Prowlis: Intelligent Metastaking Engine For Digital Assets with Full Stack Fine-Grained Control

The Prowlis Team

Table of Contents

1 Vision/Mission/Strategy

Vision: Every network keeps its value promises.

Mission: A management & control plane in every value network.

Strategy: Bringing Modularization, Fidelity, Automation, Aggregation to the Marketplace for Pooled Security.

Prowlis allows digital asset networks to align governance, policies, and goals via an intelligence platform utilizing metastaking and state of the art machine learning.

2 Goals

Prowlis introduces "Metastaking" as a novel crypto-security primitive that gives depth and breadth to the restaking ecosystem for the staker/network market. We provide nuanced controls and aggregated tooling to provide any staking goals that are needed. Our first goal in the market sector is to bring the Prowlis AI permissioned manageement & control infrastructure to the Ethereum restaking category. We are building a proof of concept with a machine learning managed curated selection of LRTs, operators, & AVSs. These new primitives, tooling, and capabilities enables the most optimal pathways to meet staking / network security objectives and goals that far exceed current restaking methods.

3 Current State

Networks are built on many spectrums. One such spectrum is the centralization-decentralization axis. On this axis, one of the challenges is the state synchronization of the data stores that are replicated amongst the nodes of the network. In blockchain and derivative based networks, this also includes the addition of new data to the network. An example of this is the method of adding the next block to the Bitcoin network's data store through the Proof-of-Work (PoW) mechanism. This is done as a sybil defense to determine who will be able to add a block to the network. Additionally, the threat of forking, by the rest of the nodes in the network not validating incorrect UTXO outputs (e.g., double-spend attacks), serves as a deterrent to the elected leader, via the PoW, of not keeping the network in a correct state. There have been other solutions to these challenges, for instance, Proof-of-Stake (PoS), amongst others. In the abstract, what these methods provides is a method to secure the network's ability to keep the network's data stores in an accepted state. In recent times, there has been work on extending the applicability of this security in different ways. We kindly request the reader to also note that these schemes can be seen as sybil resistance methods in certain circumstances.

4 Related Work

4.1 Lido

Lido is what's known as a platform that provides a liquid staking derivative (LSD) token. While there are critiques that LSDs lead to centralization in the Ethereum that is staked, it should be noted that they incentivize a higher percentage of Ethereum being staked and LSD holders do have the option to unstake if Lido or their validators go against the interests of the LSD holders as an exogenous lagged penalty. Lido is also working on Distributed Validation Technology (DVT) for endogenous incentives for validator alignment with LSD holders currently incentivizing DVT or their permissioned validator set.

Lido, on two attributes of security, increase and decrease the security aspects of Ethereum. On the Lido and current validator set, this increases centralization and lessens the security around control decisions for validation. In the worst case, this can lead to 51% attacks on the Ethereum network. However, on the other hand, by letting users partake in Ethereum validation rewards without having the requisite 32 ETH, leads to allowing more ETH to being stake, increasing the economic security value of the Ethereum network, making it harder, on an absolute monetary dollar value basis, to be able to engage in a 51% attack.

4.2 EigenLayer

As is apt in the history of software systems, new entrants come along that apply an abstraction layer to a previous concept and extend the concept in new ways. One of these is via EigenLayer. Building on the extending fluidity of value exchange in staking past the move for staking amount requirements in a solo network, EigenLayer extends this in multiple orthogonal directions / attributes, a key one being extending / hypothecating security across multiple networks along with different forms of security value exchange. This opens up new categories for the market of security (e.g., MEV, threshold/group trust, etc.). However, there is a lack of fine-grained control over various trust assumptions, goals/objectives, and management of these by the multiple sides of the market for trust. This is for both the infrastructure and the application layers. In particular, for EigenLayer, this is encapsulated via the total profit from corruption and the assumption that stakers themselves and their actions are fungible in a crypto-economic risk analysis.

4.3 Insights

There are two key insights to be gathered from the previous discussion:

1. Trust solutions have opposing and/or not structured/aligned security objectives.

2. Security is a mutable, modular, sub-composable object.

What are the implications of these insights?

1. With the right controls, security can be shaped and molded to meet needed goals.
2. New capabilities can be created to produce solutions to security & trust aspects of problems & issues in specific categories that were not possible before.

We believe that given recent related work and the aforementioned insights, now is the time to develop and deliver a platform to bring these elements together into a digital operating system of value.

5 New Work

5.1 Introduction

In the following discussion, we will frame the discussion with the analogy, in computer architecture, of the relationship of the BIOS (Basic Input Output System) with a software OS (Operating System).

From a high level, we propose a micro-kernel core architecture with a modular component-based extensibility pattern. The core security/trust controls across categories, types, and modules will be exposed by the core. For the use of the OS, there will be two different inter-faces, one that is low-level procedural interface and the other that is high-level declarative. Programs will be goal & constraint-based and can be built in a low/no-code, DSL (Domain Specific Language), or spec-based language-agnostic WASM base environment The connecting interface between these and the forthcoming Prowlis AI Stack, will be based on interaction nets.

Let **S** = *{s_1,...s_m}* and **T** = *{t_1,...t_n} be a set of m security constraints and n trust goals over the Y=SxT cartesian security/trust po-space. Let* **M** = M(S,T) be a mapping in in the DSL component subset X in the space Y := DSL(X) or a parse down-validated specification equivalent set Spec(input_string) -> ~DSL(X'). Up-stream and down-stream slashable events will be when a injective constraint mapping is violated on a homological algebraic variety **V** bounded by the implied trust goal scheme **G$_s$**. This can be encoded by the following dynamical evolution constraint formula:

Given **S**, **T**, **M**, **V**, and **G$_s$** from above,

$$\max_{g_t}\left[\prod_{t=0}^{n}\left[\widetilde{\nabla} g_t\left(\min_{f_s}\sum_{s=0}^{m} f_s\left(\left(\mathbf{S},\mathbf{T},\mathbf{M},\mathbf{V},\ \mathbf{G_s}\right)\right)(b)\ \right)\right]\right] \quad (1)$$

with the tilde nabla operator denoting the discrete approximation, g: A->B the mapping from the trust metric space A to the trust-value goals via a projection to the optimal trust subspace B, f: C->D denoting the trust metric D implied by the security constraints & trust goals C, and b denoting the block time. We call formula (1), the Security-Trust formula. In the sequel, we will further discuss this equation, its components, and how all of it fits together, both internally and externally.

From a high-level perspective, one view is to take formula (1), in a loose analogy, as taking the surface area while minimizing the security constraints that can have an impact on input into computing the maximal trust of a system. This can also be seen as a take on what some view as a tension between liberty and safety.[1] We will go from inside out while discussing the components of formula (1). f_s is the governing field relation of security (2). As we go about this discussion, we will be seeking properties we can provisionally apply to this equation. The first thing to notice is that the inputs determine the properties of the output trust metric D. For now, we will leave D as given, as we need the next two subsections so we can properly define it in the context of the overall system. At first glance, it might make sense to set f_s to some boundary $B_{D\text{-bound}}$ if **S** is at a boundary S_{bound}. Doing so can help us have the goal of slashable events be defined when an injective constraint mapping is violated. For the injective constraint mapping violation equation (3), please have given block times b_1 and b_2, such that $b_1 \lneqq b_2$.

5.2 Straight & Cross Value for Security and Trust

Before getting to the discussion of the elements of the ecosystem and market for security and trust and how they relate and interact with each other, we'd like to first bring forth the mutability of value, along the fungible-nonfungible axis. For the design of the Prowlis system, a question comes up regarding the endo/exogeneity of risk for non-identical units undergoing one or more comparisons. We hold that this is to be held outside the core security and trust apparatus, but that the same Security-Trust framework can be established to manage the cross-value for security and trust. We will have more to say on this later.

5.3 Coordinating the Ecosystem & Market for Security and Trust

As we delve further into the equation and how it relates to overall goals of Prowlis, we can look at formula (1), from the perspective of the platform/system, is for one or more actors who have security constraints and trust goals. We can denote the output, for a single actor a from the set of actors **A**, for formula (1) by $ST_a(.)$. This denotes the security-trust parametrization for actor a. We can then see that the total security-trust parametrization of the entire ecosystem can be encapsulated by:

[1] Referencing, as an example, the Benjamin Franklin quote, "Those who would give up essential Liberty, to purchase a little temporary Safety, deserve neither Liberty nor Safety."

$$U\left(\dots ST_{a_i}(.)\dots\right), a_i \in \boldsymbol{A} \quad (2)$$

We call formula (2), the Unified Security-Trust formula. A key question is what properties for and/or in relation to U be, such that we can capture security & trust parameters on the entire ecosystem. We can pick up on the discussion of the trust metric D from a couple of sections ago as one such property to further explore. A metric formally defines the notion of distance. For our purposes, we looking to define this distance on a structure that models the security constraints given for a set of actors, A_s.

Do note, we can also create partitions, poset/graph-based substrates, and other algebraic structures if needed to express different substructures. We will look to have structure implied by the principles needed to express the theoretical concepts and practical considerations needed to give a sound, yet pragmatic definition of security. For theoretical reasons, to remain consistent with foundational issues in mathematics & computer science, we do not look to a definition that is complete[2]. For the trust metric D, we want formula (2) to be defined such that it induces a gauge-free connection $C_{ig\text{-}fc}$ in a, non-necessarily strict, subspace that is a generalized manifold of the trust space where the metric D lives. Finally, to close out this section, we can use formulas (1) and (2) to determine absolute and relative calculations of trust. More will be delved into in our formal whitepaper.

5.4 Approximation: Monte Carlo, Machine Learning approaches to computing the Security-Trust equation

While the above gives a theoretical foundation upon which to evaluate security & trust in a multi-actor ecosystem, it is not tractable, in non-time resource usage, for real-world systems. However, our goal isn't to build something that "knows" all, which isn't is realistic, but to build something that can approach optimal learning under resource constraints. We'd like to briefly mention and look at two approaches to determining decisions for security & trust with formulas (1) and (2), that approximate formulas (1) and (2) under more realistic resource constraints.

One of the most straightforward is to use Monte Carlo by sampling from a random distribution for the inputs into (1) and (2), evaluating the results and keeping the highest or most desirable results. However, while this has low resource constraints, it doesn't give, quite literally, any direction to finding an optimal value. So, while it might be better in non-time resource usage, it has an unacceptable high variance in time-based resource usage. The approach we aim for is to use machine learning to determine the values of (1) as a foundation for the configuration of the implementation of a reference system. We propose to create a general-purpose foundation model trained in conjunction with a RLAIF model tied to the Prowlis system. Above these can be fine-tuned application-specific models. The former being a layer 1 and the latter a layer 2 in an AI stack. Furthermore, for the RL components, because of the structured connections in and between each S-T formula representing each actor, we propose, what is currently, in some places, being called

[2] See Gödel's Incompleteness Theorems or Turing's Halting Problem.

Q*-Learning, which is an amalgamation of Q-Learning and A* search. We then connect this with a graph-based deep learning model for formulas (1) and (2). Then, in the spirit of the recently released Google DeepMind Genie, we use a world model using a trust-security tokenizer, an autoregressive dynamics model, and a scalable latent action model.

Our goal in the former is automatic planning for user goals and in the latter is explainability and built-in structure for the base computation machine learning models to compute approximations for formulas (1) and (2). Additional, we shape the learning of the Prowlis AI Stack with a world model previously discussed. We envision these being a part of the Prowlis AI stack.

6 Architecture, Program Constructs & Algorithms

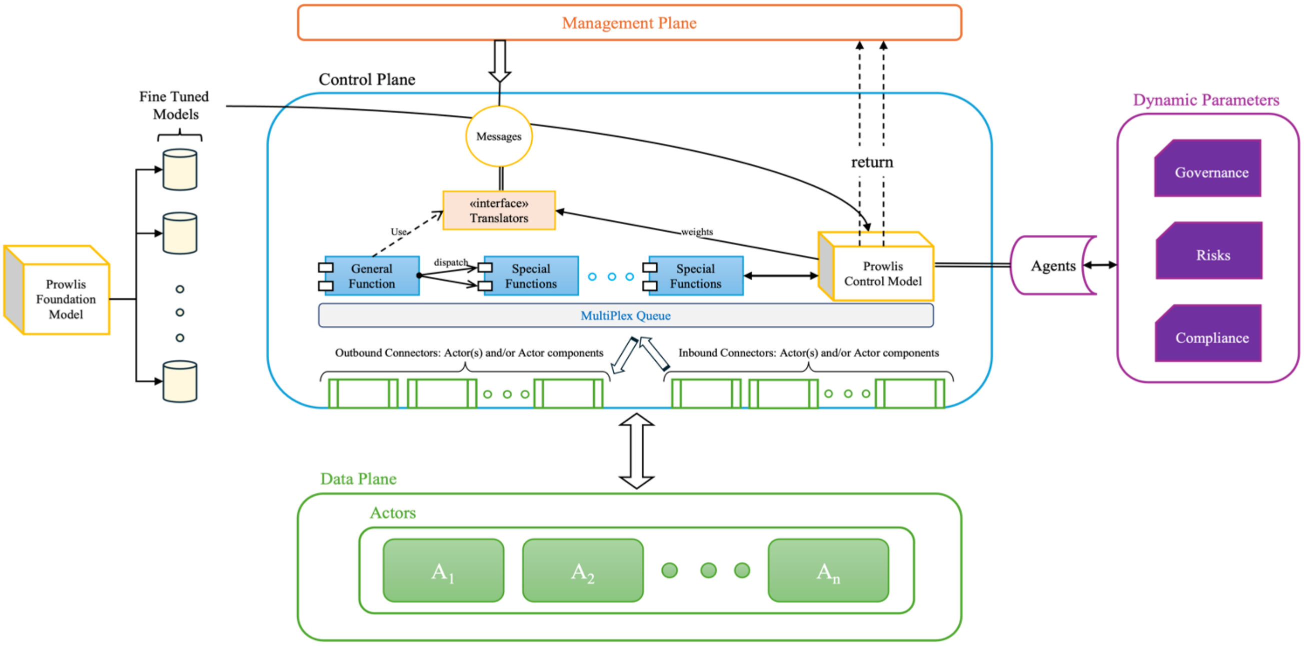
6.1 Architecture

One of the architectural slices we deploy when designing our system is via a n-tier plane construct. We propose to start with a 3-tier management, control, & data plane model. Before describing each plane in detail and giving visual representations of their relations to each other, do note that the data plane represents, in our architectural model, the current (re)-staking ecosystems data constructs, artifacts, and flows. Cross-cutting through these planes will be the aforementioned AI stack.

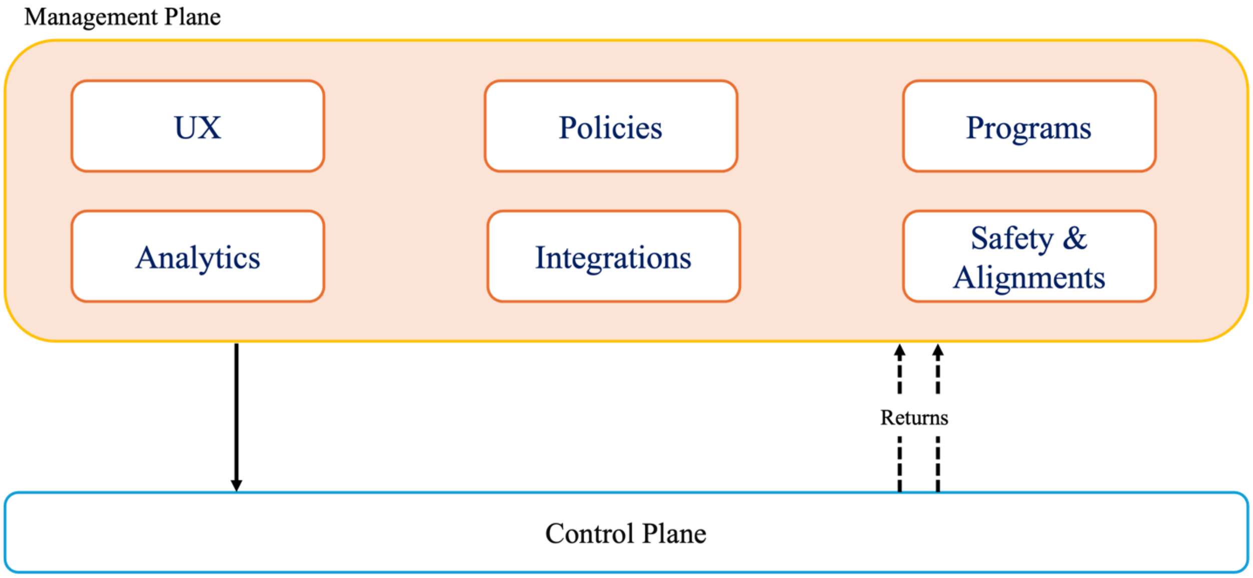
Data Plane: This is defined as narrowly where the data for staking resides. This includes details on the transaction inputs, computations, and outputs. More broadly, this encompasses the systems where any transactions or data reside that impact the staking mechanisms that are not specifically part of the control & management planes. There will also be auxiliary data planes, that are formally outside of the Data Plane. For example, from the AI stack.

The Control Plane consists of two nexuses:

1. Bidirectional Management-Data nexus
2. End-to-End AI Stack nexus



The Management Plane is as follows:



It is declarative based vs the more imperative control plane. The management plane is centered around the following six pillars:

UX – This is where the different users (stakers, operators, services, etc.) interface with the Prowlis platform. Through frontends, APIs, and other endpoints, users can interact with the Prowlis to meet their needs in managing & controlling their objectives, goals, & risks in the metastaking ecosystem.

Policies – These are the guidelines that users can employ for their staking needs. These can be created from templates, market sellers, or bespoke.

Programs – These are constructions that include structured code with one or more policies shaping their behavior.

Analytics – This is the data showing the performance of various staking activities by the users of the platform. This can be shown through dashboards, feeds, etc.

Integrations – 3rd party applications and solutions can add functionality via plugins and leverage the features & benefits of the systems via APIs.

Safety & Alignment – These are their own set of policies. They aim to implement safeguards at the ecosystem and actor levels to prevent and mitigate risks, and to align objectives and goals through scorecards.

6.2 Program Constructs Groups

Now, we will go over three important program construct groups in the Prowlis platform:

On-chain/Off-chain primitives, Scaled Modularity Mapping – Renormalization Groups, and the Transport, Observability, & Orchestration (TOO) toolchain or TooChain for short.

Now, since the Prowlis platform is agnostic to data & controls environment, yet aware, semantically, we treat any element that is a 1st party element in an on-chain environment, as still 1st party if in an off-chain or hybrid environment. This is a foundational principle for what we are calling X-chain primitives, or otherwise called on-chain/off-chain primitives.

Next, because of the feature engineering free nature of our platform, given the Prowlis AI Stack, we will be modeling the parameters, which the Prowlis AI Stack will discover, manage, and optimize for, via repurposing a notion from physics: Renormalization Group (RG). We plan to structure the approach for the data structuring through RGs and the analysis assisted and augmented with RG flows.

Finally, from a semantic level, information & transformation will be natively bi-directional, data storage agnostic, and infrastructure abstracted (modulo consensus & security guarantees). The Prowlis platform, from a tooling & language standpoint, will interface with & expose to the environment, consisting of users, actors, API calls, etc., a Transport, Observability, & Orchestration toolchain (TooChain). The three parts can be thought of as follows:

Transport – the movement of data & informational elements through the system. This will have abstractions, with a goal of generalizations of data availability and ordering.

Observability – a generalization of the oracle system in current crypto based systems. This will be a read-write-provenance omni-directional system.

Orchestration – this is a multi-network integration system, that abstracts underlying primitives or exposes them as needed for user & system functionality for alignment to actors needs, driven by the Security-Trust formula discussed earlier.

6.3 Algorithms

There are two main parts that algorithms will be introduced at the level of the Prowlis litepaper. One is at the Prowlis platform level. The other is at the Prowlis user level. From the platform level, there will be core systems running algorithms, along with maintenance systems running their needed algorithms. Some will be imperative for the base-level code, with other higher parts being declarative in nature. The user level will have, as described several sections earlier, but explicit here, for the user level, three interfaces:

- Low/no-code.
- DSL (Domain Specific Language)
- WASM base environment.

Programs will be goal & constraint-based and can be built in a low/no-code, DSL (Domain Specific Language), or spec-based language-agnostic WASM base environment.

More will be written later as to the details. As a preview, this is an initial view of how, for the US Treasury market, an end-user can direct the Transport module of the TOO chain:

DSL: Treasury -> Treasury ([meta]staking: (one-way-to, one-way-from, bi-directional hooks)]

7 Benefits, Constraints, & Tradeoffs

Given the system described above, Prowlis provides numerous benefits to the actors in the (meta)-staking ecosystem. The benefits, constraints, & tradeoffs for each of the main actors can be labeled:

Actors	**Benefits**	**Constraints**	**Tradeoffs**
Stakers	Choice in the best operators & services.	Have to communicate goals to the Prowlis system.	Limited to the providers on Prowlis versus protections.
Operators	Can have intelligence & control in providing the best management for stakers & services.	Limited by the learning curve to maximize the benefit, though a lot of the benefit can be done via low/no-code.	Coverage with low effort is great, but coverage with great effort is best.
Services	Getting the best operators and staking to deliver the value of their service.	Integrating their services system into the Prowlis platform.	Capability maximization through integration maximization.

Ecoystem	Brings alignment to the staking ecosystem.	Have to embed processes to enable use of Prowlis.	Planning vs ecosystem alignment.

8 Conclusion

The Prowlis platform introduces the concept of metastaking to enable the capability of management & control in the staking & restaking primitives being introduced in the blockchain sector, especially with, but not limited to EigenLayer on Ethereum. This extension also includes off-chain systems, with example categories including US Treasuries, private credit, and the intellectual property industry. Through the end-to-end AI stack nexus, we provide best-in-breed intelligence & goal attainment for all our users.