iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.79029

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

ARGUS - Rootkit Monitoring With Module Integrity Verification

Ananthankrishnan R[1], Devikrishna S[2], Jayesh J[3], Sivani V[4], Binoy D L[5]
[1, 2, 3, 4]*Department of Computer Science and Engineering (Cyber Security), Rajadhani Institute of Engineering and Technology, Thiruvananthapuram, Kerala, India*
[5]*Assistant Professor, Department of Computer Science and Engineering, Rajadhani Institute of Engineering and Technology, Thiruvananthapuram, Kerala, India*

Abstract: ***Argus is a Linux-based rootkit detection framework designed to identify stealth malware by comparing the system's kernel-level state with the view presented to user-space utilities. Built using a Loadable Kernel Module (LKM) and a Python-based client, Argus establishes a trusted "ground truth" by directly inspecting kernel data structures for running processes, loaded modules, and active network sockets. It then contrasts this data against outputs from standard tools such as ps, lsmod, and ss to uncover discrepancies indicative of process hiding, module hiding, or concealed network ports. The framework also supports configurable UDP alerting for centralized monitoring and provides high-level threat classification based on detected anomalies. While effective against kernel-level stealth techniques, Argus is designed primarily for educational and research purposes, highlighting both the strengths and limitations of runtime kernel integrity verification in modern Linux environments.***
Keywords: rootkit,(LKM Loadable Kernel Module), udp, network sockets.

I. INTRODUCTION

The sophistication of cyber threats has made the protection of modern operating systems more complex. In this regard, rootkits have been identified as one of the most potent threats in the modern digital landscape. The threats are able to maintain root access and remain undetected by manipulating system calls and changing the output of user-space tools such as ps, lsmod, and ss. Therefore, the detection of such threats by conventional detection tools that depend on user-space visibility has become impossible. In this context, the need for the introduction of Argus as a security framework that detects stealthy malware threats in the Linux operating system using the kernel verification approach has been identified.

Argus has been developed with two primary components: the Loadable Kernel Module and the Python-based user-space client. The kernel module functions as the trusted observer that traverses the kernel's internal data structures to obtain precise information about running processes, loaded modules, and active network sockets. The precise information obtained represents the ground truth. The user-space client receives the precise information obtained by the kernel module and compares it with the output of standard Linux tools. The client is able to identify the discrepancies that may be indicative of hidden processes, concealed modules, and unauthorized ports, which are common tactics used by rootkits.

The framework is lightweight, modular, and easy to deploy, as it only requires standard Linux commands such as "insmod" and "rmmod" to function, with no significant modifications to the system necessary. Its detection method is read-only, which means that the kernel memory space and standard system operations are left unchanged. Moreover, the Argus framework includes a UDP-based alerting system, which sends real-time alerts to a user-space listener in the event of any anomalies, thus making the system more secure. The Argus framework utilizes kernel-level monitoring, dual view comparison, and crypto integrity verification of kernel modules, making it a robust solution for rootkit detection. Unlike other detection methods, Argus does not rely on signature-based detection, which means it can detect unknown rootkits. Although there are limitations to some detection methods due to the current kernel protections, the Argus framework is beneficial in showing the challenges of low-level security issues, which can be used to bridge the gap between theory and practice, thus advancing the development of secure Linux systems.

II. LITERATURE REVIEW

The proposed Argus framework is intended to be lightweight, modular, and deployable in the Linux environment. It is implemented as a Loadable Kernel Module (LKM). This means that the module is dynamically inserted and removed using the insmod and rmmod tools available in the Linux operating system. This ensures that the system is not altered in any significant manner, hence the compatibility of the system with existing system configurations.

Another important aspect of the Argus framework is its non-intrusive nature. This is ensured through the implementation of the detection mechanism, which is strictly read-only. This means that the mechanism is implemented in such a manner that the kernel memory is not altered in any way. This is important in ensuring the stability of the system, hence the non-intrusive nature of the Argus framework.

Another important aspect of the Argus framework is the implementation of the alerting mechanism. This mechanism is implemented using the user datagram protocol (UDP). This ensures that the system is able to send real-time alerts to the user whenever there is the detection of anomalies such as the presence of hidden processes and the modification of the kernel code. This is important in ensuring the overall security of the system.

The strength of the Argus framework is its ability to combine kernel-level monitoring, dual-view comparison, and kernel module integrity verification. This way, it can compare “ground truth” data generated by kernel-level monitoring with data generated in user space. This helps in identifying inconsistencies in data, which is indicative of “stealthy” behavior of rootkits. In addition, it is also capable of identifying any unauthorized modification of kernel modules through its integration of kernel module integrity verification mechanisms.

The main difference between the Argus framework and other signature-based detection mechanisms is its focus on behavioral anomalies and structural inconsistencies. This way, it can detect unknown or “obfuscated” rootkits. Though some limitations are evident due to kernel protection mechanisms, it is also apparent that it offers valuable insights into the challenges of developing kernel-level security solutions. Therefore, it can be said that it is an excellent example of how security theories can be applied in practical scenarios.

III. METHODOLOGY

The proposed framework of Argus uses kernel-level monitoring to track stealth-oriented rootkits in Linux-based systems. The methodology is based on creating a trusted image of the system by accessing kernel data structures instead of depending solely on user-space applications. This is done by inserting a Loadable Kernel Module (LKM) in the kernel space, which provides precise details of active processes, kernel modules, and network sockets

Once the required data is available at the kernel level, the parallel collection of system-related data is performed using user-space tools available in the Linux operating system. A comparative analysis is then performed between the two sets of data. Any discrepancies that are detected during the comparison process are considered potential symptoms of rootkits.

In order to improve the accuracy of the rootkit detection process, the proposed methodology includes the cryptographic integrity verification of the kernel modules using hashing algorithms. This ensures the integrity of the system by detecting any unauthorized modifications to the kernel modules. Furthermore, the proposed system includes the implementation of the UDP protocol-based alerting mechanism, which ensures the effectiveness of the rootkit detection process.

A. Kernel Module Initialization

The proposed Loadable Kernel Module of Argus is inserted into the Linux kernel by executing standard kernel commands.

B. Kernel-Level Data Collection

The kernel module accesses kernel data structures to collect precise details of:

Active processes

Loaded kernel modules

Open network sockets

This is considered the trusted image of the system.

C. User Space Data Acquisition

In parallel, the client written in Python would continue to acquire system-related information using standard Linux-based tools such as ‘ps,’ ‘lsmod,’ and ‘ss,’ which represents the user space.

D. Data Normalization and Preprocessing

The data acquired in the kernel space and the user space would be normalized and preprocessed to ensure consistency between the two data sets.

E. Dual View Comparison

The comparison algorithm wo
uld be implemented to compare the inconsistencies between the two data sets.
Missing entries in user space imply hidden entities
Excess entries imply anomalies

F. Anomaly Detection

The inconsistencies would be analyzed to detect the following anomalies:
Hidden Processes
Hidden Kernel Modules
Hidden Network Ports

G. Kernel Module Integrity Verification

The integrity of the kernel module would be verified using cryptographic hashing algorithms such as SHA-256.

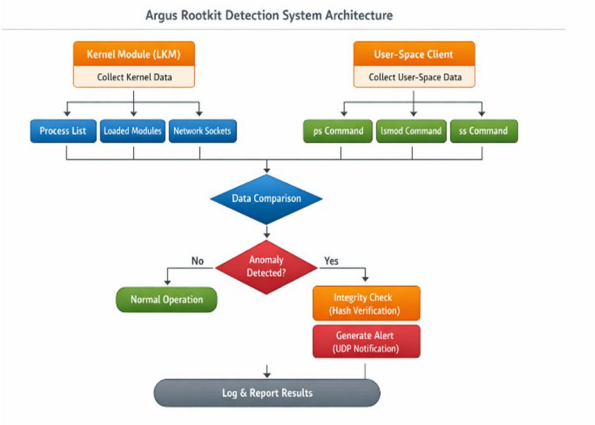
H. Alert Generation

Once the anomalies are detected, the system would generate alerts using the UDP protocol to the user space in real time.

I. Result Logging and Reporting

The anomalies detected would be logged and reported to provide a proper understanding of the threats.

IV. SYSTEM ARCHITECTURE



The Argus system architecture is structured in such a way that it provides a lightweight and modular approach in the detection of rootkits in Linux-based systems. The system is structured in such a way that it provides the layered approach in the detection of rootkits. The layered approach is realized in the system by combining the kernel space with the user space. The system maintains the data collection and analysis functions in such a way that it provides the highest level of efficiency and security.

The central component of the system is the Loadable Kernel Module. The module is responsible for running in the kernel space. The kernel space provides the most precise data regarding the running processes and the modules that are running in the system.

The kernel module accesses the kernel data structure and provides the most precise data regarding the running processes and the modules that are running in the system. The kernel module provides the most precise data because it runs in the kernel space. The user space component of the system is realized in the system in the form of the client.

The client is implemented in the system in the form of the Python-based client. The client sends the request to the kernel module and receives the data. The client receives the data and at the same time sends the request to the Linux-based utilities such as the ps, lsmod, and ss. The client receives the data and sends the data to the comparison engine.

The comparison engine compares the data and provides the output. The output is realized in the system in the form of the UDP-based alerting system. The UDP-based alerting system provides the highest level of efficiency in the system. The system is simple and provides the highest level of efficiency..

V. RESULTS

The Argus framework was implemented and tested in a controlled environment running Linux. This ensured its effectiveness in detecting stealth-oriented rootkits. It showed its capability to collect kernel-level data and compare it with user-space data without interfering with normal system functioning. The lightweight nature of the framework ensured it did not cause significant performance degradation during execution.

The dual view comparison mechanism used in testing proved to be effective in detecting inconsistencies between kernel-space and user-space data. Hidden processes and modules, which could not be detected through normal Linux environment checks, were found through kernel-level data inspection. This ensured the reliability of kernel-level data in verifying system state.

The integrity verification mechanism through the use of cryptographic hashing also proved to be effective in detecting unauthorized changes to kernel modules. This ensured that the framework could highlight possible tampering with kernel modules through hash value comparison. This further improves its detection capabilities beyond mere anomaly detection.

The UDP-based alerting mechanism also performed its role in generating alerts in real-time upon detection of suspicious activities. This ensured efficient monitoring of system activities. From the testing, it is evident that the Argus framework is effective in detecting stealth-oriented rootkits without affecting system stability.

VI. CONCLUSION

The Argus Rootkit Monitoring System offers an effective solution in the detection of malware with stealthy characteristics by employing kernel-level monitoring in conjunction with dual view comparison methodologies. By using kernel-derived “ground truth,” the system can bypass the limitations of conventional detection methodologies to provide better insight into the activities of malicious rootkits.

The incorporation of the cryptographic integrity verification process strengthens the framework in the detection of unauthorized modifications to kernel modules. With real-time notification, the non-intrusive design of the Argus Rootkit Monitoring System offers an effective solution in monitoring system integrity in an efficient manner.

Although some limitations are associated with the Argus Rootkit Monitoring System due to the use of modern kernel protection, the project effectively demonstrates the significance of low-level security analysis. It can be used as an effective foundation for future research in the development of rootkit detection methodologies, leading to the development of a more secure Linux environment.

REFERENCES

[1] Tian D, Ying Q, Jia X, Ma R, Hu C, Liu W (2021) MDCHD: a novel malware detection method in cloud using hardware trace and deep learning. Computer Networks 198:108394. https:// doi. org/ 10. 1016/j. comnet. 2021. 108394. (ISSN 1389-1286)

[2] MoonLeeHeoKimPaekKang HHIKYBB (2017) Detecting and preventing kernel rootkit attacks with bus snooping. IEEE Transactions on Depend able and Secure Computing 14(2):145–157. https:// doi. org/ 10. 1109/ TDSC. 2015. 24438 03

[3] Zhou H, Fei C, Ni L, Wu B, Li G, Han K (2022) “Detecting Kernel Rootkits in a Virtualized Infrastructure with Low-Level Architectural Features,” 2022 IEEE 5th International Conference on Electronics and Communication Engineering (ICECE), Xi’an, China. pp 244–247. https:// doi. org/ 10. 1109/ ICECE 56287. 2022. 10048 623 4. 5. 6. 7. 8. 9.

[4] Krishnamurthy P, Salehghaffari H, Duraisamy S, Karri R, Khorrami F (2019) “Stealthy Rootkits in Smart Grid Controllers,” 2019 IEEE 37th International Conference on Computer Design (ICCD), Abu Dhabi, United Arab Emir ates. pp 20–28. https:// doi. org/ 10. 1109/ ICCD4 6524. 2019. 00012

[5] Xing X, Jin X, Elahi H, Jiang H, Wang G (2022) A malware detection approach using autoencoder in deep learning. IEEE Access 10:25696 25706. https:// doi. org/ 10. 1109/ ACCESS. 2022. 31556 95

[6] I. Kuzminykh and M. Yevdokymenko, "Analysis of Security of Rootkit Detection Methods," 2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT), Kyiv, Ukraine, 2019, pp. 196-199, https:// doi. org/ 10. 1109/ ATIT4 9449. 2019. 90304 28

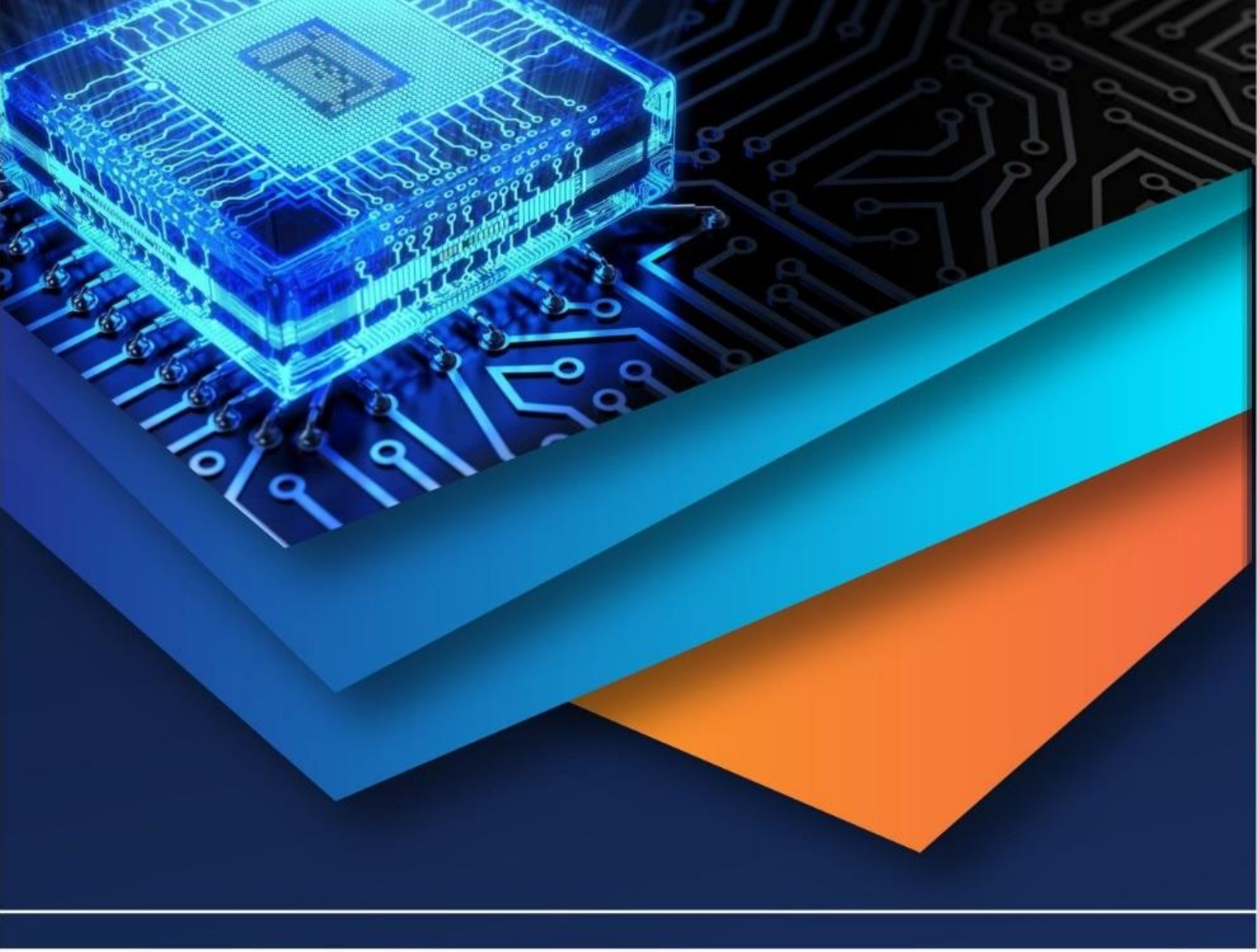
[7] Mohammadhadi Alaeiyan, Saeed Parsa, Mauro Conti, “Analysis and classification of context-based malware behavior”,Computer Communications,volume 136, February 2019, Pages 76-90, 10.1016/ j.co m c o m . 2019 .01.003.

[8] Xiao J, Lu L, Wang H, Zhu X (2016) “HyperLink: Virtual Machine Introspec tion and Memory Forensic Analysis without Kernel Source Code,” 2016 IEEE International Conference on Autonomic Computing (ICAC), Wuerz burg, Germany. pp 127–136. https:// doi. org/ 10. 1109/ ICAC. 2016. 46

[9] S. Kumar Verma, N. Anjum, A. Sharma and A. Mishra, "iSIMP with Integrity Validation using MD5 Hash," 2021 International Conference on Computational Performance 38 Department of Computer Science and Engineering (Cyber Security) ARGUS - ROOTKIT MONITORING WITH MODULE INTEGRITY VERIFICATION Evaluation (ComPE), Shillong, India, 2021, pp. 094-097, https:// doi. org/ 10. 1109/ ComPE 53109. 2021. 97524 33. 10.

[10] Alshamrani SS. Analysis of MachineLearning Based Technique for Mal ware Identification and Classification of Portable Document FormatFiles, Hindawi Security and Communication Networks Volume 2022, Article ID 7611741, 10 pages https:// doi. org/ 10. 1155/ 2022/ 76117 41

[11] Donghai Tian, Rui Ma , Xiaoqi Jia, and Changzhen Hu, "A Kernel Rootkit Detection Approach based on Virtualization and Machine Learning" IEEE Access PP (99):1-1 july, 2019. 12.
[12] Chin-Ling Chen, Supaporn Punya, "An enhanced WPA2/PSK for prevent ing authentication cracking", The International Journal of Informatics and Communication Technology (IJ ICT), Vol.10, No.2, August 2021, pp. 85-92,DOI: https:// doi. org/ 10. 11591/ ijict. v10i2. pp85- 92.
[13] Sanjay Sharma, C. Ramakrishna and Sanjay K. Sahay, "Detection of Advanced Malware by Machine Learning Techniques" Access AISC, Volume 742, 2019.
[14] Panker T, Nissim N (2021) Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in Linux cloud environments. Knowl. Based Syst. 226:107095
[15] Ullah A, Laassar I, Şahin CB, Dinle OB, Aznaoui H, "Cloud and internet-of things secure integration along with security concerns", International Journal of Informatics and Communication Technology, Vol. 12, No. 1, https:// doi. org/ 10. 11591/ ijict. v12i1. pp62- 71Agus Reza A. Nurwa, Muhammad Hasbi, Dimas F. Priambodo, Wawan L. Y. Saptomo, Daffa A. P. Yusa, and Setiyowati Z. Zaini, "Portable



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY