A single NT hash can be used to access almost any data which resides in a Windows domain environment. The theory behind the first practical "Pass the Hash" attack against Microsoft Windows NT and the Lan Manager (LM) protocol was posted to NTBugtraq in 1997 by Paul Ashton[1]. The result was a patched Samba client that would accept a user's LM password hash to connect to a Windows share. This attack demonstrated that the LM password hash was equivalent to the actual plaintext password for login purposes. The same thing holds true for the NTLM protocol, the successor to the LM protocol. An attacker can use password hashes to log in and access services without needing to know a user's password. While Microsoft's Kerberos doesn't treat the password hashes as equivalent to user passwords, it does use the password hashes as encryption keys. Therefore, a compromise of a domain's password hashes will undermine the additional protections provided by Kerberos.

The following is an outline of topics covered in this whitepaper:

- Password Hash Overview
- Windows Authentication Overview
- NTLM Overview
- Kerberos Overview
- Passing the Hash with NTLM
- Undermining Kerberos Security with Password Hashes
- Mitigations and Defenses
- Closing Thoughts
- About the Authors
- Appendix 1: Open Source Tools to Access Microsoft Services
- Appendix 2: References

Password Hash Overview

Microsoft Windows generates two different password hashes depending on security configuration, compatibility level and password length: Lan Manager (LM) hashes and NT hashes. Windows will generate both hashes depending on the length of the password and other security policies and settings.

Lan Manager (LM) hashes[2] are still supported for backwards compatibility (Windows NT and earlier) and suffer from a number of well-documented security vulnerabilities. Due to a limitation of the specification, passwords with a length greater than 14 characters can not be stored as a LM hash. Assuming 14 characters or less, the password is first converted to uppercase. This reduces the possible characters used in each position of the password from 95 (26 upper, 26 lower, 10 digits, 32 punctuation including space) to 68 possible characters (26 upper, 10 digits, 33 punctuation/symbol including space). To make matters worse, the password is null padded to 14 characters and split into two seven-byte pieces. This reduces the keyspace from 68^14 to 68^7, changing the scale of brute-forcing all possible passwords from trillions of years to 2 months or less on modern computing hardware. Each half of the password is then converted into a bitstream and a zero is inserted after every 7th bit. This bitstream is then used as a DES encryption key to encrypt the fixed string 'KGS!@#$%' into an 8 byte hash value. Both halves of the password are then hashed independently and the results concatenated into a single 16 byte hash value. Aside from brute-forcing, the LM hash is also susceptible to attack from precomputed lookup tables called rainbow tables, but this is outside the scope of this paper.

LM hashes are not stored to disk per default security policy in most modern versions of Windows. However, internally in memory, the LM hash will still be generated when a user logs in along with the NT hash if the password is 14 characters or less. A malicious attacker can use

a program such as the “Windows Credential Editor” from Hernan Ochoa to retrieve both hashes from memory for all logged-in users on a computer.

The NT password hash[3] doesn’t suffer from the same length or character restrictions that the LM hash does. The passwords can be up to 127 characters in length and can contain uppercase and lowercase characters without penalty. The hash is also a single 16 byte hash instead of two 8-byte halves.

To create the NT hash,the password is first converted to Unicode and then passed through the MD4 hash function, yielding a 16-byte hash value. While the hash function used is computationally inexpensive, it is currently infeasible to brute force the entire keyspace up to the maximum length.

Windows Authentication Overview

Individual Windows-based services do not directly support Kerberos or NTLM[4]. Instead Microsoft supported services generally use an API called Generic Security Services Application Program Interface, or GSSAPI[5], to negotiate what authentication methods are available to be used. In modern versions of Windows, these services default to using “Negotiate” as the authentication method. “Negotiate” allows the client to decide between either Kerberos or NTLM for the authentication method used. Many different services are capable of using GSSAPI to determine the authentication method to include MSSQL, Sharepoint, IIS, and Exchange, which supports legacy IMAP, SMTP, and POP3 protocols.

While modern-implementations of Active Directory default to utilizing Kerberos, there are several circumstances when NTLM is used to avoid authentication failure:

- Client is authenticating via IP address
- Client is authenticating to a server in a different domain with a legacy NTLM trust
- Client is authenticating to a server not in the domain (or vice versa)
- Firewall restriction on the ports required for Kerberos
- DNS or the Domain Controllers are unavailable

NTLM Overview

In the Wikipedia entry for NTLM it is described as “a suite of Microsoft security protocols that provides authentication, integrity, and confidentiality to users.” NTLM consists of multiple protocols, although we are primarily concentrating on NTLM version 2 (NTLMv2) since it is widely deployed.

Authentication with NTLM consists of the following message exchanges[6]:

- Type 1 Message: The client sends a message detailing its own capabilities and requesting capabilities from the server. The client can optionally send the domain the client is in, the name of the client and the OS version of the client.
- Type 2 Message: The server responds to the client with the supported capabilities and a challenge issued to the client. The server can optionally specify the server name, the server’s domain, the server’s FQDN, and the DNS domain name.
- Type 3 Message: The client responds with information about the user, the domain authenticating to, and one or more responses to the challenge issued by the Type 2 message. The response provides indirect proof that the user and password combination

are valid. The server independently calculates the expected responses and validates that the information from the Type 3 message is accurate.

The difference between the version NTLMv1 and NTLMv2 lies in the Type 3 message, the client's response to the server's challenge.

The NTLMv2 Type 3 message is constructed with the LMv2 response and the NTLMv2 response.

The LMv2 response is constructed in the following manner::

1. Obtain the NTLM password hash.
2. The Username is converted to uppercase and represented as Unicode and concatenated with the authentication target information also represented in Unicode.
3. The HMAC-MD5 message authentication code is applied to the Unicode text from step 2 with the NTLM password hash being used as the key. This portion is referred to as the NTLMv2 hash.
4. A random 8 byte nonce is concatenated with the server challenge from the Type 2 message.
5. The HMAC-MD5 message authentication code is applied to the result of step 4 using the NTLMv2 hash (step 3) as the key.
6. The result from step 5 is concatenated 8 byte client nonce to form the LMv2 response.

The NTLMv2 response is constructed in the following manner::

7. Obtain the NTLM password hash
8. The Username is converted to uppercase and represented as Unicode and concatenated with the authentication target information also represented in Unicode.
9. The HMAC-MD5 message authentication code is applied to the Unicode text from step 2 with the NTLM password hash being used as the key. This portion is referred to as the NTLMv2 hash.
10. A data 'blob' is created using preformatted text, a timestamp, a random client nonce, and the target information from the Type 2 message.
11. The server challenge from the Type 2 message is concatenated with the data blob and the HMAC-MD5 message authentication code is applied using the NTLMv2 hash (step 3) as the key.
12. The result from step 5 is concatenated with the data blob to form the NTLMv2 response.

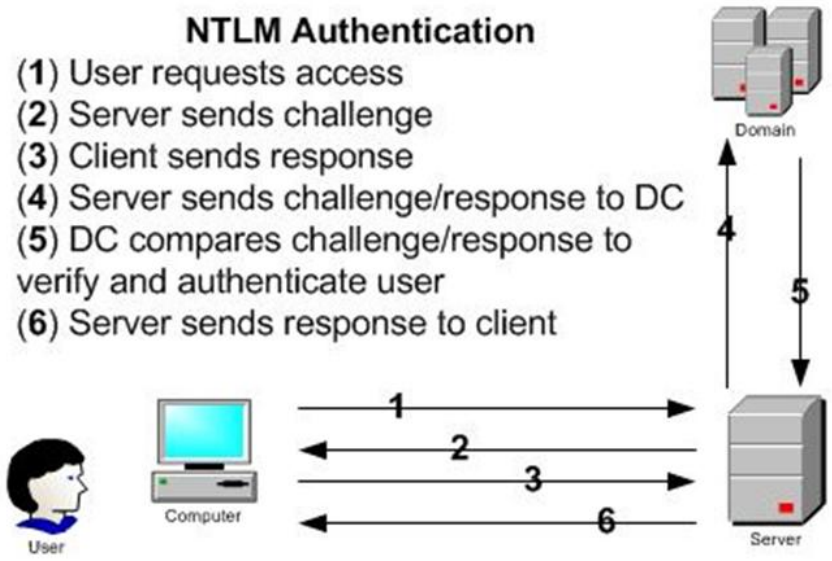

Figure 1 - NTLM Authentication Overview

Kerberos Overview

Kerberos[7] is a client-server authentication protocol used by Windows Active Directory which provides mutual authentication to all parties. Entities who authenticate or request services from each other are called "principals". The central server involved is called the Key Distribution Center, or KDC. The KDC consists of two services, the Authentication Server (AS) and the Ticket Granting Service (TGS). In a Windows domain environment both the AS and TGS services run on any writable domain controller[8].

Kerberos principals communicate through the use of tickets issued by the KDC.

A principal (user) requests access to a service from Active Directory in the following manner:

1. The client hashes the password for the user. This becomes the long term secret key used between the client and the AS.
2. The client encrypts a timestamp and sends it to the AS. The AS will decrypt the timestamp, and if successful, this demonstrates that the client knows the password for a particular user.
3. The AS replies to the client with 2 pieces of information:
    a. Short term encryption key to be used for future requests from the KDC, encrypted with the client's hash
    b. the Ticket Granting Ticket, TGT, which contains information regarding the user, the domain, the time and group membership. This information is encrypted with a key that only the TGS knows. Note that the KDC doesn't remember state. The TGT is blindly forwarded every time the client requests access to a service.
4. Once the client has the TGT, the client constructs the request for the particular principal using the session key from the TGT response. The client blindly forwards the TGT along with its request to the TGS.
5. The TGS decodes the TGT and the service request and if the request is approved, sends a service ticket containing two parts back to the client:

a. Section for the remote server - this section contains the user's group membership, a timestamp, session key for communications with the client. This section is encrypted with the KDC's key for the server.
b. Section for the client - this section contains the session key for communication between the client and the remote server. This section is encrypted with the key from the AS reply from step 3.

6. The client sends the server portion of the service ticket to the server along with its request. The server will accept the service ticket without direct communication with the KDC because the ticket is encrypted with the long term key used for communication between the remote server and the KDC. This implies that the KDC has approved the communication.

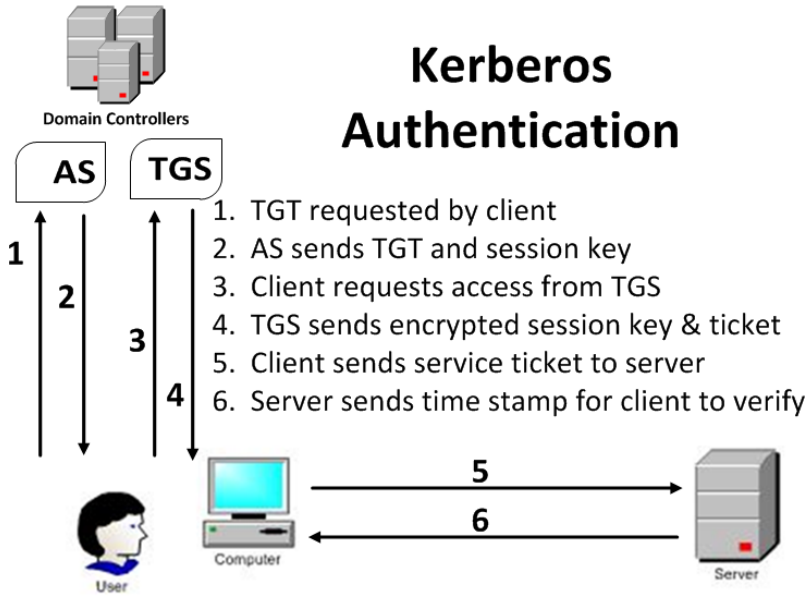

Figure 2 - Kerberos Authentication Overview

Central to the trust model of Kerberos is the notion that each principal communicates with the KDC in a secure manner using only keys that exist between the principal and the KDC. When principals communicate with each other, they use session keys assigned by the KDC.

Kerberos allows an alternate form of authentication using PKI and smart cards[9]. Instead of a password, the user is prompted for a PIN for the smart card. Windows uses the PIN to access the public key certificate on the smart card. The certificate is signed by the private key on the smart card and sent to the KDC. The KDC verifies the signature on the certificate was signed by a trusted entity. The KDC then sends the TGT encrypted with the public key certificate. Since the information can only be decrypted by the private key on the smart card, the user is authenticated to the domain. However, password hashes are still stored on the Domain Controller for the accounts that use smart card authentication. In addition, smart cards only provide protection for "interactive sessions". This means that smart card authentication can only be used to log into a computer that is a member of the domain. Network access to services can still require the use of a password.

Passing the Hash with NTLM

The NTLM protocol specifies that the password hash must be used as the cryptographic key in the HMAC-MD5. Windows needs to cache the password hash in order to use it for other authentications. This means that if the password hash for the user is stored in memory, Windows only has to ask the user for their password upon initial login. The password hash can be stored in memory and used whenever the user requests access to services. In effect, Windows passes the hash routinely during NTLM authentication.

Under normal circumstances, a malicious user cannot specify a particular hash they wish to impersonate under Windows. However, thanks to Hernan Ochoa, the Windows Credential Editor (WCE)[10] can be used to change the password stored in the access token for the currently logged in user. This allows Windows to use the native API calls to authenticate. This also means that native Microsoft tools including the Active Directory management tools, Internet Explorer, Outlook, and the MS SQL client can be used to access resources available to a different user by using the password hash of that user. WCE requires administrator level permissions on the client machine. Full details of what WCE does behind the scenes is available at Amplia Security's website.[11]

Alternatively under Linux, the source code for the open source tools mentioned previously can be modified to allow the hash to be specified instead of the password. The original Samba patch was released by JMK[12] from foofus.net and allowed the password hash to be specified in an environmental variable named "SMBHASH". By setting the environmental variable and then executing the requested Samba command-line tool, the hash would be substituted for the password when using NTLM authentication.

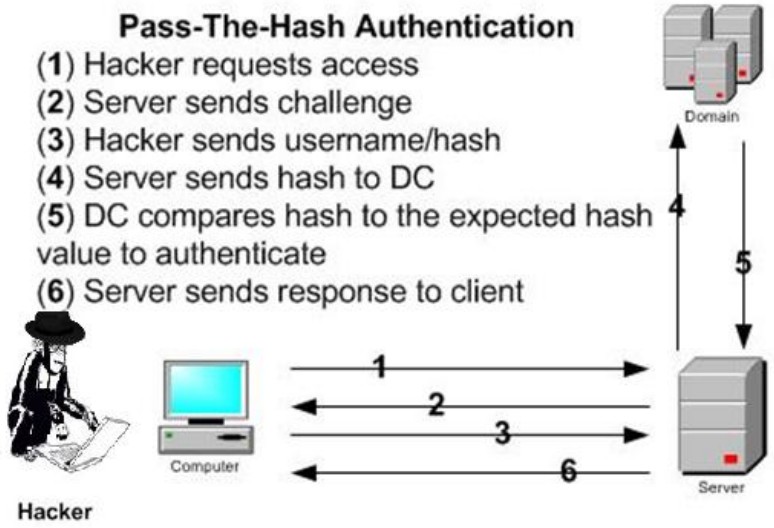

Figure 3 - Pass-The-Hash NTLM Authentication

The approach of the environmental variable has a major drawback when it comes to using other utilities other than Samba. If a tool such as Firefox wanted to pass the hash, the entire instance of Firefox would need to be killed in order to change the username and hash combination. In addition, if somebody wanted to use these utilities to script an attack on a Windows service, constantly having to re-export the environmental variable is annoying. To address these issues, a new form of receiving the hash was implemented.

The two main forms of password hashes used are either 65 characters or 68 characters in length. The 65 character form has the 32 hexadecimal character LM hash, followed by a single colon, followed by the 32 hexadecimal NT hash. The 68 character variety appends three colons after the 65 character version to mirror the output from several popular Windows hash dumping programs. Our patches will accept either the 65 or the 68 character password hash as the password passed to the program. So, for example in Firefox, when accessing a website using NTLM authentication, you provide the username and substitute either the 65 or 68 character hash for the password in the popup. The hash calculation function detects that a 65 or 68 character password was used and attempts to substitute the appropriate hash values in the NTLM authentication. Limited error-checking is performed since it is assumed that both 65/68 character passwords are rare and the tool is only going to be used by a penetration tester while on assessment.

We have released patches for Samba, OpenChange, FreeTDS, Firefox, and curl that allow the user to specify password hashes instead of passwords. Appendix 1 has brief descriptions of each of the projects. Examples of how to use these utilities can be found on our blog along with examples and tips on use.

Undermining Kerberos Security with Password Hashes

Careful examination of the Windows specific implementation of Kerberos[13] indicates that the so called “long term secret key” is in fact the NT hash for the account to include service and computer accounts. The main key used by the KDC to encrypt and sign tickets is the NT hash for the KRBTGT account. Hernan Ochoa’s Windows Credendial Editor (WCE) has the capability to both export existing tickets from and insert tickets into the Windows operating system. Armed with the password hashes and WCE, any Kerberos ticket can be created, appropriately encrypted and signed, and inserted into the OS for use. A theoretical attack follows.

There are 3 domain machines involved in the attack:

- Domain Controller (DC1)
- The Client (Yakko)
- The File Server (FS1)

We are going to violate the trusts involved in Kerberos in such a way that a regular unprivileged user named ‘Bob’ can access FS1 as if he were in the “Enterprise Admins” group.

Step 1. Request a TGT from DC1 for user Bob on Yakko

TGTs contain 2 sections

1. TGT section (Encrypted with KRBTGT hash) which contains:
    a. Bob’s username and Realm
    b. The name of the client, Yakko
    c. Validity period for the ticket

    d. Bob / DC1 session key for use until the TGT expires
    e. Privilege Access Certificate (PAC)[14,15], a list of groups that Bob belongs to in the domain
2. Bob / DC1 session key (Encrypted with Bob's Hash)

We have all the information necessary to generate our own TGT without involving DC1. However, we can use information in the DC1 TGT to generate a session ticket with FS1 more easily.

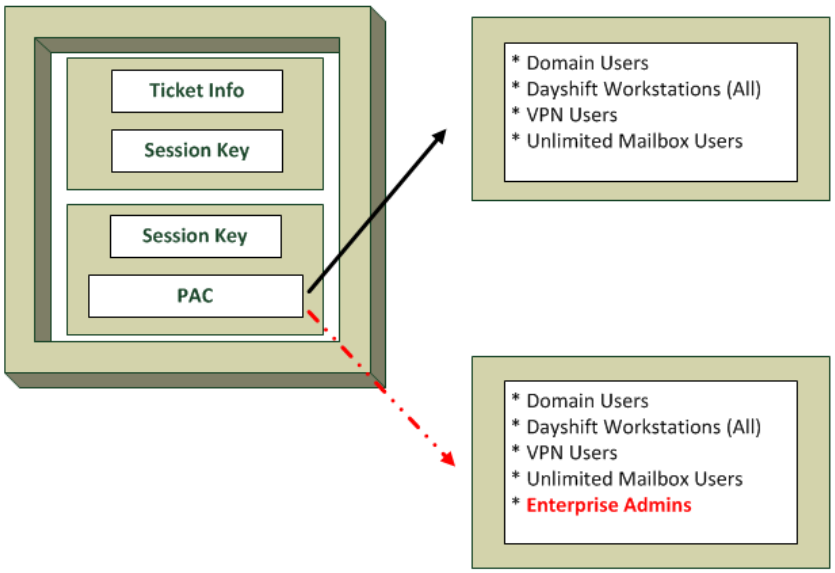

Figure 4 - Create Spoofed PAC

Step 2. Generate the Service ticket for Bob and FS1

The Service ticket contains 2 sections:

1. Section for Bob (encrypted with the Bob/DC1 session key) which contains:
    a. A session key to use between Bob/FS1
    b. Client's nonce, (which we never sent, because we created this ticket)
2. Section for FS1 (encrypted with the hash for FS1) which includes:
    a. The same session key to be used with Bob/FS1
    b. Bob's name and Realm
    c. Timestamp the ticket was generated
    d. Bob's PAC, signed twice, once with FS1's key and the other with DC1's key

Decrypting the TGT sections with the requisite keys yields Bob's PAC along with the session key between Bob/DC1. Using this information we can construct a new Service ticket for Bob/FS1.

We modify the PAC to include the "Enterprise Admins" group and sign the updated PAC with both DC1's key and FS1's key. We then package everything up and encrypt both parts of the ticket with their respective keys. We can now insert the ticket into Yakko's memory using WCE.

Step 3. Request services from FS1 as Bob from Yakko

Now we present the falsified ticket we generated to FS1. FS1 erroneously validates that the ticket came from DC1, since it is encrypted with the hash for FS1 even though we spoofed the ticket. The PAC also passes muster since it is signed by the appropriate key, and therefore Bob gains access to the services on FS1 as a member of the "Enterprise Admins" group. The entire trust model of Microsoft's implementation of Kerberos has been completely undermined.

As of right now this attack is purely theoretical as we haven't had time to work on implementing it. It's also overkill given that we can trivially exploit practically anything we want on the local network using the password hashes along with the NTLM protocol. However, we are investigating how to implement the Kerberos attack moving forward.

Mitigations and Defenses

A common misconception is that security tools and configuration changes can prevent this type of attack from being successful. Unfortunately, most antivirus tools are looking for specific, known attack tools and their behavior. For example, several antivirus products flag Hernan Ochoa's WCE as malicious and remove it. Other products detect the use the Metasploit PSEXEC exploitation module[17] due to the service it starts. However, AV products can be trivially bypassed, in many cases, by encoding or otherwise obfuscating the binary in question[16]. On the network side, a successful "network login" event is virtually indistinguishable from one created with a hash. Up until recently (Windows 2008 R2)[18] there wasn't a distinction between NTLM or Kerberos being used for logins that was noted in the event logs.

The ultimate mitigation against the "Pass The Hash" attack is to protect the hashes from being disclosed to an attacker. However, there are steps that can be taken to lessen the chance of a successful attack. The first is regular password changes. NT hashes have limited value once the corresponding account's password has been changed. Frequent password changes forces an attacker to repeat their attack which could be detected. Audit log review and monitoring of privileged accounts is also critical to detecting the attack. Networks should be segmented to prevent lateral movement by attackers. Administrators should also prevent external services from being accessed from external hosts. Sensitive ports such as SMB (TCP 445) and NetBios (TCP 139) should be blocked and integrated authentication should not be used by externally-accessible websites (e.g. SharePoint).

Closing Thoughts

The "Pass The Hash" attack is actually a documented part of the way that Windows protocols interact. Since the designers of the protocols didn't want to ask the user for their username/password every time network authentication takes place, the password hash is stored locally in memory and is constantly being recycled as long as the user stays logged in. Because it's an integral part of the design of Windows authentication, the "attack" does not have a traditional defense. The best way to prevent the attack from happening is to protect the Domain Controller from compromise with a comprehensive defense in depth approach. Enforce least privilege, enable UAC, limit the number of elevated accounts, etc. Once the DC is lost, all your data becomes accessible.

About the Authors

Alva 'Skip' Duckwall (@passingthehash) has been using Linux back before there was a 1.0 kernel and has since moved into the information security arena doing anything from computer/network auditing, to vulnerability assessments and penetration testing. Skip currently holds the following certs: GSE, CISSP, CISA, and OSCP, among others. Skip currently works for Northrop Grumman as a Sr. Cyber Something or other. Skip can be reached at exorcist@gmail.com.

Chris Campbell (@obscuresec) has worked for Northrop Grumman as a full-scope penetration tester for several years. He holds many industry certifications and a Master of Science in IA from Capitol College. Chris served over ten years in the Army with most of that time as a Signal Officer. Chris can be reached at obscuresec@gmail.com.

Our Blog: http://passing-the-hash.blogspot.com
Google Code: http://code.google.com/p/passing-the-hash/

Appendix 1: Open Source Tools to Access Microsoft Services

Several open source projects exist in an attempt to bridge the gap between the open source world and the Microsoft world. Because of their usefulness and their available source code, we were able to introduce patches to allow hashes to be specified instead of passwords.

Samba - **http://www.samba.org/**

From the Samba website: “Since 1992, Samba has provided secure, stable and fast file and print services for all clients using the SMB/CIFS protocol, such as all versions of DOS and Windows, OS/2, Linux and many others.”

Samba specifically provides libraries to interface with Microsoft DCE/RPC services on Windows machines. In addition, Samba also provides Linux command line tools with similar functionality to windows command line tools to interact with a Windows domain. As part of the release of our Blackhat USA 2012 talk and associated utilities, we have put together a “Rosetta Stone” to provide examples of how to use both the Windows and Samba commands to perform tasks in Windows Domains such as access shares, stop/start/create services, manage users and other common tasks.

Openchange - **http://www.openchange.org/**

From the website: “OpenChange aims to provide a portable Open Source implementation of Microsoft Exchange Server and Exchange protocols.”

Openchange integrates tightly with Samba and provides libraries and client utilities to interact with Microsoft Exchange servers over the DCE/RPC access provided by the Samba libraries. Openchange allows the user to send/receive email, view contacts, create appointments, etc from the linux command line. Evolution, the GNOME-based email program, has a plugin available to tie in with Openchange as well.

FreeTDS - **http://www.freetds.org/**

From the website: “FreeTDS is a set of libraries for Unix and Linux that allows your programs to natively talk to Microsoft SQL Server and Sybase databases.”

FreeTDS is an open source implementation of the Tabular Data Stream protocol, which is the client protocol used to connect with both Sybase and Microsoft SQL databases. FreeTDS only provides an API to connect to the databases. There are modules for ruby, python and perl that use FreeTDS. SQSH, the SQL Shell, is a command line utility that attempts to act as a Linux replacement for isql, a Sybase utility.

Firefox - http://mozilla.org

Firefox, the venerable web browser has built-in support for NTLM authentication used by web sites using NTLM authentication such as Sharepoint or Outlook Web Access. By default Firefox will attempt to use the native NTLM authentication mechanisms. However, there is an internal NTLM authentication implementation that can be enabled using the configuration page available by typing “About:config” into the address bar.

Curl - **http://curl.haxx.se/**

From the website: “curl is a command line tool for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP.”

Curl is an open sourced command line tool that supports multiple protocols as well as multiple authentication types including NTLM.

Appendix 2: References

1. http://en.wikipedia.org/wiki/Pass_the_hash
2. http://en.wikipedia.org/wiki/LM_hash
3. http://en.wikipedia.org/wiki/NTLM
4. http://en.wikipedia.org/wiki/Windows_authentication
5. http://en.wikipedia.org/wiki/GSSAPI
6. http://davenport.sourceforge.net/ntlm.html
7. http://en.wikipedia.org/wiki/Kerberos_%28protocol%29
8. http://tools.ietf.org/html/rfc3244
9. http://technet.microsoft.com/en-us/library/dd277362.aspx
10. http://www.ampliasecurity.com/research/wcefaq.html
11. http://www.ampliasecurity.com/research/WCE_Internals_RootedCon2011_ampliasecurity.pdf
12. http://www.foofus.net/?page_id=55
13. http://msdn.microsoft.com/en-us/library/cc233855.aspx
14. http://msdn.microsoft.com/en-us/library/cc237917%28v=prot.13%29
15. http://blogs.msdn.com/b/openspecification/archive/2010/01/01/verifying-the-server-signature-in-kerberos-privilege-account-certificate.aspx
16. http://www.offensive-security.com/metasploit-unleashed/Antivirus_Bypass
17. http://www.metasploit.com/modules/exploit/windows/smb/psexec
18. http://blogs.technet.com/b/askds/archive/2009/10/08/ntlm-blocking-and-you-application-analysis-and-auditing-methodologies-in-windows-7.aspx