Rails & Security

People should know it

Insecure-by-default means insecure

http://homakov.blogspot.com

Agenda

- GET Accessible Actions(method “match”, CSRF)
- Mass Assignment(attr_accessible, “SQL Inject”)
- JS(ON) and DOM Injects, Responders and XSS
- Regular Expressions and Validators
- Common Tips
- Headers
- [bonus?] OAuth

Rails ARE Secure

- CSRF Protection by default (authenticity_token)
- XSS Protection(HtmlSafe, sanitize by default)
- SQL Injects are impossible(active record)
- Hundreds of commits with security improvements, etc

PHP(and others) is not

- if I see PHP site with (proper)CSRF protection than .. it's facebook.com
- SQL Injects, XSS, includes, zomg etc
- "secure by default" just impossible

thus rails is more secure than most php sites are...

BUT

case 1

#routes.rb
#match usage is a common mistake
match “/follow”, to: “followings#create”
match “/followers, to: “followings#index”

case 1

Hey, “match” means GET too. GET means no csrf protection!

```
def verified_request?
  !protect_against_forgery? || request.get? ||
    form_authenticity_token == params[request_forgery_protection_token] ||
    form_authenticity_token == request.headers['X-CSRF-Token']
end
```

case 1

>This commit disallows calling +match+ without an HTTP verb constraint by default. To explicitly match all verbs, this commit also adds a :via => :all option to +match+. (@wycats)

#update code:
post “/follow”, to: “followings#create”
get “/followers, to: “followings#index”

match “/getpost_endpoint”, via: :all, to: “etc#etc”

case 1 tips

Make sure to set “post” for state-changing requests.

Avoid using of “match”

Use “get” for all data retrieval requests.

Scope your routes, be RESTful, please.

case 2

```
#comments/index.haml
:javascript
  var comments = #{@comments.to_json}

OR

:javascript
  var value = "#{current_user.name}"
```

case 2

@comments = {k:"</script><script>alert(1)</script>"}

JSON Encoder and ':javascript' (:css too!) both don't escape anything - output is RAW.

case 2

XSS?!

case 2 tips

Update rails to 4(now html entities are escaped by default) or set manually ActiveSupport.escape_html_entities_in_json = true

in initializers or don't use .to_json in templates.

case 3

```
#comments/index.haml
:javascript
  var data = #{@data.to_json} #or getJSON
  $('.datacontainer').html(data.body);
```

case 3

Pitfall. That is a pure DOM XSS - you didn't sanitize it! Escaping \u only helps JSON parser but you should sanitize it before you insert into DOM

Don't trust/use any input param until you sanitized it.

case 3

```
urn!0;this.url=$("#search_form").attr("action"),this.query=$("#q").val(),
(this.query=""),this.init_params(),this.init_pagination(),this.init_filte
ation(),this.init_price(),this.init_order(),this.model_init(),this.init_c
this.params[a]||null},set:function(a,b){this.params[a]=b},init_params:fun
initial_search_filters)this.set_param_from_given(a,initial_search_filters
c=b;switch(a){case"podrazdeli":c=b.split("|")}search_obj.set(a,c)},init_r
{if(this.reload_page)return!1;this.query!=""&&$("#q_search").html("
<div>"+search_obj.add_deselect_button("query")+this.query+"</div>").show(
```

case 3 tips

Use $.text()/innerText instead of $.html() /innerHTML when possible, always sanitize any user input even in JS(Rails just escapes). I strongly recommend this patch:

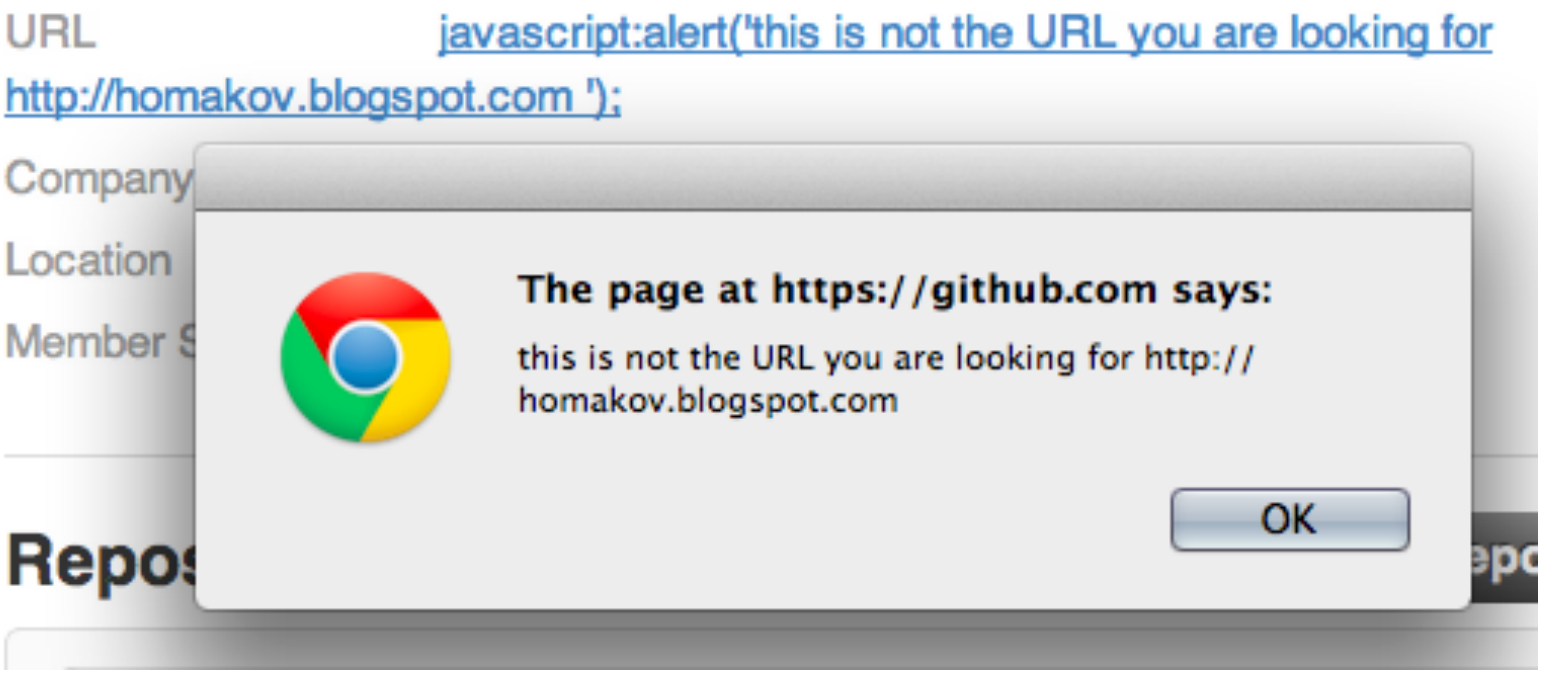
ActiveSupport::JSON::Encoding:: ESCAPED_CHARS.merge! '<' => '<'

case 4

```
params[:user][:url]="http://#{params[:user][:url]}" unless params[:user][:url] =~ /^https?/

#update attributes
```

case 4


URL
javascript:alert('this is not the URL you are looking for http://homakov.blogspot.com ');
Company
Location
Member S
Repos
The page at https://github.com says:
this is not the URL you are looking for http://homakov.blogspot.com
OK

case 4 tips

Keep in mind - in ruby $^ always match new lines. Your manuals and books lie. Use \A\z
This passes:

javascript:alert(1)/*
http://hi.com
*/
added warning/exception in RoR

case 5

```
#in application_controller.rb
skip_before_filter :verify_authenticity_token
```

case 5 tips

protect_from_forgery is a MUST. It is a hassle to deal with tokens **but don't be stupid**.

No, presence of authenticity_token input doesn't scare a hacker.

case 6

found an XSS for auto_link, remember, always *whitelist* everything - protocols too

javascript://%0Aalert(1)

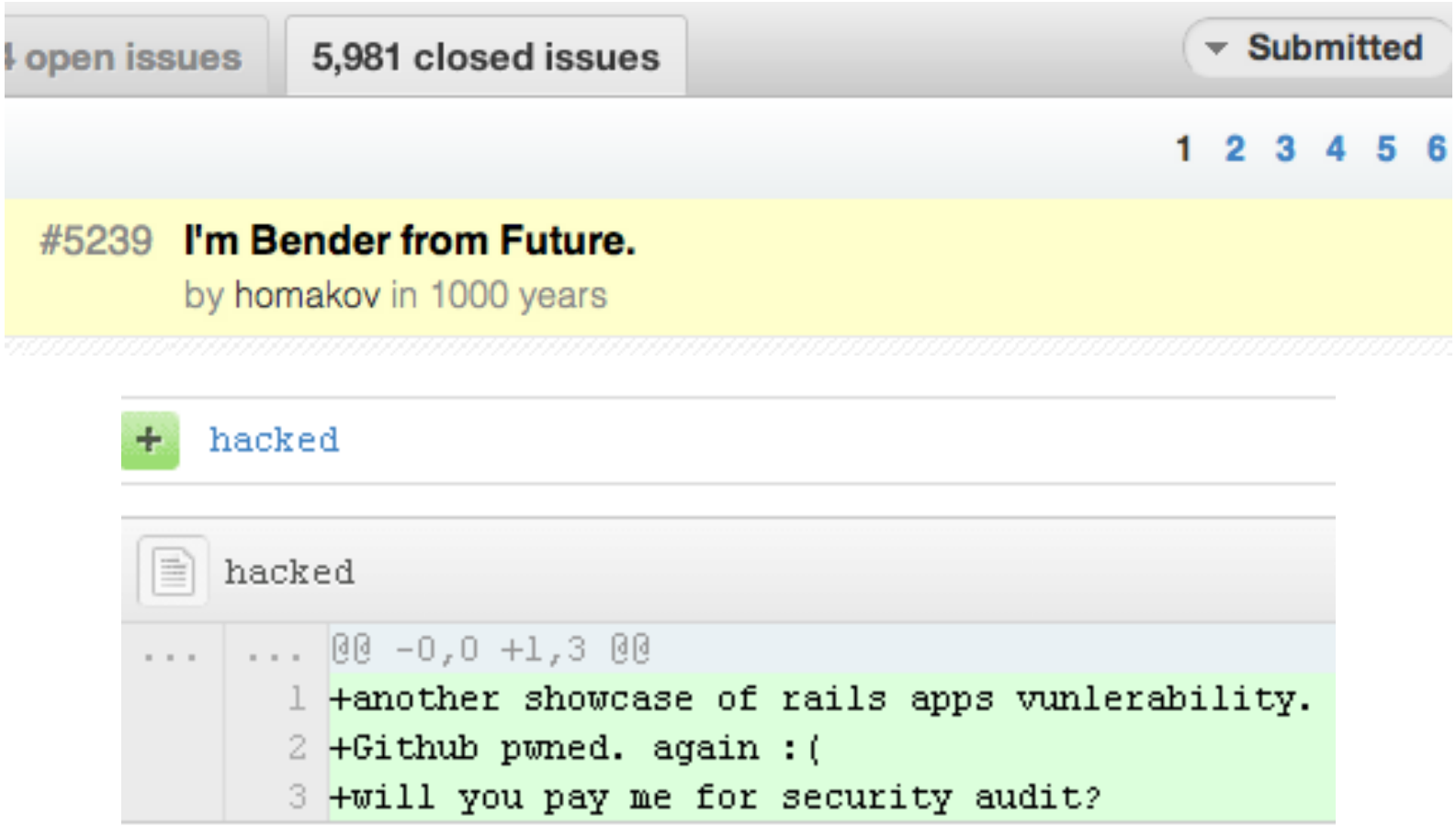
Update your bundle, if you use auto_link or rails_autolink gem


BREAKING
NEWS

case 7

```
class PublicKey < ActiveRecord::Base
  #attr_accessible, where are you...
end
```

case 7


open issues
5,981 closed issues
Submitted
1 2 3 4 5 6
#5239 I'm Bender from Future.
by homakov in 1000 years
+ hacked
hacked
@@ -0,0 +1,3 @@
+another showcase of rails apps vunlerability.
+Github pwned. again :(
+will you pay me for security audit?

case 7

Github and Assembla shared the same vulnerability.
It was easy to steal or push code into anybody’s repo 'dropping' your public key.

Also you could(still can) set “created/updated_at” to 3012 in *really* a lot of applications to have fun and get the 1st place in 'order by *_at'

case 7 tips

If use update_attributes/new/create+hash - you should set attr_accessible(If you don't use mass assignment - don't care.)

gem 'strong_parameters'

whitelist_attributes = true by default.

it takes slightly more time to write an app but it's worth it.

IT IS NOT attr_accessor :±

case 8

```
#hand-made jsonp
json = Order.all.to_json
render text: "#{params[:callback]}(#{json})"
```

https://api.github.com/user/repos?
callback=leak

case 8 tips

don't give out private data via JSONP

avoid - render text: contains_user_input

XSS - ?callback=<script>..</script>
use - render json: data, callback: params[:cb]

case 9 - CVE-2012-2660

Mass assignment[extended edition]. You can send nested arrays/hashes in any param.
params[:token] can be a huge array(brute):

?token[]=1&token[]=2&token[]=3...

it also may contain nils!
?token[] <- nil

case 9 - CVE-2012-2660

Change

User.find_by_token(params[:token]) and

User.where(token: params[:token])

use explicit casting

params[:token].to_s

common tips

- use system('ls', '.') instead of `ls .`
- before_filter{headers['X-Frame-Options'] ='SAMEORIGIN'}#application_controller.rb
- hide config/initializers/secret_token.rb
- obvious: check permissions
- WHITELIST
- RTFM

#DISCUSS

Security is not developers' business.

Web is poorly designed: Clickjacking, CSRF

bonus

bonus OAuth

CSRF + GET.

code/token

getting into master-account with no fingerprints.

omniauth fb strategy vulnerability

depends on server side logic

bonus OAuth

http://soundcloud.
com/connect/facebook/create?
**code=AQBXeR_dORPIx4RRUt_YzJ6Rdg0
eb9CWHek8J2fB4vqfdNPvznmx-d-
J36gGQlXJICRdfqFb9a_VWqke4ZamE2H
ytlXtK5c6sMaOQUQLPPhSWNv3v8z-
ze6hdT6x4LNSXC_-
jxGRecjw1WTmifzO_rBFaDl86xPo2YH3k_
ehEtw5wM9rVduymjZumXkoistF7l9g2MQ**

bonus OAuth

Mitigation: CSRF token in 'state' param.

Checking $_SESSION['state']==$_REQUEST['session'] IS NOT WORKING

Check existence and equality both.

OR use client side JS based authentication.

references

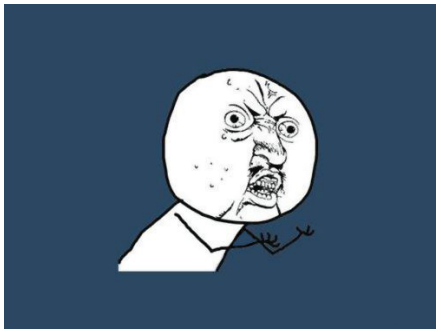
[old] http://www.rorsecurity.info/

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

http://developers.facebook.com/docs/authentication/server-side/

get new stuff 1st!: homakov.blogspot.com

Teh Edn.



Y U NO PAY ME FOR SECURITY AUDIT?