How to Detect Pass-the-Hash Attacks

blog.netwrix.com/2021/11/30/how-to-detect-pass-the-hash-attacks

Jeff Warren

Attackers frequently rely on lateral movement techniques to infiltrate corporate networks and obtain privileged access to credentials and data. In particular, one common technique is pass-the-hash: Hackers use stolen password hashes to authenticate as a user without ever having the user's cleartext password. This tactic enables them to bypass normal system access controls to move laterally within the environment.

This post explains exactly what to look for in the native Windows event logs to detect pass the hash, and offers additional options for spotting — and even preventing — these attacks.

Getting a Baseline: Understanding the Events Logged during the Normal NTLM Authentication Process

Pass the hash relies on NTLM authentication, so we need to first understand what events are normally generated during normal NTLM logon activity.

Authenticating as an Administrative User

To generate these events, I launch a new command prompt as an administrative user, using the account's actual password:

Next, I used the Sqlcmd utility to connect to a Microsoft server by its IP address. This command will generate NTLM authentication to the SQL database:

```
Sqlcmd –S [IP ADDRESS]
```

For good measure, I will also run the SELECT SYSTEM_USER command to show the user I am authenticated as:

Handpicked related content:

[SysAdmin Magazine] Practical Guidance for Fighting IT Monsters

Reviewing the Events Generated

Now let’s see what native Windows events were logged. That gives us a baseline for normal NTLM authentication behavior that does not involve pass the hash.

Workstation Logs

On my local workstation, I will see the following events:

4648 – A logon was attempted using explicit credentials.

4624 – An account was successfully logged on.

A 4624 event was logged with a Logon Type of 2, which means an interactive logon. This aligns with the way I used **runas** and entered my credentials interactively.

4672 – Special privileges assigned to new logon.

Because the account I used (Franklin.Bluth) is an administrative account, event 4672 gets logged to show what privileges are being assigned. This is a useful way to track the activity of administrative accounts.

Target Server Logs

On my SQL server, I see the following events:

> **4624 – An account was successfully logged on.**

On the SQL Server, there is a similar 4624 event; however, the Logon Type is 3, indicating a network logon.

The details show that the Authentication Package was NTLM, which confirms that we are performing NTLM authentication.

4672 – Special privileges assigned to new logon.

Because we used a privileged account, we also see a 4672 event, as illustrated earlier in the description of the workstation logs.

Domain Controller Logs

On the domain controller, we will find artifacts of both Kerberos and NTLM authentication. The Kerberos authentication, which is the default authentication method for Active Directory, happens first. That generated two events:

4768 – A Kerberos authentication ticket (TGT) was requested.

4769 – A Kerberos service ticket was requested.

Once the TGT is received, a TGS was requested for the host. With this, the user Franklin Bluth can now interact with the PC to launch the command prompt.

4776 – The computer attempted to validate the credentials for an account.

The 4776 event is specific to NTLM and will come last. It occurs when we execute the Sqlcmd command to force NTLM authentication.

Detecting Pass the Hash: Understanding Events Logged during an Attack

Now, let’s take a look at what events are generated when we use pass the hash to authenticate.

Authenticating using Pass the Hash

I can easily get the NTLM hash for the Franklin Bluth account from memory with this Mimikatz command:

```
sekurlsa::logonpasswords
```

Then I authentication using pass the hash with the following command:

```
Sekurlsa::pth /user:Franklin.Bluth /ntlm:[ntlm] /domain:jefflab.local
```

A new command window will open. By using the same Sqlcmd command to connect to the IP address of my SQL Server, we can see that I am now authenticated there as Franklin Bluth:

Reviewing the Events Generated

Let’s take a look at what events were generated by this pass-the-hash authentication.

Workstation Logs

On my local workstation, I will see the same events as for the legitimate NTLM authentication (4648, 4624 and 4672). However, there are a few key differences.

> **4624 event** — This event now has a Logon Type of 9, which is NewCredential. This was identified by a security researcher, and I reliably reproduced it in my lab. It a very useful way to identify that a pass-the-hash event took place.

Logon Type 9 is very rare. However, I was able to generate some false positives running applications that use impersonation. The main difference to key off of is the Logon Process will always be “seclogo” for pass the hash (from my tests), so you can filter on that to reduce false-positive rates.

4672 event— In the normal authentication scenario, this event identified a privileged logon for the Franklin Bluth account. With pass-the-hash authentication, it registers the user that I am logged into my workstation as.

Target Server Logs

The logs on the SQL server are identical to those we saw doing legitimate NTLM authentication:

- 4624 – An account was successfully logged on. Logon Type 3, NTLM
- 4672 – Special privileges assigned to new logon.

Domain Controller Logs

On the domain controller, the key difference is that you will **not** see Kerberos authentication. However, that isn’t a very reliable way to detect pass the hash because it can happen for lots of valid reasons, including authentications originating from non-trusted domains.

Summary of Event Logs for Normal and Pass-the-Hash Authentication

Here’s a summary of the native Windows event logs we see when performing normal NTLM authentication:

Source Host	Target Host	Domain Controller

• 4648 – A logon was attempted using explicit credentials. • 4624 – An account was successfully logged on. Logon Type 2 • 4672 – Special privileges assigned to new logon.	• 4624 – An account was successfully logged on. Logon Type 3, NTLM • 4672 – Special privileges assigned to new logon.	• 4768 – A Kerberos authentication ticket (TGT) was requested. • 4769 – A Kerberos service ticket was requested. • 4776 – The computer attempted to validate the credentials for an account.

And here is a summary of what we see when doing pass the hash, with the key differences bolded:

Source Host	**Target Host**	**Domain Controller**
• 4648 – A logon was attempted using explicit credentials. • 4624 – An account was successfully logged on. **(Logon Type 9; Logon Process "Seclogo")** • 4672 – Special privileges assigned to new logon. **(Logged-on user, not impersonated user)**	• 4624 – An account was successfully logged on. Logon Type 3, NTLM • 4672 – Special privileges assigned to new logon.	4776 – The computer attempted to validate the credentials for an account.

Detecting Pass the Hash using Sysmon

To conclusively detect pass-the-hash events, I used Sysmon, which helps to monitor process access events. With Sysmon in place when a pass the hash occurs, you will see Event ID 10 showing access to the LSASS process from Mimikatz (or other pass-the-hash tool).

Building Detections for Pass the Hash

Now that we've looked at all the evidence, the simplest way to build detections for pass the hash is to look for:

- 4624 events on your workstations with:
  - Logon Type = 9
  - Authentication Package = Negotiate
  - Logon Process = seclogo
- Sysmon 10 events for LSASS process access

With a custom event log filter, you can easily see when these two things happen at the same exact time, which indicates pass-the-hash activity on your network.

Here is a custom event filter you can use to surface that specific information.

```
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">
     *[System[(EventID='4624')]
      and
     EventData[Data[@Name='LogonType']='9']
      and
     EventData[Data[@Name='LogonProcessName']='seclogo']
     and
     EventData[Data[@Name='AuthenticationPackageName']='Negotiate']
     ]
     </Select>
  </Query>
  <Query Id="0" Path="Microsoft-Windows-Sysmon/Operational">
    <Select Path="Microsoft-Windows-Sysmon/Operational">
    *[System[(EventID=10)]]
    and
    *[EventData[Data[@Name='GrantedAccess'] and (Data='0x1010' or Data='0x1038')]]
</Select>
  </Query>
</QueryList>
```

How Netwrix Solutions Can Help

Setting up and monitoring custom event filters is tedious, and it requires enabling logging on all endpoints. To simplify the work while leveraging more advanced techniques, consider a third-party threat detection solution.

Netwrix StealthDEFEND is an effective tool for detecting pass-the-hash attacks. Here are two techniques that the solution supports:

- **Honey tokens** — You can inject fake credentials into LSASS memory on target machines and monitor for the usage of those credentials. If you see the credentials in use, you know they were retrieved from memory on one of the honeypot machines and used for lateral movement.
- **Abnormal behavior detection** — Baselining normal user behavior helps you spot anomalous use of accounts that is indicative of pass-the-hash and other lateral movement attacks. Behavior to look for includes:
    - An account is used from a host it never authenticated before
    - An account is used to access a host it never before accessed
    - An account accessing a large number of hosts across the network in a way that contradicts normal access patterns

To mitigate the risk of pass-the-hash attacks being launched in the first place, use Netwrix StealthAUDIT, which empowers you to:

- Minimize administrative rights on servers and desktops
- Prevent users from logging into workstations using administrative rights
- Monitor for suspicious PowerShell commands that can be used for performing credential extraction and pass the hash
- Restrict highly privileged accounts from logging into lower privileged systems
- Ensure that LSA Protection is enabled on critical systems to make it more difficult to extract credentials from LSASS

Jeff Warren
Jeff Warren is SVP of Products at Netwrix. Before joining Netwrix, Jeff has held multiple roles within Stealthbits - now part of Netwrix, Technical Product Management group since joining the organization in 2010, initially building Stealthbits' SharePoint management offerings before shifting focus to the organization's Data Access Governance solution portfolio as a whole. Before joining Stealthbits - now part of Netwrix, Jeff was a Software Engineer at Wall Street Network, a solutions provider specializing in GIS software and custom SharePoint development. With deep knowledge and experience in technology, product and project management, Jeff and his teams are responsible for designing and delivering Stealthbits' high quality, innovative solutions. Jeff holds a Bachelor of Science degree in Information Systems from the University of Delaware.