Global Information Assurance Certification Paper

Why Crack When You Can Pass the Hash?

GIAC (GCIH) Gold Certification

Author: Chris Hummel, chris_hummel@hotmail.com

Advisor: Joey Niem

Accepted: October 12, 2009

Abstract

A weakness exists in the design of Windows unsalted password hashing mechanism. The static nature of this password hash provides the means for someone to masquerade as another user if the victim's hash can be obtained. While the concept of passing a Windows password hash has been around for some time, the release of publicly available tools has taken the first major step towards harnessing the true power of this attack. Although such tools have not yet targeted Microsoft's implementation of Kerberos, all organizations are strongly encouraged to move towards pure Kerberos deployments in preparation for PKI integration. The evolving nature of this attack puts under pressure the issue of passwords as a valid identifier thus requiring organizations to use an alternate credential form such as digital certificates.

© SANS Institute 2009, Author retains full rights.

1. Introduction

The availability and subsequent usage of publicly available tools requires incident handlers and other security practitioners to understand the nuances of various tools and countermeasures. As is the case with most security tools, the one in possession of the tool is also the one who determines whether it will be used for good or for bad. While there will always be a group in society targeting the latter, this has been written for those well-intentioned professionals who require a deeper understanding of this topic.

The SEC504 course briefly touches upon the pass the hash attack and this will build upon those components by providing an in-depth analysis of the technical underpinnings of the attack. Beginning in the first section, there will be an overview of the main Windows components that are utilized in this attack. The following two sections explore the various methods used to obtain and pass the hash culminating with a demonstration using one of the more popular and advanced tool suites in use today, the Pass-The-Hash Toolkit (or PSHTK for short). An analysis of the attack will be broken down into systems and services risk analysis followed by attack ramifications. Lastly, the final section offers multiple defensive countermeasures to be applied in a layered fashion to proactively mitigate hash dumping and passing attempts.

1.1. What is this attack, how does it work, and why is it possible?

Passing the hash is an attack based on having a valid set of credentials (a username and its password hash) and authenticating to a remote system as that user. The process of hashing a password means that the cleartext form is never stored or transmitted. The Windows operating system generally stores two different hashed forms of a user account's password [Ochoa 2008, October 29]. In its simplest form, a pass the hash attack begins with a stolen Windows password hash where it is loaded into running memory on an attacker-controlled system. Used in conjunction with the corresponding user account and domain name, the hash in memory is then used to form a valid set of credentials when attempting to access and authenticate to a target system. An attacker-controlled system is any system that yields the ability to execute a hash passing tool at the

attacker's discretion. Thus, the hash passing tool provides the mechanism to load the stolen password hash into running memory. Once the credentials are passed from memory to the target service, access to the resource will be verified. Assuming the access control list grants permission to that user account, the attacker-controlled system now has access to the target's resource and is operating under the disguise of an authorized user. During this attack, the real password is never revealed and the password hash is never decrypted – it's just passed. The attack is made possible because the password hash is static (no salts for randomization) and the hash can be obtained in the first place [Murray 2007].

To better illustrate this attack, Figure 1.1.1 below depicts a possible scenario for an attacker or a penetration tester. For this example, an SMB connection (TCP port 139/445) will consist of mapping a drive (C$) to the target system. Also note that step one is not a requirement but is used here to demonstrate the effectiveness of this attack from external locations. An organization's Web server has just been exploited providing the attacker with complete control of the Windows system. The attacker proceeds to dump the local password hashes and identifies several domain accounts. While running the netstat command they notice an active connection from a private address that's not on the DMZ subnet, so they decide to probe further. The attacker takes advantage of a loose firewall security policy and is able to establish an SMB connection to this internal LAN system in step two. In order to authenticate to the system, the attacker had to first inject the password hash of the domain account (effectively overwriting the one in memory on the Web server). This was accomplished with the help of a hash passing tool. In addition to the hash, these tools are able to collect (and pass) the corresponding user account name and domain/workgroup as these values are stored in cleartext. In the final step, the attacker uses nslookup to verify that this system's domain controller is also located on the same internal LAN as the PC. With the domain account's hash still in memory, the attacker catches a break as it happens to be one of the domain administrators. This allows them to establish their SMB connection to the LAN Server and gain total control of the organization's domain controller.

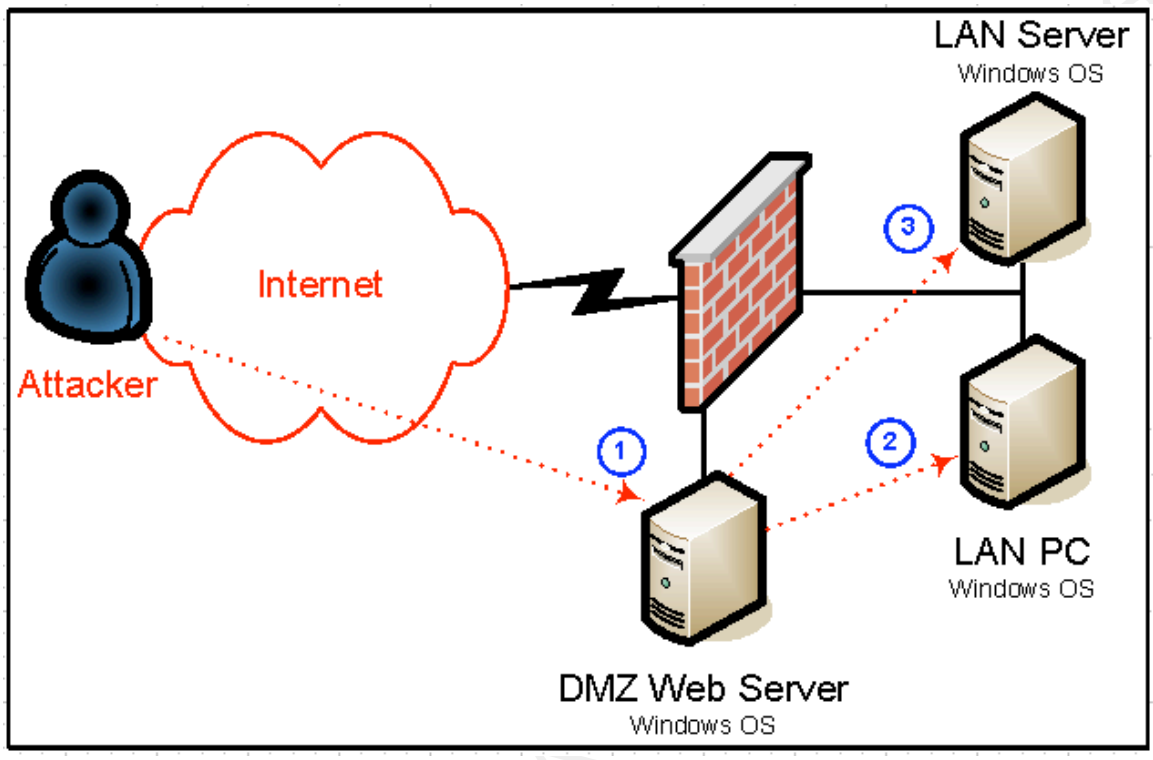

Figure 1.1.1 – One example of the hash passing attack

2. Windows Components

It is necessary to journey into the world of Windows to understand the underlying components of this attack. As this attack relies upon a Windows username and corresponding password hash, it can be used against systems implementing LM or NTLM authentication. As of this writing, there are currently no publicly available tools that perform the attack against a system using the Kerberos authentication package. In addition, all of the hash passing tools mentioned here were designed to interact with those services supported by the SMB protocol. An exception to this is the PSHTK which will be explained later on in much greater detail due to this capability.

2.1. Windows Password Hash

For modern Windows systems up to and including Windows Server 2003, there are two types of password hashes that are used: Lan Manager (LM) and the Windows NT hash [Johansson 2006]. The LM hash is only used in conjunction with the LM authentication protocol, while the NT hash serves duty in the NTLM, NTLMv2, and

Kerberos authentication protocols [Microsoft 2007]. These two mechanisms provide the hashed password representations for the four major Windows authentication protocols in use today. When a user performs an interactive login, the Local Security Authority Subsystem Service (LSASS) process is called upon to store the hashes in memory [Johansson 2005]. Additionally, if a password is used that consists of 14 or less characters, both the LM and NT hash will be created and stored in the local Security Accounts Manager (SAM) file or in Active Directory (AD) [Microsoft 2007]. To make matters worse, neither of these one way functions incorporates a degree of randomization (called a salt) into the process [Johansson 2005]. So, if two NT hashes are identical then the odds are extremely high that the passwords they're based on are also identical (the topic of hash collisions are outside the scope of this paper). Also, the NT hash is derived by taking the cleartext password and converting it to Unicode, and then hashing that value with the MD4 algorithm [Grutzmacher 2008].

2.2. SMB/CIFS Protocol

The second component revolves around the SMB protocol which was renamed by Microsoft to CIFS for its implementation in the Windows environment. While SMB is generally related to Windows, it is also used in the Unix/Linux world most notably in SAMBA deployments. The default TCP ports are 139 and 445 where the former is a carry over from the NetBIOS days and is used mainly for backwards compatibility [Hertel 1999]. From this point forward, this protocol will be referred to as SMB for the sake of simplicity and in the context of remote connectivity.

The SMB protocol is responsible for several services that many organizations use on a daily basis. This is not meant to be an exhaustive list but rather a sampling of the more popular services. At its core, SMB allows for the sharing of directories, files, and printers [Hertel 1999]. It allows for the use of the Microsoft Management Console (MMC) and many of its snapins, the Registry Editor, and the Windows "netsh" commands [Marchand 2005]. Additionally, it provides for the Windows "net" and "reg" commands [Skoudis 2008]. Beyond the native services, there are third-party tools such as pstools from SysInternals which provide various enumeration and management functions [Marchand 2005].

2.3. Windows Authentication Protocols

The third and final component necessary to implement this attack is that of the Windows authentication protocols. Over the years, the design of these protocols evolved in parallel with the Windows OS. As weaknesses were identified, protocols with enhanced security were introduced while older ones were still supported to maintain backwards compatibility. This cycle continued until Microsoft adopted (and tweaked) the industry standard Kerberos as its default authentication protocol for its Windows OS. Figure 2.3.1 provides a brief synopsis of these protocols and their level of exposure to the hash passing attack.

Authentication Protocol	Susceptible to Hash Passing?	Public Tools Availble?
LM	Yes	Yes
NTLM	Yes	Yes
NTLMv2	Yes	Yes
MS Kerberos v5	Yes	No

Figure 2.3.1 – Using IDA Pro to locate memory addresses

2.3.1. Lan Manager (LM)

The LM hash is only used in conjunction with the LM authentication protocol. With its debut in Windows 3.11, this protocol was one of Microsoft's first and as a result, it carried many security problems with it [Melber 2004]. One of the more notable issues pertains to how LM transmits the password hash in cleartext across the network thus making it prone to capture [CIAC 2008]. After many years of service, Microsoft finally released an operating system (Vista and Server 2008) where this was disabled by default [Grutzmacher 2008].

2.3.2. NT Lan Manager (NTLM)

The introduction of Windows NT 3.1 provided a new directory service and thus a new authentication protocol called NTLM [Melber 2004]. There were very few changes under the hood between LM and NTLM, but there was one significant change. NTLM, also referred to as NTLMv1 in some circles, incorporated a challenge response mechanism which effectively prevented the transmission of the hash over the network [CIAC 2008].

2.3.3. NTLMv2

Unlike the previous two protocols, NTLMv2 was not released with a major OS but with the release of a service pack – Windows NT 4 SP4 [Melber 2004]. While not a complete overhaul, NTLMv2 resolved many security deficiencies of its predecessors through the use of longer passwords, longer keys for the hash, and mutual authentication [Melber 2004]. Additionally, the security of the challenge response mechanism was also enhanced [CIAC 2008].

2.3.4. Kerberos

The last of the four protocols is an industry standard that Microsoft has made a few adjustments for its use in Windows. Needless to say, this is an entirely different protocol that shares very few similarities with the proprietary creations from Microsoft. With the release of Windows 2000, Kerberos (version 5) became the default authentication package [Microsoft 2003]. Some of its strengths include its resistance to replay attacks, the enforcement of mutual authentication, and the prevention of password transmission across the network [Melber 2004]. Kerberos is used in conjunction with resources located within a Windows AD domain [Johansson 2005] and it uses renewable session tickets which replace the pass-through authentication used by NTLM [Microsoft 2003]. In the NTLM world, application servers had to communicate with a domain controller to authenticate each client; with Kerberos, a server can authenticate a client by reviewing the credentials presented by the client [Microsoft 2003].

3. Obtaining the Hash

Before a hash can be passed, the hash must be stolen. This section covers the many potential avenues for pilfering the most critical component of this attack – the Windows password hash. It's assumed that the "thief" has already elevated their system privileges to an administrator equivalent in order to pull off the heist. Using a variety of methods, the hash can be found in memory, on the hard drive, on removable media, or floating across the network.

3.1. Boot the target system to an alternate OS and copy the files to removable media

The first method requires physical access to the system in question. Due to their free roaming nature, laptops are a prime choice followed by those systems where physical security is rather lax. Someone once said that if it can be touched it can be owned. Well, if this one system can be owned it's possible the entire domain can be owned as well. With physical access the system can be booted to an alternate OS where key files can be copied to removable media. Windows password hashes are stored in the SAM file located within %windir%\system32\config [McClure 2008]. If the Create Emergency Repair Disk function was used, there will also be a backup of the Registry including the SAM hive located in the directory %windir%\repair\RegBack [McClure 2008]. When it comes to a Windows AD domain controller, every single user account and hash for the domain resides on this system. Drawing comparisons to the crown jewels of a kingdom, the hashes are stored in the ntds.dit file located within %windir%\WindowsDS [McClure 2008]. Also, don't forget about those backup tapes!

3.2. Sniff the LM hash off the network

The LM authentication protocol was one of Microsoft's earliest attempts and was designed during a time period when security was not of the utmost importance. While this protocol should no longer be used, it may still be lurking around if there are legacy systems that require it. If one of these networks is encountered, the LM hash will be used in conjunction with the LM protocol which transmits the hash in cleartext across the network. This makes it is entirely possible to capture packets off the wire containing the hash.

3.3. Dump the hash locally

In order to obtain a hash locally on a system, a hash dumping tool must be executed while logged in via an interactive session. This means sitting at the system console or using a Terminal Services session. There are currently several tools available that provide this capability: pwdump, fgdump, and gsecdump [CIAC 2008]. These tools effectively seek out and obtain hashes via the registry and through DLL injection [LCPsoft 2004]. In addition to the local system accounts, the tools attempt to retrieve

hashes from active login sessions and cached domain logins [CIAC 2008]. Thus, if a system administrator logged into this system using their domain administrator credentials, the password hash would be cached along with other recent domain logins. It should also be noted that the tools msvctl and whosthere.exe (of the PSHTK) can also dump hashes but are limited to only those kept in memory [Gates 2008]. The last tool is actually a suite of tools and is called Cain & Abel. Within the suite there exists the NT Hash Dumper feature which uses the same technique as pwdump: DLL injection into the LSASS process [Montoro 2009].

3.4. Dump the hash remotely

Moving beyond local collection, hash gathering can also occur through remote means. The ability to reach a system across the network requires the necessary TCP/IP ports to be accessible and listening on the end system. Once the connection is established to the target system, authorized credentials must be provided to gain access. These tools require privileged access so if the local administrator's hash can be obtained from some system in the domain, the odds are good that the same hash (same password) will provide access to other systems on that network. Could those other systems be a domain controller? Absolutely, as long as the same hash (password) is used for the local admin account [Murray 2007].

The same tools used locally (pwdump, fgdump, gsecdump) also work remotely with the exception of the latter which must be used in conjunction with the psexec tool [CIAC 2008]. Psexec also provides the foundation for remote execution using msvctl and whosthere.exe [Gates 2008]. In addition to providing local hash dumping, the Cain & Abel suite offers remote execution via Abel's ability to run as a local service under the system account [Montoro 2009]. Another tool that must be mentioned due to its popularity and robust feature set is Metasploit. Within Metasploit there exists an advanced payload targeting Windows systems called the Meterpreter. When running the Meterpreter, the privileges module can be loaded to run the hashdump program [Skoudis 2009]. As for the commercial penetration testing frameworks, Immunity's Canvas and Core Security's Core Impact both supply the means for remote hash collection [Grutzmacher 2008].

Except for the post exploitation utilities (Metasploit, Canvas, Core Impact), the other tools employ a very similar formula for successful remote hash collection. The primary executable begins with an SMB connection over TCP ports 139/445 to a writeable share (C$, ADMIN$, etc.). This is generally followed by a file transfer of an executable and a DLL. At this point, the main program contacts the Windows Service Control Manager (SCM) Remote Protocol. SCM acts as an RPC server where clients can access it via the svcctl named pipe interface through an SMB connection [Microsoft 2005]. SCM operates as the services.exe program (running as NT AUTHORITY\SYSTEM) with responsibility over all service control functions. The first call to svcctl is a request to create a temporary set of registry keys necessary for the temporary usage of the transferred executable. Once this is in place, the next call to svcctl instructs the creation of a new process which loads the transferred executable onto the remote system. It then begins to load other Windows system libraries such as advapi32.dll and rpcrt4.dll along with the transferred DLL. To dump the hashes, the transferred DLL is then injected into lsass.exe whose high privileges can easily complete the request.

3.5. Dump and analyze the contents of memory

Another technique to obtain hashes involves dumping and analyzing the contents of system memory. This method can be performed locally or remotely depending on what tools are used. The ManTech Memory DD (MDD) is a tool that can copy everything in memory for multiple Windows operating systems [Gates 2009]. Depending upon the amount of system RAM, its usage will generate a fairly large dump file that is not human readable. One such tool that can perform this analysis and extract the hashes is Volatility from Volatile Systems [Gates 2009]. While this will accomplish the task locally, the services of another tool will be required for remote access. Once again, the Metasploit framework is called upon to inject the MDD executable across the wire [Gates 2009]. Once the dump file is produced, it will need to be transferred back to the source system for analysis with Volatility.

3.6. Implement a rogue SMB server to collect hashes

The final collection technique pertains to how a victim system can be tricked or lured into opening an SMB connection to an attacker controlled system. There are multiple ways this can happen, but technically speaking the end result is the same: the victim opens an SMB connection and happily transmits their username and password hash to the rogue system [Warlord 2006]. The rogue system must be running a tool capable of collecting these hashes as they are presented. The original SMB relay tool (called SMBRelay) was created by Sir Dystic (Cult of the Dead Cow) in March of 2001 [Grutzmacher 2008]. Another option is to use the auxiliary module called smb_sniffer within Metasploit that also provides this capability [Moore 2007]. In addition to collecting hashes, both of these tools are able to connect back using these credentials to either the victim host or a third system of their choosing [Grutzmacher 2008].

4. Exploring the Hash Passing Tools

While the previous collection methods may have resulted in an abundance of hashes, some may choose to travel along an older yet more familiar path of hash cracking. Tools certainly exist to provide such brute force services, but cracking can be very time intensive and there's no guarantee that the real password will be discovered within a reasonable time frame. As a result, many will opt to perform hash passing as it's similar to passing a password and this provides near instantaneous gratification.

As various hash passing tools have been created over the years, the majority of them have suffered from the same limitation: the inability to use native Windows tools. This inability reared its head in two places: command shell imprisonment and SMB service restriction. While the author cannot comment on why those tools were designed as such, it was a matter of time before someone devised a utility that was free of these restrictions. The person is Hernán Ochoa and the utility is the PSHTK. The primary advantage of the PSHTK is that it works with any program that uses NTLM authentication [Ochoa 2008, October 29]. Such programs could be from Microsoft or any third party as it does not matter. Various resources exist that falsely state that the hash passing tools only work when targeting an SMB resource. This is simply not true

for the PSHTK and is one of the biggest distinctions between itself and the others. Because of these capabilities, this tool suite will be explored in depth and featured in the attack demonstration.

4.1. Modified SMB client (Paul Ashton)

Known officially as Bugtraq ID #233, Paul Ashton put this type of attack on the map back in April 1997 [Bugtraq 1997]. The simple yet effective code allowed for the effective passing of the LM hash against the older Windows operating systems using LM authentication [Bugtraq 1997].

4.2. Modified SAMBA code (JoMo-kun of Foofus)

Using the existing code for SAMBA clients, JoMo-kun created a set of patches that allows for hash passing. The patches work against SAMBA and SAMBA-TNG to implement LM, NTLM and NTLMv2 hash passing [JoMo-kun 2003]. The code works by pre-defining an environment variable consisting of the LM hash and NT hash [Skoudis 2008]. When the SMB target is accessed, the SAMBA client passes the supplied username and hash.

4.3. SMBshell (Nicolas Pouvesle of Tenable Network Security)

For those familiar with the modularity of the Nessus NASL script plugins comes SMBshell that can operate as a standalone program [Gula 2007]. Designed to be an interactive SMB shell for various Windows security auditing functions, the tool only requires NT hashes for operation [Gula 2007].

4.4. Windows auditing feature of Nessus (Tenable Network Security)

Recognizing the important role that hash passing plays while conducting a security audit, the team at Tenable decided to incorporate this feature into their Nessus scanning product. Using the hashes collected from other tools, this Nessus feature allows its user to specify either the LM or NT hash [Gula 2007]. A total of four credentials using these values (username / password hash / domain) can be supplied per scan policy which allows some flexibility when attempting to gain remote privileged access to systems across the network.

4.5. Hydra (van Hauser of THC)

THC-Hydra is a very fast network logon cracker that supports dozens of services [van Hauser 2006]. It supports such services such as SMB, HTTP, HTTPS, and FTP. Hydra is equipped with the "smbnt" service module which accepts the NT hash values [van Hauser 2006]. As a result, the tool can be used to test remote network access over multiple services using NTLM authentication.

4.6. Penetration Testing Frameworks

There are several open source and commercial frameworks that are capable of sending a hash to a target system. Metasploit is as an open source tool that leverages the built-in psexec exploit. By itself this does not provide the hash passing capability, however Kurt Grutzmacher took it upon himself to write the code that does [Grutzmacher 2007]. The patch allows the Metasploit user to invoke the "set SMBuser" and "set SMBpass" commands to provide the necessary credentials [Skoudis 2009]. The SMBpass parameters consist of the LM hash and NT hash [Grutzmacher 2007]. The commercial frameworks are represented by Immunity's Canvas and Core Security's Core Impact [Grutzmacher 2008].

4.7. Msvctl (Johannes Gumbel of Truesec Security)

Under wraps for several years, the Truesec team eventually decided to release its internally developed hash passing tool used in many of its customer security assessments. Created by Johannes Gumbel, msvctl not only dumps hashes but can pass them as well. In similar fashion to the Grutzmacher patch, msvctl allows its operator to specify the LM hash and NT hash [Gates 2008]. Once those values are applied, the tool behaves like a "runas" session and spawns a new command window operating under the context of the supplied credentials [Murray 2007].

4.8. Pass-The-Hash Toolkit (by Hernán Ochoa of Core Security)

Last but not least is the Pass-The-Hash Toolkit from Hernán Ochoa. Released in August 2007, the PSHTK is written in C and runs on Windows XP, Windows Server 2003, and Windows Vista [Ochoa 2008, October 29]. While the previous tools are all capable of passing a hash, the PSHTK is a collection of utilities providing multiple

functions. PSHTK consists of three primary tools: genhash.exe, whosthere.exe, and iam.exe. Each program operates independently of one another as there is no main menu or GUI that ties them together. Genhash.exe generates the LM and NT hash for a given password and is used mainly to test the other two tools [Ochoa 2007]. Whosthere.exe is a hash dumping tool that retrieves user session information (including the NT hashes) from those logon sessions residing in memory [Ochoa 2007]. Iam.exe provides the hash passing capability. The current logon session information can be modified by specifying the username, domain/workgroup, and LM and NT hashes [Ochoa 2007].

Unlike the other public tools, Hernán has provided ample information on the DNA of his creation. The crux of the design decision lied within whether or not to manipulate the memory address space of the LSASS process (lsass.exe) or to inject and execute code directly inside of it [Ochoa 2008, March 6]. Each technique has their pros and cons. The former requires the tool to target precise addresses within the memory structures used by lsasrv.dll, but this library tends to change between the different versions of Windows [Ochoa 2008, March 6]. The upside to this method is that it's generally safer, it's stealthier, and it works on Vista [Ochoa 2008, March 6]. On the other hand, executing code inside lsass.exe will provide much broader "out of the box" support for the various flavors of Windows [Ochoa 2008, March 6]. The downside is that it can be riskier as it's possible to disrupt the critical LSASS system process which could lead to a system crash [Ochoa 2008, March 6].

Whosthere.exe is the recipient of the memory manipulation technique. As different versions of Windows (and patches within that version) use different versions of lsasrv.dll, it can be a daunting task to determine the specific set of memory addresses given the various combinations of Windows throughout the world. Nonetheless, the tool comes pre-loaded with a version to address mapping and using basic heuristics it's able to test unknown versions during runtime to determine if they can be used [Ochoa 2008, March 6]. The design of iam.exe took a hybrid approach as it uses elements of each technique to achieve both of its primary functions [Ochoa 2008, October 29]. Before iam.exe can modify the logon session credentials, it must first locate and read that information from memory using the same technique as whosthere.exe uses [Ochoa 2008, October 29]. With that information in hand, the tool can proceed with the injection of

iamdll.dll into lsass.exe which calls the functions from lsasrv.dll [Ochoa 2008, October 29].

Rather than compromising on one technique over the other, Hernán's accommodating soul has produced two versions of whosthere.exe and iam.exe. Known as the alternate tools, whosthere-alt and iam-alt, this pair relies upon code execution within lsass.exe by using functions from msv1_0.dll [Ochoa 2008, March 6]. This code execution is actually the injection of the supplied pth.dll which explains why it exists in each alt tools directory [Ochoa 2008, October 29]. The alt tools are preferred when broad support of Windows is required and the user accepts the risk carried by the possible perils of this technique.

In the event that this tool and the alt version will not work on a specific system, Hernán has provided yet another way to accomplish the task. Using the IDA Pro disassembler and debugger application, the problematic lsasrv.dll can be loaded and analyzed [Ochoa 2008, July 2]. While the current version of IDA Pro is for sale, a slightly older version is maintained and freely available that will suffice for this procedure. After selecting the dll to load, IDA will recognize the need to load the Microsoft symbols table. Without the symbols table lookup, the script will not be able to find the memory addresses. Once IDA has finished loading the dll, select the menu option for running an IDC script and browse to the location of the passthehash.idc script that's included with the PSHTK (source code version only). As shown in Figure 4.8.1 below, the execution of the script will reveal the memory addresses (prefaced by a #define) used by that dll in the bottom portion of the screen [Ochoa 2008, July 2]. The output provides six memory addresses, one per line, and displays them in the same order that is used at the command line. Using the tool's "-a" switch, these addresses can be manually entered for accurate operation on the system in question [Ochoa 2008, July 2]. Because these addresses are hardcoded into each dll, you can save this address set for future re-use on the same system. The memory addresses used with whosthere.exe are identical to those used with iam.exe because the tools are executed on the same system and are thus interacting with the same version of lsasrv.dll. Note however, that certain Windows OS updates may modify lsasrv.dll rendering these addresses useless.

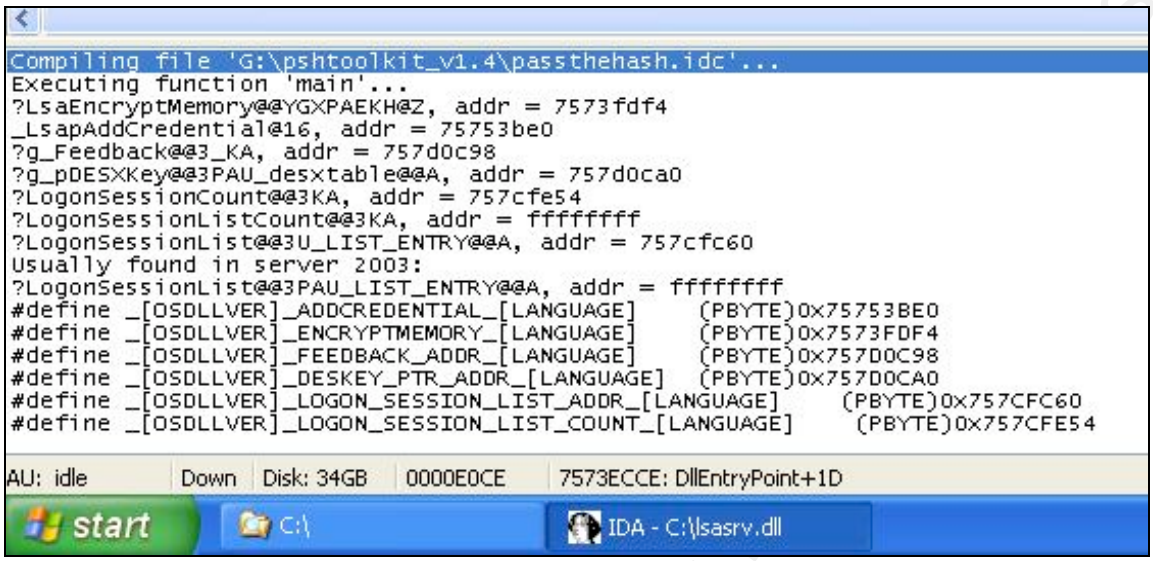

Figure 4.8.1 – Using IDA Pro to locate memory addresses

4.9. Attack Demonstration

The following demonstration has been provided to show proof that the PSHTK is effective against a non-SMB target. In Hernán's own words, "Credentials associated with logon sessions are the credentials used when you want to access a remote resource using NTLM auth. So if we change these credentials (e.g.: modify the password hashes), we modify credentials used for over the network auth and we will accomplish our goal" [Ochoa 2008, October 29]. Modifying the credentials is exactly what is accomplished by the iam.exe and iam-alt.exe tools.

The demonstration begins with the execution of the whosthere.exe tool. Notice that the "-a" switch was needed to specify the memory addresses on this particular system. The output in Figure 4.8.2 reveals that the user "test1" in the domain/workgroup "cobra2" is part of an active LSA logon session.

Figure 4.8.2 – Dumping active logon sessions with Whosthere.exe

Moving along to Figure 4.8.3, this screenshot depicts a packet capture of the client system (192.168.200.11) attempting to authenticate against a Proxy Server (192.168.187.150) over HTTP (TCP port 80). The client is attempting to access an Internet web site and its browser is configured to use the corporate Proxy. Notice how the client is engaged in the NTLM challenge/response with the Proxy while the communication occurs over HTTP. As the "test1" account is bogus and does not exist in the domain, the Proxy will deny this connection request (as expected).

Figure 4.8.3 – NTLM authentication over HTTP using the bogus account

The second part of the demonstration begins with the execution of the genhash.exe utility as seen below in Figure 4.8.4. While purposely blacked out, a password is supplied so the tool can generate the corresponding LM and NT hashes. Since the author has a valid account on this domain (and knows the password), genhash.exe can be used to produce the hashes that are fed into iam.exe.

Figure 4.8.4 – Using genhash.exe to create the new LM and NT hashes

Moving on to Figure 4.8.5, the output shows iam.exe working its magic (part of the user name and domain name have been sanitized). After supplying iam.exe with a username, domain/workgroup, and the LM and NT hashes, it proceeds with injecting its DLL into LSASS to effectively overwrite the active LSA logon session.

Figure 4.8.5 – Using iam.exe to modify the active logon session credentials

The final step of the operation concludes with the execution of whosthere.exe to verify the successful modification of the active LSA logon session credentials. As shown in Figure 4.8.6, the previous session has been overwritten with the credentials of the valid domain account supplied to iam.exe in Figure 4.8.5.

Figure 4.8.6 – Using whosthere.exe to verify the new active LSA logon session credentials

In the first test (displayed in Figure 4.8.3), the client system attempted to use NTLM authentication against a Proxy Server (over HTTP) using bogus credentials. With

the valid credentials now in memory, the client attempts the same connection again but this time the domain credentials are provided to the Proxy and authentication is granted (an expected behavior shown in Figure 4.8.7. below). As the client system successfully engages in an NTLM challenge/response session over HTTP with the Proxy, the attack demonstration proves that the PSHTK is effective against non-SMB targets.

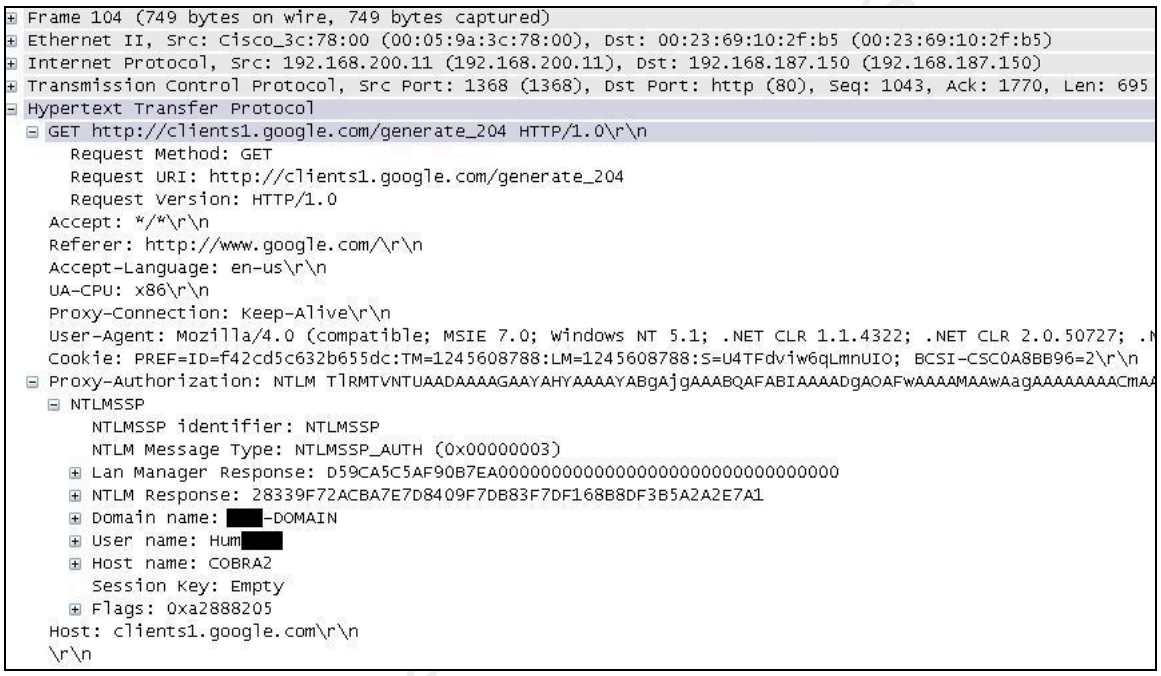

Figure 4.8.7 – Successful NTLM authentication over HTTP using the PSHTK supplied account

5. Attack Analysis

This section focuses on the risk analysis of those systems and services utilizing the Windows LM and NTLM authentication schemes. While Windows is the predominant OS affected by this, there are many non-Windows OS's that may be vulnerable based on the provided services. The far reaching scope of the attack is best illustrated through the identification of such vulnerable systems and services. Using a worse case attack scenario, the ramifications of successful hash passing will be brought to light.

5.1. What systems are at risk?

Any system that is using LM or NTLM (v1 and v2) authentication is at risk. While NTLMv2 is supported in Windows Server 2008, Vista, Server 2003, Windows 2000, and XP, the default authentication protocol is Kerberos v5 [Microsoft Feb 7, 2008]. Unless the environment is based purely on Kerberos and other steps have been taken to ensure that authentication negotiation (down to NTLM) cannot occur, one or more of these Windows platforms will be at risk. Generally speaking, the larger the environment the more challenging it will be to identify and remediate.

Moving away from the Windows OS, there is the SAMBA implementation of SMB on Unix and Linux operating systems. SAMBA is an add-on package that uses NTLM authentication thus making it every bit as vulnerable as its Windows counterpart. The SAMBA code has been modified for use on other systems such as VMS, AmigaOS, & NetWare [Hertel 2001]. Besides Unix and Linux, SMB is also supported on OS/2, Macintosh, and also on dedicated file server platforms from various suppliers [Hertel 2001]. If a system exists that's providing an SMB-like service, verify its operation through a series of packet captures. This will help determine if it's using the SMB protocol and LM or NTLM authentication.

5.2. What services are at risk?

Any service that uses the LM or NTLM (v1and v2) authentication package is at risk. In this instance a service is referred to as the offering which is obtained as a result of successful authentication rather than the protocol (SMB, HTTP, .etc) that invokes the authentication process. The following is not meant to be an exhaustive list, but a collection of the more widely used services throughout corporate environments. Some of these are native to the Windows OS while others are third party implemented.

First and foremost is the risk presented to file shares, directories, and printers. Next are the native Windows administration utilities accessed via the MMC, the Registry Editor, and the Windows commands (netsh, net, reg). Beyond the native services, there are third-party management tools such as pstools. What about the proxy servers used for accessing the Internet? Are they configured to use LM or NTLM authentication? If so, that service is also at risk. What about Web sites on the corporate intranet? The system

may never provide a user prompt but it knows who's logged in as a unique identifier is displayed on the page. The chances are that it's using NTLM. Verify this by running a packet capture on the client system. What other juicy services on the internal network leverage this type of Windows authentication? How about Payroll or HR apps? Maybe, maybe not – this should be verified. What about commercial backup solutions that use SMB file sharing to allow for the easy archival and restoration of data? Are there any network or security devices that leverage NTLM authentication? Cisco's ASA Firewall supports NTLM as an authentication option for Remote Access implementations. While LM is not a major concern nowadays, the bottom line is that the prevalence of NTLM in third party solutions will allow this attack to prevail for the foreseeable future.

5.3. What are the ramifications of this attack?

Once a hash is obtained, the possibilities from here are quite far reaching. Granted, the initial hash may 'only' be that of an average user, but one hash leads to two, two leads to three, and so on. As security professionals, it must be assumed that the worse has happened and prepare accordingly. For this scenario, assume that the domain administrator credentials have been obtained for the network in question. For those who do not know, a domain admin account is the God account of a Windows network. While the System account is the king of an individual Windows system, domain admin is the supreme ruler of the domain.

What happens next depends upon the intentions of the attacker. What's their motivation? That will vary but some of these possibilities can be explored. The first ramification encountered is that of user impersonation and masquerading. While seemingly benign in nature, this is generally a precursor to far worse things. Unauthorized access via authorized credentials is one such thing while unauthorized disclosure (data snooping/reading) is another. Many organizations store sensitive data in Windows file shares – an area where a domain admin would be granted full access. Having full access means having the ability to read (unauthorized disclosure), write (unauthorized modification), and delete (unauthorized destruction). The effective destruction of data (especially if there is no backup) leads to a denial of service condition.

Full access also allows for the copying and removal of data which is the primary means behind data theft.

The fastest way to complete domain domination involves the attacker using the domain admin credentials to logon to a domain controller and dump all of the hashes in the AD database [Murray 2007]. While domain admin provides full access to the Windows domain, this does not mean the same privilege level is granted to non-Windows services that use the Windows AD. For example, an environment may consist of an AD group that permits its members to access the Payroll application as this has been configured to leverage the existing AD via NTLM. The stolen domain admin hash could then be used to further escalate their privileges. This could be done by adding that domain admin account into the Payroll group or by targeting an authorized Payroll user and obtaining their hash. While the latter would be stealthier, both methods allow the attacker unauthorized access to sensitive information.

For those motivated by further system penetration and complete network dominance, the following is a pair of methods that feed off one another. Having domain admin means that an attacker can modify Windows system access controls at their discretion. This includes native controls such as NTFS permissions and Windows Firewall, and third-party controls such Anti-Virus, Host-based IDS/IPS, and Client Firewall. By disabling or creatively circumventing such security controls, this exposes the operating system to its worst nightmare. With all defenses down, the injection of malicious code (especially in the form of a kernel-level rootkit) would change system ownership to that of the attacker. In this sense ownership means no further use of the domain account is necessary as various backdoors would allow the attacker free access using whatever means they configured. Having complete system ownership establishes a beachhead in the target network and thus becomes a stepping stone for further system penetration.

6. Defensive Techniques

A hash, salted or unsalted, is the equivalent of static gibberish that represents a user's password and provides the equivalent functionality when obtained. While a salted

hash provides undeniable protection against brute force and rainbow table style of attacks, the hash remains unchanged until a password reset is performed [Cromwell 2009]. Thus, the problem lies not with the authentication package - it's the static nature of the password hash! As long as that remains in play, a series of countermeasures will be required to thwart this attack. While the publicly available tools mentioned here are not effective against Kerberos authentication, history often has a way of repeating itself. When Paul Ashton wrote the first hash passing tool against LM, it was just a matter of time until another tool was developed that worked against NTLM. It would be foolish to think that Kerberos offers total immunity when the underlying issue is that of the static hash. In fact, within the latest release (v1.4) of the PSHTK there exists a "TODO" file which specifically mentions the desire to add support for other authentication methods such as Kerberos. Given that reasoning, the proposed solution will start by building an effective framework while exploring some of the finer points for implementation.

6.1. An Industry Solution

Defining what objectives will be achieved is the first step in developing a working solution. This first goal is to eliminate the use of the LM/NT hash as a valid means of user identification. This can be achieved through the use of digital certificates, biometric, or one time password information. The second goal is to replace the proprietary LM/NTLM authentication protocol with one that provides support for an alternative form of credentials. This requires an authentication protocol that is capable of supporting one of these identification types. The latest implementation of Kerberos (v5) is a protocol that can provide just this.

The basic framework for a readily available solution would need to consist of a 100% pure Kerberos deployment (no LM/NTLM) coupled with the certificate services provided by a network's Public Key Infrastructure (PKI). A proper certificate would have to exist on a smart card that is supported by the Windows client operating system. The client/server tandem of Windows XP and Server 2003 provide the minimum support for smart card services while Windows Vista and Server 2008 are the preferred tandem [Microsoft 2009]. The newer operating systems have been designed with smart card features that are unavailable in the older OS's. For those organizations running third-

party services using LM/NTLM authentication, each service would have to be re-configured to use Kerberos if support for that protocol is offered by the vendor. Also, various service accounts (IIS, Exchange, backup apps) will still require passwords so those would have to be identified and secured accordingly.

6.1.1. The Migration towards an Industry Solution

Depending upon the size of the environment, implementing the proposed industry solution could range from one of modest effort to one of herculean proportions bordering on impossible. Nonetheless, the potentially devastating effects from this attack clearly show why passwords alone cannot be used as a valid form of user identification. Even if the industry solution cannot be implemented, it should serve as a useful roadmap for the IT Services and Information Security organizations alike.

While most organizations will not be able to tackle the Kerberos and PKI efforts at the same time, a reasonable action would be to choose a migration path towards a pure Kerberos environment. Moving towards Kerberos provides the foundation for enabling smart card based services based on PKI. It's mid 2009 and there are two (almost three) generations of Windows operating systems that provide default Kerberos and smart card support. The identification of Windows systems running an OS older than XP/Server 2003 provides the minimum requirements for the migration plan. It is generally easier to replace or upgrade clients so those can be tackled first. The servers on the other hand, may be running some legacy application whose version requires it to be tied to that particular OS. Be very wary of such systems as they are often rife with vulnerabilities. Identifying and assessing such legacy systems is a good thing and it's highly recommended to place them in an "Isolation DMZ" where only the required network access is permitted. Doing so allows the business function to continue unimpeded while the risk to the rest of the organization is lowered. More importantly, it allows you to continue on with the identification and segregation of systems failing to meet the minimum requirements.

Once a full sweep of the internal network has been made, it's quite possible that multiple servers will be identified as candidates for isolation. Depending upon their physical location relative to one another, one or more of these Isolation DMZ's may be

required. The goal of such a DMZ is to create a single and separate Windows domain for all of these older systems so they can continue to use domain services via LM/NTLM. Using this design, the new Windows AD server should be placed in a protected "Application Vault" segment of its own. As these systems are peeled away from the main network, the migration to an all Kerberos environment can become an achievable reality.

After the internal sweep is completed, the identification and assessment of any Windows system in an Internet facing "Service DMZ" can commence. If a system is in a DMZ, assume the worse will happen and that it will be attacked and eventually compromised. This should be treated as a hostile network segment with no trust relationships, implied or otherwise, with systems on the internal network. Under no circumstance should these systems be active members of an organization's primary Windows domain. Consider a scenario where the attacker is in control of a Service DMZ system and is running a hash dumping tool in the background. At this point, it's just a matter of time before someone with the domain admin account logs on and their hash is confiscated. Counter this by placing such systems in a workgroup or an isolation domain via the AD server in the Application Vault. Remember that only systems in a domain can use Kerberos due to the AD requirement, so workgroup systems are restricted to LM/NTLM.

6.2. A Custom Solution

While the industry solution is a proactive technique aimed at eliminating the root of the problem, an organization may seek to complement this with another layer of defense or as an interim countermeasure. A custom solution could be developed using the following framework that would provide host based protection on Windows systems. The "solution" operates under the premise that a user's interactive logon session is tied to a unique set of credentials, and passing a different set of credentials to a remote resource represents a possibly malicious behavior. At a minimum, the proposed solution would be able to detect all hash passing attempts regardless of what authentication package (NTLM, Kerberos, etc.) was used. The first four items are hard requirements that are mandatory for bare minimum operation while the remainder is recommended to provide

maximum operation. It's assumed that the host is an IT asset of an organization and not that of an attacker. Additionally, the host is assumed to be in a secure state and free of any kernel level rootkits.

- The "solution" must exist on the host Windows OS and be able to obtain the active logon credentials for that session.
- The "solution" must be able to intercept all system calls for remote resource authentication requests.
- The "solution" must be able to extract the user credentials from these requests and compare them to those in the active session.
- The "solution" must be able to detect matching and non-matching credentials and provide a mechanism for event logging and alerting.
- The "solution" should be able to identify the source and destination addresses and service of the request.
- The "solution" should provide some level of customization to handle legitimate cases for the use of non-matching credentials.
- The "solution" should be capable of prevention in addition to detection.

There are several legitimate reasons that a user would supply a different set of credentials to access a remote resource. A user accessing a resource that is not in the same the domain (workgroup or untrusted domain) is one. Depending on the environment, this could occur when a domain user needs to establish a VPN to another network to access a resource. While troubleshooting desktop issues, support personnel may attempt to access remote resources during the user's logon session to avoid the multiple logon attempts. Lastly, in those organizations practicing the concept of least privilege, exceptions would be required to account for the use of the "runas" command.

6.3. DLL Injection Prevention

While not as robust as the custom solution, a system configured to prevent DLL injection attempts can be very effective at thwarting hash passing tools utilizing this technique. The countermeasure has an added benefit of preventing other nefarious

attacks as DLL injection is rarely used for legitimate purposes (except for application debugging). Within the Local Security Settings exists a user right called "Debug programs" or more accurately the SeDebugPrivilege privilege. As Figure 6.3.1 shows, this user right is assigned to the Administrators group by default. The default setting thus provides the environment for DLL injection as any user account with administrative rights has the ability to perform this action.

Figure 6.3.1 – Default local security policy setting for SeDebugPrivilege

The countermeasure for DLL injection is straightforward: remove all users and groups assigned to the "Debug program" policy setting. For those environments that have legitimate use cases, Figure 6.3.2 provides one possible way to limit the exposure while providing this user right. A local account (ex. DebugAdmin) can be created with this user right to be used in conjunction with the "runas" command to provide temporary privilege escalation to accomplish the task at hand.

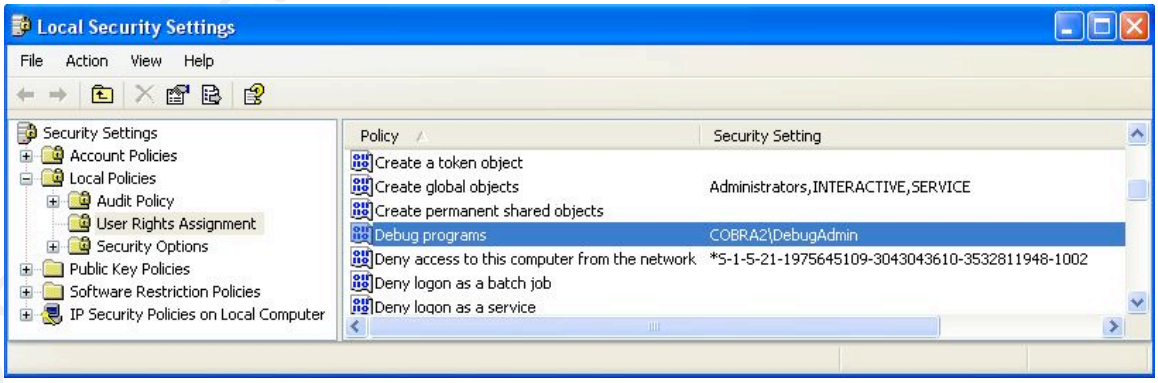

Figure 6.3.2 – Modified local security policy setting for SeDebugPrivilege

Figure 6.3.3 shows how effective disabling this user right can be. Msvctl, whosthere.exe, whosthere-alt.exe, and iam.exe all fail with various error messages as the current user logon session does not have the SeDebugPrivilege.

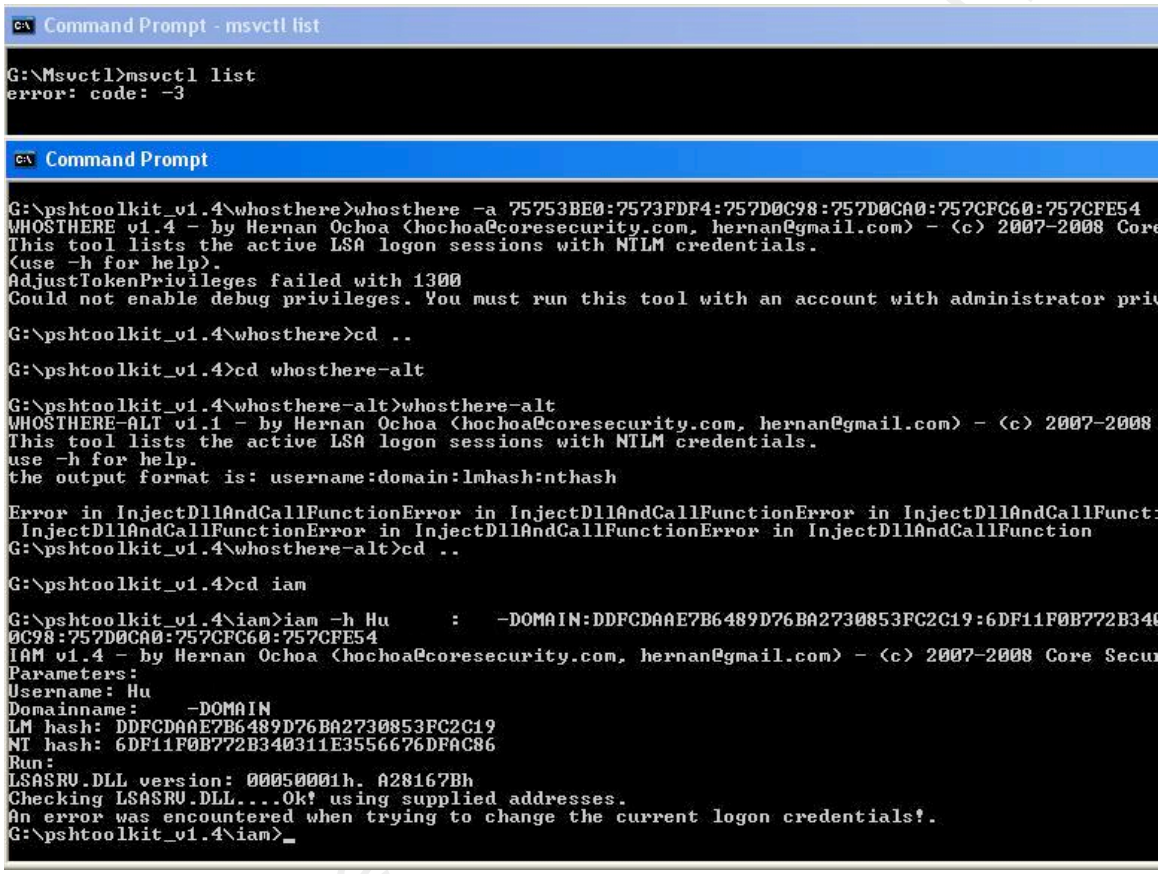

Figure 6.3.3 – Disabling the SeDebugPrivilege prevents DLL injection

6.4. Network-based IDS/IPS

The use of NIDS/NIPS may also be of value depending upon its placement within the network and its integration with other system logs for correlation by a SIM system. Such devices can be configured to identify the use of NTLM protocols on the wire and provide basic information on what system established a connection to another system. Combined with the event logs of the target system, the user account name that was attempted for authentication (granted or denied) can now be assigned to the source IP seen from the NIDS/NIPS logs. The ability to attribute an IP with a username allows for further investigation to determine if the system with that IP belongs to the same person whose username was seen transmitted from it. This type of detection would be most

useful for an organization's high value targets including systems and user accounts. A simple mapping between username and system address could allow the creation of signatures to detect when valid or invalid combinations are used. For example, if the domain admin account has been captured and its hash passed from some random internal system to a domain controller, the custom signature may be able to flag this as an unauthorized connection attempt as it was not sourced from the domain admin's system. The word "may" is used here as the traffic must pass through a network segment where the device is inspecting traffic.

This technology could also be used to identify possible remote hash dumping attempts. The device would need to key in on an SMB connection over TCP port 139 or 445, a connection to a Windows share, and an RPC call to the svcctl named pipe. A signature containing the svcctl call should suffice as its usage should be limited to authorized remote Windows administration. Connections made from non-admin source addresses or activity detected during suspicious off hours should raise a red flag. As the main hash files reside on the domain controllers, ideal NIDS/NIPS placement would allow inspection of all packets between this segment and the rest of the network.

6.5. Host-based Anti-Malware

Moving from the network to the host, endpoint security offerings providing anti-malware protection can provide a valuable defense against the hash passing attack. For those unfamiliar with Virus Total (www.virustotal.com), it is a free service that analyzes suspicious files using an impressive stable of anti-virus engines. Hosted by Hispasec Sistemas, the uploaded file is scanned by not just one, but all participating products who are updated on a continual basis. At the time of this writing, there were 41 products that were updated within a day of the scan date. Looking at Figure 6.5.1, the author supplied ten hash dumping and passing tools to be analyzed by the service. Please note that "servpw" is used in conjunction with pwdump6 when using the tool in a remote fashion. The goal was not to analyze all known tools used in this attack, but to gauge the detection rates against the more popular tools in use today.

Unfortunately, the results were rather poor as only two of the ten tools were detected by more than 80% of the products. There is a huge disparity between products

with only five products (by three vendors) achieving a perfect detection rate. It's important for every organization to conduct their own test by taking each of the hash dumping/passing tools and verify that their introduction to the OS is detected, removed, and the action logged. If it isn't, these items should be reported to the respective vendor for remediation. While such products are certainly better than nothing at all, it's crucial not to be lulled into a false sense of security with these solutions especially those based purely on signature detection. Products using signatures may fail to recognize variants whose executables have been altered using such freely available tools such as UPX or MPRESS. As a result, understand the technological underpinnings of the anti-malware solution and always conduct your own testing.

Antivirus	Version	Last Update	pwdump6 & servpw		fgdump	gsecdump	msvctl	genhash	whosthere	whosthere-alt	iam	iam-alt	Detection Rate
a-squared	4.5.0.24	2009.07.28	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	90%
AhnLab-V3	5.0.0.2	2009.07.28	-	-	-	Yes	Yes	Yes	-	Yes	-	Yes	50%
AntiVir	**7.9.0.234**	**2009.07.28**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**100%**
Antiy-AVL	2.0.3.7	2009.07.28	-	-	-	Yes	Yes	-	-	-	-	-	20%
Authentium	5.1.2.4	2009.07.28	-	Yes	Yes	-	Yes	-	-	Yes	Yes	Yes	60%
Avast	4.8.1335.0	2009.07.28	-	-	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	70%
AVG	8.5.0.387	2009.07.28	Yes	Yes	Yes	Yes	Yes	-	-	-	-	-	50%
BitDefender	7.2	2009.07.29	-	-	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	60%
CAT-QuickHeal	**10**	**2009.07.28**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**100%**
ClamAV	0.94.1	2009.07.28	-	-	Yes	Yes	-	-	-	-	-	-	20%
Comodo	1798	2009.07.29	Yes	Yes	Yes	Yes	Yes	-	-	Yes	-	-	60%
DrWeb	5.0.0.12182	2009.07.29	-	Yes	Yes	-	-	-	-	-	-	-	20%
eSafe	7.0.17.0	2009.07.28	-	-	Yes	Yes	Yes	-	-	-	-	-	30%
eTrust-Vet	31.6.6643	2009.07.28	-	-	-	-	-	-	-	-	-	-	0%
F-Prot	4.4.4.56	2009.07.28	-	Yes	Yes	-	Yes	-	-	Yes	Yes	Yes	60%
F-Secure	8.0.14470.0	2009.07.28	Yes	Yes	-	Yes	Yes	-	-	-	-	-	40%
Fortinet	3.120.0.0	2009.07.28	Yes	Yes	Yes	Yes	Yes	-	-	-	-	-	50%
GData	19	2009.07.29	-	-	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	70%
Ikarus	T3.1.1.64.0	2009.07.28	Yes	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	90%
Jiangmin	11.0.800	2009.07.28	-	-	-	Yes	-	-	-	Yes	-	-	20%
K7AntiVirus	7.10.804	2009.07.28	Yes	Yes	Yes	Yes	Yes	-	-	-	-	-	50%
Kaspersky	7.0.0.125	2009.07.29	Yes	Yes	Yes	Yes	Yes	-	-	-	-	-	50%
McAfee	**5691**	**2009.07.28**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**100%**
McAfee+Artemis	**5691**	**2009.07.28**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**100%**
McAfee-GW-Edition	**6.8.5**	**2009.07.29**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**100%**
Microsoft	1.4903	2009.07.28	-	-	Yes	Yes	-	-	-	Yes	-	-	30%
NOD32	4286	2009.07.28	Yes	Yes	-	Yes	Yes	-	-	-	-	-	40%
Norman	6.01.09	2009.07.28	-	-	Yes	-	-	-	-	-	-	-	10%
nProtect	2009.1.8.0	2009.07.28	-	-	Yes	Yes	-	Yes	-	-	-	-	30%
Panda	10.0.0.14	2009.07.28	-	Yes	Yes	Yes	Yes	N/A	-	Yes	Yes	Yes	78%
PCTools	4.4.2.0	2009.07.28	-	-	Yes	Yes	-	-	-	-	-	-	20%
Prevx	3	2009.07.29	-	Yes	Yes	Yes	-	-	-	-	-	-	30%
Rising	21.40.14.00	2009.07.28	-	-	-	Yes	Yes	-	-	-	-	-	20%
Sophos	4.44.0	2009.07.29	Yes	Yes	Yes	Yes	Yes	-	-	Yes	Yes	Yes	80%
Sunbelt	3.2.1858.2	2009.07.28	-	Yes	Yes	Yes	-	-	-	-	-	-	30%
Symantec	1.4.4.12	2009.07.29	-	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	80%
TheHacker	6.3.4.3.376	2009.07.28	-	-	-	Yes	Yes	-	-	-	-	-	20%
TrendMicro	8.950.0.1094	2009.07.28	-	-	-	-	-	-	-	-	-	-	0%
VBA32	3.12.10.9	2009.07.28	-	-	-	-	-	-	-	-	-	-	0%
ViRobot	2009.7.28.1857	2009.07.28	-	Yes	Yes	-	Yes	-	-	-	-	-	30%
VirusBuster	4.6.5.0	2009.07.28	-	Yes	-	Yes	Yes	-	-	-	-	-	30%
Avg. Detection Rate per Tool			37%	56%	73%	80%	83%	33%	15%	46%	37%	39%	

Figure 6.5.1 – VirusTotal analysis of hash dumping/passing tools

6.6. Host-based IDS/IPS

Another endpoint security solution that may provide some form of protection is HIDS/HIPS products. As the majority of these products are using a combination of

multiple attack detection technologies, trying to determine which ones can actually prevent (or at least detect) a hash passing attack can be quite difficult. While this class of products can thwart a variety of attack vectors, catching this particular type of attack requires knowledge behind the operation of the hash passing tools. As previously shown in Figure 6.3.3, it appears that some of the tool developers have chosen to use DLL injection to modify user credentials in memory. A HIDS/HIPS product that is able to detect and block DLL injection attempts would be able to defend against tools using this technique. For those tools using another technique, a product using anomaly (behavior) detection may be able to recognize potentially malicious activity. Because each vendor uses different ingredients in their secret sauce, hands-on product testing is the only way to truly determine if a particular HIDS/HIPS solution will provide an effective defense.

6.7. Other Countermeasures

Other defensive measures include several common approaches that should be a part of every organization's security plan. As an attacker must first gain high privileges (administrative or equivalent) to run these tools, bolstering host level defenses will make their task that much harder. One of the most basic and high value countermeasures is that of OS hardening and patching, especially those in a Service DMZ. A system with no vulnerabilities provides no avenue for exploitation, thus preventing system compromise. The patch management program should consist not only of the OS, but all applications as well. The latter of the two has traditionally been more difficult to implement due to the plethora of applications that are used within today's organizations. Enforcing some level of application control can go a long way towards resolving this in addition to using tools such as Secunia's Personal Software Inspector.

Another tactic is to restrict the privilege level of user accounts. In an ideal computing environment, all user accounts would be granted minimal non-privileged access. For those people requiring an elevated level of access (such as IT admins), the Windows "runas" option would be used so only that process is running under the context of the administrative account. The goal is to spend as little time as possible operating under the higher privileged account to reduce the potential attack surface. For those users requiring system administration accounts (such as domain admin), a separate account

should be created for that function. Additionally, these system administration accounts should only be used on those specific systems and not on general purpose systems. For example, use of the domain admin account should be limited to domain controllers only. Where possible, the concept of separation of duties should also be applied. Limit the broad privileges of a single account by creating multiple accounts to cover those areas of administrative responsibility. The intent is to provide some form of damage control should a privileged account fall into enemy hands.

6.8. Hash Theft Prevention

Last but certainly not least is the theft prevention of the hashes as without these the passing attack is not possible. For today's networks using NTLM and Kerberos authentication, the NT hash resides on disk (SAM files and registry) and in memory (LSASS logon sessions). A domain controller's ntds.dit file is the top prize, followed by the SAM files and active logon session credentials on domain systems. Additionally, attempts to obtain a hash can be made either locally or remotely. This provides a total of five combinations or avenues that an attacker can attempt to steal this valuable piece of information: local disk, local memory, remote disk, remote memory, and backup tapes.

The use of physical security applies to both local attacks as well as backup tapes that may contain ntds.dit and SAM files. Efforts should be made to prevent unauthorized access to the system or tape (i.e. door lock, badge swipe) as well as unauthorized access to a system's internal components (i.e. case lock). The use of disk encryption should be applied to all laptops, high value targets such as the domain controllers, and the backup tapes. If feasible, disk encryption for all remaining Windows domain systems should be considered. As for the backup tapes, these are very important as often times this is a third party arranged service with provided offsite storage. Organizations place a high degree of trust upon such providers to ensure that a requisite level of safeguards is applied to protect their assets. As such, a proper risk assessment of the backup process is strongly encouraged.

Another area of focus involves the protection of the main hash repository for an organization – the AD database (ntds.dit) on the domain controllers. All domain admins should be issued smart cards and the domain controllers configured for smart card only

logon. If time and resources permit, this solution should be applied to the other Windows servers throughout the network starting with the higher value targets. Within AD, there exists a group policy mechanism that allows a consistent Windows security policy to be applied to all domain systems. This can be used to prevent the creation of the older LM hash. While this should no longer be used, its existence has prevailed and due to its weak cryptography, it allows a captured hash to be easily cracked thus exposing the real password. Recommendations for very strong password generation include the use of passphrases greater than 14 characters consisting of upper/lower case, numbers, symbols, and select Unicode characters. One reason for this is that passwords greater than 14 characters are unable to be hashed by the LM mechanism. Additionally, the inclusion of at least one character from the Unicode character set will make cracking very difficult (but not impossible). A password that is not easy to guess or crack will delay and possibly prevent an attacker's attempt at escalating their privilege.

One additional defensive measure requires a balance between administrative overhead/usability and security. On one hand, a local administrator account configured with the same username and password provides a high level of usability. On the other hand, a single system compromise can quickly lead to mass system compromise using the hash passing technique. Followed by the smart card option, using a unique password for each local administrator account is the next best countermeasure that can be applied. While the administrative burden would certainly increase, there are many ways to create effective passwords using available information (via a creative combination) that is unique to each system such as MAC address, system asset tag, system node name, .etc. Organizations should develop a unique password derivation scheme that fits their environment's support model. Also, extra precautions should be taken to ensure that the local administrator account on high value systems do not resemble (name or password) those on general use systems. Logically, these systems belong in different security zones as their access requirements are fundamentally different.

Defending against remote hash collection requires several tactics. Removal of all Windows shares (IPC$ cannot be removed but it's not vulnerable) along with the disabling of the remote registry service will prevent attackers from sinking their hooks in. For those organizations that cannot do this due to required administrative tasks, granular

access controls must be applied at both the share level and NTFS directory level. Efforts must be made to deny unauthorized SMB file transfers to writeable shares. If this cannot be accomplished, the execute file permission will need to be explicitly denied throughout the share. Moving to the local security policy, a simple yet effective countermeasure that was discussed earlier in Section 6.3 is that of DLL injection prevention. While very effective at foiling the hash passing tools using this technique, it is also very effective at stopping the hash dumping tools as well. Although there is a noticeable overlap of tools amongst the groups, tool development techniques are few and DLL injection is a popular and relatively easy method for success.

7. Conclusion

While passwords continue to be prevalent, its use as a valid means of proving user identification continues to be a serious weakness. The attack exposes an underlying design flaw in Microsoft's password hashing implementation. As the hash passing tools are effective where ever LM/NTLM authentication is used, this type of attack is not limited to just SMB based services. While the author was unable to discover any publicly available tools that expose Kerberos, this will most likely change in the near future. The availability of these tools has opened Pandora's Box per se, as the countermeasures are both difficult and resource intensive. It is very difficult to protect Windows password hashes as they exist in running memory and in the registry on every Windows system. A thorough plan addressing physical, network, and host based security must be implemented to provide adequate levels of protection. The adoption of NTLM authentication by third party applications only compounds the problem as its enterprise presence significantly slows the migration towards an all Kerberos environment. Moving towards Kerberos provides the foundation for enabling smart card based services based on PKI. Additionally, the creation of a PKI should be evaluated and implemented beginning with those systems containing the most sensitive information. With the myriad of browser-based exploits, there's never been a time where gaining administrative level access to an internal system has been easier. Coupled with the post-exploitation hash passing technique, today's networks are facing a very serious threat.

8. References

Bugtraq (1997, April 8). *NT "pass the hash" with modified smb client vulnerability*. Retrieved from http://www.securityfocus.com/bid/233/info

CIAC (2008, May 21). *CIRCTech08-002: Understanding windows hash dumpers and crackers*. Retrieved from http://www.doecirc.energy.gov/techbull/CIRCTech08-002.shtml

Cromwell, B. (2009, January 2). *Authentication tools*. Retrieved from http://www.cromwell-intl.com/security/security-authentication.html

CVE (2008, September 10). *CVE - CVE-2008-4037 (under review)*. Retrieved from http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4037

De Clercq, J. (2007, March 26). *Comparing windows kerberos and ntlm authentication protocols*. Retrieved from http://windowsitpro.com/article/articleid/95577/comparing-windows-kerberos-and-ntlm-authentication-protocols.html

Gates, C. (2008, March 5). *Carnal0wnage blog: msvctl -- pass the hash action*. Retrieved from http://carnal0wnage.blogspot.com/2008/03/msvctl-pass-hash-action.html

Gates, C. (2009, March 3). *Dumping memory to extract password hashes*. Retrieved from http://carnal0wnage.attackresearch.com/node/123

Grutzmacher, K. (2007, May 9). *Superimposing nothing nowhere: pass the hash support for metasploit*. Retrieved from http://grutztopia.jingojango.net/2007/05/pass-hash-support-for-metasploit.html

Grutzmacher, K. (2008, August 8). *NTLM is dead – defcon 16*. Retrieved from http://squirtle.googlecode.com/files/NTLM%20is%20Dead%20-%20DefCon%2016.pdf

Gula, R. (2007, June 27). *Tenable network security: lm/ntlm hash support for smb credentials*. Retrieved from http://blog.tenablesecurity.com/2007/06/lmntlm-hash-sup.html

Hertel, C. (1999). *SMB: the server message block protocol*. Retrieved from http://ubiqx.org/cifs/SMB.html

Hertel, C. (2001, November 27). *Samba: an introduction*. Retrieved from http://us1.samba.org/samba/docs/SambaIntro.html

Jennings, L. (2008, April 14). *Security implications of Windows access tokens*. Retrieved from http://labs.mwrinfosecurity.com/publications/mwri_security-implications-of-windows-access-tokens_2008-04-14.pdf

Johansson, J. (2005, October 12). *Frequently asked questions about passwords*. Retrieved from http://technet.microsoft.com/en-us/library/cc512606.aspx

Johansson, J. (2006, August). *Security watch: The most misunderstood windows security setting of all time*. Retrieved from http://technet.microsoft.com/en-us/magazine/2006.08.securitywatch.aspx

JoMo-kun (2003, September 24). *Foofus networking services - pass the hash*. Retrieved from http://www.foofus.net/jmk/passhash.html

LCPsoft (2004, December 17). *Theory and practice of password auditing and recovery in windows nt/2000/xp/2003*. Retrieved from http://www.lcpsoft.com/english/articles/passwords.htm

Marchand, J. (2005, October 6). *Windows built-in remote administration tools*. Retrieved from http://www.governmentsecurity.org/forum/index.php?showtopic=28152

McClure, S., Scambray, J., & Kurtz, G. (2008). *Hacking Exposed 6: Network Security Secrets & Solutions*. New York, NY: McGraw-Hill.

Melber, D. (2004, October 28). *Protect against weak authentication protocols and passwords*. Retrieved from http://www.windowsecurity.com/articles/Protect-Weak-Authentication-Protocols-Passwords.html

Microsoft (2003, March 28). *What is kerberos authentication*. Retrieved from http://technet.microsoft.com/en-us/library/cc780469(WS.10).aspx

Microsoft (2005, May 23). *Understanding windows service architecture*. Retrieved from http://technet.microsoft.com/en-us/library/aa998749(EXCHG.65).aspx

Microsoft (2007, December 3). *How to prevent windows from storing a lan manager hash of your password in active directory and local sam databases*. Retrieved from http://support.microsoft.com/kb/299656

Microsoft (2008, February 8). *Windows authentication*. Retrieved from http://technet.microsoft.com/en-us/library/cc755284(WS.10).aspx

Microsoft (2008, May 8). *How the kerberos version 5 authentication protocol works*. Retrieved from http://technet.microsoft.com/en-us/library/cc772815(WS.10).aspx

Microsoft (2008, November 11). *Microsoft security bulletin ms08-068 – Important*. Retrieved from http://www.microsoft.com/technet/security/Bulletin/MS08-068.mspx

Microsoft (2009). *Windows vista smart card infrastructure*. Retrieved from http://msdn.microsoft.com/en-us/library/bb905527.aspx

Montoro, M. (2009). *Cain & abel user manual*. Retrieved from http://www.oxid.it/ca_um/

Moore, HD. (2007, August 13). *Tactical exploitation*. Retrieved from http://www.metasploit.com/data/confs/blackhat2007/tactical_paper.pdf

Murray, M. (2007, March 16). *Hash injection attacks in a windows network - Marcus Murray's blog*. Retrieved from http://truesecurity.se/blogs/murray/archive/2007/03/16/why-an-exposed-lm-ntlm-hash-is-comparable-to-a-clear-text-password.aspx

Ochoa, H. (2007, August 15). *Pass-the-hash toolkit - docs & info*. Retrieved from http://oss.coresecurity.com/pshtoolkit/doc/index.html

Ochoa, H. (2008, March 6). *Carnal0wnage blog: msvctl -- pass the hash action*. Retrieved from http://carnal0wnage.blogspot.com/2008/03/msvctl-pass-hash-action.html

Ochoa, H. (2008, July 2). *Release of pass-the-hash toolkit v1.4*. Retrieved from http://hexale.blogspot.com/search?updated-min=2008-01-01T00%3A00%3A00-02%3A00&updated-max=2009-01-01T00%3A00%3A00-02%3A00&max-results=21

Ochoa, H. (2008, October 29). *Pass-the-hash toolkit for windows implementation and use*. Retrieved from http://www.coresecurity.com/files/attachments/Ochoa_2008-Pass-The-Hash.pdf

Skoudis, E. (2008). *Computer and network hacker exploits*. Bethesda, MD: SANS Institute.

Skoudis, E. (2009, January). *The ethical hacker network - santa claus is hacking to town - answers and winners*. Retrieved from http://www.ethicalhacker.net/content/view/230/1/

Smith, R. (2003, August 18). *Access denied: using passwords with kerberos*. Retrieved from http://windowsitpro.com/article/articleid/39683/access-denied-using-passwords-with-kerberos.html

van Hauser (2006, May 5). *THC-Hydra – fast and flexible network login hacker*. Retrieved from http://freeworld.thc.org/thc-hydra/

Warlord (2006). *Attacking ntlm with precomputed hashtables*. Retrieved from http://uninformed.org/?v=3&a=2&t=sumry