Global Information Assurance Certification Paper

Consolidated Security Event Monitoring for Microsoft Windows NT 4.0 Server

Jeff Shawgo
GCNT Practical Assignment
Network Security 2000
Monterey, CA
November 20, 2000

Overview:

This document is the fulfillment of the Practical Assignment for the Global Incident Analysis Center (GIAC) Level Two Certification GCNT, Developments in Auditing for Microsoft Windows NT 4.0. The enclosed procedures are intended to be used on any size network which requires distributed monitoring of Microsoft Windows NT Audit Logs. This model supports a small or medium security network, with a wide range in numbers of Windows NT Servers and/or Workstations. It may be possible to scale these solutions to a large network as well.

Auditing a Microsoft Windows NT 4.0 network is one of the most basic responsibilities of a system administrator. Auditing needs to be manually enabled on a new installation of Windows NT. Unfortunately, enabling monitoring does little good if the system administrator does not have time to review the audit logs. In a larger environment, the problem is compounded by the fact that logs tend to be more plentiful, and systems administrators tend to be distracted by more urgent matters. According to Sherri Heckendorn's SANS GCNT Practical Assignment, regarding review of NT Event Logs, "The network has no formal policy covering review of audit logs. Network administrator stated that logs are reviewed, 'When I have time, which isn't often.'"

Any system administrator of a Microsoft Windows NT 4.0 Server based network is likely to have tried to centralize Event Logs from multiple Windows NT 4.0 Workstations and Servers. Those same administrators can also attest to the lack of tools available to effectively manage the auditing of Event Logs on multiple Windows NT computers. The focus of this document is to find a secure, cost-effective solution to fill this gap.

Disclaimer:

This document has been written as a practical assignment for the Global Incident Analysis Center NT Security Certification. The techniques discussed here may be used in a small to medium sized network. They have been tested and found to cause no major harm to the systems involved, when used as directed.

With any tool, technique, or piece of intelligence that facilitates an automated process on an Information System, there exists a potential for damage if misused. Please take care when using this or any other process on a production network. As always, please test any new procedures before introducing them into production.

Scope:

The scope of this assignment includes, specifically, the transfer of NT 4.0 audit logs from various Windows NT Servers to a central repository for the purposes of analysis and archival in a timely manner. The scope does not include generation of those events.

This document uses products in a Microsoft Windows NT 4.0 (Service Pack 6a) environment. Many of the technologies and products discussed here may be applied or adapted to a Microsoft Windows 2000 environment; however, Windows 2000 machines are beyond the scope of this GIAC NT Practical Assignment.

Background:

The Microsoft Windows NT operating system has traditionally been considered a "young" operating system, in comparison to Unix, VMS, and Mainframe platforms. Windows operating systems are also strongly targeted for their weaknesses, due to their youth, popularity, and the fact that Microsoft has favored "Ease of Use" and "Backwards Compatibility" over security in the standard "out-of-the-box" configuration. In response to this perceived lack of security in the Windows operating systems, an effective security plan needs more than strong security configurations and settings. For a security plan to be truly effective, a vigilant watch must be kept over the built-in Event Log to spot activity indicating early phases of network based attacks.

The problem is this: Microsoft Windows NT Event Logs are local to each workstation and server, and are distributed by design. An integrated event log consolidation mechanism does not exist in the native Windows NT environment. Manually checking event logs in a distributed environment is an excruciating, time-consuming task, which can become a full-time job in itself. In many cases, this function is not done properly, and becomes more of a liability than a benefit.

Third party tools are available to centralize event logs. Enterprise System Management tools, such as Tivoli and CA Unicenter, exist to perform this function. These tools, while effective, are expensive and require significant infrastructure to support themselves. Most Enterprise Management tools were originally written for Unix and Mainframe systems, and have been "ported" to cover Microsoft products. Less expensive tools such as Event Log Monitor from TNT Software can perform this function, but still take support dollars from your budget.

The goal of this practical assignment is to identify procedures for centralizing multiple Windows NT event logs to a central repository, for the purposes of analysis and archiving. The secondary goal of this document is to accomplish this task at minimum cost, with few manual processes. The only tools to be discussed here (with the exception of the central repository itself) are either inexpensive or free, and will include the tools in the Microsoft Windows NT Server 4.0 Resource Kit.

Test Environment:

The test environment is designed to represent the target server(s), repository server, and front-end analysis workstation(s). For the purposes of this document, all three of these functions will run on the same machine, a Microsoft Windows NT 4.0 Server PDC, in accordance with GIAC NT requirements. Software requirements for each of these roles are detailed in Appendix A.

The ideal target server(s) should mimic a production environment of various Windows NT 4.0 machines that need to be monitored. These machines may include an isolated intranet, or part of a corporate extranet.

The event repository will reside on the local machine, and will use an Access 2000 database. Microsoft Office 2000 Premium is installed on the collection machine. In a production environment, this function would best be served by an ODBC data source, such as Microsoft SQL Server 7.0 or Microsoft SQL Server 2000.

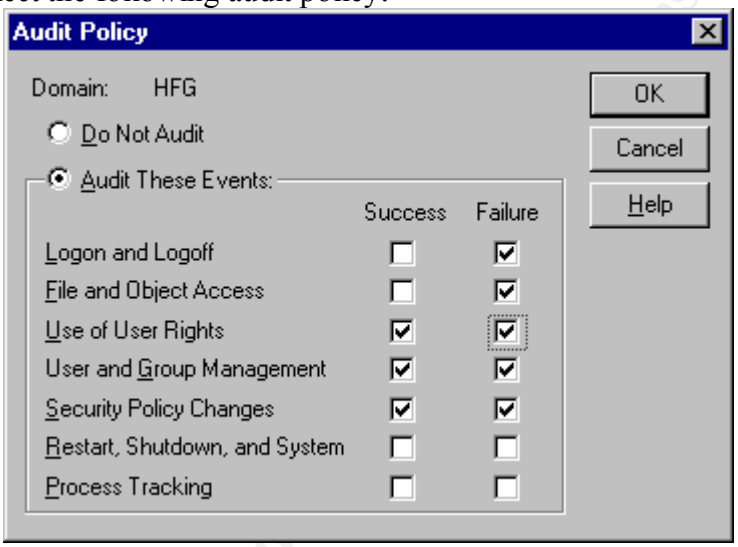
As a user interface, any ODBC compliant application would be sufficient. Here, Access 2000 will still serve this function. In a production environment, a skilled

application developer/system administrator may choose to web-enable this function in order to allow any browser to serve as the application front end.

Test Requirements:

Minimum Requirements:

At a minimum, any event consolidation system needs to do exactly as it implies: Consolidate events from the Application, Security, and System Event Logs into a single, central database. In accordance with the SANS NT Practical Assignment by Howard F. Gabert (August, 2000) NT Security Auditing on a newly installed NT System should be changed to reflect the following audit policy:



Measurements:

There are many methods available to collect and collate events from distributed Windows NT Servers to a central location. This document will discuss some of these options, their implementation, their strengths, and their weaknesses. In order to compare these different methods, several factors can be used to "score" the different methods against each other. The following factors will be taken into consideration for each method of event collection under discussion, and assigned a score of 0 to 10. These scores are subjective in nature, but can still be helpful in comparing one method to another:

- Automation – The goal is to have a fully automated process (10) as opposed to a completely manual process (0).
- Non-NetBIOS Connectivity – NetBIOS connectivity is the first and foremost protocol which should be removed from internet-connected servers. As a rule, NetBIOS should not be allowed through a firewall. A collection process that is independent of NetBIOS (10) would score better than one that cannot exist without NetBIOS (0).
- Accountability – The process of centralizing event logs should be recorded by the event logs as well (10), not immune to it (0).

- Central Code Base – In the event that a script needs to be changed, a central code base for that change (10) will be far easier to implement than a code change required on each monitored machine (0).
- Simplicity – As with any process, "Keep It Simple & Stupid." A simple process (10) will be easier to learn, implement, and maintain than a complex (0) one.
- Timeliness – Events are reported to the event log in real-time. For a collection effort to be effective, it needs to be performed in near real-time (10).
- Security – The purpose of collecting event log information is to increase the overall security of an enterprise. Any collection effort that requires a weakness of the overall security (0) is self-defeating.
- Other – Factors that are outside the scope of the existing metrics should still be considered, whether very good (+10) or very bad (-10).

Options:

The following collection models are available and are subject to discussion for comparison to complete this task:

Option #1 Distributed Batch Processing – This process involves placing an agent on each machine, and having that machine "push" its events to the central event repository using conventional batch tools.

Option #2 Centralized Batch Processing – Much like the previous option, this involves running a collection process from a central location, and "pulling" the data from the monitored machine using conventional batch tools.

Option #3 Centralized Windows Management Interface (WMI) processing – WMI is a relatively young technology used to manage Windows NT 4.0 machines since the introduction of Service Pack 4. WMI, in conjunction with the Windows Scripting Host, allows scripted operations to be performed on remote machines through an authenticated channel.

Option #1 - Distributed Batch Processing

Background:

Distributed Batch Processing is reminiscent of those nostalgic days of DOS and Windows 3.1, where mastery of the operating system involved intimate knowledge and manipulation of the AUTOEXEC.BAT and other batch files. These "batch" files have evolved into complex programs, which run under the Command Line Interpreter (CLI), processing each batch file line by line. The CLI was originally COMMAND.EXE in the DOS, Windows 3.x, and Windows 95/98, where it was the foundation of the operating system. With the advent of Windows NT came the change to CMD.EXE, which became merely another supplemental application in a much larger collection of operating system files and services.

Collection tools:

Two excellent utilities are available for exporting Event Log data from the log itself into a flat file. The NT Resource Kit utility "DUMPEL.EXE" performs a basic dump of the Event Log. The freeware program "DumpEvt" by Somarsoft (www.systemtools.com/somarsoft) is a similar but more robust utility, which does an excellent job. The fact that DumpEvt resolves some of DUMPEL.EXE's database inconsistencies make it more suitable, and make it the tool of choice for this option.

Scheduling tools:

Like most batch script tools, the scheduling tool used here will be the internal Windows NT Scheduler service for execution on each client.

Software requirements:

- Each client needs to have Somarsoft's DumpEvt installed.
- Each client needs to have the "Scheduler" service running, with automatic startup.
- The collection server needs to have an ODBC database installed. In this case, Microsoft Access 2000 will be present.
- Each administrative workstation needs Access 2000 as well.

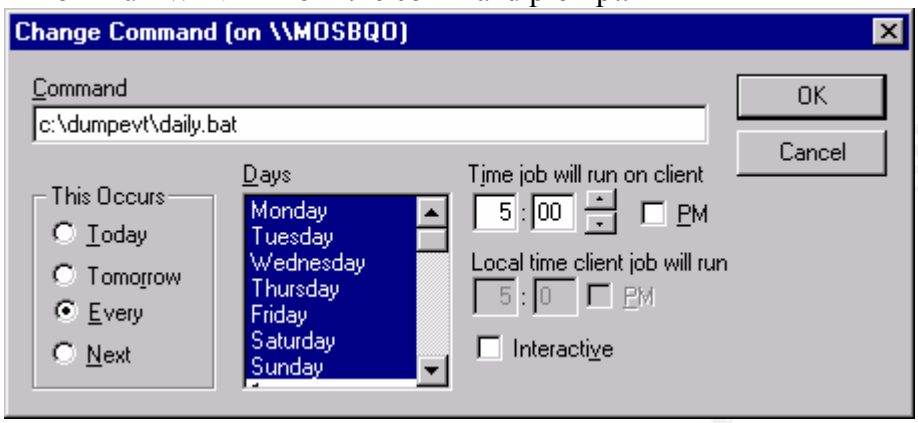
Event Log Collection Process:

On each client:

- Install DumpEvt to a folder (C:\dumpevt)
- Create another folder for log files, and share it (C:\logs shared as logs$)
- Set file and share level security to allow no more access than necessary.
- Start a daily schedule to execute a batch file each day, at 5:00 AM
  Type the following at a command prompt:

```
AT 5:00 /EVERY:SA, SU, M, TU, W, TH, F "C:\DUMPEVT\DAILY.BAT"
```

- The same procedure can be executed with the WINAT Resource Kit utility:
  - Run WINAT from the command prompt.



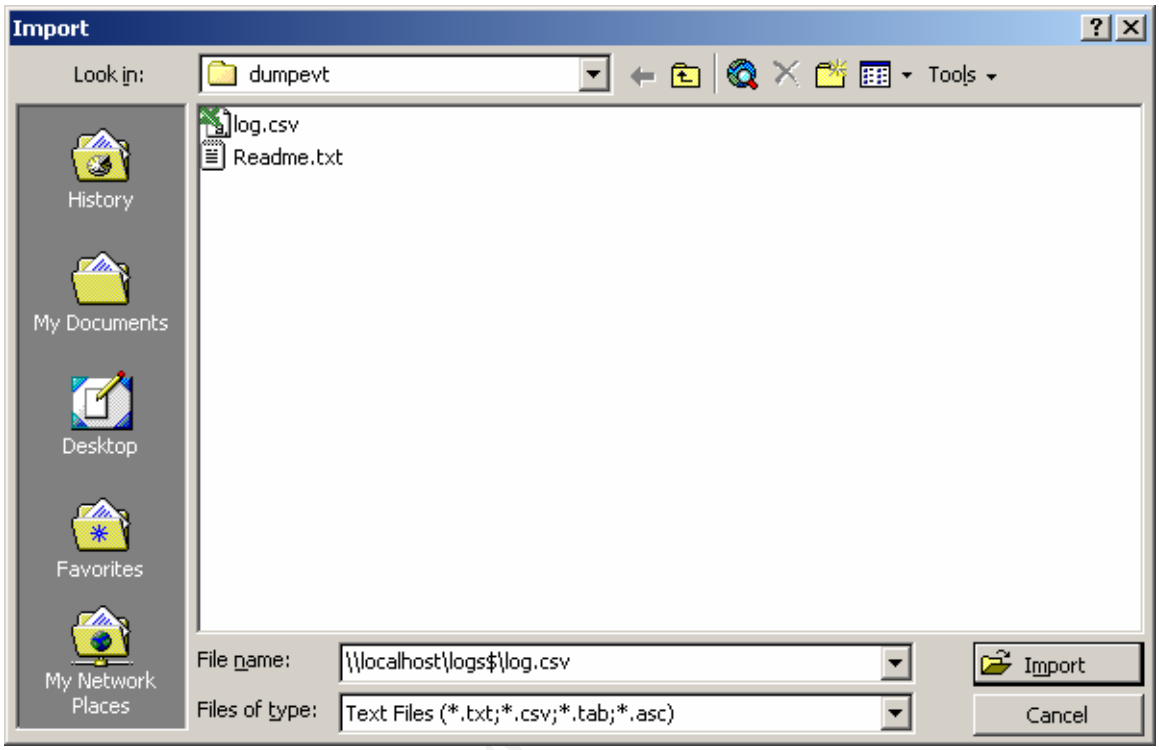
- Create the batch file to be executed (c:\dumpevt\daily.bat) as follows:

```
C:\DUMPEVT\DUMPEVT.EXE /LOGFILE=SEC /OUTFILE=C:\LOGS\LOG.CSV
    /REG=LOCAL_MACHINE
C:\DUMPEVT\DUMPEVT.EXE /LOGFILE=SYS /OUTFILE=C:\LOGS\LOG.CSV
    /REG=LOCAL_MACHINE
C:\DUMPEVT\DUMPEVT.EXE /LOGFILE=APP /OUTFILE=C:\LOGS\LOG.CSV
    /REG=LOCAL_MACHINE
```

On the collection server:

- Open the Microsoft Access database included with DUMPEVT
  - On initial open, Microsoft Access will offer to convert the database to Access 2000 format. Select "Convert" and use the new database.
- Connect to each servers' logs share \\servername\LOGS$

- Import the text file (LOG.CSV) from each server
  - Highlight the **LogRecs** table
  - Click **File**, **Get Data**, **Import**
  - Select **Text Files**



Result:

Once each server has the event collection routine in the scheduler, events will be appended to the log file. Each time the system administrator imports the text file to the existing database, he will be importing an entire collection of that server's events. Once all servers' log text files have been imported, the administrator will have a database of all events for all servers combined in one database, suitable for reports, custom queries, archival, or line-by-line analysis.

Good news/bad news:

The benefit of Distributed Batch Processing lies in its simplicity. The tools are all freeware, the batch files are all short and don't change from machine to machine. As long as the scheduler service is running, troubleshooting should be very simple.

The drawbacks are many and varied. Most notable is the requirement for NetBIOS connectivity. If any changes need to be made to the collection batch file, the change will need to be manually distributed to all machines being monitored. The collection always runs once each morning, so events may not be collected in as timely a manner as some administrators may require.

Conclusion:

The following chart summarizes how the Distributed Batch Processing solution holds up according to our established standards:

Category:	**Comments:**	**Score: 0-10**
Automation:	The process of storing the event logs to a text file is automated, but collection is still manual.	5
Non-NetBIOS:	This solution requires mapped drives, and therefore also requires NetBIOS. This might possibly be performed by other transfer mechanisms such as FTP.	2
Accountability:	The existing event log can track each administrator during each access to the event data if file auditing is enabled.	8
Central Code Base:	This code base is not centralized at all. If a change needs to be made in the collection procedure, each client needs to be changed individually.	0
Simplicity:	This process is relatively simple.	7
Timeliness:	This process collects logs only once per day, and is not very timely.	2
Security:	This procedure may be subject to any and all attacks involving NetBIOS communication and the Scheduler service.	3
Other: (-10 to +10)	This process is effectively free, and takes slightly less time than manually auditing individual logs.	(+2)
		29

Option #2: Centralized Batch Processing

Background:

Centralized Batch Processing is very similar to Distributed Batch Processing. It requires a single batch program running on a central server, which process the logs of all the monitored nodes remotely.

Collection tools:

As with the distributed model, either DUMPEL.EXE from the Resource Kit, or "DumpEvt" by Somarsoft are sufficient to perform this task. For the sake of consistency, DumpEvt will remain the tool of choice.

Scheduling tools:

Centralized scheduling will still be handled by the internal Windows NT Scheduler service, but only on the central collection server.

Software requirements:

- The collection server needs to have Somarsoft's DumpEvt installed.
- The collection server needs to have the "Scheduler" service running, with automatic startup enabled.
- The collection server needs to have an ODBC database installed. In this case, Microsoft Access 2000 will be present.
- Each administrative workstation needs Access 2000 as well.

Event Log Collection Process:

On each client:

- Ensure that the appropriate user account has rights to read the Security event log.

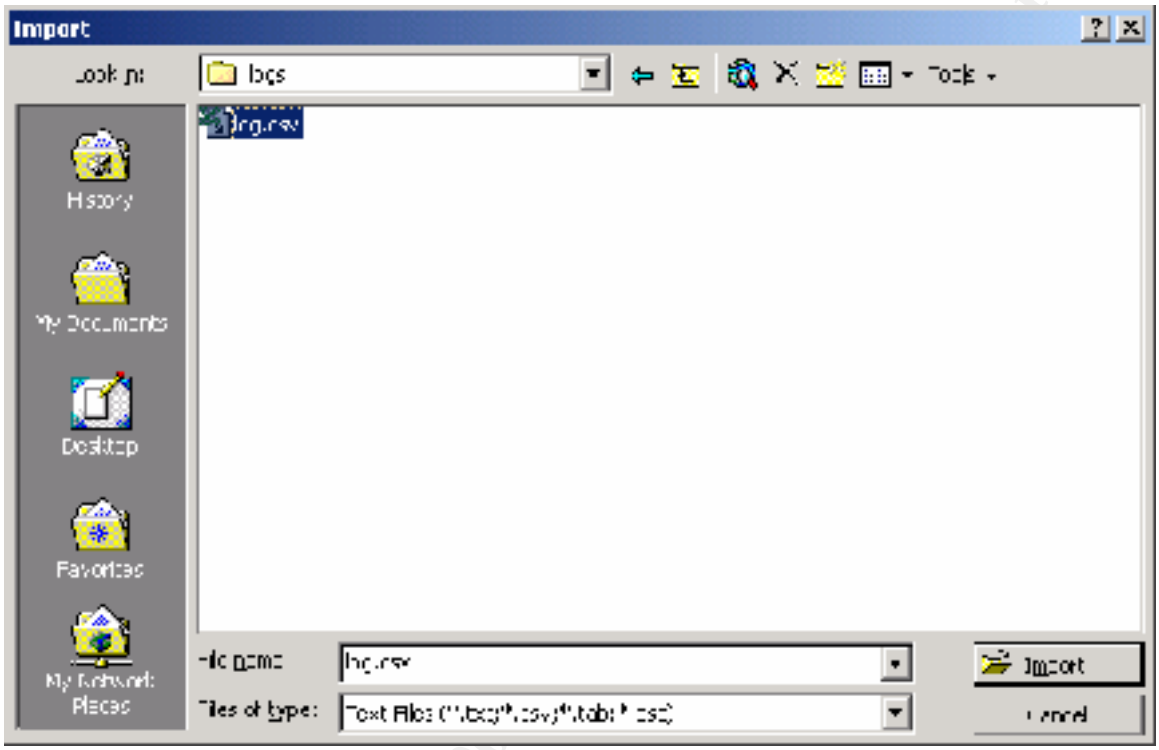
On the collection server:

- Install DumpEvt to a folder (C:\dumpevt)
- Create another folder for log files
- Set file level security to allow no more access than necessary.
- Create a batch file to be executed (c:\dumpevt\daily.bat)
- In the batch file, repeat the following commands for EACH machine to be monitored:

```
C:\DUMPEVT\DUMPEVT.EXE /COMPUTER=servername /LOGFILE=SEC
    /OUTFILE=C:\LOGS\LOG.CSV /REG=LOCAL_MACHINE
C:\DUMPEVT\DUMPEVT.EXE /COMPUTER=servername /LOGFILE=SYS
    /OUTFILE=C:\LOGS\LOG.CSV /REG=LOCAL_MACHINE
C:\DUMPEVT\DUMPEVT.EXE /COMPUTER=servername /LOGFILE=APP
    /OUTFILE=C:\LOGS\LOG.CSV /REG=LOCAL_MACHINE
```

- Execute C:\dumpevt\daily.bat from the command prompt. (Note: The account executing this command MUST have rights on all servers in question to read all event logs.

- Open the Microsoft Access database included with DUMPEVT
- Import the text file (c:\logs\log.csv)
    - Highlight the **LogRecs** table
    - Click **File**, **Get Data**, **Import**
    - Select **Text Files**



Result:

As the collection batch program processes each node, the appropriate event log information is appended to the text file LOG.CSV. Once that process is complete, the text file can be imported into the database as before.

Good news/bad news:

With the similarity between Centralized and Distributed Batch Processing, many of the strengths and weaknesses correspond as well.

The cost of the tools is non-existent. The process is manually started, so it is as timely as often as it is run. It maintains its accountability by running in a user context. In addition, this is running a central application, so changes to the code are made at one location.

Like its predecessor, it still requires NetBIOS connectivity. The centralized program gives more functionality, but more complexity as well. Also, it requires a user context with sufficient rights to read all Event Logs.

Conclusion:

The following chart summarizes how this solution holds up according to our established standards:

Category:	**Comments:**	**Score: 0-10**
Automation:	Automation is minimal once the process is started.	3
Non-NetBIOS:	Both DumpEvt and DUMPEL.EXE require NetBIOS communication to pull events from remote machines' Event Logs.	0
Accountability:	The event log can track each administrator connecting to that machine to pull event data, if auditing is enabled.	8
Central Code Base:	All processing is done by one machine. If a change needs to be made in the collection procedure, it is done in one location only.	10
Simplicity:	This process is still relatively simple.	3
Timeliness:	This process collects logs as often as required by the administrator, but is still a manual process.	4
Security:	Centralized Batch Processing requires NetBIOS. It also requires another user account to have remote access to its Event Log information.	2
Other: (-10 to +10)	This process is effectively free, and takes significantly less time than manually auditing logs.	+2
		32

Option #3: Centralized Windows Management Interface (WMI) Processing

Background:

Once Microsoft released Microsoft Windows NT Service Pack 4, along with the NT Resource Kit, Supplement 4, new tools emerged to manage Windows NT Servers and Workstations. The Desktop Management Task Force (DMTF), in coordination with major hardware and software vendors, developed a standardized method of sharing management information across a wide range of hardware and software platforms. This set of standards is referred to as Web Based Enterprise Management, or WBEM.

Microsoft's implementation of WBEM is the Windows Management Instrumentation (WMI). WMI is platform independent, native on Windows 98 and Windows 2000, and may be freely patched for Windows 95 and Windows NT 4.0 (SP4 or later). It is designed to provide a consistent management model, and to reduce the Total Cost of Ownership across the enterprise.

WMI provides easy integration with the Windows Scripting Host. This opens the capabilities of normally cryptic Application Programming Interfaces (APIs) to adaptable script languages like Jscript and VBScript.

The security model of WMI uses a process of Impersonation of a user's existing credentials. The existing user structure will sufficiently grant or deny access to the basic management functions available. This being said, WMI is a "young" technology, and will undoubtedly be subject to new security concerns at it becomes more widespread, and as it matures.

Collection tools:

The tools used to collect Event Log data from clients are integrated references to API calls of WMI, which are accessible from the Windows Scripting Host. The collection will be performed by a Visual Basic Script (VBScript), and written to the Access database by the same script.

Scheduling tools:

This script may be scheduled, or run as a service, but requires a user context, with administrative rights to read the event log of all machines to be monitored. For the purpose of this demonstration, the script will be started manually.

Software requirements:

On each client:

- Microsoft Windows Scripting Host (WSH) version 5.0 or later (located on the Microsoft Windows NT Server Resource Kit, Supplement 4 CD)
- Microsoft Windows Management Instrumentation (also located on the Microsoft Windows NT Server Resource Kit, Supplement 4 CD)

On each server:

- Required client software, listed above
- Microsoft Access 2000

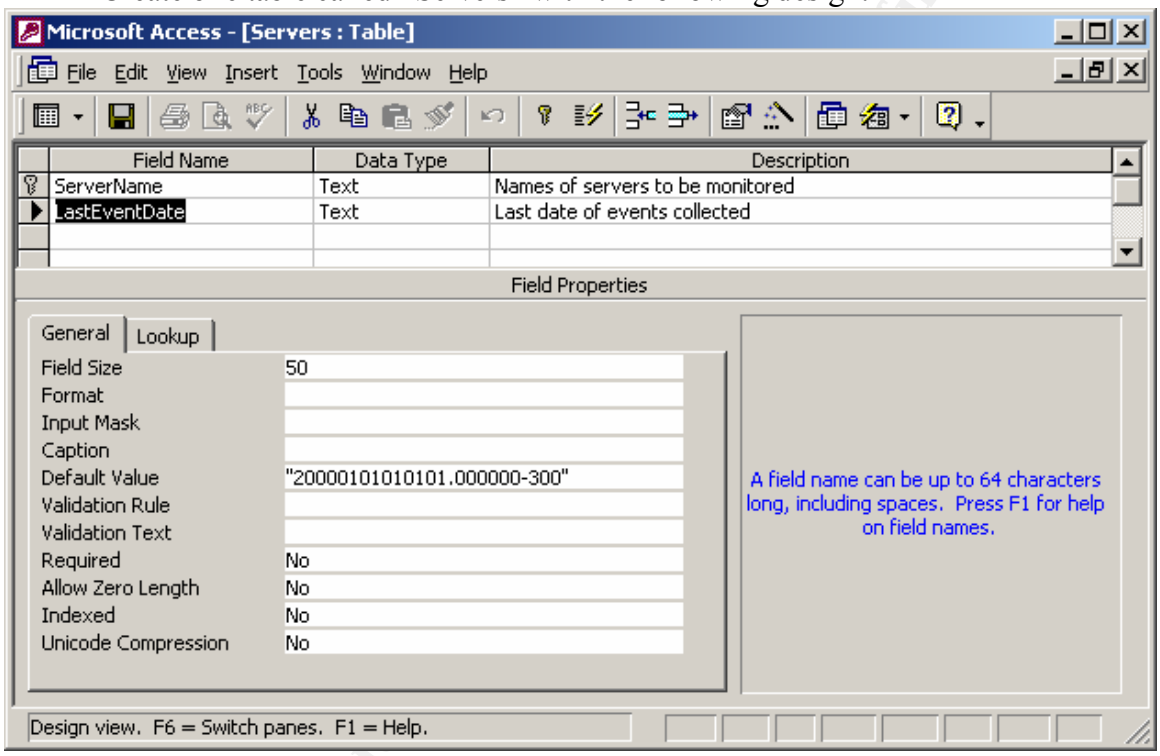
Event Log Collection Process:

On each client:

- Install required software packages.

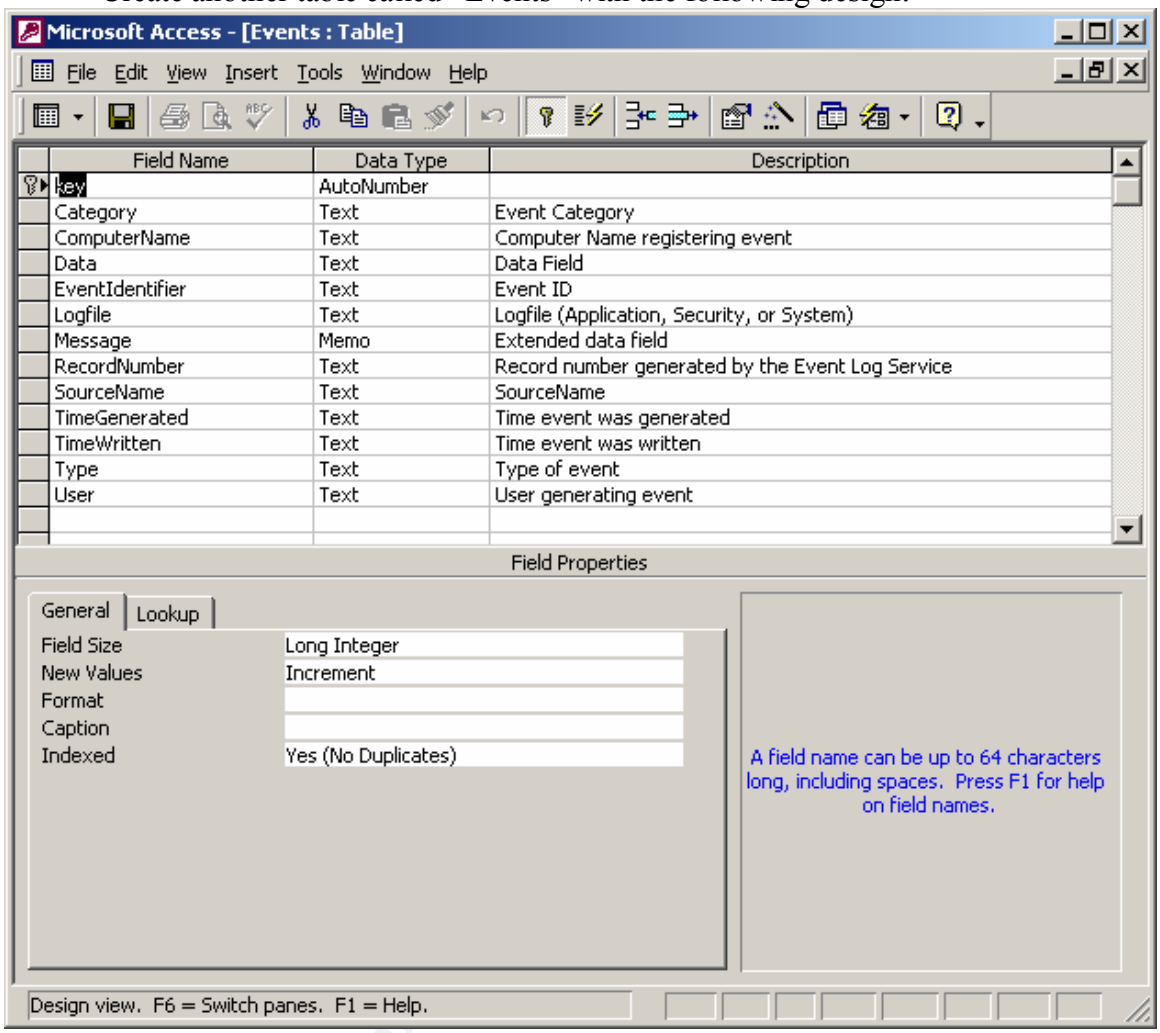
On the collection server:

- Install required software packages.
- Duplicate the script in Appendix B, saved as EVENTLOGGER.VBS
- Create a new Access database, saved a D:\EVENTS.MDB
- Create one table called "Servers" with the following design:



- Create one record in this table, with a servername of "Localhost", and accept the default date value above.

- Create another table called "Events" with the following design:



- Create one record, just so the database is not empty (this would cause the script to error on its first execution.)

- Execute the script from a command line as follows:

```
Cscript EventLogger.vbs
```

- The script will loop indefinitely.

Result:

This script requires software, which is already recommended for installation by the Microsoft Windows NT Server Resource Kit, Supplement 4. It then takes raw events, and writes them into an ODBC data repository, currently a Microsoft Access 2000 database.

Collection Explained:

Be advised that this script is a working example of how events from all NT Event Logs in multiple NT computers can be incorporated into a database. It has the potential to be augmented in countless ways. Other, more ambitious programmers are welcome to add to this script on their own. For the sake of brevity, this program is functional, but omits many of the "bells and whistles" which are certainly appealing in a production environment. This is a line-by-line analysis of the code in Appendix B.

The main program actually begins on Line 95. Error reporting is essentially disabled, and on Line 99, the program starts an infinite loop. Once the program starts, it will repeat indefinitely if allowed to do so.

The InitializeDatabase procedure (lines 10-26) establishes a connection to the "D:\Events.MDB" database. If the file location needs to be changed, or if a SQL Server or Oracle database is to be used instead, then this section would need to be changed.

The ReadServerNames procedure (lines 29-43) uses the connection to the database, and opens the "Servers" table. It reads this table data into the ServerNames and ServerDate arrays, and closes so that it does not remain locked.

The "Do Loop" which follows (lines 103-113) processes the computer names in reverse order. The first function performed on each server is to make sure there is a date from which to begin. If none exists, a date of January 1, 2000 is assigned.

The GetCompEvents procedure (lines 46-74) is the "meat and potatoes" of WMI event collection. It takes four parameters - Server Name, User Name, Password, and Last Time Written - all in string format. If the User Name and Password are left as null strings, the process of connecting to each computer (lines 49-55) substitutes the current user context to establish the connection. One noteworthy part of the connection process is the fact that, even if the user context in question has full rights on the remote (or local) machine, the WMI process must request the additional right of SeSecurityPrivilege (line 51) prior to the ConnectServer command (lines 53-54), in order for NT 4.0 computers to gain access to the Security Log Events.

The ExecQuery statement (line 56-57) actually pulls from each of the Application, Security, and System event logs, the events that have occurred since the last date recorded for each computer. This could be modified to only track the Security log, but events recorded in the Application and System logs may be important as well. This query may record duplicate records that occurred at the same second, but should remain all-inclusive of events.

The procedure then loops through each of the new events. For each new event, it creates a new record (line 61), adds the appropriate piece of data to each field (line 62-66), keeps track of the most recent "Time Written" from the events, updates the record (line 72), and moves on to the next record.

The data is transferred from a field in the event log, to a matching field in the Events database (line 64). In the case of the "EventIdentifier" field, an error is generated indicating "Type Mismatch," but the data is still transferred. Line 65 clears that error, and allows the script to continue.

Once all the events for each computer are recorded, the "Servers" database is updated with the "Time Written" which is most current for that computer (line 77-92).

Good news/bad news:

The good news is that the WMI solution, which is utilized through Visual Basic Script language, and Windows Scripting Host, fulfills all the operational requirements identified for collecting events. It maintains accountability by recording its own access to the Event Logs. It maintains a central code base by processing all code on a single machine. It can be executed in a timely manner. The interval can be changed as needed. Security is maintained, as access is only granted to those who have access on the monitored computer itself. And, it is a free download, or obtainable as part of the Windows NT Server Resource Kit, Supplement 4.

The bad news is that this is very complex, in comparison to the other solutions presented. It requires manipulation of script to be executed on production networks, in privileged user context. It is also a young technology, which is likely to be the target of future hacking and/or exploitation. As Microsoft releases patches, they will need to be applied to networked Windows NT 4.0 machines.

Conclusion:

The following chart summarizes how this solution holds up according to our established standards:

Category:	**Comments:**	**Score: 0-10**
Automation:	This process starts manually, then runs fully automated.	9
Non-NetBIOS:	This process uses COM communication, with little or no dependence on NetBIOS. Port 135 is used to negotiate a port above 1024 for communication, but ports 137 through 139 are not used at all.	9
Accountability:	WMI processing of remote machines is tracked in the Security log by a remote logon and logoff.	10
Central Code Base:	All processing is done by one machine. If a change needs to be made in the collection procedure, it is done in one location only.	10
Simplicity:	In comparison to other methods, WMI scripting is quite complex.	2
Timeliness:	The script has a user-definable delay between imports, currently a 5 minute wait between cycles. It is quite timely.	8
Security:	The security of WMI access is impersonated from the level of NT access, and is as secure as NT.	9
Other: (-10 to +10)	This process is effectively free, and makes use of new technology, which is portable to other similar operating systems. (+9) It is also a young technology, which is bound to become the target of new attacks in the future. (-4)	+5
		62

Scorecard:

	Distributed Batch Processing	Centralized Batch Processing	Centralized WMI Processing
Automation:	5	3	9
Non-NetBIOS:	2	0	9
Accountability:	8	8	10
Central Code Base:	0	10	10
Simplicity:	7	3	2
Timeliness:	2	4	8
Security:	3	2	9
Other:	2	2	5
	29	32	62

According to the standards established early in this document, Centralized WMI Processing is the clear winner. This indicates that it does the job at hand quite well, but does not imply that it is appropriate for every environment. Its major drawback – complexity – is significant. The other options may be more appealing in environments that just do not have the capacity to deal with complex scripting in a 'support' role.

In any case, several inexpensive options exist for centralizing Microsoft Windows NT 4.0 event logs to a database. Once inside that database, administrators can write queries to filter failed logons, file access auditing, or any other specific events. Queries can be written to exclude the "known" events, and look at what is left. Once all events are pooled together, the Microsoft Access 2000 wizard can be used to customize queries to fit any environment.

Appendix A – Software Requirements and availability:

Somarsoft's DumpEvt – available at http://www.somarsoft.com as freeware.

Microsoft Windows NT 4.0 Server Resource Kit utility DumpEL.Exe – Available as part of the Resource Kit, or downloadable from Microsoft at ftp://ftp.microsoft.com/bussys/winnt/winnt-public/reskit/nt40/i386/dumpel1.exe

Microsoft Windows Scripting Host v5.0 – Distributed with the Microsoft Windows NT 4.0 Server Resource Kit, Supplement 4.

Microsoft Windows Management Instrumentation (WMI) – Distributed with the Microsoft Windows NT 4.0 Server Resource Kit, Supplement 4.

‘Appendix B – VBScript (EventLogger.vbs) used in Option #3

```
option explicit

dim cnnDataBase, rsServerList, rsEventTable
dim ServerNames(100), ServerDate(100)
dim count


private sub InitializeDatabase
    dim strDBName, strConnect

    strDBName = "D:\Events.MDB"
    strConnect = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
strDBName

    set cnnDataBase = CreateObject("ADODB.Connection")
    cnnDatabase.Open strConnect

    set rsEventTable = CreateObject("ADODB.Recordset")
    rsEventTable.ActiveConnection = cnnDatabase
    rsEventTable.CursorType = 1 'adOpenKeyset
    rsEventTable.LockType = 3 'adLockOptimistic
    rsEventTable.Source = "Events"
    rsEventTable.Open
end sub


private sub ReadServerNames
    dim strQuery

    strQuery = "SELECT * FROM Servers"
    set rsServerList = cnnDataBase.Execute(strQuery)
    count = 0

    do while not rsServerList.EOF
        ServerNames(count) = rsServerList("ServerName")
        ServerDate(count) = rsServerList("LastEventDate")
        rsServerList.MoveNext
        count = count + 1
    Loop
    rsServerList.close
end sub


private sub GetCompEvents(strSrv, strUsr, strPwd, strTW)
    dim svrlocate, connct, collectn, thing, propty, tmpname

'    Establish Connection to remote computer
    set svrlocate = CreateObject ("WbemScripting.SwbemLocator")
    svrlocate.Security_.Privileges.Add 7 ' SeSecurityPrivilege

    set connct = svrlocate.ConnectServer ([strSrv],, [strUsr],
[strPwd])
    connct.Security_.ImpersonationLevel = 3 'Impersonate
```

```
56      set collectn = connct.ExecQuery("SELECT * FROM Win32_NTLogEvent
57  WHERE TimeWritten >= '" & strTW & "'")
58
59      for each thing in collectn
60          rsEventTable.MoveFirst
61          rsEventTable.AddNew
62          for each propty in thing.properties_
63              tmpname = propty.name
64              rsEventTable(tmpname) = propty.value
65              err.clear
66          next
67
68          if rsEventTable("TimeWritten") > ServerDate(count) then
69              ServerDate(count) = rsEventTable("TimeWritten")
70          end if
71
72          rsEventTable.Update
73      next
74  end sub
75
76
77  private sub ReWriteDate (SrvName, SrvDate)
78      dim rsWriteServers
79
80      set rsWriteServers = CreateObject("ADODB.Recordset")
81      rsWriteServers.ActiveConnection = cnnDatabase
82      rsWriteServers.CursorType = 1 'adOpenKeyset
83      rsWriteServers.LockType = 3 'adLockOptimistic
84      rsWriteServers.Source = "Servers"
85      rsWriteServers.Open
86
87      rsWriteServers.MoveFirst
88      rsWriteServers.Find "ServerName = '" & srvName & "'"
89      rsWriteServers("LastEventDate") = srvDate
90      rsWriteServers.Update
91      rsWriteServers.Close
92  end sub
93
94
95  'Begin Main Program
96
97  on error resume next
98
99  Do while 1=1
100     InitializeDatabase
101     ReadServernames
102
103 '    Process Servers
104     do until count = 0
105         count = count - 1
106         if Serverdate(count) = "" then
107             Serverdate(count) = "20000101010101.000000-300"
108         end if
109         GetCompEvents ServerNames(count), "", "", ServerDate(count)
110
111 '        Write new serverdate
```

```
112             ReWriteDate ServerNames(count), ServerDate(count)
113         loop
114
115     '    close output table
116         cnnDataBase.close
117
118         wscript.sleep 300000
119     loop
120
121     ' End of Program
```

Appendix C: References

Heckendorn, Sherry. GCNT Practical Assignment. The SANS Institute.
http://www.sans.org/y2k/practical/Sherri_Heckendorn.doc

Gabert, Howard. GCNT Practical Assignment. The Sans Institute.
http://www.sans.org/y2k/practical/Howard_Gabert.doc

Somarsoft DumpEvt v1.7.3 Help File. Somarsoft DumpEvt. DumpEvt.HLP.

"Windows Management Instrumentation: Background and Overview." Microsoft Technet. Microsoft Corporation.
http://www.microsoft.com/TechNet/winnt/winntas/prodfact/wmiovw.asp

"Microsoft Windows Management Instrumentation Scripting." Microsoft Developer Network. Microsoft Corporation.
http://msdn.microsoft.com/library/backgrnd/html/wmiscript.htm

"Administering Windows through Windows Script Host." Microsoft Developer Network. Microsoft Corporation.
http://msdn.microsoft.com/library/periodic/period99/adminwsh.htm

Eck, Thomas. *Windows NT/2000, ADSI Scripting for System Administration*. Indianapolis, IN: MacMillan Technical Publishing, March 2000.