# Improving Application Software Security in Linux

Sebastian Neubauer
Technische Universität München
Computer Science Department

July 19, 2017

# Improve Security on Linux

- C/C++ applications contain bugs
- Existing security mechanisms
- Still many ways for exploitation
- Close them!

# Improve Security on Linux

- C/C++ applications contain bugs
- Existing security mechanisms
- Still many ways for exploitation
- Close them!
- Problem: Performance loss
- We need to be fast!

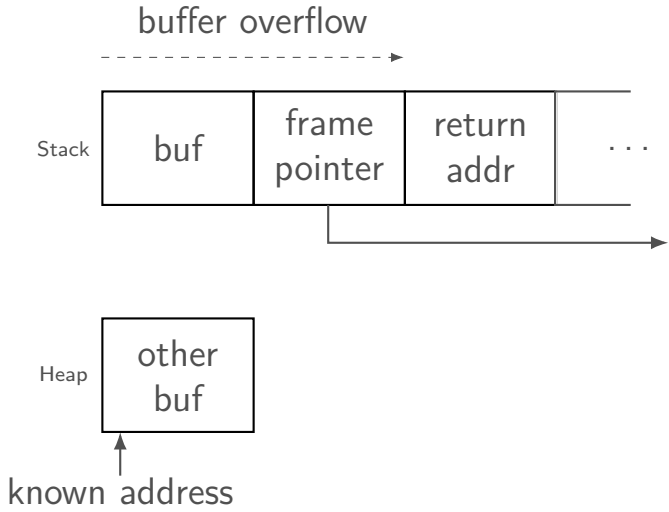
Contributions

- `mmap` randomization: Add random gaps between `mmap` allocations
- Canaries: Clear after use and random values
- Stack pinning: Check the address of the stack pointer

# Exploit: Stack pivoting

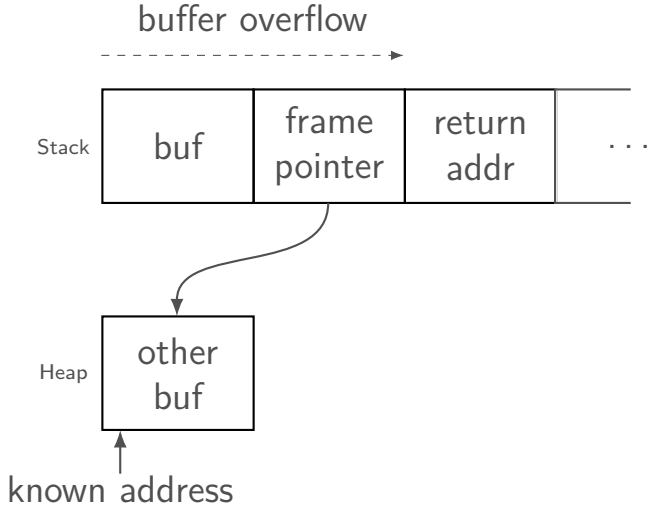
# Exploit

## Stack pivoting



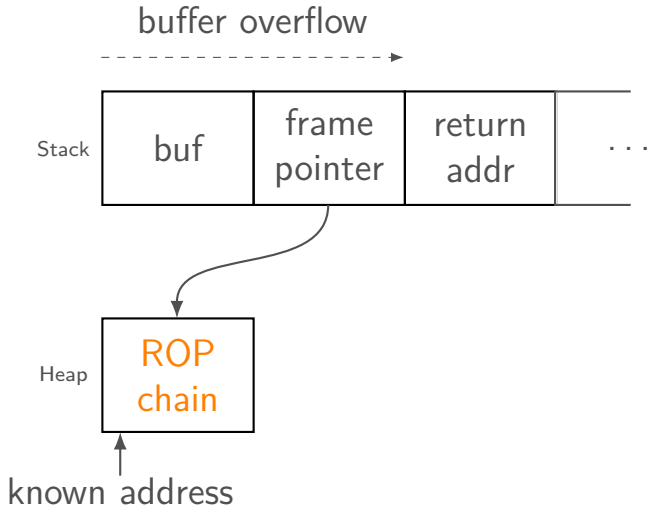
# Exploit

## Stack pivoting



# Exploit

## Stack pivoting



# Idea

## Stack pinning

- Check if the stack pointer points to the stack region

# Idea

## Stack pinning

- Check if the stack pointer points to the stack region
- Almost every exploit arrives at a `syscall`
- Check the stack pointer in every sytem call
- Save stack bounds in the kernel `task_struct` (for each process/thread)

# Pitfalls

## Stack pinning

- Forks, new threads
- Alternate signal stack
- Main stack can grow

# Pitfalls

## Stack pinning

Wine and Go

- Stack pivoting as a Feature

# Pitfalls

## Stack pinning

Wine and Go

- Stack pivoting as a Feature

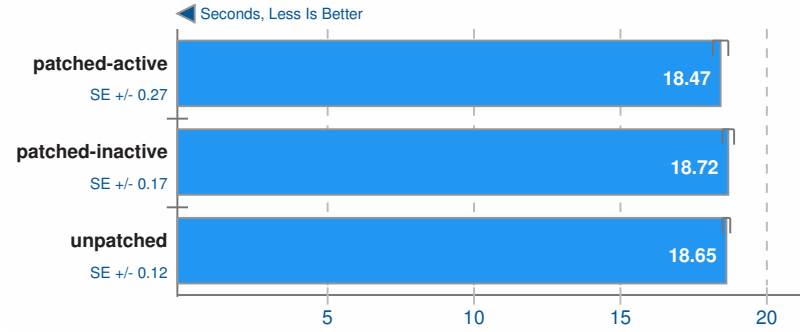
$\Rightarrow$ Only *opt-in* possible

- Save the current memory area as stack area
  `prctl(PR_PIN_STACK, ...)`

# Performance

# Performance

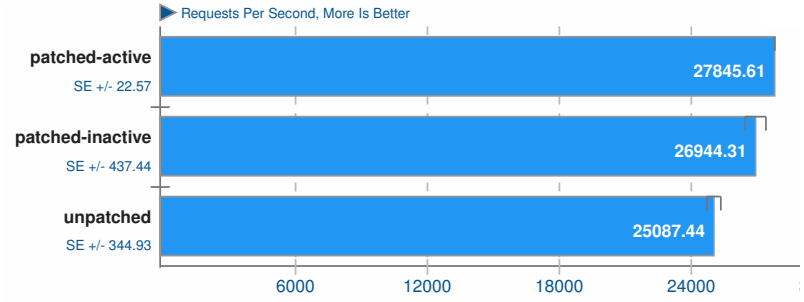
## Stack pinning




- Microbenchmark: $(1 \pm 2)\,\%$ difference

# Performance

## Stack pinning




- Microbenchmark: $(1 \pm 2)\,\%$ difference
- ApacheBench: $(11 \pm 2)\,\%$ more requests per second ☺

# Demo

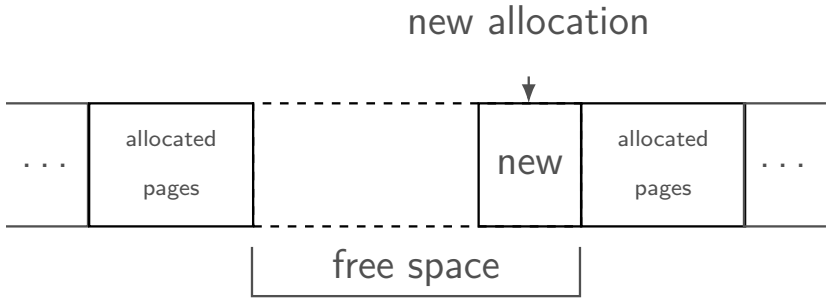
# Problem: `mmap` is deterministic

# Problem

## Deterministic `mmap`

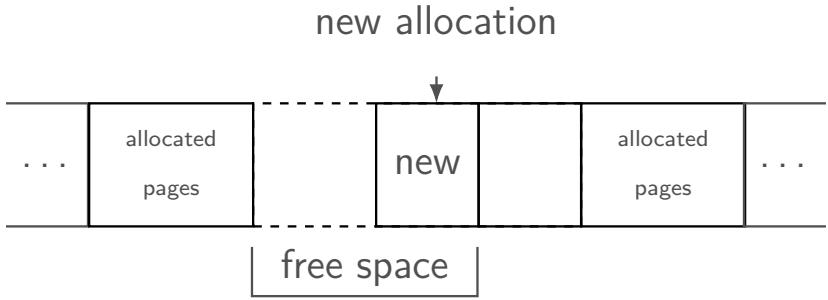
# Problem

## Deterministic `mmap`



# Problem

## Deterministic `mmap`

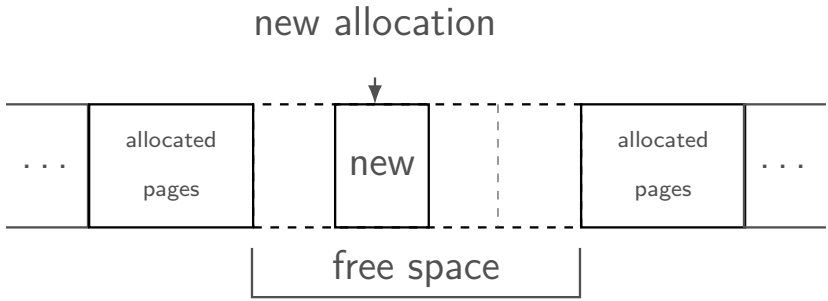


# Idea

## Random `mmap`


# Idea

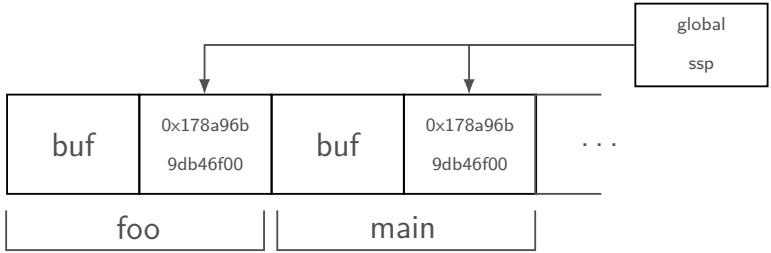
## Random `mmap`



# Problem: Canaries are static

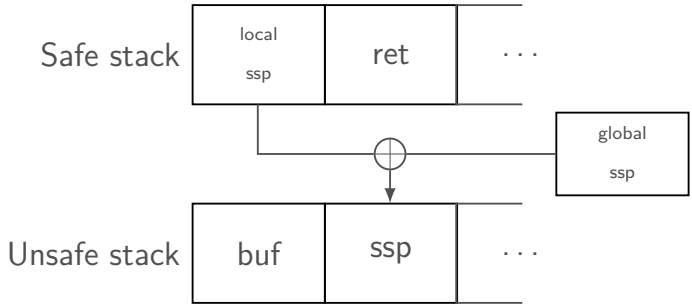
# Problem

## Static canaries



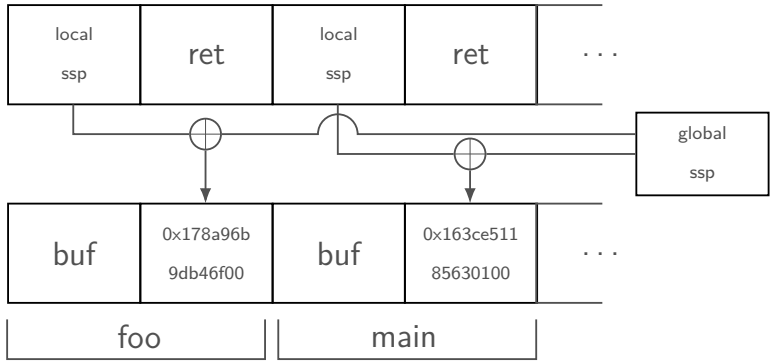
# Idea

## Random canaries



# Idea

## Random canaries



# Summary

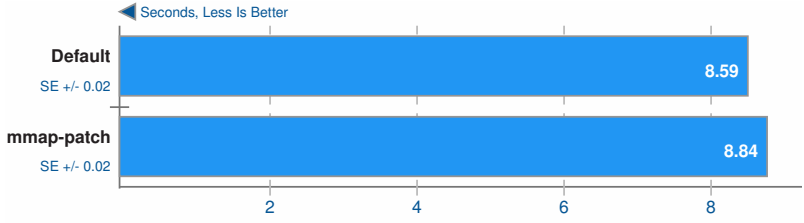
- 3 fast additions, which make exploiting harder
- Goal: Make attacking harder with low overhead

# Summary

- 3 fast additions, which make exploiting harder
- Goal: Make attacking harder with low overhead
- Propose the patches mainline

# Performance

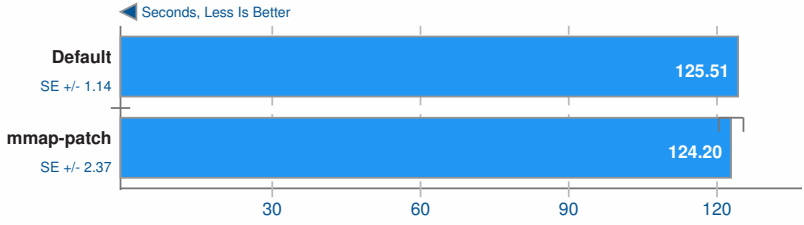
## `mmap`




- Microbenchmark: $(2.8 \pm 0.5)\,\%$ slower

# Performance

`mmap`




- Microbenchmark: $(2.8 \pm 0.5)\,\%$ slower
- Linux compilation: $(1 \pm 3)\,\%$ faster