Global Information Assurance Certification Paper

Using The WinBatch Scripting Language To Automate Security In An NT4 Environment

Terry Chapman
SANS GSEC Cert v.1.2e
August 16, 2000

Introduction: Why Scripting Is Important for the Security Administrator

A basic scripting knowledge is the Swiss Army knife of any administrator's repertoire. The use of scripting can enable the automation of virtually any repetitive task.
For any security administrator working on a network with more that one server, scripting allows automated patch distribution and time to be spent on the important task of interpreting information, rather than generating it!

There are a couple of myths about scripting that need to be dispelled;

Myth No. 1: Scripting is difficult!
Myth Buster: Anyone who has written a line in a DOS batch file is on the way to writing useful scripts! Try it now… Open notepad, type '@echo Hello World!' (without quotations), save the file as script.bat, from a command prompt now run your new batch file. Welcome to the wonderful world of scripting!!

Myth No. 2: Scripting takes a long time to learn…I'm too busy to learn how to program…
Myth Buster: Can you afford the time to not learn how to do some basic coding? Let's say that you are the security administrator for the XYZ Widget Co. Your boss has asked you to baseline the open ports on all of the company's 150 servers. It is your responsibility to monitor any new ports that open on any of the servers on a daily basis.
Recommending to your boss that you are going to need two assistant administrators to help log on to each of the 150 servers, perform a netstat, and compare the differences with the baseline survey is NOT a good career advancing move!

In this document I will endeavour to guide you through a couple of relatively simple scripts in order to demonstrate that getting started with scripting is not as a daunting task as you may have considered. For the scripting veterans, or those Sysadmins' with more ambitions goals, I will also guide you through a couple of more advanced scripts using my favourite scripting language, Wilsonware's WinBatch.

As you progress with your own scripting efforts remember if there is a will, there is a way to script it.

WinBatch Scripting: An Introduction to WinBatch

WinBatch is a scripting language designed specifically for the Windows platform. It is the brainchild of Morris Wilson, and is supported by the very useful WinBatch resource website (1). A 'tongue in cheek' biography for Mr Wilson can be found on the Wilson WindowWare website (2).
Why WinBatch?? There are plenty of other scripting languages out there for the choosing. In a recent ZDNet poll of scripting languages (3) WinBatch was clobbered by its scripting nemesis Perl. Although there is no question that Perl is a great cross-platform scripting language, WinBatch has concentrated solely on what it knows best; Windows, and has produced an easy to use, solid and powerful scripting language that takes some of the shine off Perl. At the risk of creating a religious scripting war, the gauntlet has been thrown down to all of you expert Perl programmers for security out there! By the way, I wasn't completely honest about WinBatch being a solely Windows scripting language; it also interacts well with Novell and can create cool dynamic HTML scripts with its sister product WebBatch! But that is a topic for another day…

WinBatch comes standard with WinBatch Studio which is a useful colour keyed programming interface; although the purists can feel free to write the code in their favourite text editor. As a network and security administrator one of the most useful features of WinBatch is the ability to easily compile code into distributable executable files. This feature unfortunately is not part of the base WinBatch product, and is bundled with the product at an additional expense. With the view that the executables can be run on any computer within your company without needing additional licensing, the extra expense is well worth the money.
Some of the other features of WinBatch that I have found particularly useful are;

- Simple graphical creation editor for creating such dialogs as push button, check boxes and radio button displays. Automatic script generated can be copied into existing WinBatch scripts.
- Macro creator. Create a powerful, automated script in less than 2 minutes!
- Extensible using freely available 'extenders'. Extenders are available for various functions including; networking NT, networking Novell Netware, email, web, ODBC, serial and parallel port control, active directory manipulation, 'zip' compression and various other functions.
- Easy to follow help files with corresponding examples.

Getting Started: A Practical overview of scripting with WinBatch

Firstly let me say that I am not a hardcore application programmer turned security administrator; nothing could be further from the truth!! Before I began getting creative with WinBatch scripts I was a fair hand at the old DOS batch files, and I had dabbled in a couple of other simple scripting languages, but I would have found it rather difficult to do much useful coding beyond the complexity of a logon script.
Let's get to it! Over the next few pages I have given a few of examples for you to work through. If you would like to pull them apart and play with them, I would suggest that you download the trial version of WinBatch. You can download the application using this link;
http://www.windowware.com/winware/download.html. (4)
Once you have installed the application you can copy any of the code from this document, and then modify and run it from within the application itself. If you decide that my code is worthy of use on you LAN/WAN please remember to send me lots of money, or at least acknowledge my contribution within your script!!

A word of warning and a disclaimer!
For the same reason that you should never run an executable that has a questionable source, always make sure that you understand what a script (or executable) does before running it! The following examples have

been tested in my particular environment, but may not work as expected under all encountered environments (read; all care, no responsibility!). You will notice that most of the scripts in this document have built-in error checking that stops the script if an unrecognised environment arises. If you intend to write scripts that will be directly or indirectly used by end users (e.g. logon scripts or software/patch distribution), make sure that if the script fails, it will fail to a known state.

A couple of pointers.
A seasoned word of advice for the new scripter: Test, test, test, test, and when you have finished test again!! A great way to attract bad publicity towards your scripting efforts is to have your newly distributed patch fail simultaneously on all 2000 of the company's client computers!!
Remember that scripting a task is supposed to automate and make life easier for everyone; 2000 distressed user calls to the support team will not make you Mr/Ms Popular with your colleagues!!
Personally, I have found that a good guide (depending on complexity) is to spend around 50% of the time used in producing a script in testing it. Obviously scripts that are likely to affect more than a few users should receive more attention, than one that you are only ever going to use yourself.
Sometimes testing just cannot predict certain events. In these cases you need to execute your 'break glass in case of emergency' procedures. This is no problem because you have your back-out/rollback plan at the ready…
At this point if your response is 'What back-out/rollback plan??' then it would probably be best if you started updating your CV – and maybe not ask for a reference from this particular company!

Anyone who has ever experienced the very powerful and potentially company crippling 'Windows System Policy' editor will understand when I say; 'Mass automation can be achieved with a minimum of effort with a well written script, but be aware that it also has the potential to damage the company's IT infrastructure on a very large scale'.

This information is not intended to put you off scripting, but it IS intended to make you very paranoid!

Baby Steps: A walkthrough of a couple of simple scripts

A convention – all text following the ';' character is a comment/commentary only, and does not effect the execution of the script. Try to leave a paper trail in all of your scripts; it saves you from having to rewrite chunks of code when you need to recycle.

For the purpose of these next few examples I will assume that you have the ability to write simple DOS batch files. If this is not the case, I would suggest that you complete the tutorial in the WinBatch help file before continuing on. Another excellent resource, if you are completely new to the scripting world, is the very comprehensive DOS manual, 'Using MS_DOS 6.22 Second Edition' published by QUE (5). The WinBatch help files also provide detailed explanations of all commands used in the following examples.

All of the following scripts are real live code! They are out their doing their automated work as you read this document. When I need to roll out a particular script to the general user population, I normally compile the script into an executable and distribute it via a logon script.
For the WinBatch script to run, the client needs to have access to the WinBatch system file; wbdbv32i.dll. This file can be located in the source script directory or in the system path on the client. For speed efficiency, I prefer to copy this file down to the system32 directory on the client.
The easiest way to do this is to create a batch file that copies the file to the client and then calls the WinBatch logon script.
Try using a variation of the following batch file to get 'up and running' if you would like to create a logon

script using WinBatch;

```
@echo off
rem Copy this code into a batch file called logonscript.bat
rem Put the file in the export directory of your replication domain controller.

If exist %systemroot%\system32\wbdbv32i.dll goto script

Copy %logonserver%\netlogon\wbdbv32i.dll %systemroot%\system32\

:script
start /w %0\..\logon.exe

:end
```

Notice the '/w' after the start command, which means wait. If you do not put this parameter in, the batch file will execute and then exit; immediately loading the Windows desktop while your WinBatch logon script is still running!

Note; some lines in the examples are wrapped for readability.

Script One: Disabling Windows Scripting Host

Commands used:
Display – Brings up a Windows dialog box that times-out after a specified time.
If…Endif – If the statement following the 'If' command is true, execute all lines down to 'endif'.
RegExistKey – Checks for the existence of the specified registry key. There are two parts of the command; RegExistKey (@ followed by registry hive, keyname).
RegSetValue – As the name suggests, set a registry value. Same syntax as RegExistKey with the additional field at the end that specifies what the new value is.

One of the first few tasks that I do at a new client company is to disable VBS scripting if it is not being used. This one action makes a very large number of viruses impotent even before your favourite anti-virus software gets to the virus. If I subsequently need to re-enable VBS temporarily, I simply need to use the same script with the original value (VBSFile) rewritten to the key value.
The script can be distributed most easily via a logon script.

```
; Disable Windows Scripting Host
; Author: Terry Chapman
; Version: 1

title = "Scripting Host Disable" ; This is a variable that can be reused
display (3, title, "Disabling Windows Scripting Host. Please wait...") ; Display message for 3 seconds to the end user
; Check to see if the registry key .VBS exists under hkey_classes.  If it does, execute the two lines following the statement.
If RegExistKey (@RegClasses, ".VBS")
        RegSetValue(@REGCLASSES, ".VBS", "") ; Set the .VBS value to "", or blank
        display (3, title, "Windows Scripting Host has been disabled")  ; Display a message to the end user
EndIf ; The other half of the 'If' statement
```

Script Two: Install remote control software on client workstations

The following script was written to install the remote control software 'Funk Proxy'. It forms part of a larger script. In this particular situation each machine needed to be logged on to perform manual copying,

and as there were only a few dozen computers to install it on, I associated this script with a particular user logon account. Simply logging on as the install user instigated the installation process.

Commands used:
Dirget() – Returns current working directory. This is a very useful command that avoids putting 'hard-coded' directory paths in your code.
Strcat – An easy way to tie two or more variables together.
Fileexist – As the name suggests, checks for the existence of the file in the following brackets.
! – The WinBatch equivalent of 'not'
Message – Similar to the display command, but presents an OK button that needs to be pushed before the script continues.
RunHideWait – There are three parts to this command; run – run an executable file that is external to this script, hide – do not make the called executable interactive, wait – wait for the executable to finish before continuing the current script. The syntax is RunHideWait (exe name, optional parameters).
IntControl – The WinBatch IntControl commands are able to access some of the core functions of Windows.

```
; Silently installs Proxy Host for Remote Control
; Date: 19/4/2001
; Author: Terry Chapman
; Version: 1.1

title = "Proxy Host Installer"
dir = dirget() ; Get the directory path that the installer is running in – e.g. \\servername\setuppath\
setup = strcat (dir, "setup.exe") ; ties 2 or more variables together; in this case \\servername\setuppath\setup.exe
phdir = "C:\Program Files\Funk Software\Proxy\"
phset = strcat (phdir, "phset32.exe")

if fileexist (phset) ; Error check to see if the Proxy file is already there. Display a message and exit if it is.
        display (3, title, "Proxy Application is already installed.")
        exit
endif

display (3, title, "Installing Remote Control Software. Please wait...")

if ! fileexist (setup) ; Check to see that the setup file is available.
        message (title, "Proxy setup could not be found.%@crlf%Make sure that this executable (proxyinstall.exe) is in the Proxy
        setup directory.") ; Note - This line is wrapped.  The %@crlf% simply means line return.
        exit
endif

runhidewait (setup, "") ; The line that actually does all of the work!  Run the silent installation of the application.

message (title, "Finished!  Rebooting computer.")
IntControl (67, 0, 1, 0, 0) ; Signal for the computer to reboot
```

Long Division: A walkthrough of a more advanced Script

The following script was written to automate the disabling of user accounts that have not been used for the previous 30 days. One of the complexities to overcome with writing this script was that each domain controller stores the logon information for a user when they last logged onto that particular server. For this reason each domain controller needs to be queried for the user's last logon, and the latest date is taken as the final logon.
In a nutshell the script does the following; takes each domain account, checks all domain controllers for the last logon date, if the account has not been used for over 30 days and is not already disabled the account is now disabled, at the end of the process an email is sent to the administrator with the newly disabled accounts listed, two log files are created listing users who have never logged on and the last logon date of

all domain users.
Use the 'at' command to schedule this script to run on a designated server each day.
This script has been tested on a WAN of around 500 users and 15 domain controllers. It takes around 15 minutes to complete, and doesn't work well on links less than 128KB!

Commands used:
AddExtender – Extenders add functionality to WinBatch. Note; any extenders that are added need to be located within the setup directory or in the client system path.
WntGetDc – Network extender function that determines a reachable domain controller.
Fileopen – As the name suggests, open a ASCII file.
Filewrite – Write to an ASCII file.
Fileclose – Close a file when finished reading. Once the contents of a file have been read it is no longer necessary to leave the file open.
WnServerList – Collect list of servers available.
WntUserList – Generate tab delimited list of users
Itemcount – Return the number of items in the specified variable.
For y = 1 to xyz – Recurse through the following statement for the specified amount of times.
Itemextract – Get a variable from within the delimited variable list
Random – Generate a random number up to the number within the following brackets.
*Box** - Various commands to bring up informational dialogs.
WntUserGetDat – Returns information based on users domain information.
Strsub – Extract part of a variable.
TimeYmdHms() – Returns the current time in the format of year, minute, day, hour, minute, second.
TimeDiffDays – Returns the difference in days of two given dates.
WntUserSetDat – Changes user domain information.
MSendMail – Send an email using installed email client

```
; Disable unused accounts
; Author - Terry Chapman
; Date - 8/6/2001

AddExtender("WWWNT32I.DLL") ; Dll for networking operations
AddExtender("wwmap32i.dll") ; Mail extender

IntControl (1002, 0, 0, 0, 0) ; Hide script icon

title = "Account CleanUp Utility"
dc=wntGetDc( "", "YourDomainName", 1) ; Get the Primary Domain Controller
mail = "" ; Set the variable to blank

handle = fileopen ("c:\neverlogon.txt", "WRITE") ; Create a brand new file if this text file already exists
detail = strcat ("User list of members who have never logged on:", @crlf)
filewrite(handle, detail) ; Add the above line of text, ready to be appended to later
fileclose (handle) ; Close file for the moment
handle1 = fileopen ("c:\lastlogon.txt", "WRITE")
detail1 = strcat ("Last logon time;", @crlf, "Format: User, Full name, Last logon, Last logon server, Do not expire, Currently expired,
Currently disabled")
filewrite(handle1, detail1)
fileclose (handle1)

servers = wntServerList("","",8|16) ; List all of the domain controllers on the network
servercount = itemcount (servers, @tab) ; Count the number of returned servers
userlist = wntUserList(dc, 2) ; Get a list of the users from the PDC
usercount = itemcount (userlist, @tab) ; Count number of items in user list
for y = 1 to usercount ; Repeat the following code until the above number of users have been processed
        useritem = itemextract (y, userlist, @tab) ; Extract username from list
        ran = random (7) ; Generate a random number between 0 to 7 – for dialog colour!!
```

```
year = "0" ; Set all of the following variables to 0 or blank
month = "0"
day = "0"
count = "0"
count1 = "0"
count2 = "0"
lastlogon = "0"
logonfinal = "0"
lastserver = ""
already = "0"

BoxTitle(title) ; Display information dialog
BoxesUp("250,380,750,567", @normal)
BoxColor(1,"0,128,0",ran) ; The random number above is used to generate a different colour dialog on each pass!
Useful for estimating the general speed of the script.
BoxDrawRect( 1, "0,0,1000,1000", 2)
BoxTextColor(1, "255,255,255")
BoxDrawText(1, "200,600,750,350", "Checking last logon details.", @TRUE, 5)
BoxDataTag(1,"tag")

for x = 1 to servercount ; Query each server for each user
        serveritem = itemextract (x, servers, @tab)
        boxtext ("")
        BoxDrawText(1, "200,600,750,350", "Checking user '%useritem%' on %serveritem%.", @TRUE, 5)
        lastlogon = wntUserGetDat(serveritem, useritem, "last_logon") ; Get last logon for each server
        check0 = strsub (lastlogon, 1, 4) ; Check the first four numbers of lastlogon variable
        if check0 != "0000" ; Execute the next statement if the user has ever logged onto this server
                if lastlogon > logonfinal ; If this variable is greater than the last logon variable; increment lastlogon
                to equal logonfinal
                        logonfinal = lastlogon
                        lastserver = serveritem
                endif
        endif
next  ; Go back to the for x= statement until all servers have been queried.  At the end of the for/next statement,
logonfinal will contain the date of the most recent logon on all domain controllers.

non = itemextract (1, logonfinal, ":")
if non == "0000" ; If 0000 the user has not logged into any server
        last = "Account has never been logged in"
else
        Now=TimeYmdHms( ) ; Get current time
        expire=TimeDiffDays(now, logonfinal) ; Calculate the difference between now and the last logon
        if expire == "0"
                last = "Account logged in today"
        else
                if expire < "30"
                        last = "Account logged in %expire% days ago"
                endif
        endif

        if expire > "30" ; If over 30 days old the next couple of lines do the job!
                last = "Account login over 30 days old!"
                flags=wntUserGetDat(dc, useritem,"flags")
                if flags & 2 ; Check to see if account is already disabled
                        already = "0"
                else
                        already = "1" ; Set flag for email log
                        flags=flags | 2 ; Add disable parameter to flags variable
                        wntUserSetDat(dc, useritem, "flags",flags) ; Disable user account
                endif
        endif
endif

info = wntUserGetDat(dc, useritem, "password_expired")
disab = wntUserGetDat(dc, useritem, "flags")
```

```
        fullname = wntUserGetDat(dc, useritem, "full_name") ; Get full name

        ; The following 3 variables (count, count1, count2) are used as flags to indicate on/off status in the log file and
        email for the three settings – password expired, password set not to expire, account disabled
        if disab & 2 ; Increment count1 variable by one if account is now disabled
                count1 = count1 + 1
        endif
        if disab & 65536 ; Increment count2 variable by one if account is set to never expire
                count2 = count2 + 1
        endif
        if info == 1 ; Increment count variable by one if account is set to expire on next logon
                count = count + 1
        endif

        boxdataclear (1, "tag")

        year = StrSub (logonfinal, 1, 4) Get year, month and day to transform to readable format
        month = StrSub (logonfinal, 6, 2)
        day = StrSub (logonfinal, 9, 2)

        if fullname == "" then fullname = "No name listed"

        if count >= "1" ; Reset count* to 1 if it is 1 or greater.  This makes the variable either 1 or 0.
                count = "1"
        endif
        if count1 >= "1"
                count1 = "1"
        endif
        if count2 >= "1"
                count2 = "1"
        endif

        if year == "0000" ; If user has never logged in write to log file
                handle = fileopen ("c:\neverlogon.txt", "APPEND")
                filewrite(handle, useritem)
                fileclose (handle)
                boxtext ("")
                BoxDrawText(1, "200,600,750,350", "'%useritem%' has never logged on!!", @TRUE, 5)
        Else ; or write previous logon to log file
                handle1 = fileopen ("c:\lastlogon.txt", "APPEND")
                msg = strcat (useritem, @tab, fullname, @tab, day, "/", month, "/", year, @tab, last, @tab, lastserver, @tab,
                count2, @tab, count, @tab, count1)
                filewrite(handle1, msg)
                fileclose (handle1)
                msg1 = strcat ("", useritem, "' last logon was ", day, "/", month, "/", year, ".")
                boxtext ("")
                BoxDrawText(1, "200,600,750,350", msg1, @TRUE, 5)
                if expire > 30 && already == "1" ; If account logon is over 30 days old AND hasn't been disable previously
                        mail = strcat (mail, @crlf, msg) ; Append to mail variable
                endif

        endif
next ; Carry on to next user!

if mail <> "" ; If the mail variable is populated send an email of the results of disabled users.  No email will be sent if no
user accounts were disabled.
        recip_email="Infrastructure Team[SMTP:InfrastructureTeam@company.com]"
        time=timedate()
        msg = "The following are accounts that have been disabled due to not being active for the last thirty days;"
        mail = strcat (msg, @crlf, "Format: User, Full name, Last logon, Last logon server, Do not expire password, Password
        currently expired, Currently disabled", @crlf, @crlf, mail)
        ret=mSendMail(recip_email,"Disabled Users Report for %time%.", mail, "", "") ; Send email
endif

beep ; Self explanatory!
```

boxdestroy (1)
display (15, title, "Utility complete!")

Wrapping Existing Tools: Integrating and combining existing NT tools

One of the more powerful ways to use scripting is to 'wrap' existing tools. Combining tools within WinBatch (or any scripting language) provide three main benefits;

- Extend the power and usefulness of existing tools, such as resource kit utilities.
- Save needless coding when there is a utility that already does the job.
- Replace commands within the scripting language when they are not flexible enough for the task at hand…well, no one is perfect! For instance, WinBatch cannot delete a directory while there are files still in it. This is OK for one or two directories deep, but for unadulterated annihilation of numerous embedded directories I prefer to use the DOS command 'rmdir'. Make sure that you do a sanity check BEFORE you allow the command to run; or you will have just become a virus writer!! This is another example of a good CV generating script if you have just distributed a 'buggy' rmdir command across the enterprise… Use this command with care;
  runhidewait ("cmd.exe", " /c rmdir c:\TestedDirectory /s /q").

The Windows NT Resource Kit (6) should be your first stop when looking for extremely useful tools to help automate your day.

Most of the best scripts are written when laziness is the prime motivation!! Quite often a little bit of lateral thinking goes a long way to making your working day much more restful. Let's revisit the earlier example of creating and monitoring an open port baseline on 150 servers in your enterprise.
This is the process that I would use to go about solving this problem;

1. Finish my first cup of coffee.
2. Decide how I wanted to collect a list of open ports on all of the servers. Two valid ideas would be to collect the information centrally using something like Foundstone's 'Fscan' (7) port scanner running from a single server, or generate the information on each server individually using 'netstat'.
3. At this point I would trial the Fscan utility on a couple of servers (with permission from management of course). Upon investigation I would determine that scanning 65535 ports on each server using a centralised scanner would take approximately 10 minutes per server. For the 150 servers this would take a bit over 24 hours. Not a particularly concise snapshot! In addition, if my network were compromised I would want a snapshot of open ports within a few minutes to determine potentially compromised servers.
4. Next hurdle, netstat cannot be run remotely! Time for another coffee.
   There are a couple of options for remotely running applications on servers; copy a batch file or script locally to the server and set it to run via the 'at' command, or use some sort of remote control software such as the resource kit utility 'rclient'(8).
5. Running 150 separate schedules is not only an administrative nightmare that is prone to break in no time flat, but also makes our emergency snapshot difficult.
6. Rclient it is! Now is the time to work some scripting magic!!
7. Rather than list the names of all 150 servers, I would use a centralised script to list all available servers on the network. This saves me from having to constantly update the script as new servers are added to and old servers are removed from the domain. It also eliminates the human error factor of writing out 150 hard coded server names.
8. Next I would need to find a way to run netstat on each server. To start, I would need to install the remote DOS client on each server that the script comes across. You might say that we have just created a worm…and you would be correct!! This technology is not limited just to the forces of

evil! There are a couple of things that you need to remember if you are going to go down this route; ask permission from a senior manager who understands the benefits and implications, make sure that you have locked down the security for Rclient, test every possible scenario, have an automated problem alerter built in, make sure the script fails to a known state, have an 'off' button in case of emergency, and have a back-out/repair utility at the ready. If this path makes you a bit squeamish, you could always associate an installer with a logon user to use on each server…better ask for your two assistants now…

9. As part of my automated installer I would need to check whether the server already had Rclient installed.
10. If not, my script would automatically run Rsetup on the server for the remote client installation.
11. Create a share on the new server and set secured permissions on the directory.
12. Set auditing on the log directory to help keep track of intentional and unintentional changes.
13. Copy down a wrapper for netstat that would archive the netstat results within the newly shared directory.
14. Now that Rclient is installed on the server, I would call the Netstat wrapper using Rclient batch mode. This would run the netstat command on the remote server, and log the results to a dated text file. This would be the baseline for the new server.
15. Replenish the coffee in-take.
16. If the Rclient was already installed when the utility attached to the server I would run the batched Rclient and then the NT utility 'fc' (9) from the central console to compare today's port scan with the baseline. The baseline would be the earliest dated file in the directory. Using this method would enable the baseline to change as new software or services were added to each server. Simply archiving logs prior to the change date would reset the baseline automatically.
17. Append the results of the 'fc' to a central secure log file.
18. Append failed attempts by the utility to the log file for further investigation.
19. When all servers have been queried, trawl the log file with the script to find interesting events.
20. Collate the interesting events (changed ports, unreachable servers) and email the results to your account for you to check through when you arrive in the morning.
21. Set the script to run each morning from a designated server using the 'at' command.

As you can see, with a bit of imagination and research, it is fairly straightforward to create a very powerful tool to make your life easier!

The following script is a practical example of wrapping the 'netsh' (9) utility.
I wrote the application as a quick way to query a domain user's location (via subnet) and IP address…kind of a username ping. The script assumes that you have the 'netsh' command in the system path, that you are using WINS for name resolution, and that the user is not logged onto more than one machine!
Of course if you wanted to improve on the script design, you could always include a wrapped 'nbtstat –A xxx.xxx.xxx.xxx' command to verify the IP address to username resolution!!

Commands used;
AskItemList – Return a list tab delimited list that the user can choose from.
dirmake – Create a new directory.
RegOpenKey – You need to open a registry key before you can read its contents.
Strindexnc – Search a string for a wild card string, ignoring case
Filedelete – Self explanatory! Command can contain wild cards.

```
; Query WINS database to find the current IP address of a user
; Author - Terry Chapman
; Date - 31/7/2001

AddExtender("WWWNT32I.DLL") ; Dll for networking operations
```

```
dc=wntGetDc( "", "YourDomainName", 1) ;Get the primary domain controller
temp = "c:\temp"
users = wntUserList(dc,2)
get = AskItemList("Select user to find IP for.", users,@TAB,@SORTED,@SINGLE) ; Generate a selectable list of user names

if ! direxist (temp) then dirmake (temp)
newtxt = strcat (temp, "\ip", get, ".txt")
param = strcat ("/c netsh wins server xxx.xxx.xxx.xxx show name name=", get, " endchar=3 >", newtxt)
runhidewait ("cmd.exe", param) ; Run the netsh command and write it to a temporary text file

handle = FileOpen(newtxt, "READ") ; Read the text file created by netsh
while @TRUE ; While there are still lines to read!
      x = FileRead(handle)
      If x == "*EOF*" Then Break ; If the at the end of the file, exit
      look = strindexnc(x, "IP Address", 1, @FWDSCAN) ; Scan each line for the 'IP Address' field
                        not = strindexnc (x, "The name does not exist", 1, @Fwdscan) ; Look for no WINS entry
                        if look == 1 then message (get, x) ; Once the IP address field is found, display a message
                        if not == 1 then message (get, "User has not logged on!")
endwhile

fileclose (handle) ; Close file
filedelete (newtxt) ; Delete temporary file
```

Know your registry: Managing En Mass Registry Changes

As a security administrator, some of the more frequent tasks that you perform on the Windows platforms are changes to the registry. Two extremely useful resources when looking for hints on registry manipulation are Microsoft's TechNet (10) and the registry section of Jerold Schulman's website (11).
To paraphrase Microsoft's comments in each of their TechNet articles on registry changes; make sure that you understand the changes that you are making, or you will be sorry! The registry is the brains of Windows 9x, NT and 2000, and should be treated as such.
The humble logon script is normally the best way to distribute mass changes across the enterprise.

My final scripting example is from part of a logon script. It was written to address a security problem at a client's sites. The problem was that some of the local administrators' were leaving their workstations unattended with no password-protected screensaver. A bit of a security 'no no' to say the least!! The script enables a password-protected screensaver with a timeout of 5 minutes.
If a screensaver is already enabled it is maintained, but if the timeout is greater than 5 minutes, the timeout is reset.

Before you implement this type of script make sure that you inform the people who will be affected by these changes. A bit of professional courtesy goes a long way towards end user acceptance. As you probably know, system administrators are your toughest users when it comes to enforcing company security! Make sure that you have support from management, and be firm but fair!

Commands used;
WntGetUser – Returns the username of the currently logged in user.
WntMemberGet – Check to see if the user is a member of the specified network. This is a very useful command for rolling out changes to specific groups of users.
RegOpenKey – You need to open a registry key before you can read its contents.
RegExistValue – Check to see a value exists in the registry.
RegQueryValue – Return registry value.
RegSetValue – Set a registry value.
RegCloseKey – Finish up by closing the opened registry key.

```
; This script is part of a larger logon script
; Check screensaver security of Domain Admin's
; Date: 12/4/2001
; Author: Terry Chapman
; Version: 2.3

AddExtender("WWWNT32I.DLL") ; Dll for networking operations
dc=wntGetDc( "", "YourDomainName", 1) ;Get the primary domain controller
username=wntGetUser(@default)  ;Get the logon name

group=wntMemberGet(dc,"Domain Admins",username,@globalGROUP) ; Check to see if user is part of the Domain
Admin's global group.  This includes the administrator user account.
if group ; If so, execute the following lines of code
        regkey = RegOpenKey(@REGCURRENT, "Control Panel\Desktop")
        screen = RegExistValue(@REGCURRENT, "Control Panel\Desktop[SCRNSAVE.EXE]")
        active = RegQueryValue(@REGCURRENT, "Control Panel\Desktop[ScreenSaveActive]")
        secure = RegQueryValue(@REGCURRENT, "Control Panel\Desktop[ScreenSaverIsSecure]")
        timeout = RegQueryValue(@REGCURRENT, "Control Panel\Desktop[ScreenSaveTimeOut]")
        if screen == @false ; Select default screensaver if there is none
                display (1, title, "Applying ScreenSaver...")
                RegSetValue(regkey, "[SCRNSAVE.EXE]", "%sys32%logon.scr")
        endif
        if active == 0 ; Set screensaver as active if it is not
                RegSetValue(regkey, "[ScreenSaveActive]", "1")
        endif
        if secure == 0 ; Apply password-protection if there is none
                display (1, title, "Securing ScreenSaver...")
                RegSetValue(regkey, "[ScreenSaverIsSecure]", "1")
        endif
        if timeout > "300" ; Set screensaver to timeout in 5 minutes (300 seconds) if the current value is greater
        than 5 minutes or not set
                display (1, title, "Setting 5 minute timeout...")
                RegSetValue(regkey, "[ScreenSaveTimeOut]", "300")
        endif
        RegCloseKey(regkey)
endif
```

Summary: WinBatch – The Security Administrator's Sidekick

Hopefully by now you are wondering how you ever got along without scripting…or if you are already proficient at scripting, you might like to explore some of the capabilities of the WinBatch language.
One of the most important factors of scripting, which cannot be over emphasised, is to test them thoroughly before releasing them into the big bad world. For the few people I have mentored on scripting, this advice only became useful after a distributed mistake…be warned, with power comes responsibility!
If you are writing your own scripts and begin to think 'There must be an easier way!', you are probably right! As you are trying to solve any given problem, remember that there are other people who will have come up against the same problem. A bit of research using Microsoft's TechNet and the Internet usually turns up various solutions to your problems.
Some of the best tools that I have found to automate Windows NT security solutions come from NT itself and from the Windows NT Resource Kits. In particular, the command line utilities allow the creation of batch type jobs that can run across multiple servers.
If you are not already familiar with the standard NT commands and the contents of the Resource Kits, a good couple of hours of exploration will probably be the most productive task you will do this week! Using

these tools from within WinBatch allows for a great deal of scope when trying to achieve flexible solutions. Good luck with you future scripting efforts, and in the spirit of a global security community, consider sharing the scripts that you create and find useful with your colleagues.

References

(1) Wilson WindowWare, Inc. "WinBatch: Macro Scripting Total Windows Automation of System, Networking, and Software Applications.".
URL: http://www.winbatch.com. (16 Aug. 2001)

(2) Wilson, Morris. "Morrie Wilson: The Secret History". URL:
http://www.windowware.com/winware/morrie.html. (16 Aug. 2001)

(3) Windows 2000 Magazine. "Instant Poll Results". 16 August 2001. URL:
http://www.windows2000mag.com/Poll/Index.cfm?QID=170&Action=PreviousPoll

(4) Wilson WindowWare, Inc. "Wilson WindowWare Download Central". URL:
http://www.windowware.com/winware/download.html. (16 Aug. 2001)

(5) Wyatt, Allen. Special Edition, Using MS-DOS 6.22 Second Edition. Indiana: Que Corporation, 2000

(6) Microsoft Corp. "Windows 2000 Server Resource Kit Tools". URL:
http://www.microsoft.com/windows2000/techinfo/reskit/rktour/server/S_tools.asp.
(16 Aug. 2001)

(7) Foundstone Corp. "Foundstone". URL: http://www.foundstone.com/rdlabs/proddesc/fscan.html. (10 Aug. 2001)

(8) Microsoft Corp. "Windows 2000 Resource Kit Tools Help File". Section: Network Management Tools, Remote Console. (2001)

(9) Microsoft Corp. "Windows 2000 Help File". Section: Windows 2000 Command Reference (2000)

(10) Microsoft Corp. "Technet Online". URL: http://www.microsoft.com/technet/. (16 Aug. 2001)

(11) Schelman, Jerold. "JSI, Inc's Reghacks - Windows NT / Windows 2000 Tips, Tricks and Registry Hacks."
URL: http://www.jsiinc.com/reghack.htm (13 Aug. 2001)