SAND2011-5891P

A Simplified Version of 'Complex System Modeling and Science-Based Cybersecurity'

Jackson Mayo (8953)
Sandia National Laboratories
August 15, 2011

Sandia National Laboratories is a multiprogram laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Note

- This is a simplified version of the presentation "Complex System Modeling and Science-Based Cybersecurity" (SAND2011-1375P)

Problem

- In general case, cannot find all vulnerabilities in a computer program
  - This has been proven mathematically
- So
  - Good guy cannot fix all of the vulnerabilities
  - And bad guy can always get in
    - Because bad guy only needs to find one vulnerability
- Bad guy can find a vulnerability by “fuzz” testing
  - Throw all kinds of random input at the program and see when it breaks

How Things Got This Way

- Enormously complex hardware and software are stamped out en masse
  - Producing identical, general-purpose systems (e.g., Intel CPUs, Microsoft Windows) achieves economies of scale
  - No one knows *everything* these systems can do
  - Since they are ubiquitous and cheap, an attacker can practice on an identical copy of *your* system

Nature

- Nature faces the same problem
  - Example: How defend against unknowable bacteria and viruses?
- Answer
  - Vary the implementation (but not the function)
    - Different implementations weak in different places
  - If multiple independent weaknesses are needed, chance of infection decreases exponentially
    - Principle of robustness or fault-tolerance

Computers

- Hardware
  - Example: Space Shuttle: 4 computers, identical software, different hardware, same design
    - Focus is on robustness to hardware failure
- Software
  - *N*-version software
    - Use *N* instances that process the same input in parallel
      - Same function, different implementation
    - Choose answer by majority

Attack!

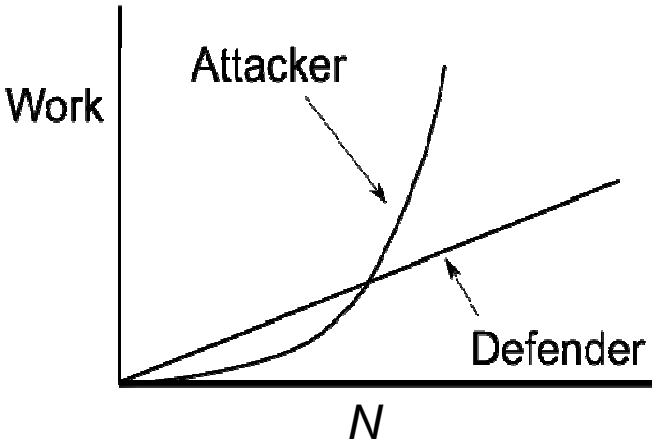
- Bad guy attacks…and breaks an instance!
- Defense
  - Only one instance broken; other $N - 1$ ok
    - This is determined by comparing their individual outputs
  - Repair:
    - Automatically generate new instance
      - Again, same function, different implementation
    - Replace broken instance
- Now the bad guy must start all over again
- Bad guy is on the run!

Key

- Different implementation → different vulnerabilities (by assumption)
- Each instance is a different implementation
- So successful attack on one instance → probably not successful on another instance
- Remember:
  - Good guy cannot find all the vulnerabilities
  - Bad guy finds vulnerabilities one at a time

Measuring the Payoff

- In an *N*-version voting system with sufficiently diverse implementations:
  - Work for attacker to simultaneously compromise a majority of versions is *exponential* in *N*
  - Work for defender to produce and run *N* versions is *linear* in *N*



Complexity Science

- Reductionism: Examining parts provides understanding of whole
  - Most real-world systems too complex for this
    - Including most digital hardware and software
    - But *formal methods* can work in simpler digital systems
- Holism: Understand new systems by abstracting from other systems
  - Complexity science helps here, with concepts such as robustness

Sandia Research

- Automatic generation of different implementations of the same function
  - Pursuing *genetic programming* techniques inspired by biological evolution
- More general system design principles for achieving robustness
  - Leveraging network models and game theory
  - May be more efficient than diverse replication
- These efforts are yielding promising results

A Cybersecurity Vision

- How are high-consequence digital systems best created?
  - Design for analyzability
    - Enable an appropriate combination of *formal methods* (exhaustive analysis of smaller systems) and *complexity science* (probabilistic analysis of larger systems)
  - Leverage human effort in potentially new ways
    - Take advantage of tools such as genetic programming
    - The human input may not be a full implementation, but a design specification that constrains the search space