KESO: A KERNEL ENFORCED SECURED OPERATIONS

Vaishali Rane[1], Kaustubh Shinde[2], Daksh Jain[3], Hrishikesh Gupta[4]

[1]Head of Computer Engineering, [2]Student, [3]Student, [4]Student

Department of Computer Engineering

Thakur Polytechnic, Kandivali East, Mumbai-400101, India

vaishali.co@tpoly.in, kaustubhshinde010@gmail.com, dsjain1102@gmail.com, hrishikeshshradgupta59@gmail.com

Abstract : Modern operating systems mainly depend on configurable security mechanisms such as firewalls, antivirus tools, and user-managed access controls. These mechanisms often assume correct administrative behavior and become ineffective once an attacker gains privileged access. This paper presents KESO, a Kernel-Enforced Secure Operating System, designed to provide mandatory and non-bypassable protection at the kernel level. The proposed system enforces process-aware network control, continuously monitors system behavior for anomalies, applies adaptive in-place data protection during active attacks, deploys deception-based defenses using decoy resources, and preserves tamper-resistant audit logs through cryptographic protection and permissioned blockchain anchoring. The architecture is intended to improve resistance against insider threats, post-compromise abuse, unauthorized data exfiltration, and audit-log manipulation. KESO offers a security-first operating system model that strengthens both operational protection and forensic reliability.

Keywords - Operating System Security, Kernel Security, Mandatory Access Control, Process-Aware Network Control, Blockchain Logging, Deception Defense, Secure Systems.

INTRODUCTION

Operating systems form the core trust boundary of modern computing platforms. However, most current systems rely heavily on optional and configurable security mechanisms. Firewalls, antivirus tools, file permissions, and log protection are often dependent on users or administrators for correct deployment and maintenance. In practice, these assumptions fail during real cyberattacks. Once an attacker gains user-level or administrative-level access, many traditional protections can be bypassed, disabled, or misused.

This limitation creates a serious problem for systems that handle sensitive information. Data can be stolen, internal misuse can go undetected, and security logs may be altered or deleted. As a result, post-compromise investigation becomes unreliable. To address this gap, security must be enforced at a deeper and more trusted layer.

This paper presents KESO, a Kernel-Enforced Secure Operating System. The proposed system enforces critical protection mechanisms directly within the kernel so that they cannot be bypassed, even by privileged users. The system combines mandatory kernel-level security, process-aware network restrictions, adaptive data protection, deception-based defenses, and tamper-resistant logging. The main goal is to create an operating system that remains resilient even during active attacks and preserves trustworthy forensic evidence.

OBJECTIVE

The objective of this project is to design a security-first operating system that enforces non-bypassable kernel-level protection, prevents unauthorized network and data access, protects sensitive information during active attacks, and preserves tamper-resistant audit logs to ensure reliable security and forensic integrity.

PROBLEM STATEMENT

Most operating systems rely on configurable security features such as firewalls, access permissions, and system logs. These features assume that administrators and users will manage them correctly. In real-world attacks, once an attacker gains user or administrator access, many security controls can be bypassed. Malware can misuse network access, sensitive data can be stolen, and security logs can be deleted or modified.

Current systems therefore provide weak protection after partial compromise and cannot guarantee trustworthy forensic evidence. This makes systems vulnerable to insider misuse, unauthorized communication, data breaches, and loss of reliable audit trails. A stronger approach is required in which security is mandatory and enforced directly by the operating system kernel.

RELATED SECURITY PERSPECTIVE

Traditional endpoint security is usually layered above the operating system core. Although such tools provide useful protection, they are still vulnerable to privilege abuse and configuration weaknesses. Similarly, standard logging systems are often stored locally and can be modified by an attacker after compromise.

The proposed KESO approach is based on the idea that the kernel should become the central enforcement point for security decisions. Instead of leaving protection to optional user-space software, the operating system itself should define and enforce security boundaries. In this way, both prevention and post-incident accountability are improved.

PROPOSED SYSTEM

KESO is designed as a security-first operating system where core protections are enforced automatically at the kernel level. Security mechanisms in the proposed design cannot be disabled or bypassed even by privileged users. This provides stronger protection in hostile or partially compromised environments.

The proposed system includes the following major components:

A. Kernel-Enforced Mandatory Security

Every process, system call, file access request, and privilege-sensitive action is evaluated according to mandatory kernel-level policies. This removes dependence on discretionary user behavior and makes policy enforcement non-bypassable.

B. Process-Aware Network Control

Network communication is tied to process identity. Only explicitly authorized processes are allowed to send or receive data through the network stack. Unauthorized or suspicious communication is blocked before data transmission occurs.

C. Continuous Behavior Monitoring

The kernel continuously observes system behavior, including system call usage, file access patterns, memory behavior, and network actions. This allows the operating system to detect suspicious or anomalous activities in real time.

D. Adaptive In-Place Data Protection

When abnormal behavior is detected, the operating system immediately protects sensitive files. Protection may include temporary encryption, access restriction, or file locking. This reduces the chance of data theft during an active compromise.

E. Deception-Based Defense

The system generates realistic decoy files and services to confuse and delay attackers. These resources resemble legitimate system assets and are intended to redirect malicious activity away from genuine data.

F. Tamper-Resistant Audit Logging

Important security events are digitally signed and recorded using a secure logging framework. Log integrity is additionally verified through a permissioned blockchain, ensuring that deletion or modification attempts can be detected.

TECHNICAL ARCHITECTURE

The overall architecture of KESO follows a kernel-centered trust model. User applications operate in user space, while security-critical decisions are made in kernel space. The system call interface acts as the transition layer through which application behavior is mediated. The kernel then interacts with hardware resources such as CPU, memory, and devices while enforcing mandatory protection boundaries.

This architecture ensures that application actions are always evaluated before access to critical resources is granted. By placing enforcement in the kernel, KESO minimizes the attack surface associated with optional user-space protection tools.

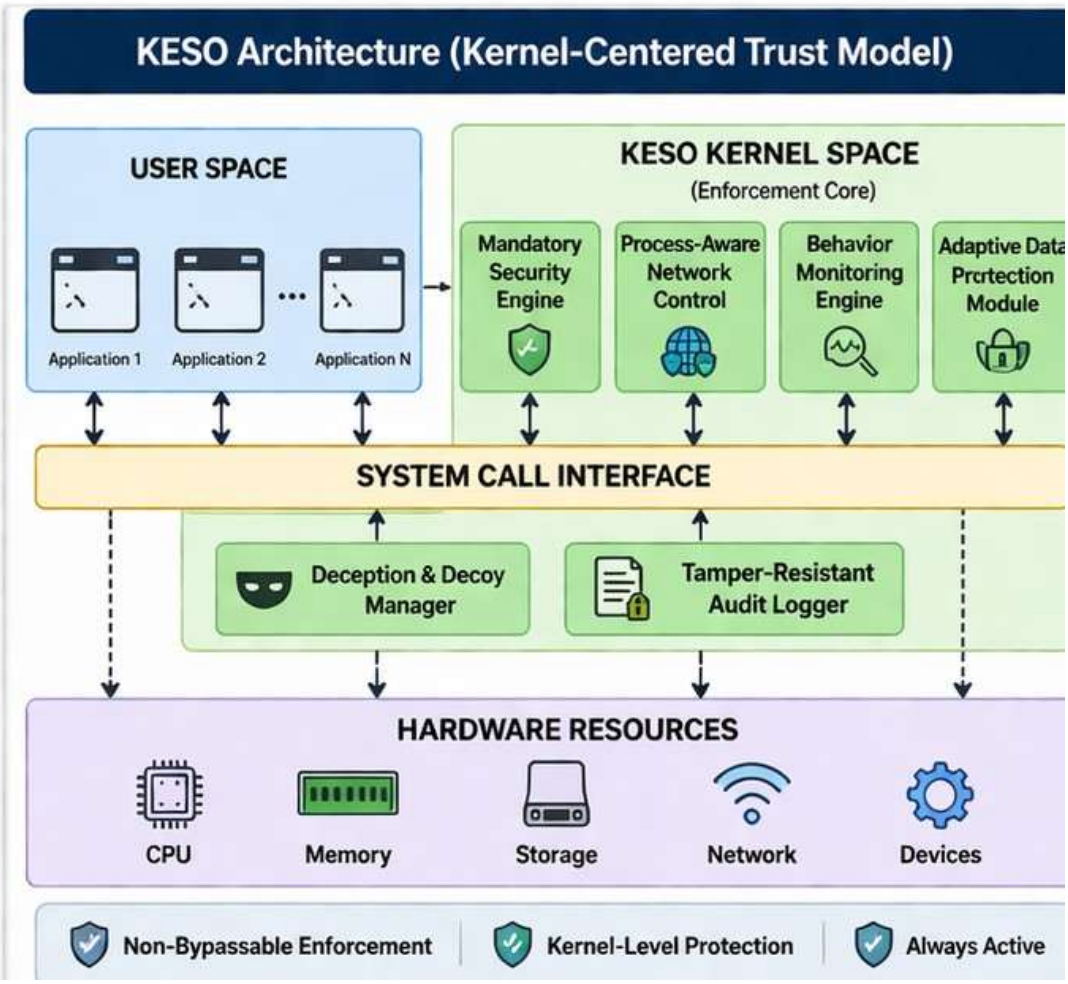

Figure 1: Kernel-centered technical architecture of KESO.

METHODOLOGY

The project follows a security-first operating system methodology in which critical protections are enforced inside the kernel. The system begins with an immutable and verified core, allowing only cryptographically signed kernel and operating system images to boot. This prevents unauthorized modification of the trusted computing base.

A mandatory kernel-level security framework is then used to control process behavior, file access, memory usage, and system calls. Process-aware network control is integrated into the kernel so that each application receives network access only through explicit authorization policies.

The kernel continuously monitors behavior to detect anomalies such as abnormal system calls, suspicious file access, privilege misuse, and unexpected network activity. When such behavior is identified, adaptive in-place data protection is activated. Sensitive files are encrypted or access-restricted in real time. In addition, deception mechanisms generate realistic decoy files and services to divert attackers. Important security events are digitally signed and recorded in a secure log system whose integrity is verified using a permissioned blockchain.

WORKING MECHANISM

The system starts with a secure boot mechanism in which only cryptographically verified operating system and kernel images are allowed to load. This ensures that the core platform remains trusted before execution begins.

Once the system is active, all security controls are enforced directly by the kernel and cannot be disabled by users or administrators. Each application and process is monitored continuously. Before any process accesses files, memory, system resources, or network channels, the kernel checks mandatory security policies.

Network communication is handled in a process-aware manner. Only approved processes are allowed to send or receive data. Unauthorized requests are blocked prior to transmission. Simultaneously, the kernel observes system call activity, privilege usage, network patterns, and file access behavior.

If suspicious activity is detected, the system immediately activates protective actions. Sensitive files are encrypted or temporarily locked to reduce data exposure. At the same time, realistic decoy resources are deployed to attract attacker interaction. All important security events are digitally signed and recorded, and log integrity is reinforced through blockchain-based verification.

Figure 2: Conceptual system flow of KESO.

SYSTEM FLOW

The conceptual system flow of KESO can be summarized as follows:

1. Verified boot loads only trusted kernel and operating system images.
2. User applications execute in user space.
3. Requests cross the system call interface into the kernel.
4. The kernel checks mandatory security policies.
5. Approved operations access CPU, memory, devices, files, and network resources.
6. Suspicious behavior triggers adaptive data protection and deception mechanisms.
7. Security events are digitally signed and recorded in tamper-resistant logs.

KEY FEATURES

The proposed system includes the following major features:

- Security is enforced directly by the operating system kernel and cannot be turned off, even by administrators.
- Each application receives limited and explicit network access.
- Sensitive data is protected automatically during active attacks.
- Decoy files and deceptive services confuse and delay attackers.
- Security logs are preserved using tamper-resistant cryptographic and blockchain-based mechanisms.

EXPECTED OUTCOMES

The expected result of this project is a secure operating system that continues protecting data even after partial compromise. It is intended to reduce unauthorized network activity, prevent data theft during attacks, mislead attackers using decoy data, and ensure reliable tamper-proof security logs for investigation and compliance.

BLUEPRINT AND DESIGN ASPECT

The main design principle of KESO is that security must be enforced directly at the kernel level rather than relying on user or administrator settings. All controls are mandatory and remain active even for privileged users. The system assumes that attacks can happen at any time, including insider misuse or partial compromise.

Network access is strictly controlled for each application. Sensitive data is protected automatically during suspicious activity without requiring user intervention. In addition, logs are designed to remain tamper-resistant so that important evidence remains reliable for investigation and compliance purposes.

IMPACT AND CLASSIFICATION

KESO belongs to the category of secure operating system design and advanced cybersecurity architecture. Its impact lies in shifting trust away from optional security tools and toward operating system-level enforcement. The proposed design improves resilience against post-compromise abuse and supports stronger digital forensics.

A. Value Addition

The following value additions are provided by the proposed system:

- Security is enforced at the kernel level and cannot be disabled.
- Data remains protected even after partial compromise.
- Unauthorized network access by applications is prevented.
- Sensitive files are secured automatically during attacks.
- Decoy data is used to confuse and delay attackers.
- Tamper-proof audit logs are preserved using blockchain-based verification.
- Reliability of forensic investigation and compliance is improved.

Figure 3: Secure blockchain-backed audit logging and verification model.

APPLICATION AREAS

The proposed secure operating system can be applied in the following domains:

- Enterprise and corporate computer systems
- Government and defense organizations
- Research and development laboratories
- Financial institutions and banking systems
- Secure workstations for sensitive operations
- Data centers and cloud infrastructure

SOCIAL IMPACT

The proposed solution significantly improves system security by enforcing protection directly at the operating system kernel level. It reduces the risk of data breaches by preventing unauthorized network access and protecting sensitive data during active attacks. Deception techniques delay attackers and limit damage. Tamper-resistant audit logging ensures reliable evidence for investigation and compliance. As a result, the system increases trust in digital infrastructure and reduces the impact of insider threats in sensitive environments.

COMMERCIAL VIABILITY

The proposed secure operating system has strong commercial potential due to increasing demand for advanced cybersecurity solutions. Enterprises, government agencies, defense sectors, financial institutions, and research laboratories all require systems that can protect sensitive data even after cyberattacks occur.

KESO can be commercialized in multiple forms, including as a secure operating system platform, a hardened enterprise workstation environment, or a licensed operating system for critical and high-risk environments. It may also be offered as a premium security product for institutions with strict compliance and forensic requirements.

ADVANTAGES

The proposed system offers several advantages over conventional operating system security approaches:

- Non-bypassable enforcement through kernel-level design
- Reduced dependence on user and administrator behavior
- Real-time protection during active attacks
- Improved detection of abnormal activity
- Better forensic trustworthiness through secure logs
- Stronger resilience against insider threats and post-compromise abuse

LIMITATIONS AND FUTURE SCOPE

Although the design is strong conceptually, practical implementation may involve significant engineering challenges. Kernel-level enforcement requires careful design to avoid performance overhead and compatibility problems. Real-time anomaly detection also requires accurate policy tuning to minimize false positives. Blockchain-backed audit logging must be integrated efficiently so that it does not become a bottleneck.

Future work may include implementing KESO using Linux Security Modules, eBPF-based monitoring, or a custom hardened kernel prototype. Additional research can also focus on policy learning, lightweight cryptographic log verification, and automated deception management.

RESULTS AND DISCUSSION

The proposed KESO model provides a strong conceptual security framework for protecting modern computing systems against insider threats, post-compromise abuse, unauthorized data access, and audit-log tampering. By enforcing controls directly in the kernel, the design strengthens system trust beyond what optional user-space security tools can provide.

The architecture also improves operational visibility and forensic reliability. Process-aware network filtering, adaptive data protection, and deception-based defense reduce attack effectiveness during active compromise. In addition, blockchain-backed audit integrity helps preserve trustworthy evidence for investigation and compliance. Overall, the model demonstrates clear advantages for environments where confidentiality, reliability, and non-bypassable protection are critical.

ABBREVIATIONS

- KESO – Kernel-Enforced Secure Operating System
- CPU – Central Processing Unit
- eBPF – Extended Berkeley Packet Filter

CONCLUSION

KESO presents a security-first operating system architecture in which critical protections are enforced directly by the kernel. Unlike conventional systems that depend on optional or configurable security tools, the proposed design ensures that security remains mandatory and non-bypassable. By combining kernel-enforced policy control, process-aware networking, adaptive data protection, deception-based defense, and tamper-resistant blockchain-backed logging, KESO improves both operational resilience and forensic integrity. The proposed model is especially relevant for environments where trust, confidentiality, and reliable audit evidence are essential.

Acknowledgment

The authors would like to express their sincere gratitude to Ms. Vaishali Rane for her valuable guidance, continuous support, and encouragement throughout the development of this project. Her insights and suggestions played a crucial role in shaping the design and implementation of KESO.

References

[1] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ, USA: Wiley, 2018.

[2] W. Stallings, *Operating Systems: Internals and Design Principles*, 9th ed. London, U.K.: Pearson, 2018.

[3] T. Jaeger, *Operating System Security*. San Rafael, CA, USA: Morgan & Claypool, 2008.

[4] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Boston, MA, USA: Addison-Wesley, 2007.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[6] L. Yuan, J. Caballero, and C. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Proc. IEEE Symp. Security and Privacy*, 2006, pp. 226–241.

[7] P. Loscocco and S. Smalley, "Integrating flexible support for security policies into the Linux operating system," in *Proc. FREENIX Track USENIX Annual Technical Conf.*, 2001, pp. 29–42.