

SECURITY

IN THE GRANDER ORDER OF THINGS

Bertrand Meyer

Professor of Software Engineering, ETH Zürich

Also: Eiffel Software (Santa Barbara, CA)

+ Monash University (Melbourne, Aust.)

© Bertrand Meyer, 2003



The stakes are higher



EPIGRAPH TO ABELSON & SUSSMAN

“I think that it’s extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don’t think we are. I think we were responsible for stretching them, setting them off in new directions, and keeping fun in the house.”

Alan Perlis

Epigraph to *The structure and Interpretation of Computer programs*, MIT Press



CARCASSONNE, 13TH CENTURY





CHAMBORD, 16TH CENTURY





VERSAILLES, 17TH-18TH CENTURY





“We’re not in Kansas any more”

**Example: wireless connections
“Warchalking”**



EPIGRAPH TO ABELSON & SUSSMAN

“I think that it’s extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don’t think we are. I think were responsible for stretching them, setting them off in new directions, and keeping fun in the house.”

Alan Perlis

Epigraph to *The structure and Interpretation of Computer programs*, MIT Press



The usual suspects:

- **Microsoft, other software makers**
 - **Incompetent programmers**
 - **Irresponsible managers, marketers**
- **System administrators**
- **Naïve users**



... The attackers?



**Frank Hayes, ComputerWorld editor, 6 August 2001,
about Code Red:**

“There's not a competent programmer on the face of the planet who can't write a buffer that won't overflow. That code shouldn't have made it through its first code review. Or past the programmer's Enter key.

There's no excuse for it. As Microsoft's chief software architect, Bill Gates should be personally ashamed.”



The stakes are higher



WHO IS AT FAULT?

Decision makers in client companies

Journalists

“The marketplace”

Prevalent conformism

“No one gets fired for buying IBM” (?)

My experience: Eiffel (since 1985)

Compare to: Taligent, NeXtSTEP



The set of methods, techniques, tools, languages for producing software that is

- Useful
- Of guaranteed quality
- Amenable to change
- Potentially large and complex

Software “engineering”



Set of quality factors:

- **Correctness**
- **Robustness**
- **Integrity**
- **Extendibility**
- **Efficiency**
- **Ease of use**



Correctness:

Ability to function according to the specification

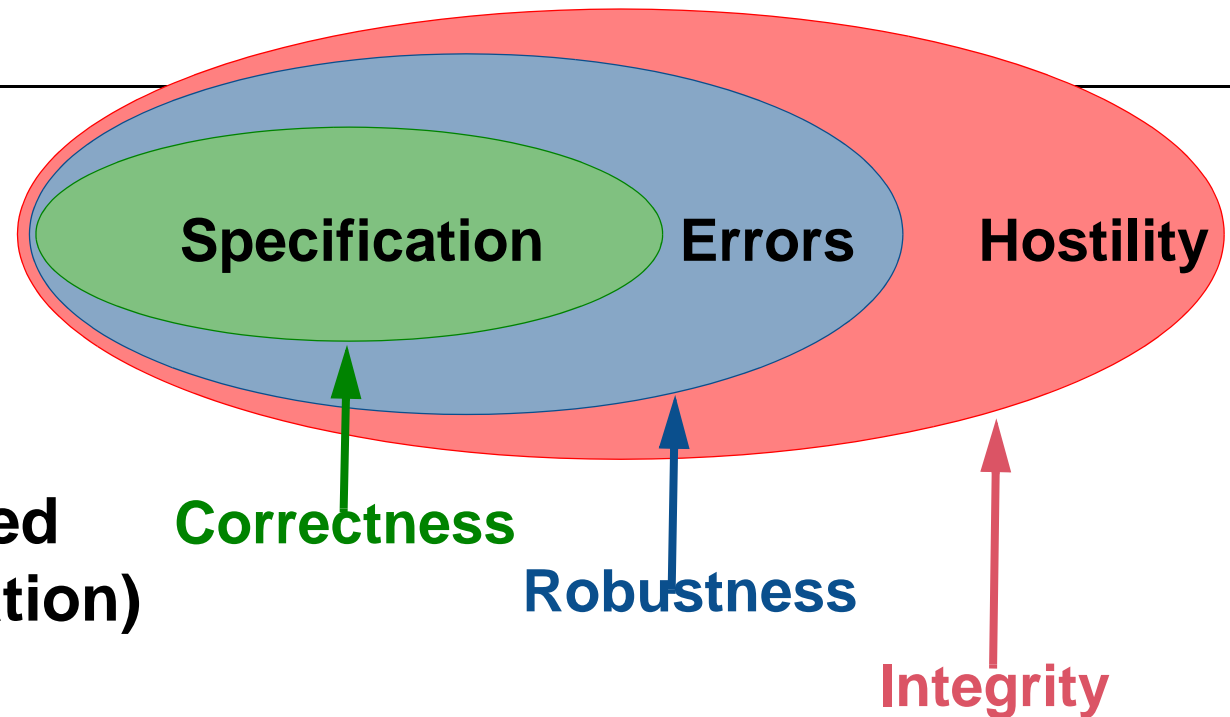
(in cases covered by the specification)

Robustness:

Ability to react “acceptably” to cases not covered by the specification

Integrity:

Ability to prevent damage in cases of hostile misuse





QUALITY IN SOFTWARE: THE BIG SECRET

(no one really cares)



QUALITY IN SOFTWARE: THE BIG SECRET

(no one really cares)

“Why use Design by Contract techniques? All that matters is to have 5-week release cycles. If there are bugs, we’ll fix them during testing”.

A project manager, 2000



Y2K

(See also: the “Windowing” technique)

Ariane 5

Mars Lunar Orbiter

London ambulance system

Overall waste (see Standish reports, 1994-present)

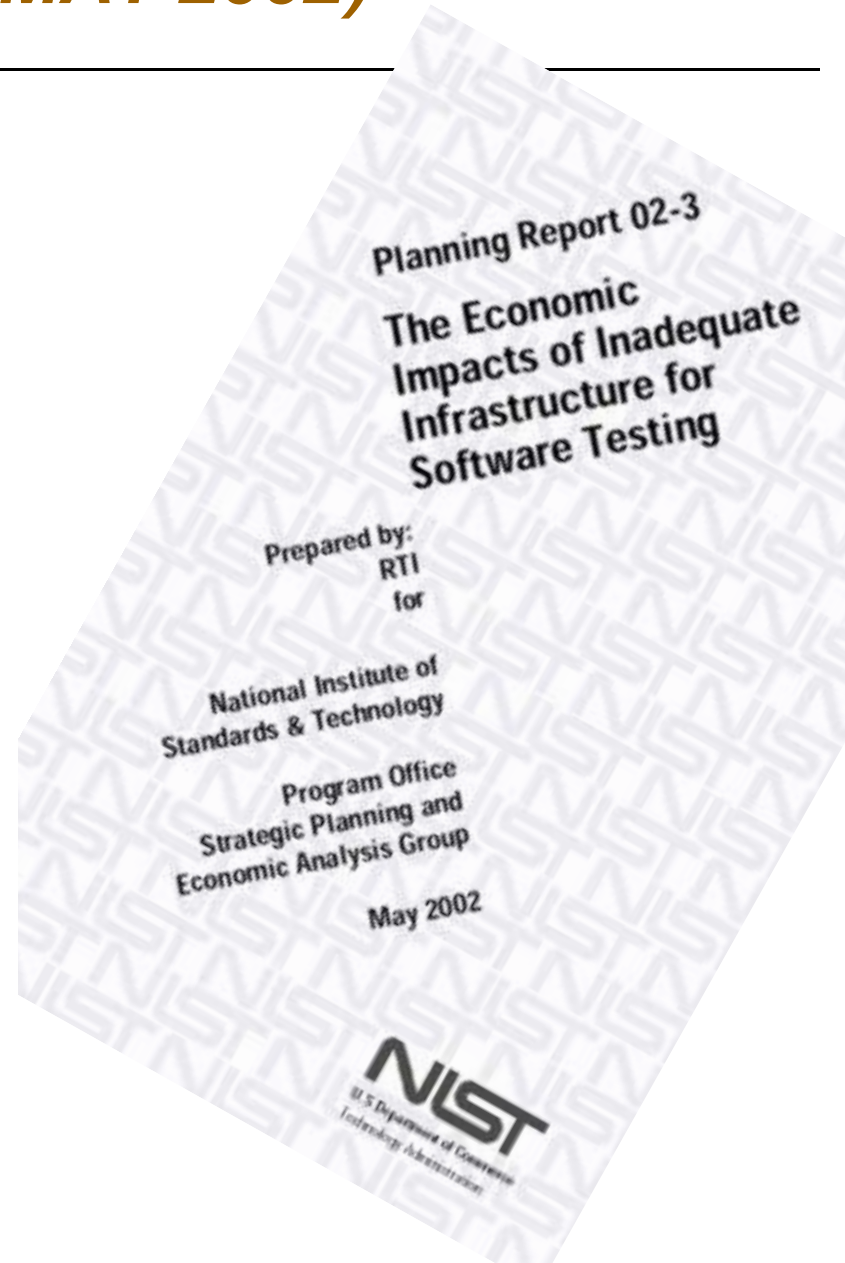


NIST REPORT ON “TESTING (MAY 2002)

**Monetary effect on
developers and
users due to
“insufficient testing infrastructure”:**

\$59.5 billion

**(Financial sector: \$3.3 billion,
auto/aerospace \$1.8 billion etc.)**





EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.



The stakes are higher

The tasks are tougher

The techniques are younger



HOW LONG CAN WE GET AWAY WITH IT?

Warranty disclaimers are subject to the decisions of the courts



QUALITY IN SOFTWARE: THE BIG SECRET

(no one really cares)



QUALITY IN SOFTWARE: THE BIG SECRET

(no one really cares)

(catastrophes change this for a small subset of the industry)



QUALITY IN SOFTWARE: THE BIG SECRET

(no one really cares)

(catastrophes change this for a small subset of the industry)

(security changes this)

Code Red, Sircam, ...

Another example: Mailtool on Solaris



SOFTWARE ENGINEERING HELPS!

Static typing

Design by Contract

Automatic garbage collection

Information hiding

Object technology: abstraction, structure, simplicity

More expressive programming languages

More restrictive programming languages (“bondage and discipline)”)

Reuse, component-based development

Formal methods: software with mathematically proved properties

Management measures: software process standards (CMM, ISO 9001...), metrics, open source...



Initially a reliability rule: prove that

- A variable declared of a certain type in the program text may only, during program execution, denote objects of a compatible type.
- Any operation that the execution applies to an object is one of the operations defined for the object's type.

x: SOME_TYPE

... x.some_operation

Doubles as a security rule: Java, .NET



ARIANE 5, JUNE 1996

**Conversion failed: 64-bit real to 16-bit signed integer
("Horizontal bias")**

Exception was not caught, "could not happen"

Analysis was based on properties of Ariane 4 parameters



integer_bias (b: REAL): INTEGER is

require
representable (b)

do

...

ensure

equivalent (b, Result)

end



syslogd "Some error message"

finger Some_name

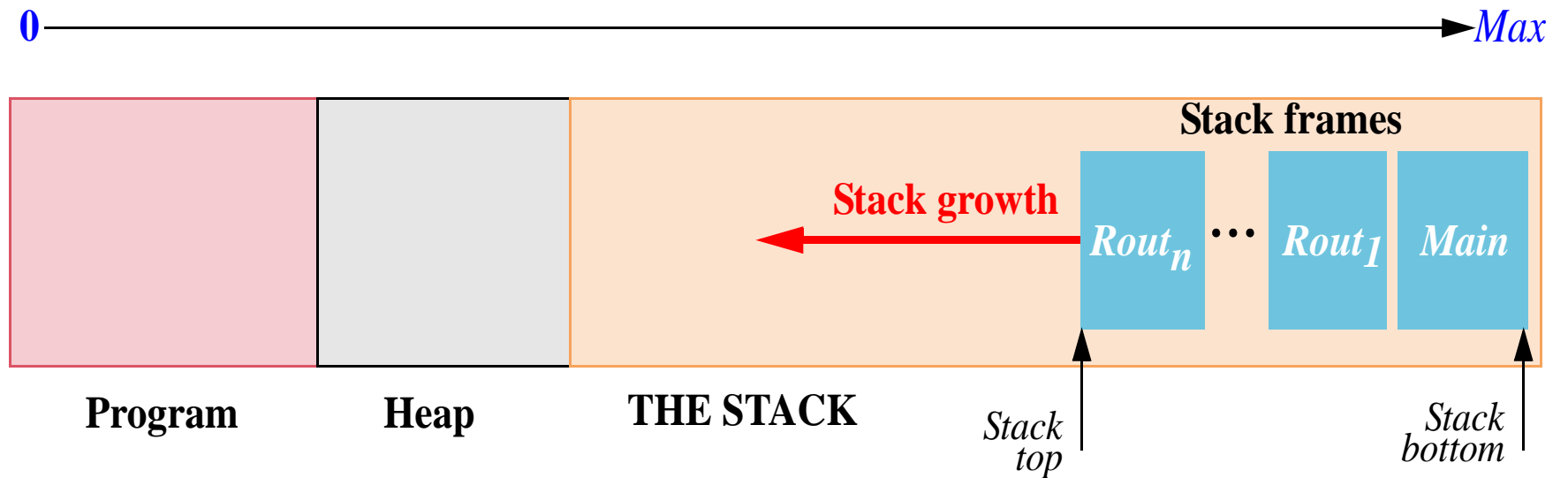
some_command "some text"

(Text input into some browser field)



BUFFER OVERFLOW

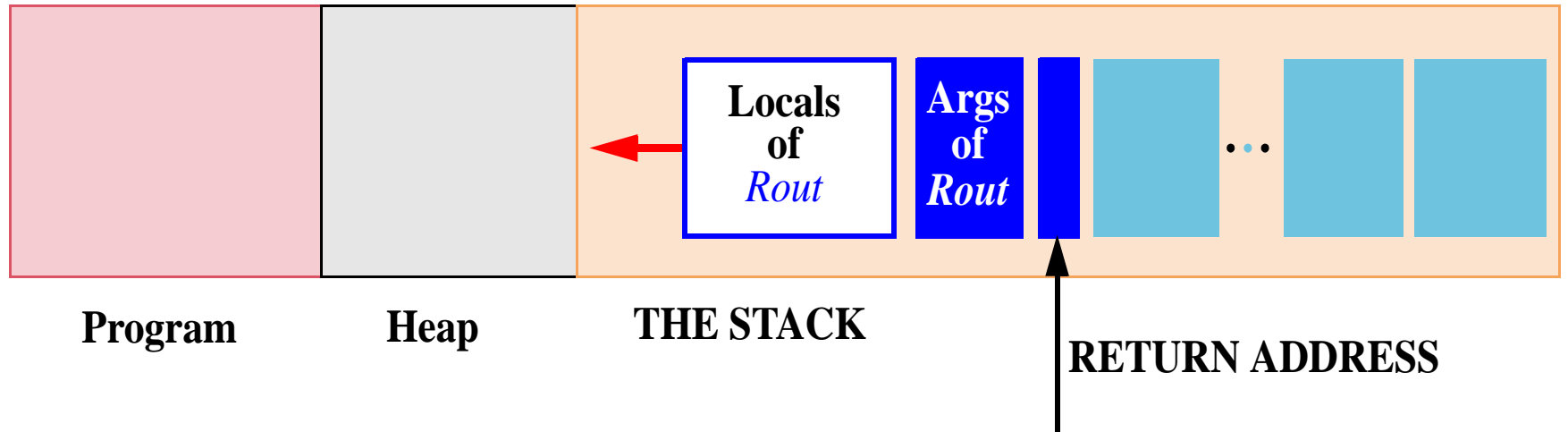
Memory setup:





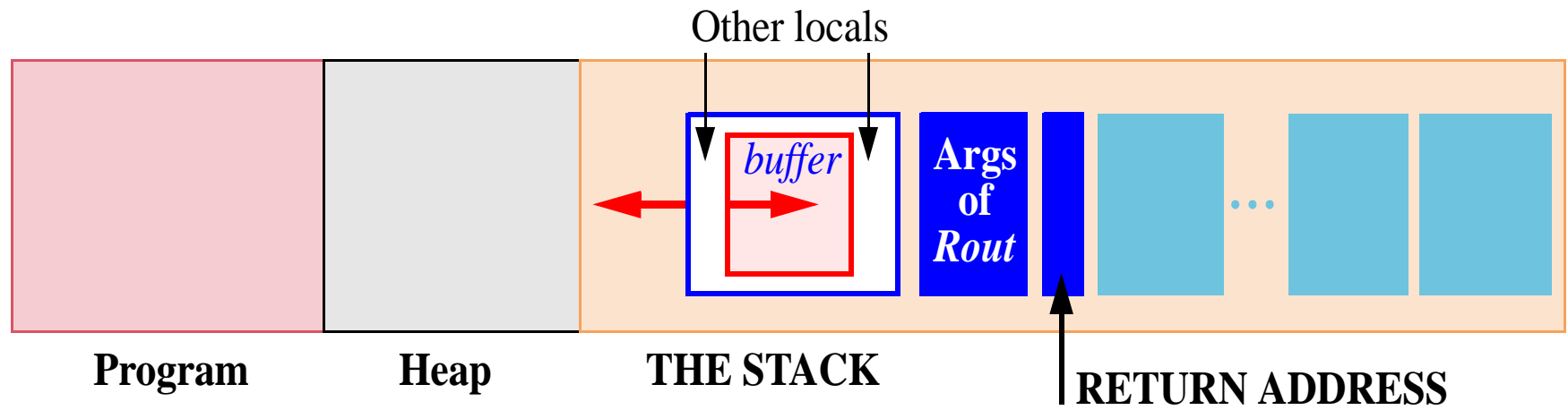
CALLING A ROUTINE

New memory setup:



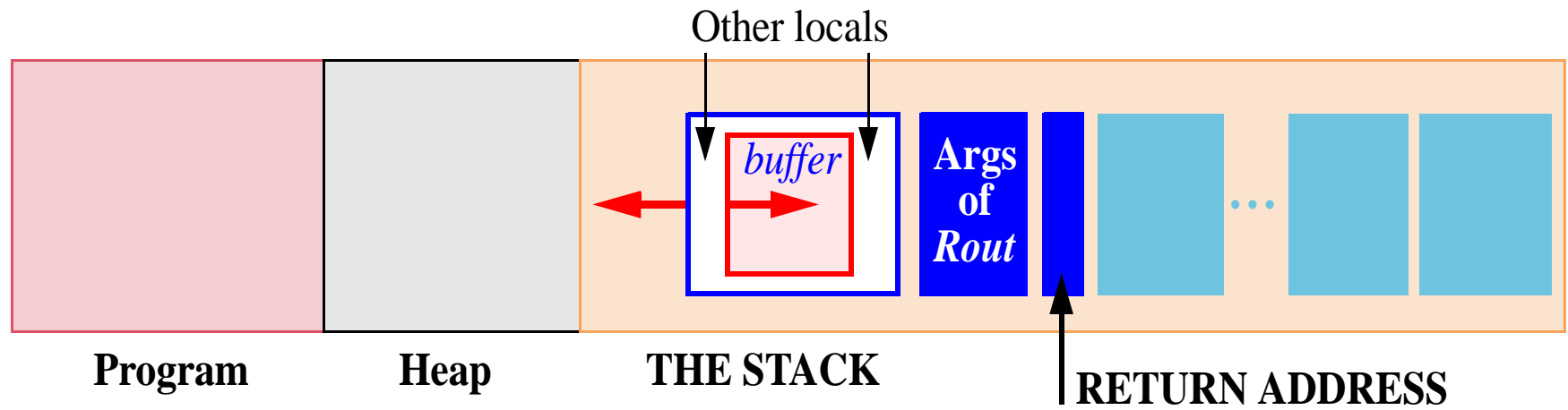


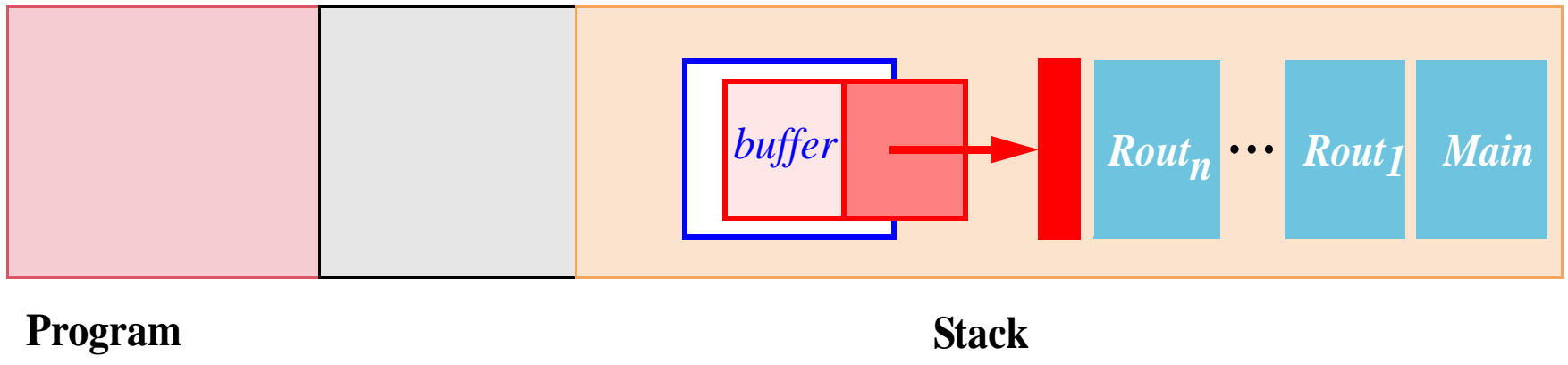
ALLOCATING THE BUFFER





ALLOCATING THE BUFFER



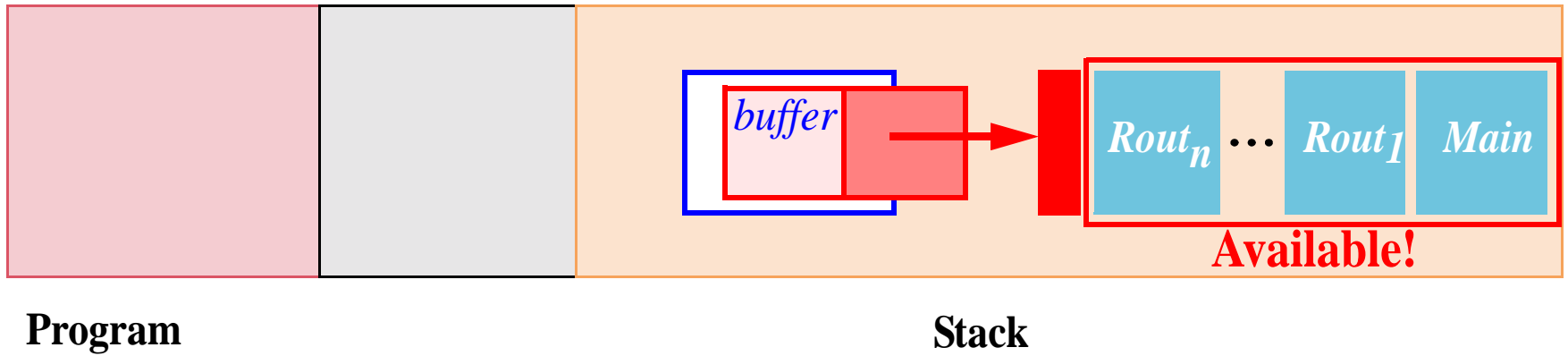


Program

Stack

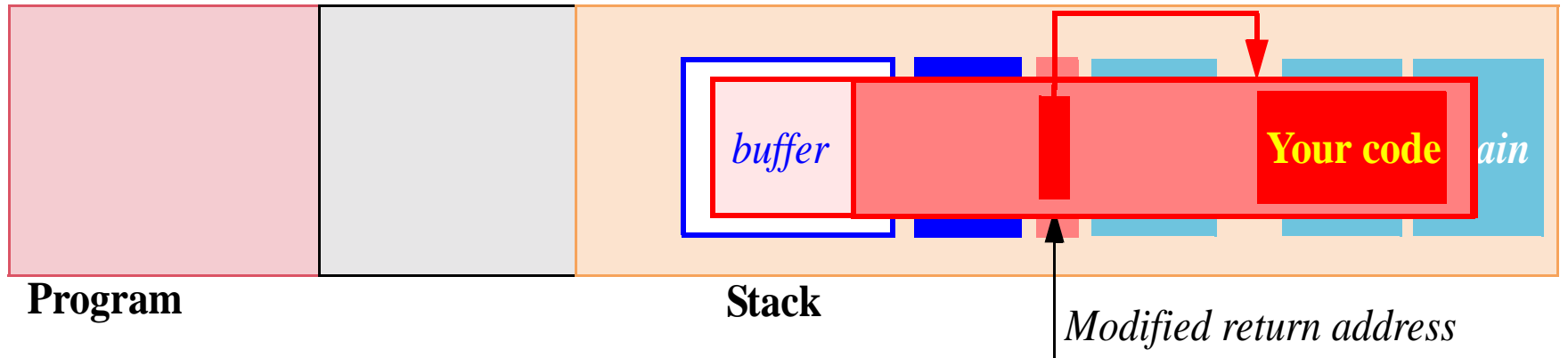


GETTING CLOSER...





INSERTING THE CODE





HOW IS THIS CODED?

```
from i := 1 until i > input_size loop    [1]
  buffer [i] := input [i]
  i := i + 1
end
```



```
from i := 1 until i > input_size loop    [1]
  buffer [i] := input [i]
  i := i + 1
end
```

OR:

```
from i := 1 until
  i > input_size or i > buffer_size
loop
  buffer [i] := input [i]
  i := i + 1
end
```

[2]



If it crashes, is it:

- **Good?**
- **Bad?**



BUFFER OVERFLOW: LESSONS

Security issues, but more fundamentally

- **Lack of specification**
- **Lack of specification enforcement**
- **Programming techniques**
- **Security concepts**

The software engineering view:

- **At the core, it's a programming methodology issue**



```
finger (target: STRING): USER is
  require
    exists: target /= Void
    not_too_big: target.count <= Max_input
  do
    ...
  ensure
    exists: Result /= Void
    conforms:
      (Result /= Unknown) implies
        (Result.user_name = target.name)
  end
```



Symlink racing:

- Many utilities, running with top privileges, create files in common directories, e.g. /tmp
- Everyone has write access to such a directory
- Entry in directory can be symbolic link to user's own file!
- Basic counter-attack: check file is not symbolic link
- Between check and use, anything can happen!

The software engineer's view: this is because of ignorance of basic concurrent programming issues and techniques, especially synchronization



“Given enough eyes, all bugs are shallow”

Eric Raymond



“Given enough eyes, all bugs are shallow”

Eric Raymond

Not true.



“Two buffer overflows recently found in Kerberos version 5 could only be exploited when used in conjunction with each other. ... Security reviews of source code tend to be complex and boring. Even many experts don't like to do them.”

“Consider the open source FTP server wu-ftpd. In the past two years, several very subtle buffer overflow problems have been found in the code. Almost all had been in the code for years, even though it had been examined many times by both hackers and security auditors. In fact, the wu-ftpd has been used as a case study for vulnerability detection techniques. One tool was able to identify one of the problems as potentially exploitable, but researchers examined the code thoroughly and came to the conclusion that there was no way the problem could be exploited. Over a year later, an expert audit finally did turn up the problem.”



“Given enough eyes, all bugs are shallow”

Eric Raymond



SOFTWARE LICENSE: THAT WAS THE GPL

EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.



“Given enough eyes, all bugs are shallow”

Eric Raymond

The software engineering view:

- **This is simply the old idea of code review**
- **It helps, but only moderately**
- **Tools do better**



Prefix (Microsoft)

Steve Lipner, Microsoft Security head, ComputerWorld, 24 September 2001:

Prefix runs a scan of an entire product's source-code base to detect patterns of potential programming errors that experience tells us are likely to be security-related and flags them for human review and correction. Prefix takes a day or two to run across the entire Windows code base. It's run every couple of weeks throughout the [Windows .Net Server] development cycle. It started to be run after Windows 2000 shipped. .Net Server will be the first product that's had a development cycle of benefit from Prefix.



LIPNER, CONTINUED

How successful have you been at rooting out those infamous buffer-overflow vulnerabilities?

We've found and eliminated a lot. That said, it's important to stress that there are an infinite number of ways to run a program. And similarly, there are a vast number of ways that one can write a buffer overflow. [Prefix] is not a closed-form solution.



“The complexity challenge is particularly acute in the most crucial place — security — where potentially hostile programs are continuously arriving across the Internet to run on potentially vulnerable computers.

Bill Gates, the chairman of Microsoft, acknowledges that problem. In an interview... he noted that the computer security aspects of his company’s distributed computing initiative, called .Net, were extremely complicated, systems that even he does not fully understand.”

John Markoff, New York Times, 14 June 2001

The software engineering view: good solutions are based on simple concepts



It is possible to prove realistic software today



For

- **Guaranteed quality**
- **Faster time to market**
- **Ease of maintenance**
- **Standardization of software practices**
- **Preservation of know-how**



The key issue

- **Bad-quality components are major risk**
Deficiencies scale up, too
- **High-quality components could transform the state of the software industry (if it wanted to — currently doesn't)**



Component design should be Formula-1 racing of software “engineering”.

In component development, perfectionism is good.



THE ABCDE OF COMPONENT CERTIFICATION

Acceptance

- A.1 Some reuse attested
- A.2 Producer reputation
- A.3 Published evaluations

Behavior

- B.1 Examples
- B.2 Usage documentation
- B.3 Preconditioned
- B.4 Some postconditions
- B.5 Full postconditions
- B.6 Observable invariants

Constraints

- C.1 Platform spec
- C.2 Ease of use
- C.3 Response time
- C.4 Memory occupation
- C.5 Bandwidth
- C.6 Availability
- C.7 Security

Design

- D.1 Consistent API rules
- D.2 Strict design rules
- D.3 Extensive test cases
- D.4 Some proved properties
- D.5 Proofs of preconditions, postconditions & invariants

Extension

- E.1 Portable across platforms
- E.2 Mechanisms for addition
- E.3 Mechanisms for redefinition
- E.4 User action pluggability



Trusted components: methods, tools, examples

Component proofs

Component Certification Center



Software engineers will, increasingly, have to put integrity at the top of their list of concerns.



The stakes are higher

Security issues are part of quality issues.

Quality is tough. No magical solution but:

- **Many of the techniques, tools, languages exist**
- **Software engineering is here to help**



“Why use Design by Contract techniques? All that matters is to have 5-week release cycles. If there are bugs, we’ll fix them during testing”.



THE LESSON FOR DECISION-MAKERS

Quality costs

Quality pays



THE LESSON FOR DECISION-MAKERS

Quality costs

Quality pays

If you dismiss quality, quality will dismiss you

These slides:

www.inf.ethz.ch/personal/meyer/down/security-2002.pdf