Central Auditing of Windows NT Using Windows Script Host (WSH)

By Roger R. McLaren

Written for GIAC GCNT Certification Practical
3/30/2001

Introduction

This paper was written to complete requirements for GIAC Certification in Windows NT Security. It is not intended to teach programming, nor does it attempt to suggest an audit policy for any network. A proper audit policy can only be determined by careful consideration of the network and it's security requirements. It is merely meant to be an example of how to use Windows Script Host (WSH) to audit common security log events as well as provide a fully functional example that may be used as-is or modified to suit the needs of any audit policy.

Microsoft corrected a number of problems with auditing in Service Pack 4. For the purposes of this paper we will assume that Service Pack 4 or later has been applied to all computers being audited.

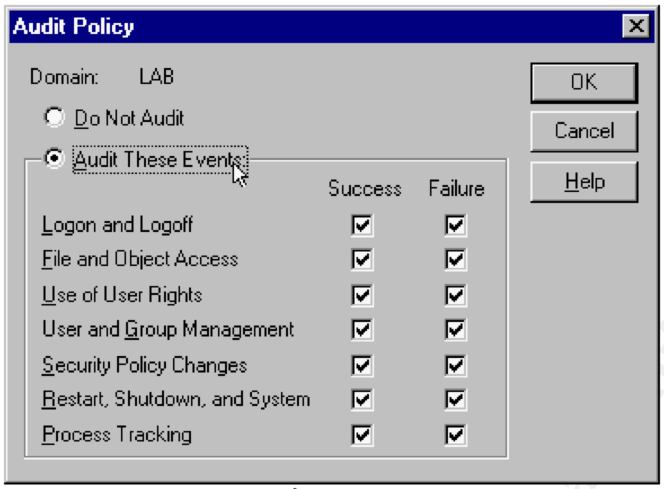
Deciding What to Audit

While most Systems Administrators will agree that network auditing is important, deciding what and how to audit is a difficult task that requires a great deal of thought, and should include every computer on the network.

Auditable events in Windows NT fall into the following 7 categories:

1. Logon and Logoff
2. File and Object Access
3. Use of User Rights
4. User and Group Management
5. Security Policy Changes
6. Restart, Shutdown and System
7. Process Tracking

Auditing successes or failures of each category can be enabled or disabled independently (See Fig. 1.).



Audit Policy

Domain: LAB

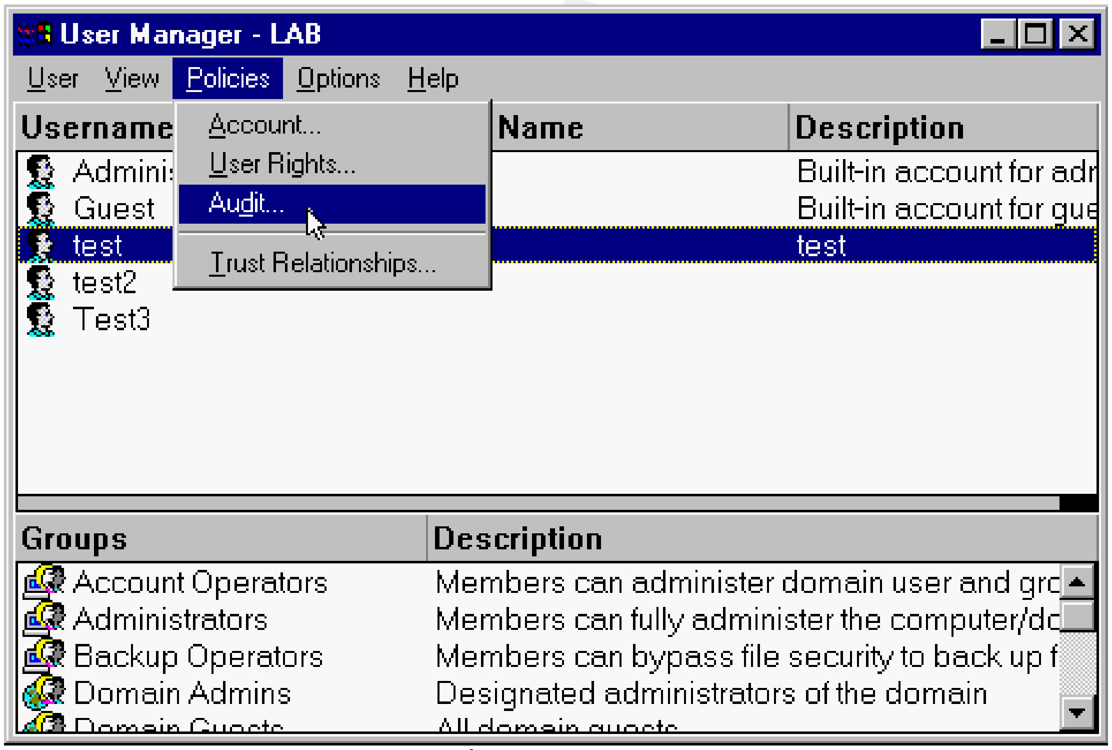
○ Do Not Audit

● Audit These Events:

	Success	Failure
Logon and Logoff	✓	✓
File and Object Access	✓	✓
Use of User Rights	✓	✓
User and Group Management	✓	✓
Security Policy Changes	✓	✓
Restart, Shutdown, and System	✓	✓
Process Tracking	✓	✓

Figure 1

Audit Settings for each computer are accessed through the User Manager (See Fig. 2)



Username	Name	Description
Adminis		Built-in account for adr
Guest		Built-in account for gue
test		test
test2		
Test3		

Groups	Description
Account Operators	Members can administer domain user and grc
Administrators	Members can fully administer the computer/dc
Backup Operators	Members can bypass file security to back up f
Domain Admins	Designated administrators of the domain
Domain Guests	All domain guests

Figure 2

NOTE: For a listing of all Windows NT Security log events and what information is recorded in each see Microsoft Knowledge Base article Q174074 – Security Event Descriptions.

Logon and Logoff

Enabling the 'Logon and Logoff' category enables auditing of the following events:

528 Successful logon
529 Logon Failure: Unknown user name or bad password
530 Logon Failure: Account logon time restriction violation
531 Logon Failure: Account currently disabled
532 Logon Failure: The specified user account has expired
533 Logon Failure: User not allowed to logon at this computer
534 Logon Failure: The user has not been granted the requested logon type at this computer
535 Logon Failure: The specified Account's password has expired
536 Logon Failure: The NetLogon component is not active
537 Logon Failure: An unexpected error occurred during logon
538 User Logoff
539 Logon Failure: Account locked out

IMPORTANT: These events are logged on the computer where the error occurred. Failed attempts to access file shares will be logged on the server where the files are stored. FAILED DOMAIN LOGONS ARE LOGGED ON THE WORKSTATION WHERE THE LOGON WAS ATTEMPTED. To successfully audit these events, all workstations must support logging, and auditing must be performed on all workstations. See Microsoft Knowledge Base article Q172402 – Auditing Logon Failures Does Not Log Remote Failures

644 Account locked out
Note: This event is generated on the Domain Controller when an account is locked out due to excessive bad logon attempts. See Microsoft Knowledge Base article Q182918 – Account Lockout Event Also Stored in Security Event Log on Domain Controller.

File and Object Access

Enabling the 'File and Object Access' category enables auditing of the following events:

560 Object Open
561 Handle Allocated
562 Handle Closed
563 Object Open for Delete
564 Object Deleted

NOTE: Enabling this category of auditing does not, by itself, generate any auditing. The files to be audited must reside on an NTFS partition, and the appropriate auditing will need to be set on each file to be monitored. This is done from the 'Security' tab of the file properties (see Fig. 3), and is done on a per-user/group basis (see fig. 4).

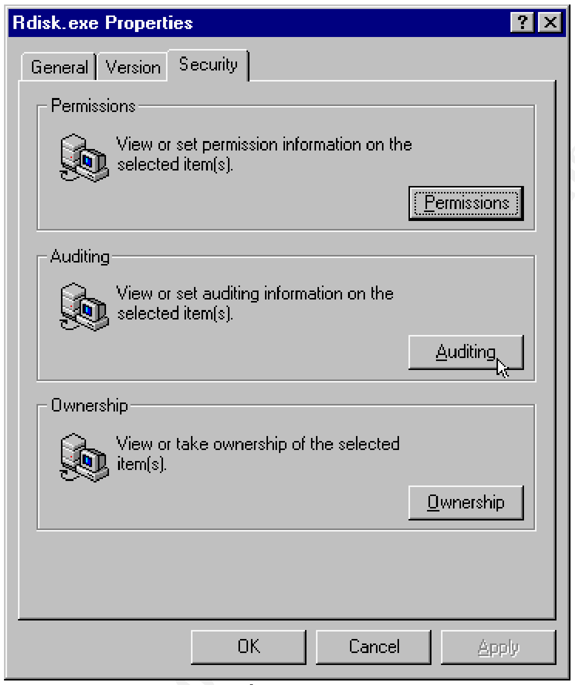
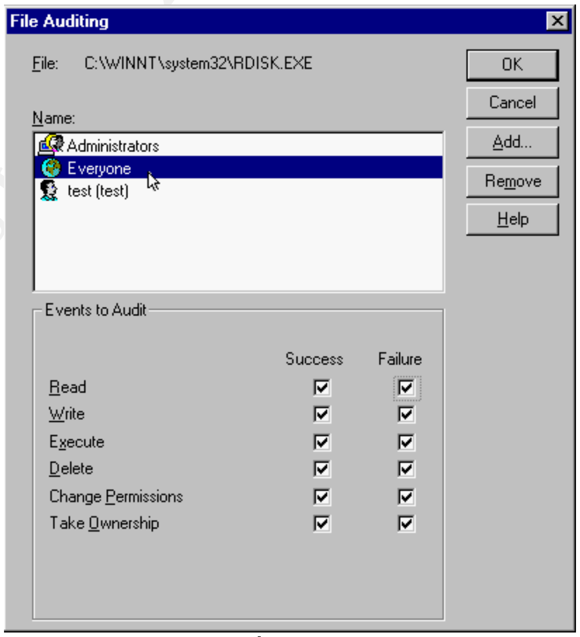

Rdisk.exe Properties
General
Version
Security
Permissions
View or set permission information on the selected item(s).
Permissions
Auditing
View or set auditing information on the selected item(s).
Auditing
Ownership
View or take ownership of the selected item(s).
Ownership
OK
Cancel
Apply

Figure 3



File: C:\WINNT\system32\RDISK.EXE

	Success	Failure
Read	✓	✓
Write	✓	✓
Execute	✓	✓
Delete	✓	✓
Change Permissions	✓	✓
Take Ownership	✓	✓

Figure4

Use of User Rights

Enabling the 'Use of User Rights' category enables auditing of the following events:

576 Special privileges assigned to new logon
577 Privileged Service Called
578 Privileged object operation

NOTE: Enabling this category of auditing will generate a very large number of audit log entries and could impact system performance. Enable auditing of these events only if absolutely necessary.

User and Group Management

Enabling the 'User and Group Management' category enables auditing of the following events:

624 User Account Created
625 User Account Type Change
626 User Account Enabled
627 Change Password Attempt
628 User Account password set
629 User Account Disabled
630 User Account Deleted
631 Security Enabled Global Group Created
632 Security Enabled Global Group Member Added
633 Security Enabled Global Group Member Removed
634 Security Enabled Global Group Deleted
635 Security Enabled Local Group Created
636 Security Enabled Local Group Member Added
637 Security Enabled Local Group Member Removed
638 Security Enabled Local Group Deleted
639 Security Enabled Local Group Changed
640 General Account Database Change
641 Security Enabled Global Group Changed
642 User Account Changed
643 Domain Policy Changed
644 User Account Locked Out

NOTE: Monitoring this category of events is crucial to any effective audit policy, as it will show when an account is created, enabled, or receives permissions by being added to a group.

If this category of auditing is enabled, erroneous event 637 messages stating that a user removed another user from a group may be logged. See Microsoft Knowledge Base article Q285148 - Event ID 637 Message Is Logged Stating that a User Removed Another User.

Security Policy Changes

Enabling the 'Security Policy Changes' category enables auditing of the following events:

608 User Right Assigned
609 User Right Removed
610 New Trusted Domain
611 Removing Trusted Domain
612 Audit Policy Change

NOTE: It is a good practice to always monitor this category of events as it will show when a user has gained elevated privileges through assignment of user rights (as opposed to being added to a group) as well as changes in auditing policy.

Restart, Shutdown, and System

Enabling the 'Restart, Shutdown, and System' category enables auditing of the following events:

512 Windows NT is starting up.
513 Windows NT is shutting down.
514 An authentication package has been loaded by the Local Security Authority.
515 A trusted logon process has registered with the Local Security Authority.
516 Internal resources allocated for the queuing of audit messages have been exhausted, leading to the loss of some audits.
517 The audit log was cleared.
518 A notification package has been loaded by the Security Account Manager.

NOTE: Any event in this category that occurs at an unexpected time should be investigated.

Process Tracking

Enabling the 'Process Tracking' category enables auditing of the following events:

592 A new process has been created
593 A process has exited

594 A handle to an object has been duplicated
595 Indirect access to an object has been obtained

NOTE: Enabling this category of auditing will generate a very large number of audit log entries and could impact system performance. Enable auditing of these events only if absolutely necessary.

Deciding How to Audit

There are many ways to audit a Windows NT network. Manually reviewing the logs, a variety of third party applications, or developing a custom application are all options.

Unless the network is very small, manual auditing of an entire network can take more time than most administrators have to devote to the task.

There are many excellent third party applications that are capable of auditing an entire enterprise network. However, the size of the network (or budget) may not warrant this type of solution.

For many small to medium sized networks a custom application using Microsoft Visual Basic or Windows Script Host (WSH) may be a perfect solution.

Using WSH To Audit Your Network

There are two distinct methods of using WSH to perform centralized auditing.

Subscription to the Event Log Service

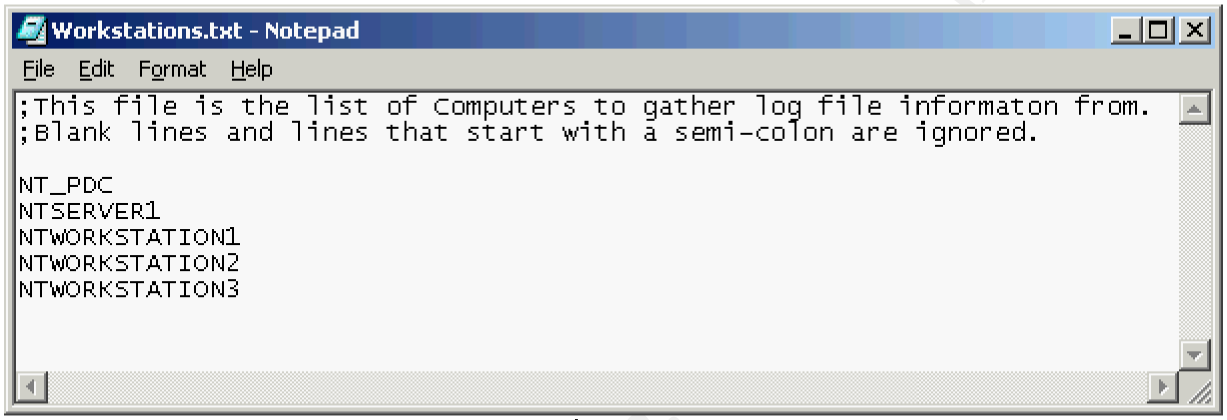
It is possible to subscribe to the Event Notification Service of a remote computer using a notification query (See example in Appendix B). This method requires Windows Management Instrumentation (WMI) to be installed on the remote computer and delivers new events to the auditing computer in real time. This method can be somewhat problematic however, as the subscription is broken if the remote computer is rebooted. The script could be written in such a way that it tries to reconnect, but attempting to audit many computers in this fashion may not be feasible.

Summarizing the Event Logs

It is also possible to use WSH to summarize the event logs of all the computers on the network and create a single report. A fully functional example of this kind of script is contained in Appendix A and will be discussed in this text.

Our Example Script

This script is written in Visual Basic Script (VBScript) as an example of how to poll the event logs of several computers and summarize them to create a single report. It receives input in the form of a file that contains a list of computer names (See Figure 5). It opens the file, reads the list of computers to audit, and then uses the Resource Kit utility Dumpel.exe to pull the data from each computer listed. It combines the data into a single temporary file consisting of one line for each event log entry. It then summarizes that file to create an output report.



```
;This file is the list of computers to gather log file informaton from.
;Blank lines and lines that start with a semi-colon are ignored.

NT_PDC
NTSERVER1
NTWORKSTATION1
NTWORKSTATION2
NTWORKSTATION3
```

Figure 5

NOTE: It is possible to query the event log of a remote computer directly using WSH and WMI (See Appendix C for an example). However, due to the use of the Common Object Model (COM) performance varies widely and is generally unacceptable, especially when the query returns more than approximately 1500 records.

The events that are summarized in this example are as follows:

512 System Startup – Lists the date, time, and workstation name for each event

517 Security Event Log Clear - Lists the date, time, and workstation name for each event

528 Successful Logon – Lists the date, time, domain\account name and workstation logged on to for each logon by an Administrator

529 Failed Logon due to Bad Username or Password – Lists the date, time, domain\account name and workstation name for each failed logon by an Administrator
Prints a summary list for all other accounts listing the domain\account name and number of failed logon attempts.

531 Attempt to Logon to a Disabled Account - Lists the date, time, domain\account name and workstation name for each event

532 Attempt to Logon to an Expired Account - Lists the date, time, domain\account name and workstation name for each event

534 User not Granted Requested Logon Type - Lists the date, time, domain\account name and workstation name for each event. NOTE: this is the event generated when an unauthorized user attempts a local logon to a server

612 Audit Policy Change – Lists the date, time, workstation where the audit policy changed, which domain\account changed the policy, and the newly effective policy.

624 Account Created – Lists the date, time, new domain\account name and the creator's domain\account

626 Account Enabled - Lists the date, time, new domain\account name and the user domain\account that enabled it.

632 Account Added to Global Group – Lists date, time, which group received a new account, and which user domain\account added it. NOTE: Only the user account's SID number is included in the event log, therefore, it is not included in the output report.

636 Account Added to Local Group – Lists date, time, which group received a new account, and which user domain\account added it. NOTE: Only the user account's SID number is included in the event log, therefore, it is not included in the output report.

644 Account Locked Out – Lists date, time, account name and workstation where the lockout was generated. NOTE: The locked out account's domain is not listed in the event log and is therefore not included in the output

A Brief Look at the Code

The following command forces all variables to be declared before they are used. A mistyped variable name within the script will flag an error. This can save a lot of time when debugging the script.

```
Option Explicit
```

This line of code calls a sub routine that outputs status. The status message to output is passed as a parameter.

```
OutputStatus "Program Started"
```

The OutputStatus routine is located at the end of the script and is used to output status messages to the administrator. WSH scripts can be executed by either of two script hosts. The CSCRIPT host is used from the command line and the WSCRIPT host is used when the script executes in a windows context (as when double-clicked from within Windows Explorer).

The Wscript.Echo method works well from the CSCRIPT host, but will cause the WSCRIPT host to pause and wait for the user to close the dialog box before the script continues. The Wscript.Popup method works well under the Wscript host as it creates a message box which will dismiss itself after a time (in this example 1 second), but this delay is unnecessary when running under the CSCRIPT host.

This routine uses the FullName property of the Wscript.Shell object to determine under which context the script is running. It then outputs the status in the manner best suited for that context.

```
'Output Status Routine
'This routine outputs status messages. This is used because the Wscript.Echo
'method causes the script to pause until the message box is dismissed when
'running under the Wscript host, therefore when running under Wscript we
'use the Wscript.Shell Popup method.
Sub OutputStatus(strStatus)

Dim objWshShell2
Dim strOutput

strOutput = time & " " & strStatus
Set objWshShell2 = WScript.CreateObject("WScript.Shell")
    If ucase(right(wscript.FullName,11))="CSCRIPT.EXE" Then
        wscript.echo strOutput
    End If
    If ucase(right(wscript.FullName,11))="WSCRIPT.EXE" Then
            return=objWshShell2.Popup (strOutput,1)
    End If
End Sub
```

All variables must be declared using the Dimension statement.

```
'declare variables for processing event 528 (Successful Logon)
Dim intEvent528Count
Dim strEvent528Date(10000)
Dim strEvent528Time(10000)
Dim strEvent528UserName(10000)
Dim strEvent528Workstation(10000)
```

NOTE: The array sizes were set to an arbitrary value (10000), this value should be sufficient for small networks. If a 'Subscript out of range error' is generated increase these values for that event number.
The text of a subscript error will look like the following:

```
D:\LOGFILE\auditlogs.vbs(254, 4) Microsoft VBScript runtime error: Subscript
out of range: '10001'
```

This line returns the path to the directory where the script file resides. This is used for locating the input and output files.

```
strScriptPath = left(wscript.ScriptFullName,
InStrRev(wscript.ScriptFullName,"\"))
```

The following objects need to be created. They can be re-used so only one of each type is necessary.

```
'create file system and shell objects
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objArguments = Wscript.Arguments
```

This section checks to see if two arguments were passed to the script. If not, the default values are used and the script checks to see if the input file exists. The script expects to receive either no arguments or both the input and output file as arguments. This was done to simplify the coding for this example.

```
'If 2 arguments are received they are assigned
'to the input and output file name, otherwise error.

If objArguments.count = 0 Then
    strInputFileName = strScriptPath & "workstations.txt"
    strOutputFileName = strScriptPath & "logout.txt"
Elseif objArguments.count = 2 Then
    strInputFileName = objArguments(0)
    strOutputFileName = objArguments(1)
Else
    wscript.echo "Useage: cscript AuditLogs.vbs WorkstationList OutputFile"
    wscript.echo "      Where: WorkstationList = Path and filename of list of
stations to audit"
    wscript.echo "             OutputFile = Path and filename of outputfile"
    wscript.quit
End If

'Check if Input File exists
If Not objFSO.FileExists(strInputFileName) Then
    wscript.echo "Input file doesn't exist!!!"
    wscript.echo
    wscript.echo "Useage: cscript AuditLogs.vbs WorkstationList OutputFile"
    wscript.echo "     Where: WorkstationList = Path and filename of list of 
stations to audit"
    wscript.echo "           OutputFile = Path and filename of outputfile"
    wscript.quit
End If
```

A temporary file for the output of the dumpel.exe command is created.

```
'Initialize temporary file to dump logs to
Set objFile = objFSO.CreateTextFile(strScriptPath & strTempFileName, True)
objFile.close
```

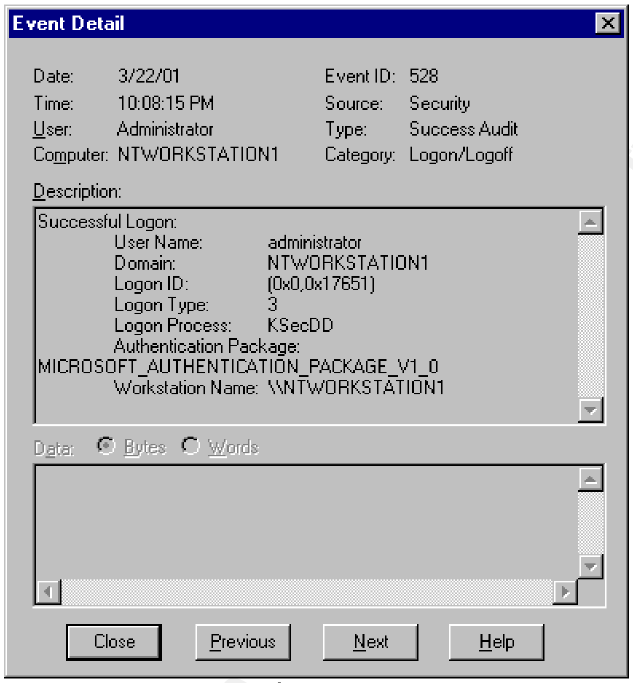
The next step is to open the input file, store the workstation names in an array and run the Resource Kit dumpel.exe utility to gather the security log information from each one and append it to the temporary file.

```
'Open input file and call the dumpel command for each entry
'Skip blank lines and lines that start with a ;
Set objTextStream1 = objFSO.OpenTextFile(strInputFileName)
Do while Not objTextStream1.AtEndOfStream
    strInputLine = objTextStream1.ReadLine
    If len(trim(strInputLine)) > 0 And left(strInputLine,1) <> ";" Then
        intNumberOfWorkstations = intNumberOfWorkstations + 1
        strWorkstationList(intNumberOfWorkstations) = _
        ucase(trim(strInputLine))

        'Create commandline to run the dumpel resource kit utility and
        'Pipe the output into our temp file.
        'If the dumpel.exe is not found, use a fully qualified path in this 
line
        '(for example c:\progra~1\ntreskit\dumpel.exe ...)
        'syntax is as follows:
        ' -s servername         (name of server to dump log from)
        ' -t                    (Use tabs for seperator)
        ' -l logfile            (name of log to dump - security, application, 
or system)
        strCommandLine = "cmd /c dumpel -s " & trim(strInputLine) & _
            " -t -l security >> " & strScriptPath & strTempFileName

        'run the dumpel utility.
            OutputStatus "Querying " & ucase(trim(strInputLine))
            return=objWshShell.Run (strCommandLine,0,True)
    End If
Loop
```

NOTE: The `0,True` in the run command causes the dumpel.exe command to run hidden, and the script to wait for it to complete before continuing.

The heart of this script is a loop that reads each line of the temporary file, splits that line into an array of data, and looks at the event number contained within the data. This text will look at the processing of a single event number (event 528 – Successful Logon).

Event 528 looks like this when viewed with the Event Viewer:



```
Date:     3/22/01                Event ID:  528
Time:     10:08:15 PM            Source:    Security
User:     Administrator          Type:      Success Audit
Computer: NTWORKSTATION1         Category:  Logon/Logoff

Description:
Successful Logon:
            User Name:          administrator
            Domain:             NTWORKSTATION1
            Logon ID:           (0x0,0x17651)
            Logon Type:         3
            Logon Process:      KSecDD
            Authentication Package:
MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
            Workstation Name:  \\NTWORKSTATION1
```

Figure 6

The SPLIT function is used to separate the input line into an array where each element contains one piece of data.

After the SPLIT function the data array consists of the following:

Array element 0 = 3/22/2001
Array element 1 = 10:08:15 PM
Array element 2 = 8
Array element 3 = 2
Array element 4 = 528
Array element 5 = Security
Array element 6 = NTWORKSTATION1\Administrator
Array element 7 =
Array element 8 = NTWORKSTATION1
Array element 9 = Successful Logon:
Array element 10 = User Name:
Array element 11 = Administrator
Array element 12 = Domain:
Array element 13 =

Array element 14 = NTWORKSTATION1
Array element 15 = Logon ID:
Array element 16 =
Array element 17 = (0x0,0x17620)
Array element 18 = Logon Type:
Array element 19 = 2
Array element 20 = Logon Process:
Array element 21 = User32
Array element 22 = Authentication Package:
Array element 23 = MICROSOFT_AUTHENTICATION_PACKAGE_V1_0
Array element 24 = Workstation Name:
Array element 25 = NTWORKSTATION1

It is important to note that array elements 0 through 8 always contain the same information. The remaining array elements contain the data strings recorded in the event.

Element 0 = the Date the event was logged
Element 1 = the time the event was logged
Element 2 = the event type (as a number)
Element 3 = the event category (as a number)
Element 4 = the event number
Element 5 = the source that logged the event
Element 6 = DOMAIN\Username of the account that generated the event
Element 7 = Blank
Element 8 = The Workstation where the event was logged

The loop consists of an If…Then statement for each event to process. When an event to be processed is encountered (in this case Event 528 – successful Logon) the data is processed accordingly. Since this script only processes logons by accounts named 'Administrator' there is a second If…Then statement to check that condition.

When an administrator logon is found, the variable that keeps track of how many event 528s have been received `(intEvent528Count)` is incremented by one. The appropriate data is then added to the next location in the data arrays. In this case the date, time, domain\username, and which workstation the administrator logged on to are recorded.

Note the use of the UCASE and TRIM functions. Using UCASE to convert the received account name to all uppercase avoids mismatching due to differences in case. The TRIM function is used to eliminate leading and trailing spaces in the account name and domain.

```
'open the text file we just created
SET objTextStream1 = objFSO.OpenTextFile(strScriptPath & strTempFileName)

'loop to read the text file and process each line
```

```
Do While Not objTextStream1.AtEndOfStream
    strInputLine = objTextStream1.ReadLine
    strSplitString = split(strInputLine, chr(9))

'***************************************************************************
'Process Event 528 - Successful Logon
    If strSplitString(4) = "528" Then
        If ucase(trim(strSplitString(11))) = "ADMINISTRATOR" Then
            intEvent528Count = intEvent528Count + 1
            strEvent528Date(intEvent528Count) = strSplitString(0)
            strEvent528Time(intEvent528Count) = strSplitString(1)
            strEvent528UserName(intEvent528Count) = _
                ucase(trim(strSplitString(14))) & _
                "\" & lcase(trim(strSplitString(11)))
            strEvent528Workstation(intEvent528Count) = _
                trim(strSplitString(25))
            If left(strEvent528Workstation(intEvent528Count),2) = "\\" Then
                strEvent528Workstation(intEvent528Count) = _
                right(strEvent528Workstation(intEvent528Count),_
                (len(strEvent528Workstation(intEvent528Count))-2))
            End If
        End If
    End If
'***************************************************************************

Loop
```

The output file is then created. A header line containing the current date and time and a list of all computers included in the audit is output. After this a series of For…Next loops adds the data for each event number.

```
'Output Routine
'Open Output File - Overwrite if it already exists
OutputStatus "Creating Output File"
Set objTextStream1 = objFSO.CreateTextFile(strOutputFileName, True)
'***************************************************************************

'***************************************************************************
'Output Header Line
    objTextStream1.WriteLine("                Audit of Security Logs " & date
& " " & time )
    objTextStream1.WriteLine
'***************************************************************************

'***************************************************************************
'Output the names of machines in the audit
    objTextStream1.WriteLine("Computers Audited:")
    For intLoopCounter = 1 To intNumberOfWorkstations step 2
    strOutputString = left(strWorkstationList(intLoopCounter) & _
        Space(50),50) & strWorkstationList(intLoopCounter +1)
    objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'***************************************************************************
```

```
'**************************************************************************
'Output Routine for Event 528 (Successful Logon) for Administrator Logons
    objTextStream1.WriteLine("Administrative Logons:")
    objTextStream1.WriteLine("   DATE        TIME              " & _
        "ACCOUNT                  LOGON TO:")
    For intLoopCounter = 1 To intEvent528Count
        strOutputString = right(space(10) & _
            strEvent528Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent528Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "  " & _
            strEvent528UserName(intLoopCounter) & _
            space(55),55) & strEvent528Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************
```

The output file is closed and the script exits.

```
objTextStream1.close
```

The resulting output file looks like this:

NOTE: In order for the report columns to line up correctly the report must be viewed and printed in a fixed-width font.

```
                Audit of Security Logs 3/290/01 10:38:05 PM

Computers Audited:
NTSERVER1                                    NTWORKSTATION1

Administrative Logons:
   DATE        TIME              ACCOUNT                  LOGON TO:
 3/17/2000   8:36:06 PM  LAB\administrator              NTWORKSTATION1
 3/18/2000   9:09:24 PM  LAB\administrator              NTSERVER1
 3/22/2001  10:08:15 PM  NTWORKSTATION1\administrator   NTWORKSTATION1

Administrative Logon Failures:
   DATE        TIME              ACCOUNT                  FAILED LOGON TO:
 3/18/2000   9:08:05 PM  LAB\administrator              NTSERVER1
 3/17/2001   8:31:55 PM  LAB\administrator              NTWORKSTATION1
 3/23/2001   3:33:44 PM  NTWORKSTATION1\administrator   NTWORKSTATION1
 3/23/2001   3:33:45 PM  NTWORKSTATION1\administrator   NTWORKSTATION1
 3/23/2001   3:33:47 PM  NTWORKSTATION1\administrator   NTWORKSTATION1
 3/23/2001   3:33:48 PM  NTWORKSTATION1\administrator   NTWORKSTATION1
 3/23/2001   3:33:49 PM  NTWORKSTATION1\administrator   NTWORKSTATION1
 3/23/2001   3:33:51 PM  NTWORKSTATION1\administrator   NTWORKSTATION1

General Logon Failures:
   ACCOUNT                  # OF FAILURES
LAB\rm                            15
LAB\ddsafsadfsadf                 3
```

```
LAB\test                        16
NTWORKSTATION1\rrrrr            1

Attempts to Logon to a Disabled Account:
   DATE        TIME            ACCOUNT                        FAILED LOGON TO:
 3/20/2001  11:05:03 PM  LAB\test                             NTWORKSTATION1

Attempts to Logon to an Expired Account:
   DATE        TIME            ACCOUNT                        FAILED LOGON TO:
 3/20/2001  11:54:17 PM  LAB\test                             NTWORKSTATION1

User Not Granted Requested Logon Type:
   DATE        TIME            ACCOUNT                        FAILED LOGON TO:
 3/18/2000   9:09:49 PM  LAB\test                             NTSERVER1

System Started:
   DATE        TIME            SYSTEM
 3/18/2000   4:58:19 PM  NTSERVER1
 3/23/2001   3:22:14 PM  NTWORKSTATION1

Security Audit Log Cleared:
   DATE        TIME            SYSTEM                         BY:
 3/17/2000   8:28:25 PM  NTSERVER1                            LAB\administrator

Audit Policy Changes:
   DATE        TIME            SYSTEM                         BY:
 3/19/2000  10:51:19 PM  NTSERVER1                            LAB\administrator
New Effective Policy:
     Logon And Logoff                SUCCESS/FAILURE
     File And Object Access          SUCCESS/FAILURE
     Use Of User Rights              SUCCESS/FAILURE
     User And Group Management       SUCCESS/FAILURE
     Security Policy Changes         SUCCESS/FAILURE
     Restart, Shutdown, and System   SUCCESS/FAILURE
     Process tracking                SUCCESS/FAILURE

New Accounts Created:
   DATE        TIME         NEW ACCOUNT                       CREATED BY:
 3/22/2000   8:53:32 PM  LAB\test3                            LAB\administrator

Account Enabled:
   DATE        TIME         ACCOUNT                           ENABLED BY:
 3/17/2000   8:56:29 PM  LAB\test                             LAB\administrator

Account Added To Global Group:
   DATE        TIME         GROUP                             ADDED BY:
 3/22/2000   8:53:32 PM  LAB\domain users                     LAB\administrator

Account Added To Local Group:
   DATE        TIME         GROUP                             ADDED BY:
 3/22/2000   9:02:52 PM  NTSERVER1\users                      LAB\administrator

Account Lockouts:
   DATE        TIME         ACCOUNT                           FROM WORKSTATION:
 3/17/2000   8:34:00 PM  test                                 ntworkstation1
```

Suggested Improvements

The script presented here is a good example of how to extract data from the security event logs of several computers. With only a minimum of programming it can be easily modified to fit the auditing needs of most networks. Here are a few enhancements to the script that may be implemented:

1. Sort the temporary file by date and time before processing it. This would cause the report to list all like events in chronological order rather than grouping them by the machine they were generated on.
2. Implement code to lookup the domain name and add it to the account name for account lockouts.
3. Write a routine that converts a user's SID to his user name so that it can be included in the reports of group additions. See Microsoft Knowledge Base article Q174073 – Auditing User Authentication for one possible way to accomplish this.
4. Direct the output of the dumpel.exe utility to a different file for each computer and save them as an audit history.
5. Write code to clear the event logs after they are processed.
6. Use a date and time stamp in the output file name and save as an audit history.
7. Convert all arrays from fixed size to dynamic arrays and increase their size only when necessary. This would minimize the resources required for the script as well as make the script more robust, but may require a substantial amount of code due to the number of arrays used in this script.
8. Implement code to enumerate the list of computers in the domain rather than using the Workstations.txt file.

References

Hill, Tim. *Windows 2000 Windows Script Host.* Macmillan Technical Publishing, 1999. ISBN 1057870-139-2

Born, Günter. *Microsoft Windows Script Host 2.0 Developer's Guide.* Microsoft Press, 2000. ISBN 0-7356-0931-4

Childs, Matt; Lomax, Paul; Petrusha, Ron. *VBScript in a Nutshell.* O' Reilly & Associates, 2000. ISBN 1-56592-720-6

Murray, James. *Windows NT Event Logging.* O' Reilly & Associates, 1998. ISBN 1-56592-514-9

Honeyman, Jeffrey. *Scripting Windows 2000.* The McGraw-Hill Companies, 2000 ISBN 0-07-212444-4

Monitoring and Auditing for End Systems. Microsoft, 2000

GIAC Practical Assignments referenced in this work:

Parish, Ruth Anne. *An In-Dept Examination of Event Viewer and Auditing.*

Laboris, Richard D.. *Developments in Auditing NT.*

Golias, Martin. *Practical T1 Track Parliament Hill, Ottawa.*

Shawgo, Jeff. *Consolidated Security Event Monitoring for Microsoft Windows NT 4.0 Server.*

Toy, Steven. *Centralized Auditing of a Windows NT Computer*.

Appendix A

NOTE: To use or modify this script Highlight the entire script and copy into a text editor such as Notepad. Save the file as ASCII text with the name ‘Auditlogs.vbs’. If necessary hide the page headers and footers by selecting ‘Normal’ from the View menu in Microsoft Word.

```
'**********************************************************************
'This script was written by Roger R. McLaren to complete requirements for
'GIAC Certification in Windows NT Security.
'
'
'Useage: cscript AuditLogs.vbs WorkstationList OutputFile
'     Where: WorkstationList = Path and filename of list of stations to audit
'            OutputFile = Path and filename of outputfile to generate
'
'If no parameters are specified the following defaults are used:
'Workstations list = workstations.txt in the script’s directory
'Output file = logout.txt in the script’s directory
'
'The WorkstationList file is a text file with the name of each
'computer to audit placed on a single line. Do not include the \\ or any
'other special characters. Blank lines and lines that begin with a
'semi-colon are ignored.
'
'**********************************************************************
'This line forces us to declare all variables
Option Explicit

OutputStatus "Program Started"
'**********************************************************************
'declare variables for objects
Dim objFSO
Dim objTextStream1
Dim objWshShell
Dim objArguments
Dim objFile

'declare misc text variables
Dim strInputLine
Dim strCommandLine
Dim strSplitString
Dim strScriptPath
Dim strInputFileName
Dim strOutputFileName
Dim strTempFileName
Dim strWorkstationList(1000)
Dim strOutputString
Dim strTempString

'declare variables for processing event 512 (system startup)
Dim intEvent512Count
Dim strEvent512Date(10000)
Dim strEvent512Time(10000)
Dim strEvent512Workstation(10000)
```

```
'declare variables for processing event 517 (event log clear)
Dim intEvent517Count
Dim strEvent517Date(10000)
Dim strEvent517Time(10000)
Dim strEvent517UserName(10000)
Dim strEvent517Workstation(10000)

'declare variables for processing event 528 (Successful Logon)
Dim intEvent528Count
Dim strEvent528Date(10000)
Dim strEvent528Time(10000)
Dim strEvent528UserName(10000)
Dim strEvent528Workstation(10000)

'declare variables for processing event 529 (failed logon)
Dim intEvent529UserCount
Dim strEvent529UserName(10000)
Dim intEvent529CountForUser(10000)
Dim bolFound
Dim strTempUserName
Dim intEvent529AdminCount
Dim strEvent529AdminDate(10000)
Dim strEvent529AdminTime(10000)
Dim strEvent529AdminName(10000)
Dim strEvent529AdminWorkstation(10000)

'declare variables For processing event 531 (Disabled Logon Attempt)
Dim intEvent531Count
Dim strEvent531Date(10000)
Dim strEvent531Time(10000)
Dim strEvent531UserName(10000)
Dim strEvent531Workstation(10000)

'declare variables for processing event 532 (Expired Logon Attempt)
Dim intEvent532Count
Dim strEvent532Date(10000)
Dim strEvent532Time(10000)
Dim strEvent532UserName(10000)
Dim strEvent532Workstation(10000)

'declare variables for processing event 534 (User not granted requested logon
type)
Dim intEvent534Count
Dim strEvent534Date(10000)
Dim strEvent534Time(10000)
Dim strEvent534UserName(10000)
Dim strEvent534Workstation(10000)

'declare variables for processing event 612 (Audit Policy Change)
Dim intEvent612Count
Dim strEvent612Date(10000)
Dim strEvent612Time(10000)
Dim strEvent612User(10000)
Dim strEvent612Workstation(10000)
Dim strEvent612System(10000)
Dim strEvent612LogonLogoff(10000)
Dim strEvent612ObjectAccess(10000)
Dim strEvent612PrivilageUse(10000)
Dim strEvent612DetailedTracking(10000)
Dim strEvent612PolicyChange(10000)
Dim strEvent612AccountManagement(10000)

'declare variables for processing event 624 (Account Created)
Dim intEvent624Count
Dim strEvent624Date(10000)
Dim strEvent624Time(10000)
Dim strEvent624UserName(10000)
Dim strEvent624Creator(10000)

'declare variables for processing event 626 (Account Enabled)
Dim intEvent626Count
Dim strEvent626Date(10000)
Dim strEvent626Time(10000)
Dim strEvent626UserName(10000)
Dim strEvent626Enabler(10000)

'declare variables for processing event 632 (Account Added to Global Group)
Dim intEvent632Count
Dim strEvent632Date(10000)
Dim strEvent632Time(10000)
Dim strEvent632Group(10000)
Dim strEvent632Adder(10000)

'declare variables for processing event 636 (Account Added to Local Group)
Dim intEvent636Count
Dim strEvent636Date(10000)
Dim strEvent636Time(10000)
Dim strEvent636Group(10000)
Dim strEvent636Adder(10000)

'declare variables for processing event 644 (Account Lockout)
Dim intEvent644Count
Dim strEvent644Date(10000)
Dim strEvent644Time(10000)
Dim strEvent644Account(10000)
Dim strEvent644Workstation(10000)

'declare variables for integer counnters
Dim intLoopCounter
Dim intLoopCounter2
Dim intNumberOfWorkstations

'declare variable for function returns
Dim return

'*****************************************************************************
'initialize variables
intNumberOfWorkstations = 0
intEvent512Count = 0
intEvent517Count = 0
intEvent528Count = 0
intEvent529UserCount = 0
intEvent529AdminCount = 0
intEvent531Count = 0
```

```
intEvent532Count = 0
intEvent534Count = 0
intEvent612Count = 0
intEvent624Count = 0
intEvent626Count = 0
intEvent632Count = 0
intEvent636Count = 0
intEvent644Count = 0
strTempFileName = "AuditLogs.tmp"

'create file system and shell objects
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objWshShell = WScript.CreateObject("WScript.Shell")
Set objArguments = Wscript.Arguments

'Get the path to this script
strScriptPath = left(wscript.ScriptFullName,
InStrRev(wscript.ScriptFullName,"\"))

'Check to see if we any arguments are passed. If not, use the defaults:
'Workstations list = workstations.txt in the application directory
'Output file = Logout.txt in the application directory

'If 2 arguments are received they are assigned
'to the input and output file name, otherwise error.

If objArguments.count = 0 Then
    strInputFileName = strScriptPath & "workstations.txt"
    strOutputFileName = strScriptPath & "logout.txt"
Elseif objArguments.count = 2 Then
    strInputFileName = objArguments(0)
    strOutputFileName = objArguments(1)
Else
    wscript.echo "Useage: cscript AuditLogs.vbs WorkstationList OutputFile"
    wscript.echo "      Where: WorkstationList = Path and filename " & _
        "of list of stations to audit"
    wscript.echo "             OutputFile = Path and filename of outputfile"
    wscript.quit
End If

'Check if Input File exists
If Not objFSO.FileExists(strInputFileName) Then
    wscript.echo "Input file doesn't exist!!!"
    wscript.echo
    wscript.echo "Useage: cscript AuditLogs.vbs WorkstationList OutputFile"
    wscript.echo "      Where: WorkstationList = Path and filename " & _
        "of list of stations to audit"
    wscript.echo "             OutputFile = Path and filename of outputfile"
    wscript.quit
End If


'Initialize temporary file to dump logs to
Set objFile = objFSO.CreateTextFile(strScriptPath & strTempFileName, True)
objFile.close

'Open input file and call the dumpel command for each entry
```

```
'Skip blank lines and lines that start with a ;
Set objTextStream1 = objFSO.OpenTextFile(strInputFileName)
Do while Not objTextStream1.AtEndOfStream
    strInputLine = objTextStream1.ReadLine
    If len(trim(strInputLine)) > 0 And left(strInputLine,1) <> ";" Then
        intNumberOfWorkstations = intNumberOfWorkstations + 1
        strWorkstationList(intNumberOfWorkstations) = _
            ucase(trim(strInputLine))

        'Create commandline to run the dumpel resource kit utility and
        'Pipe the output into our temp file.
        'If the dumpel.exe is not found, use a fully qualified path in this
        'line
        '(for example c:\progra~1\ntreskit\dumpel.exe ...)
        'syntax is as follows:
        ' -s servername          (name of server to dump log from)
        ' -t                     (Use tabs for seperator)
        ' -l logfile             (name of log to dump - security, application,
        'or system)
        strCommandLine = "cmd /c dumpel -s " & trim(strInputLine) & _
            " -t -l security >> " & strScriptPath & strTempFileName

        'run the dumpel utility.
            OutputStatus "Querying " & ucase(trim(strInputLine))
            return=objWshShell.Run (strCommandLine,0,True)
    End If
Loop
objTextStream1.close

OutputStatus "Processing Log Data"
'open the text file we just created
SET objTextStream1 = objFSO.OpenTextFile(strScriptPath & strTempFileName)

'loop to read the text file and process each line
Do While Not objTextStream1.AtEndOfStream
    strInputLine = objTextStream1.ReadLine
    strSplitString = split(strInputLine, chr(9))


'***************************************************************************
'Process Event 512 - System Startup
    If strSplitString(4) = "512" Then
        intEvent512Count = intEvent512Count +1
        strEvent512Date(intEvent512Count) = strSplitString(0)
        strEvent512Time(intEvent512Count) = strSplitString(1)
        strEvent512Workstation(intEvent512Count) = trim(strSplitString(8))
    End If
'***************************************************************************

'***************************************************************************
'Process event 517 - Event log clear
        If strSplitString(4) = "517" Then
        intEvent517Count = intEvent517Count + 1
        strEvent517Date(intEvent517Count) = strSplitString(0)
        strEvent517Time(intEvent517Count) = strSplitString(1)
        strEvent517UserName(intEvent517Count) = _
            ucase(trim(strSplitString(19))) & _
            "\" & lcase(trim(strSplitString(17)))
        strEvent517Workstation(intEvent517Count) = trim(strSplitString(8))
    End If
'**************************************************************************

'**************************************************************************
'Process Event 528 - Successful Logon
    If strSplitString(4) = "528" Then
        If ucase(trim(strSplitString(11))) = "ADMINISTRATOR" Then
            intEvent528Count = intEvent528Count +1
            strEvent528Date(intEvent528Count) = strSplitString(0)
            strEvent528Time(intEvent528Count) = strSplitString(1)
            strEvent528UserName(intEvent528Count) = _
                ucase(trim(strSplitString(14))) & _
                "\" & lcase(trim(strSplitString(11)))
            strEvent528Workstation(intEvent528Count) = _
                trim(strSplitString(25))
            if left(strEvent528Workstation(intEvent528Count),2) = "\\" then
                strEvent528Workstation(intEvent528Count) = _
                right(strEvent528Workstation(intEvent528Count), _
                    (len(strEvent528Workstation(intEvent528Count))-2))
            End If
        End If
    End If
'**************************************************************************

'**************************************************************************
'Process Event 529 - Failed Logon
    If strSplitString(4) = "529" Then
        If ucase(trim(strSplitString(14))) = "ADMINISTRATOR" Then
            intEvent529AdminCount = intEvent529AdminCount + 1
            strEvent529AdminDate(intEvent529AdminCount)= strSplitString(0)
            strEvent529AdminTime(intEvent529AdminCount) = strSplitString(1)
            strEvent529AdminName(intEvent529AdminCount) =  _
                ucase(trim(strSplitString(17))) & "\" & _
                trim(lcase(strSplitString(14)))
            strEvent529AdminWorkstation(intEvent529AdminCount) = _
                trim(strSplitString(25))
        End if
        If ucase(trim(strSplitString(14))) <> "ADMINISTRATOR" Then
            bolFound = False
            strTempUserName = ucase(trim(strSplitString(17))) & "\" & _
                trim(lcase(strSplitString(14)))
            If intEvent529UserCount = 0 Then
                intEvent529UserCount = 1
                strEvent529UserName(intEvent529UserCount) = strTempUserName
                intEvent529CountForUser(intEvent529UserCount) = 1
            Else
                For intLoopCounter = 1 To intEvent529UserCount
                If strEvent529UserName(intLoopCounter) = strTempUserName Then
                        intEvent529CountForUser(intLoopCounter) = _
                            intEvent529CountForUser(intLoopCounter) + 1
                        bolFound = True
                    End If
                Next
                If bolFound = False Then
                    intEvent529UserCount = intEvent529UserCount + 1
                    strEvent529UserName(intEvent529UserCount) = _
                        strTempUserName
                    intEvent529CountForUser(intEvent529UserCount) = 1
                End If
            End If
        End If
    End If

'**************************************************************************

'**************************************************************************
'Process event 531 - Disabled Logon Attempt
    If strSplitString(4) = "531" Then
        intEvent531Count = intEvent531Count + 1
        strEvent531Date(intEvent531Count) = strSplitString(0)
        strEvent531Time(intEvent531Count) = strSplitString(1)
        strEvent531UserName(intEvent531Count) = _
            ucase(trim(strSplitString(17))) & "\" & _
            lcase(trim(strSplitString(14)))
        strEvent531Workstation(intEvent531Count) = trim(strSplitString(8))
    End If
'**************************************************************************

'**************************************************************************
'Process event 532 - Expired Logon Attempt
    If strSplitString(4) = "532" Then
        intEvent532Count = intEvent532Count + 1
        strEvent532Date(intEvent532Count) = strSplitString(0)
        strEvent532Time(intEvent532Count) = strSplitString(1)
        strEvent532UserName(intEvent532Count) = _
            ucase(trim(strSplitString(17))) & "\" & _
            lcase(trim(strSplitString(14)))
        strEvent532Workstation(intEvent532Count) = trim(strSplitString(8))
    End If
'**************************************************************************

'**************************************************************************
'Process Event 534 - User not granted requested logon type
'This is the event generated when an un-authorized user attempts
'a local logon to a server
    If strSplitString(4) = "534" Then
        intEvent534Count = intEvent534Count +1
        strEvent534Date(intEvent534Count) = strSplitString(0)
        strEvent534Time(intEvent534Count) = strSplitString(1)
        strEvent534UserName(intEvent534Count) = _
            ucase(trim(strSplitString(18))) & "\" & _
            lcase(trim(strSplitString(15)))
        strEvent534Workstation(intEvent534Count) = trim(strSplitString(26))
    End If
'**************************************************************************

'**************************************************************************
'Process Event 612 - Audit Policy Change
    If strSplitString(4) = "612" Then
        intEvent612Count = intEvent612Count + 1
        strEvent612Date(intEvent612Count) = strSplitString(0)
        strEvent612Time(intEvent612Count) = strSplitString(1)
```

```
strEvent612User(intEvent612Count) = _
    ucase(trim(strSplitString(36))) & "\" & _
    lcase(trim(strSplitString(34)))
strEvent612Workstation(intEvent612Count) = trim(strSplitString(8))
strEvent612System(intEvent612Count) = "NONE"
If trim(strSplitString(12)) = "+" Then
    strEvent612System(intEvent612Count) = "SUCCESS"
End If
If trim(strSplitString(13)) = "+" And _
    strEvent612System(intEvent612Count) = "SUCCESS" Then
        strEvent612System(intEvent612Count) = "SUCCESS/FAILURE"
End If
If trim(strSplitString(13)) = "+" And _
    strEvent612System(intEvent612Count) = "NONE" Then
        strEvent612System(intEvent612Count) = "FAILURE"
End If
strEvent612LogonLogoff(intEvent612Count) = "NONE"
If trim(strSplitString(15)) = "+" Then
    strEvent612LogonLogoff(intEvent612Count) = "SUCCESS"
End If
If trim(strSplitString(16)) = "+" And _
    strEvent612LogonLogoff(intEvent612Count) = "SUCCESS" Then
        strEvent612LogonLogoff(intEvent612Count) = "SUCCESS/FAILURE"
End If
If trim(strSplitString(16)) = "+" And _
    strEvent612LogonLogoff(intEvent612Count) = "NONE" Then
        strEvent612LogonLogoff(intEvent612Count) = "FAILURE"
End If
strEvent612ObjectAccess(intEvent612Count) = "NONE"
If trim(strSplitString(18)) = "+" Then
    strEvent612ObjectAccess(intEvent612Count) = "SUCCESS"
End If
If trim(strSplitString(19)) = "+" And _
    strEvent612ObjectAccess(intEvent612Count) = "SUCCESS" Then
        strEvent612ObjectAccess(intEvent612Count) = "SUCCESS/FAILURE"
End If
If trim(strSplitString(19)) = "+" And _
    strEvent612ObjectAccess(intEvent612Count) = "NONE" Then
        strEvent612ObjectAccess(intEvent612Count) = "FAILURE"
End If
strEvent612PrivilageUse(intEvent612Count) = "NONE"
If trim(strSplitString(21)) = "+" Then
    strEvent612PrivilageUse(intEvent612Count) = "SUCCESS"
End If
If trim(strSplitString(22)) = "+" And _
    strEvent612PrivilageUse(intEvent612Count) = "SUCCESS" Then
        strEvent612PrivilageUse(intEvent612Count) = "SUCCESS/FAILURE"
End If
If trim(strSplitString(22)) = "+" And _
    strEvent612PrivilageUse(intEvent612Count) = "NONE" Then
        strEvent612PrivilageUse(intEvent612Count) = "FAILURE"
End If
strEvent612DetailedTracking(intEvent612Count) = "NONE"
If trim(strSplitString(24)) = "+" Then
    strEvent612DetailedTracking(intEvent612Count) = "SUCCESS"
End If
If trim(strSplitString(25)) = "+" And _
```

```
                strEvent612DetailedTracking(intEvent612Count) = "SUCCESS" Then
                    strEvent612DetailedTracking(intEvent612Count) = _
                        "SUCCESS/FAILURE"
            End If
            If trim(strSplitString(25)) = "+" And _
                strEvent612DetailedTracking(intEvent612Count) = "NONE" Then
                    strEvent612DetailedTracking(intEvent612Count) = "FAILURE"
            End If
            strEvent612PolicyChange(intEvent612Count) = "NONE"
            If trim(strSplitString(27)) = "+" Then
                strEvent612PolicyChange(intEvent612Count) = "SUCCESS"
            End If
            If trim(strSplitString(28)) = "+" And _
                strEvent612PolicyChange(intEvent612Count) = "SUCCESS" Then
                    strEvent612PolicyChange(intEvent612Count) = "SUCCESS/FAILURE"
            End If
            If trim(strSplitString(28)) = "+" And _
                strEvent612PolicyChange(intEvent612Count) = "NONE" Then
                    strEvent612PolicyChange(intEvent612Count) = "FAILURE"
            End If
            strEvent612AccountManagement(intEvent612Count) = "NONE"
            If trim(strSplitString(30)) = "+" Then
                strEvent612AccountManagement(intEvent612Count) = "SUCCESS"
            End If
            If trim(strSplitString(31)) = "+" And _
                strEvent612AccountManagement(intEvent612Count) = "SUCCESS" Then
                strEvent612AccountManagement(intEvent612Count) = _
                    "SUCCESS/FAILURE"
            End If
            If trim(strSplitString(31)) = "+" And _
                strEvent612AccountManagement(intEvent612Count) = "NONE" Then
                    strEvent612AccountManagement(intEvent612Count) = "FAILURE"
            End If
        End If
'**************************************************************************

'**************************************************************************
'Process Event 624 - Account Created
        If strSplitString(4) = "624" Then
            intEvent624Count = intEvent624Count +1
            strEvent624Date(intEvent624Count) = strSplitString(0)
            strEvent624Time(intEvent624Count) = strSplitString(1)
            strEvent624UserName(intEvent624Count) = _
                ucase(trim(strSplitString(13))) & "\" & _
                lcase(trim(strSplitString(11)))
            strEvent624Creator(intEvent624Count) = _
                ucase(trim(strSplitString(19))) & "\" & _
                lcase(trim(strSplitString(17)))
        End If

'**************************************************************************

'**************************************************************************
'Process Event 626 - Account Enabled
        If strSplitString(4) = "626" Then
            intEvent626Count = intEvent626Count +1
            strEvent626Date(intEvent626Count) = strSplitString(0)
            strEvent626Time(intEvent626Count) = strSplitString(1)
            strEvent626UserName(intEvent626Count) = _
                ucase(trim(strSplitString(13))) & "\" & _
                lcase(trim(strSplitString(11)))
            strEvent626Enabler(intEvent626Count) = _
                ucase(trim(strSplitString(19))) & "\" & _
                lcase(trim(strSplitString(17)))
        End If
'***************************************************************************

'***************************************************************************
'Process Event 632 - Account Added to Global Group
        If strSplitString(4) = "632" Then
            intEvent632Count = intEvent632Count +1
            strEvent632Date(intEvent632Count) = strSplitString(0)
            strEvent632Time(intEvent632Count) = strSplitString(1)
            strEvent632Group(intEvent632Count) = _
                ucase(trim(strSplitString(15))) & "\" & _
                lcase(trim(strSplitString(13)))
            strEvent632Adder(intEvent632Count) = _
                ucase(trim(strSplitString(21))) & "\" & _
                lcase(trim(strSplitString(19)))
        End If
'***************************************************************************

'***************************************************************************
'Process Event 636 - Account Added to Local Group
        If strSplitString(4) = "636" Then
            intEvent636Count = intEvent636Count +1
            strEvent636Date(intEvent636Count) = strSplitString(0)
            strEvent636Time(intEvent636Count) = strSplitString(1)
            strEvent636Group(intEvent636Count) = _
                ucase(trim(strSplitString(8))) & "\" & _
                lcase(trim(strSplitString(13)))
            strEvent636Adder(intEvent636Count) = _
                ucase(trim(strSplitString(25))) & "\" & _
                lcase(trim(strSplitString(22)))
        End If
'***************************************************************************

'***************************************************************************
'Process Event 644 - Account Lockout
        If strSplitString(4) = "644" Then
            intEvent644Count = intEvent644Count +1
            strEvent644Date(intEvent644Count) = strSplitString(0)
            strEvent644Time(intEvent644Count) = strSplitString(1)
            strEvent644Account(intEvent644Count) = _
                lcase(trim(strSplitString(11)))
            strEvent644Workstation(intEvent644Count) = _
                lcase(trim(strSplitString(15)))
        End If
'***************************************************************************

'***************************************************************************
'End of the loop - go back and get the next line
Loop
'***************************************************************************
```

```
'*************************************************************************
'Close Input File
objTextStream1.close
'*************************************************************************

'*************************************************************************
'Delete Temp File
objFSO.DeleteFile strScriptPath & strTempFileName
'*************************************************************************

'*************************************************************************
'Output Routine
'Open Output File - Overwrite if it already exists
OutputStatus "Creating Output File"
Set objTextStream1 = objFSO.CreateTextFile(strOutputFileName, True)
'*************************************************************************

'*************************************************************************
'Output Header Line
    objTextStream1.WriteLine("                Audit of Security Logs " & _
        date & " " & time )
    objTextStream1.WriteLine
'*************************************************************************

'*************************************************************************
'Output the names of machines in the audit
    objTextStream1.WriteLine("Computers Audited:")
    For intLoopCounter = 1 To intNumberOfWorkstations step 2
    strOutputString = left(strWorkstationList(intLoopCounter) & _
        Space(50),50) & strWorkstationList(intLoopCounter +1)

    objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'*************************************************************************

'*************************************************************************
'Output Routine for Event 528 (Successful Logon) for Administrator Logons
    objTextStream1.WriteLine("Administrative Logons:")
    objTextStream1.WriteLine("   DATE         TIME            " & _
        "ACCOUNT                  LOGON TO:")
    For intLoopCounter = 1 To intEvent528Count
        strOutputString = right(space(10) & _
            strEvent528Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent528Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent528UserName(intLoopCounter) & _
            space(55),55) & strEvent528Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'*************************************************************************

'*************************************************************************
```

```
'Output Routines for Event 529 - (Failed Logon) for Administrator Logon
Failures
    objTextStream1.WriteLine("Administrative Logon Failures:")
    objTextStream1.WriteLine("   DATE         TIME                " & _
        "ACCOUNT                 FAILED LOGON TO:")
    For intLoopCounter = 1 To intEvent529AdminCount
        strOutputString = right(space(10) & _
            strEvent529AdminDate(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent529AdminTime(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent529AdminName(intLoopCounter) & _
            space(55),55) & strEvent529AdminWorkstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'*************************************************************************

'*************************************************************************
'Output Routines for Event 529 - (Failed Logon) for General Logon Failures
    objTextStream1.WriteLine("General Logon Failures:")
    objTextStream1.WriteLine("   ACCOUNT                     # OF FAILURES")
    For intLoopCounter = 1 To intEvent529UserCount
        strOutputString = left(strEvent529UserName(intLoopCounter) & _
            space(32),32)
        strOutputString = strOutputString & _
            intEvent529CountForUser(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'*************************************************************************

'*************************************************************************
'Output Routines for Event 531 - (Disabled Logon Attempt)
    objTextStream1.WriteLine("Attempts to Logon to a Disabled Account:")
    objTextStream1.WriteLine("   DATE         TIME                " & _
        "ACCOUNT                 FAILED LOGON TO:")
    For intLoopCounter = 1 To intEvent531Count
        strOutputString = right(space(10) & _
            strEvent531Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent531Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent531UserName(intLoopCounter) & _
            space(55),55) & strEvent531Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'*************************************************************************

'*************************************************************************
'Output Routines for Event 532 - (Expired Logon Attempt)
    objTextStream1.WriteLine("Attempts to Logon to an Expired Account:")
    objTextStream1.WriteLine("   DATE         TIME                " & _
        "ACCOUNT                 FAILED LOGON TO:")
    For intLoopCounter = 1 To intEvent532Count
        strOutputString = right(space(10) & _
            strEvent532Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent532Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent532UserName(intLoopCounter) & _
            space(55),55) & strEvent532Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routines for Event 534 - (User Not Granted Requested Logon Type)
    objTextStream1.WriteLine("User Not Granted Requested Logon Type:")
    objTextStream1.WriteLine("   DATE         TIME              " & _
        "ACCOUNT                  FAILED LOGON TO:")
    For intLoopCounter = 1 To intEvent534Count
        strOutputString = right(space(10) & _
            strEvent534Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent534Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent534UserName(intLoopCounter) & _
            space(55),55) & strEvent534Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 512 (System Starting Up)
    objTextStream1.WriteLine("System Started:")
    objTextStream1.WriteLine("   DATE         TIME              " & _
        "SYSTEM")
    For intLoopCounter = 1 To intEvent512Count
        strOutputString = right(space(10) & _
            strEvent512Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent512Time(intLoopCounter),13)
        strOutputString = strOutputString & "   " & _
            strEvent512Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 517 (Security Audit Log Clear)
    objTextStream1.WriteLine("Security Audit Log Cleared:")
    objTextStream1.WriteLine("   DATE         TIME              " & _
        "SYSTEM                   BY:")
    For intLoopCounter = 1 To intEvent517Count
        strOutputString = right(space(10) & _
            strEvent517Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent517Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent517Workstation(intLoopCounter) & _
            space(55),55) & strEvent517UserName(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 612 (Audit Policy Change)
    objTextStream1.WriteLine("Audit Policy Changes:")
    objTextStream1.WriteLine("   DATE        TIME              " & _
        "SYSTEM                  BY:")
    For intLoopCounter = 1 To intEvent612Count
        strOutputString = right(space(10) & _
            strEvent612Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent612Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent612Workstation(intLoopCounter) & _
            space(55),55) & strEvent612User(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        objTextStream1.WriteLine("New Effective Policy:")
        strOutputString = "      Logon And Logoff                  " & _
            strEvent612LogonLogoff(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        strOutputString = "      File And Object Access            " & _
            strEvent612ObjectAccess(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        strOutputString = "      Use Of User Rights                " & _
            strEvent612PrivilageUse(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        strOutputString = "      User And Group Management         " & _
            strEvent612AccountManagement(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        strOutputString = "      Security Policy Changes           " & _
            strEvent612PolicyChange(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        strOutputString = "      Restart, Shutdown, and System     " & _
            strEvent612System(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
        strOutputString = "      Process tracking                  " & _
            strEvent612DetailedTracking(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    objTextStream1.WriteLine
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 624 (New Account Created)
    objTextStream1.WriteLine("New Accounts Created:")
    objTextStream1.WriteLine("   DATE        TIME           " & _
        "NEW ACCOUNT                CREATED BY:")
    For intLoopCounter = 1 To intEvent624Count
        strOutputString = right(space(10) & _
            strEvent624Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent624Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent624UserName(intLoopCounter) & _
            space(55),55) & strEvent624Creator(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 626 (Account Enabled)
    objTextStream1.WriteLine("Account Enabled:")
    objTextStream1.WriteLine("   DATE         TIME           " & _
        "ACCOUNT                     ENABLED BY:")
    For intLoopCounter = 1 To intEvent626Count
        strOutputString = right(space(10) & _
            strEvent626Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent626Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent626UserName(intLoopCounter) & _
            space(55),55) & strEvent626Enabler(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine

'**************************************************************************

'**************************************************************************
'Output Routine for Event 632 (Account Added to Global Group)
    objTextStream1.WriteLine("Account Added To Global Group:")
    objTextStream1.WriteLine("   DATE         TIME           " & _
        "GROUP                       ADDED BY:")
    For intLoopCounter = 1 To intEvent632Count
        strOutputString = right(space(10) & _
            strEvent632Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent632Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent632Group(intLoopCounter) & _
            space(55),55) & strEvent632Adder(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 636 (Account Added to Local Group)
    objTextStream1.WriteLine("Account Added To Local Group:")
    objTextStream1.WriteLine("   DATE         TIME           " & _
        "GROUP                       ADDED BY:")
    For intLoopCounter = 1 To intEvent636Count
        strOutputString = right(space(10) & _
            strEvent636Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent636Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent636Group(intLoopCounter) & _
            space(55),55) & strEvent636Adder(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
'Output Routine for Event 644 (Account Locked Out)
    objTextStream1.WriteLine("Account Lockouts:")
    objTextStream1.WriteLine("   DATE        TIME          " & _
        "ACCOUNT                    FROM WORKSTATION:")
    For intLoopCounter = 1 To intEvent644Count
        strOutputString = right(space(10) & _
            strEvent644Date(intLoopCounter),10)
        strOutputString = strOutputString & right(space(13) & _
            strEvent644Time(intLoopCounter),13)
        strOutputString = left(strOutputString & "   " & _
            strEvent644Account(intLoopCounter) & _
            space(55),55) & strEvent644Workstation(intLoopCounter)
        objTextStream1.WriteLine(strOutputString)
    Next
    objTextStream1.WriteLine
'**************************************************************************

'**************************************************************************
objTextStream1.close
OutputStatus "Program Complete"
'**************************************************************************

'**************************************************************************
'Output Status Routine
'This routine outputs status messages. This is used because the Wscript.Echo
'method causes the script to pause until the message box is dismissed when
'running under the Wscript host, therefore when running under Wscript we
'use the Wscript.Shell Popup method.
Sub OutputStatus(strStatus)

Dim objWshShell2
Dim strOutput

strOutput = time & " " & strStatus
Set objWshShell2 = WScript.CreateObject("WScript.Shell")
    If ucase(right(wscript.FullName,11))="CSCRIPT.EXE" Then
        wscript.echo strOutput
    End If
    If ucase(right(wscript.FullName,11))="WSCRIPT.EXE" Then
            return=objWshShell2.Popup (strOutput,1)
    End If
End Sub
'**************************************************************************
```

Appendix B

```
'*************************************************************************
'
'This script is an example of how to subscribe to the Event Notification
'Service of a remote machine using a notification query
'
'*************************************************************************
set objLocator = CreateObject("WbemScripting.SWbemLocator")

objLocator.Security_.Privileges.AddAsString "SeSecurityPrivilege"

strQuery = "select * from __instancecreationevent " & _
    where targetinstance isa 'Win32_NTLogEvent'"
set objEventSet = objLocator.ConnectServer("\\ntserver1")._
    ExecNotificationQuery(strQuery)
do
    set objEvent = events.nextevent
    wscript.echo "Event Number = " & objEevent.TargetInstance.EventCode
    Wscript.echo "Message ="
    WScript.Echo objEvent.TargetInstance.Message
loop
```

Appendix C

```
'**************************************************************************
'
'This script is an example of how to query Windows NT event logs for a
'specific event type (Event 528 - Successful Logon)
'
'**************************************************************************
strQuery = "select * from Win32_NTLogEvent where EventCode=528"
Set EventCollection = GetObject("winmgmts:\\NtServer1").ExecQuery(strQuery)

for each LogEvent in EventCollection
    WScript.Echo "Event Number: " & LogEvent.EventCode
    WScript.Echo "Time: " & LogEvent.TimeGenerated
    WScript.Echo "Message: " & LogEvent.Message
    WScript.Echo ""
next
```