DataShell - A Peer-to-Peer Prototyping Environment for Web Apps with User-Owned Data Vaults

Alexander Praetorius, Nina Breznik

March 2023

Abstract

Moving data to the cloud solved users' early issues with backups and data loss, but today Big Tech companies own most of our data and control how it is used, leaving users vulnerable to privacy violations and data manipulation.("TrustVault: A Privacy-First Data Wallet for the European Blockchain Services Infrastructure" 2022)

As a response to this unhealthy dependency on Big Tech, Dat Ecosystem projects have been building a neutral common p2p infrastructure over the past 10 years. But cross-platform packaging, standards on how to build p2p apps, which give users sovereignity over their data, and educating developers how to build and bring p2p apps to the users still remains a challenge.

To address this problem, DataShell aims to create a new open standard for user-owned, GDPR compliant data vaults. Furthermore, we aim to provide developers with an open in-browser prototyping environment which they can use (or self-host) to build web apps based on the vault standard in order to create an open resilient app ecosystem that gives users full control, data portability and the ability to self-authenticate, verify data and permissionlessly combine it in new ways.

Background

In the early days of personal computers, users had all their data stored on their disks, and software applications were constantly reading and writing data to disk. At the time, computers were new and operating systems were unstable, causing disks to frequently crash. As a result, backing up data was difficult and time-consuming. Eventually, users welcomed the opportunity to move their data to the cloud, as it eliminated the need for them to take care of backups themselves. However, as the amount of data users produced increased, it became clear that hosting all this data on the cloud would not be cheap. Companies started offering free storage space in exchange for access to users' private data, which has become a valuable asset and primary source of income for these companies. Unfortunately, users have little control over where and how their data is stored and used online.

Today, machines and systems are much more stable and powerful, and peer-to-peer technologies have matured. As a result, users can securely back up and restore their data across devices with ease. However, what is still missing is an environment and set of standards for building web applications that decouple user data from the application itself. With these standards in place, developers can create services that give users full control over their data, enable self-authentication and with it also data portability.

DataShell

DataShell has many similarities to the Spritely Project, particularly in terms of technical approach. While Spritely is built within the context of the ActivityPub standards, DataShell is designed for use within the Dat ecosystem and protocols with its peer-to-peer network and unique data structures. Our focus is

on creating a browser-based platform that offers developers faster feedback loops, by allowing them to bring apps directly to users, rather than requiring downloads of unknown software to their computers. Additionally, we believe that focusing on the web is advantageous due to the broad availability of JavaScript, web, and front-end technologies, which are widely used and easily accessible for prototyping purposes.

Project roadmap

v0.1 - Core Environment includes the development of a core environment for a web-based operating system. This environment includes a web server component called the portal, a default WebApp OS using a web kernel, a web runtime patch component called the web kernel, a shell for user interaction, a data package manager for managing data, a vault mode for secure communication, and a sandboxing mechanism for running user code.("The webAppOS Architecture" 2019) These components provide the foundation for further development of the DataShell.

v0.2 - Basic UX/UI focuses on basic UX/UI development for the web-based operating system. The vault component is wireframed to include a responsive data vault UI, which allows users to view all installed apps/software, datasets created, and permissions set for specific datasets and portals. Basic iframe and fullscreen views are implemented, and UI elements are added to allow users to trigger actions available via the data package manager, such as changing permissions and adding/deleting datasets. The dpm OS component includes a nav bar with user account, offline/online indicator, shell controls, and data status indicator, as well as a content area with layout design.

Initial unstable release

v0.3 - Basic Editor App includes the development of a basic editor app for the web-based operating system. The editor app includes a text editor with no syntax highlighting and actions such as new, open, and save file. Additionally, the editor app allows for multiple editor tabs, providing a simple yet functional text editing experience.

v0.35 - dpm intermediate focuses on improving the data package manager (dpm) component. The dpm now allows the installation of complex apps with dependencies directly from the web server cache, streamlining the app installation process. Additionally, the dpm now allows the export of apps with multiple dependencies that are created via the text editor in the vault.

Pre-alpha release

v0.4 - Intermediate Sandboxing focuses on improving the sandboxing feature of the web-based operating system. The sandboxing mechanism is based on inter in-process sandbox object capability network (OCapN) and uses research from Endo/Lavamoat/Agoric/SES to harden the global primordial objects. The mechanism utilizes JavaScript proxies and the with statement to restrict access and uses eval in this restricted context to run code sandboxed.("Trustworthy Proxies: Virtualizing Objects with Invariants" 2013) Additionally, the milestone integrates commonjs module import with in-process sandboxing. However, this version will not have DOM access. These improvements aim to enhance the security of the system by preventing unauthorized access to sensitive resources.("Proxies: Design Principles for Robust Object-Oriented Intercession APIs" 2010)

v0.5 - Advanced Sandboxing introduces an advanced sandboxing feature. This includes implementing a fast and lightweight virtual DOM, as well as a communication protocol between the sandbox and renderer threads. This allows for the transfer of DOM updates from the sandbox to the renderer, as well as events from the renderer to the sandbox. Additionally, a basic signaling library and protocols are added to improve usability.

v0.6 - Intermediate Vault focuses on refining the permission system of the vault. Several basic demo apps are implemented to help with this. The team is also working on defining different data types and structures that will be supported in apps, as well as creating meaningful user permission prompts based on the data. A test library is being implemented to semi-automate testing and simulate user interaction scenarios. The access control interaction is being designed, and the permission system is being tested.

Alpha release

v0.7 - Basic Deamon involves the creation of a basic cross-platform installer for a "localhost p2p data daemon", which includes the addition of hyper-dht-relay. Code to detect and start communication with the data daemon is added to the kernel.

v0.8 - Advanced dpm focuses on advancing the data package manager (dpm) by adding new features. These features include the ability to download and publish data packages to the peer-to-peer (p2p) network, exporting data packages as p2p archives for the web server cache, and downloading data packages from the web server cache using dpm. These enhancements are aimed at improving the efficiency and functionality of the dpm, allowing for easier sharing and management of data packages across the network.

v0.9 - Advanced vault introduces the advancing of the vault. The following features are added:

– User self-authentication and self-authenticating data structures

– Wrapping all vault data in hypercores or similar p2p data structures and syncing them from browser cache to device disk via localhost daemon

– Enabling sharable p2p dataset links for others to access vault data if permission is granted by the user

– Creating a p2p address book of trusted peers

– Enabling downloading datasets from peers given access was granted by those peers

– Enabling subscription to dataset updates.

Early Beta release

Future work

While we have developed a conceptual architecture for all parts of the project and have a clear roadmap, there are still some key challenges that need to be addressed. These challenges include:

– Improving the developer experience to make it more intuitive and easy to understand how to build p2p web apps with a data vault.

– Adapting the SES object capability based model to our specific use case and needs in the context of p2p web apps based on a data vault standard.

– Enhancing the user experience for access control management of their vault data while also maintaining flexibility for developers.

– Integrating all existing p2p data structures used in the dat-ecosystem with the vault model and its permission system.

We are actively working on the challenge and are committed to create a user- and developer-friendly, reliable, and sustainable p2p web bridge.

Existing work

Dat ecosystem projects build infrastructure and applications on top of protocols, designed for syncing large and constantly changing data. These protocols use a cryptographically secure register of changes to prove that the requested data version is distributed. By using dat protocols (i.e. hypercore protocol) in their applications, developers can choose to fully or partially replicate the contents of a remote data repository. To ensure writer and reader privacy, protocols use public key cryptography to encrypt network traffic. A group of clients can connect to each other to form a public or private decentralized network to exchange data between each other.(Ogden et al. 2018)

P2P Prototyping

In the Dat ecosystem several attempts to enable experimentation with the protocols have been made. Initial tool, called try-dat, relied on servers and was eventually closed due to growing hosting costs that were diverted from funding protocol development. Later attempts included browsers like Agregore and Beaker, which allowed developers to prototype apps. However, only Beaker Browser focused on a convenient developer experience with the centralized hosting service hashbase.org, but it was limited to hyperdrives, a p2p filesystem abstraction. Although this setup had some success, it was a bit centralizing and it also did not properly address the issue of giving users control of

their data and separating it from the apps, resulting in p2p apps feeling like traditional apps. Since not many users were willing to download a whole new browser for a relatively intangible p2p experience, the project founder eventually moved on and sunsetted the project.

Data Vault

While there are existing solutions like unhosted.org, remotestorage.io, and more recently solidproject.org that encourage app developers to separate app logic from data and encourage users to self-host or find hosting providers for their data backend, none of these solutions focus on p2p networks and data structures or address the need to avoid the need for specific hosting providers in the first place. While the TrustVault Data Wallet white paper from 2022 comes close to our vision, it does not particularly focus on the web and is to some degree coupled with blockchain infrastructure, whereas our focus is on leveraging the existing and thriving p2p dat ecosystem with active apps and users and building on the HTML5 stack for app development.

In-browser Sandboxing

Working with self-authenticating data structures in a p2p network requires security mechanisms beyond traditional domains, such as CORS and CSPs. The SES and Endo project use javascript proxies and lockdown of the environment to create membranes around parts of the code, enabling any kind of policy for any part of the code. Meanwhile, the Lavamoat project focuses on sandboxing individual modules within a single app and allows developers to specify such policies that restrict what each module is allowed to do in the first place. This is crucial for loading and executing code from different p2p origins in the same process environment and allowing users to control data access via their vault without compromising performance or security. Both Metamask and Agoric use these techniques already to build secure crypto wallets and smart contract environments for DeFi on a proof of stake blockchain.

Conclusion

In the Dat-ecosystem there are many organizations building solutions very compatible with the DataShell and in need of a standardized data vault solution and we are already in disccussions with many of them to explore the detailed needs and opportunities to maximize user benefits while fine-tuning the the data vault standard, which is going to enhance all dat-ecosystem projects.

With one standard across the dat-ecosystem we could finally enable the first steps towards real data portability and data sovereignity and because of the nature of the data-sharing protocol, users would also get self-authentication out of the box when sharing and collaborating with each other.

Additionally, neighbouring ecosystems will also benefit from this technology, specially blockchains, where we see the lack of good answer on how to empower their users with full data sovereignity and on how to build decentralized apps that don't just allow financial crypto transactions, which fit on a blockchain, but actually include larger real world datasets, such as the endless streams of audio, videos, photos and other kind of datasets that are too large and expensive to be stored on a blockchain. A standard for p2p web apps based on a data vault could get adopted by many of these ecosystems.

References

Ogden, Maxwell, Karissa McKelvey, Mathias Buus Madsen, and Code for Science. 2018. "Dat - Distributed Dataset Synchronization and Versioning."

"Proxies: Design Principles for Robust Object-Oriented Intercession APIs." 2010.

"The webAppOS Architecture." 2019.

"TrustVault: A Privacy-First Data Wallet for the European Blockchain Services Infrastructure." 2022.

"Trustworthy Proxies: Virtualizing Objects with Invariants." 2013.