Orchestra: Intrusion Detection Using Parallel Execution and Monitoring of Program Variants in User-Space

Babak Salamat, **Todd Jackson**, Andreas Gal, and Michael Franz

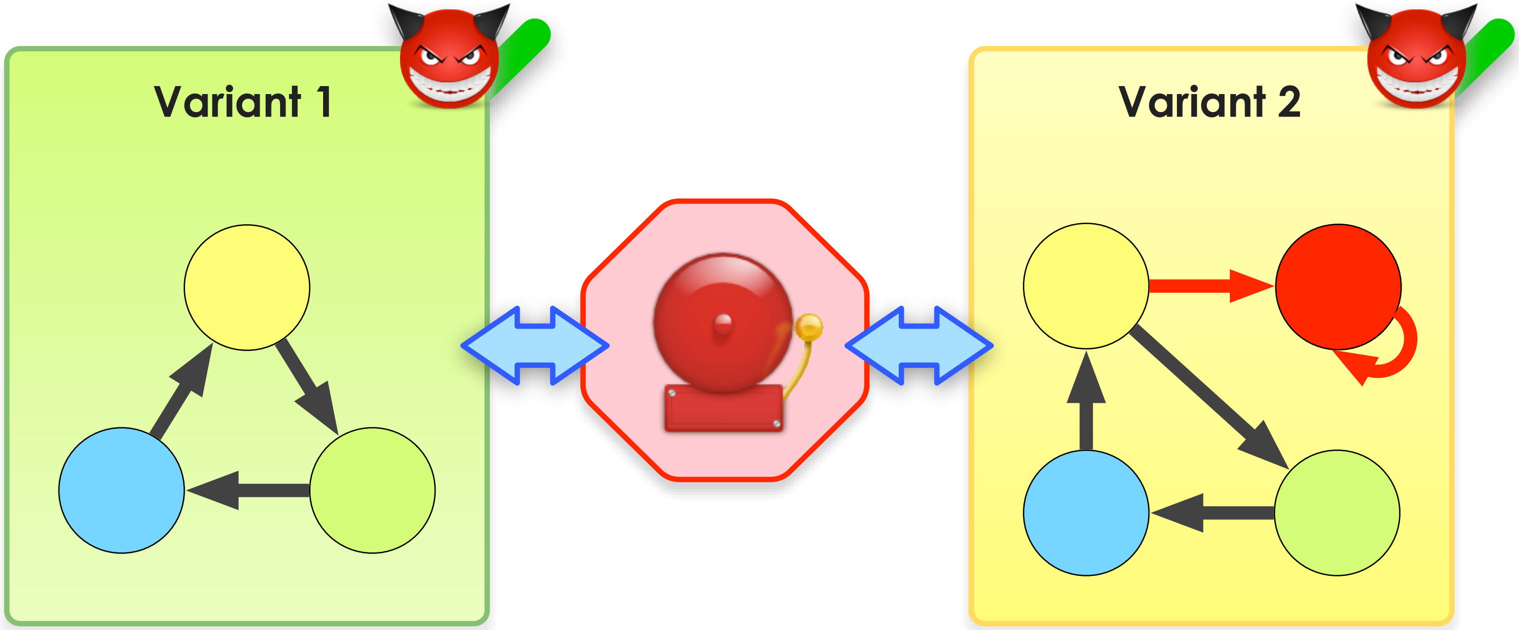
Secure Systems and Languages Laboratory

Department of Computer Science

University of California, Irvine

EuroSys 2009, April 2009

Multi-Variant Execution

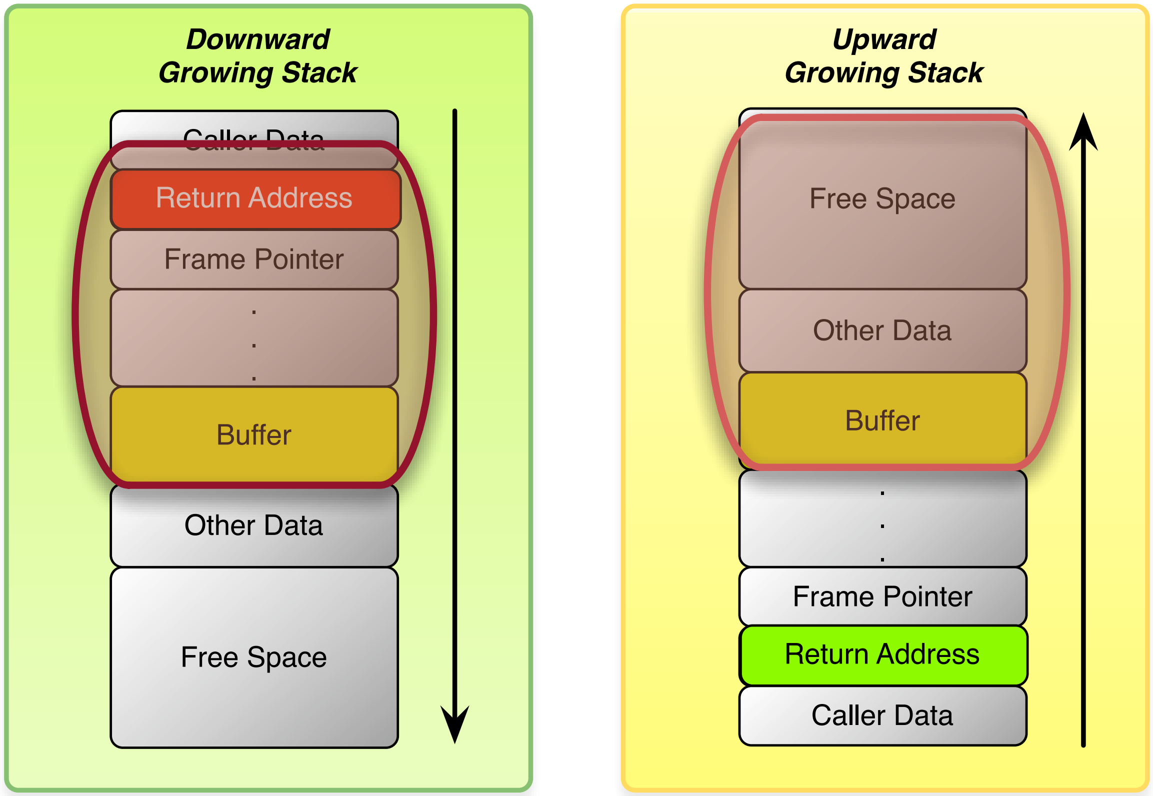


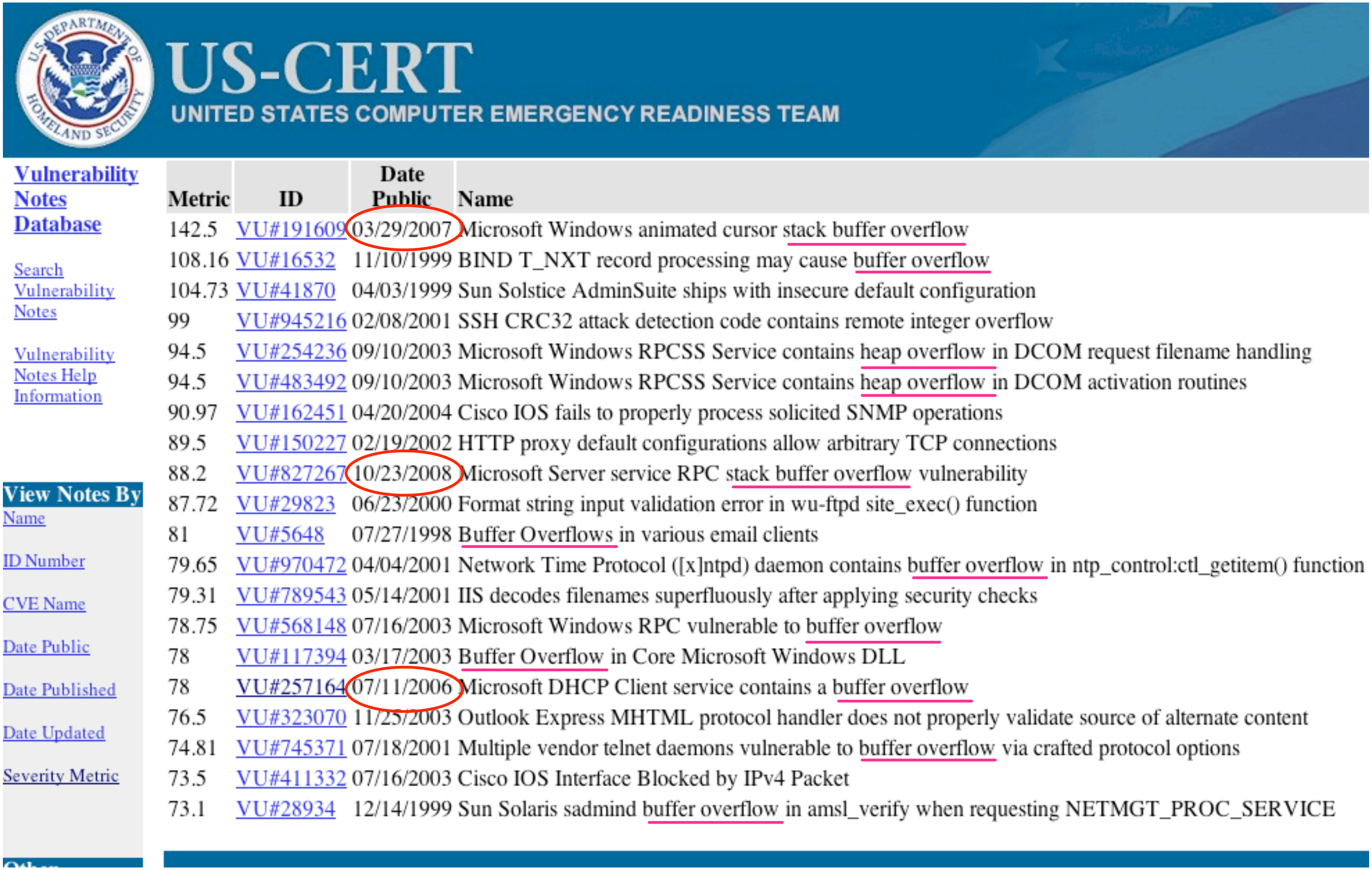
Detection Requirements

- Lock-step execution
- Feed all variants with identical input
- Variants which behave differently when attacked

Reverse Stack Growth Direction

- Stack objects are located in opposite positions





US-CERT
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

Vulnerability Notes Database

Search Vulnerability Notes

Vulnerability Notes Help Information

View Notes By

Name

ID Number

CVE Name

Date Public

Date Published

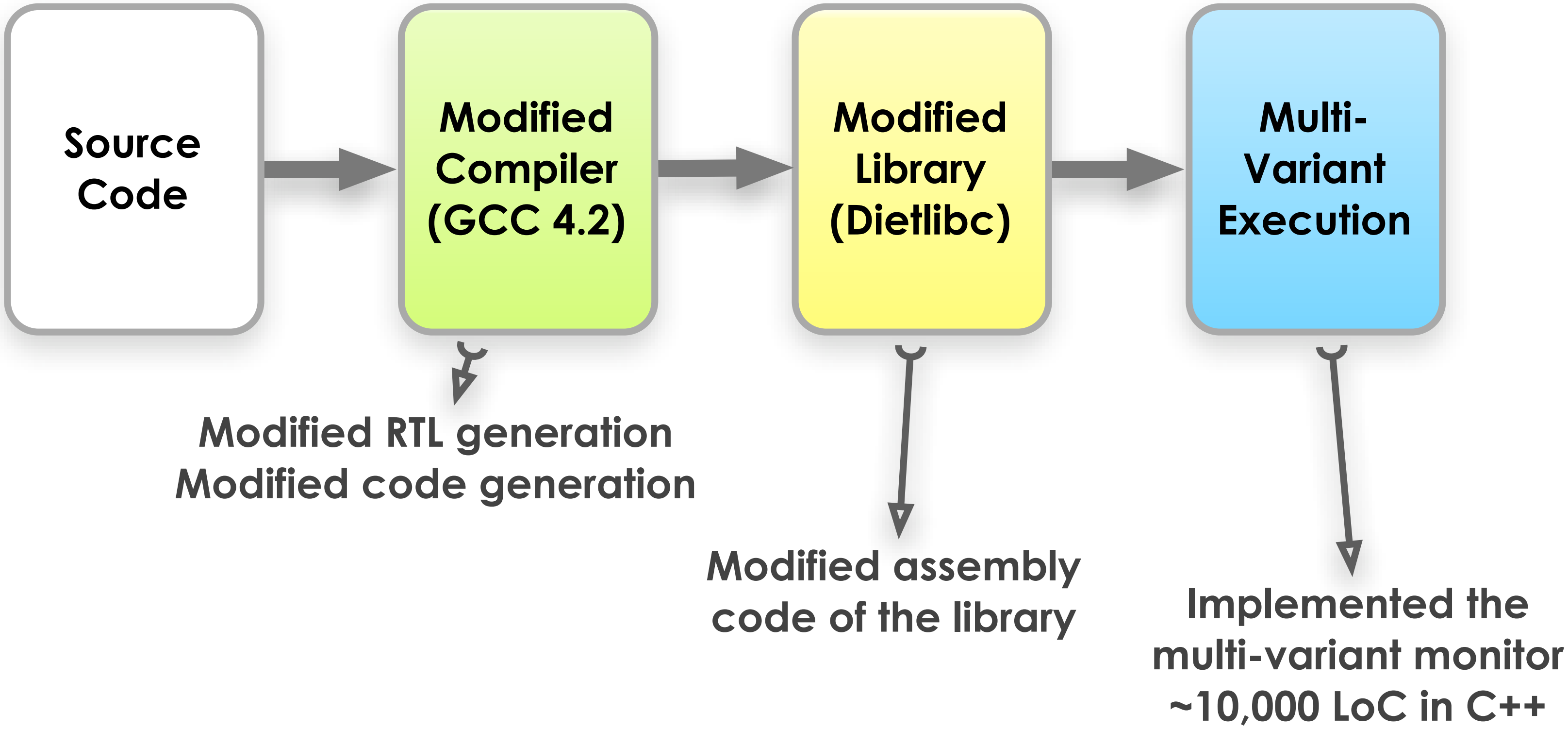
Date Updated

Severity Metric

Other

Metric	ID	Date Public	Name
142.5	VU#191609	03/29/2007	Microsoft Windows animated cursor stack buffer overflow
108.16	VU#16532	11/10/1999	BIND T_NXT record processing may cause buffer overflow
104.73	VU#41870	04/03/1999	Sun Solstice AdminSuite ships with insecure default configuration
99	VU#945216	02/08/2001	SSH CRC32 attack detection code contains remote integer overflow
94.5	VU#254236	09/10/2003	Microsoft Windows RPCSS Service contains heap overflow in DCOM request filename handling
94.5	VU#483492	09/10/2003	Microsoft Windows RPCSS Service contains heap overflow in DCOM activation routines
90.97	VU#162451	04/20/2004	Cisco IOS fails to properly process solicited SNMP operations
89.5	VU#150227	02/19/2002	HTTP proxy default configurations allow arbitrary TCP connections
88.2	VU#827267	10/23/2008	Microsoft Server service RPC stack buffer overflow vulnerability
87.72	VU#29823	06/23/2000	Format string input validation error in wu-ftpd site_exec() function
81	VU#5648	07/27/1998	Buffer Overflows in various email clients
79.65	VU#970472	04/04/2001	Network Time Protocol ([x]ntpd) daemon contains buffer overflow in ntp_control:ctl_getitem() function
79.31	VU#789543	05/14/2001	IIS decodes filenames superfluously after applying security checks
78.75	VU#568148	07/16/2003	Microsoft Windows RPC vulnerable to buffer overflow
78	VU#117394	03/17/2003	Buffer Overflow in Core Microsoft Windows DLL
78	VU#257164	07/11/2006	Microsoft DHCP Client service contains a buffer overflow
76.5	VU#323070	11/25/2003	Outlook Express MHTML protocol handler does not properly validate source of alternate content
74.81	VU#745371	07/18/2001	Multiple vendor telnet daemons vulnerable to buffer overflow via crafted protocol options
73.5	VU#411332	07/16/2003	Cisco IOS Interface Blocked by IPv4 Packet
73.1	VU#28934	12/14/1999	Sun Solaris sadmind buffer overflow in amsl_verify when requesting NETMGT_PROC_SERVICE

From Source to Execution



Source Code

Modified Compiler (GCC 4.2)

Modified Library (Dietlibc)

Multi-Variant Execution

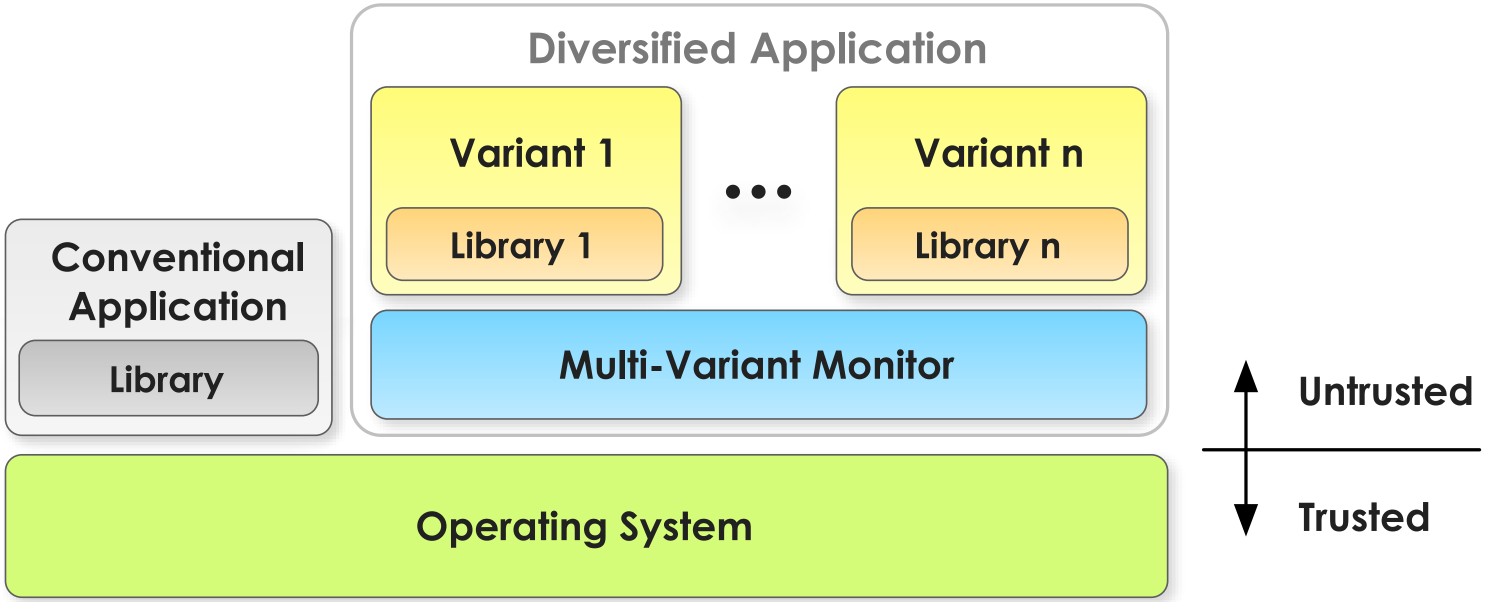
Modified RTL generation
Modified code generation

Modified assembly code of the library

Implemented the multi-variant monitor
~10,000 LoC in C++

Orchestra Architecture

- The monitor is a user-space application

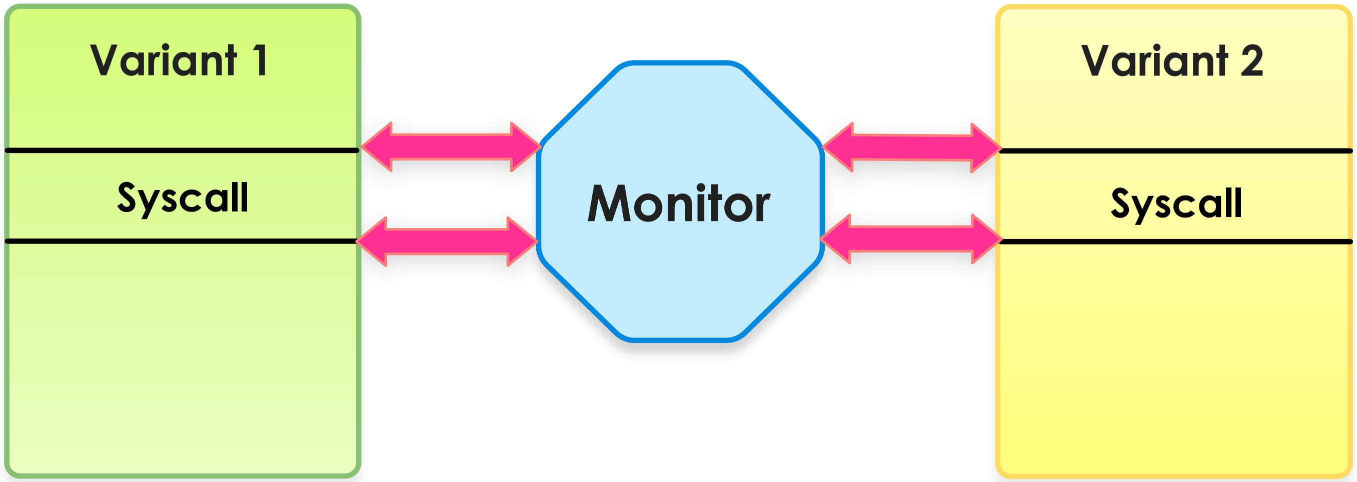


Granularity of Monitoring

- Granularity of monitoring and Synchronization
  - Ideally after each instruction
  - Not always possible
  - Performance issues
- Synchronize and monitor at system calls
  - No harm is done without invoking a system call
  - All instances must invoke the same syscall with equivalent arguments

System Call Monitoring

- Debugging facility of Linux (ptrace) is used to build the monitor
- The monitor is notified twice per system call

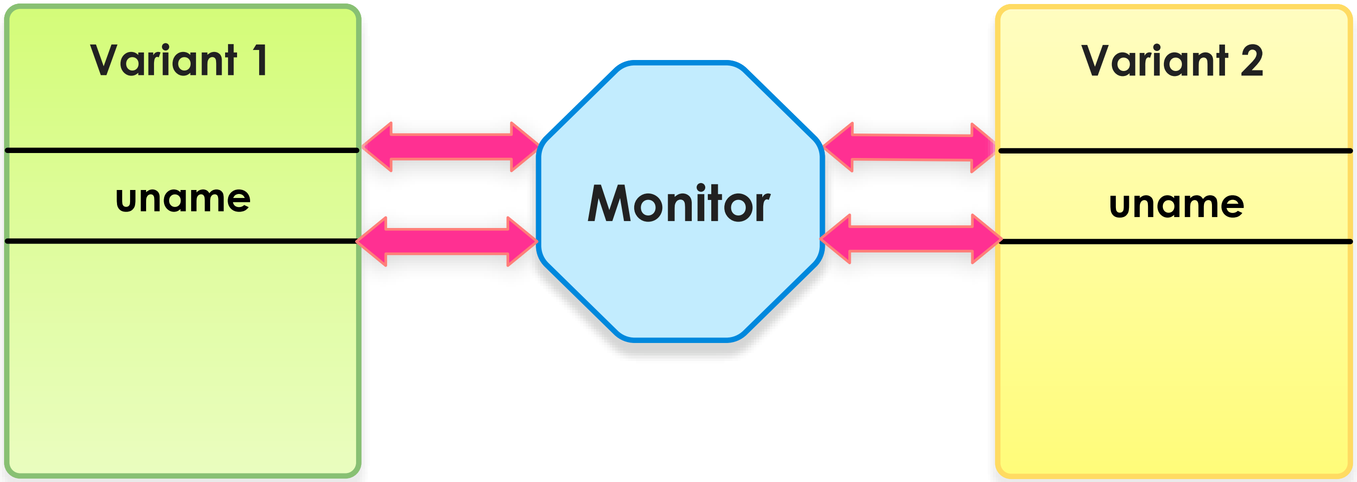


System Call Monitoring (cont.)

- Equivalency is checked at the beginning of a system call
  - The system calls must be the same
  - Arguments must be equivalent
    - Pointers (buffers) have the same content
    - Values are identical
- Results of the system call are written back to the variants at the end of the system call if needed

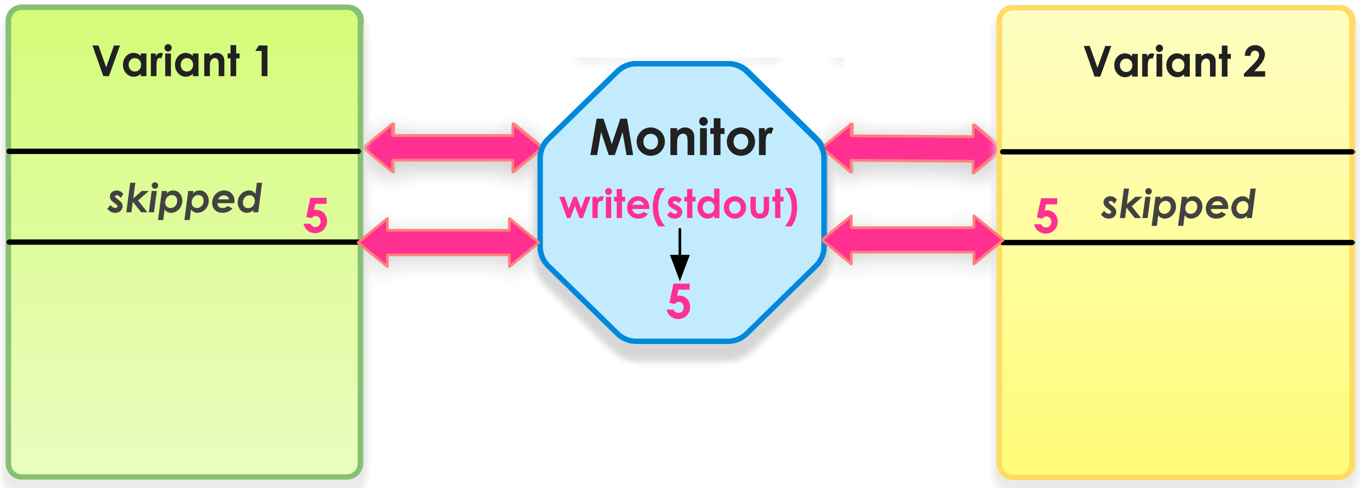
System Call Execution

- Non-state changing system call that produce immutable results are executed by all



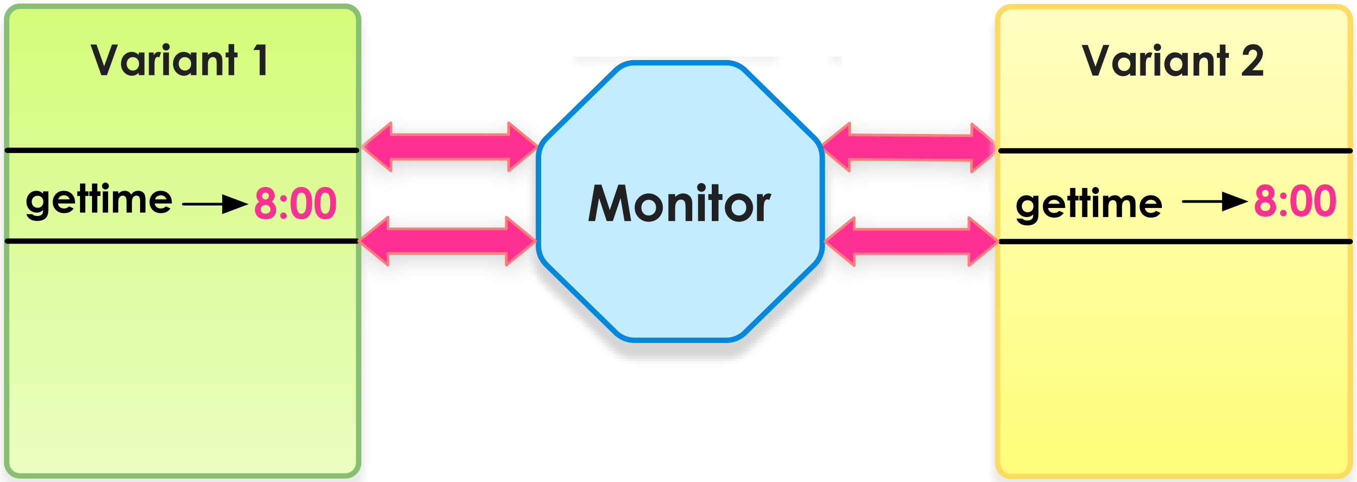
System Call Execution (cont.)

- State changing system calls are executed by the monitor



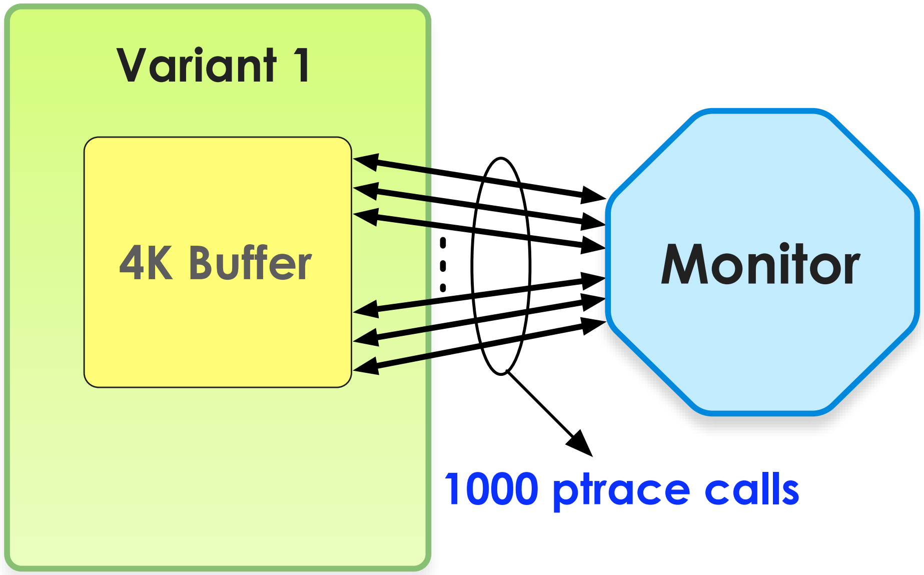
System Call Execution (cont.)

- Non-state changing system call that produce non-immutable results are executed by all, results are copied from the first variant to all



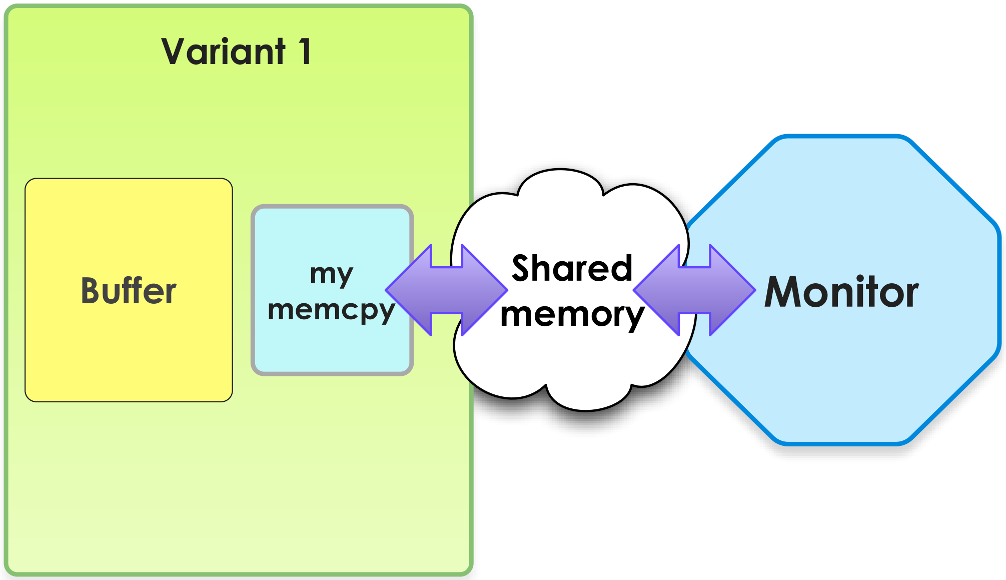
Data Transfer

- ptrace transfers only 4 bytes at a time
  - very slow in transferring large buffers



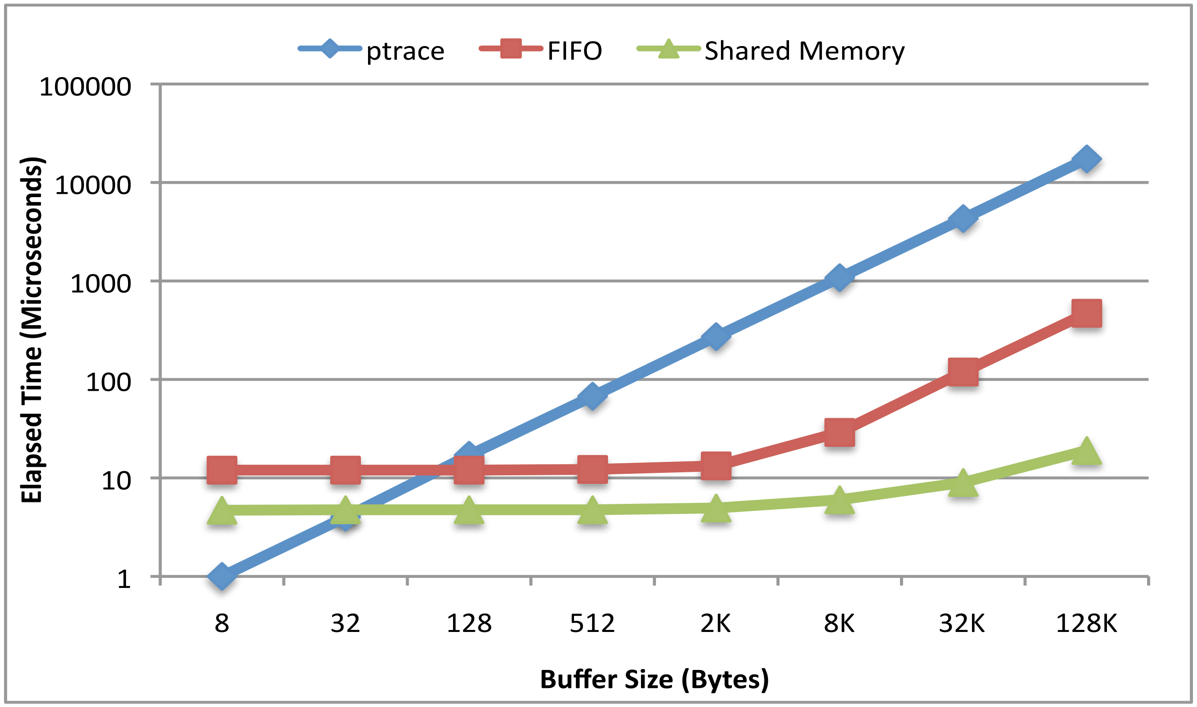
Data Transfer (cont.)

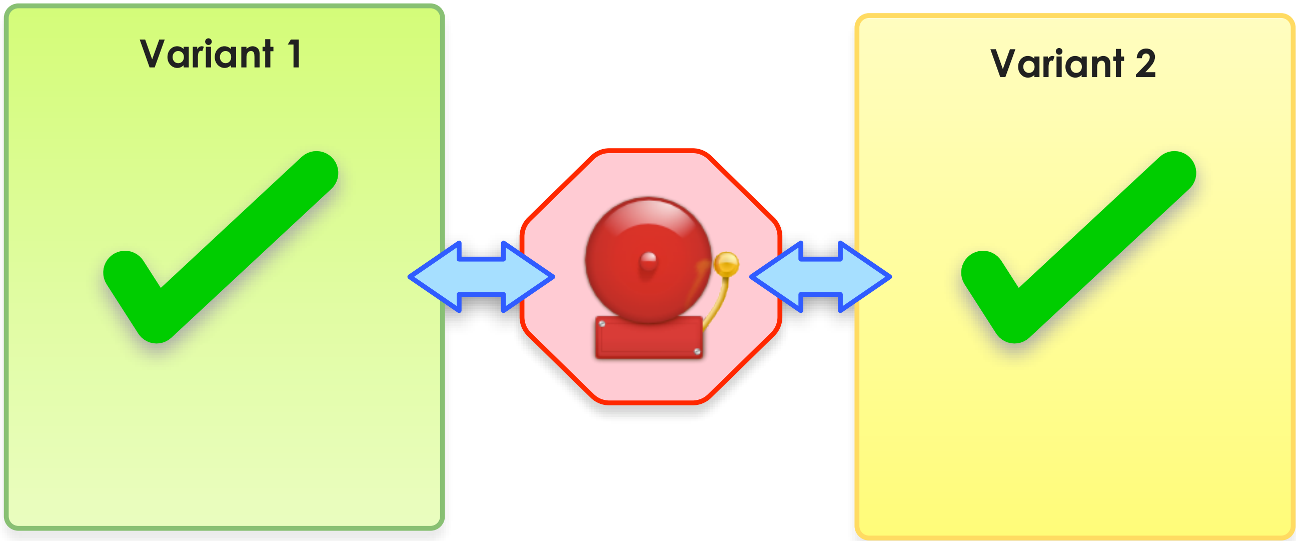
- We tried using named pipes, but they cannot transfer more than 4K bytes at a time
- Shared memory is fast and can transfer mega bytes



Data Transfer Performance

Shared memory is about 1000 times faster than ptrace and 20 times faster than FIFO in transferring a 128K buffer



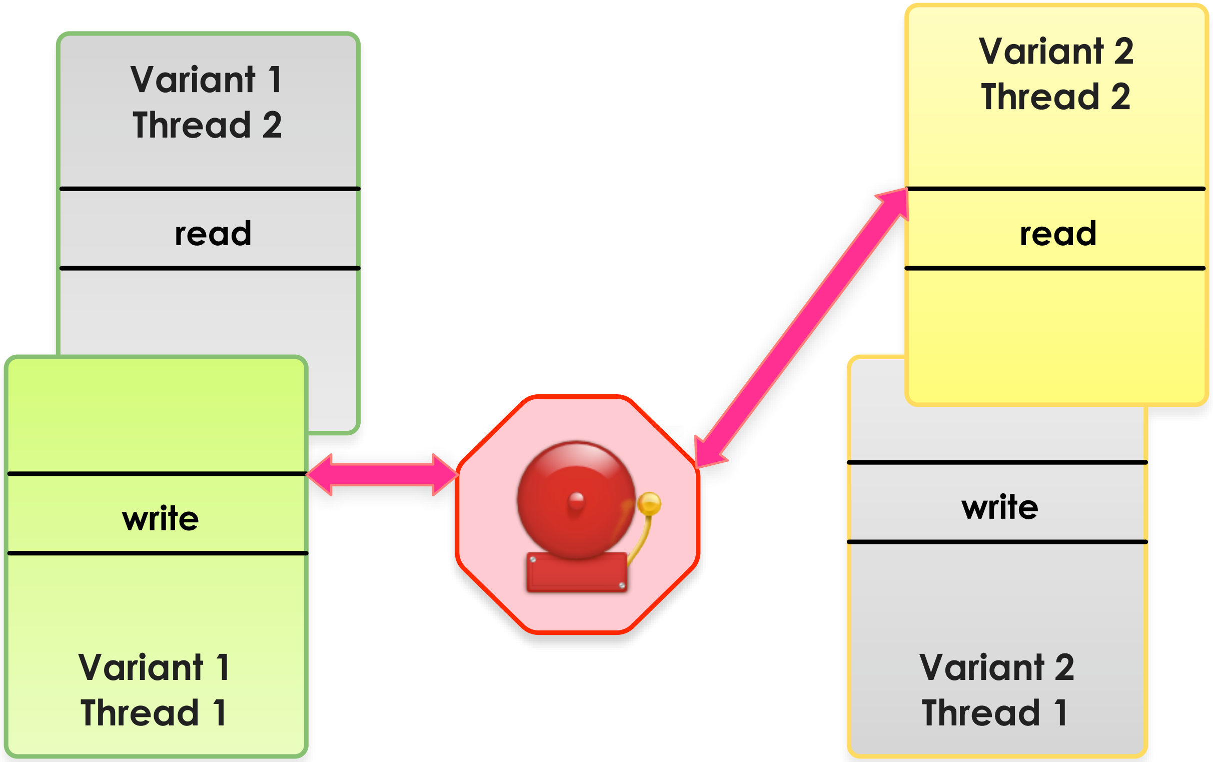


Removing False Positives

False positives are the major practical issue in using multi-variant execution

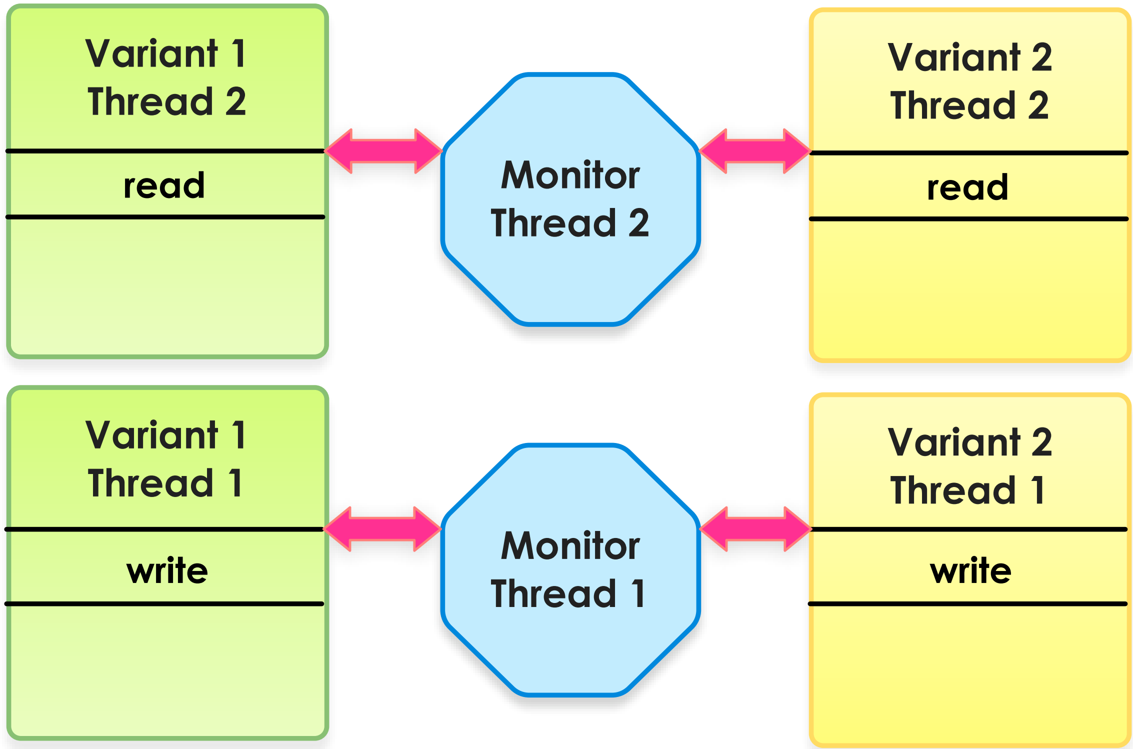
Multi-Threaded Variants

- Different scheduling of multi-threaded or multi-process applications can cause false positives



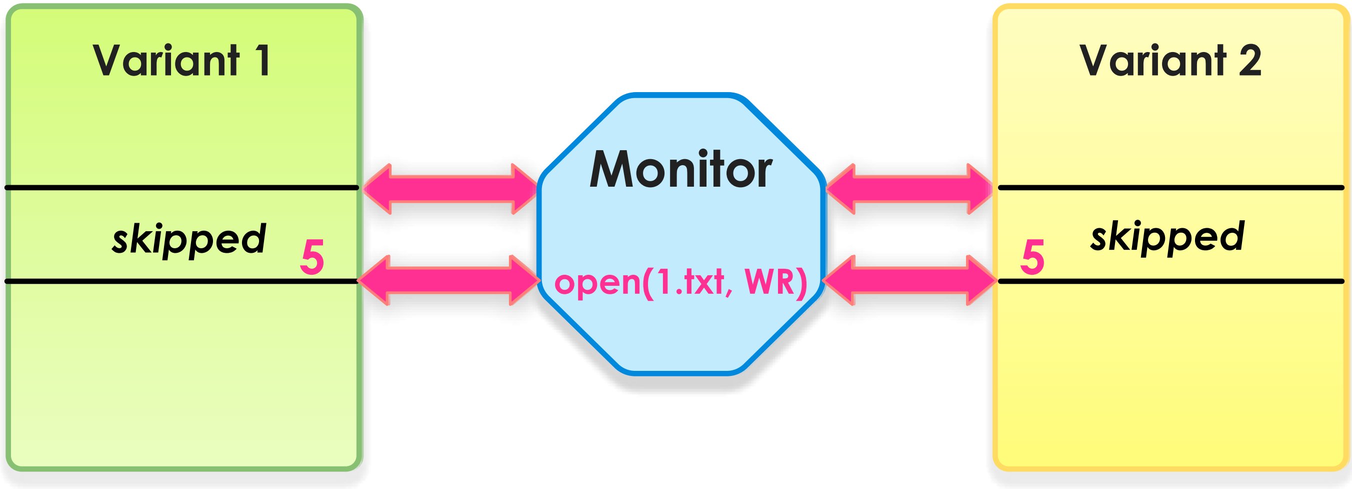
Monitoring multi-threaded variants

- Corresponding threads/processes must be synchronized to each other



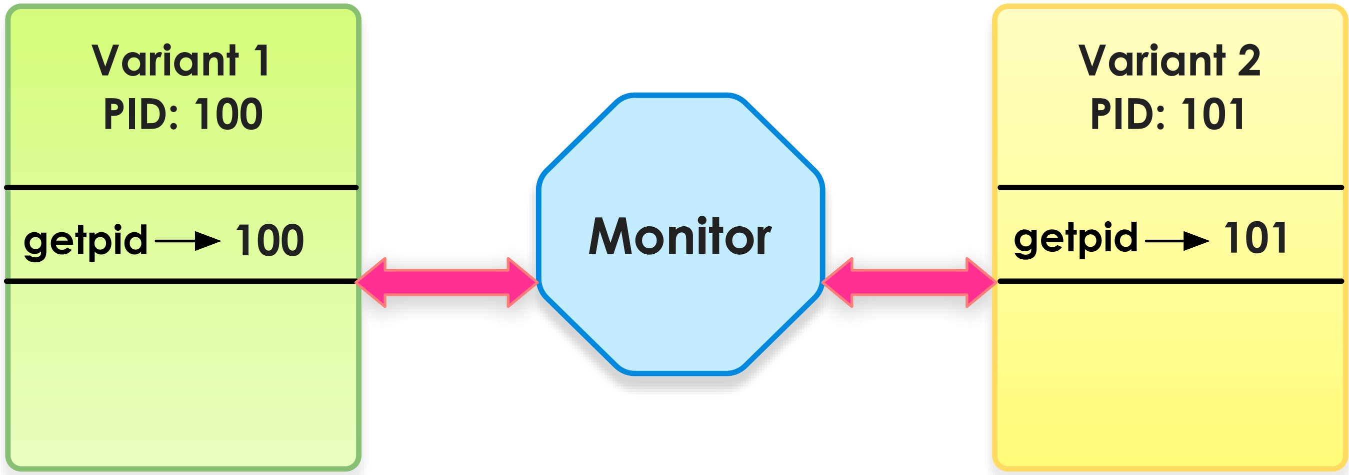
File Descriptors

- The same file descriptor is always reported to all variants when they invoke system calls that return a file descriptor



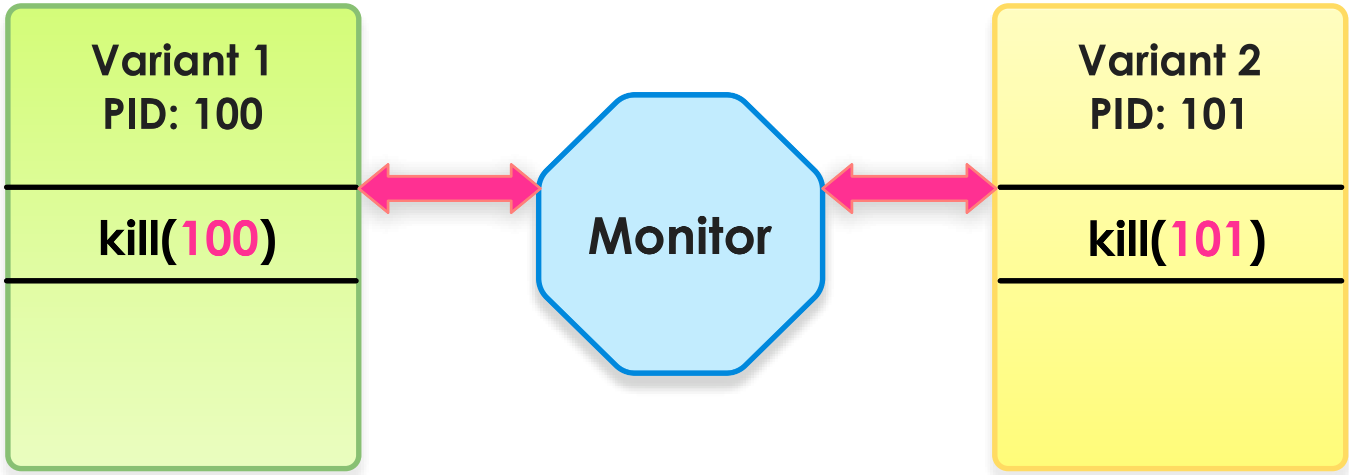
Process ID

- Monitor reports the process ID of the first variant to all
- The PID of the first variant's child process is reported as the result of `fork` or `clone` to all the variants



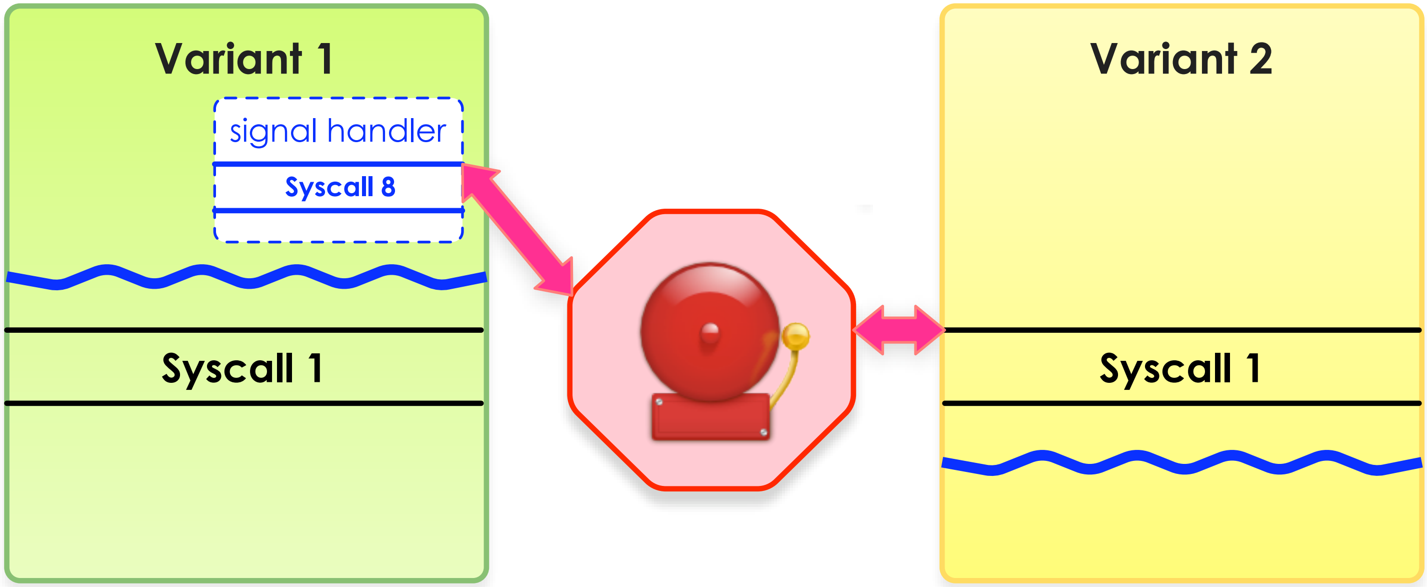
Process IDs in Arguments

- When variants need to run a system call that receives a PID, appropriate PID is restored before the execution of the system call



Asynchronous Signals

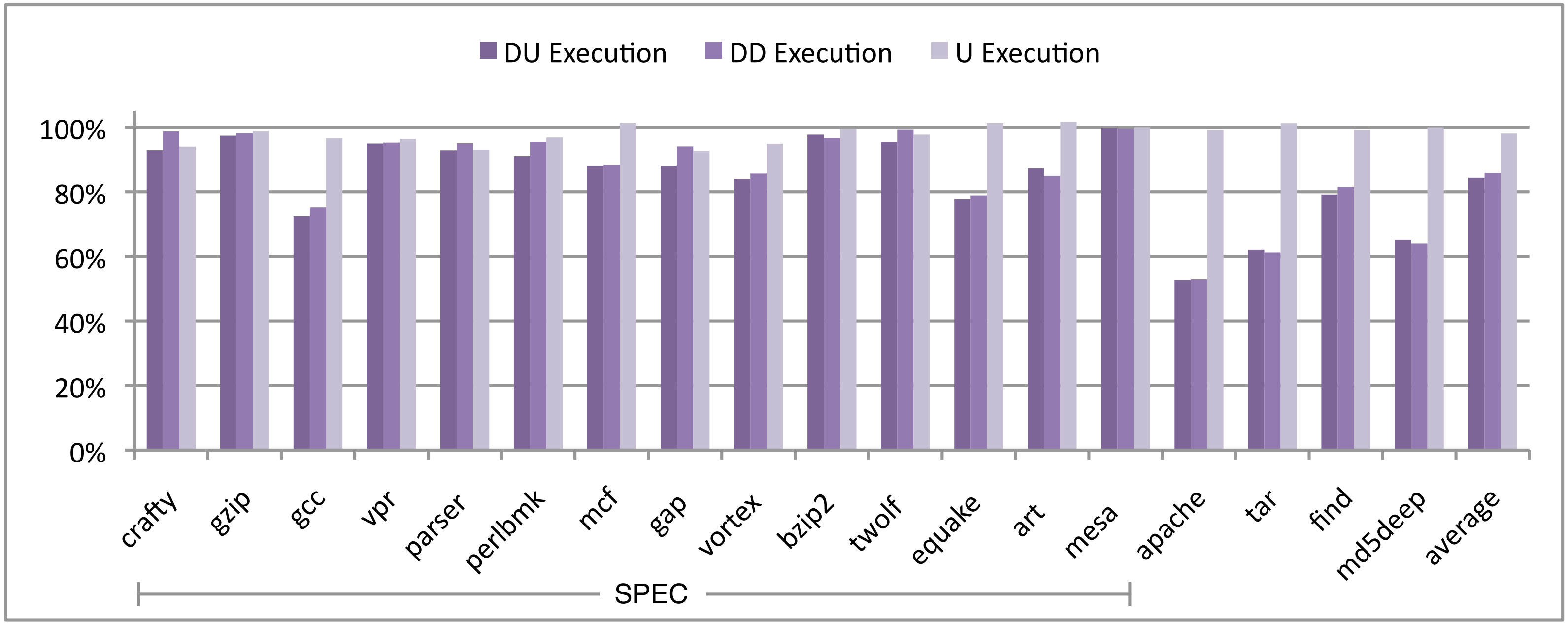
- Signal handlers can cause different sequences of system calls to be executed by the variants



Time and Random Numbers

- System calls that read time (e.g., gettimeofday) are executed by one variant and the result is copied to all
- By providing identical time and other system information to all variants, they likely use the same seed to generate random numbers
- The monitor reads /dev/urandom and copies the result to all variants
- Reading CPU time stamp counters (RDTSC) may still cause false positives

Performance



Summary

- Multi-variant execution is an effective technique in detecting and disrupting attacks
- A reverse stack executable can prevent stack-based buffer overflow vulnerabilities in a multi-variant environment
- Our methods can remove most sources of false positives in multi-variant execution
- Running two parallel variants have about 15% overhead

Thank you

Questions?