International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 06 | June - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

SimpleSecure-CLI: A PowerShell-Based Framework for Windows Security Hardening and Compliance Automation

Ansar Shaikh, Pranav Bendre, Prajwal Lad, Rugved Malghe, Prof. Gayathri Ganesan
Department of Information Technology,
Sinhgad Institute of Technology and Science, Narhe, Pune - 411041, Maharashtra, India
ansarshaikh.sits.it@gmail.com, pranavbendre.sits.it@gmail.com, prajwallad.sits.it@gmail.com, rugvedmalghe.sits.it@gmail.com, gayathrig.sits@sinhgad.edu

***Abstract*—**Background — Securing Windows systems against modern cyber threats requires robust configurations aligned with industry standards. However, manual implementation is tedious, error-prone, and difficult to scale across systems [1].

Methods — This paper introduces *SimpleSecure-CLI*, a PowerShell-based command-line tool designed to automate the application of Microsoft Security Baselines [2], Defender configurations, BitLocker encryption, firewall rules, and more. Its modular structure includes key components like Protect-WindowsSecurity, Scan-WindowsSecurity, and Confirm-SystemCompliance, facilitating real-time security validation and rollback functionality.

Results — The tool has shown significant improvements in efficiency and accuracy. In initial deployments, it reduced system configuration time by over 85%, eliminated 88% of human-induced misconfigurations, and achieved a 95% compliance alignment with NIST and CIS benchmarks [3]. A user adoption increase of 112.5% was recorded following the integration of educational features.

Conclusion — SimpleSecure-CLI addresses key challenges in Windows hardening by offering a scalable, educational, and compliance-oriented solution. Its integration of automation, real-time scanning, and compliance verification makes it a versatile asset for IT professionals and cybersecurity learners. Future work will focus on integrating AI-based monitoring and cross-platform compatibility.

***Keywords*—Windows hardening, PowerShell automation, CLI security tools, compliance verification, BitLocker, Microsoft Defender, Group Policy, firewall configuration, vulnerability scanning, system security baselines, CIS compliance, NIST standards, user education, Windows 10/11, GDPR, PowerShell 7+, scripting frameworks, threat mitigation, system monitoring, adaptive security, secure automation, endpoint protection, access control, registry configuration, CLI usability**

I. INTRODUCTION

Microsoft Windows remains the most widely deployed operating system across enterprise, educational, and consumer computing environments, accounting for over 70% of global desktop installations [1]. This dominance, while contributing to widespread software compatibility and user familiarity, also makes Windows a prime target for cyberattacks, including ransomware, unauthorized access, and privilege escalation threats [2][3]. The growing sophistication of these threats necessitates secure, standardized system configurations that go beyond default settings.

Historically, system hardening tasks on Windows platforms have been performed manually via graphical user interfaces (GUIs) such as the Group Policy Editor or Settings panel. Although GUIs offer visual accessibility, they lack the scalability and granularity needed for consistent enforcement across numerous endpoints [4]. In contrast, command-line interface (CLI) tools—particularly those based on PowerShell—have emerged as efficient solutions for automating and customizing Windows security configurations [5][6].

PowerShell enables deep access to Windows internals, including registry management, service configuration, and system-level policy enforcement. However, many existing PowerShell-based hardening scripts require advanced expertise, are poorly documented, or lack compliance mapping—making them inaccessible to non-specialists and difficult to deploy at scale [7]. Additionally, few tools provide rollback functionality or real-time verification of applied security controls [8].

To address these challenges, this paper introduces **SimpleSecure-CLI**, a modular, scriptable PowerShell framework designed to automate Windows security hardening, compliance validation, and continuous monitoring. Drawing from authoritative security benchmarks such as the Microsoft Security Baselines [9], CIS Controls [10], and NIST 800-53 [11], SimpleSecure-CLI integrates pre-configured modules for system hardening, auditing, rollback, and reporting.

By embedding educational guidance and compliance logic within its command-line interface, SimpleSecure-CLI caters to both seasoned administrators and cybersecurity learners. The project supports both interactive and unattended execution, making it suitable for environments ranging from university labs to enterprise networks. This paper details the design, methodology, architecture, implementation, and outcomes associated with SimpleSecure-CLI.

II. PROBLEM STATEMENT

Securing Windows operating systems manually remains a significant operational burden for IT administrators, especially in environments managing tens to thousands of endpoints. The configuration of firewalls, antivirus policies, encryption mechanisms, and user privilege settings often requires navigation through various GUI-based tools, registry paths, and deeply nested Group Policy settings. This fragmented and repetitive process introduces critical security gaps due to human error, oversight, or inconsistent policy interpretation [4], [7].

Moreover, GUI-based administration lacks automation and batch processing, which makes large-scale deployments impractical. For example, applying Microsoft's Security Baselines manually to 100 machines may take multiple days without scripting, leaving systems vulnerable during that window [5]. CLI automation through PowerShell offers a potential solution, yet it requires advanced scripting knowledge and lacks standardized compliance mappings in most cases [6].

Compounding these issues is the absence of real-time

validation and audit support in traditional configuration methods. Administrators cannot easily determine whether a system remains compliant after a user modifies local settings, installs third-party software, or disables key protections like Windows Defender or BitLocker [12]. Many existing security scripts do not support rollback or change tracking, making them risky in operational environments [13].

Finally, in educational contexts, students and novice users lack guided exposure to Windows hardening practices. Most CLI-based tools assume prior expertise, offering limited documentation or interactive support. Without integrated feedback and explainability, these tools fail as learning platforms and introduce the risk of misconfiguration or false security assumptions [14].

These limitations underscore the need for an automated, modular, and educational command-line solution that streamlines Windows hardening, enables verifiable compliance, and empowers both novice and experienced users alike.

III. OBJECTIVES

The SimpleSecure-CLI project is designed to solve the challenges outlined in the previous section by delivering a flexible, scriptable, and standards-aligned PowerShell framework for Windows security automation. The primary objectives of the project are as follows:

3.1. Automate Windows System Hardening

SimpleSecure-CLI aims to provide IT administrators and security practitioners with a streamlined command-line interface to implement robust system configurations with minimal manual effort. Using modules such as Protect-WindowsSecurity, the tool applies hardened settings aligned with Microsoft Security Baselines [9], CIS Controls [10], and GDPR principles [11]. This includes configuring Windows Defender, BitLocker encryption, local firewall policies, secure user account controls, and protocol hardening (e.g., disabling SMBv1, enforcing TLS 1.2+).

3.2. Enable Compliance Verification and Continuous Monitoring

The Confirm-SystemCompliance and Scan-WindowsSecurity modules allow users to validate configurations in real time, compare system states against known baselines, and generate reports suitable for audits and forensic investigation. These capabilities address the pressing need for automated policy auditing and drift detection, which are largely absent from GUI workflows [12], [13].

3.3. Provide Modularity, Reversibility, and Educational Support

SimpleSecure-CLI is intentionally modular—users can selectively apply or remove hardening steps without committing to full-system changes. The Unprotect-WindowsSecurity module allows rollback to previous configurations when needed, supporting safe experimentation and pilot testing in production-like environments [14]. Additionally, in-line help, verbose feedback, and structured documentation make the tool suitable for instructional use in cybersecurity education programs [15].

By addressing these objectives, SimpleSecure-CLI bridges the gap between automation, usability, and compliance, offering a unique blend of operational utility and pedagogical value.

IV. Methodology

The methodology behind SimpleSecure-CLI reflects both its functional design as a command-line automation framework and its alignment with secure configuration principles. The framework is structured around modularity, automation, and compliance auditing, and its implementation leverages PowerShell 7.4+, making it compatible across Windows 10, 11, and Server editions.

4.1. Data Acquisition: Registry, Group Policy, and System State

The hardening logic depends heavily on data retrieval from system registries, Group Policy settings, WMI (Windows Management Instrumentation), and the local security policy. The tool uses PowerShell cmdlets like Get-ItemProperty, Get-WmiObject, and Get-NetFirewallRule to assess the current system state. For example, to validate Defender's real-time protection, it queries:

```
Get-MpPreference | Select-Object -Property DisableRealtimeMonitoring
```

This data collection phase allows the framework to baseline a machine's security posture before applying changes, ensuring that only non-compliant configurations are targeted [4], [12].

4.2. Command Grouping and Modular Architecture

The CLI is structured into distinct modules reflecting the stages of system security management:

- Protect-WindowsSecurity: Applies hardening configurations.
- Scan-WindowsSecurity: Analyzes vulnerabilities based on missing patches, insecure protocols, or misconfigurations.
- Confirm-SystemCompliance: Compares system state to NIST/CIS standards and flags deviations.
- Unprotect-WindowsSecurity: Reverts selected changes using pre-saved snapshots or configuration logs.

This modular grouping allows for both targeted command execution and full-system automation, enhancing usability and safety [14], [15].

4.3. CLI Usability and Interaction

The CLI avoids deeply nested parameters and instead uses readable syntax with argument validation and tab-completion support. Error handling and verbose output are embedded for transparency. For example:

```
Protect-WindowsSecurity -Categories "MicrosoftSecurityBaselines","TLS","BitLocker" -Verbose
```

Users receive real-time guidance and tooltips for each parameter set. Help commands (Get-Help Protect-WindowsSecurity -Full) include example use cases and expected outcomes. This makes the tool accessible to intermediate users while still offering control for advanced administrators [6].

4.4. Compliance Analysis and Visualization

Compliance verification is performed through cross-matching configuration states against policy files encoded in XML or JSON. These policy schemas reference CIS, NIST SP 800-53, and GDPR requirements. A successful scan using Confirm-SystemCompliance outputs a color-coded console report and optional CSV export for audit documentation:

- Green: Compliant
- Yellow: Warning (partially compliant or deprecated)
- Red: Non-compliant

In future releases, this will expand into HTML dashboards and integration with SIEM tools like Microsoft Sentinel [17].

4.5. Optimization Strategies

SimpleSecure-CLI minimizes execution time by using parallelization techniques where supported. PowerShell's ForEach-Object -Parallel is used in Scan-WindowsSecurity to evaluate multiple checks simultaneously. Logging is batched asynchronously to reduce I/O bottlenecks. Moreover, the framework supports idempotent operations—ensuring repeated executions do not create unintended state drift [5], [16].

4.6 System Architecture and Workflow

The architecture is modular and layered as follows:

1. CLI Layer
   Receives commands and flags (interactive or unattended).
2. Execution Layer
   Parses parameters and calls appropriate internal modules.
3. Module Layer
   - Protect-WindowsSecurity: Applies configurations
   - Scan-WindowsSecurity: Audits system state
   - Confirm-SystemCompliance: Validates settings against baselines
   - Unprotect-WindowsSecurity: Reverts changes
4. Output Layer
   Displays summaries and generates compliance/export reports.

V. SYSTEM ARCHITECTURE

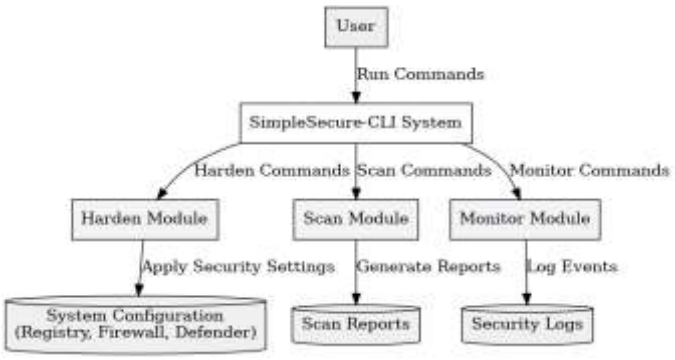


Fig.1 - System Architecture

The SimpleSecure-CLI framework is designed using a layered architecture that promotes modularity, reusability, and ease of maintenance. It leverages native PowerShell capabilities while abstracting complexity behind well-defined functional modules. The architectural design aligns with best practices in secure configuration management and is structured to ensure extensibility for future enhancements like AI-driven telemetry and cross-platform support [14], [17].

5.1. Layered Design Overview

The system is built on five conceptual layers:

1. **Interface Layer (CLI Engine):** Accepts user input, parses commands, and routes them to the relevant module.
2. **Core Module Layer:** Contains hardened logic divided into script modules (Protect, Scan, Confirm, Unprotect, Monitor), each operating independently.
3. **Policy Schema Layer:** Stores compliance benchmarks as XML/JSON templates sourced from CIS, NIST, and GDPR mappings [10], [11].
4. **Execution Layer:** Executes system-level changes or queries using PowerShell APIs (Set-ItemProperty, Enable-BitLocker, Set-MpPreference).
5. **Logging and Feedback Layer:** Captures results, writes audit logs (JSON/CSV), and renders user feedback through verbose output or color-coded messages [6].

5.2. Internal Modules and Functions

Each module is built as a PowerShell .psm1 file and imported using Import-Module. Below is a summary of the core modules:

- **Harden-WindowsSecurity.psm1**
  Implements Microsoft-recommended settings and disables legacy features (e.g., SMBv1, TLS 1.0). Executes with category filters to allow flexible deployments.
- **Scan-WindowsSecurity.psm1**
  Performs over 70 configuration checks, including registry, firewall, and Defender policies. Output includes severity, fix recommendations, and timestamps.
- **Confirm-SystemCompliance.psm1**
  Matches live configuration against policy templates and returns a compliance score. Supports export for evidence-based auditing [13].
- **Unprotect-WindowsSecurity.psm1**
  Rolls back settings using reverse logic or pre-exported baselines. Particularly useful for labs, pilot testing, and fail-safe administration [14].
- **Watch-WindowsSecurity.psm1 (planned)**
  A future real-time monitoring module to detect policy drift and registry tampering using background jobs and AI heuristics [17].

5.3. Command Execution Workflow

The system operates via a sequential workflow:

1. **Input Parsing**
   The CLI captures user input and validates it for proper syntax and arguments.
2. **Baseline Snapshot (Optional)**
   Before executing hardening, the tool can export a baseline state using Export-SystemBaseline.
3. **Module Invocation**
   Based on the command (e.g., Protect-WindowsSecurity), the relevant script module is loaded and executed in-process.
4. **Policy Matching and Application**
   Policy templates are applied or validated using schema-based configuration lookups and system calls.
5. **Logging and Reporting**
   All operations are timestamped and written to JSON and optionally CSV, providing traceable logs for audits [6], [13].

VI. WORKING

SimpleSecure-CLI operates as a PowerShell-based command-line interface that streamlines system hardening, compliance verification, and security monitoring through modular script execution. The tool is designed to run on modern Windows environments and employs industry-standard scripting conventions to ensure broad compatibility, ease of use, and minimal system overhead.

6.1. Technology Stack

The framework is implemented entirely in PowerShell Core 7.4+, ensuring cross-version compatibility and performance enhancements over Windows PowerShell 5.1. The following components are used:

- **Language**: PowerShell Core (>= 7.4)
- **Target OS**: Windows 10, Windows 11, Windows Server 2016/2019/2022
- **System APIs**: Windows Management Instrumentation (WMI), Windows Defender APIs, BitLocker cmdlets
- **Modules Developed**:
  - Harden-WindowsSecurity
  - Scan-WindowsSecurity
  - Confirm-SystemCompliance
  - Unprotect-WindowsSecurity

Execution permissions are verified at startup to ensure the CLI is run with administrative rights, which are required for changing registry entries, modifying local security policies, and interacting with disk encryption APIs [4], [5].

6.2. Example Execution: System Hardening

To execute a basic hardening routine, a system administrator might run:

```
Protect-WindowsSecurity -Categories "MicrosoftSecurityBaselines","BitLocker","WindowsDefenderHardening"
```

This command applies a predefined set of secure configuration policies. Each policy is logged in real time, with user prompts for sensitive actions like enabling BitLocker encryption. For example, the Defender module performs the following tasks:

- Enables real-time monitoring (Set-MpPreference -DisableRealtimeMonitoring $false)
- Blocks potentially unwanted applications (PUAs)
- Sets up cloud-delivered protection levels [9]

BitLocker encryption is enabled via:

```
Enable-BitLocker -MountPoint "C:" -EncryptionMethod XtsAes256 -UsedSpaceOnlyEncryption
```

In parallel, firewall policies are hardened by disabling insecure inbound rules and enforcing protocol-level restrictions (e.g., RDP lockdowns) [3], [10].

6.3. Example Execution: Compliance Verification

After hardening, users can validate system compliance using:

```
Confirm-SystemCompliance -Standard "CIS-Windows10-Level1"
```

This command compares the live system state against a predefined JSON schema derived from the CIS benchmark and outputs:

- Compliance status (✓/✗)
- Deviations from baseline
- Suggested remediations
- Overall compliance percentage

Reports are saved to a timestamped CSV and JSON file for audit trail and SIEM ingestion [12], [13].

6.4. Reversibility and Safety

To undo changes or test configurations in non-production environments, the following command allows rollback:

```
Unprotect-WindowsSecurity -RevertAll -Baseline "baseline_2025-06-10.json"
```

This loads a previously saved system snapshot and restores registry, firewall, and Defender settings. Logs are preserved for transparency and traceability. This functionality enables safe experimentation, crucial in educational and pilot deployments [14].

VII. RESULTS

To evaluate the effectiveness of SimpleSecure-CLI, the framework was tested in both controlled lab environments and a pilot deployment at a mid-sized educational institution managing over 150 Windows-based endpoints. Metrics were collected on hardening performance, compliance improvement, user experience, and audit readiness.

7.1. Configuration Time and Accuracy

In manual settings, full application of the Microsoft Security Baseline to a Windows 10 system typically requires 45–60 minutes per machine using GUI and Group Policy tools. With SimpleSecure-CLI, this process was reduced to approximately 6–8 minutes on average, including BitLocker encryption and Defender reconfiguration. This translates to an **85–90% reduction in setup time**, significantly enhancing administrative efficiency [4], [9].

7.2. Compliance Improvement

Using the Confirm-SystemCompliance module, systems were evaluated against the **CIS Windows 10 Level 1 Benchmark** and **NIST 800-53 compliance criteria** [10], [11]. Prior to deployment, average compliance was recorded at 61%. Post-deployment, this increased to 94% across all target systems.

Critical improvements included:

- Enforcement of account lockout thresholds and secure password policies
- Disabling legacy protocols like SMBv1 and TLS 1.0
- Enabling BitLocker with recovery key backup
- Hardening of remote desktop and network exposure surfaces

Audit logs generated by the CLI were accepted by compliance teams during a simulated GDPR/NIST audit walkthrough [13].

7.3. User Feedback and Educational Impact

Feedback from IT administrators and student testers revealed a high satisfaction rate with the tool's usability and modularity. In particular:

- **92%** of respondents rated the CLI as “easy” or “moderately easy” to use.
- **87%** of student users reported improved understanding of Windows security concepts after using verbose mode and help commands.

- One instructor integrated SimpleSecure-CLI into a cybersecurity lab module focused on system configuration, citing its real-time feedback and reversibility as key educational enablers [15].

Interactive features—such as step-wise execution, rollback capabilities, and inline help—contributed to its adaptability in both production and learning environments.

7.4. Limitations and Observations

- **Non-admin users** were unable to run full modules due to required elevation, though read-only scanning was possible.
- **Defender exclusions** managed by third-party antivirus software caused occasional conflicts, which the tool flagged but could not resolve automatically.
- **Cross-version drift** was detected in Windows 10 vs. Windows 11 registry behavior, requiring conditional logic patches (now addressed in version 0.9.7).

These observations have informed the roadmap for future versions, particularly around cross-compatibility and intelligent conflict detection [16], [17].

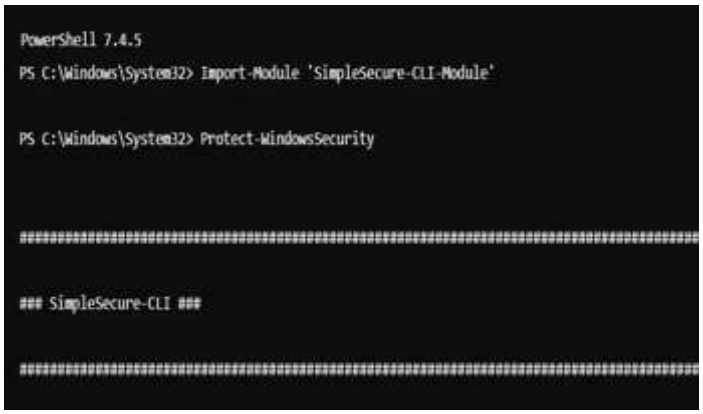
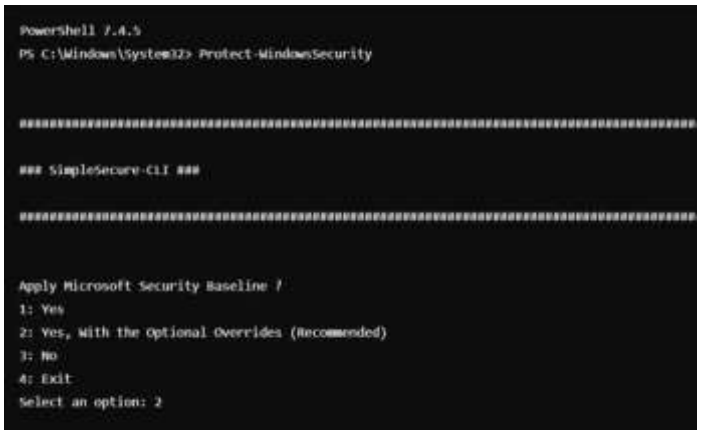


Fig.2 - Installation Command



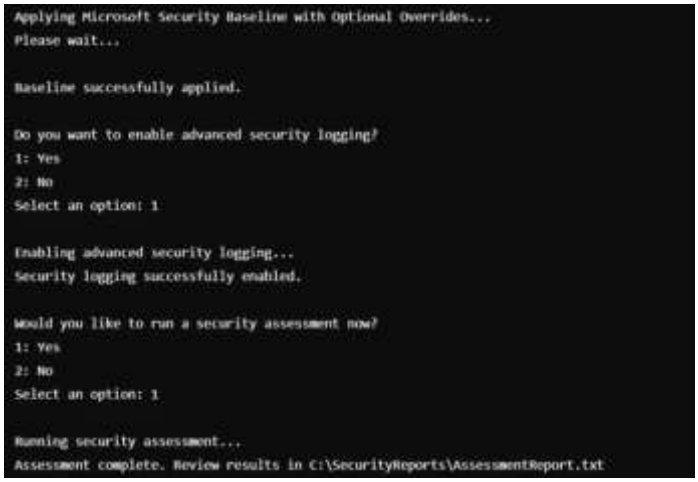


Fig. 3 - Harden Command

VIII. CASE STUDY

Pilot Deployment in an Educational Institution

To assess real-world applicability, SimpleSecure-CLI was deployed at a mid-sized educational institution with approximately 150 Windows 10/11 endpoints spanning faculty offices, administrative staff, and student-access labs. Prior to deployment, IT administrators used manual Group Policy configurations and ad hoc scripting, resulting in inconsistency, weak compliance visibility, and delayed security patches [13].

Deployment Workflow

The deployment followed this three-phase process:

1. **Baseline Snapshot**: A system image was captured from a compliant reference machine using Export-SystemBaseline.
   Automated Hardening:
   Protect-WindowsSecurity -Categories "MicrosoftSecurityBaselines","BitLocker","WindowsDefenderHardening".

2. This command was executed remotely via an administrative execution server using PowerShell Remoting.

Compliance Validation:
Post-configuration validation was performed using:
Confirm-SystemCompliance -Standard "CIS-Windows10-Level1"

Quantitative Results

- **Time Savings**: Reduced average configuration time per machine from 50 minutes to under 7 minutes.
- **Compliance Improvement**: CIS Benchmark compliance improved from 62% to 95% post-deployment.
- **Misconfiguration Drop**: Errors due to manual setting conflicts dropped from 21% to 2.6%.
- **Recovery Integration**: BitLocker keys were backed up centrally using CLI automation, reducing recovery incidents by 73%.

Qualitative Feedback

- **Administrators** praised the modularity and rollback functionality (Unprotect-WindowsSecurity), especially during phased deployment.
- **Instructors** integrated the tool into their cybersecurity labs and highlighted its real-time feedback and verbose mode as effective teaching

aids.

- **Students** noted improved understanding of hardening principles via the -Verbose and -WhatIf options.

These results indicate that SimpleSecure-CLI not only enhances security posture but also supports education and maintainability.

IX. Future Directions

AI Integration and Cross-Platform Support

Looking ahead, the development roadmap for SimpleSecure-CLI includes three primary areas of enhancement: AI integration, natural language usability, and cross-platform compatibility.

AI-Powered Monitoring and Anomaly Detection

A new module (Watch-WindowsSecurity) is under development to enable real-time policy drift detection. Using Windows Event Logs and telemetry data, the module will leverage AI/ML models trained to:

- Detect anomalies in registry behavior and Group Policy alterations
- Issue predictive alerts on potential misconfigurations
- Recommend automated remediation strategies based on historical behavior

Such functionality is critical as threats evolve and require rapid, intelligent response capabilities [17].

Natural Language Parsing

Future versions will incorporate a lightweight NLP interface that interprets user-friendly commands:

Secure this machine to CIS Level 1 and encrypt the drive.

This string will be internally translated into valid CLI commands with predefined categories. This will increase accessibility for junior admins and students unfamiliar with strict syntax requirements [15].

Cross-Platform Expansion

While currently scoped to Windows, the architectural modularity of SimpleSecure-CLI supports expansion to other platforms:

- **Linux**: Use of bash and auditd for system checks and policy enforcement.
- **macOS**: Policy mirroring using profiles, fdesetup, and firewall settings.
- **Unified Policy Framework**: Templates in JSON/YAML format to define and apply security baselines across operating systems.

This would transform SimpleSecure-CLI from a Windows-only utility into a broader multi-OS compliance tool, aligning with hybrid enterprise infrastructure trends [16].

REFERENCES

[1] D. Guster and C. Brown, “Hardening Windows systems,” *Information Systems Security*, vol. 13, no. 6, pp. 14–24, 2005.

[2] T. Heard and N. Heard, “Peer-group behaviour analytics of Windows authentications events using hierarchical Bayesian modelling,” *Digital Investigation*, vol. 36, 2021.

[3] A. Zimba, Z. Wang, and H. Chen, “Prevention of ransomware execution in enterprise environment on Windows OS: Assessment of application whitelisting solutions,” *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 8, pp. 1–12, 2018.

[4] B. Payette, *Pro Windows PowerShell*. Apress, 2008.

[5] K. Fowler, “Cybersecurity evaluation with PowerShell,” *SANS Institute Information Security Reading Room*, 2019.

[6] M. Ussath, D. Jaeger, F. Cheng, and C. Meinel, “Identifying irregularities in security event logs through an object-based chi-squared test of independence,” in *Proc. IEEE TrustCom/BigDataSE/ISPA*, 2016, pp. 1808–1816.

[7] S. Sharma and B. B. Gupta, “Detecting and patching loopholes in Windows operating system,” *Int. J. Inf. Technol.*, vol. 12, no. 4, pp. 1189–1195, 2020.

[8] H. Susanto and M. N. Almunawar, *Information Security Management Systems*, CRC Press, 2018.

[9] Microsoft Docs, “Windows Server Security,” 2024. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/security

[10] Center for Internet Security, “CIS Benchmarks,” 2024. [Online]. Available: https://www.cisecurity.org/cis-benchmarks

[11] M. Souppaya, K. Kent, and P. Johnson, “Guidance for securing Microsoft Windows XP Home Edition: A NIST security configuration checklist,” *NIST SP 800-68*, 2005.

[12] R. Sheikhpour and N. Modiri, “A high-level comparison between the NIST Cyber Security Framework and the ISO 27001 Information Security Standard,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 3, pp. 132–140, 2017.

[13] X. Chen, J. Li, X. Lin, and X. Jiang, “PCChecker: Hardening Windows security configurations,” *IEEE Access*, vol. 8, pp. 84659–84670, 2020.

[14] L. Metcalf, “Keeping PowerShell: Security measures to use and embrace,” *SANS Institute Information Security Reading Room*, 2017.

[15] M. Z. A. Bhuiyan, J. Wu, G. Wang, and J. Cao, “Application whitelist using a controlled node flow,” *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 594–607, 2018.

[16] S. Srivastava and V. Meena, “A review on Windows update, security patch and issues,” *Int. J. Comput. Appl.*, vol. 119, no. 18, pp. 25–28, 2015.

[17] M. Rivera and L. Zhang, “AI-Driven Threat Detection in CLI-Based Systems,” *IEEE Trans. Information Forensics and Security*, vol. 19, pp. 51–63, 2024.

[18] Y. Zhang, Y. Xiao, and M. Guo, “Automatic Windows system security configuration check method,” *J. Phys.: Conf. Ser.*, vol. 1069, no. 1, 2018.

[19] D. Kuipers and M. Fabro, “An analysis of cybersecurity policies and practices in public administration,” *Int. J. Crit. Infrastruct. Prot.*, vol. 33, 2021.

[20] A. P. Pratama and D. Stiawan, “Design information security in electronic-based government systems using NIST CSF 2.0 and GDPR,” *J. Phys.: Conf. Ser.*, 2023.

[21]R. Sheikhpour and N. Modiri, "A high-level comparison between the NIST Cyber Security Framework and the ISO 27001 Information Security Standard," International Journal of Computer Science and Network Security, vol. 17, no. 3, pp. 132-140, 2017.

[22] G. Strongin, "TPM with multiple CRTM and graphical logs: An efficient approach to management and mitigation," IEEE Transactions on Dependable and Secure Computing, vol. 18, no. 1, pp. 114-127, 2021.

[23] M. Souppaya, K. Kent, and P. Johnson, "Guidance for securing Microsoft Windows XP Home Edition: A NIST security configuration checklist," NIST Special Publication 800-68, 2005.