From Designed (Insecurity) to (Designed-In) Security

Briefing for HCSS Conference
May 5, 2011

In a Nutshell

Software and system development methods in use today largely take security requirements into account late, if at all

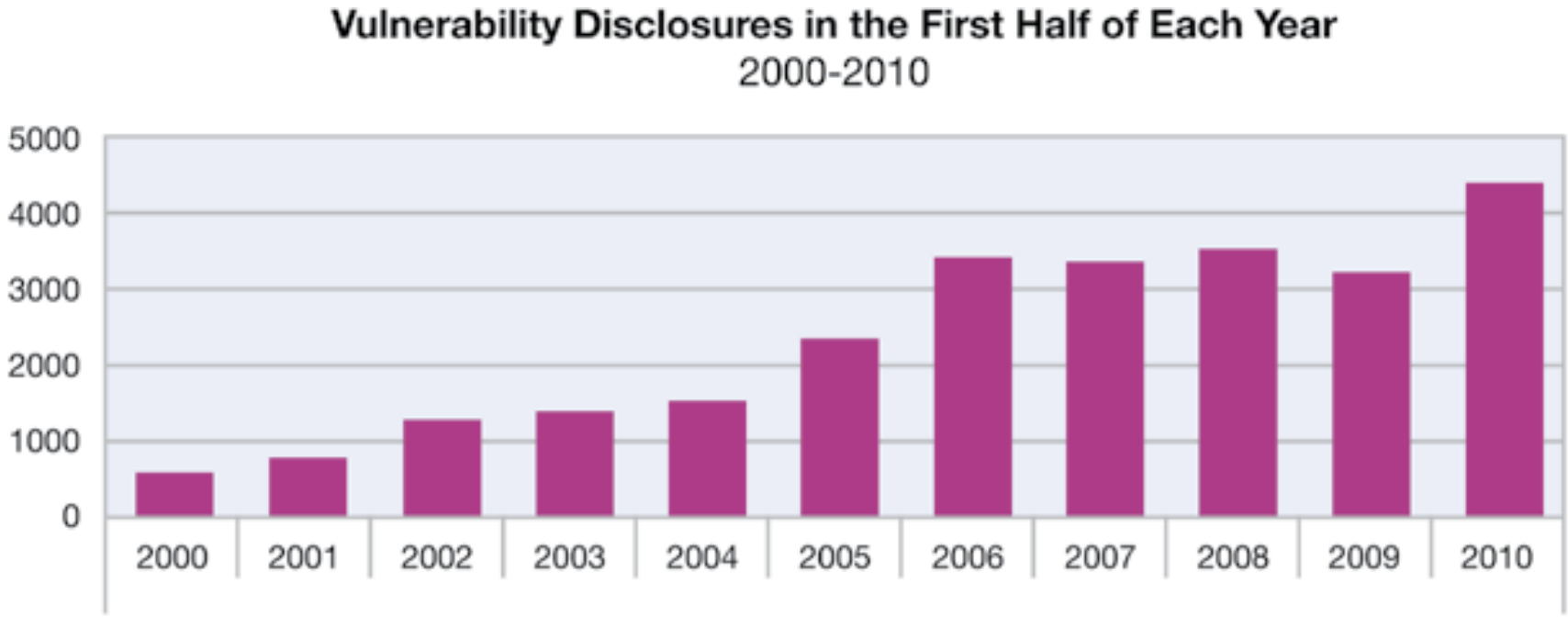
- In effect, we design security problems into our systems
- A variety of rationalizations justify current approaches
- Systems that do provide high assurance also tend to be costly to change and adapt

Significant advances in automated software analysis and system construction in the past decade raise the possibility of transformational change

- static analysis, dynamic analysis, model checking, and verification have all made significant strides
- a research push could trigger dramatic decreases in software security vulnerabilities and decrease the cost/risk of adaptation

Objective: be able to design, develop, evolve high-assurance, software-intensive systems predictably and reliably while managing risk, cost, schedule, quality, and complexity. Enable rapid adaptation while maintaining high assurance


Where are we?
0 2,6405,280 10,560 15,840 21,120 Feet
0 0.5 1 2 3 4 Miles
0 804.51,609 3,218 4,827 6,436 Meters
"The Others"
Flight 815 Trajectory
Walt!!!
DHARMA Sub "Galaga"
North Shore
Boathouse
Widmore Sub - Explosion
Raft
Pascal Plats
DHARMA Gravesite
The Flame
Barracks
Jughead (Underground)
The Temple
Pala Ferry
Tailsection of 815
The Arrow
Capsule Dump
The Door
The Orchid
French Expedition Camp
The Pearl
The Staff
The Hydra
Runway
Henry Gale's Balloon
Black Rock
Radio Tower
Taweret / Four Toed Statue
Beechcraft
The Tempest
Kahana Freighter & Helicopter
80 Miles Offshore
The Swan
"Heart" of the Island
Cockpit of 815
The Looking Glass
1950's Others Camp
Jughead - 1950's
Caves
Rousseaus' Camp
The Mesa
Well
Survivor's Beach Camp
Fuselage of 815
Rose & Bernard
Lighthouse
Jacob's Cave
N
W
E
S
5 Miles Wide
1 Mile Squares
5 Miles Wide
X
Y
Church & Sweat Lodge
Garden
Boone Hill
Ajira 316 Trajectory

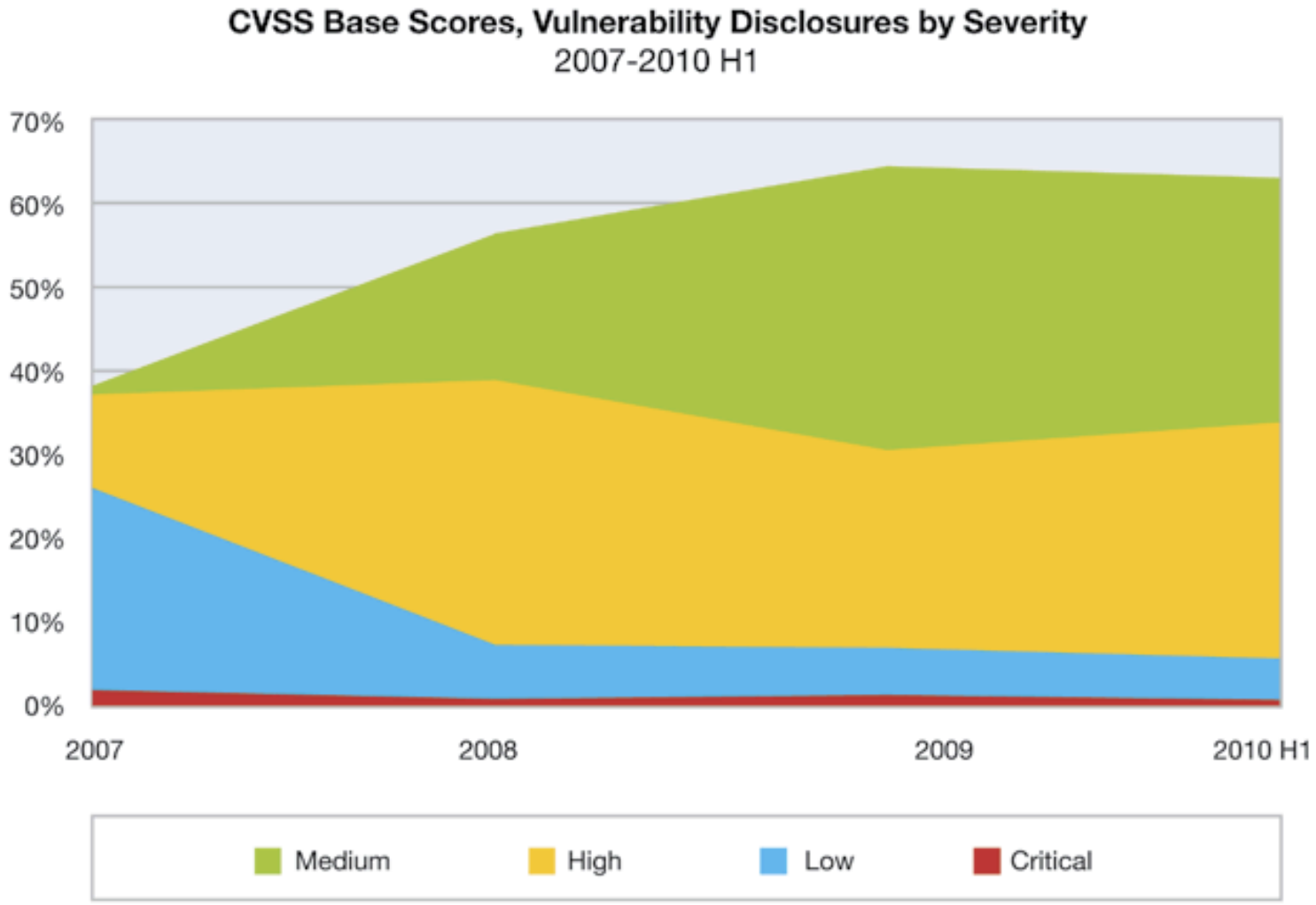


"The annual vulnerability disclosure rate now appears to be fluctuating between 6,000 and 8,000 new disclosures each year."

"Over half (55 percent) of all vulnerabilities disclosed in the first half of 2010 have no vendor-supplied patch at the end of the period. This is slightly higher than the 52 percent that applied to all of 2009."

Vulnerability is defined as a set of conditions that leads or may lead to an implicit or explicit failure of the confidentiality, integrity, or availability of an information system.

Source: IBM X-Force mid-year report, August, 2010

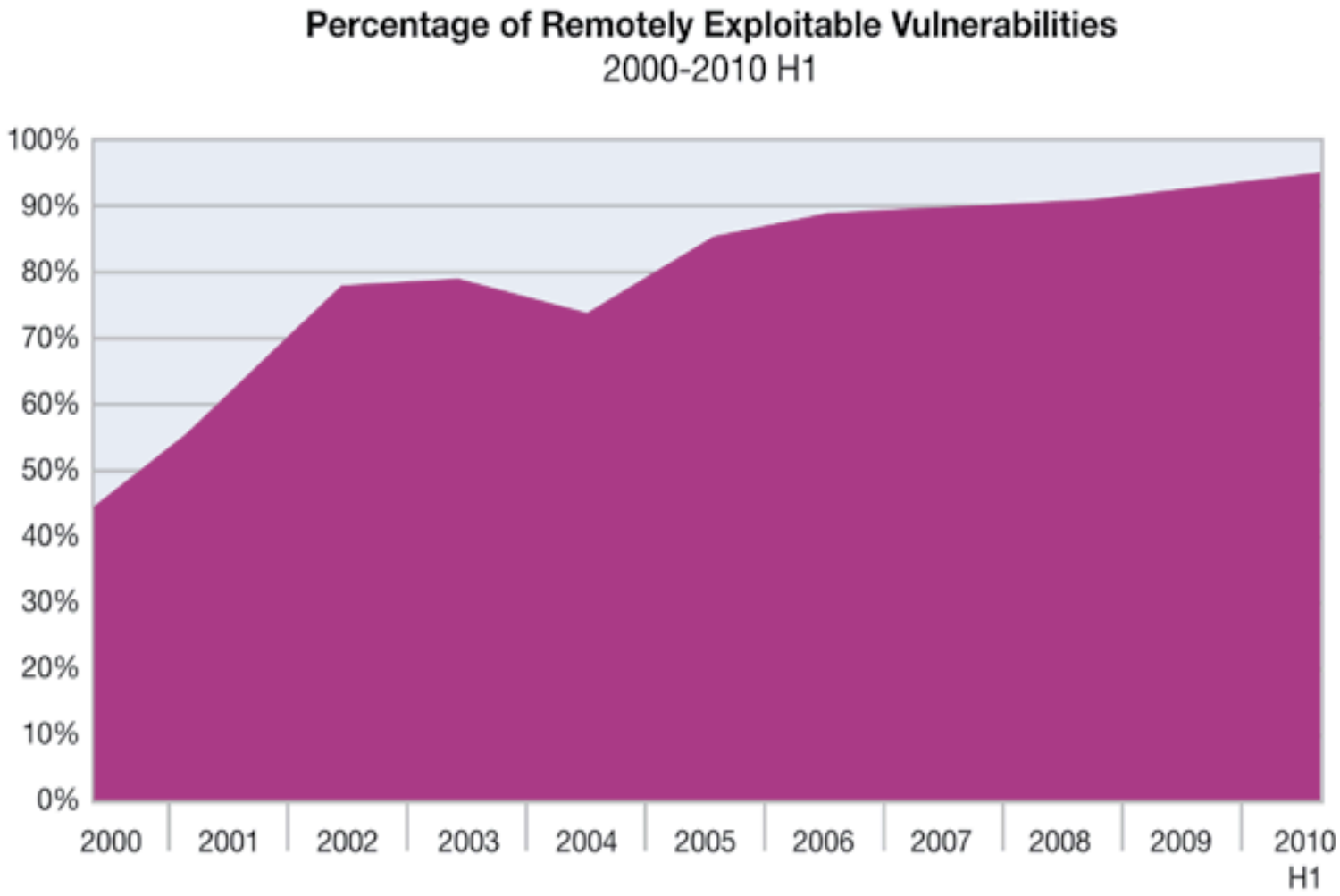


Slight relative increase in “High” vulnerabilities

“Medium” includes XSS and SQL injection

CVSS = Common Vulnerability Scoring System

Source: IBM X-Force mid-year report, August, 2010



Source: IBM X-Force mid-year report, August, 2010

Software System Development Today: Assertions without Proof

- Programmers are expensive
- Tools are used to economize on programmer time
- Programs grow in pieces from many sources
- Some tools are available for finding security vulnerabilities
- Assuring security properties of a system of programs is very difficult

➔ Most programs provide low assurance they are free of security vulnerabilities

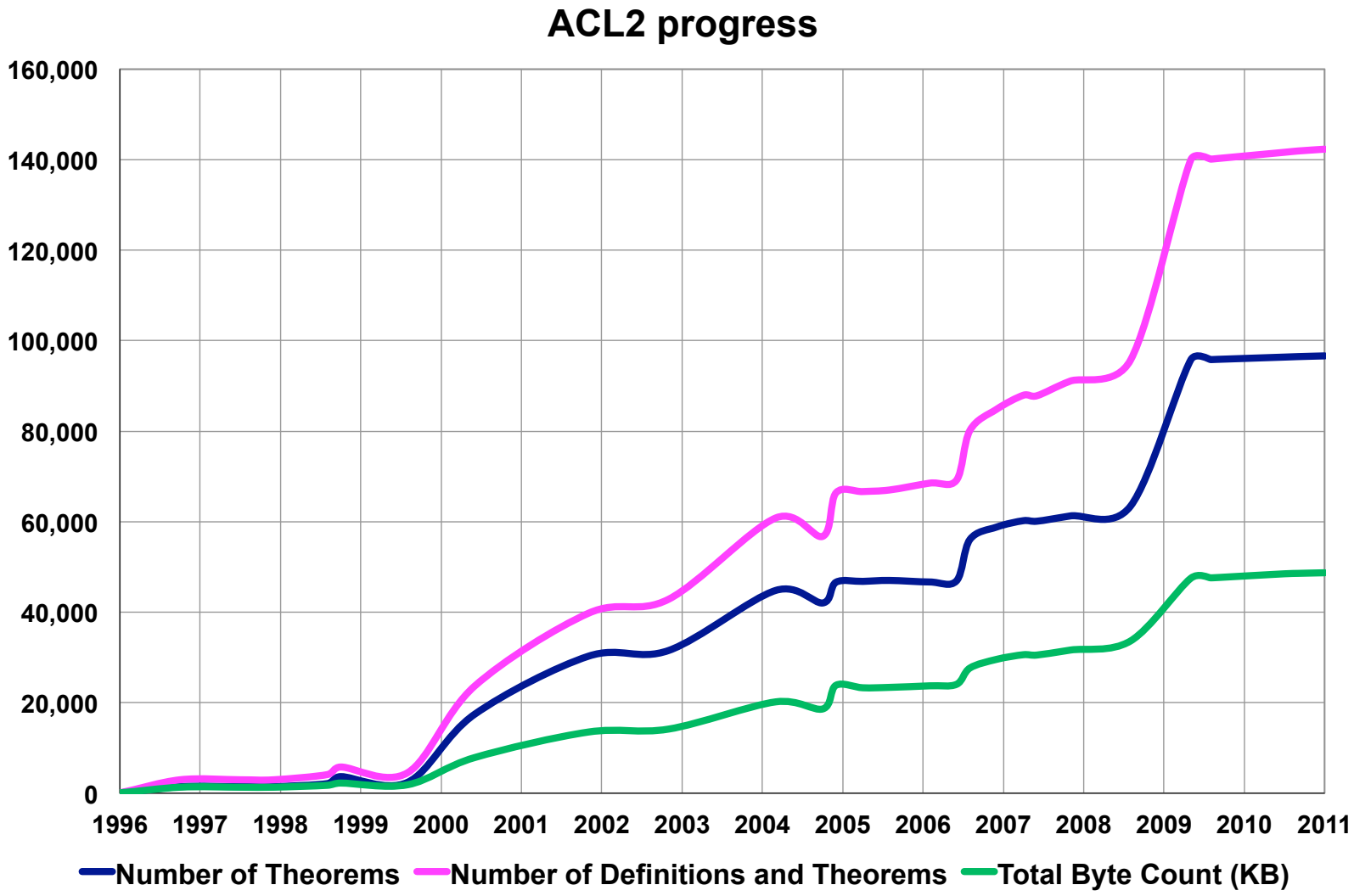
➔ Even more, most systems of programs are low assurance

➔ High assurance programs don't change very much

What Tools Can Help?

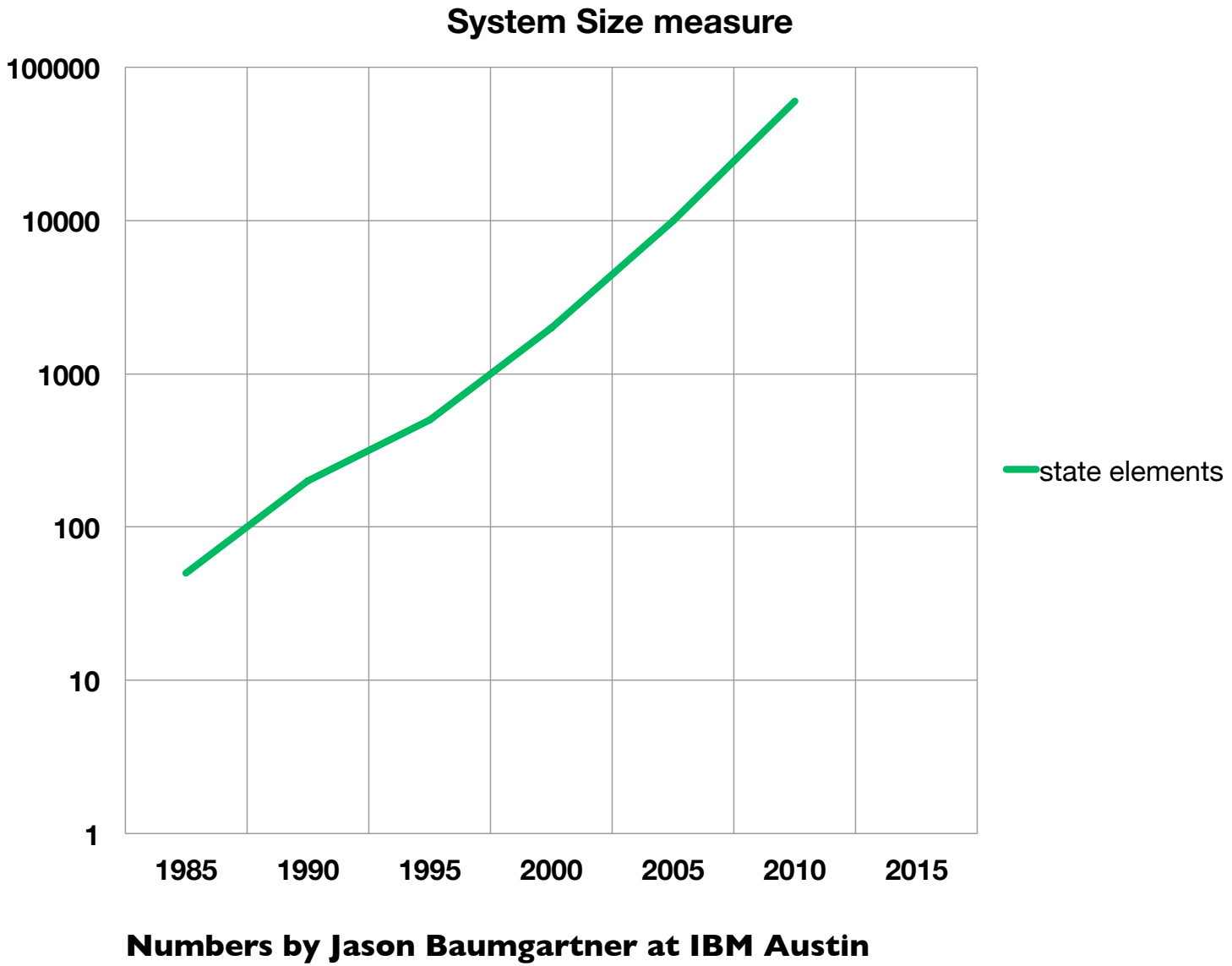
- Static Analysis
- Dynamic Analysis
- Model Checking
- Theorem Proving

Progress: Theorem Proving

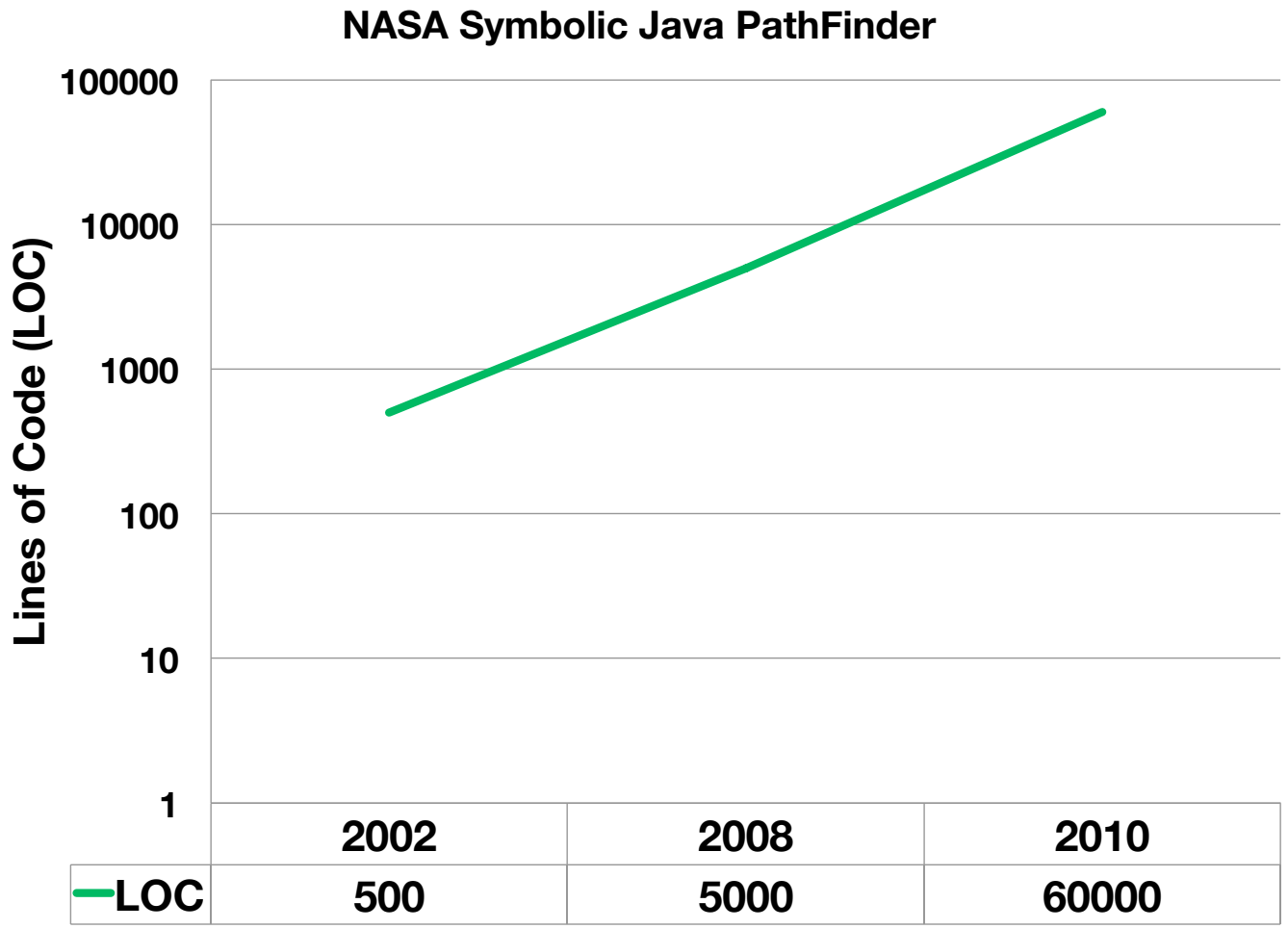


Numbers by J Moore, Matt Kaufmann, Warren Hunt, UT Austin

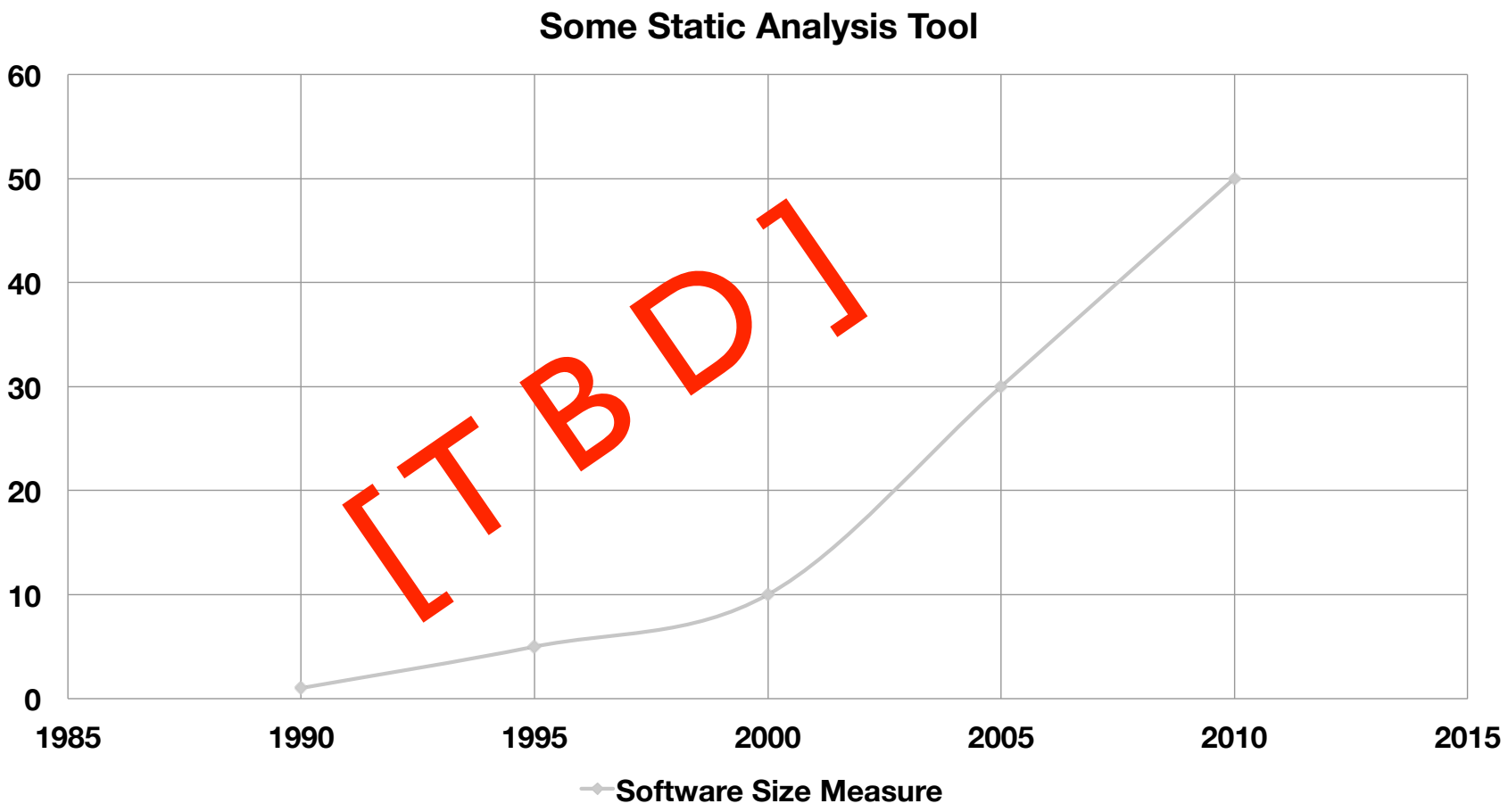
Progress: Model Checking



Progress: Dynamic Analysis



Progress: Static Analysis



What is needed to bring these and other advances to bear on system security?

Tools that

- Generate assurance evidence as a system is built
- Can be easily understood and used by real programmers (and yield benefits they can see)
- Can support integration of evidence about various components
- Can be re-applied easily as systems evolve and adapt

Some Research Challenges

- Mathematically sound techniques to support combination of models and composition of results from separate components
- Analysis techniques to enable traceable linking among diverse models and code
- Language design, processing, and tooling techniques that can provide high assurance for capable, modular, flexible systems
- Team and supply chain practices to facilitate comosition of assurance in the supply chain
- Tools to support assurance evidence management
- Learning how to make all of the above usable
- Learning what incentives (e.g. ability to quantify results) might motivate the use of these tools

What do we need to tip the balance?

Good ideas from YOU!

Unreasonably vulnerable

Reasonably Invulnerable

Thank You