Fuzzing System Call Return Values

Sripradha Karkala, Kavin Mani

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Motivation

Test the robustness and reliability of applications

Stress on the importance of good programming practices

Fuzz testing - Interesting approach to unearth bugs

Implementation

Library interposition technique
Wrapper for 13 system calls

Class	System Calls
File Management	open, read, write, seek, dup2
Memory management	malloc, calloc, realloc, brk, sbrk
Network Operations	socket, accept, connect

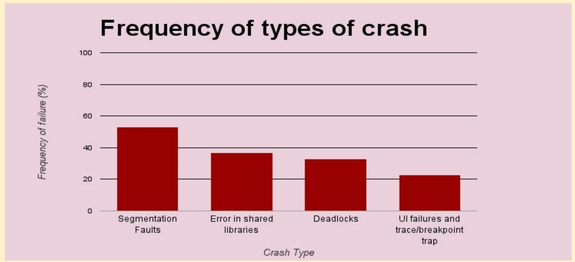
Dynamically link applications to rewritten library

```
int open(const char * file, int oflag) {
// initialization and logging information
// If random seed > failure threshold
return -1
//else
return open(file, oflag); // call to the original function - open()
}

LD_PRELOAD=/path_to_new_lib/open.so /bin/ls
```

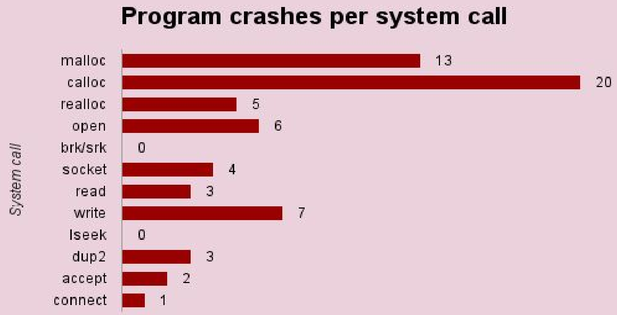
Results

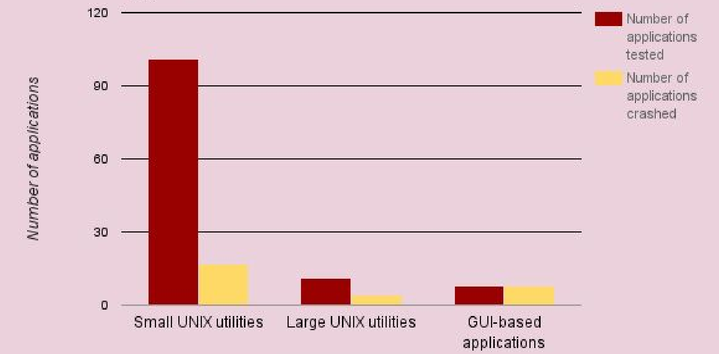
Total number of programs tested : 120



Program	Segmentation fault	Errors in shared library	Deadlock	UI failures and trace/breakpoint trap
netstat	✔			
ps	✔			
lsmod	✔			
gdb	✔			
cancel	✔			
cpan	✔			
corelist	✔			
dirsplit	✔			
ifconfig	✔			
vi		✔		
vim		✔	✔	
telnet	✔			
gedit		✔	✔	✔
sort	✔			
wish		✔	✔	
hostnamectl	✔			
host	✔			
ghostscript	✔			
evince		✔	✔	✔
dm-tool				✔
eog		✔	✔	✔
dig	✔			
checkbox-gui		✔	✔	
loginctl	✔			
firefox				✔
bitmap				✔
calendar		✔	✔	
cheese		✔	✔	✔
red		✔	✔	
less		✔	✔	

Analysis





Reasons for Crashes

Segmentation faults
Errors in shared libraries
Deadlocks
UI loads incorrectly with missing features
Trace/breakpoint traps in programs