BISCUIT - Blockchain Security Incident Reporting based on Human Observations

Benedikt Putz
benedikt.putz@ur.de
University of Regensburg
Regensburg, Germany

Manfred Vielberth
manfred.vielberth@ur.de
University of Regensburg
Regensburg, Germany

Günther Pernul
guenther.pernul@ur.de
University of Regensburg
Regensburg, Germany

ABSTRACT

Security incidents in blockchain-based systems are frequent nowadays, which calls for more structured efforts in incident reporting and response. To improve the current status quo of reporting incidents on blogs and social media, we propose a decentralized incident reporting and discussion system. Our approach guides users (security novices) towards a classification of their observations using a tiered taxonomy of blockchain incidents. Questions based on previous incidents interactively support the classification. Post submission a security incident response committee then discusses these observations on our decentralized platform to decide on an appropriate response. For evaluation, we implement our model as a decentralized application and demonstrate its practical suitability in a user study.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; **Distributed systems security**; **Domain-specific security and privacy architectures**.

KEYWORDS

Human-as-a-Security-Sensor, Blockchain, Distributed Ledger, Incident Detection

ACM Reference Format:
Benedikt Putz, Manfred Vielberth, and Günther Pernul. 2022. BISCUIT - Blockchain Security Incident Reporting based on Human Observations. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3538969.3538984

1 INTRODUCTION

Despite the widely accepted notion that blockchain systems are secure, security incidents are widespread in permissionless blockchains today. The impact is often measured in millions of dollars, especially when Decentralized Finance (DeFi) applications are targeted. Early famous examples include the DAO attack in 2016 and the Parity Multisig wallet hack in 2017 [3]. With the exponential growth of cryptocurrency market capitalization and the emergence of DeFi, attack frequency has increased enough to warrant weekly newsletters on the latest incidents [14]. While permissionless cryptocurrency-based blockchains are the primary subject of media coverage, similar security issues apply to permissioned blockchains [6, 24]. Permissioned blockchains often secure high-value real-world assets like trade-finance transactions or freight, making them an attractive target for attackers.

Still, current reporting of blockchain security incidents is mostly unstructured and centralized in the form of tweets or blog posts [14]. Structured threat intelligence is hard to find, and security professionals have to scrape together information from a variety of sources. To facilitate targeted responses, more structured reporting and incident response are needed. It is also desirable to decentralize the incident response process among the stakeholders of a permissioned network, as coordination might be required to properly deal with an incident.

In practice there are many attack cases where observations by individual users could help accelerate incident response. Many smart contract attacks are initially discovered by users alerting developers towards problems or inconsistencies [14]. However, finding the appropriate contact information can be difficult as few DApps have consistent and transparent security reporting policies. Even if it is found, users have no guidance on how to report their observations in a structured fashion. Commonly used responsible disclosure standards focus on disclosure of vulnerabilities, not security issues in general [20]. A guided security incident reporting process could benefit all parties affected by an incident. Security experts and developers benefit from more structured and detailed reports, and users benefit from a faster resolution of incidents.

There have been efforts by research and practice to detect vulnerabilities before they are exploited, for example using vulnerability scanners and security audits [3]. However, the majority of incident reporting is still done on social media and blog sites. To guide practitioners towards more structured incident reporting and discussion, this work proposes a novel approach based on the Human-as-a-Security-Sensor (HaaSS) paradigm [10]. We state the following research questions:

RQ1. Can human observations support the detection of blockchain security incidents?

RQ2. How can the incident response process for human-reported incidents be structured and made tamper-proof?

Based on these research questions, we follow the design science research methodology [23] to develop a decentralized blockchain security incident reporting and discussion system. The system focuses on recording human observations of (suspected) blockchain security incidents and their subsequent enrichment and discussion by concerned parties. Metadata of the discussion is recorded

ARES 2022, August 23–26, 2022, Vienna, Austria

ACM ISBN 978-1-4503-9670-7/22/08.
https://doi.org/10.1145/3538969.3538984

on-chain, ensuring transparency in the incident response process. It also prevents tampering by attackers, which would be possible when using out-of-band communication channels.

Application contexts. There are two apparent application contexts for on-chain security incident handling, which we term C_1 and C_2.

In application context C_1, consortium blockchains are now being adopted by large consortia in logistics, finance and other industries [24]. As they are used to track physical assets or large financial transactions, ensuring their security is a major concern. Unlike traditional centralized IT solutions, the platform isn't hosted by a single provider that could be held responsible for securing it. For this reason, a decentralized security incident handling procedure is needed, which allows participants of consortium blockchains to report and discuss possible security incidents.

Application context C_2 concerns decentralized autonomous organizations (DAOs) on public blockchains. DAOs are common in the Decentralized Finance (DeFi) space, where protocols are jointly governed by token holders. This means that token holders must reach a consensus on decisions that affect protocol operation. Such a decision might be needed if a critical vulnerability within the DAO code is discovered and its handling is controversial. Our approach allows users to report these vulnerabilities and to reach a secure and transparent consensus on incident handling.

The paper is structured as follows. We first provide some background on reporting human observations and blockchain threats in Section 2. Thereafter, we propose our model BISCUIT in Section 3 by introducing preliminaries and a formal description. Section 4 elaborates on our taxonomy for reporting observations, consisting of threats and threat indicators. Section 5 then introduces the technical architecture to support the incident reports, along with a description of our implementation on Ethereum. We proceed to evaluate the implementation with a user study in Section 6 and discuss the results in Section 7.

2 BACKGROUND

We first introduce some preliminaries, starting with the human-as-a-security-sensor (HaaSS) paradigm. We elaborate on the state of current automated blockchain threat detection systems and why they fail to prevent novel attacks. Finally, we state our contribution to research.

The HaaSS paradigm. In information security, there have been approaches for a while that try to involve humans in incident detection. For example, almost all email systems provide the possibility to report spam and phishing emails, which is used to block similar emails in the future. However, this has only been done very selectively and for specific problems, and humans have continued to be seen primarily as a problem that should be excluded from IT security as far as possible. However, this view has become outdated, and there is a paradigm shift from human-as-a-problem to human-as-a-solution [36]. In this course, the human-as-a-security-sensor paradigm emerged, attempting to actively involve humans in the security process, as they can be superior to technical security solutions in many situations [10, 33]. Especially security incidents that leave no or very limited usable technical traces, one has to rely on information provided by humans to be considered. However, this raises new problems. For example, people find it challenging to record security incidents in a structured way, especially if they are not security experts. Therefore, it is necessary to provide assistance that provides some structure without overwhelming the human. This problem has already been addressed in first approaches in the literature [32].

Blockchain threat detection. In blockchain research and practice, most attention has been given exploitation of smart contract vulnerabilities [3]. Many different tool-based automated approaches for detecting and mitigating common vulnerability classes have been proposed. They can be classified into two categories: proactive defenses used to prevent attacks, and reactive defenses for previously unknown or hidden vulnerabilities. Proactive defenses include specific languages focused on security, programming best practices, vulnerability scanners and hardening measures applied to the blockchain itself or the smart contract. Reactive defenses aim to prevent exploitation by verifying execution results at contract runtime. They monitor execution and initiate countermeasures if a violation is detected.

The aforementioned proactive and reactive defenses require the contract developers to implement these security measures before deploying the contract. In practice, the main precautions being taken are vulnerability scans, security audits and bug bounty programs. However, these measures may miss novel vulnerabilities, as evident from numerous successful attacks in the Decentralized Finance (DeFi) ecosystem. These attacks are often not only related to software bugs, but require deep understanding of DeFi business logic. Detecting such attacks is difficult with automated systems [3, 34]. Early tool-based approaches for detection like the Forta network[1] and BlockEye [34] use rule-based systems to flag suspicious transactions. However, these systems may miss attacks that are not caught by rules or generate a significant amount of false positives. For such zero-day attacks, sharing intelligence on Twitter and blogs is common practice[2].

Contributions. This work focuses on providing a structured process for users to report human observations of blockchain threats. In summary, we provide the following contributions to research:

- a process for integrating human observed incidents in smart contract incident response
- a taxonomy for blockchain security threats for use in incident reporting
- a decentralized incident discussion model for incident response without a central decision-making entity

3 THE BISCUIT MODEL

This Section introduces our conceptual approach and formal model for the BISCUIT prototype (**B**lockcha**i**n **S**e**cu**rity **I**ncident Repor**t**ing). We begin by stating our goals and establishing some definitions in Section 3.1.

3.1 Preliminaries

3.1.1 Goal. Ideally, the goal of detecting an attack would be to stop it and prevent any malicious consequences. However, few attacks

[1] forta.network
[2] blockthreat.io

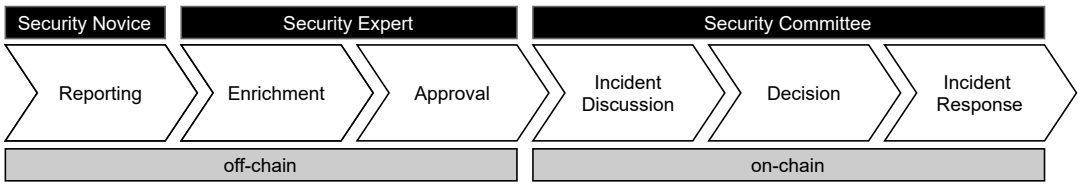


Figure 1: Roles and process steps of the Incident Reporting Flow

on smart contracts can be prevented through reactive defense [3]. The near-immediate and immutable effects of a blockchain transaction make it difficult to reverse attacks. One possibility is to post another transaction to overtake the attacker before their transaction is committed (front-running). Another is a hard fork of the underlying blockchain, which is unlikely unless attack consequences are as severe as Ethereum's DAO hack [3]. Hence, the main goal of incident detection is to minimize damage by reducing funds lost or restoring them as possible.

3.1.2 Requirements. To ensure proper support of incident detection by human observations (RQ1), we state several requirements. To ensure that users are able to report their observations, the model must be *learnable*. In addition, in a decentralized network with semi-trusted participants reporting must be *transparent* and tamper-proof (*integrity*-preserving). In summary we require:

Learnability Users should be able to accomplish their tasks the first time they use the incident reporting tool.
Transparency Anyone with access to the affected blockchain should be able to see and verify incident reports.
Integrity Incident reports should be protected from unauthorized tampering.

3.1.3 Roles. There are three types of participants involved in the incident reporting and response process. Security Novices and Security Experts represent a well-known distinction from literature [1] in terms of corporate security. The Security Committee is introduced as a collective term for all Security Experts responsible for security incident handling.

Security Novices are users who do not necessarily have deep security knowledge. These are usually every employee of a company or, in our case, any DLT user. We assume some basic knowledge regarding blockchain concepts, meaning the user is able to understand block and transaction details shown on a blockchain explorer.

Security Experts are users who have more in-depth security knowledge and therefore a deeper understanding and a broader view of security issues in DLT. An expert can evaluate if a report is an incident, and whether it should be shared with other experts in the Security Committee. He therefore has the necessary permissions to publish security incidents reported by novices.

The **Security Committee** is a set of Security Experts responsible for handling reported security incidents. The aim of the Security Committee is to make a decision regarding an appropriate response. Who constitutes the committee is application and context-dependent. For C1, a sensible choice could be administrators of the organizations' blockchain nodes. In C2, it could be the members of the decentralized development team of a DApp, or shareholders of a Decentralized Autonomous Organization (DAO).

3.2 Formal model

In order to support incident reporting, it is first necessary to capture which elements of an incident can be observed by humans and therefore reported. A frequently used model for this is provided by NIST [13]. In a similar form, the model has already been used in the context of the human-as-a-security-sensor paradigm [32] and is also used in incident reporting, though in a more complex form and thus not suitable for the use case in this work [19]. The goal of the approach is to capture the information of the incident as structured as possible in the form of a normalized incident. As presented in prior work [33], this enables an automated conversion into a common incident reporting format such as STIX[3]. Derived from this, a normalized incident consists of four binary vectors, each representing the elements that can be reported by a Security Novice: Sources $\vec{s}$ that trigger the incident, events that occur during the incident $\vec{e}$, affected entities $\vec{a}$ that are influenced negatively by the incident, and an impact $\vec{i}$ that indicates how serious the incident is considered to be by the Security Novice. Thus, the incident results from the concatenation of the vectors:

$$\overrightarrow{inc} = \vec{s} \frown \vec{e} \frown \vec{a} \frown \vec{i}$$

with every element that has been reported has a value of 1 and every element that has not been reported has a value of 0:

$$s_i, e_i, a_i, i_i \in \{0, 1\}$$

3.3 Incident Reporting Flow

The Incident Reporting Flow (see Figure 1) aims to provide a way to report security incidents in a DLT in a structured way, and to find an appropriate response to them without relying on a central decision point. It consists of six steps executed by different roles, with the first three steps executed off-chain and therefore locally, e.g., at a company. The initial reporting of the security incident, thus the triggering of the process, is done by a novice. The subsequent two steps are performed by an expert within a company. Only the last three steps, which serve the discussion and decision-making for an appropriate response, are processed on-chain. Thereby, the entire consortium can participate. A detailed description of the process steps is given in the following paragraphs.

Reporting. The process begins when the user of the DLT notices an event that he considers to be a security incident. The process itself is then triggered by the user reporting this security incident. To enable the reporting process to be as structured and complete as possible, the user is supported in two stages.

In the first stage, possible security incidents or classes of security incidents that can occur on a DLT are provided in the form of

[3] oasis-open.github.io/cti-documentation/

a taxonomy. With the help of the taxonomy, the user can work his way down from very general classes to very specific security incidents without losing the overview. In addition, the structured reporting allows for subsequent machine processing and similarity calculation that are needed in the next stage. Each element of the taxonomy corresponds to one element of the binary incident vector $\overrightarrow{inc}$. Thus the result of this stage is an instance of the incident vector ($\overrightarrow{INC}_{init}$) with each element selected by the user with a value of 1.

The next stage of reporting is based on the assumption that security incidents in many cases have some similarity to past security incidents. Thus, similar to a recommender system [17, 35] (instead of similar products, similar incidents are suggested) the user can be assisted in completing or correcting the reported security incident. The reported data is compared with past incidents ($\overrightarrow{INC}_{past}$) by calculating the similarity of the reported incident and each past incident in a database containing n incidents:

$$sim(\overrightarrow{INC}_{init}, \overrightarrow{INC}_{past,i}) \qquad i \in [0, n]$$

For calculating the similarity, different measures are applicable. In recommender systems, for example, the cosine similarity is often used for similar problems [17]. The questions are generated based on the differences to the most similar incidents. By answering these questions, the user can refine or correct the data, which results in a corrected version of the reported incident $\overrightarrow{INC}_{corr}$.

Enrichment & Approval. In this phase, the local organization's Security Expert receives the report and contributes additional information from the perspective of a privileged user. Due to elevated or administrative privileges the potential security insight can be enriched with deeper insight or discarded if it is only a temporary technical issue. Otherwise, the Security Expert approves the incident to publish it to the entire Security Committee. Thereby, he validates that the report does not contain any sensitive information and thus does not violate company policies. The approval step can be skipped in the permissionless context C_2, enabling direct submission of incidents by users for discussion by the Security Committee.

Incident Discussion. Published incidents are available to the Security Committee. Each member can contribute their view on the incident through authenticated comments c. Besides viewing the structured incident data provided by the incident issuer, Security Experts can add their own intelligence through attachments. They can also propose an update to the now shared incident by uploading an incident update. This update remains a proposal until a configurable threshold t_{vote} of upvotes for the proposal has been passed, at which point the original incident reference is replaced as part of the voting transaction. Similarly, users can propose to update the default status of the incident to one of the following states. Status updates must be approved with the same voting threshold t_{vote}.

S_0 In discussion
S_1 Response initiated
S_2 Invalid (not a security issue)
S_3 Duplicate
S_4 Completed

Decision & Incident Response. Finally, the Security Committee must come to a consensus on how to deal with the incident. Based on the jointly provided evidence, the Committee may decide to discard the incident, or to initiate an incident response action. To suggest a decision, any Expert may submit a response comment cr_i to the incident. This comment may contain multiple response actions ar_{ij}. The committee can vote on cr_i to approve $ar_{ij} \forall j \in J$. If a configurable threshold t_{vote} is passed, the response actions of ar_i are considered approved.

4 INCIDENT TAXONOMY

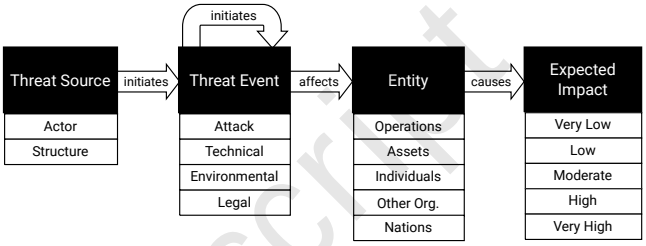


Figure 2: Abstract structure of the taxonomy [33].

To enable structured reporting of security incidents within the smart contracts, a structured collection of possible incidents is necessary. For this purpose, an existing taxonomy for security incidents [33] is extended with blockchain-specific incidents. The underlying structure of the taxonomy is shown in Figure 2. It describes an incident with four elements. Starting with the sources $\vec{s}$ of the attack, which can either be a human actor or a structure such as technical systems or other elements from the organizations' environment. Sources trigger one or more events $\vec{e}$. Intentionally caused events are grouped under attack. The other three classes, Technical, Environmental and Legal, contain events that are not necessarily caused intentionally but are instead of an accidental nature. An event can also be the trigger for other events. In the context of a security incident, an event causes damage to (affected) entities $\vec{a}$ within the company or outside it. These can be operations, assets (e.g. machines), individuals, other organizations or even entire nations. Furthermore, the user can estimate the expected impact $\vec{i}$ (Very Low to Very High), which is reflected within the taxonomy. To extend this taxonomy with blockchain-specific elements, it is necessary to extend the threat sources and threat events accordingly. The potential damage in the form of affected entities and impact of the incident, however, does not differ from those of any other incident.

For the extension of the taxonomy, the extended taxonomy design process of Kundisch et al. [15] is applied. This process enhances the prevalent methodology of Nickerson et al. [22], integrating it with the design science research method of Peffers et al. [23]. The process comprises two approaches, conceptual-to-empirical and empirical-to-conceptual. Since the literature (especially Chen et al. [3]) already provides a comprehensive analysis of possible attacks, we have chosen the empirical-to-conceptual approach, where the taxonomy is created from the already existing dimensions, or in our case, the existing dimensions are integrated into an already existing taxonomy. Furthermore, the taxonomy intended is descriptive (as it aims to describe possible incidents) in its nature rather than normative, which also suggests this approach. As a recommendation

Table 1: Assignment of blockchain sources starting from layer 2 of the taxonomy.

Individual	Insider	Transactor
	Privileged Insider	Peer
		Orderer
		CA Admin
	Outsider	External User
Software	Blockchain-specific Application	dApp
		Smart Contract

for the determination of objects and characteristics, Kundisch et al. [15] propose to conduct a literature review. Following this, a literature analysis is conducted to extend the taxonomy with the root nodes given by the taxonomy to be extended [33].

4.1 Threat Sources

The existing taxonomy already provides a general structure of possible sources $\vec{s}$ of security incidents. However, it does not contain any blockchain-specific elements. At the lowest level, a distinction can be made between actors and structures, which can also be the cause of an incident in the blockchain environment. According to Putz and Pernul [26], five different roles or classes of actors can be distinguished in a blockchain system: Certificate Authority (CA) Admin, Orderer Admin, Peer Admin, Transactor, and External User. To enable in-depth reporting of incident sources in a blockchain environment, we assign them to appropriate nodes in the original taxonomy.

The mapping of the new sources to the existing taxonomy can be seen in Table 1. Insider attacks are especially relevant in the permissioned context C_1 [25]. For example, a Transactor is a regular user, has read access to the blockchain, and can submit transactions. Thus, a transactor is categorized as an insider in C_1. Privileged participants responsible for managing components of the blockchain include Peer, Orderer, and Certificate Authority (CA) Admins. Consequently, these participants are categorized as privileged insiders. External users, in contrast, do not have privileges and are excluded from the system by access control to mitigate vulnerabilities. Thus, the external users are classified in the outsider subcategory. This is the case for most adversarial actors in C_2.

In addition to individual actors, software can be the source of a security incident. According to Shirvas et al. [30], two main types can be distinguished in the blockchain environment: decentralized applications (dApps) and smart contracts. There is currently no suitable class in the existing taxonomy, which is why the category Blockchain-Specific Application has been added. Both dApp and Smart Contract can thus be subordinated to this category on layer 3 of the taxonomy.

4.2 Threat Events

To build our taxonomy of threat events $\vec{e}$, we aggregate vulnerability information from various sources obtained in the literature review. Rameder conducts an extensive classification of smart contract vulnerabilities [27], based on a number of other works such as the DASP TOP 10 [21], the Smart Contract Weakness Classification Registry [31] and other surveys [3]. We also include other types of attacks not specific to smart contracts based on relevant literature [7, 18, 26, 28, 29].

Table 2: Assignment of blockchain events to taxonomy elements.

Nefarious activity / Abuse	Malicious Environment, Transactions or Input
	Network partitioning attack
	Transaction Malleability
	Timejacking Attack
	Malicious Consensus Behavior
	Consensus Configuration Exploitation
	Framework Vulnerability Exploitation
	Identity Provider Compromise
	Mining Pool Attack
Denial of Service	Frozen Ether
	Ether lost in transfer
	DoS with block gaslimit reached
	DoS by exception inside loop
	Insufficient gas griefing
	Gas costly loops
	Gas costly pattern
	High gas consumption variable data type
	High gas consumption function type
	Under-priced opcodes
Authentication and Access Control	Credential compromise
	Cryptographic vulnerabilities
	Sybil attack
	Unauthorized operations
Social Engineering	Address Manipulation
	Homograph attack
Software Bugs	Blockchain/Environment Dependency
	Exception & Error Handling Disorders
	Arithmetic bugs
	Bad Code Quality and Language Specifics
	Dependency vulnerabilities
Technical	Blockchain ecosystem issue

Table 2 lists the new event classes that have been incorporated into the taxonomy. Because of the large number of possible events, not all of them can be listed in this paper, and therefore Table 2 only contains the upper levels. The complete taxonomy is published in the GitHub repository[4].

The event types mentioned in the literature can essentially be assigned to the two categories *Technical* and *Attack*. Depending on whether there is an active malicious action behind the underlying event or whether the event was triggered unintentionally.

The **Nefarious Activity/Abuse** category concerns unintended abuse of software logic. Within this category, Malicious environment, transactions or input covers vulnerability exploits caused by inadequate consideration of a zero-trust environment [27]. In this attack, network participants may attempt to partition blockchain network routing [26, 28]. The goal of such an attack is to tamper with peer communication or the consensus protocol. Threats exploiting the consensus mechanism include malicious consensus

[4] github.com/biscuitsecurity/frontend/tree/master/taxonomy

behavior and manipulation of the consensus configuration [26]. Malicious consensus behavior includes the following attacks: Consensus Delay [26], Alternative History / Double-spending Attack [29], Transaction Reordering, Block Withholding, and Collusion [7]. Manipulation of the consensus configuration can take the form of a batch time attack or block size attack [24, 26]. In addition to smart contracts, the blockchain framework itself may contain exploitable vulnerabilities. An example would be a vulnerability caused by an Unrestricted Chaincode Container or within a framework dependency like CouchDB [7]. In addition, the framework's identity provider may be compromised as the result of a CA, boycott or blacklisting attack [26]. Here, the certificate authority of a blockchain is not trustworthy or does not operate correctly and is therefore either an attacker itself or enables attackers to exploit vulnerabilities. In proof-of-work blockchains, mining pools may attack the network by discarding/withholding blocks or engaging in selfish mining [18].

Denial of Service attacks aim to disrupt the availability of individual systems. Various attack points or vulnerabilities are often exploited for this purpose. Conventional network-based denial of service attacks are already covered by the existing taxonomy. A comprehensive classification of blockchain-specific denial of service attacks on smart contracts is given by Rameder [27]. These incidents predominantly exploit vulnerabilities in the transfer of Ether or vulnerabilities within the determination of transaction fees (gas). For example, weaknesses in smart contracts are exploited in order to interrupt transactions or lock ether (frozen ether) [3]. An example of exploiting vulnerabilities within the transaction fees to carry out denial of service attacks is that transactions whose costs exceed the maximum gas limit of a block are not executed [8]. Exploiting this vulnerability in a smart contract can prevent some transactions from being executed.

Incidents that relate to **Authentication and Access Control** within the blockchain are include known categories that exploit cryptographic vulnerabilities and compromise access credentials. Examples include quantum computing threats [5], hash collision resistance attacks, and digital signature vulnerabilities. In contrast, sybil attacks are specific to distributed systems. Thereby, the network is attacked by creating multiple fake identities to gain disproportionately great influence.

Ivanov et al. [12] provide an overview of possible **Social Engineering** attacks in the blockchain environment or specifically through the exploitation of Ethereum smart contracts. A distinction can essentially be made between address manipulation and homograph attacks. In the case of address manipulation attacks, properties of public addresses are chosen in a way that the user is deliberately deceived. Homograph attacks exploit the similarity of symbols in certain fonts to mislead the user.

Unintentional events are categorized as **Software Bugs** if the issue can be traced back to errors in the development of blockchain applications. Other **Technical** issues may relate to events within the blockchain ecosystem, including blockchain interoperability, oracles or off-chain storage.

4.3 Threat Indicators

From a Security Novice's point of view it is not always easy to recognize the precise type of attack in the taxonomy. For this reason, we establish several *indicators* that represent the outcomes of an ongoing or past attack that are directly observable by a novice. We provide some general indicators below, derived from observable STRIDE elements [16].

Identity abuse. It is observed that a user is impersonated by someone else, i.e. through a compromised identity.
Service unreachable. Users interact with blockchain applications using a website or service, which may be unreachable due to an attack.
Unauthorized modification of data. A user may notice that data has been changed within the application in a way that is not compliant with regular usage, for example due to a vulnerability.
Disclosure of information. Confidential information was observed in a non-private context, implying unauthorized disclosure.

For application context C_2 we determine additional indicators based on an analysis of major smart contract attacks within the DeFi ecosystem [14]. None of these indicators are indicative of an attack by themselves. However, the presence of multiple indicators make an attack likely enough to warrant a report.

Call by contract. The application smart contract is called through a dedicated attacker smart contract created by the attacker (see for example the Reentrancy attack [3]).
Flash loan. While flash loans have legitimate usage for arbitrage and reducing transaction fees, they are also often used as part of attacks that try to manipulate contract logic.
Transaction frequency spike. Attackers often perform many transactions within a short period of time.
Failed transactions. Failed transactions targeting the same contract. Attackers may attempt the same attack with similar parameters multiple times.
Account creation date. Attackers usually create new accounts for attacks, so transactions made by accounts created on the same day are more likely to be used in an attack.
Gas usage. Attack transactions often come with high gas usage and/or high gas prices (for fast execution) [11].
Tornado.Cash. The coin mixing service Tornado.Cash[5] is popular among attackers for its ability to anonymize coin flows if executed correctly. If used after an attack transaction it may indicate an attempt to launder coins.

5 PROTOTYPE

To evaluate the model, we construct a prototype artifact for participants to interact with. First, we propose a generalized architecture for any blockchain based on the current state of the art for decentralized applications in Section 5.1. We further describe our instantiation for Ethereum smart contract incidents in Section 5.2.

5.1 Architecture

The abstract architecture of our framework is shown in Figure 3 and described hereafter.

[5] tornado.cash

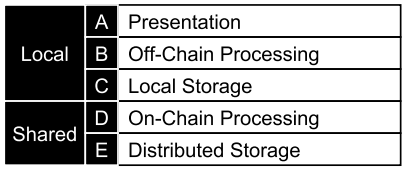


Figure 3: Abstract application components.

As detailed in Section 3, incident handling is initiated by a *Security Novice* or *Expert*. They may submit an incident report using a guided interactive form. This off-chain phase is supported by a website (frontend **A**). The client implements the *Incident Reporting* process that guides Security Novices towards submitting a structure incident report. The reported data is sent to a local server application **B**, supported by a local database **C**. The incident is stored in original and normalized form. After submission the user is then prompted with several questions, which are used to determine similar incidents.

The local database serves as intermediate storage for Novice-submitted incidents. If the local Expert deems the incident report valid, (s)he may add additional information using the frontend based on own knowledge or data only available to privileged users, such as blockchain node logs. Otherwise, incidents can be discarded if it is not relevant. This might be the case if the user has mistaken a scheduled downtime as a security incident, for example.

Valid incidents are approved by the Expert for further discussion with the Security Committee. They are also added to the local storage to serve as reference incidents for future reports. After approval, the incident reference is published to the smart contract **D**, thus proceeding to the on-chain phase. The actual incident data associated with the reference is transferred to the shared storage **E**.

The on-chain phase consists of a comment-based discussion among Security Experts. Figure 4b shows the hierarchical structure of the discussion for a single incident. Each expert can add comments, optionally with attachments, status updates or an updated incident description. Up- and downvoting incidents serves as an indicator for relevance and allows reaching consensus on status and incident updates.

5.2 Implementation

We implement the prototype using a three layer approach: a *Presentation*, *Processing* and *Storage* layer. Figure 4a details the implementation of each layer.

The *Presentation* layer **A** supports both Security Novices and Security Experts in submitting, enriching and discussing incidents. It is implemented as an Angular[6] single page application.

The *Processing* layer **B** provides APIs for the frontend to interact with the *Storage* layer. Besides incident reporting, this includes incident matching, which is based on cosine similarity. The backend is built using Python for its libraries nltk[7] (to generate interactive questions) and numpy[8] (for cosine similarity). The REST API endpoints are built using the Flask[9] library.

The *Storage* layer persists incident and discussion data. Local storage **D** is implemented using MongoDB and stores incident data as JSON documents in original and normalized form. Once original incidents are enriched by the Security Expert and approved, they are moved to a separate collection as *Reference Incidents*. Future user submissions are compared to these reference incidents to improve incident reporting quality. On approval, incident data is also published to the shared storage **E**. The shared storage is implemented using the distributed hash table IPFS[10], a commonly used off-chain DHT database solution. The IPFS reference is used to publish the incident metadata to the Incident Registry smart contract. Our smart contract is based on Solidity, which currently is the most widely used smart contract language, and the most widely studied language in terms of security vulnerabilities due to the popularity of Ethereum in research and practice [3].

The full source code of our implementation is available in multiple GitHub repositories[11].

6 EVALUATION

To determine if our model and prototype meet the goals of research questions RQ1 and RQ2, we perform a two-fold evaluation. First, we perform a user study with Security Novices and Security Experts in Section 6.1. The main goal of the user study is to determine the suitability of our prototype form and underlying taxonomy for reporting common threats. Second, we perform a technical evaluation of the prototype to determine usage cost on permissionless blockchains (context C_2) in Section 6.2.

6.1 User Study

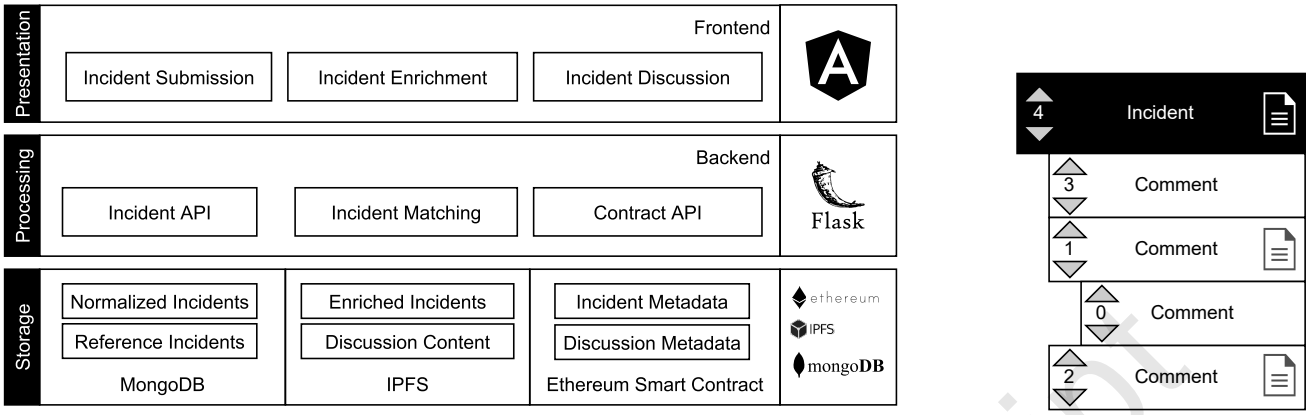
To evaluate the incident reporting model and prototype, we conduct a user study with a set of Security Novices that are familiar with blockchain technology, but not necessarily with blockchain security. We evaluate both application contexts C_1 and C_2 by providing context-specific scenario descriptions.

Setting. For C_1, the setting is a permissioned blockchain for logistics tracking, similar to TradeLens. Users are tasked to update logistics information, but face four separate issues when attempting to do so. The evaluation for C_2 is set in the Ethereum DeFi ecosystem. Participants are asked to review transactions by a single account to determine applicable threat indicators. Due to the complexity of Etherscan for Security Novices and time constraints we only consider two attacks. In total this results in six incident reports per subject for C_1 and C_2.

Participants. To evaluate **incident reporting**, we interview seven participants that have studied and used blockchain applications, with varying levels of prior knowledge. We asked participants to rate their own knowledge regarding blockchain on a Likert

[6] angular.io

[7] nltk.org
[8] numpy.org
[9] flask.palletsprojects.com
[10] ipfs.io
[11] github.com/biscuitsecurity



(a) Prototype layers and modules. **(b) Incident discussion.**

Figure 4: Prototypical implementation of BISCUIT.

scale from 1-7, yielding $\mu = 3.7, \sigma = 1.1$. For blockchain security knowledge, the self-assessment results were slightly worse at $\mu = 2.6, \sigma = 1.3$. Only four of the interviewees had previously used EtherScan. These results confirm that the interviewees are indeed Security Novices.

Procedure. For C_1, we provide a textual description of a security incident. The description includes observations and a screenshot of the interface. The attack scenarios include a website outage, identity compromise, consensus algorithm failure and a sybil attack creating fake identities.

For C_2, we provide the users with the addresses of two real attacks from the public Ethereum blockchain. Both attacks are flash loan attacks abusing an vulnerabilities in the DeFi applications Yearn and CREAM Finance. Users can inspect all relevant transaction via the EtherScan Blockchain Explorer[12]. As a guideline for transaction inspection, we provide the incident indicators established in Section 4.3. The full interview guideline with attack descriptions is available on GitHub[13]

Results. Interviews took on average 60 minutes including task explanations. We observe a slight negative correlation ($\rho = -0.55, p = 0.15$) regarding interview duration and self-assessed blockchain knowledge, although not statistically significant at our sample size.

As attack **sources**, the novices identified outsiders and insiders reliably in the taxonomy. For **events**, based on the available information the users distinguished appropriately between technical issues and nefarious activity. For C_2, transaction frequency spikes, failed transactions and high gas usage were reliably identified. Regarding affected **entities**, participants successfully identified the correct taxonomy elements, although some preferred to note assets and others individuals. **Impact** assessments were at most 2 points apart on our 5 point scale.

[12] etherscan.io
[13] github.com/biscuitsecurity/frontend

We also received useful suggestions towards improving the usability of the prototype. For example, users noted that it was sometimes difficult to find the desired entry in the taxonomy. We improved this navigability issue by adding tooltips for individual leaves in the taxonomy tree, highlighting their child nodes.

6.2 Technical performance

We investigate the costs of using our smart contract on a permissionless Ethereum-based blockchain. Costs are relevant for deployments on permissionless blockchains in C_2, where transaction fees must be paid for on-chain interactions. The costs for each transaction function call are shown in Table 3. Costs are based on the gas consumption of our smart contract at the current gas price and exchange rate. It is quickly apparent that these costs are quite high at current exchange rates. However, considering that the value at stake is often on the order of millions, costs are still manageable. Options for cost reduction are outlined in Section 7.

7 DISCUSSION

We now discuss the evaluation results with regard to the research questions and application contexts. Additionally, we take a look at applicable mitigation options.

7.1 Research questions

We revisit our research questions and discuss whether our prototype and evaluation results support their fulfillment.

Table 3: Gas consumption for our smart contract methods, assuming a Gas price of 36 Gwei and 2500 EUR/ETH.

Method	Gas Used	Ether	Euro equiv.
addIncident	211872	0.00763	19.07 €
voteIncident	44837	0.00161	4.04 €
addComment	280831	0.01011	25.27 €
voteComment	54325	0.00196	4.89 €

Table 4: Overview of ex-post mitigation options.

Mitigation	Description	C1	C2	Timeframe
Front-running	Overtake the attacker transaction in mempool by setting a higher gas price		x	short
Emergency Stop	Initiate emergency blockchain or smart contract action to prevent damage	x	x	short
Negotiation	Pressure attacker into abort by threatening legal investigations	x	x	medium
Deanonymization	Share information about the attacker with exchanges to prevent cashout		x	medium
Legal action	Provide legal authorities with identifying information of the attacker	x	x	long
Hard fork	Reset blockchain state to an earlier block before the attack occurred	x	x	long

RQ1. Can human observations support the detection of blockchain security incidents?

Both use case and user study demonstrate that users of varying knowledge levels are able to contribute to reporting of an incident. The timeliness of the incident report may be critical, i.e. when the attacker may do further damage or when it is critical to prevent cash out. Our approach leaves it to the user to decide whether detailed or fast reporting is more important. In our model, it is only mandatory to report a single Event, and only the top level in the hierarchy must be chosen.

RQ2. How can the incident response process for human-reported incidents be structured and made tamper-proof?

Structured reporting of incidents is enabled by the hierarchical taxonomy used during incident reporting. The resulting JSON document can later be transformed into applicable CTI formats such as STIX [33]. Tampering prevention is enabled by storing incident metadata on the blockchain.

7.2 Application contexts

In application context C_1, each permissioned blockchain node operator may define their Security Expert and deploy an instance of the BISCUIT frontend, backend and databases. The smart contract is shared for the entire network. The Security Committee then consists of the plenum of all Security Experts, discussing and handling incidents as they occur.

For application context C_2, each decentralized application on the permissionless blockchain may deploy its own instance of the entire BISCUIT stack in Figure 4a. The DApp developers may represent the Security Experts. The Security Committee consists of a wider range of users, for example users with a governance stake in the protocol or application.

The cost of performing transactions shown in Section 6.2 may represent a barrier for on-chain reporting or discussion of an incident. However, considering the costs of incidents in C_2, which often range in the millions of dollars, reporting costs are negligible. The cost also serves as a spam deterrent. Users will only report observations they are sure about if they have to pay a small transaction fee for reporting. Regarding discussion costs, if the Security Committee deems the cost of discussion too high on a permissionless blockchain, they may rely on a layer 2 or private sidechain to deploy the smart contract. Side chains such as Arbitrum[14] significantly reduce costs, while maintaining transparency and integrity through rollups on the main Ethereum blockchain.

[14] arbitrum.io

7.3 Incident Response

Once a Security Committee has concluded that an incident is a legitimate threat, it initiates incident response. There are several options that can be used in practice depending on the type of blockchain being attacked. Table 4 shows these options along with the applicable context. A timeframe is given to illustrate the timeframe during which the mitigation option is feasible. Short refers to a time window up to one hour, medium up to several days, and long up to several weeks. Faster responses are preferred, i.e. for front-running the timeframe is usually only several minutes or even seconds before the transaction is included on the Ethereum blockchain[15].

8 RELATED WORK

There are two categories of work related to this paper. The first category focuses on classification of blockchain security threats, while the second concerns proactive and reactive defenses for integrating humans as part of the blockchain incident security response process.

Many researchers have worked on taxonomies specific to blockchain threats [9, 18] and smart contract threats [3]. A smaller number of works classifies threats specific to permissioned blockchains [7, 24, 26]. The important role of humans in threat mitigation has been recognized with regard to humans acting as developers and users of DLT [9]. Social engineering attacks target human users in particular. As a result, humans are in the best position to report such incidents. Several such attacks have been found to be relevant to Ethereum smart contracts [12]. The authors state that fully automated detection of such attacks is impossible, as human judgement is needed to understand smart contract semantics. This further supports the need for structured reporting of human observations, augmenting and complementing automated tool reports.

Proactive defenses of blockchain threats include humans as part of software audits and bug bounty programs. Researchers have studied blockchain-based bug bounty programs to incentivize white hats to report information about vulnerabilities in smart contracts [2]. While conceptually similar, our work focuses on post-exploitation incident response, whereas bug bounty programs target pre-exploitation vulnerability reports.

Reactive defenses involve alerting humans, which act as security analysts to mitigate ongoing threats. Current research includes online transaction monitoring [4] and visualizations for monitoring, tailored to blockchain security [24]. However, a discussion-based reactive approach to decide on a response for arbitrary incidents has not been proposed so far.

[15] dependent on network congestion and gas price set by the attacker

9 CONCLUSION

This research paper proposes a novel decentralized approach for reporting blockchain security incidents termed BISCUIT. The model focuses on learnability for Security Novices, and integrity and transparency for Security Experts. We implement the model using a flexible three-layer approach, combining local and shared components to support multiple application contexts. The evaluation results show that the prototype is able to assist users with reporting arbitrary blockchain incidents. The reports provide structured data, which serves as the basis for incident mitigation discussion and as a reference for future incidents. Our prototype does not yet include specific mitigation recommendations, which we consider a subject for future work. We conclude that our prototype provides a good starting point for structuring user observations, showing a path forward towards structured threat intelligence for blockchain incidents.

REFERENCES

[1] Fabian Böhm, Manfred Vielberth, and Günther Pernul. 2021. Bridging Knowledge Gaps in Security Analytics. In *Proceedings of the 7th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 98–108. https://doi.org/10.5220/0010225400980108

[2] Lorenz Breidenbach, Philip Daian, Florian Tramèr, and Ari Juels. 2018. Enter the Hydra: Towards Principled Bug Bounties and Exploit-Resistant Smart Contracts. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*. USENIX Association, 1335–1352. https://www.usenix.org/conference/usenixsecurity18/presentation/breindenbach

[3] Huashan Chen, Marcus Pendleton, Laurent Njilla, and Shouhuai Xu. 2020. A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses. *ACM Comput. Surv.* 53, 3 (2020), 67:1–67:43. https://doi.org/10.1145/3391195

[4] Ting Chen, Rong Cao, Ting Li, Xiapu Luo, Guofei Gu, Yufei Zhang, Zhou Liao, Hang Zhu, Gang Chen, Zheyuan He, Yuxing Tang, Xiaodong Lin, and Xiaosong Zhang. 2020. SODA: A Generic Online Detection Framework for Smart Contracts. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society.

[5] Wei Cui, Tong Dou, and Shilu Yan. 2020. Threats and Opportunities: Blockchain meets Quantum Computation. In *2020 39th Chinese Control Conference (CCC)*. IEEE, 5822–5824. https://doi.org/10.23919/CCC50068.2020.9189608

[6] Ahaan Dabholkar and Vishal Saraswat. 2019. Ripping the Fabric: Attacks and Mitigations on Hyperledger Fabric. In *Applications and Techniques in Information Security - 10th International Conference, ATIS 2019, Thanjavur, India, November 22-24, 2019, Proceedings (Communications in Computer and Information Science, Vol. 1116)*. Springer, 300–311. https://doi.org/10.1007/978-981-15-0871-4_24

[7] Ahaan Dabholkar and Vishal Saraswat. 2019. Ripping the Fabric: Attacks and Mitigations on Hyperledger Fabric. In *Applications and Techniques in Information Security - 10th International Conference, ATIS 2019, Thanjavur, India, November 22-24, 2019, Proceedings (Communications in Computer and Information Science, Vol. 1116)*. Springer, 300–311. https://doi.org/10.1007/978-981-15-0871-4_24

[8] Neville Grech, Michael Kong, Anton Jurisevic, Lexi Brent, Bernhard Scholz, and Yannis Smaragdakis. 2018. MadMax: surviving out-of-gas conditions in Ethereum smart contracts. *Proceedings of the ACM on Programming Languages* 2, OOPSLA (2018), 1–27. https://doi.org/10.1145/3276486

[9] Tobias Guggenberger, Vincent Schlatt, Jonathan Schmid, and Nils Urbach. 2021. A Structured Overview of Attacks on Blockchain Systems. In *25th Pacific Asia Conference on Information Systems, PACIS 2021, Virtual Event / Dubai, UAE, July 12-14, 2021*. 100. https://aisel.aisnet.org/pacis2021/100

[10] Ryan Heartfield and George Loukas. 2018. Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework. *Computers & Security* 76 (2018), 101–127. https://doi.org/10.1016/j.cose.2018.02.020

[11] Yongfeng Huang, Yiyang Bian, Renpu Li, J. Leon Zhao, and Peizhong Shi. 2019. Smart Contract Security: A Software Lifecycle Perspective. *IEEE Access* 7 (2019), 150184–150202. https://doi.org/10.1109/ACCESS.2019.2946988

[12] Nikolay Ivanov, Jianzhi Lou, Ting Chen, Jin Li, and Qiben Yan. 2021. Targeting the Weakest Link: Social Engineering Attacks in Ethereum Smart Contracts. In *ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021*. ACM, 787–801. https://doi.org/10.1145/3433210.3453085

[13] Joint Task Force Transformation Initiative. 2012. *Guide for conducting risk assessments*. National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.SP.800-30r1

[14] Peter Kacherginsky. 2022. Blockchain Threat Intelligence. https://www.blockthreat.io/.

[15] Dennis Kundisch, Jan Muntermann, Anna Maria Oberländer, Daniel Rau, Maximilian Röglinger, Thorsten Schoormann, and Daniel Szopinski. 2021. An Update for Taxonomy Designers. *Business & Information Systems Engineering* (2021). https://doi.org/10.1007/s12599-021-00723-x

[16] Steve Lipner. 2010. Security development lifecycle - Security considerations for client and cloud Applications. *Datenschutz und Datensicherheit* 34, 3 (2010), 135–137. https://doi.org/10.1007/s11623-010-0021-7

[17] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. 2015. Recommender system application developments: A survey. *Decision Support Systems* 74 (2015), 12–32. https://doi.org/10.1016/j.dss.2015.03.008

[18] Yassine Maleh, Mohammad Shojafar, Mamoun Alazab, and Imed Romdhani. 2020. *Blockchain for cybersecurity and privacy: architectures, challenges, and applications*. CRC Press.

[19] Florian Menges and Günther Pernul. 2018. A comparative analysis of incident reporting formats. *Computers & Security* 73 (2018), 87–101. https://doi.org/10.1016/j.cose.2017.10.009

[20] Darren Mills. 2021. A Standard for Responsible Disclosure in Cryptocurrency and Related Software. RD-Crypto-Spec.

[21] NCC Group. 2018. Decentralized Application Security Project (or DASP) Top 10 of 2018. https://dasp.co/.

[22] Robert C. Nickerson, Upkar Varshney, and Jan Muntermann. 2013. A method for taxonomy development and its application in information systems. *European Journal of Information Systems* 22, 3 (2013), 336–359. https://doi.org/10.1057/ejis.2012.26

[23] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. 2007. A Design Science Research Methodology for Information Systems Research. *Journal of management information systems* 24, 3 (2007), 45–77. https://doi.org/10.2753/MIS0742-1222240302

[24] Benedikt Putz, Fabian Böhm, and Günther Pernul. 2021. HyperSec: Visual Analytics for Blockchain Security Monitoring. In *ICT Systems Security and Privacy Protection - 36th IFIP TC 11 International Conference, SEC 2021, Oslo, Norway, June 22-24, 2021, Proceedings (IFIP Advances in Information and Communication Technology, Vol. 625)*. Springer, 165–180. https://doi.org/10.1007/978-3-030-78120-0_11

[25] Benedikt Putz and Günther Pernul. 2019. Trust Factors and Insider Threats in Permissioned Distributed Ledgers - An Analytical Study and Evaluation of Popular DLT Frameworks. *Trans. Large Scale Data Knowl. Centered Syst.* 42 (2019), 25–50. https://doi.org/10.1007/978-3-662-60531-8_2

[26] Benedikt Putz and Günther Pernul. 2020. Detecting Blockchain Security Threats. In *2020 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 313–320. https://doi.org/10.1109/Blockchain50366.2020.00046

[27] Heidelinde Rameder. 2021. *Systematic Review of Ethereum Smart Contract Security Vulnerabilities, Analysis Methods and Tools*. Diploma Thesis. TU Wien. https://doi.org/10.34726/hss.2021.86784

[28] Muhammad Saad, Jeffrey Spaulding, Laurent Njilla, Charles Kamhoua, Sachin Shetty, DaeHun Nyang, and Aziz Mohaisen. 2019. Exploring the Attack Surface of Blockchain: A Systematic Overview. https://doi.org/10.48550/arXiv.1904.03487

[29] Gjorgji Shemov, Borja Garcia de Soto, and Hoda Alkhzaimi. 2020. Blockchain applied to the construction supply chain: A case study with threat model. *Frontiers of Engineering Management* 7, 4 (2020), 564–577. https://doi.org/10.1007/s42524-020-0129-x

[30] Mahendra Kumar Shrivas, Thomas Yeboah Dean, and S. Selva Brunda. 03.01.2020 - 05.01.2020. The Disruptive Blockchain Security Threats and Threat Categorization. In *2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*. IEEE, 327–338. https://doi.org/10.1109/ICPC2T48082.2020.9071475

[31] SWC. 2022. Smart Contract Weakness Classification and Test Cases. http://swcregistry.io/.

[32] Manfred Vielberth, Ludwig Englbrecht, and Günther Pernul. 2021. Improving data quality for human-as-a-security-sensor. A process driven quality improvement approach for user-provided incident information. *Information & Computer Security* 29, 2 (2021). https://doi.org/10.1108/ICS-06-2020-0100

[33] Manfred Vielberth, Florian Menges, and Günther Pernul. 2019. Human-as-a-security-sensor for harvesting threat intelligence. *Cybersecurity* 2, 23 (2019). https://doi.org/10.1186/s42400-019-0040-0

[34] Bin Wang, Han Liu, Chao Liu, Zhiqiang Yang, Qian Ren, Huixuan Zheng, and Hong Lei. 2021. BLOCKEYE: Hunting for DeFi Attacks on Blockchain. In *43rd IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 17–20. https://doi.org/10.1109/ICSE-Companion52605.2021.00025

[35] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *Comput. Surveys* 52, 1, Article 5 (Feb. 2019), 38 pages. https://doi.org/10.1145/3285029

[36] Verena Zimmermann and Karen Renaud. 2019. Moving from a "human-as-problem" to a "human-as-solution" cybersecurity mindset. *International Journal of Human-Computer Studies* 131 (2019), 169–187. https://doi.org/10.1016/j.ijhcs.2019.05.005