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

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, ...

Software may well be the weakest link in the security chain, but

> *"it may also be argued that this chain is hidden in a mud pie: it is hard to find the links, to figure out if they hang together, or if anyone notices or cares if it's removed altogether:*
>
> *…the mud pie will still be there…" [Arjen Lenstra]*

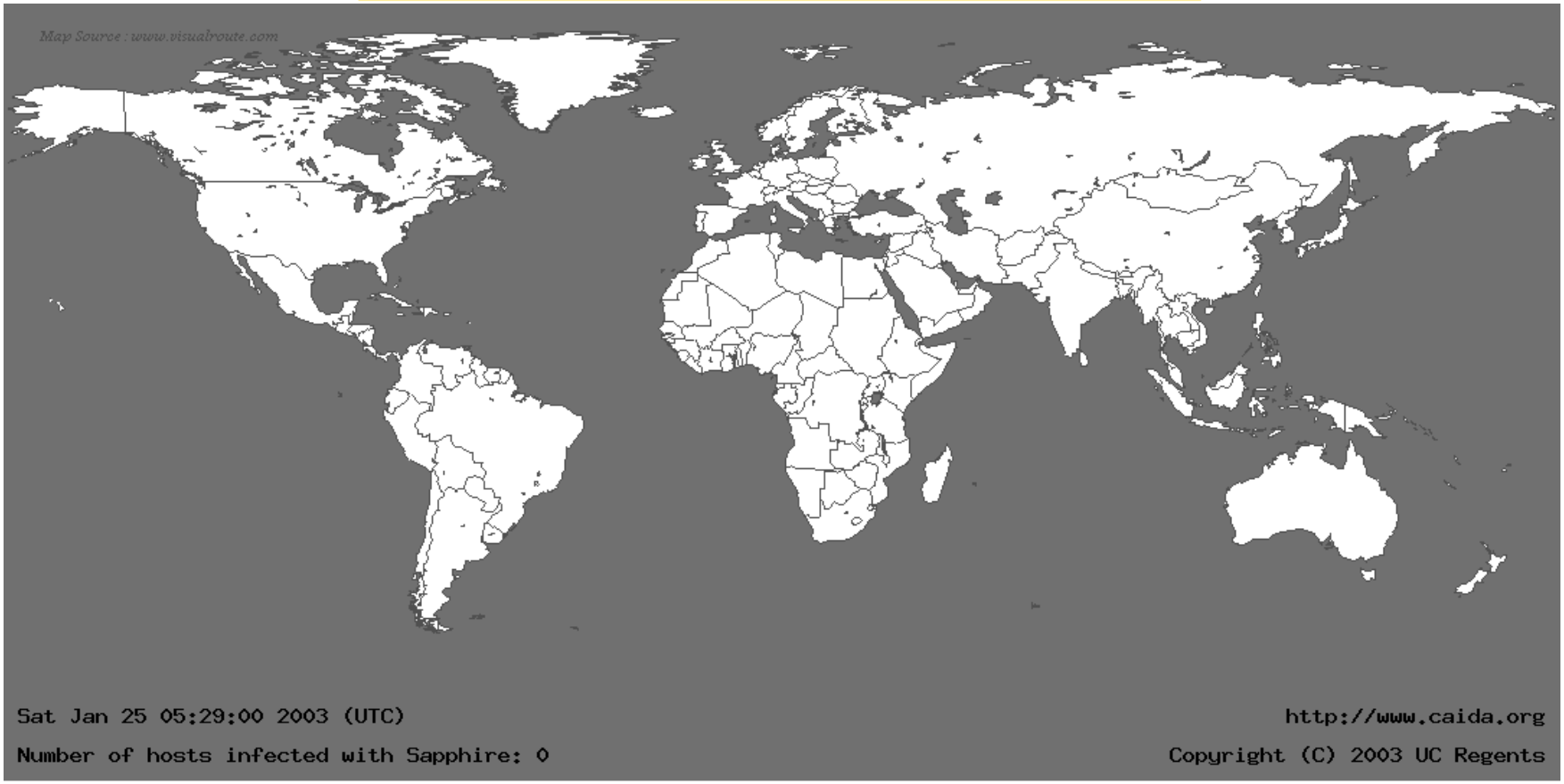
The problem

Internet worms and viruses

- virus = harmful piece of code that can infect other programs
- worm = self-replicating virus; no user action required for spreading infection

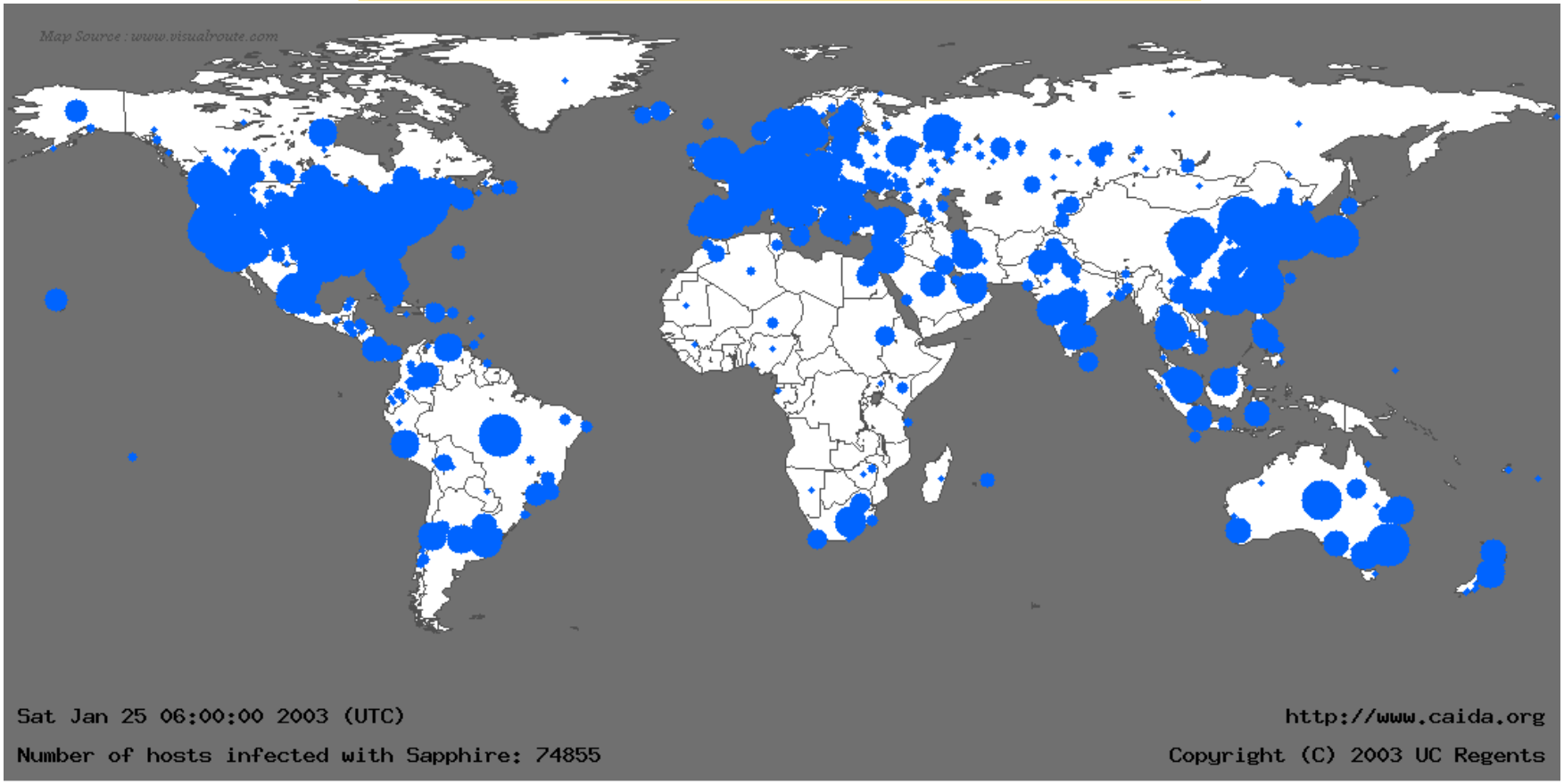
- First worm: Nov 1988, crashed 10% of internet
- More recently
  - email viruses: I Love You, Kounikova, ...
  - Worms: Slammer, Blaster, ...
- More recently still: attackers have gone underground & commercial

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 FFmeg 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 over the network, or
  - eg. in case of Slammer worm
- executes on (untrusted) input obtained over the network
  - eg. in case of FFmpeg

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

Rise of web application (in)security

- Traditionally, focus on operating system and network security
  - regular patching of OS
  - firewalls, virus scanners
- Increasingly, web applications and web browser are weak link and obvious targets to attack

Also mobile devices and embedded software are targeted.

- Traditional distinction between OS, network, and application gradually disappearing anyway:
  - OS-like functionality in programming platforms such as Java and .NET
  - webbrowser as the OS of the future ?

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_cyberweapon.html

Current prices for 0days

** All payout amounts are chosen at the discretion of ZERODIUM and are subject to change or cancellation without notice.*

The causes of the problem

Quick audience polls

- how many of you learned to program in C or C++?
- how many of you have built a web-application?
  - in which languages?

Major causes of problems are

- lack of awareness
- lack of knowledge

1. Security is 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 an application *does*, security is about what an application *should not do*

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

Functionality vs security

Lost battles?

- operating systems
  - huge OS, with huge attack surface (API),
- programming languages
  - buffer overflows, format strings, ... in C
  - public fields in Java
  - ...
- web browsers
  - plug-ins for various formats, javascript, ajax, VBscript, ...
- email clients

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]

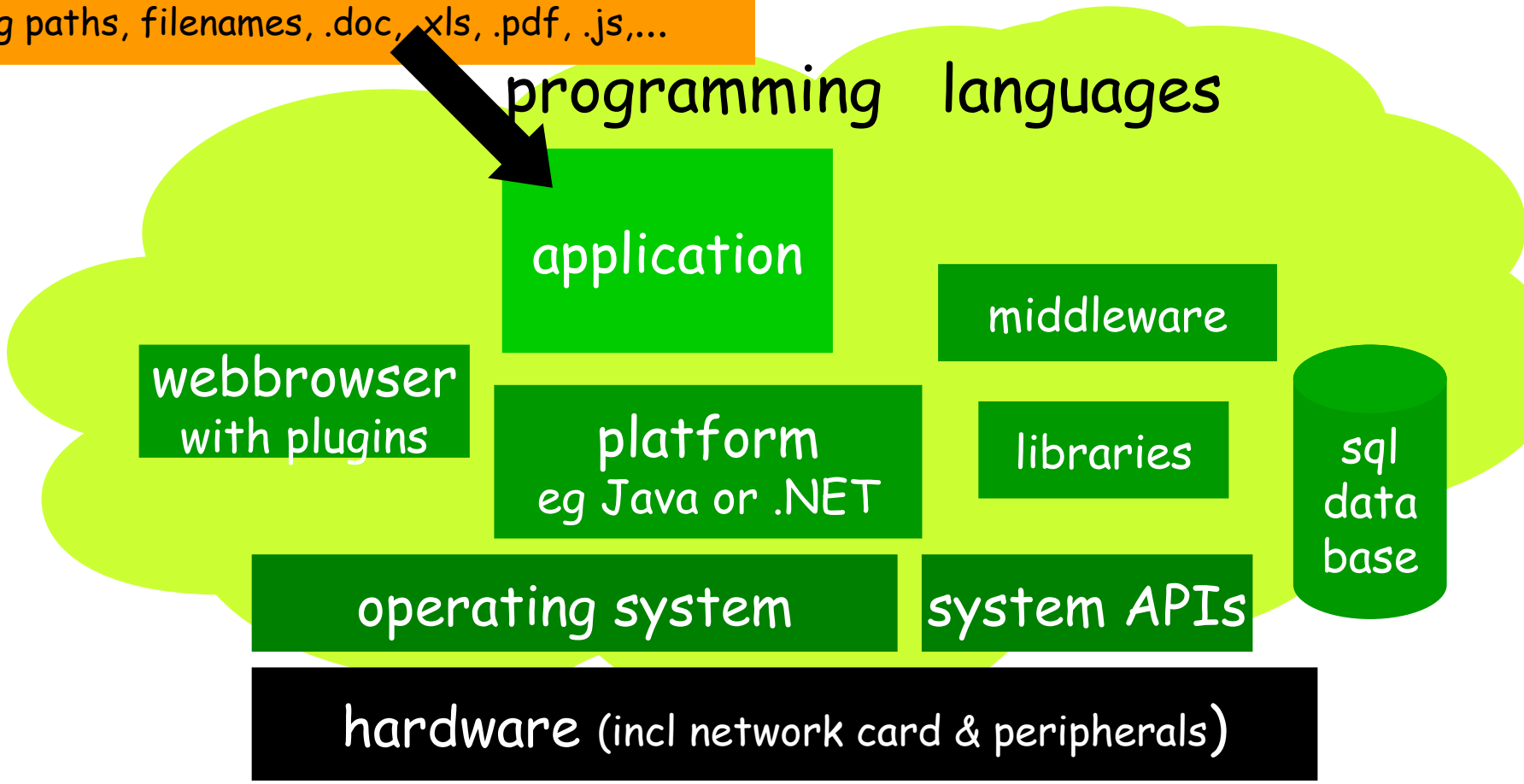
First steps in improving software security

- awareness
  - that there might be a problem
  - of what needs protecting, from which threats
  - of the fact that you might lack knowledge

2. Weakness in depth

interpretable or **executable** input data

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, and may make the introduction of security holes very easy & likely

3. Unfair battle

The fight against hackers in unfair:

the attacker only has to get lucky once,

the defender has to get it right all the time

(cf. football)

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

Flaw or Vulnerability ?

Confusing terminology

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

Important distinction

1. Security **weakness / flaw**

   Something that is wrong or could be better …

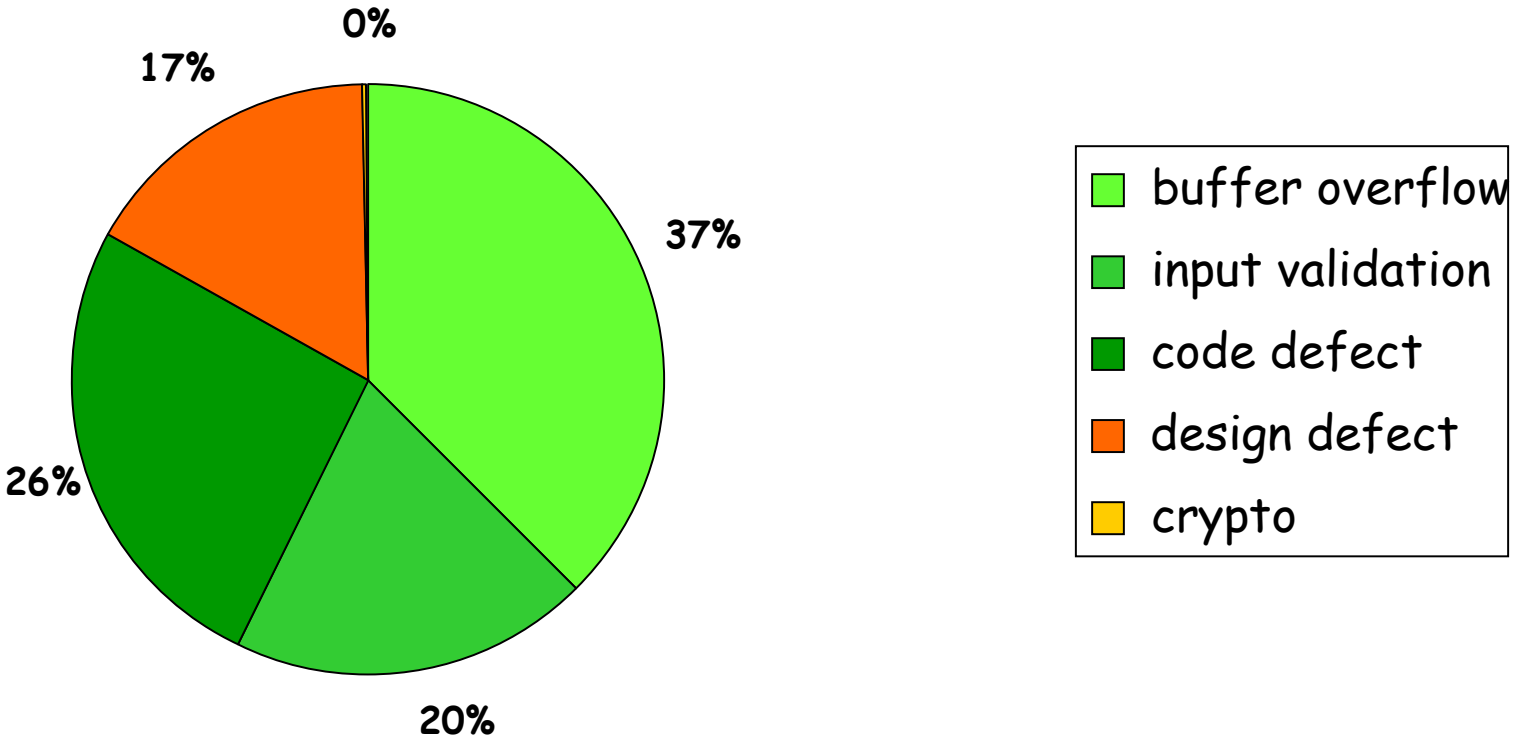
2. Security **vulnerability**

   Flaw that can be exploited by an attacker to violate a policy

   So, a flaw must be

   - Accessible: an attacker must have access to it
   - Exploitable: an attacker must be able to use it to compromise system

Typical software security vulnerabilities



Security bugs found in Microsoft bug fix month (2002)

Software Flaws

Software flaws can be introduced at two levels

1) Design flaw – the flaw is introduced during the design

2) Bug / code-level flaw - the flaw is introduced during implementation

Equally common

Vulnerabilities can also arise from other levels

- Configuration flaws – when installing the SW
- "User" flaws
- Unforeseen consequences of intended functionality (e.g., spam …)

Coding Flaws

Software flaws can be introduced during implementation can be roughly distinguished into

1) Flaws that can be understood by looking at the program

   e.g.; typos, confusing program variables, off-by-one, access to array, error in program logic, …

2) Flaws due to the interaction with the underlying platform or with other systems

   - Buffer overflow in C(++) code
   - Integer overflow/underflow in most programming languages
   - SQL injection, XSS, CSRF, … in web applications

Spot the security flaws

```
int balance;

void decrease(int amount)
{    if (balance <= amount)
          { balance = balance – amount; }
     else { printf(“Insufficient funds\n”); }
}

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

Spot the security Flaws

```
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 the sum is too large for an int ?

Different implementation Flaws

should be >)

what if amount is negative?

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

1. Logic error

   Can be found by code inspection only

2. Lack of input validation of (untrusted) user

   Design flaw or implementation flaw ?

3. Problem with interaction with underlying platform.

   lower level than previous ones

tackling software insecurity

- To prevent standard mistakes, knowledge is crucial
  - mistakes may depend on the programming language, on the platform (Op.Sys., DB, Web app, ...) and on the (type of) application

- but knowledge alone is not enough: security must be taken into account from the beginning and throughout the software development life cycle

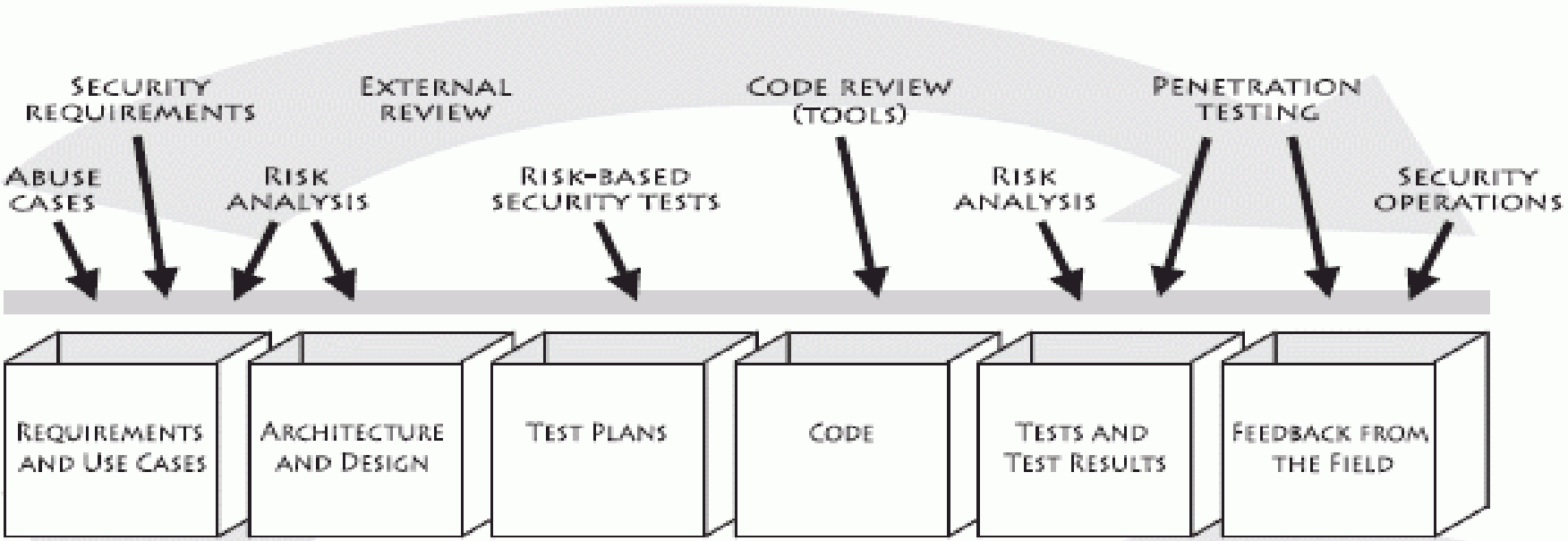
Evolution in tackling software security

Organizations tackle security at the end of the SDLC and with time have moved the concern to earlier stages

for example, chronologically:

1) First, **do nothing**
   - Some problem may happen and then we patch
2) then implement support for **regular patching**
3) Products are **pen-tested** pre-emptively
4) Use **static analysis** tools on code produced
5) then **train** programmers to know about common problems
6) then think about **abuse cases** and develop security **test** for them
7) then start thinking about security **before** starting the development

Security in software development life cycle



G.McGraw's Touchpoints

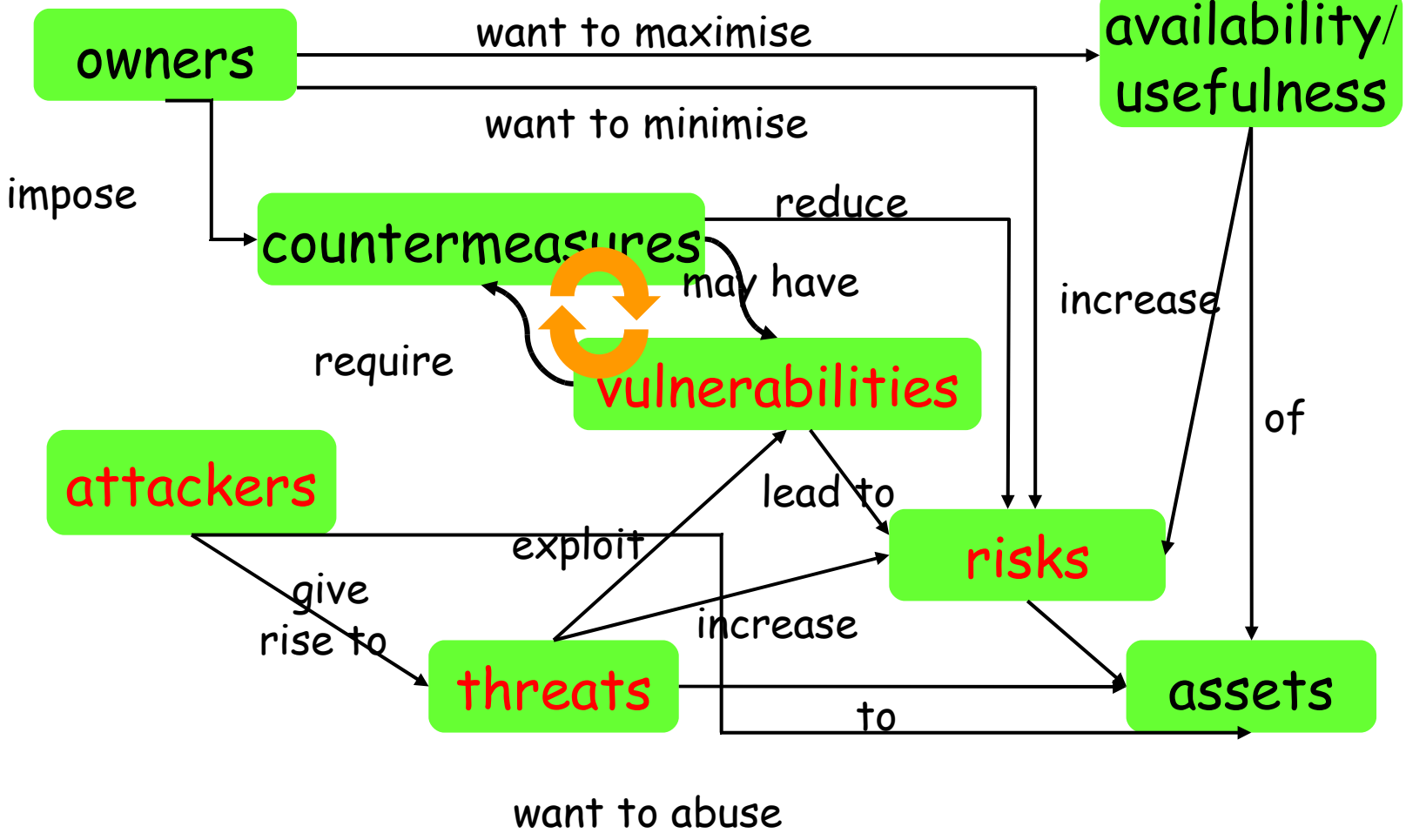
Security concepts & goals

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 **un**desirable 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* 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

Countermeasures and more vulnerabilities

Countermeasures can lead to new vulnerabilities

- eg. if we only allow three incorrect logins, as a countermeasure to brute-force password guessing attacks, which new vulnerability do we introduce?

If a countermeasure relies on software,

bugs in this software may mean

- that it is ineffective, or
- worse still, that it introduces more weaknesses

Software security

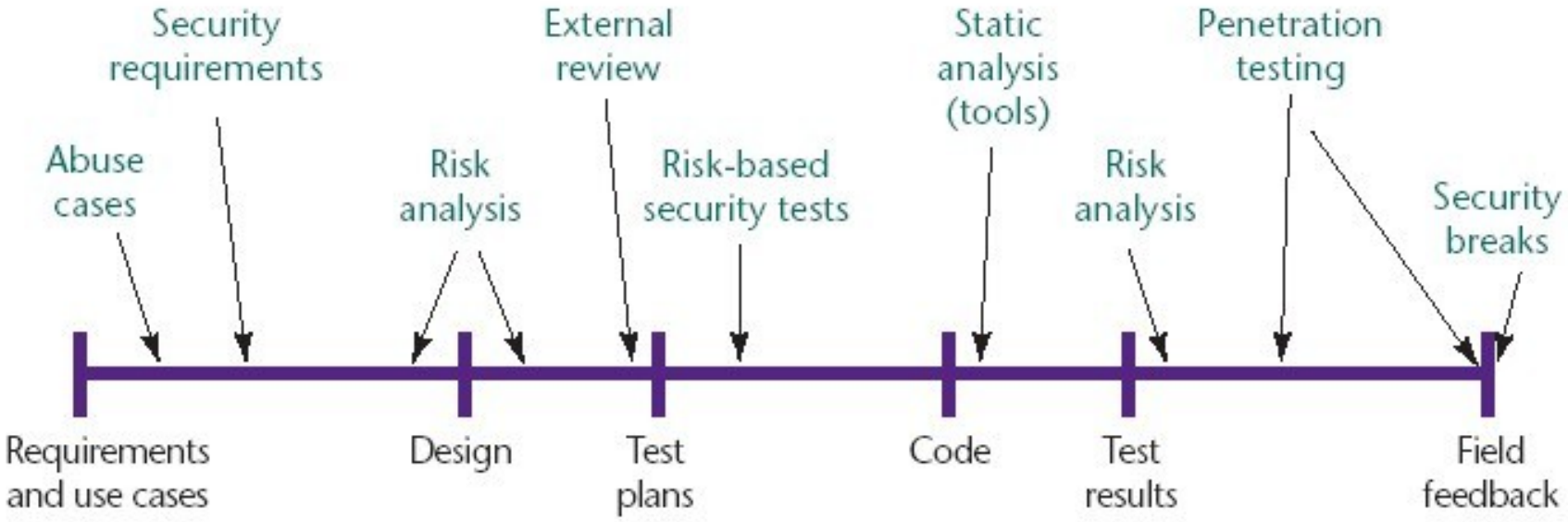
Two sides to software security: **do's** & **dont's**

- What are the methods and technologies, available to us if we want to provide security?
  - security in the software development lifecycle
  - engineering & design principles
  - security technologies

- What are the methods and technologies available to the enemy who wants to break security ?

  ie. what are the threats and vulnerabilities we're up against

Security in Software Development Life Cycle



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

Security technologies we can use

- cryptography
  - for threats related to insecure communication and storage

    (Probably adequately covered in other courses?)
- access control
  - for threats related to misbehaving users
    - eg role-based access control
- language-based security
  - for threats related to misbehaving programs
    - typing, memory-safety
    - sandboxing
  - eg Java, .NET/C#

Security technologies

- Security technologies may be provided by the infrastructure/platform an application builds on, for instance
  - networking infrastructure
    - which may eg. use SSL
  - operating system or database system
    - providing eg. access control
  - programming platform
    - for instance Java or .NET sandboxing

- Of course, software in such infrastuctures implementing security has to be secure

Software infrastructure

Applications are built on top of "infrastructure" consisting of

- operating system
- programming language/platform/middleware
  - programming language itself
    - interface to CPU & RAM
  - libraries and APIs
    - interface to peripherals
    - provider of building blocks
- other applications & utilities
  - eg database

This infrastructure provides security mechanisms,

but is also a source of insecurity

Threats & vulnerabilities

- Knowledge about threats & vulnerabilities *crucial*

- Vulnerabilities can be specific to programming language, operating system, database, the type of application... and are continuously evolving
  - we cannot hope to cover all vulnerabilities in this course
- "Fortunately", people keep making the same mistakes
  and some old favourites never seem to die,
  - esp. public enemy number 1: the buffer overflow
  and some patterns keep re-emerging

Sources of software vulnerabilities

- Bugs in the application or its infrastructure
  - ie. doesn't do what it should do
- Inappropriate features in the infrastructure
  - ie. does something that it shouldn't do
    - functionality winning over security
- Inappropriate *use* of features provided by the infrastructure.

  Main causes
  - complexity of these features
    - functionality winning over security, again
  - ignorance of developers

Topics in rest of this course

- Awareness & knowledge of vulnerabilities (don'ts)
  - general (input validation, ...)
  - specific to a kind of application (SQL injection, XSS, ...), or
  - specific to a kind of programming language (buffer overflows, ...)
- Awareness & knowledge of countermeasures (do's)

  at different points in the development lifecycle

  at level of application, programming language, or platform

  Eg security technologies (static or dynamic) such as
  - access control
  - untrusted code security
    - type-safe languages, sandboxing, code-based access control
  - runtime monitoring
  - program analyses: typing, static analysis, verification, information flow

  But beware that security software ≠ software security