SysAdmin Magazine

netwrix

May the PowerShell Be With You

SysAdmin Magazine

Contents

Discover New Users in Active Directory **and Email Their Credentials**

by Karim Buzdar

IT Expert

When new employees join a company, IT technicians have to create their accounts in Active Directory. Later, the IT specialist welcomes each newcomer and helps them log in to the domain. In this article, I'll show you how to automate this procedure with the help of PowerShell scripting. Feel free to edit this script to suit your particular needs.

This blog post specifically covers the following three subjects:

Read the email password as a secure string, convert it to an encrypted string, and save it in a text file so normal users cannot read it. Later, the script reads it and reverses it to a secure string object to be used as a credential in subsequent email message cmdlets.Create a script to identify any new users added in AD within the last 24 hours, and send them a welcome email using Gmail's SMTP server.Schedule the script to run daily at 12:00 a.m. in Task Scheduler with the help of PowerShell.

I have used the following cmdlets in this post; details of each cmdlet is available on Technet's website.

1. Read-Host (for reading the secure string from a command line as Gmail user passwords)
2. Send-MailMessage (for sending email messages using an SMTP server)
3. Get-Date (for getting the current date and time)
4. Get-Content (for reading an encrypted password from a file)
5. Get-ADUser (for getting newly added users from AD)
6. New-ScheduledTaskTrigger (for creating a new scheduled task trigger)
7. Register-ScheduledTask (for scheduling the new task in Task Scheduler)

I have run this script on Windows Server 2016. You can edit it in accordance with your environment needs. Follow these three steps to get everything working.

Step 1. Save Your Gmail Password as an Encrypted String in a Text File

Open PowerShell with elevated privileges and execute the following cmdlet. This prompts you to enter a password as a secure string and save it in text file as an encrypted string.

Read-Host -AsSecureString | ConvertFrom-SecureString | Out-File "C:\Users\securepassword.txt"

Step 2. Save the Script in a File with .ps1 Extension

Open Notepad, and copy and paste the following code. Save the file as FindOutADUsers.ps1.

```
##Beginning of functions

Function Send-Email {

Param ($Email, $Credential,$attachment)

$From = "karim.buzdar@gmail.com"
$subject = "Welcome to yourdomain.com"
$SMTPServer = "smtp.gmail.com"

$SMTPPort = "587"
```

```
### Beginning of email body

$Body = "Dear User,<br><br>"
$Body += "Welcome to yourdomain.com <br><br>"
$Body += " This email will help you log in to your domain services. Follow these steps to log in to your domain:
<br><br>"
$Body += "Step 1. Enter your username <br><br>"
$Body += "Step 2. Enter your password, and press enter <br><br>"
$Body += " Please check the attached screenshot. If you have any problems, please call the help desk at
following number: <br><br>" $Body += "<b>Extension No: 121</b><br><br>"
$Body += "Regards,<br><br>" $Body += "Yourdomain.com Helpdesk"

### End of email body

Send-MailMessage -from $From -to $Email -Subject $subject -BodyAsHtml $Body -Attachments $attachment
-SmtpServer $SMTPServer -Port $SMTPPort -Credential $Credential -UseSsl

 }

### End of Functions

##### Beginning of main function

$When = ((Get-Date).AddDays(-1))
$UserName = "karim.buzdar@gmail.com" #Gmail username which is used for sending an email
$Password =  Get-Content
"C:\Users\Administrator.YOURDOMAIN\Desktop\FindOutADUsers\securepassword.txt" | ConvertTo-SecureString
#Reading a secure password from file and reversing it back into a secure string object
$Credential = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList
($UserName, $Password) #PSCredential for send-mail message cmdlet
$Attachment = "C:\Users\Administrator.YOURDOMAIN\Desktop\FindOutADUsers\Screenshot.png" #Image
sending as an attachment with email



foreach ($EmailAddress in Get-ADUser -filter {(whencreated -ge $When)} -Properties emailaddress | Select
-ExpandProperty emailaddress) #Iterating over each email of users

{

Send-Email -Email $EmailAddress -Credential $Credential -attachment $Attachment

Write-Host "Email sent: $EmailAddress"

}

### End of main function
```

Step 3. Schedule the Script Using Task Scheduler

In Notepad, create a new file. Paste the following script and save it with .ps1 extension.

```
$Trigger= New-ScheduledTaskTrigger -At 12:00am -Daily #Trigger the task daily at 12 AM
$User= "yourdomain\administrator"
$Action= New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument
"C:\Users\Administrator.YOURDOMAIN\Desktop\FindOutADUsers\FindOutADUsers.ps1"

Register-ScheduledTask -TaskName "FindOutADUsers" -Trigger $Trigger -User $User -Action $Action -RunLevel
Highest -Force
```

Execute the above script in PowerShell with elevated privileges, and you are done!

When a scheduled task runs successfully, newly added users in Active Directory will receive the following email:

Dear User,

Welcome to yourdomain.com.

This email will help you log in to your domain services. Follow these steps to log in to your domain:

Step 1. Enter your username.

Step 2. Enter your password, and press enter.

Please check the attached screenshot. If you have any problems, please call the help desk at following number:

Extension No. 121

Regards,

Yourdomain.com Help Desk

I hope this post will be useful to you. Your feedback (please, send your emails to *sysadmin.magazine@netwrix.com*) is always welcome, especially if something is not working in this script. Good luck!

With the 2017 IT Risks Report, Netwrix continues to analyze how organizations are working to combat various IT risks. The report is based on a survey of IT pros concerning the IT risks they deal with on a daily basis, the processes and resources they have in place to address those risks, and how prepared they are for today's cyber threats. Key findings presented in the report include:

65%

of organizations have experienced security incidents, due mainly to malware and human errors

47%

of organizations had operational issues, which were most frequently caused by accidental, incorrect or malicious user activity

48%

of respondents had compliance issues, mostly due to their inability to provide proof of compliance or quickly find required data

Read Full Report

2017
IT RISKS

Read Full Report

MAIN SECURITY FOCUS

59% Virtual infrastructure

61% Endpoint

52% On-prem systems

BIGGEST SECURITY RISKS

66% Employees

25% Hackers

ONLY

26% of organizations are well prepared to beatcyber risks

WHY?

57% Lack of budget

37% Insufficient staff training

54% Lack of time

FOCUS OF FUTURE INVESTMENTS

38% Data breach

34% Frau

33% Intellectual property theft

WHERE ORGANIZATIONS MOST NEED TO GAIN VISIBILITY INTO USER ACTIVITY

56% On-prem systems

52% Cloud systems

43% Mobile devices

Find Disabled or Inactive Users and Computers in Active Directory

by Nirmal Sharma

IT Security Expert

Active Directory ships with more than 450 PowerShell cmdlets that you can use to collect information about every object in Active Directory, such as disabled computer accounts and disabled user accounts; interact with the ADSI engine to perform certain useful operations; check the health of domain controllers; collect GPO information; and more.

In the first part of the article, we'll explain how you can use simple PowerShell commands to find disabled user accounts, disabled computer accounts, and inactive user accounts from a single Active Directory domain. The second part of the article provides a handy PowerShell script that you can use to collect the same information from multiple Active Directory domains.

How to Find Disabled Accounts Information from a Single Domain

- **Collecting Disabled Computer Accounts Information**

Although Microsoft has not designed any PowerShell cmdlet specifically to collect disabled computer accounts, you can use the **Get-ADComputer** cmdlet. To collect disabled user accounts information, you can always use the **Search-ADAccount** PowerShell cmdlet, which is explained shortly below.

```
Get-ADComputer -Filter {(Enabled -eq $False)} -ResultPageSize 2000 -ResultSetSize $null -Server
<AnyDomainController> -Properties Name, OperatingSystem
```

As you can see in the command above, we are instructing the Get-ADComputer cmdlet to look for computer accounts that have the "Enabled" property set to $False, which indicates that the account is disabled.
If you wish to export the output to a CSV file, use the Export-CSV cmdlet, as shown in the PowerShell command below:

```
Get-ADComputer -Filter {(Enabled -eq $False)} -ResultPageSize 2000 -ResultSetSize $null -Server
<AnyDomainController> -Properties Name, OperatingSystem | Export-CSV "C:\Temp\DisabledComps.CSV"
-NoTypeInformation
```

- **Collecting Disabled User Accounts Information**

To find disabled user accounts in Active Directory, you will be required to use the **Search-ADAccount** cmdlet, designed to query any accounts in Active Directory. It supports a number of properties. The main parameters that you can specify in **Search-ADAccount** are either –UsersOnly or –ComputersOnly. When you specify –UsersOnly, **Search-ADAccount** searches only for the user objects, and when –ComputersOnly is specified, only computer accounts are searched. To query disabled user accounts in Active Directory, you can execute the command below:

```
Search-ADAccount –AccountInActive –TimeSpan 90:00:00:00 –ResultPageSize 2000 –ResultSetSize $null | ?{$_.Enabled
–eq $True} | Select-Object Name, SamAccountName, DistinguishedName | Export-CSV "C:\Temp\InActiveUsers.CSV"
–NoTypeInformation
```

The above command uses the –TimeSpan parameter to find user accounts that have been inactive for the last 90 days. The output is exported to the C:\Temp\InActiveUsers.CSV file.

How to Find Disabled Accounts Information from Multiple Domains

When collecting information from multiple Active Directory domains, you need to ensure that the PowerShell script is able to loop through the each domain it finds in an Active Directory forest and then execute the PowerShell commands against the domain to collect the required information. You can use **ForEach** PowerShell cmdlet to execute PowerShell commands against each Active Directory domain, but you also need to ensure that the data for each domain is collected in a separate CSV file. In a nutshell, when collecting disabled user accounts, disabled computer accounts, and inactive user accounts from Active Directory domains, you need to design a PowerShell script that can address the following needs:

- A separate IT Team for each Active Directory domain
- A single script that can collect information from all Active Directory domains. In other words, you need to design a script that can be executed only once to collect required information from all Active Directory domains
- Ability to store disabled user accounts, disabled computer accounts, and inactive user accounts information in a separate CSV file for each domain. Once you have separate CSV files for each domain, you can distribute CSV files to the IT Team of each domain for them to take any actions

Keeping the above needs in mind, we can use the PowerShell script below. However, before you use the script, make sure you address the requirements mentioned below:

- You must execute the script from a Windows Server 2012 or later Operating Systems
- The current Active Directory forest name that is being used by the script is “NetWrix.Com.” You must change the Active Directory forest name in the $CurForestName variable and then execute the script
- Make sure to install Active Directory PowerShell modules on the computer from which you will run the script
- Make sure to create a directory with the name “C:\Temp” on the local computer
- You must open the PowerShell window in an elevated mode
- You must have permission to access all Active Directory domains.

Once you have met the above requirements, you can execute the PowerShell script below:

```
$DomList = "C:\Temp\DomList.TXT"
remove-item $DomList -ErrorAction SilentlyContinue
$CurForestName="NWBlog.Com"
$GetForest=Get-ADForest $CurForestName $Items = $R.Domains
ForEach ($Domains in $Items)
{
```

```
    Add-Content $DomList $Domain.Name
 }
Write-Host "Starting Script..."
ForEach ($DomInFile in $DomList)
{
    $DisabledCompsCSV = "C:\Temp\DisabledAccounts_Computers_"+$DomInFile+".CSV"
    Remove-item $DisabledCompsCSV -ErrorAction SilentlyContinue
    $DisabledUsersCSV = "C:\Temp\DisabledAccounts_Users_"+$DomInFile+".CSV"
    Remove-item $DisabledUsersCSV -ErrorAction SilentlyContinue
    $InActiveUsersReport = "C:\Temp\InactiveUsers_"+$DomInFile+".CSV"
    Remove-item $InActiveUsersReport -ErrorAction SilentlyContinue


    Get-ADComputer -Server $DomInFile –Filter {(Enabled –eq $False)} –ResultPageSize 2000 –ResultSetSize $null
-Properties Name, OperatingSystem | Export-CSV $DisabledCompsCSV -NoTypeInformation
    Search-ADAccount -Server $DomInFile –AccountDisabled –UsersOnly –ResultPageSize 2000 –ResultSetSize $null |
Select-Object SamAccountName, DistinguishedName | Export-CSV $DisabledUsersCSV –NoTypeInformation
    Search-ADAccount -Server $DomInFile –AccountInActive –TimeSpan 90:00:00:00 –ResultPageSize 2000
–ResultSetSize $null | ?{$_.Enabled –eq $True} | Select-Object Name, SamAccountName, DistinguishedName |
Export-CSV $InActiveUsersReport –NoTypeInformation
 }
Write-Host "Script Finished collecting required information. Please check report files under C:\Temp folder"
```

To sum up, it is easy to understand the complete script, but just to ensure you understand the objectives of the script, the script performs the following functions:

The above script collects all Active Directory domains from the Active Directory forest specified in the $CurForestName variable and then stores the domain names in the C:\Temp\DomList.CSV file.The “C:\Temp\DomList.CSV” file is used by the second “ForEach” loop in the script.The script collects disabled users, disabled computer accounts, and inactive user accounts from each domain by executing the Get-ADComputer and Search-ADAccount PowerShell commands.The report is generated in a CSV file for each domain. You can find all CSV reports under the C:\Temp folder on the computer from which you run the script.

Once the CSV reports are generated, you can send output files to each IT Team via e-mail, or you can embed the “Send-MailMessage” cmdlet in the script so the script sends an e-mail with a CSV report to each IT Team. We will talk about the “Send-MailMessage” cmdlet in the upcoming part of this article series.

If you are a busy person you should try some free tools to automate the task and safe your time. For instance, armed with Inactive User Tracker you will be able to quickly clean out or lock down all of your stale user accounts.

Create Active Directory Users in Bulk and Email Their Credentials

by Karim Buzdar

IT Expert

Every year, educational institutions enroll hundreds of students. As a result, the institutions' system engineers require extra time and effort to create an AD account for each student and then to manually inform the students about their usernames and passwords.

This step-by-step guide will make system engineers' lives a little easier with the help of PowerShell scripting. The script enables you to create AD user accounts from a CSV file, assign random passwords to them, and then send those usernames and passwords to the new students in welcome emails.

I have used the following cmdlets in my script. Click on any individual cmdlet for more details.

1. Import-CSV (for importing user details)
2. Get-Credential (for getting email credentials)
3. Get-Random (for generating passwords)
4. New-ADUser (for creating new AD users)
5. Get-Content (for reading email contents from a file)
6. Send-MailMessage (for sending emails to users)

To send the emails with credentials to the new users, I am using a Gmail SMTP server. However, you can configure the script to your IT environment and your mail server.
The script below was written and tested on DC running on Windows Server 2016. If you need to run the script on Windows Server 2012, 2012 R2, or 2008, you may have to import some PowerShell modules.

Before you run the script, cross-check for the following **prerequisites**:

- A CSV file with the students' info (first name, last name, Sam account name, and email address)
- A text or Word (doc or docx) file with a welcome email message
- An enabled Internet connection (if you want to use a Gmail SMTP server)

If you have met all these requirements, you can create AD accounts and send welcome emails in bulk by following two simple steps:

Step 1.

Copy and paste the script into a text file and save it with the **.ps1** extension:

```
##Beginning of functions

Function Format-Email {

Param ([string]$WCEmailContent,[string]$UserPrincipalName,[string]$UserPassword)

return $WCEmailContent + "User Name = $UserPrincipalName
Password = $UserPassword
Thank You IT
Department"
}

Function Send-Email {

Param ($Email, $Credential,[string]$WCEmailContent)

$From = "karim.buzdar@gmail.com"
$subject = "Domain Account Details"
$Body = $WCEmailContent
$SMTPServer = "smtp.gmail.com"
$SMTPPort = "587"
```

```
Send-MailMessage -from $From -to $Email -Subject $subject -Body $Body -SmtpServer $SMTPServer -Port $SMTPPort -Credential $Credential -UseSsl

 }

###End of Functions

##### Beginning of main programme

$Credential = (get-Credential) #Getting email credentials to be used for From field in Email message $UsersFilePath = "C:\Users\Administrator.YOURDOMAIN\Desktop\WC\users.csv" # Files of users information (First name, Last name, Sam account name, Email address)
$WCEmailFile = "C:\Users\Administrator.YOURDOMAIN\Desktop\WC\WCEmail.docx" # File containing welcome email message

import-csv -path $UsersFilePath | foreach {

$Name = ($_.GivenName + " " + $_.LastName)
$UserPrincipalName = ($_.SamAccountName + ”@yourdomain.com”)
$UserPassword = Get-Random -maximum 20000 -Minimum 100
$UserPassword = "@" + "user" + $UserPassword.ToString()
#Create user in Students OU

new-aduser -name $Name -enabled $true -GivenName $_.GivenName –surname $_.lastName -accountpassword (convertto-securestring $UserPassword -asplaintext -force) -changepasswordatlogon $true -samaccountname $_.SamAccountName –userprincipalname $UserPrincipalName -EmailAddress $_.EmailAddress -Path 'OU=Students,DC=yourdomain,DC=com' -ErrorAction Stop

$WCEmailContent = Format-Email -WCEmailContent (Get-Content $WCEmailFile) -UserPrinciPalName $UserPrincipalName -UserPassword $UserPassword

Send-Email -Email $_.EmailAddress -Credential $Credential -WCEmailContent $WCEmailContent

Write-Host "User Created: $Name"

}

### End of main program
```

Step 2.

Open PowerShell with your elevated privileges and execute the script file that you created in Step 1.

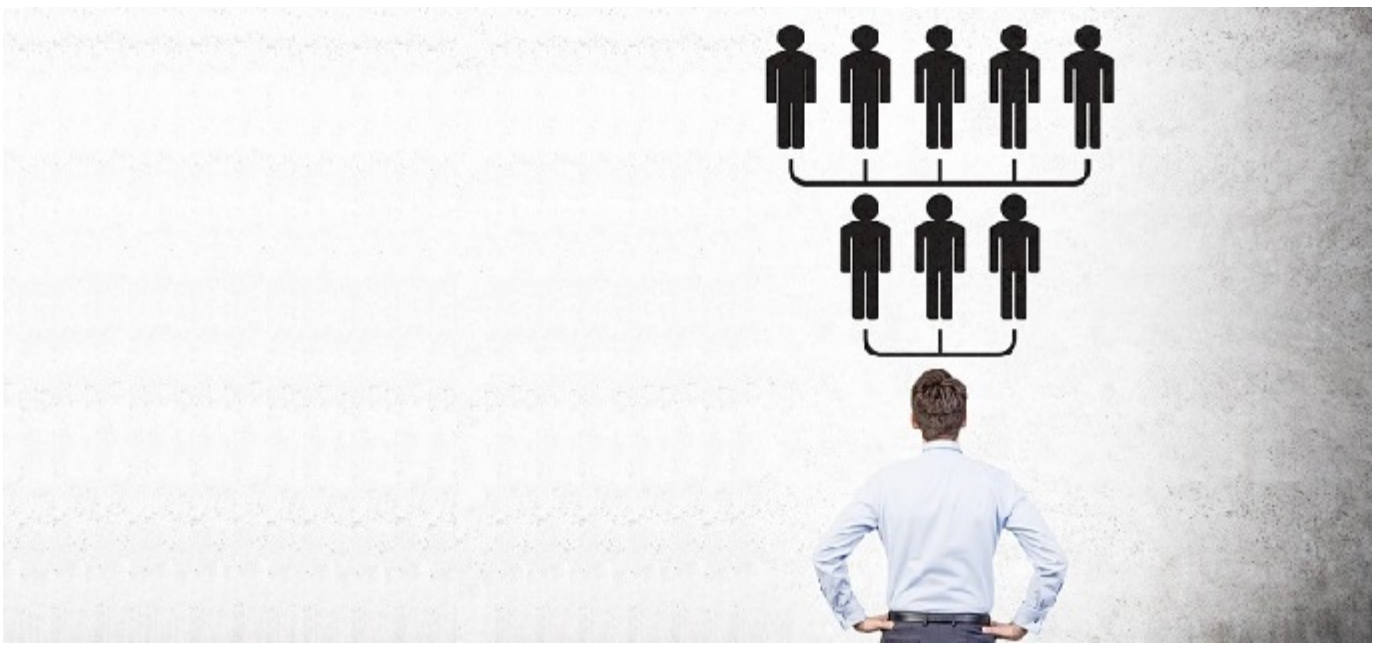
Once you have successfully executed the script, users can immediately log in to the domain. You can verify the account creation in Active Directory Users and Computers console and email the notification about the account creation by using CC field in Send-MailMessage cmdlet.

Detecting Delegated Permissions in Active Directory

by Matt Hopton
Network Architect

Security permissions in Active Directory can be a tricky topic. Not only does Microsoft hide them from you by default in Users and Computers, there is also no built-in tool to get an overall picture of how permissions have been applied to AD.

In this article, I'll take you through the basics of delegating, removing permissions, using built–in tools to find permissions that have been delegated, and finally a custom PowerShell script that scans AD.



Why Delegate?

Imagine you're the head of a large company with several departments: finance, HR, sales, upper management. If every user who forgot their password had to call the IT helpdesk, you would be swamped with calls. Instead, you could delegate permissions to the head of each department so that he or she can reset his or her own team's passwords.

Another classic use case for delegation is the ability for staff to send emails as each other—either a shared mailbox, or a PA sending email on behalf of his or her boss.

Give everyone Domain Admin?

You might've thought—okay, let's give each department head Domain Admin permissions, then they can reset the passwords when required. Whilst this is technically true, they would then be able to do anything you can do—including accessing user data. Data breach in the making!

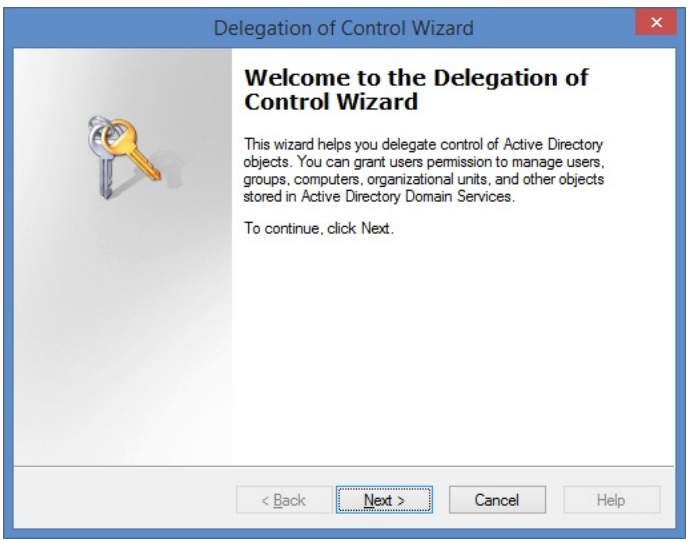
How to delegate permissions in Active Directory

The correct way of achieving this of course is by using Delegation. This will allow you to individually select permissions that you want to give away.

1. Open up Active Directory Users and Computers and connect to your favourite test domain.

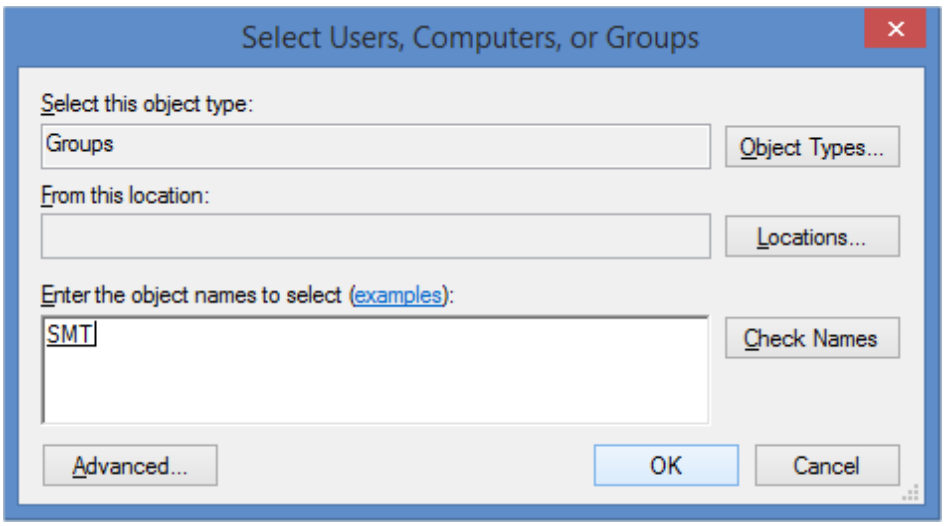
2. Right click on the department Organisational Unit that you wish to give permission to reset passwords.

3. Find the 'Delegate Control' option (this should be the first option in the list). Click this and press Next.



4. You are now prompted to choose users or groups to whom you wish to delegate control—these are the people who you want to be able to perform a task.
It is HIGHLY recommended that you create a security group for each set of permissions that you are delegating (i.e., one for 'Sales – Password Reset Ability', 'HR – Password Reset Ability'). This allows you to very easily add/remove users from these groups in the future, rather than messing around with permissions directly.

5. Go ahead and add in a group. (In the figure below I've added in our senior management team.)



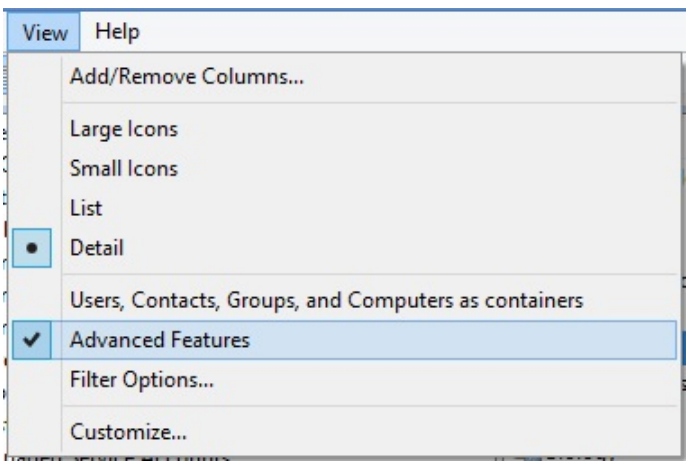
6. Press next and then pick the option 'Reset user passwords and force password change at next logon'.

7. Press next and then finish—you're done!

Removing Delegated Permissions in Active Directory

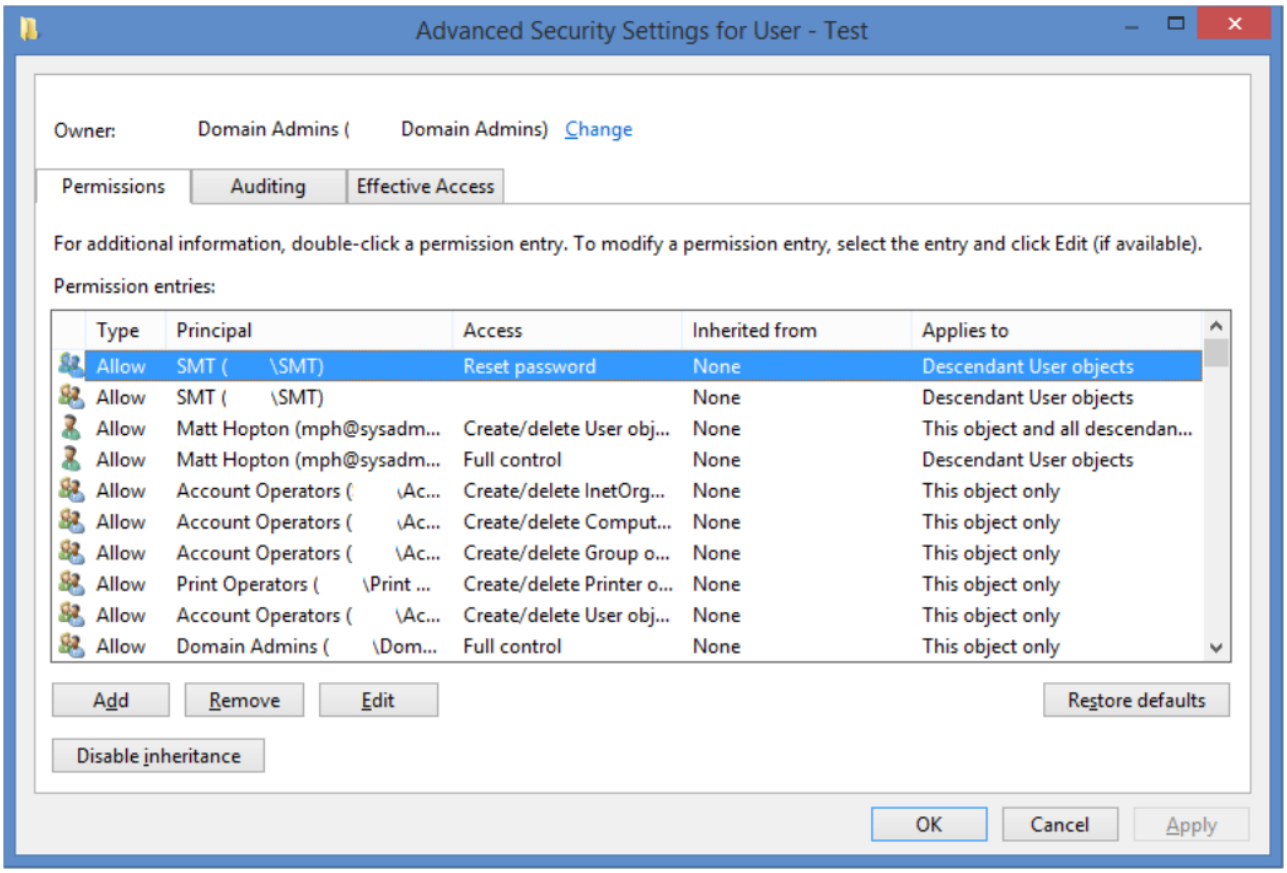
Now that you've completed the wizard, you might be wondering how you check that you did actually delegate permissions and how to remove them again.

1. From Users and Computers, press the View menu and make sure 'Advanced Features' is ticked.



2. By ticking this box, you can see the security tab when you choose Properties on objects in Active Directory.
Right click on the same OU that you just delegated permissions and choose Properties, then the Security Tab.

3. Choose 'Advanced' and then scroll up and down until you find the group to whom you just gave permissions. You should see the 'Reset Password' permission listed under 'Access'.



4. If you wanted to remove this permission, you could select it and press Remove, but leave it in place for now and press Cancel.

As you can see, finding delegated permissions is quite hard—especially when you consider that you can delegate not only to OUs, but also to security groups, and even to user objects themselves.

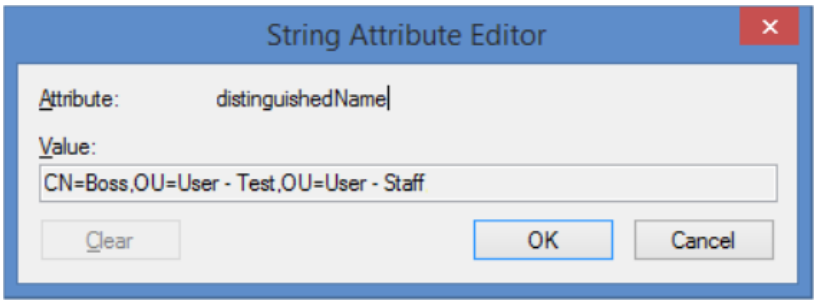
Permission Hunting (Using DSACLS)

So, a nightmare scenario for you to consider is someone has reset the boss' password and you need to find out who had permission to do it. (If you had Netwrix Auditor installed, you could have a look in there, but for now we'll assume you don't.)

1. In *Users and Computers*, navigate to the user object that you want to check permissions for.

2. Right click and select Choose Properties, then 'Attribute Editor'

3. Scroll and double click on ‘distinguishedName’. Copy this string for later.



4. Open up a command prompt and type ‘dsacls’, followed by pasting the string you just copied, enclosed in speech marks:

```
C:\Users\MPH>dsacls "CN=Boss,OU=User - Test,OU=User - Staff,DC=domain,DC=local"
```

5.Hit enter.

6. At this point you’ll probably realise there’s too many permissions on screen—you have two options:

a. Use |more on the end of the command to display one screen fully at a time.

b. Pipe the output into a text file and read that instead by using > filename.txt.

7. I went for the 2nd option, opened the file in notepad and eventually found that the senior management group has permissions to reset the boss’ password!

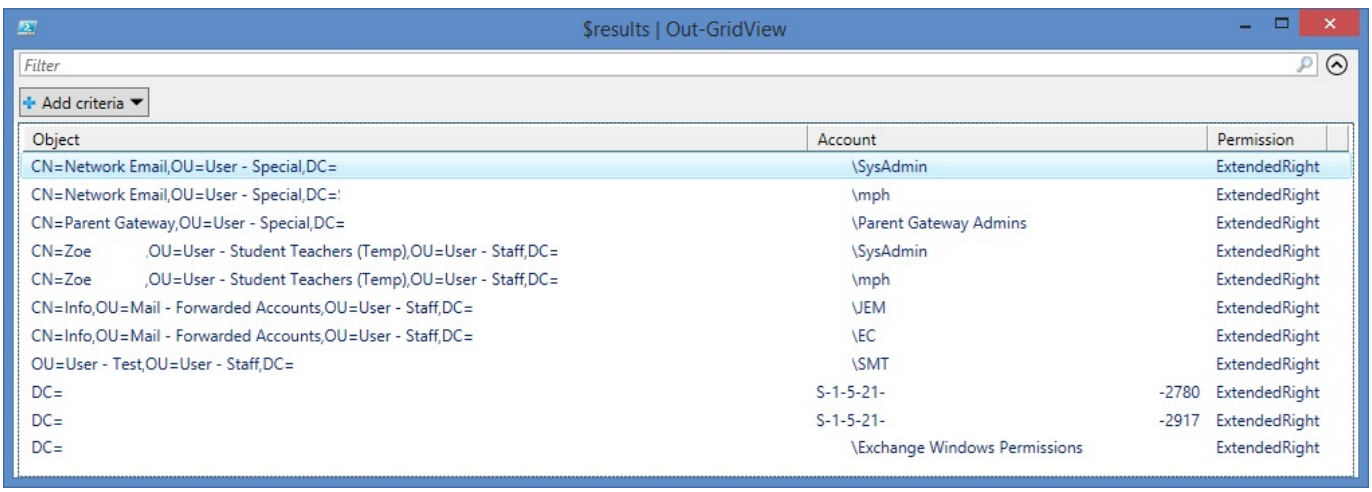
```
Allow XXXX\SMT                          Reset Password    <Inherited from parent>
```

Permission Hunting (Using PowerShell)

Now that you’ve discovered delegation, you might be wondering if there are any delegations that you don’t know about—either from past employees, or malicious administrators.

I’ve put together a short PowerShell script which will search each delegable object type and list the two common permission delegations—reset password and ‘send-as’ (from Exchange).

Here’s a sample run from a domain:

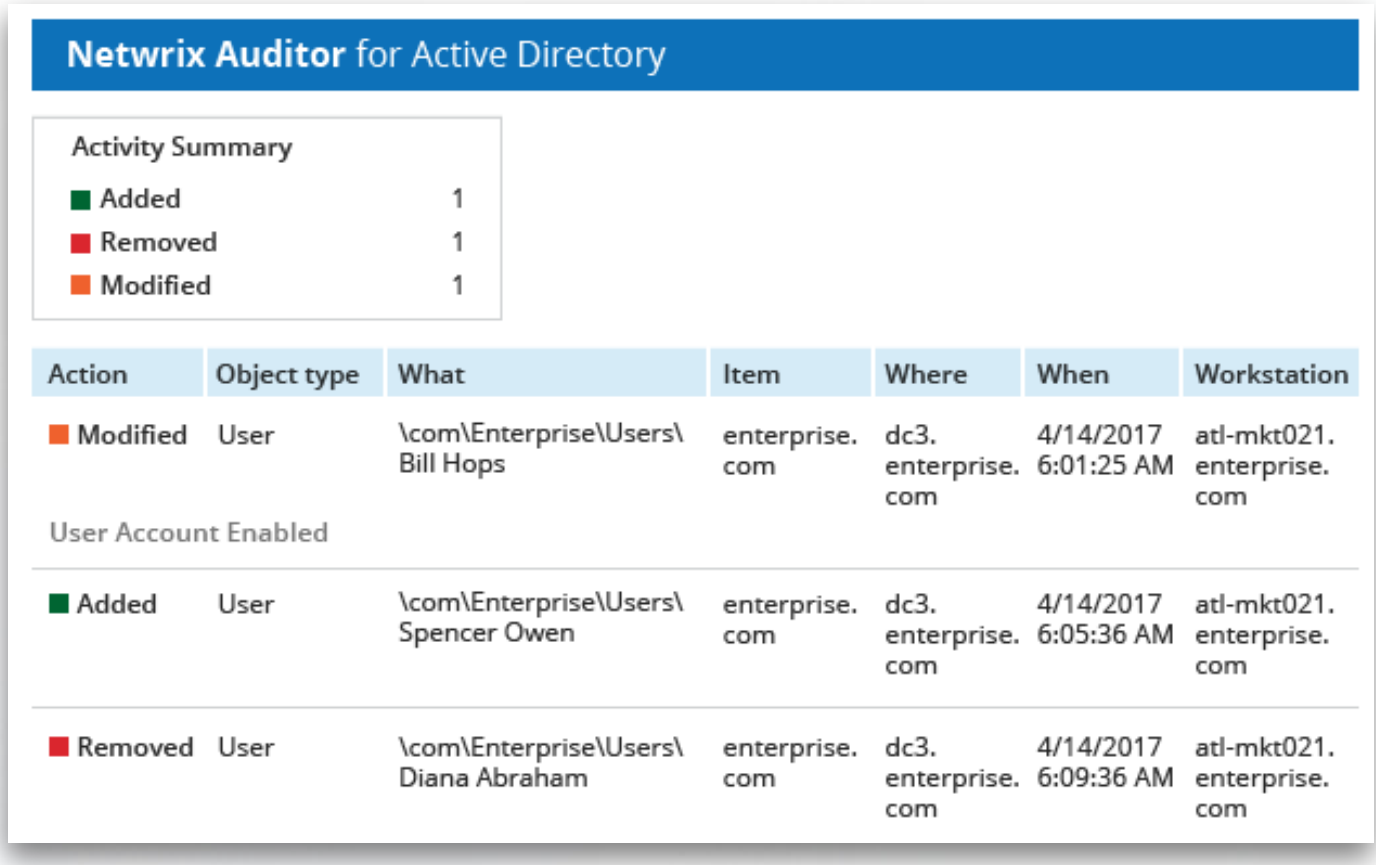


Free Tool of the Month

Netwrix Auditor Free Community Edition

Freeware tool provides visibility into activity in your hybrid cloud IT environment, so you'll no longer have to worry about missing critical changes to AD objects, file server permissions, or other security incidents

Stay updated on user activity in your hybrid cloud IT environment



Download Freeware

How to Find Out Who Unlocked a User Account

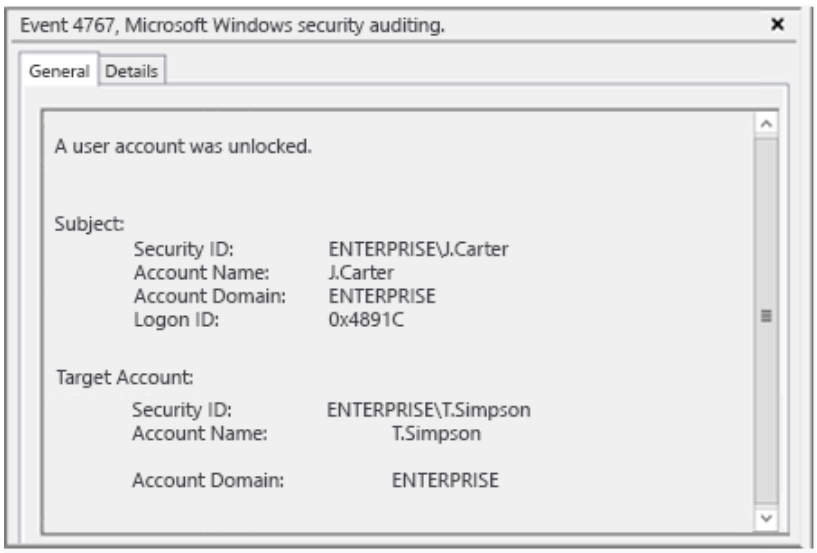
1. Run gpedit.msc → Create a new GPO → Edit it: Go to "Computer Configuration" → Policies → Windows Settings → Security Settings → Advanced Audit Policy Configuration → Audit Policies → Account Management:
   - Audit User Account Management → Define → Success and Failures.

2. Go to Event Log → Define:
   - Maximum security log size to 4gb
   - Retention method for security log to "Overwrite events as needed".

3. Link the new GPO: Go to "Group Policy Management" → Right-click domain or OU → Choose Link an Existing GPO → Choose the GPO that you created.

4. Force the group policy update: In "Group Policy Management" right-click the defined OU → Click "Group Policy Update".

5. Open Event Viewer → Search security log for event ID 4767 (A user account was unlocked).



Try Netwrix Auditor 9.0:

Shield Your IT Environment from Ransomware & Malicious Insiders

netwrix.com/auditor9.html

netwrix.com | Follow us

Corporate Headquarters: 8001 Irvine Center Drive, Suite 200 Irvine, CA 92618

Phone: 1-949-407-5125
Toll-free: 888-638-9749
EMEA: +44 (0) 203-318-02