Applications of Formal Methods in Building High-Assurance Secure Systems

Rance J. DeLong

Computer Engineering Department

Information Assurance

Santa Clara University

Credits

The work described is being performed in the author's capacity as Staff Scientist for Security and Assurance at LynuxWorks in conjunction with SRI International.

Principal sponsors/promoters of the MILS effort are: the Air Force Research Laboratory, and the National Security Agency.

Product and tool vendor partners in the MILS effort are: LynuxWorks, OIS, GHS, University of Idaho, SRI International, and others.

MILS Testbed partners are:
SRI International, Naval Postgraduate School, and others.

* Mr. DeLong is also President and CEO of Trusted Systems Laboratories.

Consumers

MILS target programs and contractors:

Weapons Platforms

F-22, C-130, UCAV, **F35 (JSF), LW, Virginia Class, ...** — **Lockheed-Martin, Boeing, General Dynamics, Raytheon, . . .**

Communications Platforms

JTRS, Crypto MOD, **AIM, PEIP, JANIS, . . .** — **Boeing, BAE, GDDS, L-3, NRL, Rockwell Collins, Harris, ...**

Command and Control

DDX, AEGIS, FCS — **Boeing, Lockheed, Raytheon**

What We Need

- Complete and coherent IDE's
  – Programming, specification, analysis and verification
  – Programming & design "in the large", delegation, interfaces
- Design methodologies that support verification
  – Visser: "programming moving from coding toward design"
  – eliminate manual "coding"
- Modular verification for modular evaluations
- Assurance preservation throughout maintenance
- Verified composability and compositionality
  – Theory and frameworks to support component model
- Shift in perspective
  – "Engineers don't see the benefit"
  – "All that really matters is the code"
- Education to elevate the 90% of programmers
  – But we have to teach them *something specific and usable*

Now, a little history (1)

- The construction of secure operating systems and "security kernels" dates back to the '70's.
  - Multics, MITRE Security Kernel, UCLA Data Secure Unix, Kernelized Secure Operating System (KSOS), Provably Secure Operating System (PSOS)
  - Many computer vendors built security kernel-based operating systems during the '80's and '90's.
- Security kernel (traditional)
  - A general purpose OS, plus enforcement of a security policy
  - ***mandatory access control*** (MAC) such as Bell-LaPadula ***multilevel security*** (MLS), Biba ***multilevel integrity*** (MLI), as well as ***discretionary access control*** (DAC) policies.
- Security Kernel and associated ***trusted software*** constitutes the ***Trusted Computing Base*** (TCB)
- TCB must be ***verified*** to correctly implement policy and be ***evaluated*** by independent body of experts

Now, a little history (2)

- TCBs grew as more and more “trusted software” was added, becoming too large and complex to be verified to a high level of assurance (max EAL 4).
- In a seminal 1981 paper John Rushby observed:
  - Complications result when a security kernel is used to impose a single system-wide security policy
  - Applications requiring guaranteed security often perform simple functions
  - Distributed systems achieve security while avoiding difficulties arising from the security kernel approach
  - A conceptually distributed system may be supported on a single processor by a ***separation kernel***
  - A separation kernel can be verified w/ high-assurance
  - Decouple verification of SK from other components

Today

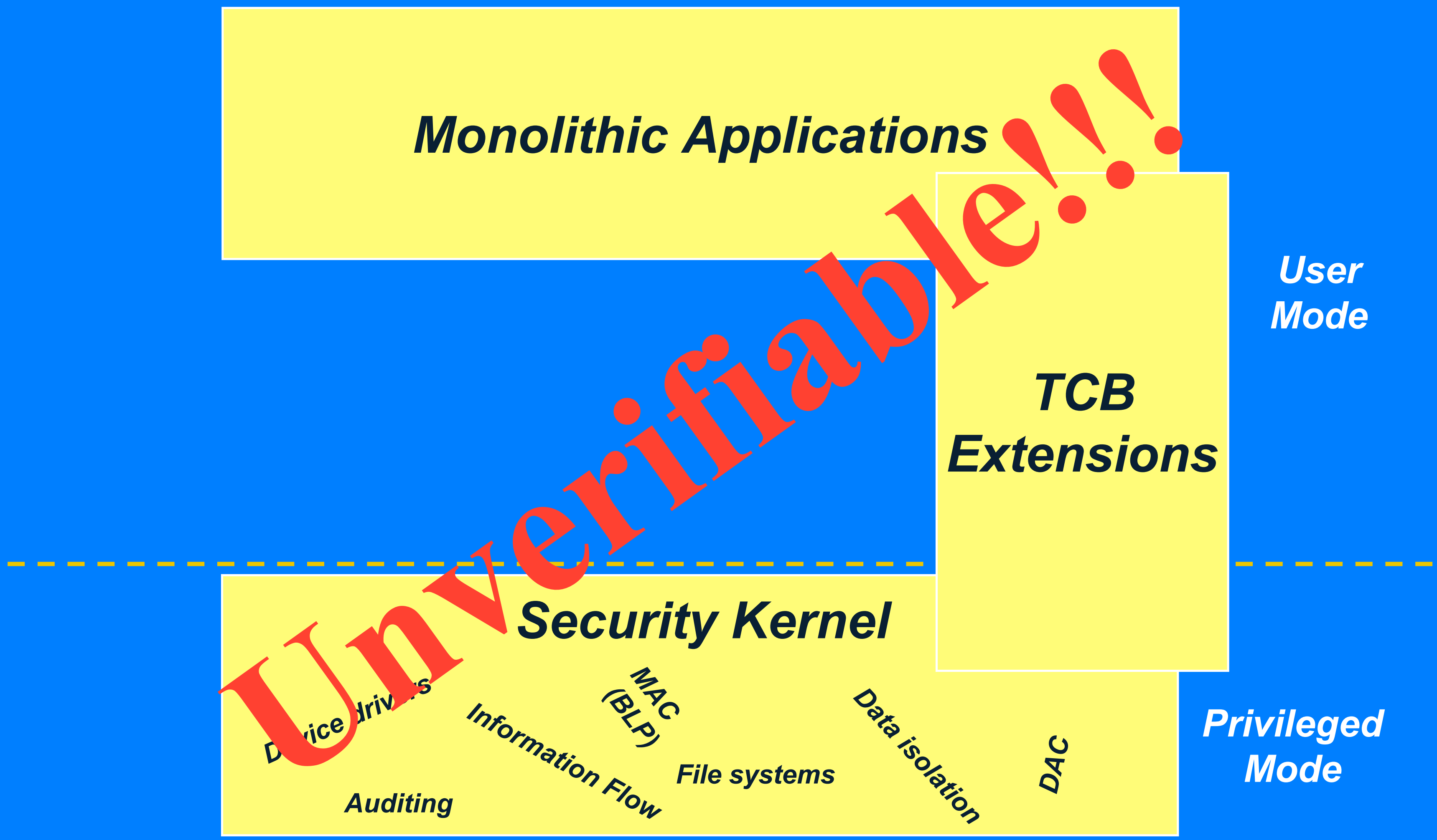
- Interest in the separation kernel concept has been renewed by advancements in processor performance.
  - Needed for safety- and security-critical apps & critical infrastructure
- The Separation Kernel is the foundation for the MILS architecture and must meet the highest standards in:
  - FAA DO-178B Level A Safety Technology (conservative)
  - Common Criteria EAL 7 Security Technology (progressive)
- SK's security policy is ***data isolation*** and ***information flow***
  - Small: ~ 4K LOC
  - SK simple enough to analyze, non-bypassable, tamper-proof
- All other OS and Middleware services and applications to reside in user mode
  - Leverage SK guarantees to enable "application" layers to enforce, manage & control their own policies
  - Implement reference monitors for higher level policies that are simple enough to analyze, non-bypassable, tamper-proof

MILS Assurance in a Nutshell

Dramatically decrease the amount of security critical code.

Dramatically increase the scrutiny of security critical code.

Security Kernel / TCB Approach



MILS Architecture Approach