Vulnerability and Information Flow Analysis of COTS

Somesh Jha, Bart Miller, Tom Reps
{jha,bart,reps}@cs.wisc.edu
Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706-1685

Cost of Software Development Motivates Use of COTS

- High cost of software development
  - increased complexity
  - increasing degree of concurrency
  - increasing quality-assurance demands
  - other factors . . .

- Increased deployment of COTS

- CIP/SW TOPIC #6
  - Protecting COTS from the inside

Advantages and Disadvantages of COTS

- Advantages
  - reduced cost
  - promotes modular design
  - partitions the testing effort
- Disadvantages
  - higher risk of vulnerabilities
  - general quality-assurance issues

Host

Kerberos version 4

Buffer overrun vulnerability (see CERT advisory)

Authenticate services

Intruder exploits buffer overrun vulnerability

Compromised Host

Kerberos version 4

Privileged access

Planning software

Intruder modifies software

E-mail results back to the intruder

Planning software with a backdoor

Staff officer uses planning software

E-mail results

S. Jha, B. Miller, and T. Reps

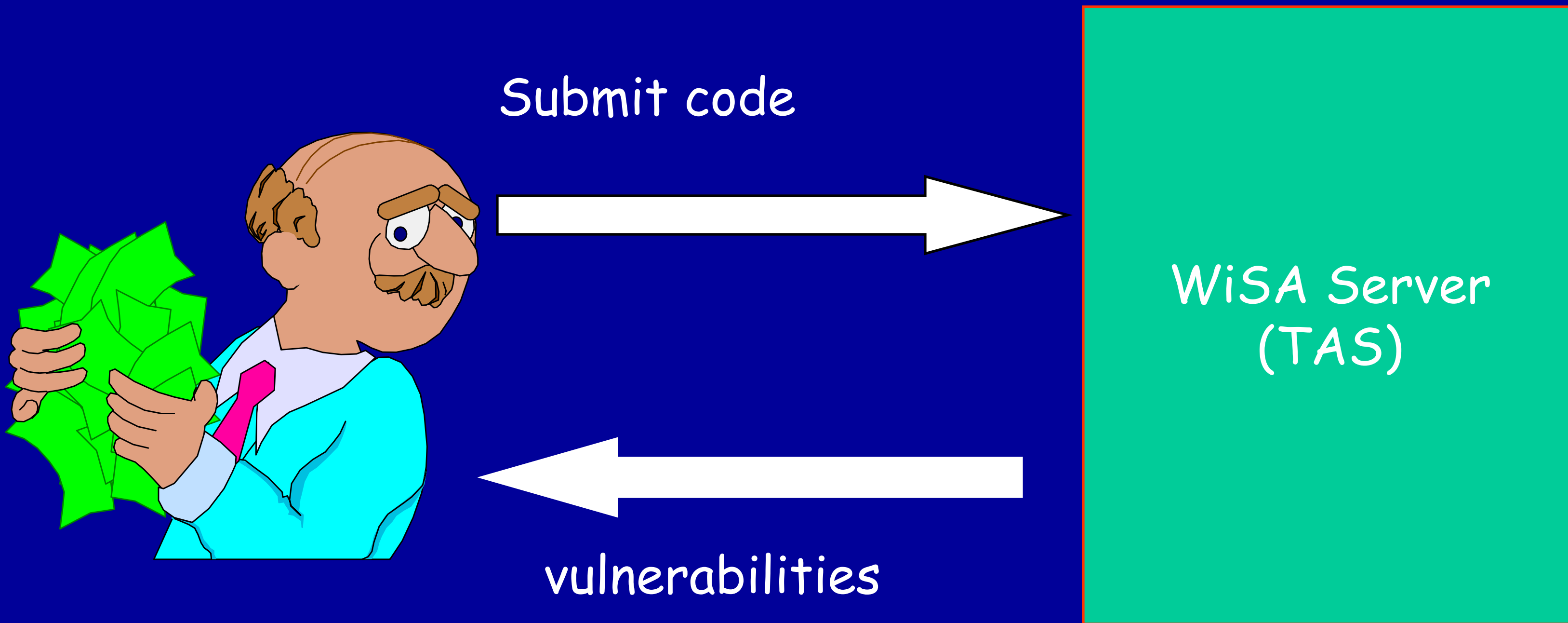
Spyware

- Install a useful program
  - Play DVDs
- But …
  - Also install "spy" software, which monitors user behavior
    - Example: Monitor web traffic
- Aureate Media, Real Networks
- Consult
  - http://grc.com/optout.htm
- Maybe can be used by advisors/managers☺

WiSA: Don't Deploy COTS Without It

- We have proposed the Wisconsin Safety Analyzer
  - vulnerability and
  - information flow analysis of COTS
- Develop technology for static analysis of binaries
- Investigate applications

Trusted verification services



Benefits to DoD

- Reduces risk of deploying COTS
- Capable of discovering vulnerabilities in COTS
  - safety related
  - information-flow related
- Assign assurance levels to COTS components

WiSA Requirements

- Requirement 1
  - cannot mandate that all COTS packages will be written in the same language
  - source code for COTS frequently not available

∴ analysis of binaries/multi-lingual techniques

- Requirement 2
  - safety depends on context
  - desire to specify
    - discretionary access control
    - mandatory access control

  ∴ need an expressive specification language

WiSA Requirements

- <u>Requirement 3</u>
  - there are tradeoffs between scalability & precision
    - generally: efficiency vs. precision
    - but sometimes: more precise = more efficient

∴ tunable precision

- <u>Requirement 4</u>
  - wish to analyze compositions of COTS packages

  ∴ rely-guarantee reasoning and reason about compositions of vulnerabilities and constructing attack graphs

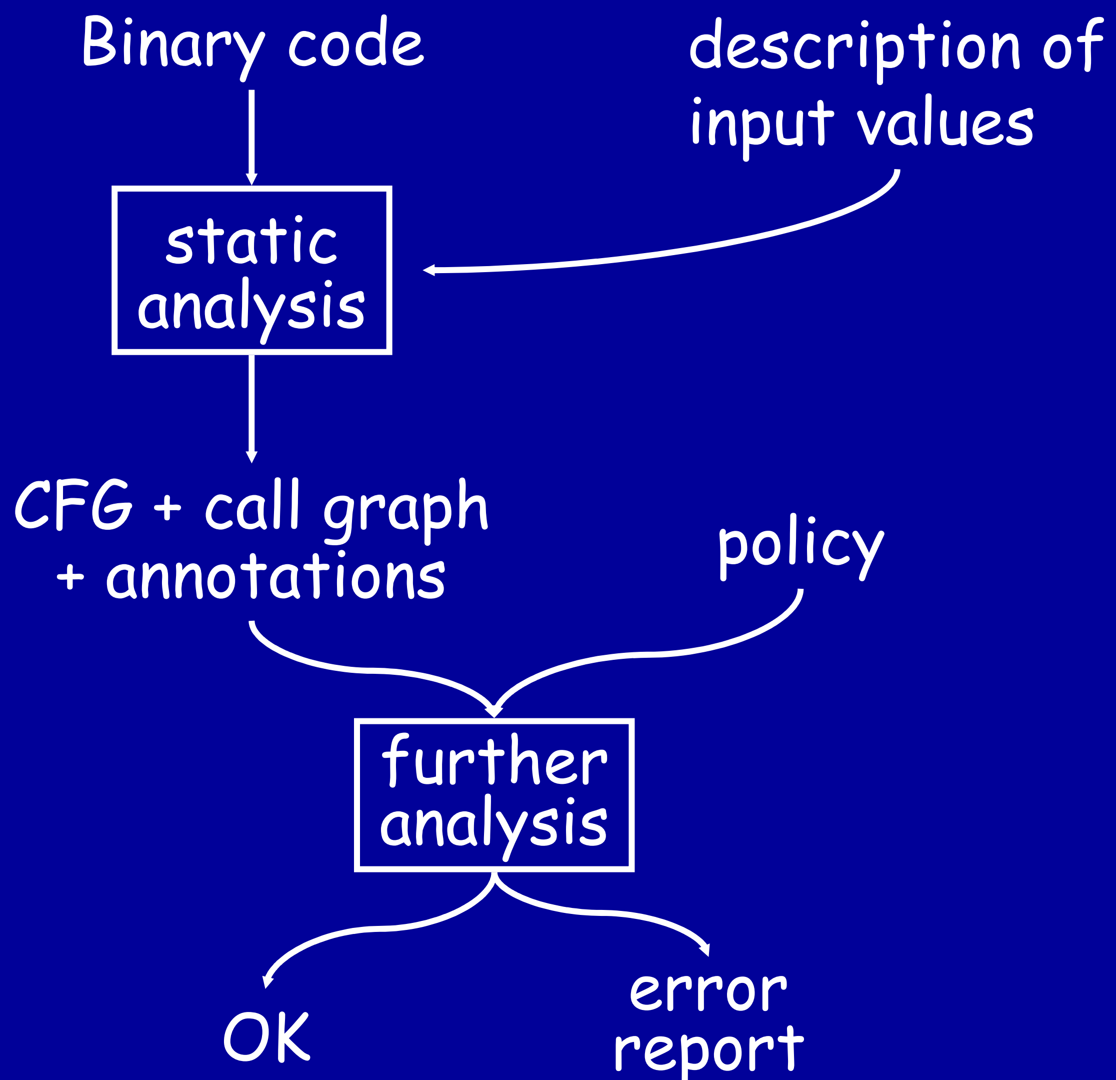
Initial Focus

- Our initial focus is on analyzing x86 binaries
- Reasons
  - high impact
    - several viruses written for the x86 platform
  - rich language
    - several hard analysis issues will be dealt with
    - can reuse architecture and experience in other settings
- partially addresses requirement 1

Malcious Code Detection as a Two Player Game

- “vanilla” virus easy to detect
- virus writers are obfuscators
  - Mihai will talk about several obfuscation transformations
  - example
    - encrypt the virus
    - distribute the virus over a large program
- virus detectors are deobfuscators
  - goal is reconstruct the “vanilla” virus from the obfuscated programs
  - static analysis helps in deobfuscation

Analysis Architecture



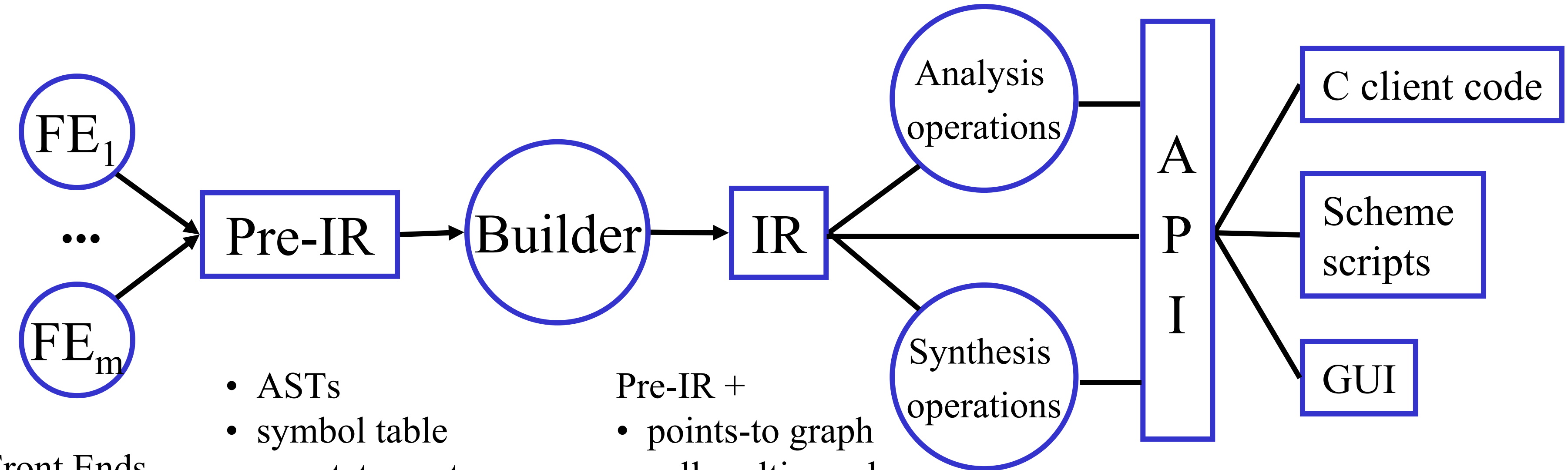
IDA Pro

- Decompilation tool
- Supports several executable file formats like COFF, ELF ....
- Gather as much information as possible
  - e.g. Names of functions, parameters to functions
- Is extensible through a built-in C like language

Codesurfer

- A program understanding tool
- Analyzes the data and control dependencies – stores in System Dependence Graph (SDG)
  - Helpful in static analysis
- Provides a API to access the information stored in SDG
- The API can be extended

CodeSurfer System Architecture



Front Ends
EDG
  ANSI C
  C++
IdaPro
  x86 binary code
Java
UML (Rose/RT)
Verilog
VHDL
Jovial

- ASTs
- symbol table
- per-statement variable usage
- pointer uses
- CFGs

Pre-IR +
- points-to graph
- call multi-graph
- GMOD / GREF
- program-wide variable usage
- PDGs / SDG

- Precise interprocedural
  - predecessors and successors
  - slice and chop
- Boolean operations on sets of PDG nodes
- **model checking**
- **constraint generation**

Other infrastructure: command-line, preprocessor, include-file instances, library, and loader support

Various Activities

- Infrastructure
  - general infrastructure for analyzing binaries
  - example
    - Gogul Balakrishnan (advisor: Tom Reps)
    - Preparing object code for static analysis
      - Giving structure to activation records
  - Understanding/fixing IDAPro internals
    - IDAPro performs a variety of analysis on binaries
    - Mihai Christodorescu (advisor: Somesh Jha)
      - investigating the IDAPro SDK
      - Fixing "instruction identification"

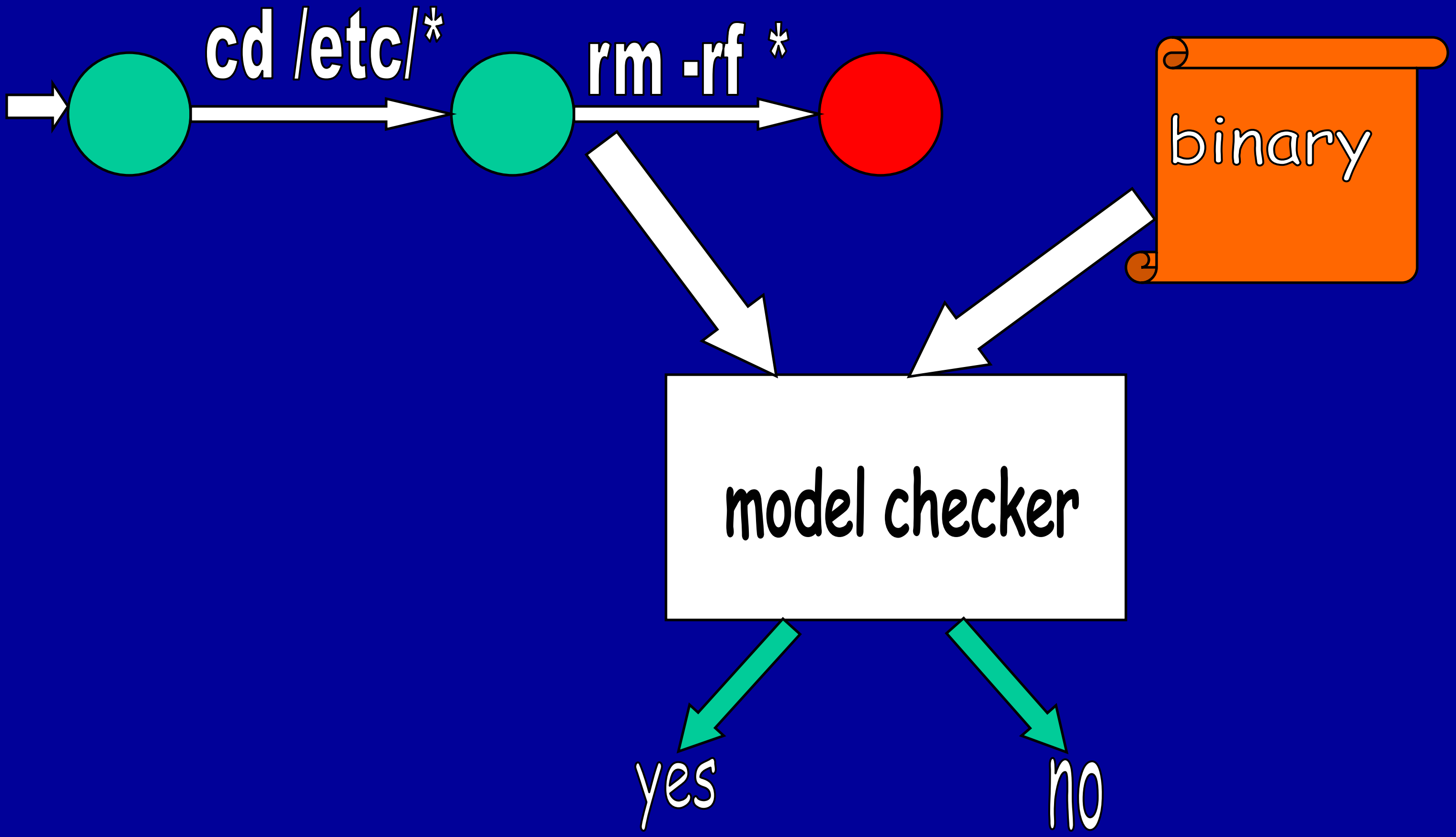
Safety Properties (Requirement 2)

- default safety conditions
  - No type violations
  - No buffer overruns
  - No misaligned loads/stores
  - No uses of uninitialized variables
  - No invalid pointer dereferences
  - No unsafe interaction with the host
- customizable safety properties
  - model checking of binaries
  - applications: smart virus scanning

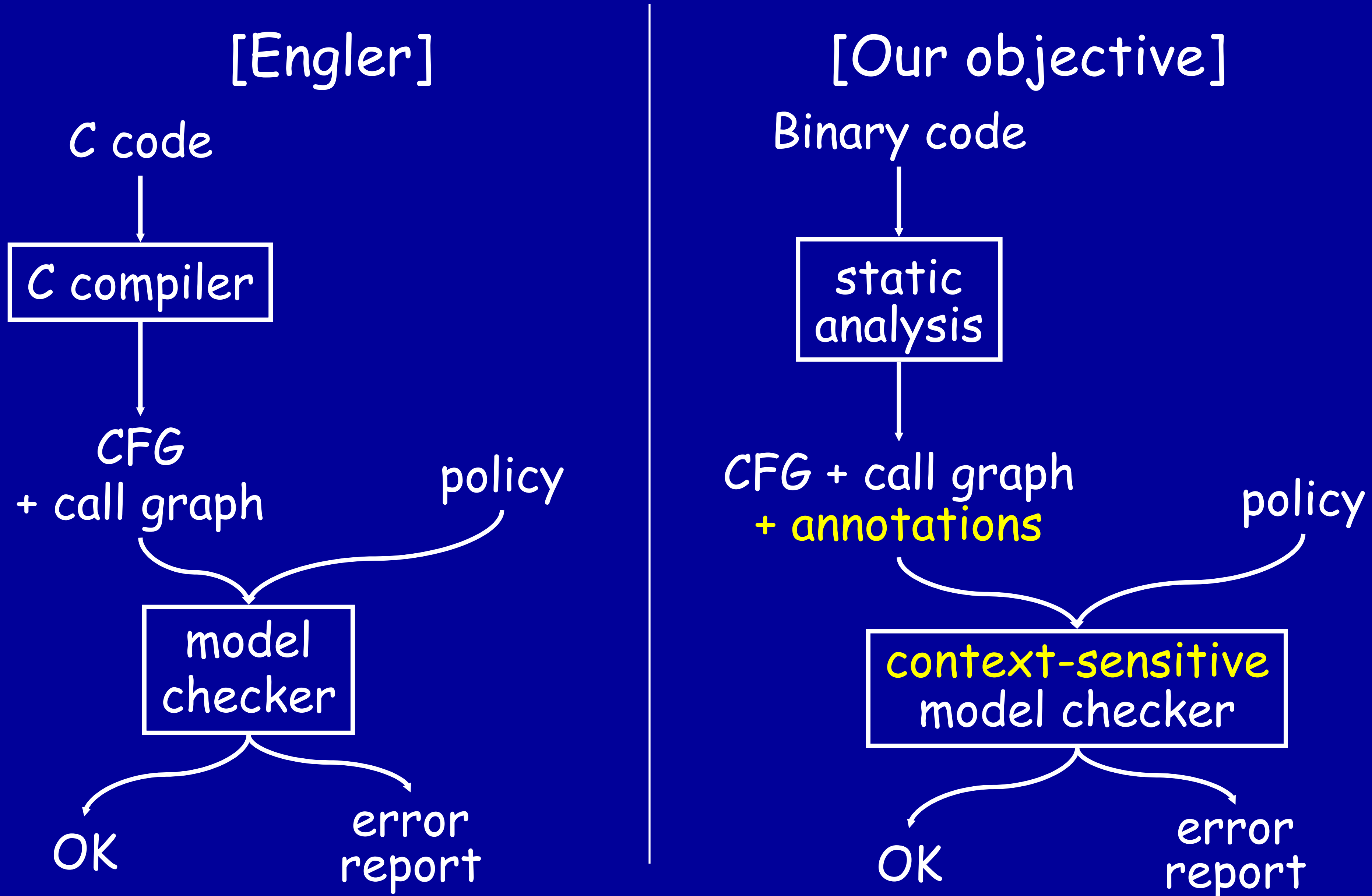
Various Activities

- Specialized analysis of binaries
  - analysis for discovering buffer overruns
  - Note: >40% of vulnerabilities in the CERT database due to buffer overrun
  - Vinod Ganapathy (advisor: Somesh Jha)
    - exploring linear programming
  - Mihai Christorescu (advisor: Somesh Jha)
    - model checking of binaries
    - application: improved scanning for viruses

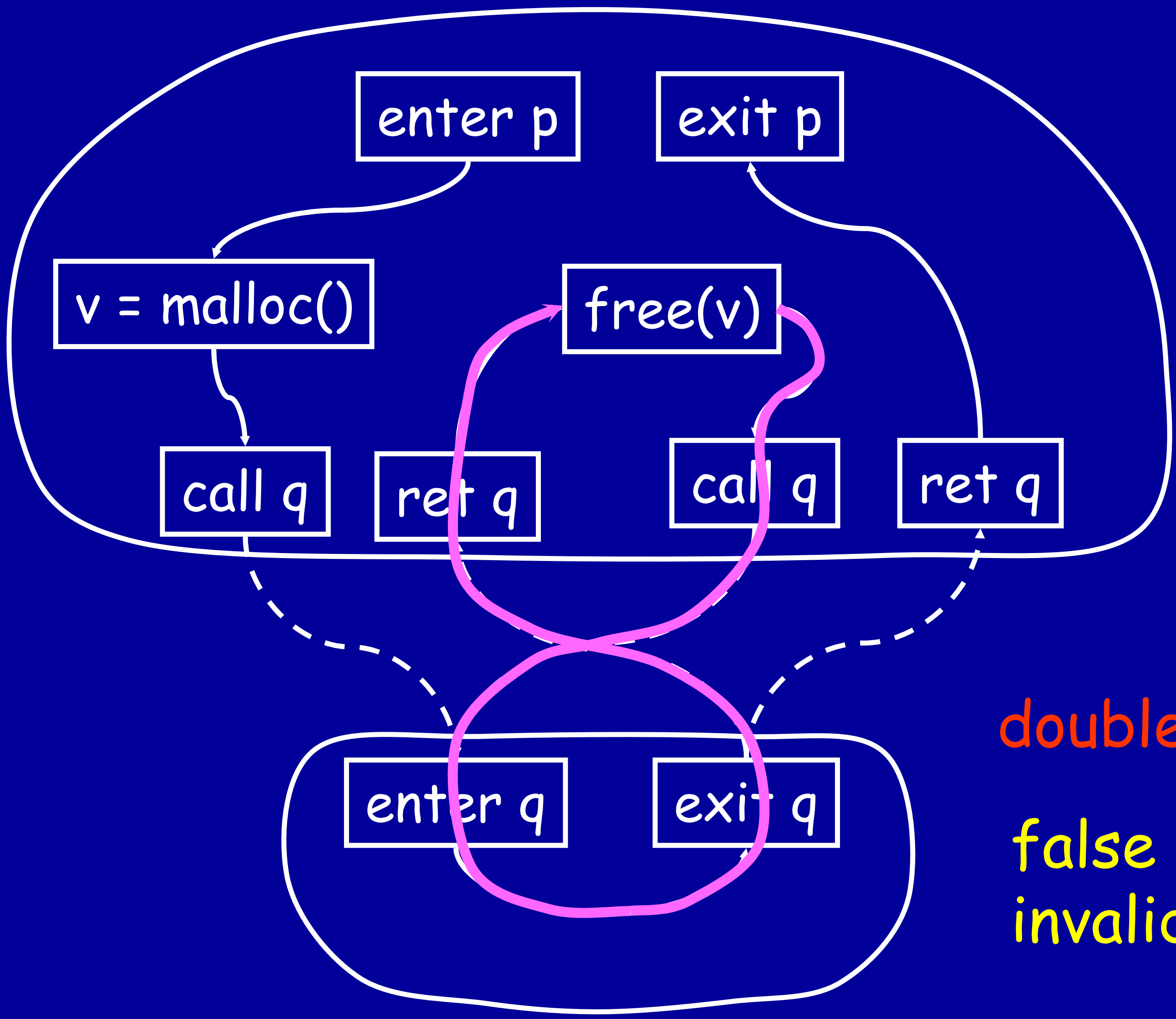
Model checking of binaries



A Richer Setting



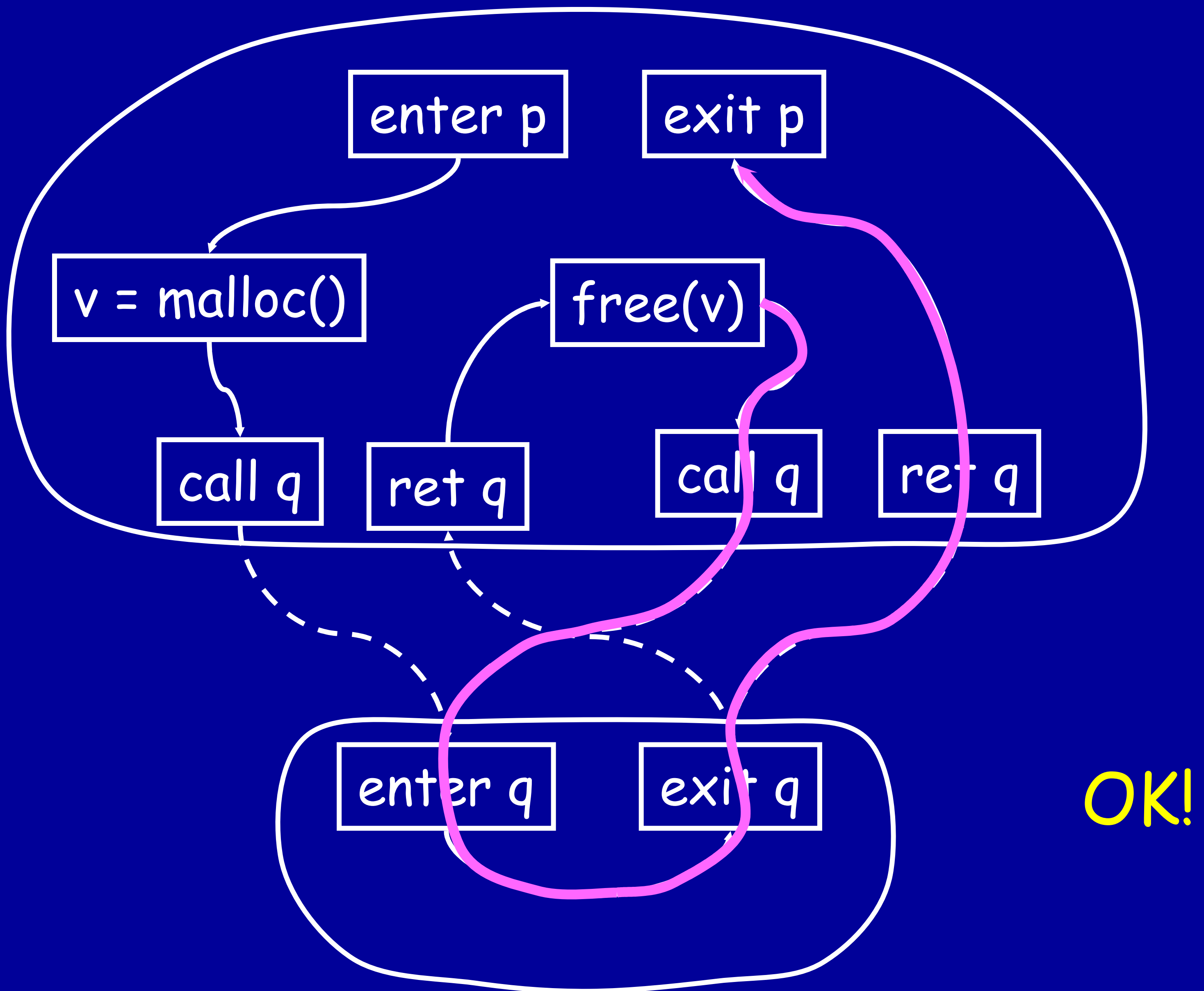
The Need for Context Sensitivity



double free!

false alarm:
invalid path!

The Need for Context Sensitivity



Analyzing Composition of COTS (Requirement 4)

- Large system composed of several components
  - (step 1) analyze individual components
  - (step 2) use vulnerabilities found in step 1 to find attacks on the entire system

- Leverage ongoing work
  - joint work with J. Wing and O. Sheyner (CMU)
  - discover attacks in a network
    - hosts "like" components
    - network "like" system

Applications of static analysis of binaries

- Applications of static analysis
  - smart virus scanning

- Secure remote execution
  - job A moves to host B (possibly malicious)
  - system calls sent to the local machine C
  - protect C from B maliciously manipulating A
  - Jon Giffin (advisors: Somesh Jha, Bart Miller)

Use specification-based monitoring
specification constructed using static analysis

Contact Information

- Prof. S. Jha
  - email: jha@cs.wisc.edu
- Prof. B. Miller
  - email: bart@cs.wisc.edu
- Prof. T. Reps
  - email: reps@cs.wisc.edu

- Computer Sciences Dept.
  1210 West Dayton Street
  Madison, WI 53706

Project home page
http://www.cs.wisc.edu/wisa