Independence from Attack

Fred B. Schneider
Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.
Fault-tolerance by Replication

The basic recipe …

- Servers are deterministic state machines. Clients make requests.
- Server replicas run on distinct hosts.
- Replica coordination protocol exists.
Independence:
Eschewing Shared Design / Code

Solution: Diversity!

- Expensive or impossible to obtain:
  - Development costs
  - Interoperability risks

- Leverage what diversity exists.

- Mechanically create “artificial diversity”.
  
  ... Employ a [program obfuscator](http://example.com/program_obfuscator).
Proactive Recovery

A mobile adversary can erode independence.

Idea: Proactively re-obfuscating server code defends against this:

- tolerates $t$ compromises over *lifetime*

- versus -

- tolerates $t$ compromises in *window of vulnerability*
Implementing Proactive Obfuscation

Challenges:
- State recovery
- Protect Obfuscator
- Mask outages

Random key: 0110101100…

System

Obfuscator

server replica
Proactive Obfuscation Prototype

Specification:
- Unlikely that attacker can gain control of the service.
- A steady stream of attacks might block service. (But service is restored once that stream is terminated.)

Implementation:
- Based on OpenBSD 3.4 (soon 3.5).
- Creates a RAM-disk kernel.
  - Server code (morph) is stored in RAM-disk.
  - Syscall numbers are changed in each morph.
  - No other source or object code obfuscation (at present).
- Trusted “kernel-builder” machine –or– pxe boot, to create new morphs.
Obfuscation: Goals and Options

Semantics-preserving random program rewriting...

**Goals:** Attacker does not know:
- address of specific instruction subsequences.
- address or representation scheme for variables.
- name or service entry point for any system service.

**Options:**
- Obfuscate source (arglist, stack layout, \( \ldots \)).
- Obfuscate object or binary (syscall meanings, basic block and variable positions, relative offsets, \( \ldots \)).
- All of the above.
Obfuscation versus Type Checking

**Thesis:** Obfuscation and probabilistic dynamic type systems can “defend against” the same attacks.

From “thesis” → “theorem” requires fixing:
- a language
- a type system
- a set of attacks

..And we’ve done this [CSFW’06]
Pros and Cons

- **Type systems:**
  - Prevent attacks (always---not just probably)
  - If static, add no run-time cost
  - Not always part of the language.

- **Obfuscation**
  - Works on legacy code.
  - Doesn’t always defend.