Brakeman and Jenkins: The Duo Detects Defects in Ruby on Rails Code

Justin Collins
Tin Zaw

AppSec USA
September 23, 2011
About Us

Justin Collins - @presidentbeef

Tin Zaw - @tzaw
Our Philosophy: Light Touch

Use *tools* to detect and report security defects in code *early* in the development cycle with *minimal impact* to development workflow.
McGraw’s Touch Point #1
Code Review (Tools)
Defect Cost Curve

- Programming defect found via Pair Programming
- Programming defect found via Continuous Integration
- Design or programming defect found via Test Driven Development (TDD)
- Requirements or design defect found via Active Stakeholder Participation
- Requirements or design defect found via Model Storming
- Defect found via independent parallel testing
- Defect found via traditional system testing
- Defect found via a review or inspection
- Requirements defect found via traditional acceptance testing

Copyright 2006-2009 Scott W. Ambler
Defect Cost Curve

- Programming defect found via Pair Programming
- Programming defect found via Continuous Integration
- Design or programming defect found via Test Driven Development (TDD)
- Requirements or design defect found via Active Stakeholder Participation
- Requirements or design defect found via Model Storming
- Defect found via independent parallel testing
- Defect found via a review or inspection
- Requirements defect found via traditional acceptance testing
- Design defect found via traditional system testing

Application: Security Testing

Length of Feedback Cycle
Defect Cost Curve

- Programming defect found via Pair Programming
- Programming defect found via Continuous Integration
- Design or programming defect found via Test Driven Development (TDD)
- Requirements or design defect found via Active Stakeholder Participation
- Requirements or design defect found via Model Storming
- Defect found via independent parallel testing
- Defect found via a review or inspection
- Requirements defect found via traditional acceptance testing
- Design defect found via traditional system testing

Brakeman + Jenkins

Length of Feedback Cycle

Copyright 2006-2009 Scott W. Ambler
Static vs. Dynamic Analysis

• Penetration Testing Pros
  – Replicates real life deployment
  – Entire application stack, configuration

• Penetration Testing Cons
  – Reports symptoms, not root causes
  – Setup time, find defects late during QA cycle
  – Incomplete view of running app
Static vs. Dynamic Analysis

• Static Code Analysis Pros
  – Early detection of defects
  – Integrated into developer’s workflow
  – No deployment required

• Static Code Analysis Cons
  – Limited to code
  – Need access to source code
# Existing Static Analysis Tools for Security Defects

<table>
<thead>
<tr>
<th>Language/Cross-Platform</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>C/C++</td>
<td>&lt;many&gt;</td>
</tr>
<tr>
<td>C#/.Net</td>
<td>&lt;many&gt;</td>
</tr>
<tr>
<td>Java</td>
<td>&lt;many&gt;</td>
</tr>
<tr>
<td>Ruby</td>
<td>?</td>
</tr>
<tr>
<td>Ruby on Rails</td>
<td>Brakeman</td>
</tr>
</tbody>
</table>
Manual Workflow

Get Latest Code → Run Tool → Examine Results
Manual Workflow

1. Get Latest Code
2. Run Tool
3. Examine Results

Repeat
Automated Workflow

Let tools alert you when there is a problem
Brakeman

http://brakemanscanner.org
Ruby on Rails

Web application framework using the Ruby language

Built on the model-view-controller design pattern

“Convention over configuration” – encourages assumptions which lead to default behavior
Brakeman Application Flow

1. Parse App Code
2. Clean up & Organize
3. Inspect Results
4. Generate Report
Vulnerabilities Brakeman Detects

Cross site scripting
SQL injection
Command injection
Unprotected redirects
Unsafe file access
Default routes
Insufficient model validation
Version-specific security issues
Unrestricted mass assignment
Dangerous use of eval()
...and more!
Example: Cross Site Scripting (Rails 2.x)

<b>Results for <%= params[:query] %></b>
Example: Cross Site Scripting (Rails 3.x)

<code>b</code>Results for <%= raw params[:query] %></code>
Example: Cross Site Scripting (Rails 3.x)

Results for <%= raw params[:query] %>

Unescaped parameter value near line 1:
params[:query]
Example: SQL Injection

```ruby
username = params[:user][:name]

User.find(:all,
    :conditions => "name like '%#{username}%'")
```
Example: SQL Injection

```
username = params[:user][:name]

User.find(:all,
  :conditions => "name like '%#{username}%'")
```

Possible SQL injection near line 87:
```
User.find(:all, :conditions => ("name like '
  %#{params[:user][:name]}%"")
```
Extended Example - Filters

class ApplicationController < ActionController::Base

  def set_user
    @user = User.find(params[:user_id])
  end

end

Method in application controller sets the @user variable
class UserController < ApplicationController
  before_filter :set_user
  def show
    end
  end

User controller calls set_user before any action
Extended Example - Filters

<%= raw @user.bio %>
Extended Example - Filters

Data flow followed from filter through to the view
Extended Example - Filters

```ruby
<% raw @user.bio %>

Unescaped model attribute near line 5:
User.find(params[:id]).bio
```
Example: Mass Assignment

class User < ActiveRecord::Base
end
Example: Mass Assignment

class UsersController < ApplicationController
  #...
  def new
    @user = User.new(params[:user])
  #...
  end
end

Excerpt of Users controller generated by Rails
Example: Mass Assignment

class UsersController < ApplicationController
    #...
    def new
        @user = User.new(params[:user])
        #...
    end
end

Unprotected mass assignment near line 43: User.new(params[:user])
Open source continuous integration server

http://jenkins-ci.org
How Jenkins Works

Monitor Conditions

Run Jobs

Aggregate Results
How Jenkins Works

Monitor Conditions

Run Jobs

Aggregate Results

git push
svn commit

brakeman

Security Warnings
Brakeman Plugin for Jenkins

Run Brakeman → Collect Warnings → Generate Reports
Some Results

## Warnings Trend

<table>
<thead>
<tr>
<th>All Warnings</th>
<th>New Warnings</th>
<th>Fixed Warnings</th>
</tr>
</thead>
<tbody>
<tr>
<td>181</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

## Summary

<table>
<thead>
<tr>
<th>Total</th>
<th>High Priority</th>
<th>Normal Priority</th>
<th>Low Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>181</td>
<td>44</td>
<td>7</td>
<td>130</td>
</tr>
</tbody>
</table>

## Details

<table>
<thead>
<tr>
<th>Type</th>
<th>Total</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross Site Request Forgery</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Cross Site Scripting</td>
<td>156</td>
<td>Red-Green-Blue</td>
</tr>
<tr>
<td>Default Routes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Dynamic Render Path</td>
<td>10</td>
<td>Red-Blue</td>
</tr>
<tr>
<td>Format Validation</td>
<td>6</td>
<td>Red</td>
</tr>
<tr>
<td>Redirect</td>
<td>6</td>
<td>Red-Blue</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>181</strong></td>
<td></td>
</tr>
</tbody>
</table>
Using Brakeman

gem install brakeman
cd your/rails/app
brakeman
Resources

- Ruby
  - http://ruby-lang.org
- Ruby on Rails
  - http://rubyonrails.org
- Ruby on Rails Security Guide
  - http://guides.rubyonrails.org/security.html
- Brakeman
  - http://brakemanscanner.org
- Jenkins
  - http://jenkins-ci.org
- Brakeman plugin for Jenkins
  - http://github.com/presidentbeef/brakeman-jenkins-plugin