**#16832**-<https://redmine.vnc.biz/issues/16832>

1. **Description**:

Requirement name: zimlet chat user should be able to access password secured chatroom

Description: If a zimlet chat user tries to access a chat room, hidden or not, which is password secure, then the zimlet should prompt an input window and request the user to enter the password to access the chatroom. This chatroom may be created in pidgin or another client, and should not be a problem to fulfill this requirement. How the user gets the password is not of our concern (e.g. telephone or email)

Origin: Rainer

When was this requirement proposed? 27.02.2015

Requirement designed by: Rainer

When was this requirement designed? 27.02.2015

Rationale: User should be able to additionally access password secure chatroom within the XMPP specification. The vnctalk zimlet should be compliant to the XMPP standard, as far as possible.

Difficulty: normal.

Customer satisfaction: the customer will be very satisfied having this feature. If it does not work will be very bad.

Relations:

Priority: normal

Validation: Create a password secure chatroom with pidgin, try to access it from the zimlet chat. Enter the wrong password, the user should not be able to access the chatroom and receive an error message. If the user enters the right password, he should be able to access the chatroom.

The XMPP standard describes, that there is a mechanism, or at least possibility, to (automatically) share the password of a secured chatroom, when the user is invited to the room.

This should be analyzed and also implemented. If possible it should be transparent to the user if the chatroom is secured or not.

1. **Analytic**
   1. **Root cause:**

Zimlet and pidgin no support create password room, so when a room (although in pidgin can can create room with password,it don’t work) was created, everyone can join room without password.

* 1. **Solution:**

+Add password field in create room popup.

+Add config #roomconfig\_roomsecret in create room IQ stanza.

+User can join secret room with right password.

1. **Implement**

|  |
| --- |
| ~/VNCtalk/prosody-packaging/mini/javascripts/mini.js  …  Add function:  self.openPromptCreateGroup = function(text) {  try {  // Initialize  var prompt = '#jappix\_popup div.jm\_prompt';  var input = prompt + ' form input';  var value\_input = input + '[type="text"]';  // Remove the existing prompt  self.closePrompt();  // Add the prompt  jQuery('body').append(  '<div id="jappix\_popup" dir="' + (JappixCommon.isRTL() ? 'rtl' : 'ltr') + '">' +  '<div class="jm\_prompt">' +  '<form>' +  text +  '<input id="group\_name" class="jm\_text" type="text" value="" />' +  'Password:' +  '<input id="group\_pass" class="jm\_text" type="text" value="" />' +  '<input class="jm\_submit" type="submit" value="' + JappixCommon.\_e("Submit") + '" />' +  '<input class="jm\_submit" type="reset" value="' + JappixCommon.\_e("Cancel") + '" />' +  '<div class="jm\_clear"></div>' +  '</form>' +  '</div>' +  '</div>'  );  // Vertical center  var vert\_pos = '-' + ((jQuery(prompt).height() / 2) + 10) + 'px';  jQuery(prompt).css('margin-top', vert\_pos);  // Focus on the input  jQuery(document).oneTime(10, function() {  jQuery(value\_input).focus();  });  // Cancel event  jQuery(input + '[type="reset"]').click(function() {  try {  self.closePrompt();  }  catch(e) {}  finally {  return false;  }  });  } catch(e) {  JappixConsole.error('JappixMini.openPrompt', e);  }  }; |
| Config in function self.acceptDefaultRoomConfigMini:  self.acceptDefaultRoomConfigMini =function(roomid,password){  JappixMini.presence('', '', '', '', roomid + '/' + MINI\_USER, hex\_md5(roomid), true, self.handleMUC);  var configIQ = new JSJaCIQ();  configIQ.setTo(roomid);  configIQ.setType("set");  configIQ.setQuery(NS\_MUC\_OWNER);  xEle = configIQ.getDoc().createElementNS("jabber:x:data","x");  xEle.setAttribute("type","submit");  var fieldEle = JappixMini.createField(configIQ,"FORM\_TYPE",["http://jabber.org/protocol/muc#roomconfig"]);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_roomname",[roomid.split('@')[0]]);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_persistentroom",['0']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_publicroom",['0']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_whois",['anyone']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_moderatedroom",['0']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_membersonly",['0']);  xEle.appendChild(fieldEle);  if (password)  {  var fieldEle = JappixMini.createField(configIQ,"muc#muc\_passwordprotected",['1']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_roomsecret",[password]);  xEle.appendChild(fieldEle);  }  configIQ.getQuery().appendChild(xEle);  con.send(configIQ,self.handleAcceptDefault);  } |

|  |
| --- |
| Config in self.groupchatPrompt function:  From  self.groupchatPrompt = function() {  try {  // Create a new prompt  self.openPrompt(biz\_zcs\_vnc\_talk\_zimlet.vnc\_talk\_room\_creat\_lable)  // When prompt submitted  jQuery('#jappix\_popup div.jm\_prompt form').submit(function() {  try {  // Read the value  var join\_this = self.closePrompt();  var pattern = new RegExp(/[<>&/'"@: ]/);  var pattern1 = new RegExp(/^\s\*$/);  if(pattern.test(join\_this) || pattern1.test(join\_this)){  appCtxt.setStatusMsg(biz\_zcs\_vnc\_talk\_zimlet.vnc\_talk\_invalid\_room\_error);  JappixMini.groupchatPrompt();  $('#jappix\_popup div.jm\_prompt form input[type="text"]').val(join\_this);  }else {  // Get the chat room to join  var chat\_room = join\_this.toLowerCase() + "@conference." + MINI\_DOMAIN;  JappixMini.acceptDefaultRoomConfigMini(chat\_room);  }    TO  self.groupchatPrompt = function() {  try {  // Create a new prompt  self.openPromptCreateGroup(biz\_zcs\_vnc\_talk\_zimlet.vnc\_talk\_room\_creat\_lable);  // When prompt submitted  jQuery('#jappix\_popup div.jm\_prompt form').submit(function() {  try {  // Read the value  var join\_this = jQuery('#jappix\_popup div.jm\_prompt form').find('#group\_name').val();  var password = jQuery('#jappix\_popup div.jm\_prompt form').find('#group\_pass').val();  // Remove the popup  jQuery('#jappix\_popup').remove();  var pattern = new RegExp(/[<>&/'"@: ]/);  var pattern1 = new RegExp(/^\s\*$/);    if(pattern.test(join\_this) || pattern1.test(join\_this)){  appCtxt.setStatusMsg(biz\_zcs\_vnc\_talk\_zimlet.vnc\_talk\_invalid\_room\_error);  JappixMini.groupchatPrompt();  $('#jappix\_popup div.jm\_prompt form input[type="text"]').val(join\_this);  }else {  // Get the chat room to join  var chat\_room = join\_this.toLowerCase() + "@conference." + MINI\_DOMAIN;  JappixMini.acceptDefaultRoomConfigMini(chat\_room,password);  } |
| Add request :Public room for room secret:  Config in function self.acceptDefaultRoomConfigMini:  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_publicroom",['1']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#muc\_passwordprotected",['1']);  xEle.appendChild(fieldEle);  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_roomsecret",[password]);  xEle.appendChild(fieldEle);  } else {  var fieldEle = JappixMini.createField(configIQ,"muc#roomconfig\_publicroom",['0']);  xEle.appendChild(fieldEle);  } |