Software Security

# Introduction


Erik Poll

Digital Security

Radboud University Nijmegen


TRU/e Master in Cyber Security

# Admin

- NB IMC051 (5EC, for TRU/e) vs ISOFSE (6EC)

- All course material will be on

  http://www.cs.ru.nl/~erikpoll/ss

- Register in Osiris (and hence Brightspace)
  - *If you cannot, send me an email to get on my back-up mailing list !!!!!*

- *For TRU/e students: get on the TRU/e mailing list !!!!!*

  *https://true-security.nl/admission/*

# Upcoming events

- **Thursday Sept 27: OWASP evening here in Nijmegen**

  **Registration opens shortly at**

  **https://www.owasp.org/index.php/Netherlands**

- **Friday Oct 5 : TRU/e borrel & BBQ after the lecture**

  **As will also be announced on true mailing list**

# Goals of this course

- 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

# Practicalities: prerequisites

- Introductory security course
  - TCB (Trusted Computing Base), CIA (Confidentiality, Integrity, Availability) , non-repudiation, …

- Basic programming skills, in particular
  - C(++) or assembly/machine code
    - eg. `malloc()`, `free()`, `*(p++)`, `&x`
      `strings in C using char*`
  - Java or some other typed OO language
    - eg. `public`, `final`, `private`, `protected`, `Exceptions`
  - bits of PHP and JavaScript

# Sample C(++) code you will see next week

```
char* copying_a_string(char* string)  {
      char* b = malloc(strlen(string));
      strcpy(b,a);
      return(b);
}
int using_pointer_arithmetic(int pin[])  {
    int sum = 0;
    int *pointer = pin;
    for (int i=0; i<4; i++ ){
        sum = sum + *pointer;
        pointer++;
    }
    return sum;
}
```

# Sample Java code you will see next month

```
public int summingAnArray(int[] pin)
    throws NullPointerException,
           ArrayIndexOutOfBoundsException      {
     int sum = 0;
     for (int i=0; i<4; i++ ){
          sum = sum + a[i];
     }
     return sum;
}
```

# Sample Java OO code you will see next month

```
final class A implements Serializable {
  public final static SOME_CONSTANT 2;
  private B b1, b2;

  protected A ShallowClone(Object o)
        throws ClassCastException       {
      x = new(A);
      x.b1 = ( (A) o).b1;
      x.b2 = ( (A) o).b2;
      return x;
  }
}
```

# Literature & other resources

- Slides + reading material available at
  http:///www.cs.ru.nl/~erikpoll/ss
- Mandatory reading: articles and lecture notes
  - see links on webpage
  - I'll be updating this as we go along

- Some additional optional suggestions for background reading, incl. books and web-sites
  - Recommended: follow the Risky.Biz podcast for weekly security news

# Practicalities: form & examination

- 2-hrs lecture every week
  - read associated papers & ask questions!
- project work
  - PREfast for C++ (individual)
  - JML program verification for Java (individual, 6EC version only)
  - group projects (with 4 people) on web-application and/or testing
- written exam
  - 50% of grade, but you *must* do the projects, and you *must* pass the exam

# Today

- Organisational stuff

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

# Motivation

# Quiz

***Why can websites, servers, browsers, laptops, smartphones, wifi access points, network routers, mobile phones, cars, pacemakers, uranium enrichment facilities, … be hacked?***

**Because they contain** 


**When it comes to cyber security**
**software is not our Achilles heel**
**but our Achilles *body***

***'Achilles only had an Achilles heel, I have an entire Achilles body'***

***- Woody Allen***

# Why a course on software security?

- Software plays a major role in providing security, and is the 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,

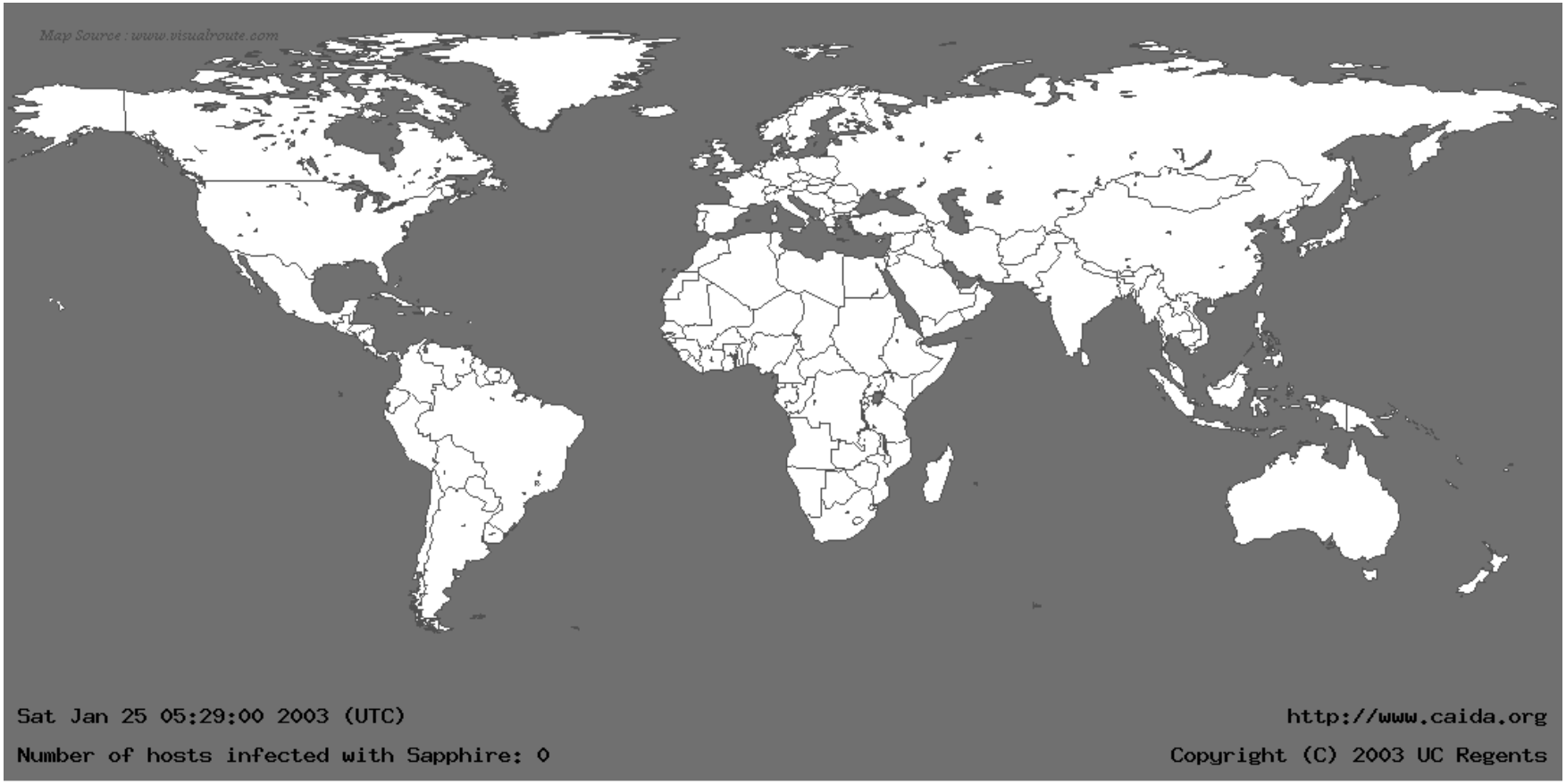
  or indeed, in much of the security literature!

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, protocols, policies & their enforcement, monitoring, auditing, legislation, cryptogaphy, 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, ...

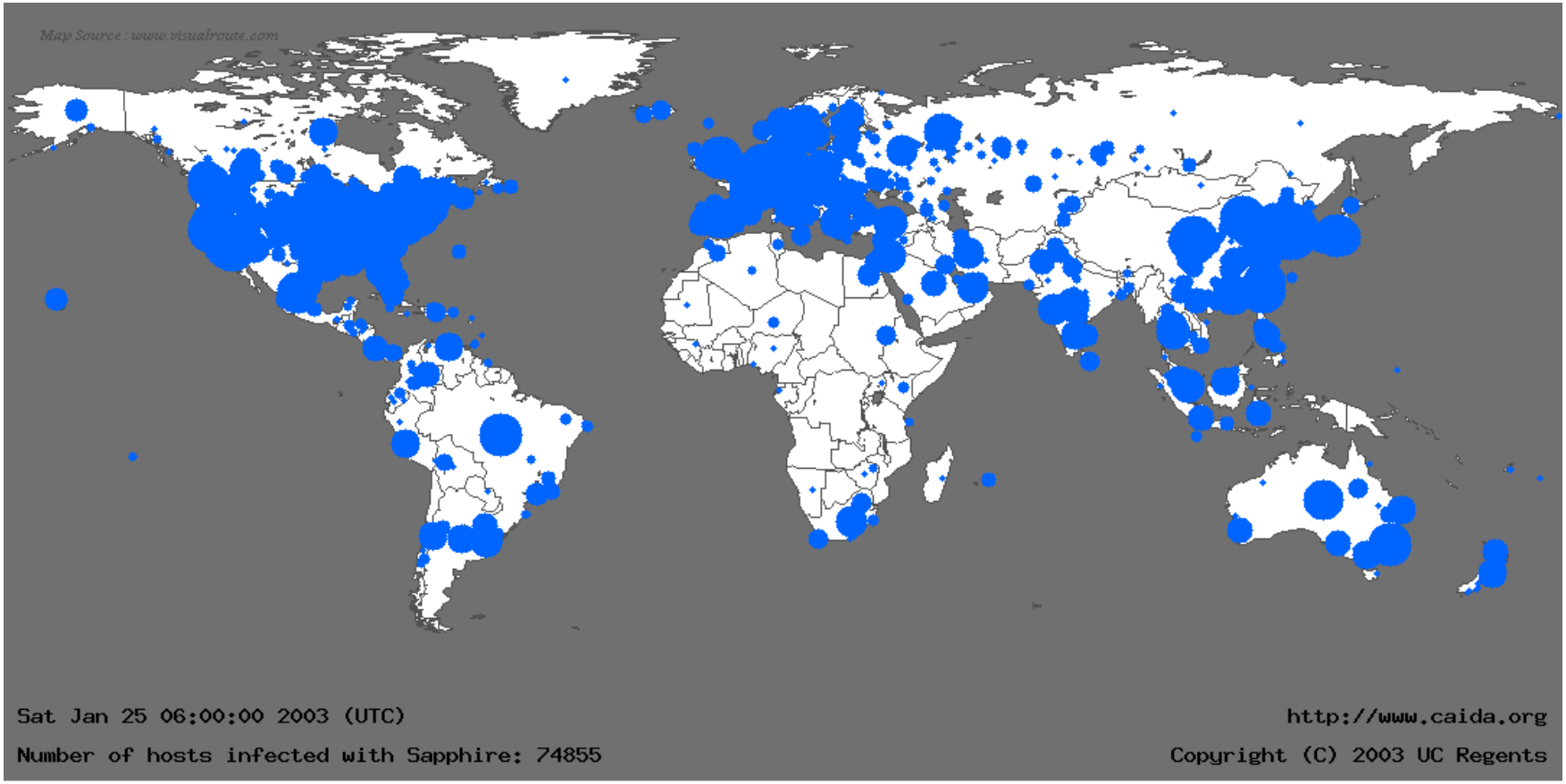
# The problem

# Slammer Worm (Jan 2002)




From *The Spread of the Sapphire/Slammer Worm*, by David Moore et al.

# Slammer Worm (Jan 2002)




From *The Spread of the Sapphire/Slammer Worm*, by David Moore et al.

# Security problems nowadays

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

http://www.us-cert.gov/ncas
bulletins & alerts
http://www.securitytracker.com/
http://www.securityfocus.com/vulnerabilities

Or subscribe to CVE twitter feed

https://twitter.com/cvenew

# Superficial analysis of the problem

1. All these problems are due to flawed software
   - Because of software flaws, constant patching is needed to keep systems secure

2. Most problems arise when software takes input over the network

With ever more software, and more network connectivity, things will only get worse…

# Changing target of attacks

Traditionally, focus on operating system and network

"Solutions"

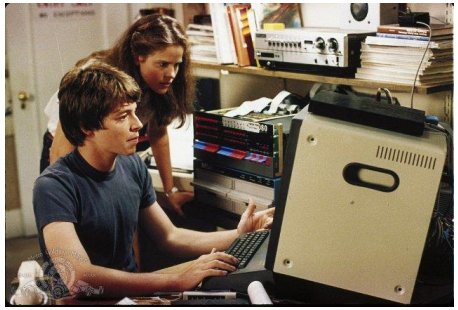
- regular patching of OS, firewalls, virus scanners

Then focus shifted to

- web applications
- web browser
- mobile devices
  - smartphones, tablets, that bypass firewalls
- embedded software
  - in cars, IoT devices, factories, critical infrastructures…

# Changing nature of attackers

hackers, 2010s



hackers, 1983

36 M€ internet banking fraud in NL in 2012
325 M$ in bitcoins collected by CryptoWall
950 M$ stolen by attack on SWIFT

Estonia DoS attack, stuxnet, Sony hack,
NSA hacks revealed by Snowden,
Ukraine electricity grid, hacks of political parties in US & elsewhere, ...

# Changing nature of attackers

Traditionally, hackers are amateurs motivated by fun

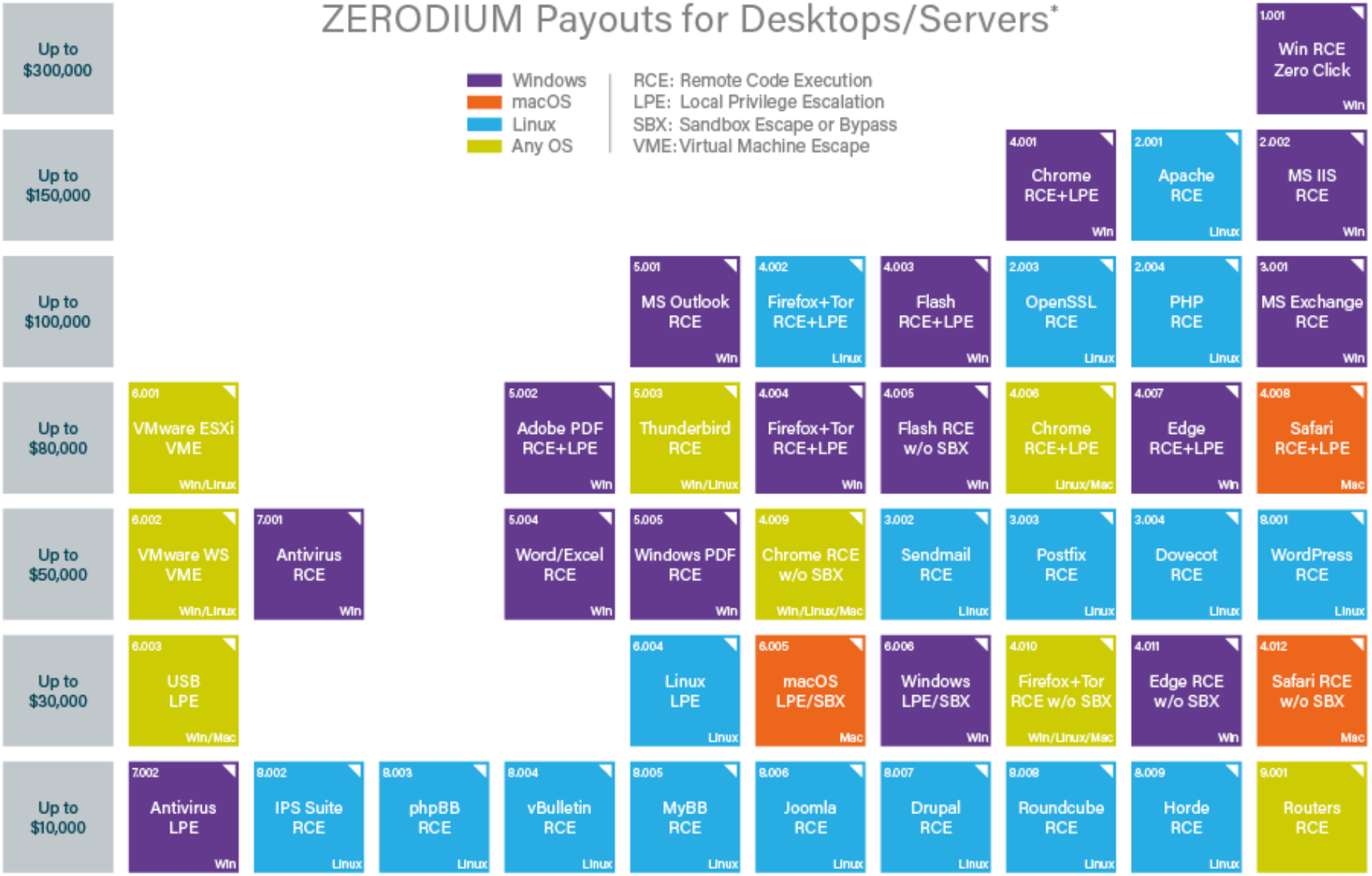
- publishing attacks for the prestige

Increasingly, hackers are professional

- attackers go underground
  - zero-day exploits are worth money
- attackers include
  - organized crime
    with lots of money and (hired) expertise

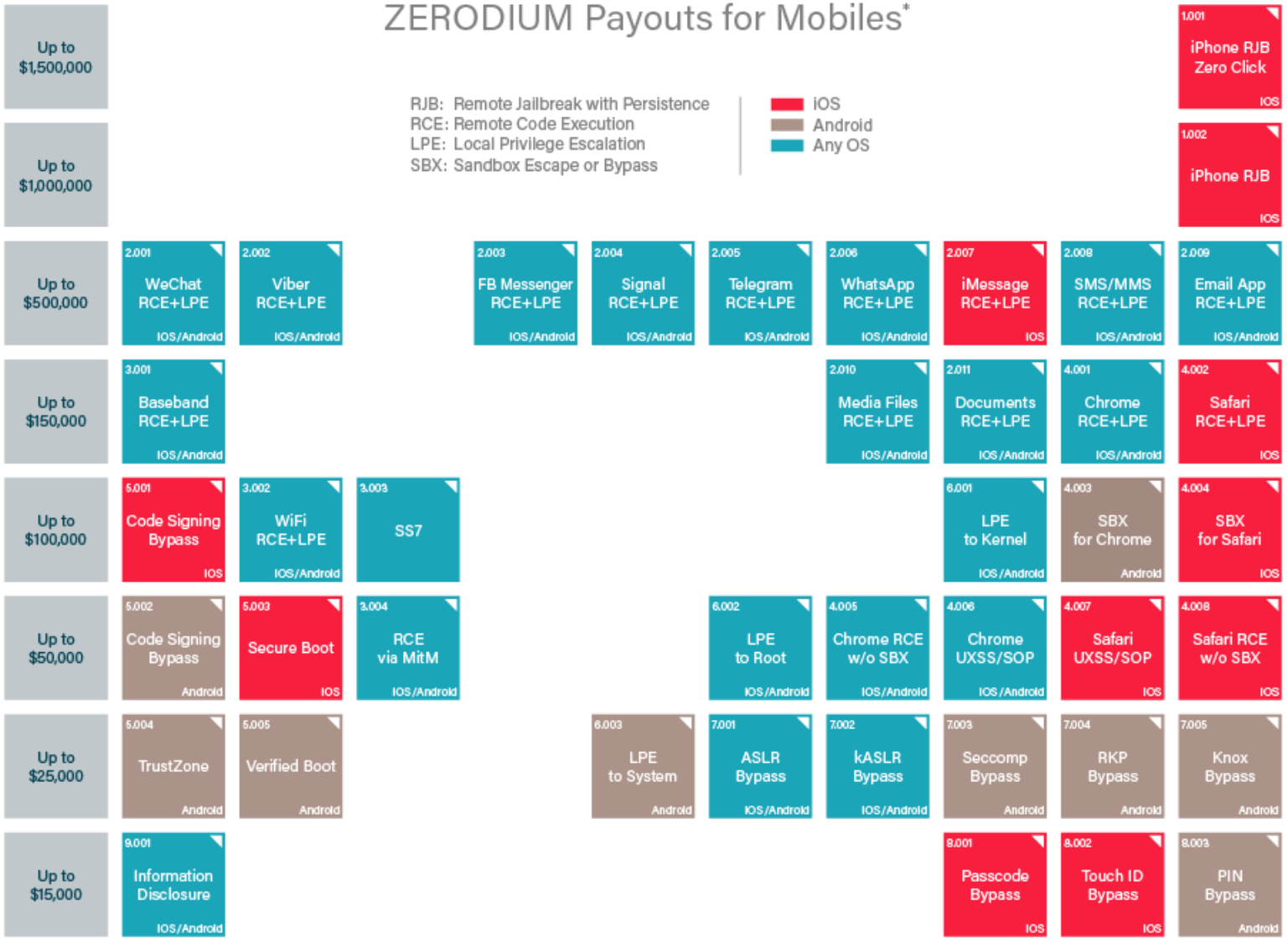
    Ransomware is an important game changer,
    as it allows attackers to monetise nearly anything.
  - government agencies:
    with even more money & in-house expertise

# Current prices for 0days




*All payouts are subject to change or cancellation without notice, at the discretion of ZERODIUM. All trademarks are the property of their respective owners.

# Current prices for 0days




** All payouts are subject to change or cancellation without notice, at the discretion of ZERODIUM. All trademarks are the property of their respective owners.*

# Software (in)security: crucial facts

- ***There are no silver bullets!***

  Crypto or special security features do not magically solve all problems

  - software security ≠ security software
  - "if you think your problem can be solved 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**
- **irresponsible teaching of dangerous programming languages**

# 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)*

**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 & convenience

  where security typically looses out
    - more examples of this later…

# 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
- web browsers
  - with plug-ins for various formats, JavaScript, ActiveX, Flash, ...
- email clients
  - which automatically cope with all sorts of formats & attachments..

# Functionality vs security : PHP

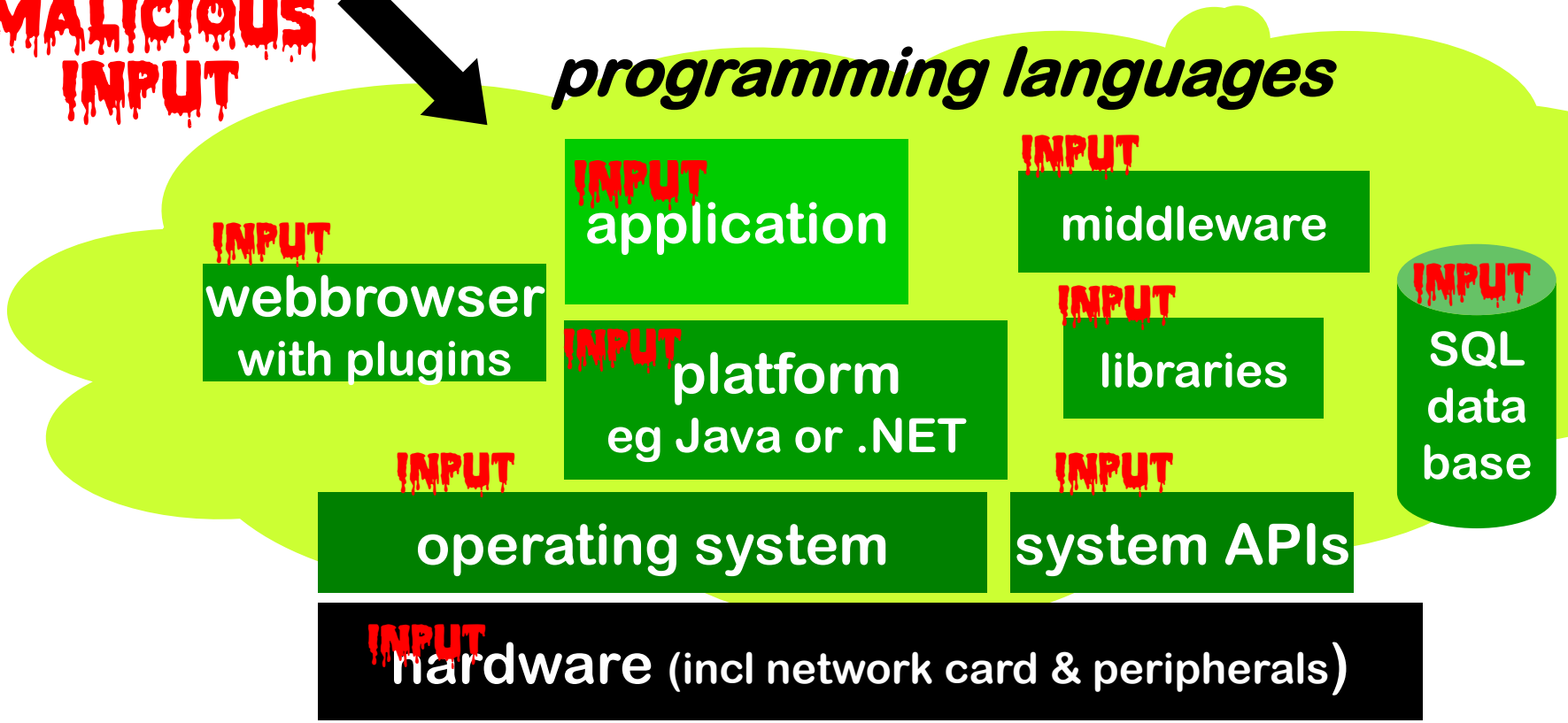
"After writing PHP forum software for three years now, I've come to the conclusion that it is basically impossible for normal programmers to write secure PHP code. It takes far too much effort. .... PHP's raison d'etre is that it is simple to pick up and make it do something useful. There needs to be 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

***input languages**, for*
*interpretable* or *executable* input, eg
pathnames, XML, JSON, jpeg, mpeg, xls, pdf...

MALICIOUS INPUT



# 2. Weakness in depth

Software

- runs on a huge, complicated infrastructure
  - HW, OS, platforms, web browser, lots of libraries & APIs, …
- is built using complicated languages
  - *programming* languages
    and *input* languages (SQL, HTML, XML, mp4, …)
- using various tools
  - compilers, IDEs, pre-processors, dynamic code downloads

All of 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
- people choosing functionality over security
- compounded by complexity
  - software written in complicated languages, using large APIs , and running on huge infrastructure

# Types of software security problems

# Flaws vs vulnerabilities

Terminology can be very confused & confusing:

security weakness, flaw, vulnerability, bug, error, coding defect…

Important distinction:

1. security weaknesses / flaws:

   things that are wrong or could be better

2. security vulnerabilities

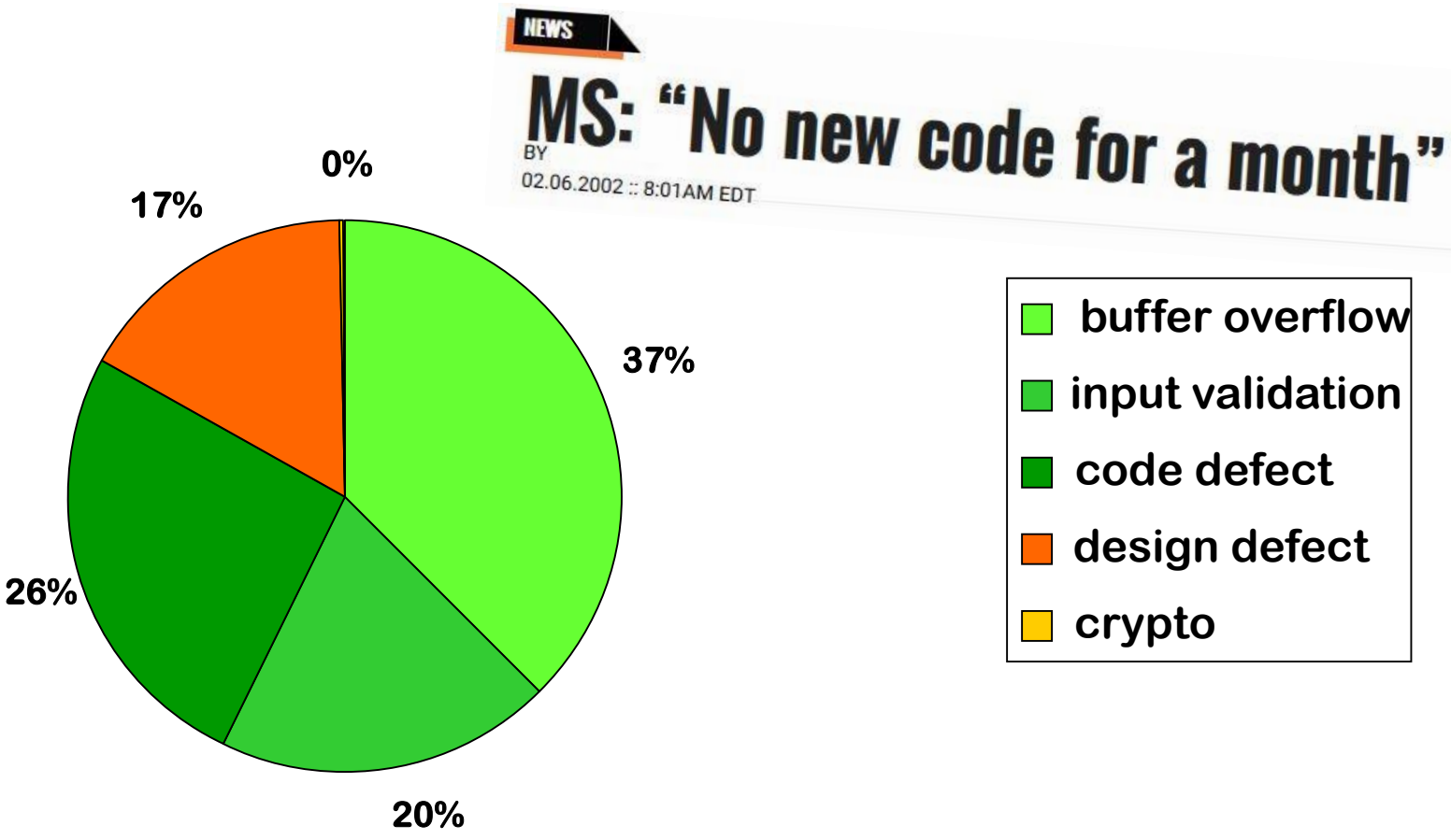
   flaws that can actually be exploited by an attacker

   This requires flaw to be

   - accessible: attacker has to be able to get at it
   - exploitable: attacker has to be able to do some damage with it

*Eg by turning off Wifi and BlueTooth network connection,*
*many security vulnerabilities become flaws*

# Typical software security flaws




Security bugs found in Microsoft's first bug fix month (2002)

# Software flaws

Software flaws can be introduced at two "levels"

1. design flaws
   vulnerability in the design
2. bugs aka implementation flaws aka code-level defects
   vulnerability in the software introduced during coding

*Overall consensus:*

*coding bugs and design flaws roughly equally common*

Vulnerabilities also arise on other levels (out of scope for now)

- configuration flaws when installing software on a machine
- the user
- unforeseen consequence of the *intended* functionality (eg spam)

# Coding flaws

For the flaws introduced during coding,

we make a rough distinction in

2a. flaws that can be understood looking at the program itself

eg. simple typos, confusing two program variables, off-by-one error in array access, errors in the program logic,...

2b. (common) problems in the interaction with the underlying platform or other systems and services, eg

- buffer overflows in C(++) code
- integer overflows in most programming languages
- SQL injection, XSS, CSRF,.... in web-applications
- ....

# The dismal state of software security

The *bad* news

people keep making the same mistakes

The *good* news

people keep making the same mistakes

…… so we can do something about it!

"Every upside has its downside" [Johan Cruijff]

# Spot the (security) flaws in electronic_purse.c

```
int balance;

void decrease(int amount)
 { if (balance <= amount)
         { balance = balance - amount; }
   else { printf("Insufficient funds\n"); }
 }

void increase(int amount)
 { balance = balance + amount;
 }
```

<= should be >=

what if amount is negative?

what if this sum is too large for an `int`?

# Different kinds of implementation flaws

what if amount is negative?

1. lack of input validation of (untrusted) user input
   - could be a design flaw rather than an implementation flaw?
   - more "fundamental" than flaws below

<= should be >=

2. logic error

what if sum is too large for a 64 bit `int`?

3. problem in interaction with underlying platform
   - "lower level" than the flaws above

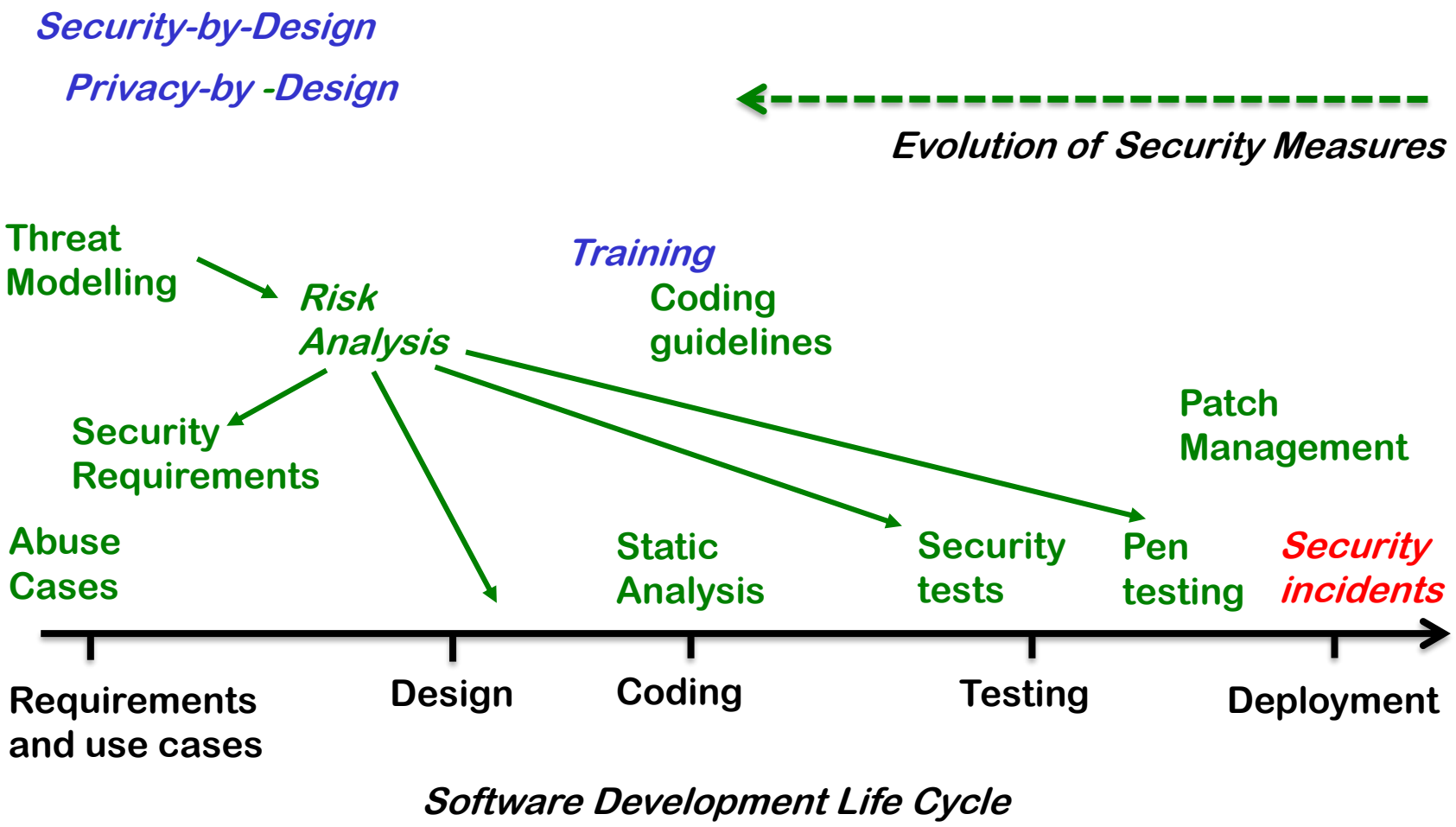
# Security in the software development life cycle (SDLC)

# 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
  - There is lots of info available on this now

- But this is not enough: security to be taken into account from the start, *throughout* the software development life cycle
  - several ideas & methodologies to do this

# Security in Software Development Lifecycle



# Evolution in tackling software security

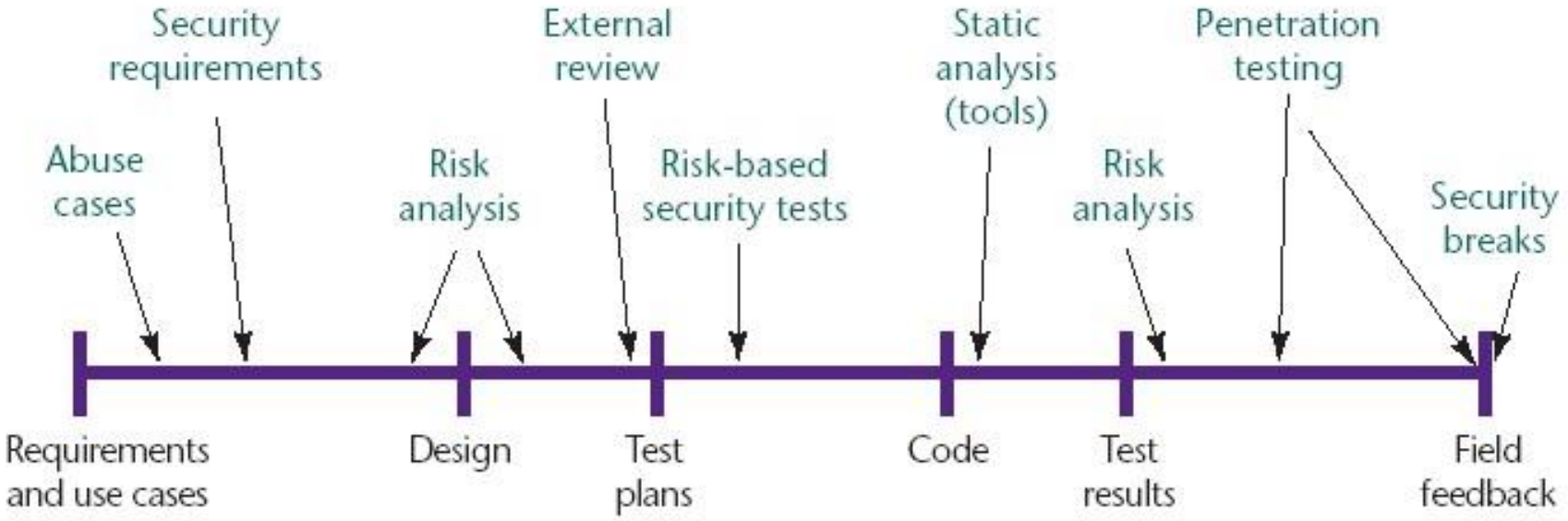
**Organisations always begin tackling security at the *end* of the SDLC, and then slowly evolve to tackle it earlier**

**For example**

1. **first, do nothing**
   - **some problems may happen & then you patch**
2. **then, implement support for regular patching**
3. **then, pre-emptively have products pen-tested**
   - **eg. hire pen-testers, set up bug bounty program, ...**
4. **then, use static analysis tools when coding**
5. **then, train your programmers to know about common problems**
6. **then, think of abuse cases, and develop security tests for them**
7. **then, start thinking about security before you even start development**

# Security in Software Development Life Cycle

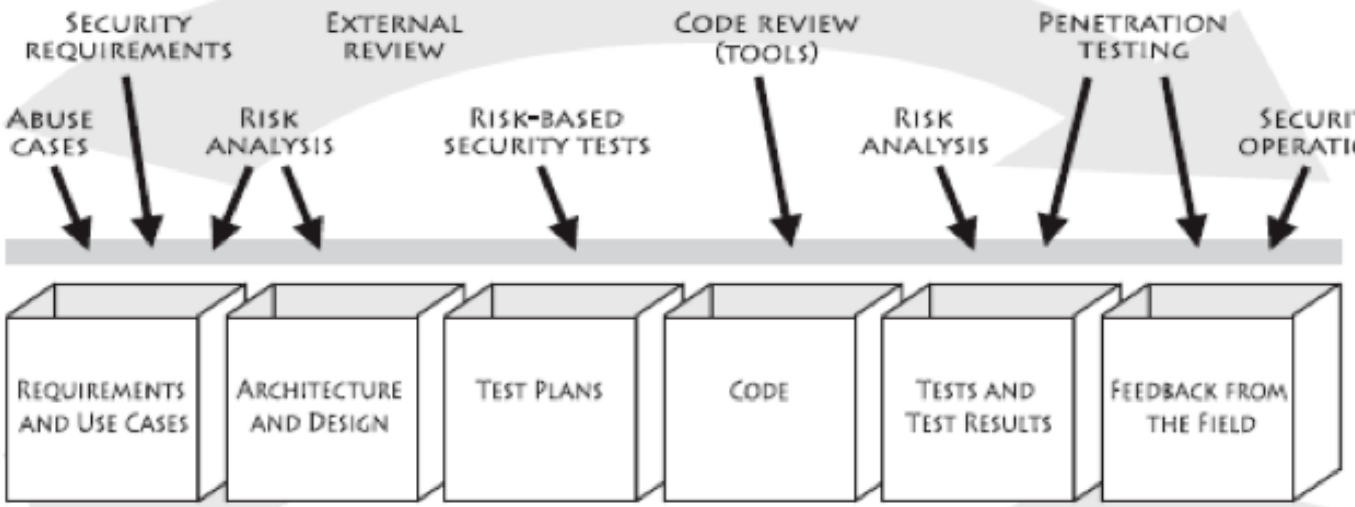
## McGraw's Touchpoints




**[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

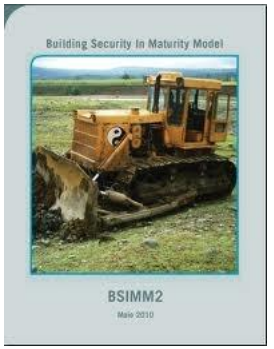




[book: Software Security: building security in, Gary McGraw, 2006]

# Methodologies for security in SDLC

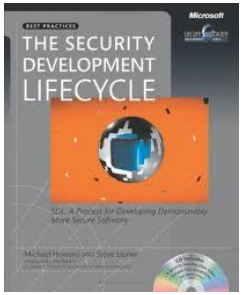
**Common/best practices, with methods for assessments, and roadmaps for improvement**




- **McGraw's Touchpoints**

  **BSIMM Building Security In – Maturity Model**

  **http://bsimm.com**




- **Microsoft SDL**

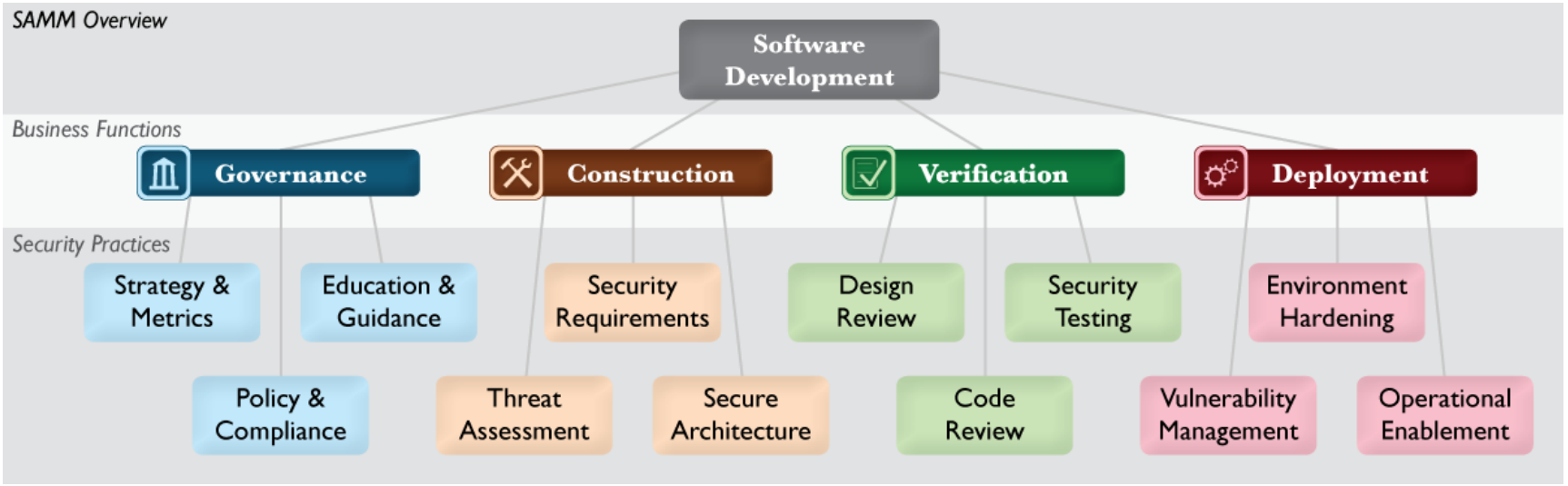
- **OpenSAMM Software Assurance Maturity Model**

  **http://opensamm.org**

OpenSAMM

# OpenSAMM's 4 business functions and 12 security practices





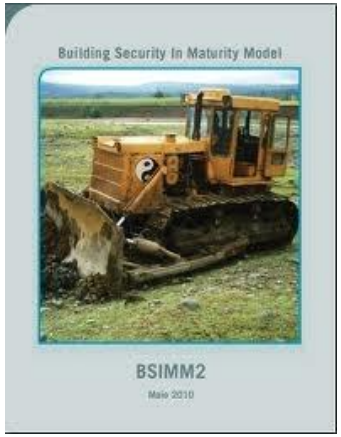
# Microsoft's SDL Optimisation Model


# BSIMM (Building Security In Maturity Model)

| 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**

**See https://www.bsimm.com/framework/**



# To read coming week

- Gary McGraw,
  *Software security*,
  Security & Privacy Magazine, Vol 2(2), pp. 80-83, 2004, IEEE

- Brian Chess & Brad Arkin
  *Software Security in Practice*
  Security & Privacy Magazine, Vol 9(2), pp. 89 - 92, 2011, IEEE

- Check out
  https://www.us-cert.gov/ncas/bulletins
  http://www.securitytracker.com/
  http://www.securityfocus.com/vulnerabilities
  for security alerts in the past week

# Security concepts & goals

**NB I assume you know all this stuff; if you don't, read up on it!**

- *"is this system secure?"*

- *"this system is secure"*

**Why are this question and this claim meaningless?**

**You have to say**

- **what it means for the system to be secure:**
  ***the security requirements***
- **against which attackers it has to be secure:**
  ***the attacker model***

# Starting point for ensuring security

Any discussion of security should start with inventory of

- the stakeholders
- their assets, esp. the crown jewels
- the threats to these assets
- attacker model
  What are the capabilities & motives of potential attackers?

  incl. employees, clients, script kiddies, criminals, NSA, or other ATPs (Advance Persistent Threats)

Any discussion of security without understanding these issues is *meaningless*

# Trusted Computing Base (TCB)

**TCB is the collection of software and hardware that we *have to* trust for our security**

- **So if any part of the TCB is compromised, we're screwed…**
- **So the attacker model and the TCB are complementary**

**NB1 We want the TCB to be as small as possible**

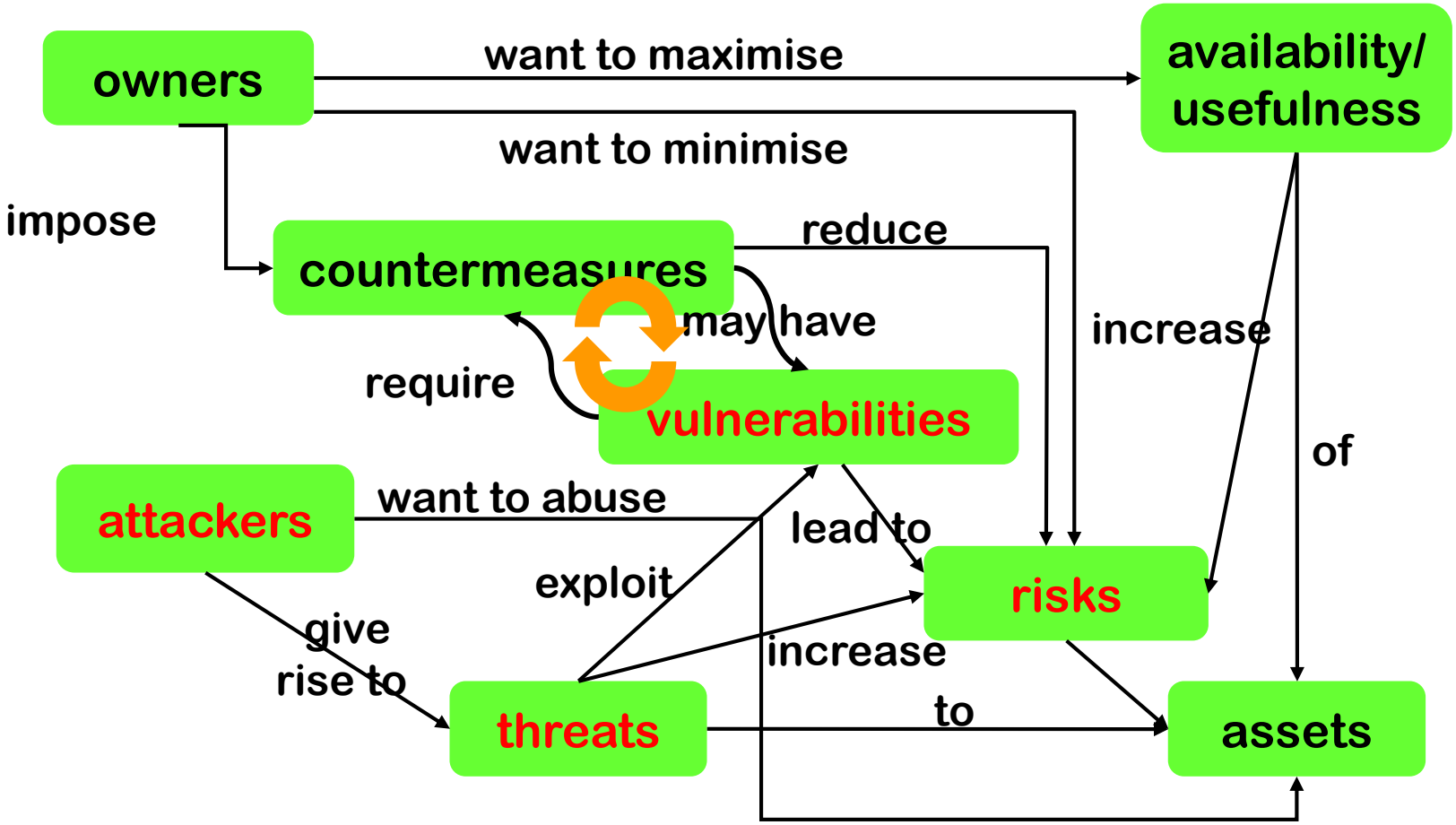
**NB2 Trust is bad; we want to minize trust**

**For a typical application, the TCB is huge, as it will usually include the operating system, the compiler, lots of third-party libraries we downloaded over the internet, …**

# Software and security

- Security is about regulating access to assets
  - incl. information and 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



# Security Objectives: CIA

- **Confidentiality** unauthorised users cannot *read* information
- **Integrity** unauthorised users cannot *alter* information
- **Availability** authorised users *can access* information

  In Dutch: **BIV** = Beschikbaarheid, Integriteit, Vertrouwelijkheid
- **Nonrepudiation** for **accountability** users *cannot deny* actions

- **Authentication** – knowing who/what you are interacting with

There are more kinds of security objectives:

- being able to do monitoring
- having logs for auditing and forensics
- privacy
- anonymity
- …

# Integrity vs Confidentiality

- Integrity nearly always more important than confidentiality

Eg think of

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

# Threats vs security requirements

Sometimes it is easier to think in terms of threats than in terms of security requirements, eg

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

# 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, in an access control policy
- 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.

# 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

# Assurance

**The crucial meta-property:**

**assurance that the system is secure,**

**ie. meets its security objectives**

**For software, level of assurance depends on**

- ***size*** **of the TCB (Trusted Computing Base)**
- ***quality*** **of the TCB**