Passwortsicherheit mit dem Overlay ppolicy und dem ppm im OpenLDAP

Autor:
Stefan Kania

Ort:
Berlin

06.05.2025

1 Einführung

Sichere Passwörter? Was sind sicherer Passwörter und wie kann ich meine Anwender dazu bringen, sichere Passwörter zu verwenden? Das BSI hat da auch eine Meinung zu. Hier werden zwei verschiedene Möglichkeiten für ein sicheres Passwort aufgeführt:

1. Wenig komplexe Passwörter, zum Beispiel, nur Groß- und Kleinbuchstaben. Dann sollte das Passwort aber mindestens eine Länge von 25 Zeichen haben.
2. Oder aber ein komplexes Passwort aus Groß- und Kleinbuchstaben und Zahlen und Sonderzeichen. Dann würde, nach Aussage des BSI auch ein 8 Zeichen langes Passwort genügen.

Aber jeder weiß heute, ein Passwort kann noch so komplex oder noch so lang sein, wenn der Benutzer es auf einen Zettel schreibt und unter die Tastatur legt, ist alles umsonst gewesen. Heute sollte die zwei Faktor Authentifizierung immer das Maß aller Dinge sein. Aber trotzdem macht es Sinn, sich einmal Gedanken über die Passwortsicherheit zu machen und zu schauen, wie die unterschiedlichsten System für sichere Passwörter sorgen. Ich werde da mal eine Blick auf die Möglichkeiten von OpenLDAP werfen.

2 Möglichkeiten mit OpenLDAP

Welche Möglichkeiten haben Sie jetzt, um im OpenLDAP für sicherer Passwörter zu sorgen:

- Der erste Schritt ist immer, welcher Passworthash soll verwendet werden? Seit der OpenLDAP-Version 2.5 gibt es dafür nur eine Antwort: *Argon2*. Denn Argon2 verhindert, dass Passwörter mittels schneller GPUs parallel ausprobiert werden können. Den GPUs besitzen meist nicht sehr viel Arbeitsspeicher und das macht sich Argon2 zu nutze. Für die Entschlüsselung eines Passworts, das mit Argon2 verschlüsselt wurde, wird immer eine vordefinierte Menge an Arbeitsspeicher für die Vektoren zur Entschlüsselung benötigt. Je größer die Vektoren, also die benötigte Menge an Arbeitsspeicher ist, um so länger dauert eine Entschlüsselung. Zusätzlich können noch die Anzahl der Iterationen angegeben werden um die Vektoren zu berechnen. Auch hier gilt: Bei steigender Anzahl an Iterationen wir mehr Zeit für die Entschlüsselung benötigt. ABER: Bei zu viel des Guten kommt irgendwann der Punkt, dass die Anmeldung der Benutzer zu lange dauert, denn auch der Anmeldevorgang verlängert sich, wenn das Passwort überprüft wird. Testen Sie, wie schnell die Passwörter bei der Eingabe geprüft werden können und ob die dafür benötigte Zeit für die Anwender noch erträglich ist. Ein guter Wert ist 500ms, länger sollte eine Anmeldung nicht dauern.
- Für die Regeln der Verschlüsselung kann, auch bei OpenLDAP 2.5 und 2.6, das Overlay *ppolicy* genutzt werden. Mit diesem Overlay können Sie verschiedene Passwort-Policies einrichten und dann unterschiedlichen Benutzer zuweisen. Es gibt immer eine *default policy*, die immer dann gilt, wenn ein Benutzer keine Policy explizit zugewiesen bekommen hat. So sind Sie in der Lagen, einen gewissen Grundschutz einzurichten, aber für bestimmte Mitarbeiter strengere Policies zu generieren und zuzuweisen.
- Eine weitere, neu Möglichkeit mit OpenLDAP 2.5, ist die Verwendung des *password policiy module (ppm)*. Das *ppm* wird ebenfalls über das Overlay *ppolicy* konfiguriert, ersetzt aber dann die Komplexitätsregel der einfachen Policies komplett. Die Regeln für die Passwörter können Sie hier sehr komplex einrichten. Das *ppm* wird immer zusätzlich zu einer einfachen Policy konfiguriert.

3 Einrichtung des Overlays ppolicy

Das Overlay *ppolicy* verwaltet lediglich die Vergabe von Passwörtern, kann aber alleine nicht feststellen, wann ein Benutzer sich das letzte (oder erste) Mal angemeldet hat. Um später, auf

Grund dieser Eigenschaften, ein Konto zu sperren oder den Benutzer zu zwingen sein Passwort bei der nächsten Anmeldung zu ändern, wird noch das Overlay *lastbind* benötigt. Für beide Overlay ist es notwendig, vor der Einrichtung das entsprechenden Modul zu laden. Im Gegensatz zu den OpenLDAP-Version vor 2.5 wird kein zusätzliches Schema mehr benötigt. Das Schema *ppolicy* ist grundsätzlich immer im OpenLDAP ab der Version 2.5 vorhanden. Listing 3.1 zeigt die LDIF-Datei für das Laden der beiden Module:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: ppolicy.la
-
add: olcModuleLoad
olcModuleLoad: lastbind.la
```

Listing 3.1: Laden der Module ppolicy und lastbind

Im nächsten Schritt werden dann die beiden benötigten Overlays geladen. Das Listing 3.2 zeigt die LDIF-Datei für das Hinzufügen der beiden Overlays zur Objektdatenbank.

Hinweis!

Das Overlay *ppolicy* sollte möglichst immer das erste Overlay in einer Datenbank sein, aus diesem Grund wird hier explizit die Nummer *0* für das Overlay vergeben.

Listing 3.2: Einspielen der Overlays ppolicy und lastbind

```
dn: olcOverlay={0}ppolicy,olcDatabase={2}mdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcPPolicyConfig
olcOverlay: ppolicy
olcPPolicyDefault: cn=default,ou=policies,dc=example,dc=net
olcPPolicyForwardUpdates: FALSE
olcPPolicyHashCleartext: TRUE
olcPPolicyUseLockout: FALSE

dn: olcOverlay=lastbind,olcDatabase={2}mdb,cn=config
objectClass: olcLastBindConfig
objectClass: olcOverlayConfig
olcOverlay: lastbind
# Only set the following on consumers
# olcLastBindForwardUpdates: TRUE
```

Hinweis!

Achten Sie bei der Übernahme der LDIF-Dateien auf die korrekte Nummer Ihrer Objektdatenbank

Hinweis!

Das Overlay wird auf eventuell vorhandenen Consumern nicht benötigt. Lediglich das Modul *ppolicy.la* muss auf den Consumern geladen sein.

Das Attribut *olcPPolicyForwardUpdates*, im Overlay *ppolicy*, wird nur benötigt, wenn auch Consumer zum Einsatz kommen. Dort setzen Sie dann der Wert des Attributs auf *TRUE*. Das Gleiche gilt auch auch für das Attribute *olcLastBindForwardUpdates* im Overlay *lastbind*.

3.1 Einrichten der Policies

Bei den Policies gibt es immer eine *default policy*, die für alle Benutzer der Datenbank gilt. Zusätzlich können Sie weitere Policies erstellen, die Sie dann einzelnen Benutzern zuweisen können. Die *default policy* wird bereits beim Anlegen des Overlays benannt. Fehlt nur noch die LDIF-Datei aus Listing 3.1.1 für die Einstellungen der Policy:

```
dn: ou=policies,dc=example,dc=net
changetype: add
objectClass: organizationalUnit
ou: policies

dn: cn=default,ou=policies,dc=example,dc=net
changetype: add
objectClass: pwdPolicy
objectClass: person
cn: default
pwdAttribute: userPassword
pwdAllowUserChange: TRUE
pwdExpireWarning: 1440
pwdGraceAuthnLimit: 5
pwdInHistory: 5
pwdLockout: TRUE
pwdFailureCountInterval: 300
pwdLockoutDuration: 3600
pwdMaxFailure: 5
pwdMaxAge: 2678400
pwdMinAge: 86400
pwdMinLength: 8
pwdCheckQuality: 2
sn: OurDefaultPolicy
```

Listing 3.1.1: LDIF für die default Policy

Es ist sinnvoll, alle Policies immer in einem eigenen Container abzulegen, so behalten Sie auch bei vielen Policies immer den Überblick. Aus dem Grund wird in dem Listing auch gleich ein Container erzeugt.

Für jede weitere Password Policy können Sie jetzt eine eigene LDIF-Datei erzeugen. Listing 3.1.2 zeigt eine erweiterte Policy:

```
dn: cn=special,ou=policies,dc=example,dc=net
changetype: add
cn: special
objectClass: pwdPolicy
objectClass: person
pwdAttribute: userPassword
pwdAllowUserChange: TRUE
pwdExpireWarning: 1440
pwdGraceAuthnLimit: 5
pwdInHistory: 10
pwdLockout: TRUE
pwdFailureCountInterval: 300
pwdLockoutDuration: 3600
pwdMaxFailure: 5
pwdMaxAge: 2678400
pwdMinAge: 86400
pwdMinLength: 16
sn: OurSpecialPolicy
```

Listing 3.1.2: Alternative Policy

Jetzt können Sie einem, oder mehreren Benutzern die neue Policy zuweisen. Die Policy wird über das Attribute *pwdPolicySubentry* bei den Benutzern verwaltet. Bei dem Attribut handelt es sich um eine internes Attribut, das Sie, mit einem einfachen `ldapsearch` nicht angezeigt bekommen. Listing 3.1.3 zeigt die LDIF-Datei für die Zuweisung der neuen Policy an einen Benutzer:

```
dn: cn=prod-al,ou=users,ou=produktion,dc=example,dc=net
changetype: modify
add: pwdPolicySubentry
pwdPolicySubentry: cn=special,ou=policies,dc=example,dc=net
```

Listing 3.1.3: Zuweisung der neuen Passwort Policy

Das Listing 3.1.4 zeigt die Internen Attribute des gerade angepassten Benutzer:

```
root@ldap01:~# ldapsearch -QY external -H ldapi:/// -LLL cn=prod-al +
dn: cn=prod-al,ou=users,ou=Produktion,dc=example,dc=net
structuralObjectClass: inetOrgPerson
entryUUID: e18e459a-922e-103f-82d9-35adf92732b2
creatorsName: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
createTimestamp: 20250310190905Z
pwdPolicySubentry: cn=special,ou=policies,dc=example,dc=net
entryCSN: 20250312084049.223081Z#000000#000#000000
modifiersName: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
modifyTimestamp: 20250312084049Z
entryDN: cn=prod-al,ou=users,ou=Produktion,dc=example,dc=net
subschemaSubentry: cn=Subschema
hasSubordinates: TRUE
```

Listing 3.1.4: Interne Attribute des geänderten Benutzer

Jetzt können Sie die Passwort Policies testen, in dem Sie für die Benutzer die Passwörter ändern.

3.2 PAM

Hier werden Sie aber feststellen, dass die lokalen PAM-Richtlinien (getestet bei Debian 12) Probleme bereiten. Legen Sie zu Beispiel eine Richtlinie mit einer Mindestlänge von 5 Zeichen fest, wird das System immer eine Meldung wie in Listing 3.2.1 ausgeben:

```
u1-prod@ldap01:~$ passwd
Aktuelles Passwort:
Geben Sie ein neues Passwort ein:
Unsicheres Passwort: Das Passwort ist kürzer als 8 Zeichen
Geben Sie ein neues Passwort ein:
```

Listing 3.2.1: PAM-Meldung für kurzes Passwort

Auch andere Werte die von PAM voreingestellt sind, würden die Passwordpolicy vom OpenLDAP beeinflussen. Der Grund dafür ist in der Datei /etc/pam.d/common-password zu sehen. In Listing 3.2.2 sehen Sie die Standardwerte in der Datei:

```
password        requisite                       pam_pwquality.so retry=3
password        [success=2 default=ignore]      pam_unix.so obscure use_authtok \
                                                try_first_pass yescrypt
password        sufficient                      pam_sss.so use_authtok
password        requisite                       pam_deny.so
password        required                        pam_permit.so
```

Listing 3.2.2: Standarwerte der Datei common-password

Dort wird das Modul pam_pwquality.so verwendet. Das Modul sollte auch immer genutzt werden, denn das regelt dann die Passwortrichtlinien für lokale Benutzer. Sie können die Einstellungen aber anpassen. Dafür haben Sie zwei Möglichkeiten:

1. Passen Sie die Datei direkt an, in dem Sie die Zeile für das Modul um den Wert *minlen=5* erweitern.
2. Oder Sie stellen die Werte in der Konfigurationsdatei, /etc/security/pwquality.conf, für das Modul ein. Dort haben Sie auch noch weitere Möglichkeiten die Passwortrichtlinien der lokalen Benutzer anzupassen.

Bei der Standardeinstellung der Datei common-password wird immer als Erstes die lokale Passwortänderung versucht und dann erst wird der aktuelle Benutzer über den *sssd* im LDAP gesucht. Ich habe deshalb die Reihenfolge wie in Listing 3.2.3 geändert:

```
password        requisite                       \
                pam_pwquality.so retry=3 minlen=5
password        [success=2 default=ignore]      \
                pam_sss.so use_authtok
password        [success=1 default=ignore]      \
                pam_unix.so obscure use_authtok try_first_pass yescrypt
password        requisite                       pam_deny.so
password        required                        pam_permit.so
```

Listing 3.2.3: Änderungen der Datei common-password

Wenn jetzt ein Benutzer sein LDAP-Passwort ändert, sehen Sie im Log die Zeilen aus Listing 3.2.4

```
conn=1022 op=3 EXT oid=1.3.6.1.4.1.4203.1.11.1
conn=1022 op=3 PASSMOD id="cn=u2-Prod,ou=users,ou=Produktion,\
                           dc=example,dc=net" old new
```

Listing 3.2.4: Log-Einträge bei der Passwortänderung

4 Komplexen Passwortrichtlinien

Mit OpenLDAP 2.5 sind zu den vorher besprochenen Passwort Policies noch die komplexen Policies hinzu gekommen. Um diese komplexen Policies nutzen zu können, wir das Zusatzmodul *Password Policy Module (ppm)* benötigt.

Damit Sie das *ppm* nutzen können, erweitern Sie das Overlay in der Objektdatenbank um das Attribut *olcPpolicyCheckModule*. Das Attribut verweist auf das Modul unter /opt/symas/lib/openldap/ppm.so. In Listing 4.1 sehen Sie die LDIF-Datei, die zur Erweiterung des Overlays genutzt werden kann:

```
dn: olcOverlay={0}ppolicy,olcDatabase={2}mdb,cn=config
changetype: modify
add: olcPpolicyCheckModule
olcPpolicyCheckModule: /opt/symas/lib/openldap/ppm.so
```

Listing 4.1: LDIF zur Erweiterung des ppolicy Overlay

Jetzt wird eine ASCCII-Textdatei mit den gewünschten Einstellungen benötigt. Diese Textdatei muss dann *base64* codiert werden, bevor Sie die Einstellungen in den LDAP einspielen können. Listing 4.2 zeigt die umzuwandelnden Werte:

```
minQuality 8
checkRDN 0
checkAttributes mail
forbiddenChars
maxConsecutivePerClass 3
useCracklib 1
cracklibDict /var/cache/cracklib/cracklib_dict
class-upperCase ABCDEFGHIJKLMNOPQRSTUVWXYZ 2 1 3
class-lowerCase abcdefghijklmnopqrstuvwxyz 2 1 3
class-digit 0123456789 1 1 0
class-special @=?,.;:-+*/ 1 5 3
class-umlaut äöüÄÖÜß 0 1 0
pwdMinLength: 10
```

Listing 4.2: Textdatei mit den Werten für das ppm

Mit dem Kommando `base64 <dateiname>` können Sie den Inhalt der Datei umwandeln. Listing 4.3 zeigt das Ergebnis:

```
root@ldap01:~/slac/einrichtung# base64 08-ppm-mypolicy.txt
bWluUXVhbGl0eSA1CmNoZWNrUkROIDAKY2hlY2tBdHRyaWJ1dGVzIG1haWwKZm9yYmlkZGVuQ2hh
cnMKbWF4Q29uc2VjdXRpdmVQZXJDbGFzcyA1CnVzZUNyYWNrbGliIDEKY3JhY2tsaWJEaWN0IC92
YXIvY2FjaGUvY3JhY2tsaWIvY3JhY2tsaWJfZGljdApjbGFzcy11cHBlckNhc2UgQUJDREVGR0hJ
SktMTU5PUFFSU1RVVldYWVogMiAyIDUKY2xhc3MtbG93ZXJDYXNlIGFiY2RlZmdoaWprbG1ub3Bx
cnN0dXZ3eHl6IDIgMSA1CmNsYXNzLWRpZ2l0IDAxMjM0NTY3ODkgMSAzIDAKY2xhc3Mtc3BlY2lh
bCBAPT8sLjs6LSsqLyAxIDUgMApjbGFzcy11bWxhdXQgw6TDtsO8w4TDlsOcw58gMCAzIDAKcHdk
TWluTGVuZ3RoOiAxMAoK
```

Listing 4.3: Umwandeln der ASCCI-Datei

Bevor die komplexe Policy eingespielt werden kann, ist es notwendig, dass Sie die *default policy* noch um eine Objektklasse und ein dazugehöriges Attribut erweitern. Listing 4.4 zeigt die LDIF-Datei mit den Erweiterungen:

```
dn: cn=default,ou=policies,dc=example,dc=net
changetype: modify
add: objectClass
objectClass: pwdPolicyChecker
-
add: pwdUseCheckModule
pwdUseCheckModule: TRUE
```

Listing 4.4: Zusätzliche Objektklasse für die default Policy

Erst jetzt kann die *default policy* um das *ppm* erweitert werden. Dazu wird die umgewandelte Textdatei in eine LDIF-Datei mit dem entsprechenden Attribut *pwdCheckModuleArg* eingetragen. Listing 4.5 zeigt die LDIF-Datei:

```
dn: cn=default,ou=policies,dc=example,dc=net
changetype: modify
replace: pwdCheckModuleArg
pwdCheckModuleArg:: bWluUXVhbGl0eSA1CmNoZWNrUkROIDAKY2hlY2tBdHRyaWJ1dGVzIG1\
                    haWwKZm9yYmlkZGVuQ2hh
cnMKbWF4Q29uc2VjdXRpdmVQZXJDbGFzcyA1CnVzZUNyYWNrbGliIDEKY3JhY2tsaWJEaWN0IC92
YXIvY2FjaGUvY3JhY2tsaWIvY3JhY2tsaWJfZGljdApjbGFzcy11cHBlckNhc2UgQUJDREVGR0hJ
SktMTU5PUFFSU1RVVldYWVogMiAyIDUKY2xhc3MtbG93ZXJDYXNlIGFiY2RlZmdoaWprbG1ub3Bx
cnN0dXZ3eHl6IDIgMSA1CmNsYXNzLWRpZ2l0IDAxMjM0NTY3ODkgMSAzIDAKY2xhc3Mtc3BlY2lh
bCBAPT8sLjs6LSsqLyAxIDUgMApjbGFzcy11bWxhdXQgw6TDtsO8w4TDlsOcw58gMCAzIDAKcHdk
TWluTGVuZ3RoOiAxMAoK
```

Listing 4.5: LDIF-Datei für das Checkmodul

Hinweis!

Achten Sie bei der Erstellung der LDIF-Datei darauf, dass hinter dem Attributnamen ein doppelter Doppelpunkt steht, der den nachfolgenden Wert als base64 codiert markiert. Wenn Sie, so wir hier im Beispiel, mit Zeilenumbrüchen arbeiten, muss am Anfang der Zeile genau ein Leerzeichen stehen, sonst kann da Attribut nicht korrekt ausgewertet werden.

Wenn Sie den LAM-Pro nutzen, können Sie die Einstellungen auch direkt dort in der Policy ändern. Der LAM-Pro übersetzt die base64 Einträge in ASCII. Bild 4.1 zeigt den Eintrag des Attributs in der *default policy*.

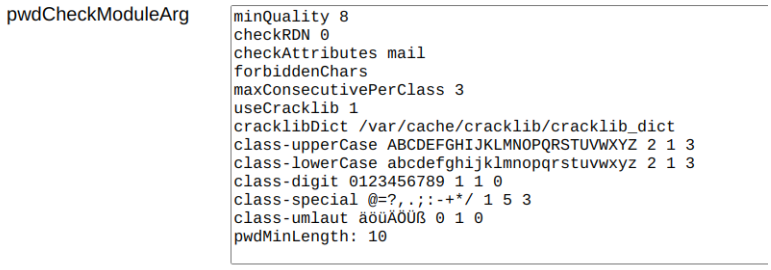

Abbildung 4.1: Die ppm-Setings im LAM

Hinweis!

Die Regeln können durch die Einstellung so komplex werden, dass es nahezu unmöglich ist, ein gültiges Passwort einzugeben. Testen Sie daher die Einstellungen bevor Sie diese produktiv nutzen.

Da die komplexen Regeln immer bei einer Policy eingetragen werden, können Sie auch hier für jede Policy eine eigene komplexe Regel erstellen.

Wichtig!

Nach jeder Änderung der *ppm* müssen Sie auf jeden Fall den *slapd* neu starten.

Sobald Sie das *ppm* nutzen, werden die in der Policy eingerichteten Komplexitätswerte ignoriert. Sorgen Sie dafür, dass jetzt alle gewünschten Einstellungen über das *ppm* realisiert werden. Alle möglichen Parameter des *ppm* finden Sie in der Manpage *slapm-ppm(5)*. In Listing 4.6 sehne Sie die Meldungen, die im Log erscheinen, wenn die Änderung eines Passworts fehlschlägt:

```
conn=1064 op=3 PASSMOD id="cn=u1-Verw,ou=users,ou=Verwaltung,\
                           dc=example,dc=net" old new
ppm: entry cn=u1-verw,ou=users,ou=verwaltung,dc=example,dc=net
ppm: Reading pwdCheckModuleArg attribute
ppm: RAW configuration: minQuality 5
         checkRDN 1
         checkAttributes mail
         forbiddenChars
         maxConsecutivePerClass 5
         useCracklib 1
         cracklibDict /var/cache/cracklib/cracklib_dict
         class-upperCase ABCDEFGHIJKLMNOPQRSTUVWXYZ 2 2 5
         class-lowerCase abcdefghijklmnopqrstuvwxyz 2 1 5
         class-digit 0123456789 1 3 0
         class-special @=?,.;:-+*/ 1 5 0
         class-umlaut äöüÄÖÜß 0 3 0
         pwdMinLength: 10
ppm: Parsing pwdCheckModuleArg attribute
ppm: get line: minQuality 5
ppm: Param = minQuality, value = 5, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: 5
ppm: get line: checkRDN 0
ppm: Param = checkRDN, value = 0, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: 0
ppm: get line: checkAttributes mail
ppm: Param = checkAttributes, value = mail, min = (null), \
           minForPoint = (null), max = (null)
ppm:  Accepted replaced value: mail
ppm: get line: forbiddenChars
ppm: No value, goto next parameter
ppm: get line: maxConsecutivePerClass 5
ppm: Param = maxConsecutivePerClass, value = 5, min \
           = (null), minForPoint = (null), max = (null)
ppm:  Accepted replaced value: 5
ppm: get line: useCracklib 1
ppm: Param = useCracklib, value = 1, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: 1
ppm: get line: cracklibDict /var/cache/cracklib/cracklib_dict
ppm: Param = cracklibDict, value = /var/cache/cracklib/cracklib_dict,\
           min = (null), minForPoint = (null), max = (null)
ppm:  Accepted replaced value: /var/cache/cracklib/cracklib_dict
ppm: get line: class-upperCase ABCDEFGHIJKLMNOPQRSTUVWXYZ 2 2 5
ppm: Param = class-upperCase, value = ABCDEFGHIJKLMNOPQRSTUVWXYZ, \
           min = 2, minForPoint = 2, max = 5
ppm:  Accepted replaced value: ABCDEFGHIJKLMNOPQRSTUVWXYZ
ppm: get line: class-lowerCase abcdefghijklmnopqrstuvwxyz 2 1 5
```

```
ppm: Param = class-lowerCase, value = abcdefghijklmnopqrstuvwxyz, \
           min = 2, minForPoint = 1, max = 5
ppm:  Accepted replaced value: abcdefghijklmnopqrstuvwxyz
ppm: get line: class-digit 0123456789 1 3 0
ppm: Param = class-digit, value = 0123456789, min = 1, minForPoint \
           = 3, max = 0
ppm:  Accepted replaced value: 0123456789
ppm: get line: class-special @=?,.;:-+*/ 1 5 0
ppm: Param = class-special, value = @=?,.;:-+*/, min = 1, minForPoint \
           = 5, max = 0
ppm:  Accepted replaced value: @=?,.;:-+*/
ppm: get line: class-umlaut äöüÄÖÜß 0 3 0
ppm: Param = class-umlaut, value = äöüÄÖÜß, min = 0, minForPoint \
           = 3, max = 0
ppm:  Accepted new value:
ppm: get line: pwdMinLength: 10
ppm: Parameter 'pwdMinLength:' rejected
ppm: 1 point granted for class class-lowerCase
check_password_quality: module error: (/opt/symas/lib/openldap/ppm.so)\
           Password for dn="cn=u1-verw,ou=users,ou=verwaltung,dc=example,\
           dc=net" does not pass required number of strength checks \
           (1 of 5).[1]
conn=1064 op=4 UNBIND
[77B blob data]
conn=1064 fd=17 closed
```

Listing 4.6: Fehlerhafte Passwortänderung

Das von mir an dieser Stelle verwendete Passwort *üPoI:12ZQ;*#* erfüllt leider nicht die geforderten Bedingungen. Es ist, bei dieser strengen Regle kaum möglich ein gültiges Passwort zu vergeben. Aus diesem Grund änder ich die Regel wie in Listing 4.7 ab:

```
minQuality 4
checkRDN 1
checkAttributes mail
forbiddenChars /$
maxConsecutivePerClass 5
useCracklib 1
cracklibDict /var/cache/cracklib/cracklib_dict
class-upperCase ABCDEFGHIJKLMNOPQRSTUVWXYZ 0 5 0
class-lowerCase abcdefghijklmnopqrstuvwxyz 0 12 0
class-digit 0123456789 0 1 0
class-special @=?,.;:-+*(){}[]% 0 1 0
class-umlaut äöüÄÖÜß 0 3 0
```

Listing 4.7: Neue Regel für das ppm

Jetzt folgt eine erneuter Versuch das Passwort zu ändern und zwar auf den Wert *ThereIsNoCow-Level1234!"+* Das Ergebnis sehen Sie in Listing 4.8:

```
conn=1011 op=2 RESULT tag=97 err=0 qtime=0.000005 etime=0.020480 text=
conn=1011 op=3 EXT oid=1.3.6.1.4.1.4203.1.11.1
conn=1011 op=3 PASSMOD id="cn=prod-al,ou=users,ou=Produktion,\
                           dc=example,dc=net" old new
ppm: entry cn=prod-al,ou=users,ou=produktion,dc=example,dc=net
ppm: Reading pwdCheckModuleArg attribute
ppm: RAW configuration: minQuality 4
     checkRDN 1
     checkAttributes mail
     forbiddenChars /$
     maxConsecutivePerClass 5
     useCracklib 1
     cracklibDict /var/cache/cracklib/cracklib_dict
     class-upperCase ABCDEFGHIJKLMNOPQRSTUVWXYZ 0 5 0
     class-lowerCase abcdefghijklmnopqrstuvwxyz 0 12 0
```

```
     class-digit 0123456789 0 1 0
     class-special @=?,.;:-+*(){}[]% 0 1 0
     class-umlaut äöüÄÖÜß 0 3 0
ppm: Parsing pwdCheckModuleArg attribute
ppm: get line: minQuality 4
ppm: Param = minQuality, value = 4, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: 4
ppm: get line: checkRDN 1
ppm: Param = checkRDN, value = 1, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: 1
ppm: get line: checkAttributes mail
ppm: Param = checkAttributes, value = mail, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: mail
ppm: get line: forbiddenChars /$
ppm: Param = forbiddenChars, value = /$, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: /$
ppm: get line: maxConsecutivePerClass 5
ppm: Param = maxConsecutivePerClass, value = 5, min = (null), \
           minForPoint = (null), max = (null)
ppm:  Accepted replaced value: 5
ppm: get line: useCracklib 1
ppm: Param = useCracklib, value = 1, min = (null), minForPoint \
           = (null), max = (null)
ppm:  Accepted replaced value: 1
ppm: get line: cracklibDict /var/cache/cracklib/cracklib_dict
ppm: Param = cracklibDict, value = /var/cache/cracklib/cracklib_dict,\
           min = (null), minForPoint = (null), max = (null)
ppm:  Accepted replaced value: /var/cache/cracklib/cracklib_dict
ppm: get line: class-upperCase ABCDEFGHIJKLMNOPQRSTUVWXYZ 0 5 0
ppm: Param = class-upperCase, value = ABCDEFGHIJKLMNOPQRSTUVWXYZ, min \
           = 0, minForPoint = 5, max = 0
ppm:  Accepted replaced value: ABCDEFGHIJKLMNOPQRSTUVWXYZ
ppm: get line: class-lowerCase abcdefghijklmnopqrstuvwxyz 0 12 0
ppm: Param = class-lowerCase, value = abcdefghijklmnopqrstuvwxyz, \
           min = 0, minForPoint = 12, max = 0
ppm:  Accepted replaced value: abcdefghijklmnopqrstuvwxyz
ppm: get line: class-digit 0123456789 0 1 0
ppm: Param = class-digit, value = 0123456789, min = 0, \
           minForPoint = 1, max = 0
ppm:  Accepted replaced value: 0123456789
ppm: get line: class-special @=?,.;:-+*(){}[]% 0 1 0
ppm: Param = class-special, value = @=?,.;:-+*(){}[]%, min = 0, \
           minForPoint = 1, max = 0
ppm:  Accepted replaced value: @=?,.;:-+*(){}[]%
ppm: get line: class-umlaut äöüÄÖÜß 0 3 0
ppm: Param = class-umlaut, value = äöüÄÖÜß, min = 0, \
           minForPoint = 3, max = 0
ppm:  Accepted new value:
ppm: 1 point granted for class class-upperCase
ppm: 1 point granted for class class-lowerCase
ppm: 1 point granted for class class-digit
ppm: 1 point granted for class class-special
ppm: Checking if prod part of RDN matches the password
ppm: al part of RDN is too short to be checked
ppm: check password against mail attribute
ppm: check password against prod-al@example.net attribute value
ppm: Checking if prod part of value prod-al@example.net of \
     attribute mail matches the password
ppm: al part of value prod-al@example.net of attribute mail \
     is too short to be checked
```

```
ppm: Checking if example.net part of value prod-al@example.net \
     of attribute mail matches the password
conn=1011 op=4 UNBIND
conn=1011 op=3 RESULT oid= err=0 qtime=0.000162 etime=0.043199 text=
conn=1011 fd=18 closed
```

Listing 4.8: Erfolgreiche Änderung des Passworts

Erst wenn die Regeln der verwendeten Buchstaben, Zahlen und Sonderzeichen stimmen, werden die anderen Parameter, wie die eventuell vorhandene Mailadresse geprüft. Setzen Sie die Regeln nicht zu streng für die Buchstaben, Zahlen und Sonderzeichen. Prüfen Sie besser, ob ein neues Passwort teile von bestimmten Attributen des Benutzer besitzt und verwenden Sie die *cracklib* Prüfung auf eventuell verwendete Worte.

So können sie jetzt die Komplexität der Passwörter der Benutzer steuern.