JOURNAL OF INTERNATIONAL CRISIS AND RISK COMMUNICATION RESEARCH
ISSN: 2576-0017
2024, VOL 7, NO S8

Blockchain and Cybersecurity: Addressing Smart Contract Vulnerabilities in Decentralized Applications

Ahmed Eraig[1], Randa Abdelrahmen Mohamed Khalid[2], Leila Abdelgader

1,2,3. Department of Computer Science, Khurmah University College, Taif University, Taif, Saudi Arabia

*Corresponding Auther's Email: a.ali@tu.edu.sa, r.khalid@tu.edu.sa, lmabdelgader@tu.edu.sa

(Abstract)
Blockchain technology can provide smart contracts with valuable attributes: distributed consensus, tamper evidence, auditability, and intelligent behavior based on asset interactions. However, there are critical risks and challenges that need to be addressed. The distinction between public and private blockchain access is a key challenge. Finding the right balance is crucial for system functionality and security. Logic and legal implications of smart contracts can lead to unintended consequences or conflicts with regulations. Careful review and analysis are essential. Immutability brings benefits and risks, requiring thorough testing and monitoring. Non-upgradeability poses a challenge for scalability. Connectivity, exposure, storage, trustworthiness, and data security are critical aspects. Researchers are working on mitigating these challenges through secure design options. Education, guidelines, and robust governance frameworks can minimize conflicts and legal issues. Successful implementation requires addressing risks and challenges for the integrity, reliability, and compliance of smart contracts. [1][2][3][4].

1. Introduction

Decentralized applications (dApps), which are built on blockchain infrastructure, use smart contracts to facilitate peer-to-peer transactions and contain application logic. Due to the financial and security implications of a privacy-focused dApp, secret contracts are being developed to protect sensitive and commercially valuable dApp data from being manipulated or disclosed. Funneling all dApp data through a single central authority both scales poorly, as it introduces centralization pressures and funnels applications' potentially most sensitive data through a single entity.

In this context, the decentralized secret contract (DeSC) framework is introduced. Building on the trusted execution environment support available on contemporary blockchain infrastructure, DeSCs modify dApp smart contracts to control sensitive data handling through cryptographic authorization. Secured with a construction suitable for dApps, DeSCs enable valuable privacy-focused and confidentiality-preserving decentralized applications without data trade-offs, whereby fully decentralized data handling is guaranteed, or otherwise data remains within the bounds of the dApp owner. In scaling to push all data through a single mechanism, secret contracts introduce a single authority problem at a new layer – data privacy and preserving confidentiality within decentralized applications. [5]

1.1. Background of Blockchain Technology

Blockchain is a distributed digital ledger that offers a new approach for information sharing and storing. The technology consists of blocks of digital information that are publicly accessible and linked together by a cryptographically secured chain. Originally, the concept was developed as a digital ledger for cryptocurrency. However, the technology attracted wide interest as an enabling technology for many other applications [6]. There are two main types of blockchains: permissioned and permissionless. In the first, transactions are validated and approved by a predefined list of participants, such as banks, while in the latter, participants can join and leave anytime, and together they validate transactions. Blockchains effectively eliminate the need for central authorities and intermediaries to process and validate transactions, enabling great cost reductions and improved transaction speed, safety, and transparency. What makes blockchains so attractive is mainly the use of cryptographic hash functions, consensus mechanisms, and smart contracts.

Cryptography plays a critical role in blockchain solutions. Cryptographic hash functions play a key role in securing the links between blocks. The hash function is a one-way transformation that takes a message, a transaction in the case of a blockchain, as an input, and produces a fixed-length output. Digital signature mechanisms and cryptographic commitments are also used. Cryptographic hash functions and digital signatures make transactions irreversible and tamper-resistant, while commitment schemes ensure that submitted proposed solutions do not leak to others and are legally binding [7]. Cryptographic consensus mechanisms serve the main objective of blockchains, namely, to agree on the state of the ledger history. In standard blockchain solutions, cryptographic consensus mechanisms rely on enormous amounts of computational power, energy consumption, and time to achieve consensus. Smart contracts allow automating complex business logic and processes. Smart contracts are self-enforcing automated contracts that can compute, validate, and execute the terms of an agreement between peers. A user, anonymously known as the submitter, submits a contract and its locally generated solution to the decentralized application. The blockchain network validates the solution, and if proven valid, commits it. Once committed, the smart contract performs the execution and commits the result. Based on the commitment procedures provided, the smart contract parties and third parties involved in the contract, or even media and regulators, can track contract execution on the blockchain. Blockchain technology provides a security layer to smart contracts. The protocol itself is inherently secure, and there is no surreptitious behavior once on-chain. [1]

1.2. Importance of Cybersecurity in Decentralized Applications

Decentralized applications are an important development in creating a private and secure digital world. They show promise in a number of traditional applications as a solution to problematic centralized security and privacy issues. Where a contract could be set up to take both parties' money, a decentralized contract can take the money and act as an impartial judge, without requiring any individual party to be relied on as the sole administrator. Additionally, decentralized technology can help developers by reducing legal and administrative overhead costs that are difficult to navigate and address in current, centralized systems. The transaction model for blockchain consensus projects necessitates that processes be simpler and execute in a predictable manner. Ideally, both blockchain technology and smart contracts are designed to prevent abuse and exploitation by malicious actors. This includes the ability to compute results in response to a predefined, legitimate transaction, and set up legally binding contracts where the security execution of the code itself negates the requirement for intermediary trust. However, it is the security properties that are a unique novelty, as none of the preceding decentralized market idea designs allow all intentional parties to execute and enforce decentralized technology. Such safety benefits have invited increased interest and investment in designing and fabricating these software and hardware solutions. Given the large stakes on which these decentralized applications rely and the increased capital entering the market, developers need to be extra cautious to ensure that their contracts are secure and cannot be used to exploit problems that could lead to disapproved legal decisions. [8]

1.3 Research Problem

An attack on a decentralized autonomous organization resulted in a notorious and widely discussed smart contract vulnerability called the reentrancy attack. In the reentrancy attack, attackers reentered contracts multiple times before updating any of the contract states to wallet balances, resulting in significant funds being stolen. However, this was a complex, recursive attack, and it has been pointed out that recursive receiving of assets is security-wise incorrect unless an upper limit is set. One version of a collection of completed smart contracts was designed intentionally as a trap for bad actors attempting to either knowingly or through the use of automated bots to attack the smart contracts contained within. Such smart contracts are only effective because they are intentionally written with vulnerabilities, which are known and apparently exploited for financial gain by the contract creators.

Thus, it can be concluded that for some unknown or computationally challenging problems, it is advantageous to deliberately create exploitable vulnerabilities in some smart contracts. However, the ramifications of such processes on the security of all other smart contracts are unclear. Vulnerability-tolerant products already appear today as allowing progression independent of the maturity of an artifact and also to keep potentially necessary evolutionary adaptations more feasible. Considering deliberately

inserted vulnerabilities is important; refinement of early integrity or maturity metrics regarding component immaturity might allow for a more directed avoidance of intentional, implemented medium-mature domain violations. [9]

1.4 Research Objectives

Beyond an extensive review of the sources and models that cover the different security levels and threats that may affect and are involved in smart contracts, our research will help address the cybersecurity risks that occur within smart contracts, especially those that arise in the implementation of the code, which are commonly ignored. As is known, smart contract code gets executed in a decentralized environment and has the capacity to move large sums of cryptocurrencies; therefore, any security breach poses a matter of concern for any organization that wants to adopt these new types of applications in a reliable way, which enables an unprecedented level of trust between entities. We will propose and implement the solution within a real blockchain application that has been adopted by real entities. The importance of this study is based on the fact that smart contracts will be, in a few years, the digital solution widely used in contracts between organizations, and code development errors pose challenges that require different and specific methodologies to reduce vulnerability and exploitability, which, until today, are key concerns in these blockchain applications.

1.5 Key Research Questions

The main goal of this research is to address the inadequacies and limitations of existing smart contract security tools to enhance and strengthen the security of Ethereum-based smart contracts and ensure the efficient operation of blockchain DApps. This study aims to answer the following research questions: RQ1: Where do we stand with vulnerability prediction on Ethereum smart contracts, and what can we learn from traditional software engineering, cybersecurity, and blockchain fields? RQ2: How can machine learning be used for the prediction of Ethereum smart contract security? Can we identify the most effective machine learning techniques and the best smart contract features used for security prediction? RQ3: Can the same machine learning model and smart contract security features be used to predict cyberattacks on the whole DApp? Can our model be used for the prediction of cyberattacks on other types of decentralized applications? RQ4: How can we use the insights from identifying smart contract vulnerabilities and predicting cyberattacks on Ethereum DApps to build more secure DApps?

1.6 Methodology

A preliminary research study was conducted. This consisted of a review of academic articles related to smart contract security to find relevant security incidents, as well as industry-level reports. This was primarily intended to identify characteristics or techniques relevant to the taxonomy and to ensure comprehensive coverage in the categories. Additionally, grey literature sources and posts on internet forums specifically addressing smart contract use and problems were utilized. This provided a broad overview of public opinion and problem statements that had been identified. The moderators collated some known bug and incident reports at this stage, making use of their own professional experience and knowledge, as well as some searches. This provided nine known categories associated with smart contract bugs [10]. The moderators then drafted a working version of the taxonomy based on the categories identified. Initially, this taxonomy aimed to be clear for developers and also applicable to smart contracts, decentralized applications, and blockchain design. Later adaptation would need to explore whether tools might assist in applying such a taxonomy to decentralized applications that do not have class or object structures related to security risks.

1.7 Expected Outcomes

The complexity of smart contract design will always present attack opportunities for a well-resourced adversary. The difficulty is reduced through peer review of smart contract standard development. Security audits of contracts mitigate the dangers inherent in contract coding and deployment. A key element in long-term stability will be the existence of entities that act as oracles reflecting the will of the consensus party, providing this authority in the real world. In this way, the blockchain can manage and provide authority over digital assets deployed in the real world. In general, it is possible to increase the system's resilience to bugs by building in allowable correction measures and automated error management systems. This may protect against a significant number of malicious and non-malicious trespassing.

Mitigating and reducing others becomes a question of system and insurance design [11]. In the longer view, the development of sufficient contract reliability techniques should create a stable enough environment for modeling, quantification, and finally for contractual development and management by DAOs. This has the potential to be a huge improvement over the current legal, social, and contractual structures that support society, and we hope this helps serve as motivation for deeper exploration into this area.

2. Smart Contracts in Blockchain

Smart contracts are self-enforcing, code-based programs that execute automatically, without the need for an intermediary. They are stored in many decentralized blockchains and can be properly reviewed and publicly audited for users supposedly placing their trust in these technologies. Due to this decentralized efficiency, smart contracts are expected to play an important part in a variety of different decentralized applications beneficial for many uses. Smart contracts can be used in blockchain for a number of different functions. To begin, smart contracts can be used to provide trust in the establishment of an identity system that allows users to interact with devices in their environment in a trustworthy manner. Smart contracts can also be used in the establishment of decentralized governance. This is useful, as organizations will often disagree about decision-making, typically because not everyone has the same incentive to reach harmonious conclusions.

Smart contracts can also be used effectively to power different voting mechanisms between different parties, reaching binding agreements with a powerful result on logging results and remaining verifiable by third-party actors. Users can leverage smart contracts to manage funds in a secure and verifiable manner. Smart contracts are usually implemented in a type of virtual machine on decentralized blockchain systems, surrounded by a consensus layer, which manages how the smart contract interacts with the system. This means that smart contracts are isolated, meaning that for applications to function similarly to the real world, they are forced to trust outside data in the consensus layer, exposed to malicious oracles, and capable of affecting the state of the blockchain in ways not properly anticipated. [12]

2.1. Definition and Functionality

Blockchain is a decentralized, distributed, public ledger, typically based on a proof-of-work mechanism, which ensures the integrity of transactional data accurately. It is a type of distributed ledger, where the ledger is a tamper-evident public record of all transactions. It is continuously updated for all participants of the network to ensure consistency and trust. Blockchain offers trust, integrity, and security commitments, which are achieved through a certificateless transaction mechanism, virtual wallet, and distributed timestamp. A smart contract is neither smart nor a contract. It is referred to as a program that encompasses machine-readable agreement logic, which can be executed and enforced in an automated and decentralized environment. Smart contracts execute programmatic or computerized logic in a blockchain virtual machine. In physical terms, smart contracts are immutable, distributed, autonomous software applications that run on a peer-to-peer network, wherein the business logic of the contract terms is directly written into the code and there is no central authority other than the consensus of the participants of a blockchain network to realize the agreement. The rise of smart contracts occurs primarily on the Ethereum blockchain, which is an open blockchain platform enabling users to build and execute self-enforced decentralized applications, also known as smart contracts. It introduced a separate paradigm from the conventional blockchain concept and allows the creation of a peer-to-peer smart contract system without a trusted third party. Upon the initiation of the transaction, it generates a contract address on the blockchain where the contract agreement can be verified publicly, negating any termination. Additionally, the code controls the behavior of smart contracts; it executes and enforces the agreement rules across a decentralized blockchain upon the terms and conditions. As an extensible blockchain technology, Ether supports smart contracts with a full Turing-complete language [1]. The advanced capabilities in Ethereum provide smart contracts with a high degree of flexibility, allowing them to represent a diverse range of agreements from escrows to multi-signature wallets and beyond, and to run any kind of application or logic.

2.2. Examples of Smart Contracts

One of the first DAOs built on the Ethereum platform was called "The DAO". "The DAO" project was a decentralized venture capital fund that allowed Ethereum token holders to set up a DAO, fund it by sending Ether, and, in turn, vote on how to spend the Ether within the DAO. If a majority vote was not reached during the voting period, participants could choose to leave the DAO and take their Ether back. The creators of "The DAO" wrote a smart contract that contained the rules and design of the fund. Unfortunately, a flaw in the smart contract allowed an attacker to "feed" the contract an infinite number of recursive withdrawal proposals that resulted in a "race to empty" and led to the loss of about US$50 million of Ether from "The DAO". When the DAO operators attempted to remove the exploit, a hard fork was proposed to roll back the Ethereum blockchain so that the funds were restored to the original addresses that funded "The DAO" smart contract, violating the underlying consensus principle of the DAO and allowing Ethereum developers to access the funds without the approval of the DAO token holders.

The attack made headlines and drove the price of Ethereum down. Concerned that this hack and Ethereum's response could tarnish the reputation of the platform, Ethereum's creators voted on whether to hard fork the Ethereum blockchain so that "a working example of a DAO project that could authorize significant amounts of currency could exist, show that it could succeed, and therefore lead to the notion that DAOs and smart contracts are the way of the future". [13]The Ethereum community decided, with a majority vote, to implement a type of fork that would (i) negate the hack, (ii) enable the DAO token holders to recoup their funds, and (iii) generate a clear "lesson learned" that would caution smart contract developers against using complex, untested, non-secure coding, especially within code that involves high amounts of funds.

3. Vulnerabilities in Smart Contracts

The potential benefits of smart contracts are significant. However, there are severe risks and challenges from security vulnerabilities in smart contracts, and attackers are currently exploiting existing vulnerabilities to steal or freeze a substantial amount of Ethereum. Smart contracts, unlike traditional contracts, are immutable once deployed, which means a contract once deployed can no longer be changed, and actions taken within contracts can only be governed by the predetermined logic in the code. These disadvantages mean any vulnerabilities in smart contract code will suffer from serious consequences. A defect cannot be fixed after deployment, and there are numerous examples in which the defects in deployed smart contract code are identified and successfully exploited by attackers in the wild. The exploitation of these vulnerabilities has led to billions of dollars in financial losses. [14]

In this work, we categorize the vulnerabilities of smart contracts in blockchain systems at two levels. The first level regards smart contracts as programs and focuses on the fundamental programming vulnerabilities. Various static analysis tools have been verified to find the first level defects of the smart contracts from the source code to defend vulnerable contracts across Ethereum consistency. Our Finding Fake Vulnerabilities approach has demonstrated detection precision over 98.5%. The second level focuses on the design problems in the smart contracts, where the weaknesses do not result from programming bugs, but from issues at the level of the smart contract's concept and architecture. Our understanding of smart contract vulnerabilities in this work can help future research design more secure smart contracts.

3.1. Overview of Common Vulnerabilities

Following are a few general vulnerabilities that are common across all decentralized applications:

Transaction ordering dependence Certain types of decentralized applications, such as betting applications or blockchains for winning games, involve transactions that are sensitive to transactions sent by other parties. As it is determined who emerged victoriously in a game or won a bet based on the result of other challenge transactions with lower block timestamps, it is important to ensure that the transaction ordering is what is anticipated [15]. This is often referred to as front running or sandwich attacks. An attacker mines a block with the ordering that benefits him and submits it in the preceding block. The first transaction is then included in the following block, at which time the attacker's submission is no longer valid because he is no longer the earliest to submit. Another developer argued that it is not possible to

prevent these attacks through any method that would also prevent miners from arbitrating transactions. This could be resolved by the introduction of new opcodes to enforce fair ordering of transactions.
Timestamp dependence Similar to the previous one, certain algorithms might depend on block timestamps, which can be manipulated by miners. A workaround is to enforce a bound on the range of block times.
Split delegate call gas proxies Gas is a fixed computation cost paid for the running of a smart contract. Usually for delegate calls, a smart contract can use the caller's remaining gas in multiple ways. An attacker can make the observer pay for a significant number of delegate calls to slow down block proposal times. Since with standard exchange implementations, calls trace through two such calls, standardizing the forwardProxy would claim any remaining gas for the contract and retain the standard behavior. Small attacker-owned data chunks. These chunks have no mining limitations for the consumer miners. To avoid abuse, data should be metered somehow; for example, a counter or a rate limiter. Nonetheless, a potential problem with this strategy is that content addressing is also possible in a fully peer-to-peer fashion. Further, the necessary level of protection and the merits of relying on separate incentive models could both also have industrial consequences.
MethodInfoCall gas proxies If miners can execute priority functions such as block header modifications during the consensus algorithm, to avoid typical errors, an explicit protection mechanism should be in place. While in some cases the opcode, which only has effect if the sender account has sufficient ether, is enough to transparently manage reentrancy, in other cases a functional need becomes apparent. Mining centralization attack.

3.2. Real-world Examples of Exploited Vulnerabilities

In June 2016, an attack against The DAO – a venture capital fund built on the Ethereum blockchain, whose investment decisions were made using smart contracts – resulted in disastrous consequences. The attacker exploited the recursive nature of the contract, jamming open the door for the attackers to grab the equivalent of millions of dollars. Although the smart contract was published by the entity that manages the DAO Token Holders, which explicitly prohibits the exploitation of bugs, the attacker did not respond to this warning. This case marked the beginning of smart contract security research for Ethereum, which remains a vivid blockchain platform in the perspective of decentralized application development. Since security best practices are not built in, smart contract developers might benevolently introduce logic bugs that could have severe implications. The used language, Ethereum's safe Math library, and other best practices might be used as a safety net, but more often this is not the case.
More than a year after The DAO attack took place, a smart contract technology ecosystem made up of programming languages, frameworks, static analysis tools, and publications on smart contract vulnerabilities has formed, specially centered on the Ethereum blockchain. However, the majority of the smart contract code that has been developed and employed does not utilize the current best practices and/or has been affected by one or more identified vulnerabilities [16]. Moreover, due to the fact that the underlying technologies are still in an early stage, security solutions did not materialize in an organized way in comparison with, for example, the patterns and frameworks for web-based systems. There is still no guarantee that the proposed solutions can effectively detect and prevent security threats based on real-world decentralized applications.

4. Cybersecurity Measures for Smart Contracts

Smart contracts hold great potential but have received much criticism for inherent vulnerabilities. Coding errors in smart contracts have led to dramatic financial loss and fading confidence in the technology. As a concern for the cybersecurity of DApps, smart contract vulnerabilities present an analog paradox with real-world vulnerabilities faced by e-commerce and web applications. This has led to the speculative question of whether smart contract security can affect the security of blockchain networks themselves. In this chapter, the anatomy of smart contract vulnerabilities and historical events is dissected to enumerate issues and inconsistencies while outlining discoveries from previous literature. By analyzing the bugs and errors found in Ethereum contracts, the focus is on development peculiarities uncovered. Drawing inspiration from vulnerability testing methodologies and decades of real-world software security

practices, a list of "best practices" is developed and presented for implementing secure smart contracts. A series of known common issues is also outlined, supported by an algorithm for their detection.

Cybersecurity Measures for Smart Contracts

The entities and roles involved in the contract lifecycle of smart contracts are referred to as contract administrators, developers, and users, respectively. With similarities to paper or legal contracts, certain "best practices" are recommended for adaptive smart contracts. Resembling the automation of extraction and pickup events, smart contracts are typically built to utilize parties that leverage the computing power of specialized distributed software to abide by predetermined rules and conditions, allowing for consensus agreement. Unfortunately, smart contracts themselves are inherently vulnerable due to the fact that they are also software and, therefore, subject to the same type of errors and bugs to which any other software system could potentially fall prey. [17][18]

4.1. Code Auditing and Testing

The audit process is of crucial importance to ensure the quality of systems, and definitely, security is something that needs special care. There are several types of audits that can be carried out, such as line-by-line verification, code following a series of forms, test coverage plans, and so forth. Regular software must also be audited through penetration testing, code review of the entire platform, etc. This type of activity is quite common in several companies that develop systems for a wide variety of types, but what about smart contracts? Are they included? There is a series of risks that public smart contracts have when incorporated into several decentralized applications; the attack surface is much greater, and consequently, there are more targets spread throughout the network.

Furthermore, it is difficult for the team to follow the implementation of systems, requiring too much time, money, and effort to scrutinize all smart contracts interacted with by the application on the network. Unit testing is very important, primarily due to the simplicity and ease of use in the development and evolution process. Tests can be reused among developers. However, by requiring these systems to store a complex state, using specific types of oracles and feeds, it is observed that performing tests is relatively simple. Meetings and data analysis are difficult to perform. Furthermore, conducting these tests is a very complex task. It is recommended to use fuzz tests or consider the most important encoding style. It is very difficult to predict all possible inputs for the system. Fuzz testing the system will increase confidence in the development and subsequent team discovery. This project is one of the initiatives to ensure the quality and security of future systems.

4.2. Secure Programming Practices

Many smart contract security best practices originated from secure software engineering processes and best practices. It is crucial to start the application design with a threat model in mind, which can quickly help in identifying potential vulnerabilities, the level of access allowed for each interaction, how to isolate sensitive tasks and defensive tasks from the rest of the codebase, among others. During development, continuous and automated testing is crucial to ensure that the smart contract is behaving as expected. Security best programming practices in smart contracts and blockchain applications, in general, are not only important for the application but represent good practices for the underlying blockchain system security. Many issues faced by Ethereum smart contracts can also be tracked down to historical security issues by the underlying Ethereum blockchain platform. [19]

The following is a list of general suggestions on security best practices for programmers working in smart contracts. Understand smart contracts by writing the paper. Be fully aware of what is accomplished by the bytecode and the intention of its logic. The features and constraints of the system are important when deciding what style to use. Use linters or static analysis tools. They help in enforcing the rules previously set by the developers by analyzing security vulnerabilities such as reentrancy. Agile methodologies are important. Due to this type of work, developers and managers can't tell when the system is ready or how much it is costing. Increasing the rounds and reviews by stakeholders may help in discovering more bugs before deployment. Talk to other experienced developers. Learn about their experiences and about common mistakes discovered in many different types of applications. Audit smart contracts and audit security requirements. In teams working on smart contracts, there should be an additional member

working as a security requirements reviewer, analyzing the work done by the programmer, delivering checkpoints, and maintaining the security measures. Monitor third-party resources and vulnerabilities.

5. Decentralized Application Security

Decentralized applications (dApps) require a specific set of cybersecurity requirements due to their unique characteristics. Traditional security measures such as firewalls, intrusion detection systems, and centralized controllers are not available or useful in a typical dApp. The primary vulnerability specifically for decentralized applications is the misuse of smart contracts. Understanding and addressing smart contract vulnerabilities is one of the first steps in securing a dApp [20]. In this paper, we provide a detailed understanding of smart contracts and discuss these vulnerabilities. We also present some suggestions and research directions for future work to address these vulnerabilities. Decentralized applications are built on top of blockchain platforms, which apply decentralized cryptographic principles to establish a chain of transaction blocks recorded on a distributed ledger. Each of these blocks contains a timestamp, a cryptographic hash of the previous block, and transaction data. In a dApp, the backend code is also executed on the blockchain to interact with the smart contract, so the backend stands with the users' direct interaction with the smart contract.

5.1. Interactions between Smart Contracts and DApps

The interactions between smart contracts and DApps embedded in a blockchain system are fundamental to their communication and joint operation. They differ from the typical procedure calls or requests between regular software components and systems. Instead, the communication between smart contracts and DApps in a blockchain system is completed using a series of published actions, such as triggers of the execution of a function, message passing, and resulting events. Thus, to use a smart contract in a DApp, first, a control function should be defined in the DApp and the smart contract’s address should be known. The second step is to access the smart contract using the known API in a programming language. Then, the control function should be called and the strict mode should be set. Lastly, event watchers can be set in the DApp, and the DApp will immediately respond to resulting events emitted from the smart contract.

6. Case Studies

6.1 The DAO and the Parity Hack: Lessons that Developers and Users Failed to Learn This section presents a novel line of analysis: we aim to understand hacker behavior that converts a smart contract vulnerability into profit, rather than the technical details of the vulnerability itself. We study the two largest financial hacks involving Ethereum to understand how amateur and experienced hackers exploit smart contract vulnerabilities to abscond with investor funds. The DAO was the largest crowdfunding project at the time, raising $150 million before a known vulnerability allowed the hacker to siphon off around $50 million by replaying a recursive call bug within the smart contract. The attack could have been prevented if the developers hard-forked the blockchain to enable investors to recover their stolen ether. However, ideological disagreements within the community about blockchain immutability prevented consensus, and detractors forked the original network, creating a separate version. Perry is a smart contract developer with a background in pragmatic formal methods and extensive experience in security. Recall from section 2 the two novel functionalities of smart contracts. Firstly, they persistently store investment funds, and secondly, these funds are programmatically accessible to users with the correct credentials; resultantly, Kickstarter-like projects can safely and securely be conducted in a fully decentralized manner without an intermediary. Perry’s earlier multi-sig ether wallet contract was audited, receiving a certification of security, and it was considered to be one of the most secure designs on the network.

6.1. Ethereum DAO Hack

One of the most famous hacks of smart contracts, which resulted in the loss of $50 million and affected the Ethereum value on the digital currency exchange, was from the Decentralized Autonomous Organization (DAO) in June 2016. The DAO initiated large-scale projects on the platform, and those who donated were given voting rights in proportion to their investment. The number of investors grew to a point where they controlled the majority of voting rights, and they initiated a vote to transfer all funds from the DAO to a child DAO with the option of recouping the entirety after 28 days. An anonymous party used a recursive calling tactic and the splitting mechanism in a way that, under normal

circumstances, it should not have been processed until the following month. The amount of the original project funds was taken into account. The DAO invalidated all split rules by activating the same rules at the same time, ensuring that all of the splitters and the contract's refund writer would receive refunds.
Lines were exploited by allowing re-entrant calls. This is a tactic where a complicated operation is used to withdraw assets and then re-enter the application at a sensitive point, modifying conditions to continually reuse assets. In addition, the same code was able to exceed the maximum gas limitations for a call using the split function. [21]These two points led to the determination that after the project had been removed from the main project, the amount of Ethereum would be sent to the new split, but then the funds could be removed. Although the DAO funds were not permanently auctioned off, this was only caused when the community, i.e., the majority of stakeholders, conducted a difficult fork process around previous events. The blockchain was modified to ignore the hack results. However, the nature of blockchain centralization has been challenged, and decentralization has been abused. This has led to instances where, after a hack, the functioning of a smart contract was modified as a result of a difficult division to increase potential growth variability.

6.2. Parity Wallet Hack

Parity is a blockchain development firm that has a significant amount of control over some Ethereum tokens, as it has several existing smart contracts that contain the funds. The services provided by Parity were exploited in a hack. Tens of millions of funds from numerous multi-signature wallets were stolen. The exploitation was performed by reinitializing a multi-signature wallet and adding code to the associated exchange wallet.
The wallet exploit occurred from a code that allowed an attacker to leverage the affected multi-signature wallet to execute a library that could self-destruct. The library destruction led to all multi-signature wallets that relied on the library as a base becoming unusable, thereby rendering the wallets permanently frozen. Locked multi-signature wallets and the associated exchange wallets were simultaneously cleaned out and emptied. [22]

7. Regulatory and Legal Implications

It is uncertain as to the regulation, application, and implications of smart contracts and other decentralized applications of the blockchain or transactions by the regulator, at both a national and international level. Although regulatory bodies can create regulations, these often take time to implement, and the decentralization aspect creates significant difficulties. It is important for developers to consider smart contracts and other decentralized applications more now so that they can also work with the regulators to ensure that there is no breach of the regulations in the future. Analogous to an audit by a business in relation to their financial statements, this analysis is a supplementary obligation when considering or creating decentralized applications, and if done correctly, it benefits the business.
An organization with a legal obligation to disclose financial and legal reports upon which investors rely is doing so through centralized entities for these reports. If the blockchain is adopted for records, but there is no oversight or accountability to anyone or incentive to be correct, it will not provide the necessary level of security or assurance, hopefully not leading to Bitcoin being seen as disruptive as a Ponzi scheme [23]. There should be public and government representation on the governance structures, and unified financial reporting standards should be required and adopted to protect the investor. There will be attempts by a country to harness the capabilities of the blockchain, and this regulation is important. Accounting and auditing requirements will need to be reassessed. The blockchain will have the biggest impact on accounting and auditing since the introduction of IT. The aim of business should not be to keep the regulator out, but to work successfully with the regulator to provide security to investors.

8. Future Trends and Technologies

The power and usefulness of blockchain technology, combined with cryptographic and consensus techniques, have attracted growing attention across different sectors, from the technology of cryptocurrencies to supply chain management and from healthcare to energy trading. Meanwhile, artificial intelligence and the Internet of Things are revolutionizing our lives – from industrial machines making data-driven production decisions based on real-time status updates and cloud-based, remote monitoring of field assets to virtual assistants setting reminders for us to turn off the lights before we go

to bed, predicting the weather, or waiting for the next query. In these worlds of our present and future, these transformative technologies address a myriad of practical problems.

They help to improve the efficiency, performance, and reliability of the systems by providing exactly what is demanded. Consequently, using their counterparts and different techniques has further resulted in important research studies in the literature. Many of these transform the way people research and design smart contracts and decentralized applications. Consequently, in these periods, the growth motives, problems, or the current focus areas in blockchain and integral techniques are changing. Therefore, harnessing complementary algorithms, automated systems, unified models, and providing sophisticated testing tools for automated software testing across these combines a brief elucidation of the future trends and technologies. [24]

9. Conclusion

Blockchain is a concept that started with the use of Bitcoin, to mitigate the necessity of financial intermediaries and decentralize the currency. Thanks to its concept, it attracted great attention and financial benefits. With the intense interest seen in Bitcoin, the idea of having a transparent and safe database first emerged; based on the Blockchain concept. The fact that the data on this database, developed by creating a new chain for each transaction and it is stored in a distributed way, creates a more reliable database. These advantages of the used Blockchain technology caused the increase of the usage areas. For example, it was used in the solution of problems such as house sales and the number of people voting in the wrong names; especially for agriculture and programs focused on producer support. From this point of view, the focus of this study is on the energy trade by using the Blockchain technology.

Electronic transactions and the technologies supporting them became ever more heavy from that time on. The Blockchain is an electronic financial dealing supportive application. The Blockchain is a safe electronic database system created and saved by the group of people or devices of each member in that group of connections. The Blockchain provides a new horizontal and peer to peer level for online transactions. It is a time based chain of blocks which are processed by the computer servers designed repetitively by using hashes. The Blockchain with its smart contacts is mostly preferred instead of normal contacts for Bitcoin and other crypto money transactions. In this study, at first, it is summarized the emergence and the development of crypto money usage. The second step we have touched with the economic and energy policy importance of the blockchain technology. Hereupon, by using the real sector data, we have shown that; if the benefits of this technology is taken into account in policymaking and implementation of its strategy, in our country the Bitcoin have the potential to contribute to the sustainability of the current account deficit.

10. Recommendations and results

Initially, we had not planned to make controversial statements concerning the evaluated sources, but the existing reality made it clear that unqualified trust in sources of data does not guarantee a good outcome. If the data provided by a well-prepared source causes IoT devices to stop functioning upon withdrawal of the operator from the market, or a gas engine to fail due to an inappropriate bill being executed, then the purpose of establishing this source becomes questionable. It seems that prior to taking down the source, it is essential to develop a mechanism to switch over from the main resource to a different one without greatly disrupting the functioning of devices or businesses. This can be implemented using pools of resources providing the same data beforehand; then when one of the sources disappears, it will be replaced by another equally reliable one. The equipment which lacks professionals skilled at resource management constantly respond to fraudulent messages, from within and outside the network, by switching off gas losses accompanied by mechanical malfunctioning. Where virtual reality and augmented reality-based systems are concerned, the risk is even greater because they accept control commands from these very same sources. These commands can create significant deformations in the digital context - typographical rupture of devices and interruption of connections, as well as a breakdown in the hydroelectric system. Additionally, it is recognized that even if a source of data is qualified for efficient equipment, they cannot be reclassified the same way for other important contexts, such as that for pedestrians.

References:

[1] S. N. Khan, F. Loukil, C. Ghedira-Guegan, et al., "Blockchain smart contracts: Applications, challenges, and future trends," Peer-to-peer Networking, Springer, 2021. springer.com
[2] A. Singh, R. M. Parizi, Q. Zhang, and K. K. R. Choo, "Blockchain smart contracts formalization: Approaches and challenges to address vulnerabilities," Computers & …, 2020. github.io
[3] J. Li and M. Kassem, "Applications of distributed ledger technology (DLT) and Blockchain-enabled smart contracts in construction," Automation in construction, 2021. northumbria.ac.uk
[4] A. M. S. Saleh, "Blockchain for secure and decentralized artificial intelligence in cybersecurity: A comprehensive review," Blockchain: Research and Applications, 2024. sciencedirect.com
[5] I. Bashir, "Mastering Blockchain: A deep dive into distributed ledgers, consensus protocols, smart contracts, DApps, cryptocurrencies, Ethereum, and more," 2020. hoasen.edu.vn
[6] B. Farahani, F. Firouzi, and M. Luecking, "The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions," *Journal of Network and Computer*, 2021. [HTML]
[7] X. Wang, Y. Chen, X. Zhu, C. Li et al., "A Redactable Blockchain Scheme Supporting Quantum-Resistance and Trapdoor Updates," Applied Sciences, 2024. mdpi.com
[8] D. Van Der Linden, P. Anthonysamy, "Schrödinger's security: opening the box on app developers' security rationale," in Proc. ACM/IEEE 42nd, 2020. bris.ac.uk
[9] A. Bakhshi, "Securing Smart Contracts: Strategies for Identifying and Mitigating Vulnerabilities in Blockchain Applications," 2024. [HTML]
[10] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical review of automated analysis tools on 47,587 ethereum smart contracts," in Proceedings of the ACM/IEEE, 2020. [PDF]
[11] I. U. Onwuegbuzie, "e-Learning and Cyber Threat; A Call for Awareness and Mitigation.," Journal for Pure and Applied Sciences (JPAS), 2022. researchgate.net
[12] A. A. Hassanein, N. El-Tazi, and N. N. Mohy, "Blockchain, smart contracts, and decentralized applications: an introduction," in *Implementing and leveraging …*, Springer, 2022. [HTML]
[13] W. W. Wai, "The Implementation of Management Rights for Ethereum Forking Governance," Studies of Applied Economics, 2021. ual.es
[14] P. Qian, R. Cao, Z. Liu, W. Li, M. Li, L. Zhang, and Y. Xu, "Empirical review of smart contract and defi security: vulnerability detection and automated repair," arXiv preprint arXiv:2023. [PDF]
[15] L. Heimbach and R. Wattenhofer, "Sok: Preventing transaction reordering manipulations in decentralized finance," in *Proceedings of the 4th ACM Conference*, 2022. [PDF]
[16] S. Sayeed, H. Marco-Gisbert, and T. Caira, "Smart contract: Attacks and protections," Ieee Access, 2020. ieee.org
[17] H. Rameder, M. Di Angelo, and G. Salzer, "Review of automated vulnerability analysis of smart contracts on Ethereum," Frontiers in Blockchain, 2022. frontiersin.org
[18] M. Soud, G. Liebel, and M. Hamdaqa, "A fly in the ointment: an empirical study on the characteristics of Ethereum smart contract code weaknesses," Empirical Software Engineering, 2024. [PDF]
[19] L. Marchesi, M. Marchesi, and L. Pompianu, "Security checklists for ethereum smart contract development: patterns and best practices," arXiv preprint arXiv, 2020. [PDF]
[20] N. F. Samreen and M. H. Alalfi, "A survey of security vulnerabilities in ethereum smart contracts," arXiv preprint arXiv:2105.06974, 2021. [PDF]
[21] C. Pop, T. Cioara, I. Anghel, M. Antal, and I. Salomie, "Blockchain based decentralized applications: Technology review and development guidelines," arXiv preprint arXiv, 2020. [PDF]
[22] A. Dahiya, B. B. Gupta, and W. Alhalabi, "A comprehensive analysis of blockchain and its applications in intelligent systems based on IoT, cloud and social media," International Journal of …, 2022. wiley.com
[23] S. S. Smith and J. J. Castonguay, "Blockchain and accounting governance: Emerging issues and considerations for accounting and assurance professionals," … Technologies in Accounting, 2020. [HTML]

[24] A. G. Gad, D. T. Mosa, L. Abualigah, and A. A. Abohany, "Emerging trends in blockchain technology and applications: A review and outlook," *Journal of King Saud*, 2022. sciencedirect.com