The Evolution of Rails Security

Justin Collins
@presidentbeef

RailsConf 2018

Ugh, About Me

8 years of application security

(AT&T Interactive, Twitter, SurveyMonkey)

8 years working on Brakeman OSS

(Static analysis security tool for Rails)

4 years working on  BRAKEMANPRO

(More **pro** static analysis security tool for Rails)

July 2004

Rails Released to the World

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-talk/107370

Subject: **[ANN] Rails 0.5.0: The end of vaporware!**
From: **David Heinemeier Hansson** <david@ u h k g o
Date: Sun, 25 Jul 2004 04:43:00 +0900

I?ve been talking (and hyping) Rails for so long that it?s all wierd to finally have it out in the open. Mind you, we?re still not talking about a 1.0 release, but the package currently on offer is still something I?m very comfortable to share with the world. Undoubtedly, there could be more documentation and more examples, but Real Artists Ship and this piece will grow in public. Enjoy Rails!

Documentation, download: http://www.rubyonrails.org

What is Rails?
==============

Rails is a open source web-application framework for Ruby. It ships with an answer for every letter in MVC: Action Pack for the Controller and View, Active Record for the Model.

Everything needed to build real-world applications in less lines of code than other frameworks spend setting up their XML configuraion files. Like Basecamp, which was launched after 4 KLOCs and two months of developement by a single programmer.

Being a full-stack framework means that all layers are built to work seemlessly together. That way you Don?t Repeat Yourself (DRY) and you can use a single language from top to bottom. Everything from templates to control flow to business logic is written in Ruby?the language of love for industry heavy-weights

In striving for DRY compliance, Rails shuns configuration files and annotations in favor of reflection and run-time extensions. This means the end of XML files telling a story that has already been told in code. It means no compilation phase: Make a change, see it work. Meta-data is an implementation detail left for the framework to handle.

December 2005

Rails 1.0

http://weblog.rubyonrails.org/2005/12/13/rails-1-0-party-like-its-one-oh-oh/

Rails 1.0: Party like it's one oh oh!

Posted by David, December 13, 2005 @ 9:02 pm in Releases

15 months after the first public release, Rails has arrived at the big 1.0. What a journey! We've gone through thousands of revisions, tickets, and patches from hundreds of contributors to get here. I'm incredibly proud at the core committer team, the community, and the ecosystem we've raised around this framework.

Rails 1.0 is mostly about making all the work we've been doing solid. So it's not packed with new features over 0.14.x, but has spit, polish, and long nights applied to iron out kinks and ensure that it works mostly right, most of the time, for most of the people. Yes, we still have pending tickets, but we will always have pending tickets. If I had accepted that fact back in February, we would probably have been at 2.0 now ;).

Alongside 1.0, we've also been working on a new web site, which premieres today as well. It's a 37signals-powered redesign that streamlines and decrufts us into a much cleaner profile that hopefully will make it even easier for people to get excited and try out Ruby on Rails. It's online at www.rubyonrails.org and includes two brand new screencasts.

August 2006

CVE-2006-4111
CVE-2006-4112

Common Vulnerabilities *and* Exposures

https://cve.mitre.org/

CVE-2006-4111 & CVE-2006-4112

Panic!

Rails 1.1.5: Mandatory security patch (and more)

Posted by David, August 9, 2006 @ 4:30 pm in **Releases**

We're still hard at work on Rails 1.2, which features all the new dandy REST stuff and more, but a serious security concern has come to our attention that needed to be addressed sooner than the release of 1.2 would allow. So here's Rails 1.1.5!

This is a MANDATORY upgrade for anyone not running on a very recent edge (which isn't affected by this). If you have a public Rails site, you MUST upgrade to Rails 1.1.5. The security issue is severe and you do not want to be caught unpatched.

The issue is in fact of such a criticality that we're not going to dig into the specifics. No need to arm would-be assalients.

So upgrade today, not tomorrow. We've made sure that Rails 1.1.5 is fully drop-in compatible with 1.1.4. It only includes a handful of bug fixes and no new features.

For the third time: This is not like "sure, I should be flossing my teeth". This is "yes, I will wear my helmet as I try to go 100mph on a motorcycle through downtown in rush hour". It's not a suggestion, it's a prescription. So get to it!

http://weblog.rubyonrails.org/2006/8/9/rails-1-1-5-mandatory-security-patch-and-other-tidbits/

Panic!

Security update: Rails 1.0, 1.1.3 not affected

Posted by David, August 10, 2006 @ 3:38 am

Good news: Rails 1.0 and prior is not affected by the latest security breach we've experienced. Neither is Rails 1.1.3. We're currently investigating further just how contaminated 1.1.0, 1.1.1, 1.1.2, and 1.1.4 are. We'll give you more updates as soon as we have the information. Our first priority has been to get a fix out, now we'll get to the very bottom of this.

Believe you me, we take this *extremely* seriously. The entire core team is working on this investigation. We've currently made the trade-off to keep the details secret to protect people still in the process of upgrading. Once ample time for upgrading has been given and we have investigated this matter to its depth, we'll release a complete report detailing all our findings.

Thank you for your patience and understanding. We fully understand that nothing can quite make your heart pump, as knowing there's something wrong, but not being entirely sure what to do about it. It's OK to vent that frustration in the comments to this post.

http://weblog.rubyonrails.org/2006/8/10/security-update-rails-1-0-not-affected/

Panic!

Rails 1.1.6, backports, and full disclosure

Posted by David, August 10, 2006 @ 4:34 pm in Releases

The cat is out of the bag, so here's the full disclosure edition of the current security vulnerability. With Rails 1.1.0 through 1.1.5 (minus the short-lived 1.1.3), you can trigger the evaluation of Ruby code through the URL because of a bug in the routing code of Rails. This means that you can essentially take down a Rails process by starting something like /script/profiler, as the code will run for a long time and that process will be hung while it happens. Other URLs can even cause data loss.

We've backported a fix to all the affected versions for those of you that can't update. You'll have to apply the diff for your version:

- Patch for Rails 1.1.0
- Patch for Rails 1.1.1
- Patch for Rails 1.1.2
- Patch for Rails 1.1.4
- Patch for Rails 1.1.5: Upgrade to Rails 1.1.6.

http://weblog.rubyonrails.org/2006/8/10/rails-1-1-6-backports-and-full-disclosure/

GET http://localhost:3000/db/schema

```
in "traverse_to_controller" method
  Recognized route segment: db
  Looking for controller DbController in module Object
  in "attempt_load" method
  Loading DbController in module Object in path db_controller
  No controller found, so looking for module: Db
  in "attempt_load" method
  Loading Db in module Object in path db

in "traverse_to_controller" method
  Recognized route segment: schema
  Looking for controller SchemaController in module Db
  in "attempt_load" method
  Loading SchemaController in module Db in path db/schema_controller
  No controller found, so looking for module: Schema
  in "attempt_load" method
  Loading Schema in module Db in path db/schema
    found ./db/schema.rb as a possible match

...
```

https://blog.evanweaver.com/2006/08/12/anatomy-of-an-attack-against-1-1-4/

Remote Code Execution

October 2007

CVE-2007-5380

http://weblog.rubyonrails.org/2007/10/5/rails-1-2-4-maintenance-release/

example.com/home/?_session_id=...

If you're using RESTful routing, pay special attention to the changes to route generation and recognition.

The previous use of the semicolon in URLs has been replaced with a regular /. For instance /person/1;edit has become /person/1/edit. This change was made as several libraries, including mongrel, mistakenly treated semi-colons as query string seperators[sic] and some browsers and http libraries misbehaved.

February 2011

CVE-2011-0447

http://weblog.rubyonrails.org/2011/2/8/csrf-protection-bypass-in-ruby-on-rails/

Cross-Site Request Forgery


```
https://bank.com/transfer?amount=100000&to=attacker1337
```

Cross-Site Request Forgery Protection

"Synchronizer Token Pattern"

1. Save a CSRF token to the session
2. Insert the CSRF token in forms
3. Match tokens on POST/DELETE/PATCH

CSRF Token in Form

```
<form accept-charset="UTF-8" action="/users" class="new_user" id="new_user" method="post">
  <input name="authenticity_token" type="hidden" value="NvyZey4G/GuBEr6RQJTrsfX1iiptwSBWiu9XOo9h4cE=" />
```

What about XHR?

It's cool, an attacker can't make an XHR request

[WEB SECURITY] CSRF: Flash + 307 redirect = Game Over

Phillip Purviance phillip.purviance at whitehatsec.com
Thu Feb 10 14:22:20 EST 2011

- Previous message (by thread): [WEB SECURITY] 5 Key Design Decisions That Affect Security in Web Applications
- Next message (by thread): [WEB SECURITY] CSRF: Flash + 307 redirect = Game Over
- **Messages sorted by:** [date] [thread] [subject] [author]

A vulnerability for Ruby on Rails was recently patched [http://weblog.rubyonrails.org/2011/2/8/csrf-protection-bypass-in-ruby-on-rails].
The default CSRF prevention built into RAILS had two components: (1) a custom HTTP Header, and (2) a CSRF token in the post body. This was implemented in a way that only one of these components were required in a request, and not both. Modern browser security makes this fairly secure because JavaScript cannot create custom HTTP Headers and have them sent across domains. However, a researcher from Google found that there is a way to exploit this issue using "certain combinations of browser plugins and HTTP redirects". The new patch for Ruby on Rails forces both of these components to be in the request, preventing exploitation.

A hidden flash file on a website will automate the sending of the following request:

http://www.attacker.com/redirect.php?status=307&url=http://www.victim.com

Flash will allow the site which it is running from to specify POST data and additional headers. But before sending the request, it will check the sites crossdomain.xml file. The attacker will set up their cross domain.xml file as follows.

http://www.attacker.com/crossdomain.xml

...

http://lists.webappsec.org/pipermail/websecurity_lists.webappsec.org/2011-February/007533.html

Better Cross-Site Request Forgery Protection

1. Save a CSRF token to the session
2. Add meta-tags for JavaScript to read tokens
3. Insert the CSRF token in forms
4. Send token with XHR requests
5. Match tokens on POST/DELETE/PATCH/XHR

CSRF Meta Tags

```
<meta content="authenticity_token" name="csrf-param" />
<meta content="NvyZey4G/GuBEr6RQJTrsfX1iiptwSBWiu9XOo9h4cE=" name="csrf-token" />
```

March 2012

Mass Assignment, GitHub, and Homakov

https://blog.github.com/2012-03-04-public-key-security-vulnerability-and-mitigation/

Mass Assignment

```
@user = User.find(params[:id])
@user.update_attributes(params[:user])
```

rails / rails

Watch 2,598 Star

Code Issues 359 Pull requests 733 Projects 0 Insights

Mass assignment vulnerability - how to force dev. define attr_accesible? #5228

Closed **homakov** opened this issue on Mar 1, 2012 · 106 comments

homakov commented on Mar 1, 2012 Contributor

Those who don't know methods attr_accesible / protected - check that article out
http://enlightsolutions.com/articles/whats-new-in-edge-scoped-mass-assignment-in-rails-3-1

Let's view at typical situation - middle level rails developer builds website for customer, w/o any special protections in model(Yeah! they don't write it! I have asked few my friends - they dont!)
Next, people use this website but if any of them has an idea that developer didnt specify "attr_accesible" - hacker can just add an http field in params, e.g. we have pursue's name edition. POST request at pursues#update

id = 333 (target's pursues id)
pursue['name'] = 'my purses name'
pursue['user_id'] = 412(hacker id)

https://github.com/rails/rails/issues/5228

rails / rails

Watch 2,598 | Star 39,359 | Fork 15,906

Code | Issues 355 | Pull requests 733 | Projects 0 | Insights

wow how come I commit in master? O_o

Browse files

master | v5.2.0 ··· v4.0.0.beta1

homakov committed on Mar 4, 2012 — 1 parent 4d391a4 — commit b83965785db1eec019edf1fc272b1aa393e6dc57

Showing 1 changed file with 3 additions and 0 deletions. — Unified | Split

3 ■■■ hacked — Show comments — View

```
... ... @@ -0,0 +1,3 @@
1 +another showcase of rails apps vunlerability.
2 +Github pwned. again :(
3 +will you pay me for security audit?
```

261 comments on commit b839657

https://github.com/rails/rails/commit/b83965785db1eec019edf1fc272b1aa393e6dc57

Speculated GitHub Code

```
class PublicKeyController < ApplicationController
  before_filter :authorize_user

  def update
    @current_key = PublicKey.find_by_id params[:public_key][:id]
    @current_key.update_attributes(params[:public_key])
  end
end
```

https://gist.github.com/peternixey/1978249

More Speculation

```
<input type="text" name="public_key[title]">
```

More Speculation

```
<input type="text" name="public_key[title]">
<input type="text" name="public_key[user_id]">
```

Mass Assignment Over Time

Rails 2

Optional white/black list in models

Rails 3.1

Option to require whitelist in models

Rails 3.2.3

Whitelist is default in new apps

Rails 4

Strong Parameters

January 2013

CVE-2013-0156
YAML -> RCE

https://groups.google.com/d/msg/rubyonrails-security/61bkgvnSGTQ/nehwjA8tQ8EJ

YAML Can Serialize Arbitrary Ruby Objects

```
2.5.0 :001 > require 'yaml'
 => true
2.5.0 :002 > YAML.dump(Object.new)
 => "--- !ruby/object {}\n"
2.5.0 :003 > YAML.load(_)
 => #<Object:0x00000000222dbf0>
```

Deserialization Can Execute Code

https://github.com/charliesome/charlie.bz/blob/master/posts/rails-3.2.10-remote-code-execution.md

Put the YAML in the XML

```
<id type="yaml">
--- !ruby/object {}
</id>
```

January 2013

YAML -> RCE
Part 2

Rails Parsed JSON as YAML

CVE-2014-0130
CVE-2016-0752

RCE via Render

http://matasano.com/research/AnatomyOfRailsVuln-CVE-2014-0130.pdf
https://nvisium.com/resources/blog/2016/01/26/rails-dynamic-render-to-rce-cve-2016-0752.html

Directory Traversal

```
render "custom/#{params[:template]}"
```

Directory Traversal

```
render "custom/#{params[:template]}"
```

```
?template=../../../../Gemfile
```

Render Defaulted to ERB

```
render "blah.txt"
```

Log a Payload

```
example.com/users/?x=<%25%3D+%60ls%60+%25>
```

Log a Payload

```
example.com/users/?x=<%25%3D+%60ls%60+%25>
```

```
Started GET "/users/?x=%3C%25%3D+%60ls%60+%25%3E" for 127.0.0.1 at
2018-04-17 23:37:52 -0700

Processing by UsersController#index as HTML

  Parameters: {"x"=>"<%= `ls` %>"}

  Rendered users/index within layouts/application (0.6ms)

Completed 200 OK in 16.9ms (Views: 16.4ms | ActiveRecord: 0.0ms)
```

Render Log

```
render "custom/#{params[:template]}"
```

```
?template=../../../../log/production.log
```

Sanitizing vs. Escaping

```
sanitize "<Some <script>prompt(1)</script>> input"
=> "> input"

h "<Some <script>prompt(1)</script>> input"
=> "&lt;Some &lt;script&gt;prompt(1)&lt;/script&gt;&gt; input"
```

CVEs Related to Sanitizing Methods

- November 2009 - No CVE
- CVE-2011-2931
- CVE-2012-3465
- CVE-2013-1855
- CVE-2013-1857
- CVE-2015-7579
- CVE-2015-7580
- CVE-2018-8048
- CVE-2018-3741

Avoid!

```
sanitize()

sanitize_css()

strip_tags()

strip_links()
```

Rails Security Features!

January 2007

Rails 1.2

Rails 1.2: Parameter Log Filtering

```
class ApplicationController < ActionController::Base
  filter_parameter_logging :password, :cc_number
end
```

Parameter Log Filtering

```
Started GET "/?password=[FILTERED]" for 127.0.0.1 at 2018-04-15 11:12:36 -0700

Processing by HomeController#index as HTML

  Parameters: {"password"=>"[FILTERED]"}

  Rendered home/index.html.erb within layouts/application (0.0ms)

Completed 200 OK in 4.2ms (Views: 4.1ms | ActiveRecord: 0.0ms)
```

Rails 4: Parameter Log Filtering

```
Rails.application.config.filter_parameters += [:password]
```

Default `config/initializers/filter_parameter_logging.rb`

August 2010

Brakeman 0.0.1!

```
gem install brakeman
brakeman your/app
```

August 2010

Rails 3.0!

August 2010

Rails 3.0!

Rails 3.0: Auto-escaping HTML

Rails 2:

```
<%= h params[:query] %>
```

Rails 3:

```
<%= params[:query] %>
```

August 2011

Rails 3.1

Rails 3.1: has_secure_password

```
class User < ActiveRecord::Model
  has_secure_password
end
```

Rails 3.1: config.force_ssl

In Rails **3.1**, this just redirected requests to HTTPS

As of Rails **5.2**, this setting:

- Redirects requests to HTTPS
- Sets **`secure`** flag on all cookies
- Sends the **`Strict-Transport-Security`** header

June 2013

Rails 4.0!

Rails 4.0: Strong Parameters

Rails 3:

```
params.class => ActiveSupport::HashWithIndifferentAccess
```

Rails 4:

```
params.class => ActionController::Parameters
```

Rails 4.0: Encrypted Session Cookies

Rails 3:

Session cookie is marshalled and signed

Rails 4:

Session cookie is JSON, signed, and encrypted

Rails 4.0: Default Headers

```
config.action_dispatch.default_headers = {
  'X-Frame-Options' => 'SAMEORIGIN',
  'X-XSS-Protection' => '1; mode=block',
  'X-Content-Type-Options' => 'nosniff'
}
```

Rails 5.0: Per-form CSRF Tokens

```
Rails.configuration.action_controller.per_form_csrf_tokens = true
```

https://medium.com/rubyinside/a-deep-dive-into-csrf-protection-in-rails-19fa0a42c0ef

Rails 5.1: Encrypted Secrets

```
rails secrets:setup

rails secrets:edit
```

https://www.engineyard.com/blog/encrypted-rails-secrets-on-rails-5.1

Rails 5.2: Encrypted Credentials

```
rails credentials:edit
```

https://www.engineyard.com/blog/rails-encrypted-credentials-on-rails-5.2

Rails 5.2: Default Headers

```
config.action_dispatch.default_headers = {
  'X-Frame-Options' => 'SAMEORIGIN',
  'X-XSS-Protection' => '1; mode=block',
  'X-Content-Type-Options' => 'nosniff',
  'X-Download-Options' => 'noopen',
  'X-Permitted-Cross-Domain-Policies' => 'none',
  'Referrer-Policy' => 'strict-origin-when-cross-origin'
}
```

Rails 5.2: Content-Security-Policy

```ruby
# config/initializers/content_security_policy.rb
Rails.application.config.content_security_policy do |policy|
  policy.default_src :self, :https
  policy.font_src    :self, :https, :data
  policy.img_src     :self, :https, :data
  policy.object_src  :none
  policy.script_src  :self, :https
  policy.style_src   :self, :https

  # Specify URI for violation reports
  policy.report_uri "/csp-violation-report-endpoint"
end
```

Content Security Policy

https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy

More Information

http://guides.rubyonrails.org/security.html

Thank You

@presidentbeef	presidentbeef.com
@brakeman	brakeman.org
@brakemanpro	brakemanpro.com