Discovery of Vulnerabilities in Binary Code

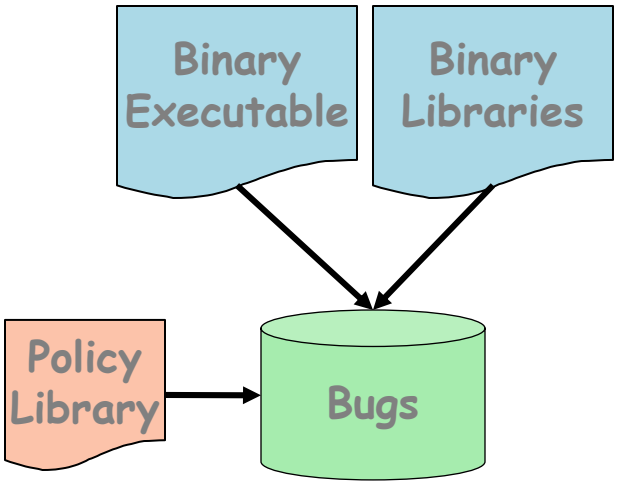
Tim Teitelbaum

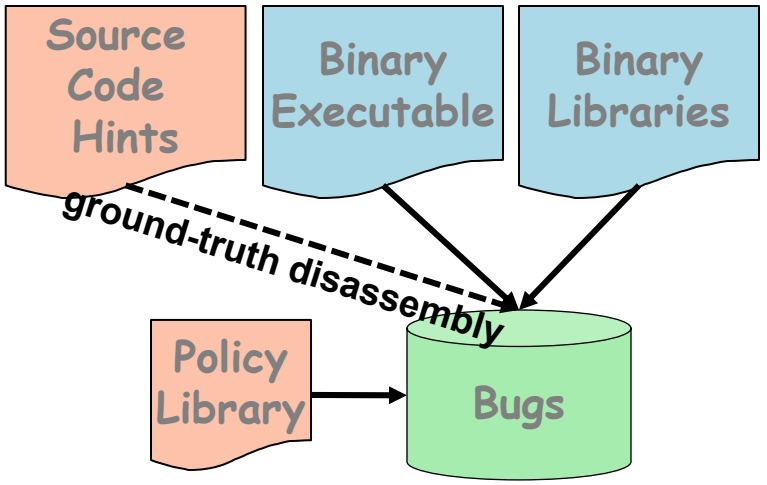
GrammaTech, Inc.
317 North Aurora Street
Ithaca, NY 14850
Tel. 607-273-7340
E-mail: tt@grammatech.com

Executive Summary

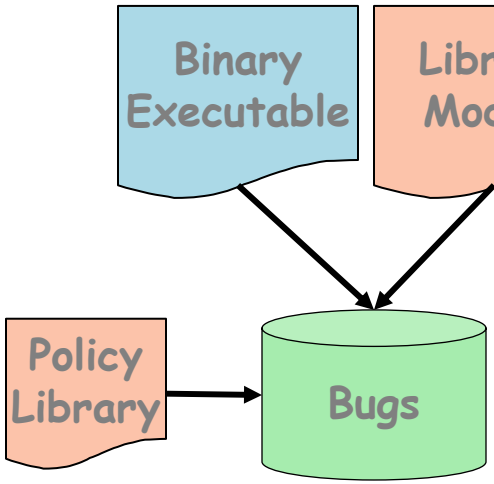
- GrammaTech is developing a static bug and vulnerability finder for binary code (CodeSonar/SWYX)
- An extension of a successful commercial bug finder for source code (CodeSonar/C,C++)
- It will analyze hybrid combinations of source, binaries, and models thereby supporting multiple use cases ranging from development through acceptance testing
- It will be a useful laboratory apparatus for answering scientific questions about source code analysis vs. binary code analysis

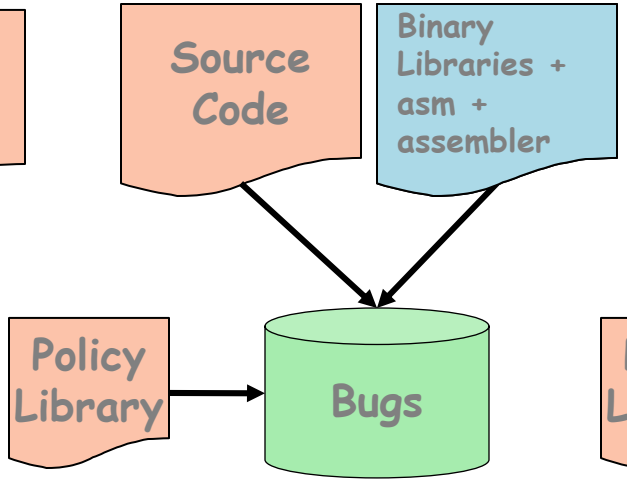
Sample Use Cases

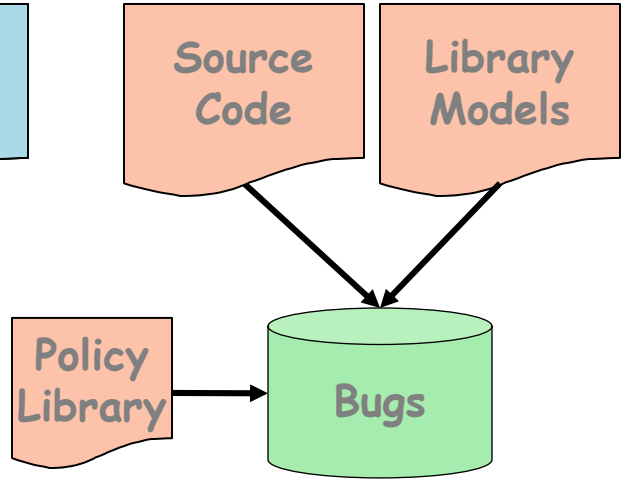




present product







GRAMMATECH

Static Analysis Technologies

	Source Code	Binary Executables
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++	CodeSurfer/x86 CodeSurfer/SWYX, where SWYX is an Instruction Set Architecture
Bug and Vulnerability Finding	CodeSonar/C,C++	CodeSonar/x86 CodeSonar/SWYX, where SWYX is an Instruction Set Architecture

Static Analysis Technologies

	Source Code
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++

CodeSurfer/C,C++

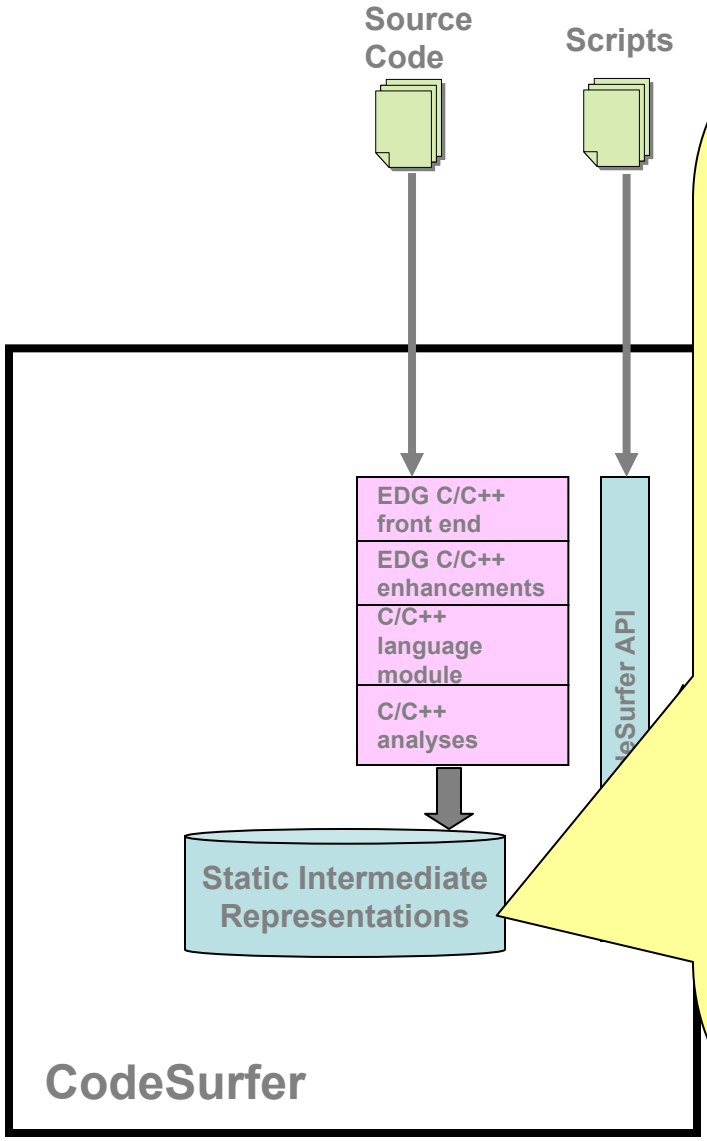
- Builds a fine grained Intermediate Representation (IR) of a whole (or partial) program

Program Understanding Tool

GUI for inspecting program wrt that underlying IR and performing late bound analyses, e.g., slicing, chopping, etc.

Implementation Platform

API for custom scripts and the implementation of derivative applications (in C or Scheme)



- Abstract syntax tree (ASTs)
  [unnormalized & normalized]
- Variables and Types
- Points-to and pointed-to-by sets
  [field & context-sensitive*, flow-insensitive]
- Control flow
  - › Call multi-graphs [direct & indirect]
  - › Basic blocks
  - › Control-flow graphs (CFGs) [interprocedural]
- Data flow
  - › Use, kill, and conditional-kill sets [direct & indirect, per normalized statement]
  - › Non-local def and use sets [direct & indirect, per procedure]
  - › Transitive in/out data dependences [per procedure]
  - › SSA form
- Control dependences
  - › What statements control the execution of what other statements

GRAMMATECH

Static Analysis Technologies

	Source Code
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++

Static Analysis Technologies

	Source Code
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++
Bug and Vulnerability Finding	CodeSonar/C,C++

CodeSonar/C,C++

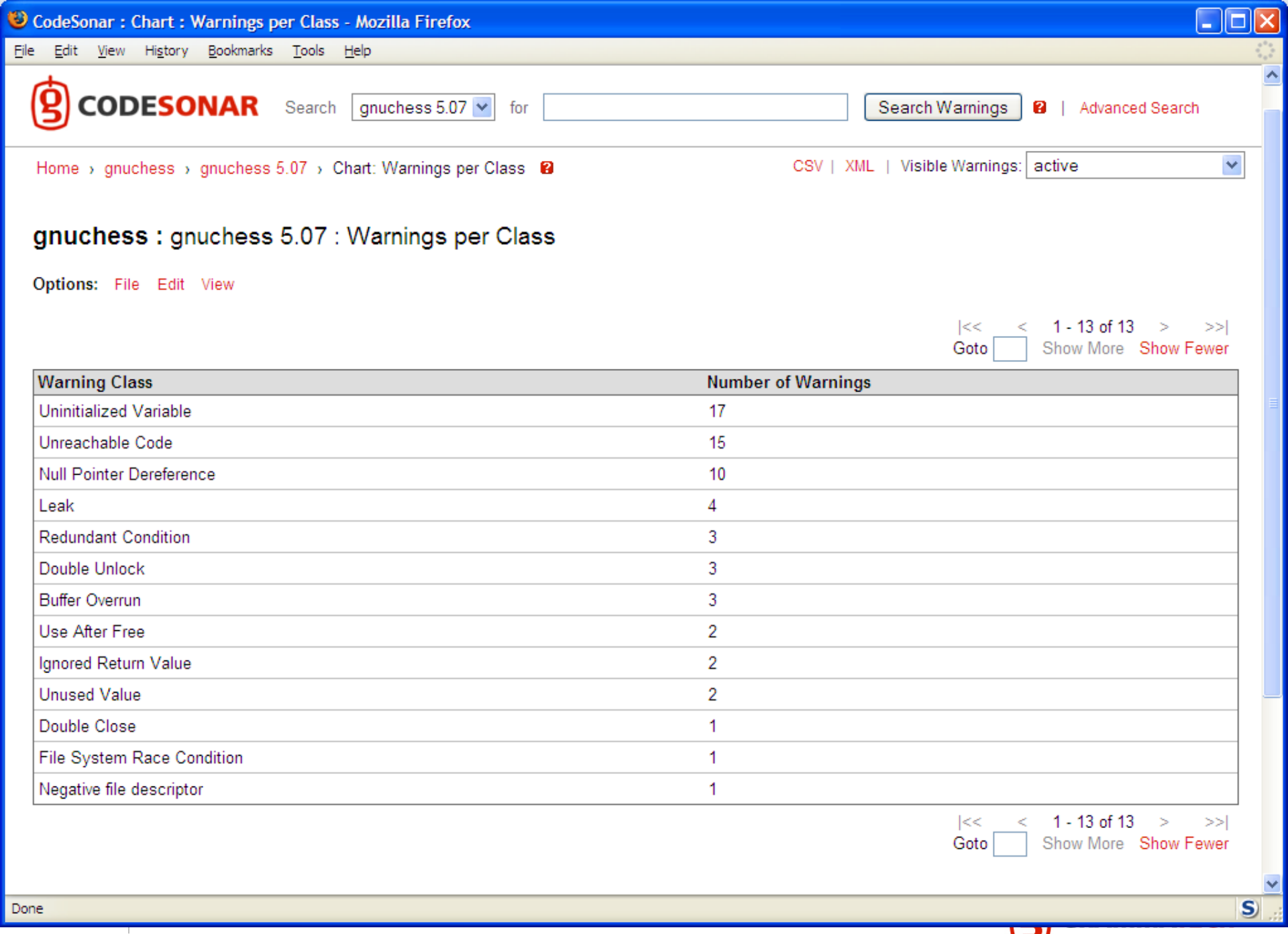
- A source-code analyzer that finds serious flaws in software
  - › Language Misuse
  - › Library Misuse
  - › Enforcement of domain-specific rules
- Sample checks
  - Buffer Overrun
  - Null-Pointer Dereference
  - Divide by Zero
  - Uninitialized Variable
  - Free Null Pointer
  - Unreachable Code
  - Dangerous Cast
  - Missing Return Statement
  - Return Pointer to Local
  - Format String Vulnerability
  - Free Non-Heap Variable
  - Use After Free/Close
  - Double Free/Close
  - Memory/Resource Leak
  - Mismatched Array New/Delete
  - Invalid Parameter
  - Unchecked Return Code
  - Race Condition

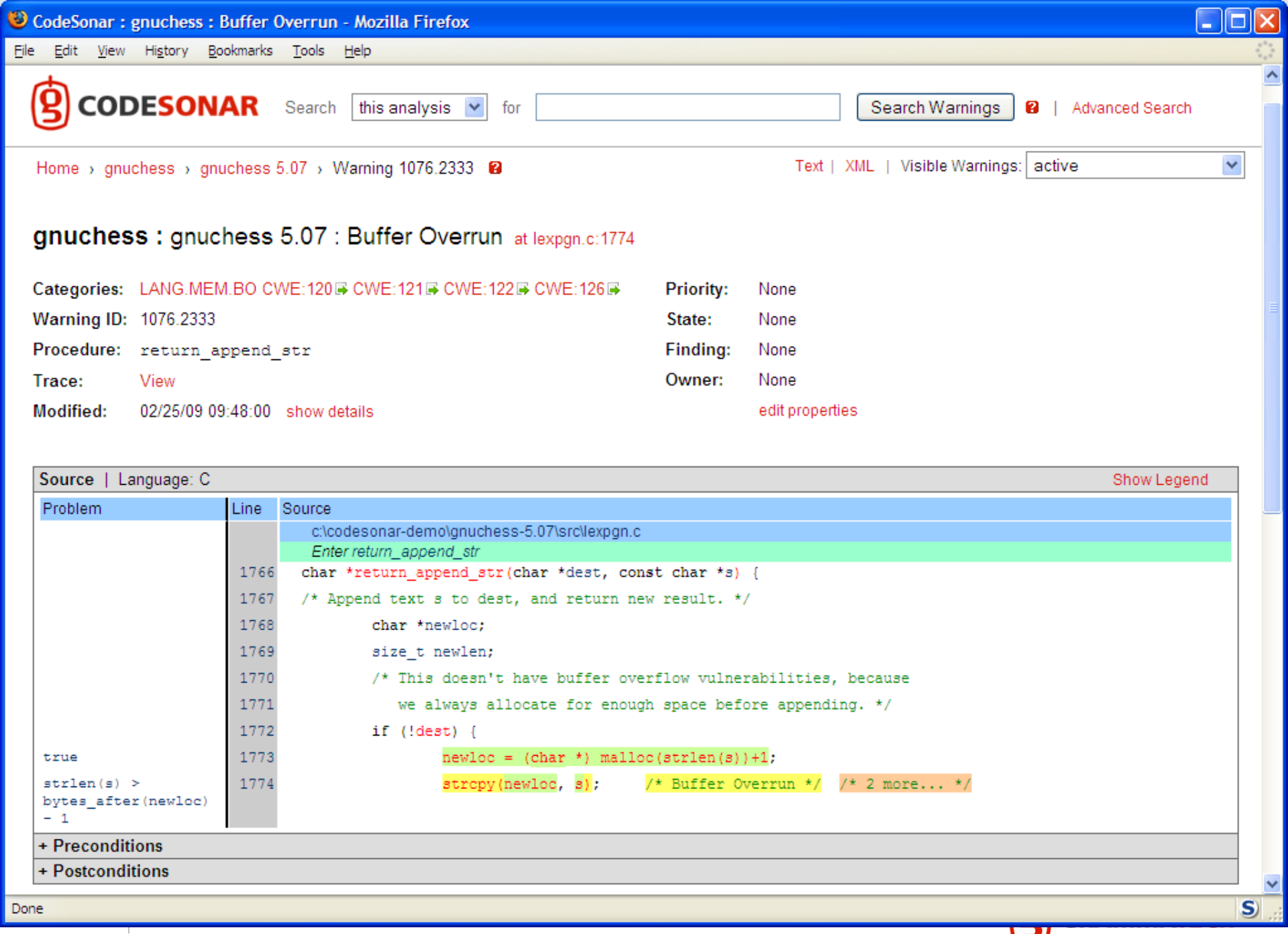
CodeSonar/C,C++ (Cont'd)

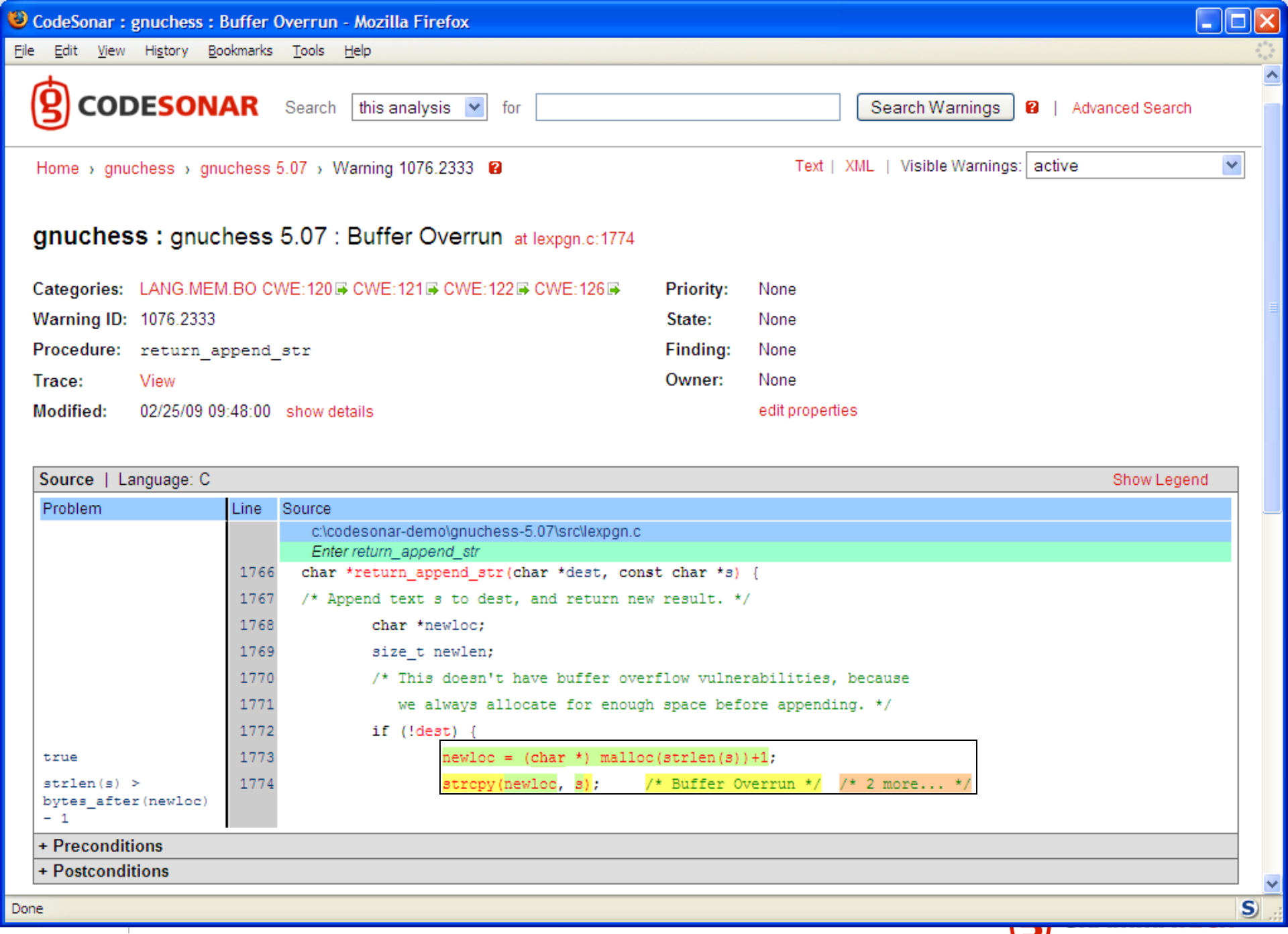
- Whole or partial program
- Fast and highly scalable
- Low false positive rate
  - At cost of missing some flaws
- Easy to invoke analysis
  - No source-code annotations necessary
  - Piggybacks on existing build systems
- Browser-based user interface
  - Web server and database for managing results
  - Setup can be performed by one user; other users point browsers to server and log in

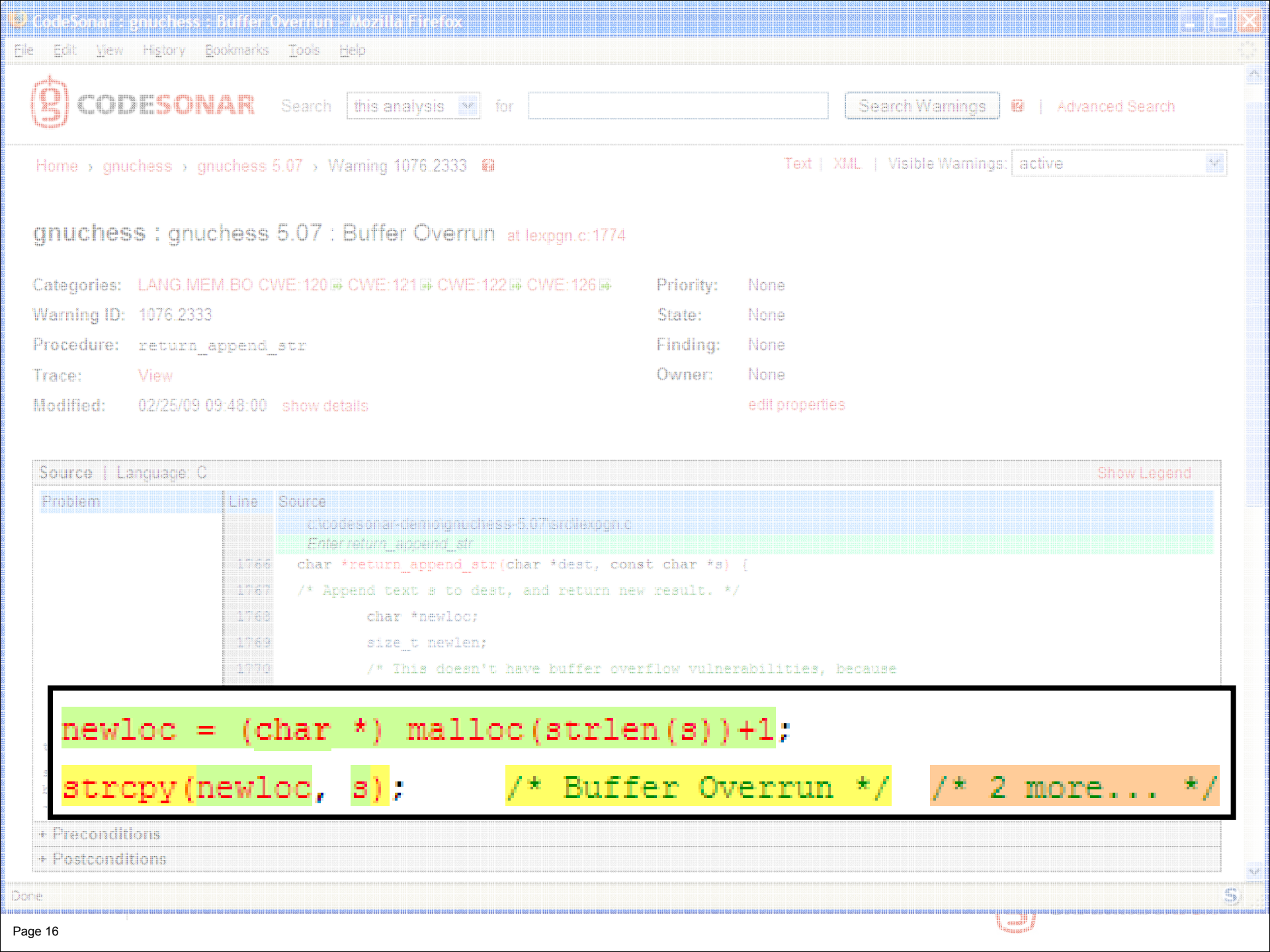
CodeSonar/C,C++ (Cont'd)

- Search interface for results database
  - Custom views of results are easy to create
  - Simple searching, advanced searching, and SQL interpreter
- User-added state information persists across builds
  - Mechanism used to suppress false positives
  - Labels and comments can be attached to a bug warning
- Interprocedural, flow sensitive, context sensitive, path sensitive, object sensitive symbolic execution


CodeSonar : Chart : Warnings per Class - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CODESONAR
Search
gnuchess 5.07
for
Search Warnings
Advanced Search
Home › gnuchess › gnuchess 5.07 › Chart: Warnings per Class
CSV | XML | Visible Warnings: active
gnuchess : gnuchess 5.07 : Warnings per Class
Options: File Edit View
|<< < 1 - 13 of 13 > >>|
Goto
Show More
Show Fewer
Warning Class
Number of Warnings
Uninitialized Variable 17
Unreachable Code 15
Null Pointer Dereference 10
Leak 4
Redundant Condition 3
Double Unlock 3
Buffer Overrun 3
Use After Free 2
Ignored Return Value 2
Unused Value 2
Double Close 1
File System Race Condition 1
Negative file descriptor 1
|<< < 1 - 13 of 13 > >>|
Goto
Show More
Show Fewer
Done

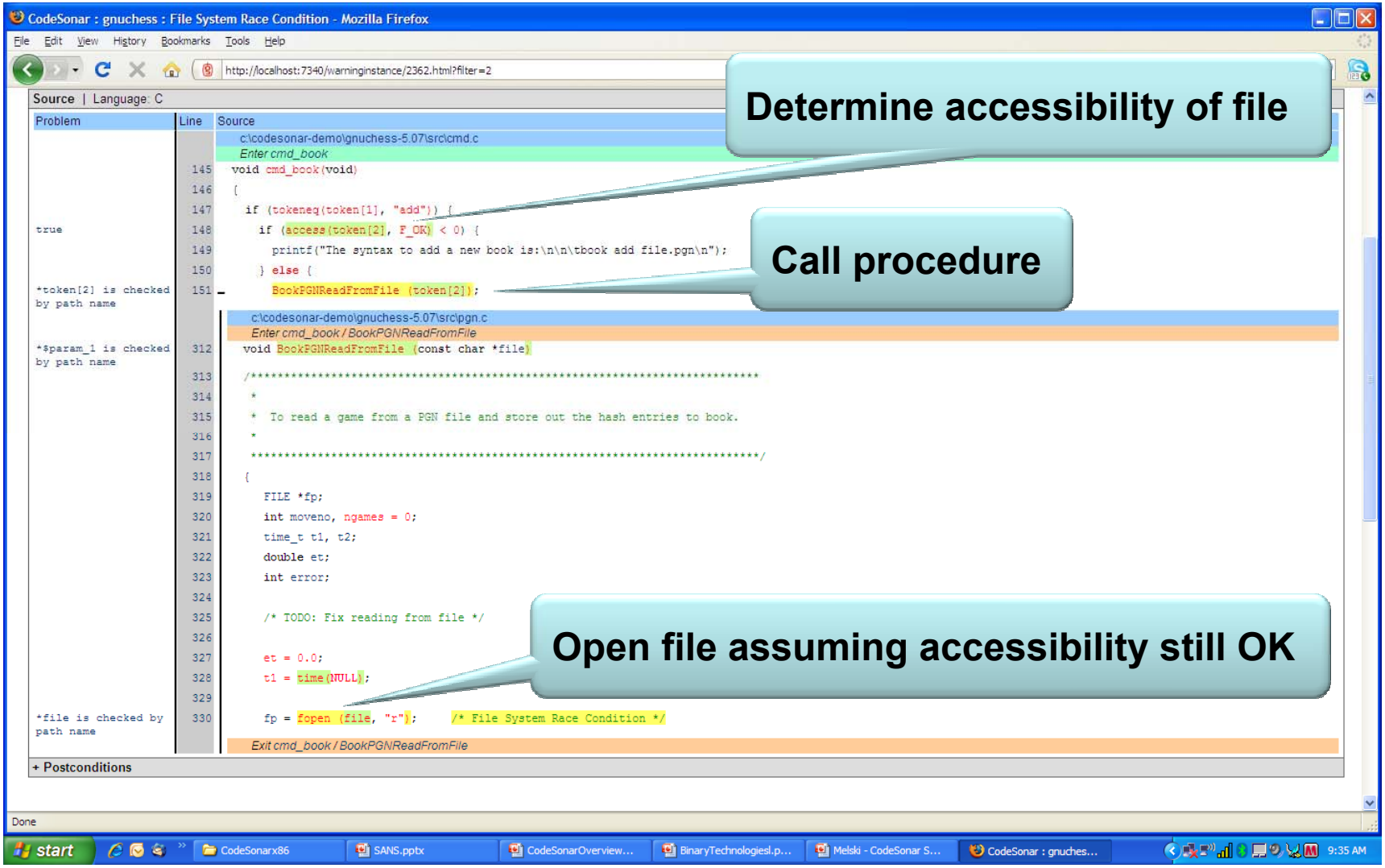

CodeSonar : gnuchess : Buffer Overrun - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CODESONAR
Search this analysis for Search Warnings | Advanced Search
Home › gnuchess › gnuchess 5.07 › Warning 1076.2333
Text | XML | Visible Warnings: active
gnuchess : gnuchess 5.07 : Buffer Overrun at lexpgn.c:1774
Categories: LANG.MEM.BO CWE:120 CWE:121 CWE:122 CWE:126
Warning ID: 1076.2333
Procedure: return_append_str
Trace: View
Modified: 02/25/09 09:48:00 show details
Priority: None
State: None
Finding: None
Owner: None
edit properties
Source | Language: C
Show Legend
Problem
Line
Source
c:\codesonar-demo\gnuchess-5.07\src\lexpgn.c
Enter return_append_str
1766 char *return_append_str(char *dest, const char *s) {
1767 /* Append text s to dest, and return new result. */
1768 char *newloc;
1769 size_t newlen;
1770 /* This doesn't have buffer overflow vulnerabilities, because
1771 we always allocate for enough space before appending. */
1772 if (!dest) {
true
1773 newloc = (char *) malloc(strlen(s))+1;
strlen(s) > bytes_after(newloc) - 1
1774 strcpy(newloc, s); /* Buffer Overrun */ /* 2 more... */
+ Preconditions
+ Postconditions
Done


CodeSonar : gnuchess : Buffer Overrun - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CODESONAR
Search this analysis for Search Warnings | Advanced Search
Home › gnuchess › gnuchess 5.07 › Warning 1076.2333
Text | XML | Visible Warnings: active
gnuchess : gnuchess 5.07 : Buffer Overrun at lexpgn.c:1774
Categories: LANG.MEM.BO CWE:120 CWE:121 CWE:122 CWE:126
Warning ID: 1076.2333
Procedure: return_append_str
Trace: View
Modified: 02/25/09 09:48:00 show details
Priority: None
State: None
Finding: None
Owner: None
edit properties
Source | Language: C
Show Legend
Problem
Line
Source
c:\codesonar-demo\gnuchess-5.07\src\lexpgn.c
Enter return_append_str
1766 char *return_append_str(char *dest, const char *s) {
1767 /* Append text s to dest, and return new result. */
1768 char *newloc;
1769 size_t newlen;
1770 /* This doesn't have buffer overflow vulnerabilities, because
1771 we always allocate for enough space before appending. */
1772 if (!dest) {
true
1773 newloc = (char *) malloc(strlen(s))+1;
strlen(s) > bytes_after(newloc) - 1
1774 strcpy(newloc, s); /* Buffer Overrun */ /* 2 more... */
+ Preconditions
+ Postconditions
Done


CODESONAR
gnuchess : gnuchess 5.07 : Buffer Overrun
newloc = (char *) malloc(strlen(s))+1;
strcpy(newloc, s);
/* Buffer Overrun */
/* 2 more... */

Sample Inter-Procedural Warning

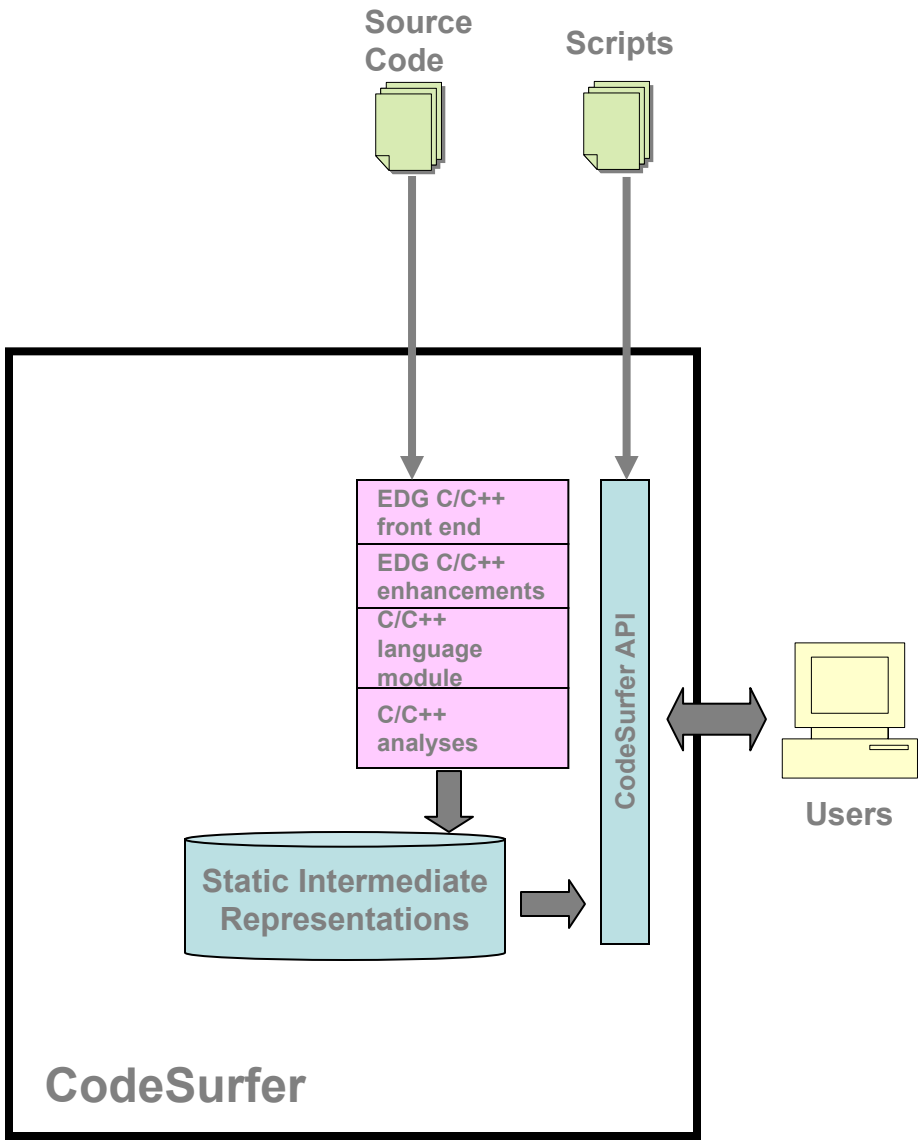
File System Race Condition (*TOC*/TOU)



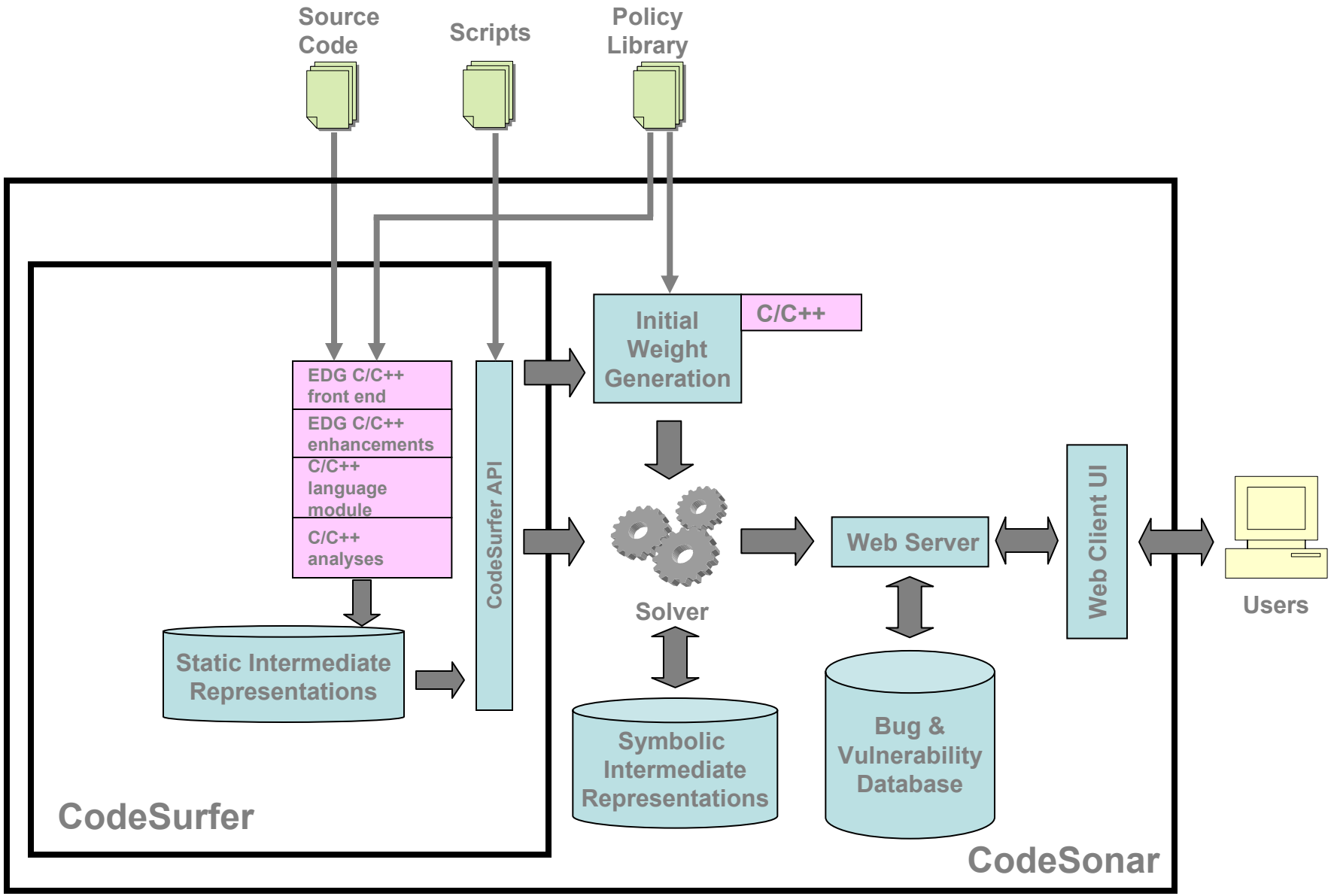
GRAMMATECH

CodeSonar C,C++ Effectiveness

- FDA used **CodeSonar** to examine 200 KLOC C program for medical device experiencing problems in the field
- Results
  - 127 serious problems detected
    - 29 unsafe casts
    - 28 null pointer dereferences
    - 36 uninitialized variables
    - 20 unreachable code fragments
    - 14 others
  - 82 of 127 had been found by manufacturer using manual inspection
  - 45 of 127 were not previously known
- See April 2008 *Embedded Systems Design Magazine* for details


Source Code
Scripts
EDG C/C++ front end
EDG C/C++ enhancements
C/C++ language module
C/C++ analyses
CodeSurfer API
Users
Static Intermediate Representations
CodeSurfer

GRAMMATECH


Source Code
Scripts
Policy Library
Initial Weight Generation
C/C++
EDG C/C++ front end
EDG C/C++ enhancements
C/C++ language module
C/C++ analyses
CodeSurfer API
Static Intermediate Representations
Solver
Symbolic Intermediate Representations
Web Server
Bug & Vulnerability Database
Web Client UI
Users
CodeSurfer
CodeSonar

GRAMMATECH

Solver

- Similar to other light-weight bug-finding products
- Visits each procedure once in bottom-up pass of call graph
- At each procedure
  - Breadth-first exploration of intra-procedural paths using summaries (previously computed) for functions called
  - Create a (truncated) summary of procedure's side-effects and error conditions
- Logic uses affine relations of two variables
- Analysis uses many heuristics:
  - Unsound
  - Highly scalable
  - Reasonable false positive rate

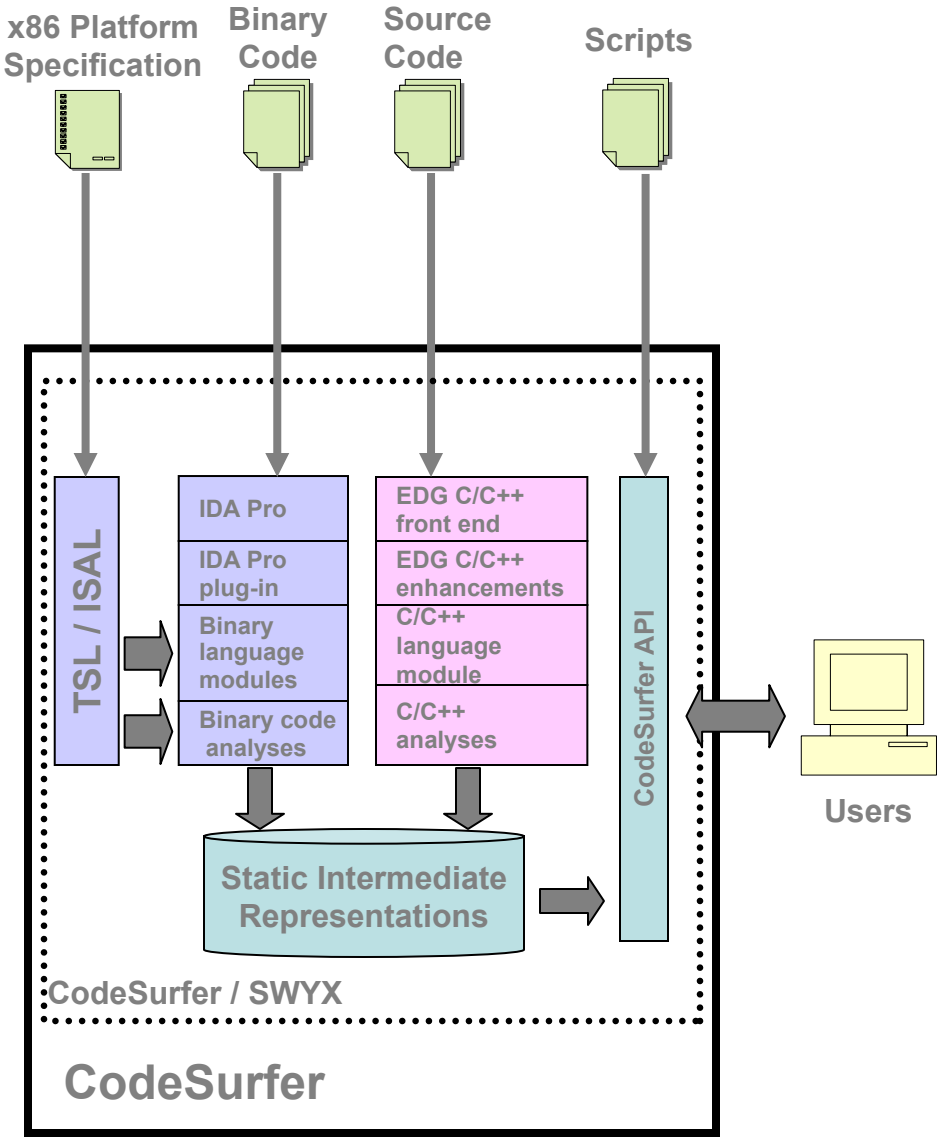
Static Analysis Technologies

	Source Code
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++
Bug and Vulnerability Finding	CodeSonar/C,C++

Static Analysis Technologies

	Source Code	Binary Executables
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++	CodeSurfer/x86 CodeSurfer/SWYX, where SWYX is an Instruction Set Architecture
Bug and Vulnerability Finding	CodeSonar/C,C++	

GRAMMATECH


x86 Platform Specification
Binary Code
Source Code
Scripts
TSL / ISAL
IDA Pro
IDA Pro plug-in
Binary language modules
Binary code analyses
EDG C/C++ front end
EDG C/C++ enhancements
C/C++ language module
C/C++ analyses
CodeSurfer API
Users
Static Intermediate Representations
CodeSurfer / SWYX
CodeSurfer

GRAMMATECH

Instruction Set Architecture (ISA) Independence

- TSL / ISAL
  - Recursive Types for defining AST representations of instructions
  - Grammar for specifying bit-level instruction layout
  - Parser + Syntax Directed Translation for instruction decode
  - Language for defining the operational semantics of instructions
  - Framework for defining static analyses about machine code in an ISA-independent manner
  - System for generating static-analysis implementations
    - Pairing of an ISA and an analysis is automatic
- Benefits
  - Independence of semantics and analyses
    - Validation of each ISA semantics is separate from static analyses
    - Validation of each static analysis is separate from ISA definitions
  - Consistency. All analyses for given ISA driven off of same definition
  - Completeness. Full analysis generated for all instructions.

Static Analysis Technologies

	Source Code	Binary Executables
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++	CodeSurfer/x86 CodeSurfer/SWYX, where SWYX is an Instruction Set Architecture
Bug and Vulnerability Finding	CodeSonar/C,C++	

Static Analysis Technologies

	Source Code	Binary Executables
Understanding, Reverse-Engineering, Rewriting, and Infrastructure	CodeSurfer/C,C++	CodeSurfer/x86 CodeSurfer/SWYX, where SWYX is an Instruction Set Architecture
Bug and Vulnerability Finding	CodeSonar/C,C++	CodeSonar/x86 CodeSonar/SWYX, where SWYX is an Instruction Set Architecture

Why CodeSonar for Binaries?

- Source code often unavailable, e.g., libraries, COTS
- Even when when available, often infeasible to configure to match release of interest
- Fidelity: may actually be better than source code analysis
  - **WYSINWYX**: What You See Is Not What You eXecute

```
memset(password, '\0', len);
free(password);
```

  - Binaries reveal platform-specific choices of compiler
  - Binary analysis can use real libraries, not hand-written models
  - Convenience
  - Supports post-development business model
  - Works for applications written in any compiled language(s)
  - But needs approach to variations in Instruction Set Architectures

gnuchess.exe : gnuchess.exe analysis 3 : Warnings per Class

Options: File Edit View

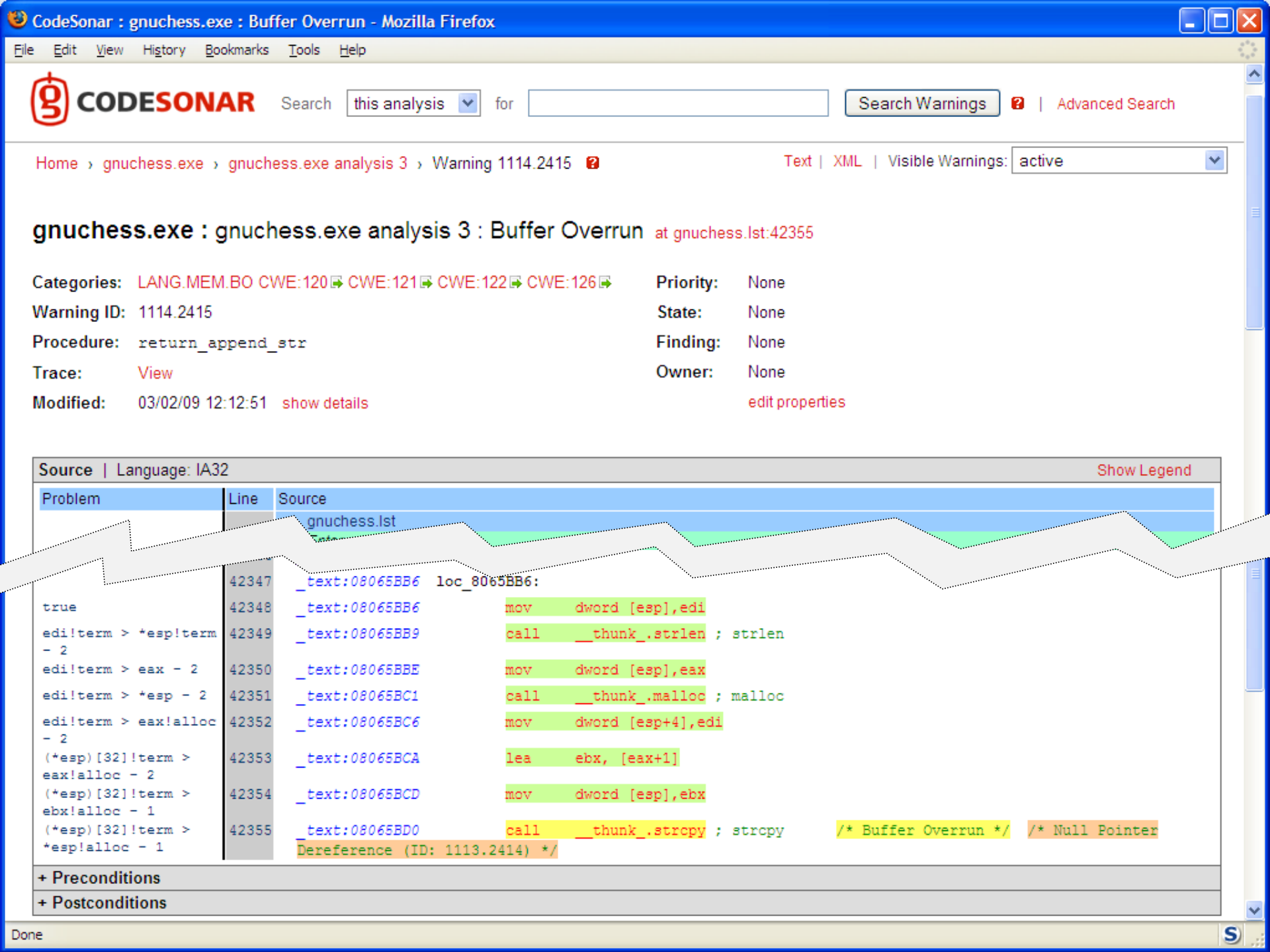
|<< < 1 - 10 of 10 > >>|
Goto Show More Show Fewer

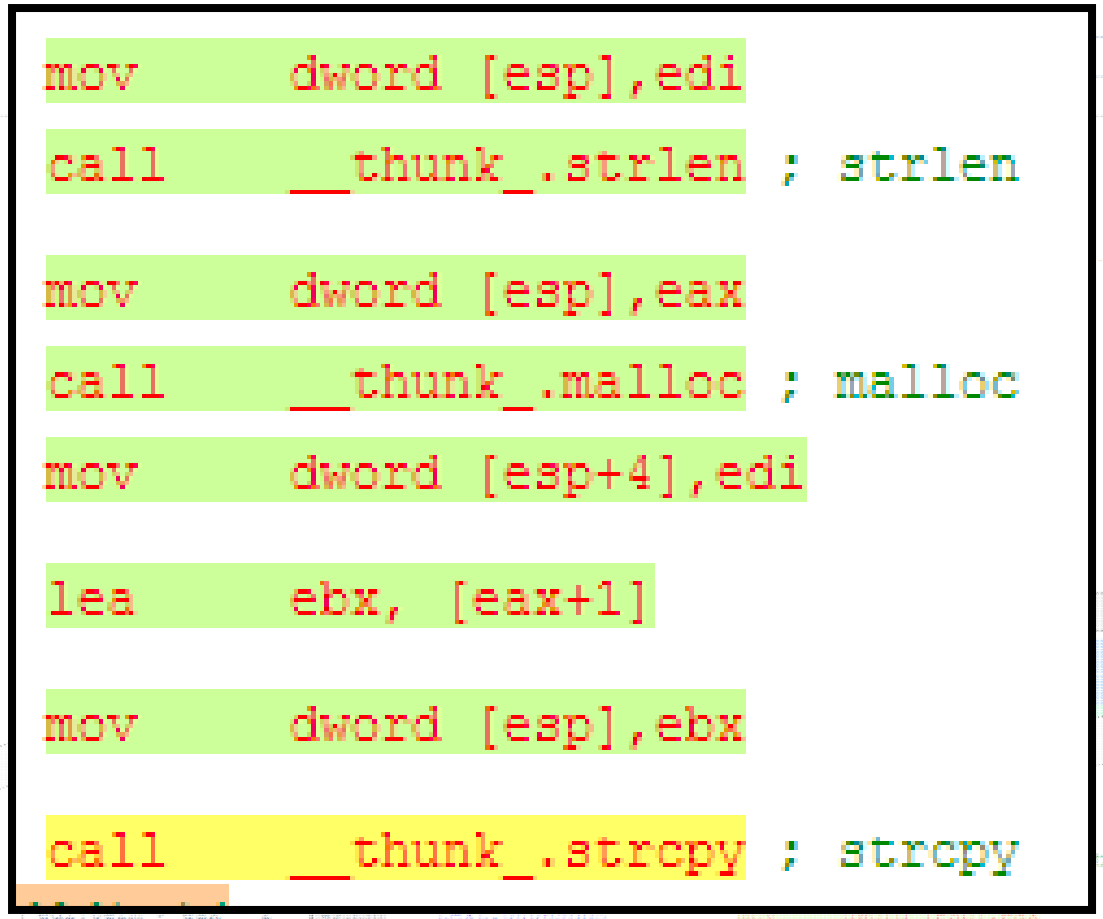
Warning Class	Number of Warnings
Unreachable Code	32
Leak	10
Null Pointer Dereference	10
Useless Assignment	6
Redundant Condition	5
Use After Free	2
Cast Alters Value	1
Negative file descriptor	1
Buffer Overrun	1
Double Close	1

|<< < 1 - 10 of 10 > >>|
Goto Show More Show Fewer

Details

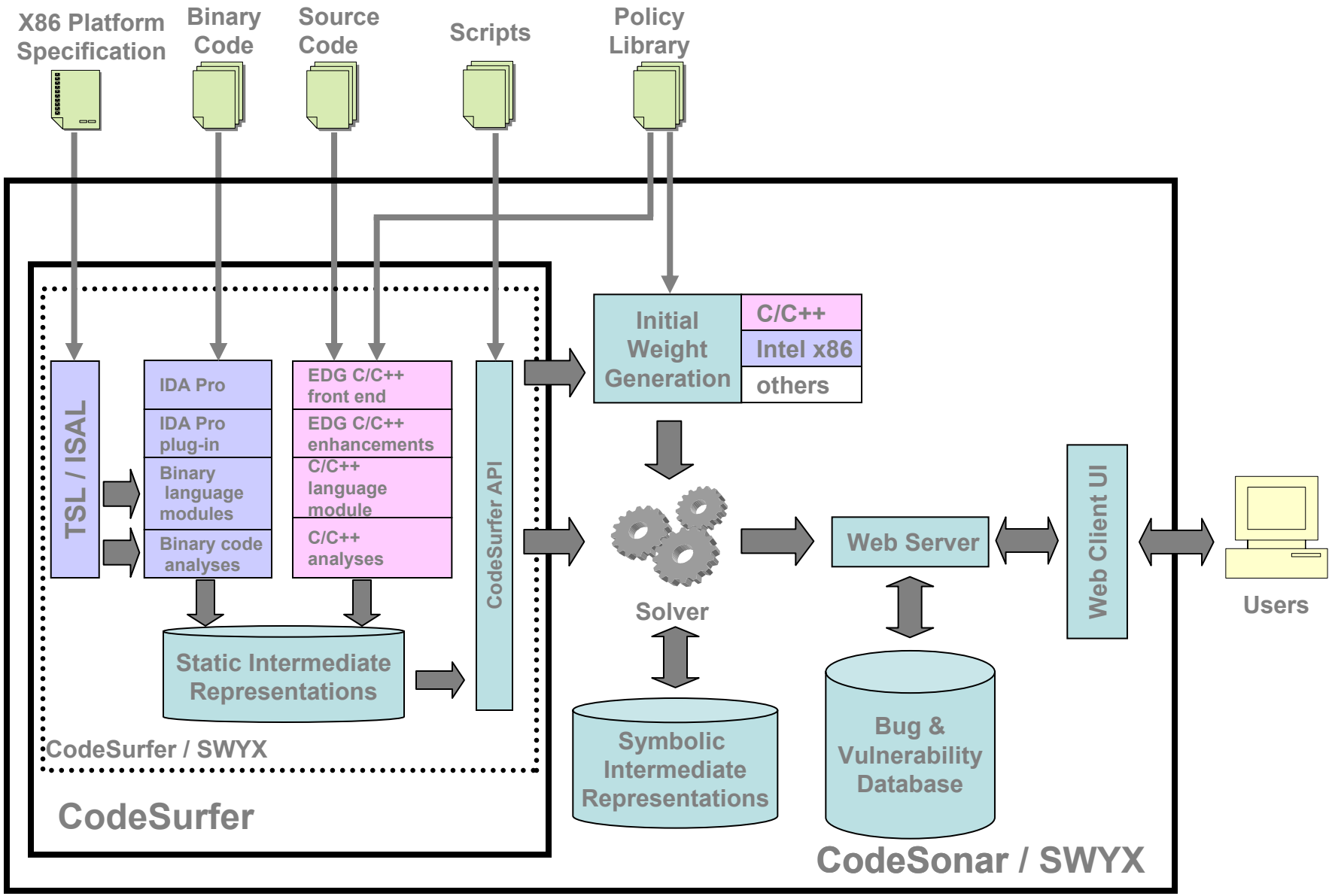
Searched all warnings in gnuchess.exe analysis 3


CodeSonar : gnuchess.exe : Buffer Overrun - Mozilla Firefox
File Edit View History Bookmarks Tools Help
CODESONAR
Search this analysis for Search Warnings | Advanced Search
Home › gnuchess.exe › gnuchess.exe analysis 3 › Warning 1114.2415
Text | XML | Visible Warnings: active
gnuchess.exe : gnuchess.exe analysis 3 : Buffer Overrun at gnuchess.lst:42355
Categories: LANG.MEM.BO CWE:120 CWE:121 CWE:122 CWE:126
Warning ID: 1114.2415
Procedure: return_append_str
Trace: View
Modified: 03/02/09 12:12:51 show details
Priority: None
State: None
Finding: None
Owner: None
edit properties
Source | Language: IA32
Show Legend
Problem
Line
Source
gnuchess.lst
42347 _text:08065BB6 loc_8065BB6:
true
42348 _text:08065BB6 mov dword [esp],edi
edi!term > *esp!term - 2
42349 _text:08065BB9 call __thunk_.strlen ; strlen
edi!term > eax - 2
42350 _text:08065BBE mov dword [esp],eax
edi!term > *esp - 2
42351 _text:08065BC1 call __thunk_.malloc ; malloc
edi!term > eax!alloc - 2
42352 _text:08065BC6 mov dword [esp+4],edi
(*esp)[32]!term > eax!alloc - 2
42353 _text:08065BCA lea ebx, [eax+1]
(*esp)[32]!term > ebx!alloc - 1
42354 _text:08065BCD mov dword [esp],ebx
(*esp)[32]!term > *esp!alloc - 1
42355 _text:08065BD0 call __thunk_.strcpy ; strcpy /* Buffer Overrun */ /* Null Pointer Dereference (ID: 1113.2414) */
+ Preconditions
+ Postconditions
Done



```
mov     dword [esp],edi
call    __thunk_.strlen ; strlen

mov     dword [esp],eax
call    __thunk_.malloc ; malloc
mov     dword [esp+4],edi

lea     ebx, [eax+1]

mov     dword [esp],ebx

call    __thunk_.strcpy ; strcpy
```

```
newloc = (char *) malloc(strlen(s))+1;
strcpy(newloc, s);      /* Buffer Overrun */   /* 2 more... */
```


X86 Platform Specification
Binary Code
Source Code
Scripts
Policy Library
TSL / ISAL
IDA Pro
IDA Pro plug-in
Binary language modules
Binary code analyses
EDG C/C++ front end
EDG C/C++ enhancements
C/C++ language module
C/C++ analyses
CodeSurfer API
Static Intermediate Representations
CodeSurfer / SWYX
CodeSurfer
Initial Weight Generation
C/C++
Intel x86
others
Solver
Symbolic Intermediate Representations
Web Server
Bug & Vulnerability Database
CodeSonar / SWYX
Web Client UI
Users

GRAMMATECH

Language-Independent Parameter Passing

	Caller		Callee	
	Code	Modeled as	Code	Modeled as

C	z = F(x, y);
	
		$param1 = x;
		$param2 = y;
		F();
		z = F$return;
			int F(int arg1,
			int arg2)
			{ …;
			return e;
			}
				void F()
				{
				int arg1 = $param1;
				int arg2 = $param2;
				…;
				F$return = e;
				}

x86	push y
	push x
	call F
	add esp, 8
	mov z, eax
		push y
		push x
		mov $param2,[esp+4]
		mov $param1,[esp+0]
		call F
		mov eax,F$return
		add esp, 8
		mov z, eax
			F: <prologue>
			…
			ret
				F: <prologue>
				mov [esp+8], $param2
				mov [esp+4], $param1
				…
				mov F$return, eax
				ret

Preliminary Results and Expectations

- Starting to apply to realistic open-source C applications
- Typical size of samples
  - Source: 51 C files; ~23.5K (non-blank) lines of code
  - Binary: ~1MB executable; 295K text section (code); ~105K instructs.
- Typical analysis time: 1 coffee break
- Binary analysis time: 2x-3x source analysis time
- Very preliminary results
  - Significant overlap on some bug classes
    - E.g., API misuse; null-pointer dereferences; heap and alloca overruns
  - Significant *potential* overlap on other bug classes; just a SMOP
    - E.g., file race conditions (require additional string/global handling)
  - Others, feasibility TBD in absence of type information
    - E.g., stack-based buffer overruns; heuristics need for buffer size

Summary

- CodeSonar will support light-weight bug finding in both source and binary code.
- Common logic, solver, and policy library.
- Expect to answer (in detail) questions such as
  - What are the relative strengths and weaknesses of source code analysis and binary code analysis?
  - Can end users find bugs in applications as effectively as developers?

Discussion