OS Research

- What is OS Research?
  - Research on the base abstractions for computation provided to programmers
  - Research on the structure and facilities of the OS.
- OS Research from 1970 is insufficient today.
  - OS has adapted to some hardware changes (SMP, Net, etc.)
  - OS has not adapted to software changes:
    - 1970 Unix Application vs. Windows Application.
  - OS has not adapted to security changes.
    - Web-based code, faulty device drivers, utility computing, etc.
- Processes, threads, file systems, etc. are probably still relevant.

An OS Research Agenda

1. Adopt an abstract instruction set (MSIL) as the system binary interface.
2. Add OS abstractions for Ubiquitous Metadata
3. Rebuild the OS security model from top to bottom.
4. Reengineer the OS Development Process
5. Reexamine other OS areas changed by 1-4.

Abstract Instruction Set

- Observation:
  - Role of OS is to abstract (virtualize, idealize, and structure) hardware.
    - Address Space (MMU) → Virtual Memory,
    - Disk →File System,
    - CPU Isolation → Process,
    - CPU Cycles → Threads
  - Abstract CPU Instruction Set have been used *only* for CPU architecture neutrality.
  - There is a lot of duplication of effort between the CLR and OS
    - Memory Management, Threading, Synchronization, Security, Process Model, etc.

Abstract Instruction Set

- Research Proposal:
  - Use an abstract instruction set with the following properties as the universal OS Binary Interface:
    - Language neutrality
    - CPU architecture neutrality
    - Metadata and Well-formed
  - Use MSIL for Kernel, Driver, and Application code.
- Benefits:
  - CPU Architecture neutrality.
  - Generate qualitatively different code from a single MSIL binary.
    - Ex. Compile for higher security vs. higher performance:
      - buffer overflows, stack walking, no-execute on data, etc.
    - Ex. Enable ubiquitous instrumentation for profiling, debugging, etc.
  - Safely inline OS privileged operations into non-OS code.

Ubiquitous Metadata

- Observation:
  - Persisting and managing metadata is a powerful primitive.
    - Java: *Reflection* is a powerful programming facility.
      - Marshaling, debugging, interpreters, etc.
    - CLR: *User extensible reflection* adds more power.
      - Custom marshalling, thread access, data management, transactions, programmer annotations, runtime extensions, etc.
    - WinFS adds metadata to the file system for the shell.
  - Storage is a lot cheaper than it was 30 years ago,
    - OS ***can*** afford to save and process metadata.

Ubiquitous Metadata

- Research Proposal:
  - Make metadata a first class OS feature.
  - Extend to cover all named OS entities.
    - Processes, Threads, Handles, Files, Identities, etc.
  - Add new OS abstractions and to bridge metadata gaps.
    - Ex. (Program → Process) like (Class → Instance)
- Benefits:
  - Move fractured and often brittle metadata systems into protected first-class OS facility.
  - Persistence and APIs for metadata for /proc, IOCTLs, ACPI, PCI, PnP, CPU features, etc.
  - Apply verification and optimization to OS ***and applications***.
    - Dependency tracking, versioning and patching through protected first-class OS facility (get rid of DLL hell and its kin).

Rebuild the OS Security Model

- Observations:
  - Current security models don’t map to usage scenarios like:
    - Web-based code
    - Utility computing (division of resources, leasing, privacy)
  - Security is incredibly hard to manage.
  - OS has huge liability because it includes drivers in the TCB
    - Drivers are the #1 source of Windows blue screens.
    - Many drivers are written by first-time developers.
    - OS extension model forces many application into device drivers.
  - DRM is a fact of life
    - NGSCB is a engineering solution to a research problem.

Rebuild the OS Security Model

- Proposal:
  - Build a new OS security model top to bottom.
    - Remove “root” and ”Administrator”.
    - Create an OS extension model that assumes drivers may be not just buggy, but malicious.
      - Drivers are no longer trusted components.
      - Could use ASI to run untrusted drivers in Ring 0.
      - Restructure OS to make driver trust relationships explicit.
      - Make hardware changes (DMA “MMU”) to ensure safety.
    - Explore OS extension mechanisms (see Windows “Drivers”)
- Benefits:
  - Robust system in face of driver failures.
  - Lower aggregate IQ required for driver development.

OS Development Process

- Observation:
  - Unix and C were shaped through synergistic co-evolution.
  - Static analysis techniques such model checking have advanced dramatically in the last decade.
- Research Proposal:
  - NP: Explore co-evolution of OS and tools.
  - Structure OS code to meet tools half way.
  - Push tools research into the design phase.
- Benefits:
  - Optimization: Generate fastpaths from single code base.
  - Correctness:
    - Remove deadlocks and ambiguous behavior.
    - Validate JIT and GC transformations.
  - Security: Verify the TCB.

Re-examine Other OS Areas

- Should OS use hardware enforced user/kernel mode boundary?
- Should every application still have its own address space?
- Can OS move to more cooperative scheduling mechanisms enforced by the generated native code?
- What traditional OS APIs can unified, replaced, or simplified by the Metadata APIs?
- What static analysis tools can be applied to, say configuration analysis, on an installed OS + applications because of Metadata?
- Given ASI and Metadata, what OS features should change given HW trends?
  - CPU cores are proliferating.
  - Memory and disk are incredibly plentiful, but latency is not.
  - Cache hierarchies are deep and transistors are available.

Singularity: A Research OS Platform

- Target scenario: Digital-convergence devices
  - Home with smart remotes, set top boxes, and media servers.
  - Trusted open software platform.
  - Zero on-site human administration.