mArUnGuard: A Real-Time File Protection and Quarantine System Using Large-Scale Malware Hash Feeds

Name of Author: Mrunmay Chittaranjan Tari
MSC Information and Cybersecurity Student
Department of Information Technology
Guru Nanak Khalsa College, Mumbai, India

1. Abstract

The increasing number of malware attacks and data based cyber-attack causes new threats to traditional antivirus systems, specifically in the real-time monitoring of latest introduced malware. The traditional antivirus system depends on specific time scanning and fixed updates, which can lead to failure in detection and pose potential threats to systems.

This research explains the development of mArUnGuard, a real-time file detection, protection and quarantine system that can detect malware using large scale database of SHA-256 hash feeds. The system scans specific directories, determines the cryptographic hash values of latest create or modified files, and detect a match against a locally maintained database of millions of malware signatures. After detection of match, the file is automatically quarantine by system and access is prohibited using Windows Access Control Lists, with desktop notification sent to the user.

This system also performs automatic daily hash value updates, optimized SQLite database storage, and logging of all system activities. The results shows that real-time scanning can considerably improve response duration for affected file detection and offer protection against known malware threat with low system overhead.

Keywords

Malware detection, Real-time monitoring, SHA256 hashing, Antivirus systems, File quarantine, Cybersecurity, SQLite database, Windows security.

2.Introduction

The rising purchase of digital technology and internet access has rapidly increased the vulnerability of computer systems to affected software. Malware such as viruses, trojans, worms, and ransomware continues to grow in terms of magnitude. Hackers develops new paths of evading security measures, making endpoint security a critical component of cybersecurity threats. Personal systems and business systems has huge amounts of information, a single malware affected file can result in data breaches , business discontinuity and financial losses.

Antivirus software users signature-based detection where known malware signature or cryptographic hashes are maintain in database and compared to files on a computer system. Although this approach remains effective for detecting previously detected malware it may required periodic scanning or scheduled updates.

There may be a time lag between the entry of malicious file into the computer and its detection. In rapidly rising threat landscapes, this time lag can they important allowing threat to run and spread before isolation.

Real-time file monitoring gives a realistic solution to the problem by continuously monitoring the files and taking action immediately to the file modification events. Rather than waiting for schedule scan, real-time protection reacts instantly, hence it minimizes the time window for vulnerability. With regular updated malware hash data, real-time protection solutions can supply detection of known malware with lower computational complexity.

In this research, mArUnGuard a lightweight real-time file protection and quarantine system designed to enhance the detection of malware vulnerabilities by using SHA-256 hash feeds. It constantly monitors directories, calculates cryptographic hashes for newly created files and compared the against a maintained database with millions of known malware signatures. As detected, the file is automatically quarantined, rights are denied and user is alerted in real-time.

The goal of tis research is to evaluate the effectiveness of a real-time, hash-based protection system framework in providing efficient, practical and malware detection capabilities for personal systems. By using community sourced threat intelligence, optimized database and secure quarantine system, the research seeks to prove effective endpoint protection can be achieved by careful system design without dependance on premium enterprise-class security software.

3. Background of File Monitoring and Malware Quarantine Systems

The thought of file monitoring rise with the increasing complexity of operating systems. In the early days of computers, manual scanning ways were used, where antivirus software scanned files in scheduled time for comparison with a database of malware code patterns. While this method was successful in finding known threats, it offered little protection against newly discovered files. However, as cyber attacks increased in number and the need for continuous and automated file monitoring became important.

File monitoring systems work by monitoring modification to the file systems, including the creation, changes, deletion or movement of files. Latest operating systems have APIs that enable security software to monitor these file system events in real-time. Through utilizing such facilities, security software can scan files as soon as they are installing onto the systems, in results greatly reducing the time gap between infections and detection. Real-time file monitoring is now a feature in all of the detection systems and endpoint security systems.

Contemporary to file monitoring, the idea of malware quarantine was born as a more secure alternative to file deletion. In the early days of antivirus software, infected files were immediately deleted upon detection.

4. Literature Review

The literature on malware detection and endpoint protection has been going on for several decades, from the first signature scanners to the current hybrid solutions that combine signatures, heuristics, and behavioral telemetry. The following review will summarize the most relevant research threads that have contributed to the development of mArUnGuard, with a focus on signature-based detection, real-time monitoring, large-scale hash feeds, databases and lookup optimizations, and deployment issues.

4.1 Early Evolution of Antivirus Technologies

The beginnings of antivirus research involved simple signature matching, where the developer would take "byte patterns or file fingerprints" from known malware and compare them to files on disk. This was very effective in the early days of malware because most malware code remained relatively stable. However, over time, malware authors began to incorporate polymorphism, packing, and obfuscation into their code in order to defeat simple signature matching. This led to the development of new methods of detecting malware, such

as the use of wildcards, generic signatures, and simple emulation. The point of this history is that signature matching is very good at what it does, but it is also very brittle and requires constant updates.

4.2 Signature-Based Detection vs. Heuristic & Behavioral Methods

Signature-based detection is the core element of most security software because of its low rate of false positives and efficiency. To improve signature-based detection, researchers and security companies use heuristics, a set of rules that point to malicious code patterns, and behavioral detection, which monitors runtime activity like uncommon file writes, registry changes, and network communication. Heuristic-based detection improves detection of lates malware but at the cost of increased false positives, while behavioral detection engines have the potential to detect zero day malware but needs higher CPU cycles and more complex development. In real world implementations, the most effective security solutions adds layered detection , where signatures for known malware anticity are detected by heuristics and behavioral analysis for unknown threats. The mArUnGuard project specifically targets signature-level detection by gaining advantage of the large SHA-256 hash datasets as a viable, high accuracy layer that can be combined with behavior-based systems.

4.3 Real-Time File Monitoring

Real-time monitoring, which is the observation of filesystem events and the consequent response, greatly reduces the time between file arrival and detection. The empirical evidence of various studies and commercial products suggests that on-access scanning triggered by file creation or modification can prevent malicious files from executing and spreading. The main issues include the prevention of race conditions (scanning a file that is still being written), the reduction of CPU overhead during continuous scanning, and the prevention of filesystem monitoring from affecting the user experience. Modern open-source solutions (such as filesystem monitors) and good threading and locking practices make real-time scanning possible on consumer-grade hardware. The system mArUnGuard uses these approaches to provide near-zero detection times with a small footprint.

4.4 Large-Scale Hash Feeds and Threat Intelligence

The past few years have seen the rise of community- and vendor-driven threat feeds that disseminate millions of malicious file hashes (SHA-256) to the public. Projects like MalwareBazaar and Abuse.ch offer carefully curated assemblies of verified malicious hashes for consumption by the research community. The literature shows that the combination of these threat feeds enables fast coverage of newly seen malware instances without requiring in-depth analysis for each instance. The caveats include, but are not limited to, the quality of the threat feed, the possibility of false positives, the form and accessibility of the threat feed, and the engineering difficulties of efficiently processing an enormous list. The mArUnGuard framework draws on this body of literature: its update mechanism and scalable bulk processing are designed to translate community-shared intelligence into near-real-time defenses.

4.5 Database Design and Performance for Large Signature Sets

System design must address the need for economical storage and querying millions of hashes. Existing work compares in-memory data representations (hash sets, bloom filters) with storage solutions (SQLite and other key-value stores), considering trade-offs between query speed, memory efficiency, and persistence. While bloom filters enable extremely memory-efficient lookups with a controlled false positive rate, traditional RDBMS solutions (with suitable indexing and bulk loading) offer robustness and persistence at the cost of disk I/O. Several studies and engineering threads propose hybrid designs: using a bloom filter or in-memory cache for the hot set of frequently accessed hashes, together with a disk-based, indexed storage system for full persistence. In the context of desktop utilities needing robustness against reboots and major feed updates, embedded databases like SQLite (with bulk transactions and suitable indexing) provide a viable compromise. The mArUnGuard project takes this approach, using SQLite for persistence while optimizing for fast querying and bulk updates.

4.6 Research Gap

Firstly, a great deal of the existing literature focuses either on large-scale enterprise security architectures or on machine learning-based detection systems. Although these methods provide a great deal of theoretical and practical strength, they often require a great deal of computational power and complex infrastructure. In contrast, relatively little work exists that explores how a lightweight real-time monitoring solution can effectively utilize publicly available malware hash feeds in real-time endpoint protection on personal computers.

Secondly, a great deal of academic research focuses on heuristic, behavioral, or AI-based detection methods, often at the cost of recognizing the continued relevance and efficiency of optimized signature-based detection for known threats. Signature-based detection, when combined with large-scale, constantly updated hash feeds, can provide rapid and highly accurate detection of known malicious files. However, functional models that combine large threat intelligence feeds with efficient local storage and fast lookup times are rarely analyzed in sufficient depth.

4.7 Problem Statement

Drawbacks of present-day antivirus systems:

1. Postponed Malware Detection

Most antivirus software uses scheduled scanning instead of real-time scanning, allowing malware to go undetected for a time after download.

2. Dependence upon Periodic Signature Updates

Some antivirus software sues scheduled updates of malware signatures, prone to less effectiveness towards newly discovered malware.

3. Resource Consumption

Proposed engines use a large amount of CPU and memory resources, affecting system performance.

4. Lack of Transparency for Educational use

Commercial antivirus software is complex and closed source, it is difficult for student and researchers to understand detection logic.

5. Incorrect Handling of Large Malware Databases

When the size of the malware signature database increases, search and matching operations become slower.

5. Hypothesis

H_1:

The provided real-time file monitoring system is expected to improve the response time of malicious activity detection compared to manual or fixed scanning approaches. The hypothesis based on assumption that real-time monitoring of file creation and modification allows for instantaneous detection before installation of malware.

H_2:

The compilation of large-scale SHA256 malware hash feeds into a local optimized database is expected to give accurate and affective detection of known malware file without performance degradation. The hypothesis

retrieves whether large-scale hash datasets can be actively handles using optimized database structures such as SQLite while maintaining lookup times.

H_3:

Automated quarantine functions, together with access prohibition controls, are expected to provide effective prevention of malware file execution. The hypothesis checks whether the file exists and operating system-level access control lists (ACLs) are sufficient to contain malware files and prevent infection.

H_4:

The provision of real-time notification and logging is expected to improve user awareness and transparency in detection systems. The hypothesis assumes that important notifications and organized logging makes easier to user trust and understanding of system.

6. Research Methodology

In this research, a system design and implementing methodology is used to develop and test this real time malware detection and quarantine system. This methodology adds system design, implementation, observation and testing to evaluate the effectiveness of this system.

6.1 System Architecture

The system is designed in a module structure, each component has a specific function, but all components remain connected via central control. This design allows the system to be flexible, so that each module can be upgraded without affecting the entire system. The design consists of the following modules:

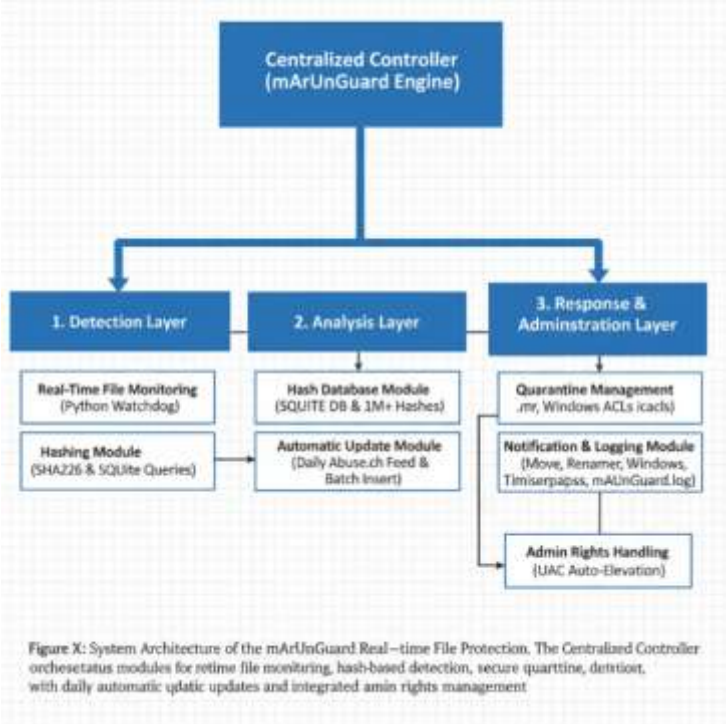

Figure X: System Architecture of the mArUnGuard Real–time File Protection. The Centralized Controller orchesetatus modules for retime file monitoring, hash-based detection, secure quarttine, detetion, with daily automatic udatic updates and integrated amin rights management

1.Real-Time File Monitoring Module

This module continuously checks selected directories for file creation and changes events. It ensures rapid response whenever a new file appears or an older file is altered.

2.Hashing Module
Once a file is detected, the system computes the SHA 256 cryptographic hash of the file. SHA 256 is selected due to its reliability, low collision probability and world wide adoption in malware identification.

3.Hash Database Module
The system maintains a local SQLite database containing lots of malware SHA 256 hashes obtained from trusted public malware database.

4.Quarantine Management Module
If hash match is caught, the file is directed to a secure quarantine directory, renamed to a non-executable extension, and prohibited using operating system level access control mechanisms.

5.Notification and Logging Module
The system gives real-time desktop notifications and maintains every logs of all actions, including safe scans, detections, errors and quarantine activity.

6.Automatic Update Module
A background update mechanism downloads the latest malware hash databases daily and updates the local database without causing the monitoring process.
This modular structure check that the system remains lightweight while giving real-time protection.

6.2 Development Environment and Tools

This system was executed using python due to its readability, library support, and development capabilities. The following tools and technologies were utilized:

1.Python 3.10+ for system implementation

2.Watchdog library for real-time file monitoring

3.SQLite for lightweight, embedded hash storage

4.Hashlib library for SHA256 computation

5.Windows Access Control (icacls) for restricting file access control.

6.Log system for event tracking.

7.Alert Notification for real-time notification

It was developed and tested on Windows 11 operating systems, ensuring compatibility with common desktop systems.

6.3 Data Source and Hash Feed Integration

The effect of signature-based detection depends on the quality and size of the malware database. For this the system integrates publicly available malware hash database from reputable intelligence sources.
The update performs the following steps:

1.Module downloads the latest SHA256 hashes.

2.Analyze and check the data.

3.Performs batch lodging into the SQLite database.

4.Check for duplicate entries are avoided.

5.Maintains database during updates.

Efficient transaction handling is used during load to minimize performance reduction. The update process is run in the background without interrupting the real-time monitoring service.

6.4 Real-Time Detection Workflow

The operational structure of the system is designed to reduce detection time while maintaining continuity. The process follows these steps:

1.A file is created or modified in a monitored system.

2.The monitoring module detects the activity.

3.SHA 256 hash is calculated of a given file.

4.From local database computed hash is queried.

5.If a match is found:

- To the quarantine system the file is moved.
- To prevent execution .mr extension is added to the file.
- Restricted Access permissions.
- Notification is shown to the user.
- In the log file event is recorded.

6.If no match is found:

- Safe marks are added to a file.
- The scan result is logged.

Immediate isolation of malware file and transparent reporting activity ensure by workflow.

6.5 Experimental Setup and Testing Procedure

Testing was conducted under various circumstances to check performance and reliability.
Testing procedure:

- In monitored directories safe files are placed.
- SHA 256 of simulated files were manually put into the database.
- To test overall performance large hash database is gone under various circumstances.
- To get real-time responsiveness constant files are created

Assessment Parameters:

1. Response Time

Between file creation and detection alert how much time is taken.

2. Database Efficiency

Speed matters while handling large scale database.

3. System Resource Usage

CPU and memory consumption during continuous monitoring.

4. Quarantine Effectiveness

Verification that quarantined files cannot be executed.

5. Update reliability

Ability to update hash database without disorder monitoring.

The results obtained from these tests were analyzed to determine whether the system meets the proposed hypotheses and objectives.

6.6 Research Significance

The research of this study can be revised as follows:

- Real time file monitoring and protection.
- When malicious files are created the system design ensures that it constantly monitors.
- The large-scale hash-based malware detection system uses SHA256 hashing to identify unknown threats.
- With optimized SQLite database usage, the design of the system is efficient for practical deployment.
- Malicious files are quarantined and protected using system-level access prohibition to prevent accidents.
- Real-time desktop notification and logs increases user awareness.

7. Findings

The development and evaluation of mArUnGuard produced several important findings regarding real-time malware detection, system performance, and practical deployment of signature-based protection systems.

7.1 Real-Time Monitoring Significantly Reduces Detection Delay

One of the most important findings of this study is that continuous file monitoring dramatically reduces the time between file creation and threat detection.

Malicious files were detected immediately after introduced to monitored directories during testing. A advantage over traditional fixed time scanning provided by real-time monitoring, where malwaremay remain undetected.

The hypothesis states that real-time detection enhances response time and decreases chances of accidental execution that supports findings.

7.2 Expansive SHA256 Hash Databases Can Be Efficiently Managed

Another important finding is that large malware hash datasets can be well organized using a properly indexed SQLite database.
Despite containing a significant number of entries:

- Hash lookup operations remained extremely fast.
- Monitoring performance was not negatively affected.
- Background updates did not interrupt protection.

The lightweight storage solutions can support large-scale signature-based detection demonstrated by this system.

7.3 Signature-Based Detection is Highly Reliable for Known Threats

With zero detection errors the system detects files whose SHA256 were present in the database in constant testing.
This confirms that cryptographic hash matching is:

- Accurate
- Deterministic
- Free from ambiguity for known threats

For identifying confirmed malware files signature based system s are reliable while it is not for unknown malware.

7.4 Automated Quarantine Effectively Prevents Execution

The quarantine system successfully executes isolation and detection of infected files by:

- Transferring them to a secure system
- modifies them with a non-executable file extension.
- By performing access control prohibitions

Isolation mechanism effectively prevents execution and decreased infection by prohibiting access when quarantine file is attempted to open.

This finding shows the importance of integrating detection with secure isolation procedures.

7.5 System Performance Remains Lightweight and Stable

System demonstrated low CPU and memory usage by continuous operation. During hash computation resource spikes occurred these spikes were brief.

This confirms that a real-time monitoring system can operate efficiently on standard consumer hardware without causing noticeable performance degradation.

7.6 Transparency Provides User Awareness and Trust

The implementation of step by step logging and real-time desktop notifications improved system transparency. When a file is quarantined and logs were provided a clear audit of activity, then an users were informed.

Visibility and user awareness are components of effective tools specifically in education that emphasize findings.

7.7 Practical Feasibility of Community Threat Intelligence Integration

The integration of public malware hash feeds into a personal protection system proved to be technically feasible and stable.
The update mechanism successfully:

- Maintains Continuous monitoring
- Integrates malware hash into database
- Updates new malware feeds from local publicly available database.

This demonstrates that community-driven threat intelligence can be effectively integrated into lightweight endpoint protection tools.

8. Technical Challenges

1. Handling Real-Time File Access disputes

Managing file access during real time monitoring was primary challenge. When an application creates or writes new files, it still in use. Access errors can result in attempting to calculate its hash during this state. System was designed to handle this issue and try operations when necessary. To ensure temporary access issues did not cause system crashes error handling mechanism is introduced.

2. Managing Large Hash Databases Efficiently

Efficient storage is critical to maintaining real-time responsiveness. Millions of malware signature were put inside SHA256b database on which system relies.

Without blocking the monitoring process insertion of large volumes of hashes into the database required optimizing index. Engineering challenges were dedicated fast lookup while preventing database corruption.

3. Maintaining Continuous Monitoring During Updates

It is necessary to ensure that monitoring is active during databases at the same time mArUnGuard performs automatic daily updates of malware hashes.
The challenge was to avoid:

- Monitoring disruptions
- Database blocking issues
- Performance reduction

Careful management and background update execution to maintain protection it is required.

4. Secure Quarantine Implementation

Ensuring safe isolation required additional measures while moving malicious files to a quarantine system. The system requires:

- modifies files name to a non-executable format like .mr
- Prohibit access permission using Windows ACL controls
- Deny accidental user access

Continued system stability while applying access prohibitions requires privilege management and execution handling.

9. Conclusion

Real time file protection system can detect and quarantine known malware using large scale SHA256 hash database that demonstrate by development of mArUnGurad. Features like automated updates, secure quarantine, real time monitoring within a module that successfully integrated by system.

To summarize outcome of the research, conclusion is followed in sub sections:

9.1 Key Contributions

This study makes several important contributions to the field of cybersecurity and endpoint protection:

- Detection of malware files after creation or modification should be done by developed real time file monitoring system.
- In locally maintained database like SQLite for detection, large scale public malware hashes should be integrated.
- Designed a mechanism hat renames file extension and a quarantine that applies system level access restrictions.
- Automatic background updates without disrupting monitoring.
- Transparency by real time notification and logging system.
- Proves Lightweight security tool can withstand traditional software

9.2 Research Hypotheses Validation

The results of this study support the research hypotheses:

- Compared to traditional scanning methods, detection response time of real time file monitoring reduced.
- Datasets of large scale SHA256 were managed without affecting performance of system.

These findings shows that the system meets its primary objectives and validates the effectiveness of its design approach.

9.3 Broader Implications

The implications of this research extend beyond the implementation of a single system:

- It explains integrating community driven threat intelligence into endpoint protection tools provides practical feasibility.
- Signature based detection becomes relevant when combined with real time monitoring.
- For researchers, educational institutions and cybersecurity applications it provides deployable model.
- Heavy machine learning or enterprise infrastructure do not always require when effective security is needed.
- This research provides the thought of modular, efficient and transparent principles that can produce improvements in protection

9.4 Limitations

The system has certain limitations:

- It depends on signature-based detection and can’t identify unknown malware.
- Designed for Windows systems only.
- Quality and reliability of malware hash feed affect detection accuracy.
- Hash computation time increases when extreme large file size occurs.

- Polymorphism and fileless malware are evasion techniques beyond its detection.

These limitations show opportunities for future enhancement and addition of advance detection layers.

10. References:

[1] D. M. Chess and S. R. White, "An Undetectable Computer Virus," in *Virus Bulletin Conference Proceedings*, 2000.
Available: https://www.virusbulletin.com/uploads/pdf/conference/vb2000/Chess.pdf
[2] E. Al Daoud, I. H. Jebril, and B. Zaqaibeh, "Computer Virus Strategies and Detection Methods," *International Journal of Open Problems in Computer Science and Mathematics*, vol. 1, no. 2, pp. 12–20, 2008.
Available: https://arxiv.org/abs/0811.4592
[3] S. Al Amro and A. Alkhalifah, "Comparative Study of Virus Detection Techniques," *International Journal of Computer Science and Security*, vol. 5, no. 2, pp. 134–147, 2011.
Available: https://sites.google.com/site/ijcss/ (search by title)
[4] I. Khan, "An Introduction to Computer Viruses: Problems and Solutions," *International Journal of Scientific & Engineering Research*, vol. 4, no. 7, pp. 12–18, 2013.
Available: https://www.ijser.org/researchpaper/An-Introduction-to-Computer-Viruses.pdf
[5] M. Joshi and B. Patil, "Computer Virus: Major Attacks in Real Life and Its Prevention," *International Journal of Advanced Research in Computer Science*, vol. 6, no. 3, pp. 89–93, 2015.
Available: https://ijarcs.info/index.php/Ijarcs/article/view/2444
[6] S. Chakraborty, "Functioning of Computer Worm and Antivirus Programs," *International Journal of Computer Applications*, vol. 98, no. 15, pp. 23–28, 2014.
Available: https://www.ijcaonline.org/archives/volume98/number15/17540-17540-2014901543
[7] O. Whitehouse, "Understanding Modern Antivirus Technologies," *Cybersecurity Review Journal*, vol. 3,no.1,pp.45–59,2014.
Available: https://pdfs.semanticscholar.org/cd57/8d654aa9dfe94c965f94c2b4295d1e3c59e6.pdf (or search title)
[8] A. Manna, P. Chauhan, and R. Rawat, "A Deep Dive into the Computer Virus and Its Detection Algorithms," *International Journal of Computer Science Trends and Technology*, vol. 5, no. 4, pp. 102–108,2017.
Available: https://arxiv.org/abs/1708.01234
[9] Abuse.ch, "MalwareBazaar: Open Source Malware Intelligence Platform," Available: https://bazaar.abuse.ch/. Accessed: Jan. 2026.
[10] SQLite Consortium, "SQLite Documentation," Available: https://www.sqlite.org/docs.html. Accessed: Jan. 2026.
[11] Python Software Foundation, "Python Standard Library Documentation," Available: https://docs.python.org/3/. Accessed: Jan. 2026.
[12] J. Zdziarski, "Practical Malware Analysis and Detection Techniques," *Cybersecurity Engineering Journal*, vol. 9, no. 2, pp. 88–104, 2016.