```
Authentication Id : 0 ; 294625 (00000000:00047ee1)
Session           : Interactive from 1
User Name         : Administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 2/1/2016 6:21:21 AM
SID               : S-1-5-21-1100472043-2579244664-397435893
        msv :
         [00010000] CredentialKeys
         * NTLM     : 1543a4536a25d208e652dba231e73cdd
         * SHA1     : 9621d462145820990 5b31ed96fe8f59d899b4c
         [00000003] Primary
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * NTLM     : 1543a4536a25d208e652dba231e73cdd
         * SHA1     : 9621d462145820990 5b31ed96fe8f59d899b4c
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : weakpass1
        kerberos :
         * Username : Administrator
         * Domain   : TESTDOMAIN.LOCAL
         * Password : weakpass1
        ssp :
        credman :
```

February 14, 2016

Defending Against Mimikatz

Intro to Mimikatz

One of the most interesting tools in a

CATEGORIES

CRYPTO

penetration tester's arsenal is mimikatz. Mimikatz is a tool that scrapes the memory of the process responsible for Windows authentication(LSASS) and reveals cleartext passwords and NTLM hashes that an attacker can use to pivot around a network. From that point they escalate privilege either by authenticating with the clear text credentials or passing the hash. Sounds deadly right? Most people have the reaction "Why hasn't Microsoft come up with a solution to this?".

If you Google the phrase "defending against mimikatz" the information you find is a bit lackluster. The best article I have found was this one. It has a lot of good suggestions like using the "Protected Users" group(SID: S-1-5-21-<domain>-525) available in recent versions of Active Directory and also limiting administrator usage, and taking advantage of not storing passwords in memory with a registry

PEN TESTING

WINDOWS

TAGS

ACTIVE DIRECTORY

AD

MIMIKATZ

NTLM

WDIGEST

WINDOWS

RECENT POSTS

Defending Against Mimikatz

OpenSSL Vs HSM Performance

Going A2DP only on Linux

SSH and X forwarding

Decrypting TLS Browser Traffic With Wireshark – The Easy

passwords in memory with a registry setting. You can limit the number of services running as system or remove debug privilege to help prevent an attacker from being able to run mimikatz. What this and other articles make you believe is that you need to have Windows 8 or 8.1 or 10 rolled out everywhere. What about the large number of Windows 7/2008 R2 machines out there? Well it turns out you can defend against mimikatz on these versions of Windows, here is how.

Step One: Active Directory 2012 R2 Functional Level

The first thing that you can do is upgrade the schema and functional level of your forest and domain(s) to 2012 R2. This domain functional level adds a new group called "Protected Users". If you read the TechNet article on Protected Users you might get the feeling that this is the thing that will make mimikatz password

Way!

CATEGORIES

apache
crypto
Disclosure
linux
mac
nginx
Pen Testing
Uncategorized
VPN
windows

TAGS

active directory AMISHA1apacheAPC BrowserscadgerCDN certificatesciphercookie cryptodisclosureEJBCA

scraping impossible against a Protected User. But what does that look like?

```
Authentication Id : 0 ; 1327833 (00000000:001442d9)
Session           : Interactive from 2
User Name         : Administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 1/31/2016 12:51:07 PM
SID               : S-1-5-21-1100472043-2579244664-397
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * NTLM     : 1543a4536a25d208e652dba231e73cdd
         * SHA1     : 9621d46214582097705b31ed961e81570
         [00010000] CredentialKeys
         * NTLM     : 1543a4536a25d208e652dba231e73cdd
         * SHA1     : 9621d46214582097705b31ed961e81570
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : (null)
        kerberos :
         * Username : Administrator
         * Domain   : TESTDOMAIN.LOCAL
         * Password : (null)
        ssp :   KO
        credman :
```

This is pretty standard mimikatz output, notice that NTLM hashes are visible Protected User

firesheepfull disclosure
googleHard Drivehttps
iMacldap linux mac
Mythsnginxopen source
OpenSSLopenvpn
PHPprojectsresponsible
disclosuresambaSHA-1SHA1
sheepstripSSD SSL
sslstripsuites TLS
upgradesvarnishvpnweb
windowswww

```
Authentication Id : 0 ; 144339 (00000000:000233d3)
Session           : Interactive from 1
User Name         : Administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 1/31/2016 10:54:46 AM
SID               : S-1-5-21-1100472043-2579244664-397
        msv :
         [00010000] CredentialKeys
         * RootKey  : 3d209d9c7e8dd2a68c9bb01c44fa4786
929004
         * DPAPI    : 514e5c8e20264c64b7de758dd8541717
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : (null)
        kerberos :
         * Username : Administrator
         * Domain   : TESTDOMAIN.LOCAL
         * Password : (null)
        ssp :   KO
        credman :
```

And when a user is added to the Protected Users group we see that there aren passwords.

So clearly the Protected Users group works. But what happens when a user is in the Protected Users group on a Windows 7 or 2008 R2 box?

```
Authentication Id : 0 ; 93291 (00000000:00016c6b)
Session           : Interactive from 1
User Name         : administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 1/31/2016 11:08:02 AM
SID               : S-1-5-21-1100472043-2579244664-397
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * LM       : [illegible]
         * NTLM     : 1543a4536a25d208e652dba231e73cdd
         * SHA1     : [illegible]
        tspkg :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : Weakpass1
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : Weakpass1
        kerberos :
         * Username : administrator
         * Domain   : [illegible].LOCAL
         * Password : Weakpass1
        ssp :
        credman :
```

Notice that even in the Protected Users group the passwords and hash is visib

This machine is unpatched though. If you think about it, the group "Protected Users" is meaningless to the computer without it knowing what being in that group means. Fortunately Microsoft backported the functionality of Windows 8.1 and 2012 R2 to older versions of Windows.

Step Two: Install KB2871997

So if you have been keeping up with your Windows updates, as you should, then KB2871997 has already been installed. This is the update that backports the functionality to older, but still supported versions of Windows. Once we install the update we see 2008 R2, as an example behaves in a similar matter.

```
Authentication Id : 0 ; 294625 (00000000:00
Session           : Interactive from 1
User Name         : Administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 2/1/2016 6:21:21 AM
SID               : S-1-5-21-1100472043-257
        msv :
         [00010000] CredentialKeys
         * NTLM     : 1543a4536a25d208e652d
         * SHA1     : 9621d46214582099 05b31
         [00000003] Primary
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * NTLM     : 1543a4536a25d208e652d
         * SHA1     : 9621d46214582099 05b31
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : Weakpass1
        kerberos :
         * Username : Administrator
         * Domain   : TESTDOMAIN.LOCAL
         * Password : Weakpass1
        ssp :
        credman :
```

This is what things look like on 2008 R2/Win 7 box post update KB2871997 w

```
Authentication Id : 0 ; 564212 (00000000:00
Session           : Interactive from 2
User Name         : administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 2/1/2016 6:43:15 AM
SID               : S-1-5-21-1100472043-257
        msv :
         [00010000] CredentialKeys
         * RootKey  : 3d209d9c7e8dd2a68c9bb
929004
         * DPAPI    : 514e5c8e20264c64b7de7
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : (null)
        kerberos :
         * Username : administrator
         * Domain   : TESTDOMAIN.LOCAL
         * Password : (null)
        ssp :
        credman :
```

Once in the Protected Users group we see the same behavior on 2008 R2/Win

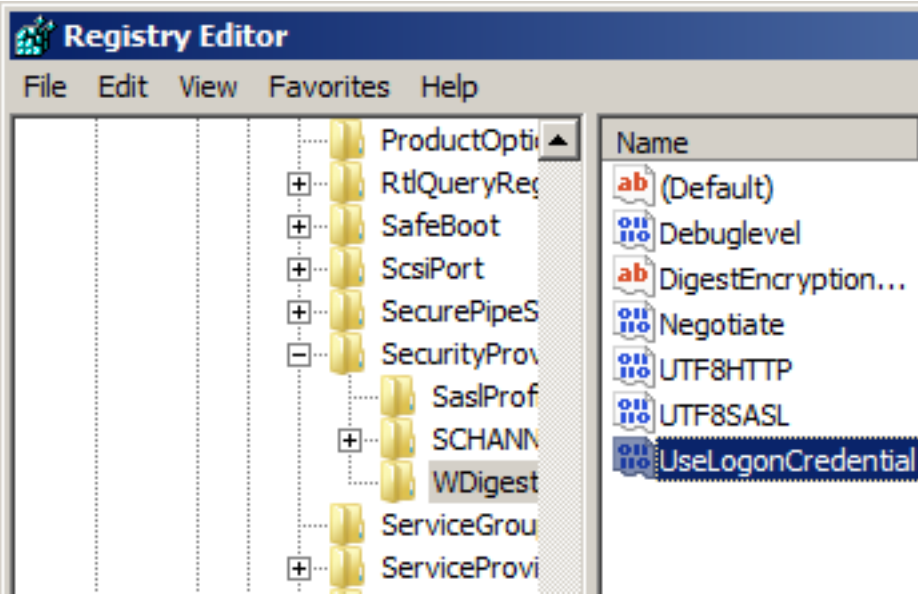
Step Three: Eliminate Passwords Storage in Memory

So this step is probably optional, as any account you care about you are going to want to make a Protected User, however you won't be able to do this with every account in your domain. Microsoft itself

recommends against putting computer accounts and service accounts into the Protected Users group. So this step strictly relates to users who are not in the Protected Users group.

If you have a keen eye, you will have no doubt noticed that in the screenshots of 2012 R2 the password is never revealed by mimikatz while under 2008 R2 the password is revealed when not a protected user. The storage of passwords in memory is governed by a registry setting. Just like the Protected Users group functionality, password storage in memory is disallowed in newer versions of Windows(8.1+ & 2012 R2+) by default. Also like the Protected Users group functionality, password storage in memory was backported in the same KB2871997 update. Unfortunately even after the update these older versions of Windows still default to storing the password in memory by default, you know because compatibility. Simply set

the “UseLogonCredential” registry setting, at the path below, to ‘0’ and you are golden.



HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProv

```
Authentication Id : 0 ; 90932 (00000000:000
Session           : Interactive from 1
User Name         : administrator
Domain            : TESTDOMAIN
Logon Server      : WIN-12UU57SPIN9
Logon Time        : 2/1/2016 6:37:50 AM
SID               : S-1-5-21-1100472043-257
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * NTLM     : 1543a4536a25d208e652d
         * SHA1     : 9621d46214582097905b31
         [00010000] CredentialKeys
         * NTLM     : 1543a4536a25d208e652d
         * SHA1     : 9621d46214582097905b31
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : TESTDOMAIN
         * Password : (null)
        kerberos :
         * Username : administrator
         * Domain   : TESTDOMAIN.LOCAL
         * Password : (null)
        ssp :
        credman :
```

This user is not in the Protected Users group but is logged into a machine whe memory as we would on 2012 R2.

Conclusion

So essentially, update Active Directory functional level to 2012 R2, keep up with Windows Update, put important accounts into the Protected Users group and set a registry setting. Also don't give account

more admin rights than they need. I hope this post finally serves as a unified perspective on how to best defend against this. Another thing to mention about the Protected Users functionality is that members of that group are protected against Kerberos related "Golden Ticket" attacks as Kerberos tickets for Protected Users go from potentially valid for up to 10 years, to 4 hours. As always you should test this functionality before rolling it out as applications that do not support Kerberos authentication are going to break. Happy hacking.

EDIT: I want to thank everyone for the great feedback to the article. As @Iansus pointed out ADSecurity talked about this a while ago. I'm going to leave my post up, as it is a more hands-on, cause and effect post.

Share this:

LinkedIn 138 | Reddit | Facebook 144 | Twitter | Google+

Like this:

Loading...

Like

One blogger likes this.

Setting up an Active Directory Domain Controller using Samba 4 on Ubuntu 14.04
July 13, 2014
In "linux"

What is the Best Open Alternative to Active Directory Certificate Services?
July 21, 2014
In "crypto"

What Did We Learn From Firesheep and SSLStrip?
January 10, 2015
In "crypto"

2 thoughts on “Defending Against Mimikatz”

Martin Handl says:
February 15, 2016 at 4:26 am

sekurlsa::pth /name:administrator /domain:domain.tld /ntlm:
starts a cmd.exe with the given identity.
Any action done throught the given identity will be done with this given identity.

If the NTLM-Hash is not stored in the lsacache it could be obtained through the AD database.

As far as I know there is actually no real protection –

even with up to date hotfixes.

See also: Bastion Forest

Reply

Jim Shaver says:
February 16, 2016 at 7:25 am

I used the domain Administrator account for simplicity in a test domain, but probably should have minted a generic account for demonstration purposes. You won't always be lucky enough to pop a domain admin or pop him while he is logged into a domain controller. These mitigations are about

helping prevent someone from getting to domain admin.

Reply

Leave a Reply

Enter your comment here...

PREVIOUS POST
OpenSSL Vs HSM Performance