Polkit

Introducció general

En general, hi ha dues maneres de fer per un procés, segons com s'hagi programat: o bé ordenant directament al kernel que faci allò que calgui en cada moment (per exemple, obrir un fitxer, establir una connexió de xarxa, etc) o bé utilitzant algun mecanisme IPC per demanar a un servei intermediari que "reenviï" ell les ordres necessàries al kernel. En el primer cas intervenen aspectes com els permisos i ACLs dels fitxers, les capabilities dels binaris o els sistemes M.A.C com ara AppArmor o SELinux (que estudiarem seguidament). En el segon cas, un servei molt habitual dissenyat per fer aquestes tasques d'"intermediació" és el servei Polkit (*systemctl status polkit*), el qual utilitza el canal D-Bus del sistema per rebre les peticions dels diferents processos i reenviar-les al kernel adientment.

¿Per què cal el Polkit si amb l'accés directe al kernel és tot més senzill? Perquè les comprovacions de seguretat que pot fer el kernel no són suficients per expressar tota la quantitat possible de polítiques que poden existir en un sistema d'escriptori o mòbil. Per exemple: si hom enxufa un pendrive USB a una màquina, el més raonable és que es munti automàticament i aparegui la icona corresponent a l'escriptori de l'usuari; no obstant, demanar al kernel que munti un pendrive USB és exactament el mateix que demanar que munti qualsevol altre tipus de sistema de fitxers: si qualsevol usuari pogués començar a (des)muntar particions sense haver de ser "root", apareixeria el caos. L'ideal seria fer que l'usuari pogués muntar sistemes de fitxers (una acció privilegiada perquè es necessita cridar al kernel) però només sota determinades condicions. Cal, per tant, un sistema que "filtri" les peticions al kernel fetes pels diferents programes d'usuari segons algunes de les característiques concretes d'aquestes peticions i decidir, a partir d'aquí, si es permet convertir-les en peticions privilegiades (reenviant-les al kernel o a algun programa/dimoni en comunicació directa amb aquest) o descartar-les. Això és el que fa el servei Polkit.

Altres exemples on és interessant l'ús de Polkit és, per exemple, a l'hora de reconfigurar/(des)activar una tarja de xarxa, instal.lar/actualitzar/desinstal.lar paquets, apagar/reinicia/suspendre/hibernar el sistema...

Introducció Polkit

Polkit és un "framework" que proporciona als desenvolupadors una API d'autorització (es pot consultar aquí: https://www.freedesktop.org/software/polkit/docs/latest/PolkitAuthority.html) mitjançant la qual els programes privilegiats (també anomenats "mecanismes") -que són els que poden parlar directament amb el kernel- poden oferir determinats serveis als programes no privilegiats (també anomenats "subjectes"). Actua com un "negociador" entre la sessió d'usuari sense privilegis i el context privilegiat del sistema: quan un procés sense privilegis de la sessió d'usuari vol realitzar una acció en el context del sistema, Polkit és invocat i, basant-se en la seva configuració (especificada en les anomenades "polítiques") la resposta pot ser "sí", "no" o "necesita autenticació".

En contrast amb programes com sudo, Polkit no ofereix permissos d'administrador per una sessió sencera sinò només per l'acció demanada depenent de l'usuari o grup que l'hagi sol.licitat i/o de moltes altres característiques de la petició

Accions

La configuració de Polkit depén de dues coses: la definició d'accions i de les regles d'autorització per aquestes accions. Les accions són arxius XML amb extensió *.policy que estan ubicats dins de la carpeta "/usr/share/polkit-1/actions". Cada fitxer pot definir una o més accions i cada acció conté descripcions i **permisos per defecte per ella mateixa**. L'administrador pot escriure les seves pròpies accions en arxius *.policy propis escrits a part però les que venen per defecte amb la distribució de Polkit (o les que s'instal.len a partir de paquets de tercers) no s'han de modificar mai perquè podrien ser sobreescrits per alguna actualització del sistema (o del paquet associat al programa responsable de les accions descrites en aquests arxius, respectivament).

Tot seguit es mostra un exemple d'arxiu *.policy (instal.lat pel paquet Gparted) que fa una sola acció:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policyconfig PUBLIC "-//freedesktop//DTD PolicyKit Policy Configuration 1.0//EN"
"http://www.freedesktop.org/standards/PolicyKit/1.0/policyconfig.dtd">
<policyconfig>
        <action id="org.opensuse.policykit.gparted">
                <message>Authentication is required to run the GParted Partition Editor</message>
                <icon_name>gparted</icon_name>
                <defaults>
                        <allow_any>auth_admin</allow_any>
                        <allow_inactive>auth_admin</allow_inactive>
                        < allow_active>auth_admin</allow_active>
                </defaults>
                <annotate key="org.freedesktop.policykit.exec.path">/usr/sbin/gparted</annotate>
                <annotate key="org.freedesktop.policykit.exec.allow_gui">true</annotate>
        </action>
</policyconfig>
```

Comentaris a l'exemple anterior:

*L'atribut "id" de l'etiqueta "action" és l'identificador de l'acció rebuda a través de D-Bus que es tractarà de filtrar. Aquesta acció l'haurà demanada un programa no privilegiat (en aquest cas, Gparted) i pot ser una qualsevol de les definides pel propi programa.

NOTA: Per exemple, el programa "udisks2" actualment implementa vàries accions totes elles relacionades amb el muntatge d'unitats però cadascuna amb detalls diferents: "org.freedesktop.udisks2.filesystem-mount", "org.freedesktop.udisks2.filesystem-mount-fstab", "org.freedesktop.udisks2.filesystem-mount-system" o "org.freedesktop.udisks2.filesystem-mount-other-seat" Per saber el significat concret de cada acció cal consultar la documentació pròpia de cada programa; per exemple, a http://storaged.org/doc/udisks2-api/2.7.2/udisks-polkit-actions.html està la d'Udisks2

*L'etiqueta "message" és una explicació que es mostrarà a l'usuari quan es necessiti autenticar.

*L'etiqueta "icon_name" és òbvia

*L'element "defaults" conté diferents tipus de permisos però a la pràctica, l'únic que s'aplica sol ser l'indicat dins l'etiqueta "allow_active", ja que aquesta etiqueta es refereix a les sessions d'usuari que estan actives en aquest moment. Els valors possibles dels permisos que es poden escriure dins d'aquesta etiqueta (o la resta d'etiquetes possibles, que són "allow_inactive" i "allow_any") són:

yes: L'usuari està autoritzat a realitzar l'acció sense cap autenticació
no: L'usuari no està autoritzat a realitzar l'acció (no cal, per tant, autenticació)
auth_self: Es necessita autenticar-se com l'usuari actual però no cal que sigui administrador
auth_admin: Es necessita autenticar-se com un usuari administrador (root)
auth_self_keep: Igual que *auth_self* però l'autenticació dura uns minuts en memòria
auth_admin_keep: Igual que *auth_admin* però " " " " " "

Aquests valors són generals per tots els usuaris i s'aplicaran per defecte a no ser que hi hagi alguna regla d'autenticació (veure més endavant) que els contradigui

*L'etiqueta "annotate" conté informació específica sobre com Polkit realitza l'acció. En aquest cas conté la ruta de l'executable i indica si es permet que aquest executable mostri una GUI

NOTA: Per més informació, llegir https://www.freedesktop.org/software/polkit/docs/latest/polkit.8.html (apartat "Declaring actions").

Regles d'autorització

Les regles d'autorització, d'altra banda, a partir de la versió 0.106 de Polkit són fitxers Javascript amb extensió *.rules que poden estar ubicats en dos llocs: a /usr/share/polkit-1/rules.d es troben les regles instal.lades pels paquets de tercers (que servirien per sobreescriue els permisos per defecte que apareixen als arxius *.policy corresponents i que no s'haurien de tocar, en principi) i a /etc/polkit-1/rules.d s'ubiquen les regles que l'administrador del sistema podrà editar.

Polkit llegeix totes les regles d'ambdues carpetes indistintament en ordre numericoalfabètic i si troba alguna regla que quadra amb les característiques de la petició demanada, s'aplica i es deixa de llegir les regles següents (és a dir, com passa també amb les regles Iptables). Per tant, interessarà anomenar als fitxers de regles pròpies amb un prefixe numèric baix. Si hi ha alguna fitxer *.rules anomenat igual tant dins la carpeta /etc com a la carpeta /usr, primer es processa el que està a /etc i després, si no s'ha trobat cap regla coincident, el que està a /usr. En resum, l'ordre seria el següent:

1- "/etc/polkit-1/rules.d/10-auth.rules"
2- "/usr/share/polkit-1/rules.d/10-auth.rules"
3- "/etc/polkit-1/rules.d/15-auth.rules"
4- "/usr/share/polkit-1/rules.d/20-auth.rules"

Si es modifica/afegeix/esborra algun arxiu *.rules, automàticament Polkit ho tindrà en compte ja que contínuament està monitoritzant les carpetes on es troben aquests arxius (és a dir, no cal reiniciar el servei)

Cada regla es refereix a una acció indicada a un arxiu *.policy i sobrescriu els permisos per defecte que allà s'hagin indicat per tothom. De fet, una regla el que fa és determinar quines restriccions concretes estan permeses per un subconjunt d'usuaris concret.

A continuació es presenta un exemple d'arxiu *.rules , el qual permet montar un sistema de fitxers pels usuaris que pertanyin al grup "storage" sense cap tipus de procés d'autenticació:

```
polkit.addRule(function(action, subject) {
        if (action.id == "org.freedesktop.udisks2.filesystem-mount-system" && subject.isInGroup("storage")) {
                return polkit.Result.YES;
        }
});
```

Un altre exemple que permet utilitzar el programa Gparted sense autenticar a tots els membres del grup d'usuaris "admin":

```
polkit.addRule(function(action, subject) {
        if (action.id == "org.opensuse.policykit.gparted" && subject.isInGroup("admin")) {
                return polkit.Result.YES;
        }
});
```

Comentaris als exemples anteriors:

*L'objecte "action" només té l'atribut "id" (corresponent a l'identificador de l'acció indicada en algun arxiu *.policy) i el mètode lookup(), el qual té com a únic paràmetre -de tipus string- el nom d'alguna variable concreta que el programa privilegiat invocat (el mecanisme) pugui publicar per aquella acció i com a valor de retorn el valor -de tipus string- obtingut d'aquesta variable. Cal consultar la documentació de cada mecanisme per saber quines variables estan disponibles.

*L'objecte "subject" té uns quants atributs, els més importants dels quals són...:

pid : El PID del procés no privilegiat que demana permís per realitzar l'acció
user : L'usuari que invoca el procés no privilegiat que demana permís per realitzar l'acció
groups : Array de cadenes que representen els grups als quals l'usuari pertany

...i el següent mètode:

isInGroup() : Com a paràmetre admet el nom d'un grup d'usuaris i retorna un valor booleà segons si subject.user està en el grup indicat (1) o no (0)

*Els valors possibles de retorn de la funció anònima poden ser *null* (llavors és com si no s'hagués llegit res del codi de la funció o bé els següents, els quals són equivalents al ja estudiats a l'apartat anterior pels permisos per defecte:

polkit.Result.YES
polkit.Result.NO
polkit.Result.AUTH_SELF
polkit.Result.AUTH_ADMIN
polkit.Result.AUTH_SELF_KEEP
polkit.Result.AUTH_ADMIN_KEEP

Per més informació, llegir: https://www.freedesktop.org/software/polkit/docs/latest/polkit.8.html (apartat "Authoritzation Rules")

Comandes de terminal oferides per Polkit

pkaction: Mostra la llista de totes les accions definides als arxius *.policy
pkaction -v -a nomAccio : Mostra la informació guardada al seu arxiu *.policy de l'acció indicada
pkaction --verbose : Similar a l'anterior però per totes les accions definides a tots els arxius *.policy

pkexec -u usuari programa: Permet a un usuari executar un programa com un altre usuari (si no s'indica, l'usuari serà "root"). Similar, doncs, a *sudo* o *su -c* però fent servir Polkit (via D-Bus). La gràcia d'aquesta comanda és que permet utilitzar Polkit (amb les regles implícites o bé les que s'especifiquin) també amb qualsevol programa encara que no estigui dissenyat per parlar amb ell, ja que d'aquesta comunicació via D-Bus se n'encarrega precisament *pkexec*

Exemples pràctics

*To access virt-manager without entering password if your user belongs to "wheel" group, just a create a file named /etc/polkit-1/rules.d/10-libvirt-manage.rules (or similar) with following content ("org.libvirt.unix.policy" file is provided by "libvirt-bin" package):

```
polkit.addRule(function(action, subject) {
  if (action.id == "org.libvirt.unix.manage" && subject.isInGroup("wheel")) {
     return polkit.Result.YES;
  }
});
```

---

*To disable suspend and hibernate for all users, just create a file named /etc/polkit-1/rules.d/10-disable-suspend.rules (or similar) with following content ("org.freedesktop.login1.policy" file is provided by "systemd" package):

```
polkit.addRule(function(action, subject) {
   if (action.id == "org.freedesktop.login1.suspend" ||
       action.id == "org.freedesktop.login1.suspend-multiple-sessions" ||
       action.id == "org.freedesktop.login1.hibernate" ||
       action.id == "org.freedesktop.login1.hibernate-multiple-sessions")
   {
       return polkit.Result.NO;
   }
});
```

In order to allow "wheel" group members to shut down or reboot the system without further authentication (even if an application has asked for those actions to be delayed until it is finished and even if there are others logged in) while everyone else will be asked for the root password, just write this:

```
polkit.addRule(function(action, subject) {
   if (action.id == "org.freedesktop.login1.reboot" ||
      action.id == "org.freedesktop.login1.reboot-ignore-inhibit" ||
      action.id == "org.freedesktop.login1.reboot-multiple-sessions" ||
      action.id == "org.freedesktop.login1.power-off" ||
      action.id == "org.freedesktop.login1.power-off-ignore-inhibit" ||
      action.id == "org.freedesktop.login1.power-off-multiple-sessions") {
            if (subject.isInGroup("wheel")) {
                  return polkit.Result.YES;
            } else {
                  return polkit.Result.AUTH_ADMIN;
            }
   }
});
```

---

*To allow all users in the "wheel" group to start, stop, restart and otherwise manage systemd services, just create a file named /etc/polkit-1/rules.d/10-systemd-manage.rules (or similar) with following content ("org.freedesktop.systemd1.policy" file is provided by "systemd" package):

```
polkit.addRule(function(action, subject) {
   if (action.id == "org.freedesktop.systemd1.manage-units") {
      if (subject.isInGroup("wheel")) {
         return polkit.Result.YES;
      }
   }
});
```

If we want to restrict this management to only certain units (for instance *apache2*) and/or only certain verbs (*start*, *stop*, etc), just write this:

```
polkit.addRule(function(action, subject) {
   if (action.id == "org.freedesktop.systemd1.manage-units") {
      if (subject.isInGroup("wheel") && action.lookup("unit") == "apache2.service") {
         var verb = action.lookup("verb");
         if (verb == "start" || verb == "stop" || verb == "restart") {
            return polkit.Result.YES;
         }
      }
   }
});
```

Other interesting Polkit action ids defined in "org.freedesktop.systemd1.policy" file are "org.freedesktop.systemd1.manage-unit-files" (useful to monitor request to manage system services or other unit files) and "org.freedesktop.systemd1.reload-daemon" (useful to monitor request to reload the systemd daemon)

---

*To allow all users in the "wheel" group to perform user administration, just create a file named /etc/polkit-1/rules.d/10-user-management.rules (or similar) with following content ("org.gnome.controlcenter.user-accounts.policy" file is provided by "control-center" package):

```
polkit.addRule(function(action, subject) {
   if (action.id == "org.gnome.controlcenter.user-accounts.administration" &&  subject.isInGroup("wheel")) {
      return polkit.Result.YES;
   }
});
```

---

*To forbid users in group "children" to change hostname configuration (that is, any action with an identifier starting with "org.freedesktop.hostname1.") and allow anyone else to do it after authenticating as themselves, just create a file name /etc/polkit-1/rules.d/10-hostname-manage.rules (or similar) with following content ("org.freedesktop.hostname1.policy" file is provided by "systemd" package):

```
polkit.addRule(function(action, subject) {
   if (action.id.indexOf("org.freedesktop.hostname1.") == 0) {
      if (subject.isInGroup("children")) {
         return polkit.Result.NO;
      } else {
         return polkit.Result.AUTH_SELF_KEEP;
      }
   }
});
```

---

*The following example shows how the authorization decision can depend on variables passed by the *pkexec* mechanism ("org.freedesktop.policykit.policy" file is provided by "polkit" package)::

```
polkit.addRule(function(action, subject) {
   if (action.id == "org.freedesktop.policykit.exec" &&  action.lookup("program") == "/usr/bin/cat") {
      return polkit.Result.AUTH_ADMIN;
   }
});
```

The following example shows another use of variables. In this case, they are passed from UDisks (here is the complete list of its defined variables: http://storaged.org/doc/udisks2-api/2.7.2/udisks-polkit-actions.html) and they are used to match on ("org.freedesktop.UDisks2.policy" file is provided by "udisks2" package) :

```
// Allow users in group 'engineers' to perform any operation
//(note the use of Javascript' indexOf() method)
//on some drives without having to authenticate
polkit.addRule(function(action, subject) {
   if (action.id.indexOf("org.freedesktop.udisks2.") == 0 &&
      action.lookup("drive.vendor") == "SEAGATE" &&  action.lookup("drive.model") == "ST3300657SS" &&
      subject.isInGroup("engineers")) {
         return polkit.Result.YES;
      }
   }
});
```