successfully provides preventive, detective, and monitoring capabilities that reduce the overall attack surface and detect abnormal system activities in real time. Testing results confirm that the firewall blocks unauthorized network access, Fail2Ban mitigates brute-force attempts, and integrity monitoring tools reliably detect file changes.

At the storage layer, the integration of ZFS snapshots and a tiered Sanoid retention policy significantly strengthens data resilience. Snapshot rollback is proven to effectively restore deleted or modified files, achieving a RPO of one hour with minimal resource overhead. Performance measurements also show that activating multiple security layers does not introduce noticeable degradation on the Orange Pi Zero 3, keeping CPU usage below 10% and memory usage at approximately one-third of total capacity.

This research concludes that an SBC-based NAS system can achieve a secure and highly available architecture when equipped with layered host security and ZFS-based data protection. The proposed model offers a practical, low-cost, and academically relevant reference for developing secure NAS solutions for SMEs, home users, and research environments. Future work may explore automation, scalability enhancements, and comparative evaluations with other security architectures to further strengthen the proposed framework.

REFERENCES

[1] M. Adila, A. S. Y. Santoso, and A. P. Sari, 'Penerapan Sistem Operasi Network Attached Storage "FreeNAS" sebagai Solusi Kegiatan Berbagi File. (Studi kasus : Fakultas Ilmu Komputer, UPN Jatim)', *Jurnal Ilmiah Teknologi Informasi dan Robotika*, vol. 5, no. 2, pp. 53–59, Dec. 2023, doi: 10.33005/jifti.v5i2.180.

[2] R. A. Firmansyah and W. Adhiwibowo, 'Performance Analysis of Low Cost Orange Pi Based NAS Server for SMEs', *Jurnal Informatika Teknologi dan Sains*, vol. 7, no. 3, 2025.

[3] H. Gunawan, A. Handijono, A. Putra, and A. Zein, 'Sistem Monitoring Serangan DOS dengan Metode Intrusion Detection System (IDS) Snort menggunakan Aplikasi Berbasis Python pada Sistem Operasi Linux', *Spectrum: Multidisciplinary Journal*, vol. 2, no. 3.

[4] D. Riyanto, K. Khairil, and E. P. Rohmawan, 'An Analysis and Design of Network Security Using Firewall at the Library and Archives Services of Bengkulu province', *Jurnal Komputer, Informasi dan Teknologi*, vol. 1, no. 2, Dec. 2021, doi: 10.53697/jkomitek.v1i2.280.

[5] S. D. Hitefield, 'A Defense-In-Depth Security Architecture for Software Defined Radio Systems', Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2019.

[6] B. Gajbhiye, S. Jain, and O. Goel, 'Defense in Depth Strategies for Zero Trust Security Models', *International Journal for Research Publication and Seminar*, vol. 15, no. 3, 2024.

[7] V. Babanov, 'Internals of Defense-In-Depth Strategy in Cybersecurity', *Scientific journal*, no. 2, Dec. 2024, doi: 10.70265/PNEZ3158.

[8] Farhannullah and M. Hardjianto, 'Sistem Monitoring Serangan SSH dengan Metode Intrusion Prevention System (IPS) Fail2ban Menggunakan Python Pada Sistem Operasi Linux', *Technology of Information and Communication*, vol. 11, no. 1, pp. 33–38, Sep. 2022, doi: 10.70309/ticom.v11i1.68.

[9] April Rustianto, Arif Fadillah, and Jemiro Kasih, 'Pencegahan Dan Visualisasi Serangan Brute Force Menggunakan Fail2ban, Prometheus, dan Grafana Studi Kasus Di Sekolah Tinggi Teknologi Terpadu Nurul Fikri', *Jurnal Publikasi Teknik Informatika*, vol. 4, no. 2, pp. 195–209, May 2025, doi: 10.55606/jupti.v4i2.5144.

[10] M. Ridho, A. Hafizh, I. Dani, and T. Ariyadi, 'Peningkatan Keamanan SSH Server Berbasis Linux melalui Implementasi Fail2Ban dan Uji Serangan Brute Force', *Jurnal Penelitian Multidisiplin Bangsa*, vol. 1, 2025.

[11] J. Sani, 'Improved Log Monitoring UsingHost-based Intrusion Detection System', *Advanced International Journal of Multidisciplinary Research*, vol. 1, no. 1, 2023.

[12] B. Havano and A. Dobush, 'Enhancing host intrusion detection systems for Linux based network operating systems', *Advances in Cyber-Physical Systems*, vol. 10, no. 1, pp. 54–58, May 2025, doi: 10.23939/acps2025.01.054.

[13] A. C. Jaya, 'Single-Board Computer For Affordable Personal Data Storage Server', *Jurnal Mantik*, vol. 4, no. 36, 2020.

[14] Z. Chen, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, 'Host-Based Network Intrusion Detection via Feature Flattening and Two-stage Collaborative Classifier', *arXiv*, vol. 1, no. 1, 2023, doi: 10.48550/arXiv.2306.09451.

[15] Y. Mawaru, M. Yahya, and A. M. Mappalotteng, 'Analisis Efektivitas IPTABLES Dalam Melindungi Jaringan Dari Serangan DDoS', *Pinisi Journal of Science & Technology*, vol. 1, no. 5, 2024.

[16] K. A. Prasetyo, M. Idhom, and H. E. Wahanani, 'Sistem Pencegahan Serangan Bruteforce pada Multiple Server dengan Menggunakan Fail2ban', *Jurnal Informatika dan Sistem Informasi (JIFoSI)*, vol. 1, no. 3, 2020.

[17] M. Á. Enríquez, J. P. Marcial, T. R. Linares, and A. C. Z. Vázquez, 'Análisis de servicios y aplicaciones en sistemas Linux con monitoreo de logs', *Abstraction & Application*, pp. 23–32, 2024.

[18] J. Stühn, J.-N. Hilgert, and M. Lambertz, 'The Hidden Threat: Analysis of Linux Rootkit Techniques and Limitations of Current Detection Tools', *Digital Threats: Research and Practice*, vol. 5, no. 3, 2024, doi: 10.1145/3688808.

[19] E. D. Ansong, E. A. Affum, and E. Donkor, 'Framework for Security Auditing in Linux: Detecting and Mitigating Privilege Escalation Vulnerabilities Using PriviLynis', *Physical Communication*, 2025.

[20] T. Fernando and D. Jayawardena, 'Leveraging ZFS Snapshots for Incremental Recovery in Hybrid Unix Networks', *International Journal of Science, Engineering and Technology*, vol. 11, no. 6, 2023.

[21] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, 'A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures', *IEEE Access*, vol. 7, pp. 82721–82743, 2019, doi: 10.1109/ACCESS.2019.2924045.

[22] W. Stallings, *Effective cybersecurity: understanding and using standards and best practices*. Upper Saddle River, NJ: Addison-Wesley, 2019.

[23] M. M. I. Jabed, M. S. Hossain, S. Ferdous, R. B. Ankhi, and A. B. Gupta, 'AI-Driven Intrusion Detection Systems: A Business Analyst's Framework for Enhancing Enterprise Security and Intelligence', *International Journal of Research Publications in Engineering, Technology and Management*, vol. 08, no. 05, Sep. 2025, doi: 10.15662/IJRPETM.2025.0805004.

[24] N. L. Beebe, S. D. Stacy, and D. Stuckey, 'Digital forensic implications of ZFS', *Digital Investigation*, vol. 6, 2009, doi: 10.1016/j.diin.2009.06.006.

[25] Z. Li, G. Liu, Y. Dang, Z. Shang, and N. Lin, 'Research on New Virtualization Security Protection Management System Based on Cloud Platform', *Journal of Applied Data Sciences*, vol. 4, no. 2, 2023.

[26] O. Rodeh, J. Bacik, and C. Mason, 'BTRFS: The Linux B-Tree Filesystem', *ACM Transactions on Storage*, vol. 9, no. 3, 2013, doi: 10.1145/2501620.2501623.