Running LAPS in the Race to Security

blog.netwrix.com/2021/08/25/running-laps-in-the-race-to-security

Kevin Joyce

Microsoft Local Administrator Password Solution (LAPS) enables organizations to manage local Administrator passwords across all their endpoints. When implemented correctly, it is an effective way to prevent some types of potential lateral movement and privilege escalation within your environment — but when implemented incorrectly, it can create a large opening for attackers.

Handpicked related content:
    [Free Guide] Privileged Access Management Best Practices

This article explains what LAPS is and how to deploy it, as well as how to use Netwrix Privilege Secure to overcome its inherent security gaps.

What is Microsoft LAPS?

Microsoft LAPS is a password manager that utilizes Active Directory to manage and rotate passwords for local Administrator accounts across all of your Windows endpoints. By ensuring that all local Administrator accounts have unique, complex passwords, LAPS helps mitigate the risk of lateral movement and privilege escalation: An attacker who compromise one local Administrator account can't move laterally to other endpoints simply by using the same password.

A benefit over other password managers is that LAPS does not require additional computers to manage these passwords; it's done entirely through Active Directory components. Plus, you can download and use LAPS for free.

Setting up Microsoft LAPS

To set up LAPS in your environment, follow the steps detailed in the guide (LAPS_OperationsGuide.docx) included in the download. Here, I'll walk through some of these steps at a high level.

Installing LAPS on your endpoints

When you install LAPS on an endpoint, the following components are laid down:

- Fat client UI
- PowerShell module
- Group Policy templates
- AdmPwd GPO extension

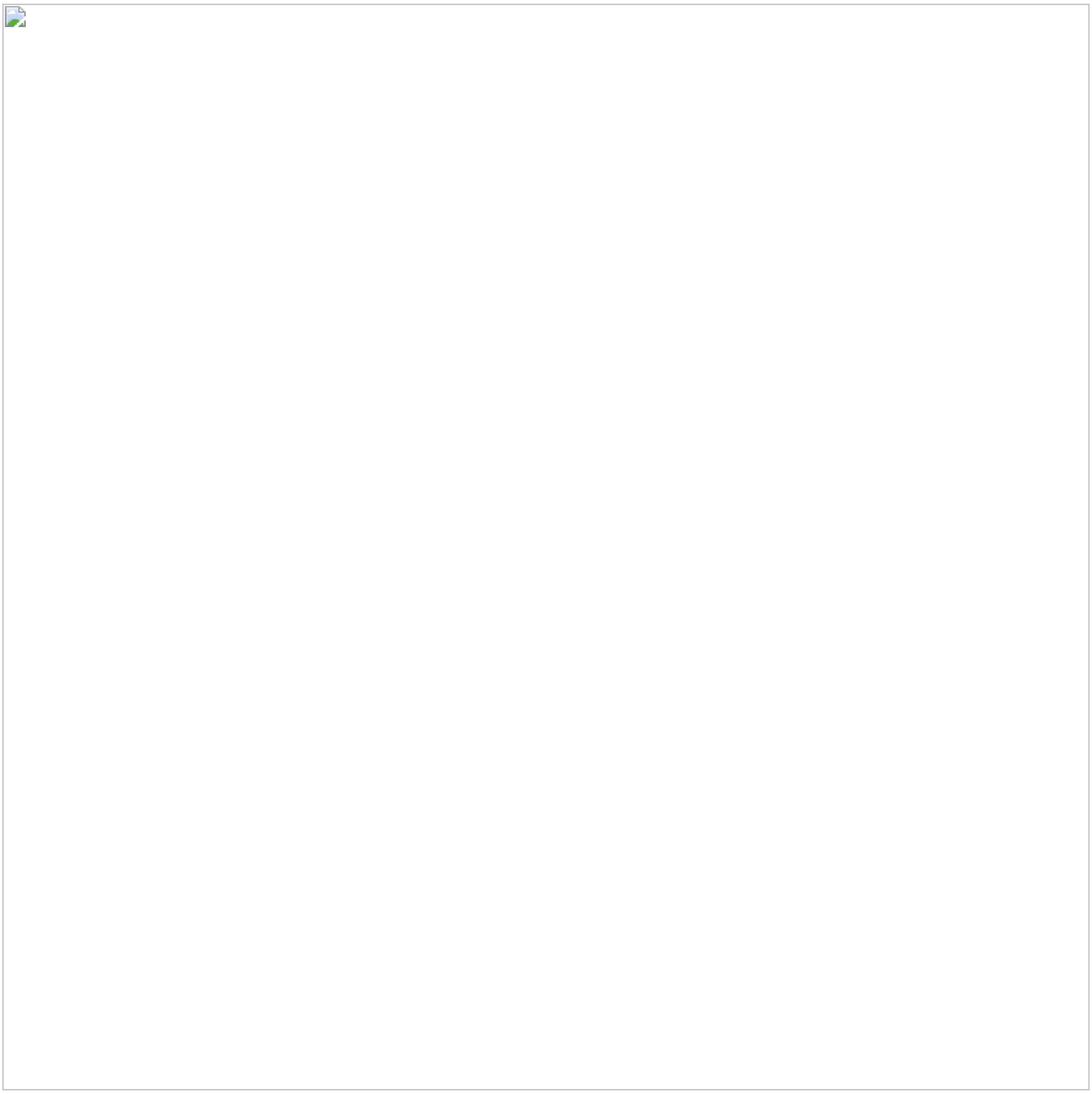
While management consoles need one or more of these features, the endpoints to be governed by LAPS need only the AdmPwd GPO extension. The Operations Guide offers a few ways to push this component to your endpoints.

Extending your Active Directory schema to accommodate LAPS

Microsoft provides a PowerShell module to help you with this step. It adds two computer attributes to your schema:

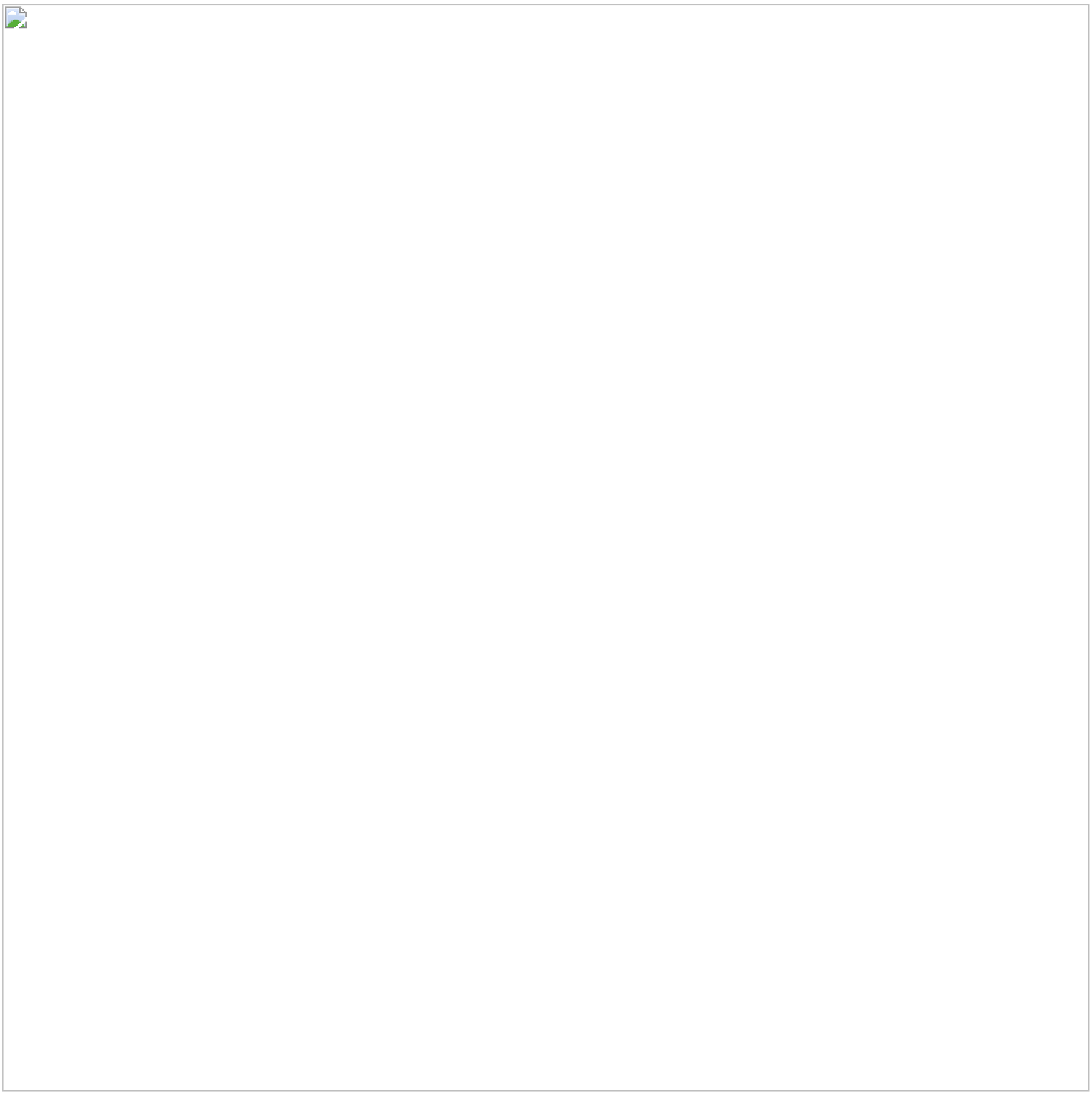
- **ms-Mcs-AdmPwd** — Stores the local Administrator password for the computer object in clear text (scary, I know, but I'll expand on this later)
- **ms-Mcs-AdmPwdExpirationTime** — Stores the time when the password expires

Once the schema is extended, you can use Group Policy to push out the configuration of LAPS to all your member servers. Then, use the Group Policy Management Editor to configure the new LAPS Group Policy object (GPO):



The LAPS GPO includes the following settings:

- **Password Settings** — Enables you to specify the complexity requirements for the passwords for local Administrator accounts, including length and age



- **Enable local admin password management** — Controls whether the endpoints governed by the GPO are being managed by LAPS
- **Name of administrator account to manage** — Allows you to manage password for non-default local administrator accounts (that is, local accounts not named ‘Administrator’)
- **Do not allow password expiration time longer than required by policy** — Allows you to ensure that no password expiration exceeds the ‘Password Age (Days)’ setting in Password Settings

Ensuring that LAPS is secure

After you’ve extended your AD schema, the next step is to ensure that the permissions to these new attributes are applied correctly. This is one of the most important steps in the process, since you want to grant access only to objects that need it. Microsoft provides PowerShell scripts to check who currently has access to the attribute, and to apply new permissions if needed.

At a high level, there a few considerations to keep in mind here:

- Remove the ‘All extended rights’ permission from users and groups on computer objects that shouldn’t be able to view the password of the local Administrator accounts (remember it’s stored in clear text in an attribute).

- The ‘SELF’ principal requires the capability to write the ‘ms-Mcs-AdmPwdExpirationTime’ and ‘ms-Mcs-AdmPwd’ attributes so it can update the passwords and expiration times when a password expires. The ACE for ‘SELF’ is required on all the computer objects governed by LAPS.
- Users and groups that are allowed to reset passwords on the local Administrator accounts must be granted access to the attributes on those computer objects.

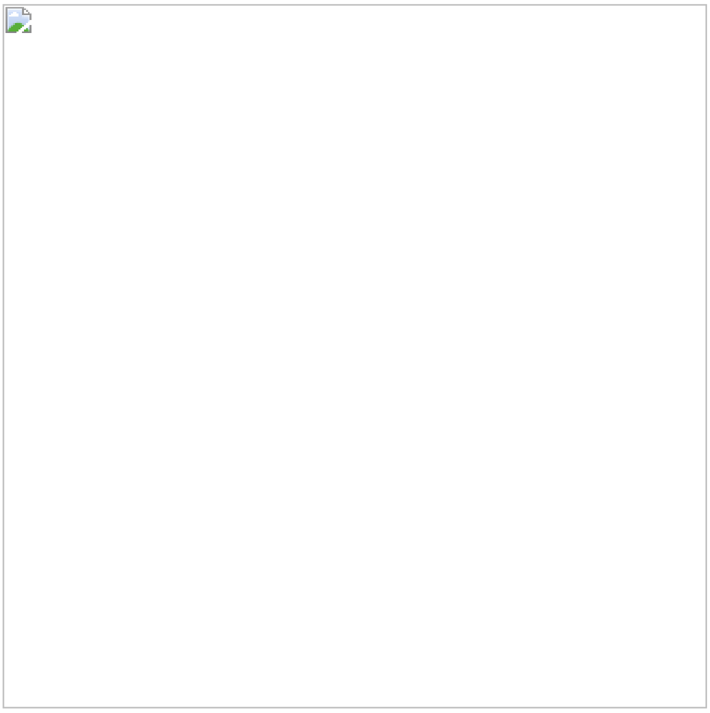
Here is a script that will help you understand who has access to these attributes today.

```
<#
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 computer objects)

Description: Looks up permissions within Active Directory on a target (OU or Computer) to determine access
to LAPS attributes (ms-Mcs-AdmPwdExpirationTime and ms-Mcs-AdmPwd).

Usage: Populate the $target variable with the DN of a computer object, or OU to search for computer
objects within.

To output the results to a text file run the following .LAPS_Permissions_Collection.ps1 > output.txt
<#

Import-Module ActiveDirectory
##Get the GUID of the extended attributes ms-Mcs-AdmPwdExpirationTime and ms-Mcs-AdmPwd from Schema
$schemaIDGUID = @{}
Get-ADObject-SearchBase (Get-ADRootDSE).schemaNamingContext -LDAPFilter '(|(name=ms-Mcs-
AdmPwdExpirationTime)(name=ms-Mcs-AdmPwd))' -Properties name, schemaIDGUID |
ForEach-Object {$schemaIDGUID.add([System.GUID]$_.schemaIDGUID,$_.name)}



<# **REPLACE DN VARIABLE BELOW**
Declare the distinguishedName of the Computer object directly or OU to search for computers within#>
$target ='CN=Computers,DC=COMPANY,DC=NET'



##Get distinguished name of all Computer objects from the OU or of the target itself
$computers =Get-ADComputer-SearchBase $target -Filter {name -like '*'}



<#Get objects that have specific permissions on the target(s):

Full Control(GenericAll)
Read All Properties(GenericRead)
Write all Properties (WriteProperty where ObjectType = 00000000-0000-0000-0000-000000000000

#>
Set-Location ad:
foreach ($computer in $computers){
(Get-Acl $computer.distinguishedname).access |
Where-Object { (($_.AccessControlType -eq 'Allow') -and ($_.activedirectoryrights -in
('GenericRead','GenericAll') -and$_.inheritancetype -in ('All', 'None')) -or (($_.activedirectoryrights -
like '*WriteProperty*')-or ($_.activedirectoryrights -like '*GenericRead*') -and ($_.objecttype -eq
'00000000-0000-0000-0000-000000000000')))} |
ft ([string]$computer.name),identityreference, activedirectoryrights, objecttype, isinherited -autosize
}
<#Get objects that have specific permissions on the target(s) and specifically the LAPS attributes:

 WriteProperty
ReadProperty
#>
Set-Location ad:
foreach ($computer in $computers){
(Get-Acl $computer.distinguishedname).access |
Where-Object {(($_.AccessControlType -eq 'Allow') -and (($_.activedirectoryrights -like '*WriteProperty*')
-or ($_.activedirectoryrights -like '*ReadProperty*')) -and ($_.objecttype -in $schemaIDGUID.Keys))} |
ft ([string]$computer.name),identityreference, activedirectoryrights, objecttype, isinherited -AutoSize
}
```

Security tip: In addition to locking down these two LAPS attributes via permissions, you should also monitor LDAP traffic against the attributes, so you can spot attackers querying them when misconfigured permissions allow. StealthDEFEND for Active Directory can provide real-time alerting on LDAP traffic against these attributes.

Seeing LAPS in action

There are a few ways to get the current local Administrator password for a particular endpoint through LAPS, but the easiest way is to use the GUI. Simply enter a computer name and it will show the plaintext local Administrator password along with its expiration date, which you can modify.



Alternatively, you can get this information using PowerShell:

Overcoming the security gaps in Microsoft LAPS with Netwrix Privilege Secure

Microsoft LAPS is a powerful solution for managing the local Administrator passwords across all of your endpoints. When implemented correctly, it is an effective way to prevent some forms of potential lateral movement or privilege escalation. Unfortunately, however, when it is not implemented correctly, it can create a very large opening for attackers.

LAPS security gaps include the following:

- The dated interface
- LAPS does not support strong authentication and authorization.
- LAPS does not provide just-in-time access, so Administrator accounts are always vulnerable to being taken over by attackers.
- There is no forced password reset after access, enabling attackers to use a stolen password.
- Users always have to know the account password, which imposes both logistical and security issues.

You can eliminate these security gaps using Netwrix Privilege Secure . This privileged access management (PAM) solution enhances LAPS by adding features such as multi-tier approval, just-in-time and scheduled access, forced password rotation, controlled exposure of password, and session recording and playback.

Other benefits of Netwrix Privilege Secure

Most PAM solutions focus solely on limiting access to highly privileged accounts like Domain Admin and local server Administrator. While this approach provides administrators with just-in-time access, all the powerful accounts remain available — and therefore vulnerable — even when they are not in use.

Netwrix Privilege Secure enables you to slash your attack surface area by eliminating most of these standing privileged accounts. Instead, administrators are provided with the exact level of rights required to complete a task, at the exact time they are required, and for only as long as they are required. This approach also empowers you to reduce the number of people who need to be members of Domain Admins, further reducing security risks. In addition, the solution removes or minimizes artifacts that are often used to breach accounts.

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.