# Some thoughts on accountable systems

David Clark

January 1, 2014

As defined in the call for the workshop, 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.

The following are some thoughts about a framework to think about this challenge.

## Protection from malicious behavior

In setting the design requirement, it is important to decide whether the goal is to detect activities that are intentionally crafted to evade the controls (e.g., activities that are explicitly malicious) or just to detect "honest mistakes". In my view a system that only deals with "honest mistakes" is not going to be of much practical value, and we should try to understand how to build a system that deals with malice.

By using the word "policy", the line of thinking may be directed towards issues such as privacy. However, with respect to business systems and data processing, there are a larger set of policy goals that have been of concern for many years. An obvious example is prevention of fraud. There are well-known process methods to limit fraud, including an online implementation of a "separation of duties" workflow.

There are specific commercial systems that are used to impose these sorts of constraints, but the general design approach is that the data being protected cannot be open to arbitrary manipulation by arbitrary software. Systems such as the familiar Unix variants store data in the file system, and any program that can parse the format of the data can manipulate it. This was a core design objective of "Unix-class" systems, but it means that the OS can offer no protection with respect to policy enforcement or data integrity. To meet that class of objective, the data has to live inside a trusted context that constrains what can be done with the data. This approach is more typical of commercial data processing systems. As well, systems such as iOS store information "inside" the application. Data can be moved from one application to another only in ways that are implemented by the application, and the only data manipulations permitted are those that are coded into the app. This design (to traditional programmers) limits flexibility, but may lead to a more secure and predictable user experience.

The language used to describe those systems is sometimes not about "accountability", although that is their goal (think about fraud), but instead the language of data integrity. An (over)-simple specification of data integrity is that

data does not get modified. But in business practices, data is transformed and new data derived from prior data. In this context, integrity means exactly that there are only some transformations on data that are valid. In this view of accountability, the role of the operating system is to restrict access to a give piece of data to an authorized set of applications. It is the role of the application to enforce and validate operations on the data.

The design process then takes one of two paths. In one path, data integrity is enforced by only allowing known and trusted tasks to manipulate the data. These tasks are written and audited, and are deemed "trustworthy". A co-author and I wrote a paper in 1987 that formalizes this class of system[1]. This approach, which reflects a normal commercial practice, implies that users of the data cannot perform arbitrary manipulations, which may restrict the creative and flexible use of the data.

## Run-time validation and enforcement

In the other approach, the users are given more flexibility, with the ability to craft new operations on the data, but there is some sort of "observer" process that audits the operations in real-time to detect inappropriate actions. This would seem to be the interesting design space for future systems. In general (e.g., if the capability given to the user is fully general), then this goal would seem to fall into the "impossibility" category. The application/system protecting the data must implement some constraints to make the auditing possible. As a research question, the goal must be to define the degree of flexibility that can still be audited for integrity.

An early example of such a constraint is the *-property of the Bell-LaPadula security model from the early 1970s. The *-property prevents the de-classification of data by associating with any process the highest classification level of any of the data the process has so far read. The process is then prevented from writing to a data set with a lower classification, a limitation implemented by a trusted system component called the reference monitor. Of course, this scheme recognized the need to reduce the classification level of data under certain circumstances, and once again appeals to the idea of a "trusted process" or "trusted agent", whether human or automated.

The levels and categories of the classification system are only one way, and a crude way to capture classes of policies. In the early days of the Bell-LaPadula work, it was hoped that the framework could capture any policy, but this hope has proved false. So one way to frame a research agenda is to propose alternative semantics for a reference monitor, which can track some simple property of the data being read, and then impose practical constraints on the process that has carried out the read. The more complex the property being tracked, of course, the more complex the

---

[1] Clark, D., and Wilson, D., "A Comparison of Commercial and Military Computer Security Policies", *Proceedings of the 1987 IEEE Symposium on Security and Privacy,* IEEE, Oakland, CA, pp. 184-194, April 1987.

semantics of the reference model. To enforce the *-property, the reference model needs to read the security classification metadata of every file read or write. Is there an equivalent specification that can deal with other classes of policy, in particular policies about proper use of PII?

## Tracking identity

Another aspect of a system that provides accountability is that some accountable entity (called the *principal* in security terminology) must be associated with the action in question. Within a single system, users are normally required to authenticate themselves at login, which provides a binding between a process and a principal. However, in the network context, connections across the network are not associated (at the connection level) with a login. As a result, many actions initiated across a network cannot be associated with a responsible principal for the purposes of accountability. This has led to calls in Washington for "an accountable Internet". However, this goal could be accomplished in more than one way. One approach would be to add identity information (that links back to a responsible principal) to the packet-level mechanisms of the Internet. Another approach is to delegate to the application the responsibility for obtaining and validating identity information. A co-author and I have argued that the former approach would be ineffective and harmful[2], and that the responsibility must be delegated to the application, since only at the application level are the needs for identity and accountability defined.

As we build systems that provide accountability it is important to remember that the needs are not uniform, and that on occasion *lack of accountability* is the desired goal.

A final point about identity and accountability is that some workflow policies such as separation of duties require that parts of a task be carried out by at least two different people. For this workflow policy to be effective, it is necessary that one human not be able to enter the system with two different identities. Enforcing this is probably something that has to happen outside the system, at the level of human administration. As a result, enforcing this for network-based interaction may be difficult, since it may not be possible to audit the enforcement of this policy at remote sites.

---

[2] Clark, D. D. , Landau, S. "*Untangling Attribution*" Harvard National Security Journal, Vol. 2, Issue 2 (2011);