Blockchain and Smart Contract Security

Kush Janani

Jarvis College of Digital Media and Computing, DePaul University

CSEC 440: Information Security Management

Andy Reeder

November 22, 2022

Blockchain and Smart Contract Security

As the technology is rapidly evolving, the attack surface area will also continue to expand. There have been innumerable uses of blockchain technology such as in payment system. While there are many resources which are available for the blockchain, there is very few contents focusing on the blockchain security. On October 7, 2022, over $100 million has been confirmed by the Binance team as having been stolen because of an on-chain breach, with $7 million of that amount being frozen. On blockchain networks, there have already been several security lapses, frauds, and hacks that have cost billions of dollars losses. These problems are damaging blockchain's credibility, along with increased government scrutiny to uncover criminal people exploiting the technology.

Overview of Blockchain Technology

"A blockchain is a distributed database or ledger that is shared among the nodes of a computer network" (Hayes, 2022). The ledger must be stored and maintained independently by each node in the blockchain network. The way the blockchain is created, each block is connected to the one before it. The blockchain can be accessed by anyone. When you want to delete a transaction, all the previous transactions should be removed. The blockchain is not stored in a single or multiple datacentres. Each miner will have its own copy of the entire blockchain.

Smart Contracts

Smart Contract can be used for the blockchain from a single function like cryptocurrency into a multifunction that can be used for deploying various applications based on decentralisation. It is a deal where two or more parties exchange digital assets. One of the parties is responsible for allocating digital assets when the contract is initiated. After allocation,

these are distributed to the parties which are involved with respect to a protocol that is predefined.

Distributed Ledger Technology

The broad term for data storage that is dispersed over several places is distributed ledger technology**.** The digital ledger of the blockchain includes security features built in to prevent changes to current blocks which makes it difficult for the hackers to change the contents of the network system. The blockchain network's nodes each keep their own copy of the ledger due to which it is more durable and alteration resistant. Single points of failure and the requirement to have faith in the central authority are eliminated by decentralization because the "official" copy of the ledger is not kept by a centralized organization on the network. Data (transactions and blocks) percolate across the network via hopping over numerous peer-to-peer connections, with each node in the network directly connected to a small number of peers. Blockchain technology is a ground-breaking innovation that can tackle several challenging issues. In many cases, a database or alternative technology is preferable to a blockchain. Although not all distributed ledger technologies (DLT) are blockchains, blockchain is an example of a DLT. There are numerous DLTs that are based on blockchain or have related goals but are not blockchains.

Issues in Distributed Ledger Technology

Data cannot be erased from the ledger and must be stored in perpetuity, which has an influence on scalability (impacts on protecting personal and sensitive data). Blockchains containing sensitive data must be properly safeguarded on each node in the network. Blockchain needs a consensus mechanism that can guarantee network synchronization and thwart malicious actors. Also, networking on a peer-to-peer basis is ineffective and expensive.

Blockchain and Smart Contracts Use Cases

Blockchain has become incredibly popular in the last ten years. However, blockchain applications now go well beyond cryptocurrencies. For many industries, including BFSI, healthcare, education, real estate, supply chain & logistics, and IoT, to name a few, blockchain technology is posing as a game-changer. Financial transactions are the first and most significant use case of blockchain. Cryptocurrency-integrated blockchains, like Bitcoin, are made to use their digital ledgers to record financial transactions without the need for a centralized authority. Because there is no central authority, Bitcoin and other blockchains may be able to conduct transactions more quickly and cheaply. For instance, a lot of financial organizations provide free transfers that take a few days or instant transfers that cost money. Since blocks are created every ten minutes, a transaction in Bitcoin is likely to be executed in the following block. Additionally, transaction costs in Bitcoin are often cheap and independent of the amount being transferred (unlike some bank fees)

Centralized Finance

The ambition to displace old, centralized banking systems was one of the main factors that led to the invention of Bitcoin. In fact, a headline about a bank bailout may be found in the first block of the Bitcoin network. This both proved that the block was created on a specific day and criticized the established financial system. In the past, the integrity and accuracy of the account’s ledger were maintained primarily by a single authority (banks and other financial institutions). The financial institution is in charge of keeping track of the money coming in and going out of customers' accounts and settling any disagreements. Because of this, users of traditional systems have limited visibility into them and are forced to rely on the financial institution's honesty and integrity.

Decentralized Finance

The use of financial systems built on top of blockchain technologies is known as decentralized finance. Important functionality is provided as code (smart contracts) that run

on top of the distributed ledger, as opposed to having a centralized entity manage the ledger. Some of the benefits are as follows.

Programmable

Code for smart contracts is executed on a decentralized digital ledger. They can execute intricate business logic in a software that takes advantage of all the built-in security features of blockchain since they are Turing-complete.

Interoperability

The blockchain is intended to function as a multi-layer ecosystem with numerous levels of abstraction. Since they are frequently not closely connected to the underlying infrastructure, this makes it simpler for various systems to interact with one another.

Immutable

All calls to these smart contracts are implemented as transactions, and smart contracts themselves are transactions that are added to the digital ledger as transactions. The immutability of the digital ledger is safeguarded by the blockchain, making the smart contract code and the data it uses impervious to hacking. In addition, operations are transparent, and all smart contract code is available for auditing because to blockchain's decentralization, which mandates that all nodes have full access to the data in the digital ledger.

dApps

Smart contract-based applications are referred to as "dApps" or "decentralized applications." Frequently, these dApps feature a standard web frontend (hosted on a traditional webserver or a decentralized storage service) and a backend that is made of smart contracts. Utilizing smart contracts to create functionality enables the development of completely decentralized systems with publicly accessible code that can be submitted to security audits and are impervious to modification (since a copy of the code is stored on the digital ledger).

Consumer Use Cases

Because cryptocurrencies are integrated into blockchains, they can store and transfer value. Platforms for smart contracts expand this functionality by enabling the creation of blockchain-based applications that are completely decentralized. Therefore, platforms for smart contracts are appropriate for implementing crowdfunding. The terms of the crowdfunding agreements can be implemented as smart contracts that store value and distribute it based on the rules laid out in the code. This ensures that donations are allocated equitably and in compliance with the rules of the crowdfunding system.

Supply Chain

Supply chain management is one area where blockchain technology is frequently used. Organizations may encounter a variety of problems in their supply chains, such as counterfeit or fraudulent parts, unlawful or unethical business operations, etc. A supply chain management blockchain gives each system or component a unique identification on the network. This component's status is updated on the blockchain as it progresses through the various supply chain steps. “For example, “ownership” of a particular component may be transferred on the blockchain at the same time that the physical component is handed over from the manufacturer to a shipper” (Walbroehl, 2022)

Ethereum

By market capitalization, Ethereum is the second-largest cryptocurrency behind Bitcoin. It is an open source, decentralized blockchain that supports smart contracts.
The cryptocurrency created by Ethereum miners as payment for work done to safeguard the blockchain is called ether. The "Ether" cryptocurrency created by Ethereum is distributed to miners as payment for operations made to safeguard the blockchain, much like Bitcoin. Over 1,900 different cryptocurrencies, including 47 of the top 100 by market capitalization, use

Ethereum as their platform. The Ethereum Virtual Computer (EVM), which is provided by Ethereum, is a decentralized virtual machine that can run programs utilizing participating nodes. In contrast to others like Bitcoin Script, the virtual machine's instruction set is Turing complete. Utilizing "Gas," an internal transaction price system, the network's spam is reduced, and resources are distributed.

Common Vulnerabilities to the Blockchain

Another crucial issue with blockchain security is the susceptibility of its endpoints. Users interact with the blockchain through electronic devices like computers and mobile phones, which serve as the endpoint of the blockchain network. Hackers can target devices and monitor user activity to steal the user's key. One of the most obvious blockchain security problems is this one.

.

Risk Area	Impact
Network Attacks	Low
Node Security	Medium
Centralized Integration	High
User and Personal Security	Very High

Figure 1. Types of Vulnerabilities

Network Attacks- The 51% Attack

The 51% attack, which has been around since the beginning, is the earliest known attack against a Proof of Work blockchain. It exists as a result of the structure of Proof of Work consensus, and it cannot be removed without some degree of system centralization. A 51% Attack makes use of the fact that Proof of Work is fundamentally a majority vote scheme. As they search for a valid version of the following block on the blockchain, miners

use their hashpower to "vote" on the blockchain. An attacker can legitimately win the vote and take control of the blockchain if they hold a majority of the "votes" or hashrate.

Node Security- Timestamp Hacking

Each node internally maintains the current network time. The average time of a node's peers, which is transmitted in the version message when peer nodes connect, is what determines this value. The nodes network time counter resets to the system time if the median time deviates from the system time by more than 70 minutes. By joining as many peers and sending false timestamps, an attacker may theoretically slow down or speed up a node's network time counter or even speed up the time of the surrounding network.

User and Personal Security- Phishing Attack

In the domain of conventional cybersecurity, the danger of phishing is well-known. It also pertains to Bitcoin and other blockchain systems' security. Every other online account is the same as a cryptocurrency exchange. An attacker might be able to access the user's exchange account and carry out transactions on their behalf if a user is tricked into divulging their credentials by a phishing email.

Common Vulnerabilities to Smart Contracts

Smart contracts can be vulnerable to many different kinds of threats. Some are the result of erroneous arithmetic calculations, some as a result of poor access controls. Others because of the contract logic or workflow's lack of enforcement

Classes of Vulnerabilities
Recursive Call Attack
Denial of Service with Block Gas Limit
Integer Overflow Attack
Transaction Order Dependence
Unprotected Selfdestruct
Timestamp Manipulation
Delegated Call to Untrusted Calee

Figure 2. Smart Contract Vulnerabilities

Recursive Call Attack

One of the primary characteristics of smart contracts is the capability to call and utilise the code of other contracts. Additionally, they deal with Ether and frequently send it to various addresses of outside users. The contract must submit an external call in order to call external contracts or deliver ether to an address. As a result, an attacker may intercept external calls and force the contract to run extra code, including calls back to itself, through a fallback function. Consequently, the contract's code execution was re-entered. Any time a contract sends ether to an unidentified address, a recursive call assault occurs. An attacker might then create a contract with malicious code in the fallback function and write it to an external address. Therefore, the malicious code will be executed whenever a contract sends ether to that address. The malicious code typically runs a function in the weak contract and carries out actions that were not intended by the developers.

Mitigation

There are a number of ways to try to protect Smart Contracts from potential recursive call problems. First, use the function transfer whenever possible when sending ether to external contracts (). Sending sufficient ether is therefore possible, but the recipient address or contract is unable to call another contract. Another method is to make sure that all state variables have updated before sending the ether (or making external calls). Putting any code that makes external calls to arbitrary addresses as the final operation of a function or piece of code. Adding a state variable during code execution to prevent potential recursive calls is another good way to prevent recursive call attacks.

Denial of Service with Block Gas Limit

A specific amount of gas is constantly needed to carry out a smart contract's functionalities. The amount of computation required to complete a transaction determines how much gas is used. The total number of transactions contained in a block cannot go over

the threshold set by the Ethereum network for each block that is mined. In Ethereum, the Block Gas Limit is dynamic. The new gas cap is chosen by a voting algorithm and the miners. An error message reading "Exceeds block gas limit" appears if someone attempts to make a transaction with a gas limit higher than the block gas limit.

Attack Vectors

By adding transactions with an extremely high gas price, an attacker can stop other transactions from being added to the blockchain. The attacker can accomplish this by issuing numerous transactions that each use the whole gas cap and have a gas price that is high enough to be included as soon as the following block is mined. No additional transactions will be included in the block if the attack is successful. Sometimes a hacker wants to stop transactions for a certain contract before a certain time. Also, if everyone pays out at once, the block gas limit can be reached. This outcome might be desired, but if an attacker establishes numerous accounts, each one must receive a very tiny refund. The gas cap could be exceeded by the cost of refunding in gas to the attacker's addresses, preventing the refund transaction.

Mitigation

It is advised to use pull over push for external calls in order to prevent a DoS attack. preventing the chance of receiving a significant payment without the participation of other players. If not, attackers will use this Smart Contract as a target. Finally, it's critical to ensure that nothing bad occurs while other transactions are being completed while awaiting the next iteration of payOut().

Integer Overflow Attack

This is the case with smart contracts' buffer overflow. When an arithmetic operation exceeds the limit or minimum size of a type, an integer overflow or underflow occurs. When a number is saved as an uint8 type, it is represented as an unsigned 8-bit number with a range

of 0 to 28-1. The flaw appears when an arithmetic operation tries to produce a numeric number that is either larger than the type's maximum (overflow) or smaller than its minimum (underflow) representable value.

Mitigation

To prevent overflow/underflow vulnerabilities, the current method is to use Safe Math Library by OpenZeppelin.

Transaction Order Dependence

This is also called as a front-running attack, racing condition, or both. Due to the design of the Ethereum blockchain, mining nodes can choose which transactions to include in the current block based on which transactions have paid the most gas fees.
Others with lower gas costs will be mined and propagated after these transactions, which will be mined first.

Mitigation

Using a commit-reveal approach is a workaround that prevents the content of a transaction from being revealed until after the transaction has been added to the block.
As a result, neither users nor miners are aware of the transaction content. However, the value of the transaction, which is likely the most significant asset, is not concealed by the commit-reveal approach.

Unprotected Selfdestruct

In order to end a contract, a self-destruct command is used to make the object unreachable from any further calls or access. Malicious parties can cause a contract to self-destruct and sometimes have all the Ethereum contained in it transferred to a chosen address before termination due to missing or insufficient access constraints**.**

Mitigation

This vulnerability is brought about via the misuse of balance. Because contract balance can be changed, it is advised that contract logic not be dependent on it. Unexpected balances could result in various security issues. Therefore, it is preferable to define precise values as self-defined variables. Therefore, a selfdestruct() call has no effect on those variables.

Timestamp Manipulation

Certain contracts, like timed token sales, need that you confirm the current date and time. Block.timestamp and block.number are examples of global namespace variables that can be used to calculate the time, but they are not impervious to external tampering. Malicious miners have the ability to change the blocks' timestamps. Although miners cannot set a timestamp that is less recent than the one before it (otherwise the block will be rejected), they can set one that is a little bit more recent, which can occasionally give a "first mover advantage" on specific contracts.

Mitigation

Block.timestamp shouldn't be used as a deciding factor to alter any state. Using block.number and predicting times by a period, such as a week, 10 seconds, and so on, could be a decent solution. As a result, a block number is implicitly supplied to alter the contract's state which makes it difficult for miners to influence the block number.

Delegate Call to Untrusted Calee

Similar to a message call, a delegatecall executes the code at the destination address within the framework of the calling contract. Because msg.sender and msg.value never change, a smart contract can dynamically load code from a different address while it is being executed. The storage, current address, and ether balance all still make reference to the calling contract. Since the code at the target address has complete control over the caller's

balance and the values in its storage, calling into untrusted contracts presents a risk for exploitation.

Mitigation

It is advised not to use the "library" keyword and libraries without states in smart contracts. Additionally, employing modifiers and defining bounds for the caller and callee's functions prevents an attacker from exploiting contract weaknesses.

Conclusion

Because blockchain technology is becoming more prevalent in the domains of business, finance, law, medical, and real estate, maintaining its security is crucial. The rapid growth of technology is allowing the attackers to easily expand their scope and exploit the blockchain technology.

References

1. Afreen, S. (2022, September 27). Why is Blockchain Important and Why Does it Matters? [2022]. Simplilearn. Retrieved October 11, 2022, from https://www.simplilearn.com/tutorials/blockchain-tutorial/why-is-blockchain-important
2. Hayes, A. (2022, September 27). Blockchain Facts: What Is It, How It Works, and How It Can Be Used. Investopedia. Retrieved October 11, 2022, from https://www.investopedia.com/terms/b/blockchain.asp
3. Guimaraes, G. (2017, May 25). *Reentrancy Attack On Smart Contracts: How To Identify The Exploitable And An Example Of An Attack…*. Retrieved November 21, 2022, from https://gus-tavo-guim.medium.com/reentrancy-attack-on-smart-contracts-how-to-identify-the-exploitable-and-an-example-of-an-attack-4470a2d8dfe4

4. Konstantopoulos, G. (2018, January 8). *How to Secure Your Smart Contracts: 6 Solidity Vulnerabilities and how to avoid them (Part 1)*. Medium. Retrieved November 21, 2022, from https://medium.com/loom-network/how-to-secure-your-smart-contracts-6-solidity-vulnerabilities-and-how-to-avoid-them-part-1-c33048d4d17d
5. @Nevesof. (2022, October 7). The #BNB Team Has Confirmed That over $100 Million Has Been Stolen as a Result of an on-Chain Hack, of Which $7 Million Was Frozen Fortunately, They Have Already Announced That It Has Been Working Well. #Bsc #Binance #Binancesmartchain. Twitter. https://twitter.com/RomuloNevesOf/status/1578353602724515842
6. Walbroehl, S. (2022). Blockchain & Smart Contract Security | SANS SEC554. SANS Institute. Retrieved October 11, 2022, from https://www.sans.org/cyber security-courses/blockchain-smart-contract-security