Mr. Harunmiya Sirajmiya Malek, 2026, 14:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Towards Intelligent Prediction of Critical Process Died Errors in Windows Systems: A Comprehensive Review of AI-Driven Detection and Prevention Techniques

Mr. Harunmiya Sirajmiya Malek
Assistant Professor, M.Sc. IT, Sardar Patel Institute of Applied Science, Gujarat, India,
ermalek148@gmail.com

Abstract- One of the most difficult problems affecting the dependability and security of contemporary Windows operating systems is still kernel-level failures. The Critical Process Died error is the most important of these errors since it causes a Blue Screen of Death (BSOD) when a crucial operating system process fails, abruptly ending system execution. In addition to disrupting regular computer operations, these failures also affect cloud platforms, enterprise services, AI workloads, and digital security infrastructures. Recent developments in AI-driven system monitoring have shown a great deal of promise for spotting unusual system behavior before disastrous catastrophes take place. Recent studies on log-based anomaly detection, graph neural networks, transformer-based language models, contrastive learning techniques, AIOps frameworks, and eBPF-enabled observability for predictive system monitoring (2022–2024) are all critically examined in this study. The study also evaluates current methods according to their capacity for detection, computational effectiveness, scalability, and suitability for Windows-based settings. In order to determine future paths for intelligent operating system reliability, current research trends, constraints, and unresolved issues are examined. Lastly, a hybrid AI-driven framework that incorporates anomaly detection, kernel telemetry, and log analytics is suggested to strengthen the resilience of next-generation computing systems and improve early failure prediction.

Keywords: Windows Operating System, Critical Process Died, Blue Screen of Death (BSOD), Artificial Intelligence, Log Anomaly Detection, Graph Neural Networks, AIOps, eBPF, Operating System Reliability, Digital Resilience, System Monitoring.

I. INTRODUCTION

In cloud platforms, educational institutions, artificial intelligence (AI) infrastructures, and enterprise settings, Windows 10 and Windows 11 have emerged as the most popular operating systems. Operating system reliability is becoming a crucial prerequisite for guaranteeing consistent service availability, safe computing, and reliable AI execution due to their extensive deployment. Even with ongoing security and stability enhancements, Windows systems are still vulnerable to kernel-level malfunctions that can suddenly stop vital services.

The Critical operation Died issue, which happens when a crucial operating system operation abruptly ends, is one of the most serious kernel errors. Windows instantly stops execution and displays a Blue Screen of Death (BSOD) in order to preserve system integrity and stop additional corruption. Despite protecting the operating system from further harm, this method frequently causes operational downtime, data loss, and service outages (Cotroneo et al., 2014; Mohan et al., 2018). Operating system failures have become more expensive due to the quick uptake of cloud-native infrastructures, edge computing, and AI-driven applications. Unexpected kernel crashes can impair the availability of AI-enabled applications, disrupt model training, stop inference pipelines, and corrupt intermediate checkpoints. Therefore, proactive systems that can detect anomalous system behavior prior to catastrophic breakdowns are necessary in current computing settings.

Predictive failure analysis has replaced reactive troubleshooting in system monitoring thanks to recent developments in artificial intelligence. Graph

neural networks, transformer-based language models, contrastive learning, and AIOps frameworks are examples of AI-based methods that have shown promise in identifying unusual patterns in telemetry and system logs prior to system crashes (Xie et al., 2022; Tian et al., 2023; Han et al., 2023; Remil et al., 2024). These methods offer fresh chances to boost digital resilience and operating system dependability.

Recent advancements in AI-assisted methods for detecting and averting the Critical Process Died mistake are critically examined in this paper. The study identifies research gaps, evaluates existing literature, contrasts cutting-edge detection methods, and suggests a hybrid architecture for enhancing the resilience and dependability of upcoming Windows-based computer systems.

II. TECHNICAL ANATOMY

The Crash Mechanism and Its Root Causes

When Windows detects an unexpected termination of a process that is thought to be necessary for operating system functionality, it generates the Critical Process Died error, a kernel-level failure. When these processes fail, the operating system is forced to stop running in order to stop further damage because they are essential for maintaining services like memory management, process scheduling, and system security (Mohan et al., 2018).

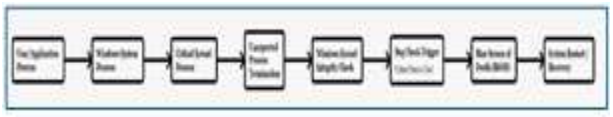

Figure 1. Workflow illustrating the sequence of events leading to the Critical Process Died error in Windows systems.

Figure 1 illustrates the typical sequence of events that results in a Critical Process Died error. Once Windows detects the unexpected termination of a critical kernel process, it initiates a bug check to protect system integrity, ultimately displaying a Blue Screen of Death (BSOD) and restarting the system to prevent further corruption.

The primary causes reported in the literature include:

- corruption of important system files brought on by storage mistakes or unsuccessful updates.
- Device drivers that run-in kernel mode is defective or incompatible.
- Memory corruption brought on by software or hardware flaws
- During kernel execution, race situations and synchronization errors occur.
- Protected system processes are compromised by malware or privilege escalation attacks.
- Long-term decline in operating system stability, resource depletion, and progressive software aging (Cotroneo et al., 2014).

Because Windows places a high priority on system integrity, these errors induce an instant BSOD instead of permitting possibly tainted execution to proceed.

Relevance for AI and Edge Systems

In AI-enabled computer systems where continuous processing and high system availability are crucial, kernel-level faults have grown in importance. Machine learning training often needs to be carried out continuously for several hours or even days. As a result, a single operating system crash can disrupt inference services, damage training checkpoints, invalidate computational progress, and drastically cut down on resource usage.

Because operating system failures can spread through distributed services and impact numerous applications at once, cloud platforms, edge devices, and enterprise AI infrastructures are especially vulnerable. These difficulties have spurred academics to look into AI-driven anomaly detection techniques that can spot unusual operating system activity before serious malfunctions happen (Pang et al., 2021).

III. MODERN AI-POWERED PREDICTION METHODS

Intelligent anomaly detection systems that can understand intricate behavioral patterns from vast amounts of operational data have replaced conventional rule-based monitoring in recent study.

Log-Based Anomaly Detection

System logs provide detailed records of operating system activities and are among the most informative sources for identifying abnormal behavior.

Xie et al. (2022) proposed LogGD, which uses Graph Neural Networks (GNNs) to describe links between log events. LogGD records structural relationships between events, which makes it possible to identify aberrant execution patterns more precisely than traditional sequence-based approaches.

Tian et al. (2023) created CLDTLog, which uses dual-objective optimization and contrastive learning to enhance anomaly identification with sparse labeled data. Robustness against changes in log formats is improved by its adaptive representation learning.

LogGPT, created by Han et al. (2023), interprets system logs as natural language using transformer-based language models. Semantic anomalies that are challenging to detect with traditional statistical methods can be found thanks to this contextual understanding.

Telemetry-Based Monitoring

Telemetry data offers ongoing insight into operating system activity through measurements such as CPU utilization, memory consumption, disk I/O, and process scheduling, while system logs offer useful historical information.

In order to find minute aberrations that precede kernel breakdowns, modern observability systems are increasingly combining telemetry with machine learning approaches. eBPF-based monitoring is especially useful for real-time anomaly identification since it allows lightweight kernel instrumentation without requiring intrusive changes to the operating system (Gregg, 2020).

Hybrid AI Frameworks

According to recent research, merging operational data from several sources greatly enhances anomaly detection ability (Remil et al., 2024).

Hybrid structures incorporate

- Event logs
- Kernel telemetry
- Resource utilization
- Process behavior
- Contextual metadata

This multi-modal technique improves early detection of anomalous operating system behavior while lowering false positives.

Table 1 : Comparison of Recent AI-Based Log Anomaly Detection Methods

Study	Method	Dataset	Main Contribution	Limitation
Xie et al. (2022)	LogGD	HDFS, BGL	Graph-based anomaly detection	High graph construction cost
Tian et al. (2023)	CLDTLog	HDFS	Contrastive learning	Requires tuning
Han et al. (2023)	LogGPT	Public logs	Context-aware reasoning	Large computational cost
Du et al. (2017)	DeepLog	HDFS	LSTM sequence learning	Limited semantic understanding
Meng et al. (2019)	LogAnomaly	HDFS	Sequential anomaly detection	Sensitive to unseen patterns

Recent AI-based log anomaly detection methods documented in the literature are contrasted in Table 1. While more modern techniques like LogGD, CLDTLog, and LogGPT use graph learning, contrastive learning, and transformer-based language models to better capture complex relationships inside system logs, older techniques like DeepLog mainly concentrate on sequential event modeling. These developments show a distinct trend toward more sophisticated and context-aware anomaly detection, strengthening the basis for anticipating kernel-level errors like the Critical Process Died error.

IV. COMPARATIVE ANALYSIS OF EXISTING METHODS

Table 2 illustrates that the most complete understanding of operating system activity is typically offered by hybrid AI frameworks that combine log analysis and telemetry monitoring. While transformer-based and graph-based models increase the accuracy of anomaly detection, combining several information sources allows for the early detection of kernel problems and strengthens system resilience.

Table 2: Comparative analysis of AI-based approaches for kernel-level anomaly detection.

Method	Main Technique	Advantages	Limitations
LogGD	Graph Neural Networks	Records the structural connections between log events.	Graph creation that is computationally costly
CLDTLog	Contrastive Learning	Adaptive learning with limited labeled data	requires careful adjustment of the parameters.
LogGPT	Transformer-based Language Model	Semantic comprehension that is context-aware	High computational demands
Telemetry + eBPF	Runtime kernel monitoring	Continuous real-time observability	Requires baseline profiling
Hybrid AI Framework	Multi-modal fusion	Improved detection accuracy and reduced false alarms	Increased implementation complexity

V. FUTURE RESEARCH & SYSTEM GAPS

AI-assisted anomaly detection has made great strides, however there are still a number of unresolved research issues:

- Supervised learning model development is limited by the lack of publicly accessible Windows-specific datasets containing Critical Process Died events.
- Rather than Windows kernel contexts, existing benchmark datasets like HDFS and BGL mostly depict distributed systems.
- The inability of current AI models to detect intricate failure processes stems from their infrequent integration of system logs with kernel telemetry.
- Administrator confidence in automated forecasts is limited by the lack of exploration of explainable AI approaches.
- Because operating system updates and changing workloads create concept drift, which gradually deteriorates model performance, constant adaptation is necessary.
- To enhance predictive maintenance in next-generation computing environments, future research should look into federated learning, retrieval-augmented AI, synthetic crash trace generation, and self-healing operating systems.

VI. CONCLUSION

One of the most serious kernel-level errors affecting Windows-based computer environments is still the Critical Process Died error. Maintaining operating system dependability has become crucial for guaranteeing cybersecurity, business continuity, and system resilience as enterprises rely more and more on artificial intelligence, cloud computing, and digital services. This paper looked at current developments in AI-driven methods, such as graph neural networks, contrastive learning, transformer-based log analysis, telemetry monitoring, and AIOps frameworks, for detecting and averting kernel-level problems. Compared to single-source monitoring methods, the comparative analysis shows that hybrid approaches that combine system logs with realtime telemetry offer higher detection capabilities.

Despite these advancements, some obstacles remain, including a dearth of Windows-specific benchmark datasets, low explainability of AI models, and the requirement for adaptable systems that can handle changing workloads and operating system changes. Addressing these difficulties would necessitate greater collaboration between operating system engineering, artificial intelligence, and cybersecurity research.

All things considered, this study demonstrates the enormous potential of AI-enabled anomaly detection to shift system failure management from reactive troubleshooting to proactive prediction. Future developments in explainable AI, federated learning, and intelligent observability are expected to play a key role in building more reliable, secure, and robust Windows computing systems.

REFERENCES

1. Xie, Y., Zhang, H., & Babar, M. A. (2022). LogGD: Detecting anomalies from system logs with graph neural networks. In 2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS) (pp. 299–310). IEEE. https://doi.org/10.1109/QRS57517.2022.00039
2. Tian, G., Luktarhan, N., Wu, H., & Shi, Z. (2023). CLDTLog: System log anomaly detection method based on contrastive learning and dual objective tasks. Sensors, 23(11), 5042. https://doi.org/10.3390/s23115042
3. Han, X., Yuan, S., & Trabelsi, M. (2023). LogGPT: Log anomaly detection via GPT. arXiv. https://doi.org/10.48550/arXiv.2309.14482
4. Remil, Y., Bendimerad, A., Mathonat, R., & Kaytoue, M. (2024). AIOps solutions for incident management: Technical guidelines and a comprehensive literature review. arXiv. https://doi.org/10.48550/arXiv.2404.01363
5. Cotroneo, D., Natella, R., Pietrantuono, R., & Russo, S. (2014). A survey of software aging and rejuvenation studies. ACM Journal on Emerging Technologies in Computing Systems, 10(1), Article 10. https://doi.org/10.1145/2513245
6. Mohan, J., Martinez, A., Ponnapalli, S., Raju, P., & Chidambaram, V. (2018). Finding crash-consistency bugs with bounded black-box crash testing. In Proceedings of the 2018 USENIX Annual Technical Conference. https://www.usenix.org/conference/atc18/presentation/mohan
7. Pang, G., Shen, C., Cao, L., & Van Den Hengel, A. (2021). Deep learning for anomaly detection: A review. ACM Computing Surveys, 54(2), Article 38. https://doi.org/10.1145/3439950
8. Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS) (pp. 1285–1298).
9. Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., Shen, Y., Zhang, X., & Tao, J. (2019). LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI) (pp. 4739–4745).
10. He, S., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE International Conference on Web Services (ICWS) (pp. 33–40).
11. Wei, X., Wang, J., Sun, C., Towey, D., Zhang, S., Zuo, W., Yu, Y., Ruan, R., & Song, G. (2024). Log-based anomaly detection for distributed systems: State of the art, industry experience, and open issues. Journal of Software: Evolution and Process, 36(8), e2650. https://doi.org/10.1002/smr.2650
12. Gregg, B. (2020). BPF Performance Tools. Addison-Wesley Professional.
13. Avizienis, A., Laprie, J.-C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing, 1(1), 11–33. https://doi.org/10.1109/TDSC.2004.2
14. He, S., He, P., & Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 207–218).
15. Le, V.-H., & Zhang, H. (2021). Log-based anomaly detection without log parsing. In Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 492–504).