Windows NT/2000 Event Logs

William Mendez

Windows NT/2000 Event Logs

William Mendez

4-21-02

Abstract

Daily tasks of every security or network administrator involve reviewing the content of log files for suspicious entries indicating that a potential attack has occurred, or is in the process of occurring. Working with logs is a very time consuming and cumbersome task. Several servers generate log files every day with hundreds of entries to search through, but the network administrator seldom has enough time to properly review and interpret them.

The goal of this paper is to address this issue by automating the process of gathering and filtering log files, and notifying the administrator when relevant events are found; while maintaining a very low implementation cost. It's also important to mention that this paper will only focus on a Microsoft based environment running Windows NT 4.0 and Windows 2000, or either one independently, and on a single log type, the "Security Log".

This paper will help one to completely automate the process of gathering, filtering and alerting when relevant events are found using inexpensive tools and resources already available. The goal is to prevent potential attacks or misusage by making it easy and cost effective to gather and review event logs. It is not an attempt to teach one how to read the logs, nor what each type of event represents.

Introduction

This paper was written to provide a very simple way to manipulate security event logs in Windows NT/2000. It is meant to be an example of using Windows Native Tools, Built-in Tools, and freeware tools to reduce the amount of work required when dealing with security event logs, as well as provide a fully functional example that may be used as-is or modified to suit specific needs.

Assuming that we have a well balanced Windows NT/2000 audit policy, we are left with a lot of information to read and process, to later decide what actions to take based on the severity of the findings.

Working with the logs on Windows NT/2000 is not an exciting journey, especially if we have to read them directly from the System, Security or Application log. Additionally, when we have to look into multiple servers or even workstations, it makes it tedious and slow. It's true that we can filter by many options right from the Event Viewer, but it's still a manual and time consuming task.

Before attending a SANS seminar for Security Essentials, I was trying to collect the information from the security logs on each of my Domain Controllers and some File and Print Servers. Essentially I was doing it the hard way, saving and clearing the logs on each server then copying the saved logs into another server on the network. After this was done, reading one by one was the next step. Of course the first log wasn't that bad, after the fourth I was discouraged to continue with this long boring task. I also noticed that the time needed to fill up the log and the size of the log, will need to be adjusted to accommodate, more than 38,000 events registered by only one of the servers in a period of 3 days.

I began to research for tools to accomplish this task that would have a minimum cost and to make this a more enjoyable job. One of my first findings was a tool on the Windows 2000 Server Resource Kit, CyberSafe Log Analyst. "CyberSafe Log Analyst (CLA) is a snap-in to the Microsoft Management Console (MMC). It is a Windows 2000 Security Event Log analysis tool that uses predefined analysis techniques. The CyberSafe Log Analyst assists you in organizing and interpreting security event logs from Windows 2000, providing more effective, system-wide user activity analysis." (as described in the help file of CLA).There was one more task pending, in which I still needed to go to each server, clear and save the log, move the log to another location, and import the log into CLA.

A few days after trying this tool I attended the SANS Security Essentials seminar and one of the topics was "SANS Security Essentials V Windows Basics", book 1.5 from the courseware materials. This book covered working with NT/2000 security logs and talks about tools and possible ways to manipulate the logs to get more out of them. It was then when I was able to clearly see my scenario and possible solutions to it.

Scenario

All Microsoft Windows NT 4.0 domains will have a Primary Domain Controller or PDC and any number of Backup Domain Controllers or BDC(s). While this is needed to create the domain (PDC) and to provide redundancy and load balancing (BDC), these servers can provide a great deal of information about what is happening in its domain. Information that can go from registering newly created accounts, groups, computer-accounts to recording latest events on existing accounts like changes on group membership or attempts to access resources, or just logons. These events are grouped into two major categories, Failure and Success. For example: A user that mistypes his password will register a failure logon event under his account. An authorized user that creates or removes an object generates a successful event type, but all this information can only be gathered if auditing is enabled.

Why would we want to collect such information if a user just mistypes a password, or if an authorized user creates or removes an object? Let's look at it from a different perspective, what if the user that mistyped his password was not the legitimate owner of the account, instead someone else executing a dictionary attack trying to guess this account password. What if the user that is authorized to perform a certain task is not the real user that is entitled to use these privileges, but instead is someone else using that user's account to create new accounts or grant privileges to other accounts that should not have them.

With this in mind, we can see that logging can be our partner when tracking actions or events that happen in a domain. It could also help in forensic investigations that sometimes go to court. These are the reasons why auditing.

In order to enable auditing, we first have to identify the servers that we want to audit. We are going to be on a single Windows NT 4.0 domain, with the required Primary Domain Controller (PDC) and 7 other Backup Domain Controllers (BDC). In addition to this we find storage, print, database, web, application, mail, ras and some especially designated servers for commercial solutions deployed in our environment, over one thousand users and computer accounts respectively are also listed. In our case, consider that all domain controllers will be auditing. In addition, any other server will be enabled as needed.

Next we are going to examine the original scenario that I started working with when I decided to use CyberSafe Log Analyst (CLA) from the Microsoft resource kit, refer to Fig1.1.

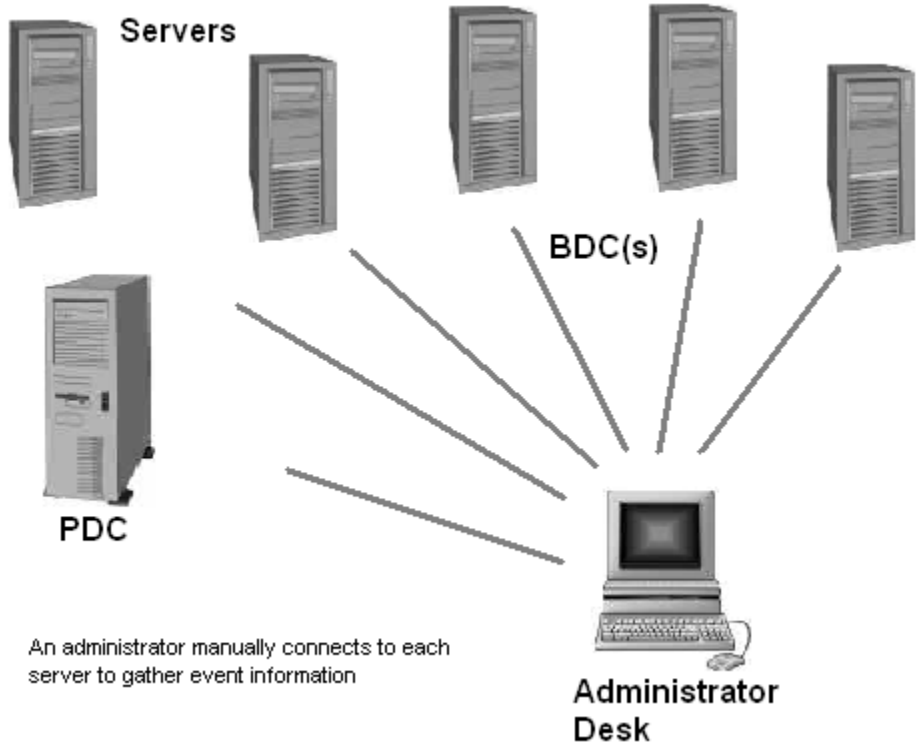


Fig1.1

First, I connected from the administrator console to one of the servers, open the security log and saved it into that server's hard drive. As I save it, I also cleared to avoid reading the same events next time around. Secondly, I

connected to the admin share for the hard drive on the server in which I had just saved the log, copied them to my terminal or PC, and started CLA to import the log file. Next, I connected to another server and repeated the process until the last server was reached. The use of CLA helped me in filtering and generating professional looking reports, and cutting the time and effort needed to filter and read from the event viewer interface. Having to go server by server, saving, moving, and filtering log files to produce a report was not very efficient. Finally, imagine that to complete the process each server was requiring about 15 minutes, times seven servers, resulting in almost two hours of doing this every day!

Considering the amount of time required everyday to complete this task, we concluded that it was not efficient to continue with this procedure.

Proposed Solution

A proposed solution is to completely automate the process by using inexpensive tools and resources already available. Provide a simple, effective and flexible solution that will allow possible future modifications requiring minimal knowledge or specific skills.

To start the new phase, a couple of things that will be require for it to succeed. One of them is an additional component, a standard PC, used to schedule jobs, should never be turned off or unplugged from the network. To centralize the scripts and facilitate any updates, having only one common location as storage space somewhere in a network server to save the log files and generated reports. Another element will be the account executing the scheduled tasks, it has to be a domain administrator account, or an account with sufficient rights to access and modify the security logs on each designated server. To send the emails an SMTP server is required, and a generic mailbox or email alias to be used as sender of the reports is also recommended.

Operating system requirements are minimum, the scripts will run on any Windows NT 4.0 or Windows 2000 system, and the target server could be running either operating system as well. It will also work on regular Workstations or Professional as targets too. I have not tested this running from or against a Windows XP version.

In order to organize the files that form this project, a layout of the directory structure is shown in appendix A. There will be a server called NETSERVER that will have most of the scripts and will serve as a working space. Log files will be transferred to this server to be filtered, and report files will be saved locally on this server and sent out on emails. The server will also hold a copy of the original log file until the logs are backed up. Support files are also placed at the server.

This is an inventory of the files needed at the server “NETSERVER” as well as in our scheduler station “WKST00”, they have been grouped by type and their role has also been identified.

NETSERVER

TEXT FILES

1. msg.txt, its content will be used as the body of the email message.
2. dcnames.txt, will contain a list of online domain controllers. It’s dynamically created and updated each time it runs.
3. serverlist.txt will be used to add non domain controllers to the list of server to pull the logs from.
4. eventlist.txt, will have a list of all event type that we want to extract from the logs.
5. empty.txt, is created to announce that mailing is completed and files can be removed.
6. today.txt, will contain the last date the scripts where executed, including date and time. This will be use to identify the log files since they are saved using the server name and this value.

EXECUTABLE FILES

1. dumpevt.exe, program to dump/convert the event log into text format. http://www.systemtools.com/somarsoft/
2. elsave.exe, console app to save and/or clear a Windows NT event log. http://www.ibt.ku.dk/jesper/ELSave/default.htm
3. erlist.exe, console app to delete files with zero size from a given directory and generate a txt file with a list of remaining files. Source code ready to compile with Delphi 6.0 on Appendix B
4. findstr.exe, searches for strings in files. It is a windows native tool. Used to extract specific events from the text version of the log files.

5. logevent.exe, log a user event to EventLog registry. Used to add an event on the application log to correlate with security events generating while manipulating the security log. http://www.microsoft.com/windows/reskits/default.asp
6. netviewx.exe, tool to list the servers in a domain or workgroup. Used to find domain controller and populate dcnames.txt. http://www.ibt.ku.dk/jesper/NetViewX/
7. vdate.exe, Displays the current date and time. The output can be tailored using a number of formatting options. http://david.tribble.com/programs.htm

SCRIPTS

1. archive.cmd,
2. dumplog.cmd,
3. evtotxt.cmd,
4. getdate.cmd,
5. getdcnames.cmd,
6. movelog.cmd,
7. scanlog.cmd,
8. mkarchive.cmd,
9. mkdumplog.cmd,
10. mkevtotxt.cmd,
11. mkmovelog.cmd,

WKST00

Also a fewer group of files are needed in:

TEXT FILES

1. rcpt.txt, a text file with the list of all recipients.

EXECUTABLE FILES

1. blat.exe, utility that sends the contents of a file in an e-mail message using the SMTP protocol.http://www.interlog.com/~tcharron/blat194.zip

SCRITPS

1. mkwf.cmd,
2. sendmail.cmd,
3. empty.cmd,

The process will be initiated from the scheduler station, WKST00. (See Fig2.1)

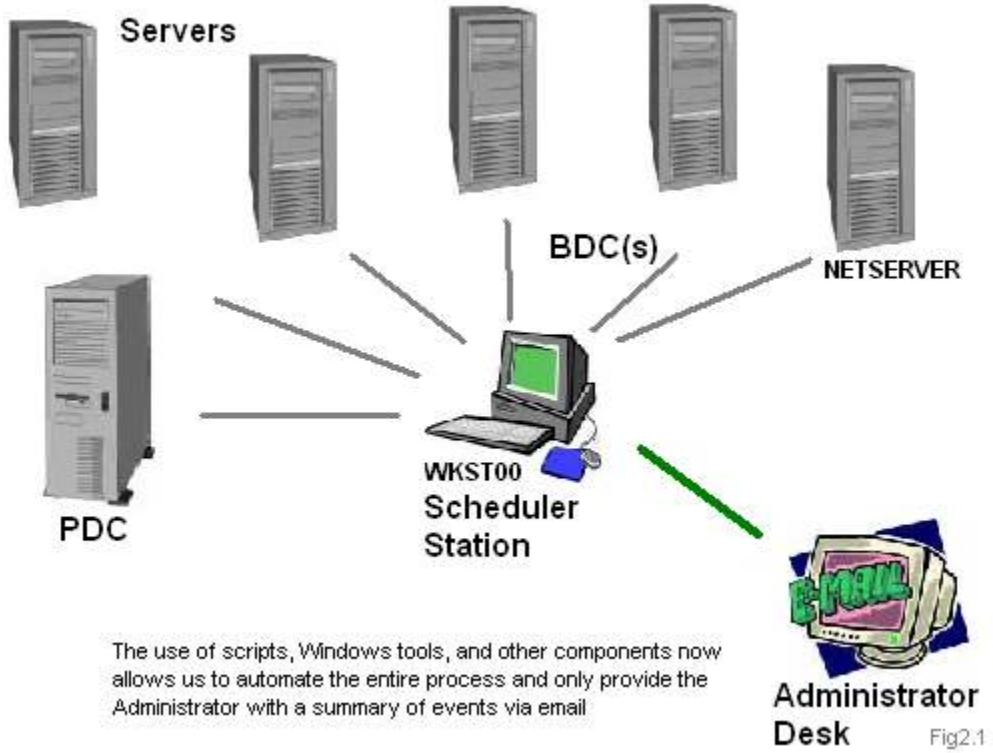


Fig2.1

Some preparation is required before we begin executing any script. First, the email client we will use is BLAT by P.Mendes, M.Neal, G.Vollant, T. Charron http://www.blat.net and it needs to be locally registered by executing it from the command prompt. A command looks like this:
C:\Blat -install smtp.mydomain.com eventchecker@mydomain.com 3
Where smtp.mydomain.com will be your smtp server, and eventchecker is the alias for the mailbox created to send the messages.

Here are some pre-steps to look at before we execute the first script.

- Review the desired scenario.
- Know the requirements.
- Set auditing in all servers of interest.
- Accommodate enough space for the log files on the servers so events are not lost.
- Place the files in their appropriate location.
- Register or install blat in our scheduler station.
- Select the account that is going to be used to schedule the execution of the scripts.
- Allocated space in “NETSERVER” to save original logs until they are backed up.
- Finally make sure to test it before running it on your environment.

Summary of the process

The entire process will begin to execute on WKST00. The first script will create a list of existing domain controllers and other servers added in the file serverlist.txt. The list of servers to work with will be in a file named dcnames.txt. After dcnames.txt is generated the information collected in it will be used to create a set of new scripts that will be server specific. The next step to complete is to connect to each server and save the log, clear it and move it to the NETSERVER. Additionally, an event will be written to the application log registering the actions performed on the security log. This is a way to identify when a change done by this script to the security log occurs. Once the log files are in NETSERVER, they are still in windows native format, therefore we will generate a text version of each log, scan these new text files for any event type and construct a report file per server. Next, the report files will be sent to the recipients on rcpt.txt and all files will be erased once the process is completed. With the exception of the original log files that will be moved into another directory for later backup and only when they are backup they will be deleted. At this point we have completed the process for the first time, the administrator has received the reports needed without too much effort, and the server is ready for the next run time. Please refer to Fig2.1.

Detailed overview of the process

In order to have a clear understanding of what each component is doing I have divided each component into main areas based on the action performed by one or more scripts and any tools as well.
Main areas by action:

- Initialization
- Save
- Clear
- Move
- Filter
- Notify
- Clean-up

INITIALIZATION

During initialization all files are copied to their respective location as described in the above "Proposed Solution". Any additional non-domain controller server will be added to serverlist.txt. The schedule and frequency for the scripts to be executed will be also set on WKST00. This is done on the Control Panel under "Schedule Task". Locally on WKST00 we will have three scripts, the first one to be scheduled is mkwf.cmd, the second one is sendmail.cmd, and the last one empty.cmd.

Within the scripts there are sub modules called steps. The steps relate directly with one or more areas as described in the "Detailed overview of the process".
mkwf.cmd, also list a small portion called "support tasks"
mkwf.cmd, will list steps 1, 2 and 3, and it relates to SAVE, CLEAR, MOVE and FILTER
sendmail.cmd, will list step 4, and will relate to NOTIFY
empty.cmd, will list step 5, that relates to CLEAN-UP

The first script that gets executed is mkwf.cmd, (See Appendix D, FILE 1) the first lines connect to the destination server "NETSERVER" and move the current working directory to the newly map network drive. This allows maintain most script files at a centralized location. The first sub module is called "support tasks". (See Appendix E, Support Task) This sub module is responsible for combining the information form dcnames.txt and serverlist.txt into a new version of dcnames.txt. This is accomplished by calling getdcnames.cmd, a script that uses NETVIEWX.EXE to enumerate all currently available domain controllers, and issuing a type command to append the content of one file into the other. (See Appendix E, Support Task)
During this module two new scripts are created, dumplog.cmd and movelog.cmd, when calling mkdumplog.cmd and mkmovelog.cmd respectively. (See Appendix D, FILE 7 and FILE 9)

The first script, mkdumplog.cmd, uses a "for" statement to read the content of dcnames.txt. Each server name listed in this file will be used to identify the server from which we will collect the log files. A command line to execute elsave.exe to save and clear the log will be issued per server. Notice that log files are saved locally on each server and are also named after the server. Lastly, it will write and event into that server's application log registering the actions just performed using logevent.exe. The above-mentioned steps are implemented in dumplog.cmd. (See Appendix D, FILE7 and FILE8)

The second script, movelog.cmd, uses native windows commands to move the log files from each server listed on dcnames.txt. Since the name of file is the same as the server, by reading dcnames.txt we know the server and the file to move to NETSERVER into DCLogs\SecLogs folder. Next, we will connect or map a network drive using a net command to access the log file on the first server listed on dcnames.txt. It's important to notice that the letter "l" will be used in this script and any existing connection using that letter will be lost. Once the connection is established with the source server, we issue the move command, and to avoid disconnecting before the move is completed, a loop to check for the presence of the file in the source is included. After the file has been moved we disconnect from that server and re-use the same letter to map the drive of the next server listed in dcnames.txt until the last entry is reached. Also an entry is written to the event log on each server as it finishes moving the log file. The above-mentioned steps are implemented in movelog.cmd. (See Appendix D, FILE9 and FILE10)

SAVE & CLEAR

These two actions, saving and clearing the log, are combined into dumplog.cmd, which are also reflected in step 1. (See Appendix D, FILE 8) This script is going to connect to each server and using elsave.exe thru switches like -C to clear the log, and -F to pass the name of the file in which we want to save the log, the two actions are combined in a single line of commands. Notice that the name of the file is the same as the name of the server. Since most of the tasks are based on information provided by dcnames.txt which contain all server names, that's the reason why we save the files using the same information, so we can always identify from which server the log is coming from. The -l switch is to specify that the security log is the one we are using. After clearing the log and saving it, we then register an event on the application log stating that we just saved and cleared the security log, so we can verify any changes into the security log done by this script. The tool used for this particular task is logevent.exe from the Windows 2000 Resource Kit. This process is repeated for each server. (See Appendix E, Step1)

MOVE

Moving the log files from their source server to the destination server is done using movelog.cmd, (See Appendix D, FILE 10). This is a combination of basic Windows native tools that connect to the remote server, access the hard drive in which the log file where saved, and move it to the destination server NETSERVER. To be able to successfully connect to the source server, the script first will disconnect any existing mapping using the letter "l" which is been assigned to map the drive for the source server. The destination is already set from mkwf.cmd when the process starts. Next, we issue a move command to transfer the file from the source to the destination. Since some of these logs can be very large, we added a loop, ":check IF EXIST l:\FILESERVER call :check", this will check for the existence of the file in the source before we continue and disconnect from that source to reuse the letter "l" for the next server. Also an event is issued to the application log, again to specify the log was just moved out of this server by this script. (See Appendix E, Step1)

FILTER

The Filter area expands over two steps. It starts at step 2 executing mkevtotxt.cmd, a script that will generate another script. (See Appendix D, FILE 11 and Appendix E, Step 2) This script depends on the information collected in dcnames.txt, to find the names of the log files previously saved using the server name as the file name. For every file that is listed on dcnames and exist in folder DCLogs\SecLogs on the NETSERVER it will generate the corresponding line into evtotxt.cmd to convert an original log file into a text version and save it with the same name into a different directory, DCLogs\Text.

The second call in this sub module is to evtotxt.cmd, (See Appendix D, FILE 12). This script will contain specific lines as to what files located in the Seclogs folder need to be converted in to text and placed into Text folder using dumpevt.exe, http://www.systemtools.com/somarsoft/. The syntax for this particular tool is very straight forward, it only needs the source file to identify what type of log it is, Application, System or Security, by using the switch "/logfile=sec=path", notice "sec" identifies the "Security Log" and the destination file to contain the text version of this security log file. After completion of this script a new set of text files containing all events from the gathered security logs resides on NETSERVER\DCLogs\Text.

One of the goals of this project is to keep original log files for as long as needed. Every new task will be done using the text version of the log files. The next call is to mkarchive.cmd, a script that based in the content of dcnames.txt creates another script, archive.cmd, with specific instructions as of what files to move and how to identify them. Since the intent of this new generated script is to move the original log files for backup into another directory and identify them per line existing in dcnames.txt, the current date and time will be added to the name of each file that can be found at SecLogs folder. (See Appendix D, FILE 13 and Appendix E, Step 2)

To identify the files by adding the current date and time, a call to another script is made, getdate.cmd. (See Appendix D, FILE 5) This script uses a freeware tool, vdate.exe, to extract this information from the system and then the output of it is saved to file. This tool can be found at http://david.tribble.com/programs.htm. The new file containing the date and time is called today.txt and is placed at DCLogs folder. Its content will be added to the original name of the log file which is also the name of the server from which the log was originally gathered. Coordination with the backup administrator is required to backup the folder named "Backup" on NETSERVER\DCLogs\SecLogs and depending on how the backup of the logs should be maintained and scheduled, it will be arranged.

The next call within this sub module is to archive.cmd, (See Appendix D, FILE 14).Upon completion, the files should be moved to the "Backup" folder and renamed to reflect the server from where it was collected and the time in which it was moved from backup. These files will remain in this directory until they get backed up, and the CLEAN-UP process takes place. This ends step 2.

Lastly, the process of extracting information from the text version of the files is done in Step 3 by a script called scanlog.cmd, (See Appendix D, FILE 6 and Appendix E, Step 3) In this particular sub module where the filtering process takes place, it also creates a list of reports generated that later will be mailed out to the specified recipients on rcpt.txt file, and the reports files based on the event type requested on the evenlist.txt file.

To start it will delete any existing version of the report list, called list2mail.txt. After it verifies the existence of dcnames.txt and if not available, it will call to create a new list of servers. Based on the list of servers provided on

dcnames.txt it will check if a text version of a log file exist and if it does, it will call a sub routine within scanlog.cmd referred as a label ":findevt" to which it will pass the name of the text file to be scanned as a parameter. This process is repeated for every file that exists on DCLogs\Text.

Each time a call is made to ":findevt", it will open eventlist.txt and will scan the text version of the log file issuing a "find.exe" command with the first event number listed on eventlist.txt and the name of the file previously passed to it. The output of this "find" statement will then be written to a report file on the Outbox directory using a new name scheme. This time the file name will be formed using the original server name and the number of the event that it was scanned for. This means that if we have one text file and ten event numbers or event types, it will go ten times into the file searching for each event at the time. The outcome of this task will be a report file per event, per server, which are plain text files.

During this last process, many blank files or empty files are created if the events are not found in the text version of the log file and also their names are registered to the report list file or file2mail.txt. Since they are just empty files there is no reason to send emails with empty attachments. A new task is required to delete all empty files or files that show zero bytes on their size, and update the report list or list2mail.txt file.

The final call during this sub module is to an executable erlist.exe (See Appendix C) for the source code of it. This file was put together using Delphi 6.0 professional and its only purpose is to go in the outbox directory and delete every file it finds that matches zero for its size. Once all files are removed, it will recreate a new version of file2mail.txt with the remaining report files. Feel free to use your tools or to modify this one to better serve your requirements. This was created because I couldn't find anything that provided the needed results.

NOTIFY

This area is done using sendmail.cmd, (See Appendix D, FILE 2 and Appendix E, Step 4) In this module all reports generated are going to be mailed out to the recipients listed in rcpt.txt file. Beginning with checking if the list of reports "file2mail.txt" exists to continue or issue and email stating that no mail is ready at this time. In case that after running the reports no events are found and all report files are empty, they will be deleted and the report list will never get created, when this module runs and finds that list2mail.txt is missing it assumes that no explicit events where found so it will send an email to relay that message and exit. On the other hand, when there is mail to send it will forward the control to a sub module within sendmail.cmd, called ":mailgo". This sub module will read the file names from file2mail.txt and send it one by one to each recipient found on rcpt.txt, using msg.txt as the body of the message. Once the emailing process is done a file will be written to NETSERVER\DCLogs named "empty.txt" that is used as a flag to tell the next process that all files can be deleted.

CLEAN-UP

The last process is clean-up, called empty.cmd (See Appendix D, FILE 3). Once all reports have been sent they are no longer needed and will be erased. Also, files that have been backed up from the "Backup" directory will be removed as well. Deleting files could be very destructive if done at the wrong time or to the wrong files. The element to control when files can be deleted is "empty.txt" and since it is only created after reports have been sent, there should be no mistakes when deleting the files involved in the process; such as log files, text version of log files and files that are dynamically updated or generated throughout the process.

This script will delete the content of:

- DCLogs\Outbox*.*
- DCLogs\Text*.*
- DCLogs\SecLogs\Backup*.* (only what have been backed up, by checking the archive attribute on files)
- DCLogs\Empty.txt

This last area is mostly designated to prepare all directory structure for the next time the scripts are schedule to run.

Comments

Although we minimize the amount of time needed to complete the process we still have to review the reports we receive and since they are sent in separate files for individual event review, we might be prompted for a new approach to this issue.

One example would be that since the files are in CSV format, they can be imported into excel, where we can sort by column or generate graphical representation of totals per events, etc. Another example would be to create only one file with all the information gathered and import it into an SQL database using DTS, and then use Crystal Report to generate more friendly and detailed reports. Also, remember that this report can be provided on a web page if the client has the viewer plug-in for Crystal Report. Another change could be instead of sending the files as attachments just send a notification with a list of events found and it will be up to the notified user to go to the web page, that by the way can also be restricted, and see what new report can be pulled.

Remember that when engaging in a project like this cost is something to consider. How much it's going to cost you to implement and maintain an in-house solution compared to what the cost will be if you buy an existing tool that will provide similar or better results.

Finally, trying not to go outside of the scope of this project, nor requiring knowledge of different tools and their combination to produce such setup, we did not implement the use of any other software than the built-in Operating System tools and commands, and the freeware software listed in this document.

Conclusion

The goal of this paper was to address the issue of gathering and filtering log files, to advice an administrator when relevant events are found, maintain a very low implementation cost through the use of Windows Native Tools, Built-in Tools, and freeware tools, and some tools from Microsoft Resource Kit. Also to provide an example, as simple as possible, on how to manipulate security event logs in Windows NT/2000 to minimize time required to review log files for entries indicating that a potential attack has occurred, or is in the process of occurring.

While this paper does not attempt to teach no one how to read log files, events that should be collected, how to manipulate event neither log files, nor it tries to show how to write scripts on any way or code in any language. It tries to present an idea through a set of scripts to bring the attention towards another way of addressing the issues already mentioned, and to be a possible solution to it.

Writing a set of scripts that one could just copy and paste, place them in the appropriate location and execute was the intent of the set included in document. It will do much more to someone that spends time to review, modify, test and even create new scripts.

With the use of the above mentioned tools and scripts in this particular scenario, we have been able to minimize the time required to collect all desired event logs from selected servers, process them to filter relevant events on the security log, and send a copy of that report to an administrator. Understanding how critical time is for daily tasks of every security or network administrator the intent of this paper was to reduce the time needed for reviewing log files.

References

Microsoft Resource Kits

1. Microsoft Resource Kits – General page
   http://www.microsoft.com/windows/reskits/default.asp

2. Microsoft Windows 2000 Resource Kit – Free Tool Downloads
   http://www.microsoft.com/windows2000/techinfo/reskit/tools/default.asp

3. Microsoft Listing of all products Resource Kits - Get easy access to product Resource Kits here.
   http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/reskits/rktmain.asp

Microsoft Support Articles – Event Descriptions

4. Microsoft Security Event Descriptions - Windows NT 3.5, 3.51 and NT 4.0
   http://support.microsoft.com/default.aspx?scid=kb;en-us;Q174074

5. Microsoft Security Event Description - Windows 2000 Security Event Descriptions Part 1
   http://support.microsoft.com/default.aspx?scid=kb;en-us;Q299475

6. Microsoft Security Event Description - Windows 2000 Security Event Descriptions Part 2
   http://support.microsoft.com/default.aspx?scid=kb;en-us;Q301677

Microsoft Event Logs general information

7. Most event messages generated by the Windows 2000
   http://www.microsoft.com/windows2000/techinfo/messages/default.asp

8. This section provides detailed information about what events are
   http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/acs/proddocs/ac2k/acjmmo_eventcon.asp

9. Disable logging for specific events
   http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/acs/proddocs/ac2k/acjmmo_disablog.asp

10. Event Logging and Viewing Windows NT 4.0
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/winntas/maintain/monitor/03wntpcc.asp

11. Auditing Windows NT Security Features and Controls
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/winntas/maintain/nt4sac/sacch13.asp

12. Event Logging and Viewing Windows 2000
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/maintain/optimize/03w2kadb.asp

13. Audit Account Logon Events
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/maintain/monitor/logevnts.asp

14. Tracking Logon and Logoff Activity in Windows 2000
    http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windows2000serv/maintain/monitor/logonoff.asp

15. Microsoft Windows 2000 online help – See under security
    http://www.microsoft.com/windows2000/en/server/help/

16. Best practices for auditing
    http://www.microsoft.com/windows2000/en/server/help/sag_SEconceptsImpAudBP.htm

Freeware Tools and Other Resources

17. Wayne's NT Resources for Administrators and Users – some event details
    http://is-it-true.org/nt/atips/atips155.shtml

18. ELSave is a tool to save and/or clear a NT event log by Jesper Lauritsen
    http://www.ibt.ku.dk/jesper/NTtools/

19. vdate.exe - Displays the current date and time allows multiple formatting options by David Tribble
    http://david.tribble.com/programs.htm

20. SomarSoft's DumpEvt is a Windows NT program to dump the event log
    http://www.systemtools.com/somarsoft/

21. Blat console utility that sends the contents of a file in an e-mail message using the SMTP protocol
    http://www.interlog.com/~tcharron/blat.html

22. NetViewX is a tool to list the servers in a domain or workgroup by Jesper Lauritsen
    http://www.ibt.ku.dk/jesper/NetViewX/default.htm

23. Other Enumeration tools like netviewx can be found at this url
    http://hackingexposed.com/win2k/links.html

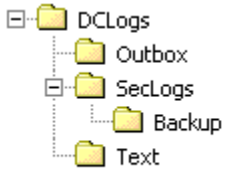
24. Information about all current DOS commands
    http://www.easydos.com/dosindex.html

25. Borland Delphi 6.0 – used to compile erlist
    http://www.borland.com/delphi

26. Frank Heyne Software – Other event related tools
    http://www.heysoft.de/index.htm

Appendix A: Folder structure

<ON NETWORK SERVER>
PC NAME: NETSERVER
FOLDER LOCATION: C:\DCLogs



DCLogs: Shared Root Folder
Outbox: Files ready to be attached and mailed
SecLogs: Security Log Files gathered from DC and listed servers
Backup: Security Log Files ready for backup
Text: Security Log Files extracted as text files

<ON LOCAL PC>
PC NAME: WKST00
FOLDER LOCATION: C:\TSL

TSL: To Schedule Locally, all files that will be scheduled by using "Scheduled Task"

Appendix B: Tools Used

Windows 2000 native tools
FINDSTR.EXE
MOVE.EXE
NET.EXE

Windows 2000 Built-in Console Commands
FOR
IF
ECHO
TYPE
CALL
ERASE
DEL
PUSHD
POPD

Microsoft Resource Kit
LOGEVENT.EXE

Freeware Tools
NETVIEW.EXE
ELSAVE.EXE
VDATE.EXE
DUMPEVT.EXE
BLAT.EXE
ERLIST.EXE

Appendix C: ERLIST source code

```
Program ErList;
{$APPTYPE CONSOLE}
Uses
 SysUtils;

Function CheckPrm:Boolean;
Begin
CheckPrm:=False;
 If (ParamCount=0) or
    (ParamStr(1)='/?') or
    (ParamCount<2) then
                                  begin
                                  Writeln('Erase files with size equal zero');
                Writeln('Dump a list of remaining files');
                                  Writeln;
                                  Writeln('ErList <PATH> <DumpList>');
                CheckPrm:=True;
                                  end;
End;
Procedure Writefilename(name, outfile: string);
var f: text;
   X: Integer;
 Begin
  {$I-}
  Assign(f,outfile);
  If FileExists(outfile) then
                    Append(f)
                   else
                    ReWrite(f);
       If IoResult = 0 Then  writeln(f, name);
 {$I+}
 Flush(f);
 CloseFile(f);
 End;

Procedure EraseFile(fname: string);
var f: file;
 Begin
  {$I-}
  Assign(f,fname);
   Reset(f);
       If IoResult<>0 Then begin
                     Exitcode:=1;
                     exit;
                    end;
 {$I+}
 CloseFile(F);
 Erase(f);
 End;

Procedure GetDir(S: String);
var
 DirInfo: TSearchRec;
begin
  if DirectoryExists(s) then Chdir(s) else exit;
  FindFirst('*.*', faArchive, DirInfo);
  If DirInfo.Name='' then exit;
               Repeat
                     If DirInfo.Size=0 then
                                 Erasefile(DirInfo.Name)
                                else
                                 IF DirInfo.Name<>ParamStr(2) then
                                                 Writefilename(DirInfo.Name, paramstr(2));
               Until FindNext(DirInfo)<> 0;
 FindClose(Dirinfo);
end;

BEGIN
 If CheckPrm then exit;
 GetDir(ParamStr(1));
END.
```

Appendix D: Scripts

Script file Layout

Script files created and generated will be using extension "**cmd**"
Text files required for the scripts to work will have extension "**txt**"
It's recommended to use NOTEPAD to edit the files

<FILE #>
Description of specific task
Tool and parameters/switches used to generate desired result
<FILE # BEGIN> script file (tested and ready to use as is)

Copy and paste everything on this area using notepad and named accordingly

<FILE # END>

<**FILE 1**>
MKWF.CMD
Description:
run locally on WKST00, collect domain controller's name
generate and execute scripts to dump, move, convert and filter log files
erase all filtered files with no events, using ERLIST.EXE
Tools:
All scripts listed
ERLIST <path> <outputfile>
ERLIST \Outbox file2mail.txt
<**FILE 1 BEGIN**>

```
@echo off
::disconnect possible mapped network drive using w:
if exist w: (net use w: /delete >null)
::map network drive using w:
net use w: \\netserver\DCLogs >null
::move directory focus to network drive w:
pushd w:

::BEGIN SUPPORT TASKs
::creation of DCNAMES.TXT a domain controllers and extra servers list
call getdcnames.cmd
::Create a script DUMPLOG.CMD to dump security event logs on each server from DCNAMES.TXT
call mkdumplog.cmd
::Create a script MOVELOG.CMD to move security event logs from each server on DCNAMES.TXT
call mkmovelog.cmd
::END SUPPORT TASKs

::BEGIN STEP1
::make a copy of the security event log to a local file on each server and clears it using DUMPLOG.CMD
call dumplog.cmd
::move saved logs from each listed server to central storage \SecLogs
call movelog.cmd
::END STEP1

::BEGIN STEP2
::Create a script to extract event log files entries into txt format
call mkevtotxt.cmd
::Convert event log entries into TXT and puts them into \Text
call evtotxt
::Create a script to archive event logs files for later backup
```

```
call mkarchive
::Move files into \Backup directory and rename them adding date and time to the name
::making the restore process easier
call archive.cmd
::END STEP2

::BEGIN STEP3
::Search txt version of log files for listed events on EVENTLIST.TXT file
call scanlog
:: SCANLOG.CMD will call FINDEVT with file name as parameter
:: the results will be placed on \Outbox to later email as an attachment
:: Consolidate the directory to send only files with events found
ERLIST.EXE \outbox file2mail.txt
::END STEP3


::return focus to original drive
popd
::disconnect from network drive and close it
net use w: /delete >null
```

<**FILE 1 END**>

<**FILE 2**>
SENDMAIL.CMD
Description:

run locally on WKST00
send emails to listed names with events found and set a flag to empty the folder when done
w:\msg.txt is a file that contains a generic email body message
w:\rcpt.txt is a file that contains the list recipients
%%s on w:\outbox\ is a variable that contains the name of the file to attach

Tools:

BLAT.EXE
Blat v1.9.4: WinNT/95 console utility used to mail a file via SMTP
by P.Mendes, M.Neal, G.Vollant, T. Charron
http://www.blat.net

Before use run this:
Blat -install <smtp server addr> <sender's addr> <try>
Blat -install smtp.mydomain.com eventchecker@mydomain.com 3

If no event files are found:
blat w:\msg.txt -to someone@mydomain.com -s "NO EVENTS FOUND AT THIS TIME"
If event files are found:
blat w:\msg.txt -tf rcpt.txt -s "NEW EVENTS FOUND" -attacht w:\outbox\%%s

<**FILE 2 BEGIN**>

```
::@echo off
::disconnect possible mapped network drive using w:
if exist w: (net use w: /delete >null)
::map network drive using w:
net use w: \\netserver\dclogs >null
::move directory focus to network drive w:
::pushd w:

::BEGIN STEP4 (MAILOUT)
if exist w:\outbox\file2mail.txt goto mailgo
blat w:\msg.txt -to someone@mydomain.com -s "NO EVENTS FOUND AT THIS TIME"
goto endmail
```

```
:mailgo
for /F "" %%s in (w:\outbox\file2mail.txt) do (
blat w:\msg.txt -tf rcpt.txt -s "NEW EVENTS FOUND" -attacht w:\outbox\%%s
)
@echo Mail is done! Directory ready to empty > w:\empty.txt
:endmail
::END STEP4

::return focus to original drive
popd
::disconnect from network drive and close it
net use w: /delete >null
```
<**FILE 2 END**>

<**FILE 3**>
EMPTY.CMD
Description:
- Run locally on WKST00
- Delete all files after they have been emailed
- Check backup directory, if files are already backed up will delete them

Tools:
- Erase

<**FILE 3 BEGIN**>
```
@echo off
::disconnect possible mapped network drive using w:
if exist w: (net use w: /delete >null)
::map network drive using w:
net use w: \\netserver\dclogs >null
::move directory focus to network drive w:
::pushd w:

::BEGIN STEP5
if not exist w:\empty.txt goto dontempty
@erase w:\outbox\*.* /q >null
@erase w:\text\*.* /q >null
@erase w:\empty.txt /q >null
::remove event files that have been backed-up
erase /A-A w:\seclogs\backup\*.* /Q >null
:dontempty
::END STEP5

::return focus to original drive
popd
::disconnect from network drive and close it
net use w: /delete >null
```
<**FILE 3 END**>

Files stored on server

<**FILE 4**>
GETDCNAMES.CMD
Description:
Called from WKST00
Get names of Domain Controllers
Tools:
NETVIEWX.EXE
http://www.ibt.ku.dk/jesper/NTtools/
NETVIEWX -T domain_ctrl domain_bakctrl -O n > dcnames.txt
<**FILE 4 BEGIN**>

```
::
:: GETDCNAMES.CMD
::
@echo off
NETVIEWX -T domain_ctrl domain_bakctrl -O n > dcnames.txt
:: Tool from JESPER NT tools http://www.ibt.ku.dk/jesper/NTtools/
:: create DC list file DCNAMES.TXT
type serverlist.txt >>dcnames.txt
:: Tool from DOS-Windows
:: Adding non-DC to the dcnames.txt list
```

<**FILE 4 END**>

<**FILE 5**>
GETDATE.CMD
Description:
Called from WKST00
Gets the current time
Each log file name will be modify using this information
Tools:
Vdate
David Tribble
http://david.tribble.com/programs.htm
<**FILE 5 BEGIN**>

```
::
:: GETDATE.CMD
::
@echo off
vdate +%%b%%e%%y_%%H%%M > today.txt
:: Tool from David Tribble http://david.tribble.com/programs.htm
:: create a line on a file with the current date and time
```

<**FILE 5 END**>

<**FILE 6**>
SCANLOG.CMD
Description:
Called from WKST00
Search the txt version of the log files for event number listed on eventlist.txt
Generates a new file per event listed in the outbox folder
Tools:
IF [NOT/EXIST]
FOR [/F]
GOTO
FINDSTR [event id number]
<**FILE 6 BEGIN**>

```
::
```

```
:: SCANLOG.CMD
:: Script to search the txt version of the log files
@echo off
if exist w:\outbox\file2mail.txt (erase w:\outbox\file2mail.txt)

if not exist dcnames.txt (getdcnames.bat)

@for /F "" %%s in (dcnames.txt) do (@if exist w:\text\%%s (call :findevt %%s) )

goto :EOF
:FINDEVT
@if exist w:\eventlist.txt (for /F "" %%e in (eventlist.txt) do (
findstr /C:",%%e," w:\text\%1 >> w:\outbox\%1%%e.txt
)
)
```
<**FILE 6 END**>

<FILE 7>
MKDUMPLOG.CMD
<FILE 7 BEGIN>

```
::
:: MKDUMPLOG.CMD
::
:: This script will only generate a new script "DUMPLOG.CMD" every run time
:: for each server within the latest server list version "DCNAMES.TXT"
:: DUMPLOG will go on each listed server and dump a copy of the security log
:: using ELSAVE from JESPER NT tools http://www.ibt.ku.dk/jesper/NTtools
:: and LOGEVENT from Windows 2000 Resource Kit to register an event in the application log
:: when events are cleared and moved
::

@echo off
if not exist dcnames.txt (getdcnames.cmd)
:: Since this script needs dcnames.txt list file
:: if is not present will call for it

if exist dumplog.cmd (erase dumplog.cmd)
:: Delete any previous version of the script

:: create new script file

@echo :: >> w:\dumplog.cmd
@echo :: DUMPLOG.CMD >> w:\dumplog.cmd
@echo :: Script created by MKDUMPLOG.CMD >> w:\dumplog.cmd
@echo :: >> w:\dumplog.cmd
@echo @echo off >> w:\dumplog.cmd


FOR /F "" %%s in (dcnames.txt) do (
@echo ::SERVER NAME: %%s >> w:\dumplog.cmd
@echo ELSave -s \\%%s -F c:\%%s -l security -q -C >> w:\dumplog.cmd
@echo logevent -m \\%%s -s I -c 5 -r "Event Collection" -e 5000 "Event Log was saved and cleared." >> w:\dumplog.cmd
)

:: new script completed!
```
<FILE 7 END>

<FILE 8>
DUMPLOG.CMD
<FILE 8 BEGIN>
```
::
:: DUMPLOG.CMD
:: Script created by MKDUMPLOG.CMD
::
@echo off
::SERVER NAME: FILESERVER
ELSave -s \\FILESERVER -F c:\FILESERVER -l security -q -C
logevent -m \\FILESERVER -s I -c 5 -r "Event Collection" -e 5000 "Event Log was saved and cleared."
::SERVER NAME: DC01
ELSave -s \\DC01 -F c:\DC01 -l security -q -C
logevent -m \\DC01 -s I -c 5 -r "Event Collection" -e 5000 "Event Log was saved and cleared."
```
<FILE 8 END>

<FILE 9>
MKMOVELOG.CMD
<FILE 9 BEGIN>
```
::
:: MKMOVELOG.CMD
::
:: This script will only generate a new script "MOVELOG.CMD" every run time
:: for each server within the latest server list version "DCNAMES.TXT"
:: This script will go on each listed server and move the saved copy of the security log
:: to \SECLOGS folder for later filtering
:: using some DOS-Windows internal commands::

@echo off
if not exist dcnames.txt (getdcnames.cmd)
:: Since this script needs dcnames.txt list file
:: if is not present will call for it

if exist movelog.cmd (erase movelog.cmd)
:: Delete any previous version of the script

:: create new script file

@echo :: >> w:\movelog.cmd
@echo :: MOVELOG.CMD >> w:\movelog.cmd
@echo :: Script created by MKMOVELOG.CMD >> w:\movelog.cmd
@echo :: >> w:\movelog.cmd

@echo @echo off >> movelog.cmd
@echo :: W: >> movelog.cmd
@echo IF exist I: (net use I: /delete) >> movelog.cmd

FOR /F "" %%s in (dcnames.txt) do (
@echo net use I: \\%%s\c$ >> movelog.cmd
@echo move /y I:\%%s w:\seclogs\ >> movelog.cmd
@echo :check IF EXIST I:\%%s call :check >> movelog.cmd
@echo net use I: /delete /y >> movelog.cmd
@echo logevent -m \\%%s -s I -c 5 -r "Event Collection" -e 5001 "Event Log was moved." >> movelog.cmd
```
<FILE 9 END>

<FILE 10>
MOVELOG.CMD
<FILE 10 BEGIN>

```
::
:: MOVELOG.CMD
:: Script created by MKMOVELOG.CMD
::
:: This script will go on each listed server and move the saved copy of the security log
:: to \SECLOGS folder for later filtering
:: using some DOS-Windows internal commands
::
@echo off
::disconnect possible mapped network drive using l:
if exist l: (net use l: /delete)
net use l: \\FILESERVER\c$
move /y l:\FILESERVER w:\seclogs\
:check IF EXIST l:\FILESERVER call :check
net use l: /delete /y
logevent -m \\FILESERVER -s I -c 5 -r "Event Collection" -e 5001 "Event Log was moved."
net use l: \\DC01\c$
move /y l:\DC01 w:\seclogs\
:check IF EXIST l:\DC01 call :check
net use l: /delete /y
logevent -m \\DC01 -s I -c 5 -r "Event Collection" -e 5001 "Event Log was moved."
```

<FILE 10 END>

<FILE 11>
MKEVTOTXT.CMD
<FILE11 BEGIN>

```
::
:: MKEVTOTXT.CMD
::
:: This script will only generate a new script "EVTOTXT.CMD" every run time
:: for each server within the latest server list version "DCNAMES.TXT"
:: EVTOTXT.CMD will convert a security log from evt format to txt
:: using DUMPEVT from somarsoft @ http://www.somarsoft.com
::
@echo off
if not exist dcnames.txt (getdcnames.cmd)
:: Since this script needs dcnames.txt list file
:: if is not present will call for it

if exist evtotxt.cmd (erase evtotxt.cmd)
:: Delete any previous version of the script

:: create new script file

@echo :: >> w:\evtotxt.cmd
@echo :: evtotxt.CMD >> w:\evtotxt.cmd
@echo :: Script created by MKEVTOTXT.CMD >> w:\evtotxt.cmd
@echo :: >> w:\evtotxt.cmd
@echo @echo off >> w:\evtotxt.cmd

FOR /F "" %%s in (dcnames.txt) do (
IF EXIST w:\seclogs\%%s (@echo DumpEvt /logfile=sec=w:\seclogs\%%s /outfile=w:\text\%%s) >> evtotxt.cmd
)
:: new script completed!
```

<FILE 11 END>

<FILE 12>
EVTOTXT.CMD
<FILE12 BEGIN>

```
::
:: evtotxt.CMD
:: Script created by MKEVTOTXT.CMD
::
:: This script will convert a security log from evt format to txt
:: using DUMPEVT from somarsoft @ http://www.somarsoft.com/
::
@echo off
DumpEvt /logfile=sec=w:\seclogs\FILESERVER /outfile=w:\text\FILESERVER
DumpEvt /logfile=sec=w:\seclogs\DC01 /outfile=w:\text\DC01
```

<FILE 12 END>

<FILE 13>
:: MKARCHIVE.CMD
<FILE 13 BEGIN>

```
::
:: MKARCHIVE.CMD
::
:: This script will only generate a new script "ARCHIVE.CMD" every run time
:: for each server within the latest server list version "DCNAMES.TXT"
:: This script will rename and move each listed server log file
:: to a backup directory. It will call GETDATE.CMD and construct a new
:: server name and extension EVT for archive purpose
:: Using standard DOS-Windows tools
::
@echo off
if not exist dcnames.txt (getdcnames.cmd)
:: Since this script needs dcnames.txt list file
:: if is not present will call for it

if exist archive.cmd (erase archive.cmd)
:: Delete any previous version of the script

:: create new script file

@echo :: >> w:\archive.cmd
@echo :: ARCHIVE.CMD >> w:\archive.cmd
@echo :: Script created by MKARCHIVE.CMD >> w:\archive.cmd
@echo :: >> w:\archive.cmd
@echo @echo off >> w:\archive.cmd

call getdate
FOR /F "" %%d in (today.txt) do (
FOR /F "" %%s in (dcnames.txt) do (
IF EXIST w:\seclogs\%%s (@echo move /y w:\seclogs\%%s w:\seclogs\backup\%%s%%d.evt >> w:\archive.cmd)
)
)
:: new script completed!
```

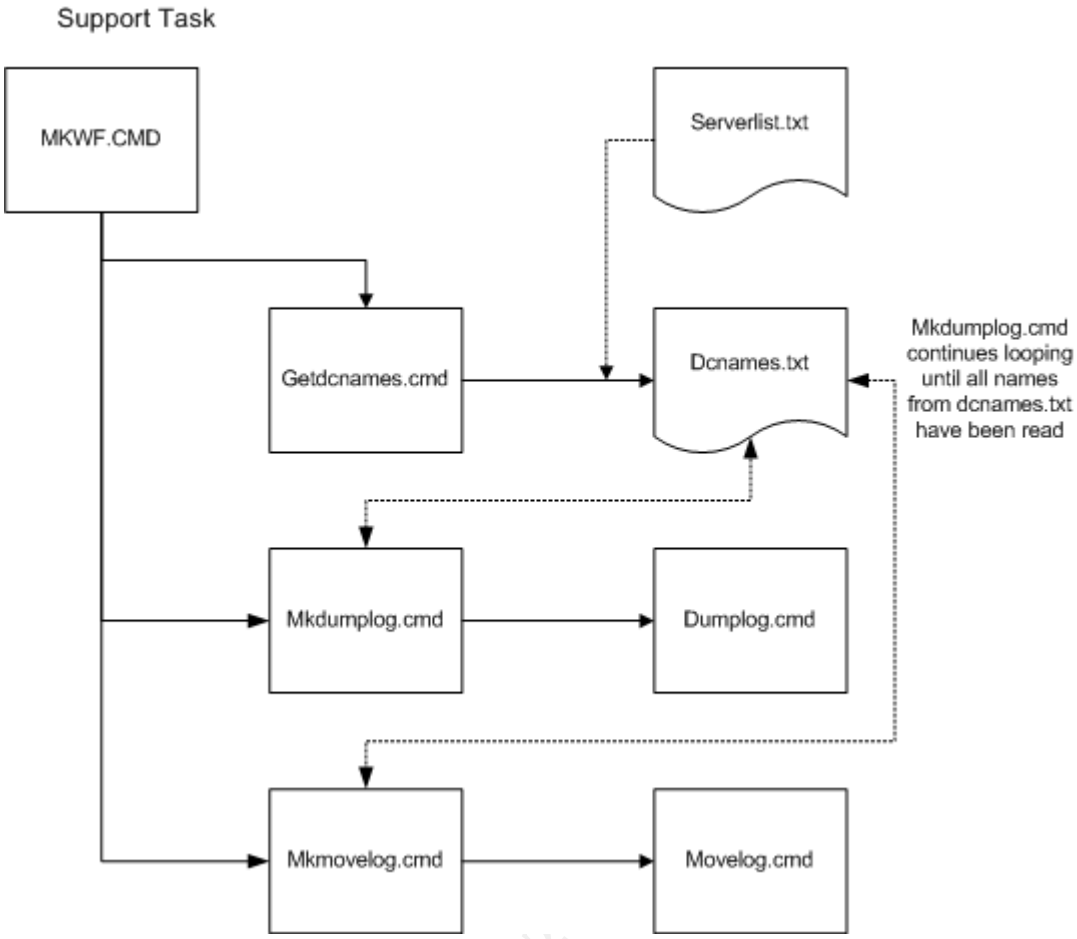
<FILE 13 END>

<FILE 14>
ARCHIVE.CMD
<FILE 14 BEGIN>

```
::
::ARCHIVE.CMD
:: Script created by MKARCHIVE.CMD
::
:: This script will rename and move each listed server log file
:: to a backup directory. It will call GETDATE.CMD and construct a new
:: server name with extension EVT for archive purpose
:: Using standard DOS-Windows tools
::
@echo off
move /y w:\seclogs\FILESERVER w:\seclogs\backup\FILESERVERMar1202_2215.evt
move /y w:\seclogs\DC01 w:\seclogs\backup\DC01Mar1202_2215.evt
```

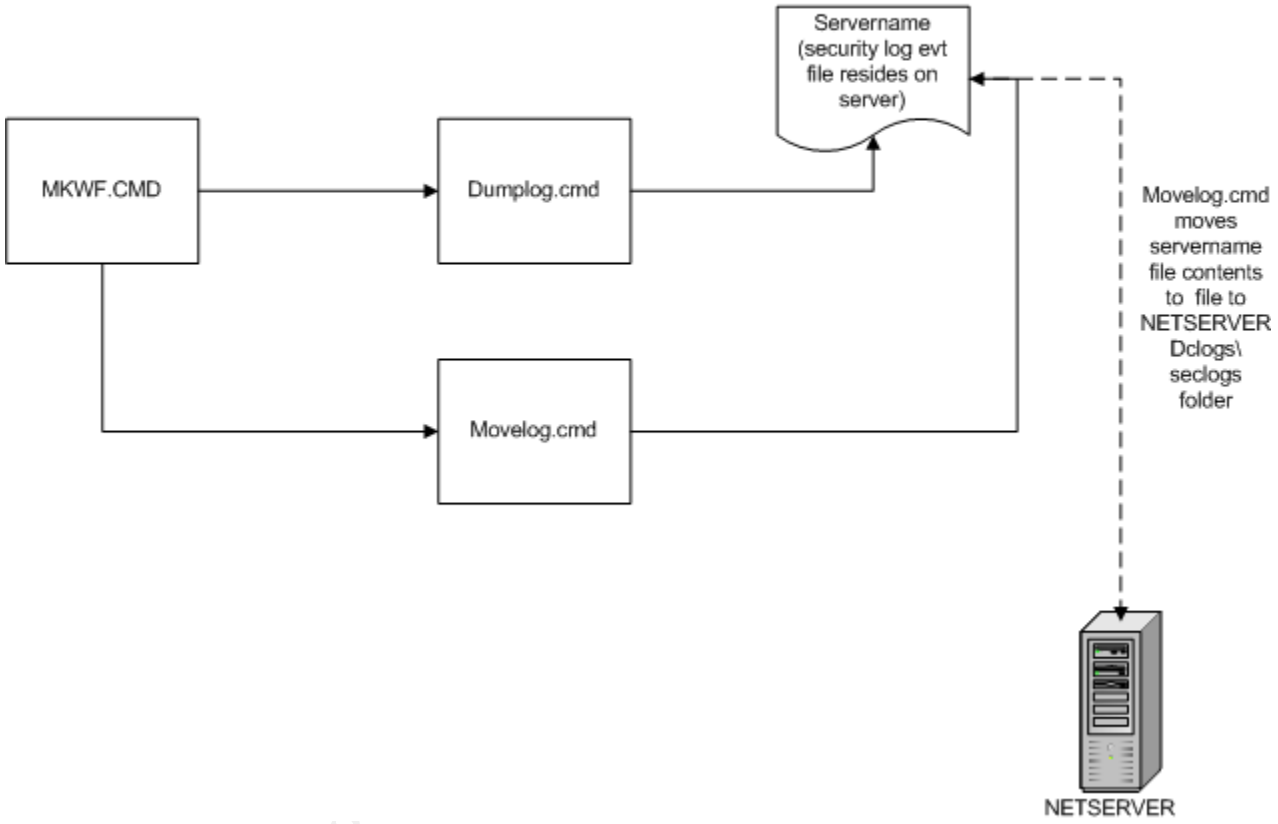
<FILE 14 END>

Appendix E: Flow Chart – MKWF.CMD



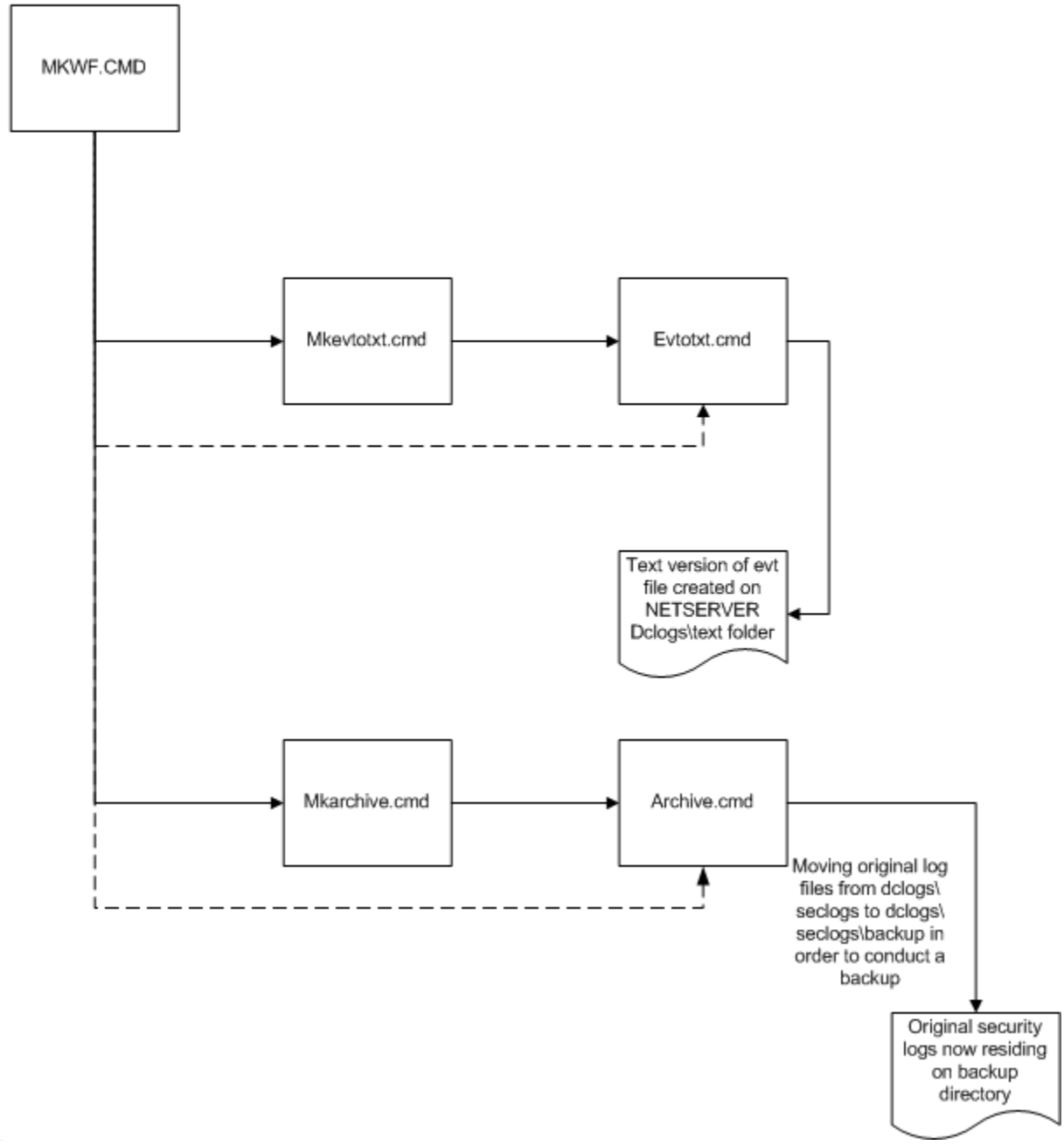
Appendix E: Flow Chart – MKWF.CMD

Step 1

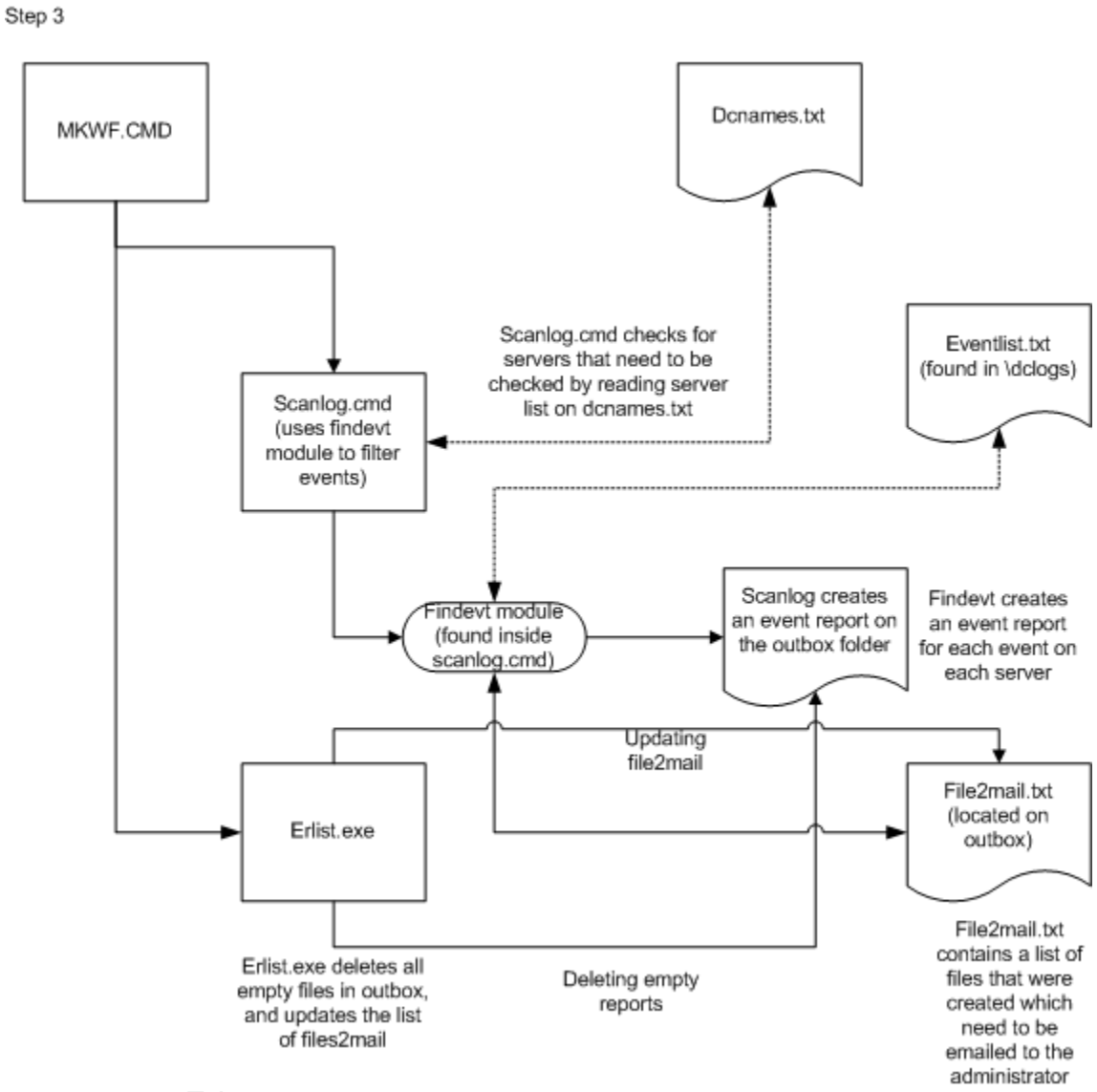


Appendix E: Flow Chart – MKWF.CMD

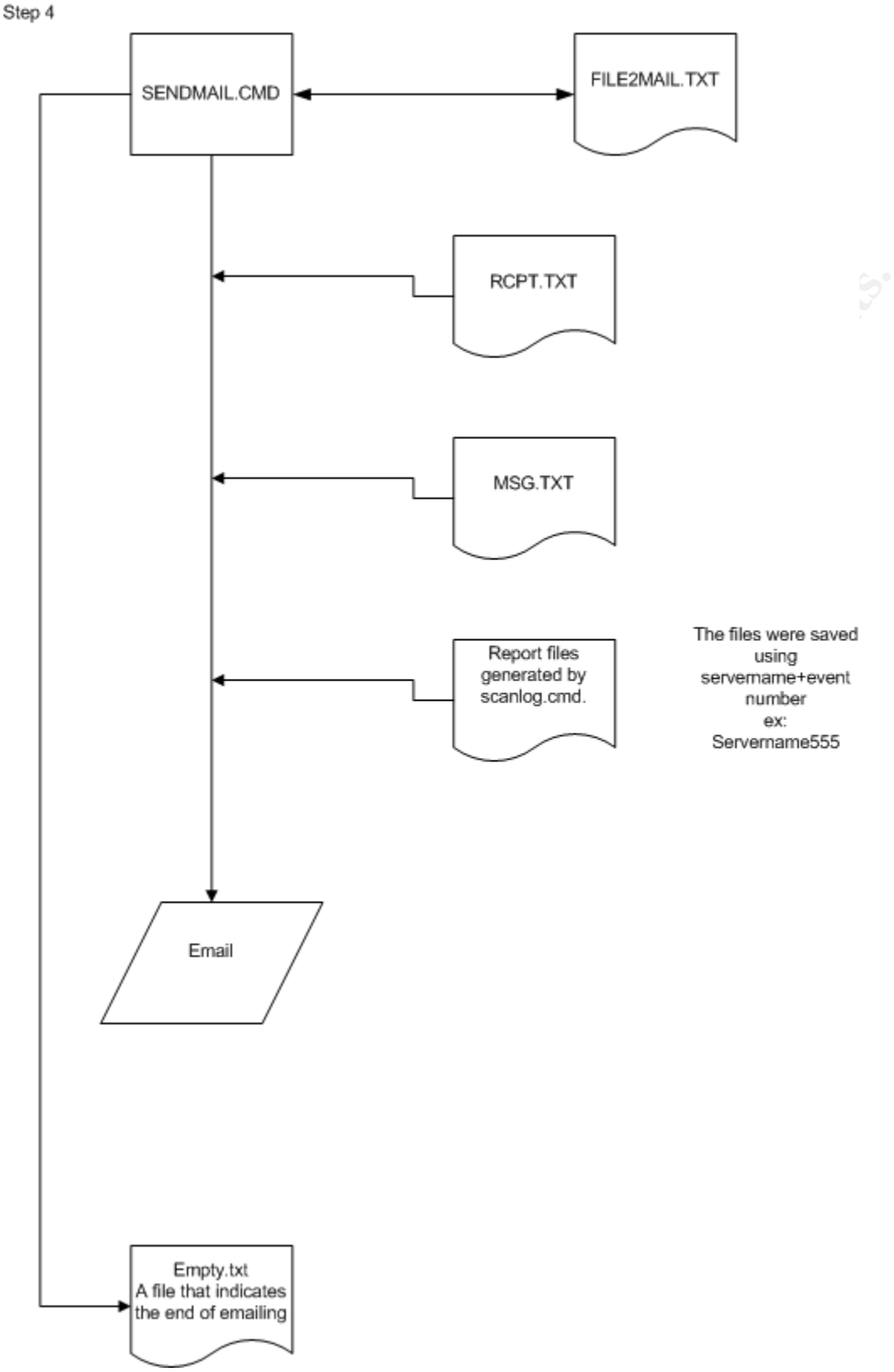
Step 2



Appendix E: Flow Chart – MKWF.CMD



Appendix E: Flow Chart – SENDMAIL.CMD



Appendix F: Event ID quick reference

528: Successful logon
Event entries list:
user name
domain
logon id
logon type
logon process
authentication package
workstation name

Successful logon types:
Type 2 : Console logon - interactive from the computer console
Type 3 : Network logon - network mapping (net use/net view)
Type 4 : Batch logon - scheduler
Type 5 : Service logon - service uses an account
Type 7 : Unlock Workstation

Unsuccessful logon events:
529 : Unknown user name or bad password
530 : Logon time restriction violation
531 : Account disabled
532 : Account expired
533 : Workstation restriction - not allowed to logon at this computer
534 : Inadequate rights - as in user account attempting console login to server
535 : Password expired
536 : NetLogon service down
537 : unexpected error
539 : Logon Failure: Account locked out
627 : NT AUTHORITY\ANONYMOUS is trying to change a password
644 : User account Locked out