The Analysis of the Risks and Improvements of ERC20 Tokens

Changjin Zhang *

Cyber security, Drew University, NJ, US

* Corresponding author email: czhang3@drew.edu

Abstract. Blockchain is a decentralized, shared ledger that aggregates blocks of data in a factual data structure in chronological order using a chain structure. Two hundred sixty thousand tokens, or 98% of the around 2.5 million smart contracts in Ethereum network, are ERC-20 tokens. The ERC20 standard specifies constructors for token contracts that establish and initialize the contract state. The Ethereum wallet and Ethereum decentralized, centralized applications can access tokens through these standard interfaces. Security vulnerabilities in ERC-20 have drawn a lot of attention lately in recent years. This paper summarizes some basic security vulnerabilities and suggests avoiding multiple withdrawal attacks. Tokens can be used to support dApps, access blockchain services, trade, and obtain voting rights. Various tokens exist depending on their properties and use cases, including governance, utility, security, transaction, and platform tokens. Tokens can be sorted as fungible tokens and non-fungible tokens. Tokens that can be fungible are identical, divisible, and can be instead of money. Non-fungible tokens have a single owner and are unique.

Keywords: Blockchain; Ethereum; ERC20; Smart Contract; Multiple Withdrawal Attack.

1. Introduction

Blockchain is a decentralised, shared ledger that aggregates blocks of data in a factual data structure in chronological order using a chain structure. This concept was introduced to the public in Satoshi Nakamoto's 2008 article "Bitcoin: a peer-to-peer electronic cash system" An abstract asset built on the blockchain is called a token on the blockchain. It can be held as a representation of assets, money, or access rights. A smart contract is a part of software performed by a blockchain's virtual machine (VM) [1]. A set of functions that blockchain transactions are able to invoke make up a smart contract. Most smart contracts are created using a sophisticated specialised programming language like Solidity or Vyper and then deployed and executed on a blockchain virtual machine (VM).

Ethereum is a decentralized blockchain platform that runs and validates smart contract application code over a point-to-point network. Smart contracts allow participants to trade with one another without the need for a trustworthy government. Because trade records are chageless, confirmable, and securely disseminated via the network, participants have total control over the transaction data. Transactions can be remittance and collected by users' Ethereum accounts. The transaction must be signed by the sender, and the fee must be paid with an Ethereum token.

Many Decentralized applications (DApps) also create or employ their coins. These tokens could represent valuable assets, internal currencies, voting rights for DApps governance, or financial products. Meanwhile, these DApps accept and utilise the ETH at the protocol level and can be developed and deployed. The Ethereum community has approved the ERC-20 token standard, created expressly for fungible tokens, to facilitate interoperability with other DApps and online apps (exchanges, wallets, etc.) [2-4]. The ERC20 standard specifies an ordinary interface for token contracts consisting of mandatory and optional functions and features. Despite various extensions and replacements being proposed, ERC-20 remains the standard. Two hundred sixty thousand tokens, or 98% of about 2.5 million of smart contracts in the Ethereum network, are ERC-20 tokens. According to Nikolay Ivanov's data, managed ERC20 tokens make up approximately 90% of all ERC20 tokens and nearly 58% of all smart contracts.

In a smart contract, the terms of the agreement between the first party and second party are directly coded in a smart contract's line of code, it makes an automatically executed contract. The protocols and basic code are distributed using a decentralized blockchain network. Transactions are reversible and traceable, and their execution is managed by code. Lacking the need for a government, legal

system, or external enforcement mechanism, smart contracts can authorize trusted transactions and agreements between dispersed unnamed group. Developers can introduce subtle and unexpected bugs into smart contracts, just like any other software implementation. These flaws pose a potential risk to the security of smart contracts by allowing malicious money extraction from contracts or, in extreme cases, contract destruction. The "DAO attack" from 2016, the "Parity Wallet Hack" from 2017 and a stochastic number generation bug in Fomo3d and LastWinner are among of most critical security incidents.

The rest of this article is organized as follows. Section 2 provides some background on the ERC20 token standard, as well as a standard functional interface that can implement the basic transaction and token management operations of a token smart contract. Section 3 introduces and analyzes some basic security vulnerabilities of ERC20 Tokens, especially multiple withdrawal attacks, and some solutions. Section 4 is based on replaceable and non-replaceable tokens, and briefly describes the concepts, uses, and differences between them

2. ERC20 Tokens Standard

The creation of several new tokens generated by various users, carry out transactions via contracts, and are openly distributed on Ether is made possible by Ethereum-based token smart contracts. Token smart contracts allow for the creation of new tokens, and the production of several new tokens necessitates the creation of a single standard to govern them. As a result, the token smart contracts that generate tokens must follow the relevant standard protocols. Different Ethereum wallets for diverse projects and platforms can support tokens that follow the token protocol [5].

The ERC20 standard is currently the most widely used mainstream token standard for Ether, where ERC (Ethereum Request Comment) denotes the proposal solicitation of Ether. In addition to the ERC20 standard, other standard protocols expand upon the content of the ERC20 standard, such as ERC223, ERC27, etc. The ERC20 standard is used as the primary standard in this work to assess and research token smart contracts.

By sending and receiving tokens across accounts, examining the total amount of tokens created, and viewing the balance of tokens in an account at an address, token smart contracts can work like conventional cryptocurrencies according to the ERC20 standard. The Ethereum wallet and Ethereum decentralised centralised applications can access tokens through these standard interfaces because the ERC20 standard specifies constructors for token contracts that establish and initialise the contract state. The standard function defines the fundamental transactional and token management operations of a token smart contract, and the standard function interfaces are as follows [6].

(1) name (): Returns a token name of string type, e.g., "MyToken."

(2) symbol (): Returns the token symbol, which is also the short name of the token, e.g., "SMT."

(3) decimals (): Returns the number of decimal places the token can support or the number of decimal places the token can support. A token is divided into 100,000,000 copies if the value is set to 8 and 1,000 copies, or 0.001 representation, if the value is set to 3.

(4) totalSupply (): The total number of tokens issued can be accessed from using the function that returns the total number of tokens issued.

(5) balanceOf (): The quantity of tokens held in the account address supplied by the function can be determined by calling this function a balance lookup function.

(6) transfer (): This function is a transfer function that allows the token owner to send a token from his account to an account at the receiving address specified in the function parameter with the value specified in the function parameter.

(7) allowance (): The query authorised tokens function allows you to find out how many tokens the specified token holder has authorised for the specified token agent or how many tokens the specified authorised token agent account may also call.

(8) approve (): The target account can be granted the specified number of tokens in the holder's account through this function, which is a token approval and authorization function. The target

account will then have the right to use the specified number of tokens in the holder's account, and the tokens with the right to use will be transferred to the target account holder's account multiple times. The quantity of tokens that can still be called from the holder's account can also be seen using this function.

(9) transferFrom (): Together with the approve () method, this function serves as a transfer. The agent account with the ability to call tokens calls the transferFrom () function to transfer the tokens after the approve () function has approved.

The ERC20 standard specifies two event kinds in addition to the standard functions. An event is a sequence of actions taken when a smart contract is executed and is noted in the Ethernet virtual machine log. The successful completion of a coin transfer must cause the Transfer event to end, and the successful invocation of the Approve () function must cause the Approval event to be triggered.

3. The Security Vulnerabilities of ERC20 Tokens

This section discusses the potential impact on ERC-20 tokens and analyzes the Multiple Withdrawal Attack [7].

Arithmetic Over or Under Flows. In many programming languages, there is a well-known problem called integer overflow. In Solidity, an exception is not thrown at runtime for integer overflows. Using the SafeMath package, which replaces a+b with a.add(b) and throws an exception in the event of arithmetic overflow, will prevent this by design. There is no need to utilise the SafeMath library because Vyper includes built-in support for this problem.

Frozen Ether. Functions must be provided to withdraw deposited ETH because ERC-20 tokens, like user accounts, can receive and keep the cryptocurrency (including unexpected ETH). Incorrectly specified functions could cause an ERC-20 token to store ETH with no means to release it. Developers may request multiple signatures to withdraw ETH, if necessary.

Unchecked return values. Another strategy is to stop the assault at the API level, which involves altering, adding, and removing API functions. This strategy can successfully eliminate the Multiple Withdrawal Attack from happening. However, it differs significantly from the ERC20 standard technique. Existing ERC20 web applications and smart contracts might not be fit with the redefined method, and calling updated, and removed methods would result in errors. Some opcodes now have higher gas costs due to EIP-1884, which interferes with transfer (). Due to this, the community has advised using call. Value () depending on one of the aforementioned re-entrancy mitigations (i.e., Mutex or CEI). There is no need to validate the return value of the transmit () method because this problem is addressed in Vyper.

Multiple Withdrawal. Two ERC-20 functions, transferFrom () and approve () can be used to give permission for a third party to transfer tokens on someone else's behalf. These functions may be used in unfavorable circumstances (such as front-running or race conditions), which could lead to the transfer of more tokens than the owner intended by an unauthorised party acting maliciously. There are several ideas for extending the ERC-20 standard by including new functions (such as decreaseApproval () and increaseApproval()). However, the secure transferFrom () method is the best while still adhering to the standard's requirements [7]. Some methods are suggested for Multiple Withdrawal Attacks [8].1) Modification of the transaction process. The goal is to avoid violating the ERC20 standard by preventing the exploitation of vulnerabilities at the transaction level [9, 10]. One strategy is to block attacks at the user level. To prevent attackers from stealing tokens through preemptive runs, according to the ERC20 standard, approvers should first set the allowed value to 0 before making any changes. 2) Modify the API of ERC20. Some other strategy is to block API-level attacks by modifying, adding, or removing API functions. This strategy can successfully prevent multiple retraction attacks, but it differs significantly from the ERC20 standard technique. Existing ERC20 web applications and smart contracts may not be fit with the redefined methods, and calling the updated and deleted methods will result in errors

4. Fungible Tokens and Non-Fungible Tokens (NFT)

In recent years, non-fungible tokens became more popular as cryptocurrencies and blockchain games proliferated.

Assets known as tokens operate on another currency's blockchain using smart contracts. They can be used in various ways and kept in crypto wallets. Tokens can be used, among other things, as a form of value storage to support decentralised apps (dApps), access blockchain services, trade, obtain voting rights, etc. Depending on their properties and use cases, there are various sorts of tokens, including governance, utility, security, transaction, and platform tokens. Tokens come in two varieties: fungible and non-fungible, depending on their properties [11].

Tokens that can be fungible are identical, divisible, and can be used like real money. Non-fungible tokens have a single owner and are entirely unique. In blockchain games, they can stand in for things like collectables or real estate. When handled properly, both can be worthwhile investments. Table 1 shows the main differences between Fungible Tokens and NFT. (COINTELEGRAPH, n.d.)

Table 1. The Differences between Fungible Tokens and Noe-fungible Tokens

	Fungible tokens	Non-Fungible tokens
Main features	Divisible Non-unique Interchangeable	Indivisible Unique Irreplaceable
Real-world purposes	Payment system Store of value	Intellectual property Academic title Artwork Music composition Gaming Utility Asserts likes stocks, shares Access to a service i.e., a subscription
Technology used	Own blockchain	Built on another blockchain
Example of tokens	Bitcoin; Litecoin; ERC-20	ERC-721
Content stored	Value	Data

Fungible Tokens. Fungible Tokens are similar to conventional money. For instance, a $10 bill in your pocket and a $10 bill belonging to a buddy have the same value. A $10 bill is equivalent to two $5 bills. Bitcoin is fungible. All Bitcoins are equivalent to each other. That means no matter where you are or how it was issued, one bitcoin is always worth one bitcoin. Fungible tokens can function like actual currency due to their properties. They are therefore most frequently utilized for transactions in the cryptocurrency ecosystem. Additionally, because there is a probability that fungible tokens can appreciate in value over time, many people elect to buy them as investments. On numerous different blockchains, there are hundreds of different fungible tokens accessible. For fungible tokens, different blockchains have different requirements. However, given that this is where fungible tokens were initially created, Ethereum is setting the bar with its ERC-20 standard. And this standard is used by many well-known tokens, including Tether (USDT), USD Coin, Shiba Inu , Binance USD , Binance Coin , and HEX .

Non-Fungible Tokens. Non-fungible tokens are distinct, untransferable, indivisible, and irreplaceable, in contrast to fungible tokens. All NFTs of the same type are unique from one another. Each token is distinct, therefore exchanging them for one another would result in a loss of value. There is verifiable evidence that they are unique, and they are on a blockchain. Because each NFT only has one owner and a single ID, it is simple to distinguish one NFT from another in smart contracts. Here is a case from the real world. Each notable work of art is a rare and expensive thing. Even though they are both renowned and priceless works of art, you cannot replace one if it is stolen or damaged because there is only one in the entire globe. There are numerous applications for non-fungible tokens, particularly in blockchain gaming. They may signify ownership of these particular goods as well as a variety of other properties. An NFT, for instance, can be a piece of land, a weapon, an avatar, a skin, etc. in video games. The fact that players can own in-game things makes this idea ideal for gaming.

These goods have real-world worth because they are uncommon and hard to find. Since anyone can produce them, there are a wide variety of non-fungible tokens, ranging from works of art to game elements. The ERC-721 standard for Ethereum, which was established by the same group as the ERC-20 smart contract, served as their starting point. The most well-known NFT collections are Doodles, Meebits, Azuki, Crypto Punks, Bored Yacht Club, and others. Games like Decentraland, Axie Infinity, and The Sandbox are setting the bar for NFTs in gaming and the metaverse.

5. Conclusion

The ERC20 standard is currently the most widely used mainstream token standard for Ethereum. It specifies constructors for token contracts that establish and initialize the contract state. The Ethereum wallet and Ethereum decentralised centralised applications can access tokens through these standard interfaces. Security vulnerabilities in ERC-20 have drawn a lot of attention lately because this standard is used by 98% of coins on Ethereum now. By variation the trading steps or modifying the API of ERC20, multiple withdrawals can be avoided. Tokens can be used to support dApps, access blockchain services, trade, and obtain voting rights. Various tokens exist depending on their properties and use cases, including governance, utility, security, transaction, and platform tokens. Tokens can be sorted as fungible tokens and non-fungible tokens. Tokens that can be fungible are identical, divisible, and can be used as money. Non-fungible tokens have a single owner and are unique. In summary, this paper is based on the mainstream academic and market views on Ethereum ERC20 and a brief analysis of its rationale.

References

[1] Alpos, O., Cachin, C., Marson, G. A., et al. (2021, July). On the Synchronization Power of Token Smart Contracts. In 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS) (pp. 640-651). IEEE.

[2] Cui, W., & Gao, C. (2022). WTEYE: On-chain wash trade detection and quantification for ERC20 cryptocurrencies. Blockchain: Research and Applications, 100108.

[3] Ivanov, N., Guo, H., & Yan, Q. (2021, September). Rectifying Administrated ERC20 Tokens. In International Conference on Information and Communications Security (pp. 22-37). Springer, Cham.

[4] Ivanov, N., & Yan, Q. (2022). Decentralization Paradox: A Study of Hegemonic and Risky ERC-20 Tokens. arXiv preprint arXiv:2209.08370.

[5] Li, X. (2020). Research on formal verification techniques for Ethereum token smart contracts. Lecture Notes in Computer Science(), vol 12470. Springer, Cham.

[6] Vogelsteller, F., Buterin, V. (2015). Erc-20 token standard [J]. Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland. Ethereum Improvement Proposals.

[7] Rahimian, R., & Clark, J. (2021). TokenHook: Secure ERC-20 smart contract. arXiv preprint arXiv: 2107. 02997.

[8] Sun, J. L., Huang, S., Zheng, C. Y., et al. (2021, December). A Novel Method to Prevent Multiple Withdraw Attack on ERC20 Tokens. In 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS) (pp. 1-7). IEEE.

[9] Toderean, L., Antal, C., Antal, M., et al. (2021, October). A Lockable ERC20 Token for Peer-to-Peer Energy Trading. In 2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP) (pp. 145-151). IEEE.

[10] Wang, D., Feng, H., Wu, S., et al. (2022, October). Penny Wise and Pound Foolish: Quantifying the Risk of Unlimited Approval of ERC20 Tokens on Ethereum. In Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses (pp. 99-114).

[11] Andrea, K., 2022. The Difference Between Fungible and Non-Fungible Tokens (NFTs). Available at: https: // blog.udonis.co/blockchain/fungible-non-fungible-tokens. (Accessed on 8th Dec 2022) https:// Doi. org / 10.1007/978-3-031-07203-1_1.