

Non-bypassable Kernel Services for Execution Security



Christopher D. Gill (PI) and William D. Smart (co-PI), Washington University in St. Louis
Douglas Niehaus (PI), University of Kansas

NSF Grant CNS-0716764

www.cse.wustl.edu/~cdgill/

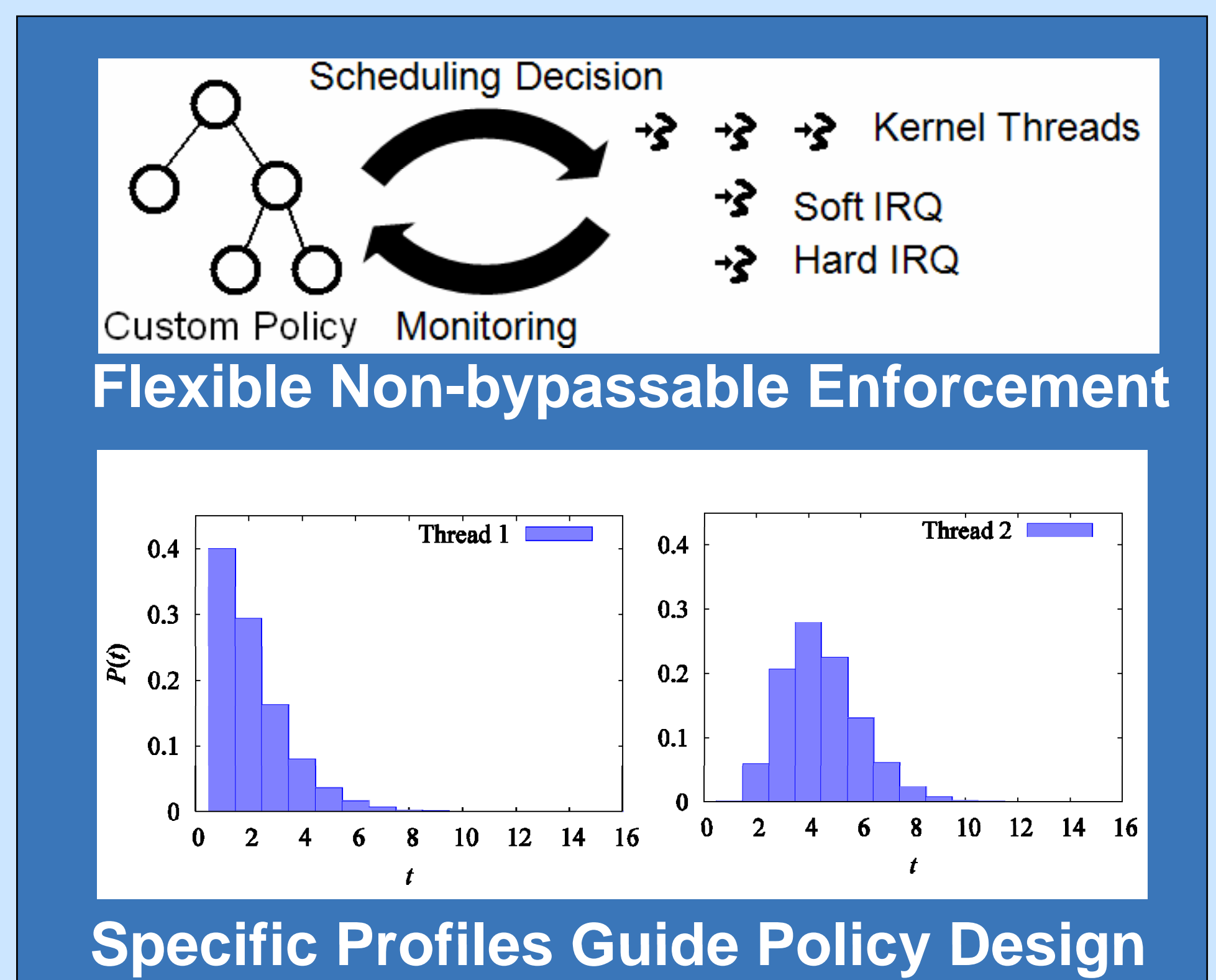
Research Objectives, Approach and Results

The goal of this research is to develop infrastructure and scheduling and verification techniques to enforce non-bypassable security of systems' execution constraints within common off-the-shelf platforms.

We are developing detailed models of the interactions among computations and the OS features they use, to provide at once:

1. precise enforcement of execution isolation
2. flexible system-specific isolation policies
3. leverage for learning in uncertain settings

Our approach targets common off-the-shelf operating systems (i.e., Linux) and important interaction scenarios (e.g., CPU contention and interrupt handling) relevant to many real-world systems.



Approach and Impact

New approach

- Schedule all relevant components under a single custom kernel policy
- Use decision models / learning to design the custom scheduling policy
- Exploit repeated structure in the resulting system state space

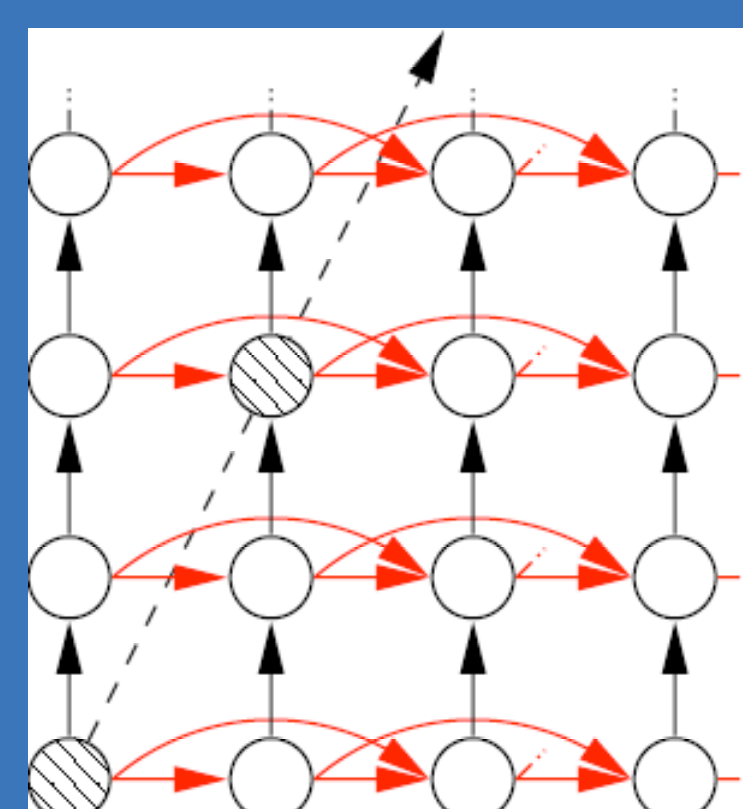
Research Impact

- Achieves precise isolation without requiring static partitioning
- Discovers and refines *appropriate* scheduling decision functions
- Reduces cost for both verification (model checking) and learning

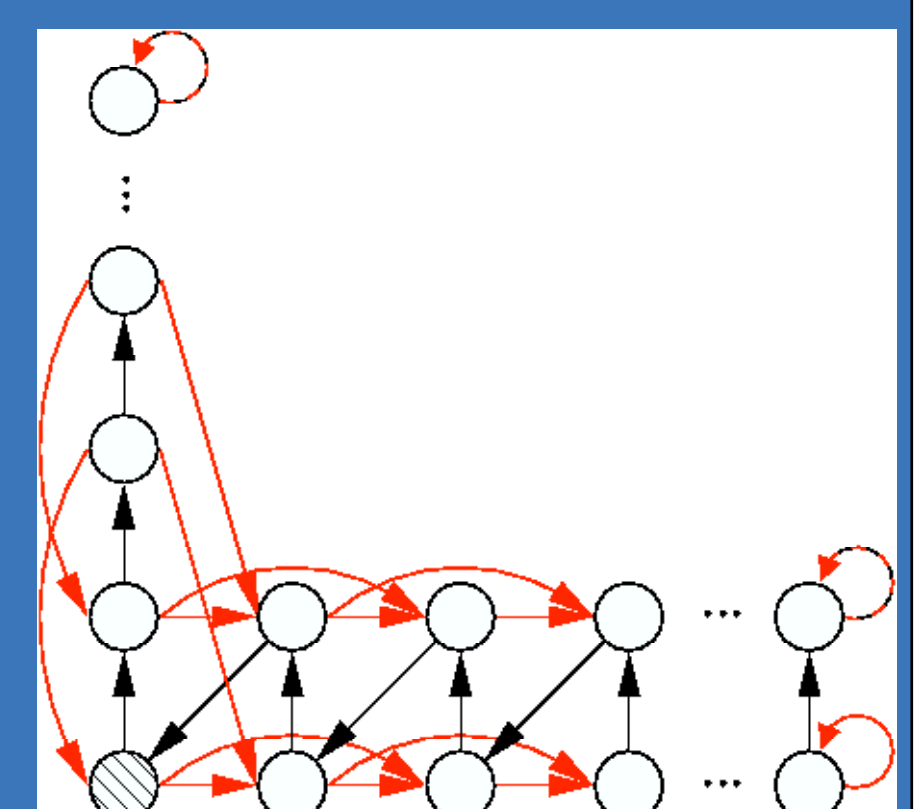
Scheduling policy design method:

1. Profile relevant component behaviors (e.g., distributions of durations of atomic actions),
2. Evaluate a decision model that encodes the interactions among system components
 - E.g., a MDP parameterized with profiles
 - Produces a scheduling decision function
3. Apply quasi-cyclic reductions to reduce and bound the resulting system state space
4. Apply (potentially timed) model checking to verify properties of the reduced state space
 - Within bounds, out to a finite horizon, etc.
5. Refine scheduling policy (and repeat 3-5)
 - E.g., add constraints to promote verification
 - E.g., use RL: unknown profiles, interactions

Scheduling Policy Design Example



Markov model provides initial policy (CPU share out to a finite horizon)



Quasi-cyclic structure bounds state space (use model checking to verify, RL to tackle uncertainty)