RS-015-aligned Windows Kiosk Wrapper for Secure Operation of Engineering Toolchains

Tae-Gyu Kang [a*], Ki-Ho Cho [a], Dong-Yeon Lee [a]
[a]*SOOSAN ENS Co., Research Institute, 2F, Soosan Building, 13, Bamgogae-ro 5-gil, Gangnam-gu, Seoul, 06367*
[*]*Corresponding author: hellkalsan@soosan.co.kr*

Keywords : KINAC/RS-015, kiosk mode, Windows Job Objects, conditional cmd.exe allowance, forensic logging

1. Introduction

Engineering workstations used for developing and maintaining digital instrumentation and control (I&C) components often require complex toolchains that spawn multiple dependent processes (e.g., batch scripts, hardware servers, and temporary console hosts). Conventional kiosk products typically mitigate abuse by globally blocking shells and system shortcuts, which can unintentionally break legitimate build flows. In nuclear environments, cyber security controls must additionally satisfy regulatory expectations such as KINAC/RS-015, which specifies a comprehensive set of cyber security requirements for computer and information systems in nuclear facilities.

This paper presents a policy-enforcing Windows kiosk wrapper that balances usability and security for engineering toolchains by combining (i) hierarchical process governance using Windows Job Objects, (ii) least-privilege conditional cmd.exe allowance based on parent process, arguments, and resolved paths, (iii) a dual-layer enforcement model (event-driven and periodic polling mechanisms) to mitigate common process-based bypass attempts, and (iv) encrypted forensic logging and availability monitoring via an external sentinel process that restarts the wrapper if unexpectedly terminated. In the current implementation, the wrapper and sentinel mutually monitor each other to improve resilience against accidental or intentional termination. We further analyze how these mechanisms map to RS-015 technical security control families and identify residual gaps and compensating measures.

2. Regulatory Context and Threat Model

KINAC/RS-015 is a domestic regulatory technical standard used for cybersecurity reviews of computer and information systems associated with nuclear facilities. Prior studies report that RS-015 structures controls into major families (e.g., access control, audit and accountability, system and communications protection, identification and authentication, and system hardening) and includes approximately 100 detailed requirements for essential digital assets. Regulatory R&D has also produced checklists and acceptance criteria to support consistent evaluation of RS-015 technical security controls during review and inspection.

In this work, the target environment is an engineering kiosk workstation that must run a trusted EDA toolchain (e.g., Vivado) for device configuration and testing in a controlled setting. The assumed attacker is a local, non-administrative user who can interact with keyboard/mouse and attempt to (a) spawn an interactive shell (cmd.exe/PowerShell), (b) launch browsers or utilities (taskmgr, regedit), (c) pivot via network paths or unauthorized peripherals, and (d) leave residual processes after the kiosk session. Kernel-level attacks, BIOS/boot compromise, and privileged administrator misuse are out of scope and should be mitigated via organizational controls (account separation, physical security, secure boot, and endpoint protection).

3. System Overview

The implementation is based on .NET WPF with an MVVM presentation layer and a modular service layer. Core services include a JobObjectService for lifecycle control, a SecurityWatchdogService for high-frequency scanning, an event-driven process guard for near-real-time blocking, hook services for input mediation, and policy services for process allowance, network rules, and device hardening. Fig. 1 provides the system block diagram across UI, service layer, and Windows kernel APIs.

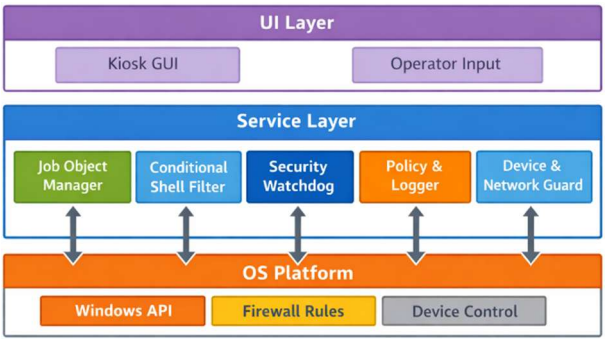

Fig. 1. System architecture (UI layer, service layer, and OS control layer).

4. RS-015-aligned Technical Controls and Implementation

RS-015 emphasizes defense-in-depth and expects technical controls to protect essential and supporting digital assets. Although the proposed wrapper does not replace plant-wide cybersecurity programs, it provides a concrete enforcement layer at the engineering workstation boundary.

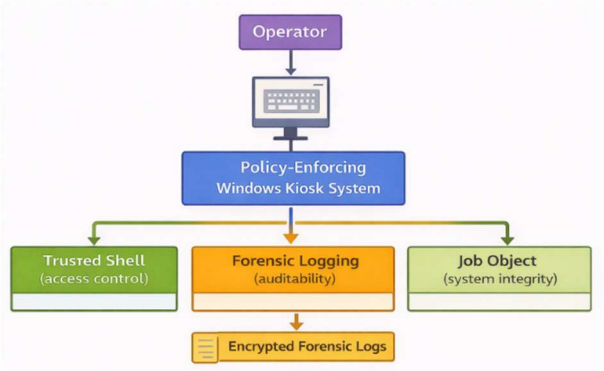

Fig. 2. RS-015-aligned control architecture of the policy-enforcing Windows kiosk wrapper.

Fig. 2 presents the high-level control architecture aligned with representative RS-015 technical control families. At the workstation boundary, the system integrates three primary enforcement domains:

(i) trusted shell policy for access control,
(ii) forensic logging for auditability and traceability, and
(iii) Job Object–based process containment for system integrity.

Encrypted forensic logs provide persistent evidence of runtime policy decisions and support accountability during compliance reviews.

4.1 Hierarchical process governance using Job Objects

A primary RS-015 concern is preventing unauthorized process persistence and ensuring controlled cleanup of session artifacts. The wrapper creates a dedicated Windows Job object with JOB_OBJECT_LIMIT_KILL_ON_JOB_CLOSE enabled and assigns the launcher and all detected toolchain processes to the Job.
When the kiosk session ends or the wrapper exits, Windows terminates processes associated with the Job when the Job handle is closed (under Windows Job semantics). This mechanism reduces residual shells and orphan processes while providing deterministic lifecycle management of the toolchain process tree.

4.2 Smart launch sequence to reduce false positives

Engineering toolchains frequently spawn transient helper processes during initialization. Naive enforcement mechanisms may misclassify such processes as policy violations and prematurely terminate legitimate workflows.
To address this issue, the wrapper implements a smart launch sequence designed to balance startup flexibility with strict containment. The procedure is summarized in Algorithm 1. In the current implementation, the timeout for target process binding is configurable (e.g., 60 seconds for PID detection), with extended waiting for window readiness where required by the toolchain initialization sequence.

Algorithm 1: Smart Launch and Job Binding (Simplified)
1: Begin controlled session (timeout policy)
2: Create Job (kill-on-close limit)
3: Pause Guard and Watchdog
4: Start launcher (cmd /c for .bat/.cmd)
5: Assign launcher to Job immediately
6: Poll for target process within Tmax
7: Assign target and key children to Job
8: Resume Guard and Watchdog
9: Enforce kiosk window constraints until exit
10: Dispose Job (terminates processes associated with the Job)

The temporary suspension of enforcement prevents false positives during controlled initialization, while early Job assignment ensures that child processes remain contained. Once the target process is bound, full enforcement resumes.

4.3 Conditional cmd.exe allowance (least privilege)

RS-015 emphasizes strict control over command interpreters and administrative utilities. However, many engineering toolchains rely on cmd.exe to execute vendor-provided environment scripts.
Rather than globally allowing or blocking cmd.exe, the wrapper validates each invocation according to a least-privilege policy.

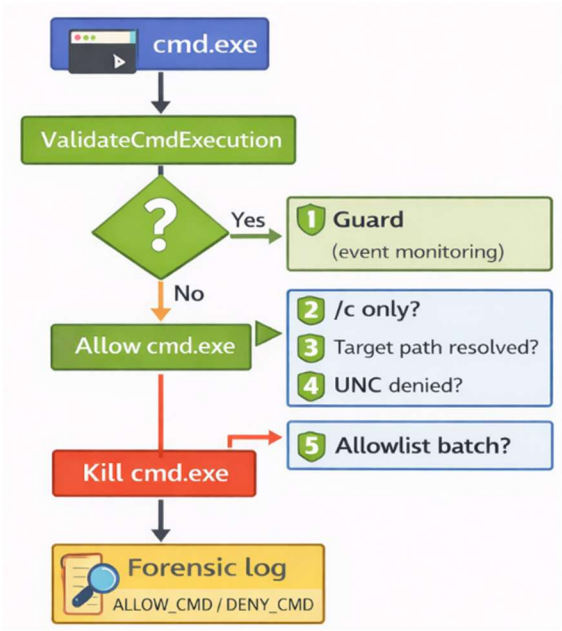

Fig. 3. Conditional cmd.exe execution decision process.

Fig. 3 illustrates the runtime decision flow used to evaluate each cmd.exe invocation. The wrapper validates the parent process, the `/C`-only constraint, path resolution status, UNC-path denial, and allowlisted batch-file membership before deciding whether to allow or terminate `cmd.exe`. The corresponding procedural logic is formalized in Algorithm 2.

Algorithm 2: Conditional cmd Allowance (Simplified)
1: if parent not in AllowedParents: deny
2: if "/k" present: deny
3: if "/c" missing: deny
4: extract first script path after /c
5: expand environment variables
6: if relative path: attempt resolution in trusted directories
7: if still relative: deny
8: if UNC path: deny
9: if filename not in AllowedBatchFiles: deny
10: allow and emit audit record

When any condition fails, execution is denied and a structured audit record containing a decision reason code is generated. The implementation also supports a configurable soft-allow (learning) mode for certain relative-path scenarios under controlled conditions, enabling gradual policy refinement without fully relaxing enforcement constraints.

4.4 Dual-layer enforcement and low-privilege process inspection

To reduce bypass opportunities, the wrapper employs a dual-layer enforcement architecture combining event-driven interception and periodic policy revalidation.

The event layer subscribes to process creation notifications (e.g., Win32_ProcessStartTrace) and can immediately terminate explicitly forbidden processes. The watchdog scans active processes at 250 ms intervals and re-evaluates policy decisions.

For robust identification under restricted permissions, the watchdog uses Win32 APIs such as OpenProcess(PROCESS_QUERY_LIMITED_INFORMATION) and QueryFullProcessImageName.

4.5 Runtime Enforcement Visibility and Audit Evidence

Interactive consoles and system navigation shortcuts represent potential escape vectors in kiosk environments. To make runtime enforcement observable and auditable, the wrapper continuously records enforcement actions in structured audit logs. Fig. 4 presents an example log excerpt showing process detection, policy evaluation, and decision reason codes generated during runtime.

During runtime, policy-relevant events such as process creation detection, conditional cmd.exe validation, Job assignment, process termination, and maintenance-mode transitions are logged with timestamps and structured reason codes. Both event-driven guard actions and periodic watchdog revalidations are recorded, allowing reconstruction of the enforcement decision sequence during the session.

Each denied execution attempt produces a decision trace that includes the evaluated parent process, command-line arguments, path resolution results, and the final policy outcome. Allowed executions are similarly logged to demonstrate consistent and traceable enforcement behavior.

Encrypted storage of these logs provides confidentiality at the workstation boundary and reduces the risk of casual tampering or unintended disclosure while supporting post-incident analysis and compliance verification.

As shown in Fig. 4, the log records both allowed and denied actions with timestamps and structured reason codes, making the enforcement sequence easier to trace and interpret.

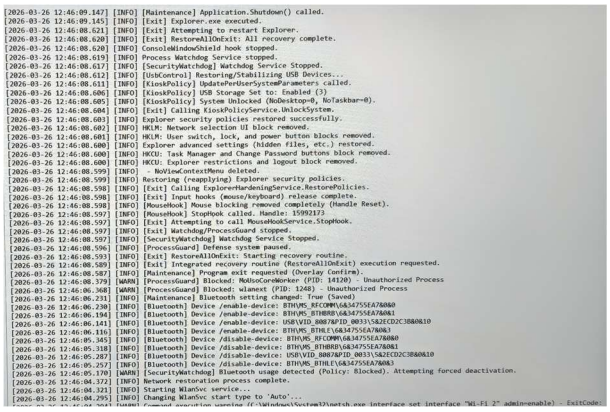

Fig. 4. Example of a runtime enforcement log illustrating process detection, policy evaluation, and structured decision reason codes.

4.6 Forensic logging and accountability

RS-015 includes requirements related to auditability and accountability of security-relevant activities. Beyond simple event recording, the proposed logging mechanism provides accountability evidence that can support post-incident analysis, internal review, and compliance verification. Because enforcement decisions are preserved with timestamps and structured reason codes, reviewers can reconstruct how the wrapper evaluated and handled security-relevant actions during a kiosk session.

The logs are preserved in encrypted form at the workstation boundary to reduce the risk of casual tampering or disclosure. Considerations regarding cryptographic key management and strengthening measures are discussed in Section 5.

5. Discussion and Limitations

Coverage with respect to RS-015 is strongest for technical controls implemented at the workstation boundary, including execution control, audit logging, and basic communications restriction.

However, several RS-015 requirement domains—such as personnel security, physical protection, supply-chain assurance, and comprehensive identity and authentication management—require organizational or infrastructure-level measures and cannot be satisfied solely by a user-mode software wrapper. In addition, because the proposed mechanism operates in user mode, it cannot independently defend against kernel-level bypass attacks or other privileged tampering scenarios.

From a security engineering perspective, polling-based enforcement introduces a bounded time-of-check/time-of-use window. Although the dual-layer architecture reduces exposure by combining event-driven interception with periodic revalidation, a small residual bypass window still remains and race conditions cannot be eliminated entirely in the current implementation.

User-mode input hooks can restrict common navigation shortcuts; however, secure attention sequences (e.g., Ctrl+Alt+Del) remain under operating system control and cannot be intercepted by user-mode mechanisms. Therefore, OS-level configuration (account lockdown, shell replacement, group policy enforcement, and endpoint protection) should be applied in conjunction with the proposed wrapper.

For encrypted logging, the current implementation can be further strengthened by integrating DPAPI- or TPM-backed key storage, per-record random IV generation, and key rotation policies to better align with stringent regulatory expectations.

6. Conclusions

This paper presented a policy-enforcing Windows kiosk wrapper designed to support secure operation of engineering toolchains in regulated nuclear environments.

Unlike conventional kiosk solutions that rely on coarse-grained blocking, the proposed approach integrates Job Object–based process containment, conditional shell validation, dual-layer enforcement, and encrypted forensic logging to balance usability and security at the workstation boundary.

By aligning core enforcement mechanisms with representative RS-015 technical control families, the wrapper provides a practical and implementable security layer while acknowledging residual risks that require complementary organizational and OS-level safeguards.

Ongoing validation demonstrates stable operation under representative engineering workloads and consistent enforcement behavior against misuse attempts. However, because the proposed system operates in user mode, it has inherent limitations in mitigating kernel-level bypass attacks. In addition, the polling-based inspection mechanism introduces a bounded time-of-check to time-of-use (TOCTOU) window, leaving limited residual race-condition exposure. Future work will focus on addressing these limitations by integrating enhanced telemetry mechanisms, strengthening cryptographic key management, and extending audit capabilities for compliance reporting.

References

[1] Korea Institute of Nuclear Nonproliferation and Control (KINAC), “RS-015: Cyber Security of Computers and Information Systems for Nuclear Facilities,” 2013.
[2] N. Y. Kim et al., “Implementation Study of Technical Security Controls from KINAC/RS-015 for Nuclear Facilities,” Journal of Information Security, vol. 27, no. 2, 2017.
[3] J. G. Song, J. S. Shin, J. W. Lee, C. K. Lee, and J. G. Choi, “A Study on Cybersecurity Function Conformance Test Methodology for Digital Controllers in Nuclear Power Plants,” Journal of The Korea Institute of Information Security and Cryptology, vol. 29, no. 6, pp. 1297-1309, Dec. 2019.
[4] Nuclear Safety and Security Commission (NSSC), “Development of a Guide for Regulatory Acceptance Criteria of Technical Security Controls (RS-015),” 2023.
[5] Microsoft, “Job Objects,” Microsoft Learn.
[6] Microsoft, “SetWinEventHook function,” Microsoft Learn.
[7] Microsoft, “QueryFullProcessImageName function,” Microsoft Learn.
[8] Microsoft, “Win32_ProcessStartTrace event class,” Microsoft Learn.