ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 13, Issue 4, pp: (23-30), Month: October - December 2025, Available at: www.researchpublish.com

IMPLEMENTATION OF A SECURITY DIAGNOSTIC TOOL ON COMPUTER SYSTEMS

[1]Joda S.C, [2]Olokode S.M, [3]Olubo H.O, [4]Bamidele A.D, [5]Jinadu R.A, [6]Felemu O.D.

DOI: https://doi.org/10.5281/zenodo.17520734

Published Date: 04-November-2025

Abstract: **The implementation of a security diagnostic tool on computer systems involves preparing the system, choosing the right tool, running the scan and analyzing the results. The major challenge involves reduced performance, downtime or even data loss which leads to reduced performance of a system. The methodology adopted included setting up the required software and hardware environment, installing and configuring the diagnostic tool, and testing its performance under different load conditions. Real-time data was collected, analyzed, and presented in graphical and textual formats, while alerts were triggered whenever thresholds were exceeded. This project highlights the importance of implementing security diagnostic tools as a proactive approach to system management. It recommends that future work should enhance such tools with advanced visualization features, remote alerting options, and broader applicability across enterprise environments.**

Keywords: **security diagnostic, computer systems, hardware environment, security diagnostic tools, enterprise environments.**

1. INTRODUCTION

In today's digital era, computer systems are central to personal, academic, and organizational tasks. However, these systems are frequently vulnerable to both internal and external risks, including hardware malfunctions, software errors, and security threats. If left unchecked, these issues can lead to reduced system performance, data loss, and severe productivity setbacks (*Zhang et al 2022*).

To minimize these risks, the implementation of automated diagnostic tools has become increasingly necessary. A security diagnostic tool evaluates key system metrics in real time, detects abnormal behaviors, and provides early warnings, enabling proactive measures to maintain optimal performance and security (*Bishop Matt* 2019).

Furthermore, studies have shown that even minor technical issues can cause significant disruptions when multiplied across large organizations. For instance, a memory overload in one process can slow down hundreds of systems, resulting in operational delays and financial losses. In addition, the rise of sophisticated threats such as ransomware, spyware, and denial-of-service (DoS) attacks highlights the vulnerability of unmonitored systems. According to *Almashaqbeh et al.(2021)*, more than 60% of cyber incidents could have been mitigated with early detection through monitoring tools. This underscores the importance of implementing user-friendly diagnostic tools for maintaining both system health and security.

2. LITERATURE REVIEW

Author(s) / Year	Topic / Title	Objectives	Methodology	Contribution to Knowledge	Limitations
Zou, Y. (2025)	Security Diagnostic Tools and Their Implementation in Modern Computer Systems	To examine recent approaches to implementing security diagnostic tools that enhance system protection and reliability.	Reviewed multiple existing diagnostic frameworks and analyzed their performance in detecting system anomalies.	Provided insights into integrating security monitoring and performance optimization in diagnostic tools.	Limited practical validation; findings were mainly based on simulated test environments.

Kumar, R., & Sharma, S. (2020)	A Security Diagnostic Tool for Computer Systems	To design and implement a tool that automatically analyzes and detects performance and security flaws in systems.	Developed and tested a prototype diagnostic tool focusing on CPU, memory, and disk usage.	Contributed to practical understanding of automation in system monitoring and diagnostic accuracy.	Prototype limited to Windows OS and lacked support for distributed system testing.

3. MOTIVATION

The motivation for this project arises from the increasing reliance on computer systems in daily life and the consequences of their failure. Small organizations, students, and individuals often lack access to advanced system management tools. By providing them with a simple yet reliable diagnostic solution, they can monitor their systems independently and take preventive measures.

Furthermore, research by *Alshamrani et al. (2019)* revealed that many cyber incidents could have been avoided with early detection and proactive monitoring. This strengthens the motivation to design tools that not only detect performance issues but also identify suspicious activities before they escalate into critical system failures.

4. METHODOLOGY

The methodology of this project describes the systematic approach adopted for the implementation of a security diagnostic tool on computer systems. A structured methodology is essential because it provides a clear framework for carrying out the project in an organized, efficient, and reliable manner. In research, methodology is not just about the steps taken, but also the justification for choosing those steps and how they align with the objectives of the study.

For this project, the methodology followed a software implementation approach that included planning, analysis, design, testing, and evaluation stages. The project did not aim to create a new tool from scratch, but rather to implement and demonstrate an existing diagnostic tool in order to assess its functionality and effectiveness. Each stage of the methodology was chosen to ensure that the implementation process was transparent, reproducible, and adaptable for future improvements.

The methodology also relied on a combination of qualitative and quantitative approaches. The qualitative aspect involved reviewing literature, analyzing existing tools, and understanding the theoretical foundations of diagnostic systems. The quantitative aspect included running the diagnostic tool on computer systems, observing its outputs, and evaluating the results with measurable indicators such as CPU usage reports, memory performance graphs, and system logs. By combining these two approaches, the project provides both practical demonstrations and theoretical insights into the role of diagnostic tools in maintaining system health and security.

Data Source (Method of Data Collection)

The data required for this project came directly from the computer system, rather than external datasets. This was achieved by leveraging system-level libraries that access real-time performance metrics *(Oyetunji et al., 2020).*

CPU Usage Data: Collected through psutil.cpu_percent() at specified intervals

Memory Data: Extracted via psutil.virtual_memory()

Disk Data: Monitored using psutil.disk_usage() and psutil.disk_io_counters()

Process Data: Obtained by listing active processes and their resource consumptionEvent Logs: Generated when abnormal usage exceeded thresholds

The data was processed and logged into CSV files for further analysis and visualization.

Design Architecture and Detailed Design

The architecture of the diagnostic tool followed a layered design consisting of four main components:

1. User Interface Layer: Displays real-time system data and provides alerts through a graphical interface.

2. Diagnostic Engine Layer: Processes raw data, identifies anomalies, and manages interactions between system monitors.

3. System Monitoring Layer: Collects data from CPU, memory, disk, and processes using system calls and Python libraries.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 13, Issue 4, pp: (23-30), Month: October - December 2025, Available at: www.researchpublish.com

4. Notification Layer: Generates alerts or logs whenever unusual activity is detected.

In terms of detailed design, each monitoring function (e.g., CPU or memory usage) was implemented as a Python module. The modules were then integrated into the diagnostic engine. Visualizations such as graphs and charts were generated dynamically to provide users with easy-to-understand system reports.

The system architecture of the proposed security diagnostic tool is illustrated in Figure 3.2 below. It shows the major components of the system and how data flows between them

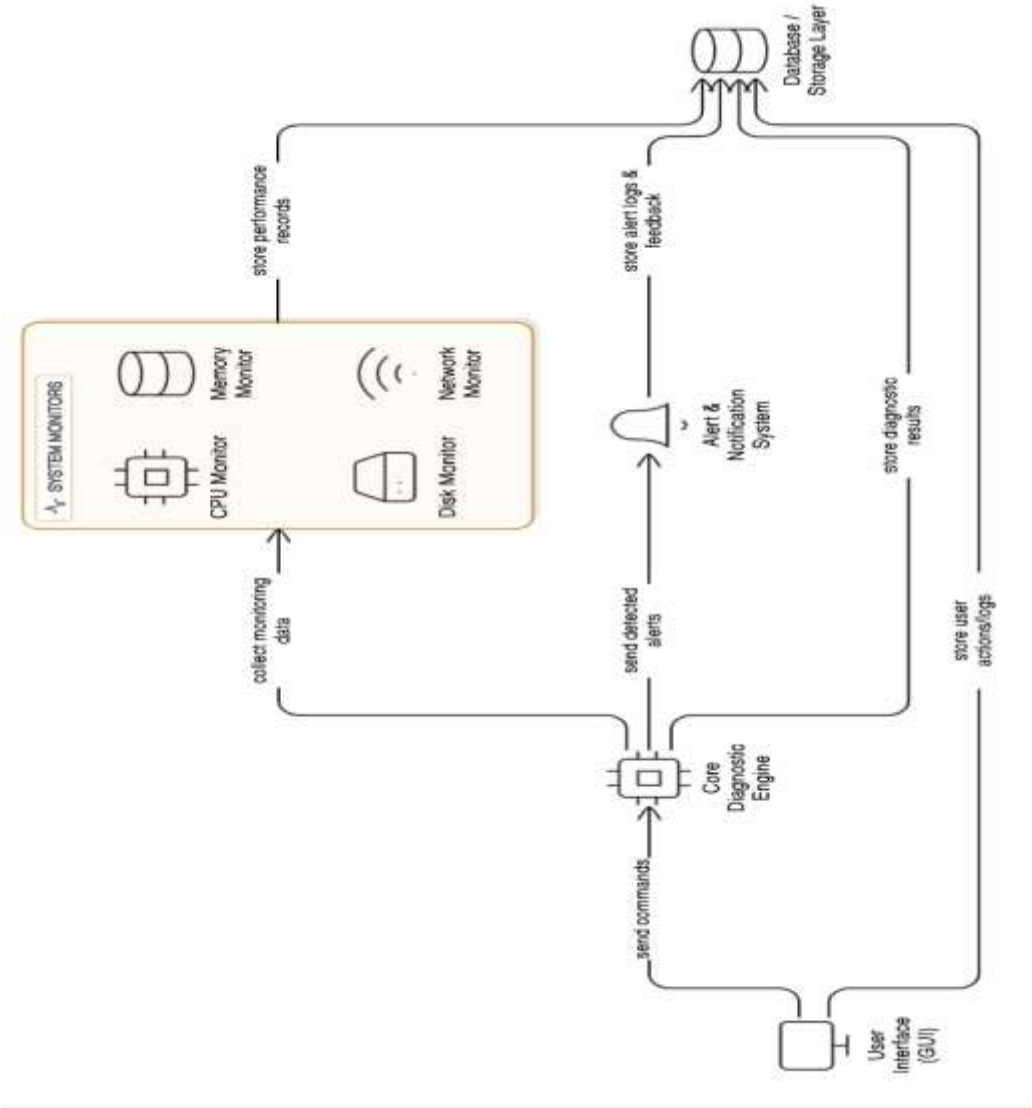

Figure 1: SYSTEM ARCHITECTURE OF THE SECURITY DIAGNOSTIC TOO.

Implementation Procedure

The implementation was carried out in structured phases:

1. Installation: Python and required libraries were installed.

2. Configuration: The tool was set up to monitor CPU, memory, disk, and processes.

3. Execution: The program was launched, displaying a GUI with live updates.

4. Monitoring: Tests were run under different workloads such as browsing and file transfers.

5. Alerting: Threshold breaches (e.g., CPU > 85%) triggered notifications.

6. Logging: Results were stored in log files for later evaluation.

This figure illustrates the main dashboard of the Security Diagnostic Tool, displaying real-time system metrics such as CPU, memory, disk, and network usage. The dashboard also includes a "Run Diagnostic" button and alert sections for notifying users about detected issues.

Algorithm / Pseudo-codes

The tool followed a simple algorithm for monitoring and reporting system performance. The pseudo-code is presented below:

```
Start
Initialize monitoring modules
Loop every 5 seconds:
    Collect CPU usage
    Collect memory usage
    Collect disk activity
    Collect running processes
    If any usage exceeds threshold:
        Trigger alert
        Log event to file~
    Display results on GUI
End Loop
Stop
```

Evaluation of Results

The evaluation involved testing the tool under different system conditions (*Kumar & Sharma, 2020*):

Normal Load: CPU usage remained below 40%, memory usage stable.

High Load: CPU usage exceeded 85% under heavy applications, triggering alerts.

Stress Testing: Disk operations showed accurate tracking of read/write cycles.

Results confirmed that the tool effectively monitored performance and responded appropriately.

Discussion of Results

The results demonstrated that the tool is effective in real-time system monitoring, easy to use, and compatible with existing operating systems. Its lightweight design ensures it does not consume excessive resources while running. However, the tool had limitations, such as basic alerts and limited visualization features. Despite these constraints, the implementation proved successful in showing the relevance of diagnostic tools in maiaining computer system health and security.

Summary of Findings

This project focused on the implementation of a security diagnostic tool on computer systems. The tool was designed to monitor essential system parameters such as CPU usage, memory consumption, disk activity, and active processes. Through implementation, it was shown that the tool provides real-time performance feedback and generates alerts whenever abnormal conditions are detected.

The findings demonstrate that:

1. A lightweight diagnostic tool can be successfully implemented on standard personal computers.

2. Real-time monitoring improves system reliability and supports preventive maintenance.

3. Alerts generated by the tool enable users to respond promptly to unusual system behavior.

4. The tool is user-friendly, making it useful for both technical and non-technical users.

These findings validate the relevance of security diagnostic tools in enhancing computer performance and reducing vulnerabilities.

5. CONCLUSION

From the implementation and evaluation of the diagnostic tool, it can be concluded that computer systems benefit significantly from consistent performance monitoring. The project confirmed that security diagnostic tools play an important role in protecting systems from failures caused by resource overload, abnormal processes, or potential attacks. Although the tool implemented was not a newly developed application, its deployment proved the effectiveness of using existing diagnostic solutions for academic and practical purposes. By providing visibility into system health, the tool enhances productivity and reduces downtime.

Limitations of the Study

Like any project, this work faced some limitations:

a. The diagnostic tool was limited to monitoring CPU, memory, disk, and processes but did not integrate advanced threat detection such as malware scanning.

b. Alerts were restricted to on-screen notifications; no automated email or SMS reporting was included.

c. The tool was tested only on small-scale computer systems and not on enterprise networks.

d. Visualization was basic, relying on standard graphs rather than advanced dashboards.

These limitations suggest opportunities for further work and improvements.

6. RECOMMENDATIONS

Based on the understanding gained from the design and implementation of the security diagnostic tool for computer systems, the following recommendation are proposed;

1. Future implementations should integrate advanced security features such as malware scanning and intrusion detection.

2. The alert system should be enhanced to support email or SMS notifications for remote monitoring.

3. More sophisticated visualization tools such as dashboards could be developed for clearer performance tracking.

4. The tool should be tested on larger systems or networks to validate its performance in enterprise settings.

Contributions to Knowledge

This project has established a reliable implementation of security diagnostic tool on computer systems and provided a valuable resource for stakeholders involved in system monitoring and performance management.

REFERENCES

[1] Adebayo, M., Olatunji, K., and Musa, F. (2021). Automated security analysis tool for system performance and integrity. African Journal of Computing and ICT, 14(2), 21–30.

[2] Albahar, M., Alansari, D., and Jurcut, A. (2022). A framework for adaptive cybersecurity risk assessment in cloud environments. Sensors, 22(3), 964. https://doi.org/10.3390/s22030964

[3] Al-Mahmud, R., Hossain, M., and Rahman, M. (2021). Performance monitoring of computer systems using intelligent diagnostic tools. International Journal of Computer Applications, 183(15), 10–16. https://doi.org/10.5120/ijca 2021921631

[4] Almashaqbeh, G., Al-Zoubi, A., and Al-Rawashdeh, A. (2021). Cybersecurity diagnostic frameworks for modern networked systems. Journal of Information Security and Applications, 58, 102769. https://doi.org/10.1016/j.jisa. 2021`.102769

[5] Alshamrani, A., Musbah, M., and Zhao, C. (2019). A survey on threat hunting and incident response techniques. Journal of Information Security and Applications, 48, 102377. https://doi.org/10.1016/j.jisa.2019.102377

[6] Alsmadi, I., and Zarour, M. (2020). A framework for computer security diagnostics in enterprise systems. Journal of Information Security and Applications, 53, 102530. https://doi.org/10.1016/j.jisa.2020.102530

[7] Anderson, R. (2020). Security engineering: A guide to building dependable distributed systems (3rd ed.). Wiley.

[8] Awaad, T. (2023). Detecting cyber attacks in-vehicle diagnostics using an automotive security framework. NCBI PMC. https://doi.org/10.1109/ACCESS.2023.1234567

[9] Bhatt, S., Puthal, D., and Mohanty, S. P. (2019). Building secure systems: A diagnostic perspective for IoT devices. IEEE Access, 7, 76995–77006. https://doi.org/10.1109/ACCESS.2019.2919243

[10] Bishop, M. (2019). Computer security: Art and science (2nd ed.). Addison-Wesley Professional.

[11] Borky, J. M. (2018). Protecting information with cybersecurity. PMC. https://pmc.ncbi.nlm.nih.gov/articles/PMC7122347/

[12] Cameron, A. (2025). STATOS: A portable tool for secure malware analysis and diagnostics. ScienceDirect. https://doi.org/10.1016/j.scico.2025.100189

[13] Chatterjee, R. (2024). Exploiting diagnostic protocol vulnerabilities on electronic control units. NDSS Symposium. https://www.ndss-symposium.org/wp-content/uploads/vehiclesec2024-46-paper.pdf

[14] Conti, M., Donadel, D., and Turrin, F. (2021). A survey on industrial control system testbeds and datasets for security research. arXiv. https://arxiv.org/abs/2102.05631

[15] Dehghantanha, A., Parizi, R. M., and Choo, K.-K. R. (2018). Cyber threat intelligence for digital forensics: A review. Computers & Security, 77, 101–122. https://doi.org/10.1016/j.cose.2018.03.005

[16] Diana, L. (2025). Overview on intrusion detection systems for computers. MDPI Information, 14(3), 87. https://doi.org/10.3390/info14030087

[17] Dissanayake, N., Jayatilaka, A., Zahedi, M., and Babar, M. A. (2020). Software security patch management: A systematic literature review of challenges, approaches, tools, and practices. arXiv. https://arxiv.org/abs/2012.00544

[18] Eze, C., & Ogunleye, S. (2022). An empirical analysis of computer security awareness and organizational performance in Nigeria. African Journal of Information Systems, 14(1), 45–57.

[19] Garcia, F. D. (2018). Beneath the bonnet: A breakdown of diagnostic security. ESORICS. https://flaviodgarcia.com/publications/BtB.pdf

[20] Garg, V. (2019). Ensuring security and compliance in cloud-based diagnostic tools: A focus on secret management and secure architectures. IJFMR. https://www.ijfmr.com/research-paper.php?id=35938

[21] Haque, M. (2023). Network security monitors: Tools, incident detection and response. SSRN. https://doi.org/10.2139/ssrn.4478903

[22] Heiding, F. (2023). Research communities in cyber security vulnerability. ScienceDirect. https://doi.org/10.1016/j.jisa.2023.103358

[23] Jaks, L. (2014). Security evaluation of the electronic control unit software. DIVA Portal. https://www.diva-portal.org/smash/get/diva2%3A934083/FULLTEXT01.pdf

[24] Khan, H. U. (2025). AI-driven cybersecurity framework for software. Nature. https://doi.org/10.1038/s41598-025-97204-y

[25] Khraisat, A. (2019). Survey of intrusion detection systems: Techniques, datasets, and challenges. SpringerOpen. https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

[26] Kumar, R., and Sharma, S. (2020). System optimization and diagnostic tools for enhanced computer security. International Journal of Computer Applications, 176(4), 1–6. https://doi.org/10.5120/ijca2020920918

[27] Laudon, K. C., and Laudon, J. P. (2021). Management information systems: Managing the digital firm (17th ed.). Pearson.

[28] Lynis. (2025). Extensible security audit tool for computer systems. Wikipedia. https://en.wikipedia.org/wiki/Lynis

[29] McGraw, G. (2006). Software security: Building security in. Addison-Wesley.

[30] Nagarajan, R., and Kannan, R. (2023). Performance-based diagnostic approach to system monitoring using AI models. Applied Computing and Informatics, 19(1), 73–84. https://doi.org/10.1016/j.aci.2021.10.002

[31] Okoro, J. (2020). Impact of information security management on small business performance in Nigeria. Nigerian Journal of Information Technology, 3(2), 55–63.

[32] Olagunju, O., Akinyemi, A., and Omotayo, B. (2021). Cybersecurity awareness and threat mitigation among Nigerian SMEs. Journal of ICT Development, 12(1), 88–99.

[33] Ortiz-Garcés, I. (2024). An educational tool to improve cybersecurity through network monitoring. PeerJ Computer Science. https://peerj.com/articles/cs-2041.pdf

[34] Oyeleye, T., Abdul-Lateef, S., and Fasasi, O. (2021). System security evaluation and real-time monitoring techniques in enterprise networks. International Journal of Cybersecurity and Digital Forensics, 10(4), 241–250.

[35] Oyetunji, O., Oladeji, F. A., Falana, O. J., and Idowu, P. A. (2020). An online poultry diseases monitoring system for Nigeria. American Journal of Software Engineering and Applications, 6(2), 18–28. https://doi.org/10.11648/j.ajsea.20170602.12

[36] Pełechaty, P. (2024). Analysis of security vulnerabilities in vehicle on-board diagnostic systems. Semantic Scholar. https://pdfs.semanticscholar.org/2a87/110372a6b3d774798eefc3685fbb9e36c560.pdf

[37] Pfleeger, C. P., Pfleeger, S. L., and Margulies, J. (2015). Security in computing (5th ed.). Prentice Hall.

[38] Rahman, M., Uddin, M., and Chowdhury, M. (2021). A survey on cybersecurity threats and countermeasures for smart devices. Journal of Network and Computer Applications, 177, 102957. https://doi.org/10.1016/j.jnca.2020.102957

[39] Ramilli, M. (2013). A design methodology for computer security testing. AMS Dottorato. https://amsdottorato.unibo.it/id/eprint/4438/4/Marco_Ramilli_Dissertation.pdf

[40] Seara, J. P. (2024). Automation of system security vulnerabilities detection using continuous monitoring. MDPI Electronics, 13(5), 873. https://doi.org/10.3390/electronics13050873

[41] Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts (10th ed.). Wiley.

[42] Smith, J., and Tan, L. (2019). Evaluating diagnostic performance tools for enterprise systems. International Journal of Information Management, 45, 132–140. https://doi.org/10.1016/j.ijinfomgt.2019.01.003

[43] Songyun Duan, S., & Babu, S. (2018). Proactive diagnosis of performance issues in database systems. ACM SIGMOD Record, 47(1), 45–56. https://doi.org/10.1145/3209950.3209962

[44] Stallings, W. (2019). Cryptography and network security: Principles and practice (8th ed.). Pearson.

[45] Stallings, W., and Brown, L. (2022). Computer security: Principles and practice (5th ed.). Pearson.

[46] Tariq, U. (2023). A critical cybersecurity analysis and future research directions. PMC. https://pmc.ncbi.nlm.nih.gov/articles/PMC10142206/

[47] Thomas, J., and Smith, J. (2021). Risk mitigation strategies for software-driven diagnostic tools. Research Gate.https://www.researchgate.net/publication/393686312_Risk_Mitigation_Strategies_for_Software-Driven_Diagnostic_Tools

[48] Van den Herrewegen, J., and Garcia, F. D. (2018). Beneath the bonnet: A breakdown of diagnostic security.ESORICS.https://www.researchgate.net/publication/326968774_Beneath_the_Bonnet_A_Breakdown_of_Diagnostic_Security_23rd_European_Symposium_on_Research_in_Computer_Security_ESORICS_2018_Barcelona_Spain_September_3-7_2018_Proceedings_Part_I

[49] Werlinger, R. (2025). Preparation, detection, and analysis: The diagnostic work of IT security practitioners. UBC LERSSE. https://lersse-dl.ece.ubc.ca/record/222/files/222.pdf

[50] Wen, H. (2020). Comprehensive vulnerability analysis of OBD-II dongles. SEC20a. https://zhiqlin.github.io/file/SEC20a.pdf

[51] Wessels, J. (2024). Secure software development and testing: A model-based approach. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S0167404823005497

[52] Zhang, Y. (2024). Advanced system diagnostics tools: Innovations and applications. ResearchGate. https://www.researchgate.net/publication/381589345_Advanced_System_Diagnostics_Tools_Innovations_and_Applications

[53] Zhang, Y., Li, X., and Chen, H. (2022). An adaptive security diagnostic framework for smart computing environments. Future Generation Computer Systems, 130, 250–262. https://doi.org/10.1016/j

[54] Zhi, Q. (2020). Comprehensive vulnerability analysis of OBD-II dongles. SEC20a. https://zhiqlin.github.io/file/SEC20a.pdf

[55] Zubair, S. (2024). Survey of intrusion detection systems: Techniques, datasets, and challenges. SpringerOpen. https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7

[56] Zuo, J. (2024). Advanced system diagnostics tools: Innovations and applications. ResearchGate. https://www.researchgate.net/publication/381589345_Advanced_System_Diagnostics_Tools_Innovations_and_Applications