Venn: Decentralized Security Infrastructure Litepaper

We introduce the Venn Security Network, a unified security infrastructure composed of customizable node operators, security teams, protocols, chains, rollups, and stakers that is capable of stopping malicious transactions and protecting against economic risks. By leveraging a unique symbiosis of off-chain mechanisms with on-chain security, the network delivers a unified and modular defense [1] layer against the evolving threats facing blockchain applications and protocols [2]. The Venn Network employs a distributed framework, enabling protocols to define the participants that must collaborate to safeguard digital assets, communities, and networks through a shared security layer. The necessity for such a platform stems from the increasing sophistication of threats [3] coupled with the limitations of traditional security measures [4], which often operate in silos, lacking the adaptability and collective intelligence required to counteract novel vulnerabilities effectively. This litepaper outlines our vision and explores the core design mechanisms, employing the wisdom of the crowds game theory to the system architecture, and the roadmap towards creating a safer, more reliable blockchain landscape for all.

I. INTRODUCTION

The security paradigms in web3 face unprecedented challenges [5], underscored by the sophistication of threats such as social engineering attacks, smart contract vulnerabilities, and zero-day exploits. These vulnerabilities expose the limitations of traditional security measures such as in-depth smart contract audits [6], monitoring algorithms [7], formal verification [8], static analysis [9], circuit breakers, and pausing methods. All these existing measures often fall short in the dynamic, fast-moving world of blockchain. As Web3 technologies empower users with greater control over their data and digital assets, this reality calls for a new approach to security, one that is permissionless, community-driven, modular, and capable of fostering defense against constantly evolving threats

Our goal is to construct an ecosystem that is designed to proactively eliminate malicious activities before they can affect the blockchain's state, without compromising user experience or the operational continuity of applications. The strategy leverages seamless integration between off-chain and on-chain security mechanisms, resulting in a security layer that is scalable, customizable, and diverse.

II. BACKGROUND

Our journey began back in the early days of smart contracts and the innovative work at Bancor (the first AMM [10] and Decentralized exchange), when our ICO momentarily challenged Ethereum's scalability and security, highlighting the blockchain's vast potential alongside its challenges.

As the landscape transitioned from 2018 towards 2024, a transition from the rigidity of traditional smart contracts to the present's more malleable, upgradeable architectures signified a critical paradigm shift, establishing new benchmarks for security and operational flexibility in blockchain technologies.

The advent of Ethereum's Proof of Stake and EigenLayer's restaking innovations necessitated a departure from monolithic mechanisms towards shared security, unified, and modular layers. This evolution underlines our belief that security should be a layer that can be upgraded and improved constantly on-chain, and the balance inside the blockchain trilemma can be optimally equilibrated, allowing builders to tailor risk management strategies to their specific needs against the evolving threats.

This evolution from the early stages, where security was often an afterthought, to a phase where audits became mandatory, and subsequently to a demand for monitoring tools, underscores the market's growing awareness and the necessity for proactive security measures. While the adoption of monitoring has increased, it remains insufficient without the capability to actively stop threats at the execution phase [11]. This is a strategic move towards incorporating on-chain solutions as key elements of a security framework, aimed at proactively managing threats during the execution phase and beyond.

III. THE PROBLEM: STAGNATION IN BLOCKCHAIN SECURITY

We are witnessing a paradox that underscores a critical vulnerability in web3's current security infrastructure. Despite the proliferation of advanced tools, solutions, and researchers, the frequency and sophistication [12] of malicious activity continues to grow. The crux of this issue lies in the evolving threat landscape, and in the fragmented [13] approach to security. Current solutions and best practices are disjointed, making it difficult to adopt effective defenses. The absence of standardized [14] protocols exacerbates this challenge, leaving the ecosystem in a constant state of vulnerability.

Current security measures, including monitoring services, smart contract audits, formal verification, and static analysis, offer critical safeguards individually but lack the interconnectedness essential for a holistic defense. While each method contributes valuable insights or protection, their isolated application limits their collective potential. The challenge lies not in their individual efficacy but in the lack of a cohesive strategy that integrates their strengths in a decentralized manner, avoiding reliance on a single vendor which often results in less resilience, and highly centralized architecture.

A second security problem arises from the inherent limitations of blockchain technology, on-chain mechanisms alone are inadequate for complete transaction evaluation as they lack broader context and necessary data on the transactions. This constraint, coupled with the deterministic nature of blockchain, limits the full effectiveness of on-chain security measures as standalone solutions. Conversely, off-chain mechanisms, while offering more extensive evaluation capabilities, do not affect the transaction during the execution phase [11]. Additionally, attackers can route transactions to any node for on-chain execution, resulting in a lack of enforceable security measures during the transaction's execution.

A third security problem is that current security measures aim to protect assets before or after a transaction's execution, missing the window to prevent damage. This is exacerbated by atomic and private transactions, executed in a single block without prior visibility. This underscores a significant gap in protection during runtime, specifically as transactions are being processed by the EVM.

As a representative example of the security gap in the transaction lifecycle, consider the Curve exploit [15, 16]. This hack (more than $60 million were exploited) was caused by a bug in the compiler of the smart contracts (Vyper, a Python-based EVM smart contract language). Despite the smart contracts undergoing audits, this bug was impossible for auditors to catch. Pre-deployment security measures were taken to release it securely, and monitoring services were also implemented, although not directly by the Curve team (MEV whitehats tried to assist the team). Despite these efforts, a malicious transaction found its way into the protocol and was executed more than once, despite the attempts to front-run it and execute an incident response. The window between the time the transaction is sent (publicly or privately [15, 16]) and when it changes the state atomically, remaining as a fact on the blockchain, is still open. Addressing these problems requires a shift in how we perceive and implement security within the blockchain ecosystem, encompassing all phases of transaction processing. An approach that leverages the fusion of on-chain and off-chain architectures, decentralized trust, and unified security.

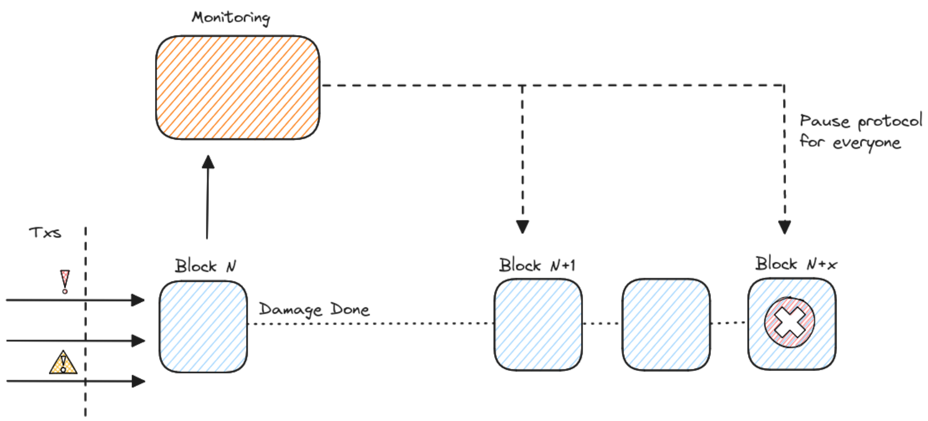


FIG. 1. Monitoring Flow Diagram

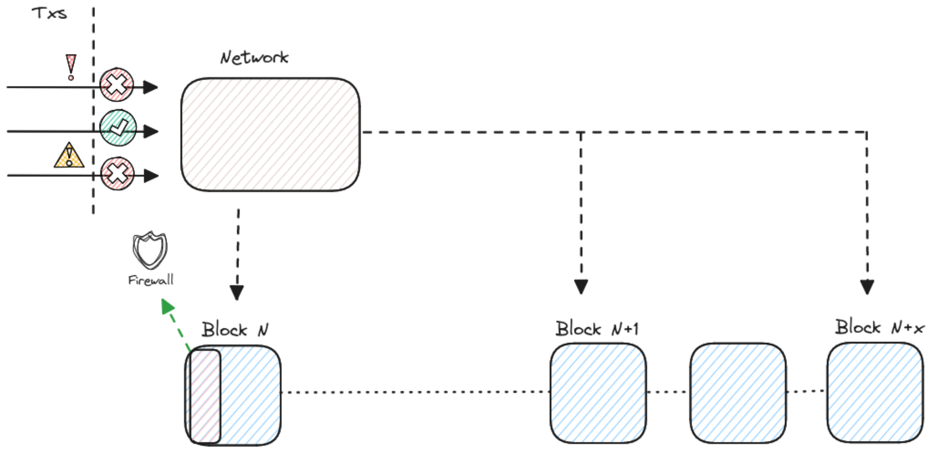


FIG. 2. Venn Network Flow Diagram

IV. THE SOLUTION: BUILDING A SECURITY NETWORK

By addressing the critical execution [17] phase where current frameworks fall short. Venn [18] is a fusion of on-chain and off-chain mechanisms [19] - an innovative modular security infrastructure seamlessly interconnected with an on-chain firewall through a set of security operators [20]. This structure establishes a robust defense against malicious activities by enabling on-chain blocking mechanisms tailored to each application's unique requirements, empowered by the flexibility of off-chain transaction validation.

This integration fosters a collaborative framework, enhancing the efficacy of each security layer. It ensures not just preemptive threat mitigation but also a permissionless, adaptable framework that empowers protocols to navigate risk according to their perceptions and criteria based on perceived threats. By integrating these components with the active participation of diverse ecosystem actors: from Networks and Protocols to Developers, Operators, Security Teams, and Researchers, we are laying the groundwork for a new security paradigm, while leveraging Ethereum's PoS and EigenLayer [21] proposed restaking model.

At the core of our approach to securing the Web3 ecosystem are two operational elements and a set of guiding principles for how they should function:

A. On-chain Firewall

This layer consists of a set of smart contracts [22] that any web3 application can implement to enable a precise on-chain blocking mechanism, designed to halt malicious activities without disrupting the protocol's functionality [23]. The Firewall hosts a wide array of security policies [24], and each policy validates specific logic within every function, action, transaction, or pattern that enters the application. Policies can be updated or configured at any moment enabling applications to stay ahead of security threats, while also enabling flexibility to create custom policies, allowing for a security layer that is highly adaptive to specific needs enabling pre and post-execution validations throughout the execution life-cycle of the transaction. Because of the decoupling between the core protocol and firewall smart contracts, the firewall can be upgraded more cavalierly than the core protocol logic which underpins the security of user funds. The On-chain Firewall is fully operational and actively deployed on-chain [25]. Detailed implementation and source code can be accessed via our GitHub repository [26].

B. Venn Security Infrastructure

The Venn Security Infrastructure comprises a set of security operators [27] who collaboratively examine and reach a consensus on the intention of transactions submitted by users for verification before they are committed to the blockchain. Each transaction undergoes diverse and thorough security checks to ensure it is not malicious. The network thus created operates on a decentralized architecture [28], making security assessments permissionless and accessible across various blockchain environments, including L1's, L2's, and Appchains. This decentralized and flexible approach ensures that the security protocols remain agile and scalable, ready to adapt to the evolving landscape of Web3. At the firewall level, the Venn solution behaves as a dedicated policy known as the Security Oracle Policy, enhancing the Firewall's capabilities and extending its reach. The Venn

Network offers versatile implementation options, these include direct connections to the firewall for proactive threat prevention, while also enabling integrations with smart wallets, rollups, appchains, and protocols for insights and alerting.

C. Ecosystem Principles

These following principles are not just aspirational, they are operational mandates that ensure the technology aligns with the Web3 ethos while effectively addressing the ecosystem's security challenges.

1. **Proactive:** Not just to mitigate but to actively stop hacks, entirely grounding our platform in proactive security measures.

2. **Adherence to Web3 [29]:** Committed to preserving decentralization, ensuring that the platform enhances security without compromising the principles of autonomy and distributed authority inherent to Web3.

3. **Diversity in Security Approaches:** Designed to be universally accessible, allowing a wide range of participants to contribute their unique methods. This diversity in approaches is essential to comprehensively address all potential security threats.

4. **Simplicity in Implementation:** Emphasizing the importance of ease of implementation, providing solutions that are straightforward to integrate and operate, and reducing the entry barrier for protocol adoption.

5. **Reliability:** Designed to mirror the blockchain's resilience, operating with the same 24/7 availability to provide continuous protection.

6. **Independence and Robustness [30]:** Built with redundancy in mind, ensuring no single point of failure can compromise the network, upholding the independence and robustness essential for security platforms.

7. **Modularity:** Flexibility is key, the architecture is inherently modular, allowing components to be added, removed, or replaced with ease. This modularity invites collaboration and enables customization to meet diverse needs.

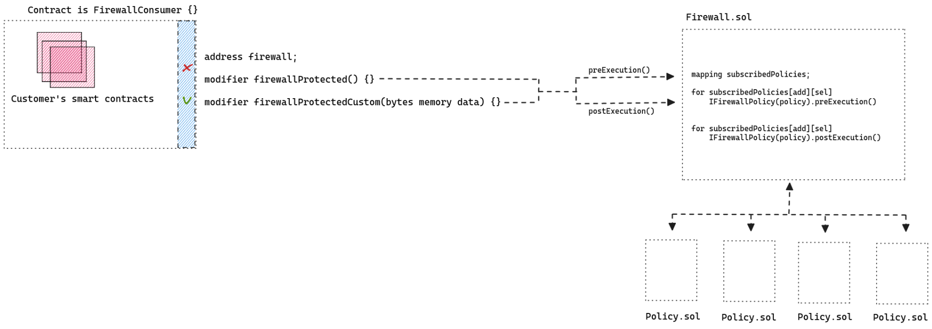
8. **User Experience:** Aiming to maintain a frictionless user experience, recognizing that security should enhance, not hinder, user interactions.

9. **Affordability and Incentivization:** Balancing cost-effectiveness with adequate incentives for security teams is crucial, ensuring the platform remains economically viable while rewarding those who contribute to its hardening.

10. **Accuracy and Precision:** As a system with the ability to halt transactions, we strive for the highest levels of accuracy, minimizing false positives and negatives to maintain trust and efficacy.

V. ON-CHAIN FIREWALL

The On-chain Firewall is an abstract smart contracts framework that enables a web3 application owner to define the validations on each call to his contracts before and after the call execution. If the validations are not approved, the transaction is reverted.



```
address firewall;
modifier firewallProtected() {}
modifier firewallProtectedCustom(bytes memory data) {}
```

```
mapping subscribedPolicies;

for subscribedPolicies[add][sel]
    IFirewallPolicy(policy).preExecution()

for subscribedPolicies[add][sel]
    IFirewallPolicy(policy).postExecution()
```

FIG. 3. Technical Firewall Diagram

1. *Concepts*

- **Firewall consumer** – the protected smart contract that interacts with the Firewall Contracts through its modifiers, the application's smart contract inherits this contract for this functionality.

- **Firewall** – a smart contract which holds a mapping of addresses and sighashes (which represents of protected functions of the Firewall consumers) to the relevant policies.

- **Policy** – A smart contract that implements logic to validate the protected function call, the policy implements the functions "preExecution" and "postExecution", preExecution runs logic before the function execution, postExecution runs logic after the function execution, if either of the function executions is not satisfied then the policy will revert the entire transaction.

- **Modifiers [31]** – A solidity syntax concept that implements generic code as part of the function call, usually used for validation of states and call data.

- **Secured payload** – A function call that is added to a transaction with call data that includes a hash of all the approved calls of that specific transaction, the approved call hash includes the order of the function calls along with its values and the relevant transaction context.

- **Execution phase** – a phase in the transaction lifecycle that changes the state in the blockchain, transaction lifecycle [11] in EVMs is - creation, signing, sending, verification, broadcasting, execution, and finalization.

- **Separation of concerns [32]** – Ensures that each system component performs a distinct function. By maintaining clear boundaries between components, it simplifies system maintenance and upgrades. The firewall enhances this architecture by distinctly separating security functions from business logic

- **On-chain limitations** – In regard to security measures and transaction validation the smart contract has three critical limitations:

  1. Data [33] - when a call enters the smart contract the smart contract has only the tx.context [34], block.context, state, and calldata to use as inputs for the validation, missing the full trace of the transaction or even historical data that can't be updated in real-time on the blockchain.

  2. Compute - the resources to run heavy lifting modules in terms of computation are limited because of the cost of the transaction and the EVM limitations.

  3. Visibility - although this feature is very important for the regular Joe in crypto and has a lot of benefits, it also carries some security disadvantages, especially if all the security measures are on-chain. The attacker can run all the tests he needs locally and the data is immutable. This gives the attacker an advantage which makes the risk to an application larger. Often, incidents occur as a surprise and with great damage.

- **Revert** – one thing that can happen only on-chain is to revert or fail a transaction.

- **Dry run** – A mode of system operation without the reverting mechanism. Useful for system evaluation.

2. Firewall Call flow

1. A dApp user initiates a transaction, sending it to the application. The transaction includes the call data from the security operators and an additional payload.

2. Before execution, the transaction is captured by the Firewall. This pre-execution check uses function modifiers to ensure transactions align with security policies.

3. The Firewall examines the incoming transaction, applying predefined security policies to validate its legitimacy and intent. If it adheres to the security criteria, the transaction is allowed to proceed.

4. Post-validation by the Firewall, the transaction is executed by the application as per normal operational flow.

5. After execution, the Firewall conducts a post-execution review to ensure the transaction has not altered the protocol's state in a manner that breaches the established business logic.

6. If any state changes contravene the security policies, the transaction is reverted.

7. Transactions that pass this final check are confirmed and their effects are finalized on the blockchain.

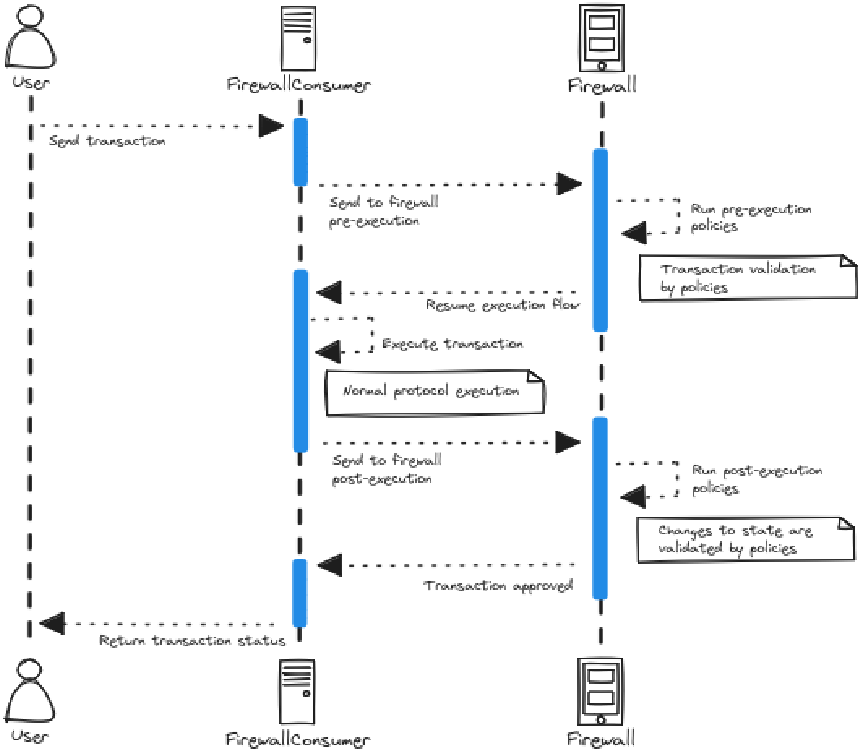

User
FirewallConsumer
Firewall
Send transaction
Send to firewall pre-execution
Run pre-execution policies
Transaction validation by policies
Resume execution flow
Execute transaction
Normal protocol execution
Send to firewall post-execution
Run post-execution policies
Changes to state are validated by policies
Transaction approved
Return transaction status
User
FirewallConsumer
Firewall

FIG. 4. Firewall Call flow

3. Added Gas cost

Every transaction needs to pay some fee for the computation and the data it's using from the network that operates the chain (in most of the chains), in some of the chains it can be significant, and in some it will be insignificant because we have components that sit on-chain there is some additional gas [33] to the transaction execution, there is two main parameters that affect the percentage of the added gas - how simple/sophisticated the transaction is (more sophisticated means lower percentage), how many policies and what policies are involved. The change will be between 5% and 25%.

4. Smart-Contract Modules

The On-chain Firewall is a combination of these main components (smart contracts):

1. **FirewallConsumer.sol** - the developer inherits from that contract to be able to add the necessary modifiers to protect his functions and connect to the Firewall.

2. **Firewall.sol** - this is a smart contract that holds the mapping of the contracts/functions to the relevant policies and runs the policies for each function call with the modifiers.

3. Optional Proxies

   - **FirewallTransparentUpgradeableProxy.sol** - replacing the normal TransparentUpgradeableProxy.sol of Openzeppelin [35], with the preExecution and postExecution hooks of the firewall for every external call to its implementation contract.

   - **FirewallProxyIntercept.sol** - This proxy can be deployed between the original proxy and the implementation contract, this way the only change the protocol needs to do is to change the address of the implementation contract in the original proxy to be the intercept and to add to implementation contract address to the intercept proxy.

   - **FirewallProxyAdmin.sol** - Is a derivative of the ProxyAdmin of OpenZeppelin for the firewall proxies.

4. **IFirewallPolicy.sol** - This interface is for creating policies that the firewall will use.

5. Built-in security policies

   - **ApprovedCallsPolicy.sol** - AKA Security Oracle, this policy is designed for the Security Network, the policy gets a list of approved calls (with the approved values), and then checks if each call enters the protected function approved, the FirewallConsumer.sol includes a function that enables these two steps of approving and checking to be in a single atomic transaction, the ones to approve the transactions calls are the Operators in the Security Network. [36]

   - **ApprovedVectorsPolicy.sol** - AKA Patterns Check, this policy is made to work entirely on-chain, this policy checks if the call is part of known calls patterns that are ok to be executed in the protocol, if it is not part of a know and legitimate patterns of calls then the transaction will be reverted, off-chain mechanisms including fuzzing, formal verification and machine learning build the patterns of calls and save it in the policy as part of the configuration of the policy.

   - **CombinedPoliciesPolicy.sol** - this policy is built to be used as an infrastructure module, enabling the application owner to create logical combinations of policies. It enables the creation of OR between different policies, instead of the built-in AND that is inside the Firewall.sol.

- **OnlyEOAPolicy.sol** - this policy is used to enforce the user of the protocol to be an EOA, the vast majority of the hacks (besides private keys compromised) are done through smart contracts as the caller to the protocol, this policy can be combined with the CombinedPoliciesPolicy.sol and another policy this way it will require only transactions through smart contracts to be validated by the other policy.
- **ForbiddenMethodsPolicy.sol** - this policy is designed as an infrastructure policy to enforce the logic of the firewall on specific flow (using CombinedPoliciesPolicy.sol) with it, by adding it to another policy it will revert specific flow (that starts from a specific function) unless the other policy/policies is approving that flow, that policy can be used as a standalone policy to disable the use of specific functions.
- **AllowlistPolicy.sol & BlocklistPolicy.sol** - these policies block or allow transactions according to their mapping of senders.
- **BalanceChangePolicy.sol** - this policy checks if the transaction is withdrawing or transferring an amount of token/native that exceeds the maximum threshold configured in the policy for the specific protected address in the specific token.
- **AdminCallPolicy.sol** - this policy works as a notary, signer can approve specific calls, it was designed to be used as a notary for admin functions, which can also be used as multi-factor for privileged actions.

5. Governance

The governance of the firewall contracts is very flexible and can be configured as the protocol desires, from completely centralized to completely decentralized. When the ownership/admin is of the protocol then it means that the protocol can control the contract with its EOA/Multisig [37] /MPC [38] /Timelock [39] /DAO [40].

The main admin roles are:

- **Policy Admin** - controls what contract can call the policy (executors), the executors are the Firewall contract or the CombinedPoliciesPolicy contract. This admin also controls who are the smart contracts that are protected by that policy (consumers). Any configuration will be made through that admin. The protocol will be the admin of the policies / delegate that role.

- **Firewall Owner** - controls the approval of the policies, the other functionality, most of it to subscribe to policies and to set Dry-Run mode, is controlled by the Firewall Admin of the consumer (protected smart contract). The firewall owner can be the protocol, Ironblocks team, or any other entity, as said that role is for approving policies to be able to be used.

- **Firewall Admin** - that role is configured inside the protocol's smart contracts (consumers), and that role controls the subscription in the firewall, this role also sets the firewall and the Security Oracle addresses. The protocol will be the admin of the policies / delegate that role.

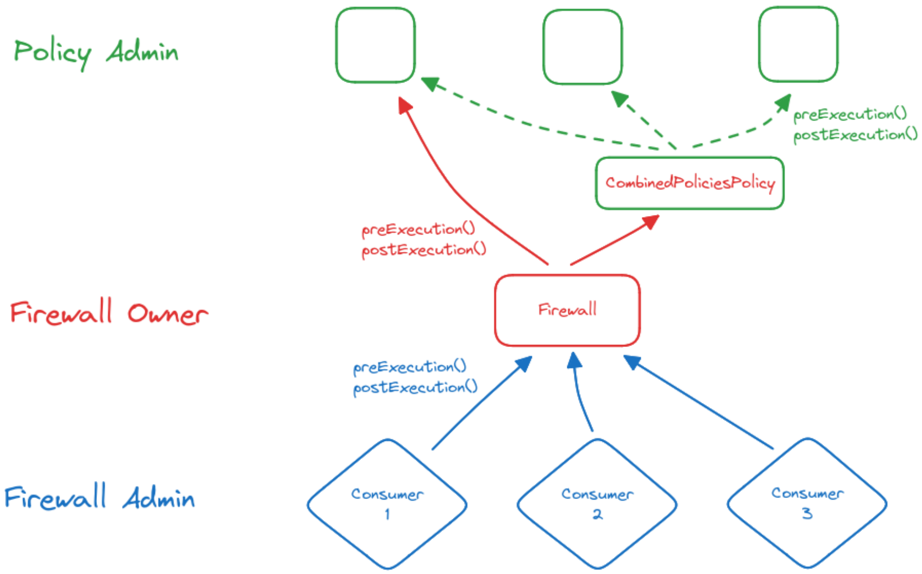


FIG. 5. Smart Contracts Admin Roles

VI. VENN SECURITY INFRASTRUCTURE

A. Definition of Ecosystem Participants

- **dApp Users**: interact with protocols through transactions.

- **Protocols**: deploy and manage smart contracts, utilizing the on-chain Firewall for an added layer of security to protect their digital assets and communities.

- **Operators** [27]: Entities responsible for running services within the Security Network, validating transactions against security validation algorithms, and attesting for an invalid state or exploited transaction.

- **Security Operators**: develop and provide the essential logic and algorithms for scrutinizing transactions, contributing to the network's collective defense capabilities.

- **Security Council** [41]: A select group of security experts tasked with resolving disputes and making determinations on security-related matters within the network.

- **Restakers and Stakers**: Participants who support the network's integrity and reliability through staking, contributing to overall cryptoeconomic security [42].

B. Security Operator Framework

The Venn [18] Security Infrastructure consists of a set of linked Operators running the security service (called Blockbeat), the service works as an extension of the blockchain node, the services are connected through p2p [43], and each service can connect to a number of blockchains via its configuration when an operator gets a transaction the service will examine the transaction and will send it to random participants of operators to examine that transaction too, each one of the operators will sign its answer and consensus will be made according to their response if the transaction is legitimate or not. As of the time of writing, the Blockbeat will only support EVM [44] chains.

1. Concepts

- **Consensus:** A mechanism ensuring decisions and validations within the network reflects a majority agreement among operators.

- **Incentives:** Rewards designed to motivate network participants to contribute positively, ensuring active and beneficial engagement that increases network security and efficiency.

- **Fees:** A financial structure implemented to compensate for services rendered within the network, balancing the cost of operations with the value provided to developers and protocols.

- **Slashing:** A punitive measure applied to participants who act maliciously or fail to meet their obligations, safeguarding the network's integrity by discouraging harmful behavior.

- **Economic Security:** Leverages EigenLayer's restaking concept to enhance the network's economic security, providing a layer of financial assurance and stability through distributed risk and reward mechanisms.

- **Privacy:** Ensures the confidentiality and security of transactions and data within the network, implementing measures to protect against unauthorized access and exposure.

- **Interoperability:** Facilitates seamless interaction and integration between various EVM networks and protocols, ensuring the network's tools and services can be broadly applied across the Web3 ecosystem.

- **Detectors** - logic that inspects transactions for a specific behavior, the detector is getting as input the transaction trace with some additional context like the involved assets/protocols, the logs of the transaction, and the state diffs. The detector will return a boolean flag or scoring.

- **Blockbeat - the Operator service name**, it is written in Golang and can connect to components written in other languages, and communicate with other Blockbeats via peer-to-peer communication.

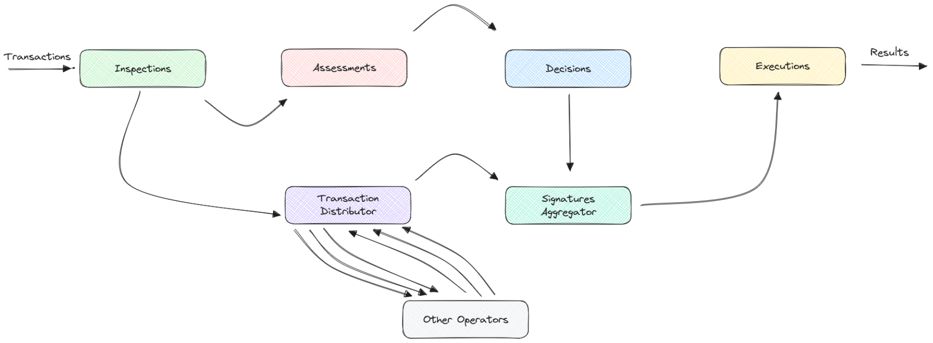


FIG. 6. Transaction journey inside the operator

C. Operator Modes

The Venn Security Infrastructure can be operated in a variety of modes to guarantee maximal tailoring for protocols needs:

1. Venn Network mode

The operator connects to an open ecosystem of Operators participating in Venn's public security network, thus participating in the establishment of open consensus. Operators in this mode can choose which, and how many, chains they want to connect to, as well as the number of security networks to participate in.

2. Standalone mode

Operator serves customers that connect only to their API/RPC instead of requiring a distributed consensus mechanism. This is beneficial for security operators that have their own unique, protocol-specific security mechanism underpinning their business model.

3. Subnet mode

The operator connects to other Operators but not to the Venn network, thus creating a new consensus sub-layer, separated from the wider security network. This is beneficial if some form of consensus is required for an operator's unique, protocol-specific security mechanism underpinning their business model.

D. Transaction flow

1. dApp users send transaction data through an SDK, submitting transaction data similarly to an 'estimate_gas' request.

2. The transaction gets to an Operator which propagates the transaction to random operators that participate in examining the transaction with him.

3. Each Operator uses detectors to examine the transaction to decide if the transaction is malicious.

4. Each Operator's signed response is aggregated with those from other Operators.

5. According to the signatures, a consensus is obtained on the degree of maliciousness of the transaction data. This result is added to the transaction data via an augmented payload that authorizes the user's calls to the protected protocol.

6. The user signs the transaction with the added payload and submits it to the blockchain.

7. The transaction execution activates the On-chain Firewall which extracts the approved payload and verifies it against the dedicated Security Oracle policy.

8. Provided the transaction's calls match those vetted and authorized by the Security Oracle, the transaction is processed as normal. Otherwise, the Firewall triggers the transaction to revert.

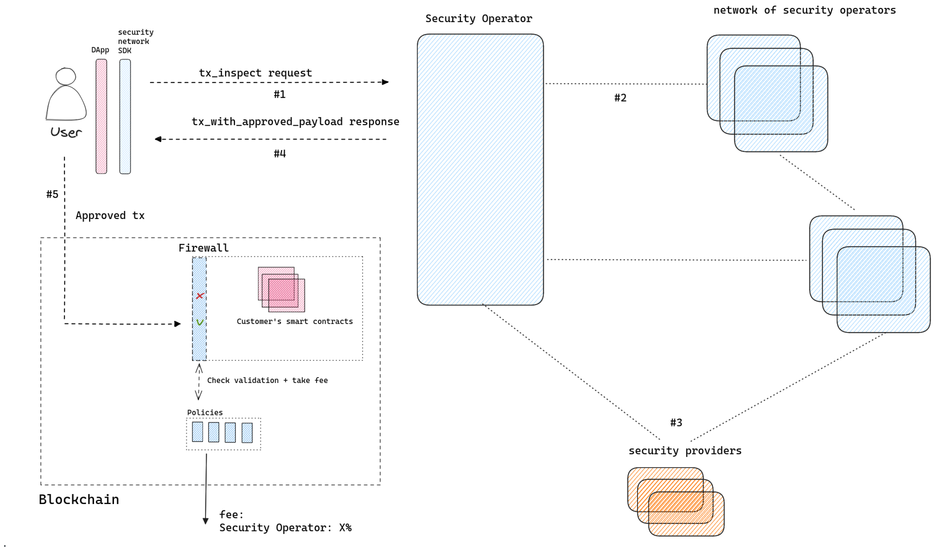


FIG. 7. Security Network + On-chain Firewall

E. Built-in detectors inside the Operators

Our Operator comes with built-in detectors, the operator can choose to use them or to turn them off. The Operator can connect his logic to the Blockbeat with our Detector-SDK, or to use other external detectors created by other security teams or whitehats. The Operator can choose the combination logic of the detectors and their configuration. The built-in detectors are:

1. Flashloan [45]

2. Reentrancy [46]

3. Suspicious smart contract sender [47]

4. Tx.origin funded by mixer [48]

5. Drastic balance change

6. Admin calls

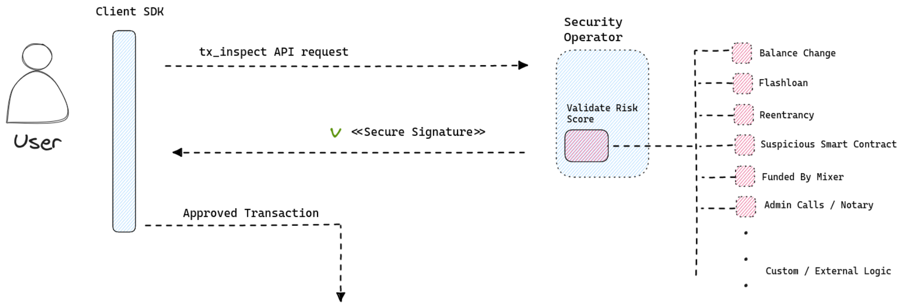


FIG. 8. Single Operator in the Security Network

The Operator can work in standalone mode and not connect to the network, it can enable the protocol to decide its configuration and add detectors to check invariants specified for the protocol it protects. The Blockbeat can run on any machine and connect to any EVM RPC node.

F. The Venn Network

1. Governance

The Venn Network's governance structure will transition towards full decentralization in several phases, ensuring the process is fully transparent and secure for all network participants. The constitution establishes these phases and tiers of governance, progressively moving towards complete decentralization to achieve on-chain credible neutrality. Initially, a security council will be established to monitor, safeguard, and maintain the network, ensuring a smooth transition to full decentralization. Then, the three layers of governance will be introduced. Each tier requires a different amount of voting power and allows for participation as a Proposer or a Voter based on criteria defined in the foundation constitution. When the required quorum is reached during the voting period, the community can then achieve the necessary majority or supermajority needed for the proposal. The proposals will cover community allocations, grants, hires, operations, versions, fees, transaction legitimacy, disputes, and slashing.

2. Consensus mechanism

The consensus mechanism in the security network will be used to determine in real-time if a transaction is legitimate or malicious, TTL will be enforced for the security operators to respond, and this TTL will be reduced over time.

The voting power for each operator will be determined by the staked, and restaked amount delegated to his address. Every predefined time - 7 days - the top 100 operators will be chosen as the security group to examine the transactions and return their responses. Consensus is reached, when more than 66% of the voting power agrees says that the transaction is legitimate. The checks for the voting power will take place on-chain, with on-chain shared security platforms (e.g. EigenLayer).

Intersubjective security resolutions - Security disputes and resolutions are, as Eigenlayer frames it, retroactive intersubjective faults. These are decisions that the blockchain can't make on its own, but that all active observers of the network can reach a broad consensus easily. This makes coming to a consensus retroactively on whether a false positive or negative infraction occurred possible and straightforward as long as there is an honest majority of voting nodes.

Example of the shared-security and the CoC (cost of corruption) in the security network with an example of a threshold of 66% for consensus and slashing for a malicious actor of 20%

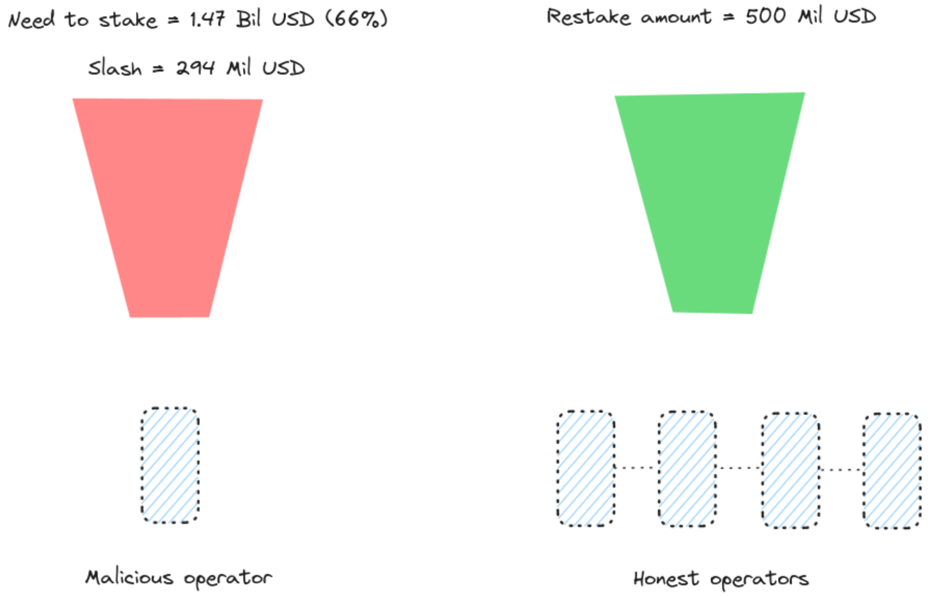

Need to stake = 1.47 Bil USD (66%)
Slash = 294 Mil USD
Restake amount = 500 Mil USD
Malicious operator
Honest operators

FIG. 9. Cost of Corrupting the Network

G. Economics

The economic model is designed to ensure the network's integrity and efficiency, providing a balanced system of incentives and responsibilities for all participants, while requiring operators to stake and receive staking delegation. By aligning the roles and rewards of Operators, Security Operators, Stakers, and other crucial actors, we establish a robust framework for network sustainability and effectiveness. Additionally, a clear and flexible fee structure supports Protocol Owners and Users, promoting a permissionless environment. This economic approach harmonizes stakeholder interplay, fostering a self-sustaining and resilient security landscape.

The security network strives to incentivize those actors that build the security and act as guardians for the entire Web3 community. These are:

1. Operators: Operate the network blockbeat nodes, validating transactions and ensuring they meet security logic before processing them.

2. Security Operators: Develops and updates the security algorithms and policies that are implemented in the different operators.

3. Builders and Protocols: Building on top of the network, in order to safeguard their applications and communities.

4. The stakers and restakers: Provide economic security to the network by staking and restaking.

5. External Security Committee: Expert oversight on security disputes and decisions, guiding the network towards best practices and resolving security challenges.

The fee payers inside the network are:

1. The protocol owners (as paymasters)

2. The users of the protocols

The fee can be paid by the protocols as paymaster or by the users as part of the fee in the transaction. The fee will be defined by mutual voting of the operators and the protocols by proposals to the DAO in the permissionless stage. Custom fee mechanisms can be implemented in the 'standalone' and 'subnet' mode of the operators.

When the network becomes permissionless, a slash/reward mechanism will be enacted according to the following general criteria:

1. Rewards

Operators who vote honestly against malicious transactions receive regular rewards when part of the voting majority and boosted rewards when part of the voting minority. Examining a transaction (uptime). Priority fee for bringing the transaction to the network.

2. *Slashing/Inactivity Leak*

Slashing will be enacted as a disincentive for unaligned operator behavior. The Venn architecture will impose slashing costs on operators when the following actions occur:

- Operators who vote in favor of malicious transactions (false negatives).
- Operators who vote against a legitimate transaction (false positives)
- Inavailability of operators when the need to examine transactions occurs (downtime)

The locking time for stakers/restakers will be 7 days, which leaves time for the security committee to decide if a specific transaction is a hack or not by a dispute that will be open for the relevant transaction.

3. *Community and Ecosystem*

At the core of the Security Network is a belief in the power of a diverse and active community that supports the security ecosystem. The builders' community is essential, encouraging participants to create, share, and contribute to a more secure Web3 world.

Builders are empowered by developing tools, protocols, and frameworks that invite global collaboration and innovation. This openness not only accelerates the advancement of security solutions but also ensures transparency and adaptability.

Central to the ecosystem is the Proof of Contribution (PoC) mechanism, which rewards community members for their active participation and contributions, moving beyond traditional engagement metrics to reward substantive contributions that enhance network security.

Our vision is a security network driven by a community that is empowered and incentivized through collaborative efforts and open-source development, paving the way for a secure and resilient blockchain landscape.

VII. ROADMAP

The roadmap unfolds a new chapter where cutting-edge security features, innovative economic models, and collaborative development converge to address the complex needs of Web3's evolving security landscape, marked by strategic milestones aimed at refining the network's infrastructure, enhancing user experience, and fostering a vibrant community of builders and contributors.

Chapter 1: Genesis Block-Beat: Development of the core infrastructure, focusing on the architecture design of the nodes - the network's backbone.

Chapter 2: Assembly: onboard operators, security teams, and protocols. Deployment on a Testnet environment while assembling the initial committee members to oversee and guide the ecosystem's development.

Chapter 3: Beta: Release of economic models alongside rewarding mechanisms to encourage network participation and contribution. Introduction of conflict detection and resolution methods. Enhancement of privacy features.

Chapter 4: Launch: Finalizing and deploying governance models, signaling the transition to a foundation framework. Release of the grants program to support ecosystem expansion. Culminates in the full launch on the Mainnet.

Future Innovation: While the initial focus is firmly rooted in security, the Venn Network paves the way for a multitude of future innovations and implementation possibilities. By leveraging its diverse and robust design, the network is designed to support a wide array of applications built on top of it, driving forward the evolution of the blockchain ecosystem. Here are some of the key areas of innovation that the Venn Network will enable: Smart Wallets and Account Abstraction (AA), Externally Owned Accounts (EOA), On-chain Compliance, On-chain Insurance, On-chain Cryptoeconomics, Oracles, and Decentralized Sequencers.

[1] Eric Jollès, Sébastien Gillard, Dimitri Percia David, Martin Strohmeier, and Alain Mermoud. Building collaborative cybersecurity for critical infrastructure protection: Empirical evidence of collective intelligence information sharing dynamics on threatfox. In *International Conference on Critical Information Infrastructures Security*, pages 140–157. Springer, 2022.

[2] Chainalysis Team. Funds stolen from crypto platforms fall more than 50`https://www.chainalysis.com/blog/crypto-hacking-stolen-funds-2024/`, 2024. Accessed: 2024-01-24.

[3] Jieren Cheng, Luyi Xie, Xiangyan Tang, Naixue Xiong, and Boyi Liu. A survey of security threats and defense on blockchain. *Multimedia Tools and Applications*, 80:30623–30652, 2021.

[4] Milan Kumar Dholey and Ananya Ganguly. Major challenges and threats of blockchain technology. In *International Symposium on Artificial Intelligence*, pages 96–108. Springer, 2022.

[5] Chuan Chen, Lei Zhang, Yihao Li, Tianchi Liao, Siran Zhao, Zibin Zheng, Huawei Huang, and Jiajing Wu. When digital economy meets web3.0: Applications and challenges. *IEEE Open Journal of the Computer Society*, 3:233–245, 2022.

[6] Isaac David, Liyi Zhou, Kaihua Qin, Dawn Song, Lorenzo Cavallaro, and Arthur Gervais. Do you still need a manual smart contract audit? *arXiv preprint arXiv:2306.12338*, 2023.

[7] Forta Network. The past, present, and future of monitoring. https://forta.org/blog/the-past-present-and-future-of-monitoring/, 2022. Accessed: 2022-11-23.

[8] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Anitha Gollamudi, Georges Gonthier, Nadim Kobeissi, Natalia Kulatova, Aseem Rastogi, Thomas Sibut-Pinote, Nikhil Swamy, et al. Formal verification of smart contracts: Short paper. In *Proceedings of the 2016 ACM workshop on programming languages and analysis for security*, pages 91–96, 2016.

[9] Asem Ghaleb and Karthik Pattabiraman. How effective are smart contract analysis tools? evaluating smart contract static analysis tools using bug injection. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 415–427, 2020.

[10] Collectiveshift. All about automated market makers (amms)'. https://collectiveshift.io/defi/amm-guide/, 2024. Accessed: 2024-01-01.

[11] Quicknode. Guides - ethereum development - transactions - 'what are ethereum transactions? https://www.quicknode.com/guides/ethereum-development/transactions/what-are-ethereum-transactions, 2023. Accessed: 2023-12-11.

[12] Ana Rita Amorim Melo. State of web3 security: analysis of vulnerabilities in bug bounty reports. Master's thesis, 2023.

[13] Andrew Park, Matthew Wilson, Karen Robson, Dionysios Demetis, and Jan Kietzmann. Interoperability: Our exciting and terrifying web3 future. *Business Horizons*, 66(4):529–541, 2023.

[14] Xiangjuan Jia, Jing Xu, Mengwei Han, Qing Zhang, Lu Zhang, and Xiaofeng Chen. International standardization of blockchain and distributed ledger technology: Overlaps, gaps and challenges. *CMES-Computer Modeling in Engineering & Sciences*, 137(2), 2023.

[15] Chainalysis Team. Vulnerability in curve finance vyper code leads to multi-million dollar hack affecting several liquidity pools. https://www.chainalysis.com/blog/curve-finance-liquidity-pool-hack/, 2023. Accessed: 2023-08-08.

[16] Etherscan. Curve aleth/eth pool exploit - private transaction. https://etherscan.io/tx/0xb676d789bb8b66a08105c844a49c2bcffb400e5c1cfabd4bc30cca4bff3c9801, 2023. Accessed: 2023-07-30.

[17] Ethereum Foundation. Documentation - transactions. https://ethereum.org/en/developers/docs/transactions/, 2024. Accessed: 2024-01-09.

[18] Lucidchart. What is a venn diagram. https://www.lucidchart.com/pages/tutorial/venn-diagram, 2024. Accessed: 2024-01-01.

[19] Jinyang Yu, Xiao Zhang, Jinjiang Wang, Yuchen Zhang, Yulong Shi, Linxuan Su, and Leijie Zeng. Robust and trustworthy data sharing framework leveraging on-chain and off-chain collaboration. *CMC-COMPUTERS MATERIALS & CONTINUA*, 78(2):2159–2179, 2024.

[20] Vijaya Killu Manda and Vedavathi Katneni. The critical role of blockchain oracles in web 3. In *Decentralizing the Online Experience With Web3 Technologies*, pages 207–224. IGI Global, 2024.

[21] EigenLayer Team. Eigenlayer: The restaking collective. *URL: https://docs.eigenlayer.xyz/overview/whitepaper*, 2024.

[22] Bhabendu Kumar Mohanta, Soumyashree S Panda, and Debasish Jena. An overview of smart contract and use cases in blockchain technology. In *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–4. IEEE, 2018.

[23] Immunebytes. All that you need to know about pausable smart contracts. `https://www.immunebytes.com/blog/are-smart-contract-pausable/`, 2022. Accessed: 2022-12-26.

[24] Huawei. Firewall security policy: What it is and how it works. `https://support.huawei.com/enterprise/en/doc/EDOC1100172309`, 2020. Accessed: 2020-12-02.

[25] Ironblocks. Ironblocks' open-source, on-chain, self-service web3 firewall is live.

[26] Ironblocks. Onchain firewall monorepo.

[27] Eigenlayer Labs. Eigenlayer docs - operator guides - introduction. `https://docs.eigenlayer.xyz/eigenlayer/operator-guides/operator-introduction`, 2024. Accessed: 2024-01-01.

[28] Sebastian Henningsen. Empirical and analytical perspectives on the robustness of blockchain-related peer-to-peer networks. 2022.

[29] Neil Efren Villanueva. Blockchain technology application: Challenges, limitations and issues. *Journal of Computational Innovations and Engineering Applications Vol*, 5(2), 2021.

[30] Shailey Singh. The emergence of web 3 and its core building blocks: Understanding the third iteration of the internet. In *Concepts, Technologies, Challenges, and the Future of Web 3*, pages 1–22. IGI Global, 2023.

[31] Solidity Team. Documentation - language description - contracts - function modifiers. `https://docs.soliditylang.org/en/latest/contracts.html#function-modifiers`, 2024. Accessed: 2024-01-01.

[32] Geeks for Geeks. Separation of concerns (soc). `https://www.geeksforgeeks.org/separation-of-concerns-soc/`, 2024. Accessed: 2024-02-13.

[33] Luca Donno. On block sizes, gas limits and scalability'. `https://ethresear.ch/t/on-block-sizes-gas-limits-and-scalability/18444`, 2024. Accessed: 2024-01-24.

[34] Solidity Team. Documentation - language description - units and globally available variables. `https://docs.soliditylang.org/en/latest/units-and-global-variables.html`, 2024. Accessed: 2024-01-01.

[35] OpenZeppelin. Docs - api - proxy. `https://medium.com/l2beat/stages-update-security-council-requirements-4c79cea8ef52`, 2024. Accessed: 2024-01-01.

[36] Fee mechanism will be added to this policy.

[37] The Investopedia Team. Multi-signature wallets: Definition and use cases. `https://www.investopedia.com/multi-signature-wallets-definition-5271193`, 2023. Accessed: 2023-10-19.

[38] Wikipedia. Secure multi-party computation. https://en.wikipedia.org/wiki/Secure_multi-party_computation, 2024. Accessed: 2024-02-01.

[39] Luca Donno. Introduction to timelock smart contracts. https://www.lcx.com/introduction-to-timelock-smart-contracts/, 2023. Accessed: 2023-08-14.

[40] Nathan Reiff. Decentralized autonomous organization (dao): definition, purpose, and example. https://www.investopedia.com/tech/what-dao/, 2024. Accessed: 2024-05-17.

[41] Luca Donno. Stages update: Security council requirements. https://medium.com/l2beat/stages-update-security-council-requirements-4c79cea8ef52, 2023. Accessed: 2023-12-07.

[42] Shermin Voshmgir, Michael Zargham, et al. Foundations of cryptoeconomic systems. *Research Institute for Cryptoeconomics, Vienna, Working Paper Series/Institute for Cryptoeconomics/Interdisciplinary Research*, 1, 2019.

[43] Zhi Li, Ali Vatankhah Barenji, and George Q Huang. Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform. *Robotics and computer-integrated manufacturing*, 54:133–144, 2018.

[44] Ethereum Foundation. Docs - foundational topics - ethereum virtual machine (evm). https://ethereum.org/en/developers/docs/evm/, 2024. Accessed: 2024-01-01.

[45] Werner Vermaak. What are flash loan attacks? https://coinmarketcap.com/academy/article/what-are-flash-loan-attacks, 2022. Accessed: 2022-01-01.

[46] Immunefi. The ultimate guide to reentrancy. https://medium.com/immunefi/the-ultimate-guide-to-reentrancy-19526f105ac, 2023. Accessed: 2023-05-23.

[47] Yeajun Kang, Wonwoong Kim, Hyunji Kim, Minwoo Lee, Minho Song, and Hwajeong Seo. Malicious contract detection for blockchain network using lightweight deep learning implemented through explainable ai. *Electronics*, 12(18):3893, 2023.

[48] Chainalysis Team. Crypto mixers and aml compliance. https://www.chainalysis.com/blog/crypto-mixers/, 2022. Accessed: 2022-08-23.