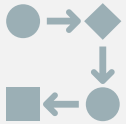
EXPLOIT MITIGATIONS

CS 361S

Spring 2020

Seth James Nielson

DEALING WITH CONTROL FLOW VIOLATIONS



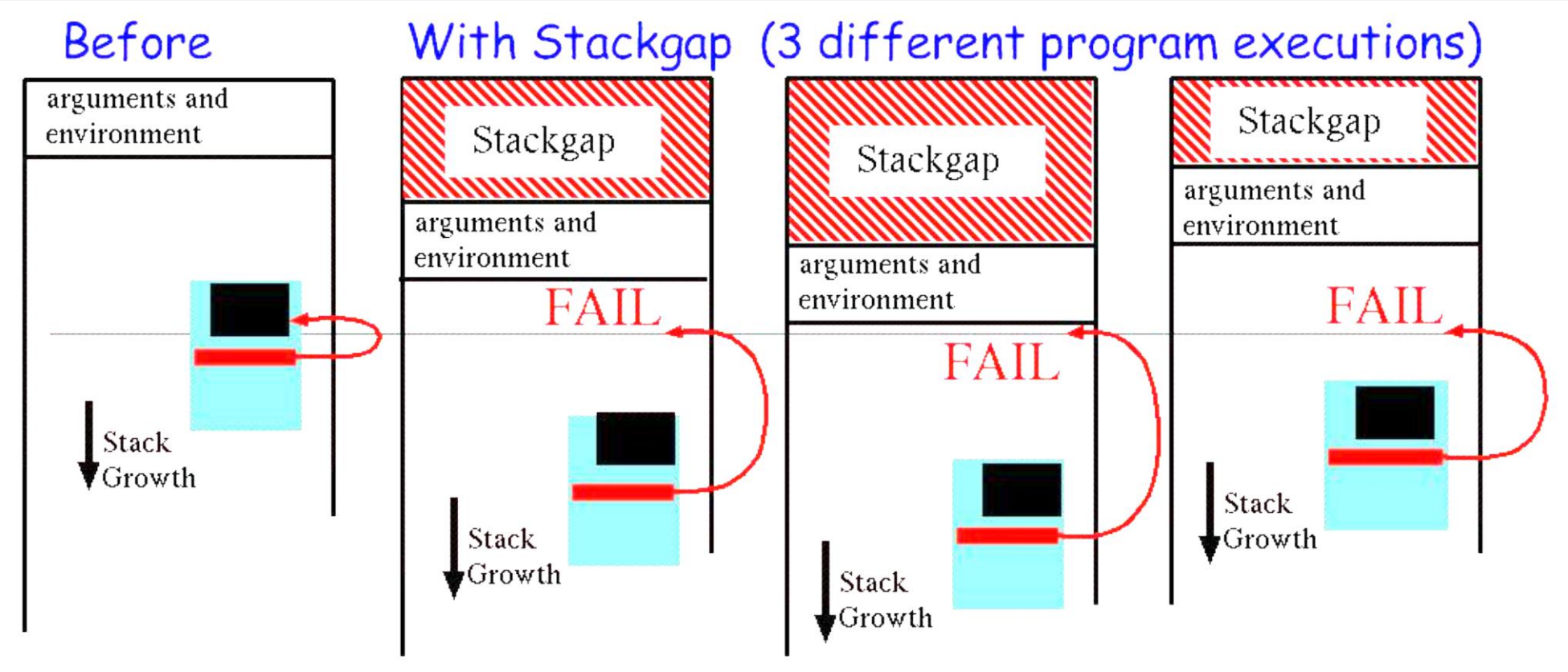
Make it harder to control a subverted flow

Make taking control of the flow innocuous

Make it harder to get control of the flow

DISRUPTING EXPLOITATIVE OPERATIONS

RANDOM STACK GAP

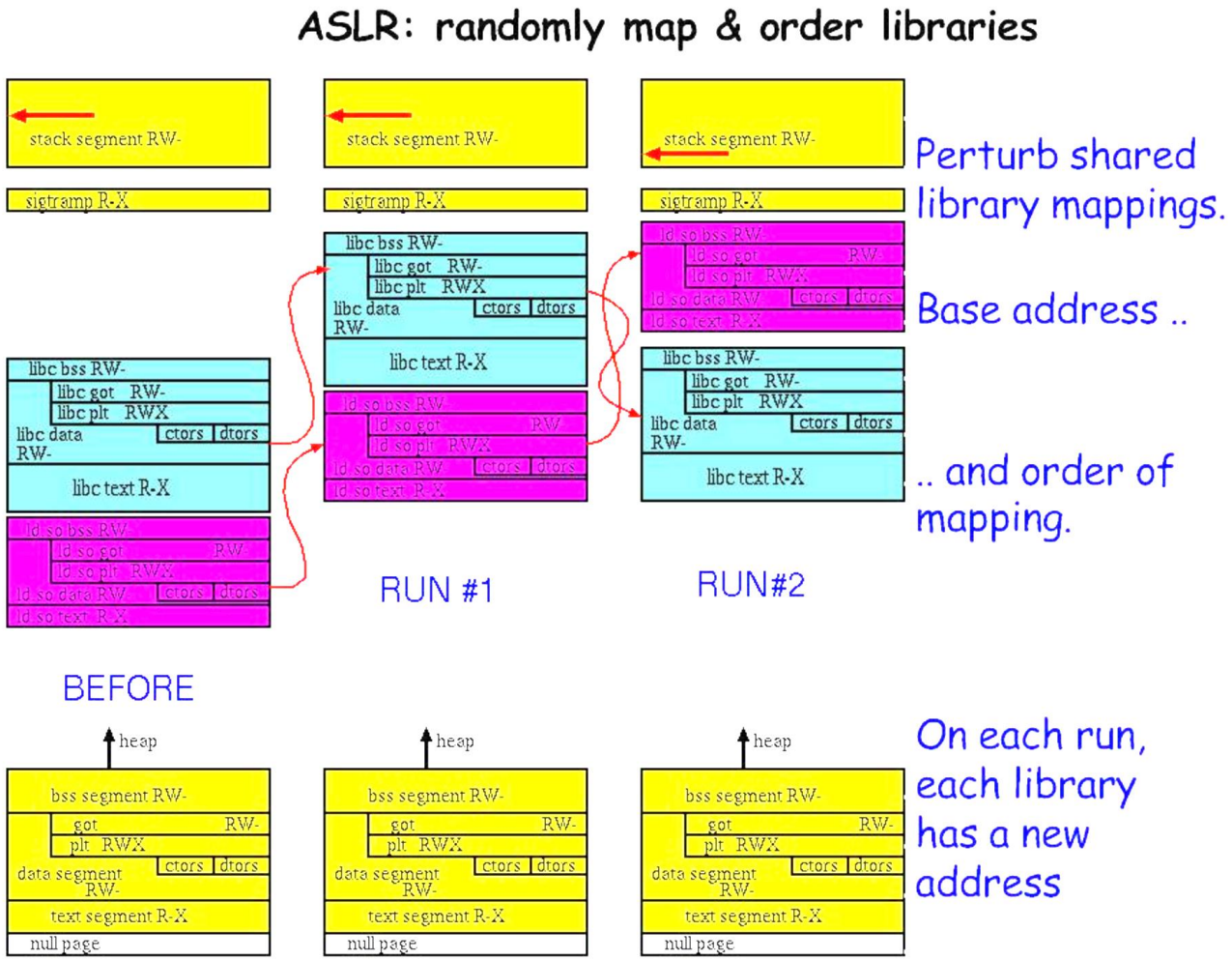


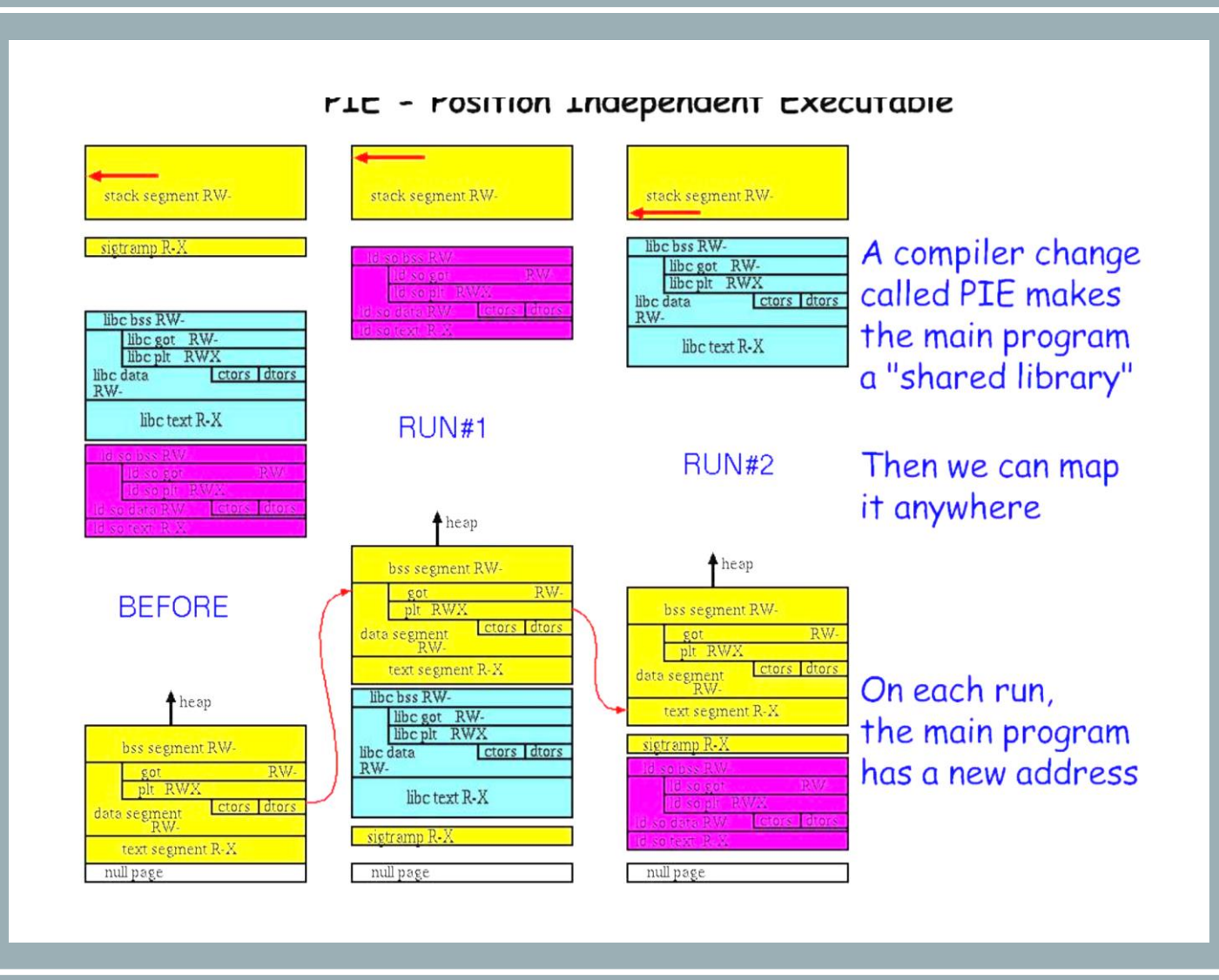
ASLR

Address Space Layout Randomization

- Subversion usually needs to know memory layout
- General goal: make layout unpredictable

START WITH LIBRARIES





ADD EXECUTABLES

FINALLY, DYNAMIC ALLOCATIONS

mmap

malloc

LIMITATIONS OF ASLR

1. **Boot-time based randomization**
2. **Unsupported executables/libraries, low-entropy.**
3. **ASLR does not *trap* the attack**
4. **ASLR does not alert in a case of an attack**
5. **ASLR does not *provide information* about an attack**
6. **ASLR is being bypassed by exploits daily**

Posted by **MORDECHAI GURI, PH.D.** on December 17, 2015

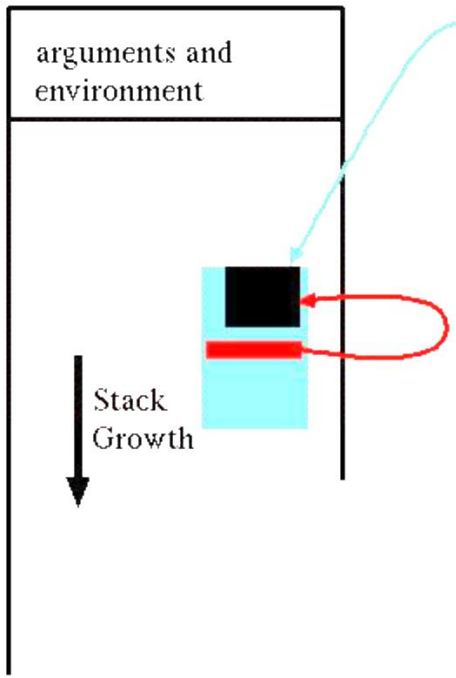
MAKING VIOLATIONS LESS DANGEROUS

W^X Permissions

rodata

W|X PERMISSIONS

Many bugs are exploitable because the address space has memory that is both writeable and executable (permissions = W | X)



this location has to be executable for the exploit to work

We could make the stack non-executable...

Hmmmm... how about a generic policy for the whole address space:

A page may be either writeable or executable, but not both (unless the program specifically requests)

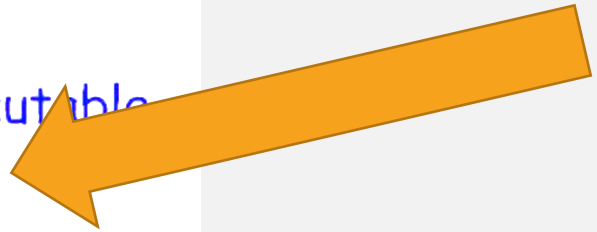
We call this policy W ^ X (W xor X)

EXECUTABLE STACKS

This is what static executables used to look like in memory.

The stack has a piece of executable called the "signal trampoline"

First problem: The stack is executable



What is this?

5.6 Returning from a signal handler

When the program was interrupted by a signal, its status (including all integer and floating point registers) was saved, to be restored just before execution continues at the point of interruption.

This means that the return from the signal handler is more complicated than an arbitrary procedure return - the saved state must be restored.

To this end, the kernel arranges that the return from the signal handler causes a jump to a short code sequence (sometimes called *trampoline*) that executes a `sigreturn()` system call. This system call takes care of everything.

In the old days the trampoline lived on the stack, but nowadays (since 2.5.69) we have a trampoline in the vsyscall page, so that this trampoline no longer is an obstacle in case one wants a non-executable stack.

LINUX TRAMPOLINE?

LINUX TRAMPOLINE!!!

No-execute stacks [edit]

Some implementations of trampolines cause a loss of no-execute stacks (NX stack). In the GNU Compiler Collection (GCC) in particular, a nested function builds a trampoline on the stack at runtime, and then calls the nested function through the data on stack. The trampoline requires the stack to be executable.

No execute stacks and nested functions are mutually exclusive under GCC. If a nested function is used in the development of a program, then the NX stack is silently lost. GCC offers the `-Wtrampoline` warning to alert of the condition.

Software engineered using secure development lifecycle often do not allow the use of nested functions due to the loss of NX stacks.[11]

.wikipedia.org/wiki/Trampoline_(computing)#No-execute_stacks

THE .RODATA SEGMENT

W^X Transition: The .rodata segment

Readonly strings and pointers were stored in the .text segment: X | R

Meaning const data could be executed (could be code an attacker could use as ROP payload)

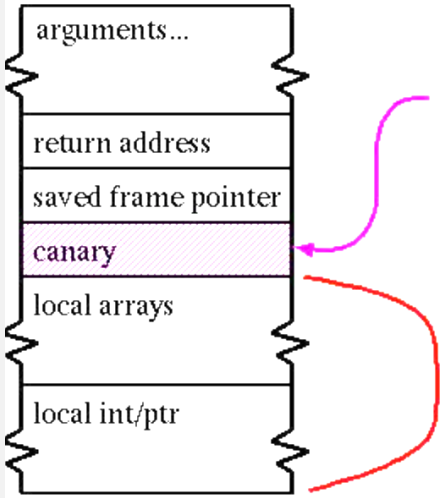
Solution: start using the ELF .rodata segment

These objects are now only R, lost their X permission

Greater policy: "minimal set of permissions"

FINALLY, BLOCKING EXPLOITS

Stack Protector



A typical stack frame...

Random value is inserted here by function prologue ...

... and checked by function epilogue

Reordering: Arrays (strings) placed closer to random value -- integers and pointers placed further away

-fstack-protector-all compiled system is 1.3% slower at make build