Secure Programming with Static Analysis

Jacob West
jacob@fortify.com

OWASP-Day II
Università "La Sapienza", Roma
31st, March 2008

The OWASP Foundation
http://www.owasp.org

Software Systems that are
- Ubiquitous
- Connected
- Dependable

Complexity

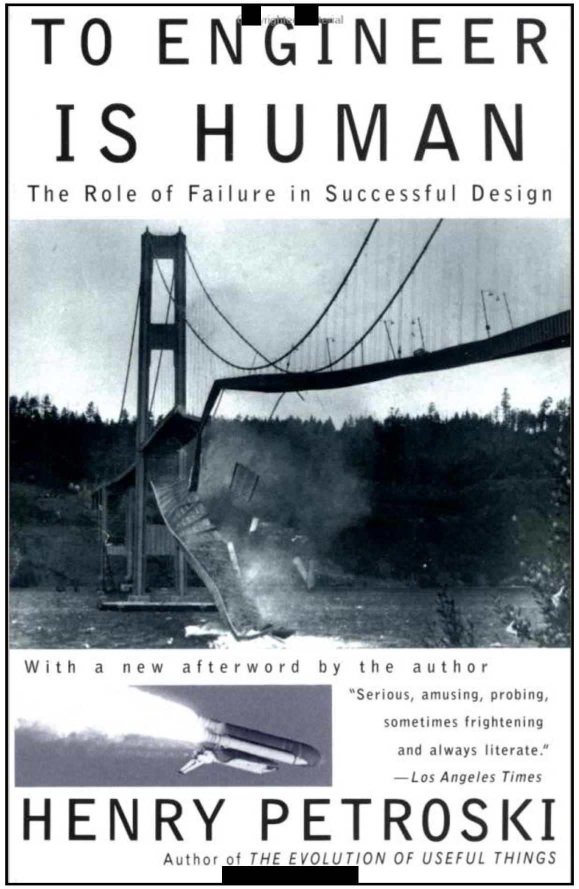
Unforeseen Consequences

Software Security Today

- The line between secure/insecure is often subtle
  - Many seemingly non-security decisions affect security
- Small problems can hurt a lot
- Smart people make dumb mistakes
  - As a group, programmers tend to make the same security mistakes over and over
- We need non-experts to get security right

Success is foreseeing failure.

– Henry Petroski


TO ENGINEER IS HUMAN
The Role of Failure in Successful Design
With a new afterword by the author
"Serious, amusing, probing, sometimes frightening and always literate."
—Los Angeles Times
HENRY PETROSKI
Author of THE EVOLUTION OF USEFUL THINGS

Reliving Past Mistakes

- Cross-site scripting looks more and more like buffer overflow

Buffer Overflow	Cross-site Scripting
Allows arbitrary code execution	Allows arbitrary code execution
Easy mistake to make in C/C++	Easy mistake to make
Exploit is hard to write	**Exploit is easy to write**
Well known problem for decades	Well known problem for a decade

Wrong Answers

Try Harder

- Our people are smart and work hard.
- Just tell them to stop making mistakes.

- Not everyone is going to be a security expert.
- Getting security right requires feedback.

Fix It Later

- Code as usual.
- Build a better firewall (app firewall, intrusion detection, etc.)

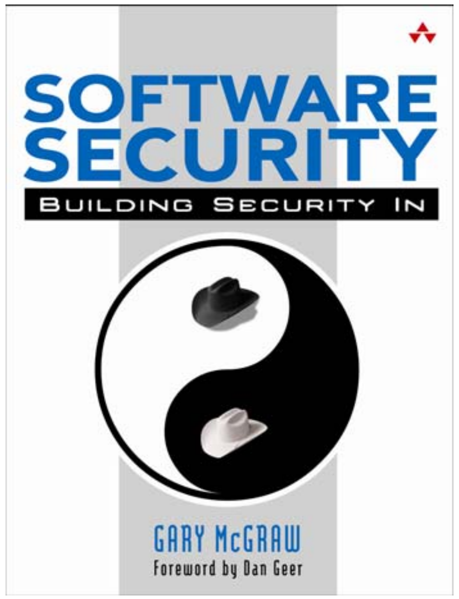
- More walls don't help when the software is meant to communicate.
- Security team can't keep up.

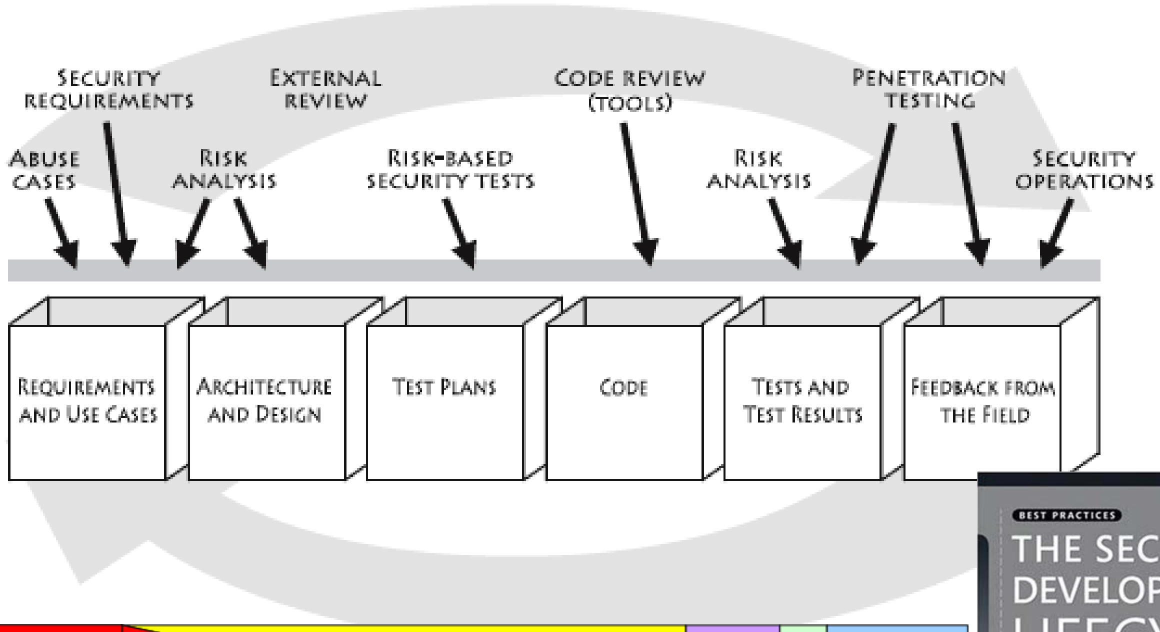
Test Your Way Out

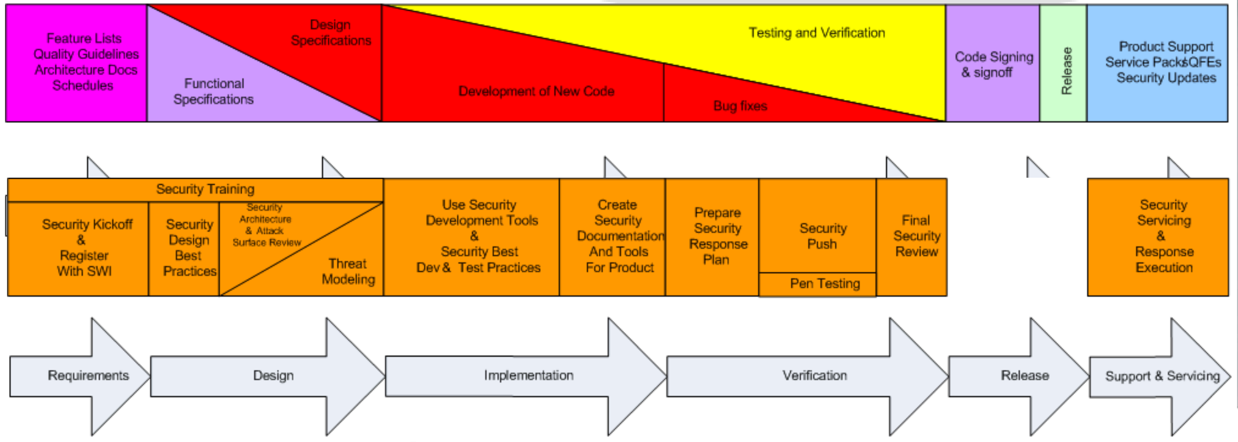
- Do a penetration test on the final version.
- Scramble to patch findings.

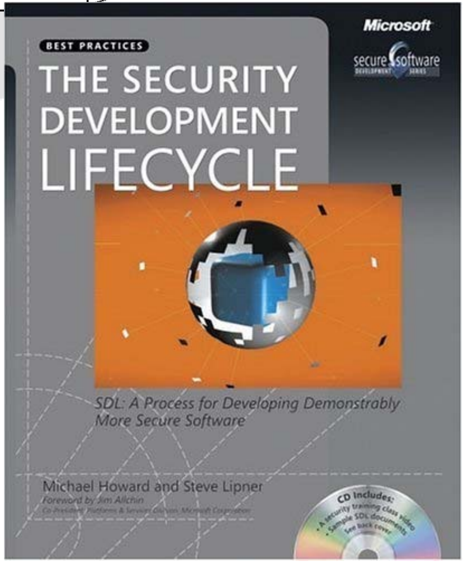
- Pen testing is good for demonstrating the problem.
- Doesn't work for the same reason you can't test quality in.

Security in the Development Lifecycle

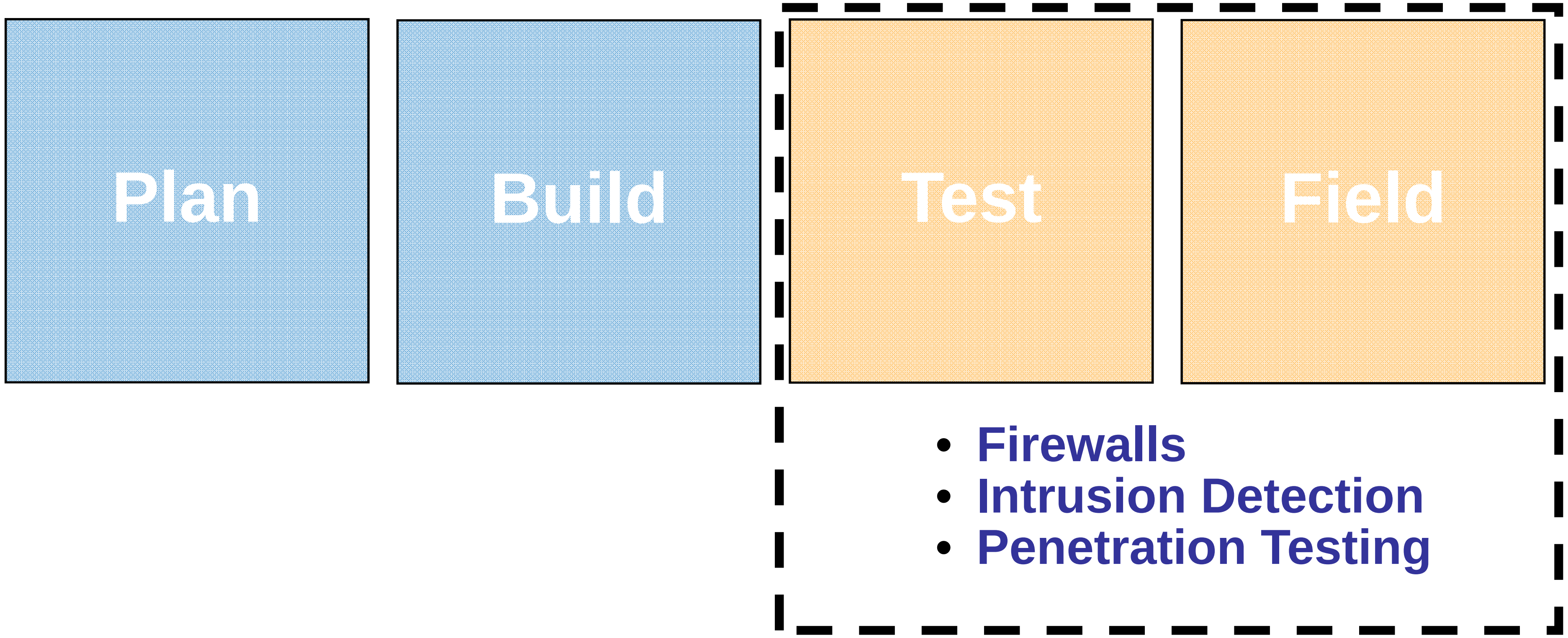






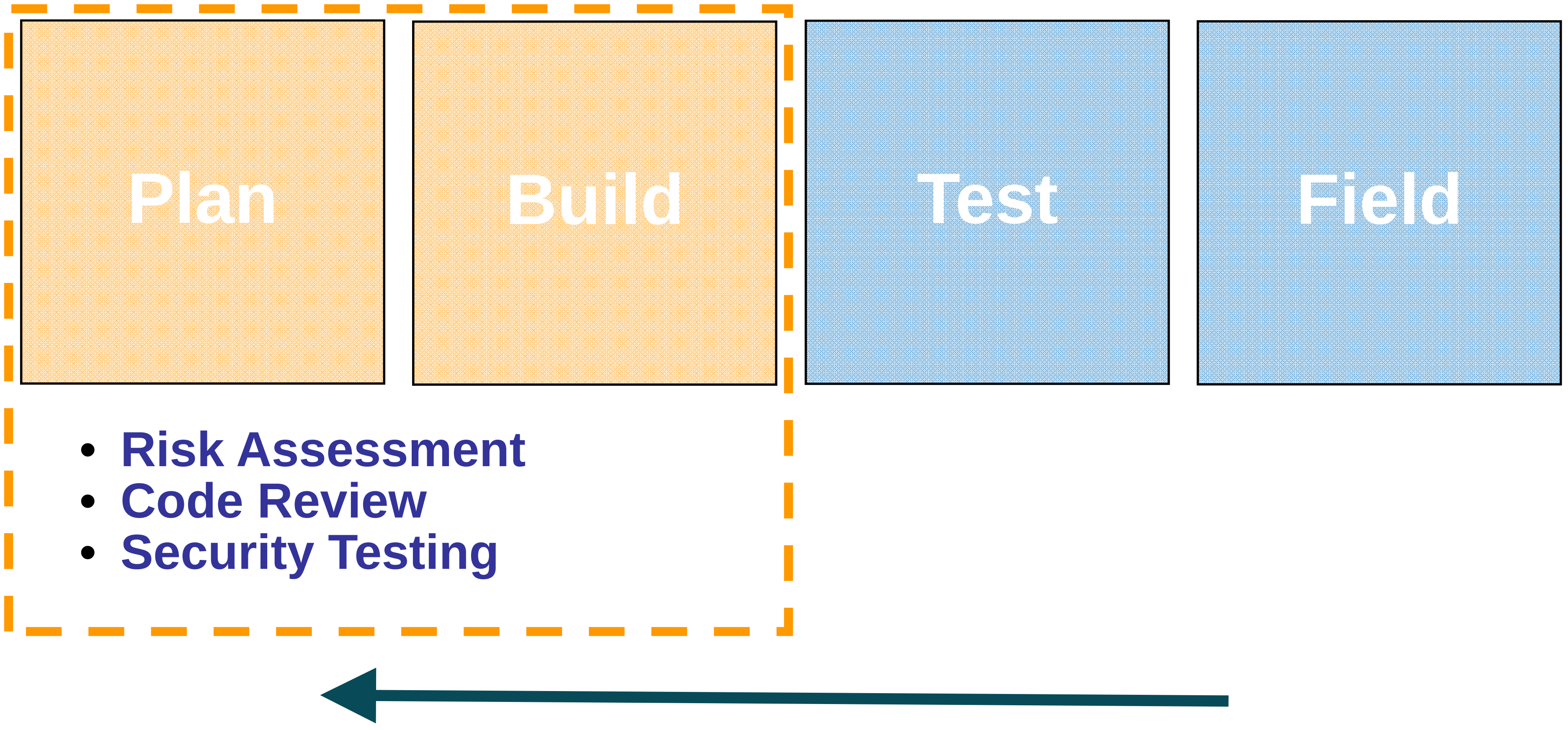

Microsoft
secure software
BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner
CD includes

Security in the Development Lifecycle



- Firewalls
- Intrusion Detection
- Penetration Testing

Security in the Development Lifecycle



- Risk Assessment
- Code Review
- Security Testing

Effective security from non-experts

Static Analysis: The Big Picture

Static Analysis Defined

- Analyze code without executing it
- Consider many more possibilities than you could execute with conventional testing
- Doesn’t know what your code is supposed to do
- Must be told what to look for

Husqvarna
137

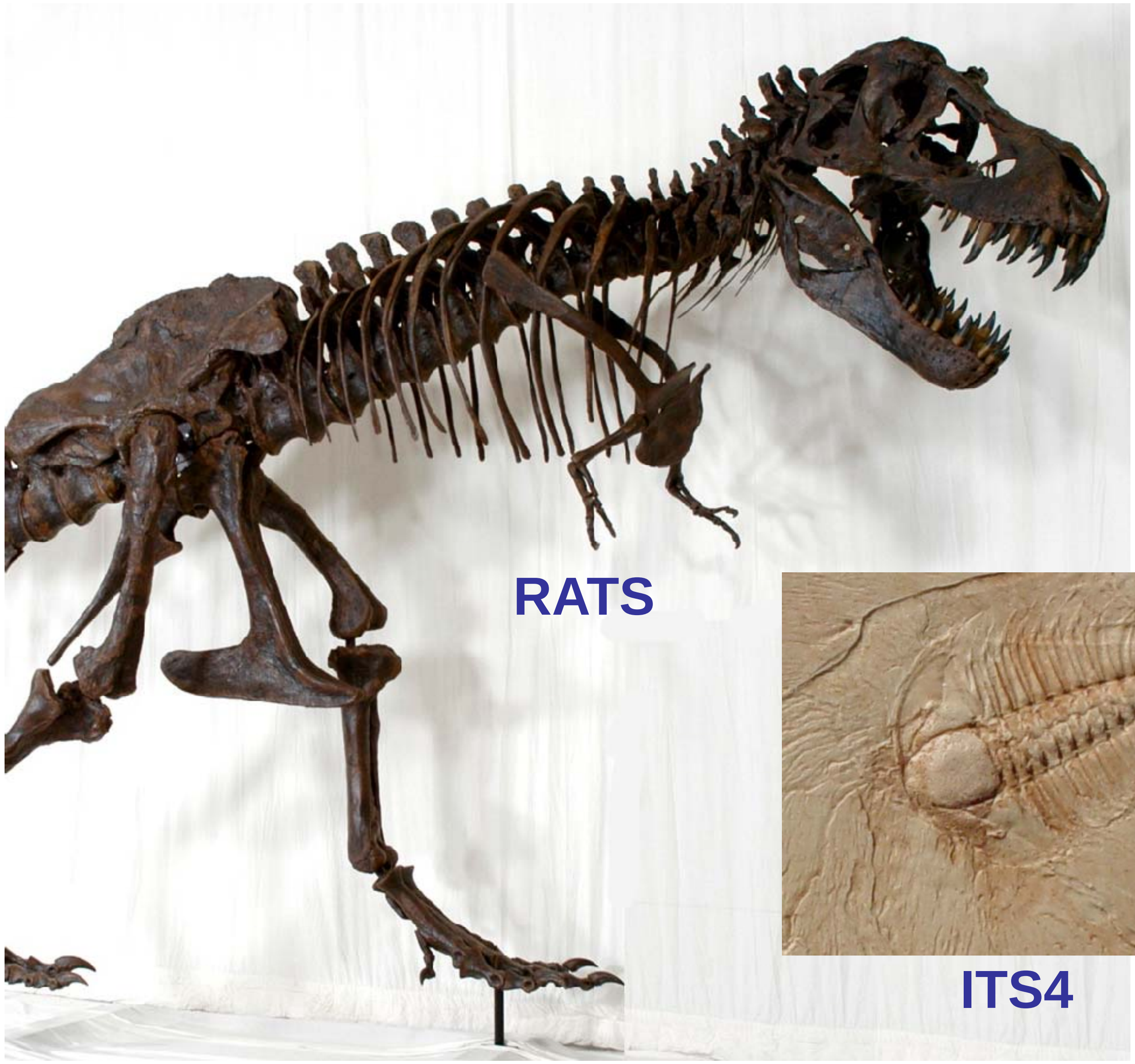
The Many Faces of Static Analysis

- Type checking
- Style checking
- Program understanding
- Program verification / Property checking
- Bug finding
- Security review

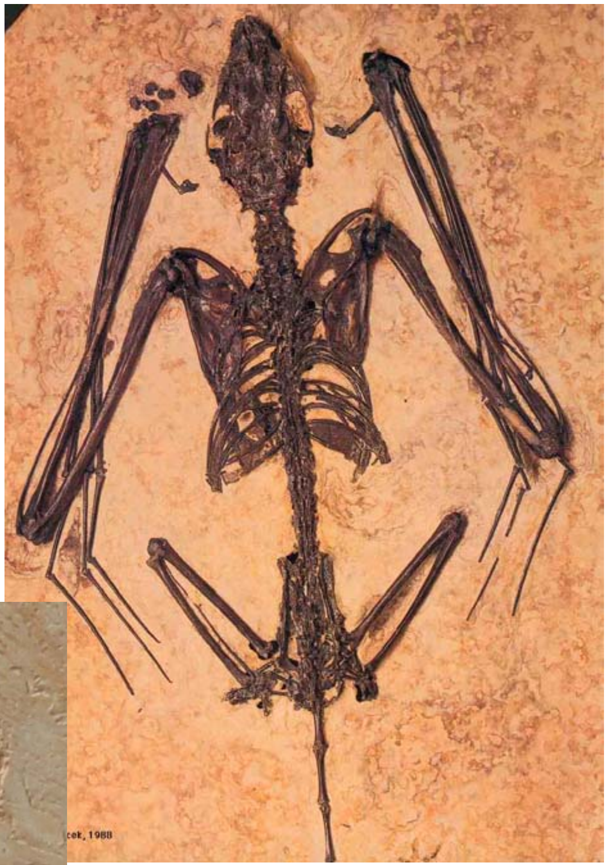
Why Static Analysis is Good for Security

- Fast compared to manual code review
- Fast compared to testing
- Complete, consistent coverage
- Brings security knowledge with it
- Makes review process easier for non-experts

Prehistoric Static Analysis Tools


RATS

ITS4


Flawfinder

Prehistoric Static Analysis Tools

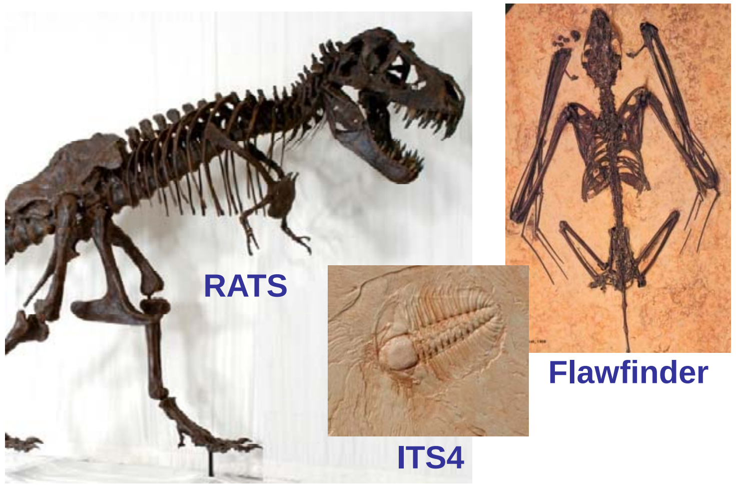
Glorified grep

(+) Good

- Help security experts audit code
- A place to collect info about bad coding practices

(-) Bad

- NOT BUG FINDERS
- Not helpful without security expertise



Advanced Static Analysis Tools: Prioritization

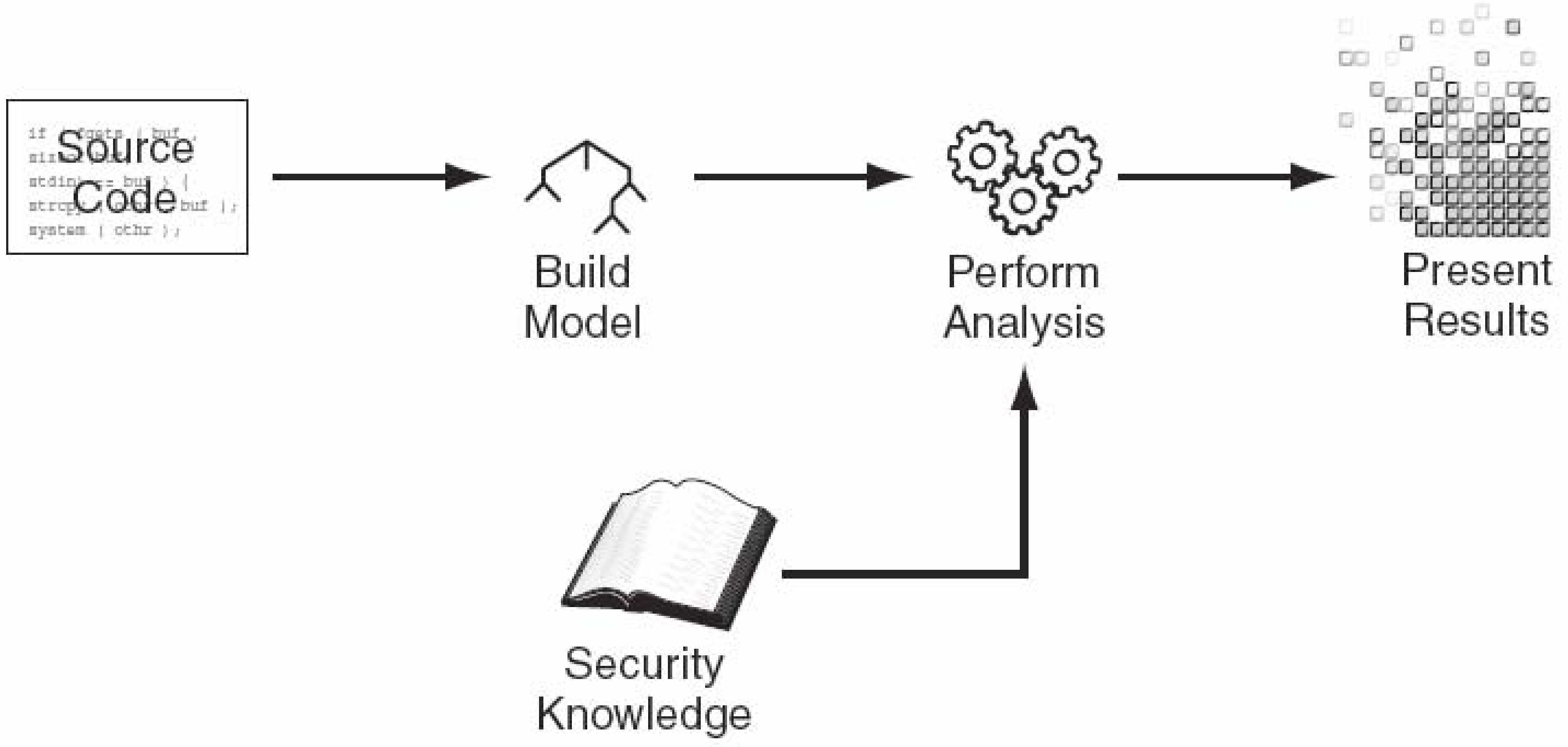
```
int main(int argc, char* argv[]) {
    char buf1[1024];
    char buf2[1024];
    char* shortString = "a short string";
    strcpy(buf1, shortString); /* eh. */
    strcpy(buf2, argv[0]);     /* !!! */
    ...
}
```

What You Won't Find

- Architecture errors
  - Microscope vs. telescope
- Bugs you're not looking for
  - Bug categories must be predefined
- System administration mistakes
- User mistakes

Inside a Static Analysis Tool

Under the Hood



Critical Attributes

- Language support
  - Understands the relevant languages/dialects
- Analysis algorithms
  - Uses the right techniques to find and prioritize issues
- Capacity
  - Able to gulp down millions of lines of code
- Rule set
  - Modeling rules, security properties
- Results management
  - Allow human to review results
  - Prioritization of issues
  - Control over what to report

Only Two Ways to Go Wrong

- False positives
  - Incomplete/inaccurate model
  - Missing rules
  - Conservative analysis
- False negatives
  - Incomplete/inaccurate model
  - Missing rules
  - Forgiving analysis

Static Analysis in Practice

Two Ways to Use the Tools

- Analyze completed programs
  - Fancy penetration test. Bleah.
  - Results can be overwhelming
  - Most people have to start here
  - Good motivator

- Analyze as you write code
  - Run as part of build
  - Nightly/weekly/milestone
  - Fix as you go

Adopting a Static Analysis Tool

1) Some culture change required
   - More than just another tool
   - Often carries the banner for software security program
   - Pitfall: the tool doesn’t solve the problem by itself

2) Define the playing field
   - Choose specific objectives
   - Build a gate

3) Teach up front
   - Software security education is paramount
   - Tool training is helpful too

Adopting a Static Analysis Tool

4) Start small

- Do a pilot rollout to a friendly dev group
- Build on your success

5) Go for the throat

- Tools detect lots of stuff. **Turn most of it off.**
- Focus on easy-to-understand, highly relevant problems.

6) Appoint a champion

- Make sure there is a point person on the dev team
- Choose a developer who knows a little about everything

Adopting a Static Analysis Tool

7) Measure the outcome

- Keep track of tool findings
- Keep track of outcome (issues fixed)

8) Make it your own

- Investigate customization
- Map tool against internal security standards.
- Best case scenario is cyclic:
  - The tool reinforces coding guidelines
  - Coding guidelines are written with automated checking in mind

9) The first time around is the worst

- Budget 2x typical cycle cost
- Typical numbers: 10% of time for security, 20% for the first time

What Next?

Seven Pernicious Kingdoms

- Catalog, define, and categorize common mistakes:

  http://www.fortify.com/vulncat

- Input validation and representation
- API abuse
- Security features
- Time and state
- Error handling
- Code quality
- Encapsulation

* Environment

Security Testing

- Popular security testing tools focus on controllability
  - Fuzzing (random input)
  - Shooting dirty data (input that often causes trouble)
- A different take: improve observability
  - Instrument code
  - Observe behavior during QA
- Benefits
  - Brings security to QA
  - Vastly improved error reporting
  - Security-oriented code coverage
- Uses rule set from static analysis tool!

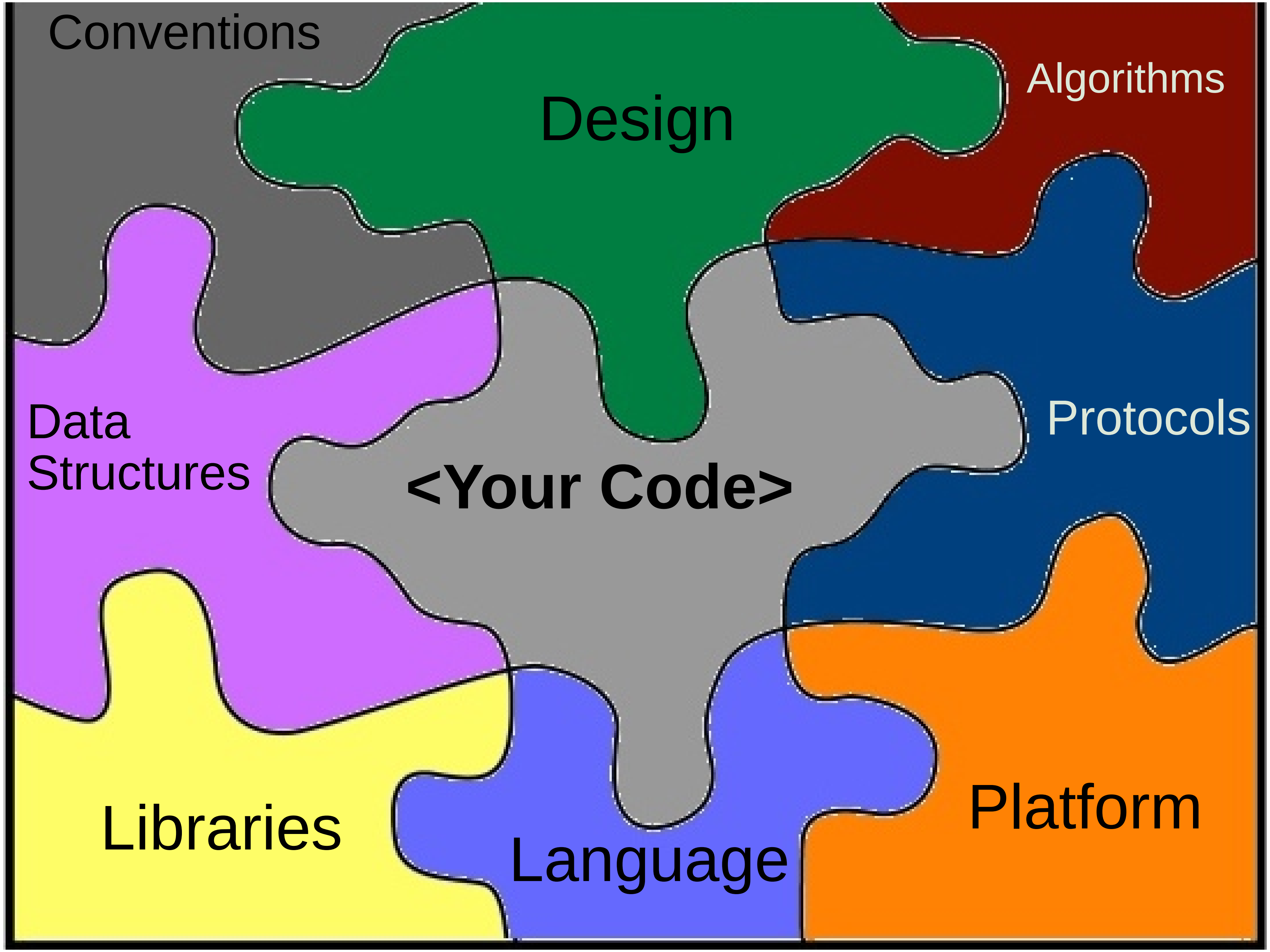
Protecting Programs at Runtime

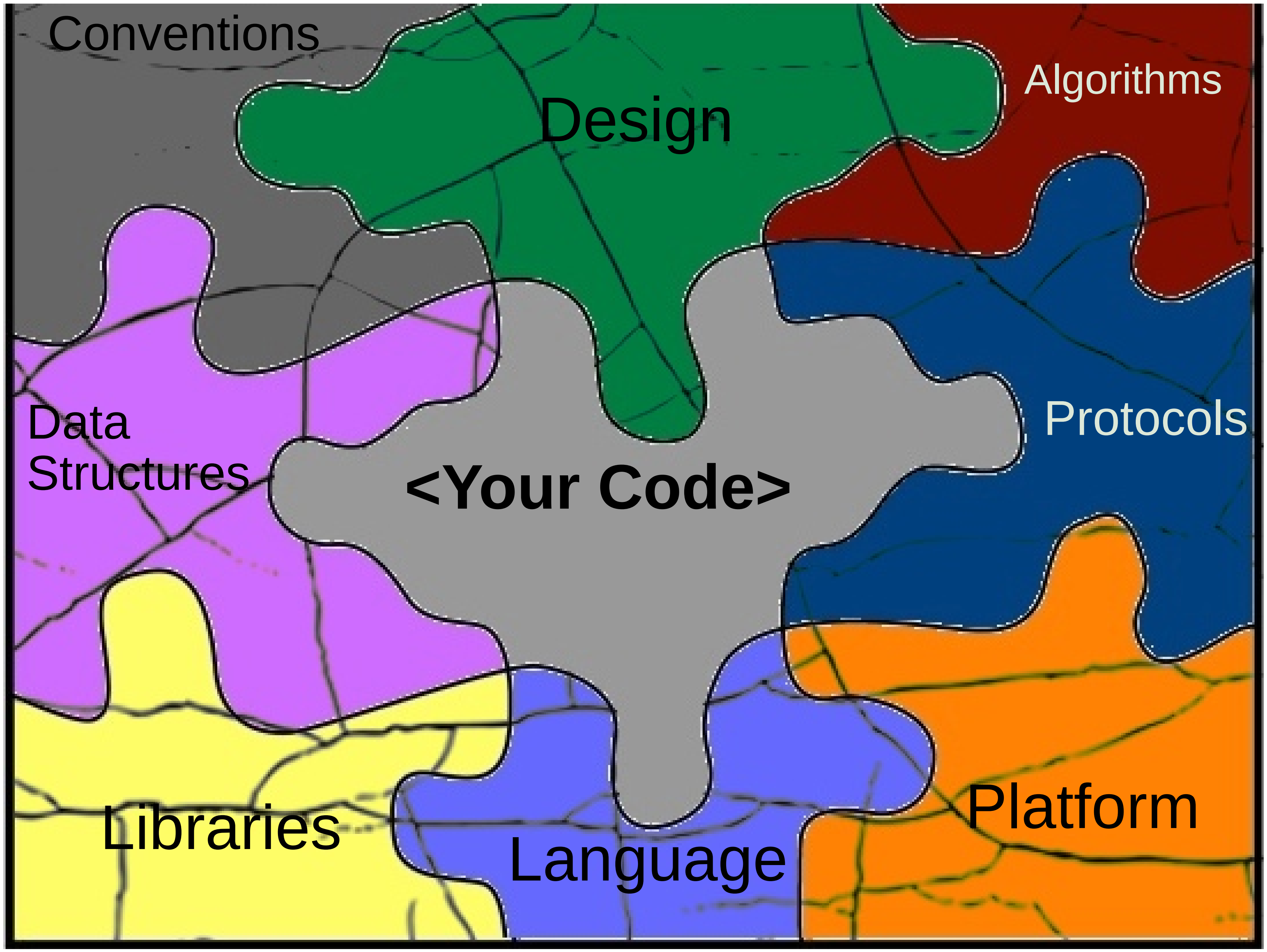
- If you can find bugs, why not fix them
  - Instrument program
  - Watch it run in production
- More context than external systems
- Flexible response: log, block, etc
- Low performance overhead is a must
- Potential to detect misuse in addition to bugs

Fortify University Program

- Universities granted free academic license
- Fortify helps professors develop course material
- Fortify provides guest lecturers on software security and static analysis

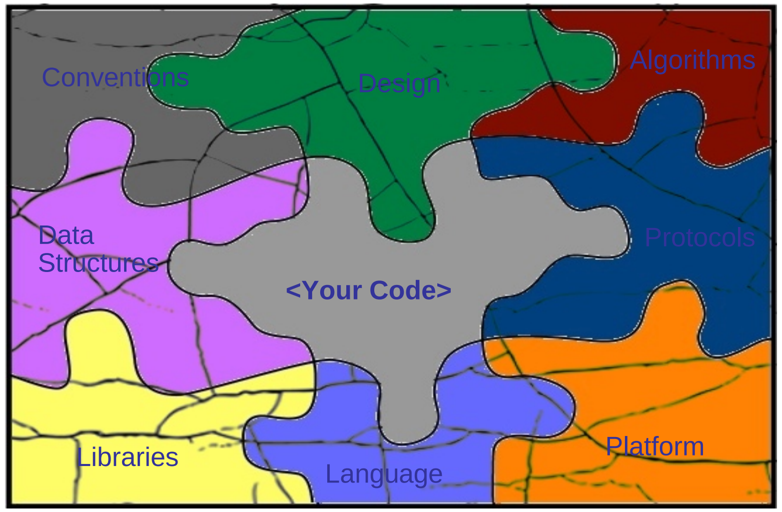
Parting Thoughts


Conventions
Design
Algorithms
Data Structures
<Your Code>
Protocols
Libraries
Language
Platform


Conventions
Design
Algorithms
Data Structures
<Your Code>
Protocols
Libraries
Language
Platform

The Buck Stops With Your Code

- **Security problems everywhere you look**
  - Languages, libraries, platforms, etc.
- **Right answer**
  - Better languages, libraries, frameworks, etc.
- **Realistic answer**
  - Build secure programs out of insecure pieces



Summary

- Mistakes happen. Plan for them.
- Security is now part of programming
- For code auditors: tools make code review efficient
- For programmers: tools bring security expertise
- Critical components of a good tool
  - Algorithm
  - Rules
  - Interface
  - Adoption Plan

Jacob West
jacob@fortify.com

Brian Chess
brian@fortify.com