How to reset an Active Directory password with PowerShell

https://4sysops.com/archives/how-to-reset-an-active-directory-password-with-powershell/

We all know the scenario—a user calls to complain that the computer does not accept the password and asks for a reset. Or you get an email from HR with several new hires that need new passwords generated. In both cases, this can be done using GUI tools (Active Directory Users and Computers and Active Directory Administrative Center). However, the GUI method still requires several steps and thus might not always be effective.

Luckily, the password reset can be done quickly with PowerShell, even for dozens of accounts. There are two ways to reset a user account password in PowerShell:

- The Set-ADAccountPassword cmdlet, included in the RSAT PowerShell module
- The Active Directory Service Interface (ADSI) method

Now, let's get down to business and have a look at both of them.

Prerequisites

To reset an AD account password with PowerShell, you need three things:

- A domain-joined Windows machine, where you will execute the task
- An administrative account with permissions to reset AD passwords
- A code editor of your choice, such as Windows PowerShell ISE

While these three things are enough to use the ADSI method, if you want to use the Set-ADAccountPassword cmdlet, you will also need:

- At least Windows PowerShell 5.1
- Remote Server Administration Tools (RSAT) installed, specifically Active Directory Domain Services and Lightweight Directory Services tools

The above-mentioned tools can be installed directly on your Windows 10 PC. Alternatively, you can connect to a dedicated management computer or server, as I did.

Using Set-ADAccountPassword

Single user account password reset

To reset an AD user password, you need to know two things—the user's identity and a password to assign. As the Set-ADAccountPassword cmdlet only accepts secure string representation of the password, you need to convert your password first.

```
$Pass = ConvertTo-SecureString "Password@123" -AsPlainText -Force
```

Now, you can reset the password by running the following command:

```
Set-ADAccountPassword -Identity worker03 -NewPassword $pass –Reset
Set-ADUser -Identity worker03 -ChangePasswordAtLogon $true
```

In my case, the user's identity is the account's SAM account name value—*worker03*. The cmdlet also accepts other values, such as the user's GUID or SID; however, it is very likely you will not use those. The Set-ADUser

command forces the user to change their password at the next logon, which is the best practice, so I always use it. As you can see in the image, the cmdlet has no output if successfully completed.

```
PS C:\Users\Administrator> $Pass = ConvertTo-SecureString "Password@123" -AsPlainText -Force

PS C:\Users\Administrator> Set-ADAccountPassword -Identity worker03 -NewPassword $pass -Reset
Set-ADUser -Identity worker03 -ChangePasswordAtLogon $true


PS C:\Users\Administrator> |
```

Set ADAccountPassword has no output if completed without errors

Multiple user account password reset

When you need to reset multiple user account passwords, you still need two things—a list of users and their password values. The list of users can be stored in a text file; in my case, those names are *worker01*, *worker02*, and *worker03*.

As for the password, you will most likely want to assign a different one to each user. While there are different methods to do so, I will use the following line of code to generate 15-character passwords:

```
$Password = -join ((33..126) | Get-Random -Count 15 | ForEach-Object { [char]$_ })
```

Now, you can use the code below to generate, reset, and force a password change for each user in your list.

```
# Load user list
$ListOfUsers = Get-Content C:\temp\list.txt
foreach ($user in $ListOfUsers) {
#Generate a 12-character random password
$Password = -join ((33..126) | Get-Random -Count 15 | ForEach-Object { [char]$_ })
#Convert the password to secure string
$Pass = ConvertTo-SecureString $Password -AsPlainText -Force
#Reset the account password
Set-ADAccountPassword $user -NewPassword $Pass -Reset
#Force user to change password at next logon
Set-ADUser -Identity $user -ChangePasswordAtLogon $true
#Display userid and password values
Write-Host $user, $Password
}
```

```
PS C:\Users\Administrator> # Load user list
 $ListOfUsers = Get-Content C:\temp\list.txt
 foreach ($user in $ListOfUsers) {
     #Generate a 12-character random password
     $Password = -join ((33..126) | Get-Random -Count 15 | ForEach-Object { [char]$_ })
     #Convert the password to secure string
     $Pass = ConvertTo-SecureString $Password -AsPlainText -Force
     #Reset the account password
     Set-ADAccountPassword $user -NewPassword $Pass -Reset
     #Force user to change password at next logon
     Set-ADUser -Identity $user -ChangePasswordAtLogon $true
     #Display userid and password values
     Write-Host $user, $Password
 }
worker01 8L|3P)r"<WFn;^:
worker02 !YWi9l/pa,<yB}n
worker03 'ej<zksm}|bN.hR

PS C:\Users\Administrator> |
```

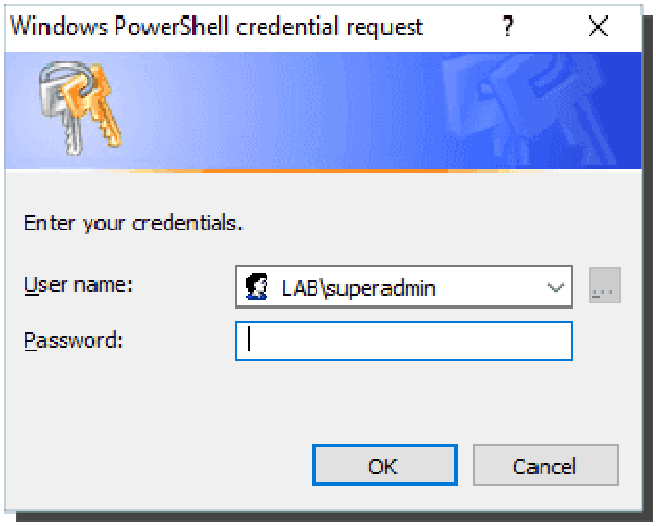
Multiple account password reset

Reset passwords using different administrative credentials

It is quite normal, especially in larger organizations, to have separate administrative accounts to perform privileged tasks, such as password resets. In such cases, running the above commands under your standard account PowerShell session would fail. Like many other cmdlets, Set-ADAccountPassword supports the *Credential* parameter, which allows it to perform the desired action under different user contexts.

```
Set-ADAccountPassword -Identity worker03 -NewPassword $Pass -Credential (Get-Credential –Credential “LAB\superadmin”)
```

The above line of code will cause a credential window to pop up, where you can provide your privileged account password.



Set ADAccountPassword allows the use of alternative credentials

ADSI method

The second method to reset account passwords in PowerShell is the ADSI method. The code to change the worker03 account password is as follows:

```
$userid = [ADSI]"LDAP://CN=worker03,OU=Usr,DC=LAB,DC=Local"
$userid.psbase.invoke("SetPassword",'Password@123')
$userid.psbase.CommitChanges()
```

While the ADSI method can be used on any Windows or PowerShell version, it has more disadvantages. First, it does not support the use of different credentials. That means you would need to run the PowerShell session under an account with the proper privileges to reset passwords. Second, it only supports the user's distinguished name (*CN=worker03,OU=Usr,DC=LAB,DC=Local)*. This limits you to targeting only users in a specific organizational unit (OU), so you would need to know in advance where the user account is located and change the code appropriately.

Obviously, this is not a pure PowerShell way to accomplish the task, and its limitations make it pretty much unusable with modern scripting techniques.