Accountable Clouds

Ashish Gehani, Gabriela F. Ciocarlie, Natarajan Shankar
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025, USA
{ashish.gehani, gabriela.ciocarlie, natarajan.shankar}@sri.com

***Abstract*—An increasing number of organizations are migrating their critical information technology services, from healthcare to business intelligence, into public cloud computing environments. However, even if cloud technologies are continuously evolving, they still have not reached a maturity level that allows them to provide users with high assurance about the security of their data beyond existent service level agreements (SLAs).**

To address this limitation, we propose a suite of mechanisms that enhances cloud computing technologies with more assurance capabilities. Assurance becomes a measurable property, quantified by the volume of evidence to audit and retain in a privacy-preserving and nonrepudiable fashion. By proactively collecting potential forensic evidence, the cloud becomes more *accountable*, while providing its regular services. In the case of a security breach, the cloud provides the appropriate reactive security framework for validating or repudiating claims. Moreover, different levels of assurance relate to different levels of storage and privacy protection requested by users, leading to an assurance-based price model for cloud services.

I. Introduction

Digital forensics is playing an increasing important role in investigating criminal activity. According to the Federal Bureau of Investigations (FBI) [6], in 2011, 4,263 TB of data was processed for digital forensic in 7,629 criminal examinations as opposed to 439 TB and 880 criminal cases in 2003. With the increased migration of critical information technology services into the cloud, digital evidence is poised to become instrumental in investigating criminal activity committed in cloud environments as well. The reasons are manifold – financially lucrative targets are now hosted on clouds, systems are so complex that vulnerabilities are regularly discovered, and digital identities are becoming widespread, making audit trails effective. Verifying security compliance using audit trails is an effective complement to access control systems. The approach supports greater flexibility since security policies can be introduced retrospectively, modified dynamically, and reasonable violations can be accommodated. It is scalable since the auditing granularity can be dynamically tuned.

When a loss is reported, the offense must be characterized. Forensic analysis in the physical world relies on the trail of environmental changes made by a crime's perpetrator. This is known as *Locard's exchange principle* [14], after the French pioneer of forensic science who articulated it in the early 20th century. In a digital world, a criminal could conceivably erase all traces of activity. It is thus incumbent on cloud computing services to provide assurance based on trustworthy audit trail with sufficient detail and reliability to allow forensic conclusions to be drawn. However, the indiscriminate addition of auditing to a runtime environment introduces performance penalties for executing applications, requires large amounts of storage, and may compromise the privacy of individuals.

Currently, cloud computing technologies offer little in terms of sound digital forensic support, and their service level agreements (SLAs) include no clauses with procedures to follow in case of forensic investigations [21]. There are numerous scenarios where this lack of appropriate forensic support as a reactive security measure could play a significant role; we review some of the most relevant ones below.

Healthcare Systems

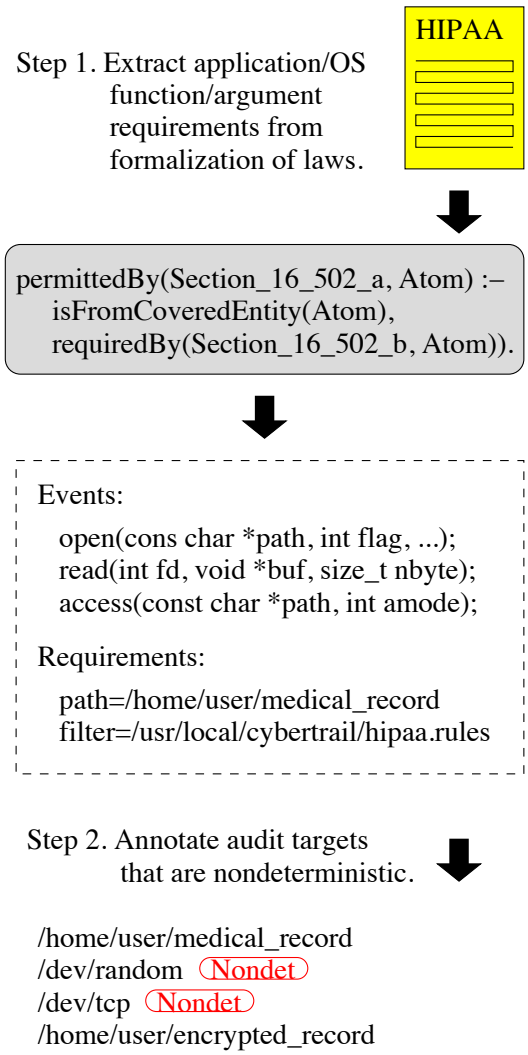
Many hospitals now outsource their information technology infrastructure to services that are deployed on clouds. A patient wishes to demonstrate that a hospital was negligent in following the proper procedures, while the hospital wishes to demonstrate that there was no lapse on its part. The hospital therefore collects nonrepudiable evidence for the procedures that it has followed. The patient has a right to examine the evidence that is relevant to his or her treatment. The evidence can be analyzed with respect to the processes that the hospital is required to follow. These processes also pertain to the sharing of medical information as regulated by the Health Insurance Portability and Accountability Act (HIPAA) [12].

Electronic Mail

Email often contains evidence similar to that of physical correspondence, but without the same degree of reliability. To support digital forensics, an electronic mail server must require email to be authenticated by the sender and registered by the receiver. The evidence of the sending and receipt of the electronic mail must be recorded by mail servers residing on clouds. The sequence of mail exchanges can then be used as evidence to establish certain claims. The electronic mail record can also be combined with other forms of evidence to corroborate a breach of contract.

Business Workflow

Many transactions involve work that is sequenced through multiple agents. Each agent can pick up the transaction only when some preconditions have been met. Since these actions are distributed over a network, it is not possible to monitor an entire transaction. Parts of the transaction might also be outsourced across multiple business domains. However, by recording the state of the transaction using cloud services at key points in the workflow, it will be possible to reconstruct the



ordering of events to reactively ensure that the business rules have been followed. Evidence-based workflow has applications beyond law enforcement in areas such as software certification and clinical trials.

Online Services

A purchase of services or goods from a service in the cloud can involve a complex sequence of exchanges. For example, an auction involves collecting bids and counterbids to determine a winner. The auditing infrastructure can ensure that the bidding process and winner selection was fair by automatically generating evidence that each participant can use to verify that the process was free of collusion after the auction ends. In particular, it will allow the seller and the bidders to check that previously observable claims are consistent with the end result.

Insider Threats

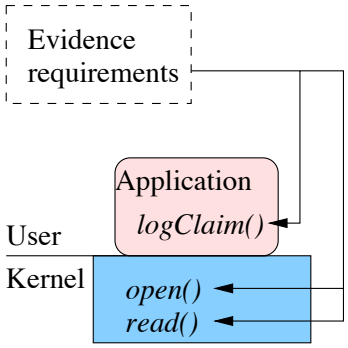
A security violation occurs when a computing agent obtains unauthorized accesses to a protected resource. By maintaining a record of the accesses along with the permissions and authorizations, a cloud service can detect security violations. This is particularly useful where there are obvious legal and employment consequences to such unauthorized access. While such breaches can also be flagged by an online intrusion detection system, an audit-based approach is open to new forms and sources of evidence and can more easily admit exceptions and policy changes, including those that have retrospective effect.

II. Cloud Forensics

Given the challenges raised by the previous scenarios, we propose a principled approach for collecting forensic evidence in the cloud environment. The approach includes different steps that lead to more accountable cloud environments, while allowing for assurance-based price models for cloud services.

Formal Representation

The first step consists of taking extant laws and extracting from them an abstract description of what elements can serve as digital evidence. For example, a substantial part of HIPAA and portions of other laws have been converted into a formal representation by members of the computer science department and law school at Stanford [3]. Such representations can serve as the basis for defining a set of operating-system- and application-level evidence requirements.



Target Mapping

The next step involves translating the evidence requirements into concrete elements of the operating-system and target applications' programming interfaces. This will inform the cloud computing runtime regarding which specific parameters and functions need to be audited. In particular, any elements that would introduce nondeterminism into the execution of a target application, such as reads from a network socket, `/dev/random`, or an asynchronous signal, need to be audited. Coupled with the target applications' binaries and linked libraries, and a record of their initial execution environments, this will suffice for validating an audit record, as demonstrated in the virtual-machine context [5]. The use of a continuous snapshotting filesystem, such as NILFS [16], will ensure the subsequent availability of data after files are modified or deleted.

Dynamic Auditing

A third step consists of mechanisms to dynamically activate auditing functionality to collect digital evidence when required. Extant systems rely on interfaces that either introduce too much overhead or do not have visibility at the abstraction levels of interest. The use of a trusted kernel can significantly reduce the overhead imposed. For example, the Linux kernel *markers* insert `noops` that do not adversely affect runtime behavior at locations where auditing functionality is inactive by default and can safely be added subsequently. When a point of interest is determined, the kernel image can be dynamically modified in memory, replacing the specific `noops` corresponding to a marker with a branch to the requisite auditing code.

Privacy-Aware Evidence Management

The fourth step aims to protect the privacy of users, starting from the point at which audit records are first generated. The approach relies on being able to separate digital evidence that has been deterministically generated from that which is not.

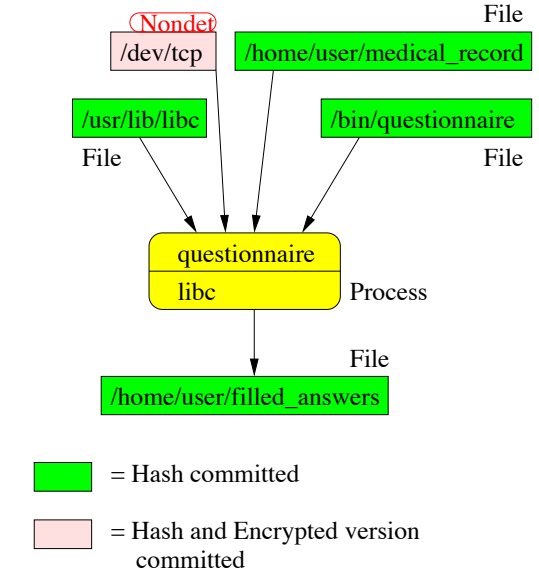
For example, the data read from a file is considered to be deterministic if it can be read again from a retained copy (such as one available from a snapshotting filesystem), whereas data read from a network connection is considered nondeterministic. The vast majority of evidence is deterministic and can be omitted from the evidence record if its sources (which include the program that created it, the runtime environment in which the program started, and all nondeterministic inputs) are retained. The remaining records are a small enough set that they can be encrypted using *attribute-based encryption* [17] such that only parties with the right set of credentials can access them. Committing signed hashes of an applications' binary and linked libraries, input files, and runtime environment will make it possible to detect if a cloud user claims to have used a different application or inputs from the ones she actually utilized.

Distributed Trails

The next critical step ensures that the audit trail is nonrepudiable. Past approaches [9] have focused on applications that can be modeled as deterministic finite state automata. Nodes that serve as witnesses are assumed to have copies of these automata to check that the evidence emitted from a remote node is consistent. However, handling arbitrary programs and maintaining user privacy precludes sharing the automata descriptions.

To create a distributed audit trail, a system service can run on all participating cloud nodes. When a cloud node boots up, it inserts itself into an overlay that performs a rendezvous using an extant structured peer-to-peer infrastructure, such as OpenDHT [19]. Since the privacy-protected evidentiary record developed in the previous step is relatively small, redundant copies of it will be maintained as a *distributed journal* in the overlay. The functioning of the latter construct will be similar to that of the log used in journaling filesystems [11] with optimization for intra-node locality.

The system service maintains a *witness store* at each cloud node that contains fragments of other nodes' journals. When nondeterministic elements of digital evidence are needed during a forensic analysis, they are retrieved from corresponding distributed journals rather than the originating nodes. In this manner, the audit trail can be repudiated only if a substantial fraction of the cloud infrastructure has been compromised. The system has the desirable property that its trustworthiness increases with scale, similarly to Tor, the Onion Router [4]. If the overlay has a sufficient number of nodes outside the cloud infrastructure, the scheme will yield *accountable cloud*



computing. The external nodes will record sufficiently many nondeterministic fragments of a computation to allow claims to be validated or repudiated.

Actuarial Forensics

Finally, rather than attempting to provide an *a priori* level of auditing, cloud providers allow customers to select a level of service. This assumes a price model based on the quantity of proactively collected forensic evidence and privacy protection, similar to insurance price models. If an application is compromised, the chance of being able to reconstruct the intruder's actions increases as more detailed evidence is available. The finer granularity of evidence could take the form of higher sampling rate in the case of statistical-based anomaly detection (which requires high fidelity data), or more classes of events and details in the case of specification-based anomaly detection. Both cases impose an overhead in storage and processing resources that the cloud service has to support.

III. Guarantees

The above steps towards an accountable cloud will offer the following security guarantees:

- **Evidence completeness**: The formal representation and target mapping steps will ensure that all the relevant evidence (according to the extant laws) is collected.
- **User privacy-preservation**: The privacy-aware evidence management step will ensure that the users' privacy is protected, by applying attribute-based encryption to the audited data generated by users' activity.
- **Audit-trail nonrepudiation**: The distributed trails steps will ensure that that the audit trails are nonrepudiable by storing them in distributed journals. The guarantee increases with scale.

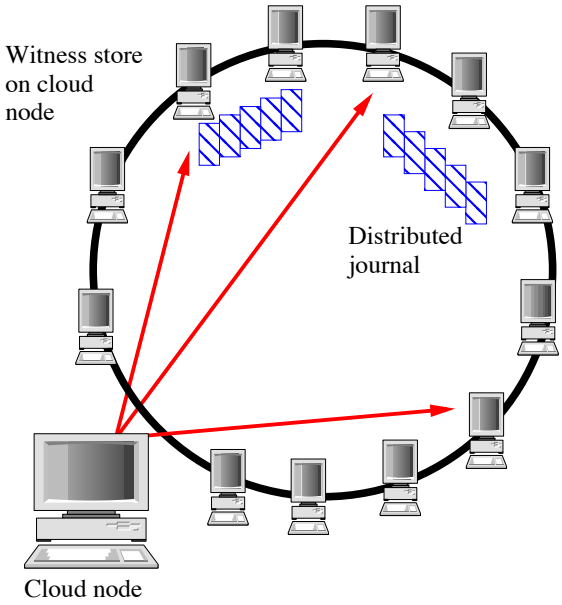
IV. Related Work

The need for accountable cloud has been identified also by Haeberlen, who emphasized the benefits for both the customers and the cloud providers [10]. His work was "intended as a call for action", which we followed, by providing more steps towards accountable cloud.

Our approach calls for proactive collection of forensic evidence which is then examined to identify criminal activity. Possible approaches for forensic analysis include online intrusion detection using a rule-based expert system to examine audit data to identify possible unauthorized access in a system. The rules are based on the typical patterns and profiles of attacker behavior. Auditing has been used to monitor threats against Air Force computers by Anderson as far back as 1980 [1]. Most host-based intrusion detection systems use logs [2]. A number of research efforts have focused on log management. Schneier analyzed the problem of maintaining the integrity of log files on a host [22]. Silbert considered the problem of secure auditing in a distributed system [23]. Flack focused on reconstructing formal semantics from logs [7]. Roger implemented an online algorithm that checks the model defined by declarative constraints stated in a temporal logic against log records [20]. Marty proposes the use of use-case oriented logging for forensic analysis of cloud applications [15]. The main concern we focus on is whether sufficient auditing can be performed within a budget to be able to prove in a court of law that the evidence supports a specific claim that a violation did or did not occur.

In the case that the cloud service provides an installed operating system, the auditing can occur in the trusted code present in the virtual machine. On Unix-based systems, the

Step 5. Nondeterministic evidence is redundantly logged in a distributed journal stored in an overlay of participating witness stores.



ptrace interface was designed for debugging and therefore allows very fine-grained intervention. However, the fact that it requires two context switches (to and from the monitoring process) to handle each system call means that it slows down the monitored process by an order of magnitude [13] for applications that make heavy use of resources (such as the filesystem or network) that are mediated through kernel interfaces. Another strategy is to use the `/proc` filesystem to gather evidence, as done by sandboxing schemes [8], or a kernel module that intercedes at the system call interface directly, such as the auditing library *libaudit* and its associated tools that are included in prominent Linux distributions. These approaches have substantially improved performance though they are unable to audit application-level function calls. The SystemTap project [18] allows a wider range of instrumentation, with visibility into the kernel itself, and allows specifications to be written at an abstract level that is then automatically compiled into C, which in turn produces a kernel module. However, all these approaches rely on *tracehooks* or *tracepoints* in the kernel where a check is done to see if the event is of interest, in which case an audit record is generated. The side effect of the branch that must be performed is that a cache miss results, slowing down performance even when no auditing is active. Our proposed approach using Linux kernel *markers* does not suffer from this.

V. Conclusions

Due to the continuous migration of organizations' critical information technology into the cloud, we believe that the cloud providers should make a concerted effort towards accountable clouds. Assurance of the security of their customers' data can be formalized by first-class SLA clauses. These can be supported in practice by a number of steps we introduce here, aiming to make cloud computing more accountable by proactively collecting forensic evidence. These steps include mapping laws into evidentiary requirements, targeting these into system-specific evidence descriptions, dynamically activating auditing as needed, cryptographically protecting the resulting evidence, distributing the audit trails so they cannot be repudiated and providing varying levels of auditing according to the level of forensic assurance needed by the customers.

References

[1] James P. Anderson, Computer security threat monitoring and surveillance, Technical Report, Fort Washington, PA, 1980.

[2] Stefan Axelsson, Intrusion detection systems: A survey and taxonomy, Technical Report 99-15, Chalmers University of Technology Department of Computer Engineering, March 2000.

[3] Adam Barth, Design and analysis of privacy policies, Ph.D. Thesis, Stanford University, 2008.

[4] R. Dingledine, N. Mathewson, and P. Syverson, Tor: The second-generation onion router, 13th Conference on USENIX Security Symposium, 2004.

[5] George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza Basrai, and Peter M. Chen, ReVirt: Enabling intrusion analysis through virtual-machine logging and replay, Symposium on Operating Systems Design and Implementation, 2002.

[6] Regional Computer Forensics Laboratory Program, Annual Report for Fiscal Year 2011, http://www.rcfl.gov/downloads/documents/RCFL_Nat_Annual11.pdf

[7] Chapman Flack and Mikhail J. Atallah, Better logging through formality: Applying formal specification techniques to improve audit logs and log consumers, Recent Advances in Intrusion Detection, 2000.

[8] Ian Goldberg, David Wagner, Randi Thomas, and Eric A. Brewer, A secure environment for untrusted helper applications, 6th Usenix Security Symposium, 1996.

[9] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel, PeerReview: Practical accountability for distributed systems, 21st ACM Symposium on Operating Systems Principles, 2007.

[10] Andreas Haeberlen, A Case for the Accountable Cloud, 3rd ACM SIGOPS International Workshop on Large-Scale Distributed Systems and Middleware, 2009.

[11] R. Hagmann, Reimplementing the Cedar file system using logging and group commit, 11th ACM Symposium on Operating Systems Principles, 1987.

[12] http://www.cms.hhs.gov/SecurityStandard/

[13] Kapil Jain and R. Sekar, User-level infrastructure for system call interposition: A platform for intrusion detection and confinement, ISOC Network and Distributed Systems Symposium, 2000.

[14] Renico Koen, The development of an open-source forensics platform, M.Sc. Thesis, University of Pretoria, 2009.

[15] Raffael Marty, Cloud application logging for forensics, ACM Symposium on Applied Computing, 2011.

[16] NILFS, http://www.nilfs.org/en/

[17] Rafail Ostrovsky, Amit Sahai, and Brent Waters, Attribute-based encryption with non-monotonic access structures, 14th ACM Conference on Computer and Communications Security, 2007.

[18] V. Prasad, W. Cohen, F. C. Eigler, M. Hunt, J. Keniston, and B. Chen, Locating system problems using dynamic instrumentation, Ottawa Linux Symposium, 2005.

[19] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiatowicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu, OpenDHT: A public DHT service and its uses, ACM SIGCOMM, 2005.

[20] Muriel Roger and Jean Goubault-Larrecq, Log auditing through model checking, 14th IEEE Computer Security Foundations Workshop, 2001.

[21] Keyun Ruan, Joe Carthy, Tahar Kechadi and Mark Crosbie, Cloud forensics: an overview, Advances in Digital Forensics, Vol. 7, 2011.

[22] Bruce Schneier and John Kelsey, Secure audit logs to support computer forensics, ACM Transactions on Information and System Security, 2(2), pp. 159-176, May 1999.

[23] W. Olin Sibert, Auditing in a distributed system: Secure SunOS audit trails, 11th National Computer Security Conference, 1988.