Global Information Assurance Certification Paper

Log survival kit
(How to implement a central log facility in a mixed environment using open source tools)

By João Martins

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4b, option 1

Abstract

As a direct result of a security problem verified in our network, and the ad-hoc manner how it was solved, we fill the necessity to centralize the logs produced in our network. We promote a search over the INTERNET to find tools that could help us to accomplish that under two major conditions: those tools must be from the non-commercial world (open-source; GNU-GPL; freeware) and the Central Log Server (syslog server) will be a machine running Linux. This document relates our findings. Special attention was dedicated to the questions related with the integration between machines running the MS-Windows operating system and machines running a flavor of UNIX (Linux and Tru64 UNIX).
We started a search to find a solution in a field that we know from the beginning that could exist serious problems concerning security (syslog). We strongly believe that the benefits of this solution overcame by large the risks involved.
We also emphasize that this solution could and should be seen as layer in the context of Defense-in-Depth.
We suggest the use of iptables in order to minimize the security questions related with the absence of trust and any type of control of what has to be accepted between the device and the collector.
We suggested also the use of a network timeserver to solve the questions about timestamp and synchronization between machines.
We have selected ***"Eventlog to Syslog Utility"*** to process and transfer the logs in realtime from MS-Windows machines to the syslog running on Linux.
On *NIX machines we will use syslog, since it is widely implemented.
To each tool we try to identify its major benefits and drawbacks (if they exist). When there are several tools available to make the same function, we try to explain ours preferences.
Finally, to get more profit from the solution, we proposed the use of a monitor/logparser in order to be notified if an event occurs that matches an expected behavior.
At the end we believe that we reached a fair solution, a bit different from others that we found and are enumerated in the bibliography, which solves our initial problem and adds security to our network.
We are aware of the existence of some weaknesses in the solution reached, and have identified solutions to some of them. The existence of a unique (central) point of failure, its not really a technical problem, but a financial one. If there is enough money, it could be easily solved.

Before: A little story

Once upon a time I found that someone dropped a mailbomb into the mailbox of one of the students of the faculty where I work. This event, ie, a large number of small messages generated an overwhelming task which sends our mail servers to a load never experienced before.
As a result from the lack of procedures to deal with such problem we started a few things to minimize the damage (stop sendmail; clear the mail queue; etc.). When we finally had a minimum control over the situation, we started to identify the source of the attack. We found that the mail was initiated in one of machines that we are in charge, but the user was identified as nobody. Since our apache (http) server was configured to run as user nobody, we suspected that it could be used a program executed by the http server (cgi script). So therefore we accessed the apache logs and found the execution of a script, by the time that the mail started, with a very suspicious name (the attacker didn't use very care at this point. One point for us zero points for him). Since we are administrators and as a result of that have full access to the home directories of the users of our systems, we try to see the code but, at that time, it had disappeared (points equal).
Our next move was to identify who was logged in the machine where the requests come from, since, for our good sake, they belong to our network. Not very good news, the logs (MS-Windows) file size was very modest so, by that time (2 days later) they were lost (points: 2-1). Fortunately we had placed an experimental logging facility that had recorded a login in one of the 3 machines involved in the attack (points: 2-2). The final, to abbreviate, had a positive end to us since we identified the author of the attack and he was punished (points: we won!).

Can we get any major lessons from this story? First of all we didn't have a plan on how to deal with such attack. In true we don't have any plan at all. So we where acting in an unordered way. Secondly, and as an indirect result of the first, we lost some time and when we tried to access the logs (evidence) they weren't there anymore. By that time we knew already the importance of the logs but never had such a necessity, and, to apply the Murphy's Law, the things got bad just when we didn't want to.
As a direct result of this problem, coupled with our knowledge that the logs were too much important to be lost in just a few days we intended to conduct a study to overcame with a solution on how to implement a centralized log facility in a mixed environment (windows, unix and linux) using open source tools (or at least freeware). That's what this document tries to address.

Definition of the concept

Before we go further lets try to define the central concept of this paper: the log. Accordingly to Scala, Inc we can define log as: "log file: a text file consisting of timestamped status and error messages, detailing the operational history of a given piece of software." This definition emphasizes some important issues related to logs:

- a text file. They don't talk about databases for instance, it's simple a text file. This reminds us possible problems with ownership, permissions, size, etc.
- timestamped. Obvious, but not so obvious, since it depends of the accuracy of the clock where the stamps come from. We will turn to this subject.
- Error messages. I would prefer to say only messages.
- Piece of software. That's the source of the message. Another question arises: every piece of software produces such output? Can we receive them?

We must keep in mind the definition and also the questions posed here. Those questions direct us to the limitations of the process by itself. Another pertinent question relates to the time when the event occurs. As we mentioned the message is marked with a timestamp, but that timestamp relates to the time when the system that collects the messages was notified. This means that it could happen that something is going on and, for some reason, the system was not informed or was informed at a later time.
Never is too much to say that all the processes and procedures have limitations, and knowing them are a first step to manage something (tools; procedures; plans; etc.) to overcame with.
One useful application of the logs is to know what is going on or have been in yours systems. Other is to use them as evidence if you have to deal with some problem: technical (debugging, for instance) or forensic.

Why Log

It seems clear that logging could help the system administrator in identify some problems during or after the occurrence. Although this is true and can be accomplished manually in a standalone machine, it could be a very difficult and tedious task in an environment with several, hundreds or even more machines. There's where centralized logging comes on. We can enumerate also other benefits from this approach:

- The central log server could be, by reducing the number of services that it offers and other measures of security, improved in its security;
- Backups of the data received could be made in a easy and efficient manner;
- Tools (IDS; Monitoring) can be applied to those logs in order to warn the administrators to take the appropriate action (if applied).

Context of application

Our scenario is a mixture of several hundreds of machines (servers and clients) using mainly three types of operating systems: MS-Windows 2000; Linux (Slackware) and Tru64© UNIX (New HP) (Figure 1).

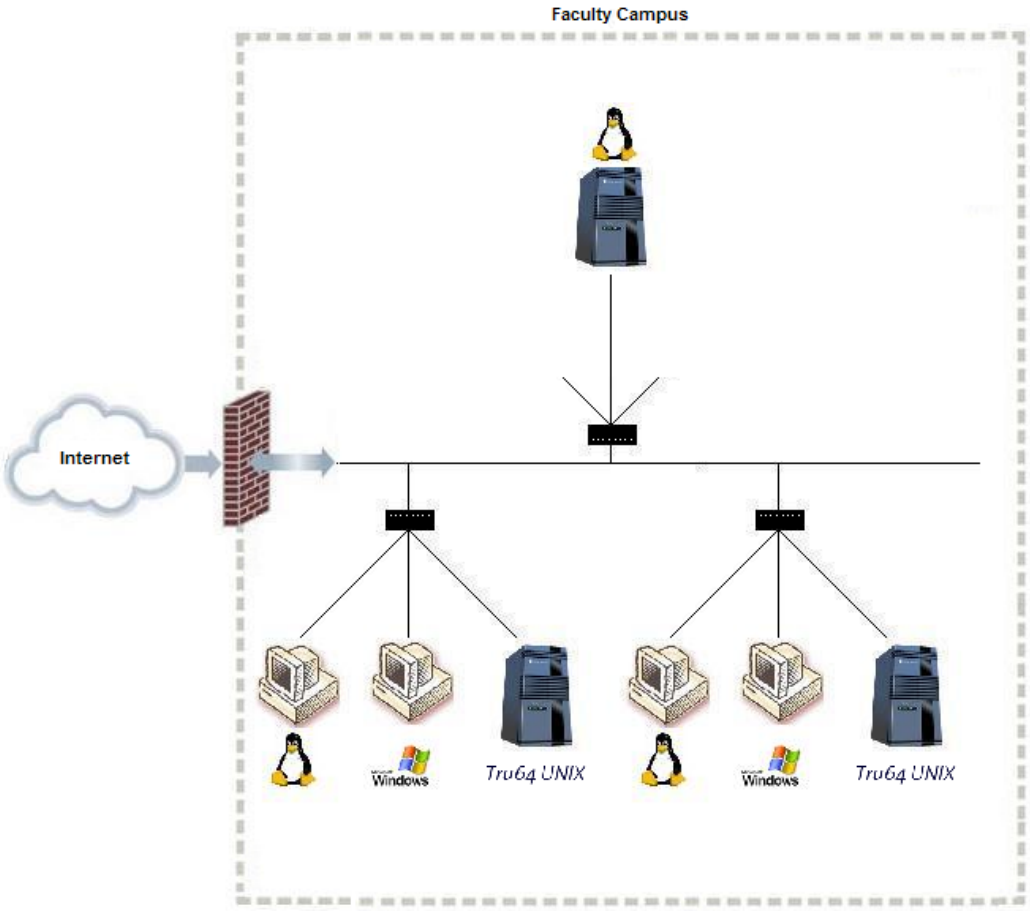

Figure 1 – Simplified layout of the network

Our job is to administrate all those machines, and it is guaranteed that we have sufficient privileges to accomplish the task.
We have identified the need (let me say: in the worst manner) to have:

- information centralized (server located in our private network);
- put in place mechanisms to process the information collected in order to be warned if something is going on;
- have a larger window of time over the information;
- the possibility to make backups and restores of the information received by the system.

Because we are an academic institution, where the knowledge about how the things are done is appreciated, coupled with the always short budget we intended to find a solution in the field of non-commercial (open source, GNU-GPL, freeware) solutions.

By this time, we decided not to include the network devices (routers and switches) because of the large amount of data that those devices could produce. We decided to find one solution, and in a later time, certainly already with more experience, include those equipments.

A few notes about the Syslog protocol

Since, as you already probably noticed, we intend to use the syslog protocol, we think that it is appropriate to give a little explanation about how it works. We cannot make a fairly resumé about what is said in RFC 3164 so, we strongly recommend its reading. For those of you that are in a rush to continue, here's a brief explanation.
The first thing to know is "The protocol is simply designed to transport (…) event messages" [RFC 3164, 1- Introduction]. There are a few results from this affirmation. One means that the UDP (User Datagram Protocol) is the protocol used to transport those messages (logs) between machines (Device->Collector). This protocol is not connection oriented, resulting that there is no guarantee of message delivery and also that the order of the messages could be altered. These two issues are very important and must be kept in mind.
There is a reserved port (514) where are expected connections from this protocol (syslog). The RFC indicates that there is no obligation to use the same port from the sender, but is a good practice (RECOMMENDED, is the word). Finally, the total length of packet "MUST be 1024 or less" [4.1].
The syslog protocol is recognized by the redactors of the RFC as having several security problems. They don't call breaches to these problems since the protocol wasn't designed with that purpose in mind. At that earlier time, the developers where only concerned with the transport of messages over the network. We must be aware of those problems, that's why we reinforce our recommendation to read the RFC 3164, especially the section 6 (Security Considerations).
At this time we think that we can leave the warnings aside and take a quick tour over the program, particularly in terms of configuration files. The syslog configuration file (syslog.conf) presents the following syntax:

Selector <tab> *action*

Where selector represents a program (in the original: Facilities) coupled with the message severity level. For example:

mail.debug /var/adm/syslog.dated/mail.log

Debug messages coming from the mail program, will be written in the file mail.log.
Notice here the separation between the selector and the action (always tabs, never spaces) and the full path to the file.

We reproduced here the Syslog severity levels accordingly to Nemeth [1995]:

Level	Approximate meaning
Emerg	Panic situations
Alert	Urgent situations
Crit	Critical conditions
Err	Other error conditions
Warning	Warning messages
Notice	Unusual things that may need investigation

Info	Informational messages
Debug	For debugging

[Extracted from Nemeth [1995], page 209]

Notice that there is a hierarchic relation between the levels of the messages. The highest level [lower Numerical Code] indicates more importance. Therefore if you indicate that you want to be informed about facility.y, the system will report all the messages coming from that facility level y and above.

Solution reached

We would like to emphasize that could and certainly exist other solutions (some of them are referred at the end of the document) to this problem[1]. This one that is going to be present here represents ours findings and our believe to what could be one good solution considering the characteristics of the environment were they are to be applied, the restrictions mentioned before and the tools available.

The Main Server Configuration

We designate by Main Log Server the server that is responsible for receiving all the logs. By convenience, it is represented in the upper part of figure 1. As we mentioned before, we have chosen the Linux Operating System to install on this machine. We could choose another OS, but our decision was mainly taken because we found more degrees of liberty in terms of configuration settings (IP stack size; IPTables) that were offered by this OS when compared to others and also found more documentation on doing so.
The role and position of this machine are nuclear to the system. If it becomes compromised all the process will be poisoned. Plus, by this time, it represents a single point of failure. In the future we can add redundancy to this machine. This redundancy could be achieved in two ways: forwarding a copy of the logs received to another machine and/or providing a mechanism of high availability, that starts the service on other machine when the main server stops to respond.
One important question related to logs besides in the absence of control and thrust between the sender and the receiver[2]. This means that if we don't apply any control, any machine on our network will be able to send logs to the central server. With little effort we can think about the damage that could arise from that fact. To minimize this thread we intend to configure IPTables[3]. IPTables are native in Linux, and can be tailored to accept connections from well known sources (IP's) and even by protocol (in our case it will be UDP).
Another problem that we must address relates to the timestamp. The timestamp registered on the logs are related to the clock of the machine that is

1 Other approach could be saw in Garbretch [2002]

2 Altough there is a workgroup under the IETF (Security Issues in Network Event Logging (Syslog)) working on those issues. More inform ation about the progresses can be found at: http://www.employees.org/~lonvick/index.shtml.

3 [2] "Iptables is used to set up, maintain, and inspect the tables of IP packet filter rules in the Linux kernel."

receiving the logs, therefore we recommend that all the machines involved in the process have their time synchronized with a network time server.
Another security issue related with the Main Log Server besides on the users that can log on that machine and the services that are available. We recommend that they should be reduced to the minimum. Fewer services imply a less concern with attacks, probes, exploits, etc. fewer users simplify the maintenance and the control.
By this time you could have already guessed that we intended to use syslog as our logger[4]. On this machine we will configure syslog (syslog.conf) keeping the default values. This is a good starting point, later with more experience you can make improvements: add some levels, delete others, forward some, keep others local. We recommend not to try all things at the same time. If anything goes wrong it's more difficult to discover what it was.

The Clients Configuration

Linux and UNIX

Since syslog is implemented in Linux and Tru64 UNIX, to forward all the logs to a central repository (Main Log Server) all we have to do is to edit the configuration file (syslog.conf) and add a line like this:

```
*.*             @main_log_server
```

This line tells to the syslog to send all the messages of all the severity levels (*.*) to the main log server. We think that you should use the IP instead of the name or if using the name put the IP in the hosts file. This reduces the time to process the message, since there's no need to resolve the IP, and minimizes the problems that could exist if the DNS goes down. After editing the file and made the alteration, send a HUP signal to the syslog daemon (syslogd) , in order to force it to re-read the configuration file and act accordingly. You could make a test if it works using the logger[5]:

```
prompt> logger Program1 Message
```

Watch in the appropriate file on the main log server to see if there is that entry. If it is there, everything is working as expected.

Windows

It is a little more difficult to configure the windows clients, since the way that the logging system was designed doesn't comply with syslog. Usually, if registering is done, windows writes its logs (Event Log Service) entries to a file in a proprietary format that can only be viewed using the Windows Event Viewer

4 Notice that there are other implementations of the syslog protocol. If you're interested in more information about this subject, try a start from: http://www.loganalysis.org/sections/syslog/syslog -replacements/index.html

5 Logger is a program that exists on *NIX like SO that allows you to make entr ies in the system logs

application. Moreover it doesn't provide mechanisms to forward those logs to a central facility. So we must find a way to solve those two problems. Fortunately others had faced already the same problem and solved it offering (GNU General Public License) to us the solution[6]: Eventlog to Syslog Utility. It comes in the form of a service and watches for the logs in real-time. "When a new message appears in the log, it is read and formatted, then forwarded to a UNIX syslog server" [extracted from the program home page].

Making more profit from the logs

So far we reached a point where we have already centralized our logs. This means that we fulfill some if not all of our initial purposes. We can make regular backup of them; they are more secure than dispersed by several different machines/OS; we can transfer them to databases in order to simplify the analysis and make other conclusions (for instance answering that question: how many failed attempts to login have we experienced last January?). It will be greater if we could also perform some automatic (due to the large amount of data, it can't be done manually) analysis of the logs, preferable in real-time, gathering signs or signatures of problems, be notified of them and be able to respond. It seems that we are talking about an IDS system. That's in fact what we are looking. We were lucky again. There are a few tools, offered under the GNU GPL, freeware or open source, that performs similarly to what we were looking for (incident.pl; LogDog; LoFiMo; Log Tool. To find more just take a look in www.freshmeat.net). Our choice was LogDog. This tool, offered under the GNU GPL, is similar to another one well known in the *nix world – swatch. We prefer this one because it has very similarity with the other one but has nowadays more support and continues to be developed. This program works in realtime (contrary to others, for instance logcheck[7], which only runs from time to time) looking in log files for strings or expressions that conform with the ones registered in the configuration file and when find one starts an action that can be a mail message a win popup, etc.
The major problem when configuring those kinds of tools derives from the fact that you must know, *a priori*, what you are looking for. This means that to identify a certain type of attack you have to know what kind of messages it will appear on the log files in result of it. This is true, but we could count with the help of more experienced people to configure our files instead of starting from scratch. At later time, looking to the log files with the help of a tool like colorize.pl that permits, as the name indicates, to add color to the logs you could find other strings or expressions that you find useful to have a reaction. Another useful reading about this is "Top 10 Syslog-based Signs You've Been Hacked" by Tina Bird [2002].

Weaknesses

6 We reinforce the idea that this is a solution. There could be others. From the non -commercial software world we could pick up also NTsyslog or BackLog for the same purpose.

7 http://www.psionic.com/abacus/logcheck

We believe that this solution addresses the answer to several of our problems. However we recognize that are a few less robust points. The first is a direct result of the protocol used by the syslog – UDP – that is a connectionless protocol and therefore doesn't have guarantees of delivery.
Another point already enunciated is represented by the Central Log Server. In case of a very prepared attack this machine could be of primary attention. If it fails, by knowing reasons (attack) or by hardware or software failure, it will compromise all the solution.

Future work

This solution was primarily addressed to unify the logs, especially those coming from windows machines. However it has capabilities to be enlarged to receive logs from other sources. Several network devices have the capacity to produce logs (usually representing a large amount of data). We intend to study their formats and the way how they could be sent to the central repository. We are aware that in those conditions we must have in the central node hardware with high processor capabilities and also a very high IO throughput.
On our readings we found references to the possibility to watch also the logs made by IDS and monitoring tools (for example, produced by NAGIOS). Several of those tools already have alert functions, but have their logs registered in a central location and archived could also be a good idea.
Another thing that must be reviewed and could be a limitation of this solution if not solved, relates to the programs or services that are used by our organization and doesn't use the syslog to produce their logs. If it is a nuclear application, keeping it aside could represent a serious risk.

Glossary

"Defense in depth is the concept of protecting a computer network with a series of defensive mechanisms such that if one mechanism fails, another will already be in place to thwart an attack. Because there are so many potential attackers with such a wide variety of attack methods available, there is no single method for successfully protecting a computer network. Utilizing the strategy of defense in depth will reduce the risk of having a successful and likely very costly attack on a network" McGuiness [2001].

Resources

Bird, Tina [2002] – Top 10 Syslog-based Signs You've Been Hacked
http://www.loganalysis.org/presentations/syslog_sans_webcast.pdf

Carnegie Mellon University [2000] - Configuring and using syslogd to collect logging messages on systems running Solaris 2.x
http://www.cert.org/security-improvement/implementations/i041.08.html

Cole, Eric et al [2002] – SANS GIAC Certification: Security Essentials Toolkit (GSEC). SANS Press.

Curry, David A. [1992] – Unix system security: a guide for users and system administrators. Reading, Addison-Wesley.

Garbretch, Frederick [2002] - Practical Implementation of Syslog in Mixed Windows Environments for Secure Centralized Audit Logging.
http://www.sans.org/rr/casestudies/mixed_win.php

Hines, Eric [2000] - Complete Reference Guide to Creating a Remote Log Server
http://linuxsecurity.com/feature_stories/feature_story-64.html

Murphy, Richard [2001] - Syslog and Netsaint: How to Integrate Centralized Logging with Centralized Monitoring.
http://www.sans.org/rr/logging/syslog.php

Nemeth, Evi et al. [1995] – UNIX system administration handbook [2nd ed]. New Jersey, Prentice Hall.

McGuiness, Todd [2001] - Defense In Depth.
http://www.sans.org/rr/securitybasics/defense.php

Scala, Inc. [no date] - Glossary of Terms. Definition of log file.
http://www.scala.com/definition/log-file.html

Smith, Randy Franklin [2000] - Archiving and Analyzing the NT Security Log.
http://www.win2000mag.com/Articles/Index.cfm?ArticleID=9043

Weinberg, Joshua [2000] - Centralized Logging with syslog.
http://www.visi0n.net/sysadmin_mag/html/v07/i10/a2.htm

Products/Organizations listed in the document:

BackLog
http://www.intersectalliance.com/projects/BackLogIIS/index.html

colorize.pl

http://freshmeat.net/projects/colorize.pl/?topic_id=245%2C92

Eventlog to Syslog Utility
https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys/

Log analysis in freshmeat:
http://freshmeat.net/browse/245/?topic_id=245

LogAnalysis.org
http://www.loganalysis.org/

LogDog
http://caspian.dotconf.net/menu/Software/LogDog/

Looper Event / Alert System
http://edgesolutions.ca/article.php?sid=7
http://software.freshmeat.net/projects/looper/?topic_id=148%2C862%2C150%2C152%2C864

Logsurfer
http://www.cert.dfn.de/eng/logsurf/

Kiwi SyslogGen 2.0.2
http://www.kiwisyslog.com/products.htm

Nagios
http://www.nagios.org/

NTsyslog
http://sabernet.home.attbi.com/software/ntsyslog.html
“Reliable Delivery for Syslog”, RFC 3195
http://www.ietf.org/rfc/rfc3195.txt

Swatch
http://freshmeat.net/projects/swatch/?topic_id=245%2C43%2C862%2C152

“The BSD Syslog Protocol”, RFC 3164
http://www.ietf.org/rfc/rfc3164.txt