Safer Log Files

Rocco Gagliardi
Defense Department, scip AG
roga@scip.ch
https://www.scip.ch

Marc Ruef (Editor)
Research Department, scip AG
maru@scip.ch
https://www.scip.ch

Keywords: Cloud, Detect, Forensic, Joke, Linux, Logging, Trust, Unix

1. Preface

This paper was written in 2013 as part of a research project at scip AG, Switzerland. It was initially published online at *https://www.scip.ch/en/?labs.20130905* and is available in English and German. Providing our clients with innovative research for the information technology of the future is an essential part of our company culture.

2. Introduction

We have (securely) collected a lot of event records from sparse generators, centrally consolidated and archived in a daily rotated archive. Which measures can we take in place to protect the content of the archives?

3. Protecting Log Files

File attributes will help. BSDs have the ability to assign extra attributes to files and directories (Linux has similar features, but who uses Linux in a secure environment? Ok! take it easy, it's a joke… but still use OpenBSD, please). This is different from the standard Unix permissions scheme in that the attributes set on a file apply universally to all users of the system, and they affect file accesses at a much deeper level than file permissions or ACLs.

Use `ls -lo` to view the attributes and use `chflags` to modify them.

One useful attribute for protecting log files is *append-only*. When this attribute is set, the file cannot be deleted and writes are only allowed to append to the end of the file.

```
# chflags sappnd filename
```

But this is just the half of the game: `root` (or a pretending one) can still remove the attribute, change the log and restore the attribute without any notice (except for the last operation: restoring the attribute).

To prevent this, you'll need to disable the ability to remove the append-only attribute.

The BSDs accomplish this thing through the use of `securelevels`. The `securelevel` is a kernel variable that you can set to disallow certain functionality. Raise the `securelevel` to `1`. Once the `securelevel` has been set to a value greater than `0`, it cannot be lowered. By default, OpenBSD will raise the securelevel to `1` when in multiuser mode. In FreeBSD, the securelevel is `-1` by default.

To change this behavior, add the following line to `/etc/sysctl.conf`:

```
kern.securelevel=1
```

Be aware that adding append-only flags to your log files will most likely cause log rotation scripts to fail. However, it is a valuable enhancement of the security of your audit trail.

4. Maintain Integrity

Basically, we want to protect log files from *tampering*, *poisoning* or *forgery*. This kind of techniques can be referred to as *integrity verification*, *incontrovertible data*, *court admissible evidentiary data* or even *signed and sequenced* data.

What methods can we use?

- **WORMs**: *Line printers* and *WORM Disk* are self explanatory. In some cases still the best solution. *Netapp Snaplock* [1] is an example of enterprise solution. Sandisk WORM SD card is/was a smaller and affordable solution to store information safely; 1GB isn't really usable to store log data, but hashes; enough to verify a logfile.

- **Remote Copy**: *Duplicate* the log stream and save on *separate* log server. This stream, stored on a server with an increased security level, can be used for a reasonably trusted verification.

- **Hash Chaining**: Use *cryptographic methods* to protect the integrity of log records, once they are stored. Basic idea is to feed the hash of the message *N* in the *N+1*. What this does is ensure that an attacker can't tamper with entries in the middle of the log, without being detected. The attacker can still replace all the entries in the *chain* or modify the last message in the chain. An issue of this method is the verification effort: If you need to verify the integrity of the message *X* you must check the integrity of all preceding messages (the first checked with a secret hash). For more, read about *secure timestamping using hash chaining* [2]. A variant, to speed up verification process, consist to use *Merkle-tree* [3] (balanced binary tree).

5. Summary

Why so serious about logfile security? If we can't prove or at least reasonably trust our logs, we are blind. Keeping log secure and verifiable is one of the most complex part of the log management solution. Many solutions exist to address the problem but must be implemented very carefully. And the clouds does not promise anything good!

6. References

- *Cryptographic Support for Secure Logs on Untrusted Machines* [4]
- *Secure audit logging with tamper-resistant hardware* [5]
- *Guide to Computer Security Log Management* [6]
- *rsyslog's first signature provider: why Guardtime?* [7]
- *Forensics syslog-ng Use Case* [8]

7. External Links

[1] http://www.netapp.com/us/products/protection-software/snaplock.aspx
[2] https://secure.wikimedia.org/wikipedia/en/wiki/Linked_Timestamping
[3] http://en.wikipedia.org/wiki/Merkle_tree
[4] http://www.schneier.com/paper-secure-logs.pdf
[5] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.101.4911
[6] http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf
[7] http://blog.gerhards.net/2013/05/rsyslogs-first-signature-provider-why.html
[8] http://www.balabit.com/support/documentation/uc_forensics_syslogng_en.pdf