Offensive WMI - Active Directory Enumeration (Part 5)

0xinfection.github.io/posts/wmi-ad-enum

October 17, 2021

2021-10-17

This blog is the fifth installation of the “Offensive WMI” series that I’ve been writing on, and this post will cover Active Directory enumeration.

Active Directory (AD) is Microsoft’s implementation of a directory and IAM service for Windows domain networks – which enables admins to manage permissions and access to resources. Anything used for managing multiple resources is handy for administrators, however, the same is also useful for evil-doers in gathering information and lateral movement.

It is important to note that there are several ways to interact with any AD environment, but for this blog we’ll stick with pure WMI.

AD and WMI

WMI has a provider called `root\directory\ldap` which can be used for interaction with the active directory environment. If we try to list the classes under that provider, we can see that there are a lot of classes that come with either of these prefixes, viz. `ads_` or `ds_`. The classes starting with `ads_` are abstract, and the ones with `ds_` are dynamic (the only ones useful to us since they allow retrieval of instances).

We’ll quickly list out the available classes using the following:

```
Get-WmiObject -Namespace root\directory\ldap -Class ds_* -List
```

```
PS C:\Users\pew> Get-WmiObject -Namespace root\directory\ldap -Class ds_* -List


   NameSpace: ROOT\directory\ldap

Name                                Methods             Properties
----                                -------             ----------
DS_LDAP_Root_Class                  {}                  {ADSIPath}
DS_LDAP_Instance_Containment        {}                  {ChildInstance, ParentInstance}
DS_LDAP_Class_Containment           {}                  {ChildClass, ParentClass}
ds_top                              {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_applicationsettings              {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_applicationsitesettings          {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_domain                           {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_domainrelatedobject              {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_dynamicobject                    {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_ipsecbase                        {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_leaf                             {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_mailrecipient                    {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_samdomain                        {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_samdomainbase                    {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_securityobject                   {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_securityprincipal                {}                  {ADSIPath, DS_accountNameHistory, DS_adminDescription, DS_adminDisplayNa...
ds_simplesecurityobject             {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_posixaccount                     {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_shadowaccount                    {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_posixgroup                       {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_iphost                           {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_ieee802device                    {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_bootabledevice                   {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_msds_claimtypepropertybase       {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
ds_msds_cloudextensions             {}                  {ADSIPath, DS_adminDescription, DS_adminDisplayName, DS_allowedAttribute...
```

Let's get started!

Finding the domain name

After gaining access to a box on a domain, one of the first steps in basic reconnaissance would be to try to figure out the domain name on which we are on:

```
Get-WmiObject -Namespace root\directory\ldap -Class ds_domain | select ds_dc,
    ds_distinguishedname, pscomputername
```

```
PS C:\> Get-WmiObject -Namespace root\directory\ldap -Class ds_domain | select ds_dc, ds_distinguishedname,
 pscomputername

ds_dc    ds_distinguishedname PSComputerName
-----    -------------------- --------------
infected DC=infected,DC=local PC01
```

In the above command, we fetched the domain name, the FQDN, and the name of the computer we're currently on.

Getting the domain policy

Now, let's try to look at the current domain policy. The same class `ds_domain` can help us out with additional information about the policy:

```
Get-WmiObject -Namespace root\directory\ldap -Class ds_domain | select
ds_lockoutduration,
    ds_lockoutobservationwindow, ds_lockoutthreshold, ds_maxpwdage,
    ds_minpwdage, ds_minpwdlength, ds_pwdhistorylength, ds_pwdproperties
```

```
PS C:\> Get-WmiObject -Namespace root\directory\ldap -Class ds_domain | select ds_lockoutduration, ds_lockoutobservationwindow,
ds_lockoutthreshold, ds_maxpwdage, ds_minpwdage, ds_minpwdlength, ds_pwdhistorylength, ds_pwdproperties


ds_lockoutduration          : -18000000000
ds_lockoutobservationwindow : -18000000000
ds_lockoutthreshold         : 0
ds_maxpwdage                : -36288000000000
ds_minpwdage                : -864000000000
ds_minpwdlength             : 7
ds_pwdhistorylength         : 24
ds_pwdproperties            : 1
```

> **NOTE:** As you might have noticed, the output of the above command includes a lot of negative values. All the timestamps in the above output are stored as **negative "filetimes"**, i.e. represented as negative integers of 100 nanosecond timeslices. For example, a time period of 20 minutes would be represented as `-12000000000`.

We can fairly derive that both the lockout duration and lockout observation window is set at 30 mins, while the password expiry duration is infinite, i.e. never expires. The minimum password length is 7, while, the password history length is 24.

Finding the domain controller

Cool, let us try to find out the domain controller. Before that, we should keep in mind that in WMI, different UAC (User Account Control) values are defined via constants. The table below presents a mapping of the UAC values to user types:

User Type	Hex Value	Constants
Normal User	0x200	512
Workstation/Server	0x1000	4096
Domain Controller	0x82000	532480

Now that we know the values, we can easily query the `ds_computer` class with the following command. Note the usage of the `where` Powershell utility to find the domain controller via the piped variable.

```
Get-WmiObject -Namespace root\directory\ldap -Class ds_computer | where {
    $_.ds_useraccountcontrol -match 532480
} | select ds_cn, ds_dnshostname, ds_operatingsystem, ds_lastlogon, ds_pwdlastset
```

```
PS C:\Users\pew> Get-WmiObject -Namespace root\directory\ldap -Class ds_computer | where {$_.ds_useraccountcontrol -match 532480}
 | select ds_cn, ds_dnshostname, ds_operatingsystem, ds_lastlogon, ds_pwdlastset


ds_cn             : DC01
ds_dnshostname    : DC01.infected.local
ds_operatingsystem : Windows Server 2012 R2 Datacenter Evaluation
ds_lastlogon      : 132789867371465153
ds_pwdlastset     : 132764969799000710
```

The output gives us a lot of data about the domain controller (DC), including its name, DNS hostname, OS, last logon timestamp and even the DC's last password update timestamp. All time-based properties in the output are filetimes. You can use this neat

converter to convert filetimes into human-readable datetime format.

That's a lot of information that we can enumerate with just an ordinary domain user account without admin privileges!

Searching user accounts

How about user accounts in the domain as well as other domains in trust relationship with the current domain? It is as simple as querying users using the `Win32_UserAccount` class:

```
Get-WmiObject -Class win32_useraccount | select name, domain, accounttype
```

```
PS C:\Users\pew> Get-WmiObject -Class win32_useraccount | select name, domain, accounttype

name               domain   accounttype
----               ------   -----------
Administrator      PC01             512
DefaultAccount     PC01             512
Guest              PC01             512
User               PC01             512
WDAGUtilityAccount PC01             512
Administrator      INFECTED         512
Guest              INFECTED         512
krbtgt             INFECTED         512
pew                INFECTED         512
wep                INFECTED         512
```

As you might have already guessed, the `AccountType` property is a constant value depicting the user type. The table below defines the type of accounts with their constant values:

Account Type	Identifier	Constant
Temporary Duplicate Account	`UF_TEMP_DUPLICATE_ACCOUNT`	256
Normal Account	`UF_NORMAL_ACCOUNT`	512
Interdomain Trust Account	`UF_INTERDOMAIN_TRUST_ACCOUNT`	2048
Workstation Trust Account	`UF_WORKSTATION_TRUST_ACCOUNT`	4096
Server Trust Account	`UF_SERVER_TRUST_ACCOUNT`	8192

In case we want to filter out just the user accounts for a single domain, we can use the `-Filter` switch of the cmdlet like this:

```
Get-WmiObject -Class win32_useraccount -Filter 'domain="infected"' | select
caption
```

```
PS C:\Users\pew> Get-WmiObject -Class win32_useraccount -Filter 'domain="infected"' | select caption

caption
-------
INFECTED\Administrator
INFECTED\Guest
INFECTED\krbtgt
INFECTED\pew
INFECTED\wep
```

Enumerating currently logged-on users

The `Win32_LoggedOnUser` classes provide information about the logged-on users in that system as well as the domain. To filter out logged-on local users, we can use the following command:

```
Get-WmiObject -Class win32_loggedonuser | where {
    $_ -match 'infected'
} | foreach {[wmi]$_.antecedent}
```

```
PS C:\> Get-WmiObject -Class win32_loggedonuser | where {$_ -match 'infected'} | foreach {[wmi]$_.antecedent}


AccountType : 512
Caption     : INFECTED\pew
Domain      : INFECTED
SID         : S-1-5-21-2553750175-4195942334-2808156689-1107
FullName    : Pew User
Name        : pew

AccountType : 512
Caption     : INFECTED\administrator
Domain      : INFECTED
SID         : S-1-5-21-2553750175-4195942334-2808156689-500
FullName    :
Name        : administrator
```

Fetching groups

Fetching the groups for a domain is simple as querying the `Win32_GroupInDomain` class. We'll use a bit of Powershell magic to give us a cleaner output.

```
Get-WmiObject -Class win32_groupindomain | foreach {[wmi]$_.partcomponent}
```

```
PS C:\Users\pew> Get-WmiObject -Class win32_groupindomain | ForEach-Object {[wmi]$_.PartComponent}

Caption                                      Domain   Name                                         SID
-------                                      ------   ----                                         ---
INFECTED\Cert Publishers                     INFECTED Cert Publishers                              S-1-5-21-2553750175-4195942...
INFECTED\RAS and IAS Servers                 INFECTED RAS and IAS Servers                          S-1-5-21-2553750175-4195942...
INFECTED\Allowed RODC Password Replication Group INFECTED Allowed RODC Password Replication Group S-1-5-21-2553750175-4195942...
INFECTED\Denied RODC Password Replication Group  INFECTED Denied RODC Password Replication Group  S-1-5-21-2553750175-4195942...
INFECTED\WinRMRemoteWMIUsers__               INFECTED WinRMRemoteWMIUsers__                        S-1-5-21-2553750175-4195942...
INFECTED\DnsAdmins                           INFECTED DnsAdmins                                    S-1-5-21-2553750175-4195942...
INFECTED\Cloneable Domain Controllers        INFECTED Cloneable Domain Controllers                 S-1-5-21-2553750175-4195942...
INFECTED\DnsUpdateProxy                      INFECTED DnsUpdateProxy                               S-1-5-21-2553750175-4195942...
INFECTED\Domain Admins                       INFECTED Domain Admins                                S-1-5-21-2553750175-4195942...
INFECTED\Domain Computers                    INFECTED Domain Computers                             S-1-5-21-2553750175-4195942...
INFECTED\Domain Controllers                  INFECTED Domain Controllers                           S-1-5-21-2553750175-4195942...
INFECTED\Domain Guests                       INFECTED Domain Guests                                S-1-5-21-2553750175-4195942...
INFECTED\Domain Users                        INFECTED Domain Users                                 S-1-5-21-2553750175-4195942...
INFECTED\Enterprise Admins                   INFECTED Enterprise Admins                            S-1-5-21-2553750175-4195942...
INFECTED\Enterprise Read-only Domain Controllers INFECTED Enterprise Read-only Domain Controllers S-1-5-21-2553750175-4195942...
INFECTED\Group Policy Creator Owners         INFECTED Group Policy Creator Owners                  S-1-5-21-2553750175-4195942...
INFECTED\Protected Users                     INFECTED Protected Users                              S-1-5-21-2553750175-4195942...
INFECTED\Read-only Domain Controllers        INFECTED Read-only Domain Controllers                 S-1-5-21-2553750175-4195942...
INFECTED\Schema Admins                       INFECTED Schema Admins                                S-1-5-21-2553750175-4195942...
```

The same could be done with the `Win32_Group` class, but the output would include the local groups as well.

Figuring out group memberships

Group membership data is provided by the `Win32_GroupUser` class. The output obtained includes all membership information for the current domain, the trusted domains as well as the trusted forest with bi-directional trust. For our example, let's say we want to fetch the accounts from the "Domain Admins" group:

```
Get-WmiObject -Class win32_groupuser | where {
    $_.groupcomponent -match 'domain admins'
} | foreach {[wmi]$_.partcomponent}
```

```
PS C:\Users\pew> Get-WmiObject -Class win32_groupuser | where {$_.groupcomponent -match 'domain admins'} | foreach {[wmi]$_.partc
omponent}


AccountType : 512
Caption     : INFECTED\Administrator
Domain      : INFECTED
SID         : S-1-5-21-2553750175-4195942334-2808156689-500
FullName    :
Name        : Administrator
```

This is equally applicable for the reverse use case. If we want to enumerate the groups that a particular user is in (`Administrator` in this case), then we can do something like:

```
Get-WmiObject -Class win32_groupuser | where {
    $_.partcomponent -match 'Administrator'
} | foreach {[wmi]$_.groupcomponent}
```

```
PS C:\Users\pew> Get-WmiObject -Class win32_groupuser | where {$_.partcomponent -match 'administrator'} | foreach {[wmi]$_.groupcomponent}

Caption                            Domain   Name                       SID
-------                            ------   ----                       ---
PC01\Administrators                PC01     Administrators             S-1-5-32-544
INFECTED\Domain Admins             INFECTED Domain Admins              S-1-5-21-2553750175-4195942334-2808156689-512
INFECTED\Domain Users              INFECTED Domain Users               S-1-5-21-2553750175-4195942334-2808156689-513
INFECTED\Enterprise Admins         INFECTED Enterprise Admins          S-1-5-21-2553750175-4195942334-2808156689-519
INFECTED\Group Policy Creator Owners INFECTED Group Policy Creator Owners S-1-5-21-2553750175-4195942334-2808156689-520
INFECTED\Schema Admins             INFECTED Schema Admins              S-1-5-21-2553750175-4195942334-2808156689-518
```

> **NOTE**: Since we're using Powershell's `Where-Object` cmdlet, we can use the `-match` argument efficiently to find any substring within a property. This helps in exploring a lot of possibilities in creatively filtering out things on specific criteria.

Finding machines in the domain

To get a list of all unique machines in our active directory environment, we can do something like:

```
Get-WmiObject -Namespace root\directory\ldap -Class ds_computer | select ds_cn
```

```
PS C:\Users\pew> Get-WmiObject -Namespace root\directory\ldap -Class ds_computer | select ds_cn

ds_cn
-----
DC01
PC01
PC02
```

Enumerating admin privileges across AD

An important thing to remember is, by default WMI provides remote access only to local administrators. Already noticed the catch there? If we can run WMI commands against a remote computer, it means we have *local administrator* access on that computer. We can use this information to write a very simple script that can enumerate local administrator privileges on remote computers.

We can chain the output of the last command with a bit of Powershell to achieve the above:

```
$pcs = Get-WmiObject -Namespace root\directory\ldap -Class ds_computer | select -ExpandProperty ds_cn
foreach ($pc in $pcs) {
    (Get-WmiObject -Class win32_computersystem -ComputerName $pc -ErrorAction silentlycontinue).name
}
```

```
PS C:\> $pcs = Get-WmiObject -Namespace root\directory\ldap -Class ds_computer | select -ExpandProperty ds_cn
PS C:\> $pcs
DC01
PC01
PC02
PS C:\> foreach ($pc in $pcs) { Get-WmiObject -Class win32_computersystem -ComputerName $pc -ErrorAction silentlycontinue | select name }

name
----
DC01
PC01
```

Now we have all the boxes on which our user has admin privileges. With this information, things now become super easy for us in lateral movement.

Conclusion

We saw how easily we could enumerate a lot of stuff from an Active Directory environment using a few WMI classes and some Powershell magic. Once again, this blog is not comprehensive and there are a lot of possibilities when it comes to reconnaissance. In our next blog, we'll focus on lateral movement via WMI.

That's it for now folks. I hope you enjoyed reading the blog. Cheers! 🥂