Dealing With Disaster: Surviving Misbehaved Kernel Extensions

Margo I. Seltzer, Yasuhiro Endo, Christopher Small, Keith A. Smith
Harvard University

Presented by: Tim Canfield

VINO Grafts (Kernel Extensions)

- Provide a richer & more powerful interface to kernel
  - Similar to SPIN extensions
  - VINO uses a different method
    - Transaction based operations
    - Software Fault Isolation (C++)
- Allow client specific code to be loaded into kernel
  - Customized policies
    - Page eviction
    - File cache
    - Signal delivery order

VINO Grafts are designed to prevent:

- Memory Misuse
  - Reading inappropriate data
  - Writing inappropriate data
  - Executing bad instructions
- Resource Hoarding
  - Memory
  - CPU
  - Locks
- Attempting to use Incorrect Interfaces
- Antisocial behavior
- Covert Denial of Service

9 Grafting Rules

- Grafts must be preemptible
  - All VINO threads are preemtible
- Cannot hold limited kernel resources for excessive periods
  - Resource accounting (to decide)
  - Transaction abort (to execute)
- No illegal accesses
  - MiSFIT
- Cannot call functions which alter or return protected data
  - Statically constructed *Graft-Callable* list
- Cannot call function to which they do not have access (7)
  - MiSFIT and Dynamic linker

More Rules

- Cannot replace restricted kernel functions
  - Downloading mechanism and dynamic loader
- Kernel cannot execute unsafe grafts
  - MiSFIT digitally signs safe grafts
- Only affect processes which have agreed to use them
- Kernel must always be able to make progress
  - Resource accounting
  - Downloading mechanism
  - Separation of global and local policy decisions

VINO Transactions

- Graft functionality is transaction based
  - Allows for grafts to fail
    - Clean aborts
    - System can maintain system consistency
  - Abort transactions that hoard resources
    - Time constrained resources
      - Uses timeouts
    - Quantity constrained resources
      - Uses resource *Limits*
      - Extends thread *Limits*

MORE ON TRANSACTIONS

- ACID
  - Atomicity
    - 2 phase locking
    - Accessor functions
      - Protect any permanent kernel state
      - Associated undo function
  - Durability
    - VINO transactions do **not** have to be durable
    - Transaction log is transient

Software Fault Isolation (SFI)

- MiSFIT
  - Compile time tool
  - Protects all loads and stores
- Direct function calls
  - Checked at link time
  - Only execute *Graft-Callable* functions
- Indirect function calls
  - Runtime Checking
  - Look functions up in a hash table

Cost of graft protection

- 3 types of grafts, 4 samples discussed
  - Stream grafts
    - Read Ahead
  - Prioritization grafts
    - Page eviction
    - Scheduling
  - Black box graft
    - Encryption/Decryption

Performance Measurement Methodology

- Individual overhead paths
  - Base Path
    - Straight kernel code
  - VINO Path
    - Indirection and return value verification
  - Null Path
    - Graft stubs, transaction begins and commits, and minimal graft
  - Unsafe Path
    - Full graft code and lock overhead
  - Safe Path
    - MiSFIT
  - Abort Path
    - Execute abort instead of commit

Overhead Components

- Transaction Operations (Consistent)
  - Begin (≈30 µs)
  - Commit (≈30µs)
  - Abort (Avg. 44 µs)
  - Locking (19/14 µs)
- MiSFIT instruction (Graft Dependent)
  - Incredibly high for Encryption/Decryption Graft
  - High anytime there are lots of loads and stores