



Multi-platform Program Analysis

DHS SBIR Phase II

Presented by:

Paul Anderson

October 10, 2012

GrammaTech, Inc.
531 Esty St.
Ithaca, NY 14850
Tel. 607-273-7340
E-mail: info@grammatech.com

Agenda

- About GrammaTech
- Key Takeaways from SBIR Phase II
- Platform-sensitive Analysis
- Phase II Approach and Execution

GammaTech in a Nutshell

Core Business and Competency

- › Software analysis and manipulation

Research, Prototypes, Products, and Customers

- › GammaTech Research
- › CodeSonar®, CodeSurfer®

Prototype Technologies and S&T Projects

- › Mix of SBIRs, BAAs, sole source, and commercial contracts

Leadership

- › Tim Teitelbaum, CEO (Professor Emeritus at Cornell U.)
- › Thomas Reps, President (Professor at U. Wisconsin)
- › Four VPs and a Controller; (21, 10, 10, 2.5, 10) years with company, respectively

Staff and Facilities

- › Size: ~47 (+4 interns) and growing
- › Expertise: ~¾ technical; ~¼ business and administrative
- › PhDs working in program analysis: 15
- › Office building: 3 floor, 15,000 sq.ft.

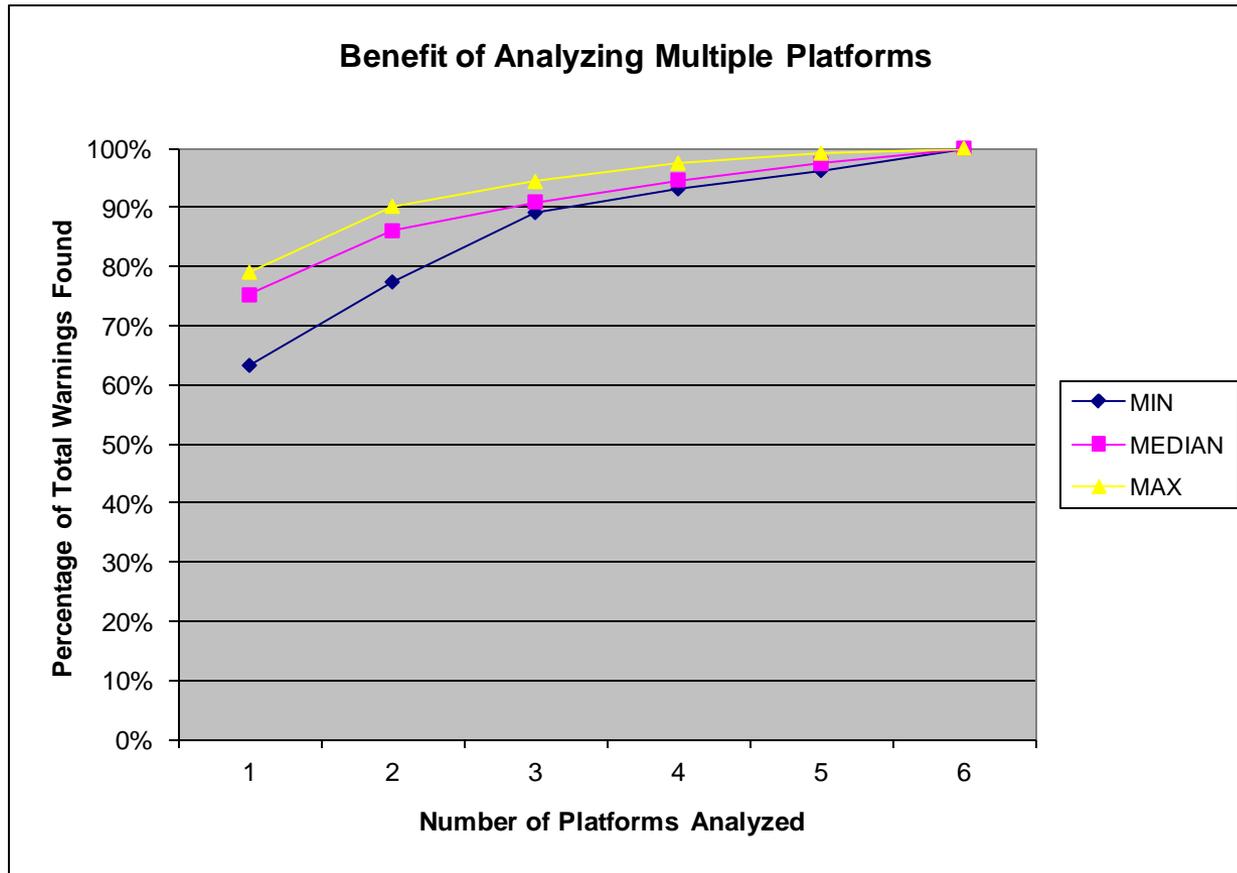
Financials

- › Founded in 1988; profitable essentially every year
- › 18% annual growth for past 10 years
- › Ownership 100% internal; no debt

Key Takeaways

- Platform-specific bugs are especially insidious
- Existing methods for finding such bugs are weak
- Extend existing methods for finding bugs
 - › Static analysis for source and binaries
 - › Test-case generation
- Use continuous build-and-test technology to harness the cloud
 - › Centralize collation and presentation of results
- Leverage existing channels to achieve maximum impact on industry and government

Benefits of Multi-Platform Analysis



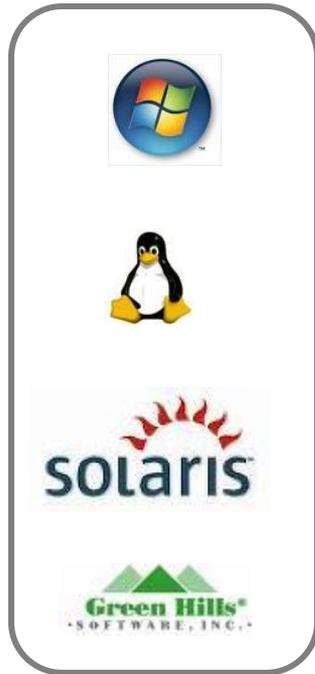
Analyzing on a single platform misses 25% to 40% of error reports

Data: 87 benchmarks, 6 platforms

Copyright © 2012, GrammaTech Inc.

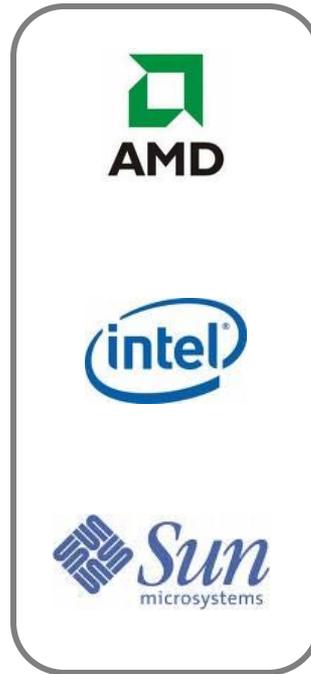
What is a Platform?

Operating System + CPU + Compiler + Libraries



X

multiple
distros



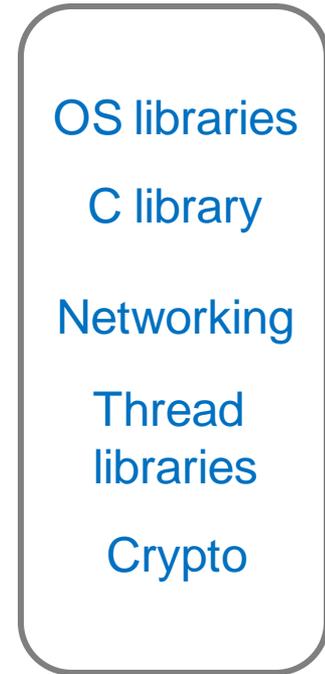
X

32-bit
64-bit



X

compiler
flags



X

multiple
versions

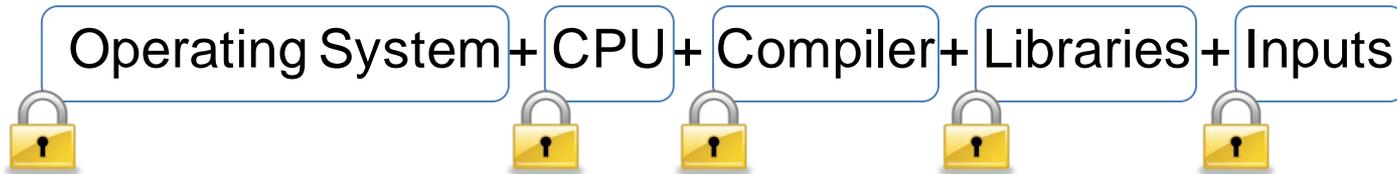
Platform-specific Bugs: Taxonomy of causes

- OS library functions behave differently
 - › Especially wrt response to failure modes
- CPUs may have different behaviors
 - › Endianness, overflow handling
- Conditional compilation
 - › `#ifdef WIN32`
- Differences in compilers and their use
 - › Interpretation of language specification
 - › Code generation, including optimization
 - › Flags passed to each invocation
- Differences in core types
 - › Is **char** signed or unsigned?
 - › Standard library structured types may have different fields and sizes

Traditional bug-finding techniques

Testing:

For each test case:

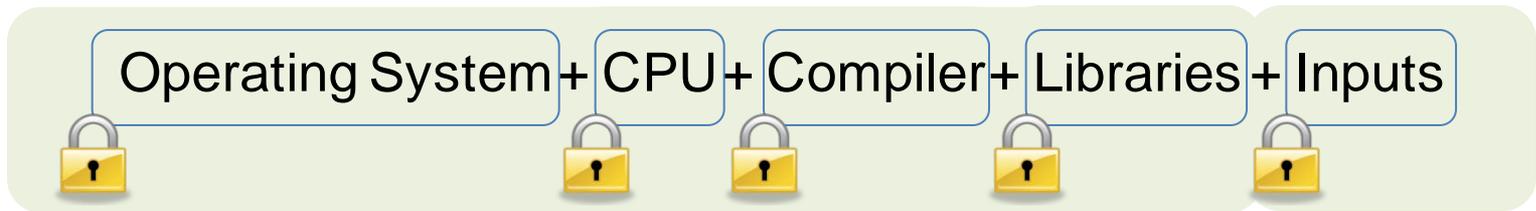


Static Analysis:

For each scan:



Vision



- Extend static-analysis technology to account for differences in Libraries
- Use binary-analysis technology to address CPU and compiler differences
- Use Concolic Execution to automatically generate Inputs
- Use automated build-and-test technology to multiply platform coverage
- Collect and organize results in a central location

Phase II Approach

- Improve static analysis to refine ability to find platform-specific bugs
- Integrate with other distributed multi-platform build-and-test systems
- Use binary analysis to further refine detection of platform-specific bugs
- Leverage results from other projects on concolic execution late in the project

Phase II Execution

■ Accomplishments

- › Engine re-architected to support distributed analyses
 - Successfully working for multiple machines that share a file system
- › UI changed to facilitate showing platform-specific results
 - E.g.: “Show me bugs that show up on 64-bit, but not 32-bit platforms”

■ Plans for remainder of contract

- › Integration with distributed build systems
 - Metronome, Electric Accelerator
- › Platform sensitivity added to library models

The End

- Questions?