

LM-Hash与NTLM-Hash

在windows下通过SAMInside提取到的密码Hash时，可以看到有两条，分别是LM-Hash和NTLM-HASH
这是对同一个密码的两种不同的加密方式，下面对其生成原理做个实验。

Windows下LM-Hash生成原理（IBM设计的LM Hash算法）

实验环境:windows server 2003

使用工具:SAMInside

LM HASH生成规则如下：

- 用户的密码被限制为最多14个字符。
- 用户的密码转换为大写。
- 密码转换为16进制字符串，不足14字节将会用0来再后面补全。
- 密码的16进制字符串被分成两个7byte部分。每部分转换成比特流，并且长度位56bit，长度不足使用0在左边补齐长度，再分7bit为一组末尾加0，组成新的编码（str_to_key()函数处理）。
- 上步骤得到的8byte二组，分别作为DES key为"KGS!@#%"进行加密。
- 将二组DES加密后的编码拼接，得到最终LM HASH值。

测试服务器密码为**123456**

- 用户的密码被限制为最多14个字符
- 用户的密码转换为大写，大写转换后仍为它本身
- 转换为16进制字符串后，结果为313233343536，不足14字节采用0进行补全，补全结果为3132333435360000000000000000

固定长度的密码被分成两个7byte部分，也就是分为31323334353600和0000000000000000，
先把31323334353600转换为比特流，比特流为110001001100100011001100110100001101010011011000000000，长度不足56bit使用0在左边补齐长度，补齐后再分7bit为一组末尾加0，组成新的编码，如下：

```
0011000 0
1001100 0
1000110 0
0110011 0
0100001 0
1010100 0
1101100 0
0000000 0
```

- 此时的密码字符串为0011000010011000100011000110011001000010101010001101100000000000
对应的8字节16进制编码（str_to_key()函数处理）：30988C6692C8D000，同理知0000000000000000对应的8字节16进制编码：0000000000000000

将以上步骤得到的两组16进制字符串，分别作为DES加密key为魔术字符串KGS!@#%\$进行加密

DES 计算器

算法选择

☒ 单 DES ☐ 三重 DES

明文 (HEX) 4B47532140232425

密钥 (HEX) 30988C6642A8D800

密文 (HEX) 44EFCE164AB921CA

加密运算 解密运算 退出

北京握奇智能科技有限公司 www.bwsmart.com

DES 计算器

算法选择

☒ 单 DES ☐ 三重 DES

明文 (HEX) 4B47532140232425

密钥 (HEX) 0000000000000000

密文 (HEX) AAD3B435B51404EE

加密运算 解密运算 退出

北京握奇智能科技有限公司 www.bwsmart.com

将两组DES加密后的编码拼接得到LM-HASH,计算结果与SAMinside提取结果相同

44EFCE164AB921CAAAD3B435B51404EE

[illegible]

python实现LM-HASH脚本

```
# coding=utf-8
import base64
import binascii
from pyDes import *

def DesEncrypt(str, Des_Key):
    k = des(Des_Key, ECB, pad=None)
    EncryptStr = k.encrypt(str)
    return binascii.b2a_hex(EncryptStr)

def Zero_padding(str):
    b = []
    l = len(str)
    num = 0
    for n in range(l):
        if (num < 8) and n % 7 == 0:
            b.append(str[n:n + 7] + '0')
            num = num + 1
    return ''.join(b)

if __name__ == "__main__":

    test_str = "123456"
    # 16个字节
    test_str = test_str.upper().encode('hex')
    str_len = len(test_str)

    # 14个字节补0
    if str_len < 28:
        test_str = test_str.ljust(28, '0')

    # 7byte
    t_1 = test_str[0:len(test_str) / 2]
    t_2 = test_str[len(test_str) / 2:]

    # 56bit补0
    t_1 = bin(int(t_1, 16)).lstrip('0b').rjust(56, '0')
    t_2 = bin(int(t_2, 16)).lstrip('0b').rjust(56, '0')

    # 7bit补0
    t_1 = Zero_padding(t_1)
    t_2 = Zero_padding(t_2)
    print t_1
    t_1 = hex(int(t_1, 2))
    t_2 = hex(int(t_2, 2))
    t_1 = t_1[2:].rstrip('L')
    t_2 = t_2[2:].rstrip('L')

    if '0' == t_2:
        t_2 = "0000000000000000"
    t_1 = binascii.a2b_hex(t_1)
    t_2 = binascii.a2b_hex(t_2)

    # 8byte DES key "KGS!@#$$%"
    LM_1 = DesEncrypt("KGS!@#$$%", t_1)
    LM_2 = DesEncrypt("KGS!@#$$%", t_2)

    # DES LM HASH
    LM = LM_1 + LM_2
    print LM
```

挑战/响应模式（鉴权协议）

鉴权协议如下的鉴权协议又被称作挑战--认证模式，使用明文口令模式时，网络上传输的就是明文口令本身，这很容易被Sniffer捕获。挑战/响应模式在传输信道是可被侦听Sniffer，但不可被篡改的情况下，这是一种简单而安全的方法。

LAN Manager Challenge/Response

LAN Manager Challenge/Response 验证机制，简称LM。该方案比NTLM响应时间更早，安全性更低。

SMB通信，Client A访问Server B通过LM身份验证的过程

首先我们假设Server B的密码为 "WELCOME", Server B已经缓存了密码的LM-HASH（原始密码在任何情况下都不能被缓存）我们通过上面的脚本计算"WELCOME"的LM-HASH为 "c23413a8a1e7665faad3b435b51404ee"

Server B -- 8bytes Challenge --> Client A
Server B向Client A发送了一个8字节挑战"0001020304050607"

Client A会根据自己的访问Server B的密码明文计算并缓存密码的LM-HASH（Client A缓存输入密码的哈希值，原始密码会被丢弃，“原始密码在任何情况下都不能被缓存”，这是一条基本的安全准则）然后在LM-HASH后5个0x00变成 "c23413a8a1e7665faad3b435b51404ee0000000000"，变为21字节，然后划分成三组，每组7字节

- ```
| C23413A8A1E766 | 5FAAD3B435B514 | 04EE000000000000 |
```
- 每组7字节做为参数传递给str\_to\_key()函数，最终得到三组DESKEY，每组8字节
- ```
| C21A04748A0E9CCC | 5ED4B47642ACD428 | 0476800000000000 |
```
- 分别用三组DESKEY对8字节挑战 "0001020304050607" 进行标准DES加密后得到

```
C21A04748A0E9CCC ---- ■0001020304050607■■■■■DES■■■ --> CA1200723C41D577
■■■■■
5ED4B47642ACD428 ---- ■0001020304050607■■■■■DES■■■ --> AB18C764C6DEF34F
■■■■■
0476800000000000 ---- ■0001020304050607■■■■■DES■■■ --> A61BFA0671EA5FC8
■■■■■
```

Client A最终获得一个24字节响应"CA1200723C41D577AB18C764C6DEF34FA61BFA0671EA5FC8"（这个结果被称为response）

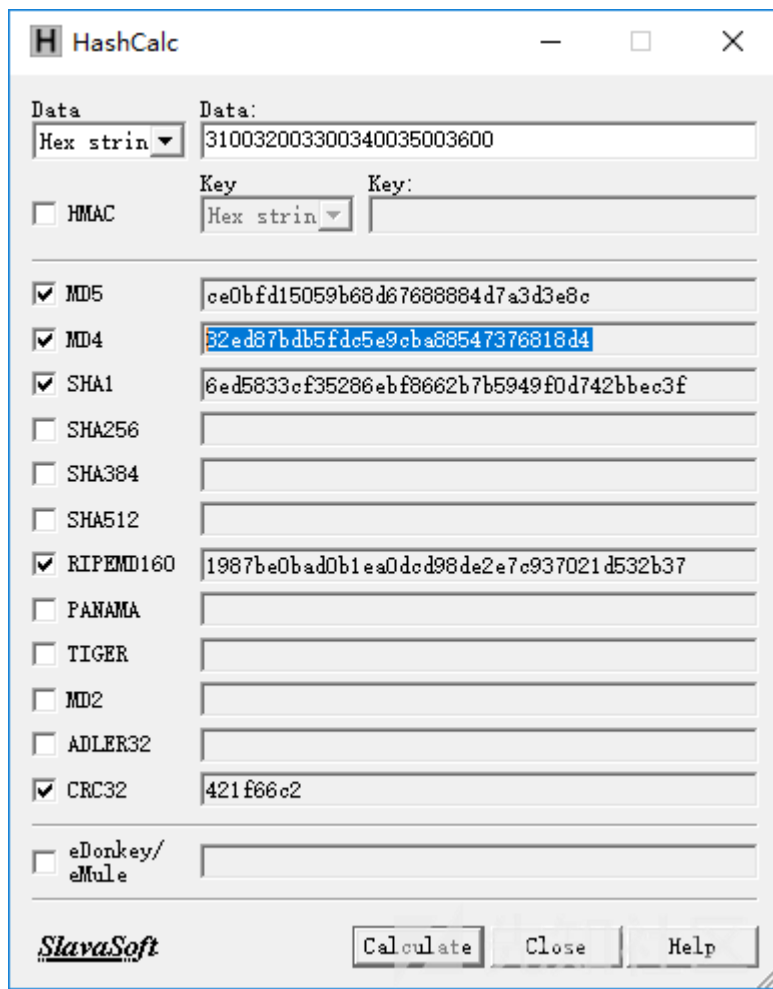
- Client A 将"CA1200723C41D577AB18C764C6DEF34FA61BFA0671EA5FC8" 送往Server B，Server B会根据自己缓存的LM-HASH进行同样的计算，并将计算结果与来自A的响应进行比较，如果匹配则身份验证通过

NTLM-HASH

IBM设计的LM Hash算法存在几个弱点，微软在保持向后兼容性的同时提出了自己的挑战响应机制，NTLM Hash便应运而生。

NTLM-HASH计算如下
密码为123456，首先将密码字符串转化为ASCII字符串，ASCII字符串再转换为十六进制字符串，十六进制字符串再转化为Unicode字符串，然后对Unicode字符串使用MD5
NTLM-HASH

- 转化为ASCII字符串 123456 ----> 49 50 51 52 53 54
- 转换为十六进制字符串 49 50 51 52 53 54 ----> 31 32 33 34 35 36
- 转化为Unicode字符串 31 32 33 34 35 36 ----> 310032003300340035003600
- 使用MD4消息摘要算法 31 32 33 34 35 36 ----> 32ed87bdb5fdc5e9cba88547376818d4



计算结果与SAMinside提取结果相同

32ed87bdb5fdc5e9cba88547376818d4

SAMInside					
File Edit View Tools Audit Service ?					
User	RID	LM-Password	NT-Password	LM-Hash	NT-Hash
<input type="checkbox"/> Administrator	500	123456	123456	44EFCE164AB921CAAAD3B435B51404EE	32ED87BDB5FDC5E9CBA88547376818D4
<input type="checkbox"/> Guest	501	<Empty>	<Empty>	AAD3B435B51404EEAAD3B435B51404EE	31D6CFE0D16AE931B73C59D7E0C089C0
<input checked="" type="checkbox"/> SUPPORT_388945a0	1001	<Empty>	??????????????	AAD3B435B51404EEAAD3B435B51404EE	D126A7392128F4FE255375ACD75569DC

各类HASH

	LM	NTLMv1	NTLMv2
Password case sensitive	No	Yes	Yes
Hash key length	56bit + 56bit	-	-
Password hash algorithm	DES (ECB mode)	MD4	MD4
Hash value length	64bit + 64bit	128bit	128bit
C/R key length	56bit + 56bit + 16bit	56bit + 56bit + 16bit	128bit
C/R algorithm	DES (ECB mode)	DES (ECB mode)	HMAC_MD5
C/R value length	64bit + 64bit + 64bit	64bit + 64bit + 64bit	128bit

NTLM消息介绍

NTLM验证是一种Challenge/Response 验证机制，由三种消息组成：通常称为类型1（协商），类型2（质询）和类型3（身份验证）。

它基本上是这样工作的：

- 客户端向服务器发送类型1消息。这主要包含客户端支持的功能和服务器请求的功能列表。
- 服务器用类型2消息进行响应。这包含服务器支持并同意的功能列表。然而，最重要的是，它包含了服务器产生的挑战。
- 客户用类型3消息回复质询。这包含有关客户端的几条信息，包括客户端用户的域和用户名。它还包含对类型2挑战的一个或多个响应。（类型3消息中的响应是最关键的）

协商消息示例（此处为NTLMv2）

协商消息从客户端发送到服务器以启动NTLM身份验证。其主要目的是通过FLAG指明支持的选项来建立认证的“基本规则”。

十六进制协商消息：

4e544c4d53535000010000000732000006000600330000000b000b0028000000050093080000000f574f524b53544154494f4e444f4d41494e

将其分解如下：

0	0x4e544c4d53535000	NTLMSSP签名
8	为0x01000000	类型1指标
12	0x07320000	标志: 协商Unicode (0x00000001) 协商OEM (0x00000002) 请求目标 (0x00000004) 协商NTLM (0x00000200) 协商域提供 (0x00001000) 协商工作站提供 (0x00002000)
16	0x0600060033000000	提供的域安全缓冲区: 长度: 6个字节 (0x0600) 分配空间: 6个字节 (0x0600) 偏移量: 51个字节 (0x33000000)
24	0x0b000b0028000000	提供的工作站安全缓冲区: 长度: 11个字节 (0x0b00) 分配空间: 11个字节 (0x0b00) 偏移量: 40个字节 (0x28000000)
32	0x050093080000000f	OS版本结构: 主要版本: 5 (0x05) 次要版本: 0 (0x00) 内部版本号: 2195 (0x9308) 未知/保留 (0x0000000f)
40	0x574f524b53544154494f4e	提供的工作站数据 (" WORKSTATION ")
51	0x444f4d41494e	提供的域数据 (" DOMAIN ")

wireshark 抓包分析协商消息数据包：

1	0.000000	192.168.10.1	192.168.10.102	TCP	66 4155 → 445 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS...
2	0.001212	192.168.10.102	192.168.10.1	TCP	66 445 → 4155 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 ...
3	0.001306	192.168.10.1	192.168.10.102	TCP	54 4155 → 445 [ACK] Seq=1 Ack=1 Win=66048 Len=0
4	0.001398	192.168.10.1	192.168.10.102	SMB	213 Negotiate Protocol Request
5	0.022350	192.168.10.102	192.168.10.1	SMB2	506 Negotiate Protocol Response
6	0.022507	192.168.10.1	192.168.10.102	SMB2	232 Negotiate Protocol Request
7	0.023562	192.168.10.102	192.168.10.1	SMB2	566 Negotiate Protocol Response
8	0.024275	192.168.10.1	192.168.10.102	SMB2	220 Session Setup Request, NTLMSSP_NEGOTIATE
9	0.024831	192.168.10.102	192.168.10.1	SMB2	401 Session Setup Response, Error: STATUS_MORE_PROCESS...
10	0.025226	192.168.10.1	192.168.10.102	SMB2	715 Session Setup Request, NTLMSSP_AUTH, User: DESKTOP...
11	0.032024	192.168.10.102	192.168.10.1	SMB2	159 Session Setup Response

> Frame 8: 220 bytes on wire (1760 bits), 220 bytes captured (1760 bits)

> Ethernet II, Src: Vmware_c0:00:08 (00:50:56:c0:00:08), Dst: Vmware_c2:ec:a7 (00:0c:29:c2:ec:a7)

> Internet Protocol Version 4, Src: 192.168.10.1, Dst: 192.168.10.102

> Transmission Control Protocol, Src Port: 4155, Dst Port: 445, Seq: 338, Ack: 965, Len: 166

> NetBIOS Session Service

> SMB2 (Server Message Block Protocol version 2)

> SMB2 Header

> Session Setup Request (0x01)

> StructureSize: 0x0019

> Flags: 0

> Security mode: 0x01, Signing enabled

> Capabilities: 0x00000001, DFS

Channel: None (0x00000000)

Previous Session Id: 0x0000000000000000

> Security Blob: 604806062b0601050502a03e303ca00e300c060a2b060104...

Offset: 0x00000058

Length: 74

> GSS-API Generic Security Service Application Program Interface

OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)

> Simple Protected Negotiation

> negTokenInit

> mechTypes: 1 item

mechToken: 4e544c4d53535000010000000978208e2000000000000000...

> NTLM Secure Service Provider

NTLMSSP identifier: NTLMSSP

NTLM Message Type: NTLMSSP_NEGOTIATE (0x00000001)

> Negotiate Flags: 0xe2088297, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version

Calling workstation domain: NULL

Calling workstation name: NULL

> Version 10.0 (Build 16299); NTLM Current Revision 15

质询消息示例（此处为NTLMv2）

质询消息由服务器发送到客户端以响应客户端的协商消息。它用于完成与客户的选择的谈判，并且向客户提供挑战。它可以选择包含有关认证目标的信息。

十六进制质询消息：

```
4e544c4d53535000020000000c000c003000000001028100
0123456789abcdef00000000000000000620062003c000000
44004f004d00410049004e0002000c0044004f004d004100
49004e0001000c0053004500520056004500520004001400
64006f006d00610069006e002e0063006f006d0003002200
7300650072007600650072002e0064006f006d0061006900
6e002e0063006f006d000000000000
```

将其分解为其组成字段给出：

0	0x4e544c4d53535000	NTLMSSP签名											
8	0x02000000	类型2指标											
12	0x0c000c0030000000	目标名称安全缓冲区: 长度: 12字节 (0x0c00) 分配空间: 12字节 (0x0c00) 偏移量: 48字节 (0x30000000)											
20	0x01028100	标志: 协商Unicode (0x00000001) 协商NTLM (0x00000200) 目标类型域 (0x00010000) 协商目标信息 (0x00800000)											
24	0x0123456789abcdef	挑战											
32	0x0000000000000000	上下文											
40	0x620062003c000000	目标信息安全缓冲区: 长度: 98字节 (0x6200) 分配空间: 98字节 (0x6200) 偏移: 60字节 (0x3c000000)											
48	0x44004f004d00410049004e00	目标名称数据 (" DOMAIN ")											
60	0x02000c0044004f004d00410049004e0001000c005300450052005600450052000400140064006f006d00610069006e002e0063006f006d00030022007300650072007600650072002e0064006f006d00610069006e002e0063006f006d0000000000	目标信息数据: <table><tr><td>0x02000c0044004f004d00410049004e00</td><td>域名子块: 类型: 2 (域名, 0x0200) 长度: 12字节 (0x0c00) 数据: " DOMAIN "</td></tr><tr><td>0x01000c00530045005200560045005200</td><td>服务器名称子块: 类型: 1 (服务器名称, 0x0100) 长度: 12字节 (0x0c00) 数据: " 服务器 "</td></tr><tr><td>0x0400140064006f006d00610069006e002e0063006f006d00</td><td>DNS域名子块: 类型: 4 (DNS域名, 0x0400) 长度: 20字节 (0x1400) 数据: " domain.com "</td></tr><tr><td>0x030022007300650072007600650072002e0064006f006d00610069006e002e0063006f006d00</td><td>DNS服务器名称子块: 类型: 3 (DNS服务器名称, 0x0300) 长度: 34字节 (0x2200) 数据: " server.domain.com "</td></tr><tr><td>00000000</td><td>终结者子块: 类型: 0 (终止符, 0x0000) 长度: 0字节 (0x0000)</td></tr></table>		0x02000c0044004f004d00410049004e00	域名子块: 类型: 2 (域名, 0x0200) 长度: 12字节 (0x0c00) 数据: " DOMAIN "	0x01000c00530045005200560045005200	服务器名称子块: 类型: 1 (服务器名称, 0x0100) 长度: 12字节 (0x0c00) 数据: " 服务器 "	0x0400140064006f006d00610069006e002e0063006f006d00	DNS域名子块: 类型: 4 (DNS域名, 0x0400) 长度: 20字节 (0x1400) 数据: " domain.com "	0x030022007300650072007600650072002e0064006f006d00610069006e002e0063006f006d00	DNS服务器名称子块: 类型: 3 (DNS服务器名称, 0x0300) 长度: 34字节 (0x2200) 数据: " server.domain.com "	00000000	终结者子块: 类型: 0 (终止符, 0x0000) 长度: 0字节 (0x0000)
0x02000c0044004f004d00410049004e00	域名子块: 类型: 2 (域名, 0x0200) 长度: 12字节 (0x0c00) 数据: " DOMAIN "												
0x01000c00530045005200560045005200	服务器名称子块: 类型: 1 (服务器名称, 0x0100) 长度: 12字节 (0x0c00) 数据: " 服务器 "												
0x0400140064006f006d00610069006e002e0063006f006d00	DNS域名子块: 类型: 4 (DNS域名, 0x0400) 长度: 20字节 (0x1400) 数据: " domain.com "												
0x030022007300650072007600650072002e0064006f006d00610069006e002e0063006f006d00	DNS服务器名称子块: 类型: 3 (DNS服务器名称, 0x0300) 长度: 34字节 (0x2200) 数据: " server.domain.com "												
00000000	终结者子块: 类型: 0 (终止符, 0x0000) 长度: 0字节 (0x0000)												

No.	Time	Source	Destination	Protocol	Leng	Info
1	0.000000	192.168.10.1	192.168.10.102	TCP	66	4155 → 445 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS...
2	0.001212	192.168.10.102	192.168.10.1	TCP	66	445 → 4155 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 ...
3	0.001306	192.168.10.1	192.168.10.102	TCP	54	4155 → 445 [ACK] Seq=1 Ack=1 Win=66048 Len=0
4	0.001398	192.168.10.1	192.168.10.102	SMB	213	Negotiate Protocol Request
5	0.022350	192.168.10.102	192.168.10.1	SMB2	506	Negotiate Protocol Response
6	0.022507	192.168.10.1	192.168.10.102	SMB2	232	Negotiate Protocol Request
7	0.023562	192.168.10.102	192.168.10.1	SMB2	566	Negotiate Protocol Response
8	0.024275	192.168.10.1	192.168.10.102	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
9	0.024831	192.168.10.102	192.168.10.1	SMB2	401	Session Setup Response, Error: STATUS_MORE_PROCESS...
10	0.025226	192.168.10.1	192.168.10.102	SMB2	715	Session Setup Request, NTLMSSP_AUTH, User: DESKTOP...
11	0.032024	192.168.10.102	192.168.10.1	SMB2	159	Session Setup Response

> Frame 9: 401 bytes on wire (3208 bits), 401 bytes captured (3208 bits)

> Ethernet II, Src: Vmware_c2:ec:a7 (00:0c:29:c2:ec:a7), Dst: Vmware_c0:00:08 (00:50:56:c0:00:08)

> Internet Protocol Version 4, Src: 192.168.10.102, Dst: 192.168.10.1

> Transmission Control Protocol, Src Port: 445, Dst Port: 4155, Seq: 965, Ack: 504, Len: 347

> NetBIOS Session Service

▼ SMB2 (Server Message Block Protocol version 2)

> SMB2 Header

▼ Session Setup Response (0x01)

> StructureSize: 0x0009

> Session Flags: 0x0000

▼ Security Blob: a182010b30820107a0030a0101a10c060a2b060104018237...

Offset: 0x00000048

Length: 271

▼ GSS-API Generic Security Service Application Program Interface

▼ Simple Protected Negotiation

▼ negTokenTarg

negResult: accept-incomplete (1)

supportedMech: 1.3.6.1.4.1.311.2.2.10 (NTLMSSP - Microsoft NTLM Security Support Provider)

responseToken: 4e544c4d53535000020000001e001e003800000015828ae2...

▼ NTLM Secure Service Provider

NTLMSSP identifier: NTLMSSP

NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)

> Target Name: DESKTOP-MI7K39V

> Negotiate Flags: 0xe28a8215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version

NTLM Server Challenge: 4e783a49fe733dce

Reserved: 0000000000000000

> Target Info

> Version 10.0 (Build 16299); NTLM Current Revision 15

NTLM Secure Service Provider (ntlmssp), 238 字节

|| 分组: 16 · 已显示: 16 (100.0%) · 加载时间: 0:0.2 || 配置文件: Default

身份验证消息示例（此处为NTLMv2）

身份验证消息

是身份验证的最后一步。该消息包含客户端对上一步挑战的响应，这表明客户知道账户密码而不直接发送密码。

身份验证消息

还指示身份验证目标（域或服务器名称）和身份验证帐户的用户名以及客户端工作站名称。

身份验证消息结构

描述	内容
NTLMSSP签名	空终止的ASCII “NTLMSSP” (0x4e544c4d53535000)
NTLM消息类型	长 (0x03000000)
LM / LMv2响应	安全缓冲区
NTLM / NTLMv2响应	安全缓冲区
目标名称	安全缓冲区
用户名	安全缓冲区
工作站名称	安全缓冲区
会话密钥 (可选)	安全缓冲区



客户端创建一个或多个挑战的响应，有六种类型的回应：

- LM (LAN Manager) 响应 - 由大多数较早的客户端发送，这是“原始”响应类型。
- NTLM响应 - 这是由基于NT的客户端发送的，包括Windows 2000和XP。
- NTLMv2响应 - 在Windows NT Service Pack 4中引入的一种较新的响应类型。它替换启用了NTLM版本2的系统上的NTLM响应。
- LMv2响应 - 替代NTLM版本2系统上的LM响应。
- NTLM2会话响应 - 用于在没有NTLMv2身份验证的情况下协商NTLM2会话安全性时，此方案会更改LM和NTLM响应的语义。
- 匿名响应 - 当匿名上下文正在建立时使用; 没有提供实际的证书，也没有真正的身份验证。“存根”字段显示在类型3消息中。

wireshark 抓包分析身份验证消息数据包：

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.1	192.168.10.102	TCP	66	4155 → 445 [SYN] Seq=0 Win=64800 Len=0 MSS=1440 WS...
2	0.001212	192.168.10.102	192.168.10.1	TCP	66	445 → 4155 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 ...
3	0.001306	192.168.10.1	192.168.10.102	TCP	54	4155 → 445 [ACK] Seq=1 Ack=1 Win=66048 Len=0
4	0.001398	192.168.10.1	192.168.10.102	SMB	213	Negotiate Protocol Request
5	0.022350	192.168.10.102	192.168.10.1	SMB2	506	Negotiate Protocol Response
6	0.022507	192.168.10.1	192.168.10.102	SMB2	232	Negotiate Protocol Request
7	0.023562	192.168.10.102	192.168.10.1	SMB2	566	Negotiate Protocol Response
8	0.024275	192.168.10.1	192.168.10.102	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
9	0.024831	192.168.10.102	192.168.10.1	SMB2	401	Session Setup Response, Error: STATUS_MORE_PROCESS...
10	0.025226	192.168.10.1	192.168.10.102	SMB2	715	Session Setup Request, NTLMSSP_AUTH, User: DESKTOP...
11	0.032024	192.168.10.102	192.168.10.1	SMB2	159	Session Setup Response

```


- ▼ Security Blob: a182023530820231a0030a0101a2820214048202104e544c...
  - Offset: 0x00000058
  - Length: 569
- ▼ GSS-API Generic Security Service Application Program Interface
  - ▼ Simple Protected Negotiation
    - ▼ negTokenTarg
      - negResult: accept-incomplete (1)
      - responseToken: 4e544c4d535350000300000018001800a200000046014601...
    - ▼ NTLM Secure Service Provider
      - NtLmSsp identifier: NtLmSsp
      - NtLm Message Type: NtLmSsp_AUTH (0x00000003)
        - > Lan Manager Response: 00000000000000000000000000000000000000000000000000000000000000000000
        - Lmv2 Client Challenge: 0000000000000000
      - ▼ NTLM Response: 03b4e9129fbe586e457d55412b39f3240101000000000000...
        - Length: 326
        - Maxlen: 326
        - Offset: 186
      - ▼ NTLMv2 Response: 03b4e9129fbe586e457d55412b39f3240101000000000000...
        - NTPProofStr: 03b4e9129fbe586e457d55412b39f324
        - Response Version: 1
        - Hi Response Version: 1
        - Z: 000000000000
        - Time: Mar 8, 2018 16:13:34.465728900 UTC
        - NTLMv2 Client Challenge: bd64ae0fc8b41228
        - Z: 00000000
        - > Attribute: NetBIOS domain name: DESKTOP-MI7K39V
        - > Attribute: NetBIOS computer name: DESKTOP-MI7K39V
        - > Attribute: DNS domain name: DESKTOP-MI7K39V
        - > Attribute: DNS computer name: DESKTOP-MI7K39V
        - > Attribute: Timestamp
        - > Attribute: Flags
        - > Attribute: Restrictions

```

NTLM Secure Service Provider (ntlmssp), 528 字节 | 分组: 16 · 已显示: 16 (100.0%) · 加载时间: 0:0.2 | 配置文件: Default

NTLM响应

NTLM响应由较新的客户端发送。该方案解决了LM响应中的一些缺陷；然而，它仍然被认为相当薄弱。此外，NTLM响应几乎总是与LM响应一起发送。该算法的弱点可以用来获取不区分大小写的密码，以及用于查找NTLM响应使用的区分大小写密码的试错法。

NTLM响应计算如下：

客户端计算密码字符串的NTLM哈希。
将16字节的NTLM散列填充为21个字节，
该值分成三个7字节。
这些值用于创建三个DES密钥（每个7字节的作为一个密钥）。
这每一个密钥用于对来自质询消息的挑战进行DES加密（产生三个8字节密文值）。
这三个密文值被连接在一起形成一个24字节的值。这是NTLM的回应。
请注意：只有散列值的计算与LM方案不同；响应的计算方式是相同的。

例子（一个使用密码“SecREt01”的用户，质询消息的挑战为“0x0123456789abcdef”）。

密码十六进制的Unicode字符串为“0x53006500630052004500740030003100”；计算该值的MD4散列值，“0xcd06ca7c7e10c99b1d33b7485a2ed808”，这是NTLM哈希。

使用0填充到21个字节，得到“0xcd06ca7c7e10c99b1d33b7485a2ed8080000000000”。

将21字节分割成三部分“ 0xcd06ca7c7e10c9 ”, “ 0x9b1d33b7485a2e ”和“ 0xd8080000000000 ”。

三个密钥中的每一个用于对来质询消息的挑战 (“ 0x0123456789abcdef ”) 进行DES加密。
这会产生结果“ 0x25a98c1c31e81847 ” (使用我们的第一个键) , “ 0x466b29b2df4680f3 ” (使用第二个键) 和“ 0x9958fb8c213a9cc6 ” (使用第三个键) 。

这三个密文值被连接在一起形成24字节的NTLM响应：0x25a98c1c31e81847466b29b2df4680f39958fb8c213a9cc6

NTLMv2响应

NTLM版本2 (“NTLMv2”) 被用来解决NTLM中存在的安全问题。当启用NTLMv2时，NTLM响应被替换为NTLMv2响应，并且LM响应被替换为LMv2响应。

NTLMv2响应计算如下：
计算获得NTLM密码哈希，方法和上文一样。
计算获得 NTLMv2哈希值，先将用户名转换为大写，然后和目标拼接在一起 (目标为 domain or server name 的值，且区分大小写) 组成字符串，然后计算这个字符串的Unicode十六进制字符串，使用上文16字节NTLM散列作为密钥，将HMAC-MD5消息认证码算法应用于Unicode构建被称为“blob”的数据块。“blob”的数据块简述：


字节偏移	描述	内容
0	Blob签名	0x01010000
4	保留的	长 (0x00000000)
8	时间戳	Little-endian，64位有符号值，表示自1601年1月1日以来的十分之一微秒数。
16	客户端随机数	8个字节
24	未知	4字节
28	目标信息	目标信息块 (来自类型2消息)。

使用16字节NTLMv2散列 (在上面步骤中计算) 作为密钥，将HMAC-MD5消息认证码算法应用于质询消息的挑战与blob连接字符串。这会产生一个16字节的HASH输出值。

使用上图NTLMv2数据包，计算NTLMv2响应示例：

domain	DESKTOP-DVIA6R3
用户名	testapp
密码	123456789
Challenge	4e783a49fe733dce
目标信息	0101000000000000891b7768f8b6d301bd64ae0fc8b412280000000002001e004400450053004b0054004f0050002d004d00490037004b0033003900560001001e004400450053004b0054004f0050002d004d00490037004b0033003900560004001e004400450053004b0054004f0050002d004d00490037004b0033003900560003001e004400450053004b0054004f0050002d004d00490037004b0033003900560007000800891b7768f8b6d3010600040002000000008003000300000000000000100000000200000cde827d049d6339086424b6e6880b54b98a279af64f927eb1db25b319c7d211e0a0010000000000000000000000000000900260063006900660073002f003100390032002e003100360038002e00310030002e003100300032000000000000000000000000

- 首先算出密码123456789的NTLM,得到C22B315C040AE6E0EFEE3518D830362B

 SAMInside

LM/NT-Hash Generator

Password:

123456789

LM-Hash:

0182BD0BD444BF867CD839BF040D93B

NT-Hash:

C22B315C040AE6E0EFEE3518D830362B

PWDUMP-format:

Test_123456789:1000:0182BD0BD444BF867CD839BF040D93B

Add

Cancel

计算获得 NTLMv2哈希值，先将用户名转换为大写

testapp --> TESTAPP

然后和domain拼接在一起（ domain or server name 的值，且区分大小写）组成字符串

TESTAPPDESKTOP-DVIA6R3

然后计算这个字符串的Unicode十六进制字符串

54004500530054004100500050004400450053004B0054004F0050002D004400560049004100360052003300

使用上文16字节NTLM散列作为密钥，将HMAC-MD5消息认证码算法应用于Unicode十六进制字符串，得到16字节的NTLMv2

a92765662d236c31c620d365c89540d1

连接质询消息的挑战与blob得到字符串

4e783a49fe733dce010100000000000891b7768f8b6d301bd64ae0fc8b41228000000002001e004400450053004b0054004f0050002d004d0049003700

使用NTLMv2散列（在上面步骤中计算）作为密钥，将HMAC-MD5消息认证码算法应用于此字符串，这会产生一个16字节的HASH输出值

03b4e9129fbe586e457d55412b39f324

该值与blob连接以形成NTLMv2响应

03b4e9129fbe586e457d55412b39f324010100000000000891b7768f8b6d301bd64ae0fc8b41228000000002001e004400450053004b0054004f005000

然后我们发现这个和我们抓包看到的NTLMv2响应是一样的:

▼ NTLM Secure Service Provider

NTLMSSP identifier: NTLMSSP

NTLM Message Type: NTLMSSP_AUTH (0x00000003)

> Lan Manager Response: 00

LMv2 Client Challenge: 0000000000000000

▼ NTLM Response: 03b4e9129fbe586e457d55412b39f3240101000000000000...

Length: 326

Maxlen: 326

Offset: 186

▼ NTLMv2 Response: 03b4e9129fbe586e457d55412b39f3240101000000000000...

NTProofStr: 03b4e9129fbe586e457d55412b39f324



Hashcat的NTLMv2密码字典暴力破解应该就是还原上述过程对比ntlmv2_response，命令如下:

hashcat64.exe -m 5600 testapp::DESKTOP-DVIA6R3:4e783a49fe733dce:03b4e9129fbe586e457d55412b39f324:010100000000000891b7768f8b6d

点击收藏 | 1 关注 | 1

[上一篇：利用winrm.vbs绕过应用程序...](#) [下一篇：CTF中常见的RSA相关问题总结](#)

1. 1 条回复



[exhades](#) 2018-07-15 11:49:36

感谢分享

0 回复Ta

[登录](#) 后跟帖

先知社区

[现在登录](#)

热门节点

[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)