Advanced Systems Security: Integrity

Trent Jaeger
Systems and Internet Infrastructure Security (SIIS) Lab
Computer Science and Engineering Department
Pennsylvania State University
Data Integrity

• What is data integrity?
  ‣ What do we need to do to ensure data integrity?
Integrity

• List some items that have integrity
  ‣ What is the source of their integrity?
Integrity

• List some items that have integrity
  ‣ What is the source of their integrity?

• Forbes “Most Trustworthy Companies”
  ‣ “In order to rank companies from the most to the least trustworthy, we look at over 60 different governance and forensic accounting measures…”
  ‣ Not likely to fail, transparent, …

• Academic Integrity
  ‣ Behavior complying with a code of conduct and ethics
Integrity in Software…

- What do expect for integrity of software?
... Impacts Data Integrity

- How does software integrity impact data integrity?
Least Privilege

• *The protection mechanism should force every process to operate with the minimum privileges needed to perform its task.*

• Due to Saltzer and Schroeder (of Multics project)

• One of many “design principles” in their paper “The Protection of Information in Computer Systems” (1975)

• Others
  ‣ Principle of Psychological Acceptability
  ‣ Principle of Fail Safe Defaults
Least Privilege

• How to compute least privilege?
  ‣ Aim: Determines the permissions required for the program to run effectively

• Run the program and see what permissions are used
  ‣ Proposed for a system called Systrace
  ‣ SELinux audit2allow: take denied permissions and add them to policy
  ‣ AppArmor Profile Wizard: Build an approximate profile statically and
Least Privilege

- Is a good goal because...
- Is a poor goal because...
- Can we use it to verify a policy is secure?
Least Privilege

• Is a good goal because…
  ‣ Unnecessary permissions lead to problems (confused deputy)
  ‣ Accounts for function

• Is a poor goal because…
  ‣ Task permissions may conflict with security
  ‣ How do we know when a permission is necessary, but makes the system insecure?

• Can we use it to verify a policy is secure?
  ‣ No. It defines a policy based on function, not security.
Information Flow for Integrity

- Another approach looks at the authorized flow of information among processes via objects
Idealized Security

- **Biba Integrity**
  - **Integrity requirement:** Do not depend on data from lower integrity principals
  - Only permit information to flow from high integrity to lower integrity
  - E.g., Can only read a file if your integrity level is dominated by or equal to the file’s

![Diagram showing flow from Low to High]
Practical vs. Ideal

- Do these idealized approaches based on information flow enable practical realization of OS enforcement?
- Secrecy is possible in some environments
  - Implemented in a paper world, previously
- Integrity has not been realized in practice
  - Many processes provide high integrity services to others
- Result: Depend on many applications to manage information flows
Assured Guards

• What do we do if a system needs an information flow from low integrity to high?
  ‣ E.g., reading from a network socket

• Not authorized by Biba
  ‣ Unless subject is fully assured to upgrade to high integrity or discard low integrity data
  ‣ Called a guard

• What does “fully assured” mean?
LOMAC [Fraser 2000]

- Subjects and objects have an integrity label
  - Level and category in a lattice policy

- When subject reads an object of a lower integrity label in lattice
  - Subject’s label is lowered to that of object
  - Define subject’s label in terms of objects accessed

- When subject writes to an object of a higher integrity label in lattice
  - Write is denied
  - Read is still allowed
Biba vs LOMAC

• What is allowed and what is the resultant label?
  ‣ Lattice A → B → C

• Subject at A reads object at C
  ‣ Biba?
  ‣ LOMAC?

• Subject at C writes object at A
  ‣ Biba?
  ‣ LOMAC?

• Subject at C reads from object at A
Self-Revocation

- Can cause revocation of own access to objects in LOMAC

Step 1: initial state.

- ps
- pipe
- grep

Step 2: ps reads file.

- ps
- pipe
- grep

/proc/327
level 1

Step 3: demotion.

- ps
- pipe
- grep

Step 4: pipe write denied.

- ps
- pipe
- grep
Avoid Self-Revocation

- What could you do to avoid self-revocation?
Avoid Self-Revocation

• What could you do to avoid self-revocation in some cases?
  ‣ Add “floors” for subjects
    • Cannot open any objects below floor label (like Biba)
  ‣ This was done in the IX system
    • An MLS UNIX system by McElroy and Reeds
Information Flow

- Is a good goal because...

- Is a poor goal because...

- Can we use it to verify a policy is correct?
Information Flow

• Is a good goal because…
  ‣ No false negatives – an attack requires an illegal information flow
  ‣ Can define data and functional security requirements

• Is a poor goal because…
  ‣ Function may conflict with security
  ‣ How do we know when a permission is illegal, but is necessary for functional requirements?

• Can we use it to verify a policy is correct?
  ‣ Yes. *It defines a policy based on security. But what about exceptions?*
Clark-Wilson Integrity Model

- Goal: define integrity in terms of commercial terms rather than military (information flow)
- Insights?
Clark-Wilson Integrity Model

- **Goal**: define integrity in terms of commercial terms rather than military (MLS/Biba)

- **Insights? Based on Double-Blind Accounting**
  - Start with high integrity data
    - Validate data integrity *(integrity verification procedures)*
  - Only apply high integrity processes to change that data
    - Distinguish high integrity code *(transformation procedures)*
  - Ensure high integrity processes protect themselves
    - When they receive low integrity inputs *(convert or reject)*
  - Recheck that data still satisfies integrity requirements *(IVP)*
Clark-Wilson Integrity Model

- Model consists of a set of certification and enforcement rules governing integrity

- Own terms
  - CDI – Constrained Data Items (High integrity data)
  - UDI – Unconstrained Data Items (Low integrity data)
  - IVP – Integrity Verification Procedures (certify CDIs)
  - TP – Transformation Procedures (High integrity programs)
Clark-Wilson Integrity Model

- Model consists of a set of certification and enforcement rules governing integrity
  - C1—When an IVP is executed, it must ensure the CDIs are valid.
  - C2—For some associated set of CDIs, a TP must transform those CDIs from one valid state to another.
  - C3—Allowed relations must meet the requirements of “separation of duty.”
  - C4—All TPs must append to a log enough information to reconstruct the operation.
  - C5—Any TP that takes a UDI as input may only perform valid transactions for all possible values of the UDI. The TP will either accept (convert to CDI) or reject the UDI.
Clark-Wilson Integrity Model

- Model consists of a set of certification and enforcement rules governing integrity
  - E1—System must maintain a list of certified relations and ensure only TPs certified to run on a CDI change that CDI.
  - E2—System must associate a user with each TP and set of CDIs.
  - E3—System must authenticate every user attempting a TP.
  - E4—Only the certifier of a TP may change the list of entities associated with that TP.
Clark-Wilson Integrity Model

- How does it work?
- Certify TPs and IVPs
  - IVPs certify CDIs and TPs modify them
  - TPs must also be able to handle an UDIs they receive securely
- Run the system
  - Authenticated users can modify a CDI if and only if:
    - They can access TP and CDI and
    - TP is authorized to change CDI
Clark-Wilson Results

- Are the information flows authorized different than information flow?

- Not really
Clark-Wilson Results

• Are the information flows authorized different than information flow?

• Not really, but CW is closer to current practice
  ‣ Test and analyze code (for integrity), certify code (e.g., signature), check code and data integrity before use (e.g., hash), and deal with untrusted inputs (e.g., filter)
Clark-Wilson Results

- If systems practice is analogous to Clark-Wilson integrity where are we going wrong?
Clark-Wilson Results

• If systems practice is analogous to Clark-Wilson integrity where are we going wrong?
  ‣ Not writing IVPs
  ‣ Not certifying TPs or CDIs
  ‣ Not systematically ensuring programs discard/upgrade UDIs
    • Or even know where programs expect to receive UDIs

• CW-Lite Philosophy: Assume we can distinguish CDIs/UDIs and allow programs to
Clark-Wilson Results

• If systems practice is analogous to Clark-Wilson integrity where are we going wrong?
  ‣ Not writing IVPs
  ‣ Not certifying TPs or CDIs
  ‣ Not systematically ensuring programs discard/upgrade UDIs

• But shouldn’t programs at least know where they expect to receive UDIs?
Clark-Wilson Lite

- **Philosophy**: Ensure programs only receive UDIs at entry points where programmers are prepared to handle untrusted inputs
  - Partition data into CDIs/UDIs from the program’s point of view
  - Allow program to declare entry points that will upgrade/discard UDIs
  - Only allow program to perform system calls that access UDIs at program-specified entry points for discard/upgrade
Take Away

• In a secure system, we must protect data integrity
  ‣ Even a prerequisite to secrecy protection

• Types of integrity – biased toward security or function
  ‣ Functional: least privilege; Security: information flow

• Integrity models
  ‣ Least privilege, Biba, LOMAC, Clark-Wilson

• Need to develop approaches to design mandatory protection system for integrity – for function and security