The Art of the Honeypot Account: Making the Unusual Look Normal

hub.trimarcsecurity.com/post/the-art-of-the-honeypot-account-making-the-unusual-look-normal

Sean Metcalf August 6, 2020

I have had the idea for a post describing how to best create a honeypot (or honeytoken) account for many years and only recently gained enough clarity around how to format and structure such an article for it to be useful. Shout-out to Carlos Perez (@Carlos_Perez) for a recent chat about Kerberoasting and the Detecting Kerberoast article I posted a while back which got me thinking enough about this to start writing.

This article covers how to create accounts used as honeypots (or honeytokens) that look like they provide something an attacker wants (access), but ultimately provides something the defender wants (detection). The focus is making honeypot accounts look normal and "real" in Active Directory and this premise should be somewhat portable to other systems.

AD Recon 101 I have previously covered AD recon in presentations (DEF CON 2016: Beyond the MCSE, Red-Teaming Active Directory), but provide expanded detail here focused on privileged AD account recon. When an attacker is performing reconnaissance of Active Directory, there are a few key items to review:

1. Identify Privileged Accounts
2. Identify Privileged Accounts with Old Passwords
3. Identify Privileged Accounts with Kerberos Service Principal Names (SPNs)
4. Identify Privileged Accounts with Network Sessions on Regular Workstations

In most Active Directory environments, we can scan the AD forest for all of these as a regular AD user (and in some cases without valid AD credentials).

1. Identify Privileged Accounts

Let's start with #1. We can either recursively enumerate the Administrators group in every domain in the AD forest or we can scan all AD user accounts in each domain that have the user attribute "AdminCount" set to 1. I presented about how useful the AdminCount attribute can be in 2015 (DerbyCon – "Red vs Blue Active Directory Attack & Defense").

```
PS C:\> get-aduser -filter {AdminCount -eq 1} -Prop AdminCount |`
 Select Name,Enabled,ObjectClass,AdminCount |`
 Format-Table

Name       Enabled ObjectClass AdminCount
----       ------- ----------- ----------
admALong      True user                  1
admGMoore     True user                  1
```

The AdminCount attribute is automatically set to 1 on any AD accounts that are added to privileged AD groups such as Administrators, Domain Admins, Enterprise Admins, etc. The limiting factor in the usefulness of this technique is that we may also find accounts that used to be in a privileged AD group but no longer a member. This means that scanning for AD accounts with AdminCount=1 provides a quick list of potentially privileged accounts (without group enumeration).

2. Identify Privileged Accounts with Old Passwords

Once we have a list of privileged accounts, we want to check for old passwords. I usually define an “old password” as older than 5 years since Active Directory Security has really only become a focus in the past 5 years or so (since around 2014/2015) for most organizations. Any account with a password older than 5 years is likely not great and any password older than 10 years is probably worse. As an attacker, I am more likely to target accounts with old passwords.

```
Lab.trimarcresearch.com AD Admins:

name          DistinguishedName                                                                  PasswordLastSet
----          -----------------                                                                  ---------------
admMBailey    CN=admMBailey,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com 11/10/2019 11:26:46 PM
admEGray      CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com   11/10/2019 11:27:06 PM
VMWareAdmin   CN=VMWareAdmin,OU=Service Accounts,DC=trimarcresearch,DC=com                      11/10/2019 11:57:14 PM
SharepointSVC CN=SharepointSVC,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com              11/13/2019 9:18:33 AM
Administrator CN=Administrator,CN=Users,DC=trimarcresearch,DC=com                                2/11/2020 2:08:55 PM
Administrator CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com                         5/19/2020 4:32:44 PM
SVC-LAB-GMSA1 CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com     6/10/2020 8:15:07 AM
```

3. Identify Privileged Accounts with Kerberos Service Principal Names (SPNs)

We can also check the list of privileged accounts to see if they have an associated Kerberos Service Principal Name (SPN). For any account with at least one SPN, we can use an attack called “Kerberoast” to potentially crack the password offline. All the attacker needs to do is request a Kerberos service ticket for the SPN (typically using RC4 which uses the NTLM password hash to encrypt the ticket) and save it to our password crack system. The cracking method is to run through potential passwords (including keyboard map/walk type passwords which are simply patterns based on where characters are placed on the keyboard – popular with admins), convert them to NTLM, and attempt to open the service ticket with this NTLM password hash. If we can open the ticket, we have successfully guessed the password. All with only user rights and minimal activity on the corporate network.

4. Identify Privileged Accounts with Network Sessions on Regular Workstations

The final check I will cover in this AD recon crash-course is checking network sessions for privileged accounts on regular workstations. This is a check that has been performed for years and Will (Harmj0y) has covered this extensively in the past (I Hunt SysAdmins). There's an original NT method (NetSessionEnum) that provides any authenticated user ("authenticated" includes accounts that have connected over a trust) the ability to request from Windows servers the accounts that have a session with it (includes account name, computer the account is calling from, session time, etc). This information provides attackers the ability to gather network session information and identify on what computers privileged accounts are being used. With this information, an attacker can identify how to compromise a single computer to gain access to admin credentials and compromise AD. This is one of the reasons why admin systems are critical for protecting administrative accounts.

Privileged Accounts – An Attacker's Perspective

Now that the attacker has a list of privileged AD accounts and has identified potential targets, what sort of checks can the attacker perform to "validate" these accounts and what can a defender do to counter?

From the attacker's perspective, if there is an AD account in Domain Admins with a password that is 15 years old and has an associated SPN, that looks like a winner. Kerberoast that account, get the password and pwn AD from there!

But how could an attacker validate a juicy target (potentially vulnerable account) before attacking it?

There are some key AD user attributes that are updated through the normal use of the account. This includes when the account last logged on, where it last logged on, when the password was last changed, etc. An attacker would want to check the following:

- When was the account created? Accounts created in the past year or so may be suspect.
- When did the account last logon? If the account has not logged on or hasn't logged on since the creation date, this may be a honeypot account.

- When was the password last changed? If the password has not changed since the creation date, this may be suspect.
- Can we determine (NetSessionEnum) where the account is being used? An account missing network session data doesn't mean it's a honeypot.
- If it is an admin account, is there a correlated user account that is active? If there is an admin account called adm-smetcalf, but there is no user account smetcalf and all other AD admin accounts can be correlated in this manner, this may be a honeypot account. I have identified honeypot accounts in real-world environments using this technique. [SEAN NOTE: if you want a Red Team to proceed really carefully in your environment and double-check everything, create a "Sean Metcalf" account in AD. Or so I've been told 😊]
- If there's an associated Kerberos SPN, is the SPN valid? A quick way to "validate" a SPN is to extract the computer name and check AD to see if the computer name is found in the AD forest. If not, this may be suspect (or a really old account).

Some easy ways for an attacker to attempt to discover a honeypot account is to check to see when the account was created and compare that date with the last password change and last logon date. If these are all approximately the same, it's likely this account is fake or inactive.

Here are some queries we can use to check AD account validity:

- Pwdlastset: Date/time in integer8 format when the account password was last set.
- BadPasswordTime: Date/time in integer8 format when the last time a bad password was attempted for an account. This is only tracked on the authenticating Domain Controller and is not replicated.
- LastLogon, LastLogonTimeStamp & LastLogonDate: Attributes that track logon date/time. Only LastLogonTimeStamp is replicated to all DCs in the domain, though only sometimes within the last 2 weeks.
- logoncount: Updates on a Domain Controller every time an account successfully authenticates (attribute is not replicated, so capturing this information reliably requires connecting every DC in the domain to determine total number).
- Logonhours: Configure to control when an account can logon.
- userworkstations (Microsoft says this should not be used anymore): Configure to control where an account can logon.

Building Our Honeypot Account

We want a target that appears desirable to an attacker (aka “bait”) and provides the defender a new detection method where attacker interaction with the target identifies malicious activity is taking place. Ideally, the defender receives sufficient detail to be useful, such as “activity detected on COMP01 computer leveraging the ADMINM3 account at 3:17pm on July 7th, 2020”.

As the defender, we have the benefit of home court advantage; we can configure the environment as we want to provide better fidelity in detecting potential malicious activity.

Now that we have decided to create our honeypot (or honeytoken) account, what makes an Active Directory (AD) account look “real”?

Since we have reviewed the attacker’s perspective on privileged accounts, we have ideas around what interests an attacker. We can use this information in order to make the honeypot account look more desirable and authentic.

We need to ensure that our honeypot account:

- Is not a recently created account: An old account can be repurposed, one that is inactive and never got cleaned up. This “ages” the account and provides some level of legitimacy. An account that is 10 to 15 years old in an AD environment that is 10 to 20 years old will often be assumed as real. While there may be some ways to determine an account has been “repurposed” via AD replication metadata, this is a more complex AD topic and outside the scope of this article.
- Has a password that is old, but only if similar accounts have old passwords: It will look suspicious if this one account has a password that is 7 years old, but all the other AD admin accounts change their passwords every year or two (including service accounts).
- If it is supposed to be a service account, that it really looks like a service account: Service accounts often require all sorts of weird configurations and don’t have the limitation accounts associated with people do. This is usually the easier honeypot account to configure (and doesn’t require an associated regular user account).

- Has logged on at least once (preferably more): Inactive accounts look suspicious, especially when all other accounts logon regularly.Configure a scheduled task on a protected server to logon with this account daily/weekly to add legitimacy.If the honeypot account is supposed to appear inactive (and forgotten), ensure there are multiple logons associated with it, since attackers may check the logoncount attribute (though this attribute is not replicated, so checking multiple DCs would be required to get an accurate count).

- Has an associated user account: This is a key item, especially if the honeypot account is supposed to be an admin account associated with a person.

- Has a bad password attempt: Real accounts have an associated bad password attempt since people make mistakes – even service accounts. Since the attributes that track this are not replicated, this is not a reliable check for an attacker to validate, but if found, may be enough to identify the account as “valid”.

- Looks like the other admin accounts: This is key. If this one account looks like the best target, but it seems really vulnerable while the rest of the environment is pretty tight, that could be a red flag for the attacker and this account may be avoided.

One of the biggest challenges with a honeypot account is to make it a definite target, one the attacker can’t help but go after, and yet limit what the account can do if the attacker can access it (or limit it in some way where it doesn’t provide attacker benefit).

There are a few approaches:

1. Add the honeypot account to a privileged AD group with real rights and ensure it has a long, complex password. An easy way to do this is to open the account, check the user option “Logon with smartcard”, click Apply, and uncheck, Apply. This will randomize the password to something that can’t be guessed since it is long and complex like an AD computer account password.

2. Add the honeypot account to a privileged AD group and provide the ability for the attacker to get a fake password. This can be done with a honeypot account that looks like a service account. Update the info attribute with something like “Vendor password: HPSIMple2005!” and don’t set the password to this.

3. Add the honeypot account to a privileged AD group and provide the ability for the attacker to get the real password (add a SPN which the attacker will Kerboeroast) , but limit the account in some way. LogonWorkstations attribute is probably the easiest way to protect the account since it can only logon to specific computers. LogonHours is another good way to limit logon ability for this account.

Note: If you decide to use LogonWorkstations to limit logon ability for a honeypot account, there is a potential escalation path if the attacker could discover the associated password, and at least 1 of the computer accounts don't exist in AD. Active Directory's default configuration allows any user to add 10 computer accounts to the AD domain (and technically many more since each computer account can add 10). This means that if there isn't a computer account associated with all the values in LogonWorkstations, an attacker could potentially use a compromised user account to create a new computer account and eventually associate it with an unjoined computer ultimately using this newly domain-joined computer account to logon interactively with the honeypot account.

Trimarc Recommendations

Trimarc does not recommend placing an account in a privileged AD group like Domain Admins and leaving valid credentials for this account in AD. If the protection of this account is not perfect, you could end up with AD being compromised due to this configuration.

Create several honeypot accounts in your environment to trip the attacker.Regardless of what you decide to deploy, make sure the accounts look like others in the environment.

Kerberoast Honeypot Account (service account or admin account): Monitor for Kerberos Service Ticket requests for this account.

Group Policy Preference Password Honeypot (not an account, necessarily): Create a random GUID folder name in the SYSVOL share on a DC in each domain and set a SACL (audit entry) on the folder (ensure Domain Controller auditing is configured to enable Object Access - Audit File System in Advanced Audit Policy Configuration). If anyone attempts to read it, that is potentially malicious since it's not associated with a real Group Policy object.

Example GUID: BD5739C3-484F-40EB-CA0B-E88F987FBC63

PowerShell code to create a random GUID is provided at the end of this post.

We can also place some XML files in these GUID folders that contain credentials to see if anyone attempts to use them. Group Policy Preference passwords are stored as "cpassword" in the XML. Here's a sample "groups.xml" file that can be adjusted to place in the SYSVOL GUID Folder structure:

```
<?xml version="1.0" encoding="utf-8" ?>
<Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
<User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="Administrator (built-
in)" image="2" changed="2019-03-17 03:17:23" uid="{D5FE7352-81E1-42A2-B7DA-
118402BE4C33}">
<Properties action="U" newName="TRDAdmin" fullName="" description="Standard Admin
Account" cpassword="RI133B2Wl2CiIOCau1DtrtTe3wdFwzCiWB5PSAxXMDstchJt3bLOUie0BaZ/
7rdQiuqTonF3ZWAKa1iRvd4JGQ" changelogon="0" noChange="0" neverExpires="0"
acctDisabled="0" subAuthority="RID_ADMIN" userName="Administrator (built-in)"
expires="2019-03-16" />
</User>
</Groups>
```

Reference: https://adsecurity.org/?p=2288

Service account with bad password in text attribute (info or description): Monitor for authentication attempts and bad password attempts.

PowerShell code to create a random GUID

```
# GPO GUID Structure: 8-4-4-4-12
Function Get-HexString
 {
 Param
 ([int]$Length)
 (1..$Length | %{ '{0:X}' -f (Get-Random -Max 16) }) -join ''
 }

$HexLengthArray = @(
'8',
'4',
'4',
'4',
'12'
)

$RandomGUIDString = $NULL
ForEach ($HexLengthItem in $HexLengthArray)
 {
 [string]$RandomGUIDString += $(Get-HexString -Length $HexLengthItem) + '-'
 [string]$RandomGUIDString2 += $(Get-HexString -Length $HexLengthItem) + '-'
 }

$RandomGUID = $RandomGUIDString.Substring(0,$RandomGUIDString.Length-1)
$RandomGUID2 = $RandomGUIDString2.Substring(0,$RandomGUIDString2.Length-1)

# Ensure generated GUID doesn't match Microsoft defaults
# Default Domain Policy 31B2F340-016D-11D2-945F-00C04FB984F9
# Default Domain Controllers Policy 6AC1786C-016F-11D2-945F-00C04fB984F9

IF ( ($RandomGUID -eq '31B2F340-016D-11D2-945F-00C04FB984F9') -OR ($RandomGUID -eq
'6AC1786C-016F-11D2-945F-00C04fB984F9') )
 { $RandomGUID2 }
ELSE
 { $RandomGUID }
```

By Sean Metcalf

Trimarc provides leading expertise in security solutions including security reviews, strategy, architecture, and implementation. Our methodology leverages our internal research and custom tooling which better discovers multiple security issues attackers could exploit to compromise the environment. Trimarc security services fit between traditional compliance/audit reviews and standard penetration testing/red teaming engagements, providing deep understanding of Microsoft and Virtualization technologies, typical security issues and misconfigurations, and provide recommendations based on our own best practices custom-tailored to balance operational and security challenges.

#ADSecurity #ActiveDirectorySecurity #AD #Honeypot #HoneyPotAccount #ADRecon #ADHoneyPot