Adding Security to File Systems, One Layer at a Time

Erez Zadok

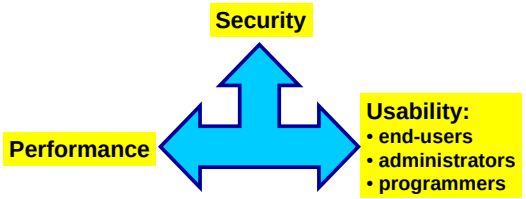
File-systems and Storage Laboratory
Stony Brook University
http://www.fsl.cs.sunysb.edu/

Why File Systems?

- Hardware can be replaced…
- Networks can be upgraded…
- Even employees come and go…

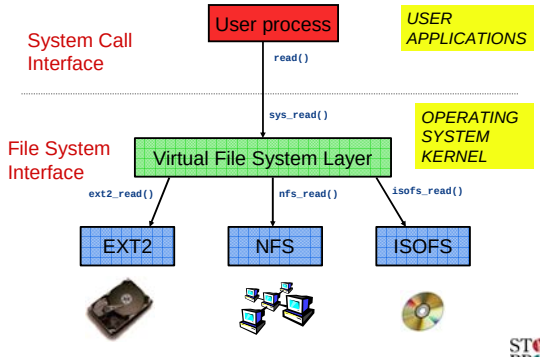
- … but a company's data (IP) must **never** be lost.
- The *file system* software is in charge of access to stored data.

Balancing Conflicting Needs

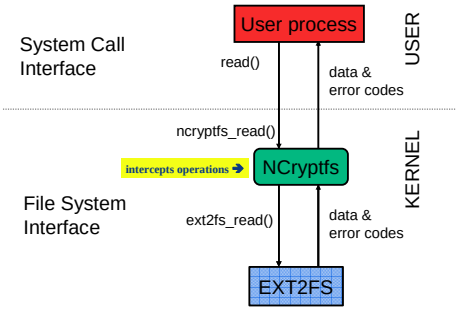


Our goal: Explore all three aspects for file systems

How File Systems Work



File System Interception



Interception for Security

- Interception a useful security concept
- Network interception
  - Easy to do without changing clients or servers
  - Doesn't impact application performance
  - Hard to infer high-level information from packets
  - Not all events produce network messages
- System call interception
  - Easy to do for most OSs
  - Does not capture all events (mmap)
  - Does not capture server-side events
  - May affect performance of every application

File-System Level Interception

- Logically: below system calls and well above network
  - Uses the Virtual File System (VFS) layer
- Pros:
  - No need to change clients or servers or applications
  - Minimal performance overhead, and only for those applications using an intercepted file system
  - Easy to infer high-level information (users, file names, etc.)
  - Capture all file accesses (including mmap)
  - Works equally well on clients, servers, or as a *proxy*
  - The file system normalizes data access to individual pages
- Cons:
  - Kernel development is difficult
  - Only captures file-related accesses

Outline

- **NCryptfs [Usenix 2003]**
- I3FS [LISA 2004]
- AVFS [Security 2004]
- Other and ongoing projects

NCryptfs: Versatile & Convenient

- Multiple ciphers, keys, and key lengths
- Encryption per user, process, group, or session
  - Ad-hoc groups
- Auth vs. Encryption keys
- Key timeouts and revocations:
  - Suspend/resume applications
  - Terminate/sandbox processes
  - Invoke user-land helper to re-authenticate

OS: Cache Cleaning

- Cached data is vulnerable
  - **`mmap`** accesses without consulting file system
  - Cached data can outlive key
- NCryptfs evicts cleartext pages
  - When keys become invalid
  - When authorizations become invalid
- The ciphertext page may still be cached, so I/O is not required to access the data again

OS: On-Exit Callbacks

- Expunge private user info on process exit, advantages over alternatives:
  - Efficiency: no periodic scans of lists
  - Security: no gap between process death and cleanup
- NCryptfs uses on-exit callbacks to
  - purge active sessions and authorizations
  - challenge-response authentication
    - the task-private data creates a session between a user process and the kernel

User Space Callbacks

- Kernel calls user-space program

- NCryptfs GUI to re-key on timeout
- Executes as the user, not as root

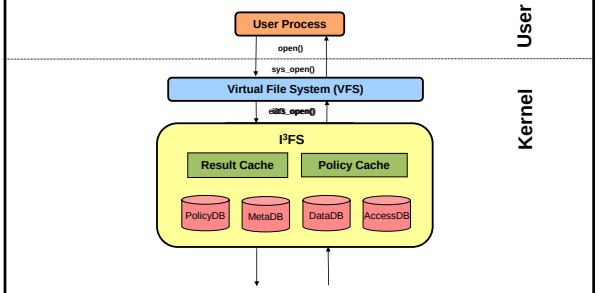
Outline

- NCryptfs [Usenix 2003]
- **I3FS [LISA 2004]**
- AVFS [Security 2004]
- Other and ongoing projects

I³FS: Integrity Checking

- Intercepts file system calls to protected files
  - A Stackable file system: inline interception
  - Performs checksum comparison
  - Reports checksum mismatch online
    - Optionally prevents access
- Files to be protected
  - Chosen by the administrator
  - Valid updates to protected files
    - Requires authentication
- File entities to be checksummed - Policy
  - Chosen by the administrator
  - E.g., inode fields, file data

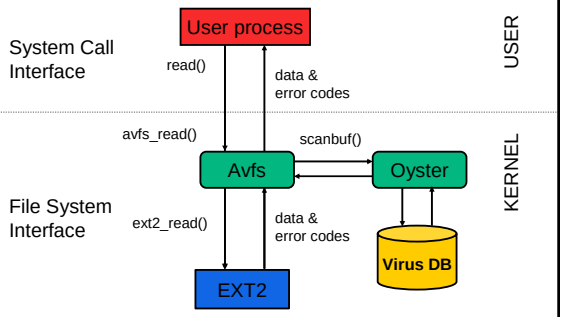
I³FS Architecture



Outline

- NCryptfs [Usenix 2003]
- I3FS [LISA 2004]
- **AVFS [Security 2004]**
- Other and ongoing projects

Avfs: Anti-Virus File System



Avfs Components

1. Oyster virus scanner
   - Based on ClamAV user-mode scanner
     - Modified Aho-Corasick scanner
   - We improved ClamAV's scalability
     - Linear memory consumption with size of virus DB
   - Linux kernel scanning API
2. Use unmodified ClamAV's virus DB
3. Avfs: a stackable file system for inline scanning
   - Forensic modes
   - Sandboxing process's file access
   - Versioning
   - Virus quarantining

Outline

- NCryptfs [Usenix 2003]
- I3FS [LISA 2004]
- AVFS [Security 2004]
- **Other and ongoing projects**

Tracefs

- **Flexibility**
  - Fine-grained control of what should be traced
- **Performance**
  - Low overheads. Trade-offs
- **Convenience**
  - Simple and self-contained trace file format
  - User-level utilities to configure and manipulate traces
- **Security**
  - Keyed checksums and encryption
  - Tracing to remote location
  - Anonymization

[FAST 2004]

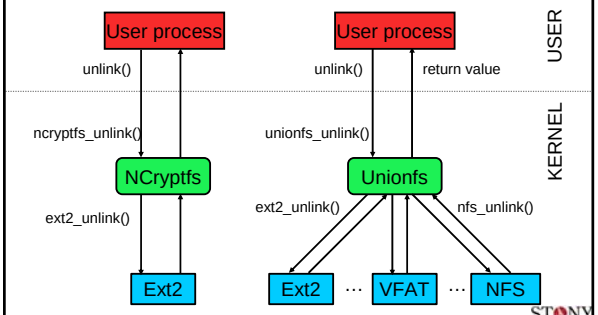
Replayfs

- Select subsets of traces to replay
  - e.g., IDS can pick parts to replay
- Compile portable traces into OS-specific format
  - even for another OS
- Replay traces at different speeds
- Single step undo/redo
  - Forensics

[FAST 2005]

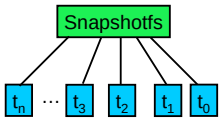
➔ *coming to a SNIA WG near you…*

Stacking and Fan-Out

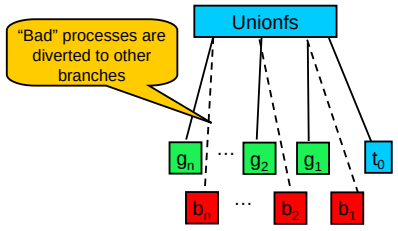


Snapshotfs

- Create a consistent image of an entire file system
- When a snapshot is taken at time *n*:
  - Mark existing branches file system read-only
  - Add a new high-priority branch (t_n)
  - All writes take place on t_n
  - Transparently migrate open files on $t_{i<n}$ to t_n
- To view snapshot *i*: union t_i through t_0
- Important fan-out features:
  - Dynamic branch management
  - File revalidation
  - Scalable to many branches



File System Sandboxing



Ongoing

- **RAIF**: Distributed storage survivability
  - code released in Jan 2007
  - [Cluster Sec 2005]
- **SDFS**: Secure Deletion of files
  - code released in Jan 2007
  - [SISW 2005, StorageSS 2006]
- **ACIDfs**: Transactions (TOCTOU)
  - code released in mid 2006
  - [ACM TOS 2007]

Take Home Messages

1. Security work must address *usability*
2. Using *layered* file systems to add security

Future

- **eCryptfs** (IBM)
  - already in Linux 2.6.19
  - More stacking support in 2007
- **Unionfs**:
  - basis for fan-out stackable file systems
  - code stability & cleanup
  - ETA 2007?

➔ *expect more*

Adding Security to File Systems, One Layer at a Time

Erez Zadok

Filesystems and Storage Laboratory
Stony Brook University
http://www.fsl.cs.sunysb.edu/