Protecting Memory

- What is there to protect in memory?
- Page tables and virtual memory protection
- Special security issues for memory
- Buffer overflows

What Is In Memory?

- Executable code
  - Integrity required to ensure secure operations
- Copies of permanently stored data
  - Secrecy and integrity issues
- Temporary process data
  - Mostly integrity issues

Mechanisms for Memory Protection

- Most general purpose systems provide some memory protection
  - Logical separation of processes that run concurrently
- Usually through virtual memory methods
- Originally arose mostly for error containment, not security

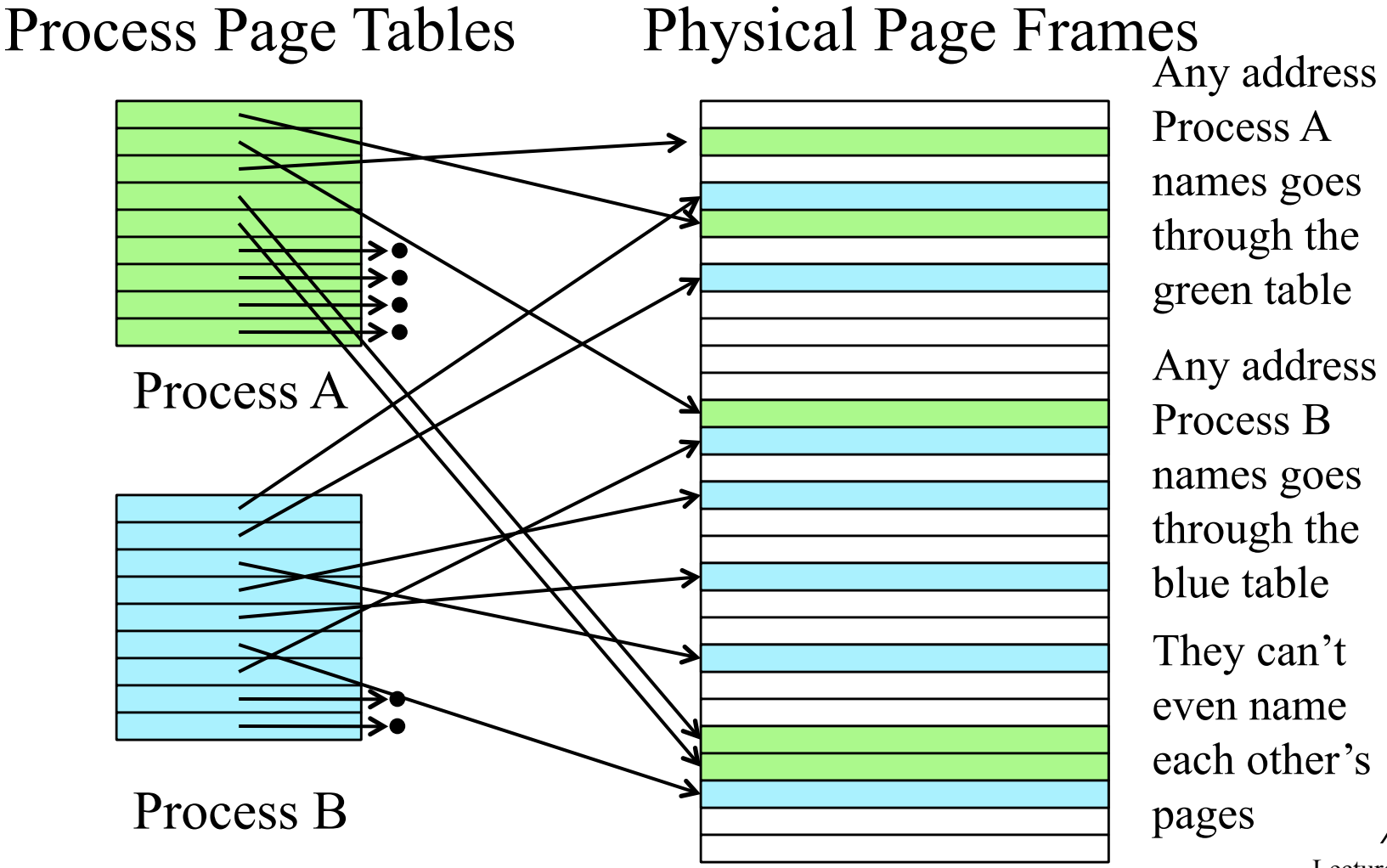
Paging and Security

- Main memory is divided into page frames
- Every process has an address space divided into logical pages
- For a process to use a page, it must reside in a page frame
- If multiple processes are running, how do we protect their frames?

Protection of Pages

- Each process is given a page table
  - Translation of logical addresses into physical locations
- All addressing goes through page table
  - At unavoidable hardware level
- If the OS is careful about filling in the page tables, a process can't even name other processes' pages

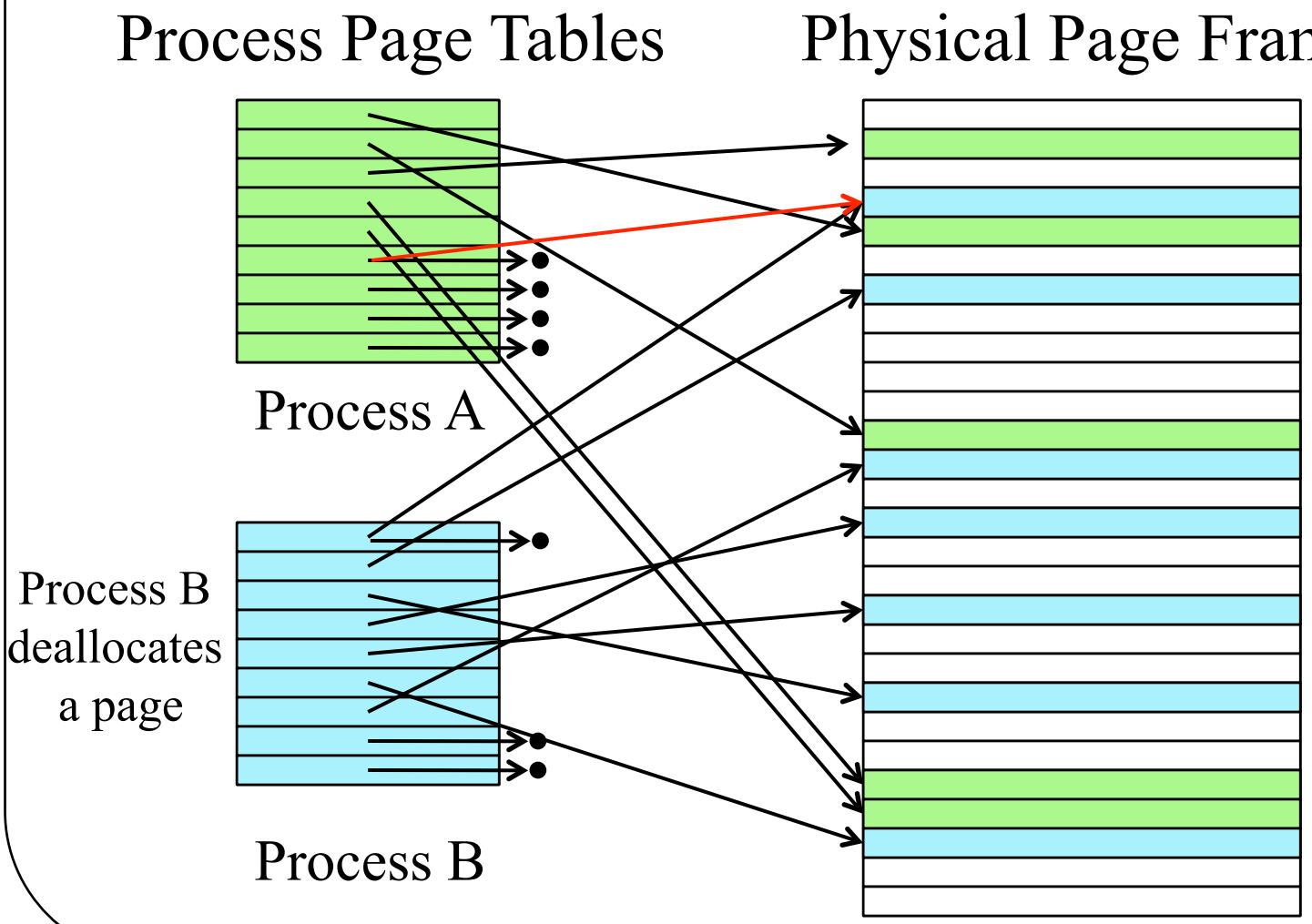
Page Tables and Physical Pages



Security Issues of Page Frame Reuse

- A common set of page frames is shared by all processes
- The OS switches ownership of page frames as necessary
- When a process acquires a new page frame, it used to belong to another process
  - Can the new process read the old data?

Reusing Pages



Strategies for Cleaning Pages

- Don't bother
  - Basic Linux strategy
- Zero on deallocation
- Zero on reallocation
- Zero on use
- Clean pages in the background
  - Windows strategy

Special Interfaces to Memory

- Some systems provide a special interface to memory
- If the interface accesses physical memory,
  - And doesn't go through page table protections,
  - Then attackers can read the physical memory
  - Letting them figure out what's there and find what they're looking for

Buffer Overflows

- One of the most common causes for compromises of operating systems
- Due to a flaw in how operating systems handle process inputs
  - Or a flaw in programming languages
  - Or a flaw in programmer training
  - Depending on how you look at it

What Is a Buffer Overflow?

- A program requests input from a user
- It allocates a temporary buffer to hold the input data
- It then reads all the data the user provides into the buffer, but . . .
- It doesn't check how much data was provided

For Example,

```
int main(){
  char name[32];
  printf("Please type your name:  ");
  gets(name);
  printf("Hello, %s", name);
  return (0);
}
```

- What if the user enters more than 32 characters?

Well, What If the User Does?

- Code continues reading data into memory
- The first 32 bytes go into `name` buffer
  - Allocated on the stack
  - Close to record of current function
- The remaining bytes go onto the stack
  - Right after `name` buffer
  - Overwriting current function record
  - Including the instruction pointer

Why Is This a Security Problem?

- The attacker can cause the function to "return" to an arbitrary address
- But all attacker can do is run different code than was expected
- He hasn't gotten into anyone else's processes
  - Or data
- So he can only fiddle around with his own stuff, right?

Is That So Bad?

- Well, yes
- That's why a media player can write configuration and data files
- Unless roles and access permissions set up very carefully, a typical program can write all its user's files

The Core Buffer Overflow Security Issue

- Programs often run on behalf of others
  - But using your identity
- Maybe OK for you to access some data
- But is it OK for someone who you're running a program for to access it?
  - Downloaded programs
  - Users of web servers
  - Many other cases

Using Buffer Overflows to Compromise Security

- Carefully choose what gets written into the instruction pointer
- So that the program jumps to something you want to do
  - Under the identity of the program that's running
- Such as, execute a command shell
- Usually attacker provides this code

Effects of Buffer Overflows

- A remote or unprivileged local user runs a program with greater privileges
- If buffer overflow is in a root program, it gets all privileges, essentially
- Can also overwrite other stuff
  - Such as heap variables
- Common mechanism to allow attackers to break into machines

Stack Overflows

- The most common kind of buffer overflow
- Intended to alter the contents of the stack
- Usually by overflowing a dynamic variable
- Usually with intention of jumping to exploit code
  - Though it could instead alter parameters or variables in other frames
  - Or even variables in current frame

Heap Overflows

- Heap is used to store dynamically allocated memory
- Buffers kept there can also overflow
- Generally doesn't offer direct ability to jump to arbitrary code
- But potentially quite dangerous

What Can You Do With Heap Overflows?

- Alter variable values
- "Edit" linked lists or other data structures
- If heap contains list of function pointers, can execute arbitrary code
- Generally, heap overflows are harder to exploit than stack overflows
- But they exist
  - E.g., one discovered in Google Chrome in February 2012

Some Recent Buffer Overflows

- Cisco Cable Modem software and Cisco Adaptive Security Appliance
- Pro-face GP Pro-EX industrial control system
- Xen Hypervisor
  - A heap overflow
- glibc
  - A bad place for a buffer overflow

Fixing Buffer Overflows

- Write better code (check input lengths, etc.)
- Use programming languages that prevent them
- Add OS controls that prevent overwriting the stack
- Put things in different places on the stack, making it hard to find the return pointer (e.g., Microsoft ASLR)
- Don't allow execution from places in memory where buffer overflows occur (e.g., Windows DEP)
  - Or don't allow execution of writable pages
- Why aren't these things commonly done?
  - Sometimes they are, but not always effective
- When not, presumably because programmers and designers neither know nor care about security