Reducing Data Leakage on Personal Data Management Systems

Robin Carpentier, Iulian Sandu Popa, Nicolas Anciaux

PDMS: Personal Data Managᵗ Systems

- Boosted by legal practices (GDPR, Data Altruism)
- Individuals assemble their data under their control
- Conflicting objectives:
  - Security on untrusted user device
  - Extensiveness of data-oriented computations
- PDMS: Apps "move" to personal data
  - VS. personal data sent to remote (cloud) services
- Trusted Executions Environments (TEE) at user side
- But TEE solutions assume trusted computation code
  - ➔ contradicts extensiveness !

Query scope & security building blocks

- Queries produce limited-length results (e.g. aggregate) using advanced and specific data-oriented computations
- Security building blocks to limit leakage:
  - Stateless Data tasks: no data persistency
  - Deterministic Data tasks: no source of randomness
  - Circumscribed leakage: Several Data tasks running on objects partitions
- Compromises and trade-offs to be explored:
  - Larger partitions → less Data tasks → efficiency
  - Smaller partitions → less leakage
  - Singleton (one object) partitions → minimal leakage
  - Reuse previous (indexed) results → efficiency

Goals

- Build a secure architecture supporting untrusted extension code for specific data-oriented computations
- Reduce information leakage in legitimate query results with a generic approach: no semantic analysis of the processing code nor results

Proposed architecture

- Secure Core (Isolated): Small trusted computing base, inextensible. Sole entry point to manipulate data
- Data Tasks (Isolated and Confined): Untrusted code extensions, attested by the Core
- Apps (Untrusted): Granted access to computation results, but not to raw personal data

Implementation and performance

- PDMS code with OpenEnclave, evaluated on SGX
- First measures on aggregates for tabular data
- Main conclusions:
  - Data-oriented computations remains bottleneck
  - Security building blocks introduce small overhead