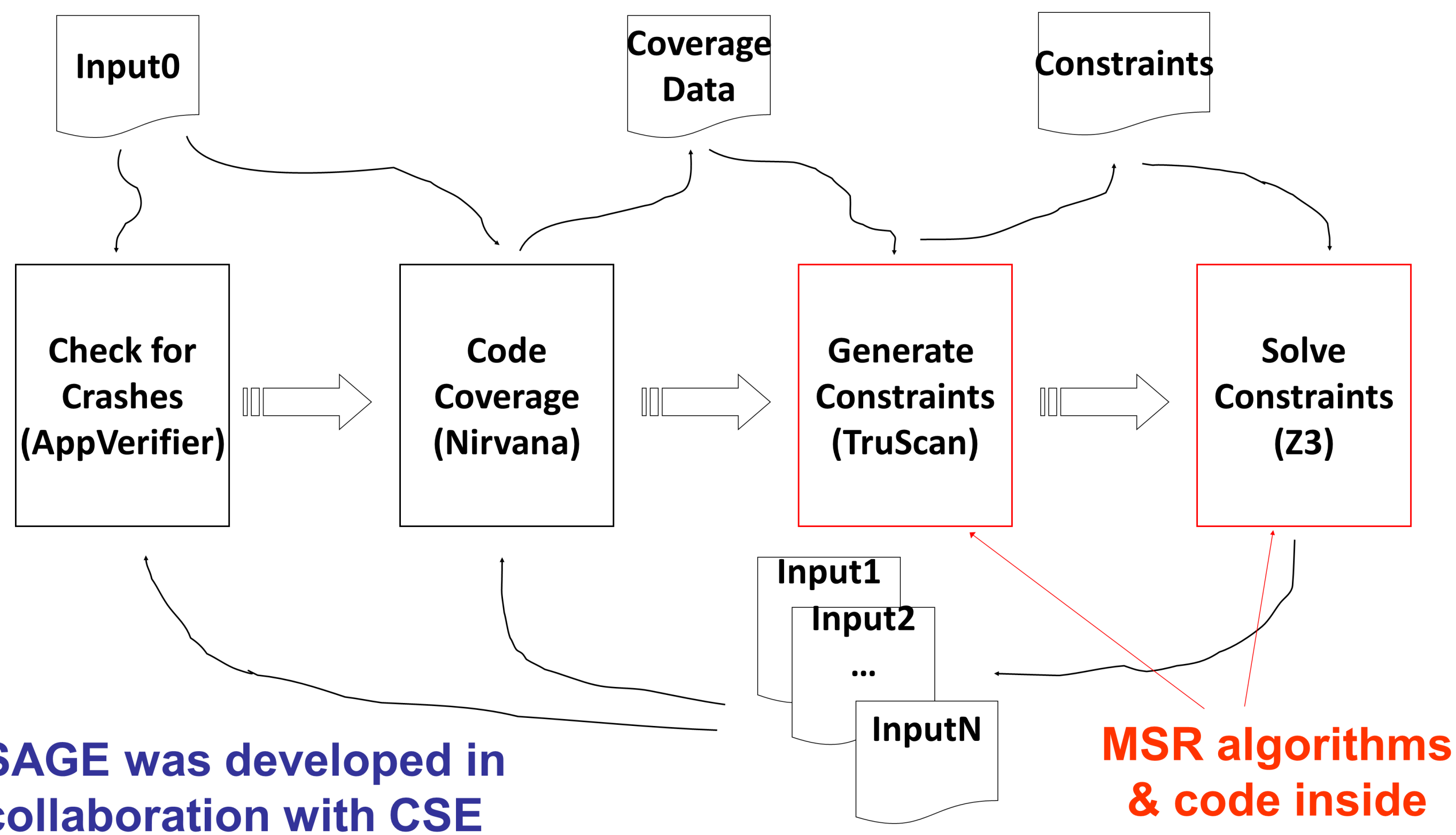




SAGE: Whitebox Fuzzing for Security Testing

Ella Bounimova Patrice Godefroid David Molnar

- Basic idea:**
1. Run the program with first inputs,
 2. gather constraints on inputs at conditional statements,
 3. use a constraint solver to generate new test inputs,
 4. repeat - possibly forever!



Research Challenges:

- How to recover from **imprecision** ? PLDI'05, PLDI'11
 - How to **scale** to billions of x86 instructions? NDSS'08
 - How to check **many properties** together? EMSOFT'08
 - How to leverage **grammar** specifications? PLDI'08
 - How to deal with **path explosion** ? POPL'07, TACAS'08
 - How to reason **precisely** about pointers? ISSTA'09
 - How to deal with **floating-point** instr.? ISSTA'10
 - How to deal with input-dependent **loops**? ISSTA'11
 - How to **synthesize x86 circuits** automatically? PLDI'12
 - How to **run 24/7/365** for months at a time? ICSE'2013
- + research on **constraint solvers**

Impact: since 2007

- 500+ machine years (in largest fuzzing lab in the world)
- 3.4 Billion+ constraints (largest SMT solver usage ever!)
- 100s of apps, 100s of bugs (missed by everything else...)
- Ex: **1/3** of **all** Win7 WEX security bugs found by SAGE →
- Bug fixes shipped quietly (no MSRCs) to 1 Billion+ PCs
- Millions of dollars saved (for Microsoft and the world)
- SAGE is now used daily in Windows, Office, etc.

