ETrace:Event-Driven Vulnerability Detection in Smart Contracts via LLM-Based Trace Analysis

Chenyang Peng[1], Haijun Wang[1], Yin Wu[1], Hao Wu[1], Ming Fan[1], Yitao Zhao[2], Ting Liu[1]
Xi'an Jiaotong University, Xi'an, Shaanxi
China
Yunnan Power Grid Co., Ltd, Yunnan
China

ABSTRACT

With the advance application of blockchain technology in various fields, ensuring the security and stability of smart contracts has emerged as a critical challenge. Current security analysis methodologies in vulnerability detection can be categorized into static analysis and dynamic analysis methods. However, these existing traditional vulnerability detection methods predominantly rely on analyzing original contract code, not all smart contracts provide accessible code. We present ETrace, a novel event-driven vulnerability detection framework for smart contracts, which uniquely identifies potential vulnerabilities through LLM-powered trace analysis without requiring source code access. By extracting fine-grained event sequences from transaction logs, the framework leverages Large Language Models (LLMs) as adaptive semantic interpreters to reconstruct event analysis through chain-of-thought reasoning. ETrace implements pattern-matching to establish causal links between transaction behavior patterns and known attack behaviors. Furthermore, we validate the effectiveness of ETrace through preliminary experimental results.

CCS CONCEPTS

• **Security and privacy → Software security engineering**.

KEYWORDS

Smart Contract, Event Detection, Large Language Models, Prompt Technique.

ACM Reference Format:
Chenyang Peng[1], Haijun Wang[1], Yin Wu[1], Hao Wu[1], Ming Fan[1], Yitao Zhao[2], Ting Liu[1] . 2025. ETrace:Event-Driven Vulnerability Detection in Smart Contracts via LLM-Based Trace Analysis. In *16th Asia-Pacific Symposium on Internetware: New Idea, June 20-June 22, 2025, Trondheim, Norway.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, blockchain technology has garnered widespread attention across various domains due to its characteristics of immutability, trustworthiness, and decentralization [1]. While decentralized exchanges (DEXs)[2] exhibit functional diversity across automated market making (e.g., Uniswap[3]) and hybrid architectures, their rapid proliferation has made them prime targets for malicious actors. In the NewFreeDAO incident($125M loss), attacker exploited a flash loan to purchase a large amount of NFD tokens, substantial rewards, and then repaid the flash loan. In the Omni NFT ($1.4M loss) hack exploits the reentrant vulnerability to hijack the contract execution process.

Current research primarily relies on code analysis tools (e.g., Slither, MythX) to detect vulnerabilities. However, most real-world DEX attacks involve third-party dependencies (e.g., oracle feeds, bridge contracts) whose source code is inaccessible.

To process this obstacle, we shift our focus to transaction logs. Due to the decentralized nature of blockchain and its tamper-resistant properties, it has garnered significant popularity. As a result, the trading volume of DEXs continues to grow steadily. As DEX trading volumes grow daily, each transaction is accompanied by the generation of transaction logs.

These logs capture event signatures that reveal both explicit invocation dependencies and implicit interaction patterns among smart contracts. Previous research has extensively explored the event information in transaction logs. For example, Kaleem et al.[4] utilized event information as the core communication mechanism to trigger contract logic, reducing the average total delay of event-triggered smart contracts by 2.2 to 4.6 times. Liu et al.[5] conducted an in-depth analysis of phantom event-related issues, addressing problems such as event forgery, inconsistent records, and contract impersonation. Meanwhile, Salzano et al.[6] employed logs to test smart contracts, aiming to compare log outputs with expected results. This demonstrates that, in the current context of frequent attack incidents, the event information of smart contracts can also provide a new perspective for vulnerability detection. However, such deficiency between event interpretation and attack causality severely limits detection accuracy.

To address these challenges, we propose ETrace, a framework that leverages event information extracted from transaction logs to infer the intent of smart contract transactions using LLMs.

As illustrated in Figure 1, ETrace first retrieves event information from smart contract transaction logs. Then, LLMs are employed as experts, processing characteristics of attack behaviors to analyze each event individually which mines the relationship between transaction behaviors and attack behaviors. Finally, events are compared with four predefined attack patterns through pattern matching to identify potential vulnerabilities in smart contracts. Our preliminary experiments have confirmed the effectiveness of ETrace. The main contributions of this paper are as follows:

- By utilizing large models to process features from events extracted by transaction logs and perform semantic intent analysis, we achieve a more precise understanding of the transaction behaviors of smart contracts.
- We propose a vulnerability detection framework that utilizes LLMs to semantically analyze events without source

Internetware: New Idea'25, June 20-June 22, 2025, Trondheim, Norway

ACM ISBN xxx-x-xxxx-xxxx-x/xxxx/xx…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

code, identifying four attack types: reentrancy, integer overflow, flash loan attacks, and DoS.
- Experiments are conducted to utilize a dataset of four real-world attack incidents, and the results successfully validate the effectiveness of ETrace.

2 MOTIVATION EXAMPLE

Through the study of event information from different attack incidents, we observed that the event characteristics of the following types of attacks exhibit high distinctiveness.

Table 1: Reentrancy Event

Event	Address	Value
Transfer	0x0eD7→0x5f2e	1.0000×10^{22}
Transfer	**0xE1E1→0x5f2e**	1.8966×10^{15}
Transfer	**0xE1E1→0x5f2e**	1.1235×10^{16}
Transfer	**0x5f2e→0xE1E1**	1.8966×10^{15}
Transfer	**0xE1E1→0x5f2e**	1.2964×10^{18}
Transfer	0x5f2e→0xE1E1	1.1235×10^{16}
Transfer	0xE1E1→0x5f2e	1.8641×10^{18}
Transfer	0x5f2e→0xE1E1	1.2964×10^{18}
Transfer	0x5f2e→0xE1E1	1.8641×10^{18}
Transfer	0x5f2e→0x0eD7	1.0030×10^{22}

As shown in Table 1, a reentrancy attack occurred on XSURGE[1] (Aug 15, 2021), where attacker 0x5f2e repeatedly invoked the victim contract (0xE1E1)'s sell and purchase functions. These emitted consecutive Transfer events, with alternating from/to addresses between 0x5f2e and 0xE1E1, forming a loop that enabled the exploit.

Table 2: Integer Overflow

Event	Address	Value
Transfer	0x09a3→0xb4D3	2^{255}
Transfer	0x09a3→0x0e82	2^{255}

Table 2 is an integer overflow incident that occurred on Beauty Chain[2] on April 22, 2018. Due to the input parameter exceeded the range of an unsigned integer, an integer overflow occurred, causing the value field to reach an unusually large magnitude (specifically, 2^{255}). This allowed the attacker to profit by 900 million dollars. The event characteristic of this case is clearly reflected in the abnormal value field.

Table 3 presents a flash loan attack on MEVBOT (Oct 14, 2022)[3]. Although MEVBOT lacks source code, event data shows the attacker transferred 1.8774×10^{20} to 0x88e6, conducted cyclic trades via 0xB4e1 and 0x8ad5, and authorized 1.1579×10^{77} tokens to 0xBA12 for arbitrage. Through multiple `Swap` operations, they gained profit and withdrew 1.8657×10^{20} via a `Withdrawal` event.

Table 4 shows a DoS attack on the GovernMental[4] contract. To claim a prize, the contract had to delete a creditor list, but due to repeated small ETH loans by malicious user 0x818d via `lendGM`, the list grew excessively. Deletion required 5,057,945 gas, exceeding the transaction limit of 4,712,388, causing 1,100 ETH to be permanently locked. Event logs confirm the attacker's repeated borrowing inflated gas cost and triggered the failure.

Table 3: Flash Loan Attack Event

Event	Address	Value
Transfer	0x2D00→0x88e6	1.8774×10^{20}
Transfer	0x0000→0xBA12	1.0000×10^{0}
FlashLoan	**0x0000→0xC02a**	1.0000×10^{0}
Transfer	0x8ad5→0x4b77	1.5825×10^{20}
Transfer	0xB4e1→0x4b77	1.6941×10^{19}
Approval	0x4b77→0xBA12	1.1579×10^{77}
Swap	None→None	out0=0, out1=0
Transfer	0x4b77→0xBA12	1.4707×10^{10}
Transfer	0xBA12→0x4b77	1.1379×10^{19}
Transfer	0x4b77→0xB4e1	2.1896×10^{10}
Transfer	0x4b77→0x8ad5	2.0454×10^{11}
Sync		in=4.6287×10^{13} $out = 3.5902 \times 10^{22}$
Swap	**0x4b77**	in=2.1896×10^{10}, out=1.6941×10^{19}
Swap	**0x4b77**	in=2.0454×10^{11}, out=0
Withdrawal	**0x4b77**	1.8657×10^{20}

Table 4: DoS Event

Event	Address	Value	Gas
lendGM	0x94bd→0xf457	0.01	**36855**
totalPayedOut()	0x490f→0xf457	0.0	21651
lendGM	0x818d→0xf457	0.001	**2532963**
lendGM	0x818d→0xf457	0.001	**5057945**
lendGM	0x818d→0xf457	0.001	**5057945**
lendGM	0x818d→0xf457	0.001	**5057945**
lendGM	0x818d→0xf457	0.001	**5057945**
Unknown Function	0x490f→0xf457	1.0	750000
Unknown Function	0x490f→0xf457	0.8236	750000
Unknown Function	0x490f→0xf457	0.01	750000

3 METHODOLOGY

This module provides a comprehensive overview of the processes and key technologies involved in ETrace. As illustrated in Figure 1, the architecture of ETrace is divided into four main stages. The process begins with Data Preprocessing, where transaction logs are extracted and transformed into Decoded Information, which is then sliced into events to ensure its suitability for analysis. Next, during the Prompt Engineering phase, the events are enriched with vulnerability conditions and passed to the Event Analysis stage. Here, the events are systematically analyzed, and a Comprehensive Judgment is made based on the event content to align with the required format and conditions. Finally, in the Pattern Matching phase, the analyzed data is evaluated against known vulnerability patterns to identify potential security risks associated with the current event.

3.1 Data Preprocessing

We retrieve the transaction log through the transaction hash, which typically contains the contract addresses and topic information for all emitted events. The topics include critical details such as the function signature, sender address, and receiver address within the transaction. By decoding the parameters in the topics using the ABI of the corresponding contract address, we obtain all function names and their associated parameters. The

[1]https://medium.com/@Knownsec_Blockchain_Lab/knownsec-blockchain-lab-comprehensive-analysis-of-xsurge-attacks-c83d238fbc55
[2]https://etherscan.io/tx/0xad89ff16fd1ebe3a0a7cf4ed282302c06626c1af33221ebe0d3a470aba4a660f
[3]https://etherscan.io/tx/0x35ecf595864400696853c53edf3e3d60096639b6071cadea6076c9c6ceb921c1
[4]https://www.reddit.com/r/ethereum/comments/4ghzhv/governmentals_1100_eth_jackpot_payout_is_stuck/

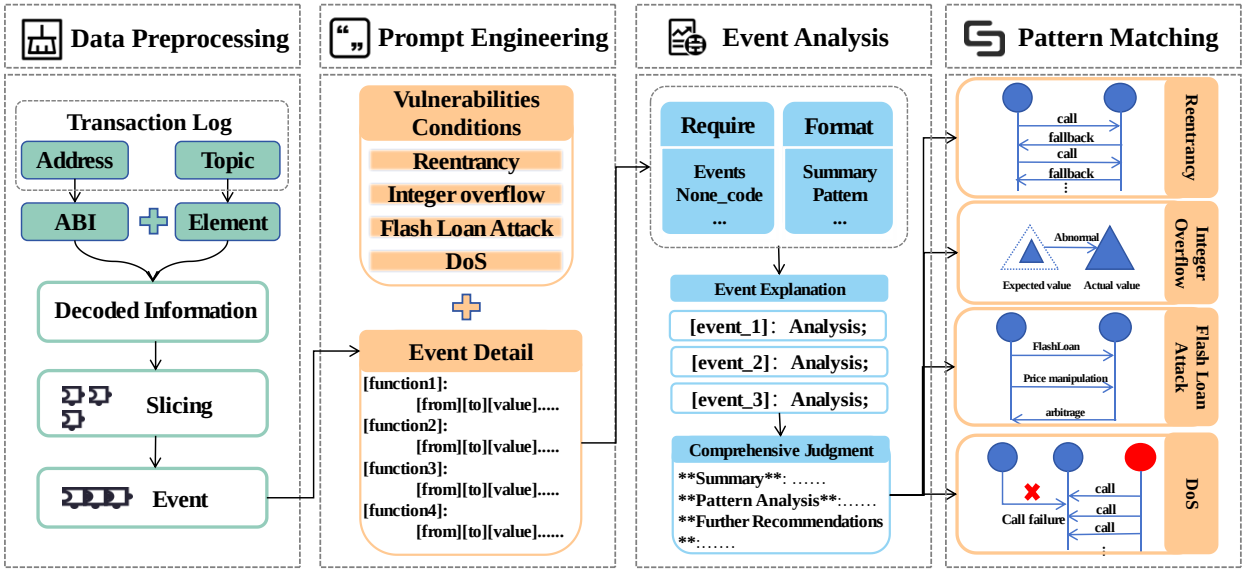

Figure 1: Architecture for ETrace.

decoded information is then segmented based on the function names, ultimately yielding fine-grained event information. We categorize event information, into three main components: function name, address, and value. The function name indicates which functions were called between contracts, while the address reveals how many contracts participated in the current behavior. The value reflects the quantitative measure of resources involved, such as tokens or ETH, associated with the behavior.

3.2 Prompt Engineering

In the prompt engineering phase, we integrate vulnerability conditions with event details. The vulnerability conditions encompass the characteristics of four known attack types: reentrancy, integer overflow, flash loan attack, and DoS. Taking reentrancy as an example, the vulnerability condition includes "*multiple calls to the same function during the contract execution, exploiting external calls before state updates, leading to malicious repeated fund transfers or tampering with the contract's state.*" Additionally, we employ the COT strategy in the prompts, instructing the LLMs to first explain the behavior associated with each event and explicitly output the reasoning process. Subsequently, the model evaluates the provided explanations holistically to generate a comprehensive judgment.

3.3 Event Analysis

In the event analysis phase, we specify the requirements and output format for LLMs to analyze events. Since the event analysis is conducted without relying on source code, we explicitly include a "None code" requirement . The initial output provides an explanation of each event, which aids in understanding the behavior of smart contracts. By interpreting these behaviors, we gain deeper insights into the details of how vulnerabilities occur. The subsequent output builds on the first, delivering a comprehensive evaluation of all event explanation. Ultimately, the model outputs a structured result in the format of "Summary - Pattern Analysis - Further Recommendation", identifying which specific vulnerabilities (or combinations thereof) the event matches. This step process ensures both granular and holistic analysis, enhancing the accuracy and interpretability of the results.

3.4 Pattern Matching

The pattern matching phase serves as a judgment to the pattern analysis results. To achieve more interpretable outcomes, we provide an additional explanation based on the identified results, aiming to elucidate the process through which the attack behavior occurs. Therefore, in the pattern analysis phase, the output not only identifies one or more vulnerabilities reflected by the event information but also analyzes and deconstructs the process by which these vulnerabilities arise. This ensures that the event definitively aligns with the pattern of the identified vulnerability. In Section 2, we provide a detailed explanation of four different types of vulnerabilities, specifically including reentrancy, integer overflow, flash loan attacks, and DoS, offering a comprehensive foundation for understanding their characteristics and behaviors.

4 EXPERIMENT

4.1 Dataset

We evaluated ETrace on four Etherscan-recorded attack events. For each, we analyzed both event-level explanations and the overall judgment. Accuracy was assessed by checking whether the outputs captured relevant attack behaviors and aligned with actual attack processes.

4.2 Performance

The result of the LLMs is shown in Table .5. Based on the results, LLMs are able to provide reasonable analysis and explanations of the vulnerabilities reflected by the events, including the causes of the attacks and the types of attacks. The experimental results demonstrate that LLMs can not only effectively parse fine-grained event content, aiding in the understanding of smart contract behavior, but also integrate multiple event details to make comprehensive judgments. Although the contextual relevance of LLMs may slightly decrease for longer event information due to token limitations, these results still validate the feasibility of our approach.

However, a limitation is that no threshold was set for the value, which caused LLMs to incorrectly identify integer overflows during the analysis of reentrancy events when the value was too high. This is an aspect we plan to improve in the future.

Event	Actual Attack	Comprehensive Judgment
XSURGE	**Reentrancy**	There are consecutive transfers between the same addresses (e.g., from address 0x5f2e to address 0xE1E1) with varying values. This could potentially be a **reentrancy risk** if the state is not updated correctly.
Beauty Chain	**Integer Overflow**	The large integer value is suspicious and could suggest potential **integer overflow**, especially in systems where such values exceed the maximum storage capacity for integers. Given that the value is extremely large, it should be examined for whether it exceeds the limits of the token's implementation.
MEVBOT	**Flash Loan Attack**	The FlashLoan event followed by the Swap events with abnormal price fluctuations could potentially indicate a **FlashLoanAttack event**.
GovernMental	**DoS**	The transaction with a value of 1.0 ETH and gas used of 750000 to an unknown function could potentially be a **DoS attack**. The abnormal gas used values in some transactions could suggest potential inefficiencies or issues in the contract code.

Table 5: Compehensive Judgement

5 RELATED WORK

Analysis Tools. Smart contract vulnerability detection is crucial, typically using three approaches: static analysis, dynamic analysis, and formal verification. Static analysis tools exemplified by Oyente [8] and Securify [9] employ symbolic execution and pattern-matching algorithms to scrutinize contract bytecode/-source code against predefined vulnerability signatures, effectively identifying risks such as reentrancy attacks and integer overflows. Dynamic analysis frameworks like Mythril [10] adopt a complementary approach by executing contracts in sandboxed Ethereum Virtual Machine (EVM) environments, enabling real-time detection of runtime vulnerabilities through transaction simulation and state monitoring. Formal verification systems represented by Sereum [11] leverage mathematical models to establish rigorous security guarantees, But their practical adoption faces challenges due to the significant manual effort for specification development and proof construction.

LLMs Analysis. As LLMs developed [13, 14] , it was gradually used to replace some manual work. At the same time, the integration of LLMs into smart contract security [15, 16] represents a novel and promising direction. However, LLMs exhibit hallucination tendencies, generating factually inconsistent or ungrounded outputs, which limits their applicability to unverified contracts. Recent researches [7, 12] have revealed that events in smart contracts demonstrate their valuable significance. This empirical evidence motivates our exploration of LLMs' pattern reasoning capability with event-driven behavioral semantics to overcome hallucination limitations.

6 CONCLUSION

To address the lack of source code access, we propose ETrace, a framework that uses LLMs and event data from transaction logs to detect smart contract vulnerabilities. ETrace performs semantic analysis of events, enhances interpretability of contract behaviors, and identifies four types of abnormal activities. We validate its effectiveness on four real-world attacks. In the future, we plan to expand ETrace by mining more event data to improve its versatility and accuracy.

7 ACKNOWLEDGMENT

This work was supported by National Key Research and Development Program of China (2022YFB2703500), and National Natural Science Foundation of China (62372367, 62272377).

REFERENCES

[1] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE international congress on big data (BigData congress)*. Ieee, 2017, pp. 557–564.
[2] S. Malamud and M. Rostek, "Decentralized exchange," *American Economic Review*, vol. 107, no. 11, pp. 3320–3362, 2017.
[3] Y. C. Lo and F. Medda, "Uniswap and the emergence of the decentralized exchange," *Journal of financial market infrastructures*, vol. 10, no. 2, pp. 1–25, 2021.
[4] M. Kaleem, K. Kasichainula, R. Karanjai, L. Xu, Z. Gao, L. Chen, and W. Shi, "An event driven framework for smart contract execution," in *Proceedings of the 15th ACM International Conference on Distributed and Event-based Systems*, 2021, pp. 78–89.
[5] Y. Liu, Y. Dong, Y. Liu, X. Luo, and Y. Li, "Phantom events: Demystifying the issues of log forgery in blockchain," *arXiv preprint arXiv:2502.13513*, 2025.
[6] F. Salzano and R. Pareschi, "A novel method based on log files for smart contract testing," in *2023 IEEE/ACM 6th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*. IEEE, 2023, pp. 1–4.
[7] C. Klinkmüller, I. Weber, A. Ponomarev, A. B. Tran, and W. van der Aalst, "Efficient logging for blockchain applications," *arXiv preprint arXiv:2001.10281*, 2020.
[8] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making smart contracts smarter," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 254–269.
[9] P. Tsankov, A. Dan, D. Drachsler-Cohen, A. Gervais, F. Buenzli, and M. Vechev, "Securify: Practical security analysis of smart contracts," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 67–82.
[10] N. Sharma and S. Sharma, "A survey of mythril, a smart contract security analysis tool for evm bytecode," *Indian J Natural Sci*, vol. 13, no. 75, pp. 39–41, 2022.
[11] M. Rodler, W. Li, G. O. Karame, and L. Davi, "Sereum: Protecting existing smart contracts against re-entrancy attacks," *arXiv preprint arXiv:1812.05934*, 2018.
[12] Y. Liu, Y. Dong, Y. Liu, X. Luo, and Y. Li, "Phantom events: Demystifying the issues of log forgery in blockchain," *arXiv preprint arXiv:2502.13513*, 2025.
[13] D. Kalla, N. Smith, F. Samaah, and S. Kuraku, "Study and analysis of chat gpt and its impact on different fields of study," *International journal of innovative science and research technology*, vol. 8, no. 3, 2023.
[14] X. Bi, D. Chen, G. Chen, S. Chen, D. Dai, C. Deng, H. Ding, K. Dong, Q. Du, Z. Fu *et al.*, "Deepseek llm: Scaling open-source language models with longtermism," *arXiv preprint arXiv:2401.02954*, 2024.
[15] Y. Wu, X. Xie, C. Peng, D. Liu, H. Wu, M. Fan, T. Liu, and H. Wang, "Advscanner: Generating adversarial smart contracts to exploit reentrancy vulnerabilities using llm and static analysis," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 1019–1031.
[16] H. Wang, Y. Hu, H. Wu, D. Liu, C. Peng, Y. Wu, M. Fan, and T. Liu, "Skyeye: Detecting imminent attacks via analyzing adversarial smart contracts," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 1570–1582.

Received 5 April 2025; revised 10 May 2025; accepted 20 April 2025