Kolab Selbst-Registrierung

Kolab bringt ein simples Selbstregistrierungssystem mit. Unter einer bestimmten URL können User_innen somit sich selbst einen Account auf einem Kolab Server registrieren.

Für die Konfiguration sind insgesamt folgende Schritte nötig:

1. Konfiguration der Apache-Konfiguration
2. Anlegen eines Ldap-Users, der über die entsprechenden Rechte verfügt, die Registrierung vorzunehmen
3. Anlegen eines Kolab User-Types, der das Attribut „hosted“ besitzt. Entweder per Skript oder im Kolab-Webadmin
4. Konfiguration in der kolab.conf
5. ggf. Anpassungen am Webinterface, zusätzliche Sicherheitsmaßnahmen gegen Mißbrauch etc.

Das folgende Howto wurde vor dem Hintergrund einer Kolab 3.4 Installation auf Debian Wheezy geschrieben, sollte aber distributionsunabhängig gültig sein.

Konfiguration

Apache: Enable hosted-kolab.conf

Eine Apache Konfiguration für die Selbstregistrierung ist unter /etc/kolab-webadmin/ bereits vorkonfiguriert. Wir aktivieren sie mit:

```
ln -s /etc/kolab-webadmin/hosted-kolab.conf /etc/apache2/sites-enabled
service apache2 reload
```

Das war's eingentlich schon. Wenn das Template für die Registrierungsmaske angepasst weden soll:

```
/usr/share/kolab-webadmin/public_html/skins/default/templates
```

Evlt. noch die Sprach-Lokalisierungen unter:

```
/usr/share/kolab-webadmin/lib/locale/de_DE.php
```

prüfen (→ hier nach signup suchen ...)

Einrichten des service-users und der ACIs

Damit die registrierten Account-Daten ins Ldap-Verzeichnis geschrieben werden können, muss ein Ldap-User existieren, der dazu berechtigt ist. Der Kolab-Service User hat dafür nicht ausreichende

Rechte.

Evtl. soll die Registrierungsfunktion in einem Hosted-Setup nicht für alle Domains gleichermaßen möglich sein. Dieses Beispiel geht davon aus, dass es eine primäre Domain gibt, für die die Registrierung nicht möglich sein soll. Ggf. können noch weitere Domains verborgen werden. Alternativ kann die Registrierung auch nur für einzelne Domains erlaubt werden.

Grundsätzlich funktioniert die Selbst-Registrierung wenn als `bind_dn` der Directory Manager konfiguriert ist. Dies kann zum Testen in einem nicht-produktiven Setup auch verwendet werden. Damit können Fehler eingegrenzt werden, weil erst mal davon ausgegangen werden kann, dass Probleme nicht an fehlenden Ldap-ACIs liegen. Im Produktivbetrieb wird das aber aus verschiedenen Gründen nicht gewünscht sein:

1. Sicherheitsgründe: dieser User hat Zugriff auf den gesamten Ldap-Baum,
2. in diesem Fall wären alle konfigurierten Domains auch für die Selbstregistrierung freigegeben.

Wir benötigen also einen User mit begrenzten Rechten, der

1. lesenden Zugriff auf die Liste der Domains hat, für die Selbstregistrierung möglich sein soll (aci: read,search,compare)
2. schreibenden Zugriff auf die Datenbank der jeweiligen Domains hat (aci: all)

Diese Einträge können mit folgendem Code hinzugefügt werden, wobei:

- regdomain.com die Domain sein soll, für die User sich registrieren können sollen
- hosted-kolab-serice der Bind-User sein soll
- und example.com die primäre domain der Kolab-Umgebung ist.
- und anotherdomain.com eine weitere Domain ist, für die die Registrierung möglich sein soll.

```
# im folgenden Kommando nutzen wir "head" statt "tail" zum ausschneiden des
Passwortes aus der kolab.conf,
# weil wir nun zwei "bind_dn"-Einträge in der kolab.conf haben. Wir wollen
den aus der [kolab_wap] Sektion,
# die vor der [kolab_hosting] Sektion steht.
/usr/lib/mozldap/ldapmodify   -h localhost -p 389 -D "cn=directory manager"
-w "$(grep ^bind_pw /etc/kolab/kolab.conf | cut -d ' ' -f3- | head -1)"

# Anlegen des neuen Nutzers:
dn: uid=hosted-kolab-service,ou=Special Users,dc=example,dc=com
changetype: add
objectclass: top
objectclass: inetorgperson
objectclass: person
uid: hosted-kolab-service
cn: Hosted Kolab Service Account
sn: Service Account
givenname: Hosted Kolab
userpassword: SECRET_PASSPHRASE

# dieser Nutzer benötigt das Recht, die Liste der Domains abfragen zu
können:
dn: cn=kolab,cn=config
```

```
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Hosted Kolab Services";allow
  (read,compare,search)(userdn = "ldap:///uid=hosted-kolab-service,ou=
  Special Users,dc=example,dc=com");)
```

```
# Wahrscheinlich soll mindestens die Primäre Kolab-Domains verborgen werden:
dn: associateddomain=example.com,cn=kolab,cn=config
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Hosted Kolab Services";deny
  (read,search)(userdn = "ldap:///uid=hosted-kolab-service,ou=Special
  Users,dc=example,dc=com");)

# Diesen Schritt für alle Domains wiederholen, die nicht sichbar sein sollen
...

# Und jetzt noch das Recht, innerhalb von regdomain.com auch neue Einträge
hinzufügen zu können:
dn: dc=regdomain,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Hosted Kolab Services";allow
  (read,search,compare,add,write,selfwrite,delete)(userdn =
"ldap:///uid=hosted-kolab-service,ou=
  Special Users,dc=example,dc=com");)
```

Alternativ: Ldap Targetfilter nutzen um Domains zu nutzen, die registrierbar sein sollen

Gerade bei mehreren Domains ist es sehr umständlich und fehleranfällig, alle Domains zu verstecken, die nicht registrierbar sein sollen. Außerdem kann das erst nach dem Anlegen der Domain erfolgen. Ist also ein Setup geplant, in dem üblicherweise neue Domains nicht als „öffentliche“ Domains verfügbar sein sollen, bietet sich ein Ldap „target-filter“ an. Statt den neuen Nutzer also pauschal alle Domains in `cn=kolab,cn=config` sehen zu lassen, setzen wir einen Target-Filter auf genau die Domains, die wir als Registrierungsdomains zulassen wollen.

```
# allow service user to search some domains
dn: cn=kolab,cn=config
changetype: modify
add: aci
aci: (targetattr = "*")
(targetfilter=(|(associateddomain=regdomain.com)(associateddomain=anotherdom
ain.com))) (version 3.0;acl "Kolab Registration Service";allow (r
 ead,compare,search)(userdn = "ldap:///uid=hosted-kolab-service,ou= Special
U
 sers,dc=example,dc=com");)
```

Einfachere Variante für nur eine Domain

Wenn es nur eine Domain gibt, für die Selbstregistrierung möglich sein soll, geht es noch etwas einfacher: Wir legen als Bind-User einen User innerhalb der in Frage kommenden Domain an und setzen für diesen die Rolle „kolab-admin“.

In diesem Fall muss kein extra Nutzer mit spezifischen ACLs angelegt werden. Es reichen die Modifikationen der aci in

```
dn: cn=kolab,cn=config
```

und das Verbergen der anderen Domains, die sonst trotzdem zur Auswahl stünden, auch wenn ein Eintrag darin fehlschlagen würde (Beides ist oben erläutert).

Nutzer-Typ hinzufügen

In einem normalen Kolab-Setup sind bereits verschiedene Kolab Nutzertypen definiert - z.B. Kolab User, Posix User etc. - auszuwählen im Kolab-Webadmin.

Für die Selbstregistrierung benötigen wir einen Nutzer-Typ, der diese Fähigkeiten besitzt - dazu muss er einen Eintrag `hosted` in der mysql-Datenbank erhalten.

Leider klappt das aktuell nicht über das kolab-webadmin Frontend. Dort gibt's zwar unter Einstellungen bei den Nutzertypen einen Button für „Hosted“, aber das scheint ein Bug zu sein, dass das nicht in der mysql-Tabelle landet. [1)]

Es kann aber per Hand konfiguriert werden, z.B.:

```
mysql> update user_types set used_for="hosted" where `key`="kolab";
```

Der Nutzertyp „kolab“ eignet sich allerdings nicht für die Selbst-Registrierung, weil dort zu viele Felder, die von der Nutzer_in nicht selbst ausgefüllt werden sollen, vorhanden sind. Statt dessen sollte ein eigener Nutzertyp erstellt werden. Von Seiten der Kolab-Entwickler gibt es dazu auch schon einen Vorschlag, an dem man sich orientieren kann.

Dort sind die nötigen Felder auf ein Minimum reduziert - und die primäre E-Mail-Adresse wird von der ldap-UID abgeleitet, was vom Namen der Nutzer_innen unabhängige E-Mail-Adressen zulässt.

Er lässt sich mit dem Skript

```
sample-insert-hosted-user_types.php
```

unter /usr/share/doc/kolab-webadmin/ erstellen. Achtung: in den ersten Zeilen steht ein TRUNCATE Befehl, der die DB sonst platt macht. Besser ist es, diese Zeile auszukommentieren - dann kommen die Nutzertypen zusätzlich dazu …

Damit das php Skript die richtigen libs findet, muss es nach /usr/share/kolab-webadmin kopiert werden.

Leider lassen sich Nutzer-Typen nicht nachträglich per Webinterface ändern. Die Frage, welchem

Nutzertyp ein Eintrag zugeordnet wird, hängt von den vorhandenen Ldap-Attributen ab. Die Kolab-Api sucht wohl den Nutzertyp heraus, der alle Attribute enthält - aber möglichst wenige nicht vergebene.

Ein Workaraound wäre, dass sich neue NutzerInnen unter einem Nutzeraccount mit weniger Attributen registrieren - und weitere Ldap-Attribute per Skript hinzugefüg werden, sobald ein Admin die Registrierung bestätigt.

kolab.conf

Jetzt kommen wir zur Konfiguration in der kolab.conf. Hier müssen die primäre Domain sowie der User konfiguriert werden, mit dem die Registrierung erfolgen soll.

Zu den Konfigurationsmöglichkeiten in der kolab.conf siehe:

- http://git.kolab.org/pykolab/tree/conf/kolab.conf

Dazu legen wir eine neue Sektion mit dem Titel `[kolab_hosting]` an:

```
[kolab_hosting]
;; Set the default domain name space for the list of domain name spaces (if
more
;; than one) that new users that register are allowed to select.
primary_domain = example.org
;
;; The following bind credentials should be allowed to search
;; "ldap/domain_base_dn" (i.e. cn=kolab,cn=config), but should not be
allowed to
;; read any domain name space LDAP entry that users are not eligible to
select.
;;
;; Note that the bind credentials usually live in the upper
;; "kolab/primary_domain".
bind_dn = uid=hosted-kolab-service,ou=Special Users,dc=example,dc=com
bind_pw = secret
;recaptcha_private_key = bla
;recaptcha_public_key = bla
```

- als bind_dn muss ein user eingerichtet werden, der entsprechende rechte (ACIs) hat (s.o.)
- für das captcha-plugin wird ein google-account benötigt, oder es wird auskommentiert.[2)]
- das attribut: `primary_domain`: wenn mehrere Domains zur Auswahl stehen: welches soll die default-Domain sein? (Funktionalität unklar)

Weitere Informationen:

- http://docs.kolab.org/architecture-and-design/kolab-wap-api.html
- http://docs.kolab.org/howtos/multi-domain.html (erster Absatz)

Testen

Jetzt sollte beim Aufrufen des Registrierungsformulars eine Registrierung möglich sein.

Fehlersuche:

- Wird das Formular korrekt aufgerufen aber es lassen sich keine Domains zum Registrieren auswählen: → wahrscheinlich gibt es ein Problem mit den Ldap-ACIs.
- Das Formular hat keinen Inhalt: Es existiert kein User-Typ mit dem Attribut „hosted“
- …

Anpassung des Registrierungs-Interfaces

Weitere Anpassungen der Registrierungsformulars lassen sich im Code von Kolab-Webadmin vornehmen. Dazu einige Anregungen:

Attribute in der Registrierungsmaske verbergen

Evtl. ist es gewünscht, dass nicht alle Felder/Attribute, die der entsprechende User-Typ im Ldap-Verzeichnis besitzt auch im Registrierungsformular angezeigt werden.

Eine einfache Variante ist, die Felder, die in der Registrierungsmaske nicht angezeigt werden sollen, innerhalb der Datei /usr/share/kolab-webadmin/lib/client/kolab_client_task_signup.php zu deklarieren.

```
        // Hide cn field
        if (isset($fields['dkknowncloudquota'])) {
            // TODO add "hidden":true to user_types attributes and use it
            $fields['dkknowncloudquota']['type'] = kolab_form::INPUT_HIDDEN;
        }
```

Reihenfolge der Felder in der Registrierungsmakse

In der Datei /usr/share/kolab-webadmin/lib/client/kolab_client_task_signup.php kann auch die Reigenfolge der Felder in der Registrierungsmaske angegeben werden:

```
private function user_form($data = array())
    {
        $attribs['id'] = 'signup-form';

        $fields_map = array(
            'type_id'                   => 'other',
            'givenname'                 => 'other',
            'sn'                        => 'other',
            'cn'                        => 'other',
            // .... weitere Attribute
```

```
    }
```

Zusätzliche Felder in der Registrierungsmaske

Zusätzliche Felder im Signup Form lassen sich in der Datei usr/share/kolab-webadmin/lib/client/kolab_client_task_signup.php ebenfalls definieren. Dabei können wir auf einen Hack zurückgreifen, der auch für die Einbindung des Google Captchas verwendet wird:

Dazu wird das fertige Formular quasi noch einmal umgeschrieben (nicht sehr elegant):

```
$form = preg_replace('/<\/tbody>/', '<tr><td
class="label">Zahlungsweise</td><td class="value"><input type="text"
name="payment" class="maxsize" /></td></tr></tbody>', $form);
```

Und eine weitere Formularzeile eingefügt.

Diese Felder können wir nun weiter verwenden:

- entsprechen Sie einem Ldap-Attribut, werden Sie ins Verzeichnis mit eingefügt
- ansonsten können wir sie zum Validieren von Informationen verwenden (z.B. Akzeptieren von AGBs), die nicht weiter verarbeitet werden.
- Sie stehen aber auch als PHP-Variable weiterhin zur Verfügung und können z.B. mit in die E-Mail integriert werden, die bei der Registrierung an eine Adresse geschickt wird.

Mailversand nach Anlegen eines neuen Nutzers

In der kolab.conf unter

```
[kolab_hosting]
;...
mail_address = mail@example.org
send_signup_mail = true
;...
```

Der Mailversand selbst wird in der Funktion `private function send_mail($data)` bewerkstelligt. Hier lassen sich leicht zusätzliche Felder einfügen.

Ausführen von weiteren Prozeduren nach dem Anlegen des Users

Um nach dem Anlegen des Nutzers weitere Prozeduren auszuführen, können kann evtl. analog zur folgenden Funktion in /usr/share/kolab-webadmin/lib/client/kolab_client_task_signup.php vorgegangen werden:

```
 private function add_to_openerp($data)
    {
```

```
    //
    }
```

Dies wird nach erfolgreichem Anlegen des Nutzers in der Funktion

```
  public function action_add_user(){
  }
```

ausgeführt. Dort lassen sich noch weitere Funktionen unterbringen.

1)

ab Kolab 3.4 funktioniert das

2)

Weil wir ungern auf die Google-Dienste zurück greifen wollen, haben wir einen anderen Weg genutzt und das Registrierungsformular geändert und dort ein Feld eingebaut, das einen Honeypot darstellt