Powerview 3.0 Cheat Sheet

SPECTER OPS

Getting Started

PowerView’s ‘bleeding edge’ will always in be the development branch of PowerSploit: http://bit.ly/1pzQCnv

Load from disk: 1) C:\> **powershell –exec bypass** 2) **PS C:\> Import-Module powerview.ps1**

Run on non-domain joined machine: 1) configure DNS to point to DC of domain, 2) **runas /netonly /user:DOMAIN\user powershell.exe**

Load in Cobalt Strike’s Beacon: **beacon> powershell-import /local/path/to/PowerView.ps1** , then **beacon> powershell CMDLET-NAME**

Getting help: PS C:\> **Get-Help Cmdlet-Name [-detailed]**

Filtering and Output

Execute a command on each result object	**… | %{…Invoke-Command $_ }**
Filter result objects by field	**… | ? {$_.Field –eq X}**
Only return certain properties	**… | Select prop1,prop2**
Display output as a list	**… | fl**
Display output as wrapped table	**… | ft -wrap**
Write out to file	**… | Out-File -Encoding Ascii out.txt**
Write to .csv	**… | Export-CSV -NoTypeInformation out.csv**
Write to .xml object	**…| Export-Clixml obj.xml**
Read .xml object	**$obj = Import-Clixml obj.xml**

Function Naming Scheme

All PowerView functions should now following a proper **Verb-PrefixNoun** format:

Get-*	Retrieve full raw data objects
Find-*	Find specific data entries in a data set or execute threaded computer enumeration
Add-*	Add a new object to a destination
Set-*	Modify a given object
Invoke-*	Lazy catch-all

Noun prefixes now give an indication of the data source:

Verb-**Domain**X	LDAP/.NET AD connections
Verb-**WMI**X	Uses WMI for connections/enumeration
Verb-**Net**X	Uses Win32 API calls

Common Options

The object to query- samaccountname, DN, SID, GUID, or dnsHostname. Wildcards accepted.	**-Identity <X>**
Display verbose status/debug information	**-Verbose**
Execute the query in a foreign domain	**-Domain foreign.com**
Utilize a custom LDAP filter	**-LDAPFilter ‘(prop-value)’**
Only return the specified properties from the server	**-Properties prop1,prop2**
Search through a particular OU	**-SearchBase “ldap://OU=…”**
Search through a *global catalog*	**-SearchBase “GC://domain.com”**
Bind to a particular server for the search	**-Server “dc.domain.com”**
Return specific security information with the search	**-SecurityMasks [Dacl/Owner/Sacl]**
Only return one result	**-FindOne**

-Credential

All PowerView functions now accept an alternate **–Credential** specification:

PS C:\> **$SecPassword = ConvertTo-SecureString 'BurgerBurgerBurger!' -AsPlainText -Force**

PS C:\> **$Cred = New-Object System.Management.Automation.PSCredential('TESTLAB\dfm.a', $SecPassword)**

PS C:\> **Get-DomainUser -Credential $Cred**

Computer Enumeration

Get-DomainComputer will enumerate computer objects on a given domain through LDAP.

Return only live hosts	**-Ping**
Machines with unconstrained delegation	**-Unconstrained**
Trusted to authenticate for other principals	**-TrustedToAuth**
Specific service principal name, wildcards accepted	**-SPN *SQL***
Specific OS, wildcards accepted	**-OperatingSystem <X>**
Specific service pack, wildcards accepted	**-ServicePack <X>**

Identifying Your Prey

Get-DomainUser will enumerate user objects on a given domain through LDAP.

Return users with “admin” in the user name	**-Identity “*john*”**
Return users who are (or were) a member of an admin protected group	**-AdminCount**
Users with a service principal name set (likely service accounts)	**-SPN**
Trusted to authenticate for other principals	**-TrustedToAuth**
“Do not require Kerberos preauthentication” set	**-PreauthNotRequired**

Get-DomainGroup will enumerate *group* objects themselves on a given domain through LDAP.

Return all groups with "admin" in the name	**-Identity *admin***
Return all groups a particular user/group is a part of	**-MemberIdentity <X>**
Return privileged groups	**-AdminCount**
Return groups with a particular scope	**-GroupScope [DomainLocal/Global/ Universal]**

Get-DomainGroupMember will enumerate the *members* of a specific group on a given domain through LDAP.

Specified group name	**-Identity "Domain Admins"**
Recursively resolve the members of any results that are groups	**-Recurse**

If you're not sure of the object type, you can use **Get-DomainObject**. **Get-DomainObjectACL** will return the ACLs associated with a specific active directory object**.** The **–ResolveGUIDs** flag resolves ACE GUIDs to their display names.

Domain [Trusts]

Info on the current forest	**Get-Forest**
Enumerate all domains in the current forest	**Get-ForestDomain**
Get all forest trusts for the current forest	**Get-ForestTrust**
Info on the current domain	**Get-Domain**
Get all domain trusts (à la nltest /trusted_domains)	**Get-DomainTrust**
Recursively map all domain trusts	**Get-DomainTrustMapping**

Find users in groups outside of the given domain (*outgoing* access)	**Get-DomainForeignUser**
Find groups w/ users outside of the given domain (*incoming* access)	**Get-DomainForeignGroupMember –Domain target.domain.com**

All Verb–Domain* functions also accept **–Domain <X>** to query the specified information from a foreign domain.

User-Hunting

Find-DomainUserLocation (old Invoke-UserHunter) will use LDAP queries and API calls to locate users on the domain. **Note:** default behavior searches for "Domain Admins" and touches every machine on the domain!

Specifies one or more *user* identifies to hunt for	**-UserIdentity <X>**
Specifies hosts to enumerate for session information	**-ComputerName X,Y**
Species one or more *groups* to query for users to hunt for	**-UserGroupIdentity <X>**
Show all results (i.e. don't filter by user targets)	**-ShowAll**
Hunt using only session information from file servers/DCs	**-Stealth**
Check if the current user has local admin access to computers where target users are found	**-CheckAccess**

Data Mining

Find-DomainShare (old Invoke-ShareFinder) will use LDAP queries and API calls to search for open shares on the domain. **Note:** default behavior touches every machine on the domain!

Only return shares the current user can read	**-CheckShareAccess**
Only return shares from machines in a given OU	**-ComputerSearchBase "ldap://OU=…"**

Find-InterestingFile will recursively search a given local/UNC path for files matching specific criteria.

Search a specific UNC path	**-Path \\SERVER\Share**
Only return files with the specified search terms in their names	**-Include term1,term2,term3**
Only return office docs	**-OfficeDocs**
Only return files accessed within the last week	**-LastAccessTime (Get-Date).AddDays(-7)**

Local Admin Enumeration

Get-NetLocalGroupMember will enumerate the local users/groups from localhost or a remote machine.

Enumerate local admins from hostname (or IP)	**-ComputerName <X>**
Use an alternate group besides local admins	**-GroupName "Remote Desktop Users"**
Uses the WinNT service provider (default) or Win32 API calls	**-Method [WinNT/API]**

Misc. Functions

Return domain OUs	**Get-DomainOU**
Return domain GPOs	**Get-DomainGPO**
Find likely file servers based on user properties	**Get-DomainFileServer**
Enumerate shares on a specific machine	**Get-NetShare <X>**
Enumerate shares on a specific machine	**Get-NetSession <X>**
Enumerate RDP sessions (and source IPs)	**Get-NetRDPSession <X>**

More Information

Recent PowerView update: http://bit.ly/2rseIm6

PowerView Tricks - http://bit.ly/2tDBAQi

http://www.harmj0y.net/blog/tag/powerview/

https://specterops.io