The Warden Protocol

The Warden Protocol Team[1,✉]

[1]Modular Blockchain Infrastructure for Omnichain Applications

This document contains forward-looking statements and information relating to Warden Labs, its business plan, strategy, and the relevant market. These statements are based on the beliefs of, assumptions made by, and information currently available to the Warden Lab's management. They reflect management's current views of future events and are subject to risks that could cause Warden Lab's actual results to differ materially from those contained in the forward-looking statements. Investors are cautioned not to place undue reliance on these forward-looking statements. The company does not undertake any obligation to update these forward-looking statements.

Modular Security | Omnichain Interoperability | Chain Abstractions
Correspondence: ***info@wardenprotocol.org***

Abstract

In this paper, we present the Warden Protocol - modular L1 blockchain infrastructure for omnichain applications, "OApps". Our mission is to empower developers to simply launch secure OApps.

Warden Protocol

In monolithic blockchain architectures, all security components of an application are tightly integrated into a single, centralised unit. For example, many early Bitcoin exchanges concentrated user funds in single hot wallets with unencrypted keys stored on single servers. Due to this component bundling, a vulnerability in the system can compromise any and all applications. Contrary to monolithic end-to-end blockchain architectures, we have modularized the Warden Protocol for security, interoperability and chain abstraction.

Application developers can assemble and disassemble a set of composable modules and use standardised, chain agnostic syntax, to create a new type of user experience - OApps. Each OApp component is developed, tested, documented, and benchmarked distinctly and be used individually or in combination with other components. All components are configurable by OApp developers. Any application developer can add Warden Protocol custom modules to their existing base app to turn their application into an OApp.

Warden Protocol is a high throughput, low latency, instant finality blockchain for OApp developers. Utilising Warden Protocol as a shared platform, OApp developers can tap into enshrined infrastructure and pool resources, granting them a competitive advantage lacking in standard applications. They can sidestep establishing and maintaining a validator set and relayer network, and can leverage built-in support for keychains, intents, block explorers, wallets, oracles, bridge, data indices and security monitors. This reduces development costs, accelerates deployment timelines, and permits OApp developers to concentrate on creating application specific moats, rather than duplicating tools, resources and infrastructure.

What are OApps?

OApps are a powerful evolution to traditional smart contracts. They consist of three parts: application and contracting logic, a stack of keychains, and a user-supplied, parameterizable intent configurator. Owing to this OApps can achieve remarkable features: they are modularly secure, omnichain interoperable and chain-abstracted.

OApps are modularly secure. They can support the same applications deployed with different security models, thereby decoupling protocol-layer from application-layer security. The result is homogeneous protocol security, with a heterogeneous application security that minimises security fragmentation, and captures a user's true intents when interacting with an application. Users can choose their trust assumptions, while application developers retain the network effects of being able to use a shared protocol security. Any TVL intensive DeFi application, that necessitates substantial deposits, such as liquid staking protocols, AMMs and DEXs or money markets, could experience significant advantages from deploying as an OApp.

OApps are omnichain interoperable. Collectively, they form an application mesh topology. This mesh is resiliently designed for cross-interoperability, overcoming isolated and fragmented ecosystems. Their connections are persistent and universal - whether it's letting users seamlessly swap across supported chains, interact with applications from other chains or exchange native assets for wrapped ones.

OApps are chain-abstracted. Whereas traditional smart contract applications only target users of a single chain, OApps can sign transactions and messages targeted for any other foreign chain. They can read and write to other chains which enables a host of completely new use cases enabled by OApps.

OApps are remarkably lightweight and straightforward to build. Developers can write in the language they love, use the tooling, frontend libraries, node and RPC providers, and wallet providers that they are most accustomed to.

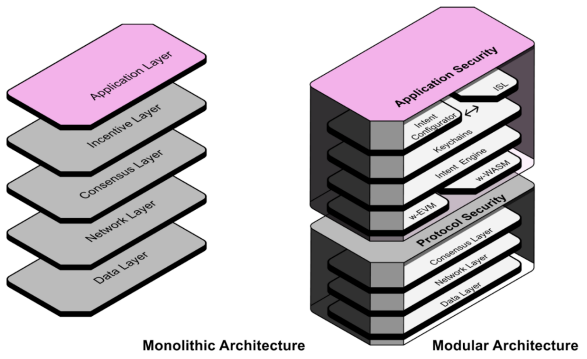

Fig. 1. Warden Protocols modular architecture unbundles the application layer for greater and more resilient security

Unbundling The Security Stack

Today digital assets are woefully vulnerable to determined adversaries, and web3 will not onboard billions of users unless we rethink, unbundle and specialize the security stack.

Shared protocol security entails applications on a given infrastructure adhering to the infrastructure's security requirements, like L2 solutions. These monolithic systems impose equal security on their applications. For instance, an application on Optimism has a 7 day dispute delay window which cannot be ignored or configured. Polkadot or Avalanche are multichain networks that use shared protocol security models to safeguard their parachains and subnets. A new rollup on Ethereum or Celestia receives an even protocol security from the validator set. This security inheritance is meant to be a guarantee for users - they can expect the guarantee to be upheld by whichever app they are using on the rollup, regardless of the applications internal security policy. However, there are drawbacks - depending on the type of error detected on the L1, vulnerabilities can cascade. A vulnerability on the protocol suddenly doesn't affect a single application, but depending on the type of bug, it can impact several, leaving developers and users with no means of recourse or correctional mechanism.

In contrast, isolated security allows each application to define its own security. This is sometimes seen on apps built on messaging protocols, like LayerZero. Each application developer defines its own relayer, oracle and validation libraries alongside a set of other security configurations. Critically, this is without guaranteeing components are independent and free from collusion. While this security configuration ensures responsiveness and adaptability, it transfers responsibility for assessing risks to end users. Each user has to separately validate the risk inclined with every application they want to use. It also assumes developers are trusted, reliable and honest third-parties. Should application developers be able to modify and cherry-pick how many nodes and which nodes execute transactions? Should they be able to edit security configurations without user consent? After all, there is no security layer on the application guarding against misuse.

Warden Protocol distinguishes between application-, and protocol-level security. Each OApp inherits protocol security from Warden Protocol, acting as a security aggregator and stabilising force for the OApp ecosystem. Security guarantees include its replicated, permissionless proof-of-stake consensus mechanism, the fault-tolerant and liveness properties of consensus, the validator set and node authentication, its secure channel communication, fork detection and handling, as well as its finality and censorship resistance. OApp developers retain network effects, and they don't have to bootstrap new validators for nascent applications. They don't incur the overhead of having to operate their own infrastructure, they have a lower security budget and are less susceptible to sybil-, long-range, eclipse or 51 percent attacks which will all contribute to lowering the barriers to new deployment.

Additionally, OApps inherit application-level security from keychains, and their intent engines. This is critical, because the application layer is closest to users, and represents the largest attack vector. With keychains and the intent engine, OApp users can configure distributed key creation, signatures, threshold signature schemes, role-based access controls and administrate signing authorization. This creates resilience against private key exploits, theft, spoofing and sweeping.

Thanks to this modularity, OApps can support the same application deployed with different security models, achieving homogenous protocol security with heterogeneous, isolated application security. Users can choose their trust assumption, while application developers retain the network effects of being able to use the same shared protocol security without incurring security fragmentation when scaling the number of applications. In addition, they stay responsive when new security technologies emerge.

Modular Security

A keychain is any custodian of private keys. Keychains generate, store keys and sign transactions. Users can use Warden's intent configurator to configure their own keychain and application security setting, putting them in control of defining their own spectrum of custody: from holding their own keys, to sharding their keys and splitting them between users and enterprises, to delegating custody to an ISO-compliant, SOC-audited digital asset custodian, through to leveraging the latest in distributed key management protocols. Warden is also exploring a new variant of multisig, composed of different keychains and custodial models collaborating via user-driven intents.

Each OApp has an intent configurator. This can be accessed via a GUI or over CLI, and lets a user interface and configure intents with their chosen keychain. Intents are a set of user-supplied conditions under which a keychain signs

a transaction. They are predicates over transactional data and external inputs; an arbitrary on-chain code evaluated at runtime by the settlement layer that enforces the terms of an interaction in a transparent, human-readable form. The OApps modular security stack embeds user intents directly into the applications security architecture.

Today there is no standard mechanism to express, compose and parse intents, similar to a time before SQL was created, when querying databases was tricky. In order to enable arbitrary use cases from several OApps, we unified the syntax with which users can express their intents and configure their keychain. This embedded, intent specific language ("ISL") standardises interface-, transmission semantics and execution behaviours. It's a composable, extensive, declarative, human-readable, English-like language purpose built so users can configure and preview the transaction conditions for their keychains.

Because web3 users have complicated interaction schemes with assets across multiple chains and wallets, we incorporated feeds into our language. Warden Protocol, provides a fully integrated, general purpose price, enshrined oracle built into the chain that leverages protocol security to provide guaranteed per block price updates at millisecond refresh rates. Using the Warden Protocols validator set, the oracle ties price updates to consensus where each update requires a two-third participation from validators to be posted on the chain. Thanks to this enshrined oracle and our ISL, user use cases like scheduled payments, subscriptions, rebalancing indexes, DCAs and swap streaming or time-based trading strategies become simple and straightforward to express.

Keychains sign transactions only when a user's intents are satisfied. Warden Protocol has an immutable on-chain, intent engine that acts as a gatekeeper. In order to prioritise security, minimise attack surface and focus on first principles, the intent engine is designed as a functional program with a boolean predicate. Its sole purpose is to determine the outcome of an intent verification, returning only either true or false. It is only when a user's supplied intents are immutably respected that a keychain can modify a user's state. Each time a transaction arrives in the mempool, a Warden Protocol validator runs the transaction against the set of user-created intents to verify if they are met. It is only when an intent validates the transactions, that the Warden Protocol validators include it in a block on the chain.

Omnichain Interoperability

Currently, there are over 1.000 different chains and over 120 different L1s - each with their own protocol standards, consensus mechanisms, hashing algorithms and more. These ecosystems are mostly siloed and inaccessible, causing a fragmentation of liquidity, services and users. In comparison, OApps were designed for unprecedented cross-chain interoperability, and specifically engineered to overcome isolated environments. They are designed to sequence actions such as transfers, swaps and liquidity provisions cross-chain, with the goal of abstracting away network boundaries.

Collectively, many OApps form an application mesh topology, where connections are universal and persistent. New OApp instances are not isolated applications; they join an ecosystem of connected OApps that can all exchange information. Users can seamlessly swap their tokens across supported chains and bridge to 64 connected chains.

Warden Protocol supports cross-chain transactions with Ethereum and other IBC-enabled chains, and any ECDSA or EDDSA-based chain supported by a keychain (e.g. Bitcoin). Tokens launched on Warden Protocol can be exposed to multiple networks by default - they can scale cross-chain without needed re-deployment. OApp developers can move beyond single chain experiences and stop compromising on audience reach.

Chain Abstraction

Spaces are users' gateways to the mesh network of OApps and any other blockchain. They are identity-abstracted, privacy-preserved, account-aggregated Warden addresses with which users can interact with OApps or entirely separate web3 applications. By using key identifiers, each OApp user can receive an infinite number of remote addresses on every ECDSA-/EDDSA based blockchain. This research and development effort introduces a novel programming primitive, where users could use their space to log into applications from every chain as a full alternative to most existing wallet options.

Traditional smart contract applications only target users of a single chain. Thanks to spaces, OApps can sign transactions and general messages targeted for any destination chain. They can sign transactions executed on other blockchains (e.g. writing to other chains), and thanks to a direct network integration of listeners can query data and events from other chains (e.g. reading from other chains). For example, a particular application chain may need to know the current price of the token of a second chain. These cross-chain queries create a multitude of novel OApp use cases in bridging, multichain DEXs, enabling non-smart contract enabled chains and many others.

For instance, traditionally web3 users defaulted to CEXs for multichain experiences because of the complexity of bridging, the difficulty of handling separate wallets and managing gas. An OApp developer could use a space with a deposit address on all chains, thereby eliminating the need to bridge assets to a single chain to swap, or to bridge to where assets have most liquidity.

Bridging assets is another potential use case. In some instances - for example when an underlying asset is too expensive to slow - like in the case of Ethereum or Bitcoin, there can be value to wrapping an asset. An OApp developer

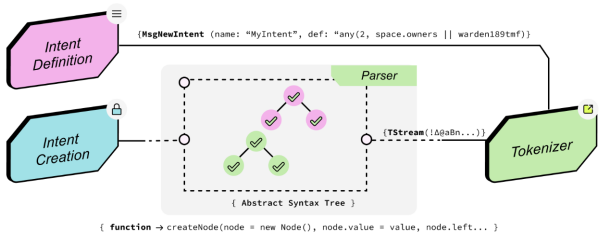

Fig. 2. Creating an intent on Warden Protocol involves an intent definition, tokenization, parsing and creating an abstract syntax tree

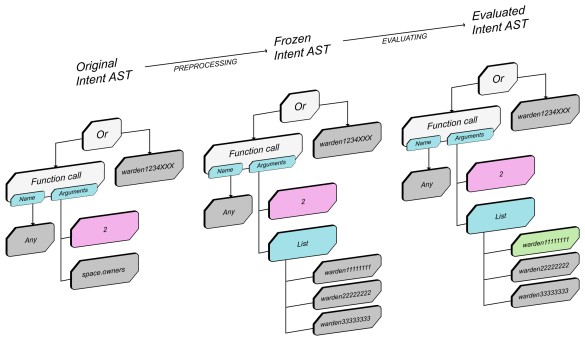

Fig. 3. Illustrative AST for a trivial intent definition, where grey rectangles denote variable identifiers, white rectangles denote language operators and purple rectangles denote literal values

could build a token bridge using account aggregation, that keeps an account of deposits of a given token on an address and issues the respective token balance on another chain. Additionally, Warden will leverage a modular implementation of Axelar's cross-chain communication protocol, to enable rapid, programmable bridging of assets.

Spaces also enable non-smart contract chains. An OApp can control externally owned accounts on non-smart contract-enabled chains, like Bitcoin, Dogecoin or Ripple. An OApp developer could build a DEX for Bitcoin Ordinals that handles deposits and executes swaps when two users agree to trade BTC for the Ordinal or any other BRC20 token.

Transaction Lifecycle

In this section, we provide an overview of how intents are created, and how signatures are generated. We show how Alice creates an intent, and how Bob can sign a transaction. Both users assumed OApp users with funded spaces. Each of them evaluated different security settings and picked a governance-verified, on-chain registered keychain to generate an ECDSA or EDDSA private key on the secp256k1 or Ed25519 elliptic curve. A node is any blockchain node in the Warden Protocol network. Nodes are responsible for routing keychain requests and responses. A client is any software interacting with the Warden Protocol that runs on the end users' machine (e.g. an OApp).

After generating a key, each OApp user can use a simple intent configurator to create intents and assign them to application keys. Intent definitions are expressions written with our intent specific language, ISL, that return true or false. They don't add additional elements to transactions. *MsgNewIntent* captures the intent definition, its creator and its name.

When a user creates a new intent, a tokenizer breaks down the intent definition into tokens, where each token represents the smallest atomic element of the intent specific language. A recursive descent parser validates that the intent definition is syntactically valid. It reads the resulting token stream and creates an abstract syntax tree (AST). ASTs are an extensible, formal representation of the syntactic structure of an intent definition and are stored on-chain.

Alice used a GUI on an OApp to define a trivial intent. She wants an arbitrary Ethereum transaction to be signed by any two signers from a list of space owners, or from Bob. Her intent definition was tokenized, parsed resulting in an illustrative AST. It's stored immutably on Warden Protocol nodes. Using *MsgUpdateKey* Alice applies her intent to a key to govern the movement of her assets.

Alice initiates a transaction on an OApp. The client submits a *MsgNewSignatureRequest* to a node, containing the full unsigned Ethereum transaction and specifying the key to be used for the signature. An action is created on-chain, preserving Alice's original message along with an immutable, frozen, representation of Alice's intent AST at this particular time. Prior to constructing the immutable AST, blockchain modules can dynamically resolve and expand variables into fixed in-time values. In this example, space.owners is expanded into a list of three addresses that were owners of Alice's space at the time of her *MsgNewSignatureRequest*.

Each construct of the intent specific language maps to a sequence of verification steps of what must be present on-chain to satisfy Alice's intent definition. The variables of the frozen intent AST are resolved and evaluated deterministically on-chain by all the Warden Protocol nodes, thereby inheriting the protocol security model where one third of validating power must agree on the outcome of the evaluation. When the final AST is positively evaluated *MsgNewSignatureRequest* is executed.

When *MsgNewSignatureRequest* is executed, the keychain responsible for the key sees the pending signature request and will pick it up. The keychain signs the data using the private key and sends a *MsgFulfillSignatureRequest* to the node containing the signature. During this process, Alice's' private key is never exposed to the Warden Protocol.

Using a simple example, we described the process of intent creation and signature generation in the Warden

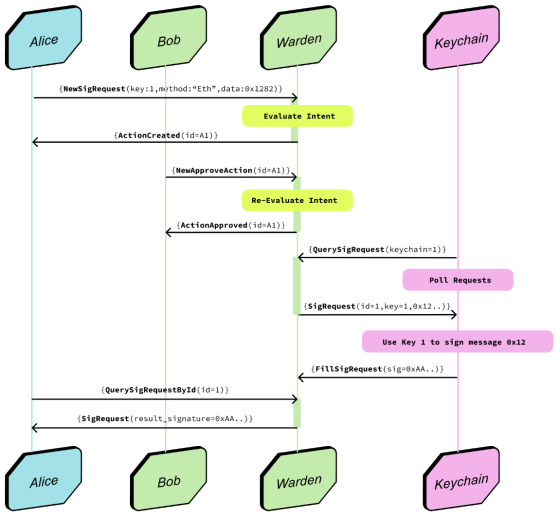

Fig. 4. The signature generation process on the Warden Protocol consists of intent evaluation and keychain operations

Protocol. We purposefully omitted on-chain economics and fees, instead referring to our tokenomics paper. This includes distinguishing between variable protocol fees per type of protocol operation, from simple transactions to keychain operators or towards more advanced intents.

Contributor Friendliness

A core principle of Warden Protocol is contributor friendliness. Contributions can come from developers wishing to support Warden Protocol or application developers wanting to deploy their own OApps.

Despite the novel features of OApps like modular security, omnichain interoperability and chain abstraction, they are straightforward to build. Warden Protocol comes with two smart contract execution engines: w-WASM and w-EVM.

w-WASM is the Turing complete smart contract platform of the Warden Protocol ecosystem. Built on CosmWasm, it empowers application developers to create high-performing, provably secure, multi-chain OApps. The platform comes with a robust build and test environment, alongside an integrated development environment. Leveraging IBC and its permissionless relaying of data packets, OApps can execute transactions across different chains. All contract code is designed to be agnostic to the details of the underlying chain. CosmWasm is industry-recognized for its proven security, effectively neutralising common web3 attack vectors like reentrancy attacks resulting from concurrent execution of contract code, short address parameter attacks or exploits stemming from overflow vulnerabilities.

w-EVM provides an EVM-compatible experience, allowing application developers to bring on existing contracts to Warden Protocol and seamlessly go omnichain. Thanks to EVM-compatibility, Warden Protocol can support existing Ethereum applications and smart contracts without modification. Ethereum developers can use Solidity or Vyper, and all the tooling they are used to including familiar environments (e.g. Foundry, Hardhat or Remix), frontend libraries (e.g. ether.js, web3.js), node and RPC providers (e.g. Quicknode or Infura) or popular wallets and protocols like Metamask, Ledger or WalletConnect. Warden Protocol EVM-compatibility facilitates an easy migration of decentralised applications and assets, creating a seamless experience for those developers that want to transition from Ethereum to Warden Protocol.

Future work

The goal of this document is to provide an outlook of what the Warden Protocol is working on. We listed features already implemented as well as features being worked on. There is no doubt in the team's mind that new needs will surface, requirements change or initiatives proposed may be abandoned - that's the nature of working on disruptive, groundbreaking technologies. We are and have been open-sourced from the first day to pay tribute to this. We encourage readers to do their own research by following the progress and even viewing the status of Github items as they unfold in real time.