会员力量，点亮园子希望 加入助力
博客园 cnblogs.com 会员 新闻 博问 闪存 班级 AI培训 代码改变世界 注册 登录

随笔 - 1181 文章 - 6 评论 - 29 阅读 - 177万

PAM详解（二）PAM开发

摘自：http://blog.chinaunix.net/uid-29479952-id-5761564.html

目录

==========

第二部分 PAM开发

一、Pam应用程序开发

任何一个支持PAM的应用程序在进行认证时必须以pam_start( )开始进行初始化，最后以pam_end( )结束。

1.1 pam_start()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. Int
3. pam_start(
4. const char *service_name,
5. const char *user,
6. const struct pam_conv *pam_conversation,
7. pam_handle_t **pamh
8. );
```

1.1.1参数讲解

service_name
应用的名字，这个名字参数非常重要。
当应用程序执行验证操作时，libpam库会在/etc/pam.d/下寻找以service_name命名的配置文件，该配置文件中指定了相关参数，其中包括将要调用哪个动态链接库进行验证。
配置文件的格式与前面讲的/etc/pam.conf的格式基本一致，只是去掉了service-name这一项，因为该配置文件的名字就是service-name。
其实linux系统中/etc/pam.conf这个配置文件在整个pam框架中已经起不到什么作用，至少在ubuntu10.04系统中是否对这个配置文件进行配置，根本什么也不影响。
前面的对/etc/pam.conf的所有介绍均适用于/etc/pam.d/目录下的配置文件。
user
用户名，即PAM框架所作用的用户的名称。说白了就是你要对哪个用户进行pam验证，该用户就是此轮验证中（pam_start——pam_end）被操作的对象。
&conv
对话函数conv，用于提供PAM与用户或应用程序进程通信的通用方法。对话函数是必需的，因为PAM模块无法了解如何进行通信。通信可以采用 GUI、命令行、智能读卡器或其他设备等方式进行。这个对话函数是一个回调函数，这里只是对它的一个注册，接下来的验证过程中，应用程序和服务模块之间的所有信息交互都要通过它。对话函数需要应用程序的作者自己编写。
&pamh
PAM句柄 pamh，即 PAM 框架用于存储有关当前操作信息的不透明句柄。成功调用 pam_start() 后将返回此句柄。要研究这个pamh，需要对pam的内部实现机制进行了解，有点复杂，暂时先不说这个。大家把它想成类似于一个socket套接字的东西就行，反正就是一个句柄而已。

1.1.2返回值讲解

PAM_ABORT
一般性的错误，我也不知道什么叫一般性的错误。
PAM_BUF_ERR
内存缓冲区错误，具体的我也不知道。
PAM_SUCCESS
成功建立验证xx，反正就是成功了。
PAM_SYSTEM_ERR
系统错误，参数中有无效的指针。有些参数必须提供指向了真实数据的指针变量。

1.2 pam_end()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. int
3. pam_end(
4. pam_handle_t *pamh,
5. int pam_status);
```

1.2.1参数讲解

pam_end()函数是整个pam验证过程中，应用程序最后调用的函数。从此以后句柄pamh不再有效，而且所有的占用的内存将被释放。
参数pamh就是pam_start()函数中创建的pamh；
参数pam_status，是应用程序在执行pam_end()函数前所执行的最后一个pam API函数的返回值。
Pam_status通常被传递给服务模块的特有的一个回调函数cleanup()，这样服务模块就知道关闭应用程序是否成功，然后继续执行和之前一样的工作，即给其他应用程序提供验证服务。
Pam_end()释放了pam_set_item()和pam_get_item()申请的所有内存，所有对象的指针在pam_end()执行后也都不在有效。

1.2.2返回值讲解

PAM_SUCCESS
应用程序的验证过程被成功终止。
PAM_SYSTEM_ERR
系统错误，例如pamt是个无效指针NULL，或者函数还在被服务模块调用。
在pam_start()和pam_end()之间就是应用程序要执行的验证操作。就是所谓的四个服务模块。当然，你可以只进行其中的一项认证或者几个认证，哪怕没有任何验证操作，只要您认为值得就可以。

公告
昵称：LiuYanYGZ
园龄：8年8个月
粉丝：60
关注：1
+加关注

< 2023年12月 >

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索
找找看
常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签
我的标签
linux(252)
java(69)
5G(53)
前端(41)
gcc(33)
数据库(32)
Python(26)
html(26)
内核(24)
网络编程(20)
更多
随笔档案
2023年12月(11)
2023年11月(17)
2023年10月(5)
2023年9月(12)
2023年7月(17)
2023年6月(11)
2023年5月(15)
2023年4月(4)
2023年3月(4)
2023年2月(2)
2023年1月(2)
2022年12月(5)
2022年11月(18)
2022年10月(5)
2022年9月(17)
更多
阅读排行榜
1. 可执行文件(ELF)格式的理解(52643)
2. 一张图让你看懂HDMI针脚定义(42860)
3. gcc编译选项(42267)
4. PCRE的安装及使用(36142)
5. 介绍几个在线画流程图的工具(35107)
评论排行榜
1. Ubus简单理解(2)
2. Win10新建文件不自动刷新(2)
3. windows10下安装source insight 4...
4. rpmbuild SPEC语法(2)
5. FastStone Capture注册破解码(1)
推荐排行榜
1. BorderLayout布局，修改各个区域大...
2. FastStone Capture注册破解码(5)
3. C/C++代码静态分析工具调研(4)
4. gcc编译选项(4)
5. Win10新建文件不自动刷新(2)
最新评论
1. Re:FastStone Capture注册破解码
棒!!!
--梦吴越
2. Re:Java Swing 创建转圈的进度提示框
这两个毫无关系啊，还有什么没有写嘛？
--大海111
3. Re:rpmbuild SPEC语法
很不错，赞
--mvpbang
4. Re:OpenSSL命令---ciphers
nice
--香蕉有毒
5. Re:malloc和calloc的差别
看源代码calloc是用 malloc申请了一段内存，然后置0了的意思吗？
--Nichoooolas

1.3认证管理pam_authenticate()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. int
3. pam_authenticate(
4. pam_handle_t *pamh,
5. int flags);
```

该函数被用来验证用户的合法性(即令牌认证)，这个用户就是在pam_start()参数传递的user。User被要求提供一个密码或者一个简单的数字输入。然后服务模块中对应的pam_sm_authenticate()函数会检测user的输入并验证是否合法。

1.3.1参数讲解

> **pamh** 参数pamh就是pam_start()函数中创建的pamh了(所有函数的pamh参数都相同，下面的函数将不再介绍pamh参数)；
> **flags** 参数flags可以被设置为0，或者设置为如下值：
> PAM_SILENT
> 不输出任何信息
> PAM_DISALLOW_NULL_AUTHTOK
> 如果用户没有注册，那么服务模块应该返回PAM_DISALLOW_NULL_AUTHTOK；

1.3.2返回值讲解

> **PAM_ABORT**
> 如果收到这个，应用程序应该立即调用pam_end()退出；
> **PAM_AUTH_ERR**
> user没有被验证；
> **PAM_CRED_INSUFFICIENT**
> 由于一些原因应用程序没有足够的凭证来验证用户；
> **PAM_AUTHINFO_UNVAIL**
> 服务模块不能获取user的验证信息，原因可能是网络或硬件配置错误；
> **PAM_MAXTRIES**
> 服务模块验证用户的次数达到上限，应用程序这边不要再提交验证请求了，也就是说不要再运行pam_ authenticate()了；
> **PAM_SUCCESS**
> 用户成功通过验证
> **PAM_USER_UNKNOWN**
> 该用户不能够被服务模块识别，我估计要么是用户名格式非法，要么是用户没有注册等其他原因。

1.4账户管理pam_acct_mgmt()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. int
3. pam_acct_mgmt(
4. pam_handle_t *pamh,
5. int flags);
```

pam_acct_mgmt()通常被用来确认用户账户是否有效。确认的内容可以包括下面的内容：
令牌认证(就是pam_authenticate()实现的功能)
账户是否过期；
访问权限
该函数一般在pam_authenticate()成功执行(即用户通过验证)之后被调用执行，所以上面内容中的第一项常可以被省略。

1.4.1参数讲解

> 与pam_authenticate完全相同。

1.4.2返回值讲解

> **PAM_ACCT_EXPIRED**
> 用户已经过期失效；
> **PAM_AUTH_ERR**
> 用户令牌认证失败；
> **PAM_NEW_AUTHTOK_REQD**
> 该用户账户是有效的但是认证令牌是过期的，正确的做法是回复该值要求用户执行pam_chauthtok()来更新令牌(密码)在用户获得其他服务之前。
> **PAM_PERM_DENIED**
> 不允许访问，应该就是所谓的权限控制；
> **PAM_SUCCESS**
> 认证令牌被成功更新，或者是用过通过认证；
> **PAM_USER_UNKNOWN**
> 该用户不能够被服务模块识别；

1.5会话管理pam_open_session()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. int
3. pam_open_session(
4. pam_handle_t *pamh,
5. int flags);
```

该函数为已经成功通过验证的用户建立一个用户会话，该会话应该在后面被函数pam_close_session()终止。

1.5.1参数讲解

> 参数**flags**可以被设置为0或者下面的值：
> PAM_SILENT
> 不输任何信息；

1.5.2返回值讲解

> **PAM_ABORT**
> 一般性的失败；
> **PAM_BUF_ERR**
> 内存缓冲区错误；
> **PAM_SESSION_ERR**
> 建立会话失败；
> **PAM_SUCCESS**
> 成功建立会话；

1.6会话管理pam_close_session()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
```

```
2. int
3. pam_close_session(
4. pam_handle_t *pamh,
5. int flags);
```
该函数被用来关闭pam_open_session()创建的会话。参数和返回值与pam_open_session()的参数和返回值完全相同。

1.7密码管理pam_chauthtok()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. int
3. pam_chauthtok(
4. pam_handle_t *pamh,
5. int flags);
```

1.7.1参数讲解

> **PAM_SILENT**
> 不输任何信息;
> **PAM_CHANGE_EXPIRED_AUTHTOK**
> 告诉服务模块只更新过期令牌，如果不设置这个参数，应用程序要求更改所有用户的令牌;

1.7.2返回值讲解

> **PAM_AUTHTOK_ERR**
> 服务模块未能获得新的用户令牌;
> **PAM_AUTHTOK_RECOVERY_ERR**
> 服务模块未能获得旧的用户令牌;
> **PAM_AUTHTOK_LOCK_BUSY**
> 又有用户令牌被锁定，服务模块不能对其进行更改;.
> **PAM_AUTHTOK_DISABLE_AGING**
> 用户令牌被至少一个服务模块禁用了;
> **PAM_PERM_DENIED**
> 没有权限;
> **PAM_SUCCESS**
> 用户令牌被成功更新;
> **PAM_TRY_AGAIN**
> 并不是所有的服务模块都能够更新用户令牌，遇到这种情况，用户令牌都不能得到更新;
> **PAM_USER_UNKNOWN**
> 该用户不能够被服务模块识别;

1.8认证管理pam_setcred()

点击(此处)折叠或打开

```
1. #include <security/pam_appl.h>
2. int
3. pam_setcred(
4. pam_handle_t *pamh,
5. int flags);
```

pam_setcred()函数被用来创建、维持、或删除一个用户的证书。Pam_setcred()应该在user已经通过验证(after pam_authenticate)并且在会话建立之前(before pam_open_ session)被调用。删除user证书的操作必须在会话被关闭之后执行(after pam_close_ session).user证书应该被应用程序创建，而不是被pam库或者服务模块创建。

1.8.1参数讲解

> 参数flags可以被设置为0或者下面的值:
> **PAM_ESTABLISH_CRED**
> 初始化用户证书;
> **PAM_DELETE_CRED**
> 删除用户证书;
> **PAM_REINITIALIZE_CRED**
> 完全重置用户证书;
> **PAM_REFRESH_CRED**
> 延长用户证书的生命周期;

1.8.2返回值讲解

> **PAM_BUF_ERR**
> 内存缓冲区错误;
> **PAM_CRED_ERR**
> 设置(创建、维持、重置、回复、删除等)用户证书失败
> **PAM_CRED_EXPIRED**
> 用户证书过期;
> **PAM_CRED_UNAVAIL**
> 回复用户证书失败;
> **PAM_SUCCESS**
> 数据被成功存储;
> **PAM_SYSTEM_ERR**
> 系统错误，例如无效的指针被传入，函数正在被其他模块调用，或者系统错误等等;
> **PAM_USER_UNKNOWN**
> 该用户不能被够服务模块识别;

二、Pam服务模块开发

2.1认证管理pam_sm_authenticate()

点击(此处)折叠或打开

```
1. #define PAM_SM_AUTH
2. #include <security/pam_modules.h>
3. PAM_EXTERN int
4. pam_sm_authenticate(
5. pam_handle_t *pamh,
6. int flags,
7. int argc,
8. const char **argv);
```

pam_sm_authenticate()函数是pam_authenticate()函数在服务模块中的接口，用于执行验证用户令牌的任务。下面将对所有pam_sm_xxx()函数统称为接口函数。

2.1.1参数讲解

> 服务模块中6个接口函数的参数完全相同，仅在这里做详细说明:
> 我们永远无法在我们设计的应用程序中直接调用这6个接口函数，也无法在我们自己设计的服务模块中让这6个接口函数相互调用，这6接口函数只能作为回调函数以动态链接库的形式存在，并且只能被PAM库调用。认证请求和认证服务是分离的，应用程序只负责提出认证请求，服务模块负责认证，两者靠PAM库连接起来。
> PAM从认证请求函数中获得pamh参数和flags参数，并准备作为参数传递给对应接口函数(分别对应接口函数中pamh和flags)。然后PAM库从配置文件(配置文件的名字在pam_start()函数中指定)中的arguments项中，获取将要传递给接口函数的第四个参数argv，并计算出参数的个数作为接口函数的第三个参数。
> 整个参数传递的过程大致描述如上。接下来将不对接口函数的参数进行讲解，因为接口函数只能被PAM库调用，传递进来的参数的含义在其他部分已做过说明。

2.1.2返回值讲解

> 接口函数pam_sm_xxx()的返回值与应用程序中对应函数pam_xxx()的返回值基本一致，不做讲解，若有疑问，请参考Man手册。

2.2账户管理pam_sm_acct_mgmt()

```
#define PAM_SM_ACCOUNT
#include
PAM_EXTERN int
pam_sm_acct_mgmt(pam_handle_t *pamh, int flags, int argc, const char **argv);
```

2.3会话管理pam_sm_open_session()

```
#define PAM_SM_SESSION
#include
PAM_EXTERN int
pam_sm_open_session(pam_handle_t *pamh, int flags, int argc, const char **argv);
```

2.4会话管理pam_sm_close_session()

```
#define PAM_SM_SESSION
#include
PAM_EXTERN int
pam_sm_close_session(pam_handle_t *pamh, int flags, int argc, const char **argv);
```

2.5密码管理pam_sm_chauthtok()

```
#define PAM_SM_PASSWORD
#include
PAM_EXTERN int
pam_sm_chauthtok(pam_handle_t *pamh, int flags, int argc, const char **argv);
```

2.6认证管理pam_sm_setcred()

```
#define PAM_SM_AUTH
#include
PAM_EXTERN int
pam_sm_setcred(pam_handle_t *pamh, int flags, int argc, const char **argv);
```

三、Pam配置文件编辑

```
使用pam_start()函数指定配置文件的文件名之后，就应该在/etc/pam.d/目录下创建对应的配置文件，并按照第一部分“Pam的配置文件”中的讲解编辑配置文件并保存，即可，不需要重启什么服务。
```


标签：linux

好文要顶 关注我 收藏该文

LiuYanYGZ
粉丝 - 60 关注 - 1
+加关注

0 推荐 0 反对

升级成为会员

« 上一篇：linux密码登陆时加入自己登陆验证模块(pam)，xshell工具可用，xftp工具使用无响应
» 下一篇：PAM详解(一)PAM介绍

posted @ 2020-03-11 10:10 LiuYanYGZ 阅读(2774) 评论(0) 编辑 收藏 举报

会员救园 刷新页面 返回顶部

登录后才能查看或发表评论，立即 登录 或者 逛逛 博客园首页

编辑推荐：
· 带团队后的日常思考(十三)
· 一次 elasticsearch 查询瞬间超时案例分析
· 聊一聊 .NET高级调试 中的一些内存术语
· 安卓端出现 https 请求失败的一次问题排查
· 初探 webpack 之单应用多端构建

阅读排行：
· Vue3+Vite+ElementPlus管理系统常见问题
· Windows风格的个人网盘，支持文档在线编辑
· [解决方案]MySQL5.7 百万数据迁移到 ElasticSearch7.x 的思考
· three.js 汽车行驶动画效果
· ML.NET 3.0 增强了深度学习和数据处理能力

Copyright © 2023 LiuYanYGZ
Powered by .NET 8.0 on Kubernetes