N-Variant Systems

A Secretless Framework for Security through Diversity

Benjamin Cox

David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong and Jason Hiser

University of Virginia

USENIX Security 2006, Vancouver BC

Attacks Require Knowledge

Heat (1995)

Artificial Diversity Methods

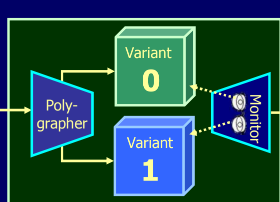
Attack Class	Assumptions	Diversity
Code Injection	Instruction Set	Instruction Set Randomization [Barrantes+ 2003], [Kc+ 2003]
Memory Corruption	Address Space Layout	Address Space Randomization [Forrest+ 97], [Bhatkar+2003]
Return to Lib-C	Calling Convention	Calling Sequence Diversity
Time Of Check To Time Of Use	Process Scheduling	???
⋮	⋮	⋮

Limitations of Diversity Methods

- Requires high entropy
  - Difficulty scales with the number of possibilities
  - Low entropy broken by brute force [Shacham+, CCS 04]
- Attacker can learn the diversity key
  - Incremental attacks [Sovarel+, USENIX Sec 05]
  - Side channels
- Security assurance difficult
  - Vulnerability changed, not removed
  - Assumes secrets can be kept

N-Variant System Framework

- Run variants in parallel with identical inputs
- Variants designed to vary assumptions
- Check behavior of variants is equivalent



Redundant Execution

- Debugging [Knowlton 68]
  - Rearrange code and memory segments of program and run in parallel
- Robustness [Berger & Zorn 06]
  - Dynamically randomize layout of heap and run multiple versions in parallel comparing output
- Security [Reynolds+ 03, Totel+ 05, Gao+ 05]
  - Design diversity with rough comparison

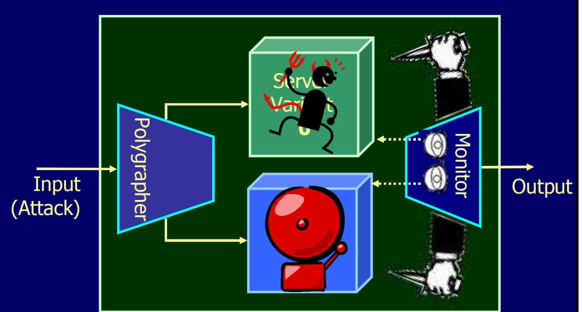
Proving Detection

- Detection property
  - Attack causes states between variants to diverge noticeably
  - If one variant is compromised another must enter alarm state
- Normal equivalence
  - Before attack, variants must be in equivalent states
  - Deterministic behavior

Example Variations

- Address space partitioning
  - Detection property: access injected address
  - Normal equivalence: addresses identical except for high order bit
- Instruction set tagging
  - Detection property: run injected code
  - Normal equivalence: instructions in variants are same except for tags

Thwarting Attacks



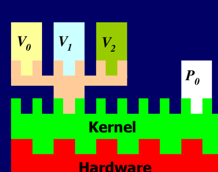
Implementation Requirements

- Polygrapher
  - Identical inputs to variants at same time
- Monitor
  - Continually examine variants completely
- Variants
  - Fully isolated, behave identically on normal inputs

Too expensive for real systems

Implementation

- Modified Linux 2.6.11 kernel
- Run variants as processes
- Create 2 new system calls
  - `n_variant_fork`
  - `n_variant_execve`
- Wrap existing system calls
  - Replicate input
  - Monitor system calls



Wrappers

- Check consistency
- I/O wrappers (e.g., read(), write())
  - Perform system call once
  - Return same result to all variants
- Reflective (e.g., setuid(), signal())
  - Perform corresponding system call on all variants
  - Check identical result

```
sys_write_wrapper(int fd, char __user * buf, int len){
  if (!IS_VARIANT(current)) { Perform System Call }
  else {
    if (!inSystemCall(current->nv_system)) {
      Save Parameters
      Sleep
      Return Result Value
    } else if (currentSystemCall(current->nv_system) !=SYS_WRITE) {
      DIVERGENCE – different system calls
    } else if (!Parameters Match) {
      DIVERGENCE – different parameters
    } else if (!isLastVariant(current->nv_system) {
      Sleep
      Return Result Value
    } else {
      Perform System Call
      Save Result
      Wake Up All Variants
      Return Result Value
    }
  }}
```

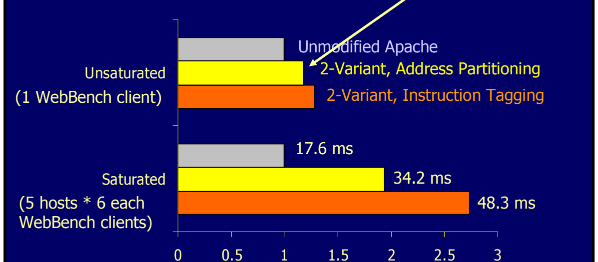

Constructing Variants

- Address Space Partitioning
  - Specify segments' start addresses and sizes
  - OS detected injected address as segmentation fault
- Instruction Set Tagging
  - Use Diablo [De Sutter+ 03] to insert tags into binary
  - Use Strata [Scott+ 02] to check and remove tags

Results

Latency increase from 2.35 to 2.77 ms

Normalized Latency



Apache 1.3 on Linux 2.6.11

Implementation Limitations

- Expensive for CPU-bound servers
- Requires deterministic behavior
  - Most sources of nondeterminism removed
  - Timing can be a problem (see poster)
- Dangerous system calls
  - `execve()`, `mmap()`
- Variants lack complete isolation
- Does not address recovery

Fundamental Limitation

- Only protects against attacks whose assumptions are broken by variations
- Opportunities
  - Low entropy variations (e.g., calling conventions, timing, root uid, …)
  - High-level variations
    - Requires knowledge of application semantics

Summary

N-Variant systems employ artificial diversity techniques to provide provable resilience against certain classes of attacks without needing secrets.

www.cs.virginia.edu/nvariant

Supported by the National Science Foundation Cyber Trust Program and DARPA SRS