Efficiently enforcing data confidentiality and integrity in large-scale distributed data processing systems

Eslam Elnikety[1], Anjo Vahldiek, Aastha Mehta, Deepak Garg, Peter Druschel
MPI-SWS

Increasingly, data is processed in data centers using parallel and distributed algorithms with complex, multi-stage workflows. These systems enable powerful data analysis, search and inference capabilities at massive scale. Hadoop, for instance, was originally designed for web search and analysis, and is now used widely in many industries. However, the complexity of the parallel algorithms employed in these systems increases the risk of bugs and misconfiguration. These could in turn lead to accidental data leaks, violating customer requirements, provider policy or legal mandates for confidentiality, integrity and access accounting. This risk is aggravated when internal corporate data and customer personal data is processed along with public data, which is often the case.

In current systems, policies are specified in many system components and processing stages; compliance depends on correct software and configuration throughout the system, and the absence of operator error. To address this problem, we are designing Thoth, a system that efficiently enforces data confidentiality, integrity and access accounting policies in distributed, multi-stage processing environments. All policy affecting a particular data object (e.g., file) is stated in a concise, declarative language and enforced by a collection of trusted virtual machine monitors, which encapsulate the system's distributed and parallel compute stages.

At the core of Thoth is an interpreter for a simple, expressive, declarative policy language. A policy associated with a file specifies the conditions under which the file can be read, updated, destroyed or declassified. The policy can refer to authentication credentials, external facts (e.g., wall-clock time as reported by a trusted time source) or the current state (content) of files. Both the input and output files, as well as intermediate files of a multi-stage computation can be associated with policies, enabling data subjects, application designers and providers to express a wide range of policies.

Thoth enforces polices by tracking and mediating the input and output of virtual machines. Conceptually, a VM is tagged with the policies of all input files it reads. When the VM writes to a temporary file, that file receives a policy as restrictive as the union of the VM's input file policies. When a VM writes to a file with a pre-defined policy, that policy must be no less restricitve than the union of the VM's input policies and the output must satisfy the files' integrity policy; else, the output is rejected.

Thoth deliberately employs a very coarse-grained form of information flow tracking at the VM level. As a result, Thoth can enforce policies with the efficiency required for large-scale, compute-intensive applications. In exchange for the efficiency of coarse-grained tracking, existing applications generally have to be refactored to some extent to prevent over-tainting. In a system that observes the policies associated with its data, such refactoring is always possible in principle and often do-able with moderate effort.

Example: Policy-compliant search Engine. Search engines typically index public web pages along with personal customer information (OSN profiles, emails), and provider corporate data. A bug or misconfiguration in any stage of the engine, which is optimized for massive scale and performance, can lead to the accidental disclosure of corporate or personal data.

Using Thoth, (classes of) input files carry polices that specify who can read the file, and an optional set of keywords that should be present in a query for the file to be included in the results. The final stage of the engine can deliver search results of a client only if the policy of each file included in the results is satisfied, preventing the accidental leakage of private information or inappropriate search results due to bugs anywhere in the engine outside the virtualization platform.

Example: Policy compliant report generation. Many organizations produce public versions of periodic (e.g. quarterly) reports, which are automatically generated from a variety of internal data sources by a complex processing pipeline. A bug or misconfiguration in any part of the system can produce inconsistent results, include information from an incorrect time period, or leak confidential information.

Using Thoth, input files carry confidentialy, integrity, and disclosure policies. The final report output file has a pre-defined policy, which allows public access, but allows only data to be written that meets specific requirements for provenance, consistency, and confidentiality. For instance, a policy might require that the report is derived only from public information produced in 2012, and that the information must include certain legal disclaimers.

Eslam Elnikety, Anjo Vahldiek and Aastha Mehta are students.
[1]Will present the poster.