Practical Secure Code Review

(Seth & Ken's Excellent Adventure in Secure Code Review)

2023

INTRODUCTIONS - SETH

Founder of Redpoint Security

Based in Salt Lake City, Utah

Consultant, Programmer, Trainer, Speaker and of course... a lot of code review.

@sethlaw

INTRODUCTIONS - KEN

CTO of DryRun Security

Based out of Virginia

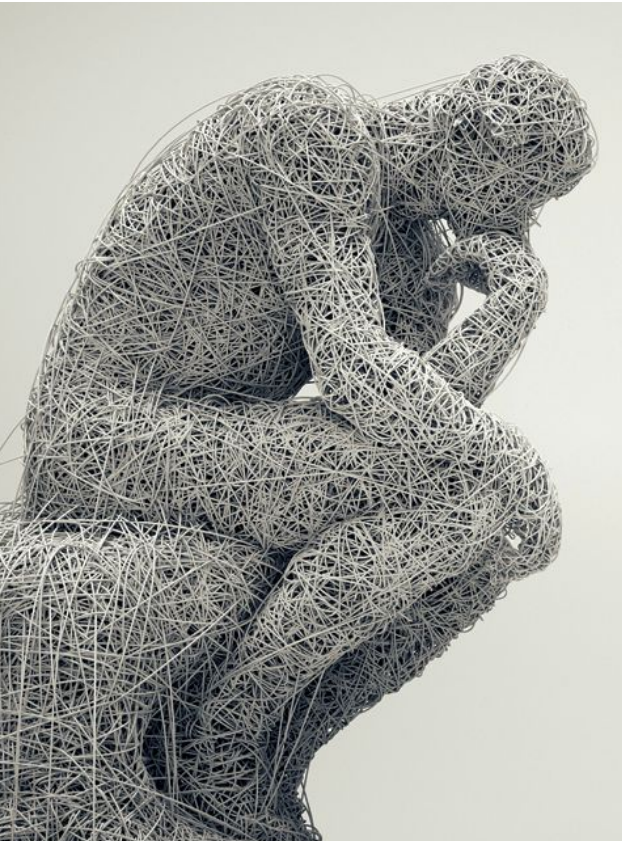
Done a lot of stuff. For realsies.

@cktricky

ABSOLUTE AppSec

ABSOLUTEAPPSEC.COM

Philosophy - Part 1



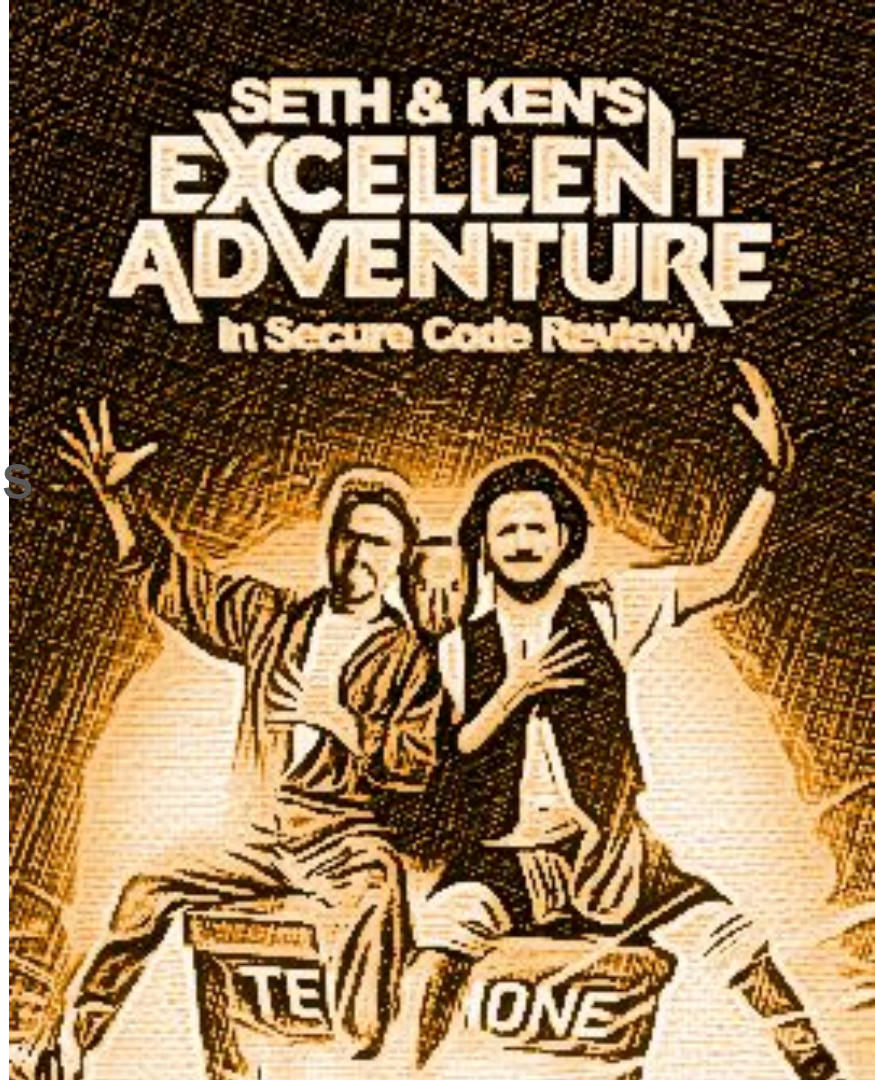
- Increase the likelihood of success.
- Not “find every bug imaginable”.
- Successful defined as:
  - Risk-based (find what matters)
  - Comprehensive
  - Timely
  - Easily-consumed notes & report

PHILOSOPHY - PART 2

- Agnostic Principle
  - Become an expert in becoming an expert
  - Tech stack expertise/proficiency will come with repetition
- Learn a repeatable, systematic approach to finding bugs
  - Caveat: You *will* tweak this approach over time to suit your needs

WHAT TO EXPECT

- Our Experience and Stories
- Framework Nuances
- Interactive Hands-On Exercises
- Practice, practice, practice



LET'S TALK ABOUT APPROACH

Review situations:

- You decide how much time you spend (ideal world)
- You have no time at all (real world)
- Consulting middleground where you have a set time and scoped (hopefully) well or at least semi “okay”

NOTE: THEIR IS NO SITUATION WHERE YOU SHOULD “JUST SCAN IT”

First, let's talk about what we're doing here

This course is intended to do the following, at a high level:

1. Explain risk-based code review principles
2. Show what to look for:
    a. Authorization and Authentication issues
    b. Vulnerabilities specific to the type of app
    c. General application security vulnerabilities
3. Show where to look:
    a. Framework nuances/issues
    b. Map attack surface
4. Discuss automation/tooling and when it is applicable

Also, let's talk about what we aren't doing

This course is ***not*** intended to:

1. Teach you the OWASP Top 10
2. Review all of the known web vulnerabilities
3. Exercises for finding vulnerabilities in running applications (ie - dynamic analysis)
4. Be a primer on building secure applications
5. Cover all the ways that vulnerabilities manifest in code.

REPOSITORIES

- https://github.com/IMA-WorldHealth/bhima - Node App
- https://github.com/railsbridge/bridge_troll - RAILS App
- https://github.com/absoluteappsec/handouts - templates
- https://github.com/absoluteappsec/skea_django
- https://github.com/absoluteappsec/skea_node
- https://github.com/sethlaw/vtm - Vulnerable Task Manager

LET'S TALK ABOUT THE OWASP IN THE ROOM

- How does the OWASP Top 10 relate?
- What resources does OWASP provide?
- How does this course differ from the OWASP Code Review Methodology?

Secure Code Review Methodology

Overview

SECURE CODE REVIEW METHODOLOGY

1. Application Overview & Risk Assessment
2. Information Gathering
3. Checklist Creation
4. Perform Reviews

Checklists

- Authorization
- Authentication
- Auditing
- Injection
- Cryptographic
- Configuration
- Reporting

THE CIRCLE-K FRAMEWORK

1. Open a file, take notes :-)
2. Identify the application purpose
3. Map the application
4. Brainstorm risks to the application
5. **Build list of review items**
6. Perform all reviews
7. Double back (3-6)

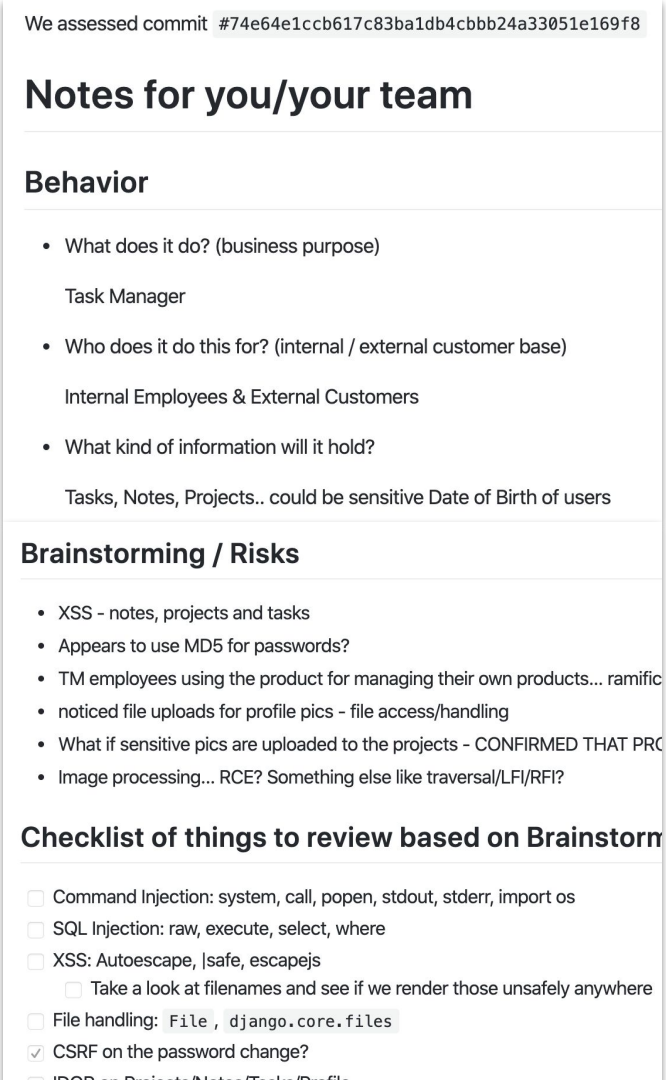
General Principles

- Give yourself adequate time
- Work in small chunks
- Stay on task with current objective
- Don’t make it personal
- Ask questions
- Framework/Code documentation is your friend
- Build the code
- Run the tests

NOTE TAKING

NOTE TAKING

- Format
  - Commit # (if available)
  - Notes from the review
  - Route mapping
  - “Threat model” details (really, rambling thoughts about what could go wrong”
    i. PAIR WITH OTHERS!!!
  - Checklist of items you’ve checked for based on your threat model but also your normal “things” you should look at



We assessed commit `#74e64e1ccb617c83ba1db4cbbb24a33051e169f8`

Notes for you/your team

Behavior

- What does it do? (business purpose)

  Task Manager

- Who does it do this for? (internal / external customer base)

  Internal Employees & External Customers

- What kind of information will it hold?

  Tasks, Notes, Projects.. could be sensitive Date of Birth of users

Brainstorming / Risks

- XSS - notes, projects and tasks
- Appears to use MD5 for passwords?
- TM employees using the product for managing their own products... ramific
- noticed file uploads for profile pics - file access/handling
- What if sensitive pics are uploaded to the projects - CONFIRMED THAT PR
- Image processing... RCE? Something else like traversal/LFI/RFI?

Checklist of things to review based on Brainstorm

- [ ] Command Injection: system, call, popen, stdout, stderr, import os
- [ ] SQL Injection: raw, execute, select, where
- [ ] XSS: Autoescape, |safe, escapejs
  - [ ] Take a look at filenames and see if we render those unsafely anywhere
- [ ] File handling: `File`, `django.core.files`
- [x] CSRF on the password change?

REVIEW TYPES

REVIEW TYPES

1. FULL ASSESSMENT 💯
2. ONLY NEW CODE 👍
3. PULL REQUEST REVIEW 🔎
4. SCANNER FINDINGS 🪲

1. Everything is applicable !!
2. Depends 👀… seen this app before?
3. Context specific bits (authz functions, logic processing, db queries, etc.)
4. Meh 🤷

APPLICATION OVERVIEW & RISK ASSESSMENT

APPLICATION OVERVIEW & RISK ASSESSMENT

Objectives:

- Identify what is most at risk
- Identify the most likely areas where opportunities to exploit said risks might manifest themselves
- Map the attack surface of an application

APPLICATION OVERVIEW & RISK ASSESSMENT

Common ways to gather information:

- README.md
- Unit Tests
- Documentation:
  - API Docs (Swagger, PandaDoc, etc.)
  - Wiki
  - /docs folder
- Database Schema & development data “seed” files
- Package library files (eg: Gemfile, Package.json, pom.xml)

```
~/code/vtm/taskManager master cat models.py
# Vulnerable Task Manager

import datetime

from django.contrib.auth.models import User

from django.utils import timezone
from django.db import models


class UserProfile(models.Model):
    user = models.OneToOneField(User,on_delete=models.CASCADE)
    image = models.CharField(max_length=3000, default="")
    reset_token = models.CharField(max_length=7, default="")
    reset_token_expiration = models.DateTimeField(default=timezone.now)
    dob = models.CharField(max_length=8, default="00/00/00")
```

APPLICATION OVERVIEW & RISK ASSESSMENT

Build a portrait of the application

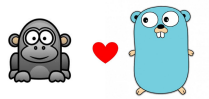
- Behavior Profile
- Technology Stack
- App Archeology

BEHAVIOR PROFILING

- What does it do? (business purpose)
- Who does it do this for? (internal / external customer base)
- What kind of information will it hold?
- What are the different types of roles?
- What aspects concern your client/customer/staff the most?

TECHNOLOGY STACK

- Framework & Language - django 


- 3rd party components:
  - Supporting libraries (rubygem, npm, jar, etc.)
  - JavaScript widgets - (marketing tracking, sales chat widget)
  - Service to Service
- Datastore - Postgresql, MySQL, Memcache, Redis, Mongodb

Application Archeology

- Look at the documentation. Examples:
  - Using SSO? SAML/OAuth?
  - What third-party elements does communicate with? (eg: external reporting, billing)
  - Custom authentication schemas - for instance - different auth for API versus web interface
- Git allows you to do a little spelunking
- Unit Tests
  - Learn about the data endpoints expect to receive
  - Learn about expected behavior
  - Often can find the way to form/craft a request

BRAINSTORMING RISKS!

- Use the information we've collected
- Work with one or more (preferably more) people because diversity of thought equates to comprehensive checklists
- Often new risks will pop up the more you perform your review… expect and embrace it

APPLICATION OVERVIEW & RISK ASSESSMENT

https://github.com/IMA-WorldHealth/bhima

```
));

const WEAK_PASSWORD = 'hello';
const MEDIUM_PASSWORD = 'L0b1Ec0simba';
const STRONG_PASSWORD = 'N@pM@ch3N#L1my3B0ndy3@!';

beforeEach(inject((_PasswordMeterService_, _SessionService_, _MockDataService_) => {
  PasswordMeterService = _PasswordMeterService_;
  Session = _SessionService_;

  const user = _MockDataService_.user();
  const project = _MockDataService_.project();
  const enterprise = _MockDataService_.enterprise();
  Session.create(user, enterprise, project);

  // make sure password validation is on
  Session.enterprise.settings.enable_password_validation = true;
}));
```

```
it('#counter() should return -1 for no password', () => {
  const count = PasswordMeterService.counter();
  expect(count).to.equal(-1);
});

it('#validate() should return false for no password', () => {
  const validate = PasswordMeterService.validate();
  expect(validate).to.equal(false);
});

it('#counter() should return 0 for a weak password', () => {
  const count = PasswordMeterService.counter(WEAK_PASSWORD);
  expect(count).to.equal(0);
});

it('#validate() should return false for a weak password', () =>
  const validate = PasswordMeterService.validate(WEAK_PASSWORD);
  expect(validate).to.equal(false);
});

it('#counter() should return 3 for a medium password', () => {
  const count = PasswordMeterService.counter(MEDIUM_PASSWORD);
  expect(count).to.equal(3);
});

it('#validate() should return true for a medium password', () =>
  const validate = PasswordMeterService.validate(MEDIUM_PASSWORD
  expect(validate).to.equal(true);
});
```

APPLICATION OVERVIEW & RISK ASSESSMENT - EXAMPLE

README.md | Preview: 'Preview: 'README.md''

skea_django

SKEA == Seth & Ken's Excellent Adventures (in Code Review) A small django application for use in demo-ing django routes, etc.

Requirements

Python3 + Django > 2.1

Quickstart

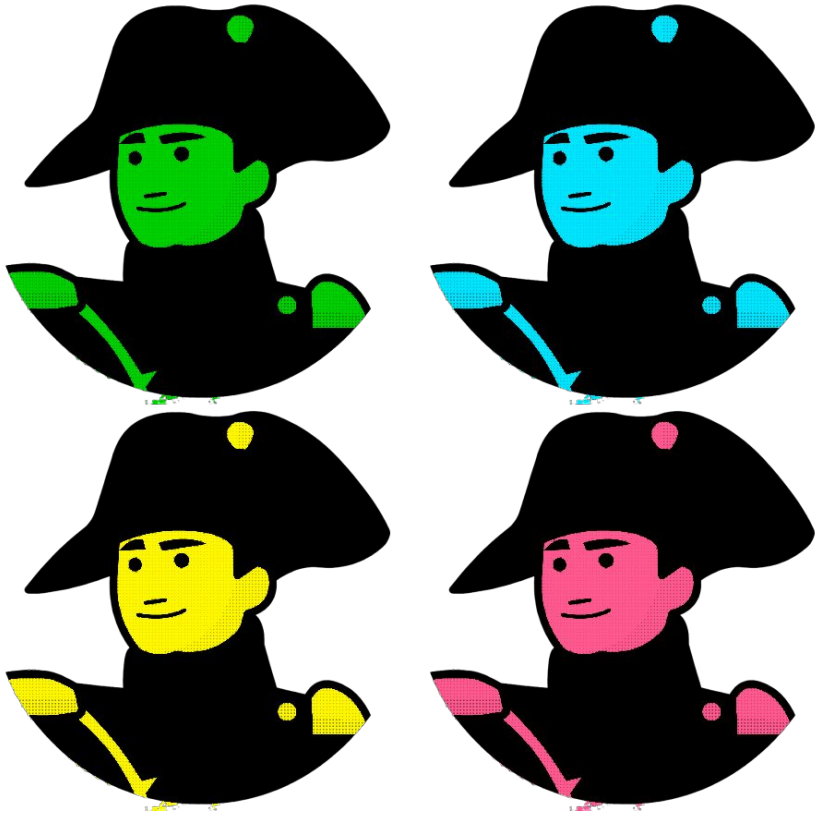
---

1. `pip3 install -r requirements.txt`
2. `./manage.py migrate`
3. `./manage.py runserver`
4. Navigate to http://localhost:8000/

APPLICATION OVERVIEW & RISK ASSESSMENT - EXERCISE

NAPOLEON

1. **bhima**
    a. Business Purpose
    b. Tech Stack
    c. Application Risk
    d. Anything else?



NAPOLEON EXERCISE - POST-MORTEM

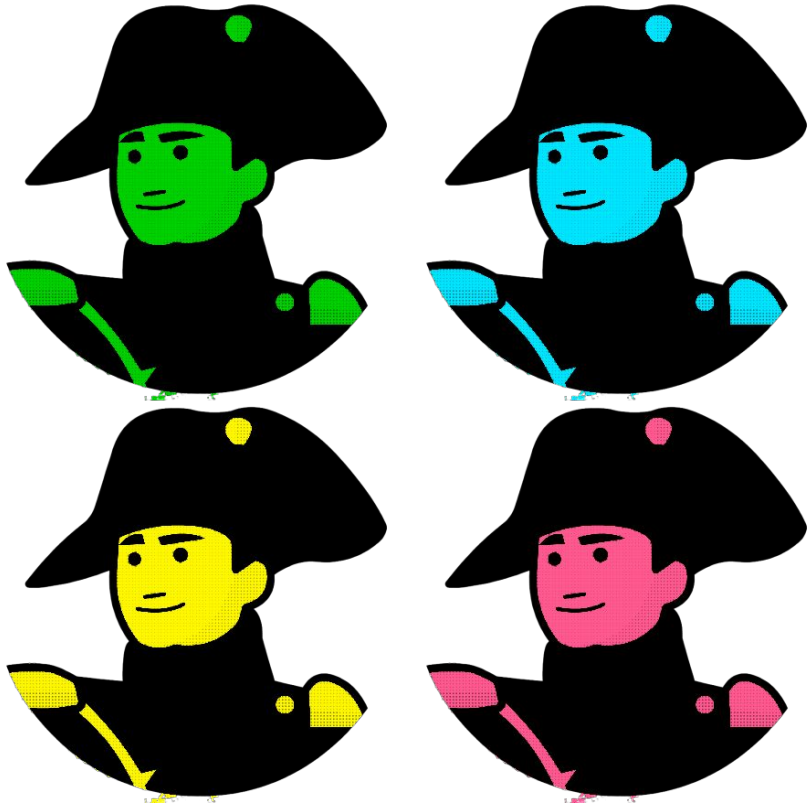
At a minimum, these items were of note:

- Business Purpose
  - This is a community management platform for Tech Communities across the world. The inspiration being the personal experiences of Community Organizers.
- Tech Stack
  - Node.js / Express
  - Angular
- Documentation
  - https://github.com/IMA-WorldHealth/bhima/wiki
  - https://docs.bhi.ma/en/
- Risks
  - Patient Data
  - Employee Payroll Information

NAPOLEON EXERCISE - POST-MORTEM

Other important security portions of the app:

- Express Configuration -
  - server/config/express.js
  - Uses default helmet.js
- User Authentication
  - server/controllers/auth.js
- Admin folder
  - Logic for admin logins - server/controllers/admin/users/index.js



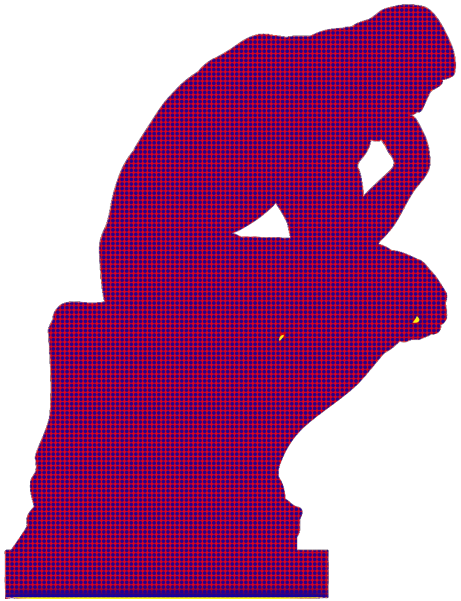
INFORMATION GATHERING

INFORMATION GATHERING

"THE MOST IMPORTANTEST THING IS TO UNDERSTAND THE APP"

"- @sethlaw"

- @cktricky

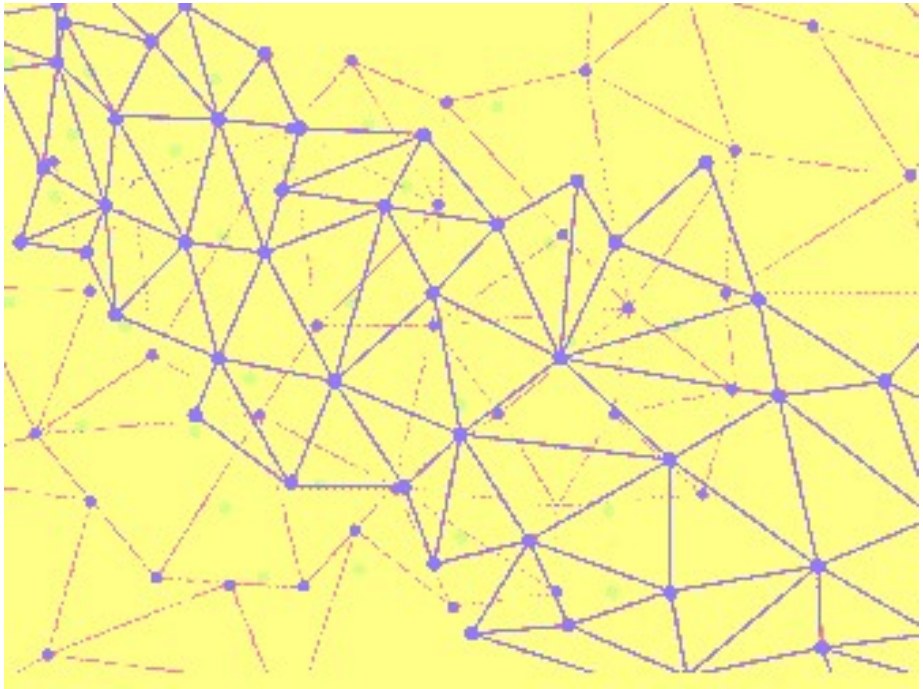


INFORMATION GATHERING

Steps

1. Create Application Map

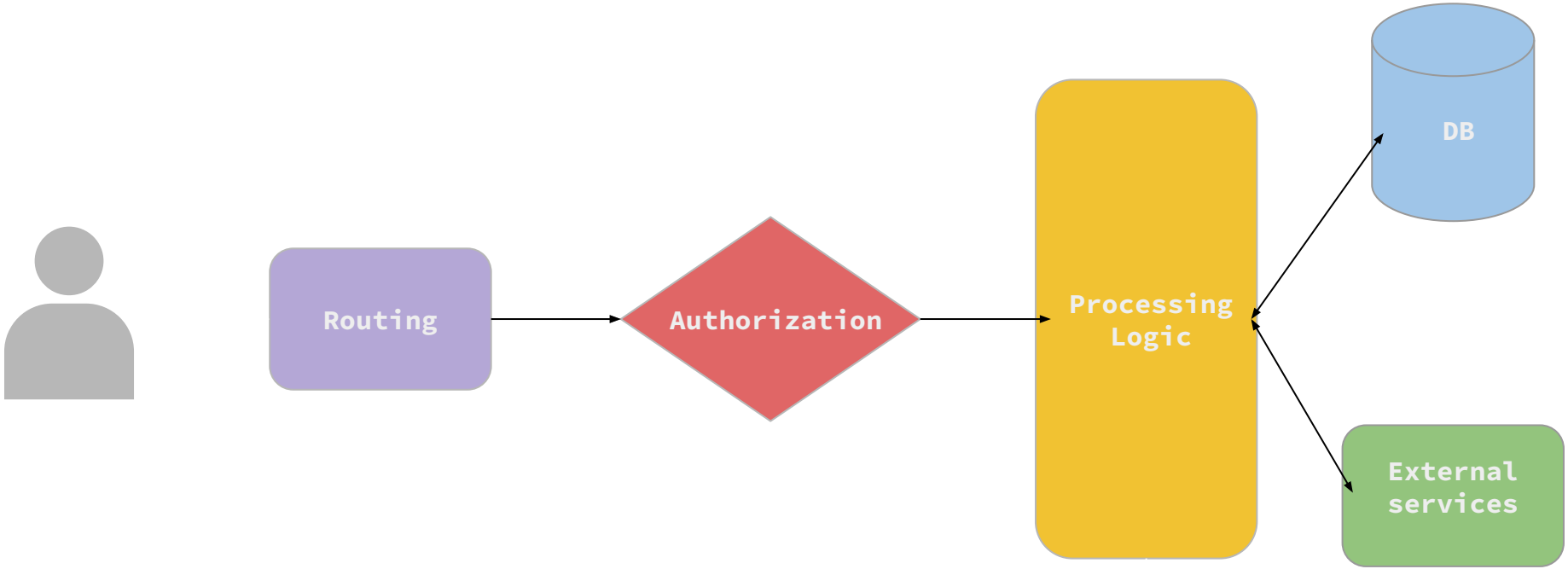
2. Identify Authorization Functions



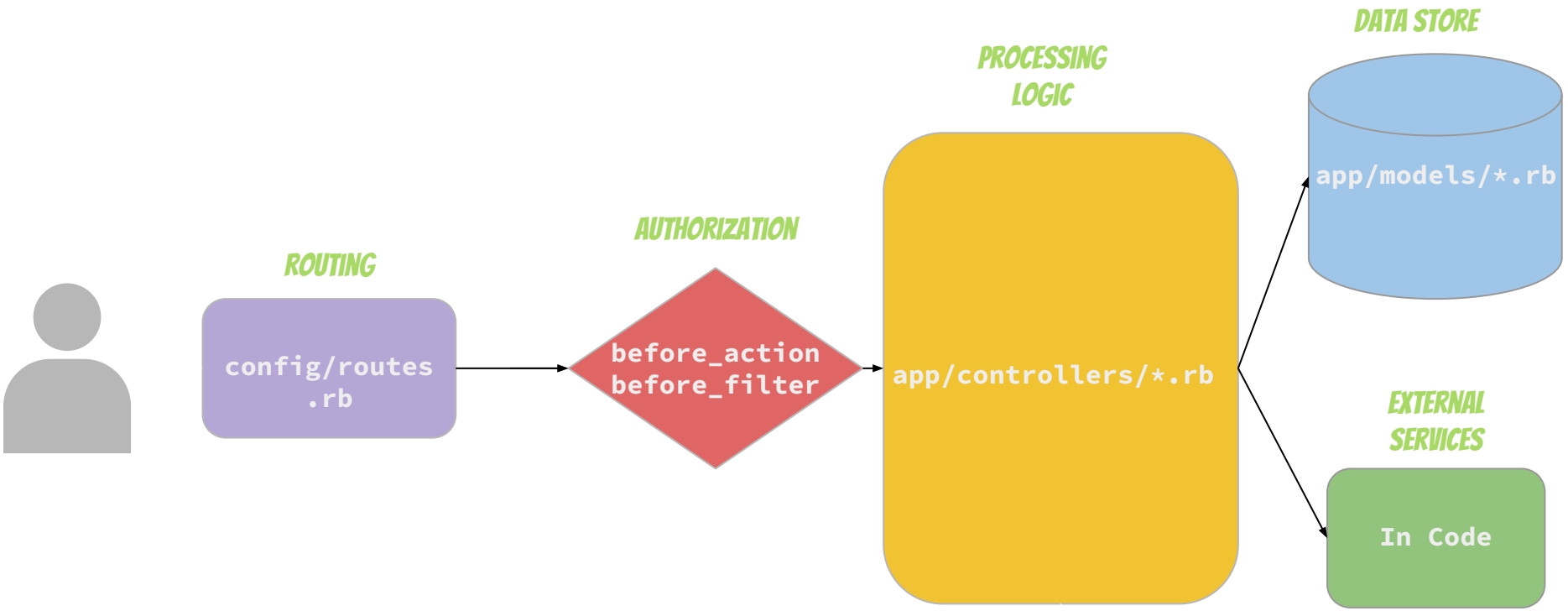
MAPPING

Information Gathering - Create a Map

- Identify endpoints, typical examples:
  - Rails = config/routes.rb - **`rake routes`**
  - Django= urls.py - **`manage show_urls`**
  - Node.js = index.js - gets a little more tricky…
- Endpoints typically have at least three qualities
  - Authorization Filter
  - Logic processing
  - Datastore access



TYPICAL APPLICATION FLOW



RAILS APPLICATION FLOW

RAILS - MAP

- Run **`rake routes`** If you can!

1. cktricky@Kens-MacBook-Pro: ~/code/railsgoat (zsh)

```
cktricky@Kens-MacBook-Pro ~/code/railsgoat master rake routes
                  Prefix Verb   URI Pattern                                       Controller#Action
                   login GET    /login(.:format)                                  sessions#new
                  signup GET    /signup(.:format)                                 users#new
                  logout GET    /logout(.:format)                                 sessions#destroy
         forgot_password GET    /forgot_password(.:format)                        password_resets#forgot_password
                         POST   /forgot_password(.:format)                        password_resets#send_forgot_password
         password_resets GET    /password_resets(.:format)                        password_resets#confirm_token
                         POST   /password_resets(.:format)                        password_resets#reset_password
           dashboard_doc GET    /dashboard/doc(.:format)                          dashboard#doc
                sessions GET    /sessions(.:format)                               sessions#index
                         POST   /sessions(.:format)                               sessions#create
             new_session GET    /sessions/new(.:format)                           sessions#new
            edit_session GET    /sessions/:id/edit(.:format)                      sessions#edit
                 session GET    /sessions/:id(.:format)                           sessions#show
                         PATCH  /sessions/:id(.:format)                           sessions#update
                         PUT    /sessions/:id(.:format)                           sessions#update
                         DELETE /sessions/:id(.:format)                           sessions#destroy
user_account_settings GET    /users/:user_id/account_settings(.:format)          users#account_settings
user_retirement_index GET    /users/:user_id/retirement(.:format)                retirement#index
```

RAILS - MAP

```
forgot_password GET    /forgot_password(.:format)                password_resets#forgot_password
```

- Four parts
  - Path name (eg: forgot_password_path)
  - HTTP Verb (eg: GET)
  - HTTP Path (eg(s): forgot_password, forgot_password.json, forgot_password.xml)
  - Controller and Action (eg: Controller = password_resets, Action = forgot_password)

RAILS - MAP

```
forgot_password GET    /forgot_password(.:format)          password_resets#forgot_password
```

- Four parts
  - Path name (eg: forgot_password_path)
  - HTTP Verb (eg: GET)
  - HTTP Path (eg(s): forgot_password, forgot_password.json, forgot_password.xml)
  - Controller and Action (eg: Controller = password_resets, Action = forgot_password)

RAILS - MAP

- If you cannot run rake routes, you'll have to review the config/routes.rb file:

routes.rb — ~/code/ra

routes.rb

```
# frozen_string_literal: true
Railsgoat::Application.routes.draw do

  get "login" => "sessions#new"
  get "signup" => "users#new"
  get "logout" => "sessions#destroy"

  get "forgot_password" => "password_resets#forgot_password"
  post "forgot_password" => "password_resets#send_forgot_password"
  get "password_resets" => "password_resets#confirm_token"
  post "password_resets" => "password_resets#reset_password"

  get "dashboard/doc" => "dashboard#doc"

  resources :sessions

  resources :users do
    get "account_settings"

    resources :retirement
    resources :paid_time_off
    resources :work_info
    resources :performance
    resources :benefit_forms
    resources :messages
```

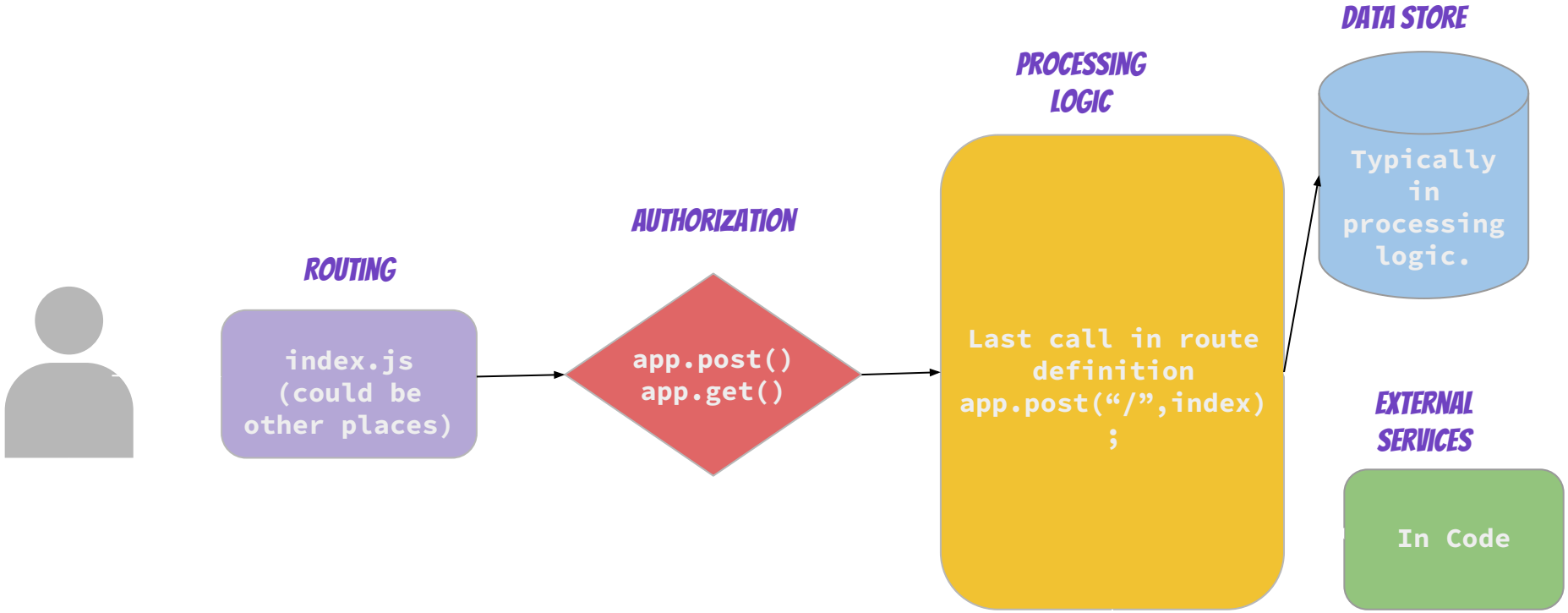
RAILS - ROUTING WEIRDNESS ALERT

- What do **`PUT`**, **`PATCH`**, **`DELETE`**, and **`POST`** all have in common?
- That's right, they are all **`POST`** requests
- Rails uses the **`_method`** parameter to determine what kind of request it is

RAILS - ROUTING WEIRDNESS ALERT

```
POST /join HTTP/1.1
Host: github.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0)
Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://github.com/join?source=header-home
Content-Type: application/x-www-form-urlencoded
Content-Length: 233
Connection: close
Cookie: logged_in=no
Upgrade-Insecure-Requests: 1

utf8=%E2%9C%93&authenticity_token=1YJBp5UZPnafW2qSy6GhEgg4ob05APDlMVUbmvLCCv
VqQS5JYA5cuVOcLWOS%2FDELy7z8J1AeH1TyxGCWojSxcw%3D%3D&user%5Blogin%5D=cktric
ky-example&user%5Bemail%5D=cktricky-example%40skea.com&user%5Bpassword%5D=d
olphin1&_method=patch
```



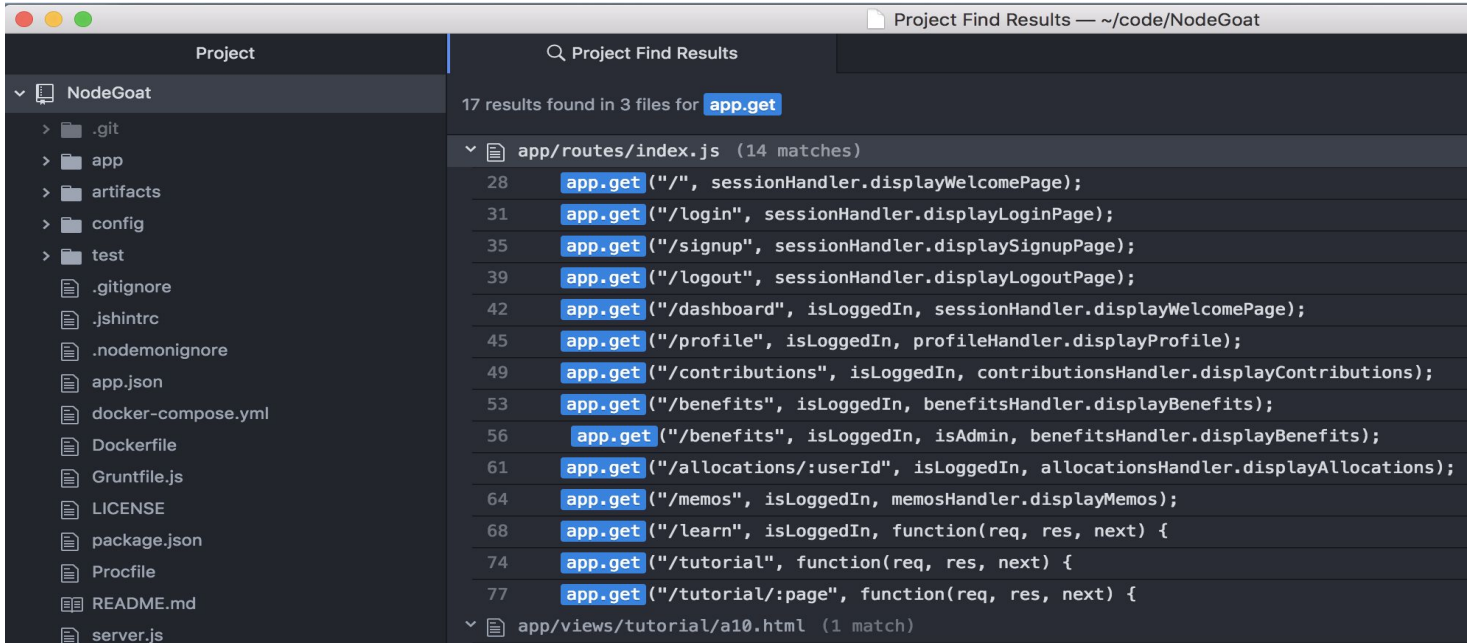
NODE.JS/EXPRESS APPLICATION FLOW

NODE.JS/EXPRESS - MAP

- Our formula used to be super basic, we'd search for:
  - `app.get`
  - `app.post`
  - `app.delete`
- Problem is that routes can be defined in a myriad of ways and using functions other than the above
- Best to ask Express directly, we show a version of this here: https://github.com/absoluteappsec/handouts/blob/master/node_js_generic_checks.md#print-all-routes

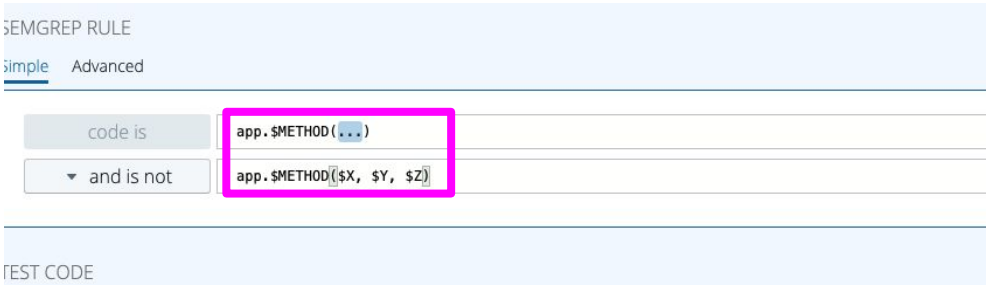
NODE.JS/EXPRESS - MAP

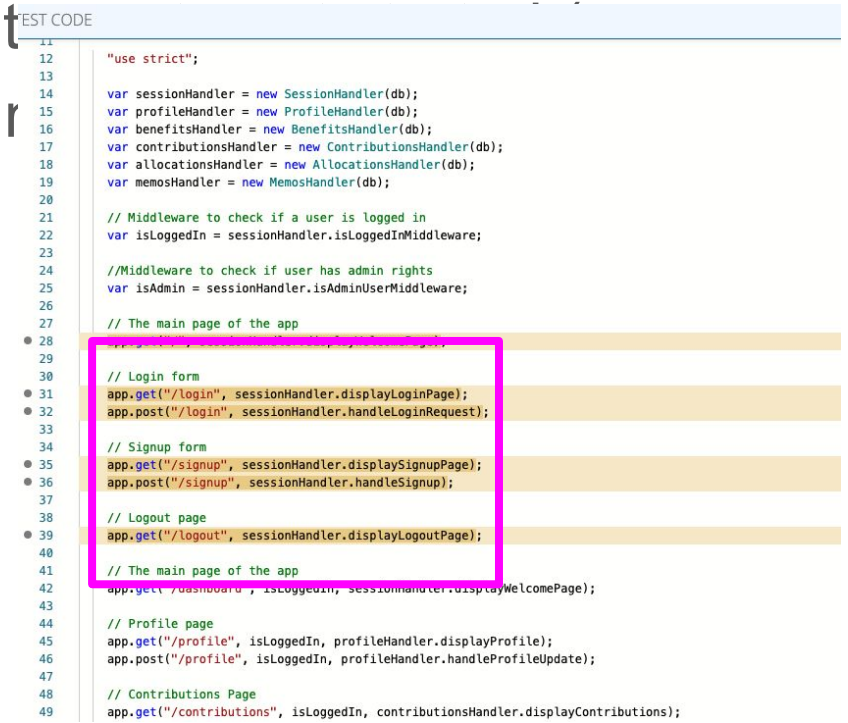
Here is an example from Nodegoat, I downloaded it, and searched for app.get



NODE.JS/EXPRESS - SEMGREP

Semgrep is a fast, easy, lightweight
don't endorse and it is free for comm



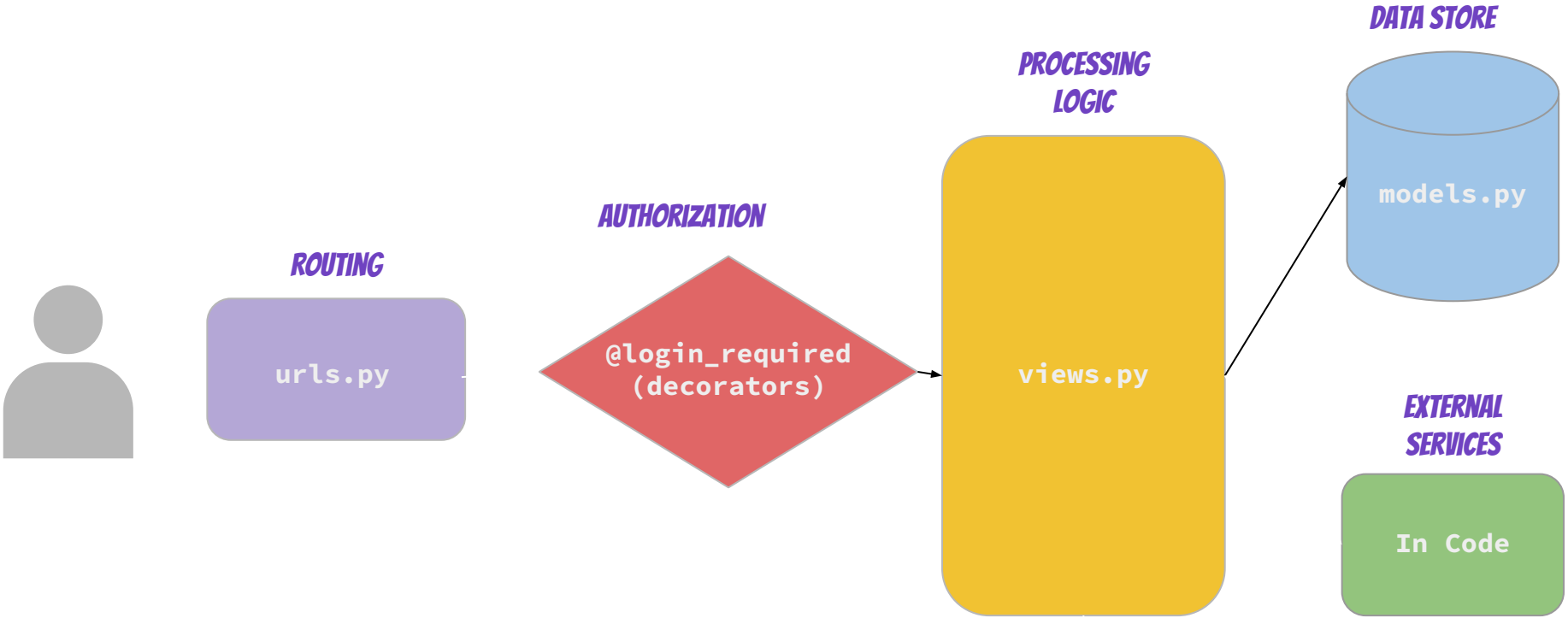


DEMO TIME! (SEMGREP)

NODE.JS/EXPRESS - MAP

Route definition, destructed

```
42    app.get("/dashboard", isLoggedIn, sessionHandler.displayWelcomePage);
```



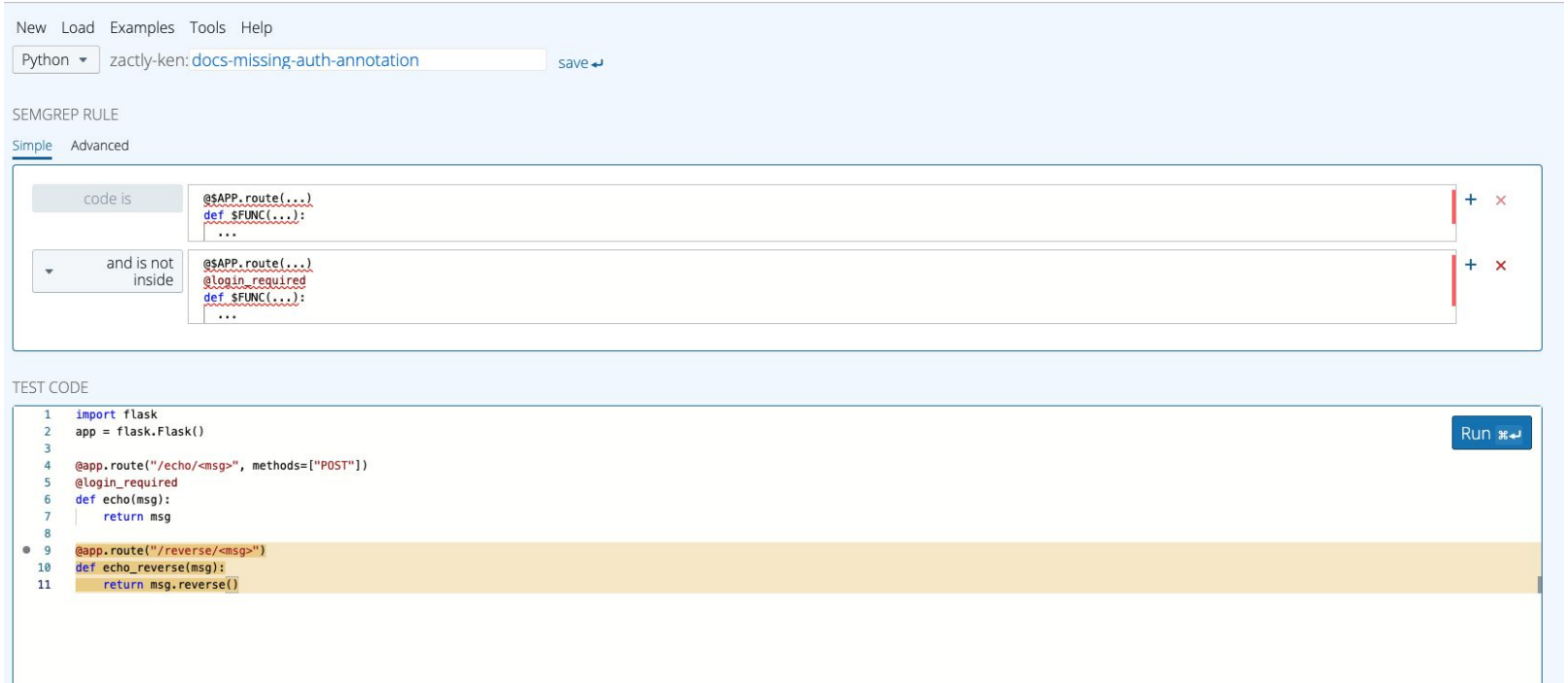
DJANGO APPLICATION FLOW

DJANGO - MAP

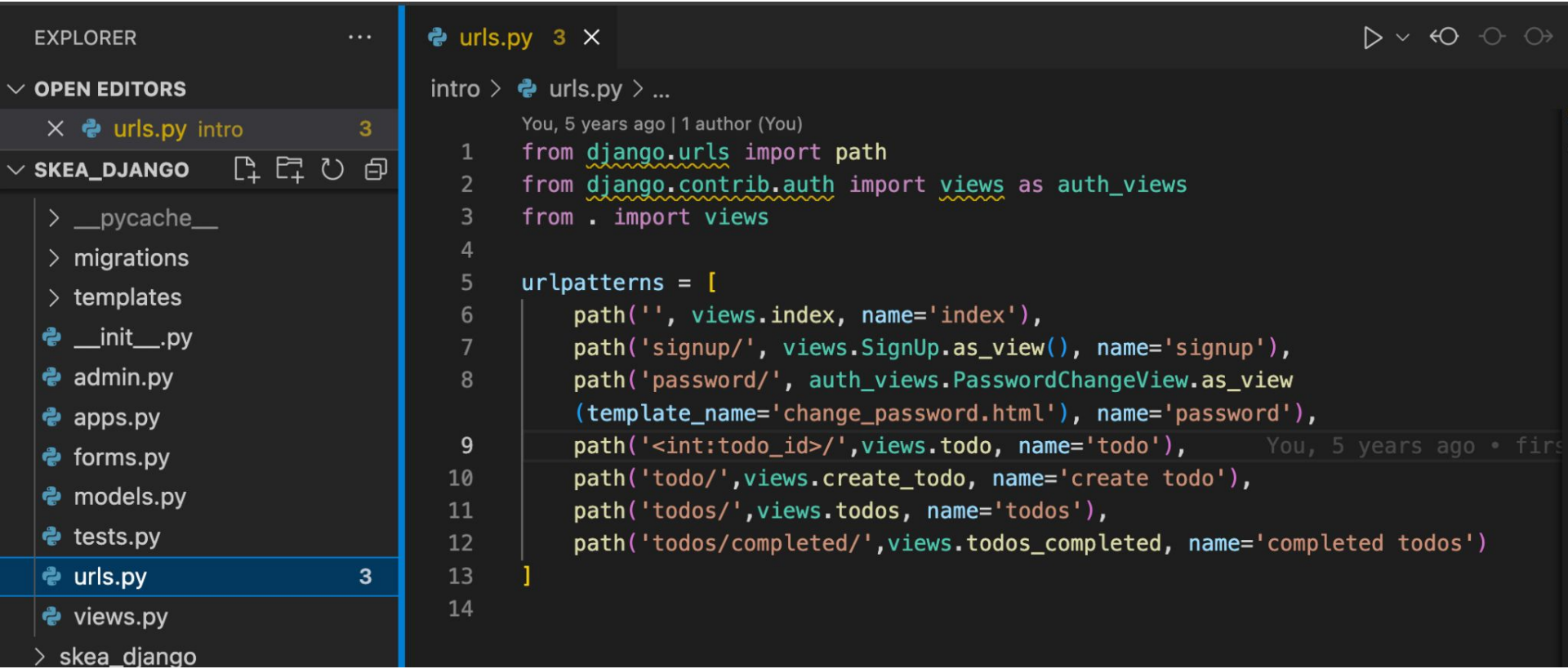
```
cktricky@Kens-MBP ~/code/vtm master ./manage.py show_urls                    ✓ 5281 01:08:18
/       taskManager.views.index index   login_required
/admin/ django.contrib.admin.sites.index        admin:index
/admin/<app_label>/     django.contrib.admin.sites.app_index    admin:app_list
/admin/auth/group/      django.contrib.admin.options.changelist_view    admin:auth_group_changelist
/admin/auth/group/<path:object_id>/     django.views.generic.base.RedirectView
/admin/auth/group/<path:object_id>/change/      django.contrib.admin.options.change_view        admin:auth_group_change
/admin/auth/group/<path:object_id>/delete/      django.contrib.admin.options.delete_view        admin:auth_group_delete
/admin/auth/group/<path:object_id>/history/     django.contrib.admin.options.history_view       admin:auth_group_history
/admin/auth/group/add/  django.contrib.admin.options.add_view   admin:auth_group_add
/admin/auth/group/autocomplete/ django.contrib.admin.options.autocomplete_view  admin:auth_group_autocomplete
/admin/auth/user/       django.contrib.admin.options.changelist_view    admin:auth_user_changelist
/admin/auth/user/<id>/password/ django.contrib.auth.admin.user_change_password  admin:auth_user_password_change
/admin/auth/user/<path:object_id>/      django.views.generic.base.RedirectView
/admin/auth/user/<path:object_id>/change/       django.contrib.admin.options.change_view        admin:auth_user_change
/admin/auth/user/<path:object_id>/delete/       django.contrib.admin.options.delete_view        admin:auth_user_delete
/admin/auth/user/<path:object_id>/history/      django.contrib.admin.options.history_view       admin:auth_user_history
/admin/auth/user/add/   django.contrib.auth.admin.add_view      admin:auth_user_add
/admin/auth/user/autocomplete/  django.contrib.admin.options.autocomplete_view  admin:auth_user_autocomplete
/admin/jsi18n/  django.contrib.admin.sites.i18n_javascript      admin:jsi18n
/admin/login/   django.contrib.admin.sites.login        admin:login
/admin/logout/  django.contrib.admin.sites.logout       admin:logout
/admin/password_change/ django.contrib.admin.sites.password_change      admin:password_change
/admin/password_change/done/    django.contrib.admin.sites.password_change_done admin:password_change_done
```

DJANGO - MAP (*SEMGREP)

https://semgrep.dev/s/minusworld:docs-missing-auth-annotation



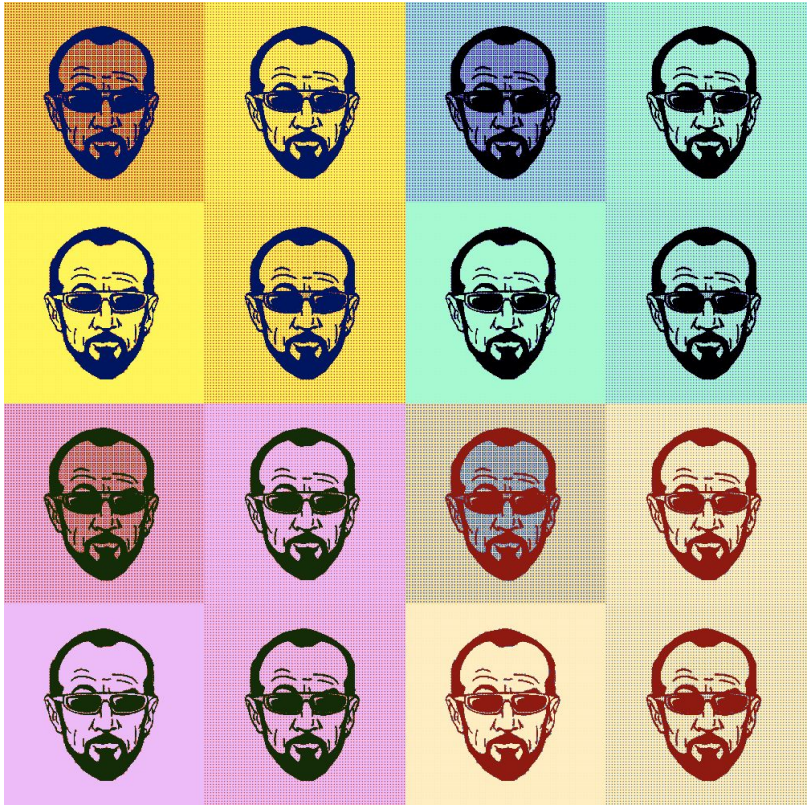
MAPPING EXAMPLE



MAPPING EXERCISE

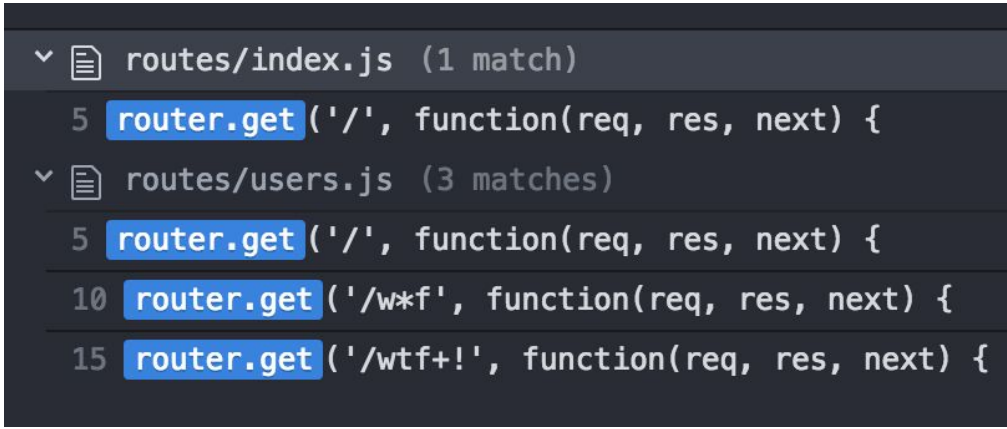
RUFUS

1. **skea_node**
    a. Perform Info Gathering
        i. Tech Stack/Purpose/Risk
    b. Map Routes
2. **bhima**
    a. Map Routes
        i. At least 10



RUFUS - POST-MORTEM

- Notes should include:
  - Commit #
  - End points shown here
  - Any else you found interesting, such as:
    - auth function doesn't do anything
    - debug statement in our app and its printing env variables



AUTHORIZATION FUNCTIONS

Information Gathering - Authorization Functions

- A later step is dedicated to creating authorization checks
- This is about getting to know the application better
- Get to know how users are identified/authorized to perform actions

Information Gathering - Authorization Functions

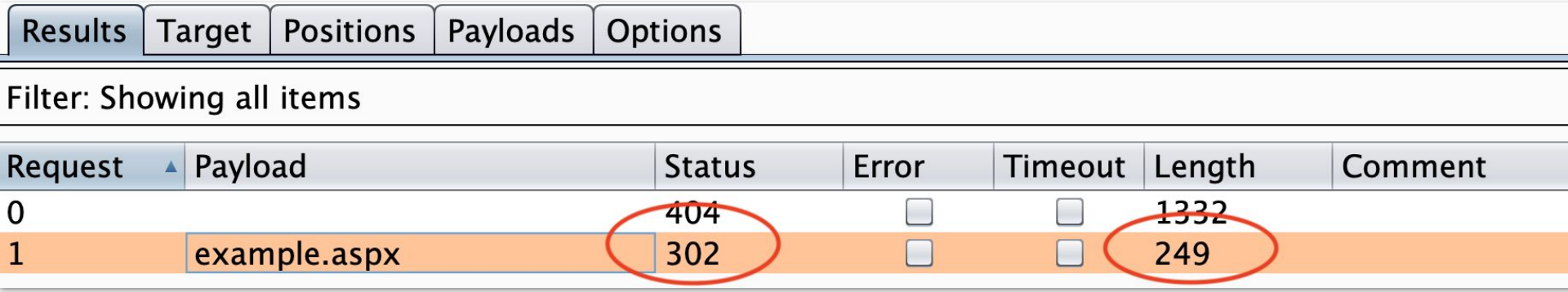
- Patterns & Anti-patterns
  - How do we identify the user? eg: Session, Token, Basic Auth
  - What is the purpose? Authenticated users, Role check, or something else?
- Identify what could go wrong
  - User-supplied parameters (IDOR)
  - CSRF
  - Client-side cookies, JWTs etc. (just general wonkiness in persistence)

Story Time:

Redirection & Authorization in .NET

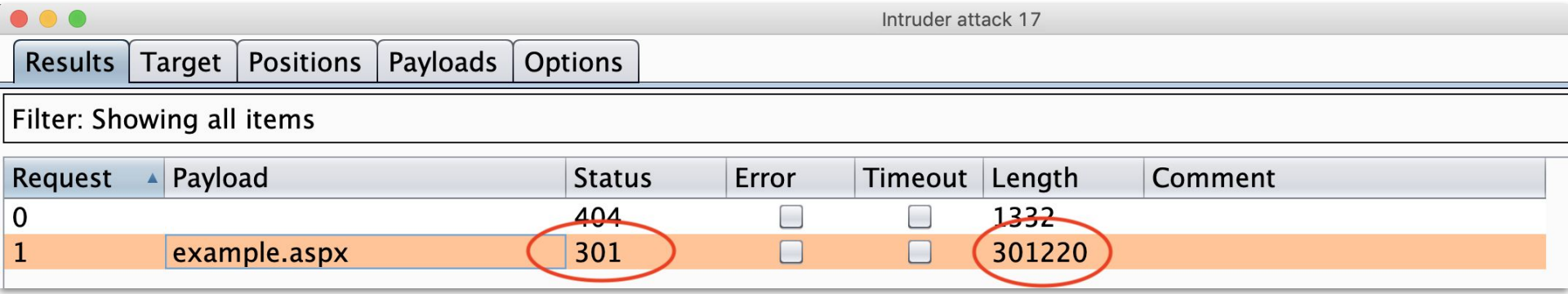
REDIRECTION & AUTHORIZATION ISSUE IN .NET

NORMAL



REDIRECTION & AUTHORIZATION ISSUE IN .NET

DEFINITELY NOT NORMAL



REDIRECTION & AUTHORIZATION ISSUE IN .NET

Redirect(String, Boolean)

Redirects a client to a new URL. Specifies the new URL and whether execution of the current page should terminate.

C# Copy

```
public void Redirect (string url, bool endResponse);
```

Parameters

`url` String

The location of the target.

`endResponse` Boolean

Indicates whether execution of the current page should terminate.

AUTHORIZATION FUNCTIONS - INCORRECT EXAMPLE

application_controller.rb

```
1  # frozen_string_literal: true
2  class ApplicationController < ActionController::Base
3    before_action :authenticated, :has_info, :create_analytic, :mailer_optic
4    helper_method :current_user, :is_admin?, :sanitize_font
5
6    # Our security guy keep talking about sea-surfing, cool story bro.
7    # Prevent CSRF attacks by raising an exception.
8    # For APIs, you may want to use :null_session instead.
9    protect_from_forgery #with: :exception
10
```

Authorization Functions Checklist

How is user access controlled?

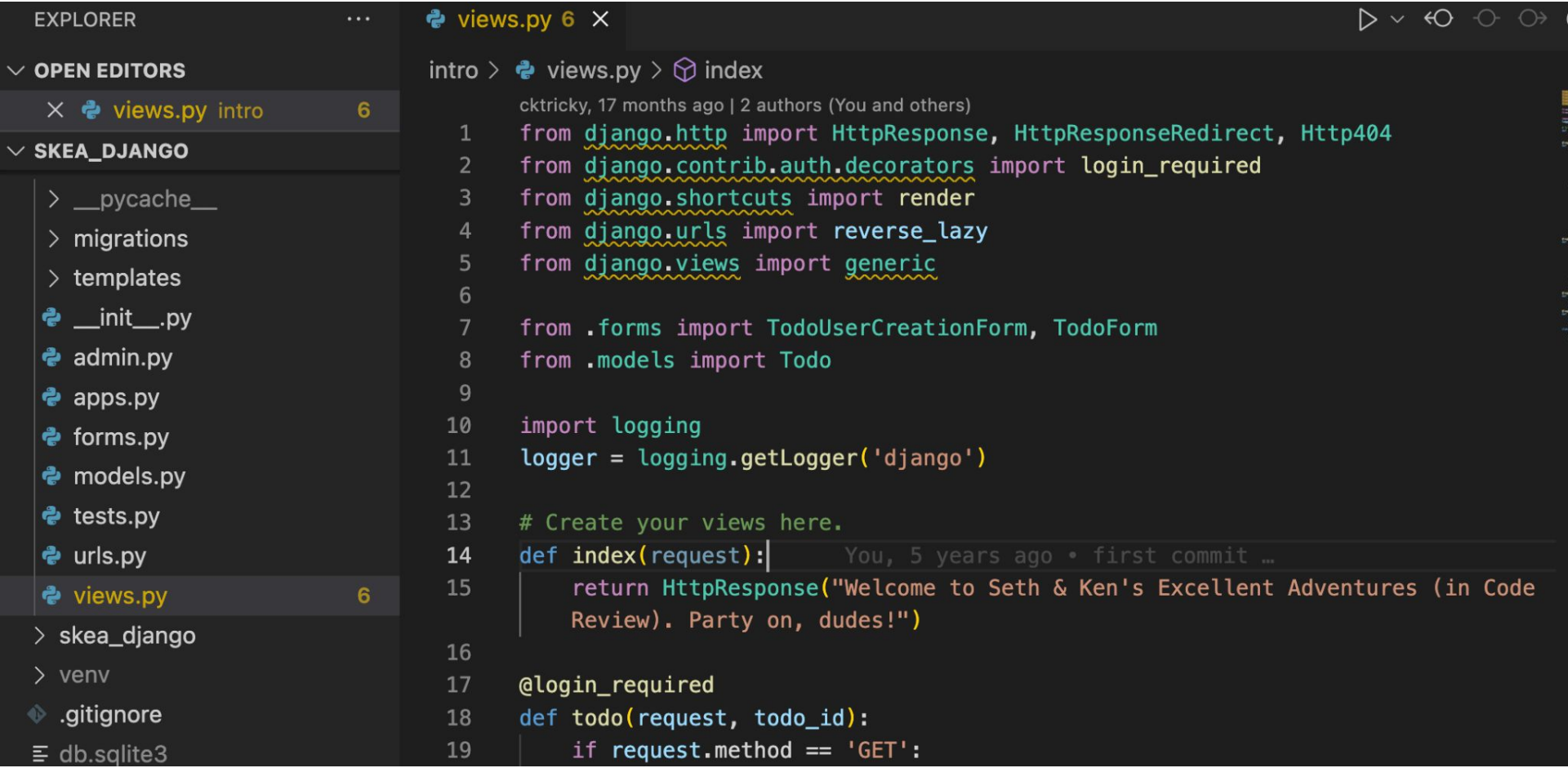
- Ask if any of the following is used:
  - Sessions
  - API Tokens
  - Basic Authentication
  - JWTs
  - Something… else?

AUTHORIZATION FUNCTIONS CHECKLIST

- Identify its purpose
  - Is it checking that we're authenticated?
    - AuthN
  - Is it looking for a specific role?
    - RBAC
  - Is it doing some sort of HMAC verification for events?
    - CSRF

AUTHORIZATION FUNCTIONS CHECKLIST

- ❏ What could go wrong?
  - Insecure timing comparisons
  - Framework nuances
  - IDOR (Insecure Direct Object reference)
  - Lack of rate limiting (see this often with basic auth or tokens)
  - Think about if it was *not* applied, what would happen
  - **Get creative, you're trying to decide what the risk, if any, is here**


EXPLORER
views.py 6
OPEN EDITORS
views.py intro 6
SKEA_DJANGO
__pycache__
migrations
templates
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py 6
skea_django
venv
.gitignore
db.sqlite3
intro > views.py > index
cktricky, 17 months ago | 2 authors (You and others)
from django.http import HttpResponse, HttpResponseRedirect, Http404
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.urls import reverse_lazy
from django.views import generic
from .forms import TodoUserCreationForm, TodoForm
from .models import Todo
import logging
logger = logging.getLogger('django')
# Create your views here.
def index(request):
You, 5 years ago • first commit …
return HttpResponse("Welcome to Seth & Ken's Excellent Adventures (in Code Review). Party on, dudes!")
@login_required
def todo(request, todo_id):
if request.method == 'GET':

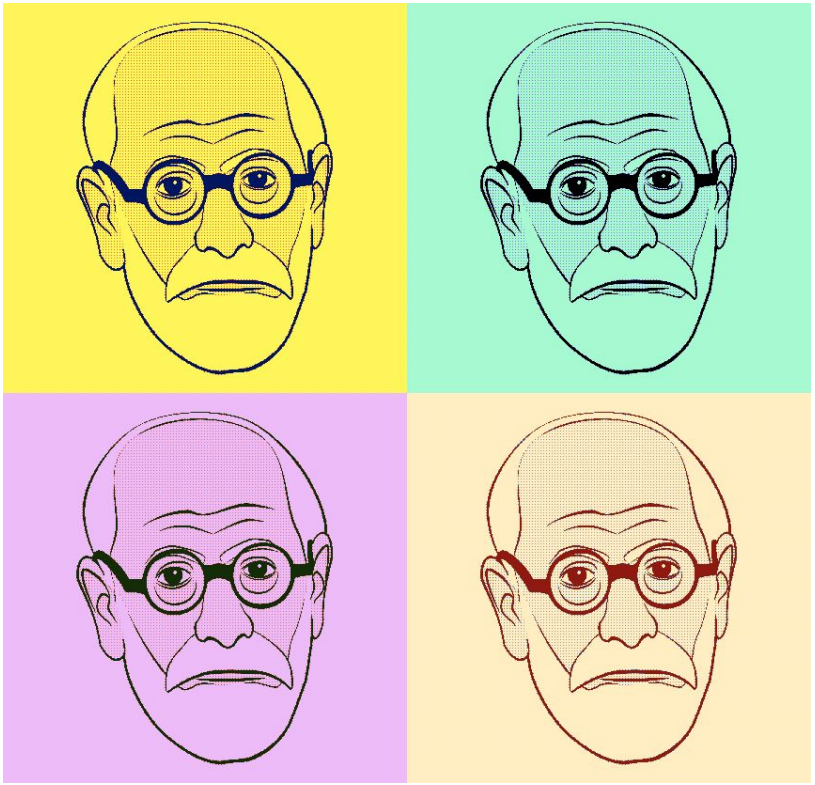
AUTHORIZATION FUNCTIONS EXAMPLE

AUTHZ FUNCTIONS EXERCISE

SIGMUND FREUD

1. **bridge_troll**
   a. Perform Info Gathering
      i. Tech/Purpose/Risk
   b. Map Routes
   c. Identify AuthZ Functions

2. How are users authorized in bridge_troll?
   a. Could take some time



SIGMUND FREUD - POST-MORTEM

Yeah, you probably have questions re: **`authenticate_user!`**

a. Which is why its part of the exercise :=D. THIS HAPPENS A LOT!

```
1  # frozen_string_literal: true
2
3  class SectionsController < ApplicationController
4    before_action :authenticate_user!
5    before_action :find_event
6
```

How to decipher where this action is defined will be defined by:

b. Access to an interactive runtime env?
c. Experience with the framework/libs?
d. No previous experience with it, only access to source?

Authorization Functions - Post-Mortem

"Framework Nuance"

```
post    "/articles", to: "articles#create"
get     "/articles/new", to: "articles#new", as: :new_article
get     "/articles/:id/:action", to: "articles#edit", as: :edit_article
get     "/articles/:id", to: "articles#show", as: :article
post    "/articles/:id/vote/:type", to: "articles#vote", as: :vote
```

AUTHORIZATION FUNCTIONS - POST-MORTEM

Anyone know what the request attack string would look like?


```
get     "/articles/:id/:action", to: "articles#edit", as: :edit_article
```

AUTHZ FUNCTIONS - POST-MORTEM

So what's the point of this whole thing?

- Do your homework on framework nuances
- Good example of why we double-back and check one last time… (story time)

PULL REQUEST REVIEWS

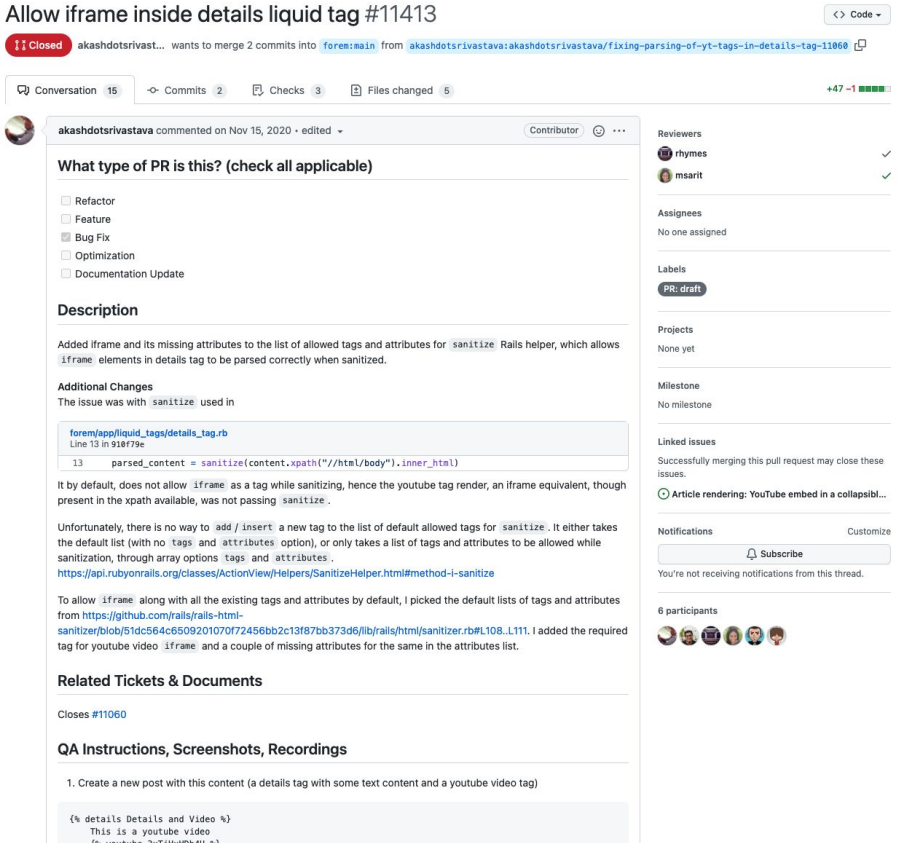
PULL REQUEST REVIEWS - OVERVIEW

Not an end-to-end code review

Snippets are often accompanied by fuzzy context and understanding context is **critical**

Common components:

- Tests (and there Needs to be tests)
- Conversation (context)
- Code



PULL REQUEST REVIEWS - OVERVIEW

General steps:

- Get to know the app
- Get to know the purpose of the pull request
- Read through other's comments
- Use git blame if necessary
- **Ask questions!!!**
- Seek data flow diagrams or something similar
- Common flaws associated with the portion of code you are reviewing (eg: controller == authz issues)

CHECKLISTS & REVIEWS

AUTHORIZATION

AUTHORIZATION REVIEW

- Analyze source for role enforcement, appropriate user boundaries, privileges required for access, and business-logic flaws
- Roles and associated enforcement routines must be identified during information gathering
- Pay attention to any endpoints that include sensitive data or functionality
  - Vertical authorization weaknesses - escalated privileges
    - Authenticated and unauthenticated access
  - Horizontal authorization weaknesses - access another user’s data

Authorization Review Vulnerabilities

- Broken Access Control - OWASP Top 10 A01:2021
  - Privilege Escalation
  - Missing Function Level Access Control
  - Insecure Direct Object Reference
- Sensitive Data Exposure - OWASP Top 10 A3:2017
- Mass Assignment
- Business Logic Flaws

```java
/** Check if userUid has access to a specific application at Organization group Level */
 @ReadOnlyTransaction
 public boolean isApplicationAllowedByUser(
     final Organization org,
     final Application app,
     final String userUid,
     final boolean includeArchived) {
   if (app.isArchived() && !includeArchived) {
     return false;
   }

   // check if application exists
   final EacUser eacUser =
       eacUserAccessControlService.getAccessControlForUser(org.getId(), userUid);
   if (eacUser.isViewAllApplications()) {
     return true;
   }

   return eacUser.getAllowedApplications().contains(app.getId());
 }
```

SPOT THE BUG

Authorization Review Checklist

- What are the different authorization functions?
  - Token/User/Role validation?
  - Is this handled by custom framework functionality?
  - Decorators vs. static source code
- Can non-privileged users view, add, or alter accounts?
- Is there functionality to add accounts with higher access levels than their own access?
- How is separation of duties handled?
- Are disabled accounts prevented from accessing content?
- Can password-protected pages be directly accessed without authentication?

Mass-Assignment

- https://cheatsheetseries.owasp.org/cheatsheets/Mass_Assignment_Cheat_Sheet.html
- Goes by a few names
    - Insecure Binder Vulnerability
    - Insecure Object Mapping
    - Mass-Assignment
- Typically happens when a database entry is created or modified and we do so by throwing in *all* user-supplied parameters
- For example...

MASS-ASSIGNMENT - NODE

```
var user = new User(req.body);
user.save();
```

MASS-ASSIGNMENT- RAILS 2/3

```ruby
def create
  user = User.new(params[:user])
  if user.save
    session[:user_id] = user.id
    redirect_to home_dashboard_index_path
  else
    @user = user
    flash[:error] = user.errors.full_messages.to_sentence
    redirect_to :signup
  end
end
```

```ruby
def create
  user = User.new(user_params)
  if user.save
    session[:user_id] = user.id
    redirect_to home_dashboard_index
```

```ruby
def user_params
  params.require(:user).permit!
end
```

AUTHORIZATION REVIEW CHECKLIST

- Is it possible to bypass authorization restrictions by accessing content directly (e.g. can a non-privileged user access the administration pages of an application)?
- Can a user escalate privileges through cookie modification, altering form input values and HTTP headers, or by fuzzing URL-based parameters?
- Are there horizontal escalation/Insecure Direct Object References in the source code?
- Are authentication and authorization flows the first logic executed for each request?
- Are authorization checks granular (page and directory level) or per-application?

AUTHORIZATION REVIEW CHECKLIST

- Are access to sensitive pages and data denied by default?
- Are users forced to re-assert their credentials for requests that have critical side-effect (account changes, password reset, etc)?
- Do authorization checks have clearly defined roles?
- Can authorization be circumvented by parameter or cookie/token manipulation?
- Are CSRF protections in place and appropriate?

AUTHORIZATION CHECKLIST EXAMPLE

AUTHORIZATION EXERCISE

GENGHIS KHAN

1. **bridge_troll**
    a. Build an authorization checklist
    b. Test checklist items

GENGHIS KHAN - POST-MORTEM

Authorization Checklist Items

- Insecure Direct Object Reference
  - Profiles, Unlisted Events
- Privilege Escalation, Mass Assignment
  - User object updates
- CSRF Implementation Checks
- Authorization Patterns for access control
- Sensitive Data Exposure
  - What, where, when, how?

GENGHIS KHAN - POST-MORTEM

app > controllers > chapters_controller.rb

```
12    def show
13      skip_authorization
14      @chapter_events = (
15        @chapter.events.includes(:organizers, :location).published_or_visible_to(current_user) + @chapter.external_ev
16      ).sort_by(&:ends_at)
17      @show_organizers = true
18
19      return unless @chapter.leader?(current_user)
20
21      @organizer_rsvps = Rsvp
22                         .group(:user_id, :user_type)
23                         .joins([{ event: :chapter }])
24                         .includes(:user)
25                         .select('user_id, user_type, count(*) as events_count')
26                         .where('chapters.id' => @chapter.id, role_id: Role::ORGANIZER.id, user_type: 'User')
27    end
28
29    def new
30      authorize Chapter
31      @chapter = Chapter.new
32    end
```

GENGHIS KHAN - POST-MORTEM

bridge_troll is dependent on the `authorize` function for access control protection. Where else do we look now?

app > controllers > application_controller.rb

Michael Glass, 3 years ago | 4 authors (Travis Grathwell and others)

```
1   # frozen_string_literal: true
2
3   class ApplicationController < ActionController::Base
4     protect_from_forgery
5
6     include Pundit
7     rescue_from Pundit::NotAuthorizedError, with: :user_not_authorized
8     after_action :verify_authorized, unless: :devise_controller?
9
10    before_action :configure_permitted_parameters, if: :devise_controller?
11
12    before_action do
13      Rack::MiniProfiler.authorize_request if current_user.try(:admin?)
```

GENGHIS KHAN

Pundit

- Uses Policies
- So what pattern are we interested in?

app > policies > course_policy.rb

```ruby
class CoursePolicy < ApplicationPolicy
  def index?
    user.admin?
  end

  def destroy?
    record.events.empty? && user.admin?
  end

  def edit?
    update?
  end

  def new?
    create?
  end

  def update?
    user.admin?
  end
```

GENGHIS KHAN - POST-MORTEM

app > controllers > users_controller.rb

Michael Glass, 2 years ago | 2 authors (Travis Grathwell and others)

```ruby
# frozen_string_literal: true

class UsersController < ApplicationController
  before_action :authenticate_user!

  def index
    skip_authorization
    respond_to do |format|
      format.html
      format.json do
        render json: UserList.new(params)
      end
    end
  end
end
```

AUTHENTICATION

AUTHENTICATION REVIEW

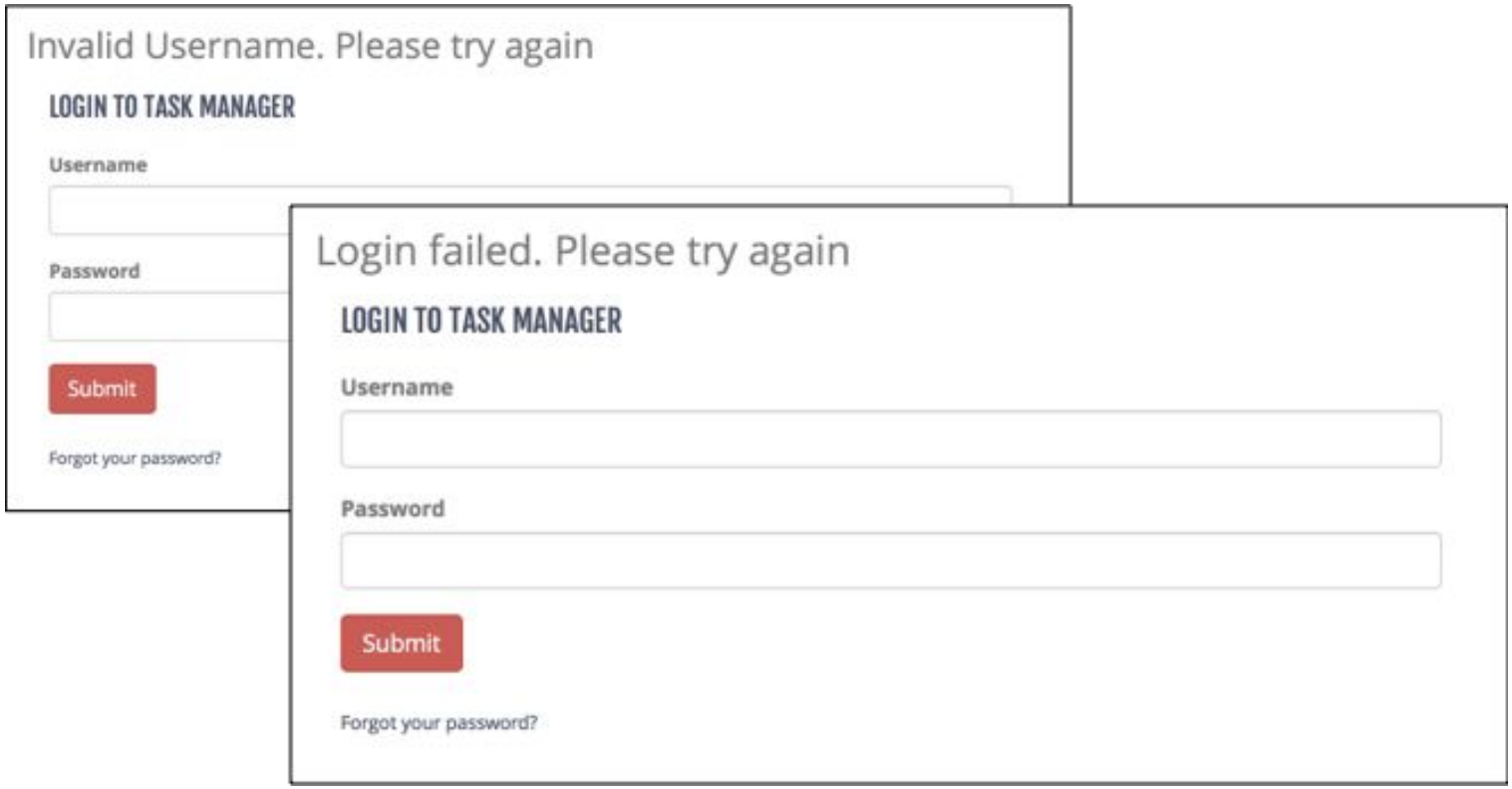
- Authentication establishes user identity
- Examine the user identification process of the application.
- Available application resources include both unidentified and identified users
- Use enumeration of the application endpoints to trace the authentication flow and functions.
- Sensitive application and business functionality should redirect as appropriate to the authentication flow to properly identify a user
- Include an application functionality that identifies a user in this review

Authentication Review

How does an application confirm identity?

AUTHENTICATION REVIEW VULNERABILITIES

- Identification and Authentication Failures - **OWASP Top 10 A07:2021**
- User Enumeration
- Session Management Issues
- Authentication Bypass
- Brute-Force Attacks


Invalid Username. Please try again
LOGIN TO TASK MANAGER
Username
Password
Submit
Forgot your password?
Login failed. Please try again
LOGIN TO TASK MANAGER
Username
Password
Submit
Forgot your password?

SPOT THE BUG

```
exports.update = function(req, res) {

    if (req.body.new_password = req.body.new_password_confirmation){
        username = req.body.username
        current_password = req.body.current_password
        isMatch = true

        db.User.find({where: {username: username}}).success(function (user){
            hash = user ? user.password : ''
            isMatch = db.User.validPassword(current_password, hash, function () { console.lo
        });
        if (isMatch) {
            req.user.username = username
            req.user.password = req.body.new_password
            req.user.save()
        } else {
            console.log('Bad Password')
        }
    }

    res.redirect('/account')
};
```

SPOT THE BUG

Authentication Review Checklist

- Identify all the authentication flows
  - User Login
  - User Registration
  - Forgot Password
- How are users identified? What information do they have to provide?
  - Username, email, password, MFA token, etc.
- How is authentication handled on each application endpoint?
  - Sensitive endpoints should require authentication
- Are there any hard-coded accounts?
- Does application allow for easily-guessed, default, or common passwords?

Authentication Review Checklist

- How are usernames determined, what naming conventions?
  - Does registration allow for easy enumeration?
- Does the application allow for account enumeration through server responses or information disclosure?
  - login/registration/forgot password flows
- Are user credentials protected in the data store using modern password hashing algorithms?
- Are security policies are configurable via environment variables and not hard-coded?
- What standard security frameworks are used?
  - Is there code specific to the application, especially when dealing with password storage?

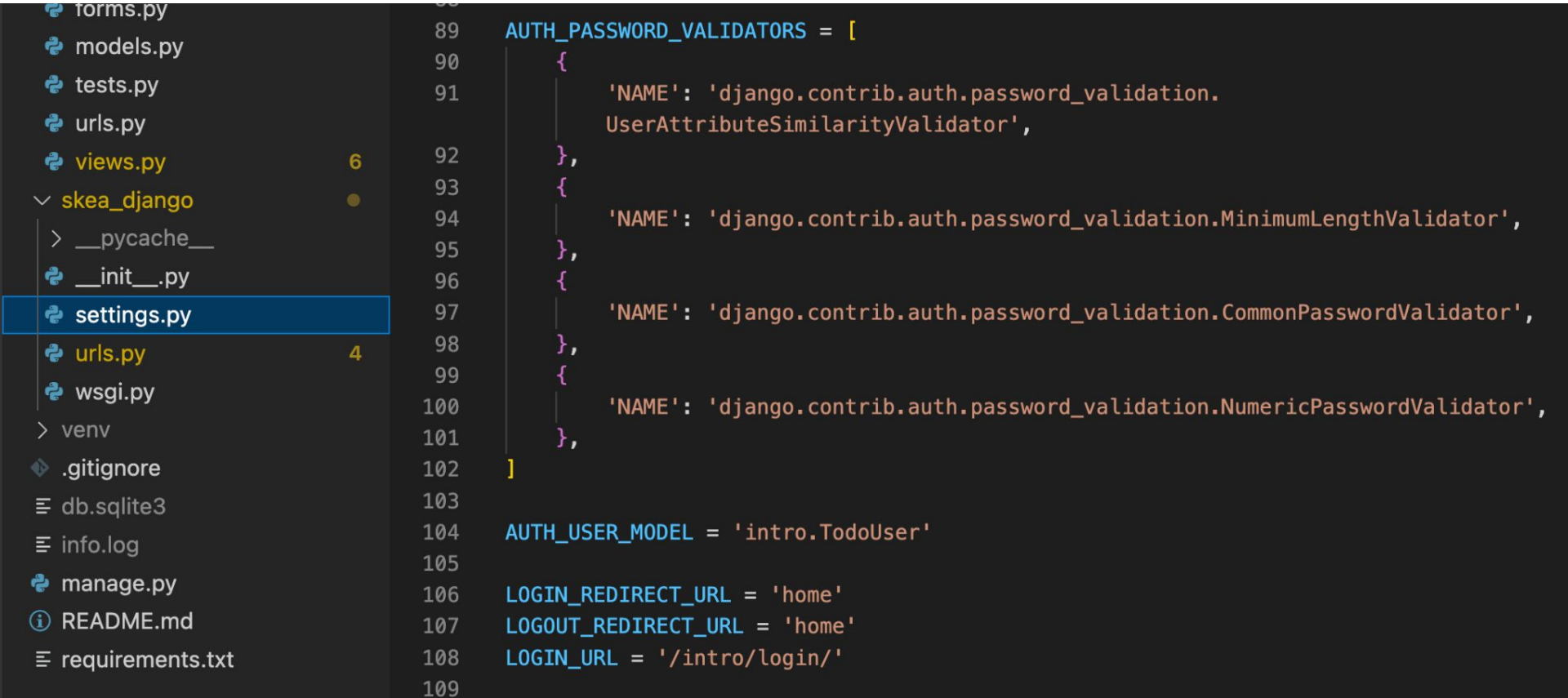
Authentication Review Checklist

- How are user management events such as authentication failures, password resets, password changes, account lockout and disabled accounts handled?
- Are suspicious event handling such as multiple failed login attempts, session replay and attempted access to restricted resources handled properly?
- Does the application implement strong password policies?
- Are authentication credentials being passed using technologies that could be cached (HTTP GET, lack of proper cache-control settings)?
- If applicable, are encryption mechanisms in place during authentication for secure communications (TLS, etc)?

Authentication Review Checklist (Registration)

- Are limits on application access, such as geographical boundaries, enforced properly?
- Is there a manual approval process or is access granted automatically?
  - Can the automated process be abused or bypassed using scripting or brute-forcing?
- In cases where a validation email and link are required, how are the tokens generated?
- Can users elevate their initial access via mass assignment or business-logic bypasses?
- Are files and objects owned by a user removed or archived?

AUTHENTICATION CHECKLIST EXAMPLE



Authentication Review Checklist (Sessions)

- Are encryption and hashing used properly?
- Do encryption protocols use strong algorithms and industry-standard key lengths?
- Are authentication tokens set with time limits?
- Are cookies security parameters set properly (e.g. Secure, HTTPOnly, path)?
- Are session IDs sent over a secure channel?
- Are session IDs invalidated before a new login is made?
- Are CSRF tokens set for all authentication requests?

AUTHENTICATION EXERCISE

SOCRATES

1. **bridge_troll**
    a. Build Authentication Checklist
        i. Abstract Identification
        ii. How and where in this application?
    b. Review Authentication items:
        i. Login
        ii. Registration
        iii. Sessions

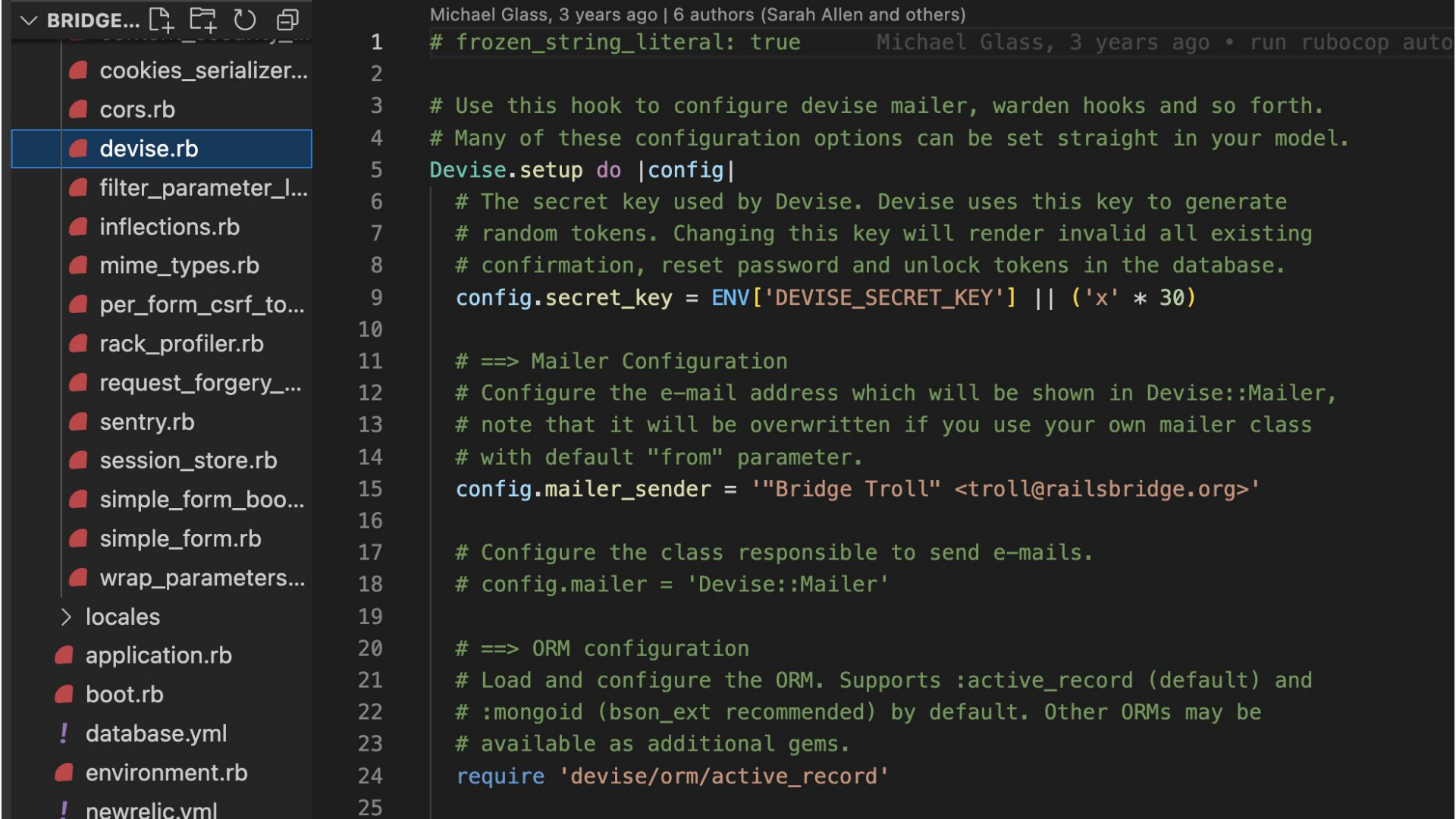
SOCRATES - POST-MORTEM

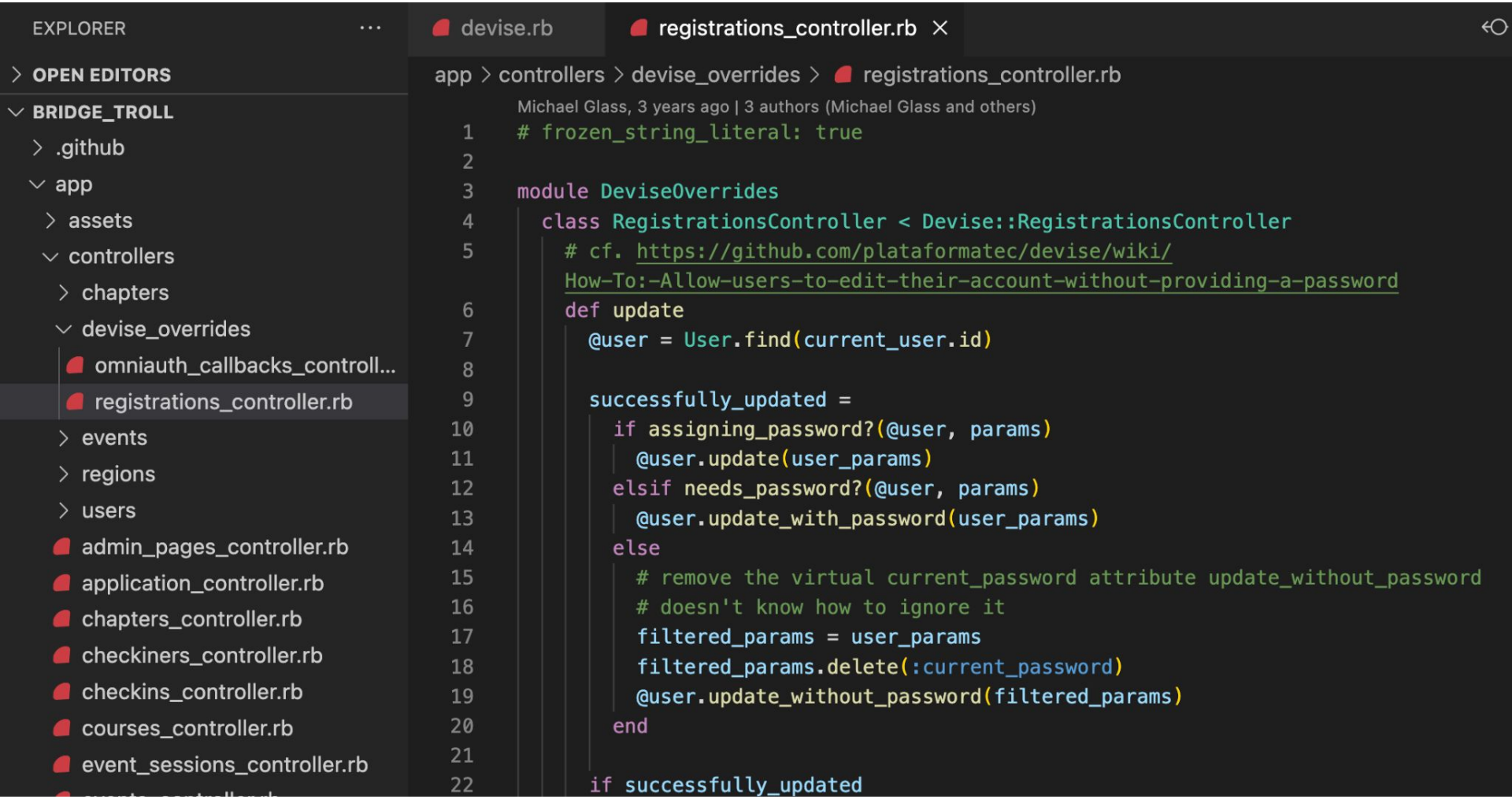
Authentication Items

- Authentication flows/user enumeration
  - Login, Forgot Password, Registration
- Session Handling
  - Session hijacking, bypassing authentication
- Privilege Assignment
  - Registration process

SOCRATES - POST-MORTEM

```
seth@MacBook-Pro-3 bridge_troll % rake routes
                                Prefix Verb     URI Pattern
 Controller#Action
                                  root GET      /
 events#index
                      new_user_session GET      /users/sign_in(.:format)
 devise/sessions#new
                          user_session POST     /users/sign_in(.:format)
 devise/sessions#create
                  destroy_user_session DELETE   /users/sign_out(.:format)
 devise/sessions#destroy
     user_facebook_omniauth_authorize GET|POST /users/auth/facebook(.:format)
 devise_overrides/omniauth_callbacks#passthru
      user_facebook_omniauth_callback GET|POST /users/auth/facebook/callback(.:format)
 devise_overrides/omniauth_callbacks#facebook
      user_twitter_omniauth_authorize GET|POST /users/auth/twitter(.:format)
 devise_overrides/omniauth_callbacks#passthru
       user_twitter_omniauth_callback GET|POST /users/auth/twitter/callback(.:format)
 devise_overrides/omniauth_callbacks#twitter
       user_github_omniauth_authorize GET|POST /users/auth/github(.:format)
 devise_overrides/omniauth_callbacks#passthru
        user_github_omniauth_callback GET|POST /users/auth/github/callback(.:format)
 devise_overrides/omniauth_callbacks#github
       user_meetup_omniauth_authorize GET|POST /users/auth/meetup(.:format)
 devise_overrides/omniauth_callbacks#passthru
```


BRIDGE...
cookies_serializer...
cors.rb
devise.rb
filter_parameter_l...
inflections.rb
mime_types.rb
per_form_csrf_to...
rack_profiler.rb
request_forgery_...
sentry.rb
session_store.rb
simple_form_boo...
simple_form.rb
wrap_parameters...
locales
application.rb
boot.rb
database.yml
environment.rb
newrelic.yml
Michael Glass, 3 years ago | 6 authors (Sarah Allen and others)
frozen_string_literal: true Michael Glass, 3 years ago • run rubocop auto
Use this hook to configure devise mailer, warden hooks and so forth.
Many of these configuration options can be set straight in your model.
Devise.setup do |config|
# The secret key used by Devise. Devise uses this key to generate
# random tokens. Changing this key will render invalid all existing
# confirmation, reset password and unlock tokens in the database.
config.secret_key = ENV['DEVISE_SECRET_KEY'] || ('x' * 30)
# ==> Mailer Configuration
# Configure the e-mail address which will be shown in Devise::Mailer,
# note that it will be overwritten if you use your own mailer class
# with default "from" parameter.
config.mailer_sender = '"Bridge Troll" <troll@railsbridge.org>'
# Configure the class responsible to send e-mails.
# config.mailer = 'Devise::Mailer'
# ==> ORM configuration
# Load and configure the ORM. Supports :active_record (default) and
# :mongoid (bson_ext recommended) by default. Other ORMs may be
# available as additional gems.
require 'devise/orm/active_record'


EXPLORER
OPEN EDITORS
BRIDGE_TROLL
.github
app
assets
controllers
chapters
devise_overrides
omniauth_callbacks_controll...
registrations_controller.rb
events
regions
users
admin_pages_controller.rb
application_controller.rb
chapters_controller.rb
checkiners_controller.rb
checkins_controller.rb
courses_controller.rb
event_sessions_controller.rb
devise.rb
registrations_controller.rb
app > controllers > devise_overrides > registrations_controller.rb
Michael Glass, 3 years ago | 3 authors (Michael Glass and others)
frozen_string_literal: true
module DeviseOverrides
class RegistrationsController < Devise::RegistrationsController
cf. https://github.com/plataformatec/devise/wiki/How-To:-Allow-users-to-edit-their-account-without-providing-a-password
def update
@user = User.find(current_user.id)
successfully_updated =
if assigning_password?(@user, params)
@user.update(user_params)
elsif needs_password?(@user, params)
@user.update_with_password(user_params)
else
# remove the virtual current_password attribute update_without_password
# doesn't know how to ignore it
filtered_params = user_params
filtered_params.delete(:current_password)
@user.update_without_password(filtered_params)
end
if successfully_updated

AUDITING

AUDITING REVIEW

- Validate that appropriate logging and exception handling are handled within application source
- One path in the trace of sensitive data from source to sink
- Logging functions and error messages are considered a data sink
- Logging should happen in any endpoint that performs a state-changing operation or has security implications
- This data is used for immediate analysis and future forensics needs.
- Check that sensitive data is appropriately handled (no credit card numbers, etc) and the correct details are logged
- Administrators must trust that logs may not be manipulated by unauthorized parties

AUDITING REVIEW VULNERABILITIES

- Cryptographic Failures - OWASP Top 10 A02:2021
- Security Logging & Monitoring Failures - OWASP Top 10 A09:2021
- Debug Messages
- Error Handling
- Information Leakage

```
  superadminService.manageOrganizationByUserUid(userUid, org);
  ImpersonateHelper.configureImpersonateWebSettings(
      request, response, ImpersonateReturnMode.USERS_MODE);
  apiResponse.setOrgUuid(org.getUuid());
  apiResponse.registerSuccess(String.format("Switching to user '%s' successfully", userUid));

  return new ResponseEntity<>(apiResponse, HttpStatus.OK);
```

SPOT THE BUG

AUDITING REVIEW CHECKLIST

- If an exception occurs, does the application fails securely?
- Do error messages reveal sensitive application or unnecessary execution details?
- Are Component, framework, and system errors displayed to end user?
- Does exception handling that occurs during security sensitive processes release resources safely and roll back any transactions?
- Are relevant user details and system actions logged?
- Is sensitive user input flagged, identified, protected, and not written to the logs?
    - Credit Card #s, Social Security Numbers, Passwords, PII, keys

AUDITING REVIEW CHECKLIST

- Are unexpected errors and inputs logged?
  - Multiple login attempts, invalid logins, unauthorized access attempts
- Are log details specific enough to reconstruct events for audit purposes?
- Are logging configuration settings configurable through settings or environment variables and not hard-coded into the source?
- Is User-controlled data validated and/or sanitized before logging to prevent log injection?

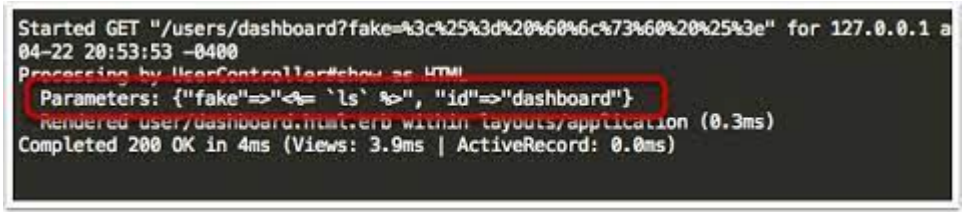
STORY TIME:

LOG INJECTION + FRAMEWORK VULNERABILITY

INJECTION + FRAMEWORK VULNERABILITY = RCE

Odd place to look for erb (html + ruby code) files

Request with ERB-style ruby code that would execute the “ls” command if executed by an ERB file



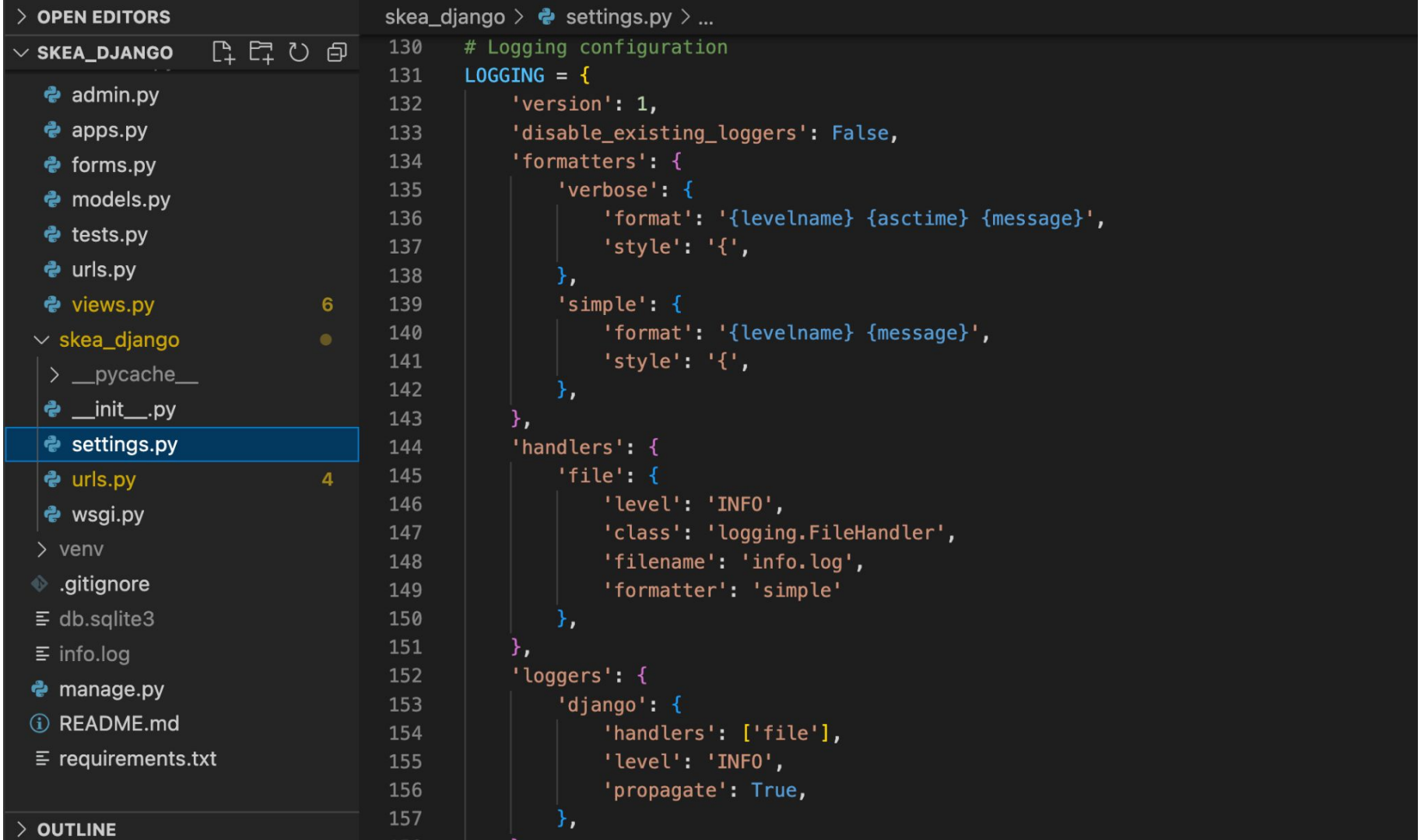
The final step was to have the render file pull in the log file which contains the valid ERB code and is executed as such…

LOGGING - JAVA

```
    return new ResponseEntity<>(apiResponse, HttpStatus.OK);
  } catch (final ServiceException e) {
    log.error("ERROR: " + e.getMessage());
    ErrorHelper.addErrorMessage(
        String.format("Failed to impersonate user '%s'", userUid), apiResponse, e);
  }
```

LOGGING EXAMPLE

```
47 @login_required
48 def create_todo(request):
49     if request.method == 'POST':
50         form = TodoForm(request.POST)
51         if form.is_valid():
52             t = Todo( todo_text=form.cleaned_data['todo_text'],
53                       todo_date = form.cleaned_data['todo_date'],
54                       completed = form.cleaned_data['completed'])
55             t.owner = request.user
56             t.save()
57             logger.info("Created todo %s by %s" % (todo_id,request.user.username))
58             return HttpResponseRedirect('/intro/todos/')
59
60     else:
61         form = TodoForm()
```


OPEN EDITORS
SKEA_DJANGO
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py 6
skea_django
__pycache__
__init__.py
settings.py
urls.py 4
wsgi.py
venv
.gitignore
db.sqlite3
info.log
manage.py
README.md
requirements.txt
OUTLINE
skea_django > settings.py > ...
130 # Logging configuration
131 LOGGING = {
132 'version': 1,
133 'disable_existing_loggers': False,
134 'formatters': {
135 'verbose': {
136 'format': '{levelname} {asctime} {message}',
137 'style': '{',
138 },
139 'simple': {
140 'format': '{levelname} {message}',
141 'style': '{',
142 },
143 },
144 'handlers': {
145 'file': {
146 'level': 'INFO',
147 'class': 'logging.FileHandler',
148 'filename': 'info.log',
149 'formatter': 'simple'
150 },
151 },
152 'loggers': {
153 'django': {
154 'handlers': ['file'],
155 'level': 'INFO',
156 'propagate': True,
157 },

AUDITING EXERCISE

ABRAHAM LINCOLN

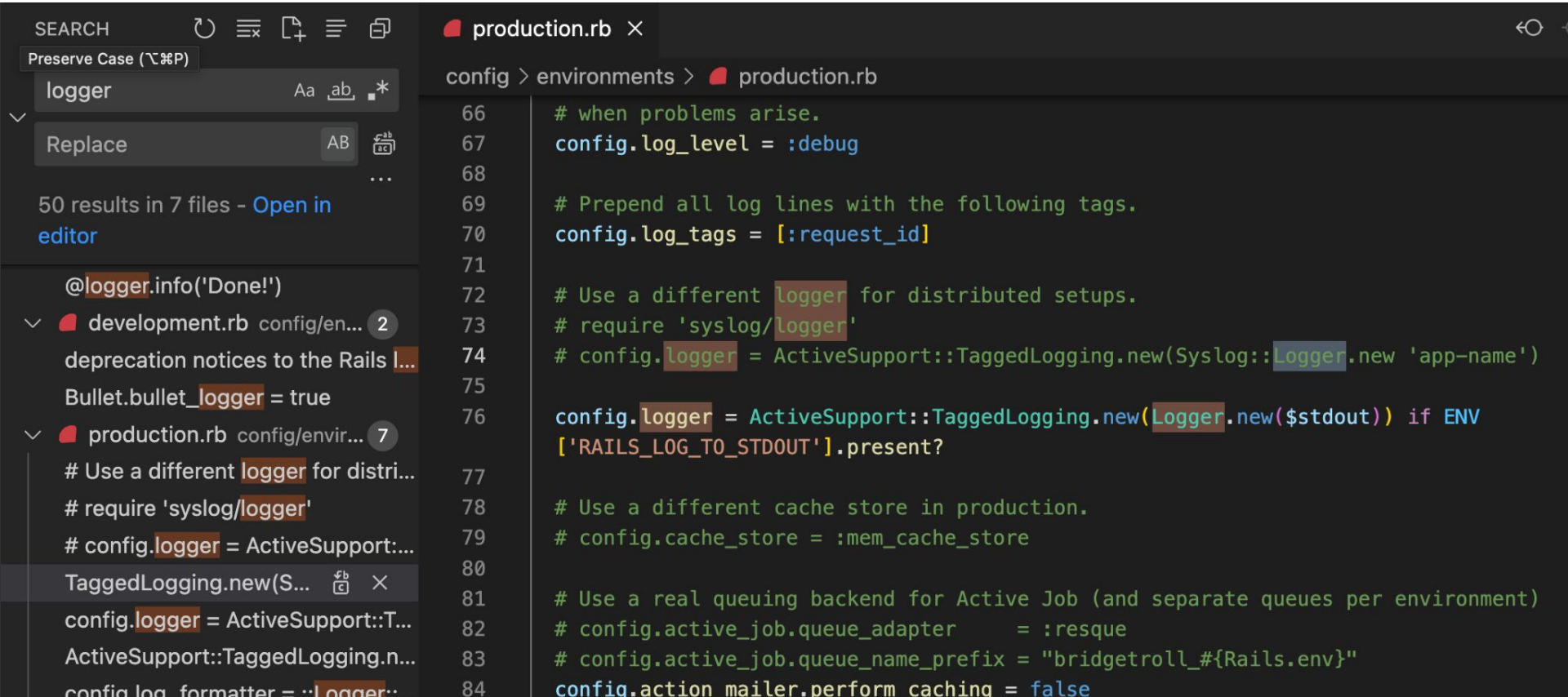
1. **bridge_troll**
    a. Build/Review Auditing Checklist
    b. Review at least 5 checklist items

Abraham Lincoln - Post-Mortem

Audit Items

- Security Event Logging
  - Devise, Pundit
- Debug/Error Messages

ABRAHAM LINCOLN - POST-MORTEM



Search: logger

logger

50 results - 7 files

.rspec_parallel:
```
1  --format progress
2: --format ParallelTests::RSpec::RuntimeLogger --out tmp/parallel_runtime_rspec.log
```

app/services/account_merger.rb:
```
15
16:     Rails.logger.info(<<-USER_PROMPT.strip_heredoc)
17
```

app/services/database_anonymizer.rb:
```
 4  class DatabaseAnonymizer
 5:   def initialize(logger = nil)
 6      require 'faker'
 7
 8:     @logger = logger || Logger.new($stdout)
 9:     @logger.formatter = proc do |_severity, _datetime, _progname, msg|
10       "#{msg}\n"

14    def anonymize_database
15:     @logger.info('Anonymizing users...')
16      User.find_each { |u| anonymize_user(u) }
17
```

INJECTION

INJECTION

- Causes:
  - Input Validation
  - Output Encoding
- Types
  - SQL Injection
  - HTML Injection (XSS)
  - LDAP, XML, Command ...

INJECTION VULNERABILITIES

- Injection - OWASP Top 10 A03:2021
- XML External Entities (XXE)
- Cross-Site Scripting (XSS)
- Redirects
- Server Side Request Forgery (SSRF) - OWASP Top 10 A10:2021

```
if (FormsAuthenticationHelper.IsRelativeUrl(redirectUrl))
{
        this.Response.Redirect(redirectUrl, true);
}
else
```

```
/// <summary>
/// Tests the incomming url to see if it relative.
/// </summary>
/// <param name="url"></param>
/// <returns></returns>
public static bool IsRelativeUrl(string url)
{
        Uri result;
        return Uri.TryCreate(url, UriKind.Relative, out result);
}
```

SPOT THE BUG

INPUT VALIDATION

- Analyze code that handles user input for type, format, and content validation before being used or stored by the application.
- Compile list of data sources to work through.
- Start with the routes identified in the information gathering phase, but also include:
  - Configuration files
  - Environment variables
  - External services
  - Database calls to external and internal databases.
  - ...

INPUT VALIDATION - CHECKLIST

- Is all input is validated without exception?
- Do the validation routines check for known good characters and cast to the proper data type (integer, date, etc.)?
- Is the user data validated on the client or server or both (security should not rely solely on client-side validations that may be bypassed)?

INPUT VALIDATION - CHECKLIST

- If both client-side and server-side data validation is taking place, are these validations consistent and synchronized?
- Do string input validation use regular expressions?
- Do these regular expressions use DenyLists or AcceptLists?
- What bypasses exist within the regular expressions?

INPUT VALIDATION - CHECKLIST

- Does the application validate numeric input by type and reject unexpected input?
- How does the application evaluate and process input length?
- Is a strong separation enforced between data and commands (filtering out injection attacks)?

INPUT VALIDATION - CHECKLIST

- Is there separation between data and client-side scripts?
- Is provided data checked for special characters before being passed to SQL, LDAP, XML, OS and third party services?
- For web applications, are often forgotten HTTP request components, including HTTP headers (e.g. referrer) validated?

SQL INJECTION - DJANGO

```
@csrf_exempt
def forgot_password(request):

    if request.method == 'POST':
        t_email = request.POST.get('email')

        try:
            result = User.objects.raw("SELECT * FROM auth_user where email = '%s'" % t_email)

            if len(list(result)) > 0:
                result_user = result[0]
                # Generate secure random 6 digit number
                res = ""
                nums = [x for x in os.urandom(6)]
```

SQL INJECTION - NODE.JS

```
10
11 exports.search = function(req,res) {
12     q = "";
13     users = [];
14     if (req.query.q) {
15         q = req.query.q;
16         db.User.findAll({attributes: ['id', 'username'], where: {username: { like: '%'
           +req.query.q+'%' } }}).success(function(users){
17             //console.log('Users:', users)
18             res.render("search.ejs", { q: q, username: req.user.username, users: users
                });
19         });
20     } else {
21         db.User.findAll().then(function(users){
22             //console.log('Users:', users)
23             res.render("search.ejs", { q: q, username: req.user.username, users: users
                });
24         });
```

OUTPUT ENCODING

- Analyze code that sends user data to client for context, type, and format before sending to uncontrolled data sinks.
- Start with a list of data sinks where data is being stored, sent, processed.
  - Source code libraries
  - 3rd-party services
  - Storage components (database in any of its possible forms)
  - File system interactions
  - Log files
- XSS, SSRF

OUTPUT ENCODING - CHECKLIST

- Do databases interactions use parameterized queries?
- Do input validation functions properly encode or sanitize data for the output context?
- How do framework-provided database ORM functions used?
- Does the source code use potentially-dangerous ORM functions? (.raw, etc)

OUTPUT ENCODING - CHECKLIST

- What output encoding libraries are used?
- Are output encoding libraries up-to-date and patched?
- Is proper output encoding used for the context of each output location?
- Are output encoding routines dependent on regular expressions? Are there any weaknesses or blind-spots in these expressions?

XSS - NODE.JS (EJS TEMPLATES)

```
106     <tbody>
107
108       <% for(var i=0; i < listings.length; i++) { %>
109       <tr>
110         <td><%- listings[i].created %></td>
111         <td><%- listings[i].name %></td>
112         <td><%- listings[i].description %></td>
113         <td><%- listings[i].deadline %>
114
115         </td>
116         <td><a class="icon-ok" href="javascript:alert('Apply for Position')"></a><a c
117       </tr>
118       <% } %>
119     </tbody>
120   </table>
121   </div>
```

XSS - DJANGO

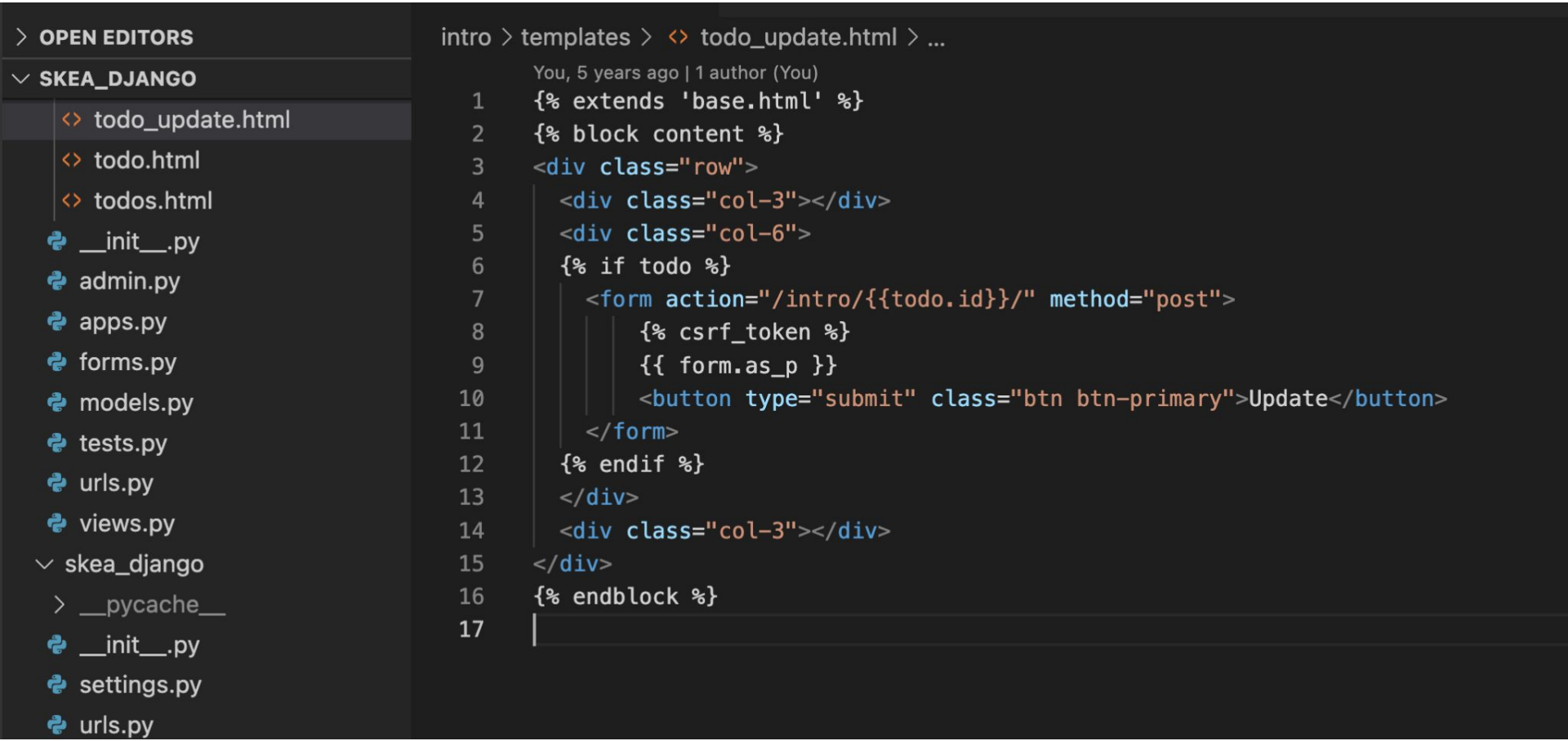
```
<!-- user login dropdown start-->
<li class="dropdown">
    <a data-toggle="dropdown" class="dropdown-toggle" href="#">
        <!--User Identity -->
        <span class="username"><i class="fa fa-user fa-fw"></i> {{ user.username|safe }}</span>
        <b class="caret"></b>
    </a>
    <ul class="dropdown-menu extended logout">
        {% if user.id %}
```

INJECTION CHECKLIST EXAMPLE

views.py 6 | urls.py 4 | settings.py

intro > views.py > index

```
17  @login_required
18  def todo(request, todo_id):
19      if request.method == 'GET':
20          try:
21              todo = Todo.objects.get(pk=todo_id,owner=request.user)
22              form = TodoForm(initial={'todo_text':todo.todo_text,
23                                       'todo_date':todo.todo_date,
24                                       'completed':todo.completed})
25          except Todo.DoesNotExist:
26              raise Http404("Todo does not exist")
27
28          logger.info("GET todo %s by %s" % (todo_id,request.user.username))
29          return render(request, "todo_update.html", {'todo': todo, 'form': form} )
30      elif request.method == 'POST':
31          form = TodoForm(request.POST)
32          if form.is_valid():
33              try:
```


OPEN EDITORS
SKEA_DJANGO
todo_update.html
todo.html
todos.html
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
skea_django
__pycache__
__init__.py
settings.py
urls.py
intro > templates > todo_update.html > ...
You, 5 years ago | 1 author (You)
{% extends 'base.html' %}
{% block content %}

{% if todo %}
<form action="/intro/{{todo.id}}/" method="post">
{% csrf_token %}
{{ form.as_p }}
<button type="submit" class="btn btn-primary">Update</button>
</form>
{% endif %}

{% endblock %}

INJECTION EXERCISE

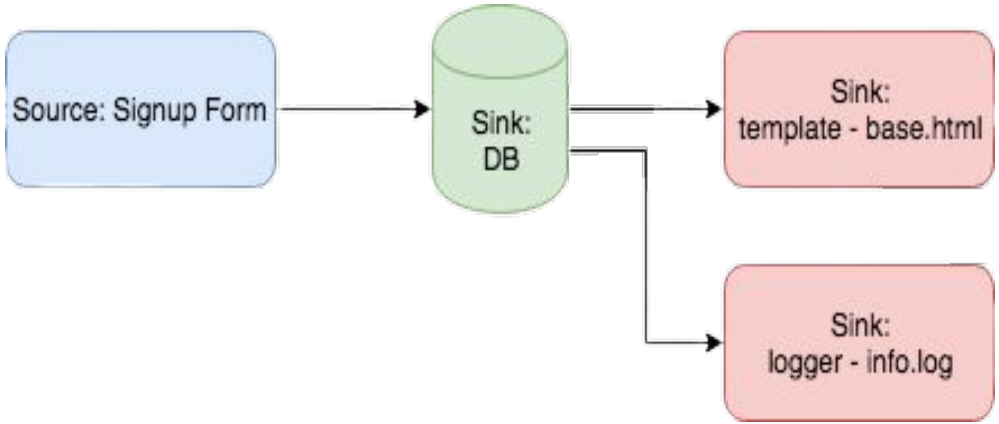
BILLY THE KID

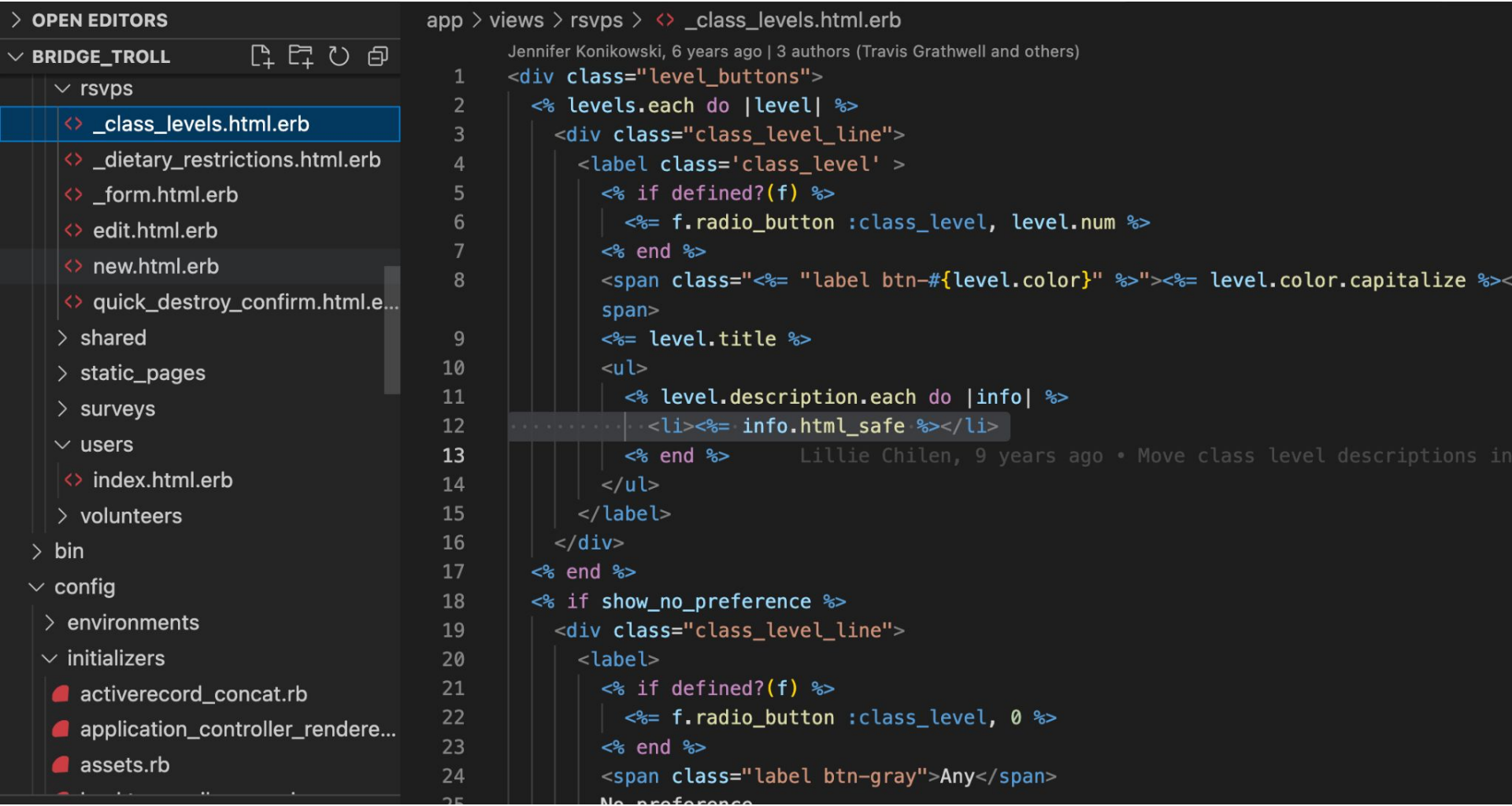
1. **bridge_troll**
   a. Build Injection Checklist
   b. Trace at least 5 sources to sinks

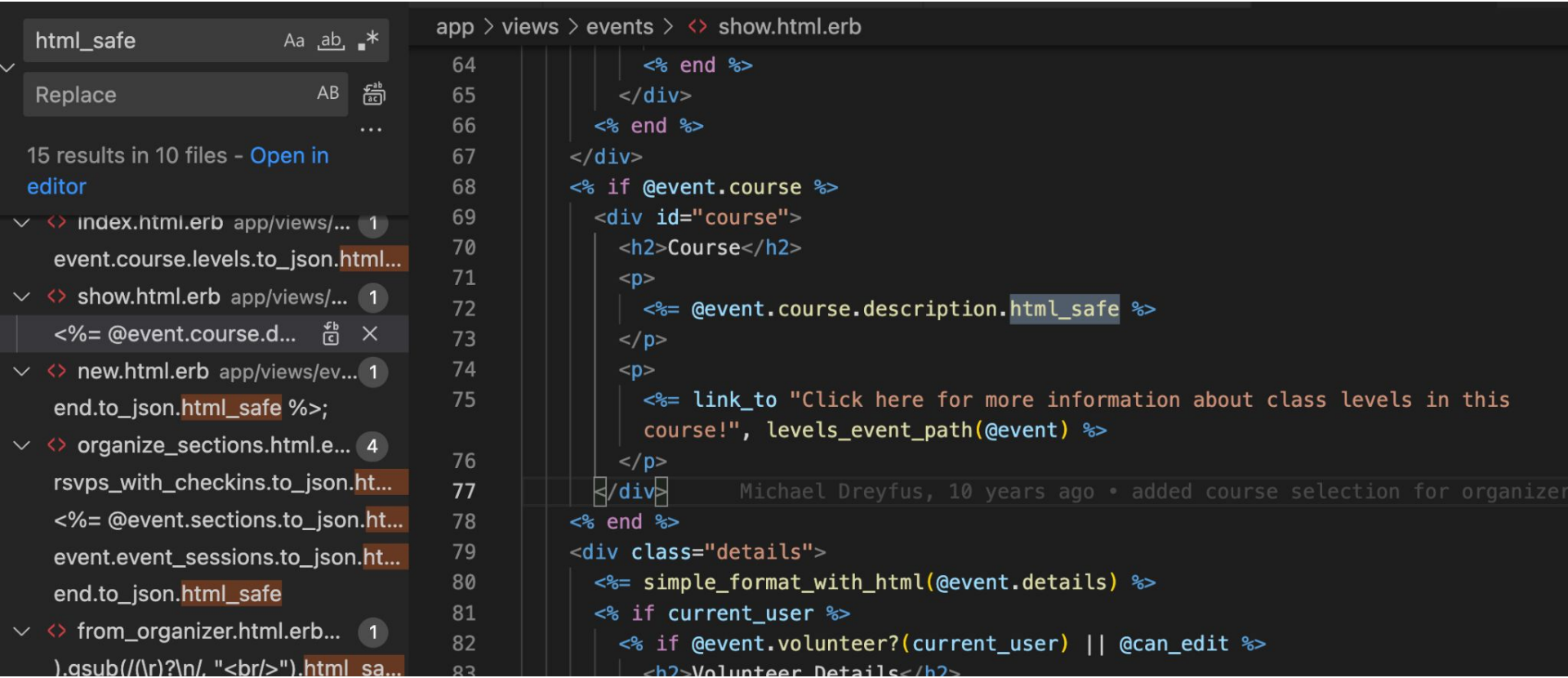
BILLY THE KID - POST-MORTEM

WebApp with a database Backend:

a. SQL Injection (database storage)
b. XSS (Stored/Reflected)
c. Log Forging/Injection




OPEN EDITORS
BRIDGE_TROLL
rsvps
_class_levels.html.erb
_dietary_restrictions.html.erb
_form.html.erb
edit.html.erb
new.html.erb
quick_destroy_confirm.html.e...
shared
static_pages
surveys
users
index.html.erb
volunteers
bin
config
environments
initializers
activerecord_concat.rb
application_controller_rendere...
assets.rb
app > views > rsvps > _class_levels.html.erb
Jennifer Konikowski, 6 years ago | 3 authors (Travis Grathwell and others)
1
2 <% levels.each do |level| %>
3
4 <label class='class_level' >
5 <% if defined?(f) %>
6 <%= f.radio_button :class_level, level.num %>
7 <% end %>
8 <span class="<%= "label btn-#{level.color}" %>"><%= level.color.capitalize %><
span>
9 <%= level.title %>
10
11 <% level.description.each do |info| %>
12 <%= info.html_safe %>
13 <% end %>
Lillie Chilen, 9 years ago • Move class level descriptions in
14
15 </label>
16
17 <% end %>
18 <% if show_no_preference %>
19
20 <label>
21 <% if defined?(f) %>
22 <%= f.radio_button :class_level, 0 %>
23 <% end %>
24 Any


html_safe
Replace
15 results in 10 files - Open in editor
index.html.erb app/views/... 1
event.course.levels.to_json.html...
show.html.erb app/views/... 1
<%= @event.course.d...
new.html.erb app/views/ev... 1
end.to_json.html_safe %>;
organize_sections.html.e... 4
rsvps_with_checkins.to_json.ht...
<%= @event.sections.to_json.ht...
event.event_sessions.to_json.ht...
end.to_json.html_safe
from_organizer.html.erb... 1
app > views > events > show.html.erb
<% end %>

<% end %>

<% if @event.course %>

<h2>Course</h2>

<%= @event.course.description.html_safe %>

<%= link_to "Click here for more information about class levels in this course!", levels_event_path(@event) %>

Michael Dreyfus, 10 years ago • added course selection for organizer
<% end %>

<%= simple_format_with_html(@event.details) %>
<% if current_user %>
<% if @event.volunteer?(current_user) || @can_edit %>

app > controllers > organizations_controller.rb

```ruby

class OrganizationsController < ApplicationController
  before_action :authenticate_user!, only: [:download_subscriptions]

  def index
    skip_authorization
    @organizations = Organization.all
    chapter_last_event_ids = Event
                             .published
                             .select('max(id) as event_id, chapter_id')
                             .group(:chapter_id)
                             .map(&:event_id)
    @chapter_locations = Event
                         .includes(:location, :chapter)
                         .where(id: chapter_last_event_ids)      Michael Glass, 3 years ag
                         .map { |e| ChapterEventLocation.new(e) }
  end

  def show
    skip_authorization
    @organization = Organization.find(params[:id])
  end

```

CRYPTOGRAPHIC ANALYSIS

Cryptographic Analysis

- Analyze code for encryption flaws, outdated protocols, custom-developed algorithms, weak encryption, and misuse
- Automated tools will uncover some of this, including
  - Use of older hashing algorithms (MD5, SHA-1, etc)

Cryptographic Analysis

- Code and routes that handles sensitive information specifically should be reviewed
  - API Tokens
  - Credit Card Numbers
  - Social Security Numbers
  - Customer Data
- IDE Search for the following terms:
  - md5, sha1, base64, encrypt, decrypt, secure

Cryptographic Analysis Vulnerabilities

- Cryptographic Failures - OWASP Top 10 A02:2021
- Lack of Encryption
- Improper Encryption
- Insecure Token Generation/Randomness

CRYPTOGRAPHIC ANALYSIS CHECKLIST

- What are the standard encryption libraries are used for?
  - Hashing functions - password hashing, cryptographic signing, etc
  - Encryption functions - data storage, communications
- Do the strength of implemented ciphers meet industry standards?
  - Less than 256-bit encryption
  - MD5/SHA1 for password hashing
  - Any RC4 stream ciphers
  - Certificates with less than 1024-bit length keys
  - All SSL protocol versions
- Are cryptographic private keys, passwords, and secrets properly protected?

CRYPTOGRAPHIC REVIEW EXAMPLE

skea_django > settings.py > ...

```
85
86  # Password validation
87  # https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators
88
89  AUTH_PASSWORD_VALIDATORS = [
90      {
91          'NAME': 'django.contrib.auth.password_validation.
            UserAttributeSimilarityValidator',
92      },
93      {
94          'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
95      },
96      {
97          'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
98      },
99      {
100         'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
101     },
```

CRYPTOGRAPHIC REVIEW EXAMPLE

For verifying passwords, Django will find the hasher in the list that matches the algorithm name in the stored password. If a stored password names an algorithm not found in **PASSWORD_HASHERS**, trying to verify it will raise **ValueError**.

The default for **PASSWORD_HASHERS** is:

```
PASSWORD_HASHERS = [
    "django.contrib.auth.hashers.PBKDF2PasswordHasher",
    "django.contrib.auth.hashers.PBKDF2SHA1PasswordHasher",
    "django.contrib.auth.hashers.Argon2PasswordHasher",
    "django.contrib.auth.hashers.BCryptSHA256PasswordHasher",
    "django.contrib.auth.hashers.ScryptPasswordHasher",
]
```

This means that Django will use PBKDF2 to store all passwords but will support checking passwords stored with PBKDF2SHA1, argon2, and bcrypt.

CRYPTOGRAPHIC ANALYSIS EXERCISE

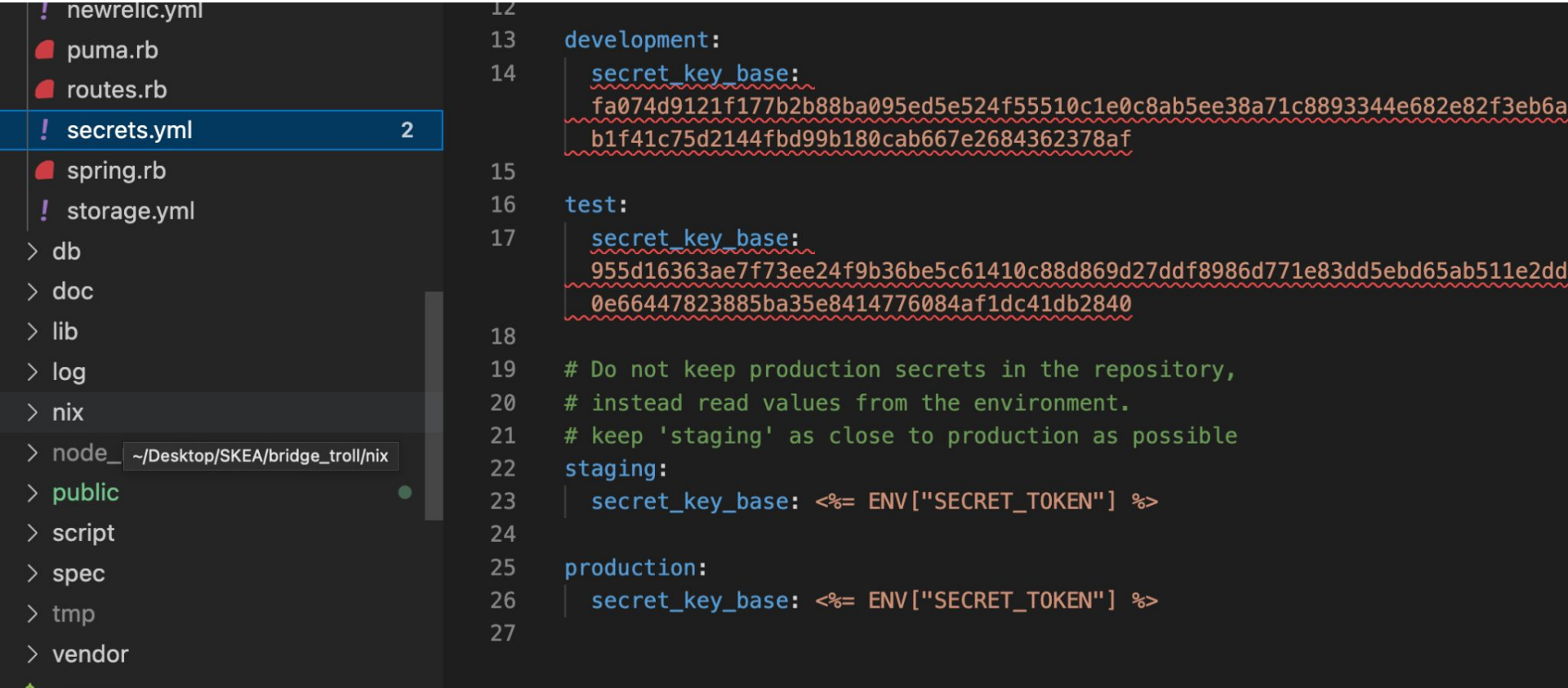
BEETHOVEN

1. **bridge_troll**
   a. Build Crypto Checklist
   b. How are passwords stored/tokens generated

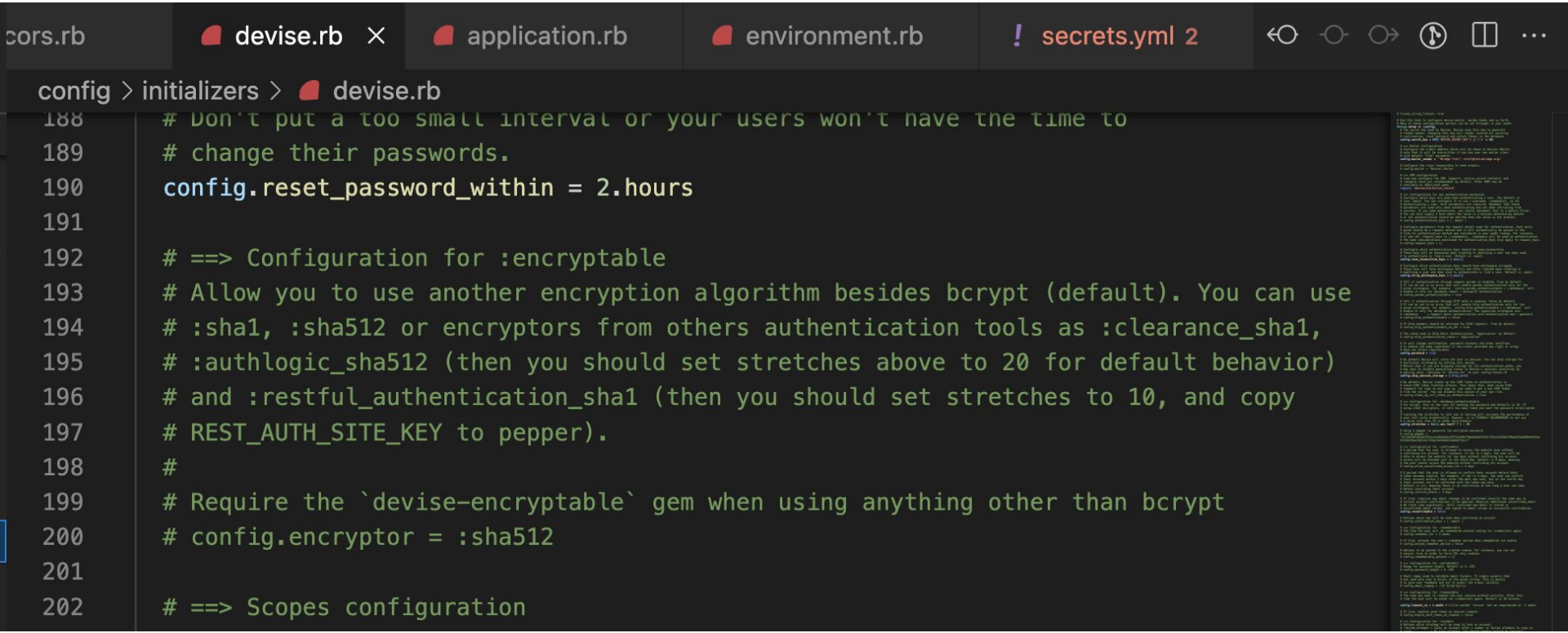
BEETHOVEN - POST-MORTEM

- Possible cryptographic issues (outside of encryption routines):
  a. Hardcoded values stored in source
  b. Same key used for dev/staging/production
  c. Password hashing

BEETHOVEN - POST-MORTEM



BEETHOVEN - POST-MORTEM



CONFIGURATION REVIEW

CONFIGURATION REVIEW

- Analyze any configurations included for security flaws
  - Includes language, framework, and server configurations
- Highly specific to the targeted language/framework
- Consult the server/framework documentation for guides on security flags and settings.
- Examples:
  - Administrative Functionality enabled through configuration files
  - CSRF settings
  - Cookie parameters

CONFIGURATION REVIEW VULNERABILITIES

- Security Misconfiguration - OWASP Top 10 A05:2021
  - Insecure defaults
  - Incomplete configurations
  - Open cloud storage
- Vulnerable and Outdated Components - OWASP Top 10 A06:2021
  - Any dependencies, libraries, services

Stopping [redacted]

By default, the [redacted] requires an administrator password to initiate a server shutdown. The default (and permanent) administrator account is ADMIN with an initial default password of ADMIN.

SPOT THE BUG

Configuration Review - Checklist

- Are any endpoints enabled through configurations properly protected with authentication and authorization?
- Are security protections implemented in framework properly configured?

Configuration Review - Checklist

- Does the target language and framework version have any known security issues?
- Are configuration-controlled security headers implemented according to recommended best practices?

CONFIGURATION REVIEW EXAMPLE

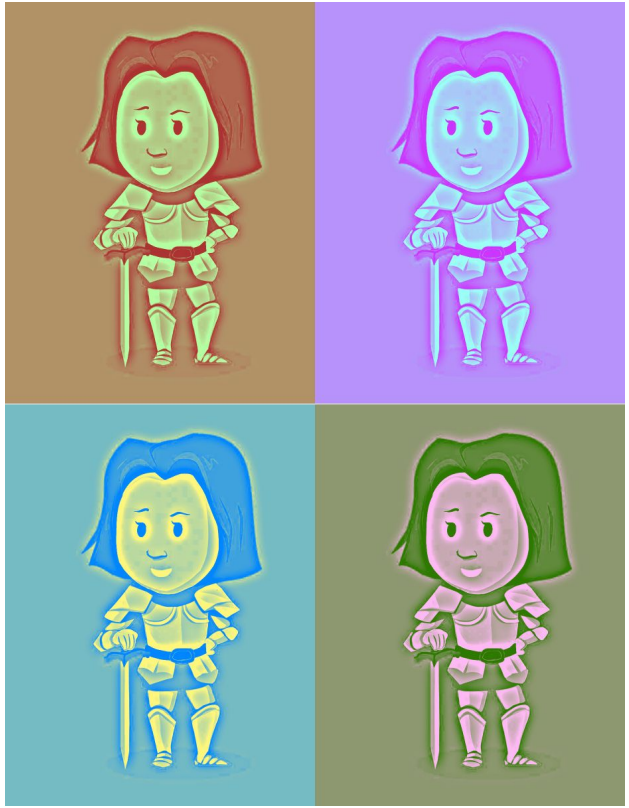
skea_django > settings.py > ...

```
12
13  import os
14
15  # Build paths inside the project like this: os.path.join(BASE_DIR, ...)
16  BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
17
18
19  # Quick-start development settings - unsuitable for production
20  # See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/
21
22  # SECURITY WARNING: keep the secret key used in production secret!
23  SECRET_KEY = '*@ty00=62-recu@lsczr-o0(%y97c(ll1gnv9pu7o^)h%#e17d'
24
25  # SECURITY WARNING: don't run with debug turned on in production!
26  DEBUG = True
27
28  ALLOWED_HOSTS = []
29
```

CONFIGURATION ANALYSIS EXERCISE

JOAN OF ARC

1. **bridge_troll**
   a. Build Config Checklist
   b. Run/review brakeman



JOAN OF ARC - POSTMORTEM

Possible Vulnerabilities

- Out of date third party libraries/dependencies
- Insecure Configurations of 3rd Party Components
  - Devise
  - Pundit
  - ???

```
Category: Unmaintained Dependency
Check: EOLRails
Message: Support for Rails 5.2.5 ended on 2022-06-01
File: Gemfile.lock
Line: 262

Confidence: Weak
Category: Cross-Site Scripting
Check: SanitizeConfigCve
Message: rails-html-sanitizer 1.3.0 is vulnerable to cross-site scripting when `select` and `style` tags are allowed (CV
E-2022-32209). Upgrade to 1.4.3 or newer
File: Gemfile.lock
Line: 287

Confidence: Weak
Category: Cross-Site Scripting
Check: LinkToHref
Message: Potentially unsafe model attribute in `link_to` href
Code: link_to("Code of Conduct", Event.find(params[:id]).code_of_conduct_url, :target => :blank, :id => "coc_url")
File: app/views/events/_form.html.erb
Line: 369

Confidence: Weak
Category: Dynamic Render Path
Check: Render
Message: Render path contains parameter value
Code: render(action => EventEditor.new(current_user, params).create.render, {})
File: app/controllers/events_controller.rb
Line: 91
```

```
Check: SanitizeConfigCve
Message: rails-html-sanitizer 1.3.0 is vulnerable to cross-site scripting when `select` and `style` tags are allowed (CV
E-2022-32209). Upgrade to 1.4.3 or newer
File: Gemfile.lock
Line: 287

Confidence: Weak
Category: Cross-Site Scripting
Check: LinkToHref
Message: Potentially unsafe model attribute in `link_to` href
Code: link_to("Code of Conduct", Event.find(params[:id]).code_of_conduct_url, :target => :blank, :id => "coc_url")
File: app/views/events/_form.html.erb
Line: 369

Confidence: Weak
Category: Dynamic Render Path
Check: Render
Message: Render path contains parameter value
Code: render(action => EventEditor.new(current_user, params).create.render, {})
File: app/controllers/events_controller.rb
Line: 91

Confidence: Weak
Category: SQL Injection
Check: SQL
Message: Possible SQL injection
Code: relation.select(:id, :first_name, :last_name).where("#{relation.connection.concat("lower(first_name)", "' '", "low
er(last_name)")} like ?", "%#{query.downcase}%")
File: app/services/user_searcher.rb
Line: 13

(END)
```

SECURITY MISCONFIGURATION - JAVA SPRING

```
# Database Configuration
spring.datasource.url=jdbc:h2:mem:AZ;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
hibernate.hbm2ddl.import_files_sql_extractor=org.hibernate.tool.hbm2ddl.MultipleLinesSqlCommandExtractor
hibernate.hbm2ddl.auto=create

# H2 Options
security.basic.enabled=true
security.basic.authorize-mode=none
spring.h2.console.enabled=true
spring.h2.console.settings.web-allow-others=true
spring.h2.console.path=/console
```

SECURITY MISCONFIGURATION - .NET

```
<sessionState mode="Off"
stateConnectionString="tcpip=localhost:42424"
sqlConnectionString="data source=localhost;user
id=sa;password=" cookieless="false" timeout="20"/>
```

ADDITIONAL RESOURCES

ADDITIONAL RESOURCES

- OWASP Code Review Guide - Includes some language-specific best practices for Java, .Net, C, C++.
- Simplicable Secure Code Review Checklist
- Infosec Institute - Secure Code Review: A Practical Approach
- OWASP Input Validation Cheatsheet
- OWASP XSS Prevention Cheatsheet
- Recommended headers for internal and external Web User Interfaces
- Wikipedia - Principle of Layered Security
- Wikipedia - Principle of Least Privilege
- OWASP ASVS (Application Security Verification Standard)

REPORTING & RETESTING

REPORTING & RETESTING

- This phase of the methodology is critical.
- Document findings in a manner that can be understood by a developer.
- Review the source after remediation using the same techniques (rinse & repeat).
- **THIS IS THE WHOLE POINT.**

Report Writing (or Ticket Creation)

- Detailed findings include:
  - Description
  - Applicable File, Function, Variables, Line Numbers
  - Risk Severity
  - Likelihood of Exploitation
  - Ease of Exploitation
  - Remediation Approach
  - References

WALKTHROUGH:
VULNERABLE TASK MANAGER

GROUP PROJECTS

THANK YOU!!! 👍

CONTACT

ABSOLUTE APPSEC - ABSOLUTEAPPSEC.COM
@ABSOLUTEAPPSEC
KEN - @CKTRICKY - KEN@ABSOLUTEAPPSEC.COM
SETH - @SETHLAW - SETH@ABSOLUTEAPPSEC.COM