Securing your rails app.

@bob_p
$ git clone https://github.com/bob-p/owasp-workshop.git
$ cd owasp-workshop
$ bundle install
$ bundle exec rake db:migrate
$ bundle exec rails server
How Secure?
SQL injection
HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TRouble.

OH, DEAR - DID HE BREAK SOMETHING? IN A WAY-

DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?

OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.

AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

http://xkcd.com/327/
User.where("email = '#{params[:email]}'").first
User.first(:conditions => "email = '#{params[:email]}'")

SELECT "users".* FROM "users" WHERE (email = '' OR 1='1') LIMIT 1
User.find_by_email(params[:email])

User.where("email = ?", params[:email]).first

User.first(:conditions => ["email = ?", params[:email]])
Sanitise all SQL input
XSS

<script>alert('h4x0r3d');</script>
<script>alert('h4x0r3d');</script>


<iframe src="http://hacker.com/hack"/></iframe>
Secure your cookies

cookies(:secure_cookie, :httponly => true, :secure => true)
Escape output

<3

html_escape("<script></script>")
SafeBuffer

> 3

“hello”.html_safe?
SafeBuffer

raw("<h1>hello</h1>")
Secure your cookies

Ensure user submitted input is sanitised
Session management
Rails.application.config.session_store :cookie_store
Rails.application.config.session_store :cache_store
Rails.application.config.session_store :active_record_store
config.

```ruby
secret_token = '
3783262ab68df94a79ab02edca8a1a9c3.....'
```

```
rake secret
```
XSS
Insecure networks
Rails.application.config.force_ssl = true
Allow logout
MyApp::Application.config.session_store :cookie_store, :key => '_my_key', :expire_after => 45.minutes

class User < ActiveRecord::Base
  devise :authenticatable, :timeoutable, :timeout_in => 45.minutes
end
reset_session
No concurrent logins
Account
lockout
Password complexity
Destroy session on logout

def logout
  reset_session
end
Hash passwords

class User
  def password=(password)
    self.encrypted_password = ::BCrypt::Engine.hash_secret(password, self.salt)
  end
end
Use bcrypt

http://codahale.com/how-to-safely-store-a-password/
No large objects
No critical data
Summary

SSL

Hash data

Clear sessions
Mass Assignment

“public_key” => {“user_id” => 4223}
def signup
    @user = User.create(params[:user])
end

params[:user] = {:username => "pwn3d", :admin => true}
class User
  attr_protected :admin
end
class User
  attr_accessible :email
end

config.active_record.whitelist_attributes = true
class User
  attr_accessible :role,
    :as => :admin
end
User.create({:username => 'Bob', :role => "admin"}, :as => :admin)
def signup
    @user = User.create(user_params)
end

def user_params
    params[:user].slice(:email)
end
https://github.com/rails/strong_parameters
Summary

attr_accessible

Slice pattern

attr_protected
Direct object reference

/users/:id/posts/:post_id
def create
    @user = User.find(params[:user_id])
    @note = Note.create(:user => @user, :text => params[:text])
end
def create
    @user = User.find(session[:user_id])
    @note = @user.notes.create(:text => params[:text])
end
def show
  @note = Note.find(params[:id])
end
def show
    @user = User.find(session[:user_id])
    @note = @user.notes.find(params[:id])
end
def show
  @user = User.find(session[:user_id])
  @note = Note.find(params[:id])
  if @note.editable_by?(@user)
    # Do things
  end
end
Find users from session

Use scoping methods
CSRF

<img src="http://demo.com/notes/1/destroy"/>
<img src="http://example.com/notes/1/destroy" />
GET

Safe requests / queries

POST

PUT

DELETE

Changes resource / orders
class ApplicationController
  protect_from_forgery
end
Correct http verbs

Rails CSRF protection
Redirection & file uploads
def login
  login_business
  redirect_to params[:from]
end
def login
  login_business
  redirect_to session[:from]
end
Sanitise file names
“../../../etc/passwd”
def cleanup_filename(filename)
    filename.gsub(/&*$+,\/:;=\?><\[]\{\}\[\]~\%\#\)/, '_')
end
Sanitise file type
class User
  validates_attachment :avatar, :presence => true,
    :content_type => { :content_type => "image/jpg" },
    :size => { :in => 0..10.kilobytes }
end
Process asynchronously
No redirect locations in params
Sanitise file name/type
Process files a-sync
Rails.application.config.force_ssl = true
ssl_ciphers  HIGH:!aNULL:!MD5;
ssl_protocols  SSLv3  TLSv1;
Use SSL!
CSRF
XSS
Whitelist.contains?
(request.remote_ip)
Summary

XSS / CSRF / Injection

Restrict access
Info
leakage
server_tokens off;
RuntimeError in MainController#index

Rails.root: /Users/thomaspomfret/Projects/waw12/cardboard-hipster

Application Trace  Framework Trace  Full Trace

app/controllers/main_controller.rb:4:in `index'
lib/mint_authorization.rb:11:in `ensure_validate_rights_was_called'
lib/hostname_support.rb:9:in `call'

Request

Parameters:

None

Show session dump
Show env dump

Response

Headers:

None
Don’t give away anything
Server-side
User privileges
config.filter_parameters += [:password]
Summary

Restrict user permissions

Obscure sensitive data
Workshop
* Ruby (1.9)
* Rubygem
* Bundler
* Git
github.com/bob-p/owasp-workshop
$ git clone https://github.com/bob-p/owasp-workshop.git
$ cd owasp-workshop
$ bundle install
$ bundle exec rake db:migrate
$ bundle exec rails server
http://localhost:3000
SQL injection
XSS
Session management
Mass Assignment

“public_key” => {“user_id” => 4223}
Direct object reference

/users/:id/posts/:post_id
CSRF
Redirection
SSL
Admin
brakeman
gem 'brakeman' (in ./Gemfile)
$ bundle install
$ brakeman
Summary
Resources
www.rorsecurity.info
Thanks!

@bob_p

http://mintdigital.com