# Running a Ruby on Rails App securely

Lightning Talk

# Intro

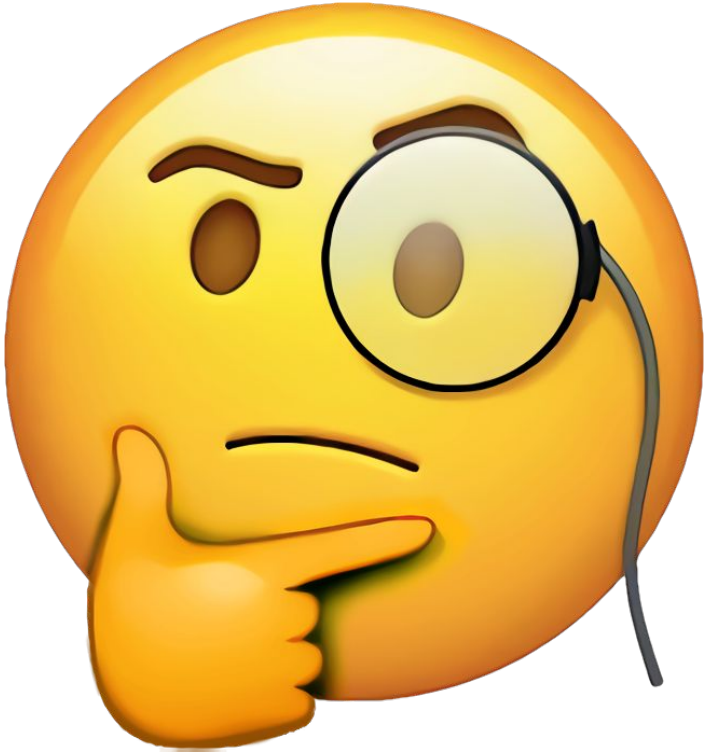
# Me

- Rails Developer
- System Administrator
- Incident Responder
- SANS Institute Student
  - M.Sc. Information Security Engineering
- jonathan.schlue@aboutsource.net

# A few words about Security

# Give me a secure system

Shortest Answer wins



# Balance

- “It *is* possible to tighten security to the point where the system is unusable. **Security and convenience must be balanced**. The trick is to create a secure and useful system.”
- https://wiki.archlinux.org/title/Security#Concepts


BALANCE
HISTORY.COM
imgflip.com


KÆRGÅR
DAS ORIGI
Balance
Streichzart – 30% weniger
Fett als Butter*
UNGESALZEN

# Threat Modeling

# 4-Step-Framework

- What are we building?
- What could possibly go wrong?
- What are we going to do about it?
- Did we do a good job?

# 4-Step-Framework

- **What are we building?**
- What could possibly go wrong?
- What are we going to do about it?
- Did we do a good job?

# (1) What are we building?

# Everything you need.

**Rails is a full-stack framework.** It ships with all the tools needed to build amazing web apps on both the front and back end.

Rendering HTML templates, updating databases, sending and receiving emails, maintaining live pages via WebSockets, enqueuing jobs for asynchronous work, storing uploads in the cloud, providing solid security protections for common attacks. Rails does it all and so much more.

https://rubyonrails.org/

# Everything you need

- Rendering HTML templates
- updating databases
- sending and receiving emails
- maintaining live pages via websockets
- enqueueing jobs for asynchronous work
- storing uploads in the cloud
- providing solid security protections for common attacks
- ...

# Everything you need

- **Rendering HTML templates**
- **updating databases**
- **sending and receiving emails**
- maintaining live pages via websockets
- enqueueing jobs for asynchronous work
- **storing uploads in the cloud**
- providing solid security protections for common attacks
- ...

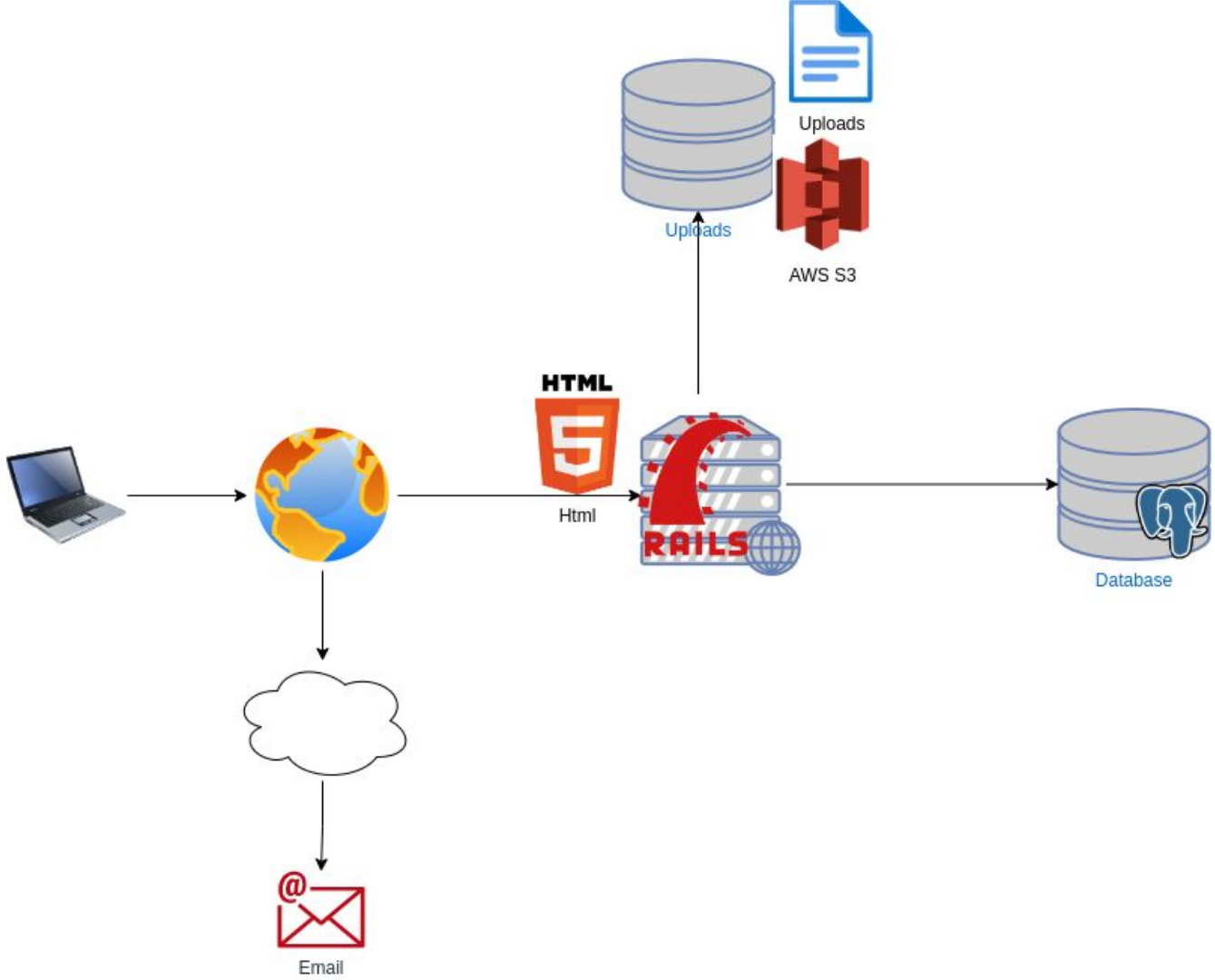
Html


Email


Uploads


AWS S3


HTML
Html
Documents
RAILS
@
Email
Database


Uploads
Uploads
AWS S3
HTML
Html
RAILS
Database
Email

# 4-Step-Framework

- ~~What are we building?~~
- **What could possibly go wrong?**
- What are we going to do about it?
- Did we do a good job?

# (2) What could possibly go wrong?

# Everything you need

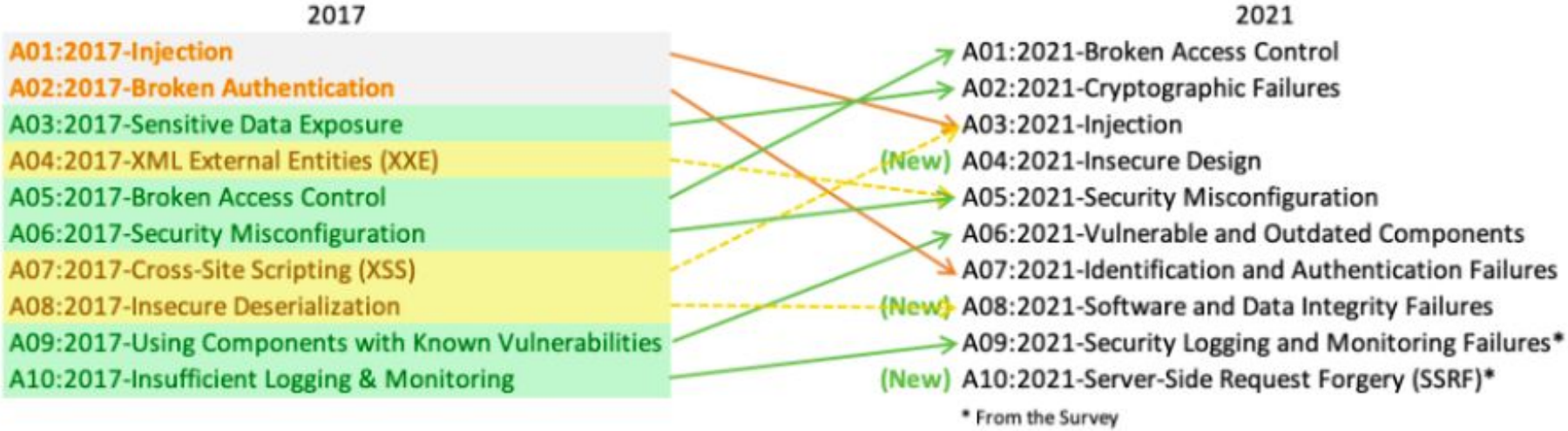
- Rendering HTML templates
- updating databases
- sending and receiving emails
- maintaining live pages via websockets
- enqueueing jobs for asynchronous work
- storing uploads in the cloud
- **providing solid security protections for common attacks**
- ...

# Common Attacks


- “The OWASP Top 10 is a standard awareness document for developers and web application security. **It represents a broad consensus about the most critical security risks to web applications**.”
- https://owasp.org/www-project-top-ten/

# Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.



# Owasp Top 10 (2021)

- A01 Broken Access Control
- A02 Cryptographic Failures
- A03 Injection
- A04 Insecure Design
- A05 Security Misconfiguration
- A06 Vulnerable and Outdated Components
- A07 Identification and Authentication Failures
- A08 Software and Data Integrity Failures
- A09 Security Logging and Monitoring Failures
- A10 Server Side Request Forgery (SSRF)

# 4-Step-Framework

- ~~What are we building?~~
- ~~What could possibly go wrong?~~
- **What are we going to do about it?**
- Did we do a good job?

# (3) What are we going to do about it?

# Rails Guide: Securing Rails Applications

# Securing Rails Applications

This manual describes common security problems in web applications and how to avoid them with Rails.

After reading this guide, you will know:

- **All countermeasures *that are highlighted*.**
- **The concept of sessions in Rails, what to put in there and popular attack methods.**
- **How just visiting a site can be a security problem (with CSRF).**
- **What you have to pay attention to when working with files or providing an administration interface.**
- **How to manage users: Logging in and out and attack methods on all layers.**
- **And the most popular injection attack methods.**

## Chapters

1. Introduction
2. Sessions
   - What are Sessions?
   - Session Hijacking
   - Session Storage
   - Rotating Encrypted and Signed Cookies Configurations
   - Replay Attacks for CookieStore Sessions
   - Session Fixation
   - Session Fixation - Countermeasures
   - Session Expiry
3. Cross-Site Request Forgery (CSRF)
   - CSRF Countermeasures
4. Redirection and Files
   - Redirection
   - File Uploads
   - Executable Code in File Uploads
   - File Downloads
5. Intranet and Admin Security
   - Additional Precautions
6. User Management
   - Brute-Forcing Accounts
   - Account Hijacking
   - CAPTCHAs
   - Logging
   - Regular Expressions
   - Privilege Escalation
7. Injection

## 1 Introduction

Web application frameworks are made to help developers build web applications. Some of them also help you with securing the web application. In fact one framework is not more secure than another: If you use it correctly, you will be able to build secure apps with many frameworks. Ruby on Rails has some clever helper methods, for example against SQL injection, so that this is hardly a problem.

In general there is no such thing as plug-n-play security. Security depends on the people using the framework, and sometimes on the development method. And it depends on all layers of a web application environment: The back-end storage, the web server, and the web application itself (and possibly other layers or applications).

The Gartner Group, however, estimates that 75% of attacks are at the web application layer, and found out "that out of 300 audited sites, 97% are vulnerable to attack". This is because web applications are relatively easy to attack, as they are simple to understand and manipulate, even by the lay person.

The threats against web applications include user account hijacking, bypass of access control,

# Securing Rails Applications - Breakdown

“This manual describes **common security problems in web applications** and how to avoid them with Rails.”

1. Introduction
2. Sessions
3. CSRF
4. Redirection and Files
5. Intranet and Admin Security
6. User Management
7. Injection
8. Unsafe Query Generation
9. HTTP Security Headers
10. Environmental Security
11. Dependency Management and CVEs
12. Additional Resources

# Owasp Top 10 (2021)

- A01 Broken Access Control
- **A02 Cryptographic Failures**
- A03 Injection
- **A04 Insecure Design**
- A05 Security Misconfiguration
- **A06 Vulnerable and Outdated Components**
- A07 Identification and Authentication Failures
- A08 Software and Data Integrity Failures
- A09 Security Logging and Monitoring Failures
- A10 Server Side Request Forgery (SSRF)

# A02 Cryptographic Failures

# Don't roll your own crypto!

- ...unless you have a PhD in Cryptography
- **Use standard crypto libraries**, e.g. gem **bcrypt**
  - "bcrypt-ruby is a **Ruby binding for the OpenBSD bcrypt() password hashing algorithm**, allowing you to easily store a secure hash of your users' passwords."
  - **Rails versions >= 3** ship with **ActiveModel::SecurePassword** which uses **bcrypt-ruby**

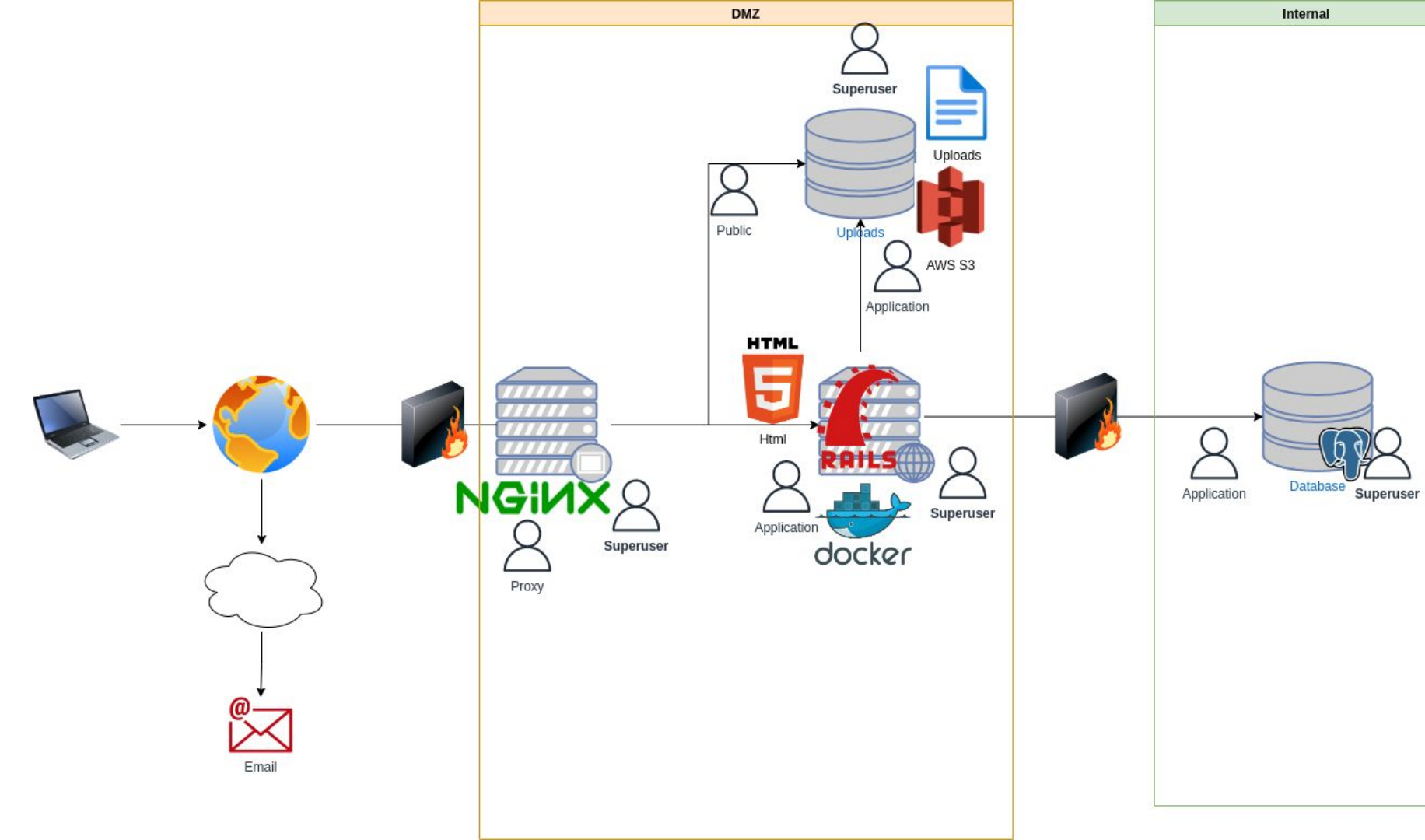
Example using Active Record (which automatically includes `ActiveModel::SecurePassword`):

```
# Schema: User(name:string, password_digest:string, recovery_password_digest:string)
class User < ActiveRecord::Base
  has_secure_password
  has_secure_password :recovery_password, validations: false
end
```

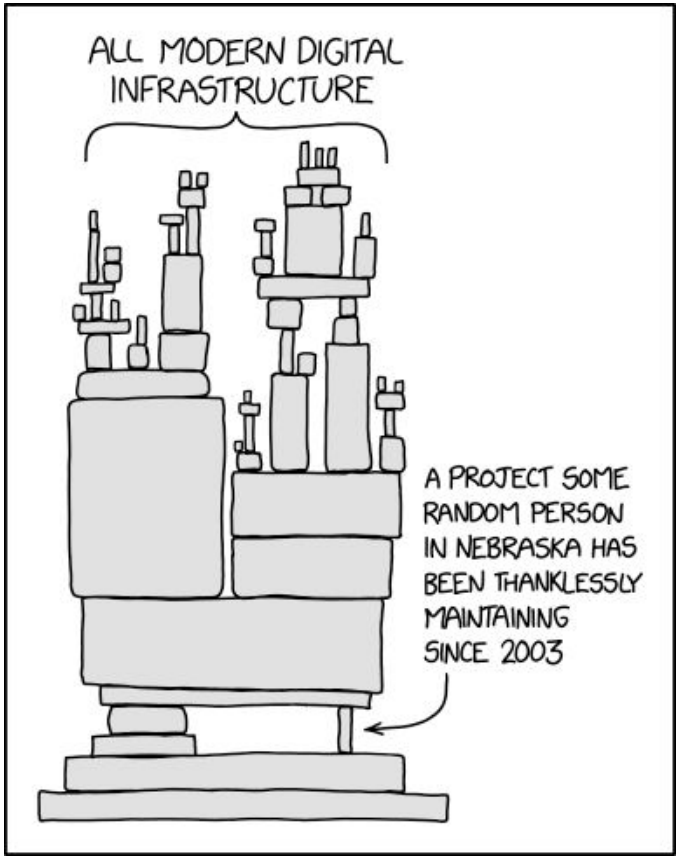
# A04 Insecure Design

# Principle of Least Privilege

“The principle that users and programs should only have the necessary privileges to complete their tasks.”


DMZ
Internal
Superuser
Uploads
Public
Uploads
AWS S3
Application
HTML
Html
RAILS
Application
Superuser
docker
NGINX
Superuser
Proxy
Application
Database
Superuser
Email

# A06: Vulnerable and Outdated Components




**https://www.explainxkcd.com/wiki/index.php/2347:_Dependency**

# Keep Software patched and up-to-date!

- Aim for a Software Bill of Materials (SBOM)
  - https://snyk.io/blog/building-sbom-open-source-supply-chain-security/
  - “is a complete list of all software components used across an organization. The software bill of material list is made up of third-party open source libraries, vendor provided packages and first-party artifacts built by the organization.”

# Supply-Chain-Security

- Monitor for Security-Patches
  - Application (Gemfile.lock, package-lock.json/yarn.lock)
    - Github Dependabot https://github.com/dependabot
  - Docker Image Vulnerabilities
    - Docker Hub Vulnerability Scanning https://docs.docker.com/docker-hub/vulnerability-scanning/
    - https://snyk.io/product/container-vulnerability-management/
  - Server Packages (apt, ...)
    - sensu, nagios
    - prometheus, grafana

# 4-Step-Framework

- What are we building
- **What could possibly go wrong?**
- What are we going to do about it?
- Did we do a good job?

# 4-Step-Framework

- ~~What are we building?~~
- ~~What could possibly go wrong?~~
- ~~What are we going to do about it?~~
- **Did we do a good job?**

# (4) Did we do a good job?

# QA & Testing

- Unit Testing
  - rspec
- Integration testing
  - capybara
- Vulnerability scanning / SAST (Static Analysis Security Testing)
  - brakeman https://brakemanscanner.org/
  - Snyk Code https://snyk.io/product/snyk-code/
- Vulnerability Assessments
- Penetration Testing
- re-evaluate Threat-Models

# Q&A

# Discussion

# Further Readings

- General Concepts
  - https://wiki.archlinux.org/title/Security#Concepts
  - https://csrc.nist.gov/glossary/term/principle_of_least_privilege
  - https://www.amazon.de/Threat-Modeling-Designing-Adam-Shostack/dp/1118809998
- Common Attacks & Protections
  - https://owasp.org/www-project-top-ten/
  - https://owasp.org/Top10/
  - https://guides.rubyonrails.org/security.html
  - https://github.com/bcrypt-ruby/bcrypt-ruby
  - https://api.rubyonrails.org/classes/ActiveModel/SecurePassword/ClassMethods.html#method-i-has_secure_password
- Supply-Chain-Security
  - https://snyk.io/blog/building-sbom-open-source-supply-chain-security/
  - https://github.com/dependabot
  - https://docs.docker.com/docker-hub/vulnerability-scanning/
  - https://snyk.io/product/container-vulnerability-management/