Foreman - Bug #6746

ovirt VM's power and console buttons are disabled for non-admin users even when the required permissions are granted

07/23/2014 03:17 AM - Neil Miao

Status:	Duplicate		
Priority:	Normal		
Assignee:			
Category:			
Target version:			
Difficulty:		**Fixed in Releases:**	
Triaged:		**Found in Releases:**	
Bugzilla link:		**Red Hat JIRA:**	
Pull request:			

Description

When a ovrit-backed host page is loaded, the "power on/off" and "console" buttons always greys out for non-admin users, doesn't matter if the right host/managed and compute resource permissions are granted to the user.

code trace:

(./app/views/compute_resources_vms/_details.html.erb)

```
<%= content_tag(:div, *vm_power_actions*(@vm) + vm_console(@vm) , :id=>'power_actions') %>
```

(./app/helpers/compute_resources_vms_helper.rb)

```
def vm_power_actions(vm)
    button_group(
      if vm
         html_opts = vm.ready? ? {:confirm => _('Are you sure?'), :class => "btn btn-danger"} : {
:class => "btn btn-success"}
         *link_to_if_authorized* _("Power%s") % state(vm.ready?), hash_for_power_host_path(
:power_action => vm.ready? ? :stop : :start).merge(:auth_object => vm, :permission =>
'power_hosts'),
...

def link_to_if_authorized(name, options = {}, html_options = {})
    enable_link = *authorized_for*(options)
...

def authorized_for(options)
    action          = options.delete(:auth_action) || options[:action]
    object          = options.delete(:auth_object)
    user            = User.current
    controller      = options[:controller] || params[:controller]
    controller_name = controller.to_s.gsub(/::/, "_").underscore
    id              = options[:id]
    permission      = options.delete(:permission) || [action, controller_name].join('_')

    if object.nil?
      user.allowed_to?({ :controller => controller_name, :action => action, :id => id }) rescue
false
    else
      authorizer = options.delete(:authorizer) || Authorizer.new(user)
      authorizer.*can?*(permission, object) *+rescue false+*
    end
...
```

(./app/services/authorizer.rb)

```
  def can?(permission, subject = nil)
    if subject.nil?
      user.permissions.where(:name => permission).present?
    else
      return true if user.admin?
      collection = @cache[subject.class.to_s][permission] ||= *find_collection*(subject.class, 
:permission => permission)
      collection.include?(subject)
    end
  end

  def find_collection(resource_class, options = {})
    permission = options.delete :permission

    base = user.filters.joins(:permissions).where(["#{Permission.table_name}.resource_type = ?", *
resource_name*(resource_class)])
    all_filters = permission.nil? ? base : base.where(["#{Permission.table_name}.name = ?", 
permission])

    organization_ids = allowed_organizations(resource_class)
    location_ids     = allowed_locations(resource_class)

    organizations, locations, values = taxonomy_conditions(organization_ids, location_ids)
    all_filters = all_filters.joins(taxonomy_join).where(["#{TaxableTaxonomy.table_name}
.id IS NULL " +
                                                              "OR (#{organizations}) " +
                                                              "OR (#{locations})",
                                                          *values]).uniq

    all_filters = all_filters.all 
# load all records, so #empty? does not call extra COUNT(*) query
    return resource_class.where('1=0') if all_filters.empty?

    unless @base_collection.nil?
      if @base_collection.empty?
        return resource_class.where('1=0')
      else
        resource_class = resource_class.where(:id => base_ids)
      end
    end

    return resource_class.scoped if all_filters.any?(&:unlimited?)

    search_string = build_scoped_search_condition(all_filters.select(&:limited?))
    resource_class.search_for(search_string)
  end

  def resource_name(klass)
    return 'Operatingsystem' if klass <= Operatingsystem
    return 'ComputeResource' if klass <= ComputeResource

    case name = klass.to_s
      when 'Audited::Adapters::ActiveRecord::Audit'
        'Audit'
      when /\AHost::.*\Z/
        'Host'
      else
        name
      end
  end
```

The problem begins when **vm_power_actions** passes the **vm** object to **link_to_if_authorized** helper function. The vm object's class name is eventually used in Authorizer's **find_collection** as parameter **resource_class**. In function find_collection, **resource_name** is used to generate the resource_type string from the class name. It seems the code is expecting that vm.class is a child of

ComputeResource, however, a ovirt vm's class is actually **Fog::Compute::Ovirt::Server** and its **NOT** a child of ComputeResource. As a result, resource_name(Fog::Compute::Ovirt::Server) returns 'Fog::Compute::Ovirt::Server' which subsequently crashes find_collection.

(I had to comment out the highlighted '*rescue false*' in function **authorized_for** so I can actually see this error in the log)

```
Failed to fetch vm information: undefined method `where' for Fog::Compute::Ovirt::Server:Class
Original exception backtrace:
/usr/share/foreman/app/services/authorizer.rb:38:in `find_collection'
/usr/share/foreman/app/services/authorizer.rb:17:in `can?'
/usr/share/foreman/app/helpers/application_helper.rb:140:in `authorized_for'
/usr/share/foreman/app/helpers/application_helper.rb:153:in `link_to_if_authorized'
/usr/share/foreman/app/helpers/compute_resources_vms_helper.rb:7:in `vm_power_actions'
```

With the 'rescue false' in place, function authorized_for return false all the time as long as the user is not admin.

Related issues:

Is duplicate of Foreman - Bug #5994: Power and Console links are disabled for...	**Closed**	**05/30/2014**

History

#1 - 07/23/2014 03:55 AM - Dominic Cleal

- Is duplicate of Bug #5994: Power and Console links are disabled for non-admin users added

#2 - 07/23/2014 03:56 AM - Dominic Cleal

- Status changed from New to Duplicate

Thanks for the report, but I think this has been fixed via #5994 ready for 1.5.2.