



assuria

**Forensic Soundness  
of Log Data**

## 1 Introduction

The term “forensic soundness” is often used when discussing log management and SIM/SIEM requirements and solutions. However, there are two key problems with the use of the term in this context. Firstly, the strict definition of the term varies significantly in different jurisdictions and secondly, it is a heavily misused (and abused) term.

This White Paper discusses the term, its implications for log collection, management and analysis, and discusses relevant product features in Assuria Log Manager (ALM) as an illustration. We assume a basic familiarity with Log Management and SIM/SIEM architecture; further information about this, and on the topics covered herein, is available from Assuria on request: please contact [info@assuria.com](mailto:info@assuria.com).

*This Note covers the use of log data in a legal context. This does not constitute legal advice: Assuria does not provide such advice.*

## 2 Forensic Soundness

“Forensic soundness” is not a well-defined concept and can mean different things to different people. The word “forensic” implies use in a legal context, and we shall assume here that this means presenting log data in a tribunal or court of law. However, there is no definition of “forensic soundness” in UK law (or in several other jurisdictions): it is purely a marketing term. An important criterion is whether the log data are admissible in court. Different jurisdictions and courts have different rules of admissibility: some admit almost anything but with differing weight; in other courts admissibility is all-or-nothing. For example, “in the [UK] criminal court we’d argue about admissibility [of log data] all day long.”<sup>1</sup>

General guarantees of forensic soundness or legal admissibility are therefore not possible. A log management solution can, though, provide support in the following areas that should support an argument for the admissibility of log data:

1. Providing logs in their original form.
2. Preventing undetectable addition of log data.
3. Preventing undetectable deletion of log data.
4. Preventing undetectable modification of log data.
5. Demonstrating chain of custody of log data.

The remainder of this document discusses these areas in more depth and describes the features in ALM that provide them.

## 3 Original Form

Any processing that modifies log data introduces scope for doubt about the data’s integrity. One would expect to have to demonstrate that the processing has not materially altered the log content, particularly if we are using the logs to demonstrate that a particular event did not occur (and is therefore absent from the log). Partly for this reason, more recent versions of various compliance standards have emphasised the importance of retaining the “original” log data.

---

<sup>1</sup> Personal communication with a UK barrister.

Many vendors claim retention of “original” or “raw” logs, but they usually normalise the logs into some common format (such as syslog), and claim retention of the original data. They’ve normalised *form* but not normalised *content*. However, they’ve still processed the original log data, which from a forensic perspective provides room for doubt about log data integrity, and necessarily cannot handle content in the original log that can’t be represented by the normalised format.

The collection and storage infrastructure within ALM has been designed to retain all logs in their original form, unchanged. It makes no assumptions nor imposes any requirements on the format of the log data: for instance, ALM can collect network packet captures and application logs that contain screenshots. ALM provides functionality to read, interpret and analyse the various log formats that it can collect, but it always retains the original log, unchanged and digitally signed, irrespective of any follow-on processing.

## 4 Addition Deletion and Modification

If we look at the general problem of *integrity*, the challenge is to prove that the collected log data are exactly those data that existed on the source system, nothing more and nothing less. The solution to this problem has three parts:

1. What integrity protections are applied (see 4.1 below).
2. When/where the protections are applied (see 4.2 below).
3. What guarantees the protections provide (see 4.3 below).

Contrary to the claims of many SIEM vendors:

- *[Cryptographic] hashing of received log data does not provide integrity.* Cryptographic hashing is an important *part* of some approaches, but by itself it does not solve the integrity problem.
- *Encrypting logs (e.g. by transmitting them over an SSL or TLS connection) does not provide integrity.* It provides confidentiality, which is a separate problem.
- *In general it is impossible to prevent addition, deletion or modification of log data (depending on the determination and skill of the attacker).* Such alterations can, however, be made difficult to achieve, and impossible<sup>2</sup> to achieve undetectably.

Integrity problems may not necessarily be malicious: they can arise due to operational problems such as insufficient disk space, file corruption, disk failure or network transmission errors. While some of these may at times be unavoidable, a log management or SIEM solution should protect against them where possible, and characterise the scenarios where integrity will be compromised.

### 4.1 Integrity Protection Mechanisms

Assuria Log Manager (ALM) uses *data source plugins* to collect log data in a manner that protects integrity. These use appropriate techniques to rotate logs whilst avoiding race conditions to prevent loss of events generated while the log is being rotated. Where this proves difficult, ALM errs on the side of integrity rather than latency: for example, if a log file is temporarily locked then ALM waits.

---

<sup>2</sup> Modulo certain events such as a private key compromise.

ALM's collection processes use sequence numbered, hashed and digitally signed datasets. *Hashing alone is insufficient to guarantee log data integrity:* cryptographic signatures or message authentication codes (MACs) are required.

Datasets can be signed at each point in the collection chain (i.e. on the source machine by the ALM Agent and by the Collector that receives datasets centrally). The ability to sign datasets at source is a major advantage of an agent-based approach to Log Management.

When transmitting to a central point, ALM spools logs until they are successfully received centrally. This ensures that integrity isn't compromised by network outages. The size of the spool is limited by the available disk space; the agent will generate warnings if this falls below a configurable threshold.

ALM's network protocol for log transfer confirms successful receipt before deleting log data from the source system. It has features to detect and recover from corruption in transit: under standard statistical assumptions it will suffer an undetectable bit flip in transit approximately once per yottabyte of log data. (Achieving perfection here is impossible but we can make the failure probability arbitrarily small, as demonstrated by Shannon<sup>3</sup>).

ALM network connections are mutually authenticated: the central infrastructure will not accept logs from sources that aren't legitimate (and won't send logs to destinations that aren't legitimate, but again, that's confidentiality rather than integrity).

## 4.2 Where to Apply Integrity Protection

ALM is an "Agent Optional" solution: it supports agent-based and remote collection of logs. However, if log data integrity is important we would strongly recommend the use of agents. In these instances the ALM Agent is installed on the machine generating the logs: this means that logs are sequence-numbered and digitally signed as close to the source of the data as possible, and certainly before any network hop.

Each side of the agent-based versus agentless debate has advantages and drawbacks (hence ALM's support of both approaches). From an integrity point of view, agents have significant advantages:

- By being co-located, an agent is best-placed to interact with the application or system generating the log data and therefore avoid any race conditions when collecting the logs.
- Integrity protection mechanisms are applied earlier in the collection chain (i.e. closer to the log's source) by the log management system.
- The log management system controls both ends of the network connection between the agent and the central infrastructure and can therefore apply appropriate authentication mechanisms. (Authentication of sources of data by the central infrastructure is essential to prevent unauthorised injection of log data into the system.)
- Agents can spool log data to accommodate network outages or other causes of the central infrastructure being unavailable.

---

<sup>3</sup> C. E. Shannon: *A Mathematical Theory of Communication*. Bell System Technical Journal, Vol. 27, pp379–423, 623–656, 1948.

### 4.3 Protection Guarantees

As noted in section 4.1, integrity is in general impossible to guarantee, regardless of the selected log management solution. Any such absolute guarantees should therefore be viewed with caution; vendors should instead be able to characterise the properties of their solution from an integrity perspective. Again, using Assuria Log Manager as an example:

1. Addition or deletion of whole datasets can be detected because the digitally-signed sequence numbers will be discontinuous.
2. Modification of the collected log data will invalidate the digital signature(s).
3. Authentication of Agents by [central] Collectors prevents spoofed agents or from injecting log data into the central store. (This is one reason why unauthenticated transports such as traditional syslog should be avoided if integrity is important.)

As discussed, no log management system can protect the integrity of data before it is in its possession: it can't control what events are written to (or deleted from) the original logs in the first place. However, the window of opportunity for modifying the original log can be made arbitrarily small, and modifications to audit policy and audit logs should themselves be audited, ideally elsewhere.

All cryptographic protections rely on the suitable protection of private keys and on the security of the cryptographic algorithms used. The log management solution should use standard cryptographic algorithms and key storage mechanisms. Key sizes should be sufficiently large given the expected log data retention period.

Failure of the disk[s] holding the central store can cause loss of log data. We would therefore recommend an appropriate backup policy and use of fault-tolerant storage (e.g. RAID).

Another threat to integrity (and confidentiality and availability) are standard attacks against the log management software itself, such as SQL injection attempts and exploitation of buffer overflows. A solution should include a variety of protections against these attacks, as does ALM.

## 5 Chain of Custody

Chain of custody involves identifying entities (users or systems) that have had access to or otherwise processed log data. In the case of ALM:

1. Datasets are digitally signed at all points in the collection chain.
2. Access to collected log data is recorded in ALM's own audit log.

## 6 Summary

In this white paper, we have tried to dispel some of the myths and untruths around forensic soundness of log data and also to illustrate why the forensic claims of some vendors may be disingenuous at best. We have also tried to illustrate processes and mechanisms within Assuria Log Manager that have been designed to facilitate forensic soundness and log data integrity in general.

Assuria Log Manager was specifically designed to meet the forensic log management and SIEM requirements of Government, defence and commercial organisations of any size. For more information on Assuria Log Manager or this white paper, please contact Assuria by emailing us at [info@assuria.com](mailto:info@assuria.com) or by visiting us at [www.assuria.com](http://www.assuria.com).