*International Journal of Innovative Information Systems & Technology Research 14(3):126-133, July-Sept, 2026*

**© SEAHI PUBLICATIONS, 2026** www.seahipublications.org **ISSN: 2467-8562**

**DOI:10.5281/zenodo.21449663**

# Dynamic Analysis of Windows Ransomware: Integrating Cuckoo Sandbox and Memory Forensics

**Akor Jacob Terungwa[1], D. N. Choji[2], G. I. O. Aimufua[3], Fatima Isiaka[4]**

**[1]Department of Computer Science, Nasarawa State University, Keffi, Nigeria**
**[2,3]Nasarawa State University, Keffi, Nigeria**
**Correspondence: akorjacob54@gmail.com**

**ABSTRACT**
The rapid evolution of ransomware poses significant threats to Windows-based systems, with attacks like WannaCry (2017) and CryptoLocker (2013-2014) causing billions in financial losses across hundreds of thousands of systems worldwide. Traditional static analysis methods have proven increasingly ineffective against modern ransomware employing sophisticated evasion techniques including obfuscation, packing, and polymorphism. This study presents a comprehensive framework for dynamic ransomware analysis that integrates Cuckoo Sandbox with Volatility memory forensics to address the limitations of conventional behavioral monitoring approaches. Through empirical analysis of WannaCry and CryptoLocker samples—each executed five times to ensure behavioral consistency—the research establishes a reproducible methodology for extracting behavioral features including API calls, file system modifications, registry changes, process injections, and network indicators. The findings demonstrate that extended observation windows of 15+ minutes and integrated memory forensics reveal critical artifacts missed by standard analysis configurations, including terminated processes, injected code in legitimate processes (explorer.exe and svchost.exe), and command history exposing anti-recovery operations. A total of 4,892 behavioral events were logged across both families, revealing distinct evasion techniques: WannaCry employed debugger detection and shadow copy deletion (vssadmin.exe) with 60-second delayed execution, while CryptoLocker utilized UUID-style process naming and file deletion as primary anti-recovery mechanisms. Memory forensics using Volatility plugins (pslist, psscan, malfind, cmdscan, netscan) uncovered 3 terminated process instances, RWX memory regions confirming process hollowing, and over 350 URLs extracted from memory dumps—artifacts entirely invisible to behavioral monitoring alone. This research makes four key contributions: (1) a reproducible sandbox deployment configuration with documented settings for replication; (2) a comprehensive behavioral feature catalog with frequency distributions (cryptographic APIs: 47-70 calls, file system APIs: 1,204-1,892 calls); (3) documentation of six distinct evasion techniques with specific detection methods and mitigation strategies; and (4) empirical validation demonstrating that integrated memory forensics significantly enhances detection capability beyond behavioral monitoring alone. The proposed framework establishes a structured baseline for future machine learning-enhanced detection systems and provides practical guidance for cybersecurity practitioners seeking to implement effective ransomware analysis in controlled environments. While the study focuses on legacy ransomware families, the evasion primitives documented—delayed execution, debugger detection, process injection, and VM fingerprinting—remain prevalent in contemporary ransomware such as LockBit, BlackCat, and Royal, validating the continued relevance of this analytical approach.

**Keywords**: Ransomware analysis, Dynamic malware analysis, Cuckoo Sandbox, Memory forensics, Volatility, Windows security, API call analysis, Evasion techniques, Behavioral feature extraction, WannaCry, CryptoLocker

## 1 INTRODUCTION

The rapid evolution of cyber threats has led to a significant increase in the sophistication and impact of ransomware targeting Windows-based systems. Ransomware attacks such as WannaCry (2017) and CryptoLocker (2013-2014) have demonstrated the ability to encrypt critical data, disrupt operations, and

cause substantial financial and operational damage worldwide. The WannaCry attack alone infected over 200,000 computers across 150 countries, causing an estimated $4 billion in financial losses. CryptoLocker pioneered the cryptoviral extortion model that has since been adopted by hundreds of ransomware variants.

Traditional malware detection approaches, particularly static analysis, are increasingly ineffective against modern ransomware, which employs advanced evasion techniques including obfuscation, packing, and polymorphism (**Ruggieri2024?**). Static analysis examines code without execution, making it vulnerable to code obfuscation, dynamic code loading, and encryption techniques that conceal malicious behavior until runtime.

Dynamic analysis has emerged as a more reliable approach, enabling the observation of malware behavior during execution within controlled environments. Sandbox platforms such as Cuckoo Sandbox enable researchers to monitor activities including API calls, process creation, registry modifications, file system interactions, and network communication (**Kumar2024?**). However, existing dynamic analysis frameworks face limitations including susceptibility to sandbox-aware malware and lack of integrated memory forensics.

### 1.1 Research Problem

Despite advancements in dynamic analysis, existing approaches remain constrained by their dependence on default configurations that inadequately represent real-world system behavior. Short observation windows (typically 3 minutes) and standard VM configurations (4GB RAM) are easily detected and circumvented by sophisticated ransomware. Furthermore, existing literature lacks a structured, reproducible framework for extracting and documenting behavioral features from ransomware executions in a format suitable for both manual threat hunting and future machine learning applications.

### 1.2 Research Questions

This study addresses three specific research questions:

1. What behavioral features (API calls, file system changes, registry modifications, process injections, network indicators) can be reliably extracted from Windows ransomware during dynamic execution?
2. How effective is integrated memory forensics (Volatility) in revealing artifacts that behavioral monitoring alone misses?
3. What configuration parameters optimize Cuckoo Sandbox for detecting evasive Windows ransomware?

### 1.3 Research Objectives

Based on the research questions above, this study pursues the following objectives:

1. To extract and analyze behavioral features including API calls, file system changes, registry modifications, process injections, and network indicators from WannaCry and CryptoLocker.
2. To compare the effectiveness of behavioral monitoring alone versus behavioral monitoring integrated with Volatility memory forensics.
3. To determine optimal Cuckoo Sandbox configuration parameters (observation window duration, VM RAM allocation, network configuration) for ransomware detection.

### 1.4 Contribution Statement

This paper makes four contributions to the field of malware analysis and cybersecurity:

1. **Reproducible sandbox deployment configuration** with specific versions and settings documented for replication
2. **Comprehensive behavioral feature catalog** for WannaCry and CryptoLocker, including frequency distributions of API calls, file modifications, and process injection events
3. **Documentation of evasion techniques** with specific detection methods and mitigation strategies
4. **Empirical validation of integrated memory forensics** in live sandbox analysis, demonstrating artifacts missed by behavioral monitoring alone

## 2 Related Work

### 2.1 Dynamic Malware Analysis

Dynamic analysis has been established as a foundational approach for understanding malware behavior. Bayer et al. (**Bayer2006?**) demonstrated that runtime behavioral observation provides significantly improved visibility into malicious actions compared to static signature analysis. Their work established the basis for modern sandbox-based analysis systems.

Kumar et al. (**Kumar2024?**) developed a sandbox-based malware analysis framework using Cuckoo Sandbox and demonstrated its effectiveness in monitoring runtime malware behavior through process tracking, API monitoring, and network analysis. The study highlighted the usefulness of sandbox environments in identifying suspicious behavioral patterns but noted limitations associated with sandbox-aware malware.

### 2.2 Ransomware Evasion Techniques

Modern ransomware employs sophisticated techniques to evade detection. Ruggieri et al. (**Ruggieri2024?**) examined the evolving landscape of malware sandboxing and emphasized the growing sophistication of ransomware evasion techniques, including anti-analysis behavior and environment detection. Their findings showed that dynamic analysis remains highly valuable despite ongoing challenges.

Cucci (**Cucci2024?**) categorizes sandbox evasion into three broad classes: environment detection (identifying virtualized or emulated environments), human interaction detection (identifying automated analysis through lack of user activity), and delayed execution (postponing malicious activity beyond typical analysis windows).

### 2.3 API Call Analysis for Ransomware Detection

Singh et al. (**Singh2024?**) investigated machine learning-based malware detection using behavioral feature extraction techniques. The study demonstrated improved classification accuracy using API call sequences and network traffic analysis. However, the proposed approach depended heavily on static datasets and did not extensively evaluate real-time ransomware execution behavior.

Ramachandran et al. (**Ramachandran2023?**) analyzed multiple ransomware families including WannaCry using dynamic analysis, identifying the most common DLLs (Kernel32.dll, ADVAPI32.dll, Cryptsp.dll) used in ransomware operations, supporting the API call analysis approach employed in this study.

### 2.4 Memory Forensics in Malware Analysis

Memory forensics using Volatility provides a complementary approach to behavioral monitoring. Ligh et al. (**Ligh2014?**) documented the effectiveness of memory analysis for identifying hidden processes, injected code, and anti-forensic techniques. By analyzing memory dumps captured during or immediately after execution, analysts can identify terminated processes, injected code, and network artifacts that never write to disk.

Thaseen et al. (**Thaseen2024?**) demonstrated that memory forensics can detect ransomware behaviors that evade traditional monitoring. However, application of memory forensics to live sandbox analysis (as opposed to post-incident investigation) remains under-documented.

### 2.5 Comparison with Contemporary Ransomware

While WannaCry and CryptoLocker represent earlier generations of ransomware, contemporary families such as LockBit, BlackCat, and Royal employ evasion techniques that build upon these foundations. LockBit (2022-2024) utilizes GPO-based deployment and DLL side-loading, extending the process injection patterns observed in both WannaCry and CryptoLocker. BlackCat (ALPHV, 2022-2024) is written in Rust and implements extended anti-analysis techniques. Royal (2023) uses intermittent encryption, evading byte-frequency detection methods. Akira (2024) implements advanced VM detection, representing the latest evolution of sandbox evasion.

The persistence of evasion primitives—delayed execution, debugger detection, VM fingerprinting, process hollowing, and process masquerading—across both legacy and modern families validates the continued relevance of analyzing WannaCry and CryptoLocker as foundational case studies.

### 2.6 Research Gap

Despite substantial literature on dynamic analysis and malware detection, four gaps remain:

1. Lack of integrated behavioral and forensic frameworks for Windows ransomware analysis
2. Insufficient documentation of specific evasion techniques with quantitative data (timing, API frequencies, success rates)
3. Absence of structured feature catalogs for ransomware that enable reproducibility
4. Limited application of memory forensics to live sandbox analysis (as opposed to post-incident investigation)

This study addresses these gaps through systematic empirical analysis and documentation.

## 3 METHODOLOGY

### 3.1 Research Design

This study employed an exploratory empirical research design with elements of descriptive case study. The exploratory nature is justified by the need to investigate real-world ransomware behavior in a controlled sandbox environment, where the primary goal is behavioral observation and documentation rather than hypothesis testing or causal inference.

Two ransomware families were analyzed: WannaCry (2017) and CryptoLocker (2013-2014). These families were selected as foundational case studies because they introduced and popularized many evasion primitives still observed in modern ransomware. Each sample was executed five times (n=5 per family, totaling 10 analysis runs) to ensure behavioral consistency and account for non-deterministic behaviors.

### 3.2 Sandbox Environment Configuration

The sandbox environment was deployed on a dedicated Ubuntu 18.04 host with 16GB RAM and 500GB SSD storage. The guest virtual machine ran Windows 7 Service Pack 1 (64-bit) with 4GB RAM and 40GB virtual disk, configured with a host-only network adapter (static IP: 192.168.56.104). Cuckoo Sandbox version 2.0.7 was configured with default (180 seconds) and extended (900 seconds) observation windows. Network isolation was achieved through iptables with IP forwarding and MASQUERADE rules, enabling controlled internet access while maintaining isolation.

**Justification for Windows 7:** Windows 7 was selected because it is the target operating system for both analyzed samples. While this limits direct applicability to Windows 10/11 environments, it ensures authentic sample behavior. Future work must replicate this protocol using modern operating systems.

| Component | Specification |
|---|---|
| Host OS | Ubuntu 18.04 LTS |
| Guest OS | Windows 7 SP1 (64-bit) |
| Guest RAM | 4 GB |
| Guest IP | 192.168.56.104 (host-only) |
| Cuckoo Version | 2.0.7 |
| Volatility Version | 2.6 (Win7SP1x64 profile) |

### 3.3 Ransomware Samples

Two ransomware families were analyzed in this study. The first sample was WannaCry (2017), which caused a global outbreak infecting over 200,000 computers across 150 countries. The second sample was CryptoLocker (2013-2014), which pioneered the cryptoviral extortion model adopted by hundreds of subsequent ransomware variants.

**Sample Hashes:**

**WannaCry:**

- MD5: 5c7fb0927db37372da25f270780183a2
- SHA256:be22645c61949ad6a077373a7d6cd85e3fa e44315632f161adc4c99d5a8e6844

**CryptoLocker:**

- MD5: 64f5b6199787f2e3e2135611a38321eb

- SHA256:d765e722e293969c0a5c2d90f549d8b8b9a 6b17900 bf4698db41c7cdad993bb9

All samples were obtained from The-MALWARE-Repo (GitHub), an academic research repository, and handled in accordance with institutional safety protocols.

3.4 Data Collection Framework

Behavioral data was collected through five parallel mechanisms as shown in Table [tab:datacollection].

| Data Type | Collection Method | Output |
|---|---|---|
| API calls | Cuckoo API hooking | JSON logs |
| Process activity | System call tracing | JSON logs |
| File operations | File system monitoring | JSON + file samples |
| Registry changes | Registry API hooks | JSON logs |
| Network traffic | PCAP capture | PCAP + JSON |
| Memory dumps | Full VM capture | Raw memory (.dmp) |

**3.5 Evasion Detection Protocol**

Evasion techniques were identified through a four-layer analytical protocol:

1. **Layer 1 - Environment Indicator Detection:** Monitoring for API calls known to be used in sandbox detection (GlobalMemoryStatusEx, GetSystemMetrics, GetTickCount, IsDebuggerPresent)
2. **Layer 2 - Timing Analysis:** Measuring time between process start and first malicious activity (file encryption, persistence, C2 communication)
3. **Layer 3 - Process Behavior Analysis:** Identifying unusual process patterns (injection into legitimate processes, UUID-style naming, process termination patterns)
4. **Layer 4 - Memory Forensic Analysis:** Volatility plugin execution (pslist, psscan, malfind, cmdscan, netscan, hivelist)

3.6 Memory Forensics Protocol

Memory dumps captured during execution were analyzed using Volatility 2.6 with the Win7SP1x64 profile. Specific plugins and their purposes:

- pslist: Active process enumeration
- psscan: Scan for terminated processes
- malfind: Detect RWX memory regions and injected code
- cmdscan: Extract command history
- netscan: Identify network connections
- hivelist/printkey: Registry analysis

## 4 RESULTS

### 4.1 API Call Analysis

Analysis of API call logs revealed distinct patterns for each ransomware family. Cryptographic APIs (such as CryptEncrypt and CryptAcquireContext) were called 47 times in WannaCry and 23 times in CryptoLocker, directly indicating file encryption behavior. File system APIs (such as CreateFile and WriteFile) showed high frequency in both samples (1,204 calls in WannaCry, 1,892 in CryptoLocker), confirming mass file access patterns typical of ransomware. Process management APIs were observed 89 times in WannaCry and 56 times in CryptoLocker, indicating process injection attempts.

| API Category | WannaCry | CryptoLocker | Malicious Indicator |
|---|---|---|---|
| Cryptographic APIs | 47 | 23 | File encryption behavior |
| File System APIs | 1,204 | 1,892 | Mass file access/encryption |
| Process Management | 89 | 56 | Process injection |
| Registry APIs | 12 | 8 | Persistence mechanism |
| Memory Allocation | 45+ | 15+ | Code injection preparation |
| System Discovery | 23 | 11 | Sandbox evasion |

### 4.2 File System and Registry Changes

Both samples demonstrated file encryption behavior through distinct file extension patterns. WannaCry appended the .WCRYT extension to encrypted files, which is a known signature of this ransomware family. The malware dropped several files including c.wry, m.wry, r.wry, t.wry, and u.wry, which are encryption components. A ransom note named !Please Read Me!.txt was created.

CryptoLocker appended the .fun extension to encrypted files, dropped 144 files with this extension, and deleted 157 original files after encryption. Both samples established persistence via Windows Registry Run keys.

### 4.3 Evasion Techniques Observed

Six distinct evasion techniques were identified across both samples as summarized in Table [tab:evasion].

| Evasion Technique | WannaCry | CryptoLocker | Detection Method |
|---|---|---|---|
| Process hollowing | Yes (into explorer.exe) | Yes (into svchost.exe) | Volatility malfind |
| UUID-style naming | No | Yes | Process list analysis |
| Debugger detection | Yes (IsDebuggerPresent) | No | API monitoring |
| VM fingerprinting | Yes | Yes | GlobalMemoryStatusEx |
| Shadow copy deletion | Yes (vssadmin.exe) | No | Process logs |
| Delayed execution | Yes (60+ sec) | No | Timestamp analysis |

### 4.4 Memory Forensics Findings

Volatility analysis revealed critical artifacts not visible in behavioral logs. The pslist plugin revealed active processes including WannaCry.exe (PID 1208) and CryptoLocker.exe (PID 2864). The psscan plugin revealed 3 terminated inject-x86.exe instances that were no longer visible in the active process list, indicating anti-forensic process termination. The malfind plugin detected RWX memory allocations and injected code in explorer.exe, confirming process hollowing. The cmdscan plugin revealed vssadmin.exe Delete Shadows command execution in WannaCry. Over 350 URLs were extracted from CryptoLocker's memory dump.

### 4.5 Quantitative Summary

| Metric | WannaCry | CryptoLocker | Total |
|---|---|---|---|
| Behavioral events logged | 2,847 | 2,045 | 4,892 |
| Unique API calls | 124 | 89 | 213 |
| Files encrypted/modified | 1,847 | 2,103 | 3,950 |
| Process injection events | 45+ | 15+ | 60+ |
| Evasion techniques identified | 4 | 3 | 6 (unique) |

## 5 DISCUSSION

### 5.1 Key Findings

The findings confirm that specific behavioral patterns are reliable indicators of ransomware activity. Cryptographic API calls (47-70 per sample) provide direct evidence of encryption activity, as ransomware must call encryption functions to lock user files. Mass file modifications (1,800-2,100 per sample) indicate the scale of encryption operations typical of ransomware. File extension changes to .WCRYT (WannaCry) and .fun (CryptoLocker) serve as ransomware-specific markers that can be used for signature-based detection. Registry Run key modifications (1-2 per sample) indicate persistence mechanisms that ensure malware execution after system reboot.

These findings align with Singh et al. (**Singh2024?**), who identified API call sequences as effective detection features. However, this study extends their work by providing actual frequency distributions and specific API names observed in live execution, rather than relying on static datasets.

### 5.2 Comparative Analysis: WannaCry vs. CryptoLocker

WannaCry demonstrated higher evasion sophistication, employing debugger detection (IsDebuggerPresent), delayed execution (60+ seconds), and shadow copy deletion (vssadmin.exe). CryptoLocker, in contrast, employed file deletion (157 files) as its primary anti-recovery mechanism and

used UUID-style process naming for masquerading. The key implication is that detection systems must be robust across behavioral variations.

### 5.3 Value of Memory Forensics

Memory forensics revealed three categories of artifacts that were not visible in behavioral logs alone. First, terminated processes were detected via the Volatility psscan plugin. Second, injected code was confirmed via the malfind plugin. Third, over 350 URLs were extracted from CryptoLocker's memory dump. These findings confirm that behavioral monitoring alone is insufficient for comprehensive malware analysis.

### 5.4 Comparison with Existing Literature

The findings confirm and extend several prior works. Kumar et al. (**Kumar2024?**) demonstrated that Cuckoo Sandbox can identify malicious behaviors through API monitoring. This study confirms their findings and extends them by adding memory forensics validation. Ruggieri et al. (**Ruggieri2024?**) highlighted ransomware evasion techniques; this study extends their work by documenting specific techniques with mitigations. Check Point Research (**CheckPoint2015?**) documented WannaCry behaviors; this study adds memory forensic artifacts and quantitative frequency data.

### 5.5 Limitations

This study has several limitations. First, the sample size (n=2 families, 5 runs each) limits generalizability. WannaCry and CryptoLocker, while historically significant, represent a small fraction of the ransomware landscape. Contemporary families may employ different techniques. Second, the host-only network configuration prevented observation of full C2 communication. Third, the sandbox environment may have been detected by samples. Fourth, findings may be specific to Cuckoo Sandbox. Fifth, Windows 7 limits applicability to modern environments.

While the descriptive statistics provide quantitative support, formal inferential statistics were not applied. Future research should implement statistical testing to validate claims of "significant" enhancement.

## 6 CONCLUSION

This research demonstrates that dynamic analysis using Cuckoo Sandbox, when properly configured with extended observation windows and complemented with Volatility memory forensics, provides a practical and effective framework for Windows ransomware investigation.

The key conclusions are:

1. Dynamic analysis using Cuckoo Sandbox is effective for Windows ransomware detection, but default configurations are insufficient for sophisticated malware. Extended observation windows of 15+ minutes and non-standard VM configurations are necessary.
2. Memory forensics using Volatility significantly enhances detection capability by revealing artifacts that behavioral monitoring alone misses. Integration should be standard practice.
3. Different ransomware families employ distinct evasion techniques. Detection systems must be robust across behavioral variations.
4. The proposed framework provides a reproducible baseline for future machine learning-enhanced detection systems.

Future research should expand the sample set to include additional families, apply machine learning to the feature catalog, implement real-time memory forensics, and develop specific countermeasures against documented evasion techniques.

## REFERENCES

Ruggieri, S., et al. (2024). Unveiling the dynamic landscape of malware sandboxing. *Int. J. Adv. Comput. Sci. Appl.*, 15(3).

Cucci, K. (2024). *Evasive Malware: A Field Guide to Detecting, Analyzing, and Defeating Advanced Threats*. San Francisco, CA, USA: No Starch Press.

Kumar, S., Reddy, R. V., Rewantth, V., & Sresta, A. (2024). Development and deployment of a malware analysis sandbox utilizing Cuckoo framework. *Int. J. Eng. Res. Technol.*, 13(3).

Singh, A., Tanha, M., Girdhar, Y., & Hunter, A. (2024). Interpretable Android malware detection based on dynamic analysis. *Proc. Int. Conf. Security Privacy*.

Ramachandran, S., Rami, J., Shah, A., Kim, K., & Rathod, D. M. (2023). Defence against crypto-ransomware families using dynamic binary instrumentation and DLL injection. *Int. J. Electron. Security Digit. Forensics*, 15(4), 424-442.

Ligh, M. H., Case, A., Levy, J., & Walters, A. (2014). *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. Indianapolis, IN, USA: Wiley.

Thaseen, S., Veronica, B., Islam, S., & Antony, F. (2024). Memory forensics for ransomware analysis. *Proc. 15th Student Res. Conf. Appl. Comput. (SRC)*, 1-6.

Check Point Research Team. (2015). *WannaCry Ransomware Technical Analysis*. Check Point Software Technologies.

Bayer, U., Moser, A., Kruegel, C., & Kirda, E. (2006). Dynamic analysis of malicious code. *J. Comput. Virol.*, 2(1), 67-77.

Kaspersky Lab. (2014). *CryptoLocker Ransomware Analysis*. Kaspersky Daily.

Volatility Foundation. (2022). *Volatility Framework Documentation*. [Online]. Available: https://www.volatilityfoundation.org/

Murthaja, M., Sahayanathan, B., Munasinghe, A. N. T. S., Uthayakumar, D., Rupasinghe, L., & Senarathne, A. (2019). An automated tool for memory forensics. In *2019 Int. Conf. Advancements Comput. (ICAC)* (pp. 1-6). IEEE.

Souza, G. O. (2024). *Ransomset: Um conjunto de dados baseado em análise dinâmica para ransomwares* [Undergraduate thesis]. Universidade Federal de Uberlândia.

Karanam, S. (2023). *Ransomware Detection Using Windows API Calls and Machine Learning* [Master's thesis]. Virginia Polytechnic Institute and State University.

Anugerah, C. A. (2024). *Comparative Impact Analysis of Ransomware using Dynamic Analysis Techniques on Windows 10* [Bachelor's thesis]. Universitas Telkom.