GuardRails

A (Nearly) Painless Solution to Web Application Security

Jonathan Burket, Patrick Mutchler, Michael Weaver, Muzzammil Zaveri

University of Virginia
http://guardrails.cs.virginia.edu
So What?

Web applications are easy to create!
Web security is not easy!

wahoo
Annotated Ruby on Rails Code

Attach Policies to Data
Fine grained taint-tracking

GuardRails

Nearly effortless

Secure Ruby on Rails Code

Improve Readability

Reduce implementation errors
Access Control Policies

Patrick Mutchler
An Example

Project A

Project B

Project C

Project D

User’s Project Memberships

A
B
C

RedMine
Access Control is Annoying and Tedious

```ruby
if include_subprojects && !active_children.empty?
  ids = [id] + active_children.collect { |c| c.id }

  conditions = ['#{Project.table_name}.id IN (#{ids.join(',', ')')})']
```

RedMine
Access Control is Annoying and Tedious

```ruby
if include_subprojects && !active_children.empty?
  ids = [id] + active_children.collect { |c| c.id }

  conditions = [
    "#{Project.table_name}.id IN (#{ids.join(',')})"
  ]
```

RedMine
if include_subprojects && !active_children.empty?
  ids = [id] + active_children.collect { |c| c.id }

  conditions = [
    "#{Project.table_name}.id IN (#{ids.join(',')})" AND #{Project.visible_by}"
  ]
Object Oriented Policy Enforcement

GuardRails Proxy

GuardRails Alias

Protected Field

Protected Object
Access Control Policies

- **Read**: `User.find(..)`
- **Write**: `@usr.secret = x`
- **Append**: `addToMailingList(@usr)`
- **Create**: `User.create`
- **Delete**: `@usr.destroy`
Access Control Policy Annotations

# @ (policy_class, targets, expression)

# @ delete access, :admin

# @ write access, password, lambda{|user|user.id == self.id }

# @ read access, lambda{|user|
    self.is_public or user.memberships.include? self.id
}

13
Policy Violations

Forbidden
You don't have permission to access /cgi-bin on this server.
Apache/2.0.52 (ASPLinux) Server at 192.168.0.106 Port 80
def require_user
    unless current_user
        self.notice = I18n.t("page_only_viewable_when_logged_in")
        redirect_to new_user_session_url
        return false
    end
end

def require_no_user
    if current_user
        self.notice = I18n.t("page_only_viewable_when_logged_out")
        redirect_to root_url
        return false
    end
end

def store_location
    # disallow return to login, logout, signup pages
    disallowed_urls = [signup_url, login_url, logout_url]
    ..
Dynamic Taint Tracking

Jonathan Burket

xkcd.com
Dynamic Taint Tracking

Protects against injection attacks

SQL Injection:

“SELECT profile FROM users WHERE username="" + user_name + ""

Good: user_name = “jazzFan26”
Bad: user_name = “"; DROP TABLE users--”

Cross-Site Scripting:

“User: <a href=‘profile_page’>” + user_name + “</a>”

Good: user_name = “DrKevinPhillips”
Bad: user_name = “<script language=’javascript’>
alert(‘document.cookie’);</script>”
Expressive Taint Status

“<a href="profile?id=184392"><h1>SoccerFan1985</h1></a>”

String

Value:

“<a href="profile?id=184392"><h1>SoccerFan1985</h1></a>”

Taint:

<table>
<thead>
<tr>
<th>Character Index</th>
<th>HTML Taint</th>
<th>SQL Taint</th>
<th>Ruby Eval Taint</th>
<th>Untainted</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Untainted</td>
</tr>
<tr>
<td>51</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>{:default=&gt;NoHTML}</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Untainted</td>
</tr>
</tbody>
</table>
Taint Propagation

```
"foo" + "bar" \rightarrow "foobaz"
```
Taint Lattice

- Combination of taint statuses
- Extendable by developers
- Subtyping
Context-Appropriate Sanitization

# @ taint, username, AlphaNumeric

# @ taint, full_name, NoHTML,
TitleTag: LettersAndSpaces

# @ taint, profile, BoldItalicUnderline,
"//script[@language='javascript']": Invisible
Group

Group name: UVAResearch

Welcome to University of Virginia
Demo and Early Results

Michael Weaver
Other real-word examples
Looking to Try Out GuardRails?

• Our Contact Info: guardrails@cs.virginia.edu
• WebPage: http://guardrails.cs.virginia.edu
Questions?
Access Control Policies

Context-Dependent Policies

Privileged Functions