Ruby On Rails Security Project

Heiko Webers
42@bauland42.de

About

Heiko is the founder of bauland42.

I'm running the Ruby on Rails Security Project
at http://www.rorsecurity.info

The Ruby on Rails Security Guide was made possible by the OWASP Spring of Code 2007 & Summer of Code 2008.

The Guide

The Guide version 2 covers Ruby on Rails, MySQL and (a bit) server security.

Every chapter starts with an introduction and sometimes images of attack scenarios.

The countermeasures are highlighted, so you can easily scan the Guide.

It is now the official security manual for Ruby on Rails at http://guides.rubyonrails.org/security.html

Thanks to my reviewers Steve Jones and Anthony Shireman.

Ruby on Rails

Ruby is a dynamic object-oriented programming language based (partly) on Perl, Smalltalk, Eiffel, Ada and Lisp.

```
"Ruby".index("u")
```

Ruby on Rails is an open source web application framework based on Ruby. Rails embraces „convention over configuration“, Model-View-Controller (MVC), Don‘t-Repeat-Yourself (DRY) and testing.

Rails includes the abstract database wrapper ActiveRecord.

```
class Project < ActiveRecord::Base
  has_many :comments
end
```

```
<% @project.comments.each do |c| %>
  <li><%= c.comment %></li>
<% end %>
```

```
class ProjectsController < ApplicationController
  def show
    @project = Project.find(params[:id])
  end
end
```

SQL Injection

Rails is potentially vulnerable to SQL Injection, too.

```
Project.find(:all, :conditions => "name =
'#{params[:name]}'")
```

But Rails has some clever helper methods, so SQL Injection is hardly a problem.

```
Project.find_by_name(params[:name])
```

```
Project.find(:first, :conditions => ["name = ?", params[:name]])
```

```
Project.find(:first, :conditions => {:name => params[:name]})
```

XSS

Rails provides a whitelist filter for user input.

```
tags = %w(b strong i em)
s = sanitize(user_input, :tags => tags)
```

And there is an output filter for SGML.
The SafeErb plugin reminds you to escape external strings.

```
<%= html_escape(user_input) %>
```

The Guide talks about several other forms of injection, too: Ajax, CSS, Textile, Command Line, Header Injection.

CSRF on Rails

```
<img src="http://www.webapp.com/project/1/destroy">
```

Make sure POST actions may not be used over GET.

```
verify :method => :post,
       :only => [:destroy],
       :redirect_to => {:action => :index}
```

Include a security token in non-GET requests and check it on the server side.

```
protect_from_forgery :secret =>
   "123456789012345678901234567890"
```

Mass Assignment

Mass assignment saves you much work.

```
def signup
  @user = User.create(params[:user])
end
```

But it may lead to new problems.

```
params[:user] #=> {:name => “ow3ned”,
                   :admin => true}
```

As a countermeasure, tell Rails which attributes may be changed by mass assignment.

```
attr_accessible :name
```

Sessions

You have several options for sessions in Rails: ActiveRecordStore and CookieStore are the most popular.

CookieStore stores the session data in the cookie, in clear-text. Don‘t use a weak server-side secret, don‘t store any secrets in the session and beware of replay attacks.

```
Set-Cookie: app_session=
3b3c1b4398d8cba2ac1ec8687e4ab166
```

```
config.action_controller.session = {
  :session_key => ‘app_session’,
  :secret      =>
 ‘0x0dkfj3927dkc7djdh36rkckdfzsg’ }
```

User Management

There is a popular plugin for user management: restful_authentication. It takes care of activation of the user account, session management and saves the password as a salted hash.

`http://webapp.com/user/activate/a32409deaed49adee5382`
`http://webapp.com/user/activate/`

Other topics

Redirection, File Up- and Downloads.

Intranet and Admin security.

Logging, good passwords, regular expressions and privilege escalation.

Thank you