OPEN

WHITEPAPER

Abstract

OPEN: Making blockchain accessible

A core impediment to large scale adoption of blockchain platforms is their inability to make it easy for anyone to integrate with already present infrastructure. There are many ICOs and projects that attempt to provide payment gateways, or niche applications to specific problems, but these projects tend to be hardcoded for specific applications which forces more projects to attempt to reinvent the wheel.

The OPEN team has years of experience founding, growing, and developing mobile application companies, but instead of making another hardcode blockchain solution to address the issue of in-app payments, we have decided to create a platform that is designed to facilitate the connection between centralized and decentralized systems.

This white paper describes the core technologies that operate together within the OPEN platform to create a comprehensive approach to decentralized payment schemes for any developer application or software service. The architecture presented in this paper introduces a novel decentralized application payment gateway that integrates seamlessly with existing application backends. Additionally, OPEN's architecture publishes proofs of valid transactions that hold information surrounding the transaction, enabling limitless application level use cases. In a sense, the OPEN platform is the foundation for the first decentralized API for centralized applications.

Furthermore, OPEN is designed to make it easy for developers to integrate decentralized technologies into their current technology infrastructure. This is meant to facilitate mainstream adoption of decentralized networks.

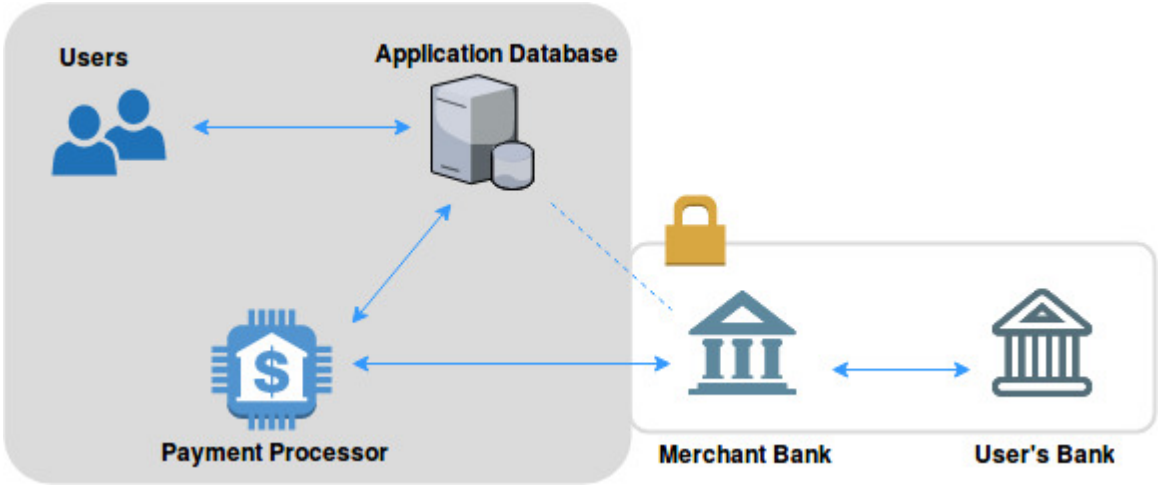
Table of Contents

Introduction

Blockchain technologies can provide incredible benefits to application developers. Technical limitations exist in providing decentralized solutions for required core aspects of how applications function in a conventional setting. Presently, there exists no easy solution for application developers to integrate blockchain technologies into their core infrastructure. Infrastructure to accommodate an application's specific pricing scheme and the ability to determine which users on a decentralized platform have met conditions of a specific pricing scheme, i.e paid or have enough in application currency, are lacking. Specifically, developers require a way in which they can integrate a decentralized payment gateway into their application's custom payment scheme that is able to accept any cryptocurrency and synchronize such transactions with their existing backend.

Moreover, centralized systems display incredibly inefficiencies that can easily be eliminated by incorporating aspects of a decentralized stack. Consider an example when using a centralized system to host or operate an application. Often times developers must go through a long auditing process by a centralized figure in which every single application update is scrutinized, often resulting in long backlogs for developers to publish their latest update containing a more efficient data structure or pricing model. Further issues with payment verification gateways exist as the current flow of in-app purchases go through countless processes and third-parties, adding unnecessary time and cost until reaching the developer's account. OPEN provides a solution to this problem by utilizing systems of sophisticated smart contracts as transaction and verification tools for software applications. Not only do payments come faster, they come at lower transaction costs with the added benefits of a trustless environment. This enables unique benefits for the application developer and it's users.

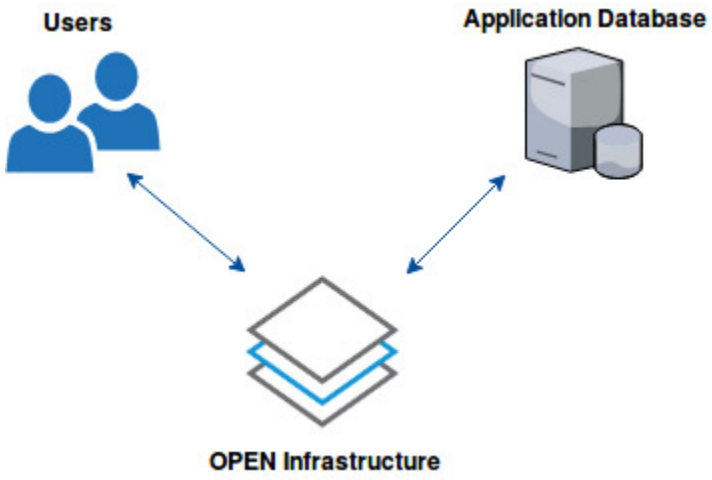
Current Payment Infrastructure



Grey area represents transactions occurring over the internet, while the solid-bordered area represents a private financial network. Solid arrows represent real time transactions, while dashed arrows are not.

As can be seen in the diagram above, the current in-app payment process requires numerous steps and a significant amount of infrastructure. This process often costs the developer upwards of 30% in fees if the application in question is a mobile app utilizing a traditional app store. These transactions can take 30-60 days before the remaining funds are settled and accessible by the developer. Furthermore, geolocation boundaries and regulations imposed by hosting platforms in order to drive their own profits, limits an application's success and the user's experience. Blockchain technology allows for this process to be significantly streamlined and democratized, thus allowing for substantial time and monetary savings.

OPEN Money Enabled Infrastructure

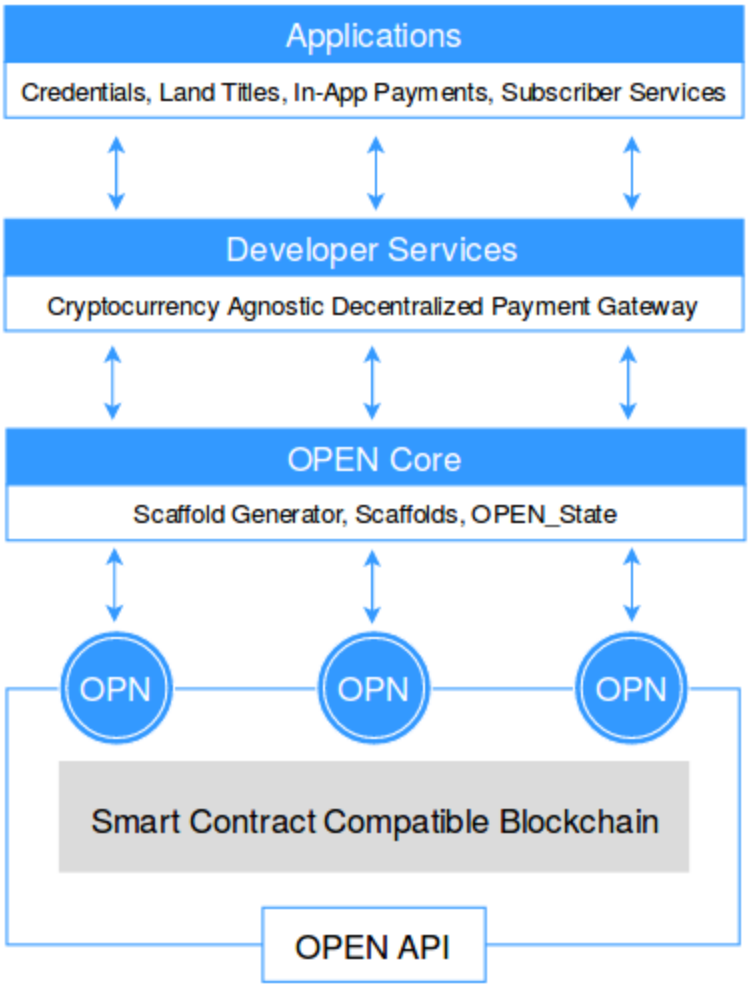


The diagram above shows the OPEN platform interacting on the blockchain to form a decentralized payment verification gateway that can accommodate the flexibility of numerous pricing schemas required by any conventional or unconventional applications. Further enabling a decentralized stacked solution that securely connects user payments to a centralized database, which is often required for application architecture in accepting cryptocurrency based payments. This architecture leverages blockchain for things it excels at: being an immutable data anchor, while utilizing existing centralized infrastructure to handle other application requirements, such as data storage and scalable computational power. It is not currently necessary, efficient, or realistic to decentralize everything within this stack and as a result OPEN has created a modular solution that can be easily integrated into existing applications, enabling many traditional applications to utilize blockchain architecture. OPEN creates a needed developer payment gateway solution that is fully self-contained and simple to implement on pre-existing infrastructure. Switching from a centralized to a decentralized payment gateway through this method can be as easy as using a function call, providing unique benefits that can be acquired with minimal integration resources.

Decentralized Approach

By leveraging the power of decentralization, it is possible to create a reliable ecosystem that enables any application developer to deploy customized payment schemes across a decentralized network and accept cryptocurrencies as the method of payment. Further enabling the reliability of information, eliminating single points of failure, and ensuring a level of data integrity that is impossible to replicate in centralized application architectures. By allowing developers to anchor their existing centralized database to the blockchain, OPEN provides the ability for any software developer to plug into blockchain technologies without drastically altering their existing backend processes. Also enabling developers to deploy pre-existing successful applications immediately onto a blockchain where the OPEN infrastructure provides a turnkey framework a developer and application would need in place. This furthers the possibility of an unprecedented level of interoperability and an entire ecosystem that enables developers to build a plethora of digital products without a coercive intermediary charging significant fees and enforcing limiting restrictions.

OPEN has created a blockchain agnostic platform with a comprehensive API structured to specifically create ease of access and integration into pre-existing application infrastructure. The OPEN API abstracts away the complexities of the OPEN platform's decentralized architecture by introducing an application layer across a decentralized platform developers can connect to with their existing codebases. This makes it easy for a developer to switch out centralized functions, like a payment gateway or a hosting environment, for functions that utilize OPEN's decentralized application platform. At the moment, the functionality that OPEN seeks to provide with its platform and API primarily deals with payment gateways. The OPEN API will leverage functionalities already present in external libraries like web3.js and ETHJS to develop a secure way to interact with the blockchain. With this implementation, OPEN will unlock the potential within blockchain technology in order to fundamentally alter the way applications are built, hosted, trusted and monetized.



High level overview of the OPEN Platform model. Different layers from a smart contract compatible blockchain to the OPEN Core where our unique Scaffold operate within. Developer services include the acceptance of a variety of cryptocurrencies on any application payment scheme. Developer applications utilizing the developer scaffold are on top of the developer services within the OPEN ecosystem. Aspects of each and their interactions with applications, users and developers are discussed in more detail below.

Architectural Design Approach

Modularity is a focal point of the OPEN implementation. The decentralized backend is designed to be as flexible as possible so that developers interact with an API that abstracts away the technical details of utilizing the blockchain. The OPEN platform utilizes several different components of blockchain architecture, along with a Scaffold protocol which acts as payment infrastructure and verification scheme for a specific application, and an OPEN_State, which acts as the verified output that contains within it the desired result of the transaction or state of any payment scheme for a user of the application. At a high level the OPEN API serves as the connection between these components on the blockchain and the developer's existing infrastructure.

OPEN included Scaffolds as template smart contracts that can be created to the specifications required by a developer, instantiated through a simple API call. OPEN utilizes customizable Scaffolds as a component to allow developers the unique flexibility over pricing structures and types as well as transactions required by a broad array of applications. Additionally, because OPEN establishes one point in the architecture that the user interacts with via the OPEN API, it is simple to switch out and replace Scaffolds as the application builds new features. A Scaffold can just as easily represent a subscription model payment scheme, an enterprise software pricing scheme, or an in-game currency pricing scheme. The beauty in this approach comes from the decentralized manner in which a user can pay into a Scaffold and subsequently be verified on the blockchain using the OPEN API. Not only does this increase the speed of transactions, it gives developers, applications and users a decentralized payment environment by using the OPEN API to verify that a user payment is on the blockchain.

Scaffolds enable our technology to work with any blockchain that supports smart contracts. The elegance behind Scaffolds is not only their ability to ensure a seamless workflow by responding automatically to any user payment interaction, but also their ability to simplify the process of tweaking an application. A Scaffold's specific role, technical implementation, and interaction within the ecosystem will be discussed heavily in the following pages.

The OPENToken is a utility token that allows a developer to activate a Scaffold after it has been created. By making the token's function developer-centric, the introduction of a new token is not a barrier to adoption by the end user. This is essential to enabling OPEN's vision of becoming a catalyst for blockchain adoption and integration by mainstream software developers. The developer is responsible for purchasing OPENTokens and using them to create Scaffolds to power their applications.

This paper uses "state" to reference the specific information that gives context to a user on an application. As an example, the "state" of a user on a mobile gaming application could encompass an amount of in-app currency a user could have, whether or not they have a premium plan, or the assets that they have purchased. By viewing information through this lens, it simplifies the translation of a user's actions into data on the blockchain. Instead of representing attributes as a multitude of different tokens, which congests the ecosystem and adds a layer of superfluous complexity, it is possible to attach data to a transaction that uses an

on-chain smart contract to simplify changing values of one of these states. In this way, any operation a user makes in an application can be represented as a transaction on the OPEN blockchain architecture. This maintains an efficient, linear, and most importantly, secure workflow. Expanding a little on a potential use case above, an example of an operation is the application user desiring an additional amount of in-app currency. Instead of representing the currency or gems as some sort of individual token, creating an excessive amount of unique tokens, this method represent the number of gems a user has as a variable in the state of the user. By doing so, the user can simply pay into the Scaffold, which will publish a new state for that user onto the blockchain, signifying that indeed the user has paid and has a right to that new amount of currency. Not only is this experience simple and intuitive, it also protects against potentially malfeasant acts by dishonest users.

The OPEN API gives developers the ability to create customized Scaffolds from Scaffold templates that can correspond to a variety of payment schemes. The OPEN API call is the initial point of interaction for a developer to create a Scaffold that conforms to the developer or application specific requirements. Having a high level interaction scheme makes it easy for developers to change or update certain aspects of their application. They can simply close-out their original Scaffold and make a new Scaffold with updated specifications through a simple and familiar API call. The significance of this system is that it enables developers to change their application scheme and through the OPEN API without having to redeploy the entire payment structure.

Key Terms

OPENWallet:
An OPENWallet is the storage unit for the user's cryptocurrencies (e.g. ETH, BTC, etc.) as well as the OPEN_States the user has purchased through the OPEN API. These OPEN_States provide payment verification for the user on different applications. The OPENWallet provides an intuitive user interface that abstracts away having to learn how a blockchain works, and maintains the same user experience as current centralized implementations.

Developer Wallet:
A developer wallet is the wallet (address) where the Scaffold transfers the funds that the user has sent to it for an OPEN_State (see diagram below). While OPEN allows developers to use any wallet, it is recommended that the developer use the OPENWallet, as it leverages the OmiseGO SDK, which has a built-in decentralized cryptocurrency exchange and great community support. Through the OmiseGo Wallet SDK, the developer can automatically convert any Scaffold transfer into any cryptocurrency they desire or fiat.

OPENToken:
An ERC20 token that will be distributed in the OPEN ICO and allows for developers to create Scaffolds in the OPEN ecosystem. As the OPEN platform evolves the OPENToken could gain additional utility such as giving users that pay in OPENToken a discount for the the services purchased.

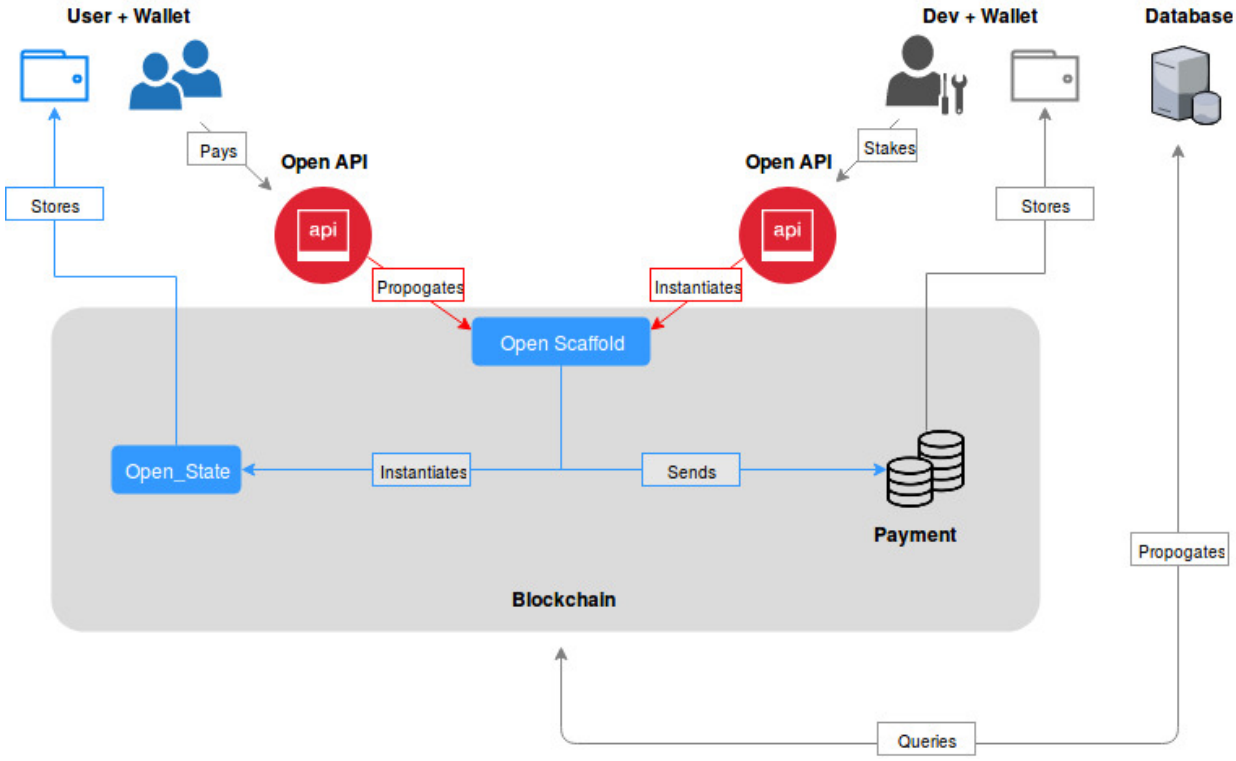
OPEN_State:

A smart contract that stores a cryptographically hashed state of a user on an application (implementations for what the OPEN_State can represent are discussed in the "Future Proofing" section). An application's current database state at the time of purchase can be made private using a cryptographic hash and then anchored to the blockchain. The OPEN_State will include information to allow for verification that the Open_State belongs to the specific user and to a Scaffold. This information includes the user's OpenWallet address and the Scaffold's address.

Scaffold:
A template smart contract that is created by a developer and includes application specific information, such as the pricing scheme or number of gems owned in the network. The Scaffold is instantiated by the developer when the developer pays into a Scaffold Creator. The Scaffold becomes active when it is staked with enough OPENTokens. The Scaffold staking process entails tying up a certain number of OPENTokens in the Scaffold in order for it to run. These OPENTokens are required to keep the Scaffold active, but will be returned to the developer when they deactivate the Scaffold. The Scaffold is responsible for creating the OPEN_States for the developers application. As such, the Scaffold needs a specific pricing scheme for the OPEN_State (e.g. $10 for 10 gems), and it uses this pricing scheme as a way to check if it is being paid enough. The Scaffold is also responsible for transferring the payment given to it for the OPEN_State into the developer's wallet. A Scaffold requires the developer's address for transfers, a pricing scheme, and access to a pricing oracle, but can also can hold any number of additional variables and data.

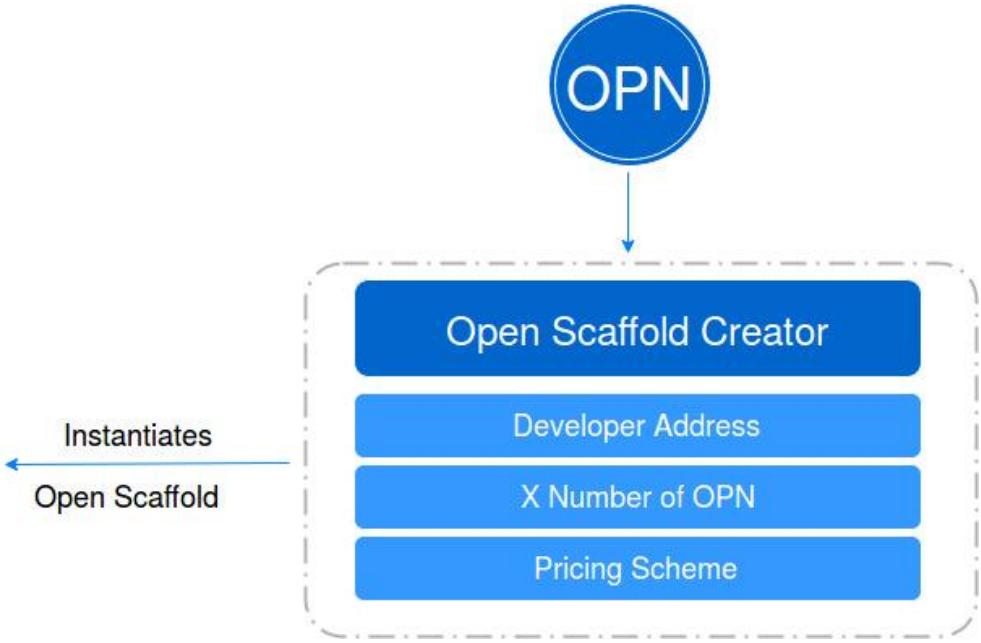
Scaffold Creator:
The API that OPEN provides to create Scaffolds for developers through a simple API call. The Scaffold Creator can create Scaffolds with ETH, but for the Scaffolds to become active it is necessary to stake them with OPENTokens.

Overview



High level overview of the workflow for both developers and application users through the OPEN architecture, as well as where the API fits in. Details on the most important interactions below.

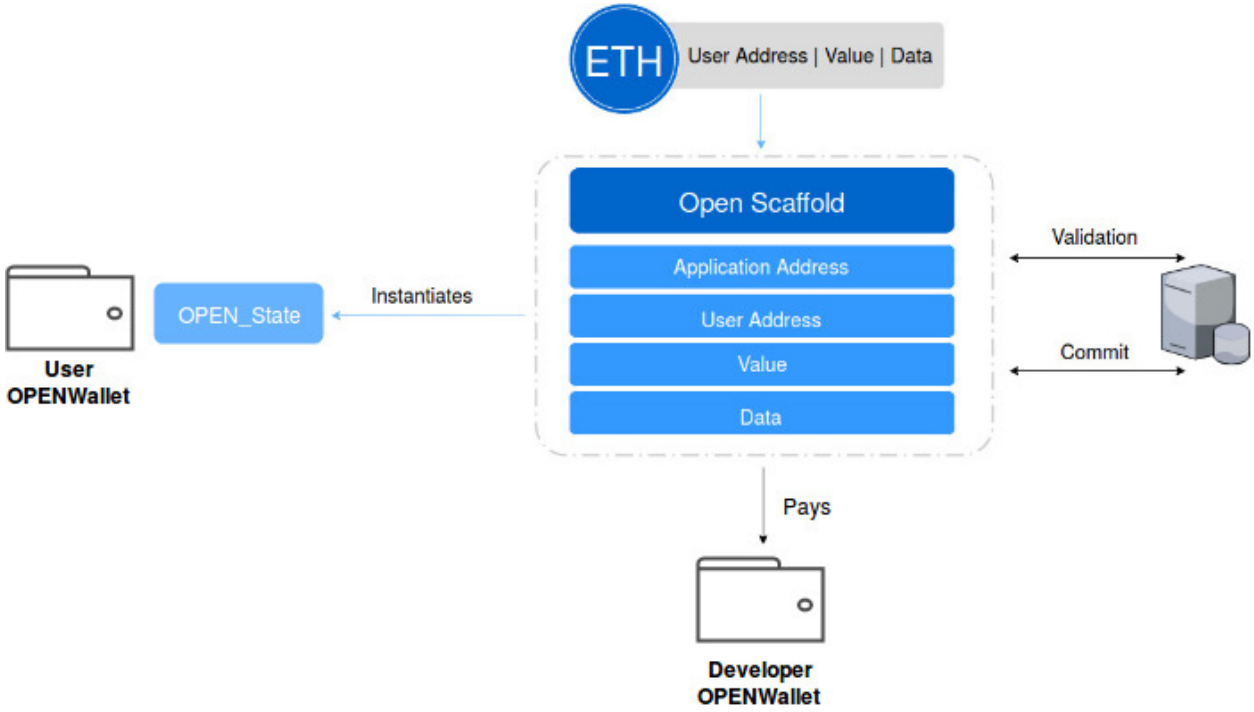
Scaffold Creation by Developer



The OPEN API will have a built-in functionality that gives developers the ability to generate new Scaffolds set to the developer's address as the owner. By utilizing OPENTokens to initialize the Scaffold, the developer can create a unique Scaffold in the blockchain ecosystem.

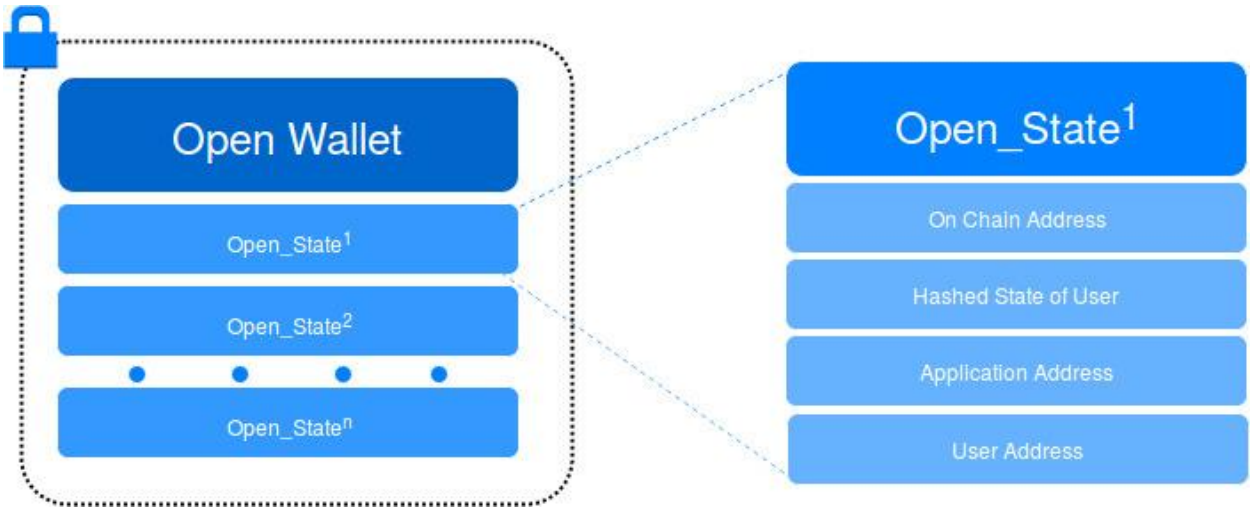
The second template contract is the Scaffold. As stated above, the Scaffold is created through the API with OPENTokens. The Scaffold itself accepts a transaction that represents the payment given by the end user, as well as some data representing the action that the user wishes to take. This is propagated by the API. The Scaffold then creates an OPEN_State that represents the result of executing the specified action, and stores it in a decentralized manner in order to reduce overhead faced by the developer and to leverage the data integrity and rigor offered by blockchains. The OPEN_State is kept in a the user's OPENWallet to intuitively organize data and increase execution efficiency by reducing querying time. From the user's perspective, the user wallet could be a user interface similar to one in function enforced by Google Play or the Apple App Store. The final aspect of the Scaffold is that once it verifies that a payment amount is correct, it does not store any cryptocurrency, rather it automatically transfers the funds to the developer's wallet, allowing for instant liquidity and protection from volatility for the application and developer. This nearly instantaneous transfer from the customer's wallet to the developer's is a dramatic improvement over the current in-app payment system, which can take as long as 30-60 days to credit purchases to a developer's account, while mitigating volatility issues that may occur in such period when dealing with cryptocurrencies.

User Interacts with Scaffold



This diagram shows the user sending a payment of ETH to the developer's Scaffold through the OPEN API. Since the user's payment could be processed via the OmiseGo (OMG) decentralized exchange hub, it is possible for the user to pay in a variety of cryptocurrencies and have that payment converted into ETH to pay the Scaffold. A developer's database management system can then propagate changes by checking the OPEN_State for the user address and update the relevant state by decrypting the information with a key given to it by the developer. Simultaneously, ETH payment will be sent from the Scaffold to the developer wallet (which can be selected by the developer) where they can choose to retain the amount, transfer immediately to USD, or convert to some other cryptocurrency the developer prefers.

OPENWallet + OPENState Scheme



The OPENWallet represents a collection of a user's different states for each application they are involved in, these states are represented as unique smart contracts. Each state, called the OPEN_State, is a smart contract on the Ethereum network that anchors the data on the blockchain and allows for connections and verifications on the developer's existing backend. This is created by feeding in data to the Scaffold. Modularization is key to having flexibility and iterative capabilities.

Applications of Scaffolds and OPEN_States

The OPEN_States and Scaffolds that comprise the OPEN platform enable the application of countless digital services. Below is a an example implementation for OPEN_States with regards to a video streaming application, but other use cases include: adding in-app currencies, creating software licenses, distributing coupons, and providing partial administrator access to databases.

Permissions for video streaming (permissions for online use):

Through the OPEN platform, any video publisher, for example Jon, who would like to monetize their content can do so without worrying about an elaborate payment backend. In this structure, a Scaffold would be initiated by Jon and his application to allow a user to gain access to Season 1 of a show he is producing. He would initialize this Scaffold by inputting the cost of access to this streaming service and a variable value indicating if the user has access.

When the user pays into this Scaffold, the amount of money he paid is validated through an oracle to see if it matches the amount needed as specified by Jon. If it does, an OPEN_State is created that is sent to the purchaser's wallet, providing access to the purchaser. At this point, all Jon's database has to do is query the OPEN_State to see if that state exists, and if it does, edit its own database to reflect what the OPEN_State signals (for example that the purchaser can watch Season 1). With that change made, the purchaser has access to the videos and can view them.

The efficiency of this system is orders of magnitudes better than existing payment gateways. Additionally, the increased efficiency results in much fewer middlemen to extract exorbitant fees that prevent Jon from capturing the true value of his content. OPEN's value is clear.

Developer Wallet and OPENToken

The developer's wallet can be any online wallet that is capable of receiving cryptocurrencies and holding OPENToken. While OPEN allows developers to use any wallet, it is recommended that they use the OPENWallet as it leverages the OmiseGO SDK which has great community support and a built-in decentralized exchange. The OmiseGo SDK is publicly available for use and easily implemented. It is important to note that OPENWallet's architecture is however generalizable over any wallet for widespread use.

The developer's wallet is used in three different scenarios. The first is when a developer wishes to create a new Scaffold by paying into the Scaffold Creator. This transaction would include a staked payment in OPENTokens along with some data and would be propagated by the OPEN API to the Scaffold Creator, effectively abstracting away underlying blockchain technology.

The second scenario in which the developer's wallet is used is when the developer receives payments from the user. As mentioned above, this transaction would be sent by the Scaffold to which the user is paying. Once the Scaffold validates that the amount paid is sufficient for the action requested, it instantiates an OPEN_State and immediately forwards payment to the developer's Wallet's address. OPEN utilizes automatic liquidation to reduce exposure to volatility. Volatility risk is important to address in a decentralized developer ecosystem because cryptocurrencies can be extremely volatile and the value they hold is subject to unforeseeable fluctuations. This represents an obstacle to blockchain adoption, so in order to mitigate this, OPEN minimizes the time cryptocurrency is locked up in its decentralized architecture. By immediately forwarding all payments to the developer's wallet at the time of the transaction, the developer can then choose if they would like to hold the cryptocurrency as-is, or if they would like to convert it into another cryptocurrency or fiat. This allows for increased functionality and flexibility for the developer.

The third scenario is when the developer's wallet would be used is in maintaining an account of the transactions that go through the application Scaffolds. Since transactions are public to everyone on the blockchain, a wallet can be built with functionality to analyze all the interactions users have had with a Scaffold, and from there provide the developer with valuable user statistics. For example, a developer can learn how many users are interacting with Scaffolds, how frequently they interact, average transaction size, revenue data, and more. This key functionality makes the OPEN ecosystem competitive with current centralized app stores.

OPENToken

The OPENToken is used in a variety of ways throughout the OPEN ecosystem, with the first being Scaffold staking.. At a high level, a developer can create a Scaffold by using the OPEN API, but for that Scaffold to be used, the developer must stake a certain amount of OPENToken with it. When the developer wants to retire a Scaffold, they may close it out and receive their staked OPENTokens back.

To mitigate the rapid activation and deactivation of Scaffolds, OPENTokens that are staked in a Scaffold will be locked-up for a specific time period. After the lock-up period, the developer is free to close-out their Scaffold or keep the OPENTokens in the Scaffold. So essentially the staking mechanism deters malevolent behavior.

Additionally, 3% of every transaction is kept as a type of network gas fee. That allotment automatically is sent to a development pool from which the OPEN Team is partitioned some, but most is used to incentivize developers to come onto the OPEN platform by giving them a certain amount of value of tokens. This creates an automatic feedback loop to boost our platform's growth by giving value back to the developers. The feedback loop looks roughly like:

1. By being developer centric and using our ICO proceedings as incentives, we can bring developers onto the platform.
2. By integrating their applications, the developers bring payments.
3. Each payment gives 3% to the growing developer pool.
4. Funds from the developer pool are used to incentivize more developers to come onto our platform, thus solving the two sided market dynamic problem.

As a part of our future considerations written below, we will develop a host of other services to make it even easier for traditional developers to integrate blockchain technology into their stack. An example of such a service would be the OPEN Node service - a tool that developers can use to watch the blockchain if they don't want to run a node themselves or rely on a third party node. Such a service would take payment only in OPEN token and further tie together the ecosystem.

Finally, the OPENToken can provide utility to the users if application Scaffolds are implemented to give discounts or currency bonuses to individuals that hold OPENTokens in their OPENWallet, or pay the Scaffolds in OPENTokens. This added utility would provide an incentive for users to purchase and utilize cryptocurrencies such as ETH and BTC and carry OPENTokens over fiat currencies to support a conventional app in a decentralized environment.

Third Party Dependencies

The OPEN platform is built with interoperability and modularity so that current components can easily be swapped as better iterations of the technology surface in the future. This

modularity gives us the necessary flexibility to be able to adapt to advances in the space, while keeping security independent of the underlying protocol. In this way, OPEN will truly be able to create an application network or layer appealing to the entire industry. In designing our architecture great thought was given to which 3rd party platforms OPEN can currently best leverage. Highlighted below is a list of 3rd party dependencies that would aid the OPEN platform, both extending the platform's capabilities and focus.

OmiseGo White-Label Wallet SDK

OPEN uses the OmiseGO wallet SDK to connect to the OmiseGO decentralized cryptocurrency exchange (DEX). This provides users with the ability to pay into Scaffolds (smart contracts that act as payment gateways) with a variety of different cryptocurrencies. Moreover, using the OmiseGO DEX, developers can convert the ETH that they receive from the users into other cryptocurrencies or fiat.

https://omg.omise.co/

Tendermint

While Tendermint is not necessary for the implementation of OPEN, its use comes from its high throughput, low transaction fees, and its ability to run the Ethereum Virtual Machine (EVM) via Ethermint. Tendermint provides the ability to replicate state machines (e.g. the EVM) on its blockchain, which allows it to utilize smart contracts written in Solidity, making the transition from Ethereum to Ethermint a simple operation that provides a huge increase in throughput and reduces costs. These reduced transaction costs would translate into a lower fee structure for developers accepting payments through OPEN, and the enhanced throughput would make it faster to process transactions on-chain. This helps to overcome adoption and scalability issues that arise from accepting many payments at one time. It would make sense to utilize Ethereum for the PoC, Tendermint for increased throughput to test the OPEN platform as it scales, and eventually switch back to the Ethereum network once more of the Ethereum scaling roadmap has been implemented. The switch back to Ethereum, once it implements Proof-of-Stake, would be dependent on which platform provided the best transaction times and fees.

https://tendermint.com/

GNOSIS, Oraclize, ChainLink

Since the price of cryptocurrencies is known to be volatile it makes sense to base the prices within the Scaffolds on fiat currency. This prevents users from being upset if the prices increase and they have to pay more to make purchases, and it prevents developers from being upset if the prices decrease and they get less money from the same purchases. To solve for this issue of

volatility, it is necessary to have a trusted source of data for the price of various cryptocurrencies. This trusted source can use the correct, real-time price in order to validate the user's transaction. To do this, an oracle is needed to query the API. There are several potential oracle networks that can be used by the OPEN API. These include the centralized oracle from Gnosis, the oracle that is provided by Oraclize, and the oracle network provided by ChainLink. The choice of the oracle will depend on the speed at which the oracle can get the price data, and the security of the oracle itself. An oracle that is unable to achieve fast throughput would be a bottleneck for the transactions and could lead to inaccurate pricing data, and an oracle that lacks data security poses an obvious threat.

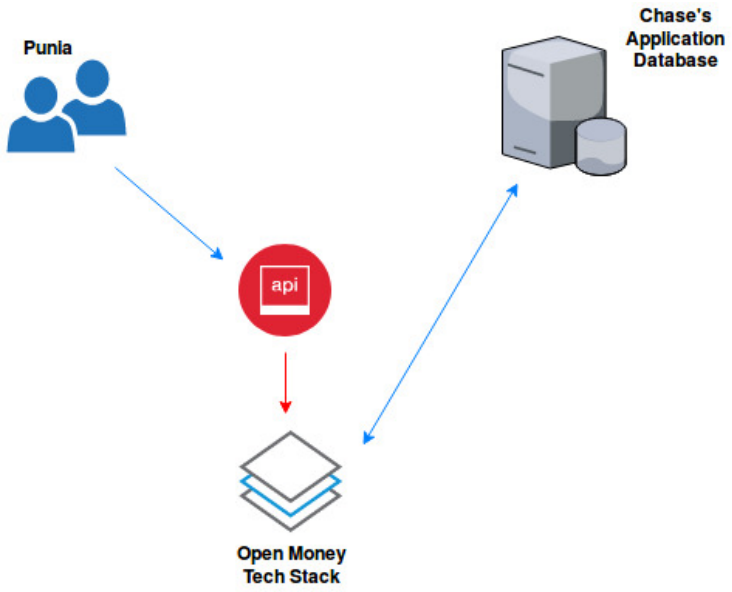
https://gnosis.pm/
http://www.oraclize.it/
https://link.smartcontract.com/#chainlink

Web3, ETHJS

OPEN intends to use pre-existing libraries like web3 and ETHJS to interact with the Scaffold Creator, the Scaffold, and the OPEN_State Tokens. This is done both to prevent errors that arise from rolling your own connections, and to inspire developers to participate in the process of helping OPEN to continue to innovate. Web3.js is a Javascript API to Ethereum and allows interactions via JSON by providing the Scaffold and Scaffold smart contracts with JSON RPC. ETHJS is a javascript library that is based on web3.js, but features a more light-weight implementation for interacting with Ethereum.

https://github.com/ethereum/web3.js/
https://github.com/ethjs/ethjs

A Simple Use Case Example



1. Chase is a developer that wants to make a payment method for his new video streaming application so that users can pay to become subscribers.

2. He decides to use the OPEN platform to manage his payment protocols.

3. He uses the OPEN API to send ETH to the Creator Scaffold and obtain a Scaffold that he can use for his application.

4. With the OPEN API and its user-friendly interface, Chase adds a pricing scheme that gives each user the ability to gain the Subscriber state on his database for $10.

5. Then Chase uses the OPEN API to fund this Scaffold by purchasing OPENTokens and staking a certain amount of them in the Scaffold.

6. When a user, Punia, decides to become a subscriber through OPEN, he creates an OPENWallet with the OPEN API. Punia then sends payment through the OPENWallet, and the OPENWallet then sends the payment, as well as Punia's desire to become a subscriber, to Chase's Scaffold.

7. The Scaffold will receive the payment and verify that Punia is paying enough money to become a subscriber through an oracle.

8. If Punia's payment is validated, then the Scaffold will create an OPEN_State instance with the necessary variables, and send it to Punia's OPENWallet.

9. The OPEN API can now let Chase's database know that there has been an update and can provide Chase with the ability to check the validity of the update by ensuring that it is recorded on the blockchain.

10. Chase receives the value in ETH from Punia and is able to convert it to fiat, or another cryptocurrency, immediately if he is using the OPEN wallet.

11. Now Punia can view a video or sign into Chase's application because he is a subscriber.
    Note: Punia can log into Chase's application through conventional means because his OPENWallet represents state change requests, not identities. A single OPENWallet could have the potential to update multiple different users by sending different hashed states.

Security Considerations

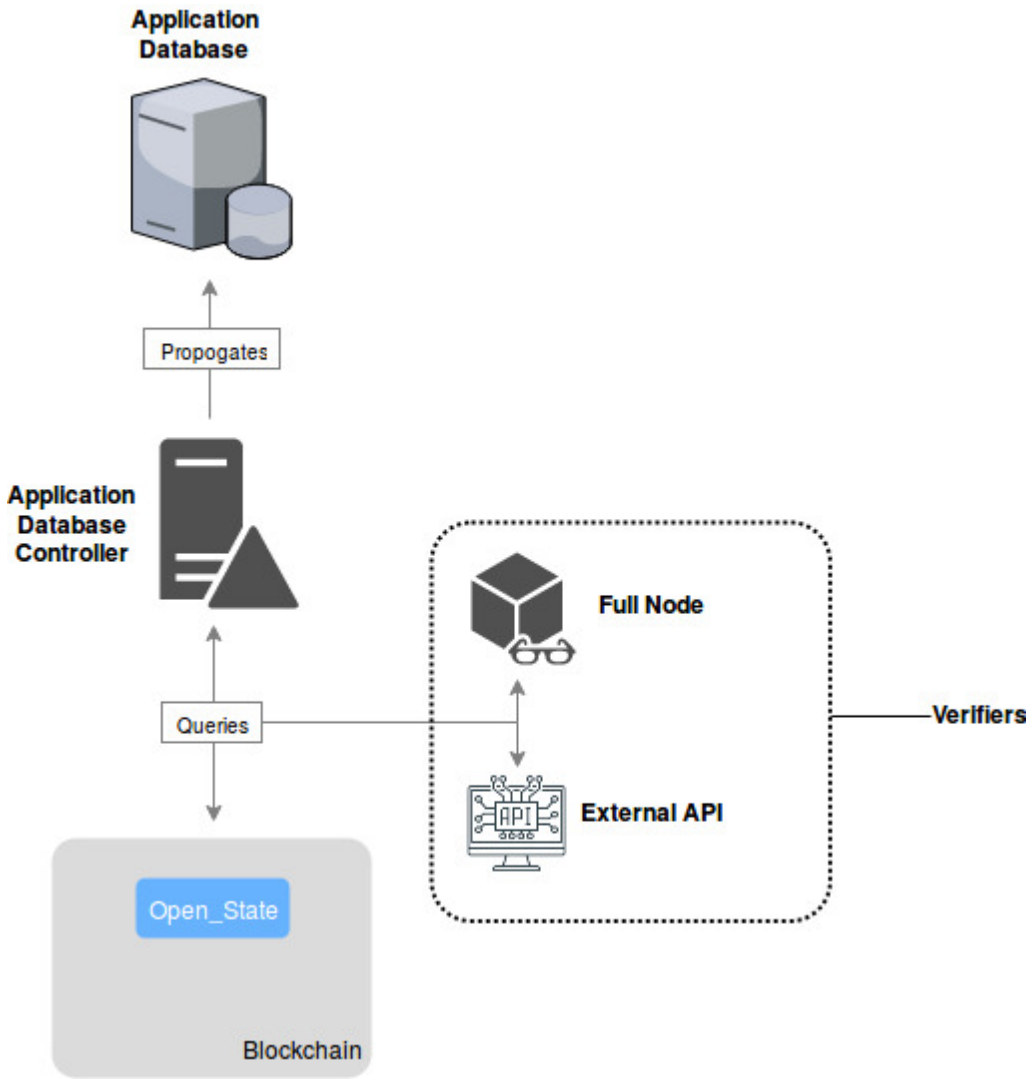
The OPEN platform incorporates various off-chain components that can pose security and implementation challenges. This section is dedicated to addressing the most pressing of these challenges and incorporates some proposed workarounds.

Database Synchronicity

One potential challenge for OPEN is the difficulty in determining whether the database is up to date with the decentralized ledger of OPEN_States (that can be viewed on the Ethereum Blockchain). To solve this issue, OPEN could force the blockchain to record every API call to the database and save it to an OPEN_State, but this would put a limit the number of “harmless” API calls (for example obtaining a username, or changing an email address) that applications must make on a daily basis. Therefore, it is not feasible to keep the database and the blockchain completely synchronized to maintain a realistic in practice solution for cost effectiveness.

Instead, OPEN has opted for a partially synchronous database. This partial synchronicity utilizes the knowledge that every database already has permissioned API calls and database changes that they run through a payment gateway. In this manner, it is only necessary to keep track of the number of payments and states or actions that the user has made that are permissioned.

Diagram of Database Synchronicity



This diagram demonstrates how an application's database recognizes a specific user's on-chain OPEN_State. Normally, the OPEN API can be implemented on the application controller to check the Scaffold for any new OPEN_States, and then propagate those changes. The application database then monitors the blockchain for OPEN_States being instantiated from relevant Scaffolds. Upon discovery, the database propagates the state change represented by the OPEN_State. The verifier can be one of a multitude of implementations, but it is meant to check that the OPEN_State was indeed created by the Scaffold of the developer and that this OPEN_State is the most recent OPEN_State that has been created. If the OPEN_State is verified, then the application controller makes the database change.

Note: For operations where a user is trying to purchase in-app currency, it is better to use the action operation instead of the state to prevent a malicious user from using in-app currency before a transaction is verified. Actions are explained in more detail further on.

This partial database synchronicity relies, in many ways, on a consistent and fast verification technique. The verification technique can be done one of several ways: using an external API that checks the Ethereum Blockchain for the OPEN_State, which can pose serious security and time issues, or employing a full node which would require the developer to be running a full node on the Ethereum network.

Since both of these options create issues with either security or with computational cost, it might make sense for OPEN to host its own API to query the blockchain and check for the existence of a specific OPEN_State.

Multiple OPEN_States

Another potential issue for OPEN is that an OPENWallet can and should have multiple OPEN_States for the same database, as a malicious user could request a previous OPEN_State to the database, at which point the database might accept it without accepting payment because it exists on the blockchain. Therefore, it is necessary for a method to verify the latest OPEN_State that the user has in a database. While it is trivial to distinguish which OPEN_State is the youngest by checking the blockchain, this takes a lot of time and allows for malicious users to attack the database by requesting multiple previous OPEN_States and forcing the database to run checks on them to verify they are the most recent.

Instead, a type of history implementation is employed for the user to keep track of all of their states through the wallet. This history could be a part of the OPEN API, on the database, or as a part of the developer's wallet, and would be structured as a hash chain where each new state depends on the hash of the previous state. This history allows for quick verification of user OPEN_States, and provides safety against DDoS attacks.

Multiple OPEN_States History Implementation

State vs. Action in the OPEN_State

While it is not a formal security consideration for the OPEN platform, it should be noted that the OPEN_State could also incorporate actions as well as states in its schema. An action would differ from a state in that an action would provide a manipulation of a current state, whereas a state would provide the state itself. The action and state schema of the OPEN_State should be

implemented for different reasons. Included below is a set of general guidelines for using an action versus a state, as well as potential security exploits for both.

States

The state should be reserved for permission-based applications, where a user cannot exploit the system by changing a state before they are granted their new state.

An example of a good use case would be changing a user from being a subscriber to a non-subscriber, or representing a set of finite states that a user can take on and cannot be spent by the user in the interim before it is verified.

However, in some applications, the state may be exploited. An example of this exploit would be a user who has 500 in app currencies, say gems for example, and requests a desired state of 550 gems, then before the database makes the change the user spends all 500 gems. Since the transaction is valid and the database should verify that the Open_State is correct and not in its history, the user will have stolen 500 gems from the application through the system. To prevent this, it is necessary to use an action.

Actions

The action should be reserved for applications where the user wants to add a specific amount of in-game money, or there exists many game states that a user can have, and the user can spend or trade these before the database can propagate the changes.

The potential exploit for states provides a great use case for actions. With an action, the user would specify that they want to add 50 gems to their account. Once they pay for the transaction they are given the ability to run this function an allotted number of times (in this case one). No matter what amount they spend, they will not be able to steal gems. The action will be placed in the database history after it is run, which means that there is also no exploit in regards to running an action more than its allotted number of times.

An action can be exploited by running bad actions for the developer, so the developer needs to include specific limitations on the arguments. The state is not as much of a problem because it codes for some set of actions that the developer has already preprogrammed.

OPEN Future Functionality

The infrastructure within the blockchain ecosystem is constantly changing and adapting. In order for OPEN to not only be a part of the future of blockchain, but to lead the future, OPEN has been designed to grow as the technology around blockchain evolves. As a platform that conventional application developers may depend on to utilize blockchain and cryptocurrencies within their applications, it's in OPEN's best interest to continuously stay relevant.

The first consideration that OPEN has made for the future is building the infrastructure on Ethereum and dealing with scalability issues and transaction fees by porting it's system to Tendermint at a later date. Since OPEN is written in Solidity (and eventually Viper) and runs on the Ethereum Virtual Machine (EVM), it is possible to transition from the Ethereum network to the Tendermint managed EVM known as Ethermint (via an ABCI). Not only will this allow OPEN to take advantage of the higher throughput and lower transaction fees provided by Tendermint's consensus algorithm, the transition will be frictionless because Ethermint runs the same EVM. OPEN could even port back to Ethereum once it has implemented the faster Proof-of-Stake consensus algorithm. By design, the OPEN platform is portable between numerous systems and not limited to a single blockchain protocol. In the future, the growth of OPEN may make it desirable to develop our own chain to improve upon efficiency, Scaffold features and transaction issues.

The modularity of OPEN provides an option for developers to not only include new off-chain systems seamlessly, it allows the OPEN community to provide safety upgrades and additions to the OPEN platform, thereby fostering innovation on and off-chain.

A concerted effort was made to ensure that developers and owners of Scaffolds would not be limited by the OPEN ecosystem when integrating with their pre-existing application infrastructure. Enabling significant flexibility for developers to choose which type of database structure, wallets, and which verification method they require. While OPEN provides developer wallets and verifiers, OPEN focuses on creating an interconnected application layer to act as the glue between various blockchain components. For instance, if a developer wants to use decentralized databases, it is a simple matter to include these kinds of database interactions in OPEN.

As a nimble approach, OPEN is meant to be changed and upgraded to include new features for new systems and its modular design allows for greater freedoms when opening the codebase up to the open-source community. These additions to the OPEN framework could be as simple as making Scaffold accounting easier or including a wallet verification check to increase the security within the system.

Next Steps

OPEN will open-source all tools of it's platform and implement a utility token to support the OPEN platform project. The OPEN platform is developing the backbone and future infrastructure required by software application developers to increase adoption and utility of cryptocurrencies. By pushing this platform forward, OPEN promotes mainstream blockchain adoption by providing required tools for applications and creating open-source tools for the community.

This project leverages the incredible amount of work being done to make protocol level improvements, while supporting the application layer that enables the use of this infrastructure by applications and their developers.