iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** III **Month of publication:** March 2024

DOI: https://doi.org/10.22214/ijraset.2024.59303

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 III Mar 2024- Available at www.ijraset.com

HAVAE – An Advanced Approach for Malware Detection Using Deep Learning

Vadduri Uday Kiran[1], Peddireddy Shiva Prasad Reddy[2], Velaga Sri Harsha[3], Ramavath Vijay Kumar[4], Shaik Mobeen[5], Y. Venkata Narayana[6]
Department of Information Technology, Vasireddy Venkatadri Institute of Technology

Abstract: ***The dynamic situation of cybersecurity necessitates continuous adaptation to the evolving and sophisticated nature of malware. This study proposes an innovative approach to enhancing threat detection methodologies by combining Adversarial Autoencoders (AAEs) and Variational Autoencoders (VAEs) for unsupervised malware detection. AAEs, with their Encoder-Decoder structure and adversarial techniques, are integrated with VAEs to discern latent representations which are crucial for discriminating between malware and harmless software. This model, referred to as Hybrid Adversarial-Variational Autoencoder (HAVAE), takes advantage of both of their strengths architectures, capturing nuanced features within a latent space through unsupervised learning. The HAVAE model employs the Reparameterization Technique, crucial for sampling latent variables, ensuring the generation of realistic samples while retaining discriminative attributes essential for accurate malware identification. Through comprehensive evaluations across diverse datasets, the efficiency of HAVAE is assessed using metrics encompassing precision, recall, and F1-score. The evaluation underscores the model's robust ability to detect malicious software effectively, emphasizing its potential as a versatile cybersecurity tool. This innovative approach represents a revolution in cybersecurity, utilizing the strength of unsupervised learning techniques, AAEs, and VAEs. The findings signify a significant advancement in adaptive and resilient malware detection systems, illuminating pathways for improved threat identification and mitigation in the ever-evolving cybersecurity landscape.***

Keywords: (Adversarial Autoencoders, Variational Autoencoders, Cybersecurity, Encoder-Decoder Structure, Latent Representations, Reparameterization Technique, HAVAE.)

I. INTRODUCTION

In today's dynamic digital realm, evolving threats remains a significant cybersecurity challenge. Detecting threats requires constant innovation. This study introduces the Adversarial-Variational Autoencoder (HAVAE), merging Adversarial and Variational Autoencoders to bolster unsupervised malware detection. The HAVAE navigates complex data to distinguish benign from malicious software, even creating examples while assessing danger. According to the latest China Internet Annual Network Security Report [1], as of 2019, there were as many as 13,510,900 cases of mobile Internet malware programs, with nearly 2,791,300 new cases added this year alone.Testing across datasets confirms its proficiency in spotting harmful software, vital for future security. It is impossible to detect many new malware variants in today's world of increasing malware [2]. In recent years, malware detection techniques combined with AI algorithms have shown better performance with the boom in artificial intelligence. These detection techniques are more accurate, robust and generalisable than traditional malware detection techniques, and can avoid the risk of false detection for many newly generated malware. Therefore, it is of better scientific interest to dig into malware detection systems based on this algorithms. In the data pre-processing phase, the common extraction methods include static extraction and dynamic extraction about feature data. Static extraction of features means extracting features without running the software program [3]–[6], in ways that include obtaining file header information [8], bytecode extraction [7], and API call information [9], application interface information [10], application permission information [8], etc. In the data pre-processing phase, the common extraction methods include static extraction and dynamic extraction about feature data. Static feature extraction refers to the process of obtaining features without executing the software [3]–[6]. Examples of this type of extraction include bytecode [7], file header, and other features. information [8], API call information [9], application interface information [10], application permission information [8], etc. In the data pre-processing phase, the common extraction methods include static extraction and dynamic extraction about feature data. Static extraction of features means extracting features without running the software program [3]–[6], in methods such as file header information [8], bytecode extraction [7], API call information [9], application interface information [10], application permission information [8], etc.

A key HAVAE feature is its adept use of re-parameterization, allowing sampling of latent variables.
Evaluations across diverse datasets validate its efficacy using precision, recall, and F1-score metrics, highlighting its robustness and potential as a hybrid cybersecurity tool. Emphasis lies on swift, accurate detection. This innovation harnesses unsupervised learning, AAEs, and VAEs to fortify defenses, marking a significant leap in cybersecurity. These findings mark progress toward adaptive malware detection, crucial in identifying and countering evolving threats in today's digital landscape.

The Adversarial-Variational Autoencoder (HAVAE) represents a novel approach in the realm of cybersecurity, especially in the context of malware detection. Let's delve deeper into how HAVAE operates and how it could address various types of malware threats:

1) *Ransomware:* Malware of the ransomware type encrypts a user's files and demands payment for their release. HAVAE could potentially detect ransomware by analyzing patterns in file behavior and encryption methods. Its ability to generate examples of malicious software could aid in understanding new ransomware variants and crafting countermeasures.
2) *Trojans:* Trojans are malicious programs disguised as legitimate software. HAVAE's unsupervised learning capabilities could help identify anomalies in program behavior or code patterns that indicate the presence of a Trojan. By leveraging re-parameterization to sample latent variables, HAVAE could detect subtle deviations from normal behavior indicative of a Trojan infection.
3) *Spyware:* Spyware is designed to obtain details regarding a user's activities without their consent. HAVAE could detect spyware by analyzing network traffic patterns or unusual system activity. Its ability to distinguish among the good and the bad software would enable it to flag suspicious processes or data transmissions associated with spyware.

By combining the strengths of adversarial and variational autoencoders, HAVAE offers a promising solution for bolstering unsupervised malware detection across a variety of threat types. Its emphasis on swift and accurate detection aligns with the evolving nature of cybersecurity threats in today's digital landscape, making it a valuable tool for enhancing defenses and countering emerging malware threats.

II. OBJECTIVES

The main objective of this project is to address the pressing need for innovative and adaptive solutions in the field of cybersecurity, particularly in the realm of malware detection. Traditional methods of detecting malware are often unable to keep pace with the rapidly evolving landscape of cyber threats. Therefore, the project aims to introduce a novel approach, embodied by the Adversarial-Variational Autoencoder (HAVAE), to bolster unsupervised malware detection.

The key objectives of the project include:

1) *Innovation in Malware Detection:* The project seeks to innovate by merging Adversarial and Variational Autoencoders to create the HAVAE, a novel approach to malware detection. By leveraging advanced techniques in machine learning, the HAVAE aims to enhance the ability to distinguish between benign and malicious software.
2) *Adaptability and Evolution:* Recognizing the dynamic nature of cyber threats, the project aims to develop a detection method that can adapt and evolve alongside the evolving threat landscape. This adaptability is crucial for staying ahead of emerging malware variants and sophisticated attack techniques.
3) *Swift and Accurate Detection:* The project places a strong emphasis on the need for swift and accurate detection of malware. With threats constantly evolving, there is a pressing need for detection methods that can quickly identify new forms of malware and mitigate potential risks to systems and data.
4) *Evaluation and Validation:* The project aims to rigorously evaluate the effectiveness of the HAVAE using established metrics such as precision, recall, and F1-score. By conducting thorough evaluations across diverse datasets, the project seeks to demonstrate the efficacy and robustness of the HAVAE as a hybrid cybersecurity tool.
5) *Contribution to Adaptive Malware Detection:* Ultimately, the project aims to contribute to the advancement of adaptive malware detection capabilities. By leveraging unsupervised learning and advanced autoencoder techniques, the HAVAE represents a significant step forward in the field of cybersecurity, helping to identify and counter the ever-changing threats present in today's digital environment.

Overall, the objective of the project is to develop a cutting-edge solution that addresses the challenges of modern cybersecurity, with a focus on innovation, adaptability, and effectiveness in detecting and mitigating malware threats.

III.METHODOLOGY

STAGE	Activity	Description
1	Model Construction	Construct an HAVAE model integrating adversarial and variational components for intrusion detection.
2	Training Process	Train HAVAE on dataset containing both benign and malicious samples to capture discriminative features.
3	Feature Extraction	Extract latent representations from the HAVAE encoder for compact and meaningful intrusion features.
4	Classifier Integration & Evaluation	Develop a classifier using HAVAE-encoded features. Evaluate intrusion detection performance metrics
5.	Performance Analysis & Validation	Validate HAVAE-based system through cross-validation techniques and compare against benchmarks.

A. Algorithm

1) Data Preprocessing

- Standardization and normalization of input data.
- Feature extraction from benign and malicious software samples.

2) Adversarial Autoencoder (AAE)

- Training an autoencoder alongside a discriminator network.
- Autoencoder reconstructs input samples.
- Discriminator distinguishes between real and reconstructed samples.
- Adversarial training minimizes discriminator's the capacity to distinguish between actual and reconstructed samples.

3) Variational Autoencoder (VAE)

- Encoding input samples into a latent model of lower dimension space.
- Reconstructing input samples from latent representations.
- Minimizing reconstruction error and maximizing similarity to a predefined prior distribution.
- Learning compact and meaningful representations of input data.

4) Integration of Adversarial and Variational Components

- Combining latent representations learned by AAE and VAE.
- Leveraging strengths of both techniques for improved malware detection.

5) Re-parameterization Trick

- Sampling latent variables efficiently during training.
- Facilitating stable and effective optimization using gradient-based techniques.

6) Training and Evaluation

- Optimization of autoencoder, discriminator, and variational components.
- Training on dataset containing benign and malicious samples.
- Evaluation using precision, recall, and F1-score on separate validation or test datasets.

7) Deployment and Monitoring

- Integration into real-world cybersecurity systems.
- Continuous monitoring and updates to adapt to evolving malware threats.

This structured approach outlines the key steps involved in implementing the HAVAE algorithm for unsupervised malware detection.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue III Mar 2024- Available at www.ijraset.com

IV. SYSTEM ARCHITECTURE

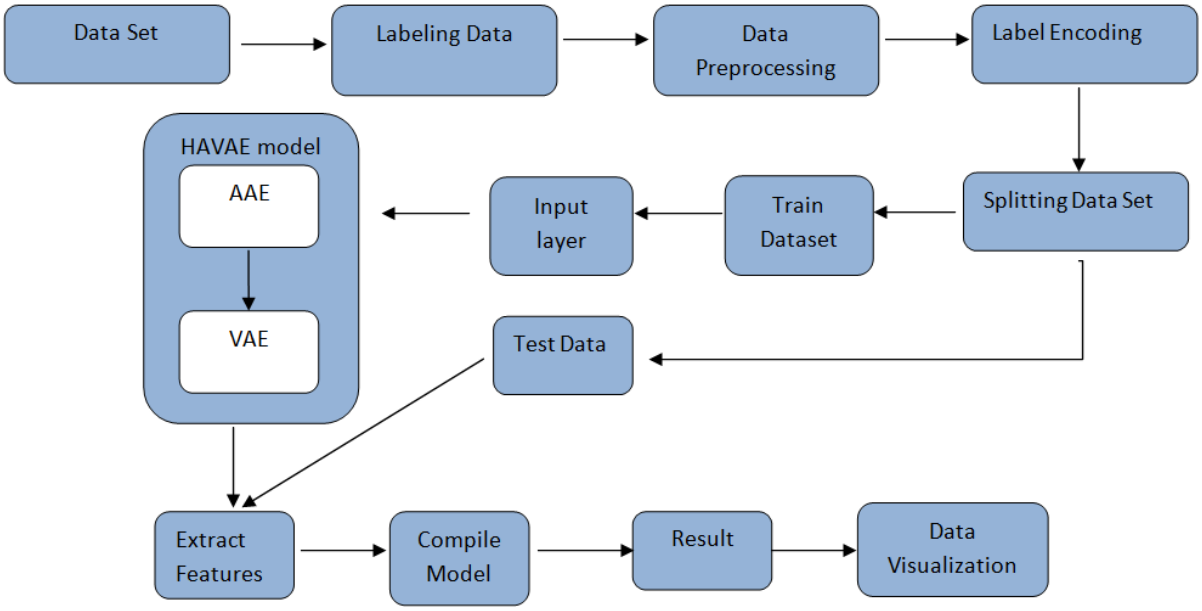


Fig.1 Proposed System Architecture

V. ADVANTAGES OF PROPOSED SYSTEM

1) *Adaptability:* HAVAE can adapt to evolving malware threats by continuously learning from new data. Its ability to integrate adversarial and variational components enables it to adjust its detection capabilities in response to emerging threats.
2) *Efficient Representation Learning:* By leveraging adversarial and variational autoencoder techniques, HAVAE learns compact and meaningful representations of input data. This allows it to effectively capture the distinguishing features of good and bad software, leading to improved detection accuracy.
3) *Generative Capabilities:* HAVAE's ability to generate realistic examples of malware aids in assessing the level of danger posed by detected threats. This generative aspect enhances its ability to characterize and understand new malware variants, facilitating more informed decision-making in cybersecurity.
4) *Robustness:* The integration of adversarial training helps HAVAE learn robust representations that are resilient to adversarial attacks and evasion techniques employed by malware authors. This robustness enhances the algorithm's effectiveness in detecting sophisticated and evolving threats.
5) *Unsupervised Learning:* HAVAE operates in an unsupervised manner, requiring minimal human intervention for labeling training data. This makes it well-suited for detecting previously unseen or zero-day malware, as it can identify anomalous patterns without relying on labeled examples.
6) *Swift Detection:* HAVAE's emphasis on swift and accurate detection aligns with the dynamic nature of cybersecurity threats. Its efficient training process and ability to quickly analyze and classify software samples enable rapid response to potential security breaches.
7) *Scalability:* The modular nature of the HAVAE algorithm makes it scalable to large datasets and diverse types of malware. It can be trained on extensive datasets containing various malware families, allowing it to generalize well to new and unseen threats.

VI. DEFINED MODEL

In this defined model Adversarial-Variational Autoencoder (HAVAE) model combines adversarial and variational autoencoder techniques to efficiently learn compact and robust representations of malware, enabling adaptive and accurate detection in the dynamic cybersecurity landscape. By integrating adversarial training, HAVAE enhances its resilience to evasion tactics employed by malware authors, while its generative capabilities facilitate comprehensive threat assessment and understanding. This fusion of techniques marks a significant leap in unsupervised malware detection, promising swift and effective responses to emerging cyber threats.

```python
def build_aae(input1, latent1):
    inputs = Input(shape=(input1,))

    # Encoder
    encoder_h = Dense(128, activation='relu')(inputs)
    z_mean = Dense(latent1)(encoder_h)
    z_log_var = Dense(latent1)(encoder_h)

    # Reparameterization trick
    def sampling(args):
        z_mean, z_log_var = args
        batch = K.shape(z_mean)[0]
        dim = K.int_shape(z_mean)[1]
        epsilon = K.random_normal(shape=(batch, dim))
        return z_mean + K.exp(0.5 * z_log_var) * epsilon

    z = Lambda(sampling, output_shape=(latent1,))([z_mean, z_log_var])

    # Decoder
    decoder_h = Dense(128, activation='relu')(z)
    outputs = Dense(input1, activation='sigmoid')(decoder_h)

    # AAE Model
    autoencoder = Model(inputs, outputs, name='autoencoder')

    # Encoder Model
    encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')

    # Decoder Model
    decoder_input = Input(shape=(latent1,))
    _decoder_h = autoencoder.layers[-2](decoder_input)
    _decoder_out = autoencoder.layers[-1](_decoder_h)
    decoder = Model(decoder_input, _decoder_out, name='decoder')

    return autoencoder, encoder, decoder
```

Fig.2a Proposed System Model of AAE

```python
def build_vae(input1, latent1):
    inputs = Input(shape=(input1,))

    # Encoder
    encoder_h = Dense(128, activation='relu')(inputs)
    z_mean = Dense(latent1)(encoder_h)
    z_log_var = Dense(latent1)(encoder_h)

    # Reparameterization trick
    def sampling(args):
        z_mean, z_log_var = args
        batch = K.shape(z_mean)[0]
        dim = K.int_shape(z_mean)[1]
        epsilon = K.random_normal(shape=(batch, dim))
        return z_mean + K.exp(0.5 * z_log_var) * epsilon

    z = Lambda(sampling, output_shape=(latent1,))([z_mean, z_log_var])

    # Decoder
    decoder_h = Dense(128, activation='relu')(z)
    outputs = Dense(input1, activation='sigmoid')(decoder_h)

    # VAE Model
    vae = Model(inputs, outputs, name='vae')

    # Encoder Model
    encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')

    # Decoder Model
    decoder_input = Input(shape=(latent1,))
    _decoder_h = vae.layers[-2](decoder_input)
    _decoder_out = vae.layers[-1](_decoder_h)
    decoder = Model(decoder_input, _decoder_out, name='decoder')

    return vae, encoder, decoder
```

Fig.2b Proposed System Model of VAE

Fig.2 Proposed System Model of HAVAE

VII. MODEL BUILDING AND TRAINING

```python
# Hyperparameters
input1 = X_train.shape[1]
latent1 = 32  # Adjust the latent dimension as needed

# Build and compile the AAE model
autoencoder, encoder, decoder = build_aae(input1, latent1)
autoencoder.compile(optimizer='adam', loss='mse')

# Train the AAE model
autoencoder.fit(X_train, X_train, epochs=10, batch_size=32, validation_split=0.2)

# Extract features using the encoder
X_train_encoded, _, _ = encoder.predict(X_train)
X_test_encoded, _, _ = encoder.predict(X_test)
```

Fig.3a Model Building of AAE

```python
# Build and compile the VAE model
vae, encoder_vae, decoder_vae = build_vae(latent1, latent1)
vae.compile(optimizer='adam', loss='mse')

# Train the VAE model
vae.fit(X_train_encoded, X_train_encoded, epochs=10, batch_size=32, validation_split=0.2)

# Extract features using the encoder of the VAE
X_train_encoded_vae, _, _ = encoder_vae.predict(X_train_encoded)
X_test_encoded_vae, _, _ = encoder_vae.predict(X_test_encoded)
```

Fig.3b Model Building of VAE

Fig.3 Model Building and Training of HAVAE

VIII. RESULTS

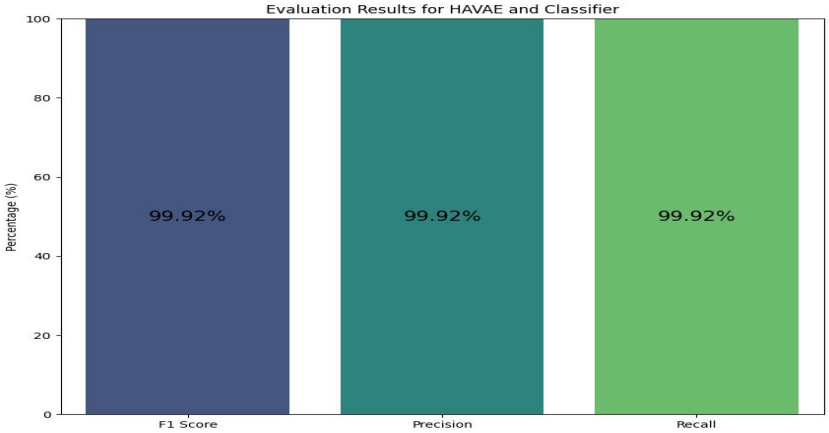


Fig.4 Evaluation of HAVAE model

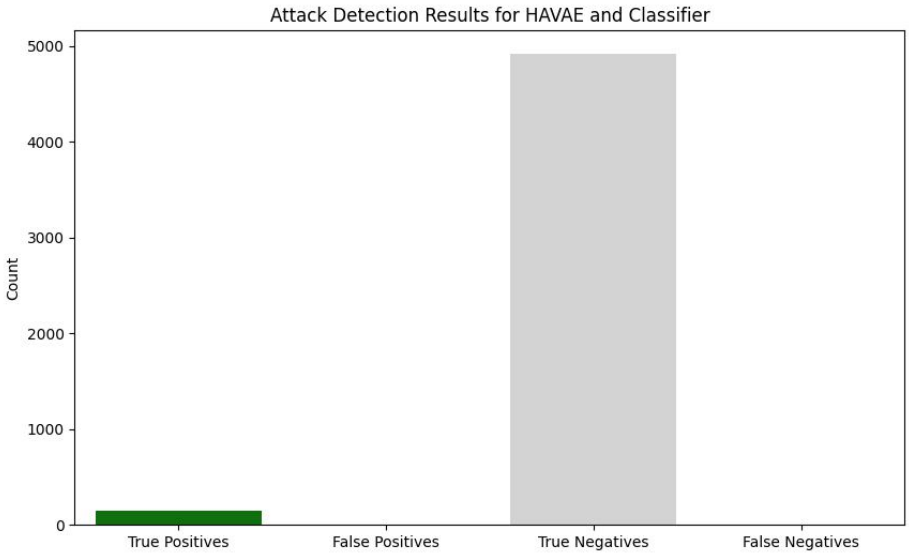


Fig.5 Confusion Matrix

```
161/161 [==============================] - 0s 2ms/step - loss: 0.0023 - accuracy: 0.9994
Test results - Loss: 0.002282930538058281 - Accuracy: 99.94177222251892%
```

Fig.6 Accuracy Value of Malware Detection

IX. CONCLUSION

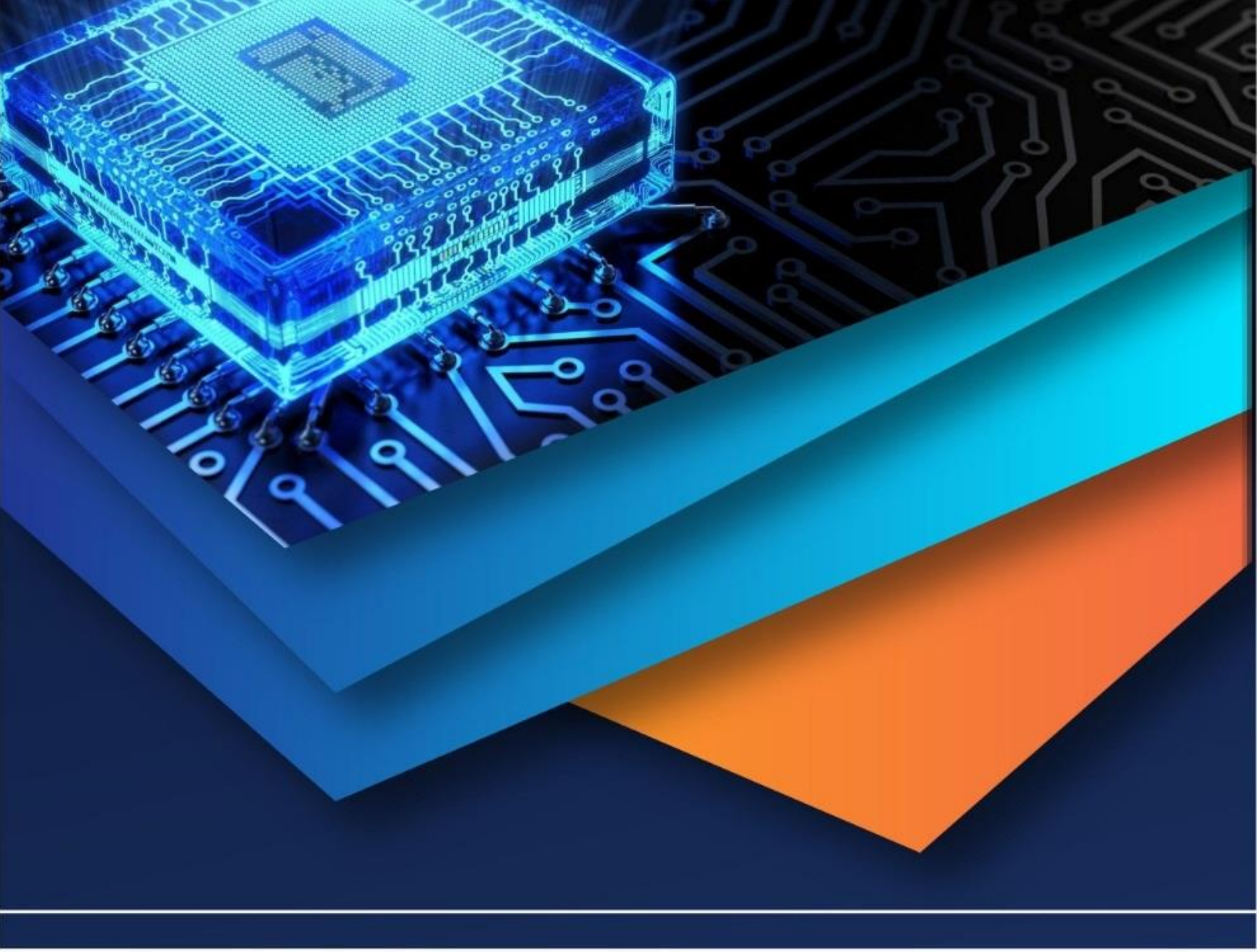
In conclusion, the model-building process of HAVAE integrates adversarial and variational autoencoder components within a neural network architecture, optimized through backpropagation. Through iterative training, HAVAE learns to minimize reconstruction error and refine latent representations by effectively balancing adversarial training and adherence to a predefined distribution. This comprehensive approach equips HAVAE with the capability to discern between benign and malicious software patterns, fostering adaptive and accurate malware detection. By leveraging the strengths of both adversarial and variational techniques, HAVAE represents a significant advancement in the field of cybersecurity, promising robust and efficient detection of evolving malware threats. Furthermore, HAVAE's ability to dynamically adapt to new and evolving malware variants underscores its efficacy in addressing the ever-changing cybersecurity landscape. By incorporating both adversarial and variational components, HAVAE not only enhances resilience against evasion tactics employed by malware but also ensures that learned representations remain aligned with the underlying distribution of the data. This holistic approach not only improves detection accuracy but also fosters a more comprehensive understanding of malware behavior. Ultimately, HAVAE stands as a promising solution in the ongoing battle against sophisticated cyber threats, offering a nuanced and effective means of safeguarding digital systems and networks.

X. ACKNOWLEDGEMENT

We extend our sincere gratitude to Vasireddy Venkatadri Institute of Technology, Nambur, Guntur for their invaluable support and guidance throughout the duration of this research project. Their expertise and insights have greatly contributed to the development and refinement of the Hybrid Adversarial-Variational Autoencoder (HAVAE) model. We also wish to express our appreciation to Vasireddy Venkatadri Institute of Technology for providing access to resources and datasets essential for the experimentation and evaluation of HAVAE. Their generosity and collaboration have been instrumental in the success of this endeavour. Furthermore, we acknowledge the contributions of our colleagues and peers who provided feedback, encouragement, and assistance at various stages of the project. Their constructive criticism and encouragement have been immensely beneficial in shaping the direction and outcomes of our research efforts. Lastly, we are grateful to the research community for their ongoing efforts in advancing the field of cybersecurity and machine learning. The collective pursuit of knowledge and innovation continues to inspire and motivate our work.

REFERENCES

[1] (2019). China Internet Security Research Report. (Nov. 15, 2020).[Online]. Available. https://www.cert.org.cn/publish/main/upload/File/2019Annual%20report.pdf
[2] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, ''A survey on malware detection using data mining techniques,'' ACM Comput. Surv., vol. 50, no. 3, pp. 1–40, May 2018.
[3] S. Rastogi, K. Bhushan, and B. B. Gupta, ''Android applications repackaging detection techniques for smartphone devices,'' Proc. Comput. Sci., vol. 78, pp. 26–32, Jan. 2016.
[4] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, ''WHYPER: Towards automating risk assessment of mobile applications,'' in Proc. 22nd USENIX Secur. Symp. (USENIX Security), 2013, pp. 527–542.
[5] W. Klieber, L. Flynn, A. Bhosale, L. Jia, and L. Bauer, ''Android taint flowanalysis for app sets,'' in Proc. 3rd ACM SIGPLAN Int. Workshop State Art Java Program Anal. (SOAP), 2014, pp. 1–6.
[6] Z. Wang, J. Cai, S. Cheng, and W. Li, ''DroidDeepLearner: Identifying Android malware using deep learning,'' in Proc. IEEE 37th Sarnoff Symp, Sep. 2016, pp. 160–165, doi: 10.1109/SARNOF.2016.7846747.
[7] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, ''Data mining methods for detection of new malicious executables,'' in Proc. IEEE Symp. Secur. Privacy. (S&P), May 2001, p. 2001, doi: 10.1109/SECPRI.2001.924286.
[8] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, ''Android permissions: A perspective combining risks and benefits,'' in Proc. 17th ACM Symp. Access Control Models Technol. (SACMAT), 2012,pp. 13–22.
[9] C. Zhao, W. Zheng, L. Gong, M. Zhang, and C. Wang, ''Quick and accurate Android malware detection based on sensitive Apis,'' in Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT), Aug. 2018, pp. 143–148.
[10] H. Fereidooni, M. Conti, D. Yao, and A. Sperduti, ''ANASTASIA: ANdroid mAlware detection using STatic analySIs of applications,'' in Proc. 8th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS), Nov. 2016, pp. 1–5.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY