SECURE FRAMEWORK: Seguridad por Defecto para Aplicaciones Web con Ruby on Rails

Director: Antonio Pérez Carrasco

José Antonio Torres Coca (53914813F)

Año académico 2024/2025

Universidad Isabel I

Universidad Isabel I

Tabla de Contenidos

1 Introducción y motivación

La Brecha de Seguridad

- Existe una **brecha significativa entre las buenas prácticas de seguridad y su implementación real** en el desarrollo web.
- Los desarrolladores a menudo **carecen de formación en ciberseguridad** o trabajan bajo presión, priorizando la funcionalidad sobre la protección.
- Consecuencias: **aplicaciones vulnerables, exposición de datos, pérdidas económicas y daños reputacionales**.

El Problema: Vulnerabilidades Comunes

- Las aplicaciones quedan expuestas a riesgos críticos definidos por el **OWASP Top 10**.
- Ejemplos clave: Inyección SQL, Cross-Site Scripting (XSS), Fallos de Autenticación y Control de Acceso.
- La solución no es aplicar parches, sino un enfoque proactivo: **"Seguridad por Diseño" (Security by Design)**.

Universidad Isabel I

Tabla de Contenidos

2 Objetivos

Universidad Isabel I

Objetivos del Proyecto

- **Objetivo General:** Crear un framework para Ruby on Rails que facilite la construcción de aplicaciones web seguras por defecto.
- **Objetivo Específicos:**
  - Analizar el estado del arte de la seguridad en Rails.
  - Diseñar una arquitectura modular y reutilizable.
  - Construir el framework como una solución práctica.
  - Validar su eficacia mediante pruebas de seguridad.

Universidad Isabel I

Tabla de Contenidos

3 Secure Framework

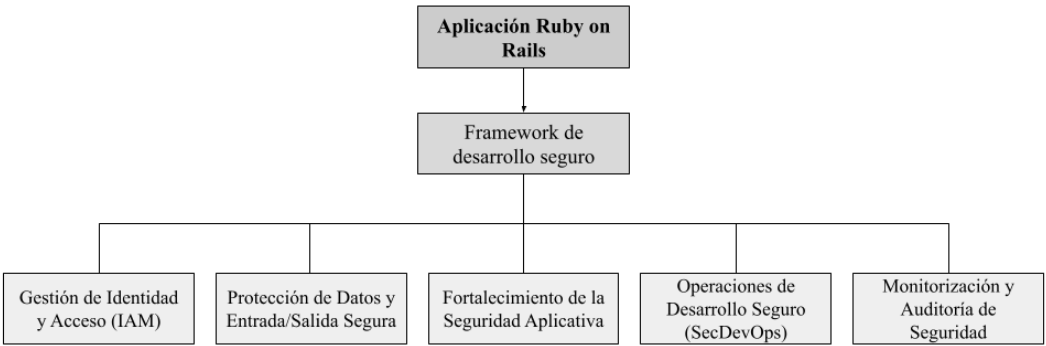
Universidad Isabel I

Propuesta: secure_framework

- Una solución distribuida como **Gema de Ruby**, para una fácil integración en aplicaciones.
- Adopta un enfoque de **"Seguridad por Defecto"**, abstrayendo la complejidad.
- **Orquesta** la instalación y configuración de componentes de seguridad esenciales.
- El objetivo es empoderar al desarrollador no experto en seguridad.

Arquitectura General del Framework

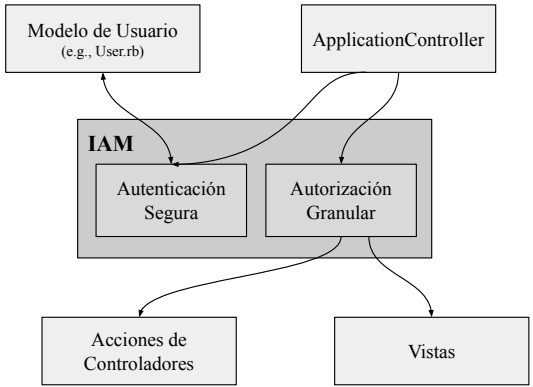
3 Secure Framework



Gestión de Identidad y Acceso (IAM)

3 Secure Framework

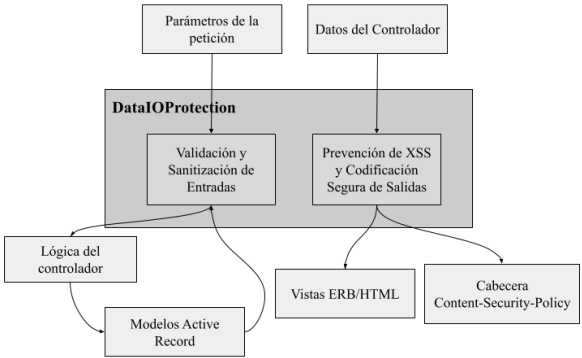
- Instala y configura las gemas **Devise** y **Pundit** en la aplicación anfitriona para garantizar que solo usuarios legítimos accedan y únicamente a los recursos permitidos.



Protección de Datos y E/S Segura

3 Secure Framework

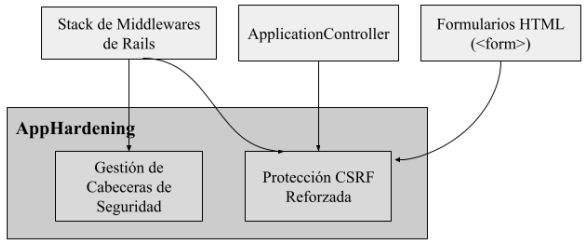
- Aplica un doble filtro de seguridad: sanitiza las entradas para prevenir **inyecciones (SQLi)** y codifica las salidas con una estricta **Política de Seguridad de Contenido (CSP)** para neutralizar ataques XSS.



Fortalecimiento de la Seguridad Aplicativa (Hardening)

3 Secure Framework

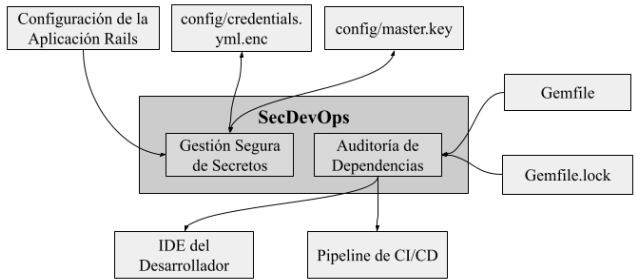
- Bloquea vectores de ataque a nivel de navegador: instala y configura cabeceras HTTP (**X-Frame-Options**, **X-Content-Type-Options**) y refuerza la protección **CSRF** con la estrategia :exception para rechazar peticiones fraudulentas.



Operaciones de Desarrollo Seguro (SecDevOps)

3 Secure Framework

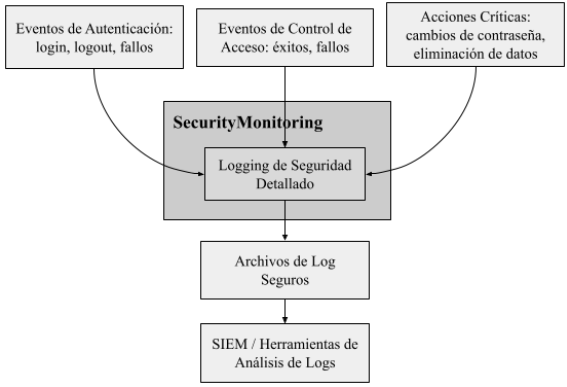
- Integra la seguridad en el flujo de trabajo mediante la **gestión segura de secretos** en el entorno y la instalación de **bundler-audit** para escanear dependencias vulnerables.



Monitorización y Auditoría de Seguridad

3 Secure Framework

- Establece un **sistema de logging dedicado** que registra eventos de seguridad críticos, como intentos de acceso fallidos y errores de autorización, para facilitar la detección de amenazas y el análisis forense.



Universidad Isabel I

Tabla de Contenidos

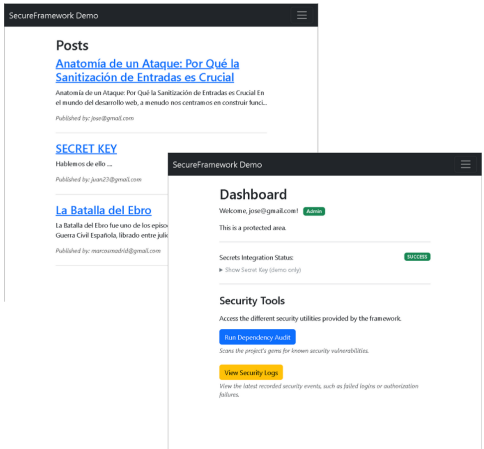
4 Validación y Pruebas

Entorno de Pruebas

4 Validación y Pruebas

Para la validación del framework se construyó una aplicación de demostración (**demo_app**) y se realizarón:

- **Análisis Estático de Código (SAST)**: Inspección del código fuente en reposo.
- **Pruebas Funcionales de Seguridad**: Simulación de escenarios de ataque en ejecución.



Análisis Estático de Código (SAST)

4 Validación y Pruebas

Brakeman (Escáner de Vulnerabilidades)

```
== Brakeman Report ==

Application Path: C:/Users/josea.JOSEANTONIOTC/Desktop/demo_app
Rails Version: 7.2.2.1
Brakeman Version: 7.0.2
Scan Date: 2025-06-12 18:15:34 +0200
Duration: 3.249935 seconds
Checks Run: BasicAuth, BasicAuthTimingAttack, CSRFTokenForgeryCVE, ContentTag, Cookie
Serialization, CreateWith, CrossSiteScripting, DefaultRoutes, Deserialize, DetailedEx
ceptions, DigestDoS, DynamicFinders, EOLRails, EOLRuby, EscapeFunction, Evaluation, E
xecute, FileAccess, FileDisclosure, FilterSkipping, ForgerySetting, HeaderDoS, I18nXS
S, JRubyXML, JSONEncoding, JSONEntityEscape, JSONParsing, LinkTo, LinkToHref, MailTo,
 MassAssignment, MimeTypeDoS, ModelAttrAccessible, ModelAttributes, ModelSerialize, N
estedAttributes, NestedAttributesBypass, NumberToCurrency, PageCachingCVE, Pathname,
PermitAttributes, QuoteTableName, Ransack, Redirect, RegexDoS, Render, RenderDoS, Ren
derInline, ResponseSplitting, RouteDoS, SQL, SQLCVEs, SSLVerify, SafeBufferManipulati
on, SanitizeConfigCve, SanitizeMethods, SelectTag, SelectVulnerability, Send, SendFil
e, SessionManipulation, SessionSettings, SimpleFormat, SingleQuotes, SkipBeforeFilter
, SprocketsPathTraversal, StripTags, SymbolDoSCVE, TemplateInjection, TranslateBug, U
nsafeReflection, UnsafeReflectionMethods, ValidationRegex, VerbConfusion, WeakRSAKey,
 WithoutProtection, XMLDoS, YAMLParsing

== Overview ==

Controllers: 4
Models: 3
Templates: 26
Errors: 0
Security Warnings: 0

== Warning Types ==


No warnings found
```

Rubocop (Analizador de Calidad y Estilo)

```
C:\Users\josea.JOSEANTONIOTC\Desktop\demo_app>bundle exec rubocop
Inspecting 3 files
...

3 files inspected, no offenses detected

Tip: Based on detected gems, the following RuboCop extension libraries might be helpf
ul:
  * rubocop-capybara (https://rubygems.org/gems/rubocop-capybara)
  * rubocop-factory_bot (https://rubygems.org/gems/rubocop-factory_bot)
  * rubocop-rspec (https://rubygems.org/gems/rubocop-rspec)
  * rubocop-rspec_rails (https://rubygems.org/gems/rubocop-rspec_rails)

You can opt out of this message by adding the following to your config (see https://d
ocs.rubocop.org/rubocop/extensions.html#extension-suggestions for more options):
  AllCops:
    SuggestExtensions: false
```

Universidad Isabel I

Análisis Dinámico (RSpec)

4 Validación y Pruebas

Se ha desarrollado una suite de 29 tests, utilizando la herramienta **rspec**, que abordan:

- Pruebas sobre el ciclo de vida del usuario y bloqueo de cuentas para mitigar ataques de fuerza bruta.
- Simulación de ataques de escalada de privilegios y de Referencia Directa Insegura a Objetos (IDOR).
- Verificación de que las entradas con scripts maliciosos (XSS) son sanitizadas y las peticiones fraudulentas sin token de seguridad (CSRF) son rechazadas.
- Comprobación de que la aplicación envía cabeceras de seguridad HTTP (CSP, X-Frame-Options) y auditoría de dependencias para detectar librerías vulnerables.
- Registro de eventos críticos, como inicios de sesión fallidos y errores de autorización, en un log de seguridad dedicado cuyo acceso está protegido.

Universidad Isabel I

Análisis Dinámico (RSpec)

4 Validación y Pruebas

```
C:\Users\josea.JOSEANTONIOTC\Desktop\demo_app>bundle exec rspec
.............................

Finished in 3.47 seconds (files took 3.51 seconds to load)
29 examples, 0 failures
```

Universidad Isabel I

Tabla de Contenidos

5 Conclusiones Finales

Conclusión

5 Conclusiones Finales

Secure Framework ha demostrado mediante pruebas experimentales ser una solución eficaz y robusta para la implementación e integración de medidas de seguridad en aplicaciones Ruby on Rails, evitando vulnerabilidades comunes y rompiendo la barrera entre las buenas prácticas de seguridad y su implementación real.

Universidad Isabel I

¡Gracias por su atención!

¿Alguna pregunta?