在Linux中执行有些程序时，这些程序在执行前首先要对启动它的用户进行认证，符合一定的要求之后才允许执行，例如login, su等。在Linux中进行身份或是状态的验证程序是由PAM来进行的，PAM（Pluggable Authentication Modules）可动态加载验证模块，因为可以按需要动态的对验证的内容进行变更，所以可以大大提高验证的灵活性。

一、PAM模块介绍

Linux-PAM（即linux可插入认证模块）是一套共享库,使本地系统管理员可以随意选择程序的认证方式。换句话说，不用(重新编写)重新编译一个包含PAM功能的应用程序，就可以改变它使用的认证机制，这种方式下，就算升级本地认证机制，也不用修改程序。
PAM使用配置/etc/pam.d/下的文件，来管理对程序的认证方式.应用程序 调用相应的配置文件，从而调用本地的认证模块.模块放置在/lib/security下，以加载动态库的形式进，像我们使用su命令时，系统会提示你输入root用户的密码.这就是su命令通过调用PAM模块实现的。

二、PAM的配置文件说明

PAM配置文件有下面两种写法:
1）写在/etc/pam.conf文件中，但centos6之后的系统中，这个文件就没有了。
2）将PAM配置文件放到**/etc/pam.d/**目录下,其规则内容都是不包含 service 部分的，即不包含服务名称，而/etc/pam.d 目录下文件的名字就是服务名称。如： vsftpd,login等，只是少了最左边的服务名列。
如：/etc/pam.d/sshd

```
[root@centos6-test06 ~]# cat /etc/pam.d/sshd
#%PAM-1.0
auth       required pam_sepermit.so
auth       include      password-auth
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so      close
session    required     pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed i
session    required     pam_selinux.so      open env_params
session    required     pam_namespace.so
session    optional     pam_keyinit.so      force revoke
session    include      password-auth
```

由上面的pam模块文件内容看，可以**将pam配置文件分为四列**，

- 第一列代表模块类型
- 第二列代表控制标记
- 第三列代表模块路径
- 第四列代表模块参数

1）第一列：PAM的模块类型

Linux-PAM有四种模块类型，分别代表四种不同的任务，它们是:
认证管理（auth），账号管理（account），会话管理（session）和密码（password）管理，一个类型可能有多行，它们按顺序依次由PAM模块调用。

管理方式	说明
auth	用来对用户的身份进行识别.如:提示用户输入密码,或判断用户是否为root等.
account	对帐号的各项属性进行检查.如:是否允许登录,是否达到最大用户数,或是root用户是否允许在这个终端登录等.
session	这个模块用来定义用户登录前的,及用户退出后所要进行的操作.如:登录连接信息,用户数据的打开与关闭,挂载文件系统等.
password	使用用户信息来更新.如:修改用户密码.

对以上四种模块类型进一步补充说明:

auth：表示鉴别类接口模块类型用于检查用户和密码，并分配权限；

这种类型的模块为用户验证提供两方面服务。让应用程序提示用户输入密码或者其他标记，确认用户合法性；通过他的凭证许可权限，设定组成员关系或者其他优先权。

account：表示账户类接口，主要负责账户合法性检查，确认帐号是否过期，是否有权限登录系统等；

这种模块执行的是基于非验证的帐号管理。他主要用于限制/允许用户对某个服务的访问时间，当前有效的系统资源（最多可以多少用户），限制用户位置（例如：root只能通过控制台登录）。

多数情况下auth和account会一起用来对用户登录和使用服务的情况进行限制。这样的限制会更加完整。比如下面是一个具体的例子：login是一个应用程序。Login要完成两件工作——首先查询用户，然后为用户提供所需的服务，例如提供一个shell程序。通常Login要求用户输入名称和密码进行验证。当用户名输入的时候，系统自然会去比对该用户是否是一个合法用户，是否在存在于本地或者远程的用户数据库中。如果该账号确实存在，那么是否过期。这些个工作是由account接口来负责。

如果用户满足上述登录的前提条件，那么它是否具有可登录系统的口令，口令是否过期等。这个工作就要由auth接口来负责了，他通常会将用户口令信息加密并提供给本地（/etc/shadow）或者远程的(ldap，kerberos等)口令验证方式进行验证。

如果用户能够登录成功，证明auth和account的工作已经完成。但整个验证过程并没有完全结束。因为还有一些其他的问题没有得到确认。例如，用户能够在服务器上同时开启多少个窗口登录，用户可以在登录之后使用多少终端多长时间，用户能够访问哪些资源和不能访问哪些资源等等。也就是说登录之后的后续验证和环境定义等还需要其他的接口。这就是我们下面要提到的两组接口：

session：会话类接口。实现从用户登录成功到退出的会话控制；

处理为用户提供服务之前/后需要做的些事情。包括：开启/关闭交换数据的信息，监视目录等，设置用户会话环境等。也就是说这是在系统正式进行服务提供之前的最后一道关口。

password：口令类接口。控制用户更改密码的全过程。也就是有些资料所说的升级用户验证标记。

注意：上述接口在使用的时候，每行只能指定一种接口类型，如果程序需要多种接口的话，可在多行中分别予以规定。

2）第二列：PAM的控制标记

PAM使用控制标记来处理和判断各个模块的返回值.（在此只说明简单的认证标记）。

控制标记	说明
required	表示即使某个模块对用户的验证失败，也要等所有的模块都执行完毕后,PAM 才返回错误信息。这样做是为了不让用户知道被哪个模块拒绝。如果对用户验证成功，所有的模块都会返回成功信息。
requisite	与required相似,但是如果这个模块返回失败,则立刻向应用程序返回失败,表示此类型失败.不再进行同类型后面的操作.
sufficient	表示如果一个用户通过这个模块的验证，PAM结构就立刻返回验证成功信息（即使前面有模块fail了，也会把 fail结果忽略掉），把控制权交回应用程序。后面的层叠模块即使使用requisite或者required 控制标志，也不再执行。如果验证失败，sufficient 的作用和 optional 相同
optional	表示即使本行指定的模块验证失败，也允许用户接受应用程序提供的服务，一般返回PAM_IGNORE(忽略).

规定如何处理PAM模块鉴别认证的结果，简而言之就是鉴别认证成功或者失败之后会发生什么事，如何进行控制。单个应用程序可以调用多种底层模块，通常称为"堆叠"。对应于某程序按照配置文件中出现顺序执行的所有模块成为"堆"，堆中的各模块的地位与出错时的处理方式由control_flag栏的取值决定，他的四种可能的取值分别为required、Requisite、sufficient或_optional：

required：表示该行以及所涉及模块的成功是用户通过鉴别的必要条件。换句话说，只有当对应于应用程序的所有带required标记的模块全部成功后，该程序才能通过鉴别。同时，如果任何带required标记的模块出现了错误，PAM并不立刻将错误消息返回给应用程序，而是在所有模块都调用完毕后才将错误消息返回调用他的程序。 反正说白了，就是必须将所有的模块都执行一次，其中任何一个模块验证出错，验证都会继续进行，并在执行完成之后才返回错误信息。这样做的目的就是不让用户知道自己被哪个模块拒绝，通过一种隐蔽的方式来保护系统服务。就像设置防火墙规则的时候将拒绝类的规则都设置为drop一样，以致于用户在访问网络不成功的时候无法准确判断到底是被拒绝还是目标网络不可达。

requisite：与required相仿，只有带此标记的模块返回成功后，用户才能通过鉴别。不同之处在于其一旦失败就不再执行堆中后面的其他模块，并且鉴别过程到此结束，同时也会立即返回错误信息。与上面的required相比，似乎要显得更光明正大一些。

sufficient：表示该行以及所涉及模块验证成功是用户通过鉴别的充分条件。也就是说只要标记为sufficient的模块一旦验证成功，那么PAM便立即向应用程序返回成功结果而不必尝试任何其他模块。即便后面的层叠模块使用了requisite或者required控制标志也是一样。当标记为sufficient的模块失败时，sufficient模块会当做 optional对待。因此拥有sufficient 标志位的配置项在执行验证出错的时候并不会导致整个验证失败，但执行验证成功之时则大门敞开。所以该控制位的使用务必慎重。

optional：他表示即便该行所涉及的模块验证失败用户仍能通过认证。在PAM体系中，带有该标记的模块失败后将继续处理下一模块。也就是说即使本行指定的模块验证失败，也允许用户享受应用程序提供的服务。使用该标志，PAM框架会忽略这个模块产生的验证错误，继续顺序执行下一个层叠模块。

include：表示在验证过程中调用其他的PAM配置文件。在RHEL系统中有相当多的应用通过完整调用/etc/pam.d/system-auth来实现认证而不需要重新逐一去写配置项。这也就意味着在很多时候只要用户能够登录系统，针对绝大多数的应用程序也能同时通过认证。

另外还有一种比较复杂的格式为value = action的语法来设置控制标志，标志之间会以空格分开。格式如下：

```
value1 = action1 value2 = action2 ……
```

其中value可以是下列Linux PAM库的返回值：
success、open_err、symbol_err、service_err、 system_err、buf_err、perm_denied、auth_err、cred_insufficient、authinfo_unavail、user_unknown、maxtries、new_authtok_reqd、acct_expired、session_err、cred_unavail、cred_expired、cred_err、no_module_data、conv_err、 authtok_err、

authtok_recover_err、authtok_lock_busy、authtok_disable_aging、 try_again、ignore、abort、authtok_expired、module_unknown、bad_item和default。

最后一个(default)能够用来设置上面的返回值无法表达的行为。

actionN可以是一个非负整数或者是下面的记号之一：ignore、ok、done、bad、die和reset。如果是非负整数J，就表示需要忽略后面J个同样类型的模块。通过这种方式，系统管理者可以更加灵活地设置层叠模块，模块的层叠路径由单个模块的反应决定。

关于这几个记号的详细解释：
ignore：如果使用层叠模块，那么这个模块的返回值将被忽略，不会被应用程序知道。
bad：他表示这个返回码应该被看作是模块验证失败的标志。如果这个模块是层叠模块的第一个验证失败的模块，那么他的状态值就是整个层叠模块验证的状态值和结果。
die：终止层叠模块验证过程，立刻返回到应用程序。
ok：告诉PAM这个模块的返回值将直接作为所有层叠模块的返回值。也就是说，如果这个模块前面的模块返回状态是PAM_SUCCESS，那这个返回值就会覆盖前面的返回状态。注意：如果前面的模块的返回状态表示模块验证失败，那么不能使用这个返回值再加以覆盖。
done：终止后续层叠模块的验证，把控制权立刻交回应用程序。
reset：清除所有层叠模块的返回状态，从下一个层叠模块重新开始验证。

3）模块路径
模块路径.即要调用模块的位置. 如果是64位系统，一般保存在/lib64/security,如：pam_unix.so，同一个模块,可以出现在不同的类型中.它在不同的类型中所执行的操作都不相同.这是由于每个模块，针对不同的模块类型,编制了不同的执行函数。

4）模块参数
模块参数,即传递给模块的参数.参数可以有多个,之间用空格分隔开,如:password required pam_unix.so nullok obscure min=4 max=8 md5。

三、PAM模块的工作原理和流程

以RHEL系统为例，当pam安装之后有两大部分：在/lib/security目录下的各种pam模块以及/etc/pam.d和/etc/pam.d目录下的针对各种服务和应用已经定义好的pam配置文件。当某一个有认证需求的应用程序需要验证的时候，一般在应用程序中就会定义负责对其认证的PAM配置文件。以vsftpd为例，在它的配置文件/etc/vsftpd/vsftpd.conf中就有这样一行定义：

```
pam_service_name=vsftpd
```

表示登录FTP服务器的时候进行认证是根据/etc/pam.d/vsftpd文件定义的内容进行。

那么，当程序需要认证的时候已经找到相关的pam配置文件，认证过程是如何进行的？下面我们将通过解读/etc/pam.d/system-auth文件予以说明。

首先要声明一点的是：system-auth是一个非常重要的pam配置文件，主要负责用户登录系统的认证工作。而且该文件不仅仅只是负责用户登录系统认证，其它的程序和服务通过include接口也可以调用到它，从而节省了很多重新自定义配置的工作。所以应该说该文件是系统安全的总开关和核心的pam配置文件。

下面是/etc/pam.d/system-auth文件的全部内容：

```
[root@centos6-test06 ~]# grep -v ^# /etc/pam.d/system-auth
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so

account     required      pam_unix.so
account     sufficient    pam_localuser.so
```

```
account     sufficient    pam_succeed_if.so uid < 500 quiet
account     required      pam_permit.so

password    requisite     pam_cracklib.so try_first_pass retry=3 type=
password    sufficient    pam_unix.so sha512 shadow nullok try_first_pass
password    required      pam_deny.so

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond
session     required      pam_unix.so
```

第一部分表示，当用户登录的时候，首先会通过auth类接口对用户身份进行识别和密码认证。所以在该过程中验证会经过几个带auth的配置项。

其中的第一步是通过pam_env.so模块来定义用户登录之后的环境变量， pam_env.so允许设置和更改用户登录时候的环境变量，默认情况下，若没有特别指定配置文件，将依据/etc/security/pam_env.conf进行用户登录之后环境变量的设置。

然后通过pam_unix.so模块来提示用户输入密码，并将用户密码与/etc/shadow中记录的密码信息进行对比，如果密码比对结果正确则允许用户登录，而且该配置项的使用的是“sufficient”控制位，即表示只要该配置项的验证通过，用户即可完全通过认证而不用再去走下面的认证项。不过在特殊情况下，用户允许使用空密码登录系统，例如当将某个用户在/etc/shadow中的密码字段删除之后，该用户可以只输入用户名直接登录系统。

下面的配置项中，通过pam_succeed_if.so对用户的登录条件做一些限制，表示允许uid大于500的用户在通过密码验证的情况下登录，在Linux系统中，一般系统用户的uid都在500之内，所以该项即表示允许使用useradd命令以及默认选项建立的普通用户直接由本地控制台登录系统。

最后通过pam_deny.so模块对所有不满足上述任意条件的登录请求直接拒绝，pam_deny.so是一个特殊的模块，该模块返回值永远为否，类似于大多数安全机制的配置准则，在所有认证规则走完之后，对不匹配任何规则的请求直接拒绝。

第二部分的三个配置项主要表示通过account账户类接口来识别账户的合法性以及登录权限。

第一行仍然使用pam_unix.so模块来声明用户需要通过密码认证。第二行承认了系统中uid小于500的系统用户的合法性。之后对所有类型的用户登录请求都开放控制台。

第三部分会通过password口另类接口来确认用户使用的密码或者口令的合法性。第一行配置项表示需要的情况下将调用pam_cracklib来验证用户密码复杂度。如果用户输入密码不满足复杂度要求或者密码错，最多将在三次这种错误之后直接返回密码错误的提示，否则期间任何一次正确的密码验证都允许登录。需要指出的是，pam_cracklib.so是一个常用的控制密码复杂度的pam模块，关于其用法举例我们会在之后详细介绍。之后带pam_unix.so和pam_deny.so的两行配置项的意思与之前类似。都表示需要通过密码认证并对不符合上述任何配置项要求的登录请求直接予以拒绝。不过用户如果执行的操作是单纯的登录，则这部分配置是不起作用的。

第四部分主要将通过session会话类接口为用户初始化会话连接。其中几个比较重要的地方包括，使用pam_keyinit.so表示当用户登录的时候为其建立相应的密钥环，并在用户登出的时候予以撤销。不过该行配置的控制位使用的是optional，表示这并非必要条件。之后通过pam_limits.so限制用户登录时的会话连接资源，相关pam_limit.so配置文件是/etc/security/limits.conf，默认情况下对每个登录用户都没有限制。关于该模块的配置方法在后面也会详细介绍。

可见，不同应用程序通过配置文件在认证过程中调用不同的pam模块来定制具体的认证流程。其中我们不难看出，其实可以根据实际的需要对pam的配置文件进行修改以满足不同的认证需求，例如下面的例子：

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth    required    pam_env.so
auth    required    pam_tally.so onerr=fail deny=5
```

```
auth    sufficient  pam_unix.so nullok try_first_pass
auth    requisite   pam_succeed_if.so uid >= 500 quiet
auth    required    pam_deny.so

account required    pam_unix.so
account sufficient  pam_succeed_if.so uid < 500 quiet
account required    pam_permit.so

password    requisite pam_cracklib.so try_first_pass retry=3 minlen=10 lcr
password    requisite pam_passwdqc.so use_first_pass enforce=everyone
password    sufficient pam_unix.so md5 remember=6 shadow nullok try_first_
password    required pam_deny.so

session     optional pam_keyinit.so revoke
session     required pam_limits.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond
session     required pam_unix.so
```

在其中就增加了对用户密码修改时复杂度的限制，用户多次错误输入密码之后的锁定限制以及用户使用密码历史等限制选项。

所以我们通过对上述system-auth配置文件的修改，模块的增加和选项的变化，从很大的程度上增加了用户登录验证的安全性要求。我们会在之后的文章中对该配置进行详细说明。

另外也一定需要注意，在整个的PAM配置文件当中，配置项以及模块调用的逻辑顺序非常关键。因为PAM是按照配置项的先后顺序来进行验证。错误的模块调用顺序很可能导致严重的安全问题甚至系统错误。所以对PAM配置进行修改的时候务必要考虑这一点。

四、常用的PAM模块介绍

PAM模块	结合管理类型	说明
pam_unix.so	auth	提示用户输入密码,并与/etc/shadow文件相比对.匹配返回0
	account	检查用户的账号信息(包括是否过期等).帐号可用时,返回0.
	password	修改用户的密码. 将用户输入的密码,作为用户的新密码更新shadow文件
pam_shells.so	auth account	如果用户想登录系统，那么它的shell必须是在/etc/shells文件中之一的shell
pam_deny.so	account auth password session	该模块可用于拒绝访问
pam_permit.so		模块任何时候都返回成功.

	auth account password session	
pam_securetty.so	auth	如果用户要以root登录时,则登录的tty必须在/etc/securetty之中.
pam_listfile.so	auth account password session	访问应用程的控制开关
pam_cracklib.so	password	这个模块可以插入到一个程序的密码栈中,用于检查密码的强度.
pam_limits.so	session	定义使用系统资源的上限，root用户也会受此限制，可以通过/etc/security/limits.conf或/etc/security/limits.d/*.conf来设定

五、PAM模式使用说明

1）pam_access.so模块

pam_access.so模块主要的功能和作用是根据主机名（包括普通主机名或者FQDN）、IP地址和用户实现全面的访问控制。pam_access.so模块的具体工作行为根据配置文件/etc/security/access.conf来决定。该配置文件的主体包含了三个字段——权限、用户和访问发起方。格式上是一个用""隔开的表。
第一个字段：权限（permission），使用"+"表示授予权限，用"-"表示禁止权限。
第二个字段：用户（user），定义了用户、组以及用"@"表示的在不同主机上的同名用户和同一主机上不同名用户。
第三个字段：访问发起方（origins），定义了发起访问的主机名称、域名称、终端名称。

而且/etc/security/access.conf文件提供了很多范例供修改时参考，并且都给出了具体的说明，例如：

```
#禁止非root用户通过tty1访问相关服务
#-:ALL EXCEPT root:tty1

#禁止除了wheel、shutdown以及sync之外的所有用户访问相关服务
#-:ALL EXCEPT wheel shutdown sync:LOCAL

#禁止wheel用户通过.win.tue.nl之外的其它它终端访问相关服务
#-:wheel:ALL EXCEPT LOCAL .win.tue.nl

# 禁止下面的用户从任何主机登录。其它用户可以从任意地方访问相关服务
#-:wsbscaro wsbsecr wsbspac wsbsym wscosor wstaiwde:ALL

# root用户允许通过cron来使用tty1到tty6终端访问相关服务
#+ : root : cron crond :0 tty1 tty2 tty3 tty4 tty5 tty6

# 用户root允许从下面的地址访问相关服务
#+ : root : 192.168.200.1 192.168.200.4 192.168.200.9
#+ : root : 127.0.0.1

```

```
# 用户root可以从192.168.201.网段访问相关服务
#+ : root : 192.168.201.

# 用户root可以从.foo.bar.org中任何主机访问相关服务
#+ : root : .foo.bar.org

# 用户root不允许从任何主机访问相关服务
#- : root : ALL

# 用户@nis_group和foo可以从任何主机访问相关服务
#+ : @nis_group foo : ALL

# 用户john只能从127.0.0.0/24来对本机相关服务进行访问
#+ : john : 127.0.0.0/24

# 用户john可以通过ipv4和ipv6的地址对本机相关服务进行访问
#+ : john : ::ffff:127.0.0.0/127

# 用户john可以通过ipv6的地址访问本机相关服务
#+ : john : 2001:4ca0:0:101::1

# 用户john可以通过ipv6的主机IP地址来访问本机
#+ : john : 2001:4ca0:0:101:0:0:0:1

# 用户john可以通过ipv6的IP地址和掩码来访问相关服务
#+ : john : 2001:4ca0:0:101::/64

# 开放所有用户对本机所有相关服务的访问
#- : ALL : ALL
```

示例说明（vsftp）：

如果要在网络内架设一个FTP服务器，而且在该FTP服务器上需要强制地指定某个用户只能通过某个IP地址登录，这个时候pam_access.so模块就派上用场了。假设我的FTP服务器是使用vsftp来构建的，具体操作是：

```
1）修改FTP服务器的/etc/pam.d/vsftpd文件，在调用account接口处插入"account    r
[root@centos6-test06 ~]# vim /etc/pam.d/vsftpd
#%PAM-1.0
session    optional     pam_keyinit.so    force revoke
auth       required pam_listfile.so item=user sense=deny file=/etc/vsftpd/
auth       required pam_shells.so
auth       include  password-auth
account    include  password-auth
account    required     pam_access.so                  //添加这一行内容
session    required     pam_loginuid.so
session    include  password-auth

上述配置表示当针对FTP访问执行用户类接口的时候会增加pam_access.so的认证。

2）修改/etc/security/access.conf配置文件，
```

```
在文件底部添加下面的两行：
[root@centos6-test06 ~]# vim /etc/security/access.conf
- : kevin : ALL EXCEPT 192.168.10.101
- : grace : ALL EXCEPT 192.168.10.102

前提是已经在系统上事先建立了kevin和grace两个用户。上面的配置表示：
kevin用户不能从192.168.10.101之外的任何客户端访问FTP服务器；
grace用户不能从192.168.10.102之外的任何客户端访问FTP服务器。

3）修改/etc/vsftpd/vsftpd.conf文件，禁用匿名登录：
[root@centos6-test06 ~]# vim /etc/vsftpd/vsftpd.conf
......
Anonymous_enable = NO

这样当重启vsftpd服务之后，用户kevin将只能从192.168.10.101访问ftp服务，而grace
所以当针对这种需求而且不想使用防火墙以及应用程序自带的认证机制的时候，通过pam_ac
```

2）pam_listfile.so

pam_listfile.so模块的功能和pam_access.so模块类似，目标也是实现基于用户/组，主机名/IP，终端的访问控制。不过它实现的方式和pam_access.so会稍微有些不同，因为它没有专门的默认配置文件。访问控制是靠pam配置文件中的控制选项和一个自定义的配置文件来实现的。而且除了针对上述访问源的控制之外，还能够控制到ruser，rhost，所属用户组和登录shell。所以有些用户认为它的功能似乎比pam_access.so更加灵活和强大一些。

对于pam_listfile.so的配置方法，可以参考vsftpd文件中对pam的调用方式。熟悉vsftpd的人都知道，在vsftpd默认配置中，root用户是不允许通过ftp方式直接访问FTP服务器的。这个功能实际上是由/etc/vsftpd/vsftpd.conf，/etc/vsftpd/ftpusers和/etc/pam.d/vsftpd共同控制的。 因为在/etc/pam.d/vsftpd中有这样的一行配置：

```
[root@centos6-test06 ~]# cat /etc/pam.d/vsftpd
......
auth       required pam_listfile.so item=user sense=deny file=/etc/vsftpd/
```

表示当用户试图登录FTP服务器的时候，会调用pam_listfile.so模块来验证用户是否可以登录，这里item=user表示访问控制是基于user即用户实现的。那么哪些用户可以登录呢？就是除了file选项所定义的/etc/vsftpd/ftpusers文件之外的用户，这是由另外一个选项sense=deny所决定的。而在/etc/vsftpd/vsftpd.conf中明确指定了对用户的认证需要通过/etc/pam.d/vsftpd中的配置调用pam模块：

```
[root@centos6-test06 ~]# cat /etc/vsftpd/vsftpd.conf |grep pam_service_nam
pam_service_name=vsftpd
```

而恰好root用户又在/etc/vsftpd/ftpusers文件中，所以这成为了制约root登录FTP服务器的一个必要条件（但不是唯一条件）。所以针对这种情况，要开放和允许root用户登录FTP的权限，至少有三种改法：

```
1）修改/etc/pam.d/vsftpd文件，将sense=deny改成sense=allow。这样会正好将情况反
2）修改/etc/pam.d/vsftpd文件，注释掉调用pam_listfile.so那行。这样FTP服务器在认
3）将root从/etc/vsftpd/ftpuser文件中注释掉；
```

不过需要注意的是，root用户比较特殊，因为它在vsftpd配置中的限制不仅仅来自于pam，vsftpd本身的配置中也对其做了限制。当我们看/etc/vsftpd/user_list文件的时候，还将会看到这样的配置说明：

```
[root@centos6-test06 vsftpd]# cat /etc/vsftpd/user_list
# vsftpd userlist
# If userlist_deny=NO, only allow users in this file
# If userlist_deny=YES (default), never allow users in this file, and
# do not even prompt for a password.
# Note that the default vsftpd pam config also checks /etc/vsftpd/ftpusers
# for users that are denied.
........
........
```

表示当vsftpd.conf中userlist_deny=NO的时候，系统将只允许user_list中的用户登录FTP服务器；如果userlist_deny=YES，情况将截然相反——此时user_list变成了黑名单，里面的用户将一概不允许登录FTP服务器。所以要彻底开放root登录FTP的权限，我们还要在/etc/vsftpd/vsftpd.conf中增加userlist_deny=YES或者注释掉user_list中的root。

不过不管怎么说，vsftpd中禁用root用户的直接登录是在绝大多数FTP服务器上默认的安全措施，所以开放root权限时应该慎重。

另外除了通过pam_listfile.so实现基于用户的访问控制之外，还可以实现基于其它条件的访问控制。这个可以具体看看pam_listfile.so模块的选项就会比较清楚了，使用pam_listfile.so模块配置的格式分为五个部分：分别是item、onerr、sense、file以及apply。 其中：
a）**item**=[tty|user|rhost|ruser|group|shell]：定义了对哪些列出的目标或者条件采用规则，显然，这里可以指定多种不同的条件。
b）**onerr**=succeed|fail：定义了当出现错误（比如无法打开配置文件）时的缺省返回值。
c）**sense**=allow|deny：定义了当在配置文件中找到符合条件的项目时的控制方式。如果没有找到符合条件的项目，则一般验证都会通过。
d）**file**=filename：用于指定配置文件的全路径名称。
e）**apply**=user|@group：定义规则适用的用户类型（用户或者组）。

而至于file文件的写法就简单了，每行一个用户或者组名称即可。所以，当需要对其它服务进行类似的访问控制的时候，就可以照葫芦画瓢。例如现在需要在SSH服务器上对ssh客户端实现基于用户的访问控制

示例说明一：
不允许bobo账号通过ssh方式登录。做法如下：

```
1）针对这种需求只需要更改/etc/pam.d/sshd文件，并在该文件中添加一行（一定要添加到
[root@centos6-test06 ~]# vim /etc/pam.d/sshd
auth required pam_listfile.so  item=user sense=deny file=/etc/pam.d/denyus
......

2）创建bobo账号
[root@centos6-test06 ~]# useradd bobo
[root@centos6-test06 ~]# passwd bobo
Changing password for user bobo.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.

3）建立文件/etc/pam.d/denyusers，并在文件中写入用户信息。
[root@centos6-test06 ~]# echo "bobo" > /etc/pam.d/denyusers
```

```
[root@centos6-test06 ~]# cat /etc/pam.d/denyusers
bobo

4）测试使用bobo账号通过ssh方式登录不上了
[root@centos6-test06 ~]# ssh -p22 bobo@localhost
bobo@localhost's password:
Permission denied, please try again.
bobo@localhost's password:

表示用户以ssh登录必须要通过pam_listfile.so模块进行认证，认证的对象类型是用户，来
这样在该条目添加到该文件之后，使用bobo账号从其它主机远程ssh访问服务器会出现密码错

再次强调，要注意pam模块使用的顺序，刚才的规则一定要添加到/etc/pam.d/sshd文件的a
```

示例说明二：

仅仅允许kevin用户可以通过ssh远程登录。做法如下：

在/etc/pam.d/sshd文件中添加一条（务必添加到文件的第一行！）：

auth required pam_listfile.so item=user sense=allow file=/etc/sshdusers onerr=succeed

```
[root@centos6-test06 ~]# cat /etc/pam.d/sshd
#%PAM-1.0
auth       required     pam_listfile.so item=user sense=allow file=/etc/sshdusers onerr=succeed
auth       required     pam_sepermit.so
auth       include      password-auth
account    required     pam_nologin.so
account    include      password-auth
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user context
session    required     pam_selinux.so open env_params
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      password-auth
```

添加两个用户kevin和grace

```
[root@centos6-test06 ~]# useradd kevin
[root@centos6-test06 ~]# passwd kevin
[root@centos6-test06 ~]# useradd grace
[root@centos6-test06 ~]# passwd grace
```

编辑file指定的文件，添加上一个用户kevin（这一步是关键）

```
[root@centos6-test06 ~]# echo "kevin" >/etc/sshdusers          //文件/etc/
```

然后验证，发现使用kevin账号能正常ssh登录，使用grace账号就不能正常ssh登录了！

```
[root@centos6-test06 ~]# ssh -p22 kevin@localhost
kevin@localhost's password:
Last login: Thu Mar 29 12:02:18 2018 from 192.168.10.206
```

```
[root@centos6-test06 ~]# ssh -p22 grace@localhost
grace@localhost's password:
Permission denied, please try again.

[root@centos6-test06 ~]# ssh -p22 root@localhost
root@localhost's password:
Permission denied, please try again.
```

注：此处如果root也使用ssh远程连接，也会受到pam_listfile.so限制的。

温馨提示：
如果发生错误，Linux-PAM 可能会改变系统的安全性。这取决于你自己的选择，你可以选择不安全(开放系统)和绝对安全(拒绝任何访问)。通常，Linux-PAM 在发生错误时，倾向于后者。任何的配置错误都可能导致系统整个或者部分无法访问。配置 Linux-PAM 时，可能遇到最大的问题可能就是 Linux-PAM 的配置文件/etc/pam.d/*被删除了。如果发生这种事情，你的系统就会被锁住。有办法可以进行恢复，最好的方法就是用一个备份的镜像来恢复系统，或者登录进单用户模式然后进行正确的配置。

3）pam_limits.so模块

pam_limits.so模块的主要功能是限制用户会话过程中对各种系统资源的使用情况。缺省情况下该模块的配置文件是/etc/security/limits.conf。而该配置文件的基本格式实际上是由4个字段组成的表，其中具体限制的内容包括：

```
Domain              type            item
用户名/组名         软/硬限制
                                    core—core文件大小 (KB)
                                    data—最大数据大小(KB)
                                    fsize—最大文件大小(KB)
                                    memlock—最大可用内存空间(KB)
                                    nofile—最大可以打开的文件数量
                                    rss—最大可驻留空间(KB)
                                    stack—最大堆栈空间(KB)
                                    cpu—最大CPU使用时间（MIN）
                                    nproc—最大运行进程数
                                    as—地址空间限制
                                    maxlogins—用户可以登录到系统最多次数
                                    locks—最大锁定文件数目
```

需要注意的是，如果没有任何限制可以使用"-"号，并且针对用户限制的优先级一般要比针对组限制的优先级更高。使用pam_limits.so模块的最常见的场景是在运行Oracle数据库的RHEL服务器中，因为一般Oracle数据库在安装之前，按照其官方文档的说明需要先对某些用户（Oracle）使用系统资源的情况进行限制。所以我们总是能够在Oracle数据库服务器的/etc/security/limits.conf文件中看到类似这样的配置：

```
[root@centos6-test06 ~]# vim /etc/security/limits.conf
.......
oracle           soft    nproc   2047
oracle           hard    nproc   16384
oracle           soft    nofile  1024
oracle           hard    nofile  65536
```

结合上面的配置文件说明，可知Oracle数据库需要对Oracle用户使用资源的情况进行一些限制，包括： oracle用户最大能开启的进程数不超过16384，最大能打开的文件数不超过65536。

至于soft和hard的区别，不同于磁盘配额中的软限制和硬限制。普通用户可以调整自己的soft limit但最高不能超过hard limit，而且除了root以外的普通用户也不能够随意更改hard limit。该调整完成之后一般可以使用ulimit命令查看。

顺便提一下，针对nofile，这个只是基于用户层面的限制和调整方法。基于系统层面的限制和调整方法是修改/etc/sysctl.conf文件，直接改fs.file-max参数，调整之后sysctl -p生效。

示例说明：

pam_limits.so模块也可以使用在对一般应用程序使用的资源限制方面。如果需要在SSH服务器上对来自不同用户的ssh访问进行限制，就可以调用该模块来实现相关功能。例如，当需要**限制用户bobo登录到SSH服务器时的最大连接数（防止同一个用户开启过多的登录进程）**。限制操作如下：

```
由于/etc/pam.d/system-auth中，默认就会通过pam_limits.so 限制用户最多使用多少系
[root@centos6-test06 ~]# cat /etc/pam.d/system-auth|grep limits.so
session     required      pam_limits.so

因此只需要在/etc/security/limits.conf文件中增加一行对bobo用户产生的连接数进行限
[root@centos6-test06 ~]# vim /etc/security/limits.conf
......
bobo              hard    maxlogins       2

最后测试
从客户端以bobo身份登录SSH服务器时，在客户端上可以打开两个控制台登录。
但当客户端开启第三个登录窗口的时候会被服务器拒绝，但其它用户不会受到限制。

注意：这样限制的只是从客户端以ssh方式登录次数的场景，如果从xshell登录，则不受限制
```

4）pam_rootok.so模块

一般情况下，pam_rootok.so模块的主要作用是使uid为0的用户，即root用户能够直接通过认证而不用输入密码。pam_rootok.so模块的一个典型应用是插入到一些应用程序的认证配置文件中，当root用户执行这些命令的时候可以不用输入口令而直接通过认证。比如说"su"命令，为什么当以root用户执行"su"切换到普通用户身份的时候是不需要输入任何口令而可以直接切换过去？当我们查看一下/etc/pam.d/su文件的内容就不会奇怪了。因为该文件的第一行就是：

```
[root@centos6-test06 ~]# cat /etc/pam.d/su
......
auth        sufficient  pam_rootok.so
```

而如果将该行配置注释掉的情况下，就会发现即便以root用户切换普通用户的时候仍然要求输入口令。

另外一种方法，只需要将上述的"sufficient"改成"required"即可。因为这样，pam_rootok.so模块的验证通过就成为了必要条件之一。

pam_rootok.so模块的另外一个应用是在chfn命令中。Chfn命令用于改变/etc/passwd中的用户的说明字段。当以root身份执行chfn命令修改用户信息的时候是不用输入密码的。但是以普通用户身份执行chfn则需要输入密码之后才能改变自己的用户说明。这实际上也是因为在/etc/pam.d/chfn配置文件中的第一行调用了pam_rootok.so的结果。

不过这里即便将该配置中的第一行注释掉，root用户通过chfn修改自己信息的时候仍然不需要使用密码。所以恐怕效果不是很明显。究其原因主要是很多PAM模块对root用户是不会产生限制的。

示例说明（禁用用户间使用su切换命令）：

su的缺点
1）不安全su工具在多人参与的系统管理中，并不是最好的选择，su只适用于一两个人参与管理的系统，毕竟su并不能让普通用户受限的使用；超级用户root密码应该掌握在少数用户手中。
2）麻烦：需要把root密码告知每个需要root权限的人。

可以在/etc/pam.d/su文件里设置禁止用户使用su命令
[root@centos6-test06 ~]# vim /etc/pam.d/su
auth sufficient pam_rootok.so
Uncomment the following line to implicitly trust users in the "wheel" group.
#auth sufficient pam_wheel.so trust use_uid
Uncomment the following line to require a user to be in the "wheel" group.
#auth required pam_wheel.so use_uid
.....

a）以上标红的两行是默认状态（即开启第一行，注释第二行），这种状态下是允许所有用户间使用su命令进行切换的！（或者两行都注释也是运行所有用户都能使用su命令，但root下使用su切换到其他普通用户需要输入密码；如果第一行不注释，则root使用su切换普通用户就不需要输入密码）
b）如果开启第二行，表示只有root用户和wheel组内的用户才可以使用su命令。
c）如果注释第一行，开启第二行，表示只有wheel组内的用户才能使用su命令，root用户也被禁用su命令。

5） pam_userdb.so模块

pam_userdb.so模块的主要作用是通过一个轻量级的Berkeley数据库来保存用户和口令信息。这样用户认证将通过该数据库进行，而不是传统的/etc/passwd和/etc/shadow或者其它的一些基于LDAP或者NIS等类型的网络认证。所以存在于Berkeley数据库中的用户也称为虚拟用户。

pam_userdb.so模块的一个典型用途就是结合vsftpd配置基于虚拟用户访问的FTP服务器。

相对于本地用户以及匿名用户来说，虚拟用户只是相对于FTP服务器而言才有用的用户，这些用户被严格地限定在pam_userdb数据库当中。所以虚拟用户只能访问FTP服务器所提供的资源，因而可以大大提高系统安全性。另外相对于匿名用户而言，虚拟用户必须通过用户名和密码才能够访问FTP的资源。这样也提高了对FTP用户下载的可管理性。

基于虚拟用户实现的vsftpd的原理基本上是这样一个过程：先定义一些专门针对FTP的虚拟用户，然后将用户信息加入到系统自带的数据库中（但不是passwd）从而生成一个访问FTP的虚拟用户列表，这里使用的数据库是db4也就是Berkeley DB。然后可以通过使用pam_userdb.so模块来调用该数据库存储用户信息以及实现FTP用户认证。当然同时也可以在系统中通过对配置文件的定义和划分来实现对不同虚拟用户不同类型的访问控制。

FTP虚拟账号登录环境部署：http://www.cnblogs.com/kevingrace/p/5587140.html

6）pam_cracklib.so模块

pam_cracklib.so是一个常用并且非常重要的PAM模块。该模块主要的作用是对用户密码的强健性进行检测。即检查和限制用户自定义密码的长度、复杂度和历史等。如不满足上述强度的密码将拒绝用户使用。pam_cracklib.so比较重要和难于理解的是它的一些参数和计数方法，其常用参数包括：

```
debug:      将调试信息写入日志；
type=xxx:   当添加/修改密码时，系统给出的缺省提示符是"New UNIX password:"以及"R
retry=N:    定义登录/修改密码失败时，可以重试的次数；
Difok=N:    定义新密码中必须至少有几个字符要与旧密码不同。但是如果新密码中有1/2以
minlen=N:   定义用户密码的最小长度；
dcredit=N:  定义用户密码中必须至少包含多少个数字；
ucredit=N:  定义用户密码中必须至少包含多少个大写字母；
lcredit=N:  定义用户密码中必须至少包含多少个小些字母；
ocredit=N:  定义用户密码中必须至少包含多少个特殊字符（除数字、字母之外）；

特别要注意：
当N>0时，N代表新密码中最多可以有N个指定的字符！！
当N<0时，N代表新密码中最少可以有N个指定的字符！！
```

```
14
15  同时建议重启系统使之生效!
```

/etc/pam.d/login文件里包含了/etc/pam.d/system-auth文件的配置

```
1  [root@centos6-test06 ~]# cat /etc/pam.d/login|grep system-auth
2  auth       include      system-auth
3  account    include      system-auth
4  password   include      system-auth
5  session    include      system-auth
```

如下看pam_cracklib.so的一个应用实例：在/etc/pam.d/system-auth中使用pam_cracklib.so来限制用户修改自己密码时必须满足一定的强健性要求。

```
1  [root@centos6-test06 ~]# cat /etc/pam.d/system-auth
2  #%PAM-1.0
3  # This file is auto-generated.
4  # User changes will be destroyed the next time authconfig is run.
5  auth        required      pam_env.so
6  auth        sufficient    pam_unix.so nullok try_first_pass
7  auth        requisite     pam_succeed_if.so uid >= 500 quiet
8  auth        required      pam_deny.so
9
10 account     required      pam_unix.so
11 account     sufficient    pam_localuser.so
12 account     sufficient    pam_succeed_if.so uid < 500 quiet
13 account     required      pam_permit.so
14
15 password    requisite     pam_cracklib.so try_first_pass retry=3 type=
16 password    sufficient    pam_unix.so md5 shadow nullok try_first_pass use
17 password    required      pam_deny.so
18
19 session     optional      pam_keyinit.so revoke
20 session     required      pam_limits.so
21 session     [success=1 default=ignore] pam_succeed_if.so service in crond
22 session     required      pam_unix.so
23
24 -----------------------------------------------------------------------
25 从上面使用pam_cracklib.so的策略看，要求用户修改密码时必须要满足9位，并且密码中至
26
27 但是实际上像minlen和所有credit所对应的数值可以是非0之外的正负整数，那么这些数值至
28 很多人将其简单地理解为某一类字符的位数，其实远远没有那么简单。
29
30 下面我们对这些数值和关系做一个简短的说明：
31
32 首先要明确整个环境中密码的长度要满足下面的计算公式：
33 计算公式：
34 最小密码长度（minlen）应该小于或者等于 dcredit+ucredit+lcredit+ocredit+其它分
35
```

```
注意：
*credit=-1表示至少有一个的意思
*credit=N（N表示当满足条件的时候加N分，例如dcredit=2表示一个数字加2分，两个数字
所以这里minlen其实更准确的表述应该是mincredit。

所以在下面的例子中，当pam_cracklib.so的参数按如下方式指定：
password    requisite     pam_cracklib.so try_first_pass retry=3 minlen=12

那么当用户执行命令修改密码的时候：
[bobo@centos6-test06 ~]$ passwd
Changing password for user bobo.
Changing password for bobo
(current) UNIX password:
New UNIX password:                    输入密码"1\=poiuyt"    不成功       1个数

此时密码有一个数字，2分，其它的字符每个1分，总共10分，不满足minlen的位数需求，所
BAD PASSWORD: is too simple
New UNIX password:                    输入密码"12\=poiuyt"    成功      2个数字
Retype new UNIX password:

此时密码有两个数字，4分，其它的字符每个1分，总共12分，满足minlen的位数需求，所以
New UNIX password:                    输入密码"1\=poiuytre"    成功      1个数
Retype new UNIX password:

此时密码有1个数字，2分，其它的字符每个1分，总共12分，满足minlen的位数需求，所以密

因此通过上述的配置基本可以得出这样的结论：

当某类credit为正数N的时候，表示密码中该类字符一个可以加N分；当某类credit为负数N的

所以当输入的密码所有的字符总分大于或者等于minlen，并且满足所有credit的要求，该密

所以pam_cracklib.so模块在系统安全管理策略和管理中的用途是非常重要和广泛的。
```

7）pam_pwhistroy.so模块

pam_pwhistory.so模块也是一个常用模块，一般辅助pam_cracklib.so，pam_tally.so以及pam_unix.so等模块来加强用户使用密码的安全度。不过pam_pwhistory.so模块起的是另一类的作用，即专门为用户建立一个密码历史档案，防止用户在一定时间内使用已经用过的密码。

示例说明：（特别注意：/etc/pam.d/system-auth下的配置针对的是普通用户，在root用户下是无效的）
当需要限定用户在90天之内不能重复使用以前曾经使用过的10个密码，那么具体操作方法是去修改/etc/pam.d/system-auth文件，在password接口处增加：

```
[root@centos6-test06 ~]# vim /etc/pam.d/system-auth
......
password  required  pam_cracklib.so  retry=3  password  required  pam_pwhi

-----------------------------------------------------------------
此时用户使用过的密码将会记录到/etc/security/opasswd文件中。但是pam_pwhistory.s
```

```
所以上述的90天是无法配置的。一个简单的解决方法就是当90天左右的时候，手动清空一次c

当然，如果要实现同样的功能除了pam_pwhistory.so模块之外还有其它的办法。比较常用的
具体方法是修改/etc/pam.d/system-auth文件，给pam_unix.so模块里加上remember=10
[root@centos6-test06 ~]# vim /etc/pam.d/system-auth
.......
password required pam_unix.so md5 remember=10 use_authtok

这样系统将同样记住10个已经使用的密码。

不过此时/etc/security/opasswd文件因为记录了N个使用过的密码，所以安全性就十分关
[root@centos6-test06 ~]# touch /etc/security/opasswd
[root@centos6-test06 ~]# chown root:root /etc/security/opasswd
[root@centos6-test06 ~]# chmod 600 /etc/security/opasswd
```

==========总结几个PAM模块比较常见的实操案例==========

1）怎样才能强迫用户设置的密码不能与过去3次内的密码重复？（特别注意：/etc/pam.d/system-auth下的配置针对的是普通用户，在root用户下是无效的）

```
修改/etc/pam.d/system-auth,增加pam_unix.so的参数，如下：
[root@centos6-test06 ~]# vim /etc/pam.d/system-auth
......
password    sufficient    pam_unix.so md5 shadow nullok try_first_pass use
```

2）如何要求用户设置的密码必须至少包含5个数字和3个特殊符号？

```
修改/etc/pam.d/system-auth，在password使用pam_cracklib.so设置的最后附加 dcre
[root@centos6-test06 ~]# vim /etc/pam.d/system-auth
......
password    requisite     pam_cracklib.so try_first_pass retry=3 dcredit=-

------------------------------------------------------------------------
同时注意：
密码中的n个数字和n个特殊字符不能全部放在一起！！
```

3）如何限制kevin用户最多同时登陆4个？（同时可以限制root用户）

```
这需要pam_limits.so模块。由于/etc/pam.d/system-auth中，默认就会通过pam_limits
[root@centos6-test06 ~]# cat /etc/pam.d/system-auth|grep limits.so
session     required      pam_limits.so

因此只需要在/etc/security/limits.conf 中加入以下内容：
[root@centos6-test06 ~]# vim /etc/security/limits.conf
......
kevin        hard       maxlogins       4
```

4）某用户连续登陆失败2次就锁定该账号，禁止登陆？（默认只能尝试登录三次，由retry=N决定的）

```
现在很多地方都有限制用户登录的功能，Linux也是如此，当你登录失败多次后就可以限制用
Linux有一个pam_tally2.so的PAM模块，来限定用户的登录失败次数，如果次数达到设置的

编译PAM的配置文件
[root@centos6-test06 ~]# vi  /etc/pam.d/login    //在#%PAM-1.0下面添加一行
#%PAM-1.0
auth       required     pam_tally2.so  onerr=fail deny=2 unlock_time=300 e
......

参数解释：
onerr=fail          表示定义了当出现错误（比如无法打开配置文件）时的缺省返回值；
even_deny_root      表示也限制root用户；
deny                表示设置普通用户和root用户连续错误登陆的最大次数，超过最大次
unlock_time         表示设定普通用户锁定后，多少时间后解锁，单位是秒；
root_unlock_time    表示设定root用户锁定后，多少时间后解锁，单位是秒；

此处使用的是 pam_tally2模块，如果不支持pam_tally2，可以使用pam_tally模块。
另外，不同的pam版本，设置可能有所不同，具体使用方法，可以参照相关模块的使用规则。

特别注意：
一定要将内容添加到#%PAM-1.0的下面，即在第二行添加内容，一定要写在前面！！
如果写在后面，虽然用户被锁定，但是只要用户输入正确的密码，还是可以登录的！

----------------------------------------
也可以将上面的内容添加到/etc/pam.d/password-auth文件中，因为/etc/pam.d/login文
[root@centos6-test06 ~]# vim /etc/pam.d/system-auth      //同样添加到auth区
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth       required     pam_tally2.so  onerr=fail deny=2 unlock_time=300 e

================================================================================
特别注意：
上面的配置只是限制了用户从tty终端登录，而没有限制远程ssh登录，如果想限制远程登录，
[root@centos6-test06 ~]# vim /etc/pam.d/sshd            //同样添加到auth区域
#%PAM-1.0
auth       required     pam_tally2.so  onerr=fail deny=2 unlock_time=300 e

这样的话，使用ssh远程登录的时候，连续输入两次错误密码，就会被锁定了！
如果输入错误次数没有达到deny设置的次数，再输入正确密码就可以登录。


验证：
[bobo@centos6-test06 ~]$ ssh -p22 kevin@localhost
kevin@localhost's password:                         //第一次输入密码错误
Permission denied, please try again.
kevin@localhost's password:                         //第二次输入密码错误，此时已
```

```
Permission denied, please try again.
kevin@localhost's password:                          //接着第三次再输入正确密码，
Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).

因为被锁定300s了！在这个时间内，就算再输入正确密码也是无法顺利登录到机器的！
[bobo@centos6-test06 ~]$ ssh -p22 kevin@localhost
kevin@localhost's password:
Permission denied, please try again.

接着可以解锁！！
查看用户登录失败的次数。由下面命令可知，已经输错3次了
[root@centos6-test06 ~]# pam_tally2 --user kevin
Login           Failures Latest failure     From
kevin                3    03/30/18 07:35:49  ::1

解锁指定用户
[root@centos6-test06 ~]# pam_tally2 -r -u kevin
Login           Failures Latest failure     From
kevin                3    03/30/18 07:35:49  ::1
[root@centos6-test06 ~]# pam_tally2 --user kevin
Login           Failures Latest failure     From
kevin                0

接着，就可以输入正确密码登录机器了
[bobo@centos6-test06 ~]$ ssh -p22 kevin@localhost
kevin@localhost's password:
Last login: Fri Mar 30 07:40:04 2018 from ::1
[kevin@centos6-test06 ~]$
```

5）如何限制root只能从kevin.com这台计算机使用ssh远程登陆?

```
由于ssh服务器的程序文件使用sshd,而sshd刚好支持PAM，验证如下：
[root@centos6-test06 ~]# ldd /usr/sbin/sshd | grep libpam.so
    libpam.so.0 => /lib64/libpam.so.0 (0x00007f36f254d000)

修改/etc/pam.d/sshd,加入第二行，如下：
[root@centos6-test06 ~]# vim /etc/pam.d/sshd
auth       include      system-auth
account    required     pam_access.so accessfile=/etc/deny_sshd
account    required     pam_nologin.so

创建/etc/deny_sshd文件
[root@centos6-test06 ~]# touch /etc/deny_sshd
[root@centos6-test06 ~]# vim /etc/deny_sshd
-:root:ALL EXCEPT kevin.com
```