ADMIN WEB PAGE FOR IMAP ACLS AND SHARED FOLDERS INCOMING EMAILS

Document created on 2014-01-16
Nicolas Bondier

[pdf][doc][htm]

Contents

Introduction

This document presents the source code of the administration web page for IMAP ACLs and for routing email to shared folders and subfolders.

It describes the architecture, the configuration of the Debian servers and the source code the web page.

Prerequisites

For this how-to, we require the installation of a Dovecot and Postfix server as described in the precedent document presented here : http://switzernet.com/3/public/131212-shared-imap-dovecot/index.pdf. However, it is not required to have a Ceph cluster and shared block device for continuing.

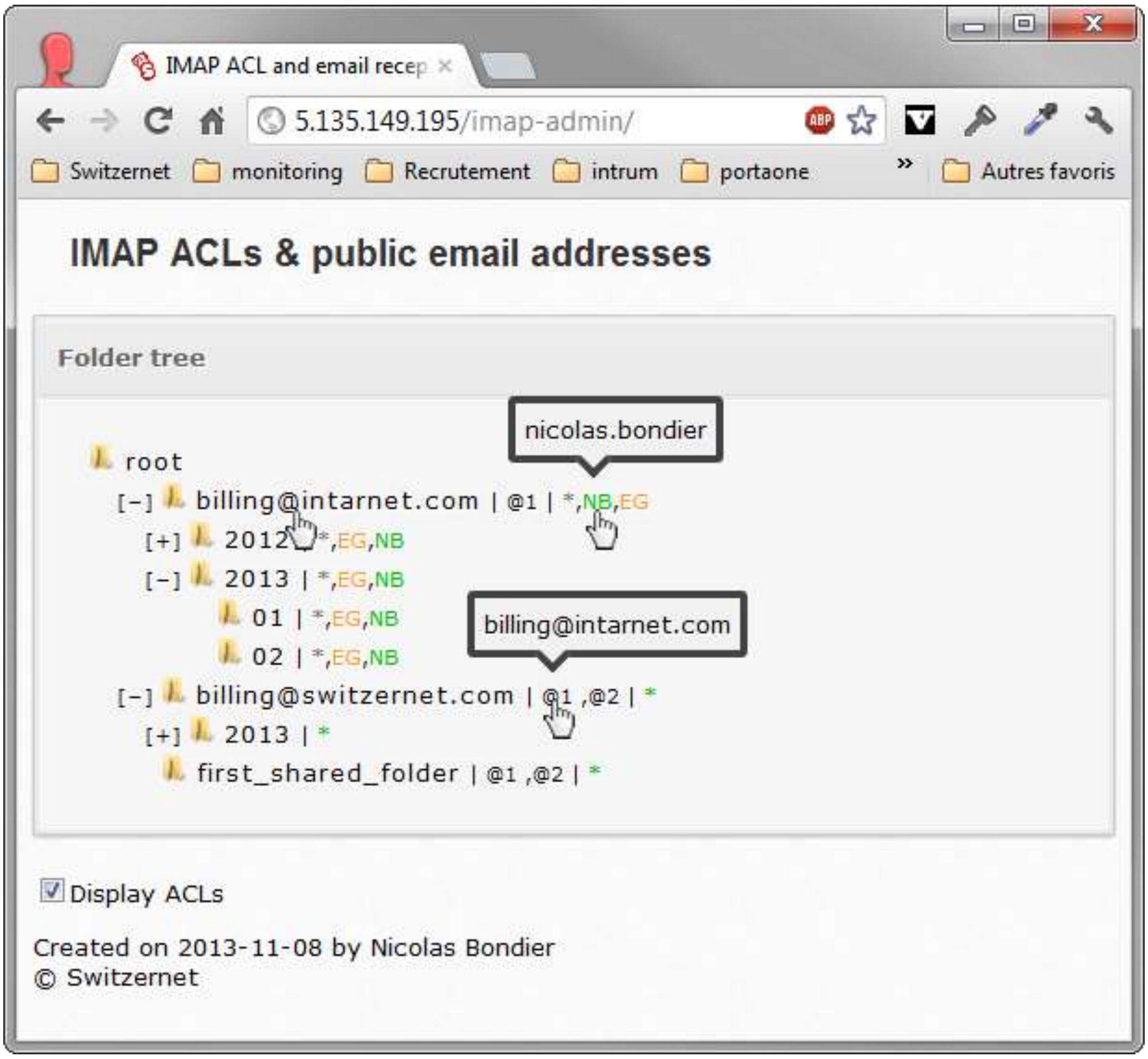
Presentation

In order to have a full control on IMAP ACLs and each public folder incoming mail address, we decide to create a web interface for that purpose.

The page is protected with http authentication. This authentication permit only on group of the LDAP users to login.

The index page display the shared folders tree. On each folder, a summary of the settings is displayed:

- User initials with a color meaning the permission set.
- Email address on which the folder receive emails.



IMAP ACLs & public email addresses

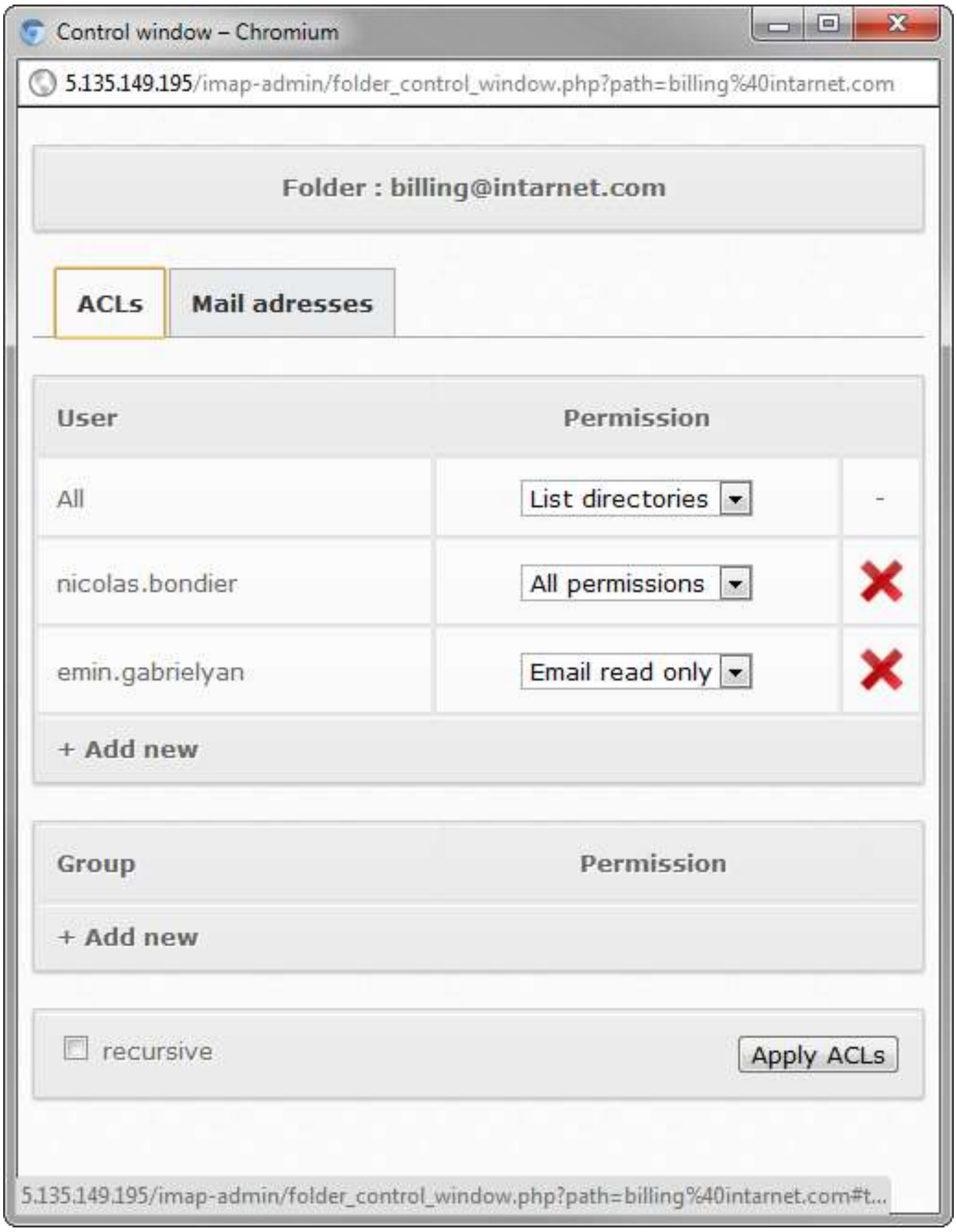
Folder tree

```
root
  [-] billing@intarnet.com | @1 | *,NB,EG
    [+] 2012 | *,EG,NB
    [-] 2013 | *,EG,NB
          01 | *,EG,NB
          02 | *,EG,NB
  [-] billing@switzernet.com | @1 ,@2 | *
    [+] 2013 | *
      first_shared_folder | @1 ,@2 | *
```

☑ Display ACLs

Created on 2013-11-08 by Nicolas Bondier
© Switzernet

On clicking on any of the folders, the settings window is opened to view and edit information. There are two tabs, one for the ACLs and one for the mail addresses.


Control window – Chromium

5.135.149.195/imap-admin/folder_control_window.php?path=billing%40intarnet.com

Folder : billing@intarnet.com

ACLs | Mail adresses

User	Permission	
All	List directories	-
nicolas.bondier	All permissions	✗
emin.gabrielyan	Email read only	✗

+ Add new

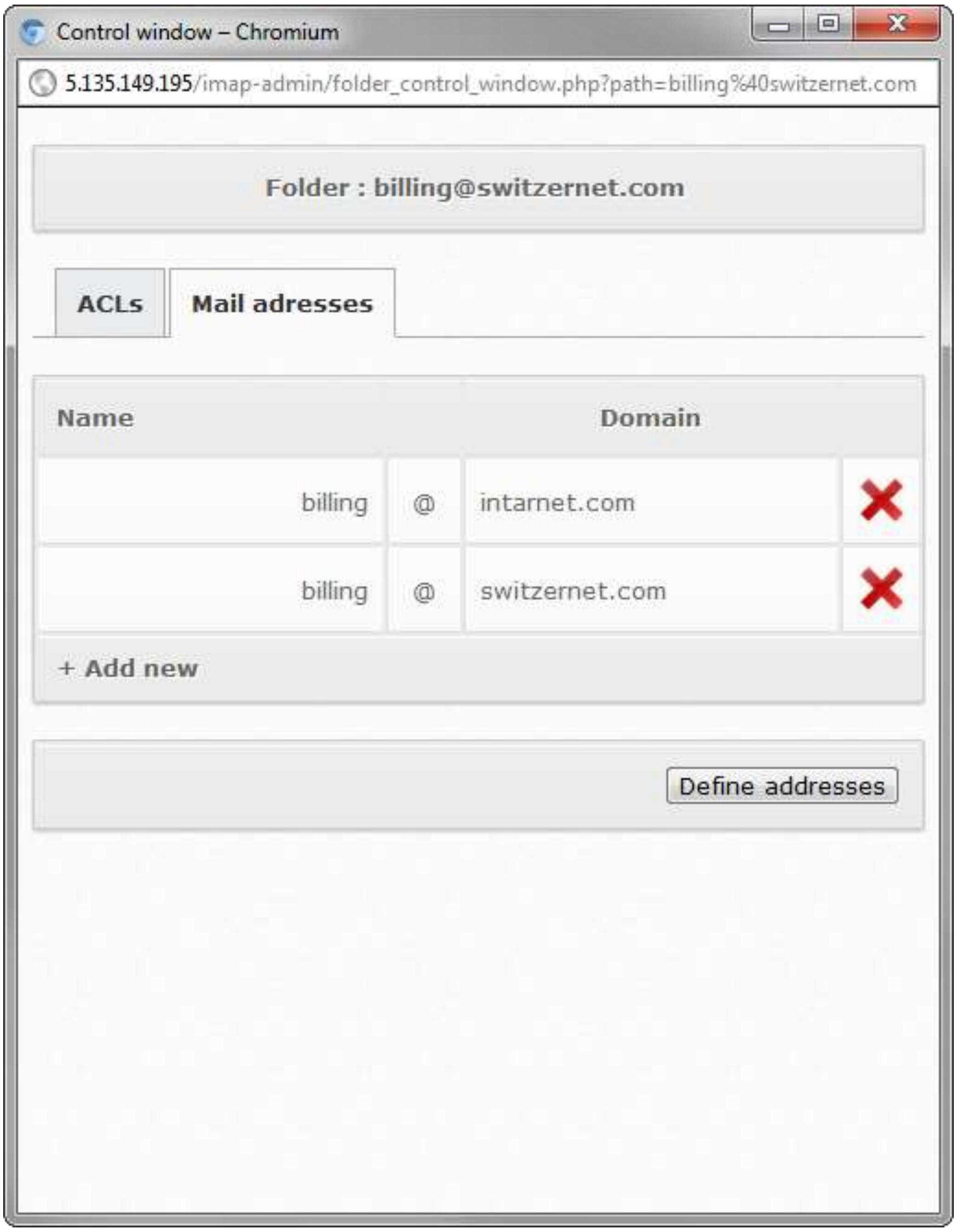
Group	Permission

+ Add new

☐ recursive

Apply ACLs

5.135.149.195/imap-admin/folder_control_window.php?path=billing%40intarnet.com#t...


Control window – Chromium

5.135.149.195/imap-admin/folder_control_window.php?path=billing%40switzernet.com

Folder : billing@switzernet.com

ACLs | Mail adresses

Name		Domain	
billing	@	intarnet.com	✗
billing	@	switzernet.com	✗

+ Add new

Define addresses

Configuration of apache server

Installation of the web server

Install apache and libraries used for the web pages:

```
root@ceph-client-2:~# aptitude update; aptitude install apache2 libapache2-
mod-php5 php5-sqlite
```

We use a special path for the IMAP administration web pages, not under the traditional "/var/www" folder. The files are located under "`/mnt/switzernet_rbd/131205-mail-server-www/131108-imap-admin`".

Basic authentication with PAM

For authenticating to the IMAP admin page, we use a simple basic authentication. But instead of using simple "`.htpasswd`" files, we use PAM authentication with the user of the LDAP server.

```
root@ceph-client-2:~# aptitude install libapache2-mod-authnz-external
pwauth
root@ceph-client-2:~# aptitude install libapache2-mod-authz-unixgroup
```

We edit the config file for our host. It is "`/etc/apache2/sites-available/default`".

We add the following clause:

```
<IfModule mod_authnz_external.c>
AddExternalAuth pwauth /usr/sbin/pwauth
SetExternalAuthMethod pwauth pipe
AddExternalGroup unixgroup /usr/sbin/unixgroup
SetExternalGroupMethod unixgroup environment
</IfModule>
```

In the virtual root directory clause, we add the following lines:

```
<Directory / >
AuthType Basic
AuthName "Restricted Area"
AuthBasicProvider external
AuthExternal pwauth
GroupExternal unixgroup
Require group imap-administrator
Options FollowSymLinks +Indexes
AllowOverride None
</Directory>
```

The "`imap-administrator`" group is set in the LDAP directory. It groups the users that will be able to access the IMAP administration web page.

For activating the group authentication, we first download the last version of "`pwauth`" in order to extract the "`unixgroup`" script. Last version number is available at https://code.google.com/p/pwauth/.

```
wget "http://pwauth.googlecode.com/files/pwauth-2.3.11.tar.gz"
tar xzvf ./pwauth-2.3.11.tar.gz
cp pwauth-2.3.11/unixgroup /usr/sbin/
chmod 775 /usr/sbin/unixgroup
```

Enable the new modules:

```
root@ceph-client-2:~# a2enmod authnz_external authz_unixgroup
```

Restart apache:

```
root@ceph-client-2:~# service apache2 restart
Restarting web server: apache2 ... waiting .
root@ceph-client-2:~#
```

Sudo for www-data user

Our PHP functions use some UNIX commands in order to retrieve and modify files on the system. We need to enable "`sudo`" access to apache. We added the following line in "`/etc/sudoers`":

```
# User alias specification
www-data ALL=(ALL) NOPASSWD:ALL
```

Code documentation

Files tree

```
root@ceph-client-1:/mnt/switzernet rbd/131205-mail-server-www# tree 131108-imap-admin/
131108-imap-admin/
|-- favicon.png
|-- folder control window.php
|-- include
|   |-- functions.php
|   |-- get acl short list.php
|   |-- get adr short list.php
|   |-- get ids and paths to reload.php
|   |-- get select group list.php
|   |-- get select user list.php
|   |-- get subfolders.php
|   `-- global variables.php
|-- index.php
|-- js
|   |-- jquery-1.10.2.min.js
|   `-- jquery.blockUI.js
|-- styles
|   |-- images
|   |   |-- background.jpg
|   |   |-- del.png
|   |   |-- folder.png
|   |   `-- switzernet-logo.png
|   |-- style.css
|   `-- table.css
`-- virtual.sqlite
```

index.php

```
<?php


session_start();

include "include/functions.php";
include "include/global_variables.php";
verify_user();

?>
```

Starting script and including external functions and global variables.

We verify the user.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>IMAP ACL and email reception administration</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <link rel="icon" type="image/png" href="favicon.png" />
    <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
    <script type="text/javascript" src="js/jquery.blockUI.js"></script>
```

Document type definition and headers.

We use jQuery libraries.

```
    $(window).    <script type="text/javascript">
    $(window).bind(
      "beforeunload", function() {
        if( childWindow && !childWindow.closed ){
          childWindow.close();
        }
    });
    </script>
```

This is a jQuery function to check if the child window is closed when main window is closing. “childWindow” is called on folder click.

```
    <link rel="stylesheet" href="styles/style.css" type="text/css" />
    <link rel="stylesheet" href="styles/table.css" type="text/css" />
  </head>
```

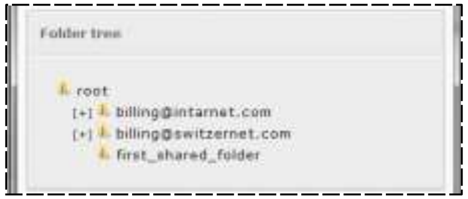
Links to the style sheets.

Closing the head.

```
  <body>
    <div class="content">
      <h2>IMAP ACLs and public email addresses</h2>
      <table style="margin-left:auto; margin-right:auto; width : 100%;">
        <tr>
          <th>
            Folder tree
          </th>
        </tr>
        <tr>
          <td>
            <ul id="root">
              <li class="folder-no-perms"> <img src="styles/images/folder.png" alt="folder" height="16"
width="16"/> root </li>
                <ul>
                <?php directory_display(''); ?>
                </ul>
            </ul>
          </td>
        </tr>
      </table>
      <script type="text/javascript">
```

Table containing the folder tree.

PHP function “directory_display” build on level of the tree. Argument is empty because this is the base folder.



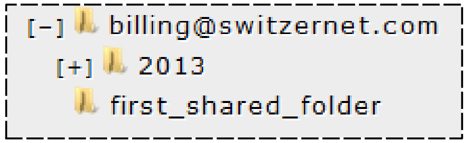
```
Folder tree

root
  [+] billing@intarnet.com
  [+] billing@switzernet.com
      first_shared_folder
```

Beginning of the JavaScript functions.

```
        function open_close_tree(id,path,e){
          if (e.innerHTML == '[+]'){
            e.innerHTML = "[-]";
            var content = get_html_content("include/get_subfolders.php?path="+path);
            $("#"+id).after(content);
          } else {
            e.innerHTML = "[+]";
            if ( $("#"+id).next("li").has("ul").length > 0 ){
              $("#"+id).next("li").remove();
            }
          }
        }
```

This function is called when clicking on the “[+]” or “[-]”.

It detect if the folder is open or not. If yes, we have to close the folder, meaning removing the subfolders from the page.
If not, the “include/get_subfolders.php” is called with the path in the name. This PHP page return the HTML for the subfolders and is inserted after.



```
[-] billing@switzernet.com
  [+] 2013
      first_shared_folder
```

```
        function urldecode(str) {
           return decodeURIComponent((str+'').replace(/\+/g, '%20'));
        }
```

A simple application for converting back URL to normal character chain.

```
function get_html_content( url )
{
  var IE_buster = '';
  // This is for forcing Internet Explorer to update !
  IE_buster = new Date().getTime();
  var result = null;
  var scriptUrl = url+"&buster="+IE_buster;
  $.ajax({
    url: scriptUrl,
    type: 'get',
    dataType: 'html',
    async: false,
    success: function(data) {
        result = data;
    }
  });
  return result;
}
```

This function loads and return the html content of any page passed as argument.

There is a special adaptatuion for Internet Explorer which keep the content in cache if we do not change the URL.

```
function open_control_window(url){
  childWindow = window.open(url,'acl_control_window',config='height=600, width=500, toolbar=no,
menubar=no, scrollbars=no, resizable=no, location=no, directories=no, status=no');
}
```

Function for opening a new floating window or load a page always in the same floating window. Defining a size permit to not open the window in a tab.

```
function load_acls_list_to_span(id,path,recursive){
  if (recursive == 'true'){
    var content2 = get_html_content("include/get_ids_and_paths_to_reload.php?path="+path);
    var arr = $.parseJSON(content2);
    t = '';
    for (var k in arr){
      if (typeof arr[k] !== 'function') {
        var content3 = get_html_content("include/get_acl_short_list.php?path="+arr[k]);
          if( document.getElementById("span_acls_"+k) ){
            span = document.getElementById('span_acls_'+k);
            span.innerHTML = content3;
            if ( $("#acl_displaying").prop('checked') ){
              $("#span_acls_"+k).show();
            } else {
              $("#span_acls_"+k).hide();
            }
          }
          t = t + "\nKey is " + k + ",\n value is " + arr[k];
      }
    }
  } else {
    var content = get_html_content("include/get_acl_short_list.php?path="+path);
    span = document.getElementById(id);
    span.innerHTML = content;
```

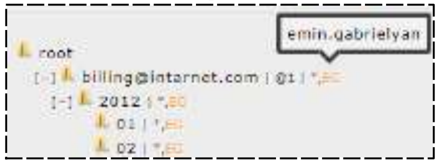
This function permit to update the short information about permissions on main page.

- “id” argument is the id of the html span to reload.
- “path” is the folder path for reading the ACL files.
- “recursive” is the argument for knowing if all subfolders permissions should be refreshed too.

If the uptade must be processed recursively, “get_ids_and_paths_to_reload.php” list all the subfolder tree and return a JSON array with all span ID to reload.

```
          }
        }
```

After a recursive applying, all subfolders are updated:



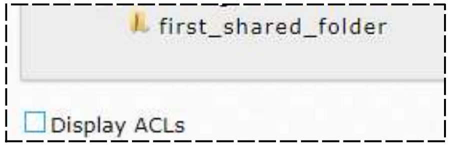
```
        function load_mail_list_to_span(id,path){
          var content = get_html_content("include/get_adr_short_list.php?path="+path);
          span = document.getElementById(id);
          span.innerHTML = content;
        }
```

Same, but it update only one folder with email address:

```
        function display_short_info(cb) {
          if ( $(cb).prop('checked') ){
            $(".acls_short_list").show();
          } else {
            $(".acls_short_list").hide();
          }
        }
```

Displaying or hiding the ACLs and receiving emails addresses information with CSS.

Hide:



```
      </script>
```

```
      <label>
        <input id="acl_displaying" type='checkbox' checked='checked'
onclick='handleClick(this);'>Display ACLs
      </label>
```

The button for toggle on or off the displaying of the informations on the page. It launch the "display_short_info" function.

```
      <div>
        <p>Created on 2013-11-08 by Nicolas Bondier <br/> © Switzernet<br/></p>
      </div>
    </div>
  </body>
</html>
```

Foot div and closing the HTML document.

folder_control_window.php

```php
<?php

session_start();

include_once "include/global_variables.php";
include_once "include/functions.php";
verify_user();
```

Start of the PHP page.
Session starting. For getting all server variables if needed.

Includes to global variables and common PHP functions.
Function to verify the user.

```php
$path = '';
if (isset($_GET['path'])){
  $path = $_GET['path'];
  $path = preg_replace("/^\//", '', $path);
}
```

Formatting the path set as parameter in the URL

```php

$update_parent_window_acls = false;
$update_parent_window_addr  = false;
$recursion = false;
$active_tab = 0;

if( isset( $_POST['recursive_acls'] ) && $_POST['recursive_acls'] == 'true' ){
  $recursion = true;
}
```

Before loading the content of the page, we check if a form was submitted in order to process an update.

There are two types of updates, ACLs and receiving email address.

For the ACLs update, we need:
- Groups and/or users array with permissions for each.

```
if ( ( isset( $_POST['users_acls'] ) || isset( $_POST['groups_acls'] ) ) && isset($_POST['file_path']) 
){
  if ( $recursion == true ){
    recursively_replace_acls( $_POST );
  } else {
    replace_acls( $_POST );
  }
  $update_parent_window_acls = true;
  $active_tab = 0;
}


if ( isset( $_POST['adr'] ) && isset($_POST['file_path']) ){
   // print_r($_POST['adr']);

  save_incoming_adresses($_POST['adr'],$_POST['file_path']);
  $update_parent_window_addr = true;
  $active_tab = 1;
}
```

- The path of the folder to modify.
- If the modification is recursive or not.

We launch the "replace_acls" or "recursively_replace_acls" functions depending if the recursive option is set.

For the receiving email addresses, we need:
- The array of email addresses.
- The folder path, which is the folder identifier in the database.

We launch the "save_incoming_adresses" function for saving the mail addresses.

```
$span_id_to_update = id_from_path(format_path($path));

$available_perms = unserialize(PERMS_GUI);

$perms_imap = unserialize(PERMS_GUI_TO_IMAP);

$perms_imap = array_flip($perms_imap);

$permissions = get_folder_perms( $path );

$available_domains = unserialize(AVAILABLE_DOMAINS);

$folder_name = format_folder_name($path);
```

Variables that will be needed latter.

Some variables are originate from the "global_variables.php" file.

We get the current folder permissions in "$permissions" with the "get_folder_perms".

Var "$span_id_to_update" is calculated the same way as in the "index.php" page (with the "directory_display" function) for easely retireve it and replace the content.

```
?>
```

End of the first PHP part.

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Control window</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <link rel="icon" type="image/png" href="favicon.png" />
    <script type="text/javascript" src="js/jquery-1.10.2.min.js"></script>
    <script type="text/javascript" src="js/jquery.blockUI.js"></script>
    <script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>
    <link href="styles/style.css" rel="stylesheet" type="text/css" media="screen" />
```

Document type definition and headers.

We use jQuery libraries.

```
    <link rel="stylesheet" href="styles/table.css" type="text/css" />
    <script>
    $(function() {
      $( "#tabs" ).tabs({ active: <?php print $active_tab ?> });
    });
    </script>
  </head>
  <body>
```

CSS style sheets. One for the texts. One for the tables.

Set the first tab as the active one.

```
  <table>
  <tr>
    <th align="center" style="text-align:center">Folder : <?php print $path ?></th>
  </tr>
  </table>
```

Here begin the tables. This one has only one row for displaying the folder path.

```
  <div id="tabs" >
  <div id="tabs-container">
  <ul>
    <li><a href="#tab-1">ACLs</a></li>
    <li><a href="#tab-2">Mail adresses</a></li>
  </ul>
  </div>
```

“tabs” div contains the div “tabs-container” for the displaying the available tabs and the two div for the tab’s contents.

```
    <div id="tab-1">
```

Beginning of the content of the first tab.

```
  <form method="post" id="acls" name="acls" action="">
```

Opening user’s ACL form.

```
  <input type='hidden' name='file_path' value='<?php print $path; ?>'/></td>

  <table id='users_acls' name='users_acls'>
  <tr>
    <th>User</th><th>Permission</th><th> </th>
  </tr>
```

We open the form.

An hidden input contain the file path for resubmitting latter.

Header of the table.

```
<?php

  print "<tr>
  <td>All</td>\n
  <td>
    <select name='users_acls[0][permission]' id='auth_name_0' >";
  foreach ($available_perms as $key => $value) {
    $s = '';
    if( $key == $perms_imap[$permissions['anyone']['anyone']] ){
      $s =  ' selected="selected" ';
    }
```

Here is the first line of the user’s permissions.
By default, we always have an “All” user for the definition of anyone permissions.

Available permissions come from the variable “$available_perms” build from global variables. There are:
- ld: For directory listing only.

```
    print "\n       ".'<option value="'.$key.'"'.$s.'>'.$value.'</option>';
  }
  print "\n    </select>";
  print "<input type='hidden' name='users_acls[0][auth_name]' value='anyone'/>\n";
  print "<input type='hidden' name='users_acls[0][auth_type]' value='anyone'/>\n";
  print "
  </td>
  <td>
    -
  </td>
</tr>";
```

- ro: For reading only the emails in the folders.
- rw: For reading and modifying emails and folders.

For the permissions, an array is built from the form. Example:
"users_acls[0][auth_name]='anyone'"
"users_acls[0][auth_type]=' anyone'"
"users_acls[0][permission]='ld'"

We set the not editable values in the hidden inputs.

```
$i = 1;
if ( isset( $permissions['user'] ) ){
  foreach ($permissions['user'] as $key => $value) {
    print "<tr><td>".$key."</td>";

    print "<td><select name='users_acls[$i][permission]' >";
    foreach ($available_perms as $k => $v) {
      $s = '';
      print_r($perms_imap);
      if ( $k == $perms_imap[$value] ){
        $s = '  selected="selected" ';
      }
      print '<option '.$s.' value="'.$k.'">'.$v.'</option>';
    }
    print "</select>";
    print "<input type='hidden' name='users_acls[$i][auth_name]' id='user_auth_name_".$i."'
value='".$key."'/>\n";
    print "<input type='hidden' name='users_acls[$i][auth_type]' value='user'/></td><td>
    <img alt='Del.' src='styles/images/del.png' width='25px' height='25px'
onclick='remove_row(this,\"users_acls\")'/>
  </td>
\n";
    $i++;
  }
}
?>
```

User permission are defined here. The "$permissions" variable contains all the current set permission for this folder. The form is rebuilt with those values. Ex:

"users_acls[1][auth_name]='name'"
"users_acls[1][auth_type]='user'"
"users_acls[1][permission]='rw'"

"users_acls[2][auth_name]='name2'"
"users_acls[2][auth_type]='user'"
"users_acls[2][permission]='rw'"

In this part, the loop build a select input, to permit the change of the permission for an user but the username and type are not editable. We set those values in the hidden inputs. It is possible to remove the lines with the "Delete button" (represented by a red cross picture) that launch the "remove_row()" JavaScript function.

```
      <tr>
        <td colspan="3" >
          <a href="#" onclick="add_Row('users_acls')" id="edit_href"> + Add new</a></td>
      </tr>
    </table>
```

Button for adding a new line at the end of the user's ACLs table.

Closing the user's ACLs table.

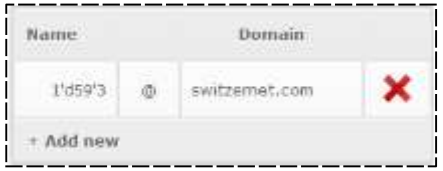
```
  <table id='groups_acls' name='groups_acls' >
  <tr>
    <th>Group</th><th>Permission</th>
    <th> </th>
  </tr>
```

Creating a new group's ACL table with headers.

```
<?php

$i = 1;

if ( isset( $permissions['group'] ) ){
  foreach ($permissions['group'] as $key => $value) {
    print "<tr><td>".$key."</td>";

    print "<td><select name='groups_acls[$i][permission]' >";
    foreach ($available_perms as $k => $v) {
      $s = '';
      print_r($perms_imap);
      if ( $k == $perms_imap[$value] ){
        $s = '  selected="selected" ';
      }
      print '<option '.$s.' value="'.$k.'">'.$v.'</option>';
    }
    print "</select>";
    print "<input type='hidden' name='groups_acls[$i][auth_name]' id='group_auth_name_".$i."'
value='".$key."'/>\n";
    print "<input type='hidden' name='groups_acls[$i][auth_type]' value='group'/></td><td>
    <img alt='Del.' src='styles/images/del.png' width='25px' height='25px'
onclick='remove_row(this,\"groups_acls\")'/>
  </td>
\n";
    $i++;
  }
}
?>
```

Here we build the table the same way as un the user's table.

We increment the html inputs in order to post an array of group's ACLs.



Name		Domain	
1'd59'3	@	switzernet.com	✗
+ Add new			

```
      <tr>
        <td colspan="3" >
          <a href="#" onclick="add_Row('groups_acls')" id="edit_href"> + Add new</a></td>
      </tr>
    </table>
```

Button for adding a new line at the end of the group's ACLs table.

```
    <table>
      <tr>
        <td>
          <input type="checkbox" name="recursive_acls" value="true"> recursive
```

A checkbox for applying the ACLs recursively is added to the form.

```
          <button style="float: right;" onclick="formSubmit('acls');">Apply ACLs</button>
        </td>
      </tr>
    </table>
```

```
    </form>
```

En of the ACL form.

```
</div>
```

End of the content of the first tab.

```
<div id="tab-2">
```

Beginning of the content of the second tab.

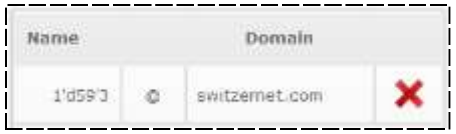
```

  <form method="post" id="addresses" name="addresses" action="">
  <input type='hidden' name='file_path' value='<?php print $path; ?>'/></td>

<?php

  $incoming_addresses = directory_incoming_addresses($path);
  #print_r($incoming_addresses);

?>
  <table id='mail_adresses' name='mail_adresses'>
  <tr>
    <th>Name</th><th style="width:0px">  </th><th>Domain</th><th></th>
  </tr>
  <?php
    if( count($incoming_addresses) > 0 ){
      $i = 0;
      foreach ($incoming_addresses as $key => $value) {
        $name = preg_replace('/\@.*$/','',$value['address']);
        $domain = preg_replace('/^[^\@]+@/','',$value['address']);
        print "<tr>";
        print "

        <td style='text-align:right;'>$name<input type='hidden' name='adr[$i][name]'
value=\"".$name."\"/>
        </td>
        <td style=\"width:0px\" >&nbsp;@&nbsp;</td>
        <td align=\"left\">$domain<input type='hidden' name='adr[$i][domain]' value='".$domain."'/>
        </td>
        <td>
          <img alt='Del.' src='styles/images/del.png' width='25px' height='25px'
onclick='remove_row(this,\"mail_adresses\")'/>
        </td>";
        print "</tr>";
        $i++;
```

A form for the addresses is created.

We set a hidden input with the path of the folder for that form.

We create a table of email addresses with information from the database we get with "directory_incoming_addresses($path)" function.

Ex:



Name		Domain	
1'd59'3	@	switzernet.com	✗

```
      }
    }
  ?>
```

```
  <tr>
    <td colspan="4" >
      <a href="#" onclick="add_Row_addr('mail_adresses')" id="edit_href"> + Add new</a>
    </td>
  </tr>
```

The "+ Add new" button that launch the "add_Row_addr('mail_adresses')" function in order to add a new row in t the "mail_addresses" table.

```
  </table>
```

End of the table.

```
  </form>
```

End of the form.

```
  <table>
    <tr>
      <td>
        <button style="float: right;" onclick="formSubmit('addresses');">Define addresses</button>
      </td>
    </tr>
  </table>
  </div>
</div>
```

The button for posting the"Mail addresses" form.

```
    <script type="text/javascript">
```

Beginning of the JavaScript part.

```
    function add_Row(table) {
      type = '';
      if ( table == 'users_acls' ){
        type = 'user';
      } else if ( table == 'groups_acls' ) {
        type = 'group';
      }
      var table = document.getElementById(table);
      var rowCount = table.rows.length;
      list = '';
      list_lenght = 1;
      for (var i = 1; i < rowCount; i++) {
        var name = '';
        if ( document.getElementById(type+"_auth_name_"+i) ){
          var input = document.getElementById(type+"_auth_name_"+i);
          if ( input.tagName.toLowerCase() == "select" ){
```

Function for adding a new line in the tables before the "+ Add new" button.

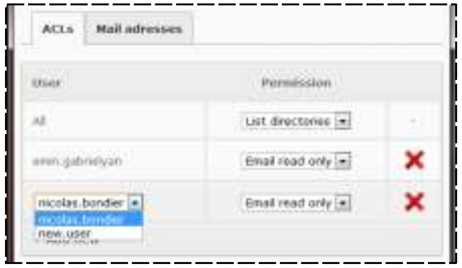
The table is the only parameter.

The first loop get all precedent set user names or group names in order to not suggest them again.

Then, the script pass the parameters to pages "get_select_user_list.php" or "get_select_group_list.php" with the names present in the tables as paramters. These scripts return an

```
             name = input.options[input.selectedIndex].value;
           } else {
             name = input.value;
           }
         }
         if ( name != '' ){
           list = list+name+",";
           list_lenght = list_lenght + 1;
         }
       }
       present = '';
       if ( list_lenght > 0 ){
         present = '?present='+list;
       }
       url = "include/get_select_"+type+"_list.php"+present;
       var select_list = get_html_content(url);
       if ( select_list != ''){
         var row = table.insertRow(rowCount-1);
         var cell1 = row.insertCell(0);
         var select = '<select name="'+type+'s_acls['+list_lenght+'][auth_name]"
id="'+type+'_auth_name_'+list_lenght+'">'+select_list+'</select>';
         cell1.innerHTML = select;
         var permissions = "<?php
     $i = $i+1;
     print '<select name=\""+type+"s_acls["+list_lenght+"][permission]\">';
     foreach ($available_perms as $key => $value) {
       print '<option value=\"'.$key.'\">'.$value.'</option>';
     }
     print '</select>';
     ?>"+"<input type='hidden' name=\""+type+"s_acls["+list_lenght+"][auth_type]\" value='"+type+"'/>";
         var cell2 = row.insertCell(1);
         cell2.innerHTML = permissions;
         var dell_cell = "<img alt='Del.' src='styles/images/del.png' width='25px' height='25px'
onclick='remove_row(this,\""+type+"s_acls\")'/>";
         var cell3 = row.insertCell(2);
         cell3.innerHTML = dell_cell;
       } else {
         alert("There are no more "+type+"s to add to this table.");
       }
     }
```

html code with the available option like:
“<option>name</option>
<option>name2</option>”.
The options are included in a select list and past in the table with other available fields (available ACLs).



User	Permission	
All	List directories	-
armen.gabrielyan	Email read only	✗
nicolas.bondier	Email read only	✗

If all users or groups have been added to the table, a message appears “There are no more "+type+"s to add to this table”.

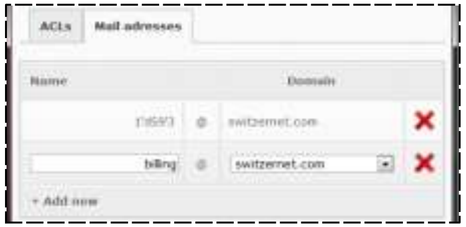
```
     function add_Row_addr(table) {
       var table = document.getElementById(table);
       var rowCount = table.rows.length;
       list = '';
       list_lenght = 1;
       url = "include/get_select_domains.php";
       var suggested_name = get_html_content(url);
```

This function concern the adding of a row in the address table of the “Mail addresses” panel.

It works the same as the “add_Row()” function, except it doesn’t get all the used values for the “Names” and

```
      var row = table.insertRow(rowCount-1);
      var cell1 = row.insertCell(0);
      var local_part = '<input type="text" name="adr['+(rowCount - 2)+'][name]" style="text-align:right;
width:100%" value="<?php print $folder_name ?>"/>';
      cell1.innerHTML = local_part;
      var cell2 = row.insertCell(1);
      cell2.style.width = '0px';
      cell2.innerHTML = '&nbsp;@&nbsp;';
      var cell3 = row.insertCell(2);
      cell3.innerHTML = '<select name="adr['+(rowCount - 2)+'][domain]" ><?php
      foreach( $available_domains as $dom ){
        print "<option value=\"$dom\">$dom</option>";
      }
      ?></select>';
      var dell_cell = "<img alt='Del.' src='styles/images/del.png' width='25px' height='25px'
onclick='remove_row(this,\"mail_adresses\")'/>";
      var cell4 = row.insertCell(3);
      cell4.innerHTML = dell_cell;
    }
```

"Domain" but these values are suggested by the "include/get_select_domains.php" PHP script, based on the folder name.



```
    function remove_row(r,table_id){
      var row= r.parentNode.parentNode;
      var table = document.getElementById(table_id);
      table.deleteRow(row.rowIndex);
    }
```

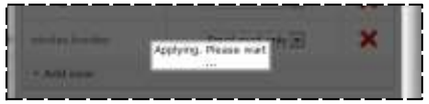
The remove row function, only get the cell element usually passed with "this" as argument, retrieve the row and remove it.
This way, inputs of this row will not be present anymore in the form.

```
    function get_html_content( url ) {
         var result = null;
         var scriptUrl = url;
         $.ajax({
            url: scriptUrl,
            type: 'get',
            dataType: 'html',
            async: false,
            success: function(data) {
                result = data;
            }
         });
         return result;
    }
```

A function for getting html content of a PHP page.

```
    function formSubmit(form){
      $.blockUI({ message: "Applying. Please wait ..." });
      setTimeout($.unblockUI, 100000);
      document.getElementById(form).submit();
    }
```

The function for submitting form. Not that we use a library, "jquery.blockUI.js" for blocking the window UI and displaying a waiting message.



```
    <?php
      if ($update_parent_adr == true){
    ?>
?       window.opener.load_mail_list_to_span(?"<?php echo 'span_mail_'.$span_id_to_update ?>",?"<?php
echo $path ?>");
    <?php
      }
    ?>
```

This window launch the "load_mail_list_to_span()" of the parent window in order to reload span with mail addresses information for this folder.

```
    <?php
      if ($update_parent_acls == true){
    ?>
      window.opener.load_acls_list_to_span(?"<?php echo 'span_acls_'.$span_id_to_update ?>",?"<?php echo
$path ?>",?"<?php echo json_encode($recursion); ?>");
    <?php
      }
    ?>
```

This window launch the "load_acls_list_to_span()" of the parent window in order to reload span with ACLs information for this folder and subfolders if the option is set to "true".

```
    window.focus()
```

Once this is done, we focus back to our child window.

```
  </script>
```

End of the scripts.

```
</body>
</html>
```

End of the document.

global_variables.php

```
<?php

//Variables générales
define('ROOT', __DIR__ .'../');
define('MAIL_ROOT_FOLDER', '/mnt/switzernet_rbd/131001-dovecot-data/Maildir/root');
define('PERMS_GUI', serialize( array( "ro" => "Email read only", "rw" => "All permissions", "ld" =>
"List directories" ) ) );
```

The general variables are explicit enough. I used them for retrieving information across many files, but also for easily move the project path.

```
define('PERMS_GUI_TO_IMAP', serialize( array( "ro" => "lr", "rw" => "kxeilrwts", "ld" => "l" ) ) );
define('PERMS_COL_TO_IMAP', serialize( array( "FE9A2E" => "lr", "00BB00" => "kxeilrwts", "6E6E6E" => "l"
) ) );
define('DEBUG', true);
define('GROUP_MAIL_ACCESS', 'dovecot-mail');
define('SYSTEM_SHARED_MAIL_USER', 'vmail');
define('VIRTUAL_MAILBOXES_DB', '/mnt/switzernet_rbd/131205-mail-server-www/131108-imap-
admin/virtual.sqlite');
define('AVAILABLE_DOMAINS', serialize( array( "switzernet.com", "intarnet.com",
"testmail.switzernet.com", "testmail.intarnet.com" ) ) );

?>
```

functions.php

```
<?php

function directory_display($base_path){
  $path = array();
  $base_path = format_path( $base_path );
  foreach (list_directory( MAIL_ROOT_FOLDER . "/" . $base_path ) as $dir) {

    $l_path = '';
    array_push($path, $dir);
    $l_path = implode('/', $path);
    $t_path = $base_path . "/" . $l_path;
    $t_path = format_path( $t_path  );

   # print 'base_path :'."$base_path"."<br>";
   # print 'l_path    :'."$l_path"."<br>";
   # print 't_path    :'."$t_path"."<br>";

    $html = acl_short_list( $t_path );
    $id = id_from_path(format_path($t_path));
    $r_mailbox = directory_incoming_addresses_short($t_path);

    print "<li id=\"li_".$id."\" class=\"dir_class\">";
    if (!empty_directory( MAIL_ROOT_FOLDER . "/" . $t_path)){
      print "<a style=\"font-weight:normal;font-style:normal;font-family:monospace; text-decoration:
none; color: #000000;\" href=\"#\"
onclick=\"open_close_tree('li_".$id."','".urlencode($t_path)."',this); return false;\" >"."[+]"."</a>";
    } else {
```

This function display the folder tree structure.

The "$base_path" argument is the path of the folder that will be opened.

The "list_directory()" function return the list of subfolders.

Then, for each of the subfolders, we get the list of ACLs, an id for the html elements, and the html code of the receiving mail addresses for the folders.

We then create a "<li></li>" element for the folder.

If it is not empty (tested by "empty_directory()" function) , we add the "[+]" notation at the beginning of the line and a function for opening/closing the folder on a "onclick" action.

```
      print "<span style=\"font-weight:normal;font-style:normal;font-family:monospace; text-decoration:
none; color: #000000;\">&nbsp;&nbsp;&nbsp;</span>";
     }
    print "<img src=\"styles/images/folder.png\" alt=\"\" height=\"16\" width=\"16\">
      <a style=\"font-weight:normal;color:#000000;\" class=\"folder-no-perms\"  href=\"#\"
onclick=\"open_control_window('folder_control_window.php?path=".urlencode($t_path)."'); return false;\"
>".$dir."</a> <span id='span_mail_".$id."'>".$r_mailbox."</span>";
    print " <span  id='span_acls_".$id."' class='acls_short_list' style='font-size:11px;color:#000000'>
| ".$html."</span>";

    print "</li>\n";
    array_pop($path);
  }
}
```

Then we add the folder image and the directory name. The directory name has has a link with a JavaScript action for opening the management window.

Finally, we and the spans with the receiving mailboxes and ACLs short description

```
function directory_incoming_addresses_short($path){
  $array = array();
  $html = '';
  $font_size = "11px";


  $dbsql = new SQLite3( VIRTUAL_MAILBOXES_DB );
  $req = 'SELECT * FROM \'virtual_Mailboxes\' WHERE path = \''.$path.'\'';
  $results = $dbsql->query($req);
  #print "$req";
  while ($row = $results->fetchArray()) {
    array_push($array, $row);
  }

  foreach ($array as $key => $value) {
      $html .= "<span class='short_acl' style='font-size:11px;' >@". ($key + 1 )."<span class='label'>
".$value['address']."</span> </span>,";

  }

  if ( strlen($html) > 0 ){
    $html = substr_replace($html ,"",-1);
    return "<span class='acls_short_list' style='font-size:11px;color:#000000'> | ".$html."</span>";
  } else {
    return '';
  }
}
```

This create the spans with the content “@1”, “@2”, … for each email address.

The information comes from the database used by both Dovecot and Postfix.

Each contain a span “class=‘label’” with the complete email address. It is not displayed until the mouse go over.

```
function directory_incoming_addresses($path){
  #$ret = false;
  $array = array();
```

Same function as above, but it returns only the array of email addresses for one directory.

```
  $html = '';
  $font_size = "11px";

  #print VIRTUAL_MAILBOXES_DB;
  $dbsql = new SQLite3( VIRTUAL_MAILBOXES_DB );
  $req = 'SELECT * FROM \'virtual_Mailboxes\' WHERE path = \''.$path.'\'';
  $results = $dbsql->query($req);
  #print "$req";
  while ($row = $results->fetchArray()) {
    array_push($array, $row);
  }
  return $array;
}
```

```
function save_incoming_adresses($adresses,$path){
  $ret = false;

  $available_domains = unserialize(AVAILABLE_DOMAINS);
  $dbsql = new SQLite3( VIRTUAL_MAILBOXES_DB );

  $req = "DELETE FROM 'virtual_Mailboxes' WHERE path = '".$path."';";
  $dbsql->exec($req);

  # control
  foreach ($adresses as $element) {
    //print "<br>-------<br>";
    //print_r($element);
    if ( preg_match('/[A-Za-z0-9\!\#\$\%\&\\\'\*\+\-\/\=\?\^\_\`\{\|\}\~]+/', $element['name'] ) ){
      $req = 'REPLACE INTO \'virtual_Mailboxes\' (`address`, `path`) VALUES (
"'.format_folder_name($element['name']).'@'.$element['domain'].'", "'.$path.'" );';
      //print $req."<br>";
      $dbsql->exec($req);
    }
  }
}
```

Function for saving the incoming email addresses for a folder the submission of the folder management window.

It matches only the available characters for email addresses.

```
function empty_directory($path){
  $cmd = 'sudo ls --file-type "'.$path.'"';
  // $cmd = escapeshellcmd($cmd);
  // print($cmd);
  exec($cmd, $files, $success);
  if ( $success == 0 ){
    foreach ( $files as $val ) {
      #print $val;
      if( preg_match('/\/$/',$val) && $val != 'new/' && $val != 'cur/' && $val != 'tmp/'){
        return false;
      }
    }
  }
```

Test if a directory has subfolders. For the Maildir format, we must exclude the regular “new”, “cur”, and “tmp” subfolders.

```
  return true;
}
```

```
function acl_short_list($path){
  $path = format_path($path);
  $colors_acl = unserialize(PERMS_COL_TO_IMAP);
  $colors_acl = array_flip($colors_acl);
  # print "acl_short_list : ".MAIL_ROOT_FOLDER . "/" . $path."<br>";
  $perms = get_folder_perms($path);

  #print MAIL_ROOT_FOLDER . "/" . $path;

  $html = "";
  $font_size = "11px";


  foreach ($perms as $key => $type) {
    if ( $key == 'anyone' ){
      $html .= "<span class='short_acl' style='font-
size:$font_size;color:#".$colors_acl[$type['anyone']]."' >*<span class='label'>anyone</span></span>";
      $html .= ',';
    } elseif ($key == 'user') {
      foreach ($type as $u => $p) {
        $initials = '';
        $ns = explode('.', $u);
        foreach ($ns as $n) {
           $initials = $initials . strtoupper(substr($n, 0, 1));
        }
        $html .= "<span class='short_acl' style='font-size:$font_size;color:#".$colors_acl[$p]."'
>".$initials."<span class='label'> ".$u." </span></span>";
        $html .= ',';
      }
    } elseif ( $key == 'group' ) {
      foreach ($type as $g => $p) {
        $html .= "<span class='short_acl' style='font-size:$font_size;color:#".$colors_acl[$p]."'
>".get_group_gid($g)."<span class='label'> group:".$g." </span></span>";
        $html .= ',';
      }
    }
  }
  if ( strlen($html) > 0 ){
    $html = substr_replace($html ,"",-1);
  }
  return $html;
}
```

This function built the short informations about ACLs on the main page.

It get the info from the “get_folder_perms()” function.

Then, it displays “*” for “anyone” the initials for users and GID for groups with a color corresponding to the permission.

The info of each is in a span. Each span contains a “class=‘label’” span with the full name of the user or group. There are not displayed until the mouse go over.

```
function list_directory($path){
  #echo($path);
  $ret = array();
  $cmd = 'sudo ls --file-type "'.$path.'"';
  $cmd = escapeshellcmd($cmd);
  # echo "running as : " . exec("sudo whoami")."<br>";
  # echo $cmd."<br>";
  exec($cmd, $files, $success);
  if ( $success == 0 ){
    foreach ( $files as $val ) {
      if( preg_match('/\/$/',$val) && $val != 'new/' && $val != 'cur/' && $val != 'tmp/'){
        $val = preg_replace('/\/$/', '', $val);
        array_push($ret,$val);
      }
    }
  }
  return $ret;
}
```

This function list subfolders of the folder locate under the “$path” argument and return an array.

```
function get_folder_perms($path){
 #print "ici : $path<br>";
  $arr = array();
  $cmd = 'sudo cat "'.$path."/".'dovecot-acl"';
  $cmd = escapeshellcmd($cmd);
  exec($cmd, $acls, $success);
    #print_r($acls);

  if ( $success == 0 ){
    foreach ($acls as $key => $value) {
      if (preg_match('/.+/', $value)){
        $type = '';
        $name = '';
        $perms = '';
        if ( preg_match('/^anyone/', $value ) ){
          $type  = 'anyone';
          $name = 'anyone';
          $perms = preg_replace('/^anyone(.*)/', '$1', $value);
        } elseif ( preg_match('/^user=/', $value) ) {
          $type  = 'user';
          $name  = preg_replace('/^user=([^ ]+).*/', '$1', $value);
          $perms = preg_replace('/^user=[^ ]+(.*)/', '$1', $value);
        } elseif ( preg_match('/^group=/', $value) ) {
          $type  = 'group';
          $name  = preg_replace('/^group=([^ ]+).*/', '$1', $value);
          $perms = preg_replace('/^group=[^ ]+(.*)/', '$1', $value);
        }
        $perms = trim($perms);
        $arr[$type][$name] = $perms;
```

Function for getting the ACLs from a “dovecot-acl” file.

For each line, it stores the permission in a hash with this structure:

“Array[type][name] = permission”

Then, it pass the array to the “correct_perms()” function that makes verifications and replacement on the array.

We verify it the returned array is equal to the one passed as argument.

If not, it means that something were wrong and we replace all the permissions with default ones.

```
      }
    }
  }
  if( !correct_perms($arr) ){
    error_log("perms are not correct ! reseting_file ...");
    return reset_file_perms($path);
  }
  return $arr;
}
  return $arr;
}
```

```
function correct_perms($perms){
  $owner_set = false;
  $available_perms = unserialize(PERMS_GUI_TO_IMAP);
  $available_perms_flip = array_flip($available_perms);
  foreach ($perms as $type => $arr) {
    if ($type == 'anyone'){
      $owner_set = true;
    }
    foreach ($arr as $name => $user_perm) {
      if( ! isset( $available_perms_flip[$user_perm] ) ){
        return false;
      }
    }
  }
  if( !$owner_set ){
    return false;
  }
  return true;
}
```

Function that verify the permissions. It checks if the set values are available in the list of allowed permissions. And it verify that the “anyone” user has permissions. The return is true or false.

```
function reset_file_perms($folder){
  $imap_perms = unserialize(PERMS_GUI_TO_IMAP);
  $new = array();
  $new['file_path'] = $folder;
  $new['users_acls'][0]['auth_name'] = 'anyone';
  $new['users_acls'][0]['auth_type'] = 'anyone';
  $new['users_acls'][0]['permission'] = 'ld';
  replace_acls($new);
  return array( 'anyone' => array( 'anyone' => $imap_perms['ld'] ) );
}
```

Resetting the folder permissions consist in deleting all users from the “dovecot-acl” file, resetting the owner of the file and resetting the owner of the files in the UNIX file system.

This is done through the “replace_acls()” function.

```
function replace_acls($perms){
  $imap_perms = unserialize(PERMS_GUI_TO_IMAP);
  $folder = MAIL_ROOT_FOLDER . "/" . $perms['file_path'];
```

This is the main function for setting the ACLs.

```
  $dovecot_file = $folder."/".'dovecot-acl';
  $user_exists_D_ACL  = array();
  $group_exists_D_ACL = array();
  $user_exists_U_ACL  = array();
  $group_exists_U_ACL = array();
  $folder_files = array();
  $success_listing = 1;
  $cmd = 'sudo ls --file-type "'.$folder.'" | egrep "[^\/]$" |  grep -v "dovecot-acl" | grep -v
"dovecot.index.log"';
  exec($cmd, $folder_files, $success_listing);

  $success = $success_listing;

  $files = $folder_files;

  system('sudo rm "'.$dovecot_file.'"', $retval);
```

We define variables.

Here is the command to get all files in folder. We do not get the "dovecot.index.log" file, because dovecot bug when we try to midify anything on it. It cause lock problems. The "dovecot-acl" file must be readable by dovecot only. We change this better.

```
  ###########################################
  # Control default ACLs :
  #


  if( !isset($perms['users_acls']) ){
    $perms['users_acls'] = array( 0 => array( 'permission' => 'ld', 'auth_type' => 'anyone', 'auth_name'
=> 'anyone' ) );
  }

  if( !isset($perms['users_acls'][0]) || $perms['users_acls'][0]['auth_name'] != 'anyone' ){
    $perms['users_acls'] = array( 0 => array( 'permission' => 'ld', 'auth_type' => 'anyone', 'auth_name'
=> 'anyone' ) );
  }
```

Control of the presence of the "anyone" user. If it is not set in the "$perms" array, then we erase everything and set default values.

```
  ###########################################
  # UNIX ACLs :
  #

  # Default :
  $cmd = "sudo chown ".SYSTEM_SHARED_MAIL_USER.":".SYSTEM_SHARED_MAIL_USER." '".$folder."'";
  system($cmd, $retval);
  if ( $success == 0 ){
    foreach ( $files as $val ) {
      $cmd = 'sudo chown '.SYSTEM_SHARED_MAIL_USER.':'.SYSTEM_SHARED_MAIL_USER.'
\''.$folder.'/'.$val.'\'';
      system($cmd, $retval);
    }
  }
```

Here, we begin to set UNIX ACLs for a folder.

First we set the owner of the folder to the "vmail" user.

We do the same for all files included in the folder we get above.

The same owner is applied to the Maildir regular folders : "new", "cur" and "tmp".

```
$cmd = "sudo chown -R ".SYSTEM_SHARED_MAIL_USER.":".SYSTEM_SHARED_MAIL_USER." '".$folder."/cur'";
system($cmd, $retval);

$cmd = "sudo chmod -R 770 '".$folder."/cur'";
system($cmd, $retval);

$cmd = "sudo chown -R ".SYSTEM_SHARED_MAIL_USER.":".SYSTEM_SHARED_MAIL_USER." '".$folder."/new'";
system($cmd, $retval);

$cmd = "sudo chmod -R 770 '".$folder."/new'";
system($cmd, $retval);

$cmd = "sudo chown -R ".SYSTEM_SHARED_MAIL_USER.":".SYSTEM_SHARED_MAIL_USER." '".$folder."/tmp'";
system($cmd, $retval);

$cmd = "sudo chmod -R 770 '".$folder."/tmp'";
system($cmd, $retval);

$cmd = "sudo chmod g+rwxs '".$folder."'";
system($cmd, $retval);
```

We also set the UNIX permissions to "rwx" for this user and group and no other permission for other users.

The "chmod g+rwxs" command is the permission to keep the group as default for contained new files/subfolders.

```

# Remove special ACLs

$cmd = "sudo setfacl --remove-all '".$folder."'";
system($cmd, $retval);

$cmd = "sudo setfacl -R --remove-all '".$folder."/cur'";
system($cmd, $retval);

$cmd = "sudo setfacl -R --remove-all '".$folder."/new'";
system($cmd, $retval);

$cmd = "sudo setfacl -R --remove-all '".$folder."/tmp'";
system($cmd, $retval);

if ( $success == 0 ){
  foreach ( $files as $val ) {
      $cmd = 'sudo setfacl --remove-all \''.$folder.'/'.$val.'\'  >/dev/null';
      #print "$cmd\n<br>";
      system($cmd, $retval);
  }
}
```

We remove all UNIX ACLs on the files and folders.

```
if( isset($perms['users_acls']) ){
  foreach ($perms['users_acls'] as $value) {

    ###########################################
```

Here, we begin to read all ACLs passed as parameters.

```
      # IMAP ACLs :
      #

      $cmd = 'exit;';
      $p = $imap_perms[$value['permission']];
      if ( $value['auth_name'] == 'anyone' ){
        $cmd = 'echo "anyone '.$p.'" | sudo tee -a "'.$dovecot_file.'" 1>/dev/null';
      }
      if ( $value['auth_type'] == 'user' && !isset( $user_exists_D_ACL[$value['auth_name']] ) ){
        $cmd = 'echo "user='.$value['auth_name'].' '.$p.'" | sudo tee -a "'.$dovecot_file.'" >/dev/null
2>&1';
        $user_exists_D_ACL[$value['auth_name']] = 1;
      }
      system($cmd, $retval);
```

If it is the "anyone" user, we add a line, for example:
"anyone lr".

If it is another user, the format is:
"user=username ld".

Format is explained at this page:
http://wiki2.dovecot.org/ACL

```
      ###########################################
      # UNIX ACLs :
      #

      # Anyone :
      if( $value['auth_name'] == 'anyone' ){

        if( $value['permission'] == 'ld' ){

          $cmd = "sudo setfacl -m o:r-x '".$folder."'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -dm o:r-x '".$folder."'";
          system($cmd, $retval);

          if ( $success == 0 ){
            foreach ( $files as $val ) {
                $cmd = 'sudo setfacl -m o:--- \''.$folder.'/'.$val.'\'';
                system($cmd, $retval);
            }
          }

          $cmd = "sudo setfacl -m o:rwx '".$folder."/dovecot-uidlist'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m o:--- '".$folder."/cur'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m o:--- '".$folder."/new'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m o:--- '".$folder."/tmp'";
```

For the UNIX ACLs, I divided the script for each kind of users and permissions. This was easier for debugging the dovecot bugs.

We set the minimum permission on each file for each kind of access.

If the permission is "ld" (List Directories).

We set the permissions for anyone to "r-x" on the folder and the default to the same value for new files and directories created inside.

"r" on directory is used for listing, and "x" is used for entering the directory.

For each of the files (without the exception file for which we do not change the permissions) we give no access, except for the "dovecot-uidlist" file, which need to have "rwx" permissions (for Dovecot).

The Maildir folders "new", "cur", "tmp" and their content are not accessible by "anyone".

```
        system($cmd, $retval);


      } elseif ( $value['permission'] == 'ro' ) {

        $cmd = "sudo setfacl -m o:r-x '".$folder."'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -dm o:r-x '".$folder."'";
        system($cmd, $retval);

        if ( $success == 0 ){
          foreach ( $files as $val ) {
              $cmd = 'sudo setfacl -m o:r-- \''.$folder.'/'.$val.'\'';
              system($cmd, $retval);
          }
        }

        $cmd = "sudo setfacl -m o:rwx '".$folder."/dovecot-uidlist'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -R -m o:r-x '".$folder."/cur'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -R -m o:r-x '".$folder."/new'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -R -m o:r-x '".$folder."/tmp'";
        system($cmd, $retval);


      } elseif ( $value['permission'] == 'rw' ){

        $cmd = "sudo setfacl -m o:rwx '".$folder."'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -dm o:rwx '".$folder."'";
        system($cmd, $retval);

        if ( $success == 0 ){
          foreach ( $files as $val ) {
              $cmd = 'sudo setfacl -m o:rw- \''.$folder.'/'.$val.'\'';
              system($cmd, $retval);
          }
        }

        $cmd = "sudo setfacl -m o:rwx '".$folder."/dovecot-uidlist'";
        system($cmd, $retval);
```

If the permission is "ro" (email Read Only).

We authorize access to the directory, without being able to change the name or delete it.

All dovecot files under this folder can be accessed for reading.

Except "dovecot-uidlist" which need to be "rwx" by the user for afford Dovecot bug.

The "new", "cur" and "tmp" folders and included files (emails) can be accessed but cannot be modified.

Finally, for the "rw" (emails Read Write) permission, we set the permissions to the maximum values.

```
      $cmd = "sudo setfacl -R -m o:rwx '".$folder."/cur'";
      system($cmd, $retval);

      $cmd = "sudo setfacl -R -m o:rwx '".$folder."/new'";
      system($cmd, $retval);

      $cmd = "sudo setfacl -R -m o:rwx '".$folder."/tmp'";
      system($cmd, $retval);

    }
  }
```

```
  # Users :

  if ( $value['auth_type'] == 'user' && !isset( $user_exists_U_ACL[$value['auth_name']] ) ){

    $user_exists_U_ACL[$value['auth_name']] = 1;

    if( $value['permission'] == 'ld' ){

      $cmd = "sudo setfacl -m u:".$value['auth_name'].":r-x '".$folder."'";
      system($cmd, $retval);

      $cmd = "sudo setfacl -dm u:".$value['auth_name'].":r-x '".$folder."'";
      system($cmd, $retval);

      if ( $success == 0 ){
        foreach ( $files as $val ) {
          $cmd = 'sudo setfacl -m u:'.$value['auth_name'].':--- \''.$folder.'/'.$val.'\'';
          system($cmd, $retval);
        }
      }

      $cmd = "sudo setfacl -m u:".$value['auth_name'].":rwx '".$folder."/dovecot-uidlist'";
      system($cmd, $retval);

      $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":--- '".$folder."/cur'";
      system($cmd, $retval);

      $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":--- '".$folder."/new'";
      system($cmd, $retval);

      $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":--- '".$folder."/tmp'";
      system($cmd, $retval);

    } elseif ( $value['permission'] == 'ro' ) {

      $cmd = "sudo setfacl -m u:".$value['auth_name'].":r-x '".$folder."'";
```

```
Unix permission for users are set
the same way as for "anyone", except
we define the name of the user.
```

```
          system($cmd, $retval);

          $cmd = "sudo setfacl -dm u:".$value['auth_name'].":r-x '".$folder."'";
          system($cmd, $retval);

          if ( $success == 0 ){

            foreach ( $files as $val ) {
                $cmd = 'sudo setfacl -m u:'.$value['auth_name'].':r-- \''.$folder.'/'.$val.'\'';
                system($cmd, $retval);

            }
          }

          $cmd = "sudo setfacl -m u:".$value['auth_name'].":rwx '".$folder."/dovecot-uidlist'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":r-x '".$folder."/cur'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":r-x '".$folder."/new'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":r-x '".$folder."/tmp'";
          system($cmd, $retval);

        } elseif ( $value['permission'] == 'rw' ){

          $cmd = "sudo setfacl -m u:".$value['auth_name'].":rwx '".$folder."'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -dm u:".$value['auth_name'].":rwx '".$folder."'";
          system($cmd, $retval);

          if ( $success == 0 ){
            foreach ( $files as $val ) {
                $cmd = 'sudo setfacl -m u:'.$value['auth_name'].':rwx \''.$folder.'/'.$val.'\'';
                system($cmd, $retval);
            }
          }

          $cmd = "sudo setfacl -m u:".$value['auth_name'].":rwx '".$folder."/dovecot-uidlist'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":rwx '".$folder."/cur'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":rwx '".$folder."/new'";
          system($cmd, $retval);
          $cmd = "sudo setfacl -R -m u:".$value['auth_name'].":rwx '".$folder."/tmp'";
          system($cmd, $retval);

        }
      }
    }
  }
```

```
  if( isset($perms['groups_acls']) ){
    foreach ($perms['groups_acls'] as $value) {

      $p = $imap_perms[$value['permission']];
      $cmd = 'exit;';
      if ( $value['auth_type'] == 'group' && !isset( $group_exists_D_ACL[$value['auth_name']] ) ){
        $cmd = 'echo "group='.$value['auth_name'].' '.$p.'" | sudo tee -a "'.$dovecot_file.'" >/dev/null
2>&1';
        $group_exists_D_ACL[$value['auth_name']] = 1;
      }
      system($cmd, $retval);

      if ( $value['auth_type'] == 'group' && !isset( $group_exists_U_ACL[$value['auth_name']] ) ){
        $group_exists_U_ACL[$value['auth_name']] = 1;

        if( $value['permission'] == 'ld' ){

          $cmd = "sudo setfacl -m g:".$value['auth_name'].":r-x '".$folder."'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -dm g:".$value['auth_name'].":r-x '".$folder."'";
          system($cmd, $retval);

          if ( $success == 0 ){
            foreach ( $files as $val ) {
              $cmd = 'sudo setfacl -m g:'.$value['auth_name'].':--- \''.$folder.'/'.$val.'\'';
              system($cmd, $retval);
            }
          }

          $cmd = "sudo setfacl -m g:".$value['auth_name'].":rwx '".$folder."/dovecot-uidlist'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":--- '".$folder."/cur'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":--- '".$folder."/new'";
          system($cmd, $retval);
```

Unix permission for groups are set the same way as for "anyone", except we define the name of the group.

```
        $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":--- '".$folder."/tmp'";
        system($cmd, $retval);

      } elseif ( $value['permission'] == 'ro' ) {

        $cmd = "sudo setfacl -m g:".$value['auth_name'].":r-x '".$folder."'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -dm g:".$value['auth_name'].":r-x '".$folder."'";
        system($cmd, $retval);

        if ( $success == 0 ){

          foreach ( $files as $val ) {
              $cmd = 'sudo setfacl -m g:'.$value['auth_name'].':r-- \''.$folder.'/'.$val.'\'';
              system($cmd, $retval);

          }
        }

        $cmd = "sudo setfacl -m g:".$value['auth_name'].":rwx '".$folder."/dovecot-uidlist'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":r-x '".$folder."/cur'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":r-x '".$folder."/new'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":r-x '".$folder."/tmp'";
        system($cmd, $retval);

      } elseif ( $value['permission'] == 'rw' ){

        $cmd = "sudo setfacl -m g:".$value['auth_name'].":rwx '".$folder."'";
        system($cmd, $retval);

        $cmd = "sudo setfacl -dm g:".$value['auth_name'].":rwx '".$folder."'";
        system($cmd, $retval);

        if ( $success == 0 ){
          foreach ( $files as $val ) {
              $cmd = 'sudo setfacl -m g:'.$value['auth_name'].':rwx \''.$folder.'/'.$val.'\'';
              system($cmd, $retval);
          }
        }

        $cmd = "sudo setfacl -m g:".$value['auth_name'].":rwx '".$folder."/dovecot-uidlist'";
```

```
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":rwx '".$folder."/cur'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":rwx '".$folder."/new'";
          system($cmd, $retval);

          $cmd = "sudo setfacl -R -m g:".$value['auth_name'].":rwx '".$folder."/tmp'";
          system($cmd, $retval);

        }
      }
      system($cmd, $retval);
    }
  }

  $cmd = 'sudo rm "'. MAIL_ROOT_FOLDER . "/dovecot-acl-list\" >/dev/null 2>&1 ";
  system($cmd, $retval);
  $cmd = 'sudo chown '.SYSTEM_SHARED_MAIL_USER.':'.SYSTEM_SHARED_MAIL_USER.' "'.$dovecot_file.'"
>/dev/null 2>&1';
  system($cmd, $retval);

  $cmd = 'sudo chown '.SYSTEM_SHARED_MAIL_USER.':'.SYSTEM_SHARED_MAIL_USER.' \''.$dovecot_file.'\'';
  system($cmd, $retval);

  $cmd = 'sudo chmod 664 "'.$dovecot_file.'" >/dev/null 2>&1';
  system($cmd, $retval);

  $cmd = 'sudo setfacl -b \''.$dovecot_file.'\'';
  system($cmd, $retval);
}
```

```
function recursively_replace_acls( $perms , $subfolder = '' ){
  $perms['file_path'] = $perms['file_path'] . $subfolder;
  $folder = MAIL_ROOT_FOLDER . "/" . $perms['file_path'];
  $subs = list_directory($folder);
  replace_acls($perms);
  foreach ($subs as $dir) {
    recursively_replace_acls( $perms , "/" . $dir );
  }
}
```

This function is called for applying the ACLs recursively.

It increments the folder path with subfolder, then launch the "replace_acls()" function, and enter in subdirectories, and launch again itself with new path arguments.

```
function recursive_path_and_id( $path, $array = array() ){
  $folder = MAIL_ROOT_FOLDER . "/" . $path;
  $array[id_from_path(format_path($path))] = $path;
  $subs = list_directory($folder);
  foreach ($subs as $dir) {
```

This function returns an array of all subfolders of the folder passed as path in the email storage directory.

```
    $array = recursive_path_and_id( $path . "/" . $dir, $array );
  }
  return $array;
}
```

It is recursive and can retrieve all branches of the tree.

```
function is_in_group($username, $groupname){
  return fasle;
}
```

Not used.

```
function id_from_path($path){
  $id = format_path($path);
  $id = preg_replace("/\//", '-', $id);
  $id = preg_replace("/[^A-Za-z0-9-]/", '_', $id);
  $id = $id . "_" . md5($id);
  return $id;
}
```

A function for helping in JavaScript coding. It creates an unique id depending on the full path of the directory (the base is the root Maildir folder). This make html elements for one folder, unique.

```
function format_path($path){
  $path = preg_replace("/^\//", '', $path);
  $path = preg_replace("/\/$/", '', $path);
  return $path;
}
```

Just a function to trim the backslashes at the beginning and at the end of the path strings.

```
function get_group_gid($group_name){
  #return $group_name;
  $ret = false;
  $cmd = 'getent group '.$group_name.' | cut -d":" -f3';
  exec($cmd, $gid, $success);
  if ($success == 0 && $gid != ''){
    $ret = $gid[0];
  }
  return $ret;
}
```

Function for getting the group id of a group.

```
function verify_user(){
  if ( !isset($_SERVER['REMOTE_USER']) || $_SERVER['REMOTE_USER'] == '' ||
!is_administrator($_SERVER['REMOTE_USER']) ){
      print '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
      <title>Access forbidden</title>
    </head>
    <body>
  ';
      print "<h1>Access forbidden</h1><p>You are not authorised to access this page !</p>
      </div>
```

A function to verify if the user is authorized to access the administration web page.

If not, it display a text and exit for returning the page to the user browser.

```
    </body>
  </html>
  ";
  exit;
  }
}
```

```
function is_administrator($user){
  $cmd = 'getent group imap-administrator | cut -d":" -f4';
  exec($cmd, $users_string, $success);
  if ($success == 0){
    $users = explode( ',' , $users_string[0] );
    foreach ($users as $name) {
      if( $name == $user){
        return true;
      }
    }
  }
  return false;
}
```

A function to verify is the user is in the admin group.

```
function format_folder_name($t){
  $t = preg_replace('/.*\//','',$t);
  $t = preg_replace('/@.*$/','',$t);
  $t = str_replace(' ','_',$t);
  $t = str_replace('"','\'',$t);
  return $t;
}
```

This function is used for formatting the folder names.

```
?>
```

get_acl_short_list.php

```
<?php
```

```
include_once "functions.php";
include_once "global_variables.php";
verify_user();
```

The page returning the summary of ACLs short list.

```
$path = '';
if ( isset( $_GET['path'] ) ){
  $path = $_GET['path'];
  $path = urldecode($path);
}
$html = acl_short_list($path);
print " | ".$html."";
```

```
?>
```

The parameter passed in the GET is the path of the folder for which we want to know the ACLs.

get_adr_short_list.php

```
<?php
```

```
include_once "functions.php";
include_once "global_variables.php";
verify_user();

$path = '';
if ( isset( $_GET['path'] ) ){
  $path = $_GET['path'];
  $path = urldecode($path);
}


$html = directory_incoming_addresses_short($path);
//error_log($html);

print " ".$html."";
```

```
?>
```

The page returning the summary of reception email address.

The parameter passed in the GET is the path of the folder for which we want to know the reception email address.

get_ids_and_paths_to_reload.php

```
<?php
```

```
include_once "functions.php";
include_once "global_variables.php";
```

This page return a JSON array with the html ids of the html elements to

```
verify_user();

$path = '';
if ( isset( $_GET['path'] ) ){
  $path = $_GET['path'];
  $path = urldecode($path);
}

$return = array();
$return = recursive_path_and_id($path);

echo json_encode($return);
```

```
?>
```

reload. These elements are the ACLs summary and the summary of reception email addresses on the folder tree.

get_select_group_list.php

```
<?php
```

```
include_once "functions.php";
include_once "global_variables.php";
verify_user();

$groups = array();
$p1 = array();
$p2 = array();

$cmd = "sudo getent passwd";
$result = array();
exec($cmd, $result, $success);
$ldap_user_primary_groups = array();
foreach ($result as $key => $value) {
  if ($success == 0 && preg_match_all('/^([^:]*):[^:]*:([^:]*):([^:]*):(.*)/', $value,
$o,PREG_PATTERN_ORDER)){
    if( $o[2][0] > 1001 ){
      $ldap_user_primary_groups[$o[3][0]] = 1;
    }
  }
}
$cmd = 'sudo getent group';
$result = array();
exec($cmd, $result, $success);
foreach ($result as $key => $value) {
```

This PHP script is returning the list of available groups that is inserted in the select on the setup page.

First, it gets all users settings and save the primary group of each one to exclude them later.

Then, we get all groups with a GID greater than 1001. This is the starting GID in the LDAP.

If the group is not the mail group or any of the user’s primary group, we save it.

```
  if ($success == 0 && preg_match_all('/^([^:]*):[^:]*:([^:]*):(.*)/', $value,$out,PREG_PATTERN_ORDER)){
    if ( $out[2][0] > 1001 && $out[2][0] <= 10000 && $out[1][0] != SYSTEM_SHARED_MAIL_USER && $out[1][0]
!= GROUP_MAIL_ACCESS && ! isset($ldap_user_primary_groups[$out[2][0]]) ){
      $groups[$out[1][0]] = 1;
    }
  }
}


if(isset($_GET['present'])){
  $p1 = explode(',',$_GET['present']);
  foreach ($p1 as $key => $value) {
    if( $value != ''){
      $p2[$value] = 0;
    }
  }
}

foreach ($groups as $key => $value) {
  if( ! isset($p2[$key]) ){
    print "<option value=\"$key\">$key</option>\n";
  }
}

?>
```

We get the groups present in the table of the administration window that are passed as parameters and we exclude these groups for our final list.

get_select_user_list.php

```
<?php
  include_once "functions.php";
  include_once "global_variables.php";
  verify_user();

  $users = array();
  $mail_group = 'dovecot-mail';
  $cmd = 'sudo getent group dovecot-mail';
  $text_group = '';
  exec($cmd, $result, $success);
  $p1 = array();
  $p2 = array();
  $p3 = array();
```

The PHP script return the option list of the available users to set permissions to.

First, we get all users that are present in the array "$_GET['present']", which is the array of username already set in the administration window.

Then, we get all users that are in the "dovecot-mail" group, which is

```
  if(isset($_GET['present'])){
    $p1 = explode(',',$_GET['present']);
  }

  foreach ($p1 as $key => $value) {
    $p2[$value] = 0;
  }

  if ( $success == 0 ){
    $t_group = $result[0];
    #print_r $t_group."\n";
  }

  if ( preg_match('/^'.$mail_group.':[^:]*:[^:]*:.*/',$t_group) ){
    $pattern = '/^'.$mail_group.':[^:]*:[^:]*:(.*)/';
    $replacement = '$1';
    $users = explode(',', preg_replace($pattern, $replacement, $t_group));
  }

  foreach ($users as $key => $value) {
    if( ! isset($p2[$value]) ){
      print "<option value=\"$value\">$value</option>\n";
    }
  }
```

```
?>
```

the LDAP group authorized to access to mails.

For each users, if it is present in the array of already set users, we do nothing. Else we add the new option.

get_subfolders.php

```
<?php
```

```
include_once "functions.php";
include_once "global_variables.php";
verify_user();

$path = '';
if ( isset( $_GET['path'] ) ){
  $path = $_GET['path'];
  $path = urldecode($path);
}
```

The PHP script which is used for displaying a layer of subfolders of one folder in the folder tree.

We get the path parameter in the posted GET.

```
$directories = list_directory( MAIL_ROOT_FOLDER . "/" . $path );

if ( isset($directories) && count( $directories ) > 0 ){
print "<li>
  <ul>
";

directory_display($path);

print "  </ul>
</li>";

}

if (false){ ?>

  <script type="text/javascript">
  //alert(<?php print "'$path'" ?>);
  </script>
```

```
<?php } ?>
```

Then we call the function to display subdirectories of the folder with the full system path.

Some debug.

References

This document: http://switzernet.com/3/public/140116-imap-addresses-and-acls-admin/

Dovecot shared folders and LDAP: http://switzernet.com/3/public/131212-shared-imap-dovecot/index.pdf

Apache 2 and HTTP Authentication with PAM: http://icephoenix.us/linuxunix/apache-and-http-authentication-with-pam/

*   *   *