

Software Security

Introduction

Erik Poll

Digital Security

Radboud University Nijmegen

Admin

- If anyone thinks they should still do the 5EC version, contact me asap
- Most course material will be on
<http://www.cs.ru.nl/~erikpoll/ss>
but some things will be in Brightspace
- **Keep track of Brightspace announcements**
 - If you do not log into Brightspace regularly, have these announcements forwarded to your email

Goals of this course

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

Focus more on defence than on offense

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**

The kind of C(++) code you will see next week

```
char* copy_and_print(char* string) {
    char* b = malloc(strlen(string));
    strcpy(b, string); // copy string to b
    printf("The string is %s.", b);
    free(b);
    return(b);
}

int sum_using_pointer_arithmetic(int a[]) {
    int sum = 0;
    int *pointer = a;
    for (int i=0; i<4; i++ ){
        sum = sum + *pointer;
        pointer++; }
    return sum;
}
```

The kind of 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;
}
```

The kind of OO Java code you will see next month

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

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

Exam material

- 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

Not exam material

The **Risky.Biz** podcast to keep up with weekly security news



Join the student CTF group if you're interested in the practical side of security: ctf-ru.slack.com

Not exam material (yet...)

Next week Thursday:

online presentation by Andrew van der Stock on

The OWASP Top 10 2021

at OWASP Belgium Chapter meeting

See <https://owasp.org/www-chapter-belgium/>

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
 - project on static analysis with Semmle (individual or in pairs)
- written exam

Bonus point for group project, computed as $(\text{grade}-6) / 4$

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 **software**

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!

Fairy tales

Many discussions about 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**

The *really big* problem

Alice & her computer are communicating with *another computer*



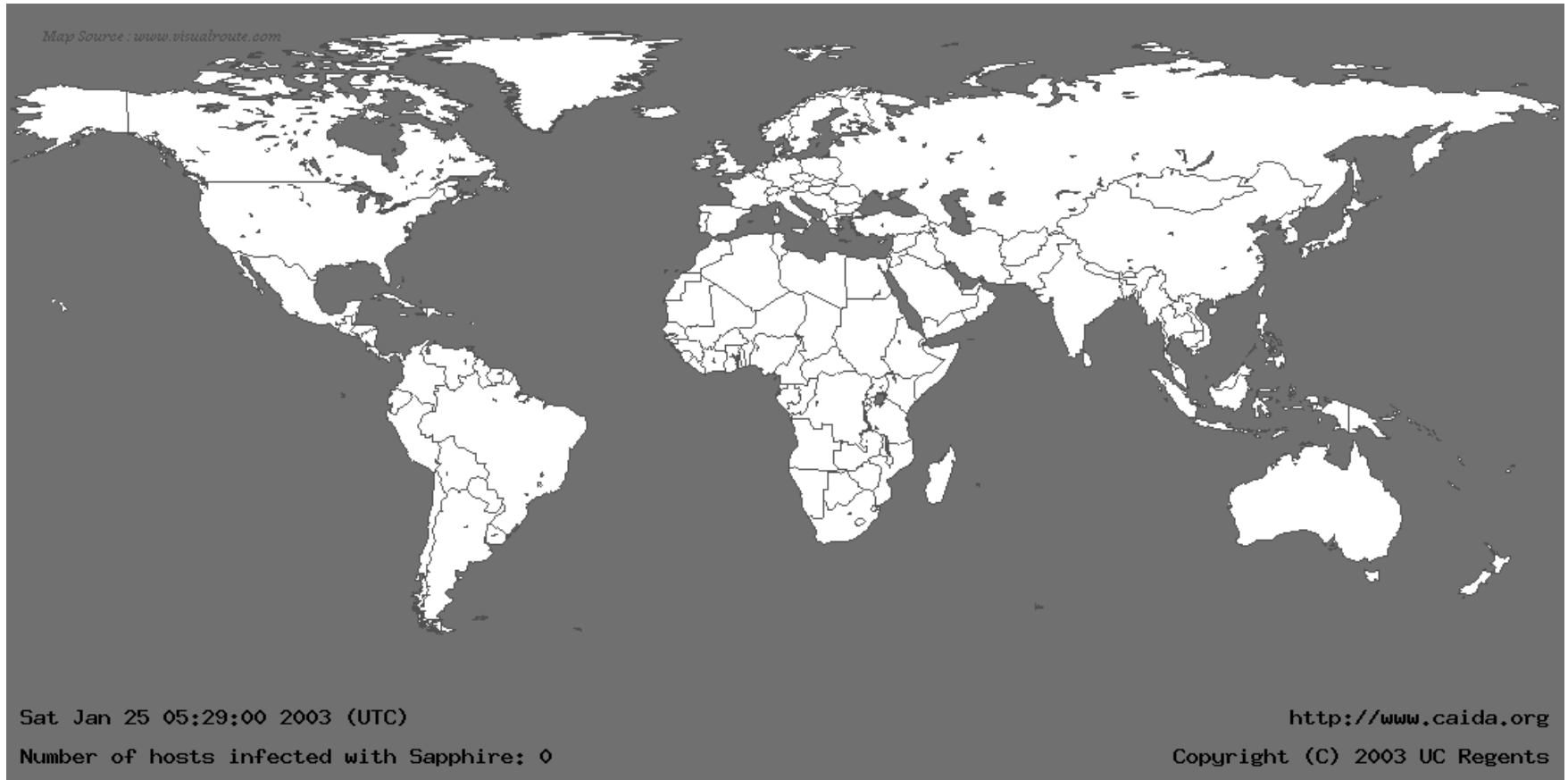
How to prevent Alice's computer from getting hacked?

Or how to detect this? And then react ?

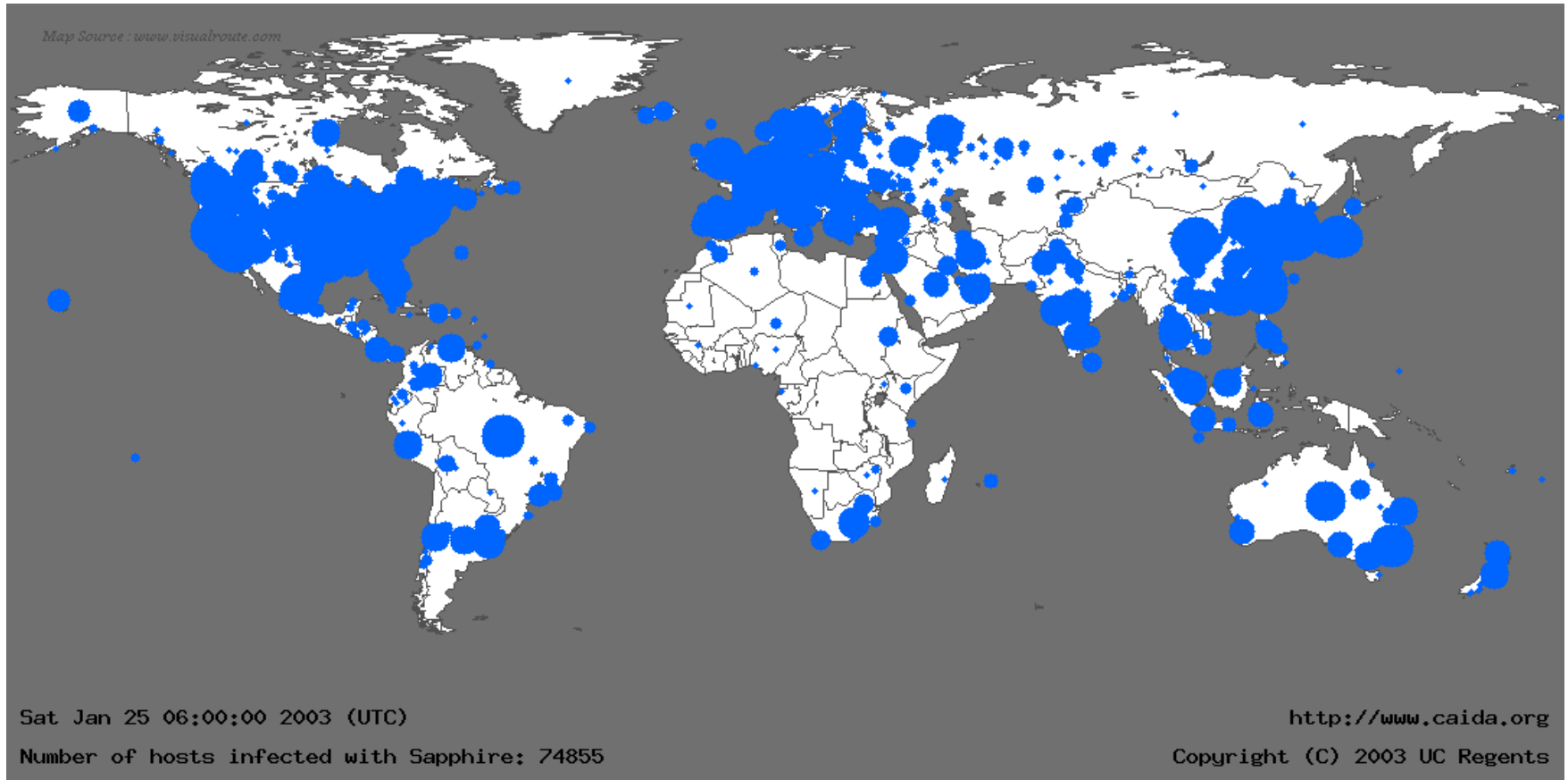
Solving earlier problem, securing the communication, does *not* help!

The problem

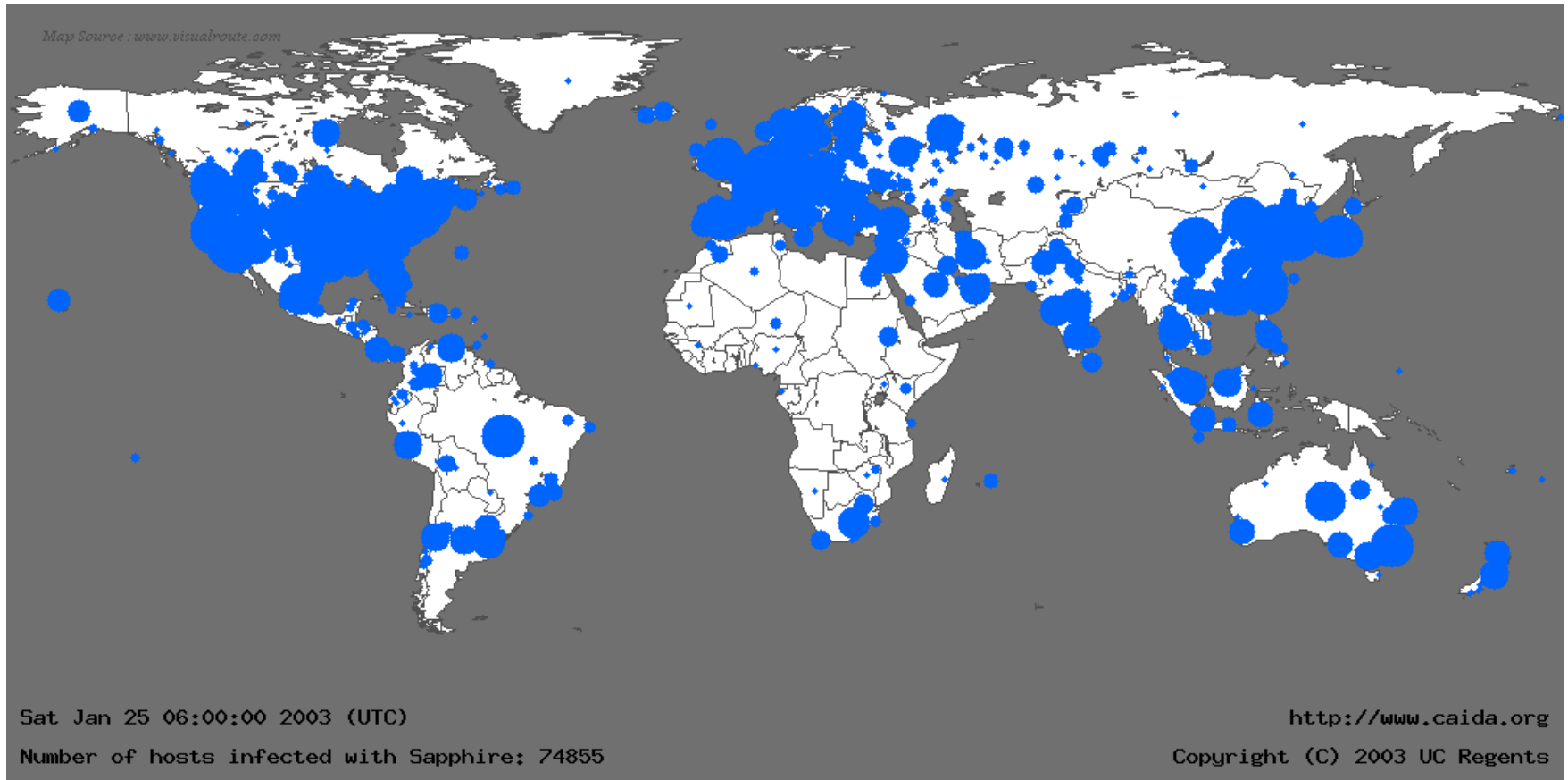
25th January 2003, 5:29 AM



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

<https://us-cert.cisa.gov/ncas/bulletins>

CVE (Common Vulnerability Enumeration)

<https://cve.mitre.org/cve/>

NIST's vulnerability database

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

Or subscribe to the CVE twitter feed

<https://twitter.com/cvenew>

Changing nature of attackers

Traditionally, hackers were **amateurs motivated by 'fun'**

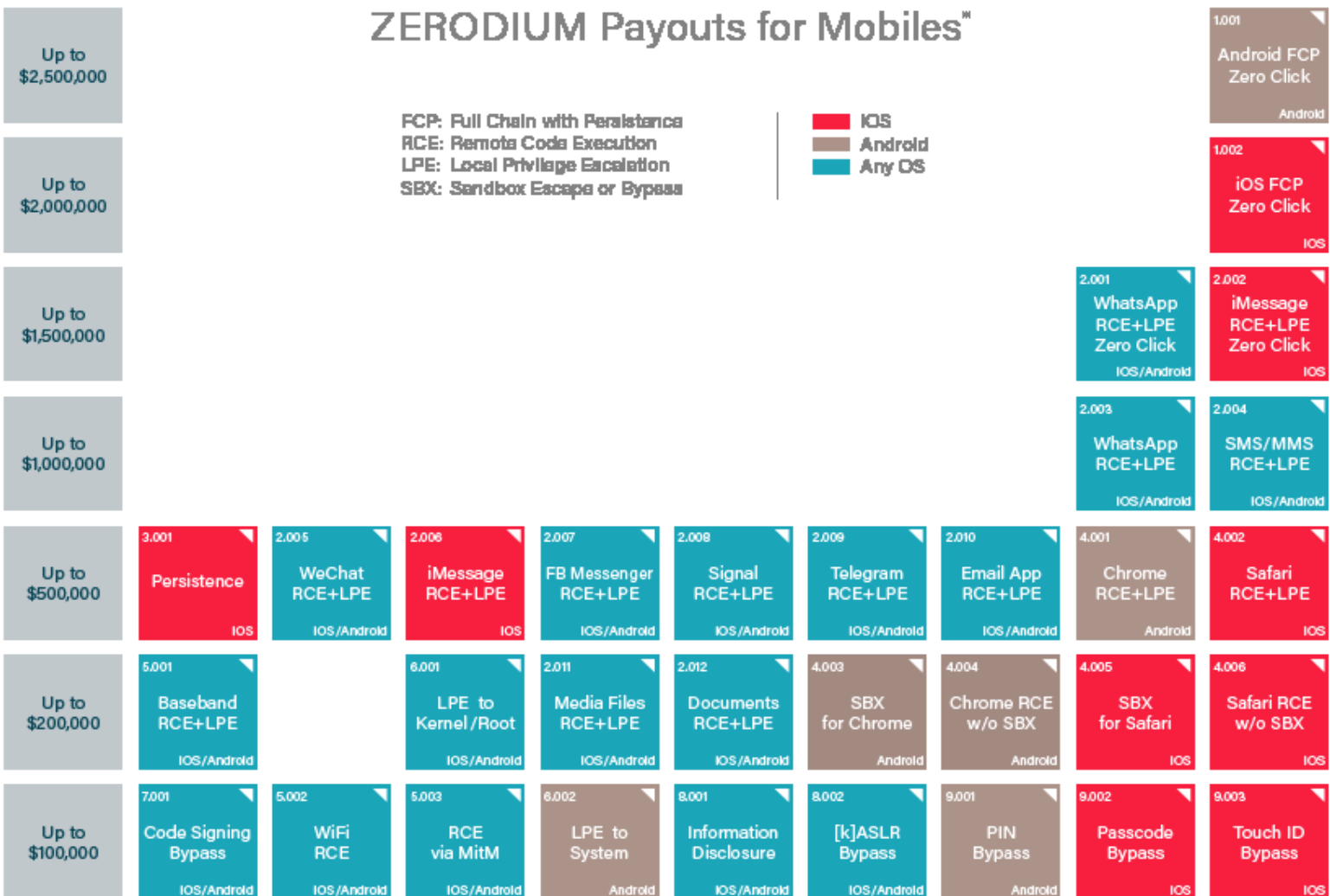
- by **script kiddies** & more **skilled hobbyists**
- NB if you like that, join the RU-CTF team!

Nowadays hackers are **professional:**

- **cyber criminals**
 - with lots of money & (hired) expertise
 - Important game changers: **ransomware** & **bitcoin**
- **state actors**
 - with even more money & in-house expertise
- **hackers for hire**
 - like NSO group, Zerodium, ...

Prices for 0days

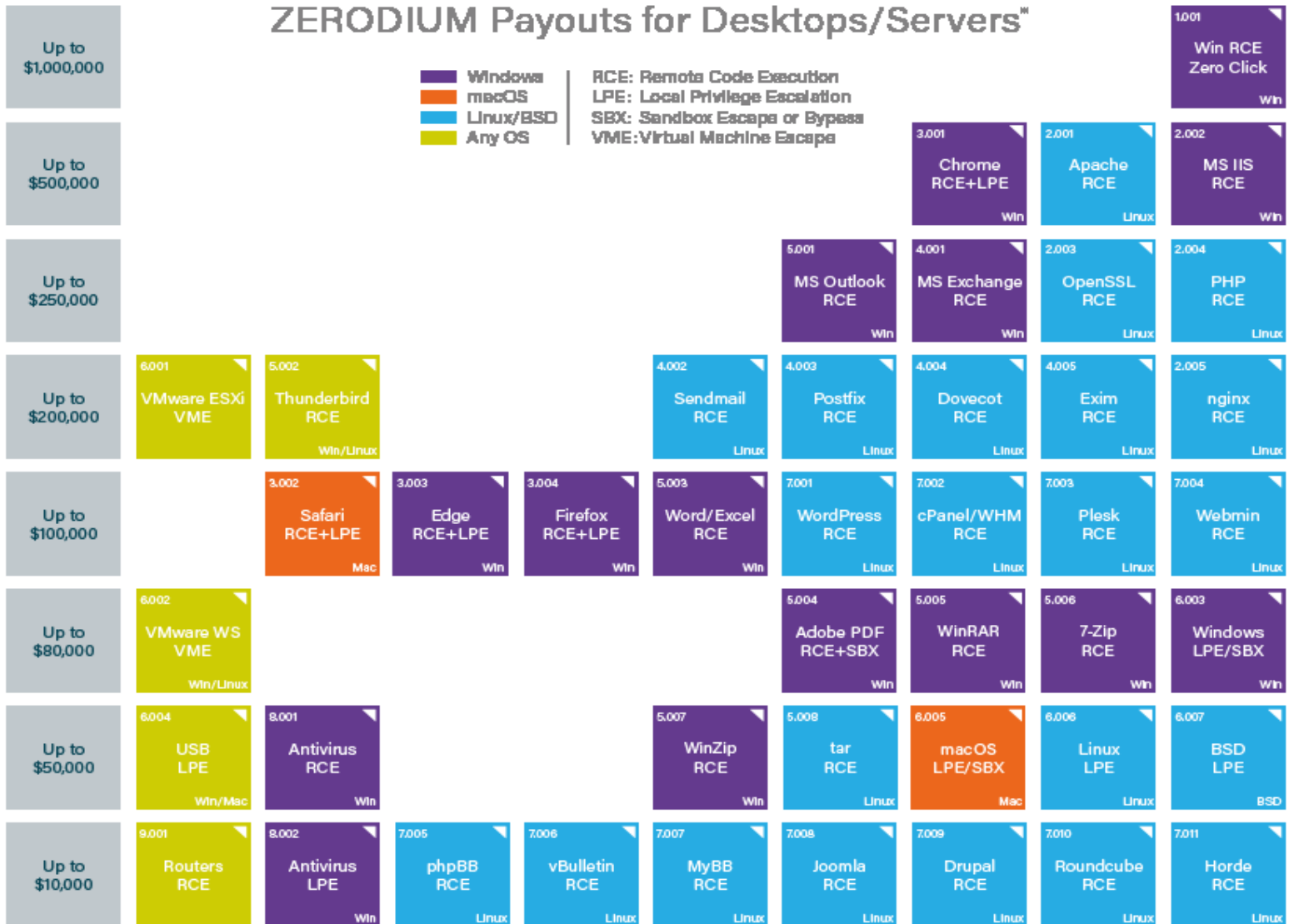
ZERODIUM Payouts for Mobiles™



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

Prices for 0days

ZERODIUM Payouts for Desktops/Servers™



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

Software security: crucial facts

- *There are no silver bullets!*

Firewalls, crypto, or special security features do not magically solve all problems

- “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

- like quality
- or maybe: property of the ongoing process?

- Security should be - but hardly ever is - integral part of the design, right from the start

security software ≠ software security

Adding **security software** can make a system more secure

i.e. **software specifically for security**, such as

- **network-level protection** with **TLS, IPSEC, firewall, VPN, ...**
- **AV** (AntiVirus), **WAF** (Web Application Firewall)
- **access control**, with eg **2FA, logging, monitoring, ...**
- **NIDS** (Network Intrusion Detection System)
- **EDR** (Endpoint Detection and Response)
- ...

BUT

All **software must be secure**, not just the security software

- That buffer overflow in your PDF viewer will still kill you...
- Adding security software may add software bugs and make things less secure:

Check out <https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=firewall>

<https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=VPN>

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 SQLi)?*

Major causes of problems are

- lack of awareness
- lack of knowledge

More root causes

Security is always a **secondary concern**

- **primary goal** is to **provide functionality & services**;
 - **managing** associated **risks** is a secondary concern
-
- There is often a trade-off/conflict between
 - security
 - functionality , convenience , speed , ...where security typically loses out
 - Users are likely to complain about missing or broken functionality, but not about insecurity

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 and Web APIs to access 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."

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

More root causes: Weakness in depth

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

MALICIOUS INPUT



programming languages

INPUT
application

INPUT
middleware

INPUT
webbrowser
with plugins

INPUT
platform
eg Java, .NET
or JavaScript

INPUT
libraries

INPUT
SQL
data
base

INPUT
operating system

INPUT
system APIs

INPUT
hardware (incl network card & peripherals)

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 complex languages, using large complex APIs, and running on complex platforms

Types of software security problems

Weaknesses vs vulnerabilities

Terminology can be messy & confusing

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

Important distinction:

1. (potential) security weaknesses

Things that could go wrong & could be better

2. (real) security vulnerabilities

Flaws that can actually be exploited by an attacker

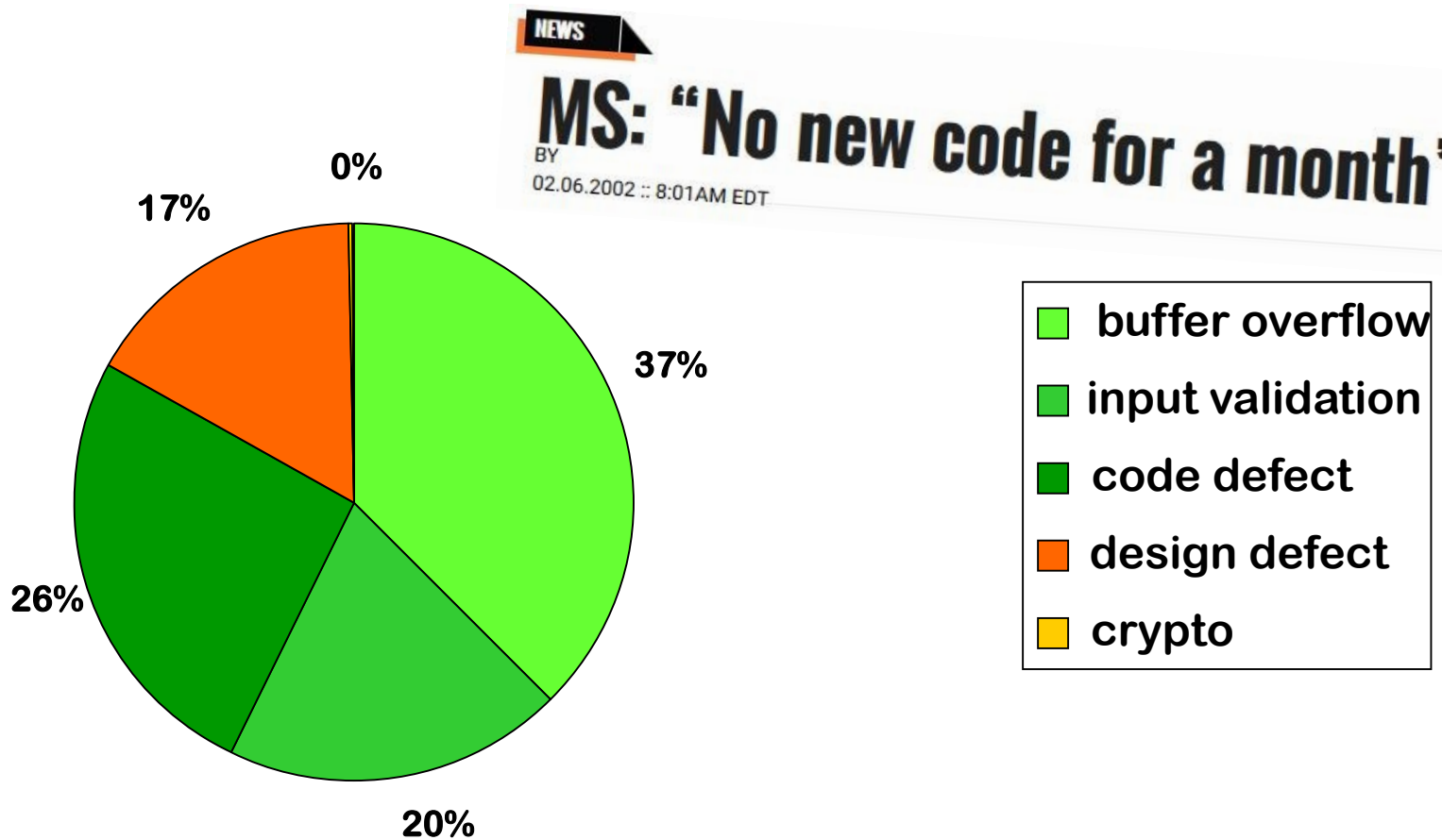
Requires flaw to be

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 on my laptop,
many vulnerabilities become weaknesses*

Typical software security flaws



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

'Levels' at which security flaws can arise

1. Design flaws
introduced *before* coding
2. Implementation flaws aka bugs aka code-level defects
introduced *during* coding

As a rule of thumb, coding & design flaws equally common

Vulnerabilities can also arise on other levels

3. Configuration flaws
4. Unforeseen consequences of the *intended functionality*
 - eg. **spam**: not enabled by flaws, but by **features**!

Types of implementation flaws

2a. Flaws that can be understood by looking at program itself

Eg. **typos**, **< instead of <= ...**, **mistake in the program logic with wrongly nested if-statements**, ...

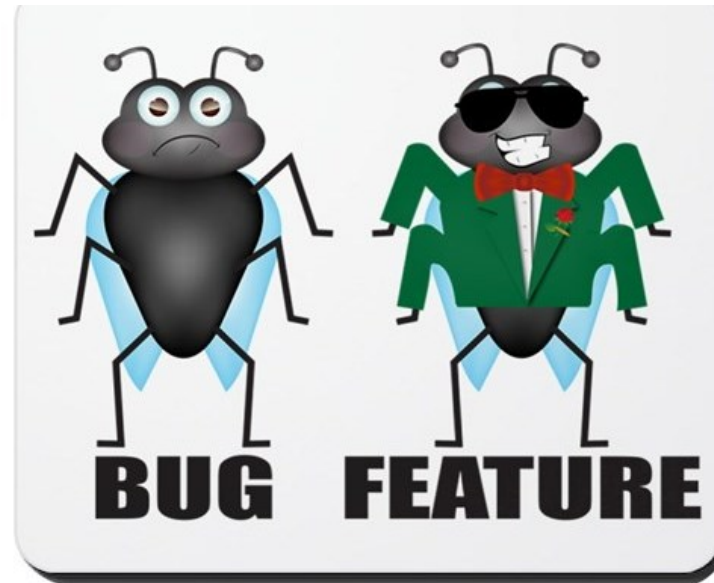
2b. Problems in the interaction with the underlying platform or other systems and services, eg

- **memory corruption** in **C(++) code**
- **SQL injection** in program that uses an SQL database
- **XSS, CSRF, SSI, XXE,** in **web-applications**
- **Deserialisation attacks** in **many programming languages**
- ...

Bug vs features, yet again

Attacks can not only exploit **bugs**, but also **features**

Eg: SQL injection uses a **feature** of the back-end database



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;
```

<= should be >=

```
void decreaseBankBalance(int amount)
{ if (balance <= amount)
    { balance = balance - amount; }
  else { println("Insufficient funds\n"); }
}
```

what if amount
is negative?

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

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 Laurie Williams, see course web page]

How can we make software secure?

We do *not* know how to do this!

Even if we formally verify software, we may

- miss security properties that need to be verified
- make implicit assumptions
- overlook attack vectors
- ...

How can we make software more secure?

We *do* know how to do this!

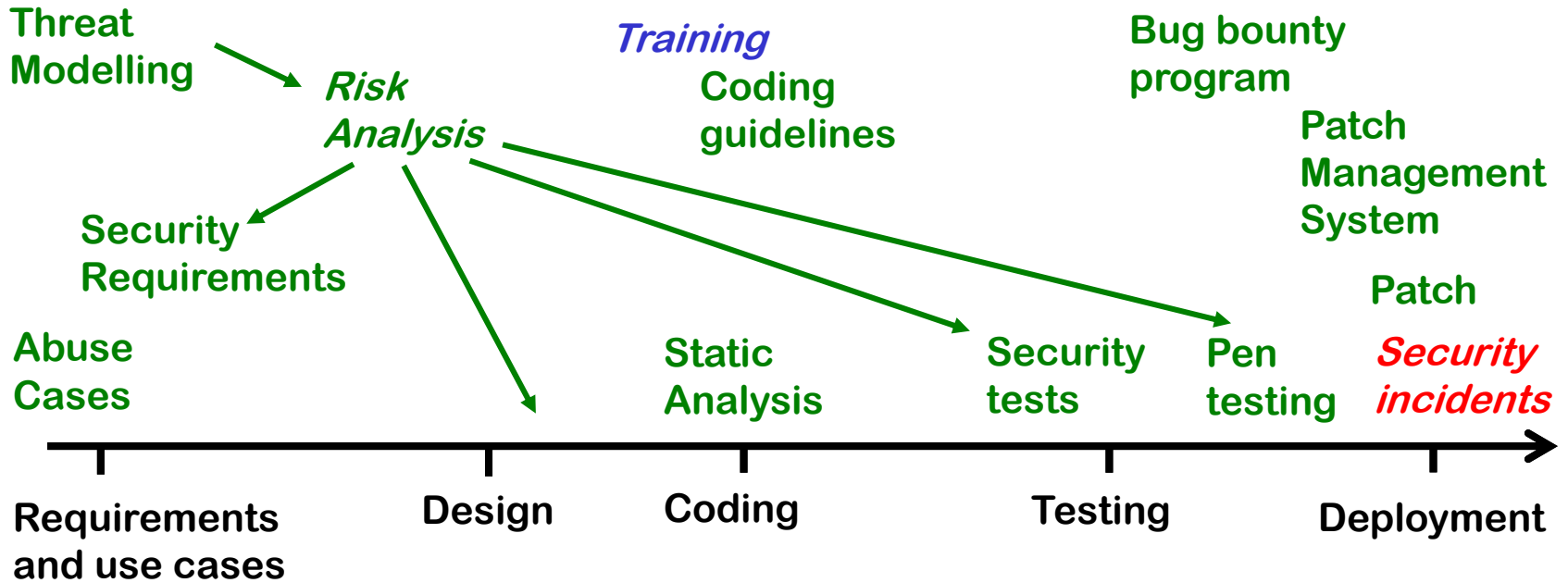
- Knowledge about standard mistakes is crucial
 - These depends on programming language, “platform”, APIs/technologies used, type of application
 - There is LOTS of info available on this nowadays
- But this is not enough: security to be taken into account from the start, *throughout* the software development life cycle
 - Several ideas, best practices, methodologies to do this

Security in Software Development Lifecycle

Security-by-Design

Privacy-by-Design

← Evolution of Security Measures



“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 or 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

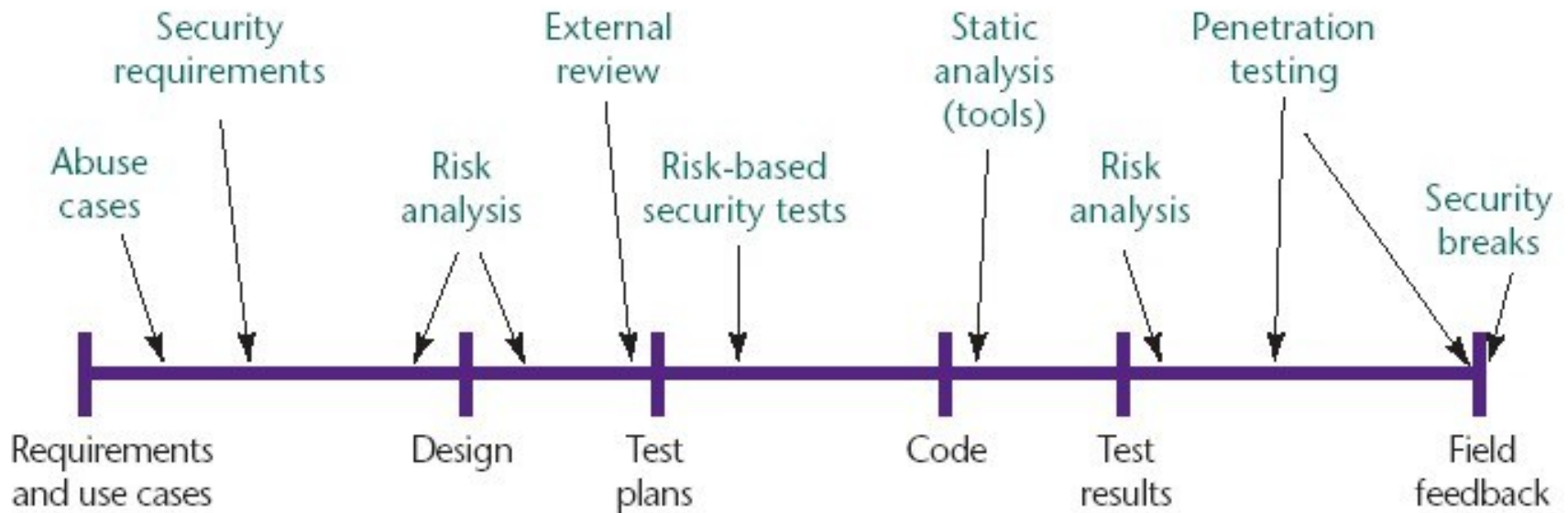
DAST, SAST, RASP

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

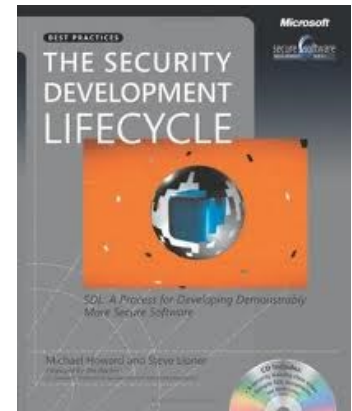
McGraw's Touchpoints



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

Methodologies for secure software development

- Gary McGraw's Touchpoints
- **Microsoft SDL**
with extension for **Secure DevOps (DevSecOps)**
- **Open SAMM** (Software Assurance Maturity Model)
- **OWASP SAMM**
- **BSIMM**
- **Grip op SSD** (Secure Software Development)
Ongoing initiative by Dutch government organisations
<https://www.cip-overheid.nl/en/category/products/secure-software/>
- ...



These come with best practices, checklists, methods for assessments, roadmaps for improvement, ...

Microsoft's SDL Optimisation Model

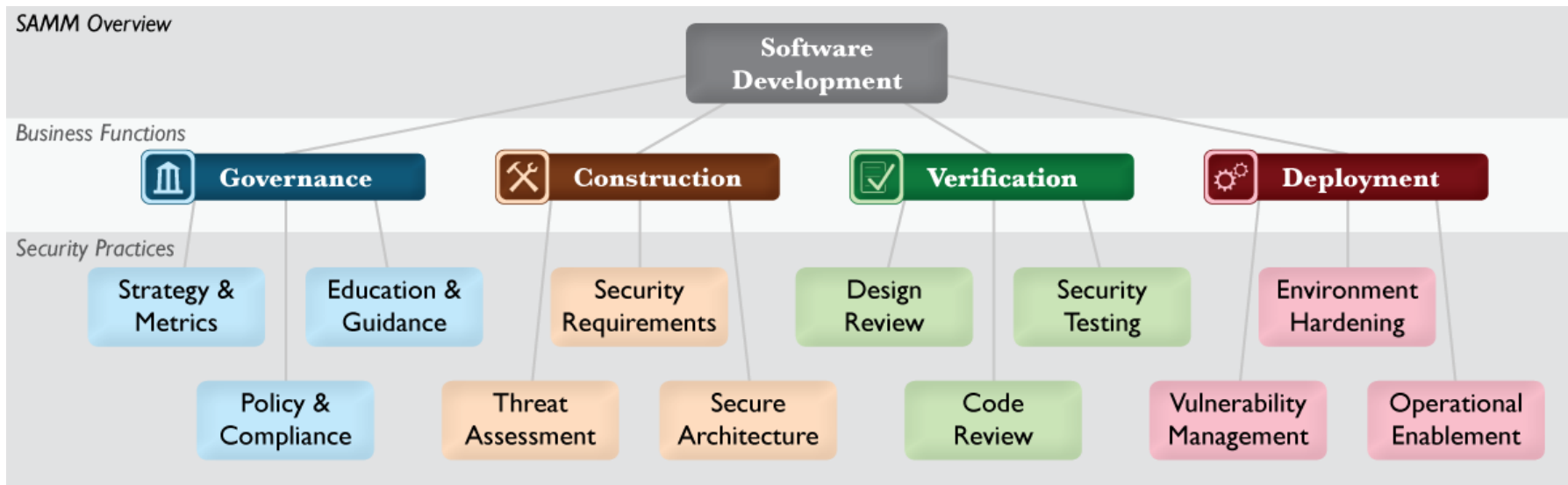
The four security maturity levels of the SDL Optimization Model



The five capability areas of the software development process



With 4 business functions and 12 security practices



BSIMM (Building Security In Maturity Model)

Framework to compare your software security efforts with other organisations

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

<https://www.bsimm.com/framework/>

BSIMM: comparing your security maturity



But first...

Crucial first steps in any security discussion!

1. What are your **security requirements**?

What does it mean for the system to be secure?

2. What is your **attacker model**?

Against what does the system have to be secure?

- **Attack surface / attack vectors**
- **Attacker's motivations & capabilities**
- **What are your security assumptions ?**
 - Including: what is the **TCB (Trusted Computing Base)** ?

Any discussion of security without answering these questions is *meaningless*

Aka **threat modelling** using eg **Microsoft STRIDE** or **MITRE ATT&CK**

Security requirements

a) 'This application cannot be hacked'

- generic default requirement 😊
- vague & not actionable ☹️

b) More specific security requirements

- In terms of **Confidentiality, Integrity and Availability (CIA)**
- Or, usually better, in terms of **Access Control**
(i.e. **Authentication & Authorisation**)

and not just **Prevention**

but also about **Detection** and **Reaction/Response**

For you to read & do

1. To read: CyBok chapter on **Secure Software Lifecycle** by Laurie Williams

2. To do: check out
 - a) the CVEs in latest US-CERT bulletin
 - b) recent CVEs for the web-browser & PDF viewer you are using
 - c) how some of the CVSS score was determined for some of these CVEs