Drosera

The Security Automation Layer

Samuel Glenn
Jacob Veal
Fernando D. Reyes, Jr.
Richard Malone

Abstract

Complex smart contracts and multi-contract protocols pose significant risks to users and the integrity of the ecosystem. While individual contracts can be formally verified, it becomes practically impossible when dealing with modular multi-protocol systems. We propose Drosera, a decentralized security automation layer on Ethereum, to reduce these risks. Drosera is a set of smart contracts that allow operators to perform zero-knowledge incident response roles for protocols. This means that operators act on the security intents defined by Protocols. Security infrastructure is offloaded to a network of operators, which reduces gas costs for users, increases security, and decreases incident response time. Operators can trigger emergency response mechanisms to alert the community and mitigate damage. These operators have additional slashing conditions imposed on them if they perform malicious or incorrect attestations. Rather than creating a new trust network, Drosera aims to leverage Ethereum's existing trust network to bolster its crypto-economic security.

1. The Problem: Risk Mitigation

Decentralized networks have revolutionized the way we manage digital assets. In 2022, nearly $3.8 billion was siphoned from the DeFi ecosystem through various smart contract compromises [1]. Compromised protocols remain a significant challenge because of reliance on insufficient security coverage and poor incident response plans. Ethereum needs a native security complement to mitigate these risks. Drosera aims to create a framework that facilitates innovation and real-world advancements in these areas.

1.1 Insufficient Security Coverage

In EVM smart contracts, advanced validation logic presents a double-edged sword. Protocols must navigate the fine line between implementing comprehensive validation checks and ensuring users aren't burdened with high gas costs incurred in transactions. This issue increases the risk of a compromise to maintain user accessibility. As Ethereum's gas prices remain volatile, implementing complex validation checks can deter

users because of increased transaction costs. Theoretically, formal verification methods could assure protocol correctness and minimize the need for intricate validation logic [2]. However, its cost, complexity, and time-consuming nature hinder its adoption and render it fundamentally impractical for complex modular systems. The protocol owners bear the burden of round-the-clock incident response, a strategy that is invariably too late to mitigate damage, making additional security supplements indispensable.

1.2 Robust Security Requires Auditing and Monitoring

Auditing remains a leading security measure that protocols employ. However, the reliance on auditors has risks. Data validation flaws, which are highly common and hazardous, comprise 36% of smart contract findings, and access control issues account for another 10%. These vulnerabilities, if exploited, can lead to significant damage [3]. Due to the potential for human error, protocols cannot fully trust the auditor's contract assessment. This reality underlines the importance of additional protective measures. Despite the potential of automated tools, only 26% of all findings could likely be detected using feasible static approaches, and 37% using dynamic methods. Even considering only the worst high-low findings, static tools are less effective than dynamic ones (33% vs. 63%) [3]. Therefore, it would be prudent to complement auditing with failsafe logic and responsive actions in case of any identified risk.

2. Drosera: The Security Automation Layer

The name Drosera comes from a genera of carnivorous plants. The team liked the idea that a little liquidity could be used to help protocols catch bugs. It made us think about how protocols can be symbiotic and how quickly catching a bug should be incentivized. Most people are familiar with the toothed snap traps of the Venus Flytrap, which led us to the name “Traps” for our new security primitive.

Drosera aims to leverage the decentralized nature of Ethereum consensus to create a base layer for security. Protocols define incident response logic for Operators to carry out. Slashing and reward mechanisms ensure honesty and accountability. This approach to security expands monitoring and bug bounty programs to a dynamic model. Drosera maintains an open and efficient platform, encouraging collaboration and shared knowledge among the participating parties. This could result in the evolution of advanced security solutions and best practices, contributing to the overall enhancement and resilience of the Ethereum ecosystem.

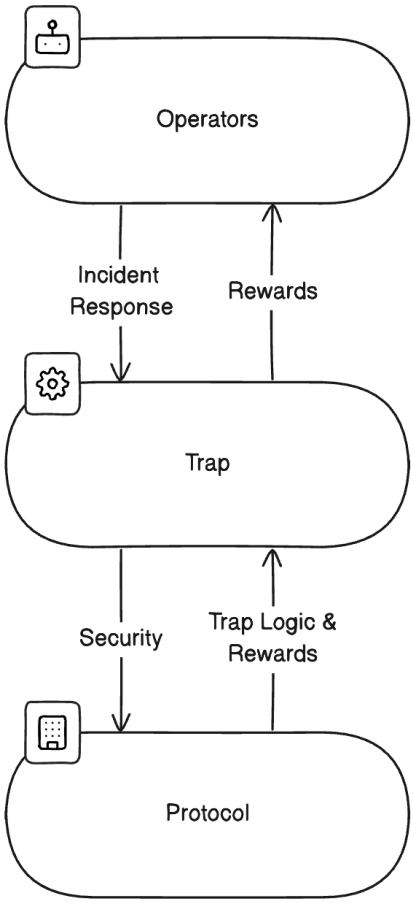

Figure 1: Drosera Value & Trust Model.
Operators receive rewards for performing incident detection and response.
Traps are security infrastructure as solidity code.
Protocols create Traps and provide operators rewards for application security.

2.1 Drosera’s Role & Responsibilities

Drosera aims to revolutionize incident response by providing a framework that enables decentralized incident response systems. It creates a platform for protocols and the community to collaborate on incident detection and response. Drosera's core role is facilitating trust between participants, while its responsibilities center around fostering a secure, collaborative, and rewarding environment. This empowers both Protocols and Operators to contribute to a safer decentralized ecosystem.

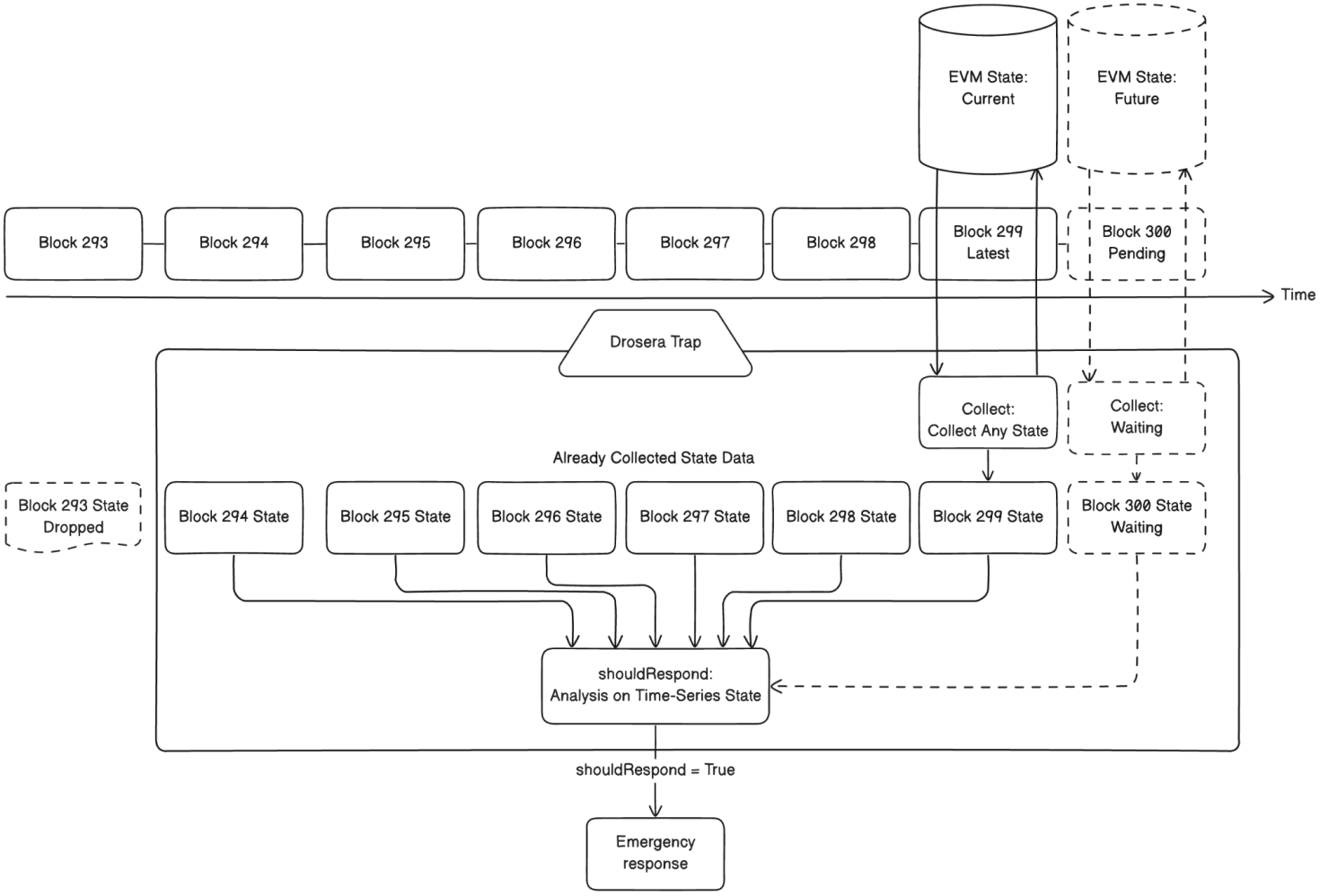

Figure 2: A Trap

2.2 Core Concepts

- **Traps**: A Trap is a security infrastructure smart contract that defines instructions for performing incident response when run by the Drosera network.
- **Operators**: Operators are nodes in the Drosera network that opt into Traps to receive rewards and secure the EVM.
- **Hydration Streams**: A Hydration Stream is a stream of tokens created for a specific Trap that distributes rewards to operators running the Trap. Anyone can make a hydration stream at any time.
- **Bloom Boost**: ETH deposited on a Trap that boosts the priority of the emergency response transaction.

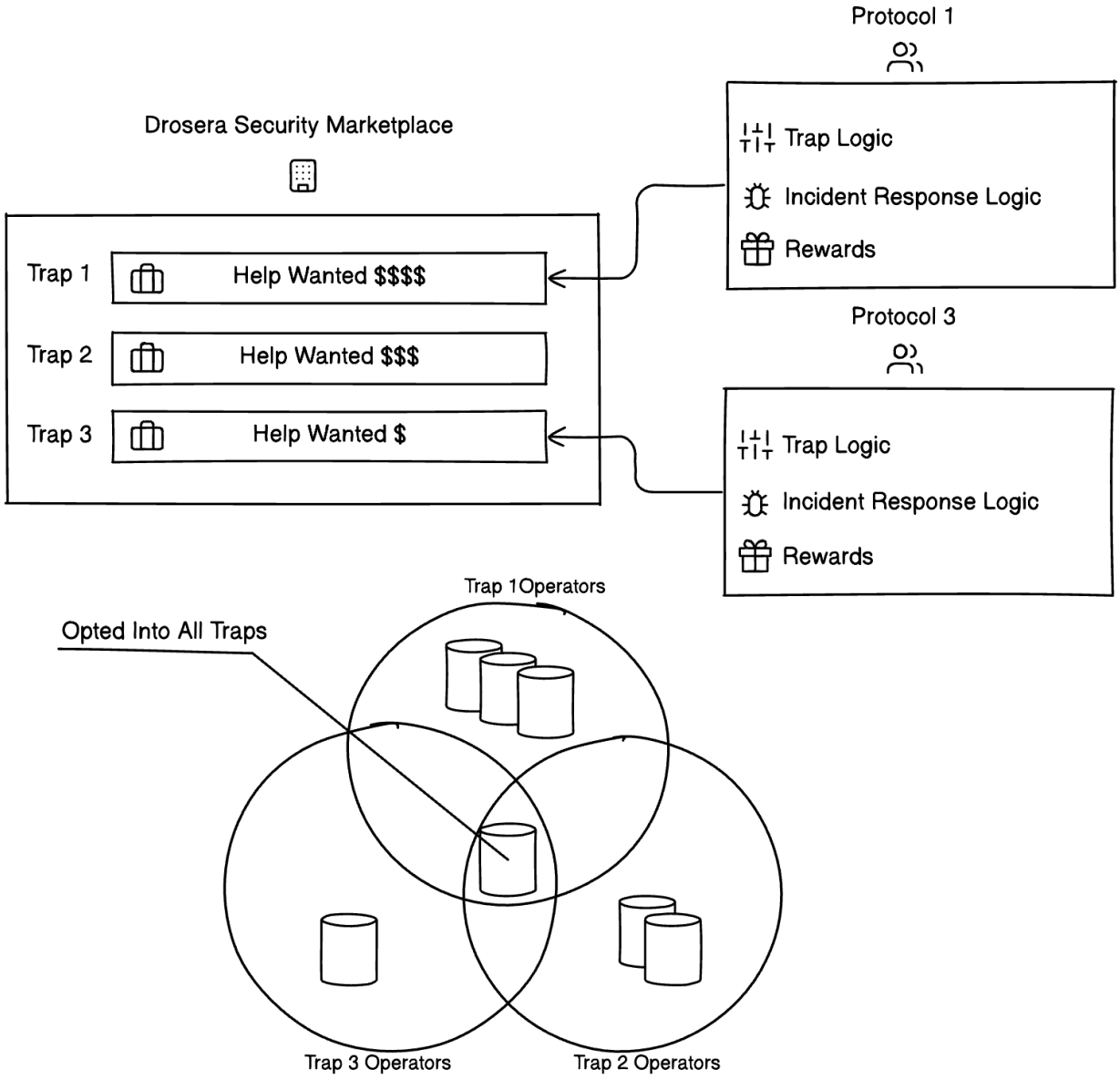

Figure 3: Drosera Security Marketplace

3. Protocols: Cultivating Security

Protocols use Drosera as a security marketplace.

The Protocol's responsibilities include:

1. Creating a "detection and response" contract, otherwise termed a **Trap**
    a. Trap contract bytecode lives off-chain while other configurations for the Trap live on-chain in a TrapConfig contract
2. Allow for the TrapConfig contract to trigger the emergency response mechanism.
3. Filling up the reward pool for operators that opt into the Trap.

    a. Rewards are facilitated through **Hydration Streams**. These allow token rewards to be streamed to operators over time and when incident response is performed.
4. Boosting the emergency response transaction speed via **Bloom-Boost**

3.1 Trap Logic

Traps are smart contracts that collect and analyze state data to decide whether an incident has occurred. If a Trap's shouldRespond function returns true, an incident has been detected. A simple check may compare two variables in a smart contract's storage to ensure they are always the same. A more advanced check may collect and analyze data from multiple smart contracts. The Drosera Operators run these checks off-chain, and consensus on the results for each block is reached via BLS signature aggregation. For example, suppose the trap logic is to determine whether the weighted sum of the elements of vector x equals the sum of the vector y's components. We can represent this function as follows:

$$V(x, y, w) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w[i] \cdot x[i] = \sum_{j=1}^{m} y[j] \\ 0, & \text{otherwise} \end{cases}$$

This function returns true if the weighted sum of all elements in vector x equals the sum of all elements in vector y and false otherwise. The result is included in the signed message and broadcasted to other Operators. Each Operator i has a private key x_i and a corresponding public key g^{xi}, where g is a generator of the elliptic curve group.

Signature Generation: Each Operator i signs a message m using their private key x_i. The signature σ_i is computed as $(H(m))^{xi}$, where $H(m)$ is a cryptographic hash function that maps m to a point on the elliptic curve.

$$\sigma_i = (H(m))^{x_i}$$

Signature Aggregation: An aggregator collects all signatures σ_i and aggregates them to form the aggregate signature sigma.

$$\sigma = \prod_{i=1}^{n} \sigma_i$$

Signature Verification: To verify the aggregate signature σ, check if the following equation holds:

$$e(\sigma, g) = e\left(\prod_{i=1}^{n} H(m), g^{x_i}\right)$$

where e is a bilinear pairing function.

3.2 Incident Response Mechanisms

When Operators detect that an incident has occurred, Drosera's incident response mechanism is executed immediately upon a ⅔ majority consensus and emergency transaction inclusion. Each operator has the opportunity to submit the claim signed by the network. The first incident response transaction included in a block will execute the Traps response function, and the operator that submitted the transaction will get a bonus reward for the incident response. This is where Drosera's rapid reaction capabilities play a critical role in reducing the impact of any potential compromise.

4. Operators: Guarding The Greenhouse

Operators serve as protocol sentinels by opting into Traps that fit their risk profile. Carrying out real-time incident response tasks involves running off-chain logic and vigilantly examining the EVM over time. Operators perform consensus with other operators through peer-to-peer network communication.

The Operator’s responsibilities include:

1. Opting into one or more Traps.
2. It runs an off-chain shadow fork of the EVM that executes the Trap contract’s bytecode.
3. Aggregate claims of “shouldRespond” from the network of operators to collect ⅔ of the Operator’s signatures.
4. Broadcast claims of “shouldRespond”.
5. It executes the incident response mechanism when a shouldRespond=true.

By opting into a trap, the operator commits to additional slashing conditions. Slashing holds Operators accountable for bad behavior, while rewards incentivize Operators to perform their tasks.

4.1 Attestations of incident response

Similar to Ethereum consensus, Operators aggregate data that other Operators broadcast. Operators use peer-to-peer communication to achieve majority consensus on the protocol's Trap contract logic results. This allows Operators to detect exactly when the Trap analysis identifies an incident, which triggers an incident response mechanism.

4.1.1 Claims

A ‘claim’ is the data broadcasted from an Operator to the network of operators via peer-to-peer communication. Claims are a set of data packets, each with a block number and the Trap shouldRespond result. To be considered valid, the claim must be signed by at least ⅔ of the total Operators.

4.1.2 Submissions

A submission is a claim with ⅔ of all signatures and has been submitted to the Ethereum network. Submissions are sent to the main Drosera contract, and the incident response is executed on the specific Trap’s TrapConfig contract. A submission is only performed when an incident is detected, and the emergency response must be executed.

4.2 Rewards

Drosera contains multiple reward systems. There are Trap Rewards for passive/active participation in the Trap and Staking Rewards for users that stake Drosera’s native token, DRO. Trap reward percentages are applied to the hydra

Trap Rewards

- Passive Reward
  - 70% rewards per second.
  - Operators are passively given rewards over time based on this reward percentage, the number of operators opted into the trap, and active hydration streams.
- Active Reward
  - 20% rewards per second
  - A bonus reward pool fills up over time based on this percentage.

        - An operator is given 50% of the bonus reward for being the first to submit a claim that reaches 2/3 consensus. While the other 50% is distributed to operators that participated in signing the claim that was sent on-chain.
    - Staking Reward
        - 10% rewards per second
        - Drosera's staking pool accumulates this percentage of rewards from each trap.

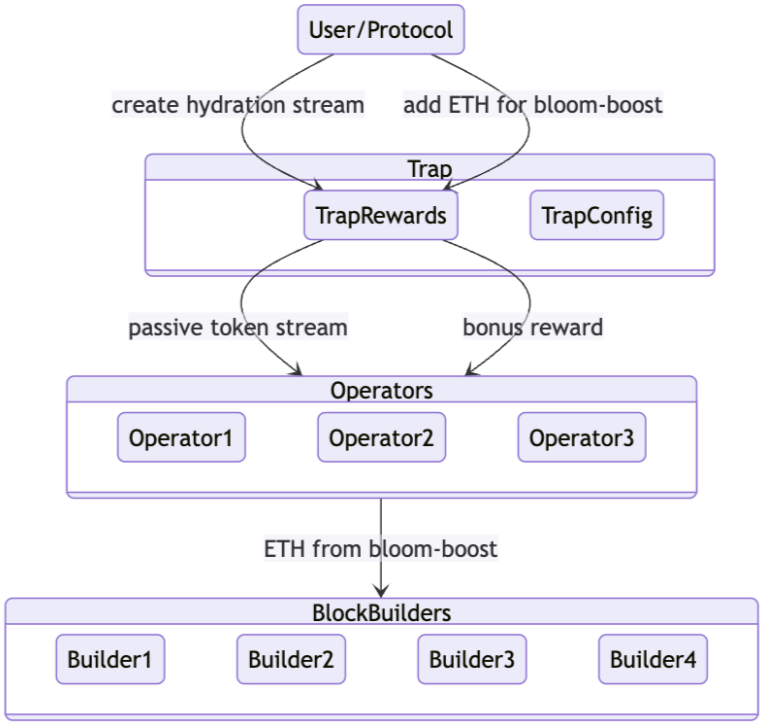

Figure 4: Reward Flow

4.2.1 DRO Token

The DRO token is the native token of the Drosera protocol and is used for various functions within the ecosystem. It is an ERC20-compatible token that leverages the ERC1363 standard to create Hydration Streams and Traps. Additionally, the DRO token is used for staking and staking reward facilitation.

4.2.2 Hydration Streams

The Hydration Stream system allows for the distribution of rewards to a set of operators. The rewards are distributed over time and dynamically allocated based on the number of operators who have opted into the Trap. Additionally, the

rewards are distributed to a bonus reward pool (for emergency response) and the Harvester Pool (for staking rewards).

4.2.3 Bonus Rewards

A bonus reward pool exists for each Trap, accumulating a percentage of tokens from hydration streams over time. Half of the reward amount is allocated to an operator that submits an emergency claim that reaches 2/3 consensus. The other half of the bonus reward is split between the operators that participated by signing the submitted claim. Each Trap has its reward contract and bonus reward pool. The Trap's associated reward logic funds the pool.

4.2.4 Staking

Staking DRO tokens into the Harvester allows users to earn rewards from the Harvester Pool. The rewards are distributed based on a weighted distribution of the DRO staked and the duration of the stake. All hydration streams donate a percentage of rewards to the Harvester Pool. This allows users to stake their DRO to generate yield to their protocol security budget.

4.3 Penalties & Slashing

If the claims from a submission are proven incorrect via a SNARK proof, then the signers who attested to the submission are slashed. This is a case where malicious behavior can be proven, and the punishment is detrimental to the Operators. Slashing kicks the operators who colluded from the network and freezes them so they cannot participate in Traps. Integrating with restaking platforms in the future will also allow for stakes to be slashed.

4.3.1 Restaking

The Drosera team is primarily focused on developing Drosera’s core business logic at this time. Restaking integration is a feature we plan to build as Drosera matures. Leveraging re-staking protocols for additional crypto-economic security allows Drosera to bolster its security posture, making it a valuable addition to Drosera’s feature set. Additional research is being conducted to identify how restaking can be leveraged on EVM L2s.

4.3.2 Mitigating Inactive Signing

The Drosera team plans to develop a mechanism that detects when an Operator has not signed any of the last X submissions and then penalizes the Operator.

Monitoring peer-to-peer activity is a simple measure to detect if an Operator attests to blocks and communicates with peers.

4.3.3 Mitigating Censorship

The Drosera team plans to develop a mechanism to detect repeated censorship by Operators. Suppose an Operator has repeatedly not included a signer in their last X submissions. In that case, the operator can be penalized for censorship unless the signer is identified as not actively signing.

Mitigating inactivity and censorship is important for ensuring a healthy set of Operators. The team plans to develop these slashing conditions once the core protocol business logic has matured and a wider audience can participate.

5. Drosera: Use Cases

There have been a plethora of major hacks since the beginning of DeFi. Common security vulnerabilities tend to be caught in the smart contract auditing phase, but many hacks occur because of misconfigurations, unforeseen scenarios, and undetected design flaws. Critical vulnerabilities are found daily in the tech industry; as these systems become more complex, there is a higher likelihood of human error. Looking back at these, we can identify how Drosera could be used in real-world scenarios. This shows how valuable it is for a protocol to have a built-in defense mechanism.

5.1 Addressable Market

The addressable market for Drosera is expansive as the decentralized finance (DeFi) ecosystem continues to grow and evolve. DeFi protocols recognize the importance of security and often implement various security measures to protect their users and assets. Drosera presents itself as a valuable solution to this growing market by offering a decentralized security base layer.

Here are just a few pre-existing projects that have built-in defense mechanisms:

Lending and Borrowing	Decentralized Exchanges	Derivatives, Synthetics, and Options
MakerDAO Compound Aave	Curve PancakeSwap Bancor Kyber Network	Synthetix DyDx Opium

C.R.E.A.M Finance Venus Protocol	0x Balancer	

Yield Farm/Aggregator	Liquidity Management and Staking
Yearn.Finance Harvest Finance Rari Capital Alchemix Idle Finance BadgerDAO	mStable Gnosis KeeperDAO

As DeFi becomes more complex and additional precautions must be taken, protocols will look for ways to protect their applications in a decentralized capacity.

5.2 Nomad Hack

The Nomad hack originated with a transaction at 9:32 pm UTC, where an individual successfully extracted 100 Wrapped Bitcoin (WBTC) from the bridge, valued at roughly $2.3 million. Following the community's concerns regarding a possible breach, the Nomad team acknowledged the situation at 11:35 pm UTC, stating they were aware of the incident and actively investigating it. The perpetrators extracted tokens over a multi-block time period, with each withdrawal amounting to approximately 202,440 USDC. The withdrawal transaction was executed over 200 times. [6]

This hack is a good example of a multi-block compromise where Drosera could help mitigate ongoing damage. Imagine if Drosera operators opted into a Nomad Trap and ran off-chain infrastructure to detect compromises by looking at state changes in the EVM. There are two scenarios where Drosera could provide the protocol and community with support:

Existing Defense Mechanism: If Nomad implemented an emergency pause button in their smart contracts.

- In this scenario, the Drosera participants detect an incident for Nomad. Each operator signs an attestation that a compromise occurred. The incident response is triggered after the detected compromise and could mitigate the other exploits.

Fire Signal Mechanism: Nomad has no response mechanism implemented in their smart contracts to stop or mitigate compromises.

- In this scenario, the Drosera operators detect an incident state for Nomad. Each participant signs an attestation that a compromise occurred, and the incident

response is triggered. Due to the lack of a defense mechanism for the protocol, this response action can only flag the state change, acting as a fire signal that the protocol is compromised. This can be a public warning that other protocols, users, and organizations can immediately be aware of the issue. With no response mechanisms available, the protocol cannot defend itself. However, the protocol could be used to perform white hat transactions. During the Nomad Hack, Moonbeam Network exercised its way to protect itself and entered “maintenance mode.”

5.3 Wormhole Hack

Wormhole served as a decentralized token bridge, enabling the transfer of cryptocurrencies across multiple blockchain networks. The bridge fell victim to a security exploit, leading to the theft of 120,000 Wrapped Ether (wETH) tokens, equivalent to $321 million. The attacker exploited the system by minting wETH on Solana and exchanging 93,750 wETH for $254 million in ETH on the Ethereum network. In response, the Wormhole team announced a $10-million bug bounty to recover the stolen funds. [7]

This hack is a good example of a protocol using a defense mechanism. The protocol hopes the bug bounty will incentivize the hacker to return the funds. Unfortunately, hope is not an effective strategy for damage mitigation. Drosera would enable a defense mechanism where the bug bounty funds could be used to pay participants for their services. Imagine if Drosera operators opted into a Wormhole Trap run off-chain infrastructure to detect compromises. The same scenarios apply in the Nomad hack case provided above. This allows the operators to take emergency action when they have all come to consensus.

6. A World With Drosera

6.1 Drosera Enables New Applications

Drosera's approach to security enables the development of new applications with built-in defense mechanisms to protect against risks. Drosera allows developers to explore novel use cases and build cutting-edge decentralized applications, knowing that their projects will be protected by robust security infrastructure. Drosera reinforces the existing infrastructure, ensuring decentralized applications and protocols can confidently operate.

6.2 Drosera Expands Response Mechanisms

With Drosera, the Ethereum ecosystem benefits from a diverse array of response mechanisms that go beyond current security measures. Drosera's approach enables the rapid detection of compromises, ensuring swift and effective responses to security threats ultimately mitigating potential damage. Drosera's approach to incident response encourages a collaborative environment where developers and security researchers work together to improve the overall security of the ecosystem.

6.3 Flexible Marketplace for Customized Responses

Drosera's design enhances security and offers a versatile marketplace for protocols seeking customized responses based on specific conditions. For instance, protocols can trigger actions like swapping tokens in a treasury if a stablecoin de-pegs, bridging tokens after a certain block number, or updating a vault's strategy once enough stakers attest. This flexibility allows protocols to adapt and respond effectively to various scenarios, fostering a more resilient and efficient decentralized ecosystem.

6.4 Hidden Security Intents

Keeping an intent hidden is an important factor in adversarial security. Knowing which conditions trigger a protocol's emergency systems is like giving an adversary the blueprint of a maze. They will know which paths lead to traps and deadends and which levers to pull. Essentially, hiding intents allows for new applications that require the “why” an action is performed to be obscured/hidden.

6.5 Using Block Building for Rapid and Bundled Execution

Analyzing Drosera from a block-building perspective reveals intriguing possibilities for leveraging Block builders to support the network's emergency actions. By incentivizing incident response, Drosera can promote faster and bundled response execution, enhancing the system's efficiency and ability to react to potential compromises. This innovative approach to security further strengthens the Ethereum ecosystem and its attractiveness to decentralized applications and protocols.

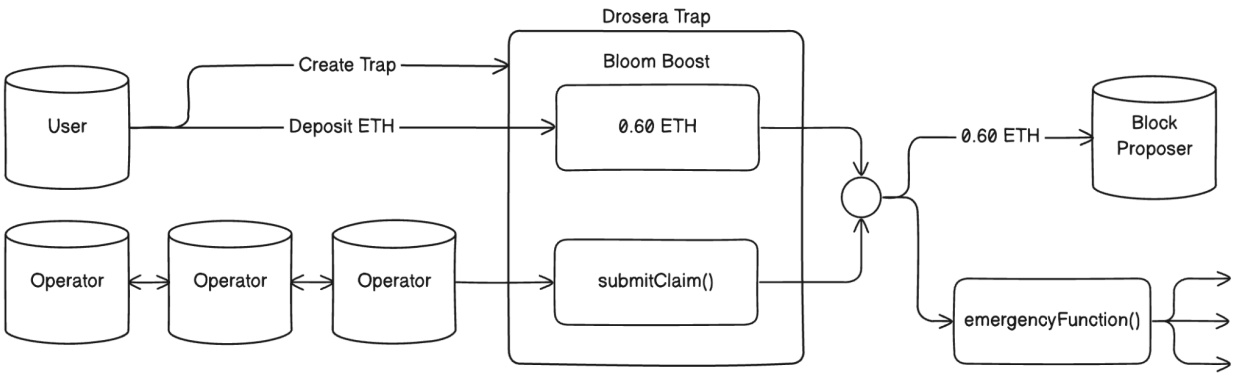

Figure 5: Bloom Boost Diagram

7. Summary: Roots To Leaves

Drosera is a revolutionary security marketplace that allows protocols to create an oasis within Ethereum's dark forest. It enhances the security and resilience of the Ethereum ecosystem, driven by the community, for the community - all while supporting both new and existing applications to thrive amidst the ever-changing landscape. By fostering decentralized responses to security incidents, Drosera provides a framework for fighting back against looming threats.

References

[1] Compromises 2022:
https://www.reuters.com/technology/crypto-hacks-stole-record-38-billion-2022-led-by-north-korea-groups-report-2023-02-01/

[2] Formal Verification:
https://ethereum.org/en/developers/docs/smart-contracts/formal-verification/

[3] Trail of Bits Findings
https://blog.trailofbits.com/2019/08/08/246-findings-from-our-smart-contract-audits-an-executive-summary/

[4] Beacon Node Event Stream:
https://ethereum.github.io/beacon-APIs/#/Events/eventstream

[5] Ethereum RPC API:
https://ethereum.org/en/developers/docs/apis/json-rpc/#json-rpc-methods

[6] Nomad Hack:

https://cointelegraph.com/news/nomad-token-bridge-drained-of-190m-in-funds-in-security-exploit

[7] Wormhole Hack:
https://cointelegraph.com/news/wormhole-token-bridge-loses-321m-in-largest-hack-so-far-in-2022