GETTING USER CREDENTIALS IS NOT ONLY ADMIN'S PRIVILEGE

Anton Sapozhnikov
@snowytoxa
SyScan 2014

Abstact

Everybody is familiar with such a great tools like mimikatz and wce. Everybody knows that password hashes are stored inside SAM. Fgdump, pwdump are also cool tools. But the sad part of this story is you need SYSTEM or equivalent privileges on attacked machine to run all mentioned arsenal.

Unfortunately if you have only user level access to machine inside corporate internal network that means you have quite limited number of ways to get password of that user which machine were infected by your tool. Already known techniques require additional access to network or great amount of luck.

This paper covers a brand new technique to grab credentials from pwned machine even without admins privileges. Having no access to internal network and absence of admin privileges is common case during spear phishing attacks and social engineering activities. The technique is possible due to design flow in Windows SSPI implementation. Proof of concept tool is also provided.

Problem definition

My daily job is doing penetration tests, a lot of pentests. During one of such projects I had an idea of attack that described in that paper.
Consider the following scenario: during the penetration test remote access to the user's workstation was obtained. My goal was to find out the password of the current user.

To achieve defined goal, there are various tools known and most popular of them are mimikatz and windows credentials editor (wce). Such tools previously allowed getting the user's password hash and now the plain text password.

The disadvantage of these tools is that attacker must have the privilege SeDebugPrivilege to work properly, usually such a privilege is owned by system administrators only.

As it usually happens, users do not have administrative privileges on their computers, in addition, all updates and patches have been installed that excluded the exploitation of any well-known vulnerability to elevate privileges in the system.

Next step could be searching for third-party services with weak file system or configuration permissions, as well as potential victims for DLL-hijacking attacks, but time was very limited, and brief analysis returned no results.

Thus, the elevation of privileges on the local system fails, and use mimikatz or wce remains impossible.

Alternative ways

Are there any other possibilities to gather current user password?

It could be, for example, showing the user a window asking him to enter his password for the corporate domain. Unfortunately this method requires user's action that can be ignored. Also, the user could enter invalid password or not those that needed. The most negative consequence for attacker may be attraction the user's attention to himself, which can lead to calling security staff and attack detection.

As far as Microsoft Windows have got Single Sign On (SSO) mechanism, we could use it to leak authentication data to us.

To do that, we should make Windows authorize on some system and intercept data transmitted over network. There are two protocols for which transparent user authentication implemented - HTTP and SMB.

If the user himself or with attacker's help will go to some host on the network, attacker could intercept network traffic and extract authentication data, which then use to get password.

SMB

Described earlier method is implemented, for example, in the Metasploit Framework incognito module with snarf_hashes:

```
msf  auxiliary(smb) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > snarf_hashes 192.168.0.107
[*] Snarfing token hashes...
[*] SMB Captured - 2014-03-23 11:54:38 -0400
NTLMv2 Response Captured from 192.168.0.99:1209 - 192.168.0.99
USER:user DOMAIN:TESTBOX2 OS:Windows 2002 Service Pack 3 2600
LM:Windows 2002 5.1
LMHASH:e3b8d79bdf9b63d5bc6f3155c83bfb78
LM_CLIENT_CHALLENGE:4a44bb4da09c6401
NTHASH:7d082900120d95a3ecca4c219ae55a1a
NT_CLIENT_CHALLENGE:0101000000000000040674a32b046cf014a44bb4da0
9c64010000000020000000000000000000000
…
[*] Done. Check sniffer logs
```

To intercept data transmitted over the network, attacker need to have sufficient privileges on the user's system. Since attacker doesn't have, it remains the only option that attacker has high privileged access to another host on the network, but it does not coincide with the terms of our problem.

HTTP

The HTTP protocol can use the built-in Windows authentication. Web browser can authenticate the current user on the server using its domain account.

It may be something like the following:

c--->s	GET /vy6BSjy HTTP/1.1
c<---s	401 Unauthorized HTTP/1.1 WWW-Authenticate: NTLM
c--->s	GET /vy6BSjy HTTP/1.1 Authorization: NTLM TlRMTVNTUAABAAAAB7IIogkACQAwAAAACAAIACgAAAAFASgKAAAAD 1RFU1RCT1gyV09SS0dST1VQ
c<---s	401 Unauthorized HTTP/1.1 WWW-Authenticate: NTLM TlRMTVNTUAACAAAAEgASADAAAAAHMgOiESIzRFVmd4gAAAAAAAAAA GwAbABCAAAAVwBPAFIASwBHAFIATwBVAFAAAgASAFcATwBSAEsARw BSAE8AVQBQAAEAEABUAEUAUwBUAEIATwBYADIABAAWAGUAeABhAG0 AcABsAGUALgBjAG8AbQADACQAUwBFAFIAVgBFAFIALgBlAHgAYQBt AHAAbABlAC4AYwBvAG0AAAAAAA==
c--->s	GET /vy6BSjy HTTP/1.1 Authorization: NTLM TlRMTVNTUAADAAAAGAAYAHAAAABKAEoAiAAAABAAEABIAAAACAAIA FgAAAAQABAAYAAAAAAAAADSAAAABQIAogUBKAoAAAAPVABFAFMAVA BCAE8AWAAyAHUAcwBlAHIAVABFAFMAVABCAE8AWAAyAC2JiLCSFSk lLByCToW06pkG1IgWSSLH86BiJh/Fdbatt+p+xqTDuUYBAQAAAAAA AMB78mW6Rs8BBtSIFkkix/MAAAAAAgASAFcATwBSAEsARwBSAE8AV QBQAAAAAAAAAAA
c<---s	200 OK HTTP/1.1

Attacker can run on hacked host following command:

```
iexplore http://evillocalsite
```

This will execute Internet Explorer browser, which will open in it's window specified website, and if the server requires authentication, if possible, the browser automatically authorize on it using the current user credentials.

Since the evillocalsite server is controlled by attacker it will forge authentication request to user's browser. Browser will authenticate, and attacker will get user's credentials.

But there are some nuances. First, attacker has to have own server. Secondly,

server must be accessible over the network from the attacked host. Thirdly, server has to be on corporate network and registered on the local DNS server that is not always possible. Fourth, most likely the system is configured to use a proxy server, and likely evillocalsite will not be available through a proxy server.

Attacker could reconfigure attacked system and add evillocalsite to exceptions, but it may be prohibited by group policy.

Yet another option is changing the proxy server in the browser to proxy controlled by attacker, for example, running on the user's host and opening in the browser a web site that requires authentication.

Anyway the user will notice Internet Explorer execution on his computer.

So many "ifs" and subtleties that could reveal and disrupt attack, that's why this method does not seem to be the best.

Security Support Provider Interface

Security Support Provider Interface (SSPI) allows an application to use various security models available on a computer or network without changing the interface to the security system. SSPI does not establish logon credentials because that is generally a privileged operation handled by the operating system.

A security support provider (SSP) is contained in a dynamic-link library (DLL) that implements SSPI by making one or more security packages available to applications. Each security package provides mappings between the SSPI function calls of an application and the functions of an actual security model. Security packages support security protocols such as Kerberos authentication and LAN Manager. The SSPI interface is available in kernel mode as well as user mode. [1]

Developers have many options for building distributed applications. Security Support Provider Interface provides an abstraction layer between application-level protocols and security protocols. Applications can take advantage of the SSPI security protocols in several ways:

- Call SSPI routines directly (for traditional, socket-based applications). The routines use request/response messages to implement the application protocol that carries SSPI security-related data.
- Use COM to call security options that are implemented by using authenticated RPC and SSPI at lower levels. These applications do not call SSPI functions directly.
- Use Windows Sockets 2 (WinSock) with the extended WinSock interface to allow transport providers to use security features. This approach integrates the security support provider into the network stack and provides both security and transport services through a common interface.
- Use the Windows Internet Extensions API (WinInet) and an interface designed to support Internet security protocols such as the Secure Sockets Layer (SSL) protocol. Applications use the SSPI interface to the

Secure Channel (Schannel) security provider to implement WinInet security. Schannel is the Microsoft implementation of SSL. [2]

Following are the SSP authentication packages provided by Microsoft:

- Credential Security Support Provider
- Microsoft Negotiate
- Microsoft NTLM
- Microsoft Kerberos
- Microsoft Digest SSP
- Secure Channel

Let's go a little deeper into some of them.

Microsoft Negotiate is a security support provider that acts as an application layer between Security Support Provider Interface and the other SSPs. When an application calls into SSPI to log on to a network, it can specify an SSP to process the request. If the application specifies Negotiate, Negotiate analyzes the request and picks the best SSP to handle the request based on customer-configured security policy.

Currently, the Negotiate security package selects between Kerberos and NTLM. Negotiate selects Kerberos unless it cannot be used by one of the systems involved in the authentication or the calling application did not provide sufficient information to use Kerberos. [3]

Microsoft NTLM or NTLMSSP (NT LAN Manager Security Support Provider) is a binary messaging protocol used by the Microsoft Security Support Provider Interface to facilitate NTLM challenge-response authentication and to negotiate integrity and confidentiality options.

NTLMSSP is used wherever SSPI authentication is used including, but not limited to, Server Message Block/CIFS extended security authentication, HTTP Negotiate authentication and MSRPC services. The NTLMSSP and NTLM challenge-response protocol have been fairly well documented in Microsoft's Open Protocol Specification. [4, 8]

NTLM credentials are based on data obtained during the interactive logon process and consist of a domain name, a user name, and a one-way hash of the user's password. NTLM uses an encrypted challenge/response protocol to authenticate a user without sending the user's password over the wire. Instead, the system requesting authentication must perform a calculation that proves it has access to the secured NTLM credentials.

Interactive NTLM authentication over a network typically involves two systems: a client system, where the user is requesting authentication, and a domain controller, where information related to the user's password is kept. Non interactive authentication, which may be required to permit an already logged on user to access a resource such as a server application, typically involves three systems: a client, a server, and a domain controller that does the authentication

calculations on behalf of the server.

The following steps present an outline of NTLM non interactive authentication. The first step provides the user's NTLM credentials and occurs only as part of the interactive authentication (logon) process.

1. (Interactive authentication only) A user accesses a client computer and provides a domain name, user name, and password. The client computes a cryptographic hash of the password and discards the actual password.
2. The client sends the user name to the server (in plaintext).
3. The server generates a 16-byte random number, called a challenge or nonce, and sends it to the client.
4. The client encrypts this challenge with the hash of the user's password and returns the result to the server. This is called the response.
5. The server sends the following three items to the domain controller:
   - User name
   - Challenge sent to the client
   - Response received from the client
6. The domain controller uses the user name to retrieve the hash of the user's password from the Security Account Manager database. It uses this password hash to encrypt the challenge.
7. The domain controller compares the encrypted challenge it computed (in step 6) to the response computed by the client (in step 4). If they are identical, authentication is successful.[5]

Microsoft Digest is a security support provider that implements the Digest Access protocol, a lightweight authentication protocol for parties involved in Hypertext Transfer Protocol (HTTP) or Simple Authentication Security Layer (SASL) based communications.

Microsoft Digest provides a simple challenge response mechanism for authenticating clients. This SSP is intended for use by client/server applications using HTTP or SASL based communications [6].

Vulnerability

Microsoft has developed so many different interfaces for user authentication that we have to exploit them somehow.

We created application that uses SSPI to authenticate current user not on the other system but on user’s host.

SSPI allows to directly call it’s routines to obtain request/response messages. It is assumed that the application itself should somehow transmit messages further, e.g., over a network. But our application will not pass them over network. All messages will stay in the application memory.

Depending on which SSP selected transmitted messages will be different, so if we choose NTLM application will transfer the following messages:

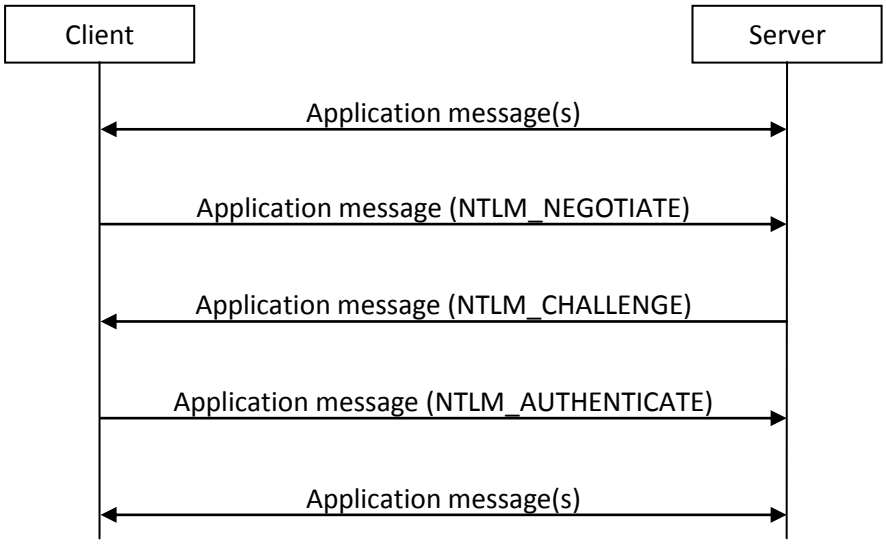

Figure 1 Authentication with NTLM

1. NTLM_NEGOTIATE. The client sends a Type 1 message to the server. This primarily contains a list of features supported by the client and requested of the server.
2. NTLM_CHALLENGE. The server responds with a Type 2 message. This contains a list of features supported and agreed upon by the server. Most importantly, however, it contains a challenge generated by the server.
3. NTLM_AUTHENTICATE. The client replies to the challenge with a Type 3 message. This contains several pieces of information about the client, including the domain and username of the client user. It also contains one or more responses to the Type 2 challenge.[7]

Since all exchange performed in memory there is no need to transfer any data over network and intercept it later. It means that we do not need any high level privileges. SSPI will prepare for us Type1, 2 and 3 messages which will contain all necessary authentication data. It only remains to get authentication data out.

Now data flow should be the following:

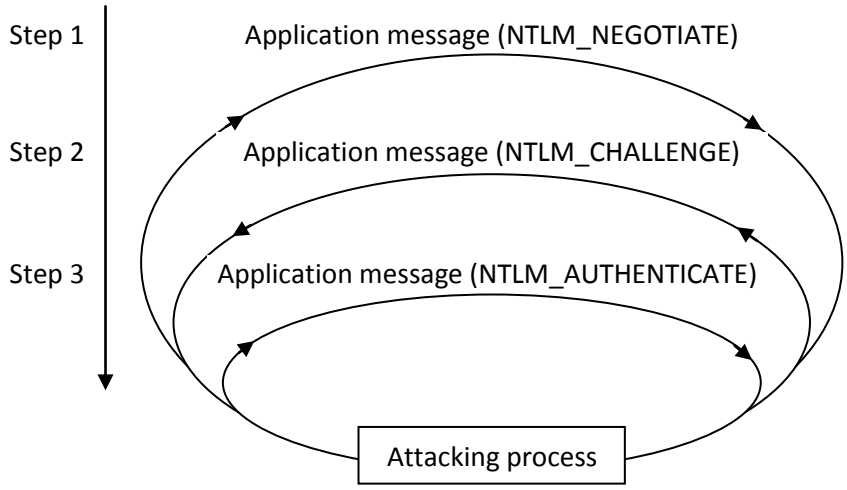

Figure 2 NTLMSSP data exchange

For ease of implementation we do not modify messages. To improve attack we could generate them and force SSPI to use less secure protocol NTLMv1 instead of NTLMv2.

Sample of data exchange is show on figure below.

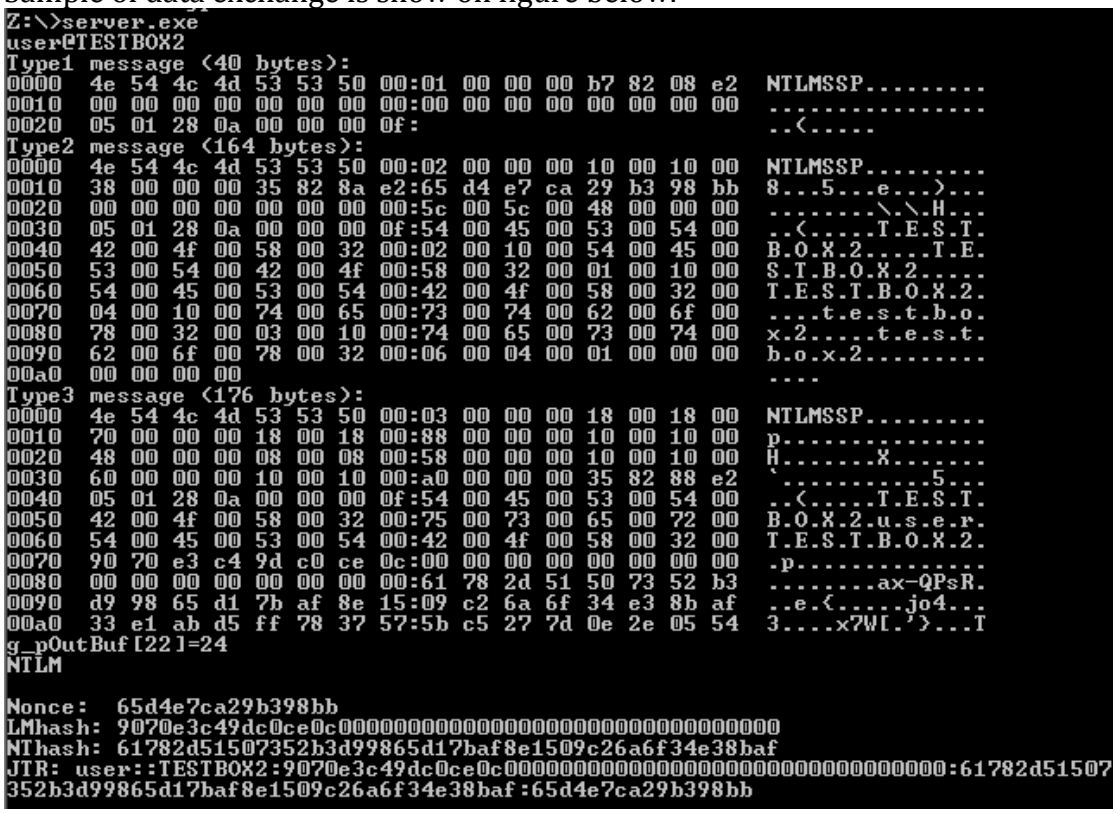

Figure 3 NTLMSSP data exchange

When challenge\response received we can run brute force attack on them and get user's password.

Due to wide spread of pass-the-hash attacks there is a trend to partially or totally disabling of NTLM protocols family and the transition to Kerberos, but in this case we still have a protocol Digest defined in RFC 2617 and RFC 2069.

For example, here is the recovery speed for different algorithms on average PC:

```
Benchmarking: HTTP Digest access authentication [HDAA-MD5]...
DONE
Many salts:      1064K c/s real, 1065K c/s virtual
Only one salt:   1042K c/s real, 1048K c/s virtual

Benchmarking: NTLMv1 C/R MD4 DES [ESS MD5] [netntlm]... DONE
Many salts:      2112K c/s real, 2130K c/s virtual
Only one salt:   1413K c/s real, 1413K c/s virtual

Benchmarking: NTLMv2 C/R MD4 HMAC-MD5 [netntlmv2]... DONE
Many salts:      520906 c/s real, 515779 c/s virtual
Only one salt:   423631 c/s real, 424661 c/s virtual
```

As you may notice, Digest is twice as fast as NTLMv2 that means that we can use

it instead of NTLMv2 to perform more effective brute force attacks.

Thus, without any privileges in the system, we were able to get a local user's password. Then we can use the recovered password to connect to enterprise services from the Internet, such as webmail, vpn, citrix and so on. The whole process shown in figure below:

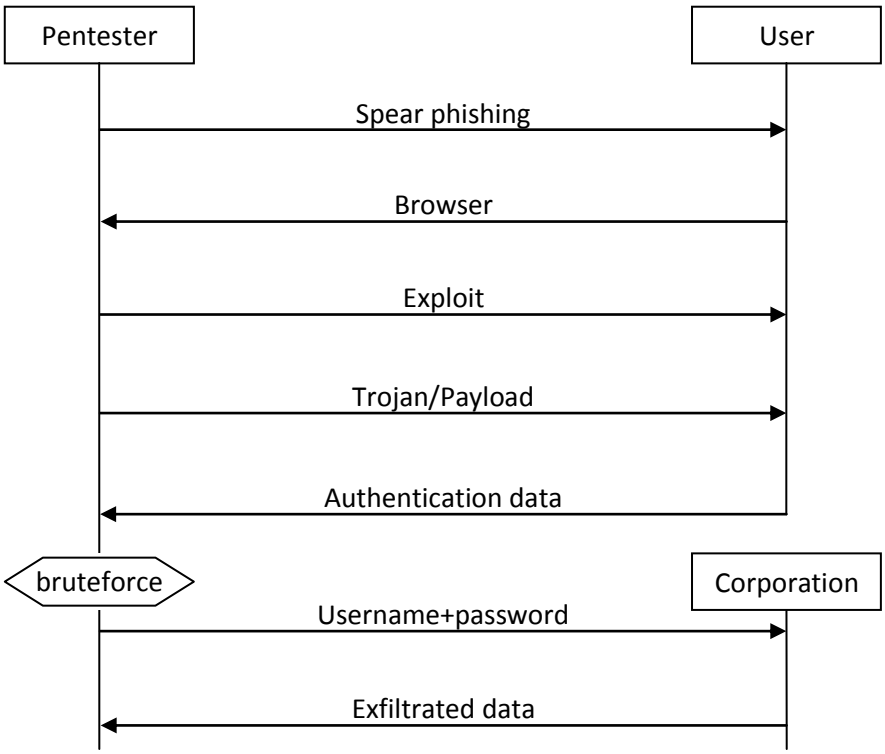

Figure 4 Exfiltration flow

Mitigation

The most effective method to counter this attack is to use two-factor authentication and strong enough passwords.

Release

Proof of concept tool mentioned here is published [9].

References

1. SSPI Model // http://msdn.microsoft.com/en-us/library/aa380497.aspx
2. SSPI Options for Distributed Applications // http://msdn.microsoft.com/en-us/library/aa380498%28v=vs.85%29.aspx
3. Microsoft Negotiate // http://msdn.microsoft.com/en-us/library/aa378748%28v=vs.85%29.aspx
4. NT LAN Manager Security Support Provider // http://en.wikipedia.org/wiki/NTLMSSP
5. Microsoft NTLM // http://msdn.microsoft.com/en-us/library/aa378749%28v=vs.85%29.aspx
6. Microsoft Digest SSP // http://msdn.microsoft.com/en-us/library/aa378745%28v=vs.85%29.aspx
7. The NTLM Authentication Protocol and Security Support Provider // http://davenport.sourceforge.net/ntlm.html
8. NT LAN Manager (NTLM) Authentication Protocol Microsoft Open Protocol Specification // http://msdn.microsoft.com/en-us/library/cc236621.aspx
9. https://github.com/snowytoxa/selfhash