# Summary of

# “*Gauntlet* “

* Short Definition: Gauntlet, a tool for finding bugs in packet-processing compilers for languages such as P4. Gauntlet combines random program generation, translation validation, and model-based testing to find both crash and semantic bugs in P4 compilers.

They search for both abnormal termination of the compiler (Crash bugs) and miscompilation (semantic bugs). They use 3 key ideas: Random Program generation, Translation Validation, and Model-based testing.

* Random Program generation:

“Random program generation produce syntactically correct and well-typed P4 programs that still induce P4 compiler crashes. Because P4 has very little undefined behavior, random program generation is considerably simpler for P4 than for C. The generator does not have to painstakingly avoid generating programs

with undefined and unspecified behavior, which can be interpreted differently across different compilers. The smaller and simpler grammar of P4 relative to C also simplifies the development of a random program generator.”

* Translation Validation:

“Translation validation finds miscompilations in P4 compilers in which we can access the transformed program after every compiler pass. Translation validation has been used in the context of C compilers before, but has suffered one of two shortcomings. It either needs considerable manual effort per compiler pass or suffers from a small rate of false positives and false negatives. Fundamentally, this is inevitable for unrestricted C: proving program equivalence in the presence of unbounded loops is undecidable. In this case, however, the finite nature of P4 makes P4 program equivalence decidable and addresses both shortcomings. Thus, our use of translation validation is both precise and fully automated, requiring manual effort only to develop semantics for the P4 language—not manual effort per compiler pass.”

* Model-based Testing

“Model-based testing generates input-output test packets for P4 programs based on the semantics they had to develop for translation validation. They use these test packet pairs to find miscompilations in black-box and proprietary P4 compilers where they cannot access the transformed program after every compiler pass. Testing for general-purpose languages is effective at generating inputs that provide sufficient path coverage by finding inputs satisfying path conditions. But without language semantics, determining the correct output for these test inputs is hard. By creating formal semantics for P4 for translation validation, they can generate both input and output test packets, which can then be used to test the implementation produced by the compiler for a P4 program.”