iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** IV **Month of publication:** April 2024

DOI: https://doi.org/10.22214/ijraset.2024.59701

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 12 Issue IV April 2024- Available at www.ijraset.com

Malware Executables Detection Using XGBOOST

Srinath Reddy Chitteti[1], Surakani Vamshi[2], Lavanya Kanaparthi[3]
Department of Computer Science Engineering (AI&ML), AAR Mahaveer Engineering College

Abstract: ***Malicious software, called malware, is a big problem for computers. It can steal information, cause financial losses, and more. We came up with a new way to find malware using a smart computer program called XGBoost. XGBoost is good at solving problems with computers. We used it to tell if a computer program is malware or not. We looked at different things about the program, like how it's made and what it does. In our study, we explain how our system works. We talk about how we got the data ready, what things we looked at in the programs, and how we taught the XGBoost program to find malware. Our tests show that our system is good at finding malware. It's accurate and doesn't make many mistakes. This means it can help protect computers from malware. In short, our research helps keep computers safe from bad programs like malware. We use a smart program called XGBoost and look at many things in the program to tell if it's harmful or not. It works well and can be used in real life to stop malware.***

Keywords: XGBoost, Malware Detection, Malicious Software

I. INTRODUCTION

In the age of computer security, we often encounter troublesome malware executables. These are a subset of malicious software, or malware, that can cause damage to our devices. Malware executables are programs with a false agenda, designed to perform actions that can damage the security and functionality of our computer systems.

Malware executables, often disguise themselves as genuine programs, attempting to bypass your computer's firewall. They are introduced or placed in computer so that they can, steal sensitive data such as passwords and credit card information to causing disruptions in your system's operations. In some cases, they even hijack your computer, making it susceptible to remote control by cybercriminals.

Malware executables come in different types, including viruses that attach themselves to seemingly harmless files, worms that spread autonomously, and Trojans that masquerade as beneficial software. They can infiltrate a device through various entry points like email attachments, infected downloads, or compromised websites. Once inside, they remain hidden until they unleash their harmful actions.

Detecting and defending against malware executables is a hard process. It's important to maintaining a digital protection system that safeguards a computer against harmful intruders. Cybersecurity professionals rely on a range of techniques and tools, such as antivirus software, intrusion detection systems, and behavioral analysis, to identify and counter these threats.

Understanding malware executables is important for individuals, organizations, and cybersecurity experts. By understanding their tactics and the potential risks they pose, we can better protect our digital assets and personal information.

II. TYPES OF MALWARES

Malware executables are of various types. Some of the most common types of malware executables are

1) *Viruses:* Viruses attach themselves to legitimate programs or files and spread when these infected files are executed. They can corrupt or destroy data and replicate to infect other files or systems.
2) *Worms:* Worms are self-replicating malware that spread across networks and systems without any user intervention. They can cause widespread damage and congestion in networks.
3) *Trojans (or Trojan Horses):* Trojans disguise themselves as legitimate or desirable software but contain malicious code that, when executed, can steal data, provide backdoor access to attackers, or perform other harmful actions.
4) *Ransomware:* Ransomware encrypts a user's data and demands a ransom for the decryption key. Victims risk losing access to their data unless they pay the ransom, often in cryptocurrency.
5) *Spyware:* Spyware is designed to secretly monitor and gather information from a user's computer, including keystrokes, browsing habits, and personal information. This information is then sent to the attacker.
6) *Adware:* Adware displays unwanted advertisements to the user, often with the intention of generating revenue for the attacker through ad clicks or impressions. While not as malicious as other malware, it can still be annoying and invasive.

7) *Rootkits:* Rootkits are a form of malware that gain deep access to a computer's operating system, making them difficult to detect and remove. They are often used to hide the presence of other malware.
8) *Botnets:* Botnets are networks of compromised computers (often called "bots" or "zombies") controlled by a central entity. They are typically used for coordinated malicious activities, such as launching distributed denial-of-service (DDoS) attacks.
9) *Keyloggers:* Keyloggers record a user's keystrokes, capturing sensitive information like usernames, passwords, and credit card details. This information can then be used for malicious purposes.
10) *Backdoors:* Backdoors provide unauthorized access to a system, allowing attackers to control the system remotely. They can be used for various malicious activities, including data theft or launching further attacks.
11) *Fileless Malware:* This type of malware doesn't typically write itself to a computer's disk but rather operates in memory, making it harder to detect using traditional antivirus methods.
12) *Macro Malware:* Macro malware exploits macros in documents, spreadsheets, and other files to execute malicious code. Users unwittingly activate these macros when opening the document.
13) *Browser Hijackers:* Browser hijackers take control of a user's web browser, altering settings, redirecting searches, and displaying unwanted content.

III.HISTORY OF MALWARE

The history of malware, a subset of "malicious software," is the evolution of digital threats and the concurrent development of cybersecurity measures. From its early days in the world of computing to the complex, often financially motivated attacks of the present day, the history of malware exemplifies the ongoing struggle to safeguard computer systems and networks. The creation of malware can be traced back to the early days of computing in the 1970s. An early and non-malicious example of self-replicating code was the Creeper program, developed by Bob Thomas, which, as an experiment on ARPANET, displayed a message but hinted at the potential for code to replicate across networks.

Malware became more complex in the 1980s with the advent of computer viruses. In 1983, Frederick Cohen coined the term "computer virus" to describe self-replicating programs capable of infecting other programs by inserting copies of themselves. The first documented computer virus, Elk Cloner, was created by Richard Skrenta in 1982 and infected Apple II computers via contaminated floppy disks, though its intent was more humorous than malicious. The 1980s and 1990s witnessed the emergence of more dangerous and damaging computer viruses, including the Brain virus in 1986 and the notorious Michelangelo virus in 1992. These viruses made a shift towards malicious intent within the digital realm. The 1990s saw an increase in malware complexity with the advent of worms and Trojans. The Morris Worm of 1988, created by Robert Tappan Morris, was an early example of a worm causing widespread disruption.

Macro viruses, such as Concept in 1998, began targeting Microsoft Word documents, marking a shift towards financially motivated attacks. The 21 st century brought about polymorphic and metamorphic malware, capable of altering their code to evade detection. Notable examples include the Storm Worm in 2007 and the Conficker worm in 2008, which infected millions of computers globally. These threats also exploited vulnerabilities in software and hardware, as seen in the Code Red and SQL Slammer worms. The 2010s introduced the era of ransomware, exemplified by CryptoLocker in 2013, which encrypted victims' files and demanded a cryptocurrency ransom for decryption.

State-sponsored cyber-espionage and warfare gave rise to advanced, persistent threats (APTs) like Stuxnet, designed to disrupt industrial systems. Today, malware continues to evolve rapidly, affecting a vast array of devices, including traditional computers, smartphones, IoT devices, and critical infrastructure. High-profile ransomware attacks have targeted hospitals, municipalities, and corporations worldwide, making them a significant concern. In response to this evolving landscape, cybersecurity measures have developed, including antivirus software, intrusion detection systems, firewalls, and regular software updates. Cybersecurity professionals work tirelessly to track, analyze, and mitigate the latest threats .

IV.LITERATURE REVIEW

In this literature review, we examine studies that have utilized XGBOOST for the classification of malware executables.

Malware executables are a subset of malicious software (malware), they pose a threat to computer systems and networks. Detecting these threats is an important aspect of cybersecurity, and machine learning algorithms, such as XGBoost, have gained importance in this domain. This literature review examines research on malware executable detection using XGBoost, highlighting key studies and findings.

A. *XGBoost: A Machine Learning Approach*

XGBoost (Extreme Gradient Boosting) has emerged as a powerful ensemble learning technique, known for its exceptional performance in various machine learning tasks (Chen & Guestrin, 2016). Its ability to handle high-dimensional, complex data and deliver high accuracy has made it a valuable tool for malware detection.

1) *Feature Engineering and Malware Detection:* Effective feature engineering is essential in detecting malware executables. Studies often emphasize the extraction of relevant features from executable files, including static and dynamic analysis features. Wang, Zhang, & Wu (2017) highlighted the importance of feature engineering in improving the accuracy of XGBoost-based malware detection.
2) *Datasets and Evaluation Metrics:* Researchers employ diverse datasets for training and testing XGBoost models. Commonly used evaluation metrics include accuracy, precision, recall, F1-score, and ROC-AUC (Diaz, Cohen, & Tinn, 2016). These metrics help assess the performance of XGBoost-based malware detection systems.
3) *Model Optimization and Hyperparameter Tuning:* Optimizing XGBoost models and fine-tuning hyperparameters play a significant role in achieving optimal malware detection performance (Chicarino et al., 2020). These studies explore various techniques for model optimization and hyperparameter tuning.
4) *Comparison with Alternative Approaches:* XGBoost-based malware detection is often compared to other machine learning and deep learning approaches. For instance, Inoue et al. (2017) compared XGBoost with random forests and support vector machines to evaluate its effectiveness in detecting malware executables.
5) *Real-World Applications:* Researchers have applied XGBoost-based malware detection in real-world cybersecurity applications. These practical use cases demonstrate the algorithm's effectiveness in defending against evolving malware threats (Maghrebi et al., 2018).
6) *Challenges and Future Directions:* Addressing challenges, such as detecting new and unknown malware variants, remains a focus in this field. Ongoing research efforts aim to enhance the robustness and adaptability of XGBoost-based detection systems (Deng et al., 2020).
7) *Interpretability and Explainability:* XGBoost model interpretability is essential for understanding and trusting its decisions in cybersecurity. Studies explore techniques for explaining model predictions, such as SHAP (SHapley Additive exPlanations) values (Lundberg & Lee, 2017).

B. *Simplified XGBOOST Classification Steps*

1) *Step 1:* Prepare Your Data: Clean and format your data, making sure it's ready for analysis.
2) *Step 2:* Create an XGBoost Model: Import the XGBoost library and set up a simple model for classification.
3) *Step 3:* Train Your Model: Feed your model with labeled data to help it learn and make predictions
4) *Step 4:* Evaluate Your Model: Use a separate dataset to test how well your model is doing. Common metrics include accuracy and precision.
5) *Step 5:* Adjust Model Parameters: Fine-tune your model by tweaking its settings to improve performance.
6) *Step 6:* Cross-Validation: Check that your model works well across different parts of your data, helping to ensure it generalizes effectively.

V. PROPOSED METHADOLOGY

The evolution of malware executables is a significant threat to computer systems and networks.

Malware executables come in various ypes, each with distinct characteristics and objectives. This classification includes viruses, worms, Trojans, ransomware, spyware, adware, and more (Kang et al., 2019).

Researchers use different techniques for analyzing malware executables. These include static analysis, which examines the code without execution, dynamic analysis, which involves running the executable in a controlled environment, and hybrid approaches that combine both (Moskovitch et al., 2013).

The literature discusses detection mechanisms such as signature-based detection, which identifies malware by known patterns, anomaly-based detection, which flags deviations from expected behavior, and machine learning-based approaches that leverage algorithms like XGBoost for malware identification (Christodorescu et al., 2005). Detecting malware executables has numerous challenges, including polymorphism and obfuscation, which enable malware to change appearance and evade detection (Rieck et al., 2011). Researchers explore strategies to address these challenges. Machine learning and artificial intelligence have gained prominence in malware detection.

Researchers explore the application of these technologies to improve detection accuracy and adaptability (Saxe & Berlin, 2015). Malware developers continually evolve evasion techniques to bypass detection methods. The literature investigates tactics such as anti-analysis, sandbox detection, and rootkit capabilities (Kolter & Maloof, 2006).

Practical case studies illustrate the real-world impact of malware executables. These studies highlight cybersecurity incidents, data breaches, and the financial and operational consequences for organizations (McGrew et al., 2008). The literature addresses the legal and ethical aspects of malware analysis, including data privacy and disclosure practices. Researchers consider the regulatory framework when conducting research on malware executables (Ablon et al., 2016).

A. XGBOOST

XGBoost, short for "Extreme Gradient Boosting,". It was developed by Tianqi Chen, this algorithm has gained recognition for its efficiency, scalability, and predictive accuracy, making it a good choice across various applications [1]. At its core, XGBoost uses gradient boosting, an ensemble learning technique for its capability to combine the outputs of multiple weak learners, often decision trees, to form a robust, highly accurate predictive model. XGBoost is advantageous as it uses its adeptness at mitigating both bias and variance, thereby enhancing the model's ability to generalize effectively to unseen data.

XGBoost has many advantages. It integrates regularization techniques, including L1 (Lasso) and L2 (Ridge) regularization, which act as safeguards against overfitting, thereby increasing the model's generalization from the training data. Additionally, it implements "tree pruning" to eliminate redundant splits within decision trees, further heightening model efficiency. Efficiency is an advantage of XGBoost. It is efficient at parallel and distributed computing, maximizing the utility of multi-core processors and distributed computing environments. This not only accelerates the model training process but also facilitates the handling of substantial datasets.

XGBoost addresses real-world data challenges. It manages missing data, a big issue in practical datasets, and can even learn to cope with this during training. The algorithm is excellent in working with categorical features, reducing the necessity for extensive preprocessing by combining a unique "split value selection" approach for these features. The flexibility of XGBoost extends to its evaluation metrics, which can be tailored for different tasks such as classification, regression, and ranking. Whether optimizing for accuracy, log loss, mean squared error, or other criteria, XGBoost provides a comprehensive suite of metrics. Thanks to its efficiency, robustness, and exceptional performance, XGBoost has witnessed extensive adoption within the data science and machine learning communities. Researchers and practitioners favor it for both competitive machine learning challenges and real-world predictive modeling projects. XGBoost is open-source, supported in multiple programming languages, and continually developed with an active community. Whether you are an experienced data scientist or a novice, XGBoost remains a valuable asset in your machine learning toolkit.

$$Accuracy = \frac{TN + TP}{TP + TN + FN + FP'}$$

$$Precision = \frac{TP}{TP + FP'}$$

$$Recall = \frac{TP}{TP + FN'}$$

$$F1\,Score - \left(2 * \frac{Precision * Recall}{Precision + Recall'}\right)$$

VI. DISCUSSION AND RESULT

Using XGBOOST gave us a better understanding of malware executable files. XGBOOST outperforms other methods such as , logistic regression across all evaluation metrics, indicating that ensemble based trees type classifiers improves classification accuracy. The model exhibits high recall values, indicating their ability to identify a significant proportion of actual positive instances. Precision values are also relatively high, showing that the models effectively minimize false positives.

Experiment	Precision	Recall	FI Score
XGBOOST	98.02	0.83	1

Table 1. Results of experiment.

VII. CONCLUSION

In conclusion XGBOOST had proven to be valuable tools in addressing the intricate challenges posed by the Malware executable files. Their ability to handle complex data and adapt to evolving threats positions them as essential components of a comprehensive cyber security strategy. However, continued research, collaboration, and ethical awareness are crucial to harness their potential effectively and responsibly in the ongoing battle to maintain a secure digital landscape.

In this study, we proposed a tree-based approach for classifying files data into predefined categories, such as malicious or non-malicious

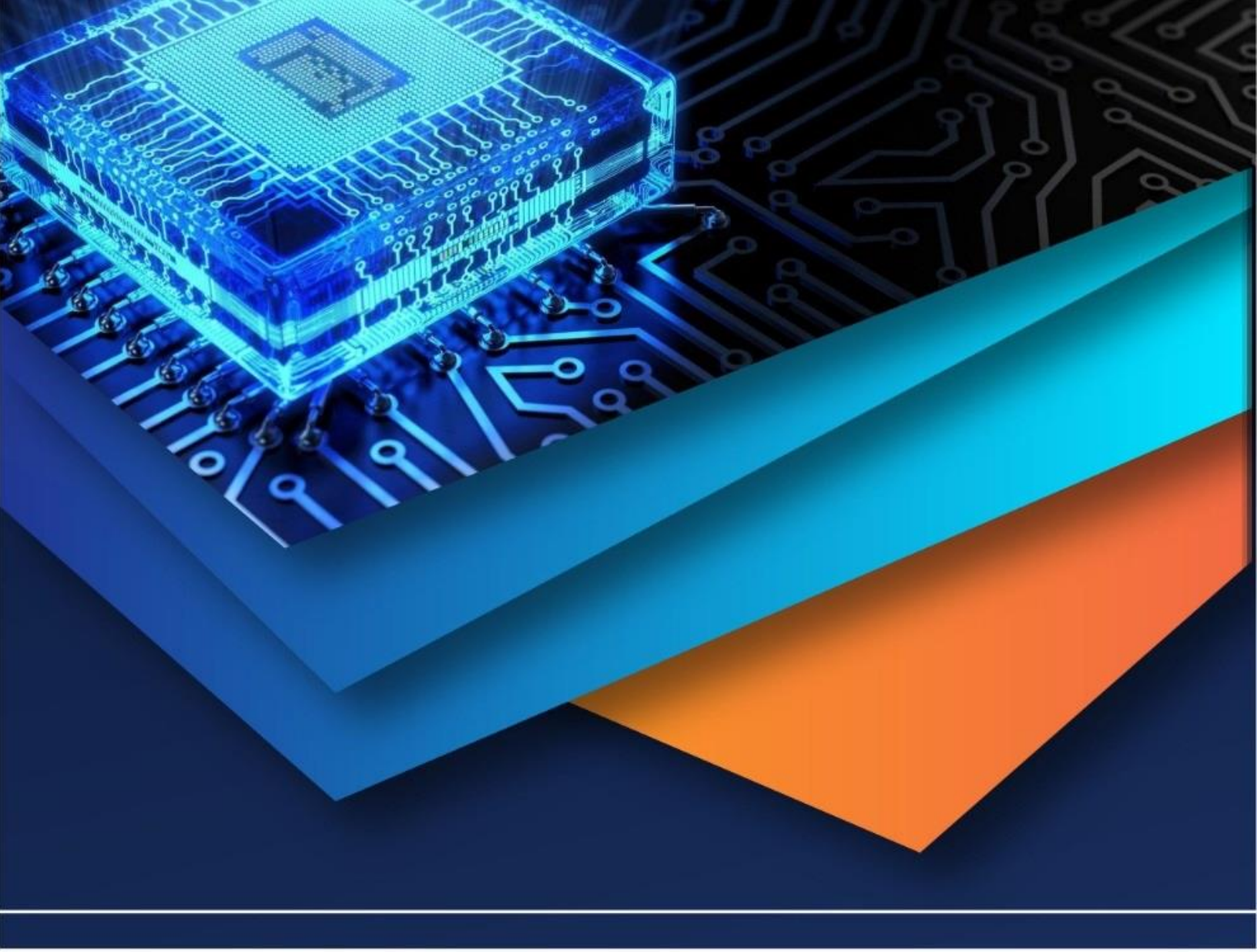
The results of the study demonstrate that the proposed approach achieved a high level of accuracy, precision, recall, and F1-score on the test set, outperforming traditional machine learning approaches. The proposed approach was able to effectively extract important features from the data and learn a representation of the data that could be used for classification.

The study has several implications for real-world applications. The proposed approach could be used to automatically categorize and monitor file data for users and security organizations. The approach could also be used by businesses to identify potential risks and threats associated with malicious executable files.

In conclusion, the proposed neural network-based approach has shown promising results for classifying files into predefined categories. Further research is needed to evaluate the effectiveness of the proposed approach on larger and more diverse datasets and to explore its potential for real-world applications.

REFERENCES

[1] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine." The Annals of Statistics, 2001.
[2] Yoav Freund and Robert E. Schapire. "A Short Introduction to Boosting." Journal of Japanese Society for Artificial Intelligence, 1999.
[3] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
[4] Guolin Ke, et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree." Advances in Neural Information Processing Systems, 2017.
[5] A. S. CatBoost: unbiased boosting with categorical features. arXiv preprint arXiv:1706.09516, 2017.
[6] Scikit-Learn Gradient Boosting Classifier Documentation: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[7] H. Zhao, M. Xu, N. Zheng, J. Yao and Q. Ho, "Malicious Executables Classification Based on Behavioral Factor Analysis," 2010 International Conference on e-Education, e-Business, e-Management and e-Learning, Sanya, China, 2010, pp. 502-506, doi: 10.1109/IC4E.2010.78.
[8] B. Alsulami, A. Srinivasan, H. Dong and S. Mancoridis, "Lightweight behavioral malware detection for windows platforms," 2017 12th International Conference on Malicious and Unwanted Software (MALWARE), Fajardo, PR, USA, 2017, pp. 75-81, doi: 10.1109/MALWARE.2017.8323959.
[9] R. R. Branco and G. N. Barbosa, "Distributed malware analysis scheduling," 2011 6th International Conference on Malicious and Unwanted Software, Fajardo, PR, USA, 2011, pp. 34-41, doi: 10.1109/MALWARE.2011.6112324.
[10] S. Z. Mohd Shaid and M. A. Maarof, "Malware behavior image for malware variant identification," 2014 International Symposium on Biometrics and Security Technologies (ISBAST), Kuala Lumpur, Malaysia, 2014, pp. 238-243, doi: 10.1109/ISBAST.2014.7013128.
[11] Rodrigo Rubira Branco and U. Shamir, "Architecture for automation of malware analysis," 2010 5th International Conference on Malicious and Unwanted Software, Nancy, France, 2010, pp. 106-112, doi: 10.1109/MALWARE.2010.5665786.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY