WSRC-TR-2004-00011

Tracking the Inside Intruder Using Net Logon Debug Logging in Microsoft® Windows® Server Operating Systems

Christina S. Davis
Principle Engineer
Westinghouse Savannah River Company
Savannah River Site
Aiken, South Carolina 29808
christy.davis@srs.gov

ABSTRACT

In today's well-connected environments of the internet, intranets, and extranets, protecting the Microsoft Windows network can be a daunting task for the security engineer. Intrusion Detection Systems are a must-have for most companies, but few have either the financial resources or the people resources to implement and maintain full-scale intrusion detection systems for their networks and hosts. Many will at least invest in intrusion detection for their internet presence, but others have not yet stepped up to the plate with regard to internal intrusion detection. Unfortunately, most attacks will come from within.

Microsoft Windows server operating systems are widely used across both large and small enterprises. Unfortunately, there is no intrusion detection built-in to the Windows server operating system. The security logs are valuable but can be difficult to manage even in a small to medium sized environment. So the question arises, can one effectively detect and identify an inside intruder using the native tools that come with Microsoft Windows Server operating systems? One such method is to use Net Logon Service debug logging to identify and track malicious user activity.

This paper discusses how to use Net Logon debug logging to identify and track malicious user activity both in real-time and for forensic analysis.

KEYWORDS

Netlogon, Inside Intruder, Tracking, Windows® Server 2003, Windows® 2000, Logon

1. Introduction

Microsoft® Windows® networks are often the target of malicious computer hacker activity. While some companies invest in third party intrusion detection systems, others cannot afford them or do not have the people resources to manage them. They must rely on the native tools and features of Microsoft® Windows® Server 2003 and Windows® 2000, such as the security logs, to look for questionable activity.

While the security logs are useful, they can be unwieldy even in a small to medium sized domain. It can be difficult to obtain real-time data from the security logs. The security engineer needs additional methods of identifying malicious user activity. One such method is the built-in debug logging feature of the Net Logon Service. This paper discusses how to use Net Logon Service debug logging to track suspicious user activity in Windows® Server 2003 and Windows® 2000 networks. It also discusses methods for refining the data obtained from the Net Logon debug logs.

2. What is the Net Logon Debugging Feature?

In order to understand debug logging for the Net Logon Service, we must define the Net Logon Service. According to Microsoft®, the Net Logon Service is defined as "a user-mode service that runs in the Windows® security subsystem. The Net Logon service passes the user's credentials through a secure channel to the domain database and returns the domain security identifiers and user rights for the user. In addition, the Net Logon service performs a variety of other functions related to the user logon process, such as periodic password updates for computer accounts and domain controller discovery." [1]

2.1. Enabling Net Logon Debugging

The Net Logon Service maintains an activity log on the server in the directory, %systemroot%\debug\netlogon.log. By default, the log is empty. However, if one has a need to troubleshoot net logon activity, such as why user accounts keep locking out for no apparent reason, a debug value may be set in the registry of the domain controllers, to begin capturing net logon authentication data. We will use the debug log to track the intruder.

In our case, we wish to identify and track only the actual user logon activity, and so we will set the debug value to 0x20000004. This will weed out extraneous information such as Site location and other session information that we will not need for user tracking. We set this value by typing **nltest /dbflag:0x20000004** at a command prompt.

Complete information on how to enable Net Logon debugging and all of its associated flag options can be found on the Microsoft® web site [2].

NOTE: When the netlogon.log file grows to 19Mb, it is copied to the file, %systemroot%\debug\netlogon.bak, and a new netlogon.log file is created. You may consider backing up the netlogon.bak file each day and retaining the backups for a period so that you will have some history if you need to track activity over a period. 120 days is a good period to retain logs.

2.2. Contents of the Netlogon.log File

Let us examine the netlogon.log file with the 0x20000004 debug option set (Table 1).

```
12/09    07:11:10    [LOGON]    SamLogon:
Transitive      Network      logon      of
MYDOMAIN\CONNIEH from \\JACK-SPRATT (via
SERV3) Entered
12/09    07:11:10    [LOGON]    SamLogon:
Transitive      Network      logon      of
MYDOMAIN\CONNIEH from \\JACK-SPRATT (via
SERV3) Returns 0x0
12/09    07:11:10    [LOGON]    SamLogon:
Transitive      Network      logon      of
MYDOMAIN\CONNIEH from \\JACK-SPRATT (via
SERV3) Entered
12/09    07:11:10    [LOGON]    SamLogon:
Transitive Network logon of MYDOMAIN\ANNJ
from \\ANN-JONES (via SERV7) Entered
12/09    07:11:10    [LOGON]    SamLogon:
Transitive      Network      logon      of
MYDOMAIN\CONNIEH from \\JACK-SPRATT (via
SERV3) Returns 0x0
12/09    07:11:10    [LOGON]    SamLogon:
Transitive Network logon of MYDOMAIN\ANNJ
from \\ANN-JONES (via SERV7) Returns 0x0
```

Table 1: 0x20000004 Debug Option Set

We see the type of logon (e.g. network or interactive), the domain/workgroup name of the user, the user name, the machine from which the user is logging on, and the machine to which the user is attempting to connect. The code 0x0 means the logon was successful.

2.3. Parsing the Netlogon Log

When looking for hacker activity, we want to look first for potential scanning activity. We will see a great deal of failed logon attempts due to an unknown user name (code 0xC0000064) or bad password (code 0xC000006A) and in some cases, a clear footprint of the utility used to perform the scan (such as Nessus® or ISS®). By issuing a simple *findstr* command at the command prompt or from within a batch file, one can extract only the failed attempts, such as those cases where an unknown user name attempted to authenticate.

findstr /I "0xC0000064" c:\winnt\debug\netlogon.log >> **d:\save\failed.txt**

When we examine the output of the *findstr* command, we see that the local user "Administrator" on a machine named WIN2KPRO has attempted to authenticate or access many different machines within a few seconds (Table 2).

```
01/28    16:16:11    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
MYLAPTOP) Returns 0xC0000064
01/28    16:16:11    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
MYLAPTOP) Returns 0xC0000064
01/28    16:16:16    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
ANDYT) Returns 0xC0000064
01/28    16:16:16    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
ANDYT) Returns 0xC0000064
01/28    16:16:17    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
SALES1) Returns 0xC0000064
01/28    16:16:17    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
SALES1) Returns 0xC0000064
01/28 16:16:17 [LOGON] SamLogon: Network
logon   of   WIN2KPRO\Administrator   from
\\WIN2KPRO     (via     CHUCKY)     Returns
0xC0000064
01/28    16:16:20    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
BOSS) Returns 0xC0000064
01/28    16:16:20    [LOGON]    SamLogon:
Transitive      Network      logon      of
WIN2KPRO\Administrator from WIN2KPRO (via
BOSS) Returns 0xC0000064
```

Table 2: Failed Logon Data

The activity is suspicious because the WIN2KPRO machine is attempting to authenticate over the network with many different computers using a pass-through of the built-in Administrator account. Because the WIN2KPRO machine is attempting authentication to so many machines in such a short period and the failures are unknown account names, this could suggest the use of a scanning tool. In this instance, it could be someone looking for blank or easily guessed passwords as an easy means to hack into computers on the network. Also in this instance, the Administrator account does not exist on the target machines. It has been renamed. A similar search on bad password attempt (code 0xC000006A) would yield even more data for machines that have not renamed the Administrator account but where the scan attempted to use a bad password.

Using the *findstr* command to parse data, we can obtain a clear view of some scanning utilities, such as Nessus® (Table 3). The data are suspicious because we have one machine targeted by many user names and passwords in a short period. (Note: This Nessus® scan was executed from a Linux computer. While the true hacker may not use Nessus®, a footprint like this may allow you to identify when your own computer security folks are running internal scans).

```
12/09    02:05:15    [LOGON]    SamLogon:
Transitive    Interactive     logon     of
nessus\nessus  from  TARGET  (via  TARGET)
Returns 0xC0000064
12/09    02:05:16    [LOGON]    SamLogon:
Transitive    Interactive     logon     of
/etc/passwd\nessus    from    TARGET    (via
TARGET) Entered
12/09    02:05:16    [LOGON]    SamLogon:
Transitive    Interactive     logon     of
/etc/passwd\nessus    from    TARGET    (via
TARGET) Returns 0xC0000064
12/09    02:05:17    [LOGON]    SamLogon:
Transitive    Interactive     logon     of
```

```
../../../../../../../../etc/passwd\nessus
from TARGET (via TARGET) Entered
12/09    02:05:17    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from   TARGET   (via   TARGET)   Returns
0xC0000064
12/09    02:05:18    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from TARGET (via TARGET) Entered
12/09    02:05:18    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from   TARGET   (via   TARGET)   Returns
0xC0000064
12/09    02:05:18    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from TARGET (via TARGET) Entered
12/09    02:05:18    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from   TARGET   (via   TARGET)   Returns
0xC0000064
12/09    02:05:19    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from TARGET (via TARGET) Entered
12/09    02:05:19    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../../etc/passwd\nessus
from   TARGET   (via   TARGET)   Returns
0xC0000064
12/09    02:05:19    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../etc/passwd\nessus
from TARGET (via TARGET) Entered
12/09    02:05:19    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
../../../../../../../etc/passwd\nessus
from   TARGET   (via   TARGET)   Returns
0xC0000064
12/09    02:05:20    [LOGON]    SamLogon:
Transitive    Interactive    logon    of
Li4vLi4vLi4vLi4vLi4vLi4vZXRjL3Bhc3N3ZAo=\
nessus from TARGET (via TARGET) Entered
```

Table 3: Nessus Scan Footprint

We can also obtain data that suggest an ISS® scan of web servers was performed (Table 4).

```
01/30    19:06:40    [LOGON]    SamLogon:
Transitive     Network     logon     of
(null)\@@#ISS#@@    from    WIN2KPRO   (via
WEB03) Returns 0xC0000064
01/30    19:06:40    [LOGON]    SamLogon:
Transitive     Network     logon     of
WIN2KPRO\@@#ISS#@@   from   WIN2KPRO   (via
WEB03) Entered
```

Table 4: ISS Scan Footprint

3. Making Sense of the Data

Now that we have seen some examples of how useful the net logon debug log can be, we need to streamline the gathering and analysis of this data. We do not want to sift through mounds of data that may or may not indicate a problem. The best way to make the data gathering efficient is to write some simple scripts to parse the information.

3.1. The FINDSTR Command

To keep life simple, the *findstr* command is sufficient for quickly parsing the data. The *findstr* command can be executed from a command prompt or within a batch file. For example, you may choose to write sophisticated scripts that dump the data into a database or that send alerts via email if there are many occurrences of logon attempts from one machine within a few seconds of each other.

A typical simple search looks like this:

findstr /I "0xC0000064" c:\winnt\debug\netlogon.log >> **d:\save\failed.txt**

This command parses out all failed attempts caused by an unknown user name from the netlogon.log file into a text file. We start with the code unknown user name (0xC0000064) because the hacker is likely to try to access built-in accounts such as the Administrator account first. Since best practices dictate that we rename the Administrator account to something else, we will see logon failures if someone attempts to use this account name.

Let us say we have looked at the failed.txt file and have determined that there is suspicious activity coming from a machine name WIN2KPRO. Perhaps we want to see all activity associated with this machine. We would execute the command:

findstr /I "WIN2KPRO" c:\winnt\debug\netlogon.log >> d:\save\win2kpro.txt

The results of this query may key us into additional user names, such as NESSUS852812095, as well as bad password attempts and successful logons. We will search on those names and will continue to find perhaps even other machines this user is using. In other words, once we start pulling the string, we may begin to see the whole seam unravel into a clear picture of hacker activity on our Windows® domain.

If you use *findstr* commands in batch jobs, you can execute a series of them at one time. You can also schedule them using Task Scheduler and have the results emailed to you or placed on a server somewhere. All of this requires no additional software and uses only the native tools on the server.

3.2. A True Story

I was involved in a case (it prompted me to write this paper) in which penetration testing was being performed in my company. I had no idea what specific tests were being conducted on my systems only that auditors were present somewhere in the company. I discovered an attempt to use an unusual user name within my network and began looking at my logs for information on the user name.

Through a series of searches for the user name and subsequent searches for all computers and all user names that were related to the original account, my colleagues and I were able to identify more than ten computer names, and a dozen user names being used to test the security of our Windows® network. We identified the computers and some of the methods used in the test against us (Nessus®, ISS®, and nmap scans, as well as some well-known exploits).

By doing DNS and DHCP lookups, we were then able to determine IP and MAC addresses and ultimately the actual physical location of the auditors. We were also able to notify other employees that their systems were being scanned while they were actually being scanned.

In this case, because a penetration test was being performed, there was no damage. Through the use of simple native feature in Windows®, we identified the auditors.

4. Don't Forget the Security Logs

You may be thinking that this is all well and good, but you want alerts when this stuff happens so that you do not have to look at these logs unless there is a reason. To that I would say, do not forget the security logs! Make sure you are auditing all logon events and all account management events at a minimum. Turn on auditing of system events, object access, and policy changes as well. Be sure to retain your logs for at least 120 days. Be sure to back up and retain the netlogon.bak file. Set up timed scripts to send you information from the netlogon.log file. All of these methods can help you watch your Windows® network.

Hackers on the inside are patient, and you may find you need to retrace the hacker's steps after the fact. Set up alerting or use third party monitoring products capable of alerting you to a series of failed logon attempts. For example, Adiscon® EventReporter® is an inexpensive product that is useful for sending alerts from the event logs. You may wish to refine the alerting process to send you alerts only if the same alert occurs say more than five times in one minute also. You can get very creative with how you set up your scripts and alerts.

5. Analysis! Analysis! Analysis!

Setting alerts will help minimize the time spent chasing windmills; however, you still have to be vigilant! You, the engineer, still have to look at the logs yourself with regularity. Human analysis of the data is the single most important thing you can do. The alerts can clue you into a potential problem, but nothing can replace human diligence in analyzing the logs. The adept hacker will be patient, and this will require you to be patient also. The hacker may not be as obvious as running scans that hit dozens of machines at once. He may be stealthy and hit one or two machines at a time making it much more difficult for automated processes to notice his activity.

Get to know your data through and through. Run scripts daily to look for trends. If you know what is normal for your environment, you will be able to spot the abnormal more easily. As you get to know your environment, you will be able to streamline your scripts to identify changes in trends. For example, if your net logon log files usually take three days to fill up, and you begin to see them filling up daily, you will need to investigate.

6. Conclusion

While intrusion detection systems are extremely valuable and companies should invest in them where at all possible, there are circumstances in which this is not feasible. These companies still need to have some way of identifying hacker activity in their Windows® networks. Native tools, while sometimes unwieldy, can be useful. This paper has sought to provide a simplistic yet powerful option to the security engineer for identifying hackers on their networks. It has also sought to stress the importance of vigilance and analysis of one's own internal environment. By using the Netlogon log and other native logs and indicators of the Windows® Server operating systems, the security engineer can track and identify some forms of malicious activity in the Windows® network.

7. References

[1] Microsoft® Corporation. (2003). Quick Guide for Finding Tools, Retrieved 12/1/2003 from the Microsoft Technet web site: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/windowsserver2003/proddocs/standard/glossary_srv.asp

[2] Microsoft® Corporation. (2003). Enabling Debug Logging for the Net Logon Service (109626), Retrieved 12/1/2003 from the Microsoft Support web site: http://support.microsoft.com/default.aspx?scid=kb;en-us;109626

8. Acknowledgements

Thanks to Jason Oliver for help in refining the data analysis process and for writing scripts.

Thanks to Bruce Page for his mentorship, leadership, and encouragement.