Towards accountable information systems

Peter Druschel, *Max Planck Institute for Software Systems (MPI-SWS)*

Joint work with Eslam Elnikety, Deepak Garg, Aastha Mehta, Anjo Vahldiek

In this extended abstract, we sketch our current work on enforcing declarative policies for data confidentiality, integrity, and access accounting in general computer systems. Enforcing data policies is a step towards the long-term vision of fully accountable information processing systems.

In our system model, each data object has an attached policy, which reflects the data subject or owner's choices, the data handler's policy, as well as applicable laws. The policy is then enforced regardless of how and where the data is processed and stored. Moreover, authorized parties can obtain cryptographic certificates that attest data objects and the policy in effect.

A policy-enforcement and attestation facility of this type enables stakeholders to specify the policy in effect for (a class of) data objects in a high-level, declarative fashion. Policy compliance rests only on the integrity of an enforcement kernel with a small trusted computing base and attack surface, as well as any explicit policy dependencies (e.g. time, user credentials).

Stakeholders can reason about compliance without requiring access to, or an understanding of, data handlers' computing infrastructure. At the same time, data handlers can verify and demonstrate their compliance by disclosing their enforcement kernel.

Policies can specify who can access a data object (access control), in which compute environment (remote attestation), and under what conditions (e.g., time). Update policies may specify structural integrity constraints (invariants) within and across objects. Finally, policies may specify mandatory side effects for accesses to a data object, such as access logging to a log file with an append-only policy.

1 Guardat

As a first step, we have designed and implemented *Guardat*, an architecture that enforces data policies at the *storage layer*. Users, application developers and system administrators can provide per-object policies to Guardat. Guardat enforces these policies and provides attestations about the state of stored objects. With Guardat, the data integrity, confidentiality and access accounting rules for a collection of objects can be stated as a single declarative policy. Policy enforcement relies only on the integrity of the Guardat controller and any external policy dependencies; it does not depend on correct software, configuration and operator actions in other parts of a system. Guardat allows developers, system administrators and third-party hosting platform providers to enforce concise, system-wide data protection and accounting policies based on a small trusted computing base, and to demonstrate their compliance to any party that trusts the Guardat enforcement component.

Motivation As the volume and value of digital assets stored in persistent mass storage keep increasing, so do the risks to the integrity, confidentiality and usage accountability of said data. Computer and storage systems are increasing in complexity, exposing data to risks from software bugs, security vulnerabilities and human error. In addition, data is increasingly processed and stored on third-party platforms, introducing additional risks like unauthorized data use by third parties.

In existing systems, the confidentiality, integrity and usage accountability of persistent data depend on the absence of design errors, malware and bugs in all layers of software (including device drivers, storage subsystem, file system and operating system). For data stored on third-party platforms, data confidentiality and integrity, as well as proper accounting of data use, additionally depend on the absence of malice and error on the part of the third-party provider.

Design Guardat provides policy enforcement and data certification at the storage layer. With Guardat, users, developers and administrators can state the integrity, confidentiality, and accounting rules for a collection of data objects using a concise, declarative policy language. Applications communicate with Guardat through secure channels, tunneling through untrusted system layers like storage servers or hosting

platforms. Applications send policies, commands and evidence of policy compliance (e.g., proof of authentication) to Guardat and request attestations of stored data and their policies from Guardat. Guardat enforces the policies while relying only on its own interpreter and enforcement logic and any explicit policy dependencies, thus minimizing both the computing base relied upon for enforcement and its attack surface.

Policies A Guardat policy specifies the conditions under which an object may be read, updated, or have its policy changed. These conditions, written in a simple but expressive declarative language, may depend on client authentication, the initial and final states of the object (size and content) in an update transaction, or signed statements by external trusted components (certifying, for instance, the current wall-clock time or the configuration of the client's compute platform). Guardat stores the policy as part of its own metadata and ensures that each access to the object complies with the policy.

Following are some example Guardat policies that mitigate important threats: System binaries can be protected from viruses through a policy that allows modification only when the updated binary is signed by a trusted party; system log corruption and tampering can be avoided through a Guardat-enforced append-only policy; accidental deletion or corruption of backup data can be prevented by a policy that prevents modification for a specific period of time; confidentiality of a user's private data can be enforced by allowing reads only in a session authenticated by the user's public key; and, accesses to a data object can be permitted only if a corresponding access record is added to an append-only log file, enforcing mandatory access logging.

While these policies can be readily implemented in higher software layers, the merit of using Guardat is that the policy applicable to a collection of data objects can be specified using a concise, declarative language, and enforced by a small trusted computing base (TCB) with a small attack surface. Guardat complements existing techniques for ensuring the reliability of data processing systems, including software testing, verification, security auditing, sealed data and trusted computing. While no technique can provide comprehensive protection, Guardat provides a safety net that protects a system's persistent data from a wide range of threats. Moreover, Guardat can demonstrate compliance with client and provider policies, as well as applicable laws to any party that trusts Guardat.

Guardat can provide additional benefits in multi-party environments where all parties trust Guardat, e.g., a client storing her data at a hosting provider, or a service provider allowing caching of parts of its database on a client device. Here, Guardat can enforce the data owner's policies on third-party data accesses, and the data holder can use Guardat to demonstrate its compliance with client and provider policies.

Design principles The Guardat design is based on three principles. First, enforcing policy at the storage layer minimizes the TCB and its attack surface. Second, a simple, declarative policy language allows the concise specification of all policy related to a collection of data objects. Third, the policy language supports a small set of declarative primitives expressive enough to specify the access policy, leaving it to untrusted code to provide the mechanism required to satisfy the policy.

Implementation Guardat can be implemented in different ways, depending on the deployment scenario and threat model. For instance, an integration into a VMM or secure OS enforces policy on all VM/application storage accesses, but requires the VMM or OS to be trusted; an integration into a SAN server enforces the policy on all accesses from network clients as long as the server is trusted; while an integration into the controller of a storage device or host adapter enforces the policy on all accesses as long as the controller chip is not compromised.

Preliminary results We have built an initial TS prototype by extending the open-source iSCSI Enterprise Target (IET) SAN server. The server contains a conventional hard-disk drive to store data and a small solid-state drive to store metadata including policies. To export the Guardat functionality to applications, we extend the familiar POSIX file API. A library linked with applications implements this API and communicates with Guardat devices through IOCTL calls (which the OS and file system pass to the Guardat device driver uninterpreted), thus maintaining compatibility with existing, unmodified file systems. Prelimiary experimental results based on micobenchmarks, as well as an Apache web server, indicate that Guardat can enforce polices with low overhead.

2 Extensions

In ongoing work, we are developing extensions to enable the use of Guardat in hosted Cloud environments. The goal is to enable Cloud operators to en-

force user-provided integrity, confidentiality and accounting policies, their own privacy, integrity and data retention policies as well as applicable laws, and to enable them to demonstrate their compliance.

The existing Guardat design and prototype enforces per-object policies stored on individual Guardat devices, effectively constraining information flow to and from the device according to a single policy provided by the data owner. In a Cloud storage environment, this design is too limited. For instance, the storage operator is unable to control the replication and migration of user data in a policy compliant manner.

We are designing extensions to the Guardat policy language, interface and implementation suitable for a distributed third-party storage environment. The extensions (i) allow customers to specify policies for the replication and migration of data and guarantee policy compliance, while enabling an untrusted storage provider to orchestrate the replication and migration of user data; (ii) enable customers to specify rich accounting policies like mandatory access logging in the presence of data replication and migration; and (iii) enable storage providers to enforce their own policies as well as applicable laws using Guardat. The extensions enable policy-compliant replication and migration of user data across Guardat devices, orchestrated by the untrusted provider system.

In further work, we are extending the existing Guardat capability to remotely attesting a client's compute platform to also assert certain properties of an otherwise untrusted client computation. For instance, if a trusted hardware, OS or VMM platform attests that a client computation can perform I/O only to and from Guardat devices, then it is possible to ensure policy compliance even while granting such an encapsulated but untrusted computation access to data. With this extension, it will be possible to enforce data confidentiality, integrity and accounting policies, as well as provenance and declassification policies, on an entire multi-stage, distributed data processing system.