**ADMIN WEB PAGE FOR**

**IMAP ACLs**

**AND**

**Shared FOLDERS INCOMING EMAILS**

Document created on 2014-01-16  
Nicolas Bondier

[[pdf](index.pdf)][[doc](index.docx)][[htm](index.htm)]

Contents

[Introduction 3](#_Toc377576013)

[Prerequisites 4](#_Toc377576014)

[Presentation 5](#_Toc377576015)

[Configuration of apache server 8](#_Toc377576016)

[Installation of the web server 8](#_Toc377576017)

[Basic authentication with PAM 8](#_Toc377576018)

[Sudo for www-data user 9](#_Toc377576019)

[Code documentation 10](#_Toc377576020)

[Files tree 10](#_Toc377576021)

[index.php 11](#_Toc377576022)

[folder\_control\_window.php 15](#_Toc377576023)

[﻿global\_variables.php 24](#_Toc377576024)

[functions.php 25](#_Toc377576025)

[get\_acl\_short\_list.php 41](#_Toc377576026)

[get\_adr\_short\_list.php 42](#_Toc377576027)

[get\_ids\_and\_paths\_to\_reload.php 42](#_Toc377576028)

[get\_select\_group\_list.php 43](#_Toc377576029)

[get\_select\_user\_list.php 44](#_Toc377576030)

[get\_subfolders.php 45](#_Toc377576031)

[References 47](#_Toc377576032)

# 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.

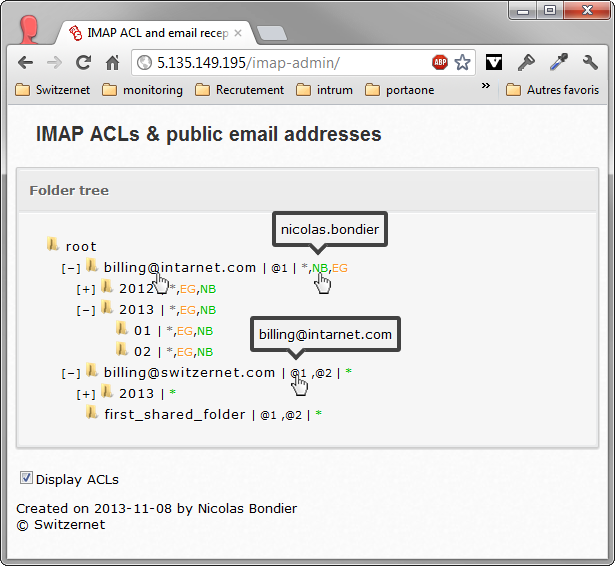
# Presentation

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.

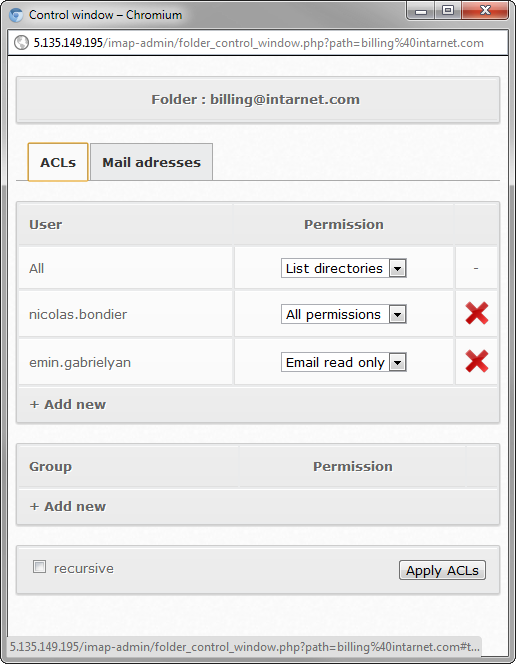
The page is protected with http authentication. This authentication permit only on group of the LDAP users to login.

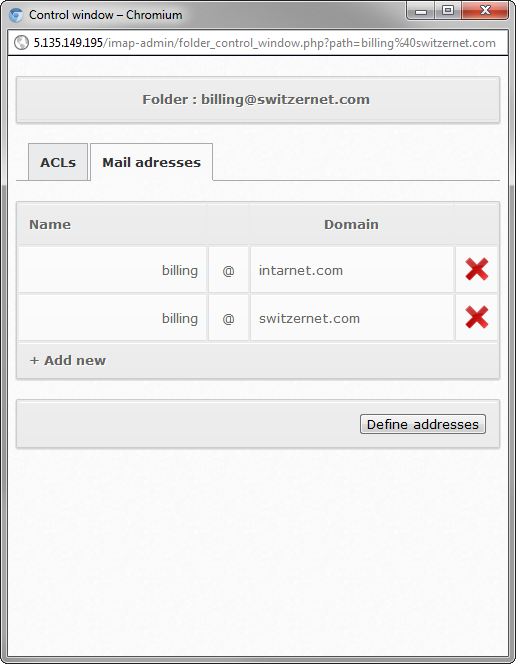
The index page display the shared folders tree. On each folder, a summary of the settings is displayed:

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



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.





# 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](data/131108-imap-admin)/

|-- [favicon.png](data/131108-imap-admin/favicon.zip)

|-- [folder\_control\_window.php](data/131108-imap-admin/folder_control_window.zip)

|-- [include](data/131108-imap-admin/include)

|   |-- [functions.php](data/131108-imap-admin/include/functions.zip)

|   |-- [get\_acl\_short\_list.php](data/131108-imap-admin/include/get_acl_short_list.zip)

|   |-- [get\_adr\_short\_list.php](data/131108-imap-admin/include/get_adr_short_list.zip)

|   |-- [get\_ids\_and\_paths\_to\_reload.php](data/131108-imap-admin/include/get_ids_and_paths_to_reload.zip)

|   |-- [get\_select\_group\_list.php](data/131108-imap-admin/include/get_select_group_list.zip)

|   |-- [get\_select\_user\_list.php](data/131108-imap-admin/include/get_select_user_list.zip)

|   |-- [get\_subfolders.php](data/131108-imap-admin/include/get_subfolders.zip)

|   `-- [global\_variables.php](data/131108-imap-admin/include/global_variables.zip)

|-- [index.php](data/131108-imap-admin/index.zip)

|-- [js](data/131108-imap-admin/js)

|   |-- [jquery-1.10.2.min.js](data/131108-imap-admin/js/jquery-1.10.2.min.zip)

|   `-- [jquery.blockUI.js](data/131108-imap-admin/js/jquery.blockUI.zip)

|-- [styles](data/131108-imap-admin/styles)

|   |-- [images](data/131108-imap-admin/styles/images)

|   |   |-- [background.jpg](data/131108-imap-admin/styles/images/background.zip)

|   |   |-- [del.png](data/131108-imap-admin/styles/images/del.zip)

|   |   |-- [folder.png](data/131108-imap-admin/styles/images/folder.zip)

|   |   `-- [switzernet-logo.png](data/131108-imap-admin/styles/images/switzernet-logo.zip)

|   |-- [style.css](data/131108-imap-admin/styles/style.zip)

|   `-- [table.css](data/131108-imap-admin/styles/table.zip)

`-- [virtual.sqlite](data/131108-imap-admin/virtual.zip)

## 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.  C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_20_44.png  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.  C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_19_39.png |
| **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: C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_12_56.png |
| **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:  C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_14_29.png |
| **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:  C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_16_11.png |
| </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    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. |
| $path = '';  **if** (isset($\_GET['path'])){    $path = $\_GET['path'];    $path = preg\_replace("/^\//", '', $path);  } | Formatting the path set as parameter in the URL |
| $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**;  }    **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;  } | 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.  - 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" />      <link rel="stylesheet" href="styles/table.css" type="text/css" />      <script>      $(**function**() {        $( "#tabs" ).tabs({ active: <?php **print** $active\_tab **?>** });      });      </script>    </head>    <body> | Document type definition and headers.        We use jQuery libraries.      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']] ){        $s =  ' selected="selected" ';      }  **print** "\n      ".'<option value="'.$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>"; | 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.  - 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.    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_43_24.png |
| <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            <button style="float: right;" onclick="formSubmit('acls');">Apply ACLs</button>          </td>        </tr>      </table> | A checkbox for applying the ACLs recursively is added to the form. |
| </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:    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_43_24.png |
| <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.    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_43_24.png |
| </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.    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_46_39.png |
| <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" ){              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.");        }      } | 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 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).    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_02_57.png    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);  **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;      } | 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 “Domain” but these values are suggested by the “include/get\_select\_domains.php” PHP script, based on the folder name.    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_00_17.png |
| **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.    C:\Users\Nicolas Bondier\Pictures\Captures\Screenshot - 2013-12-18 , 16_09_23.png |
| <?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" ) ) );  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" ) ) );    **?>** | The general variables are explicit enough. I used them for retrieving information across many files, but also for easily move the project path. |

## 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($t\_path)."',this); return false;\" >"."[+]"."</a>";      } **else** {  **print** "<span style=\"font-weight:normal;font-style:normal;font-family:monospace; text-decoration: none; color: #000000;\">&nbsp;&nbsp;&nbsp;</span>";      }  **print** "<img src=\"styles/images/folder.png\" alt=\"\" height=\"16\" width=\"16\">        <a style=\"font-weight:normal;color:#000000;\" class=\"folder-no-perms\"  href=\"#\" onclick=\"open\_control\_window('folder\_control\_window.php?path=".urlencode($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);    }  } | This function display the folder tree structure.    The “$base\_path” argument is the path of the folder that will be opened.    The “list\_directory()” function return the list of subfolders.    Then, for each of the subfolders, we get the list of ACLs, an id for the html elements, and the html code of the receiving mail addresses for the folders.    We then create a “<li></li>” element for the folder.    If it is not empty (tested by “empty\_directory()” function) , we add the “[+]” notation at the beginning of the line and a  function for opening/closing the folder on a “onclick” action.    Then we add the folder image and the directory name. The directory name has has a link with a JavaScript action for opening the management window.    Finally, we and the spans with the receiving mailboxes and ACLs short description |
| **function** directory\_incoming\_addresses\_short($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**();    $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** $array;  } | Same function as above, but it returns only the array of email addresses for one directory. |
| **function** save\_incoming\_adresses($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/' && $val != 'tmp/'){  **return** **false**;        }      }    }  **return** **true**;  } | Test if a directory has subfolders. For the Maildir format, we must exclude the regular “new”, “cur”, and “tmp” subfolders. |
| **function** acl\_short\_list($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] = $perms;        }      }    }  **if**( !correct\_perms($arr) ){      error\_log("perms are not correct ! reseting\_file ...");  **return** reset\_file\_perms($path);    }  **return** $arr;  }  **return** $arr;  } | Function for getting the ACLs from a “dovecot-acl” file.    For each line, it stores the permission in a hash with this structure:    “Array[type][name] = permission”    Then, it pass the array to the “correct\_perms()” function that makes verifications and replacement on the array.    We verify it the returned array is equal to the one passed as argument.    If not, it means that something were wrong and we replace all the permissions with default ones. |
| **function** correct\_perms($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'];    $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.'"', $retval); | This is the main function for setting the ACLs.    We define variables.            Here is the command to get all files in folder. We do not get the “dovecot.index.log” file, because dovecot bug when we try to midify anything on it. It cause lock problems. The “dovecot-acl” file must be readable by dovecot only. We change this better. |
| *###########################################*  *# Control default ACLs :*  *#*      **if**( !isset($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);      }    }      $cmd = "sudo chown -R ".SYSTEM\_SHARED\_MAIL\_USER.":".SYSTEM\_SHARED\_MAIL\_USER." '".$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); | 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”.    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) {    *###########################################*  *# 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); | Here, we begin to read all ACLs passed as parameters.    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'";            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);              $cmd = "sudo setfacl -R -m o:rwx '".$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);            }        } | 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”.          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. |
| *# 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 '".$folder."'";            system($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, $retval);            }        }      }    } | Unix permission for users are set the same way as for “anyone”, except we define the name of the user. |
| **if**( isset($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);              $cmd = "sudo setfacl -R -m g:".$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);  } | Unix permission for groups are set the same way as for “anyone”, except we define the name of the group. |
| **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) {      $array = recursive\_path\_and\_id( $path . "/" . $dir, $array );    }  **return** $array;  } | This function returns an array of all subfolders of the folder passed as path in the email storage directory.    It is recursive and can retrieve all branches of the tree. |
| **function** is\_in\_group($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>      </body>    </html>    ";  **exit**;    }  } | 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. |
| **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();    $path = '';  **if** ( isset( $\_GET['path'] ) ){    $path = $\_GET['path'];    $path = urldecode($path);  }  $html = acl\_short\_list($path);  **print** " | ".$html.""; | The page returning the summary of ACLs short list.    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";  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); | This page return a JSON array with the html ids of the html elements to 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 => $value) {  **if** ($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";    }  } | 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.            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**();    $p3 = **array**();    **if**(isset($\_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\">$value</option>\n";      }    } | The PHP script return the option list of the available users to set permissions to.    First, we get all users that are present in the array “$\_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 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);  }    $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> | 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.                    Then we call the function to display subdirectories of the folder with the full system path.        Some debug. |
| <?php } **?>** |  |

# 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/>

\* \* \*



Copyright © 2014 by Switzernet