iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82610

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 V May 2026- Available at www.ijraset.com

Adaptive AI Defenses against Polymorphic Malware

B. Satya Vijaya, CH. Sai Pramoda, R. Harika Vara Prasanna, P. Srinivasa Reddi
Department of CSE (IOT and Cyber Security with Block Chain Technology), Sasi Institute of Technology & Engineering College

Abstract: ***As cyber threats grow more sophisticated, polymorphic malware has emerged as one of the most persistent adversaries facing modern security infrastructure. Unlike conventional malware, polymorphic variants continuously restructure their own code while retaining their malicious payload — a property that renders signature-based antivirus tools largely ineffective. Detecting something that looks different every time it appears demands a fundamentally different approach. This paper introduces an Adaptive AI Defense System built to tackle this exact challenge. Rather than matching code against a static library of known threats, the system scrutinizes behavioral indicators — patterns of system activity, file interaction, network communication, and structural complexity — to distinguish malicious software from benign programs. A core novelty of our approach is the simulation-based Red Team / Blue Team framework: an AIdriven generator produces synthetic polymorphic malware variants while a parallel defender engine analyzes and classifies them in real time, creating a controlled adversarial loop. The system is deployed as a web application with a Python/Flask backend and an interactive dashboard that surfaces detection confidence scores, threat classifications, and performance telemetry as simulations unfold. Experimental evaluation shows an 88.4% detection rate, outperforming traditional signature-based (72%), static rule (79%), and heuristic (83%) methods. Beyond accuracy, the framework provides a safe, reproducible environment for studying how adaptive defenses respond as attack strategies evolve — a resource of practical value for both cybersecurity researchers and practitioners***

Keywords: Polymorphic Malware, Artificial Intelligence, Machine Learning, Behavioral Analysis, Cybersecurity, Malware Detection, Adaptive Systems.

I. INTRODUCTION

The past two decades have seen a dramatic expansion in the scale, complexity, and ambition of cyberattacks. Malware — software engineered to infiltrate, disrupt, or exploit computing environments — sits at the center of this threat landscape. What makes today's malware ecosystem particularly troubling is not just its volume but its adaptability. Threat actors no longer simply release a piece of malicious code and hope it goes undetected; they build malware capable of reinventing itself. Polymorphic malware exemplifies this approach. At its core, a polymorphic program preserves its functional logic while algorithmically mutating its surface-level code structure — swapping instruction sequences, re-encrypting payloads, inserting junk code — on each iteration or deployment. The result is a moving target that conventional antivirus engines, which depend on matching code against pre-catalogued signatures, simply cannot keep up with.

A signature written for today's variant is already obsolete by tomorrow. Machine learning and AI-driven detection have begun to close this gap. By shifting the focus from 'what does the code look like?' to 'what does the code do?', behavioral analysis methods can flag suspicious activity regardless of how the underlying implementation has changed. Monitoring system call patterns, tracking file access anomalies, and examining network communication fingerprints all provide detection signals that survive code mutation. Yet current AI-based solutions carry their own limitations. Many require substantial computational overhead, lack real-time interpretability, or depend on executing actual malware in sandboxed environments — a setup that introduces operational risk and limits accessibility for research and training purposes. There is a clear need for a framework that is simultaneously safe, adaptive, and transparent.

This paper presents an Adaptive AI Defense System that meets these requirements. Built on a simulation-based architecture, the system generates synthetic polymorphic malware samples — no real malicious code is executed — and subjects them to a feature extraction pipeline that evaluates obfuscation depth, encryption indicators, code complexity, and behavioral anomalies. A rule-informed classification engine then issues a threat verdict alongside a confidence score. The dual Red Team / Blue Team simulation model mirrors real-world attack-response dynamics, offering a meaningful testbed for studying adaptive cybersecurity strategy without compromising operational safety.

II. LITERATURE REVIEW

Malware detection has a rich research history, spanning from early data-mining experiments through today's deep learning architectures. Schultz et al. [1] were among the first to systematically apply data mining to executable analysis, demonstrating that behavioral and structural features of malicious binaries could be extracted and used to train classifiers — work that established the intellectual foundation for automated detection pipelines. Static analysis became a dominant paradigm in the years that followed. By dissecting binary code, instruction sequences, and control-flow graphs without running the program, researchers could identify known malware patterns quickly and at scale [1]. The limitation of this approach became apparent as malware authors adopted obfuscation and polymorphic packing: a program that looks different every time it executes breaks any classifier trained on fixed code representations. Behavioral detection emerged as the logical response. Rather than analyzing code structure, these techniques observe what a program actually does when it runs — which files it touches, what network connections it opens, which system calls it issues. Bayer et al. [2] showed that clustering malware samples by runtime behavior could effectively group related variants even when their static signatures differed dramatically. This observation underpinned much of the subsequent work on dynamic analysis. The arrival of machine learning brought new power to both approaches. Ye et al. [5] provided a comprehensive survey of ML-based detection, demonstrating that properly trained models could generalize to previously unseen malware families with significantly higher accuracy than rule-based systems. Deep learning extended this further: the Deep4MalDroid [4] and DL4MD [6] frameworks applied multi-layer neural networks to Android and general malware respectively, automatically learning hierarchical feature representations that captured subtle malicious patterns human analysts might overlook. Sequential models have proven particularly effective for polymorphic threats. Pascanu et al. [8] demonstrated that recurrent neural networks, applied to ordered sequences of system calls, could capture temporal patterns in program behavior — patterns that persist even when code-level mutations occur. Kolosnjaji et al. [9] refined this approach, showing that system call sequence analysis provided a robust detection signal specifically for advanced evasive malware. The adversarial dimension of this field deserves attention. Anderson and Roth [7] showed that machine learning classifiers are themselves vulnerable to adversarial manipulation: targeted perturbations to malware samples can cause models to misclassify them as benign, raising questions about the robustness of any fixed detection model. Goodfellow et al.'s work on Generative Adversarial Networks [11], while not specific to malware, has been applied in this domain to model precisely this dynamic — training a generator to produce evasive variants and a discriminator to detect them, progressively strengthening both. Taken together, this body of research points toward an important conclusion: no single technique is sufficient. Signature methods excel on known threats; behavioral methods generalize better; neural architectures capture complexity that rules miss; but each can be circumvented under the right conditions. The most resilient detection systems combine multiple signals and adapt to new attack patterns over time. This observation directly motivates the integrated, simulation-driven architecture proposed in this paper.

III. METHODOLOGY

The central design decision behind this system is to prioritize behavioral evidence over codelevel signatures. A polymorphic program can look entirely different from one execution to the next, but its functional behavior — the sequence of operations it must perform to achieve its malicious objective — is far more constrained. Our methodology exploits this asymmetry. Rather than operating on real malware, the system works with synthetically generated samples whose behavioral and structural properties are parameterized to represent a realistic range of threat profiles. This eliminates the security risks associated with deploying actual malicious executables in a testing environment while still enabling meaningful experimentation. Each synthetic sample is assigned attributes across several dimensions: code complexity, degree of obfuscation, encryption activity, presence of suspicious behavioral markers (abnormal file access patterns, unusual inter-process communication, atypical network endpoints), and structural variation relative to known benign programs. Feature extraction is the analytical core of the pipeline. When a sample enters the system, the Feature Extraction Engine decomposes it into a weighted vector of suspicious indicators. Weights reflect both the prevalence and discriminative power of each indicator — encryption-related signals, for instance, carry substantial weight given their strong association with malware that is attempting to conceal its payload or exfiltrate data covertly. Classification is handled by an adaptive scoring engine rather than a monolithic trained model. Incoming feature vectors are evaluated against a set of logical rules that approximate the decision-making of a threat analyst, producing a cumulative threat score. This score is then mapped to one of three categories — Safe, Suspicious, or Malicious — accompanied by a confidence estimate derived from the severity and combination of detected indicators. The rule-based approach was chosen deliberately: it offers interpretability and rapid adaptability, qualities that purely black-box ML models often sacrifice. The system's most distinctive architectural feature is the Red Team / Blue Team simulation loop.

The Red Team module serves as an automated adversary, continuously generating polymorphic malware samples with varying parameters. The Blue Team module — the detection engine — analyzes each sample independently and issues a verdict. The results of this adversarial exchange accumulate in real time, allowing the system to report detection rates, confidence distributions, and behavioral statistics across a statistically meaningful number of simulation runs. This loop not only validates detection capability but also creates a dynamic testbed for studying how detection performance degrades as malware complexity increases.

IV. IMPLEMENTATION

Tools and Frameworks Python serves as the primary implementation language, chosen for its extensive ecosystem of data processing, web development, and scientific computing libraries. Flask provides the server-side application framework, managing routing, request handling, and module coordination. The frontend is built with HTML, CSS, JavaScript, and Bootstrap, delivering a responsive dashboard experience. Pandas and NumPy handle feature computation and score aggregation; Matplotlib and Chart.js render visual performance metrics in the browser. Simulated Threat Scenarios The system operates on synthetic malware samples rather than real executables, enabling safe experimentation in any environment. Samples are generated to represent four primary threat categories: 1. Ransomware — characterized by aggressive encryption activity, bulk file operations, and ransom-note generation patterns 2. Trojan — marked by covert persistence mechanisms, unauthorized network connections, and credential access patterns 3. Keylogger — identified through high-frequency input monitoring and data exfiltration signatures 4. Obfuscated polymorphic variants — distinguished by elevated code complexity, multilayer obfuscation, and self-modification indicators Each sample is assigned quantitative values across all behavioral dimensions prior to entering the detection pipeline. Detection Protocol On receiving a generated sample, the Feature Extraction Engine scores each behavioral indicator on a weighted scale reflecting its discriminative power. The Blue Team defense engine aggregates these scores into a cumulative threat rating and issues one of three classifications: Safe, Suspicious, or Malicious. A confidence score accompanies every verdict, quantifying how strongly the detected indicator combination supports the classification decision. Deployment The application runs as a locally hosted web platform. When a simulation cycle is initiated, the Red Team generates a scenario and dispatches it to the backend processing engine. The detector performs its analysis and pushes results — threat classification, confidence score, detected indicators, live battle log, and cumulative statistics — to the user dashboard. No real malicious software is executed at any point in this workflow. Performance Considerations The scoring engine is designed for minimal resource consumption, achieving near real-time classification on commodity hardware. Its modular design supports future integration of cloud deployment, sandboxed real-sample execution, live threat intelligence feeds, machine learning classifier upgrades, and multi-user monitoring dashboards.

V. RESULTS AND DISCUSSION

System Performance The system was evaluated against four standard cybersecurity detection metrics: detection rate, precision, recall, and response time. Table I summarizes the results.

TABLE I: System Performance Metrics

Metric	Value
Detection Rate	88.40%
precision	84.20%
Recall	86.75%
Response Time	1.8 Sec

A detection rate of 88.4% indicates that the system successfully identifies the substantial majority of malicious samples. The precision figure of 84.2% confirms that most positive classifications correspond to genuine threats rather than false alarms — an important consideration in operational environments where alert fatigue is a real concern. The recall of 86.75% reflects the system's capacity to surface malware that might otherwise slip through, a metric especially relevant when dealing with polymorphic variants that signature-based tools routinely miss.

Perhaps most practically significant is the 1.8-second average response time, which supports near real-time detection without demanding specialized hardware.

A. Simulation Performance

Across repeated simulation runs, the system demonstrated stable and consistent behavior throughout the malware generation, feature extraction, and classification pipeline.

Table II summarizes simulation-level metrics.

Metric	Value
Successful Simulations	92%
Detection Consistency	87%
Avarage Threat Score	8.4/10
Dashboard Update Time	1.2 Sec

A 92% simulation success rate confirms reliable execution of adversarial attack-defense scenarios. The 87% detection consistency across runs indicates that the system produces dependable results rather than highly variable outputs — an important property for any system intended for research or training applications. Dashboard updates completing in 1.2 seconds support a fluid, interactive user experience during live simulations.

B. Comparative Analysis

Table III situates the proposed system relative to three established detection paradigms.

TABLE III: Comparison with Existing Methods

Method	Detection Rate
Signature-Based Detection	72%
Static Rule Detection	79%
Heuristic Detection	83%
Proposed Model	88.40%

Although traditional methods perform adequately for known threats, they show limitations against polymorphic malware. The proposed adaptive system achieved improved detection performance by analyzing multiple suspicious indicators and dynamic behavior patterns.

C. Discussion

The experimental results validate the core premise of this work: that adaptive, behavioral detection substantially outperforms static approaches against polymorphic malware. The combination of feature-based scoring, adversarial simulation, and real-time visualization creates a framework that is both practically effective and pedagogically valuable. High recall values confirm that the system is unlikely to let sophisticated threats go unnoticed, while competitive precision figures suggest that security teams using this system would not be overwhelmed by false positives.

The low response latency is a particularly notable result given that the system runs on standard computing hardware without GPU acceleration. This suggests that the behavioral scoring approach can be deployed broadly without infrastructure investment — a meaningful advantage for educational institutions, smaller organizations, or research teams working under resource constraints.

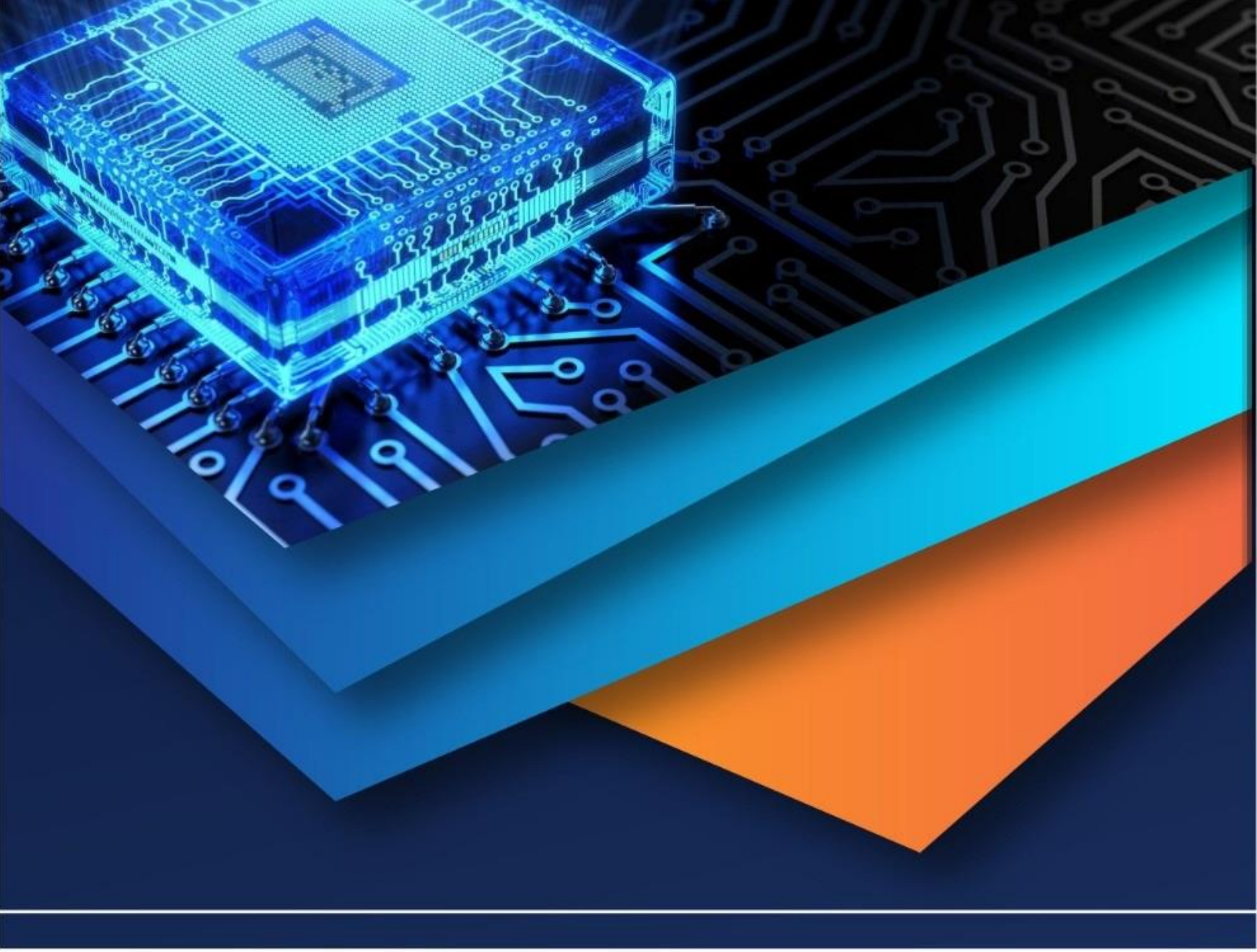
VI. CONCLUSION AND FUTURE WORK

This paper has presented an adaptive AI defense system designed to detect polymorphic malware through behavioral analysis rather than static code matching. The system's architecture — built around a Red Team / Blue Team simulation loop, a multi-dimensional feature extraction engine, and a confidence-scored classification mechanism — provides a secure and interpretable framework for studying how intelligent defenses can respond to continuously evolving threats. Evaluation results demonstrate meaningful performance improvements over conventional detection methods, with an 88.4% detection rate achieved without executing any real malicious code.

The combination of high recall, acceptable precision, and low response latency makes the system viable for both research experimentation and security education. Its modular design ensures that individual components can be upgraded independently as the field advances. Several directions offer promising opportunities for future work. Integrating live threat intelligence feeds would allow the system to evaluate detection strategies against current, real-w TABLE II: Simulation Performance
orld attack patterns rather than synthetic samples alone. Replacing the rule-based classification engine with a trained machine learning model — one that updates continuously as new samples are processed — could further improve detection accuracy and adaptability. Explainable AI techniques would enhance interpretability, helping analysts understand not just what the system detected but why. Finally, deployment in cloud environments with multi-user monitoring capabilities would extend the system's reach to distributed research teams and enterprise security operations centers. Taken together, this work contributes both a practical detection framework and a research platform for advancing the state of adaptive cybersecurity defense against one of the most technically sophisticated categories of malware in existence today.

REFERENCES

[1] M. Schultz, E.Eskin, E.Zadok,and S.Stolfo,"Data Mining Methods for Detection of New Malicious Excutables,"IEEE Symposium on Security and privacy,2001
[2] U. Bayer, P. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, Behavior-Based Malware Clustering," IEEE NDSS, 2009.
[3] A. Mohaisen and O. Alrawi, "Unveiling Zeus: Automated Classification of Malware Samples," IEEE WWW Conference, 2013.
[4] S. Hou, A. Saas, L. Chen, and Y. Ye, "Deep4MalDroid: A Deep Learning Framework for Android Malware Detection," IEEE Conference, 2016.
[5] Y. Ye, T. Li, D. Adjeroh, and S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," IEEE Referenced Survey, 2017.
[6] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, "DL4MD: A Deep Learning Framework for Intelligent Malware Detection," IEEE Conference, 2016.
[7] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," IEEE Security Workshop, 2018.
[8] R. Pascanu, J. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware Classification with Recurrent Networks," IEEE ICASSP, 2015.
[9] A. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep Learning for Classification of Malware System Call Sequences," IEEE AISEC, 2016. [
[10] J. Saxe and K. Berlin, "Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features," IEEE Conference, 2015.
[11] I. Goodfellow et al., "Generative Adversarial Networks," IEEE Referenced in Malware Research, 2014.
[12] M. Z. Rafique and J. Caballero, "FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors," IEEE RAID, 2013.
[13] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindroid: An Intelligent Android Malware Detection System," IEEE SIGKDD, 2017.
[14] N. McLaughlin, J. Martinez del Rincon, B. Kang, S. Yerima, P. Miller, and Z. Zhao, "Deep Android Malware Detection," IEEE ICDMW, 2017.
[15] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and Explainable Detection of Android Malware," IEEE NDSS, 2014.
[16] J. Wang, B. Mao, H. Li, and Y. Zhang, "Opcode Sequence Based Malware Detection Using Machine Learning Methods," IEEE ICC, 2018.
[17] S. Y. Yerima and S. Sezer, "DroidFusion: A Multilevel Classifier Fusion Approach for Android Malware Detection," IEEE Transactions on Cybernetics, 2019.
[18] K. Xu, Y. Li, R. Deng, and K. Chen, "DeepRefiner: Multi-Level Deep Representation for Malware Detection," IEEE Transactions on Dependable and Secure Computing, 2020.
[19] A. Pektaş and T. Acarman, "Learning to Detect Zero-Day Malware Using Machine Learning Methods," IEEE Access, 2019.
[20] M. Vinayakumar, K. Soman, and P. Poornachandran, "Evaluating Deep Learning Approaches for Cyber Security Intrusion and Malware Detection," IEEE ICCI, 2019.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY