draft-behera-ldap-password-policy-10

Stratégie de mot de passe pour les annuaires LDAP

La stratégie de mot de passe comme décrit dans ce document est un jeu de règles qui contrôle comment les mots de passes sont utilisés et administrés dans les annuaires basés sur LDAP. Pour améliorer la sécurité les annuaires LDAP et rendre la tâche des programmes de cracking de mot de passe plus difficile. Cet règles sont faites pour s'assurer que les utilisateurs changent leur mot de passe périodiquement, et que ces mots de passe soient soumis à des obligations, et une restriction sur la réutilisation des anciens mots de passe.

Application des stratégies de mot de passe

La stratégie de mot de passe définie dans ce document peut être appliquée à tout attribut maintenant un mot de passe utilisateur utilisé pour les opérations de bind LDAP. Dans ce document, le terme utilisateur représente toute application LDAP cliente qui a une identité dans l'annuaire.

Cette stratégie est typiquement appliquée à l'attribut userPassword dans le cas de la méthode d'authentification simple bind ou dans le cas d'un mot de passe basé sur une authentification SASL tel que CRAM-MD5 et DIGEST-MD5.

La stratégie décrite dans ce document assume que l'attribut du mot de passe maintient une simple valeur. Aucune considération n'est faite pour les annuaires ou systèmes qui permettent à un utilisateur de maintenir des attributs de mot de passe multi-valués.

Les implémentations de serveur peuvent inclure une stratégie interne où certaines identités (tels que les administrateurs d'annuaire) ne soient pas soumis à une stratégie de mot de passe. Dans ce cas, le mot de passe pour un administrateur d'annuaire n'expire jamais ; le compte n'est jamais bloqué, etc.

Stratégie d'utilisation de mot de passe

Cette section décris la stratégie utilisée quand un mot de passe est utilisée. L'objectif général de cette stratégie est de réduire au minimum la menace d'intrus une fois un mot de passe utilisé.

Stratégie de validité de mot de passe

Ces mécanismes permettent de contrôler l'utilisation de compte indépendemment de toute stratégie d'expiration de mot de passe. La stratégie définis la période de temps absolue pour lequel un compte peut être utilisé. Cela permet à un administrateur de définir une date de début après laquelle un mot de passe devient valide, et une date de fin après laquelle le mot de passe est désactivé. Un mécanisme est également fournis pour définir la période de temps pour laquelle un compte peut rester non-utilisé avant d'être désactivé.

Limiter les mots de passe devinés

Pour empêcher un intrus de deviner le mot de passe d'un utilisateur, un mécanisme existe pour traquer le nombre de tentative d'authentification échouées, et agir lorsque la limite est atteinte. Cette stratégie consiste de nombreuses parties :

- Un compteur pour suivre le nombre de tentative d'authentification échouées.
- La quantité de temps à attendre dans la première authentification échouée.
- La quantité maximum de temps à attendre pour les échecs suivants
- Un timeframe dans lequel la limite de tentative d'authentification échouées doit être atteinte avant qu'une action soit prise.
- Une limite configurable de tentative d'authentification échouées.
- L'action a prendre quand la limite est atteinte. L'action sera soit "rien", ou le compte sera bloqué.
- Une quantité de temps durant lequel le compte est bloqué.

Noter qu'utiliser la fonctionnalité de blocage de compte est une aide précieuse pour les attaques DOS sur les comptes utilisateurs. Cette fonctionnalité est découragée en faveur d'un délai entre les différentes tentatives. Le délai va doubler à chaque tentative, "x<int>" pour les multiplicateurs constant, "^<int>" pour les géométriques.

Stratégie de modification de mot de passe

Cette section décris la stratégie appliquée quand un utilisateur modifie son mot de passe. Le focus général est de s'assurer que lorsqu'un utilisateur ajoute ou change son mot de passe, la sécurité et l'efficacité de ce mot de passe est maximisé. Dans ce document, le terme "opération de modification de mot de passe" réfère à toute opération qui est utilisée pour ajouter ou modifier un attribut de mot de passe. Souvent cela est fait en mettant à jours l'attribut du mot de passe durant une opération d'ajout ou de modification, mais peut être faite par d'autres moyens tels que les opérations étendues.

Expiration, alerte et période de grâce

Si un mot de passe est fréquemment changé, les chances que le compte de cet utilisateur soit cassé sont minimisés. Les administrateurs de stratégie de mot de passe peuvent déployer une stratégie de mot de passe qui force les mots de passe à expirer après un temps donné, ce qui force les utilisateurs à changer leur mot de passe périodiquement.

Il doit cependant y avoir un moyen par lequel les utilisateurs soient informés de cette nécessité de changer leur mot de passe avant d'être effectivement bloqué. Une ou les deux méthodes suivantes le gère :

- Un message d'alerte peut être retourné à l'utilisateur quelques temps avant que son mot de passe n'expire.
- L'utilisateur peut s'authentifier auprès de l'annuaire un certain nombre de fois après que son mot de passe ait expiré.

Historique de mot de passe

Quand la stratégie d'expiration de mot de passe est utilisée, un mécanisme additionnel est utilisé pour empêcher les utilisateurs de ré-utiliser leur précédent mot de passe. Pour cela, un historique les mots de passe est conservé. L'administrateur peut définir le nombre de mots de passe à conserver. Les utilisateurs n'ont pas le droit d'utiliser un mot de passe présent dans cette liste.

Âge minimum de mot de passe

Les utilisateurs peuvent outrepasser le mécanisme d'historique de mot de passe en effectuant une série de changement de mot de passe, jusqu'à ce que leur mot de passe favoris soit disponible. Pour empêcher cela, l'âge minimum d'un mot de passe peut être forcés.

Longueur minimum et qualité de mot de passe

Pour empêcher les utilisateurs de créer ou mettre à jours les mots de passe qui sont facile à deviner, une stratégie de qualité de mot de passe peut être employée. Cette stratégie consiste de 2 mécanismes généraux - s'assurer que le mot de passe est conforme aux critères de qualité et s'assurer qu'ils ont une longueur minimale. Forcer un mot de passe à être conforme avec une stratégie peut impliquer divers choses, incluant :

- Interdire les mots triviaux et bien connus
- Forcer un certain nombre de chiffres à utiliser
- Interdire les anagrammes du nom de l'utilisateur

L'implémentation de cette stratégie rencontre les problèmes suivants :

- Si le mot de passe à ajouter ou mettre à jours est chiffré par le client avant d'être envoyé, le serveur ne peut pas forcer cette stratégie.
- Il n'y a pas de définition spécifique de ce que signifie une vérification de gualité. Cela peut poser problèmes dans des milieux hétérogènes.

Mots de passes définis par les utilisateurs

Dans certains cas, il est désirable d'interdire les utilisateur d'ajouter ou modifier leur propres mots de passe. Cette stratégie rend cela possible.

Changement de mot de passe après réinitialisation

Cette stratégie force l'utilisateur à mettre à jours son mot de passe après qu'il ait été définis pour la première fois, ou a été réinitialisé.

Modification sûre

Vu que les annuaires sont communément utilisés, il devient fréquent que les clients se connecte à l'annuaire et laissent la connexion ouverte pour une période étendue. Cela permet à un attaquant d'effectuer des modifications sur un mot de passe utilisateur pendant que la machine de l'utilisateur est connectée. Cette stratégie force l'utilisateur à prouver son identité en spécifiant l'ancien mot de passe durant l'opération de modification de mot de passe.

Restriction de la stratégie de mot de passe

La stratégie de mot de passe définie dans ce document peut s'appliquer à tout attribut contenant un mot de passe. Les informations d'état de stratégie de mot de passe est maintenu dans l'entrée de l'utilisateur. Un serveur doit donc s'assurer que l'attribut du mot de passe est sujet à la stratégie de mot de passe, qu'il contient une et une seule valeur de mot de passe.

ObjectClass

pwdPolicy Contient les attributs définissant une stratégie de mot de passe pour un jeu d'utilisateur.

Attributs

pwdAttribute Maintient le nom de l'attribut auquel la stratégie de mot de passe s'applique

pwdMinAge Nombre de secondes entre chaque changement de mot de passe. Si non présent, assume 0

pwdMaxAge Nombre de secondes avant qu'un mot de passe n'expire. Si vide ou 0, le mot de passe n'expire jamais.

pwdInHistory Spécifie le nombre maximum de mot de passes stoqués dans l'attribut pwdHistory

pwdCheckQuality!!! Indique comment la qualité de mot de passe est vérifié lors d'un ajout ou d'une modification. Si non présent ou 0, aucune vérification n'est faite. 1, indique que le serveur vérifie la qualité, et s'il n'est pas capable de le faire, le mot de passe est accepté. 2 vérifie la qualité, et s'il n'est pas capable de le faire, retourne une erreur.

pwdMinLengh Quand la vérification de mot de passe est actif, cet attribut maintient le nombre de caractères minimum à utiliser dans un mot de passe.

pwdMaxLengh Quand la vérification de mot de passe est actif, cet attribut maintient le nombre de caractères maximum à utiliser dans un mot de passe.

pwdExpireWarning Spécifie le nombre maximum de secondes avant qu'un mot de passe soit près d'expirer et qu'un message d'alerte soit retourné à l'utilisateur. La valeur doit être plus petite que la valeur de pwdMaxAge.

pwdGraceAuthNLimit Spécifie le nombre de fois qu'un mot de passe expiré peut être utilisé pour s'authentifier. Si non présent ou à 0, l'authentification échoue.

pwdGraceExpiry Spécifie le nombre de secondes de la validité de la période de grâce.

pwdLockout À TRUE, indique que le mot de passe ne peut plus être utilisé pour s'authentifier après le nombre de tentative spécifié dans pwdMaxFailure.

pwdLockoutDuration Maintient le nombre de secondes que le mot de passe ne peut être utilisé pour s'authentifier due à trop grand nombre d'échec. À 0,le mot de passe ne peut plus être utilisé jusqu'à un reset administratif.

pwdMaxFailure Spécifie le nombre maximum d'authentification consécutives échouées.

pwdFailureCountInterval Maintient le nombre de secondes après lesquel le compteur d'échec est purgé, même quand aucune authentification réussie ne s'est produite. Si non présent ou à 0, ne peut être ré-initialisé qu'avec une authentification réussie.

pwdMustChange À TRUE, les utilisateurs doivent changer leur mot de passe la prochaine fois qu'ils s'authentifie.

pwdAllowUserChange Indique si les utilisateurs peuvent changer leur propre mot de passe, bien que cette opération reste sujette à contrôle d'accès. Cet attribut est conçus pour être utilisé en l'absence de mécanisme de contrôle d'accès.

pwdSafeModify Spécifie si le mot de passe existant doit être envoyé avec le nouveau lors du changement de mot de passe.

pwdMinDelay Spécifie le nombre de secondes d'attente après la première tentative d'authentification échouée. Si définis, pwdMaxDelay doit l'être également.

pwdMaxDelay Délai maximum après une authentification échouée. Ce temp est spécifié en pwdMinDelay utilisé comme point de départ et est doublé à chaque tentative jusqu'à atteindre pwdMaxDelay.

pwdMaxIdle Nombre de secondes qu'un compte peut rester non-utilisé avant d'être bloqué.

Les informations d'état de stratégie de mot de passe peuvent être maintenus pour chaque utilisateur. Ces informations sont localisées dans chaque entrée de l'utilisateur comme jeu d'attributs opérationnels. Ces attributs opérationnels sont : décris ci-dessous.

Vu que les stratégies de mot de passe peuvent s'appliquer à de nombreux attributs utilisés pour stocker les mots de passe, chaque attribut opérationnel doit avoir une option pour spécifier à quel pwdAttribute il s'applique. L'option est définie comme suit :

pwd-<passwordAttribute

où passwordAttribute est le nom court de l'attribut

pwdChangedTime ;pwd-userPassword : 20000103121520Z

Attributs Opérationnels

pwdChangedTime Spécifie la date du dernier changement de mot de passe. Utilisé par la stratégie d'expiration du mot de passe. Si non présent, le mot de passe n'expire jamais.

pwdAccountLockedTime Maintient la date à laquelle le compte utilisateur a été bloqué. Une valeur 000001010000Z signifie que le compte a été bloqué de manière permanente, et que seul un reset administratif peut débloquer le compte.

pwdFailureTime Maintient les date des derniers échec d'authentification.

pwdHistory Maintient l'historique des précédents mots de passe. Les valeurs de cet attribut sont au format (ABNF) :

```
pwdHistory = time "#" syntaxOID "#" length "#" data
time = GeneralizedTime
syntaxOID = numericoid ; the string representation of the
  ; dotted-decimal OID that defines the
  ; syntax used to store the password.
length = number ; the number of octets in data.
data = <octets representing the password in the format specified by syntaxOID>.
```

pwdGraceUseTime Maintient les périodes de grâce après qu'un mot de passe a expiré.

pwdReset Maintient un flag pour indiquer (à TRUE) que le mot de passe a été mis à jours par un administrateur et doit être changé par l'utilisateur.

pwdPolicySubentry Pointe vers une subentry pwdPolicy effectif sur l'objet.

pwdStartTime Spécifie la date à laquelle le mot de passe devient valide pour l'authentification.

pwdEndTime Spécifie la date après laquelle le mot de passe devient invalide pour l'authentification.

pwdLastSuccess Maintient la date de la dernière authentification réussie

ObjectClass

```
( 1.3.6.1.4.1.42.2.27.8.2.1
 NAME 'pwdPolicy'
 SUP top
 AUXILIARY
 MUST ( pwdAttribute )
 MAY ( pwdMinAge $ pwdMaxAge $ pwdInHistory $ pwdCheckQuality $
 pwdMinLength $ pwdMaxLength $ pwdExpireWarning $
 pwdGraceAuthNLimit $ pwdGraceExpiry $ pwdLockout $
 pwdLockoutDuration $ pwdMaxFailure $ pwdFailureCountInterval $
 pwdMustChange $ pwdAllowUserChange $ pwdSafeModify $
 pwdMinDelay $ pwdMaxDelay $ pwdMaxIdle ) )
```

Attributs

```
( 1.3.6.1.4.1.42.2.27.8.1.1
 NAME 'pwdAttribute'
 EQUALITY objectIdentifierMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.38 )

( 1.3.6.1.4.1.42.2.27.8.1.2
 NAME 'pwdMinAge'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )
```

```
( 1.3.6.1.4.1.42.2.27.8.1.3
 NAME 'pwdMaxAge'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.4
 NAME 'pwdInHistory'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.5
 NAME 'pwdCheckQuality'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.6
 NAME 'pwdMinLength'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.31
 NAME 'pwdMaxLength'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.7
 NAME 'pwdExpireWarning'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.8
 NAME 'pwdGraceAuthNLimit'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.30
 NAME 'pwdGraceExpire'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.9
 NAME 'pwdLockout'
```

```
 EQUALITY booleanMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.10
 NAME 'pwdLockoutDuration'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.11
 NAME 'pwdMaxFailure'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 ORDERING integerOrderingMatch
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.12
 NAME 'pwdFailureCountInterval'
 EQUALITY integerMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 ORDERING integerOrderingMatch
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.13
 NAME 'pwdMustChange'
 EQUALITY booleanMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.14
 NAME 'pwdAllowUserChange'
 EQUALITY booleanMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.15
 NAME 'pwdSafeModify'
 EQUALITY booleanMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.24
 NAME 'pwdMinDelay'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.25
 NAME 'pwdMaxDelay'
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.8.1.26
 NAME 'pwdMaxIdle'
```

```
 EQUALITY integerMatch
 ORDERING integerOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
 SINGLE-VALUE )
```

Attributs opérationnels

```
( 1.3.6.1.4.1.42.2.27.8.1.16
 NAME 'pwdChangedTime'
 DESC 'The time the password was last changed'
 EQUALITY generalizedTimeMatch
 ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 SINGLE-VALUE
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.17
 NAME 'pwdAccountLockedTime'
 DESC 'The time an user account was locked'
 EQUALITY generalizedTimeMatch
 ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 SINGLE-VALUE
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.19
 NAME 'pwdFailureTime'
 DESC 'The timestamps of the last consecutive authentication
 failures'
 EQUALITY generalizedTimeMatch
 ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.20
 NAME 'pwdHistory'
 DESC 'The history of user s passwords'
 EQUALITY octetStringMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.21
 NAME 'pwdGraceUseTime'
 DESC 'The timestamps of the grace authentication after the
 password has expired'
 EQUALITY generalizedTimeMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.22
```

```
 NAME 'pwdReset'
 DESC 'The indication that the password has been reset'
 EQUALITY booleanMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
 SINGLE-VALUE
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.23
 NAME 'pwdPolicySubentry'
 DESC 'The pwdPolicy subentry in effect for this object'
 EQUALITY distinguishedNameMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
 SINGLE-VALUE
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.27
 NAME 'pwdStartTime'
 DESC 'The time the password becomes enabled'
 EQUALITY generalizedTimeMatch
 ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 SINGLE-VALUE
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.28
 NAME 'pwdEndTime'
 DESC 'The time the password becomes disabled'
 EQUALITY generalizedTimeMatch
 ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 SINGLE-VALUE
 NO-USER-MODIFICATION
 USAGE directoryOperation )

( 1.3.6.1.4.1.42.2.27.8.1.29
 NAME 'pwdLastSuccess'
 DESC 'The timestamp of the last successful authentication'
 EQUALITY generalizedTimeMatch
 ORDERING generalizedTimeOrderingMatch
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.24
 SINGLE-VALUE
 NO-USER-MODIFICATION
 USAGE directoryOperation )
```

Contrôles utilisés pour la stratégie de mot de passe

Cette section détaille les contrôles utilisés lors de l'application des stratégies de mot de passe. Un contrôle de request est définie et est envoyé par un client avec une opération request pour déclencher un contrôle de réponse. Cette réponse contient diverses alertes et erreurs associés avec une stratégie de mot de passe.

Request Control

Ce contrôle peut être envoyé avec tout message de demande LDAP pour transmettre au serveur est prêt et peut traiter le contrôle de réponse décris dans ce document. Le **controlType** est **1.3.6.1.4.1.42.2.27.8.5.1**. Il n'y a pas de **controlValue**.

Response Control

Si le client a envoyé un contrôle **passwordPolicyRequest**, le serveur envoie ce contrôle avec les opérations de réponses suivantes : **bindResponse, modifyResponse, addResponse, compareResponse** et **extendedResponse**, pour informer des divers conditions, et peut l'envoyer avec d'autres opérations (dans le cas de l'erreur **changeAfterReset**). Le **controlType** est **1.3.6.1.4.1.42.2.27.8.5.1** et **controlValue** est le BER du type :

```
PasswordPolicyResponseValue ::= SEQUENCE {
 warning [0] CHOICE {
  timeBeforeExpiration [0] INTEGER (0 .. maxInt),
  graceAuthNsRemaining [1] INTEGER (0 .. maxInt) } OPTIONAL,
 error [1] ENUMERATED {
  passwordExpired (0),
  accountLocked (1),
  changeAfterReset (2),
  passwordModNotAllowed (3),
  mustSupplyOldPassword (4),
  insufficientPasswordQuality (5),
  passwordTooShort (6),
  passwordTooYoung (7),
  passwordInHistory (8) } OPTIONAL }
```

timeBeforeExpiration spécifie le nombre de secondes avant qu'un mot de passe n'expire.

graceAuthNsRemaining spécifie le nombre restant de fois qu'un utilisateur sera autorisé à s'authentifier une fois le mot de passe expiré.

passwordExpired signifie que le mot de passe a expiré et qu'il doit être changé.

passwordModNotAllowed est définis quand un utilisateur ne peut pas changer son mot de passe.

insufficientPasswordQuality est définis quand un mot de passe ne passe pas la vérification de la qualité

passwordTooYoung Définis si l'âge du mot de passe à modifier n'est pas suffisamment ancien

Stratégie de points de décision

La suite est une série de procédure utilisées pour effectuer des décisions de stratégie. Ces procédures sont généralement effectués par le serveur durant le traitement d'une opération.

Vérification de compte bloqué

Un statut TRUE est retourné pour indiquer que le compte est bloqué si une de ces conditions est rencontrée :

- La valeur de pwdAccountLockedTime est 000001010000Z
- L'heure courante est antérieure à pwdStartTime
- L'heure courante est postérieure à pwdEndTime

- L'heure courante est supérieur ou égal à la valeur de pwdLastSuccess ajouté à la valeur de pwdMaxIdle
- L'heure courante est inférieure à la valeur de pwdAccountLockedTime ajouté à la valeur de pwdLockoutDuration

Sinon un statut FALSE est retourné.

Vérification de changement de mot de passe

Un statut TRUE est retourné pour indiquer que le mot de passe doit être changé si toutes les conditions suivantes sont rencontrées :

- L'attribut pwdMustChange est à TRUE
- l'attribut pwdReset est à TRUE

Vérification d'expiration de mot de passe

Un statut de TRUE est retourné indiquant que le mot de passe a expiré si l'heure courante moins la valeur de pwdChangedTime est supérieur à la valeur de pwdMaxAge. Sinon, un statut de FALSE est retourné.

Vérification de AuthN Grace restant

Si l'attribut pwdGraceExpiry est présent, et que la date courante est supérieur au temps d'expiration du mot de passe plus la valeur pwdGraceExpiry, 0 est retourné. Si l'attribut pwdGraceUseTime est présent, le nombre de valeurs dans cet attribut soustrait de la valeur de pwdGraceAuthNLimit est retourné. Un résultat positif spécifie le nombre d'authentifications de grâce restant.

Vérification de date avant expiration

Si l'attribut pwdExpireWarning n'est pas présent un statut de 0 est retourné. Sinon, soustrait le temp stocké dans pwdChangedTime de l'heure courante pour arriver à l'âge du mot de passe. Si l'âge du mot de passe est supérieur à la valeur de pwdMaxAge, un status 0 est retourné. Soustrait la valeur de pwdExpireWarning de la valeur de pwdMaxAge pour arriver à l'âge d'alerte, la valeur de pwdMaxAge moins l'âge du mot de passe est retourné.

Vérification de blocage d'intrus

Un statut de TRUE indiquant qu'un intrus a été détecté est retourné si les conditions suivantes sont rencontrées :

- L'attribut pwdLockout est TRUE
- Le nombre de valeurs dans pwdFailureTime qui sont plus récent que pwdFailureCountInterval est supérieur ou égal à pwdMaxFailure.

Sinon, un statut de FALSE est retourné. Tout en effectuant cette vérification, les valeurs de pwdFailureTime qui sont maintenus par plus d'un pwdFailureCountInterval sont purgés et non comptés.

Vérification de délai d'intrusion

Si l'attribut pwdMinDelay est 0 ou non définis, 0 est retourné. Sinon, un délai est calculé basé sur le nombre de valeurs dans l'attribut pwdFailureTime. Si la valeur calculée est supérieur à pwdMaxDelay, la valeur pwdMaxDelay est retournée. Tout en effectuant cette vérification, les valeurs de pwdFailureTime qui sont maintenus par plus d'un pwdFailureCountInterval sont purgé et non comptés.

Vérification de mot de passe trop jeune

Si la vérification de changement de mot de passe retourne TRUE, cette vérifiction retourne FALSE, pour permettre le changement de mot de passe. Un statut de TRUE indique qu'il ne s'est pas écoulé suffisamment de temps depuis le dernier changement de mot de passe. TRUE est retourné si :

- La valeur de pwdMinAge est non-zéro et pwdChangedTime est présent
- La valeur de pwdMinAge est supérieur à l'heure courante moins la valeur de pwdChangedTime.

Sinon, un statut de FALSE est retourné.

Points de renforcements de stratégies du serveur

Le serveur devrait forcer l'attribut du mot de passe sujet à une stratégie de mot de passe tel que définis dans ce document, contenant une et une seule valeur. Note : Dans le cas où un seul mot de passe est stocké dans plusieurs formats est considéré comme une seule valeur de mot de passe.

Les scénarios dans les opérations suivantes assument que le client a attaché un contrôle passwordPolicyRequest à la demande. Dans le cas où passwordPolicyRequest n'a pas été envoyé, aucun contrôle passwordPolicyResponse n'est retourné. Toutes les autres instructions restent les même.

Authentification basé sur un mot de passe

Cette section contient les règles de stratégie utilisé pour valider un mot de passe. Les opérations qui valident les mots de passe incluent, entre autre, l'opération Bind et l'opération Compare où l'attribut à comparer contient un mot de passe. Noter que l'opération Compare n'authentifie pas un utilisateur, mais peut être utilisé par une application externe pour authentification.

Echec en cas de compte bloqué

Si le compte est bloqué, le serveur échoue l'opération avec le code 49 invalidCredentials dans le cas d'une opération Bind, le code 5 compareFalse dans le cas d'une opération Compare, etc. Le serveur peut définir cette erreur : accountLocked (1) dans le passwordPolicyResponse dans le champ contrôle du message.

Procédures de mot de passe validé

Si l'opération de validation indique que le mot de passe est validé, cet procédures sont suivie dans l'ordre :

- Supprime les attributs pwdFailureTime et pwdAccountLockedTime.

- Définis la valeur de pwdLastSuccess à l'heure courante

Mot de passe à changer

Si le mot de passe doit être changé, le serveur envoie au client une réponse avec un resultCode success (0), compareTrue (6), etc. Et inclus le passwordPolicyResponse dans le champ controls du message bindResponse avec l'alerte : changeAfterReset spécifié. Pour un bind, le serveur doit interdir toutes les autres opérations fournies par cet utilisateur excepté la modification du mot de passe, bind, unbind, abandon et StartTLS.

Mot de passe expiré

Si le mot de passe a expiré, le serveur retourne soit un succès ou un échec en fonction de l'état des authentification de grâce.

Authentification de grâce restant

S'il reste les authentification de grâce, le serveur ajoute une nouvelle valeur avec l'heure courante dans pwdGraceUseTime. Puis il envoie au client une réponse avec un succès, et inclus le passwordPolicyResponse dans la réponse avec l'alerte graceAuthNsRemaining définis au nombre d'authentification restante.

Plus d'authentification de grâce restante

S'il ne reste plus d'authentification de grâce, le serveur échoue l'opération, et inclus le passwordPolicyResponse dans controls avec l'erreur passwordExpired.

Alerte d'expiration

Si la vérification d'expiration de mot de passe réussie, le serveur envoie au client une réponse avec un resultCode approprié, et inclus le message timeBeforeExpiration.

Procédures d'échec AuthN

Si le processus d'authentification indique que le mot de passe échoue la validation du à des accréditifs invalides, ces procédures sont suivies :

Mise à jour d'état de stratégie Ajoute l'heure courant comme valeur de pwdFailureTime.

Détection d'intrusion Si la vérification d'intrus retourne TRUE, le serveur bloque le compte en définissant la valeur de pwdAccountLockedTime à l'heure courante. Une fois le compte bloqué, le serveur échoue l'opération avec l'erreur accountLocked.

Opérations de mise à jour du mot de passe

Parce que le mot de passe est stocké dans un attribut, divers opérations peuvent être utilisées pour créer ou modifier un mot de passe. Mais certains mécanismes ont été définis ou peuvent être définis tel que la rfc3062 (LDAP Password Modify). Tout en traitant une mise à jour d'un mot de passe, le serveur effectur les étapes suivantes :

Modification sûre Si pwdSafeModify est à TRUE et qu'il existe un mot de passe, le serveur s'assure que l'opération de mise à jour inclus le mot de passe existant de l'utilisateur. Si le mot de passe n'est pas fournis, le serveur échoue avec l'erreur mustSupplyOldPassword.

Changement après reset Si la vérification de changement de mot de passe à changer retourne true, le serveur s'assure que l'opération de mise à jours ne contient rien d'autre que la modification du mot de passe. Si d'autres modification existent, le serveur envoie un message avec le resultCode insufficientAccessRights avec l'erreur changeAfterReset.

Vérification des droits Vérifie si l'identité a les droits suffisants pour mettre à jours le mot de passe. Si l'utilisateur n'a pas les droits suffisants, le serveur retourne un resultCode insufficientAccessRights avec l'erreur passwordModNotAllowed.

Trop tôt pour la mise à jour Si la vérification d'âge minimum du mot de passe retourne true, le serveur envoie un resultCode constraintViolation avec l'erreur passwordTooYoung.

Qualité du mot de passe Vérifier la valeur de l'attribut pwdCheckQuality. Si la valeur est non zéro, le serveur :

- S'assure que le mot de passe respecte les critères de qualité définis par le serveur. Si le serveur ne peut pas vérifier la qualité du mot de passe, la valeur de pwdCheckQuality est évalué. Si le serveur peut vérifier la qualité du mot de passe et que la vérification échoue, le serveur envoie un constraintViolation avec l'erreur insufficientPasswordQuality.
- Vérifie la valeur de pwdMinLengh. Si la valeur est non zéro, il s'assure que le nouveau mot de passe a au moins cette longueur minimum. Si le serveur n'est pas capable de vérifier la longueur, la valeur de pwdCheckQuality est évaluée. Si le serveur peut vérifier la longueur, et qu'il échoue, retourne un constraintViolation avec l'erreur passwordTooShort.

Réutilisation invalide Si pwdInHistory est présent et sa valeur est non-zéro, le serveur vérifie si le mot de passe existe dans pwdHistory ou dans l'attribut du mot de passe. S'il est trouvé, le serveur envoie un constraintViolation avec une erreur passwordInHistory.

Mise à jours d'état de stratégie Si les étapes ont été complétés sans erreur, le serveur effectue les étapes suivantes sur l'entrée à l'heure courante :

- Si la valeur de pwdMaxAge ou pwdMinAge est non zéro, le serveur ajoute le précédent mot de passe (s'il en existait un) à l'attribut pwdHistory. Si le nombre de valeurs dans pwdHistory excède la valeur de pwdInHistory, le serveur supprime le mot de passe le plus ancien.
- Si la valeur dans pwdMustChange est TRUE et que la modification est effectuée par un administateur, l'attribut pwdReset est mis à TRUE. Sinon, pwdReset est supprimé de l'entrée.
- Les attributs pwdFailureTime et pwdGraceUseTime sont supprimés de l'entrée de l'utilisateur s'ils existent.

Autres opérations

Pour les opérations autre que bind, password update, unbind, abandon ou StartTLS, si la vérification de mot de passe à changer retourne true, le serveur envoie un insufficientAccessRights avec l'erreur changeAfterReset.

Client Policy Enforcement Points

Ces sections illustrent les scénarios possibles pour chaque opération LDAP et définis les types de réponses qui identifient cet scénarios. Les scénarios dans les opérations suivantes assument que le client a attaché un contrôle passwordPolicyRequest au message de demande de l'opération, et peut recevoir un contrôle passwordPolicyResponse dans le message de réponse. Dans le cas où le contrôle passwordPolicyRequest n'a pas été envoyé, aucun contrôle passwordPolicyResponse n'est retourné. Toutes les autres instructions restent les même.

bind

Pour chaque réponse bind reçus, le client vérifie le resultCode de bindResponse et recherche un contrôle passwordPolicyResponse pour déterminer si une des conditions est vrai et peut demander à l'utilisateur :

bindResponse.resultCode = insufficientAccessRights (50), passwordPolicyResponse.error = accountLocked (1)
La limite d'échec a été atteinte et le compte est bloqué. L'utilisateur doit retenter plus tard ou contacter un administrateur.
bindResponse.resultCode = success (0), passwordPolicyResponse.error = changeAfterReset (2) :
L'utilisateur s'authentifie pour la première fois depuis que l'administateur a définis le mot de passe et l'utilisateur doit changer son mot de passe.
bindResponse.resultCode = success (0), passwordPolicyResponse.warning = graceAuthNsRemaining :
Le mot de passe a expiré mais il reste des authentification de grâce. L'utilisateur doit le changer.
bindResponse.resultCode = invalidCredentials (49), passwordPolicyResponse.error = passwordExpired (0) :
Le mot de passe a expiré et il n'y a plus d'authentification de grâce. L'utilisateur doit contacter un administrateur pour réinitialiser son mot de passe.
bindResponse.resultCode = success (0), passwordPolicyResponse.warning = timeBeforeExpiration :
Le mot de passe de l'utilisateur va expirer dans n secondes

modify

Si l'application ou le client chiffre le mot de passe avant de l'envoyer dans une opération de modification (soit dans une opération modifyRequest ou un autre mécanisme de modification de mot de passe), il devrait vérifier les valeur de pwdMinLengh, pwdCheckQuality et devrait imposer ces stratégies.

Si l'opération medifyRequest a été utilisée pour changer le mot de passe, ou si un autre mécanisme est utilisé - tel que extendedRequest - la réponse peut contenir des informations pertinente pour la stratégie de mot de passe. Le client vérifie le resultCode de la réponse et vérifie que les conditions suivantes sont vraie et optionnellement notifie l'utilisateur des conditions.

pwdModResponse.resultCode = insufficientAccessRights (50), passwordPolicyResponse.error = mustSupplyOldPassword (4) :
L'utilisateur a tenté de changer son mot de passe sans spécifier l'ancien mot de passe et c'est requis par la stratégie.
pwdModResponse.resultCode = insufficientAccessRights (50), passwordPolicyResponse.error = changeAfterReset (2) :
L'utilisateur doit changer son mot de passe avant d'envoyer une autre demande LDAP
pwdModResponse.resultCode = insufficientAccessRights (50), passwordPolicyResponse.error = passwordModNotAllowed (3) :
L'utilisateur n'a pas les droit suffisants pour changer son mot de passe
pwdModResponse.resultCode = constraintViolation (19), passwordPolicyResponse.error = passwordTooYoung (7) :
Il est trop tôt pour changer le mot de passe
pwdModResponse.resultCode = constraintViolation (19), passwordPolicyResponse.error = insufficientPasswordQuality (5) :
La vérification de la qualité du mot de passe a échoué
pwdModResponse.resultCode = constraintViolation (19), passwordPolicyResponse.error = passwordTooShort (6) :
La longueur du mot de passe est trop courte
pwdModResponse.resultCode = constraintViolation (19), passwordPolicyResponse.error = passwordInHistory (8) :
Le mot de passe a déjà été utilisé, l'utilisateur doit en choisir un différent

add

Si un mot de passe est spécifié dans un addRequest, le client vérifie le resultCode de addResponse et vérifie le contrôle passwordPolicyResponse pour déterminer si une des conditions est vrai et pour notifier l'utilisateur en conséquence :

addResponse.resultCode = insufficientAccessRights (50), passwordPolicyResponse.error = passwordModNotAllowed (3) :
L'utilisateur n'a pas les droits suffisants pour ajouter son mot de passe
addResponse.resultCode = constraintViolation (19), passwordPolicyResponse.error = insufficientPasswordQuality (5) :
Le mot de passe a échoué la vérification de la qualité
addResponse.resultCode = constraintViolation (19), passwordPolicyResponse.error = passwordTooShort (6) :
La longueur du mot de passe est trop courte

compare

Quand une opération compare est utilisée pour comparer un mot de passe, le client vérifie le resultCode de compareResponse et vérifie le passwordPolicyResponse pour déterminer si une des conditions suivantes est vrai et peut notifier l'utilisation en conséquence. Ces conditions assument que le résultat de la comparaison était vrai :

compareResponse.resultCode = compareFalse (5), passwordPolicyResponse.error = accountLocked (1) :
La limite du nombre d'échec a été atteinte et le compte a été bloqué. L'utilisateur doit retenter plus tard ou contacter un administrateur.
compareResponse.resultCode = compareTrue (6), passwordPolicyResponse.warning = graceAuthNsRemaining :
Le mot de passe a expiré mais il reste des authentifications de grâce. L'utilisateur doit le changer
compareResponse.resultCode = compareFalse (5), passwordPolicyResponse.error = passwordExpired (0) :
Le mot de passe a expiré et il ne reste plus d'authentification de grâce. L'utilisateur doit contacter un administrateur
compareResponse.resultCode = compareTrue (6), passwordPolicyResponse.warning = timeBeforeExpiration :
Le mot de passe de l'utilisateur va expirer dans n secondes

Autres opérations

Pour les opérations autres que bind, unbind, abandon ou StartTLS, le client vérifie le resultCode et de contrôle pour déterminer si l'utilisateur doit changer le mot de passe immédiatement.

<Response>.resultCode = insufficientAccessRights (50), passwordPolicyResponse.error = changeAfterReset (2) :
L'utilisateur doit changer son mot de passe immédiatement

Administation des stratégies de mot de passe

Une stratégie de mot de passe est définie pour une sous-arborescence du DIT en ajoutant une subentry dont le supérieur immédiat est la racine de la sous-arborescence, l'objectclass auxiliaire pwdPolicy. Le scope de la stratégie est définie par l'attribut SubtreeSpecification.

Il est possible de définir des stratégies de mot de passe pour différents attributs de mot de passe dans la même entrée pwdPolicy, en spécifiant plusieurs valeurs de pwdAttribute. Les stratégies de mot de passe peuvent également être dans les subentry séparés.

Seul une stratégie ne peut être effectif pour un attribut de mot de passe dans une entrée. Si plusieurs stratégies existent, qui se chevauchent dans un scope, le résultat est indéfinis.

Il devrait être possible d'écraser une stratégie de mot de passe pour un utilisateur en définissant une nouvelle stratégie dans une subentry de l'entrée de l'utilisateur.

Chaque objet qui est contrôlé par une stratégie de mot de passe annonce le subentry à utiliser pour contrôler sa stratégie dans son attribut pwdPolicySubentry. Les clients qui souhaitent examiner ou gérer une stratégie de mot de passe peuvent interroger le pwdPolicySubentry pour cet objet pour connaître le subentry pwdPolicy.

Stratégie de mot de passe et la réplication

l'objet pwdPolicy définis la stratégie de mot de passe pour une portion du DIT et doit être répliqué dans tous les répliquas de cette sous-arborescence. Les éléments de la stratégie de mot de passe qui sont liés aux utilisateurs sont stockés dans les entrées elles-même en tant qu'attributs opérationnels. Vu que ces attributs sont sujets à modification même sur un répliqua lecture seule, leur réplication doit être considéré avec attention.

l'attribut **pwdChangedTime** doit être répliqué sur tous les répliquas, pour permettre l'expiration du mot de passe. L'attribut **pwdReset** doit être répliqué sur tous les répliquas, pour refuser l'accès aux opérations autre que bind et modify. L'attribut **pwdHistory** doit être

répliqué auprès des répliquas en écriture. Il n'a pas à être répliqué sur un répliqua lecture seul vu que le mot de passe ne sera jamais modifié sur ce serveur.

Les attribut **pwdAccountLockedTime**, **pwdFailureTime** et **pwdGraceUseTime** devraient être répliqués sur les répliquas en écriture, rendant la stratégie de mot de passe global à tous les serveurs. Quand l'entrée utilisateur est répliquée sur un répliqua lecture-seul, ces attributs ne devraient pas être répliqués. Celà signifie que le nombre d'échec, d'authentification de grâce et de blocage sont dans chaque serveur répliqué. Par exemple, le nombre effectif de tentatives échoués sur un mot de passe utilisateur sera N x M (où N est le nombre de serveurs et M la valeur de pwdMaxFailure). Répliquer ces attributs à un répliqua lecture seule peut réduire le nombre de tentatives global mais peut aussi introduire certaines inconsistances dans la manière dans la stratégie est appliquée.

Note : Il y a certaines situations où la réplication globale de ces attributs d'état peuvent être non-désirable. Par exemple, si 2 clusters de répliquas sont distants géographiquement et joins par un lien lent, et leur utilisateurs se loggent seulement sur un des 2 emplacement, il peut être préférable de ne pas propager tous les changements d'état entre les clusters. Les serveurs devraient permettre aux administrateurs de contrôler quels attributs sont répliqués au cas-par-cas.

Les serveurs participant à une réplication multi-master devaient employer un mécanisme qui s'assure de l'unicité des valeurs en populant les attributs pwdFailureTime et pwdGraceUseTime. Pour faire ça, c'est un problème locat et peut consister en l'utilisation d'un simple source autoritative pour la génération de valeurs de temps unique, ou peut consister de l'utilisation de la partie fractionnelle des secondes pour maintenir un identifiant de répliqua.

Considérations de sécurité

Ce document définis un jeu de règles à implémenter dans un serveur LDAP. Pour mitiger certains risques de sécurité associés avec l'utilisation de mots de passe et augmenter la difficulté de crackage de mot de passe.

- L'authentification avec un mot de passe doit suivre les recommandations de la rfc4513
- Les modifications de mot de passe devraient être se produire seulement quand la connexion est protégées pour la confidentialité et une authentification sécurisée.
- Les contrôles d'accès devraient être utilisés pour restreindre l'accès aux attribut de stratégie de mot de passe. Les attributs définis pour maintenir les information de stratégie de mot de passe devaient seulement être modifiable pas l'administrateur de mot de passe ou par une haute autorité.
- Vu qu'il est possible de définir une stratégie de mot de passe pour un utilisateur spécifique en ajoutant un subentry immédiatement sous l'entrée de l'utilisateur, les contrôles d'accès devraient être utilisés pour restreindre l'utilisation de l'objet pwdPolicy ou du subentry.
- Quand une stratégie de mot de passe à détection d'intrus est appliquée, l'annuaire LDAP est sujet aux attaques DOS. Un utilisateur malicieux peut délibérément bloquer le compte d'un utilisateur (ou tous) en envoyant des bind avec de faux mots de passe. Il n'y a aucun mécanisme pour parer ces attaques. Le serveur LDAP devraie logger autant d'informations qu'il peut (tel que les adresses IP) quand un compte est bloqué, pour être capable d'identifier l'origine de l'attaque. Refuser les accès anonymes à l'annuaire permet de réduire ce genre d'attaque. Utiliser un délai d'authentification à la place aide à éviter ces attaques DOS.
- En retournant certains codes de status (tel que passwordPolicyResponse.error=accountLocked) permet à un attaquant DOS de savoir qu'il a réussit à bloquer le compte. Les serveurs devraient implémenter des vérifications additionnelles qui retournent le même statut quand il est détecté qu'un certain nombre de demandes d'authentification échoué s'est produit sur une simple connexion, ou depuis une adresse client.