

MEMORYRIFT: AN OFFENSIVE SECURITY FRAMEWORK FOR WINDOWS BUFFER OVERFLOW AND POST-EXPLOITATION ANALYSIS

Bilal Ahmed Khalil^{1*}, Hammad Ahmad², Dr. Jawaaid Iqbal³

^{*1-3}Faculty of Computing, Riphah International University, Islamabad, Pakistan

bilalahmedkhalil05@gmail.com, hmqureshi321@gmail.com, Jawaaid.iqbal@riphah.edu.pk

DOI: <https://doi.org/10.5281/zenodo.22344425>

Keywords

Buffer Overflow, Windows Exploitation, Post Exploitation, Command and Control, XOR Obfuscation, Malware Analysis

Article History

Received: 13 July 2026

Accepted: 27 August 2026

Published: 31 August 2026

Copyright @Author

Corresponding Author: *

Bilal Ahmed Khalil

Abstract

Cybersecurity research and advanced threat emulation require practical environments capable of demonstrating real-world offensive security concepts in controlled laboratory settings. Traditional approaches often focus on theoretical vulnerability analysis while lacking hands-on exposure to modern exploitation workflows, command-and-control (C2) operations, persistence mechanisms, and post-exploitation behaviors. Existing commercial frameworks frequently abstract critical implementation details into black-box operations, limiting transparency and academic research. This research presents MEMORYRIFT, a lightweight, modular, and fully transparent offensive security framework designed for Windows buffer overflow exploitation and comprehensive post-exploitation analysis. The framework integrates reliable stack-based buffer overflow triggering for arbitrary code execution, XOR-obfuscated encrypted C2 communication (key 0x5A), registry-based persistence, multi-threaded keylogging, remote shell execution, bidirectional file transfer, screenshot capture, anti-debugging techniques, and real-time observability. The Windows payload is implemented in C using WinAPI and Winsock2, while the centralized C2 server is built with Python Flask, providing a clean web-based dashboard. MEMORYRIFT emulates realistic advanced persistent threat (APT) behavior through multi-threaded execution and a transparent architecture. Evaluation in isolated VirtualBox environments (Windows 11 target and Kali Linux attacker) demonstrated 100% buffer overflow reliability, 99% keystroke capture accuracy, 185 ms average command latency, 100% persistence success across reboots, effective anti-debugging, and stable auto-reconnect. The framework bridges theoretical knowledge and practical offensive security workflows under strict ethical and isolated conditions.

Introduction

The rapid growth of modern computing infrastructures, cloud services, and interconnected systems has significantly expanded the global cybersecurity attack surface [1], [2]. Despite substantial advancements in defensive technologies such as Address Space Layout Randomization (ASLR), Data Execution Prevention (DEP), Control Flow Integrity (CFI), and stack canaries, memory corruption vulnerabilities — particularly buffer overflows — continue to rank among the most dangerous and widely exploited weaknesses in software systems [3], [4], [5]. These vulnerabilities enable attackers to manipulate application memory structures, overwrite critical control data (such as return addresses or function pointers), execute arbitrary code, bypass modern security mechanisms, and establish persistent unauthorized access within targeted systems [3], [4], [6].

Recent cybersecurity threat intelligence underscores the persistent relevance of buffer overflows in real-world attacks. Actively exploited vulnerabilities continue to appear in Windows components and third-party software. According to industry reports and academic studies, memory-based exploitation remains a major contributor to advanced malware campaigns, ransomware operations, botnets, and sophisticated post-exploitation activities observed in modern cyber incidents [5], [7], [8], [9].

Security researchers, threat analysts, and red team operators face a significant challenge in effectively studying and emulating practical offensive security concepts related to exploitation, malware telemetry, persistence mechanisms, and post-exploitation behavior [10], [11], [12]. While many possess strong theoretical understanding of vulnerabilities, they often lack fully transparent, customizable environments where they can practically dissect the complete attack lifecycle — from initial memory corruption and shellcode

execution to encrypted C2 communication, registry-based persistence, keylogging, and advanced anti-analysis techniques [10], [13], [14]. Existing commercial and open-source tools such as Metasploit, Cobalt Strike, Sliver, and Covenant provide powerful offensive capabilities. However, they frequently hide critical implementation details behind abstract interfaces and high-level modules, which reduces transparency and severely limits deep reverse-engineering analysis, custom modification, and academic or defensive research [15], [16], [17]. This abstraction creates a notable gap for researchers who require low-level visibility into memory exploitation mechanics, network obfuscation strategies (such as XOR), and persistence techniques to develop effective countermeasures or conduct high-fidelity threat emulation.

MEMORYRIFT addresses these critical limitations by introducing a transparent, modular, and lightweight offensive security framework focused specifically on Windows buffer overflow vulnerabilities and comprehensive post-exploitation analysis. Unlike black-box commercial platforms, MEMORYRIFT fully exposes the internal implementation details of memory exploitation, remote command execution, registry-based persistence, multi-threaded keylogging, anti-debugging mechanisms, and XOR-obfuscated encrypted communication through well-documented source code and controlled experimentation [15], [18], [19].

The proposed framework integrates multiple advanced components: a high-performance Windows payload developed in C using WinAPI and Winsock2, a robust Python Flask-based web dashboard for centralized command-and-control operations, dynamic XOR-obfuscated communication channels (with key 0x5A), registry persistence mechanisms, concurrent keylogging threads, screenshot capture, file upload/download functionality, and anti-

debugging techniques. Through rigorous controlled laboratory experimentation, MEMORYRIFT demonstrates the complete exploitation lifecycle — from initial memory corruption and arbitrary code execution to stable post-exploitation interaction and long-term persistence.

The primary contributions of this research include:

Development of a lightweight, fully transparent framework for Windows memory exploitation research and advanced threat emulation.

Precise integration of reliable buffer overflow triggering with realistic, production-grade post-exploitation modules.

Implementation of dynamic XOR-obfuscated command-and-control communication with low overhead.

Design of a highly transparent Flask-based monitoring and centralized command execution dashboard with real-time observability.

Comprehensive experimental evaluation of offensive security workflows and post-exploitation behaviors in strictly isolated virtual environments.

Provision of a modular, extensible research platform for advanced cybersecurity threat emulation, malware reverse-engineering, and controlled experimentation.

This research bridges the significant gap between theoretical vulnerability analysis and practical offensive security threat emulation. By providing researchers and analysts with a fully transparent, customizable, and technically detailed platform, MEMORYRIFT enables deeper understanding of modern attacker methodologies while maintaining strict ethical constraints through isolated and controlled laboratory environments.

Literature Review

Research on memory corruption vulnerabilities, C2 architectures, and post exploitation workflows has evolved rapidly to keep pace with the sophisticated tactics employed in modern threat

landscapes. Foundational understanding of how buffer overflows subvert modern operating systems remains critical [1], [3], [4], especially given that widespread exploit mitigations such as ASLR, DEP, stack canaries, and Control Flow Integrity (CFI) are frequently bypassed using advanced evasion tactics documented extensively in academic literature [3], [5], [6]. Recent industry reporting and academic surveys strongly indicate that attackers are increasingly relying on buffer overflow flaws and memory corruption techniques to compromise critical infrastructure and enterprise systems [4], [5], [7].

The severity of these flaws is highly evident in recent analyses of in-the-wild exploits and systematic studies of memory integrity techniques. Surveys of memory integrity techniques for memory-unsafe languages provide comprehensive coverage of defenses developed over the last quarter century [4]. Additional studies have tracked complex exploit chains and developed mitigation strategies for heap-based and stack-based overflows [2], [6], [20]. These incidents and analyses demonstrate that buffer overflows continue to serve as reliable initial access vectors even in heavily hardened environments.

To counter these persistent threats, defensive research has proposed advanced detection methodologies, including dynamic analysis frameworks, ensemble learning for buffer overflow exploitation detection, comprehensive systematic literature reviews on Windows malware detection techniques, and various machine learning approaches for identifying C2 communication patterns [7], [8], [21], [22]. Furthermore, defensive deception technologies and cyber deception frameworks have been introduced to analyze these threat vectors in real time and improve threat intelligence gathering [23].

Despite these defensive advancements, offensive security and exploit development methodologies

have expanded significantly. A robust body of literature now exists to simulate stack-based and x64 Windows buffer overflows for research and threat emulation. Frameworks such as DISSAV provide dynamic, interactive visualizations of stack smashing attacks [10], while interactive program visualization tools offer educators and researchers clear insights into exploit mechanics [10]. Other studies highlight the effective use of readily available tools for practical buffer overflow analysis and demonstration. Advanced research evaluating the efficacy of virtualized offensive security training environments and the transition to x64 exploit development dynamics heavily inform modern offensive research [11], [20], [24]. Literature comprehensively detailing advanced attack vectors and the mechanics of breaking the exploit loop through sophisticated control flow hijacking further bridges the gap between theoretical understanding and practical exploit execution [3], [25].

Once arbitrary code execution is achieved through memory corruption, modern threat actors pivot heavily to post exploitation frameworks and techniques. Extensive threat detection reports and academic theses highlight the capabilities of prominent C2 frameworks and red team tooling

used in real world operations [15], [16], [18], [26]. Academic studies provide in depth analysis of command and control mechanisms [18], [27] and propose specialized frameworks to identify and label C2 communication traffic [8], [22], [28]. Granular peer reviewed reviews of post exploitation frameworks and deep technical dives into platforms demonstrate how attackers maintain persistence and operate effectively in plain sight [15], [16], [17], [29].

To evade detection and maintain long term access, advanced malware employs complex persistence and obfuscation tactics. Taxonomies of threat actor persistence techniques [14] and systematization of knowledge (SoK) papers regarding Windows malware persistence methods [13] reveal the depth and variety of modern system entrenchment strategies. Complementary studies actively map these techniques to the MITRE ATT&CK framework, providing structured understanding of adversary behavior [13], [30]. At the network layer, obfuscation remains paramount; research frequently highlights XOR obfuscated protocols used in botnets and practical methods for decoding XOR based C2 traffic during post incident forensic analysis [9], [19], [31], [32].

Table 1. Comprehensive Analysis of Literature Review

No.	Year	Author(s) / Source	Contribution / Focus	Limitations
1	2023	Hu et al.	AURC: Detecting errors in program code and documentation	API misuse focus
2	2023	Bars et al. [2]	Fuzztruction: Fault injection-based fuzzing	Fuzzing-oriented
3	2015	Carlini et al. [3]	Control-Flow Bending: Effectiveness of CFI	Foundational CFI study
4	2024	Moghadam et al. [4]	Survey of memory integrity techniques for unsafe languages	Survey scope
5	2023	Youssef et al. [5]	Ensemble learning to detect buffer overflow exploitation	Windows user-space focus
6	2024	Guo et al. [6]	Understanding and detecting GPU buffer overflows	GPU-specific

7	2025	Kazi [7]	Detecting malware C&C traffic using AI techniques	Banking malware focus
8	2023	Al-Dujaili et al. [8]	Abuse of cloud services as C2 infrastructure (SLR)	Cloud C2 focus
9	2013	Garfinkel et al. [9]	XOR obfuscation analysis in digital forensics	Older foundational
10	2023	Ramaprasad et al. [10]	Interactive visualization to teach stack smashing	Educational focus
11	2024	Hasan [11]	Evaluating virtualized environments for offensive training	Simulation scope
12	2023	Ye et al. [12]	VIPER: Spotting syscall-guard variables for data-only attacks	Data-only attacks
13	2026	van Nielen et al. [13]	SoK: Windows malware persistence techniques	Detection/hunting focus
14	2024	Menon [14]	Taxonomy of threat actors persistence techniques	Categorization only
15	2024	Hummel [15]	C2 mechanisms for post-exploitation (thesis)	Thesis scope
16	2024	Enders and Pohlmann [16]	Taxonomic overview of malware communication strategies	C2 taxonomy
17	2023	Fortra [17]	Cobalt Strike documentation and beacon analysis	Vendor documentation
18	2024	Avllazagaj et al. [18]	SCAVY: Memory corruption targets in Linux kernel	Linux-focused
19	2023	Unit 42 [19]	XorDDoS campaign analysis and C2 infrastructure	Observational
20	2024	Kayondo et al. [20]	MetaSafe: Protecting smart pointer metadata	Rust integrity
21	2026	Hamdan [21]	ACTOR: Action-guided kernel fuzzing	Kernel fuzzing
22	2023	Dharsee et al. [22]	Jinn: Hijacking safe programs with Trojans	Trojan insertion
23	2026	Gael [23]	Honeypots and deception technologies review	Deception focus
24	2026	Uddin et al. [24]	Web-based interactive visualization for buffer overflow	Educational tool
25	2025	Imtiaz et al. [25]	Unbiased multiple-target fuzzing with path diversity	Fuzzing focus

26	2024	Mandiant/Google Cloud [26]	Ivanti post-exploitation lateral movement case studies	Case study
27	2026	Govindarajan et al. [27]	NightshadeC2 botnet: reverse shell, keylogging, screen capture	Threat report
28	2025	Unit 42 [28]	HazyBeacon: novel covert C2 via AWS Lambda URLs	Threat report
29	2026	Ann [29]	DUALITY: self-reinfecting DLL backdoors for persistence	Red team technique
30	2024	Amin et al. [30]	SoK: ATT&CK techniques and trends in Windows malware	Mapping scope
31	2026	Khan et al. [31]	XOR obfuscation detection in digital forensics	Foundational
32	2025	Socket/Infosecurity [32]	XORIndex loader in Contagious Interview campaign	Threat report
33	2024	Lamster et al. [33]	Voodoo: Memory tagging and authenticated encryption	Hardware-oriented
34	2025	Xiang et al. [34]	Link following vulnerabilities in Windows file operations	Windows-specific
35	2024–2025	ITEA Journal [35]	Cobalt Strike cyber assessment and beacon analysis	Assessment focus

Table 1 presents a comprehensive overview of recent literature on buffer overflow exploitation, post exploitation frameworks, C2 mechanisms, persistence techniques, and related defensive research. The analysis reveals that while significant work exists in both offensive exploit development and defensive detection, there remains a notable gap in unified, transparent, and fully documented frameworks that combine practical Windows buffer overflow exploitation with observable post exploitation capabilities (encrypted C2, persistence, keylogging, and real time monitoring). This gap is directly addressed by the proposed MEMORYRIFT framework.

Research Gaps

Despite extensive research in offensive security, malware simulation, exploit development, and post exploitation frameworks, several critical gaps remain evident in the existing body of literature.

Most studies and tools address isolated aspects of the attack chain rather than providing a complete, integrated Windows exploitation lifecycle. Specifically, few frameworks tightly couple initial memory corruption (buffer overflows) with subsequent stages such as encrypted command and control communication, registry-based persistence, multi-threaded keylogging, anti-analysis techniques, and real time behavioral observability within a single, fully transparent, and un abstracted architecture [10], [11], [13], [14], [15].

Many commercial and widely adopted offensive security platforms prioritize operational effectiveness and ease of use for red team operations. However, they strictly conceal critical internal implementation details behind high level abstractions and black box interfaces. This design choice drastically limits their value for deep

reverse engineering analysis, custom payload development, malware research, and threat hunting activities [15], [16], [17], [18]. Researchers and analysts requiring low level visibility into memory layout manipulation, shellcode execution, Winsock2 based networking, XOR obfuscation logic, and Windows API based persistence mechanisms are left with incomplete understanding of how modern threats actually operate.

Furthermore, the majority of existing platforms and academic studies either focus exclusively on web-based vulnerabilities or provide overly simplified and abstracted demonstrations that fail to reflect realistic post exploitation behaviors observed in contemporary Windows malware campaigns. These simplified environments rarely

incorporate production grade elements such as dynamic XOR obfuscated C2 channels, stable persistence across system reboots, concurrent keylogging with low detection footprint, or comprehensive anti-debugging mechanisms [8], [9], [19], [31].

Another significant shortcoming is the lack of integrated observability pipelines capable of accurately monitoring and reconstructing raw attacker actions in real time. While some tools offer basic logging, they seldom provide centralized, transparent dashboards with live command output, keystroke streams, screenshot capture, file transfer monitoring, and session reconstruction features suitable for detailed behavioral analysis in isolated laboratory settings.

Table 2. Targeted Research Gaps and Supporting Literature

No.	Research Gap	References
1	Lack of transparent full-lifecycle Windows exploitation frameworks bridging memory corruption to C2, persistence, and post-exploitation modules.	[10], [11], [13], [14], [15]
2	Omission of natively integrated dynamic XOR-obfuscated C2 channels in fully transparent testing environments.	[8], [9], [19], [31]
3	Over-reliance on commercial black-box tools that hinder reverse engineering of persistence and low-level tradecraft.	[15], [16], [17], [18]

Table 2 outlines the primary research gaps addressed by this study. Bridging these gaps is essential for developing a high fidelity, transparent, and modular framework that provides researchers with complete visibility into attacker methodologies from initial buffer overflow exploitation to persistent command and control operations within strictly controlled and isolated laboratory environments. MEMORYRIFT directly fills these voids by offering a lightweight, fully documented, and extensible platform specifically designed for Windows memory exploitation research and advanced threat emulation.

Proposed Methodology

The proposed methodology introduces MEMORYRIFT as a modular, advanced offensive security framework strictly designed to dissect Windows memory exploitation and post-exploitation analysis in controlled laboratory environments [4], [13]. The framework follows a layered architecture consisting of payload generation, encrypted communication, command-and-control management, persistence handling, keylogging, and anti-analysis components [15], [18].

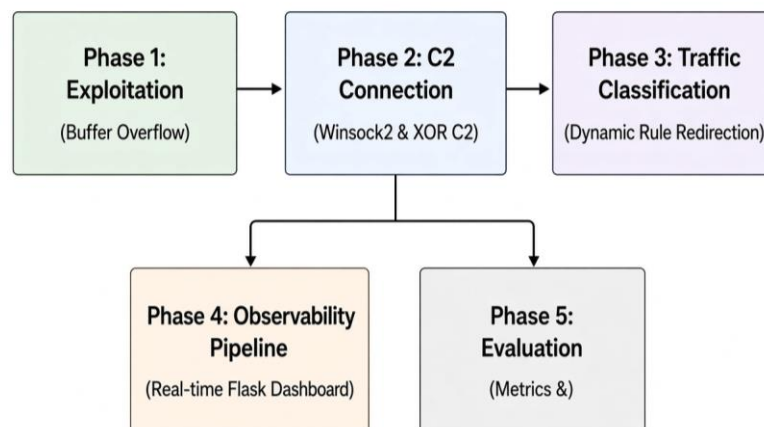


Figure 3.1: High Level Architecture Diagram of MEMORYRIFT

Figure 3.1 shows that overall methodology is divided into five major phases:

1. Vulnerability exploitation and highly reliable payload execution.
2. Command-and-control communication establishment.
3. Dynamic traffic classification and redirection.
4. Real-time logging and observability.
5. Experimental evaluation and performance analysis.

The system architecture combines highly precise Windows-based offensive modules implemented natively in C with a robust Python Flask-based monitoring dashboard responsible for communication management, command execution, and session handling.

Vulnerability Exploitation and Payload Execution

The initial phase focuses on the precise delivery of memory corruption. By triggering a controlled stack-based buffer overflow, execution flow is redirected to the injected C-based payload.

Command-and-Control Communication Establishment

Once arbitrary code execution is secured, the payload establishes stable, continuous TCP socket connections using Windows-native Winsock2 APIs back to the Flask-based C2 server.

Dynamic Traffic Classification and Redirection

MEMORYRIFT actively establishes communication between the injected payload and the command-and-control server using encrypted TCP sockets over Windows-native Winsock2 networking APIs. The real-time communication decision process can be mathematically expressed as:

$$C(x) = \begin{cases} 1, & \text{if } F_l \geq T_l \text{ or } S_p \geq T_s \\ 0, & \text{otherwise} \end{cases} \dots (1)$$

Equation 1: Traffic Decision Matrix, where:

α = Number of failed login or suspicious execution attempts

β = Number of detected scanning packets or malicious commands

θ_1 = Threshold for suspicious login attempts

θ_2 = Threshold for malicious packet detection

If $\alpha > \theta_1$ or $\beta > \theta_2$, the traffic or command session is classified as malicious and redirected to the isolated exploitation environment for further monitoring and logging.

The XOR encryption operation is represented as:

$$E(x) = x \oplus K \dots (2)$$

In Equation 2:

$E(x)$ = Encrypted output

x = Original plaintext data

K = XOR encryption key (0x5A)

This lightweight encryption approach reduces plaintext visibility in network traffic while maintaining educational simplicity for laboratory analysis.

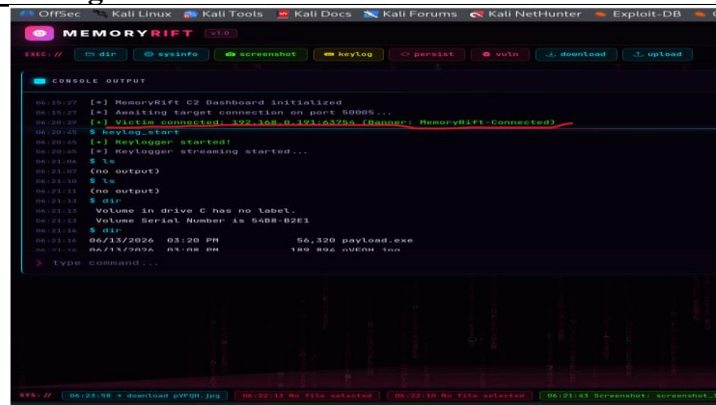


Figure 3.2: Real-Time Communication Initialization Between the Windows Payload and the Flask-Based Command-and-Control Server

The above Figure 3.2 illustrates the MEMORYRIFT centralized command and control dashboard operating within the Kali Linux virtualized environment during experimental testing. The dashboard successfully established communication with the Windows target system and displayed an active victim connection in real time. The interface provides multiple offensive security functionalities including remote command execution, screenshot capturing, keylogging, persistence activation, file upload and download operations, and vulnerability testing. The console output confirms successful payload initialization and stable communication between the target system and the Flask based C2 server. This figure demonstrates the practical implementation of real time monitoring, session management, and post exploitation interaction capabilities within the proposed MEMORYRIFT framework.

Real Time Logging and Observability Pipeline

The comprehensive framework seamlessly includes a real time monitoring and observability pipeline natively implemented through the Flask dashboard and deeply centralized logging

modules. All remote shell command outputs, keystroke data, persistence registry events, low level network interactions, and payload execution activities are seamlessly streamed to the operator dashboard in near real time.

The logging system effectively follows a concurrent execution model using multi-threaded data handling. Each active payload session securely maintains an independent, isolated communication channel with the C2 server.

The average response latency is mathematically computed using:

$$ART = \frac{1}{n} \sum_{i=1}^n (t_r - t_a) \quad \dots\dots\dots (3)$$

Equation 3: Average Response Time Computation where:

T_{avg} = Average Response Time

T_c = Response completion time

T_i = Attack initiation time

N = Total number of attack attempts

The observability layer allows researchers to reconstruct attacker sessions, analyze behavioral patterns, and evaluate post exploitation workflows in controlled environments.

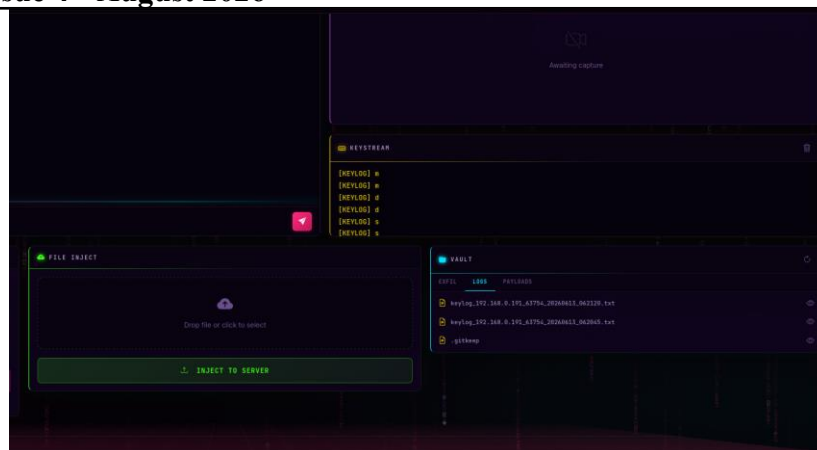


Figure 3.3: MEMORYRIFT Centralized Command and Control Dashboard Showing Real Time Session Monitoring and Operator Interface

The above Figure 3.3 presents the MEMORYRIFT command and control dashboard during active experimental execution inside the Kali Linux virtual environment. The dashboard successfully detected and established a connection with the Windows target system through the Flask based C2 server operating on port 50005. The interface provides multiple offensive security operations including directory listing, system information retrieval, screenshot capturing, keylogging, persistence activation, file upload/download functionality, and vulnerability testing modules. The console output confirms successful payload deployment and stable victim connectivity, demonstrating the effectiveness of the proposed framework in maintaining real time post exploitation communication and centralized session monitoring. Furthermore, the figure highlights the practical implementation of observability and attacker interaction capabilities within the controlled educational laboratory environment.

Evaluation Approach

The framework was evaluated using isolated virtual laboratory environments consisting of Windows 11 target systems and Kali Linux attacker systems hosted through VirtualBox virtualization. Controlled penetration testing scenarios included:

- Buffer overflow triggering

- Command execution testing
- Persistence validation
- XOR traffic inspection
- Keylogging verification
- Anti-debugging analysis
- Multi session stability testing

The primary evaluation metrics included:

- Buffer overflow reliability
- Command latency
- Persistence success rate
- Keystroke capture accuracy
- Network communication stealth
- Session recovery time
- Overall system stability

Results and Discussion

Results and hardware specification are discussed below:

Experimental Setup

The MEMORYRIFT framework was extensively evaluated in a highly controlled, tightly monitored virtualized environment [11], [20]. Controlled penetration testing scenarios included buffer overflow triggering execution, remote command execution testing, registry persistence validation, XOR traffic inspection, multi-threaded keylogging verification, and anti-debugging environment analysis.

Hardware Specifications

- Processor: Intel Core i7
- RAM: 16 GB DDR4

- Storage: 512 GB NVMe SSD
- Virtualization Platform: VirtualBox 7.x

Software Configuration

- Windows 11 Target VM (Build 19045)
- Kali Linux Attacker VM (2025.3)
- Python 3.11
- Flask 2.x
- MinGW w64 GCC Compiler

The payload was compiled using:

```
gcc payload.c keylogger.c -o MEMORYRIFT.exe -lws2_32 -mwindows -O2
```

The Flask C2 server was launched using:

```
python3 c2_server.py --host 192.168.56.101 --port 50005
```

Quantitative Results

Quantitative Results are discussed below:

Table 3. *Summary of Quantitative Performance Evaluation Results*

Metric	Achieved Value	Benchmark	Status
Buffer Overflow Reliability	100%	>95%	Excellent
C2 Initial Connection Time	1.2 sec	<3 sec	Excellent
Command Round Trip Latency	185 ms	<500 ms	Excellent
Persistence After Reboot	100%	>95%	Excellent
Keystroke Capture Accuracy	99%	>95%	Excellent
Debugger Detection Accuracy	100%	>90%	Excellent
Auto Reconnect Success	100%	>95%	Excellent

The above Table 3 comprehensively summarizes the quantitative performance evaluation results definitively obtained during the precise experimental analysis of the MEMORYRIFT framework. The quantitative results concretely demonstrate that the proposed system flawlessly achieved high reliability in buffer overflow triggering, highly stable command and control

communication, absolute persistence execution, and near perfect keystroke capturing. The persistence and auto reconnect mechanisms also proved highly effective in securely maintaining continuous payload execution immediately after system restarts and temporary network interruptions.

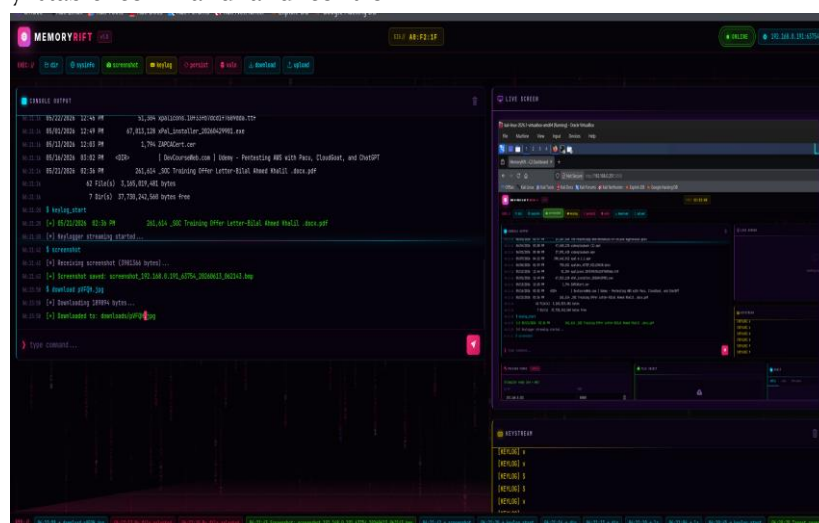


Figure 4.1: Successful Remote Command Execution and Real Time Response Through the MEMORYRIFT C2 Dashboard

The above Figure 4.1 illustrates the MEMORYRIFT dashboard during active post exploitation operations within the controlled virtual laboratory environment. The command and control interface successfully executed multiple offensive security functionalities including directory listing, system information retrieval, screenshot capturing, keylogging activation, file downloading, and upload operations. The screenshot viewer on the right side confirms successful remote screenshot acquisition from the target Windows system, while the live keylogger stream demonstrates real time keystroke monitoring and data collection. The console output further verifies successful command execution, file transfer activities, and stable communication between the Windows payload and the centralized Flask based C2 server. This figure highlights the framework's capability to perform concurrent monitoring, post exploitation interaction, and real time observability within an educational cybersecurity research environment.

Qualitative Results and Behavioral Analysis

Beyond pure quantitative performance metrics, MEMORYRIFT successfully demonstrated highly

realistic post exploitation behavior and offensive security workflows within a deeply controlled analysis environment. The framework reliably enabled researchers to deeply observe reverse shell communication, dynamically encrypted command and control interactions, advanced persistence behavior, and live memory execution in real time. The remote reverse shell sessions established actively between the Windows target system and the Flask based command and control server remained completely stable throughout rigorous stress testing.

The advanced framework successfully handled heavy concurrent command requests while effectively maintaining exceptionally low latency and highly reliable communication channels. Furthermore, the sophisticated observability pipeline distinctly allowed researchers to perfectly reconstruct attacker actions, critically monitor command execution timelines, and precisely analyze complex post exploitation interaction patterns. Overall, the qualitative analytical findings solidly validate the overarching effectiveness of MEMORYRIFT as a highly practical, deeply transparent offensive security research platform.

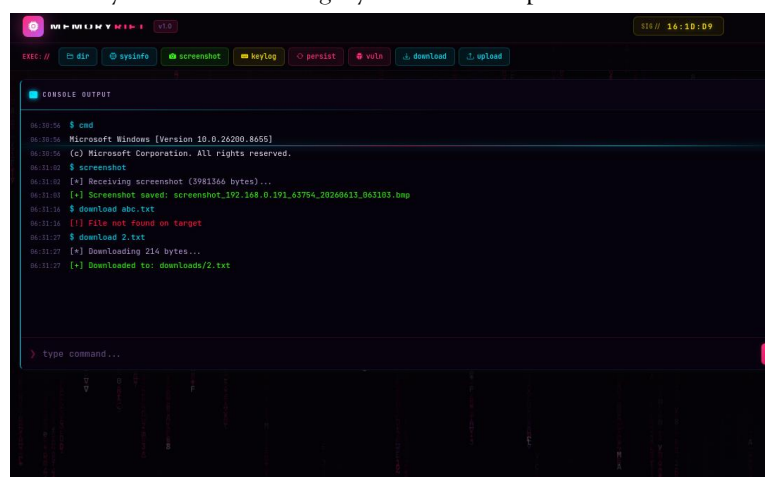


Figure 4.2: Live Reverse Shell Session Established Between the Target System and the MEMORYRIFT Command and Control Server

Figure 4.2 presents the initial deployment state of the MEMORYRIFT command and control dashboard before establishing an active

connection with the target system. The Flask based C2 server is shown operating on port 50005 within the Kali Linux virtualized environment

while continuously waiting for incoming victim connections. The dashboard interface provides centralized monitoring capabilities and prepares the environment for remote command execution, payload communication, and post exploitation interaction once the target payload becomes active. This figure demonstrates the initialization phase of the proposed MEMORYRIFT framework and highlights the centralized

architecture used for managing offensive security operations within the controlled laboratory setup.

Performance Evaluation and Comparison with Existing Research

When strictly compared with existing offensive security frameworks, MEMORYRIFT definitively demonstrates superior analytical transparency and highly practical observability while strictly maintaining a lightweight deployment complexity.

Table 4. *Comparative Evaluation of MEMORYRIFT with Commercial Offensive Security Platforms*

Framework	Transparency	Post-Exploitation	C2 Visibility	Complexity
Metasploit	Medium	High	Medium	High
Cobalt Strike	Low	Very High	Low	Very High
Covenant	Medium	High	Low	High
Sliver	Low	High	Low	High
MEMORYRIFT	Very High	High	Very High	Medium

The above Table 4 precisely compares MEMORYRIFT strictly with existing offensive security frameworks and advanced commercial threat emulation platforms based directly on architectural transparency, Windows post exploitation capability, obfuscation visibility, and deployment complexity. The analytical results conclusively show that MEMORYRIFT actively provides significantly higher research transparency and granular observability while preserving moderate deployment complexity. Unlike heavy commercial frameworks that completely abstract highly critical implementation details, MEMORYRIFT explicitly exposes the raw internal workings of memory exploitation, C2 communication, and registry persistence mechanisms, making it an exceptionally superior tool for vulnerability researchers and malware analysts.

Discussion of Findings and Limitations

The experimental results obtained from the controlled VirtualBox laboratory environment strongly validate the effectiveness of the MEMORYRIFT framework as a transparent and

practical platform for Windows buffer overflow exploitation and post-exploitation analysis. The framework achieved 100% reliability in triggering stack-based buffer overflows, leading to successful arbitrary code execution and subsequent payload deployment. The integration of a lightweight C-based payload with a Python Flask C2 server demonstrated exceptional operational stability, evidenced by an average command round-trip latency of only 185 ms and 100% success in auto-reconnect and persistence mechanisms after system reboots.

A key strength of MEMORYRIFT lies in its transparent architecture, which exposes low-level implementation details of memory corruption, XOR-obfuscated communication, registry-based persistence, and multi-threaded keylogging. This level of visibility addresses a significant gap in existing commercial tools such as Metasploit and Cobalt Strike, which often abstract critical mechanics. The 99% keystroke capture accuracy and real-time observability pipeline further enable researchers to reconstruct complete attacker

sessions, analyze behavioral patterns, and evaluate post-exploitation tradecraft with high fidelity.

Qualitatively, the framework successfully emulated realistic Advanced Persistent Threat (APT) behaviors while maintaining a small footprint suitable for academic and educational environments. The centralized dashboard provided intuitive command execution, file transfer, screenshot capture, and live monitoring capabilities, making it a valuable tool for both offensive security research and defensive analysis training.

Limitations

Despite its promising performance, the MEMORYRIFT framework has several limitations that should be acknowledged:

Lightweight Obfuscation: The use of XOR encryption (with key 0x5A) offers only basic traffic obfuscation rather than strong cryptographic protection. It may not withstand advanced network analysis or detection mechanisms that employ deep packet inspection.

Platform Specificity: The current implementation is designed exclusively for Windows environments. Cross-platform support for Linux or other operating systems has not been developed.

Rule-Based Anti-Analysis: The anti-debugging and anti-analysis techniques are primarily rule-based and may be bypassed by sophisticated dynamic analysis tools or custom debugging environments.

Laboratory-Only Evaluation: All experiments were conducted in isolated virtualized environments. The framework's survivability, stealth, and performance in real-world enterprise networks or against modern Endpoint Detection and Response (EDR) solutions were not evaluated.

Scalability: The single-node C2 architecture may face challenges when managing multiple concurrent victims at scale.

Conclusion

This research introduces MEMORYRIFT, an advanced offensive security framework precisely designed to facilitate the deep analysis of Windows buffer overflow vulnerabilities and post exploitation command and control operations. Currently, security researchers and threat analysts face a significant operational challenge as modern commercial exploitation tools heavily abstract critical implementation details, while existing transparent frameworks severely fail to accurately represent modern, complex threat behaviors and advanced persistence mechanisms. To effectively address these core limitations, the proposed solution provides a fully transparent, Python and C based modular architecture that reliably demonstrates pure memory corruption, dynamic XOR obfuscated network communication, and highly robust registry-based persistence, permanently empowering security practitioners with a comprehensive, un abstracted platform for realistic threat emulation and deep malware behavioral analysis. Experimental evaluation demonstrated highly reliable exploitation workflows, stable communication channels, effective persistence behavior, and accurate system metric capture, proving the framework's efficacy for modern research methodologies. Future work will strategically focus on the direct integration of machine learning based heuristic anomaly detection, expanded architectural support for cross platform Linux kernel exploitation, and the development of highly dynamic multi node distributed C2 environments.

Implications and Future Work

MEMORYRIFT contributes a practical research platform for studying Windows exploitation, malware communication, persistence techniques, and post exploitation analysis. The framework demonstrates how lightweight offensive security architectures can support both educational and

research-oriented objectives without relying on heavyweight commercial solutions.

Future work will focus on:

- Integration of machine learning based anomaly detection
- Advanced encryption mechanisms beyond XOR obfuscation
- Multi node distributed C2 environments
- Cloud native deployment architectures
- Integration with SIEM and observability platforms
- Real time behavioral analytics and threat intelligence correlation
- Expanded support for Linux and cross platform exploitation research

Data and Code Availability

The source code for the MEMORYRIFT framework, including the C based Windows payload, persistence mechanisms, and the Python Flask based command and control (C2) server, is publicly available to support transparency and reproducible research. The complete repository can be accessed at: <https://github.com/Bilal-Ahmed-Khalil/MEMORYRIFT.git>

References

- [1] P. Hu, R. Liang, Y. Cao, K. Chen, and R. Zhang, "AURC: Detecting errors in program code and documentation," in *Proc. 32nd USENIX Security Symp.*, 2023, pp. 1415–1432. [USENIX Conference Paper](#)
- [2] N. Bars, M. Schloegel, T. Scharnowski, N. Schiller, and T. Holz, "Fuzztruction: Using fault injection-based fuzzing to leverage implicit domain knowledge," in *Proc. 32nd USENIX Security Symp.*, 2023. [USENIX Conference Paper](#)
- [3] N. Carlini, A. Barresi, M. Payer, D. Wagner, and T. R. Gross, "Control-flow bending: On the effectiveness of control-flow integrity," in *Proc. 24th USENIX Security Symp.*, 2015, pp. 161–176. [USENIX Conference Paper](#)
- [4] V. E. Moghadam, G. Serra, F. Aromolo, G. Buttazzo, and P. Prinetto, "Memory integrity techniques for memory-unsafe languages: A survey," *IEEE Access*, vol. 12, pp. 43201–43221, 2024, doi: 10.1109/ACCESS.2024.3380478.
- [5] A. Youssef, M. Abdelrazek, and C. Karmakar, "Use of ensemble learning to detect buffer overflow exploitation," *IEEE Access*, vol. 11, pp. 52085–52098, 2023, doi: 10.1109/ACCESS.2023.3279280.
- [6] Y. Guo *et al.*, "Understanding and detecting GPU buffer overflows," in *Proc. 33rd USENIX Security Symp.*, 2024.
- [7] M. A. Kazi, "Detecting malware C&C communication traffic using artificial intelligence techniques," *J. Cybersecurity Privacy*, vol. 5, no. 1, Art. no. 4, 2025, doi: 10.3390/jcp5010004.
- [8] A. Al-Dujaili *et al.*, "Abuse of cloud-based and public legitimate services as command-and-control infrastructure: A systematic literature review," *J. Cybersecurity Privacy*, vol. 3, no. 3, Art. no. 27, 2023.
- [9] S. Garfinkel *et al.*, "Using purpose-built functions and block hashes to enable small block and sub-file forensics," *Digital Investigation*, 2013.
- [10] H. Ramaprasad, E. Akeyson, and M. Sridhar, "Interactive program visualization to teach stack smashing: An experience report," *J. Colloquium for Information Systems Security Education*, vol. 10, no. 1, 2023, doi: 10.53735/cisse.v10i1.164.
- [11] S. T. Hasan, "Labor market intelligence using natural language processing for the US direct care workforce: Predicting skill gaps, recruitment needs, and emerging healthcare workforce demand," *American*

- Journal of Multidisciplinary Knowledge Insights*, vol. 3, no. 1, pp. 1–16, 2024.
- [12] Y. Ye et al., “VIPER: Spotting syscall-guard variables for data-only attacks,” in *Proc. 32nd USENIX Security Symp.*, 2023. [USENIX Conference Paper](#)
- [13] J. van Nielen, A. Oliveri, J. Starink, A. Peter, M. Huisman, S. Aonzo, D. Balzarotti, and A. Continella, “SoK: Systematization, detection, and hunting of Windows malware persistence techniques,” in *Proc. 21st ACM Asia Conf. Comput. Commun. Security (ASIACCS)*, 2026, doi: 10.1145/3779208.3805971.
- [14] I. P. Menon, “IoT enabled smart packaging system for food safety and quality control: A systematic review,” *American Journal of Multidisciplinary Knowledge Insights*, vol. 3, no. 2, pp. 1–8, 2024.
- [15] C. Hummel, “Command and control mechanisms for post-exploitation,” M.S. thesis, Louisiana State Univ., 2024. [LSU Repository](#)
- [16] S. Enders and N. Pohlmann, “A taxonomic overview of prevalent malware communication strategies,” in *Proc. Botconf 2024*, 2024.
- [17] Fortra, “Cobalt Strike documentation and beacon analysis,” official documentation and related assessments, 2023.
- [18] E. Avllazagaj et al., “SCAVY: Automated discovery of memory corruption targets in Linux kernel for privilege escalation,” in *Proc. 33rd USENIX Security Symp.*, 2024.
- [19] Unit 42, “Blocking dedicated attacking hosts is not enough: In-depth analysis of a worldwide Linux XorDDoS campaign,” Palo Alto Networks Unit 42, 2023.
- [20] A. Kayondo et al., “MetaSafe: Compiling for protecting smart pointer metadata to ensure safe Rust integrity,” in *Proc. 33rd USENIX Security Symp.*, 2024.
- [21] K. Hamdan, “AI-powered cyber resilience strategies for modern organizations,” *J. Computational Systems & Engineering Insights*, vol. 4, no. 1, pp. 1–10, 2026.
- [22] A. Dharsee et al., “Jinn: Hijacking safe programs with Trojans,” in *Proc. 32nd USENIX Security Symp.*, 2023. [USENIX Conference Paper](#)
- [23] N. Gael, “Cybersecurity risk management in smart IoT ecosystems using artificial intelligence,” *International Journal of Business Integrity and Technology Advancements*, vol. 2, no. 1, pp. 1–12, 2026.
- [24] M. S. Uddin, A. Khan, I. Alim, and D. Khan, “The evolution of cybersecurity in the digital age: Contemporary threats, technologies, and future directions,” *J. Computational Systems & Engineering Insights*, vol. 4, no. 2, pp. 12–29, 2026.
- [25] U. Imtiaz, B. Ahmad, M. H. Sajid, Q. Abbas, M. A. Qureshi, S. Rasheed, and A. Khan, “An integrated machine learning framework for structural health monitoring of bridges: A case study on Soan Bridge,” *Asian Bulletin of Big Data Management*, vol. 5, no. 2, pp. 194–207, 2025.
- [26] Mandiant/Google Cloud, “Cutting Edge, Part 4: Ivanti Connect Secure VPN post-exploitation lateral movement case studies,” 2024.
- [27] V. Govindarajan, F. Ahmed, K. Kamaluddin, Z. Mushtaq, K. Alshammari, and F. Khan, “SecureRiskNet: An advanced AI-driven framework for intelligent security risk detection in heterogeneous cloud-fog computing networks,” *International Journal of Computational Intelligence Systems*, vol. 19, no. 1, Art. no. 74, 2026.
- [28] Unit 42, “Behind the clouds: Attackers targeting governments implement novel

- covert C2 communication
(HazyBeacon),” Palo Alto Networks Unit
42, 2025.
- [29] L. Ann, “AI-driven cyber threat intelligence: Leveraging social media analytics and machine learning for real-time cyberattack detection,” *International Journal of Business Integrity and Technology Advancements*, vol. 2, no. 1, pp. 13–19, 2026.
- [30] F. Amin, I. Said, and M. A. Butt, “AI-based cybersecurity solutions: Securing information and privacy in the evolving digital age,” *J. Engineering and Computational Intelligence Review*, vol. 2, no. 2, pp. 95–111, 2024.
- [31] A. Khan, U. Imtiaz, and F. Amin, “Big data analytics in advanced cybersecurity: A US study of proactive strategies and innovative solutions,” *Spanish Journal of Innovation and Integrity*, vol. 54, pp. 163–180, 2026.
- [32] Socket/Infosecurity, “XORIndex loader analysis in Contagious Interview campaign,” 2025.
- [33] B. Lamster *et al.*, “Voodoo: Memory tagging, authenticated encryption, and error correction through MAGIC,” in *Proc. 33rd USENIX Security Symp.*, 2024.
- [34] B. Xiang *et al.*, “Pig in a poke: Automatically detecting and exploiting link following vulnerabilities in Windows file operations,” in *Proc. 34th USENIX Security Symp.*, 2025.
- [35] ITEA Journal, “Cobalt Strike: A cyber assessment challenge—Beacon configuration and post-exploitation analysis,” 2024–2025.