ChaosBear: A Framework for Inducing Systematic Failures in Software Defined Networks

Ali Mir

Senior Capstone Abstract

Data centers are complex entities running a fusion of traditional protocols and custom protocol uniquely designed for propriety workloads. Modern efforts to employ verification, emulator-based validation, and traditional software engineering tests fail to enable a significant class of bugs that create infrastructure downtime.

In this paper, we introduce ChaosBear, a simple, scalable approach to systematically fuzzing large production data centers to efficiently identify failures with minimal impact to production traffic. To minimize the number of tests, ChaosBear introduces data-center specific fuzz-algorithms that leverage symmetry inherent in ports of the data center to minimize the fuzz-space, and when available, leverages program analysis to drive fuzzing-efforts towards poorly tested paths.

Although solutions exist for testing compute and storage, testing of networking infrastructure presents a markedly distinct problem. Unlike the compute and storage, where-in fuzzing can be truly random because (1) there is sufficient redundancy in all component to enable recovery, and (2) the network provides the underlying connectivity required to bootstrap recovery efforts. However, with the network, poor-fuzzing logic could result in irrevocable partitions and create scenarios that the network control plane software is unable to recover from. Existing applications of the 'Chaos Principle' to networking focus on SDN where logic is centralized thus simplifying design and implementation.

In this work, we focus on design for modern data centers, while inspired by SDN-principles that leverage BGP-based dataplanes that are both stateful and distinct form the SDN controller. This difference necessitates novel design choices to efficiently coordinate fuzz across a distributed infrastructure while effectively hiding the ramifications of testing from the end-users. To enable efficiency, we introduce novel fuzzing algorithms inspired by data center topologies. Essentially, we leverage the

symmetry abundant in the higher levels of data-centers and operation knowledge about failure probabilities to design provable efficient heuristics.