Product Solutions Resources Open Source Enterprise Pricing Sign in Sign up

This repository has been archived by the owner on Sep 2, 2022. It is now read-only.

BloodHoundAD / SharpHound3 Public archive

Notifications Fork 156 Star 520

Code Issues 7 Pull requests 9 Actions Projects Security Insights

Files

7d96b99

Go to file

- SharpHound3
  - Enums
  - JSON
  - LdapWrappers
  - PowerShell Output
  - Producers
  - Properties
  - Tasks
  - App.config
  - Cache.cs
  - CommonPrincipal.cs
  - DirectorySearch.cs
  - Exceptions.cs
  - Extensions.cs
  - FodyWeavers.xml
  - FodyWeavers.xsd
  - Helpers.cs
  - LdapBuilder.cs
  - LdapTypeEnum.cs
  - Options.cs
  - ResolutionHelpers.cs
  - SharpHound.cs
  - SharpHound3.csproj
  - UserDomainKey.cs
  - favicon.ico
  - packages.config
- .gitignore
- LICENSE
- README.md
- SharpHound3.sln

SharpHound3 / SharpHound3 / LdapBuilder.cs

rvazarkar Add missing ldap property 37a3bf5 · 4 years ago History

Code Blame 136 lines (120 loc) · 6.41 KB Raw

```
using System.Collections.Generic;
using System.Linq;
using SharpHound3.Enums;

namespace SharpHound3
{
    internal class LdapBuilder
    {
        /// <summary>
        /// Builds the necessary attributes and ldap query for the specified set of opt
        /// </summary>
        /// <param name="methods"></param>
        /// <returns></returns>
        internal static LdapQueryData BuildLdapQuery(CollectionMethodResolved methods)
        {
            var ldapFilterParts = new List<string>();
            var ldapProperties = new List<string>();

            //We always want these properties to ensure we can at least pass type findi
            ldapProperties.AddRange(Helpers.ResolutionProps);
            ldapProperties.Add("samaccountname");
            //Add this property to check for GMSAs
            ldapProperties.Add("msds-groupmsamembership");
            //LAPS is weird and several collection methods depend on it, but its easier
            ldapProperties.Add("ms-mcs-admpwdexpirationtime");

            //Add the operatingsystem property for WindowsOnly so we can pre-filter hos
            if (Options.Instance.WindowsOnly)
                ldapProperties.Add("operatingsystem");

            //Group membership collection
            if (methods.HasFlag(CollectionMethodResolved.Group))
            {
                ldapFilterParts.Add("(|(samaccounttype=268435456)(samaccounttype=268435
                ldapProperties.AddRange(new[] { "member", "primarygroupid" });
            }

            //Computer collection methods: ask for non-disabled computer objects
            if (methods.HasFlag(CollectionMethodResolved.LocalAdmin) ||
                methods.HasFlag(CollectionMethodResolved.Sessions) ||
                methods.HasFlag(CollectionMethodResolved.LoggedOn) || methods.HasFlag(C
                methods.HasFlag(CollectionMethodResolved.DCOM) || methods.HasFlag(Colle
            {
                ldapFilterParts.Add("(&(sAMAccountType=805306369)(!(UserAccountControl:
            }

            //ACL Collection
            if (methods.HasFlag(CollectionMethodResolved.ACL))
            {
                ldapFilterParts.Add("(|(samAccountType=805306368)(samAccountType=805306
                ldapProperties.AddRange(new[]
                {
                    "ntsecuritydescriptor", "displayname", "name"
                });
            }
            }

            //Trust enumeration
            if (methods.HasFlag(CollectionMethodResolved.Trusts))
            {
                ldapFilterParts.Add("(objectclass=domain)");
            }

            //Object Properties
            if (methods.HasFlag(CollectionMethodResolved.ObjectProps))
            {
                ldapFilterParts.Add("(|(samaccounttype=268435456)(samaccounttype=268435
                ldapProperties.AddRange(new[]
                {
                    "pwdlastset", "lastlogon", "lastlogontimestamp",
                    "sidhistory", "useraccountcontrol", "operatingsystem",
                    "operatingsystemservicepack", "serviceprincipalname", "displayname"
                    "homedirectory","description","admincount","userpassword","gpcfiles
                    "msds-behavior-version","objectguid", "name", "gpoptions", "msds-al
                    "sidhistory"
                });
            }

            //Container enumeration
            if (methods.HasFlag(CollectionMethodResolved.Container))
            {
                ldapFilterParts.Add("(|(&(&(objectcategory=groupPolicyContainer)(flags=
                ldapProperties.AddRange(new[] { "gplink", "gpoptions", "name", "display
            }

            //GPO Local group enumeration
            if (methods.HasFlag(CollectionMethodResolved.GPOLocalGroup))
            {
                //ldapFilterParts.Add("(&(&(objectcategory=groupPolicyContainer)(flags=
                //ldapProperties.AddRange(new[] {"gpcfilesyspath", "displayname"});
                ldapFilterParts.Add("(&(|(objectcategory=organizationalUnit)(objectclas
                ldapProperties.AddRange(new[] { "gplink", "name" });
            }

            //SPN Target Enumeration
            if (methods.HasFlag(CollectionMethodResolved.SPNTargets))
            {
                ldapFilterParts.Add("(&(samaccounttype=805306368)(serviceprincipalname=
                ldapProperties.AddRange(new[]
                {
                    "serviceprincipalname"
                });
            }

            //Take our query parts, and join them together
            var finalFilter = string.Join("", ldapFilterParts.ToArray());
            //Surround the filters with (|), which will OR them together
            finalFilter = ldapFilterParts.Count == 1 ? ldapFilterParts[0] : $"(|{finalF

            //Add the user specified filter if it exists
            var userFilter = Options.Instance.LdapFilter;
            if (userFilter != null)
            {
                finalFilter = $"(&({finalFilter})({userFilter}))";
            }

            if (Options.Instance.CollectAllProperties)
            {
                ldapProperties = new List<string>();
                ldapProperties.Add("*");
            }

            //Distinct the attributes
            return new LdapQueryData
            {
                LdapFilter = finalFilter,
                LdapProperties = ldapProperties.Distinct().ToArray()
            };
        }
    }

    internal class LdapQueryData
    {
        public string LdapFilter { get; set; }
        public string[] LdapProperties { get; set; }
    }
}
```