Smart Contracts Quality Measurement using Bayesian Networks

K.Lakshmi Narayana1, Dr.K.Sathiyamurthy2

*1Ph.D Scholar, Dept. of CSE, Puducherry Technological University, Puducherry, India.*

*2Associate Professor, Dept. of CSE, Puducherry Technological University, Puducherry, India.*

**Abstract.** Smart contracts (SC) is a key component in the Ethereum Blockchain. Smart contracts (SC) are normal computer programs which are written mostly in solidity object-oriented programming language. Ethereum Blockchain allows to complete the transactions only if transaction obeys the SC rules. SC are not modifiable once they deployed into the Blockchain. Thus it is essential to verify the quality of smart contract before deploying into the Blockchain. In this paper, Bayesian Network Model was designed and constructed to measure the SC quality. The results showed that the proposed design is able to measure the SC quality in terms of probability and also suggest the reasons for the good or poor results. The accuracy of the proposed model results are improved (accuracy 8% increased for both Reentracy and Tx.origin, 6% increased for DOS), compared with traditional method LSTM. This proposed design and implementation is the first attempt to measure the smart contract quality using Bayesian Networks.

Keywords: Blockchain, Ethereum, Smart Contracts, Quality Metrics, Bayesian Network, Expert Knowledge*.*

**1. Introduction**

Smart Contracts(SC) are the programs of predefined rules which are deployed into the Blockchain and these programs execute automatically to determine that every transaction has to satisfy the predefined conditions to complete the transaction. In a Blockchain, transactions among two parties are recorded in an efficient, verifiable, and unchangeable [1,2,3]. Blockchain can present an innovative solution for long-standing problems of security and data storage in centralized systems[4]. Smart Contracts worked based on conditional reasoning statements. Nowadays smart contracts have been used widely in the business among a group of untrusted persons, where every transaction can be completed according to rules agreed upon by all business stakeholders without the involvement of third-party verification[5]. SC are written in Ethereum Blockchain using “Solidity” object oriented programming language, it is inspired from java script, python and C++ languages. Other languages can also be used to write smart contracts in other Blockchain platforms which are Golang, Vyper, Yul, DAML, java, Javascript and C++.

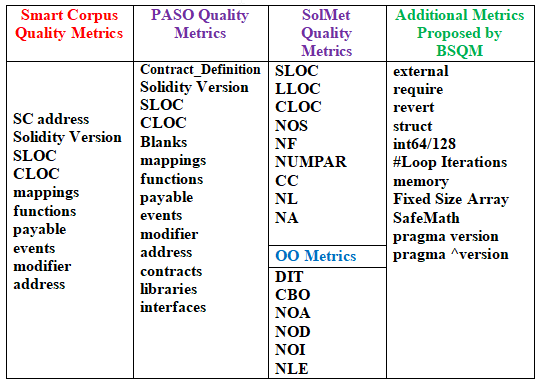
The reasons for attacks on smart contracts are classified into three categories: first, the smart contracts of Ethereum are mainly money-oriented transactions; second, deployed vulnerable smart contracts can’t be modifiable; and finally, smart contracts have no defined quality measures/metrics[7]. Thus the proposed paper providing solution to the problem of lack of SC quality measures using Bayesian Networks.

Explanation about s/w metrics and SC related metrics and mapping them with SC Quality.

Definitions of each metric. Reseach problem.

**General software quality measures:** can be classified into code quality, reliability, performance, code re-usability, maintainability, performance and security[16]. Code quality metrics are number of lines, complexity, number of functions and rate of bugs generation. Reliability quality metrics are MTBF (Mean Time Between Failures) and MTTR (Mean Time To Repair). Performance quality metrics are time and resources being utilized. Code re-usability metrics represents how easy to integrate with other required software’s. Maintainability represents time requires to adapt new features or new functionality to existing software and MTTC (Mean Time to Change). Security metrics represents no unauthorized changes, no fear of cyber attacks, when the software product is in use by the end-user.

**Smart Contract related quality metrics:** In the literature authors used smart corpus, PASO and SolMet tools to generate smart contract related quality metrics as shown in **Table1.** The tool PASO generating many metrics compared with other tools. But using these limited metrics not sufficient to meet all software quality measures. Hence the proposed BSQM model used additional metrics to cover all software quality measures.



SLOC: Source Lines of Code

Bayesian Networks are significant as a unified probabilistic framework for classification. Over the past few years, Bayesian Network has found successful applications in various areas such as medicine, document classification, information retrieval, semantic search, image processing, spam filter, system biology etc [12].

The major contributions of the proposed work are as follows:

* In this research, Bayesian network(BN) model proposed for measuring the quality of SC, because of following reasons.
* BN are suitable for the problems which are having limited data sets.
* BN can predict the probability (severity) of each quality measure, rather than predicting only YES/NO answers.
* BN can also suggest reasons for good, medium or poor quality of SC.

The Association of the remaining paper follows the sequence, section2 gives literature work on software quality metrics and smart contract metrics, section3 explains Proposed BSQM design for SC quality measurement, section4 demonstrates experimental setup, comparison table and outcomes, finally conclusion & future scope will be in section5.

1. **Literature Work**

In the literature, papers [1,2,3,4,5,6] were published on behalf of smart contract quality metric. Nemitari Ajienka et al. [1] investigated the SCs nature in terms of their OO attributes to understand the relation with GasUsed attribute. The authors[1] extracted the OO metrics which are comment lines of code (CLOC), logical lines of code (LLOC), source lines of code (SLOC), number of functions (NFs), number of parameters (NUMPARs), nesting level (NL), cyclomatic complexity (CC), nesting level without else-if (NLE), number of statements (NOSs) and number of attributes or states (NA) using a tool “SolMet”. The gasUsed attribute is more sensitive to the size measurements (SLOC, LLOC) and less to the structural characteristics (CBO, NOC or LCOM). The authors[1] came to conclusion that GasUsed attribute have a significant correlation with the size attributes (NOS, LLOC and NL). NOS metric (is part of the SLOC) and inheritance nesting levels (NL) must be reduced to reduce gas costs. Limitations of this paper[1] is only concentrated on GasUsed measure with respect to to commonly used OO metrics of SC.

1. Tonelli et al. [2] collected the blockchain addresses, the Solidity source code, the ABI and the bytecode of 12000 contracts from Etherscan.io website and extracted a smart contract specic software metrics such as blank lines, line of comments number of functions, SLOC, number of events calls, cyclomatic complexity, number of mappings to addresses, number of payable, number of modiable and perform analysis of which and to what extent the SC metrics influence samrt contract performance and their results proved that SC metrics have more restricted than the corresponding metrics in traditional software system.

Giuseppe Antonio et al. [3] have noticed the lack of SC dataset and lack of SC quality measures and proposed web based Ethereum SC souce code and its metrics repository (https://aphd.github.io/smac-corpus/) for last five years, which is easy to use, large and well organized repository, which is very much helpful to users, researchers and developers of Ethereum Blokchin. Pierro and Tonelli [7] also provided a web based SC repository (https://aphd.github.io/paso/), called PASO, it can provide SC commonly used metrics for the given smart contract address.

N'Da AAK et al. [4] study focuses on the 15 security metrics which are common in both SC security metrics and general software security metrics. Authors[4] noticed that SC is security vulnerable if SC with high complexity in code structure. Hence SC developers have to take care SC complexity must be low and be simple as possible before deploying into Blokchain. The authors also stated that only the metrics WMC, DIT, CBO and McCabe complexity can be colleted using the “Solmet” tool and no tools are available to extract remaining metrics which have to be colleted manually. Autoextraction of these remaining metric is one of the reseach issue.

Damian Rusinek et al. [5] proposed 14-part checklist called SC Security Verification Standard (v1.1) to standardize the SC security at every stage of SC-DLC (development Life Cycle) from design to implementation stages. This chek-list will very much helpful to security reviewers, developers, architects, and vendors to avoid the majority of known security problems.

Andrea Pinna et al. [6] collected 10K smart contracts from Etherscan website and prepared meta-data regarding their interations with Blockchain to understand the relationship between SC metrics and their impact on quality. Authors[6] found that the SC metrics showing less values than tranditional software metric values, but high variance. Authors stated different analysis results which are (i) found strong evidance on the practice of code reuse, that is new SC are developing by making use of existing SC’s, (ii) SC name is not always maching with its actual work, (iii) number of transactions of SC showing power-law distribution, that is SC with low balance may have many transactions and vice versa, (iv) number of transactions of a SC is not correlated to the SLOC of a SC that is well written SC., (v) SC complexity is middly coorelated to the no. of functions ( that is five time more than avg no. of functions per SC) and SLOC (greater than 300 lines)., (vi) average number of lines per SC is about 180 lines, (vii) SC are heterogeneous type and many deployed SC are deployed by unexperienced developers as a trail and experiment purpose without following standard SC structure.

Marco Comuzzi et al. [8] proposed that data carried by transaction payload plays a significant role on transaction execution in terms of time and cost. Data Quality (DQ) controls that are dependent on oracles (fetching off-line data) have a high impact on the amount of required resources. DQ is evaluated by considering different DQ dimensions which are data consistency, metrics, completeness and accuracy. Each dimension can be assessed with multiple assessment metrics (single variable- single value, single variable-multiple values, multiple variables- single values, multiple variables- multiple values). Hence authors suggesting Blockchain should also consider data controls into consideration while measuring its quality.

Bayesian Networks are significant for prediction or classification problems if we have prior probabilities of required events. Daniel Kottke et al. [15] proposed a Bayesian approach to deal with uncertainties to determine posterior probabilities with help of prior distribution. Eunjeong Park et al. [21] proposed a paper for predicting post-stroke outcomes with available risk factors probabilities using Bayesian Networks. Daniel Kottke et al. [22] proposed a Bayesian approach to deal with uncertainties to determine posterior probabilities with help of prior distribution. Zhao et al. [23] proposed a Bayesian networks to mine the knowledge and data information from web text and present in a way that users can easy to understand. Chen et al. [24] proved probability technique called Bayesian networks are the good choice for complex engineering systems with limited data to prevent failures.

**Gap in the Literature:** Generally software quality measures are classified into code quality, code correctness, reliability, code usability, maintainability, performance and security. In the literature authors used tools (Smart Corpus, SolMet and PASO) to extract only few metrics of SC, by which not possible to measure the all software quality measures of SC.

**Reason to choose proposed methodology:** Bayesian Networks are preferable for limited data set applications[24]. Bayesian networks describe the severity (probability) of each quality measure, and they can also describe the reasons for each quality measure outcome (Low, Medium, High). After analyzing the limitations in the literature and the benefits of Bayesian Networks, the proposed work focused to use Bayesian Networks to measure the smart contract quality using the prior knowledge of metrics and their impact on SC quality. Best of our knowledge, usage of the proposed BSQM design is the first attempt to measure the smart contract quality. The next section described the architecture of the BSQM for SC Quality measure.

**3. Proposed System**

The architecture of the proposed model consists of two parts which are BNMC design phase and BNMC validation phase as shown in Figure2. The first phase (BNMC design) consists of mainly five modules which are data set preparation, pattern extraction, preparation of pattern frequency table, construction of Bayesian Network model and filtering top important patterns for each vulnerability. The second phase (BNMC validation) of the architecture consists of total four modules which are pattern extraction for a given new SC, BN information Table preparation, finding severity of each vulnerability and providing final results. The BNMC design phase continued in this section. The detailed discussion about BNMC validation phase provided in section 4.

In the BNMC design phase, initially a dataset of smart contracts and their vulnerabilities have been created by the usage of online resources [9, 11, 30] as shown in the first module of Figure2. All smart contracts related to a particular vulnerability are maintained in a single document (DOCj).

*3.1 Metrics Extraction*

Key patterns can be extracted from the smart contracts data set which was created in the previous step as shown in Figure3, it is showing examples of key patterns for a given smart contract. Different patterns lead to different vulnerability possibilities. Sometimes a sequence of patterns is also important to identify a particular vulnerability. To detect the re-entrancy vulnerability, important patterns are msg.sender.call.value() invocation (let it consider pattern P1), balance[msg.sender]=0 (let it P2). The patterns order P1 followed by P2 leads to re-entrancy vulnerability, however, the sequence P2 followed by P1 doesn't re-entrancy vulnerability. “Tx.origin” is the required pattern to detect “transaction origin” vulnerability. The required patterns to detect DOS vulnerability are if( function\_call()), .gas(value), send(digits) etc as shown in **Table1**. This Table 1 was prepared, after a careful analysis of the relationship between vulnerabilities and patterns as per the information from [11].

From this Table1, some patterns that are common for more than one vulnerability are treated as independent patterns (Ex: P1, P4), for which we may give low priority. The vulnerability of a given SC can’t be determined by considering these common patterns alone, hence common patterns are treated as independent patterns(IP). Some patterns are unique for a particular vulnerability are treated as dependent patterns(DP), for which we have to give high priority (For example, P11, P12, P13, P14 are unique for V2). For the V3 vulnerability, P15 is unique.

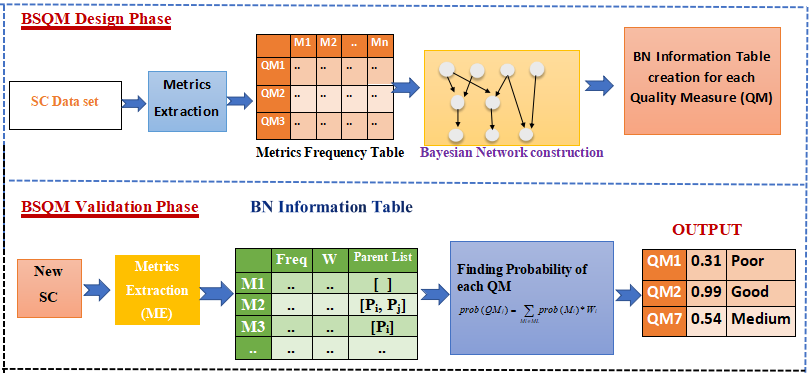


Figure 2. Architecture of BSQM for MeasuringSmart Contract Quality

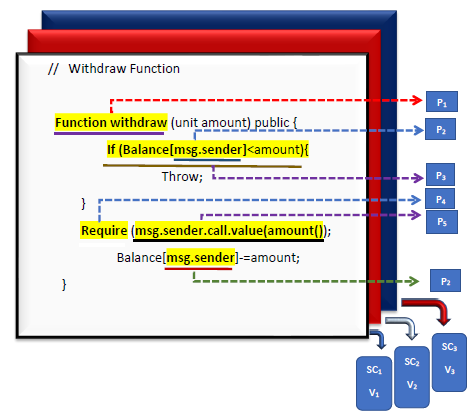


Figure 3. Pattern Extraction from Smart Contract

*3.2 Preparation of Metric Frequency Table*

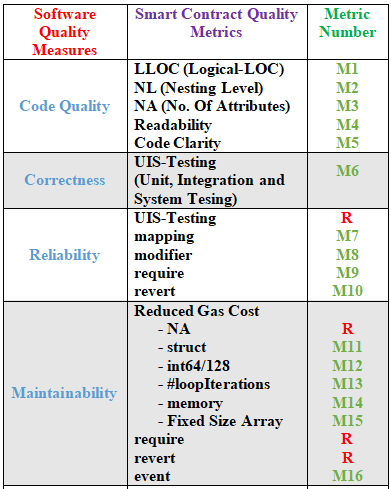
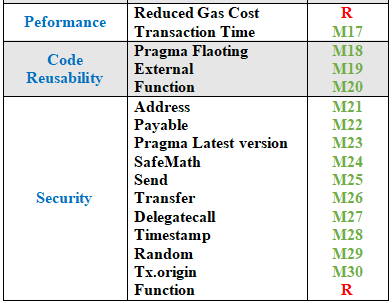
From each vulnerability document (DOCj), the frequency of each pattern was recorded in the Pattern Frequency Table (PFT), which helps to calculate pattern probabilities and to prepare CPT values. These patterns were re-arranged based on the importance (Highest to lowest) of the patterns [11]. To construct Bayesian Network, top patterns were selected for each vulnerability from PFT.

*3.3 Bayesian Network Construction*

Bayesian networks are a probabilistic graphical model, that consists of nodes and directed edges between nodes. All variables/attributes are represented with nodes and conditional dependency between nodes are represented with directed edges. The missing connections between the nodes in the network indicates conditionally independent. BN models can be prepared by experts after careful analysis of data, then the constructed model can be used to predict the test events. BN models can be challenging to design, since lack of domain information completely to specify conditional dependence between variables. Even if available, it requires many calculations to specify full conditional probabilities for an event. Hence alternative solution is to specifying dependencies between variables as per available data and treating remaining all variables are conditionally independent. In the proposed BNMC design, all patterns (Ex: P1, P2, ..) are considered as nodes/circles and sequence of edges between the nodes represents conditional dependencies between the patterns that are influencing for a particular vulnerability. All vulnerability types (Ex: V1, V2, V3) are represented as a leaf nodes in the network.

A Bayesian network as shown in [Figure4](#F4), can be constructed after analyzing functional dependencies and sequences between patterns for each vulnerability. In figure4, the patterns P1, P4, P6 are influencing more than one vulnerability and are considered independent patterns. The patterns P1 and P4 are influencing both V1 and V2 vulnerabilities with different probabilities; P1, and P6 are influencing V1 and V3 vulnerabilities. In the Bayesian Network, all independent patterns are placed in the first row. The sequence of patterns that influencing more for vulnerability are represented by arrows between the nodes in a network [[11](#SCDataset)]. Each node in the Bayesian Network will maintain CPT (Conditional Probability Table) which gives the probability of each pattern that influences the vulnerability given by the presence or absence of parent patterns as shown in [Figure 6](#F6)&7. A detailed discussion about the CPT explained in section4. The next section describes the experiment details and comparison results.

Table 1. SC Quality Metric Vs Metric Number

**4. Experimental Setup and Comparison Results**

*4.1 CPT Preparation Phase*

A data set is prepared with three separate documents for three vulnerabilities which are Re-entrancy, DOS, and Tx.origin. All SCs of the same vulnerability could be maintained in the same document. Patterns were extracted (as shown in Table1) from each vulnerability document to prepare the Pattern Frequency Table. For extracting patterns, string pattern concepts in python are essential. The frequency of each pattern is shown in Figure5.

These patterns were rearranged based on the importance (High to Low) of the patterns to detect particular vulnerability [11] and the top important patterns for each vulnerability were selected.

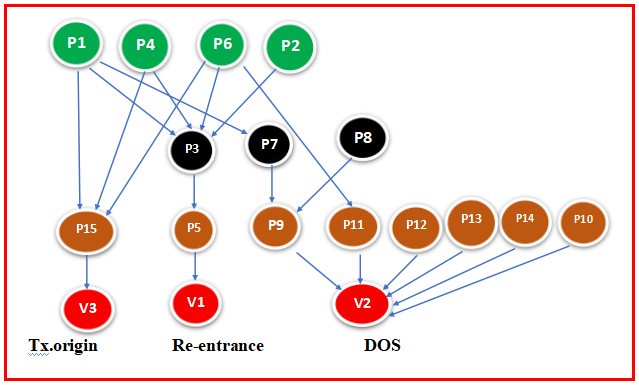


Figure4. Bayesian Network Construction

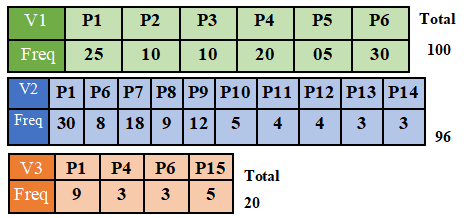


Figure 5. Frequency Table for Input Data Set

The Bayesian Network which is a directed acyclic graph was constructed for the important selected patterns by considering the relationship between patterns sequence and vulnerability as shown in figure4. The network starts with independent patterns in the first level(Level1). Then based on the sequence between patterns that influence vulnerability, the further network will grow in a downwards direction. The last row in the network is the vulnerability deciding level, which consists of V1, V2, and V3. The patterns which are influencing more for vulnerability were placed in the last before level(Level3) in the network. Each node in the Bayesian Network maintains a Conditional Probability Table (CPT). For independent patterns, CPT consists of only a single entry that is the probability of that pattern itself because those are not dependent on any other patterns. For remaining patterns in the network, the number of entries in CPT for a given pattern is 2n, where n is the number of parent nodes on which the pattern depends on. The assignment of weight for the patterns can be calculated using the following equations.

 --->Eq(1)

Wil = Weight for the pattern-i in the level-L.

TL = Total number of levels in the BN

For all the nodes in level-1 which are Independent Patterns (IP), the weight assumed is 0.5 (low priority), since IP is less influencing the vulnerability. The probability of Independent patterns will get half when multiplied by 0.5. For the remaining patterns in the lower levels, weight increases as the level increases as per the Eq(1). To increase the vulnerability prediction accuracy, the weight has to be high for the dependent patterns, if its actual probability is low i.e weight of the dependent pattern is inversely proportions to its probability as shown in Eq(2).

 ---> Eq(2)

Where Wid = Weight of dependent pattern-i

The Bayesian Network and CPT for the re-entrancy vulnerability are shown in **Figure6** & **Figure7** respectively. Probability(P3) depends on four parent patterns which are P1, P2, P4, and P6, hence the number of entries in CPT of P3 is 24 = 16. But in F**igure7**, only three entries were shown because of space restrictions. The second row in **Figure7**, is Prob(P3|P1=T, P4=F,P6=T, P2=F)=0.22. This value is calculated as 10/(100-(25+30)), where 100 is the total number of patterns in V1 smart contracts(DOC1); 10, 25 and 30 are the frequency count of patterns P3, P1 and P6 respectively. The remaining entries of CPT for other patterns can be calculated as done in the earlier step. Bayesian Network and CPTs for the DOS and tx.origin vulnerabilities are shown in **Figure8 & Figure9** respectively.

*4.2 BNMC Validation Phase*

To find the severity of each vulnerability for a given new smart contract, first, we have to extract patterns from it. For the extracted patterns, severity of each vulnerability can be calculated by using CPT values of the Bayesian Network and pattern weight. Bayesian Network information can be maintained the table using 2D Arrays to access efficiently for vulnerability prediction as shown in Table2.

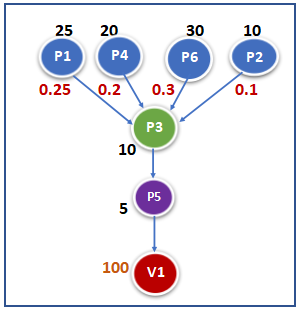


Figure 6. BN for Re-entrancy Vulnerability.

The Table2 is prepared from figure6. In Table2, column1 represents, patterns which are influencing V1 (Re-entrancy) vulnerability, second column is frequency of each pattern from V1 smart contract dataset(DOC1), third column represent whether pattern is dependent(1) or independent(0), fourth column is weight assigned to a pattern and last column is parent node list for dependent patterns. Bayesian Network Information tables for figure8 and figure9 also be created as Table2 for DOS and Tx.origin vulnerabilities respectively. The algorithm for BNMC validation phase is shown in Figure 10.

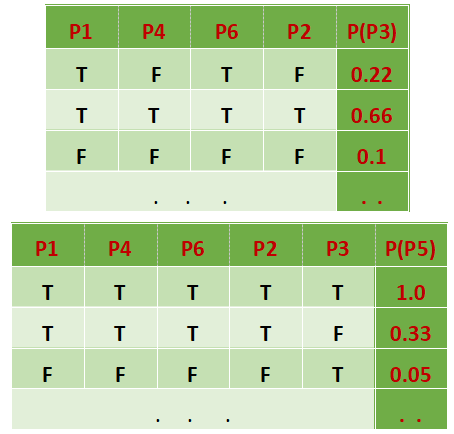


Figure 7. CPT for Re-entrancy patterns

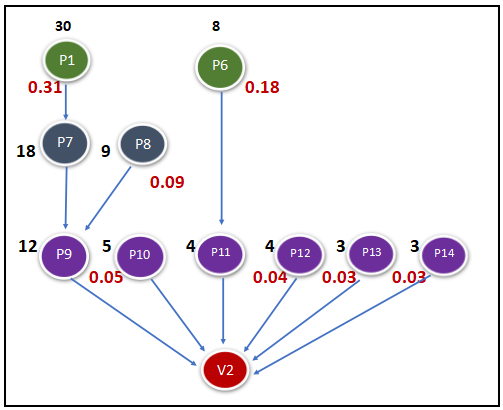


Figure 8.1 BN for DOS Vulnerability.

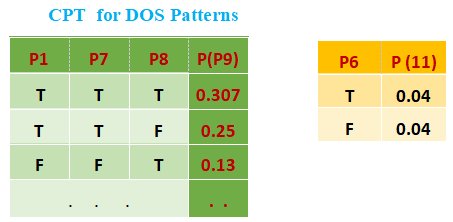
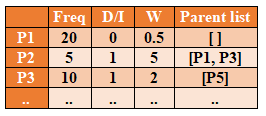


Figure 8.2 CPT for DOS Patterns

Table2. Bayesian Network Information for V1



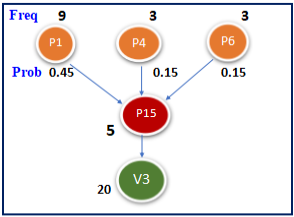


Figure 9.1 BN for Tx.origin Vulnerability

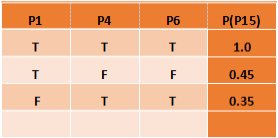


Figure 9.2 CPT for Tx.origin Vulnerability

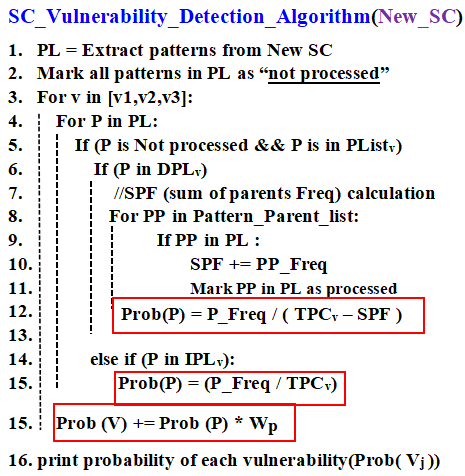


Figure10. Algorithm for BNMC Validation

SPF = Sum of Parent Frequencies

TPCv = Total Patterns Count in DOCv

PP = Pattern Parent

DPL = Dependent Pattern List

PL = Pattern List

IPL = Independent Pattern List

Updated probability value after multiplying with weight can be calculated using Eq(3.1 & 3.2).

 --> Eq(3.1)

--> Eq(3.2)

Severity of each vulnerability can be calculated using the equations 4&5.

--> Eq(4)



-->Eq(5)

In the Eq(5), DP=Dependent Pattern

IP = Independent Pattern

PFi= Pattern Frequency of Pi

TPj= Total Patterns in DOCj

SPFi = Sum of Parent Frequencies for patterni

Prob(VJ)= Probability(Vj) where j is from 1 to 3.

Prob(PI J)= Probability of pattern Pi in DOCj

 --> Eq(6)

*Vulnerability Testing:* For example, for a given new smart contract which is actually having DOS vulnerability, the extracted patterns are stored in PL(Pattern List). PL={P1, P4, P6, P7, P9}.

To calculate the severity of V1 (reentrancy vulnerability), consider only the patterns P1, P4, and P6, because other patterns P7 and P9 are not in the V1-list i.e these two patterns are not influencing the V1. Probability of reentrancy vulnerability(V1) can be calculated using the Eq(4) as follows.

P(V1)=(P(P1)\*WP1)+(P(P4)\*WP4)+(P(P6)\*WP6)

= (0.25\*0.5)+(0.2\*0.5)+(0.3\*0.5)= 0.37

To calculate the severity of V2 (DOS vulnerability), directly use P9-CPT value for the chain of patterns P1--> P7--> P9 as shown in Bayesian Network. P6 is an independent event. P4 is not in the V2 list.

P(V2)= (P(P9|P7, P1)\*WP9)+(P(P6)\*WP6)

= (0.25\*4) + (0.08\*0.5)= 1.04 >1

P(V2)updated = 0.99 as per [Eq(3.2)](#Eq32)

P(P9|P7, P1) value can directly get from P9-CPT, which in turn is dependent on P7, which is dependent on P1. WP9 = 4/(4-3)=4, as per the Eq(1), where total number of levels(TL) in BN is 4 and pattern P9 is present at level(level-3).

Probability of tx.origin vulnerability(V3) can also be calculated using Eq(4) as follows, by considering only the patterns P1, P4 and P6, other two patterns P7, P9 are not in the V3 list.

P(V3)= P(P1)\*WP1)+(P(P4)\*WP4)+(P(P6)\*WP6)

= (0.45\*0.5)+(0.15\*0.5)+(0.15\*0.5)= 0.36

By comparing the above probabilities, P(V2) is greater than the vulnerability deterministic threshold, hence the conclusion is, given smart contract is vulnerable of type DOS and there is no influence of other two vulnerabilities because probabilities of both V1 and V3 are less than the Vthreshold value as shown in the output in Table3 and as per the Eq(6). Output of the validation phase is also describing the reasons to have the vulnerabilities and suggestion to avoid the vulnerabilities (as per table4) in addition to the severity of vulnerable probabilities, so that it is possible to correct the given smart contract to make sure vulnerable free before deploying into the Blockchain.

Table3. Output from Testing phase

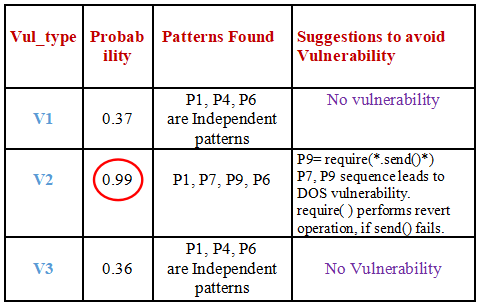
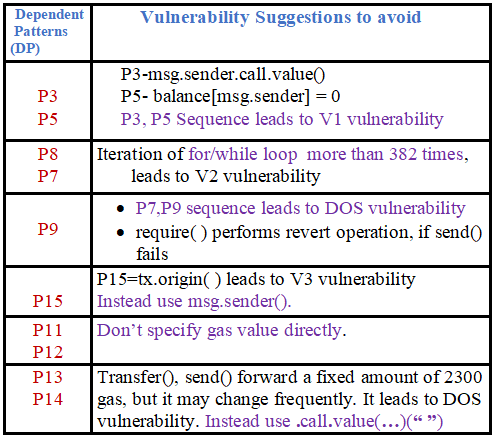
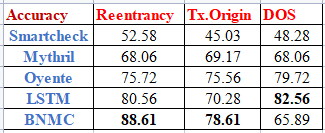


Table4. List of suggestions to avoid SC vulnerabilities [[11](#SCDataset)]

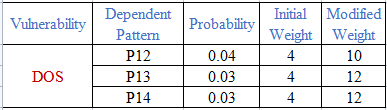


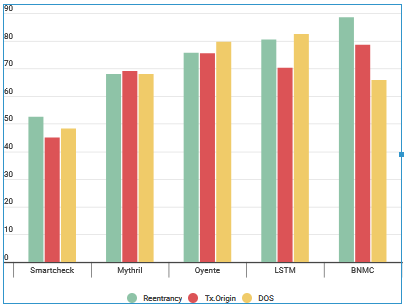
Evaluation of the proposed BNMC design is done on the test data set (new smart contracts) to detect classification accuracy for each vulnerability. Quality of the proposed model is measured by considering classification metrics which are a confusion matrix, precision, recall, and accuracy. The Proposed model results are compared with traditional vulnerability detection methods such as Smartcheck [31], Mythril [32], Oyente [33], and LSTM model [8]. Initially, Bayesian learning(BL) [13] was applied to detect SC vulnerabilities without combining with Bayesian networks, after that tested with Bayesian Networks to improve detection accuracy results. In the beginning of the experiment, the proposed model results got less accuracy for DOS vulnerability as shown in Table 5 and Figure11, since dependent patterns of DOS vulnerabilities have less probability.

**Table 5.** Accuracy Comparison between Proposed and traditional Models



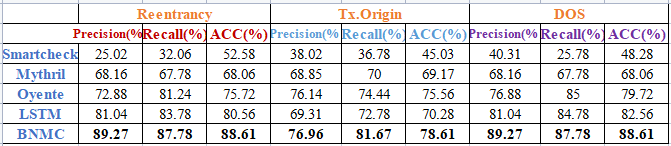
**Table6.** Updated Weight to increase the accuracy of DOS vulnerability.



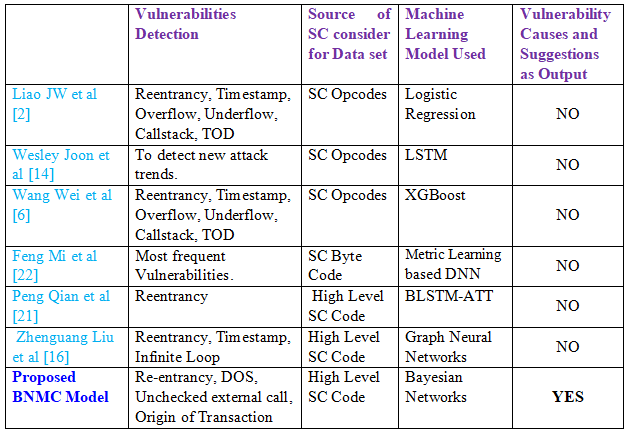


**Figure 11.** SC Vulnerability detection accuracy comparison with traditional methods.

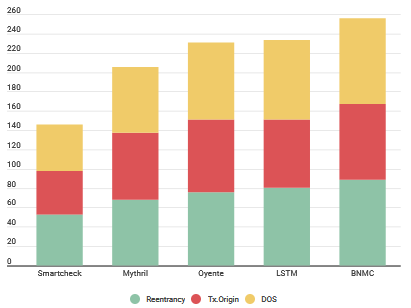
**Table 7.** SC Vulnerabilities detection metrics for Proposed and traditional Models



**Table 8.** Proposed BNMC Model comparison with Existing Models



Later BNMC model results improved by increasing the weight of dependent patterns for DOS vulnerability as per Eq (2). Equation(2) specifies that increase the weight of dependent patterns as much as its probability is low as shown in Table 6. Compared with the existing methods, the proposed BNMC design produced outperformed results to detect Reentrancy, DOS and Tx.origin vulnerabilities as shown in Table 7 and Figure 12. . The novelty of the proposed BNMC model can be observed from Table 8, as it specifies the causes or reasons for each vulnerability and makes suggestions to avoid vulnerabilities using Bayesian networks, in addition to detecting security vulnerabilities.



**Figure 12.** SC Vulnerability detection accuracy comparison with traditional methods.

**5. Conclusion & Future work**

In this work, BNMC design was proposed and implemented to detect smart contract vulnerabilities. In Ethereum Blockchain all transactions get completed by following the rules defined in a smart contracts. Vulnerable smart contracts leads to loss of money for users by the attackers. Prior identification of vulnerabilities in smart contracts is essential task to avoid attacks. Proposed BNMC design was implemented, tested on new smart contracts and its results are showing improved vulnerability detection accuracy compared with traditional techniques since proposed model considers key patterns causes for vulnerabilities, pattern sequences, their probabilities and expert knowledge. Compared with other models, proposed model specifies reasons to have each vulnerability and suggestion to avoid vulnerabilities. The proposed model can able to detect only security vulnerabilities which are reentrancy, DOS and tx.origin. Detection of other smart contract vulnerabilities using Bayesian Networks and automation of the Bayesian Network construction is our future work.

**References**

[1] Ajienka, N., Vangorp, P., & Capiluppi, A, An empirical analysis of source code metrics and smart contract resource consumption, Journal of Software: Evolution and Process, 32, 2020.

[2] Tonelli, R., Destefanis, G., Marchesi, M., & Ortu, M, Smart Contracts Software Metrics: a First Study. ArXiv, abs/1802.01517, 2018.

[3] Giuseppe Antonio Pierro, Roberto Tonelli, Michele Marchesi, An Organized Repository of Ethereum Smart Contracts’ Source Codes and Metrics, Future Internet, MDPI, 2020.

[4] N'Da AAK, Matalonga S, & Dahal K, Applicability of the software Security code metrics for Ethereum smart contract using Solidity. International Conference on Deep Learning, Big Data and Blockchain (DEEP-BDB), 2021.

[5] Marco Comuzzi, Cinzia Cappiello, Giovanni Meroni, An Empirical Evaluation of Smart Contract-Based Data Quality Assessment in Ethereum, 2021.

1. Pinna, A.; Ibba, S.; Baralla, G.; Tonelli, R.; Marchesi, M. A Massive Analysis of Ethereum Smart Contracts Empirical Study and Code Metrics. IEEE Access, 7, 78194–78213, 2019.
2. Pierro, G.A.; Tonelli, R. PASO. Conference Proceedings on Object-Oriented Programming Systems, Languages, and Applications. 2020. Available online: <https://ieeexplore.ieee.org/document/9050263> (accessed on 26 Sept 2022).
3. Comuzzi, M., Cappiello, C., Meroni, G, An Empirical Evaluation of Smart Contract-Based Data Quality Assessment in Ethereum. Internation Conference on Business Process Management: Blockchain and Robotic Process Automation Forum. PP: 51-66, Springer, 2021.

[9] Damian Rusinek, Paweł Kuryłowicz, SMART CONTRACT SECURITY VERIFICATION STANDARD, Accessed on: Sept 23, 2022. [Online] Available: *[https://securing.github.io/SCSVS/SCSVS\_v1.1.pdf](https://analyticsindiamag.com/a-complete-guide-to-lstm-architecture-and-its-use-in-text-classification/.)*[.](https://analyticsindiamag.com/a-complete-guide-to-lstm-architecture-and-its-use-in-text-classification/.)

[10] Ethereum Smart *Contract* Best Practices, (2022), Accessed on: Sept 7, 2022. [Online] Available: *[https://consensys.github.io/smart-contract-best-practices/known\_attacks/](https://consensys.github.io/smart-contract-best-practices/known_attacks/.)*[.](https://consensys.github.io/smart-contract-best-practices/known_attacks/.)

[11] S. Kalra, S. Goel, M. Dhawan, and S. Sharma, “ZEUS: Analyzing Safety of Smart Contracts,” no. February, 2018.

[12] Smart Contract Dataset, (2020), Accessed on: Sept 7, 2022. [Online] Available: *[https://swcregistry.io/ docs/SWC-107#modifier-reentrancy-fixed sol](https://github.com/shaunazzopardi/bpmn-to-solidity.)*[.](https://github.com/shaunazzopardi/bpmn-to-solidity.)

[13] Codetta-Raiteri D, Editorial for the Special Issue on Bayesian Networks: Inference Algorithms, Applications, and Software Tools. Algorithms, 14, 138, 2021.

[14] Tom M Mitchell, In *Machine Learning, McGraw Hill*, Chapter-6, pp. 177-184, 1997.

[15] Daniel Kottke, Marek Herde: “Toward optimal probabilistic active learning using a Bayesian approach”; In Machine Learning, 2021, PP: 110:1199–1231.

[16] [Satyabrata\_Jena](https://auth.geeksforgeeks.org/user/Satyabrata_Jena/articles), Accessed on: Sept 26, 2022; [Online]. Available: [https://www. geeksforgeeks.org/measuring-software-quality-using-quality-metrics/](https://www.geeksforgeeks.org/measuring-software-quality-using-quality-metrics/).

[17] Gsalzer, Smart Contract Dataset, (2022), Accessed on: Sept 7, 2022. [Online] Available: *[https://github.com/smartbugs/smartbugs](https://github.com/smartbugs/smartbugs.)*[.](https://github.com/smartbugs/smartbugs.)

[18] S. Tikhomirov, & E. Voskresenskaya, Smartcheck: Static analysis of ethereum smart contracts, In *WETSEB*, 2018, pp. 9–16.

[19] B. Mueller (2017). ConsenSys/Mythril. Accessed on: May 7, 2022; [Online] Available: *<http://github.com/ConsenSys/>mythril*.

[20] Meon.fund, Oyente, (2020), Accessed on: May 7, 2022; [Online]. Available: *https://github.com/ melonproject/ oyente*.

[21] Eunjeong Park, A Bayesian Network Model for Predicting Post-stroke Outcomes With Available Risk Factors, In *Frontiers in Neurology*, volume 9, 2018.

[22] Daniel Kottke, Marek Herde: “Toward optimal probabilistic active learning using a Bayesian approach”; In Machine Learning, 2021, PP: 110:1199–1231.

[23] Zhao, Wei and Luo, Zeju, Web Text Data Mining Method Based on Bayesian Network with Fuzzy Algorithms, [Journal of Intelligent & Fuzzy Systems](https://content.iospress.com/journals/journal-of-intelligent-and-fuzzy-systems), vol. 38, no. 4, 2020, pp. 3727-3735.

[24] Chen, Yong et al. A Bayesian Network Structural Learning Algorithm for Calculating the Failure Probabilities of Complex Engineering Systems with Limited Data, [Journal of Intelligent & Fuzzy Systems](https://content.iospress.com/journals/journal-of-intelligent-and-fuzzy-systems), vol. 42, no. 3, pp. 1991-2004, 2022.