A Beginner’s Guide On Remove User-profiles With Powershell

Created	@August 2, 2021 2:27 PM
Created By	Fardin Barashi
Last Edited By	Fardin Barashi
Last Edited Time	@August 8, 2021 6:50 PM
Property	https://github.com/fardinbarashi
Property 1	https://www.linkedin.com/in/fardin-barashi-a56310a2/
Property 2	Fardin.barashi@gmail.com
Stakeholders	
Status	
Type	

Table of Content

Introduction.

User profiles that have not logged in for several days can be a security issue and take up unnecessary hard disk space. In this how-to article, you are going to learn how to remove user profiles with Powershell.

Prerequisites.

Local.

(Logged in to a computer and execute Powershell)

Prerequisites for Get-WmiObject -Class Win32_UserProfile

.	On Client-Machine
Powershell-Version	Windows PowerShell version 5.1 or higher would work
Privileges	Administrator's privileges
Firewall-Ports	None

Remotely.

(executing Powershell by using PsSession)

Prerequisites for Get-WmiObject -Class Win32_UserProfile

Aa .	On Client-Machine
Powershell-Version	Windows PowerShell version 5.1 or higher would work
Privileges	Administrator's privileges
Firewall-Ports	TCP-Ports: 135, 445, and additional dynamically assigned ports between 1024 to 1034

Prerequisites for PsSession

Aa .	On Client-Machine	Other
Powershell-Version	Windows PowerShell version 5.1 or higher would work	
Privileges	Administrator's privileges or Remote Management Users	
Firewall-Ports	TCP-Ports: 5985 (HTTP) and 5986 (HTTPS)	
NetConnection profile	Domain or Private	If your NetConnection profile is public change it to Private / Domain by using `Set-NetConnectionProfile -NetworkCategory Private` or `Set-NetConnectionProfile -NetworkCategory Domain`
Enabled PsRemoting	Enabled PsRemoting	Run command `Enable-PSRemoting` in powershell as a administrator

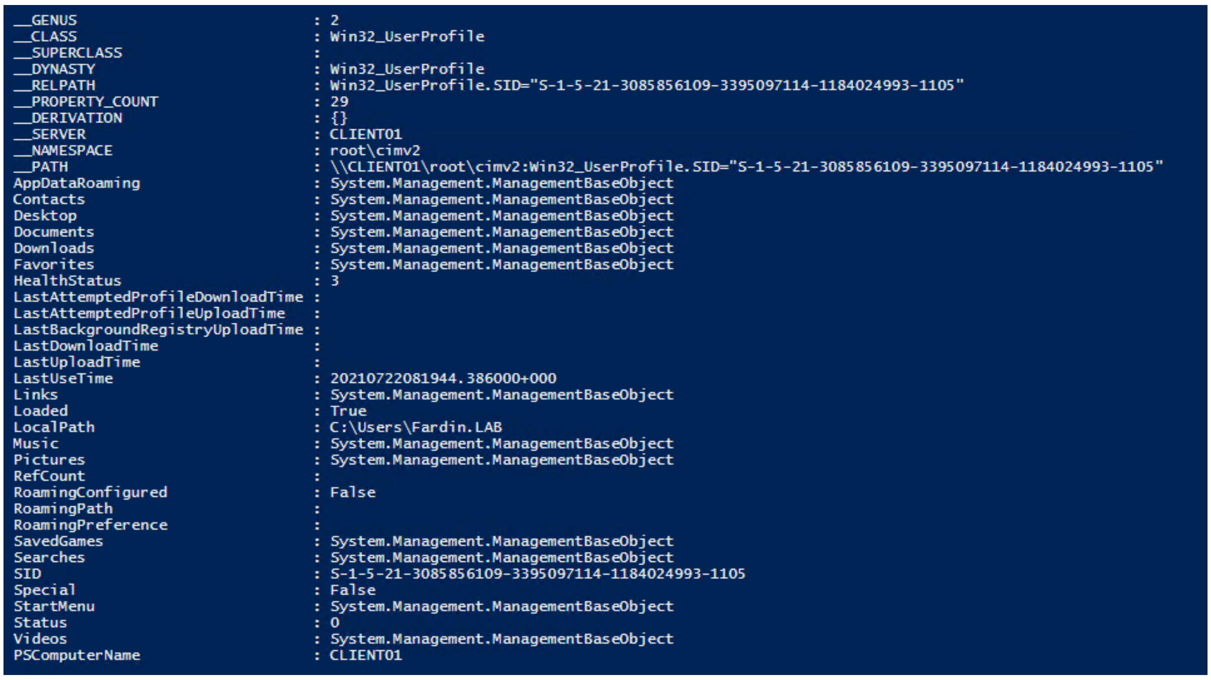
Demo 1.

We are going to log in to a local computer to get all the profiles and then remove one profile.

Get-Profiles.

Open Powershell ISE or Vs-Code with Powershell extension and run the code

```
# Get Profiles
 Get-WmiObject -Class Win32_UserProfile
```

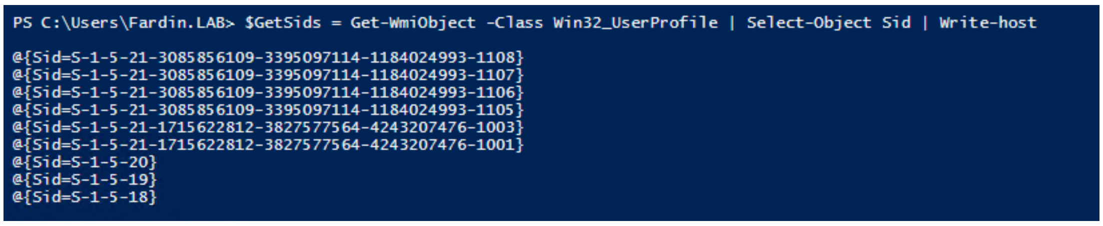


```
__GENUS                          : 2
__CLASS                          : Win32_UserProfile
__SUPERCLASS                     :
__DYNASTY                        : Win32_UserProfile
__RELPATH                        : Win32_UserProfile.SID="S-1-5-21-3085856109-3395097114-1184024993-1105"
__PROPERTY_COUNT                 : 29
__DERIVATION                     : {}
__SERVER                         : CLIENT01
__NAMESPACE                      : root\cimv2
__PATH                           : \\CLIENT01\root\cimv2:Win32_UserProfile.SID="S-1-5-21-3085856109-3395097114-1184024993-1105"
AppDataRoaming                   : System.Management.ManagementBaseObject
Contacts                         : System.Management.ManagementBaseObject
Desktop                          : System.Management.ManagementBaseObject
Documents                        : System.Management.ManagementBaseObject
Downloads                        : System.Management.ManagementBaseObject
Favorites                        : System.Management.ManagementBaseObject
HealthStatus                     : 3
LastAttemptedProfileDownloadTime :
LastAttemptedProfileUploadTime   :
LastBackgroundRegistryUploadTime :
LastDownloadTime                 :
LastUploadTime                   :
LastUseTime                      : 20210722081944.386000+000
Links                            : System.Management.ManagementBaseObject
Loaded                           : True
LocalPath                        : C:\Users\Fardin.LAB
Music                            : System.Management.ManagementBaseObject
Pictures                         : System.Management.ManagementBaseObject
RefCount                         :
RoamingConfigured                : False
RoamingPath                      :
RoamingPreference                :
SavedGames                       : System.Management.ManagementBaseObject
Searches                         : System.Management.ManagementBaseObject
SID                              : S-1-5-21-3085856109-3395097114-1184024993-1105
Special                          : False
StartMenu                        : System.Management.ManagementBaseObject
Status                           : 0
Videos                           : System.Management.ManagementBaseObject
PSComputerName                   : CLIENT01
```

The result of the code above

We really don't need all the data to remove the account, only the SecurityIdentifier (Sid).
For example: Get all the sids in the client-machine

```
#Get Profiles Sid
 $GetSids = Get-WmiObject -Class Win32_UserProfile |
 Select-Object Sid |
 Write-host
```



```
PS C:\Users\Fardin.LAB> $GetSids = Get-WmiObject -Class Win32_UserProfile | Select-Object Sid | Write-host

@{Sid=S-1-5-21-3085856109-3395097114-1184024993-1108}
@{Sid=S-1-5-21-3085856109-3395097114-1184024993-1107}
@{Sid=S-1-5-21-3085856109-3395097114-1184024993-1106}
@{Sid=S-1-5-21-3085856109-3395097114-1184024993-1105}
@{Sid=S-1-5-21-1715622812-3827577564-4243207476-1003}
@{Sid=S-1-5-21-1715622812-3827577564-4243207476-1001}
@{Sid=S-1-5-20}
@{Sid=S-1-5-19}
@{Sid=S-1-5-18}
```

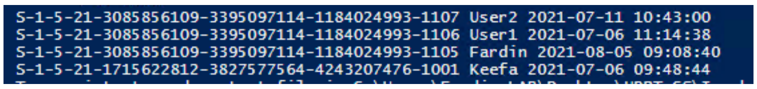
The result of the code above

Translate profiles SecurityIdentifier (Sid).

We are going to translate the Sid's to profiles username and advance the code further.
We are going to

1. Monitor the code with Try and Catch.
2. Log PowerShell with Start-Transcript.
3. Save the log file where we saved our script by using $PsScriptRoot.
4. Skip service accounts by using a filter.
5. Convert LastUseTime to DateTime expression.
6. Translate the Sid's.

```
# Start Logging
 Start-Transcript -Path "$PSScriptRoot\RemoveAccountLog.txt" -Force -Append
 Get-Date -Format "yyyy-mm-dd HH:MM"

 # Get Profiles
  Try
   { # Start Try, Get Sid and translate them
    $GetSids = Get-WmiObject -Class Win32_UserProfile -Filter "Special=False" | Select-Object Sid, @{Label='LastUseTime';Expression={$
    ForEach ($UserSid in $GetSids)
    { # Start ForEach ($UserSid in $GetSids)
     $SID = New-Object System.Security.Principal.SecurityIdentifier($UserSid.sid)
     $UserAccount = $SID.Translate([System.Security.Principal.NTAccount])
     $Profile = $UserAccount.value.split("\")[1];

     Write-Host $SID, $Profile, $UserSid.LastUseTime
    } # End ForEach ($UserSid in $GetSids)
   } # End Try, Get Sid and translate them

 Catch
  {  # Start Catch, Get Sid and translate them
   Write-Warning -Message "## ERROR## "
   Write-Warning -Message "## Script could not start ## "
   Write-Warning $Error[0]
  }  # End Catch, Get Sid and translate them

# End Logging
 Stop-Transcript
```



```
S-1-5-21-3085856109-3395097114-1184024993-1107 User2 2021-07-11 10:43:00
S-1-5-21-3085856109-3395097114-1184024993-1106 User1 2021-07-06 11:14:38
S-1-5-21-3085856109-3395097114-1184024993-1105 Fardin 2021-08-05 09:08:40
S-1-5-21-1715622812-3827577564-4243207476-1001 Keefa 2021-07-06 09:48:44
```

The result of the code above

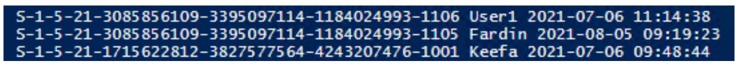
Remove Profile.

To remove profiles from the computer, we will use Remove-WmiObject.

```
# Get Profile, Remove Profile
 Get-WmiObject -Class Win32_UserProfile -Filter "SID = 'TYPE IN THE SID' " |
 Remove-WmiObject
```

For example: I want to Remove the User2 profile and get a visual confirmation in the console

```
# Start Logging
 Start-Transcript -Path "$PSScriptRoot\RemoveAccountLog.txt" -Force -Append
 Get-Date -Format "yyyy-mm-dd HH:MM"

# Remove Profile
 Try
  { # Start Try, Remove Profile
   Get-WmiObject -Class Win32_UserProfile -Filter "SID = 'S-1-5-21-3085856109-3395097114-1184024993-1107' " |
   Remove-WmiObject # Remove Profile
   } # End Try, Remove Profile

 Catch
  {  # Start Catch, Remove Profile
   Write-Warning -Message "## ERROR## "
   Write-Warning -Message "## Script could not remove profile ## "
   Write-Warning $Error[0]
  }  # End Catch, Remove Profile


 # Get Profiles
 Try
  { # Start Try, Get Sid and translate them
   $GetSids = Get-WmiObject -Class Win32_UserProfile -Filter "Special=False" | Select-Object Sid, @{Label='LastUseTime';Expression={$_
    ForEach ($UserSid in $GetSids)
    { # Start ForEach ($UserSid in $GetSids)
     $SID = New-Object System.Security.Principal.SecurityIdentifier($UserSid.sid)
     $UserAccount = $SID.Translate([System.Security.Principal.NTAccount])
     $Profile = $UserAccount.value.split("\")[1];

     Write-Host $SID, $Profile, $UserSid.LastUseTime
   } # End ForEach ($UserSid in $GetSids)
  } # End Try, Get Sid and translate them

Catch
  {  # Start Catch, Get Sid and translate them
   Write-Warning -Message "## ERROR## "
   Write-Warning -Message "## Script could not start ## "
   Write-Warning $Error[0]
  }  # End Catch, Get Sid and translate them

# End Logging
  Stop-Transcript
```



```
S-1-5-21-3085856109-3395097114-1184024993-1106 User1 2021-07-06 11:14:38
S-1-5-21-3085856109-3395097114-1184024993-1105 Fardin 2021-08-05 09:19:23
S-1-5-21-1715622812-3827577564-4243207476-1001 Keefa 2021-07-06 09:48:44
```

The result of the code above

Demo 2.

We are going to create a PowerShell session to a remote computer.
When we have connected our PowerShell to the remote computer, we will get profiles and then remove one profile.
Because we need to know **SID** and **LastUseTime**, we need to have an interaction with the remote computer and therefore we will not use Invoke command.

Configure a computer to receive remote commands.

We need to configure the computers to accept PsSessions.
This can be done with GPO *(Allow Remote Server Management Through WinRM)* or log in on each computer, open PowerShell as admin with cmdlet Enable-PsRemoting.

`Enable-PSRemoting` cmdlet performs the following operations:

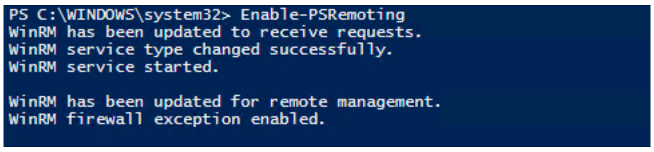
Runs the Set-WSManQuickConfig cmdlet, which performs the following tasks:

- Starts the WinRM service.
- Sets the startup type on the WinRM service to Automatic.
- Creates a listener to accept requests on any IP address.

- Enables a firewall exception for WS-Management communications.
- Creates the simple and long name session endpoint configurations if needed.
- Enables all session configurations.
- Changes the security descriptor of all session configurations to allow remote access.
- Restarts the WinRM service to make the preceding changes effective

For example: Enable Powershell-remoting

```
# Enable PsSession
  Enable-PSRemoting
```



```
PS C:\WINDOWS\system32> Enable-PSRemoting
WinRM has been updated to receive requests.
WinRM service type changed successfully.
WinRM service started.

WinRM has been updated for remote management.
WinRM firewall exception enabled.
```

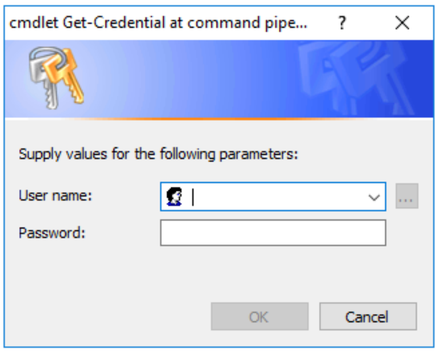
The result of the code above

Connect To Remote-Computer, get user profiles, translate the SID's, Remove User.

To create our PS-connection, we will use New-PsSession. Open Powershell ISE or Vs-Code

For example: Create a New-PsSession to a remote computer

```
# String Credential
 $Credential = (Get-Credential)

# Connect to remote machine
 $PSSession = New-PsSession -ComputerName ComputerName -Credential $Credential |
 Enter-PSSession
```



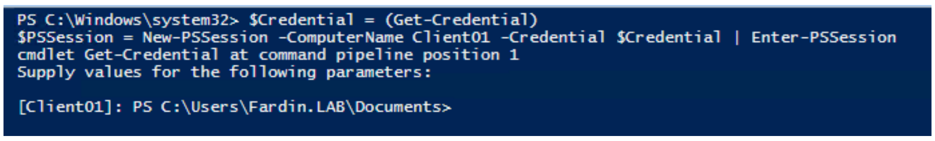
cmdlet Get-Credential at command pipe...

Supply values for the following parameters:

User name:

Password:

Enter Domain\Username and Password



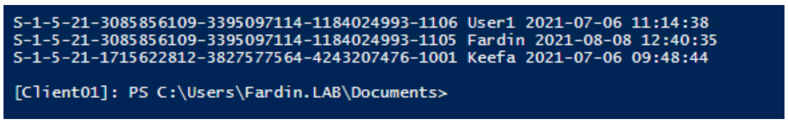
```
PS C:\Windows\system32> $Credential = (Get-Credential)
$PSSession = New-PSSession -ComputerName Client01 -Credential $Credential | Enter-PSSession
cmdlet Get-Credential at command pipeline position 1
Supply values for the following parameters:

[Client01]: PS C:\Users\Fardin.LAB\Documents>
```

The result of the code above

Now that we have created our session, we will get user profiles, translate the SID's and remove one profile just like we did in Demo 1.

For example: Get user profiles, translate the SID's

```
# Get Profiles
 Try
  { # Start Try, Get Sid and translate them
   $GetSids = Get-WmiObject -Class Win32_UserProfile -Filter "Special=False" |
   Select-Object Sid, @{Label='LastUseTime';Expression={$_.ConvertToDateTime($_.LastUseTime)}}

   ForEach ($UserSid in $GetSids)
    { # Start ForEach ($UserSid in $GetSids)
     $SID = New-Object System.Security.Principal.SecurityIdentifier($UserSid.sid)
     $UserAccount = $SID.Translate([System.Security.Principal.NTAccount])
     $Profile = $UserAccount.value.split("\")[1];

     Write-Host $SID, $Profile, $UserSid.LastUseTime
    } # End ForEach ($UserSid in $GetSids)

  } # End Try, Get Sid and translate them

Catch
  {  # Start Catch, Get Sid and translate them
   Write-Warning -Message "## ERROR## "
   Write-Warning -Message "## Script could not start ## "
   Write-Warning $Error[0]
  }  # End Catch, Get Sid and translate them
```

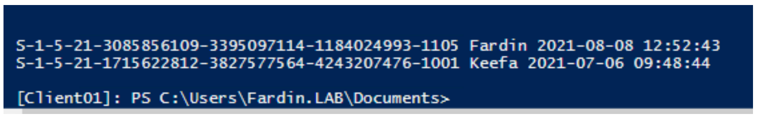


```
S-1-5-21-3085856109-3395097114-1184024993-1106 User1 2021-07-06 11:14:38
S-1-5-21-3085856109-3395097114-1184024993-1105 Fardin 2021-08-08 12:40:35
S-1-5-21-1715622812-3827577564-4243207476-1001 Keefa 2021-07-06 09:48:44

[Client01]: PS C:\Users\Fardin.LAB\Documents>
```

The result of the code above

For example: Remove user1 from the remote machine

```
# Get Profile, Remove Profile
Get-WmiObject -Class Win32_UserProfile -Filter "SID = 'S-1-5-21-3085856109-3395097114-1184024993-1106' " |
Remove-WmiObject # Remove Profile
```



```
S-1-5-21-3085856109-3395097114-1184024993-1105 Fardin 2021-08-08 12:52:43
S-1-5-21-1715622812-3827577564-4243207476-1001 Keefa 2021-07-06 09:48:44

[Client01]: PS C:\Users\Fardin.LAB\Documents>
```

The result of the code above

Conclusion.

We have now removed a user in two different ways.

The journey does not end here, we must advance.
The next step is to create a GUI for the helpdesk department.

Github Links.

Demo 1.

Demo 2.