

Immunizing Application Communities By Cooperative Signature Generation

R. Sekar

Center for Cyber Security

Stony Brook University

State of Art

- Vendor-provided patches
 - Long delay for vendor patch availability
- Signature-based filtering (“IPS”)

Result: Inability to cope with emerging threats

- Large-scale, automated, repetitive attacks
- Zero-day exploits
- Fast-spreading attacks

Our Approach

- **Automatic signature generation**
 - Signatures filter-out attack-bearing inputs
 - Human involvement minimized (or eliminated) to reduce delays
- **Members of an AC can collaborate in**
 - **Signature generation**
 - Benefit from aggregating attacks across the AC
 - Increased signature quality (e.g., reduced FPs)
 - **Distribution and deployment**
 - Members of an AC lacking signature generation capabilities can still benefit from these signatures

Benefits of Input Filtering

- **Simplifies recovery**
 - Most programs expect and handle input errors, e.g., broken network connections
 - Error-handling code built into the application can be used for recovery
 - Contrast this with error-checking predicates inserted at arbitrary points in programs
- **Eases deployment**
 - Filtering can be implemented in an external module, e.g., an inline network filter

Approach Overview

1. Attack detection

- ❑ Process crash, execution of `/bin/sh`, ...

2. Traceback to input

- ❑ Correlate attack effect to some part of input

3. Inferring semantic context of attack

- ❑ e.g., protocol field or message type involved in attack

4. Signature generation

- ❑ Identify characteristics of attack-bearing inputs that aren't shared by benign inputs

Signature Generation for Buffer Overflow Attacks

1. Attack detection

- Address Space Randomization

2. Traceback to input

- Forensic analysis of process memory image

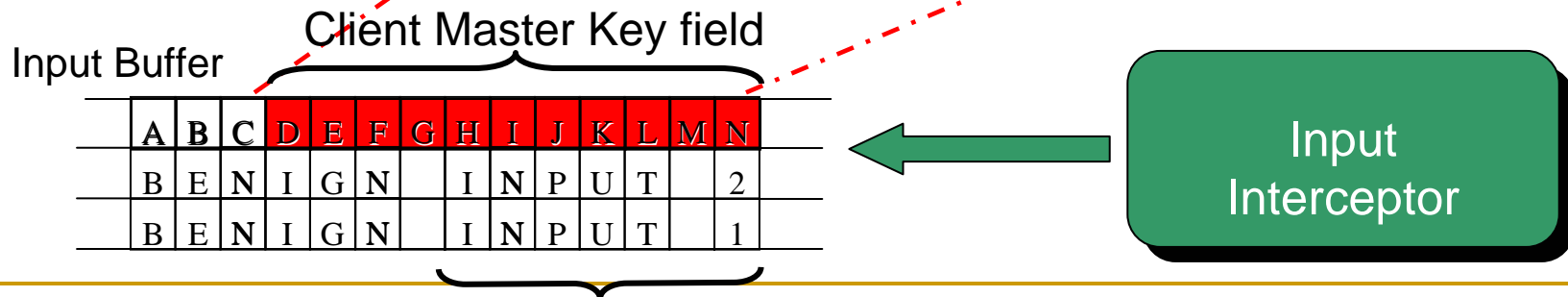
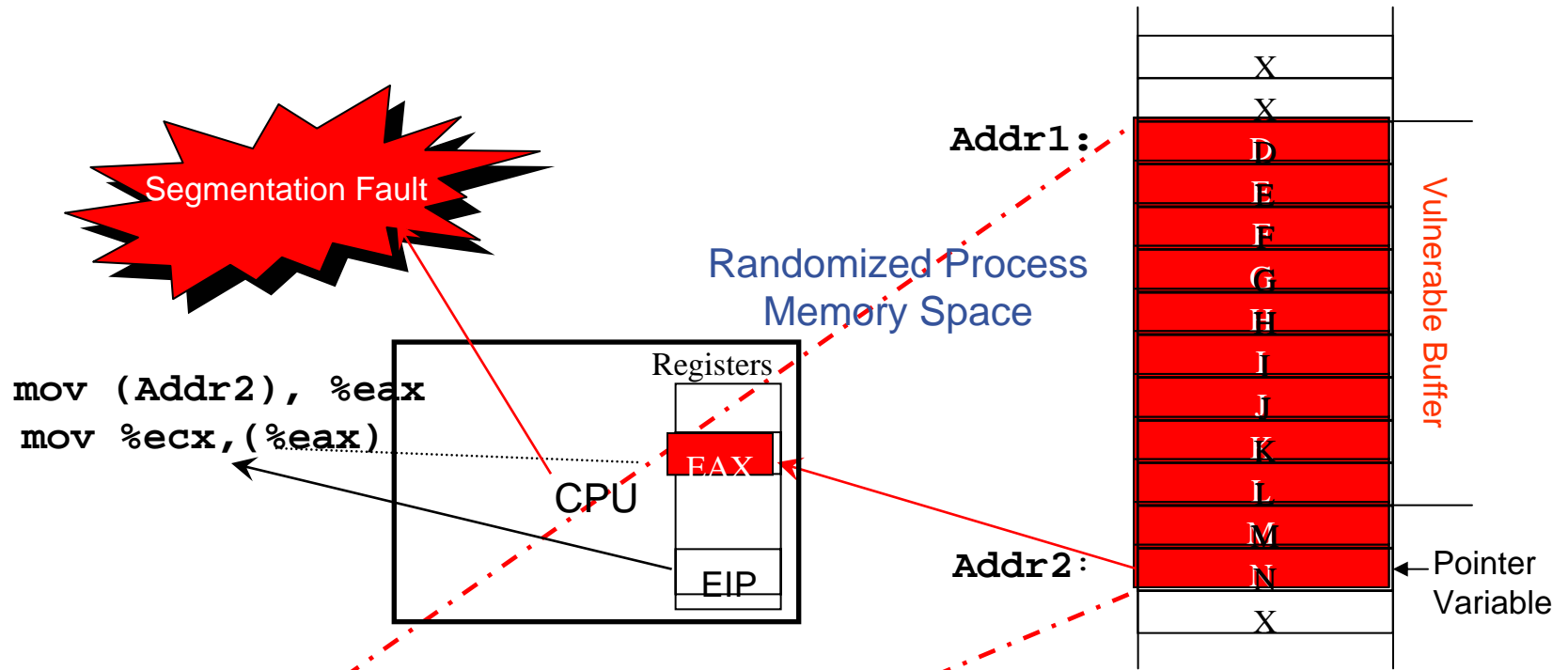
3. Inferring semantic context of attack

- Using simplified message format specifications

4. Signature generation

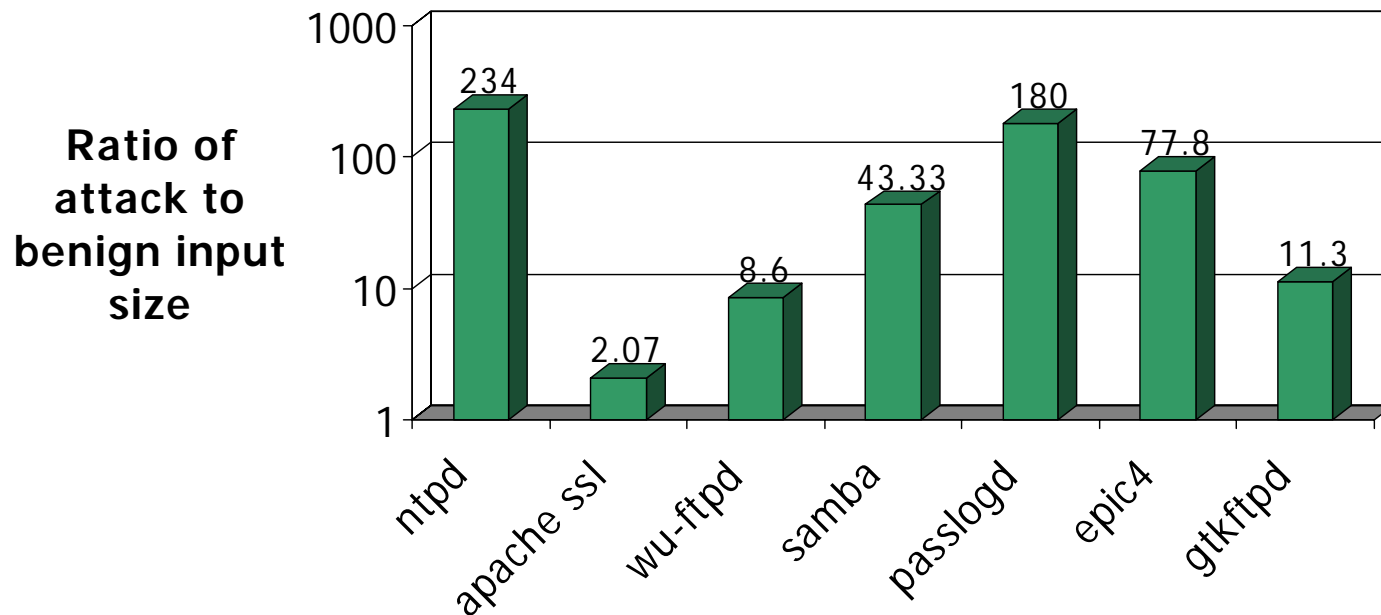
- Uses characteristics of buffer overflows, e.g., excessively long input field

OpenSSL Heap Overflow (Slapper worm)



Signature Generation: Preliminary Results

- Runtime overheads under 10%
- ~millisecond latency for signature generation



Key Technologies for Signature Generation

1. Attack detection

- ❑ Address Space Randomization
- ❑ Behavior or policy-based intrusion detection

2. Traceback to input

- ❑ Forensic analysis of process memory image
- ❑ Fine-grained taint analysis

3. Inferring semantic context of attack

- ❑ Simplified message format specifications
- ❑ Program behavior models

Additional Information

- <http://seclab.cs.sunysb.edu/pubs.htm>
 - Automated, sub-second attack signature generation [ACM CCS 2005]
 - Immune-system inspired approach for protection from repetitive attacks [ACSAC 2005]
 - Efficient Techniques for Comprehensive Protection from Memory Error Exploits [USENIX Security 2005]

Related Work

- **Memory corruption prevention and detection**
 - StackGuard, ASR, ISR, CCured – restarts lead to DoS
- **Network level filtering**
 - Shield -- Rely on manual generation of filter rules
- **Source code transformation**
 - Failure-oblivious computing, Automatic patch generation, DIRA
- **Automated worm signature generation**
 - Rely on characteristics of worm behavior
- **FLIPS (PayL), HACQIT**

Summary

- **Adaptive response to large-scale attacks**
 - Automatically discovers the “rules” that distinguish attack-bearing data from normal data
 - Automatically invokes necessary recovery actions by leveraging protected program’s error handling code
- **Benefits**
 - Low overheads
 - Apply generated signature to other systems
 - Preserving service availability
 - Applicable to COTS, no source code required
 - Protection against brute force attacks on randomization techniques