Leveraging Complexity in Software for Cybersecurity

Robert C. Armstrong and Jackson R. Mayo

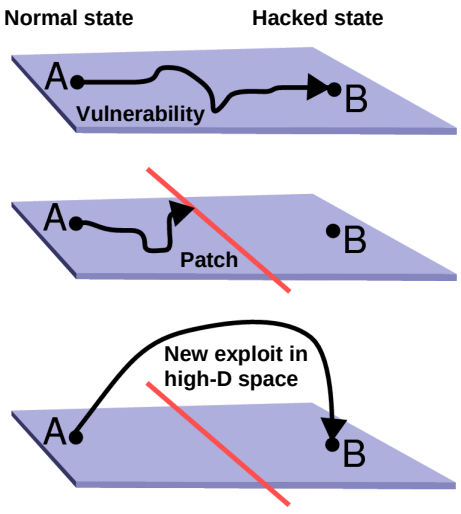
Sandia National Laboratories
Livermore, CA

13 April 2009

Sandia National Laboratories

Combinatorial explosion generates fundamental asymmetry of cyber defense

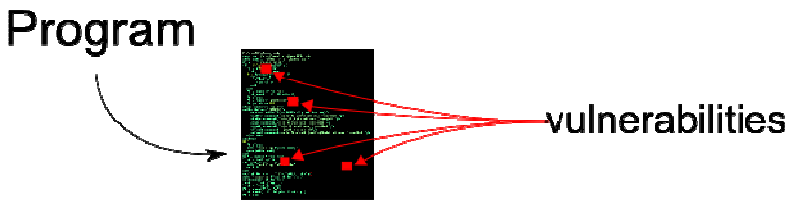
- **Vast state space defies enumeration, even for desktop computers; new attacks every day**
- **Turing completeness makes behavior effectively unpredictable**
- **Numerous hidden attack vectors give attackers asymmetric advantage**
- **Current cyber defenses amount to one "Maginot Line" after another**



Sandia National Laboratories

Securing an arbitrary code is not just hard; it's impossible

- **Restated: Generic code has vulnerabilities that are unprovable and unknowable**
  - – *Not* statistical, even in principle
  - – Turing completeness demands that a generic code is undecidable



- **So now what?**

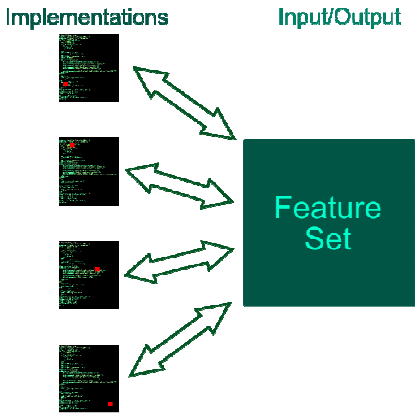
Sandia National Laboratories

Complexity makes cyber threats *asymmetric*

- **Developer, user, *and attacker* all don't know where the vulnerabilities are (*undecidable*)**
- **Asymmetry: One vulnerability compromises the whole code**
  - – Developer has to find all of them (impossible in general)
- **No one can guarantee "this code is clean" or even quantify improvement**

Sandia National Laboratories

Observation #1: A program's feature set has many implementations

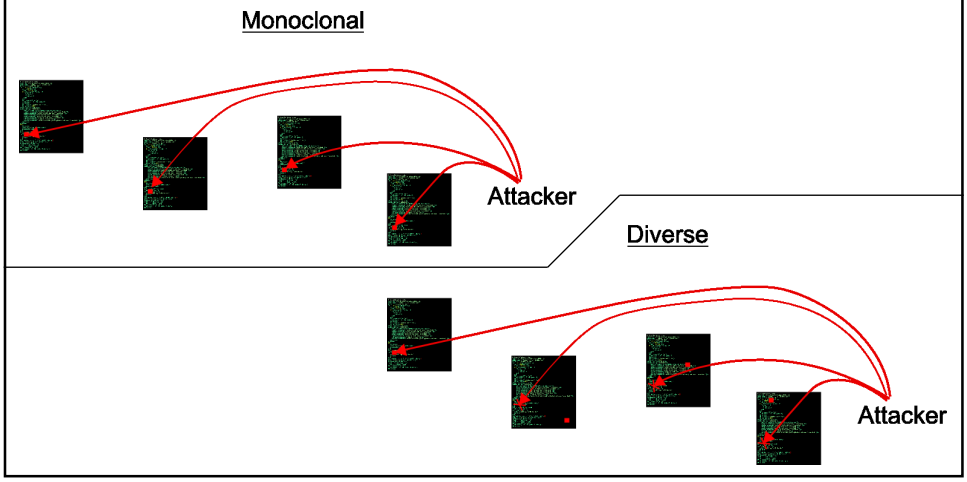


- **Feature set is defined by a test suite**
- **Test suite verifies that an implementation conforms to desired functionality**
- **Test suite is a sample; cannot realistically cover all possible input/outputs**
- **Vulnerabilities arise from untested input/outputs**
- **Any feature set has infinitely many implementations**
  - Finite large number if size is bounded

Sandia National Laboratories

Observation #2: Ensemble of instances permits the formulation of statistics

- **Assume: Multiple implementations randomize security holes**
- **Ensemble of multiple-version, "randomized" undecidable codes allows formation of security *improvement* statistics**



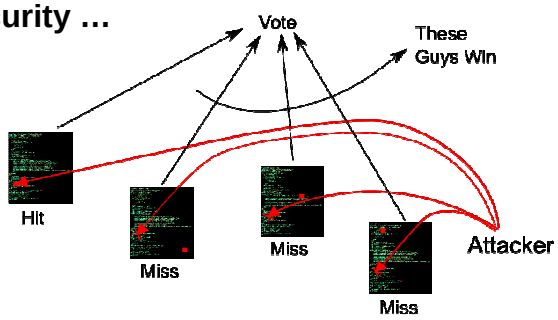
High-reliability systems can be constructed from "*N*-version software"

- **Space Shuttle: 4 computers, identical software, different hardware components, same design**
  - Focus is on hardware faults
- **Similarly, software redundancy used mostly for control systems up to now**
  - *N*-version software: Multiple versions implemented to the same feature set by different developers
- **Models of *N*-version software view the control system as a stochastic process that walks the code graph of the software**
  - Control system takes the place of a "fuzzer"

Sandia National Laboratories

Similarly, *N*-version software can quantifiably improve cybersecurity

- **Clear generalization of *N*-version reliability to cybersecurity …**

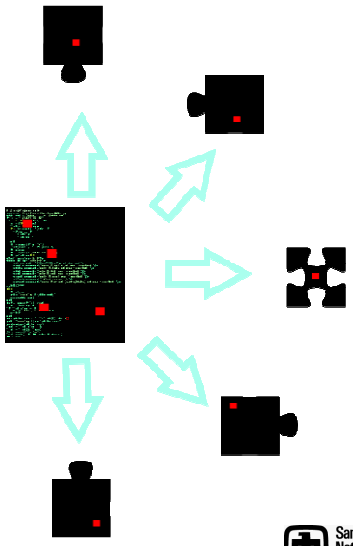


- **… but there are important differences requiring enabling technology**
  - Compromised versions must be removed and replaced
  - Hand-made new versions are time-consuming and expensive
    - **May repeat previous mistakes**

Sandia National Laboratories

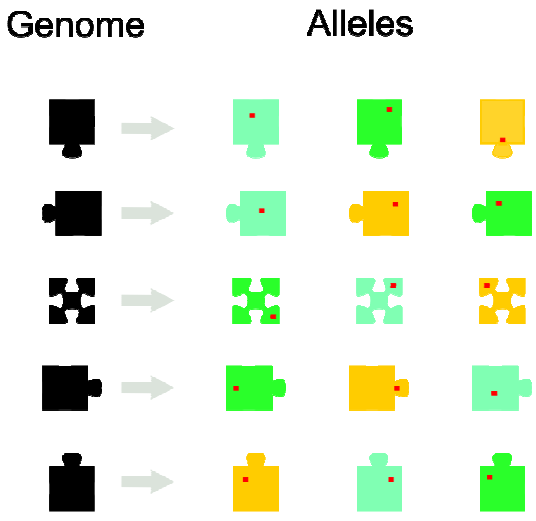
A simple example: Diverse software can be constructed from components

- **Component-based codes automatically conform to a feature set if the constituent components conform to their individual feature sets (semantic interfaces)**
  - Multiple implementations of the code amount to multiple versions of components
  - Components can be mixed and matched to form a combinatorial number of code implementations



Sandia National Laboratories

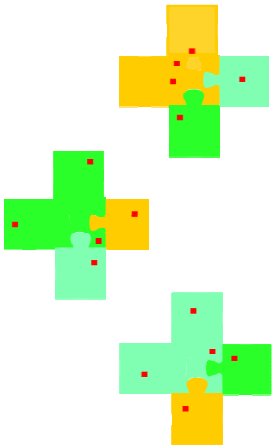
Living systems adapt to cope with unknowable attacks



- **A component type is similar to a gene; component implementations are similar to alleles of a gene**

Sandia National Laboratories

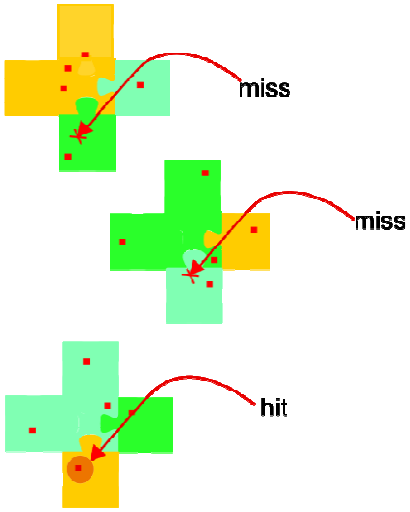
Reassemble alleles into individuals



- Different alleles can be assembled into new individuals that have “randomized” security holes
- New individuals are differently vulnerable and potentially adaptive

Sandia National Laboratories

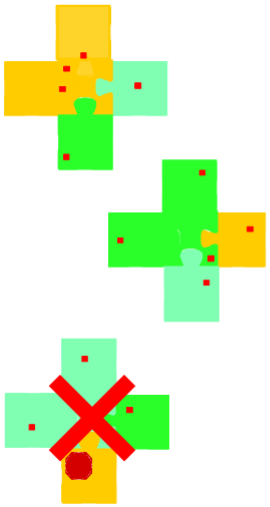
Compare responses from individuals



- Now different individuals will produce the same feature set but react differently to attacks

Sandia National Laboratories

Evolve new and more robust individuals

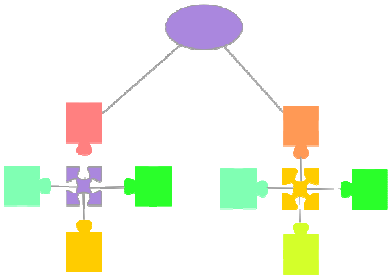


- **Eliminate the one with the differentiated response**

Sandia National Laboratories

Genetic approach is a special case of fault-tolerant system design

- **Previous genetic approach is a contrived example of a network of entities robust to attack …**



- **… others certainly exist**
  - Seek more efficient examples that do not require total replication of every attackable system
  - Seek an arrangement of entities that has no single point of failure

Sandia National Laboratories

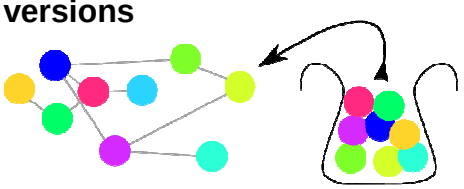
Research needs: Quantify and automate "program randomization"

- **Exploit existing compiler code-rewriting techniques**
  - Stack randomization, "semantically invariant" rewrites, etc.
  - Obfuscation techniques
- **Quantify, or at least *qualify*, sufficient randomness to expect that vulnerabilities do not repeat**
  - Use "fuzzers," emulation, and automated software analysis to find and compare *some* implementation-induced vulnerabilities
  - Use data to model the prevalence and rate of exposing new vulnerabilities
- **Automate the process of finding new versions**
  - Genetic programming techniques hold promise
    - **We are starting from a known implementation**
    - **Seeking only diversity**

Sandia National Laboratories

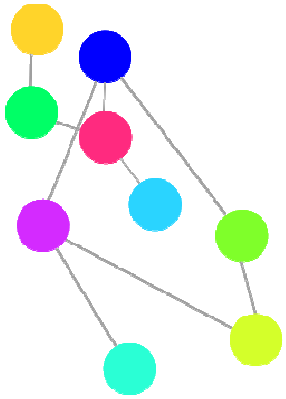
Research needs: Algorithms to exploit *N*-version concept for cybersecurity

- **Algorithms that implement the *N*-version idea but automatically remove compromised versions and create new ones**
  - Similar to biological selection process
  - Diversity over ensemble and over time
- **In-depth distributed redundancy and voting**
  - Decentralized methods to prevent single points of attack
- **Selecting entities from a previously generated collection to form software versions**
  - Starting from one or a few implementations, derive variants by "diffusing" code while maintaining feature set



Sandia National Laboratories

Generalization: Seek the emergent property of robustness to attack



- **Seek a generalized approach similar to RAID, where both diversity and voting are incorporated in-depth**
  - Eliminate the need for complete replication of diverse programs
  - Eliminate single points of failure
- **Seek a network of entities that are collectively robust to attacks "in the cloud"**
- **Entities can generalized to**
  - Programs
  - Hardware
  - Networks
  - Systems of these systems
  - And so on …

Sandia National Laboratories

Conclusion: Software complexity enables a new approach to cybersecurity

- **Complexity science helps cybersecurity confront the unknown and unknowable**
  - A single implementation is unknowable and unpredictable
  - Ensembles of unknowables provide potentially quantitative vulnerability measures
- **Practical systems will benefit from means for generating and measuring diversity**
  - Is there a "Hamming distance" for differing implementations?
- **Diversity is one way to turn complexity on the attacker**
  - Attacker cannot assume uniformity and permanence of target
    - **Vary over space (ensemble) and time (annealing)**
  - Redundancy offers a way to detect and recover from attacks

Sandia National Laboratories