Detecting Kerberoasting Activity Part 2 – Creating a Kerberoast Service Account Honeypot

adsecurity.org

Sean Metcalf February 8, 2017

In my previous post, "Detecting Kerberoasting Activity" I explain how Kerberoasting works and describe how to detect potential Kerberoasting activity. The trick to this is understanding what activity is normal in order to filter out the noise and increase the fidelity of the alert.

This post describes how to filter from millions of events to a single one to detect Kerberoasting activity.

Detecting Kerberoasting Activity
As I noted in the previous post, looking for TGS service tickets with RC4 encryption was a good method to discover Kerberoasting activity.

Windows added Kerberos AES (128 & 256) encryption starting with Windows Server 2008 and Windows Vista which means that most Kerberos requests will be AES encrypted with any modern Windows OS. Any Kerberos RC4 tickets requested should be the exception. There are systems that only support Kerberos RC4 by default (NetApp). Inter-forest Kerberos tickets also use RC4 unless configured for AES – ensure your forest trusts support AES and then enable AES over the trust.

Once all Domain Controllers are configured to log 4769 events, these events need to be filtered before sending the data into a SIEM/Splunk. Since we are only really interested in Kerberos TGS service tickets with RC4 encryption, it's possible to filter the events. As shown above, Kerberos events with AES encryption has Ticket Encryption Type set to 0x12.
Kerberos RC4 encrypted tickets have Ticket Encryption Type set to 0x17.

These events can be filtered using the following which greatly reduces the amount of events flowing into the SIEM/Splunk:

- Ticket Options: 0x40810000
- Ticket Encryption: 0x17

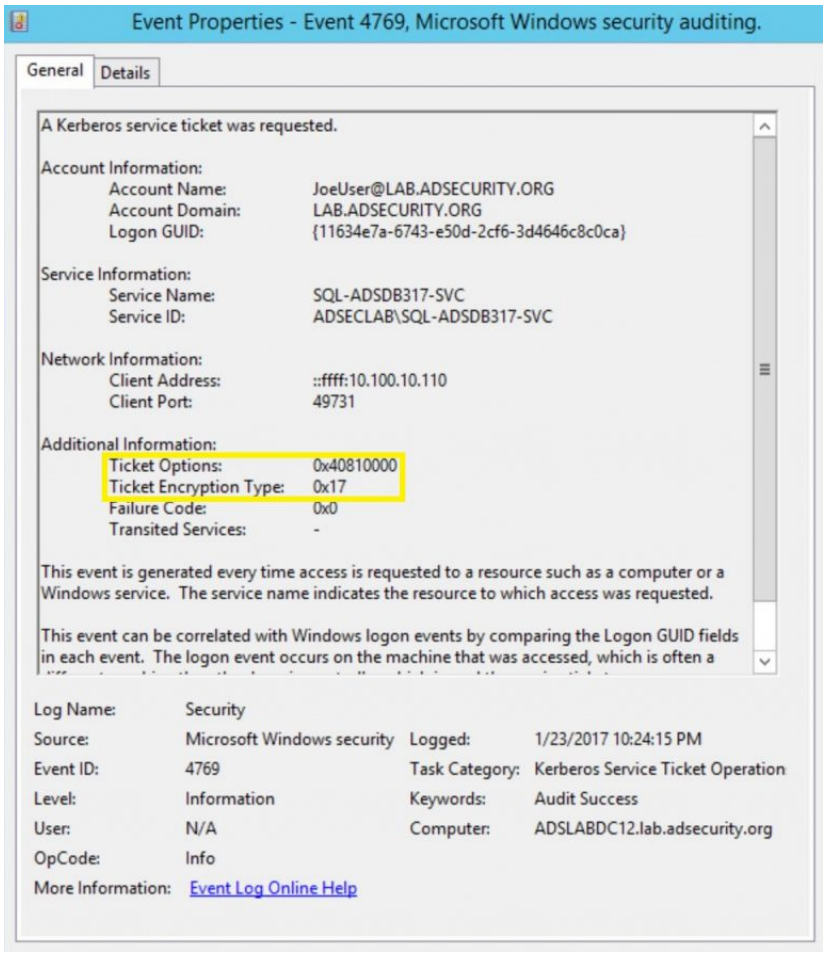
With this information, we can start investigating potential Kerberoasting activity and reduce the number of 4769 events.

We can further reduce the number of 4769 events that flow into SIEM/Splunk:

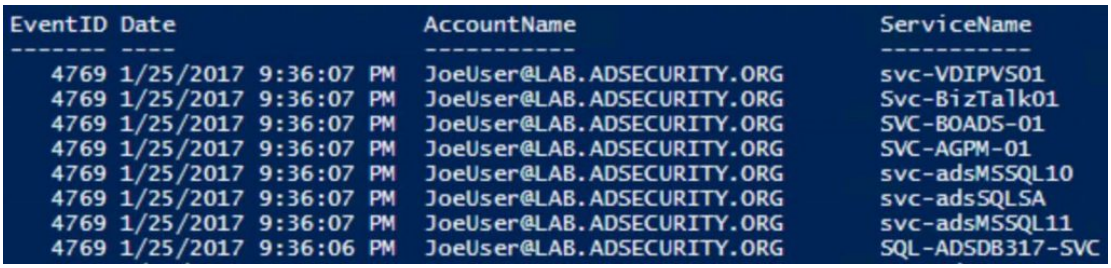
- Filter out requests from service accounts (ads45service@lab.adsecurity.org)
- Filter on Audit Success
- Filter out requests for service names with a "$" which are typically for computer accounts (or trusts or Managed Service Accounts, all accounts where Windows automatically generates a long, complex password).

In limited testing, I've seen 4769 event totals reduced from millions to thousands and hundreds using these filtering techniques.

Here's a 4769 event that may potentially be from Kerberoasting activity:



Following this line of thought, we can look at TGS ticket requests with specific ticket encryption & ticket options to identify potential Kerberoast activity.

> Using the information regarding ticket encryption type and ticket options, we can use PowerShell to parse the DC's event log looking for 4769 events with this info.
>
> 
>
> | EventID | Date | AccountName | ServiceName |
> |---|---|---|---|
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | svc-VDIPVS01 |
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | Svc-BizTalk01 |
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | SVC-BOADS-01 |
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | SVC-AGPM-01 |
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | svc-adsMSSQL10 |
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | svc-adsSQLSA |
> | 4769 | 1/25/2017 9:36:07 PM | JoeUser@LAB.ADSECURITY.ORG | svc-adsMSSQL11 |
> | 4769 | 1/25/2017 9:36:06 PM | JoeUser@LAB.ADSECURITY.ORG | SQL-ADSDB317-SVC |
>
> That looks really odd. Why would any account request several different service names (Citrix PVS, BizTalk, Business Objects, AGPM GPO management, and several SQL service accounts) within a second or two of each other?
>
> That stands out and looks really suspicious and is very likely Kerberoasting activity. This provides great information on what users could be compromised and what activity on which computers should be scrutinized.
>
> A single user requesting RC4 encrypted TGS tickets for several services, such as lots of SQL service principal names is suspicious and it's worth investigating the IP (client address) the requests came from. The same thing is true for multiple RC4 encrypted TGS requests over time for lots of service principal names. A pattern does emerge when there's one or two accounts that request a variety or RC4 TGS tickets.

Read the entire post for more information on how Kerberoasting works, etc.

Filtering the Noise to Find Malicious Activity
I note in that post that 4769 events on Domain Controllers are extremely numerous, some of the most numerous events on a network.

But, what if we only cared about 1 event?
That would reduce the number of 4769 events down to a single event that only occurs when something malicious is happening?

This post describes how to further narrow down to best detect Kerberoasting activity on a network: Creating a Service Account Honeypot to detect Kerberoasting.

Note: I still recommend filtering 4769 event IDs on Domain Controllers and flowing them into SIEM/Splunk since this will provide information on resources users are accessing as well as help flag when a single user is requesting multiple service principal names (which is suspicious).

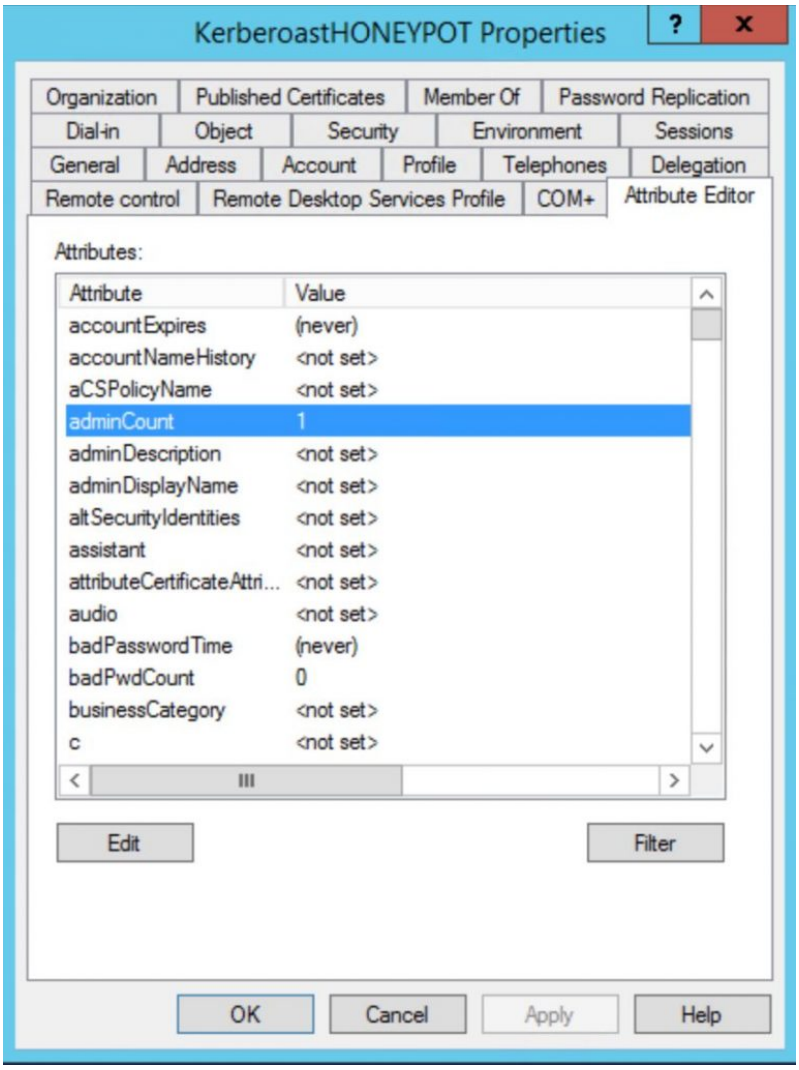
Creating a Kerberoast Service Account Honeypot

In order to create a Kerberoast Service Account Honeypot, we need to create a user account in AD and give it a fake service principal name (SPN). It has to be fake so we know that when it's requested, this request is not valid and therefore is malicious. It's also

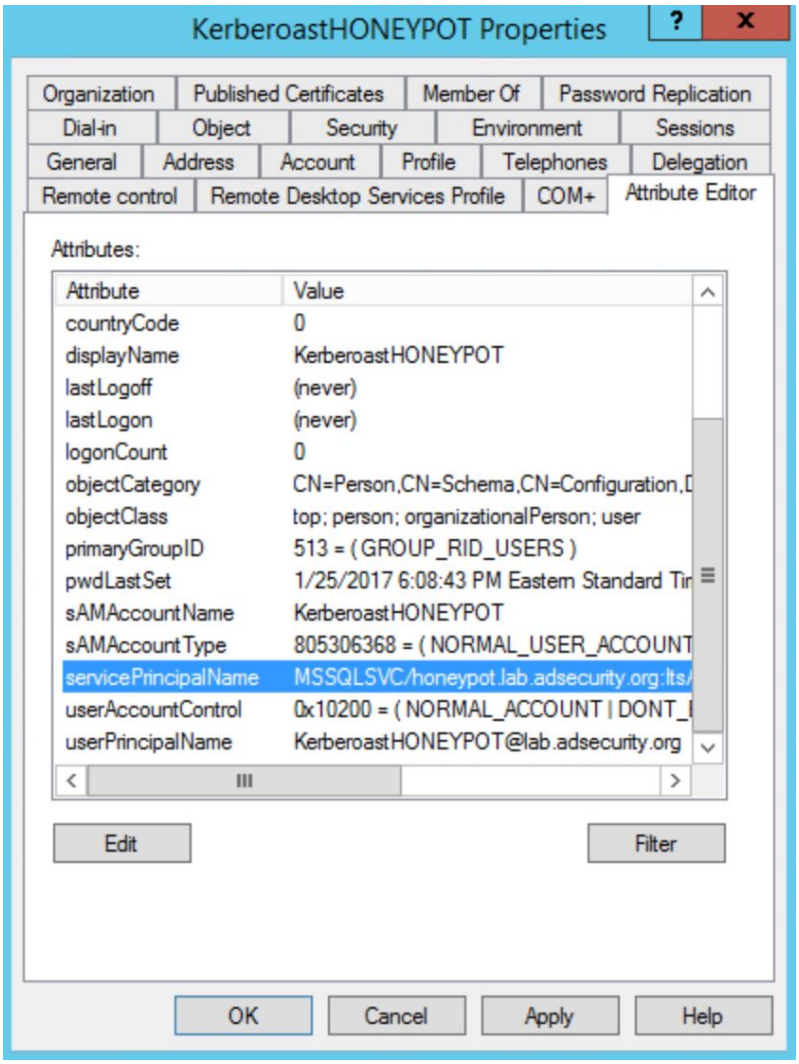
important to make this account look attractive by setting the “AdminCount” attribute to 1 as this flags the account as potentially having elevated AD rights. Adding this account to a bunch of fake groups made to look like it’s providing additional elevated rights helps to add to the illusion.

Step 1: Create a new AD user account.

Step 2: Set the “AdminCount” attribute to 1.



Step 2: Add a Service Principal Name (SPN) to the account. This SPN needs to be unique, so it should not simply be copied from another system. SQL service accounts are pretty common, so that’s not a bad one to use (MSSQLSvc/sql01.lab.adsecurity.org). Just don’t reuse one that already exists.
The following example is a bit less… subtle.



Step 3: It may also be useful to add the honeypot account to a fake group that looks like it might have admin rights.

Note:
To make things interesting, you could give this account an easy password that could be guessed, something like “Password1234” (or a keyboard combination). This way we can monitor if someone logs on with this account. However, the Kerberos TGS service ticket request is enough to know that Kerberoasting activity is occurring and we know from which computer it’s being done thanks to the event information (“Client Address”).

Detecting Kerberoast Service Account Honeypot “Access”

Once the honeypot account is created with a service principal name that doesn’t do anything and therefore should never be requested or used. The reason this service principal name should never be requested is that we made it up and it isn’t linked to any real application that would be requesting it. There is no reason for anyone to every request a Kerberos TGS service ticket for this since there is no actual associated service running for it. Therefore, if we see that someone requested a Kerberos TGS ticket, they are very likely attempting to Kerberoast this account.

An attacker gains access to the internal network and searches for accounts with service principal names and have admincount set to 1.

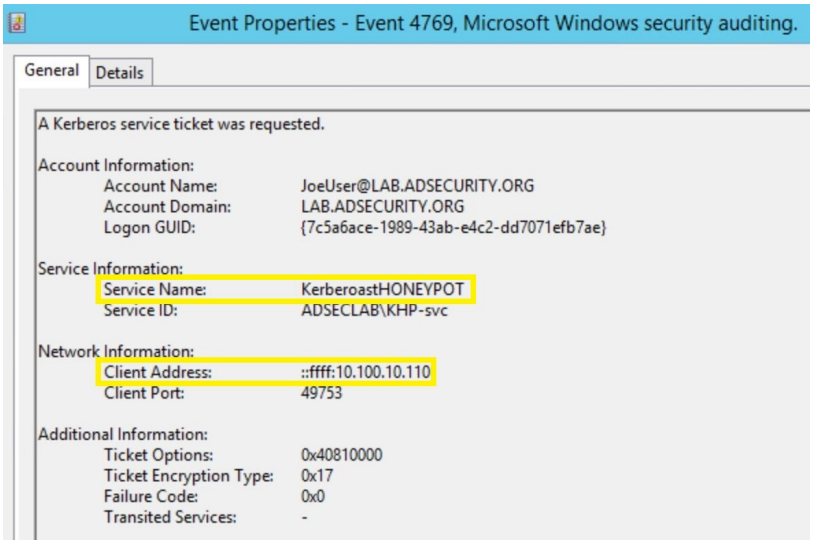
```
PS C:\> Get-ADUser -Filter { (AdminCount -eq 1) -AND (ServicePrincipalName -like "*") } `
   -Property * | Select SAMAccountname,ServicePrincipalName

SAMAccountname      ServicePrincipalName
--------------      --------------------
krbtgt              {kadmin/changepw}
KerberoastHONEYPOT {MSSQLSVC/honeypot.lab.adsecurity.org:ItsATrap}
```

This shows our new honeypot account to the attacker who is now interested in this account and requests an RC4 encrypted Kerberos TGS ticket for the SPN. A klist shows the attacker got the TGS ticket in memory.

```
#1> Client: JoeUser @ LAB.ADSECURITY.ORG
    Server: MSSQLSVC/honeypot.lab.adsecurity.org:ItsATrap @ LAB.ADSECURITY.ORG
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a10000 -> forwardable renewable pre_authent name_canonicalize
    Start Time: 1/25/2017 15:10:27 (local)
    End Time:   1/26/2017 1:10:27 (local)
    Renew Time: 2/1/2017 15:10:27 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called: ADSLABDC12.lab.adsecurity.org
```

By looking for 4769 events on Domain Controllers with the ticket encryption option 0x12 (along with other filters I describe in the Kerberoast detection post), we can see that Joe User requested a Kerberos ticket for a SPN that doesn't exist and should never be requested!
The Account Name shows which account was used and Client Address provides the computer IP from where the attacker requested the honeypot Kerberos service account.




```
PS C:\Windows\system32> $KerberoastEventData | where {$_.ServiceName -like "*Honeypot*"} `
 | select EventID,Date,AccountName,ClientAddress,ServiceName | ft -AutoSize

EventID Date                  AccountName                ClientAddress          ServiceName
------- ----                  -----------                -------------          -----------
   4769 2/8/2017 7:54:21 AM JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 KerberoastHONEYPOT
```

When we use our Kerberoast discovery PowerShell script against the Domain Controller event logs, we find that Joe User has requested a lot of Kerberos service tickets, including the one for our Honeypot (which again should never be requested since it doesn't exist).

```
EventID Date                  AccountName                ClientAddress          ServiceName
------- ----                  -----------                -------------          -----------
   4769 2/6/2017 10:20:28 AM JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 RD.ADSECURITY.ORG
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 SQL-ADSDB317-SVC
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 HanSolo
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 svc-adsMSSQL11
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 svc-adsSQLSA
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 svc-adsMSSQL10
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 KHP-svc
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 SVC-AGPM-01
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 SVC-BOADS-01
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 Svc-BizTalk01
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 svc-VDIPVS01
   4769 2/8/2017 7:29:12 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 C3PO
   4769 2/8/2017 7:34:43 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 KHP-svc
   4769 2/8/2017 7:53:15 AM  JoeUser@LAB.ADSECURITY.ORG ::ffff:10.100.10.110 KerberoastHONEYPOT
```

```
PS C:\Windows\system32>  $KerberoastEventData | where {$_.ServiceName -like "*Honeypot*"} `
 | select EventID,Date,AccountName,ClientAddress,ServiceName


EventID       : 4769
Date          : 2/8/2017 7:54:21 AM
AccountName   : JoeUser@LAB.ADSECURITY.ORG
ClientAddress : ::ffff:10.100.10.110
ServiceName   : KerberoastHONEYPOT
```

Using a service account honeypot, this changes this detection from “potential” Kerberoasting activity, to definite Kerberoasting activity.

Note that we can also configure an IDS rule that looks at TGS-REQ packets with the service name “KerberoastHoneyPot” (again, or something more boring and subtle).

Conclusion

Kerberoasting requires requesting Kerberos TGS service tickets with RC4 encryption which shouldn’t be regular activity on a network. Logging 4769 events on Domain Controllers, filtering these events by ticket encryption type (0x17), known service accounts (Account Name field) & computers (Service Name field) greatly reduces the number of events forwarded to the central logging and alerting system. Gathering and monitoring this data also creates a good baseline of what’s “normal” in order to more easily detect anomalous activity.

Detecting Kerberoasting activity is possible by logging the correct activity on Domain Controllers. Determining if this activity is malicious is not requires in-depth knowledge of how RC4 TGS tickets are used in the environment. Creating a service account honeypot with a SPN that doesn’t do anything, provides another data point. Any Kerberos ticket request involving the honeypot service account should be seen as malicious.

Kerberoasting References

- Detecting Kerberoasting Activity (part 1)
- Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain
- Attack Methods for Gaining Domain Admin Rights in Active Directory
- Sneaky Persistence Active Directory Trick #18: Dropping SPNs on Admin Accounts for Later Kerberoasting
- 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).