



Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permitted without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Securing Windows and PowerShell Automation (Security 505)"
at <http://www.giac.org/registration/gcwn>

Auditing NT User Account Information with VBScript and ADSI Scripting

Author: Brad Sanford
Date: 4/22/2001
Track: GCNT
Version: 1.6a (Option 1)

INTRODUCTION

Protecting user accounts and monitoring their activity are two essential components of every effective security program. I can't imagine a primer on computer security that does not address the danger of poorly managed or poorly audited user accounts. Unfortunately, even with all the warnings, many system administrators routinely overlook this aspect of computer security, sometimes with devastating results. We've all heard the horror stories about computer systems that were compromised due to users who chose poor passwords or no password at all, or the system that was compromised when the password to a dormant account was cracked and the use of that account went undetected. In my experience, poorly managed and monitored user accounts along with the failure to keep software patches current account for well over 80% of all remote server compromises.

That being said, It's not surprising that so many practical assignments from the SANS Windows Security track address issues related to NT user account protection and monitoring. In the remainder of this paper I will attempt to consolidate many of the best practices and techniques from previous practical assignments and then expand upon this base, by providing some VBScript based tools which will allow better auditing of individual NT account properties. Specifically these VBScripts will provide the ability to do the following:

- Dump individual account properties from the NT SAM database for each user
- Identify and report all NT accounts that are not required to change their password at regular intervals.
- Identify and report all NT accounts that have not logged into the domain for some user specified period of time
- Identify and report all NT accounts that have never logged into the domain.

Furthermore, these scripts may provide valuable insight into the power of VBScript and the Active Directory Service Interfaces (ADSI) as an auditing tool for not only NT user account information but also all other information exposed by ADSI. The information exposed by ADSI includes:

- NT User Accounts
- NT Groups
- NT Computers and Services
- NT File and Print Resources
- IIS Metabase
- IIS Web Site Properties

- IIS FTP Site Properties
- LDAP Infrastructures
- Windows 2000 Active Directory

These tools could prove especially useful in large environments or multiple domain environments where third party tools which provide this functionality are often prohibitively expensive.

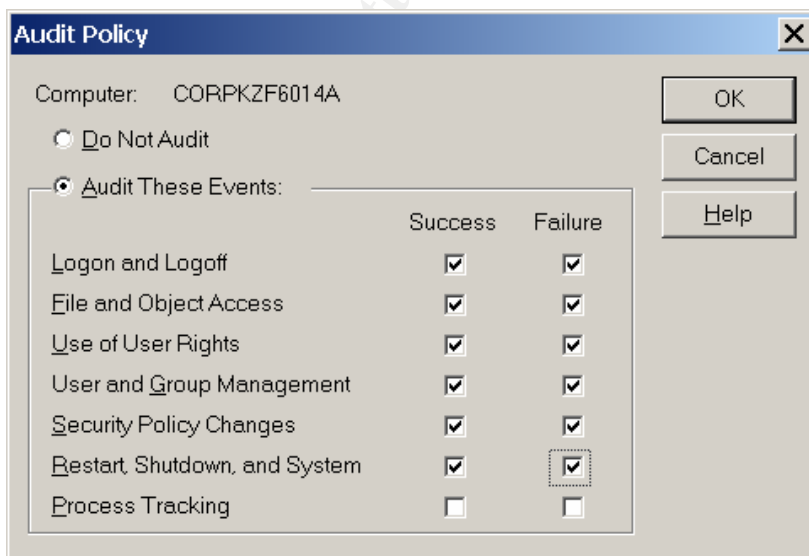
PRIOR WORK

One needs to look no further than previous SANS practical assignments to see that a tremendous amount of effort has already been invested in documenting appropriate practices for auditing NT user accounts. While most of the effort has been focused on documenting best practices in four general areas (Audit Policy, Account Policy, Strong Password Enforcement, and Password appraisal), several authors have alluded to the need to audit individual user account properties.

Audit Policy

There is general consensus among practically all security professionals that all user logon and logoff activity should be logged and routinely monitored in even low to medium security networks. Specifically the SANS Institute recommends enabling auditing for both "Logon and Logoff" successes and failures (Hackendorn S., 2000).

To enable auditing for all Logon and Logoff activity simply open the User Manager, and select Policies → Audit from the menu bar.



On the Audit Policy screen that appears, ensure that the "Audit These Events" radio button is selected and that both the "Success" and "Failure" check boxes

beside "Logon and Logoff" are selected. While the scope of this document is limited to user account auditing, security best practices obviously require enabling additional audit settings based on the security requirements of your environment.

Account Policy

Account policies are configured in the User Manager by selecting Policies → Account from the menu bar. As you can see, the default account policies are abysmally weak (like so many of NT's security defaults), allowing user accounts with no passwords, no password history, and no account lockout.

Computer: CORPKZF6014A

Password Restrictions

Maximum Password Age

☐ Password Never Expires

☒ Expires In 42 Days

Minimum Password Age

☒ Allow Changes ImmEDIATELY

☐ Allow Changes In Days

Minimum Password Length

☒ Permit Blank Password

☐ At Least Characters

Password Uniqueness

☒ Do Not Keep Password History

☐ Remember Passwords

☒ No account lockout

☐ Account lockout

Lockout after bad logon attempts

Reset count after minutes

Lockout Duration

☐ Forever (until admin unlocks)

☐ Duration minutes

☐ Forcibly disconnect remote users from server when logon hours expire

☐ Users must log on in order to change password

OK Cancel Help

While there is not absolute agreement on the specific values for these account policies, there is general agreement on the following points:

- Passwords should be forced to expire at regular intervals
- Minimum password lengths should be enforced

- Accounts should be locked out for some period of time after a few unsuccessful login attempts
- A minimum password age should be established
- Several passwords should be remembered to keep users from reusing the same passwords continuously

Based on the practical assignments submitted by Berdahl, DiEugenio, Hackendorn, and Otis the following specific values seem to be reasonable ranges for these configuration settings:

Maximum Password Age:	45 to 90 days
Minimum Password Age:	1 to 5 days
Minimum Password Length:	8 characters or more
Password History:	5 to 10 passwords
Lockout Threshold:	3 to 5 invalid logon attempts
Lockout Duration:	4 hours (240 minutes) to forever
Lockout Reset:	15 to 60 minutes
Forcibly Disconnect:	True

(Berdahl, 2000), (DiEugenio, 2000), (Hackendorn, 2000) and (Otis, 2000).

Strong Password Enforcement

In the arena of user passwords there is again overwhelming general consensus that some form of strong password enforcement should be implemented and utilized. There is also a strong general consensus of what makes a password a good password:

- Passwords should be at least 8 characters long
- Passwords should consist of uppercase and lowercase letters, as well as numbers and special characters
- Passwords should not contain any part of your username or full name
- Passwords should be easy to remember but hard to guess
- Passwords should not be a word or combination of words contained in a dictionary of any language

Of the practicals which addressed the enforcement of strong passwords, three specific utilities were recommended by the authors:

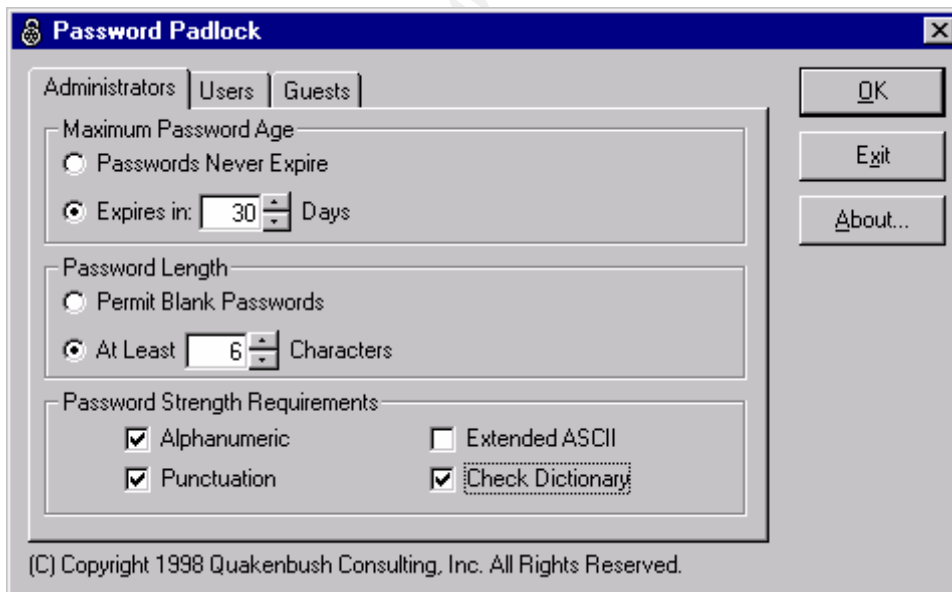
- PASSFILT.DLL by Microsoft (Berdahl, 2000), (DiEugenio, 2000), (Hackendorn, 2000), and (Loser, 2001)
<http://support.microsoft.com/support/kb/articles/Q161/9/90.asp>
- Password Padlock by Quakenbush Consulting (Berdahl, 2000), (DiEugenio, 2000), and (Hutchinson, 2001)

<http://www.quakenbush.com>

- Password Policy Enforcer by TP Information Systems (McDowall, 2000)
<http://www.tpis.com.au/products/ppe/default.htm>

PASSFILT.DLL is the only free utility of the three, but it enforces only minimal password strength requirements. Namely, it requires that passwords be at least 6 characters long, that the password not contain the username or any part of the full name, and that the password be composed of at least 3 of the 4 following classes of characters (uppercase characters, lowercase characters, numbers, and special characters). Microsoft also explains how to create a custom PASSFILT.DLL on their website at http://msdn.microsoft.com/library/psdk/logauth/pswd_about_5z77.htm

Password Padlock is a “free” component that comes with the purchase of Quakenbush’s password appraisal utility, Password Appraiser. The combined toolset is a relatively inexpensive commercial product costing \$495 per domain for unlimited users. For this price it provides a significant amount of functionality and flexibility. According to their website, Password Padlock allows you to set password policies based on the user’s level of access to the system. For example, different policies including password length, composition, and maximum age requirements could be implemented for Administrators, Users, and Guests. Additionally Password Padlock “can require that a password include at least 1 extended ASCII character and fail to be found in a 440,000 word password dictionary built into Password Padlock.” (SANS, 2000).



Password Policy Enforcer is another commercial strong password enforcement utility. PPE is even more feature rich than Quakenbush’s Password Padlock,

providing more granularity of control and additional functionality not offered by the Quakenbush utility. Here's what the TP Information Systems website has to say about their product:

- Rules, rules and more rules
Twelve policy rules allow PPE to enforce virtually any password policy imaginable.
- Multiple password policies
Highly privileged accounts can be assigned to stronger password policies.
- Character substitution detection
Password crackers aren't fooled by users that substitute the letter S with a \$. PPE knows this and can reject passwords that contain character substitution.
- Configurable compliance level
Compliance with some rules can be made optional. For example, the administrator may require compliance with at least four out of six rules.
- Web Server support
The optional PPE/Web client ensures that Internet and Intranet users also comply with the password policy.
- Does not require a Windows NT/2000 domain
PPE can enforce the password policy for part of a domain, or even where there is no domain controller (e.g. Novell networks).
- Novell NetWare Compatible
PPE can enforce the password policy on Novell NetWare networks.
- Password Change Notification
PPE can execute a program or script whenever a password is changed. The Username and Password are passed to the program or script as command-line parameters. This is ideal for password synchronization applications.
- Administrative override
Network administrators can be permitted to override the password policy. (TP Information Systems, 2001).

Furthermore here's a sample of the functionality that this product provides for the 12 policy rules available:

Rule Properties

This rule ensures that passwords contain the required mix of Alpha characters. The default character set includes a - z and A - Z.

☒ Enabled

Settings

Character set name: [default]

Characters: [default]

Password must ☒ contain ☐ not contain

at least 1 Alpha character(s)

in position to Embedded

Client Messages

Policy: [default]

Reason: [default]

OK Cancel

As you can see, you have very granular control over most aspects of user password composition at your disposal with this tool, including positional requirements and customized character sets not available within the other tools.

Depending on the specifics of your environment this tool may be more expensive than the Quakenbush alternative, costing anywhere from \$99 for a 50 user license up to \$5300 for an unlimited enterprise license. But if added functionality is something you require in your environment, this tool can likely meet your needs.

Password Appraisal

Unlike the general consensus we have in the areas of Audit Policy, Account Policy, and Strong Password Enforcement, some security professionals dispute the utility of password appraisal. Those in favor of password appraisal accurately argue that password cracking tools are widely available and widely used in the hacker community and that we need to find the weak passwords and get them corrected before the bad guys find them. Those opposed to password appraisal legitimately argue that we already know that we can crack NT passwords if we

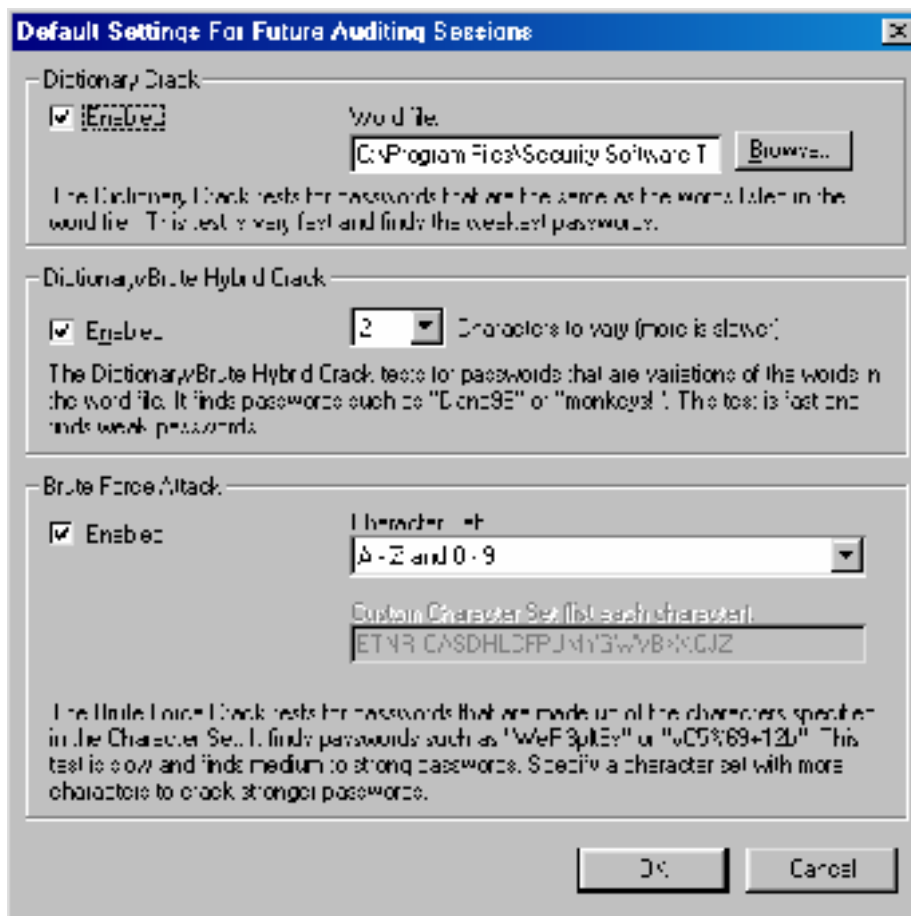
can gain access to the password hashes, and that our time would be better spent making sure that those password hashes are kept protected from prying eyes. Many also feel that the risk of having an individual with access to the cracked passwords of potentially thousands of users is an accident or an incident waiting to happen. I can see the legitimacy of both arguments, nevertheless, this document would not be complete without addressing this aspect of auditing NT user accounts.

Two of the most common NT based password appraisal utilities are L0phtcrack by Security Software Technologies (or is that Security S0phtware Technologies), and Password Appraiser by Quakenbush.

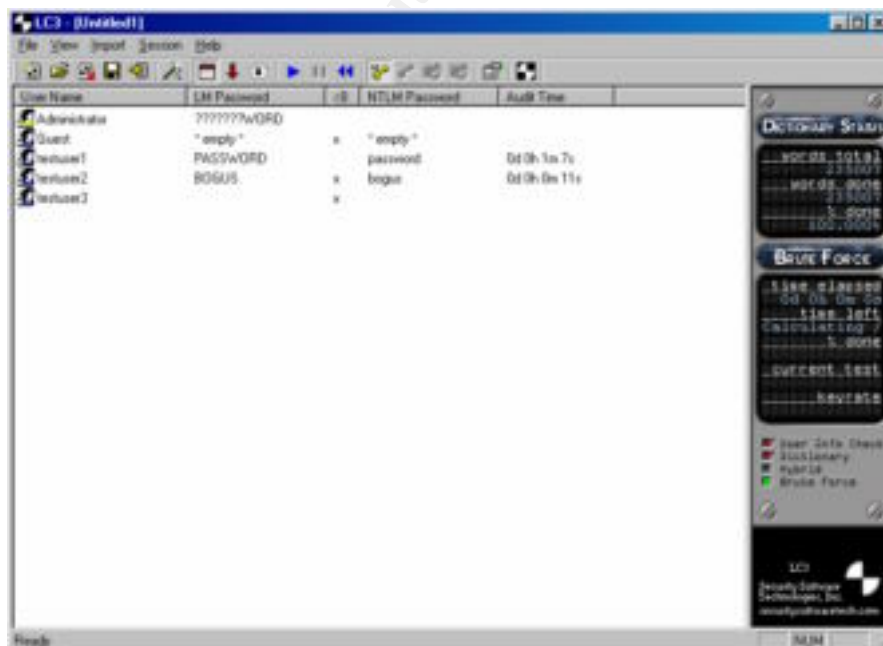
L0phtcrack is one of the most commonly recommended and used "password appraisal" tools among security professionals and hackers alike. In its purest sense, l0phtcrack is a password cracker that is brutally effective at ferreting out weak passwords. It works by extracting password hashes from any number of locations, including the local machine's LSA, remote registries, SAM database files, PWDUMP files and even by sniffing them from the network. L0phtcrack can even extract password hashes from remote systems protected by SYSKEY!

Once the password hashes have been imported into L0phtcrack the user simply has to configure a few simple cracking options such as whether or not to use a dictionary, brute force, or hybrid cracking method (or all three) to crack the passwords, then sit back and wait for L0phtcrack to rip through the hashes.

© SANS Institute 2000 - 2002



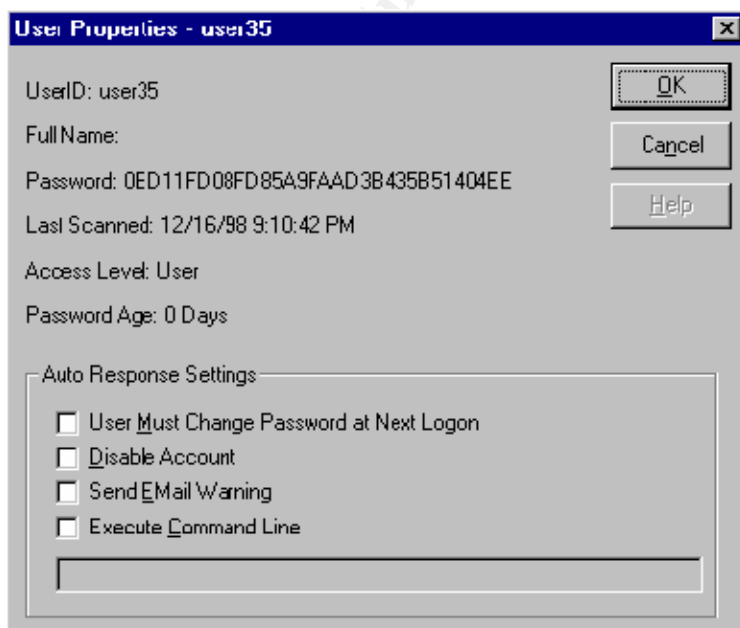
In most environments it won't take long before you start seeing results like the following:



Unfortunately, l0phtcrack seems to have been written more as a cracking tool than an auditing tool, so this unwieldy screen is the final deliverable for your password auditing efforts. You can imagine how difficult it would be initiate corrective action for all the accounts with weak passwords in a large environment of say several thousand accounts. Additionally, it was not until the latest release (version 3) that l0phtcrack even offered to hide the passwords of the cracked accounts as a configuration option.

It is worth pointing out that l0phtcrack is only capable of cracking passwords which consist of the 256 ASCII characters, while there are nearly 400 acceptable characters allowed for NT passwords. This means that it is possible to create a non-crackable password as far as l0phtcrack is concerned. If your password contains one of these non-ASCII characters it will be immune to analysis by l0phtcrack. For further information on this topic I recommend that you read Lois Loser's SANS practical entitled "Making the Crackable Password Non-Crackable". (Loser, 2000).

The Quakenbush Password Appraiser, which comes recommended by (Berdahl, 2000) and (DiEugenio, 2000) uses an easy password database along with an extended analysis database to check for over 60 million easily cracked passwords, but unfortunately, does not have the more robust brute force cracking capabilities like those included in l0phtcrack. Where Password Appraiser shines in comparison with it's less administrator friendly cousin, is in the automated corrective action that it can take on behalf of user accounts with weak passwords as they are discovered. Password Appraiser can require the user to change their password at the next login, disable the account, send an email warning to an administrator, or even execute a command line utility.

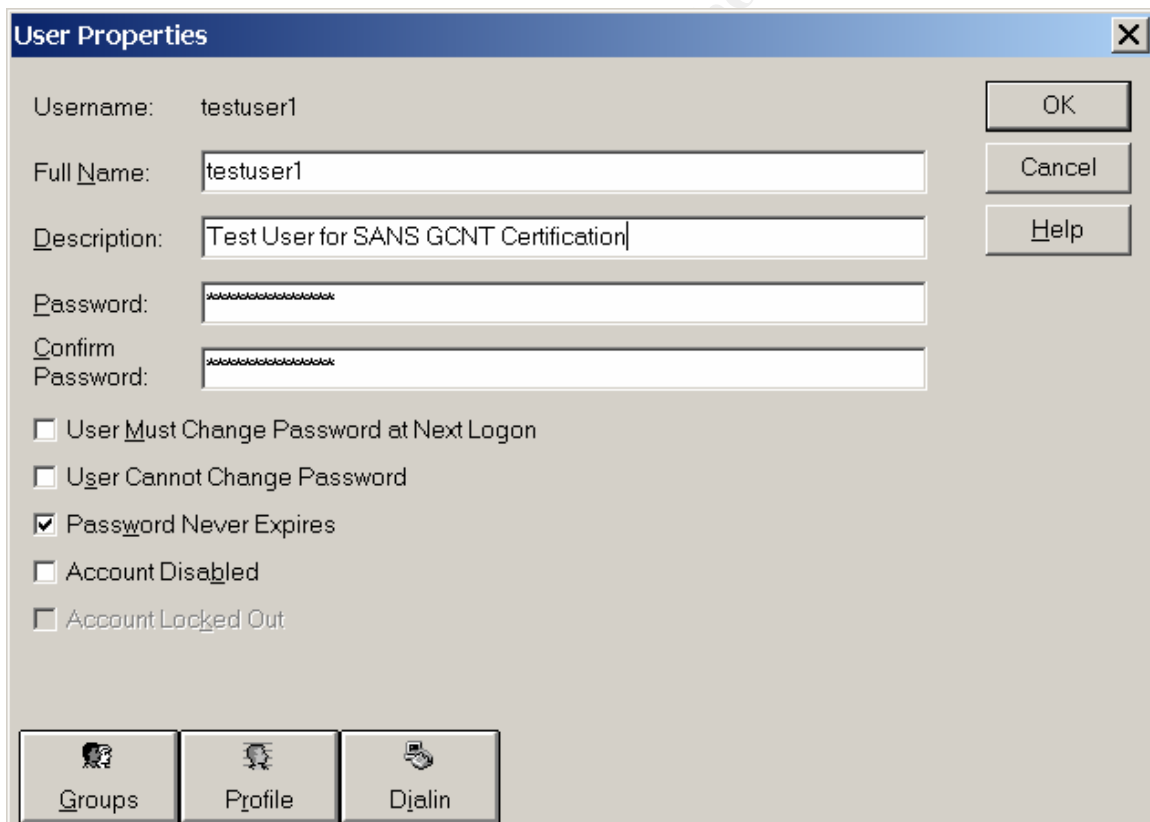


Additionally, the unlimited version of Password Appraiser, which costs only \$495 includes a single domain license for Password Padlock, the strong password enforcement utility described earlier.

Auditing Individual Account Properties

Unfortunately, since NT provides no easy mechanism for even extracting individual user account information from the SAM database, auditing of that information has been largely ignored. You might be asking why it's even necessary to audit individual account information all. I can think of two very important reasons.

First, you need to identify all users that have been configured so that their password never expires. When enabled, this flag overrides the Maximum Password Age requirement specified in the Account Policies for this individual.



The screenshot shows the 'User Properties' dialog box for a user named 'testuser1'. The dialog has a title bar with a close button. On the right side, there are three buttons: 'OK', 'Cancel', and 'Help'. The main area contains several fields and checkboxes:

- Username:** testuser1
- Full Name:** testuser1
- Description:** Test User for SANS GCNT Certification
- Password:** [masked with asterisks]
- Confirm Password:** [masked with asterisks]
- ☐ User Must Change Password at Next Logon
- ☐ User Cannot Change Password
- ☒ Password Never Expires
- ☐ Account Disabled
- ☐ Account Locked Out

At the bottom, there are three buttons with icons: 'Groups' (a group of people icon), 'Profile' (a person icon), and 'Dialin' (a computer icon).

Furthermore, users configured in this manner may not have passwords that comply with the current Account Policy standards because these policies are only enforced at password change time. These users could have passwords that are the same as the account name, passwords that are less than the minimum allowable length, or worse yet they could have no password at all. Everything depends on the account policy that was in force when the password was last

changed. While there are a few specialized cases where this flag may indeed be required, there is general consensus that non-service accounts should very seldom, if ever, be configured in such a manner.

The second reason for auditing individual user account information is so that you can identify and disable or delete dormant accounts. I know that most organizations have some type of termination report which is supposed to serve as a trigger to make sure that user accounts of all terminated employees is revoked. In practice this administrative process almost always breaks down in some critical aspect. Maybe the report doesn't always get routed to all of the people who need to receive it, maybe some managers don't realize that the termination report needs to include contractors who leave the organization as well as employees, or maybe a busy sysadmin innocently misplaces the report before completing all of the revocations. Whatever the mechanism, administrative processes like this are certain to break down at some time or another, and under certain circumstances, the results of such a breakdown could be catastrophic.

Otis recommends disabling dormant accounts that have not been used for 60 days (Otis, 2000), and McDowall recommends a third party product called "Unused Account Ferret" to assist in the process of identifying dormant accounts (McDowall, 2000).

According to TP Information Systems, the publisher of Unused Account Ferret,

"Unused accounts are undesirable because they:

- Provide a means of unauthorized access: Individuals that have left the organization should not have access to network resources, especially employees or contractors that have left in dispute or taken positions with a competitor.
- Are a prime target for brute force password crackers: Password expiration policies have no effect on unused accounts. This gives a password cracker enough time to complete a brute force crack.
- Can cost money: Some organizations determine software licensing requirements by the number of active user accounts. If this is the case, you may be purchasing software licenses for users that do not exist." (TP Information Systems, 2000)

Auditing Individual NT account information accomplished in 2 steps. First, you have to extract the user account information from the SAM database and Second, you need to process the data. Although it's technically possible to combine these two steps together, I generally prefer to perform the tasks separately.

For the data extraction, there are a few third party tools that will allow you to extract this information, and in my mind DumpSec is probably the best free tool that I have found. However, for the purposes of this paper I have written a Visual Basic Script that utilizes the Active Directory Service Interfaces to extract most of the user information from the SAM database.

When it comes to processing this data in any meaningful way, your options are severely limited. Most tools in this product space (in fact, all of the tools I'm aware of) are commercial tools, and almost all of those are prohibitively expensive. Large Enterprise Security Management systems like Axent's ESM and PentaSafe's VigilEnt are certainly up to the task, but they are monstrously complicated applications designed to do much, much, more than audit account information. Even the niche product, Unused Account Ferret is quite expensive for large multiple domain environments, costing \$1995 per domain. It's architecture also require a separate installation for each domain you want to audit, and lastly it provides no mechanism for reporting against local accounts, as only domains are supported. Therefore, I have written three additional Visual Basic Scripts to process the file created by my the extract script. These scripts will report all accounts that are not required to change their password, all accounts that have not logged in for a user specified number of days, and all accounts that have never logged in.

ORIGINAL WORK

The core deliverables from this practical assignment are the 4 Visual Basic Scripts alluded to earlier and fully documented in Appendix A.

- USERDUMP.VBS - VBScript which utilizes the Active Directory Service Interfaces (ADSI) to extract the user account information from the SAM database.
- PASSWORD.VBS - VBScript which reads the extract file created by USERDUMP.VBS and generates a delimited file and an HTML File listing all user accounts with passwords that never Expire.
- INACTIVE.VBS - VBScript which reads the extract file created by USERDUMP.VBS and generates a delimited file and an HTML File listing all user accounts that have not logged into the specified system for X number of days.
- UNUSED.VBS - VBScript which reads the extract file created by USERDUMP.VBS and generates a delimited file and an HTML File listing all user accounts that have never logged into the

specified system.

All of these scripts require the Windows Scripting Host (WSH) to be installed on the system where the scripts are to be run. Additionally, the USERDUMP.VBS script requires ADSI to be installed on the system where the scripts are to be run. Both of these services are installed by default on Windows 2000 systems. For Windows NT systems, WSH is installed by default with service pack 4 or later. ASDI is not installed by default, however. In order to successfully install ADSI on a Windows NT server, you must first install Service Pack 6a, then download and install ADSI 2.5 from the following URL:

<http://www.microsoft.com/NTWorkstation/downloads/Other/ADSI25.asp>
(Microsoft, 2000).

To audit your NT user account information you will need to logon with an account that has administrative access to the domain or computer you are attempting to audit and perform the following steps:

- 1) Cut and paste the source code of each script into a file on your system with the same filename and save the files.
- 2) Edit the USERDUMP.VBS script and configure the user defined variables.
- 3) Open a command prompt, and change to the directory where your scripts are stored.
- 4) At the command prompt enter the following command:
cscript USERDUMP.VBS
This script may take a while to run depending on the size of your domain.

```
C:\vbscript\SANS-GCNT>cscript userdump.vbs
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights
reserved.
```

```
The user extract process has completed.
```

```
C:\vbscript\SANS-GCNT>
```

- 5) Edit the PASSWORD.VBS script and configure the user defined variables.
- 6) At the command prompt enter the following command:
cscript PASSWORD.VBS

```
C:\vbscript\SANS-GCNT>cscript password.vbs
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights
reserved.
```


The account password reporting process has completed.

C:\vbscript\SANS-GCNT>

7) Edit the INACTIVE.VBS script and configure the user defined variables.

8) At the command prompt enter the following command:

cscript INACTIVE.VBS

```
C:\vbscript\SANS-GCNT>cscript inactive.vbs
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights
reserved.
```

The inactive account reporting process has completed.

C:\vbscript\SANS-GCNT>

9) Edit the UNUSED.VBS script and configure the user defined variables.

10) At the command prompt enter the following command:

cscript UNUSED.VBS

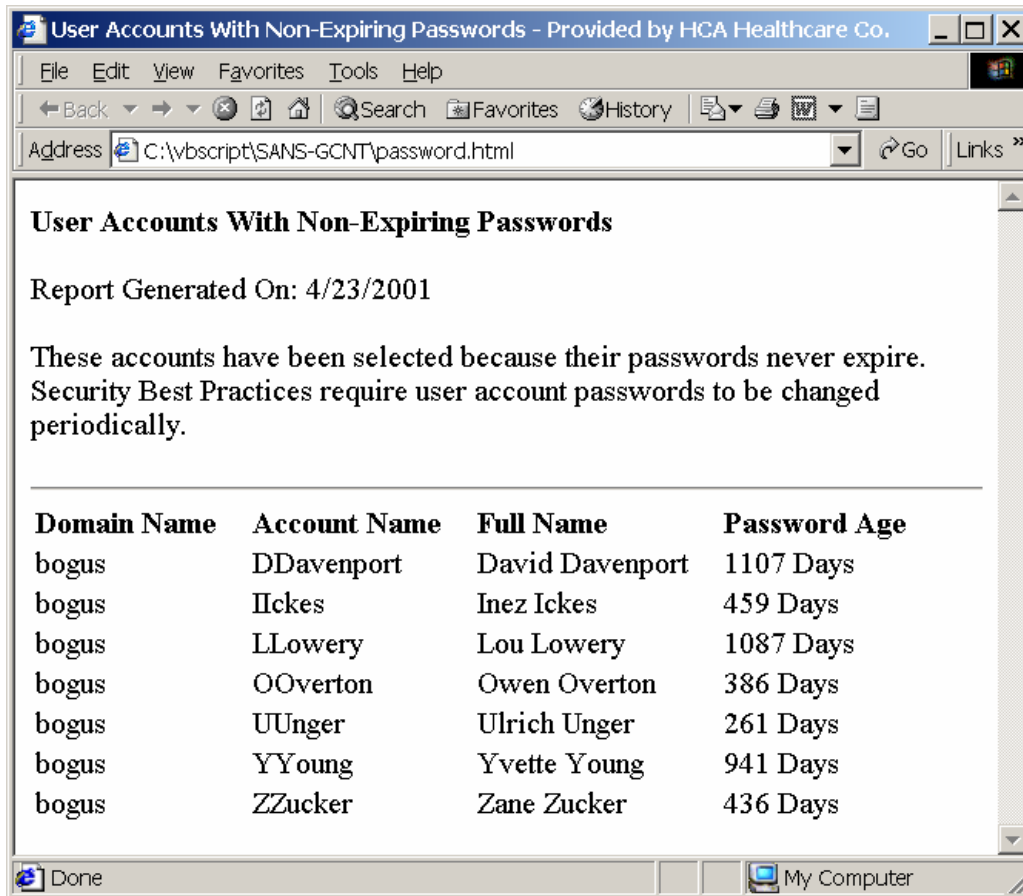
```
C:\vbscript\SANS-GCNT>cscript unused.vbs
Microsoft (R) Windows Script Host Version 5.1 for Windows
Copyright (C) Microsoft Corporation 1996-1999. All rights
reserved.
```

The unused account reporting process has completed.

C:\vbscript\SANS-GCNT>

11) Examine the output files. Here are samples from my execution of the scripts.

Output files from PASSWORD.VBS



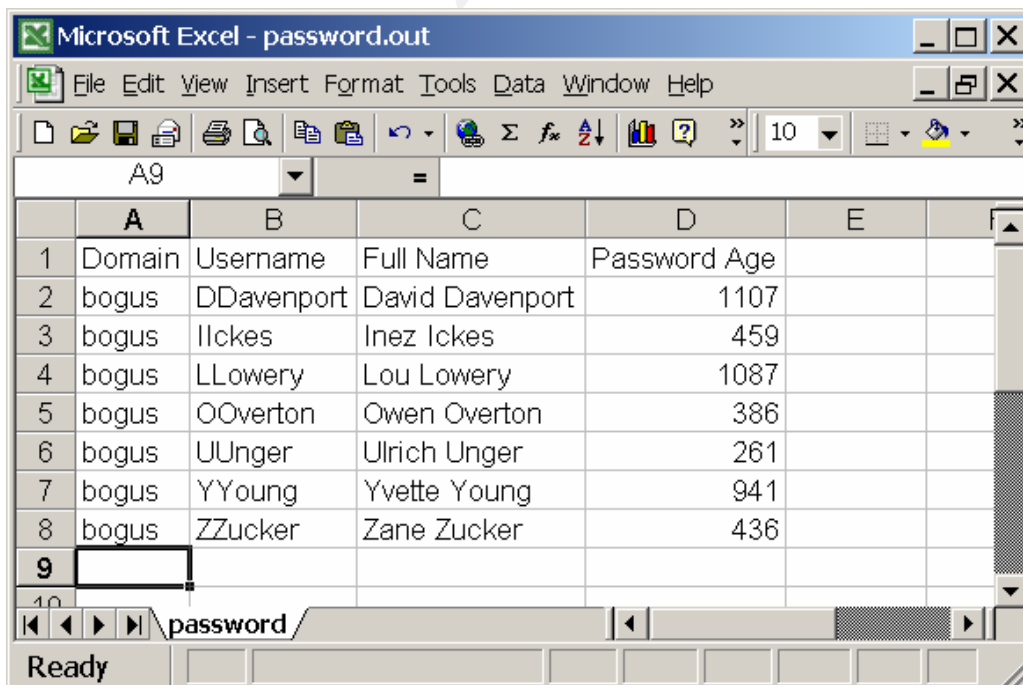
The screenshot shows a web browser window with the title "User Accounts With Non-Expiring Passwords - Provided by HCA Healthcare Co.". The address bar shows the path "C:\vbscript\SANS-GCNT\password.html". The report content includes a title, a generation date, a warning about password expiration, and a table of user accounts.

User Accounts With Non-Expiring Passwords

Report Generated On: 4/23/2001

These accounts have been selected because their passwords never expire. Security Best Practices require user account passwords to be changed periodically.

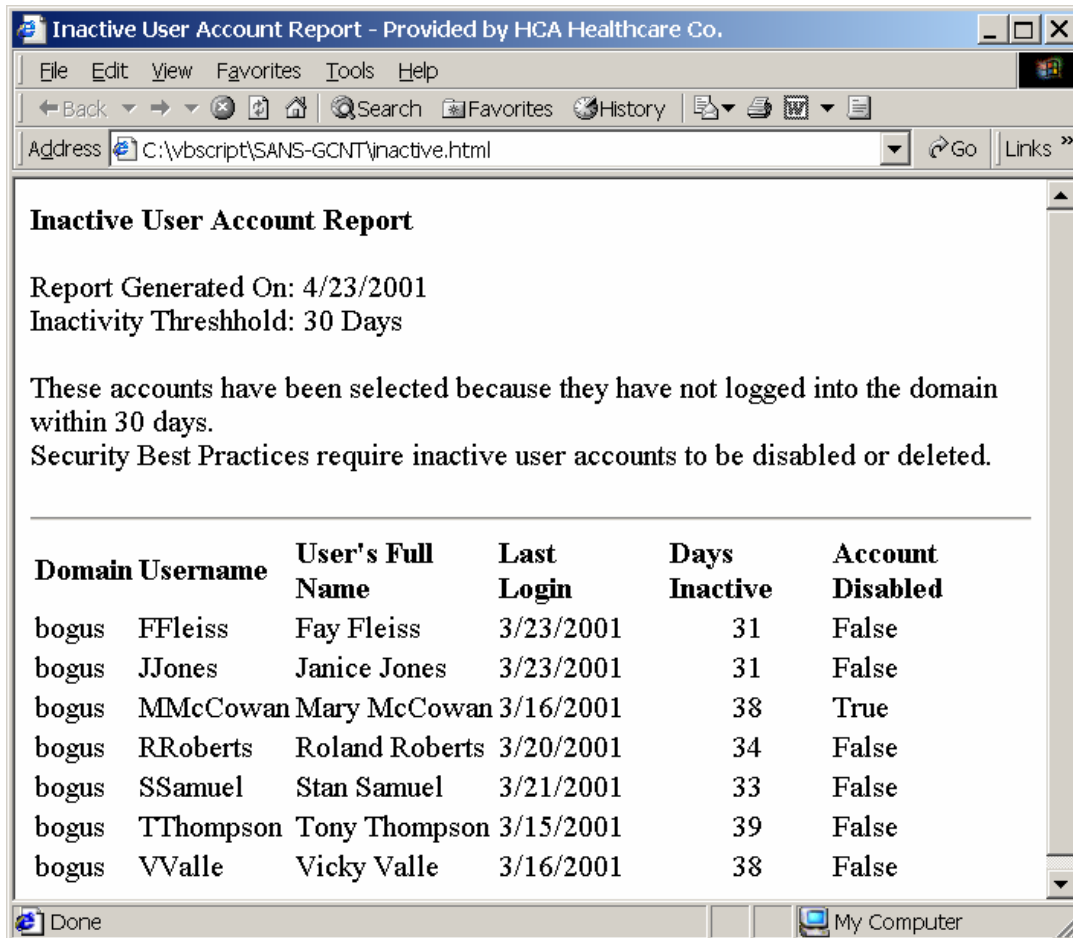
Domain Name	Account Name	Full Name	Password Age
bogus	DDavenport	David Davenport	1107 Days
bogus	IIckes	Inez Ickes	459 Days
bogus	LLowery	Lou Lowery	1087 Days
bogus	OOverton	Owen Overton	386 Days
bogus	UUnger	Ulrich Unger	261 Days
bogus	YYoung	Yvette Young	941 Days
bogus	ZZucker	Zane Zucker	436 Days



The screenshot shows a Microsoft Excel spreadsheet titled "password.out". The data is organized into columns: Domain, Username, Full Name, and Password Age. The rows correspond to the data in the report above.

	A	B	C	D	E
1	Domain	Username	Full Name	Password Age	
2	bogus	DDavenport	David Davenport	1107	
3	bogus	IIckes	Inez Ickes	459	
4	bogus	LLowery	Lou Lowery	1087	
5	bogus	OOverton	Owen Overton	386	
6	bogus	UUnger	Ulrich Unger	261	
7	bogus	YYoung	Yvette Young	941	
8	bogus	ZZucker	Zane Zucker	436	
9					

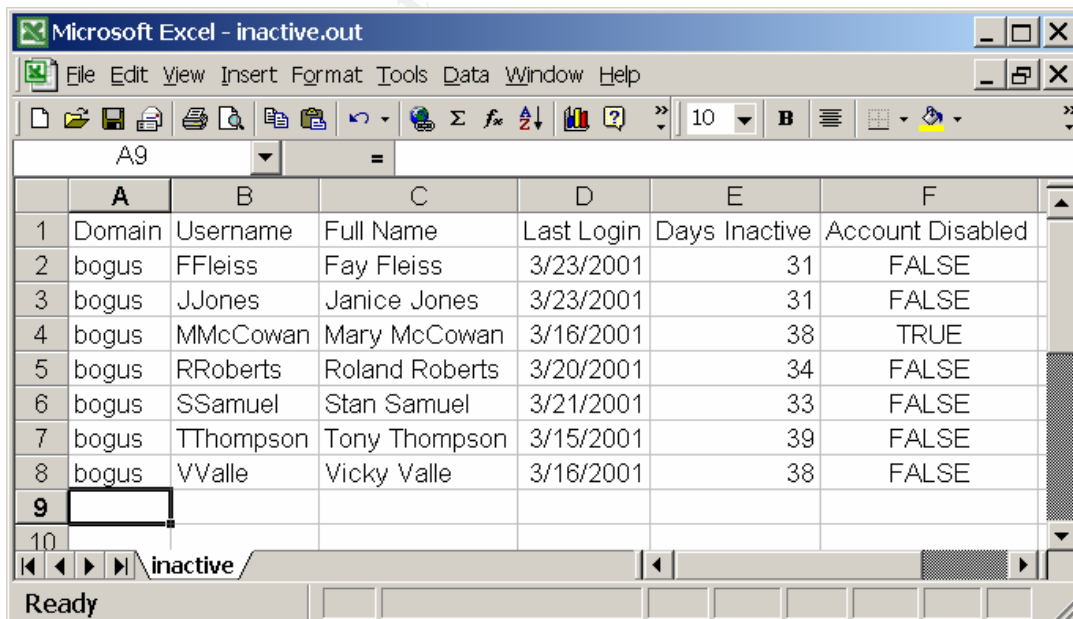
Output files from INACTIVE.VBS



Inactive User Account Report
 Report Generated On: 4/23/2001
 Inactivity Threshold: 30 Days

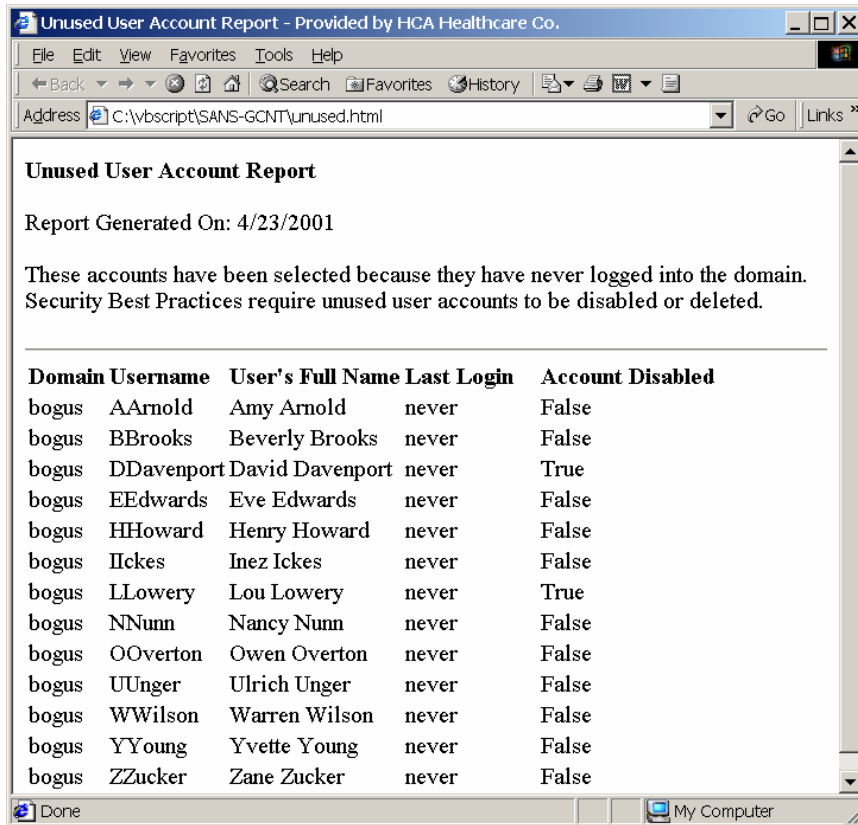
These accounts have been selected because they have not logged into the domain within 30 days.
 Security Best Practices require inactive user accounts to be disabled or deleted.

Domain	Username	User's Full Name	Last Login	Days Inactive	Account Disabled
bogus	FFleiss	Fay Fleiss	3/23/2001	31	False
bogus	JJones	Janice Jones	3/23/2001	31	False
bogus	MMcCowan	Mary McCowan	3/16/2001	38	True
bogus	RRoberts	Roland Roberts	3/20/2001	34	False
bogus	SSamuel	Stan Samuel	3/21/2001	33	False
bogus	TThompson	Tony Thompson	3/15/2001	39	False
bogus	VValle	Vicky Valle	3/16/2001	38	False



	A	B	C	D	E	F
1	Domain	Username	Full Name	Last Login	Days Inactive	Account Disabled
2	bogus	FFleiss	Fay Fleiss	3/23/2001	31	FALSE
3	bogus	JJones	Janice Jones	3/23/2001	31	FALSE
4	bogus	MMcCowan	Mary McCowan	3/16/2001	38	TRUE
5	bogus	RRoberts	Roland Roberts	3/20/2001	34	FALSE
6	bogus	SSamuel	Stan Samuel	3/21/2001	33	FALSE
7	bogus	TThompson	Tony Thompson	3/15/2001	39	FALSE
8	bogus	VValle	Vicky Valle	3/16/2001	38	FALSE
9						
10						

Output files from UNUSED.VBS

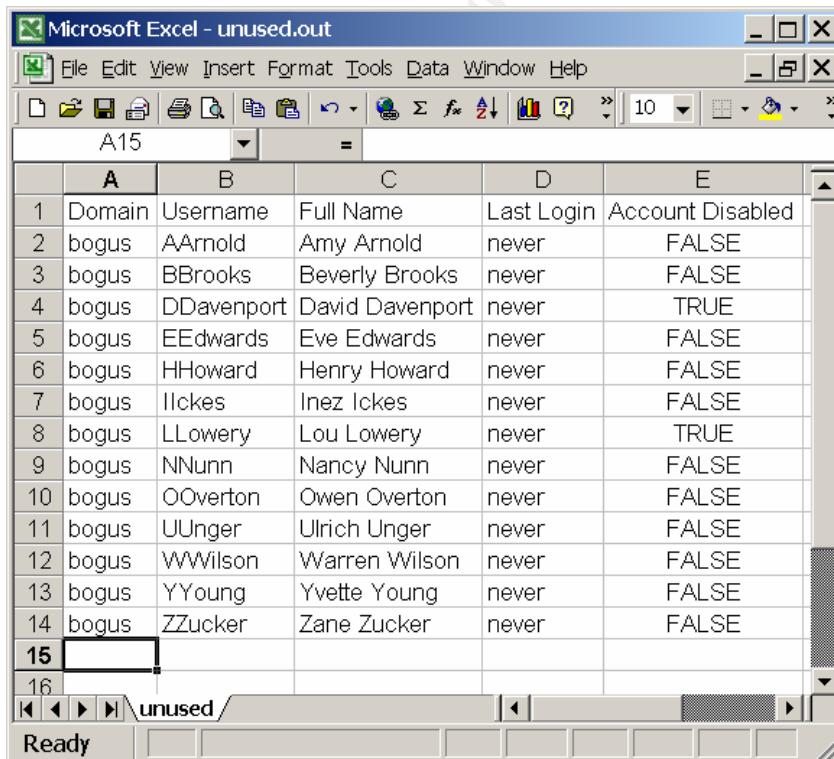


Unused User Account Report

Report Generated On: 4/23/2001

These accounts have been selected because they have never logged into the domain. Security Best Practices require unused user accounts to be disabled or deleted.

Domain	Username	User's Full Name	Last Login	Account Disabled
bogus	AArnold	Amy Arnold	never	False
bogus	BBrooks	Beverly Brooks	never	False
bogus	DDavenport	David Davenport	never	True
bogus	EEwards	Eve Edwards	never	False
bogus	HHoward	Henry Howard	never	False
bogus	Ickes	Inez Ickes	never	False
bogus	LLowery	Lou Lowery	never	True
bogus	NNunn	Nancy Nunn	never	False
bogus	OOverton	Owen Overton	never	False
bogus	UUnger	Ulrich Unger	never	False
bogus	WWilson	Warren Wilson	never	False
bogus	YYoung	Yvette Young	never	False
bogus	ZZucker	Zane Zucker	never	False



Microsoft Excel - unused.out

	A	B	C	D	E
1	Domain	Username	Full Name	Last Login	Account Disabled
2	bogus	AArnold	Amy Arnold	never	FALSE
3	bogus	BBrooks	Beverly Brooks	never	FALSE
4	bogus	DDavenport	David Davenport	never	TRUE
5	bogus	EEwards	Eve Edwards	never	FALSE
6	bogus	HHoward	Henry Howard	never	FALSE
7	bogus	Ickes	Inez Ickes	never	FALSE
8	bogus	LLowery	Lou Lowery	never	TRUE
9	bogus	NNunn	Nancy Nunn	never	FALSE
10	bogus	OOverton	Owen Overton	never	FALSE
11	bogus	UUnger	Ulrich Unger	never	FALSE
12	bogus	WWilson	Warren Wilson	never	FALSE
13	bogus	YYoung	Yvette Young	never	FALSE
14	bogus	ZZucker	Zane Zucker	never	FALSE
15					
16					

CONCLUSION

While the scope of this paper very narrowly addressed the auditing of NT User Account Information, it should be noted that VBScript and ADSI can be leveraged to audit many different resources and properties. If you will recall from the introduction the following items are all exposed via ADSI and can be reported and audited in a very similar fashion to how we audited the user accounts:

- NT User Accounts
- NT Groups
- NT Computers and Services
- NT File and Print Resources
- IIS Metabase
- IIS Web Site Properties
- IIS FTP Site Properties
- LDAP Infrastructures
- Windows 2000 Active Directory

Windows NT/2000 ADSI scripting for system administration (Eck, 2000) is an excellent reference for those wishing to explore these possibilities further.

If you are new to VBScript I also recommend VBSCRIPT in a nutshell: A desktop quick reference Childs, M., Lomax, P., & Petrusha, R. (2000). as an excellent reference.

© SANS Institute 2000 - 2002, Author retains full rights.

Appendix A – Source Code

USERDUMP.VBS

```
'
' PROGRAM NAME:  USERDUMP.VBS
' AUTHOR:       BRAD SANFORD
' DATE WRITTEN:  04/18/2001
'
' NOTE:         THIS PROGRAM CONTAINS USER DEFINED VARIABLES
'               THAT MUST BE CONFIGURED PRIOR TO EXECUTION OF
'               THE SCRIPT
'
' DESCRIPTION:  THIS VISUAL BASIC SCRIPT USES THE ACTIVE
'               DIRECTORY SCRIPTING INTERFACE (ADSI) TO EXTRACT
'               USER INFORMATION FROM THE NT SAM DATABASE.
'
'               THE EXTRACT CONSISTS OF A DELIMITED FILE
'               CONTAINING THE FOLLOWING DATA ELEMENTS FOR
'               EACH USER:
'
'               DOMAIN, USERNAME, HOME DIRECTORY, DESCRIPTION,
'               LOGIN SCRIPT, FULL NAME, PRIMARY GROUP ID,
'               PROFILE, HOME DIRECTORY DRIVE, PASSWORD AGE
'               (IN SECONDS), LAST LOGIN DATE, LAST LOGOFF DATE,
'               ACCOUNT EXPIRATION DATE, BAD LOGIN COUNT,
'               PASSWORD EXPIRATION FLAG, ACCOUNT DISABLED FLAG,
'               HOME DIRECTORY REQUIRED FLAG, ACCOUNT LOCKED
'               FLAG, PASSWORD REQUIRED FLAG, PASSWORD CANNOT BE
'               CHANGED FLAG, PASSWORD NEVER EXPIRES FLAG
'
' Explicitly define all variables used in the program
Option Explicit

Dim Filesys, Contents, Outfile, OutfileName, OutfileHeading
Dim OutfileDelim, Domain, DomainItem, LastLogin, LastLogoff
Dim AccountExpirationDate, BadLoginCount, UserFlags
Dim PasswordExpired, PasswordNeverExpires, PasswordCannotChange
Dim HomeDirRequired

' Handle error conditions instead of aborting script
On Error Resume Next

' =====
' USER DEFINED VARIABLES
' =====
' Domain          - Domain from which to dump user information
' OutfileName      - Fully qualified file name for the extract file
```

```

' OutfileHeading - Print field titles on the 1st line of the
'               extract file?
' OutfileDelim   - Field delimiter for extract file
' =====
Domain = "bogus"
OutfileName = "c:\vbscript\sans-gcnt\userdump.txt"
OutfileHeading = "y"
OutfileDelim = vbTab

' File Constants
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

' Open the extract file
Set Filesys = CreateObject("Scripting.FileSystemObject")
Set Outfile = Filesys.OpenTextFile(OutfileName, ForWriting, True)

' Print field titles if desired
If OutfileHeading = "y" then
    Outfile.WriteLine "Domain" & OutfileDelim & _
        "Username" & OutfileDelim & _
        "HomeDir" & OutfileDelim & _
        "Description" & OutfileDelim & _
        "LoginScript" & OutfileDelim & _
        "FullName" & OutfileDelim & _
        "PrimaryGroupID" & OutfileDelim & _
        "Profile" & OutfileDelim & _
        "HomeDirDrive" & OutfileDelim & _
        "PasswordAge" & OutfileDelim & _
        "LastLogin" & OutfileDelim & _
        "LastLogoff" & OutfileDelim & _
        "AccountExpirationDate" & OutfileDelim & _
        "BadLoginCount" & OutfileDelim & _
        "PasswordExpired" & OutfileDelim & _
        "AccountDisabled" & OutfileDelim & _
        "HomeDirReq" & OutfileDelim & _
        "IsAccountLocked" & OutfileDelim & _
        "PasswordRequired" & OutfileDelim & _
        "PasswordCannotChange" & OutfileDelim & _
        "PasswordNeverExpires"
End If

' Bind to the domain
Set Domain = GetObject("WinNT://" & Domain)

' Cycle through all users in the domain
For each DomainItem in Domain
    If DomainItem.Class = "User" then

' Get the last login date from the PDC
        LastLogin = "never"

```

```

    LastLogin = DomainItem.LastLogin
    ErrorCheck

' Get the last logoff date from the PDC
    LastLogoff = "never"
    LastLogoff = DomainItem.LastLogoff
    ErrorCheck

' Get the account expiration date
    AccountExpirationDate = "never"
    AccountExpirationDate = DomainItem.AccountExpirationDate
    ErrorCheck

' Get the number of Bad Logins
    BadLoginCount = "unknown"
    BadLoginCount = DomainItem.BadLoginCount
    ErrorCheck

' Convert the PasswordExpired flag from an integer to a boolean
    PasswordExpired = False
    If DomainItem.Get("PasswordExpired") = 1 then
        PasswordExpired = True
    End If

' Get the UserFlags field
    UserFlags = DomainItem.Get("UserFlags")

' Extract the PasswordNeverExpires flag from the UserFlags field
    PasswordNeverExpires = False
    If (UserFlags and &H10000) <> 0 then
        PasswordNeverExpires = True
    End If

' Extract the PasswordCannotChange flag from the UserFlags field
    PasswordCannotChange = False
    If (UserFlags and &H00040) <> 0 then
        PasswordCannotChange = True
    End If

' Extract the HomeDirRequired flag from the UserFlags field
    HomeDirRequired = False
    If (UserFlags and &H00008) <> 0 then
        HomeDirRequired = True
    End If

' Build an extract record for the user
    Contents = Domain.Name & OutfileDelim & _
               DomainItem.Name & OutfileDelim & _
               DomainItem.HomeDirectory & OutfileDelim & _
               DomainItem.Description & OutfileDelim & _
               DomainItem.LoginScript & OutfileDelim & _
               DomainItem.FullName & OutfileDelim & _

```



```

        DomainItem.Get("PrimaryGroupID") & OutfileDelim & _
        DomainItem.Profile & OutfileDelim & _
        DomainItem.Get("HomeDirDrive") & OutfileDelim & _
        DomainItem.Get("PasswordAge") & OutfileDelim & _
        LastLogin & OutfileDelim & _
        LastLogoff & OutfileDelim & _
        AccountExpirationDate & OutfileDelim & _
        BadLoginCount & OutfileDelim & _
        PasswordExpired & OutfileDelim & _
        DomainItem.AccountDisabled & OutfileDelim & _
        HomeDirRequired & OutfileDelim & _
        DomainItem.IsAccountLocked & OutfileDelim & _
        DomainItem.PasswordRequired & OutfileDelim & _
        PasswordCannotChange & OutfileDelim & _
        PasswordNeverExpires

' Write the user's extract record
    Outfile.Writeline Contents

End If
Next

' Close the extract file
Outfile.Close
Wscript.Echo "The user extract process has completed."

' Subroutine to process errors
Public Sub ErrorCheck()

' Error numbers -2147463155 and 424 are acceptable, all other
' error codes should be investigated
If Err.Number <> 0 then
    If (Err.Number = -2147463155) or (Err.Number = 424) then
        Err.Clear
    Else
        Wscript.Echo Err.Number & " - " & Err.Description
        Outfile.Writeline Err.Number & " - " & Err.Description
        Wscript.Quit
    End If
End If

End Sub

```

PASSWORD.VBS

```
'
' PROGRAM NAME:  PASSWORD.VBS
' AUTHOR:       BRAD SANFORD
' DATE WRITTEN: 04/18/2001
'
' NOTE:         THIS PROGRAM CONTAINS USER DEFINED VARIABLES
'               THAT MUST BE CONFIGURED PRIOR TO EXECUTION OF
'               THE SCRIPT
'
' DESCRIPTION:   THIS VISUAL BASIC SCRIPT READS THE DELIMITED
'               EXTRACT FILE CREATED BY USERDUMP.VBS AS INPUT
'               AND GENERATES AS OUTPUT A DELIMITED EXTRACT
'               FILE AS WELL AN HTML FILE CONTAINING A LIST
'               OF ALL USERS WITH THE PASSWORD NEVER EXPIRES
'               FLAG SET.
'
'               THE EXTRACT AND HTML FILES CONTAIN THE FOLLOWING
'               DATA ELEMENTS FOR EACH USER:
'
'               DOMAIN, USERNAME, FULL NAME, AND PASSWORD AGE
'
' Explicitly define all variables used in the program
Option Explicit

Dim Infile, InfileName, InfileHeading, InfileDelim
Dim Outfile, OutfileName, OutfileHeading, OutfileDelim
Dim OutfileHTML, OutfileHTMLName
Dim TodaysDate, Filesys, Contents
Dim ReadDomain, ReadArray, ReadUserName, ReadFullName
Dim ReadPasswordAge, ReadPasswordNeverExpires, PasswordAge

' =====
' USER DEFINED VARIABLES
' =====
' InfileName      - Fully qualified file name for the extract file
'                  created by USERDUMP.VBS
' InfileHeading   - Does the extract file created by USERDUMP.VBS
'                  list field titles on the 1st line of the
'                  extract file?
' InfileDelim     - Field delimiter used in the USERDUMP.VBS
'                  extract file
' OutfileName     - Fully qualified file name for the extract file
' OutfileHeading  - Print field titles on the 1st line of the
'                  extract file?
' OutfileDelim    - Field delimiter for extract file
' OutfileHTMLName - Fully qualified file name for the HTML file
' =====
InfileName = "c:\vbscript\sans-gcnt\userdump.txt"
InfileHeading = "y"
```

```

InfileDelim = vbTab
OutfileName = "c:\vbscript\sans-gcnt\password.out"
OutfileHeading = "y"
OutfileDelim = vbTab
OutfileHTMLName = "c:\vbscript\sans-gcnt\password.html"

' File constants
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

' Get the current date
TodaysDate = Date

' Open the files
Set Filesys = CreateObject("Scripting.FileSystemObject")
Set Infile = Filesys.OpenTextFile(InfileName, ForReading)
Set Outfile = Filesys.OpenTextFile(OutfileName, ForWriting, True)
Set OutfileHTML = Filesys.OpenTextfile(OutfileHTMLName, ForWriting, True)

' Skip field titles if necessary
If InfileHeading = "y" then
    Infile.Skipline
End If

' Write field titles if desired
If OutfileHeading = "y" then
    Outfile.WriteLine "Domain" & OutfileDelim & _
        "Username" & OutfileDelim & _
        "Full Name" & OutfileDelim & _
        "Password Age"
End If

' Build the header portion of the HTML file
OutfileHTML.WriteLine "<html>"
OutfileHTML.WriteLine "<head>"
OutfileHTML.WriteLine "<title>"
OutfileHTML.WriteLine "User Accounts With Non-Expiring Passwords"
OutfileHTML.WriteLine "</title>"
OutfileHTML.WriteLine "</head>"
OutfileHTML.WriteLine "<body>"
OutfileHTML.WriteLine "<b>"
OutfileHTML.WriteLine "User Accounts With Non-Expiring Passwords"
OutfileHTML.WriteLine "</b><br><br>"
OutfileHTML.WriteLine "Report Generated On: " & TodaysDate
OutfileHTML.WriteLine "<br><br>"
OutfileHTML.WriteLine "These accounts have been selected because"
OutfileHTML.WriteLine "their passwords never expire."
OutfileHTML.WriteLine "<br>"
OutfileHTML.WriteLine "Security Best Practices require user"
OutfileHTML.WriteLine "account passwords to be changed"
OutfileHTML.WriteLine "periodically."

```

```

OutfileHTML.Writeline "<br><br><hr>"
OutfileHTML.Writeline "<table>"
OutfileHTML.Writeline "<tr>"
OutfileHTML.Writeline "<td><b>Domain Name</b></td>"
OutfileHTML.Writeline "<td width=15></td>"
OutfileHTML.Writeline "<td><b>Account Name</b></td>"
OutfileHTML.Writeline "<td width=15></td>"
OutfileHTML.Writeline "<td><b>Full Name</b></td>"
OutfileHTML.Writeline "<td width=15></td>"
OutfileHTML.Writeline "<td><b>Password Age</b></td>"
OutfileHTML.Writeline "</tr>"

' Cycle through all records in the input file
Do While Infile.AtEndOfStream <> true
' Read each line and split the file into individual fields
Contents = Infile.ReadLine
ReadArray = Split(Contents, InfileDelim)

' Load pertinent data into meaningful fieldnames
ReadDomain = ReadArray(0)
ReadUsername = ReadArray(1)
ReadFullName = ReadArray(5)
ReadPasswordAge = ReadArray(9)
ReadPasswordNeverExpires = ReadArray(20)

' Does the user's password ever expire?
If ReadPasswordNeverExpires = "True" then
' Convert the password age from seconds to days
PasswordAge = Int(ReadPasswordAge / 86400)
' Write an extract record for the user
Outfile.WriteLine ReadDomain & OutfileDelim & _
                  ReadUsername & OutfileDelim & _
                  ReadFullName & OutfileDelim & _
                  PasswordAge
' Write an HTML table entry for the user
OutfileHTML.Writeline "<tr>"
OutfileHTML.Writeline "<td>" & ReadDomain & "</td>"
OutfileHTML.Writeline "<td></td>"
OutfileHTML.Writeline "<td>" & ReadUsername & "</td>"
OutfileHTML.Writeline "<td></td>"
OutfileHTML.Writeline "<td>" & ReadFullName & "</td>"
OutfileHTML.Writeline "<td></td>"
OutfileHTML.Writeline "<td>" & passwordage & " Days</td>"
OutfileHTML.Writeline "</tr>"
End If

Loop

'Close the files and end
Infile.Close
Outfile.Close
OutfileHTML.Writeline "</table>"

```

```
OutfileHTML.Writeline "</body>"
OutfileHTML.Writeline "</html>"
OutfileHTML.Close
WScript.Echo "The account password reporting process has " & _
             "completed."
```

© SANS Institute 2000 - 2002, Author retains full rights.

INACTIVE.VBS

```
'
' PROGRAM NAME:  INACTIVE.VBS
' AUTHOR:       BRAD SANFORD
' DATE WRITTEN: 04/18/2001
'
' NOTE:         THIS PROGRAM CONTAINS USER DEFINED VARIABLES
'               THAT MUST BE CONFIGURED PRIOR TO EXECUTION OF
'               THE SCRIPT
'
' DESCRIPTION:   THIS VISUAL BASIC SCRIPT READS THE DELIMITED
'               EXTRACT FILE CREATED BY USERDUMP.VBS AS INPUT
'               AND GENERATES AS OUTPUT A DELIMITED EXTRACT
'               FILE AS WELL AN HTML FILE CONTAINING A LIST
'               OF ALL USERS THAT HAVE NOT LOGGED INTO THE
'               DOMAIN WITHIN X NUMBER OF DAYS.
'
'               THE EXTRACT AND HTML FILES CONTAIN THE FOLLOWING
'               DATA ELEMENTS FOR EACH USER:
'
'               DOMAIN, USERNAME, FULL NAME, LAST LOGIN DATE,
'               DAYS INACTIVE, AND THE ACCOUNT DISABLED FLAG
'
' Explicitly define all variables used in the program
Option Explicit

Dim Infile, InfileName, InfileHeading, InfileDelim
Dim Outfile, OutfileName, OutfileHeading, OutfileDelim
Dim OutfileHTML, OutfileHTMLName
Dim TodaysDate, Filesys, Contents, InactiveDays
Dim InactiveThreshold, DateArray, LastLoginDate
Dim ReadDomain, ReadArray, ReadUserName, ReadFullName
Dim ReadLastLogin, ReadAccountDisabled

' =====
' USER DEFINED VARIABLES
' =====
' InfileName      - Fully qualified file name for the extract file
'                  created by USERDUMP.VBS
' InfileHeading   - Does the extract file created by USERDUMP.VBS
'                  list field titles on the 1st line of the
'                  extract file?
' InfileDelim     - Field delimiter used in the USERDUMP.VBS
'                  extract file
' OutfileName     - Fully qualified file name for the extract file
' OutfileHeading  - Print field titles on the 1st line of the
'                  extract file?
' OutfileDelim    - Field delimiter for extract file
' OutfileHTMLName - Fully qualified file name for the HTML file
' InactiveThreshold - Report users that have been inactive for
```

```

'           this many days
' =====
InfileName = "c:\vbscript\sans-gcnt\userdump.txt"
InfileHeading = "y"
InfileDelim = vbTab
OutfileName = "c:\vbscript\sans-gcnt\inactive.out"
OutfileHeading = "y"
OutfileDelim = vbTab
OutfileHTMLName = "c:\vbscript\sans-gcnt\inactive.html"
InactiveThreshold = 30

' File constants
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

' Get the current date
TodaysDate = Date

' Open the files
Set Filesys = CreateObject("Scripting.FileSystemObject")
Set Infile = Filesys.OpenTextFile(InfileName, ForReading)
Set Outfile = Filesys.OpenTextFile(OutfileName, ForWriting, True)
Set OutfileHTML = Filesys.OpenTextFile(OutfileHTMLName, ForWriting, True)

' Skip field titles if necessary
If InfileHeading = "y" then
    Infile.SkipLine
End If

' Write field titles if desired
If OutfileHeading = "y" then
    Outfile.WriteLine "Domain" & OutfileDelim & _
        "Username" & OutfileDelim & _
        "Full Name" & OutfileDelim & _
        "Last Login" & OutfileDelim & _
        "Days Inactive" & OutfileDelim & _
        "Account Disabled"
End If

' Build the header portion of the HTML file
OutfileHTML.WriteLine "<html>"
OutfileHTML.WriteLine "<head>"
OutfileHTML.WriteLine "<title>"
OutfileHTML.WriteLine "Inactive User Account Report"
OutfileHTML.WriteLine "</title>"
OutfileHTML.WriteLine "</head>"
OutfileHTML.WriteLine "<body>"
OutfileHTML.WriteLine "<b>"
OutfileHTML.WriteLine "Inactive User Account Report"
OutfileHTML.WriteLine "</b><br><br>"
OutfileHTML.WriteLine "Report Generated On: " & TodaysDate

```

```

OutfileHTML.Writeline "<br>"
OutfileHTML.Writeline "Inactivity Threshold: "
OutfileHTML.Writeline InactiveThreshold & " Days"
OutfileHTML.Writeline "<br><br>"
OutfileHTML.Writeline "These accounts have been selected because"
OutfileHTML.Writeline "they have not logged into the domain"
OutfileHTML.Writeline "within " & InactiveThreshold & " days."
OutfileHTML.Writeline "<br>"
OutfileHTML.Writeline "Security Best Practices require inactive"
OutfileHTML.Writeline "user accounts to be disabled or deleted."
OutfileHTML.Writeline "<br><br><hr>"
OutfileHTML.Writeline "<table>"
OutfileHTML.Writeline "<tr>"
OutfileHTML.Writeline "<td><b>Domain</b></td>"
OutfileHTML.Writeline "<td><b>Username</b></td>"
OutfileHTML.Writeline "<td><b>User's Full Name</b></td>"
OutfileHTML.Writeline "<td><b>Last Login</b></td>"
OutfileHTML.Writeline "<td width=15></td>"
OutfileHTML.Writeline "<td><b>Days Inactive</b></td>"
OutfileHTML.Writeline "<td><b>Account Disabled</b></td>"
OutfileHTML.Writeline "</tr>"

' Cycle through all records in the input file
Do While Infile.AtEndOfStream <> true
' Read each line and split the file into individual fields
  Contents = Infile.ReadLine
  ReadArray = Split(Contents, InfileDelim)

' Load pertinent data into meaningful fieldnames
  ReadDomain = ReadArray(0)
  ReadUsername = ReadArray(1)
  ReadFullName = ReadArray(5)
  ReadLastLogin = ReadArray(10)
  ReadAccountDisabled = ReadArray(15)

' Has the user ever logged into the domain?
  If ReadLastLogin <> "never" Then
' Extract the date portion of the Last Login date/time
    DateArray = Split(ReadLastLogin, " ")
    LastLoginDate = DateArray(0)
' Calculate how long the account has been inactive
    InactiveDays = DateDiff("d", LastLoginDate, TodaysDate)
' Has account been inactive longer than the user specified
' threshold?
    If InactiveDays >= InactiveThreshold then
' Write an extract record for the user
      Outfile.WriteLine ReadDomain & OutfileDelim & _
        ReadUsername & OutfileDelim & _
        ReadFullName & OutfileDelim & _
        LastLoginDate & OutfileDelim & _
        InactiveDays & OutfileDelim & _
        ReadAccountDisabled

```



```

' Write an HTML table entry for the user
  OutfileHTML.Writeline "<tr>"
  OutfileHTML.Writeline "<td>" & ReadDomain & "</td>"
  OutfileHTML.Writeline "<td>" & ReadUsername & "</td>"
  OutfileHTML.Writeline "<td>" & ReadFullName & "</td>"
  OutfileHTML.Writeline "<td>" & LastLoginDate & "</td>"
  OutfileHTML.Writeline "<td></td>"
  OutfileHTML.Writeline "<td align=" & chr(34) & "center" & _
    chr(34) & ">" & InactiveDays & _
    "</td>"
  OutfileHTML.Writeline "<td>" & ReadAccountDisabled & "</td>"
  OutfileHTML.Writeline "</tr>"
End If
End If

Loop

' Close the files and end
Infile.Close
Outfile.Close
OutfileHTML.Writeline "</table>"
OutfileHTML.Writeline "</body>"
OutfileHTML.Writeline "</html>"
OutfileHTML.Close
WScript.Echo "The inactive account reporting process has " & _
  "completed."

```

UNUSED.VBS

```
'
' PROGRAM NAME:  UNUSED.VBS
' AUTHOR:       BRAD SANFORD
' DATE WRITTEN: 04/18/2001
'
' NOTE:         THIS PROGRAM CONTAINS USER DEFINED VARIABLES
'               THAT MUST BE CONFIGURED PRIOR TO EXECUTION OF
'               THE SCRIPT
'
' DESCRIPTION:   THIS VISUAL BASIC SCRIPT READS THE DELIMITED
'               EXTRACT FILE CREATED BY USERDUMP.VBS AS INPUT
'               AND GENERATES AS OUTPUT A DELIMITED EXTRACT
'               FILE AS WELL AN HTML FILE CONTAINING A LIST
'               OF ALL USERS THAT HAVE NEVER LOGGED INTO THE
'               DOMAIN.
'
'               THE EXTRACT AND HTML FILES CONTAIN THE FOLLOWING
'               DATA ELEMENTS FOR EACH USER:
'
'               DOMAIN, USERNAME, FULL NAME, LAST LOGIN DATE,
'               AND THE ACCOUNT DISABLED FLAG
'
' Explicitly define all variables used in the program
Option Explicit

Dim Infile, InfileName, InfileHeading, InfileDelim
Dim Outfile, OutfileName, OutfileHeading, OutfileDelim
Dim OutfileHTML, OutfileHTMLName
Dim TodaysDate, Filesys, Contents
Dim ReadDomain, ReadArray, ReadUserName, ReadFullName
Dim ReadLastLogin, ReadAccountDisabled

' =====
' USER DEFINED VARIABLES
' =====
' InfileName      - Fully qualified file name for the extract file
'                  created by USERDUMP.VBS
' InfileHeading   - Does the extract file created by USERDUMP.VBS
'                  list field titles on the 1st line of the
'                  extract file?
' InfileDelim     - Field delimiter used in the USERDUMP.VBS
'                  extract file
' OutfileName     - Fully qualified file name for the extract file
' OutfileHeading  - Print field titles on the 1st line of the
'                  extract file?
' OutfileDelim    - Field delimiter for extract file
' OutfileHTMLName - Fully qualified file name for the HTML file
' =====
InfileName = "c:\vbscript\sans-gcnt\userdump.txt"
```

```

InfileHeading = "y"
InfileDelim = vbTab
OutfileName = "c:\vbscript\sans-gcnt\unused.out"
OutfileHeading = "y"
OutfileDelim = vbTab
OutfileHTMLName = "c:\vbscript\sans-gcnt\unused.html"

' File constants
Const ForReading = 1
Const ForWriting = 2
Const ForAppending = 8

' Get the current date
TodaysDate = Date

' Open the files
Set Filesys = CreateObject("Scripting.FileSystemObject")
Set Infile = Filesys.OpenTextFile(InfileName, ForReading)
Set Outfile = Filesys.OpenTextFile(OutfileName, ForWriting, True)
Set OutfileHTML = Filesys.OpenTextFile(OutfileHTMLName, ForWriting, True)

' Skip field titles if necessary
If InfileHeading = "y" then
    Infile.SkipLine
End If

' Write field titles if desired
If OutfileHeading = "y" then
    Outfile.WriteLine "Domain" & OutfileDelim & _
        "Username" & OutfileDelim & _
        "Full Name" & OutfileDelim & _
        "Last Login" & OutfileDelim & _
        "Account Disabled"
End If

' Build the header portion of the HTML file
OutfileHTML.WriteLine "<html>"
OutfileHTML.WriteLine "<head>"
OutfileHTML.WriteLine "<title>"
OutfileHTML.WriteLine "Unused User Account Report"
OutfileHTML.WriteLine "</title>"
OutfileHTML.WriteLine "</head>"
OutfileHTML.WriteLine "<body>"
OutfileHTML.WriteLine "<b>"
OutfileHTML.WriteLine "Unused User Account Report"
OutfileHTML.WriteLine "</b><br><br>"
OutfileHTML.WriteLine "Report Generated On: " & TodaysDate
OutfileHTML.WriteLine "<br><br>"
OutfileHTML.WriteLine "These accounts have been selected because "
OutfileHTML.WriteLine "they have never logged into the domain. "
OutfileHTML.WriteLine "<br>"
OutfileHTML.WriteLine "Security Best Practices require unused "

```

```

OutfileHTML.Writeline "user accounts to be disabled or deleted."
OutfileHTML.Writeline "<br><br><hr>"
OutfileHTML.Writeline "<table>"
OutfileHTML.Writeline "<tr>"
OutfileHTML.Writeline "<td><b>Domain</b></td>"
OutfileHTML.Writeline "<td><b>Username</b></td>"
OutfileHTML.Writeline "<td><b>User's Full Name</b></td>"
OutfileHTML.Writeline "<td><b>Last Login</b></td>"
OutfileHTML.Writeline "<td width=15></td>"
OutfileHTML.Writeline "<td><b>Account Disabled</b></td>"
OutfileHTML.Writeline "</tr>"

```

```

' Cycle through all records in the input file
Do While Infile.AtEndOfStream <> true
' Read each line and split the file into individual fields
Contents = Infile.ReadLine
ReadArray = Split(Contents, InfileDelim)

' Load pertinent data into meaningful fieldnames
ReadDomain = ReadArray(0)
ReadUsername = ReadArray(1)
ReadFullName = ReadArray(5)
ReadLastLogin = ReadArray(10)
ReadAccountDisabled = ReadArray(15)

' Has the user ever logged into the domain
If ReadLastLogin = "never" Then
' Write an extract record for the user
    Outfile.WriteLine ReadDomain & OutfileDelim & _
        ReadUsername & OutfileDelim & _
        ReadFullName & OutfileDelim & _
        ReadLastLogin & OutfileDelim & _
        ReadAccountDisabled
' Write an HTML table entry for the user
    OutfileHTML.Writeline "<tr>"
    OutfileHTML.Writeline "<td>" & ReadDomain & "</td>"
    OutfileHTML.Writeline "<td>" & ReadUsername & "</td>"
    OutfileHTML.Writeline "<td>" & ReadFullName & "</td>"
    OutfileHTML.Writeline "<td>" & ReadLastLogin & "</td>"
    OutfileHTML.Writeline "<td></td>"
    OutfileHTML.Writeline "<td>" & ReadAccountDisabled & "</td>"
    OutfileHTML.Writeline "</tr>"
End If

```

Loop

```

' Close the files and end
Infile.Close
Outfile.Close
OutfileHTML.Writeline "</table>"
OutfileHTML.Writeline "</body>"
OutfileHTML.Writeline "</html>"

```

```
OutfileHTML.Close  
WScript.Echo "The unused account reporting process has completed."
```

© SANS Institute 2000 - 2002, Author retains full rights.

References

Childs, M., Lomax, P., & Petrusha, R. (2000). VBSCRIPT in a nutshell: A desktop quick reference (1st ed.). Sebastapol, CA: O'Reilly & Associates.

Eck T. (2000). Windows NT/2000 ADSI scripting for system administration (1st ed.). Indianapolis, IN: Macmillan Technical Publishing.

Norberg, S. (2001). Securing Windows NT/2000 servers for the Internet (1st ed.). Sebastapol, CA: O'Reilly & Associates.

Scambray, J., McClure, S., & Kurtz, G. (2001). Hacking exposed: Network security secrets & solutions (2nd ed.). Berkeley, CA: Osborne/McGraw Hill.

SANS Institute. (2000). Windows NT security step-by-step. Bethesda, MD: SANS Institute.

Berdahl, A. (2000). Practical assignment for GIAC Certification [On-Line Word Document]. URL http://www.sans.org/y2k/practical/Andrew_Berdahl_GCNT.doc

DiEugenio, D. (2000). Windows NT security step by step [On-Line Word Document]. URL http://www.sans.org/y2k/practical/Dave_DiEugenio.zip

Hackendorn, S. (2000). No title [On-Line Word Document]. URL http://www.sans.org/y2k/practical/Sherri_Hackendorn.doc

Hutchinson, G. (2001). Securing Windows NT 4.0 based networks [On-Line Word Document] URL http://www.sans.org/y2k/practical/George_Hutchinson.zip

Loser, L. (2001). Making the crackable password "non-crackable" [On-line Word Document]. URL http://www.sans.org/y2k/practical/Lois_Loser_GCNT.doc

McDowall, T. (2000). Developments in auditing NT information technology [On-Line Word Document] URL http://www.sans.org/y2k/practical/Tracey_McDowall.doc

Otis, B. (2000). SANS practical: Track 5: Windows security [On-Line Word Document]. URL http://www.sans.org/y2k/practical/brig_otis_GCNT.zip

Microsoft Corporation. (2000). Active Directory Service Interfaces (ADSI) 2.5 [Web page]. URL <http://www.microsoft.com/NTWorkstation/downloads/Other/ADSI25.asp>

Microsoft Corporation. (2000). How to enable strong password functionality in Windows NT [Web page]. URL <http://support.microsoft.com/support/kb/articles/Q161/9/90.asp>

Microsoft Corporation. (2000). Password Filter Programming Considerations [Web page]. URL http://msdn.microsoft.com/library/psdk/logauth/pswd_about_5z77.htm

Pedestal Software. (2001). NTSEC Windows 2000 [Web page]. URL <http://www.pedestalsoftware.com/ntsec/index.htm>

Quakenbush Consulting, Inc. (2000). Who has the keys to your business? [Web page]. URL <http://www.quakenbush.com>

Rudnyi, E. B. (1998). Name of built-in administrator [NTBUGTRAQ Newsgroup posting]. URL <http://www.chem.msu.su/~rudnyi/NT/sid.txt>

Security Software Technologies. (2001). LC3 [Web Page]. URL <http://www.securitysoftwaretech.com/lc3>

SystemTools.com. (2001). Somarsoft utilities – System reporting [Web page]. URL <http://somarsoft.com>

TP Information Systems. (2001). Password Policy Enforcer: Overview [Web page]. URL <http://www.tpis.com.au/products/ppe/default.htm>

TP Information Systems. (2001). Unused Account Ferret: Overview [Web page]. URL <http://www.tpis.com.au/products/uaf/default.htm>

© SANS Institute 2000 - 2002. All rights reserved. Author retains full rights.