

# Software Security Introduction

Erik Poll

Digital Security

Radboud University Nijmegen

# Goals of this course

- What is software security ?  
Understanding the role that software plays
  - in providing security
  - as source of insecurity
- Principles, methods & technologies to make software more secure
  - incl. practical experience with some of these
- Typical threats & vulnerabilities that make software less secure, and how to avoid them

# Today

- Organisational stuff
- What is "software security"?
- The problem of software insecurity
- The causes of the problem
- Security concepts
- The solution to the problem?

## Practicalities: prerequisites

- Introductory security course
- Basic programming skills, in particular
  - C(++)
  - Java
  - possibly PHP

## Literature & other resources

- Slides etc. available on-line at  
<http://www.cs.ru.nl/~erikpoll/ss>  
But please do register in blackboard
- No book, but **obligatory literature** in the form of articles
  - see links on webpage
  - **I'll be updating this as we go along**
- More suggestions for background reading, incl. some good books and web-sites

## Practicalities: form & examination

- 2-hrs lecture every week
  - read associated papers & ask questions!
- project work
  - PRefast for C++ (individual)
  - OWASP open code review of PHPBB (To be decided)
  - JML for Java (individual)
- written exam
  - 50% of grade, but you *must* do the projects, and you *must* pass the exam

Motivation

## Quiz

*What do web-sites, web-browsers, operating systems, wifi access points, network routers, mobile phones, PDAs, smartcards, firewalls, intrusion detection systems, and video-conferencing equipment have in common?*

***SOFTWARE!***

*Why can all these things be hacked, if we are not very careful?*

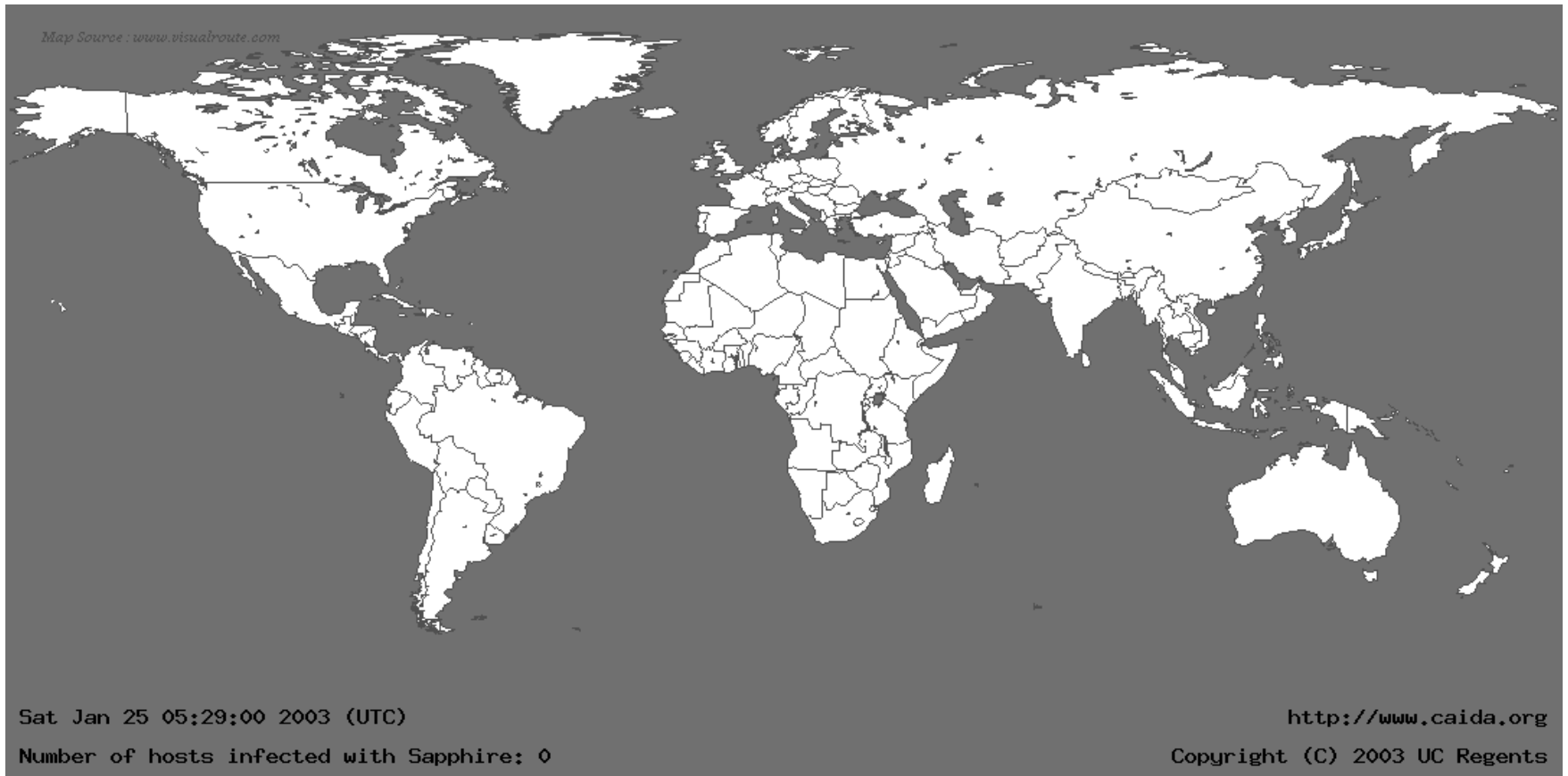
## Why a course on software security?

- Software plays a major role in providing security, and is a major source of security problems.
  - Software is *the weakest link* in the security chain, with the possible exception of "the human factor"
- Software security does not get much attention
  - in other security courses, or
  - in programming courses,or indeed, in much of the security literature!

We focus on software security, but don't forget that security is about, in no particular order, *people* (users, employees, sys-admins, programmers,...), access control, passwords, biometrics, cryptology, protocols, policies & their enforcement, monitoring, auditing, legislation, persecution, liability, risk management, incompetence, confusion, lethargy, stupidity, mistakes, complexity, *software*, bugs, verification, hackers, viruses, hardware, operating systems, networks, databases, public relations, public perception, conventions, standards, physical protection, data protection, ...

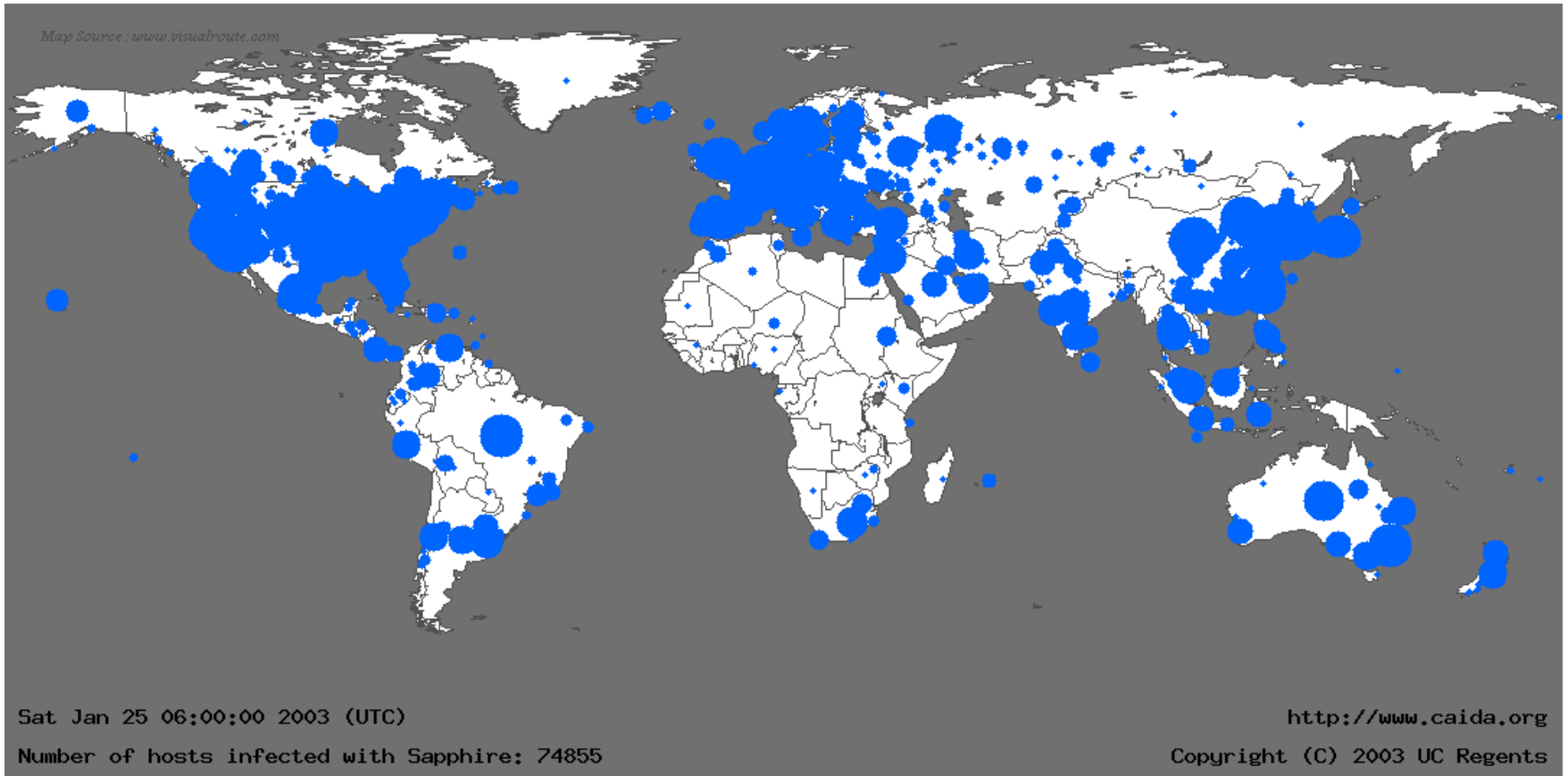
The problem

# Slammer Worm (Jan 2002)



Pictures taken from *The Spread of the Sapphire/Slammer Worm*, by David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, Nicholas Weaver

# Slammer Worm (Jan 2002)



Pictures taken from *The Spread of the Sapphire/Slammer Worm*, by David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, Nicholas Weaver

# Vulnerability in Cisco Router (source US-CERT)

Published: 2011-01-24

Vulnerability No: CVE-2011-0352

CVSS Severity Score: 7.88

Vendor/Product cisco -- linksys\_wrt54gc\_router\_firmware

Buffer overflow in the web-based management interface on the Cisco Linksys WRT54GC router with firmware before 1.06.1 allows remote attackers to cause a denial of service (device crash) via a long string in a POST request.

# Vulnerability in FFmpeg (source US-CERT)

Published: 2011-01-24

Vulnerability No: CVE-2010-4705

CVSS Severity Score: 9.3

Vendor/Product: ffmpeg -- ffmpeg

Integer overflow in the vorbis\_residue\_decode\_internal function in libavcodec/vorbis\_dec.c in the Vorbis decoder in FFmpeg, possibly 0.6, has unspecified impact and remote attack vectors, related to the sizes of certain integer data types. NOTE: this might overlap CVE-2011-0480.

# Vulnerability in Linux/Windows/MACOS

Published: 2011-01-24

Vulnerability : CVE-2011-0638 CVE-2011-0640 CVE-2011-0639

CVSS Severity Score: 9.3

Vendor/Product: Apple Mac OS X  
Microsoft - windows  
Linux - Linux kernel

Microsoft Windows /Mac OS X/ Linux does not properly warn the user before enabling additional Human Interface Device (HID) functionality over USB, which allows user-assisted attackers to execute arbitrary programs via crafted USB data, as demonstrated by keyboard and mouse data sent by malware on a smartphone that the user connected to the computer.

# Vulnerability in Mozilla/Bugzilla

Published: 2011-01-28

Vulnerability : CVE-2010-4568

CVSS Severity Score: 7.5

Vendor/Product: Mozilla - Bugzilla

Bugzilla 2.14 through 2.22.7; 3.0.x, 3.1.x, and 3.2.x before 3.2.10; 3.4.x before 3.4.10; 3.6.x before 3.6.4; and 4.0.x before 4.0rc2 **does not properly generate random values** for cookies and tokens, which allows remote attackers to obtain **access to arbitrary accounts** via unspecified vectors, related to an insufficient number of calls to the srand function

# Vulnerability in Tandberg videoconferencing

Published: Feb 2 2011

Vulnerability : CVE-2011-0354

Vendor/Product: Tandberg- video conferencing

TANDBERG Videoconferencing Systems Default Account Lets Remote Users Gain Root Access

The device includes a root administrator account with no password. A remote user can access the system with root privileges.

The root user account is used for advanced debugging and is not required for normal operations.

## Mini-assignment for coming week

To get an impression of the problem, have a look at

<http://www.securityfocus.com/vulnerabilities>

<http://www.us-cert.gov/cas/bulletins>

<http://www.securitytracker.com/>

Links are on the course webpage

Superficial analysis of the problem

# Observation 1

All these problems are due to *(bad) software*

Namely

- the Linux/Windows/Mac Operating System (OS)
- the router software
- the videoconferencing system software
- the FFmpeg graphics engine
- ...

Such software bugs are why constant patching of system is needed to keep them secure

## Observation 2

All these problems are due to bad software that

- can be executed/addressed over the network
  - eg. in case of Slammer worm
- executes on (untrusted) input obtained over the network
  - eg. in case of FFmpeg
- or - in the worst case - both

With ever more network connectivity, ever more software can be attacked.

# Changing target of attacks

- Traditionally, focus on **operating system** and **network**  
"Solutions"
    - **regular patching of OS**
    - **firewalls**
    - **virus scanners**
  - Increasingly, focus on
    - **web applications**
    - **web browser**
    - **mobile devices**
      - smartphones, tablet, that pass through firewalls
    - **embedded software**
      - software in cars, factories, infrastructure...
- and **targetted attacks** on specific organisation or person

## Changing nature of attackers

- Traditionally, hackers are **amateurs** motivated by fun
  - publishing attacks for the prestige
- Increasingly, hackers are **professional**
  - attackers go **underground**
    - zero-day exploits are worth money
  - attackers include
    - **organized crime**  
with lots of money and (hired) expertise
    - **government agencies:**  
with even more money & in-house expertise
      - 'Classic' example: Stuxnet  
[http://www.ted.com/talks/ralph\\_langner\\_cracking\\_stuxnet\\_a\\_21st\\_century\\_cyber\\_weapon.html](http://www.ted.com/talks/ralph_langner_cracking_stuxnet_a_21st_century_cyber_weapon.html)

# Software (in)security: crucial facts

- No silver bullets!

crypto or special security features do not magically solve all problems

- software security  $\neq$  security software
- “if you think your problem can be solve by cryptography, you do not understand cryptography and you do not understand your problem” [Bruce Schneier]

- Security is emergent property of entire system

- just like quality

- (Non-functional) security aspects should be integral part of the design, right from the start

The causes of the problem

## Quick audience poll

- How many of you learned to program in C or C++?
- ~ as a first programming language?
- How many of these courses
  - warned you about buffer overflows?
  - explained how to avoid them?

Major causes of problems are

- lack of awareness
- lack of knowledge

## Quick audience poll

- How many of you have built a web-application?
  - in which programming languages?
- What is the *secure* way of doing a SQL query in this language?
  - to avoid SQL injection flaws

Major causes of problems are

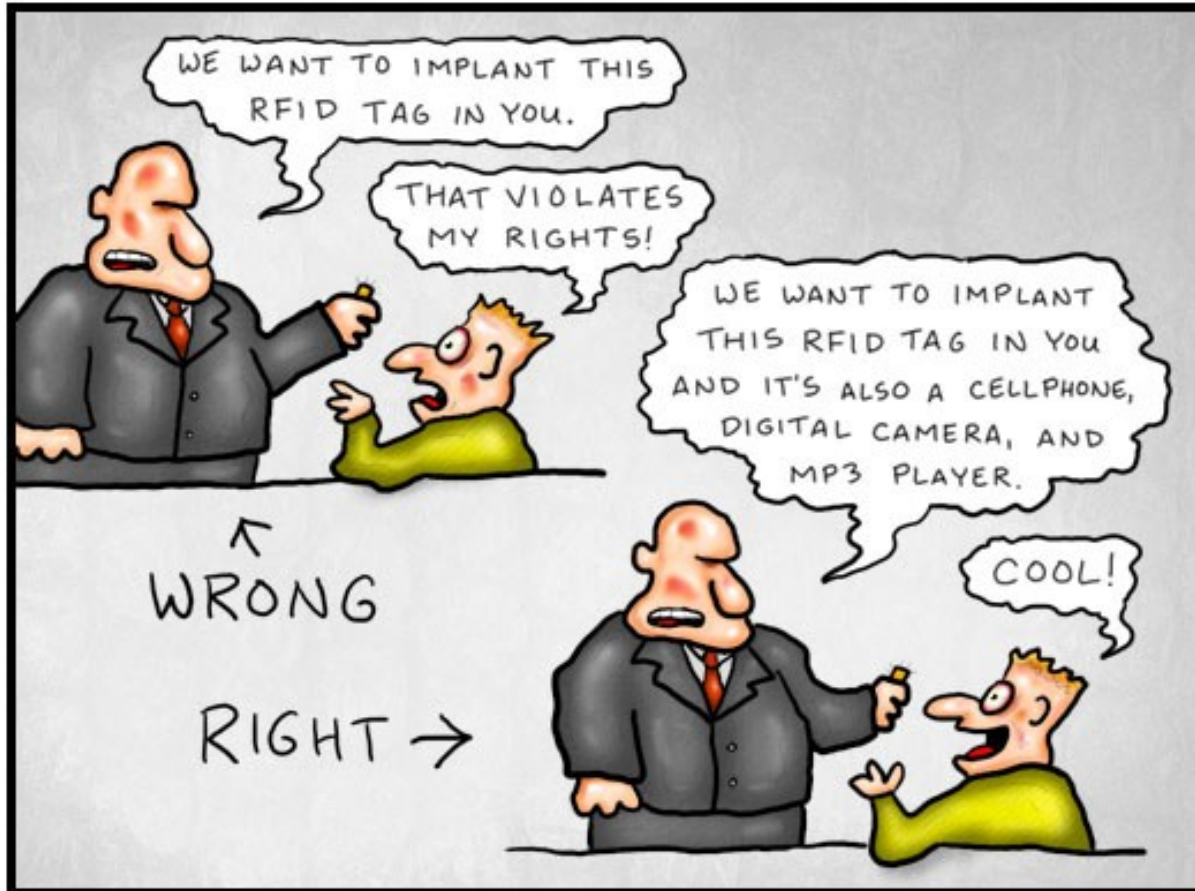
- lack of awareness
- lack of knowledge

# 1. Security is always a secondary concern

- Security is always a **secondary concern**
  - **primary goal** of software is to *provide* some **functionality** or **services**; *managing* associated **risks** is a derived/secondary concern
- There is often a trade-off/conflict between
  - security
  - functionality & conveniencewhere security typically loses out
  - more examples of this later...

# DOCTOR FUN

16 Jan 2006



Copyright © 2006 David Farley, d-farley@ibiblio.org  
<http://ibiblio.org/Dave/drfun.html>

This cartoon is made available on the Internet for personal viewing only. Opinions expressed herein are solely those of the author.

## Functionality vs security

- **Functionality** is about what software *should do*, **security** is (also) about what it *should not do*

*Unless you think like an attacker, you will be unaware of any potential threats*

# Functionality vs security: Lost battles?

- operating systems (OSs)
  - with huge OS, with huge attack surface
- programming languages
  - with easy to use, efficient, but very insecure and error-prone mechanisms
- webbrowsers
  - with plug-ins for various formats, javascript, ActiveX, VBscript, ...
- email clients
  - which automatically cope with all sorts of formats & attachments..

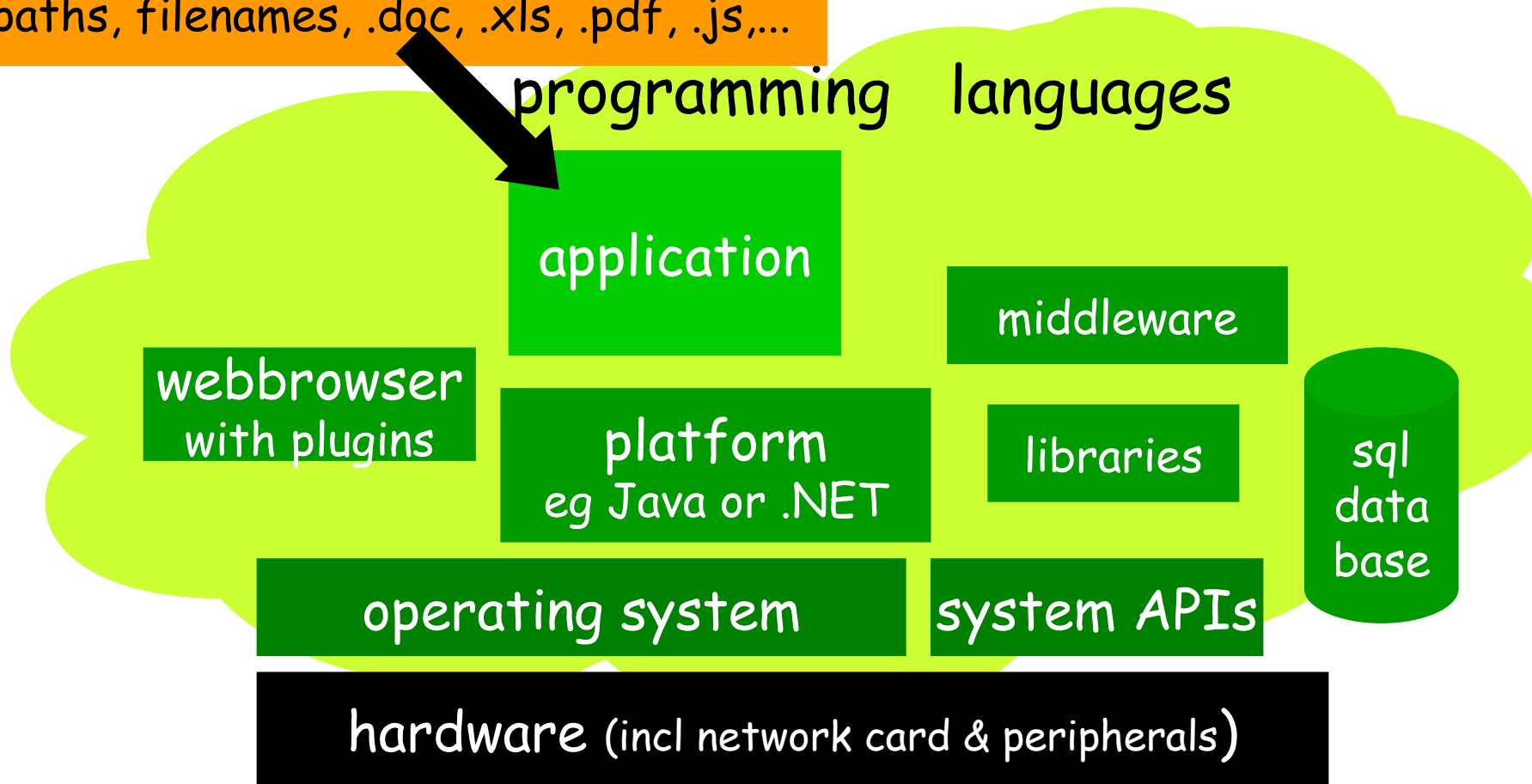
## Functionality vs security : PHP

"After writing PHP forum software for three years now, I've come to the conclusion that it is basically impossible for normal programmers to write secure PHP code. It takes far too much effort. .... PHP's raison d'etre is that it is simple to pick up and make it do something useful. There needs to be a major push ... to make it safe for the likely level of programmers - newbies. Newbies have zero chance of writing secure software unless their language is safe. ... "

[Source <http://www.greebo.cnet/?p=320>]

## 2. Weakness in depth

*interpretable or executable input*  
eg paths, filenames, .doc, .xls, .pdf, .js,...



## 2. Weakness in depth

### Software

- runs on a **huge, complicated infrastructure**
  - OS, platforms, webbrowser, lots of libraries & APIs, ...
- is built using **complicated languages**
  - programming languages, but also SQL, HTML, XML, ...
- using various **tools**
  - compilers, IDEs, preprocessors, dynamic code downloads

These may have **security holes**, or may **make the introduction of security holes very easy & likely**

## Recap

Problems are due to

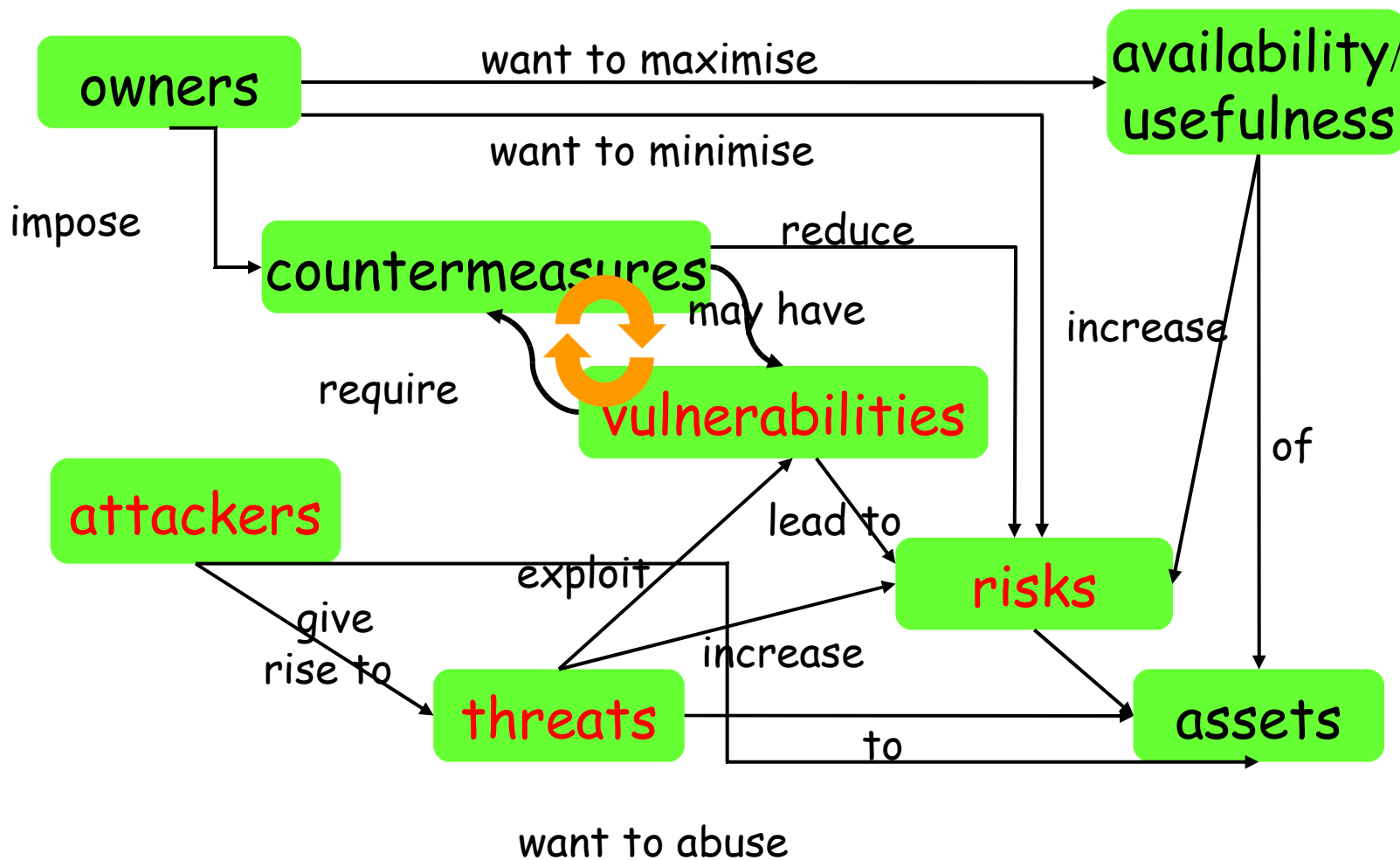
- lack of awareness
  - of threats, but also of what should be protected
- lack of knowledge
  - of potential security problems, but also of solutions
- compounded by complexity
  - software written in complicated languages, using large APIs, and running on huge infrastructure
- people choosing functionality over security

# Security concepts & goals

# Software and Security

- Security is about **regulating access to assets**
  - eg. information or functionality
- Software provides **functionality**
  - eg on-line exam results
- This functionality comes with certain **risks**
  - eg what are risks of on-line exam results?
- (Software) security is about **managing these risks**

# Security concepts



## Starting point for ensuring security

- Any discussion of security should start with an inventory of
  - the stakeholders,
  - their assets, and
  - the threats to these assets by possible attackers
    - employees, clients, script kiddies, criminals
- Any discussion of security without understanding these issues is *meaningless*

# Security concepts

- Security is about imposing *countermeasures* to reduce *risks* to *assets* to acceptable levels
- A *security policy* is a specification of what *security requirements/goals* the countermeasures are intended to achieve
  - secure against what and from whom ?
- *Security mechanisms* to enforce the policy
- Bottlenecks:
  - expressing what we (don't) want in a policy
  - enforcing this, dynamically or statically

# Security Objectives: CIA

- Confidentiality
  - unauthorised users cannot *read* information
- Integrity
  - unauthorised users cannot *alter* information
- Availability
  - authorised users *can* access information
- Non-repudiation for accountability
  - authorised users *cannot deny* actions

# Security objectives

- **Integrity** nearly always more important than **confidentiality**

Eg think of

- your bank account information
- your medical records
- *all* your software, incl. entire OS

- **Availability** may be **undesirable** for privacy
  - you want certain data to be or become **unavailable**

# Security goals

The well-known trio

- confidentiality, integrity, authentication (CIA)

but there are more "concrete" goals

- traceability and auditing (forensics)
- monitoring (real-time auditing)
- multi-level security
- privacy & anonymity
- ...

and meta-property

- assurance - that the goals are met

# How to realise security objectives? AAAA

- Authentication
  - who are you?
- Access control/Authorisation
  - control who is allowed to do what
  - this requires a specification of who is allowed to do what
- Auditing
  - check if anything went wrong
- Action
  - if so, take action

# How to realise security objectives?

Other names for the last three A's

- Prevention
  - measures to stop breaches of security goals
- Detection
  - measures to detect breaches of security goals
- Reaction
  - measures to recover assets, repair damage, and persecute (and deter) offenders

**NB *don't ever* be tempted into thinking that good prevention makes detection & reaction superfluous.**

Eg. breaking into any house with windows is trivial; despite this absence of prevention, detection & reaction still deter burglars.

# Threats vs security requirements

- information disclosure
  - confidentiality
- tampering with information
  - integrity
- denial-of-service (DoS)
  - availability
- spoofing
  - authentication
- unauthorised access
  - access control

# Countermeasures

- Countermeasures can be non-IT related
  - physical security of building
  - screening of personnel
  - legal framework to deter criminals
  - police to catch criminals
  - ...

but we won't consider these

Software security

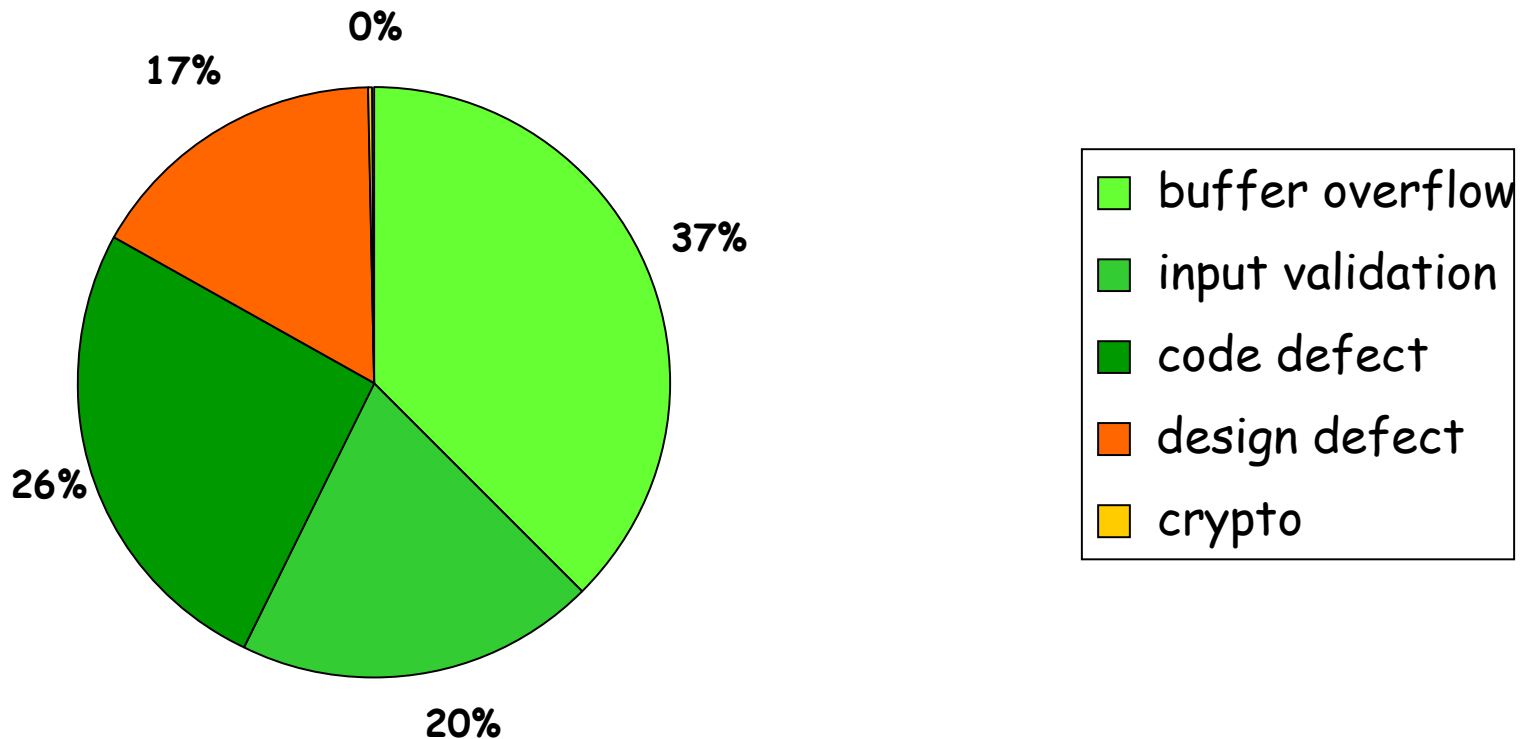
# Software security: vulnerabilities

Different types of software vulnerabilities

1. bugs aka implementation flaws or code-level defects  
vulnerability in the software introduced when implementing a system
2. design flaws  
vulnerability in the design

*Consensus: roughly speaking, bugs and design flaws are equally common*

# Typical software security vulnerabilities



Security bugs found in Microsoft bug fix month (2002)

- *The bad news*  
people keep making the same (types of) mistakes
- *The good news*  
people keep making the same (types of) mistakes  
  
..... so we can do something about it!

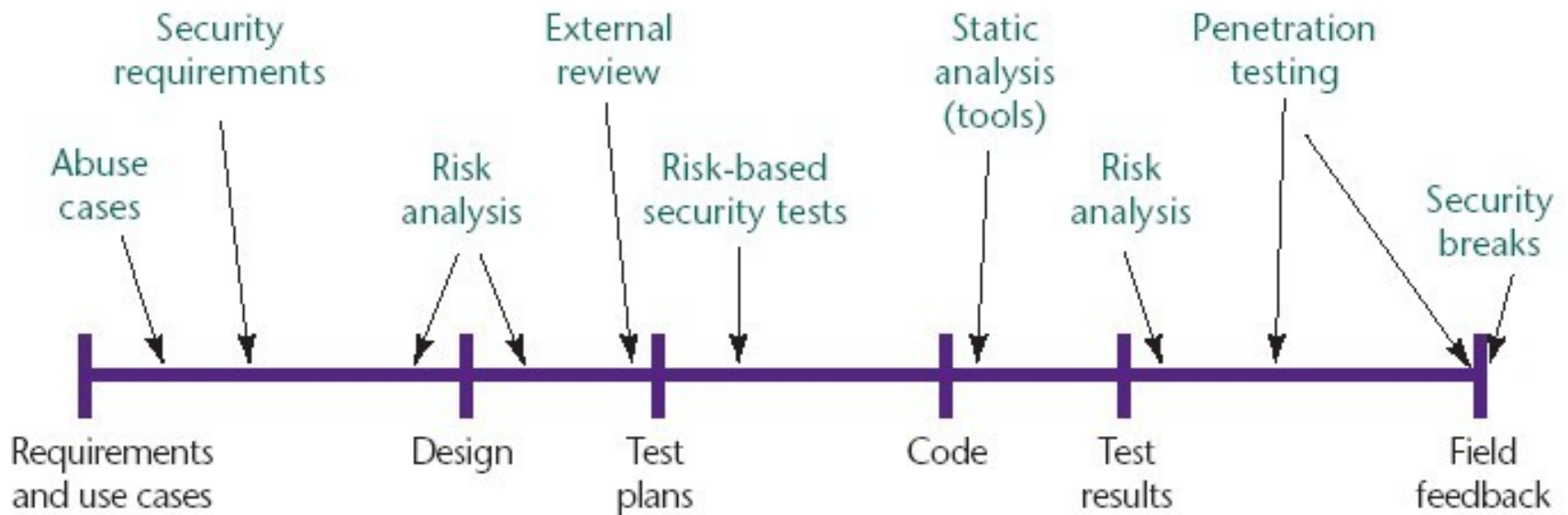
"Every upside has its downside" [J. Cruijff]

# Tackling Software Insecurity

- Knowledge about standard mistakes is crucial in preventing them
  - these depends on the programming language, the “platform” (OS, database systems, web-application framework,...), and the type of application
  - lots of info available on this now
- But this is not enough: security to be taken into account from the start, throughout software development life cycle
  - several ideas & methodologies to do this

# Security in Software Development Life Cycle

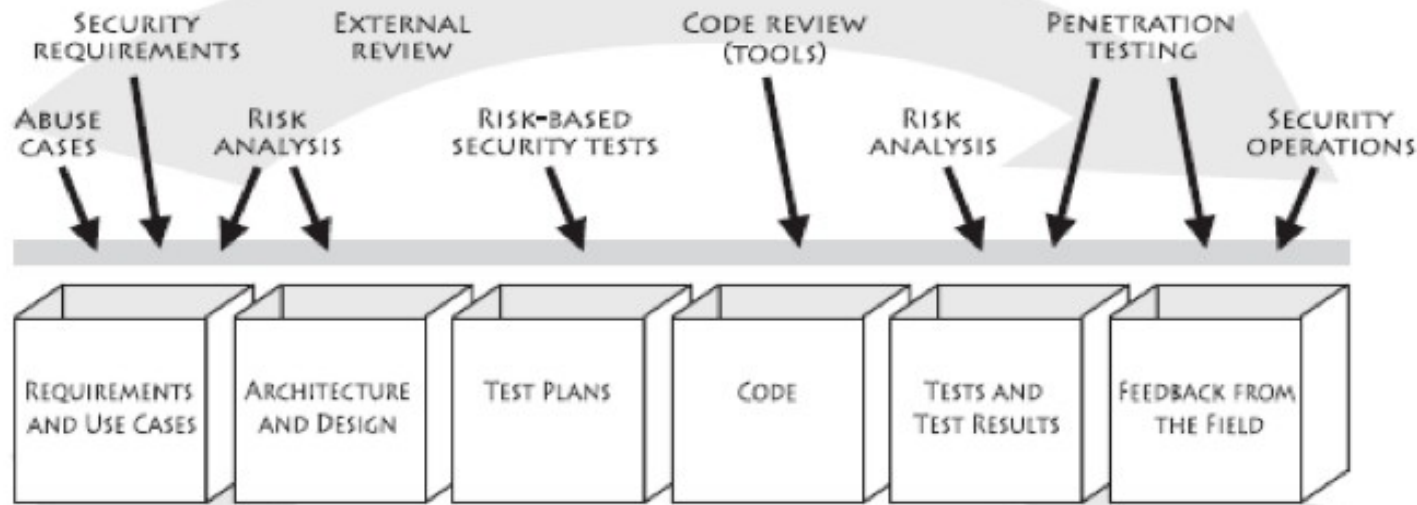
## McGraw's Touchpoints



[Source: Gary McGraw, *Software security*, Security & Privacy Magazine, IEEE, Vol 2, No. 2, pp. 80-83, 2004. ]

# Security in Software Development Life Cycle

## McGraw's Touchpoints



[book: Software Security: building security in, Gary McGraw, 2006]



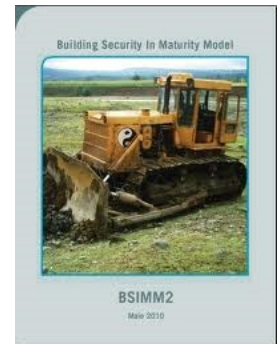
# Methodologies for security in development life cycle

Common/best practices, with methods for assessments, and roadmaps for improvement

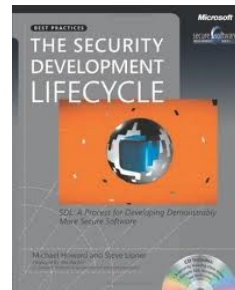
- McGraw's Touchpoints

BSIMM Building Security In - Maturity Model

<http://bsimm.com>



- Microsoft SDL

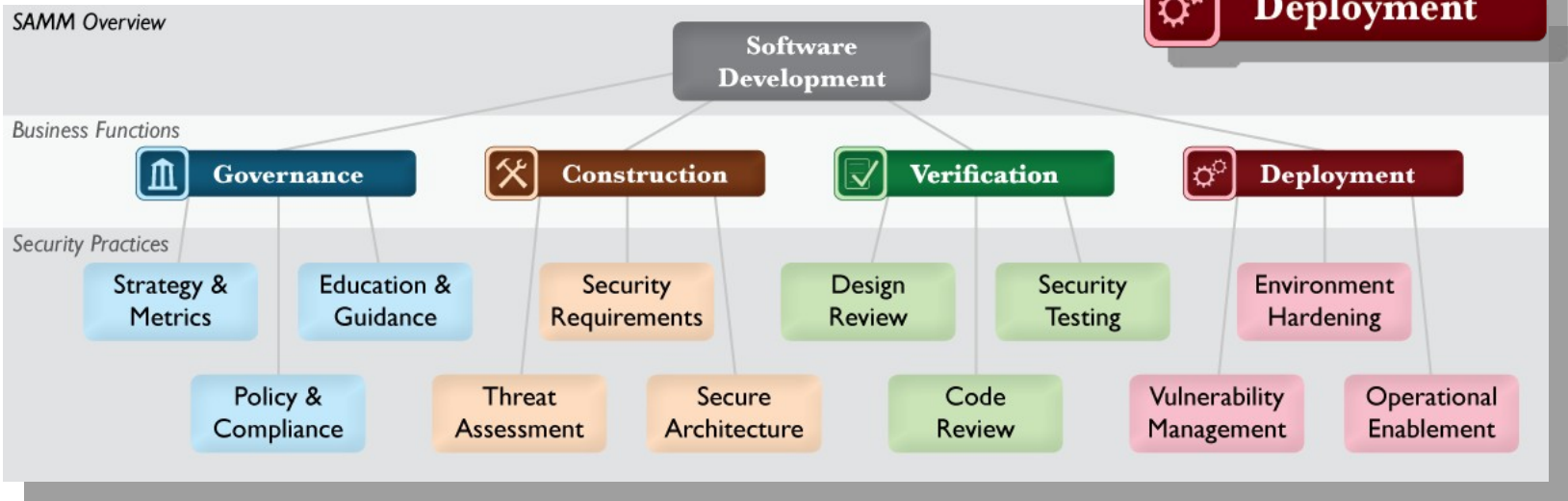


- OpenSAMM Software Assurance Maturity Model

<http://opensamm.org>



# OpenSAMM's 4 business functions and 12 security practices



# Microsoft's SDL Optimisation Model

## The four security maturity levels of the SDL Optimization Model



## The five capability areas of the software development process



## Microsoft SDL: STRIDE

- As part of SDL, Microsoft proposed STRIDE as methodology for threat modelling

S - Spoofing

T - Tampering

R - Repudiation

I - Information Disclosure

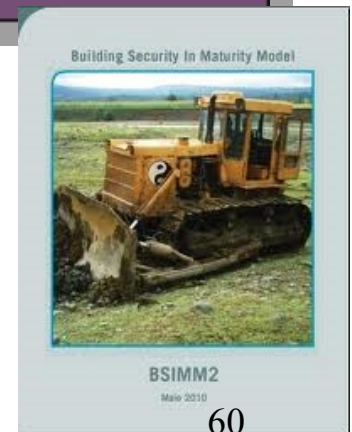
D - Denial-of-Service

E - Elevation of Privilege

# BSIMM

Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

Based on data collected from large enterprises



## To read coming week

- Gary McGraw,  
*Software security*,  
Security & Privacy Magazine, IEEE, Vol 2, No. 2, pp. 80-83,  
2004.
- Gary McGraw and Greg Morrisett,  
*Attacking Malicious Code: A Report to the Infosec Research  
Council*,  
IEEE Software, Vol. 17, No. 5, pp. 33-41, 2000.
- Check out websites for security alerts in the past week

See links on webpage