Asbestos Operating System

Presented by Sherley Codio and Tom Dehart
This Talk

- Recap on Information Flow
- Asbestos Overview
- Labels
- Special Rules
  - Discretionary Contamination
  - Declassification/Decontamination
  - Preventing Contamination
- Implementation
Information flow

- Long-term confinement of information to authorized receivers
- Controls how information moves among data handlers and data storage units
- Applied at language, system, or application levels
Information Flow - Example

- Guarantee that the anti-virus (AV) scanner cannot leak to the network any data found in its scan of user files

- Possible leak methods
  - Send data directly to a network connection
  - Conspire with other processes (e.g., sendmail or httpd)
  - Subvert another process and use its network access to send data
  - Leave data in /tmp for other processes (e.g., the AV update daemon) to send
  - Use other in/direct means of communication with the update daemon
Information Flow – Lattice

least upper bound

\[(S, [mil]) \oplus (S, [dip]) \rightarrow (S, [dip, mil])\]

\[(TS, []) \oplus (S, [dip, mil]) \rightarrow (TS, [dip, mil])\]

greatest lower bound

\[(TS, [dip]) \otimes (TS, [mil]) \rightarrow (TS, [])\]

\[(TS, []) \otimes (S, [dip, mil]) \rightarrow (S, [])\]
Problem

- Web servers regularly divulge private information through exploitable software flaws (SQL injection, buffer overruns, etc).
- Instead of trying to fix all exploits, design a system that limits the impact of exploits.
The principle of least privilege: each system component should have the minimum privilege required to accomplish its task.

We still have a problem: current operating systems can’t enforce the principle of least privilege.
A prototype operating system that provides labeling and isolation mechanisms that help contain the effects of exploitable software flaws.

Labels

Event Processes
Asbestos labels determine which services a process can invoke and with which other processes it can interact.

Labels can also track and limit the flow of information within system- and application-defined compartments.
Asbestos event processes lets server applications isolate many concurrent users.

Event processes keep a private state for every user but isolates each state so that exploits affect only one user’s data.
“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.”
Large-scale

- Network requests from a dynamically changing population of thousands or even hundreds of thousands of users
- Examples: web commerce, bulletin board systems, etc.
Efficient

- Asbestos server should be able to cache user data with low overhead.
- This would be simple with a trusted cache but we want to isolate each user’s data (thus we use the event process abstraction).
The system administrator has granted the application the minimum privilege required to complete its job.
Each application can define its own notion of principal. One application’s users can be distinct from another’s or they can overlap.
- A process acting for one user cannot gain inappropriate access to other users’ data.
- The application defines which of its parts should be isolated, and how via application policies.
- Should also support flexible sharing among users for data that doesn’t need isolation.
Asbestos must support mandatory access control that isolates processes by tracking and limiting the flow of information.
Mandatory access control (MAC) systems provide enforcement of security policies by following links between processes.

Operating systems have been using labels to enforce policies for a while.
Related Work

- Labels assign each subject and object a security level which usually consists of a hierarchical sensitivity classification (like unclassified, secret, top-secret) and a set of categories (nuclear, crypto, etc).

- To observe an object, a subject’s security level must dominate the object’s security level.

- For example, a file with secret, nuclear data should only be readable by processes whose clearance is at least secret and whose category set includes nuclear.
Asbestos labels differ from previous work since Asbestos lets any process create label categories, or *compartments*.

It also provides a few novel features like temporary voluntary restrictions and split send/receive labels with different defaults.
Each process in Asbestos has two labels

- Send label $P_S$: tracking current contamination
- Receive label $P_R$: tracking contamination limits (clearance)

A process $P$ may send to process $Q$ if $P_S \sqsubseteq Q_R$.

When the message is delivered, $Q$'s send label is contaminated by $P$'s send label $Q_S \leftarrow Q_S \sqcup P_S$. The least upper bound.

In Send label: lower levels are more permissive
In Receive label: lower levels are more restrictive
Labels: A function from handles to levels.
- \{a 0, b 1, 2\}

Handles:
- 61-bit unique identifiers to name compartments.
- Privileges are represented by Levels, members of the ordered set \{*, 0, 1, 2, 3\}

- Label Comparison:
  \[ L_1 \subseteq L_2 \text{ iff } L_1(h) \leq L_2(h) \text{ for all } h. \]

- Least Upper Bound
  \[ (L_1 \sqcup L_2)(h) = \max(L_1(h), L_2(h)) \]

- Greatest Lower Bound
  \[ (L_1 \cap L_2)(h) = \min(L_1(h), L_2(h)) \]
- {*, 0, 1, 2, 3 }

- **Send labels:**
  - * is the lowest or most privileged level
  - 1 usually corresponds to the absence of taint (default)
  - 2 to a “partial taint”
  - 3 to full taint, where most communication is prevented (least privileged level)

- **Receive labels**
  - 1 prevents communication with any tainted process
  - 2 is the default
  - 3 indicates the right to be tainted arbitrarily

- 0 is used for integrity and capabilities
- $U_S \leq UT_R$
  - $U_S(u_T) = UT_R(u_T)$, $U$ can send to $U_T$

- $V_S$ is not $\leq UT_S$
  - $V_S(v_T) = 3$, $UT_R(v_T) = 2$
  - $V$ cannot send to $U_T$
To avoid the exploitation of covert channels
Special Rules

- Discretionary Contamination (Effective labels)
- Declassification/Decontamination (*-level)
- Integrity (Grant handles)
- Preventing Contamination (Ports)
In order to maintain the system’s information flow properties, the file server must label files.

So a process that reads user $u$’s files must become tainted with $u_T \triangleleft 3$.

But the file server must be able to taint different users’ processes in different ways.
In Asbestos, the file server can selectively taint messages with the appropriate handle by providing an optional \textit{contamination label} $C_s$

This label raises the sender’s send label to a new \textit{effective send level} $E_s = P_s \cup C_s$

Thus, the effective label, not the true send label, is used to check information flow and contaminate the receiver’s send label.
1. $P_S \subseteq Q_R$

2. $Q_S \leftarrow Q_S \cup P_S$

3. $E_S \subseteq Q_R$

4. $Q_S \leftarrow Q_S \cup E_S$
Discretionary Contamination

- Since contamination only restricts information flow, it requires no special privilege (i.e., we still maintain the idea of least privilege)
- Thus, processes can arbitrarily contaminate the messages they send.
- When processes can control their interactions with the label system (without violating information flow properties) the label system can implement more security policies.
- $\star$ level is used for declassification
- $P_S(h) = \star$ has declassification privilege
- $P$ cannot be contaminated by other processes
- If $P_R$ receives from $Q_S$ ($h = 3$)
  - $P_S(h)$ remains $\star$, the lowest level
- $P$ can forward data from $Q$ to less tainted processes (declassifying)
Only a process itself can remove $\star$ levels from its send label.

\[
L^* = \begin{cases} 
\star & \text{if } L(h) = \star \\
3 & \text{otherwise.}
\end{cases}
\]
Process with privilege can distribute privilege to other processes:
- Forking
- Using a mechanism called decontamination.

A process with declassification privilege can decontaminate other processes’ labels by:
- Lowering their send labels
- Raising their receive labels

Two more optional label arguments to the send system call:
- decontaminate-send label $D_S$
- decontaminate-receive label $D_R$. 
- Decontaminate-send label: lower the receiver’s send label
- Decontaminate-receive label: raise the receiver’s receive label

- The operations make the system more permissive
  - require special privilege with respect to the handle involved

- Whenever a decontamination label change the receiver’s labels, the sender controls the relevant compartments:
  \[ P_S(h) = \star \text{ whenever } D_S(h) < 3 \text{ or } D_R(h) > \star \]
The file server can accept requests from any user without fear of contamination and can declassify user data when needed.

But, any useful file server must also implement an \textit{integrity} policy to prevent processes from overwriting users’ data.

Two types of integrity: discretionary and mandatory.
In a discretionary policy, only processes that *speak for* user $u$ can write to $u$’s files, but their writes are free to incorporate data from less trusted sources.

Speaking for $u$ is a positive right, not a taint. Thus, a process speaking for $u$ does not imply that it has read any of $u$’s secret data.

So, we need a new compartment to represent “speaking for” a user $u$. 

Discretionary Integrity
Thus, we introduce a new handle $u_G$, user $u$’s grant handle.

A process can speak for $u$ only if $P_S(u_G) \leq 0$. Hence, the file server must verify $P_S(u_G) \leq 0$ before accepting a write to $u$’s file from $P$.

Asbestos supports such checks with an optional label argument to the `send` command, the `verification label` $V$. 
The verification label temporarily *lowers* (restricts) the receiver’s effective receive label.

The sender thus proves with $V$ that its labels are below a constraint independent of the receive label.

6. $E_S \subseteq Q_R \cup D_R$

8. $E_S \subseteq (Q_R \cup D_R) \cap V$
Since we know that $E_S = P_S \cup C_S$, we also know that $P_S \subseteq V$.

Thus for the verification to succeed, $V$ needs to be an upper bound on the sender’s send label.

In the file server example, a process writing to $u$’s file must supply $V = \{u_g 0, 3\}$ to prove it speaks for $u$. The file server then verifies the process speaks for $u$ by checking $V(u_g) \leq 0$. 
The 0-level allows us to implement mandatory integrity policies.

A process $P$ with $P_s(u_G) = 0$ can speak for $u$, but since 0 is less than the default send level of 1, it cannot further disseminate the privilege.

For example, the moment $P$ receives a message from $Q$ that does not speak for $u$ ($Q_S(u_G) \geq 1$), $P_s$ will become tainted and $P$ will lose its ability to speak for $u$.

This means that $P$ can’t act for $Q$ and pass along low-integrity data into $u$’s files.
Application can choose to ignore message after examination

- Application’s labels have already been contaminated with the message’s taint.
- Taint cannot be undone
- Verification label can flexibly verify integrity, cannot prevent inappropriate contamination
A way to shift a simple form of message filtering into the kernel
In Asbestos, the integration of communication ports with the label system
The result prevents undesired contamination
Port namespace, the same as the handle value space
  - Port names can be used as label compartments

Every port \( p \) is associated with a port receive label \( p_r \).
  - \( p_r \) is used to lower, or restrict, the process’s receive label, for messages delivered to that port.
  - It acts like a verification label imposed by the receiver

\[ E_S \equiv (Q_R \cup D_R) \cap V \cap p_R \] (label check equation)

Port label restricts how much a receive label can be decontaminated

Port labels, like verification labels, are entirely discretionary

Each process controls the port labels for all ports for which it has receive rights
Labels alone don’t work well for processes with multiple users’ private data since processes get over-contaminated.

Declassification privilege for each relevant user would leave them over-trusted and vulnerable.

User-level threads are efficient, but don’t provide isolation.

Forking a separate process per user provides isolation, low performance.

An abstraction combining performance benefits with isolation benefits.
- 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 labels make *mandatory access control* more practical

- Labels provide decentralized compartment creation & privilege
- Event processes avoid accumulation of contamination