

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

Phone: 608-262-9519

FAX: 608-262-9777



Cost of Software Development Motivates Use of COTS software

- 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

COTS Spending on the Rise

- In 1991 DoD's SAI initiative mandates defense contractors to consider COTS in their programs
- Today, a significant percentage of their IT budget is allocated to COTS
- Other countries are taking similar steps



Source: Jane's Information Group

<http://www.janes.com/>

Note: IT Budget refers to total spent on

Advantages and Disadvantages of COTS

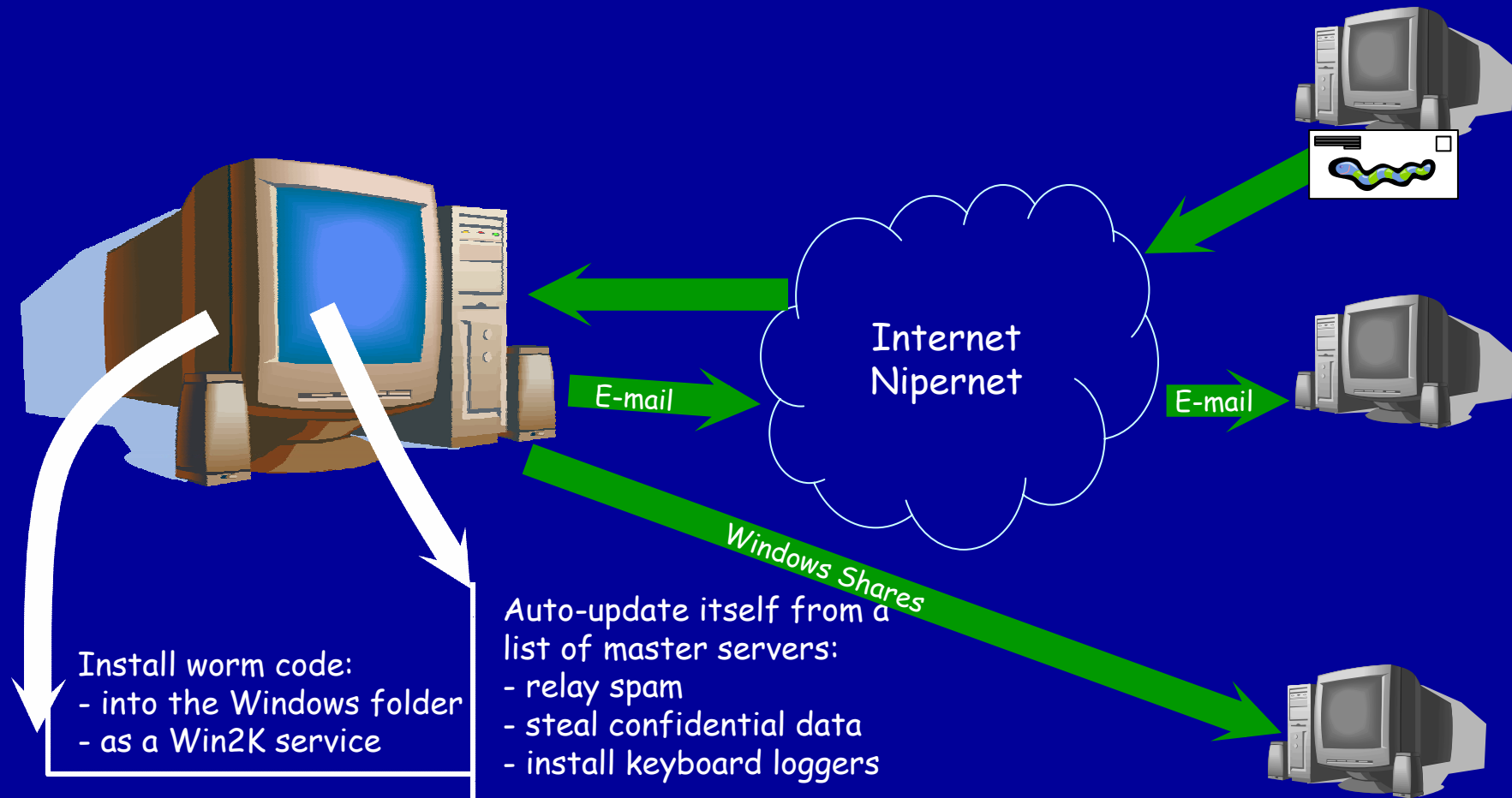
- Advantages
 - reduced cost
 - promotes modular design
 - partitions the testing effort
- Disadvantages
 - higher risk of vulnerabilities
 - general quality-assurance issues

Unsafe Malicious Code

- Viruses
 - Gain access through infected files
- Worms
 - Spread over the network
- Trojans
 - Hide harmful behavior under the guise of useful programs
- Most often: combined code
 - worm + virus + trojan
- Distinguishing characteristics: something observable happens

Malicious Code Example:

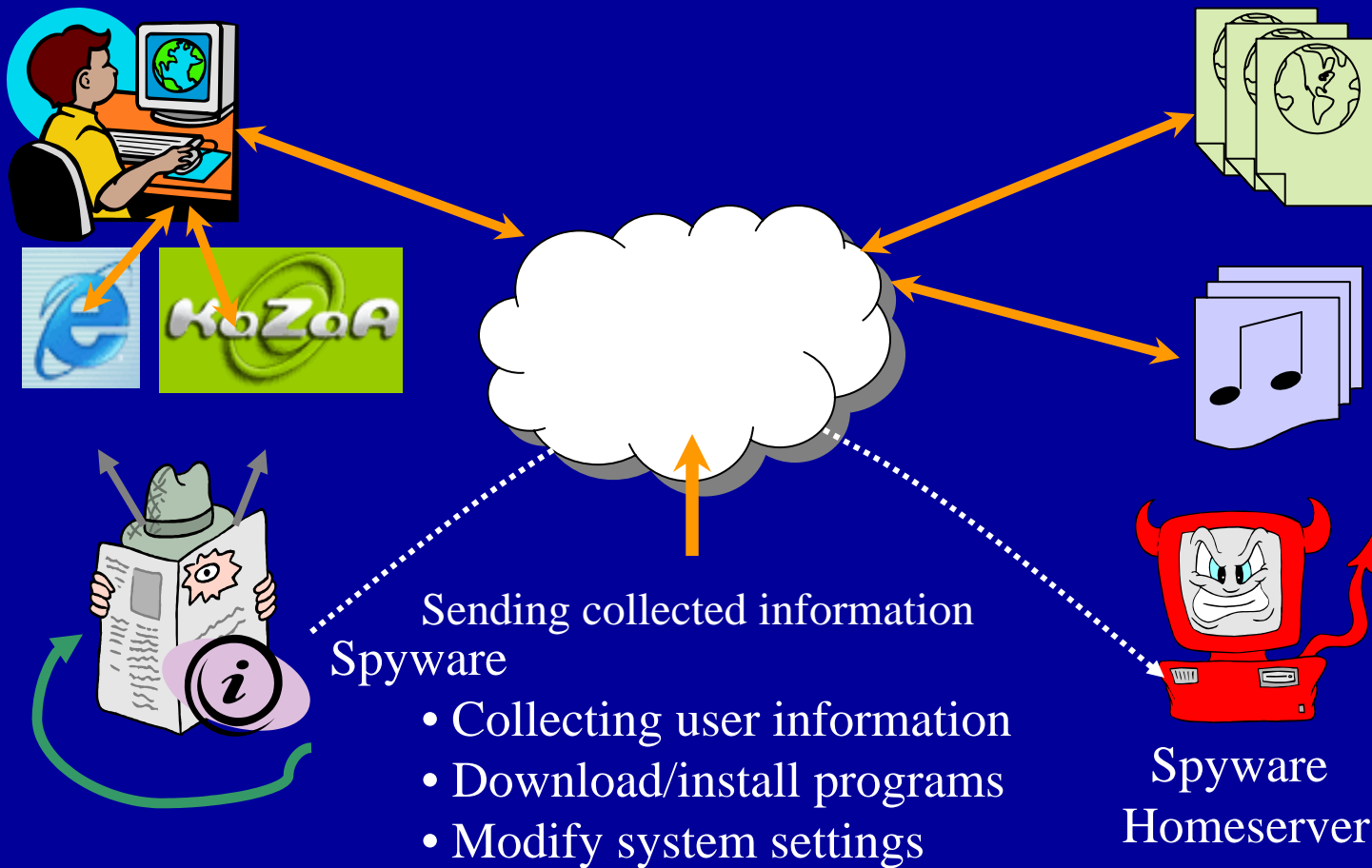
Internet worm Sobig.E



What Is Spyware?

- **Spyware** is software that
 - Is non-destructive (unlike a virus)
 - Operates in background—not easily observable
 - Is often installed silently by other software
 - Usually integrated with desired functionality
- **Privacy-violating malicious code**
 - Provides useful functionality
 - But, “leaks” sensitive information

KaZaa in Operation



Spyware Summary

- 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☺

Problems and Challenges

- Cannot expect to have source code for COTS software
 - **Solution:** we target executables
- Should handle unsafe and privacy-violating malicious code
 - **Solution:** initially targeted unsafe malicious code, but have started work on Spyware
- Certain executables are very hard to analyze statically
 - **Solution:** developed a sandboxing technology

WiSA and SandboX86: Static and Dynamic Approaches for COTS

- We have proposed the Wisconsin Safety Analyzer
 - vulnerability analysis
 - Handles unsafe malicious code
 - information flow analysis of COTS
 - Handles privacy-violating malicious code (Spyware)
- Develop technology for static and dynamic analysis of binaries
 - Original plan to focus on static analysis
 - Realized that we need multiple-lines of defense
 - Started working on dynamic analysis as well and developed a sandboxing system called SandboX86
- Investigate applications

Tools for Reducing the Risk of COTS Deployment

Static analysis and rewriting of executables

Sandboxing and dynamic slicing

Evaluation and testing

Tools for Reducing the Risk of COTS Deployment

Static analysis and rewriting of executables

Malicious code detection
Model-based HIDS
Program Obfuscation

Sandboxing and dynamic slicing

Containing malicious behavior
Discovering potential privacy violations

Evaluation and testing

Testing malware detectors
Testing NIDS

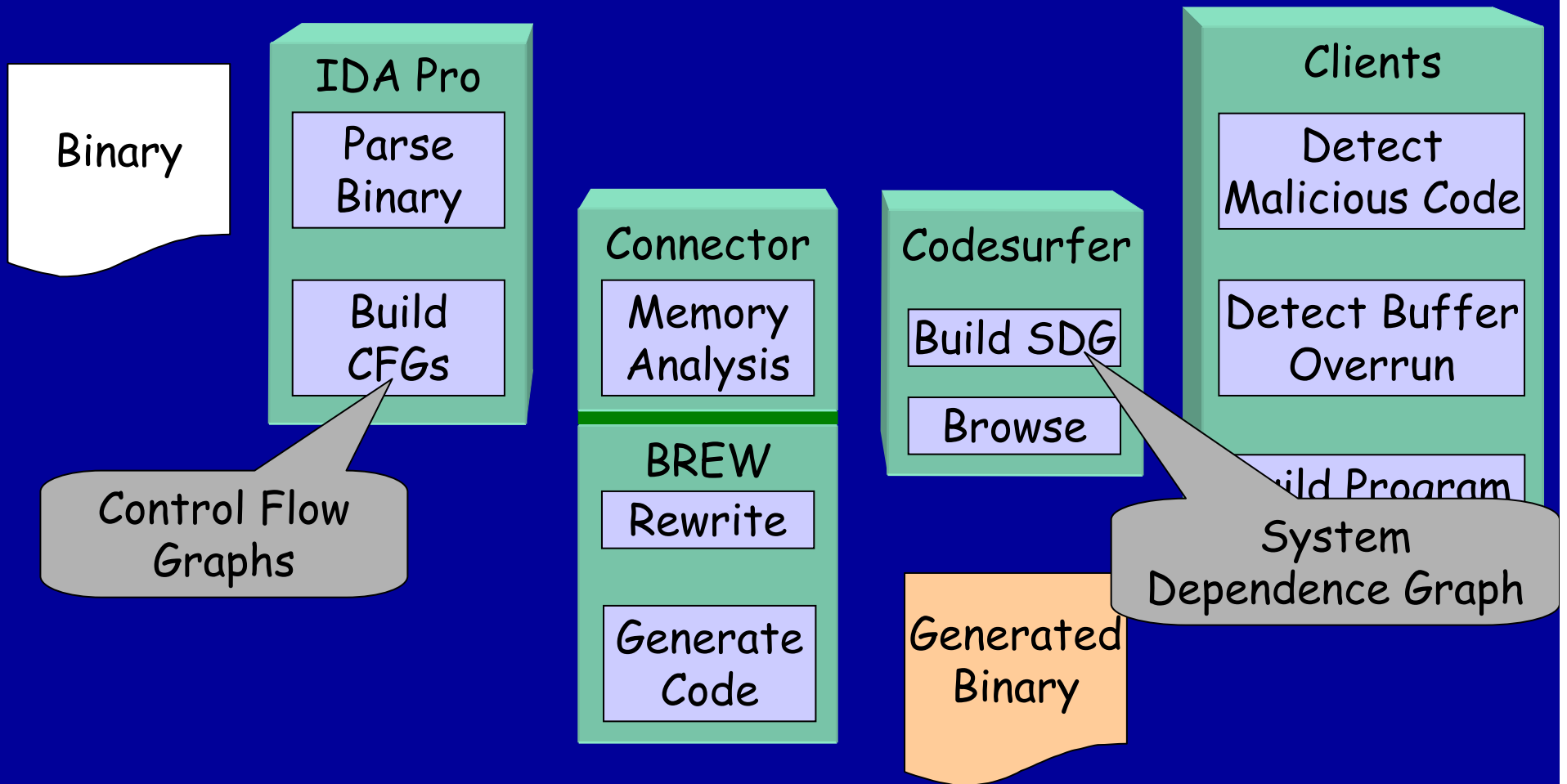
IDA Pro

- 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

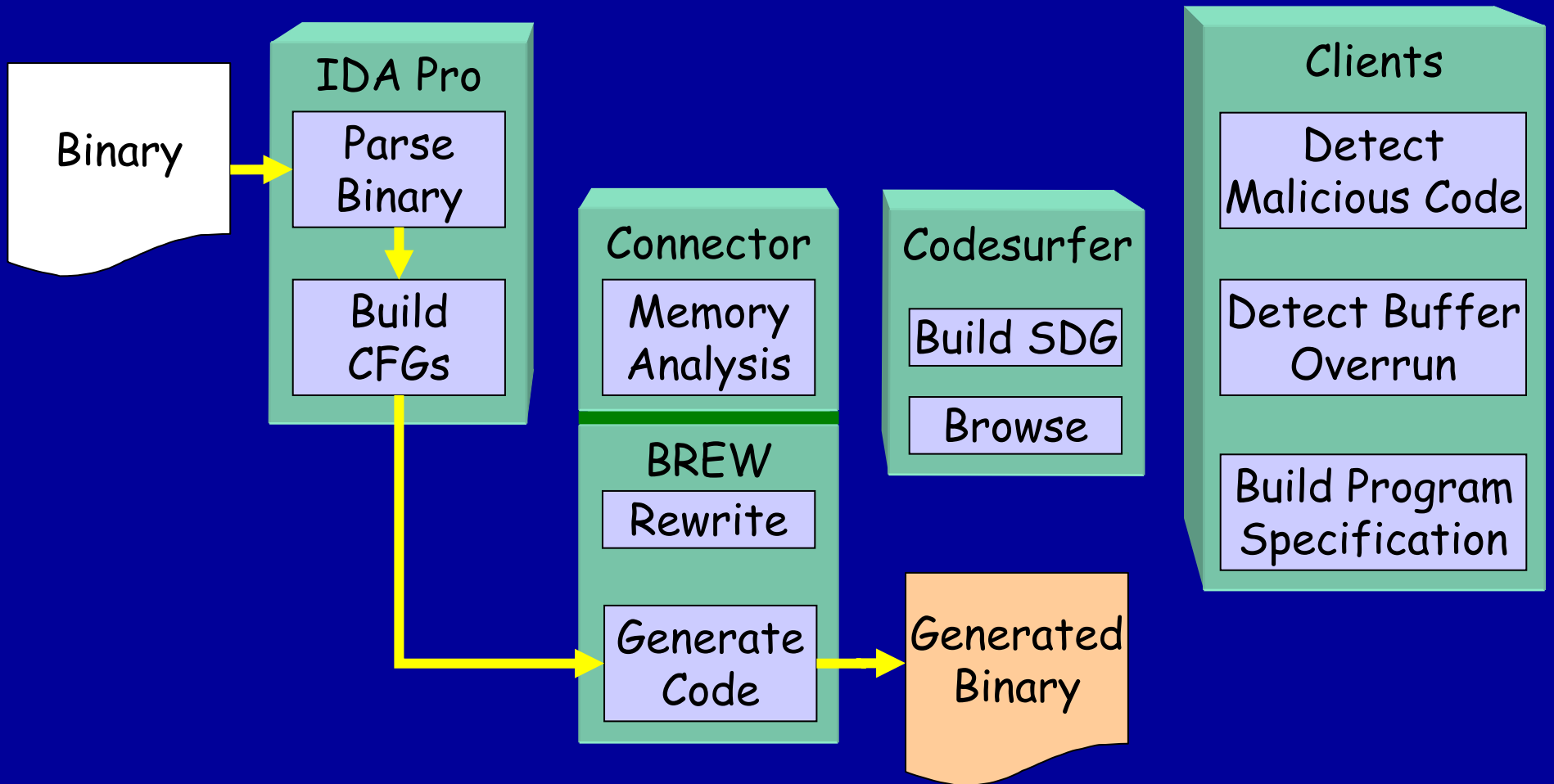
Codesurfer

- A program-understanding tool
- Analyzes the data and control dependences
 - stores in System Dependence Graph(SDG)
 - Helpful in static analysis
- API to access information stored in IRs
 - Platform for additional static analysis
- The API can be extended

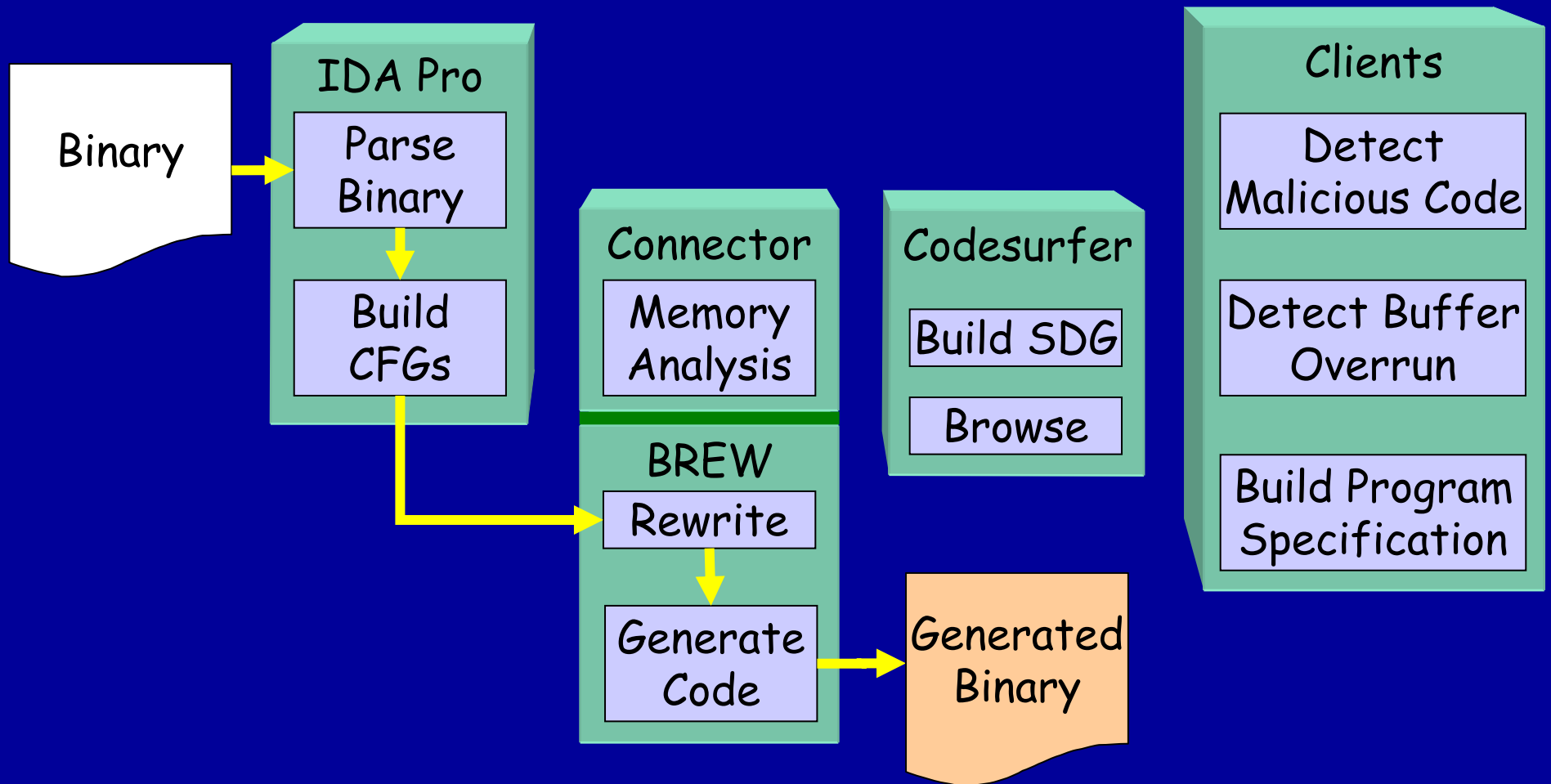
Architecture



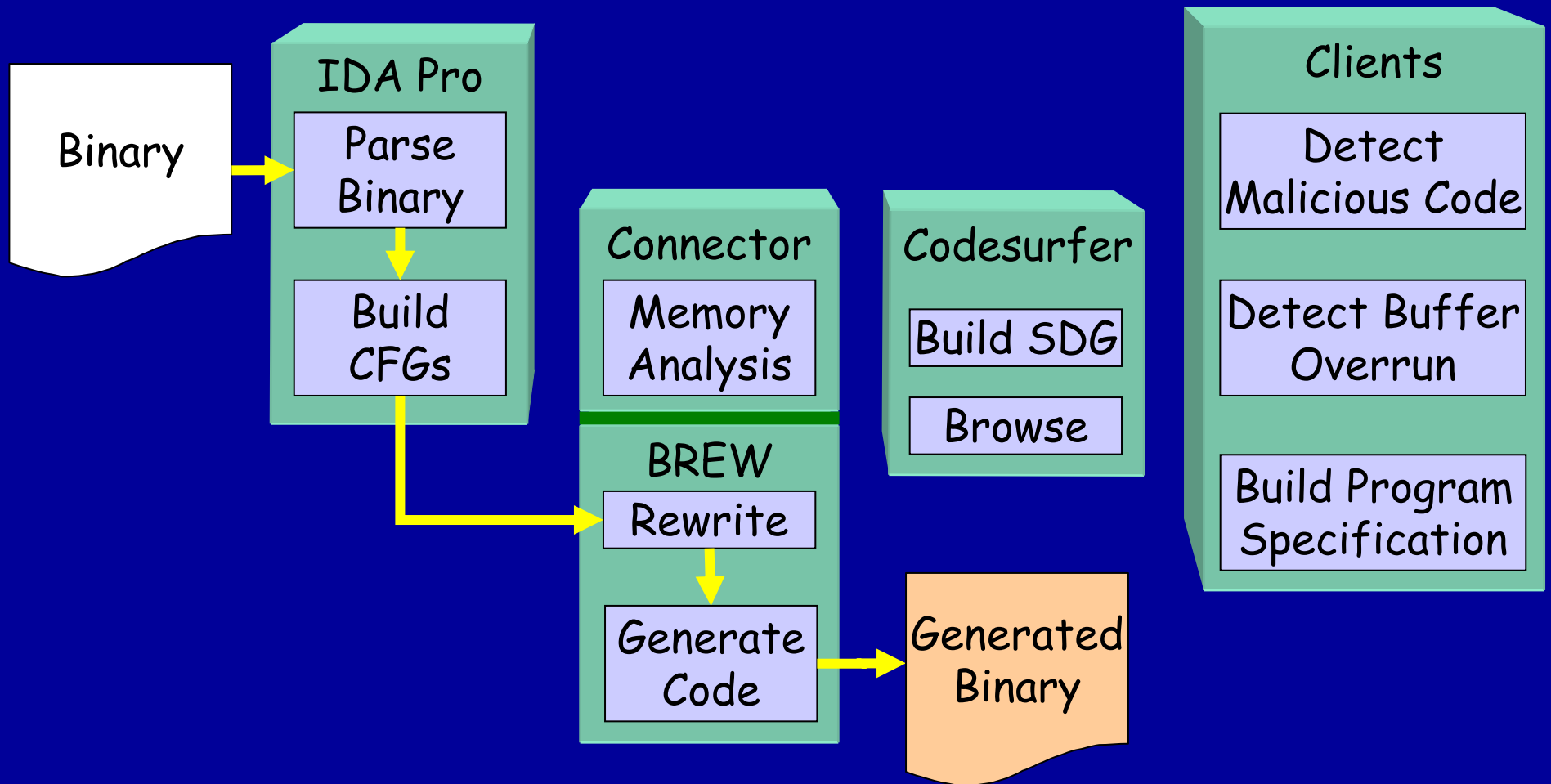
Code Generation



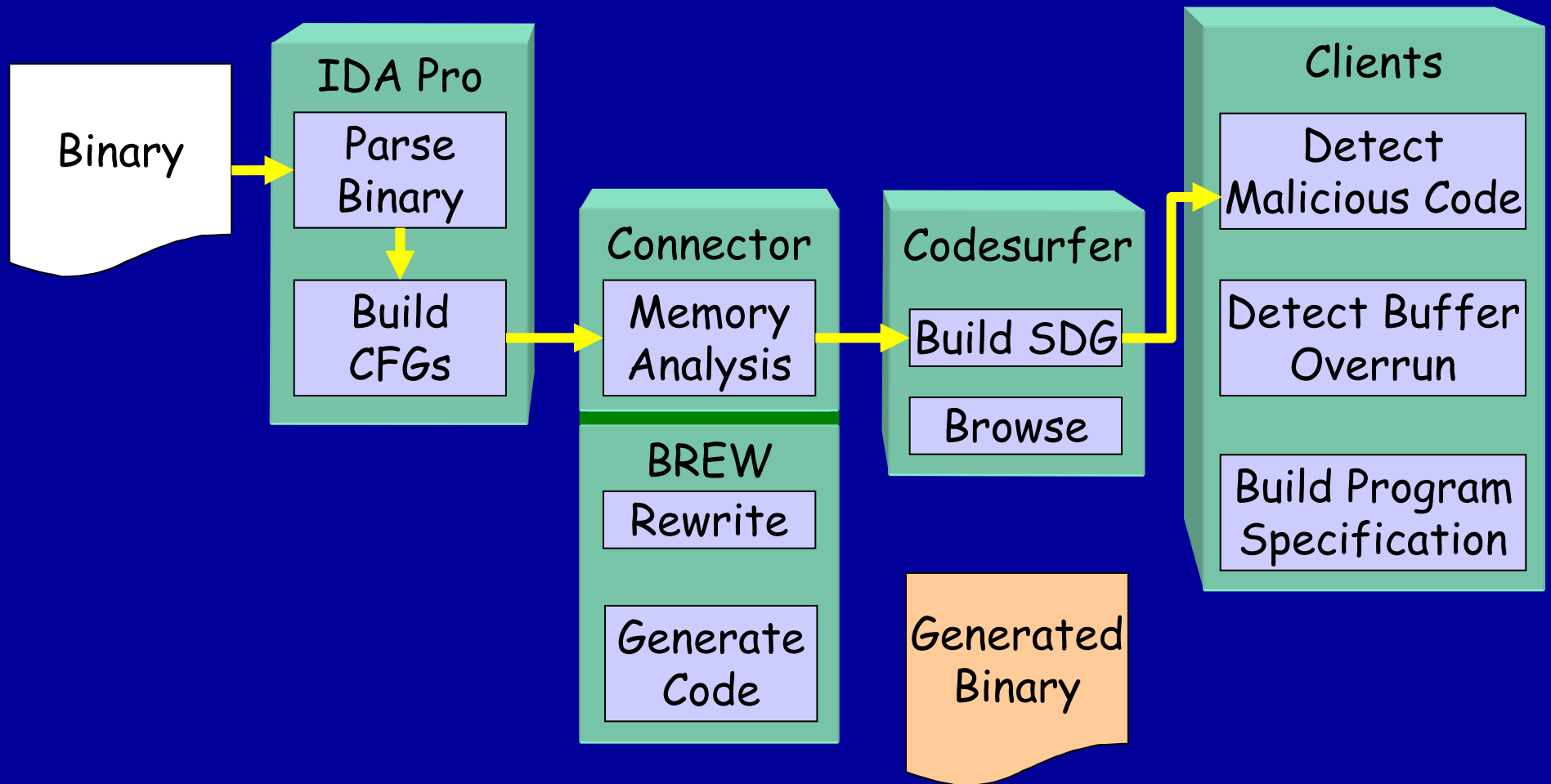
Binary Code Rewriting



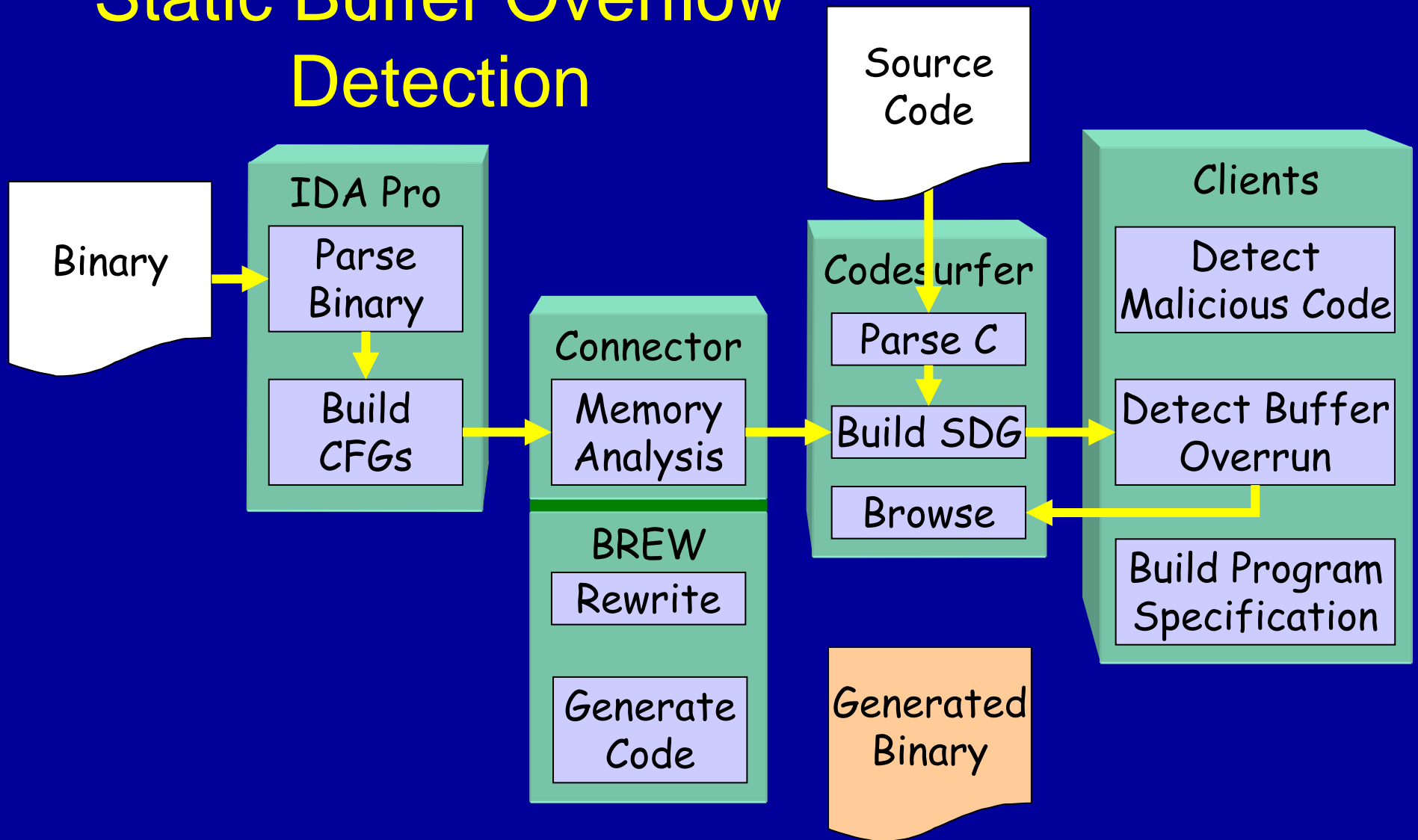
Dynamic Buffer Overflow Detection



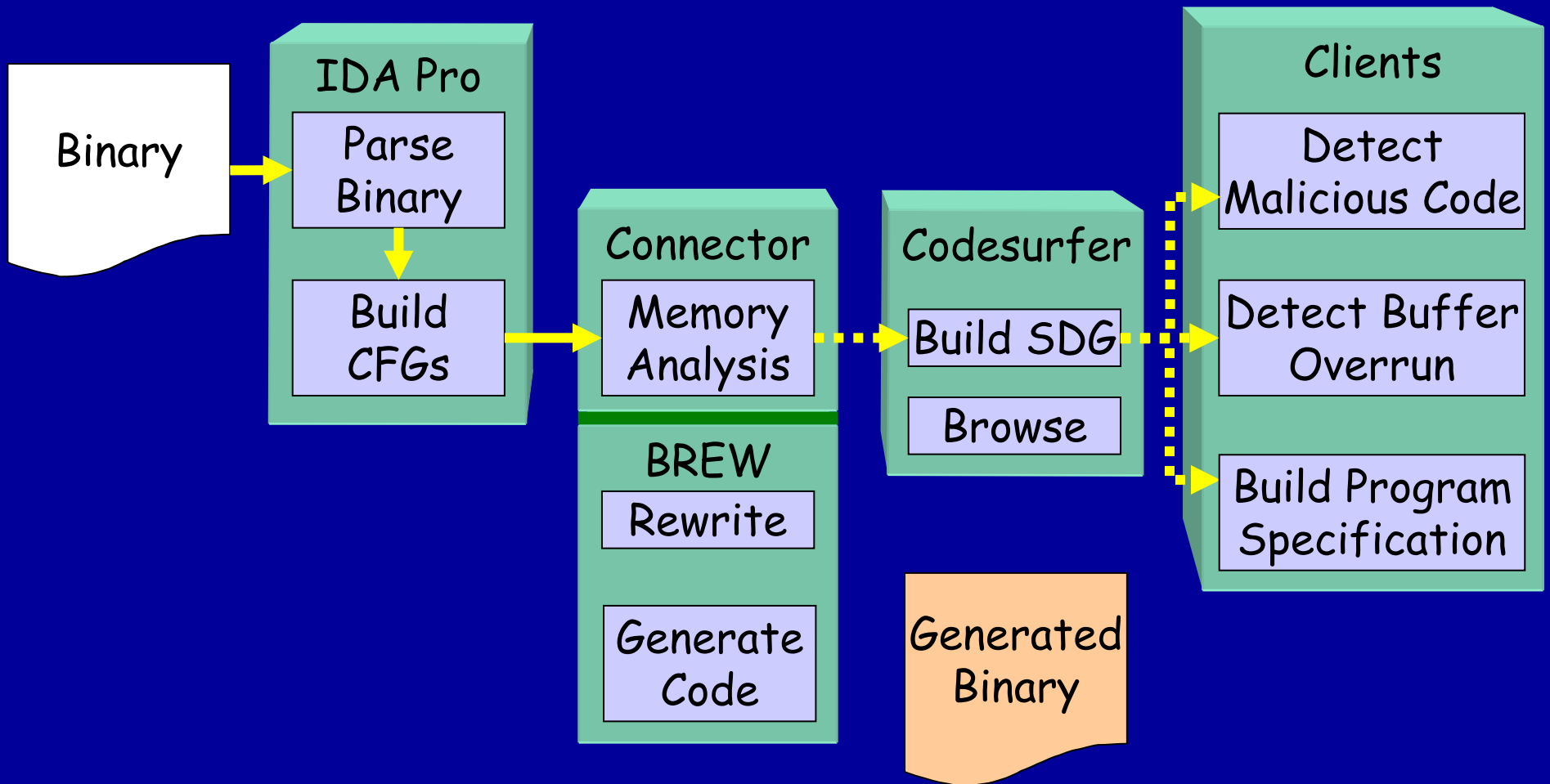
Malicious Code Detection



Static Buffer Overflow Detection



Value Set Analysis



Example – Value-Set Analysis

```
int arrVal=0, *pArray2;

int main() {
    int i, a[10], *p;
    /* Initialize pointers */
    pArray2 = &a[2];
    p = &a[0];
    /* Initialize Array */
    for(i = 0; i<10; ++i) {
        *p = arrVal;
        p++;
    }
    /* Return a[2] */
    return *pArray2;
}
```

```
|
| ; ebx ⇔ variable i
| ; ecx ⇔ variable p
|
| sub    esp, 40          ;adjust stack
| lea    edx, [esp+8]    ;
| mov    [8], edx        ;pArray2=&a[2]
| lea    ecx, [esp]      ;p=&a[0]
| mov    edx, [4]        ;
|
| loc_9:
|     mov    [ecx], edx   ;*p=arrVal
|     add    ecx, 4       ;p++
|     inc    ebx          ;i++
|     cmp    ebx, 10      ;i<10?
|     jl     short loc_9 ;
|
|     mov    edi, [8]     ;
|     mov    eax, [edi]   ;return *pArray2
|     add    esp, 40
|     retn
```

Example – Value-Set Analysis

```
int arrVal=0, *pArray2;

int main() {
    int i, a[10], *p;
    /* Initialize pointers */
    pArray2 = &a[2];
    p = &a[0];
    /* Initialize Array */
    for(i = 0; i<10; ++i) {
        *p = arrVal;
        p++;
    }
    /* Return a[2] */
    return *pArray2;
}
```

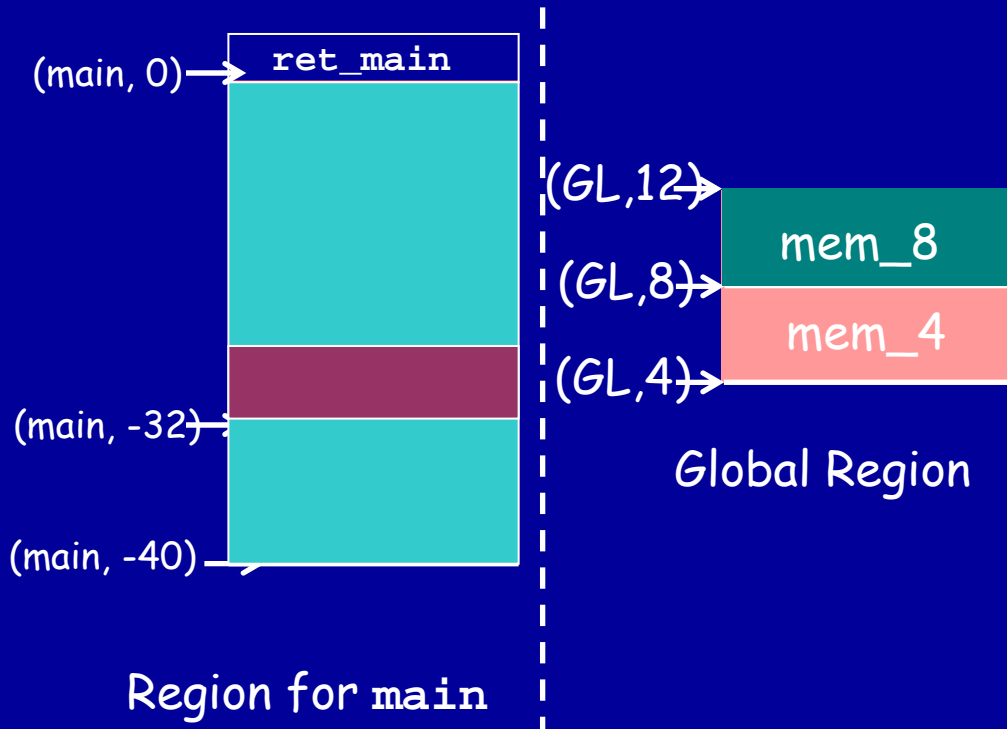
```
; ebx ⇔ variable i
; ecx ⇔ variable p

sub    esp, 40        ;adjust stack
lea    edx, [esp+8]  ;
mov    [8], edx      ;pArray2=&a[2]
lea    ecx, [esp]    ;p=&a[0]
mov    edx, [4]      ;

loc_9:
mov    [ecx], edx ;*p=arrVal
add    ecx, 4        ;p++
inc    ebx           ;i++
cmp    ebx, 10       ;i<10?
jnl   short loc_9   ;

mov    edi, [8]      ;
mov    eax, [edi] ;return *pArray2
add    esp, 40
retn
```


Example – Value-Set Analysis



1 → **ecx** → (\emptyset , 4[0,9]-40)

2 → **edi** → (\emptyset , -32)

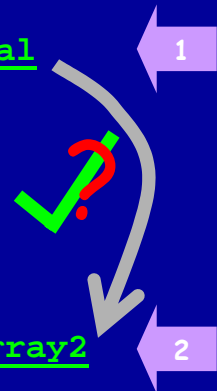
; ebx \leftrightarrow variable i
; ecx \leftrightarrow variable p

```
sub    esp, 40      ;adjust stack
lea    edx, [esp+8] ;
mov    [8], edx    ;pArray2=&a[2]
lea    ecx, [esp]  ;p=&a[0]
mov    edx, [4]    ;
```

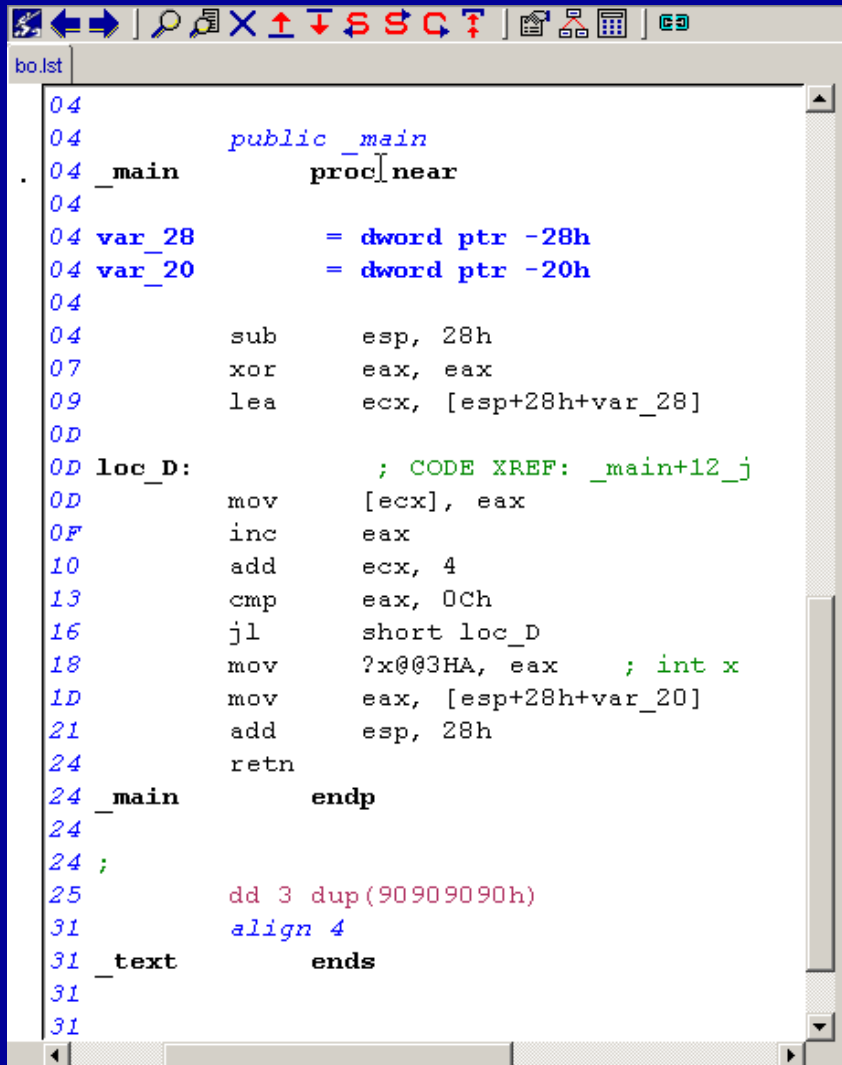
loc_9:

```
mov    [ecx], edx ;*p=arrVal
add    ecx, 4     ;p++
inc    ebx        ;i++
cmp    ebx, 10    ;i<10?
j1     short loc_9 ;
```

```
mov    edi, [8]   ;
mov    eax, [edi] ;return *pArray2
add    esp, 40
retn
```

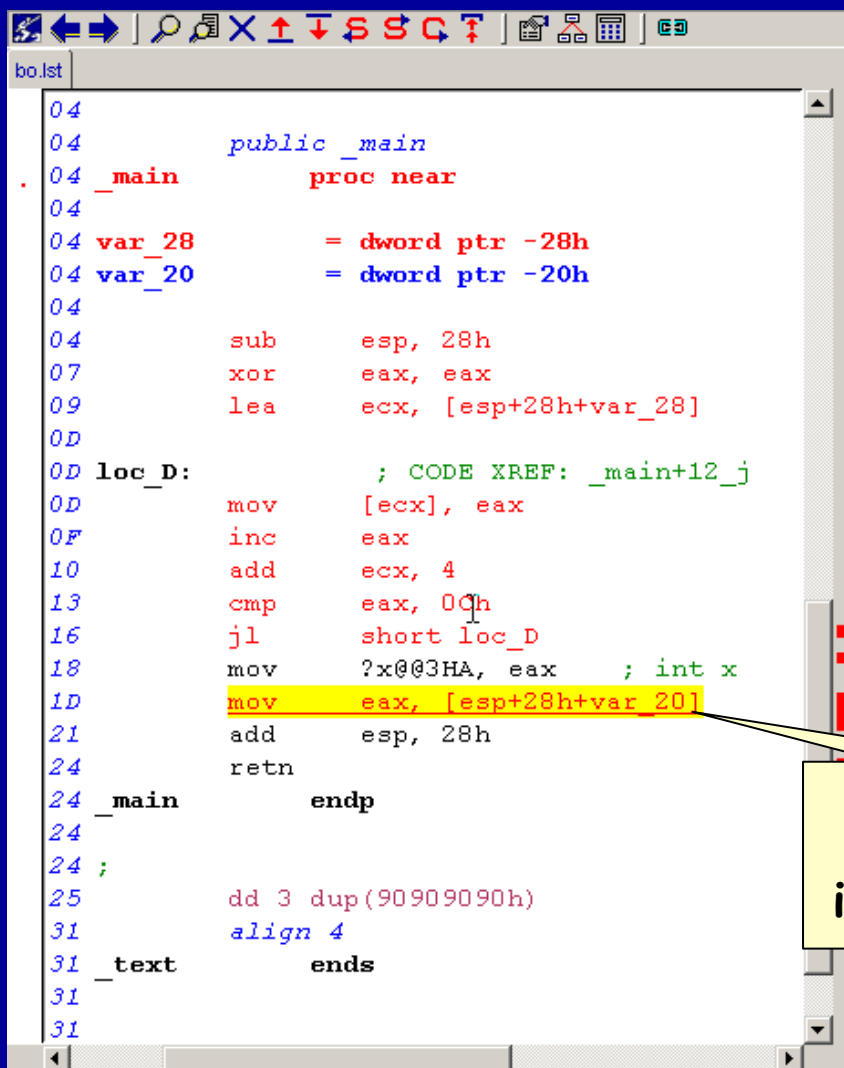


CodeSurfer/x86 Tool



```
bo.lst
04
04      public _main
04 _main      proc near
04
04 var_28      = dword ptr -28h
04 var_20      = dword ptr -20h
04
04      sub     esp, 28h
07      xor     eax, eax
09      lea    ecx, [esp+28h+var_28]
0D
0D loc_D:      ; CODE XREF: _main+12_j
0D      mov     [ecx], eax
0F      inc     eax
10      add     ecx, 4
13      cmp     eax, 0Ch
16      jnl    short loc_D
18      mov     ?x@@3HA, eax ; int x
1D      mov     eax, [esp+28h+var_20]
21      add     esp, 28h
24      retn
24 _main      endp
24
24 ;
25      dd 3 dup(90909090h)
31      align 4
31 _text      ends
31
31
```

CodeSurfer/x86 Tool



```
bo.lst
04
04      public _main
04 _main      proc near
04
04 var_28      = dword ptr -28h
04 var_20      = dword ptr -20h
04
04      sub     esp, 28h
07      xor     eax, eax
09      lea    ecx, [esp+28h+var_28]
0D
0D loc_D:      ; CODE XREF: _main+12_j
0D      mov     [ecx], eax
0F      inc     eax
10      add     ecx, 4
13      cmp     eax, 0Ch
16      jl     short loc_D
18      mov     ?x@@3HA, eax ; int x
1D      mov     eax, [esp+28h+var_20]
21      add     esp, 28h
24      retn
24 _main      endp
24 ;
25      dd 3 dup(90909090h)
31      align 4
31 _text      ends
31
31
```

Backward slice
with respect to
instruction at 1Dh

CodeSurfer/x86 Tool

```
bo.lst
04
04      public _main
04 _main      proc near
04
04 var_28      = dword ptr -28h
04 var_20      = dword ptr -20h
04
04      sub     esp, 28h
07      xor     eax, eax
09      lea    ecx, [esp+28h+var_28]
0D
0D loc_D:      : CODE_XREF: _main+12_j
0D      mov     [ecx], eax
0F      in     eax, eax
10      ecx, 4
13      eax, 0Ch
16      short loc_D
18      ?x@@3HA, eax      ; int x
                                ; _20]
24      retn
24 _main      endp
24 ;
25      dd 3 dup(90909090h)
31      align 4
31 _text      ends
31
31
```

Possible buffer-overflow

Program point: loc_D: mov [ecx], eax

Program point kind: expression

Basic | Predecessors | Successors | CFG

Killed

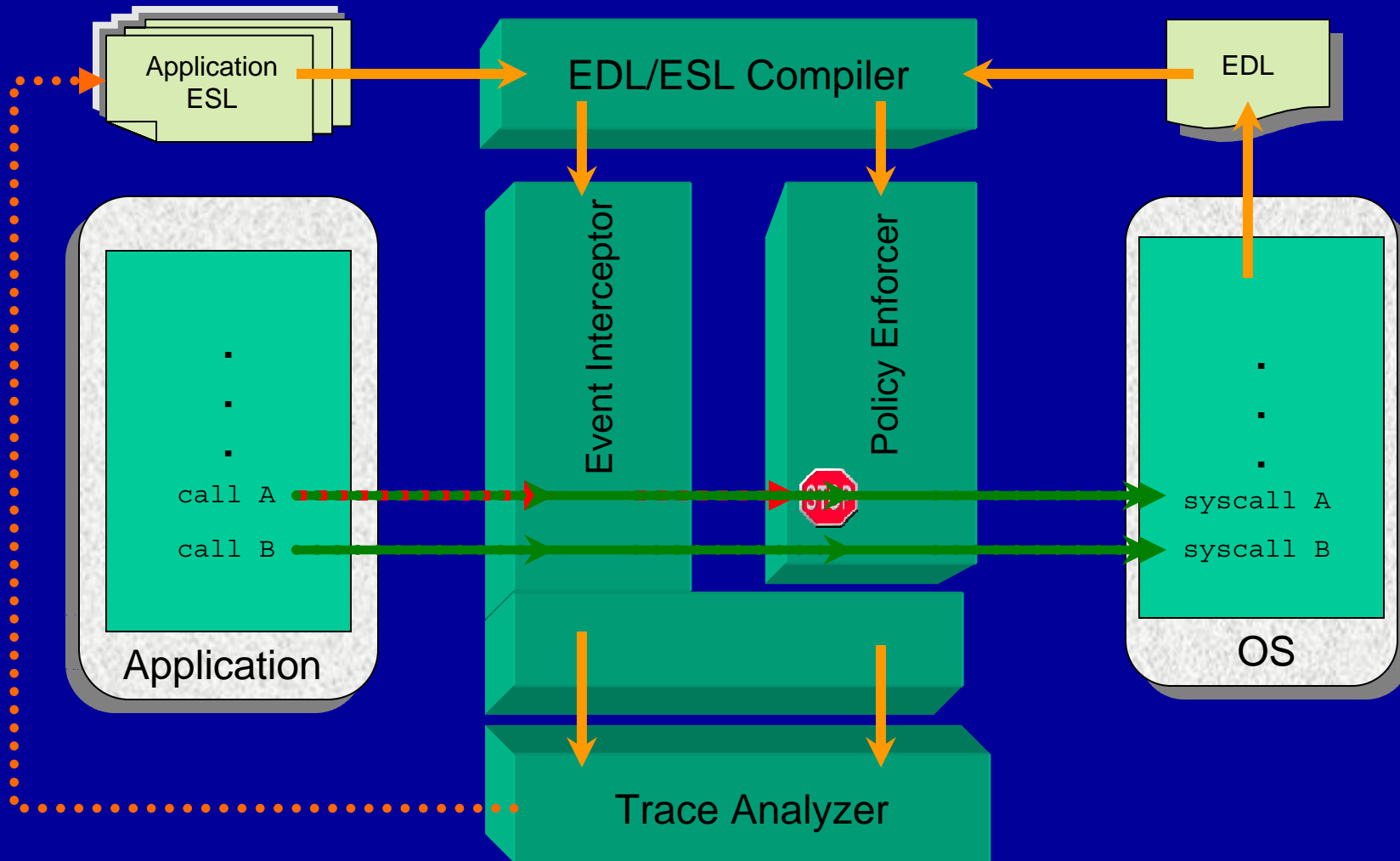
Conditional Killed

- (Function Static) return-addr\$
- (Local) var_20
- (Local) var_28

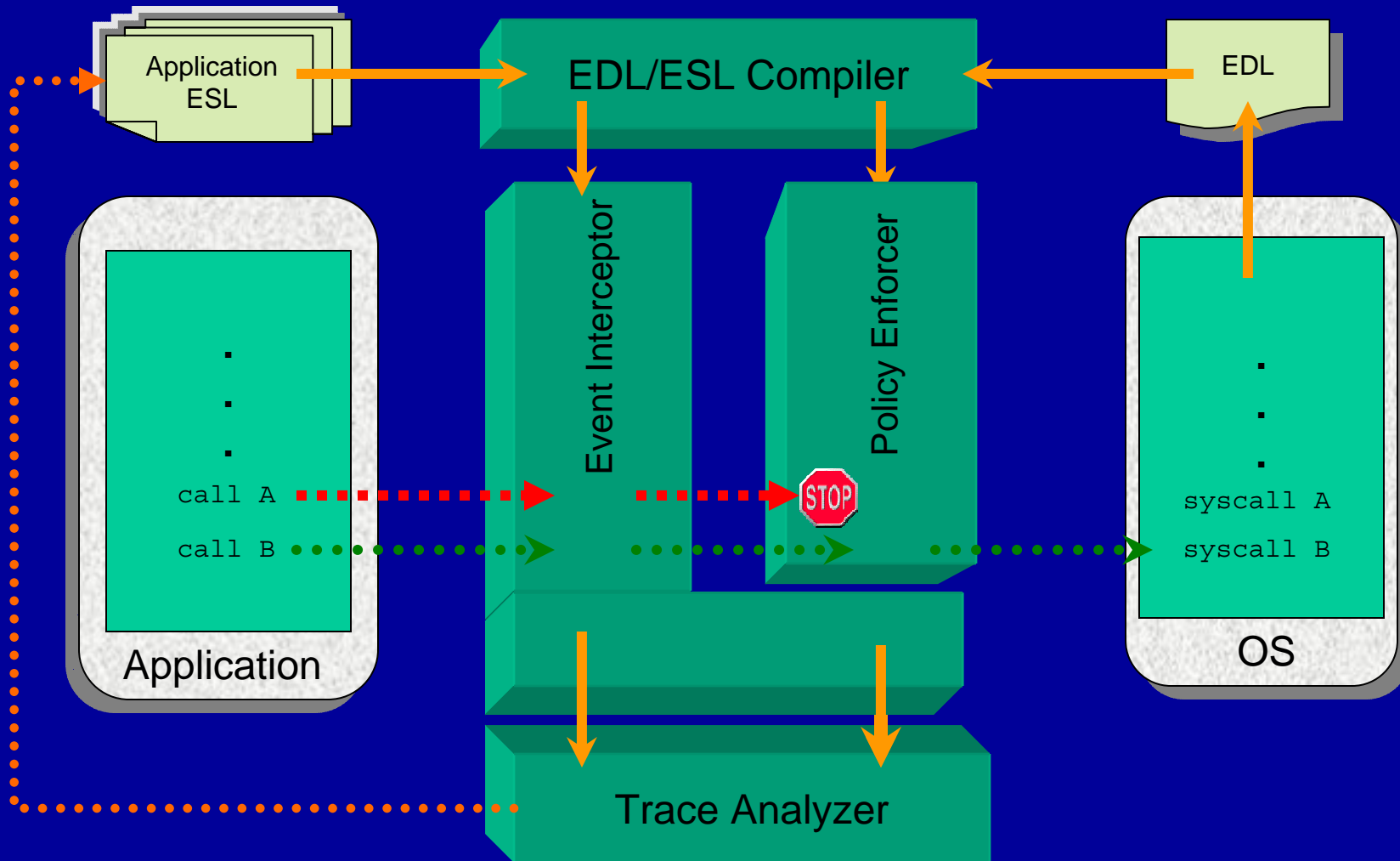
Close | Help | Use Case Window

Return address modified at loc_F

Sandboxing Architecture: SandboX86



SandboX86



Posters and Demo

- Posters
 - Codesurfer for x86 (T. Reps)
 - Security testing using threat models (M. Christodorescu)
 - Efficient context-sensitive intrusion detection (J. Giffin)
- Demos
 - Code patching using BREW

Team

- Somesh Jha
 - Analysis of malicious code, intrusion detection, verification of security protocols, and trust management
- Bart Miller
 - Distributed computing, kernel instrumentation, intrusion detection
- Tom Reps
 - Static-analysis techniques, trust management, and model checking

Six Graduate Students

- Gogul Balakrishnan
- Mihai Christodorescu (US citizen)
- Vinod Ganapathy
- Jon Giffin (US citizen)
- Shai Rubin (Prelim)
- Hao Wang (US citizen)
- Summary
 - Three US citizens
 - All are Ph.D. students and have passed their qualifiers
 - Three students very close to their prelims

Science and
Technology

Research

DoD Relevance

Education

- Research Papers
 - 12 papers accepted in major conferences (USENIX Security, Oakland, CCS, NDSS, CSFW)
 - 2 under submission
 - > 10 related publications
- PIs served on several program committees and reviewed for several journals
- See the overview document for details

Science and
Technology

Technology

DoD Relevance

Education

- Developed a significant infrastructure for analyzing and rewriting x86 binaries
 - Collaboration with GrammaTech
- Applicable to several research problems
 - Identifying buffer overruns
 - Malicious code detection
 - Protection, event logging, remediation..
- Created many technology-transfer and collaborative opportunities

Science and
Technology

Technology

DoD Relevance

Education

- Developed a significant infrastructure for sandboxing Windows applications
 - Enforce a security policy at the interface between the application and OS
- Developed a dynamic-slicing tool to discover dependences between events
 - Used to discover potential spyware features in applications
 - Form of information flow
- Applications and research
 - Sandbox popular applications (KaZaa and RealOne Player)

Science and
Technology

DoD Relevance

Education

Tools

- WiSA infrastructure
 - Discovering buffer overruns
 - Malicious-code detection
 - Constructing models for intrusion detection
 - Many more under development ...
- SandboX86
 - Sandbox applications using a security policy
 - Discovering spyware features in unknown applications
- Our analysis techniques do not require access to source code
 - Can be readily applied to COTS software
- Reduces risk of deploying COTS

Science and
Technology

DoD Relevance

Education

Tech Transfer

- GrammaTech (GT) an important vehicle for technology transfer
- GT → UW
 - GT implemented an important piece of the architecture
- UW → GT
 - Value-set analysis (Gogul)
 - BREW infrastructure (Jon, Mihai, and Hao)
 - Buffer-overflow-detection tool (Vinod)

Science and
Technology

DoD Relevance

Education

Tech Transfer

- Starting to explore collaborative opportunities with **Sandia National Laboratories**
 - **System Assessment and Research Center**
- Doug Ghormley from Sandia came and gave a talk
- Louis Kruger (UW) is a summer intern at Sandia
 - Working on using BREW for "classified" applications

SAFE for Software Protection

- DoD Anti Tamper and Software Protection Initiative (Dec. 2001)



- AFRL S/W Protection Compilation (Nov. 2003)
 - Workshop to develop a framework to use compilers for software protection
 - SAFE research presentation

SAFE for Exploit Classification



- ATL is planning to develop an intrusion-tolerant system based on biological metaphors
- Advanced Technology Laboratories
(Cherry Hill, NJ)
 - Interested in using SAFE technology to classify exploit code
- Meeting in October 2003 established feasibility of approach

Science and
Technology

DoD Relevance

Education

Ph.D Students

- Gogul Balakrishnan
 - **Status:** Passed qualifiers in programming languages (PL)
 - **Subject:** Static analysis of executables
 - **Advisor:** Tom Reps
- Mihai Christodorescu
 - **Status:** Passed qualifiers in PL
 - **Subject:** Malicious code detection
 - **Advisor:** Somesh Jha
- Vinod Ganapathy
 - **Status:** Passed qualifiers in PL
 - **Subject:** Verifying security APIs
 - **Advisor:** Somesh Jha

Science and
Technology

DoD Relevance

Education

Ph.D Students

- Jon Giffin
 - **Status:** Passed qualifiers in operating systems (OS)
 - **Subject:** Static analysis techniques for intrusion detection
 - **Advisors:** Somesh Jha and Bart Miller
- Shai Rubin
 - **Status:** Passed qualifiers in PL
 - **Subject:** Formalizing network intrusion detection systems (NIDS)
 - **Advisors:** Somesh Jha and Bart Miller
- Hao Wang
 - **Status:** Passed qualifiers in OS
 - **Subject:** Detecting and containing Spyware
 - **Advisor:** Somesh Jha

Science and
Technology

DoD Relevance

Education

Courses

- Introduction to Information Security
 - Audience: Seniors
 - Topics covered
 - Basic cryptography
 - Various attacks and malicious code
 - Security protocols
 - System security (firewalls and IDSs)
 - Instructor: Somesh Jha
- Analysis of Software Artifacts
 - Audience: Graduate students
 - Topics covered
 - Model checking
 - Other formal methods (SCR, Alloy, ...)
 - Other assorted topics (real-time systems, ...)
 - Analysis techniques for security properties
 - Instructor: Somesh Jha

Science and
Technology

DoD Relevance

Education

Courses

- Distributed Systems
 - Audience: Graduate students
 - Topics covered
 - Language issues
 - Distributed shared memory
 - Replication and fault tolerance
 - Authentication
 - Mobile computing
 - Instructors: Bart Miller
- Other related course taught by B. Miller and T. Reps

Science and
Technology

DoD Relevance

Education

Seminars

- Established a security seminar series
 - Several external speakers presented on various topics related to INFOSEC
 - Several internal speakers presented their work and some recent work by others
 - Topics covered
 - Applied cryptography
 - Watermarking
 - Legal issues such as DMCA

Science and
Technology

DoD Relevance

Education

Seminars

- Distinguished lecture series was organized by Somesh Jha has a security focus
 - Amir Pnueli
 - Fred Schneider
 - David Dill
 - Dan Boneh
 - Doug Tygar
- Established a security reading group
 - Mostly graduate students
 - Read papers from major conferences (Oakland, CCS, Usenix Security)
 - Read some classic papers (suggested by Connie Heitmeyer and Jon McHugh at the Williamsburg meeting)

Order of Presentations

- **Somesh Jha:** WiSA Architecture Overview and Applications
 - Analysis of executables
 - Sandboxing applications
- **Tom Reps:** Static Analysis of x86 Binaries
- **Bart Miller:** Attacks and Defenses
- **Somesh Jha and Tim Teitelbaum (GT):** Wrap-up
- **Afternoon:** Demos and posters by students

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>