International Journal of Information Security

In the Webs of Ethereum: Analyzing Smart Contracts Vulnerabilities

Vaios Bolgouras[1] · Vasilis Magkoutis [1] · Apostolis Zarras[1,2] · Aristeidis Farao[3] · Christos Xenakis[1,3]

Received: date / Accepted: date

Abstract Ensuring the security of smart contracts is essential for maintaining the reliability and trustworthiness of decentralized applications, which are deployed across various domains, including industrial applications. In pursuit of this goal, it is imperative to analyze the common errors developers make when crafting smart contracts on the infrastructure that gave birth to them, i.e., the Ethereum blockchain. In this paper, we present a comprehensive analysis of the vulnerabilities in Ethereum smart contracts. Our methodology involves downloading the entire Ethereum blockchain and identifying smart contracts, which we then scan for vulnerabilities using various tools. We have discovered numerous vulnerabilities across many deployed smart contracts, highlighting the need for improved development practices. This analysis provides critical insights into the prevalence of security issues and underscores the urgency of raising development standards. By promoting the adoption of secure-by-design principles, our research seeks to enhance security standards within the Ethereum smart contract ecosystem.

Keywords Ethereum · Smart Contracts · Vulnerabilities · Cybersecurity

1 Introduction

Blockchain technology has garnered significant interest across various industries and applications, particularly with the emergence of smart contracts—self-executing agreements with terms directly written into code, enabling the automation of transactions and reducing the need for intermediaries. One of the leading platforms in this domain is Ethereum, which has been pivotal in advancing the use of smart contracts. However, as Ethereum-based solutions gain popularity, concerns about their security and reliability arise. This concern is particularly pertinent for decentralized systems, which demand high levels of privacy, security, and trust. Yet, developing decentralized systems based on smart contracts and blockchain technology presents numerous challenges and existing vulnerabilities that demand attention. Over time, numerous vulnerabilities have been discovered in Ethereum-based systems [5], resulting in significant security breaches, financial losses, and compromises of sensitive information [4, 6]. Addressing these challenges and vulnerabilities will facilitate decentralized systems' broader adoption.

One of the principal vulnerabilities encountered in Ethereum-based solutions is the inadequate input validation and sanitization in smart contracts. This deficiency can result in code injection attacks, wherein attackers can manipulate the contract's behavior to steal or tamper with data. For instance, in 2016, an attacker exploited a vulnerability in the renowned smart contract *"The DAO"*, absconding with approximately 3.6 million ETH (equivalent to roughly $70 million at the time) [1]. Another vulnerability observed in Ethereum-based solutions is the lack of proper access control and authorization in smart contracts. This flaw can lead to unauthorized access to sensitive data or unintended execution of actions. For instance, in 2018, a vulnerability in a smart contract associated with the Parity wallet service resulted in the loss of over 500,000 ETH (approximately $300 million at the time) [18]. Additionally, Ethereum-based solutions are susceptible

✉ Vaios Bolgouras <vaiosbolgouras@ssl-unipi.gr>
[1] Department of Digital Systems, University of Piraeus, Piraeus, Greece
[2] Foundation for Research and Technology – Hellas, Greece
[3] InQbit Innovations SRL., Romania

to Denial-of-Service (DoS) attacks, wherein adversaries can overwhelm the system, impeding its proper functioning. Such attacks can cause significant disruptions and financial losses, particularly in decentralized exchanges and other financial applications. This occurred in 2018 when a series of DoS attacks on Ethereum-based decentralized exchanges led to substantial disruptions and trading delays [2]. Numerous tools have been developed since then to detect vulnerabilities in smart contracts [7, 12, 14].

All those incidents highlight the paramount importance of identifying and mitigating developer errors when creating smart contracts. Such errors often lead to vulnerabilities that compromise the security and reliability of blockchain technologies. In this paper, through a comprehensive analysis of these vulnerabilities, particularly within the Ethereum blockchain, we aim to highlight common mistakes and guide developers towards more secure coding practices. The novelty of our approach lies in its thorough evaluation of all smart contracts within the Ethereum network, ranging from 2016 to 2022, providing a unique dataset for analysis. Leveraging multiple open-source scanning tools such as Oyente [8], Osiris [17], Maian [10], and Mythril [9], our research systematically categorizes prevalent security risks. This exhaustive scanning and analysis process enables us to pinpoint specific weaknesses recurring across different smart contracts, offering developers a roadmap to avoid such pitfalls in future implementations. Moreover, this study underscores the critical role of smart contracts in unlocking blockchain's transformative potential across various industries. By rigorously assessing these contracts' vulnerabilities, we lay the groundwork for enhancing their design and execution, fortifying the overall blockchain ecosystem and fostering greater trust and adoption among users and stakeholders.

Our findings serve as a beacon for the ongoing development of smart contracts, highlighting the need for enhanced security protocols and development practices. It is a call to action for the Ethereum community and beyond to prioritize the development of *secure by design* smart contracts. This steadfast commitment to security promises to bolster the reliability of smart contracts and enhance their applicability across diverse industries and use cases. As such, we substantially contribute to smart contract security by furnishing actionable insights and guidelines to mitigate prevalent vulnerabilities, pursuing this way more secure, reliable, and efficient blockchain-based solutions and laying the groundwork for the future advancement and adoption of smart contract technology.

In summary, we make the following main contributions:

- We provide a detailed examination of vulnerabilities in Ethereum smart contracts by analyzing the entire Ethereum blockchain from 2016 to August 2024. Using static and dynamic analysis tools, we categorize common vulnerabilities and identify specific weaknesses that recur across different smart contracts.
- We highlight the trends in smart contract vulnerabilities over time, showing how certain vulnerabilities have become more or less prevalent. We debate the implications of these trends for developers, including the need for ongoing education and adaptation of security best practices in smart contract development.
- We discuss how security vulnerabilities in smart contracts can affect the broader adoption of blockchain. We argue that high-profile security breaches and the resulting financial losses not only undermine trust in blockchain platforms but also pose significant challenges to the scalability and regulatory acceptance of these technologies.

The remainder of this article is organized as follows. Section 2 provides essential background on Ethereum smart contracts, including their operational mechanisms and the static/dynamic analysis tools employed for vulnerability detection. Section 3 outlines the threat model, categorizing adversaries and their potential impacts on confidentiality, integrity, and availability. Section 4 details the methodology, encompassing blockchain data acquisition, smart contract processing, and the integration of vulnerability assessment tools. Section 5 evaluates the results, presenting trends in verified/unverified contracts, vulnerability prevalence over time, and their severity based on CVSS scoring. Section 6 discusses the broader implications of the findings, financial risks, limitations of the study, and future research directions. Section 7 reviews related work, contrasting existing studies with the comprehensive scope of this analysis. Finally, Section 8 concludes the paper by synthesizing key insights and advocating for enhanced security practices to fortify the Ethereum ecosystem.

2 Background

In blockchain, smart contracts function as autonomous agents, implementing pre-defined rules when specific conditions are met. The appeal of these self-executing contracts on the Ethereum blockchain is their potential for increased efficiency, transparency, and immutable enforcement of contracts. Nevertheless, these innovations have vulnerabilities that require thorough investigation to protect the integrity of these digital contracts. Addressing the complexities and hidden vulnerabilities

of Ethereum smart contracts necessitates the combined use of static and dynamic analysis tools.

Static analysis, the primary defensive strategy, evaluates the source code without running it, identifying potential security risks inherent to the static state of the code, such as input validation errors or other misuses that could be exploited. Tools like Oyente [8] and Maian [10] provide critical insights. Oyente employs symbolic execution to analyze contract logic and identify vulnerabilities accurately, whereas Maian uses a pattern-based approach to expose flaws, thus complementing static code analysis. However, static analysis does not encompass all potential vulnerabilities. Dynamic analysis fills this gap by executing the smart contracts in a controlled environment, allowing for the detection of runtime errors and vulnerabilities that only become apparent when the contract is in operation. This method is essential for detecting vulnerabilities overlooked by static analysis. Leading tools in dynamic analysis include Osiris [17] and Mythril [9]. Osiris uses taint analysis to trace contract data flow, pinpointing vulnerabilities that could cause unintended behaviors or exploits. Mythril combines symbolic with concolic execution—a synthesis of symbolic analysis' depth and concrete execution's realism—to detect vulnerabilities that could undermine contract integrity.

Utilizing both types provides a comprehensive evaluation of smart contracts, facilitating the detection of vulnerabilities during the coding and runtime phases. This extensive analysis is crucial for ensuring the security and reliability of smart contracts on the Ethereum blockchain. Integrating static and dynamic analyses fosters a layered security approach, vital in the complex field of smart contracts where failures can lead to substantial financial and reputational losses.

Further, smart contracts are classified as verified or unverified within the Ethereum blockchain ecosystem. Developers often publish their source code to make the smart contract transparent and verifiable. This is usually done on Etherscan, a popular Ethereum blockchain explorer. Etherscan provides a feature for verifying smart contracts, where the developer uploads the contract's source code and specifies the compiler version and settings used. Etherscan then recompiles the source code and compares the resulting bytecode to the bytecode already on the blockchain. If the bytecode from the new compilation matches the bytecode on the blockchain, the smart contract is marked as verified. This indicates that the source code accurately represents the deployed contract. This verified status builds trust among stakeholders, reinforcing confidence in the absence of critical flaws or deficiencies. Conversely, unverified smart contracts are those deployed to the blockchain without providing their source code, potentially due to time constraints, budget limitations, or unawareness of the importance of this process. These contracts may contain hidden vulnerabilities, coding errors, or unintended functionalities. Depending solely on the assurances of the contract's author or developer, without external validation, stakeholders face increased risks.

3 Threat Model

Our model addresses the complex threats to smart contracts by considering adversaries equipped with capabilities like code injection, reentrancy attacks, and exploiting integer overflow/underflow vulnerabilities. These adversaries aim to alter or disrupt smart contract functions for personal gain or to cause damage. Our threat model classifies these threats based on their impact on confidentiality, integrity, and availability (i.e., CIA triad) and considers both external attackers and malicious insiders. External attackers might exploit public vulnerabilities without direct blockchain access, while malicious insiders could leverage their privileged access. We also evaluate the influence of smart contract dependencies, where vulnerabilities in one contract can jeopardize others, particularly in intricate DeFi ecosystems. To address these threats methodically, our analysis examines the bytecode of smart contracts to identify vulnerabilities that adversaries could exploit. By applying static and dynamic analysis tools, we establish a robust framework for detecting and mitigating potential security risks, thereby enhancing the security and resilience of the Ethereum blockchain ecosystem. This threat model lays the groundwork for our subsequent analyses, guiding our choice of tools and methodologies to pinpoint vulnerabilities that represent the most substantial risks. It underscores the imperative for continual vigilance and adaptable security strategies to combat the evolving threats in the blockchain environment.

4 Methodology

In this study, we conduct a thorough analysis of Ethereum smart contracts. Our methodology involves downloading the entire Ethereum blockchain, which requires fetching and storing the data from each block and its associated transactions. The initial phase of our study consists of systematically collecting blockchain blocks, which are subsequently stored locally to enable offline analysis. We achieve this by employing a full node on the Ethereum network to systematically query and download the blocks.

```
{
  "blockHash": "0xaebf4...124b8",
  "blockNumber": 1000050,
  "from": "0x20064...b0894",
  "gas": 1000000,
  "gasPrice": "50000000000",
  "hash": "0xa8b37...3a313",
  "input": "0x60606...04052",
  "nonce": 4,
  "to": null,
  "transactionIndex": 0,
  "value": "0",
  "type": 0,
  "v": "0x1b",
  "r": "0x98733...b50d4",
  "s": "0x3af45...2277c"
}
```


Listing 1: Example of contract-creation transaction

4.1 Acquiring the Smart Contracts

Due to the vast size of the blockchain, which exceeds a Terabyte, we apply compression techniques to reduce the storage requirement to approximately 360 Gigabytes and store the compressed data in JSON format. After securely storing the blockchain data, we focus on identifying potential contract-creation transactions. This process involves scrutinizing each transaction for indicators of contract creation, notably those targeting a non-existent ("null") address. We rigorously validate these transactions through their corresponding receipts to confirm the establishment of a new contract and its unique address. We store these validated contract transactions uncompressed in our local database to enhance data retrieval and processing efficiency.

In the JSON structure depicted in Listing 1, we detail a transaction that initiates a contract creation. The "input" field is particularly noteworthy because it contains the bytecode of the instantiated contract. The "to" field is also crucial; in this case, we set it to "null" to denote the inception of a new contract entity.

4.2 Smart Contracts Assessment

Initially, the Ethereum Network provides blocks and transaction receipts that help identify new contracts. These contracts are handled by the *Main Handler Process*, a central script that orchestrates data collection and processing. The process utilizes both the Etherscan API and blockchain transaction data to retrieve verified contracts and their Application Binary Interface (ABI) data, as well as to collect information about unverified contracts. After retrieval, contracts and their bytecodes undergo vulnerability assessment using various tools. These tools, integrated through a Vulnerability Assessment API, analyze the bytecode to identify potential vulnerabilities. After analysis, the results are compiled and presented, marking the end of the smart contract acquisition and vulnerability assessment phase.

For effective data management and analysis, all extracted smart contracts, along with their bytecode and ABI data for verified contracts, are stored locally. This storage enables swift access and manipulation of data, which is crucial for timely vulnerability assessments. Generally, ABI data is particularly vital as it allows the conversion of bytecode to the human-readable Solidity format, facilitating the use of static and dynamic analysis tools.

The vulnerability assessment process involves establishing a virtualized environment using Docker technology. This provides a consistent and controlled setting for deploying analysis tools, ensuring result reproducibility. The Virtual Machine (VM) is set up with Debian 5.10 for its reliability and community support. Selected open-source vulnerability assessment tools, (i.e., Oyente, Maian, Osiris, and Mythril) offer diversity in vulnerability detection. These tools are installed in the Docker-equipped VM, configured for persistent operation across system restarts.

In the virtual environment, we develop an API to streamline the vulnerability assessment of smart contracts. This API, implemented in PHP, is essential for managing the submission and processing of smart contracts bytecode. Due to the absence of remote capabilities in the tools, we craft a PHP script capable of receiving bytecode via HTTP POST requests. This script executes vulnerability assessments within Docker containers and returns the results in structured JSON format. The API allows users to specify the type of code-submitted—either bytecode or Solidity source code—and to choose the assessment tool, enhancing automation and enabling batch processing.

The most time-intensive stage of our methodological framework is interpreting and analyzing data from the vulnerability assessments. The assessment phase takes about three months for the roughly 5.3 million contracts included in this study. This critical process transforms raw data into actionable insights, providing a clear picture of the security landscape of Ethereum smart contracts. Our workflow for processing results is straightforward, yet detailed. We begin by parsing the JSON output from each tool, identifying vulnerabilities in the smart contracts. We then compile these results, link them to their respective contracts, and catalog them by vulnerability type and severity.

A key aspect of our analysis is categorizing vulnerabilities into a taxonomy that reflects their nature and

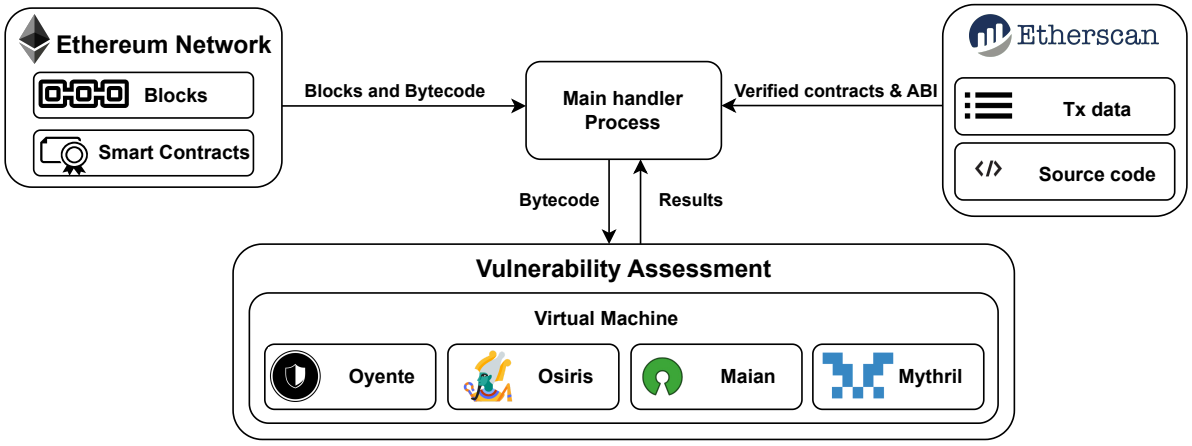

Figure 1: Process followed to acquire and process smart contracts

potential impact. We group vulnerabilities into categories such as reentrancy, integer overflow, or denial of service, and conduct statistical analyses. This involves calculating the frequency of each type of vulnerability, their distribution across contracts, and instances of overlap where multiple vulnerabilities occur in a single contract. Furthermore, we examine trends, such as the rise or fall of certain vulnerabilities over time and their association with specific contract patterns. We also adhere to rigorous ethical research standards to ensure the confidentiality and security of the data. This includes anonymizing contract addresses and complying with all relevant privacy laws and ethical guidelines to protect the security and privacy of contract owners. At a glance, the systematic process for acquiring and analyzing Ethereum smart contracts to assess vulnerabilities is depicted in Figure 1.

5 Evaluation

This section describes our experimental approach, detailing the complete process from gathering and preparing data on smart contracts to conducting vulnerability assessments. By meticulously analyzing a large dataset of verified and unverified Ethereum smart contracts, we aim to illuminate the extent of security vulnerabilities and provide insights into their consequences. Our evaluation demonstrates our method's ability to identify known vulnerabilities and assesses its effectiveness in exposing previously undetected security risks. This underscores the potential for improving the security landscape of Ethereum smart contracts.

5.1 Setting Up the Scene

We conducted all experiments on a machine equipped with an Intel Core i7 6700K processor and 32 GB of RAM. To manage the considerable workload, we ran six clones of our vulnerability assessment VM in parallel, applying load balancing to distribute analysis tasks efficiently. Each VM (Debian 5.10) was outfitted with all the necessary tools housed within Docker containers. Additionally, we set up a web server in each VM to support an API that accepts external submissions of smart contracts bytecode. We automated the assessment processes using scripts that coordinated the workflow: submitting bytecode, initiating vulnerability scans, and compiling and storing results. Each tool was operated through specific commands within their Docker containers to enhance process efficiency and reduce human error. During this study, we processed bytecode and Solidity codes, aggregated the findings, and issued reports. This intensive effort spanned three months, during which we analyzed 4,682,850 smart contracts within the Ethereum blockchain.

5.2 Verified and Unverified Trends

The extracted data (see Figure 2) reveals a nuanced and evolving relationship between verified and unverified smart contracts on the Ethereum blockchain. This evolution reflects a complex interplay of technological, economic, regulatory, and community-driven factors. Over time, there has been a shift towards a higher proportion of verified contracts, underscoring the Ethereum ecosystem's increasing maturity and commitment to robust security standards. The observed trends highlight a clear trajectory of progress within the Ethereum ecosystem. The increasing dominance of verified smart

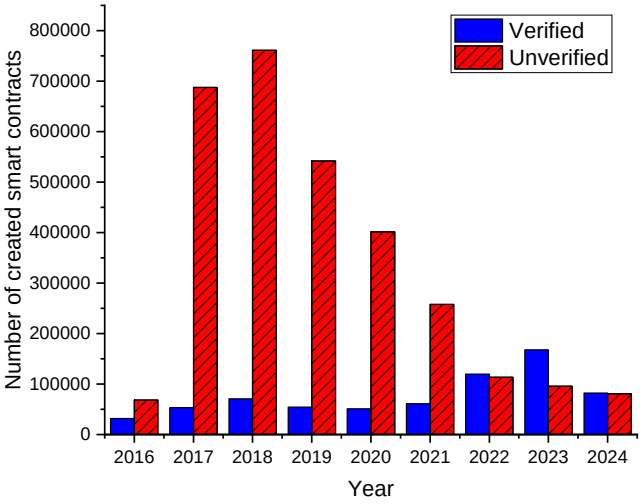

Figure 2: Verified versus unverified smart contracts

contracts, alongside the declining prevalence of unverified ones, underscores the community's collective efforts to enhance security and reliability. Ethereum has positioned itself as a resilient and trusted platform for decentralized applications by addressing vulnerabilities, embracing technological advancements, and prioritizing robust development practices. These trends reflect a maturing ecosystem that continues to adapt to the evolving challenges and opportunities of blockchain technology.

In the early years of Ethereum, particularly in 2016 and 2017, the dominance of unverified smart contracts was notable. During this formative period, 68,202 unverified contracts were deployed in 2016, compared to only 31,796 verified ones. Similarly, in 2017, the number of unverified contracts skyrocketed to 687,371, dwarfing the 53,156 verified contracts. This imbalance can be attributed to several factors, including the novelty of the platform, limited awareness of best practices, and the lack of accessible verification tools. Many developers, attracted by Ethereum's promise of decentralized automation, rushed to deploy contracts without rigorous testing or verification processes. The lack of a robust ecosystem for auditing and verifying smart contracts during this period further exacerbated the prevalence of unverified deployments.

The high-profile security breach involving "The DAO" in 2016, which resulted in the theft of approximately 3.6 million ETH, served as a watershed moment for the Ethereum community. This event highlighted the critical vulnerabilities inherent in unverified and poorly tested contracts, catalyzing a collective awareness of the importance of verification. In the wake of this and similar incidents, the community began to prioritize developing and adopting tools designed to facilitate the verification of smart contracts. This shift is evident in the data, as the proportion of verified contracts increased steadily from 2018 onwards.

By 2018, verified contracts accounted for 70,881 deployments, a significant improvement compared to previous years, even as unverified contracts remained high at 760,987. This year marked a turning point in the Ethereum ecosystem, as the introduction and refinement of tools such as Etherscan's verification services, static analysis platforms like Mythril, and dynamic analysis tools like Oyente enabled developers to integrate security checks into their workflows. This period also saw an increasing emphasis on community-driven initiatives, such as bug bounty programs and educational campaigns, which further incentivized developers to prioritize contract verification.

From 2019 to 2021, the data indicates a gradual but steady decline in the number of unverified contracts, from 541,896 in 2019 to 257,457 in 2021, while verified contracts remained relatively stable or increased slightly, growing from 54,389 in 2019 to 61,009 in 2021. This trend can be explained by the Ethereum community's growing proficiency in smart contract development, the widespread adoption of secure-by-design principles, and the introduction of Ethereum 2.0, which encouraged the migration of many existing projects to the more scalable proof-of-stake infrastructure. Additionally, regulatory scrutiny and heightened awareness of the financial risks associated with unverified contracts likely motivated developers to adopt more stringent security practices.

The years 2023 and 2024 stand out as a period of resurgence in contract deployment, particularly for verified contracts. In 2023, the number of verified contracts surged to 167,593, representing a significant increase from previous years, while unverified contracts fell to 95,773. This trend continued into 2024, with 82,071 verified contracts compared to 80,613 unverified ones, marking the first time verified contracts nearly equaled unverified ones in deployment volume. This shift reflects the Ethereum ecosystem's successful efforts to embed verification practices into the development lifecycle. The adoption of modular and upgradable contract architectures has further contributed to this trend, as developers now have the ability to enhance and extend existing contracts without deploying entirely new ones.

Economic factors have also played a critical role in shaping these trends. Between 2019 and 2021, high gas prices and network congestion created significant barriers to deploying new contracts, particularly for smaller developers and experimental projects. These economic constraints likely incentivized the optimiza-

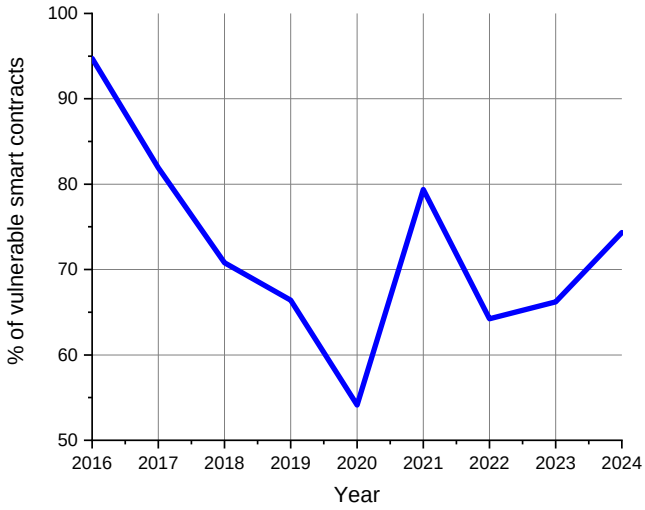

Figure 3: Percentage of vulnerable smart contracts created each year

tion and upgrading of existing contracts rather than the creation of new, unverified ones. By 2023, the adoption of Layer 2 scaling solutions, such as Optimistic Rollups and zk-Rollups, provided a more cost-effective environment for contract deployment, facilitating the resurgence observed in the data.

Another critical factor influencing these trends is the increasing regulatory oversight in the blockchain and cryptocurrency sectors. Developers have responded to the demands for transparency, accountability, and security by adhering to stricter verification protocols. Compliance with standards set by entities like the European Union, through regulations such as the MiCA (Markets in Crypto-Assets) framework, has further incentivized the adoption of verified contract practices. These measures not only mitigate risks but also build trust among users and stakeholders, fostering broader adoption of Ethereum-based solutions.

5.3 Vulnerability of Smart Contracts

Figure 3 illustrates the percentage of vulnerable smart contracts from 2016 to 2024, revealing significant trends that reflect the Ethereum ecosystem's evolving security landscape. The data demonstrates a general descending trend from 2016 to 2019, with the percentage of vulnerable smart contracts decreasing from 94.7% in 2016 to 66.4% in 2019. This marked decline signifies the initial maturation of smart contract development practices. During this period, the Ethereum community became increasingly aware of the critical need for security following high-profile incidents such as "The DAO" breach in 2016. In response, developers began to adopt more secure coding practices and utilize emerging tools for vulnerability detection, such as Mythril, Oyente, and Maian. Additionally, the rise of best practices and educational initiatives helped embed security awareness within the community, leading to more robust and secure deployments.

The percentage of vulnerable smart contracts saw a further reduction in 2018, reaching 70.8%, as developers increasingly integrated static and dynamic analysis tools into their workflows. This progress reflects the broader adoption of secure-by-design principles and the Ethereum community's concerted efforts to address known vulnerabilities such as reentrancy and integer overflows. However, the proportion of vulnerable contracts remained significant, underscoring the persistent challenges in ensuring comprehensive security for all deployments. A notable disruption in this downward trend occurred in 2021, with the percentage of vulnerable contracts rising to almost 80%. While this represented an improvement over previous years, the period was characterized by significant challenges that complicated the security landscape. The rapid expansion of decentralized finance (DeFi) during this time resulted in a surge of new smart contracts, many of which introduced novel functionalities that increased complexity. This complexity, coupled with the industry's accelerated growth, likely outpaced the existing capabilities of auditing tools and practices, leading to undetected vulnerabilities in newly developed contracts. Several external factors in 2020 also contributed to the observed dynamics. The COVID-19 pandemic led to a global shift towards remote work, which, while enabling continued development, may have impacted the thoroughness of security oversight during the design and deployment phases. The transition to remote collaboration tools and altered workflows potentially introduced gaps in communication and coordination among development teams, leading to an increased likelihood of overlooked vulnerabilities. Simultaneously, the pandemic accelerated digital transformation efforts, driving more projects into the blockchain space and contributing to the overall surge in smart contract deployments. Year 2021 also marked the introduction of Ethereum 2.0 and advancements in Layer 2 scaling solutions, which provided developers with more cost-effective and scalable platforms for testing and deployment. These technological shifts enabled broader adoption of automated security checks and enhanced verification processes, partially mitigating the overall vulnerability percentage.

In 2022, the percentage of vulnerable smart contracts declined again to 64.2%, continuing the long-term trend of improved security practices. This decrease coincided with the Ethereum ecosystem's growing em-

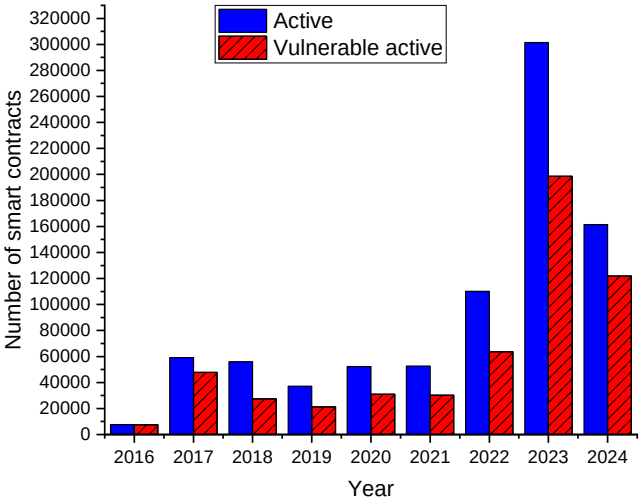

Figure 4: Active smart contracts

phasis on modular and upgradable contract architectures, which allowed developers to address vulnerabilities in existing contracts without redeploying entirely new ones. The period also saw significant investment in developer education and the proliferation of bug bounty programs, fostering a culture of proactive security management. By 2023 and 2024, the vulnerability percentage showed similar values, reaching 66.2% and 74.3%, respectively. This stabilization suggests that while security practices have advanced, the increasing complexity and innovation within the blockchain ecosystem continue to pose challenges. The rise in vulnerabilities in 2024 may reflect the influx of new developers and projects seeking to capitalize on Ethereum's improved scalability and reduced transaction costs. These newer entrants may lack the same level of expertise or access to established security resources, leading to a temporary reversal in the trend of declining vulnerabilities.

The enduring decrease in the percentage of vulnerable smart contracts from 2016 to 2024 underscores the Ethereum ecosystem's continuous efforts to address security challenges. While significant progress has been made, the consistently high proportions of vulnerabilities highlight the persistent need for vigilance and the advancement of security practices. The dynamic nature of the ecosystem, driven by innovation and rapid growth, necessitates ongoing investment in tools, education, and community-driven initiatives to secure the next generation of blockchain applications.

5.4 Active Smart Contracts

The data (see Figure 4) reveals significant insights into the trends of active smart contracts and their associated vulnerabilities over time. Active smart contracts, defined as those with more than 12 transactions in the preceding year, represent the operational backbone of the Ethereum blockchain. Their analysis provides a clear picture of the ecosystem's maturity and its efforts to mitigate security risks.

In 2016, active smart contracts numbered 7,578, with an overwhelming 99.1% classified as vulnerable. This high proportion underscores the immaturity of security practices at the time and the community's limited awareness of the risks inherent in deploying poorly secured contracts. The year 2017 witnessed a substantial rise in active contracts, growing to 59,121. However, the number of vulnerable active contracts also surged to 47,837, representing 81% of all active contracts. This decrease in the vulnerability percentage, although slight, marked the beginning of an increasing focus on security practices within the Ethereum development community, likely influenced by high-profile breaches such as the "Parity Wallet" vulnerability in 2017.

In 2018, the number of active contracts slightly declined to 56,015, while vulnerable active contracts fell significantly to 27,421, accounting for 48.9% of active contracts. This sharp reduction reflects the introduction and adoption of improved auditing tools, security frameworks, and best practices within the developer community. Static and dynamic analysis tools such as Mythril and Oyente gained prominence, providing developers with powerful means to identify and address vulnerabilities before deployment.

The years 2019 and 2020 saw a moderate increase in active contracts, from 37,197 to 52,329, with corresponding increases in vulnerable active contracts, which rose from 21,267 to 30,970. However, the proportion of vulnerabilities remained stable at approximately 59%. This period highlights the growing complexity of smart contracts and the challenges posed by the rapid expansion of DeFi platforms, which often required intricate and novel contract designs. The increased scrutiny by the community and the rise of bug bounty programs during this period helped mitigate the proportion of vulnerabilities, preventing more significant increases despite the growing number of active contracts.

In 2021, the trend of declining vulnerabilities became more apparent, with the proportion of vulnerable active contracts dropping to 57.5%. This decrease, despite the stable number of active contracts at 52,688, signifies ongoing advancements in secure coding practices, the integration of automated auditing processes, and the adoption of modular contract architectures, which facilitated upgrades without necessitating new deployments. A pivotal shift occurred in 2022, with ac-

tive contracts reaching a new peak of 110,047, almost doubling the figures from previous years. Vulnerable active contracts also increased significantly to 63,708; however, their proportion decreased to 57.9%. This period reflects the Ethereum ecosystem's ongoing maturation, characterized by a substantial rise in network activity driven by the adoption of Ethereum 2.0 and Layer 2 scaling solutions. These technological advancements reduced deployment costs, enabling broader participation while simultaneously emphasizing security.

In 2023, the number of active contracts saw a dramatic rise to 301,502, more than doubling from 2022. The number of vulnerable active contracts also increased to 198,568, representing 65.8%. This temporary reversal in the trend of declining vulnerabilities can be attributed to the influx of new developers and projects capitalizing on Ethereum's improved scalability and reduced costs. Many of these projects may not have adhered to the rigorous security practices observed in more mature contracts, highlighting the need for ongoing education and support for new entrants to the ecosystem. By 2024, the ecosystem displayed signs of stabilization, with active contracts declining to 161,479 and vulnerable active contracts dropping to 121,840. This decline in both absolute and proportional terms (75.5%) underscores the Ethereum community's ability to adapt to security challenges, retire insecure contracts, and emphasize robust development practices. The increasing use of automated testing tools, combined with community-driven initiatives such as bug bounty programs and collaborative code reviews, played a crucial role in achieving this outcome.

The observed trends highlight the dynamic nature of the Ethereum ecosystem and its continuous efforts to address security challenges. While the absolute number of active contracts and vulnerabilities has fluctuated, the overall trajectory indicates an ecosystem that is increasingly resilient and adept at managing risks. These findings underscore the critical importance of fostering a culture of security awareness and innovation within the blockchain community to ensure the sustained growth and adoption of smart contracts.

5.5 Top Vulnerabilities

The evolution of Ethereum smart contract vulnerabilities from 2016 to 2024 highlights a dynamic and evolving threat landscape. Some vulnerabilities have persisted over time, reflecting ongoing challenges in smart contract security, while others have been mitigated due to advancements in security tools, best practices, and community-driven initiatives. We delve here into the most prevalent vulnerabilities identified across Ethereum smart contracts. Figure 5 illustrates the top 10 vulnerabilities encountered over this period, and Figure 6 showcases their trends, providing insights into their prevalence and the broader security landscape.

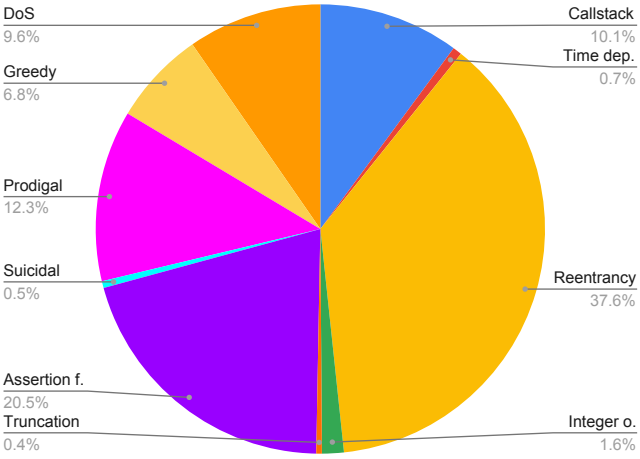

Figure 5: Vulnerable active smart contracts

Callstack. Callstack vulnerabilities, which emerged due to the Ethereum Virtual Machine's (EVM) inability to handle deeply nested calls efficiently, were particularly prominent in the early years of Ethereum. These issues often resulted in DoS conditions that could halt contract execution. While improvements in the EVM architecture and the adoption of robust design patterns have significantly reduced these vulnerabilities, their persistence in 2024 underscores the need for continued technical refinements.

Greedy Contracts. Greedy contract vulnerabilities involve the indefinite locking of funds, preventing their retrieval. Although these vulnerabilities have declined steadily over the years, their ongoing presence—33,283 cases reported in 2024—reflects challenges in implementing effective termination strategies for contracts. Advances in development frameworks and the growing use of automated testing tools have mitigated their prevalence but have not eliminated them entirely.

Reentrancy. Reentrancy vulnerabilities, made infamous by the DAO attack in 2016, allow attackers to recursively withdraw funds before the initial transaction is settled. Despite significant progress in addressing these vulnerabilities through Solidity updates, reentrancy guards, and the adoption of secure libraries like OpenZeppelin, their prevalence remains noteworthy, with 11,839 cases recorded in 2024. This persistence highlights the importance of continuous education for developers and the need for rigorous code reviews.

Integer Overflow. Integer overflows and underflows occur when arithmetic operations exceed data type lim-

its, resulting in unexpected behavior. These vulnerabilities were particularly prevalent in 2017 but have since declined sharply, with only 2,863 cases reported in 2024. The widespread adoption of SafeMath libraries, which enforce secure mathematical operations, has been instrumental in mitigating these issues. However, the emergence of more complex contract logic necessitates ongoing vigilance.

Truncation. Truncation vulnerabilities, from improper data handling during storage or processing, remain relatively rare but impactful. By 2024, only 41 cases were reported, reflecting improved data-handling practices and the integration of comprehensive testing frameworks into development workflows.

Prodigal Contracts. Prodigal contracts, which release funds to unauthorized entities, represent a high-risk vulnerability due to their potential for significant financial losses. A peak of 120,553 cases was observed in 2023, reflecting the growing complexity of DeFi platforms and their associated risks. A reduction to 67,091 cases in 2024 demonstrates progress, driven by improved auditing practices and the adoption of secure design principles.

Assertion Failure. Assertion failures, indicative of flawed logic checks, can halt contract execution or freeze funds, leading to severe impacts. After peaking at 521,488 cases in 2017, their prevalence has declined substantially, with 87,283 cases recorded in 2024. This trend highlights the success of enhanced debugging tools and the community's focus on secure coding practices.

Time Dependency. Time dependency vulnerabilities, which stem from reliance on block timestamps for critical contract functions, have shown a consistent downward trend, with cases dropping to 1,152 in 2024. This reduction reflects widespread adoption of best practices that discourage timestamp dependency, reducing risks associated with miner manipulation.

Denial of Service. Denial-of-service vulnerabilities, caused by excessive gas usage or resource depletion, have remained a persistent issue. While the number of cases peaked in 2018 with 261,022 reported incidents, their prevalence declined to 8,654 by 2024. This reduction is attributed to better contract design and enhanced understanding of gas-related risks among developers.

Suicidal Contracts. Suicidal contract vulnerabilities, which allow contracts to self-destruct under inappropriate conditions, have decreased steadily, with 1,156 cases reported in 2024. This decline reflects improved awareness among developers and better practices for managing self-destruct operations.

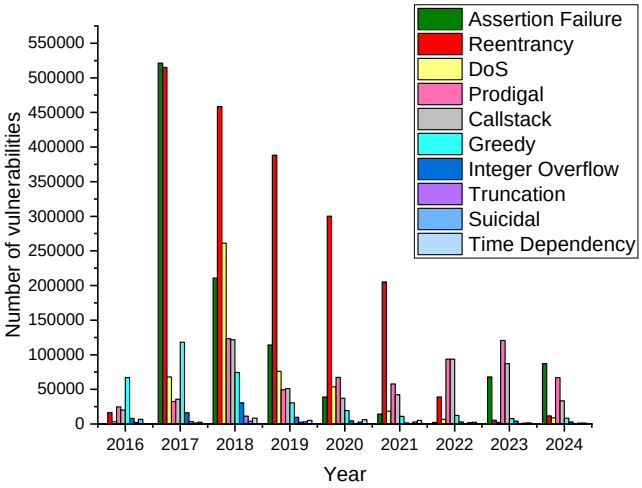

Figure 6: Amount and type of vulnerabilities found in smart contracts deployed per year

The trends observed in these vulnerabilities underscore the Ethereum ecosystem's progress in addressing security challenges. While significant advancements have been made, the persistence of certain issues, such as reentrancy and DoS vulnerabilities, highlights the need for continued vigilance. The community's ability to adapt to evolving threats through education, innovation, and collaboration remains critical to securing the next generation of blockchain applications.

5.6 Vulnerabilities Tendency

The subsequent analysis delves into the underlying reasons for the observed trends in the prevalence of vulnerabilities such as Assertion Failure, Reentrancy, DoS, and others, highlighting the dynamic landscape of smart contract security from 2016 to 2024. These trends reflect the interplay of evolving developer practices, advancements in security tools, and broader ecosystem changes within the Ethereum community. Figure 6 provides a comprehensive visual representation of these changes, illustrating the ongoing challenges and advancements in Ethereum smart contracts.

Assertion failures peaked dramatically in 2017, accounting for 521,488 cases, before undergoing a consistent decline in subsequent years. This initial surge likely stemmed from widespread misunderstandings of state validation processes and improper use of assertions in Solidity contracts. The decline in assertion failures after 2017 corresponds with the adoption of more rigorous testing protocols and an enhanced focus on validating contract logic. The Ethereum community's emphasis on education, alongside the introduction of better debugging tools, contributed to this shift. By

2024, assertion failures decreased significantly to 87,283 cases, marking a 83.3% reduction from their peak. This trend underscores the effectiveness of improved development standards and increased awareness of secure coding practices. Reentrancy vulnerabilities, which were infamously exploited in the DAO attack of 2016, remain a persistent challenge. While the number of cases peaked at 515,115 in 2017, efforts to mitigate this issue, such as the introduction of reentrancy guards and best practices for handling external calls, have led to a steady decline. However, a resurgence in 2020, with 300,201 reported cases, reflects the proliferation of DeFi platforms and the growing complexity of smart contracts during this period. By 2024, reentrancy cases had reduced to 11,839, indicating the success of community-driven initiatives to educate developers and the widespread adoption of libraries like OpenZeppelin's contract templates.

DoS vulnerabilities show a relatively stable but concerning presence across the timeline, with peaks in 2018 and 2020 at 261,022 and 53,743 cases, respectively. The persistent nature of DoS vulnerabilities highlights inherent flaws in gas-limit mechanisms and resource allocation within the EVM. Although the prevalence of these vulnerabilities decreased to 8,654 cases by 2024, ongoing discussions about improving the EVM and implementing more sophisticated resource management strategies remain crucial to addressing these issues comprehensively. Prodigal contract vulnerabilities, characterized by unintentional fund leakage, have demonstrated growth over time, reaching a peak of 120,553 cases in 2023. This increase parallels the expansion of DeFi platforms, which manage substantial assets through increasingly complex smart contracts. The sharp decline to 67,091 cases in 2024 reflects the Ethereum community's proactive measures, including code audits, bug bounty programs, and enhanced contract design practices that prioritize fund recovery and error prevention mechanisms.

Callstack vulnerabilities exhibited a notable reduction from 121,737 cases in 2018 to 33,358 cases in 2024. This trend suggests significant technical evolution in the Ethereum network, potentially facilitated by hard forks addressing EVM limitations. These improvements reduced the risks associated with call depth restrictions and stack overflow errors, providing a more robust environment for contract execution. Greedy contract vulnerabilities, which result in indefinite locking of funds, peaked in 2018 with 74,579 cases and have since steadily declined to 8,327 cases in 2024. This reduction reflects the Ethereum community's focus on implementing code patterns that ensure fund recovery and terminate contract operations gracefully. Initiatives such as early detection tools and community awareness campaigns have further contributed to this improvement.

Integer overflow and underflow vulnerabilities, once prevalent with a peak of 30,510 cases in 2018, have been significantly mitigated, with only 2,863 cases reported in 2024. The adoption of libraries like OpenZeppelin's SafeMath has played a pivotal role in eliminating arithmetic errors, making contracts more resilient to exploitation. Truncation vulnerabilities, while less common, have demonstrated a gradual decline, reaching only 41 cases in 2024. This improvement stems from better data handling practices in Solidity and the use of comprehensive testing frameworks that catch these issues before deployment.

Suicidal contracts, which allow self-destructive behavior under inappropriate conditions, have remained relatively rare but show consistent declines, from 6,686 cases in 2016 to 1,156 in 2024. This trend highlights the effectiveness of community-led initiatives to educate developers about safer self-destruct patterns and conditions. Time dependency vulnerabilities have also decreased significantly, from 8,410 cases in 2018 to just 1,152 in 2024. This reduction can be attributed to best practices discouraging reliance on block timestamps for critical functions, which mitigate risks associated with miner manipulation.

The trends in vulnerability prevalence demonstrate the Ethereum ecosystem's continued evolution and commitment to enhancing security. While significant progress has been made, the persistent challenges posed by certain vulnerabilities, such as reentrancy and DoS, highlight the need for ongoing vigilance, innovation, and collaboration within the blockchain community. These findings underscore the importance of fostering a robust security culture to ensure the sustained growth and adoption of Ethereum-based smart contracts.

5.7 Severity of Vulnerabilities

We utilize a structured method aligned with the Common Vulnerability Scoring System (CVSS) to assess the severity of vulnerabilities in Ethereum smart contracts. This approach provides an objective evaluation framework by scoring vulnerabilities on a scale from 0 to 10, based on factors such as potential damage, ease of exploitation, and broader impact. Below, we detail the severity levels of the top 10 vulnerabilities identified in our research—for a quick overview, see Figure 7:

Reentrancy Vulnerabilities: Scoring consistently between 7 and 9 on the CVSS scale, reentrancy vulnerabilities represent one of the highest-risk issues in Ethereum smart contracts. These vulnerabilities allow attackers

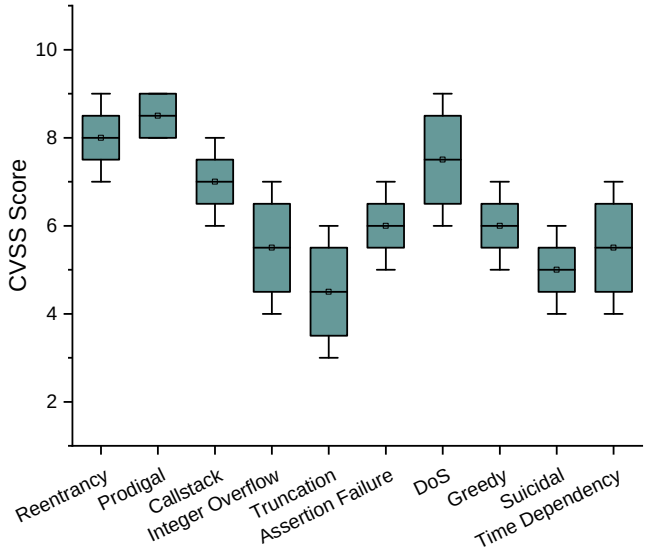

Figure 7: Boxplot visualization of the CVSS score ranges for each type of vulnerability

to recursively deplete funds from a contract, as seen in the infamous DAO attack. Their high score reflects the significant financial harm they can cause, coupled with the relative ease of exploitation in poorly designed contracts. Despite progress in mitigating these vulnerabilities through secure coding practices and reentrancy guards, their persistence underscores the need for vigilant oversight.

Prodigal Vulnerabilities: With CVSS scores ranging from 8 to 9, prodigal vulnerabilities pose severe risks by enabling unauthorized parties to withdraw funds, often leading to immediate and irreversible financial losses. The high prevalence of this issue, particularly in DeFi platforms, emphasizes the need for enhanced contract design practices that prevent unintended fund leakage.

Callstack Vulnerabilities: Scoring between 6 and 8, callstack vulnerabilities highlight the challenges associated with stack depth limitations in the EVM. These vulnerabilities often lead to DoS conditions, rendering contracts inoperable. While updates to the EVM and better coding practices have reduced their prevalence, they remain a critical issue requiring continuous attention.

Integer Overflow: With scores ranging from 4 to 7, integer overflows and underflows can lead to benign issues or critical exploits, depending on the contract's logic. The widespread adoption of secure libraries like OpenZeppelin's SafeMath has significantly mitigated these vulnerabilities. However, their historical prevalence underscores the importance of maintaining secure arithmetic practices, especially as contract complexity grows.

Truncation Errors: These errors, with scores from 3 to 6, occur when data is improperly truncated during storage or processing. Their severity is highly context-dependent, ranging from moderate to high. While the prevalence of truncation errors has declined due to improved data handling practices, their potential impact remains significant in certain scenarios.

Assertion Failures: Scoring between 5 and 7, assertion failures can halt contract execution or freeze funds, resulting in severe operational and financial impacts. These vulnerabilities were particularly prominent in the early years of Ethereum, with a sharp decline in recent years reflecting improved debugging tools and stricter coding standards.

Denial of Service: DoS vulnerabilities score between 6 and 9, depending on their ability to disrupt contract availability and the broader network. Higher scores are attributed to DoS conditions that can cause significant financial loss or facilitate additional exploits. Despite progress in mitigating these vulnerabilities, their persistence highlights the need for continuous improvements in gas efficiency and resource management.

Greedy Contracts: Scoring between 5 and 7, greedy contracts prevent the recovery of locked funds, leading to potential financial losses. While improvements in contract termination strategies have reduced their prevalence, their persistence in certain cases underscores the importance of incorporating fail-safe mechanisms during contract design.

Suicidal Contracts: Suicidal vulnerabilities score between 4 and 6, reflecting their moderate to high impact when exploited. These vulnerabilities allow contracts to self-destruct under inappropriate conditions, potentially disrupting dependent contracts. Improved awareness and development practices have significantly mitigated their occurrence.

Time Dependency: Scoring between 4 and 7, time dependency vulnerabilities stem from reliance on block timestamps for critical functions, leaving contracts susceptible to miner manipulation. While their prevalence has declined significantly, their impact remains significant in certain edge cases.

This CVSS-based scoring framework provides a systematic basis for prioritizing remediation efforts, helping developers and stakeholders focus on critical vulnerabilities that demand immediate attention. Each score is tailored to the specific conditions and risks associated with individual Ethereum smart contracts, ensuring precision in vulnerability assessment.

Determining CVSS scores requires a thorough evaluation of vulnerabilities based on the following criteria:

1. **Base Score Metrics:** These constant metrics include attack vector, attack complexity, privileges required, user interaction, scope, and the direct impact on confidentiality, integrity, and availability.
2. **Temporal Score Metrics:** These metrics, which evolve over time, encompass exploit code maturity, remediation level, and report confidence.
3. **Environmental Score Metrics:** These metrics are customized for specific user environments and include factors such as potential collateral damage, target distribution, and organizational security requirements.

A detailed analysis of these metrics is essential to accurately score vulnerabilities in Ethereum smart contracts. While CVSS offers a standardized framework for evaluation, it must be adapted to each vulnerability's specific context to ensure precise and actionable assessments. By leveraging this framework, the Ethereum community can prioritize security efforts effectively, fostering a safer and more resilient ecosystem.

6 Discussion

The financial impact of the vulnerabilities identified in this study is profound and far-reaching. Exploiting any of these vulnerabilities can result in substantial financial losses for individuals and companies, undermining trust in blockchain technologies and potentially delaying their broader adoption. For instance, the prodigal vulnerability remains one of the most prevalent and severe issues identified, allowing attackers to drain funds from smart contracts. This vulnerability has been particularly problematic in the context of DeFi platforms, which often manage significant amounts of capital, amplifying the financial repercussions of such exploits. The findings highlight critical concerns regarding the use of Solidity as the primary programming language for Ethereum smart contracts. While Solidity has enabled the rapid growth of the Ethereum ecosystem, its design has also contributed to the prevalence of vulnerabilities, such as reentrancy, integer overflows, and time dependencies. This underscores the need for enhanced language design and tooling to address inherent weaknesses. The high prevalence of vulnerabilities identified in this study emphasizes the necessity for adopting better security practices in smart contract development. Companies and developers must prioritize security considerations throughout the development lifecycle to ensure the reliability and robustness of blockchain applications.

Raising awareness of these vulnerabilities is vital for Solidity developers and the broader Ethereum community. Developers must not only understand the risks posed by each vulnerability but also implement effective countermeasures, such as utilizing secure libraries like OpenZeppelin, adhering to best practices for contract design, and conducting rigorous audits. The persistence of vulnerabilities even after deployment highlights the importance of continuous security monitoring and management. Smart contract auditing should not be a one-time effort but an ongoing process, ensuring that deployed contracts remain secure against evolving threats. This study also underscores the progress made in mitigating certain vulnerabilities over time, such as integer overflows and truncation errors, thanks to community-driven initiatives, the adoption of automated testing tools, and improved educational efforts. However, the persistence of other issues, such as reentrancy and denial-of-service vulnerabilities, reveals gaps in current practices and the need for further innovation and collaboration.

Despite its contributions, this work is not without limitations. The analysis focuses exclusively on security vulnerabilities within smart contracts deployed on the Ethereum blockchain, a key area of study within the current research landscape. However, it is important to note that other blockchain platforms, such as Solana, Avalanche, and Hyperledger Fabric, also face unique security challenges that merit investigation. For example, variations in consensus mechanisms, programming environments, and smart contract designs across these platforms may introduce different types of vulnerabilities. Future research should expand its scope to examine these alternative ecosystems, providing a more comprehensive understanding of blockchain security. Additionally, this study primarily examines vulnerabilities at the smart contract level, leaving broader systemic risks—such as those introduced by interoperability protocols or cross-chain interactions—largely unexplored. As the blockchain ecosystem continues to evolve, addressing these interconnected risks will become increasingly critical to ensuring the security and reliability of decentralized systems.

The findings highlight the dual nature of blockchain technology's potential and its challenges. While Ethereum smart contracts have revolutionized decentralized applications, their vulnerabilities underscore the importance of a proactive and collaborative approach to security. By addressing these challenges through innovation, education, and vigilance, the blockchain community can lay a solid foundation for the sustained growth and adoption of this transformative technology.

7 Related Work

Kushwaha et al. [5] meticulously compile and analyze security vulnerabilities in Ethereum smart contracts, establishing a structured taxonomy that supports understanding and remediation efforts for these issues. While crucial for its detailed classification and consolidation of existing knowledge, this work primarily consolidates findings from previous research without directly employing tools for active vulnerability detection. This approach, though valuable, offers an indirect method for identifying potential security challenges within smart contracts, relying predominantly on the analysis of vulnerabilities documented in prior studies.

Tikhomirov et al. [16] introduce SmartCheck, an extensible static analysis tool that detects a broad range of code issues in Solidity, outperforming existing alternatives. While SmartCheck's advancements in static analysis are significant, our research integrates both static and dynamic analysis tools to provide a more comprehensive evaluation of smart contract vulnerabilities. This dual approach uncovers issues that static analysis alone might miss, offering a deeper security assessment of Ethereum smart contracts. Additionally, by including unverified contracts in our study, we cover a wider range of the Ethereum network, enhancing the depth and breadth of our vulnerability assessment. Our comprehensive evaluation, which combines bytecode and source code analysis with a diverse set of tools, represents a progressive contribution to enhancing security measures for Ethereum smart contracts. By expanding our analytical scope to include unverified contracts and integrating various analytical techniques, our work significantly improves the tools for securing Ethereum smart contracts against emerging threats.

Similarly, other studies [11, 13, 15, 19, 20] have explored vulnerabilities within Ethereum smart contracts, primarily focusing on subsets of contracts or theoretical vulnerabilities without empirical support from a blockchain-wide examination. Although these efforts are insightful, they only partially capture the entire landscape of potential security flaws. We still need to scan the entire Ethereum blockchain to collect and holistically examine all smart contracts using static and dynamic analysis techniques, particularly assessing the bytecode. Our research addresses this significant gap, employing an unprecedented scope and methodological rigor to deliver a more detailed and nuanced understanding of smart contract vulnerabilities.

The closest to our work is the study by Khan and Namin [3], which investigates smart contract vulnerabilities by introducing a framework that integrates 17 distinct tools to evaluate the Solidity source code of smart contracts. Although this framework effectively detects potential vulnerabilities, it limits its application to only verified smart contracts. In contrast, our methodology evaluates the bytecode of smart contracts, thus encompassing all contracts on the Ethereum network, including unverified ones. We comprehensively evaluate potential vulnerabilities across the Ethereum network by focusing on four essential tools capable of assessing bytecode. This approach significantly contributes to the field of smart contract security and aids both researchers and developers in mitigating potential security risks.

8 Conclusion

This paper provides a detailed analysis of vulnerabilities in Ethereum smart contracts and delivers essential insights into their security status.Its findings are alarming: out of the entire corpus of scanned contracts, 64.1% were found to have vulnerabilities. This underscores the urgent need to enhance security measures in smart contract development. The financial repercussions of these vulnerabilities can be considerable, potentially affecting the reputations and financial health of the companies involved. Such security flaws may also impede technological progress and slow blockchain technology adoption. To mitigate these risks, developers must emphasize security from the onset of the smart contract development process. Adhering to best practices, including secure coding standards and comprehensive testing, is essential to identify and address vulnerabilities. Furthermore, educational campaigns and training programs tailored for Solidity developers can enhance their understanding of potential risks and encourage adopting secure coding habits. Overall, this study underscores the critical nature of secure smart contract development and offers valuable insights for the community. By tackling these security issues, we can safeguard the integrity of the blockchain network and support the sustained growth and adoption of this innovative technology.

Acknowledgements

This research has received funding from European Commission's Horizon Europe research and innovation programs under grant agreements No. 101070214 (TRUSTEE), No. 101139031 (SAFE-6G), and No. 101092702 (OASEES).

Declarations

Compliance with Ethical Standards.
This article does not contain any studies with human participants or animals performed by any of the authors.

Competing Interests.
The authors declare no conflict of interest.

Research Data Policy and Data Availability.
All data used during this study are included in this published article.

References

1. K. Christidis and M. Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE access*, 4:2292–2303, 2016.
2. D. He, R. Wu, X. Li, S. Chan, and M. Guizani. Detection of vulnerabilities of blockchain smart contracts. *IEEE Internet of Things Journal*, 10(14):12178–12185, 2023.
3. Z. A. Khan and A. S. Namin. Ethereum smart contracts: Vulnerabilities and their classifications. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1–10, Atlanta, GA, USA, 2020. IEEE.
4. S. Khanum and K. Mustafa. A systematic literature review on sensitive data protection in blockchain applications. *Concurrency and Computation: Practice and Experience*, 35(1):e7422, 2023.
5. S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee. Systematic review of security vulnerabilities in ethereum blockchain smart contract. *IEEE Access*, 10:6605–6621, 2022.
6. A. Lazarenko and S. Avdoshin. Financial risks of the blockchain industry: A survey of cyberattacks. In *Proceedings of the Future Technologies Conference (FTC) 2018: Volume 2*, pages 368–384. Springer, 2019.
7. O. Lutz, H. Chen, H. Fereidooni, C. Sendner, A. Dmitrienko, A. R. Sadeghi, and F. Koushanfar. Escort: ethereum smart contracts vulnerability detection using deep neural network and transfer learning. *arXiv preprint arXiv:2103.12607*, 2021.
8. L. Luu, D.-H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269, 2016.
9. B. Mueller. Smashing ethereum smart contracts for fun and real profit. *HITB SECCONF Amsterdam*, 9:54, 2018.
10. I. Nikolić, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor. Finding the greedy, prodigal, and suicidal contracts at scale. In *Proceedings of the 34th annual computer security applications conference*, pages 653–663, 2018.
11. B. Prasad et al. Vulnerabilities and attacks on smart contracts over blockchain. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(11):5436–5449, 2021.
12. P. Qian, Z. Liu, Q. He, B. Huang, D. Tian, and X. Wang. Smart contract vulnerability detection technique: A survey. *arXiv preprint arXiv:2209.05872*, 2022.
13. N. F. Samreen and M. H. Alalfi. A survey of security vulnerabilities in ethereum smart contracts. *arXiv preprint arXiv:2105.06974*, 2021.
14. C. Sendner, H. Chen, H. Fereidooni, L. Petzi, J. König, J. Stang, A. Dmitrienko, A.-R. Sadeghi, and F. Koushanfar. Smarter contracts: Detecting vulnerabilities in smart contracts with deep transfer learning. In *NDSS*, 2023.
15. X. Tang, K. Zhou, J. Cheng, H. Li, and Y. Yuan. The vulnerabilities in smart contracts: A survey. In *Advances in Artificial Intelligence and Security: 7th International Conference, ICAIS 2021, Dublin, Ireland, July 19-23, 2021, Proceedings, Part III 7*, pages 177–190. Springer, 2021.
16. S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov. Smartcheck: Static analysis of ethereum smart contracts. In *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain*, pages 9–16, 2018.
17. C. F. Torres, J. Schütte, and R. State. Osiris: Hunting for integer bugs in ethereum smart contracts. In *Proceedings of the 34th annual computer security applications conference*, pages 664–676, 2018.
18. Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran. An overview on smart contracts: Challenges, advances and platforms. *Future Generation Computer Systems*, 105:475–491, 2020.
19. H. Zhou, A. Milani Fard, and A. Makanju. The state of ethereum smart contracts security: Vulnerabilities, countermeasures, and tool support. *Journal of Cybersecurity and Privacy*, 2(2):358–378, 2022.
20. Y. Zhuang, Z. Liu, P. Qian, Q. Liu, X. Wang, and Q. He. Smart contract vulnerability detection using graph neural networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3283–3290, 2021.