

Toward Automated Information-Flow Integrity Verification for Security-Critical Applications

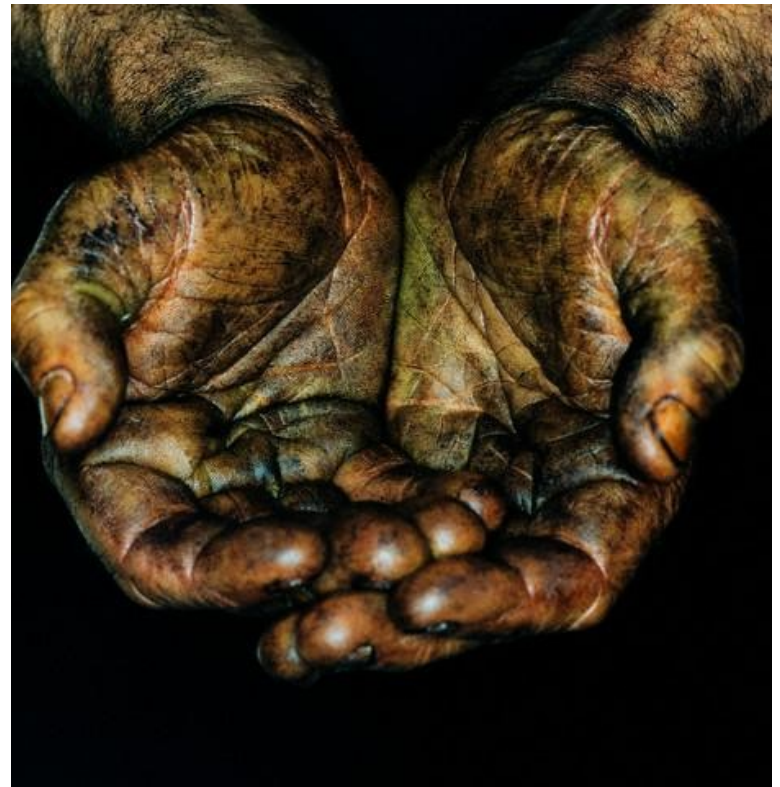
Umesh Shankar, Trent Jaeger and Reiner Sailer

*Presented by
Sandra Rueda*

Integrity

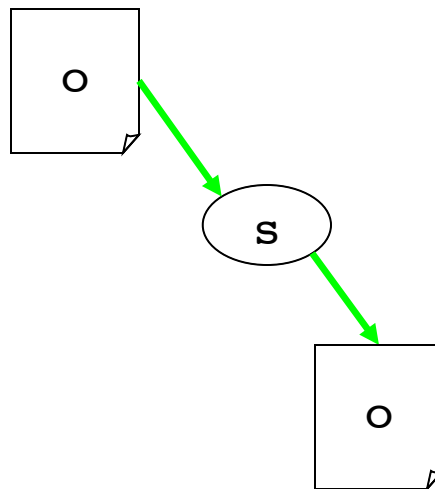
- Adherence to a strict code
- The quality of being honest
- The condition of being free from flaws

Why do we care?



Biba

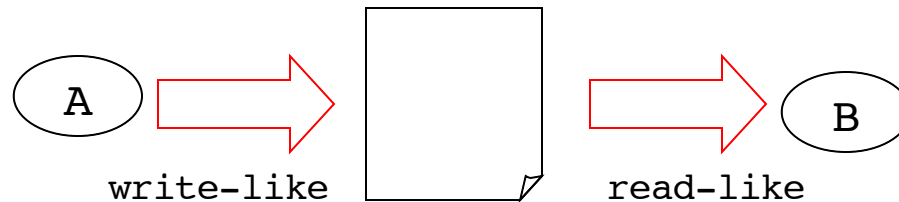
- i is a function that returns the integrity level of a subject or an object
- s can read o iff $i(s) \leq i(o)$
- s can write o iff $i(s) \geq i(o)$
- s_1 can execute s_2 iff $i(s_2) \leq i(s_1)$



- Is this a problem for some processes ?

Information Flow Integrity

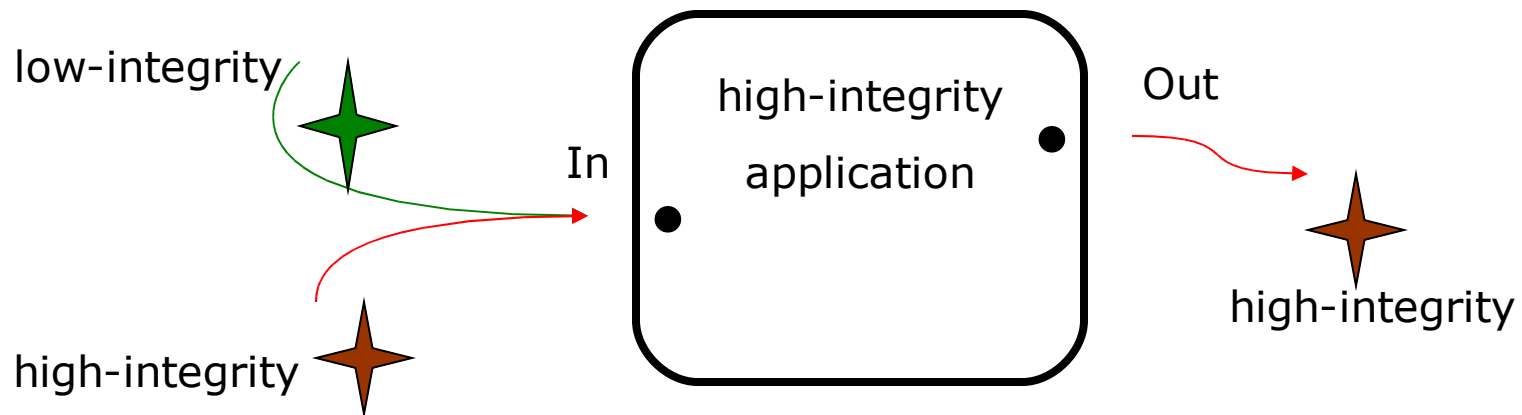
- There is an information flow between A and B if A can write to a resource on which B depends.
- There are transitive information flows



- The information flow integrity verification problem is to prove that a high integrity process does not depend on information flows from low integrity processes.

Clark-Wilson

- Clark-Wilson Integrity Model defines a set of certification and enforcement rules that guarantee integrity for constraint data items.
- Clark-Wilson requires:
 - Verification of application (formal approach which is not possible with our current technology)
 - Verification of information integrity properties of the inputs (filtering interfaces)
- Information Flow Integrity for inputs and outputs requires:
 - Proper configuration
 - Filtering code



- CW-Lite vs. Clark Wilson
 - Omits the formal semantics verification
 - Retains the same inter-process dependency semantics
- Motivations behind CW-Lite
 - Clark Wilson involves two goals that may be checked in a separate way
 - High level integrity programs read low level integrity inputs from a small number of locations



- Clark Wilson:
 - Application developer does not know the flows allowed in the system in advance thus all inputs must be filtered
- CW-Lite:
 - Use system knowledge to detect the inputs that must be filtered
- CW-Lite is preserved for a subject s if:
 - All high integrity subjects meet integrity requirements initially
 - All trusted code is identifiable as high integrity
 - All information flows are from subjects of equal or higher integrity unless they are filtered

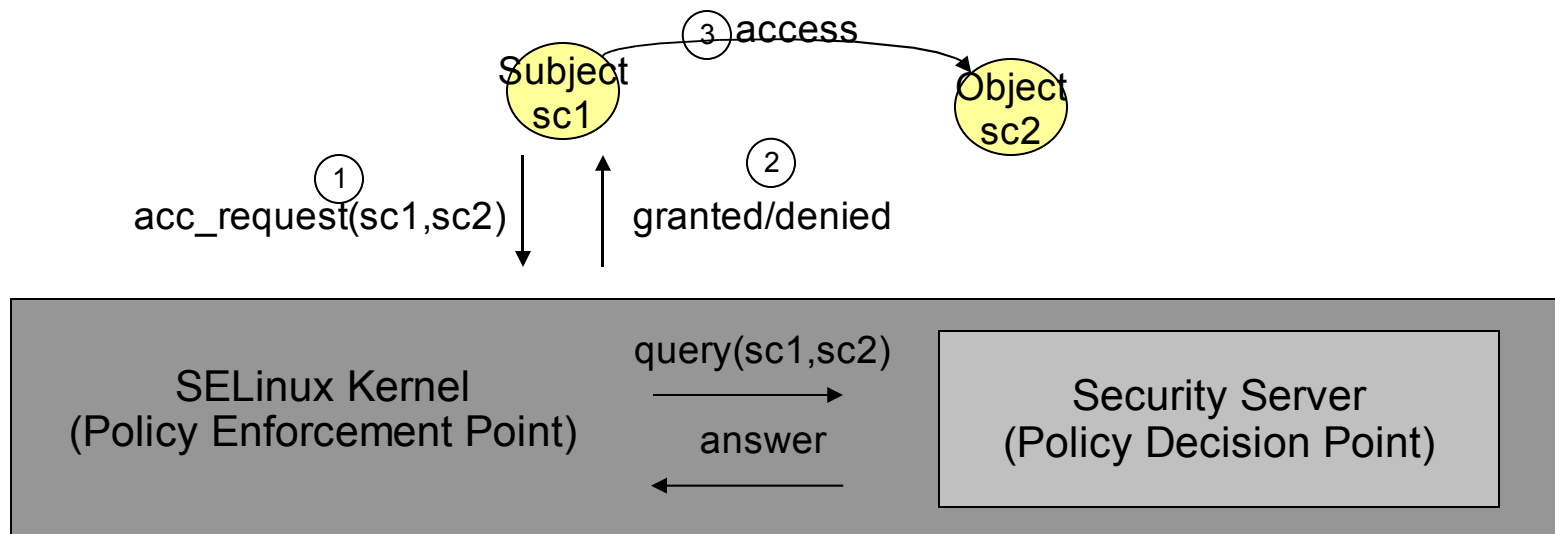


- Developers:
 - Identify untrusted inputs and implement filtering interfaces for them
 - Annotate those interfaces (DO_FILTER)
 - Develop filtering interfaces
 - Construct a policy to identify: ←
 - Trusted inputs
 - Untrusted inputs
- System Administrators:
 - Define the system TCB (or per-application)
 - Run the security policy analysis tool (Gokyo) for the target application ←
 - The tool identifies integrity violations, decide how to remove the illegal flow

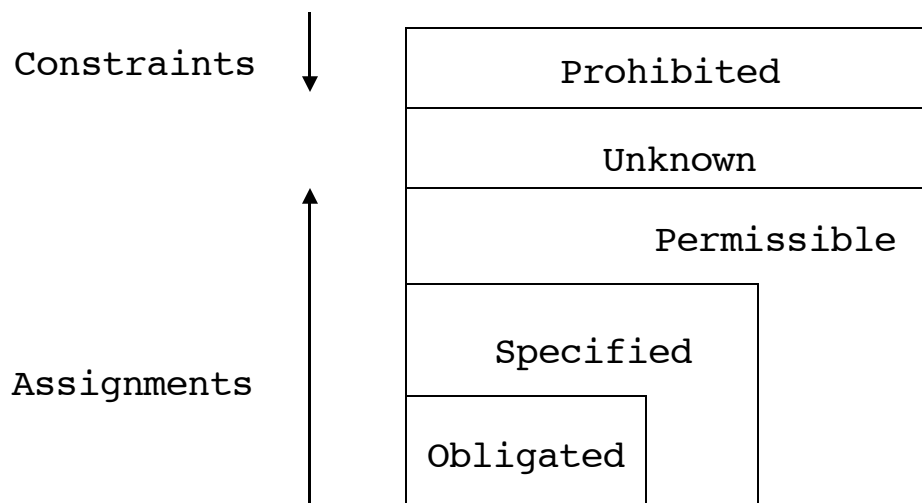
Compliance Services



- Linux Security Module developed by the NSA
- Implements Mandatory Access Control (MAC) on Linux
- Every subject and object in the system is assigned a collection of security attributes known as security context
- When a security decision is required, the security contexts of the subject and the objects are given to the security server, based on that and some predefined rules the security server makes the decision



- Access Control Space: set of permissions assigned to a subject



- Gokyo may be used to manage Access Control Policies
- Gokyo may also be used to check integrity constraints
- Integrity constraints enforce Biba-style integrity semantics

- Integrity Checking
- Define the set of constraints
- For each read permission of the higher integrity subject type, the corresponding write permissions are added to the constrained set of the lower integrity subject type
- For each write permission of the lower integrity subject type, the corresponding read permissions are added to the constrained set of the higher integrity subject type



Example

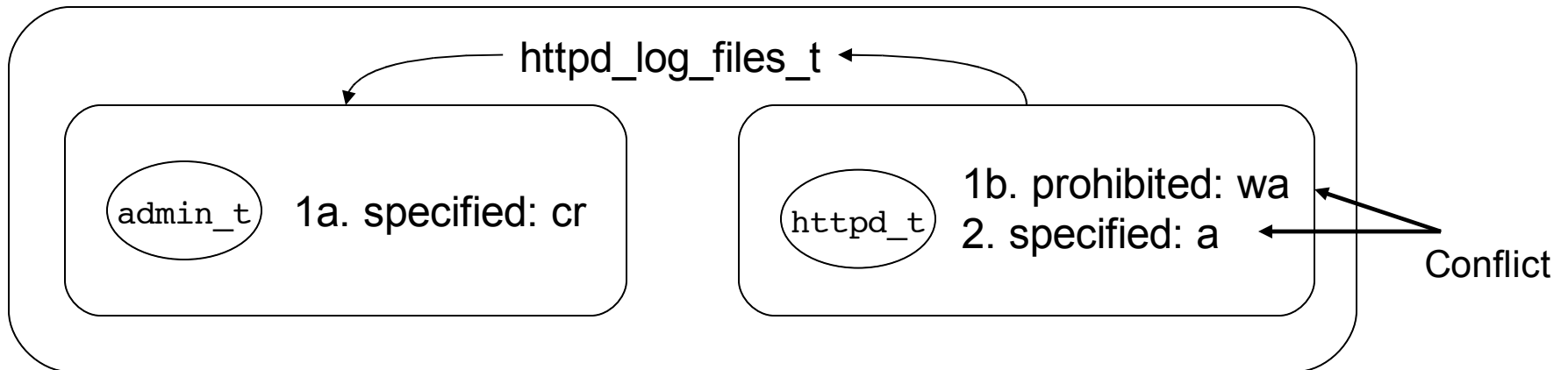
[Jaeger et al. SACMAT02]

Object Type	Description	Subject Type	Perm
httpd_log_files_t	Application Logs	admin_t	cr
		httpd_t	a
		user_script_t	a
		sys_script_t	a

Web Server
file system permission assignments

Node1	Node2	Type	Aspect
admin_t	http_d	integrity	perms
admin_t	user_script_t	integrity	perms
admin_t	sys_script_t	integrity	perms

Initial web server
policy constraints



Supporting Filtering Interfaces

- MAC Operating System : SELinux
- SELinux access control system was modified to enforce CW-Lite
- Two subject types per process instead of one
 - The default type allows inputs only from subjects in the TCB
 - The filtering subject type is used for interfaces annotated with DO_FILTER
- Gokyo checks that there is no untrusted input to the application's default type
 - It computes the set $F = \{ s' \mid \text{mod}(s', o) \quad \text{obs}(s, o) \}$
- `settraceonerror` is used to detect untrusted interfaces without the annotation



Example

Before

Source Code

```
conn = accept()  
get_http_request_sanitized(conn)
```

Policy

```
Apache: allow read httpd.conf  
Apache: allow accept
```

After

Source Code

```
DO_FILTER(conn = accept())  
get_http_request_sanitized(conn)
```

Policy

```
Apache: allow read httpd.conf  
Apache-filter: allow accept
```

- OpenSSH and vsftpd
- Goal: check that all information flows into privileged components are high integrity level or filtered via declared interfaces
 - The server-side daemon was decomposed into privileged and unprivileged components
 - Add default and filtered types to the OS policy
 - Define the TCB
 - Add annotations to the privileged components
 - inputs from network
 - inputs from unprivileged components
 - Verify interfaces with `settraceonerror`
 - The developer must build the filtering interfaces
 - Run Gokyo
 - 21 conflicts for OpenSSH
 - 8 conflicts for vsftpd
 - Conflict resolution requires system/application knowledge

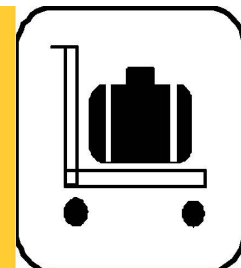


Future Work

- Apply CW-Lite to the TCB of the system
- Is it possible to prove that filtering interfaces are semantically correct?



Take Away



- Anything about Integrity Verification is important
 - Big problem in computer science
- Verification of integrity for a given system is not an easy task
 - CW-Lite applies divide and conquer (check only inputs)
 - It also defines area of improvement (check only untrusted inputs)
- CW-Lite identifies sub-problems and propose a way to solve one of them
 - it is SELinux-based. May be extended to MAC-based
- From comments in class:
 - Who defines what is integrity?
 - This property is evaluated based on parameters (looks like there is no universal definition)