I'm a Lumberjack and I'm OK: System Logging

Prof M. E. Kabay, PhD, CISSP-ISSMP[1]

One of the more arcane aspects of running a multiuser computer system is *logging*. No, not the kind of logging that the Monty Python character sings about in his famous (or notorious) song about British Columbia[2] – this kind of logging refers to keeping records of many different types of events on the system. For example, operating systems on mainframes, minicomputers, and some LANs (or on LANs equipped with appropriate monitoring software) can record events such as

- System Boot
- System Shutdown
- Process Initiation
- Process Termination
- Session Initiation
- Session Termination
- Invalid Logon
- File Open
- File Close
- Invalid File Access
- System Console Messages
- Network Activity
- Resource Utilization
- CPU Activity
- Disk Space
- Memory Consumption.

Functions

Logging serves many functions. Basically, logging keeps a record of who was doing what to which data on what systems at which times. Simply knowing that logging is taking place and contribute to self regulation: knowing that actions are monitored and reduce harmful behavior. Log files provide information for controlling the system; for example, system administrators and managers can limit access to certain resources in response to observations of abuse and they can change parameters and

[1] Sincere thanks to MSIA Administrative Director Elizabeth Templeton for proofreading.
[2] Jones, T. M. Palin, & F. Tomlinson (1969). I'm a Lumberjack and I'm OK. Monty Python Show.
Official video at < http://pythonline.com/node/18116931 >.
Lyrics at < http://www.metrolyrics.com/im-a-lumberjack-lyrics-monty-python.html >.

resources in response to trends. Log files also serve a fundamental purpose in forensic investigations.

Volume & Archiving

In the early days of computing, system managers had to decide how often to close log files; disk space was expensive. However, these days, disk space is not much of an issue. In 1980, as I've mentioned in other commentaries, a 120 MB hard disk the size of a washing machine cost U$25,000 – about U$75,000 in current value and roughly U$625/MB. In contrast, today as I write in early 2009, a 1 TB Maxtor hard disk the size of a book costs less than $200 or roughly U$0.0002/MB (~1/50th of a penny per MB). So the price was ~3,276,800 times greater 30 years ago, which incidentally works out to about a 61% drop per year in compounded change.[3] So the main issue today with respect to logging onto disk is simply preventing data loss if the system or the logging process crashes; one approach is to disable buffered I/O and force immediate writes to the disk. Since one can and should dedicate a storage device exclusively, performance should not be a problem as long as the logging process is separate and does not hold up what is being monitored.

The system manager must decide how long to archive log files. Usually, there are legal requirements which can guide the establishment of definite policies. These policies must be monitored and enforced to avoid serious problems such as contempt of court citations if records are deleted in violation of those requirements. For legal and functional purposes, log files must be archived in environmentally sound and secure storage facilities; normally, these are off site just like those for other system backups.

Analysis & Exception Reports

Each operating system can have particular variations in log file structure; you should look for log-file analysis tools that are specific to your environment. A search engine such as GOOGLE provides a wealth of references for the string *"operating system" "log file analysis"*; for example, at the top of the 13,400 hits located in 0.94 seconds in my search, there were dozens of products listed right from the start of the results. The *Exefind Software Search* service alone which was located by the GOOGLE search listed five pages of names and descriptions of log-analysis utilities.[4]

Such analytical software is necessary because it is usually impossible to examine all the records - there could be millions. One needs to be able to break out the unusual events. Using appropriate software, one can set filters to scan for unusual conditions. Some systems define baseline events – that is, the norm – and spot unusual ones using statistical methods. Human beings can scan the exception reports and look for patterns; more sophisticated software systems can use artificial intelligence (AI) to help spot patterns and anomalies.

AI systems may use accumulated observations and statistical methods to spot outliers that signal unusual and perhaps dangerous events; for example, suppose that no more than one user logon in 1,000 over the last year has used an ID from the accounting department between the hours of midnight and six in the morning – so why did "Ralph" try to logon at 3:30 in the morning overnight? And if the real "Ralph" from accounting has not had to try his password more than twice

[3] (1/3,276,800)^(1/30)=0.6065
[4] http://www.exefind.com/log-file-analysis/

in the last 1,523 logons, then how come this "Ralph" tried 18 passwords at 3:30 in the morning before giving up?

Chargeback Systems

In the early days of computing, users normally got charged for every aspect of resource utilization; for example they might accumulate charges of $0.00001/disk I/O and $0.00002/process initiation. Users received the itemized bills showing their resource utilization – typically monthly; such bills promoted efficient use of resources. I remember personally being alerted by the Vice President of Operations at Mathema, the big data center where I worked in the 1980s, that one of our clients was generating reports that were three to four times more expensive in the last few months than in previous years. He told me to find out why and to help reduce the costs.[5] Searching the log files for changes in their resource utilization, I noted that their total disk I/O charges had been climbing rapidly in recent months; further analysis showed that both online access and batch job I/O during the reports on orders had gone out of line. I investigated their databases and found that they had not repacked their most-used detail data set for a long time. As a result, records for line items in individual orders were no longer placed in contiguity according to the *primary index*[6], order number. Thus printing out lists of all the order details forced the database to make multiple reads over the data set. I told the client database administrator to repack the data set so that related records would end up in the same data block, reducing the number of I/Os in reads using the order number by the blocking factor (records per block). The clients' report costs for their monthly global report dropped from $1,200 back down into the $300 range.

Chargeback systems can also play a direct role in improving system security by increasing user involvement. Any user being charged more than expected can alert system management to the anomaly, which may be the result of hacking or of malware.

Protecting Log Files Against Alteration

If log files are to be useful in forensic work, they must be protected against unauthorized alterations. There are many ways of doing so including checksums, digital signatures, and encryption. Checksums are hash totals generated using a standard algorithm which are appended to each record; any change to the record that does not recompute the checksum using the right algorithm will allow

[5] Students may be surprised to learn that the data center manager would want to *reduce* income from the charges to a client. However, we took a long-term view of our relationship with our clients; our job was to help them make the best of our services so that they would stay with us as long as it made sense and then buy computer systems from us (we were an authorized HP distributor) when it was time for them to become autonomous.

[6] A detail data set might have many indexes, but one of them was supposed to be identified as the most frequently used one and could be designated the primary index. In a special operation called a data set condense or repack, all the records would be rewritten so that they would fall into place according to the increasing value of their primary index. Thus if the order number were the primary, as many records as possible for order #1245 would be placed one after another – perhaps in the same *block* (the unit of physical I/O; i.e., what got read into a memory buffer by the disk driver). Thus reading all the records for a specific order number would be measurably faster from a packed data set because the blocking factor would determine the total number of I/Os required to read the entire file. If there were, say, 100,000 record packed 10 records per block, only 10,000 I/Os would be required instead of 100,000. Since the drives of that time could not reasonably complete more than 40 I/Os per second, the difference between 100,000 I/Os and 10,000 I/Os was the difference between 41 minutes just for physical I/O and 4 minutes. CPU overhead would normally add at least as much time, so the difference in job time could be 80 minutes for the unpacked data set versus 8 minutes for the properly packed data set.

immediate identification of the corrupted record. In addition, if the checksum is computed by including the checksum from the preceding record, an attacker would have to recompute the checksums for every single record following an insertion, deletion, or modification of the original records.
Once a log file has been closed, one can use public key cryptography to sign the entire file with an authorized signing key. No one without access to the signing key will be able to generate a valid digital signature for a modified log file.

Memory Dumps

Large systems often made use of memory dumps, which in the 1980s were still frequently called core dumps.[7] These core dumps were files containing the entire contents of active memory (RAM) and were very useful for debugging and forensics. There were two types of dumps: some were obtained through diagnostic utilities (debuggers) in real time and others were captured after abnormal system shutdown using special utilities.

System level DEBUG utilities gave administrators complete access to RAM, thus allowing a total bypass of the system security. These were extremely powerful and therefore dangerous tools; they allowed users with root privileges to copy or alter any portion of memory, to access system tables by name and make changes such as stopping processes, altering priorities and so on. It was critically important to control access to these tools; often, it was forbidden for any one user to initiate the system debug without the presence of a colleague.

Early core dumps were so tiny by today's standards that they could actually be printed to paper; for example, an HP 3000 Series III maximum memory size in 1980 was 2 MB. A perfectly normal PC today can easily have a minimum of 2 GB of RAM – quite impossible to print and read manually.[8] Even in the 1980s, however, we much preferred to work directly on electronic versions of the core dumps (usually mag tapes) and do our analysis on our terminals using the analytical utilities.

A core dump can be a major security vulnerability. It contains cleartext versions of vast amounts of confidential and possibly of encrypted data; it can included I/O buffers such as input from keyboards and files or output to displays and files. It would be a disaster to release a core dump to unauthorized personnel. There is even a serious question about whether vendors should be permitted to see memory dumps.

---

[7] Here's one of my favorite stories about tech support from the days when I was in charge of the *Phone In Consulting Service* (PICS) at Hewlett-Packard in the early 1980s. Here's an excerpt from a real conversation with a customer from that time:

Customer:	The HP3000 crashed 10 minutes ago.
Mich:	So did you take a dump?
Customer:	[long pause] Yes, but I don't see what that has to do with the computer.
Mich:	NO, a *core* dump!
Customer:	Oh – a *core* dump – yeah, sure, I did that.

[8] Suppose we use 256 characters x 88 lines = 22,528 bytes/page. Then 1 MB would take ~46 pp and 2 GB would take ~95,000 pp. If the manual inspection rate were 1 minute per page (and that would be fast), it would take ~66 days to read the dump once.

This informal overview should get you interested in finding out about logging in your own case-study organization or industry. Ask your system and network managers about their practices and discuss the costs and benefits of current standards and possible enhancements with them.

And we haven't even touched the issue of application logging. . . .