polkit鉴权管理说明

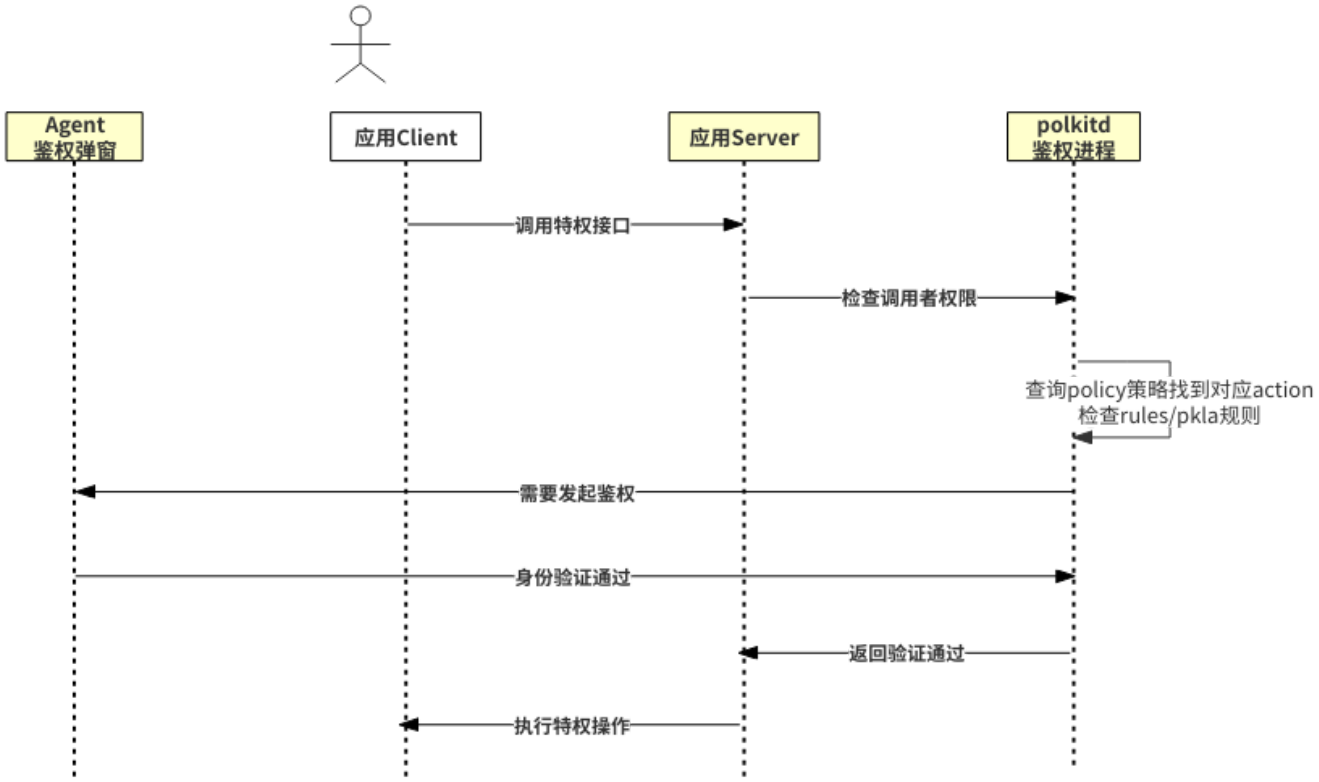
如果您只是想了解如何让自己的应用的某些接口被调用时，需要用户验证身份后才能进行，请移步《应用如何发起鉴权》。

如果您想了解操作系统中的身份验证弹窗程序是如何工作的，请移步《鉴权验证程序》。

名词介绍

`action`:动作，本文中主要指应用在`policy`指定的某一操作。

鉴权时序图



在实际使用中，我们一般会设计两个进程，分别是普通的客户端进程Client和具有root权限的服务端进程Server。

1、当Client需要执行某些操作时，可以调用Server进程的接口实现，从而完成普通进程实现root权限才有的操作。

2、Server进程在设计接口时，为了安全考虑，首先会向polkitd进程查询此当前操作action(action在Server程序安装时被提供给polkitd)是否允许，并根据结果来决定调用者是否能完成此项操作

3、polkit再从policy文件根据action id找到其授权策略，如果存在rules/pkla规则文件，将以规则文件中的授权策略为准，如果需要验证用户身份，agent进程此时会收到通知

4、agent进程在收到通知后，一般以弹窗的时候形式和用户产生交互，用户需要向agent证明自己的身份(polkit会通过PAM机制鉴别用户身份)

5、agent将验证用户身份的结果返回给polkitd，polkitd再返回给Server进程

6、Server根据结果决定此接口是否继续执行，从而完成一次完整的鉴权交互请求。

应用如何发起鉴权

场景

在uos操作系统中，用户打开gparted程序，打算对磁盘的分区情况进行修改，但先出现的是如下界面，只有在输出密码正确的情况下，才允许用户使用gparted进行操作。

图1 UOS的鉴权管理验证程序

目的

如果我们在设计gparted的时候，并不要求用户输入密码，打开就能使用，极有可能导致我们的电脑被别人借用时，误删了磁盘分区，导致系统无法启动。

鉴权机制就可以很好的避免的这一点，在运行前，gparted软件会通过polkit机制验证使用者的身份，确保使用者身份是"可信”的，才会对本次动作放行。

策略文件安装

`polkit` 是一个很简单的机制，但经过了很多年的发展，寥寥几句无法说得很清楚，这里只介绍如何利用这种机制，避免高权限操作被低权限进程随意调用导致的安全问题。

在polkit中，每一个操作称为一个 `action` ，可以使用 `pkaction` 查看系统中已经安装的policy策略，action被定义在policy策略文件中。这是一个xml格式的文件，以 `.policy` 结尾。下面的策略文件，我们在策略文件中定义了两个操作，分别是 `com.deepin.polkit.demo.fun1` 和 `com.deepin.polkit.demo.fun2` ，其中 `com.deepin.polkit.demo.fun1` 默认允许， `com.deepin.polkit.demo.fun2` 需要进行管理员身份验证。(policy文件的字段描述见polkit的 `DECLARING ACTIONS` 一章)

com.deepin.polkit.demo.policy XML 复制代码

```xml
<?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="com.deepin.polkit.demo.fun1">
    <description>echo hello</description>
    <message>Operation1</message>
    <message xml:lang="zh_CN">操作1</message>
    <defaults>
      <allow_inactive>no</allow_inactive>
      <allow_active>yes</allow_active>
    </defaults>
  </action>

  <action id="com.deepin.polkit.demo.fun2">
    <description>write file to /etc</description>
    <message>Operation2</message>
    <message xml:lang="zh_CN">操作2</message>
    <defaults>
      <allow_inactive>no</allow_inactive>
      <allow_active>auth_admin</allow_active>
    </defaults>
  </action>
</policyconfig>
```

也可以将 `allow_active` 设置为 `auth_self`，将只允许当前用户进行验证。`auth_admin` 意味这电脑上所有的管理员账户都可以验证。如果设置为 `yes` 则表示此动作默认允许。

定义好应用将来要提权的 `action` 后，需要将此 `xml` 文件放置到 `/usr/share/polkit-1/actions` 目录中，这也是polkit要求的固定路径。

应用鉴权

(以C++举例)

应用运行后，当执行到某个动作时，打算验证使用者的身份再继续，否则可能因身份问题拒绝此动作，我们的代码可以这样写：

main.cpp C++ 复制代码

```cpp
#include <QDebug>

#include <polkit-qt5-1/PolkitQt1/Authority>

using namespace PolkitQt1;
int main(int argc, char **argv)
{
    (void)(argc);
    (void)(argv);

    Authority::Result result;
    result = Authority::instance()->checkAuthorizationSync("com.deepin.polkit.demo.fun1"
                                                           , UnixProcessSubject(getpid())
                                                           , Authority::AllowUserInteraction);
    if (result == Authority::Yes) {
        qDebug() << "We are free, continue";
        // do something
        qDebug() << "Done, good job!";
        exit(0);
    }else{
        qWarning() << "So sad, permission denied!";
        exit(-1);
    }
}

```

`checkAuthorizationSync` 方法有三个参数：

第一个参数声明要鉴权的动作id，它在 `policy` 策略中被定义，是 `action` 的身份识别码

第二个参数一般传递需要授鉴权进程的id，用于 `polkit` 识别待鉴权进程

第三个参数默认指定 `Authority::AllowUserInteraction` 即可(或者为 `None` ，此时直接检查此动作是否被允许，如果 `policy` 或者 `rules` 文件中未表明此动作为允许，则直接返回失败，不再弹出鉴权管理的弹窗)。

checkAuthorizationSync C++ 复制代码

```cpp
/**
* Synchronous version of the checkAuthorization method.
*
* \param actionId the Id of the action in question
* \param subject subject that the action is authorized for (e.g. unix process)
* \param flags flags that influences the authorization checking
*
* \see checkAuthorization Asynchronous version of this method.
*/
Result checkAuthorizationSync(const QString &actionId, const Subject &subject,
                              AuthorizationFlags flags);
```

您也参照示例上的用法，在UOS操作系统中，大多数鉴权都是采用完整示例的demo2的方式，我们建议您也采用此种方案。

Q&A

1、如何通过polkit提权，让普通的用户进程执行一些root权限才能做的事情。

A: 设计两个进程，分别为A、B，其中A为root进程，B为普通进程。A通过进程间通信的方式(例如DBus)提供接口Method1供进程B调用。

2、设计的root进程提供了接口，但应用调用后所有的鉴权都是通过，拦截不住。

A：检查checkAuthorizationSync方法的第二个参数，传递的pid是否是需要鉴权的进程，而不是提供接口的进程。如果pid对应的进程本身是root权限，polkit会直接允许。

鉴权验证程序

在不同的发行版上，鉴权管理的验证程序并不总是一样，polkit允许不同的发行版开发符合其视觉特色的界面程序。在某些没有界面的Linux发行版上， `polkit` 还提供了 `PolkitAgentTextListener` 类型，用于开发基于文本交互的鉴权管理验证程序。

当普通进程通过 `polkit` 机制进行某些操作时，验证程序总算是能够'拦截',并根据策略文件选择是否和使用者交互的方式(例如弹窗)要求使用者鉴权。

概述

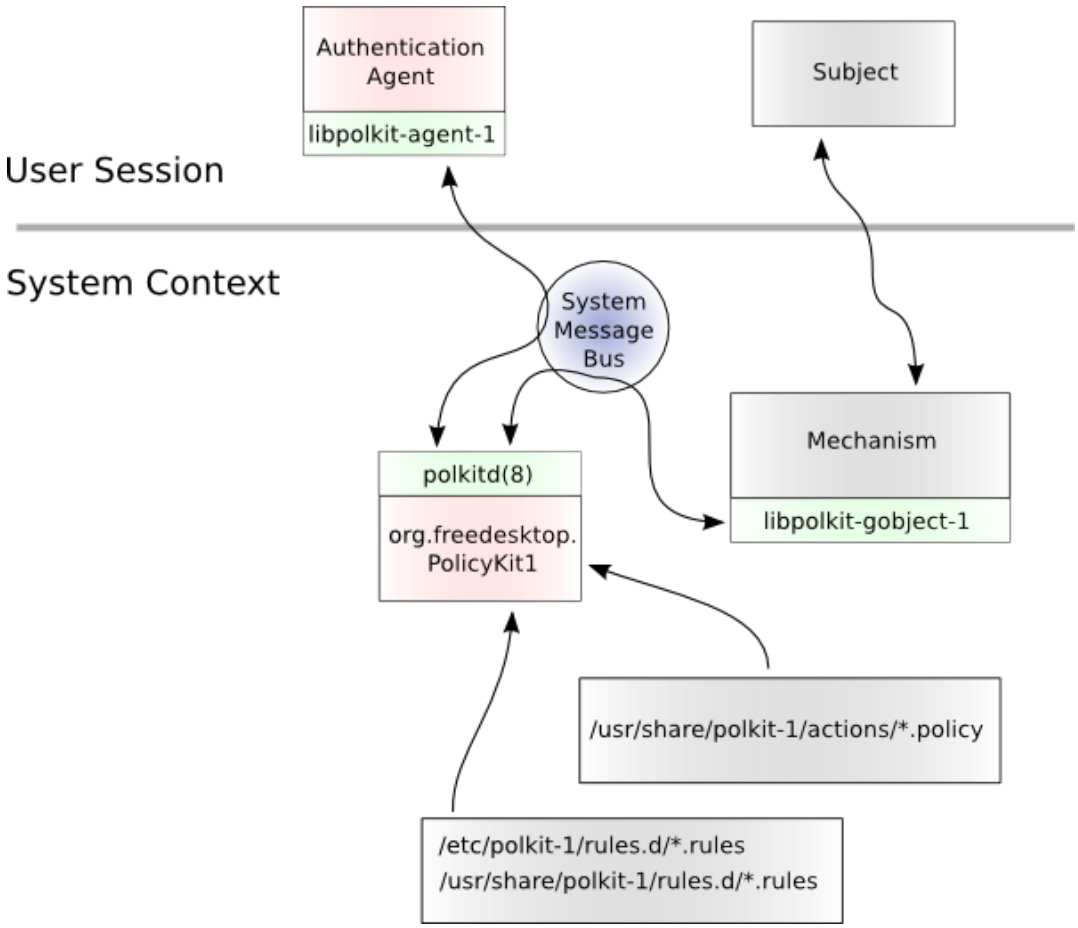
`polkit` 提供了一个鉴权 API，供特权程序（“ MECHANISMS ”）使用，通常通过某种形式的进程间通信机制为非特权程序（“ SUBJECTS ”）提供服务。在这种情况下，该机制通常将主体视为不受信任。对于来自主体的每个请求，该机制需要确定该请求是否被鉴权，或者它是否应该拒绝为主体提供服务。使用 `polkit` API，可以将此决定转交给受信任的一方： `polkit` 认证。

polkit 认证被作为系统守护进程来实现，这就是polkitd，它本身没有什么特权，因为它以 `polkitd` 系统用户身份运行。特权进程、待鉴权进程和认证验证程序使用系统消息总线与认证机制进行通信。

除了作为鉴权之外， `polkit` 还允许用户通过验证管理用户或客户端所属会话的所有者来获得临时鉴权。这对于需要验证当前系统的操作员是用户还是管理员用户的场景很有用。

系统架构

`polkit` 的系统架构由 `polkitd` （作为系统消息总线上的服务实现）和每个用户会话的身份验证代理（由用户的图形环境提供和启动）组成。动作由应用程序定义(以 `policy` 策略文件的形式)。供应商、站点和系统管理员可以通过鉴权规则(见附录)控制鉴权策略。

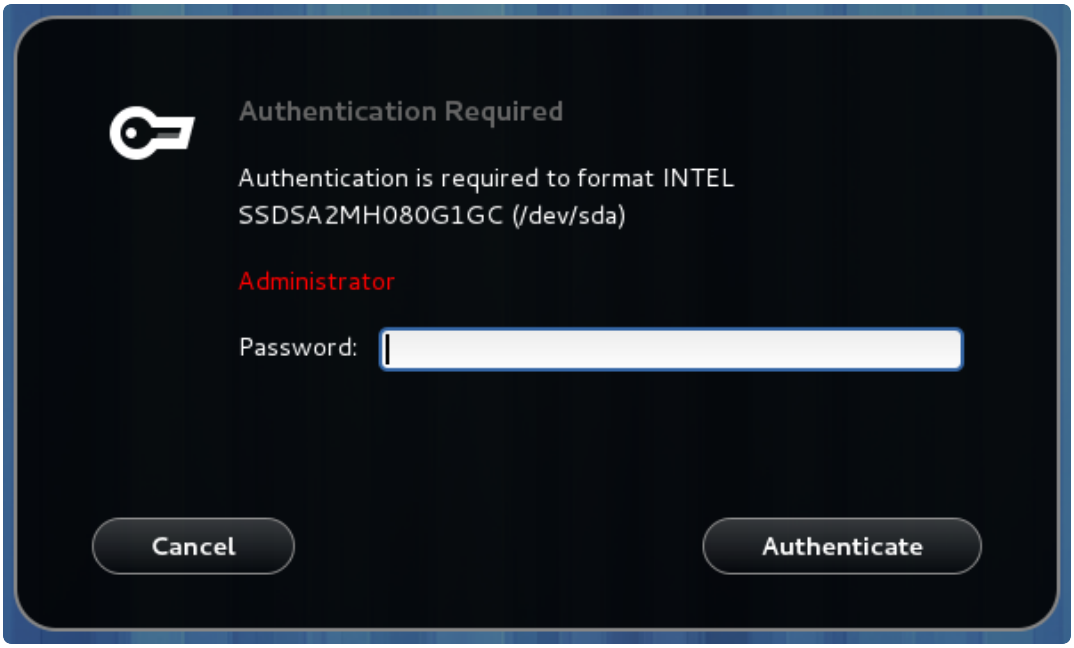


为方便起见，该 `libpolkit-gobject-1` 库封装了 `polkit` D-Bus API，可用于任何 C/C++ 程序以及支持 `GObjectIntrospection` 的高级语言，例如 `Javascript` 和 `Python`。也可以使用 D-Bus API 或 `pkcheck` 命令来检查鉴权。该 `libpolkit-agent-1` 库提供了本地身份验证系统的抽象，例如 `pam`，还提供了与 `polkit` D-Bus 服务的注册和通信。

有关编写 `polkit` 身份验证代理程序的更多信息，请参阅开发人员文档。

认证代理

身份验证代理用于使会话的使用者证明确实自己是当前用户（对使用者进行身份验证）或管理员用户（通过以管理员身份进行身份验证）。为了能更好和系统的其他组件一起集成发布（例如界面设计和体验与系统其他部分保持统一），身份验证代理一般由系统的发行商提供。例如，身份验证代理程序可能如下图所示：



不在桌面环境下运行的应用程序（例如，如果从 `ssh` 登录启动）可能没有与之关联的身份验证代理。此类应用程序可以使用 `PolkitAgentTextListener` 类型或 `pkttyagent` 帮助程序，以便用户可以使用文本界面进行身份验证。

声明操作

为了使用 `polkit`，您需要声明一系列 `action`。这些 `action` 对应不同的操作，客户端可以通过 `polkit` 机制去请求执行对应 `action` 的操作，这些 `action` 以 `XML` 文件的方式被安装到 `/usr/share/polkit-1/actions` 目录中。

关乎这些XML文件的书写规范详见附录

认证规则文件

在policy文件中，我们可以设置 `allow_active` 的值为 `yes` 或 `no` 来控制结果，但我们并不建议您这样做，而应该通过认证规则文件，规则文件可以在不修改策略文件的情况更改鉴权的默认行为。

(例如在policy文件中定义了某个action的allow_active为no，鉴权时将默认不通过，此时可以通过增加rules文件，修改此action的默认鉴权结果)

`polkit` 有版本之分,可以使用 `pkexec --version` 查询

pkexec — Bash | 复制代码

```
[uos@uos-PC 13:41:53 ~]$pkexec --version
pkexec version 0.105
```

106之前版本的规则文件是 `pkla` 格式，之后的版本是 `rules` .

pkla

105版本的规则文件以pkla结尾，polkitd 以字典顺序读取 /etc/polkit-1/rules.d 和 /usr/share/polkit-1/rules.d 目录中的 .rules 文件。如果两个文件采用同样的名称，/etc 中的文件在 /usr 中的文件前面。处理旧的 .pkla 文件时，最先采用的是最后的规则。使用新的 .rules 文件时，首先采用的是第一个匹配的规则。

在pkla规则文件中，我们只需要在Action字段中指明规则对应的action id,所有的配置均以Key=value的形式，其中Key为固定字段，value视情况而定，附上一个示例的pkla规则文件，表示action id = org.kubuntu.qaptworker3.commitchanges的操作默认允许，[]内您可以自定义内容用于提示此规则文件的作用。

您可以按照这种格式修改成自己想要的规则文件，并重命名后放于对应的目录中，从而让其生效。

org.kubuntu.qaptworker3.commitchanges.pkla — Bash | 复制代码

```
[commitchanges without auth]
Identity=unix-group:sudo
Action=org.kubuntu.qaptworker3.commitchanges
ResultAny=no
ResultInactive=no
ResultActive=yes
```

另外要注意的一点是，pkla的规则要放置的路径为/etc/polkit-1/localauthority/10-vendor.d/或同级别其他目录下，这里和rules文件是不一样的。

rules

`rules` 规则文件是用 `JavaScript` 编程语言编写的，并通过全局对象（类型为 `Polkit`）与 `polkitd` 交互。需要放置到系统的 `/etc/polkit-1/rules.d` 和 `/usr/share/polkit-1/rules.d` 目录中才会生效。

列举一些常用的规则文件：

允许admin组中的所有用户在不更改其他用户策略的情况下执行用户管理

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

将管理用户定义为wheel组中的用户

```Bash
polkit.addAdminRule(function(action, subject) {
    return ["unix-group:wheel"];
});
```

禁止组 `children` 中的用户更改主机名配置（即，任何以 `org.freedesktop.hostname1.` 标识符开头的操作）并允许其他任何人在验证身份后执行此操作：

```Bash
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;
        }
    }
});
```

附录

polkit介绍

polkit身份验证代理程序开发

Polkit_(简体中文)

链接