Article

Blockchain Technology and Related Security Risks: Towards a Seven-Layer Perspective and Taxonomy

Sepideh Mollajafari * and Kamal Bechkoum

School of Computing and Engineering, University of Gloucestershire, Cheltenham GL50 4AZ, UK; kbechkoum@glos.ac.uk
* Correspondence: smollajafari2@glos.ac.uk

Abstract: Blockchain technology can be a useful tool to address issues related to sustainability. From its initial foundation based on cryptocurrency to the development of smart contracts, blockchain technology promises significant business benefits for various industry sectors, including the potential to offer more trustworthy modes of governance, reducing the risks for environmental and economic crises. Notwithstanding its known benefits, and despite having some protective measures and security features, this emerging technology still faces significant security challenges within its different abstract layers. This paper classifies the critical cybersecurity threats and vulnerabilities inherent in smart contracts based on an in-depth literature review and analysis. From the perspective of architectural layering, each layer of the blockchain has its own corresponding security issues. In order to have a detailed look at the source of security vulnerabilities within the blockchain, a seven-layer architecture is used, whereby the various components of each layer are set out, highlighting the related security risks and corresponding countermeasures. This is followed by a taxonomy that establishes the inter-relationships between the vulnerabilities and attacks in a smart contract. A specific emphasis is placed on the issues caused by centralisation within smart contracts, whereby a "one-owner" controls access, thus threatening the very decentralised nature that blockchain is based upon. This work offers two main contributions: firstly, a general taxonomy that compiles the different vulnerabilities, types of attacks, and related countermeasures within each of the seven layers of the blockchain; secondly, a specific focus on one layer of the blockchain namely, the contract layer. A model application is developed that depicts, in more detail, the security risks within the contract layer, while enlisting the best practices and tools to use to mitigate against these risks. The findings point to future research on developing countermeasures to alleviate the security risks and vulnerabilities inherent to one-owner control in smart contracts.

Keywords: blockchain; Ethereum; smart contract vulnerabilities; centralisation; one-owner control

Citation: Mollajafari, S.; Bechkoum, K. Blockchain Technology and Related Security Risks: Towards a Seven-Layer Perspective and Taxonomy. *Sustainability* **2023**, *15*, 13401. https://doi.org/10.3390/su151813401

Academic Editor: Marc A. Rosen

Received: 6 July 2023
Revised: 31 August 2023
Accepted: 3 September 2023
Published: 7 September 2023

1. Introduction

The notion of blockchain technology was introduced by Nakamoto, who published an article about cryptocurrency in 2008, and in 2009, bitcoin became the first decentralised cryptocurrency [1]. This emerging technology has recently received a great deal of attention from industry and academia because of its apparent benefits. The merits of decentralisation of control, reliability and consistency of data and transactions, immutability, and anonymity are well articulated in the literature [2,3]. Since the first generation of blockchain based on bitcoin, developers started to believe that blockchain could do more than simple currency transactions. The second generation of blockchain introduced Ethereum, an open and decentralised platform, which enables users to develop smart contracts using a programming language called Solidity [4]. With the introduction of smart contracts, blockchain technology gained significant attention, enabling mutually distrusted users to complete data exchange or transactions without the need for intermediaries [3]. A smart contract is a computer program written using a programming language, such as Solidity, that runs on a decentralised basis, and the overall state of the system is stored in a blockchain. A good description of the use of Solidity as a programming language for blockchain can be found in [5]. Currently, there are several platforms that can support smart contracts such as Ethereum, Hyperledger Fabric, Corda, Stellar, Rootstock, Polkadot, and Solana [6,7]. This work focuses on Ethereum, one of the most popular smart contract platforms.

Core to the technology is its decentralisation. Decentralised networks utilise spare computing power and storage resources from participants, reducing the need for centralised data centres and promoting resource sharing and energy efficiency. This can provide a more robust and scalable infrastructure to manage distributed energy resources [8] Moreover, blockchain underlying smart contracting systems can be particularly useful in enabling more flexible decentralised energy markets [9]. A good overview of the potential benefits that blockchain can contribute to sustainability can be found in [10]. Using the case of stock markets, the merits of decentralisation are further accentuated by Dodman et al. [11] as a mechanism to level up the playing field between the big corporates and the small traders by addressing both information and infrastructure asymmetries that exist in a centralised mechanism.

Notwithstanding its potential benefits for sustainability and for numerous business domains, and despite having some protective measures and security features, blockchain still faces significant security challenges. One of the main vulnerabilities that recently raised security concerns is centralisation in blockchain, which affects a number of conceptual layers within the blockchain architecture. In the application layer, the risk stems from centralised end user applications, which are provided by centralised organisations known as exchanges [8]. Centralisation also poses a risk within the consensus and incentive layers, which control the consensus power and incentive distributions [12]. In addition, in the network layer, centralised DNS services control DNS seed addresses and can cause an eclipse attack, as explained in [12,13]. Centralisation risks also affect the contract layer with "owner control", whereby developers and external attackers can exploit blockchain through contract ownerships [14,15]. Dodman et al. [11] proposed a Data Highway Protocol that boosts the transaction rate and thus supports decentralisation through the use of blockchain. Although this reinforces decentralisation, the work uses smart contracts as a reference, which can themselves be a source of vulnerability, as described in this paper.

A detailed description of security threats and vulnerabilities related to smart contracts is provided in Section 4.3 below. For smart contracts written in Solidity, although they are meant to be decentralised, developers can exploit the network to inject centralisation into the smart contract. This is the case because when digital assets are in the control of developers/owners, the risk moves to the smart contract itself. This makes smart contracts one of the major areas of security concerns in blockchain transactions, as highlighted by several researchers [3,4,12,16]. The threats of centralisation caused by smart contracts pose a direct risk to the technology' potential to support sustainability through its decentralised networks. To mitigate against this risk, it is important to tackle the security threats affecting the technology, in general, and its crucial decentralised feature, in particular.

This article attempts to develop a better understanding of the vulnerabilities within each of the blockchain layers. A blockchain network is normally conceived of as comprising of six layers: the data layer, the network layer, the consensus layer, the incentive layer, the contract layer, and the application layer. Using a low granularity level (i.e., fewer layers) is believed to pose a high risk of missing the source of vulnerabilities, and thus understanding the nature of the security threats. For this reason, and in order to develop a deeper understanding of the sources of vulnerabilities within the blockchain, a seven-layer architecture is adopted in this work. Given the prominent role that smart contracts play in completing transactions and the resulting impact on centralisation within the blockchain, a particular emphasis is placed on security threats caused by smart contracts in the contract layer. More specifically, the following research questions (RQs) are addressed:

RQ1: What is the best layering approach for a deeper understanding of security issues and the associated counter measures?

RQ2: What are the best practices to mitigate against security risks?

With the above questions in mind, a general taxonomy is devised for a seven-layer blockchain architecture, describing the inter-relationships between vulnerabilities, attacks, and the related consequences. The security impact of centralisation on the blockchain is discussed. Major security risks caused by centralisation, particularly within five specific layers of the blockchain, are identified. The impact of each vulnerability and threat is analysed and current detection tools and preventive measures examined. Of particular concern is centralisation in the contract layer, which causes major security risks, some of which are specific to smart contract implementation, which allows for a lack of appropriate access control. In other words, a smart contract code that allows "one-owner" to control access thus threatens the very decentralised nature that blockchain is based on. To mitigate against these risks, a model application is developed depicting the security risks within the contract layer, while enlisting the best practices and tools to use.

Following this introductory section, this paper is organised as follows. Section 2 sets out the adopted research method, while Section 3 is dedicated to the architecture of the blockchain technology, describing each of the abstract layers that provide the conceptual framework for the subsequent analysis. Section 4 sets out the initial research results, providing a comprehensive overview of the different vulnerabilities associated with each layer of a seven-layer blockchain. For each vulnerability, the potential consequences are highlighted and countermeasures are suggested, yielding a taxonomy outlining the inter-relationships between the vulnerabilities, attacks, and the corresponding potential consequences within each layer. Building on an overview of the results, Section 5 focuses on the contract layer—arguably the most vulnerable of the blockchain layers—and discusses countermeasures to alleviate the security risks and vulnerabilities inherent to smart contracts. Concluding remarks and recommendations for further research are found in Section 6.

2. Research Methods

To address the key research questions outlined above, a systematic literature review (SLR) was carried out to investigate and analyse the extant literature on blockchain technology, platforms, layering, and related key features. SLR is mainly based on secondary data using an inductive approach and an interpretivist research philosophy, as defined by Saunders et al. [17]. The review used 478 relevant academic papers from peer-reviewed research databases such as IEEE, ACM, and Science direct; conference papers; book chapters; surveys; and technical reports published between 2015 and 2023. Following a title and abstract screening, some of the articles were excluded based on their abstract. In this first iteration (abstract screening), a number of articles, although pertinent to blockchain, mainly focused on issues related to cryptocurrency and business considerations, including legal aspects. Other papers were not accessible due to being written in a different language. As a result, of the 478 full-text articles, 389 were shortlisted and accessed for data extraction. In the second iteration, 98 articles of the 389 shortlisted ones were excluded, leaving us with 291. Some of the articles within the short list were excluded because they focused on a layering architecture or security threats within platforms other than Ethereum environments. The key inclusion and exclusion criteria are shown in Table 1. The articles were analysed with a view to develop a comprehensive list of vulnerabilities/attacks and their consequences and detection tools, or preventive techniques, with a particular focus on issues related to centralisation. Emphasis was placed on the identified factors before collecting measurement techniques. The extracted data were mapped against appropriate layers of the blockchain. Out of all shortlisted articles, only 14 addressed centralisation risks within the application layer, the consensus and incentive layers, the network layer, and the contract layer. Only five articles covered centralisation risks in the contract layer. The steps used in the overall research strategy adopted for selecting the right publications are shown in Figure 1.

Table 1. Inclusion and exclusion criteria for blockchain technology.

Criteria for Inclusion	Criteria for Exclusion
The paper must be peer-reviewed and published in research databases. The technical report must be reviewed by reputable blockchain security analysis companies. The paper must contain information associated with blockchain technology or related to blockchain layering, key components, vulnerabilities and attacks on Ethereum.	Papers focusing on business or legal impacts of blockchain applications. Papers focusing on other blockchain platforms other than Ethereum. Papers written in a language other than English

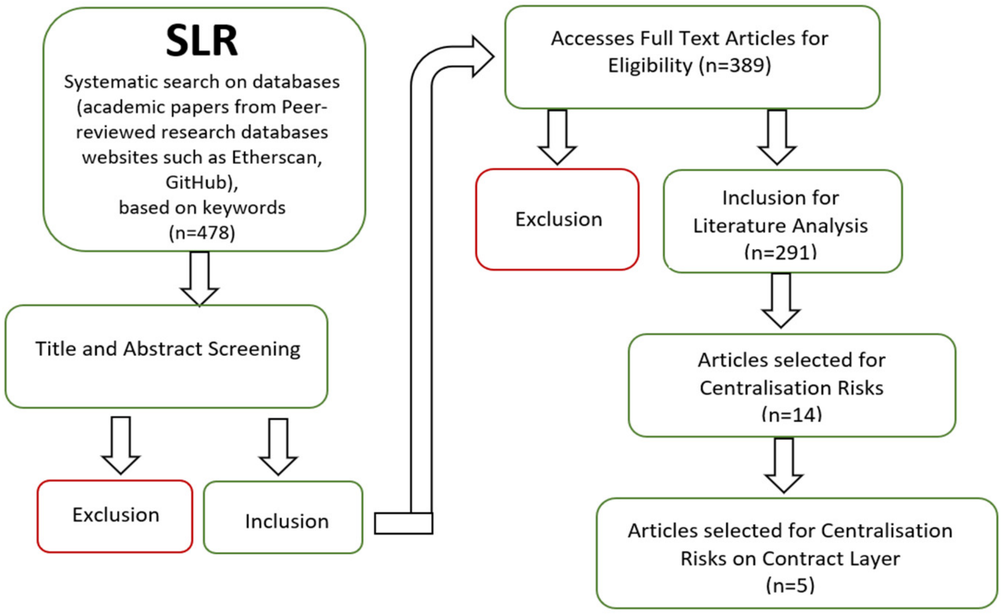
SLR
Systematic search on databases (academic papers from Peer-reviewed research databases websites such as Etherscan, GitHub), based on keywords (n=478)
Title and Abstract Screening
Exclusion
Inclusion
Accesses Full Text Articles for Eligibility (n=389)
Exclusion
Inclusion for Literature Analysis (n=291)
Articles selected for Centralisation Risks (n=14)
Articles selected for Centralisation Risks on Contract Layer (n=5)

Figure 1. Overview of the systematic literature review.

A thematic analysis was adopted for the identification of meaningful patterns linking threats to each of the blockchain layers. Although no specific coding was used, the thematic analysis approach was supported by a content analysis, yielding a classification of the key categories around threats, attacks, and countermeasures and how they are related to each layer. All data extracted helped to develop a more comprehensive and an in-depth classification of security threats and attacks within the different layers of blockchain. A complete classification of the available detection tools and preventive techniques was provided for each vulnerability. The main factors that caused centralisation risks are also identified. To collect the source code of the smart contracts, GitHub and Etherscan repositories were used.

3. Conceptual Framework: A Seven-Layer Architecture for Blockchain Technology

Most researchers describe the architecture as a six-layer model (e.g., [14,18,19]). Others condense the architecture into a four-layer model [4,20]. Other work [21], on the other hand, uses a seven-layer architecture, adding a physical layer to the six-layer model. Having reviewed the literature, this research adopts a seven-layer architecture as its conceptual framework, which provides a better granularity to account for all possible security risks. In order to develop a more detailed understanding of the sources of vulnerabilities within each of the seven layers of blockchain, an understanding of the role and components of each layer is needed. For this, a brief description of each layer is provided below, highlighting some of the key vulnerabilities that will be discussed in more detail in Section 4.

3.1. Application Layer

This layer comprises various forms of application scenes such as programmable currency, programmable finance, and programmable society. The introduction of smart contracts provides a great opportunity to implement blockchain solutions for use across different applications and industries [14,22]. Within this layer, threats are broad and can include internal and external attackers, malicious exchanges/service providers, malware, design, and configuration [20]. One of the risks is that users' assets can be controlled by the exchange operator (or malicious operator), which provides full control over the funds on their servers [23,24]. Currently, large centralised exchanges lead to centralised staking activities, with large companies potentially having the majority share of the network [25]. To eliminate the single point of failure, which emanates from centralisation [20], it is important to use decentralised exchanges. However, they may contain some vulnerabilities that come from smart contracts or other features of blockchain.

3.2. Contract Layer

The contract layer contains components such as script code, smart contract, and algorithms. In order to run a smart contract, the codes should compile to the low-level bytecode that executes in the Ethereum virtual machine (EVM). Once compiled, the smart contract deploys on the Ethereum blockchain and is identified by a unique contract address generated upon a successful creation transaction [26,27]. Algorithms define the mechanism for all participating nodes to interact with each other and set relative execution and data resource. When the pre-defined rules are met, the relative operation will be performed in the network [14,28]. As mentioned in the Introduction, centralisation poses a serious risk to this layer, with "owner control" enabling developers and external attackers to exploit the blockchain through contract ownerships.

3.3. Incentive Layer

To ensure security and decentralisation, the blockchain system needs a large number of honest nodes (greater than 50%) to verify and validate each transaction. Incentive mechanisms are required to motivate nodes to participate in maintaining the safety of the system [29]. Therefore, the incentive mechanism plays a vital role in blockchain system to ensure that the majority of the network is honest [12]. However, to increase their chances of mining, individual miners use a mining pool to increase the chance of obtaining any reward from block creation. This process leads to a centralised point, at which mining power and control over incentive distribution would be in the hands of a few individuals in the blockchain network [29]. Furthermore, if honest nodes withdraw from being active miners, it will impact on the value of hashing power of the network. As a result, the distribution of rewards can be skewed towards a few participants, namely, a small number of participants that are part of a mining pool. This leads to centralisation of hashing power of the mining pool, reward centralisation, and control over the network [12,30]; ultimately, this may decrease the number of participations due to unfair incentive distribution, thus reducing security due to centralisation in mining pools.

3.4. Consensus Layer

The consensus layer contains various algorithms that are utilised so as to ensure all nodes reach an agreement regarding the data validity of newly generated blocks in the decentralised network [3,14]. As part of the mining process, blockchain networks use a consensus-based agreement to verify and add new blocks successfully to the chain [31]. This agreement among nodes is achieved using proof of work (PoW), which works based on the hash rate. Miners use a significant computational power and compete to solve a cryptographic puzzle (find a nonce value) to verify transactions and add a new block. With more nodes joining the network, mining becomes harder. A few individuals of mining pools would be able to combine their computational powers and control a large proportion of (hold 51%) of hash rate. This process affects the security and the decentralisation of the network [28,31,32].

The PoW algorithm requires a significant amount of computational power to verify transactions and add a new block to the ledger [29]. Ethereum announced in September 2022 that it moved to Proof of Stake (PoS). With the PoS mechanism, Ethereum relies on validators, not miners, to add new blocks to the chain [33]. PoS promises to provide a more energy-efficient system with a short consensus time that reduces the centralisation risk [34]. A new consensus algorithm that speeds up the consensus process to speeds that compare favourably to those of PoW or PoS is proposed in [11]. Centralisation remains a risk, however, because the validation of blocks is controlled by validators who hold the majority of the token and they have staked their coins (or other assets) through large centralised exchanges [25,29,33]. Researchers [35–37] have categorised consensus algorithms into two types: proof-based and voting-based algorithms.

3.5. Network Layer

The network layer comprises transmission protocols, a propagation mechanism, and a verification mechanism. These protocols and mechanisms are deployed using a Peer to Peer (P2P) network for data transmission and verification across the distributed nodes [38]. Transmission protocols allow blockchain nodes to communicate directly with each other and to synchronise data among them. Each node has the opportunity to broadcast blocks or transactions in a shared ledger. Transmission protocols help nodes to be aware of all the data and broadcast only valid data to the network [26,38–40].

As part of the communication between nodes on a P2P network, a node discovery protocol is required. This protocol works based on DNS protocol that distributes the address of other active nodes on the network [12]. DNS itself is a weak protocol that suffers from a security and privacy point of view due to its weak verification mechanism, which can be exploited by malicious developers to make it centralised [41].

3.6. Data Layer

This layer acts as the blockchain data structure. A block is a collection of valid transactions in a shared ledger, made of a block header and a block body. A header is composed of several components such as block version, Merkle root hash, timestamp, Nonce, bits, and asymmetric encryption. The block body holds a long list of transactions [14,18,42]. Security concerns are linked to creating a modified version of the transaction signature or hash collisions, as explained later.

3.7. Physical Layer

The physical layer is the actual medium that transports the bits. The main component of this layer is IoT devices, which connect to the internet and act as nodes on the blockchain. The decentralisation of the blockchain is made possible by smart contracts that translate the existing contractual clauses into embedded hardware and software [43]. To establish a connection with a blockchain-based system, all IoT devices need to interact with smart contracts and perform a digital signature and additional authentication processes [18,44]. Lack of integration of IoT devices in the blockchain poses a threat to device security and data privacy.

4. Results: Security Analysis within Blockchain Layers (Towards a Taxonomy and Model of Key Concepts)

4.1. Vulnerabilities and Attacks in Seven-Layer Blockchain

As highlighted above, the decentralised nature of blockchain has the potential to contribute to sustainability initiatives in various ways. Decentralisation offers transparency, security, and decentralised decision making, which can be advantageous for promoting sustainable practices and addressing global challenges. The use of smart contracts to manage and allocate funds transparently ensures that resources are efficiently directed towards sustainable projects, and stakeholders can track how funds are being utilised. Using smart contracts, stakeholders can vote on funding projects based on their alignment with sustainability goals. This will only work if adequate measures are put in place to counter the security threats to decentralisation and smarts contracts, hence the need for a deep dive into the architecture of blockchain looking at each component of a seven-layer blockchain. This section describes the initial research findings, providing a comprehensive overview of the different vulnerabilities associated with each of the seven layers.

In the current literature, most blockchain architectures are presented as comprising between four to six layers. An exception is the work in [20], as highlighted above. This poses a high risk of missing the source, and thus understanding the nature, of the security threats. Having a more granular architecture enables a closer look at the components of blockchain, and a more detailed examination of security risks and their location within the architecture. Therefore, a more detailed architecture, comprising seven layers, was adopted, as depicted in Figure 2.

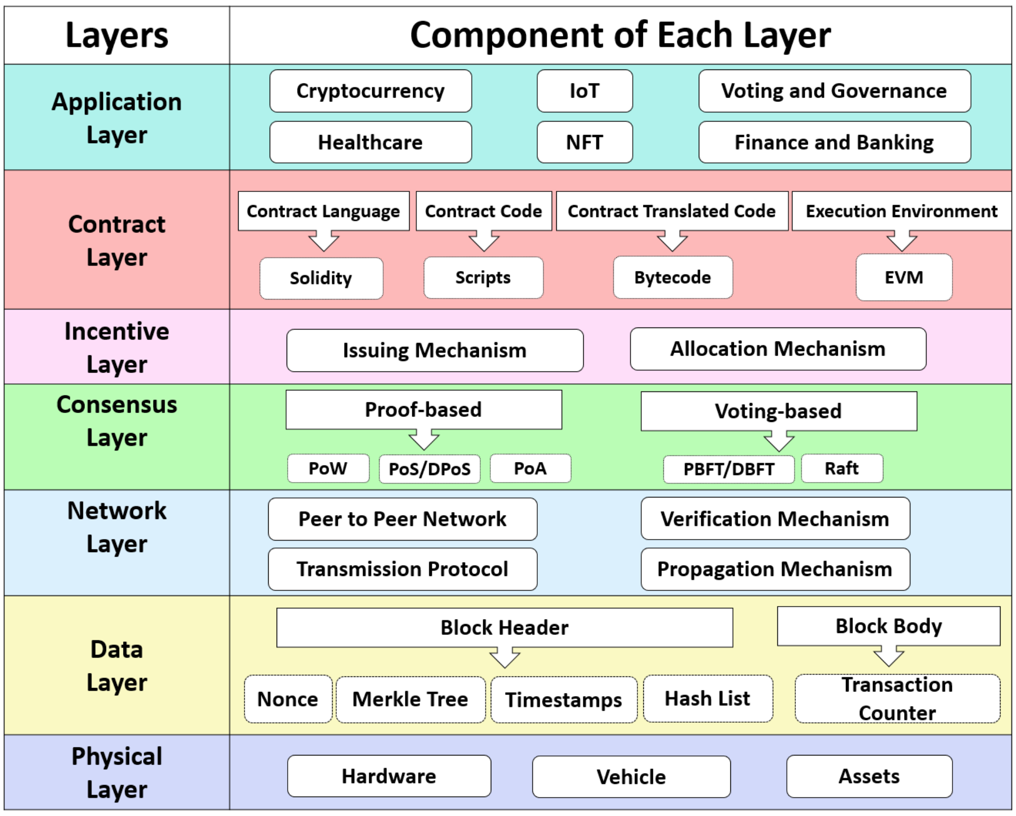

Figure 2. The seven-layer blockchain system architecture, adapted from [7,14,18–21].

Ethereum vulnerabilities and attacks are outlined based on their location. Their root causes and consequences are analysed, and the possible detection tools and preventative techniques, drawn from the literature, are discussed. Figure 3 provides a summary of attacks/vulnerabilities associated with each of the seven layers of the Ethereum blockchain, which are described in detail in Sections 4.2–4.8.

This was used as a basis for developing a taxonomy of the Ethereum vulnerabilities/attacks and their consequences, as discussed below. This taxonomy is shown in Figure 4 (Section 4.9), summarising existing work on detection tools and preventive techniques used for securing Ethereum systems for each of the seven layers.

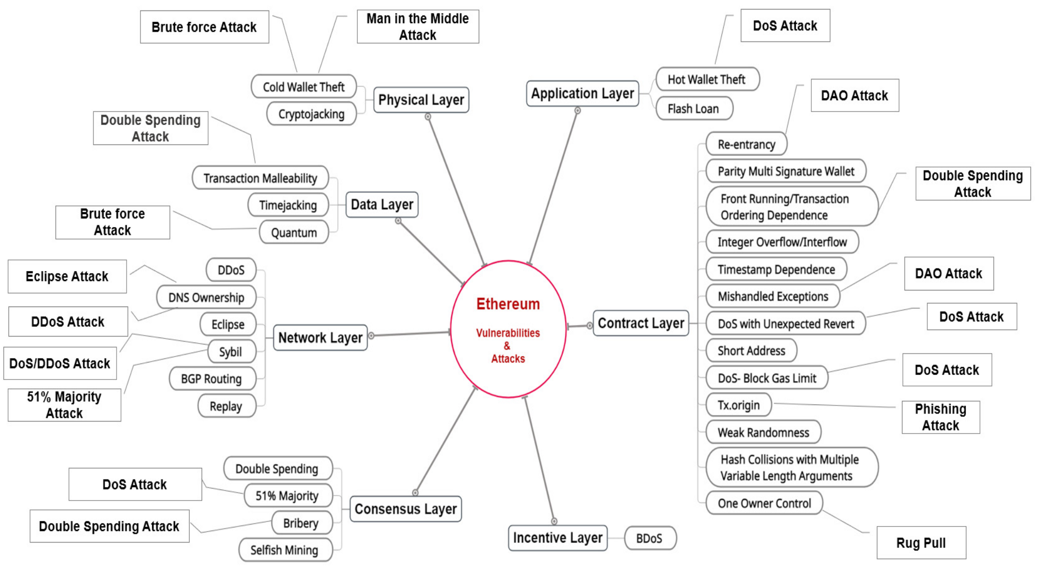

Figure 3. Vulnerabilities and related attacks within each of the seven layers of the Ethereum Blockchain.

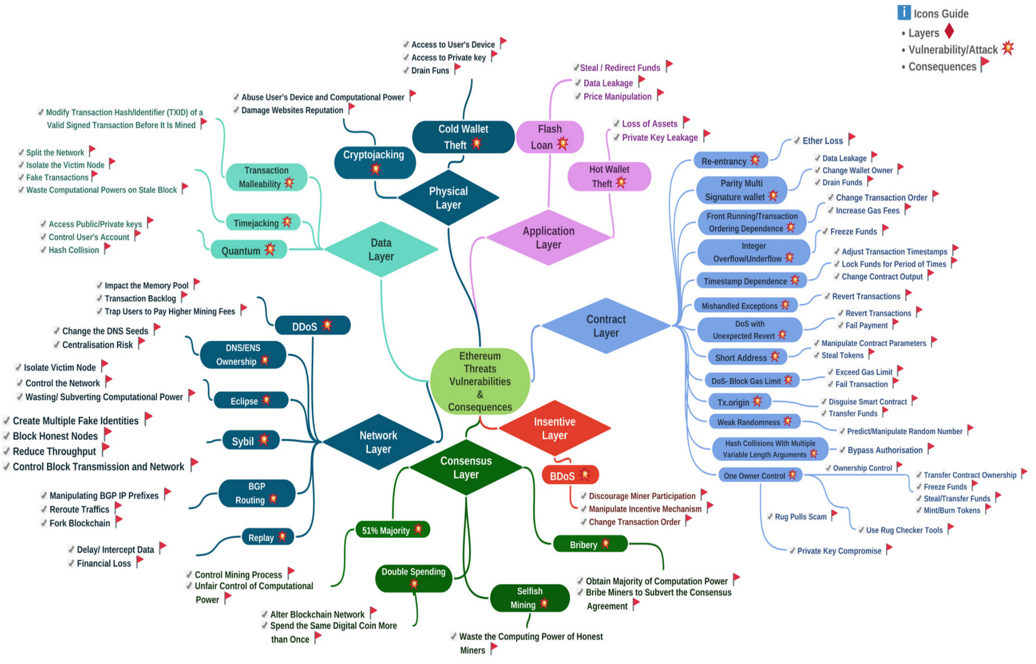

Figure 4. Vulnerabilities, attacks, and consequences: a taxonomy for the seven-layer architecture.

4.2. *Vulnerabilities/Attacks on the Application Layer*

4.2.1. Hot Wallet Theft

A crypto wallet is used to store and manage private keys. There are several crypto wallets with different security levels, such as the hot wallet, cloud wallet, paper wallet, and hard wallet [45]. Ethereum remote clients (mobile/browser wallets) are able to manage private keys, broadcast transactions, and interact with smart contracts, but are not able to store the full Ethereum blockchain [26]. As the crypto wallet is simply used for a key storage, when connecting to a transaction network, it is vulnerable to key theft, causing loss of assets in wallets [1,46]. Reports from cryptocurrency exchanges, such as Bilaxy exchange and AscendEX, stated that tokens had been lost from Ethereum via hot wallets [47,48]. Therefore, it is vital that exchanges keep most funds in cold storage.

4.2.2. Decentralised Finance (DeFi) Flash Loan Attack

DeFi relies on smart contracts and uses automated protocols to provide financial services without intermediaries. A flash loan is uncollateralised and unsecured loan in a DeFi system that allows borrowers to take loans without needing upfront collateral and then repay the loans with a single blockchain transaction, guaranteed by a smart contract [49]. DeFi poses security risks on the Ethereum blockchain due to smart contract weaknesses and new unsecure protocols such as MakerDAO. Flash loan attacks can lead to the following:

- Data leakage via phishing: attackers attempt to trick users and direct them to a fake website to access user's sensitive data, such as private key [49].
- Market price manipulation: attacker borrows a large amount of digital assets via flash loan and uses that fund to manipulate the price of that specific asset on a certain DeFi platform. Furthermore, a malicious arbitrage or attacker create an arbitrage opportunity and manipulate the token price. If greedy arbitrageurs do not have large sums of tokens in their wallet, they use flash loan to borrow tokens to leverage their trading position sizes and gain more profit [48]. There were a number of DeFi attacks that happened in 2020 and 2021 [50,51].
- Stealing or redirecting funds: bugs or vulnerabilities within a smart contract provide a great opportunity for attackers to steal or redirect funds [49,50].

4.3. Vulnerabilities/Attacks on Contract Layer

4.3.1. Re-Entrancy Vulnerability

"One of the features of Ethereum smart contracts is their ability to call, and utilise, code from other external contracts" [23], p. 173. The attack happens when attackers create a contract at an external address that contains malicious code using the fallback function. As a result, attackers would be able to have control of this vulnerable contract and call back into the original function, and invoke the same function again continually before the state has been updated. As a consequence, attackers can drain the contract's funds and the honest accounts lose Ether [4,26,52]. The most vulnerable built-in functions contain *transfer*(), *call*(), and *send*(). Among these three functions, the *call* function is more vulnerable [53].

4.3.2. Parity Multi-Signature Wallet

As users' personal information and daily withdrawal limits are stored on wallets, users should have multiple signatures or multiple private keys to own a multi-signature wallet to withdraw digital assets from the wallet [53]. As a parity multi signature wallet depends on the public library, the centralised setup of this weak library coupled with the non-restricted calls to the external wallet library functions have made the parity multi signature wallet a target for attacks [4,54].

4.3.3. Front Running/Transaction-Ordering Dependence

Transaction ordering is a race condition attack whereby malicious nodes increase the transaction gas price and try to select and execute own transactions first [52]. In the Ethereum, miners can use their power to choose transactions and order them based on the highest gas price to obtain more profit and pose frontrunning attacks [26].

When a transaction is broadcast to the Ethereum network, it goes to the Mempool (or Memory pool, a repository of unconfirmed transactions). In this type of attack, malicious nodes observe transactions and their details that are visible in the Mempool. Attackers are then able to control the order of transactions, they will select their own transactions and order them in a way that is beneficial for them. As a result, high fees paid for priority transaction ordering poses a security risk, including double spending attacks [26].

4.3.4. Integer Overflow and Underflow

Both Solidity and EVM support integers up to 256 bits. Integer Overflow and Underflow vulnerability happen when the number of bits is incremented higher than the maximum value or below the minimum value, respectively [54–56].

4.3.5. Timestamp Dependence

When a block is mined successfully, the miner has to provide the timestamp for the block. The miner will check the timestamp of a new block after mining and carry out the verification process to make sure that the timestamp of the new block is larger than the timestamp of the last block and that the local machine timestamp is not greater than 900 s [54]. The vulnerability happens in the Ethereum when malicious miners can adjust the timestamp of a new block to manipulate the outcome of timestamp-dependent smart contracts [26,52,54,57]. This vulnerability can increase the probability of front running attacks [26].

4.3.6. Mishandled Exceptions

This Solidity vulnerability is known by other names in different literature, such as "Unchecked send", "Unchecked External Call", and "Exception Disorders" [58]. An Ethereum smart contract performs an external call by using "call", "transfer", and "send" functions to fulfil the required functionalities. The exception handling is based on the execution of callee contracts and the interaction between contracts [4,58]. Therefore, it is important how a function is called and how exceptions are handled. Out-of-gas exception is one of the famous exceptions in the Ethereum. If an exception occurs in the callee, it may or may not propagate to the caller. The calling transaction will thus terminate entirely and revert the state and all gas is lost [7,54,58]. Other authors have stated that a mishandled exception may cause Denial of Service (DoS) attack on the on-going contract [21,55,59].

4.3.7. DoS with Unexpected Revert

This issue appears when a transaction is reverted due to improper handling of an incomplete transaction [59]. When Ether is sent to a contract, the fallback function or other functions should execute. If the execution of the caller contract fails, the contract' fallback function only performs the *revert*() function, which can disrupt the execution of the caller contract and cause a DoS state in the caller contract [55,60].

4.3.8. Short Address—Parameter Attack

A weakness of EVM is causing short address vulnerability, which happens when a contract receives encoded parameters that are shorter than the expected parameter length. If EVM detects an underflow, it adds a zero to the end of the encoded parameters to make up the expected length (256 bits). A malicious user can take advantage of this vulnerability by removing the last zero from the Ether [26,61].

4.3.9. Denial of Service—Block Gas Limit

As mentioned earlier, Solidity uses *send*(), *transfer*(), and *call*() functions to transfer Ether to externally owned accounts (EOAs) or between smart contracts. A contract would receive Ether by executing either the fallback or receive function. The payable modifier is used in Solidity to ensure that the function can send and receive Ether [62]. EVM allocates gas at the start of execution. Each block in the Ethereum has an upper limit on the amount of gas that can be spent for computation. The gas limit per execution is 2300, and both send and transfer functions forward 2300 gas to the receiving contract to complete the operation. The block gas limit prevents the security risk involved in executing expensive state changing code in the fallback function of the contract receiving the Ether. However, if the gas usage of a transaction exceeds this limit, the transaction will collapse, which may lead to a DoS attack [62]. Nonetheless, there is no gas limit associated with the "call" function, making it more vulnerable [62].

4.3.10. Tx.origin

Tx.origin is a global variable on Solidity that returns the address of account that sent the call or transaction. Using the tx.origin variable for authentication makes the smart contract vulnerable to phishing attacks [26]. A malicious contract can trick the victim by sending Ether, when the victim sends a transaction to a malicious contract, it will invoke the "*fallback*" function and call the "*withdraw*" function of the phishable contract and transfer to itself all the funds belonging to another address [26].

4.3.11. Weak Randomness

Blockchain uses randomness to process cryptographical tasks [63]. Ethereum produces 256 random bits by using the underlying operating system's random number generator to create keys. Most of Ethereum contracts are open source and the variables are public on blockchain. Therefore, it is vital to find a secure source of entropy or randomness to create keys, otherwise attackers/malicious miners can easily predict the generated random number [59]. For example, malicious miners can control *block.timestamp*, *block.difficulty*, *blockhash*, and *block.number* [64]. Several methods have been used to generate pseudorandom numbers, but weak randomness remains an issue that can lead to centralisation risks [65].

4.3.12. Hash Collisions with Multiple Variable Length Arguments

Hash Collision happens if two separate input strings of a hash function produce the same hash output [66]. Data are encoded according to its type and Solidity provides some global functions to encode various data types. Application Binary Interface encoding functions (ABI) can be used to interact with contracts and the external contract call on the Ethereum [67]. The *abi.encodePacked*() function is a non-standard packed mode that performs packed encoding of the given arguments and returns the packed encoding of the data as bytes [67,68]. This function can lead to hash collision in specific situations when different parameters return the same value/encoding, yielding signature match and making the attacker an admin [66,69]. In a signature verification situation, an adversary can exploit this by adjusting the position of elements in a previous function call to effectively bypass authorisation [66].

4.3.13. One Owner Control—Centralisation

As mentioned earlier, Solidity is used by the Ethereum, and it is one of the most popular smart contract platforms. Unfortunately, Solidity was not designed with a permission-based security model in mind [70]. Lack of a stable security mechanism, such as access control, makes smart contracts vulnerable [71]. Therefore, smart contract developers implement access control checks based on their judgment, and in an ad hoc manner, leading to several vulnerabilities, called access control vulnerabilities/bugs [70]. With access control in the "hands of the owner/developer", they would be able to access critical functions, perform sensitive operations such as "moderating the smart contract", "minting tokens", "burning tokens", "transferring ownership", "setting any address as validator", "voting on proposals", "freezing funds", and many other operations [72,73]. Access to these critical operations poses serious security risks caused by the centralised ownership of the smart contract. This includes the possibility of the owner acting maliciously or making errors that compromise the contract's integrity. *AChecker* is proposed in [70] as a mechanism for a more efficient overall access control, including one-owner control. However, the proposed mechanism does not resolve the issues emanating from one-owner control due to its focus on general access control. Other works, such as [12], focus on measuring the negative impact of one-owner control.

4.4. Vulnerabilities/Attacks on Incentive Layer

BDoS Attack

A Blockchain Denial of Service (BDoS) is an incentive-based attack, whereby the malicious actor manipulates the incentive mechanism [74]. The malicious attacker invests resources by generating a block and only publishes a proof that s/he mined, without publishing the block itself. This, to honest miners, is regarded as an advantage gained by the malicious actor, which leads to reducing miners' incentive to mine. As miners cease to mine, the entire blockchain can grind to a halt. Incentive-based attacks can force a certain order of transactions or transaction omission [74].

4.5. Vulnerabilities/Attacks on Consensus Layer

4.5.1. Double-Spending Attack

Double-spending refers to the risk of the cryptocurrency being spent twice. The attacker would send a copy of the currency transaction to make it look legitimate, thus disrupting the blockchain network and, essentially, stealing the cryptocurrency. There are mainly three different types of double-spending attacks: the Race, the Finney, and the Vector [14].

4.5.2. 51% Majority Attack

Here, the malicious actor is in a position to control (at least) 51% of the computing power so as to control the mining process [14]. They would create a chain of blocks that is fully isolated from the real (honest) version of the chain. Using their 51% advantage, they can process their blocks faster, and with time, the isolated (malicious) chain is established as a genuine one. Many regard the 51% majority as a form of double spending [14]. Malicious miners can perform a full-fledged DoS attack through controlling a majority of mining power, generate empty block, and ignore other blocks [74]. In [75], an "agreement algorithm" is devised as a basis for a scheme to strengthen resilience against 51% attacks.

4.5.3. Selfish Mining Attack

Malicious miners can compromise the blockchain to obtain higher block rewards [76]. One of the drawbacks of consensus mechanisms such as PoW is that miners are able to collaborate with each other and use a set of selfish strategies to gain more rewards than they would otherwise do if they mine individually. Such miners are called selfish miners and their "illegitimate" mining collaboration is called selfish mining. This is not fair for the other honest miners who stick to the rules specified by the consensus mechanism used [14]. The Data Highway Protocol, proposed in [11], has the potential to reduce the risks of selfish mining.

4.5.4. Bribery Attack

Attackers can increase the probability of double-spending by bribing other miners [76]. Several mechanisms for bribery have been proposed, with various trust and risk properties [77–79]. The evaluation of these different bribery mechanisms remains problematic due to the lack of systematic methods to quantify them. Bonneau [78] presented a few schemes to render bribery attacks ineffective. Such schemes, coupled with the fact that PoW makes it very costly for a bribery to be set, make it fair to say that bribery attacks are not the worst "headache" for the consensus mechanism.

4.6. Vulnerabilities/Attacks on Network Layer

4.6.1. DDoS Attack

As with any network infrastructure, the blockchain network layer is vulnerable to distributed denial of service (DDoS) attacks. Such attacks can impact the memory pools and cause a massive transaction backlog and trap users into paying higher mining fees [80].

4.6.2. Domain Name Service—Centralisation

The Domain Name System (DNS) plays a vital role in the internet. Nodes on peer to peer networks are communicating with other contributors to transmit data through the node discovery protocol. This protocol works based on DNS seed addresses that distribute the address of other active nodes on the network [12]. Researchers explained that the current DNS system is vulnerable to many attacks such as eclipse attack, DDOS attack, cache poisoning attack, single point of failure, and centralisation [81].

Current DNS suffer security and privacy issues due to the poor process of node discovery protocol, a weak verification mechanism that leads to a cache poisoning attack and makes domain owners observe nodes on the network, claim their domain ownership, and change the IP addresses of their domains. As secure DNS are not yet in place, this would move ownership and control of the authentication keys to the user's security domain, and results in centralised DNS services that can act as a single point of failure, which makes legacy DNS vulnerable to DDoS attacks [81,82]. Blockchain-based DNS assist to minimise some of the security concerns. Blockchain-based ENS, which is a distributed, decentralised naming system built on the Ethereum blockchain, provides decentralised ownership. However, because ENS is stored on a smart contract, the ENS registry contains a list of domain names, subdomains, important information about owner of domain name, the resolver of the domain, and the caching time for all records under the domain [40]. ENS relies on a smart contract to manage domain name ownership [40]. Therefore, it may be controlled/manipulated by a malicious developer/owner or attacker.

4.6.3. Eclipse Attack

In an eclipse attack, the malicious actor attempts to own plenty of IP addresses to take control of all honest node connections. Adversary node isolates a node and manipulates it into illegitimate action. Attackers typically use botnet to compromise the node and seal it off. The victim node is isolated within an environment that is completely separate from the actual network activity. Because the attack relies heavily on exploiting the victim's neighbouring nodes, its success will depend on the structure of the blockchain network [14].

4.6.4. Sybil Attack

In a Sybil attack, the malicious actor(s) can take over the entire network. Attackers are then able to out-vote the honest nodes if they create multiple fake identities (or Sybil identities). They can then control the reception and transmission of blocks, effectively blocking other honest users from the network [14]. The malicious pool operator can add a large number of miners with zero power into a mining pool and run a Sybil attack. These miners cannot mine any blocks, they can participate in data propagation for malicious users and stop propagating honest users' data. Therefore, only the attacker's block would join the network and the attacker receives higher rewards and decrease the throughput of network [83]. This attack may lead to several attacks such as DoS, DDoS, and 51% majority [83].

4.6.5. BGP Routing Attack

The Border Gateway Protocol (BGP) is a routing protocol used to exchange routing information (IP packets) among autonomous systems (ASes) on the internet [84]. A BGP routing attack, known as BGP hijacks or prefix hijack, can happen when a malicious AS broadcasts a fake IP prefix announcement and propagates the wrong routing information. Thus, the network can be split into two or more disjoint components, controlling communication within components, and rerouting the traffic and blockchain forks into parallel chains [14,84].

4.6.6. Replay Attack

Replay attack is more likely to happen during a hard fork when the blockchain is split into two, when a malicious actor spoofs the communication between two valid nodes and gains access to the hash key [26]. The adversary captures a signed message and attempts to delay or retransmit data as a valid user to subvert the receiver [85].

4.7. Vulnerabilities/Attacks on Data Layer

4.7.1. Transaction Malleability Attack

This attack can be associated with either or both the network layer and the data layer [14]. A transaction carries data that are stored on the blockchain. To protect this data, blockchain uses cryptography. Transaction ID (TXID) is given to every transaction that is verified and added to the chain. It is an illegitimate modification to a transaction that is being broadcast, prior to being accepted in a block. In a blockchain peer-to-peer network, transactions are passed from one node to another. A malicious node receives the transaction and creates a modified version of the signature by altering the transaction identifier (TXID), before passing it to other nodes in the blockchain [14,86]. The consequence of a successful transaction malleability attack can result in additional attacks such as double-spending [87].

4.7.2. Timejacking Attack

Timejacking happens due to the vulnerability of timestamp processing in a blockchain. All of the participant nodes in a blockchain network maintain a time counter, which displays the network time. Hackers can add multiple Sybil nodes to the network and alter the node time at the same time. This can slow down the median time of the targeted node by sending inaccurate timestamps, as well as splitting the network into several parts and isolating the targeted node from the network [14]. Thus, miners are wasting computational power on stale blocks and the network suffers fake transactions [88].

4.7.3. Quantum Attack

Attackers can launch a quantum attack on the cryptographic part of blockchain to calculate the private key from the public key by using Shor's algorithm. The level of the risk in the Ethereum is high and quantum attackers can launch this attack to do hash collision. They can take complete control of an account and drain all of the funds [14]. Researchers are working on post-quantum cryptography to protect blockchain systems against quantum attacks [89,90].

4.8. Vulnerabilities/Attacks on Physical Layer

4.8.1. Cold Wallet Theft

With the aim of using hot wallets on portable devices, attackers attempt to use different techniques to disrupt the confidentiality, integrity, and availability of valuable assets on wallets. As the software wallets store the keys on a computer or smartphone, they are more vulnerable to security breaches. The alternative, which is an offline wallet or cold wallet, is introduced to users. A cold wallet, which is a more secure wallet, has no internet connection and transfers keys and transactions through a USB stick, Bluetooth device, or smart card with special embedded software to perform cryptography functions [91]. However, cold wallets suffer from a lack of secure backup and recovery process of private keys. Some cold wallets use a terminal such as a smartphone or a computer to communicate with the user. Hackers can then capture NFC wireless communication or install malware on the terminal and perform a Man-In-The-Middle attack. Another vulnerability is the brute force attack used to work out what the passphrase is [45]. Moreover, wallets are hosted in an operating system and the running environment may be exploited, resulting in a security threat posed to the crypto wallet [92].

4.8.2. Cryptojacking Malware

Cybercriminals employ various techniques to hijack the computational resources of target devices to mine cryptocurrency. Attackers use two types of cryptojacking malware. They install an application on a target device (executable-type cryprojacking) that computes hashes secretly or they use browser-based cryptojacking. In the latter case, users visit the infected website and provide their CPU power to compute hashes [93].

4.9. Towards a Conceptual Taxonomy and Classification

This section provided a comprehensive overview of the different vulnerabilities and attacks associated with each layer of a seven-layer blockchain. For each vulnerability, an explanation is given about how it is exploited and the potential consequences of such exploitations. Defensive methods are described and countermeasures proposed. An overview of vulnerabilities, attacks, and their consequences is depicted in the taxonomy shown in Figure 4.

5. Discussion: Security Risks Associated with Smart Contracts in the Contract Layer

Considering the prevalence of smart contracts and their related security risks, and taking into account the vulnerabilities and attacks within each layer, as outlined in Section 4 above, the contract layer is, arguably, the most vulnerable layer in a blockchain architecture. This is, in part, a consequence of the fact that smart contracts are prone to security vulnerabilities due to the high dependence on programmers and exposure to bugs [52]. Based on the nature of blockchain-based programs, once smart contracts are deployed, they cannot be modified. It is argued, therefore, that particular attention should be paid to security risks emanating from smart contracts. This section provides a detailed account of the current work on the security risks and countermeasures associated with smart contracts. This is then used as a basis for developing a more detailed model application for smart contract security risks within the contract layer. The model is described here outlining the best practice towards developing more secure smart contracts.

As smart contracts are still recent, new bugs and security risks are constantly being discovered. This has led to developers using several smart contract security tools to check and validate the code and detect some of the vulnerabilities. As described in Section 4, the literature review revealed that different techniques and tools exist to detect vulnerabilities within the contract layers. These are summarised in Table 2.

To alleviate the risks associated with smart contracts, recommendations include manual code review to detect bugs, as well as checking access control to critical functions and the flow of function calls. Researchers have also suggested using testing frameworks such as foundry and hardhat to run tests and debug solidity code [94]. Source code metrics can be used for quality assurance and the performance of blockchain-oriented software (e.g., measure complexity and to calculate smart contract resource consumption such as gas in the Ethereum system) [95]. The SWC Registry provided smart contract weakness classification, which includes real-world smart contracts as test cases for each vulnerability [96]. It is vital that developers, researchers, and auditors use available smart contract security techniques/tools, best practices, and remediation steps, as suggested by CWE [96], ConsenSys [97], and Mastering Ethereum [26].

Table 2. Current work on vulnerabilities/attacks within the contract layer and related Counter-measures.

Vulnerabilities/ Attacks Location	Typical Vulnerabilities/ Attacks	Authors of Key Works	Detection Tools/Preventive Techniques
Contract Layer	Re-entrancy	Antonopoulos and Wood (2018) [26] Shahda (2019) [53] Khan and Namin (2020) [58] Alkhalifah et al. (2021) [98] Feng et al. (2019) [99] Fang et al. (2021) [100]	- Limit calls to external contract [26,53] - Use Mutex to lock some function states [26] - Security analysis static tools such as Oyente, Teether, Gasper, Vandal, Securify, smartcheck, Zeus [58,98] - Security analysis dynamic tools such as Vultron, Sereum, Regaurd [58] - Fuzzing tool such as ContractFuzzer [58,99] - Use taint analysis and symbolic execution such ad OSIRIS, EasyFlow, SmartScopy, [58,99]. - Sereum (Secure Ethereum) to perform run-time monitoring of SC execution [58] - Jyane, a dynamic path profiling solution for SC [100]
	Parity multi signature wallet	Chen et al. (2021) [4] Vivar et al. (2020) [5] Praitheeshan et al. (2020) [54] Goldberg (2018) [101] Wang et al. (2020) [102]	- Build stateless libraries [4] - Use static security analysis tools such as Oyente [5] - Adopt the private modifier by default [54] - Avoid using "delegateCall" as a catch-all forwarding mechanism [101] - Use verification tool such as Artemis [102]

Table 2. *Cont.*

Vulnerabilities/ Attacks Location	Typical Vulnerabilities/ Attacks	Authors of Key Works	Detection Tools/Preventive Techniques
Contract Layer	Front running/Transaction ordering dependence	Praitheeshan et al. (2020) [54] Eskandari et al. (2019) [103] Najafi (2020) [104] Mense and Flatscher (2018) [105]	- Enforce rules such as first in first out (FIFO) by adding a complex consensus-based solution [54,104] - Use static security analysis tools such as Oyente, Securify, Mythril [54,105] - Use cryptographic commit-reveal scheme to limit visibility of transaction details [103] - Remove miner's ability to arbitrarily order transaction by forcing queuing/ordering for the transactions [103]
	Integer overflow/Underflow	Praitheeshan et al. (2020) [54] Ma et al. (2019) [55] Gao et al. (2019) [56] Khan and Namin (2020) [58]	- Use static security analysis tools such as Oyente, Zeus [54]. - Create dedicated mathematical libraries and use SafeMath [55] - Check the validity of math output [55] - Use taint analysis and symbolic execution such ad OSIRIS [58], EasyFlow [56] - Use dynamic security analysis tools such as Vultron [58]
	Timestamp dependence	Antonopoulos and Wood (2018) [24] Solorio et al. (2019) [52] Praitheeshan et al. (2020) [54] Jiang et al. (2018) [57] Khan and Namin (2020) [58] Feng et al. (2019) [99]	- Avoid using block.number as a timestamp [24] - Use The 15-second Rule. [52] - Not rely on block.timestamp or blockhash as a source of randomness [52] - Use static security analysis tools such as Oyente, Remix, Mythril, SmartCheck, Zeus [54,58] - Fuzzing tool such as ContractFuzzer [57] - Use SmartScopy as an attack synthesiser [99]
	Mishandled exceptions	Praitheeshan et al. (2020) [54] Khan and Namin (2020) [58]	- Handle the error manually in the caller contract and check the return value of functions [55] - Use static security analysis tools such as Oyente, Remix, Mythril, SmartCheck, Securify, GasFuzzer [55,59]
	DoS with unexpected revert	Samreen and Alalfi (2021) [62]	- Proposed a framework called SmartScan that combines static and dynamic analysis to identify vulnerable pattern and detect [62] - Isolate if/for statements with an external function call [62]
	Short address	Vivar et al. (2020) [5] Wen et al. (2021) [14] Antonopoulos and Wood (2018) [26] Feng et al. (2019) [99] Kushwaha et al. (2022) [106]	- Use static security analysis tools such as SmarCheck [5] - Use SmartScopy as an attack synthesiser [14,99] - Validate input parameters in external applications before sending them [26] - Check parameter ordering [26] - Use dynamic analysis tool Etherolic and SoliAudit [106]

Table 2. *Cont.*

Vulnerabilities/ Attacks Location	Typical Vulnerabilities/ Attacks	Authors of Key Works	Detection Tools/Preventive Techniques
Contract Layer	DoS- Block gas limit	Chen et al. (2021) [4] Ghaleb et al. (2022) [107] Grech et al. (2018) [108]	- Not use loops over data structures [4] - Splitting the loop over multiple transactions to alleviate the risk of an unbounded loop [107] - Implement access control to restrict the call of the public function to only the owner of the contract or specific addresses [107] - Use static analyser technique/tool eTainter [107], MadMax [108]
	Tx.origin	Chen et al. (2021) [4] Antonopoulos and Wood (2018) [26] Tikhomirov et al. (2018) [60]	- Check the authorisation of ownership by using msg.sender' in place of 'tx.origin'. [4,26] - Use static security analysis tools such as SmartCheck [60]
	Weak randomness	Chatterjee et al. (2019) [65] Amiet (2021) [109]	- Designed a well-incentivised and unmanipulable approach which provides a trustworthy source of randomness that is not rely on malicious miners or off-chain oracles [65] - RANDAO, a secure random number generator [109]
	Hash Collisions with Multiple Variable Length Arguments	swcregistry (2020) [66]	- Ensure matching signature cannot be achieved using different parameters [66] - Avoid using abi.encodePacked() and alternatively use abi.encode() instead [66]
	One owner control (Centralised ownership)	CertiK (2023) [110] Mou et al. (2021) [111] Li et al. (2022) [112] Ghaffari et al. (2021) [113] CertiK (2021) [114] Shanzson (2022) [115]	- Manual analysis [110] a. Check contract's ownership b. Correct permission to critical functions. c. Renounce the ownership/never claim the privileged roles. d. Remove the risky functionality. - Set up time-based access control on privilege operations [111,114] - Implement multi signature accounts, use an efficient asymmetric encryption scheme by combining homomorphic encryption and state-of-the-art multi-signature key aggregation and non-interactive zero knowledge proof to preserve privacy and verify valid transactions [112] - Implement access control mechanisms [113] - Use rug checker tools to detect a rug pull [115]

From the analysis of the work in [26,96,97] and the related literature cited in Table 2, a model application is developed depicting, in more detail, the security risks within the contract layer. For each vulnerability, the model proposes a best practice to adopt when writing Solidity Code, best practice to be adopted by developers in general, and suggested analysis tools to use. This model is presented in Figure 5.

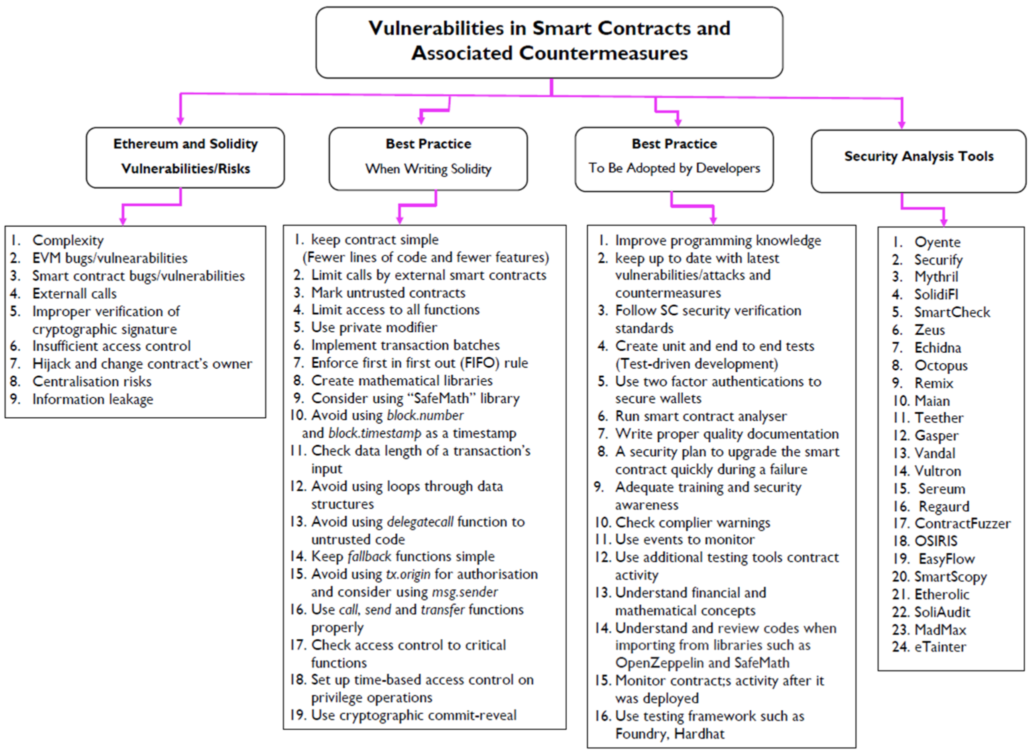
Vulnerabilities in Smart Contracts and Associated Countermeasures
Ethereum and Solidity Vulnerabilities/Risks
Best Practice When Writing Solidity
Best Practice To Be Adopted by Developers
Security Analysis Tools
1. Complexity
2. EVM bugs/vulnearabilities
3. Smart contract bugs/vulnerabilities
4. Externall calls
5. Improper verification of cryptographic signature
6. Insufficient access control
7. Hijack and change contract's owner
8. Centralisation risks
9. Information leakage
1. keep contract simple (Fewer lines of code and fewer features)
2. Limit calls by external smart contracts
3. Mark untrusted contracts
4. Limit access to all functions
5. Use private modifier
6. Implement transaction batches
7. Enforce first in first out (FIFO) rule
8. Create mathematical libraries
9. Consider using "SafeMath" library
10. Avoid using *block.number* and *block.timestamp* as a timestamp
11. Check data length of a transaction's input
12. Avoid using loops through data structures
13. Avoid using *delegatecall* function to untrusted code
14. Keep *fallback* functions simple
15. Avoid using *tx.origin* for authorisation and consider using *msg.sender*
16. Use *call, send* and *transfer* functions properly
17. Check access control to critical functions
18. Set up time-based access control on privilege operations
19. Use cryptographic commit-reveal
1. Improve programming knowledge
2. keep up to date with latest vulnerabilities/attacks and countermeasures
3. Follow SC security verification standards
4. Create unit and end to end tests (Test-driven development)
5. Use two factor authentications to secure wallets
6. Run smart contract analyser
7. Write proper quality documentation
8. A security plan to upgrade the smart contract quickly during a failure
9. Adequate training and security awareness
10. Check complier warnings
11. Use events to monitor
12. Use additional testing tools contract activity
13. Understand financial and mathematical concepts
14. Understand and review codes when importing from libraries such as OpenZeppelin and SafeMath
15. Monitor contract;s activity after it was deployed
16. Use testing framework such as Foundry, Hardhat
1. Oyente
2. Securify
3. Mythril
4. SolidiFI
5. SmartCheck
6. Zeus
7. Echidna
8. Octopus
9. Remix
10. Maian
11. Teether
12. Gasper
13. Vandal
14. Vultron
15. Sereum
16. Regaurd
17. ContractFuzzer
18. OSIRIS
19. EasyFlow
20. SmartScopy
21. Etherolic
22. SoliAudit
23. MadMax
24. eTainter

Figure 5. Model application for best practice towards a more secure smart contract development.

6. Conclusions and Recommendations

6.1. A Seven-Layer Architecture and Best Practices to Mitigate against Security Risks

Blockchain technology, with its decentralised network architecture, can have a significant contribution to sustainability initiatives. This contribution can only be realised if appropriate measures are put in place to counter the security threats to centralisation. This requires a meticulous look into security sources within the blockchain architecture. This paper argues that a seven-layer architecture provides a better framework that enables a more detailed, and thus a more comprehensive, approach for analysing the security risks in the blockchain. On this basis, a seven-layer architecture is adopted providing an in-depth scrutiny of security threats and vulnerabilities associated with each of the seven layers. For each layer, the different vulnerabilities and the type of attacks that can exploit these vulnerabilities are highlighted. A summary of the existing countermeasures is provided. Of the seven layers, particular attention is given to the Contract Layer, and more specifically, the vulnerabilities associated with how smart contracts are written. This particular attention is justified by the fact that smart contracts, being programmable units, are inherently prone to security vulnerabilities. A model application is proposed to enhance the security of smart contracts.

6.2. Key Contributions

The main contributions of this article can be summarised as follows:

- A review of the blockchain architecture is conducted and a more detailed seven-layer architecture is adopted.
- In each of the seven layers of blockchain, the different types of vulnerabilities and attacks are highlighted. The inter-relationships between these vulnerabilities, their exploitation, and the related consequences are described, with particular focus on the case of Ethereum blockchain.
- A systematic investigation is carried out, covering the mechanisms proposed by researchers to detect/prevent vulnerabilities and attacks. The outcome of this investigation is summarised in a taxonomy of vulnerabilities, attacks, and countermeasures in a seven-layer blockchain architecture.

- The contract layer is found to be the most vulnerable layer in a blockchain architecture, due to smart contracts being prone to security vulnerabilities. A model application is proposed to achieve best practice towards a more secure smart contract development.

6.3. Future Work

Although this work generates a new perspective and a platform for a greater understanding of blockchain security risks, it is primarily based on secondary research. More work is needed to ascertain the level of confidence in the proposed countermeasures and best practices, and to identify new techniques and methods to mitigate against security risks, particularly as the threat landscape keeps on changing.

This applies to all the layers of the blockchain. For the contract layer in particular, an area of continuing interest is related to the potential centralisation that can be caused by smart contracts. Smart contracts with centralised ownership pose major security issues and act as a single point of failure, which contradicts the very decentralised nature of blockchain. To mitigate against the risks associated with centralised control, decentralised autonomous organisations (DAOs) promise to alleviate some of these risks by enforcing automated rules that are encoded in smart contracts, thus reinforcing community-based governance. With creating a decentralised decision-making process, the power of decision-making will be distributed and thus preventing smart contract ownership, ensuring that no single individual or team has complete control over the network. A potential focus here is to use an Ethereum blockchain with a DAO structure to develop a method that forces smart contracts to be written in such a way as to prevent one-owner control, thus enabling genuine DAO.

Genuine DAO has the potential to contribute to sustainability initiatives in various ways. DAOs offer transparency, security, and decentralised decision making, which can be advantageous for promoting sustainable practices and addressing global challenges. Additional studies could profitably focus on parallel developments in other smart contact platforms, as well as lessons learnt from cross-platform comparisons and contrasts. The wider implications for industry and organisations considering adopting blockchain also warrant detailed research, including the implications for IT skillsets and cybersecurity policies.

Author Contributions: This research was conducted by S.M. under the supervision of K.B.; conceptualization, S.M. and K.B.; methodology, S.M. and K.B.; investigation, S.M.; data curation, S.M.; writing—original draft preparation, S.M.; writing—review and editing, K.B.; supervision, K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zamani, E.; He, Y.; Phillips, M. On the Security Risks of the Blockchain. *J. Comput. Inf. Syst.* **2020**, *60*, 495–506. [CrossRef]
2. Lin, I.-C.; Liao, T.-C. A Survey of Blockchain Security Issues and Challenges. *Int. J. Netw. Secur.* **2017**, *19*, 653–659. [CrossRef]
3. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A Survey of Distributed Consensus Protocols for Blockchain Networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465. [CrossRef]
4. Chen, H.; Pendleton, M.; Njilla, L.; Xu, S. A Survey on Ethereum Systems Security. *ACM Comput. Surv.* **2021**, *53*, 1–43. [CrossRef]
5. Vivar, A.L.; Castedo, A.T.; Orozco, A.L.S.; Villalba, L.J.G. An Analysis of Smart Contracts Security Threats alongside Existing Solutions. *Entropy* **2020**, *22*, 203. [CrossRef] [PubMed]
6. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An Overview on Smart Contracts: Challenges, Advances and Platforms. *Future Gener. Comput. Syst.* **2020**, *105*, 475–491. [CrossRef]
7. Mosakheil, J.H. Security Threats Classification in Blockchains. 2018. Available online: https://www.semanticscholar.org/paper/Security-Threats-Classification-in-Blockchains-Mosakheil/91bbbb31101cbc2e803726d7210b4100f7b09ac5 (accessed on 20 March 2023).

8. Neumeyer, X.; Cheng, K.; Chen, Y.; Swartz, K. Blockchain and Sustainability: An Overview of Challenges and Main Drivers of Adoption. In Proceedings of the 2021 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD), Marrakech, Morocco, 23–25 November 2022; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6. [CrossRef]
9. Morstyn, T.; Farrell, N.; Darby, S.J.; McCulloch, M.D. Using Peer-to-Peer Energy-Trading Platforms to Incentivize Prosumers to Form Federated Power Plants. *Nat. Energy* **2018**, *3*, 94–101. [CrossRef]
10. Wu, J.; Tran, N. Application of Blockchain Technology in Sustainable Energy Systems: An Overview. *Sustainability* **2018**, *10*, 3067. [CrossRef]
11. Dodmane, R.; K. R., R.; N. S., K.R.; Kallapu, B.; Shetty, S.; Aslam, M.; Jilani, S.F. Blockchain-Based Automated Market Makers for a Decentralized Stock Exchange. *Information* **2023**, *14*, 280. [CrossRef]
12. Sai, A.R.; Buckley, J.; Fitzgerald, B.; Le Gear, A. Taxonomy of Centralization in Public Blockchain Systems: A Systematic Literature Review. *Inf. Process. Manag.* **2021**, *58*, 102584. [CrossRef]
13. Marcus, Y.; Heilman, E.; Goldberg, S. Low-Resource Eclipse Attacks on Ethereum's Peer-to-Peer Network. *Cryptol. ePrint Arch.* **2018**.
14. Wen, Y.; Lu, F.; Liu, Y.; Huang, X. Attacks and Countermeasures on Blockchains: A Survey from Layering Perspective. *Comput. Netw.* **2021**, *191*, 107978. [CrossRef]
15. Tapsell, J.; Naeem Akram, R.; Markantonakis, K. An Evaluation of the Security of the Bitcoin Peer-To-Peer Network. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1057–1062. [CrossRef]
16. CertiK. What Is Centralization Risk? *2022.* Available online: https://certik.medium.com/what-is-centralization-risk-41cf848f5a74 (accessed on 25 March 2023).
17. Saunders, M.N.K.; Lewis, P.; Thornhill, A. *Research Methods for Business Students*, 8th ed.; Pearson: New York, NY, USA, 2020.
18. Yang, J.; Bi, H.; Liang, Z.; Zhou, H.; Yang, H.J. A Survey on Blockchain: Architecture, Applications, Challenges, and Future Trends. In Proceedings of the IEEE Congress on Cybermatics: 2020 IEEE International Conferences on Internet of Things, iThings 2020, IEEE Green Computing and Communications, GreenCom 2020, IEEE Cyber, Physical and Social Computing, CPSCom 2020 and IEEE Smart Data, SmartD, Rhodes, Greece, 2–6 November 2020. [CrossRef]
19. Deng, W.; Huang, T.; Wang, H. A Review of the Key Technology in a Blockchain Building Decentralized Trust Platform. *Mathematics* **2022**, *11*, 101. [CrossRef]
20. Homoliak, I.; Venugopalan, S.; Reijsbergen, D.; Hum, Q.; Schumi, R.; Szalachowski, P. The Security Reference Architecture for Blockchains: Toward a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 341–390. [CrossRef]
21. Huang, J.; Lei, K.; Du, M.; Zhao, H.; Liu, H.; Liu, J.; Qi, Z. Survey on Blockchain Incentive Mechanism. In Proceedings of the ICPCSEE 2019 International Conference of Pioneering Computer Scientists, Engineers and Educators, Guilin, China, 20–23 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 386–395. [CrossRef]
22. Ahmed, K.B.; Kumar, D. Blockchain Use Cases in Financial Services for Improving Security. In Proceedings of the 2019 Third International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 10–11 January 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 220–224. [CrossRef]
23. Annessi, R.; Fast, E. Improving Security for Users of Decentralized Exchanges Through Multiparty Computation. In Proceedings of the 2021 IEEE International Conference on Blockchain (Blockchain), Melbourne, Australia, 6–8 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 229–236. [CrossRef]
24. Sexer, N. Decentralized Exchanges vs. Centralized Exchanges: Overview. 2018. Available online: https://consensys.net/blog/news/decentralized-exchanges-overview-benefits-and-advantages-over-centralized-exchanges/ (accessed on 1 April 2023).
25. Jha, P. Ethereum at the Center of Centralization Debate as SEC Lays Claim. 2022. Available online: https://cointelegraph.com/news/ethereum-at-the-center-of-centralization-debate-as-sec-lays-claim (accessed on 2 April 2023).
26. Antonopoulos, A.; Wood, G. *Mastering Ethereum: Building Smart Contracts and Dapps*; O'Reilly Media: Sebastopol, CA, USA, 2018.
27. Destefanis, G.; Marchesi, M.; Ortu, M.; Tonelli, R.; Bracciali, A.; Hierons, R. Smart Contracts Vulnerabilities: A Call for Blockchain Software Engineering? In Proceedings of the 2018 International Workshop on Blockchain Oriented Software Engineering (IWBOSE), Campobasso, Italy, 20 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 19–25. [CrossRef]
28. DevCon, G. What are Blockchain Protocols and How Do they Work? Available online: https://medium.com/@genesishack/draft-what-are-blockchain-protocols-and-how-do-they-work-94815be5efa7 (accessed on 5 April 2023).
29. Han, R.; Yan, Z.; Liang, X.; Yang, L.T. How Can Incentive Mechanisms and Blockchain Benefit with Each Other? A Survey. *ACM Comput. Surv.* **2023**, *55*, 1–38. [CrossRef]
30. Leonardos, N.; Leonardos, S.; Piliouras, G. Oceanic Games: Centralization Risks and Incentives in Blockchain Mining. In Proceedings of the Mathematical Research for Blockchain Economy: 1st International Conference MARBLE 2019, Santorini, Greece, 6–9 May 2019. [CrossRef]
31. Beikverdi, A.; Song, J.S. Trend of Centralization in Bitcoin's Distributed Network. In Proceedings of the 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Takamatsu, Japan, 1–3 June 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6. [CrossRef]
32. Minima. Is Bitcoin Incentivizing Its Own Centralization? 2022. Available online: https://www.minima.global/post/is-bitcoin-incentivizing-its-own-centralization (accessed on 10 January 2023).

33. Liott, S. Has Proof of Stake Made Ethereum More Centralized? 2022. Available online: https://decrypt.co/111485/has-proof-of-stake-made-ethereum-more-centralized (accessed on 13 March 2023).
34. Ethereum. Proof-of-Stake (PoS). 2023. Available online: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/ (accessed on 12 May 2023).
35. Alsunaidi, S.J.; Alhaidari, F.A. A Survey of Consensus Algorithms for Blockchain Technology. In Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 3–4 April 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [CrossRef]
36. Suresh, A.; Nair, A.R.; Lal, A.; Kumaran, S.M.; Sarath, G. A Hybrid Proof Based Consensus Algorithm for Permission Less Blockchain. In Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 15–17 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 707–713. [CrossRef]
37. Nguyen, G.-T.; Kim, K. A Survey about Consensus Algorithms Used in Blockchain. *J. Inf. Process. Syst.* **2018**, *14*, 101–128. [CrossRef]
38. Huang, J.; Tan, L.; Mao, S.; Yu, K. Blockchain Network Propagation Mechanism Based on P4P Architecture. *Secur. Commun. Netw.* **2021**, *2021*, 8363131. [CrossRef]
39. Essaid, M.; Kim, H.W.; Guil Park, W.; Lee, K.Y.; Jin Park, S.; Ju, H.T. Network Usage of Bitcoin Full Node. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 17–19 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1286–1291. [CrossRef]
40. Xu, Y. Section-Blockchain: A Storage Reduced Blockchain Protocol, the Foundation of an Autotrophic Decentralized Storage Architecture. In Proceedings of the 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS), Melbourne, VIC, Australia, 12–14 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 115–125. [CrossRef]
41. Liu, Y.; Zhang, Y.; Zhu, S.; Chi, C. A Comparative Study of Blockchain-Based DNS Design. In Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications, Xi'an, China, 9–11 December 2019; ACM: New York, NY, USA, 2019; pp. 86–92. [CrossRef]
42. Liang, Y.-C. Blockchain for Dynamic Spectrum Management. In *Dynamic Spectrum Management, Proceeding of the Cognitive Radio to Blockchain and Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 121–146. [CrossRef]
43. Choo, K.-K.R.; Dehghantanha, A.; Parizi, R.M. (Eds.) Blockchain Cybersecurity, Trust and Privacy. In *Advances in Information Security*; Springer International Publishing: Cham, Switzerland, 2020; Volume 79. [CrossRef]
44. Edgcombe, J. So, You Want to Connect Your IoT Device to the Blockchain? 2016. Available online: https://www.cambridgeconsultants.com/insights/so-you-want-to-connect-your-iot-device-to-the-b)lockchain (accessed on 10 November 2022).
45. Rezaeighaleh, H.; Zou, C.C. New Secure Approach to Backup Cryptocurrency Wallets. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [CrossRef]
46. Sung, S. A New Key Protocol Design for Cryptocurrency Wallet. *ICT Express* **2021**, *7*, 316–321. [CrossRef]
47. Partz, H. Bilaxy Exchange Suspends Website after ERC-20 Hot Wallet Hack. 2021. Available online: https://cointelegraph.com/news/bilaxy-exchange-suspends-website-after-erc-20-hot-wallet-hack (accessed on 12 December 2022).
48. Thomas, D. AscendEX Hacked, $77.7M Lost From Hot Wallets. 2021. Available online: https://beincrypto.com/ascendex-hacked-77-7m-lost-from-hot-wallets/ (accessed on 5 June 2022).
49. Werapun, W.; Karode, T.; Arpornthip, T.; Suaboot, J.; Sangiamkul, E.; Boonrat, P. The Flash Loan Attack Analysis (FAA) Framework—A Case Study of the Warp Finance Exploitation. *Informatics* **2022**, *10*, 3. [CrossRef]
50. Qin, K.; Zhou, L.; Livshits, B.; Gervais, A. Attacking the DeFi Ecosystem with Flash Loans for Fun and Profit. In Proceedings of the International Conference on Financial Cryptography and Data Security, Virtual Event, 1–5 March 2021; pp. 3–32. [CrossRef]
51. Thurman, A. Cream Finance Exploited in Flash Loan Attack Netting Over $100M. 2021. Available online: https://www.coindesk.com/business/2021/10/27/cream-finance-exploited-in-flash-loan-attack-worth-over-100m (accessed on 6 June 2022).
52. Solorio, K.; Hooper, D.; Kanna, R. *Hands-On Smart Contract Development with Solidity and Ethereum: From Fundamentals to Deployment Paperback*; O'Reilly Media: Sebastopol, CA, USA, 2019.
53. Shahda, W. Protect Your Solidity Smart Contracts from Re-entrancy Attacks. 2019. Available online: https://medium.com/coinmonks/protect-your-solidity-smart-contracts-from-reentrancy-attacks-9972c3af7c21 (accessed on 4 September 2022).
54. Praitheeshan, P.; Pan, L.; Yu, J.; Liu, J.; Doss, R. Security Analysis Methods on Ethereum Smart Contract Vulnerabilities: A Survey. *arXiv* **2019**, arXiv:1908.08605.
55. Ma, R.; Gorzny, J.; Zulkoski, E.; Bak, K.; Mack, O.V. *Fundamentals of Smart Contract Security, Kindle ed.*; Momentum Press: New York, NY, USA, 2019.
56. Gao, J.; Liu, H.; Liu, C.; Li, Q.; Guan, Z.; Chen, Z. EASYFLOW: Keep Ethereum Away from Overflow. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), Montreal, QC, Canada, 25–31 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 23–26. [CrossRef]
57. Jiang, B.; Liu, Y.; Chan, W.K. ContractFuzzer: Fuzzing Smart Contracts for Vulnerability Detection. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, Montpellier, France, 3–7 September 2018; ACM: New York, NY, USA, 2018; pp. 259–269. [CrossRef]
58. Khan, Z.A.; Siami Namin, A. Ethereum Smart Contracts: Vulnerabilities and their Classifications. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020. [CrossRef]

59. Huang, Y.; Bian, Y.; Li, R.; Zhao, J.L.; Shi, P. Smart Contract Security: A Software Lifecycle Perspective. *IEEE Access* **2019**, *7*, 150184–150202. [CrossRef]
60. Tikhomirov, S.; Voskresenskaya, E.; Ivanitskiy, I.; Takhaviev, R.; Marchenko, E.; Alexandrov, Y. SmartCheck. In *Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*; ACM: New York, NY, USA, 2018; pp. 9–16. [CrossRef]
61. Sayeed, S.; Marco-Gisbert, H.; Caira, T. Smart Contract: Attacks and Protections. *IEEE Access* **2020**, *8*, 24416–24427. [CrossRef]
62. Samreen, N.F.; Alalfi, M.H. SmartScan: An Approach to Detect Denial of Service Vulnerability in Ethereum Smart Contracts. In Proceedings of the 2021 IEEE/ACM 4th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), Madrid, Spain, 31 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 17–26. [CrossRef]
63. Bouichou, A.; Mezroui, S.; El Oualkadi, A. An Overview of Ethereum and Solidity Vulnerabilities. In Proceedings of the 2020 International Symposium on Advanced Electrical and Communication Technologies (ISAECT), Marrakech, Morocco, 25–27 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7. [CrossRef]
64. Swcregistry. Weak Sources of Randomness from Chain Attributes. 2020. Available online: https://swcregistry.io/docs/SWC-120 (accessed on 10 October 2022).
65. Chatterjee, K.; Goharshady, A.K.; Pourdamghani, A. Probabilistic Smart Contracts: Secure Randomness on the Blockchain. In Proceedings of the 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Seoul, Republic of Korea, 14–17 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 403–412. [CrossRef]
66. swcregistry. Hash Collisions with Multiple Variable Length Arguments. 2020. Available online: https://swcregistry.io/docs/SWC-133 (accessed on 17 October 2022).
67. Chittoda, J. *Mastering Blockchain Programming with Solidity: Write Production-Ready Smart Contracts for Ethereum Blockchain with Solidity*; Packt Publishing: Birmingham, UK, 2019.
68. Solidity Programming Language. Contract ABI Specification. 2021. Available online: https://docs.soliditylang.org/en/v0.8.11/abi-spec.html (accessed on 20 February 2023).
69. Zipfel, K. New Smart Contract Weakness: Hash Collisions with Multiple Variable Length Arguments. 2020. Available online: https://medium.com/swlh/new-smart-contract-weakness-hash-collisions-with-multiple-variable-length-arguments-dc7b9c84e493 (accessed on 2 November 2022).
70. Ghaleb, A.; Rubin, J.; Pattabiraman, K. AChecker: Statically Detecting Smart Contract Access Control Vulnerabilities. In Proceedings of the 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 14–20 May 2023.
71. Dai, W.; Wang, C.; Cui, C.; Jin, H.; Lv, X. Blockchain-Based Smart Contract Access Control System. In Proceedings of the 2019 25th Asia-Pacific Conference on Communications (APCC), Ho Chi Minh City, Vietnam, 6–8 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 19–23. [CrossRef]
72. OpenZeppelin. Access Control. 2022. Available online: https://docs.openzeppelin.com/contracts/4.x/access-control (accessed on 7 March 2023).
73. Code4rena. Frax Ether Liquid Staking Contest Findings & Analysis Report—Centra. 2022. Available online: https://code4rena.com/reports/2022-09-frax/#m-01-centralization-risk-admin-have-privileges-admin-can-set-address-to-mint-any-amount-of-frxeth-can-set-any-address-as-validator-and-change-important-state-in-frxethminter-and-withdraw-fund-from-frcethminter- (accessed on 9 May 2023).
74. Mirkin, M.; Ji, Y.; Pang, J.; Klages-Mundt, A.; Eyal, I.; Juels, A. BDoS: Blockchain Denial-of-Service. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, 9–13 November 2020; ACM: New York, NY, USA, 2020; pp. 601–619. [CrossRef]
75. Kitakami, M.; Matsuoka, K. An Attack-Tolerant Agreement Algorithm for Block Chain. In Proceedings of the 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC), Taipei, Taiwan, 4–7 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 227–228. [CrossRef]
76. Saad, M.; Njilla, L.; Kamhoua, C.; Mohaisen, A. Countering Selfish Mining in Blockchains. In Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC), Honolulu, HI, USA, 18–21 February 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 360–364. [CrossRef]
77. Sun, H.; Ruan, N.; Su, C. How to Model the Bribery Attack: A Practical Quantification Method in Blockchain. In Proceedings of the 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, 14–18 September 2020; pp. 569–589. [CrossRef]
78. Bonneau, J. Why Buy When You Can Rent? Bribery Attacks on Bitcoin-Style Consensus. In Proceedings of the International Conference on Financial Cryptography and Data Security, Christ Church, Barbados, 26 February 2016; pp. 19–26.
79. Liao, K.; Katz, J. Incentivizing Double-Spend Collusion in Bitcoin. In Proceedings of the Financial Cryptography Bitcoin Workshop, Sliema, Malta, 7 April 2017.
80. Saad, M.; Thai, M.T.; Mohaisen, A. POSTER. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Republic of Korea, 4–8 June 2018; ACM: New York, NY, USA, 2018; pp. 809–811. [CrossRef]
81. Li, Z.; Gao, S.; Peng, Z.; Guo, S.; Yang, Y.; Xiao, B. B-DNS: A Secure and Efficient DNS Based on the Blockchain Technology. *IEEE Trans. Netw. Sci. Eng.* **2021**, *8*, 1674–1686. [CrossRef]

82. Ren, S.; Liu, B.; Yang, F.; Wei, X.; Yang, X.; Wang, C. BlockDNS: Enhancing Domain Name Ownership and Data Authenticity with Blockchain. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [CrossRef]
83. Swathi, P.; Modi, C.; Patel, D. Preventing Sybil Attack in Blockchain Using Distributed Behavior Monitoring of Miners. In Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 6–8 July 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [CrossRef]
84. Saad, M.; Anwar, A.; Ahmad, A.; Alasmary, H.; Yuksel, M.; Mohaisen, D. RouteChain: Towards Blockchain-Based Secure and Efficient BGP Routing. *Comput. Netw.* **2022**, *217*, 109362. [CrossRef]
85. Hu, B.; Zhou, C.; Tian, Y.-C.; Qin, Y.; Junping, X. A Collaborative Intrusion Detection Approach Using Blockchain for Multimicrogrid Systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *49*, 1720–1730. [CrossRef]
86. Sward, A.; Vecna, I.; Stonedahl, F. Data Insertion in Bitcoin's Blockchain. *Ledger* **2018**, 3. [CrossRef]
87. Khan, K.M.; Arshad, J.; Khan, M.M. Simulation of Transaction Malleability Attack for Blockchain-Based e-Voting. *Comput. Electr. Eng.* **2020**, *83*, 106583. [CrossRef]
88. Sigurdsson, G.; Giaretta, A.; Dragoni, N. Vulnerabilities and Security Breaches in Cryptocurrencies. In Proceedings of the 6th International Conference in Software Engineering for Defence Applications: SEDA 2018, Rome, Italy, 7–8 June 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 288–299. [CrossRef]
89. Kearney, J.J.; Perez-Delgado, C.A. Vulnerability of Blockchain Technologies to Quantum Attacks. *Array* **2021**, *10*, 100065. [CrossRef]
90. Khalifa, A.M.; Bahaa-Eldin, A.M.; Sobh, M.A. Quantum Attacks and Defenses for Proof-of-Stake. In Proceedings of the ICCES 2019: 2019 14th International Conference on Computer Engineering and Systems, Cairo, Egypt, 17 December 2019. [CrossRef]
91. Conti, M.; Sandeep Kumar, E.; Lal, C.; Ruj, S. A Survey on Security and Privacy Issues of Bitcoin. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 3416–3452. [CrossRef]
92. Hu, Y.; Wang, S.; Tu, G.-H.; Xiao, L.; Xie, T.; Lei, X.; Li, C.-Y. Security Threats from Bitcoin Wallet Smartphone Applications. In Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy, Virtual Event, USA, 26–28 April 2021; ACM: New York, NY, USA, 2021; pp. 89–100. [CrossRef]
93. Tanana, D. Behavior-Based Detection of Cryptojacking Malware. In Proceedings of the 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, 14–15 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 0543–0545. [CrossRef]
94. Sm4rty. Smart Contract Audit Methodology & Tips. 2022. Available online: https://sm4rty.medium.com/smart-contract-audit-methodology-tips-6e529a3f3435 (accessed on 15 May 2023).
95. Ajienka, N.; Vangorp, P.; Capiluppi, A. An Empirical Analysis of Source Code Metrics and Smart Contract Resource Consumption. *J. Softw. EVolume Process* **2020**, *32*, e2267. [CrossRef]
96. SWC. Smart Contract Weakness Classification and Test Cases. 2020. Available online: https://swcregistry.io/ (accessed on 2 June 2023).
97. ConsenSys. Ethereum Smart Contract Best Practices—Known Attacks. Available online: https://consensys.github.io/smart-contract-best-practices/ (accessed on 7 June 2023).
98. Alkhalifah, A.; Ng, A.; Watters, P.A.; Kayes, A.S.M. A Mechanism to Detect and Prevent Ethereum Blockchain Smart Contract Reentrancy Attacks. *Front. Comput. Sci.* **2021**, *3*, 598780. [CrossRef]
99. Feng, Y.; Torlak, E.; Bodik, R. Precise Attack Synthesis for Smart Contracts. *arXiv* **2019**, arXiv:1902.06067.
100. Fang, Y.; Wang, C.; Sun, Z.; Cheng, H. Jyane: Detecting Reentrancy Vulnerabilities Based on Path Profiling Method. In Proceedings of the 2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS), Beijing, China, 14–16 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 274–282. [CrossRef]
101. Goldberg, O. How to Not Destroy Millions in Smart Contracts. 2018. Available online: https://hackernoon.com/how-to-not-destroy-millions-in-smart-contracts-pt-2-85c4d8edd0cf (accessed on 15 June 2023).
102. Wang, A.; Wang, H.; Jiang, B.; Chan, W.K. Artemis: An Improved Smart Contract Verification Tool for Vulnerability Detection. In Proceedings of the 2020 7th International Conference on Dependable Systems and Their Applications (DSA), Xi'an, China, 28–29 November 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 173–181. [CrossRef]
103. Eskandari, S.; Moosavi, S.; Clark, J. SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain. In Proceedings of the Financial Cryptography and Data Security: FC 2019 International Workshops, VOTING and WTSC, St. Kitts, St. Kitts and Nevis, 18–22 February 2019.
104. Najafi, S. Front-Running Attacks on Blockchain. 2020. Available online: https://medium.com/codechain/front-running-attacks-on-blockchain-1f5ba28cd42b (accessed on 14 April 2023).
105. Mense, A.; Flatscher, M. Security Vulnerabilities in Ethereum Smart Contracts. In Proceedings of the 20th International Conference on Information Integration and Web-Based Applications & Services, Yogyakarta, Indonesia, 19–21 November 2018; ACM: New York, NY, USA, 2018; pp. 375–380. [CrossRef]
106. Kushwaha, S.S.; Joshi, S.; Singh, D.; Kaur, M.; Lee, H.-N. Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract. *IEEE Access* **2022**, *10*, 6605–6621. [CrossRef]
107. Ghaleb, A.; Rubin, J.; Pattabiraman, K. ETainter: Detecting Gas-Related Vulnerabilities in Smart Contracts. In Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, Virtual Event, Republic of Korea, 18–22 July 2022; ACM: New York, NY, USA, 2022; pp. 728–739. [CrossRef]

108. Grech, N.; Kong, M.; Jurisevic, A.; Brent, L.; Scholz, B.; Smaragdakis, Y. MadMax: Surviving out-of-Gas Conditions in Ethereum Smart Contracts. *Proc. ACM Program. Lang.* **2018**, *2*, 1–27. [CrossRef]
109. Amiet, N. Blockchain Vulnerabilities in Practice. *Digit. Threat. Res. Pract.* **2021**, *2*, 1–7. [CrossRef]
110. CertiK. Better Security for Blockchains and Smart Contracts. 2023. Available online: https://www.certik.com/products/formal-verification (accessed on 27 June 2023).
111. Mou, T.; Coblenz, M.; Aldrich, J. An Empirical Study of Protocols in Smart Contracts. *arXiv* **2021**, arXiv:2110.08983. [CrossRef]
112. Li, X.; Ma, Z.; Luo, S. Blockchain-Oriented Privacy Protection with Online and Offline Verification in Cross-Chain System. In Proceedings of the 2022 International Conference on Blockchain Technology and Information Security (ICBCTIS), Huaihua City, China, 15–17 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 177–181. [CrossRef]
113. Ghaffari, F.; Bertin, E.; Crespi, N.; Behrad, S.; Hatin, J. A Novel Access Control Method Via Smart Contracts for Internet-Based Service Provisioning. *IEEE Access* **2021**, *9*, 81253–81273. [CrossRef]
114. CertiK. What Is a Timelock? 2021. Available online: https://www.certik.com/resources/blog/Timelock (accessed on 27 June 2023).
115. Shanzson. Smart Contract Auditor Tools and Techniques. 2022. Available online: https://github.com/shanzson/Smart-Contract-Auditor-Tools-and-Techniques (accessed on 28 June 2023).