Project Conference I
Eleanor Eng, Kate Nelson, Koyena Pal, Aalia Habib
Problem addressed

Bugs in deletion code undermine the integrity of the service and can lead to exploitable vulnerabilities that allow users to delete arbitrary data. Additionally, failing to effectively delete user data can trigger substantial regulatory fines.

Why is it important?

The ability to delete data is a core privacy expectation for users who entrust applications with their personal information.

Solution proposed

Extending the DELF framework to websubmit-rs (MySQL) and SignMeUp (MongoDB) web applications.
Understanding delf-rs
DELF inspired deletion framework (delf-rs) available in Rust written by CS2390 alum!

Initial Learning
First we must familiarize ourselves with Rust, Rocket, mysql, and Diesel. Then, to apply the DELF framework, we must learn about the workings of websubmit-rs and SignMeUp.

Adding MongoDB Compatibility (Stretch Goal)
delf-rs is only compatible with MySQL. We would like to modify it to be compatible with MongoDB as well.
What's Happening Right Now?

We are creating a simpler version of websubmit-rs built on a mysql backend so it will be compatible with delf-rs.

Current Hurdles

- Learning to code in Rust
- creating websubmit-rs with a mysql backend instead of Noria
What's Next?

- Finish implementing simpler version of websubmit-rs and ensure delf-rs is compatible with it
- Create a design plan for how we will extend delf-rs so that it supports applications that use MongoDB
- Add functionality based on our design so that delf-rs is compatible with SignMeUp
3. Implementing O-DORY: Reproducing Keyword Search and Document Retrieval in Encrypted File-storing Systems

Emmie He
4. Solid Pod Storage for Mail Delivery Agents

Chris Sarli
s**mail**—Solid mail storage

1. It’s hard to evaluate the functionality of Solid’s underlying technologies because of its strong opinions of what types of apps and systems should exist.

2. Email is decentralized by design (SMTP), but semi-centralized in practice (Gmail and other providers do not provide direct access to email storage).

**s**mail: Use Solid Pods as the mechanism for mailbox storage of email.

![Diagram of s**mail**—Solid mail storage](image)
Making SignMeUp GDPR Compliant
Problem Statement

● SignMeUp is not GDPR compliant

● Due to the pandemic, students studied remotely from different areas
  ○ E.g. from the EU!

● SignMeUp must respect the GDPR to remain compliant for its users in the EU.
Solution

- Make SignMeUp GDPR compliant!
  - In particular, address Article 15 and Article 17:
    - Article 15: Right of access by the data subject
    - Article 17: Right to erasure (‘right to be forgotten’)
- Allow users to request and view the data that SignMeUp has collected on them
- Allow users to delete their data from SignMeUp
Current Progress and Findings

- Explored the structure of the SignMeUp code
- Started up a test instance locally, ran a normal use case, and inspected the database
  - (thanks Kinan for helping me get set up!)
- It appears the database keeps every ticket a user has submitted for a queue, even if the queue has ended!
Purpose Limitation for Pelton

Amrit Singh Rana
The problem and why it is important

- Article 5(1b) of the GDPR states that Personal data shall be collected for specified, explicit and legitimate purposes and not further processed in a manner that is incompatible with these purposes.
- Essentially means checking data every time it leaves the DB to ensure it is being used for a purpose that is GDPR compliant.
- Implementing purpose limitation manually is labor intensive and error prone.
- While large companies have the resources to modify their systems to guarantee compliance, it is difficult and expensive for small companies to do that.
- Non compliance can lead to large fines, even for small and medium companies.
Solution

- A purpose limitation system for databases that
  - Requires minimal effort to understand and use
  - Minimizes scope of error
  - Is useful to prove compliance

- Key idea:
  - Store purpose of use for each record when it is created
  - Do not let it leave DB for any purpose other than that

- Assumption: Developer is honest

- An extension to Pelton and its compliance-by-construction philosophy
Preliminary Design

Web Application -> SQL Queries -> Purpose Limiter

Pelton (DB)
Proposed changes to queries

Queries intercepted/modified:

- CREATE TABLE: add columns to store purpose metadata
- INSERT INTO TABLE: expects purposes for each row
- SELECT:
  - Expects purpose of query - One query can have only one purpose
  - Filters out records which do not match to the query purpose

Queries proposed:

- UPDATE PURPOSE: to update purpose for records
Q&A/Feedback
7. Mimi: A Client-Side Password Manager

Mason Zhang, Amanda Lee
8. Multi-Party Secure Document Search

Wyatt Howe, Junewoo Park
### Background: Private Search

We want the user’s query, search result, and retrieved document to be private.

<table>
<thead>
<tr>
<th>doc\key</th>
<th>Law</th>
<th>EU</th>
<th>School</th>
<th>Apple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brown U</td>
<td>10</td>
<td>0</td>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>Fruits</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>GDPR</td>
<td>100</td>
<td>90</td>
<td>30</td>
<td>25</td>
</tr>
</tbody>
</table>

We want the user’s query, search result, and retrieved document to be private.
Background: **HE vs MPC**

### Homomorphic Encryption

- **F(x)** encrypted
- **F( )** decrypted

### Multi-Party Computation

- **F₂( ◮ )**
- **F₃( ◯ )**
- **F₄( ◬ )**

---

**Multi-Party Secure Document Search**

---

Left:

- User
- **F( )**

---

Right:

- User
- **F( )**
The Protocol

Keyword encoding
- Search vector is big and sparse -> Can we encode better?

Matrix multiplication
- FHE is expensive $O(n^2)$
- Matrix is public -> public multiplications are free under MPC

Most relevant document lists
- New bottleneck!
- Can find $k$-max in $O(n \cdot k^2)$, but $n$ is very large -> approximate it?
Variable-Owner Shared-Data in a Privacy-By-Design Datastore

CSCI 2390 Project by Justus Adam
Traditional Datastores

All the data is mixed in a single place
Malte and Kinan Have a Project called Pelton

Traditional Datastores

All the data is mixed in a single place
Malte and Kinan Have a Project called Pelton

Traditional Datastores

All the data is mixed in a single place

Pelton

Every user has their own database
Malte and Kinan Have a Project called Pelton

Traditional Datastores

All the data is mixed in a single place

Pelton

Every user has their own database

Queries are made efficient by computing views
How it works:
Putting data in the right database entails knowing which user it belongs to.
How it works:

Putting data in the right database entails knowing which user it belongs to.

Plates

<table>
<thead>
<tr>
<th>license_plate</th>
<th>user_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>A23 7U7</td>
<td>1</td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
</tr>
</tbody>
</table>
### How it works:
Putting data in the right database entails knowing which user it belongs to.

### Plates
<table>
<thead>
<tr>
<th>license_plate</th>
<th>user_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>A23 7U7</td>
<td>1</td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
</tr>
</tbody>
</table>

### Users
<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Malte</td>
</tr>
<tr>
<td>2</td>
<td>Kinan</td>
</tr>
</tbody>
</table>

Foreign keys usually help discover ownership.
How it works:
Putting data in the right database entails knowing which user it belongs to.

**Plates**

<table>
<thead>
<tr>
<th>license_plate</th>
<th>user_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>A23 7U7</td>
<td>1</td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
</tr>
</tbody>
</table>

**Users**

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Malte</td>
</tr>
<tr>
<td>2</td>
<td>Kinan</td>
</tr>
</tbody>
</table>

Rule:
If a row links to a data subject, the row belongs to that subject.

Foreign keys usually help discover ownership.
The Trouble With Variable Owners

How it works:
Putting data in the right database entails knowing which user it belongs to.

<table>
<thead>
<tr>
<th>Plates</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>license_plate</td>
<td>user_id</td>
<td></td>
</tr>
<tr>
<td>A23 7U7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Users</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>Malte</td>
</tr>
<tr>
<td>2</td>
<td>Kinan</td>
</tr>
</tbody>
</table>

Foreign keys usually help discover ownership

Rule:
If a row links to a data subject, the row belongs to that subject

What about variable-owner data?
The Trouble With Variable Owners

How it works:
Putting data in the right database entails knowing which user it belongs to.

<table>
<thead>
<tr>
<th>Plates</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>license_plate</td>
<td>user_id</td>
<td></td>
</tr>
<tr>
<td>A23 7U7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Users

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Malte</td>
</tr>
<tr>
<td>2</td>
<td>Kinan</td>
</tr>
</tbody>
</table>

Foreign keys usually help discover ownership

Rule:
If a row links to a data subject, the row belongs to that subject

What about variable-owner data?

Documents

<table>
<thead>
<tr>
<th>id</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2390 student comments</td>
</tr>
</tbody>
</table>

Document_belongs_to

<table>
<thead>
<tr>
<th>user_id</th>
<th>document_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
The Trouble With Variable Owners

How it works:
Putting data in the right database entails knowing which user it belongs to.

<table>
<thead>
<tr>
<th>license_plate</th>
<th>user_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>A23 7U7</td>
<td>1</td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Malte</td>
</tr>
<tr>
<td>2</td>
<td>Kinan</td>
</tr>
</tbody>
</table>

Documents

<table>
<thead>
<tr>
<th>id</th>
<th>document</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2390 student comments</td>
</tr>
</tbody>
</table>

Document_belongs_to

<table>
<thead>
<tr>
<th>user_id</th>
<th>document_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Rule:
If a row links to a data subject, the row belongs to that subject.
The Trouble With Variable Owners

How it works:
Putting data in the right database entails knowing which user it belongs to.

<table>
<thead>
<tr>
<th>license_plate</th>
<th>user_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>A23 7U7</td>
<td>1</td>
</tr>
<tr>
<td>BG6 30L</td>
<td>1</td>
</tr>
<tr>
<td>45L 0PO</td>
<td>2</td>
</tr>
</tbody>
</table>

Users
- id
- name
1 Malte
2 Kinan

Documents
- id
- document
1 2390 student comments

Document_belongs_to
- user_id
- document_id
1 1
2 1

Rule:
If a row links to a data subject, the row belongs to that subject.

Indirection table obscures ownership

The arrow is reversed

Foreign keys usually help discover ownership.
What Am I Going To Do?

- Storage capabilities for variable-owner data
What Am I Going To Do?

- Storage capabilities for variable-owner data
- Annotations/inference

How much can we infer, when does the programmer have to be explicit?
What Am I Going To Do?

- Storage capabilities for variable-owner data
- Annotations/inference
- ORM Integration

Can we infer more, if an ORM is involved?

How much can we infer, when does the programmer have to be explicit?
What Am I Going To Do?

- Storage capabilities for variable-owner data
- Annotations/inference
- ORM Integration

How much can we infer, when does the programmer have to be explicit?

Can we infer more, if an ORM is involved?

Explicit many-to-many relationships could guide inference

```python
class User(model.Model):
    documents = fields.ManyToMany()
```
Project Conference: Language for Machine Readable Policy

by Zhuo Wang, Wei Li, Jikai Zhang
Project’s Pitch

● Problem
  - Different users have different policies
  - How to execute SQL queries based on their policies?
● Why is it important
  - DB needs to follow user’s policies
  - Doesn’t make sense to let users alter their SQL queries
● Solution
  - Rewrite SQL queries
Simple approach

```
<table>
<thead>
<tr>
<th>Sorted ID</th>
<th>Users</th>
<th>Policy</th>
<th>Grades</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
<td>3</td>
<td>85</td>
</tr>
<tr>
<td>2</td>
<td>Wei</td>
<td>3</td>
<td>90</td>
</tr>
<tr>
<td>3</td>
<td>Sam</td>
<td>3</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>Jikai</td>
<td>4</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>Wilson</td>
<td>9</td>
<td>90</td>
</tr>
</tbody>
</table>
```

“policy = n” means can execute user’s data if there are at least n user’s data executed
Challenges

- We need to use the same database, hard to synchronize everyone’s DB
  - Plan to use server MySQL
- Come up with other policies
- Confirm the dataset to use in later steps
- Rewrite SQL queries more efficiently if dataset is super large
11. User-defined Privacy Policies for a GDPR-Compliant Database

Paul Biberstein, Max Heller, Julia McClellan
SELECT * FROM ORDERS
WHERE Date=10/26/21
SELECT * FROM ORDERS
WHERE Date=10/26/21

Data flow

Materialized View

σ

Paul's Orders
Paul's microDB

Max's Orders
Max's microDB

Julia's Orders
Julia's microDB
SELECT * FROM ORDERS
WHERE Date=10/26/21
SELECT * FROM ORDERS
WHERE Date=10/26/21
Step 1

How to limit data flow by query purpose?

Step 2

How can we provide a proof to end-user that policy rules are enforced?

What would a more extensive policy set look like? (eg: differential privacy requirements)

How can we efficiently and correctly update policies in production?
Case study in Data Transfer between Platforms Spotify and Youtube Music

Zlatko-Salko Lagumdzija
Project Pitch

- Article 20: The Right to Data Portability
  - “...the data subject shall have the right to have the personal data transmitted directly from one controller to another, where technically feasible.”
- How is *technically feasibility* defined?
- Who is responsible for it?
- Case study
Solution Outline

1. Search for solutions that do parts of this and see what they are able to do
2. Download YouTube Music and Spotify Data
3. Manually see how many items are technically convertible
4. Use Spotify’s and Youtube Music’s API to do some parts of the transfer automatically
   a. This could be limited to downloading the data
5. For the rest, make some script that does
GDPR-compliant data Access and Erasure tool for Java (GAEL4J)

Hugo Huang, Barry Zhang
GAEL is made for you if ...

You are developing a Web App and you want to implement an easy-use interface for users to erase and access their data.

You already have a Web App up and running, you want to introduce privacy conscious features without messing up the existing code.
How are we going to do that

Importability

Non-invasiveness
Upgrading the Cookie Glasses Web Browser Extension to TCFv2

Katie Ta, Charles Tan
What is Cookie Glasses?

- Browser extension that informs users if their cookie preferences are respected
- Utilizes IAB Europe's Transparency and Consent Framework (TCF)
Problem: TCFv1 is now TCFv2!

“The vision for TCF v2.0 is to provide enhanced transparency and choice to consumers and greater control to publishers.”

Non-technical differences:

- More purposes and more “user friendly” explanations of these purposes
- Users can now reject legitimate interests, not just deny consent

Technical differences:

- Updated API that is more flexible & provides more functionality
- More information included in the consent string: legitimate interests, new vendor purposes for collection, if GDPR applies to the user, etc.
μGDPR: Privacy Conscious Microservices

Yongjeong, Ayush
Hmmm… Microservices?

Microservices are an architectural and organizational approach to software development where software is composed of small independent services that communicate over well-defined APIs.

Sample Social Network Service Architecture (Gan et al., ASPLOS’19)
It is hard to keep track of processing of request for GDPR purposes ….
What we want to do?

- Benchmark the distributed log tracing (similar to audit-trail in GDPRBench) overheads for request-level logging per-user for different workloads.

- Use these logs for advanced use cases e.g., breach analysis i.e., if a particular node in the cloud was compromised which users were affected during the breach.
Reference

[1] An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems Gan et al., ASPLOS’19
16. A new user-focused architecture for internet apps

Chitradeep Dutta Roy
Objectives

- Data attributable to a user must stay under user’s control at all times.
- A service provider can hold information about how to locate and query user data, but can’t house it.
- Design should allow a user to limit access to data through some kind of permission system.
Use case: Twitter

Client (Follows Alice)

Login

Follow Charlie

Twitter

Userbox Addresses
Relationship graph

Alice's Data

Alice's Box

Charlie's Data

Charlie's Box

Bob's Data

Bob's Box

Fetch Alice's tweets (SUCCESS)

Fetch Bob's Tweets (FAIL)
BEAVER: Resin-Style IFC for Statically-Typed Languages

Livia Zhu, Sreshtaa Rajesh
What is our Motivation?

● Some drawbacks of RESIN which came up during our class discussion:
  ○ RESIN-style IFC framework was specifically designed for interpreted languages
  ○ RESIN does not have support for distributed applications

● Statically-typed languages make up the majority of computer systems and are the basis for a wide variety of distributed applications
Research Question: Can we implement Resin-style IFC in a statically-typed language?
What is BEAVER?

User Application

Policy
Object
field1

Filter
I/O

User code
Our library
Extern Library
What is BEAVER?

User Application

Policy
Object
field1

Filter
I/O

Policy propagation

User code
Our library
Extern Library
What is BEAVER?
Current status of BEAVER

Strategy 1: Have policies be traits wrapping objects

Strategy 2: Objects contain policies which implement certain methods
Private Document Search and Retrieval

Neophytos Christou, Richard Abou Chaaya
Private document retrieval

PIR : Requires the user to already know the document key in the database
Problem

1. Private keyword

2. Matrix multiplication

3. List of documents

Client
2. Matrix multiplication using homomorphic encryption (EXPENSIVE, ~ 90%)

1. Encrypted private keyword

3. Encrypted scores of all documents

4. Document key in the server DB
Our solution

1. Encrypted private keyword
2. R1

1. Encrypted private keyword
2. R2

3. \( f(R1, R2) = \text{Scores of documents} \)
Change from the initial proposal: 3 parties are involved (client and 2 servers).

TO DOs:

☑ Find a PIR library to use

☑ Find and implement a scheme assuming the two servers don’t collude

☐ Implement an end-to-end Proof of Concept

☐ Compare the performance of the different schemes

☐ (MAYBE) Come up with and implement a scheme that still works if the two servers collude (if possible)
Expanding the scope of GDPRBench

Singh Saluja
Project Pitch

GDPR-Bench is the standard benchmark for measuring overhead of GDPR-compliance in Database Systems.

Expand scope of GDPR-Bench by comparing it with more databases:

- MongoDB
- OracleDB (optional)
Approach

Divided into four sprints:

Sprint One - Understand current implementation of GDPRbench Workloads

Sprint Two - Working implementation of MongoDB with GDPRBench

Sprint Three - Implement and compare situations like MetaData Explosion, GDPR Anti-Patterns, encryption etc.

Sprint Four - Further optimizations kernel, fs-level etc.
Progress so far

GDPRBench Redis, controller-workload

25 / 10 / 2021 - start at 08:49:52.702 slope 0.04207
Progress so far

Bug - Port (6379) is open, database is running but YSCB is unable to write.

2021-10-26 09:30:10.079 10 sec: 0 operations; est completion in 106751991167300 days 15 hours
Questions
20. Establishing Strong Consistency in Pelton

Ishan Sharma
Futures and Promise

```cpp
std::promise<bool> p;
std::future<bool> f = p.get_future();
...
...
...
bool val = f.get();
p.set_value(true);
```

Paul Xu
The Data Request Process

Idea → Request → Data Provider → Model building

Months!!
Synthetic Data Generation

Idea → Request → Data Provider → Synthetic dataset → Model building

Request for original data
SynthBench

Real-world Dataset

Synthetic Data Generator

Synthetic Data Generator

Synthetic Data Generator

Synthetic Dataset

Model Testing

Metrics

Model Training