7 Kasım 2023 Salı 14:34

Mail gönderme

```
$username = "usr8121";
$Password = "+adasdsaadsadsa";
$path = "C:\BTOKS\Reboot-LOG.txt";
$email = "smyonetimveraporlama@albarakaturk.com.tr";
function Send-ToEmail([string]$email, [string]$attachmentpath){

   $message = new-object Net.Mail.MailMessage;
   $message.From = $email;
   $message.To.Add("smyonetimveraporlama@albarakaturk.com.tr");
   $message.Subject = "ALBISTSMNGAP01 SM Sunucu Hk:";
   $message.Body = "Haftalık ALBISTSMNGAP01 Sm sunucusu yeniden başlatılmıştır.";
   $attachment = New-Object Net.Mail.Attachment($attachmentpath);
   $message.Attachments.Add($attachment);

   $smtp = new-object Net.Mail.SmtpClient("mail.albarakaturk.com.tr", "25");
   $smtp.EnableSSL = $true;
   $smtp.Credentials = New-Object System.Net.NetworkCredential($Username, $Password);
   $smtp.send($message);
   write-host "Mail Sent" ;
   $attachment.Dispose();
 }
Send-ToEmail  -email $email -attachmentpath $path;
```

---

yetkisiz dosya tasıma

```
robocopy.exe "D:/DTOM_RPA"  "D:/RPA/DTOM_RPA" /MIR /COPY:D /R:3 /W:10 /MT:16 /Z /NP /NDL
```

yetkili dosya taşıma

```
robocopy.exe "T:\DATA03" "C:\Users\usr7572a\Desktop\Liste_yedek" /MIR /COPYALL /DCOPY:DAT /SECFIX /R:3 /W:10 /MT:16 /Z /NP /NDL /TEE /log:D:\Robocopy.log
```

---

belirli pathde bir dosyayı silme

```
$user= Read-Host "username girin"
$pc=Read-Host "pc name girin"
$folders= "\\$pc\c$\Users\$user\AppData\Roaming\Microsoft\Teams\logs.txt"
Remove-Item $folders
```

Gizli klasörü açığa çıkarma

```
attrib -h "Folder Path" -s -r
```

-h	gizli olmaktan çıkarıyor
-s	SYSTEM özelinden çıkarıp değişikliği sağlıyor
-r	Klasör korumasını kaldırıyor

---

manager bilgisi eksik olanlar

```
Get-aduser -f *   -Properties name,manager  | where {$_.enabled -eq "Enabled" -and $_.Manager -eq $null -and $_.sAMAccountName -like "USR*" -and $_.sAMAccountName -like "USR*" -and $_.sAMAccountName -notlike "*a*" -and $_.sAMAccountName -like "USR*" -and $_.sAMAccountName -notlike "*b*"} |  select  -Property name,sAMAccountName,manager|measure
```

---

uzak pcdeki processler ve bir prosesi durdurma

```
Invoke-Command -ComputerName  $pc_name -ScriptBlock {Get-Process}
Invoke-Command -ComputerName  $pc_name -ScriptBlock {Get-Process Teams |Stop-Process }
```

---

uzak pc ekranlara uyarı gönderme

```
$name =  "ALBISTRDSSH001"
$msg =  "TEST SUNUCUSU BU AKSAM ITIBARIYLE SONLANDIRILACAKTIR. YARINDAN ITIBAREN LUTFEN ESKI RDS linkiniz ile GIRIS YAPINIZ!!!"

Invoke-WmiMethod -Path Win32_Process -Name Create -ArgumentList "msg * $msg" -ComputerName $name
```

---

uzak pc bağlı kullanıcıları dışarı atma

```
Invoke-Command -ComputerName albistrdssh081 -ScriptBlock {

## Yukardaki değişkenlerden ihtiyacınız olanı script alanına koyup calıstırabilirsiniz
quser | Select-String "ACT" | ForEach {logoff ($_.tostring() -split ' +')[2]} ## logoff all user
## Yukardaki değişkenlerden ihtiyacınız olanı script alanına koyup calıstırabilirsiniz

}
```

---

txt dosyasındaki disable userları bul

```
$userkontrol= (Get-content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\liste.txt")
Clear-Content C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt
Set-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt" "Hesabı Disable Olan Kullanıcılar"
ForEach ($sicil in $userkontrol) {


Get-ADUser  $sicil -Properties  enabled | Select name, enabled  | Select-String -Pattern  'False' >> "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt"
```

```
}

$message= C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt
```

Ou ya göre kullanıcı filtreleme (dış kurum personellerinden disable olmayanlar)

```
Get-ADUser -Filter  'Enabled -eq $true' -SearchBase "OU=USERS,OU=998-KURUM-DISI,OU=GENEL-MUDURLUK,DC=albarakaturk,DC=local" |select UserPrincipalName , Enabled
```

---

uzak pclerde belli bir dizindeki son değiştirilme tarihi ve uptime ı(liste.txt'de pc adları var)

```
$Foldername=(Get-Content "C:\Users\USR8121b\Desktop\liste.txt")

ForEach ($Folder in $Foldername) {

   write-Host $Folder

   Invoke-Command -ComputerName $Folder -ScriptBlock {

   $a= [system.io.directoryinfo]"C:\ProgramData\{12DA930D-C763-4261-B16D-3E7261A6601F}" | Select-Object LastWriteTime
   $b= Get-CimInstance -ClassName Win32_OperatingSystem | Select LastBootUpTime

   write-Host  $a$b
   }

}
```

SID TO USERNAME

```
$SID = New-Object -TypeName System.Security.Principal.SecurityIdentifier -ArgumentList ('S-1-5-21-2660546281-4054462683-3821121609-45218')
$user = ($SID.Translate( [System.Security.Principal.NTAccount])).Value
$user
```

USERNAME TO SID

```
$objUser = New-Object System.Security.Principal.NTAccount("Usr3265")
$strSID = $objUser.Translate([System.Security.Principal.SecurityIdentifier])
$strSID.Value
```

SUNUCU EKRANLARINA BİLDİRİ GÖNDERME

```
# $computers=(Get-Content "C:\D\list.txt")                          ### Liste hali


$computers=(02..98 + 111..130 | %{"albistrdssh$("{0:D3}" -f$_)"}).ToLower()       ### RDS PC vs Aralık hali


foreach($computer in $computers) {
$msg =  "SADECE EDGE GECISI YAPILAN SUBE VE BIRIMLARIMIZ ICIN ActiveX UYGULAMASINDA (YAZDIRMA) SORUNU ILE ILDIGI CALISMA YAPILMISTIR.CALISMANIN AKTIF OLABILMESI ICIN LUTFEN OTURUMUNUZU KAPATIP TEKRAR ACINIZ."
Invoke-WmiMethod -Path Win32_Process -Name Create -ArgumentList "msg * $msg" -ComputerName $computer
}
```

SUNUCU EKRANINA BİLDİRİ GÖNDERME

```
$name =  "ALBISTRDSSH001"
$msg =  "TEST SUNUCUSU BU AKSAM ITIBARIYLE SONLANDIRILACAKTIR. YARINDAN ITIBAREN LUTFEN ESKI RDS linkiniz ile GIRIS YAPINIZ!!!"

Invoke-WmiMethod -Path Win32_Process -Name Create -ArgumentList "msg * $msg" -ComputerName $name
```

---

tüm seeionlarda gez dosya dizinini sil

```
$computers=(02..98 + 111..130 | %{"albistrdssh$("{0:D3}" -f$_)"}).ToLower()

 foreach($computer in $computers) {

      Invoke-Command -ComputerName $computer  -ScriptBlock {


         rmdir /q /s %programdata%\{12DA930D-C763-4261-B16D-3E7261A6601F}


   }

}
```

---

tüm sessionlarda gez dosyayı sil

```
$computers=(02..98 + 111..130 | %{"albistrdssh$("{0:D3}" -f$_)"}).ToLower()

 foreach($computer in $computers) {

      Invoke-Command -ComputerName $computer  -ScriptBlock {

      Remove-Item -Path C:\ditto.exe
      Remove-Item -Path C:\ditto.reg

   }

}
```

tek kullanıcı tek session ditto sil

```
Invoke-Command -ComputerName albistrdssh078 -ScriptBlock {

## Yukardaki değişkenlerden ihtiyacınız olanı script alanına koyup calıstırabilirsiniz


Set-ItemProperty Registry::HKEY_USERS\S-1-5-21-2660546281-4054462683-3821121609-7281\SOFTWARE\Ditto -Name “ExpiredEntries” -Value “3” -Type DWord
Set-ItemProperty Registry::HKEY_USERS\S-1-5-21-2660546281-4054462683-3821121609-7281\SOFTWARE\Ditto -Name “MaxEntries” -Value “25” -Type DWord


## Yukardaki değişkenlerden ihtiyacınız olanı script alanına koyup calıstırabilirsiniz

}
```

toplu ditto temizle

```
$computers=(02..98 + 111..130 | %{"albistrdssh$("{0:D3}" -f$_)"}).ToLower()

foreach($computer in $computers) {

Invoke-Command -ComputerName  $computer -ScriptBlock {


$PatternSID = 'S-1-5-21-\d+-\d+\-\d+\-\d+$'
$ProfileList = Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList\*' | Where-Object {$_.PSChildName -match $PatternSID} |
    select  @{name="SID";expression={$_.PSChildName}}|FT
          @{name="UserHive";expression={"$($_.ProfileImagePath)\ntuser.dat"}},
         @{name="Username";expression={$_.ProfileImagePath -replace '^(.*[\\\/])', ''}} | FT
## Yukardaki değişkenlerden ihtiyacınız olanı script alanına koyup calıstırabilirsiniz
$LoadedHives = gci Registry::HKEY_USERS | ? {$_.PSChildname -match $PatternSID} | Select @{name="SID";expression={$_.PSChildName}}

Foreach ($item in $LoadedHives) {


Set-ItemProperty Registry::HKEY_USERS\$($Item.SID)\SOFTWARE\Ditto -Name “ExpiredEntries” -Value “3” -Type DWord
Set-ItemProperty Registry::HKEY_USERS\$($Item.SID)\SOFTWARE\Ditto -Name “MaxEntries” -Value “25” -Type DWord


}

}
}
```

**** taşıma

```
#### Zaman içinde öldür. klasor aç aktar

$zaman = (Get-Date).AddDays(-60) # 60 gün oncesi
$mt940_liste = Get-ChildItem -Path D:\MT940_Ekstre -Directory | Select-Object -ExpandProperty Name   | Out-File -FilePath F:\MT940_Ekstre\mt940.txt  # Dizinleri bul
$mt940_list=(Get-Content "F:\MT940_Ekstre\mt940.txt")
foreach($mt940 in $mt940_list){
$nereden = "D:\MT940_Ekstre\$mt940"
$nereye = "F:\MT940_Ekstre\$mt940"

New-Item -Path "F:\MT940_Ekstre\$mt940" -ItemType Directory # dizin yoksa ekle

Get-ChildItem $nereden -ErrorAction SilentlyContinue | Where-Object {$_.LastWriteTime -le "$zaman"} |move-item  -destination $nereye # 60 gün oncesi taşı

}




$sourceFolder = "C:\Users\yunus.kaya\Desktop\sil"
$destinationFolder = "C:\Users\yunus.kaya\Desktop\sil\Zip"

Get-ChildItem -Path $sourceFolder | ForEach-Object {
   $fileName = $_.Name
   $zipFileName = $fileName + ".zip"
   $fullZipPath = Join-Path -Path $destinationFolder -ChildPath $zipFileName
   Compress-Archive -Path $_.FullName -DestinationPath $fullZipPath
}
```

**** excelden pdfye çevirme çoklu

```
$path = “C:\Users\usr8121\Desktop\pdfolcak”
$xlFixedFormat = “Microsoft.Office.Interop.Excel.xlFixedFormatType” -as [type]
$excelFiles = Get-ChildItem -Path $path -include *.xls, *.xlsx -recurse
$objExcel = New-Object -ComObject excel.application
$objExcel.visible = $false
foreach($wb in $excelFiles)
{
$filepath = Join-Path -Path $path -ChildPath ($wb.BaseName + “.pdf”)
$workbook = $objExcel.workbooks.open($wb.fullname, 3)
$workbook.Saved = $true
“saving $filepath”
$workbook.ExportAsFixedFormat($xlFixedFormat::xlTypePDF, $filepath)
$objExcel.Workbooks.close()
}
$objExcel.Quit()
```

****tarama altındaki klasorleri gezip 7 gün onceki dosyları silecek

```
$isimler = Get-Content -Path "E:\liste.txt"
foreach ($isim in $isimler) {
Get-ChildItem -Path E:\tarama\$isim\  -File -Recurse | Where-Object {$_.LastWriteTime -le $(get-date).Adddays(-7)} | foreach { $_.Delete()}

}
```

**** PowerShell kullanarak klasör iznini dışarı aktarma

Kaynak <https://thesleepyadmins.com/2020/04/28/export-folder-permission-using-powershell/>

```
## Setting script parameters
param(
[parameter(Mandatory = $true)]
[String]$FolderPath,
[parameter(Mandatory = $true)]
[String]$ExportPath
)

## Results variable
$results = @()


## Get Folders
$error.clear()
$Folders = Get-ChildItem -Path $FolderPath -Directory |  Select-Object Name,FullName,LastWriteTime,Length
foreach ($err in $Error) {
$err.Exception.Message | Out-File $ExportPath\AccessDenied.txt -Append
}

## Loop through folders
foreach ($Folder in $Folders){

## Get Size of each folder
$size = ((Get-ChildItem -Path $Folder.FullName -Recurse | Measure-Object -Property Length -Sum -ErrorAction SilentlyContinue).Sum / 1MB)

## Get access control list
$Acls = Get-Acl -Path $Folder.FullName -ErrorAction SilentlyContinue

## Loop through ACL
foreach ($Acl in $Acls.Access) {

if ($Acl.IdentityReference -notlike "BUILTIN\Administrators" -and $Acl.IdentityReference -notlike "CREATOR OWNER" -and
$Acl.IdentityReference -notlike "NT AUTHORITY\SYSTEM" -and $Acl.FileSystemRights -notlike "-*" -and  $Acl.FileSystemRights -notlike "268435456"`
-and $Acl.IdentityReference -notlike "S-1-*"){

## format properties for result hash table
$properties = @{
FolderName = $Folder.Name
FolderPath = $Folder.FullName
IdentityReference = $Acl.IdentityReference.ToString()
Size = [math]::Round($size,2)
Permissions = $Acl.FileSystemRights
AccessControlType = $Acl.AccessControlType.ToString()
IsInherited = $Acl.IsInherited
}
$results += New-Object psobject -Property $properties
            }
        }

    }

## Export results
$results | Select-Object FolderName,FolderPath,IdentityReference,Size,Permissions,AccessControlType,IsInherited |
Export-Csv -Path $ExportPath\PermissionExport.csv -Append -NoTypeInformation
```

Not:

(Tüm alt klasörleri de aramak istiyorsanız, 14. satırı değiştirin ve Get-ChildItem komutuna -Recurse ekleyin)

Kaynak <https://thesleepyadmins.com/2020/04/28/export-folder-permission-using-powershell/>

****

```
$zip_adi = Get-Date -Format 'ddMMyyyy_BPO_GIB_ARSIV'
$zaman = (Get-Date).AddDays(-233)
$nereden = "D:\EBOSYasalBildirimler\YASAL_BILDIRIMLER_PROD\GIB_2017\GIDEN\BPO_GIB_ARSIV"
$nereye = "F:\EBOSYasalBildirimler_ARSIV\YASAL_BILDIRIMLER_PROD\GIB_2017\GIDEN\BPO_GIB_ARSIV"

Get-ChildItem $nereden -ErrorAction SilentlyContinue | Where-Object {$_.LastWriteTime -le "$zaman"} |move-item  -destination $nereye
& "C:\Program Files\7-Zip\7z.exe" a F:\EBOSYasalBildirimler_ARSIV\YASAL_BILDIRIMLER_PROD\GIB_2017\GIDEN\BPO_GIB_ARSIV\$zip_adi.zip F:\EBOSYasalBildirimler_ARSIV\YASAL_BILDIRIMLER_PROD\GIB_2017\GIDEN\BPO_GIB_ARSIV
bağlam menüsü var
```

*** klasor yetkilerini bul (Sunucuda işe yarar) ve masaüstüe kaydet*****

```
$FolderPath = Get-ChildItem -Directory -Path 'D:\\MSK_SIB' -Recurse -Force
$Output = @()
ForEach ($Folder in $FolderPath) {
   $Acl = Get-Acl -Path $Folder.FullName

   ForEach ($Access in $Acl.Access) {
      if ($Acl.IdentityReference -notlike "BUILTIN\Administrators" -and $Acl.IdentityReference -notlike "CREATOR OWNER" -and
         $Acl.IdentityReference -notlike "NT AUTHORITY\SYSTEM" -and $Acl.FileSystemRights -notlike "-*" -and  $Acl.FileSystemRights -notlike "268435456"`
         -and $Acl.IdentityReference -notlike "S-1-*"){
      $Properties = [ordered]@{'Folder Name'=$Folder.FullName;'Group/User'=$Access.IdentityReference;'Permissions'=$Access.FileSystemRights;'Inherited'=$Access.IsInherited}
      $Output += New-Object -TypeName PSObject -Property $Properties
      }
```

```
    }
}
 $Output

Start-Transcript C:\Users\usr8121a\Desktop\\MSK_SIB.txt
```

**** 7gün ve öncesi .log dosyalarını ilgili dizinden sil

```
$TargetFolder = "C:\PerfLogs\System\Performance\"
      $filetype= "*.log"
      $file_age = "7"

      foreach ($i in Get-ChildItem $TargetFolder -recurse -include $filetype)
      {
      if (($i.Lastwritetime -lt($(Get-Date).AddDays(-$file_age))))
      {
      $fname=$i.FullName
      Remove-Item $fname -Force
      }
      }
```

**** web.config encrypt işlemi

```
#Not: folderpathe encrpt olacak web configin yolunu yaz bacupfoldera eski hallerini yedekleyecek

# Klasör ve yedekleme klasörü değişkenleri
$folderPath = "D:\IIS"
$backupFolderPath = "D:\webConfigBackups"

# Klasördeki tüm web.config dosyalarını al
$configFiles = Get-ChildItem -Path $folderPath -Filter "web.config" -Recurse

foreach ($configFile in $configFiles) {
   $configDirName = Split-Path -Path $configFile.DirectoryName -Leaf
   Write-Host "*******----------$configDirName----------*********"
   Write-Host "İşlem başlıyor: $($configFile.FullName)"

   # web.config dosyasının içeriğini oku
   [xml]$xmlConfig = Get-Content $configFile.FullName

   # ConnectionStrings bölümünü kontrol et
   $connectionStrings = $xmlConfig.configuration.connectionStrings

   if ($connectionStrings) {
      # Şifreli olup olmadığını kontrol et
      $isEncrypted = $connectionStrings.'configProtectionProvider'
      if ($isEncrypted) {
         Write-Host "$($configFile.FullName) dosyası zaten şifreli."
      } else {
         Write-Host "$($configFile.FullName) dosyası şifreli değil, işlem devam ediyor."

         # Uygulama havuzunu bul
         $appPoolName = (Get-WebConfigurationProperty -Filter system.applicationHost/sites/site/application -Name applicationPool -PSPath "IIS:\" | Where-Object { $_.physicalPath -eq $configFile.DirectoryName }).value
         Write-Host "Uygulama havuzu bulundu: $appPoolName"

         # Yedekleme klasörünü oluştur

         $appBackupFolder = Join-Path -Path $backupFolderPath -ChildPath $configDirName
         if (-not (Test-Path -Path $appBackupFolder)) {
            New-Item -Path $appBackupFolder -ItemType Directory
            Write-Host "Yedekleme klasörü oluşturuldu: $appBackupFolder"
         }

         # Yedekleme işlemi
         $backupFileName = "$($appBackupFolder)\web.config.$((Get-Date).ToString('yyyyMMdd-HHmmss')).bak"
         Copy-Item -Path $configFile.FullName -Destination $backupFileName
         Write-Host "Yedek alındı: $backupFileName"

         # Application Pool bilgilerini al
         $appPool = Get-Item "IIS:\AppPools\$appPoolName"
         $clrVersion = $appPool.managedRuntimeVersion
         $enable32Bit = $appPool.enable32BitAppOnWin64
         Write-Host "CLR Versiyonu: $clrVersion, 32-bit: $enable32Bit"

         # aspnet_regiis.exe yolunu belirle
         $aspnetRegiisPath = if ($clrVersion -eq "v4.0") {
            if ($enable32Bit -eq $true) {
               "$env:windir\Microsoft.NET\Framework\v4.0.30319\aspnet_regiis.exe"
            } else {
               "$env:windir\Microsoft.NET\Framework64\v4.0.30319\aspnet_regiis.exe"
            }
         } else {
            if ($enable32Bit -eq $true) {
               "$env:windir\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis.exe"
            } else {
               "$env:windir\Microsoft.NET\Framework64\v2.0.50727\aspnet_regiis.exe"
            }
         }
         Write-Host "aspnet_regiis.exe yolu: $aspnetRegiisPath"

         # Şifreleme işlemi
         & $aspnetRegiisPath -pef "connectionStrings" $configFile.DirectoryName
         Write-Host "$($configFile.FullName) dosyasının connectionStrings bölümü şifrelendi."
      }
   } else {
      Write-Host "$($configFile.FullName) dosyasında connectionStrings bölümü bulunamadı."
   }

   Write-Host "İşlem tamamlandı: $($configFile.FullName)"
}
```

**** ad de bulunmayan kullanıcıları listeden log dosyasına yazar disabled olanlar değil

```
$userkontrol = Get-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\liste.txt"
Clear-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt"
Set-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt" "Hesabı Olmayan Kullanıcılar"

ForEach ($sicil in $userkontrol) {
   $user = Get-ADUser -Filter { SamAccountName -eq $sicil } -ErrorAction SilentlyContinue
   if (-not $user) {
      Add-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt" $sicil
   }
}
```

**** ad de bulunmayan kullanıcıları listeden log dosyasına yazar disabled olanlar hariç

```
$userkontrol = Get-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\liste.txt" | ForEach-Object { $_.Trim() }

# Log dosyasını oluştur ve başlık ekle
$logPath = "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt"
if (-not (Test-Path $logPath)) {
   New-Item -Path $logPath -ItemType File -Force
}
Clear-Content $logPath
Set-Content $logPath "Hesabı Olmayan Kullanıcılar"

ForEach ($sicil in $userkontrol) {
   # Kullanıcıyı SamAccountName ile AD'de kontrol et
   $user = Get-ADUser -Filter { SamAccountName -eq $sicil } -ErrorAction SilentlyContinue
   if (-not $user) {
      Add-Content $logPath "$sicil - Kullanıcı bulunamadı."
   }
}
```

**** ad de bulunmayan veya disable kullanıcıları loga yazar

```
$userkontrol = Get-Content "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\liste.txt" | ForEach-Object { $_.Trim() }

# Log dosyasını oluştur ve başlık ekle
$logPath = "C:\Users\usr8121b\Desktop\disable kulalnıcı tespiti\log.txt"
if (-not (Test-Path $logPath)) {
   New-Item -Path $logPath -ItemType File -Force
}
Clear-Content $logPath
Set-Content $logPath "Hesabı Olmayan veya Disable Olan Kullanıcılar"

ForEach ($sicil in $userkontrol) {
   # Kullanıcıyı SamAccountName ile AD'de kontrol et
   $user = Get-ADUser -Filter { SamAccountName -eq $sicil } -Properties Enabled -ErrorAction SilentlyContinue
   if (-not $user) {
      Add-Content $logPath "$sicil - Kullanıcı bulunamadı."
   }
   elseif ($user.Enabled -eq $false) {
      Add-Content $logPath "$sicil - Kullanıcı disable."
   }
}
```

**** taşıma yapar ana klasörleri bırakır 10 günedn eski dosyaları yetkileriyle taşır

```
robocopy "D:\ibomer" "F:\ibomer" *.* /MOV /MINAGE:10 /E /COPY:DATS
```

**** evdenmi ofistenmi çalışıyor

```
$EventData = Get-WinEvent -FilterHashtable @{LogName='Microsoft-Windows-TerminalServices-RemoteConnectionManager/Operational';StartTime=$StartTime;EndTime=$EndTime;ID=1149} | ForEach {
[PScustomObject]@{

      User = $_.Properties[0].Value
Domain = $_.Properties[1].Value
Client = $_.Properties[2].Value
      PSComputerName =  $_.$computer
      TimeCreated = (Get-Date ($_.TimeCreated) -Format 'dd-MM-yyyy hh:mm:ss')

}
}

$IP = $EventData  | select -expandproperty Client
$Nerede =If ($IP -like "*199*" -or "*172*") { Write-Output 'EVDE' } else { Write-Output 'OFISTE'}
$Nerede
```

**** KLASÖRÜYLE DOSYA TAŞIMA

```
robocopy.exe "C:\zinnet1" "C:\zinnet2" *.exe /E /MOV /R:3 /W:10 /MT:16 /Z /NP /NDL /LOG:"C:\zinnet2\LOG.txt"
```

**** KLASÖRÜYLE DOSYA TAŞIMA 2 -mp4leri taşır

```
$nereden = "\\10.242.6.6\dosya\stratejidonusum_gmy\iletisimmarkyon_mud\birimortak\Kurumsaliletisim\usr6345"
$nereye = "\\10.242.6.25\iletisim_marka_mud\usr6345"
```

```
Robocopy $nereden $nereye *.exe *.mp4 /E /MOV /R:3 /W:10 /MT:16 /Z /NP /NDL /LOG:"C:\zinnet2\LOG.txt"
```

**** cmd ile kullanıcıyı oturumdan atma ve yardım gönderme

```
For /f "tokens=3 delims= " %i in ('quser /server:ALBISTRDSSH020 ^| find "usr3167a"') do @echo %i | start mstsc /shadow:%i /v:ALBISTRDSSH020 /control

For /f "tokens=3 delims= " %i in ('quser /server:ALBISTRDSSH120 ^| find "usr3167a"') do @echo %i | Logoff %i /server:ALBISTRDSSH120
```

**** dizindeki mp4 dosyalarını bul ve dosyaya yaz

```
# Dizin yolunu belirleyin
$directoryPath = "D:\file\f\p\MTQ=\M\gm\finansinsankiymetleri_gmy\egitimkariyeryon_mud"

# Arama yaparak .mp4 dosyalarını bulun
$mp4Files = Get-ChildItem -Path $directoryPath -Recurse -Filter *.mp4

# Çıktı dosyasının yolunu belirleyin
$outputFilePath = "C:\zinnet2\mp4FilesList.txt"

# Dosya yollarını .txt dosyasına yazın
$mp4Files.FullName | Out-File -FilePath $outputFilePath

Write-Host "MP4 dosyalarının listesi $outputFilePath dosyasına yazıldı."
```

**** dizindeki x dosyalarını ve boyunutunu bul ve dosyaya yaz

```
# Dizin yolunu belirleyin
$directoryPath = "D:\file\f\p\MTQ=\M\gm\finansinsankiymetleri_gmy\egitimkariyeryon_mud"

# Arama yaparak .mp4 dosyalarını bulun
$mp4Files = Get-ChildItem -Path $directoryPath -Recurse -Filter *.mp4

# Çıktı dosyasının yolunu belirleyin
$outputFilePath = "C:\zinnet2\mp4FilesList3.txt"

# Her bir dosyanın yolunu ve boyutunu yazın
$mp4Files | ForEach-Object {
    $fileInfo = "$($_.FullName) = $($_.Length) bytes"
    $fileInfo | Out-File -FilePath $outputFilePath -Append
}

Write-Host "MP4 dosyalarının listesi ve boyutları $outputFilePath dosyasına yazıldı."
```

**** BKM ve Telekom TLS1.2. güncellemelerinde 2022 öncesi sunucularda POLY1305 active için

```
Enable-TlsCipherSuite -Name ECDHE-ECDSA-CHACHA20-POLY1305-SHA256
Enable-TlsCipherSuite -Name ECDHE-RSA-CHACHA20-POLY1305-SHA256
```

**** RDS de activex-proxy logları temizleme

```
$tdc="C:\Users"
$dirs = gci $tdc -directory -recurse | select -expandproperty FullName
$dirs

  foreach($list in $dirs){
Get-ChildItem -Path $list\AppData\Local\Logs\activex-proxy | ? {$_.LastWriteTime -lt (Get-Date).AddDays(-5) } | Remove-Item -Verbose
}
```

**** klasördeki dosya adları ve içeriklerini txtye yaz

```
# Kaynak ve hedef dosya yolları
$sourceFolder = "C:\Users\usr8121b\Desktop\Sorgular"
$destinationFile = "C:\Users\usr8121b\Desktop\DosyaAdlari.txt"

# .sql dosya uzantılı dosyaların listesini al
$sqlFiles = Get-ChildItem -Path $sourceFolder -Filter "*.sql"

# Hedef dosyayı temizleyerek başlat
Clear-Content -Path $destinationFile

# Dosya numaralandırma için sayaç başlat
$counter = 1

foreach ($file in $sqlFiles) {
    # Dosya adını numaralandırarak yaz
    Add-Content -Path $destinationFile -Value "$counter- Dosya Adı: $($file.Name)"

    # Dosya içeriğini yaz
    $fileContent = Get-Content -Path $file.FullName
    Add-Content -Path $destinationFile -Value $fileContent

    # 3 boş satır ekle
    Add-Content -Path $destinationFile -Value "`n`n`n"

    # Sayaç artır
    $counter++
}
```

Write-Output "Dosya adları ve içerikleri $destinationFile dosyasına yazıldı."

**** Bazı user propertisleri basit

Get-ADUser -Filter {LDbranchNumber -notlike "*993*"} -Properties *

Get-ADUser -Filter {LDtitleCode -eq "0500"} -Properties CN | Select-Object CN

Get-ADUser -Filter {LDtitleCode -eq "1120"} -Properties CN, SamAccountName, LDtitleCode | Select-Object CN, SamAccountName

Get-ADUser -Filter {LDtitleCode -eq "1120" -and Enabled -eq $true} -Properties CN, SamAccountName, LDtitleCode | Select-Object CN, SamAccountName

Get-ADUser -Filter {LDtitleCode -eq "6120" -and Enabled -eq $true} -Properties CN, SamAccountName, LDtitleCode | Select-Object CN, SamAccountName

Get-ADUser -Filter {SamAccountName -eq "usr8121"} -Properties *

Get-ADUser -Filter {LDbranchNumber -like "*993*" -and Enabled -eq $true} -Properties CN, SamAccountName, LDtitleCode | Select-Object CN, SamAccountName
cls
Get-ADUser -Filter {SamAccountName -eq "usr7532"} -Properties LDtitleCode, SamAccountName | Select-Object LDtitleCode, SamAccountName

Get-ADGroup -Filter {Name -eq "Hizmet Masası"} -Properties *
Get-ADGroup -Filter {Name -eq "RDS1"} -Properties *
Get-ADGroup -Filter {Name -eq "ANKARA ŞUBESİ"} -Properties * | Select-Object Name, mail

Get-ADUser -Filter {UserTitle -eq "BİRİM MÜDÜRÜ"} -Properties *

Get-ADUser -Filter {SamAccountName -eq "atg.sevilay.saltan"} -Properties *

Get-ADUser -Filter {SamAccountName -eq "atg.sevilay.saltan"} -Properties UserTitle | Select-Object SamAccountName, UserTitle

---

****