Thesis Proposal: Helping Web Developers Give Users Control Over Their Data

by

Lillian Tsai

A.B., Harvard University (2017)
S.M., Harvard University (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

© Massachusetts Institute of Technology 2024. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 13, 2024

Certified by .....
M. Frans Kaashoek
Charles Piper Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Co-Certified by .....
Malte Schwarzkopf
Professor of Computer Science, Brown University
Thesis Co-Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Thesis Proposal: Helping Web Developers Give Users Control Over Their Data

by

Lillian Tsai

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2024, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Users today give up control of their data to web services in order to use these services. This puts user data at risk of data breaches and exposes it to third parties in ways unwelcome or unknown to the user. Incentivized by recent legal developments such as the GDPR and CCPA, as well as growing public demand, developers increasingly need to implement better user controls in their web applications.

However, doing so is challenging: the data that users want to protect is often important for the service's function. This thesis explores the gray space where users' need for data protection and application functionality requirements overlap.

Thesis Supervisor: M. Frans Kaashoek
Title: Charles Piper Professor of Electrical Engineering and Computer Science

Thesis Co-Supervisor: Malte Schwarzkopf
Title: Professor of Computer Science, Brown University

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Users today have little to no control over the data they give to web applications. They must either avoid using an application, or leave their data susceptible to leaks, e.g., via SQL injections or compromise of other users' accounts. Incentivized by growing public demand and laws such as the GDPR and CCPA [**? ?**], developers increasingly need to implement mechanisms to give users control over their data's exposure.

Central to this conversation are two needs: users want to protect their data, and applications require data to be available in order to function. Problems arise when application functionality requires the same data users want to protect: for example, deleting a post may orphan other users' comments on it, crashing the application when it expects comments to have an associated post. Handling this correctly is difficult for developers—naively removing user data can break the application or reduce its functionality. Removal is also usually permanent, hurting both users and applications: once a user deletes their data, there is no coming back.

Large companies have built systems to help developers get data deletion right [**?**], but the problem goes beyond data deletion. Users may want to take a break from using an application, but rest easy that their data is protected while they're inactive; or users may want more fine-grained control—i.e., have their data exist in a state where it is both visible and protected. For example, a user might want to disassociate content with a specific hashtag from a social media account. Because applications today lack support for this, users resort to ad-hoc alternatives like throwaway accounts and "finstas" [**? ?**] to post content that is visible but disconnected from the "real" account. This burdens users with managing multiple accounts and requires planning ahead, since moving posts into a throwaway account or reclaiming them is impossible.

1.2 Approach

This thesis investigates novel systems to let web application developers provide users with means to manage their data in the gray space where users' need for data protection and application functionality requirements overlap.

We start by building Edna, a library that integrates with web applications and provides abstractions that let developers specify not just wholesale removal of data, but also flexible redaction or decorrelation of data. Edna then changes the database contents in well-defined ways that avoid breaking the application. Developers and users both benefit from Edna: applications can expose and advertise new privacy-protection features and reduce their exposure in the event of a data breach, while users gain more control over their sealed data, including the convenience of returning to the application as if their data had been there all along.

Edna introduces *sealing*, which removes or redacts some or all of a user's data, and *revealing*, which restores the sealed data at a user's request. Sealed data remains on the server, but is encrypted and inaccessible to the web application. Sealing changes the database contents and replaces the data to seal with placeholder values where necessary (e.g., comments require an associated post).

We designed Edna's seal and reveal abstractions to flexibly meet applications' diverse needs while still protecting sensitive user data. Edna provides developers with three well-defined primitives for anonymizing data: remove, modify, and decorrelate. Developers compose these to create *seal specifications*, which describe the data to seal and which primitives to apply. When invoked with a seal specification, Edna changes the database contents as specified: "remove" drops rows, "modify" changes the contents of specific cells, and "decorrelate" introduces *pseudoprincipals*, anonymous placeholder users. These pseudoprincipals help maintain referential integrity, but also can act as built-in "throwaway accounts." Rather than users having to create and maintain throwaways themselves, Edna's pseudoprincipals enable the user to later reassociate with their throwaway's data via revealing, or create throwaways to disown posts after-the-fact.

Chapter 2

Related Work

2.0.1 Encrypted Data Storage

CryptDB, Mylar

2.0.2 Policy-Enforcing Systems

Qapla

Chapter 3

Research

3.1 Edna

Users today give web applications their data but have little control over how these applications store and protect this data. Giving users control is challenging for developers, as application functionality sometimes requires the very same data that users want to protect.

Edna is a library that lets web application developers give users flexible and correct control over their data's exposure without breaking the application. Edna minimizes application changes and developer effort, and enables applications to securely hide user data and users to later restore it. Edna introduces abstractions for *sealing*, which selectively renders user data inaccessible via encryption, and *revealing*, which enables the owning user to restore their data to the application.

We implemented a prototype of Edna, and evaluated it with three real-world web applications: Lobsters, a discussion forum; HotCRP, a conference review system; and WebSubmit, a homework submission application. Integrating Edna into these applications required minor code changes. For example, supporting three types of sealing in the 160k-LoC Lobsters application required adding 697 LoC. The modified applications retained their expected semantics and functionality. Edna's sealing and revealing performance depends on the amount of data affected, but they generally complete in seconds and have a small effect on the performance of concurrent operations.

In this thesis, I will discuss Edna's relationship to prior research, its limitations, and the following contributions:

1. A new paradigm for developers to provide flexible user control over data in web applications, with cryptographic protection and while maintaining application functionality and invariants.
2. Abstractions for sealing and revealing, and a small set of data-anonymizing primitives

(remove, modify, decorrelate) that cover a wide range of application needs and compose cleanly.

3. The design and implementation of Edna, our prototype library that implements user data control via sealing and revealing, as well as Ednacrypt, which additionally encrypts unsealed data.
4. Case studies of integrating Edna with three real-world web applications, an evaluation of Edna's effectiveness, performance, and security, and a comparison to Qapla.

3.2 Hydra

Hydra narrows down on a specific aspect of Edna, namely how users control their identities and links between their identities and data in web applications. Users today will

Users often want to Hydra specifically on

In Hydra, users authenticate with an application like normal,

Finstas (fake instagrams), Reddit throwaways. Multi-account containers in Firefox let users act as distinct identities as they browse the web.

3.3 Funhouse

Chapter 4

Execution Plan

4.1 Timeline

- **Spring 2023**:
  - Focused work on Funhouse to bring it to paper-ready status.
  - Prepare a conference-ready paper and talk on Edna.
  - Complete the RQE using an Edna talk.
  - Complete the thesis proposal (this document).
- **Summer 2023**:
  - Focused work on Funhouse, or potential internship.
  - Potential conference presentation of Edna.
  - Potential workshop presentation of Funhouse HotOS.
- **Fall 2023**:
  - Funhouse conference submission if ready.
  - Prepare job talk materials (Funhouse or Edna).
- **Spring 2024**:
  - Finalize last Edna-related thesis research (Hydra).
  - Finalize Funhouse contributions for thesis, if any.
  - Write and defend thesis.

Bibliography