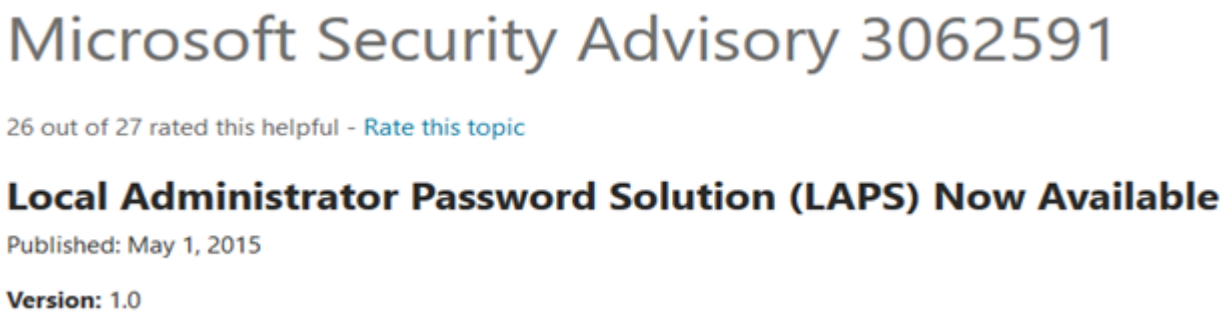
Running LAPS Around Cleartext Passwords

Intro

Managing credentials for local administrator accounts is hard to do. From setting strong passwords, to setting unique passwords across multiple machines, we rarely see it done correctly. On the majority of our pen tests we see that most of the domain computers are configured with the same local admin credentials. This can be really handy as an attacker, as it provides us lateral access to systems across the network.

One of the reported fixes (from Microsoft) is to store the local admin passwords in LDAP as a confidential attribute of the computer account. This can be automated using Microsoft tools and strong local passwords can be enforced (and automatically changed). In theory, this is a nice idea. But in practice it results in the cleartext storage of passwords (not good). Previous attempts at local administrator credential management (from Microsoft) resulted in local administrator credentials being exposed to all users on the domain (through group policy preferences). The GPP cpassword storage passwords issue was fixed (5/13/14) and we're not seeing it as frequently any more.

LAPS


Microsoft Security Advisory 3062591

26 out of 27 rated this helpful - Rate this topic

Local Administrator Password Solution (LAPS) Now Available

Published: May 1, 2015

Version: 1.0

LAPS is Microsoft's tool to store local admin passwords in LDAP. As long as everything is configured correctly, it should be fine to use. But if you don't set the permissions correctly on the LDAP attributes, you could be exposing the local admin credentials to users on the domain. LAPS uses two LDAP attributes to store the local administrator credentials, ms-MCS-AdmPwd (stores the password) and ms-MCS-AdmPwdExpirationTime (stores when it expires). The Microsoft recommendations says to remove the extended rights to the attributes from specific users and groups. This is a good thing to do, but it can be a pain to get set up properly. Long story short, if you're using LAPS, someone on the domain should be able to read those local admin credentials in cleartext. This will not always be a privilege escalation route, but it could be handy data to have when you're pivoting to sensitive systems after you've escalated. In our demo domain, our LAPS deployment defaulted to allowing all domain users to have read access to the password. We also could have screwed up the install instructions.

I put together a quick PowerShell script to pull the LAPS specific LDAP attributes for all of the computers joined to the domain. I used Scott Sutherland's Get-ExploitableSystems script (now included in PowerView) as the template. You can find it on my GitHub page.

Script Usage and Output

Here's the output using an account that does not have rights to read the credentials (but proves they exist):

```
PS C:\> Get-LAPSPasswords -DomainController 192.168.1.1 -Credential DEMO\karl |
Format-Table -AutoSize

Hostname                    Stored Readable Password Expiration
--------                    ------ -------- -------- ----------
WIN-M8V16OTGIIN.test.domain   0      0               NA
WIN-M8V16OTGIIN.test.domain   0      0               NA
ASSESS-WIN7-TEST.test.domain  1      0               6/3/2015 7:09:28 PM
```

Here's the same command being run with an account with read access to the password:

```
PS C:\> Get-LAPSPasswords -DomainController 192.168.1.1 -Credential
DEMO\administrator | Format-Table -AutoSize

Hostname                    Stored Readable Password       Expiration
--------                    ------ -------- --------       ----------
WIN-M8V16OTGIIN.test.domain   0      0                      NA
WIN-M8V16OTGIIN.test.domain   0      0                      NA
ASSESS-WIN7-TEST.test.domain  1      1      $sl+xbZz2&qtDr 6/3/2015 7:09:28 PM
```

The usage is pretty simple and everything is table friendly, so it's easy to export to a CSV.

Special thanks to Scott Sutherland for letting me use his Get-ExploitableSystems script as the bones for the script. The LDAP query functions came from Carlos Perez's PoshSec-Mod (and also adapted from Scott's script). And the overall idea to port this over to a Powerview-friendly function came from a conversation with @_wald0 on Twitter.

Links

- http://blogs.technet.com/b/askpfeplat/archive/2014/05/19/how-to-automate-changing-the-local-administrator-password.aspx
- https://code.msdn.microsoft.com/Solution-for-management-of-ae44e789
- https://technet.microsoft.com/en-us/library/security/3062591
- http://www.obscuresecurity.blogspot.com/2012/05/gpp-password-retrieval-with-powershell.html

LDAP is a great place to mine for sensitive data. Here's a couple of good examples:

- Custom Properties and Descriptions in AD
- Faster Domain Escalation using LDAP

Bonus Material

If you happen to have the AdmPwd.PS PowerShell module installed (as part of LAPS), you can use the following one-liner to pull all the local admin credentials for your current domain (assuming you have

the rights):

```
foreach ($objResult in $colResults){$objComputer = $objResult.Properties;
$objComputer.name|where {$objcomputer.name -ne $env:computername}|%{foreach-
object {Get-AdmPwdPassword -ComputerName $_}}}
```