Pass the hash

By Contributors to Wikimedia projects

Published: 2010-09-11 · Archived: 2026-04-05 16:20:09 UTC

From Wikipedia, the free encyclopedia

In computer security, **pass the hash** is a hacking technique that allows an attacker to authenticate to a remote server or service by using the underlying NTLM or LanMan hash of a user's password, instead of requiring the associated plaintext password as is normally the case. It replaces the need for stealing the plaintext password to gain access with stealing the hash.

The attack exploits an implementation weakness in the authentication protocol, where password hashes remain static from session to session until the password is next changed.

This technique can be performed against any server or service accepting LM or NTLM authentication, whether it runs on a machine with Windows, Unix, or any other operating system.

On systems or services using NTLM authentication, users' passwords are never sent in cleartext over the wire. Instead, they are provided to the requesting system, like a domain controller, as a hash in a response to a challenge–response authentication scheme.[1]

Native Windows applications ask users for the cleartext password, then call APIs like LsaLogonUser[2] that convert that password to one or two hash values (the LM or NT hashes) and then send that to the remote server during NTLM authentication.[Notes 1][3]

If an attacker has the hashes of a user's password, they do not need the cleartext password; they can simply use the hash to authenticate with a server and impersonate that user.[4][5][6] In other words, from an attacker's perspective, hashes are functionally equivalent to the original passwords that they were generated from.

The *pass the hash* technique was originally published by Paul Ashton in 1997[6] and consisted of a modified Samba SMB client that accepted user password hashes instead of cleartext passwords. Later versions of Samba and other third-party implementations of the SMB and NTLM protocols also included the functionality.

This implementation of the technique was based on an SMB stack created by a third-party (e.g., Samba and others), and for this reason suffered from a series of limitations from a hacker's perspective, including limited or partial functionality: The SMB protocol has continued to evolve over the years, this means that third parties creating their own implementation of the SMB protocol need to implement changes and additions to the protocol after they are introduced by newer versions of Windows and SMB (historically by reverse engineering, which is very complex and time-consuming). This means that even after performing NTLM authentication successfully using the *pass the hash* technique, tools like Samba's SMB client might not have implemented the functionality the attacker might want to use. This meant that it was difficult to attack Windows programs that use DCOM or RPC.

Also, because attackers were restricted to using third-party clients when carrying out attacks, it was not possible to use built-in Windows applications, like Net.exe or the *Active Directory Users and Computers* tool amongst others, because they asked the attacker or user to enter the cleartext password to authenticate, and not the corresponding password hash value.

In 2008, Hernan Ochoa published a tool called the "Pass-the-Hash Toolkit"[7] that allowed 'pass the hash' to be performed natively on Windows. It allowed the user name, domain name, and password hashes cached in memory by the Local Security Authority to be changed at runtime *after* a user was authenticated — this made it possible to 'pass the hash' using standard Windows applications, and thereby to undermine fundamental authentication mechanisms built into the operating system.

The tool also introduced a new technique which allowed dumping password hashes cached in the memory of the lsass.exe process (not in persistent storage on disk), which quickly became widely used by penetration testers (and attackers). This hash harvesting technique is more advanced than previously used techniques (e.g. dumping the local Security Accounts Manager database (SAM) using *pwdump* and similar tools), mainly because hash values stored in memory could include credentials of domain users (and domain administrators) that logged into the machine. For example, the hashes of authenticated domain users that are not stored persistently in the local SAM can also be dumped. This makes it possible for a penetration tester (or attacker) to compromise a whole Windows domain after compromising a single machine that was a member of that domain. Furthermore, the attack can be implemented instantaneously and without any requirement for expensive computing resources to carry out a brute force attack.

This toolkit has subsequently been superseded by "Windows Credential Editor", which extends the original tool's functionality and operating system support.[8][9] Some antivirus vendors classify the toolkit as malware.[10][11]

Before an attacker can carry out a pass-the-hash attack, they must obtain the password hashes of the target user accounts. To this end, penetration testers and attackers can harvest password hashes using a number of different methods:

- Cached hashes or credentials of users who have previously logged onto a machine (for example at the console or via RDP) can be read from the SAM by anyone who has Administrator-level privileges. The default behavior of caching hashes or credentials for offline use can be disabled by administrators, so this technique may not always work if a machine has been sufficiently hardened.
- Dumping the local user's account database (SAM). This database only contains user accounts local to the particular machine that was compromised. For example, in a domain environment, the SAM database of a machine will not contain domain users, only users local to that machine that more likely will not be very useful to authenticate to other services on the domain. However, if the same local administrative account passwords are used across multiple systems the attacker can remotely access those systems using the local user account hashes.
- Sniffing LM and NTLM challenge–response dialogues between client and servers, and later brute-forcing captured encrypted hashes (since the hashes obtained in this way are encrypted, it is necessary to perform a brute-force attack to obtain the actual hashes).
- Dumping authenticated users' credentials stored by Windows in the memory of the lsass.exe process. The credentials dumped in this way may include those of domain users or administrators, such as those logged

in via RDP. This technique may therefore be used to obtain credentials of user accounts that are not local to the compromised computer, but rather originate from the security domain that the machine is a member of.

Any system using LM or NTLM authentication in combination with any communication protocol (SMB, FTP, RPC, HTTP etc.) is at risk from this attack.[1] The exploit is very difficult to defend against, due to possible exploits in Windows and applications running on Windows that can be used by an attacker to elevate their privileges and then carry out the hash harvesting that facilitates the attack. Furthermore, it may only require one machine in a Windows domain to not be configured correctly or be missing a security patch for an attacker to find a way in. A wide range of penetration testing tools are furthermore available to automate the process of discovering a weakness on a machine.

There is no single defense against the technique, thus standard defense in depth practices apply[12] – for example use of firewalls, intrusion prevention systems, 802.1x authentication, IPsec, antivirus software, reducing the number of people with elevated privileges,[13] pro-active security patching[14] etc. Preventing Windows from storing cached credentials may limit attackers to obtaining hashes from memory, which usually means that the target account must be logged into the machine when the attack is executed.[15] Allowing domain administrators to log into systems that may be compromised or untrusted will create a scenario where the administrators' hashes become the targets of attackers; limiting domain administrator logons to trusted domain controllers can therefore limit the opportunities for an attacker.[12] The principle of least privilege suggests that a least user access (LUA) approach should be taken, in that users should not use accounts with more privileges than necessary to complete the task at hand.[12] Configuring systems not to use LM or NTLM can also strengthen security, but newer exploits are able to forward Kerberos tickets in a similar way.[16] Limiting the scope of debug privileges on system may frustrate some attacks that inject code or steal hashes from the memory of sensitive processes.[12]

Restricted Admin Mode is a new Windows operating system feature introduced in 2014 via security bulletin 2871997, which is designed to reduce the effectiveness of the attack.[17]

- Metasploit Project
- Mimikatz
- Reflection attack
- SMBRelay

1. ^ Note that Windows may use Kerberos authentication by default.

1. ^ Jump up to: ***a*** ***b*** *Chris Hummel (12 October 2009). "Why Crack When You Can Pass the Hash?". SANS Institute.*
2. ^ *"LsaLogonUser". Microsoft. 7 September 2011. Retrieved 25 October 2011.*
3. ^ *"How Interactive Logon Works". Microsoft. 22 January 2009. Retrieved 25 October 2011.*
4. ^ *"What is a Pass-the-Hash Attack (PtH)?". BeyondTrust. 2023-08-04. Archived from the original on 2024-05-15. Retrieved 2024-06-23.*
5. ^ *Lenaerts-Bergmans, Bart (2024-02-21). "What is a Pass-the-Hash Attack?". crowdstrike.com. Archived from the original on 2024-04-07. Retrieved 2024-06-23.*

6. ^ Jump up to: *a b* *Daniel Stirnimann (9 August 2010). "Windows Attack — Gain Enterprise Admin Privileges in 5 Minutes" (PDF). Compass Security AG. Archived from the original (PDF) on August 26, 2014. Retrieved 10 October 2010.*
7. ^ *Hernan Ochoa (2 July 2008). "What is Pass-The-Hash Toolkit?". Retrieved 20 October 2011.*
8. ^ *Hernan Ochoa (2011). WCE Internals. RootedCON.*
9. ^ *Hernan Ochoa (2011). "Windows Credentials Editor (WCE) F.A.Q." Amplia Security. Retrieved 25 October 2011.*
10. ^ *"SecurityRisk.WinCredEd". Symantec. 21 March 2011. Archived from the original on April 13, 2012. Retrieved 25 October 2011.*
11. ^ *"HackTool:Win32/Wincred.A". Microsoft. 1 October 2011. Retrieved 25 October 2011.*
12. ^ Jump up to: *a b c d* *Bashar Ewaida (21 January 2010). "Pass-the-hash attacks: Tools and Mitigation". SANS Institute.*
13. ^ *Roger Grimes (26 July 2011). "Stop pass-the-hash attacks before they begin". InfoWorld. Retrieved 25 October 2011.*
14. ^ *Rob Kraus; Brian Barber; Mike Borkin; Naomi Alpern (2010). Seven Deadliest Microsoft Attacks. Syngress. pp. 12–14. ISBN 978-1-59749-551-6.*
15. ^ *"Preventing Pass-the-Hash Attacks and Cached Credential Attacks". Berkley Lab Computer Protection Program. Archived from the original on 4 May 2011. Retrieved 20 October 2011.*
16. ^ *"Microsoft Windows Kerberos 'Pass The Ticket' Replay Security Bypass Vulnerability". securityfocus.com. 13 August 2010. Archived from the original on 12 March 2016. Retrieved 20 October 2010.*
17. ^ *"Microsoft Security Advisory 2871997". 14 October 2022.*

- Microsoft Pass the Hash Mitigation Guidance
- Amplia Security
- SMBShell
- Patrick Jungles et al.: *Mitigating Pass-the-Hash (PtH) Attacks and Other Credential Theft Techniques*, Microsoft Corp., 2012, retrieved on Feb. 3, 2015
- Uninformed Break-the-hash paper
- Reducing the Effectiveness of Pass-the-Hash(NSA)
- CWE-836: Use of Password Hash Instead of Password for Authentication

---