Using Formal Methods to Enable More Secure Vehicles: DARPA's HACMS Program

Kathleen Fisher
Tufts University
16 April 2015

(Slides based on original DARPA HACMS slides)
Pervasive Vulnerability to Cyber Attack

SCADA Systems  Medical Devices  Vehicles

Computer Peripherals  Appliances
Modern Automobile: Many Remote Attack Vectors

<table>
<thead>
<tr>
<th>Mechanic</th>
<th>Short-range wireless</th>
<th>Long-range wireless</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="source" alt="Mechanic" /></td>
<td><img src="source" alt="Short-range wireless" /></td>
<td><img src="source" alt="Long-range wireless" /></td>
</tr>
</tbody>
</table>

Source: www.custom-build-computers.com
Source: CanOBD2
Source: www.diytrade.com
Source: Koscher, K., et al. “Experimental Security Analysis of a Modern Automobile”
Securing Cyber-Physical Systems: State of the Art

Control Systems

- Air gaps & obscurity
  - Forget the myth of the air gap – the control system that is completely isolated is history.  
  -- Stefan Woronka, 2011
  Siemens Director of Industrial Security Services
  
- Trying to adopt cyber approaches, but technology is not a good fit:
  - Resource constraints, real-time deadlines
  - Extreme cost pressures
  - Patches may have to go through lengthy verification & validation processes
  - Patches could require recalls

We need a fundamentally different approach

Cyber Systems

- Anti-virus scanning, intrusion detection systems, patching infrastructure
- This approach **cannot** solve the problem.
  - Not convergent with the threat
  - Focused on known vulnerabilities; can miss zero-day exploits
  - Can introduce new vulnerabilities and privilege escalation opportunities

October 2010 Vulnerability Watchlist

<table>
<thead>
<tr>
<th>Vulnerability Title</th>
<th>Fix Avail?</th>
<th>Date Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Kernel Controller Area Network Protocol Local Privilege Escalation Vulnerability</td>
<td>No</td>
<td>8/25/2010</td>
</tr>
<tr>
<td>Red Hat VSDM Module SSL Connection Denial of Service Vulnerability</td>
<td>Yes</td>
<td>8/24/2010</td>
</tr>
<tr>
<td>PHP ‘base64_encode’ Function off-by-one Buffer Overflow Vulnerability</td>
<td>No</td>
<td>8/20/2010</td>
</tr>
<tr>
<td>Internet Explorer &amp; IIS/ATM Employee/HTML Sanitization Bypass Vulnerability</td>
<td>No</td>
<td>8/18/2010</td>
</tr>
<tr>
<td>Cisco Unified Wireless Network (UWN) Multiple Security Vulnerabilities</td>
<td>Yes</td>
<td>8/16/2010</td>
</tr>
<tr>
<td>OpenSSL ‘ssl_get_key_exchanged’ Use-After-Free Memory Corruption Vulnerability</td>
<td>No</td>
<td>8/12/2010</td>
</tr>
<tr>
<td>Adobe Acrobat and Reader Font Parsing Where Code Execution Vulnerability</td>
<td>No</td>
<td>8/10/2010</td>
</tr>
<tr>
<td>OpenOffice Impress File Multiple Buffer Overflow Vulnerabilities</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Linux Kernel Pthread ‘stack’ Stack Buffer Overflow Vulnerability</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>vulnerabilities in 2010 show the need for a fundamentally different approach</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1/3 of the vulnerabilities are in security software!
SAT Solvers and Infrastructure Development: Critical Enablers for High Assurance Systems

Interactive Theorem Provers
- seL4 microkernel
  [9000 LoC:C, SOSP 09]
- compCert verifying C compiler
  [6K LoC:ML, POPL 06]

Automatic Theorem Provers
- Verve OS Nucleus
  [1.5K LoC:x86, PLDI 10]
- Baby Hypervisor
  [1K LoC:C, VSTTE 10]

Model Checkers
- Microsoft device drivers
  [30K LoC:C, PLDI 01, CACM 11]
- ADGS-2100 Window Manager
  [16K Simulink blocks, CACM 10]

[A] significant part of the effort in existing projects was spent on the further development of verification tools, on formal models for low-level programming languages and paradigms, and on general proof libraries. The sharing of substantial parts of the verification tools between Verisoft and L4.verified demonstrates that there is a significant degree of re-usability... Future efforts will be able to build on these tools and reach far-ranging verification goals faster, better, and cheaper.

Gerwin Klein, *Formal OS Verification—An Overview.*
HACMS: Clean-Slate Methods for High-Assurance Software

High Assurance: Ensuring Correctness, Safety, Security
**HACMS Program Structure**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Boeing</strong></td>
<td></td>
<td></td>
<td></td>
<td><em><em>DRAPER</em>/AIS/ U. Oxford</em>*</td>
</tr>
<tr>
<td>Pilot-able Unmanned</td>
<td>NICTA</td>
<td>Galois</td>
<td>RC* /U. Minn</td>
<td>Traditional penetration testing; novel formal methods approach</td>
</tr>
<tr>
<td>Little Bird Helicopter</td>
<td>Synthesize file systems, device drivers, glue code; Verified sel4 kernel; Verified RTOS</td>
<td>Embedded DSLs; Synthesize and verify control system code</td>
<td>Compositional verification; Integrated workbench</td>
<td></td>
</tr>
<tr>
<td><em><em>HRL</em>/GM</em>*</td>
<td></td>
<td></td>
<td></td>
<td><em><em>SRI</em>/U. Iowa</em>*</td>
</tr>
<tr>
<td>American-Built</td>
<td><em><em>SRI</em>/UIUC</em>*</td>
<td><strong>SRI</strong></td>
<td></td>
<td><strong>SRI</strong></td>
</tr>
<tr>
<td>Automobile</td>
<td>EF-SMT solvers; Synthesize monitors and wrappers</td>
<td>Synthetic sensors; Synthesis for controllers of hybrid systems</td>
<td>Lazy Composition; Evidential Tool Bus &amp; Kernel of Truth; Vehicle Integration</td>
<td></td>
</tr>
<tr>
<td><em><em>Princeton</em>/Yale/MIT</em>*</td>
<td><em><em>CMU</em>/Drexel/ SpiralGen/UIUC</em>*</td>
<td><strong>Columbia</strong></td>
<td><em><em>UPenn</em>/UCLA</em>*</td>
<td></td>
</tr>
<tr>
<td>Build &amp; verify in Coq OS for vehicle control; Verifying compiler for concurrent code; Program logics</td>
<td>Map high-level spec into low-level C code; Extend Spiral for hybrid systems</td>
<td><strong>SRI</strong></td>
<td>Synthesize attack-resilient control systems</td>
<td></td>
</tr>
<tr>
<td><strong>Kestrel</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synthesize protocols: refinement of high-level spec to low-level implementations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Program Timeline:**
- Kick-Off: Aug 8-10, 2012
- End of Phase 1: Jan 2014
- End of Phase 2: July 2015
- End of Phase 3: Jan 2017

**Performers:**
- 8 Primes (*)
- 22 Organizations Total
Attacker could crash legitimate ground control station & hijack quadcopter in flight.

(Systems were designed to ensure connectivity, not security)
The Evolving SMACCMMCopter Architecture

**Phase 1**

- **Rockwell Collins / UMN**
  - Monolithic SW
  - No RTOS
  - No security
  - Ardupilot Software
  - HW Abstraction Layer
  - FreeRTOS
  - PX4: ARM Cortex M4

- **NICTA**
  - NICTA RTOS
  - Embedded DSL (Ivory)
  - Factored autopilot tasks
  - FM Workbench
  - AADL model of HW & SW

- **Galois**
  - AADL translation, generate glue code
  - Verification of system requirements
  - HAL
  - Generate executable
  - Response to DoS
  - Other
  - Legacy
  - Monitor
  - Ardupilot
  - Glue code
  - FreeRTOS / eChronos
  - PX4: ARM Cortex M4

**Research Vehicle**

- Glue code
  - PX4: ARM Cortex M4
The SMACCMMCopter: 18-Month Assessment

- The SMACCMMCopter flies:
  - Stability control, altitude hold, directional hold, and DOS detection and response.
  - GPS waypoint navigation 80% implemented.
- Red Team: *Found no security flaws in six weeks with full access to source code.*
- Air Team proved system-wide security properties:
  - The system is memory safe.
  - The system ignores malformed messages.
  - The system ignores non-authenticated messages.
  - All “good” messages received by SMACCMMCopter radio will reach the motor controller.
- Penetration Testing Expert: The SMACCMMCopter is probably “the most secure UAV on the planet.”

Open source: autopilot and tools available from http://smaccmpilot.org
Task Summary
- Develop formal architecture model for SMACCMCopter and Boeing’s Unmanned Little Bird (ULB)
- Develop compositional verification tool (AGREE) and architecture-based assurance case tool (Resolute)
- Develop code synthesis tools to generate build code

Performance Summary
- Generated software for Research Vehicle (~75KLOC), 60% high assurance.
- Created AADL models of HW & SW architecture for SMACCMCopter (~3.6K LOC) and ULB
- Extended AGREE tool for compositional reasoning and proved 10 properties about vehicle safety
- Developed Resolute tool for capturing & evaluating assurance case arguments linked to AADL model
- Developed assurance cases for 6 security requirements for information flow and memory safety
- Developed synthesis tool to generate configuration data & glue code for OS/platform HW

References
- Your What is My How, IEEE Software (March 2013)
• Task Summary
  • Synthesize flight-control code, models, and properties from one specification
  • Generate safe low level-code in a scalable way by creating embedded domain-specific languages (Ivory and Tower) and using the host language (Haskell) as an expressive macro language.

• Performance Summary
  • Created Ivory, an open-source EDSL for synthesizing safe low-level code.
    • No buffer overflows, no null pointer dereference, no memory leaks, safe system calls.
  • Created Tower, an open-source EDSL for describing tasks and the connections between them.
    • Hides dangerous low-level scheduling primitives, tracks channel type information, generates AADL code to support analysis and glue-code generation
  • Designed & built SMACCMopter, the first high-assurance UAV autopilot, in <2 engineer-years
    • ~10KLOC Ivory & Tower yields ~50KLOC C++
    • EDSL compilers automatically generate >2500 properties, 6KLOC of architecture models
    • Hardware Abstraction Layer (HAL) from SMACCMPilot in current use by hobbyist UAV community with over 40K members
    • Flew demo at Pentagon (altitude hold, position hold, stability, DOS detection)
  • Designed & built secure communication system:
    • Open-source, low-bandwidth secure communication protocol for small UAVs
    • Transitioned to Boeing and hobbyist community

Reference:
Building Embedded Systems with Embedded DSLs (Experience Report), ICFP (Sept 2014)
NICTA – Technical Area 2

• Task Summary
  • Formally verify OS kernels: seL4 microkernel *(now open-source!)* and eChronos RTOS
  • Synthesize OS components and automated proofs from DSLs (file systems and device drivers)
  • Provide verified CAmkES component platform for rapid system construction

• Performance Summary
  • seL4: First formally-verified OS microkernel
    • Ported to run on SMACCMCopter and ULB
    • Formal specification and implementation of new HW-virtualization features
    • Previously verified: correctness of kernel binary
    • Security properties: integrity and confidentiality
    • Code: 8830LoC C; Proof: 400KLoC Isabelle
  • eChronos: high-assurance RTOS product line
    • 6 RTOS variants generated (76 possible)
    • Code: 2.4KLoC, Variant Specification: 650LoC Isabelle
    • Automatic proof of safe execution.
    • Proof of high-level properties, e.g. scheduler fairness, correct signaling: 5 KLoC
  • Formally Verified OS Components
    • Generated high-assurance FLASH file system from 2 domain specific languages (3KLoC), 10KLoC language correctness proofs. File system design performs on par with mainstream file systems.
    • High-performance CAN and SPI drivers implemented as CAmkES components (5.6KLoC)
    • Security analysis of air-ground link protocol
  • CAmkES: High-Assurance Component Platform
    • Formal semantics for CAmkES component platform ADL (1.2KLoC)
    • Generated glue-code in Isabelle/HOL (generated glue code spec, 5.3KLoC generator)
    • Generated correctness proofs (1.2KLoC) & proof of safe execution

Reference:
Comprehensive Formal Verification of an OS Microkernel, TOCS (Feb 2014)
Air Team: SMACCMCopter

**Phase 2**

- **18**
  - System requirements
  - FM Workbench
    - AADL model of Flight + Mission
  - Verification of system requirements
    - AADL translation
      - generate glue code

- **24**
  - New hardware
    - Driver/protocol verification
  - Embedded DSL (Ivory)

- **30**
  - Distributed functionality

- **34mo.**
  - CAmkES
    - Mission = CommSec + GCS
    - seL4
      - Glue code
    - Odroid: A15
      - CAN
    - Pixhawk
      - Flight Computer
    - Mission Computer

**Rockwell Collins / UMN**

**NICTA**

**Galois**

**Research Vehicle**

- **Flight**
  - Glue code
  - eChronos
    - PX4: ARM Cortex M4
  - HAL

- **CommSec**
  - Glue code
  - eChronos

- **Ardupilot**
  - Glue code
  - seL4
    - CAN
  - Pixhawk
    - Flight Computer
  - Odroid: A15
    - CAN
  - Pixhawk
    - Mission Computer

**RED TEAM**

- **Mission**
  - Glue code
  - seL4
  - Odroid: A15
  - CAN
  - Pixhawk
  - Mission Computer

**Flight Computer**

**Mission Computer**

**Architecture**

- **FM Workbench**
  - AADL model of Flight + Mission
  - Verification of system requirements
    - AADL translation
      - generate glue code

**New hardware**

- Driver/protocol verification
- Embedded DSL (Ivory)

**Research Vehicle**

- **Flight**
  - Glue code
  - eChronos
    - PX4: ARM Cortex M4
  - HAL

- **CommSec**
  - Glue code
  - eChronos

- **Ardupilot**
  - Glue code
  - seL4
    - CAN
  - Pixhawk
    - Flight Computer
  - Odroid: A15
    - CAN
  - Pixhawk
    - Mission Computer
Boeing – Technical Area 1

• Task Summary
  • Integrate HACMS technologies into ULB
    • Substitute eChronos on the Flight Control Computer and seL4 on the Vehicle Specific Module (VSM)
    • Use HACMS-generated secure components to replace elements of the existing ULB software
  • Use the HACMS workbench to verify security properties of the resulting system
  • Support flight demo at the end of Phase 3.

• Performance Summary
  • Ported VSM to run on seL4
    • New hardware supports seL4 memory protection
    • Incorporates Air Team authentication protocol
  • Phase 2 VSM architecture designed to support application of all 3 Air Team technologies
    • Completed initial AADL model of Phase 2 architecture for use in HACMS Workbench

The air team is on-track for a live flight demo on the Unmanned Little Bird at the end of the Phase 3
Tech Transition

- The cyber-physical systems industry knows they have a cyber-security problem.
- Barriers to adoption of HACMS-like technology:
  - Lack of trained workforce (estimated <1000 formal methods experts in US)
  - Lack of commercial support for formal-methods tools (COTS rules!)
  - Difficulties interfacing with legacy tools (thousands) and code bases (millions)
  - Uncertainty about maintainability of high-assurance artifacts
    - The B-52 has been flying since 1955
  - Qualification of tool chain (eg, DO-178C, DO-326)
  - Need for traceability
  - Resource constraints (hardware, SWAP)
  - Multicore (gulp!): chips may be multicore whether desired or not
  - What is the business case? Quantification is important.
Promising, but lots more to do!

**Building High-Assurance Systems**
- Proof Engineering
- Secure composition of high-assurance components
- Architecture-aware proof support
- Verified, reusable, exquisite artifacts

**Formal Tools**
- Verified high-level languages
- First-class domain-specific languages
- Program/Proof synthesis
- Improved tactics for theorem provers
- Model checker/theorem prover integration

**Control Systems**
- Attack-resistant control systems
- Generated safety-envelope monitors
- Models of “good” and “bad” behaviors
- Certifying advanced control systems

**Resources**
- Reasoning about time
- Reasoning about memory usage
- Verified protocols for distributed systems

**Specifications**
- Specification analysis
- Specs for environmental assumptions
- Specs for attacks

**Tech Transition Issues**
Questions?