Achieving Accountability with Distributed Data Usage Control Technology

Alexander Pretschner
Technische Universität München, Germany
pretschner@in.tum.de

ABSTRACT

Distributed data usage control technology enforces obligations on future data usage in a preventive or a detective manner. The goal of preventive enforcement is to make sure that a data usage policy is adhered to. The goal of detective, or optimistic, enforcement is to maintain a log of policy violations, which directly provides technical means for accountability. Depending on the underlying trust model, available technology, and economic considerations, either enforcement strategy may be applicable in a specific context. Technically speaking, these two seemingly different strategies can be implemented by identical technologies. In this position paper, we present such technology for data usage control enforcement that turns out to be a natural candidate for systems that enable accountability.

1. INTRODUCTION

Distributed data usage control (UC) policies stipulate rights and duties concerning the future usage of data [10, 11]. Preventive enforcement is regularly used in systems for digital rights management where the value of the protected data items is comparably small, e.g., 99 cents for a song. The underlying assumption is that the recipient of a data item cannot be trusted. In contrast, detective enforcement is used in situations where a baseline of trust has already been established. This is the case, for instance, in outsourcing contracts, or if non-disclosure agreements are signed [12]. In this paper, we present UC enforcement technology that can be used for both detective and preventive enforcement, and thus for implementing accountability. To do so, we present UC requirements in §2 and show how they can be enforced by UC technology in §3, which includes a discussion of policies, enforcement, data flow tracking within and across layers of abstraction and within and across single machines, declassifications based on data quantities, provenance tracking, and ways to protect the infrastructure itself. We connect this technology to accountability in §4 and conclude in §5.

Position paper for the 2nd International Workshop on Accountability: Science, Technology and Policy at MIT, Boston, January 30th, 2014. Workshop website `http://dig.csail.mit.edu/2014/AccountableSystems2014/`.

.

2. REQUIREMENTS

In addition to classical attribute-based access control and separation of duty requirements, UC policies stipulate temporal constraints ("never disseminate," "delete after thirty days," "retain at least five years," "don't publish before 6pm," "don't show before explicit consent is given"), spatial constraints ("don't store outside the EU," "must not leave the hospital"), cardinality constraints ("not more than two copies," "show at most five times"), duties or rights triggered by events ("upon notification, delete data," "don't disseminate after notification"), quantitative constraints ("release at most 10% in total," "never release more than .1% per day"), organizational and technical environment constraints ("only store in environments certified according to ISO 27001," "only store in systems with firewalls certified according to the common criteria," "don't release to a mobile device"), and purpose ("only use for statistical purposes," "don't show in public"). This list is open to future additions; we believe it captures the essence of data usage policies and, therefore, a majority of the constraints relevant for accountability.

In addition to these more technical requirements, data UC requirements include the need to distinguish between data and representations of the data. If a colored photo stored in file f must not be copied, then one likely wants that *all* representations of that photo should not be copied: no screenshots of the photo when displayed on a screen, no verbatim copies of file f, no black-and-white transformations of the photo, no versions of the photo in a different format, possibly no cropped versions of the picture, etc. These examples also show that one may want to specify that derivations of a data item are not be legal, which becomes particularly relevant in aggregation contexts for statistics or big data. This then implies the need to specify declassifications of data [16] ("if data item is aggregated with >1000 other items, the original policy does not apply").

Then, in addition to the above-mentioned requirements as *constraints* on data usage, the usage itself must be defined. "Deletion" comes with various technical semantics [15] pertaining to the actual degree of deletion (removing a FAT entry as opposed to overwriting a file randomly 1000 times), the location (on one hard disk as opposed to all archival tapes in addition), the nature of deletion (throwing away the key of an encrypted file as opposed to delting the file). Deletion becomes particularly challenging if data exists in various representations at multiple, physically possibly disjoint, locations. Similarly, "copying" needs to be defined. One approach is to specify that copying has taken place whenever a new representation of a data item is created. Given that ev-

ery information processing device creates copies of data by simply loading it to memory, more fine-grained definitions of "copy" need to be defined. As in the case of tracking representations, sensible declassifications need to be understood, specified, and integrated into the policy language.

Similar to the need of precisely defining "usage," "data" needs to be defined. For instance, precisely what data constitutes the profile of a social network? Does this include, for instance, the fact that a page is liked? Does it include comments or traffic data?

The example of a social network profile also motivates the need for policy compositions. Arguably, the photos and posts pertaining to a profile are equipped with a policy designed by the owner of the profile. Comments from other parties that relate to parts of the profile, in contrast, are likely to be equipped with different, possibly (and likely) conflicting schemas. The policy language's semantics hence need to come with useful and pragmatic definitions of composition and/or conflict resolution.

Finally, since data usage policies are security policies, one may want to parameterize them by the risk incurred if the policy was violated. This includes estimates of the impact of the violation and possibly its likelihood, two parameters that turned out to be hard to measure and/or predict, in part because these parameters are highly situation-dependent.

3. DISTRIBUTED DATA USAGE CONTROL

3.1 Policies

The first set of requirements of Section 2 (constraints) can be formalized using a combination of temporal and first order logics, as proposed in various forms by several authors [18, 5, 1]. At the logical level, constraints concerning (attribute-based) access control, separation of duties, environment and purpose can be pushed into an external predicate $eval(\cdot)$ that provides the necessary information and that is outside the scope of the logic itself. For instance, requirements pertaining to purpose can be pushed to the *eval* predicate that encapsulates complex logics to decide if a data usage adheres to a given purpose [17].

When specifying constraints on data usage, it is rather simple to distinguish between usages of one specific representation and the set of all representations, provided that respective tracking technology is in place. This kind of policies can be specified, for instance, by the language presented by Lovat et al. [14], also for quantitative constraints [8].

These policies need to be augmented by declassification policies. A large body of work on declassification policies exists but has, to our knowledge, not yet been integrated with data usage policies.

Because of the necessity of defining semantics for "usage" and "data", there must be sophisticated refinement schemas as described by Kumari et al. [7]. These early experiences show that these definitions are extremely complex, and that there is a need for methodological support.

Clearly, a user-friendly concrete syntax must be provided to hide the details of the abstract syntax of first order temporal logics. One approach here is to use standardized templates in natural language that are particularly appealing because in our experience, most real-world policies tend to be rather simple and captured by a set of five to six templates. Similar findings have been discussed for formal specifications of reactive systems [4].

3.2 UC Enforcement

UC policies can be specified declaratively and operationally. A declarative policy "unanonymized data must never leave the system without logging" can preventively be enforced by several operational strategies [13]: by anonymizing data (modification), blocking send events (inhibition), and logging the send action (execution). It can also detectively be enforced by detecting and logging policy violations. Either way, enforcement relies on checking a condition, which is usually embodied in a policy decision point (PDP), and also on taking one of the specified actions, which is usually embodied in a policy enfocrement point (PEP). Depending on the deployment context and on the policies, both runtime verification and complex event processing technologies can be used to implement PDPs.

3.3 Data Flow Tracking

Because of the need to protect data in the sense of protecting all representations of the data, it is necessary to track the various representations of data. This problem has been studied extensively in the context of information flow, or data flow, tracking, and can be performed both statically and dynamically. Practical problems usually concern the identification of sources and sinks; the problem of protection boundaries; and, more problematically, the problem of overapproximations, specifically so if data flows are considered that are a result of branching over a secret. If a policy language distinguishes between data and representations [14], then the respective monitoring technology (PDP+PEP) needs to be extended by a policy information point (PIP) that maintains the mapping between data and all its representations and that is consulted by the PDP.

3.4 Multiple Layers of Abstraction

If a picture (data) exists in the form of several identical files (representations), then these representations reside at the same level of abstraction, namely the operating system, as does a file that contains the same picture in a different format. However, the picture may also exist as a DOM object in a browser application or as a pixmap on a screen. In this case, data flow tracking technology must cater to the problem of data being converted into representations at several layers of abstraction (OS, browser, window manager, etc.). In general, this can only be done at runtime; in a respective system, there are hence multiple PEPs, one per layer of abstraction. One proposal to perform this cross-layer tracking is described by Lovat et al. [14].

3.5 Distributed Systems

The above considerations assume that representations of a data item exist within one machine only. However, the idea easily generalizes to distributed systems where a remote machine is considered as another (hierarchical) layer of abstraction, as proposed by Kelbert et al. [6]. The idea here is that whenever data is shipped to another system, the respective UC policies are shipped along with it.

3.6 Quantitative Measurements

Because of the overapproximations induced by information flow tracking, declassification [16] has long been acknowledged to be a key enabler for information flow tracking technology. One possbility to perform declassifications is on the grounds of quantities, as proposed by Lovat et al.

[8]: data is considered to be non-sensitive if only a "small" amount of data has been leaked.

3.7 Provenance Tracking

The general idea of performing data flow tracking within and across systems as discussed above immediately provides one possible way to perform provenance tracking. *Within* one system, one policy simply specifies that every access needs to be logged. If data is passed *across* systems, then not only the policy is shipped along with the system, but also information on its prior usage and the locations where it was stored [2].

3.8 Securing the Infrastructure

In order to enforce UC requirements, the sketched distributed data UC infrastructure must be deployed at the sites of all communication partners. In general, this seems like a strong assumption which may, however, be justified in specific scenarios such as company-owned devices. In a sense, app stores like iTunes already provide such half-closed environments. However, in any case, it must be ensured that the infastructure is not tampered with. If the user of a usage-controlled system can be assumed not to possess root privileges, this appears feasible. If the user is a privileged root, however, then protection becomes more problematic and, depending on the desired level of protection, needs to be rooted in hardware with tamper-proof logs [9] for external auditing; or can, with fewer guarantees, be done via obfuscation and white-box cryptography [3].

4. ACCOUNTABILITY

So far, we have concentrated on possible UC requirements and their formalization in policies; on how to technically enforce policies (preventive enforcement) or detect whether or not they are adhered to (detective enforcement); and how to make sure that the respective monitoring technology is not tampered with. What is missing is the link with accountability. We adopt the definition that "an accountable system is one that can be examined to assess whether policies (e.g., system specifications, information use and disclosure policies) are being followed, and possibly facilitates holding individuals or institutions responsible in the event that the policies are violated.[1]" In this paper, we tackle the technical perspective only and therefore do not discuss how to hold institutions responsible. We thus need to show how to use the introduced UC infrastructure for (technical) accountability. In fact, this is straightforward: If only declarative policies are used in conjunction with detective enforcement, policy violations can simply be logged. If operational policies are used for preventive enforcement, we can use the same technology to log the fact that potentially policy-violating user actions have been attempted. In terms of ensuring system-wide accountability, we can use the results described above in terms of cross-layer data flow tracking and cross-machine provenance tracking. Using the (then possibly distributed) logs, we can re-use the UC infrastructure to issue warnings if specific conditions are met.

In the abstract, we have argued that detective and preventive enforcement essentially rely on the same technology. The reason is that the PDPs from §3.2 are independent of the respective PEPs; and logging or notifying about policy violations can be seen as preventive enforcement strategies for executing specified actions. In other words, if a policy violation can be observed, then it is in many cases also technically possible to prevent its violation, and vice versa. Practically speaking, this is not always the case, however: if system calls at the level of an operating system, for instance, are blocked, many processes will crash because the respective code has not been implemented defensively. In this case, UC technology is too intrusive. Moreover, to avoid an undesired situation, a preventive strategy may need to "look into the future" to establish whether or not the undesired situation would actually occur if the data usage request was granted. This is often impractical for, say, data bases because the need of rollbacks. However, it is also possible to specify policies on the grounds of undesired events rather than undesired situations, i.e., system states.

In sum, apart from several challenges discussed in the next section, existing UC technology seems fit to be used for accountability purposes.

5. CHALLENGES AND CONCLUSIONS

There are several challenges with technical realizations of accountability with UC technology. A major one is the establishment of adequate levels of abstraction: what exactly does usage mean? Usage at the level of system calls is likely too fine-grained [7]. This challenge comes along with scalability concerns both in terms of space (storage of usage information and provenance graphs) and time (for monitoring a system).

Then, if data leaves the digital realm, e.g., by photographing a screen, protection becomes more difficult. This media break, however, can sometimes be alleviated by means of watermarks or infrared LEDs or special monitors.

Technically speaking, it is not always simple to perform conflict resolution between policies, particularly so if they pertain to independent parts of a data item but make statements about the entire data item.

Usability of course is far from trivial: who would specify the policies, who would determine the level of abstraction for monitoring, and how would policy violations be communicated?

It is not entirely clear who should be allowed to specify a policy. Potential candidates are owners of the data, possessors of the data, or further parties, e.g., a company, the provider of a service, or the government. The complexity of ownership of data is exemplified by traffic data in a social network: is this traffic data owned by the telecommunication provider? By the provider of the social network? By the person who accesses a profile? By the person whose profile is accessed? Any combinations hereof?

From a less technical perspective, data protection quickly becomes an issue. If provenance tracking technology for enabling accountability is fully implemented, then this means that data from different sources can be combined. While European data protection legislation contains the right to be informed about which personal data is stored by a company or government, implementing this right may create a data protection problem that is larger than the one we set out to handle: combined data is in many cases more sensitive than the sensitivity of the atomic parts alone. This then directly implies the need for controlling the usage or users of the accountability system, a recursive implementation of the above ideas.

[1] `dig.csail.mit.edu/2014/AccountableSystems2014/`

References

[1] D. Basin, F. Klaedtke, and S. Müller. Monitoring security policies with metric first-order temporal logic. In *Proceedings of the 15th ACM Symposium on Access Control Models and Technologies*, SACMAT '10, pages 23–34, 2010. ISBN 978-1-4503-0049-0.

[2] C. Bier. How usage control and provenance tracking get together - a data protection perspective. *2012 IEEE Symposium on Security and Privacy Workshops*, 0:13–17, 2013.

[3] S. Chow, P. A. Eisen, H. Johnson, and P. C. van Oorschot. White-box cryptography and an aes implementation. In *Selected Areas in Cryptography*, pages 250–270, 2002.

[4] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 411–420, 1999. ISBN 1-58113-074-0.

[5] M. Hilty, A. Pretschner, D. A. Basin, C. Schaefer, and T. Walter. A policy language for distributed usage control. In *ESORICS*, pages 531–546, 2007.

[6] F. Kelbert and A. Pretschner. Data usage control enforcement in distributed systems. In *CODASPY*, pages 71–82, 2013.

[7] P. Kumari and A. Pretschner. Model-based usage control policy derivation. In *ESSoS*, pages 58–74, 2013.

[8] E. Lovat, J. Oudinet, and A. Pretschner. On quantitative dynamic data flow tracking. In *Proc. CODASPY*, 2014. To appear.

[9] R. Neisse, D. Holling, and A. Pretschner. Implementing trust in cloud infrastructures. In *CCGRID*, pages 524–533, 2011.

[10] J. Park and R. Sandhu. The ucon abc usage control model. *TISSEC*, pages 128–174, 2004.

[11] A. Pretschner, M. Hilty, and D. A. Basin. Distributed usage control. *Commun. ACM*, 49(9):39–44, 2006.

[12] A. Pretschner, F. Massacci, and M. Hilty. Usage control in service-oriented architectures. In *TrustBus*, pages 83–93, 2007.

[13] A. Pretschner, M. Hilty, D. Basin, C. Schaefer, and T. Walter. Mechanisms for Usage Control. In *Proc. ASIACCS*, pages 240–245, 2008.

[14] A. Pretschner, E. Lovat, and M. Büchler. Representation-independent data usage control. In *Proc. SETOP/DPM*, pages 122–140, 2011.

[15] J. Reardon, D. A. Basin, and S. Capkun. Sok: Secure data deletion. In *IEEE Symposium on Security and Privacy*, pages 301–315, 2013.

[16] A. Sabelfeld and D. Sands. Dimensions and principles of declassification. In *CSFW*, pages 255–269, 2005.

[17] M. C. Tschantz, A. Datta, and J. M. Wing. Purpose restrictions on information use. In *ESORICS*, pages 610–627, 2013.

[18] X. Zhang, J. Park, F. Parisi-Presicce, and R. S. Sandhu. A logical specification for usage control. In *SACMAT*, pages 1–10, 2004.