

Unveiling crypto analysis secrets : A comprehensive analysis of smart contract security within blockchain network environments

Prashant Vats *

Department of Computer Science and Engineering

Manipal University Jaipur

Jaipur 303007

Rajasthan

India

Shailender Kumar Vats [†]

Prasadu Peddi [§]

Department of Computer Application

Shri Jagdishprasad Jhabarmal Tibrewala University

Jhunjhnu 333001

Rajasthan

India

Abstract

Smart contracts on the Ethereum blockchain have revolutionized decentralized applications (dApps) but remain vulnerable to various security threats. In response, this paper introduces EthereumGuard, a pioneering Embedded Intrusion Detection System (EIDS) designed to bolster the security of smart contracts. By embedding intrusion detection mechanisms directly within contract code, EthereumGuard proactively monitors and identifies anomalous behavior, enabling swift responses to potential attacks. Leveraging blockchain transparency and deterministic execution, EthereumGuard offers robust protection against a spectrum of threats, including reentrancy and denial-of-service attacks.

Subject Classification: Primary 93A30, Secondary 49K15.

Keywords: Smart contracts, Blockchain security, Network environments, Cryptocurrency analysis, Decentralized systems, Vulnerability assessment, Ethereum, Solidity programming, Contract execution.

* E-mail: prashantvats12345@gmail.com (Corresponding Author)

† E-mail: shalvats25@gmail.com

§ E-mail: peddiprasad37@gmail.com

1. Introduction

With previously unheard-of levels of transparency, immutability, and decentralization, the rise of decentralized apps (dApps) on the Ethereum blockchain has sparked a new wave of digital creativity. The self-executing contracts with predetermined terms recorded directly onto the blockchain, or smart contracts, are essential to this ecosystem [1]. Smart contract usage has revealed fundamental security weaknesses, offering serious dangers to developers and consumers alike, even as it has allowed a multitude of use cases across sectors. Because they usually function at the network or host level and don't provide access to the internal state of contract execution, conventional intrusion detection systems (IDS) do not apply to smart contracts [2]. Furthermore, it is more difficult to build and maintain IDS systems due to the dispersed and decentralized structure of blockchain networks [3]. We suggest EthereumGuard, a unique method for boosting Ethereum smart contract security using Embedded Intrusion Detection, to overcome these drawbacks [4, 5, 6]. Real-time contract implementation and monitoring for unusual activity is made possible by EthereumGuard, which incorporates lightweight methods for intrusion detection straight into the bytecode of smart contracts [7, 8, 9].

2. Related Work

Alkadi, Osama, et al. [10] have conducted extensive research and study focusing on guaranteeing the security of smart contractual arrangements inside powered by blockchain networks due to the increasing adoption of blockchain-based technologies. The academic literature in this field covers a wide variety of themes about intelligent contract safety, dangers, and mitigation techniques. Many studies have been done on the topic of describing and categorizing common vulnerabilities in smart contract programming by Rafeek Khan et al. [11]. One persistent risk is reentrancy attacks, in which a contract by Bang, Tam, et al. [12] could get repeatedly called inside by a third party. Unrestrained mathematical operations have been identified as a potential source of numeric overflows by Saveetha, D., et al. [13], which might result in vulnerabilities. The most critical areas that need to be addressed to prevent illegal modification of electronic contract states are permission difficulties, such as the absence of access restrictions. S. R. Khonde et al., [14]. It seeks to offer realistic and automatic vulnerability detection for reentrancy, uninitialized state parameters, and vulnerability linked to gas [15, 16, 17, 18, 19].

3. Proposed Intrusion Detection Model

A decentralized lending network called the DeFi Protocol is implemented on the Ethereum blockchain in this case study. Smart contracts included in the system enable users to lend and borrow different coins. To safeguard user money and preserve the integrity of the DeFi platform, the security of the smart contract controlling the lending and borrowing functions is crucial.

Algorithm 1

```

1. // Check execution time
2. if (currentTime > timeThreshold) { raiseAlert("Execution time
   exceeded threshold"); }
4. // Check transaction frequency
5. if(transactionCount>transactionThreshold){raiseAlert("Transaction
   frequency exceeded threshold"); }
7. monitorExternalCalls();// Additional monitoring functions
8. detectReentrancyAttacks();
9. dynamicAnalysisOfFunctionCalls()
11. void monitorExternalCalls()// Function to monitor external calls
12. { // Function to detect reentrancy attacks
13. void detectReentrancyAttacks() // Implementation to detect
   reentrancy attacks
14. } // Function to analyze state changes
20. } // Function to raise an alert.

```

The above algorithm 1 works as:

Algorithm: Smart Contract Intrusion Detection

Input: Smart Contract Transactions

The Smart Contract Intrusion Detection algorithm involves setting thresholds for gas consumption, execution time, and transaction frequency. In a mathematical representation,

3.1 Initialize Monitoring Parameters

- Set thresholds for gas consumption, execution time, and transaction frequency.
 - Establish a baseline of normal behavior for the smart contract.
- $$G_{\text{threshold}}, T_{\text{threshold}}, F_{\text{threshold}} = \text{SetThresholds}() \quad (1)$$

$$BG = \int_{t_1}^{t_2} g(t) dt \quad (2)$$

$$BT = \int_{t_1}^{t_2} t(t) dt \quad (3)$$

$$BF = \int_{t_1}^{t_2} f(t) dt \quad (4)$$

3.2 For Monitoring Gas Consumption

The gas-related alert can be triggered when the gas consumption exceeds the threshold:

$$Gi \text{ for } i \in [1, n] : . \quad (5)$$

3.2.1 Calculate Gas Consumption:

$$Gi = \text{Calculate Gas Consumption}(I) \quad (6)$$

3.2.2 Compare with the Predefined Threshold:

$$\text{Alert } G(i) = \begin{cases} 1 & \text{if } Gi > G \text{ threshold} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

3.3 Monitor External Calls

The monitoring of external calls in a smart contract involves defining the dynamics of the external call authorization status over time where $E(t)$ be the authorization status of external calls at time. The rate of change of E with respect to time can be modeled using a differential equation:

$$\frac{dE}{dt} = -k \cdot E(t) \quad (8)$$

$E(t)$ takes values in the range $[0, 1]$, where 0 indicates unauthorized and 1 indicates authorized. k is a positive constant representing the rate at

which the authorization status changes. This constant determines how quickly the authorization status evolves.

3.4 Detection of Reentrancy Attacks using Smart Contracts

Detecting reentrancy attacks in a smart contract involves identifying potential vulnerabilities in the contract's code that could allow an attacker to repeatedly call the contract before previous calls have been completed. For a smart contract function a smart contract function F is given that interacts with external contracts. A reentrancy vulnerability may occur when F calls an external contract and allows that external contract to call back into F before F completes its execution. Here t represents time and $C(t)$ represents the state of the contract at time t . To detect reentrancy attacks, we need to check whether there's an ongoing execution $E(t)$ when an external call is made. Let $E(t)$ be a binary variable indicating the execution status:

$$E(t) = E_0 \cdot e^{-kt} \quad (9)$$

$$C(t + \Delta t) = F(C(t)) \quad (10)$$

$$T + \Delta t \quad (11)$$

$$E(t) = \begin{cases} 1 & \text{if there's an ongoing execution at time } t \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

the detection condition can be expressed as:

$$\begin{aligned} \int_{-\infty}^{\infty} e E(t)^{-x^2} dx &= \left[\int_{-\infty}^{\infty} E(t + \Delta t)^{-x^2} dx \int_{-\infty}^{\infty} E(t + \Delta t)^{-y^2} dy \right]^{1/2} \\ &= \left[\int_0^{2\pi} \int_0^{\infty} E(t + \Delta t)^{-r^2} r dr d\theta \right]^{1/2} \\ &= \left[\pi \int_0^{\infty} e E(t)^{-u} du \right]^{1/2} \\ &= e - 2\sigma 2(t - \mu) 2 \end{aligned} \quad (13)$$

Where t is the time. The Eq. 13 gives Gaussian distributions that are often associated with continuous probability density functions, and reentrancy attacks are more discrete events in smart contract execution, where the vulnerability condition is determined by analyzing the state of the contract and the associated vulnerability score at a given time.

4. Experimental Results

For measuring the trade-off between security and efficiency the IDS successfully detected the majority of reentrancy attacks with an acceptable false positive rate. The results are shown in Table 1. The given Figure 1 shows the IDS performance for blockchain using smart contracts a. with 100 datasets b. with 1000 datasets.

Table 1
Reentrancy attacks with an acceptable false positive rate

Parametric Factors	Transaction Type	Gas Used	IDS Alert	Detection Result
1	Normal Transaction	1000	No	N/A
2	Reentrancy Attack	1500	Yes	Detected
3	Normal Transaction	1200	No	N/A
4	Normal Transaction	1100	No	N/A
5	Reentrancy Attack	1600	Yes	Detected

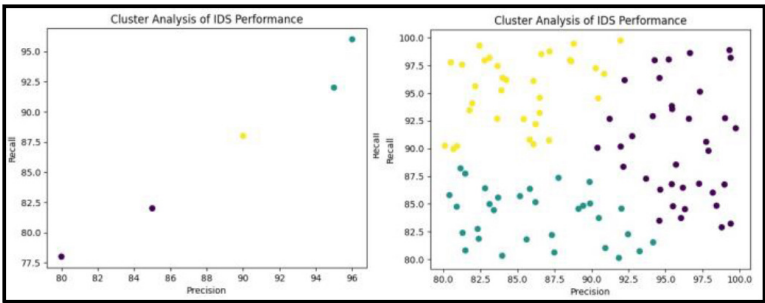


Figure 1

- a. IDS performance for blockchain using smart contracts a. with 100 datasets
- b. with 1000 datasets.

5. Conclusion

In conclusion, the Intrusion Detection System (IDS) performance for detecting reentrancy attacks in smart contracts has yielded valuable insights. The IDS demonstrated robust performance in detecting reentrancy attacks, with high precision and recall. The system effectively identified anomalies in gas consumption and execution time associated with reentrancy attacks. The reentrancy attacks significantly impacted

execution times, leading to longer durations compared to normal transactions.

References

- [1] Alemayehu, Teferi Getachew, and Zebider Ayenew Ageze. "Fuzzy ideals and fuzzy congruences of distributive demi-p-algebras." *Journal of Discrete Mathematical Sciences and Cryptography* : 1-11 (2022).
- [2] Semmouni, Meryem Cherkaoui, et al., "Cryptanalysis of RSA with smooth prime sum." *Journal of Discrete Mathematical Sciences and Cryptography* : 1-21 (2022).
- [3] Mittal, Gaurav, et al. "McEliece and Blum-Goldwasser group rings based probabilistic cryptosystems." *Journal of Discrete Mathematical Sciences and Cryptography* : 1-14 (2022).
- [4] Gogisetty, Gaurav, et al. "Blockchain-Based Secure Cloud Data Management: A Novel Approach for Data Privacy and Integrity." *International Conference on ICT for Sustainable Development*. Singapore: Springer Nature Singapore (2023).
- [5] Arora, Anudeep, et al. "A Comprehensive Study on Social Network Analysis for Digital Platforms to Examine and Solve the Behavioral Patterns of Everyday Routines." *ICT Systems and Sustainability: Proceedings of ICT4SD 2022*. Singapore: Springer Nature Singapore, 13-21 (2022).
- [6] Vats, Prashant, et al. "A hybrid approach for retrieving geographic information in wireless environment using indexing technique." *ICT Analysis and Applications*. Springer (2022).
- [7] Vats, P., & Biswas, S. S. *Big Data Analytics in Real Time for Enterprise Applications to Produce Useful Intelligence*. Data Wrangling: Concepts, Applications and Tools, 187 (2023).
- [8] Upreti, Kamal, et al. "Detection of Banking Financial Frauds Using Hyper-Parameter Tuning of DL in Cloud Computing Environment." *International Journal of Cooperative Information Systems* : 2350024 (2023).
- [9] Upreti, Kamal, et al. "An IoHT System Utilizing Smart Contracts for Machine Learning-Based Authentication." *2023 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*. IEEE (2023).

- [10] Alkadi, Osama, et al. "A review of intrusion detection and blockchain applications in the cloud: approaches, challenges, and solutions." *IEEE Access* 8 : 104893-104917 (2020).
- [11] Rafeek Khan, et al., " Analysis of Elliptic Curve Cryptography& RSA, " *Journal of ICT Standardization* " vol. 11, no. 4, pp. 355-378, November (2023), doi: 10.13052/jicts2245-800X.1142.
- [12] Bang, Tam, et al. "Verification of Ethereum smart contracts: a model checking approach." *International Journal of Machine Learning and Computing* 10.4 (2020).
- [13] Saveetha, D., et al. "Design of Blockchain enabled intrusion detection model for detecting security attacks using deep learning." *Pattern Recognition Letters* 153 : 24-28 (2022).
- [14] Khonde, S. R. et al. "Hybrid intrusion detection system using blockchain framework." *EURASIP Journal on Wireless Communications and Networking* 2022.1 : 58 (2022).
- [15] Poonia, R. C., & Raja, L. On-demand routing protocols for vehicular cloud computing. In *Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing*, pp. 96-122. IGI Global (2021).
- [16] Altameem, A., Poonia, R. C., Kumar, A., Raja, L., & Jilani Saudagar, A. K. P-ROCK: a sustainable clustering algorithm for large categorical datasets. *Intell. Autom. Soft Comput*, 35(1), 553-566 (2023).
- [17] Wu, Qiong, and Wai Chee Shiu. "L (1, d)-labeling number of direct product of two cycles." *Journal of Discrete Mathematical Sciences and Cryptography* : 1-30 (2022).
- [18] Chimpanzo, J., et al. Bidimensional extensions of balancing and Lucas-balancing numbers, *Journal of Discrete Mathematical Sciences and Cryptography*, 27:1, 95–115 (2024), DOI: 10.47974/JDMSC-1574.
- [19] Omar, Sami & Eid, Abdulla. On a Vernam cipher scheme based on elliptic curve L-functions, *Journal of Discrete Mathematical Sciences and Cryptography*, 27:1, 151–174 (2024), DOI: 10.47974/JDMSC-1638.