# Blockchain Security

*Marcel Njighe-Tezeh*

*Cybersecurity, Wilmington University*

## *Cyb8100: Cybersecurity Capstone*

*Professor Maureen Mcwhite*

*Sept 04, 2022*

**The Definition of the Problem**

Blockchain technology is often hailed as a technology that will change the world, and in many respects, it will. However, contrary to what many evangelists would have you believe, it is not always the all-encompassing answer to the world's security problems (Marr, 2021). While the adoption of blockchain has skyrocketed, so has the number of attacks compromising the security and trust in the ecosystem. Smart contracts and decentralized applications are paving the way to new business models and a revolutionary new autonomous incarnation of the internet. And security must be ingrained from day one.

Lack of regulation makes the blockchain dangerous and raises concerns about its cybersecurity models. Once more, this primarily affects Bitcoin and other value-based blockchain networks. But the reality is that it's an extremely volatile market, as many people who have just invested in Bitcoin or other cryptocurrencies for the first time have discovered to their cost. Scams and market manipulation are widespread because of the absence of governmental control.

Blockchain technology is the spine of Web3.0, crypto, NFTs, and the metaverse. Researching, developing, and implementing security measures to understand the security maturity of the blockchain technology and its distributed applications (dapps) is of the essence as their day-to-day importance has been fast growing. With this growth, this technology will become a target for hackers.

**Research Question**

Innovative technologies strive to influence cyber-security innovation from a scientific and practical standpoint. The use of blockchain technology may change or eliminate security paradigms that are based on formal models of access rights, models of computation, models of distributed computing, and or models with no specific theoretical underpinning at all.

A qualitative and exploratory viewpoint is required to examine and better comprehend the change before its full acceptance because there hasn't been much research on blockchain technology in the context of its security, with the assumption by many that the technology's security is full-grained.

The primary research question, which is based on the study objective, is:

* **Is blockchain technology secure, and is it the antidote to cyber-security in this modern, advanced technology era?**

The following two sub-questions have been developed to help this research answer the main research question.

* **What existing security vulnerabilities, weaknesses, and threats is blockchain technology susceptible to?**

From the lens of security, this question will help look at the blockchain technology attack surface and potential attack vectors by applying a four-factor security model to cover the security aspects of the blockchain ecosystem and its consensus protocols, smart-contracts programming, and third-party services, such as wallets, exchanges, and Oracles. And finally, the end user.

* **How can blockchain technology influence the innovation of cyber security models?**

For this thesis, it is essential to develop a certain grasp of the possibilities and cooperativeness that blockchain technology offers because it is still a relatively new concept and has numerous definitions. I will use a range of technical and non-scientific materials from the previous questions to develop findings geared toward security models.

**Justifying the Problem**

Blockchain networks are susceptible to fraud and hacking even though they provide a tamper-proof ledger of transactions. People with malicious intentions have successfully carried out several hacks and frauds over the years by manipulating known weaknesses in blockchain technology (IBM, 2021). The fundamental structure of how this young technology operates—using decentralization, consensus, and cryptography—guarantees that transactions are essentially tamper-proof. Nevertheless, throughout time, hackers have nonetheless discovered ways to take advantage of the system. Twelve cryptocurrency exchanges were compromised in 2019 (How Safe Is Blockchain? 2021).

Smart contracts are blockchain-based computer program code that execute themselves automatically to carry out all or a portion of an agreement (Levi et al., 2018). This agreement is embedded into the blockchain, which renders it irrevocable and immutable. They are typically used to automate the implementation of an agreement so that all parties are immediately certain of the outcome and no middlemen are required. A poorly written DAO smart contract that was instantiated on the Ethereum Blockchain to serve as a contract for users to vote on projects and funds to support and invest in potential profitable smart contract ventures led to a loss of $120 million to hackers (Asmakov, 2021).

The vulnerable parts of the multi-sig wallet library are within the initWallet () function and the "kill" function (Petrov, 2017). The purpose of implementing this sort of centralized architecture in a decentralized network is to save gas fees on deployment transactions when new wallets are created. A newly deployed multi-sig wallet can invoke the code from the already deployed library and not pay the extra gas fee on that EVM bytecode. However, because a new wallet that was created required interaction with this library, it created a single point of failure. This security issue led to a parity multisig wallet hack in July 2017.

The careful management of the risks connected to the blockchain is necessary for its adoption and functioning. The underlying architecture of every organization will eventually incorporate blockchain technology, which is more than just an application (Blockchain Security Risks for Financial Organizations | Deloitte US, 2017). A unique instance of distributed ledger technology is the blockchain protocol, in which the consensus protocol establishes a daisy-chain immutable ledger of all transactions shared among all participants.

Blockchain technologies present sophisticated technologies that companies must account for. These technologies expose institutions to threats that are like those connected to traditional enterprise processes. Since blockchain enables peer-to-peer value transfer without the requirement for a central middleman, value transfer dangers exist. The transferred value may take the shape of assets, identities, or data. Due to this new business model, the interacting parties are now subject to new risks that were previously controlled by central intermediaries.

**Deficiencies in What We Know**

Existing published research is limited to a few blockchain technology areas, making it difficult to determine industry-wide vulnerabilities. Existing research has a narrow scope, focusing on the top-level layers such as smart contracts, oracles, end-users, etc., and not the effect of attacks on the entire blockchain infrastructure. Existing research also fails to address and analyze real-world incidents that evolve around existing technology weaknesses.

Many of the vulnerability types that exist in conventional software are listed in the Common Weakness Enumeration (CWE) dictionary, which is available online. A crucial resource is CWE. It serves as the foundation for the kinds of vulnerabilities that many code-scanning technologies attempt to find. CWE does not, however, specifically address blockchain or smart contracts (Seifried, 2021).

Blockchain technology is barely ever mentioned in databases of security vulnerabilities. This means that the majority of blockchain adopters won't be knowledgeable of security upgrades unless they specifically follow vendor release notes. This lack of coverage, particularly by the Common Vulnerabilities and Exposures (CVE) database and the U.S. National Vulnerability Database (NVD), is a major issue because many big companies do not know about the vulnerabilities if they are not formally acknowledged.

For the simple reason that the industry is so young, the present crop of blockchain and smart contract code-scanning tools is not particularly developed. Many smart contracts are implemented without a security audit because there aren't any reliable methods for doing so, which is also a problem in the industry. This results in a lack of thorough experimentation across the entire blockchain ecosystem and models, a lack of systematic analysis of attacks on current enterprise infrastructure into other areas of crucial business infrastructure, and finally, a lack of comprehension of attack motivations and analysis of existing attacks.

The blockchain ecosystem is vulnerable to outside threats. Studies, however, have failed to look at reasons for attacking or how attacks are carried out. Due to a lack of access to case study reports on blockchain hacks, enterprises are unable to comprehend the nature of the threats in the technology ecosystem. There is no known way to estimate or determine real-world occurrences, and neither are there any state or federal laws in place to validate and record events for continuous innovation.

**Defining the Audience**

Policymakers, blockchain designers and developers, system administrators, network security architects, cyber security specialists, and IT experts with experience in cyber security can all benefit from the material in this study. This research will assist in identifying and separating security concerns from actual attacks on blockchain networks. Best practices for blockchain security and risk reduction.

**Research Approach**

This study seeks to close a knowledge gap by investigating how blockchain technology can affect cybersecurity innovation in the context of a distribution system's operation. Because the phenomena being studied are extremely complicated, abstraction is utilized to keep the study's scope within the theoretical framework. Specifically, very novel blockchain technology phenomena are examined in the context of the security of the blockchain distribution system.

The research statement problem will be approached by first demystifying blockchain by providing an overview of what blockchain is, what smart contracts and dApps are, and how blockchain works. Decentralization, consensus, and cryptography are used in the fundamental design of this blockchain technology to ensure that transactions are effectively tamper-proof. So, this study will define the security properties of the blockchain, such as consensus, immutability, and cryptography, as findings will come from some of these properties. Keeping within the security model for the blockchain ecosystem attack surface and potential attack vectors, this study will apply a four-factor security model to cover the security aspects of the blockchain ecosystem.

**Figure 1.0**A screenshot of a computer

Description automatically generated with low confidence

This includes its consensus protocols, smart-contract programming, and third-party services like wallets, exchanges, and Oracles. And finally, the end user. Within the four-factor security model, the initial sections of this study will look to define the scope of the blockchain technology attack surface and vectors. In this order, blockchain platform security proof of work and proof of stake protocol attacks will follow. Smart contracts will come in with an analysis of open Ethereum smart contracts on GitHub.

This research will look at the existing blockchain security hack use cases and the gaps in how blockchain adopters are failing to design, deploy, and enforce security measures to safeguard blockchain technology from cyber-attacks. In line with existing research on blockchain security as detailed in the statement of the problem, this research will address the gaps where they fall short.

**Case Studies: Root Cause Analysis of Security Incidents in the Blockchain Ecosystem**

The case study methodology mentioned in the research technique is used in this part to analyze the blockchain-related incidents. This study will examine instances that are publicly available and conduct a thorough analysis of prominent representative cases from among the known incidents. An overview of the results will be presented in a tabular style. The goal is to assist prospective users of this young technology in learning from past mistakes and avoiding similar problems.

According to Single Grain, investments in blockchain companies are set to outpace last year – with over 300 investments and over $800 million in funding in 2017. With all these investments going into blockchain technology, security protection is of paramount importance. Hence, learning from past errors that cost fortunes is needed.

**Research method**

This study will be derived from a mixed method of research that will utilize data collected by existing research done in this domain, like open blockchain security survey research, data from use cases like the DAO hack, parity smart contract breach, live smart contract repos on GitHub, and SWC registry data(a system for Smart Contract Weakness Classification and Test Cases) to develop findings on existing vulnerabilities in the blockchain architecture, smart contracts, and end users. Then, follow up next with an analysis of blockchains' existing weaknesses and vulnerabilities to validate the findings. Finally, provide a comprehensive view of security mitigation and recommendations for the findings.

**Background and Review of Literature**

Innovative technologies aim to have a scientific and practical impact on cyber-security innovation. Blockchain technology may alter or replace security paradigms. To assess and better understand the security maturity of blockchain technology in the context of safeguarding data storage and transmission through decentralized, trustless, peer-to-peer systems, systematic qualitative and quantitative viewpoints are needed.

The primary objective of this literature review, based on the study objective, is to review blockchain technology security issues by examining research on existing security vulnerabilities, weaknesses, and threats that blockchain technology is susceptible to and their influence on the principles of cyber-security.

While the security of blockchain topologies has received a lot of attention, few academics have considered a systematic security review and analysis of all blockchain components. By using a four-factor security model to cover the security aspects of the blockchain ecosystem and its consensus protocols, smart-contracts programming, end users, and third-party services, such as wallets, exchanges, and Oracles, this literature examines the blockchain technology attack surface and potential attack vectors.

**Demystifying Blockchain**

**What is blockchain?** A blockchain is a decentralized and distributed digital ledger that is tamper-evident and tamper-resistant, usually operating without a central repository or authority (Yaga et al., 2018). Blockchain is designed to be secure through built-in design mechanisms. It comes with three built-in security properties. Consensus, immutability, and security through cryptography.

 A crude but easy way to internalize the concept of blockchain is to think of it as a special kind of database where the data is spread and stored across geographically distributed machines without a single database administrator or a management layer.

It is called "blockchain" because data is stored in a series of blocks where each block is linked to its predecessor, almost like a chain. What makes blockchain unprecedented compared to other technologies is the combination of unique features it carries. When data is written to a block, the data is replicated to multiple nodes in a blockchain network. A node in the blockchain network may hold a copy of the entire ledger and coordinate with other nodes using specialized protocols to ensure that data is recorded in the manner intended.

**Blockchain App.** To illustrate these concepts of distributed and decentralized, Bitcoin is a cryptocurrency that uses blockchain technology, its network, and specialized protocols. When one sends or receives bitcoins to another person using a mobile wallet app, they are announcing to the "Bitcoin Network" that they have a transaction they wish to conduct. The blockchain ecosystem will verify the financial accounting and record the transaction on the block ledger.

**Smart contracts and dApps.** Back in 1994, Nick Szabo, a computer scientist, and legal scholar created the term "smart contract" and defined it as: "A smart contract is a computerized transaction protocol that executes the terms of a contract (Szabo 1994)." A smart contract is a computer program that is a collection of code and data that is stored and executed on the blockchain. They are deployed on the blockchain. By nature, any user can interact with a smart contract and execute its function. This distinguishing feature of smart contracts makes them vulnerable to attacks.

A deployed smart contract or a validated transaction cannot be easily changed because of the blockchain's immutability. A faulty smart contract may result in a significant financial loss (Andesta et al., 2020). Smart contracts are the building blocks that help users create decentralized business applications (dApps). Smart contracts are developed in blockchain-specific languages like Solidity. dApps are a new breed of applications that take advantage of the properties of the blockchain.

Bitcoin and Ethereum are the most common examples of web 3.0 dApps. Non-fungible tokens (NFTs), voting and identity management, DAO, and Defi, are specific use cases addressed by many upcoming decentralized dApps. Many industry-specific applications are also emerging in the fields of the legal, supply chain, and the Internet of Things.

**Figure 1.2: How blockchain works**

Diagram

Description automatically generated

A blockchain is a chain of blocks. A block is a collection of data structures that includes a header and a list of transactions. Transaction records are kept chronologically in blocks together with timestamps and a distinct reference number (i.e., hash) to earlier blocks (HACKETT, 2017).

**Transaction.** The very first block of a blockchain is known as the Genesis block. It does not contain a hash of the previous block. A user initiates a transaction by using wallet software. That transaction is signed with a private key.

**Verification.** The transaction is then broadcast to the block in a network to ensure that the information reaches every other node. The network is a peer-to-peer network of nodes. The miner nodes ensure that the transaction is not malicious and adheres to the process's basic rules.

**Structure.** Each block is identifiable by a unique 256-bit hash value generated by a message digest method. Each block contains a transaction or collection of transactions. The SHA256 message-digest algorithm is frequently employed (Mosakheil, 2018).

**Validation**. Before being incorporated into the blockchain, blocks are verified. Proof-of-Work (PoW), the most common consensus technique used to validate blocks, relies on miners deriving the answer to a mathematical puzzle from the block's header (Mosakheil, 2018).

**Mining.** The mathematical puzzle is solved by miners using Proof-of-Work or other consensus procedures, and the block is then certified. Finding an input to a cryptographic hash function that hashes lower than or equal to a predetermined target value is the essence of mining. Because the content to be hashed is somewhat altered at each repetition in the pursuit of a suitable hash, it is brute force.

**The Security Properties of Blockchain**

1. **Consensus**

 In a blockchain, there is no central governing entity that dictates or oversees what transactions are to be written to the blockchain. The decision is made through consensus among the participating nodes within the blockchain. The process of arriving at a consensus by the nodes is materialized through a specialized and robust protocol. The consensus algorithm is robust and able to handle corrupt nodes, colluding nodes, and even faults due to communication failures.

There are many types of consensus protocols prevalent in the blockchain ecosystem. The Proof of resource protocol family is a general term for the family of protocols that require nodes to provide evidence that they have spent personal resources when participating in the mining process. Such resources must not be abundant or free. For example, Bitcoin and Ethereum use proof of work protocols that require proof of computing. Miners will not be able to solve the puzzle without exerting their CPU power.

Proof of space or storage protocols requires expanding storage resources as used by SpaceCoin. Proof of burn protocols require spending crypto resources. Nodes in the Proof of State protocol family must invest in crypto tokens. These nodes are called validators. The higher the stake, the higher the probability of being selected as a validator. The design of fault-tolerant consensus protocols has been the subject of research for many decades. Such protocols fall into the family of protocols known as Byzantine Fault Tolerance, or BFT, protocols.

1. **Immutability**

Once a block of transactions is added to the blockchain, the data cannot be modified. One can continue to append new blocks, but the previous blocks are immutable. Why does that matter from a security perspective? Maintaining data integrity is one of the key elements of the CIA test of security. "CIA" stands for confidentiality, integrity, and availability. The data integrity goal requires that data not be tampered with maliciously or by error and hence should always remain trustable. So how does blockchain achieve immutability? Every new block added to the blockchain contains a hash of the previous block. A hash is a mathematical function that takes an input of variable length and returns an output of fixed length. The Blockchain uses the SHA-256 hashing algorithm, which means the hash is 256-bit length regardless of the input data.

1. **Cryptography**

Concerns about blockchain technology and cryptocurrencies are widespread across a variety of organizations. Users want to know that the best algorithms available are adequately protecting their assets, but certain organizations around the world are against that or impose their standards (Storublevtcev, 2019).

Blockchain security features, such as consensus and immutability, as well as many other features, rely on several cryptographic constructs.  Cryptographic keys and blockchain addresses are constructs used heavily in both the Bitcoin and Ethereum blockchains and smart contracts. Blockchain uses public and private key pairs for use cases such as encryption, proving the identity of a user, and non-repudiation. There are many algorithms historically used for public key cryptography. The blockchain uses elliptic cryptography, or ECC, to generate private and public key pairs.

**Blockchain Ecosystem Security Model**

1. **Blockchain architecture**

For blockchain to be valuable to real-life business applications beyond cryptocurrencies, several other players must step in to create a rich ecosystem.

**Figure1.3**

**Diagram

Description automatically generated**

**Figure 1.3, Blockchain architecture, based on Piscini, Guastella, Rozman, & Nassim, 2016**

 The foundation of this ecosystem is the blockchain platform. The platform includes a distributed ledger that stores and processes transactions across blockchain networks. The consensus layer provides the transaction validation capability through the consensus protocols.

Several different external services enable the blockchain platform to be used for end users and decentralized applications. Host wallet services offer users the ability to store their private keys and cryptocurrencies with a third-party provider. Smart contracts are the building blocks for creating decentralized apps, or dApps. They bring programmability and business logic into the world of blockchain. Lastly, there are the end users and business domain applications that utilize or consume everything blockchain has to offer.

1. **Security model**

Networks, information systems, computing techniques, and security and privacy are the four criteria used by the ACM Computing Classification System, which is extensively used in the field of computer science, to characterize technology maturity (Wang et al., 2016).

**Figure 1.4:** Threats can emerge from traditional and well-known attack vectors or due to a new breed of attacks unique to blockchain. All threats fall into the four-factor security model. Each factor comprises a set of security controls that work to protect the ecosystem.

1. Securing the foundations of the blockchain. This includes its transactions, distributed ledger, and consensus protocol.
2. Smart contract security. This factor includes secure design principles and secure coding practices, especially targeted for smart contracts and testing strategies to discover bugs before deploying contracts.
3. Securing the crucial ecosystem players, such as wallets, providers, exchanges, and Oracles.
4. End user and security control to protect the interactions of the users with the blockchain ecosystem.

**Attack surface and vectors**

The attack surface simply means the areas of a system that are vulnerable and can be exploited.

**Figure 1.4**

Graphical user interface, application

Description automatically generated

* The concentration of consensus power. Malicious miners and transaction validators can collude to manipulate the consensus protocols.
* Nodes on the blockchain network can become rogue, assume civil identities, and compromise trust in the blockchain. These weaknesses can cause a ripple effect and lead to 51% attacks and double spending attacks.
* Vulnerabilities in the blockchain platform can also lead to an attack on the promise of privacy that the blockchain technology is founded upon.
* Smart contracts are also vulnerable to attacks such as reentrancy attacks, front-running attacks, and denial of service attacks.
* Ecosystem enabler services such as wallets and exchanges can also be compromised due to phishing attacks, denial of service attacks, other insider threats, and conflicts of interests.
* The end user and client apps running on devices are also prone to well-known attacks such as malware infections, threats of private keys, and a host of other attacks that are possible due to application-level vulnerabilities also defined by OWASP Top 10.

**Blockchain Platform Security Issues**

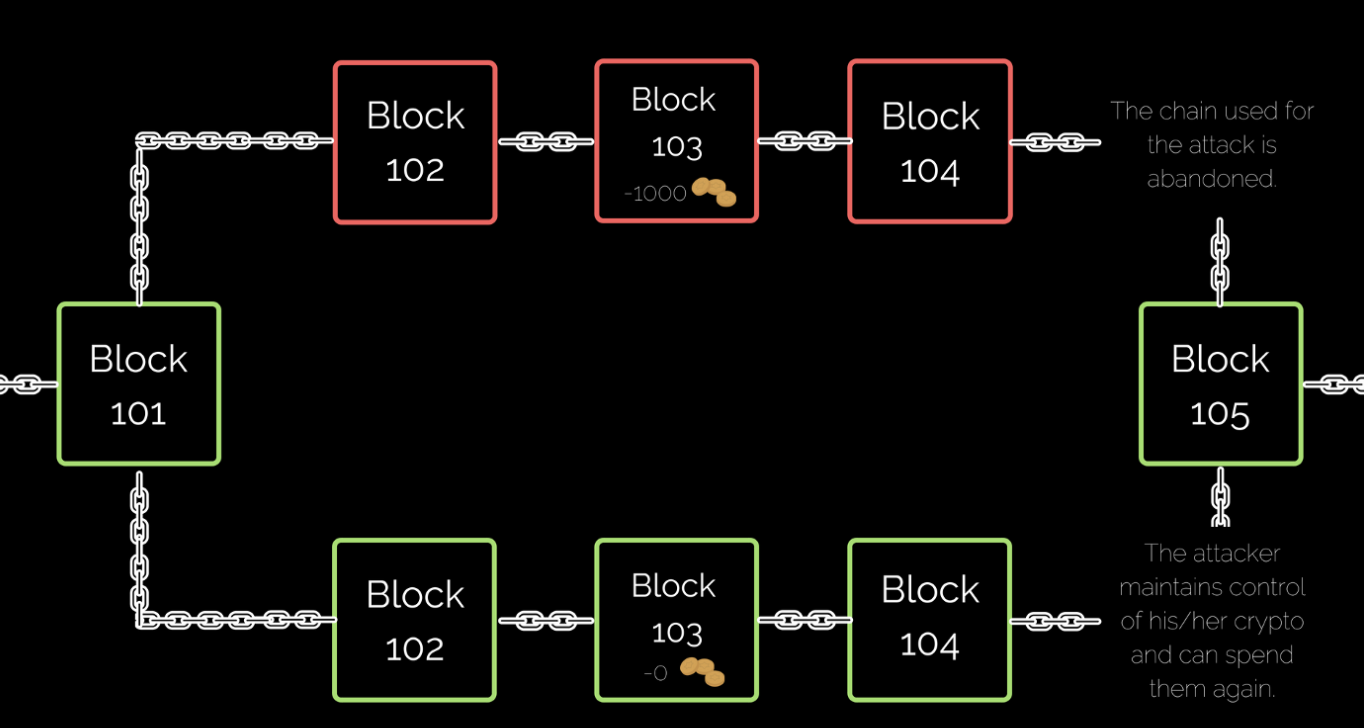
1. **Proof-of-work and proof-of-stake protocol attacks**

Blockchain networks are transitioning to proof of stake or a hybrid of proof of stake and proof of work protocols. But the proof of stake protocols is not immune to security attacks either. In proof of work, an attacker needs to have 51% of hash power, which means they need to pony up and spend physical resources to outvote genuine miners. In the proof of stake protocol, an attacker can participate in more than two fork chains and keep creating blocks without risking their stake. Since there is no proof of work required and nothing to lose, they can keep putting the same stake at each fork. They will be able to get the reward regardless of which chain becomes final.

1. **The 51% attack.**

The blockchain relies on distributed consensus mechanisms to maintain mutual trust in the network (Mosakheil, 2018). A 51% attack occurs when a group of miners grabs more than 51% of the hashing power of the blockchain network. It is a hypothetical scenario in which a single miner or group of miners decides to take advantage of their control over more than half of the network's computational power. Aside from preventing transactions from being completed, the attacker also could stop other miners from creating new Bitcoins (cloudsecurityalliance, 2020). A 51% attack is difficult to accomplish on a larger blockchain, such as Bitcoin or Ethereum. They're more likely to take place on smaller chains. In the Bitcoin whitepaper, Satoshi Nakamoto made the erroneous assumption that it would be impossible to amass 51% of the network's hashrate, failing to consider the financial incentives for doing so (MIT, 2019).

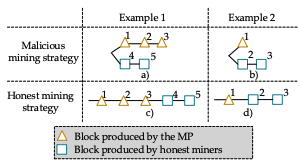
**Figure 1.5**



1. **Double-spending attack.**

Attackers could alter the transaction record for their gain. This type of circumstance, in which the verifier spends the same money twice, is known as "double spending" in Satoshi's terminology. This is not permitted to occur in a trusted system (Chou et al., 2018). An attacker tries to execute numerous double-spends by sending closing transactions for all channels that correspond to earlier states of the protocol. As soon as any errant transaction is validated, the trustworthy nodes that keep an eye on the blockchain respond by posting disputed transactions on the blockchain (Sguanci & Sidiropoulos, 2022). When an honest disputed transaction is not confirmed on the chain before the wait has passed, the adversary is successful in taking the cash.

**Figure 1.6**



Honest and malevolent miners are the two different categories of miners. The first type always uses an ethical mining strategy to carry out sustainable mining (Yang et al., 2021). The latter creates a Malicious Pool (MP) to increase their unjustified mining income. If a block is created by the MP, it can be strategically published as opposed to being published right off the bat.

1. **Sybil attacks.**

The Sybil attack is not exclusive to blockchain technology. A peer-to-peer network's security is generally under threat. In this attack, a malicious party generates numerous identities and makes use of them to control the network. This attack can also be seen in social media, where network participation and reliability are crucial. Obviously, in blockchain, the trustworthiness of the nodes is critical. To another user on the blockchain, such Sybil identities appear as new users, but behind the scenes, they are orchestrated by the same attacker. In the case of blockchain, a hacker can take over several nodes and tamper with them or, in the worst-case scenario, prevent legitimate nodes from taking part in the consensus mechanism. A well-planned Sybil attack may result in a 51% attack because of the attackers' combined hash power.

A Sybil attacker can modify the transmission power for each fictitious ID, resulting in a unique backscatter signature for every transmission (Huang et al., 2020). To disrupt their long-term geographical similarity, attackers can work together and erratically switch phony IDs during each transmission. These attacks will thus result in vastly diverse signal characteristics.

**Distributed Apps and Smart Contract Security**

Blockchain platform security vulnerabilities affect the underlying core of the ecosystem. A poorly written smart contract can still lead to a security breach regardless of how secure the platform is. The security risks associated with a traditional centralized application are still manifest in a modern Web 3.0 application.

A contract may only contain one fallback function, which is an unnamed function. Fallback functions must be visible to the outside world, cannot return anything, and cannot take any arguments. If someone tries to call a function that is not specified in the contract, it is activated. For reentrancy attacks, this approach is frequently used (Chinen et al., 2020).

1. **The reentrancy attacks.**

Attackers recognize that the crypto balances at the blockchain address are protected by the mining layer, which is difficult to exploit. But smart contracts, which also track the balances in the contract, are susceptible to coding errors. Attackers can trick a smart contract and create a mismatch between balances as known to the smart contract and as known to the blockchain protocol layer.

The fundamental issue is that the smart contract code allows itself to be called repeatedly before finishing the first call. The multiple invocations can enable an attacker to take advantage of the incorrect state of a variable created as a result.

This issue is a manifestation of a known software weakness, classified by the MITRE organization as CWE-664. As per MITRE, the software does not maintain or incorrectly maintain control over a resource throughout its lifetime of creation, use, and release.

**Scenario —** A banker's smart contract holds the crypto tokens. The smart contract offers a withdraw function that allows others to withdraw funds. Now, there is the attacker's malicious smart contract, which calls the Withdraw function. The Withdraw function in turn invokes SendEther, leading to the "fallback" function of the attacker being invoked subsequently. Now, the problem is that the "fallback" function calls withdraw again, leading to another withdrawal of funds.

Finally, control reaches the "UpdateBalance" function of the banker's contract. The job of this function is to update the balance of funds in the contract. But by that time, the 'UpdateBalance' is called a bit too late. By the time the control reaches here, the attacker would have called "Withdraw" multiple times.

1. **Front-running attacks.**

 Front-running is the practice of taking advantage of early knowledge of ongoing transactions. Knowing about pending transactions is not harmful in and of itself; the real issue is being able to act on this knowledge (Zhang et al., 2022). The origins of the front-running attack are from the traditional financial markets. Since all the transactions before their recording into the blockchain are stored in a mempool (memory pool), they are visible to everyone. Attackers can take advantage of this information and get to the so-called front of the line and get their transaction executed first. The transactions that are sitting in the memory pool will eventually be picked up to become part of a block yet to be added to the blockchain.

How does it decide which transaction will be picked up first? This depends upon the fees that are set by the user. The higher the fee, the greater the incentive for the block creator to pick a transaction over the other. The front-running attack exploits this weakness by placing a transaction before the other transactions. CWE-362 states, "the program contains a code sequence that can run concurrently with other code, and the code sequence requires temporary, exclusive access to a shared resource, but a timing window exists in which the shared resource can be modified by another code sequence that is operating concurrently" (Rodler et al., 2018).

1. **Self-destruct function attack.**

Because the immutability feature can be compromised, the self-destruct capability can occasionally be dangerous. In contrast to conventional programs, immutability is a unique and significant characteristic of smart contracts (Chen et al., 2022). No one, not even the owner, is allowed to change a contract after it is deployed to the blockchain.

The Solidity language offers a self-destruct function that allows you to deactivate a contract already deployed at an Ethereum address. The data on the blockchain is immutable. So, self-destruct means that no other users or contracts can send a transaction to it. The self-destruct function takes the Ethereum address as the input argument. When the self-destruct operation on a smart contract is executed, the crypto balance in the contract is sent to the address passed as the input argument. After that, the contract that contains the self-destruct is no longer active. But the problem is that the Fallback function of the contract that will receive the Ethers from the destructed contract will not be executed.

A fallback function in Solidity is a special handler function that is executed when no other function in a contract matches the caller's request. When a contract is receiving ethers, it must have a fallback function that is marked payable. If no such function is defined, the contract cannot receive ethers. A payable Fallback function will have logic that will not allow the contract to receive payment when it is not appropriate. But when the Ethers are being sent due to self-destruct operation, the Fallback of the receiver will not be executed. An attacker can take advantage of this and start sending ethers to a contract, depriving the receiving contract of the opportunity to validate the request.

**Threats to Non-distributed Apps and Enabler Services**

Blockchain technology uses public key infrastructure (PKI) to authenticate entities and guarantee the blockchain's integrity. For private keys, seeds, and keys held in external hardware in the Blockchain infrastructure, the Bitcoin wallet needs to be properly protected (Pal et al., 2019). For smart contracts and decentralized applications to be useful in the real world, they rely on many enabler services. Blockchain platform security vulnerabilities affect the underlying core of the ecosystem. A poorly written smart contract can also lead to a security breach. Enabler services that make up the blockchain ecosystem are also prone to security attacks.

1. **Wallets.**

Bitcoin Wallet is like the bank account where Bitcoin currency is kept. Wallet’s act as the custodians of the tokens and provide a unified, user-friendly interface to the blockchain. Wallets fall into two categories: hosted and non-hosted, or custodial and non-custodial. Wallets where users are in full control and are responsible for their tokens are known as non-custodial or non-hosted wallets.

For custodial or hosted wallets, users rely on a third-party provider for storage and management of their private keys. Some hosted wallets also allow users to store their private keys in their browsers. Hosted wallets are regular applications with web and mobile entry points accessible to users.

These applications are vulnerable to well-known classic security attacks. These are documented under the OWASP Top 10. These hosted wallet providers are also the single point of failure as they are vulnerable to attacks such as denial of service and phishing attacks. Regardless of whether it is a hosted or non-hosted wallet setting, if the keys are stored and managed by a user, they are subject to additional attacks. Keys stored locally are prone to malicious attacks using keyloggers and other malware.

These attacks can either lead to theft of the private key or malicious intrusion during the signing of a blockchain transaction. An end user's computer runs software for managing keys. Software may have its own security weaknesses. It performs cryptographic operations such as encrypting and signing transactions, but the software may be built with security bugs. Poor data management, secrets stored in plain text, or unprotected memory access are examples.

1. **Exchanges.**

 Cryptocurrency owners usually want to buy, sell, or exchange their currencies. Obviously, they can do so directly with others, but it limits their ability to locate potentially interested parties. Third-party exchanges solve this problem by offering the ability to swap crypto tokens. In centralized exchanges, a third-party intermediary facilitates, monitors, and secures the transactions. Users of the centralized exchanges are subject to "Know Your Customer" or KYC requirements.

Decentralized exchanges have no third-party intermediaries involved. Instead, the exchange protocol facilitates the transaction. End users continue to maintain control of their funds while relying on the blockchain for clearing the funds. The exchange protocol uses smart contracts to execute the transaction. One of the rapidly growing protocols that operate on Ethereum is Uniswap. There is no requirement for KYC, but liquidity is low, and orders take a long time to fill.

Both centralized and decentralized exchanges are vulnerable to attacks, but considering the attacks in recent years, centralized exchanges have been attacked more often as they have been more active. Centralized exchanges are similar to a hosted wallet in terms of their threat vectors. They are susceptible to OWASP Top 10 application-level attacks, denial of service attacks, and authentication attacks. Decentralized exchanges are based on exchange protocols. It is built on the blockchain smart contracts and is prone to the same attacks that normal blockchain transactions are. In April 2020, for example, attackers stole $25 million in currency from Uniswap by leveraging the reentrancy attack. The reason this attack was possible was due to the security bug in the Uniswap code. More details in the case studies and analysis section.

1. **Oracles.**

 Blockchain-external data sources are frequently used to supply data for blockchain transactions. The assumptions and applicability of Oracle solutions vary widely, and each system only partially solves the problem of data on-chaining (Heiss et al., 2019) Blockchain networks are isolated networks, and so are the smart contracts and decentralized apps that are built on them. These apps need to rely on data from external and legacy Web 2.0 applications to be of meaningful use.

Oracles are the off-chain middleware that enables these apps to interface with the rest of the world. Oracles handle the communication between blockchain and external data providers, business applications, IOT sensors, and financial markets. Oracles are not embedded into the blockchain network but are run as their own independent Oracle networks.

Compromised oracles undermine the trustworthiness of the data they provide. This can lead to a ripple effect on trust in smart contracts and decentralized apps. Oracles carry the same burden of providing trust and security as the rest of the blockchain networks. Oracles can be centralized or decentralized. Centralized Oracles pose a problem in the industry.

Blockchain technology is built on the principle of decentralization. Reliance on a central Oracle as a trusted source brings back a centralized model with a single point of failure. Just like with any centralized organization, Oracles are susceptible to conflicts of interest. When the people behind the Oracle stand to gain financially, they are more likely to be swayed and manipulate the data. This can lead to a breach of trust in the Oracle-dependent smart contracts. Centralized Oracles are the single points of failure and are vulnerable to attacks such as denial of service.

**End User Security**

In the IoT ecosystem, developing trusting connections across many decentralized entities is crucial. Trust data storage, which determines whether trust information is maintained inside the blockchain (on-chain) or outside, is the first blockchain-based factor (off-chain) (Shala et al., 2020).

When interacting with online exchanges or wallet providers in a web session using your browser, attacks can hijack your session. Some of the known attacks are the manipulator in the middle attack, cross-site request forgery, cross-site scripting, and injection attacks. Now, the bulk of the responsibility for protecting your session lies with the provider, but by enabling web protection control through your antivirus provider, you can add another layer of defense.

Phishing is a social engineering attack in which the attacker, disguised as a trusted entity, tricks a victim into taking any action that the victim would not have taken otherwise. For example, by clicking on a malicious link in an email. This further leads to the installation of malware keyloggers or starting a ransomware attack on victim devices.

**Case Studies: Root Cause Analysis of Security Incidents in the Blockchain Era**

The current blockchain-related occurrences are examined in this section. Of 5 high profile known publicly available events, 2 typical examples are examined and given an in-depth analysis. The results are summarized in Table 1.0. With this still-evolving technology, this research hopes to help future users learn from past mistakes and steer clear of similar mistakes.

**Root Causes of Blockchain Incidents and Preventative Measures**

**Table 1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| Incidents | Incident Type | Incident Root Causes | Prevention |
| DAO Attack | Vulnerability in Smart Contract Code | Calling a function repeatedly before the first one(s) finish, allowing you to withdraw money repeatedly | Make sure all internal state changes are performed before the call is executed |
| Parity | Vulnerability in Smart Contract Code | Malicious parties could self-destruct the contract due to missing or inadequate access controls. | If the self-destruct feature isn't necessary, think about deleting it. It is advised to construct a multisig system if there is a legitimate use-case such that several parties must consent to any such activity. |
| Bitfinex | Infrastructure Breach | The multi-signature key management system's key participants blindly approving transactions | systemic measures to stop and identify transactions |
| Zerocoin | Vulnerability in Smart Contract Code | The incorrect operator is mistakenly used by the programmer, changing the application logic | The weakness can be avoided by performing pre-condition checks on any math operation or using a vetted library for arithmetic calculations |

**Root Causes Analysis of Blockchain Incidents and Preventative Measures**

**Case study 1: The DAO (Vulnerability in Smart Contract Code)**

A DAO (Decentralized Autonomous Organization) is an organization represented by rules encoded as a transparent computer program, controlled by its members, and not influenced by a central authority. A DAO smart contract was developed and deployed on the Ethereum Blockchain in 2016. To serve as a contract for users to vote on projects and (in Ethereum) to support and invest in potentially profitable smart contract ventures via crowd sale.

**Figure 1.7**

Graphical user interface, text, application

Description automatically generated

In the DAO smart contract, a state variable is changed after a contract uses call.value. The attacker implements a fallback function (Executed after Ether is transferred from the victim contract) to execute the vulnerable function again before the state variable is changed.

**Figure 1.8**

Diagram

Description automatically generated

The transfer mechanism of The DAO lets participants transfer the ether to an external address before the state variable is changed, and checks if the balance was already transferred. As a result, an attacker could withdraw more ether than assigned by a reentrancy attack. The fallback function was used to perform the reentrancy attack. Every Smart Contract bytecode contains a fallback function that can contain arbitrary code. A smart contract can accept ether if "payable" is voided. Moreover, whenever ether is passed to the contract, the function is executed. In Solidity, there are three ways to transfer ether between wallets and smart contracts: send(), transfer() and call().value(). Call().value() was used to transfer the ether in The DAO smart contract. That method let that transfer using the maximum possible gas limit and that the state could be reversed, despite the exceptions. As a result, attackers created a similar contract to create a sequence of recursive calls to divert funds from the DAO.

**The Exploit**

**Figure 1.9**

**Graphical user interface, text, application, email

Description automatically generated**

The first step of the exploit was to propose a malicious contract and then wait a week for the proposal to see approval from curators. The original proposal that initiated this attack can be seen in Figure 1.9 "DAO Proposal #59 -Lonely, so Lonely" The fact that the curators did not do their job is not a technical flaw, but a lack of oversight.

As shown in Figure 2.0, to recap the purpose of this function: a subset of the DAO token holders decided they'd like to "split" - either because they do not agree with a proposal, or because they wish to withdraw funds. The mechanism for doing so is to create a split proposal. Split proposals take seven days to 'mature' and get participants in. Any participants voting "yes" in the split will be given the right to call splitDAO.

**Figure 2.0**

**Chart, text

Description automatically generated**

Any participants voting "yes" in the split will be given the right to call splitDAO. After approval, this function starts transferring from parent DAO to childDAO contract. The source code is in TokenCreation.sol, and it transfers tokens from the parent DA to the child DAO. The attacker was using this to transfer more tokens than they should be able to into their 'child' DAO.

**Figure 2.1**

Graphical user interface, text, application

Description automatically generated

In Figure 2.1 the code is calculating how much to move for this particular caller, and then calls the createTokenProxy() function.

Because p.splitData[O] is going to be the same every time the attacker calls this function (it's a property of the proposal p, not the general state of the DA), and because the attacker can call this function from withdrawRewardFor before the balances array is updated, the attacker can get this code to run arbitrarily many times using the described attack, with funds ToBeMoved coming out to the same value each time.

**Figure 2.2**

Graphical user interface, text, application

Description automatically generated

The withdrawRewardFor() function shown in Figure 2.2 above is getting called, and then the totalSupply, balances, and paidOut variables are getting set after the call. This is a vulnerability. If withdrawRewardFor can be attacked with Race To Empty, it will be called before the balances or paidOut hash tables are updated. At this point, a big mistake was made —the call to the transfer() function was missing. In Solidity, event logging functions are capitalized, so Transfer() is an event logger while transfer() is a function. As a result, the transfer function couldn't be called. If transfer is written in lowercase, DAO tokens are burned.

**Figure 2.3**

Graphical user interface, text, application, email

Description automatically generated

As shown in Figure 2.3 when rewardAccount.payOut is called without a gas amount, no rewards are generated. rewardAccount is a "Managed Account" contract, so the way to define it is: function withdrawalRewardFor (address, account, uint amount). balanceOf refers to balance that never gets updated. In addition, paidOut and totalSupply also never gets updated because that code in SplitDAO() will never be executed. As a result, the attacker can easily both claim their reward and execute splitDAO() function again. accumulatedlnput is the sum of ether that has been sent to the contract. uint public accumulatedInput; If default function of the reward account is used, accumulatedInput is easily manipulable.

***function (f***

***accumulatedInput += msg.value;)***

The attacker sends some ether to the reward account, and it will be evaluated to false. This fact will happen every time the ether is sent.

***if ((balanceOf(\_account) \****

***rewardAccount. accumulatedInput()) /***

***totalSupply < paidOut/ \_accountl)***

Later, it was shown that the attacker did not need ether to perform the attack because the DAO pay them although amoutToBePaid is 0. The attacker's operation did not depend on reward account was full or empty.

Finally, every time the attacker calls p.splitData[0] function the attack will arbitrarily run the code many times. Before the balance is updated, the attacker can call p.splitData[0] function from withdrawRewardFor.

In summary, the DAO hack was a confluence of bad programming practices, a typo error (Transfer instead of transfer), and many complex calls. Balances must be checked before a transaction and not after a transaction.

**Solution.** One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished (*SWC-107 · Overview*, n.d.). This may cause the different invocations of the function to interact in undesirable ways. All internal state changes should be performed before the call is executed and use a reentrancy lock (*SWC-107 · Overview*, n.d.).

When The DAO attack was performed, security patterns in Smart Contracts design had not yet been established (Ethereum Smart Contract Best Practices, 2022). From The DAO attack, security best practices started to be developed and the used attack vectors in the project were checked in other contracts to prevent an attack like The DAO from happening again. Developers consider the security aspect when they write a Smart Contract in Solidity and many Smart Contract auditing companies have strongly emerged in the last four years.

**Best Practices**

1. Using functions send() or transfer() instead of call.value
2. Use functions with greater gas stipend.
3. Limiting the amount of gas passed to call.value if it must be used.
4. Don't allow internal state updates to happen AFTER ether is transferred.
5. Validate any external function call inside a method is properly controlled.
6. Scan all solidity files for security vulnerabilities, along with manual review.

**Case Study 2: Parity Hack (Vulnerability in Smart Contract Code)**

In 2017, a well-trusted and established development team named Parity created a smart contract that was a "multi-signature wallet." This is like regular Ethereum wallets but requires multiple approvals to withdraw Ether from the contract/wallet (Petrov, 2017). A multi-sig wallet is not exactly like Multifactor authentication. It requires 2/3 keys to interact with the contract functions rather than 2 or more factors (like a passcode or an SMS message). The developers overlooked some very critical vulnerabilities, and these were exploited after the users had depended on this contract to store over 280 million dollars’ worth of funds.

The vulnerability was dealing with a "self-destruct" feature that could be called by anyone. The Parity Wallet bug was a prominent vulnerability on the Ethereum blockchain which caused 280 million USD worth of Ethereum to be frozen on the Parity wallet account. It was due to a very simple vulnerability: a library contract used by the parity wallet was not initialized correctly and could be owned or destructed by anyone.

Once the library was destructed, any call to the wallet library would then fail, effectively locking all funds.

**Figure 2.4**

Graphical user interface, text

Description automatically generated

The vulnerable parts of the multi-sig wallet library are within the initWallet() function, and the kill() function shown in Figure 2.4. The purpose of performing this sort of "centralized" architecture in a "Decentralized" network is to save gas fees on deployment transactions when new wallets are created. A newly deployed multi-sig wallet can invoke the code from the already deployed library, and not pay the extra gas fee on that EVM bytecode. However, because a new wallet that was created required interaction with this library, it created a single point of failure. Every parity multi-sig wallet that was created up to July

20, 2017 relied on this library contract.

**Solution.** If the self-destruct feature isn't necessary, think about deleting it. It is advised to construct a multisig scheme if there is a legitimate use case such that many parties must consent to the self-destruct operation.

**Findings, recommendations, and conclusion**

**Findings**

Reentrancy vulnerability is continuously in evaluation (Fatima Samreen & Alalfi, 2020). There are a couple of studies and proposals that could be used to detect and used as a precaution to double-spending attack, particularly on the Bitcoin network. Ghassan et al. (Karame, Androulaki, & Capkun, 2012) discusses the double-spending attacks on fast payments in Bitcoin and propose an attack model that enables the detection of double spending attacks in fast transactions. The three detection techniques the paper presented are: (a) listening period, (b) inserting observers, and (c) forwarding double spending attempts.

Forwarding Double-Spending Attempts in the Network Analysis presented in Karame, (Androulaki et al, 2015) states nearby peers should notify the merchant about the attempt that double-spending the same coins in the Bitcoin network. Specifically, whenever a peer receives a new transaction, it checks whether the transaction uses coins that have not been spent in any other transaction that exists in the blockchain and their memory pool. If the transaction does not have already spent coins in it, peers add the transaction to their memory pool and forward it in the network. Podolanko, Ming, and Wright (n.d.) provide a countermeasure against double-spending attacks on Bitcoin Fast-Pay transactions. They proposed a solution called Enhanced Observers (ENHOBS), a hybrid of observers and the peer alert system. In this scheme, the ENHOBS will do more in-depth inspections of all transactions received and compares their outputs and inputs.

The degree of openness and decentralization it offers, together with an appropriate amount of security and privacy, which were previously thought to be impractical, is one of the main contributions of blockchain. Although there are many intriguing security and privacy aspects in the blockchain ecosystem, there are still significant security and privacy risks because blockchain is still a developing technology. According to an extensive assessment on the security features of the Blockchain, there are security dangers now in existence because of the Blockchain's existing flaws (Bahack, 2013) The blockchain network, blockchain miners, mining pools, smart contracts, and private key security are all at risk of attack in addition to the security risks that result in double-spending assaults. Based on the characteristics of each vulnerability, this study also analyzed the nature of each vulnerability discussed in and classified the security threats to different fields of the blockchain.

**Future Research Directions and Recommendations**

**Consensus mechanism**. A proof-of-work (PoW) based consensus method is used by the majority of well-known public blockchain applications, including Bitcoin, Ethereum, and others, to secure user actions. Proof-of-work also produces distributed consensus and offers a workable answer to the Byzantine Generals' conundrum. But PoW-based blockchains are vulnerable to a variety of security risks. Double spending, which is virtually always feasible with Bitcoin, poses the biggest threat. The wasting of computer resources on blockchains using PoW algorithms is another significant drawback. Some blockchains, like Ethereum, intend to adopt a hybrid PoW and PoS consensus method to address this issue. However, PoW always necessitates a significant resource waste, both in terms of power and computing resources, as Bitcoin continues to dominate the blockchain. Therefore, one viable study area is to provide consensus algorithms that are more reliable, safe, and readily scalable. The degree of openness and decentralization that blockchains like Bitcoin have enabled, which was previously seen as inconceivable, is one of their most important achievements.

**Mining pool protocols.** The original idea of mining, which may be based on a Proof of Work, Proof of Stack, Proof of Burns, or other method, not only protects the blockchain but also offers distributed consensus. Without mining schemes, phony identities might easily sabotage the consensus process and stop systems from operating normally by launching a Sybil attack. However, the decentralization of some blockchains, like Bitcoin, is under danger due to the fast-expanding number of mining pools.

**Game theory and stability.** Since the majority of blockchain operations are dependent on the mining, miners might act egotistically by hoarding their blocks and releasing them at their discretion. Between the greedy miners and the honest miners in the network, this type of selfish behavior may present a game theoretic challenge. In contrast to the network's honest chain miners, the selfish miners or attackers may attempt to lengthen their own chains to conduct various attacks, such as double spending. So, one viable study area is finding equilibrium to bring about network stability.

**Cryptographic and keying techniques.** A common vulnerability of the existing Simplified Payment Verification (SPV) protocol is its susceptibility to attacks like Sybil or double spending. SPV is a lightweight protocol used to verify the transactions submitted by the users. It is necessary to use a stronger verification procedure. A strategy that is dispersed rather than centralized is always desirable for important computations and operations. Innovative key calculation and storage methods in the Blockchain are thus a potential study area in this approach. Additionally, Bitcoin heavily relies on the ECDSA method and features hash functions like SHA-256 as well as bad randomness qualities, opening a new area of study.

**Improving blockchain protocol.** Despite blockchain potential, it still faces significant concerns in term of privacy and scalability. The immutable nature of the blockchain makes it impractical for many other applications. Recently, the research called “Redactable Blockchain” (Ateniese, Magri, Venturi, & Andrade, 2016) present modification in blockchain techniques that allows operations such as re-writing one or more blocks, compressing any number of blocks into a smaller one

**Conclusion**

This study focuses on the security issues of the blockchain technology. By studying the different fields of the blockchain such as consensus mechanisms, blockchain network, mining process, data storage and key management, and smart contracts functionality, the study reviewed all the existing vulnerabilities in this area. Additionally, based on the understanding from the threat, the study analyzes the two most significant hacks.

**References**

Andesta, E., Faghih, F., & Fooladgar, M. (2020, October 1). *Testing Smart Contracts Gets Smarter*. IEEE Xplore. https://doi.org/10.1109/ICCKE50421.2020.9303670

Bahack, L. (2013). Theoretical Bitcoin Attacks with less than Half of the Computational Power (draft). *ArXiv:1312.7013 [Cs]*, *v1*. https://arxiv.org/abs/1312.7013

Chen, J., Xia, X., Lo, D., & Grundy, J. (2022). Why Do Smart Contracts Self-Destruct? Investigating the Selfdestruct Function on Ethereum. *ACM Transactions on Software Engineering and Methodology*, *31*(2), 1–37. https://doi.org/10.1145/3488245

Chinen, Y., Yanai, N., Cruz, J. P., & Okamura, S. (2020). Hunting for Re-Entrancy Attacks in Ethereum Smart Contracts via Static Analysis. *ArXiv:2007.01029 [Cs]*, *v1*. https://arxiv.org/abs/2007.01029

Chou, C.-N., Lin, Y.-J., Chen, R., Chang, H.-Y., Tu, I.-P., & Liao, S. (2018). Personalized Difficulty Adjustment for Countering the Double-Spending Attack in Proof-of-Work Consensus Protocols. *ArXiv:1807.02933 [Cs]*, *V1*. https://arxiv.org/abs/1807.02933

cloudsecurityalliance. (2020, August 26). *Glossary/51% attack (aka Consensus HiJacking).md at master · cloudsecurityalliance/Glossary*. GitHub. https://github.com/cloudsecurityalliance/Glossary/blob/master/glossary/5/51%25%20attack%20(aka%20Consensus%20HiJacking).md

*Ethereum Smart Contract Best Practices*. (2022). Consensys.github.io. https://consensys.github.io/smart-contract-best-practices/

Fatima Samreen, N., & Alalfi, M. H. (2020). Reentrancy Vulnerability Identification in Ethereum Smart Contracts. *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)*, *v1*. https://doi.org/10.1109/iwbose50093.2020.9050260

HACKETT, R. (2017, August 22). *Why Big Business Is Racing to Build Blockchains*. Fortune. https://fortune.com/2017/08/22/bitcoin-ethereum-blockchain-cryptocurrency/

Heiss, J., Eberhardt, J., & Tai, S. (2019, July 1). *From Oracles to Trustworthy Data On-Chaining Systems*. IEEE Xplore. https://doi.org/10.1109/Blockchain.2019.00075

Huang, Y., Wang, W., Jiang, T., & Zhang, Q. (2020). Detecting Colluding Sybil Attackers in Robotic Networks using Backscatters. *ArXiv:2012.14227 [Cs]*, *V1*. https://arxiv.org/abs/2012.14227

*MIT 51% Attacks*. (2019). MIT Digital Currency Initiative. https://dci.mit.edu/51-attacks

Mosakheil, J. (2018). *Security Threats Classification in Blockchains* (A. Abu Hussein, S. Herath, & T. Ghosh, Eds.; p. 48). https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1093&context=msia\_etds

Pal, O., Alam, B., Thakur, V., & Singh, S. (2019). Key management for blockchain technology. *ICT Express*, *v7*(Issue 1). https://doi.org/10.1016/j.icte.2019.08.002

Petrov, S. (2017, November 8). *Another Parity Wallet hack explained*. Medium. https://medium.com/@Pr0Ger/another-parity-wallet-hack-explained-847ca46a2e1c

Rodler, M., Li, W., Karame, G. O., & Davi, L. (2018). Sereum: Protecting Existing Smart Contracts Against Re-Entrancy Attacks. *ArXiv:1812.05934 [Cs]*, *v1*. https://arxiv.org/abs/1812.05934

Sguanci, C., & Sidiropoulos, A. (2022). Mass Exit Attacks on the Lightning Network. *ArXiv:2208.01908 [Cs]*, *v1*. https://arxiv.org/abs/2208.01908

Shala, B., Trick, U., Lehmann, A., Ghita, B., & Shiaeles, S. (2020). Blockchain and Trust for Secure, End-User-Based and Decentralized IoT Service Provision. *IEEE Access*, *8*, 119961–119979. https://doi.org/10.1109/access.2020.3005541

Storublevtcev, N. (2019). Cryptography in Blockchain. *Computational Science and Its Applications – ICCSA 2019*, 495–508. https://doi.org/10.1007/978-3-030-24296-1\_39

*SWC-107 · Overview*. (n.d.). Swcregistry.io. Retrieved October 15, 2022, from https://swcregistry.io/docs/SWC-107#modifier-reentrancysol

Szabo, N. (1994). *Smart Contracts*. Www.fon.hum.uva.nl. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html

Wang, H., Chen, K., & Xu, D. (2016). A maturity model for blockchain adoption. *Financial Innovation*, *2*(1). https://doi.org/10.1186/s40854-016-0031-z

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018). Blockchain technology overview. *National Institute of Standards and Technology*. https://doi.org/10.6028/nist.ir.8202

Yang, R., Chang, X., Mišić, J., & Mišić, V. B. (2021). Deep-Dive Analysis of Selfish and Stubborn Mining in Bitcoin and Ethereum. *ArXiv:2112.02588 [Cs]*, *1*. https://arxiv.org/abs/2112.02588

Zhang, H., Merino, L.-H., Estrada-Galinanes, V., & Ford, B. (2022). F3B: A Low-Latency Commit-and-Reveal Architecture to Mitigate Blockchain Front-Running. *Arxiv.org*, *v1*. https://doi.org/10.48550/arXiv.2205.08529

Reused cita

Ellison, C., & Schneier, B. (2000). Ten risks of PKI: What you're not being told about public key infrastructure. *Comput Secur J*, *16*(1), 1-7.