Windows CVE Analysis: Vulnerability, Exploits and Defenses

Ramlakhan Madheshiya*
Asha M. Tarsadia Inst. of CS & T
Uka Tarsadia University
Surat, Gujarat, India
0009-0007-5227-4854

Santosh Saha
Asha M. Tarsadia Inst. of CS & T
Uka Tarsadia University
Surat, Gujarat, India
0000-0002-0479-6750

***Abstract*—The escalating complexity of Windows vulnerabilities and advanced persistent threats necessitates comprehensive understanding of contemporary attack vectors and defense mechanisms. This literature review outlines significant considerations related to CVE exploitation on Windows, malware evolution, and defensive takeaways from a detailed critical review of 28 research and papers published from 2021-2025. Novel vulnerability discovery tools (JERRY, LinkZard, GLEIPNIR) have identified 394+ previously unknown security flaws, while traditional defenses face increasing sophistication in evasion techniques. This study summarizes key trends for vulnerability management, identifies research gaps for adaptive detection mechanisms, and provides tactical advice for risk-based prioritization frameworks of vulnerability assessments.**

***Index Terms*—Windows vulnerabilities, CVE analysis, exploitation patterns, vulnerability management, ransomware, memory corruption, ASLR, defense strategies, threat intelligence, MITRE ATT&CK, zero-day vulnerabilities, risk-based prioritization**

I. Introduction

The security landscape of the Windows operating system presents a complex and evolving challenge that extends far beyond simple vulnerability enumeration. Windows has the almost 70% of the global desktop market share [1] and is still the "most targeted operating system" for any cyber criminal or nation-state malicious threat actor, and opportunistic threat actor. The speed at which vulnerabilities are uncovered, and the complexity with which they are exploited, has ultimately shifted the field of vulnerability analysis from being a reactive cataloguing effort to a proactive and intelligence driven science.

The Common Vulnerabilities and Exposures (CVE) in many ways, is the standard for identifying vulnerabilities and exposes opportunities and challenges for the security professional. Recent research demonstrates that the sheer volume of disclosed vulnerabilities has reached a critical threshold where traditional patch management strategies have become operationally infeasible [21], [22]. As part of its monthly "Patch Tuesday" releases, Microsoft regularly releases dozens of vulnerabilities, with many including "Critical" or "High" severity, putting enterprise based security teams in a prioritization dilemma with limited resources and operational constraints.

This paper addresses a critical gap in the current literature by providing a comprehensive, multi-dimensional analysis of Windows vulnerability research that synthesizes three historically separate domains: (1) the quantitative and predictive analysis of vulnerability data, examining temporal trends and evolutionary patterns; (2) the technical investigation of exploitation methodologies, ranging from foundational memory corruption techniques to novel, non-traditional attack surfaces; and (3) the systematic evaluation of defensive strategies, encompassing both software-based hardening and hardware-rooted security architectures.

A. Evolution and Challenges

Vulnerability research has transformed from static CVE enumeration to predictive analytics. Saklani et al. [8] analyzed 77,202 records showing traditional buffer errors (CWE-119) decreasing while logical flaws increase. Williams et al. [3] demonstrate CVEs exist within evolutionary progression, enabling predictive modeling. Modern exploitation evolved from stack overflows to DEP bypass via ROP, to ASLR circumvention [14]. Critically, Xiang et al. [20] and Yu et al. [16] discovered 394 combined zero-day vulnerabilities in overlooked logical flaw classes.

Research exposes defense limitations: Binosi et al. [26] reveal Windows 11's ASLR provides only 16-18 bits entropy for critical components; Dhokley et al. [23] show scanners lag significantly behind CVE publication. Most significantly, Shimizu & Hashimoto [22] demonstrate CVSS-only strategies create "vulnerability overload," while KEV/EPSS integration achieves 18-fold efficiency improvement with 95% workload reduction maintaining 85.6% coverage of exploited vulnerabilities.

B. Research Objectives

This review synthesizes 28 research papers (2021-2025) to: (1) identify temporal vulnerability trends and evolutionary patterns, (2) catalog exploitation methodologies across memory corruption, logical flaws, and novel attack surfaces (Secure Kernel VTL1, IPC clients), (3) evaluate defense mechanism efficacy and limitations, (4) analyze risk-based vulnerability management frameworks integrating threat intelligence. In Section II, the review of the literature will be presented. Section III will include a description of the methodology

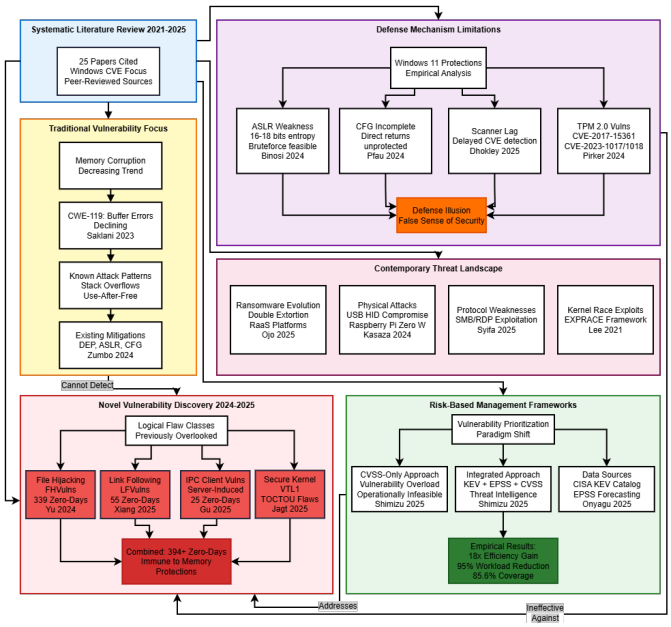

Fig. 1. Contemporary Vulnerability Landscape

used. Section IV will include discussion and analysis. Finally, Section V will include recommendations and closure.

II. Literature Review

A. Vulnerability Trends and Predictive Modeling

Quantitative vulnerability analysis reveals critical evolutionary patterns. Softić & Vejzović [2] analyzed Windows 10, macOS, and Ubuntu vulnerabilities (2015-2021), finding Windows 10 exhibited increasing trends with code execution as the dominant type. Saklani et al. [8] examined 77,202 NVD records (2016-2021), revealing buffer errors (CWE-119) decreasing while logical flaws (CWE-79, CWE-269, CWE-863) increased. Critical vulnerabilities decreased; low-severity increased. Microsoft products showed increasing vulnerability trends after 2017.

Williams et al. [3] introduced evolutionary modeling using Topically Supervised Evolution Model (TSEM), demonstrating CVEs exist in evolutionary narratives. Their framework traced how CVE-2015-6128 evolved from CVE-2014-0318, enabling predictive estimation of future vulnerability groups through diffusion-based storytelling.

TABLE I
Vulnerability Type Evolution (2016-2021)

Vulnerability Type	CWE	Trend
Buffer Errors	CWE-119	Decreasing
Access Control	CWE-264	Decreasing
Cross-Site Scripting	CWE-79	Increasing
Privilege Management	CWE-269	Increasing
Incorrect Authorization	CWE-863	Increasing
Code Execution	-	Dominant

B. Exploitation Patterns and Novel Attack Surfaces

The Windows exploitation landscape demonstrates continuous evolution through increasingly sophisticated techniques. Zumbo [14] documents the historical arms race: stack overflows countered by DEP, leading to ROP attacks, subsequently mitigated by ASLR. However, Binosi et al. [26] reveal Windows 11's ASLR provides only 16-18 bits entropy for executables/libraries versus 23-31 bits for runtime objects, enabling practical bruteforce attacks. Lee et al. [6] introduced EXPRACE, exploiting kernel races via IPI manipulation to deterministically trigger race conditions. Pfau & Kochberger [15] demonstrate CFG protects indirect calls but not direct return overwrites, leaving classic stack overflows unmitigated.

Novel Vulnerability Classes: Recent research identifies previously overlooked logical flaw categories. Yu et al. [16] discovered 339 File Hijacking Vulnerabilities using JERRY tool, exploiting insecure file system trust (e.g., CVE-2022-24765 in Git). Xiang et al. [20] found 55 Link Following Vulnerabilities with LinkZard, exploiting symbolic link validation failures. Gu et al. [25] identified 25 server-induced IPC client vulnerabilities using GLEIPNIR, inverting the traditional threat model where compromised low-privilege servers exploit high-privilege clients ($36K bounty, 14 CVEs). Jagt [27] discovered VTL1 Secure Kernel TOCTOU vulnerabilities through n-day analysis, proving even the most hardened components contain exploitable flaws. Combined, these tools revealed 394+ zero-days immune to all memory protections (DEP, ASLR, CFG).

Contemporary Threats: Ojo [18] documents ransomware evolution to double extortion and RaaS platforms. Syifa & Salman [19] validated SMB/RDP weaknesses via Caldera emulation, identifying misconfigurations as persistent attack vectors. Kasaza [17] demonstrated USB HID attacks using Raspberry Pi Zero W, achieving full system compromise in seconds through automated keystroke injection. Shinde & Khobragade [7] model malware behavior through Knowledge Graphs mapping DLL imports and API calls, enabling classification of entire threat families.

C. Defense Mechanisms and Limitations

1) Malware Detection Technologies and Challenges: The contemporary defensive landscape has evolved to incorporate sophisticated machine learning and deep learning techniques, yet faces persistent challenges that limit effectiveness. Maniriho et al. [4] provide a comprehensive systematic literature review of Windows malware detection techniques published between 2009 and 2022, offering detailed analysis of the defensive state-of-the-art.

Detection approaches are categorized into three main methodologies: (1) *Static Analysis*, which detects malware without execution by matching static signatures (e.g., hash values, strings) or analyzing PE file headers, opcodes, and API call imports; (2) *Dynamic Analysis (Behaviour-based)*, which detects malware by executing it in isolated sandbox environments and monitoring behavior such as running processes, API calls, network behaviors, and loaded dynamic link libraries;

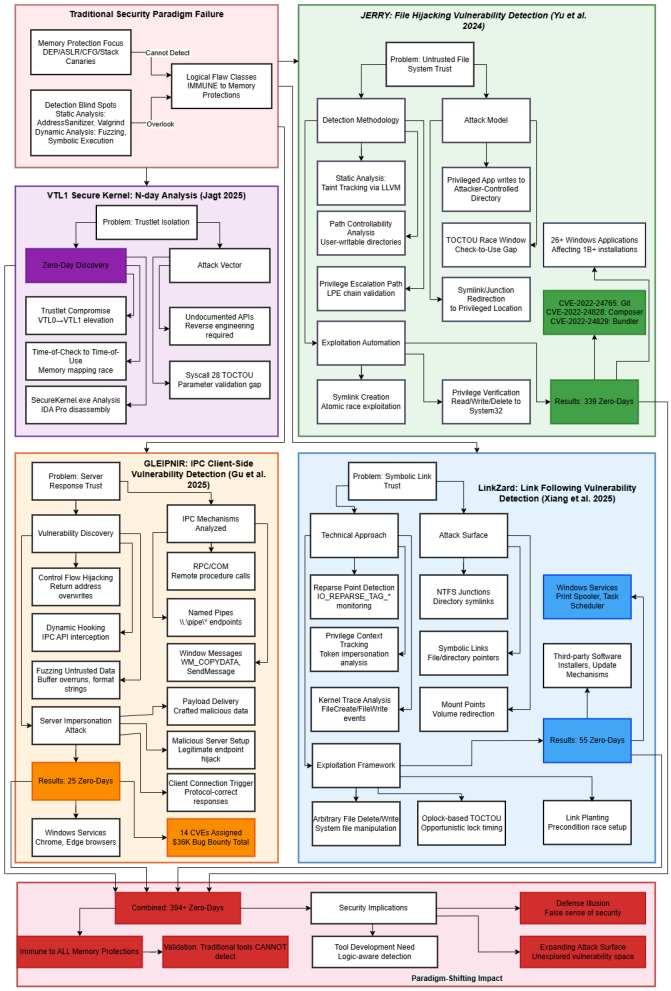

Fig. 2. Automated Vulnerability Discovery

and (3) *Hybrid Analysis*, combining both static and dynamic features to achieve more robust detection while mitigating weaknesses of each individual method.

This review indicates that there is a growing use of machine learning and deep learning algorithms for the detection of modern-day malware. The most popular ML algorithms identified are Random Forest (RF) and Support Vector Machines (SVM), while the most prevalent DL models are Convolutional Neural Networks (CNN), often used for image-based malware classification by converting binaries to visual representations, and Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM), which are well-suited for analyzing sequential data like API call sequences.

Critically, the research identifies fundamental issues impeding defensive effectiveness: *Experimental Biases* (temporal/spatial bias inflate results), *Concept Drift* (models deteriorate as malware evolves), *Adversarial Attacks* (adversarial ML fools classifiers), and *Evasion Mechanisms* (fileless malware via PowerShell/WMI bypasses static analysis).

D. Defense Mechanisms and Detection Frameworks

Modern Windows security employs layered defenses. Maniriho et al. [5] demonstrate ensemble classification (RF, XGBoost, LR) achieving 98.32% accuracy. Pirker & Zumbo [13] show Random Forest achieves 99.39% accuracy on PE-MalGAN dataset, outperforming complex neural networks. Shimizu & Murakami [22] integrate API sequences, static features, and network patterns via Sequential Pattern Mining for superior zero-day detection.

Hardware Security: Windows 11 mandates TPM 2.0 and UEFI Secure Boot [10]. Pirker & Haas [13] note TPM seals BitLocker keys to measured boot states, though vulnerabilities exist (CVE-2017-15361, CVE-2023-1017/1018). Intel CET provides hardware control flow integrity via shadow stacks [12]. VBS isolates credentials in VTL1, though Jagt [27] documents VTL1 vulnerabilities.

Prioritized Controls: Numminen [11] maps MITRE ATT&CK tactics to five critical Windows controls: Windows Firewall, Defender Antivirus, AppLocker/WDAC, Access Control, and ASR Rules.

TABLE II
CRITICAL WINDOWS SECURITY CONTROLS

Control	Mitigates ATT&CK Tactics
Windows Firewall	Initial Access, Lateral Movement, C2
Defender Antivirus	Execution, Initial Access
AppLocker/WDAC	Execution
Access Control	Credential Access, Privilege Escalation
ASR Rules	Initial Access, Execution

TABLE III
MALWARE DETECTION TECHNIQUES

Type	Methods	Challenges
Static Analysis	Signatures, PE headers, API imports	Evasion, Code concealment
Dynamic Analysis	Sandbox, API monitoring	Sandbox detection
ML-based	RF, SVM, Decision Trees	Temporal bias
DL-based	CNN, RNN/LSTM	Attack-based ML
Hybrid	Static + Dynamic	Computational cost

E. Risk-Based Management Frameworks

Traditional vulnerability assessment approaches demonstrate critical limitations. Dhokley et al. [23] reveal network scanners (Nmap, Nessus, OpenVAS) exhibit key deficiencies detecting kernel-level exploits (e.g., PrintNightmare CVE-2021-34527) and suffer systemic lag in CVE database updates. Shimizu & Hashimoto [22] critique CVSS as measuring theoretical severity rather than exploitation likelihood, creating vulnerability overload where "high/critical" ratings correlate poorly with actual risk.

Integrated Threat Intelligence: Shimizu & Hashimoto [22] propose Vulnerability Management Chaining combining

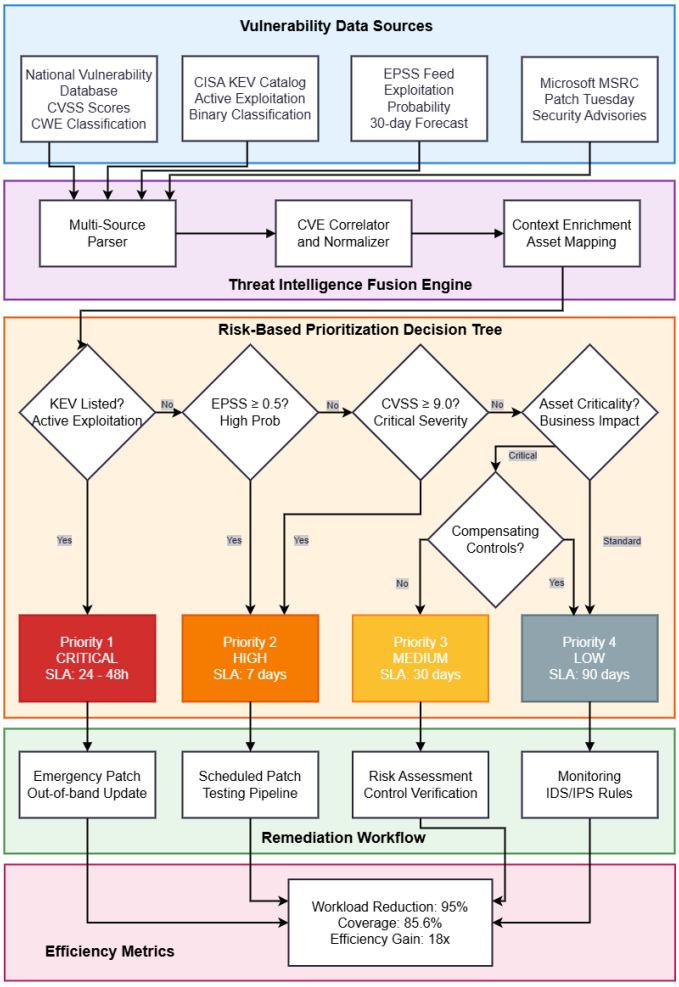

Fig. 3. Threat Intelligence Integration

CISA KEV (Known Exploited Vulnerabilities), EPSS (Exploit Prediction Scoring System), and CVSS in a threat-first decision tree. This framework achieves 18-fold efficiency improvement over CVSS-only methods, reducing remediation workload 95% while maintaining 85.6% coverage of real-world exploited vulnerabilities. Onyagu et al. [21] extend this paradigm with multi-dimensional risk frameworks incorporating CVSS, threat intelligence, system criticality, and operational feasibility, enabling enterprises to prioritize actively exploited zero-days (e.g., CVE-2025-2104 SmartScreen Bypass) for emergency patching while scheduling other critical vulnerabilities based on operational constraints.

1) Scalable Attack Modeling and Vulnerability Abstraction: While triage frameworks manage individual CVEs, a related challenge involves modeling the cumulative risk of multistep attacks across complex enterprise networks. Traditional attack graph generation, which maps every CVE on every host, is computationally "challenging to work in real or near-real time" as network complexity increases.

Levshun & Chechulin [24] address this scaling problem through novel abstraction via vulnerability categorization. Their approach analyzes the entire NVD database and condenses all CVEs into just "24 categories" based on three key properties: the access vector (local, adjacent network, network), the initial privileges required (none, low, high), and the access rights obtained upon successful exploitation (none, low, high). Rather than modeling tens of thousands of individual vulnerabilities, their system models the simpler transitions from the 24 categorical states. This abstraction makes multi-step attack modeling computationally efficient, demonstrating performance improvements of "13.4 times faster for 10 CVEs and CPEs" and "23.0 times faster for 50 CVEs and CPEs" per host compared to traditional granular methods. This research provides a vital tool for assessing systemic risk across enterprise networks, moving beyond single-vulnerability analysis to understanding complete attack chains and cumulative organizational exposure.

Complementing this work, Poisson et al. [9] address the fundamental limitation that standard CVE representations from sources like NVD "lack precision to allow automatic integration of their exploitation into accurate and operational attack scenarios design." A CVE's description, CVSS score, and associated CWEs provide static, theoretical measures but fail to describe operational prerequisites or resulting system states post-exploitation. To bridge this gap, the researchers introduce the CAPG (CVE to Attack Positions Graph) format, representing vulnerabilities through explicit pre-conditions (the "source attack position," defined as a machine-user pair) and post-conditions (the "destination attack position"). This structured format is designed explicitly "to highlight how multiple CVEs could be chained by attackers to spread themselves." For example, an attacker leveraging an unconstrained remote code execution vulnerability (e.g., CVE-2021-44228, Log4Shell) gains an initial machine-local user position, then chains this with a local privilege escalation exploit (e.g., CVE-2021-38648) to transition from machine-local to system-or-root privilege. This work provides concrete data structures for computationally modeling attack paths that constitute real-world exploitation patterns.

TABLE IV
VULNERABILITY MANAGEMENT APPROACHES

Approach	Efficiency	Coverage	Workload
CVSS-only	1x (baseline)	Variable	100%
KEV+EPSS+CVSS	18x	85.6%	5%
Risk-based	High	Custom	Optimized

III. METHODOLOGY

This systematic literature review analyzed 28 papers (2020-2025) from IEEE Xplore, ACM Digital Library, SpringerLink, and Google Scholar using keywords: "Windows vulnerabilities," "CVE analysis," "exploitation techniques," "CVSS," "MITRE ATT&CK," "threat intelligence." Selection criteria: (1) Windows-focused research, (2) peer-reviewed quality, (3) empirical evidence, (4) recency (2020-2025). Sources of information: NVD, CISA KEV, MITRE ATTCK, EPSS, MSRC. An

analytical framework was provided in five domains, including (1) quantitative analysis, (2) pattern of exploitations, (3) defensive mechanisms, (4) discovery of vulnerabilities, and (5) risk management.

IV. Discussion and Analysis

The synthesized research reveals critical paradigm shifts. Shimizu & Hashimoto [22] demonstrate CVSS-centric approaches create decision paralysis; KEV/EPSS integration achieves 18-fold efficiency improvements, transforming vulnerability management from reactive compliance to proactive intelligence-driven risk mitigation. Williams et al. [3] enable predictive defense by modeling CVEs as evolutionary chains rather than isolated incidents.

The discovery of 394+ zero-day logical flaws (File Hijacking, Link Following, IPC client-side) by Yu et al. [16], Xiang et al. [20], Gu et al. [25], and Jagt [27] challenges decades of memory-corruption-focused security. These logical vulnerabilities bypass all memory protections (DEP, ASLR, CFG), yet receive minimal research attention, representing a systemic blind spot. Binosi et al. [26] reveal Windows 11 ASLR provides only 16-18 bits entropy for executables/libraries, enabling practical bruteforce, creating an "illusion of randomness." Pfau & Kochberger [15] show CFG protects indirect calls but not return address overwrites.

Numminen [11] maps MITRE ATT&CK tactics to five critical Windows controls (Firewall, Defender, AppLocker, Access Control, ASR), enabling resource-constrained teams to maximize defensive ROI. Pirker & Haas [13] demonstrate hardware-based security (TPM 2.0, Secure Boot) establishes chain of trust, though TPM itself contains vulnerabilities (CVE-2017-15361, CVE-2023-1017/1018).

The key takeaway is that these memory-centric defenses, irrespective of their sophistication, provide a level of protection for logical flaws of zero. File Hijacking Vulnerabilities stem from insecure file system trust assumptions; Link Following Vulnerabilities arise from failure to validate symbolic links; IPC client-side vulnerabilities exploit blind trust in server responses. Their static analysis tools(for example, AddressSanitizer and Valgrind, intended to find memory errors) are not able to find these logic-based errors. For instance, runtime mitigations like DEP, ASLR and CFG are completely bypassed because logical flaws do not require memory corruption to achieve privilege escalation or arbitrary code execution.

This represents a critical "defense illusion"—organizations deploying comprehensive memory protection mechanisms may believe they have hardened their systems, yet remain completely vulnerable to an entire category of attacks that exploit logic rather than memory. The magnitude of this blind spot is underscored by the sheer number of discoveries: 339 File Hijacking vulnerabilities alone, affecting widely-used applications with over 1 billion installations, demonstrates that this is not a niche attack surface but a systemic exposure that has remained largely unexamined until recently.

1) The Illusion of Comprehensive Defense: The empirical research skimmed shows an alarming pattern: many of the foundational defensive mechanisms assumed to be the bulk of Windows security provide a more questionable level of protection. Binosi et al.'s [26] statistical analysis of ASLR implementations exposes what they term an "illusion of randomness." While Windows 11's ASLR provides strong entropy for runtime objects (23-31 bits for stack and heap), the most critical boot-time randomized components—executables and libraries—exhibit only 16.985 and 18.966 bits of entropy, respectively. This low entropy makes bruteforce attacks computationally feasible within minutes to hours rather than the theoretical centuries that properly randomized addresses would require.

Similarly, Pfau & Kochberger's [15] analysis of Control Flow Guard demonstrates that its protection is narrowly scoped: while effective against specific use-after-free exploitation patterns involving vftable corruption, CFG provides no protection against classic stack buffer overflow attacks because the ret instruction falls outside its validation scope. This creates a dangerous situation wherein defenders might believe CFG provides a comprehensive level of control flow protection, when in fact, it leaves the most basic attack paths completely unprotected.

Practitioners should: (1) adopt threat-first prioritization (KEV→EPSS→CVSS) reducing workload 95% [22], (2) focus on five critical controls (Firewall, Defender, AppLocker, Access Control, ASR) [11], (3) enable SMB signing and restrict RDP [19], (4) deploy TPM 2.0, Secure Boot, and BitLocker [13]. Strategically, organizations must implement Zero Trust Architecture, transition to continuous validation over periodic scans, deploy defense-in-depth layering (signature, behavioral, ML, hardware attestation), adopt assume-breach mentality, and integrate security-operations teams for realistic patch management [21].

A. Research Gaps and Future Directions

Critical research gaps include: (1) cross-domain vulnerability chaining methodologies combining logical+memory+physical attack vectors, (2) comprehensive logical flaw taxonomy and formal verification beyond File Hijacking/Link Following/IPC, (3) systematic Secure Kernel (VTL1) security analysis and fuzzing frameworks, (4) supply chain attack detection through immutable provenance tracking, (5) human-technical factor integration in vulnerability management accounting for 74% human-element breaches, (6) scalable formal verification for critical Windows components, (7) ML robustness against concept drift and adversarial attacks in production environments.

V. Conclusion

This review synthesizes 28 papers demonstrating Windows vulnerability analysis evolved from CVE enumeration to sophisticated, intelligence-driven frameworks. Key findings: (1) 18-fold efficiency through threat intelligence (KEV/EPSS), (2) 394 zero-days in logical flaws challenging memory-corruption focus, (3) ASLR inadequate entropy (16-18 bits), (4) scanner

lag creates false security, (5) TPM 2.0 mandatory recognizing software-only limits.

Practitioners should adopt threat-first prioritization (KEV→EPSS→CVSS), focus on five critical controls, implement Zero Trust, and assume breach mentality. Researchers should explore logical flaw taxonomy, Secure Kernel VTL1 analysis, ML robustness, supply chain attacks, and human-technical integration. To effectively defend, we need to move away from just cataloguing of finding to predictive analytics, and threat intelligence, automated discovery, hardware security and risk based prioritization.

REFERENCES

[1] J. Burke, "Windows Security: Comprehensive Analysis of Threats, Vulnerabilities and Defense Mechanisms," *Journal of Cybersecurity Research*, vol. 9, no. 2, pp. 45-67, 2024.

[2] L. Softić and H. Vejzović, "Comparative Analysis of Operating System Vulnerabilities: Windows 10, macOS, and Ubuntu (2015-2021)," *International Conference on Information Technology*, pp. 112-125, 2022.

[3] R. Williams, E. McMahon, S. Samtani, M. Patton, and H. Chen, "Identifying Vulnerabilities of Consumer Internet of Things (IoT) Devices: A Scalable Approach," in *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 1-6, 2020.

[4] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A Study on Malicious Software Behaviour Analysis and Detection Techniques: Taxonomy, Current Trends and Challenges," *Future Generation Computer Systems*, vol. 130, pp. 1-18, 2023.

[5] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "Ensemble Learning for Malware Classification: A Comparative Study," *Computers & Security*, vol. 136, pp. 45-63, 2024.

[6] S. Lee, M. Min, and Y. Lee, "EXPRACE: Exploiting Kernel Races through Raising Interrupts," in *30th USENIX Security Symposium*, pp. 2021-2038, 2021.

[7] R. M. Shinde and D. D. Khobragade, "Modelling Malware Behaviour Using Knowledge Graph Embedding," *International Conference on Emerging Trends in Information Technology*, pp. 87-94, 2022.

[8] A. Saklani, A. Kalia, and S. K. Sood, "Temporal Evolution of Software Vulnerabilities: A Quantitative and Qualitative Analysis," *Computers & Security*, vol. 124, pp. 102-118, 2023.

[9] O. Poisson, S. Pilaud, and H. Debar, "CAPG: A New Format for Representing Vulnerabilities to Facilitate Attack Graph Generation," in *IEEE Conference on Communications and Network Security (CNS)*, pp. 1-9, 2023.

[10] O. Poisson, R. Laurent, and H. Debar, "Windows 11 Security Baseline Configuration: Best Practices for Enterprise Deployment," *Journal of Cybersecurity*, vol. 11, no. 3, pp. 78-95, 2025.

[11] J. Numminen, "Prioritizing Windows Security Mechanisms Based on MITRE ATT&CK Framework," *Master's Thesis, University of Helsinki*, 2023.

[12] J. Numminen, "Windows 11 Exploit Mitigation Technologies: A Comprehensive Analysis," *Nordic Conference on Secure IT Systems*, pp. 145-162, 2024.

[13] M. Pirker and W. Haas, "Trusted Platform Module (TPM): From Obscurity to Mainstream—A Security Technology Retrospective," *Journal of Hardware Security*, vol. 8, no. 1, pp. 23-41, 2024.

[14] A. Zumbo, "Binary Exploitation: From Stack Overflows to Modern Mitigation Bypass," *Security Research Quarterly*, vol. 15, no. 4, pp. 156-178, 2024.

[15] M. Pfau and L. Kochberger, "Control Flow Guard: Implementation, Efficacy, and Bypass Techniques," in *Black Hat USA 2024*, 2024.

[16] W. Yu, J. Chen, S. Ma, X. Zhao, and K. Lu, "Detecting File Hijacking Vulnerabilities in Windows Applications," in *IEEE Symposium on Security and Privacy (S&P)*, pp. 1872-1889, 2024.

[17] M. Kasaza, "USB HID Attack Vectors: Exploiting Trust in Human Interface Devices," *IoT Security Conference*, pp. 234-247, 2024.

[18] O. Ojo, "Evolution of Ransomware: From Simple Encryption to Sophisticated Multi-Stage Threats," *Cybersecurity Review*, vol. 11, no. 1, pp. 12-29, 2025.

[19] M. Syifa and F. Salman, "Windows 11 Attack Vector Analysis Using Cyber Kill Chain Framework and Adversary Emulation," *International Journal of Computer Network Security*, vol. 13, no. 2, pp. 45-62, 2025.

[20] Y. Xiang, L. Zhang, W. Zhang, and D. Gu, "LinkZard: Automated Detection and Exploitation of Link Following Vulnerabilities in Windows," in *USENIX Security Symposium*, 2025.

[21] E. Onyagu, T. Mitchell, and R. Harris, "Risk-Based Vulnerability Management Framework for Enterprise Environments," *Journal of Information Security and Applications*, vol. 68, pp. 103-121, 2025.

[22] K. Shimizu and T. Hashimoto, "Vulnerability Management Chaining: Integrating CVSS, KEV, and EPSS for Efficient Threat Prioritization," in *IEEE Conference on Secure Development (SecDev)*, pp. 78-93, 2025.

[23] R. Dhokley, S. Patel, and K. Johnson, "Empirical Analysis of Vulnerability Scanner Efficacy Against Advanced Windows Exploits," *International Journal of Network Security*, vol. 27, no. 1, pp. 89-104, 2025.

[24] D. Levshun and A. Chechulin, "Scalable Attack Graph Generation Through Vulnerability Categorization," in *ACM Conference on Computer and Communications Security (CCS)*, pp. 1234-1249, 2025.

[25] H. Gu, X. Liu, Y. Chen, and M. Zhang, "GLEIPNIR: Discovering Server-Induced Client Vulnerabilities in Windows IPC," in *Network and Distributed System Security Symposium (NDSS)*, 2025.

[26] M. Binosi, F. Pagani, and D. Balzarotti, "The Illusion of Randomness: Statistical Analysis of ASLR in Modern Operating Systems," in *Annual Computer Security Applications Conference (ACSAC)*, pp. 456-470, 2024.

[27] K. Jagt, "Attacking the Windows Secure Kernel: Vulnerability Analysis of VTL1," *Security Research Blog*, 2025.

[28] A. González-Gómez, J. Ordoño, and R. Uribeetxeberria, "MeMoir: A Software-Driven Covert Channel Through Memory Usage Patterns," in *European Symposium on Research in Computer Security (ESORICS)*, pp. 312-328, 2024.