Jurnal Teknik Informatika Vol. 19 No. 1, April 2026 (12-21)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v19i1.46816

JURNAL TEKNIK INFORMATIKA
Homepage : http://journal.uinjkt.ac.id/index.php/ti

A Socio-Technical Windows 11 Hardening Framework Integrating Ferret-Windows and CIS Benchmarks

Prasetyo Adi Wibowo Putro[1*], Ray Novita Yasa[2], Mochammad Latief Reswandana Musonip[3], Dimas Nugroho Putro[4], Agry Zharfa[5]

[1,2,3,4,5] Cryptographic Engineering, Politeknik Siber dan Sandi Negara
JL H. Usa, Putat Nutug, Ciseeng, Bogor, Indonesia

Article:
Accepted: February 17, 2026
Revised: December 13, 2025
Issued: April 30, 2026

***Correspondence Address:**
prasetyo.adi@poltekssn.ac.id

ABSTRACT

The high level of vulnerability in the Windows operating system requires implementing a hardening strategy that focuses not only on technical security but also on end-user convenience. This study, using a Design Science Research (DSR) methodology, aims to design and evaluate a Windows-based host security configuration by applying 35 parameters developed by integrating the Ferret-Windows open-source tool and the Microsoft Windows 11 Benchmark CIS standard. The research method includes a descriptive study to analyze the effectiveness of the tool and its parameter formulation, and a prescriptive study to formulate a combined configuration. The evaluation was carried out through functional testing and usability measurements using the System Usability Scale (SUS) instrument for 31 end-users (e.g., staff and students) in a public institution context, using the System Usability Scale (SUS) instrument on a standardized test laptop. The results showed that the configured system obtained an average SUS score of 73.63, which falls within the "good" category according to the standard interpretation scale, indicating that the resulting system is usable by most users. These findings indicate that the proposed hardening framework results in a system configuration that achieves acceptable usability without significant user sacrifice. While the study's limitations include a focus on usability rather than quantitative pre-/post-security validation, it provides a practical contribution, a socio-technical hardening framework adaptable for public institutions.

Keywords : *protection; operating system; Windows 11; CIS standard; ferret.*

Jurnal Teknik Informatika Vol. 19 No. 1, April 2026 (12-21)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v19i1.46816

1. INTRODUCTION

Windows is widely used in both personal and organizational environments, which makes it an attractive target for various cyberattacks. Its broad deployment means that a single successful compromise can affect many users and devices simultaneously, amplifying its potential impact [1]. In addition, Windows provides built-in administrative tools, such as PowerShell, that are essential for system management but can also be misused by attackers to execute malicious commands that blend with normal operations and are challenging to detect. [2]. Over time, attackers have increasingly adopted techniques that exploit legitimate system processes and stealthy execution methods, indicating that traditional reactive defenses alone are no longer sufficient to secure Windows systems [3] effectively.

Securing Windows through system hardening involves challenges across both software and hardware dimensions. For example, kernel-level or transient execution attacks can bypass security controls even without software bugs, and mitigation strategies for such attacks often incur performance limitations. [4], Advanced Persistent Threats (APTs) are also capable of evading Windows security mechanisms to access or exfiltrate sensitive data [5]. At the hardware level, the complexity of device supply chains and the presence of direct memory access pathways increase exposure to low-level attacks that are difficult to detect or prevent using software-only solutions [6], while software-implemented trusted execution environments still face limitations under everyday operational conditions [7]. These challenges highlight the need for well-designed and consistently applied configuration controls rather than relying solely on reactive or single-layer protection mechanisms.

However, improving security often reduces user convenience. Security configurations that are too restrictive may interfere with everyday computing activities, reduce user flexibility, and create resistance to long-term adoption. [8]. In many organizations, this results in hardening being applied inconsistently or only partially. A practical approach to hardening, therefore, requires balancing technical protection with usability, ensuring that the system remains secure while still functional for daily use. At the same time, existing approaches are often fragmented: vulnerability assessment tools tend to focus on identifying issues without providing clear remediation steps, while security frameworks and benchmarks offer recommended configurations but do not provide implementation procedures that can be reproduced easily across machines. This gap contributes to challenges in achieving effective and sustainable hardening practices.

To address this issue, this study integrates the Ferret-Windows open-source host configuration analysis tool with the CIS Microsoft Windows 11 Benchmark to develop a structured and reproducible hardening configuration. The resulting configuration includes both automated and manually applied parameters, allowing it to be adapted to different system environments. Furthermore, the usability of the hardened system is evaluated using the System Usability Scale (SUS) to ensure that increased security does not significantly reduce user experience. This approach positions hardening not only as a technical configuration process but also as part of a broader socio-technical effort to maintain both system protection and operational practicality.

2. METHODS

This research employs a qualitative approach grounded in Design Science Research (DSR) methodology, utilizing primary and secondary data sources to systematically identify, confirm, and refine relevant criteria for evaluating vulnerability assessment tools on Windows-based hosts. The DSR framework can facilitate the structured design, development, validation, and evaluation of practical artifacts, such as comprehensive evaluation criteria. Details of the method used in each step of DSR are shown in Figure 1.

Jurnal Teknik Informatika Vol. 19 No. 1, April 2026 (12-21)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v19i1.46816

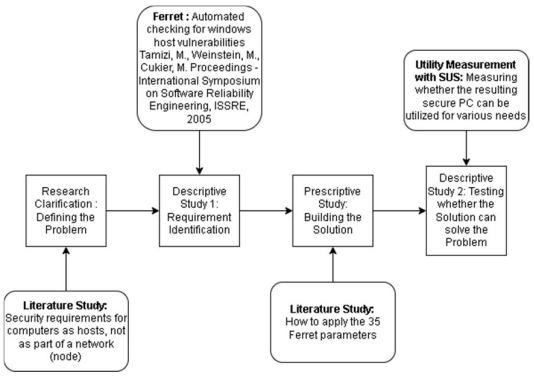

Figure 1. *Research design*

2.1. Data Source

Data for this study were derived from two distinct sources:

1. Primary Data: Collected through expert interviews with cybersecurity professionals possessing at least one year of relevant experience in vulnerability assessments for Windows hosts. A total of six cybersecurity experts were interviewed in this study. The number of participants was determined by data saturation, as no new criteria or themes emerged after the fifth interview. This approach aligns with qualitative research standards, suggesting that data saturation typically occurs with five to ten expert participants when domain knowledge is homogeneous and the research objectives are specific.
2. Secondary Data: Gathered through an extensive literature review comprising scientific publications, peer-reviewed journals, conference proceedings, and existing cybersecurity standards and guidelines related to vulnerability assessment tools

2.2. Data Collection Procedures

The data collection process consisted of two key phases:

1. Literature Review: An initial literature review was conducted to comprehensively identify and document relevant criteria for evaluating vulnerability assessment tools for Windows-based hosts. This phase involved reviewing and analysing published scholarly articles, research reports, cybersecurity guidelines, and international security standards.
2. Expert Interviews: Following the literature review, semi-structured interviews were conducted with subject-matter experts—the interviews aimed at validating the identified criteria and eliciting additional criteria from expert perspectives. Interview questions were meticulously designed based on the preliminary criteria extracted from the literature. Interviews continued until at least two independent experts confirmed each criterion identified through the literature, achieving data saturation and robust validation.

2.3. Data Analysis

Qualitative data from the literature review and expert interviews were analyzed using an open coding approach [9]. This coding method enabled systematic categorization and thematic synthesis of qualitative data, facilitating the identification and refinement of evaluation criteria. Open coding involved iterative reading, labeling, and grouping textual data, enabling the emergence of coherent themes and categories relevant to vulnerability assessment criteria

2.4. Validity and Reliability

Triangulation was employed to ensure the validity and reliability of research findings by integrating multiple data sources and iterative expert validation. Member checking was used to confirm the accuracy of the interpretations with the experts interviewed. Continuous comparison of emerging findings with established criteria from the literature maintained methodological rigor throughout the research process.

2.5. Ethical Considerations

Ethical approval was obtained before data collection, and informed consent was obtained from all participating experts. Confidentiality and anonymity were maintained by anonymizing interview transcripts and securely storing data, accessible only to the researchers involved.

3. RESULTS AND DISCUSSION

This section presents the results of each stage of the DSR framework, described in several subsections that outline the DSR process. This section presents the results of each stage of the DSR framework, described in several subsections that outline the DSR process.

3.1. Research Clarification

Merging security threats targeting physical vulnerabilities in Windows-based computers have revealed persistent weaknesses in system defense, particularly against low-level hardware and peripheral-based attacks. These threats include cold-boot attacks, Direct Memory Access (DMA) attacks, login bypass techniques, and malicious USB exploits. Cold boot attacks remain a critical concern due to the residual data in DRAM after power-off, which can be exploited to retrieve cryptographic keys and other sensitive information, exposing a fundamental flaw in memory volatility assumptions [10], [11]. Similarly, DMA attacks allow adversaries to bypass operating system protections by directly accessing memory through peripherals, making traditional software-based controls ineffective [12]. Physical login bypass methods exploit hardware flaws or insecure software configurations, enabling unauthorized access without requiring credentials [10]. Meanwhile, the use of malicious USB devices has proven to be a reliable vector for executing unauthorized code upon physical connection, resulting in data theft and system compromise [13].

These converging attack vectors highlight a systemic gap in physical-layer security, a dimension often underemphasized in mainstream Windows hardening frameworks. Despite existing countermeasures, no single solution has proven effective in mitigating these threats. This underscores the need for renewed attention to low-level vulnerabilities and the inadequacies of current defensive models in addressing physical access scenarios. As these attacks become increasingly sophisticated, the absence of comprehensive protection at the hardware-software boundary exposes Windows hosts to significant risk.

3.2. Descriptive Study 1

The initial stage of this study involved a comparative analysis of various tools used to assess vulnerabilities in Windows operating systems. The objective was to understand each tool's scope, depth, and limitations in detecting host-level misconfigurations. Among the tools reviewed were Microsoft's Baseline Security Analyzer (MBSA), a group of widely recognized scanners including Nmap, Nessus, and OpenVAS [14], and a specialized academic tool, Ferret-Windows [15].

Most mainstream tools prioritize network- and application-layer vulnerabilities, often leaving host-level configurations, such as insecure registry entries, weak local policies, and residual access permissions underrepresented. Although relatively underutilized in commercial settings, Ferret-Windows offers a modular, extensible framework tailored for granular inspection of Windows host vulnerabilities, including detection and partial remediation. This aligns well with scenarios where fine-grained host hardening is essential, especially in operational environments with elevated physical access risks.

To complement this tool-based comparison, a focus group discussion (FGD) was conducted involving six practitioners from the National Cyber and Crypto Agency of the Republic of Indonesia (BSSN RI). The experts emphasized that the theme of this research is more appropriately referred to as Windows-Based Host Hardening, Host Hardening, or Windows Hardening. These terms more accurately capture the intent of not merely scanning for vulnerabilities but actively addressing and mitigating them through configuration adjustments. They also noted that this research aligns with the CIS Microsoft Windows 11 Benchmark, a recognized baseline for secure configurations in Windows environments.

Critically, the practitioners highlighted that if a study not only assesses but also provides concrete hardening solutions, it fills a substantial gap in current cybersecurity practices. One expert shared that they had independently developed automation scripts for Windows host hardening to support internal operational needs, reinforcing the practical relevance of tool-supported, script-based

ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v19i1.46816

approaches like those introduced in this study. This feedback validates the study's direction and underlines the potential impact of Ferret-Windows as a basis for operational host-hardening frameworks in Indonesian institutions.

3.3. Prescriptive Study

Ferret-Windows is a modular tool that detects and manages vulnerabilities in Windows. The tool works by inspecting local configuration parameters, including registry settings, file access policies, and controls over system features [15]. Meanwhile, the CIS Microsoft Windows 11 Benchmark provides a comprehensive security configuration guide. The guide is based on industry standards and best international practices for creating a secure and controlled Windows system [16], [17].

By combining parameters from Ferret-Windows and CIS Benchmark, researchers compiled 35 security parameters as listed in Table 1. From the analysis results, there are 15 overlapping parameters between the two (parameters No. 1–15), such as Check_Shares (No. 1), GPO_Password_Policy (No. 4), and Reg_WindowsFirewall (No. 9). This conformity shows that Ferret-Windows has adopted the basic principles of system security that are in line with industry standards [13], [15].

Table 1. *Security parameters for Windows 11*

No	Parameter	Ferret	CIS
1	Check_Shares	X	X
2	Check_Shares_GROUPNAME_PERMISSION	X	X
3	FAT_FS	X	X
4	GPO_Password_Policy	X	X
5	GPO_Lockout_Policy	X	X
6	Reg_Drive_Restrict	X	X
7	Reg_No_C-A-D	X	X
8	Reg_AutoPlay	X	X
9	Reg_WindowsFirewall	X	X
10	Windows Blank Passwords	X	X
11	Reg_DisableCMD	X	X
12	Reg_DisableTaskMgr	X	X
13	Reg_DisableRegistryTools	X	X
14	Reg_DisableControlPanel	X	X
15	Reg_DisableSoftwareInstallation	X	X
16	Reg_TSC_Restrict	X	
17	Reg_Port_Redirection	X	
18	Reg_Logons_Cache	X	
19	Reg_Anony_Shares	X	
20	Reg_LocalProfiles	X	
21	Reg_HideDrives	X	
22	Reg_DisableRun	X	
23	Reg_DisableSecurityCenter	X	
24	Reg_DisableWindowsUpdate	X	
25	VNC Blank Passwords	X	
26	Credential Guard		X
27	Device Guard & VBS		X
28	Secure Boot		X
29	AppLocker / SRP		X
30	PowerShell Logging & Transcription		X
31	LAPS		X
32	Exploit Guard		X
33	WDAC		X
34	BitLocker		X
35	Windows Sandbox		X

Ferret-Windows also includes 10 exclusive parameters (parameters No. 16–25), such as Reg_TSC_Restrict (No. 16), Reg_Port_Redirection (No. 17), and VNC Blank Passwords (No. 25). All these parameters highlight Ferret's strength in handling physical and local security aspects, such as Remote Desktop session protection, external device port access restrictions, and prevention of unauthenticated remote access. In contrast, CIS Benchmark tends to be less accommodating in these areas because it focuses on strategic policies and enterprise-level features.

On the other hand, the CIS Benchmark includes 10 additional parameters (No. 26–35) that are not available in Ferret-Windows. Examples include Credential Guard (No. 26), Secure Boot (No. 28), and BitLocker Drive Encryption (No. 34). These features emphasize virtualization-based protection, encryption, and policy control, and are highly relevant for modern system security in enterprise environments.

The combination of the two approaches shows their complementary nature. Ferret-Windows offers high-granularity local configuration inspection and technical flexibility, while CIS Benchmark provides broad policy coverage and is consistent with global standards. The synergy of the two forms is a strong foundation for developing a comprehensive Windows-based host Hardening framework that can be customized to the needs of small and large organizations.

From the integration results between Ferret-Windows and CIS Microsoft Windows 11 Benchmark, 35 security parameters are obtained and divided into three groups. The first

Jurnal Teknik Informatika Vol. 19 No. 1, April 2026 (12-21)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v19i1.46816

group includes 15 overlapping parameters adopted by both approaches, reflecting the compliance of basic security principles such as file sharing access restrictions (Check_Shares), password policies, and account lockouts (GPO_Password_Policy,GPO_Lockout_Policy), and registry settings to turn off risky system functions such as AutoPlay, CMD, and Task Manager. This compliance confirms that Ferret-Windows has integrated basic security principles that align with industry standards.

The second group comprises 10 Ferret-Windows-exclusive parameters focused on local and physical protection. Some include restricting Remote Desktop sessions via Reg_TSC_Restrict, preventing device port redirection via Reg_Port_Redirection, and restricting anonymous sharing access with Reg_Anony_Shares. Other parameters, such as Reg_HideDrives and Reg_DisableRun, limit user interaction with the system, adding a layer of protection against unauthorized access. Ferret-Windows' strength lies in the granularity of local configuration inspection and control over user behavior on the host side.

Meanwhile, the third group includes 10 exclusive parameters from the CIS Benchmark, focused on enterprise policy-based strengthening and modern security features. Examples include Credential Guard and Device Guard, which leverage virtualization to prevent credential theft, and BitLocker, which provides full encryption to protect physical storage drives. In addition, features such as AppLocker, WDAC, and Windows Sandbox provide granular control over application execution and isolated environments for running untrusted software. These parameters offer advanced policy coverage to meet the needs of large-scale organizations with high-risk profiles.

After formulating all the parameters, researchers applied the 35 security configurations to test the Windows system. The implementation approaches were divided into two categories based on technical characteristics: automation via batch scripts and manual configuration.

A total of 25 parameters were automated using batch files (*.bat) containing CMD or PowerShell commands. When running, these files directly adjust the system configuration, especially parameters related to the registry, service restrictions, and local security. Meanwhile, the other 10 parameters could not be fully automated and required manual configuration. Configuration is done through a graphical user interface (GUI) or the system policy console. Some examples include:

- Credential Guard, enabled through the Group Policy Editor, requires a system restart.
- Secure Boot, enabled through BIOS/UEFI and verified through Windows Security Center.
- BitLocker Drive Encryption, configured through the BitLocker control panel, including setting a recovery key.
- AppLocker Policy, set through Local Security Policy, restricts which applications can run.

This combination of implementation methods enables efficient, comprehensive security configuration while maintaining technical precision in areas that do not lend themselves to automation.

To enhance the reproducibility of the automation artifact, the batch scripts were replaced by a robust software artifact. A host-scanning application was developed in Rust using the Tauri framework. This application is designed to programmatically scan system registry settings against the defined baseline and apply the necessary remediations. Figure 2 illustrates a typeset code snippet from the application's file, detailing the remediation logic for applying specific security settings.

Figure 2. *Typeset Code Snippet*

This application-based approach provides superior error handling and a more extensible foundation than the original batch scripts. The complete source code for the scanning and remediation tool, including the artifact, is publicly available for reproducibility and peer review at the following repository: github.com/reswandana00/Hardening-Windows-Operating-System-2025

One noteworthy advantage of using Ferret-Windows lies in its educational applicability. Its open-source nature, modular design, and transparent parameter logic make it

Jurnal Teknik Informatika Vol. 19 No. 1, April 2026 (12-21)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v19i1.46816

an excellent tool for capacity-building in public institutions. IT staff and system administrators can use it as a learning platform to understand low-level system configurations, test their effects, and gradually automate security policies without needing proprietary licenses. This aspect aligns well with the resource constraints in many developing countries and supports digital sovereignty through open tool chains. Ferret-Windows, therefore, is not only a practical tool but also a catalyst for building long-term cybersecurity competence.

3.4. Descriptive Study 2

The system usability was evaluated by involving 31 respondents who were academicians of cybersecurity and all active Windows 11 users for at least 1 year. The respondents had diverse backgrounds and experiences, depending on the length of their Windows use and the type of work they performed using Windows. However, in this test, all respondents were treated equally as end users of the system. After the Windows system was configured with 35 predefined security parameters, the respondents were asked to rate the user experience using the System Usability Scale (SUS). The SUS questionnaire was used to measure perceptions of convenience, ease of use, and consistency of system features after security enhancement. [18].

The usability testing used a standardized hardware setup to ensure consistent evaluation. Researchers prepared a laptop with a 10th-generation Intel Core i5 processor, 16 GB of RAM, 128 GB of SSD v3 storage, and a 13.5-inch display. The computer was installed with a standard Windows 11 24H2 configuration and Microsoft Office 365, which is commonly used for daily office tasks. Following this, 35 predefined hardening parameters were applied to the system. Each respondent was asked to use the configured laptop for approximately 10 minutes, performing everyday computing activities. Immediately after the session, respondents were invited to complete the System Usability Scale (SUS) questionnaire. After each test, researchers verified that all 35 security parameters remained correctly applied to ensure the integrity and consistency of the evaluation across participants.

The measurement results, shown in Table 2, indicate that individual SUS scores range from 47.5 to 100, with an average of 73.63. Based on the standard interpretation by [19] The score is in the “good” category and indicates that the Windows system has been strengthened. With 35 security parameters, it is still considered usable by most users. Although there are minor reports of functional limitations, Users can still complete daily computing activities effectively, such as longer login times due to the deletion of the logon cache (parameter No. 18) and the unavailability of the Control Panel and Run features (parameters No. 14 and No. 22).

These findings indicate that implementing 35 security parameters can function effectively and be used comfortably, as perceived by respondents who tested the system. Thus, this configuration can be considered a balanced approach between increasing security resilience and preserving user experience.

3.5. Discussion

The findings of this study show that integrating technical security approaches derived from open-source tools, such as Ferret-Windows, with international standards, such as the CIS Benchmark, can create a comprehensive hardening framework that is not only technically well-defined but also functionally acceptable to end users. Usability testing shows that, despite some system adjustments, most users can still complete daily computing tasks well. This indicates that strict security configurations need not conflict with usability principles.

Table 2. *SUS Result*

RES	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS
1	4	2	4	2	4	2	4	2	4	1	77,5
2	5	1	5	1	4	1	5	1	4	1	95
3	4	4	3	4	3	3	3	3	4	4	47,5
4	5	1	5	2	5	1	5	1	5	1	97,5
5	3	2	4	4	4	3	4	2	3	4	57,5
6	3	4	4	4	4	4	3	3	3	2	50
7	3	5	5	1	5	3	5	3	5	3	70
8	5	1	5	3	4	1	5	1	4	3	85
9	5	2	5	2	5	1	5	1	5	1	95
10	5	2	5	2	5	2	5	1	5	2	90
11	5	1	5	1	5	1	5	1	5	1	100
12	5	2	5	2	5	1	5	1	4	2	90

13	4	2	4	1	5	1	5	1	5	3	87,5
14	3	4	4	3	5	2	3	2	3	4	57,5
15	3	2	4	2	4	2	4	1	4	2	75
16	4	2	4	3	4	1	4	2	4	2	75
17	5	1	5	4	5	2	5	1	5	5	80
18	4	3	4	3	4	3	5	2	4	4	65
19	3	3	4	2	4	2	4	2	4	2	70
20	4	5	4	5	5	2	5	2	5	4	62,5
21	5	5	5	5	5	5	5	5	5	5	50
22	3	4	3	4	4	3	3	4	4	5	42,5
23	4	3	2	4	4	4	3	2	3	4	47,5
24	4	3	4	2	4	1	4	2	4	4	70
25	5	4	5	5	5	5	5	5	5	5	52,5
26	4	2	4	3	3	4	2	3	4	2	57,5
27	5	1	5	1	5	1	5	1	5	1	100
28	5	3	5	1	4	1	4	1	4	1	87,5
29	4	1	5	1	5	5	5	1	5	1	87,5
30	4	2	4	2	4	2	4	2	4	2	75
31	4	2	4	3	5	1	5	1	5	2	85

However, this study goes beyond demonstrating the effectiveness of configurations; it emphasizes the importance of a contextual, adaptive approach to hardening Windows-based systems. Discussions with practitioners and SUS test results indicate that security policies cannot be applied uniformly without considering the organization's user landscape, operational needs, and technical capabilities. In other words, hardening is not merely compliance with standards, but rather the result of an intelligent compromise between protection and productivity.

Compared with prior studies that used the System Usability Scale to assess hardening frameworks, the average SUS score in this study (73.63) aligns with a 2023 study by Marcilly et al [18], which reported an average score of 70.8 in a usability evaluation of secure alert systems. Another relevant benchmark from Brooke in 1995 [[19] shows that scores above 70 are typically considered “good” and acceptable for end users in general computing environments. These comparisons strengthen the conclusion that a secure Windows configuration, designed with user experience in mind, can achieve both effectiveness and acceptability.

Recent international practices have also highlighted the growing emphasis on usability-centered security configurations. For example, in Germany's BSI (Federal Office for Information Security) guidelines, system hardening is accompanied by user experience validation before deployment in public institutions. Similarly, in Japan’s NISC (National Center of Incident Readiness and Strategy for Cybersecurity), adaptive configuration templates are created based on behavioral analysis of government staff. These examples support the importance of socio-technical alignment in hardening practices. Although they differ in execution, these international practices reinforce the relevance of this study’s focus on balancing configurational security with usability metrics in authentic contexts.

In this context, this study reflects a socio-technical security hardening approach, in which the design of security policies involves not only technical aspects (parameters, scripts, system architecture) but also social dimensions such as user perceptions, usage patterns, and the organization's ability to maintain long-term configurations. This approach is particularly relevant for the public sector, which often faces limited resources but is highly responsible for maintaining the reliability of information systems.

Thus, the approach proposed in this study presents an essential novelty in hardening Windows operating systems. Unlike conventional methods, which are generally limited to vulnerability scanning and reporting, this study emphasizes integrating open-source tools and industry standards to produce security configurations that are effective and proven usable through direct testing with end users. Ferret-Windows, which has been underutilized in practice, shows great potential when combined with formal benchmarks such as CIS. Furthermore, the socio-technical approach adopted broadens the scope of hardening beyond the technical level to encompass part of an adaptive, user-oriented organizational policy architecture. Previous studies on hardening Windows systems in the public sector have not widely explored this approach.

However, some limitations should be noted. This study has not measured the long-term resilience of post-hardening systems to real-world attacks or advanced exploitation

scenarios. Furthermore, usability testing is limited to a single user group within a specific organizational context. Adaptation to other industry contexts, such as critical systems or edge computing devices, may require redeveloping the parameters or adopting a more modular, flexible approach. This study opens space for further research on three dimensions: first, how automation of parameter application can be done through scripts or policy-based frameworks; second, how integration with threat detection systems (EDR, SIEM) can strengthen the effectiveness of this configuration; and third, how this framework can be extended to more specific sectors with different risk profiles. In this way, this Windows-based host hardening approach does not stop at technical configuration but becomes part of a sustainable and evidence-based security policy architecture.

CONCLUSION

This study designs and evaluates a Windows-based system hardening configuration by implementing 35 security parameters through the integration of the Ferret-Windows open-source tool and the CIS Microsoft Windows 11 Benchmark. Usability testing with 31 users yielded an average SUS score of 73.63, indicating that the hardened system can still be used comfortably in daily computing activities. This study develops a usable hardened configuration and introduces a socio-technical hardening approach that integrates technical and social aspects into the design of system security policies.

Further exploratory work could focus on developing a centralized dashboard to monitor hardening status across multiple endpoints in real time. Additionally, integration with Endpoint Detection and Response (EDR) or Security Information and Event Management (SIEM) platforms could enhance the detection of deviations from the applied configurations. A promising direction is creating a modular policy engine that adapts hardening parameters based on organizational risk profiles, user behavior analytics, or specific regulatory requirements. Such extensions would transform static hardening into a dynamic, context-aware security posture.

The main novelty lies in shifting the focus from scan-based risk identification to contextual, configurative mitigation that can be applied adaptively in the public sector. These findings can potentially be the basis for developing a more concrete, modular, and user-oriented national hardening policy. Further research is recommended to automate implementation, test resilience to real threats in production environments, and adjust parameters to the needs of specific sectors such as education, finance, and healthcare.

REFERENCES

[1] B. Fischer, D. Meissner, R. Nyuur, and D. Sarpong, “Cyber-attacks, strategic cyber-foresight, and security,” IEEE Transactions on Engineering Management, vol. 69, no. 6, pp. 3660–3663, Dec. 2022, doi: 10.1109/TEM.2022.3204165.

[2] D. Hendler, S. Kels, and A. Rubin, “AMSI-based detection of malicious PowerShell code using contextual embeddings,” in Proc. 15th ACM Asia Conf. Computer and Communications Security (ASIA CCS), New York, NY, USA, Oct. 2020, pp. 679–693, doi: 10.1145/3320269.3384742.

[3] A. B. Ajmal, M. A. Shah, C. Maple, M. N. Asghar, and S. U. Islam, “Offensive security: Towards proactive threat hunting via adversary emulation,” IEEE Access, vol. 9, pp. 126023–126033, 2021, doi: 10.1109/ACCESS.2021.3104260.

[4] V. Duta, C. Giuffrida, H. Bos, and E. V. D. Kouwe, “PIBE: Practical kernel control-flow hardening with profile-guided indirect branch elimination,” in Proc. ASPLOS, New York, NY, USA, Apr. 2021, pp. 743–757, doi: 10.1145/3445814.3446740.

[5] N. Mohamed, “Study of bypassing Microsoft Windows security using the MITRE CALDERA framework,” F1000Research, vol. 11, p. 422, Apr. 2022, doi: 10.12688/f1000research.109148.

[6] S. Akter, K. Khalil, and M. Bayoumi, “A survey on hardware security: Current trends and challenges,” IEEE Access, vol. 11, pp. 77543–77565, 2023, doi: 10.1109/ACCESS.2023.3288696.

[7] U. Lee and C. Park, "SofTEE: Software-based trusted execution environment for user applications," IEEE Access, vol. 8, pp. 121874–121888, 2020, doi: 10.1109/ACCESS.2020.3006703.

[8] P. L. Gorski, L. L. Iacono, and M. Smith, "Eight lightweight usable security principles for developers," IEEE Security & Privacy, vol. 21, no. 1, pp. 20–26, Jan. 2023, doi: 10.1109/MSEC.2022.3205484.

[9] J. Saldaña, *The Coding Manual for Qualitative Researchers*, 3rd ed. London, U.K.: SAGE Publications, 2016.

[10] J. Mahmod and M. Hicks, "UnTrustZone: Systematic accelerated aging to expose on-chip secrets," in Proc. IEEE Symp. Security and Privacy (SP), 2024, pp. 4107–4124, doi: 10.1109/SP54263.2024.00069.

[11] J. A. Halderman et al., "Lest we remember: Cold-boot attacks on encryption keys," Communications of the ACM, vol. 52, no. 5, pp. 91–98, 2009, doi: 10.1145/1506409.1506429.

[12] P. Stewin, "A primitive for revealing stealthy peripheral-based attacks on the computing platform's main memory," in *Research in Attacks, Intrusions, and Defenses*, Berlin, Germany: Springer, 2013, pp. 1–20, doi: 10.1007/978-3-642-41284-4_1.

[13] Himanshu, M. Sharma, and G. Sujatha, "Enhanced Windows fuzzy firewall for DoS attack prevention," in Proc. ICERCS, 2023, pp. 1–4, doi: 10.1109/ICERCS57948.2023.10434091.

[14] I. Odun-Ayo et al., "Comparative review of vulnerability analysis tools," in Proc. SEB4SDG, 2024, pp. 1–6, doi: 10.1109/SEB4SDG60871.2024.10630138.

[15] [M. Tamizi, M. Weinstein, and M. Cukier, "Automated checking for Windows host vulnerabilities," in Proc. ISSRE, 2005, pp. 139–148, doi: 10.1109/ISSRE.2005.11.

[16] P. H. Ambika and G. Sujatha, "System hardening using CIS benchmarks," in Proc. ACCAI, 2024, pp. 1–6, doi: 10.1109/ACCAI61061.2024.10602274.

[17] Center for Internet Security, *CIS Microsoft Windows 11 Stand-alone Benchmark v3.0.0*, Center for Internet Security, 2024. [Online]. Available: https://www.cisecurity.org/benchmark/microsoft_windows/

[18] R. Marcilly et al., "Comparison of the validity, perceived usefulness, and usability of I-MeDeSA and TEMAS," International Journal of Medical Informatics, vol. 175, p. 105091, Jul. 2023, doi: 10.1016/j.ijmedinf.2023.105091.

[19] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, and I. L. McClelland, Eds. London, U.K.: Taylor & Francis, 1996, pp. 207–212.