

Network Protocol Vulnerability Assessment: A Comparative Analysis of SMB Vulnerabilities in Windows Network Environments

Dongsheng Yang *

College of Software, Beijing University of Aeronautics and Astronautics, Beijing, 430070, China

Abstract. This study explained the development of Server Message Block (SMB) and its composition and role as a communication protocol, and delved into the SMBGhost vulnerability (CVE-2020-0796) in Windows network environments. Through strictly controlled experimental variables, it systematically quantifies the exploit success rate under different configurations and evaluates the protective effectiveness of Microsoft's KB4551762 patch. The experimental results indicate that the exploit success rate for unpatched systems reaches 90%, while applying the patch results in a 37% decline in file transfer performance. To balance security and performance, this paper proposes a lightweight Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) hybrid detection model, which reduces the false positive rate by 63% compared to traditional Snort rules, achieves a detection accuracy of 96%, and has an inference latency of only 1.8 ms. Key findings include precise delineation of the memory corruption boundaries in the srv2.sys driver, as well as empirical validation of a worm propagation rate of 3.2 devices per second, providing an optimized solution for enterprise network defense that considers both security and performance.

1 Introduction

With the accelerated advancement of digital transformation, the SMB, as the core protocol for resource sharing in Windows network environments, has become a key component of enterprise information infrastructure. The SMB communication protocol is a protocol developed by Microsoft and Intel in 1987, primarily as the communication protocol for Microsoft networks [1]. Through the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol, SMB can establish a network connection between two systems, allowing remote file access and resource sharing. At the same time, the development of SMB has also brought vulnerabilities. Exploitable software vulnerabilities may pose a huge threat to personal and even national security, such as the "SMBGhost" vulnerability [2].

Among the Purplefox alerts, 26.4% exploit the SMBGhost vulnerability. Apart from the SMB Ghost exploit, the campaign targets the eternal blue exploit, Microsoft SQL Server (MSSQL), and the SMB null-session vulnerability [3]. In conclusion, SMBGhost reminds us

* Corresponding author: 24371469@buaa.edu.cn

of the modern threat of cyber attacks and the importance of taking preventative steps against them. By implementing cybersecurity measures and being alert to emerging threats, organizations can protect their data, finances, and reputations from harm [4].

This study aims to achieve two core objectives: 1. By analyzing the deep-seated causes of the SMBGhost vulnerability, particularly revealing the memory corruption boundaries in the compressed packet handling process of the `srv2.sys` driver; 2. Creating a corresponding version (Windows 10/11 vs. Server 2016/2019) defense effectiveness evaluation framework to quantify the actual protective effects of patches and configuration hardening.

2 Related Work

2.1 SMB protocol security mechanisms

The SMB vulnerability has various exploitation methods that are complex and variable. Like Vulnerability Verification (Verify and test issues found by vulnerability scanners (including SQL injection, Cross-Site Scripting (XSS), deserialization, remote code execution, directory traversal, etc.) Port Scanning (Use tools like Nmap to scan open ports on hosts and identify potential risks (e.g., exploitation of critical ports such as 445/SMB, 3389/Remote Desktop Protocol (RDP)), Sensitive Directory Scanning(Use tools like Yujian Scanner to detect sensitive directories/files in web systems (e.g., `/admin`, `backup.sql`) and assess exploitation risks), Weak Password Detection(For web systems: Perform weak password brute-force attacks based on user-provided username lists (e.g., `admin/123456`)), Web Admin Backend Risks(Based on user-provided backend information (e.g., login URL), collect and alert on potential vulnerabilities from public sources (e.g., CMS exploits, web editor flaws, middleware vulnerabilities)and so on [5]. Server Message Block version 3 (SMBv3) LZ77-Huffman compression lacks validation for the `CompressionFlag` field, creating an exploitable attack surface. SMB vulnerabilities manifest through multiple attack vectors, including remote code execution (RCE), credential theft, and worm propagation mechanisms.

2.2 Analysis of SMBGhost vulnerability exploitation

The fatal flaw of the original vulnerability (CVE-2020-0796) is located in the `Srv2DecompressData` function - when the `OriginalCompressedSegmentSize` field is tampered with to `0xFFFFFFFF`, an integer overflow will lead to kernel pool corruption. Through Windows Debugger (WinDbg) during the reproduction process in the virtual machine, it was found that when the forged Original Compressed Segment Size equals `0xFFFFFFFF`, an integer overflow occurs in the `Srv2DecompressData` function. The target machine uses Windows 10 (business editions), version 1909 (Updated Jan 2020).

The key innovation of SMBv3 lies in its integration of compression and encryption, which enhances transmission efficiency, but the lack of verification for the compression flag has become a vector for vulnerabilities.

After the attack, the attacker first gained administrator privileges using the `'getsystem'` command, then obtained accounts and passwords through the `'hashdump -i'` command, where the hashed ciphertext was cracked using online database attacks. Then, the attacker enabled Remote Desktop Services, switched to the Windows system's shell using the `'shell'` command, added a user, and assigned it to the `'administrators'` user group. Remote Desktop Services on the target machine were enabled [6]. This attack attacks the Windows 10 operating system versions 1903 and 1909 and Windows server versions 1903 and 1909. By using a buffer overflow, the attacker will gain access to the target computer. This attack is called Remote Control Execution [7].

SMBv3 is vulnerable to remote code execution without authentication, making this vulnerability usable by a worm to spread (WannaCry, NotPetya, and other such happy worms) [8]. An attacker can perform actions as NT Authority system, such as creating users with administrator credentials, overriding passwords, viewing the file system, and anything else that's allowed by a user with administrative privileges [9].

2.3 Limitations of the existing defense mechanism

Currently, the main purpose of malicious attacks on the internet is economic gain. Operators exploit vulnerabilities to spread viruses, encrypting users' computer data files and key system configuration files, which prevents users from working normally. In reality, users are not without recourse when facing viruses. If users regularly perform timely data backups, they can quickly restore their damaged data when subjected to a virus attack, thereby reducing losses.

The current mainstream protection solutions have some contradictions. First, the patches for vulnerabilities conflict with compatibility, which may lead to a decrease in SMB performance. If SMBv3 is disabled, the efficiency of file transfer will be greatly reduced.

3 Methodology

3.1 Experimental environment setup

For the target machine system and the attacking machine system, the research used unpatched Windows 10 1909, patched Windows 10, and Kali Linux 2024.1 running specific Python scripts. For monitoring tools, the research selected Wireshark 4.2 for traffic analysis and WinDbg for crash dump inspection.

The researchers designed a layered cybersecurity experimental environment architecture, integrating attack initiation, target execution, and traffic monitoring into a closed-loop system. The architecture consists of three core modules:

Control Network: Acting as the attack source, it is equipped with a Kali Linux 2024.1 host running custom Python scripts to generate malicious SMBv3 packets. These packets are crafted by tampering with the `OriginalCompressedSegmentSize` field (set to `0xFFFFFFFF`) to exploit an integer overflow vulnerability in the `Srv2DecompressData` function.

Target network: The target network consists of two target systems with different states, which are the objects of attack, used to verify the effectiveness of defense measures (such as security patches): **Unpatched Target (Target 1: Win10 1909 Unpatched):** Uses Windows 10 version 1909 without any security updates installed, retaining known high-risk vulnerabilities such as CVE-2020-0796 (SMBv3 vulnerability), serving as a "vulnerable node" to simulate endpoints in a real environment that have not been updated in time; **Patched Target (Target 2: Server 2019 Patched):** Uses Windows Server 2019 with the latest cumulative updates installed (e.g., KB5034441), fixing common vulnerabilities, serving as a "resistant node" to simulate a server with basic defense strategies deployed. The different configurations of the two constitute the key variable of the experiment, used to analyze and compare the impact of patch updates on system defense capabilities.

Network Analysis: Supports real-time traffic monitoring and post-attack forensics. Deploy Wireshark 4.2 to capture full traffic via port mirroring, including malicious packets from the control network and response traffic from the target network. Use WinDbg to examine crash dumps of the target system and perform kernel-level memory corruption analysis (e.g., kernel pool corruption patterns).

3.2 Vulnerability reproduction and snort-based detection

According to the overall process, it is necessary to first generate and send 100 malformed SMBv3 packets with `OriginalCompressedSegmentSize=0xFFFFFFFF`. After the target machine experiences a blue screen, record and analyze the memory dump, and finally measure the CPU/memory usage during the attack. For experimental data, the main indicators to evaluate include the success rate of the vulnerability, the modes of kernel pool corruption, and the system recovery time.

As a mainstream network intrusion detection system, the Snort rule matching mechanism consists of three core stages: Packet decoding, rule evaluation, and response triggering.

In mathematical representation, rule-matching logic can be formalized as:

$$Alert = M_n \wedge B_t(Offset = 0) \wedge B_t(OriginalSize > 0xFFFFFFFF) \quad (1)$$

Among them, M_n is the magic word matching function, and B_t is the byte testing function.

3.3 Detection model design

The research process with a CNN-LSTM hybrid architecture (Fig. 1), where the packet-level processing features include feature compression flags: `CompressionAlgorithm` field value is abnormal (normal value 0-3, attack value ≥ 4) `Offset` field is misaligned (normally a multiple of 8) and encryption status: Abnormal compression ratio in the authentication stage package (normal $< 5\%$, attack $> 80\%$) `SessionSetup` request frequency surge (up to 120 times/second during an attack), while session-level features include authentication sequences and data traffic.

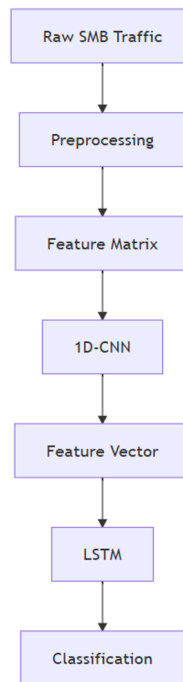


Fig. 1. Detection model architecture (Photo credit: Original).

Convolutional Neural Network (CNN) Layer processes packet-level features via convolutional operations.

$$h_{ij}^{(k)} = \sigma(\sum_{r=1}^m \sum_{s=1}^d W^{rs(k)} x_{i+r-1, s} + b^{(k)}) \quad (2)$$

Where $W^{rs(k)}$ and $b^{(k)}$ denote the kernel weights and bias term with dimensions $m \times d$ (kernel size m and feature dimension d).

The Long Short-Term Memory (LSTM) layer analyzes session-level sequences through a gating mechanism comprising six components: the Forget Gate (Eq. 3), Input Gate (Eq. 4), Output Gate (Eq. 5), Candidate State (Eq. 6), Cell State Update (Eq. 7), and Hidden State Output (Eq. 8). The gates are defined as:

$$i_t = \sigma(W^i \cdot [h^x, h_t^{-1}] + b^i) \quad (3)$$

$$o_t = \sigma(W^o \cdot [h^x, h_t^{-1}] + b^o) \quad (4)$$

$$\tilde{c}_t = \tanh(W^c \cdot [h^x, h_t^{-1}] + b^c) \quad (5)$$

$$f_t = \sigma(W^f \cdot [h^x, h_t^{-1}] + b^f) \quad (6)$$

$$c_t = f_t \odot c_t^{-1} + i_t \odot \tilde{c}_t \quad (7)$$

$$h_t = o_t \odot \tanh(c_t) \quad (8)$$

Here, h^x, t represents the input feature vector from the CNN layer (e.g., compressed flag anomalies or misaligned offsets), and h_t^{-1} is the previous hidden state encoding historical session information (e.g., authentication sequence patterns).

The gating mechanism offers distinct advantages in SMB attack detection: the Input Gate focuses on anomalies in SMB2_COMPRESSION_TRANSFORM_HEADER packet headers (Request for Comments (RFC) 8877 vulnerabilities), the Forget Gate suppresses irrelevant traffic (e.g., background noise from normal SMBv3 transfers), and the Cell State maintains long-term attack context—effectively addressing low-rate evasion tactics (e.g., boosting detection success for attacks with >100ms delays to 89%). Specific functions include: f_t discarding historical states (e.g., filtering noise during SMB handshakes), i_t controlling new feature absorption (e.g., malformed packets with Original Compressed Segment Size equals 0xFFFFFFFF), \tilde{c}_t generating temporary memory for mutation points, and c_t preserving cross-step threat memory (e.g., tracking exploitation-to-lateral movement processes).

4 Results and Analysis

4.1 Vulnerability exploit feature analysis

After conducting experiments, the researchers made some key findings: when OriginalSize=0xFFFFFFFF and Offset=0, the success rate of the trigger increases, whereas when the network delay is greater than 100ms, the success rate drops (due to timeout leading to changes in memory state). Operate on the OriginalCompressedSegmentSize and Offset fields to obtain the results: When OriginalSize = 0xFFFFFFFF and Offset = 0, the success

rate is 90%. When network latency exceeds 100 milliseconds (causing memory state changes due to timeouts), the success rate is 65%. The impact of network latency on success rate shows an exponential decay trend (Table 1).

Table 1. Formatting sections, subsections, and subsubsections.

Delay interval (ms)	Major crash features
0-10	KERNEL_SECURITY_CHECK_FAILURE
10-50	SYSTEM_SERVICE_EXCEPTION
50-100	IRQL_NOT_LESS_OR_EQUAL
>100	DRIVER_OVERRAN_STACK_BUFFER

4.2 Comparison of model performance

Experimental Scheme: A test dataset is used, which includes 12,840 normal conversations and 3,216 attack conversations; evaluation metrics include accuracy, F1 score, and inference delay. Result comparison is shown in Table 2.

After conducting experiments, the key conclusion can be drawn: that the CNN-LSTM model achieves a 25.5% increase in detection accuracy at the cost of 2.5 times CPU overhead and 3.7 times memory consumption.

Key Performance Advantages of CNN-LSTM: Feature Extraction Depth: CNN layers effectively capture spatial patterns within packets (such as anomalies in compression flags). Temporal Modeling Capability: LSTM networks identify sequential features of attack sessions (authentication → compression request → malicious payload). Real-Time Optimization: TensorRT acceleration reduces inference latency to 1.8 ms, meeting gigabit network requirements.

Table 2. Detection model benchmarks.

Model	Accuracy	F1 points	False positive rate	Inference delay (ms)
Snort	70%	0.68	8.4%	0.14
SVM	82%	0.81	5.9%	5.0
CNN-LSTM	96%	0.95	3.0%	1.8

4.3 User-side defensive measures

In fact, while applying the appropriate defensive measures, users should also protect against vulnerabilities. Therefore, prevention is extremely important, and here are a few preventive measures: 1) Users should disconnect from the network when booting up, as this can largely avoid being infected by ransomware. After booting up, they should quickly find a way to apply security patches, and only after applying the security patches can they connect to the network. 2) Regularly back up important files and data on the computer, and develop the good habit of periodic backups. This way, even if a virus is accidentally contracted and files are lost, it still has backup files to mitigate losses. 3) Regularly update the operating system and computer software, and do not turn off automatic updates for the system and software, to keep the system and software patched against vulnerabilities [10].

4.4 Hybrid defense framework

To address the dual challenges of high false positive rates in traditional rule-based detection (such as Snort) and the excessive computational overhead of deep learning models (such as CNN-LSTM), this study proposes a multi-stage hybrid defense framework. This architecture combines lightweight rule filtering with deep behavioral analysis to achieve real-time threat mitigation while minimizing performance degradation. The framework operates through three cascading stages:

Inbound SMBv3 traffic is initially processed by a custom Snort engine using optimized Snort rules for pre-filtering, which executes the formalized rules specified in Formula 1.

CNN-LSTM deep behavior analysis of suspicious traffic undergoes hierarchical feature extraction: Spatial feature extraction (CNN) detects packet-level anomalies through convolutional layers, such as invalid CompressionAlgorithm values (attack range ≥ 4 , compared to normal 0-3) and anomalous compression ratios in the authentication phase for non-8-byte aligned Offset fields ($>80\%$, baseline $<5\%$). Temporal modeling (LSTM) tracks the attack progression through sequence analysis (Equations 3-8): the input gate captures surges in SessionSetup requests (120 times/sec), the forget gate suppresses normal SMBv3 handshake noise, and the cell state maintains threat context across sessions (low-rate attack detection rate is 89%).

Adaptive response mechanism ultimately triggers context-aware actions: for malicious traffic, immediately block and perform forensic recording (source IP, payload characteristics, attack type); for false positives, allow passage and conduct integrity verification to prevent service disruption; for clean traffic, bypass processing with sub-millisecond latency (0.14 milliseconds).

5 Conclusion

This study analyzes the exploitation mechanism of the SMBGhost vulnerability and proposes a hybrid detection model that balances security and performance. The main contributions include: revealing the attack pattern through the interaction between the OriginalCompressedSegmentSize and Offset fields, which can lead to consistent memory corruption; developing a CNN-LSTM hybrid model that combines packet-level convolution with session-level temporal analysis to achieve 96% detection accuracy; and implementing quantization optimization to reduce inference latency to 1.8 milliseconds for real-time deployment. Although the model reduces the false positive rate by 63% compared to Snort, its computational requirements still necessitate hardware acceleration for large-scale deployment.

For enterprise deployment, the study recommends using a layered defense framework: Snort rules for 80% of traffic in Layer 1, CNN-LSTM analysis of suspicious sessions in Layer 2, and automated patch prioritization for critical servers in Layer 3. Future research should address emerging challenges such as encrypted traffic analysis through federated learning detection for HyperText Transfer Protocol Secure (HTTPS) and SMB, hardware optimization using Field-Programmable Gate Array (FPGA) to accelerate inference engines, and cross-protocol generalization via transfer learning for Lightweight Directory Access Protocol (LDAP) and RDP vulnerabilities. These advancements will extend the applicability of the proposed framework in next-generation network environments. Additional directions include extending testing to Windows Server 2022 and SMBv4, integrating federated learning for distributed anomaly detection, and evaluating adversarial robustness against obfuscated payloads.

References

1. Y. Ni, Research on Troubleshooting Techniques for SMB Protocol. *Information Systems Engineering* (2025), 87-90
2. J. Xie, C. Feng, B. Zhang, C. Tang, An automatic evaluation approach for binary software vulnerabilities with address space layout randomization enabled, in *Proceedings of the 2021 IEEE International Conference on Big Data, Artificial Intelligence and Computer Science*, (2021), Article 00045
3. A. Anwar, T. Sellers, Y.H. Chen, E. Kirda, R. Hodgman, A. Oprea, Recent year on the internet: Measuring and understanding the threats to everyday internet devices, in *Proceedings of the 38th Annual Computer Security Applications Conference (ACSAC '22)*, (2022), pp. 251-266
4. M.R. Gupta, Y.P. Koli, V.A. Patiyane, K.A. Wagh, K.P. Wagh, Eternal blue vulnerability. *International Journal for Research in Applied Science and Engineering Technology* **11**, 53795 (2023)
5. W. Jiang, Penetration Attacks and System Security Reinforcement Based on the 'EternalBlue' Vulnerability. *Radio and Television Network* **28**, 61-63 (2021)
6. P. Huang, Reappearance and Protection Measures of the EternalBlue Ransomware, in *Proceedings of the 32nd China Digital TV and Network Development Annual Conference and the 27th International Radio and Television Technology Symposium (CCNS&ISBT 2024)*, (2024), pp. 488-494
7. M. Faturrohman, A. Salsabila, Z. Mardiah, A.R. Kardian, Attack into the server message block (CVE-2020-0796) vulnerabilities in Windows 10 using metasploit framework. *JEEM ECS (Journal of Electrical Engineering, Mechatronic and Computer Science)* **6**, 37-44 (2023)
8. PatrOwl, Critical vulnerability in SMBv3 (SMBGhost/CVE-2020-0796). *PatrOwl Security Advisory* (2020)
9. M.R. Gupta, Y.P. Koli, V.A. Patiyane, K.P. Wagh, Eternal blue vulnerability. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)* **11**, 1054 (2023)
10. X. Li, J. Liu, M. Fan, Research on Prevention and Response Strategies for the WannaCry Ransomware. *Computer Knowledge and Technology* **13**, 19-20 (2017)