Testing Defensive Systems

1. NIDS
   Problem: Find an attack instance that eludes a NIDS.
   Solution: Attack generation using natural deduction.
   Shai Rubin · Somesh Jha · Bart Miller

2. Virus scanners
   Problem: Generate virus sample that evades AV tool.
   Solution: Guided attack generation using oracle access.
   Mihai Christodorescu · Somesh Jha
Problem

Given:
- a defensive system (NIDS, virus scanner)
- a known attack
- a set of transformation rules: TCP/IP fragmentation, code obfuscation, etc.

- How can we test, or even verify, that a defensive system detects all instances of a given attack?
NIDS Are Untrustworthy

- Many false positives.
- More troubling, false negatives: attacker has succeeded to elude a NIDS.
- Attack transformation: alter an attack, but keep its semantics, so it no longer matches the NIDS signature.

Problem: How can we test, or even verify, that a NIDS detects all instances of a given attack?
Previous Solutions

• Random testing
  - Not exhaustive
  - No control over testing
  - Not always sound

• Manual testing
  - Not efficient
Our Approach

• Formally represent attackers’ abilities as transformation rules of a natural deduction system.

• Use inference engine to exhaustively apply the rules to generate all possible mutations.
  ✓ exhaustive
  ✓ sound
  ✓ efficient
AGENT: **Attack Generation for NIDS Testing**

- Transformation Rules
- Representative Attack
- Inference Engine
- Set of Attack Variants
- Attack Simulator
- Snort
- Detect?
  - Yes, check another
  - No
    - Eluding Instance

No
Current Status

• Issues addressed:
  - Formulating rules
  - Finding a representative attack
  - Large set of mutations (millions)

• Results:
  - Prototype implemented (TCP + Payload mutations)
  - Four serious vulnerabilities in Snort (reported + fixed)
AV Tools Are Untrustworthy

• Critical problem: false negatives
  - An active virus sneaked in undetected

• Program obfuscation
  - Alter a virus through various transformations
  - Maintain virus semantics
  - Mutated virus is no longer detected

Problem: How can we test the limits of a virus scanner with respect to the mutations of a given virus?
Our Approach

• Formalize attacker obfuscations as transformation rules

• Find the minimal obfuscation that renders an undetected virus variant
  ✓ Automatic signature discovery
  ✓ Minimal information needed:
    ▪ Oracle access to virus scanner
  ✓ Efficient binary search
Virus Scanner Test Generator

Obfuscation Rules

Representative Virus

Obfuscation Engine

Set of Variants

Parameter Generator

Virus Signature

Detected?

Yes

No

Virus Scanner

Detected?

Yes

Set of Variants

Obfuscation Engine

Obfuscation Rules
Past, Present & Future

Current Results:
- Prototype for Visual Basic worms implemented
- VB worm signatures discovered for several virus scanners

Future:
- Formalize search space
- Automatically discover detection heuristics
- Performance improvements
Testing Defensive Systems

WiSA Security Group

University of Wisconsin, Madison
Automatic Test Generation

Malware

Obfuscation Engine

Malware Detector

Result Analysis

Library of parametrized obfuscations.

Behaviorally-equivalent variants of the original malware sample.

Code reordering parameters:
- program range
- type of reordering (physical, execution)
- new instruction order
The More You Know...

- Can the attacker precisely evade detection?

Yes, use a *signature discovery algorithm*.

**Given**: program P with n insns.

**Assume**: signature has k program insns.

**Algorithm**:
1. Find the first signature instruction using binary search and opaque obfuscations
2. Mask the found signature instruction
3. Repeat until no more sig. insns found
Signature Discovery

Use the malware detector as an oracle.