Playing Inside the Blackbox: Using Dynamic Instrumentation to Create Security Holes

Barton P. Miller

bart@cs.wisc.edu

Computer Sciences Department

University of Wisconsin

Madison, Wisconsin 53705

USA

Overview

1. How to easily do dangerous and malicious things to a running program.

2. How to detect when someone does something evil to your program.

A New View

Running programs are objects to be easily manipulated. Kinds of manipulations might include:

- Instrumentation
- Optimization
- Control

The Vehicle:The DynInst API

A machine-independent library for machine level code patching.

- Eases the task of building new tools.
- Provides the basic abstractions to patch code on-the-fly

Dynamic Instrumentation

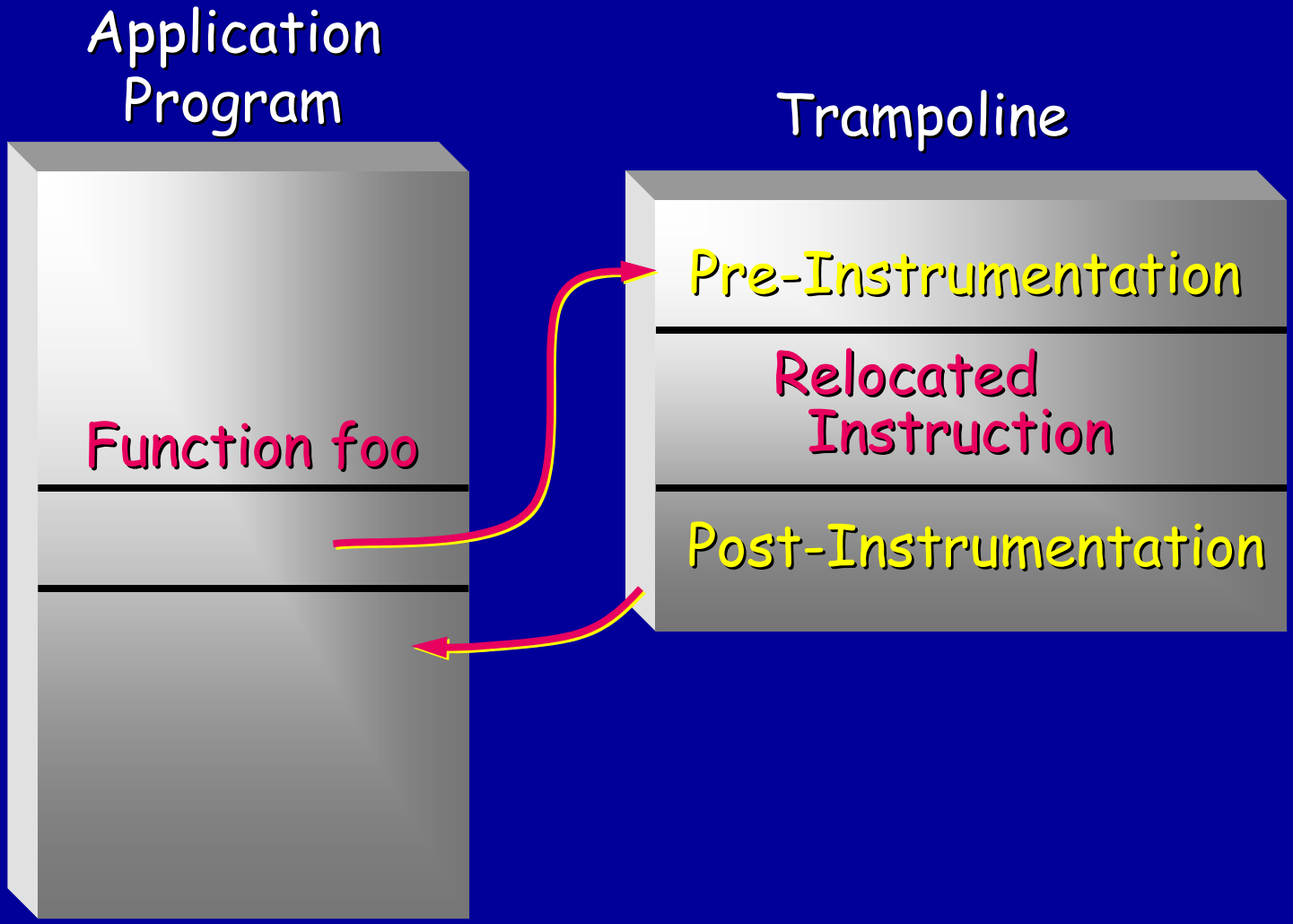
- Does not require recompiling or relinking
  - Saves time: compile and link times are significant in real systems.
  - Can instrument without the source code (e.g., proprietary libraries).
  - Can instrument without linking (relinking is not always possible.
- Instrument optimized code.

Dynamic Instrumentation (con'd)

- Only instrument what you need, when you need
  - No hidden cost of latent instrumentation.
  - Enables "one pass" tools.

- Can instrument running programs:
  - Servers.
  - Application programs.
  - Systems with complex start-up procedures.

The Basic Mechanism



The DynInst Interface

- Machine independent representation
- Object-based interface to build Abstract Syntax Trees (AST's)
- Write-once, instrument-many (portable)
- Hides most of the complexity in the API
  - Process Hijacker: only 700 lines of user code!
  - MPI tracer: 250 lines

Basic DynInst Operations

- Process control:
  - Attach/create process
  - Monitor process status changes
  - Callbacks for fork/exec/exit
- Image (executable program) routines:
  - Find procedures/modules/variables
  - Call graph (parent/child) queries
  - Intra-procedural control-flow graph

Basic DynInst Operations

- Inferior (application processor) operations:
  - Malloc/free
    - Allocate heap space in application process
  - Inferior RPC
    - Asynchronously execute a function in the application.
  - Load module
    - Cause a new .so/.dll to be loaded into the application.

Basic DynInst Operations

- Inferior operations (continued):
  - Remove Function Call
    - Disable an existing function call in the application
  - Replace Function Call
    - Redirect a function call to a new function
  - Replace Function
    - Redirect all calls (current and future) to a function to a new function.

Basic DynInst Operations

- Building AST code sequences:
  - Control structures: if and goto
  - Arithmetic and Boolean expressions
  - Get PID/TID operations
  - Read/write registers and global variables
  - Read/write parameters and return value
  - Function call

Security Applications of DynInst

Lots of tool applications of Dyninst by lots of groups.

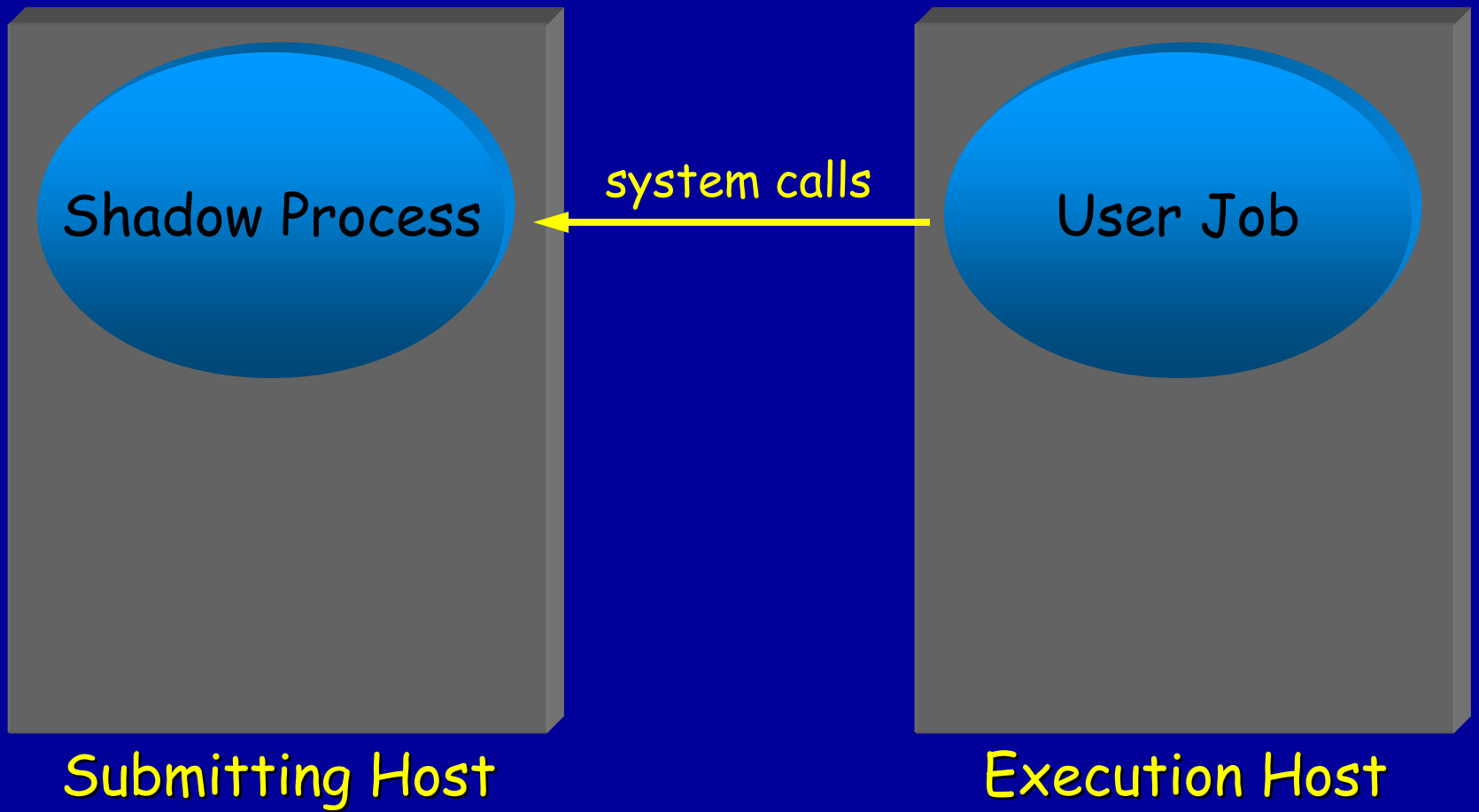
Here are two security-oriented ones:

- License server bypassing
- Condor security attacks

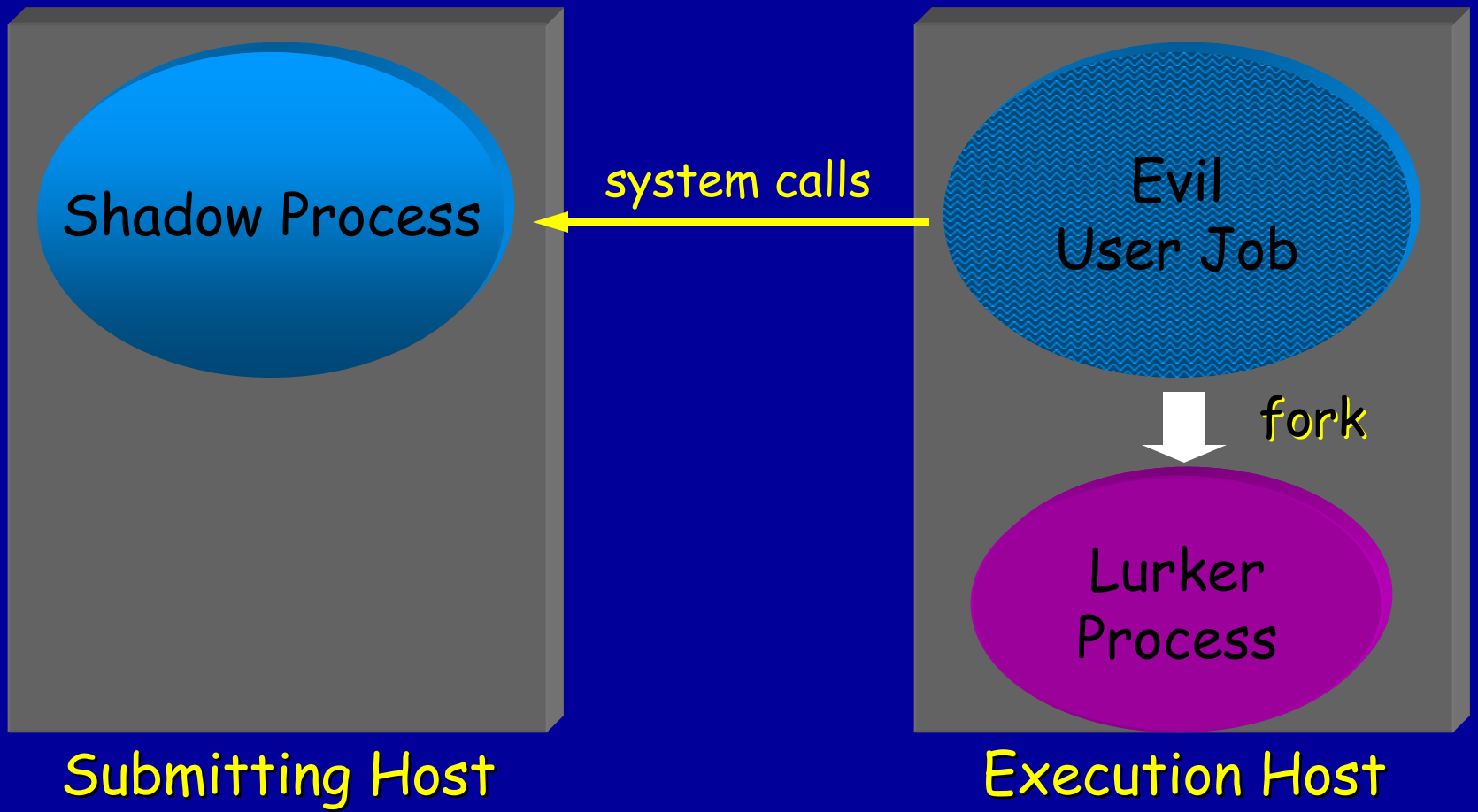
Condor Attack: Lurking Jobs

- Condor schedules jobs on idle workstations
- In a normal mode, jobs run as a common, low-privilege user ID: "nobody".
- This common user ID provides an opportunity for an evil lurking process to ambush subsequent jobs (from other users):

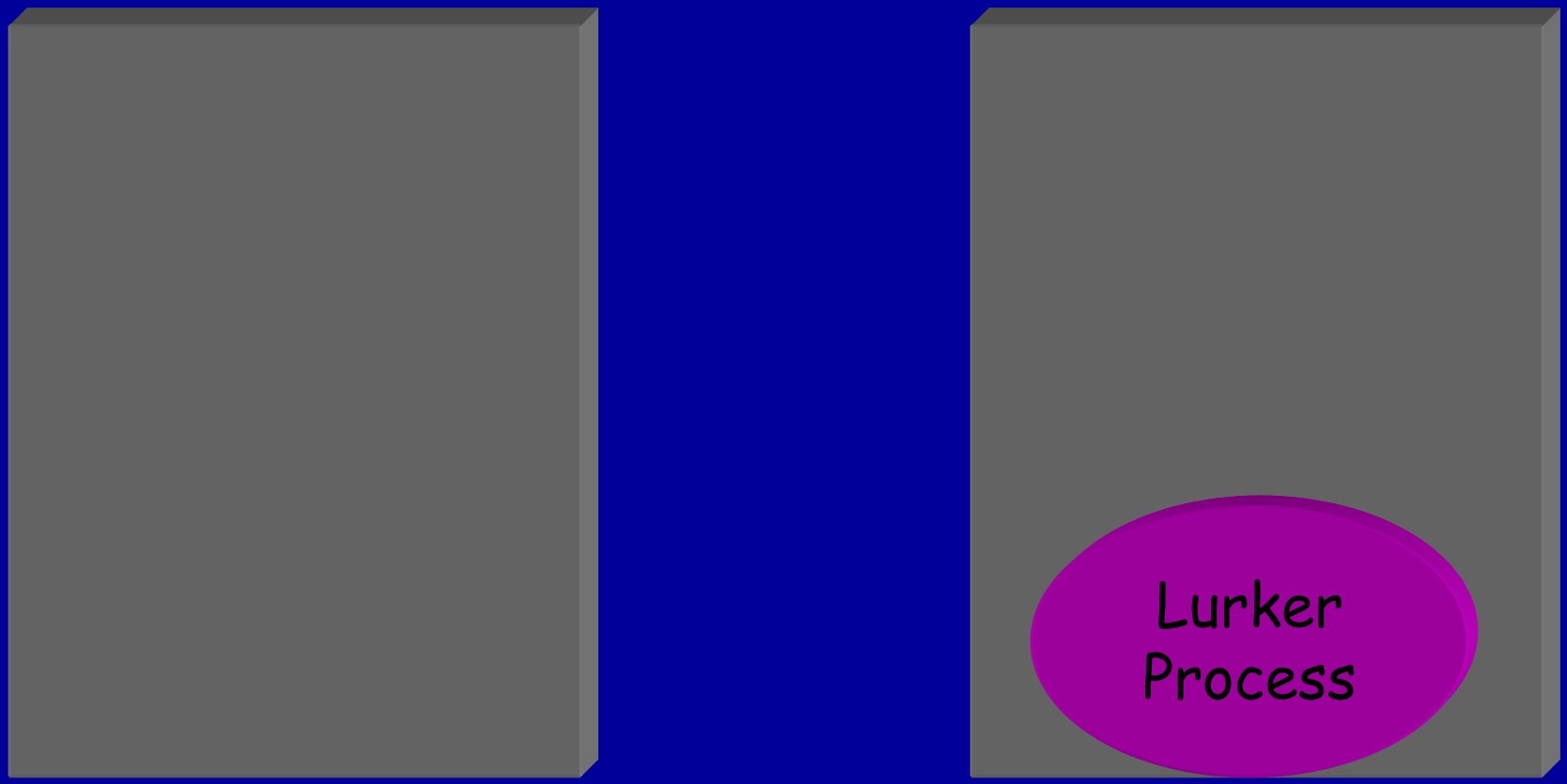
Condor Job Structure



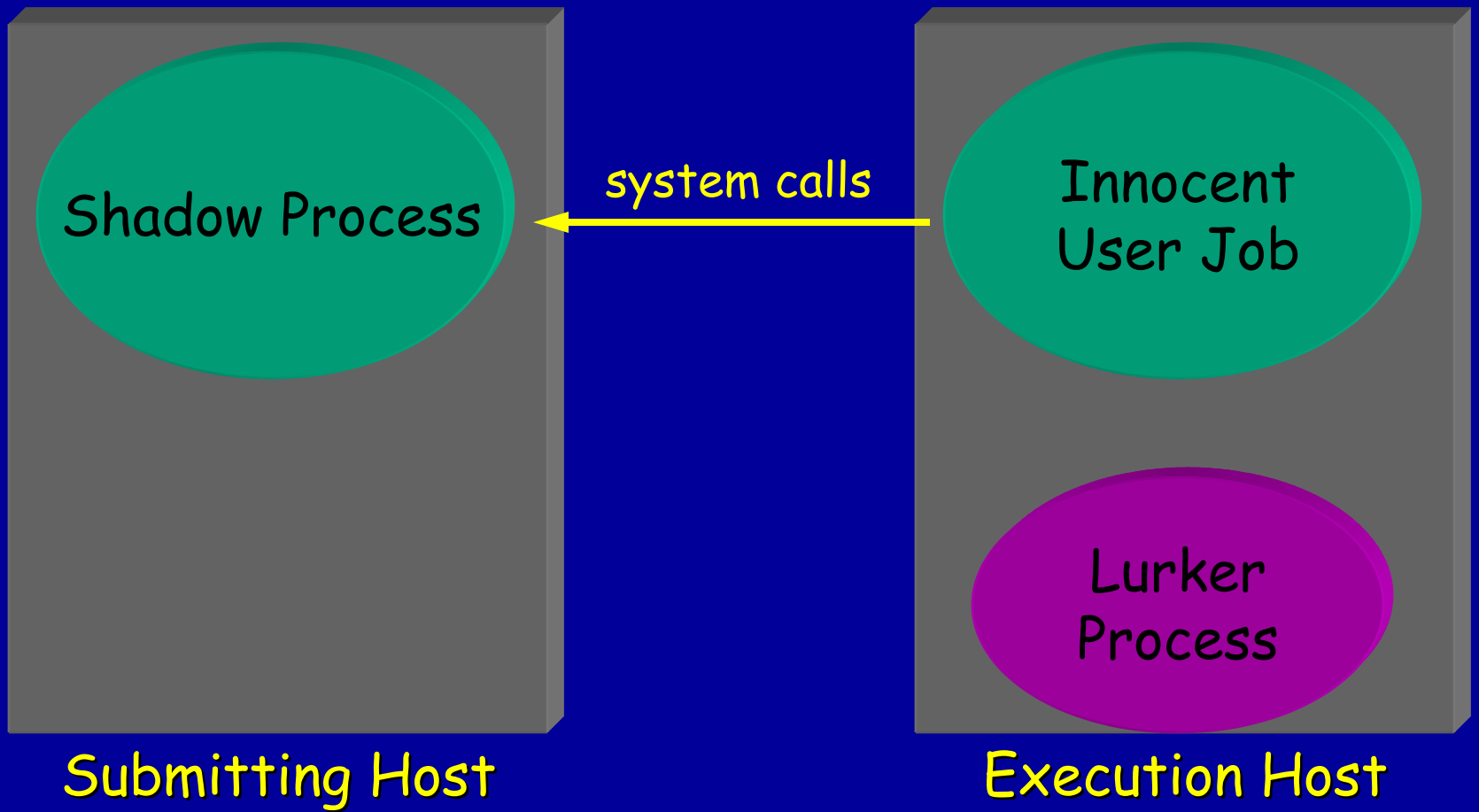
Condor Job Structure



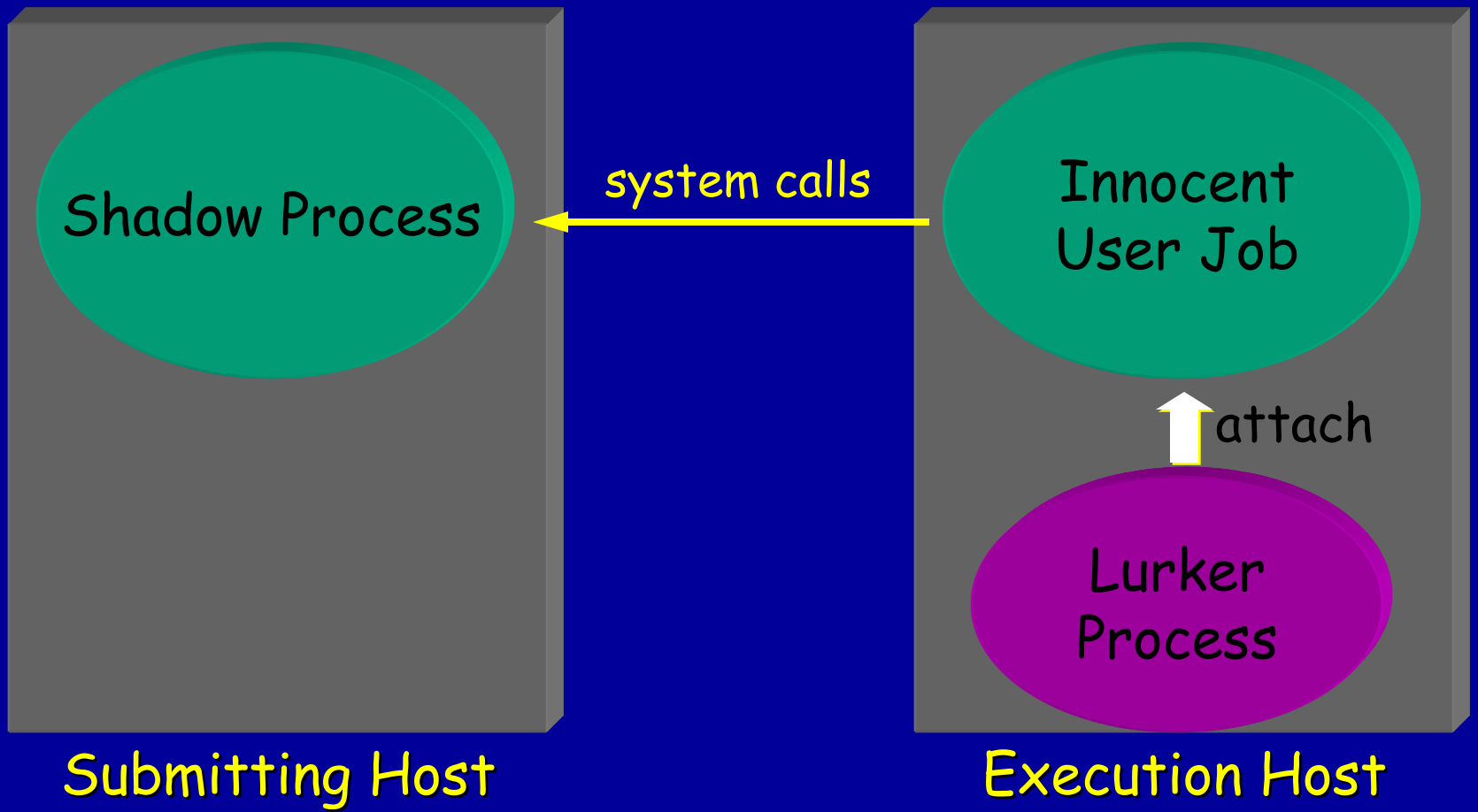
Condor Job Structure



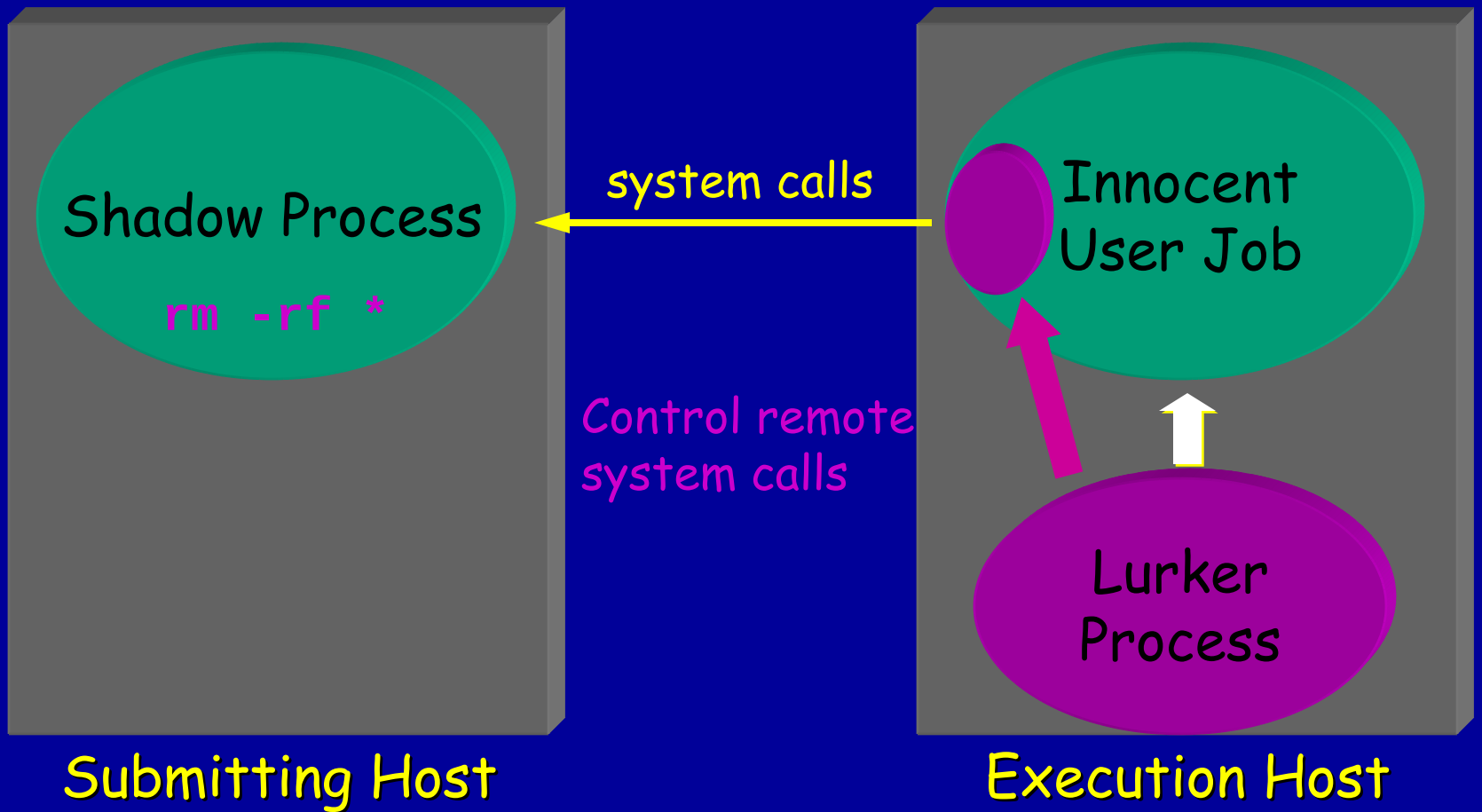
Condor Job Structure



Condor Job Structure



Condor Job Structure



Can We Trust a Remote Job?

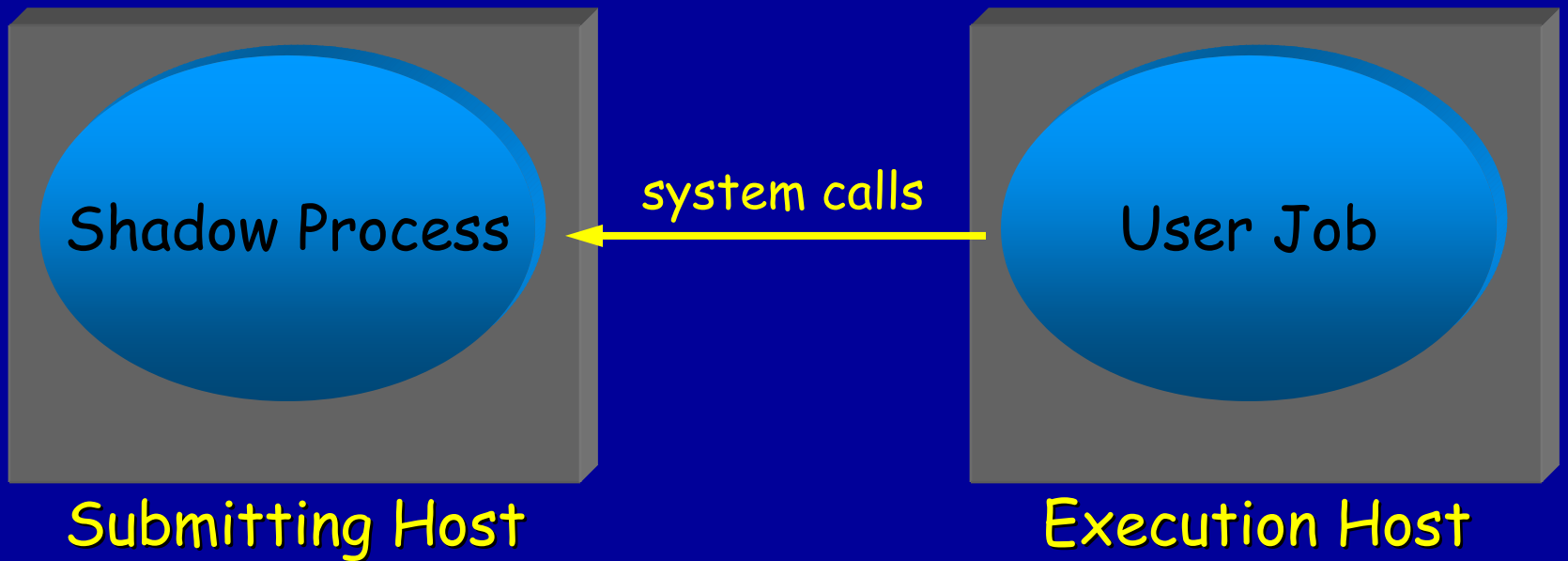
The threats:

1. Cause the job to make improper remote system calls.
2. Cause the job to calculate an incorrect answer.
3. Steal data from the remote job.

Threat protection strategies:

- File sand-boxing (#1)
- System call sand-boxing (#1)
- Obscure and encode binary (#1)
- Replicate remote job (#2)

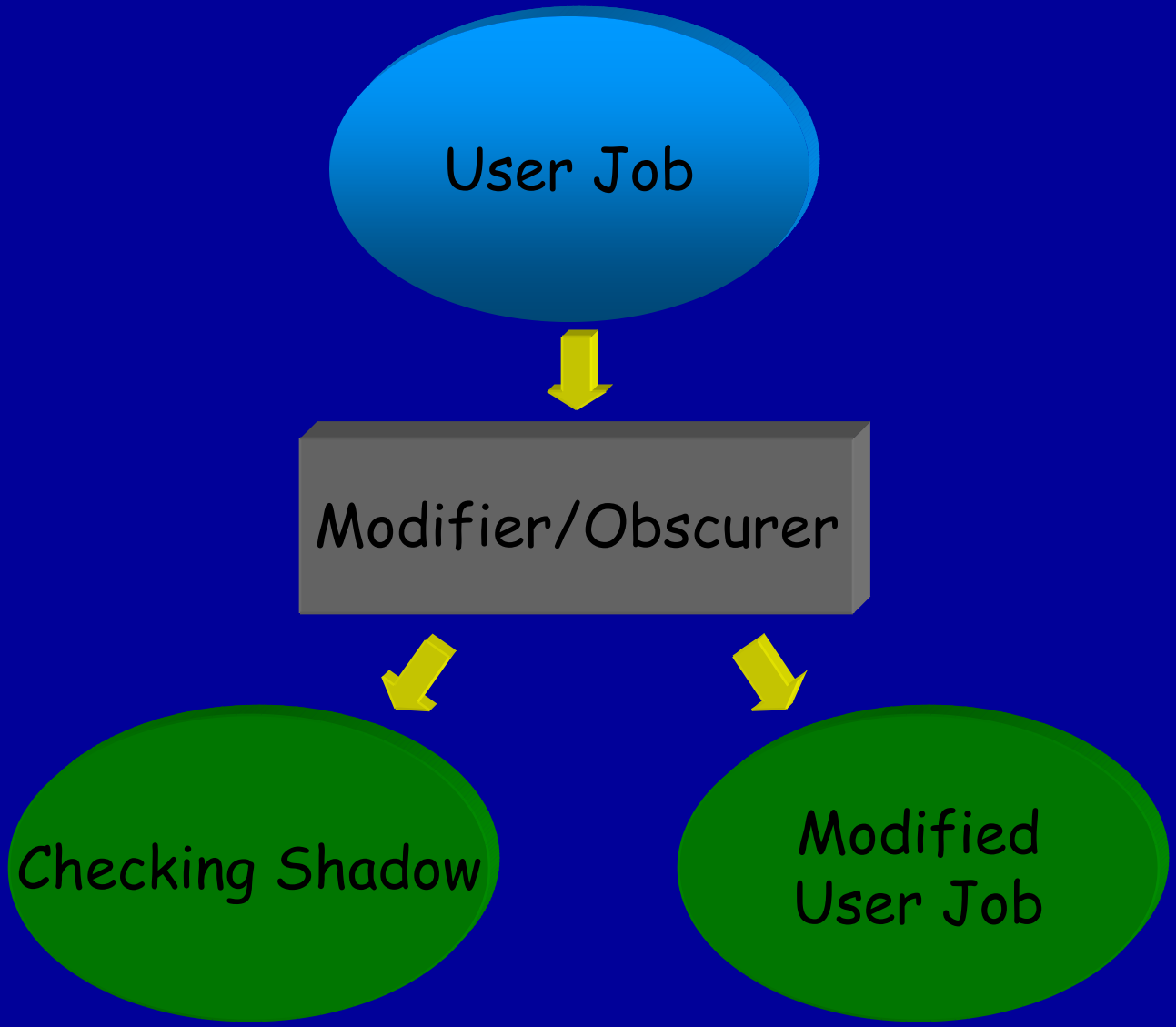
Sand-Boxing



Shadow process selectively rejects system calls:

- Restrict access to specific files or directories
- Disallow certain system calls
- Disallow certain system call parameter values

Obscuring the Executable



Obscuring the Executable

Goal:

Even if an intruder can see, examine, and fully control the remote job, no harm can come to the local machine.

How to Get a Copy of DynInst:

Release 2.3 (release 3.0 imminent)

- Free for research use.
- Runs on Solaris (SPARC & x86), Windows NT, *AIX/SP2*, Linux (x86), Irix (MIPS), Tru64 Unix (Alpha).

http://www.paradyn.org

http://www.dyninst.org

paradyn@cs.wisc.edu