Systems and Internet Infrastructure Security

Network and Security Research Center
Department of Computer Science and Engineering
Pennsylvania State University, University Park PA

# Runtime Analysis

November 28, 2011

# Analysis So Far

- Prove whether a property always holds
  - *May analysis*
- Prove whether a property can hold
  - *Must analysis*
- Key step: abstract interpretation to overapproximate behavior of program
- But, it can be expensive and complex

# Runtime Analysis

- Collect traces of program runs to evaluate a property
- Testing
  - Run test cases to determine if property holds (or fails to hold) in all cases
  - Inherently incomplete
- Traces
  - Compare several runs to determine if a property holds across runs
  - Incomplete?

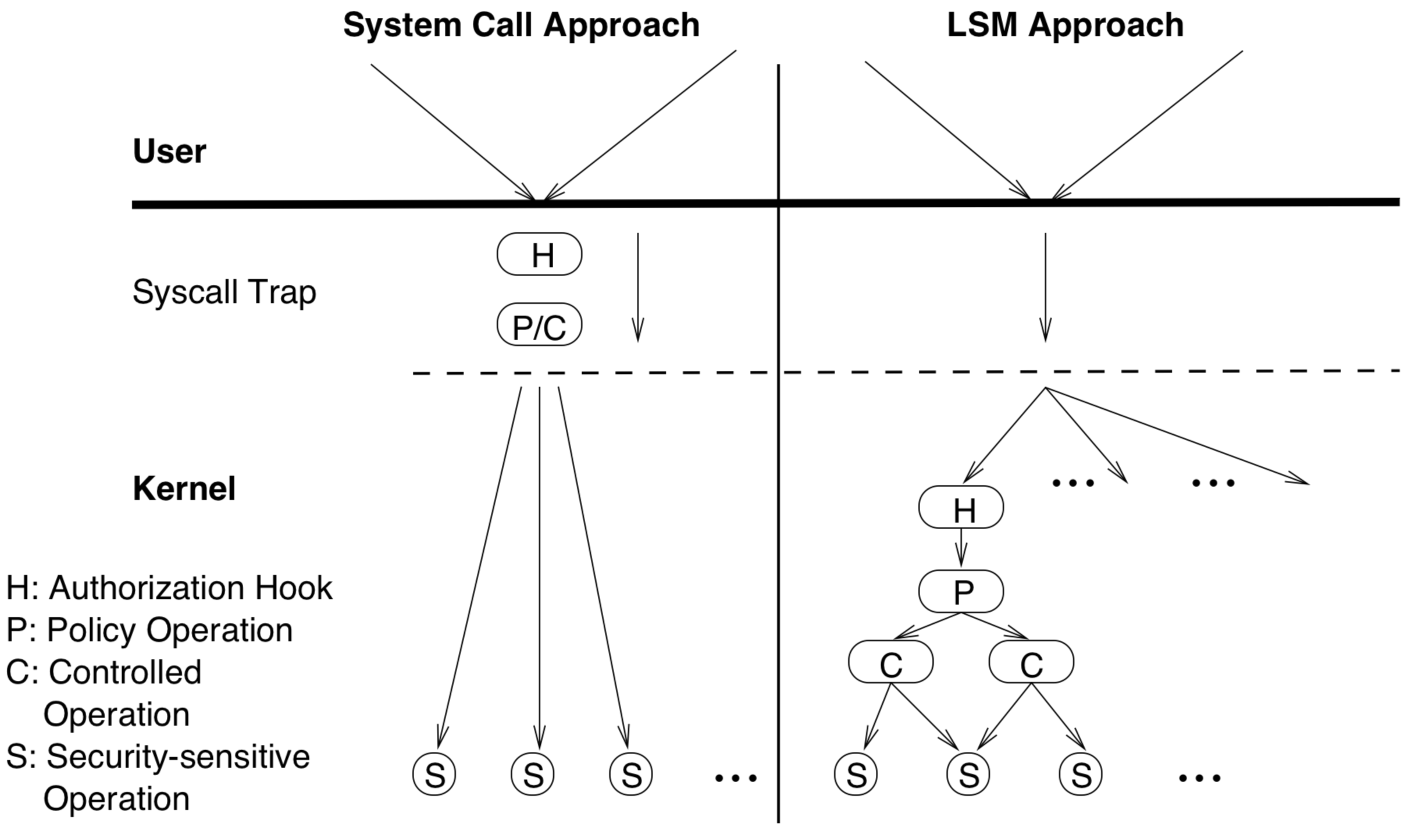
# Example

- Runtime Verification of Authorization Hook Placement for the Linux Security Modules Framework
- Linux Security Modules (LSM) framework
- Problem: Are authorization hooks placed correctly?
  - What does that mean?

# Mediation

- *Security-sensitive Operations*: These are the operations that impact the security of the system.
- *Controlled Operations*: A subset of security-sensitive operations that mediate access to all other security-sensitive operations. These operations define a *mediation interface.*
- *Authorization Hooks*: These are the authorization checks in the system (e.g., the LSM-patched Linux kernel).
- *Policy Operations*: These are the conceptual operations authorized by the authorization hooks.

# Mediation Overview



H: Authorization Hook
P: Policy Operation
C: Controlled Operation
S: Security-sensitive Operation

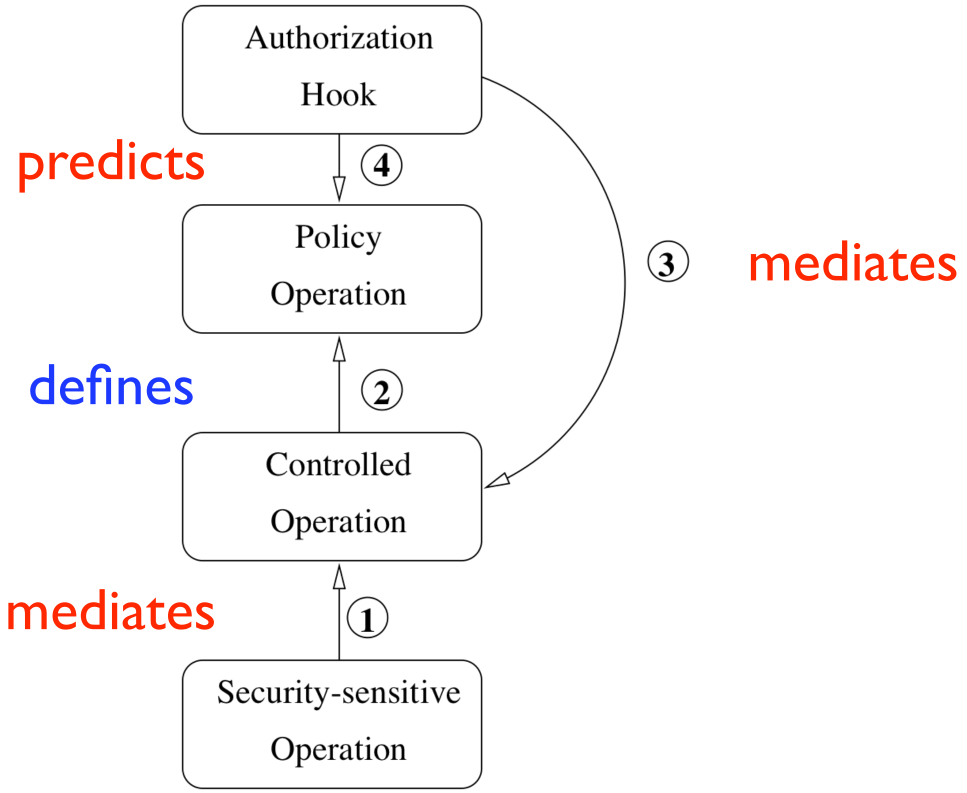
# Security-Sensitive Ops

- What code-level operations indicate security-sensitivity?
- Variable access?
- Structure member access?
- Global access?

# Key Challenges

- *Identify Controlled Operations*: Find the set of security-sensitive operations that define a mediation interface
- *Determine Authorization Requirements*: For each controlled operation, identify the policy operation
- *Verify Complete Authorization*: For each controlled operation, verify that the correct authorization requirements (policy operation) is enforced
- *Verify Hook Placement Clarity*: Controlled operations implementing a policy operation should be easily identifiable from their authorization hooks

# Key Relations



Authorization Hook

predicts ④

Policy Operation

③ mediates

defines ②

Controlled Operation

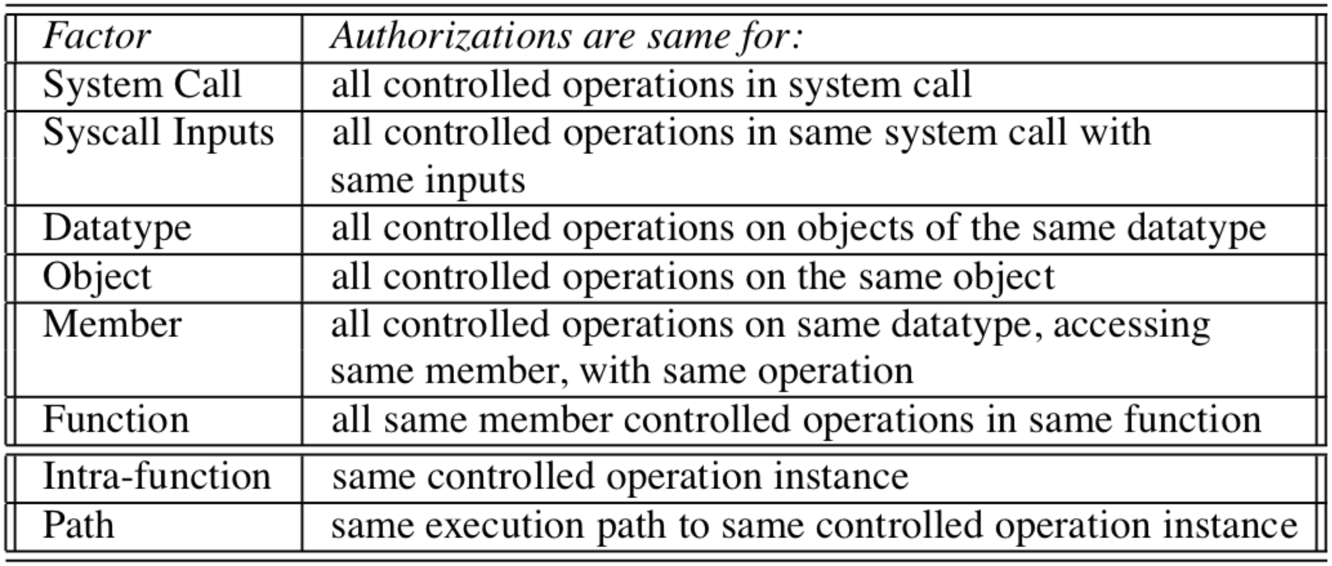
mediates ①

Security-sensitive Operation

# Analysis Approach

- Check consistency between hooks and security-sensitive operations
  - Traces
- Sensitivity
  - Structure member accesses
  - Hooks
- Consistent relationship indicates hook is associated with SMAs (make a controlled op)
  - Sensitivity can vary in granularity

# Sensitivities



| *Factor* | *Authorizations are same for:* |
|---|---|
| System Call | all controlled operations in system call |
| Syscall Inputs | all controlled operations in same system call with same inputs |
| Datatype | all controlled operations on objects of the same datatype |
| Object | all controlled operations on the same object |
| Member | all controlled operations on same datatype, accessing same member, with same operation |
| Function | all same member controlled operations in same function |
| Intra-function | same controlled operation instance |
| Path | same execution path to same controlled operation instance |

**Table 1: Authorization Sensitivity Factors: names and effects on authorizations**

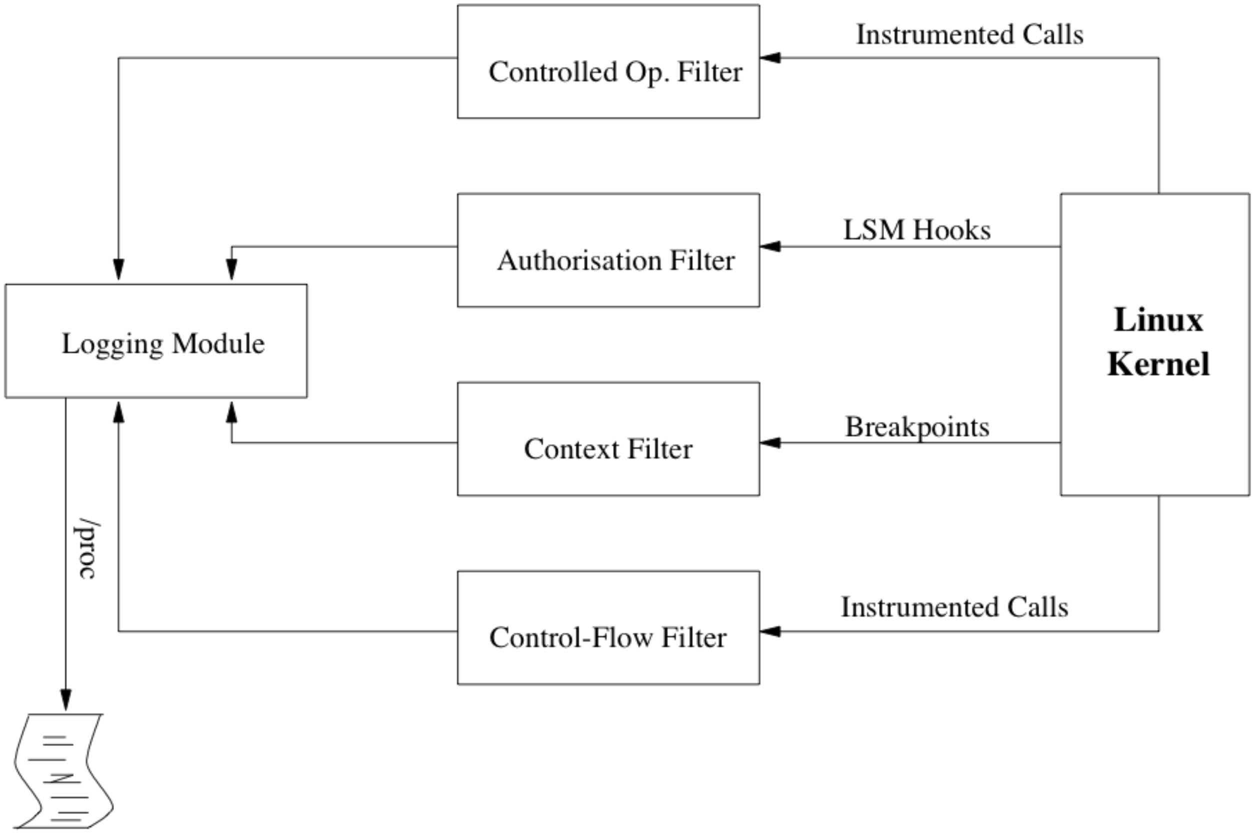
# Anomalies

- For SMAs to be a controlled op
  - Path: all traces with SMA should have same hooks
    - Not dependent on paths taken to get there
  - Function: all traces with same SMA type in same function should have same hooks
    - SMA in function defines controlled op if always associated with hook

# Implementation

- Propose sensitivity rules for system call processing
  - Propose relationship between hooks and controlled ops
- Log traces of system call processing
  - Collect syscall entry/exit/args, function entry/exit, controlled ops, and hooks
- Compute whether hooks always/sometimes/never in trace for each controlled op
  - Evaluate whether the current sensitivity rules express the expected consistency
- Update sensitivity rules

# Implementation

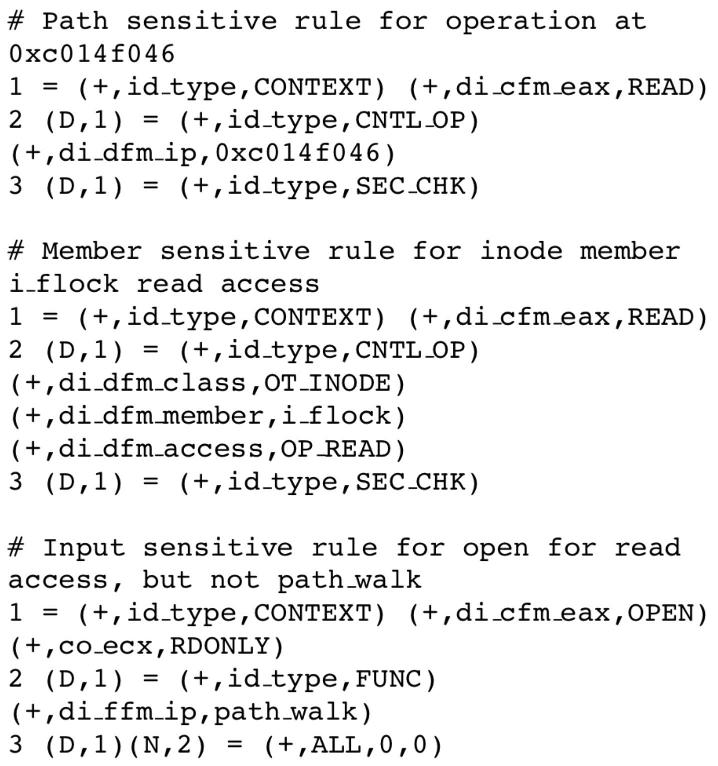


# Logging

- Authorization hooks
  - LSM itself
- Controlled operations (SSOs)
  - GCC module
- Control data
  - GCC flag
- System call contexts
  - Kernel scheduling loop

# Log Filtering Rules

- For sensitivity
  - Filter log entries processed to determine sensitivity



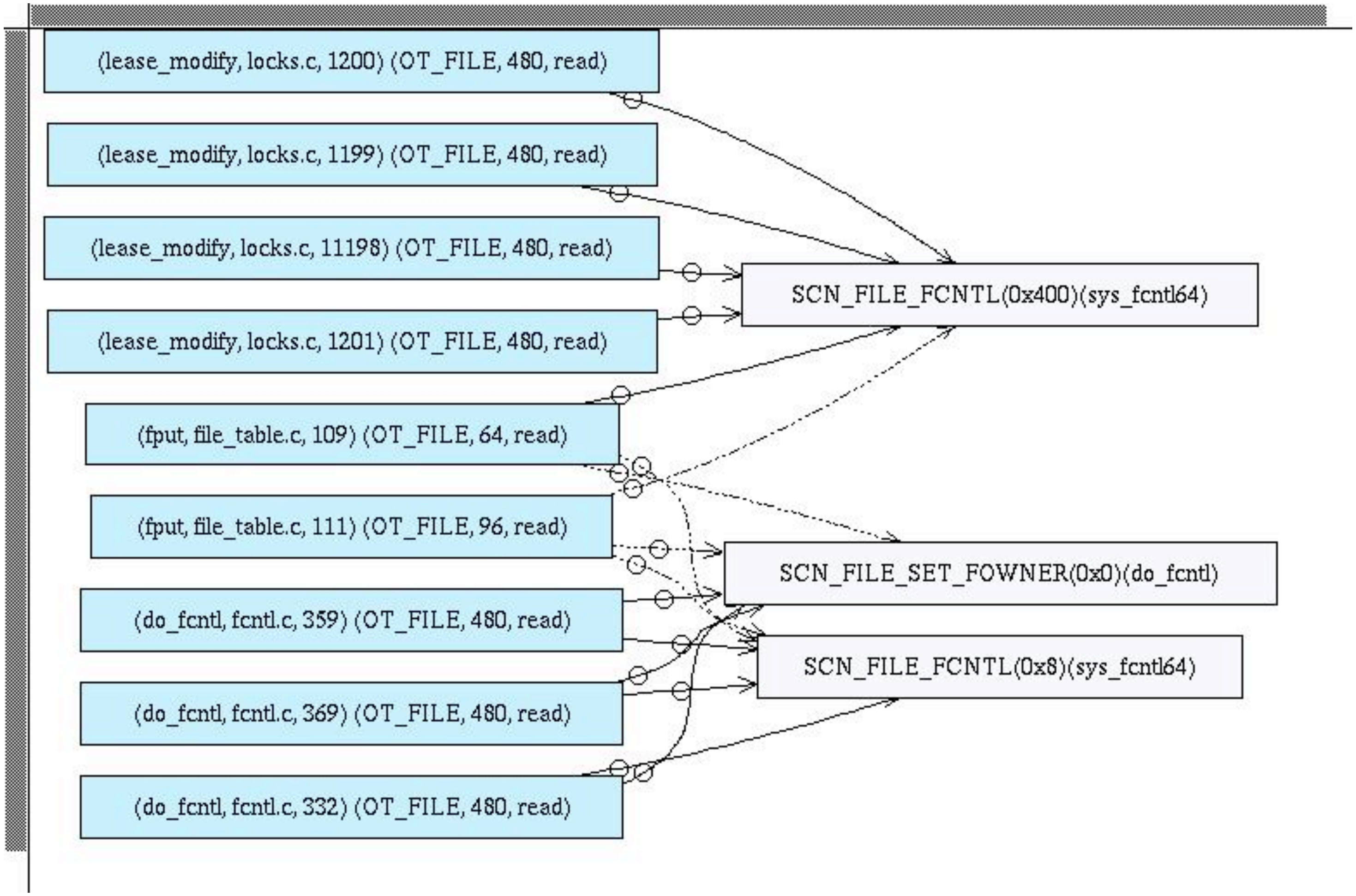
```
# Path sensitive rule for operation at
0xc014f046
1 = (+,id_type,CONTEXT) (+,di_cfm_eax,READ)
2 (D,1) = (+,id_type,CNTL_OP)
(+,di_dfm_ip,0xc014f046)
3 (D,1) = (+,id_type,SEC_CHK)

# Member sensitive rule for inode member
i_flock read access
1 = (+,id_type,CONTEXT) (+,di_cfm_eax,READ)
2 (D,1) = (+,id_type,CNTL_OP)
(+,di_dfm_class,OT_INODE)
(+,di_dfm_member,i_flock)
(+,di_dfm_access,OP_READ)
3 (D,1) = (+,id_type,SEC_CHK)

# Input sensitive rule for open for read
access, but not path_walk
1 = (+,id_type,CONTEXT) (+,di_cfm_eax,OPEN)
(+,co_ecx,RDONLY)
2 (D,1) = (+,id_type,FUNC)
(+,di_ffm_ip,path_walk)
3 (D,1)(N,2) = (+,ALL,0,0)
```

**Figure 4: Example authorization sensitivity filtering rules**

# Log Filtering Rules



(lease_modify, locks.c, 1200) (OT_FILE, 480, read)

(lease_modify, locks.c, 1199) (OT_FILE, 480, read)

(lease_modify, locks.c, 11198) (OT_FILE, 480, read)

(lease_modify, locks.c, 1201) (OT_FILE, 480, read)

(fput, file_table.c, 109) (OT_FILE, 64, read)

(fput, file_table.c, 111) (OT_FILE, 96, read)

(do_fcntl, fcntl.c, 359) (OT_FILE, 480, read)

(do_fcntl, fcntl.c, 369) (OT_FILE, 480, read)

(do_fcntl, fcntl.c, 332) (OT_FILE, 480, read)

SCN_FILE_FCNTL(0x400)(sys_fcntl64)

SCN_FILE_SET_FOWNER(0x0)(do_fcntl)

SCN_FILE_FCNTL(0x8)(sys_fcntl64)

# Results

- Missing hook
  - Setgroups16
- Have different numbers of hooks
  - Fcntl (set_fowner)
- Missing hook
  - Fcntl (signal)
- Missing hook
  - Read (Memory mapped files)

# Runtime Analysis

- Choose test cases
- Collect traces (content of traces)
- Analyze traces
- Evaluate property

# Hook Placement

- A variety of analysis for hook placement and testing
- Zhang [USENIX 2002]
- Ganapathy [CCS 3005, Oakland 2006, ICSE 2007]
- Tan [USENIX 2008]
- [AsiaCCS 2008]
- Son [OOPSLA 2010]
- King etal [ESOP 2010]
- We are working on a purely static analysis