Mitigating the Ronin Protocol Vulnerability in the Context of RBAC Policy

Halyna Huzenko and Leonid Galchynsky
Educational and Research Institute of Physics and Technology, Igor Sikorsky Kyiv Polytechnic Institute, Beresteiskyi Avenue 37, Kyiv, Ukraine
gguzenko989@gmail.com, hleonid@gmail.com

Keywords: Vulnarability, Ronin Protocol, RBAC Standard, ERD, Simulation Modelling.

Abstract: The article discusses the structure of the Ronin protocol and its components, focusing on consensus mechanisms and validators. The purpose of the study was to identify the vulnerability of the protocol and to develop methods for its resolution. It was determined that the bridge component of the protocol has a certain vulnerability. Analyzing and investigating the structure and mechanics of Ronin smart contracts, it was found that all validators are Bridge Validators. This prompted a more detailed study of the protocol structure. Audits for 2022 and 2023 were analyzed, which indicated the presence of privileged functionality in some roles in the system. The conclusion was that the protocol has an unformalized role-based access distribution model. By comparing with the NIST standard, it was found that the role-based access control system in the Ronin protocol (Ronin RBAC) is a Flat Model. By increasing the level of the model to the level of the Restricted Model, it was possible to increase the security level of the protocol. Using the MySQL environment, a simulation model was developed that confirmed the vulnerability of the considered access control system. Based on the analysis of the standard requirements, steps were formulated to make changes to the simulation model. To solve this problem, it was proposed to change the role model of access distribution to Level 3 of the NIST RBAC standard.

1 INTRODUCTION

Decentralized finance is becoming increasingly popular. DeFi protocols that use blockchain technology for secure and transparent record-keeping are becoming increasingly widespread. The key area of blockchain application is cryptocurrencies, and a notable innovation is sidechains.

A sidechain is a type of blockchain scaling solution that allows the creation of new independent blockchain networks that can interact with an existing blockchain. They are considered to be the 2nd layer of the Ethereum protocol, extending and inheriting its security guarantees while providing scalability and speed. The relevance lies in the fact that they are able to eliminate some of the scalability and functionality limitations of existing blockchain networks such as Bitcoin and Ethereum.

This allows for the creation of specialized ones that can help reduce the load on the main network, increase throughput, and improve overall performance.

The simple blockchain architecture hides a serious problem: it is impossible to achieve decentralization, security, and scalability at the same time. So, to get a secure and decentralized blockchain, you need to sacrifice scalability.

This problem can be solved with the help of sidechains - independent blockchains with their own security rules. A layer 2 blockchain interacts with Ethereum by sending transaction packets to ensure security and decentralization without changing the Ethereum protocol. This allows layer 1 to manage data security and availability, while layer 2 provides scalability. Layer 2 sends completed proofs back to layer 1 and removes the transaction load from layer 1. Many sidechains also include binding mechanisms to securely move assets between the main blockchain network and the sidechain.

One of the well-known sidechain implementations is the Ronin protocol, developed specifically for the Axie Infinity game [1]. Initially, it used the Proof-of-Authority consensus algorithm for low fees and fast transactions, but later switched to a hybrid of Proof-of-Authority and Proof-of-Stake

mechanisms. Ronin's main goal is to increase transaction efficiency and profitability, serving as a valuable solution for blockchain-based games, especially in the Play-to-Earn (P2E) genre. Developers are increasingly exploring its adaptability for other games in the blockchain ecosystem.

However, while solving the problem of the underlying network to some extent, the Ronin protocol has its own problems, primarily security issues.

We have organized this paper as follows: in Section 2, we describe the Ronin protocol and analyze its vulnerabilities. In Section 3, we present the result: a new RBAC model of the Ronin protocol and conduct simulations in the MySQL Workbench environment to show how the vulnerability of the Ronin protocol can be reduced by changing the roles of network participants.

In the fourth section, we summarize the work and offer suggestions for further research.

2 STRUCTURE OF THE RONIN PROTOCOL AND ITS VULNERABILITIES

The Ronin protocol is developed by Axie Infinity, one of the leading blockchain game developers. The goal is to introduce a traditional economic system into the P2E (play-to-earn) game model.

2.1 Structure of the Ronin Protocol

The main elements of the sidechain are:

1) The main blockchain: this is the blockchain used to store the main data and ensure consensus between users.
2) Smart contracts: These are applications that enable complex transactions on the blockchain, including the transfer and storage of assets. Smart contracts can be deployed both on the main blockchain and on sidechains.
3) Sidechain is a blockchain that runs in parallel with the main blockchain, providing security and consensus with it. Sidechains can have their own consensus rules and security mechanisms.
4) Bridges: these are elements that allow assets to be transferred between sidechains and the main blockchain. Bridges provide security and consensus between blockchains.
5) Messaging protocols: These are protocols that allow messages to be exchanged between sidechains and the main blockchain [2].

Transactions can be used to transfer assets between users, make payments within the network, and interact with third-party services such as network researchers, exchanges, etc.

Smart contracts are used to define the conditions for executing transactions, where the contract code is executed automatically when certain conditions are met [3].

The Ronin protocol supports all tokens, including the main regulatory token RON. Users can pay for transactions on the platform and use DeFi features such as community management and rewards through validators. With the PoS (Proof of Stake) mechanism, validators stake 250,000 RON to receive the right to verify blocks and reward in RON. RON holders can delegate their tokens and become delegates of validators, receiving rewards in RON and participating in network governance. The rewards are distributed between validators and their delegates, and if a validator fails to create correct blocks, its share of RON may decrease, as well as the public share of RON for accepting applications [4].

2.2 Ronin Consensus Algorithm

The Ronin protocol uses a PoA (Proof of Authority) network as a verification mechanism. In the Ronin network, authorities are certain nodes that are responsible for confirming transactions and creating new blocks. They are pre-selected by Sky Mavis itself. Each authority has its own public key, which it uses to sign blocks and transactions. The advantage of PoA is that it does not require the high computing power and energy consumption required for PoW (Proof of Work) mining. In addition, PoA provides speed and scalability of transactions, which allows Ronin to process large amounts of data. However, PoA also has its drawbacks: it can be less decentralized than PoW, as control over the network is in the hands of elected authorities. In addition, PoA does not allow users to mine and receive rewards for maintaining the network. This algorithm is too centralized and has led to the possibility of the largest attacks [5].

To modernize the consensus mechanism, the developers chose the modifiable DPoS. Its advantage over PoS is the presence of delegates, which makes the network even more decentralized [6]. The difference between these two mechanisms can be seen in Figure 1.

The most significant change is that token holders can vote for themselves or delegate a share to a representative. The more tokens a validator receives, the higher his or her chance of being elected. This is

how the validator reliability system works. Each block created on the network must be confirmed by validators, otherwise it can be fined. This helps to maintain network security and the discipline of validators. Thus, the interaction between validators and the Ronin protocol ensures the security, stability, and efficiency of the network.

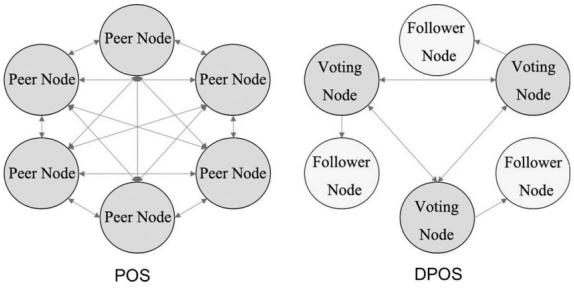

Figure 1: The difference between consensus mechanisms [7].

2.3 Smart Contracts of Ronin Protocol

Like other protocols for blockchain networks [8, 9], the RONIN protocol has a list of smart contracts. By definition, a smart contract is a program code that contains terms and rules that are automatically executed under certain conditions. Smart contracts are a safe and reliable means of ensuring compliance with the terms of contracts, as they are executed automatically without the involvement of intermediaries. Currently, there is a package of smart contracts that support the fulfillment of DPoS consensus conditions. Smart contracts in the Ronin protocol have an impressive set of features: In particular,

1) Transaction processing: Smart contracts in RON can be used to automatically process transactions between users.
2) Token creation: Smart contracts can be used to create and manage tokens on RON. This can be useful for issuing your own tokens, stablecoins, or organizing an ICO.
3) Revenue distribution: Smart contracts can be used to automatically distribute revenue to different participants.
4) Organization of voting: Smart contracts can be used to organize voting for various decisions. The contract can collect votes from different participants and automatically tally the results.
5) Asset management: Smart contracts can be used to manage assets in the marketplace. For example, the contract can automatically distribute profits.

According to the audit by Verichains Lab, the interaction of smart contracts and sidechains with the result of recording the main information looks like in Figure 2:

It is also worth noting such an important type of smart contract as the Ronin Bridge smart contract. Ethereum and Ronin are two separate networks with different protocols, and assets cannot be directly transferred between them. To transfer assets from Ethereum to Ronin or vice versa, users must go through the Ronin Bridge. Ronin Bridge smart contracts are written in Solidity using the OpenZeppelin library and are designed to support multiple chains. When an event deposit occurs on the main chain, the Bridge validator component picks it up and passes the corresponding transaction to Ronin. For output and control events, it starts with Ronin and then passes it to the other chains.

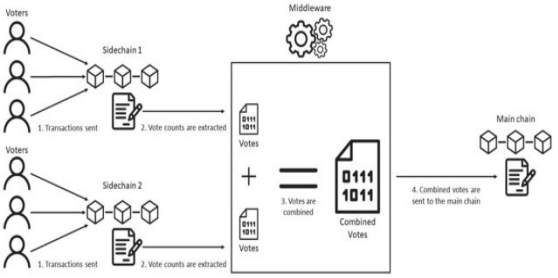

Figure 2: Scheme of interaction between smart contracts and sidechains [10].

2.4 Vulnerability Analysis of the Ronin Protocol

Like all blockchain networks, Ronin is at risk of hacker attacks, as these peer-to-peer exchanges, unfortunately, have vulnerabilities. The topic of vulnerability of blockchain systems has recently received considerable attention, with in-depth analytical studies providing a taxonomy of vulnerabilities that correspond to the general blockchain technology stack [11]. However, there is a lack of such materials specifically for the Ronin network. The fact that the issue of Ronin vulnerabilities is currently relevant is evidenced by the fact that in March 2022, the Ronin network suffered one of the largest DeFi hacks to date. The attackers stole approximately USD 624 million. The hack took place through compromised Ronin and Axie DAO validator nodes. This resulted in the compromised private keys being used to fake two fake network hijackings that swallowed hundreds of millions of cryptocurrency from the bridge. An investigation by independent companies found that

the Ronin protocol was too centralised, with no clear separation of access and monitoring. At least one of the validator nodes had privileged permission to provide its signature to other nodes via a whitelist. The CryptoPotato audit [12] raised the issue of the PoA mechanism's flaw and what causes it - the consensus on the proof-of-authority rule, which is part of the protocol's problems. The audit focused on smart contracts, code and database quality, compliance of smart contract logic with requirements, cross-references to the structure, and implementations based on similar smart contracts. The analysis showed that the "significant" category includes functions with the problem of privilege distribution in the role-based management model. The "significant" category indicates that the built-in logic of rights and privileges is quite vulnerable and requires appropriate changes.

As an example, for the MainChain-GovernaceAdmin smart contract, the Governor role can be assigned the Relayer_Role, which will elevate it to GovernanceRelay. And this, in turn, can lead to critical roles being assigned to validators, allowing them to administer the blockchain network. This is unacceptable, as, by definition, a validator is a cryptocurrency user who verifies blocks and transactions, and only that. Granting an administrator role to a validator account is an abuse of privilege. Obviously, the Ronin RBAC protocol has an informal model of access delimitation, which is manifested in the available roles and the provision of access to certain functionality only if the role is available. In particular, assigning an administrative role to a validator leads to a conflict of interest when a hacker, having hacked into the validator's account, can immediately gain administrator access. We need to look for ways to solve this problem. And using the NIST RBAC role assignment model [13] can help.

3 RESULTS

Role-Based Access Control (RBAC) is a security methodology that is based on managing user access to protect resources and is a standard developed by the National Institute of Standards and Technology (NIST) of the United States. Currently, according to this standard, the separation of roles can be implemented at one of four levels, each of which has the capabilities of the previous one:

1) Flat RBAC.
2) Hierarchical RBAC.
3) Limited RBAC.
4) Symmetric RBAC.

Given the limited size of the article and the needs of use in building the model, we will limit ourselves to considering the first three levels, which are sufficient for model development.

Flat RBAC can support user access permissions to resources through roles. In this case, the procedure for granting user roles on a many-to-many basis, as well as assigning many-to-many permissions to roles. Users can use the permissions of several roles at the same time.

Hierarchical RBAC supports all the functionality of flat RBAC and additionally supports a hierarchy of roles (partial, arbitrary, or limited order).

Restricted RBAC has all the functionality of hierarchical RBAC and must additionally provide a separation of duties (SOD). Figure 3 shows a schematic representation of hierarchical RBAC with restrictions.

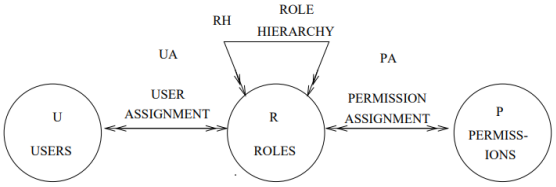

Figure 3: Hierarchical RBAC with restrictions. [16].

3.1 The RBAC Ronin Model

Based on the protocol description, we define roles and resources as entities as follows:

1) Governor(ID, ProposalDurance, Trusted Organisation, Signature) - responsible for the validators that were selected by the Proof of Authority (PoA) consensus mechanism. This entity has access to vote on proposals and submit them.
2) Validator(ID, OnlyAdminID, GovernorID, RelayerRoleID) - an entity representing a regular validator who is assigned certain roles. Each validator has a unique identifier (ID) and can have additional roles, such as the administrator role (OnlyAdminID), the governor role (GovernorID), and the relayer role (RelayerRoleID) to transmit offers from another network.
3) RelayerRole(ID) - an entity that has the right to relay offers from another network.
4) OnlyAdmin(ID, Access) - an entity representing a regular validator to whom certain roles are assigned. Each validator has a unique identifier (ID) and can have additional roles, such as the administrator role (OnlyAdminID), the governor role (GovernorID), and the relayer role

(RelayerRoleID) to forward offers from another network.

According to the description of the protocol, the relationships between entities can be shown in the Figure 4 as an ERD diagram in UML notation. Let's see what level of the RBAC standard the model shows

1) Ronin users access functions through rolesıo. For example, only onlyAdmin gets permission to change thresholds.
2) Ronin provides the ability for one user to assume multiple roles, while the same.
3) Roles in Ronin can be assigned to many permissions.
4) The logic of the protocol states that the Validator can use the Validator and GovernorID permissions.
5) All four of these features correspond to the flat RBAC level.

Since there are no hierarchical relationships and restrictions in the scheme, it can be argued that the Ronin protocol implements the first level according to the NIST standard, namely flat RBAC.

3.2 Modeling the Vulnerability of the Ronin Network

Let's now show the existence of a Ronin protocol vulnerability in this role model by simulating it. We will use MySQL Workbench as a simulation environment. Roles that will be further considered and implemented:

OnlyAdmin, RELAYER_ROLE, and OnlyGovernor. These roles can be assigned to the validator user, which is the user who appeared in the previous sections and attack scenarios. For modeling, a database corresponding to the ERD in Figure 4 was developed and several procedures were created: PauseContract; ResumeContract; UpdateContract-Address; SetThreshold; RelayProposal; Governor-Propose; ReplaceBridgeValidator; GovernorVotes; ChangeTrustedOrganization.

Creating the Validator entity looks like this:

```
CREATE TABLE IF NOT EXISTS `Validator`
(
  `ID` int NOT NULL PRIMARY KEY,
  `OnlyAdminID` INT, `GovernorID` INT,
  `RelayerRoleID` INT,
  FOREIGN KEY (`GovernorID`) REFERENCES
`onlyGovernor`(`ID FOREIGN KEY
(`RelayerRoleID`) REFERENCES
RelayerRole`(`ID
  FOREIGN KEY (`OnlyAdminID`)
REFERENCES `onlyAdmin`(`ID`).
```

This means that a validator can have several roles if they are assigned by the corresponding smart contract:

```
INSERT INTO `Validator` (`ID`,
`OnlyAdminID`, `GovernorID`,
`RelayerRoleID`)
VALUES
(1, 87250316, 91263457, 38529741)
```

– has all the roles,

`(2, NULL, 57389126, NULL)` – validator is assigned via PoA,

`(3, NULL, NULL, NULL)` – the validator is assigned via DpoS.

After describing all three validators with the DESCRIBE statement, we will execute several test queries to confirm at least one attack scenario in which a validator tries to perform the administrative function of changing the contract address. On Figure 5 validators who have not been assigned this role will be denied access.

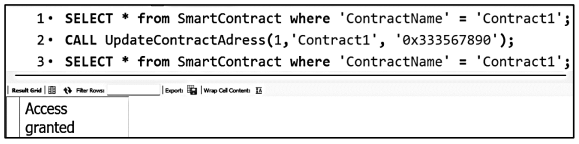

Figure 5: Access of Roles.

A validator with the Administrator role assigned, even temporarily, has access to critical functions, including changing the OnlyGovernor Trusted Organization. After that, the hacker can access the validator's signature through a fictitious organization.

The hacker can also change the delay time between smart contracts to execute them at the same time. Then, returning all the Trusted Organizations with the Validators' signatures. Then the hacker is able to take over the network as follows:

1) Obtain 10 out of 22 validator positions in the Delegated Proof of Stake (DPoS) mechanism.
2) Intercept the session information of a validator wallet that has been recently spotted with administrator activity by capturing cookies.
3) Obtain a signature and take over the account.

We will not describe further steps, but cyber incidents with the Ronin network began with this.

How can this vulnerability be mitigated? Our answer is to change the role policy.

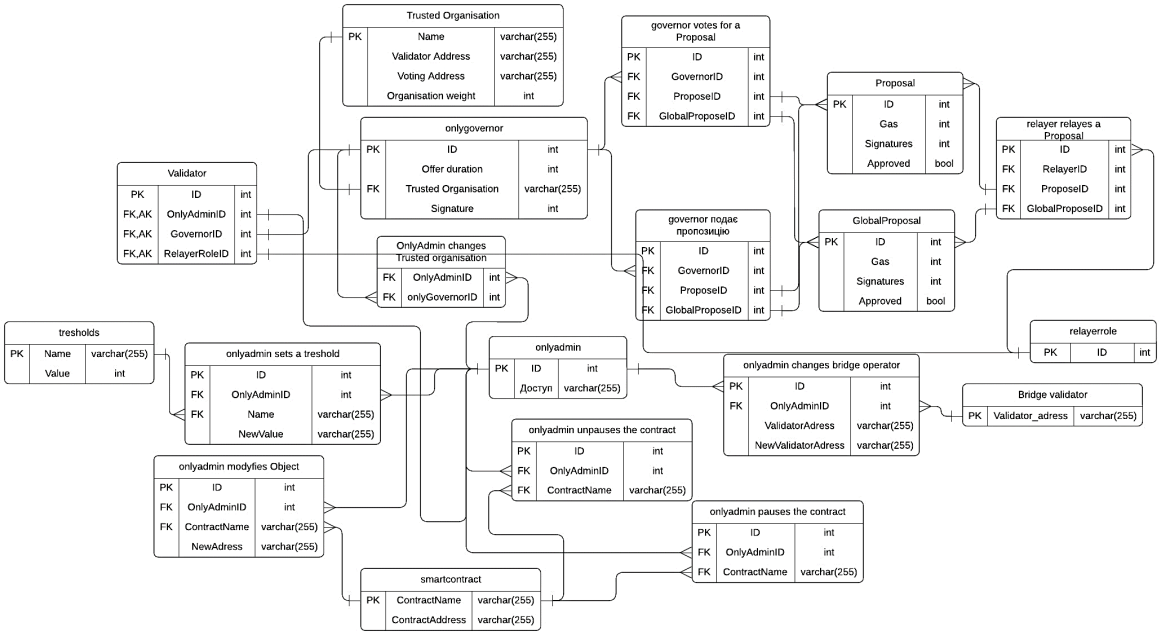

Figure 4: ERD diagram of the Ronin protocol in UML notation.

3.3 Change of Roles in the Ronin Protocol

Let's try to raise Ronin's RBAC level to higher levels. To avoid accidentally assigning the RelayerRole to a user with the onlyGovernor role, which could lead to an unauthorized increase in privileges, we will make the RelayerRole a parent and the onlyGovernor a child. Then the system will always see which of the validators has the RelayerRole. At the same time, the OnlyAdmin role should be made the highest, that is, in our case, parent to the RelayerRole. Thus, the Ronin role model is raised to the level of hierarchical RBAC.

The Validator mustn't have access to administrative functions, because its address is in the public network and it is more vulnerable. This can only be realized through restriction. It is necessary to restrict the role of OnlyAdmin to use the functionality by the separation of duties (SOD), and it will be appropriate to use a static restriction so that the user who has the Validator client on his system does not have access to administrative functions in case the attacker left a backdoor. This will block many potential attack vectors. Thus, the protocol will have RBAC of the third level according to the NIST standard.

The RelayerRole hierarchy was added using the GovernorID foreign key:

```
CREATE TABLE IF NOT EXISTS
`RelayerRole` (
  `ID` int NOT NULL PRIMARY KEY,
  `GovernorID` INT UNIQUE NOT NULL,
   FOREIGN KEY (`GovernorID`)
REFERENCES `onlyGovernor`(`ID`)
Conclusions
```

To implement a static restriction, you need to create a trigger to prohibit a user from having the OnlyAdmin role with any other role.

```
DELIMITER //
CREATE TRIGGER forbid_onlyadmin
FOR EACH ROW
BEGIN
    DECLARE governor_count INT;
    DECLARE admin_count INT;
    DECLARE relayer_count INT;
         SELECT COUNT(*) INTO
governor_count FROM `onlyGovernor`
WHERE `ID` =
NEW.`GovernorID`;
    SELECT COUNT(*) INTO admin_count
FROM `onlyAdmin` WHERE `ID` =
NEW.`OnlyAdminID`;
    SELECT COUNT(*) INTO relayer_count
FROM `RelayerRole` WHERE `ID` =
NEW.`RelayerRoleID`;
       IF governor_count > 0 AND
admin_count > 0 THEN
        SIGNAL SQLSTATE '45000'
```

```
            SET MESSAGE_TEXT = 'Cannot
have both OnlyGovernor and OnlyAdmin
roles';
  END IF;
 IF governor_count > 0 AND
relayer_count > 0 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Cannot
have both RelayerRole and OnlyAdmin
roles';
...
```

The above role changes entail changes in the ERD for the Ronin protocol. Unfortunately, the length of the article does not allow us to provide an updated diagram, but we note that the Validator has changed to a User, which will not allow the Validator to get the Administrator role. The RelayerRole is now a parent to OnlyGovernor, which will allow you to assign individual validators, while simplifying monitoring and management.

Now let's show the same attack scenario for the modified ERD, i.e., we will make queries to the updated database. We will use the same roles as before the ERD change.

When we try to assign all the roles to the user, we get an error (Figure 6).

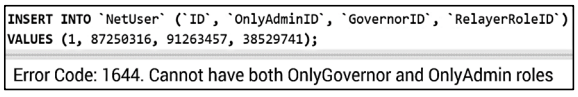

Figure 6: Trigger for unauthorised role assignment.

When we try to assign OnlyAdmin and another role to the user, we also get an error (Figure 7).

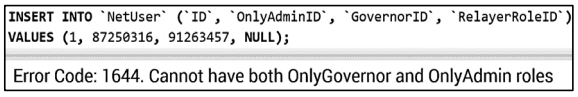

Figure 7: Trigger for unauthorised role assignment.

Therefore, it can be argued that the vulnerability that allowed hackers to take over the network can be eliminated by changing the role policy of the Ronin protocol.

4 CONCLUSIONS

The relevance of identifying and eliminating Ronin protocol vulnerabilities is caused by high-profile cyber incidents with thefts of hundreds of millions of dollars. A detailed analysis shows that the Ronin protocol has numerous vulnerabilities that lead to such incidents. This is especially true for the vulnerabilities related to the consensus mechanism, in which validators were given unjustifiably broad rights, which led to the vulnerability. This vulnerability allowed hackers to take over the network, resulting in multimillion-dollar losses for the network participants. The vulnerability was rooted in a flawed role-based rights policy. A simulation model of the Ronin protocol was developed, which showed the possibilities of illegal penetration under the existing role-based policy, which can be qualified as a Flat Model of the NIST RBAC standard. It was shown that changing the role policy to the level of the Restricted Model of the NIST RBAC standard eliminates the consensus vulnerability, which is currently the main source of user losses. The obtained result shows that the applied methodology gives the prospect of continuing research to combat other attack vectors on the Ronin network.

REFERENCES

[1] M. S. Mavis, "Official Ronin Whitepaper: Consensus," Apr. 28, 2023.

[2] M. S. Mavis, "Official Axie Infinity Whitepaper," Jan. 1, 2023.

[3] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact," Future Generation Computer Systems, vol. 102, pp. 905-913, Aug. 2019.

[4] E. Castronova et al., "As real as real? Macroeconomic behavior in a large-scale virtual world," New Media & Society, vol. 11, no. 5, pp. 685-707, 2009.

[5] R. Behnke, "Explained: The Ronin Hack," Mar. 30, 2022.

[6] V. B. Vishal and A. B. Aniruddha, "Preferential Delegated Proof of Stake (PDPoS) – Modified DPoS with Two Layers towards Scalability and Higher TPS," 2023.

[7] S. Wan et al., "Recent advances in consensus protocols for blockchain: A survey," Springer Science+Business Media, LLC.

[8] M. Alharby et al., "Blockchain-based smart contracts: A systematic mapping study of academic research," in 2018 ICCBB, IEEE, pp. 1–6, 2018.

[9] S. N. Khan et al., "Blockchain smart contracts: Applications, challenges, and future trends," Peer-to-Peer Netw. Appl., vol. 14, pp. 2901–2925, 2021.

[10] Verichains Lab, "Report for Sky Mavis: Security Audit – Ronin Bridge Smart Contracts. 1.1 - Public Report," Jun. 28, 2022.

[11] X. Li et al., "A survey on the security of blockchain systems," Future Generation Computer Systems, pp. 841–853, 2020.

[12] J. Lyanchev, "The Biggest Ever Crypto Hack: What Happened in the Ronin Bridge Attack on 'cryptopotato'," Mar. 30, 2022.

[13] D. Ferraiolo and R. Kuhn, "Role-Based Access Controls: Conference 15th National Computer Security Conference (NCSC)," Oct. 13-16, 1992.