Motivation

• Engineering Large Aerospace Software Systems is Difficult
  - Integration of multiple potentially unreliable components into a robust system
  - Competing Requirements: Safety vs. Security and Reliability vs. Security

• Traceability of Requirements
  - Requirements may trace incorrectly in Waterfall or vanish altogether
  - Typically, only a few static instructions write to a given program location

• Search in Abstract Space
  - Define property (PC) XOR (tag)
  - Complex property (PC) XOR (tag)

• Write to a location
  - Aid in discovering missing/incorrect requirements

• Security Critical Data
  - System call arguments
  - Function Call and Return addresses
  - Control-flow data
  - Pointers on the stack and the heap

• Decision Aids:
  - Small Aircraft Transportation System-High Volume Operation (SATS-HVO)

• Quality-Oriented Design & Validation

Security: Verification and Reachability

Safety: Boolean Predicates

Hybrid System

Search in Abstract Space

Counter Example

Additional Predicates

Property

Holds

Real Counter Example Found

Safe Sets: Ellipsoidal Overapproximations

• Initial Set and Input/Control Set can be bounded by ellipsoids

• Ellipsoidal matrix form:

\[
\phi(t) = \phi(0) + \int_0^t \phi'(s) ds
\]

Find a tight external overapproximation such that the ellipsoid touches the exact reachable set at time \( t \).

• Attempt to Verify Property

• Refine the overapproximation to the reachable set using counter-examples to eliminate unreachable states

• Verify property over refined overapproximation

Project Vision

• Refinement Techniques for identifying residual flaws in application-specific settings
  - Requirement Flaws
  - Implementation Flaws

• Mechanisms for detecting and/or masking of hazards, accidental errors and security attacks
  - Application specific assertion generation
  - Detection of runtime data errors

• Assessment/Benchmark in Field Conditions similar to Real World settings
  - Testbed available for experimentation

• Combination of simulated and real scaled aircraft and flight applications

Security: Information-Flow Signatures

• Use detection of program data-flow violations as an indicator of malicious tampering with the system
  - Prevent or isolate from exploiting disconnect between source-level semantics and execution semantics of program

• Employ a compile-time static program analysis to extract instructions allowed (at runtime) to write to a given memory word

• Sign each identified location by the PC(tag) of instruction(s) allowed to write to this location

• Typically, only a few static instructions write to a given program location

• Special hardware maintains a tag for each memory word

• Write to a location
  - Create runtime signature corresponding to the location

• Reference to a location
  - Check the tag against the set of allowed signatures (derived at compile-time)
  - If there are no matches – the operation is disallowed

• Extract a set of security preconditions, which if satisfied guarantee secure code

• Automatically translated to formal representation

• Security Critical Data
  - System call arguments
  - Function Call and Return addresses
  - Control-flow data
  - Pointers on the stack and the heap

• Security Testing in presence of failures

Testbed

• Inject/Exert Errors to cause misbehavior
  - Evaluate detection coverage

• Verify timing assumptions under extreme environmental conditions
  - Wind, rain and snow
  - Communications
  - Power consumption
  - Malicious events

• Discover and remediate security issues that have been isolated to implementation

• Underwrite 15 Visa and P-III Pentagons

• Onboard Embedded Systems
  - Picture: Xeop, Canavas, PicCo V10