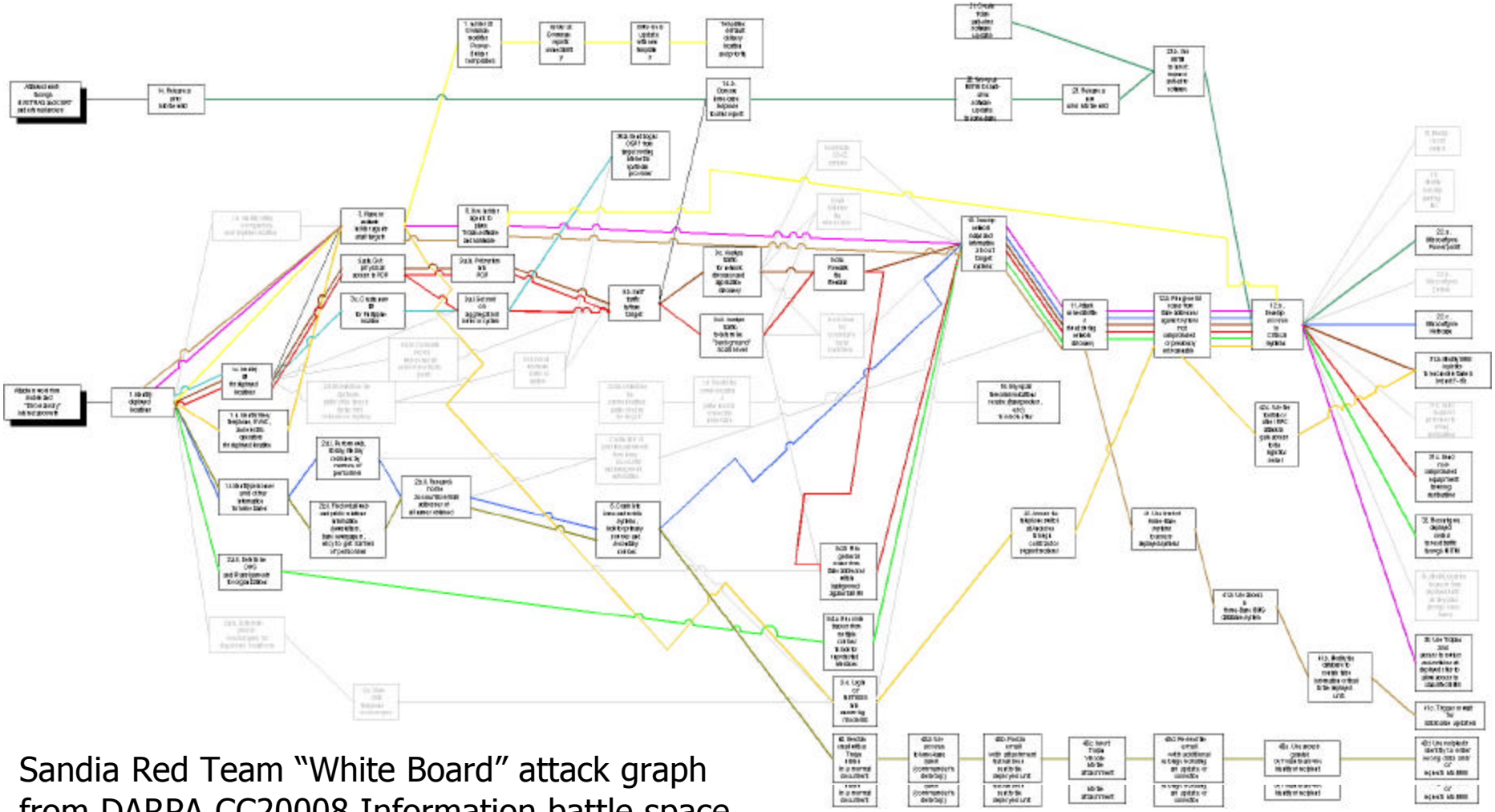
Generation of Scenario Graphs Using Model Checking

Somesh Jha (University of Wisconsin), Oleg Sheyner (CMU), Jeannette Wing (CMU)

Example of Attack Graph Developed by a Professional Red Team



Sandia Red Team "White Board" attack graph from DARPA CC20008 Information battle space preparation experiment

Drawn By Hand

Definitions

- Given
  - a finite state model M
  - a correctness property Φ
- An failure scenario is an execution of M that violates Φ.
- An scenario graph is a set of failure scenarios of M.

Properties of Scenario Graphs

- **Exhaustive**
  - All possible failure scenarios are represented in **G**.

- **Succinct**
  - Only relevant states are contained in **G**.
  - Only relevant transitions are contained in **G**.

Problem Statement

- **Problem:** Generating scenario graphs by hand is tedious, error-prone, and impractical for large systems.

- **Our Goal:** Automate the generation and analysis of scenario graphs.

  - Generation

    Must be fast and completely automatic

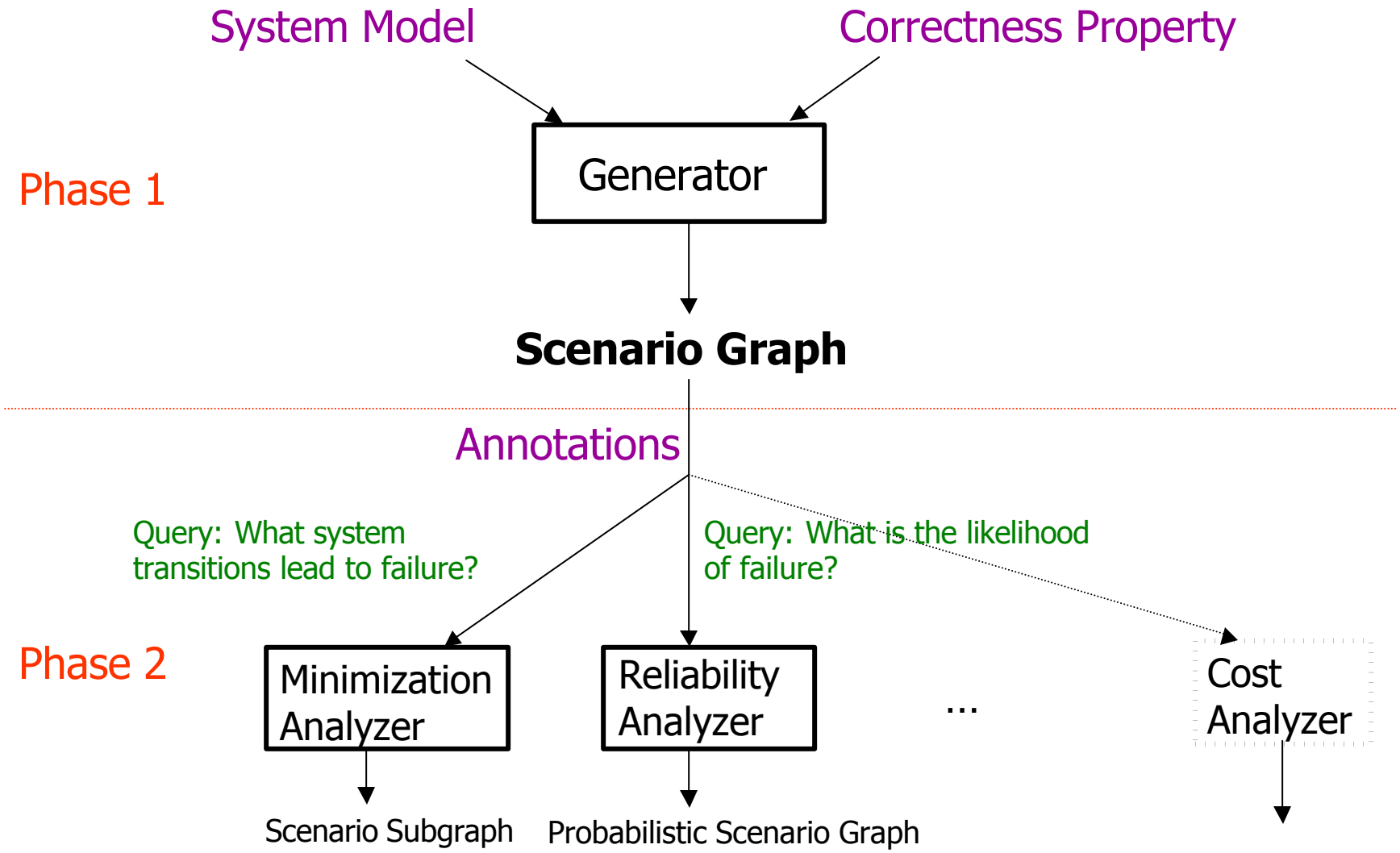
    Must handle large, realistic examples

    Should guarantee properties of scenario graphs

  - Analysis

    Enables tool-aided post-generation analysis

Overview of Our Method



Symbolic Scenario Graph Generation

- **Inputs**
  - $S, S_0 \subseteq S, R \subseteq S \times S$
  - F = AG (~unsafe) (a safety property)
- **Output**

  Scenario graph G = (S_{unsafe}, S_0^F, R^F)
- **Algorithm**

  1. S_{unsafe} = modelCheck(S, S_0, R, F)

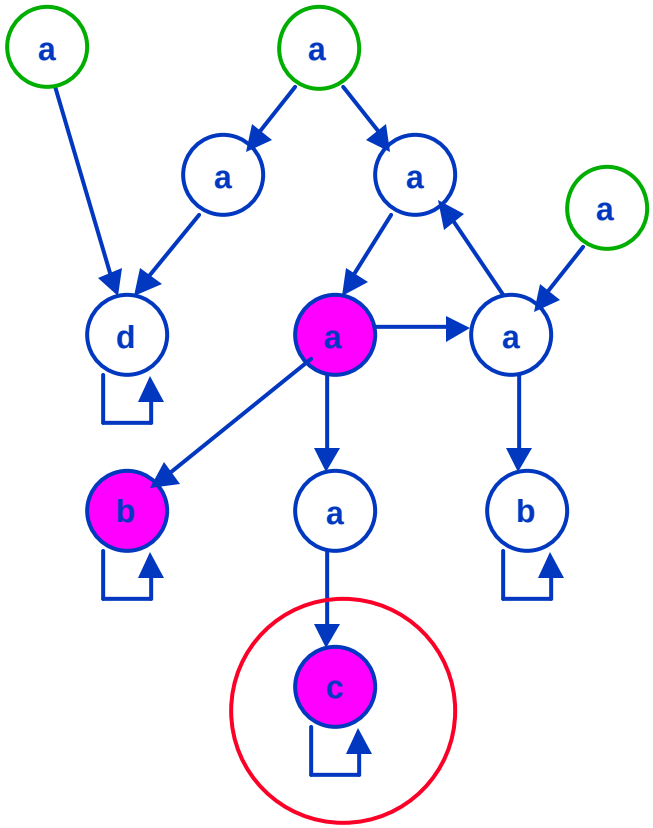
     (* Use an iterative algorithm derived from the fixpoint characterization of AG operator. *)
  2. $S_0^F = S_0 \cap S_{unsafe}$
  3. $R^F = R \cap (S_{unsafe} \times S_{unsafe})$

Explicit-State Scenario Graph Generation

- Based on Automata-Theoretic Model Checking
  - Interpret both model M and correctness property $\mathbb{F}$ as Buchi automata.
  - M and $\mathbb{F}$ induce languages L(M), L($\mathbb{F}$).
  - L(M)\L($\mathbb{F}$) = executions of M that violate $\mathbb{F}$.
  - Construct M $\cap$ ~$\mathbb{F}$ by computing intersection of Buchi automata.
- $\mathbb{F}$ can be any LTL property.

Explicit-State Algorithm Illustrated

Model M



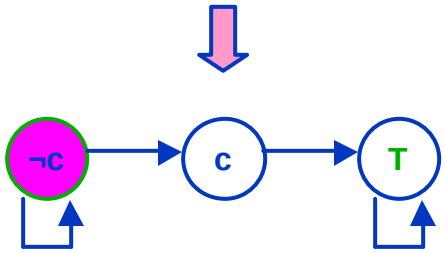
LTL Property Never c

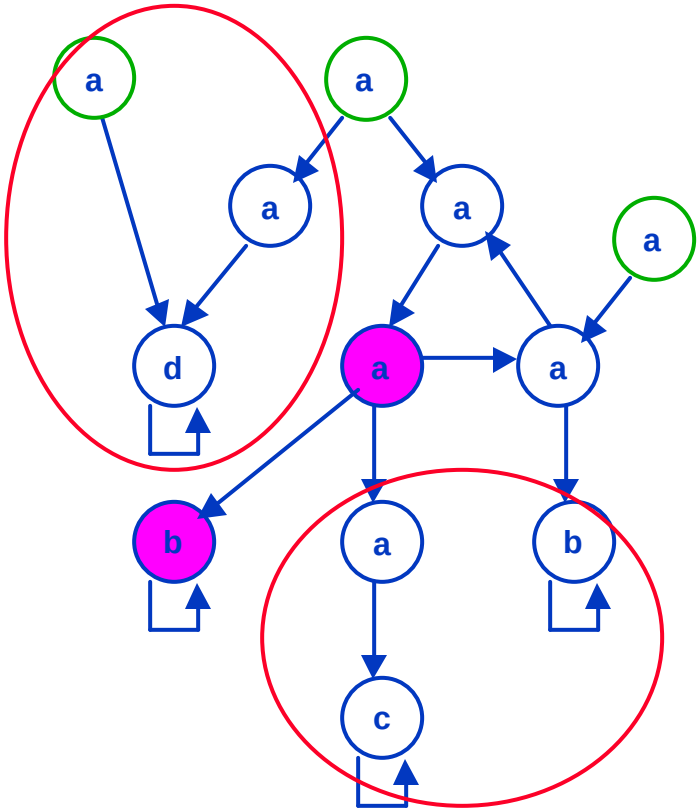
$\models \neg F c$

$\models F c$

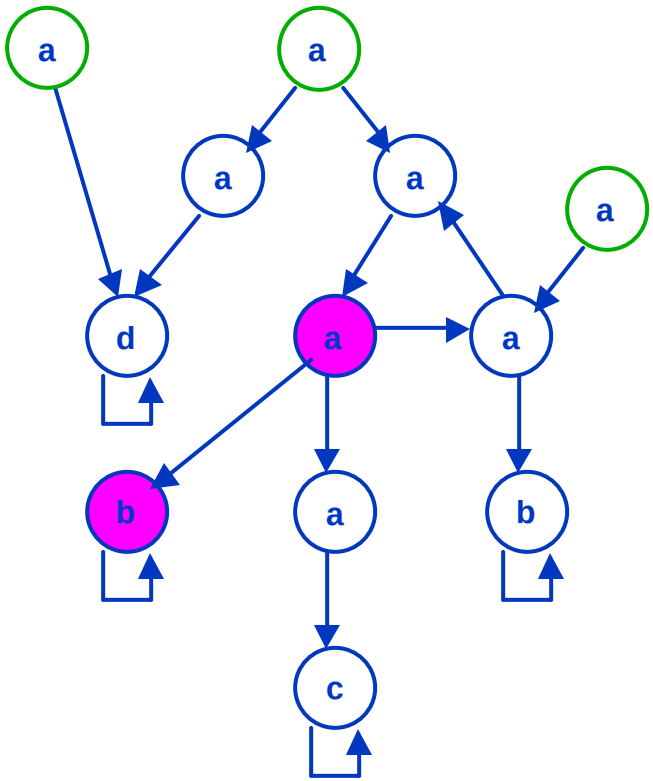
$\models G \neg c$

n





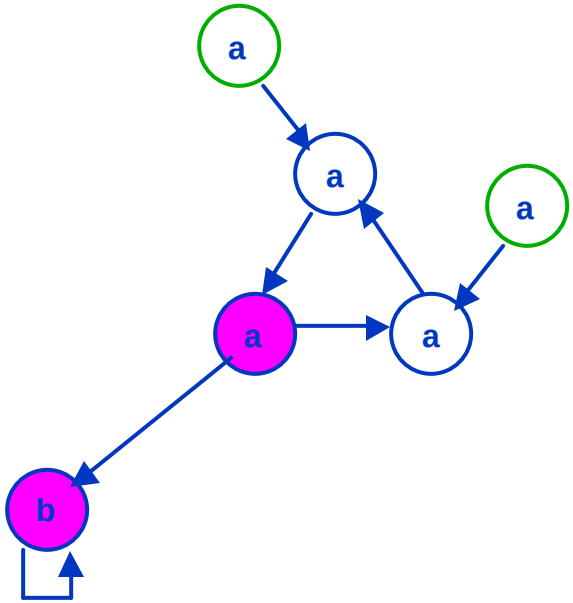
Explicit-State Algorithm (Cont.)



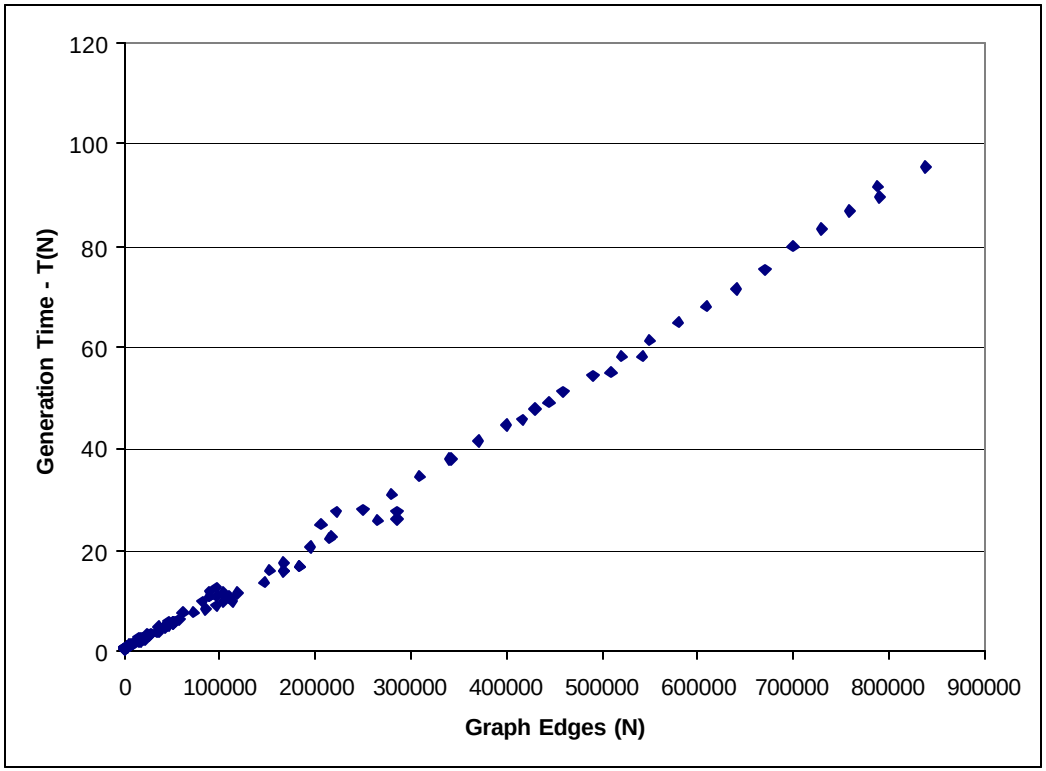
Find strongly connected components (SCCs) (R. Tarjan '72)

Collect SCCs with acceptance states

Add paths from initial states



Performance



Linear Regression R^2 = 0.9967

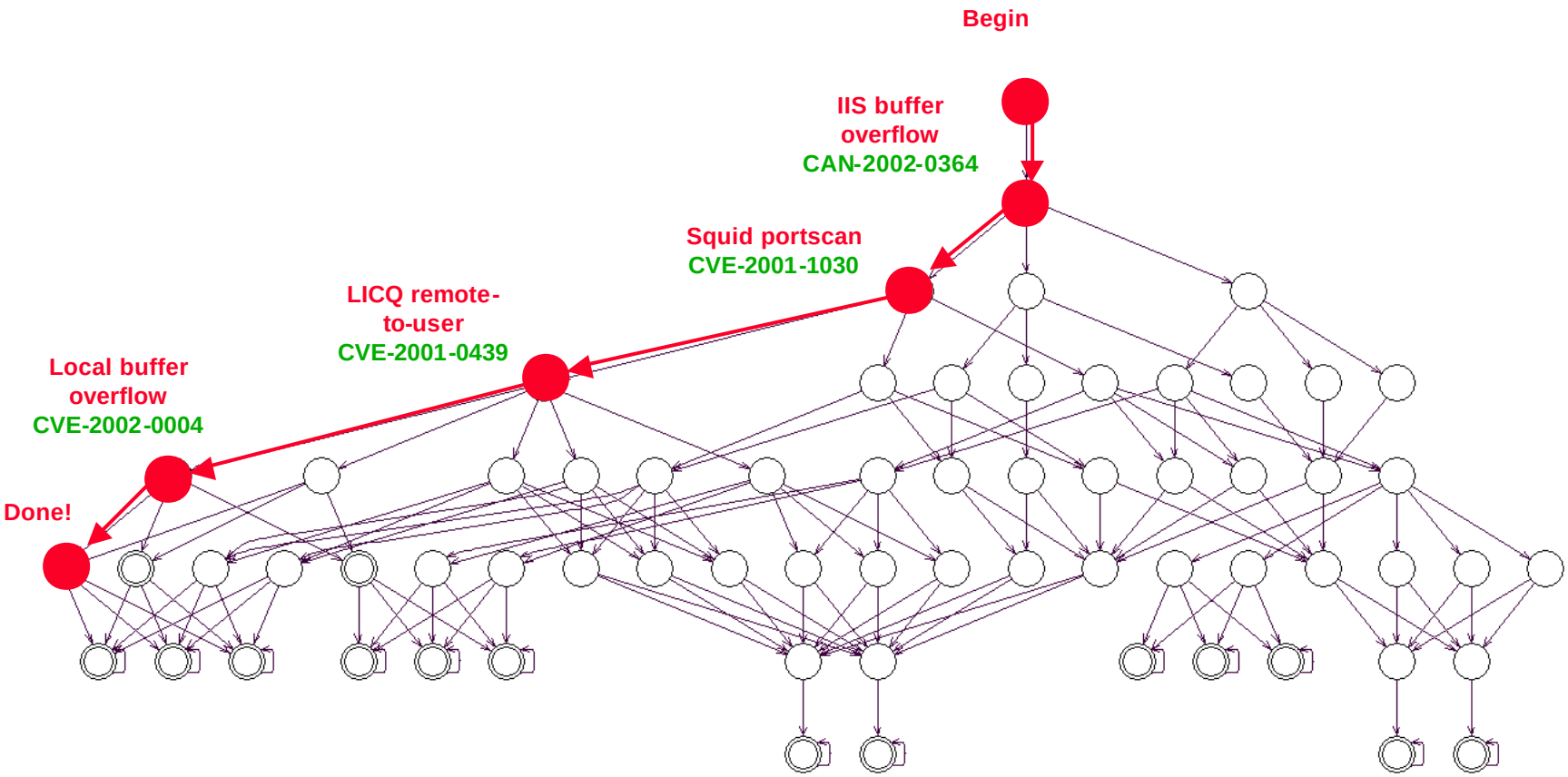
State Hashing

Method	Full State	Hashcompact	Traceback
Performance (Amortized)	O(E)	O(E)	O(E)O(depth)
Memory Overhead per State	Full State Size	8 bytes	14 bytes
Completeness	Complete Coverage	Partial Coverage	Complete Coverage

Example Attack Graph

Security property (LTL):

G (intruder.privilege(host) < root)



Application: Attack Graphs