ELEC 377 – Operating Systems

Week 11 – Class 2

Last Class

- Security and Program Threats

Today

- Security
  - ◊ Other Program Threats

Buffer Overflow (Globals)

- Variants

◊ function pointers in the heap within range of a global buffer (simple overwrite)

```
char buffer[1024];
struct proc_dir{
   int (*read_proc)(char *page, char**start...)
} theProcDir;
```

◊ theProcDir is after buffer in memory, overwrite read_proc variable, next time called, calls our code

Buffer Overflow (Globals)

- Variants

◊ vtable pointers (C++)

```
class A {                                          class B:
  public A {
    virtual int foo(){....};                virtual int foo(){....};
    int bar(){.....};                                  int
  bar(){.....};
}                                                      }
```

- call to bar is known at compile time (called directly)
- foo is based on type of instance in variable
- called through a global table of functions

Buffer Overflow in the Heap

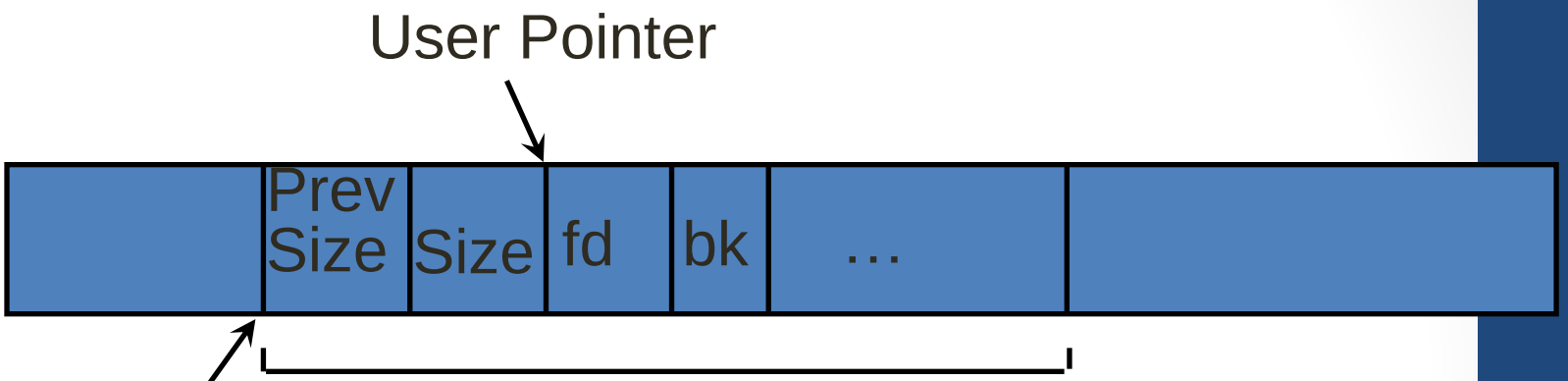
- What if the buffer is in the heap (after pointers)?
  - unused memory is kept in bins based on size of block
  - each bin is represented by a double linked list

```
#define INTERNAL_SIZE_T size_t

struct malloc_chunk {
    INTERNAL_SIZE_T prev_size;
    INTERNAL_SIZE_T size;
    struct malloc_chunk * fd;
    struct malloc_chunk * bk;
};
```

This section based on “Smashing the Heap for Fun and Profit”, Michel "MaXX" Kaempf, http://doc.bughunter.net/buffer-overflow/heap-corruption.html

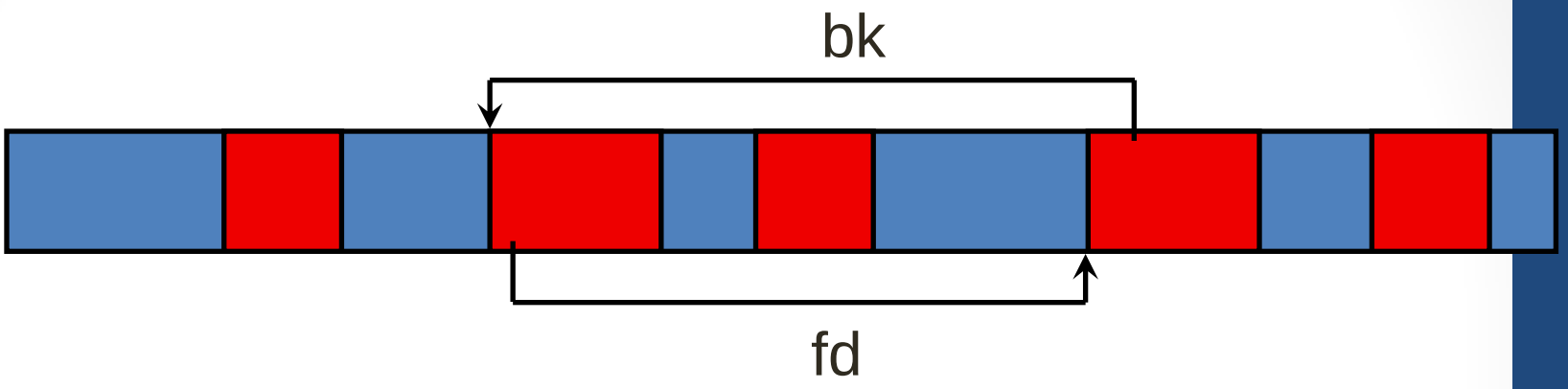
Heap Data Structure



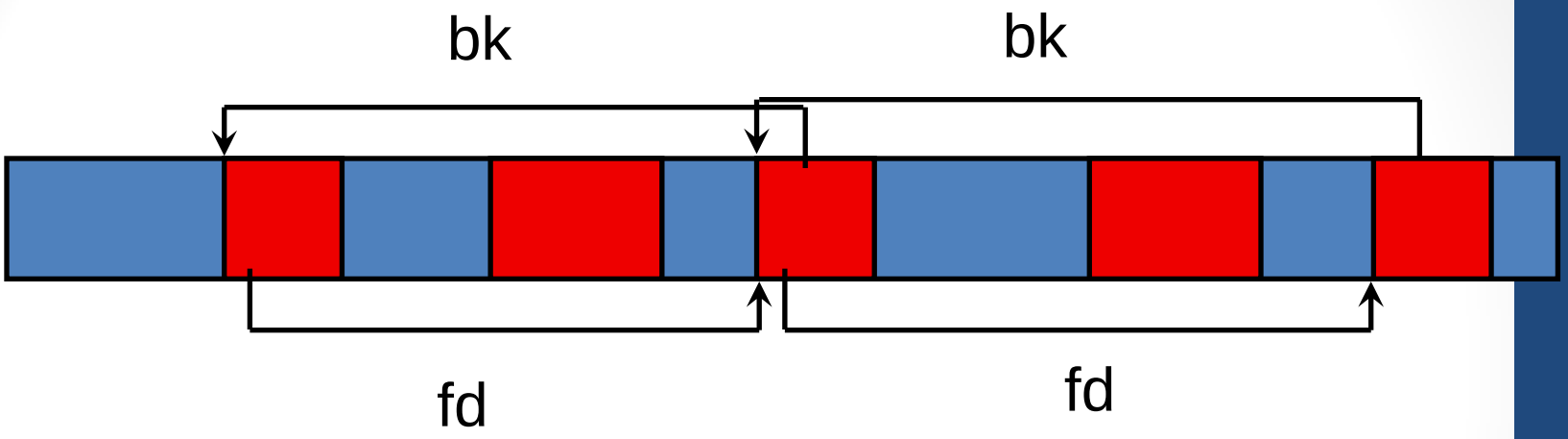
fd (forward) and bk (backward) are only used when the block is unallocated

Prev Size and Size are always used

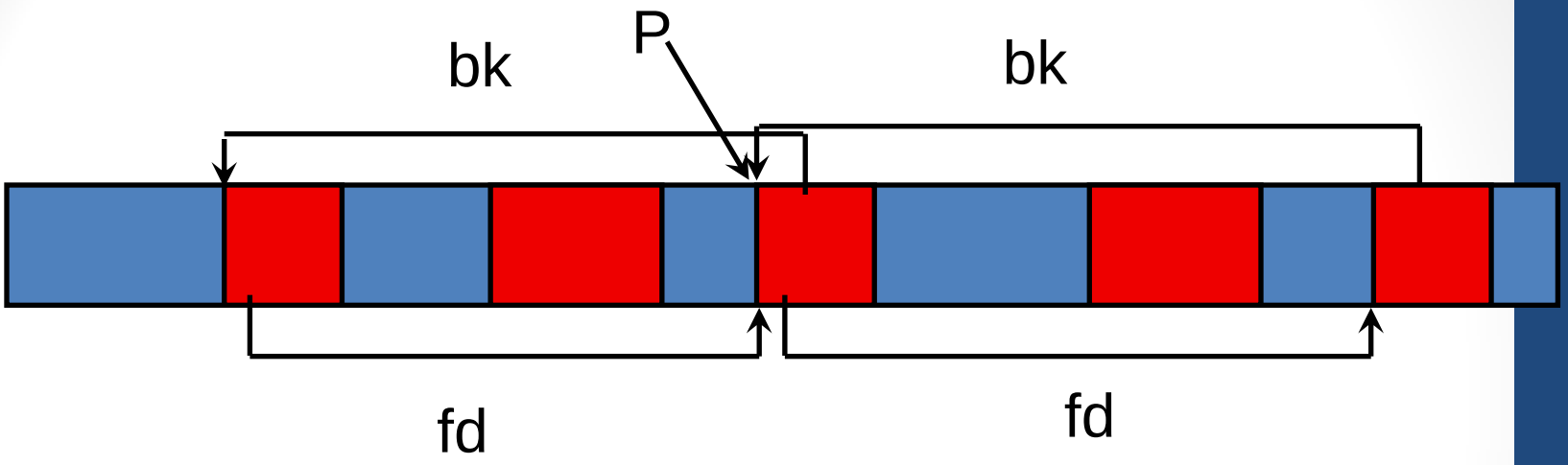
Linking Blocks



Linking Blocks

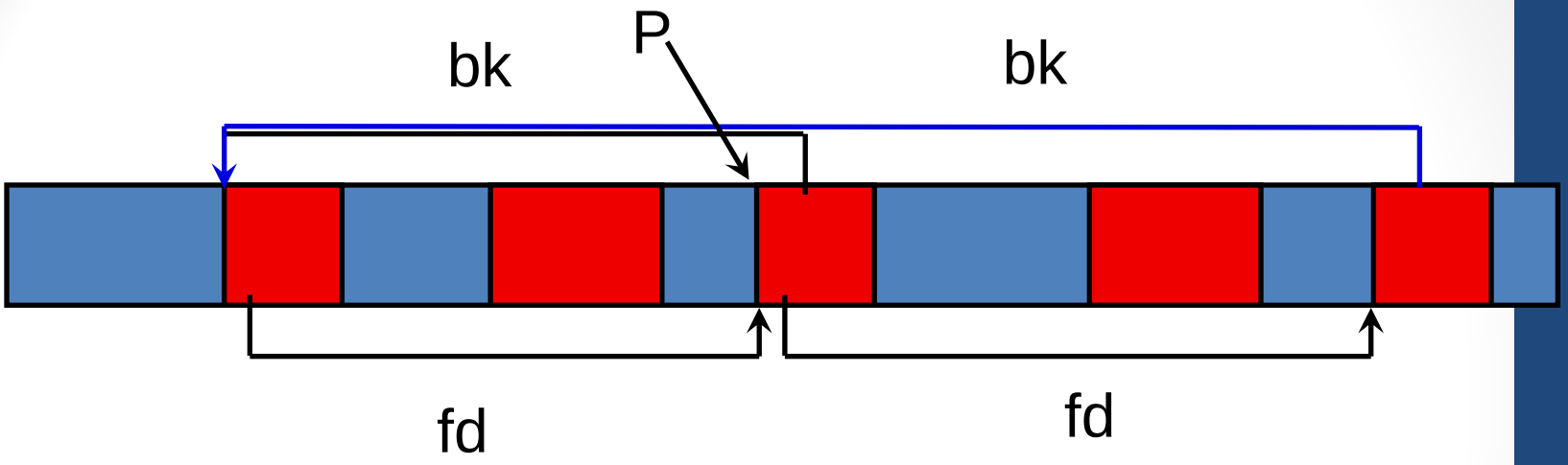


Unlinking



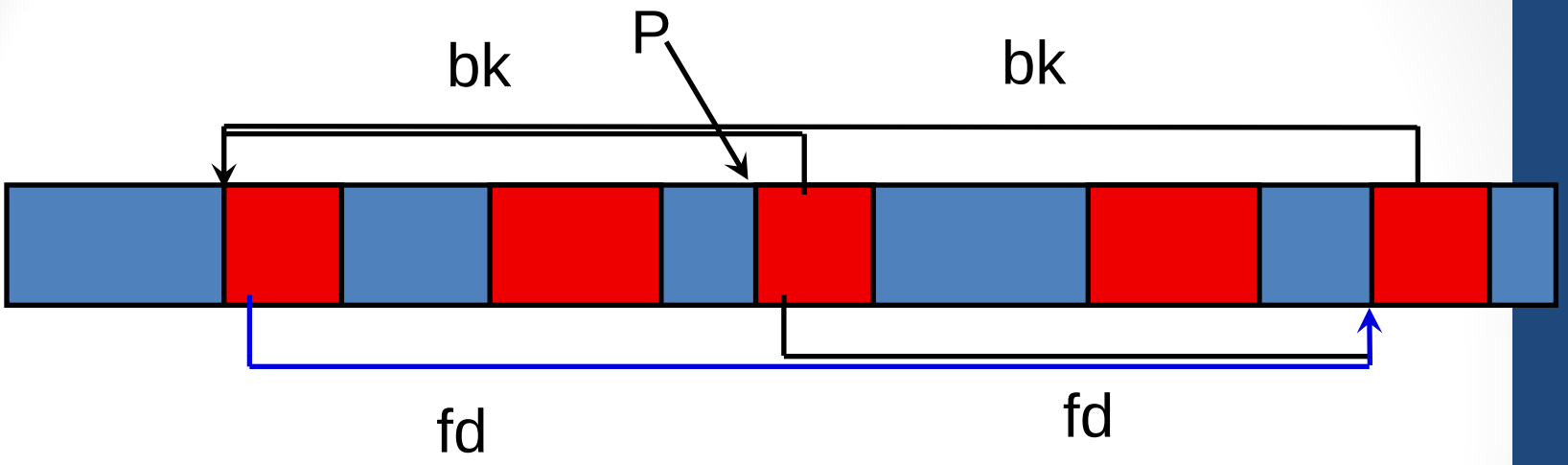

```
#define unlink( P, BK, FD ) { \
  BK = P->bk;                 \
  FD = P->fd;                 \
  FD->bk = BK;                \
  BK->fd = FD;                \
}
```

Unlinking



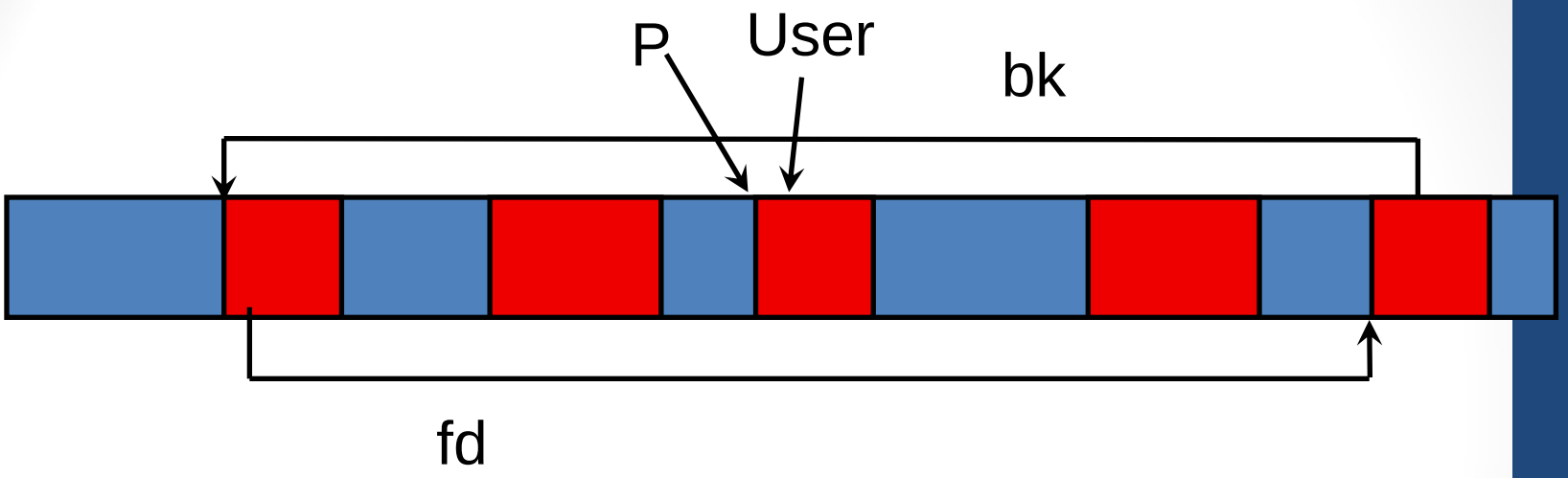

```
#define unlink( P, BK, FD ) { \
  BK = P->bk;                 \
  FD = P->fd;                 \
  FD->bk = BK;                 \
  BK->fd = FD;                 \
}
```

Unlinking



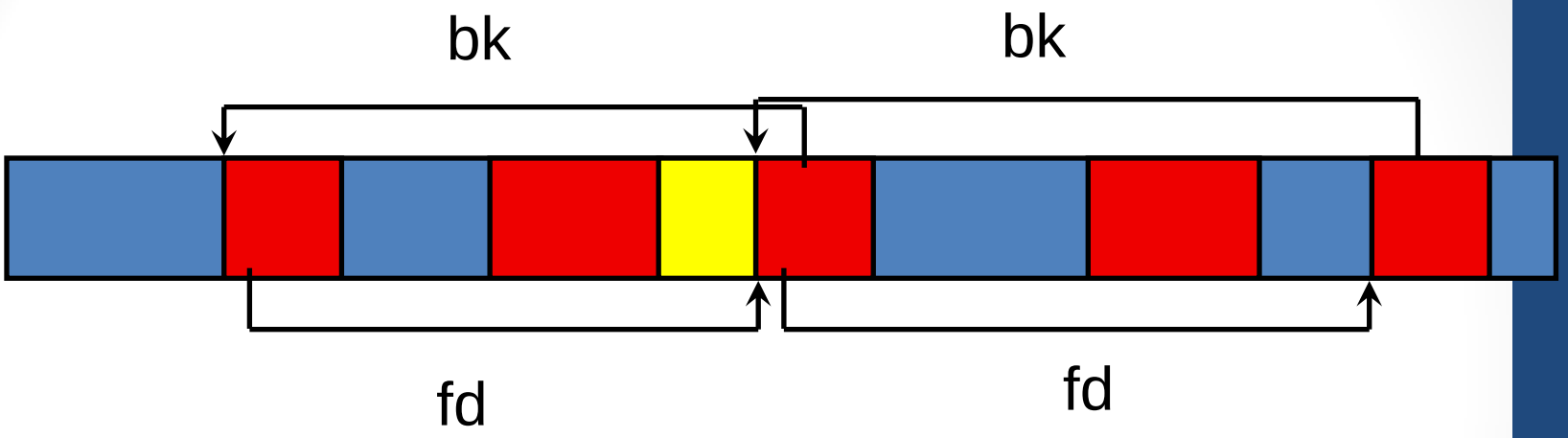

```
#define unlink( P, BK, FD ) { \
  BK = P->bk;                 \
  FD = P->fd;                 \
  FD->bk = BK;                \
  BK->fd = FD;                \
}
```

Unlinking

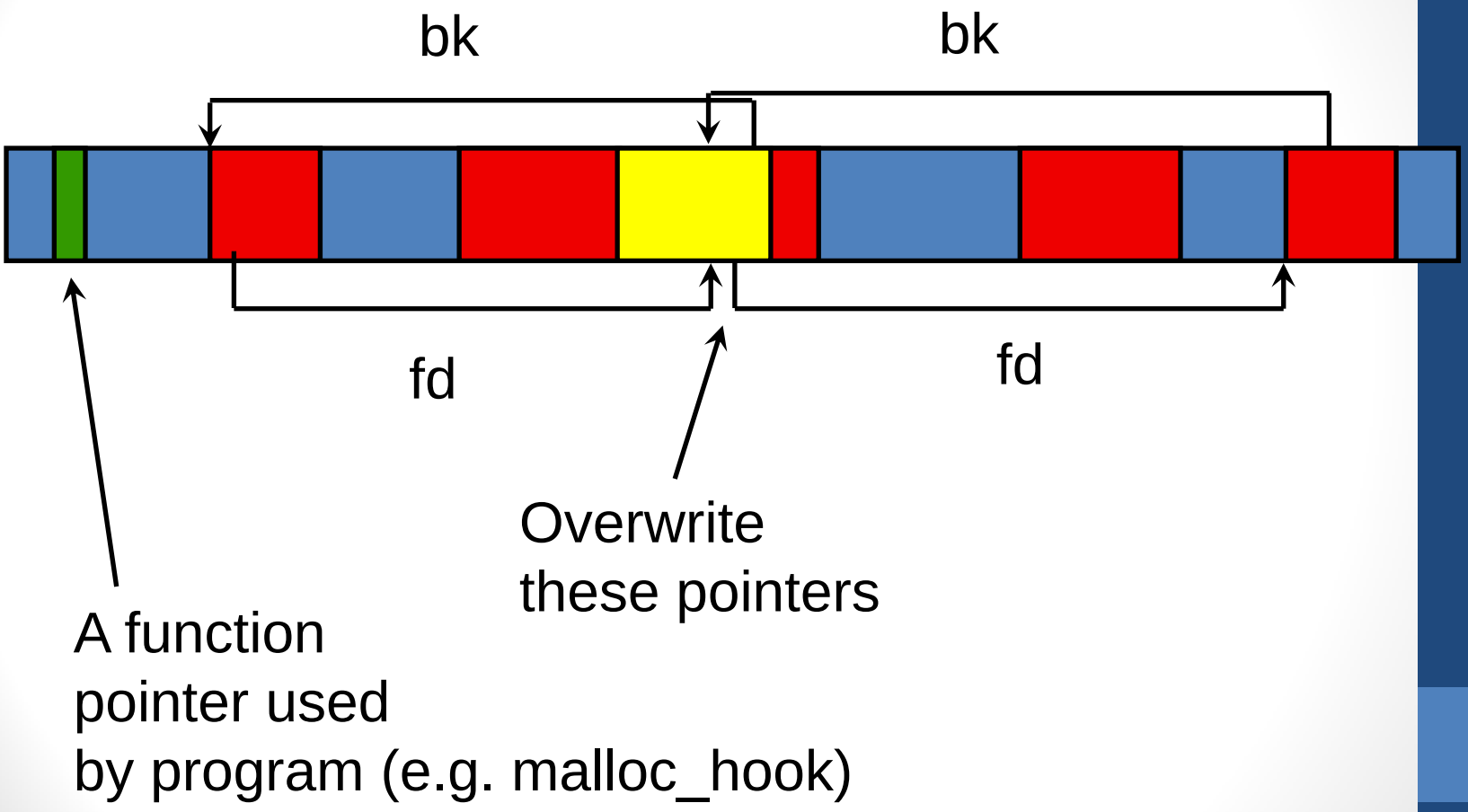



```
#define unlink( P, BK, FD ) { \
  BK = P->bk;                 \
  FD = P->fd;                 \
  FD->bk = BK;                \
  BK->fd = FD;                \
}
```

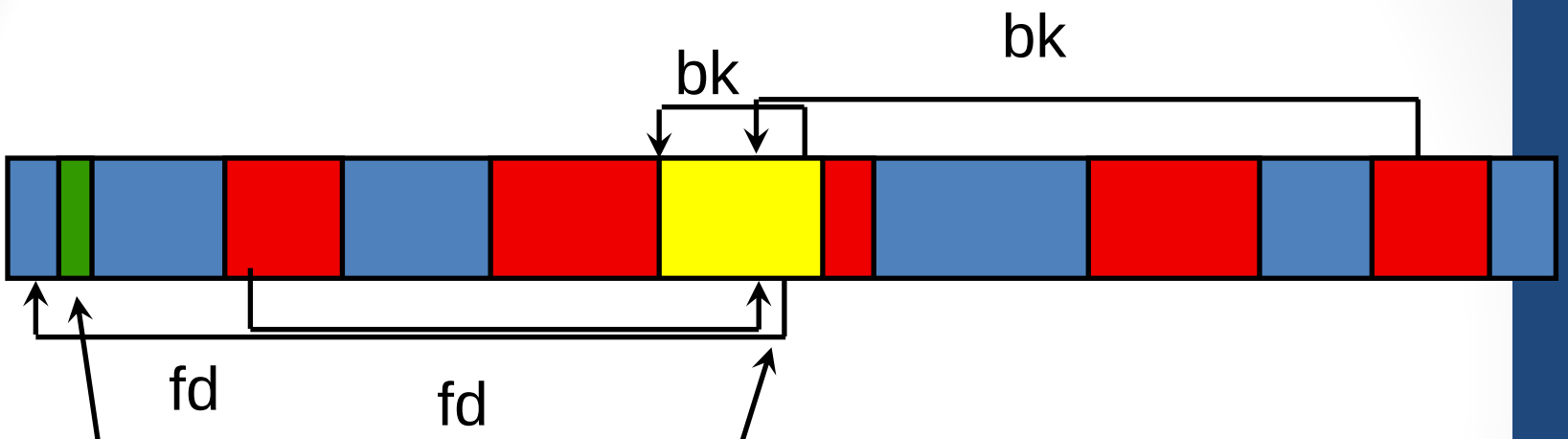
The Vulnerable Buffer



The Vulnerable Buffer



The New Pointers



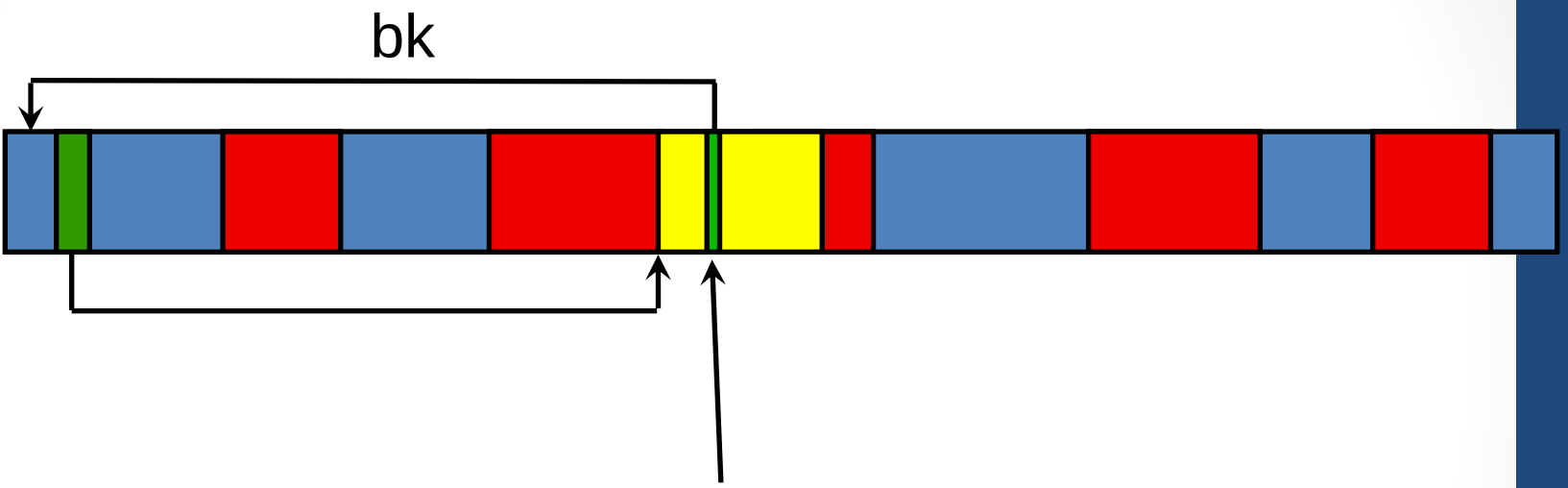
A function pointer used by program (e.g. _malloc_hook) re-located fd+8

Overwrite these pointers

** pointers no longer point to free blocks

** wait for a malloc call.....

After Unlinking...



- 4 bytes at offset 8 get overwritten
- shell code has to jmp around..

**Next time the function pointer is used...
Our code gets executed!!

Buffer Overflow

- Other Examples
  - PDF Javascript Bug
  - Outlook Date Bug

- Whats the point
  - not here to teach you how to break in.
  - illustrate how easy it is to take advantage of errors
  - implications of certain classes of errors in code.

Program Threats

- Race Conditions
  - ◊ suid programs (programs that run with administrator priveledges)
  - ◊ make a security check before doing an action
  - ◊ do the action

  In the moment between check and do, attacker switches the action. Often involves files in /tmp directory (writable by anyone)

  protection: don’t execute something the user can change!!

Program Threats

- Checking parameters
  - ◊ shell scripts on unix. File contains:
    %!/bin/sh
    …shell commands…
  - ◊ execute with -i flag (means interactive shell)
  - ◊ if setuid shell script, now interactive shell in other users name
  - ◊ Most Unixes now do not support setuid shell scripts

Program Threats

- Checking parameters
  - ◊ web parameters
  - ◊ execute a system command using parameters taken from a web form
    e.g. “mail -f confirmation $remote_address”
    - where remote_address comes from web form
    - remote_address contains
      “joe@foo.com ; rm -rf /*”
  - ◊ cannot rely on javascript to verify form data
    - anyone can write a program to send data to a web server!!

Program Threats

- Checking parameters
  - ◊ SQL Injection
  - ◊ Take user input and insert into a query

  SELECT from Table1 where Parm='<user input here>'

  user input = fred';update employee set salary=70000 where emp='barney

Program Threats

- Checking parameters
  - ◊ SQL Injection
  - ◊ Take user input and insert into a query

SELECT from Table1 where Parm='<user input here>'

user input = fred';update employee set salary=70000 where emp='barney

Program Threats

- Checking parameters
  - ◊ SQL Injection
  - ◊ Take user input and insert into a query

  SELECT from Table1 where Parm='<user input here>'

  user input = fred';update employee set salary=70000 where emp='barney

  ◊ must scan input for key characters before issuing database commands

System Threats

- Virus
  - ◊ covers a lot of ground
  - ◊ trogan horse as vector
  - ◊ infects boot sector/other programs
  - ◊ macro viruses
  - ◊ mail viruses
  - ◊ often combined with other attacks
    - date overflow bug
  - ◊ more sophisticated
    - contains own mail servers
    - camouflage

System Threats

- Worms
  - ◊ Automated program that breaks into another system and creates a copy on the new system
  - ◊ soon running on many vulnerable systems
  - ◊ can take a delayed action (Code Red)
  - ◊ Major Commercial Activity (Organzied Crime) - child pornography, software piracy, spam

- Distinction between worm and virus is the vector. Virus needs a human action, worm contains code to attack the next machine.
  - ◊ fuzzy distinction, two techniques are merging…

Recent Developments

- Metamorphic Virus and Worms
  mov eax,0 xor eax,eax; nop
  ◊ multiple rewrites of code that are the same
    - change registers,
    - change constants
    - invert tests
  ◊ malware detectors are signature based
    - software changes its signatures
      – exponential number of signatures
    - must be normalized to compare to signatures
      – extra computation, more expensive to detect

Recent Developments

- Botnets networks of malware (zombies)
  - ◊ After infecting a machine, connect to a given server and await commands
    - update
    - download and execute code
  - ◊ early malware connected to regular IRC servers
    – password protected channels
  - ◊ now connect to private IRC servers in foreign countries
  - ◊ several projects to break into the channels and shutdown the botnets
    - spreading faster than can be shut down

Botnets

- Botnets networks of malware
  - ◊ latest development
    - low bandwidth p2p network
    - zombies divided into cells of several 100 CPUs
    - redundant connections between cells
  - ◊ If you shut down the server, and the owner of the botnet has a connection to any one of the zombies, can use the p2p network to give them a new IRC network to connect to.

Botnets

- Summer 2005

◊ Worm Botnet

- collects registration codes of commercial software
- backdoor to video camera
    - Student Residences
    - Young Adult/Children Bedrooms
    - Camera light?

Securing Systems and Facilities

- Periodic Scans
  - ◊ check passwords
  - ◊ set uid programs
  - ◊ unauthorized programs in special directories
  - ◊ long running processes
  - ◊ directory and file protection bits
  - ◊ system search path
  - ◊ changes to system programs

Securing Systems and Facilities

- Cannot lock up the machines
- firewalls
  - ◊ in automobile, between engine and passengers
  - ◊ in network, between wild jungle of internet and (almost) secure network
  - ◊ limit connections between outside and inside
  - ◊ Demilitarized Zone (DMZ)
  - ◊ network address translation (NAT)
  - ◊ covert tunnels
  - ◊ spoofing

Intrusion Detection

- Aspects
  - ◊ real time vs after intrusion
  - ◊ what is examined (commands, system calls, network packets, etc.)
  - ◊ response

- What is an Intrusion?
  - ◊ signature based detection
    - virus, multiple login attempts
  - ◊ anomaly based detection
    - something not normal

Intrusion Detection

- Issues
  - ◊ Delay in adding signatures
  - ◊ Errors in signatures
    - AVG accidentally removes user32.dll

  - ◊ stealth channels
    - some intruders only want limited information
    - other want to stay and spy a while....

Intrusion Detection

- Audits and Logs
  - ◊ UNIX syslog daemon
  - ◊ most daemons use the syslog daemon to log activities
  - ◊ swatch - scans daemons for anomalous activity

- Tripwire
  - ◊ Purdue University
  - ◊ checksum of system files and attributes - detect modifications
  - ◊ detect modification of tripwire?

Security is Increasingly Important

- Continue to be interesting in ways never thought of before

  - photo of keys??
    - can now cut keys from keys appearing in a picture, even from a distance of 200 feet

Legal issues of Networks..

- File Sharing....
  - few lawyers, courts or politicians that understand
- automated infringement notices
  - sent to a printer.....
- Net neutrality(Barak Obama Cabinet)
  - who controls, new protocols, competition, conflict of interest..
- Firewalls
  - ssh/http only(everything runs over http?)
  - vpn compatible at both ends?

Root Kits

- Root Kit is software to hide the evidence of system modification
- Originally used by intruders in Unix systems to hide changes to systems
  - ◊ add a back door process such as a chat daemon or ftp server running on non-standard port
  - ◊ changes to ps, netstat, w, passwd and other system commands to hide the back door
- Now applies to any operating system
  - ◊ Changes are now usually made to kernel and system libraries rather than to system commands
  - – Although some combine both system libraries and system commands

What is a Root Kit?

- Not the initial vulnerability
  - ◊ initial vulnerability is used to gain access, root kit is used to maintain access to compromised system
  - ◊ Sometimes the intruder patched vulnerability to keep ‘exclusive’ access to the system
  - ◊ root kit may attempt to maintain ownership of the system - one part of root kit notices when another part has been removed and reinstalls that component
- Often used by viruses and worms to disguise activities.
  - ◊ Thus rootkit detection is a concern for Security Vendors.

Root Kit Research

- Commercial and Personal Systems
  - ◊ when you get malware, you want to remove it
  - ◊ limit its damage
- Sensitive Systems.
  - ◊ You don't want to eradicate the malware
  - ◊ You need to observe it

  -- who is it reporting to?
  -- what kind of information is it interested in
  -- limit access to sensitive information
  - ◊ Problem: it is checking to see if anyone is watching

  -- may self destruct/or may attempt to destroy system.
  -- may change its behaviour.

Sensitive Systems

- Counter-Intelligence Operations
  - ◊ after detecting malware, you provide a simulated environment (including new operator)
  - ◊ replace systems it has access to, with fake systems with fake information
- Observe the malware
  - ◊ CASCON paper
  - ◊ Use root kit techniques to hide the anti malware software from the malware
  - ◊ Installed at time OS is installed -- we are in first!!

Root Kit Research

- Kernel Level Asynchronous Procedure Calls(APC)
  - ◊ register a call back routine for a process inside the kernel
  - ◊ call back executes with knowledge of the processes virtual memory tables, and other process info
  - ◊ Our anti-malware executes entirely as APC callbacks.
  - ◊ copy to different memory location
  - ◊ register callbacks on different threads
  - ◊ Can inject into malware's thread and look at malware in malware's context
  - ◊ jump onto thread to exfiltrate information