Labels and Event Processes in the Asbestos Operating System

Petros Efstathopoulos, Maxwell Krohn, et al.

KARTHIK ANANTAPUR BACHERAO
10/28/2005

MOTIVATION

- Computer Systems do not provide adequate security
- Exploitable software flaws (Buffer Overflows, etc)
- Source of Problem:
  - Bugs in Software
  - Users willing to run untrusted code
- No isolation of services

Motivation (Contd)

- Principle of Least Privilege (POLP) not enforced.
  - Each bit of code that executes in a machine should run with least amount of privilege.
  - Developers should follow five requirements:
    - Split application into protection domains or compartments
    - Assign exact privileges to the compartments
    - Engineer communication between compartments
    - Compartments should be isolated from one another
    - Should be easy to perform a security audit

OUTLINE

- SECURITY MODELS
- ASBESTOS OS
- ASBESTOS LABELS
- ASBESTOS EVENT PROCESSES
- PERFORMANCE

Security Models

- Mandatory Access Control:
  - Power with the owner of the system.
  - Uses labels.
  - Generally employs a variant of the "Property":
    - Whenever a process P can observe Object O1 and modify Object O2, O2's security level should dominate O1's.
- Discretionary Access Control
  - Security by Ownership.
- POLP with MAC

Asbestos: A New Operating System

- Asbestos should support efficient, unprivileged and large-scale server applications whose application-defined users are isolated from one another by the operating system according to application policy.
- A message passing micro-kernel based architecture.
- New Labeling and isolation mechanism:
  - Asbestos labels provide both mandatory and discretionary access control
  - Decentralized MAC
  - A process can bypass the "property" by declassifying information
Asbestos Labels (Contd)

**LABEL BASICS**

- **Handles:**
  - 64-bit unique identifiers to name compartments.
  - Handles are members of the ordered set $\{*, 0, 1, 2, 3\}$.
- **Labels:**
  - A function from handles to levels.
  - Levels are represented by $\{*, 0, 1, 2, 3\}$.
  - Label $\ell(h) = \max\{A(h), B(h)\}$ for all $h$.
  - Least Upper Bound: $A \cup B(h) = \max(A(h), B(h))$.
  - Greatest Lower Bound: $A \cap B(h) = \min(A(h), B(h))$.

**Asbestos Labels (Contd)**

- **Label Basics (Contd):**
  - Each process in Asbestos has two labels:
    - A send label $P_s$.
    - A receive label $P_r$.
  - A process $P$ may send to process $Q$ if:
    - $P_s \leq Q_r$.
  - When the message is delivered, $Q_s$ send label is contaminated by $P_s$ send label
    - $Q_s = Q_s \cup P_s$.
  - In Send label: lower levels are more permissive.
  - In Receive label: lower levels are more restrictive.

**Asbestos Labels (Contd)**

- **Four Levels:**
  - Default send level is 1, default receive level is 2.
  - Default labels are in the middle of the labeling order.
  - Flexible isolation schemes possible.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_s$</td>
<td>(h, 3, 1)</td>
<td>(1)</td>
<td>(h, 2, 1)</td>
</tr>
<tr>
<td>$Q_r$</td>
<td>(2)</td>
<td>(h, 0, 2)</td>
<td>(h, 1, 2)</td>
</tr>
</tbody>
</table>

**Effective Labels**

- Ability to taint different user processes in different ways.
- Uses Contamination and Verification Labels $C_s$ and $V$.
  - Label $E_{C_s}$:
    - $E_{C_s} = P_s \cup C_s$.
  - Label $E_{V}$:
    - $E_v = Q_r \cup V$.
Asbestos Labels (Contd)

- **Declassification Privileges**
  - Uses "-level to decentralize declassification.
  - A process $P$ with $\text{Ps}(h) = \ast$, is said to have declassification with respect to $h$.
  - Modified equation:
    - $Q_s = Q_s \cup (Es \cap Q_r^\ast)$ is same as:
      - $Q_s(h) = Q_s(h)$, if $Q_s(h) = \ast$
      - $Q_s(h) = \text{Qs}(h)$, otherwise

- **Decontamination**
  - A process with declassification privilege can decontaminate other processes.
  - Done by lowering their send labels and raising their receive labels.
  - Uses two optional arguments $Ds$ and $Dr$ to the send system call
  - Modified Equations:
    - $Es \leq Q_r \cup Dr$
    - $Q_s(h) = (Q_s \cap Ds) \cup (Es \cap Ds) \cup (Es \cap Q_s^\ast)$, $Q_r = Q_r \cup Dr$

Asbestos Labels (Contd)

- **Preventing Contamination**
  - To prevent processes from getting contaminated unwillingly.
  - Every port $p$ is associated with a port receive label $pr$.
  - This acts like a verification label imposed by the receiver rather than the sender.
  - Modified Equation:
    - $Er = Q_r \cap \forall \cap pr$

Event Processes

- **Handling multiple users data:**
  - User level threads
  - Separate Process per user
  - Simple event-driven dispatch loop:
    ```
    event = get_next_event();
    user = lookup_user(event);
    if(user not yet seen)
      user.state = create_state();
      process_event(event, user);
    ```
  - No isolation of user states.

Asbestos Event Process

- **Isolates different event process’s state.**
- Each event process associated with one base process
- Event process’s kernel state consists of:
  - Send label, Receive label, Receive rights for a port and a set of memory pages and book keeping information.

Asbestos Event Process (contd)

- A typical event process dispatch loop
  ```
  ep_checkpoint(&msg);
  if(!state.initialized())
    initialize_state(state);
  state.reply = new_port();
  }
  process_msg.initState(state);
  ep_yield();
  ```
- Uses the following system calls:
  ```
  ep_checkpoint, ep_yield, ep_clean, ep_exit.
  ```
Web Server Design using Asbestos

Data Path of a Web Request:

1. netd accepts incoming connection. Sets Ucr to (Uc 0, 2).
2. netd grants ok-demux Uc at level *.
3. Authenticates user.
4. If authenticated, idd grants ok-demux Ut, Ug at level *.
5. ok-demux grants Ut * to netd. Netd raises Ucr to (Uc 0, Ut 3, 2).
6. If the requested service exists in W, ok-demux forwards requis, grants Ug *, and contaminates it with Ut 3.
7. W returns from ep_checkpoint into W(u).
8. W(u) creates new port Uw, grants it to netd at *.
9. W(u) calls ep_exit.

Performance

- Memory Use
  - Cached session: Requires additionally ~1.5 4KB pages
  - Active sessions: Requires additionally ~9.5 4KB pages

- Web Server Performance
  - Throughput
    - With one cached session, the avg. no. of connections is greater than that of apache's
  - Latency
    - With 1000 cached sessions, almost same as that of apache's

- Label Costs
  - Linear degradation in performance.

Thank You!
Questions?