Collection Certifications

Préparation à la certification **LPIC-2**

LINUX

3ième édition

EXAMENS LPI 201 et LPI 202

35 Travaux pratiques
127 Questions-réponses

OFFERT :
UN EXAMEN BLANC en ligne
avec réponses commentées et détaillées

Sébastien BOBILLIER
Philippe BANQUET

eni éditions

Les éléments à télécharger sont disponibles à l'adresse suivante :
http://www.editions-eni.fr
Saisissez la référence ENI du livre CE3C2LIN dans la zone de recherche et validez.
Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Descriptif

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

Chapitre 8 Serveurs web

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

Chapitre 9 Messagerie

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

© Editions ENI - All rights reserved

Chapitre 5

© Editions ENI - All rights reserved

Pré-requis

Les connaissances acquises lors de la certification LPI niveau 1, notamment :

☒Connaître la structure du fichier /etc/passwd.

☒Connaître l'existence et le principe du fichier hosts.

Objectifs

À la fin de ce chapitre, vous serez en mesure de :

☒Interpréter une configuration NSS.

☒Comprendre l'authentification modulaire PAM.

☒Connaître les principaux modules PAM.

☒Modifier la configuration PAM pour permettre un changement du mode d'authentification.

☒Connaître le format de fichier LDIF.

☒Interroger un annuaire LDAP.

☒Gérer les mots de passe dans un annuaire OpenLDAP.

☒Ajouter ou modifier des éléments d'un annuaire OpenLDAP.

☒Configurer l'authentification d'un système Linux sur un annuaire OpenLDAP.

A. Évolution de l'authentification

1. Les premiers systèmes Unix et le fichier passwd

a. Mots de passe dans le fichier /etc/passwd

Depuis le début de leur existence, les systèmes Unix utilisent le fichier **/etc/passwd** comme base de comptes des utilisateurs. Ce fichier est utilisé naturellement pour les ouvertures de session sur le système. Comme son nom l'indique encore, il contenait en plus des identifiants utilisateurs leurs mots de passe chiffrés. Si des éléments logiciels autres que l'ouverture de session ont besoin des informations de compte (connexion ftp, ouverture de session distante, etc.), ils vont également consulter ce fichier. Dans cette situation originelle simple, on a affaire à une base de comptes unique et des applications multiples qui exploitent cette base de comptes. Toutes les applications doivent reconnaître le format de cette base d'information.

b. Mots de passe dans le fichier /etc/shadow

Avec l'évolution des techniques d'attaques des mots de passe, le besoin est venu de placer les mots de passe dans un fichier non accessible aux utilisateurs ordinaires. Ils sont alors stockés dans un fichier **/etc/shadow** fermé aux utilisateurs. Les paramètres d'authentification avec shadow sont gérés par un fichier **/etc/login.defs**. Les paramètres présents par défaut dans ce fichier sont en général satisfaisants.

Gestion des erreurs d'authentification dans le fichier login.defs

Parmi les nombreux paramètres du fichier login.defs, ceux concernant le login sont les plus fréquemment modifiés.

```
toto@ubuntu:~$ grep LOGIN /etc/login.defs
LOGIN_RETRIES      5
LOGIN_TIMEOUT      60
toto@ubuntu:~$
```

2. D'autres bases d'informations

Pour la consultation des éléments d'identification, la situation s'est compliquée quand il a fallu intégrer d'autres bases de comptes, différentes du fichier **passwd** et surtout plus complexes. Ces bases d'identités sont souvent centralisées, comme c'est le cas pour NIS (*Network Information Server*) ou LDAP (*Lightweight Directory Access Protocol*). La première solution envisagée fut naturellement de réécrire les programmes qui exploitaient initialement le fichier **/etc/passwd** afin qu'ils soient capables de consulter les bases centralisées sur le réseau. Cette méthode manquait cruellement de souplesse, puisqu'elle obligeait à reprendre beaucoup de programmes en profondeur à chaque fois qu'une modification était apportée au mode de stockage des bases centralisées.

3. NSS

NSS (*Name Service Switch*) est une première réponse à la multiplicité des bases d'information locales ou centralisées. NSS a pour objet de normaliser la résolution de noms au sein d'un système. NSS permet de résoudre un nom en une autre information associée, comme par exemple un nom d'utilisateur et son uid, un nom de groupe et son gid, ou encore un nom d'hôte et son adresse IP.

© Editions ENI - All rights reserved

Dans un fonctionnement NSS, un fichier **/etc/nsswitch.conf** détermine pour différents types de résolutions la source d'information à privilégier, et les applications ayant besoin de ces informations vont consulter les sources dans l'ordre imposé par le fichier **nsswitch.conf**. La résolution s'appuie alors sur des bibliothèques NSS (**libnss_*X*.so** où *X* représente le service de résolution employé), et les applications n'ont pas besoin de connaître directement la méthode de résolution employée.

Format du fichier nsswitch.conf

résolution: source_1 source_n

nsswitch.conf : format du fichier	
résolution	Le type de résolution à effectuer.
source_1	Obligatoire. La première source de résolution à employer.
source_n	Facultatif. La ou les autres sources de résolution possibles à utiliser après la première.

Exemple de fichier nsswitch.conf

On voit dans cet exemple que les résolutions de type passwd, group et shadow feront leur résolution grâce à la bibliothèque libnss_compat.so, alors que la résolution de noms d'hôtes se fera par les bibliothèques libnss_files.so et libnss_dns.so. Ce qui veut dire que les éléments d'identification des utilisateurs seront trouvés dans les fichiers locaux de /etc, alors que la résolution de noms d'hôtes s'appuiera d'abord sur le fichier local (/etc/hosts) avant de se reporter sur un service dns.

```
passwd:         compat
group:          compat
shadow:         compat

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

☞*Sur un système Linux moderne, NSS n'est plus utilisé que pour des opérations d'identification, c'est-à-dire trouver des informations sur une identité. Tout ce qui relève de l'authentification est dévolu à un mécanisme plus élaboré : PAM.*

4. Modules d'authentification

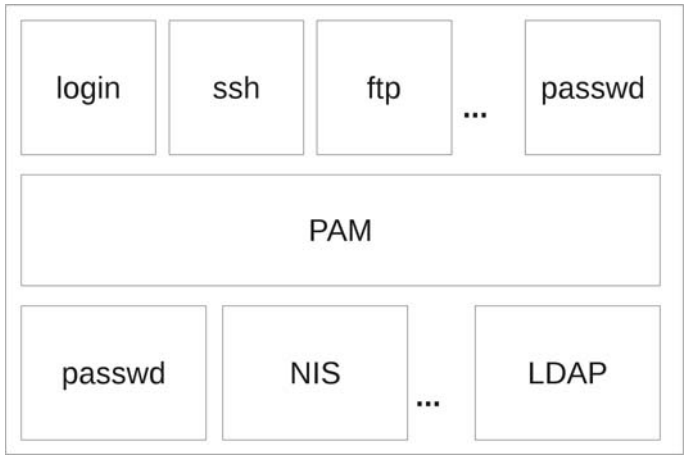
Si NSS représente déjà un progrès par rapport aux fichiers statiques utilisés dans les premiers temps, la révolution viendra avec PAM (*Pluggable Authentication Module*). PAM est un mécanisme complémentaire de NSS qui assure une authentification sur mesure par l'exécution de modules au choix de l'administrateur.

Lors d'une ouverture de session Linux, l'utilisateur va présenter un identifiant et un mot de passe. Grâce à la résolution NSS, on en déduira les identifiants uid/gid, ainsi que les autres paramètres nécessaires (date d'expiration, etc.). PAM de son côté va en fonction de sa configuration exécuter des modules pour assurer l'authentification mais aussi éventuellement pour effectuer certaines tâches liées à l'ouverture de session, comme la définition de variables par exemple.

B. PAM

1. Le principe

PAM se positionne en interface entre les applications et les méthodes d'authentification.



Le principal objectif de PAM est de proposer une couche d'abstraction entre les applications et les méthodes d'authentification. Ainsi, une application qui se veut souple et évolutive quant aux méthodes d'authentification qu'elle emploie n'aura d'autre besoin que d'être compatible avec PAM. Cela signifie qu'elle devra être capable de s'adresser à la couche d'authentification PAM, et le reste ne la regarde pas. En parallèle, les procédés d'authentification quels qu'ils soient, doivent être exploitables par la mécanique PAM.

Une application demande à PAM si un utilisateur peut se connecter. PAM en fonction de sa configuration, appelle des modules fonctionnels qui vont exploiter une méthode d'authentification. Si le résultat est positif (l'utilisateur a fourni les bons éléments d'authentification), PAM renvoie l'autorisation de connexion à l'application.

PAM a un autre avantage. Nous venons de voir que la demande d'authentification entraînait le chargement de modules. Il se trouve que le nombre de ces modules n'est pas limité et qu'ils peuvent être cumulés. Il est donc tout à fait possible de demander une double authentification selon deux méthodes différentes. De plus, on peut profiter de la séquence d'authentification sous PAM pour provoquer le chargement de bibliothèques sans rapport avec l'authentification. De nombreuses actions peuvent donc être gérées dès l'authentification réussie.

© Editions ENI - All rights reserved

En résumé : lors de la demande d'authentification, des modules PAM sont chargés en fonction d'un fichier de configuration, et ces modules provoquent certaines actions, relevant de l'authentification proprement dite ou d'autres actions.

2. Les modules PAM

a. Les principaux modules PAM

Les modules PAM, appelés lors des opérations d'authentification sont nombreux et d'usages variés. Certains d'entre eux sont néanmoins rencontrés très fréquemment et leur existence est à connaître. D'autres sont plus ou moins fréquents selon les distributions, mais connaître leur fonctionnement et leurs objectifs permet de mieux comprendre la mécanique et la philosophie de PAM.

Ces modules sont dans des fichiers dont l'emplacement normalisé est **/lib/security**.

Principaux modules PAM	
pam_securetty.so	Interdit le login par le compte root excepté sur les terminaux listés dans /etc/securetty.
pam_nologin.so	Si le fichier /etc/nologin existe, affiche son contenu à toute tentative d'ouverture de session et interdit le login à tout autre que root.
pam_env.so	Déclare des variables d'environnement lues dans /etc/environnement ou dans le fichier donné en référence par le paramètre « envfile= ».
pam_unix.so	Permet l'authentification par la méthode traditionnelle des fichiers /etc/passwd et /etc/shadow.
pam_deny.so	Voie de garage. Est généralement exécuté si aucun autre module n'est exécuté avec succès.
pam_permit.so	Renvoie un retour positif inconditionnellement.
pam_limits.so	Affecte certaines limitations fonctionnelles à des utilisateurs ou des groupes en fonction des données du fichier /etc/security/limits.conf.
pam_cracklib.so	S'assure que le mot de passe employé présente un niveau de sécurité suffisant.
pam_selinux.so	Si selinux est activé sur le système, ce module va s'assurer que le shell sera bien exécuté dans le contexte de sécurité adéquat.
pam_lastlog.so	Affiche les informations sur la dernière ouverture de session réussie.
pam_mail.so	Vérifie la présence de nouveaux mails pour un utilisateur (messagerie interne).