



Sign in

lkys37en / Start-ADEnum Public

Notifications

Fork 19

Star 88

[Code](#) [Issues 2](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#)

Start-ADEnum / Functions / Start-ADEnum.ps1



845 lines (715 loc) · 48.1 KB

Code Blame

Raw



```
1  Function Start-ADEnum {
2      <#
3      .SYNOPSIS
4      Author: @lkys37en
5      Required Dependencies: Powershell Version 5.x, Windows 10 1803 or 1809
6      Optional Dependencies: None
7
8      .DESCRIPTION
9      This tool is used to automate Active Directory enumeration. The tool requires a domain context
10
11     .PARAMETER ClientName
12     Enter clientname for folder structure.
13
14     .PARAMETER Path
15     Enter path where evidence will be placed. If folder doesn't already exist, the script will create it.
16
17     .PARAMETER Domains
18     Enter individual domains to enumerate or let the script automatically identify all available domains.
19
20     .PARAMETER Scan
21     Enter individual scan(s) to perform.
22
23     .EXAMPLE
24     PS C:\> Start-ADEnum -ClientName lkylabs -Path C:\Projects -Scan All
25     Collects a list of all domain/forest trusts and runs all scans against each domain found.
26
```

```

27 .EXAMPLE
28 PS C:\> Start-ADenum -ClientName lkylabs -Path C:\Projects -Domain lkylabs.com -Scan All
29 Runs all scans against a single domain.
30
31 .EXAMPLE
32 PS C:\> Start-ADenum -ClientName lkylabs -Path C:\Projects -Domain lkylabs.com,corp.lkylabs.co
33 Runs PowerView and Bloodhound scans against domains lkylabs.com and corp.lkylabs.com.
34
35 #>
36 [CmdletBinding()]
37 Param (
38     [Parameter(Mandatory = $True)]
39     [ValidateNotNullOrEmpty()]
40     [String]
41     $ClientName,
42
43     [Parameter(Mandatory = $True)]
44     [ValidateNotNullOrEmpty()]
45     [String]
46     $Path,
47
48     [Parameter(Mandatory = $False)]
49     [String[]]
50     $Domains,
51
52     [Parameter(Mandatory = $True)]
53     [ValidateSet("ADCS", "Bloodhound", "GPOReport", "PowerView", "PingCastle", "PowerUPSQL", "P
54     [String[]]
55     $Scan,
56
57     [Parameter(Mandatory = $False)]
58     [String]
59     $UserName
60 )
61
62 Begin {
63     #Folders variable
64     $Folders = @(
65         "PingCastle"
66         "PowerView"
67         "Bloodhound"
68         "GPO"
69         "Microsoft Services\Exchange"
70         "Microsoft Services\ADCS"
71         "PowerUPSQL"
72         "QuickScan"

```

```
73         )
74
75         #Installs all prereqs if missing
76         Write-Host -ForegroundColor Green "[*] Performing prereqs check"
77         Start-PrereqCheck
78
79         Import-Module C:\Tools\PowerSploit\Recon\PowerView.ps1
80
81         #Checking Domain Context before moving forward
82         try {
83             [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain() | Out-Null
84         }
85
86         catch {
87             throw "[*] Not currently associated with a domain account, perform runas /netonly before
88         }
89
90         #Creating Path and evidence folder structure
91         if ((Test-Path $Path) -eq $false) {
92             try {
93                 Write-Host -ForegroundColor Green "[+] Creating $Path Folder "
94                 mkdir -Path $Path | Out-Null
95             }
96
97             catch {
98                 throw "An error has occurred $($_.Exception.Message)"
99             }
100
101
102             try {
103                 Write-Host -ForegroundColor Green "[+] Creating $ClientName Folder"
104                 mkdir -Path $Path\$ClientName | Out-Null
105             }
106
107             catch {
108                 throw "An error has occurred $($_.Exception.Message)"
109             }
110
111             try {
112                 if (!$Domains) {
113                     Write-Host -ForegroundColor Green "[*] Collecting a list of domains.."
114                     $Domains = (Get-DomainTrustMapping -API).TargetName | Select-Object -Unique
115                     if ($Domains -eq $null) { $Domains = (Get-NetDomain).Name }
116                 }
117
118                 foreach ($Domain in $Domains) {
```

```
110         foreach ($domain in $domains) {
```































```
772     "All" {
773         foreach ($Domain in $Domains) {
774             Write-Host -ForegroundColor Green "[+] Starting All AD Enum for $Domain"
775             Start-Job -ScriptBlock $PowerViewScriptBlock -ArgumentList $ClientName, $Path,
776             Start-Job -ScriptBlock $PowerUPSQLScriptBlock -ArgumentList $ClientName, $Path,
777             Start-Job -ScriptBlock $PingCastleScriptBlock -ArgumentList $ClientName, $Path,
778             Start-Job -ScriptBlock $BloodhoundScriptBlock -ArgumentList $ClientName, $Path,
779             Start-Job -ScriptBlock $GPOReportScriptBlock -ArgumentList $ClientName, $Path,
780             Start-Job -ScriptBlock $PrivExchangeCheck -ArgumentList $ClientName, $Path, $Domain
781             Start-Job -ScriptBlock $ADCS -ArgumentList $ClientName, $Path, $Domain -Name AD
782         }
783     }
784
785     "ADCS" {
786         foreach ($Domain in $Domains) {
787             Write-Host -ForegroundColor Green "[+] Starting AD Certificate Services Enum for $Domain"
788             Start-Job -ScriptBlock $ADCS -ArgumentList $ClientName, $Path, $Domain -Name AD
789         }
790     }
791
792     "PowerView" {
793         foreach ($Domain in $Domains) {
794             Write-Host -ForegroundColor Green "[+] Starting PowerView Enum for $Domain"
795             Start-Job -ScriptBlock $PowerViewScriptBlock -ArgumentList $ClientName, $Path,
796         }
797     }
798
799     "PowerUPSQL" {
800         foreach ($Domain in $Domains) {
801             Write-Host -ForegroundColor Green "[+] Starting PowerUPSQL Enum for $Domain"
802             Start-Job -ScriptBlock $PowerUPSQLScriptBlock -ArgumentList $ClientName, $Path,
803         }
804     }
805 }
```

```
806         "PingCastle" {
807             foreach ($Domain in $Domains) {
808                 Write-Host -ForegroundColor Green "[+] Starting Ping Castle AD Enum for $Domain"
809                 Start-Job -ScriptBlock $PingCastleScriptBlock -ArgumentList $ClientName, $Path, $Domain
810             }
811         }
812
813         "Bloodhound" {
814             foreach ($Domain in $Domains) {
815                 Write-Host -ForegroundColor Green "[+] Starting Bloodhound AD Enum for $Domain"
816                 Start-Job -ScriptBlock $BloodhoundScriptBlock -ArgumentList $ClientName, $Path, $Domain
817             }
818         }
819
820         "GPOReport" {
821             foreach ($Domain in $Domains) {
822                 Write-Host -ForegroundColor Green "[+] Starting GPO Report AD Enum for $Domain"
823                 Start-Job -ScriptBlock $GPOReportScriptBlock -ArgumentList $ClientName, $Path, $Domain
824             }
825         }
826
827         "PrivExchange" {
828             foreach ($Domain in $Domains) {
829                 Write-Host -ForegroundColor Green "[+] Starting PrivExchange AD Enum for $Domain"
830                 Start-Job -ScriptBlock $PrivExchangeCheck -ArgumentList $ClientName, $Path, $Domain
831             }
832         }
833
834         "QuickScan" {
835             foreach ($Domain in $Domains) {
836                 Write-Host -ForegroundColor Green "[+] Starting QuickScan PingCastle AD Enum for $Domain"
837                 Start-Job -ScriptBlock $QuickScanPingCastle -ArgumentList $ClientName, $Path, $Domain
838             }
839             Write-Host -ForegroundColor Green "[+] Starting QuickScan PowerUPSQL AD Enum for $Domain"
840             Start-Job -ScriptBlock $QuickScanMSSQL -ArgumentList $ClientName, $Path, $Domain
841         }
842     }
843 }
844 }
845 }
```