Software Security

Introduction

Erik Poll

Digital Security

Radboud University Nijmegen

TU/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

  but video recordings will be in Brightspace

- 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/

Goals of this course

- How does security typically fail in software?
- Why does software often fail?

  ie. what are the underlying root causes?
- What are ways to make software more secure?

  incl. principles, methods, tools & technologies

  - incl. practical experience with some of these

Practicalities: prerequisites

- Introductory security course
  - TCB (Trusted Computing Base),
    CIA (Confidentiality, Integrity, Availability),
    Authentication ...

- 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);
      free(b);
      return(b);
}
int lets_do_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 sumOfArray(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:
  - 2 CyBok book chapters
  - my lecture notes
  - some articles

  I'll be updating this as we go along

- Some additional optional suggestions for background reading on website
- Highly recommended: the Risky.Biz podcast to keep up with weekly security news

Practicalities: form & examination

- 2-hrs lecture every week
  - read associated papers & ask questions!
- project work
  - PREfast for C++ (individual or in pairs)
  - group project (with 4 people) on fuzzing
  - group project on static analysis with Semmle
  - JML program verification for Java (6EC version only)
- written exam

Bonus point rule for project

Today

- Organisational stuff

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

Motivation

Quiz

Why can websites, servers, browsers, laptops, mobile phones, wifi access points, network routers, mobile phones, cars, pacemakers, the electricity grid, 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 is a MAJOR source of security problems and plays MAJOR role in providing security**

  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!

How do computer systems get hacked?

By attacking

- **software**

- **humans**

- **the interaction between software & humans**

- crypto
- hardware
- …

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, ...

Fairy tales

Many discussions of security begin with Alice and Bob

How can Alice communicate securely with Bob,
when Eve can modify or eavesdrop on the communication?

This is an interesting problem,
but it is *not* the biggest problem

Hard reality & the *bigger* problem

Alice's computer is communicating with *another computer*

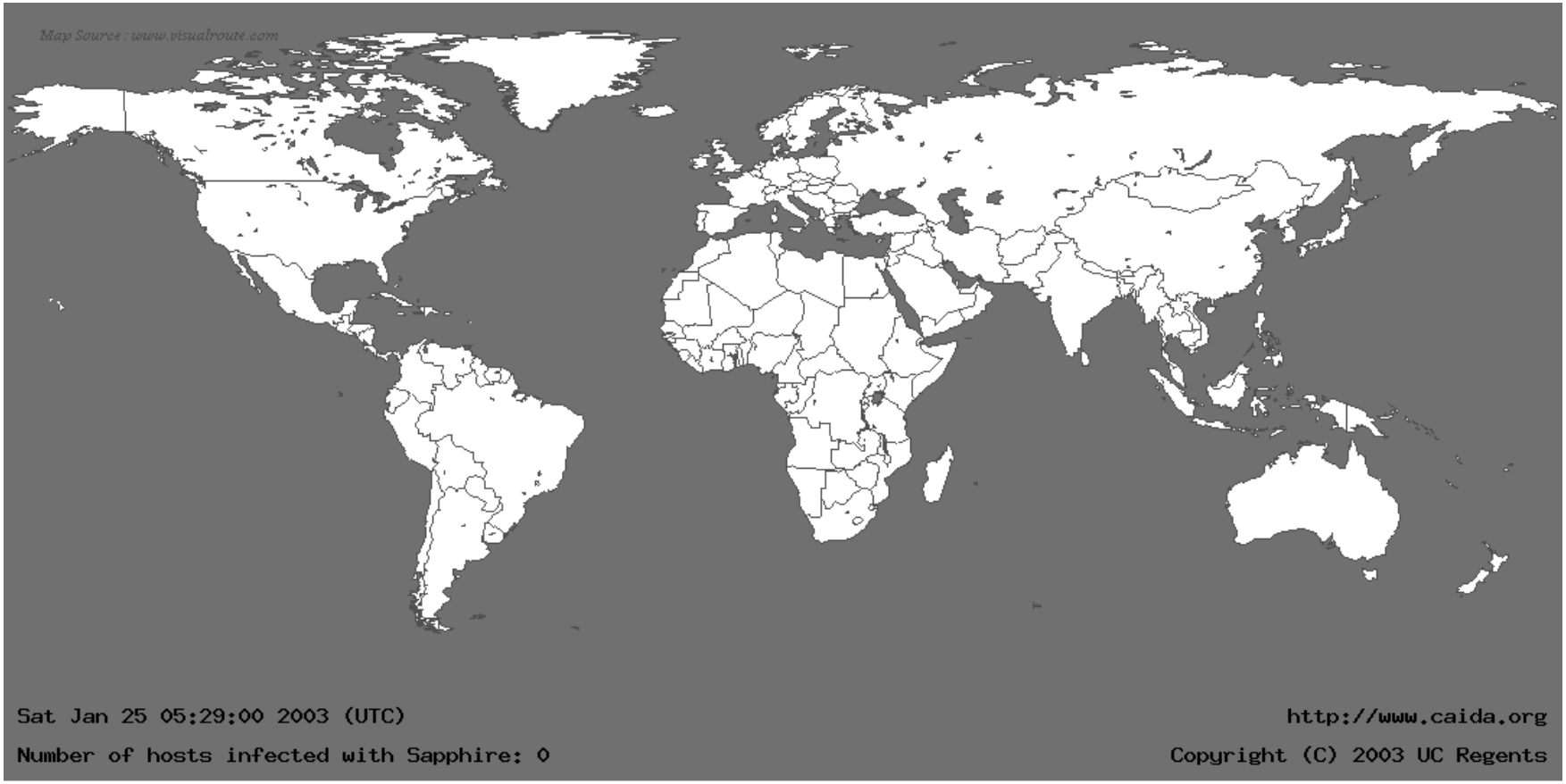
How to prevent Alice's computer from getting hacked, when it communicates with some other computer?

Or how to detect this? And then react ?

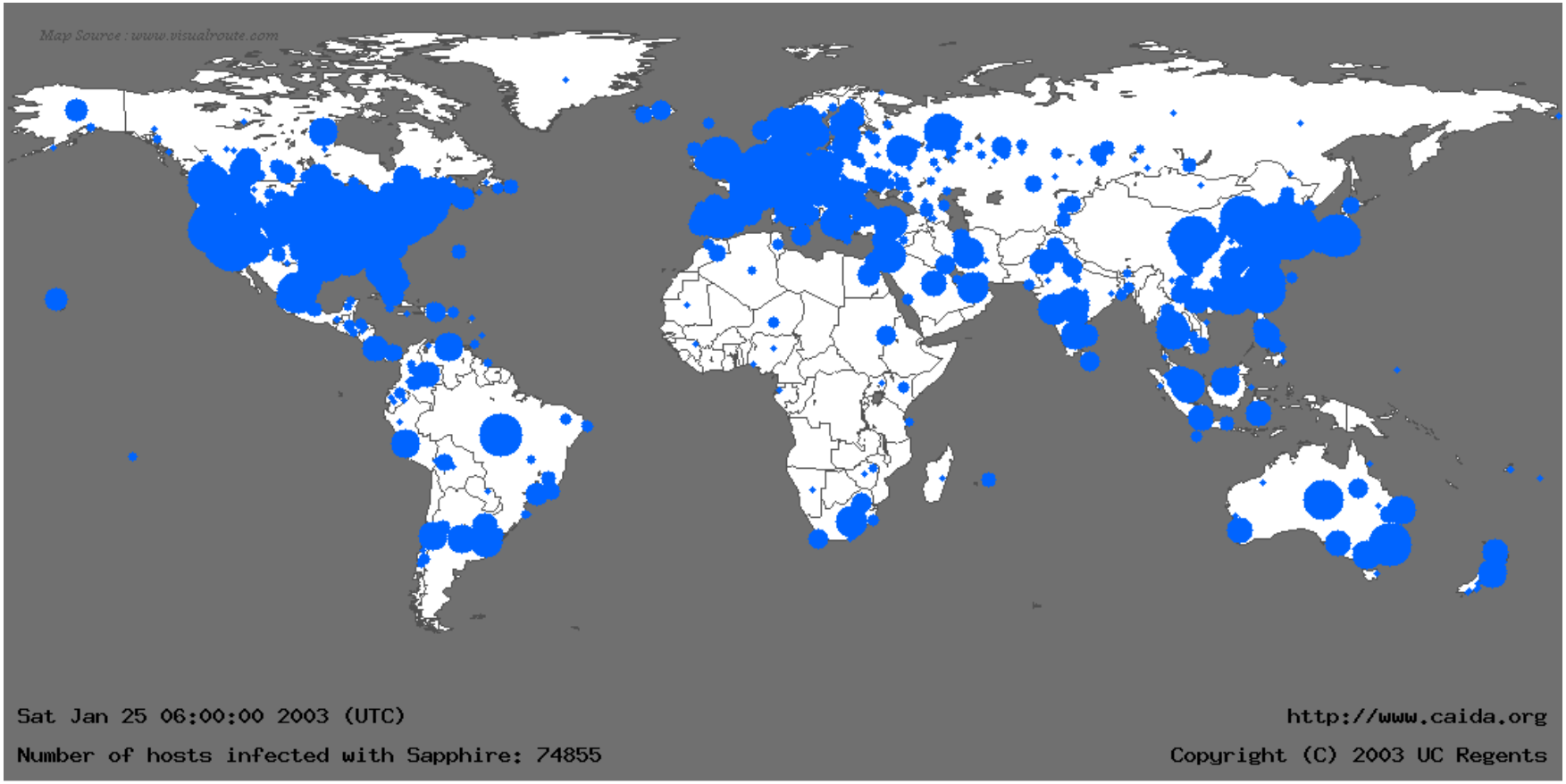
Solving the 1st problem - securing the communication - does *not* help!

The problem

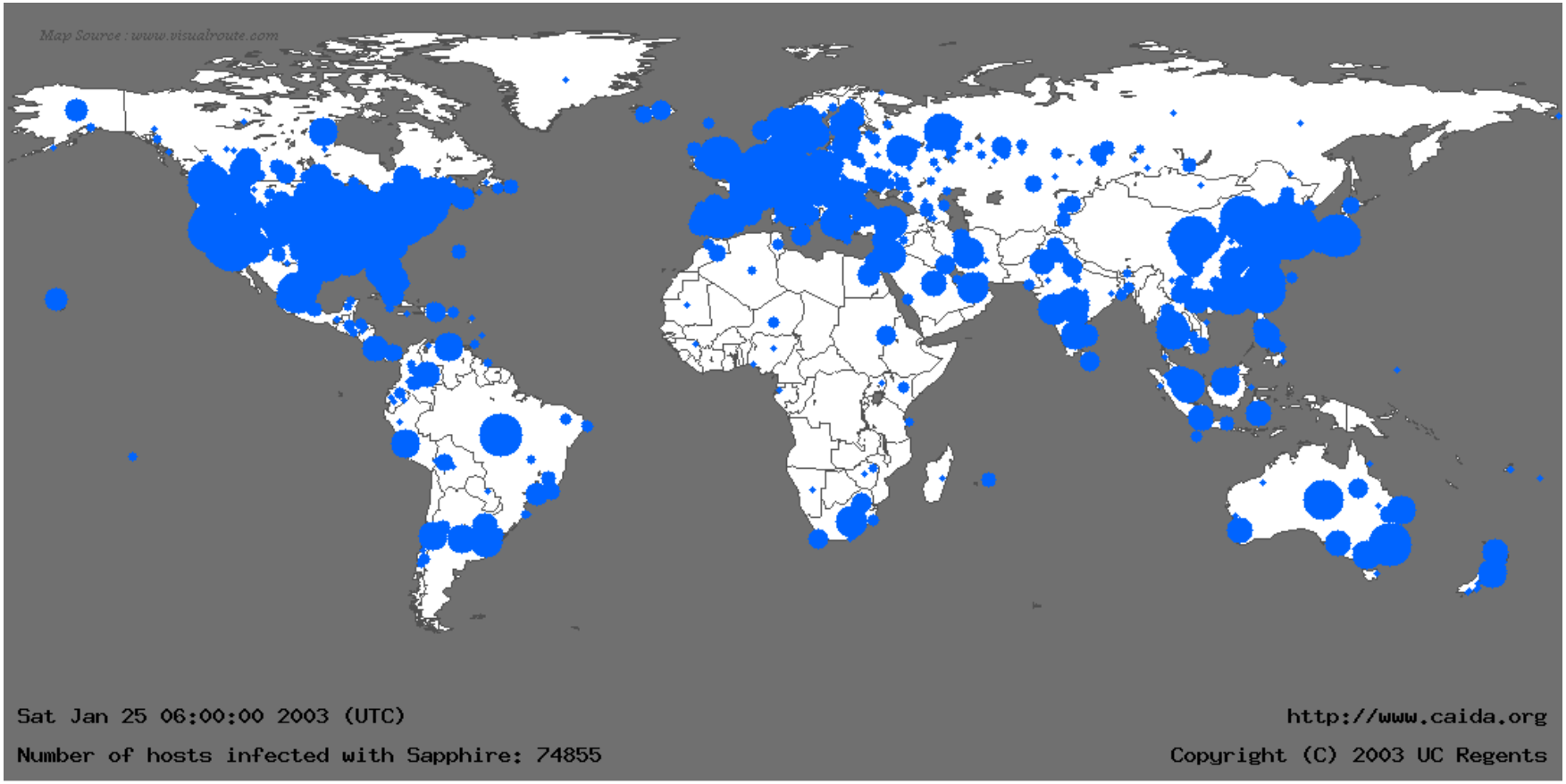
25th January 2003, 5:29 AM



25th January 2003, 6:00 AM



Slammer Worm



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

US-CERT bulletins

http://www.us-cert.gov/ncas

CVE (Common Vulnerability Enumeration)

https://cve.mitre.org/cve/

NIST's vulnerability database

https://nvd.nist.gov/vuln/search

Or subscribe to CVE twitter feed

https://twitter.com/cvenew

Changing nature of attackers

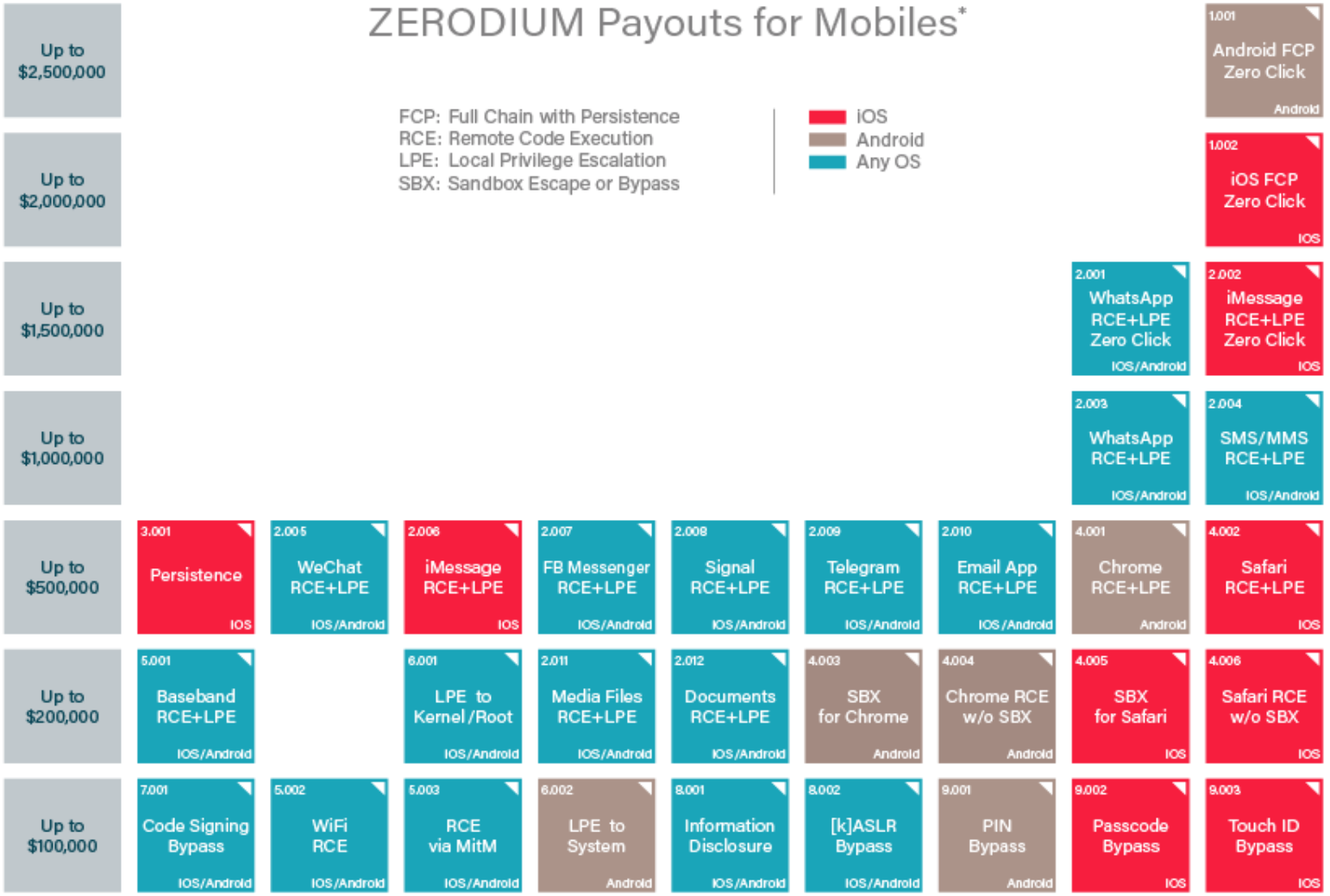
Traditionally, hackers were amateurs motivated by 'fun'

- publishing attacks for the prestige

Nowadays hackers are professional

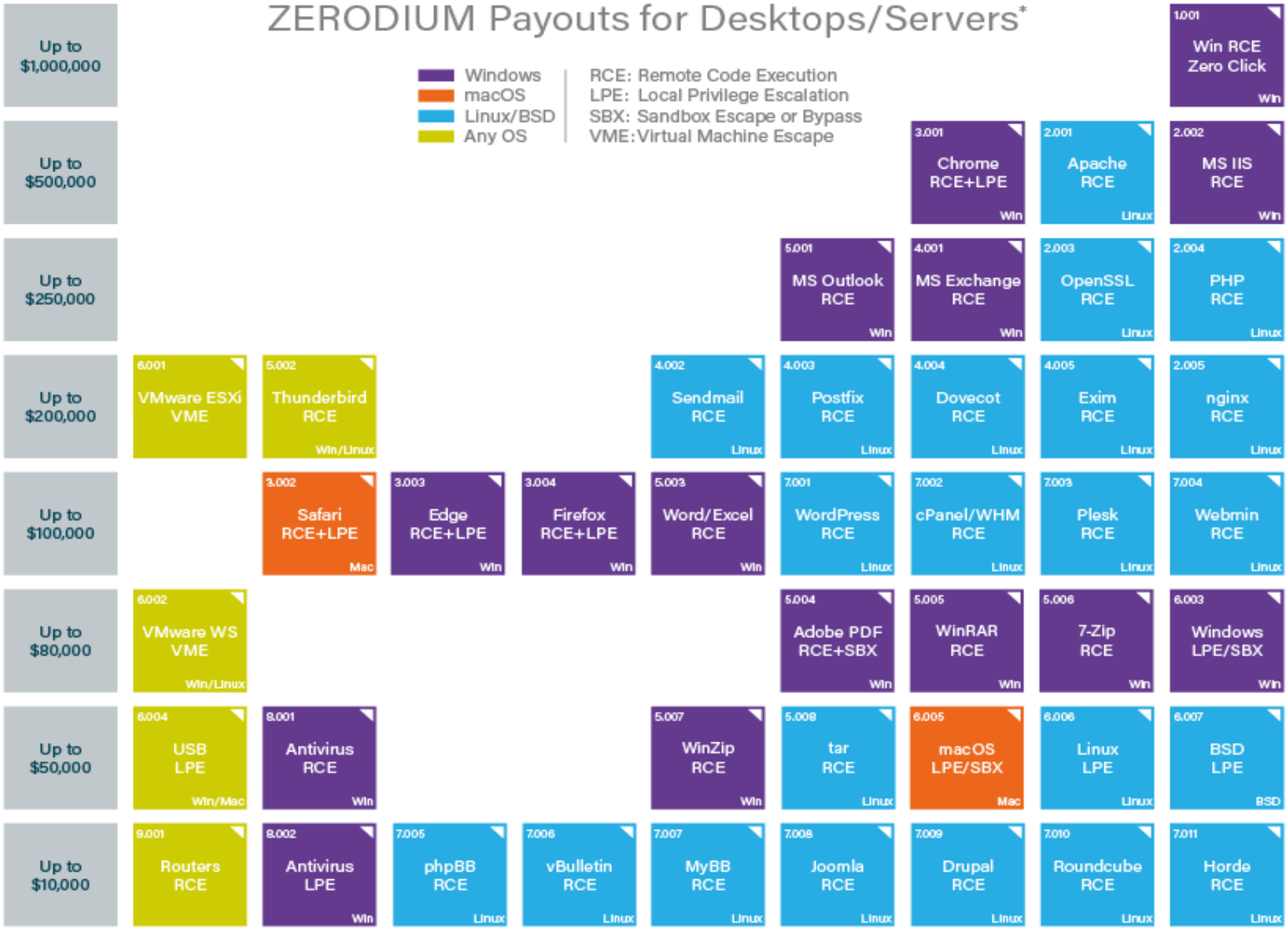
- attackers go underground
  - zero-days are worth good money
- main categories of attackers
  - (organized) criminals
    with lots of money and (hired) expertise
    Ransomware & bitcoin as important game changers
  - state actors:
    with even more money & in-house expertise

Current prices for 0days



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

Current prices for 0days



** All payouts are subject to change or cancellation without notice. 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

Root causes

Quick audience polls

- *Did you ever take a course on C(++) programming ?*
- *Were you taught C(++) as a first programming language?*
- *Did this these courses*
  - *warn about buffer overflows?*
  - *explain how to avoid them?*

Major causes of problems are

- **lack of awareness**
- **lack of knowledge**
- **irresponsible teaching of dangerous programming languages**

Quick audience poll

- *Did you ever build a web-application?*
  - *in which programming languages?*
- *Do you know the secure way of doing a SQL query in this language (to prevent 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 functionality & 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

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 JavaScript, plug-ins for Flash & Java, access to microphone, web cam, location, …
- 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.

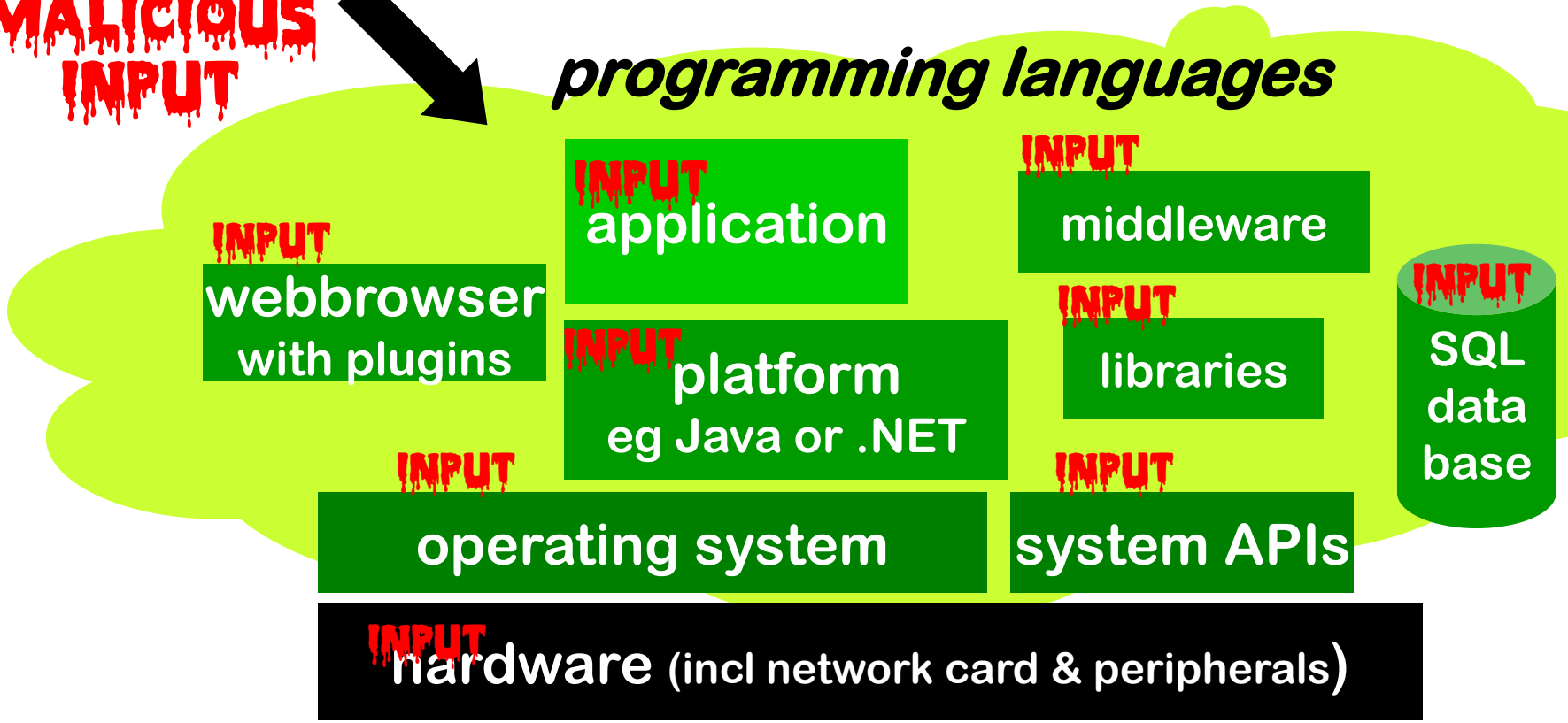
Newbies have zero chance of writing secure software unless their language is safe. ... "

[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

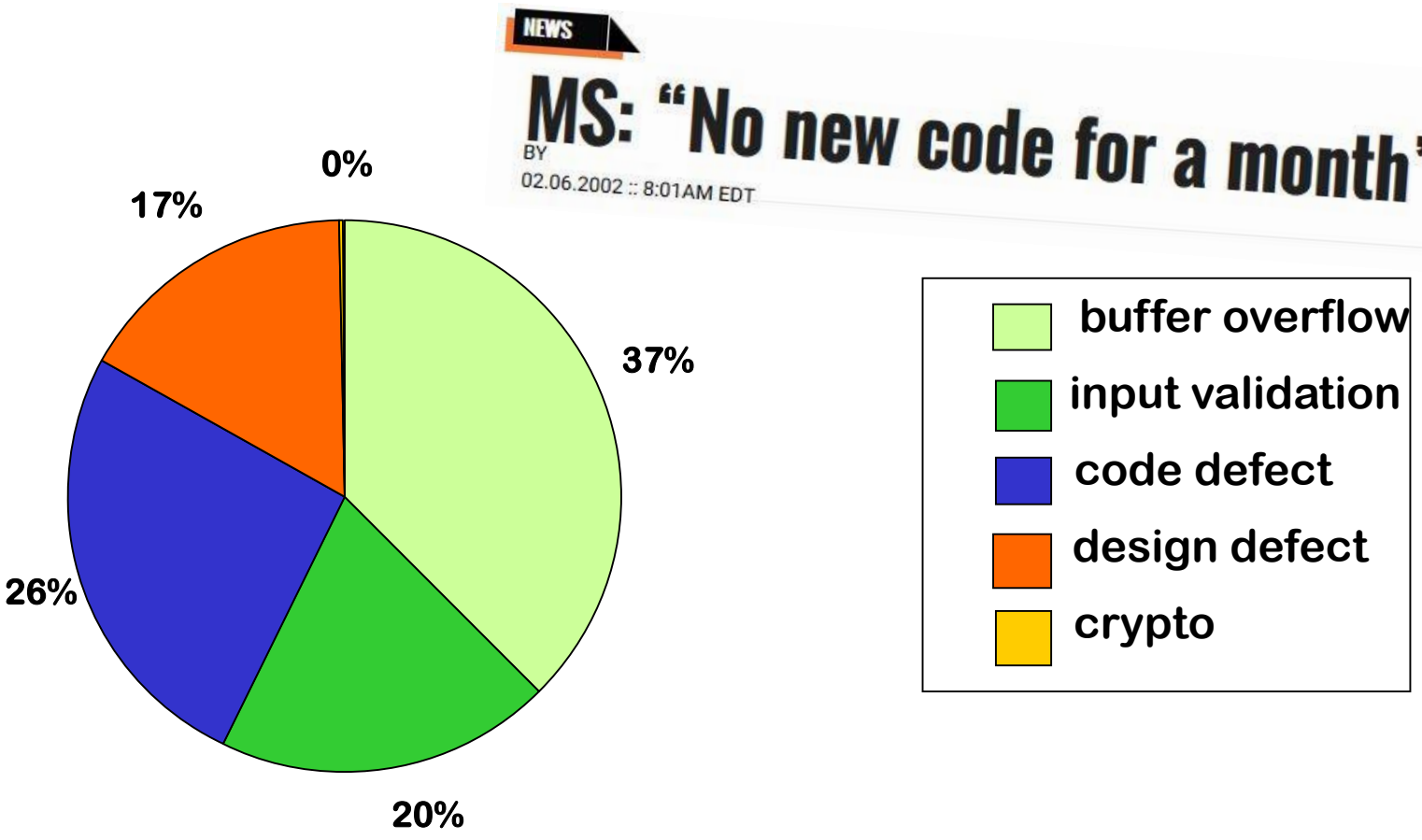
   flaws that can actually be exploited by an attacker

   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,
many security vulnerabilities become flaws

Typical software security flaws



Flaws found in Microsoft's first security bug fix month (2002)

Other useful distinctions

1. design flaws

2. implementation flaws aka bug aka code-level defects
   introduced during coding

Overall consensus:
coding bugs & design flaws equally common

Vulnerabilities also arise on other levels

3. configuration flaws
4. unforeseen consequences of the *intended functionality*
   - eg. spam
   - not a bug, but a feature!

Types of implemention aka coding flaws

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,...

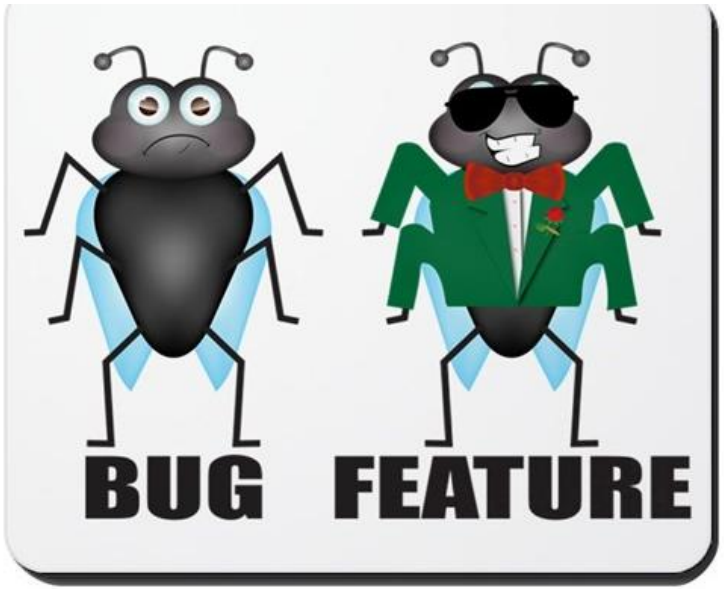
2b. (common) problems in the interaction with the underlying platform or other systems and services, eg

- buffer overflows in C(++) code
- SQL injection, XSS, CSRF,.... in web-applications
- Deserialisation attacks in many programming languages
- ...

Bug vs features, yet again

Coding flaws can be

1. bugs
   - eg buffer overflow, as discussed next week
2. (abuse of) features
   - eg SQL injection
   - unintended access to features
   - interaction / combination of features



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!

```
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

   Maybe this is a design flaw? We could decide not use signed integers..

   Root cause: IMPLICIT ASSUMPTION

<= 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

   Root cause: BROKEN ABSTRACTION

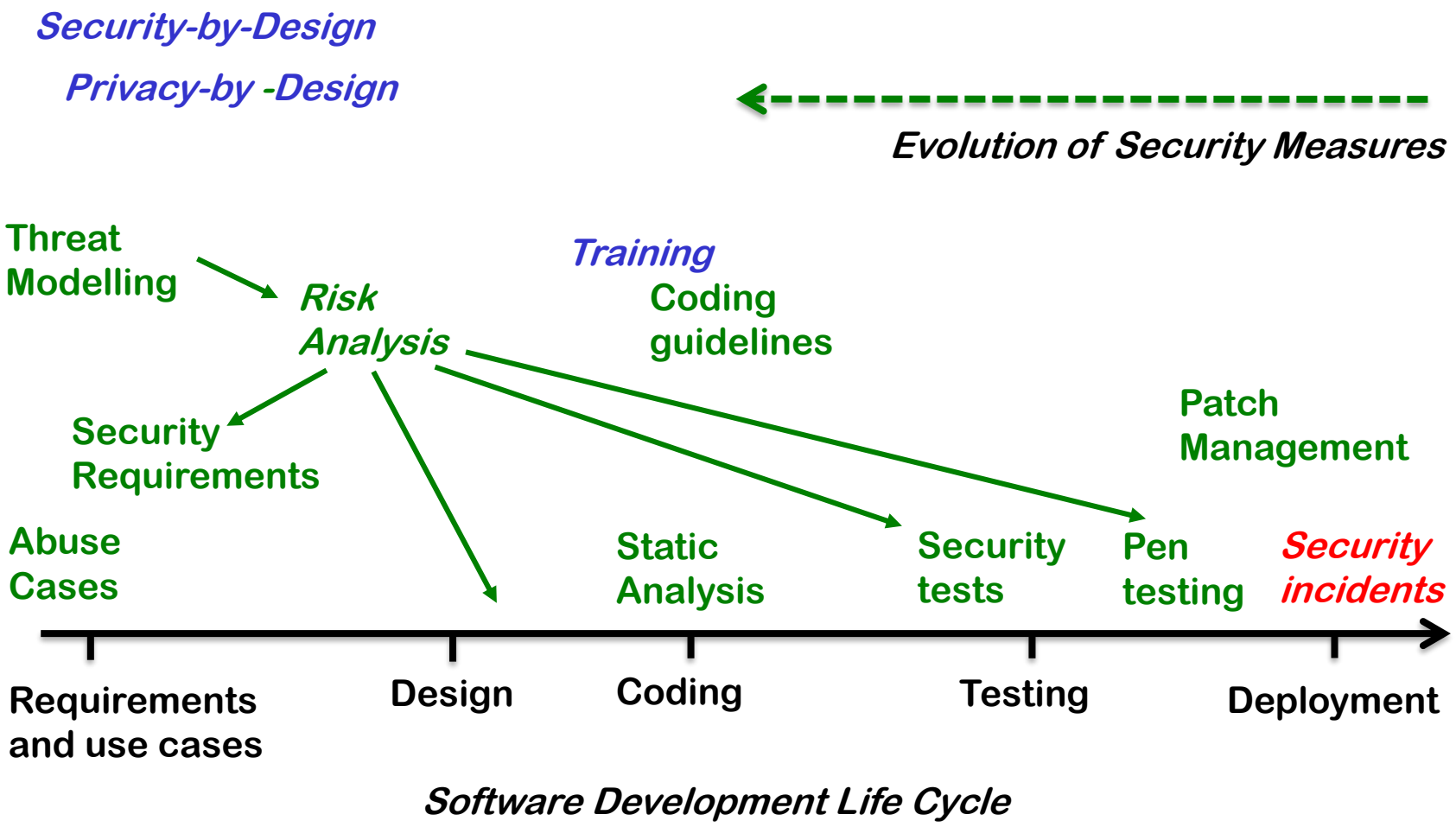
Security in the Software Development Life Cycle (SDLC)

[Material cover in CyBok chapter on Secure Software Lifecycle by Williams, see course web page]

How to improve software insecurity?

- We know how to do this!
- 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



Shifting left

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**

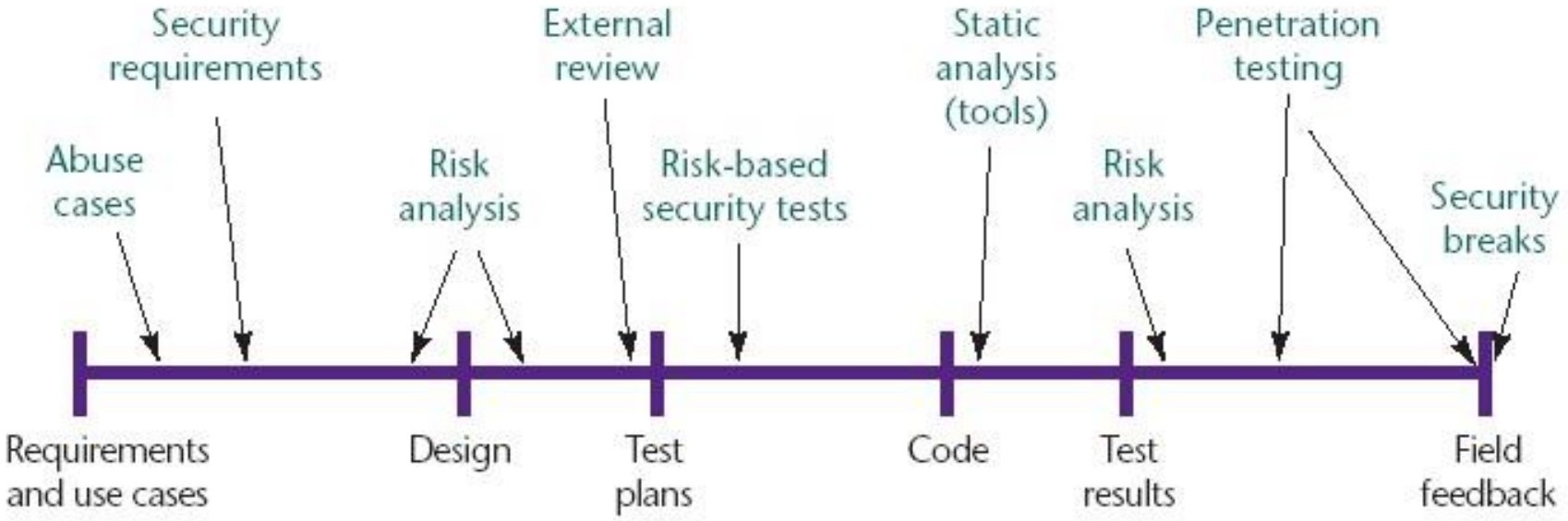
DAST, SAST

Security people keep inventing trendy new acronyms

- DAST
  - Dynamic Application Security Testing
  - ie. testing
- SAST
  - Static Application Security Testing
  - ie. static analysis
- RASP
  - Run-time Application Security Protection
  - ie. monitoring

Security in the Software Development Life Cycle

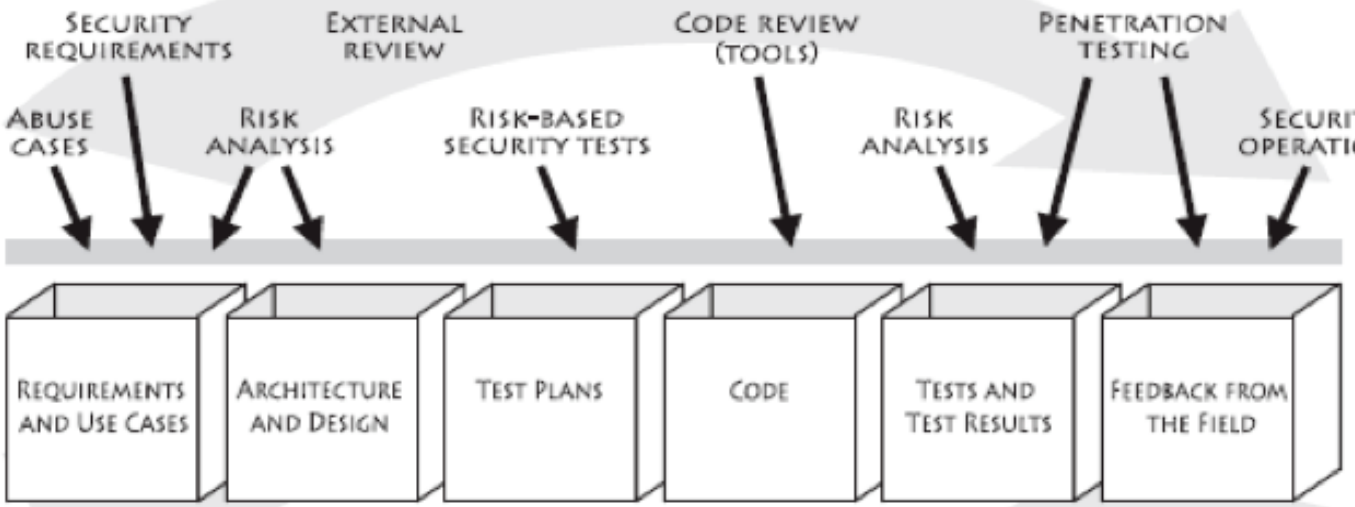
McGraw's Touchpoints



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

Security in the Software Development Life Cycle

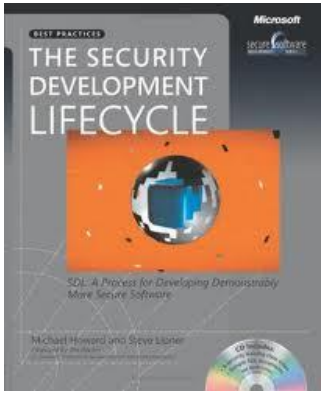
McGraw's Touchpoints



[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

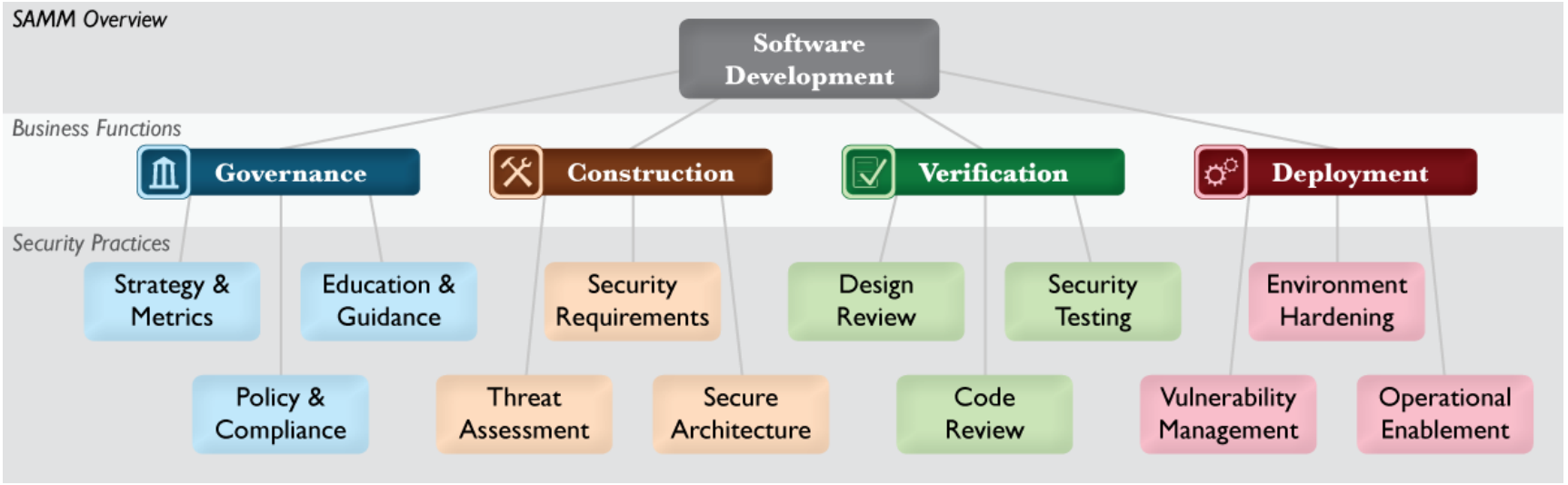


- Microsoft SDL
- OpenSAMM Software Assurance Maturity Model

  http://opensamm.org

- OWASP CLASP, Touchpoints, …

OpenSAMM's 4 business functions and 12 security practices



Microsoft's SDL Optimisation Model

BSIMM (Building Security In Maturity Model)

Framework to make compare your SSI (Software Security Initiative) with that of other companies

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/

BSIMM: comparing your security maturity

d

Threat modelling

Crucial first step in any security discussion!

1. **what are your security requirements?**
   - **Not just thinking about Confidentiality, Integrity and Availability, but also about Authentication, and not just Prevention but also about Detection and Reaction**
2. **what is your attacker model?**
   - **attack surface**
   - **attacker's motivations & capabilities**
   - **What is the TCB (Trusted Computing Base) ?**
   - **What are your security assumptions ?**

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

For you to do

- To read: CyBok chapter on Secure Software Lifecycle
  by Laurie Williams, 2019

- To do: check out recent US-CERT bulletins & CVEs

- Send me an email if you are not (yet) in Brightspace

Fundamental security concepts

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

Threat Modelling

Any discussion of security must start with inventory of

1. The stakeholders & their assets, esp. the crown jewels

1. The attacker model aka threat modelling
   - What is the attack surface?
   - What are the attack vectors the attacker can use?
   - What are the capabilities & resources of the attacker?
     script kiddies, criminals, insiders, APTs, … ?
   - Possibly also: What are the motives of the attacker?
   - For detailed analysis for whole IT infrastructure of an organisation you can use MITRE's ATT&CK framework

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

Security objectives

- **Confidentiality** unauthorised users cannot *read* information
- **Integrity** unauthorised users cannot *alter* information
- **Authentication** knowing who/what you are interacting with
- **Availability** authorised users *can access* information

  In Dutch: **BIV** = Beschikbaarheid, Integriteit, Vertrouwelijkheid
- **Non-repudiation** for **accountability**

  users *cannot deny* actions
- **Privacy**
- **Anonimity**
- …

Integrity vs Confidentiality

Integrity is nearly always way more important than confidentiality

Eg think of

- **your bank account information**
- **your medical records**
- ***all* the software you use, incl. the 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, elevation of privilege attacks
  - access control

Trusted Computing Base (TCB)

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

If any part of the TCB is compromised, we're screwed.

The attacker model and the TCB are complementary.

- We want the TCB to be as small as possible
  - Unfortunately, typically the TCB is huge, as it include the operating system, lots of third-party libraries downloaded over the internet, the compiler, the IDE, …
- TRUST is bad; we want to minimize trust
  - being TRUSTED ≠ being trustworthy
- The TCB for different security properties can be different
  - eg. making backups makes the TCB for confidentiality larger, but the TCB for availability smaller

How to realise security objectives? AAAA

- Authentication
  - who are you?
- Access control/Authorisation
  - control 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
- Detection
- Reaction
  - to recover assets, repair damage, …
  - to persecute (and hence deter) offenders

prevention vs detection & reaction

- We naturally think of prevention as way to ensure security, but detection & response are often much more important and effective
  - Eg. breaking into a house with large windows is trivial; despite this absence of prevention, detection & reaction still provides security against burglars
  - Most effective security requirement for most persons and organisations: make good back-ups, so that you can recover after an attack
- *NB don't ever be tempted into thinking that good prevention makes detection & reaction superfluous.*
- Hence important security requirements include
  - being able to do monitoring
  - having logs for auditing and forensics
  - having someone actually inspecting the logs
  - …