Faster Domain Escalation using LDAP

If you're a penetration tester, then you probably already know that escalating from a local administrator to a Domain Admin only requires a few steps. Those steps typically involve stealing Domain Admin passwords, password hashes, or authentication tokens via various methods. However, if you aren't lucky enough to have a Domain Admin logged into the system you just compromised, then you'll have to go looking for one that does. A while back I wrote a short blog called "5 Ways to Find Systems Running Domain Admin Processes" that outlines a few common approaches, but recently I found another one. So in this blog I'll cover how to find systems where Domain Admins are likely to be logged in by querying the "ServicePrincipleName" attribute of accounts in LDAP. I've also written a little PowerShell module to automate the process. Hopefully it will be useful to penetration testers and admins interested in knowing where their domain accounts are running services on the domain.

LDAP Overview

For those who are not familiar with the Lightwieght Directory Access Protocol (LDAP), it is exactly what it sounds like – a directory of information. Although LDAP is used across many platforms, in Windows domain environments it lives at the heart of Active Directory Services (ADS). ADS performs authentication and authorization for Windows Domains, but also houses a ton of information. That information includes, but is not limited to domain accounts, computer accounts, domain groups, security policies, and software updates. Each object has multiple attributes associated with it, and almost all of them can be queried via LDAP. For example, every user account has a "Created" attribute that indicates when the account was created. Similarly every account has a "ServicePrincipleName" attribute which will be the focus of the rest of this blog.

ServicePrincipleName Overview

Microsoft's documentation states, "*A service principal name (SPN) is the name by which a client uniquely identifies an instance of a service*." Based some light reading it sounds like the official use is to facilitate Kerberos authentication on Windows domains. However, we are going to use it for something else. For our purposes, the multi-value ServicePrincpleName attribute is handy, because each user and computer object in Active Directory stores all of the services that the account runs on the domain. So it's it easy to locate common servers like IIS, SQL Server, and LDAP. It's also handy because it can tell us where accounts are configured to be running or logged in (like Domain Admins). This is relatively easy, because SPNs have a standardized naming convention. SPNs are usually formatted as SERVICE/HOST, but sometimes they also include a port like SERVICE/HOST:PORT.

For example, if a domain account was used to run DNS and SQL Server services on the acme.com domain then the SPN entries would look something like the following:

```
DNS/Server1.acme.com MSSQLSvc/Server2.acme.com:1433
```

Querying LDAP for basic SPN information is pretty straight forward. For example, adfind.exe

(www.joeware.net) can be used to list all of the SQL Server instances registered on a domain with the following command as an authenticated user:

```
C: >Adfind.exe -f "ServicePrincipalName=MSSQLSvc*"
```

Also, the “setspn.exe” tool that comes with Windows Server 2008 can be used to quickly lookup the SPNs for a specific account.

```
C: >setspn.exe –l user1
```

I’ve found during penetration tests that most enterprise environments have Domain Admins accounts that are used to run services on the domain. As result, it is possible to simply query LDAP for all of the Domain Admins and review their SPN entries for servers where they are likely to be logged in during escalation. No shell spraying required. However, adfind and setspn lack default options to quickly run SPN queries against groups so I wrote a little PowerShell module called “Get-SPN” to help make my life easier.

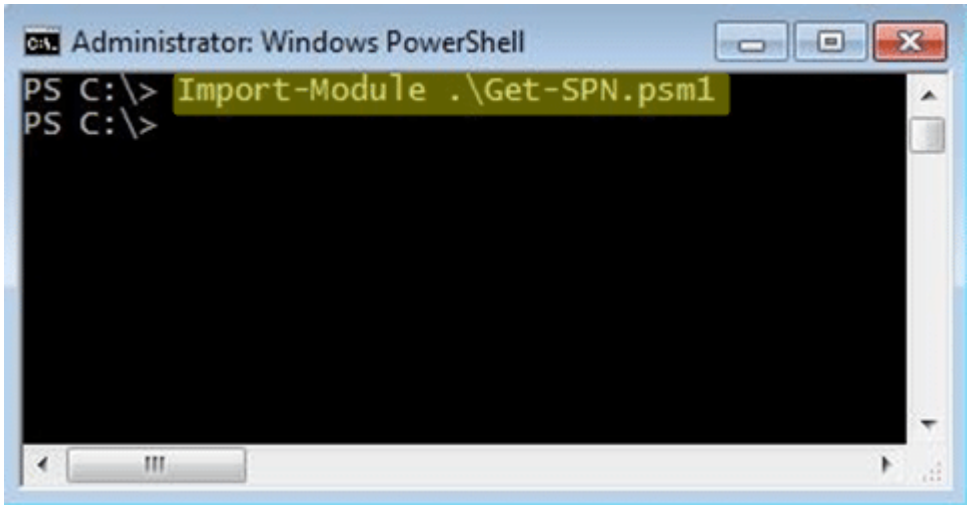
Get-SPN PowerShell Module

The Get-SPN PowerShell module provides an easy way to quickly search LDAP for accounts that match a specific user, group, or SPN service name. For those who are interested it can be downloaded from my Github account here. Please note that it does require PowerShell v3. The module can be installed manually by downloading the Get-SPN.psm1 file to one of two locations:

```
%USERPROFILE%DocumentsWindowsPowerShellModules
%WINDIR%System32WindowsPowerShellv1.0Modules
```

It can then be imported with the following command:

```
Import-Module .Get-SPN.psm1
```



After it’s installed for your current session you should be good to go. Below are a few examples from my small test environment to help get you started. More examples can be found in the help for the command.

```
Get-Help Get-SPN -full
```

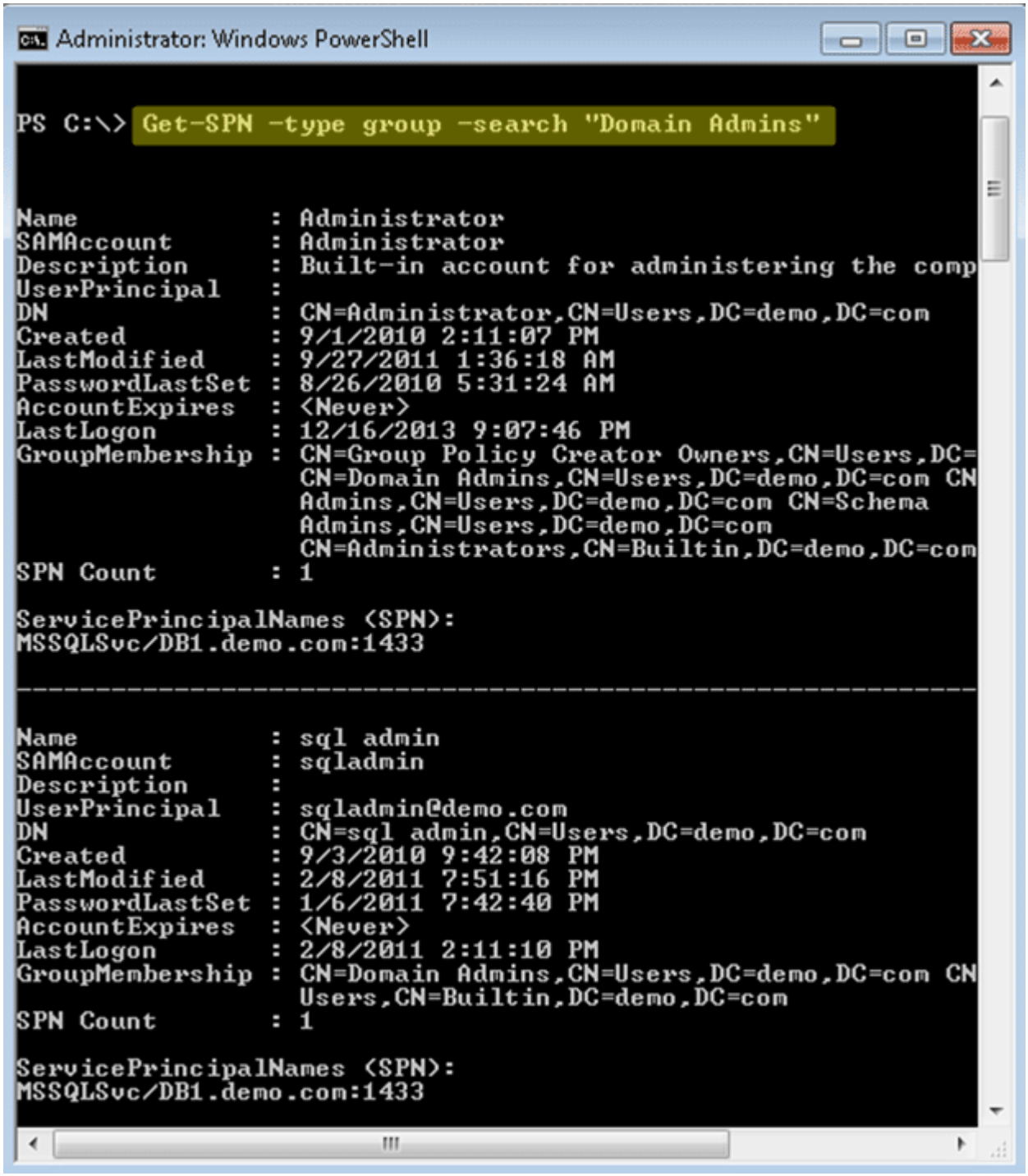
Find All Servers where Domain Admins are Registered to Run Services

If you are executing the command as domain user or LocalSystem from a domain computer then you can use the command below:

```
Get-SPN -type group -search "Domain Admins" -List yes | Format-Table –Autosize
```

The command can also be run without the “-list” options for more verbose output. For example:

```
Get-SPN -type group -search "Domain Admins"
```



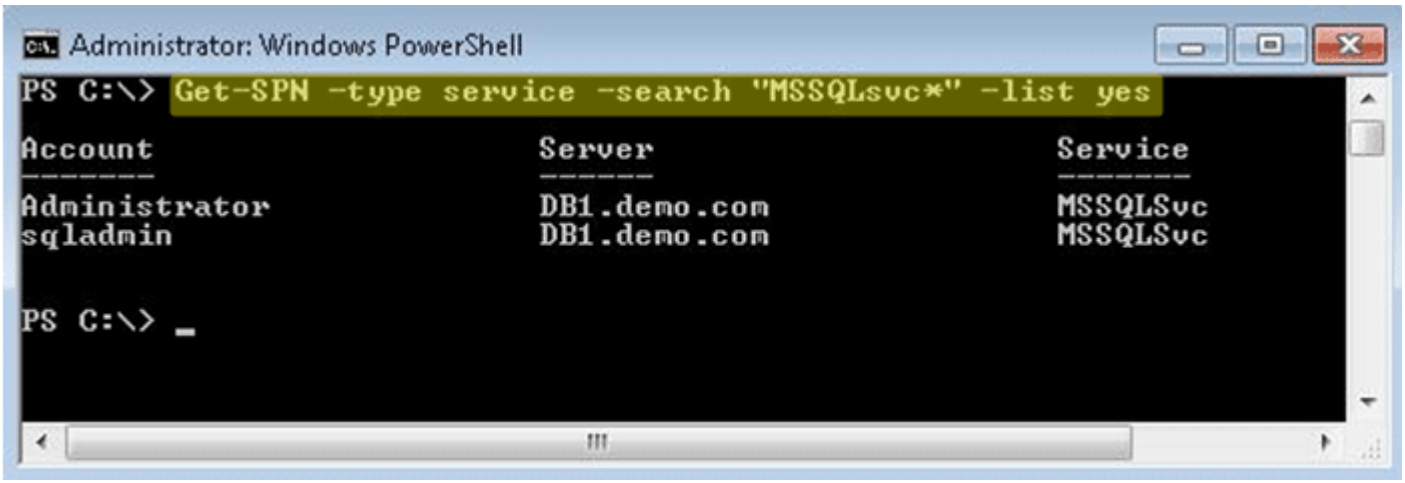
If you are executing from a non domain system with domain credentials you can use the command below. The “DomainController” and “Credential” options can be used with any of the Get-SPN queries.

```
Get-SPN  -type group -search "Domain Admins" -List yes -DomainController
192.168.1.100 -Credential domainuser | Format-Table –Autosize
```

Find All Registered SQL Servers on the Domain

If you are executing the command as domain user or LocalSystem from a domain computer then you can use the command below:

```
Get-SPN  -type service -search "MSSQLSvc*" -List yes | Format-Table –Autosize
```



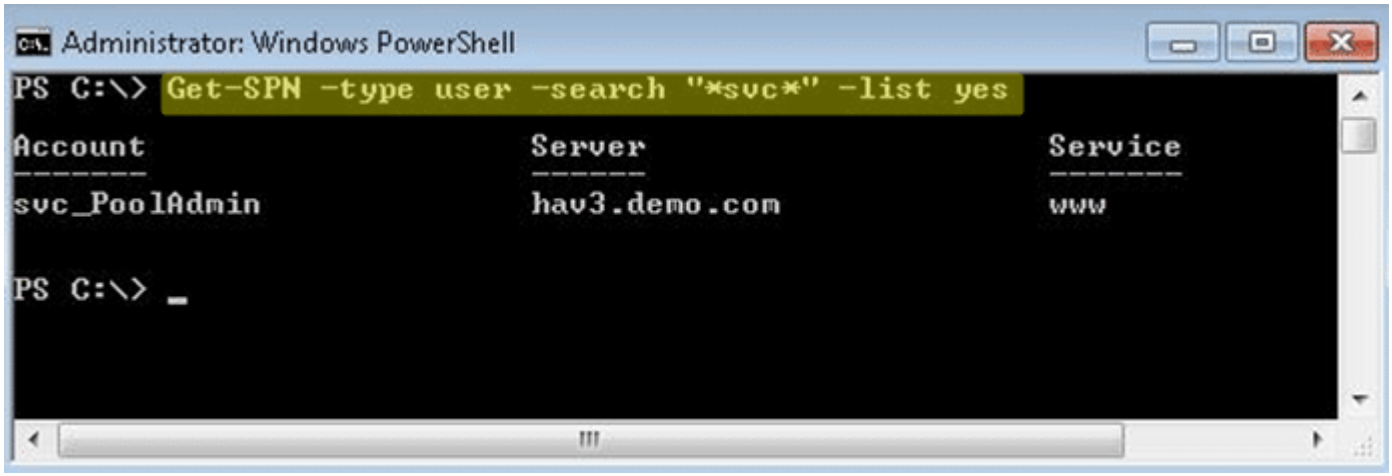
For those interested in services other than SQL Server below is a list of standard SPN service names.

alerter,appmgmt,browser,cifs,cisvc,clipsrv,dcom,dhcp,dmserver,dns,dnscache,eventlog,eventsystem,fax
,
http,ias,iisadmin,messenger,msiserver,mcsvc,netdde,netddedsm,netlogon,netman,nmagent,oakley,plug play,policyagent,
protectedstorage,rasman,remoteaccess,replicator,rpc,rpclocator,rpcss,rsvp,samss,scardsvr,scesrv,schedule,scm,seclogon,
snmp,spooler,tapisrv,time,trksvr,trkwks,ups,w3svc,wins,www

Finding All ServicePrincipalName Entries for Domain Users Matching String

If you are executing the command as domain user or LocalSystem from a domain computer then you can use the command below:

```
Get-SPN  -type user -search "*svc*" -List yes
```



Wrap Up

At this point there are a few limitations I should call out if you are planning to use SPNs to locate

systems where Domain Admin accounts are logged on:

1. Not all Domain Admin accounts will be used to run services.
2. SPNs are automatically registered when an application is installed, but in most cases if the account is changed after the initial installation then it will not be reflected in the SPN unless manually added.

As a result, most of the time SPNs are very useful for finding Domain Admins, but in some environments they are relatively useless. Regardless, leveraging them to find Domain Admin systems means that you don't have to perform scanning or shell spraying over the network. This is nice in the end, because it helps reduce the attack fingerprint and detection during penetration tests. Finally, don't forget that ServicePrincpleNames can be used to locate high value data targets such as SQL Servers, Web Servers, etc on the domain. Good hunting. Have fun and hack responsibly.

References

- http://technet.microsoft.com/en-us/library/cc731241.aspx
- http://msdn.microsoft.com/en-us/library/dd878324(v=vs.85).aspx
- http://msdn.microsoft.com/en-us/library/windows/desktop/ms677949(v=vs.85).aspx
- http://go.microsoft.com/fwlink/?LinkId=198395
- http://www.microsoft.com/en-us/download/details.aspx?id=15326
- http://technet.microsoft.com/en-us/library/aa996205%28v=exchg.65%29.aspx