IBM Research

Storage-Based Intrusion Detection
for Storage Area Networks (SANs)

Mohammad Banikazemi **Dan Poff** **Bulent Abali**

Thomas J. Watson Research Center
IBM Research

Outline

- Background
- Motivation
- Storage-Based Intrusion Detection Systems
- Prototype Implementations
- Conclusions

Background

- Intrusion Detection Systems are mainly
  - Network-based
    - Scanning network traffic for suspicious traffic, known signatures, etc.
    - Firewalls, Sniffers
  - Host-based
    - Integrated with the host OS
    - ID utilities that run on host systems (such as Tripwire)
    - Looking for signs of intrusion and suspicious behavior

- Storage systems are another place where ID techniques could be deployed

Motivation

- Most intrusions are visible at storage system level
  - Changing attributes of files
  - Changing/replacing system utilities
  - Deleting/modifying log and other important files
- Compromise is persistent across reboots ➔ visible to the storage system
- Storage based ID may be effective even if host OS is compromised
  - When a machine is compromised, host-based IDSs can become ineffective
- More difficult to compromise storage systems
  - Narrow interface (e.g. SCSI); Less known architectures in comparison with servers
- Another level of detection/protection; Not a replacement for other ID techniques

Storage-Based Intrusion Detection Systems (IDSs)

Storage-based ID can be deployed in

- Files servers [CMU]
- Object storage devices
- **Block storage devices (this paper's contribution)**

1. Real Time Storage-Based (RTSB) IDS

- Embedded in IBM SAN Volume Controller (SVC), a storage virtualization engine

2. File Level Storage Based (FLSB) IDS

- Loosely coupled with IBM DS4000 (FAStT)

- Both prototypes are rule-based (policy-based) ID systems

Basic components of such systems:

- Access rules
- Rule violation detection (Intrusion Detection)
- Response to violation

Rule-Based IDSs: Access Rules

- Some Common Rules
  - Data/attribute modification
    - Block modification: e.g. boot sector
    - Some files are modified only rarely
    - For some files any update can be sign of an intrusion
  - Update pattern
    - Log files
  - Content integrity
    - Known files such as password files
  - Suspicious content
    - Known virus signatures
    - Can be scanned and detected even when the host is compromised/ineffective
- Most interesting and effective rules require information about file systems

Rule-Based IDSs: Rule Violation Detection

- Most interesting and effective rules require information about file systems
- Detection at block storage systems more difficult
- Two choices:
  - Use file system implementations to monitor/evaluate files at file system level
    - Difficult to do at block storage devices in a real time manner
  - Convert file system level rules to storage block level rules
    - Monitor block accesses in real time

Rule-Based IDSs: Response to Violation

- Possible responses:

1. Generating alerts
2. Preventing requests from completing
3. Slowing storage requests
4. Initiating versioning
5. Using space/time efficient point-in-time copy

Prototype Implementations

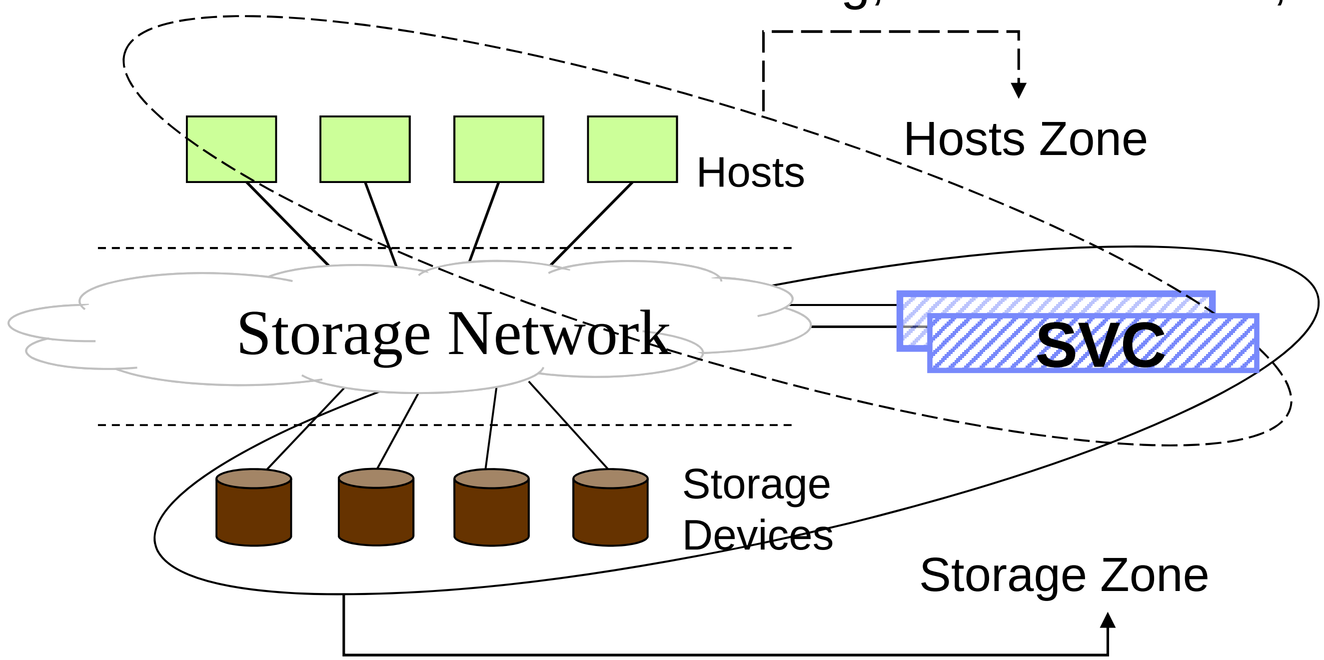
Two IDSs for SAN environments:

1. Real Time Storage-Based (RTSB) IDS
   - Monitors access to storage devices at storage block level
   - Embedded in IBM SAN Volume Controller (SVC), a storage virtualization engine
   - Can prevent intrusion (and damage to storage)

2. File Level Storage Based (FLSB) IDS
   - Monitors block storage devices at file system level
   - Takes advantage of space/time efficient point-in-time copy
   - Loosely coupled with IBM DS4000 (FAStT)
   - Provides quick recovery

RTSB: Background

- SVC: block storage controller & virtualization system (Fibre Channel)
- Cluster of Pentium-based servers; redundancy, modularity and scalability
- Uses an in-band approach
- Almost entirely in user mode -> easy to debug -> Perfect platform for development/evaluation
- Advanced functions: data block caching, fast-write cache, copy services

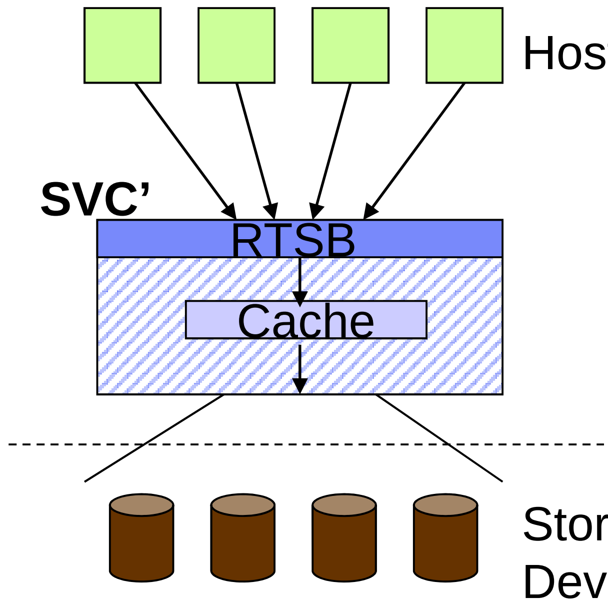


RTSB: Design

- SVC converts File level access rules are to block level rules
  - SVC knows ext2 filesystem format
  - traverses the storage data blocks and interprets the super block and i-node information to find blocks associated with a given file

- All accesses are monitored; If any access rule associated with a given block, the access is further examined

- In the current implementation, for each LUN a bitmap is used to keep track of blocks with access rules.



RTSB: Design (2)

- Cases where a block is shared by more than one file, and/or only accesses to certain portions of the block causes a violation are more complicated
  - old content is compared with the incoming block write request to determine which part of the block is being modified

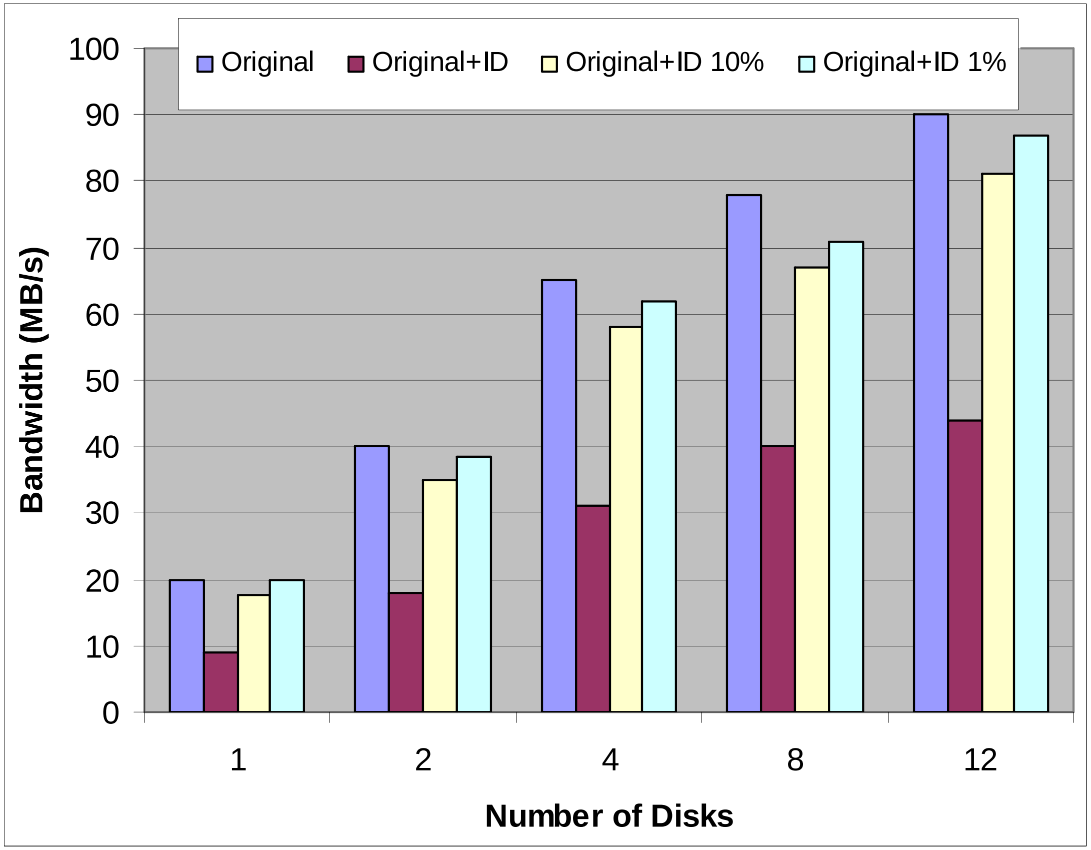
- Access rules are checked in a layer above the SVC cache layer
  - The current content may be residing in the storage cache. In such cases, overhead of accessing old content is very low
  - Storage blocks corresponding to files which are being monitored can be given higher priority for caching purposes (not implemented)

- Any rule violations triggers an email to the system administrator
  - The violation is not only identified by the storage block but also the file(s) and file-based rules which have resulted in the block level rule.

RTSB: Performance Impact

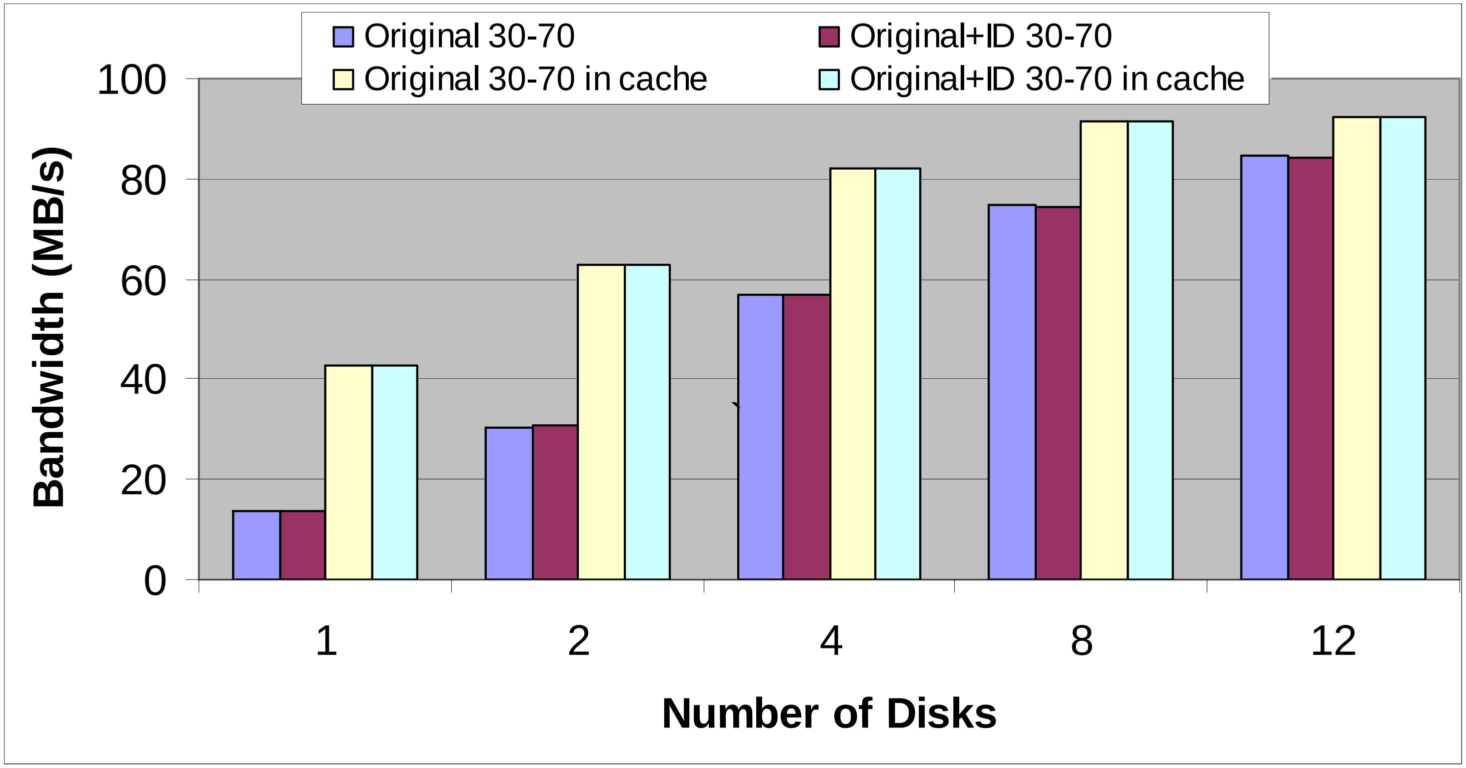
- 16k and 128K request sizes
- All writes; Random and sequential accesses
- Content of varying percentage of blocks are inspected

- Impact small even for all write accesses with 1% inspection rate
- CMU study shows well less than 1% of blocks require inspection of their content
- 128K random writes; similar results with other request sizes



RTSB: Performance Impact

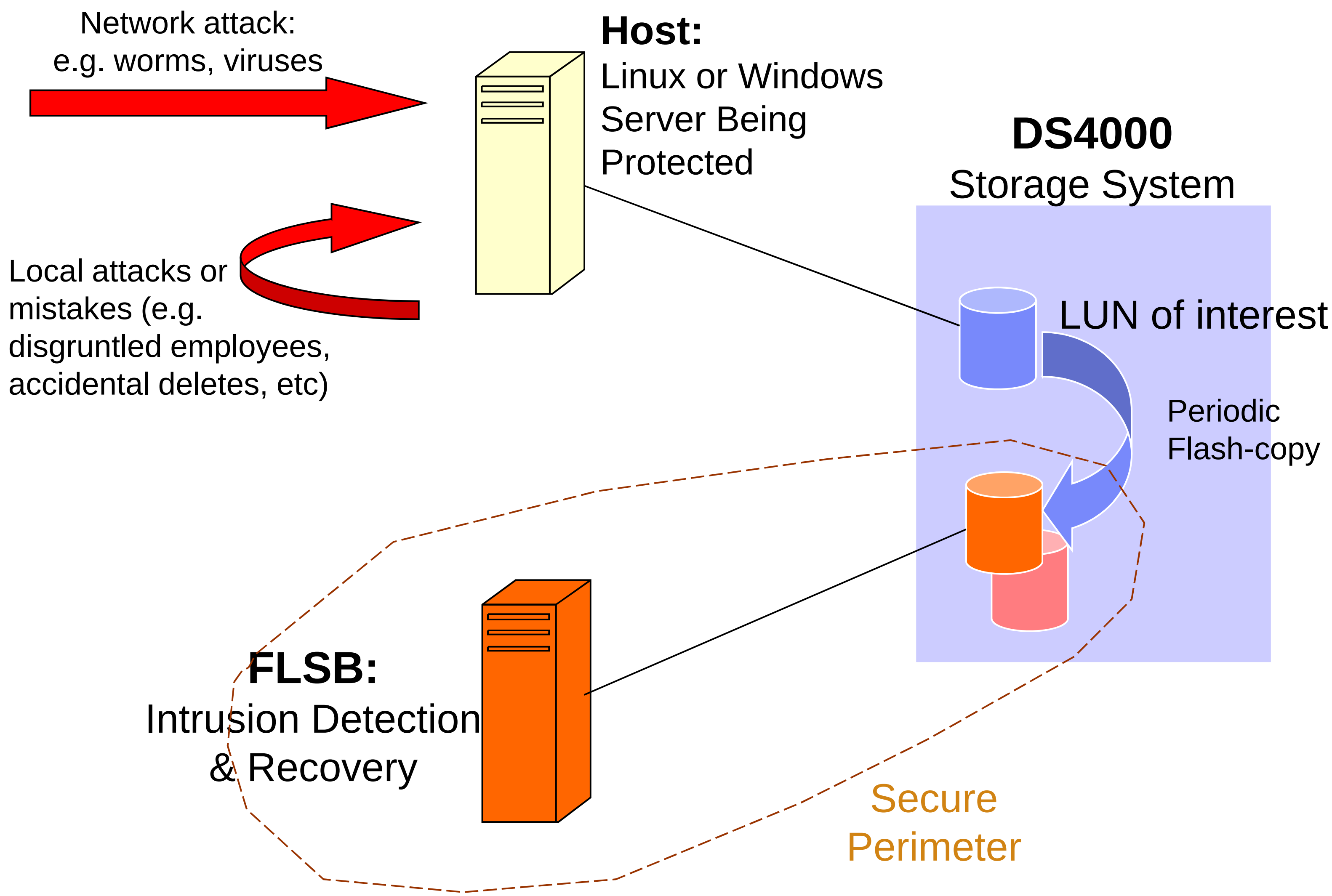
- Use 30% write 70% read accesses
- Inspecting 1% of writes
- Negligible impact on performance



FLSB: Background

- Different from RTSB!
- Performs ID at file system level (however not on the host system)
- No modifications to the storage system software; nor the host filesystem
- Works as an appliance loosely coupled with the storage system
- Takes advantage of the time and space efficient point-in-time copy operation of block storage system

FLSB: Design (Schematic)

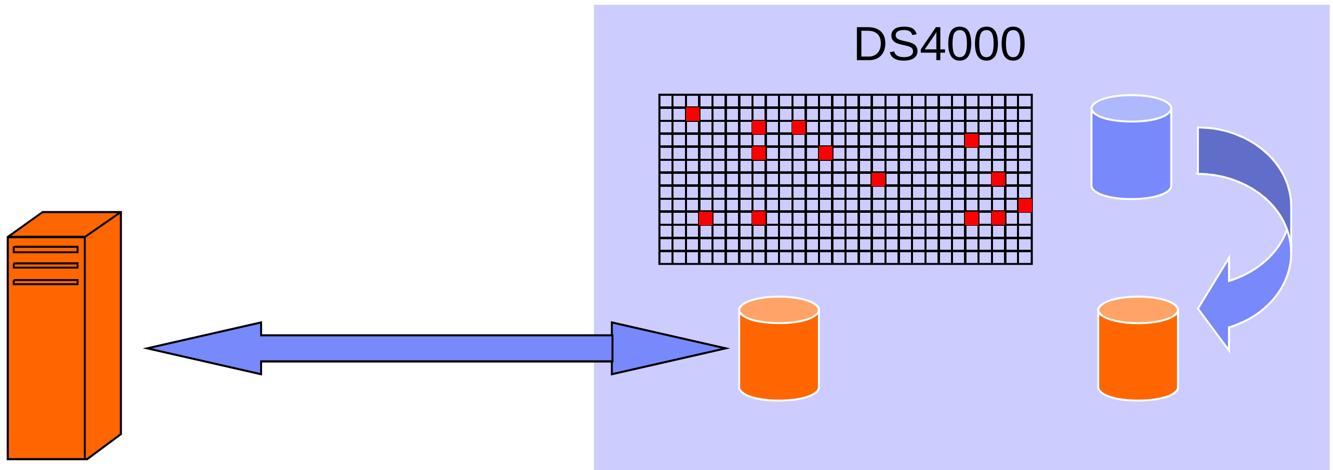


FLSB: Design

- Signatures of files of interest are created (using Tripwire)
  - These are stored in LUNs not accessible by client/host systems
- Copies of LUNs of interest are periodically made using Flash-copy operation (time/space efficient point-in-time copy)
- After each copy, the copy is mounted and inspected at file system level (using Tripwire) and newly generated signatures are compared with the original ones for signs of intrusion
- Throws out old copies
  - Keep at least one “good” copy such that compromised data can be recovered

FLSB: Optimization

- Flash-copy already keeps track of modified blocks (i.e. diff bitmap)
- Need not examine some blocks if they have not been touched since last copy
- Obtain the list of modified blocks from the storage system



Conclusions

- We discussed merits of storage-based ID
- We presented two prototype implementations for block storage devices in SAN environments
- We showed that the performance impact of ID is very low and negligible

- Demo: I have a recorded demo on my laptop; if you are interested please see me after the talk.