Interested in learning more about security?

SANS Institute InfoSec Reading Room

Slogging (syslog-ging) through the Mud

Defense in depth -- This concept is the basis of true Information Technology (IT) security. Without multiple levels of security it is folly to believe that our IT systems approach anything close to security. In this paper I'll be focusing on what I feel are some of the most important mechanisms of defense in depth: logging and auditing. Logs are often thought to be things that are consulted after an incident. I suggest that logging is at the root of a much broader tree that encompasses such things as Intrusion Detectio...

Slogging (syslog-ging) Through The Mud

Michael Sullivan
Assignment Version 1.3
4/9/2002

Abstract

Defense in depth -- This concept is the basis of true Information Technology (IT) security. Without multiple levels of security it is folly to believe that our IT systems approach anything close to security. In this paper I'll be focusing on what I feel are some of the most important -- but often taken for granted – mechanisms of defense in depth: logging and auditing. Logs are often thought to be things that are consulted after an incident. I suggest that logging is at the root of a much broader tree that encompasses such things as Intrusion Detection Systems (IDS) and Forensics. It is intimately related to security policies.

Logging takes many forms and is done differently on the various hardware/software platforms in use today. In order to make this a manageable subject I plan to limit this exploration to the Unix system log "*syslog*" and the program-tool *swatch,* a program that facilitates the use of logs. *Syslog* will often be known by different names; depending on the flavor of Unix being run. My plan is to address general logging and auditing concepts and then to explore Unix *syslog* and *swatch*.

To be sure, the logging on Unix goes far beyond this one log system. Depending on the version of Unix being used, there are logs that focus on users – who's logged in/out, who has tried to login but was not allowed, even commands run by every user; plus more. The *syslog* however is virtually universal to all Unix flavors and has tremendous capabilities when properly configured.

I will also touch on forensics as it relates to logs. This will be a cursory exploration as the subject of forensics is a field in and of its own.

Logging

Every year, at one point or another, I think how nice it would be to keep a personal–life journal. I would spend some time at the end of each day reviewing and assessing what I had done. Later these "logs" would be available to me to use as a reference of what was going on in my life at a particular time. Alas I never do it; it's just not something I want to do just before going to sleep. I think doing a good job at logging and auditing a computer system is not unlike this – it's boring. But… in this case, it's got to be done.

The Unix O/S will do a significant job of logging even if you do nothing but boot the system with the default settings. However, at the very minimum, you should know *what, how and where* the information is being logged. After that there is a group of concepts

that should be addressed. These concepts include logging according to a security policy, "secure logging", where to put the logs, how to back them up, … There are many more.

Logging and auditing have some side benefits. One such benefit is facilitating future system requirements planning. Another – and in my opinion a very big one – is that it will help promote a healthy system; it is less likely that a healthy system will have a rushed patch performed. Modifications that get done in a rush are worrisome -- if not statistically more prone to bugs/security-exposures.

Security Policy

The security policy should address logging and auditing very explicitly. Such things as the types of activities to be logged, how the logs are to be backed up and retrieved, …

In order to achieve the U.S. Dept. of Defense C2 Audit certification what must be done is very clear and can be used to drive the development of the policy. Not only does it indicate what must be logged it also deals with how the data is handled – who has access, how the log files are written, … -- and how it's to be protected. [1]

Secure Logging

As if it were not enough to log; the logging process itself must also be secure. The integrity of the logs must be addressed since they can be susceptible to deletion or modification by the attacker. Secure logging can and must be done in a number of ways.

The access permissions to the log files must be controlled. Only users that are documented as trusted should be allowed access; and any changes must be documented as well with the "who, what, and why" of the action. Encryption of the data in the log files will add another layer of security.

One of the most straight forward methods, that in and of itself adds multiple layers, would be to keep the logs files on a machine other than the system that is being monitored with the logging facility; better yet would be multiple separate and remote systems. This logging can be done easily over a network; perhaps to a machine that has the absolute minimum of services running to support just the network and file system. The remote logging systems can even be running a different OS than the machine that the logs are being maintained for. The network connection should have it's own set of security mechanisms as well. If several systems are to be monitored, a central logging site and master log will help guard against an attack done to more than one host. Such an attack might have been missed if they had been audited as individual logs. At the very minimum logging to a different system represents an increased challenge to an intruder; he/she must gain access to two machines in order to stay covert

A Logging Denial of Service must be guarded against. An attacker could cause activity that fills the logs to the point where the logging system stops. This could be remedied by having the system perform a live notification, perhaps by pager, of the system admin that the logs have filled or are about to be filled.

Probably the biggest exposure to secure logging is that a savvy attacker will alter or erase the logs to hide the fact that the system has been compromised. There are two ways the logs can be altered by the attacker. First is to use the computer systems themselves to gain write authority. This form of compromise can be guarded against by use of a write once form of media such as a printer or CD-R. The second form of this exposure is if the attacker has physical access to the logging system; if that is the case there is almost nothing that will guarantee the security of the logs.

It should be noted that the improper disposal of any physical media such as CD-R, paper, even hard disks that contain prior logs could be a security exposure as well, and needs to be guarded against.

Real-Time Notification

My first exposure to computer security issues came about some number of years ago when I read the book **"The Cuckoo's Egg"** by Clifford Stoll [7]. I remember reading the part where he enables a paging system to notify him when the attacker would log on; I thought this was so cool. Today this is done in some places as a matter of course. It enables people with IT security responsibility the very powerful ability of not only responding in a timely manner to an intrusion, but to other system problems as well. Conditions such as a system crash or failure in the physical environment of the computing facility (e.g. air-conditioning) can be guarded against. This feature might allow for much better chances at catching the intruder in the act since you essentially know about it as its happening [8]. The filtering/auditing tool *swatch* we will describe latter has extensions to allow just this type of function.

Problems With Logging

"There Ain't No Such Thing As A Free Lunch" (TANSTAAFL) and logging is certainly no exception. There are many costs and problems associated with logging. These problems include – but are not limited to – the space the logs will consume on your storage, the time a person spends looking at the logs, having so much data that it is essentially meaningless because you cannot find the needle in the haystack. Some logs can also contain such security exposing information as account names, numbers and passwords.

Characterization

If you don't know what a normal/typical log looks like it will be difficult to identify what a log that contains malicious entries looks like. As a first level action a system administrator with sufficient experience should be employed to identify and document expected log entries. After that a period of time should be spent collecting data on the typical activity the system is logging. If for some reason the expected activity is not what the typical activity is indicating then there needs to be an investigation and the differences explained; if necessary the expected log entries document should be updated. The process of typical characterization should be repeated according to the security policy. This process is sometimes known as taking a baseline.

Backing-up Logs

This concept is obviously not limited to logs. However there are a few particular aspects that should be taken into account when doing backup for logging. As I mentioned before the log files will have an impact on size; they grow very fast with facilities such as sendmail running. It would be wise to enable file compression in the log backup process. There will also probably need to be a rotation of the media used for the backups. This action should be accounted for in the security policy.

Many of the Unix systems have built in applications for helping with backup of the file system. These can be employed to help with the logs. Also important for the backup file is a naming convention that will allow for knowing the dates and particular system the log is for.

Forensics

Forensics, the application of scientific knowledge to legal problems, could possibly represent the most exciting aspect of logging. I think I would find it intense to participate in the actual prosecution of a criminal – certainly more exciting than jury duty. Logs play a very important roll in computer forensics.

The most important characteristic of a log with respect to forensics is its integrity. "A computer that logs various kinds of network activity needs to have the entries undeletable and unalterable, even in the event that an attacker gains access to the logging machine over the network." [2]

In addition to the computer log files, it's important to keep hand written documents, which will include the chain of events and how the evidence is handled. The documents should include: [3]

- Name of system
- Owner of system
- Date / time
- Actions taken
- What data collected
- Who was notified
- Who has access
- Who, when and why info disseminated
- What was submitted to legal consul

Best Practices

- The logs should be characterized (baseline)
- With multiple systems and multiple logs to be audited it is recommended that a time synchronizing mechanism such as Network Time Protocol (NTP) be employed. [4]
- There should be sufficient personnel to audit at the frequency identified in the security policy guidelines.
- Physical security data (paper logs, CD-Rs, …) need to be disposed of in a proper manor.
- The Logs should be guarded against obsolescence. Logs that can't be read by current technology are useless. This could be addressed by keeping equipment that can read the media – this requires space, skill… -- or by transferring the logs to new media, which your current equipment can use – an exposure of data corruption – TANSTAAFL.
- Program-tools used to perform, audit and manipulate logs should be from reliable sources
- Logs should be backed up according to the security policy
- Make sure logging is restarted after reboot. [1]
- Logs should be preserved in such a manner as to support forensics

Auditing

Auditing, the process of formally examination and verification of the log files, is just as important as keeping logs. If this is only done after the fact of a system compromise then you may have missed an opportunity to save time, money, aggravation, … (you pick the metric) by having prevented the compromise in the first place.

Now, it's unlikely that anyone ever said that examining a log file is their idea of a job to aspire to; nonetheless it must be done and there are some tools to aid in the process; we'll examine one of them shortly. With all the suspicious activity to be found either from an Internet connection or in a large group of systems it's unlikely that the defensive security counter measures of a computer are not being challenged. I know not much more than a day goes by that my firewall does not get challenged with suspicious probes. I actually find it a good feeling to see these rejected accesses – I know that the firewall is working. Knowing that your defenses are working is just as important as knowing about when your counter measures have failed. Auditing the logs is the only way to get this knowledge.

While the auditing tools that filter the information from the logs facilitate auditing – and perhaps make the whole process possible given that there can be so much information in a log file – it is also necessary for periodical examination of the raw log files. This will help to see if you are missing any thing that might be going on but not obvious due to the filtering of the audit tools being applied.

Auditing essentially turns the logging process into a form of Intrusion Detection System (IDS). With the addition of such tools as TCPWrappers, this IDS like feature can be extended to the network. It must be noted that false entries in a log must be guarded against as well. [6].

SysLog

Syslog is the fundamental logging facility of the Unix system. The file produced contains data on a very broad set of facilities of the host system(s). Any program can have information logged by sending a message to *syslog*. The message will contain four parts:

- Program Name
- Facility
- Priority
- Log message

There are many Unix facilities that have built in logging to *syslog*. Though not complete – and some facilities may not be on all version of Unix – a sample list is:

- kern – the kernel
- user – user processes
- mail – the mail system
- daemon – system daemons
- auth – systems that require authorization such as passwords

Priority (levels) include:

- emerg – emergency condition
- crit – critical condition such as hardware failure
- warn – warning
- debug – debugging information

User written shell scripts can use the *logger* command to send messages to the *syslog*. An example format/syntax for the *logger* command is:

logger –t tag(program name for example) –p facility.priority "descriptive message string"

The functional operation of syslog is configurable by entries in a configuration file (usually /etc/syslog.conf). These entries determine what kinds of events will be logged. The format of these entries will vary from Unix flavor to Unix flavor; my Red Hat system has the format:

facility.priority [; facility.priority] destination

As with the logger command syntax, there is the ability to control what facility and level of priorities to log. It should be noted that there is a hierarchical aspect to the priority token. Any entry that indicates a facilities priority (e.g. alert) will also log any priority that is higher. So that if the priority is *warn* then *crit* and *emerg* messages will also be logged.

After the types of messages to process is set the daemon then starts listening to three sources – /dev/klog, /dev/log and UDP port 514. These sources represent messages generated by the kernel, messages generated by processes running on local machine, and messages sent over the local area network respectively.

The forgoing discussion is by no means intended to be a complete description of the syslog or its configuration and use. This will always depend on the specific flavor of Unix being use.

Swatch – Simple Watcher

This program-tool is intended as an auditing aid for system administrators; it can be applied to perform several actions on logs such as filtering and monitoring. *Swatch* was written by Todd Atkins, a UNIX system administrator at Stanford University. The program is available for free down load from several sites. I obtained it from ftp://ftp.stanford.edu/general/security-tools/swatch**.**

The Program is written in PERL and as such will need the PERL libraries installed. The libraries came as part of the installation of my Red Hat 7.2 installation. It will also be of great benefit to have knowledge of regular-expressions; any good book on PERL will have a section on how to write them. It is my intent to give you a feel for how *swatch* might be applied to facilitate the auditing of syslog and thus improve security of your system; it is not intended to be a substitute for the formal documentation that already exists for the program. [5,6]

The author had four operational goals for *swatch*:

- Easy to teach
- Take action on its own
- Allow for user defined action
- Allow for Reconfiguration without stopping and restarting by hand

Swatch's fundamental function is log filtering. Through application of regular expression matching, the logs files can be audited to expose highly suspicious entries and ones of specific system interest.

Operationally it can be applied in three modes:

- Single pass through a file
- Examine messages as they are appended to a file
- Examine the standard out of a program

Swatch's fundamental control mechanisms are the command line options it is started with and the associated configuration files read as input.

Command line arguments control the behavior and what files *swatch* will use as input. The command line options are as follows:

- -c config_file (file that will contain the regular-expression/action statements)
- -r restart_time (automatically restart at specific time)
- -f filename (file to examine if performing a single pass audit on that file)
- -p program_name (examine the standard out of a program)
- -t filename (examine lines as they are appended to a file)

Here is an example command line:

swatch -c /root/sys_log_config/ -t var/log/messages

This would cause *swatch* to run using a set of regular-expression matching commands read from the file ***/root/sys_log_config/*** and apply them to lines as they are being appended to the file ***/var/log/messages.***

Regular expression pattern matching is at the heart of the filtering/action behavior of *swatch*. Almost without exception, all users will be familiar with the very simple regular expression of "*.pdf", which is a pattern that matches any text string that ends with the characters ".pdf". This form is used very frequently when listing the files in a file system directory (e.g. "ls *.pdf"). The discussion that follows is predicated on the use of *swatch* to monitor the appending of lines to a log file.

The configuration file has a series of lines whose format, at its simplest syntax, looks like:
/pattern/[,/pattern/] action[,action] – a pattern(s) to match and an action(s) to perform.

Upon startup the program reads a configuration file. The pattern in the pattern/action lines tells the program what text pattern in the appended line to check for. If the line being appended has text that matches that pattern then the action that is associated with that pattern will be executed. The actions allowed are:

- Bell (sound a beep from the consol)
- Echo (print message to controlling terminal)
- Ignore (do nothing; process next line)
- Mail (send mail message to a user)
- Write (essentially same as Mail)
- Pipe (send matched appending line as input to another program)

- Exec (run a command on the system; selected fields from the matched line can be passed as arguments to the command)

As you should recall from the section on *syslog* the log entry will have four parts Program Name, Facility, Priority, and Log message. If one wanted to filter for all *kern* facility messages, regardless of priority, and sound a bell and echo the message to the terminal, the config file line here will cause that to happen:

/kern.*/ echo, bell

To ignore the logging for a program such as sendmail and the faxspooler programs the following line will enable that behavior:

/sendmail/,/faxspooler/ ignore

The documentation that can be found on-line, in books, as well as what is downloaded as part of *swatch* goes into considerable depth concerning the configuration file syntax and functionality and is the best source for this information. [5,6]

I feel the most powerful feature of *swatch* is the examination of messages as they are appended to a log file and having the config file associate attacker-like behavior with a response action such as dialing a pager with an error message. There is a small set of useful programs available with the *swatch* download; "callpager" is one of them. This function can be extended beyond strictly security usage to help in such areas as system availability exposures (OS crashes, equipment failure, …).

One first-time user-type problem I ran into was that the program when run without any command line parameters expects a set of default conditions such as a configuration file in the home directory. The error message, when quickly read looks like it's telling you it couldn't find the program *swatch*; I would have preferred a syntax/usage error message.

Swatch's output is directed to standard-out.

Conclusion

One thing I think that maybe hard to remember is that security, while absolutely necessary, is not in and of itself a consumable product; you can't eat it and it does not exist for its own sake. It is inevitably classed as overhead of a business. As such, it will always take extra work and expense; not only to do a successful job at it but also to justify that amount of work and expense. This is true even if it's just our own home systems we administer. Therefore it is important that we be as efficient as possible in how we implement security. I hope that the ideas and information presented in this paper help you in just that respect.

References:

[1] "Manage logging and other data collection mechanisms" May 1, 2001; URL: www.cert.org/security-improvement/practices/p092.html; (4/9/02)

[2] "Secure Audit Logs to Support Computer Forensics"; URL: www.counterpane.com/audit-logs.pdf; (4/9/02)

[3] "Collect and protect information associated with an intrusion"**;** URL: www.cert.org/security-improvement/practices/p048.html; (4/9/02)

[4] "Time Synchronization Server"; 02/02/02; URL: www.eecis.udel.edu/~ntp/index.html; (4/9/02)

[5] Todd Atkins and Stephen Hansen "Centralized System Monitoring With Swatch" *LISA Conference 1993*; www.oit.ucsb.edu/~eta/swatch/lisa93.html; (4/9/02)

[6]. Simson Garfinkel & Gene Spafford. Practical UNIX & Internet Security, Second Edition, April 1996. Published by O'Reilly & Associates, Inc. Cambridge, MA, ISBN 1-56592-148-8.

[7] Clifford Stoll "The Cuckoo's Egg"; October 2000; Published by Pocket Books; ISBN: 0743411463

[8] "Using swatch for log analysis" 11/14/2000; URL: http://www.linuxsecurity.com/tips/tip-27.html; (4/9/02)

Last Updated: August 19th, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Chicago 2017	Chicago, ILUS	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VAUS	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Adelaide 2017	OnlineAU	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced