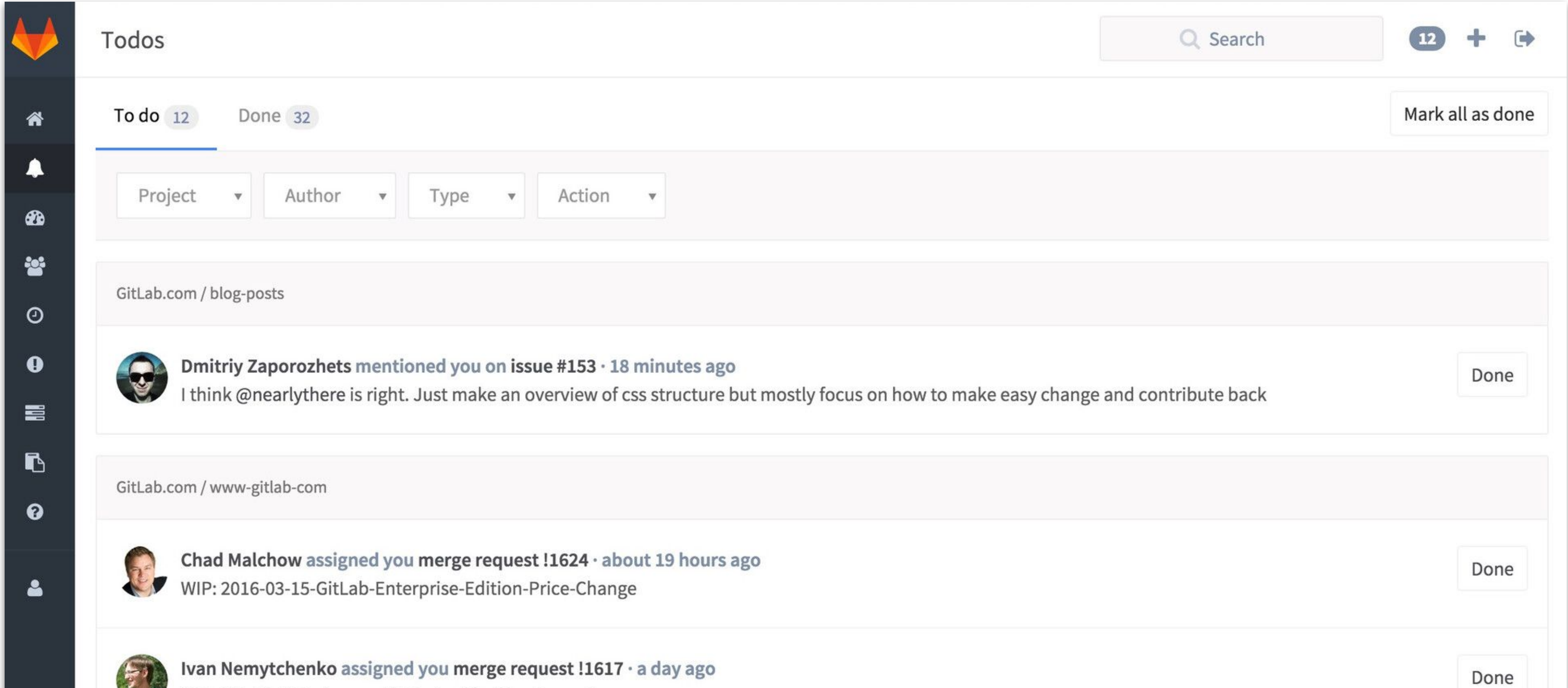
Safe Policy Migrations

John Renner, Alex Sanchez-Stern, Sorin Lerner, Deian Stefan

UC San Diego

GitLab

2016 - Introducing Todos!



GitLab.com / blog-posts

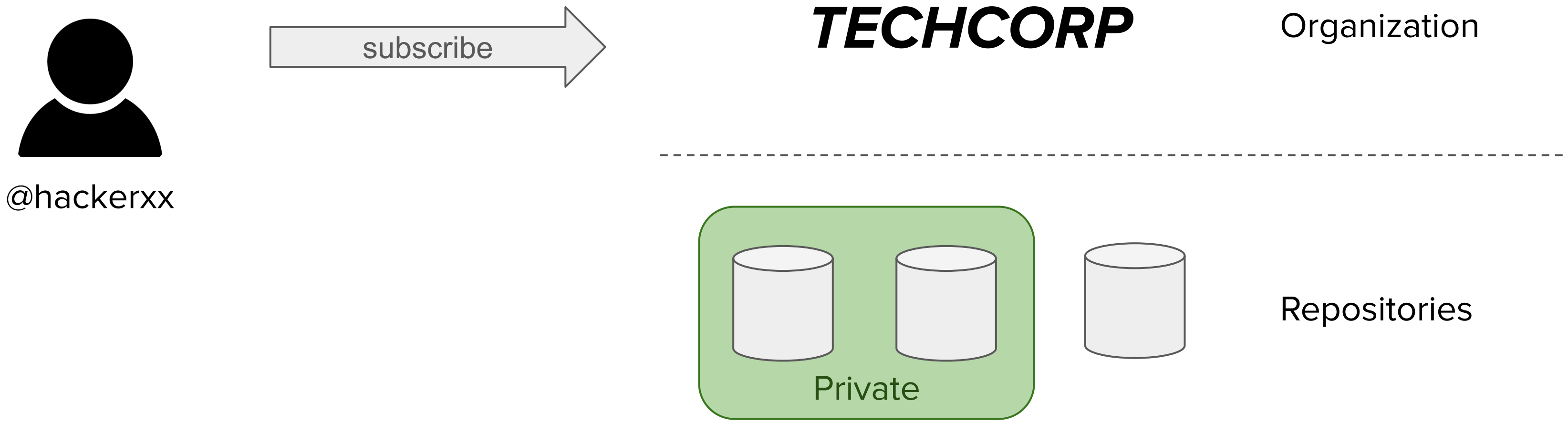
Dmitriy Zaporozhets mentioned you on issue #153 · 18 minutes ago
I think @nearlythere is right. Just make an overview of css structure but mostly focus on how to make easy change and contribute back

GitLab.com / www-gitlab-com

Chad Malchow assigned you merge request !1624 · about 19 hours ago
WIP: 2016-03-15-GitLab-Enterprise-Edition-Price-Change

Ivan Nemytchenko assigned you merge request !1617 · a day ago

2019 - Leak discovered in Todos



@hackerxx

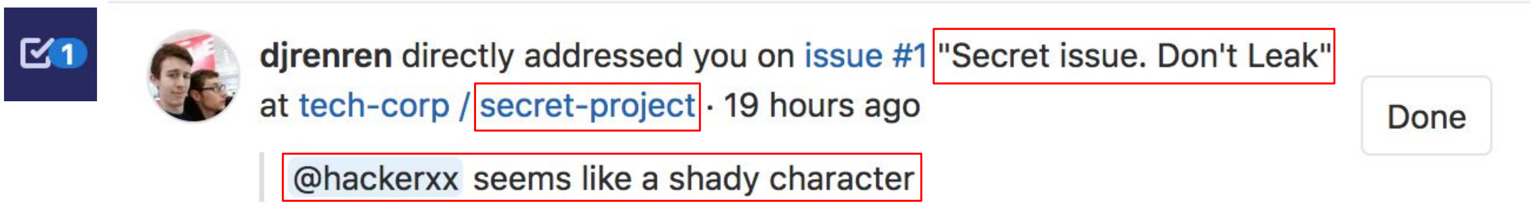
subscribe

TECHCORP

Organization

Private

Repositories

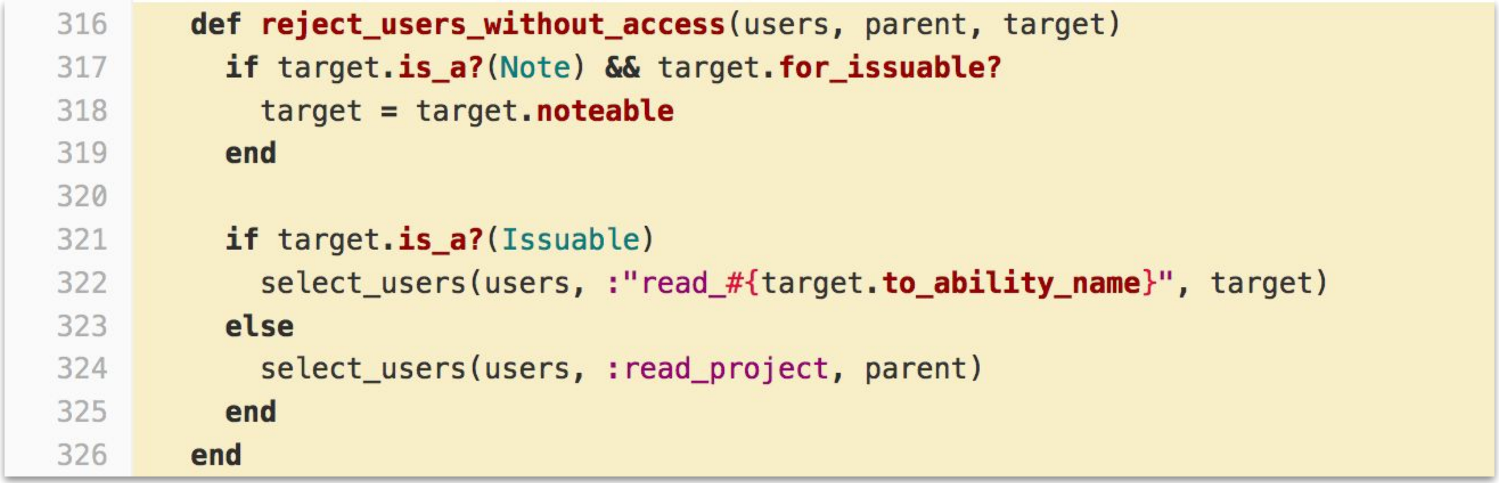


djrenren directly addressed you on issue #1 "Secret issue. Don't Leak" at tech-corp / secret-project · 19 hours ago

@hackerxx seems like a shady character

Buggy permissions check

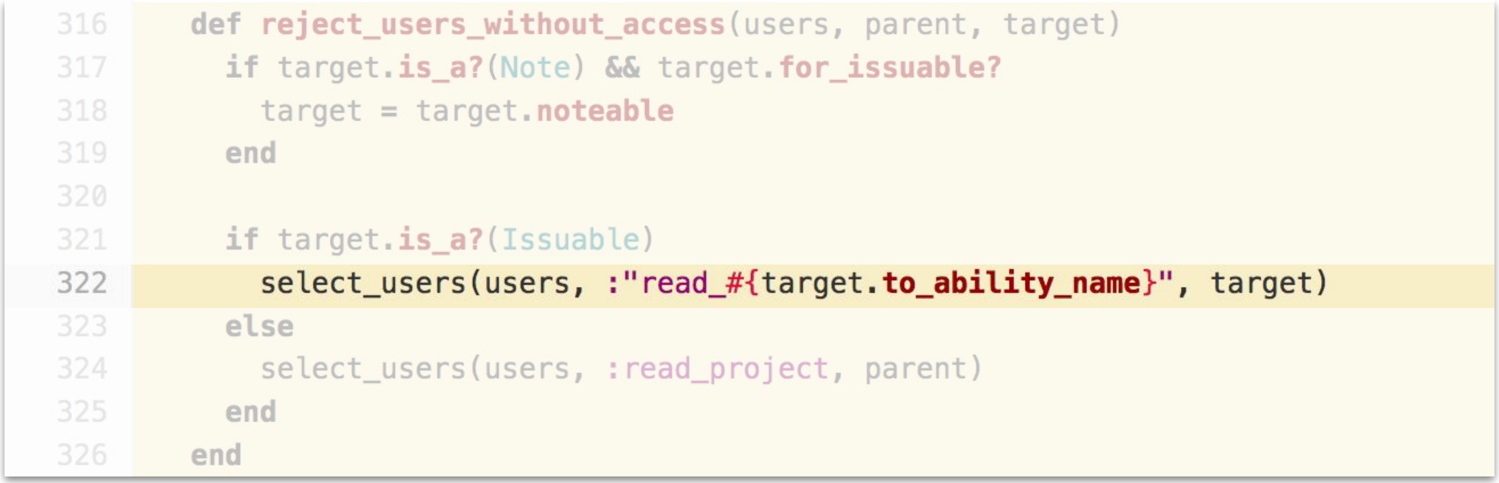
todo_service.rb



```ruby
  def reject_users_without_access(users, parent, target)
    if target.is_a?(Note) && target.for_issuable?
      target = target.noteable
    end

    if target.is_a?(Issuable)
      select_users(users, :"read_#{target.to_ability_name}", target)
    else
      select_users(users, :read_project, parent)
    end
  end
```

Buggy permissions check

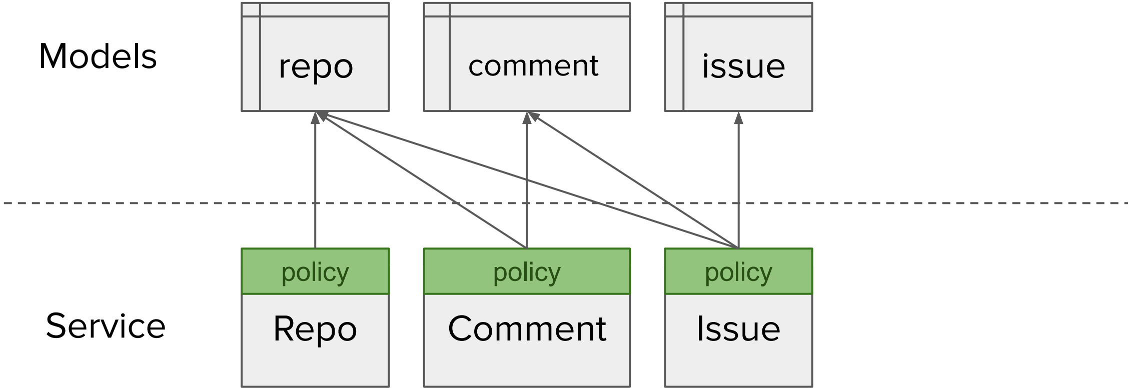
todo_service.rb



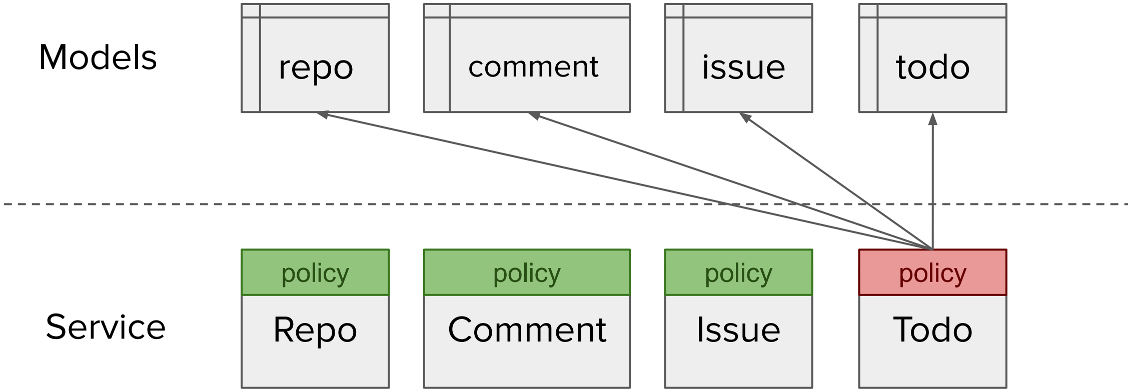
```ruby
def reject_users_without_access(users, parent, target)
  if target.is_a?(Note) && target.for_issuable?
    target = target.noteable
  end

  if target.is_a?(Issuable)
    select_users(users, :"read_#{target.to_ability_name}", target)
  else
    select_users(users, :read_project, parent)
  end
end
```

What went wrong?

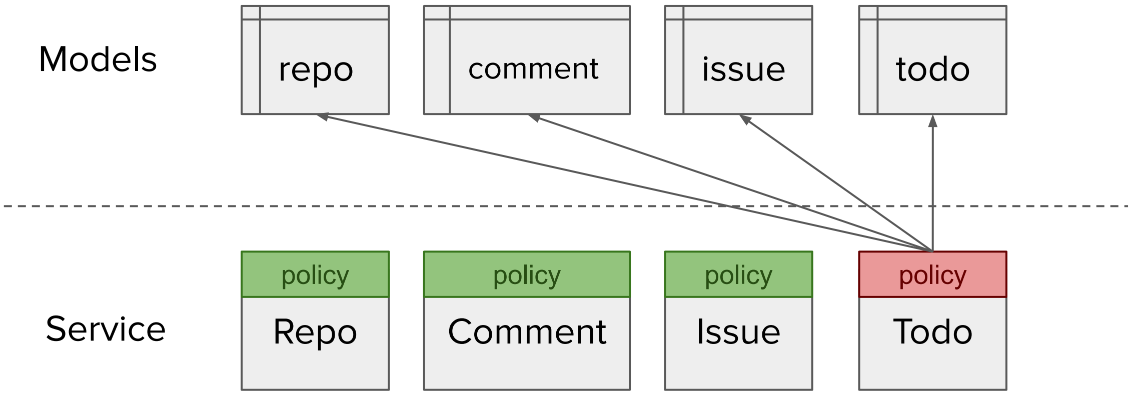
Update induced leakage



Update induced leakage



Update induced leakage

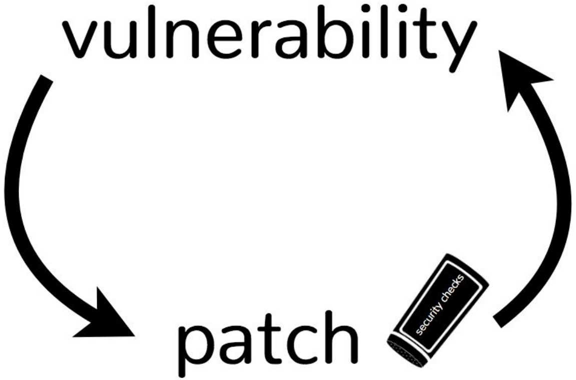


New feature required reimplementation of security checks

New checks were subtly incomplete

How can we prevent update-induced leakage?

Find & fix bugs

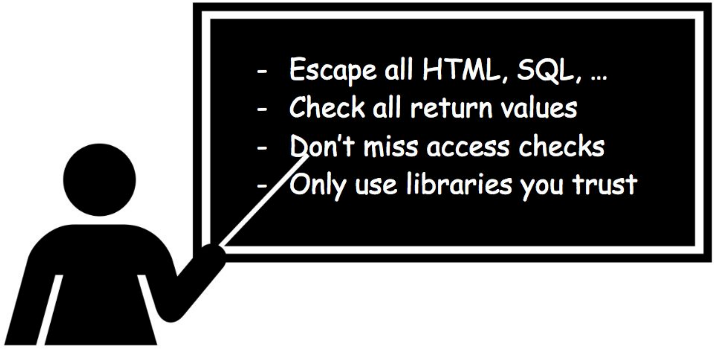


vulnerability

patch

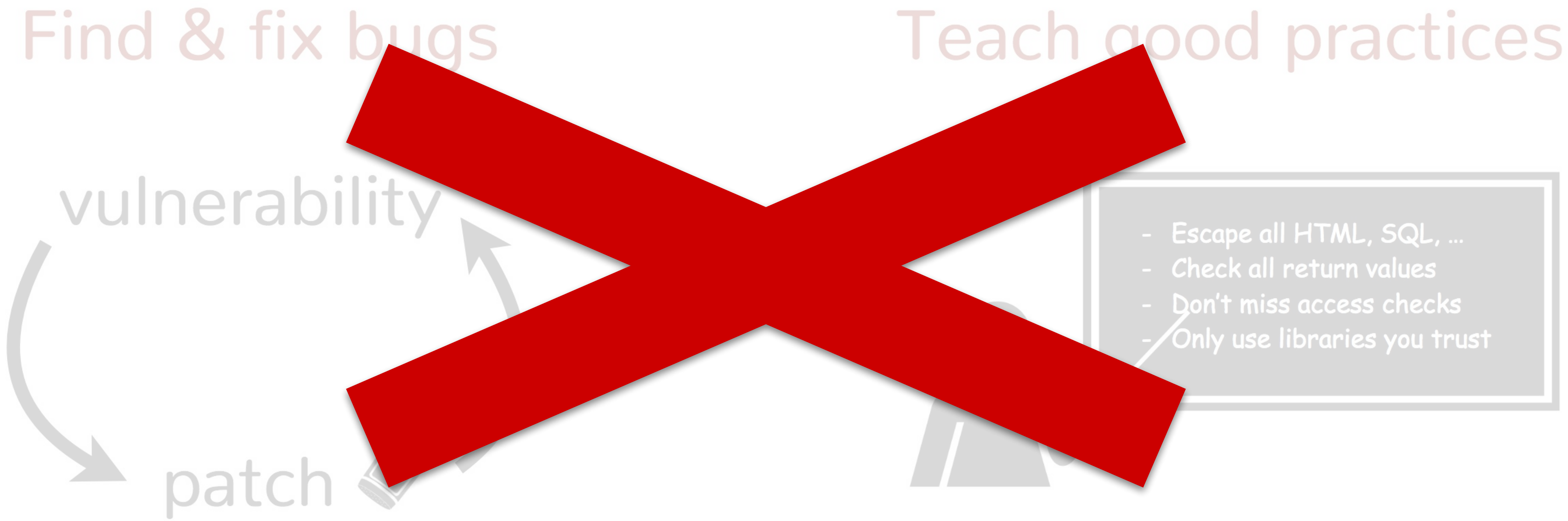
+

Teach good practices



- Escape all HTML, SQL, …
- Check all return values
- Don't miss access checks
- Only use libraries you trust

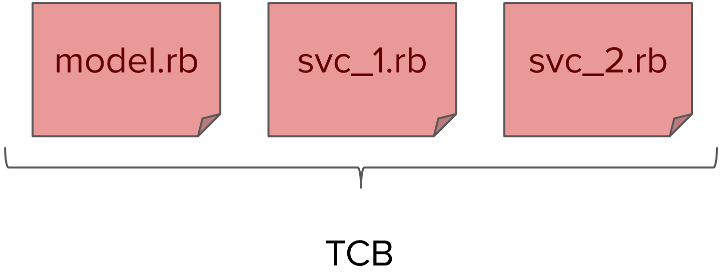
How can we prevent update-induced leakage?



- Escape all HTML, SQL, …
- Check all return values
- Don't miss access checks
- Only use libraries you trust

Doesn't work!

Decrease the amount of code we have to trust



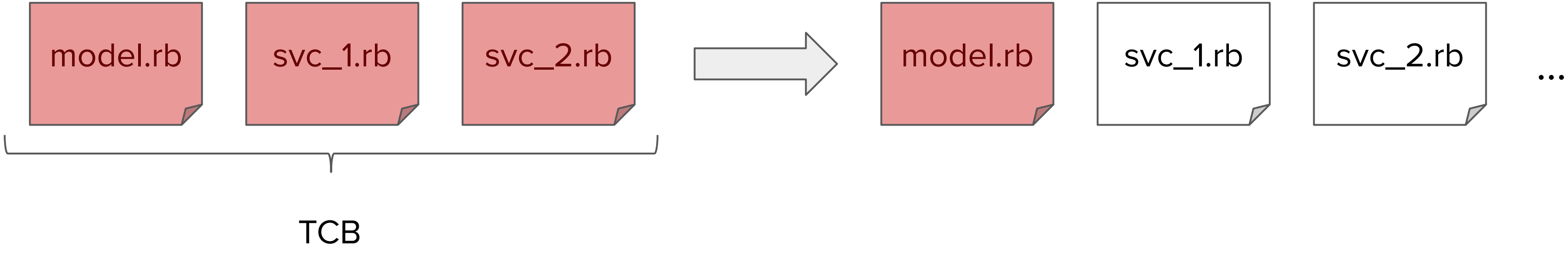
model.rb

svc_1.rb

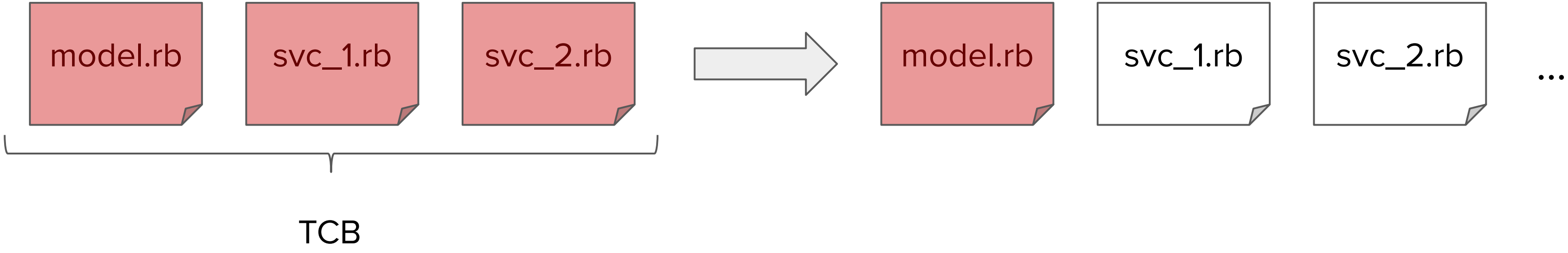
svc_2.rb

TCB

Decrease the amount of code we have to trust



Decrease the amount of code we have to trust



Single statement of policy for each resource

Maintains security in the presence of most bugs

Policy Description Languages - Not a new idea

Hails

Security Policy

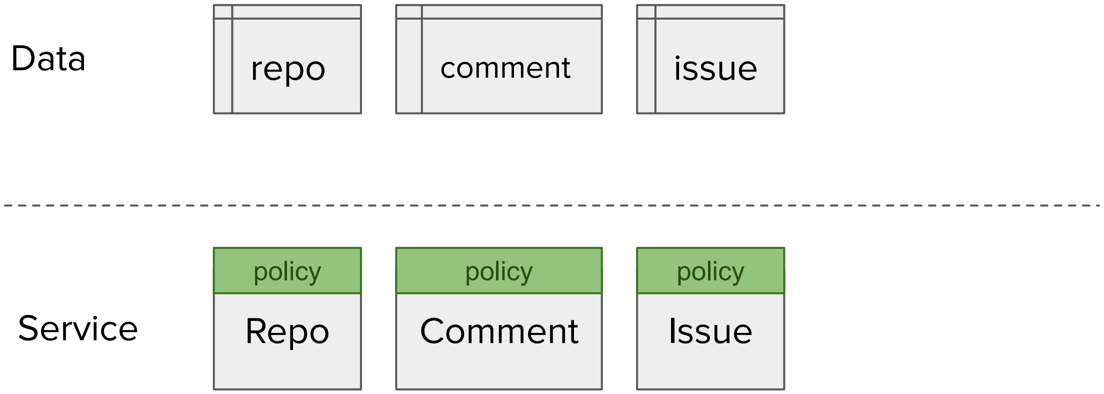
SQL

Row-level Security

Firebase

Security Rules

How it would fix this GitLab Example



Data

repo

comment

issue

Service

policy

Repo

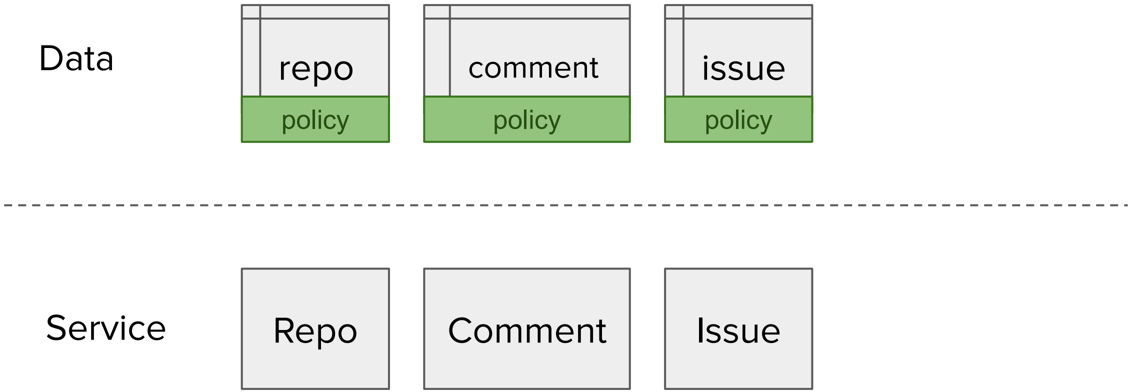
policy

Comment

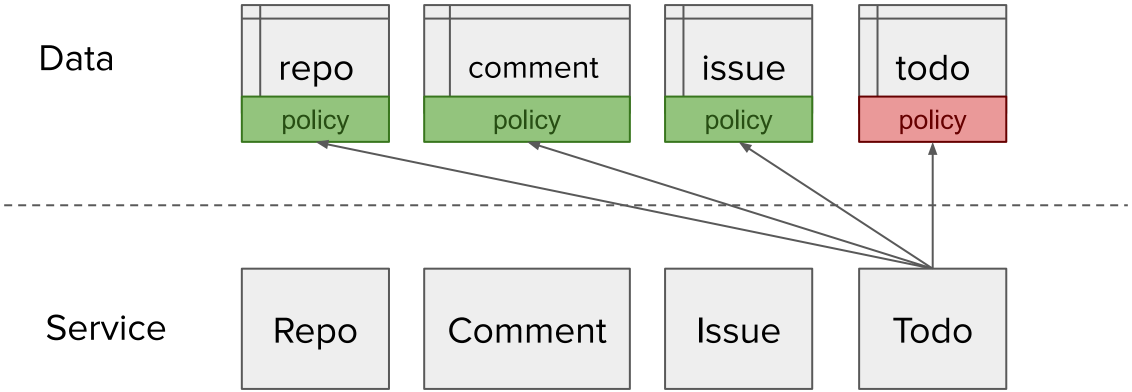
policy

Issue

How it would fix this GitLab Example



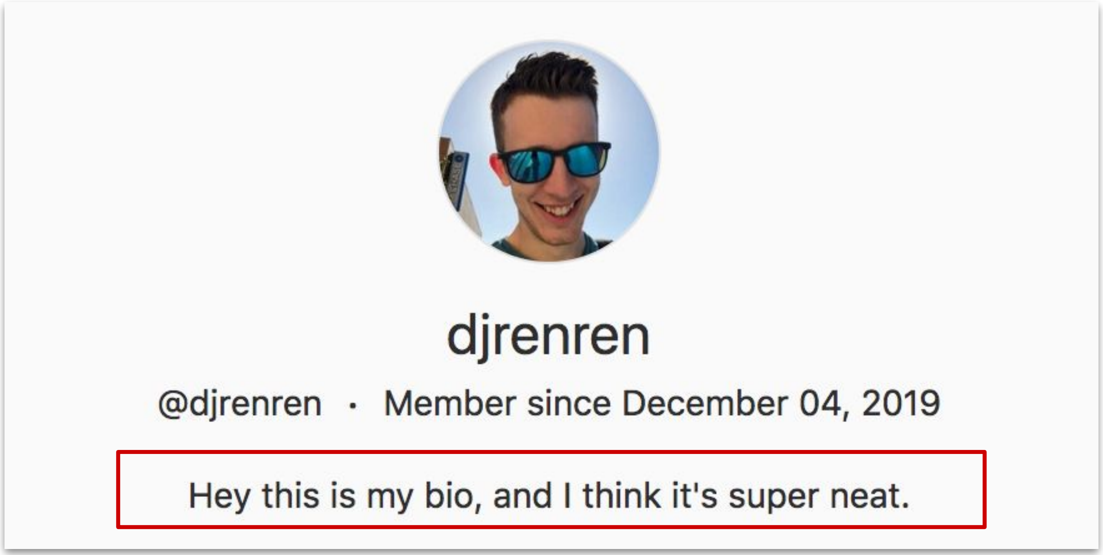
How it would fix this GitLab Example



New app code can’t leak old data!

Runtime protection is not enough

Runtime protection is not enough - Example



djrenren

@djrenren · Member since December 04, 2019

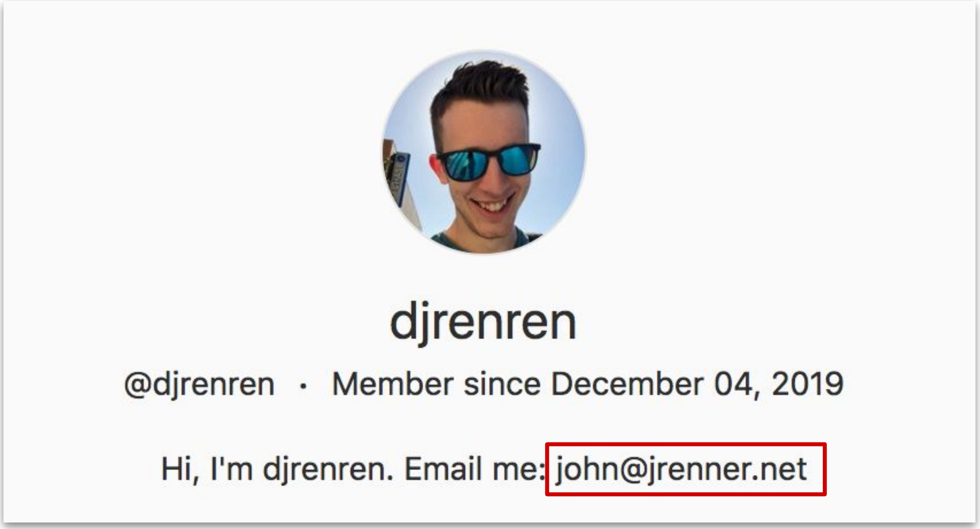
Hey this is my bio, and I think it's super neat.

Let’s add a bio to GitLab profiles

Migrations can also leak

```
ALTER TABLE User
        ADD bio TEXT;
```

```
UPDATE User
   SET bio =
       CONCAT(‘Hi, I’m ’, username,
              ‘. Email me: ’, email)
```



djrenren

@djrenren · Member since December 04, 2019

Hi, I'm djrenren. Email me: john@jrenner.net

We leaked user emails!

We need a unified approach

A DSL to:

- Define the data model
- Specify policies on that model
- Express & reason about changes to both

What do data models and policies look like

```
User {

    name: String {

        read: anyone,

        edit: u -> u.id

    }
}
```

Datatype name (Table)

Field name and type

Policy for field reads

Policy for field edits

Policy function. Returns a list of ids

What do migrations look like

```
User.addField(“bio”, String, u ->
    “Hi, I’m ” + u.name + “. Email me: ” + u.email);
```

What do migrations look like

```
User.addField(“bio”, String, u ->
   “Hi, I’m ” + u.name + “. Email me: ” + u.email);
```

All Operations

```
.addField          addCollection
.removeField       removeCollection
.changeField       renameCollection
.forEach
.create
.update
```

Maintaining policies across migrations

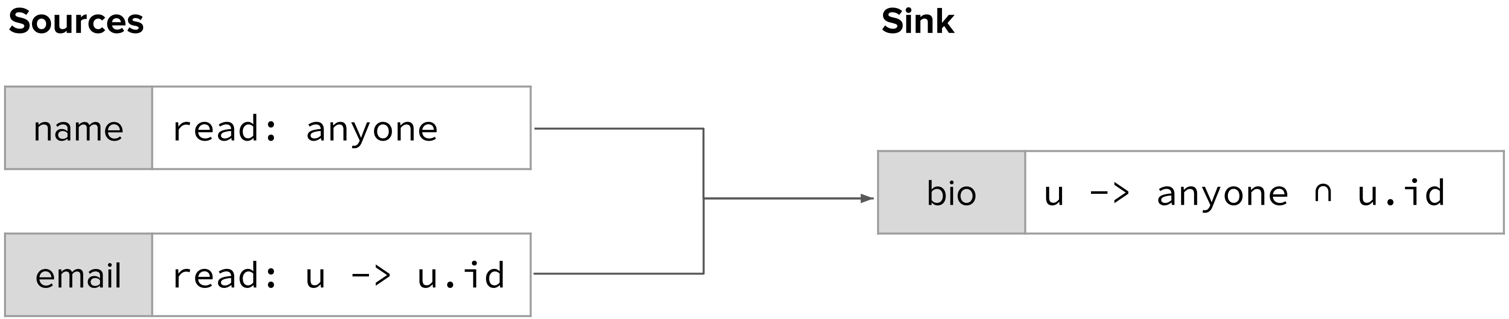
Migrations don’t run as single users

Enforcing policies mid-migration would be difficult

Allow all migrations, generate an “at least as strict” policy

Information flow in migrations

```
User.addField(“bio”, String, u ->

   “Hi, I’m ” + u.name + “. Email me: ” + u.email);
```



Sources

Sink

Automating policy updates from migrations

Field renamed

Rename “contact” to “email”

Automating policy updates from migrations

Field renamed - Replace references with new name

Rename “contact” to “email”

Automating policy updates from migrations

Field renamed - Replace references with new name

Rename “contact” to “email”

Data relocated

Move “email” to a Profile which is 1:1 with User

Automating policy updates from migrations

Field renamed - Replace references with new name

Rename “contact” to “email”

Data relocated - Replace references with traversals

Move “email” to a Profile which is 1:1 with User

Automating policy updates from migrations

Field renamed - Replace references with new name

Rename “contact” to “email”

Data relocated - Replace references with traversals

Move “email” to a Profile which is 1:1 with User

Data modified

Increase `User.permission_rank` by 1

Automating policy updates from migrations

Field renamed - Replace references with new name

Rename “contact” to “email”

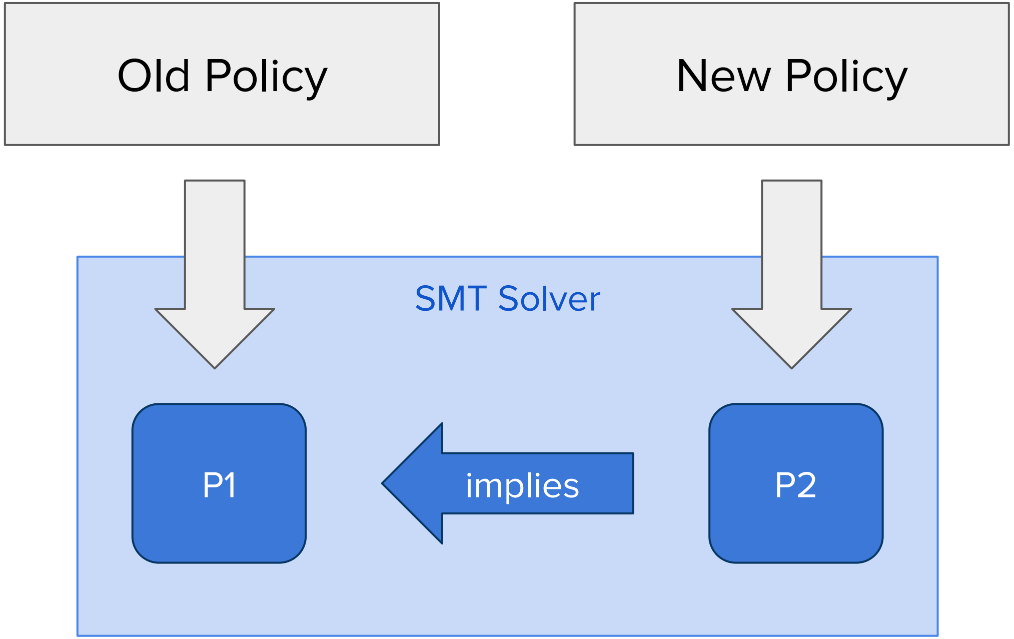
Data relocated - Replace references with traversals

Move “email” to a Profile which is 1:1 with User

Data modified - Replace references with inverse operation

Increase `User.permission_rank` by 1

Preventing leaks due to policy refactoring



New policy is

“at least as strict”

Where are we now?

Done

MVP of policy specification

Verifying policy changes w/ SMT

A few simple migrations

To-do

Expand policy language

Migration flow analysis

Summary

Explicit policy descriptions prevent leaks in app code

We also need to prevent leaks due to migrations

Our DSL addresses both problems