Языки Интернет-программирования

Лекция 13. Cookies/Sessions. Проблемы безопасности

- Хранение данных браузером и сервером
- Cookies
- Sessions
- Авторизация
- Безопасность RoR-приложений

МГТУ им. Н.Э. Баумана, доц. каф. ИУ-6,
к.т.н. Самарев Роман Станиславович

samarev@acm.org

Cookies

27.11.2018

- RFC 2965 HTTP State Management Mechanism
- 300 cookies всего
- 20 cookies на один вебсервер
- по 4 Кбайт данных на один cookie
- navigator.cookieEnabled

- Код на Javascript:

```
var nextyear = new Date();
    nextyear.setFullYear(nextyear.getFullYear() + 1);
    document.cookie = "version=" +
                                document.lastModified +
                                "; expires=" + nextyear.toGMTString();
```

Другие способы клиентского хранения

27.11.2018

- userData в IE
- SharedObject подключаемого Flash-модуля
- **HTML5 Web Storage**
  http://www.w3schools.com/html/html5_webstorage.asp

Sessions

27.11.2018

- Механизм для идентификации пользователей
- хэш функция (обычно от текущего времени)
  Пример – MD6, SHA-2, ГОСТ Р 34.11-2012,...

- Способы хранения сессий
  - ActionDispatch::Session::CookieStore – Хранение на клиенте.
  - ActiveRecord::SessionStore – Хранение в БД через Active Record.
  - ActionDispatch::Session::CacheStore – Хранение в кэше Rails.
  - ActionDispatch::Session::MemCacheStore – Хранение в кластере memcached.

- Файл конфигурации config/initializers/session_store.rb:

```
YourApp::Application.config.session_store :cookie_store,
        :key => '_your_app_session', :domain => ".example.com“
```

- Секретный ключ приложения config/initializers/secret_token.rb

```
YourApp::Application.config.secret_token = '49d3f3de9ed86c74b94ad6bd0...'
```

- http://guides.rubyonrails.org/action_controller_overview.html

Некоторые термины

27.11.2018

- **Идентификация** – сопоставление субъекта по идентификатору
- **Аутентификация** - процедура проверки подлинности (например по логину и паролю)
- **Авторизация** - предоставление определённому лицу или группе лиц прав на выполнение определённых действий

Аутентификация
HTTP Basic Authentication

27.11.2018

```
class AdminController < ApplicationController
  http_basic_authenticate_with :name => "humbaba",
                               :password => "5baa61e4"
end
```

- Пароль передаётся и хранится в не кодированном виде

- http://guides.rubyonrails.org/action_controller_overview.html#http-authentications
- http://api.rubyonrails.org/classes/ActionController/HttpAuthentication/Basic.html

Аутентификация HTTP Digest Authentication

27.11.2018

```
require 'digest/md5'
class AdminController < ApplicationController
  REALM = "SuperSecret"
  USERS = {
    "dhh" => "secret", # plain text password
    "dap" => Digest::MD5.hexdigest(["dap",REALM,"secret"].join(":"))}  #ha1 digest

  before_filter :authenticate

  def view
  end

  private
  def authenticate
    authenticate_or_request_with_http_digest(REALM) do |username|
      USERS[username]
    end
  end
end
```

- http://edgeapi.rubyonrails.org/classes/ActionController/HttpAuthentication/Digest.html

Аутентификация
Собственная реализация

27.11.2018

- Расширяем модель для хранения пользователей

```
class User < ActiveRecord::Base
  …
  def has_password?(submitted_password)
    encrypted_password == encrypt(submitted_password)
  end

  def self.authenticate(email, submitted_password)
    user = find_by_email(email)
    return nil  if user.nil?
    return user if user.has_password?(submitted_password)
  end

  private
    …
end
```

Проверка авторизации с помощью сессий

27.11.2018

- Добавляем метод определение текущего пользователя
- **:current_user_id** – ключ ассоциативного массива сессии
- **session[:current_user_id]** – объект, связанный с сессией

```
class ApplicationController < ActionController::Base

  private
  # Возвращает объект User с ID, сохраненный в session с ключом
  # :current_user_id Это стандартный способ для Rails
  # Процедура входа устанавливает значение сессии
  # Процедура выхода сбрасывает значение сессии
  def current_user
    @_current_user ||= session[:current_user_id] &&
      User.find_by_id(session[:current_user_id])
  end
end
```

Проверка авторизации с помощью сессий

27.11.2018

```
class SessionsController < ApplicationController
  # выдать форму ввода логина/пароля
 def new
  end

  # "Создать" – войти под своим именем
  def create
    if user = User.authenticate(params[:username], params[:password])

      # Сохраняем идентификатор пользователя ID в сессию
      # для последующего использования
      session[:current_user_id] = user.id
      redirect_to root_url
    end
  end
end
```

Проверка авторизации с помощью сессий

27.11.2018

```
class SessionsController < ApplicationController
  # "Удалить" – отключить пользователя (убрать из сессии)
  def destroy
    # Удалить id пользователя из session
    @_current_user = session[:current_user_id] = nil
    redirect_to root_url
  end
end
```

- http://railstutorial.ru/chapters/4_0/sign-in-sign-out#top

Проверка авторизации
Фильтр действия

27.11.2018

```
class UsersController < ApplicationController


  before_action :authenticate, except: [:new, :create]


  def index
    @users = User.all
    @user_name = current_user ?
          current_user.name : "unknown"
  end
end
```

Использование сторонних средств авторизации

27.11.2018

- Библиотеки для Rails:
  Authlogic, Devise, Clearance, CanCan

- Технологии распределенной авторизации
  - OpenId
  - OpenSSO
  - LDAP
  - x509
  - OAuth

Authlogic
Модули аутентификации

27.11.2018

- Для встраивания аутентификации в Rails
- http://rubygems.org/gems/authlogic
  - gem install authlogic
- http://www.rubydoc.info/gems/authlogic/3.6.1#Quick_Rails_example
- http://railscasts.com/episodes/160-authlogic

Authlogic "add ons"

27.11.2018

- Authlogic OpenID addon: http://github.com/binarylogic/authlogic_openid
- Authlogic LDAP addon: http://github.com/binarylogic/authlogic_ldap
- Authlogic Facebook Connect: http://github.com/kalasjocke/authlogic_facebook_connect
- Authlogic Facebook Connect (New JS API): http://github.com/studybyte/authlogic_facebook_connect
- Authlogic Facebook Shim http://github.com/james2m/authlogic_facebook_shim
- Authlogic OAuth (Twitter): http://github.com/jrallison/authlogic_oauth
- Authlogic Oauth and OpenID: http://github.com/viatropos/authlogic-connect
- Authlogic PAM: http://github.com/nbudin/authlogic_pam
- Authlogic x509: http://github.com/auth-scc/authlogic_x509

Authlogic
Специальный тип модели

27.11.2018

```
class UserSession < Authlogic::Session::Base
  # specify configuration here, such as:
  # logout_on_timeout true
  # ...many more options in the documentation
end
```

Authlogic
User Migration

27.11.2018

```
create_table :users do |t|
    t.timestamps
    t.string :login, :null => false
    t.string :crypted_password, :null => false
    t.string :password_salt, :null => false
    t.string :persistence_token, :null => false
    t.integer :login_count, :default => 0, :null => false
    t.datetime :last_request_at
    t.datetime :last_login_at
    t.datetime :current_login_at
    t.string :last_login_ip
    t.string :current_login_ip
end

add_index :users, :login
add_index :users, :persistence_token
add_index :users, :last_request_at
```

Authlogic
Создание сессий

27.11.2018

- UserSession.create(:login => "bjohnson",
  :password => "my password", :remember_me => true)

- session = UserSession.new(:login => "bjohnson",
  :password => "my password", :remember_me => true);
  session.save

- UserSession.create(:openid_identifier => "identifier",
  :remember_me => true)
  # requires the authlogic-oid "add on" gem

- UserSession.create(my_user_object, true)
  # skip authentication and log the user in directly, the true means "remember me"

Authlogic
Требование аутентификации

27.11.2018

```
class User < ActiveRecord::Base
  acts_as_authentic do |c|
    c.my_config_option = my_value
  end # the configuration block is optional
end
```

Authlogic
Хранение сессий

27.11.2018

```
class ApplicationController
  helper_method :current_user_session, :current_user

  private
    def current_user_session
      return @current_user_session if defined?(@current_user_session)
      @current_user_session = UserSession.find
    end

    def current_user
      return @current_user if defined?(@current_user)
      @current_user = current_user_session && current_user_session.user
    end
end
```

- https://github.com/binarylogic/authlogic

Authlogic
Контроллер для сессий

27.11.2018

```
class UserSessionsController < ApplicationController
  def new
    @user_session = UserSession.new
  end

  def create
    @user_session = UserSession.new(params[:user_session])
    if @user_session.save
      redirect_to account_url
    else
      render :action => :new
    end
  end

  def destroy
    current_user_session.destroy
    redirect_to new_user_session_url
  end
end
```

Devise
Модули аутентификации

27.11.2018

- Для встраивания аутентификации в Rails

- http://rubygems.org/gems/devise
  - gem install 'devise'

- https://github.com/plataformatec/devise

Devise
Встраивание в приложение

27.11.2018

- Добавить в Gemfile строку gem 'devise'
- Сгенерировать основные структуры **rails generate devise:install**
- Сгенерировать модель для хранения пользователей (MODEL – любое имя) **rails generate devise MODEL**

Devise
Код для контроллеров

27.11.2018

- Требование аутентификации before_action :authenticate_user!

- Проверка аутентификации user_signed_in?

Безопасность WEB-приложений

27.11.2018

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

Сессии

27.11.2018

- Идентификатор - Хэш (MD6, SHA-2, ...)
- Может быть перехвачен в сети и подделан

- Защита
  - использование SSL (https://letsencrypt.org/)
  - config.force_ssl = true
  - Периодическое обновление

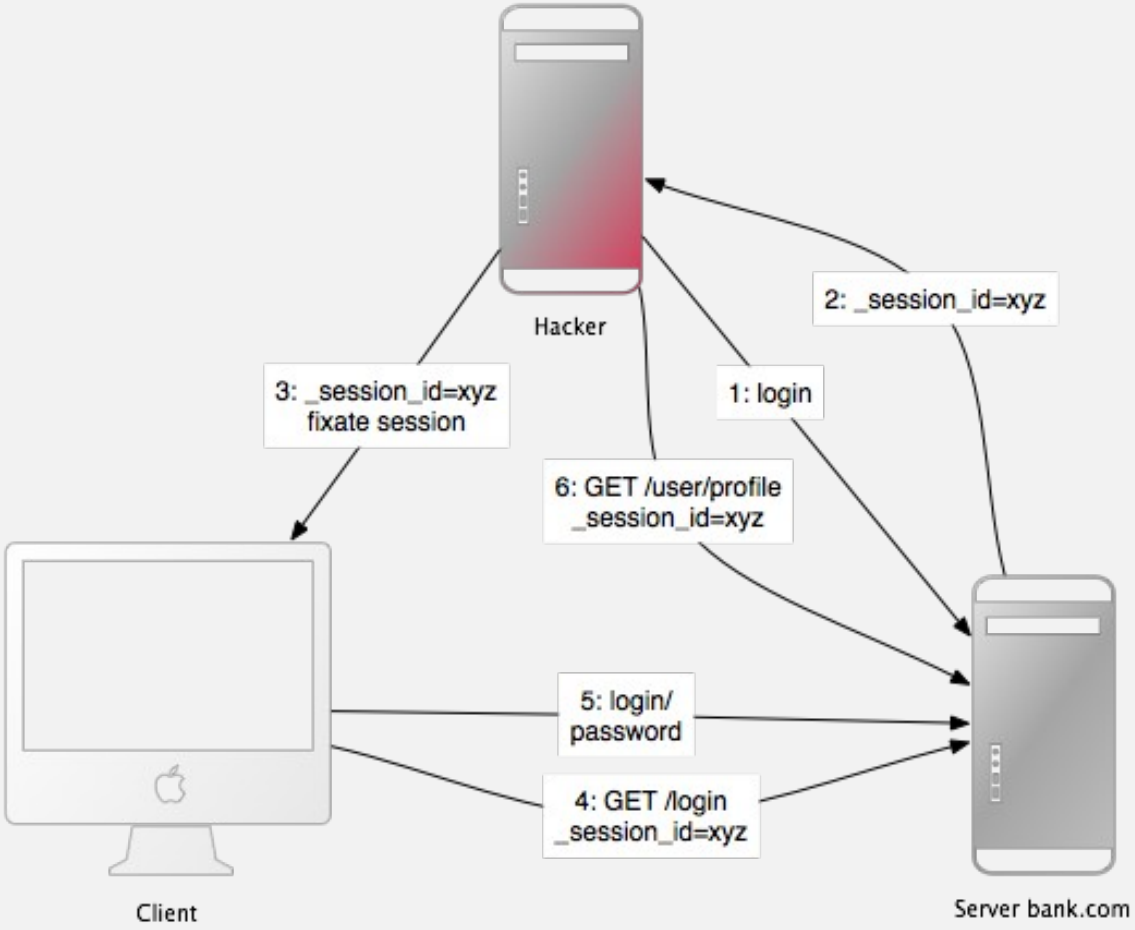
Сессии

27.11.2018

- Не следует хранить критически важные данные, которые могут быть потеряны
- Размер данных ограничен cookies

- Replay Attacks for CookieStore Sessions
  - Сессия хранит размер кредита
  - Пользователь снимает копию сессии
  - Пользователь совершает покупку, кредит уменьшается
  - Пользователь повторяет покупку с копией сессии

Session Fixation



- Решение проблемы: SessionsController#create
  - **reset_session**

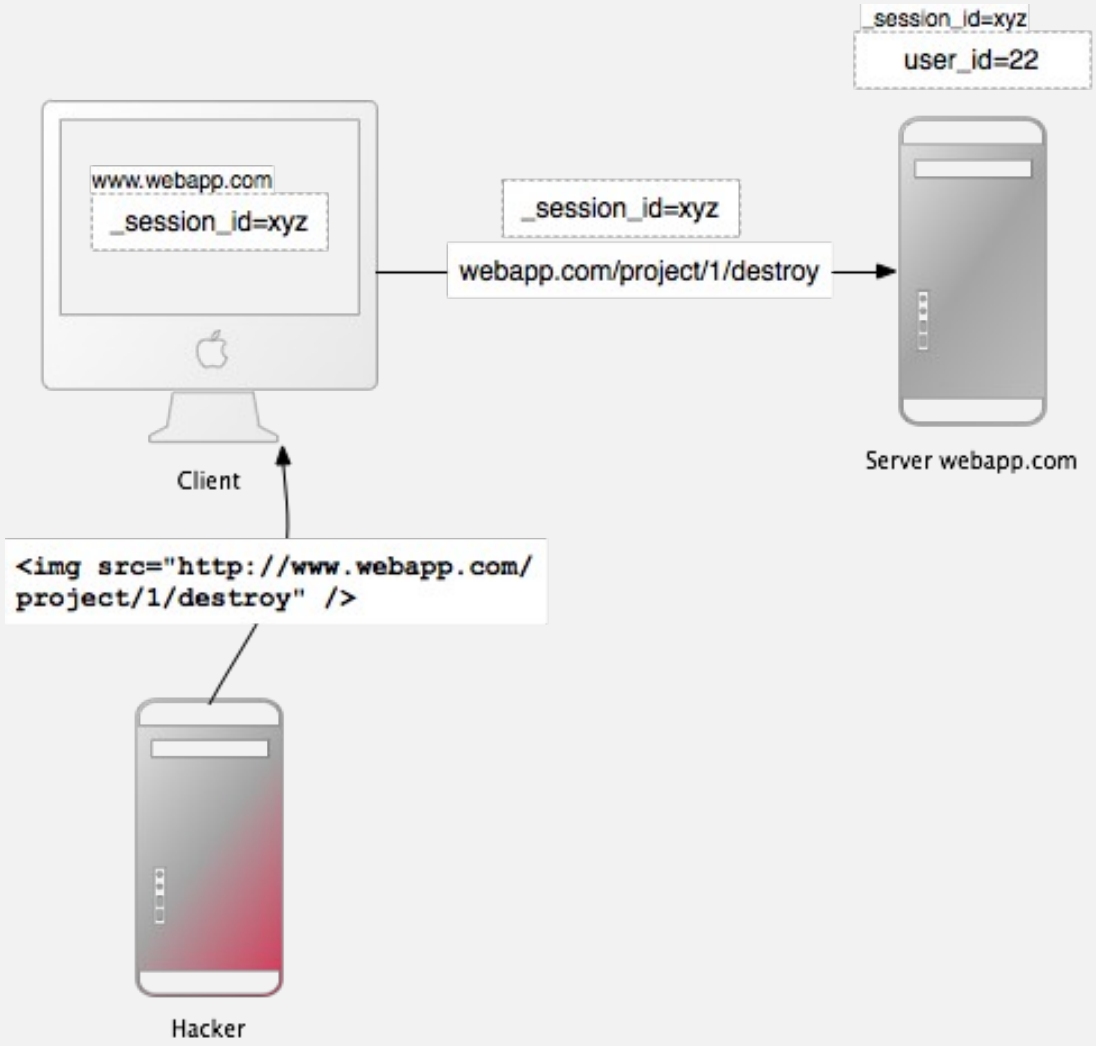
Ограничение времени жизни сессии

27.11.2018

```
class Session < ActiveRecord::Base
  def self.sweep(time = 1.hour)
    if time.is_a?(String)
      time = time.split.inject { |count, unit|
    count.to_i.send(unit) }
    end

    delete_all "updated_at < '#{time.ago.to_s(:db)}'"
  end
end
```

Cross-Site Request Forgery (CSRF)



- http://guides.rubyonrails.org/security.html#session-fixation

Решение проблем CSRF

27.11.2018

- Корректное использование GET/POST
  - GET
    - Если взаимодействие похоже на запрос
  - POST
    - Если взаимодействие похоже на приказ
    - Изменяется состояние ресурса

- Cross-Origin Resource Sharing (CORS) - https://developer.mozilla.org/ru/docs/Web/HTTP/CORS

- Rails. Дополнительный ключ шифрования
  - **protect_from_forgery** :secret => "12345678901234567890123456789O…"
  - В формы добавляется поле **authenticity_token**, содержащее ключ
  - https://nvisium.com/resources/blog/2014/09/10/understanding-protectfromforgery.html

Примеры разметки с подделкой запроса

27.11.2018

```
<a href="http://www.harmless.com/" onclick="
  var f = document.createElement('form');
  f.style.display = 'none';
  this.parentNode.appendChild(f);
  f.method = 'POST';
  f.action = 'http://www.example.com/account/destroy';
  f.submit();
  return false;">To the harmless survey</a>

<img src="http://www.harmless.com/img" width="400"
    height="400" onmouse=…>
```

Перенаправление

27.11.2018

- Rails-action

```
def legacy
  redirect_to(params.update(:action=>'main'))
end
```

- Строка злоумышленника:

http://www.example.com/site/legacy?
param1=xy&param2=23&host=**www.attacker.com**

- Необходимо следить за передаваемыми параметрами!

Отправка файлов

27.11.2018

- Необходимо следить за именами файлов

```
def sanitize_filename(filename)
  filename.strip.tap do |name|
    # NOTE: File.basename не воспринимает Windows-пути на Unix
    # необходимо фильтровать только имя, но не путь!
    name.sub! /\A.*(\\|\/)/, ''
    # Все не алфавитно-цифровое заменяем на подчеркивание
    name.gsub! /[^\w\.\-]/, '_'
  end
end
```

Получение файлов

27.11.2018

- Контроль параметров!!!

send_file('/var/www/uploads/' + params[:filename])

Пример filename='../../../etc/passwd'

- Очистка имени файла

```
# имя директории для хранения вычисляем по текущему rb-файлу
basename = File.expand_path(File.join(File.dirname(__FILE__), '../../files'))
# приклеиваем имя, полученное от пользователя
filename = File.expand_path(File.join(basename, @file.public_filename))

# указываем относительный путь и проверяем совпадение директорий
raise if basename != File.expand_path(File.join(File.dirname(filename), '../../../'))
send_file filename, :disposition => 'inline'
```

Групповое присваивание

27.11.2018

- Пример из старых версий Rails

http://www.example.com/user/signup?user[name]=ow3ned&**user[admin]=1**

```
def signup
  params[:user] # => {:name => "ow3ned", :admin => true}
  @user = User.new(params[:user])
end
```

Групповое присваивание в Rails ≥ 4

27.11.2018

- Обязательное наличие в списке **permit**

```
# model
class User < ActiveRecord::Base
  #  attr_accessible :username # атрибуты Rails ≤3!
end

# controller
def create
  # User.create!(params[:user]) # Rails ≤3
  User.create!(params.require(:user).permit(:username)) # Rails ≥4!
end
```

- http://blog.sensible.io/2013/08/17/strong-parameters-by-example.html

Взлом учетных записей

27.11.2018

- Пароли
  - отслеживать утечки паролей, /etc/passwd,…
  - затруднять подбор паролей

  E-Mail
  - не активировать JS и не переходить по ссылкам в подозрительных письмах
  - не сохранять сессии и пароли в браузере
  - использовать раздельные браузеры для разных целей

- CSRF and XSS
  - разрешать только ограниченным доменам
  - использовать CORS

Работа с паролями

27.11.2018

- Пароли не хранить в отрытом виде
- Использовать hash + salt

hash_md5("12345678") → 25d55ad283aa400af464c76d713c07ad

salt="180320"
hash_md5("12345678" + salt) → 1fc89e9b60e91498a241ace9e6b0da7a

- Использовать многофакторную аутентификацию (пароль + SMS/почта/….)
- Менять пароли по регламенту

Защита от перебора и спама

- CAPTCHA

- reCAPTCHA - http://www.google.com/recaptcha

- reCAPTCHA for Rails
  http://github.com/ambethia/recaptcha

Взлом через анализ журналов операций

27.11.2018

- config.filter_parameters << :password

Проблемы регулярных выражений

27.11.2018

```
class File < ActiveRecord::Base
  validates :name, :format => /^[\w\.\-\+]+$/
end
```

- Одна строка злоумышленника: "file.txt%0A<script>alert('hello')</script>"
- Будет преобразована в две строки "file.txt**\n**<script>alert('hello')</script>"….
- /^[\w\.\-\+]+$/ только для первой строки!!!
- Исправление ошибки фильтра **/\A[\w\.\-\+]+\z/**

Повышение привилегий

27.11.2018

- Опасный код

```
@project = Project.find(params[:id])
# получим всё, независимо от прав доступа
```

- Безопасный код

```
@project =
    @current_user.projects.find(params[:id])
# получим только то, что доступно
    пользователю
```

Типы инъекций

- SQL
- HTML
- Javascript
- Cross-Site Scripting (XSS)
- CSS Injection
- Textile Injection
- Command Line Injection
- Header Injection

Инъекции

27.11.2018

- before_action :only => […]
  вместо *:except => […].*
- **attr_accessible** вместо **attr_protected (permit для Rails 4)**

- Не исправлять некорректные строки пользователя!
  - "**<sc**<script>**ript>**".gsub("<script>", "")

Литература

27.11.2018

- http://www.tutorialspoint.com/ruby-on-rails/rails-session-cookies.htm
- http://guides.rubyonrails.org/action_controller_overview.html#session
- http://rusrails.ru/action-controller-overview/
- http://railstutorial.ru/chapters/4_0/sign-in-sign-out#top
- http://rubygems.org/gems/authlogic
- http://guides.rubyonrails.org/security.html