Trustworthy Log Reconciliation for Distributed Virtual Organisations

Jun Ho Huh and John Lyle

Oxford University Computing Laboratory
Wolfson Building, Parks Road
Oxford, OX1 3QD
{jun.ho.huh,john.lyle}@comlab.ox.ac.uk

Abstract. Secure management of logs in an organisational grid environment is often considered a task of low priority. However, it must be rapidly upgraded when the logs have security properties in their own right. We present several use cases where log integrity and confidentiality are essential, and propose a log reconciliation architecture in which both are ensured. We use a combination of trusted computing and virtualization to enable *blind log analysis*, allowing users to see the results of legitimate queries, while still withholding access to privileged raw data.

1 Introduction

The notion of a *Virtual Organisation* (VO) runs commonly through many definitions of what constitutes a grid: "many disparate logical and physical entities that span *multiple administrative domains* are orchestrated together as a single logical entity" [15]. The rise of many types of organisational grid systems, and associated security threats, makes the provision of trustworthy audit-based monitoring services necessary; for instance, to monitor and report violation of service-level agreements [18], or to detect events of dubious user behaviour across multiple domains and take retrospective actions [17].

In reality, a lot of these audit-based controls are prone to be compromised due to the lack of verification mechanisms for checking the correctness and the integrity of logs collected from different sites; and also because some of these logs are highly sensitive, and without the necessary confidentiality guarantees, neither trusts the other to see the raw data. Many log anonymisation techniques have been proposed [9,12,19] to solve the latter issue; however, adapting such techniques and assuring that these anonymisation policies will be correctly enforced at a remote site, is a whole new security issue. The problem with existing solutions is that they provide only weak protection (or none) for such security properties upon distributed log collection and reconciliation (Section 3).

In our previous work [8] we have proposed a logging infrastructure using the driver virtualization in Xen that enables trustworthy generation and storage of the log data. In this paper, we take a step further and describe a log reconciliation method for guaranteeing their integrity and confidentiality.

L. Chen, C.J. Mitchell, and A. Martin (Eds.): Trust 2009, LNCS 5471, pp. 169–182, 2009.

The rest of the paper is organised as follows. In Section 2, we present a number of motivational examples and highlight distinct security challenges with processing distributed log data. Section 3 discusses the security gaps of existing solutions. Then, in Section 4, we present the trustworthy log reconciliation requirements which address these security gaps. Mindful of such requirements, we describe a reconciliation infrastructure for a VO in Sections 5 and 6. Finally, in Section 7 we discuss the contribution of this paper and the remaining work.

2 Motivational Examples

2.1 Healthcare Grids and Dynamic Access Control

The first example application arises in the context of a healthcare grid. In 'e-Health', many data grids are being constructed and interconnected in order to facilitate the better provision of clinical information. Each clinic (an independent legal entity) participating in the grid owns and manages physical databases which together form the virtualized clinical data and log stores. To motivate the use cases described later in this section we use an abstract view of the VO (see Figure 1): each node consists of external and internal services where the virtualization of data sources takes place; it also has its own local data and logs; a standard external service enables communication between different nodes.

Consider the following example in the context of Figure 1. A simplified healthcare grid consists of two nodes, a GP Practice (GP) and a Specialist Clinic (SC). A patient in GP is often referred to SC to see a specialist. We shall assume that a single table at each clinic (T_1, T_2) is made accessible to a researcher R, and that the National Health Index (NHI) uniquely identifies a patient across the grid to enable the linking of data. R is carrying out a study that looks at association between smoking status (T_1) and development of lung cancer (T_2) in the population of Oxfordshire.

R has originally been granted full access to both T_1 (at GP) and T_2 (at SC) to conduct this research. By joining the data across two clinics, R would have access to potential identifiable information about patients: for example, R could

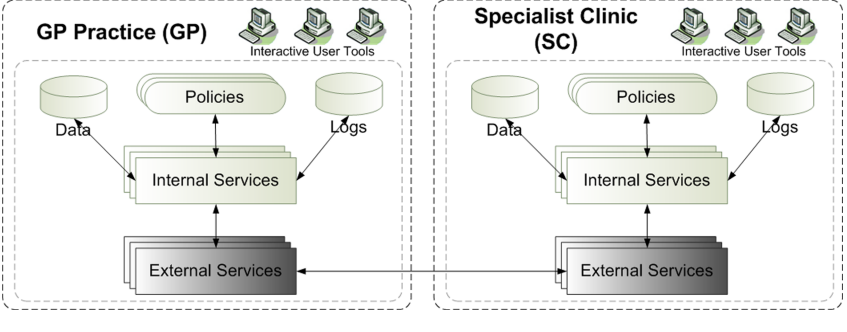

Fig. 1. Abstract View of the Virtual Organisation

find out that patient 1, born on the 20/05/88 and living in OX2 5PS who has Dr. Anderson as their GP, is a smoker and has a lung cancer.

GP Practice (GP) T_1				
NHI	DOB	GP	$Smoke$	$Risks$
1	20/05/88	Dr. Anderson	yes	overweight
2	30/07/88	Dr. Anderson	no	allergies

Specialist Clinic (SC) T_2		
NHI	$Postcode$	$LungCancer$
1	OX2 5PS	yes
2	OX2 6QA	no

In a secure VO, as soon as R finds out from querying T_2 that patient 1 has lung cancer, R's access on T_1 for patient 1 needs to be restricted to, for example, only the NHI and $Smoke$ fields. For GP to have restricted R's access rights to information pertaining to patient 1 on T_1, would have required GP to collect *data access logs* from SC to build up a picture of what R already knows, and to update its own access control policies to prevent R from collecting potential identifiable information. Although, in general, SC would never give out patients' lung cancer status in the form of audit logs to an untrusted GP.

This type of distributed audit approach has been suggested [17] to detect patterns of behaviour across multiple administrative domains by combining their audit logs. However, the problem arises from the fact that log owners do not trust other sites to see their privileged raw logs. This approach will only work if log owners can be assured of *confidentiality* during transit and reconciliation.

2.2 The Monitorability of Service-Level Agreements (SLAs)

The provision of Service-Level Agreements (SLAs) and ensuring their *monitorability* is another example use for trustworthy log reconciliation.

A SLA is a contract between customers and their service provider which specifies the levels of various attributes of a service like its availability, performance and the associated penalties in the case of violation of these agreements. Consider a case where the client receives no response for a service (for which they have entered into a SLA) within the agreed interval of time, complains to the provider that a timely response was not received and requests financial compensation. The provider argues that no service request was received, and produces a log of requests in their defense. There is no way for the client to find out the truth: the provider could have delivered tampered evidence regarding this event. The problem with this type of SLA is that it is defined in terms of events that the client cannot directly monitor, and they must take the word of the provider with respect to the service availability.

Skene et al [18] suggest a way of achieving the monitorability with trusted computing. This involves generating trustworthy logs, ensuring that unmodified

logs have been reported by both parties and that these logs have been used for monitoring SLAs. For instance, if the client is able to verify with remote attestation that trustworthy logging and reporting services operate at a remote site, then the client may place conditions on any event of their interest and construct more useful SLAs. This approach needs to guarantee the *integrity* of all service request/response logs to an evidential standard (i.e. to a standard acceptable for judicial usages) upon distributed reconciliation and analysis. A monitoring service would then be able to generate a reliable SLA report for the client to make claims.

Logs often contain sufficient information to be used as evidence in a variety of context. However, the inability of a site to verify the integrity of logs collected from other sites and the lack of guarantees that their own logs are being used unmodified at remote sites, make it extremely challenging for one to adapt the usual audit-based monitoring method to the VO.

3 Relevant Work and a Gap Analysis

Having identified the security challenges of imposing audit-based controls, we are now in a position to present a gap analysis on existing solutions.

DiLoS (Distributed General Logging Architecture for Grid Environments) [5] provides general logging facilities in service oriented grid environments to enable tracking of the whole system. One of its application models is to facilitate accounting for resource-providing services: to measure and annotate who has used which services, and to bill usage prices. In this accounting domain, however, DiLoS does not consider the log integrity issues and the possible threats that have been covered in Section 2.2. Without security mechanisms to protect log integrity, their architecture cannot be relied upon to perform calculating and billing functions.

Piro et al [14] have developed a more secure and reliable data grid accounting system based on metering resource usage. All communications are encrypted [13]; but a privileged user may still configure the Home Location Register (HLR), a component that collects remote usage records for accounting, to disclose sensitive usage records. A rogue resource owner may modify the Computing Element (CE), which measures the exact resource usage, in order to fabricate records and prices for profit.

The NetLogger Toolkit [20] provides client application libraries (C, Java, Python APIs) that enable one to generate log messages in a common format. It also includes monitoring tools for log collection and analysis at a central point. Again, the log integrity and confidentiality threats discussed in the previous section undermine their approach: access requests are processed without any authorisation policy enforcement, and the logs are transferred across the network in an unencrypted and unsigned format. No attempt is made to safeguard the logs while they are being collected and processed at the reconciliation point.

Similar security problems undermine other existing grid monitoring tools such as APEL (Accounting Processor for Event Logs) [3], which builds accounting

records from system and gatekeeper logs generated by a site; and GMS (Grid Monitoring System) [11], a system that captures and stores job information in a relational database, and supports resource usage monitoring and visualising.

4 Trustworthy Log Reconciliation Requirements

To fill the security gaps identified above, we provide a high-level overview of the key requirements with respect to our motivational examples.

Log Migration Service. Due to the number of potential security vulnerabilities, complex grid middleware services can not be relied upon to perform trusted operations [4]. Instead the security controls required for safe log data transfer need to operate within a more secure migration service. This implies data flow encryption and signing requirements upon log access and transfer requests. These are integral in preventing intruders from sniffing the logs processed through insecure grid middleware, and from launching man-in-the-middle type of attacks. It is also possible for a log owner to deliver fabricated logs, as the service provider might do in the SLA example. To provide a safeguard against such a threat, the migration service needs to access the signed logs (and the original logs) directly from the protected log storage. This would give sufficient information for an end user to verify the log integrity.

Log Reconciliation Service. Our examples have a common set of requirements for a trustworthy reconciliation service. They require each site to negotiate with others and grant permissions to view their logs. These sites (before granting permissions) need to be assured that their logs will not be compromised and will be used without modification. The integrity and the confidentiality of the collected logs as well as the processed results (e.g. summaries on SLA violation) need to be protected to prevent a malicious user from modifying or stealing them.

To make it harder for insiders to gain unauthorised access and modify the reconciled logs, this service needs to run in a strongly isolated compartment with robust memory protection. It should also be a small and simple code to minimise the number of security holes that might be exploited.

Blind Analysis of the Logs. Returning to our healthcare example, imagine that SC has agreed to share their logs with GP for dynamic access control. But at the same time they are unwilling to let the system administrator at GP see the actual contents of the logs; or only let part of the data be seen as a summary information. For example, "R's access rights on T_1 for a patient with NHI 1, aged 20 and living in OX2 area, have been restricted to NHI and $Smoke$ fields.". Such anonymisation of end results ensures that the administrator cannot find out about a patient's lung cancer status, and yet, still know exactly how the access control policy has been changed for R.

Log owners need to be assured that any sensitive information contained in their logs will only be revealed to an extent that has been agreed and stated in

anonymisation policies: this requires a mechanism, possibly within the reconciliation service, to carry out a blind analysis of the collected logs so that a user only sees the running application and the end results, which are just sufficient for them to carry out post log analysis or to know about the important system updates.

5 Trustworthy Logging Architecture

In our previous work [8], we have developed a logging architecture based on Virtual Machine (VM) isolation and remote attestation (see Figure 2). Upon installation of this architecture, each VO participant will be capable of generating and storing log data, and proving to other sites that these logs are trustworthy.

The Trusted Computing Group (TCG) [1] has developed a series of technologies based around a Trusted Platform Module (TPM) which helps to provide two novel capabilities [7]: a cryptographically strong identity and reporting mechanism for the platform, and a means to *measure* reliably a hash of the software loaded and run on the platform (from the BIOS upwards); such measurements are stored and retrieved from Platform Configuration Registers (PCRs) in the TPM. These provide the means to *seal* data so that it is available only to a particular platform state, and to undertake *remote attestation*: proving to a third party that a remote device is in a particular software state. TPM-generated Attestation Identity Keys (AIKs) are used to sign PCR values and to prevent tracking of platforms. These trusted computing capabilities can be used in a virtualized environment where a physical host is segmented into strongly isolated compartments to make attestation feasible (with robust memory protection), and to limit the impact of any vulnerability in attested code. Our architecture uses the Xen Virtual Machine Monitor (VMM) [2] to achieve this isolation: a thin layer of software operating on top of the hardware to enable VM abstraction and control the way a VM accesses the hardware and peripherals.

All log security functions are enforced by the *log security manager* VM, a small amount of code running inside *back-end driver* VMs and the *log analysis manager* VM; each of which has been designed to perform a small number of simple operations so that it can be compartmented with a high degree of assurance.

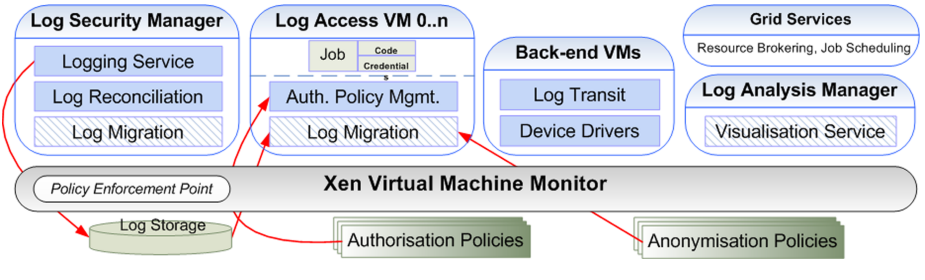

Fig. 2. Abstract View of Trusted Logging Services

Attestation of these compartments, the Policy Enforcement Point (PEP) and the VMM is sufficient for one to establish trust with a VO platform, and to be assured that its log security functions have not been subverted; this is our Trusted Computing Base (TCB).

A small number of trusted back-end driver VMs are responsible for generating all logging requests upon use of device drivers. All other VMs must communicate with one of these driver VMs to access the physical hardware. Inside these VMs, the *log transit* component collects important I/O details and submits requests to the log security manager. Applications and middleware services running in other compartments are no longer relied upon to generate trustworthy log data.

The log security manager performs a range of security functions through the following services:

- The *logging service* ensures that no adversaries can access or modify the log data dispatched from log transits. It filters out untrustworthy logging requests and verifies their integrity before storing them.
- The *reconciliation service* facilitates trustworthy reconciliation and transformation of the collected logs. It enables blind analysis of the logs by enforcing anonymisation policies.
- The *migration service* is an external service which facilitates secure communication between VMs in one or more sites by enforcing security controls required for safe log transfer.

End-user applications only have access to the externally facing *visualisation service* running inside the log analysis manager, which provides the minimal interface necessary for user applications to interactively analyse the processed log information. A compartment manager within the PEP executes a job in a per-user log access VM configured with trustworthy services. The grid services compartment isolates the middleware stack and is untrusted; it performs resource brokering and job scheduling.

6 Trustworthy Log Reconciliation

Based on the work in previous section and the requirements analysed in Section 4, we present a trustworthy reconciliation infrastructure.

6.1 The Configuration Resolver

We expand our abstract view of the VO to include a Configuration Resolver (CR) that manages metrics about the available sites in the VO and their current software configurations. To become part of the VO a site needs to first register itself with the CR by submitting the PCR representations of its TCB and log access VM image files, and a credential containing its public key for which the private-half has been sealed to both PCR values. The CR then creates a Configuration Token (CT) from this information:

$$CT = (PCR_{AIKS(N)}(TCB), PCR_{AIKS(N)}(LA), cred_{AIKS(N)}(P_K))$$

A trustworthy $PCR(TCB)$ value proves that secure logging VMs have been responsible for generating and protecting the log data; this allows a participant to have high confidence in the correctness of the logs stored in node N. Furthermore, a trustworthy $PCR(LA)$ value guarantees the security configurations of a log access VM. A value of $PCR(LA)$ is stored in a *resettable* PCR 23 because these VM image files will be remeasured and verified by the PEP at run-time before being launched.

The CR acts as a token repository in our system and offers no other complex functionality. The burden of verifying tokens is left to the participant. This is attractive from a security perspective, as the CR can remain an untrusted component. The worst that a malicious CR can do is affect the availability of the infrastructure. However, the simple CR does increase the management overhead on each node. They will all need the ability to check tokens. This involves maintaining a list of trustworthy software (a white-list), and keeping a revocation list of compromised TPMs and platforms. The security of our system depends on the proper management of this information. We suggest that a suitable compromise might be to devolve some of this functionality to local proxy-CRs, which would perform the token filtering for one specific administrative domain. This keeps control local to one site, but would decrease the effort at each individual node.

To conform to existing standards, we imagine that CR would be implemented as a WS-ServiceGroup [10]. Each node would then be a member of this CR's group, and have a ServiceEntry in its list. The membership constraints would be simple, requiring only a valid token and identity. We assume that there is a public key infrastructure available to verify their identity. As a result, the levels of indirection introduced by the TCG to prevent any loss of anonymity are unnecessary. We would suggest that the Privacy CA is not a key component of the system, and a publically-available one could be used. AIKs can be created as soon as the platform is first installed, and should very rarely need updating.

6.2 Trustworthy Log Reconciliation Infrastructure

With the resolver in place, security procedures of the reconciliation infrastructure have been carefully designed. Our healthcare example in Section 2.1 has been revisited to explain these procedures.

Creation and Distribution of a Log Access Grid Job. All end user interactions with a clinic node are made via the visualisation service. It provides the minimal interface (APIs) necessary for development of grid-enabled applications. An analysis tool should be designed to allow a user to select acceptable host configurations and user credentials, and enter the log access code/query (**1**, numbers refer to Figure 3).

A system administrator at GP, using one of these tools, requests for dynamic updates on the local access control policies. The visualisation service requests for the list of available configuration tokens (CTs) (**2**, **3**, Figure 3); the list is

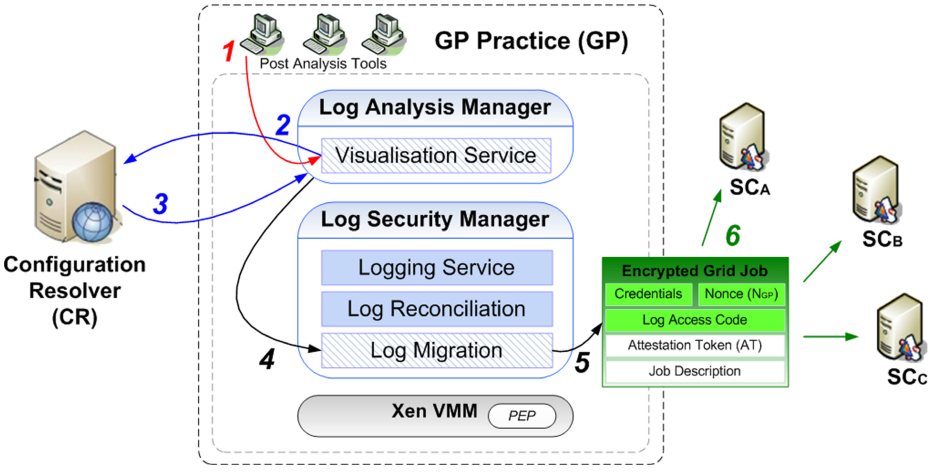

Fig. 3. Creation and Distribution of a Log Access Grid Job

forwarded to the log migration service running inside the log security manager (**4**, Figure 3). The migration service makes a list of acceptable hosts by comparing each CT against a user-specified white-list. It then creates a set of grid jobs, each of which contains the administrator's credential, log access code, job description, a nonce (N_{GP}) and an Attestation Token (AT) that can be used by any SCs to verify the security state of the system running at GP (**5**, Figure 3); AT consists of the following information:

$$AT = (PCR_{AIKS(GP)}(TCB_{GP}), cred_{AIKS(GP)}(P_K))$$

$cred_{AIK(GP)}(P_K)$ is GP's P_K credential which identifies the corresponding S_K as being sealed to $PCR(TCB_{GP})$.

For each job, the credential, the code and N_{GP} are encrypted with a $P_K(SC)$ obtained from a CT to prevent an adversary from modifying the code and to ensure that the credential is only revealed to a trustworthy SC. The use of the nonce, N_{GP}, is explained further on in this section. After encryption, these jobs are sent across the network via an untrusted grid middleware compartment which can only read the job description to identify the target SC; jobs are submitted to the PEPs of their target nodes that handle job submission (**6**, Figure 3).

Operations of a Trusted Log Access VM. In Figure 4 we take a closer look at how a job gets processed at one of the target nodes, SC_A. Any security processing required before becoming ready to be deployed in a per-user log access VM is done through the PEP: it compares $PCR_{AIKS(GP)}$ (from AT) with its set of known-good values stated in a policy to verify that the job has been created and dispatched from a correctly configured log security manager; this is how the job is authenticated at SC_A (**1**, Figure 4). Upon successful attestation, the PEP first measures the local copy of log access VM image (and a configuration

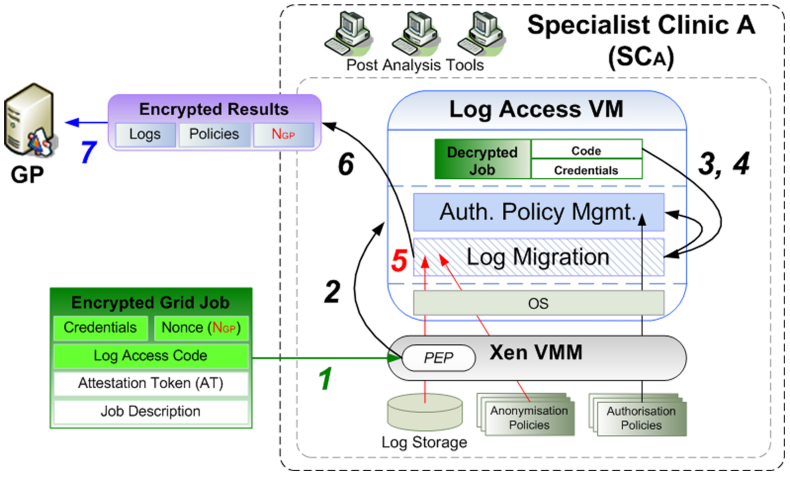

Fig. 4. Submission of a Grid Job, Creation and Operations of a Log Access VM

file), and resets PCR 23 with the new measurement, $PCR(LA_A)$; this image consists of the guest OS and the trusted middleware stack (authorisation policy management and migration services) which provides a common interface for a job to access the logs. The PEP then attempts to unseal the decryption key, $S_K(SC_A)$ (bound to $PCR(TCB_A)$ and $PCR(LA_A)$), in order to decrypt the job. Note that $S_K(SC_A)$ will only be available if SC_A is still running with trustworthy configurations *and* the VM image files have not been subverted. This is intended to guarantee that only a trusted VM has access to the decrypted credential, code and N_{GP}.

If these security checks pass, the compartment manager launches a trusted VM from the verified VM image files, and deploys the decrypted job on top of the middleware stack (**2**, Figure 4). The migration service first requests the policy management service to decide whether the administrator is authorised to view the requested logs (**3**, **4**, Figure 4). If the conditions are satisfied, the code gets executed. A log anonymisation policy ($Pols$) specified by the log owner, which states what part of the requested log data should be available to the administrator at GP, is also selected (**5**, Figure 4): in this scenario $Pols$ would restrict disclosure of $LungCancer$ status (see T_2). Existing log anonymisation techniques such as FLAIM [19] can be used in specifying these policies, in order to sanitise the sensitive data while pertaining sufficient information for analysis.

The migration service then generates a secure message containing these results (**6**, Figure 4):

$$R = \{Logs, Pols, N_{GP}\}_{P_K(GP)}$$

GP's nonce, N_{GP}, is sufficient to verify that this message has been generated from a trusted VM and unmodified code has been executed. The entire message is encrypted with $P_K(GP)$ so that it can only be decrypted if the system

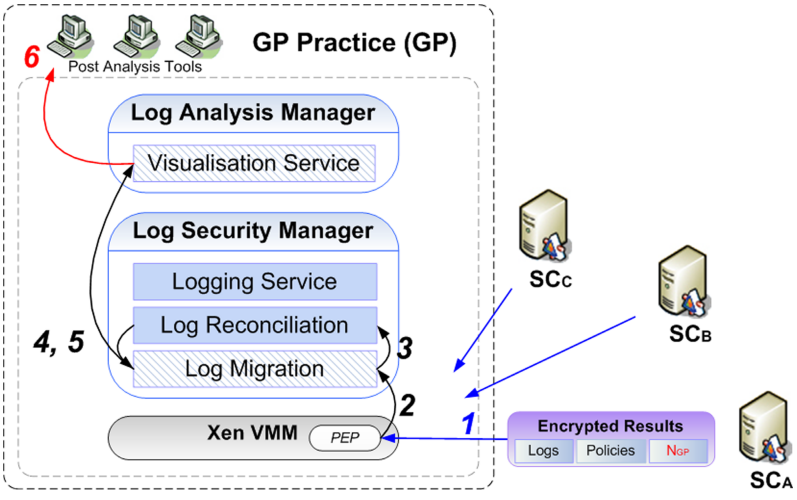

Fig. 5. Reconciliation of Collected Logs

at GP is still configured to match $PCR(TCB_{GP})$ (from AT); a compromised system will not be able to decrypt this message. An attacker will not be able to tamper with it since the private-half, $S_K(GP)$, is strongly protected inside the TPM.

Reconciliation of Collected Logs. This message arrives at the PEP of GP's system where it is decrypted using $S_K(GP)$ (**1**, Figure 5). The decrypted message is then forwarded to the migration service which compares the returned N_{GP} with the original nonce (**2**, Figure 5). A matching value verifies the correctness and the integrity of the collected *Logs*.

The internal reconciliation service reconciles the logs collected from SC_A, SC_B and SC_C and updates the access control policies according to what users have previously seen from these three specialist clinics (**3**, Figure 5). During this process *Pols* are enforced to fully anonymise the log data. Attestation of GP's log security manager (**1**, back in Figure 4) is sufficient to establish that these anonymisation policies will be imposed correctly during reconciliation. VM isolation and its robust memory protection prevent an attacker from accessing the memory space of the log security manager to steal the raw data.

A summary of the policy updates is then generated using the anonymised data and forwarded to the original requestor, the visualisation service (**4**, **5**, Figure 5). The administrator only sees this summary information on how the policies for their patient data have been updated for different users, and performs blind log analysis (**6**, Figure 5). VM Policy Attestation [6] may be used on the log analysis VM to verify that it does not permit the summary to be exported to an unauthorised device.

Table 1. Trustworthy Log Reconciliation Features

Security Goals	Trustworthy Log Reconciliation Features
Logs need to be protected from the grid middleware services	Isolation of untrusted grid middleware services; the log migration service encrypts logs using a log owner's public key for which the private-half is strongly protected inside log owner's TPM.
A log requester needs to be able to verify the integrity of the collected logs	Trustworthy log-generating sites are selected from configuration token verification; only a trusted log access VM is able to decrypt the grid job and return the logs from a remote site.
A log owner needs to be assured that their logs will be safeguarded from compromise and used unmodified at remote sites	Attestation token (part of the grid job) is used to verify the trustworthiness of a log requester's platform and its reconciliation services; the logs are encrypted using requester's public key for which the private-half is sealed to a trustworthy configuration.
Blind log analysis	Log anonymisation policies are enforced by the reconciliation service and the raw data never leaves the log security manager; an end user only sees the fully anonymised data.

6.3 Observations

Configuration Token Verification. The trustworthiness of our architecture is dependent on the ability for each participant to make the right decision about the security provided by software at other nodes. The identity of this software is reported in the PCR values contained in the CTs. We imagine that these values will then be compared to a white-list of acceptable software. However, this assumes prior knowledge of all trusted node configurations, which may not be the case if the VO is particularly large. Such a scalability issue is magnified when considering settings files, many of which will have the same semantic meaning but different measured values. It is difficult to assess how big a problem this is, but future work may look at using Property-Based Attestation [16] as a potential solution.

Node Upgrades. The most significant overhead of our system is the cost of upgrading existing nodes to support the new infrastructure. This involves installing the Xen VMM and various logging VMs. While this is a large change, the advantage of our architecture is that legacy operating systems and middleware can still be used in their own VMs. The overall administration task is therefore not so large. Furthermore, virtualization is increasing in popularity, and it seems likely that the scalability and management advantages will persuade VO participants into upgrading to a suitable system anyway.

7 Conclusions and Future Work

In this paper, we have described a trustworthy log reconciliation infrastructure to facilitate audit-based monitoring in distributed virtual organisations with strong

guarantees of the log integrity and confidentiality. Table 1 summarises how our infrastructure satisfies the security requirements analysed in Section 4.

Prototype implementations of some of these features will be constructed and their inherent security and practicality will be carefully evaluated.

We intend to extend and generalise this work into a Digital Rights Management (DRM) framework in the future. Our reconciliation and migration VMs, as the root of trust, will enforce DRM policies to protected data and ensure that they are safeguarded wherever they move in a virtual organisation.

Acknowledgements

The work described is supported by a studentship from QinetiQ. Andrew Martin reviewed an early draft of this paper and provided constructive comments and suggestions. David Power and Peter Lee provided help with the healthcare grid example.

The authors would also like to thank the anonymous reviewers for their careful attention and insightful comments.

References

1. Trusted computing group backgrounder (October 2006), `https://www.trustedcomputinggroup.org/about/`
2. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T.: Xen and the art of virtualization. Technical report, University of Cambridge, Computer Laboratory (2003)
3. Byrom, R., Cordenonsi, R., Cornwall, L., Craig, M., Djaoui, A., Duncan, A., Fisher, S.: Apel: An implementation of grid accounting using r-gma. Technical report, CCLRC - Rutherford Appleton Laboratory, Queen Mary - University of London (2005)
4. Cooper, A., Martin, A.: Trusted delegation for grid computing. In: The Second Workshop on Advances in Trusted Computing (2006)
5. de Alfonso, C., Caballer, M., Carrión, J.V., Hernández, V.: Distributed general logging architecture for grid environments. In: Daydé, M., Palma, J.M.L.M., Coutinho, Á.L.G.A., Pacitti, E., Lopes, J.C. (eds.) VECPAR 2006. LNCS, vol. 4395, pp. 589–600. Springer, Heidelberg (2007)
6. England, P.: Practical techniques for operating system attestation. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 1–13. Springer, Heidelberg (2008)
7. Grawrock, D.: The Intel Safer Computing Initiative, pp. 3–31. Intel Press (2006)
8. Huh, J.H., Martin, A.: Trusted logging for grid computing. In: 3rd Asia-Pacific Trusted Infrastructure Technologies Conference, China (2008)
9. Lincoln, P., Porras, P., Shmatikov, V.: Privacy-preserving sharing and correction of security alerts. In: 13th conference on USENIX Security Symposium, p. 17 (2004)
10. Maguire, T., Snelling, D.: Web services service group 1.2 (ws-servicegroup). Technical report, OASIS Open (June 2004)

11. Ng, H.-K., Ho, Q.-T., Lee, B.-S., Lim, D., Ong, Y.-S., Cai, W.: Nanyang campus inter-organization grid monitoring system. Technical report, Grid Operation and Training Center, School of Computer Engineering - Nanyang Technological University (2005)
12. Pang, R.: A high-level programming environment for packet trace anonymization and transformation. In: ACM SIGCOMM Conference, Germany (2003)
13. Piro, R.M.: Datagrid accounting system - basic concepts and current status. Workshop on e-Infrastructures (May 2005)
14. Piro, R.M., Guarise, A., Werbrouck, A.: An economy-based accounting infrastructure for the datagrid. In: Fourth International Workshop on Grid Computing (2003)
15. Power, D.J., Politou, E.A., Slaymaker, M.A., Simpson, A.C.: Towards secure grid-enabled healthcare. Software Practice And Experience (2002)
16. Sadeghi, A.-R., Stüble, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. In: NSPW 2004: Proceedings of the 2004 workshop on New security paradigms. ACM Press, New York (2004)
17. Simpson, A., Power, D., Slaymaker, M.: On tracker attacks in health grids. In: 2006 ACM Symposium on Applied Computing, pp. 209–216 (2006)
18. Skene, J., Skene, A., Crampton, J., Emmerich, W.: The monitorability of service-level agreements for application-service provision. In: 6th International Workshop on Software and Performance, pp. 3–14 (2007)
19. Slagell, A., Lakkaraju, K., Luo, K.: Flaim: A multi-level anonymization framework for computer and network logs. In: 20th Large Installation System Administration Conference (2006)
20. Tierney, B., Gunter, D.: Netlogger: A toolkit for distributed system performance tuning and debugging. Technical report, Lawrence Berkeley National Laboratory (December 2002)