# The Difficult Art of Managing Logs

We all know that *logs* are a critical component of any software element, from operating systems to the tiniest applications, of a ICT system. By *logs* we mean in general the information that a software component reports about itself and its activities: what actions it has taken, which potentially troublesome situations have appeared, who has invoked it and so on.

Logs have always been used as the primary means for debugging applications and the way for applications to report errors and warnings.

Another very important use of logs, the one we are interested in here, is to record who has used the system and for doing what. We can call these *Audit Logs* but what distinguish them from the normal logs is most often the information we want to extract from them and not their form or content. It is true that there are specialized Audit Logs generated for example by some operating systems or some particular applications, but today these are more exceptions than rules.

Logs are becoming every day more important not so much as debugging tools, but for security reasons. Potentially they are a fundamental source of information for understanding and then being able to control what is happening in our ICT systems.

Moreover, more and more legislation is introduced which requires to produce, analyse and securely store logs to be able to prove to the authorities that ICT systems are correctly managed and to provide information about possible unlawful activities. The key words for all of this is that logs have to be produced, analysed and securely stored.

Before considering what this implies, let us add the final ingredient, the logs' sources. Today ICT systems are complex and distributed environments, they are comprised by many different components with different purposes. For the sake of simplicity, we can group them in

1. workstations
2. communication/network components
3. servers.

Within a single host, logging can be done at different levels by different components. Again we can generically classify them as

1. operating system logs
2. data (-base) logs
3. application logs.

Obviously, each host can be source of more than one log of each type.

## Using Logs

Why are we so interested in logs? Let us make an extreme but not unrealistic example. Suppose a judge orders a company to provide detailed information about all activities done by a particular employee a particular day within the company ICT infrastructure, because this person is suspected to have perpetrated a crime using it. It is of course of interest of the company to show

a) that reasonable security policies, procedure and practices are in place to prevent misuse of its resources

b) that controls are in place to prevent or records activities which violates the policies or the law.

In this way the company should be able to satisfy the judge request and show that it has no responsibility in the actions of the employee since it has put in place measures to prevent and track these actions.

In practice the company has to find within the logs produced that day by all ICT components, information like:

1. at which workstation the employee has logged-in/out and at which time;
2. which applications he has used and at which time;
3. which data he has read, created, deleted or modified and when;
4. which other hosts both internal and external to the company network he has logged-in, which operations he has done, etc.

The list can be obviously much longer, this being just a simple example.

More generally, securely managing ICT systems requires to be able to trace potentially risky events. Typically interesting events include:

a) the modification of software configuration, both application and operating systems

b) the modification of software components

c) the access to credential databases

d) the access to systems with privileged credentials

e) all actions done with privileged credentials

f) the access to any kind of *sensitive* data

and so on. To be able to trace the events we need at least to record *Who* has done *What* and *When*. Doing this is more difficult than what it seems.

An event on a system can be caused by an action by an user on another system, and this user can be logged in with different credentials on a third system, his workstation. For example a transaction in a database is usually triggered by an application server which is accessed through a web-server by a browser on a user workstation. We need to find enough information in the logs to be able to trace back the transaction in the database to the physical person sitting at the workstation. It should be clear that this is not always an easy task.

But with all this in mind we should go back now to our problem: how to produce, analyse and store logs.

## The Syslog Inheritance

Syslog is the most common protocol/format for transferring and archiving logs. Syslog has many useful features which have made it very successful over the years:

1. platform-independent
2. easy to implement
3. adaptable
4. simple format

5. logs are in human-readable form

and so on. But it has also serious drawbacks which make it in practice not acceptable for our purposes. Among them, the following two are the most important:

1. in the best of old-time traditions, communications and log transfers are best effort and not authenticated, which means that log lines can be lost or modified without notice (new versions of syslog have been proposed which address some of these shortcomings, but they are not very popular yet);
2. being in a human readable and almost completely free format, syslog messages are usually not structured and practically impossible to parse automatically, this makes it impossible to analyse automatically large quantities of data.

If we now consider these shortcomings within today's complex and multi-faced ICT environments, it is immediately obvious that syslog-like logs cannot practically be used for our purposes. They still absolve their original purpose, when stored locally and manually analysed to debug software problems.

Instead, what we need to do to achieve our purposes, is:

1. collect large amount of logs from many different sources
2. store all of them in a single repository which we can efficiently search
3. correlate the logs and extract the information we look for.

These are not easy tasks and we need to discuss them in turn.

## Correlation and Data Format

Correlation between logs produced by different sources and type of sources is fundamental to achieve our purposes. In the examples presented above, to establish a complete time-line of all ICT activities, we need to correlate information from operating systems, network devices, applications etc. Having large and complex systems which produce huge amounts of logs, this requires to parse logs automatically and it is only possible if logs are all structured in the same format.

As an illustrative example, instead of the syslog free-format human-readable, suppose that all logs from all types of sources are formatted as follows: each record is terminated by a record termination character which does not appear in the record, and is composed by fields separated by a field separation character which does not appear in the fields. Then assume that the first field of each log record is the time-stamp of the event and the second field the user-id of the user responsible for the event, and so on. In this way we could, for example, efficiently select all actions of a user in a day in chronological order.

But to be really efficient, not only the format of all logs should be the same, but there should be also a unique taxonomy of events so that the same event seen by different applications is classified in the same way and reported with the same name (or event number).

Of course all of this can be possible only if there is an international standard widely adopted and implemented. (Here we are only suggesting the creation of a standard, which of course would be quite different from the simple illustrative example we have just described.)

Considering the number of operating systems, databases and applications which all will require to be modified, we are not so optimistic that such a standard will be created and adopted any time soon.

## Contents Above All

But the most important point of a global standard for logs' format is not so much for their formatting but for their contents. Today logs usually contain information good for debugging and problem-solving but not for tracing user operations or auditing. Thus often information is incomplete and it can be impossible, independently from the format, to match the same event as logged by two different sources. Indeed in some cases the only information that two log records from different sources share is the time-stamp, and this assuming that the two sources have synchronized clocks.

A standard should mandate that each log record reports a minimum set of information about the event, and that the events are classified in the same way by all sources. This will make it possible to identify and trace the event through different log sources.

Actually this is all what we are after: to be able to extract consistent and meaningful information from the logs.

Obviously after we have stored logs in the correct format and with the correct contents, we will need to develop intelligent search engines which will be able to:

1. create reports of all significant events
2. on request, trace single events in all details.

## Dimensions at Last

We should not forget that ICT systems are huge and complex, and all components will produce logs. Already today it is not unusual to produce hundreds of Giga-Bytes of uncompressed logs per day. And we need to collect and store them securely.

In some situations we could be required by legislation to digitally sign and encrypt the logs and keep them for years but still offering the possibility of searching through them and extracting information in reasonable short time:

1. Where are we going to put all this data? How many Tera/Peta-Bytes of disk space we will need?
2. How can we efficiently find what we look for in this vast amount of data?

Even if we cannot say that we have all the answers to these questions, recent advances both in storage and database technologies can help us already today.

For example, specialized compressed column-stored databases achieve very high compression of column-homogeneous data, such as formatted logs, offering at the same time extremely fast search engines. At the same time specialized storage components are starting to be offered on the market which are optimized for data which is written once, never modified, read many times and deleted in bulk after a specified retention time. This is exactly what we need to store our logs.

## Secure Storage

Since the information stored in our logs can be used to prove to a third party, a judge for example, what has actually happened even many years after the event, we need to store securely the logs and prove that the information they contain is *original* and has not been modified in all this time. In forensic terms, we need to establish a *chain of custody* for our logs. This is easy to say but again almost impossible to achieve exactly.

First of all, what is the *original* log? By this we intend the information contained in the logs or the data which represents it?

How can we prove that the information is not modified from the moment when it is created to the moment in which it is used?

What we can technically do is to prove that some data has not been modified, for example by digitally signing it at the moment of the data creation. But consider the simple example of a database which logs in a DB table and that the table is periodically exported to a file. We would say that the original information is represented by the data in the DB table which we should *sign* in this format, even if it is not very clear what this actually means technically. But then we should export this information to a file in another data format, which then will be transferred through various systems potentially being modified in the transfer, and finally loaded in another format in another database, the log storage database. At the final point we could be able to guarantee that the data is not modified once written in the log database, but it seems difficult to prove that the information is exactly the same as the one which has been created originally in the first database.

We have two possibilities to approach this problem. The first is to be able to prove a priori that all transformations of the data representation of the original information do not modify it. This is theoretically appealing but very difficult in practice. The second way, as difficult in practice but less theoretical, is to implement reversible procedures for transferring and archiving logs. In this way we should be able in any moment to reconstruct the original form of the log and prove that it has not been modified.

Of course the proof of not modification requires something to prove it. We have already mentioned digital signatures, the alternative would be an a priori proof of non modifiability of the information. Digital signatures seems at first sight to be a practical solution, but any expert in cryptography knowns that today they are very expensive operations which can add an impossible high load to a system. For obvious reasons, production servers are usually running at almost full capacity. Already the fact of logging information can add a heavy burden both in CPU and I/O utilization to a server. Requiring it also to digitally sign each log record is just impossible, and this has been proven in practice in many occasions.

A practical way out which only partially solves this problem is to collect a bunch of logs in an intermediate system, digitally sign them in bulk on this system, and then forward the logs for loading into the log database. In this way we can prove that logs have not been modified starting from the intermediate system, after the digital signature. To prove that a log record has not been modified we need to extract from the log database all records belonging to the same bunch as the record in question, and verify the digital signature. This procedure is today possible in practice, but it falls short from a full proof of non modification of the logs.

Today to securely manage ICT systems we need to be able to access the audit information which logs can give us. Unfortunately this information is very often not produced and even if it is produced it is not formatted, analysed and stored in an efficient and useful way. We need to understand and solve many issues, from practical to methodological and theoretical, to be able to manage the logs as it is required by today's complex ICT environments.

Andrea Pasquinucci

PhD CISA CISSP

http://www.ucci.it/