

Towards a Secure Logging Mechanism for Dynamic Systems*

Rafael Accorsi

Department of Telematics

Albert-Ludwigs-Universität Freiburg, Germany

accorsi@iig.uni-freiburg.de

Extended Abstract

Logging is a central service in computing systems. It collects information about the events happening in distributed devices and, thus, provides a basis for other services within the system. In highly dynamic, mixed-mode systems, logging takes a new dimension: resource-poor devices need to securely relocate log data to resource-rich collectors not necessarily residing within the same trust domain. This scenario brings several challenges to logging, in particular regarding its security requirements. In this paper, we report on ongoing work in developing a logging mechanism to securely store log data in marginally trusted remote collectors.

Characterizing Secure Log

The need for secure logging mechanisms is evident. But, irrespective of the envisaged application of log data, it must be authentic, for *ex falsum quod libet*, i.e., from a falsity everything follows. However, the increasing distribution and openness of computing systems pose challenges to logging, in particular if the device notifying the events is not the same as the service storing this data. In this context, while the device's log data is stored in partially trusted collectors, accurate security guarantees regarding the authenticity of log data should not only hold, but also be proved to the device.

We define authenticity of log-data as the conjunction of *faithfulness* and *uniqueness*: while the former denotes that the information logged is a faithful representation of the messages sent by the device, the latter requires that log data shall not allow for parallel realities. Furthermore, log services based providing these properties need to fulfill two other requirements, namely: *tamper evidence*, i.e. attempts to illicitly modify log data must be detectable to a verifier [4]; and *forward integrity*, that is, should contain sufficient information to confirm or rebuke allegations of log data modification before the moment of the compromise [1]. The approach we propose aims at fulfilling these properties.

Attacks upon log services have different facets. Given a threat model based on Dolev-Yao attacker

model [2], faithfulness and uniqueness can be attacked by replaying of log messages, modifying sent and stored log data, impersonating the device, violating the confidentiality of log messages, and damaging the availability of log service.

Describing our Approach

Our approach to securing log data in dynamic systems builds on and extends the techniques proposed in [5]. We assume a mixed-mode environment with resource-poor devices and resource-rich collectors. In particular, we assume that devices have limited storage, while collectors have unlimited storage.

Our approach consists of delegating storage of log data to partially trusted collectors which are then in charge of proving the receipt of log messages. To this end, the following main steps are used, where in this paper we focus on the last two steps:

- *Mutual authentication of device and collector.* Apart from authentication, the two services also agree on a secret number that will be used to ascertain authenticity of log messages.
- *Construction of the logfile.* The device is in charge of applying cryptographic techniques to safeguarding the integrity of its logfile. With regard to how the log data is sent to the collector, we assume that chunks of log data are sent from the device to the collector. In our approach, we employ hash-chains to ensure that log data fulfills faithfulness and uniqueness.
- *Collector's acknowledgement of receipt.* The collector computes a hash value based on the device's messages and sends it signed together with a timestamp and protocol sequence number back to the device. The device then stores this unambiguous piece of information, as it demonstrates that the collector received the chunk of log data correctly and can be held accountable for attacks upon this data.

Construction Phase

We consider log entries as depicted in Fig. 1, where

1. $A_j = Hash(A_{j-1})$ denotes the authentication key of the j th log entry. The confidentiality of

*A detailed version of this paper is electronically available at <http://www.informatik.uni-freiburg.de/~accorsi/>

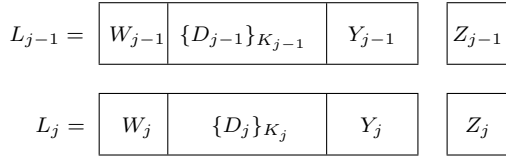


Figure 1: Adding an entry to the logfile.

this information is essential for the security of the log data, as it is used, either directly or indirectly, in every step of the protocol. Thus, we assume that the computation of the new value irretrievably overwrites the previous value.

2. $K_j = Hash(W_j, A_j)$ is the cryptographic key with which the j th log entry is encrypted. This key is based on the permission mask W_j . Thus, only permitted services may access the entry.
3. $Y_j = Hash(Y_{j-1}, \{D_j\}_{K_j}, W_j)$ is the j th value of the hash-chain. Each link of the hash-chain is based on the corresponding encrypted value of the log data. This ensures that the chain can be verified without the knowledge of the actual log entry.
4. $L_j = W_j, \{D_j\}_{K_j}, Y_j$, i.e., the log entry consists of the permission mask, the encrypted log data and the hash-chain value.
5. $Z_j = MAC_{A_j}(Hash(L_j, Z_{j-1}))$ is the authenticator of the j th log entry. Note that we compute the message authentication code for the whole entry instead of a field of it (in the case of Schneier and Kelsey, the hash-chain value Y). While this increases the computational cost involved in calculating this field, it improves the security of the acknowledgment phase.

Acknowledgement Phase

The last phase of the protocol aims to provide irrefutable evidence regarding collector’s possession of the chunk of log data sent by the device, as well as the chunk’s integrity.

To this end, the following steps are carried out:

1. by receiving the chunk starting at L_j and ending at L_k (with $j < k$), the collector computes for each entry L_i the corresponding A_i and Z_i values.
2. after $k - j$ iterations, the collector obtains the authenticator $MAC_{A_k}(Hash(L_k, Z_{k-1}))$.
3. the signed proof value is sent to the device $Sign(MAC_{A_k}(Hash(L_k, Z_{k-1})))_{K_c^{-1}}$. This message includes a timestamp and protocol step identifier. Moreover, the collector irretrievably overwrites the memory space that stores A_k .

The device then checks whether the authenticator matches with the authenticator computed during the second phase. If it does, the collector frees the storage by deleting the chunk.

Ongoing and Future Work

We have reported on work in progress towards a secure logging mechanism for dynamic systems based on UbiComp environments. This is the first step in our investigation and some issues remain to be examined. First, although our method provides new insights into the security properties and techniques for remote logging, at the moment it strongly relates to [5] and it is unclear whether simplifications are possible. A realization of the method shall shed a light on its performance and practical adequacy.

Second, we plan to develop protocols to employ the permission mask against authorized services, e.g. audit or configuration services.

Third, in our approach we cannot make strong statements about the actual behavior of the collector. To address this problem, we currently investigate the use of trusted computing. The idea is to remotely attestate collector’s platform and, thus, its behavior [3].

Finally, if misused, such a powerful log mechanism is a privacy *endangering* technology, especially in an environment where virtually *every* piece of information can be collected and processed in order to derive further characteristics of an individual. We plan to address the privacy issues hidden in our approach to logging and whether one can draw benefits from it.

References

- [1] M. Bellare and B. Yee. Forward integrity for secure audit logs. Tech. report, UCSD, Dept. of Computer Science & Engineering, 1997.
- [2] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 2(29):198–208, 1983.
- [3] A. Hohl, L. Lewis, and A. Zugenmaier. Look who’s talking – Authenticating service access points. In D. Hutter and M. Ullmann, editors, *Proc. of the Conf. on Security in Pervasive Computing*, vol. 3450 of *LNCIS*, pages 151–162, 2005.
- [4] G. Itkis. Cryptographic tamper evidence. In *Proc. of the Conf. on Computer and Communication Security*, pages 355–364, 2003.
- [5] B. Schneier and J. Kelsey. Security audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999.