Flexible Containment for Executing Untrusted Code

Darrell Hyatt
Introduction

- Standard security model is insufficient for security-critical applications

- Result – sandboxing (among other things)
  - Tailoring of policies to programs
  - Confining programs, supporting policy enforcement
  - Variety of options available for design
    - But these have both strengths & weaknesses
Sandboxing Process

- Process creates a new sandbox by calling sbxcreate()
  - Handle is returned to the process
  - Only the process can access the new sandbox
- Process configures the sandbox (including the privileges allowed) and forks; child inherits the sandbox descriptor
- Using sbxapply(), child applies the sandbox to itself
  - Forfeiture of control of all other sandbox descriptors, including the one it is in
Sandboxing Process

- Process (parent) retains full control of the sandbox
  - may launch other programs within it
  - may close sandbox descriptor
    - Eliminates parent as possible point of attack
    - Sandbox now unalterable by any process
    - Processes external to sandbox can suspend or terminate processes within the sandbox
- Descendants of a child inherit its sandbox
- Sandboxes are destroyed by the kernel through reference counting
Design Considerations for Sandboxing

- Organization of Privileges
  - Extensible – should be able to enforce new security policies as the system grows
  - Privileges are denied by default
  - Expressive – should be able to cope with fine-grained security policies
    - Binary – allow or deny
    - Quantitative – number indicating an allowed limit (e.g. memory allocation)
  - Set-oriented view of privileges
    - Operations expressive, theory is understood, clarifying that policies are enforced
Privilege Set Operations

- Create union \[ z := x \| y \]
- Union w/ self \[ x := x \| y \]
- Create intersection \[ z := x \& y \]
- Intersect w/ self \[ x := x \& y \]
- Create complement \[ z := \neg x \]
- Complement self \[ x := \neg x \]
- Bob changes positions: Personnel → Finance
  - \( B := (B \& \neg P) \| F \)
- Bob needs access to George's non-confidential files
  - \( B := B \| (G \& \neg Gc) \)
Privilege Interval Lists

- Interval list – allow specification of intervals of values over a fixed range
  - Include – merge intervals
  - Exclude – remove interval
  - Intersection, Union and Complement
  - Query point – checks if integer exists in the interval
Privilege Sandbox Sets

- Sandbox sets – allow sandbox processes to perform actions in relation to other processes
  - A sandbox process is allowed to access processes in its own box or in descendants
  - Access rights may be delegated to child sandboxes
  - Processes not in any sandbox are considered part of the “null sandbox”
  - Intersections, Unions, and Complements may be computed
Design Considerations for Sandboxing

- Location of enforcement mechanisms
  - Runtime environment
    - Allows tailoring of security policies
    - Security mechanisms can be fine-grained
    - Only supports code that uses the runtime
  - Sandbox program (e.g. Proof-carrying code)
    - Security mechanisms can be fine-grained
    - Requires binaries to be modified
    - Not applicable to all types of programs
  - User space (e.g. developer sandbox)
    - Easily deployed on existing systems
    - Process tracing is not applicable to setuid programs
  - Kernel
    - Unlimited options and most resistant to attack
    - Code difficult to write & debug (if accessible at all)
Design Considerations for Sandboxing

- Monitoring
  - Passive – sandbox structures consulted before requests are allowed to proceed
  - Active – restrictions enforced by separate processes that monitor programs as they execute

- Application Scope
  - Global – enforcing restrictions on all programs
  - Local – program is isolated w/in its sandbox

- Access Control – mandatory vs. discretionary

- Policy Enforcement – static vs. dynamic

- Sandbox Privileges – generic vs. specific

- Destruction – disposable vs. persistent sandboxes
Sandbox vs. ACLs – opposing points of view

- ACLs associate privileges with those requesting the use of resources
- Sandboxes associate privileges with the resources themselves
- Intended that the two complement one another