

# RESEARCH STATEMENT

Francis M. David (fdavid@uiuc.edu)

My research focuses on the dependability of software that drives computers. I'm especially interested in the design of systems software to achieve high reliability and security. These aspects are constantly growing in importance, fueled by the increased ubiquity of computers in our daily lives.

## Background and Current Research

One of the approaches typically adopted in order to improve operating system (OS) reliability is the development of techniques to detect and recover from a wide variety of errors caused by both hardware and software faults. Designs such as the Minix and L4 microkernels, which componentize the OS and enforce inter-component isolation are able to provide improved reliability by limiting error propagation. Individual OS services can be restarted after a failure in order to keep the system running. While this restart-based recovery works well for services that are stateless, it does not handle stateful services very well. OS state that is lost because of a restarted service can result in cascading failures in other components and in user applications. A naïve solution that just preserves state across a restart does not solve the problem of intra-component error propagation that may have resulted in irrecoverable corruption of the saved state.

My thesis introduces CuriOS, a new operating system that restructures OS service state in order to minimize intra-component error propagation and to allow state persistence across service restarts [3]. This is accomplished by lightweight distribution, isolation and persistence of application-specific state information used by OS services. A microkernel OS component providing a service is also commonly referred to as a server. Application-specific state is stored in application-associated, but application-inaccessible memory called Server State Regions (SSRs) and servers are only granted access to this information when processing a request. This prevents errors in servers from affecting state related to all applications. Because SSRs are not associated with the server, servers can be transparently restarted without loss of this state. This distribution of state is illustrated in figure 1. CuriOS provides a state management framework that can be used by OS services to allocate and manage SSRs. SSR access control is lightweight and is implemented using virtual memory maps. Many traditional OS services can be written to use SSRs for storing application-related state. Fault-injection experiments with several OS services show that it is possible to recover from 87-100% of all manifested errors.

CuriOS incorporates several other novel research ideas that were developed during the course of my PhD research. It includes support for a unified exception handling framework that allows developers to use similar constructs to handle both software and hardware exceptions [5]. Self-healing properties are achieved through the isolation and micro-rebooting of system components [2]. Hardware watchdog timers can be used to detect

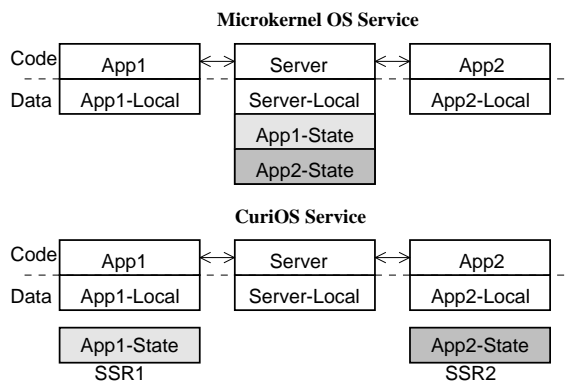


Figure 1: State Distribution

lockup errors within the OS. We have demonstrated that it is possible to recover from processor resets generated by such hardware in both CuriOS and Linux [4]. This exploits the fact that, on many systems, main memory contents are preserved across a processor reset and can be used for recovery.

I am also actively involved in several projects exploring OS security. Cloaker is a sophisticated rootkit that illustrates the weaknesses of existing malware detection techniques [1]. It survives by compromising the integrity of hardware state and thus evades detection by tools that only ensure integrity of software state. We advocate non-generic hardware specific checks to detect and neutralize such threats.

We are currently investigating low-level memory analysis techniques to automatically identify program structures. This has applications in post-intrusion forensic analysis and data structure recovery. We are also studying a potential attack against computer systems that exploits the fact that RAM contents are not cleared after a processor reset. While we have investigated using this feature to recover the system and improve reliability [4], this same technique may allow an attacker to access sensitive information in memory. We expect to publish our research on these topics soon.

## Future Directions

I have always been interested in exploring innovative and exciting directions in systems research. While I am flexible and willing to adapt to business requirements in an industrial research environment, I do have several items on my research agenda that I would like to pursue if given the opportunity.

**Reliability:** We have seen a rapid growth of parallelism in processors as a result of the move towards multi-core architectures. With a large number of threads expected to be available on processors in the future, I see great potential in exploiting this parallelism to provide significant improvements in reliability. I believe that it would be possible to re-create expensive and highly reliable systems like the Tandem computers on inexpensive modern desktops with multiple cores. The state separation approach advocated in my thesis can also exploit parallel threads to potentially speed up OS services. If a request cannot be processed by an OS server on one core, it may be dispatched to a duplicate server on another core. This is possible for stateless servers since any required state is dispatched along with the request.

I would also like to work further on techniques for evaluating the reliability of system software. There is a need for a freely available tool that can be used to evaluate system software through automatic injection of complex faults such as software bugs. We have built a simple tool that uses the standardized gdb protocol to inject arbitrary faults into virtual machine environments. But a significant number of open research questions still remain that would need to be addressed in order to perform meaningful fault injection experiments.

**Security:** I would like to continue exploring the boundary between system software and architecture and investigate security failures that occur because architectural features are not carefully considered in the design of techniques that protect system software. The Cloaker rootkit work illustrates one such vulnerability. Uncleared RAM after a reset presents yet another similar vulnerability that can be exploited. I am working in close collaboration with several researchers in order to identify and fix such vulnerabilities in current systems.

## References

- [1] **Francis M. David**, Ellick M. Chan, Jeffrey C. Carlyle, Roy H. Campbell, "Cloaker: Hardware Supported Rootkit Concealment", *IEEE Symposium on Security and Privacy*, May, 2008.
- [2] **Francis M. David**, Roy H. Campbell, "Building a Self-Healing Operating System", *IEEE International Symposium on Dependable, Autonomic and Secure Computing*, September, 2007
- [3] **Francis M. David**, Jeffrey C. Carlyle, Ellick M. Chan, Philip A. Reames, Roy H. Campbell, "Improving Dependability by Revisiting Operating System Design", *Workshop on Hot Topics in Dependability*, June, 2007
- [4] **Francis M. David**, Jeffrey C. Carlyle and Roy H. Campbell, "Exploring Recovery from Operating System Lockups", *USENIX Annual Technical Conference*, June, 2007
- [5] **Francis M. David**, Jeffrey C. Carlyle, Ellick M. Chan, David K. Raila and Roy H. Campbell, "Exception Handling in the Choices Operating System", *Advanced Topics in Exception Handling Techniques*, LNCS, Springer-Verlag Inc, 2006