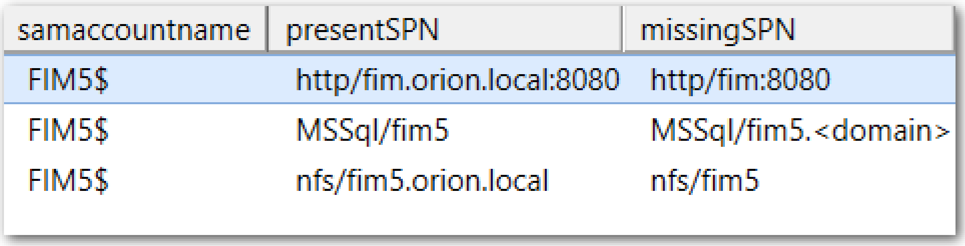
Find missing SPN registrations

learn.microsoft.com/en-us/archive/blogs/389thoughts/find-missing-spn-registrations



samaccountname	presentSPN	missingSPN
FIM5$	http/fim.orion.local:8080	http/fim:8080
FIM5$	MSSql/fim5	MSSql/fim5.<domain>
FIM5$	nfs/fim5.orion.local	nfs/fim5

- Article
- 03/19/2017

Active Directory admins are probably well aware of how Kerberos works. If you need a little refresher, check out the article over at askds: Kerberos for the busy admin. Kerberos requires a service principle name (SPN) for each Kerberos enabled network service offered in the forest: a file service, KDC, web farm, whatever. Typical examples of SPNs are: HOST/s1.contoso.com, HTTP/webfarm.contoso.com:8080.

There are a few things to know about SPNs:

- Each SPN must be unique in the forest. Anything else is just wrong.
- SPNs can belong to computer accounts and service accounts, nothing else.
- Each account can have multiple different SPNs.

It is easy to spot duplicate SPNs; just run: *setspn -x -f* on any machine with the AD management tools. However, it is not so easy to spot missing SPNs. For services that are missing the correct SPNs it simply means that they cannot use Kerberos. With luck, they fall back to NTLM. With no luck, they fail.

There is one simple case where an automated inspection can show potentially missing SPNs. Here is the thing: for a lot of services, you need a registration for *both* the FQDN and the hostname, because the client (such as Explorer) might as for any of them. For instance:

- HOST/s1.contoso.com
- HOST/s1

Or, in a situation that I encountered at a customer recently:

- CIFS/nas.fabrikam.com
- CIFS/nas

For example, if you ask Explorer to go to \\s1.contoso.com\userdata, it will use the SPN HOST/s1.contoso.com. But if you just use the hostname like this: \\s1\userdata, it will ask for HOST/s1.

A little automation goes a long way, so I wrote a script for this. It is called Find-PossibleMissingSPNs.ps1, and you can find it at the TechNet Gallery: Find missing Kerberos SPNs. Let's have a look at it. This is the full script, minus some comments:

```
[powershell]
[CmdletBinding()]
Param
(
# start the search at this DN. Default is to search all of the domain.
[string]$DN = (Get-ADDomain).DistinguishedName
)

$servicesclasses2check = @("host", "cifs", "nfs", "http", "mssql")
$filter = '(&(servicePrincipalname=*)(|(objectcategory=computer)
(objectcategory=person)))'
$propertylist = @("servicePrincipalname", "samaccountname")
Get-ADObject -LDAPFilter $filter -SearchBase $DN -SearchScope Subtree -Properties
$propertylist -PipelineVariable account | ForEach-Object {
#
# Create list of interesting SPNs for each account. Strong assumption for all code: SPN is
syntactically correct.
#
$spnlist = $account.servicePrincipalName | Where-Object {
($serviceclass, $hostname, $service) = $_ -split '/'
($servicesclasses2check -contains $serviceclass) -and -not $service
}

#
# Look for cases where there is no pair of (host, host.domain) SPNs.
#
foreach ($spn in $spnlist)
{
($serviceclass, $hostname, $service) = $spn -split '/'
if ($service) { $service = "/$service" }
($fullname, $port) = $hostname -split ':'
if ($port) { $port = ":$port" }
($shortname, $domain) = $fullname -split '[.]'
#
# define the regexp matching the missing SPN and go look for it
#
if ($domain) {
$needsSPN = "${serviceclass}/${shortname}${port}${service}`$"
$needsSPNtxt = "${serviceclass}/${shortname}${port}${service}"
} else {
$needsSPN = "$serviceclass/${shortname}[.][a-zA-Z0-9-]+.*${port}${service}`$"
$needsSPNtxt = "$serviceclass/${shortname}.<domain>${port}${service}"
}
#
# search the array of SPNs to see if the _other_ SPN is there. If not, we have problem case.
#
if (-not ($spnlist -match $needsSPN))
{
[PSCustomobject] @{
samaccountname = $account.samaccountname
presentSPN = $spn
missingSPN = $needsSPNtxt
}
}
}
}
[/powershell]
```

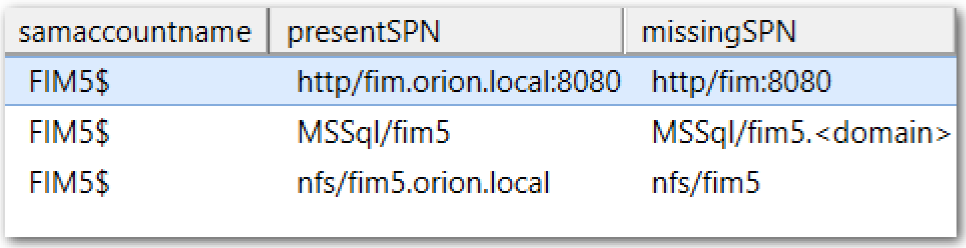
The scripts starts with searching for all users and computers having an SPN within the search scope, which is the domain by default, but can be any OU that you want. For each of this accounts, only the "interesting" SPN types are filtered. Systems SPNs such as those for DFSR or GC are ignored for now, although you could add them to list if needed. Note that all three-part SPNs are assumed to be correct. All SPNs in the "interesting" list are split into their parts, with the assumption that they are syntactically correct.

The next step is to define a regular expression search pattern for the potentially missing SPN. If the current SPN has the domain form (HOST/s1.contoso.com), we look for the hostname form (HOST/s1), and vice versa. If the current SPN list does not contain any record matching the regular expression, we have potentially missing SPN records. Finally, the results are pushed into the pipeline.

Save or download the script as Find-PossibleMissingSPNs.ps1, and execute it:

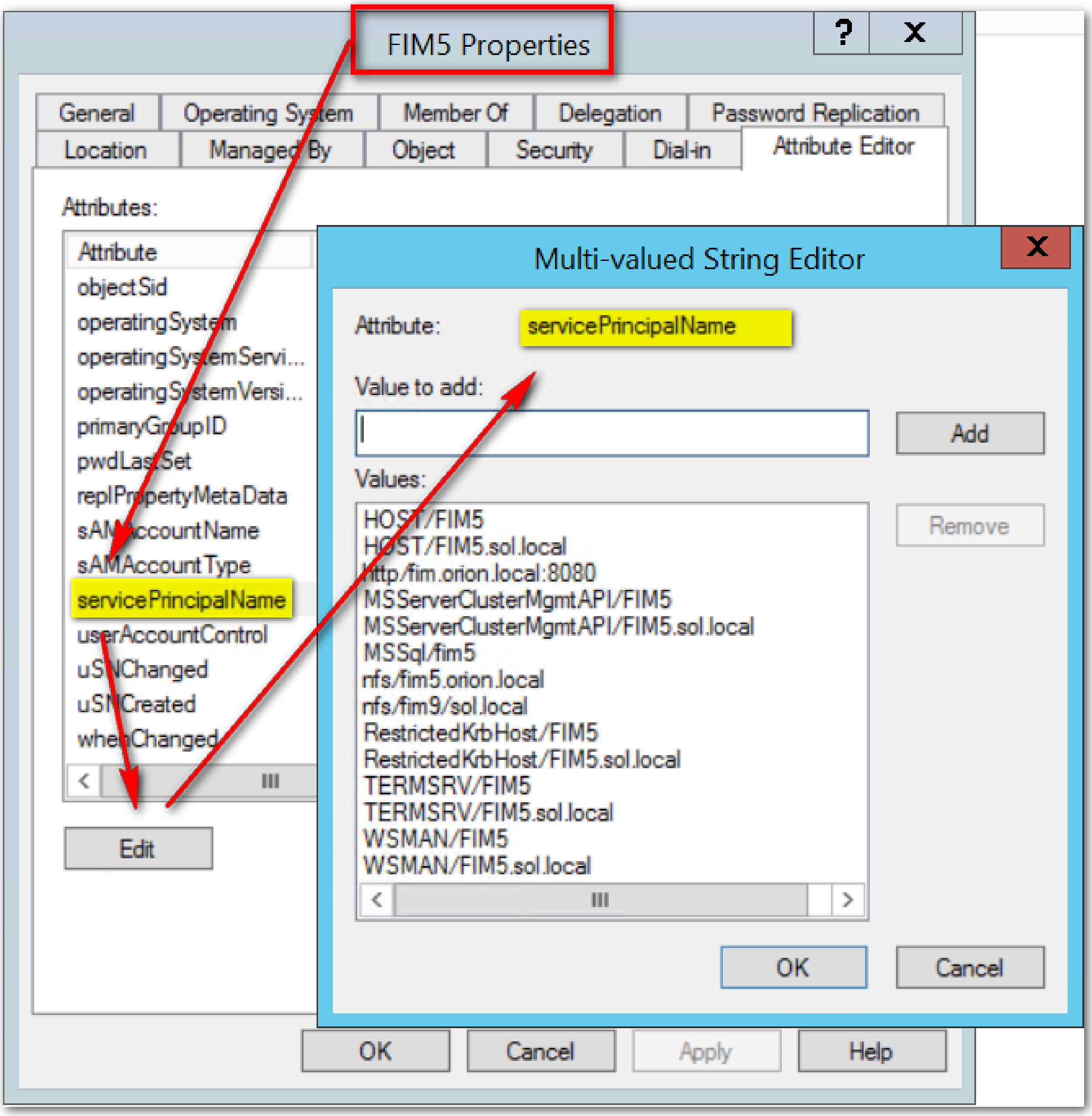
```
[powershell]
Find-PossibleMissingSPNs.ps1 | out-gridview
[/powershell]
```

In my lab, the results look like this:



samaccountname	presentSPN	missingSPN
FIM5$	http/fim.orion.local:8080	http/fim:8080
FIM5$	MSSql/fim5	MSSql/fim5.<domain>
FIM5$	nfs/fim5.orion.local	nfs/fim5

As you can see, I managed to misconfigure this box rather badly. There seems to be a website for fim.orion.local, defined for the FQDN but not for the hostname. SQL has the reverse problem (host, but no domain) and somebody installed an NFS service without considering the hostname. Confirm these findings by looking at the full SPN list. Look up the computer account in AD Users & Computers:



Attribute: servicePrincipalName

Value to add:

Values:

```
HOST/FIM5
HOST/FIM5.sol.local
http/fim.orion.local:8080
MSServerClusterMgmtAPI/FIM5
MSServerClusterMgmtAPI/FIM5.sol.local
MSSql/fim5
nfs/fim5.orion.local
nfs/fim9/sol.local
RestrictedKrbHost/FIM5
RestrictedKrbHost/FIM5.sol.local
TERMSRV/FIM5
TERMSRV/FIM5.sol.local
WSMAN/FIM5
WSMAN/FIM5.sol.local
```