Operating System Support for Secure, Distributed Object Systems

Jonathan S. Shapiro, Ph.D.

Systems Research Laboratory
Johns Hopkins University
Information Security Institute
Crypto != Security

♦ If I can crack the end system, here are some things that can’t possibly matter:
  – Cryptography
  – Authentication
  – Integrity

♦ If you want to build a secure, distributed object system, you first need to build a secure single node.
The Golden Crypto Bridge  
(Marketing Photo)

The view is sure pretty....
Crypto Bridges (Reality)

... but the bridge doesn’t go anywhere!
The Problem

- *Anybody* can hack today’s systems:

- Current operating systems *can’t* solve this. A new approach is needed.
- Firewalls don’t work
  - They don’t know what the app is doing
  - Communication necessarily bypasses the firewall
- There is no “defense in depth!”
The Solution… (?)

♦ “My thesis is that the software industry is weakly founded, and that one aspect of this weakness is the absence of a software components subindustry.”

– M. D. McIlroy

♦ “We must somehow or other form conceptual framework in which we can talk about these things in a clean and comprehensible way.”

– C. Strachey
Component-Based Systems

- **Naming**: what thing is to be invoked?
- **Protection**: components must be isolated from one another
- **Interface(s)**: what are the (sets of) operations exported by each component?
- **Encapsulation**: there must be no means to bypass the naming and interface primitives.
- **Persistence**: objects must survive shutdown

*Can a real system be built this way?*
EROS

- EROS is a secure, real-time operating system.
- It is based on capabilities, a different approach to protection than is used in current systems.
- Capabilities allow us to provide confinement – running each piece of software inside its own protective box.
- This lets us structure the system in an entirely different and more secure way.
  - Provide secure environment that applications can exploit for their own use
- EROS runs on desktops and servers, and soon on handhelds.
Capabilities

- A capability is a pointer to an object
- That specifies access rights
- It is unforgeable
- Holding a capability is a necessary and sufficient proof of authority to exercise the capability’s access rights on the named object.
EROS: A Simple Abstract Machine

- Only two objects:
  - Pages hold application data bytes (machine defined)
  - Nodes hold capabilities (32)
  - Processes are constructed out of pages and nodes
  - Capabilities unforgeably encode object names and permissions
  - Entire machine is persistent

- Style: persistent microkernel
  - Much is outside the kernel.
  - There is a set of “primordial” objects (c.f. Scheme or Java)
  - System image integrity assured by transacted checkpoint

- Pure Capability System
Nodes ↔ Memory Mappings
Nodes ⇔ Processes

Process = Secure Component
Basic Idea

- Design a “toolkit” where most components can be used securely in any situation.
- Design an operating system (or runtime) that lets these be composed to form secure systems.
- Build software by composing secure components.
Essential Building Blocks

♦ Assume: Software is binary. You cannot practically “inspect” it.
♦ Solution: control the *environment* of the software.

Program → Program → Environment Boundary
Confinement Policy

- Software inside the environment boundary can transmit across the boundary, but only if the communication channel is authorized by the client.
- Software outside the environment boundary cannot examine things inside the boundary (unless the stuff inside permits it).

Confinement == Sandboxing + Proprietary Content
A Primitive Building Block for Reference Monitors

- Confinement provides sandboxing at the OS level
- Confinement allows secure reference monitors to be constructed outside the kernel. [KeySafe]
The Constructor

♦ Constructor certifies confinement.
♦ Yield is confined if initial capabilities are safe
♦ A capability is “safe” if
  – It trivially provides no mutate authority
  – It is a read-only, weak capability
    • read-only implies no direct writes
    • weak implies no transitive writes
  – It is a requestor capability to a frozen constructor whose yield is confined
♦ Once yield is running, constructor out of the loop.
Some Issues

♦ Feasibility on commodity hardware:
  – Historically capability systems have been slow
  – Security *always* penalizes performance
  – Goal: meet or exceed conventional monolithic performance

♦ Microkernel approach known fatal  [Bershad/Chen ‘93]

♦ Good choice of system objects not known
  – OS level (solved)
  – Application level (in progress, but promising)

♦ Selective revocation challenging  ([Redell], but see KeySafe)

♦ Verifiable security policies uncertain  [Karger ‘84, Boebert ‘84]
Performance

Pipe BW (Mbyte/sec)
Process Create (ms)
Directed CSwitch (us)
Pipe Lat (us)
Grow Heap (us)
Page Validate (us)
Null I/O (us)
Triv. Syscall (us)

log(microseconds) except as noted

Semantically comparable operations

EROS
Linux v2.0.30
Secure Distribution?

- Proxy objects at the boundaries
- Cryptographically protected links
- Secure endpoints