Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Universal Serial Bus Physical Security System

Mr. S. Venkatesh
Assistant Professor, Department of Cyber Security
Sreenidhi Institute of Science and Technology, Ghatkesar

T. Vishnuvardhan
22311A6246
Department of Cyber Security Sreenidhi Institute of Science and Technology

V. Nikhil
22311A6251
Department of Cyber Security Sreenidhi Institute of Science and Technology

Nampally Shyni
22311A6233
Department of Cyber Security Sreenidhi Institute of Science and Technology

***Abstract* - USB devices represent a critical security vulnerability in modern computing environments, serving as primary vectors for malware transmission and unauthorized data exfiltration. Traditional security approaches rely on signature-based detection, which fails against zero-day exploits and polymorphic threats. This project presents USB Physical Security System v2.0, a comprehensive desktop application that combines user authentication, multi-method malware detection, USB port control, and behavioral monitoring to protect against data theft and malware infections. The system implements SHA-256 password hashing with account lockout protection, detects 16+ dangerous file types through extension-based and hash-based signature matching, enforces role-based access control, and maintains comprehensive audit trails of all security-relevant activities. Built entirely with Python standard library (zero external dependencies), the application provides a Tkinter GUI, Windows Registry-based USB control, real-time email alerts via Gmail SMTP, and cross-platform support. Performance testing demonstrates rapid scanning (2–5 seconds for 100 files), strong threat detection accuracy, and reliable authentication mechanisms. The modular architecture ensures flexibility for future enhancements. This work demonstrates the feasibility of implementing intelligent, multi-layered USB security systems that provide defense-in-depth protection superior to traditional single-method approaches.**

Keywords* - *USB Security; Malware Detection; Access Control; Authentication; File Quarantine; Behavioral Monitoring; Cybersecurity

I. INTRODUCTION

USB storage devices have become ubiquitous in modern computing environments, enabling convenient data transfer across systems. However, this widespread adoption has created significant security vulnerabilities. Unlike network- based threats that encounter multiple layers of perimeter defense, USB devices bypass network security entirely and connect directly to systems. Recent cybersecurity reports indicate that USB-based malware incidents represent the fastest-growing category of security breaches, with organizations losing millions annually to data theft and system compromise originating from compromised USB devices.

Traditional detection methods employ signature-based detection, maintaining databases of known malware signatures and comparing scanned files against these databases. Although useful for known malware, these methods fail against new variants of ransomware and polymorphic threats that continuously change their code to avoid detection. As attackers resort to more sophisticated evasion techniques, there is a critical need for adaptive detection systems that monitor behavior rather than relying solely on signatures.

Behavioral-based USB security involves identifying malicious activity patterns including sudden file modification events, high-frequency file renaming operations, unusual CPU consumption patterns, and suspicious file characteristics. Through a comprehensive multi-layered approach combining extension analysis, hash-based signature matching, suspicious filename detection, and file size analysis, it is possible to detect USB threats at an early stage before system compromise occurs.

II. LITERATURE SURVEY

USB security research has evolved from simple port-locking mechanisms to sophisticated software-based protection systems. Traditional approaches employed static signature detection and rule-based systems. Although computationally efficient, these solutions require constant updates and prove ineffective against novel malware variants not previously encountered. Recent research has demonstrated the efficacy of multi-method threat detection approaches. Extension-based analysis, hash-based signature matching, and suspicious filename detection provide complementary protection mechanisms. File size analysis has proven valuable in identifying packed or polymorphic malware attempts.

Behavioral monitoring studies have shown that tracking authentication failures, login patterns, and access anomalies significantly improves threat detection. Access control research emphasizes the importance of role-based permission management and account lockout protection against brute- force attacks. Studies show that organizations

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

implementing comprehensive audit trails achieve substantially better compliance and faster incident response times. File quarantine and isolation mechanisms prevent execution of suspicious files, containing threats before system compromise occurs. Based on these findings, this research work combines multi-method threat detection with

behavioral monitoring, strong authentication, and comprehensive logging to develop an efficient, practical USB security system.

III. EXISTING SYSTEM

Current USB security tools employ three primary approaches. First, signature-based antivirus software maintains large malware databases and searches files for matches. While effective for known threats, this approach fails against zero-day attacks and polymorphic malware. Second, policy-based restrictions through Group Policy Objects disable USB ports but provide no active threat detection. Third, hardware solutions like physical port locks provide basic protection but lack scalability and can be circumvented through firmware modification.Most commercial solutions are closed systems, limiting transparency and customization. The primary limitation

IV. EXISTING SYSTEM

Current USB security tools employ three primary approaches. First, signature-based antivirus software maintains large malware databases and searches files for matches. While effective for known threats, this approach fails against zero-day attacks and polymorphic malware. Second, policy-based restrictions through Group Policy Objects disable USB ports but provide no active threat detection. Third, hardware solutions like physical port locks provide basic protection but lack scalability and can be circumvented through firmware modification.

Most commercial solutions are closed systems, limiting transparency and customization. The primary limitation

V. EXISTING SYSTEM

Current USB security tools employ three primary approaches. First, signature-based antivirus software maintains large malware databases and searches files for matches. While effective for known threats, this approach fails against zero-day attacks and polymorphic malware. Second, policy-based restrictions through Group Policy Objects disable USB ports but provide no active threat detection. Third, hardware solutions like physical port locks provide basic protection but lack scalability and can be circumvented through firmware modification.

Most commercial solutions are closed systems, limiting transparency and customization. The primary limitation across all existing approaches is the lack of comprehensive behavioral monitoring and audit trails necessary for regulatory compliance and forensic investigation.

VI. PROPOSED SYSTEM

The proposed USB Physical Security System v2.0 employs a multi-layered defense strategy combining four complementary threat detection methods with behavioral monitoring, strong authentication, and comprehensive logging. Rather than relying on a single detection mechanism, the system uses extension-based analysis to identify files with dangerous extensions, hash-based signature matching to detect known malware, suspicious filename detection to identify system utilities commonly exploited by malware, and file size analysis to flag unusually large executables.

The system implements SHA-256 password hashing with cryptographic salt, enforces account lockout after three failed login attempts, implements role-based access control distinguishing between administrative and standard users, and maintains detailed audit trails of all security-relevant activities. Real-time email notifications through Gmail SMTP enable rapid incident response. File quarantine isolation prevents execution of detected threats. Windows Registry-based USB port control provides system-level protection. The pure Python implementation using only standard library ensures lightweight deployment with zero external dependencies.

VII.SYSTEM ARCHITECTURE

The proposed USB Physical Security System has a modular architecture designed for scalability, maintainability, and extensibility. The system comprises seven primary components. UserManager handles authentication, password management, and role-based access control using SHA-256 hashing. MalwareScanner implements multi-threaded threat detection through extension-based, hash-based, filename- based, and size-based analysis. SecurityManager orchestrates all security operations, coordinating authentication, logging, scanning, and notifications. SecurityLogger maintains comprehensive audit trails with timestamps, severity levels, and detailed activity descriptions. EmailNotificationService sends real-time alerts through Gmail SMTP. USBSecurityGUI provides a Tkinter-based user interface with color-coded status indicators and modal dialogs. Helper functions manage cross-platform USB drive enumeration and Windows Registry operations.

The Data Layer stores user accounts, security logs, and email configuration in JSON files. The Authentication Layer validates credentials and enforces access control policies. The Threat Detection Layer performs comprehensive malware scanning through multiple complementary methods. The Behavioral Monitoring Layer tracks failed login attempts and suspicious activity patterns. The Notification Layer sends real-time security alerts. The Logging Layer maintains detailed audit trails. The Presentation Layer provides intuitive user interaction through the Tkinter GUI. This layered approach ensures clean separation of concerns, straightforward testing, and easy integration of future enhancements.

TOOLS AND TECHNOLOGIES USED

The USB Physical Security System v2.0 is developed using a carefully selected technology stack optimized for security, performance, and portability. The entire application is built using Python 3.6+ with exclusive reliance on the Python standard library, eliminating external dependencies and reducing deployment complexity. Table I presents the complete list of tools and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

technologies used in the system implementation.

Component/Tool	Version	Purpose
Python	3.6+	Primary language for application logic
Tkinter	Built-in	GUI framework for interface development
hashlib	Built-in	Password hashing (SHA-256) and file signatures
json	Built-in	Data serialization and persistence
os/pathlib	Built-in	File system operations
threading	Built-in	Multi-threaded malware scanning
smtplib	Built-in	Gmail SMTP email integration
email.mime	Built-in	HTML email message construction
platform	Built-in	Cross-platform OS detection
datetime/time	Built-in	Activity logging timestamps
re (regex)	Built-in	Pattern matching for validation
typing	Built-in	Type annotations for code clarity

Fig. 1. Table I. Tools and Technologies Used

Python 3.6+ provides excellent cross-platform compatibility with a comprehensive standard library ideal for security applications. The language combines dynamic typing with optional type hints, enabling rapid development while maintaining code quality. Tkinter provides native GUI widgets ensuring consistent appearance across Windows, Linux, and macOS. Being Python's built-in toolkit, it eliminates external dependencies while offering sufficient functionality for desktop security applications. The hashlib module implements SHA-256 hashing with cryptographic salt for password security, protecting user credentials against rainbow table attacks. JSON format enables human-readable data storage for user accounts, security logs, and email configuration. Python's threading module enables multi- threaded malware scanning where background threads process files while the main thread continues handling GUI updates. The smtplib and email.mime modules enable real- time security alerts with color-coded severity indicators through Gmail's SMTP server.

VIII. IMPLEM ENTATION AND METHODOLOGY

The system is implemented in approximately 1,200 lines of Python 3.6+ code using exclusively the Python standard library. The User Manager class (~140 lines) implements password hashing using hash lib for SHA-256 generation with salt, password strength validation via regex patterns enforcing complexity requirements, security question verification for password recovery, and user account persistence through JSON files. The Malware Scanner class (~180 lines) implements multi-threaded scanning using Python's threading module, maintaining sets of dangerous extensions (16+ types including .exe, .bat, .dll,.vbs), suspicious filenames commonly targeted by malware, and malware signature hashes for comparison. The Security Manager class (~60 lines) orchestrates security operations and enforces lockout policies preventing brute-force attacks. The Security Logger class (~80 lines) implements comprehensive audit logging with JSON persistence, timestamp tracking, and severity classification. The Email Notification Service class (~120 lines) manages Gmail SMTP integration for HTML-formatted alert delivery. The USB Security GUI class (~600+ lines) implements the Tkinter-based interface with color-coded threat indicators, modal dialogs, and real-time progress updates.

The methodology employs comprehensive testing across all functional areas. Authentication testing verified that valid credentials enable login, invalid credentials trigger appropriate errors, password strength validation prevents weak passwords, failed attempts are tracked accurately, and account lockout functions correctly after three consecutive failures. Malware detection testing confirmed detection of dangerous extensions with 100% accuracy, hash-based signature matching correctly identifies known malware, suspicious filename identification works as designed, and large executable flagging functions properly. USB control testing verified Registry modifications on Windows systems and multi-drive detection across different platforms. Cross-platform testing confirmed correct operation on Windows 10/11, Linux, and macOS with graceful feature degradation on unsupported platforms.

IX. RESULTS AND DISCUSSION

Operation	Performance Metric	Result
Authentication	Login time (including disk I/O)	< 100 ms
Quick USB Scan	100 files with extension/hash matching	2–5 seconds
Deep Malware Scan	1000 files with multi-threaded analysis	30–60 seconds
Email Notification	SMTP send time (network dependent)	2–5 seconds
Log Query	Search across 1000 entries	< 500 ms
Password Reset	Security question verification	< 100 ms

Fig. 2. Table II. System Performance Metrics

Testing demonstrated robust threat detection across multiple vectors. Extension-based detection achieved 100% accuracy on files with dangerous extensions. Hash-based signature matching correctly identified all known malware samples in the database. Suspicious filename detection successfully identified system utilities exploited by malware. File size analysis appropriately flagged large executables that may indicate packed malware. Authentication security proved strong with SHA-256 hashing preventing password compromise, password complexity validation rejecting weak passwords, and account lockout mechanisms preventing brute-force attacks.

Behavioral monitoring accurately tracked failed login attempts and generated timely administrator alerts. The multi-threaded architecture maintained UI responsiveness during intensive scanning operations. The system successfully provides comprehensive USB security through multi-method threat detection, strong authentication with role-based access control, behavioral monitoring and activity logging, file quarantine isolation preventing malware execution, real-time email notifications enabling rapid incident response, and detailed audit logging providing compliance documentation. Real-world deployment testing across multiple platforms confirmed reliable operation and acceptable performance characteristics. The pure Python implementation with zero external dependencies enables lightweight deployment in diverse organizational environments.

XI. SECURITY ANALYSIS

A. Strengths

The multi-layered defense approach provides defense-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

in- depth protection against diverse threat types. SHA-256 password hashing with cryptographic salt provides strong credential protection resistant to rainbow table attacks. Account lockout after three failed attempts prevents brute-force attacks. Role-based access control enforces the principle of least privilege. Comprehensive audit trails enable forensic investigation and compliance reporting. File quarantine isolation prevents malware execution. Real-time email alerts enable rapid incident response. Cross-platform support increases deployment flexibility. The pure Python implementation reduces the attack surface significantly.

B. Limitations and Recommendations

MD5 hashing for file signatures could be upgraded to SHA-256 for production deployments to address theoretical collision vulnerabilities. The malware signature database is extensible but initially limited; integration with the Virus Total API would expand coverage. Local JSON file storage would benefit from SQL encryption for production use. USB port control is Windows-specific due to OS API limitations; equivalent features would require platform-specific implementations for Linux and macOS. The single- machine design could be extended to support centralized policy management for enterprise deployments. Despite these limitations, the modular architecture enables straightforward enhancement as requirements evolve

XII. FUTURE SCOPE

Several enhancements would increase production applicability. Database migration from JSON to SQLite with proper indexing would improve query performance for large- scale deployments supporting thousands of security events. Encryption of sensitive files using AES-256 would protect credentials in production environments. Integration with the VirusTotal API would expand malware signature coverage enabling detection of emerging threats. Implementation of two-factor authentication using TOTP would strengthen account security. Real-time file system monitoring using a watchdog library would track actual file operations rather than simulated behavior.

Machine learning models could analyze behavioral patterns to detect anomalies and identify novel attack types. A REST API would enable integration with enterprise SIEM systems and security platforms. A web-based management dashboard would support centralized administration across multiple machines. The system can be deployed as a background operating system service providing constant protection, and could be further developed as a microservice.

X. CONCLUSION

This project proves the feasibility of implementing an intelligent, multi-layered USB security system combining strong authentication, multi-method threat detection, behavioral monitoring, and comprehensive logging. Unlike traditional signature-based security approaches that fail against zero-day exploits and polymorphic threats, the proposed system provides defense-in-depth protection through four complementary detection methods. The multi-threaded architecture maintains excellent performance while scanning large USB devices. Comprehensive testing across authentication, threat detection, access control, logging, and cross-platform compatibility confirmed robust system operation.

The system design is modular and maintainable, ensuring flexibility to incorporate emerging security technologies. The experimental results confirm the efficacy of multi-method threat detection combined with behavioral monitoring in protecting against USB-based attacks. The pure Python implementation using only standard library enables lightweight deployment with zero external dependencies. The comprehensive audit trail provides compliance-ready documentation. This project demonstrates the viability of intelligent USB security systems as an effective defense against evolving threats, suitable for deployment in enterprise, educational, healthcare, financial, and government environments where USB security is critical.

REFERENCES

[1] Lalonde Levesque, F., & Pooch, J. (2011). A Survey on USB Protocol Vulnerabilities. In Proceedings of the 4th International Conference on Cyber Security, pp. 45–58.

[2] Kumar, A., & Lim, T. (2016). USB Device Security: Implementation and Evaluation of Access Control Mechanisms. IEEE Transactions on Information Security, 45(3), 234–248.

[3] Richardson, R. (2018). 2018 CSI Computer Crime and Security Survey. Computer Security Institute.

[4] Park, S., Kim, J., & Lee, H. (2019). Real-Time Malware Detection using Behavioral Analysis and File Characteristics. Journal of Cybersecurity Research, 12(4), 445–462.

[5] Thompson, M., Anderson, B., & White, K. (2020). Enterprise USB Security: Policy, Implementation, and Compliance. International Journal of Information Security Management, 28(1), 78–95