Complete Information Flow Tracking from Gates Up

Mohit Tiwari, Xun Li, Hassan M G Wassel, Frederic T Chong, Timothy Sherwood

Presented by Mengjia Yan
Based on slides from Mohit Tiwari
Goal: Non-Interference

- **Non-Interference**: a change in a High input can never be observed or inferred from changes in the Low output. That is, High data should never leak to Low.

- **Confidentiality-Integrity Duality**: “High” is more conservative label. Secret or Tainted/Untrusted.
Information Flow for Privacy

- General lattice policies
- Secret vs. Unclassified Data
  - Secret: data with restricted access permission
  - Unclassified: data with unrestricted access
- Enforce the property of non-interference:
  - Verify information never flows from high to low.
  - Secret information is never used to modify unclassified data
Information Flow for Integrity

• Trusted vs. Untrusted Tasks
  • Trusted: processes which are critical to the correct functionality of the space vehicle systems
  • Untrusted: mission processes, diagnostics, anything whose malfunction will not cause a vehicle loss

• Enforce the property of non-interference:
  • Verify information never flows from high to low.
  • Untrusted information is never used to make critical (trusted) decisions nor to determine the schedule (real-time)
Threat Model

• Low output can include
  • Program output
  • Timing
  • Contention on system resources

• Not include
  • Untrusted hardware component problem
  • Physical attacks that may tamper with memory
  • Non-digital side-channel attacks (power distribution and RF signals)
Highlights

• A secure SW/HW co-design which is verifiable

• Gate-level information flow tracking
  • More precise than conventional IFT

• ISA restrictions to prevent taint explosion
  • Handling conditional branch
  • Handling loops
  • Handling loads/stores

A new way to look at IFT from a new perspective.

Usage: GLIFT + Information Flow Policy
The Vision: Hardware Design for Software Security Verification

Applications
Language
Compiler/OS
Instruction Set (ISA)
Microarchitecture
Logic Gates

Sound Information Flow Analysis

Hardware/Software Design for Verifiable Security

Security Properties
Information Flow Analysis

• Information flows through **Space**
  • Registers, Memory, Micro-architectural state etc.

\[
\text{out1} = \text{ld}(\text{high})
\]

if (high == 1)
  \[
  \text{out1} = 1
  \]
else
  \[
  \text{out2} = 0
  \]

(implicit flow)  (explicit flow)
Static and Dynamic Information Flow Tracking

- Static analysis is conservative (need alias analysis for precise results)
- Dynamic analysis has difficulty in analyzing implicit flow

\[
\begin{align*}
\text{out1} &= \text{ld}(\text{high}) \\
\text{out2} &= \text{ld}(\text{low})
\end{align*}
\]

(\text{explicit flow})

\[
\begin{align*}
\text{if } (\text{high} == 1) \\
\text{else}
\end{align*}
\]

\[
\begin{align*}
\text{out1} &= 1 \\
\text{out2} &= 0
\end{align*}
\]

(\text{implicit flow})

out2 is tainted if the address or the memory value is tainted
Information Flow Analysis

• Information flows through **Space**
  • Registers, Memory, Micro-architectural state etc.

• Information flows through **Time**
  • Observable events such as PC, I/O channels etc.
The paper addresses two challenges

• How to account for all information flows in a system?
  ➔ So that the security property can be verifiable
  ➔ Avoid taint explosion

• How to construct practical systems that won’t leak?
  ➔ Use the concept of GLIFT to guide the design
High-level View: Track all flows

• Flatten design to a (giant) state machine
• Does every output have desired label?
High-level View: Track all flows

- **Insight:** All flows explicit at the gate level
High-level View: Track all flows

- **Outputs**: Logic function of state and inputs
- **Output Labels**: Logic func. of state, inputs, and labels
Analysis Technique: GLIFT

AND Shadow AND for labels

Conservative.
If one of a and b is tainted, the output is tainted.
Motivation: Require Precise Information Flow

• Conventional OR-ing of labels **monotonic**
Precise Information Flow: AND Gate

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

When a=0, b can not affect the value of the output. → no-interference

Use both inputs and input labels
Analysis Technique: GLIFT
Sound Composition of Shadow Logic
MUX: Gatekeeper of trust
Conditional execution taints critical state (PC)
Convert Implicit Flow to Explicit Flow

If (secret == 1)
out = 1

tmp = 5

PO = secret
(P0) out = 1

tmp = 5
Convert Implicit Flow to Explicit Flow

\[
\text{if (secret==1)} \\
\text{out = 1} \\
\text{tmp = 5}
\]

\[
\text{P0 = secret} \\
\text{(P0) out = 1} \\
\text{tmp = 5}
\]
Similar Mechanisms for Loop/Load/Store

• Variable length loops $\rightarrow$ fixed size loops (bounding)
• Indirect loads/stores $\rightarrow$ Direct loads/stores

- Harder to program and inefficient
+ Verifiable system

• Recommend to read their follow-on work:
  • Execution Leases: A Hardware-Supported Mechanism for Enforcing Strong Non-Interference; Tiwari et al.; MICRO’09
Evaluation

+ Security
- Area overhead/Power consumption
- Performance overhead
- Programmability

Appropriate use cases:

- When critical or sensitive operations need to be performed, a co-processor augmented with these abilities could be an attractive option.
Discussion Questions
Discussion Questions on Taint Tracking

• Who designates an input as untrusted/trusted? Where in the architecture/implementation does an input first get marked as untrustworthy?

• Can/should there be a method of promoting data from untrusted to trusted? (from High to Low)

• How does the GLIFT method handle optimizations such as out-of-order execution, speculation etc? Will the proposed architecture be able to detect covert and side channel attacks such as Meltdown and Spectre?
Example MLS System

Example Satellite Application. [Tzvetan Metodi, Aerospace Corp.]

Note: Since this is not a real schedule, the processes are not in any sensible execution order

- Kernel and Diagnostics
- Time Keeping
- Crypto
- Command Telemetry Interface
- I/O Secret
- Mission Secret
- Mission Unclass.

Primary Execution Schedule

Execution Time
Example: Satellite System

- **Untrusted & Secret**
  Libraries (e.g. encryption) that operate on Secret data

- **Untrusted & Unclassified**
  Diagnostics, Telemetry Interfaces

- **Trusted & Secret**
  Custom code on Secret data

- **Trusted & Unclassified**
  Kernel, Interrupt Handlers (Unclassified), Time Keeping Programs
Discussion Questions on Use Cases

• One specific use case: What if we needed to load in a new firmware blob to compute a new function. Could we send it in encrypted and have a way of validating and then trusting it?

• In the end, it seems the ISA is the secure step, and the trust bits are just useful in validating the design. (Does the restricted ISA make program secure against side channels?)

• This kind of processor would be a pain to program. If most applications on it are small, important kernels, such as AES, would it not be better to produce a specially tuned ASIC/IP core?

• Are there any programs or algorithms that are rendered impossible (or extremely difficult) to write as a result of the limitations that they've placed on loops?
Discussion Questions on Future Work

• Rather than implementing a CPU with this restricted ISA, since this is used only for edge case computation, could an FPGA-based enclave in a traditional CPU be used with this ISA instead as a cost-effective implementation?

• Rather than apply the concept of gate level flow tracking to the ISA, I envision further work that could apply the same concepts to FPGA tooling.

Great idea.
Discussion Questions on Side Channels

• How does the GLIFT detect a side channel/covert channel? What is the “sink” of taint tracking in such cases?

• If we do not plan to use GLIFT to track side channel leakage, do we need to ISA restriction on indirect loads? (not indirect stores)

• How GLIFT different from static taint analysis and traditional dynamic taint analysis?