Immutable Views

Access control (to your information) for masses

Yan Shvartzshnaider
Princeton University
yansh@princeton.edu

ABSTRACT

There are a lot of on going efforts in the research community as well as industry around providing privacy-preserving and secure storage for personal data. Although, over time it has adopted many tag lines such as Personal Information Hub [12], personal container [5], DataBox [4], Personal Data Store (PDS) [3], and many others, these are essentially reincarnations of a simple idea: provide a secure way and place for users to store their information and allow them to provision who has access to that information.

In this paper, we would like to discuss a way to facilitate access control mechanism (AC) in the various “personal cloud” proposals.

1. INTRODUCTION

Personal information storage is the something we seem to take for granted. Today, we choose to store our data in varied “clouds” because we love the simplicity and convenience these services provide. However, this flourishing relationship relies on the implicit trust between users and service providers that users’ privacy will remain intact. While often we choose to believe that (major) service providers are the good guys4, recent cases, where this trust was broken, are forcing us to realize that things are not as rosy as they might seem.5 We gradually become aware of the fact that our information is being exploited by the big cooperations for a relatively low price—free service.

How to ensure the our information is safe and how to (re)gain control? These questions are currently preoccupy many researchers. The answers are not trivial and will probably require a clean-slate approach in how we build and manage our information systems. In addition to exploring new business models6 and incentives to change the general behavior of everyone involved.

Recent efforts [12, 8, 4, 3] look at facilitating “personal clouds” to store users’ data. While many seem to agree that such approach can offer a viable alternative, questions on how such systems should be designed and implemented remain open.

On one extreme, works like [2] offer a completely decentralised cloud using P2P-based approach; storage is crowd-sourced from a pool of participating users. This approach requires strong incentive schemes to reach critical mass. Alternatively, we can also use various cryptographic approaches to help us secure our information. But one can never be sure, so other approaches [1] employ secret sharing techniques to split and spread content’s chunks across different third party clouds. Although, it prevents any single third party from obtaining the whole content, this approach limits the access control to share all or nothing options.

However, what if the goal is to share parts of the information? Users might want to share some of their information with a third party to benefit from its services.

In this paper we propose an approach for provisioning third party access to the bits released by the user. Our approach is based on snapshots, we call them “immutable views”. The views are created by running a query against a personal information space. The result is an immutable view containing only the information that the requester (and only him) can “see”. The requester only deals with views, raw information is never retrieved. This means that immutable views can be shared and stored by a third party without compromising user’s privacy—since only the authorized requester can query it.

We believe, the ideas described in this paper can be leveraged by many of the ongoing efforts, however, our implementation aligns well with the platform offered by the European User-Centric Networking project [12]. In particular, our system uses Irmin [11], a git-like distributed store and Mirage OS [7]. Nonetheless, our current implementation [13] is programmed to support other backends.

2. INFORMATION SHARING

Our AC model is based on Moana [11] service abstraction. Moana provides a graph-based service abstraction and information model. The Moana service model exposes two functions ADD and MAP. The ADD function annotates the graph, while the MAP function is a standing graph query on the global information space.

AC model. As depicted by Figure 1 the AC model comprises three conceptual views. The first view from the bottom—global information space (GIS)—contains all the facts and made assertions. Note that assertions also include the policies used to determine what gets through to the second view. The second view, is a result of a query, referred by us as policy MAP, onto the first view for all the information the requester can access. Finally, the last view is specific to the requester’s query, we refer to it as an interest MAP.

In other words, any request triggers a chain of MAP operations that narrows down the information scope accessible to the requester.

Protocol. In layman’s terms, we propose git inspired

---

4This despite explicit statements by some of the services that their business models are mainly depend on going through our private information.
protocol for sharing information which enforces the above AC model. To store information, the user will *init* a master information repository (IR), which will store all of its information. This essentially creates the first view, which the owner can populate with relevant AC policies. A third party service that would want to access it for their purposes will need to *clone* it into a local view. “Clone” operations rely on the policy MAPs in the AC model. This ensures that only permitted information is cloned. The third party service is then left with a “copy” they can use for querying (using interest MAPs) and running their algorithms. It is worth noting again that the query doesn’t return any raw information, rather an immutable view wrapped in a sandbox container that can only be queried according to the specified AC policies. This is similar to SafeAnswers proposal in [3], which only returns *answers* to the requester, not raw metadata. For view updates, the *ADD* function is used, which creates a new immutable instance of a view. An updated view from the master repository will be *pulled* and new view instances created by third parties will be *pushed* to the master repository, following respective merge of the views.

**Social network example.** To illustrate intuitively how we can use this protocol, let’s consider how a simple social network can be implicitly built on top of it. Our social network follows a simple model: users have followers, who are permitted to get updates e.g., on new photos, status updates, events, etc. They can also comment on users’ content. Users publish new information such as events, photos and status updates to their personal IR. The repository, as previously mentioned, contains and enforces the AC policies describing who can see what. Users’ followers, those who wish to obtain his content will clone the repository. Consequently, similar to git, the followers will be able to pull updates from the master IR and also push (if permitted) their comments, as a new view instance, to be merged with the master view.

**Implementation.** For our system design we are looking at leveraging some recent work into unikernels [6] based on Mirage OS [7]. In particular, we envision IRs served as unikernels. Unikernels are virtual machines that run specialised Operating System (OS) to support individual software components. Mirage OS provides with the needed flexibility in deployment and management. In particular, thanks to Mirage OS, unikernels can be compiled and run against any environment such as a personal computer, custom open-source hardware and cloud. Most importantly, unikernels can be programmed with internal logic to ensure that users’ policies will be respected. Once a requester authenticates against the IR service, hosted on a unikernel, another unikernel is spun off containing information subset (in an immutable view) to which the requester has access. The requester will essentially interact with IR service specially “designed” for him.

3. REFERENCES

[1] Bessani, A., Correia, M., Quaresma, B., André, F., and Sousa, P. Depsky: dependable and secure storage in a cloud-of-clouds. ACM Transactions on Storage (TOS) 9, 4 (2013), 12.
[2] Boshevski, T., Brandoff, J., and Buterin, V. Shawn Wilkinson (shawn@storj.io).
[3] de Montjoye, Y.-A., Shmueli, E., Wang, S. S., and Pentland, A. S. opensids: Protecting the privacy of metadata through safeanswrs. PloS one 9, 7 (2014), e98790.
[4] Haddadi, H., Howard, H., Chaudhry, A., Crowcroft, J., Madhavapeddy, A., and Mortier, R. Personal data: Thinking inside the box. arXiv preprint arXiv:1501.04737 (2015).
[5] Ion, I., Sachdeva, N., Kumaranaguru, P., and Capkun, S. Home is safer than the cloud!: Privacy concerns for consumer cloud storage. In Proceedings of the Seventh Symposium on Usable Privacy and Security (2011), SOUPS ’11.
[6] Madhavapeddy, A., Mortier, R., Rotos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., and Crowcroft, J. Unikernels: Library operating systems for the cloud. In ACM SIGPLAN Notices (2013), vol. 48, ACM, pp. 461–472.
[7] Mirage OS website. [http://www.openmirage.org/](http://www.openmirage.org/) month viewed April 2015.
[8] Mortier, R., Greenhalgh, C., McAuley, D., Spence, A., Madhavapeddy, A., Crowcroft, J., and Hand, S. The personal container, or your life in bits. In Proc. Digital Futures (2010).
[9] Ng, I., Maull, R., Parry, G., Crowcroft, J., Scharf, K., Rodden, T., and Speed, C. Making Value Creating Context Visible for New Economic and Business Models: Home Hub-of-all-Things (HAT) as Platform for Multisided Market powered by Internet-of-Things. In Hawaii International Conference on Systems Science (HICSS), Hawaii, USA (2013).
[10] Shvartzshnaider, Y., and Ott, M. Moana: a case for redefining the internet service abstraction. In Proceedings of the 9th Workshop on Middleware for Next Generation Internet Computing (2014), ACM, p. 2.
[11] Thomas Gazagnaire. Introducing Irmin: Git-like distributed, branchable storage. [http://openmirage.org/blog/introducing-irmin](http://openmirage.org/blog/introducing-irmin) month viewed April 2015.
[12] website, U. The User-centric networkofficial. [http://usercentricnetworking.eu](http://usercentricnetworking.eu) month viewed August 2014.
[13] Yan Shvartzshnaider. Moana source repository. [https://github.com/yansh/MoanaML](https://github.com/yansh/MoanaML) month viewed April 2015.