**RELIABILITY**

**Instruction Counter Heartbeat (ICH)**
- Uses performance counters, present in most modern microprocessors.
- When a process completes a fixed number of instructions, it updates a counter.
- The counter is periodically checked by a hardware module embedded along with the processor.

**Issues:**
- Context Switch, Cache Misses, Networking, I/O events – take non-deterministic time
- Visibility at the hardware level through other routines
- The timers can be disabled to ignore these events

**Advantages:**
- Application need not be instrumented
- Leverages performance counters, already present in current CPUs, and information about process state and I/O events
- Performance overhead is minimal since software-level OS interaction is eliminated
- Minimal hardware required – timers and mechanism to update counters.

**Process Hang Detection**
- Loop Hang Detection Module
  - Detects application hang by tracking entry and exit points of a loop
  - Timer is started when loop is entered
  - Waits for Loop exit signal
  - Timeout value estimated by profiling of application
- Sequential Code Hang Detection Module
  - Detects application hang by looking for a repetition of instructions while application is executing sequential code (within a basic block)
  - Maintains a log of previously committed instructions
  - Parameterized at the beginning of a block with the maximum length of repeated sequence to look for
  - Repetition in log indicates illegitimate loop, possibly leading to hang

**Application-Level Dependency extraction**

**Checkpointing & Recovery: Rendezvous of Security and Reliability**
- **Recovery**
  - Common concern for Security and Reliability
  - When the security of a system is compromised it can be crashed or restored to a safe state
  - When a system behaves unreliably due to failures need to bring it back to a consistent checkpoint state
- **Data Dependency Tracker**
  - Uses a copy-on-write mechanism to save checkpoints of memory pages when an error is detected.
  - On the crash of a thread due to a security attack or a failure two methods of recovery are used:
    - Crash of the thread (1)
      - No easy way to coordinate inter-thread applications
    - Crash of all threads depending on the crashing thread
      - Dependent threads are threads which consume output of the crashed thread
  - Rollbacks the crashed thread and its dependent threads to a consistent and safe point in execution

**Future Directions**
- Synthesize framework with DLX
  - Integrate reliability and security
- Fault-injection based assessment of techniques
- Dynamic Reconfiguration
  - Insert assertions
  - Compiler Support

**MEMORY LAYOUT RANDOMIZATION**
- Randomly relocates position independent regions across stack, heap
- Can mask a broader range of vulnerabilities due to level programming errors, e.g., buffer overflow, format string, signed integer overflow
- MSR modules implement:
  - the executable header parsing, address computation algorithm and random relocation of the stack and heap in hardware
  - Runtime location of stack and heap calculated from information embedded in the header of the executable file
- Allocation of position dependent regions
- Global Offset Table with OS Support
- Can be potentially used by different operating systems while the previous software implementation is always system specific

**Data Dependency Tracker Module**
- In multi-threaded applications, on a single thread crash the traditional all-thread approach leads to heavy losses.
- Execution of healthy threads should be kept intact in such a case.
- Data Dependency Tracker (DDT)Module uses a page-based memory dependency tracking mechanism to track dependencies between threads.
- DDT tracks memory accesses and uses a copy-on-write mechanism for memory pages to checkpoint the memory state.
- All threads dependent on faulty thread (either malicious or non-malicious) are killed
- Recovery uses two algorithms:
  - Without execution rollback
  - With execution rollback

**Secure Return Address Stack**
- Detects stack-based buffer overflow attacks when a malicious over-read return address is being used for returning to the invoking function
- Prevents the attacker from successfully seizing control of the targeted application program
- Secure Return Address Stack (SRAS) checking done in parallel with the processor’s main pipeline – Value, the checking latency
- Typical overhead of the proposed solution is of the order of 0.1%
- Compared to 24% for software-based approach shown in our previous study

**Security**

**Data Dependency Tracker Module**
- In multi-threaded applications, on a single thread crash the traditional all-thread approach leads to heavy losses.
- Execution of healthy threads should be kept intact in such a case.
- Data Dependency Tracker (DDT)Module uses a page-based memory dependency tracking mechanism to track dependencies between threads.
- DDT tracks memory accesses and uses a copy-on-write mechanism for memory pages to checkpoint the memory state.
- All threads dependent on faulty thread (either malicious or non-malicious) are killed
- Recovery uses two algorithms:
  - Without execution rollback
  - With execution rollback

**Future Directions**
- Synthesize framework with DLX
  - Integrate reliability and security
- Fault-injection based assessment of techniques
- Dynamic Reconfiguration
  - Insert assertions
  - Compiler Support

**Reliability and Security Engine (RSE)**

**Application Aware Checking – Reliability and Security Engine**

Nithin M. Nakka, Zbigniew T. Kalbaczyk, Ravi K. Iyer

www.itr.uiuc.edu