DEFENCE R&D DÉFENSE

DEFENCE R&D DÉFENSE

Secure Software from Design to Binary

Martin Salois & Robert Charpentier

R et D pour la défense Canada
Defence R&D Canada
Canada

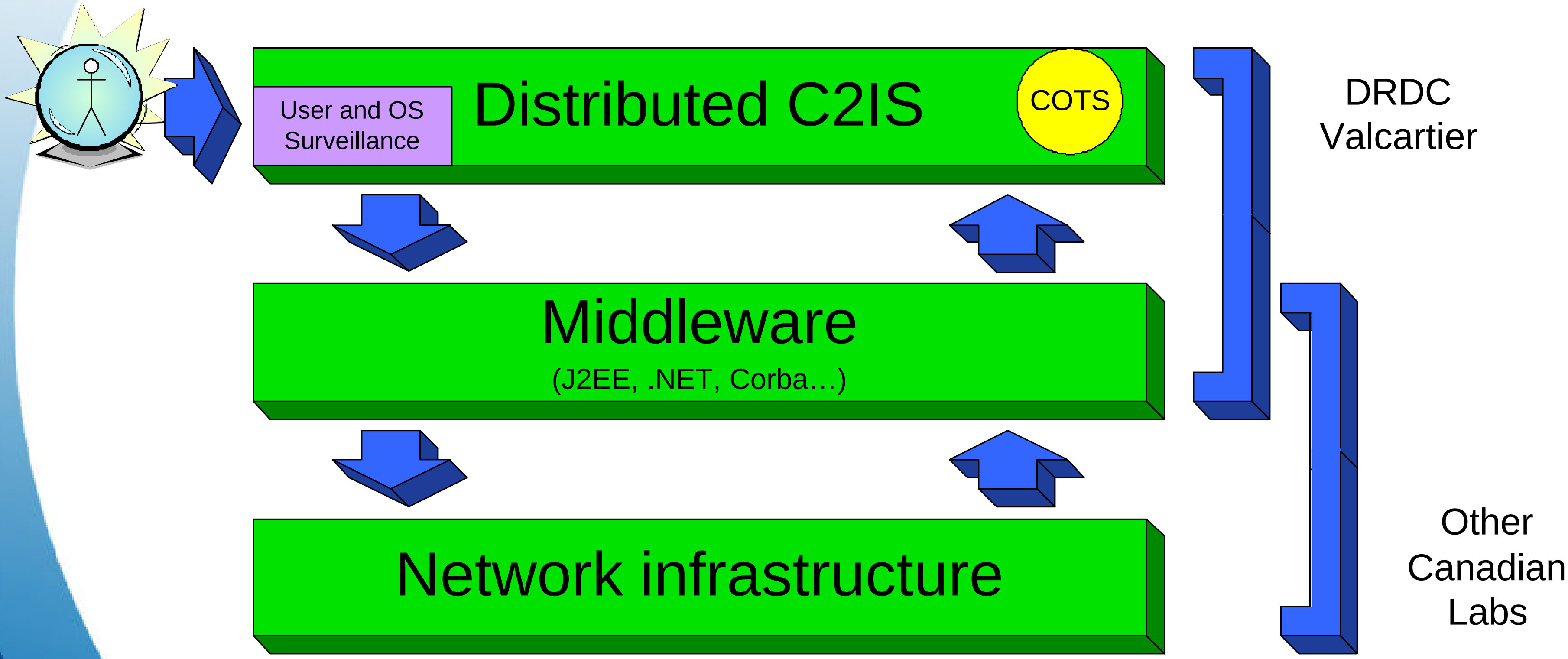
Plan

- DRDC Valcartier
- The MaliCOTS Project
  - Software Certification Techniques
    - Static Analysis of Code
    - Dynamic Monitoring
    - Certified Compilation
- The SOCLe Project
  - Formalized UML/OCL
- Other Projects
  - Unified Security Policies
  - Certified ASN.1
  - Software Visualization
  - Java Hybrid Analysis
- Summary and Perspectives

Defence Research & Development Canada Valcartier

- Québec City
  - ~360 employees
  - Next to a military base (Land Forces)
- 3 main areas
  - Optronic Systems
  - Combat Systems
  - Information Systems
    - System of Systems
    - Decision Support Systems
    - Information and Knowledge Management
      - Trusted Command and Control

Trusted Command and Control



Distributed C2IS

User and OS Surveillance

COTS

Middleware

(J2EE, .NET, Corba…)

Network infrastructure

DRDC Valcartier

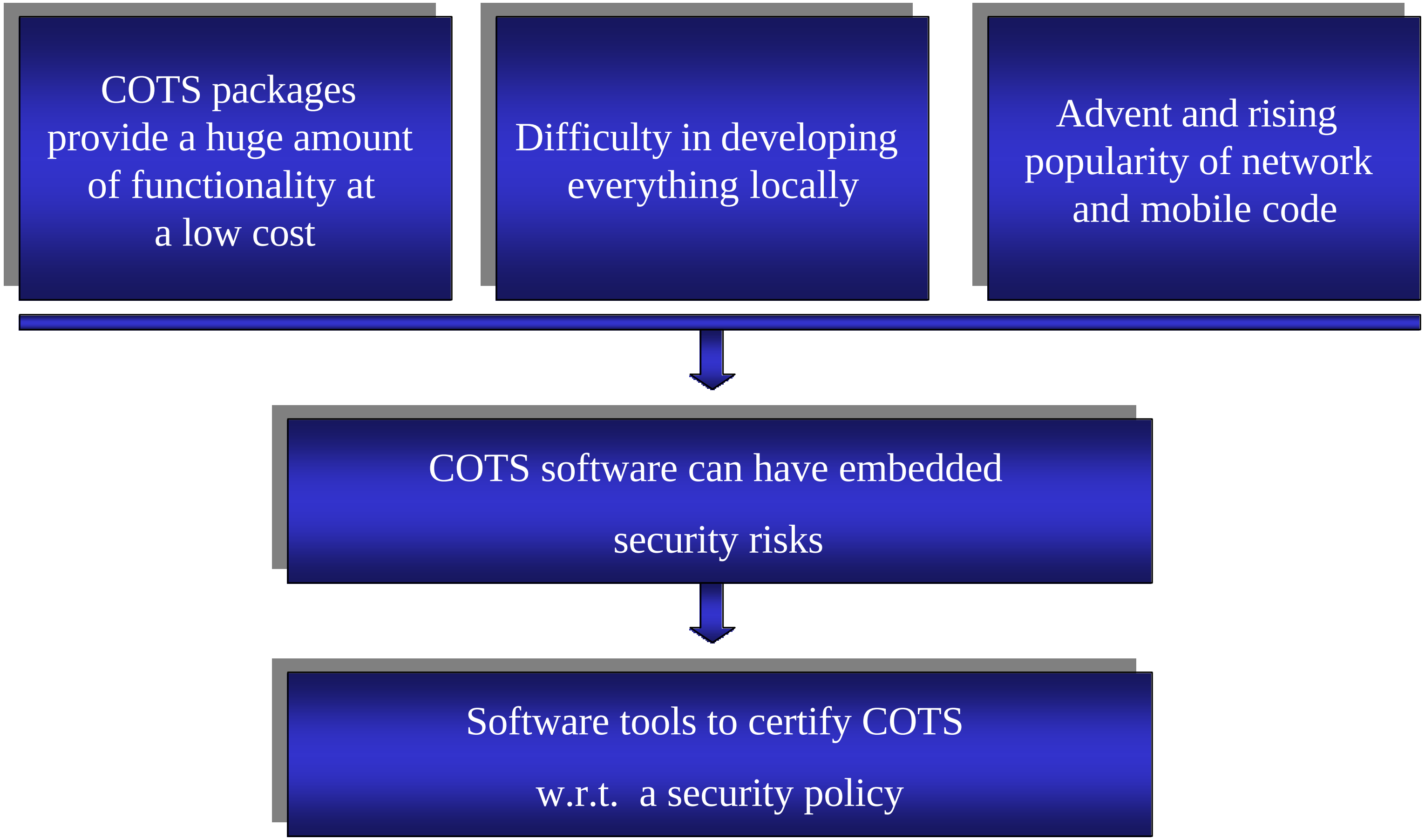
Other Canadian Labs

Projet

MaliCOTS

Project

Motivation



COTS packages provide a huge amount of functionality at a low cost

Difficulty in developing everything locally

Advent and rising popularity of network and mobile code

COTS software can have embedded security risks

Software tools to certify COTS w.r.t. a security policy

Prominent Concerns with COTS Software

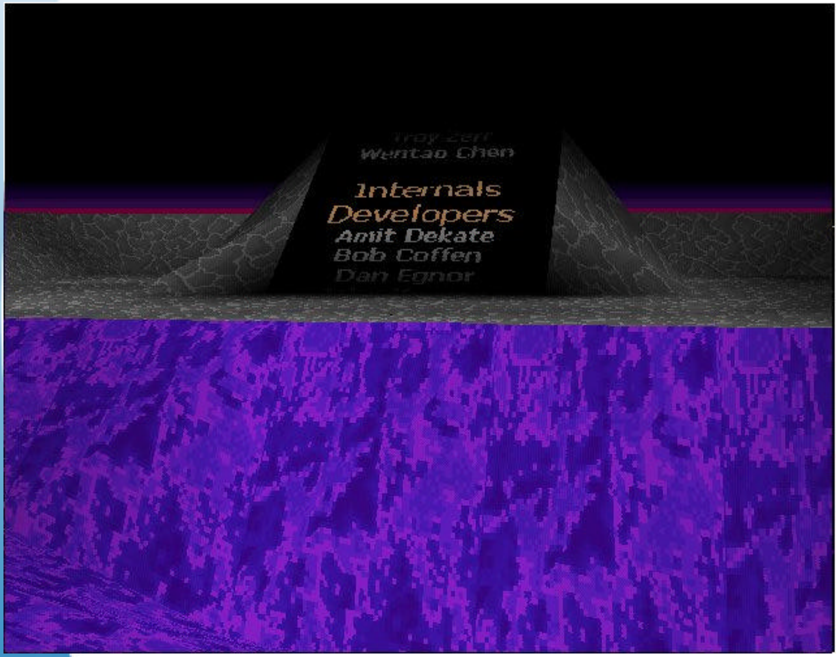
- Trap-doors
  - ex: Lotus Notes, Windows NT & ‘98 for MS administrators & NSA
- Software expiration logic
  - ex: COTS software encrypts the hard disk after expiration of the license
- Hidden communications
  - ex: CD player sends “your listening preferences” to a distributor periodically
- Management of temporary files during and after execution
- Undesired functionalities
  - ex: EXCEL’97 Flight Simulator, Word’97 Pinball Machine

Excel '97

A Flight Simulator ?

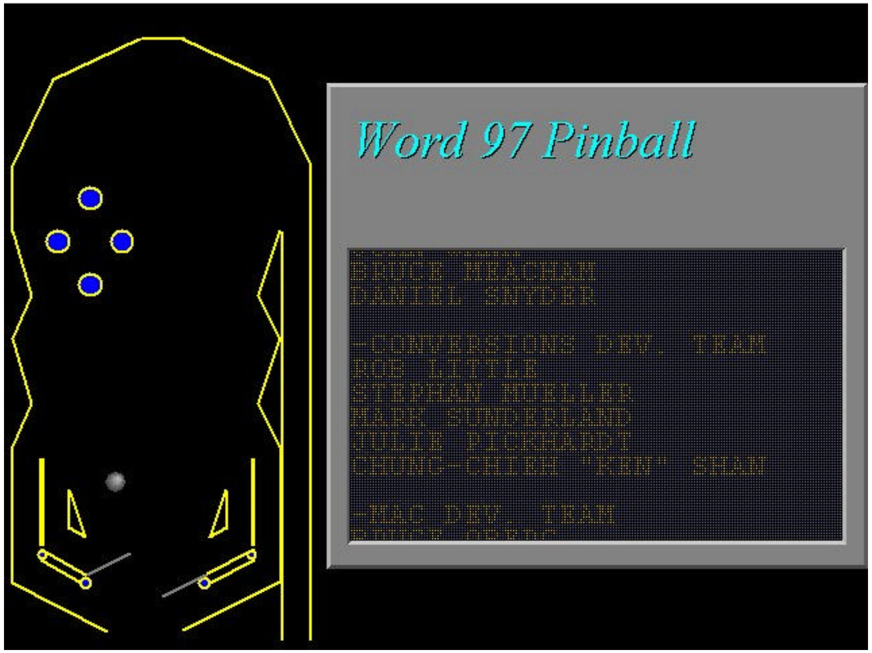
Word '97

A Pinball Machine ?



Wentao Chen

Internals
Developers
Amit Dekate
Bob Coffen
Dan Egnor



Word 97 Pinball

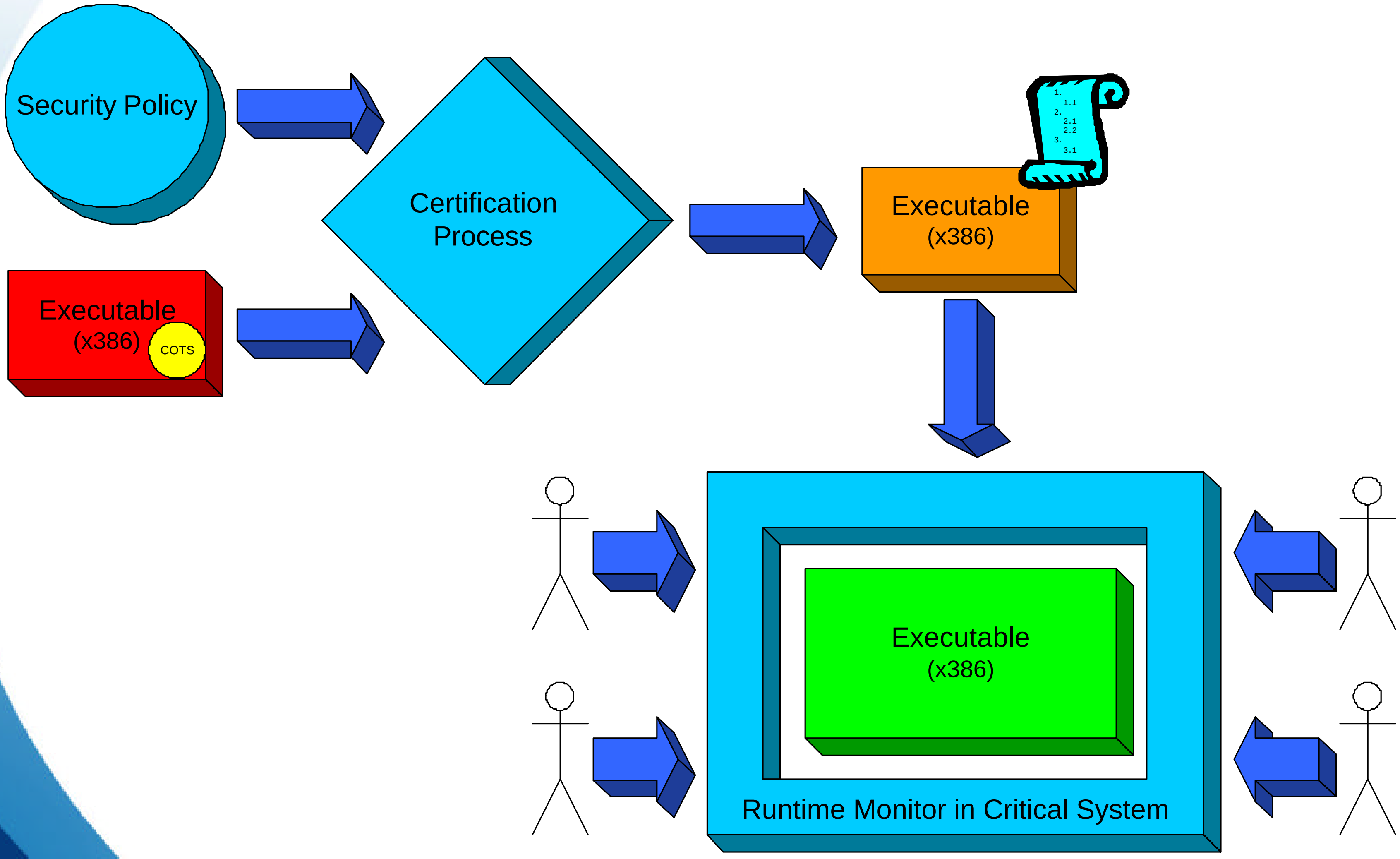
BRUCE MEACHAM
DANIEL SNYDER

-CONVERSIONS DEV. TEAM
ROB LITTLE
STEPHAN MUELLER
MARK SUNDERLAND
JULIE PICKHARDT
CHONG-CHIEH "KEN" SHAN

-MAC DEV. TEAM

Ref: http://www.eeggs.com/tree/1-1.html

Main Idea:
Safe and Secure Distributed C2IS



MaliCOTS Project: Malicious Commercial-Off-The-Shelf

- Objective:
  - *Detect and prevent malicious code in critical systems*

- 4 sub-projects in parallel (1997-2001)
- Highly motivated and competent team
  - Partnership DRDC / Université Laval
  - *12 graduate students + 4 Professors + 2 DRDC Scientists*
- Financing
  - *Technological Investment Fund (TIF) (National Defence)*
  - *NSERC (Industry Canada)*
  - *FCAR (Research, Science and Technology – Québec)*

Software Certification Techniques

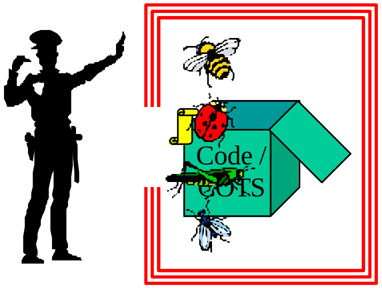
- Static Analysis of Code
- Dynamic Monitoring
- Certifying Compiler Technology (C and Java)

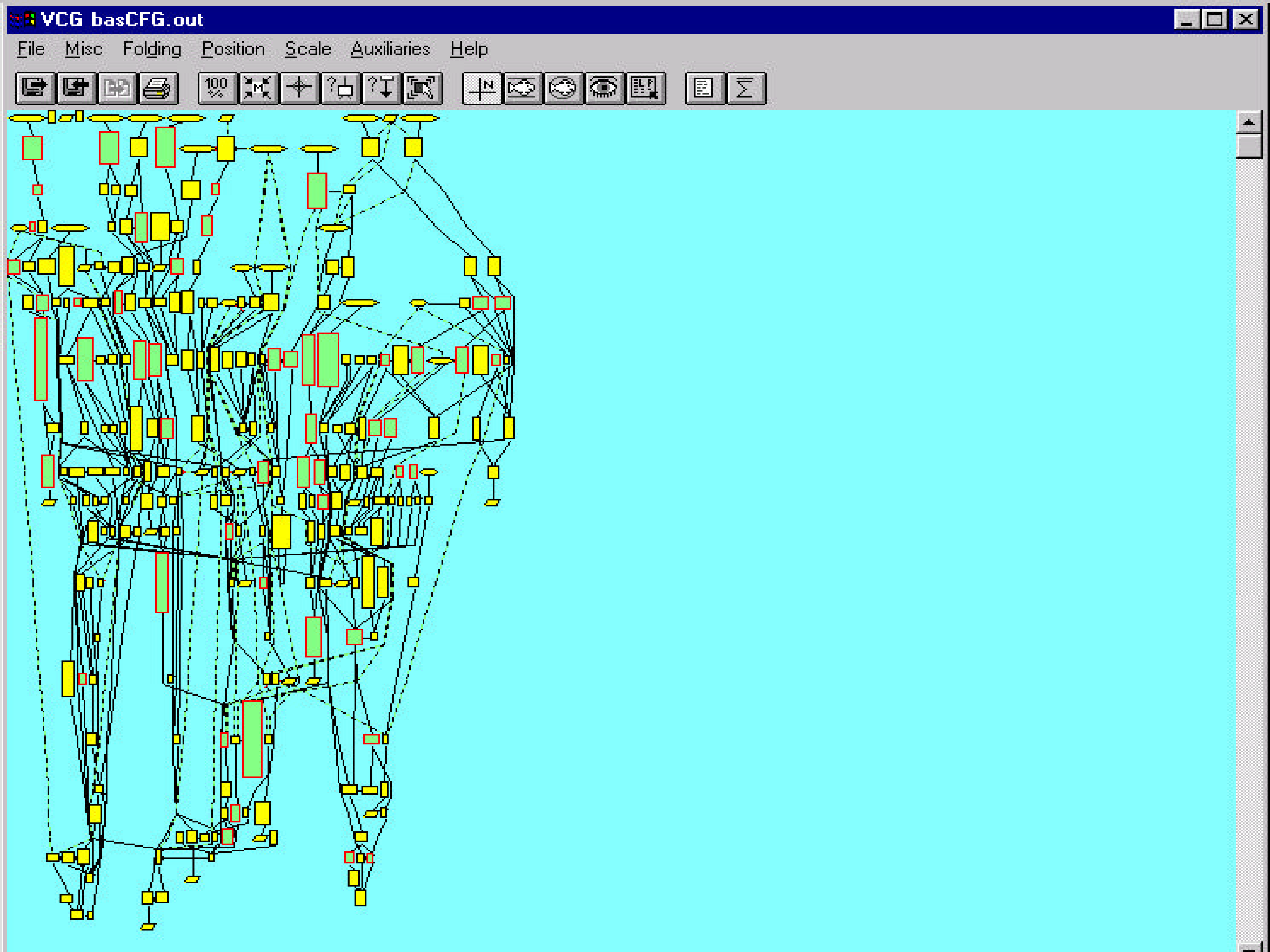
Software Certification Techniques

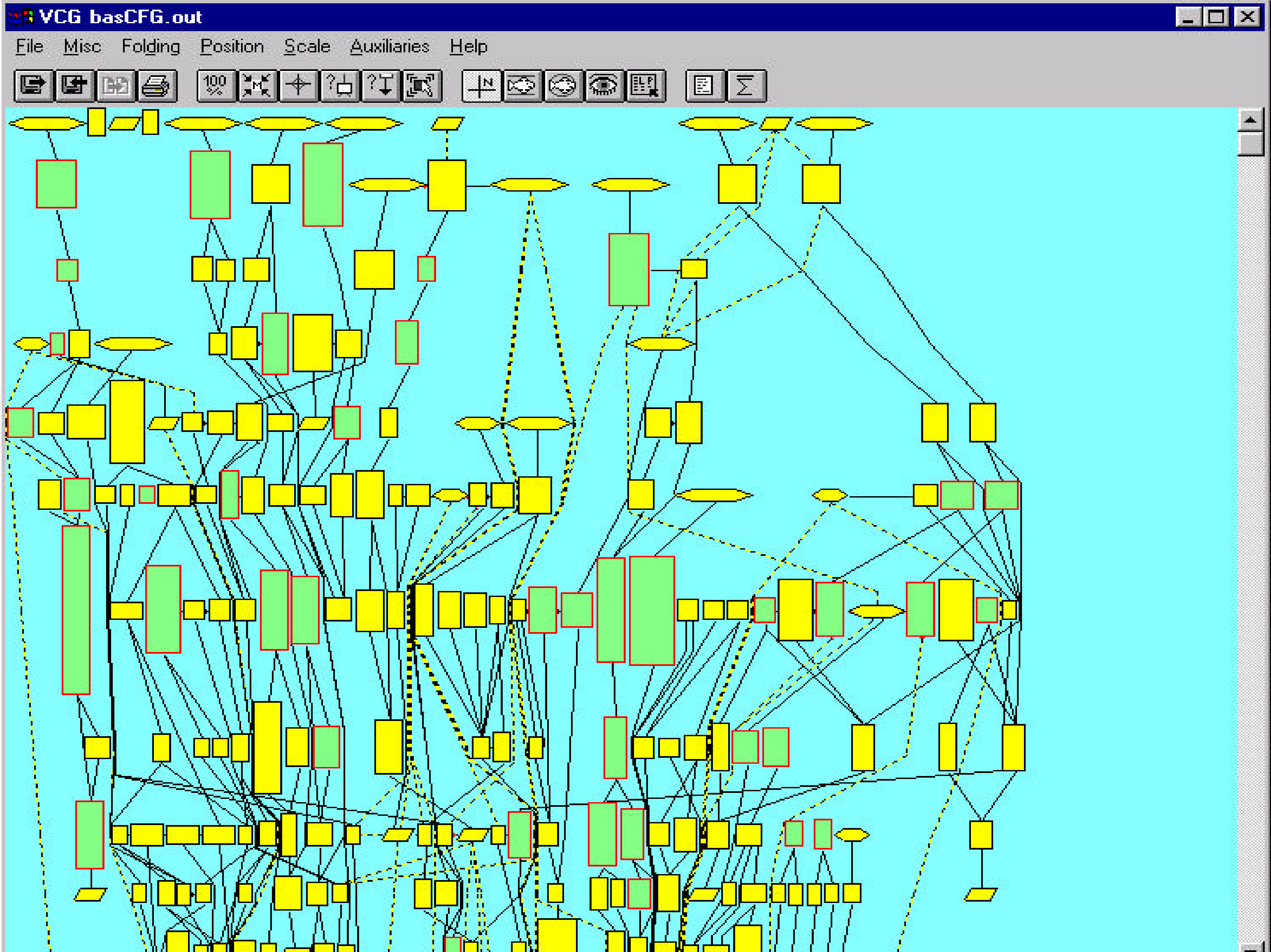
- Static Analysis of Code
- Dynamic Monitoring
- Certifying Compiler Technology (C and Java)

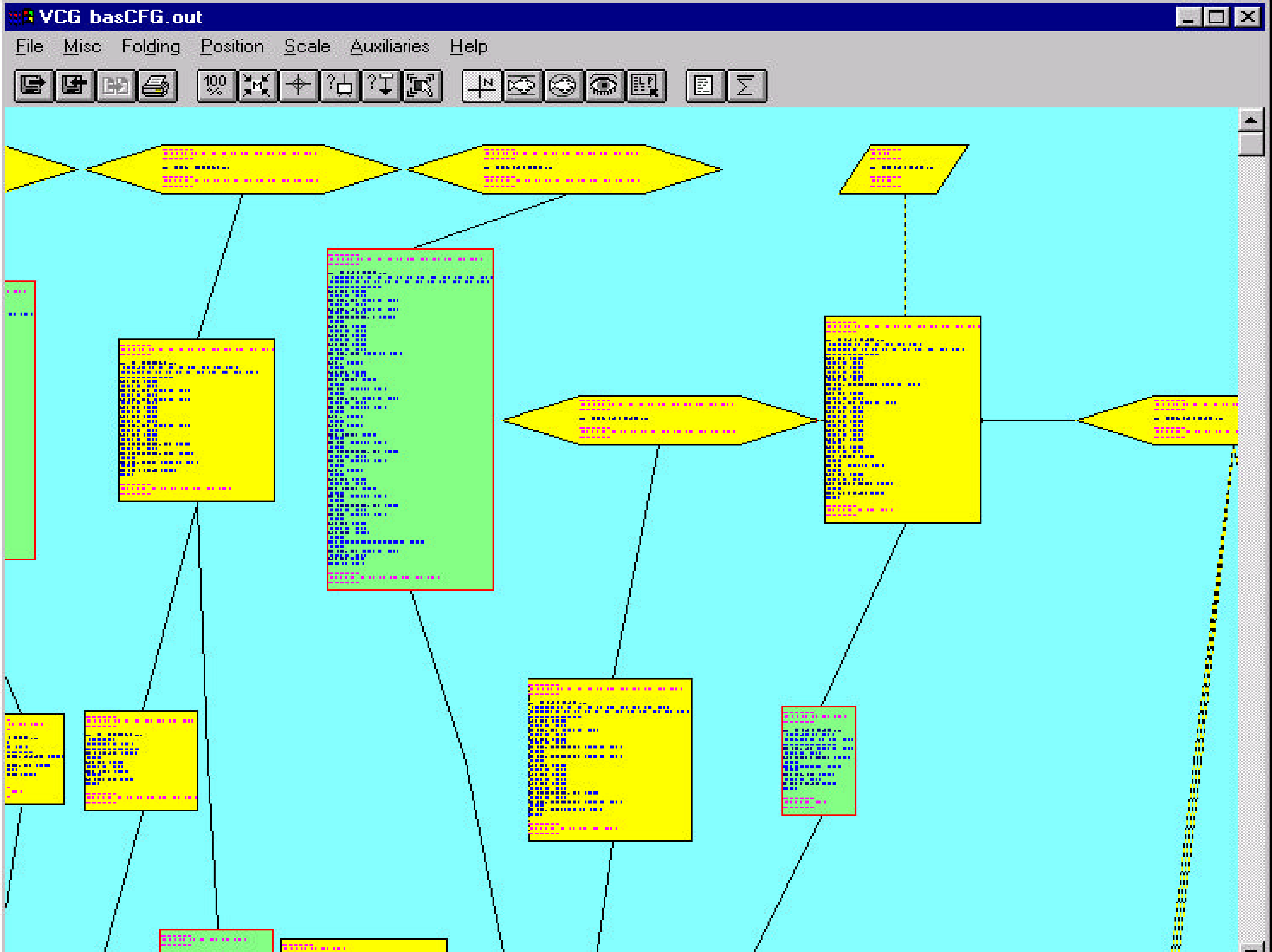
Static Analysis of Code

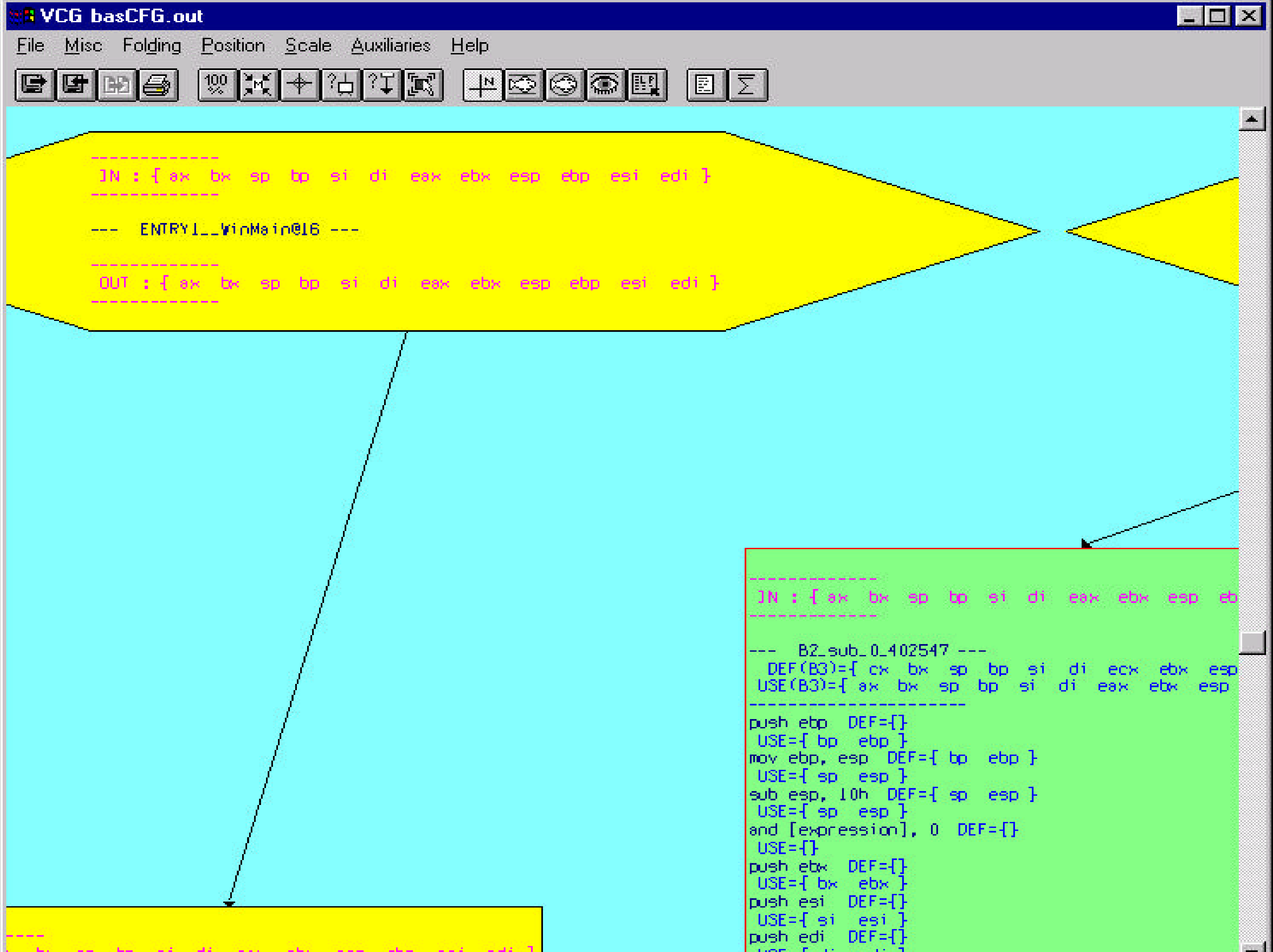
- Program is not running
- Based on
  - Control flow analysis
  - Typed-based analysis




VCG basCFG.out
File
Misc
Folding
Position
Scale
Auxiliaries
Help
100 %


VCG basCFG.out
File
Misc
Folding
Position
Scale
Auxiliaries
Help
100 %

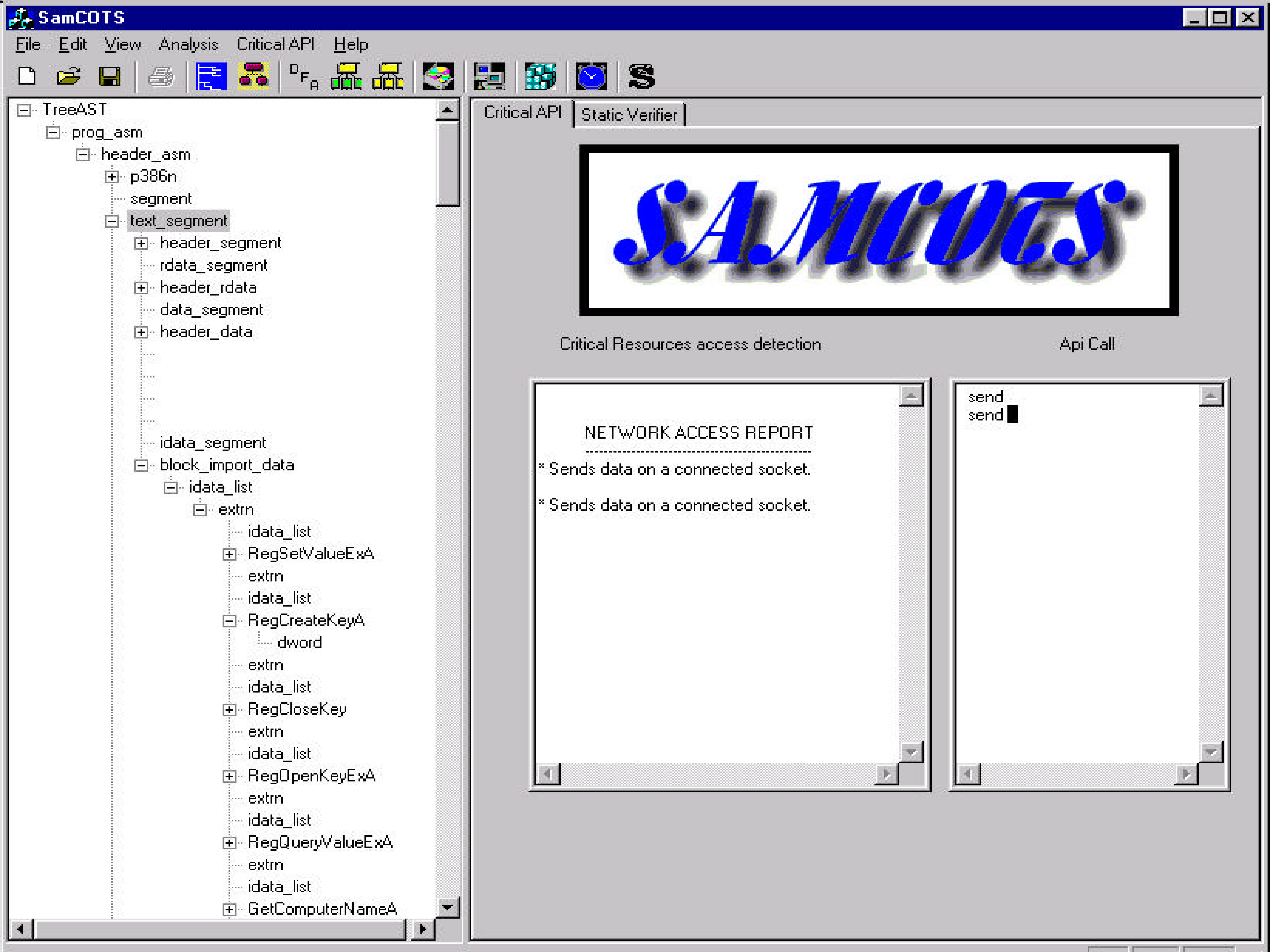

VCG basCFG.out
File
Misc
Folding
Position
Scale
Auxiliaries
Help
100 %



```
-------------
 IN : { ax  bx  sp  bp  si  di  eax  ebx  esp  ebp  esi  edi }
-------------

---  ENTRY1__WinMain@16 ---

-------------
 OUT : { ax  bx  sp  bp  si  di  eax  ebx  esp  ebp  esi  edi }
-------------
```

```
-------------
 IN : { ax  bx  sp  bp  si  di  eax  ebx  esp  eb

-------------

---  B2_sub_0_402547 ---
  DEF(B3)={ cx  bx  sp  bp  si  di  ecx  ebx  esp
 USE(B3)={ ax  bx  sp  bp  si  di  eax  ebx  esp
----------------------
push ebp  DEF={}
 USE={ bp  ebp }
mov ebp, esp  DEF={ bp  ebp }
 USE={ sp  esp }
sub esp, 10h  DEF={ sp  esp }
 USE={ sp  esp }
and [expression], 0  DEF={}
 USE={}
push ebx  DEF={}
 USE={ bx  ebx }
push esi  DEF={}
 USE={ si  esi }
push edi  DEF={}
```

```
----
```



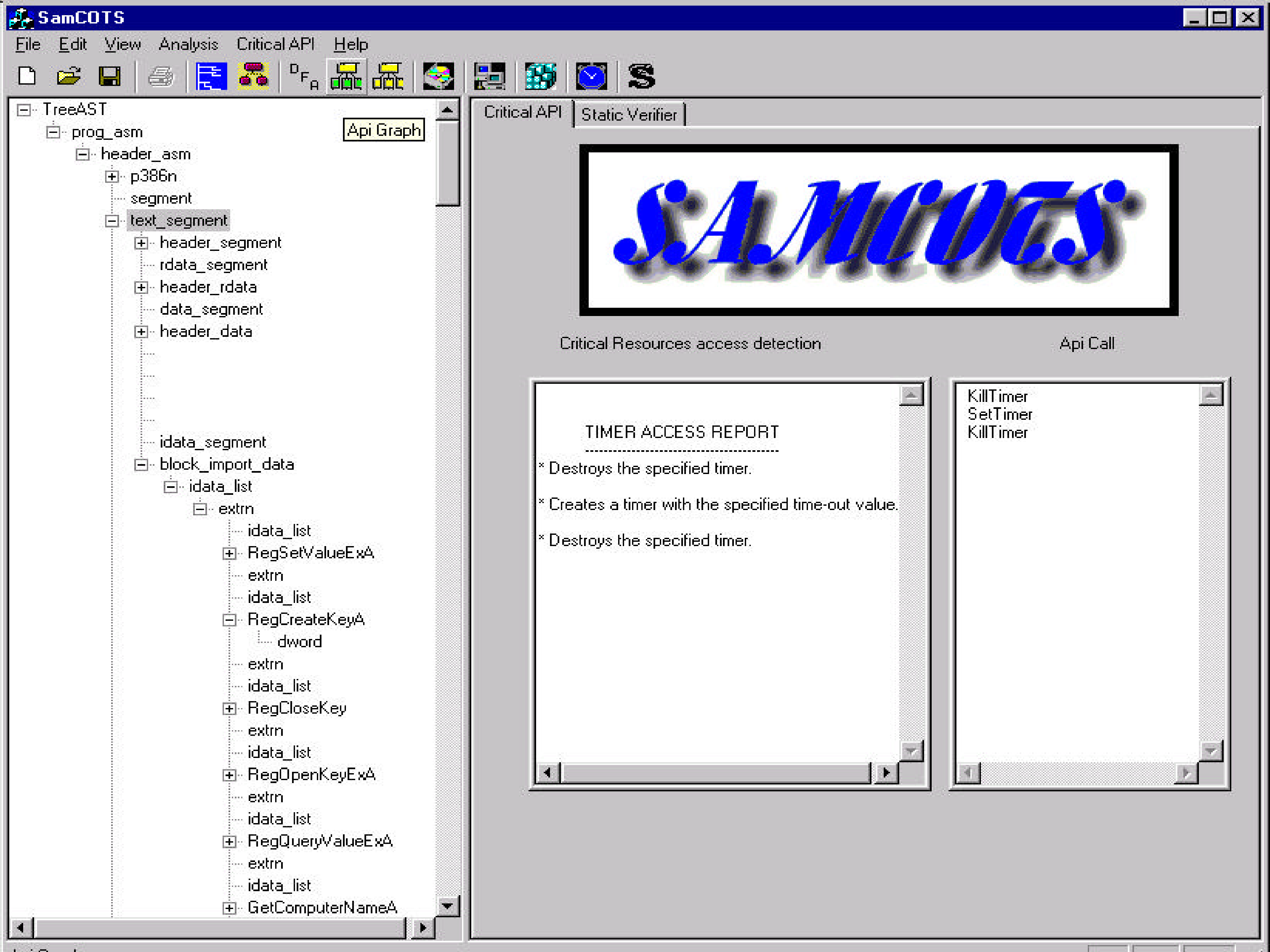
```
TreeAST
  prog_asm
    header_asm
      p386n
      segment
      text_segment
        header_segment
        rdata_segment
        header_rdata
        data_segment
        header_data
        idata_segment
        block_import_data
          idata_list
            extrn
              idata_list
              RegSetValueExA
              extrn
              idata_list
              RegCreateKeyA
                dword
              extrn
              idata_list
              RegCloseKey
              extrn
              idata_list
              RegOpenKeyExA
              extrn
              idata_list
              RegQueryValueExA
              extrn
              idata_list
              GetComputerNameA
```

Critical Resources access detection

```
NETWORK ACCESS REPORT
------------------------------------------
* Sends data on a connected socket.

* Sends data on a connected socket.
```

Api Call

```
send
send
```



```
TreeAST
  prog_asm
    header_asm
      p386n
      segment
      text_segment
        header_segment
        rdata_segment
        header_rdata
        data_segment
        header_data




        idata_segment
        block_import_data
          idata_list
            extrn
              idata_list
              RegSetValueExA
              extrn
              idata_list
              RegCreateKeyA
                dword
              extrn
              idata_list
              RegCloseKey
              extrn
              idata_list
              RegOpenKeyExA
              extrn
              idata_list
              RegQueryValueExA
              extrn
              idata_list
              GetComputerNameA
```

TIMER ACCESS REPORT

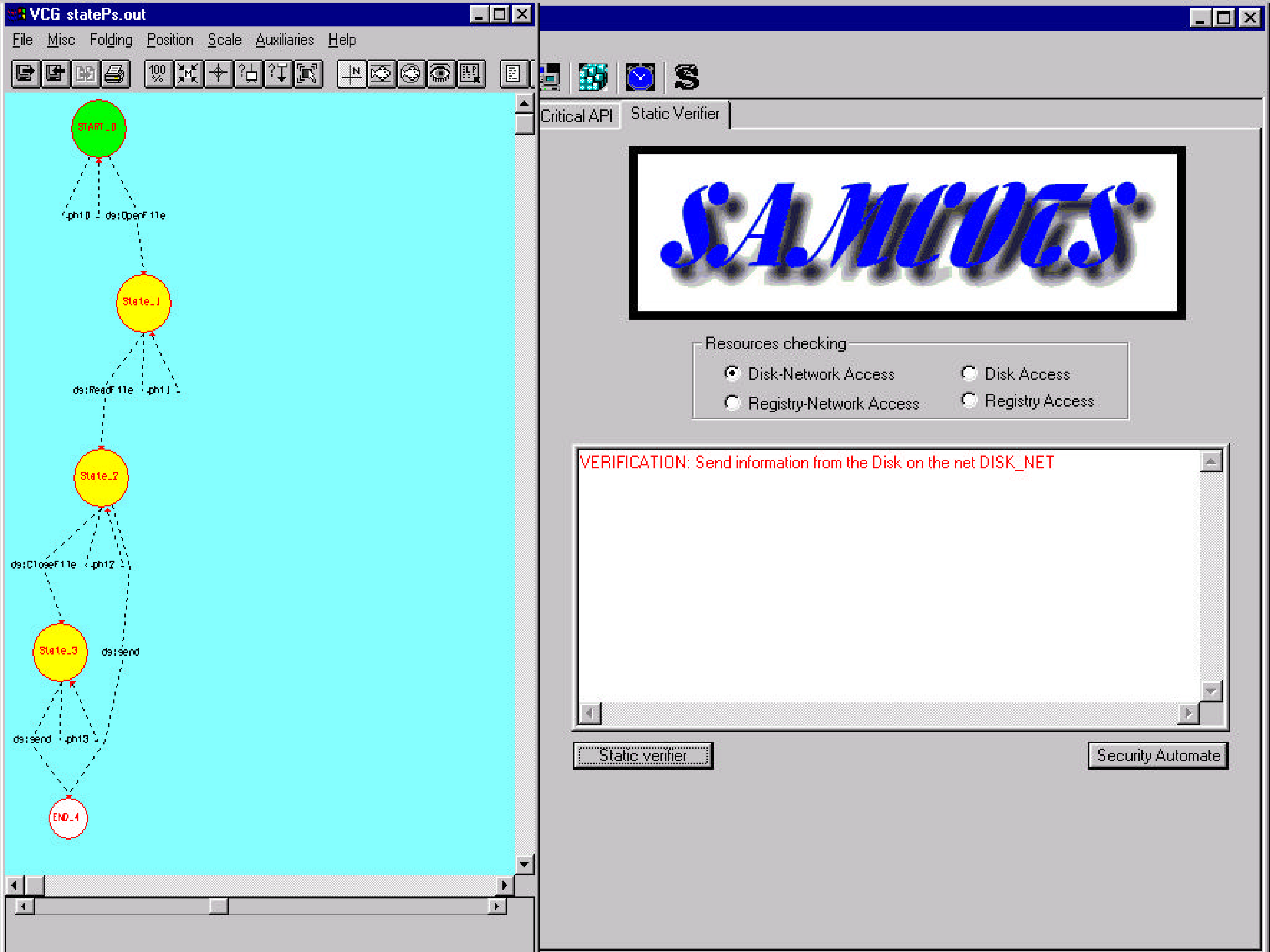
---

* Destroys the specified timer.

* Creates a timer with the specified time-out value.

* Destroys the specified timer.

KillTimer
SetTimer
KillTimer



Resources checking

● Disk-Network Access ○ Disk Access

○ Registry-Network Access ○ Registry Access

```
VERIFICATION: Send information from the Disk on the net DISK_NET
```

Static Analysis

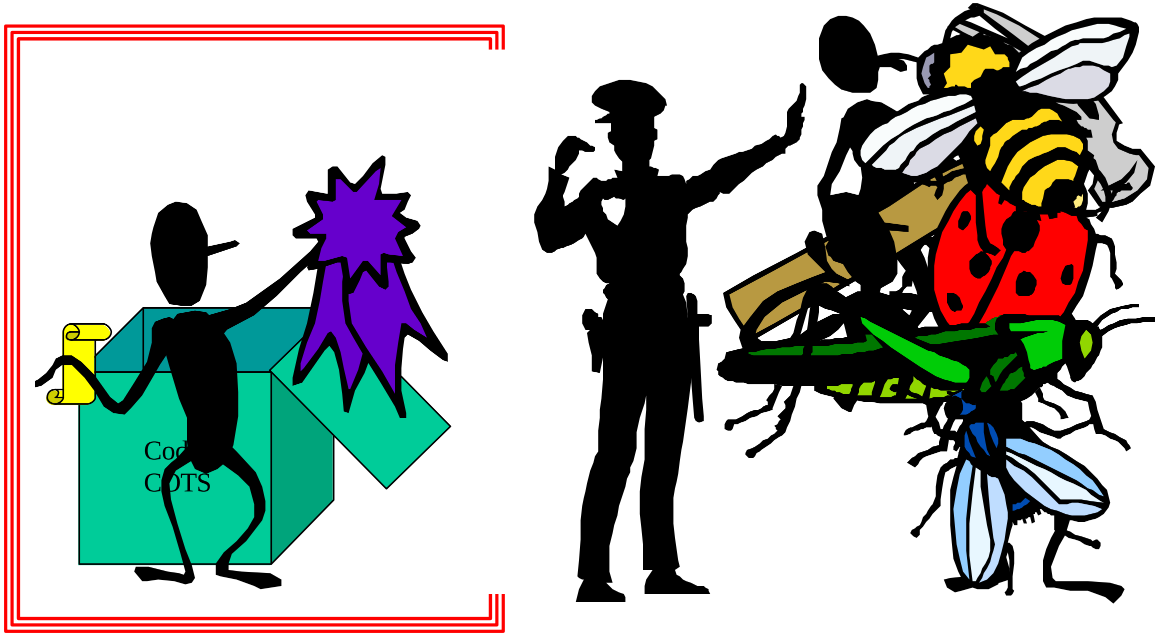
- PROS:
  - No need to run the program
  - Analysis of program behaviour over all possible execution paths
  - Analyze once, execute everywhere
- CONS:
  - Undecidability of many interesting properties
  - Hard on binary executables
  - Illegal on most COTS software

Software Certification Techniques

- Static Analysis of Code
- Dynamic Monitoring
- Certifying Compiler Technology (C and Java)

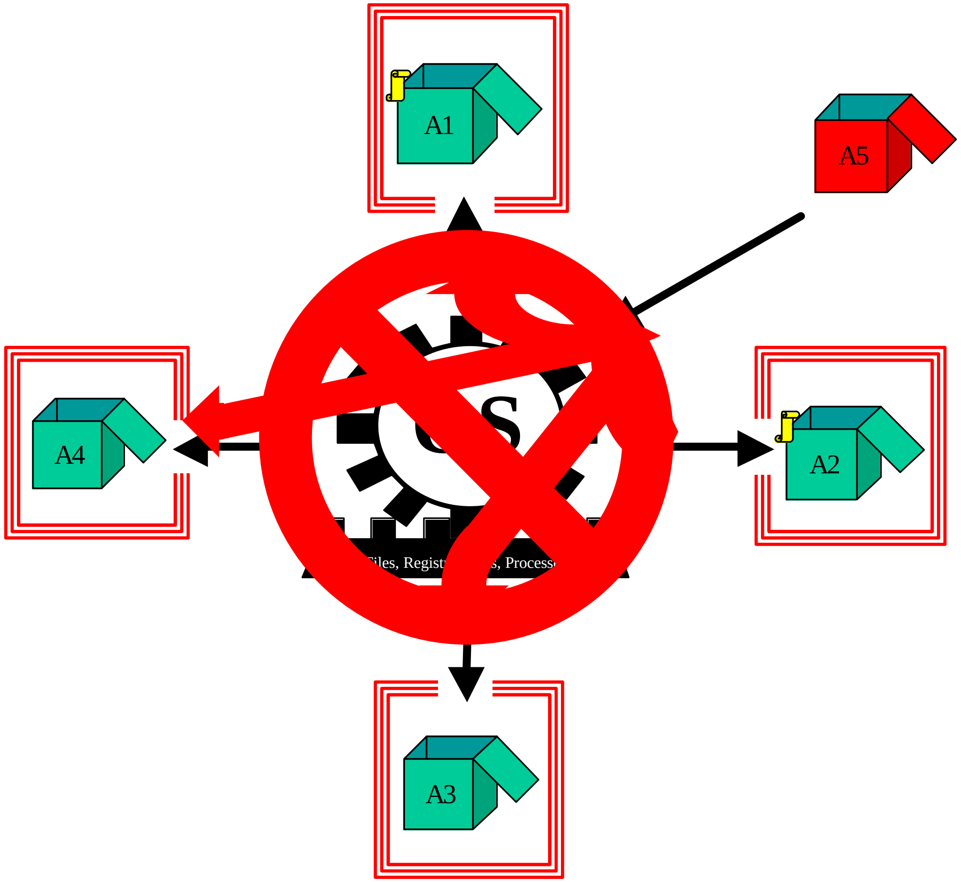
Monitoring

Surveillance of software at runtime (wrapping)



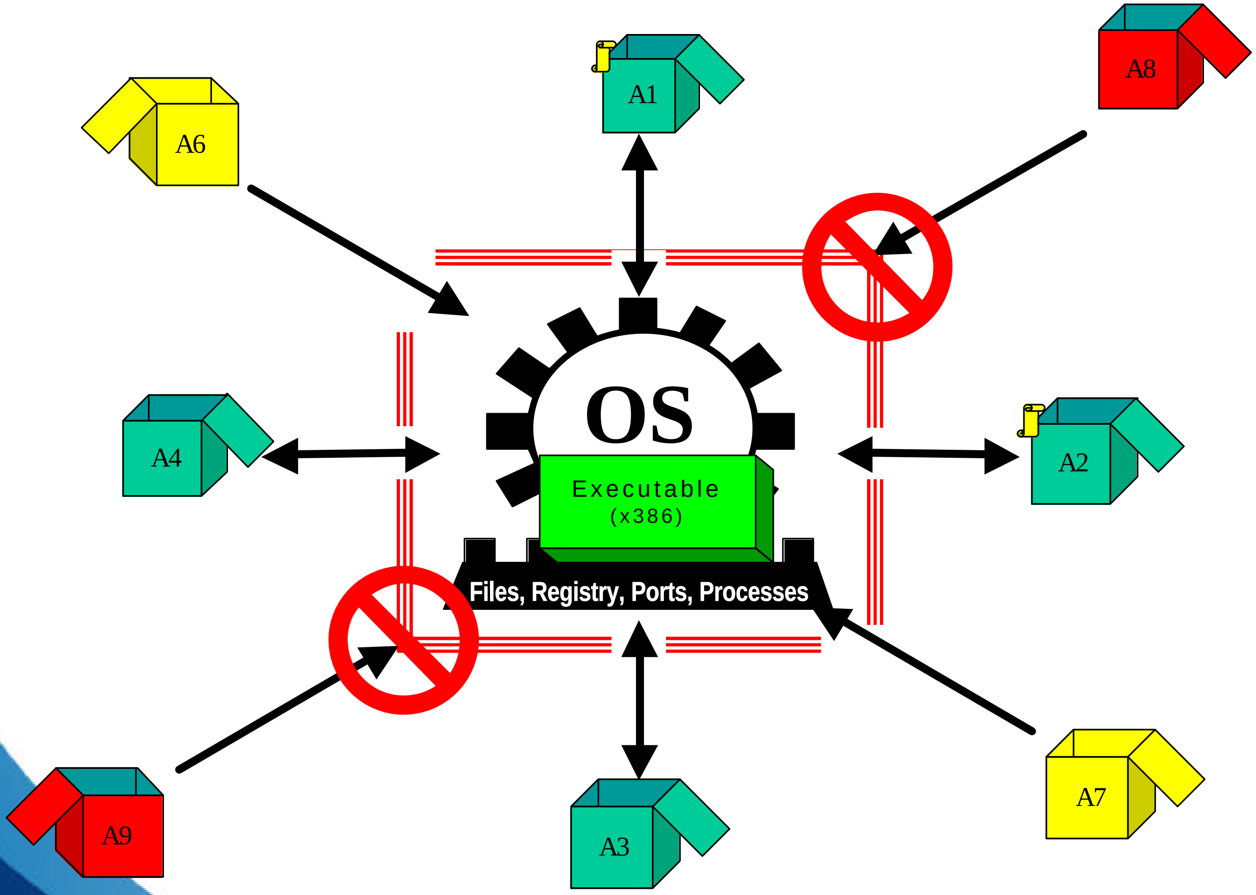
Traditional Wrapping

- Typically, each critical application is wrapped
- This does not work because the OS and other applications are left defenceless against malicious applications



Wrapping the OS

- Wrapping critical resources:
  - Critical resources: Files, Registry, Ports, Processes
- To:
  - Let certified and benign applications in
  - Prevent unwanted access

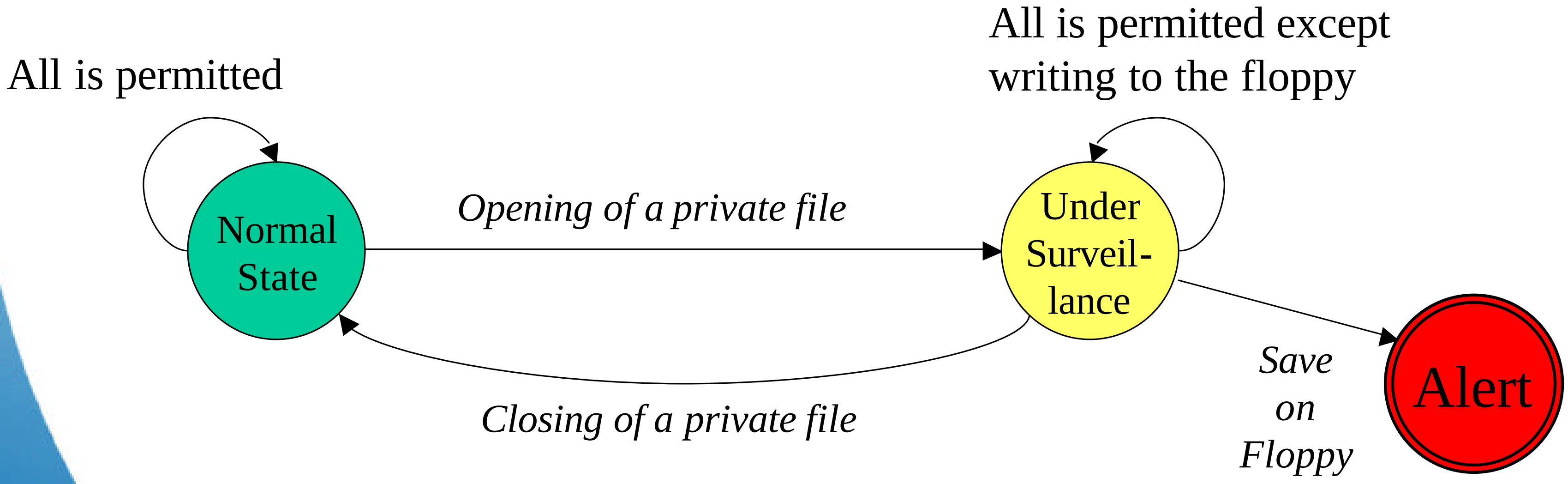


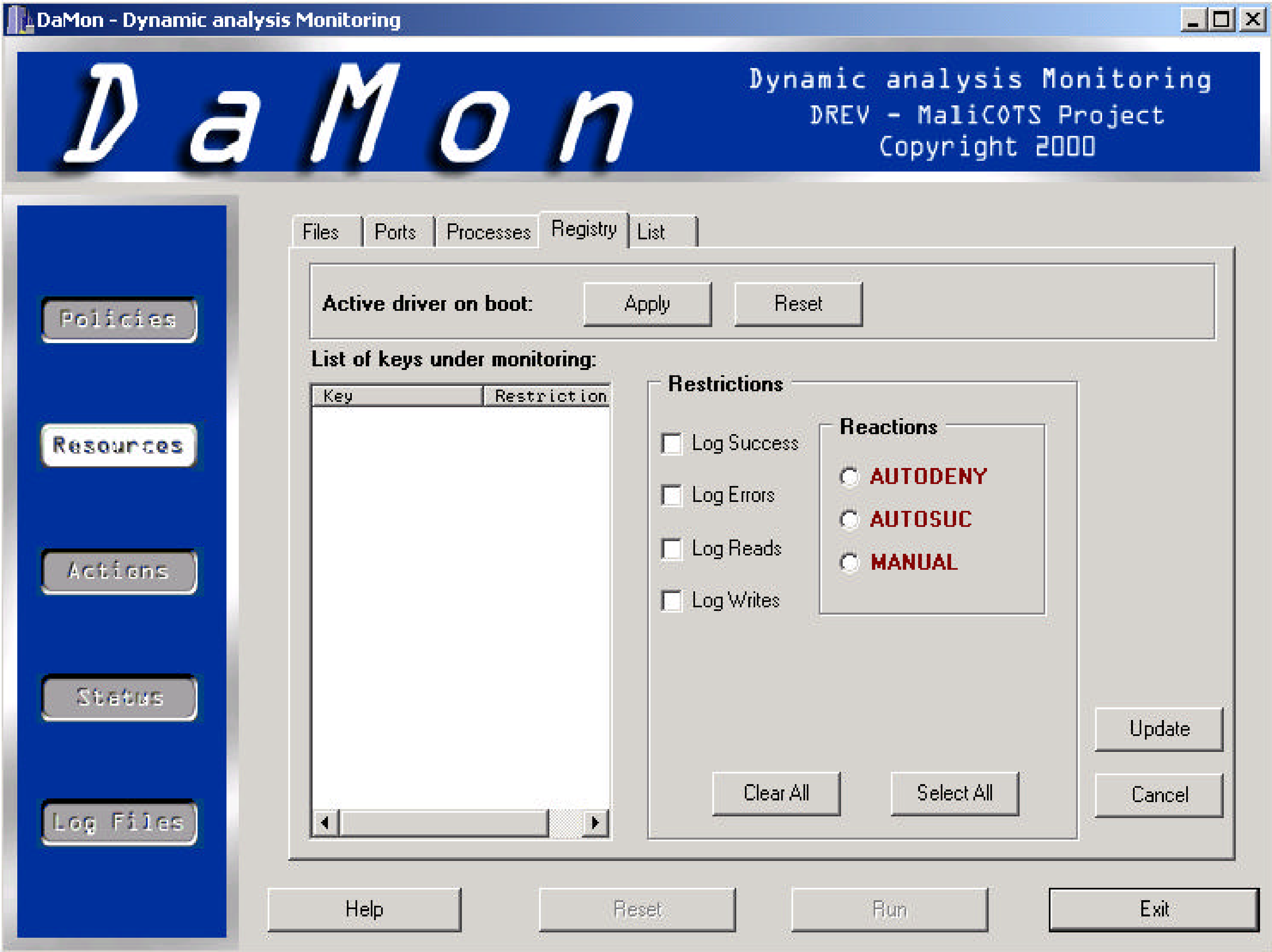
Schneider's Automata

Security Policy

For example

- If a private file is opened, writing to the floppy is forbidden


All is permitted
All is permitted except writing to the floppy
Normal State
Opening of a private file
Under Surveil-lance
Closing of a private file
Save on Floppy
Alert



Active driver on boot:

List of keys under monitoring:

Key	Restriction

Restrictions

□ Log Success

□ Log Errors

□ Log Reads

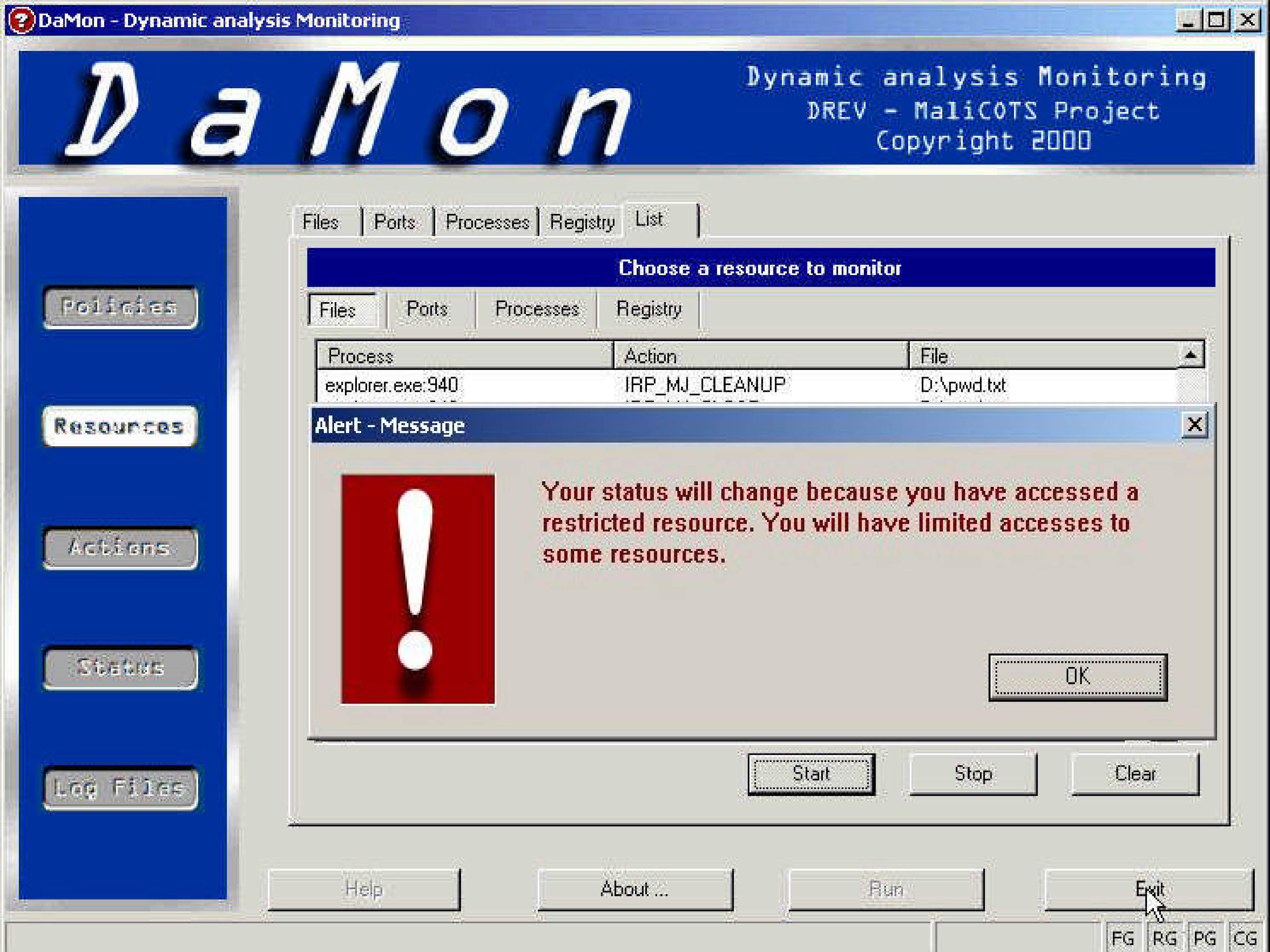
□ Log Writes

Reactions

○ AUTODENY

○ AUTOSUC

○ MANUAL



DaMon - Dynamic analysis Monitoring

DaMon

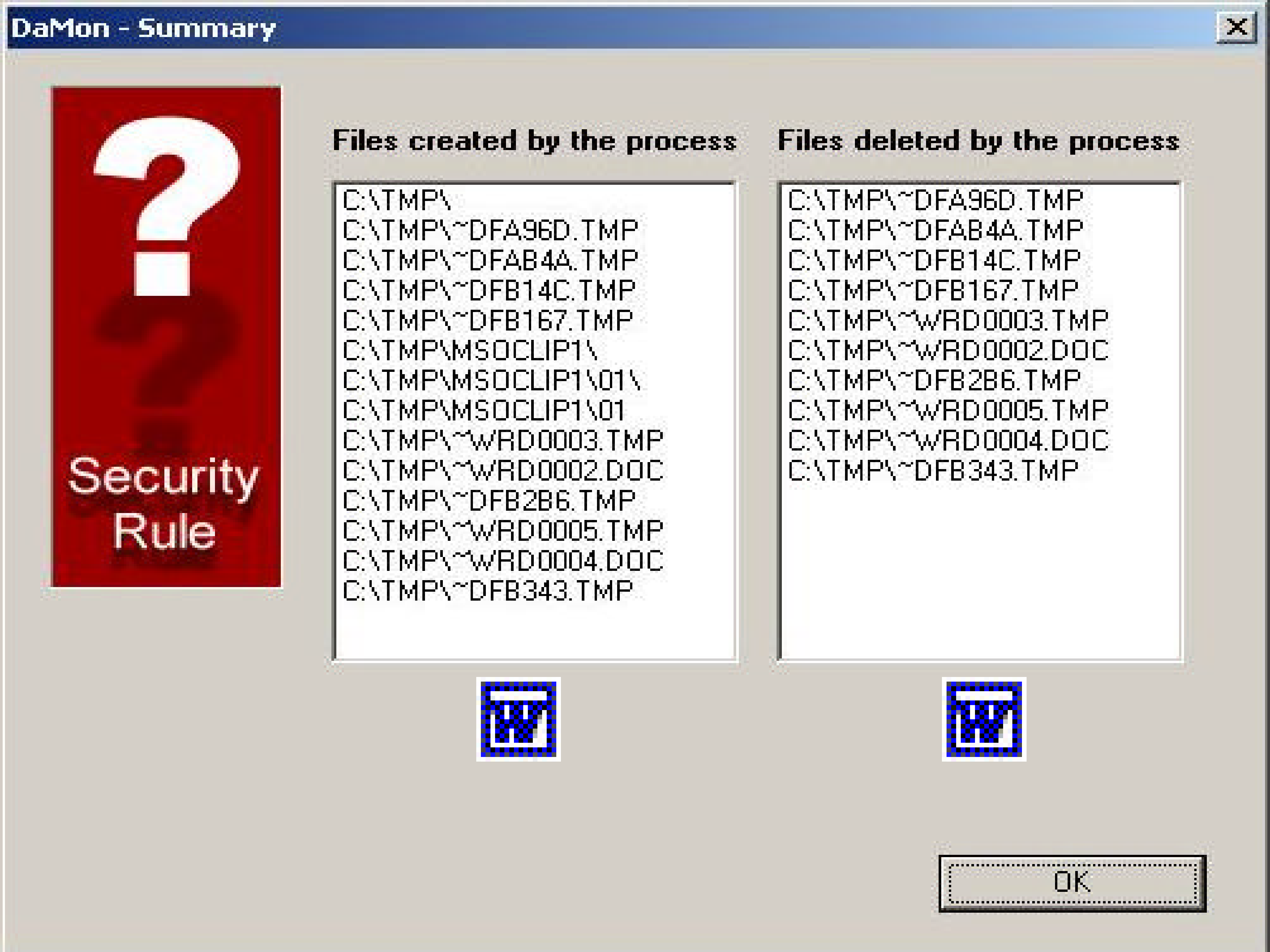
Dynamic analysis Monitoring
DREV - MaliCOTS Project
Copyright 2000

Choose a resource to monitor

Process	Action	File
explorer.exe:940	IRP_MJ_CLEANUP	D:\pwd.txt

Alert - Message

Your status will change because you have accessed a restricted resource. You will have limited accesses to some resources.



Files created by the process

```
C:\TMP\
C:\TMP\~DFA96D.TMP
C:\TMP\~DFAB4A.TMP
C:\TMP\~DFB14C.TMP
C:\TMP\~DFB167.TMP
C:\TMP\MSOCLIP1\
C:\TMP\MSOCLIP1\01\
C:\TMP\MSOCLIP1\01
C:\TMP\~WRD0003.TMP
C:\TMP\~WRD0002.DOC
C:\TMP\~DFB2B6.TMP
C:\TMP\~WRD0005.TMP
C:\TMP\~WRD0004.DOC
C:\TMP\~DFB343.TMP
```

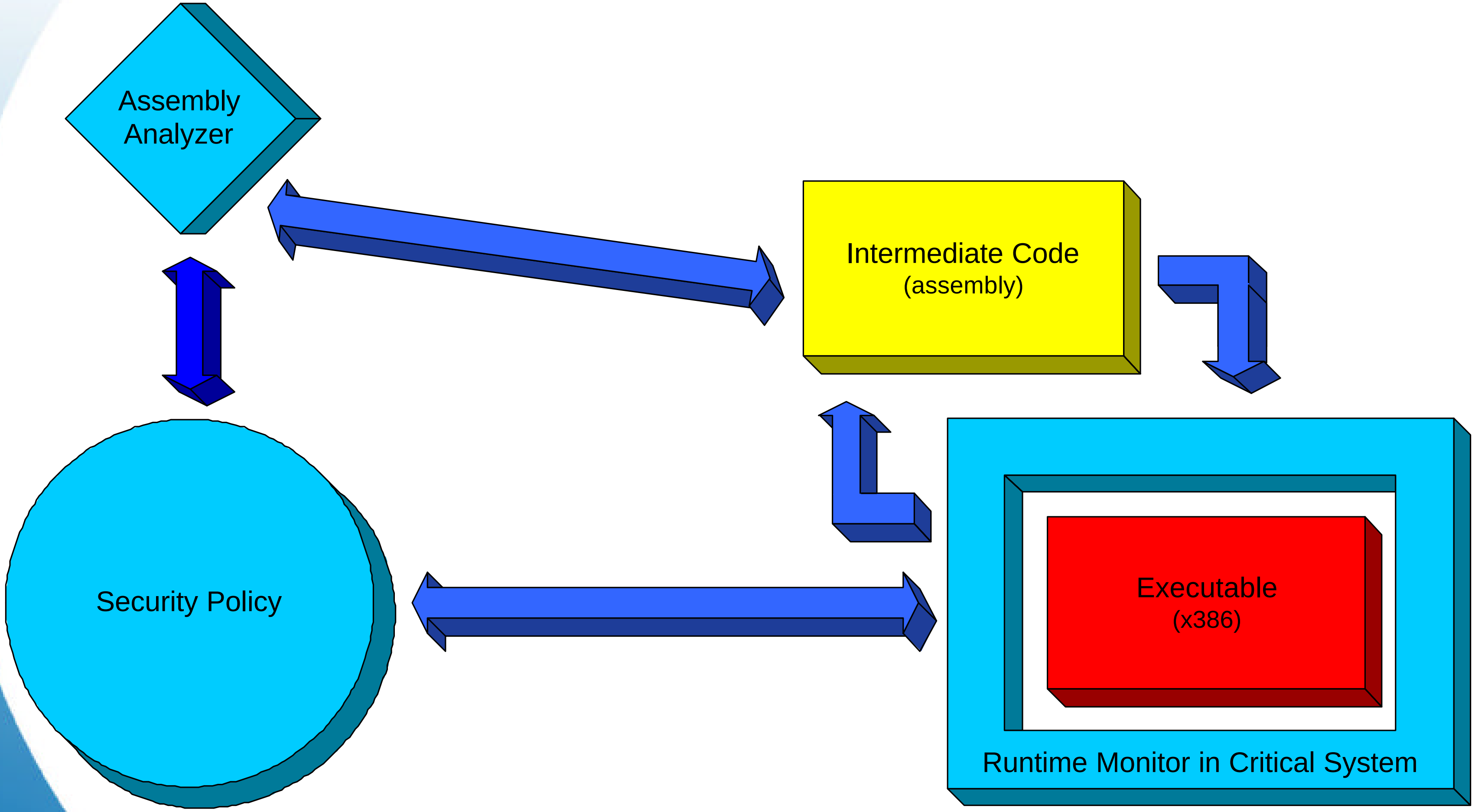
Files deleted by the process

```
C:\TMP\~DFA96D.TMP
C:\TMP\~DFAB4A.TMP
C:\TMP\~DFB14C.TMP
C:\TMP\~DFB167.TMP
C:\TMP\~WRD0003.TMP
C:\TMP\~WRD0002.DOC
C:\TMP\~DFB2B6.TMP
C:\TMP\~WRD0005.TMP
C:\TMP\~WRD0004.DOC
C:\TMP\~DFB343.TMP
```

Monitoring at Runtime

- PROS:
  - Exploits the knowledge that can be gained by running the program
  - Ideal complement to the static analysis of code
  - Acceptable for software vendors
- CONS:
  - Significant overhead in run-time performance
  - “Infinite number” of possibilities and conditions
  - Too much information to manage

MaliCOTS Testbed


Assembly Analyzer
Intermediate Code (assembly)
Security Policy
Executable (x386)
Runtime Monitor in Critical System

Increasing Complexity

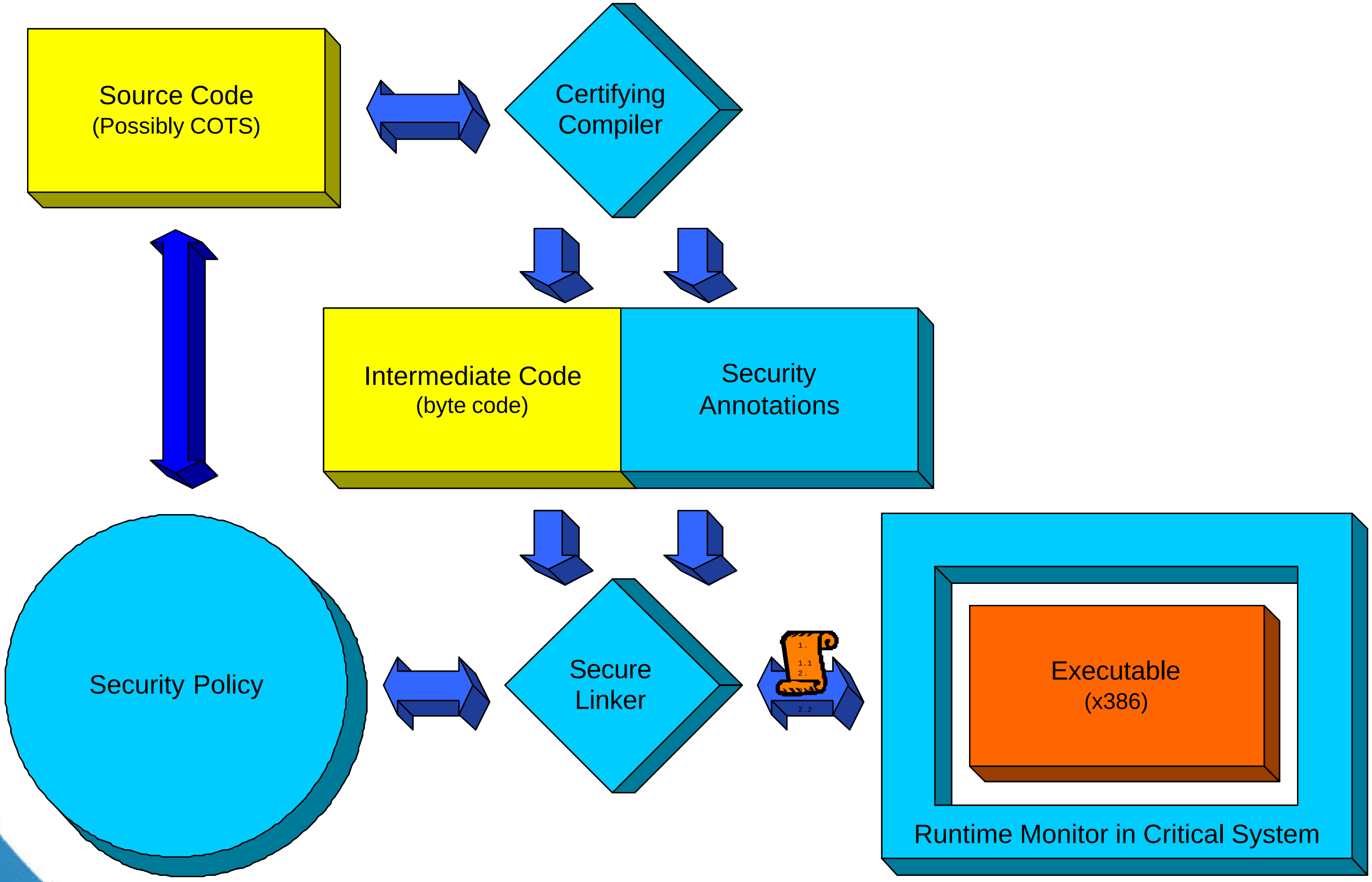
		Lines of code
Win 3.1	1992	*3 million*
Win NT	1992	*4 million*
Win 95	1995	*15 million*
Win NT 4	1996	*16.5 million*
Win 98	1998	*18 million*
Win 2000	2000	*40-60 million*

Ref: M. Sues, M. Gingras, *Secure Programming and Development Practices*, Cinnabar Networks, CITSS Symposium , June 2001

Software Certification Techniques

- Static Analysis of Code
- Dynamic Monitoring
- Certifying Compiler Technology (C and Java)

Certifying Compiler Concept



Source Code
(Possibly COTS)

Certifying
Compiler

Intermediate Code
(byte code)

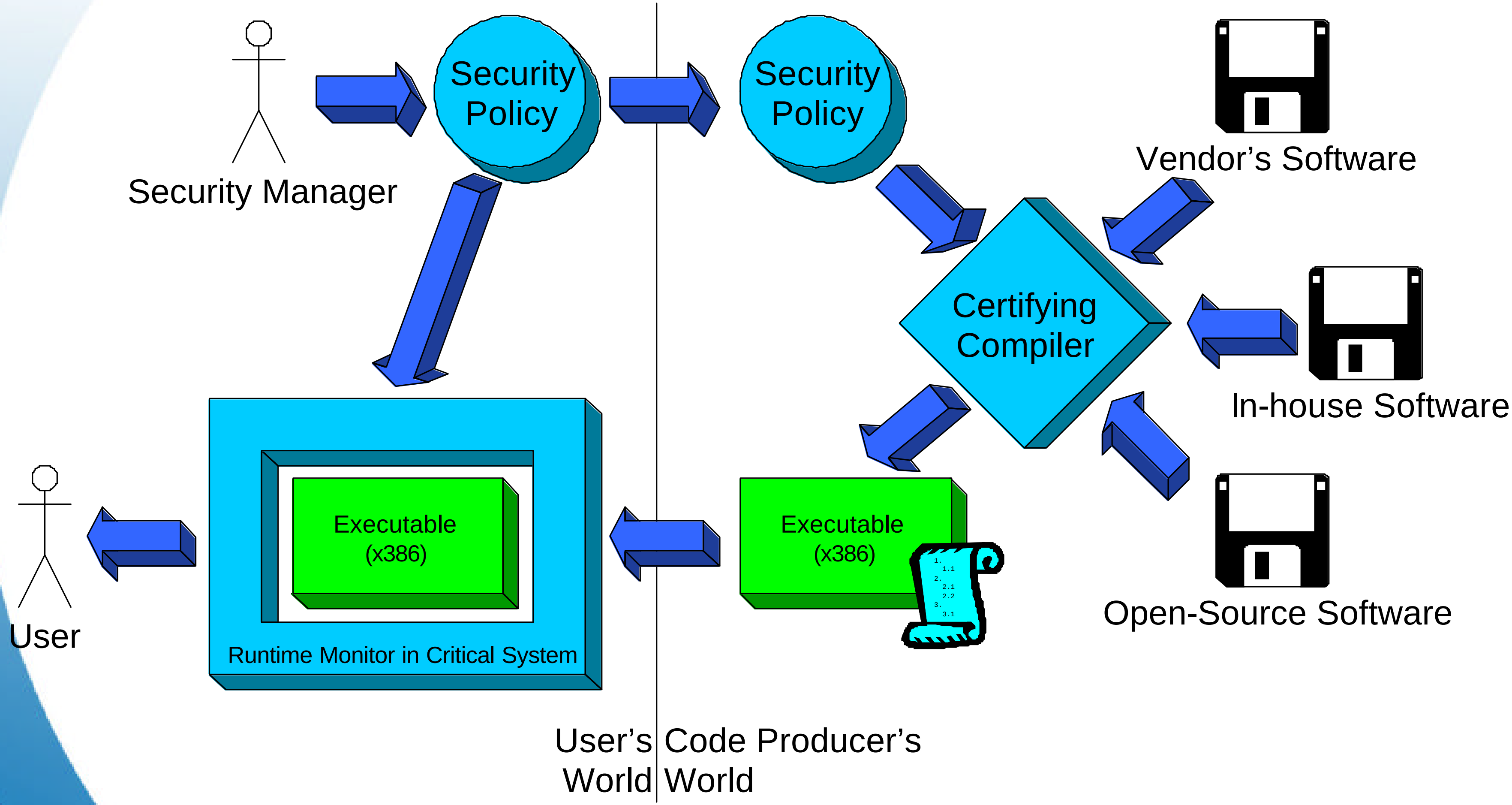
Security
Annotations

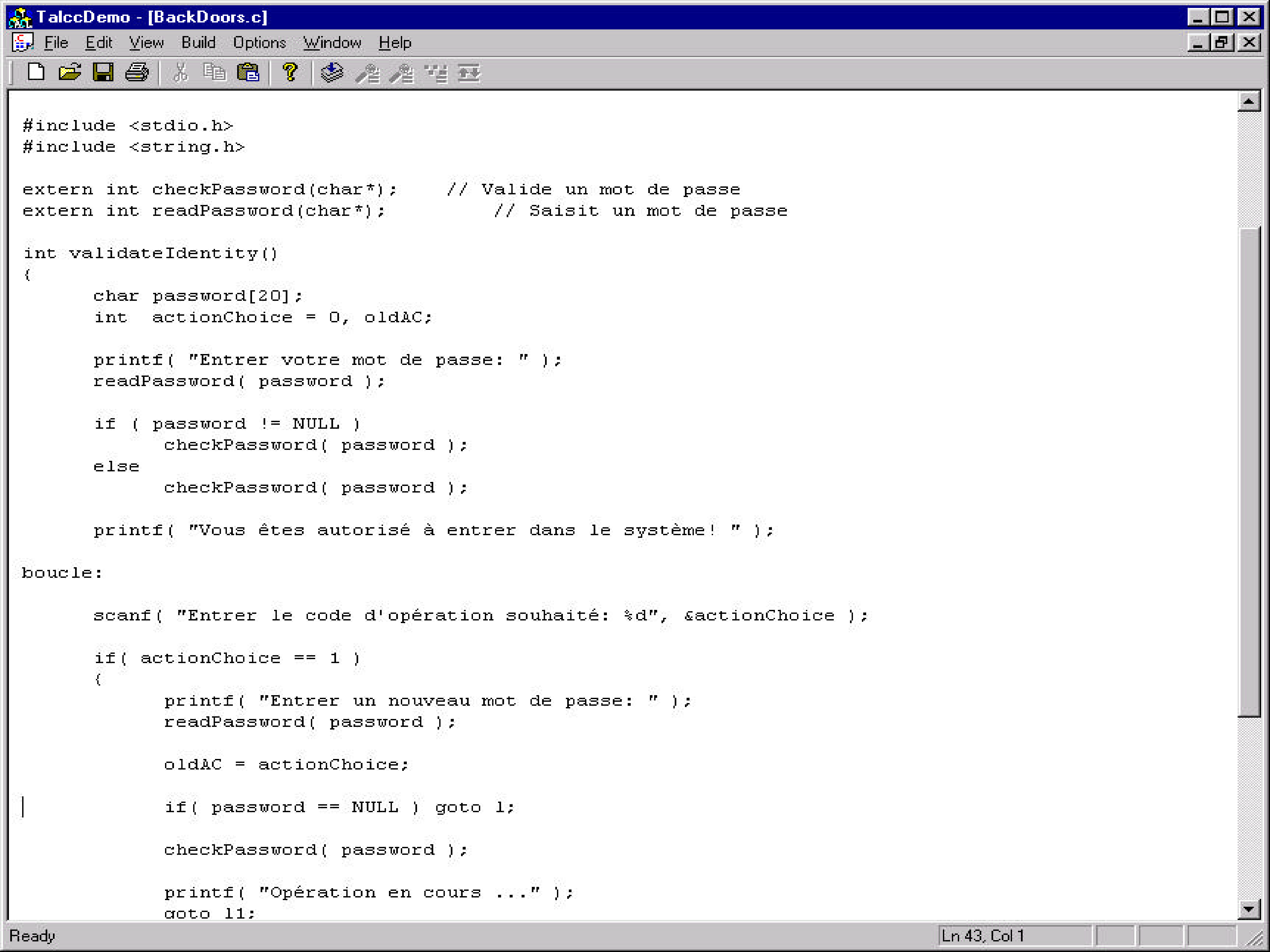
Security Policy

Secure
Linker

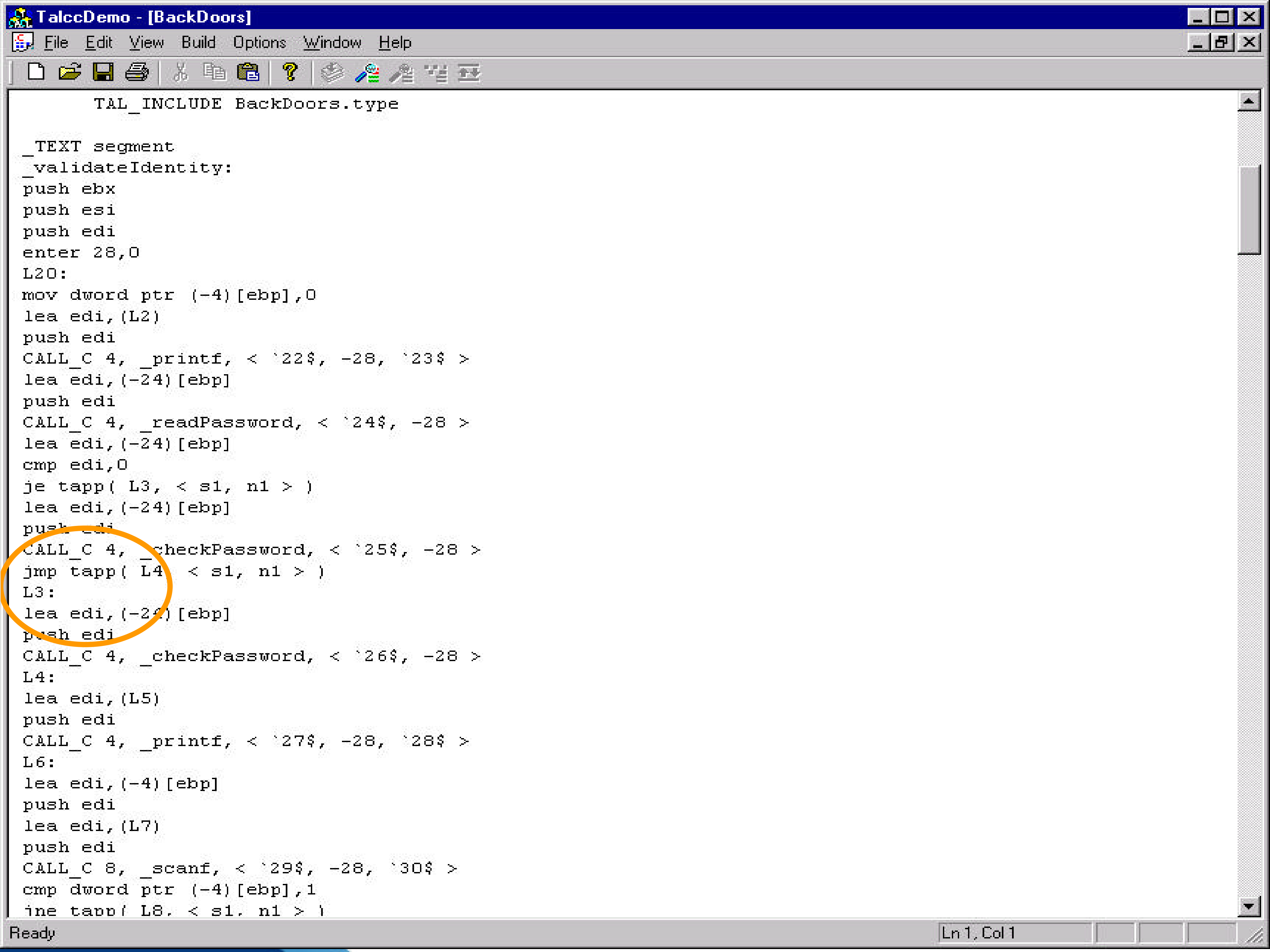
Executable
(x386)

Runtime Monitor in Critical System

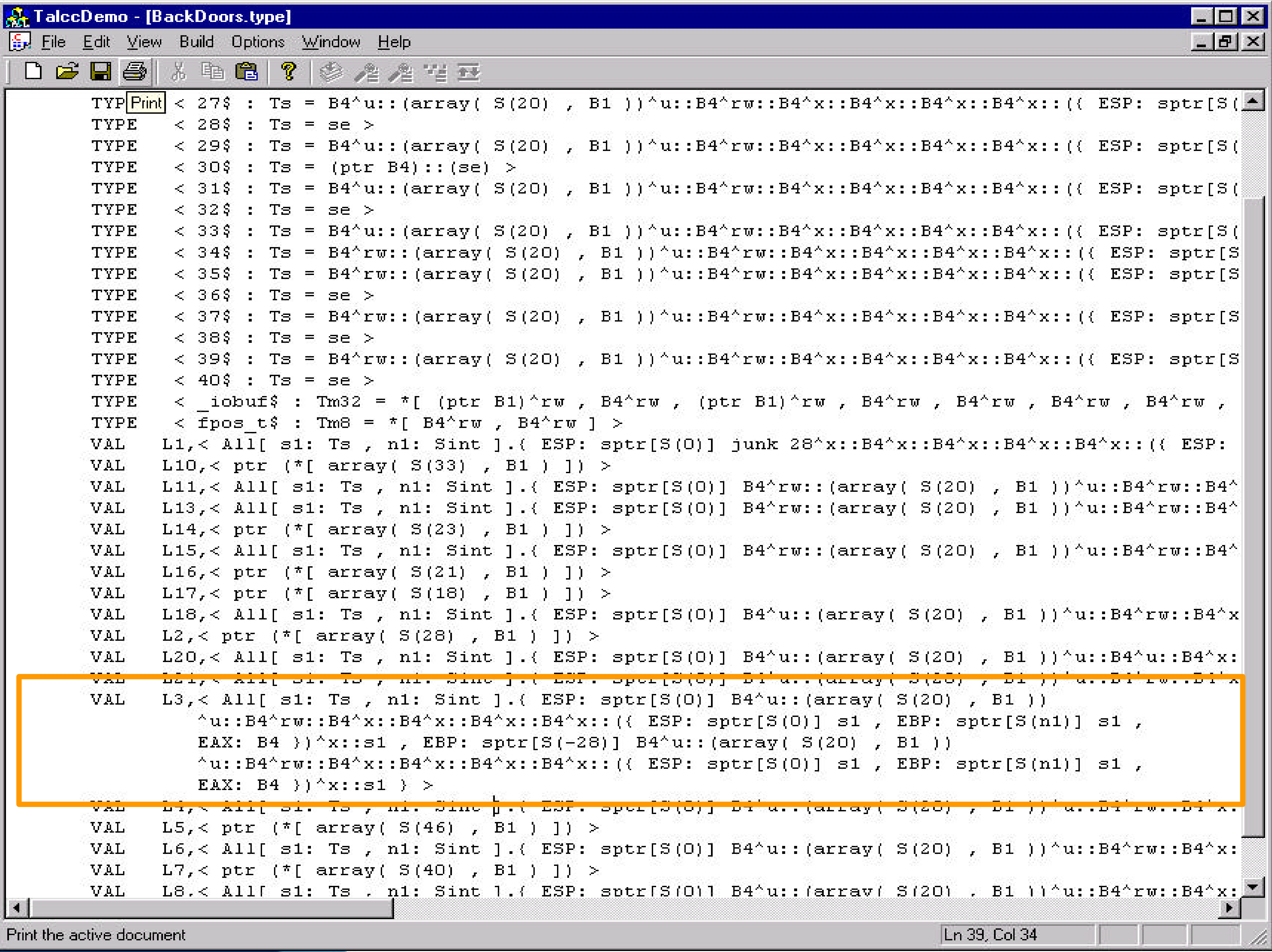

Security Manager
Security Policy
Security Policy
Vendor's Software
Certifying Compiler
In-house Software
Open-Source Software
User
Executable (x386)
Runtime Monitor in Critical System
Executable (x386)
1.
1.1
2.
2.1
2.2
3.
3.1
User's World
Code Producer's World


TalccDemo - [BackDoors.c]
File Edit View Build Options Window Help
Ready
Ln 43, Col 1

```
#include <stdio.h>
#include <string.h>

extern int checkPassword(char*);    // Valide un mot de passe
extern int readPassword(char*);         // Saisit un mot de passe

int validateIdentity()
{
      char password[20];
      int  actionChoice = 0, oldAC;

      printf( "Entrer votre mot de passe: " );
      readPassword( password );

      if ( password != NULL )
            checkPassword( password );
      else
            checkPassword( password );

      printf( "Vous êtes autorisé à entrer dans le système! " );

boucle:

      scanf( "Entrer le code d'opération souhaité: %d", &actionChoice );

      if( actionChoice == 1 )
      {
            printf( "Entrer un nouveau mot de passe: " );
            readPassword( password );

            oldAC = actionChoice;

            if( password == NULL ) goto l;

            checkPassword( password );

            printf( "Opération en cours ..." );
            goto l1;
```



```
       TAL_INCLUDE BackDoors.type

 _TEXT segment
 _validateIdentity:
 push ebx
 push esi
 push edi
 enter 28,0
 L20:
 mov dword ptr (-4)[ebp],0
 lea edi,(L2)
 push edi
 CALL_C 4, _printf, < `22$, -28, `23$ >
 lea edi,(-24)[ebp]
 push edi
 CALL_C 4, _readPassword, < `24$, -28 >
 lea edi,(-24)[ebp]
 cmp edi,0
 je tapp( L3, < s1, n1 > )
 lea edi,(-24)[ebp]
 push edi
 CALL_C 4, _checkPassword, < `25$, -28 >
 jmp tapp( L4, < s1, n1 > )
 L3:
 lea edi,(-24)[ebp]
 push edi
 CALL_C 4, _checkPassword, < `26$, -28 >
 L4:
 lea edi,(L5)
 push edi
 CALL_C 4, _printf, < `27$, -28, `28$ >
 L6:
 lea edi,(-4)[ebp]
 push edi
 lea edi,(L7)
 push edi
 CALL_C 8, _scanf, < `29$, -28, `30$ >
 cmp dword ptr (-4)[ebp],1
 jne tapp( L8, < s1, n1 > )
```



```
        TYPE    < 27$ : Ts = B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(
        TYPE    < 28$ : Ts = se >
        TYPE    < 29$ : Ts = B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(
        TYPE    < 30$ : Ts = (ptr B4)::(se) >
        TYPE    < 31$ : Ts = B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(
        TYPE    < 32$ : Ts = se >
        TYPE    < 33$ : Ts = B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(
        TYPE    < 34$ : Ts = B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S
        TYPE    < 35$ : Ts = B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S
        TYPE    < 36$ : Ts = se >
        TYPE    < 37$ : Ts = B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S
        TYPE    < 38$ : Ts = se >
        TYPE    < 39$ : Ts = B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S
        TYPE    < 40$ : Ts = se >
        TYPE    < _iobuf$ : Tm32 = *[ (ptr B1)^rw , B4^rw , (ptr B1)^rw , B4^rw , B4^rw , B4^rw , B4^rw ,
        TYPE    < fpos_t$ : Tm8 = *[ B4^rw , B4^rw ] >
        VAL    L1,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] junk 28^x::B4^x::B4^x::B4^x::B4^x::({ ESP:
        VAL    L10,< ptr (*[ array( S(33) , B1 ) ]) >
        VAL    L11,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^
        VAL    L13,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^
        VAL    L14,< ptr (*[ array( S(23) , B1 ) ]) >
        VAL    L15,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^rw::(array( S(20) , B1 ))^u::B4^rw::B4^
        VAL    L16,< ptr (*[ array( S(21) , B1 ) ]) >
        VAL    L17,< ptr (*[ array( S(18) , B1 ) ]) >
        VAL    L18,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x
        VAL    L2,< ptr (*[ array( S(28) , B1 ) ]) >
        VAL    L20,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))^u::B4^u::B4^x:
        VAL    L21,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x
        VAL    L3,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))
                 ^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] s1 , EBP: sptr[S(n1)] s1 ,
                 EAX: B4 })^x::s1 , EBP: sptr[S(-28)] B4^u::(array( S(20) , B1 ))
                 ^u::B4^rw::B4^x::B4^x::B4^x::B4^x::({ ESP: sptr[S(0)] s1 , EBP: sptr[S(n1)] s1 ,
                 EAX: B4 })^x::s1 } >
        VAL    L4,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x:
        VAL    L5,< ptr (*[ array( S(46) , B1 ) ]) >
        VAL    L6,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x:
        VAL    L7,< ptr (*[ array( S(40) , B1 ) ]) >
        VAL    L8,< All[ s1: Ts , n1: Sint ].{ ESP: sptr[S(0)] B4^u::(array( S(20) , B1 ))^u::B4^rw::B4^x:
```

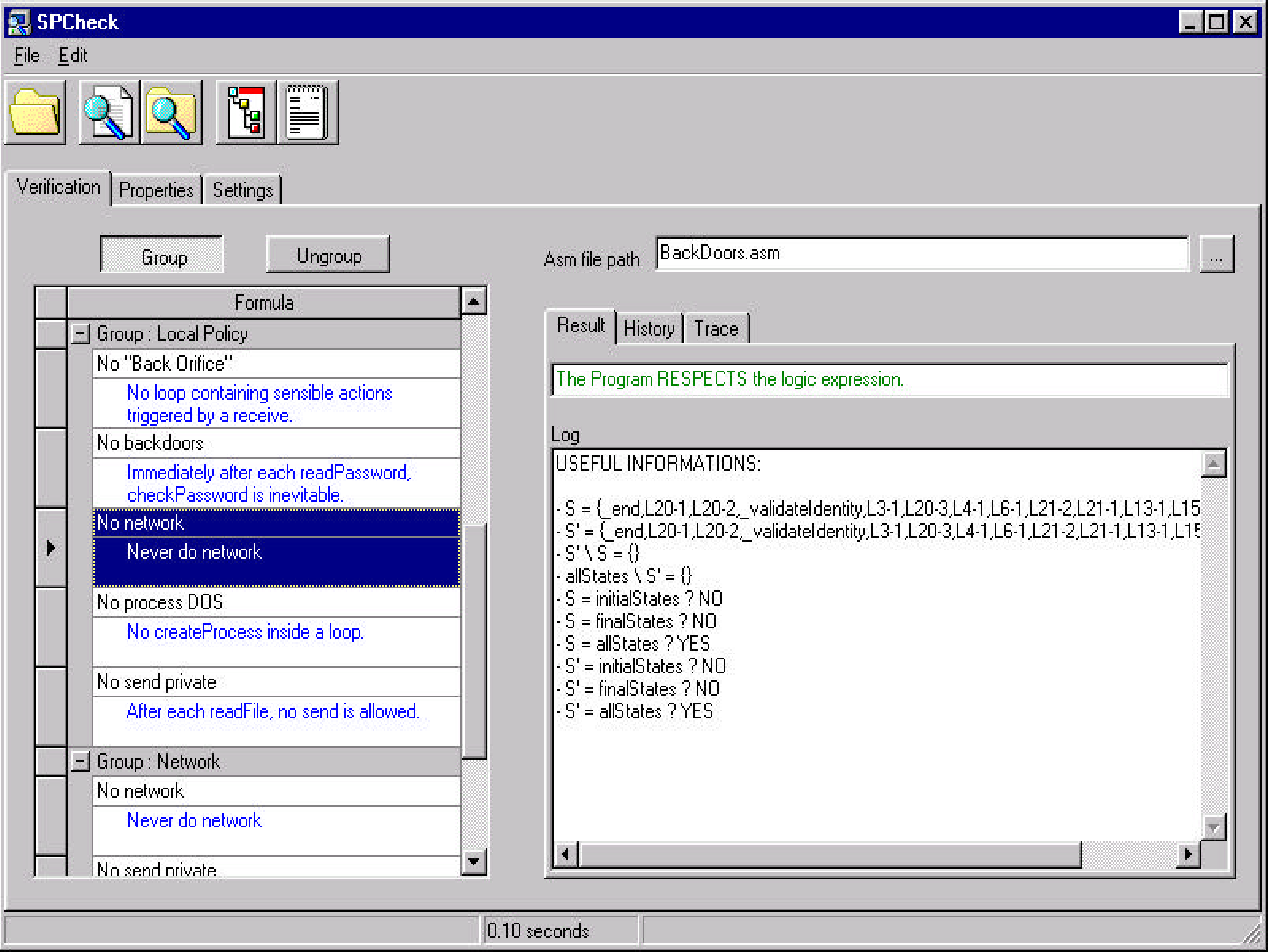
Inference Rules

$$\text{(ArithBin)}\quad \frac{\varepsilon \vdash \boldsymbol{op_1} : \mathrm{B4} \qquad \varepsilon \vdash \boldsymbol{op_2} : \mathrm{B4} \qquad \varepsilon \vdash \mathrm{Writeable}(\boldsymbol{op_1}) \\ \varepsilon \vdash \mathrm{ValidBinops}(\boldsymbol{op_1},\boldsymbol{op_2})}{\varepsilon \vdash \mathtt{\textbf{arithbin}}\ \boldsymbol{op_1},\boldsymbol{op_2} : \varepsilon}$$

$$\text{(Call)}\quad \frac{\varepsilon \vdash \boldsymbol{cop} : \{g1\} \qquad \varepsilon \vdash g_1(\mathrm{esp}) = \ sptr(g_2::c') \\ \varepsilon \vdash \varepsilon.\gamma[\ \mathrm{esp} : sptr\{g_2::(\varepsilon.\gamma(esp))\}\ ] \preceq g_1}{\varepsilon \vdash \mathtt{\textbf{call}}\ \boldsymbol{cop} : \varepsilon[\gamma : g_2\ ]}$$

Safety Properties

- *Control flow safety*
  - Programs cannot jump to code that has not been verified
  - Stack preservation
- *Memory safety*
  - Access to initialized memory locations
  - Array bounds check
- *Type safety*
  - Compatible type in operations
- … , … , ...



Group : Local Policy

No "Back Orifice"
No loop containing sensible actions triggered by a receive.

No backdoors
Immediately after each readPassword, checkPassword is inevitable.

No network
Never do network

No process DOS
No createProcess inside a loop.

No send private
After each readFile, no send is allowed.

Group : Network

No network
Never do network

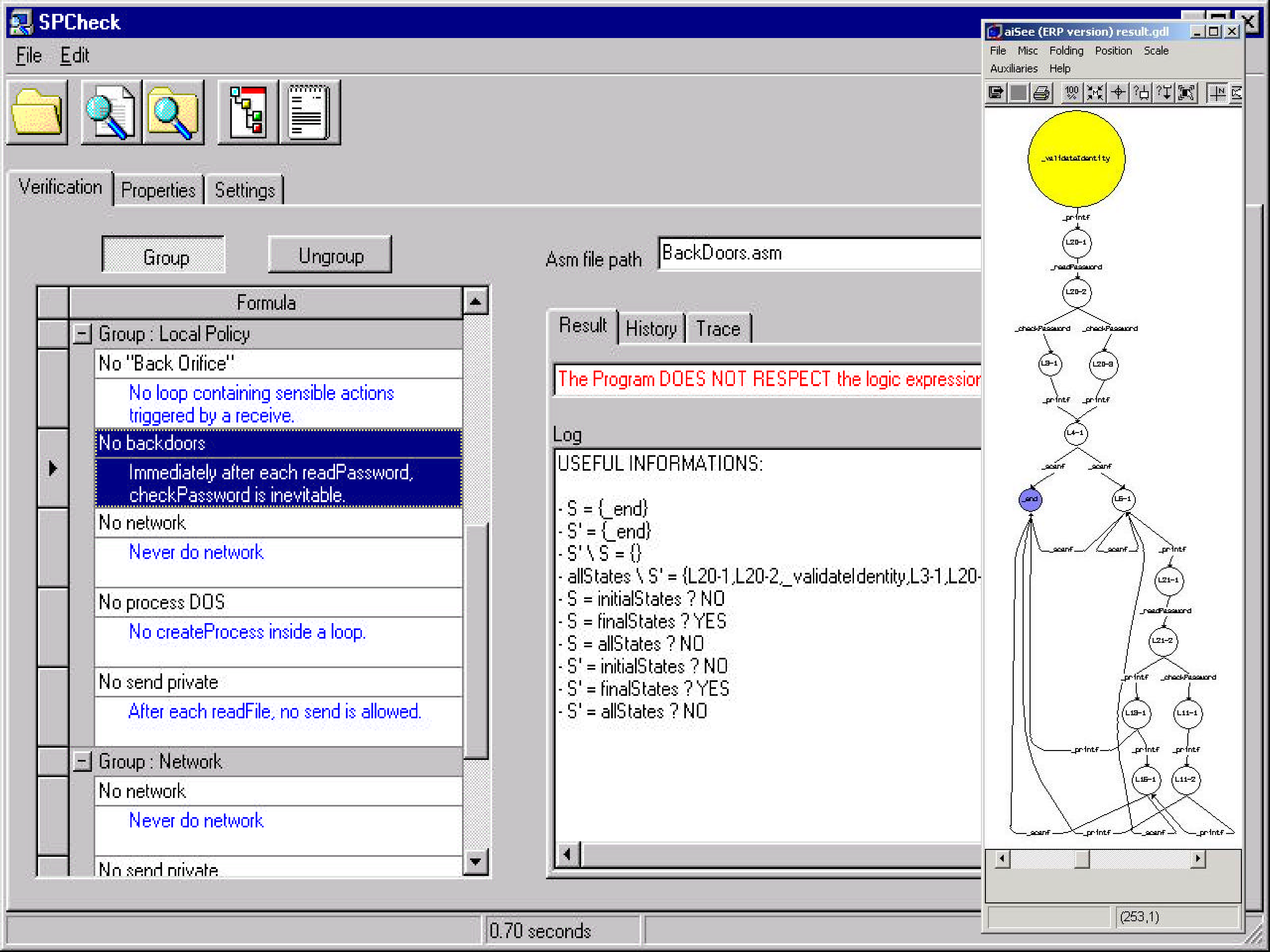
No send private

Asm file path: BackDoors.asm

The Program RESPECTS the logic expression.

```
USEFUL INFORMATIONS:

- S = {_end,L20-1,L20-2,_validateIdentity,L3-1,L20-3,L4-1,L6-1,L21-2,L21-1,L13-1,L15
- S' = {_end,L20-1,L20-2,_validateIdentity,L3-1,L20-3,L4-1,L6-1,L21-2,L21-1,L13-1,L15
- S' \ S = {}
- allStates \ S' = {}
- S = initialStates ? NO
- S = finalStates ? NO
- S = allStates ? YES
- S' = initialStates ? NO
- S' = finalStates ? NO
- S' = allStates ? YES
```

0.10 seconds



Group : Local Policy

No "Back Orifice"

No loop containing sensible actions triggered by a receive.

No backdoors

Immediately after each readPassword, checkPassword is inevitable.

No network

Never do network

No process DOS

No createProcess inside a loop.

No send private

After each readFile, no send is allowed.

Group : Network

No network

Never do network

No send private

Asm file path BackDoors.asm

The Program DOES NOT RESPECT the logic expressior

```
USEFUL INFORMATIONS:

- S = {_end}
- S' = {_end}
- S' \ S = {}
- allStates \ S' = {L20-1,L20-2,_validateIdentity,L3-1,L20-
- S = initialStates ? NO
- S = finalStates ? YES
- S = allStates ? NO
- S' = initialStates ? NO
- S' = finalStates ? YES
- S' = allStates ? NO
```

0.70 seconds

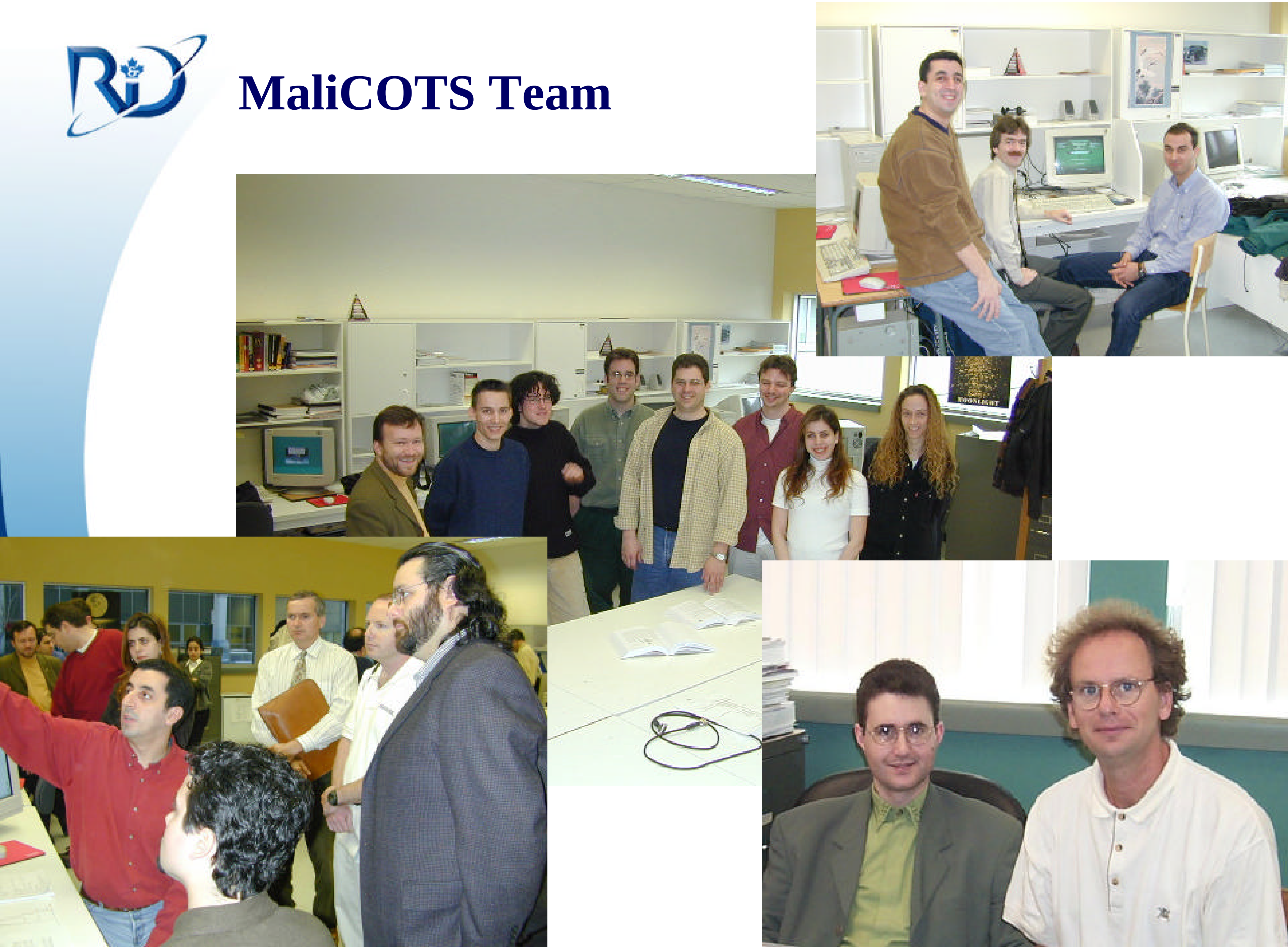
Certifying compiler

- PROS:
  - Large software certification → rapidly
  - Detailed and exhaustive enforcement
  - Intellectual Property is protected
  - Execution is not slowed
  - Possibility of enforcing security, maintainability, and interoperability (...) specifications
- CONS:
  - Emerging technology

MaliCOTS Project Research Outcomes

- Market Surveys / States of the art
- MaliCOTS Prototypes:
  - *SamCOTS* → *Static Code Analyzer*
  - *DaMon* → *Runtime Monitor*
  - *TalCC* → *ANSI C Certifying Compiler*
  - *TalJAVA* → *Java Certifying Compiler*
  - *SPCheck* → *Security Policy Checker*
- Lots of publications
  - http://www.drdc-rddc.gc.ca/researchtech/malicots/home_e.asp
  - Many of them on the CD

MaliCOTS Team



MaliCOTS Team

Professors

Mourad Debbabi Ph.D.

Jean Bergeron Ph.D.

Jules Desharnais Ph.D.

Nadia Tawbi Ph.D.

DRDC Scientists

Robert Charpentier

Martin Salois

M.Sc.

Stéphane Doyon

Mourad Ehrioui

Emmanuel Giasson

Marc Girard

Vincent Labbé

Yvan Lavoie

Frédéric Michaud

Frédéric Painchaud

Ph.D.

Myriam Fourati

Lamia Ketari

Béchir Ktari

Emna Menif

Interns

Sylvain Daigle

Patrice Lamarche

The MaliCOTS Project

A very successful Project

TechnoFed Gold Medal 2000
Partnership

OCTAS
2001

Octas 2001
Future Scientist

CIPA

CANADIAN
INFORMATION
PRODUCTIVITY
AWARDS

CIPA 2001 Institutions Awards

MaliCOTS Conclusions

- The earlier the certification starts in the software design process, the better
- Static and dynamic approaches combined in a test-bench:
  - Offer a short-term solution
  - May be a lengthy and cumbersome process
  - Realistic only for smaller programs such as embedded code

MaliCOTS
Conclusions

- Security policy
  - This is currently the weakest link in most approaches
  - It must be clearly defined to be manageable and enforceable
    - Multiple levels of abstraction
    - Modular
    - Integration at the design level is highly desirable
- Certifying compilers:
  - Emerging technology for large software certification → rapidly
  - Capabilities confirmed by the MaliCOTS prototypes

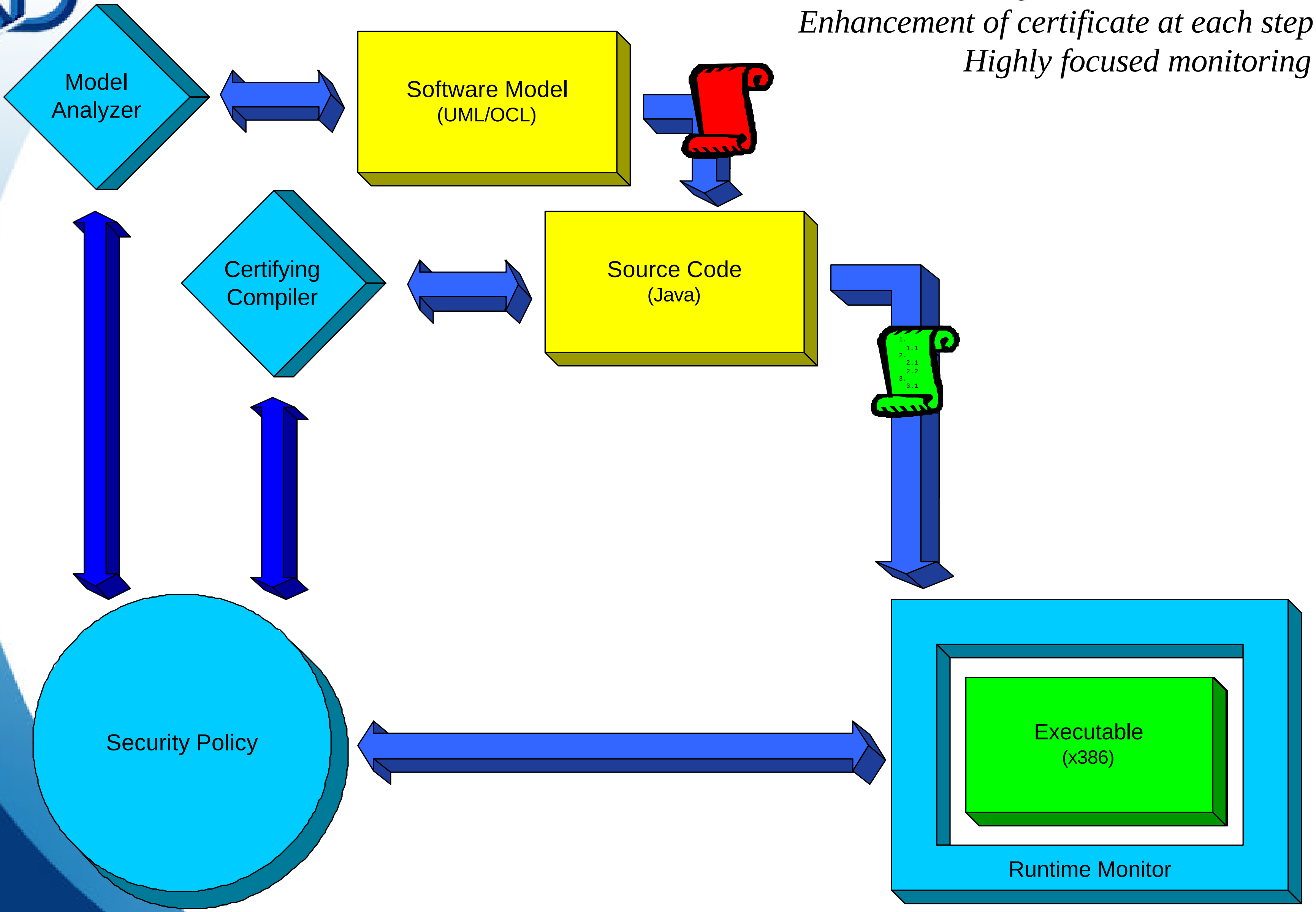
Philosophy:

Maximum risk managed early in the chain

Progressive risk reduction

Enhancement of certificate at each step

Highly focused monitoring



Socle
DEFENCE RD DÉFENSE
(poly

The SOCLe Project

- SOCLe: Secure OCL expressions
  - UML/OCL is a choice of reason, not love
  - Likewise for Java (generated language)
- A UML/OCL prototype
  - Preliminary assurance of coherence/completeness for quality and security
  - Modularization of the design to manage the explosion of the state space
  - Suggest and enforce the use of secure design patterns

The SOCLe Project

- Many studies show that OCL is a good tool
  - Improves quality
    - Laurendeau (1997)
    - Nurun (1999)
  - Can be used for security
    - SecureSoft – OCL Expressivity (2001)
    - SecureSoft – Insider Mitigation (2001)
  - UML/OCL can be formalized to a large extent
    - Polytechnique – Literature Review (2002)
    - Polytechnique - Model-checking OCL Constraints (2002)

The SOCLe Project

- Same type of organisation as MaliCOTS
  - 3 DRDC Scientists
  - 3 Professors at l’École Polytechnique de Montréal
  - 2-3 Ph.D.,
  - 4-5 M.Sc.,
  - + interns
- 6 months feasibility study completed in 2002
- Starting a 2-4 years project

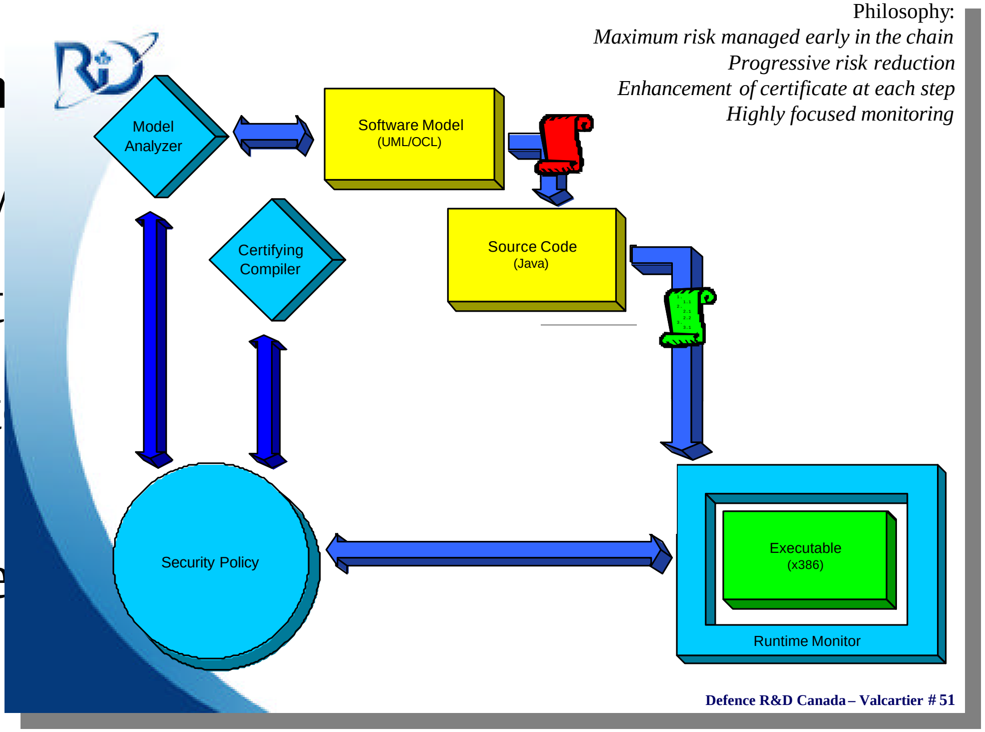
Other projects…

Unified Security Policy

- In all projects, a security policy is needed
- Unifying all the requirements into one security policy language
  - Notes of conformity
    - What has/hasn't been certified
  - Delegation
    - From UML checker to certifying compiler to runtime monitor
  - Identify precisely what's left to verify using traditional testing method

Certified ASN.1

- Following the OULU problems identified last year
- Currently in th
- 2 options (and/
  - A Java cert
    - Generat
  - Certify the completene



Philosophy:
Maximum risk managed early in the chain
Progressive risk reduction
Enhancement of certificate at each step
Highly focused monitoring

Defence R&D Canada – Valcartier # 51

Software Visualization

- A derivative of the knowledge gained with the static analysis prototype (SamCOTS)
- Develop a Canadian expertise
- Develop tools to help in understanding programs without the documentation (or very limited)
- State of the art completed
- Moving on to project definition

Java Hybrid Analysis

- Derivative of the Java certifying compiler (JACC) and the dynamic monitor (DaMon) expertise
- Ph.D. subject for one of our scientists
  - Will be working with a professor and a team of M.Sc. (maybe another Ph.D.)
- Main Idea: To tightly couple the Java certifying compiler with the Java monitor
  - Compensate the weakness of one with the strength of the other

Summary and Perspectives

- Expertise in
  - Static Analysis
  - Dynamic Analysis
  - Certifying Compiler (C and Java)
- Ongoing Projects (approximate timeframe…)
  - UML/OCL formal verification (2-3 years)
  - Unified Security Policies (?)
  - Certified ASN.1 (1-2 years)
  - Software Visualization (2-3 years)
  - Java Hybrid Analysis (3-5 years)
- Quite open to collaboration

DEFENCE R&D DÉFENSE

Martin.Salois@drdc-rddc.gc.ca

http://www.drdc-rddc.gc.ca/researchtech/malicots/home_e.asp