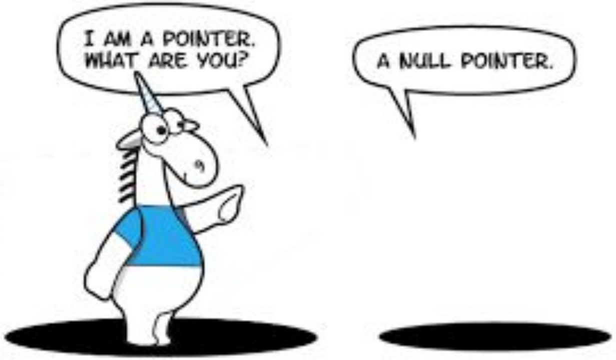
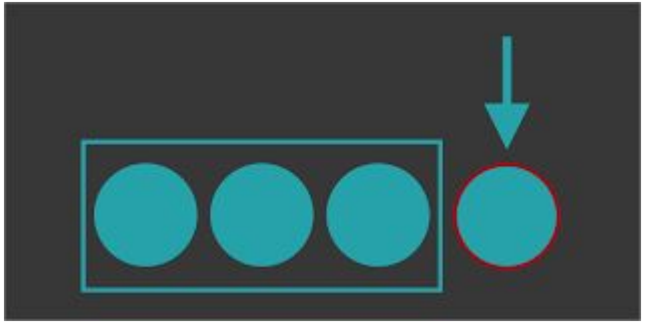
Enhancing Server Availability and Security Through Failure-Oblivious Computing

JunYi Yu

Memory Errors

- Example
  - Invalid pointers
  - Out of bounds memory access
- From
  - Programmers’ faults
  - Attackers
- Damage
  - Termination
  - Infinite loop
  - Unacceptable interaction sequence
  - Data Structure Corruption
  - Unacceptable results





Current Solution

- Safe Languages use dynamic checks to eliminate such errors
    - Intercept and throw an exception
    - Java NullPointerException, IndexOutOfBoundsException
- Unsafe Language
    - Distinguish, and terminate

Failure-Oblivious Computing

- For invalid read
  - Manufacture a value as return value And **continue to execute**
- For invalid write
  - Discard the value And **continue to execute**

Basic Assumption

- Hypothesis
    - At least some programs, this continued execution through memory errors would product acceptable results
    - Simple test to observe the execution of failure-oblivious versions programs
- Acceptable Continued Execution
    - Eliminates the security vulnerabilities
    - Enables the server to successfully execute
- Acceptable Performance
    - Slower but acceptable due to interactive computations.
- Conclusion
    - As long as the server's address space or data structures are not corrupted, continued execution can produce completely acceptable results.

Why failure-oblivious computing works well?

- Availability: Still provide acceptable service through memory errors.
  - As long as the server's address space or data structures are not corrupted, continued execution can produce completely acceptable results.
- Security: Buffer-Overrun Problem
  - Simply discard invalid out-of-bounds write.
  - Attack request ---> anticipated invalid input.
- Acceptable Performance
  - Slower but acceptable due to interactive computations.

Implementation

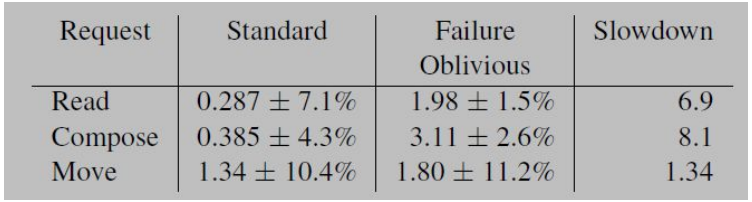
- Checking Code
  - Maintain a table {locations: data_units} to find out-of-bounds pointers
- Continuation Code
  - When checking code detects an attempt to perform illegal access
  - Discard erroneous writes
  - Manufactures a **sequence** of values for erroneous reads
    - Redirect the read to preallocated buffer of values
    - The generated value should work, avoid step into infinite loop.
- Optional logging
  - Track down errors

Experience

- Comparison
  - Standard version
  - Bounds Check version (safe C compiler)
  - Failure Oblivious version
- Security and Resilience
  - Produce input that exploits a security vulnerability, observe the behavior after error.
- Performance
  - Measure the processing time of request
- Stability
  - Use failure oblivious version applications in daily work and try to trigger memory errors in workload.

Pine: A widely used mail user agent

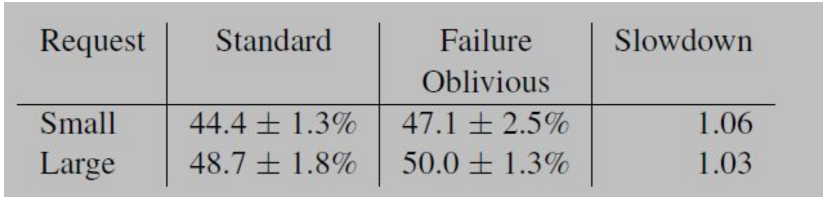
- Memory Error
  - Incorrect calculation for maximum buffer size.
- Security and Resilience
  - C heap corruption, segmentation violation
  - Safe C termination
  - FOC continue to execute, the erroneous form field will be fixed under the hood
- Stability
  - Executed successfully through all errors to perform all requests flawlessly.
- Performance



Request	Standard	Failure Oblivious	Slowdown
Read	0.287 ± 7.1%	1.98 ± 1.5%	6.9
Compose	0.385 ± 4.3%	3.11 ± 2.6%	8.1
Move	1.34 ± 10.4%	1.80 ± 11.2%	1.34

Apache: The most widely used web server

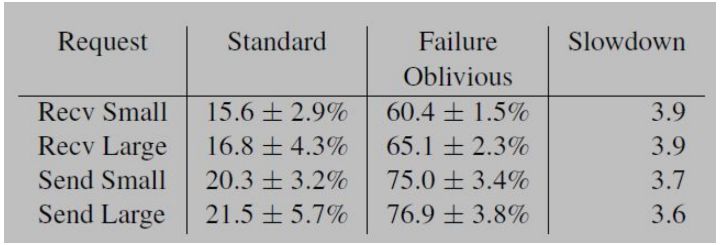
- Memory Error
  - Not enough rooms for captures.
- Security and Resilience
  - C stack corruption, segmentation violation
  - Safe C Create a chile process to handle the error. (Child Process Pool)
  - FOC Child process redirected the attacking request to a non existent URL
- Stability
  - Running for nine months, executed successfully even under some attacks.
- Performance



Request	Standard	Failure Oblivious	Slowdown
Small	44.4 ± 1.3%	47.1 ± 2.5%	1.06
Large	48.7 ± 1.8%	50.0 ± 1.3%	1.03

Sendmail: A standard mail transfer agent

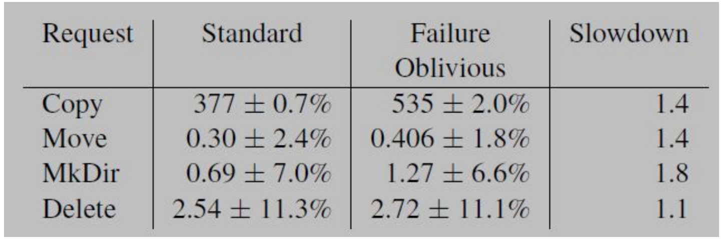
- Memory Error
  - Skip the check to see if the buffer has enough space for lookahead character
- Security and Resilience
  - C stack corruption, attackers can cause the server to execute any injected code
  - Safe C exit with an error message
  - FOC not vulnerable to the attack, the erroneous address will be rejected.
- Stability
  - All of the messages were correctly delivered with no problems.
- Performance



Request	Standard	Failure Oblivious	Slowdown
Recv Small	15.6 ± 2.9%	60.4 ± 1.5%	3.9
Recv Large	16.8 ± 4.3%	65.1 ± 2.3%	3.9
Send Small	20.3 ± 3.2%	75.0 ± 3.4%	3.7
Send Large	21.5 ± 5.7%	76.9 ± 3.8%	3.6

Midnight Commander: A file management tool

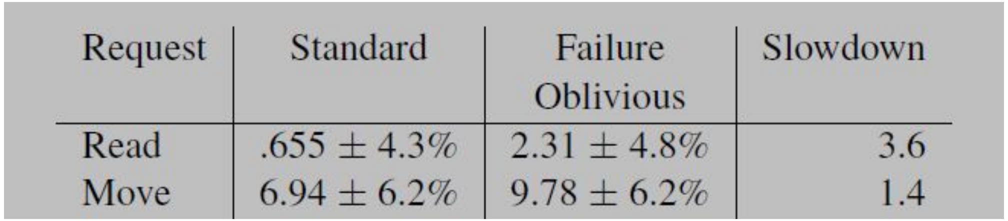
- Memory Error
  - Strcat will write the component names beyond the end of the buffer
- Security and Resilience
  - C stack corruption, segmentation violation
  - Safe C exit with an error message
  - FOC continue to run successfully
- Stability
  - Execute successfully, but by logs we know memory errors were triggered.
- Performance



Request	Standard	Failure Oblivious	Slowdown
Copy	377 ± 0.7%	535 ± 2.0%	1.4
Move	0.30 ± 2.4%	0.406 ± 1.8%	1.4
MkDir	0.69 ± 7.0%	1.27 ± 6.6%	1.8
Delete	2.54 ± 11.3%	2.72 ± 11.1%	1.1

Mutt: A customizable, text-based mail user agent

- Memory Error
  - The buffer for UTF-7 name is not long enough
- Security and Resilience
  - C stack corruption, segmentation violation
  - Safe C exit with an error message
  - FOC continue to run successfully
- Stability
  - Execute all requests correctly
- Performance



Request	Standard	Failure Oblivious	Slowdown
Read	.655 ± 4.3%	2.31 ± 4.8%	3.6
Move	6.94 ± 6.2%	9.78 ± 6.2%	1.4

Discussion

- Failure oblivious computing will ignore memory errors and continue to run. How can we know detailed information about errors? What data should be logged during memory errors?
- Do we really need Failure Oblivious computing?
    - What kind of errors can be ignored during runtime?