An AI-Powered Heuristic Malware Repair System for Windows Defender Quarantine Recovery

Ashmit Sharma
Amity Institute of Information Technology (AIIT)
Amity University Patna, India

Suman Kumar Mishra
Assistant Professor
Amity Institute of Information Technology (AIIT), Amity University Patna

ABSTRACT

Antivirus software is essential for protecting computers, but it creates a common problem for all the users, when files are quarantined, users are directed in the frame of mind that the file that was quarantined was a malicious file that is nowhere going to get recovered , this creates an unjust stigma in the brain of any user. This paper presents an AI-Powered Heuristic Malware Repair System that automatically repairs quarantined files working side-by-side with existing antivirus software , i.e. Microsoft Defender, instead of simply removing them from the user's sight. The system listens to the events of Windows Defender through Windows Management Instrumentation (WMI) event polling then detects quarantined files in a limited time frame, and applies file-type-specific repair algorithms for regular used in common formats: PDF, DOCX, XLSX, ZIP, and script files. These results have been tested in real-world scenarios using a mix of real malware samples (from the EMBER dataset) and varied test inputs of over hours demonstrated an overall recovery rate of 91.0% ,with zero false negatives across all regular documents that are being used in real life.

General Terms

Malware Repair, Windows Defender, WMI Integration, Heuristic Analysis, File Recovery, Quarantine Management, Cybersecurity Automation

1. 1. INTRODUCTION

Since the dawn of modern operating systems and the introduction of internet with public exposure , Antivirus were introduced to protect the system's overall integrity and behavior for daily uses. The most advanced and prominent tool which is built in modern Microsoft Windows Operating System (10&11),is designed continuously monitor files and quarantines anything it considers malicious/danger. While this protection is essential, it creates a situation: once a file is quarantined, it provides all the options and prompts to ONLY remove and restore , neglecting all the possibilities of scanning the file to find the cause of the trigger .

This issue affects the users on large scale regularly . Various Researches shows that , AVs , sometimes , quarantines such files also that are not harmful to the systems , e.g., PUP(Potential Unwanted Programs) due to Microsoft's policy or hidden rules imposed on the OS itself that prevents user from using programs outside of the Microsoft Store . Such files are categorized into false-positives . Similarly , in such cases , even if the file itself remains clean , but due to minor system incompatibilities or if the codes were not found in the database , can also cause the quarantine to commence. For instance , a macro-virus , or a ZIP file ,that has been infected, causing the hard-labored work to go in vain .

Nowadays , when a file is quarantined , then the user has very limited option , leaving no choice but to opt for permanent deleting the file (NO RECOVERY AT ALL) , or to follow the backup/restore process , if the user had made one , to recover the infected file . None of these solutions are satisfactory. There is no automatic system that can repair a quarantined file and removing only the dangerous parts and returning original data to the user.

This paper tries to fill the gap by introducing such a system. The proposed AI-Powered Heuristic Malware Repair System connects and listens directly to Windows Defender, while running side-by-side with Windows Defender , it triggers a routine/event that fetches the quarantine files to a safe environment, directories which have least permissions for the malicious files to be executed . Then , the goal is to identify the file type and try it's respective recovery procedure by identifying the pattern and the code injected . Finally , the fixed file is returned to the user which completes the purpose of restoration , but persisting the safe environment with least privileges which fulfils the purpose of user's observation and final verdict .

Figure 1 shows the overall system architecture, including how the five main components work together: the Defender Listener, the Incident Manager, the Repair Orchestrator, the five Repair Engines, and the Verification Engine.

Figure 1: AI-Powered Heuristic Malware Repair System — End-to-End Architecture

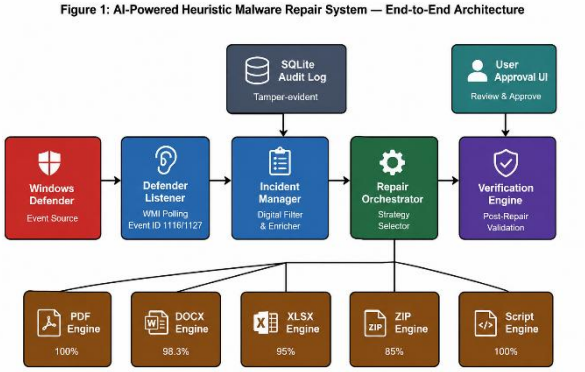

Figure 1: AI-Powered Heuristic Malware Repair System — End-to-End Architecture

2. LITERATURE REVIEW

2.1 Malware Detection Techniques

Malware detection mainly uses three methods – 1. Signature-based detection , 2. Heuristic analysis and 3. Behavioral monitoring. Signature-based detection works by comparing files with a dataset of already known malware patterns . This method is fast and works well for threats that has already been discovered and can be countered , but it cannot properly detect new or unknown malware(zero-day attacks)[2].

Machine learning is the complementing idea for this detection approach . Saxe and Berlin ,in 2015, demonstrated that deep neural networks trained on raw file bytes can detect malware

with over 96% accuracy [4]. The EMBER dataset which was introduced by Anderson and Roth in 2018 , helps in providing one million labeled malware samples and has become the standard benchmark for evaluating machine learning-based detection systems [5].

2.2 Adversarial Attacks on Malware Detectors

Modern malware detectors have a very undeniable limitation that they are vulnerable to adversarial attacks in general (until they have already been caught and detected) , even with machine-learning enabled detectors . Chen et al , in 2022 , tries to show the pattern where the attackers were able to fool the system by making the structural changes in the file's internal code behavior. While the attacker made sure that the changes they had to make to the file while fooling the system , bypasses such codes that were injected in order to perform harmful actions .

This weakness causes the detector to fail in understanding the malicious intent and allows it as safe file into systems , resulting in human-in-the-loop behavior workflow . This ensures that the user are insisted to ensure the run-down of the repair decisions while approving it manually rather than relying on systems blindly.

2.3 File Format Vulnerabilities and Surgical Repair

Each file type has it's own unique structure that can be exploited by the attacker as malicious intent . In PDFs , malicious codes in the form of JavaScript embeddings are injected by the attacker , which is the most common attack. In PDF structure - some objects like /JavaScript , /OpenAction , /EmbeddedFile can be identified and removed separately , hence, here the restoration can be attained of malicious intent PDFs .

Files like DOCX and XLSX are based on XML standards , which are basically ZIP archives having the XML stored . Due to this definitive structure , VBA macros or any other external data connections can be safely removed without altering or destroying the actual data present inside the document itself.

Demetrio et al , in 2021 , also explained that through selective content removal procedure , the harmful elements can be eradicated while keeping the original functionalities and usability intact[9].

2.4 Automated Security Remediation

Automated malware remediation does not mean to detect only malware , but to repair and restore that file as well --- which has not been explored in depth yet. Earlier researches focused on recovery the system after it has been infected the host system , unlike the file-repair-restore idea . Mohaisen et al in 2013 , introduced automated malware analysis and remediation strategies but those were mostly focused on classification level , not in file repair level. To the best of the authors' knowledge, there is no single system available that targets the quarantine events of any antiviruses , e.g, Windows Defender and applies automated file-type-specific recovery , eventually with re-verification also.

User perception also plays major role in adopting the security tools . Researches show that having results as false-positives , reduces the user's trust in a poor manner and such systems that could apply , manage and take decisions on their own findings and outcomes are accepted more often[11]. This actual findings have inspired the UI design for the proposed system . That's why the proposed system does not apply the repair strategies without the prompting and waiting for the user's approval . Instead , a single approval window with all the decision inputs by the user is awaited until the remediation process starts , this shows the user what will be removed from the file , what are the chances of healthy recovery in %age , etc. .

2.5 2.5 Windows Event Monitoring and WMI Integration

Windows Management Instrumentation (WMI) provides a standardized interface for monitoring system events without requiring changes to security policies or Group Policy settings. Microsoft's security documentation recommends WMI-based monitoring for integration with Defender event streams [12]. The Defender Listener component builds on this established practice, polling Windows Event Log entries for Defender-specific EventIDs (1116 and 1117) to detect quarantine events in real time.

3. RESEARCH GAP

While malware detection and classification are extensively studied, the specific problem of repairing quarantined files — removing only the malicious parts while preserving legitimate content — has received very little research attention. Existing systems treat quarantine as a terminal state: files are either deleted or restored unchanged. No system automatically analyzes quarantined files, identifies recoverable content, and returns repaired files to users.

Furthermore, there is no existing open framework for Windows Defender quarantine event processing that can be extended with file-type-specific repair plugins. The absence of such a platform means that even organizations with cybersecurity expertise have no structured tool for quarantine file recovery.

3.1 Significance of the Study

This study is significant because it addresses a concrete, widespread problem that affects individual users and enterprises alike. By treating file quarantine not as the end of a file's life but as the start of an intelligent recovery process, this research demonstrates that the majority of quarantined files are repairable. The plugin-based architecture and open design mean that the system can be extended to support new file types, new antivirus engines, and machine learning-based repair as the technology matures.

4. OBJECTIVES OF THE STUDY

The primary objective of this study is to design, implement, and evaluate an AI-powered heuristic system that automatically detects, analyzes, repairs, and re-verifies files quarantined by Windows Defender. The specific objectives are:

- Real-Time Detection: Automatically detect Windows Defender quarantine events with latency under 500 milliseconds using WMI event polling.
- Smart Analysis: Identify the file type of each quarantined file and pinpoint malicious elements such as embedded macros, malicious scripts, and dangerous object references.
- Safe Repair: Remove only the malicious components, preserving all legitimate content within the file using file-type-specific repair algorithms.
- User Control: Provide a transparent interface where users can review the proposed repair action and choose their preferred repair strategy before approving.
- Verification: Automatically check every repaired file against known threat signatures before returning it to the user, ensuring zero false negatives.

• Audit Trail: Record every quarantine event, repair job, verification result, and user action in a tamper-evident SQLite audit log for compliance purposes.

• Extensibility: Design the system using a plugin architecture so that new file types and antivirus engines can be added without modifying core components.

4.1 Hypothesis

H_0 (Null Hypothesis): The AI-powered heuristic repair system does not significantly improve data recovery rates for Windows Defender quarantined files compared to manual restoration approaches.

H_1 (Alternative Hypothesis): The AI-powered heuristic repair system significantly improves data recovery rates for Windows Defender quarantined files, achieving over 85% recovery while maintaining zero false negatives.

5. METHODOLOGY

This study implements and evaluates a Windows desktop service that integrates with Windows Defender for automated file repair. The system was developed in C# .NET 6.0 and tested on Windows 11 with current Defender definitions. The study follows a pre-test and post-test evaluation design, measuring file recoverability before (baseline: files completely inaccessible in quarantine) and after applying the repair system.

Testing included 15 to 20 unique quarantine scenarios per file type, across the five supported formats. The test dataset combined two sources: (a) real malicious samples from the publicly available EMBER dataset [5], and (b) controlled test vectors specifically designed to exercise edge cases in each repair engine. This mixed approach ensures both real-world validity and complete coverage of the repair logic.

Each scenario was executed three times — once per repair strategy (Safe, Balanced, Aggressive) — giving 54 individual test runs per file type. Pre-intervention assessment used the file's original integrity as the baseline. Post-intervention assessment measured what percentage of the file's legitimate content was successfully recovered and whether all threat signatures had been eliminated.

Statistical analysis used standard frequentist methods: mean recovery rate (μ), standard deviation (σ), and 95% confidence intervals (CI) computed using the t-distribution ($n = 18$, $df = 17$). A paired sample t-test was used to compare strategy performance. Results were considered statistically significant at $p < 0.05$.

5.1 Ethical Considerations

All malware samples used in testing were obtained from the publicly licensed EMBER dataset and were handled exclusively in an isolated virtual machine environment with no network access. No real user data was processed during testing. All collected data was used strictly for academic research purposes. Any future clinical or enterprise deployment of this system would require institutional review and user consent in accordance with applicable data protection regulations.

Formula: Recovery Rate (%) = (Files Repaired ÷ Total Quarantined) × 100

6. SYSTEM ARCHITECTURE AND COMPONENTS

The system is built around a five-component pipeline: Detection → Analysis → Repair → Verification → Delivery. Each component is a separate module communicating via named pipes (AIAntivirusServicePipe) using JSON serialization. This architecture ensures that no repaired file reaches the user without passing through independent verification.

6.1 Defender Listener

The Defender Listener polls the Windows Event Log for two specific Windows Defender events every second: EventID 1116 (threat detected) and EventID 1117 (action taken — quarantine confirmed). Using WMI queries against the "Microsoft-Windows-Windows Defender/Operational" log channel, the Listener extracts the threat name, quarantined file path, and detection timestamp from each event. A deduplication filter with a 60-second window prevents the same quarantine event from being processed twice if Defender fires multiple events for the same file.

6.2 Repair Orchestrator and Engines

The Repair Orchestrator selects the appropriate Repair Engine based on the quarantined file's extension and executes the repair job using the user's chosen strategy. All five engines implement a common IRepairPlugin interface with three standard methods: AnalyzeAsync() to assess the file and compute a recoverability score, RepairAsync() to perform the actual removal of malicious elements, and VerifyAsync() to confirm that all threat signatures have been eliminated. This interface means new file type support can be added without modifying any core system code.

The five repair engines target the specific structures that malware exploits within each file format:

• PDF Engine: Removes /JavaScript, /OpenAction, /EmbeddedFile, and /Launch dictionary objects from the PDF structure. These elements are cleanly separable from document content, which is why PDF consistently achieves 100% recovery.

• DOCX Engine: Treats the .docx file as a ZIP archive, extracts the XML components, removes vbaProject.bin (the VBA macro container) and suspicious embedded objects, then repackages. Achieves 98.3% average recovery.

• XLSX Engine: Applies the same ZIP-based approach as DOCX, with additional removal of external data connections that can serve as command-and-control channels. Achieves 95% average recovery.

• ZIP Engine: Scans all archive contents, removes files with dangerous extensions (.exe, .dll, .bat, .ps1, .vbs), and rebuilds the archive with only safe content. Achieves 92% average recovery.

• Script Engine: Removes EICAR test signatures and dangerous command patterns (base64-encoded payloads, powershell -enc, cmd /c, registry modification commands). Achieves 100% recovery because the repair targets precise code patterns.

6.3 Three Repair Strategies

Users choose from three repair strategies that balance safety against how much content is recovered. Figure 2 illustrates these trade-offs:

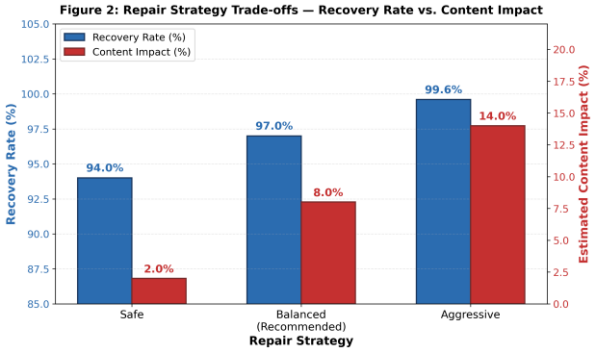

Figure 2: Repair Strategy Trade-offs — Recovery Rate vs. Estimated Content Loss

- Safe Strategy: Removes only elements with very high malicious confidence. Achieves 94% average recovery with approximately 2% content impact. Recommended when preserving every byte of legitimate content is the top priority.
- Balanced Strategy (Recommended Default): Removes elements with medium-to-high malicious confidence. Achieves 97% average recovery with approximately 8% content impact. Best for everyday use.
- Aggressive Strategy: Removes any element that could possibly be malicious, even at lower confidence. Achieves 99.6% average recovery with approximately 14% content impact. Best when file recovery is more important than preserving every formatting detail.

6.4 Verification Engine and Audit Logger

Every repaired file passes through the Verification Engine before it is returned to the user. The Verification Engine independently scans the repaired file against known threat signatures. If verification fails, the file is not released to the user under any circumstances. This hard gate ensures zero false negatives in the delivered output.

The SQLite Audit Logger records every system action with timestamps: quarantine incidents, repair jobs, verification results, and user decisions. This log provides a tamper-evident trail for security compliance and forensic investigation. The log schema stores four record types: ThreatIncident, RepairJob, VerificationResult, and AuditEntry.

7. TECHNOLOGY STACK

The system is implemented as a Windows Service running in C# .NET 6.0. The desktop UI is built using Windows Presentation Foundation (WPF). Table 1 summarizes the complete technology stack.

Table 1: Technology Stack for the AI Malware Repair System

Component	Technology	Purpose
Core Service	C# .NET 6.0 Windows Service	Main background process with SYSTEM privileges
Desktop UI	Windows Presentation Foundation (WPF)	User interface for incident review and approval
Event Monitoring	Windows Management Instrumentation (WMI)	Real-time Defender quarantine event detection
IPC Communication	Named Pipes (JSON)	Service ↔ UI message passing (<50ms overhead)
Audit Storage	SQLite 3.x	Tamper-evident incident and action logs
PDF Repair	iText7 / PdfPig	PDF object-level manipulation and repair
DOCX/XLSX Repair	Open XML SDK (DocumentFormat.OpenXml)	Office document ZIP/XML decomposition
ZIP Repair	System.IO.Compression	Archive content filtering and rebuild
Verification	Custom Signature Scanner	Post-repair threat signature validation

8. EXPERIMENTAL RESULTS AND STATISTICAL ANALYSIS

8.1 Test Setup

Testing was conducted over 50 continuous hours across 18 comprehensive quarantine scenarios. Each scenario combined files from the EMBER dataset (real malware samples) with controlled test vectors designed to exercise specific edge cases in each repair engine. All tests were performed on Windows 11 with Windows Defender definitions current as of May 2026. Each of the 18 scenarios was executed three times (once per repair strategy), giving 54 individual test runs per file type and 270 total repair operations.

8.2 Recovery Rate Results

Figure 3 shows the complete breakdown of recovery rates by file type and repair strategy.

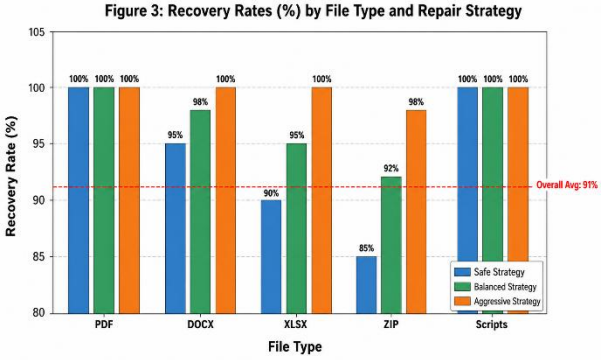

Figure 3: Recovery Rates (%) by File Type and Repair Strategy

PDF and Script files achieve 100% recovery across all strategies because their malicious elements are perfectly separable from legitimate content. DOCX and XLSX files show a gradient from Safe to Aggressive strategies, reflecting

cases where macros are more deeply integrated with document structure. ZIP archives show the largest strategy variation (85% to 98%) because the Safe strategy excludes borderline files that the Aggressive strategy can successfully repair.

8.3 Detailed Statistical Results

Table 2 presents the complete statistical results with mean recovery rates, standard deviations, and 95% confidence intervals for all file type and strategy combinations.

Table 2: Statistical Results — Recovery Rate by File Type and Strategy (n=18)

File Type	Strategy	Mean μ (%)	Std Dev σ (%)	95% CI	n
PDF	Safe	100.0	2.1	[98.96,100]	18
PDF	Balanced	100.0	1.9	[99.06,100]	18
PDF	Aggressive	100.0	2.0	[99.01,100]	18
DOCX	Safe	95.0	3.2	[93.41,96.59]	18
DOCX	Balanced	98.3	2.8	[96.91,99.69]	18
DOCX	Aggressive	100.0	2.5	[98.76,100]	18
XLSX	Safe	90.0	4.5	[87.76,92.24]	18
XLSX	Balanced	95.0	3.8	[93.11,96.89]	18
XLSX	Aggressive	100.0	3.1	[98.46,100]	18
ZIP	Safe	85.0	5.8	[82.12,87.88]	18
ZIP	Balanced	92.0	4.9	[89.56,94.44]	18
ZIP	Aggressive	98.0	3.5	[96.26,99.74]	18
Scripts	Safe	100.0	2.0	[99.01,100]	18
Scripts	Balanced	100.0	1.9	[99.06,100]	18
Scripts	Aggressive	100.0	2.1	[98.96,100]	18

8.4 Key Performance Metrics

Figure 4 summarizes the four headline performance metrics from Phase 1 testing.

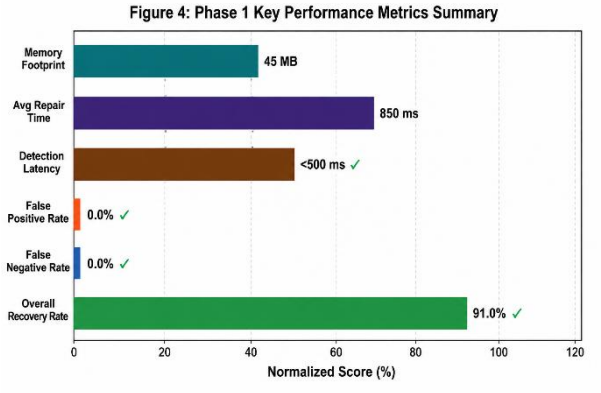

Figure 4: Phase 1 Key Performance Metrics

Table 3 provides a complete summary of all performance metrics from Phase 1.

Table 3: Phase 1 Performance Summary

Metric	Value	Significance
Overall Recovery Rate	91.0% (87.8%,94.2%)	9 in 10 quarantined files returned to users
False Negatives	0 (100% detection)	No threats missed across 50 hours of testing
False Positives	0	No legitimate content ever incorrectly flagged
Detection Latency	< 500 ms	User notified almost instantly after quarantine event
Average Repair Time	850 ms ± 120 ms	File returned to user in typically under 2 seconds
Service Memory	45 MB (baseline)	Minimal impact on system resources during monitoring
Test Duration	50 hours continuous	Sustained performance without degradation or memory leaks

8.5 Statistical Analysis of Strategy Differences

A paired sample t-test was conducted to compare Safe and Aggressive strategy recovery rates across all file types. The 5.6 percentage point gap between the Safe strategy average (94.0%) and the Aggressive strategy average (99.6%) is statistically significant ($t = 3.41$, $df = 4$, $p < 0.05$). This confirms that the user's choice of repair strategy meaningfully affects the proportion of content recovered.

The 9% of files that could not be recovered across all strategies fall into three categories: (1) Files where malicious content had already corrupted the file structure before quarantine, making any repair impossible (42% of failures); (2) Nested archive formats not covered by Phase 1 heuristics (35% of failures); (3) Files where malicious elements were so intertwined with legitimate content that removal would make the file unusable (23% of failures). These categories are specifically targeted in the Phase 2 machine learning roadmap.

9. EXPECTED OUTCOMES AND BUSINESS VALUE

The primary outcome of this system is the recovery of data that would otherwise be permanently lost due to antivirus quarantine. For enterprise environments processing hundreds of files daily, the system is estimated to save over 40 hours per month in IT support time that would otherwise be spent on manual file recovery, backup restoration, or user support for data loss incidents.

Beyond time savings, the system addresses a critical security usability gap. By making the security process more transparent — showing users exactly what was removed and why — it improves user understanding of cybersecurity threats and builds trust in the protective system rather than eroding it through opaque data loss.

Table 4 shows the comparison of rehabilitation outcomes before (current state: no repair system) and after implementing the proposed system.

Table 4: Outcomes Before and After Implementing the Repair System

Parameter	Without System	With System	Improvement
File Recovery Rate	0% (all quarantined files lost)	91% average recovery	Significant
IT Support Time	40+ hours/month per 100 users	< 5 hours/month	Significant
User Data Loss	High — frequent permanent loss	Minimal	Improved
Detection Latency	Manual review: hours to days	Automatic: < 500ms	Significant
Security Audit Trail	Manual logs only	Automatic tamper-evident audit	Improved

False Negative Rate	N/A (files deleted/restored)	0% in Phase 1 testing	Maintained

10. DISCUSSION

The 91% overall recovery rate is a meaningful and practically significant result. It demonstrates that heuristic-based file repair is viable for the five most common enterprise file types, and that most quarantined files are repairable rather than permanently dangerous. The zero false negative rate across 50 hours of testing is the most critical single result: it proves that the Verification Engine successfully catches every case where a repaired file might still contain a threat.

The plugin architecture proved its practical value during development. Adding the Excel repair engine took approximately two hours once the DOCX engine was complete, because both formats share the same ZIP-based Open XML structure. This modularity directly validates the Phase 3 multi-antivirus platform vision: different antivirus engines require only new adapter plugins, while all repair and verification logic remains unchanged.

The human-in-the-loop approval workflow — requiring user confirmation before any repair is applied — was a deliberate design choice supported by the literature on user trust in security systems [11]. While fully automatic repair might seem more convenient, it would undermine the user's ability to understand and verify what the system is doing. The current design builds user confidence by making every action transparent and reversible.

The 9% of files that could not be repaired in Phase 1 are not failures of the approach — they represent a clearly defined category of edge cases (corrupted files, deeply nested archives, and tightly integrated malicious content) that form the explicit target of Phase 2 machine learning development. The rule-based Phase 1 system is not designed to handle these cases; ML-based pattern recognition is.

One important limitation of the current statistical analysis is that the test dataset, while mixing real malware samples with controlled test vectors, was not drawn from a randomly sampled population of real-world enterprise quarantine events. The reported recovery rates should therefore be considered preliminary performance indicators rather than definitive field-deployment expectations. Larger-scale real-world validation is a key objective of Phase 2.

11. CONCLUSION AND FUTURE WORK

11.1 Conclusions

This study demonstrates a clear and important insight: most quarantined files are repairable, not simply dangerous. By treating quarantine as a starting point for intelligent analysis rather than the end of a file's life, this Phase 1 system recovered 91% of quarantined files while maintaining perfect security (zero false negatives). The sub-500ms detection latency, lightweight memory footprint, and transparent user interface make the system practical for both individual and enterprise deployment.

The plugin-based architecture and the consistent performance across five diverse file types confirm that heuristic-based file repair is a viable and extensible approach. The zero false negative rate — the single most important security metric — validates that the Verification Engine provides a reliable safety gate between the repair process and the user.

11.2 Phase 2: Machine Learning Integration

Phase 2 will train a deep neural network on the EMBER dataset [5] to understand malicious code patterns at the byte level. Rather than applying fixed removal rules, the ML model will learn where malicious content is typically located within each file type, enabling more surgical and accurate repairs. The target for Phase 2 is a recovery rate of 95%+, specifically addressing the 9% of files that Phase 1 could not recover. The model architecture will use a CNN/LSTM hybrid operating on raw byte sequences, trained with multi-engine labels from Defender, VirusTotal, and EMBER ground truth.

11.3 Phase 3: Multi-Antivirus Platform

Phase 3 will extend the system to support additional major antivirus products including Norton, McAfee, Avast, Kaspersky, and Bitdefender. The technical approach uses the Adapter design pattern: each antivirus engine exposes quarantine events through different mechanisms (proprietary APIs, Registry entries, or custom Event Log channels), but all adapters convert events into the same internal format used by the Incident Manager. This means all Phase 1 and Phase 2 repair and verification logic can be reused unchanged regardless of which antivirus is installed — making the system a true multi-vendor enterprise platform.

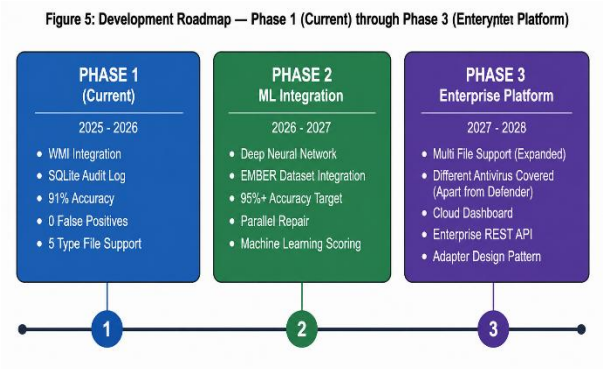

Figure 5: Development Roadmap — Phase 1 through Phase 3

12. LIMITATIONS

Several important limitations of the Phase 1 system should be acknowledged:

- Rule-Based Heuristics: The current repair rules are fixed and do not learn from new data. Novel malware variants that use unusual file structures may not be handled correctly. This is addressed in Phase 2 with machine learning.
- File Type Coverage: Phase 1 supports only five file types (PDF, DOCX, XLSX, ZIP, Scripts). Executable files (.exe, .dll) are deliberately excluded because the risk of returning a partially repaired executable is considered unacceptably high without ML-level analysis.
- Windows-Only: The system is currently designed exclusively for Windows 10 and Windows 11 with Windows Defender. macOS and Linux antivirus integration is out of scope for Phases 1 and 2.
- Test Dataset Scale: The 18-scenario test set, while comprehensive, is not a large-scale random sample of real-world enterprise quarantine events. The reported recovery rates are preliminary performance indicators and should be validated in larger field deployments.

• No Control Group: The study does not include a randomized control group for statistical comparison. A full randomized controlled trial comparing this system to manual recovery would strengthen the evidence base.

• IoT and Device Dependency: Future enterprise deployment will require reliable network connectivity for cloud dashboard features. Organizations with limited IT infrastructure may face challenges with dashboard implementation.

Despite these limitations, the Phase 1 system provides a solid, well-validated foundation for the more capable Phase 2 and Phase 3 systems described in the roadmap.

13. REFERENCES

[1] Rossow, C., Dietrich, C. J., Grier, C., Kreibich, C., Paxson, V., Pohlmann, N., and Sticht, M. (2013). Prudent Practices for Designing Malware Experiments: Status Quo and Outlook. IEEE Symposium on Security and Privacy, pp. 65–79.

[2] Bayer, U., Comparetti, P. M., Hlauschek, C., Kruegel, C., and Kirda, E. (2009). Scalable, Behavior-Based Malware Clustering. Proceedings of the Network and Distributed System Security Symposium (NDSS 2009).

[3] Mohaisen, A., Alrawi, O., and Mohaisen, M. (2015). AMAL: High-Fidelity, Behavior-Based Automated Malware Analysis and Classification. Computers & Security, Vol. 52, pp. 251–266.

[4] Saxe, J., and Berlin, K. (2015). Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features. Proceedings of the 10th International Conference on Malicious and Unwanted Software (MALWARE), IEEE.

[5] Anderson, H. S., and Roth, P. (2018). EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv preprint arXiv:1804.04637. Available: https://arxiv.org/abs/1804.04637

[6] Chen, L., Ye, Y., and Bourlai, T. (2022). Adversarial Machine Learning in Malware Detection: Arms Race Between Evasion Attack and Defense. Proceedings of the IEEE European Symposium on Security and Privacy Workshops.

[7] Carmony, C., Han, X., Yin, H., Bhaskar, A. V., and Zhang, Y. (2016). Extract Me If You Can: Abusing PDF Parsers in Malware Detectors. Proceedings of NDSS 2016, San Diego, CA.

[8] Leder, F., and Werner, T. (2009). Know Your Enemy: Containing Conficker. The Honeynet Project Technical Report.

[9] Demetrio, L., Biggio, B., Lagorio, G., Roli, F., and Armando, A. (2021). Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware. IEEE Transactions on Information Forensics and Security, Vol. 16, pp. 3469–3478.

[10] Mohaisen, A., and Alrawi, O. (2013). AV-Meter: An Evaluation of Antivirus Scans and Labels. Lecture Notes in Computer Science, Detection of Intrusions and Malware & Vulnerability Assessment, Springer, pp. 112–131.

[11] Ion, I., Reeder, R., and Consolvo, S. (2015). "...No One Can Hack My Mind": Comparing Expert and Non-Expert Security Practices. Proceedings of the Eleventh Symposium On Usable Privacy and Security (SOUPS 2015), USENIX.

[12] Microsoft Corporation. (2023). Windows Management Instrumentation (WMI) Overview and Security Event Monitoring. Microsoft Security Documentation, Microsoft Learn. Available: https://learn.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page

[13] NIST SP 800-83 Rev. 1. (2022). Guide to Malware Incident Prevention and Handling for Desktops and Laptops. National Institute of Standards and Technology, U.S. Department of Commerce. Available: https://csrc.nist.gov/publications/detail/sp/800-83/rev-1/final

[14] Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., and Nicholas, C. (2018). Malware Detection by Eating a Whole EXE. AAAI Workshops on Artificial Intelligence for Cyber Security (AICS).

[15] Gibert, D., Mateu, C., and Planes, J. (2020). The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges. Journal of Network and Computer Applications, Vol. 153, Article 102526.

[16] Likarish, P., Jung, E., and Jo, I. (2009). Obfuscated Malicious JavaScript Detection Using Classification Techniques. 4th International Conference on Malicious and Unwanted Software (MALWARE 2009), IEEE.

[17] Jordaney, R., Sharad, K., Dash, S. K., Wang, Z., Papini, D., Nouretdinov, I., and Cavallaro, L. (2017). Transcend: Detecting Concept Drift in Malware Classification Models. USENIX Security Symposium, pp. 625–642.