

Nitro Isolation Engine: Formally Verifying a Production Hypervisor

Hanno Becker ✉ 

Automated Reasoning Group,
Amazon Web Services, Cambridge, UK

Robert Dockins ✉ 

Automated Reasoning Group,
Amazon Web Services, Portland, OR, USA

Jason Hu ✉ 

Automated Reasoning Group,
Amazon Web Services, Seattle, WA, USA

Dominic P. Mulligan ✉ 

Automated Reasoning Group,
Amazon Web Services, Cambridge, UK

Bryan Parno ✉ 

Automated Reasoning Group,
Amazon Web Services, PA, USA

Konrad Slind¹ ✉ 

Unaffiliated, Minneapolis, MN, USA

Nathan Chong ✉ 

Automated Reasoning Group,
Amazon Web Services, FL, USA

Jim Grundy ✉ 

Automated Reasoning Group,
Amazon Web Services, Austin, TX, USA

Ike Mulder ✉ 

Automated Reasoning Group,
Amazon Web Services, Cambridge, UK

Paul Mure¹ ✉ 

Massachusetts Institute of Technology,
Cambridge, MA, USA

Lawrence C. Paulson ✉ 

Automated Reasoning Group,
Amazon Web Services, Cambridge, UK

Abstract

Cloud computing relies on hypervisors to enforce isolation between co-tenanted virtual machines. Hypervisors are therefore critical security infrastructure, and assurance of their correctness is paramount. Traditional engineering techniques – code review, testing, fuzzing – provide strong assurance but cannot exhaustively verify that isolation holds across all possible execution paths. Formal verification extends and complements these approaches by establishing mathematical guarantees about system behaviour.

This talk presents our experience applying interactive theorem proving to verify a production hypervisor component: the Nitro Isolation Engine. This is a trusted, minimalist computing base written in Rust, enforcing isolation between virtual machines on AWS Graviton5 EC2 instances. Designed for verification from inception, we have specified the intended behaviour of this component and verified correctness in the Isabelle/HOL interactive theorem prover, producing approximately 330,000 lines of machine-checked models and proofs, and establishing three key classes of property:

1. **Functional correctness:** The system behaves as specified for all operations including virtual machine creation, memory mapping, and abort handling. Our total verification approach additionally establishes memory-safety, termination, and absence of runtime errors.
2. **Confidentiality:** A noninterference-style property demonstrates that guest virtual machine state remains hidden from an expansive definition of observer monitoring system actions, formalised as indistinguishability preservation up to permitted declassification flows.
3. **Integrity:** Guest virtual machine private state is unaffected by operations on distinct virtual machines.

Currently, our proof coverage extends to verification of the core virtual machine-management hypercalls, guest power management, various utility hypercalls, and a subset of data, instruction, and asynchronous abort handling, and will continue to expand to cover more functionality including PCI device management and virtual GIC (Generic Interrupt Controller) handling.

¹ work done whilst at Amazon Web Services



1:2 Nitro Isolation Engine

The talk will discuss the verification approach, key proof techniques, and challenges in applying formal methods to production systems. Note that this work builds on decades of academic research across interactive theorem proving, formal specification, separation logic and its automation, and programming language semantics.

2012 ACM Subject Classification Security and privacy; Security and privacy → Formal methods and theory of security; Security and privacy → Logic and verification

Keywords and phrases Isabelle/HOL, Rust, verification, separation logic, hypervisors

Digital Object Identifier 10.4230/LIPIcs.ITP.2026.1

Category Invited Talk