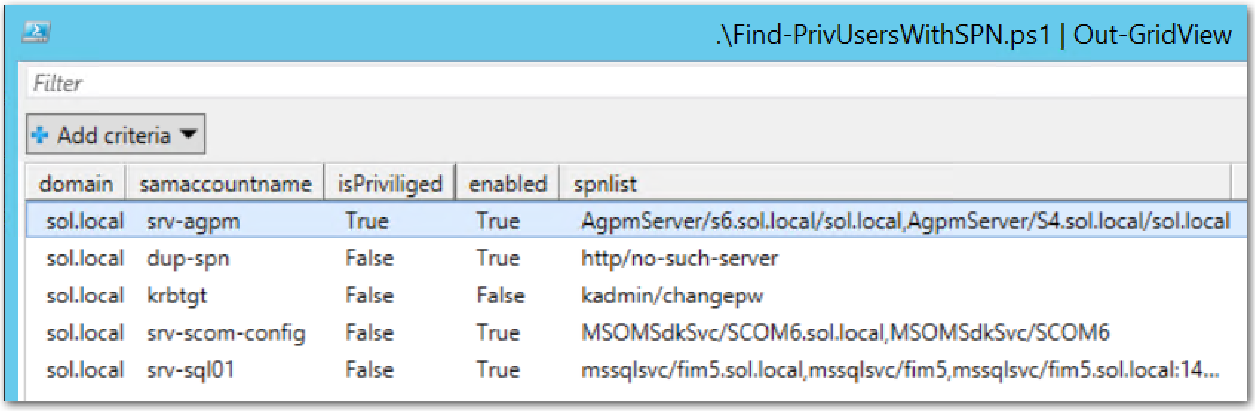
Quickly find potential Kerberoast victims

learn.microsoft.com/en-us/archive/blogs/389thoughts/quickly-find-potential-kerberoast-victims


.\Find-PrivUsersWithSPN.ps1 | Out-GridView

domain	samaccountname	isPriviliged	enabled	spnlist
sol.local	srv-agpm	True	True	AgpmServer/s6.sol.local/sol.local,AgpmServer/S4.sol.local/sol.local
sol.local	dup-spn	False	True	http/no-such-server
sol.local	krbtgt	False	False	kadmin/changepw
sol.local	srv-scom-config	False	True	MSOMSdkSvc/SCOM6.sol.local,MSOMSdkSvc/SCOM6
sol.local	srv-sql01	False	True	mssqlsvc/fim5.sol.local,mssqlsvc/fim5,mssqlsvc/fim5.sol.local:14...

- Article
- 04/25/2018

Lately I have been talking a lot about the Kerberoast exploit with my customers. Before I dive in let me start off by giving you the two great resources that I used myself to get up to speed:

- Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain
- Service Accounts – Weakest Link in the Chain?

What is this aptly named Kerberoast exploit? It is a way to obtain the password of a service account, abusing the Kerberos protocol in a very elegant way. For this (conceptual) discussion I am going to assume that you know your way around Kerberos. If not (quite), read up at: Kerberos for the Busy Admin.

The central idea is that if you can get hold of a ticket generated for a particular service, you can use this to bruteforce the password: the correct password is the one that generates the hash that in turn is used to reconstruct the ticket. Same ticket as the original one? You just found the password.

The brilliant part of the idea is that it turns out to be very simple to get such a ticket. No need to put wireshark on the network to wait for something interesting. Instead, you just ask for the ticket! Well, there is a little more to it, but not much.

Any Kerberos service must have a Service Principle Name (SPN), such as `HOST/s1.sol.local`. All computer accounts have SPNs, but are completely unsuitable for bruteforcing because they have 128-byte random passwords. The same holds for Managed Service Accounts. On the other hand, normal user accounts do not have SPNs and cannot be used to represent a Kerberos service. The exceptions are user accounts that *do* have SPNs and are used as service accounts representing a network service such as SQL, a web service, ADFS, etc. And you know user accounts: their passwords are certainly not 128-byte random binary strings!

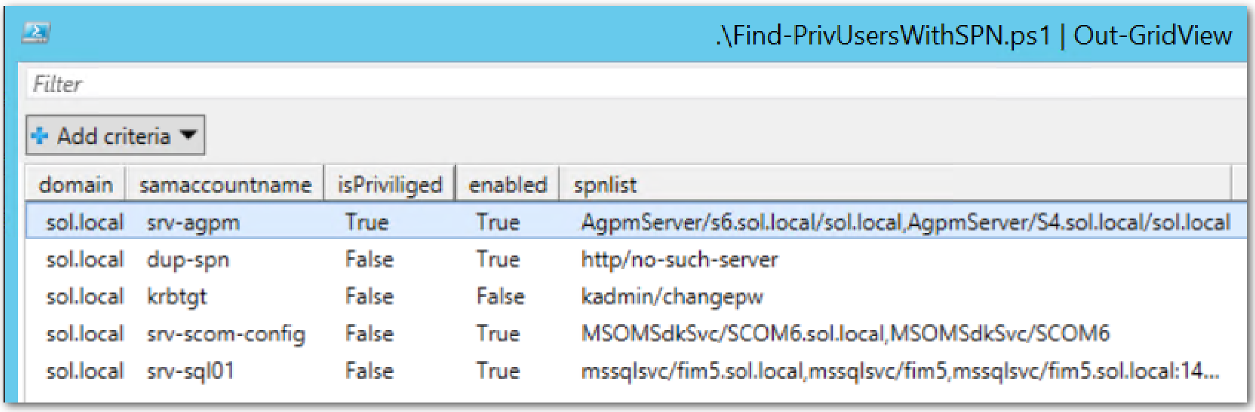
If you know of a user account with an SPN, you can simply ask for a Kerberos Service Ticket for this account using nothing more than `Domain User` credentials. That's right, no admin permissions anywhere. Any normal domain user can do this which is another aspect that makes Kerberoasting so interesting. Once you have the ticket, you can start guessing the password that generates the hash, which generates the ticket. And by the way, nothing stops you from asking for tickets for different services at the same time, and try to bruteforce them in parallel.

Note that this bruteforcing is *invisible* to you as an admin. There is no network traffic involved anymore once the ticket is issued, no logon attempts to DCs, nothing like that. Instead, all the work is local to the machine, executed at full speed of the CPU, for as long as it takes. The ticket is only invalidated when the account behind it changes its password.

A technical complication here is that AD accounts can have multiple hashes. Without going into details, the two most relevant hashes nowadays are the RC4 and AES hashes. And you guessed it, only the tickets based on RC4 hashes are easy to bruteforce. Unfortunately, this is the greater majority in most networks.

The point of my post is to warn you for the suddenly popular Kerberoast approach. The most interesting targets will be highly privileged service accounts such as the ones that are member of Domain Admins. But there are also others cases, such as accounts giving access to important data, or those that are configured for Kerberos delegation.

I wanted to show you a simple, easy to understand script that shows you which privileged accounts have SPNs. I put it up on GitHub: Find-PrivUsersWithSPN.ps1. Run it with no arguments and it will analyze the current domain and show the users having an SPN, with a flag telling you if it belongs to a privileged group.



domain	samaccountname	isPriviliged	enabled	spnlist
sol.local	srv-agpm	True	True	AgpmServer/s6.sol.local/sol.local,AgpmServer/S4.sol.local/sol.local
sol.local	dup-spn	False	True	http/no-such-server
sol.local	krbtgt	False	False	kadmin/changepw
sol.local	srv-scom-config	False	True	MSOMSdkSvc/SCOM6.sol.local,MSOMSdkSvc/SCOM6
sol.local	srv-sql01	False	True	mssqlsvc/fim5.sol.local,mssqlsvc/fim5,mssqlsvc/fim5.sol.local:14...

So in this case I have one user that I need to investigate right away: `srv-agpm` is apparently privileged. It turns out to be a member of Backup Operators. Not great, but no disaster either.

Again, for the full details, see the links at the beginning of this post. What I'm writing here is just derived work that does not retrieve all relevant details. It just reveals the accounts that you should focus on first.

What about remediation? You have several options.

- Make sure that the password is really long, long enough to make bruteforcing impractical. Say, 25 characters or more.
- Force the service account to use AES.
- Move to using Managed Service Accounts.
- Stop using highly privileged service accounts...