International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Detection of Re-Entrancy, Timestamp Dependence and Infinite Loop Attack in Smart Contracts Using Graph Convolution Network

[1]D. Saveetha, [2]G. Maragatham

Submitted: 29/05/2023 **Revised**: 17/07/2023 **Accepted**: 29/07/2023

Abstract: Smart Contract Attack Detection Using Graph Convolution Network (GCN) is a research area that focuses on identifying and preventing malicious activities within smart contracts deployed on blockchain platforms. Smart contracts are self-executing digital agreements that run on decentralized networks, such as Ethereum. While smart contracts provide transparency and automation, they can also be vulnerable to various attacks, leading to financial losses or system disruptions. To address this challenge, the concept of Graph Convolution Network is leveraged. GCN is a deep learning technique that operates on graph-structured data, where nodes represent entities, and edges represent relationships between them. In the context of smart contracts, a graph can be constructed to capture the dependencies between different functions, variables, and transactions within the contract. The goal of utilizing GCN in smart contract attack detection is to learn patterns and detect anomalies in the graph structure. By training the model on a large dataset of known secure and malicious smart contracts, it can learn to identify suspicious patterns that might indicate an ongoing attack. The GCN model can consider features such as function calls, control flow, and data dependencies to detect potential vulnerabilities or abnormal behavior. In this paper we are going to address the detection of reentrancy attack, timestamp dependence attack and infinite loop attack using Graph Convolution Network. Smartbugs wild dataset is used for performing the attack detection. By using GCN we are able to detect these attacks accurately and our model is compared with the existing models and it shows that our model is better than the existing models in terms of performance metrics.

Keywords: *Smart contract, vulnerability, Graph Convolution Network,re-entrancy,time stamp dependence attack ,infinite loop attack.*

1.Introduction

Blockchain technology has gained traction in a variety of industries due to its potential to revolutionize how data is stored and transactions are processed. Ethereum is a blockchain proposed by Vitalik Buterin.Ethereum blockchain is more versatile, allowing for the creation of smart contracts and decentralized applications, or dApps. Ethereum makes use of its own cryptocurrency, Ether, to pay for transaction fees and gas. Ethereum has attracted developers from all over the world to build decentralized applications on the Ethereum blockchain.The popularity of Ethereum has resulted in the development of many tools and frameworks to facilitate the development of decentralized applications. The tools and frameworks include Truffle, Embark, and Hardhat. Smart contracts are self-executing contracts which was introduced by Nick Szabo in the year 1996.It is made up of a series of rules and conditions that must be satisfied for the contract to be executed. Decentralized applications, or dApps, are frequently built using smart contracts. A decentralized application (dApp) is a program that runs on a decentralized network. The Ethereum Blockchain is a popular choice for many decentralized application developers. The use of smart contracts eliminates the need for a third party to mediate a contract, such as a bank or a lawyer. This not only reduces the contract's cost, but also the time required to complete the contract. Smart contracts are stored on the blockchain and are immutable, which means they cannot be modified once deployed.

The number of smart contracts established over the last few years has been steadily increasing. These contracts are used for a variety of functions, from electronic voting to the storage of medical information and decentralized financial systems. Though the popularity of smart contracts has increased, there has been a corresponding increase in the number of vulnerabilities found in these contracts. These vulnerabilities have resulted in the loss of millions of dollars in cryptocurrency assets. Early vulnerability discovery is critical for mitigating the hazards posed by these vulnerable smart contracts.

[a].*Department of Networking and Communications, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India*
saveethd@srmist.edu.in
[2]*Department of Computational Intelligence, SRM Institute of Science and Technology, Kattankulathur 603203, Tamil Nadu, India*
maragatg@srmist.edu.in

Graph Convolution Networks (GCN) is a type of deep neural networks that can be used to represent the smart contracts as a graph structure.GCN is designed to process and analyse data that is structured as graphs. Traditional neural networks are primarily used for tasks like grid-like data structures, such as images or sequences, but they may not effectively capture the relationships and dependencies present in graph-structured data.

GCNs address this limitation by incorporating graph-based operations and learning mechanisms. They can operate directly on graph data, taking into account the connectivity and interactions between nodes and edges. GCNs have gained significant attention and popularity in various domains where data is naturally represented as graphs, such as social networks, recommendation systems, molecular chemistry, knowledge graphs, and more.

It iteratively updates node representations by aggregating information from neighbouring nodes in the graph. This process allows each node to gather and incorporate information from its local neighbourhood, capturing the structural patterns and dependencies present in the graph. The updated node representations are then used for downstream tasks like node classification, link prediction, or graph-level predictions. It has wide range of applications, including node classification, link prediction, graph classification, recommendation systems, and knowledge graph reasoning.

There are various types of vulnerabilities in smart contracts but we are going to address only few attacks like re-entrancy attack, timestamp dependence attack and infinite loop attack.

Re-entrancy attack: When a contract calls an external contract without properly handling the flow of execution this type of attack arises. An attacker can exploit this by repeatedly calling back the vulnerable contract before executing the previous execution, resulting in loss of funds or causing damage to the system. DAO attack was a famous re-entrancy attack that resulted in the loss of 60 million ethers because of this vulnerability.

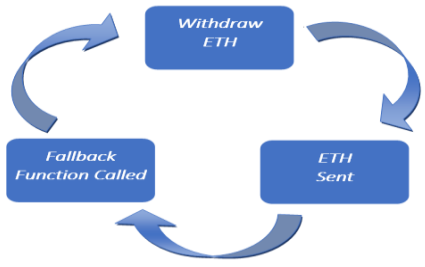

Fig 1:re-entrancy attack

In Fig 1: Shows the re-entrancy attack where a function is used to send eth and it then calls a fallback function which can be used to call a withdraw function.

Timestamp dependence attack: By varying the timestamp or referring to external timestamps can lead to vulnerabilities because time could easily be manipulated or changed to produce incorrect results.

Infinite loop attack: when a program using functions iterate in a loop with no exit condition or when the exit condition cannot be reached the system will hang leading to infinite loop.

2. Related Work

Vulnerabilities in smart contracts arise due to bugs in the coding, or attackers trying to take control of the contract or steal the funds contained within. It can lead to potential risks and issues like security breaches that can allow hackers to access private keys and steal cryptocurrencies or tamper with transactions. Network disruptions cause the network to malfunction or halt, disrupting the normal flow of transactions and creating a loss of trust in the system. Vulnerabilities in smart contracts can cause unexpected behavior and allow attackers to exploit vulnerabilities to drain funds, or to create a fork in the blockchain.It is crucial to implement robust security measures and regularly conduct security audits to identify and fix vulnerabilities.

Due to the blockchain's immutability, the smart contracts cannot be modified later after deployment if any issues arise. This demonstrates how critical it is to identify and mitigate vulnerabilities prior to deploying a smart contract. In June 2016, one of the earliest widely publicized smart contracts on the Ethereum blockchain, known as the DAO, was attacked; approximately $50 million in ETHER was lost as a result. The attacker took advantage of a recursive call bug to gain control of the DAO contract and drain its funds. Numerous studies have been conducted in the past attempting to analyze the security risks associated with smart contracts however the majority of the research work doesn't delve into the technical details of the most prevalent vulnerabilities. Reentrance attacks, indirect execution of unknown code via a fallback function, interface naming issues, and time component attacks are just a few examples of common smart contract vulnerabilities.

Smart Contract Vulnerability Detection can be done using two methods, static analysis and dynamic analysis. Tools are used to identify the vulnerabilities. In static code analysis the code is analyzed without executing,e.g. Oyente, Mythril, and Securify are popular examples. Smart contract analysis can be beneficial for identifying potential vulnerabilities and for

understanding how the code works.It enables automated security reviews and runs quickly on large contract files, it is also believed to be a cost-effective method of discovering vulnerabilities.There are usually three stages in static analysis 1. Building an intermediate representation 2. Enrichment of Intermediate Representation using algorithms such as symbolic execution and abstract interpretation 3. Vulnerability detection. In dynamic analysis we execute the code for identifying the vulnerabilities. e.g.Manticore. There are many issues that may not be identified by static analysis, because static analysis cannot identify every possible execution path of the code. However, dynamic analysis is a more expensive approach to security testing than static analysis.

Priyanka Bose et al developed SAILFISH, a prototype to detect two state-inconsistency flaws, viz., re-entrancy and transaction order dependence in Ethereum smart contracts [1].Noama Fatima Samreen and Manar H. Alalf combined the static with dynamic analyser to detect re-entrancy vulnerabilities [2]. Lejun Zhang et al proposed a novel hybrid deep learning model named CBGRU that combines different word embedding (Word2Vec, FastText) with different deep learning methods (LSTM, GRU, BiLSTM, CNN, BiGRU).They have used a dataset named SmartBugs-Wild.Their model has achieved an accuracy of 93.30%,Precision of 96.30% Recall of 85.95% and F1 score of 90.92 for re-entrancy attack.Jianbo Gao et al have developed a fuzzing-based analyser to detect re-entrancy bugs automatically in Ethereum smart contracts[4].Yinxing Xue et al have developed Clairvoyance to detect re-entrancy vulnerabilities in real world with significant higher accuracy by using a cross-function cross-contract static analysis.Yuchiro Chinen et al have found Re-entrancy Analyser which is a static analysis tool that uses both symbolic execution and equivalence checking by a satisfiability modulo theory solver[12].Daojun Han at al have described a model where the features are obtained from the AST and control flow graph of smart contract through TextCNN and GNN. The syntactic and semantic features are fused, and the fused features are used to detect vulnerabilities. They have used Eth2Vec dataset and they have got an average precision of 96% and recall of 90% [13].Jing Huang et al have used a two layer model for multitask learning. A CNN model is used to construct a classification model for learning and extracting features. It identifies 3 types of vulnerabilities while obtaining a precision of 70.31%, recall of 77.83%, and F1 score of 73.87% [14].

3. Proposed System

In our proposed system we have designed a system to detect the various attacks like re-entrancy attack, timestamp dependence attack, infinite loop attack in smart contracts using GCN.Smartbugs-wild is the dataset we have used for the experimental purpose.The diagrammatic representation of GCN is shown below in Fig 2. A smart contract is taken from the dataset and it is converted to a transaction graph. The contract is converted to graph structure with all the nodes, edge and function calls. The graph is then normalized so that all the values are similar and then passed through the GCN network .The GCN model classifies the type of attack. Hyper parameter tuning is done in order to achieve better results. The final output of the model will predict whether the attack is present or not.

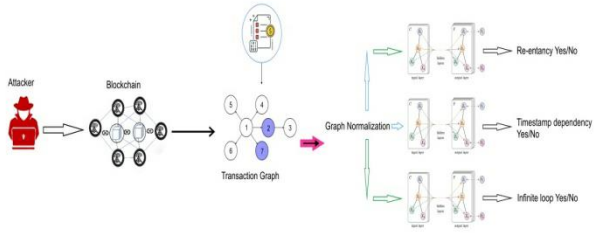

Fig 2: diagrammatic representation of GCN to detect the attacks.

Algorithm -Code to Graph

Input: Code

Output: Graph

Step 1- Map user-defined names to symbolic names (var1, var2, var3, var n)

Step 2- List the function types and its limit (Fun1, fun2, fun3, .fun n), Boolean expression, an assignment expression

Step 3- Split all the functions of the contract

Step 4- Generate a fallback node (nodes, edges)

Step 5- Generate the graph

Step 6- Store the functions

Step 7- Traverse the graph

Step 8- Get payable function (pf1,pf2,….pf n) also get non payable functions(npf1,npf2,….npf n)

Dataset Description

In table 1, The dataset used for our research is the Smartbugs Wild dataset, which serves as a benchmark dataset for our research. This dataset comprises a substantial collection of smart contracts, with a total count exceeding 47,398 contracts.

Table 1: Dataset description

SmartBugs-Wild Dataset
47,398 smart contracts extracted from the Ethereum network.
35,151 contracts have vulnerability and 12,247 contracts are without vulnerability

4. Experimental Results

The experiment was carried out on a Lenovo ThinkPad system which is of the given specification Intel(R) Core(TM) i5-8365U CPU @ 1.60GHz 1.90 GHz, 8.00 GB,X64 based processor. Python-Keras was used to build the model. Hyper parameter tuning is done to achieve the expected results.20% of the smart contracts is used for training while the remaining is utilized for testing.After running the model it is able to identify re-entrancy attack, time dependence attack and infinite loop attack with better accuracy, precision, recall and F1 score. Our results are compared with other models and found out that our model works better than other models in identifying the vulnerabilities. The specific configurations and settings used throughout the experiment are specified by the simulation parameters in Table 2.These parameters cover the number of hidden layers in the GCN model, the activation functions employed, the learning rate, the optimizer, and the loss function. The experiment creates the architecture and training parameters for the GCN model by defining these parameters, allowing simulation and evaluation of the model's performance on the provided task or dataset.

Table 2: Simulation setup for GCN

Parameter	**Value**
GCN layers	2
Learning rate	0.01
Epoch	100
Optimization algorithm	Adam
L2 regularization	10
Batch size	5
Feature channels	100
Drop out rate	0.5

Re-entrancy attack

The model accuracy for re-entrancy attack is shown in Fig 3.The model stabilizes over time, initially training and testing process had a range of fluctuations but with the increase in the number of iterations they fitted well in the later stage. Better the fitting of the accuracy curve during training and testing the stronger the generalization ability of the model.

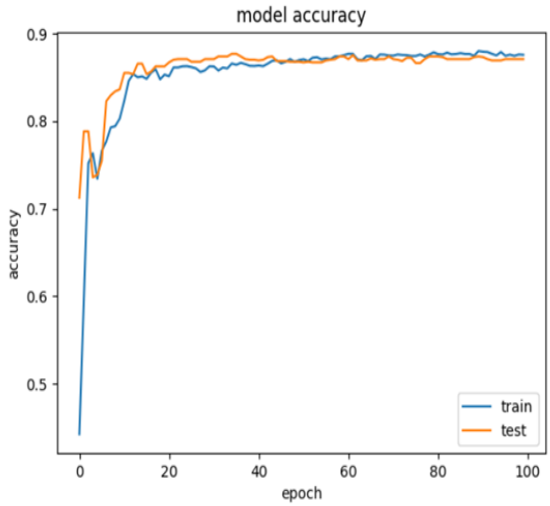

Fig 3:re-entrancy attack model accurcay

The model loss for re-entrancy is shown in Fig 4.The loss decreases gradually over time.The loss value curve had certain fluctuations in the initial stage but with the increase in the number of iterations the model fitted well for training and tesing. So the stronger is the generalization ability of the model.

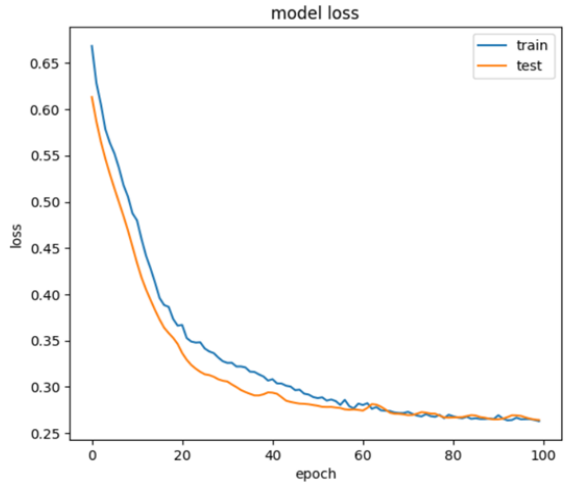

Fig 4:re-entrancy attack model loss

Re-entrancy attack comparison with other models

The outcomes is shown in Fig 5,where the performance metrics like precision,recall,F1 score and accuracy are used. The graph shows that GCN model outperforms the existing model CB-GRU. The accuracy of the GCN model was 97%, which is 4.27% more accurate than the accuracy of the CB-GRU. The GCN model might be able to effectively categorize and detect the desired targets or behaviour's as a result of the increase in accuracy. The comparison shows how the suggested GCN model outperforms the current CB-GRU model in terms of its capacity to deliver more accurate and trustworthy outcomes for the given task.

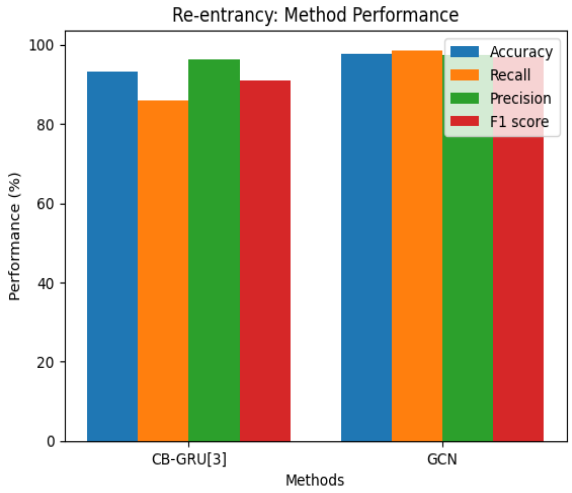

Fig 5:Re-entrancy comparison with other models

Timestamp Dependence attack

This type of attack is caused by the diffeence in timestamp of the blocks in the blockchain. In Fig 6,we can find the model accuracy increases as the no of epochs(iterations) increases.

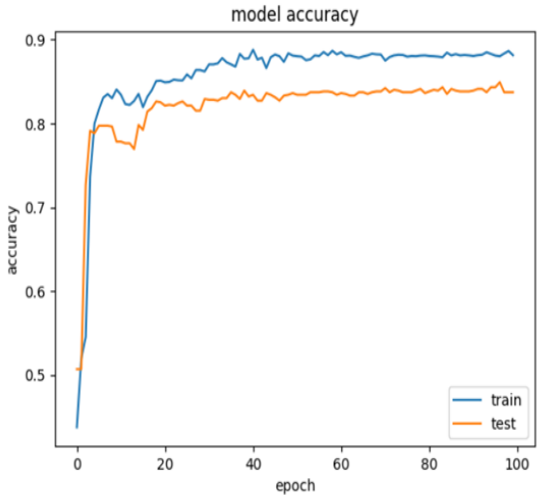

Fig 6: timestamp attack model accuracy

As training progresses, the model appears to get better at correctly predicting the timestamp attacks. This increase in accuracy demonstrates that the model is successfully capturing the patterns and traits connected to the timestamps in the provided dataset.Similar trends in the testing and training data indicate that the model can generalize and perform well outside of the training set. The fact that timestamp accuracy is constantly higher during the testing phase supports the model's capacity to generate precise predictions about data that has not yet been viewed.The model's general capacity to comprehend and make appropriate use of temporal information is demonstrated by the overall trend of rising accuracy for timestamps in both training and testing data. It illustrates the model's development in learning and using the timestamp information to provide precise predictions or classifications over the course of training.

In Fig 7,we can find the model loss decreases as the iteration increases. The graph shows how the loss has been steadily declining over time. Initial training and testing results showed a strong match, demonstrating that the model was successfully reducing error. The loss curve saw a range of changes as a result of the fit degree decreasing with the number of repetitions. This shows that the parameters of the model were being adjusted to better match the timestamp data. The changes in the loss curve show that corrections are being made to lower the mistakes and boost the model's efficiency.

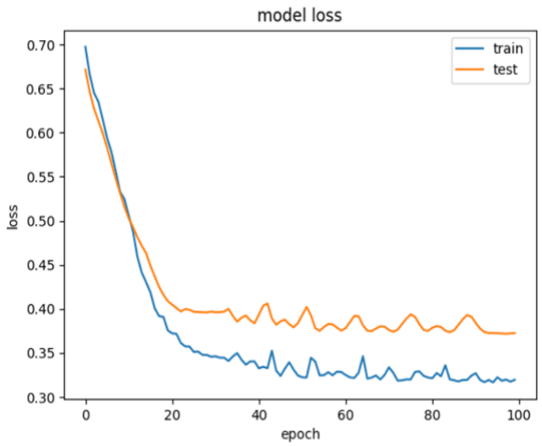

Fig 7: Timestamp dependence model loss

Timestamp dependence comparison with other models:

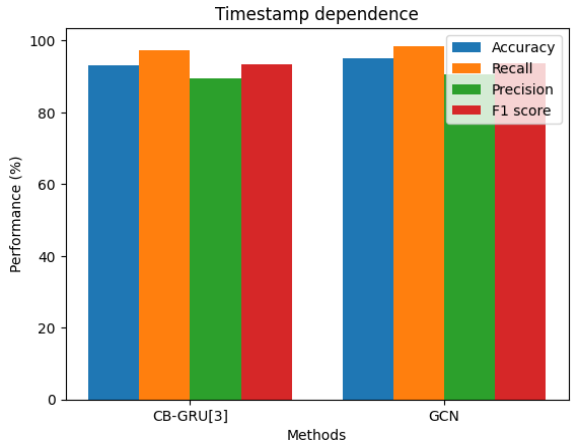

Fig 8: Timestamp dependence comparison with other models.

As shown in Fig 8, we can find our model is having better performance metrics than the other model CB-GRU. The Timestamp dependence dataset comparison with other models, the GCN model has better performance metrics than the other model for timestamp dependence. It has achieved 95% accuracy, which is 2.13 % more than the existing model CB-GRU.

Infinite Loop

In Fig 9, the graph shows a gradual improvement in accuracy over time, demonstrating the model's enhanced capacity to recognize endless loops. The growing accuracy indicates that as training proceeds, the model gets better at identifying the presence of endless loops in the input dataset. This increase in accuracy shows that the model is better able to represent the patterns and traits connected to infinite loops, resulting in more accurate predictions.

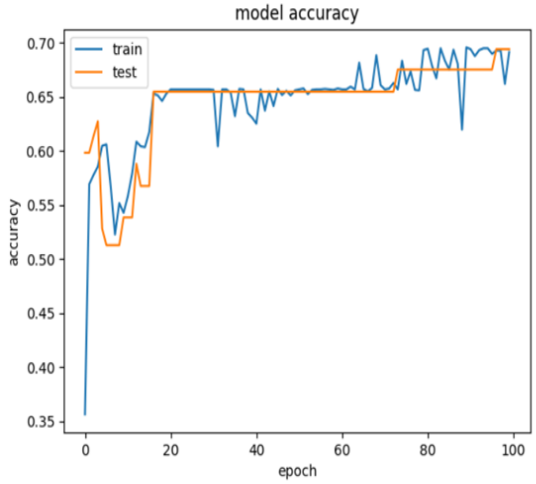

Fig 9: Infinite loop model accuracy

In Fig 10, we can find very high fluctuations in initial stage but as the epoch increases the model is trying to fit in near future. The graph shows how the loss has been steadily declining over time. The loss function initially did not fit well during the initial stages of training and testing, showing that the model's predictions differed from the actual data. However, as the number of iterations increased, the fit degree improved, resulting in a decrease in the loss function. This suggests that the model was able to learn from the data and adjust its parameters to minimize the error in predicting infinite loops.

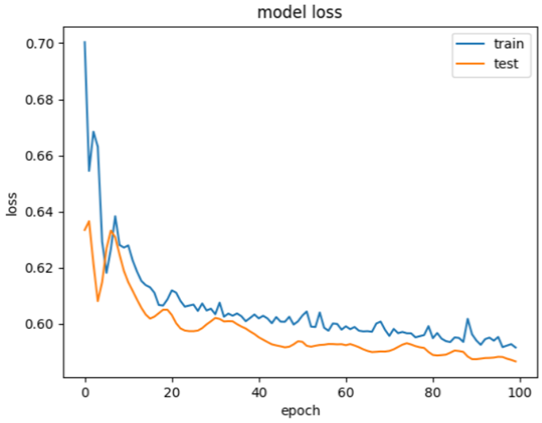

Fig 10: Infinite loop model loss

Despite the overall decreasing trend, there is a certain range of fluctuations in the loss curve. These fluctuations indicate adjustments and fine-tuning made by the model to further reduce the errors and improve its performance

Infinite loop comparison with other models

Fig 11, depicts the model is having better performance metrics than the other model CB-GRU. The GCN model

performed admirably, obtaining an astounding accuracy of 96%. This accuracy rate is 3.24% higher than that of the current CB-GRU model. These findings demonstrate how well the GCN model handles the intricate connections and patterns found in the endless loop dataset.

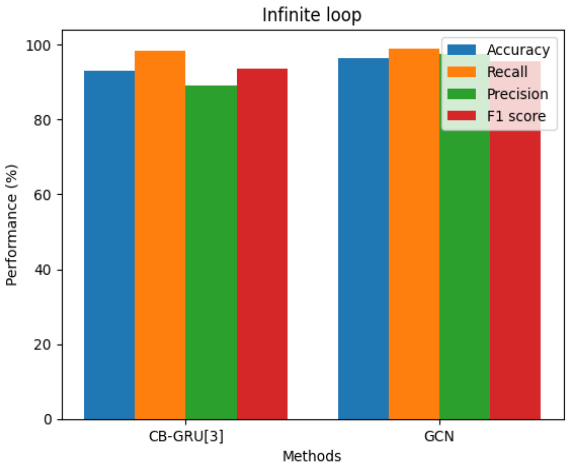

Fig 11: infinite loop comparison with other models.

Comparison of Performance Metrics:

As depicted in table 3, by using GCN attacks like re-entrancy, timestamp and infinite loop in the code are detected.GCN model is able to achieve higher accuracy, precision, recall and F1 score than the existing model CB-GRU.Compared to CB-GRU, the model has 4% increase in accuracy, 13% increase in recall, 1% increase in precision, and 6% increase in F1 score for re-entrancy attack. Comparing GCN model with CB-GRU with respect to timestamp dependence attack the model has 2% increase in accuracy, 1% increase in recall, 1% increase in precision, and 0.32% increase in F1 score .Compared to CB-GRU, GCN model is having 3% increase in accuracy, 0.74% increase in recall, 9% increase in precision, and 2% increase in F1 score for infinite loop attack.

Table 3: Comparison of Performance Metrics

	Re-entrancy				**Timestamp dependence**				**Infinite loop**			
Methods	Accuracy (%)	Recall (%)	Precision (%)	F1 (%)	Accuracy (%)	Recall (%)	Precision (%)	F1 (%)	Accuracy (%)	Recall (%)	Precision (%)	F1 (%)
CB-GRU[3]	93.30	85.95	96.30	90.92	93.02	97.45	89.47	93.29	93.16	98.29	89.15	93.50
GCN	97.57	98.61	97.34	96.95	95.15	98.52	90.72	93.61	96.40	99.03	97.57	95.68

5. Conclusion and Future Work

Based on the results we are able to demonstrate that GCN model is able to achieve more accuracy than the existing models and it is best to predict the vulnerabilities like re-entrancy, timestamp dependence and infinite loop attack that are present in the smart contract. This work can be enhanced by applying to other graph neural network and its sub-types. A global model can be developed to detect all types of vulnerabilities existing in blockchain. NIST framework can be used to test the model.

References:

[1] Priyanka Bose, Dipanjan Das, Yanju Chen, Yu Feng, Christopher Kruegel, and Giovanni Vigna University of California, Santa Barbara,IEEE,2022.

[2] Noama Fatima Samreen, Manar H. Alalfi ,A Survey of Security Vulnerabilities in Ethereum Smart Contracts, arXiv,2021.

[3] Lejun Zhang,Weijie Chen,Weizheng Wang ,Zilong Jin, Chunhui Zhao, Zhennao Cai and Huiling Chen,"CBGRU: A Detection Method of Smart Contract Vulnerability Based on a Hybrid Model", Sensors , Vol 22, 3577,2022.

[4] Jianbo Gao, Han Liu, Yue Li, Chao Liu, Zhiqiang Yang, Qingshan Li, Zhi Guan, Zhong Chen," Towards automated testing of blockchain-based decentralized applications**," ICPC '19: Proceedings of the 27th International Conference on Program Comprehension,May 2019.**

[5] Qian, Peng, Zhenguang Liu, Qinming He, Butian Huang, Duanzheng Tian, and Xun Wang. "Smart Contract Vulnerability Detection Technique: A Survey." arXiv preprint arXiv:2209.05872 (2022).

[6] Zhang, L.; Wang, J.; Wang, W.; Jin, Z.; Zhao, C.; Cai, Z.; Chen, H. A Novel Smart Contract Vulnerability Detection Method Based on Information Graph and Ensemble Learning,Sensors 2022.

[7] H. Wu et al., "Peculiar: Smart Contract Vulnerability Detection Based on Crucial Data Flow Graph and Pre-training Techniques," 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), 2021, pp. 378-389, doi: 10.1109/ISSRE52982.2021.00047.

[8] Jiaming Ye, Mingliang Ma, Yun Lin, Lei Ma, Yinxing Xue, Jianjun Zhao,Vulpedia: Detecting vulnerable ethereum smart contracts via abstracted vulnerability signatures,Journal of Systems and Software, Volume 192, 2022.

[9] Yuan Zhuang, Zhenguang Liu, Peng Qian, Qi Liu, Xiang Wang, and Qinming He.," Smart contract vulnerability detection using graph neural networks" Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI'20). Article 454, 3283–3290,2021.

[10] N. Atzei, M. Bartoletti, and T. Cimoli. A survey of attacks on ethereum smart contracts (sok). In Principles of Security and Trust", pages 164–186. Springer, 2017.

[11] Jennifer.j.Xu ",Are blockchains immune to all malicious attacks?",2016 Xu Financial Inclusion.

[12] S.Nakamoto,"Bitcoin: A peer-to-peer electronic cash system",2008.

[13] Yuichiro Chinen, Naoto Yanai, Jason Paul Cruz, Shingo Okamura," Hunting for Re-Entrancy Attacks in Ethereum Smart Contracts via Static Analysis", IEEE Blockchain 2020.

[14] Daojun Han, Qiuyue Li,Lei Zhang and Tao Xu ,"A Smart Contract Vulnerability Detection Model Based on Syntactic and Semantic Fusion Learning", Wireless Communications and Mobile Computing, Vol 2023, Article ID 9212269,2023.

[15] Jing Huang, Kuo Zhou, Ao Xiong, Dongmeng Li," Smart Contract Vulnerability Detection Model Based on Multi-Task Learning", Sensors, Vol 22, 1829, 2022

[16] Kumar, S. A. S., Naveen, R., Dhabliya, D., Shankar, B. M., & Rajesh, B. N. (2020). Electronic currency note sterilizer machine. Paper presented at the Materials Today: Proceedings,37(Part 2) 1442-1444. doi:10.1016/j.matpr.2020.07.064 Retrieved from www.scopus.com

[17] Wanjiku, M., Ben-David, Y., Costa, R., Joo-young, L., & Yamamoto, T. Automated Speech Recognition using Deep Learning Techniques. Kuwait Journal of Machine Learning, 1(3). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/135

[18] Elena Petrova, Predictive Analytics for Customer Churn in Telecommunications , Machine Learning Applications Conference Proceedings, Vol 1 2021.