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

- Friday Sept 20 : Master BBQ
- Tuesday Sept 24: discussion with Dick Schoof of AIVD
- Monday Oct 7, 17:00: Thalia PokéCTF

  https://thalia.nu/events/495/

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),
    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);
   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 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:

  articles, 2 book chapters and lecture notes

  - see links on webpage
  - I'll be updating this as we go along

- Some additional optional suggestions for bac incl. books and web-sites

  - Recommended: the Risky.Biz podcast to keep up with weekly security news

RISKY.BIZ

It's a jungle out there

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 fuzzing
  - group project on web-application code analysers

- 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
- The solution to the problem
- Security concepts

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 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 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!

How do computer systems get hacked?

By attacking

- software

```
require( TEMPLATEPATH.DS."yjsgcore/yjsg_stylesw.php");
$renderer     = $document->loadRenderer( 'module' );
$options      = array( 'style' => "raw" );
$module       = JModuleHelper::getModule( 'mod_menu' );
$topmenu      = false; $subnav = false; $sidenav = false;
Main Menu
if ($default_menu_style == 1 or $default_menu_style== 2) {
        $module->params = "menutype=$menu_name\nshowAllChildren=1\nclass_sfx=menu_
        $topmenu = $renderer->render( $module, $options );
        $menuclass = 'horiznav';
        $topmenuclass ='top_menu';
elseif ($default_menu_style == 3 or $default_menu_style== 4) {
        $module->params = "menutype=$menu_name\nshowAllChildren=1\nclass_sfx=menu
        $topmenu = $renderer->render( $module, $options );
        $menuclass = 'horiznav_d';
        $topmenuclass ='top_menu_d';
SPLIT MENU  NO SUBS
elseif ($default_menu_style == 5) {
        $module->params = "menutype=$menu_name\nstartLevel=0\nendLevel=1\nclass_sfx
        $topmenu = $renderer->render( $module, $options );
        $menuclass = 'horiznav';
        $topmenuclass ='top_menu';
```

- humans

- the interaction between software & humans

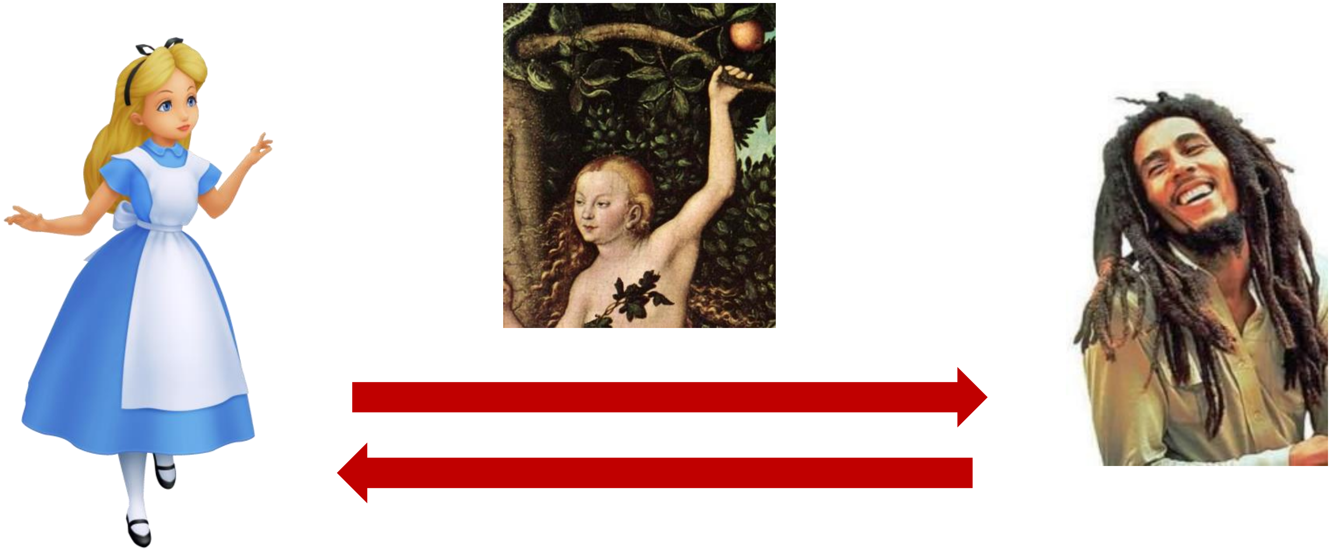
- crypto
- hardware
- …

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

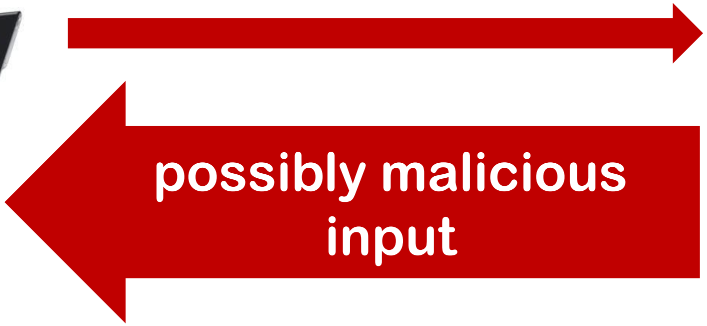


How can Alice communicate securely with Bob,
when Eve can modify or eavesdrop on the communication?

This is an interesting problem,
but it is *<u>not</u>* the biggest problem

Hard reality & the *bigger* problem

Alice’s computer is communicating with *another computer*



possibly malicious input

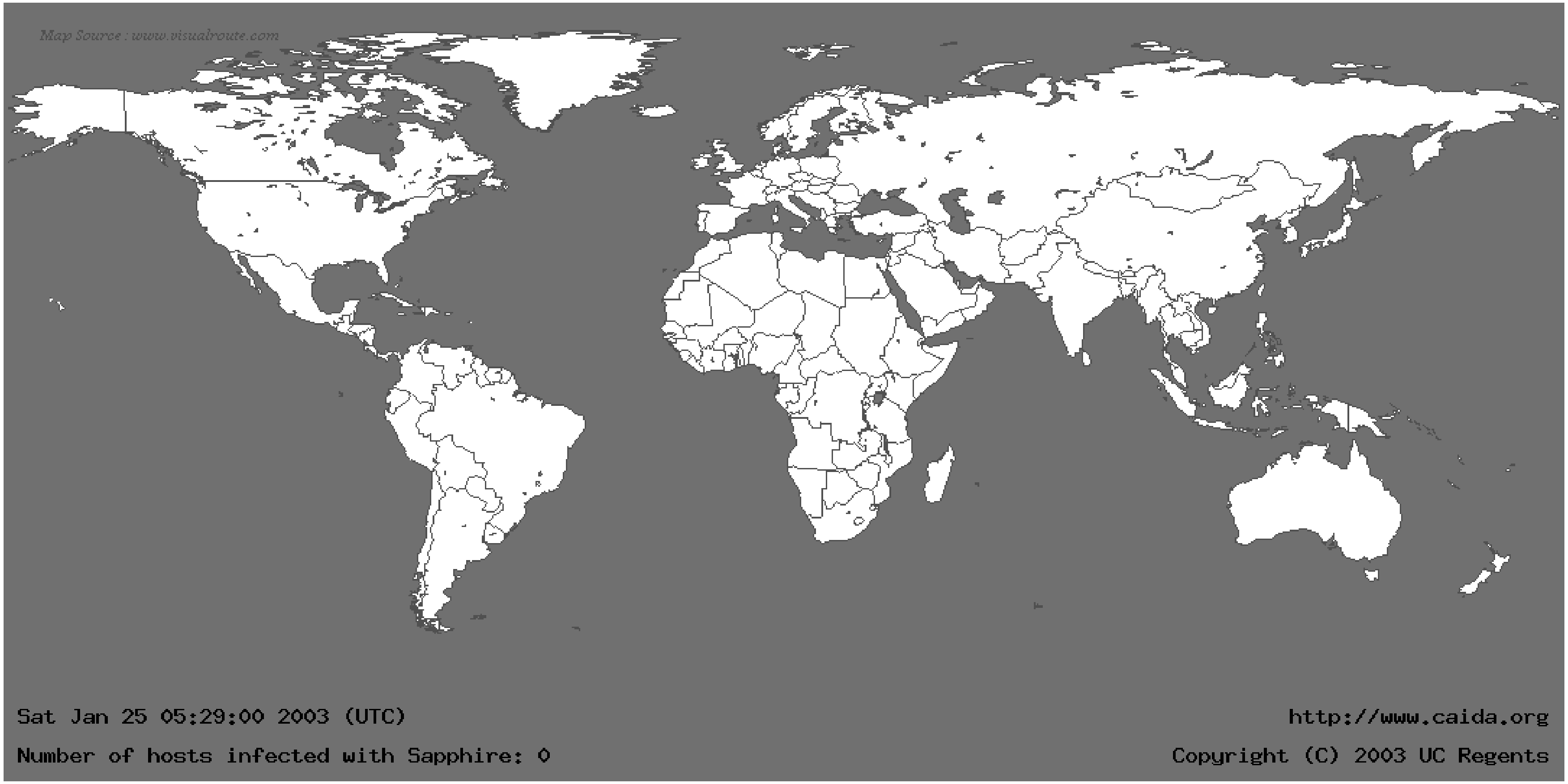
How can we prevent Alice’s computer from being hacked, when it communicates with some other computer?

Or detect this? And then react ?

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

The problem

Slammer Worm (Jan 2002)



Sat Jan 25 05:29:00 2003 (UTC)

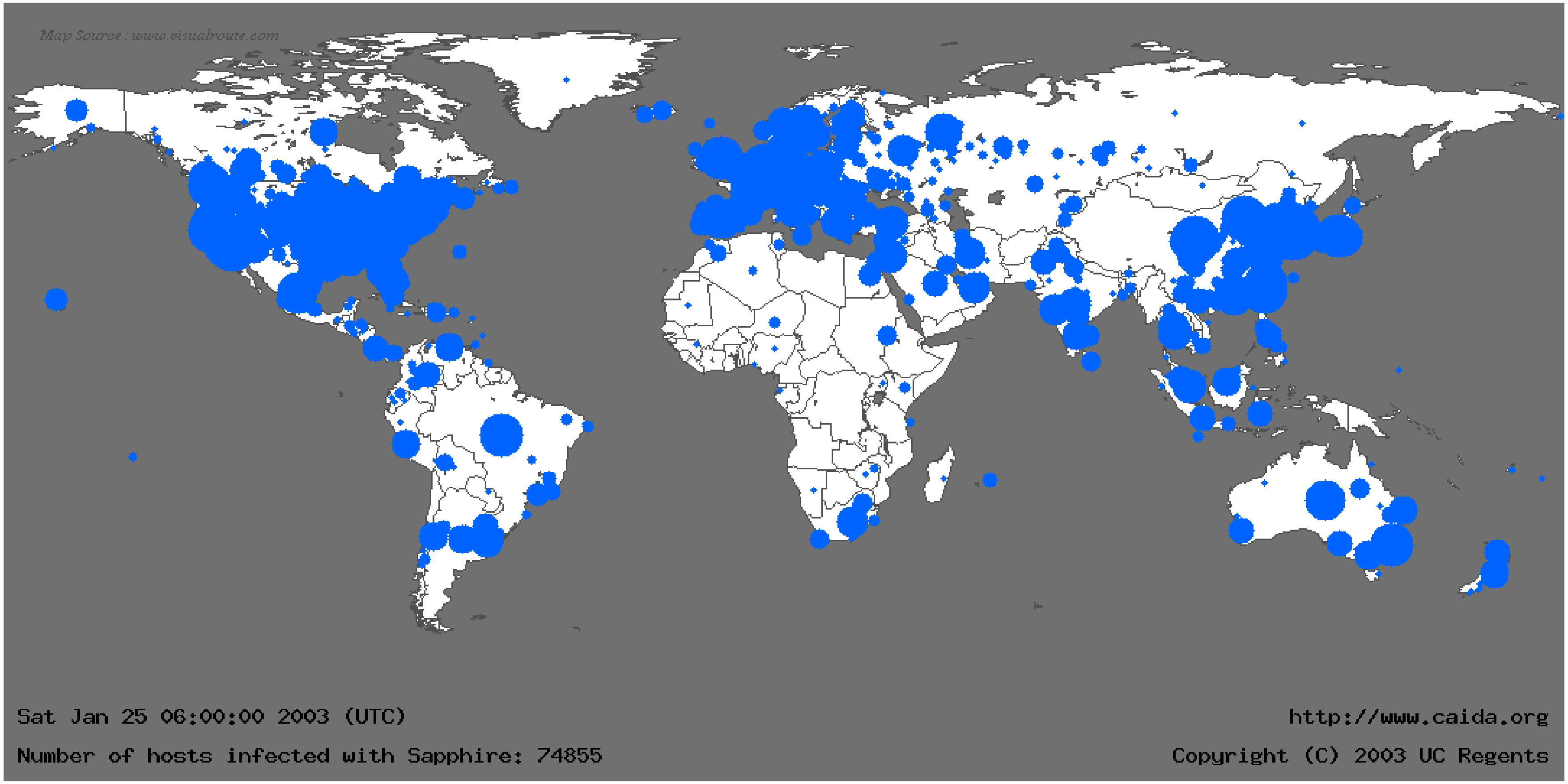
Number of hosts infected with Sapphire: 0

http://www.caida.org

Copyright (C) 2003 UC Regents

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

Slammer Worm (Jan 2002)


Map Source : www.visualroute.com
Sat Jan 25 06:00:00 2003 (UTC)
Number of hosts infected with Sapphire: 74855
http://www.caida.org
Copyright (C) 2003 UC Regents

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 are amateurs motivated by fun

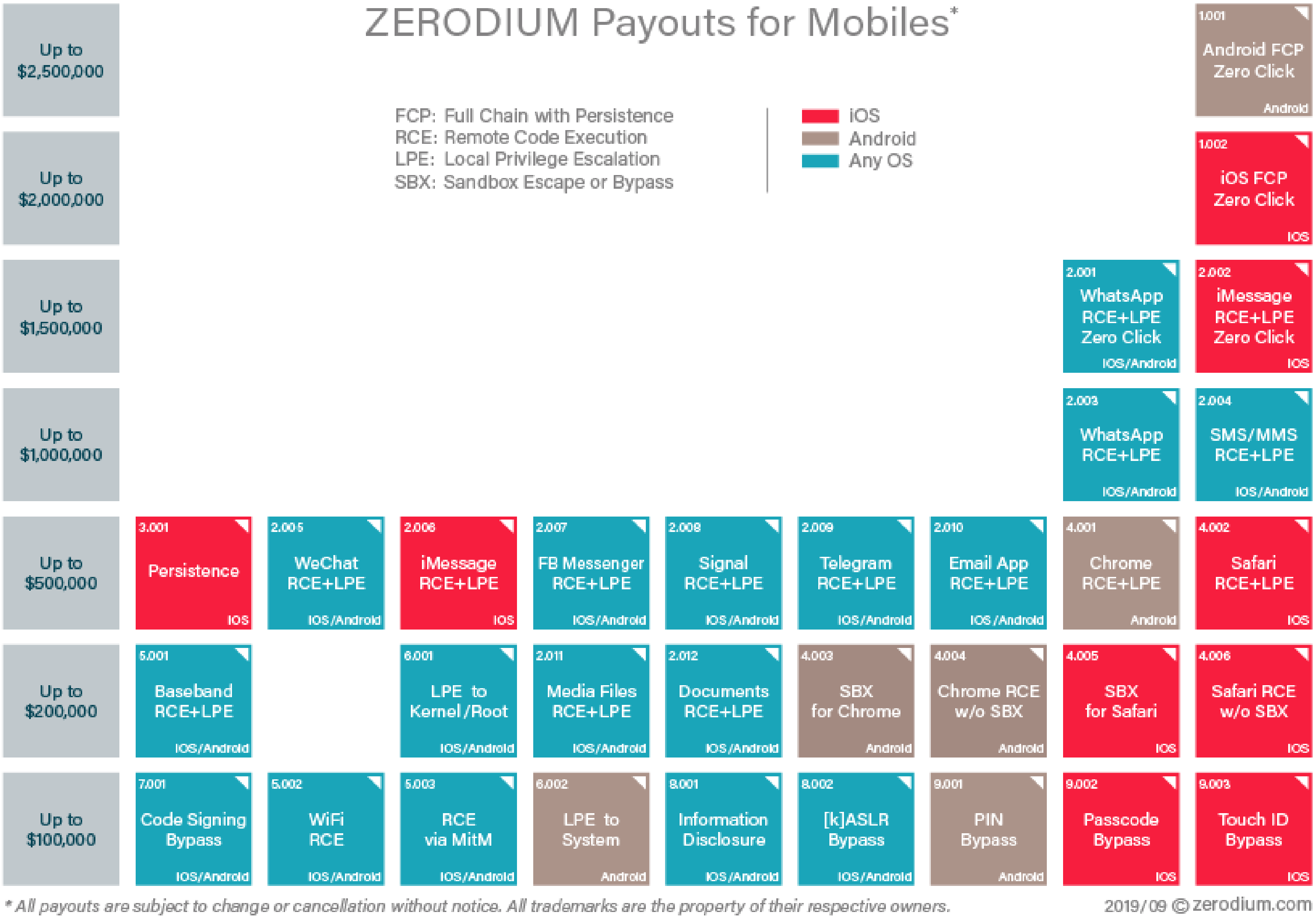
- publishing attacks for the prestige

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.
  - state actors:
    with even more money & in-house expertise

Current prices for 0days



ZERODIUM Payouts for Mobiles*

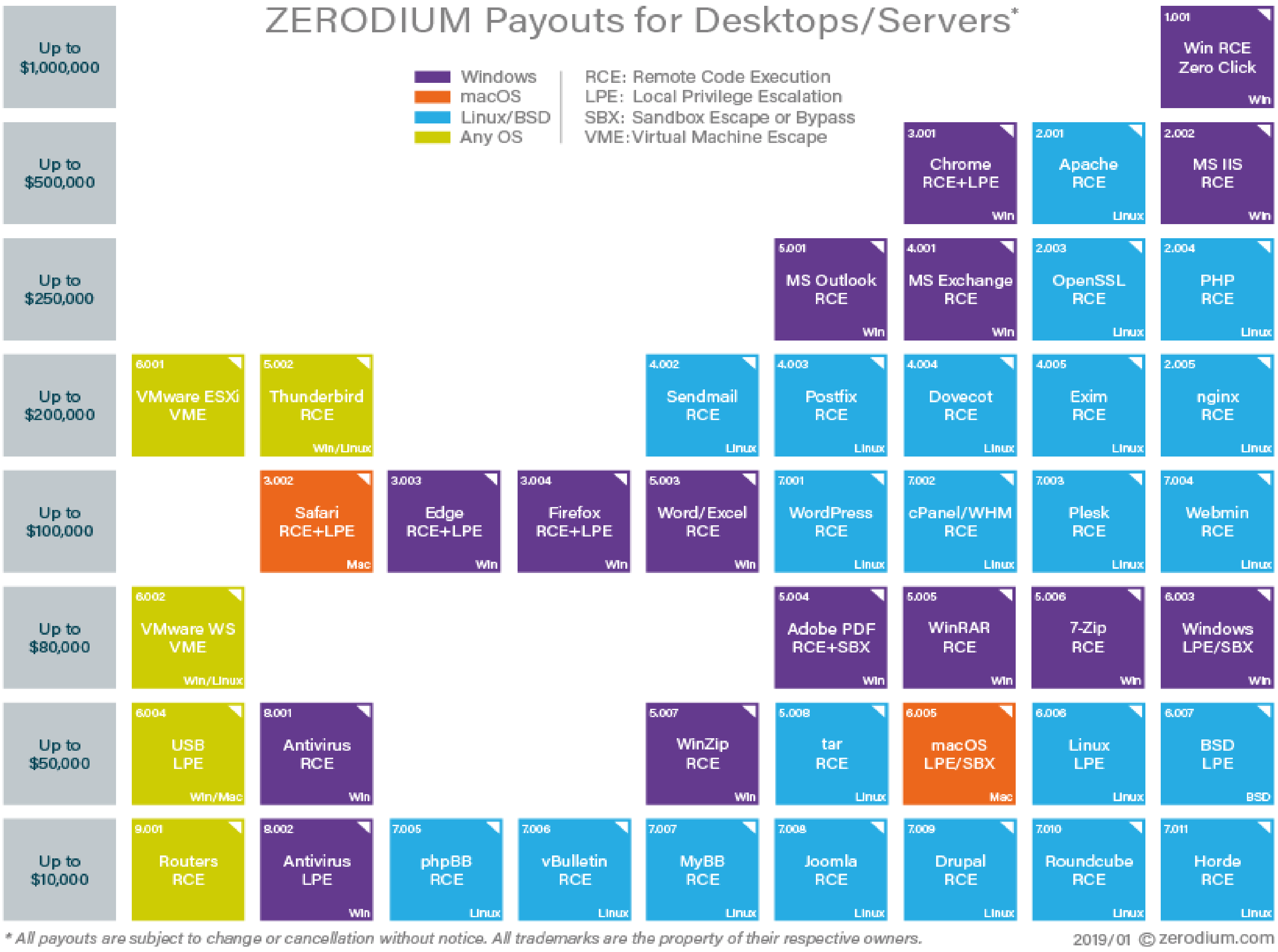
FCP: Full Chain with Persistence
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass

🟥 iOS
🟫 Android
🟦 Any OS

Up to $2,500,000									1.001 Android FCP Zero Click Android
Up to $2,000,000									1.002 iOS FCP Zero Click iOS
Up to $1,500,000								2.001 WhatsApp RCE+LPE Zero Click iOS/Android	2.002 iMessage RCE+LPE Zero Click iOS
Up to $1,000,000								2.003 WhatsApp RCE+LPE iOS/Android	2.004 SMS/MMS RCE+LPE iOS/Android
Up to $500,000	3.001 Persistence iOS	2.005 WeChat RCE+LPE iOS/Android	2.006 iMessage RCE+LPE iOS	2.007 FB Messenger RCE+LPE iOS/Android	2.008 Signal RCE+LPE iOS/Android	2.009 Telegram RCE+LPE iOS/Android	2.010 Email App RCE+LPE iOS/Android	4.001 Chrome RCE+LPE Android	4.002 Safari RCE+LPE iOS
Up to $200,000	5.001 Baseband RCE+LPE iOS/Android		6.001 LPE to Kernel/Root iOS/Android	2.011 Media Files RCE+LPE iOS/Android	2.012 Documents RCE+LPE iOS/Android	4.003 SBX for Chrome Android	4.004 Chrome RCE w/o SBX Android	4.005 SBX for Safari iOS	4.006 Safari RCE w/o SBX iOS
Up to $100,000	7.001 Code Signing Bypass iOS/Android	5.002 WiFi RCE iOS/Android	5.003 RCE via MitM iOS/Android	6.002 LPE to System Android	8.001 Information Disclosure iOS/Android	8.002 [k]ASLR Bypass iOS/Android	9.001 PIN Bypass Android	9.002 Passcode Bypass iOS	9.003 Touch ID Bypass iOS

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

Current prices for 0days



ZERODIUM Payouts for Desktops/Servers*

🟪 Windows	RCE: Remote Code Execution
🟧 macOS	LPE: Local Privilege Escalation
🟦 Linux/BSD	SBX: Sandbox Escape or Bypass
🟨 Any OS	VME: Virtual Machine Escape

Up to $1,000,000									1.001 Win RCE Zero Click Win
Up to $500,000							3.001 Chrome RCE+LPE Win	2.001 Apache RCE Linux	2.002 MS IIS RCE Win
Up to $250,000						5.001 MS Outlook RCE Win	4.001 MS Exchange RCE Win	2.003 OpenSSL RCE Linux	2.004 PHP RCE Linux
Up to $200,000	6.001 VMware ESXi VME	5.002 Thunderbird RCE Win/Linux			4.002 Sendmail RCE Linux	4.003 Postfix RCE Linux	4.004 Dovecot RCE Linux	4.005 Exim RCE Linux	2.005 nginx RCE Linux
Up to $100,000		3.002 Safari RCE+LPE Mac	3.003 Edge RCE+LPE Win	3.004 Firefox RCE+LPE Win	5.003 Word/Excel RCE Win	7.001 WordPress RCE Linux	7.002 cPanel/WHM RCE Linux	7.003 Plesk RCE Linux	7.004 Webmin RCE Linux
Up to $80,000	6.002 VMware WS VME Win/Linux					5.004 Adobe PDF RCE+SBX Win	5.005 WinRAR RCE Win	5.006 7-Zip RCE Win	6.003 Windows LPE/SBX Win
Up to $50,000	6.004 USB LPE Win/Mac	8.001 Antivirus RCE Win			5.007 WinZip RCE Win	5.008 tar RCE Linux	6.005 macOS LPE/SBX Mac	6.006 Linux LPE Linux	6.007 BSD LPE BSD
Up to $10,000	9.001 Routers RCE	8.002 Antivirus LPE Win	7.005 phpBB RCE Linux	7.006 vBulletin RCE Linux	7.007 MyBB RCE Linux	7.008 Joomla RCE Linux	7.009 Drupal RCE Linux	7.010 Roundcube RCE Linux	7.011 Horde RCE Linux

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

2019/01 © zerodium.com

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 functionality & services;
  - managing associated risks is a derived/secondary concern

- There is often a trade-off/conflict between
  - security
  - functionality & convenience

  where security typically loooses 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.

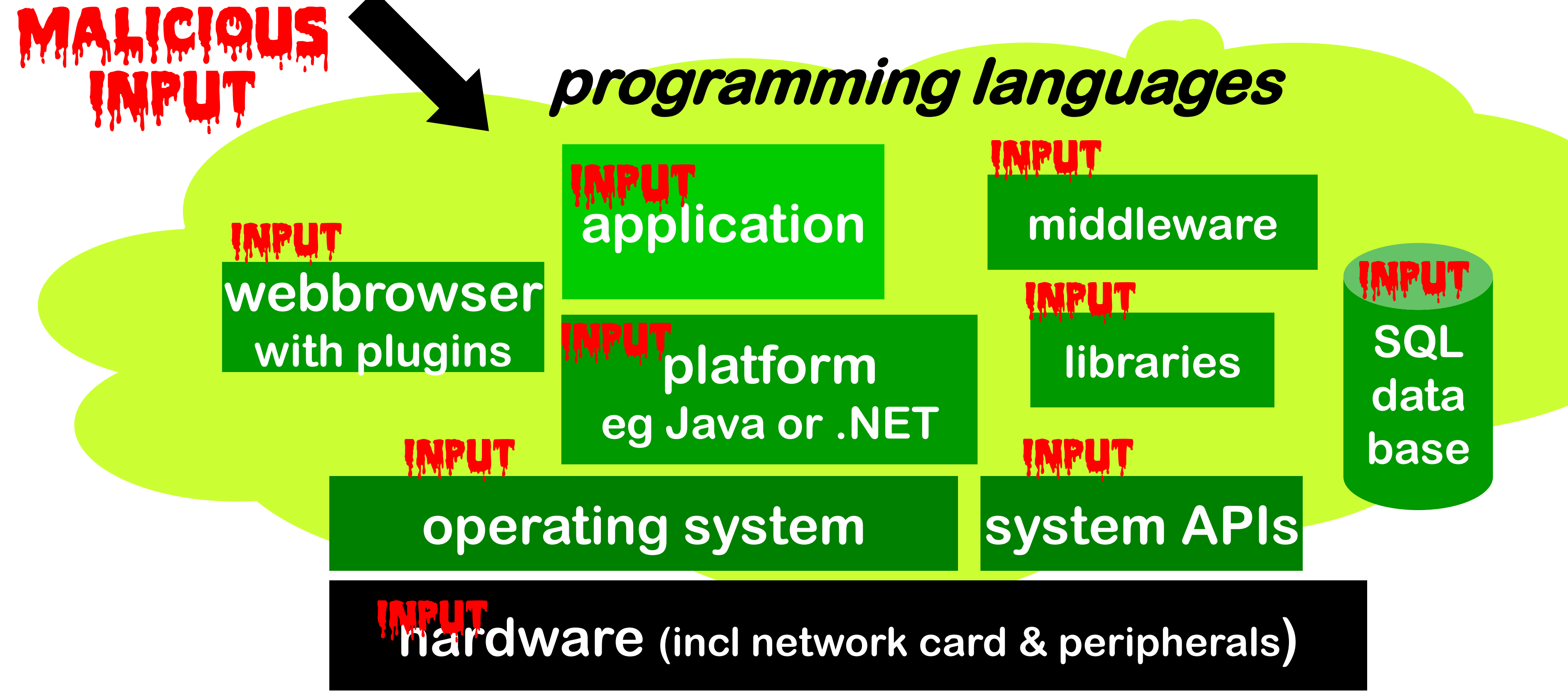
It takes far too much effort. .... PHP's raison d'etre is that it is simple to pick up and make it do something useful. There needs to be a major push ... to make it safe for the likely level of programmers - newbies.

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...



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**

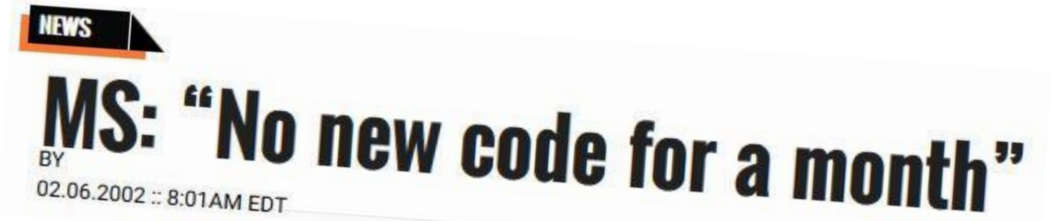
   flaws that can actually be exploited by an attacker

   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

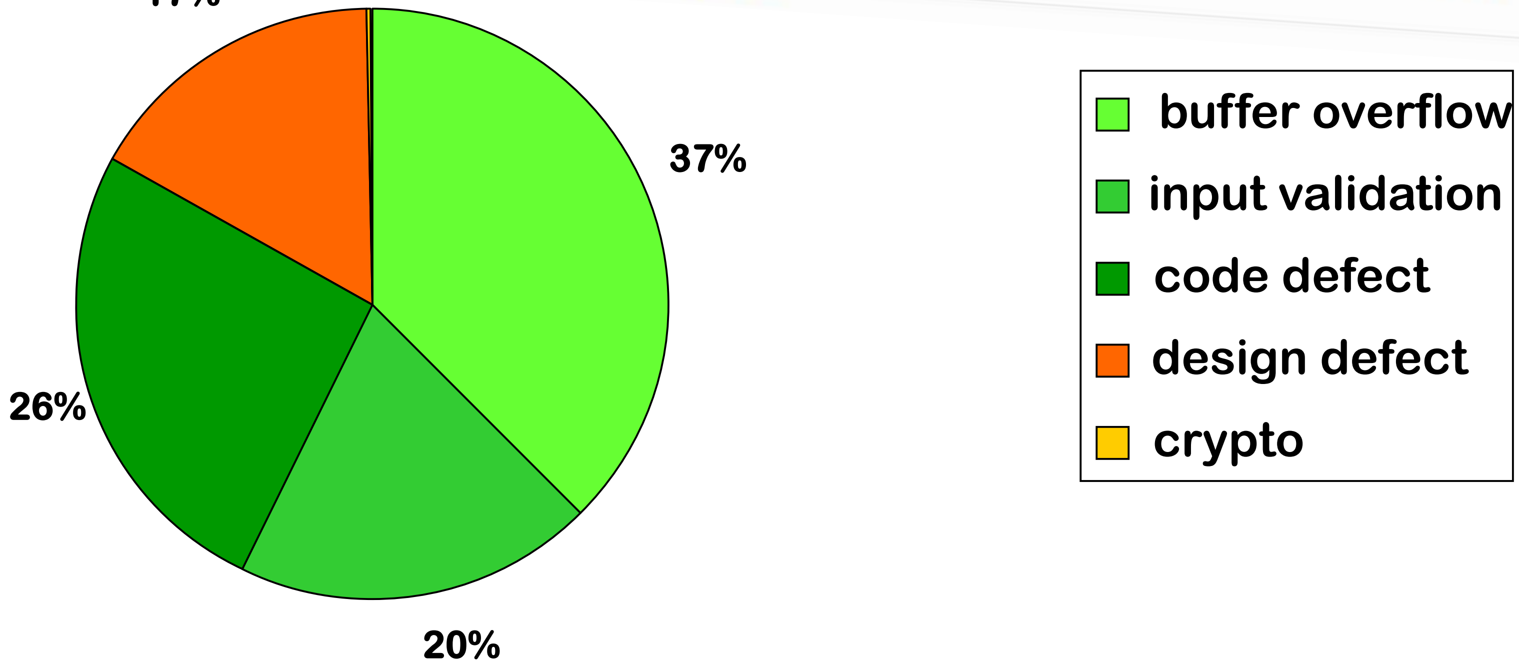


NEWS

MS: “No new code for a month”

BY

02.06.2002 :: 8:01AM EDT



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

Design vs implementations flaws

Another useful distinction:

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

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

Coding flaws

For flaws introduced during coding, we can distinguish

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,...

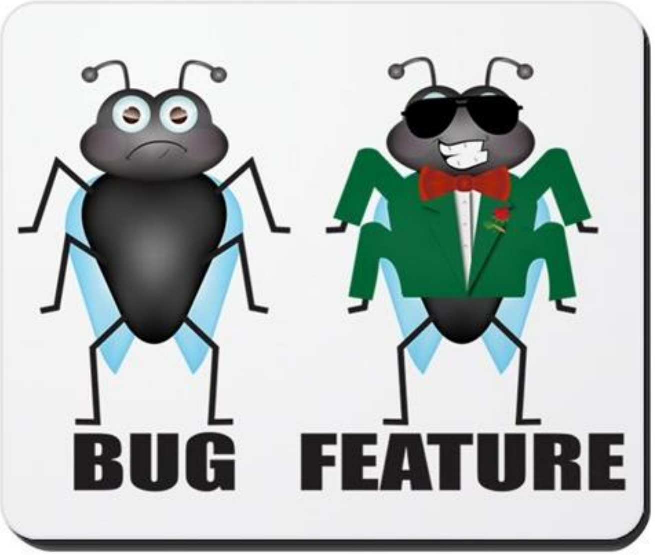
2b. (common) problems in the interaction with the underlying platform or other systems and services, eg

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

Bug vs features

Another useful distinction: security flaws can be

1. bugs
2. features
    - 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 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)

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

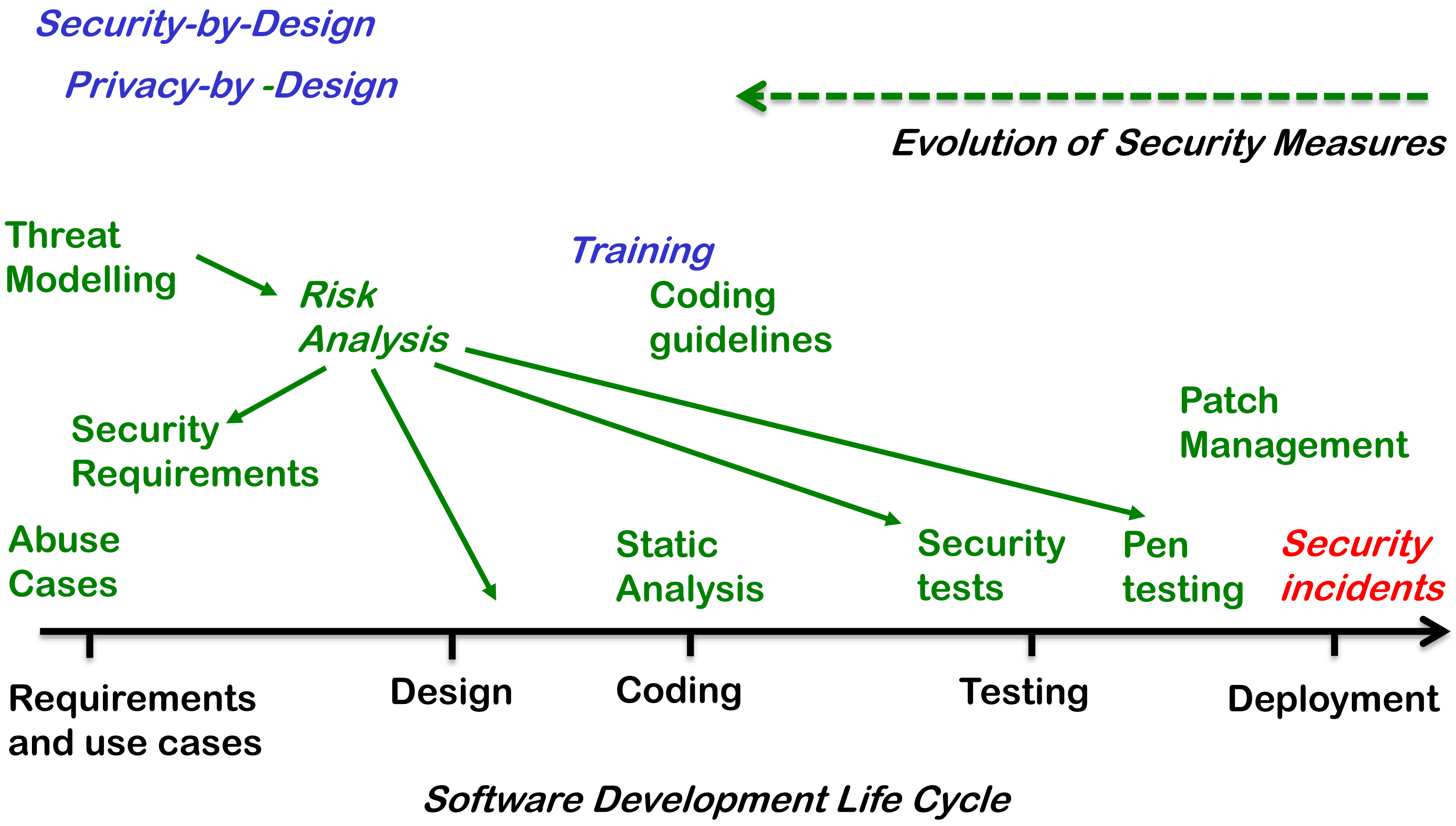
How to improve software insecurity?

- 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



Security-by-Design

Privacy-by -Design

Evolution of Security Measures

Threat Modelling

Risk Analysis

Training

Coding guidelines

Security Requirements

Patch Management

Abuse Cases

Static Analysis

Security tests

Pen testing

Security incidents

Requirements and use cases

Design

Coding

Testing

Deployment

Software Development Life Cycle

Evolution in tackling software security

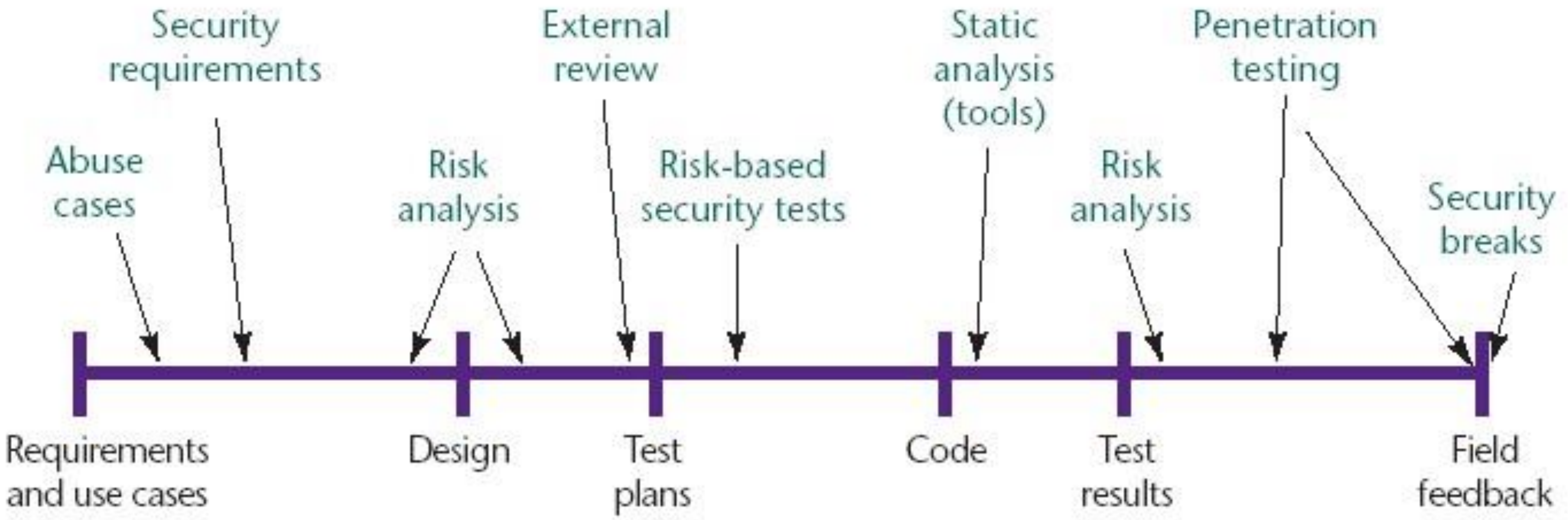
Organisations always begin tackling security at the *end* of the SDLC, and then slowly evolve to tackle it earlier

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 the Software Development Life Cycle

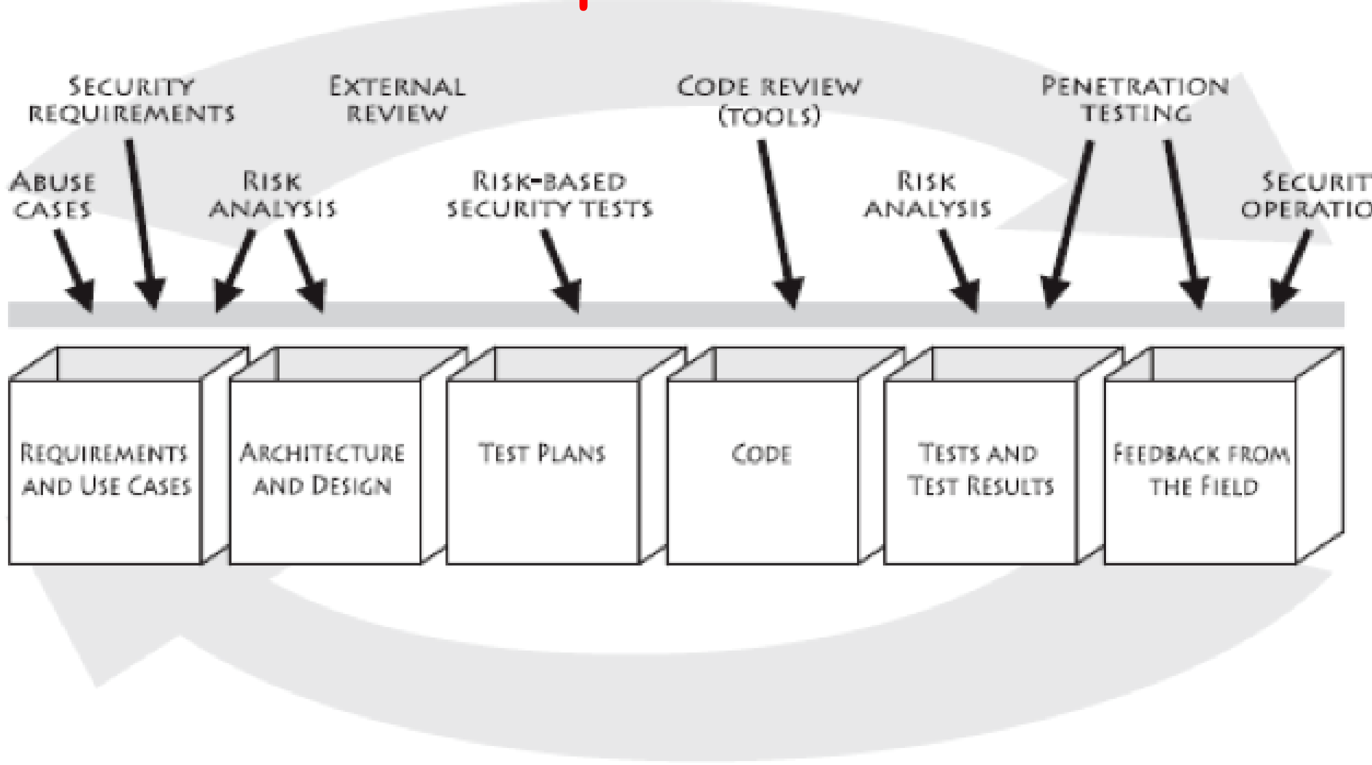
McGraw’s Touchpoints



[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]

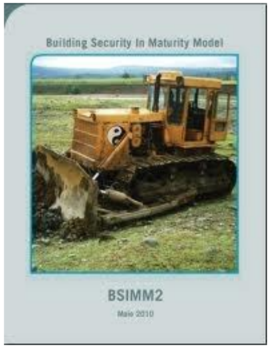
Methodologies for security in SDLC

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

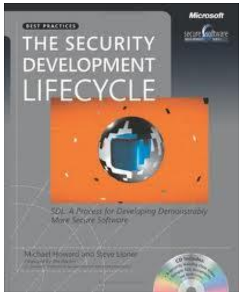
- McGraw's Touchpoints

  BSIMM Building Security In – Maturity Model

  http://bsimm.com



- Microsoft SDL


Microsoft
BEST PRACTICES
THE SECURITY DEVELOPMENT LIFECYCLE
SDL: A Process for Developing Demonstrably More Secure Software
Michael Howard and Steve Lipner

- OpenSAMM Software Assurance Maturity Model

  http://opensamm.org

OpenSAMM

OpenSAMM's 4 business functions and 12 security practices

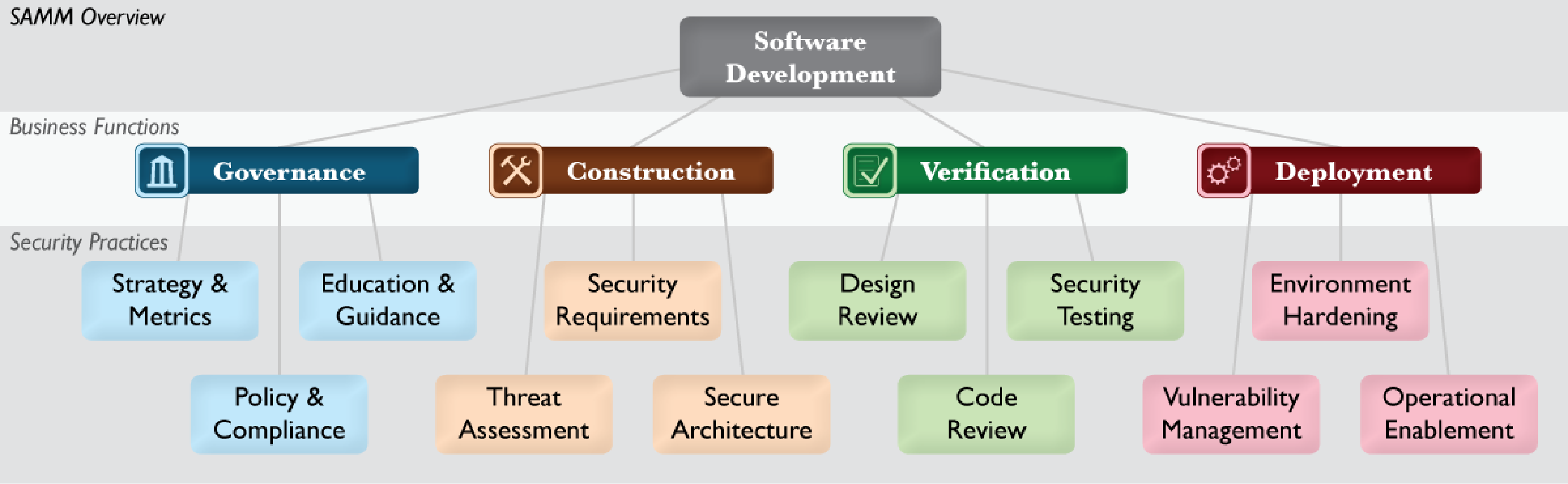
OpenSAMM

Governance

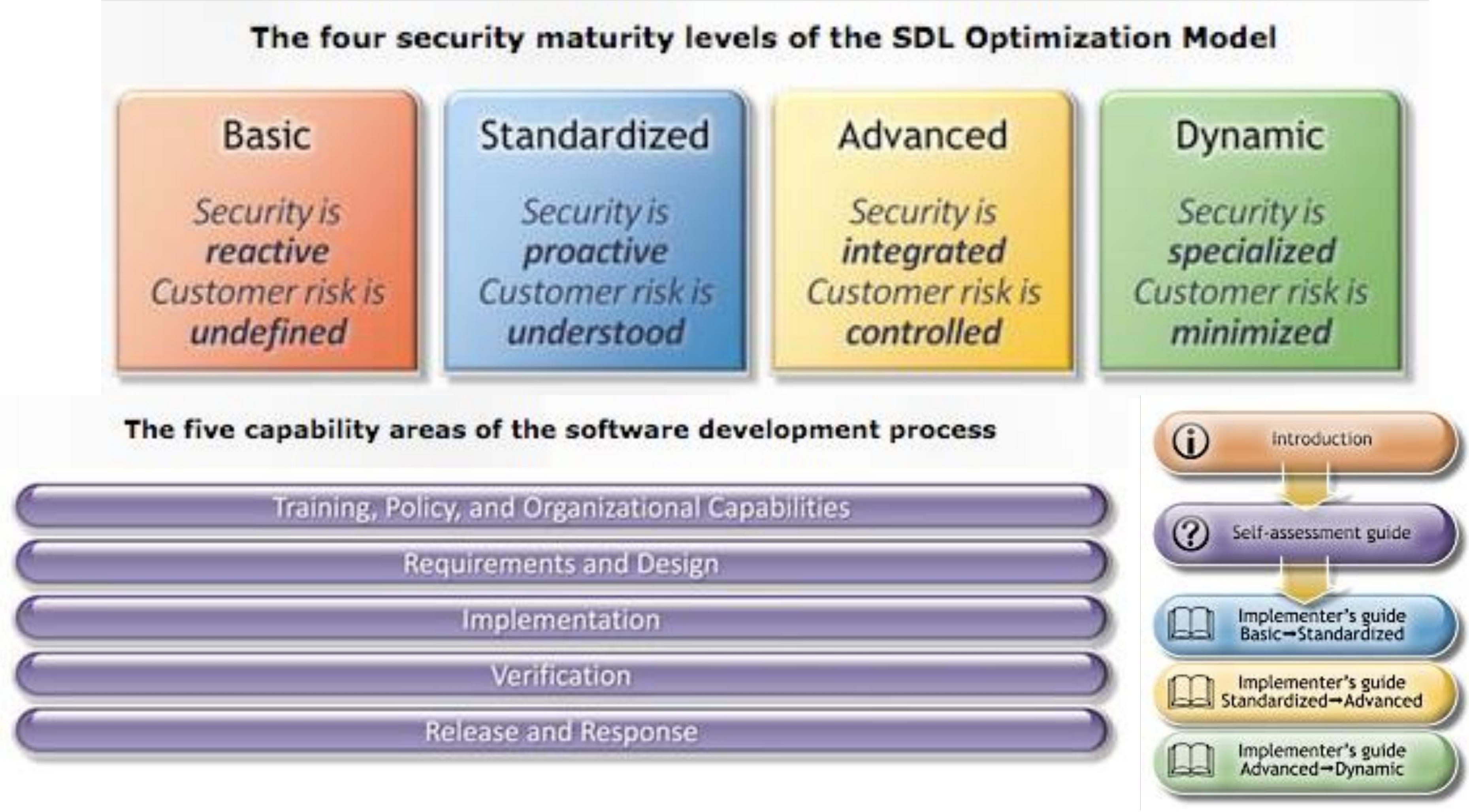
Construction

Verification

Deployment



Microsoft's SDL Optimisation Model



The four security maturity levels of the SDL Optimization Model

Basic: Security is ***reactive*** Customer risk is ***undefined***

Standardized: Security is ***proactive*** Customer risk is ***understood***

Advanced: Security is ***integrated*** Customer risk is ***controlled***

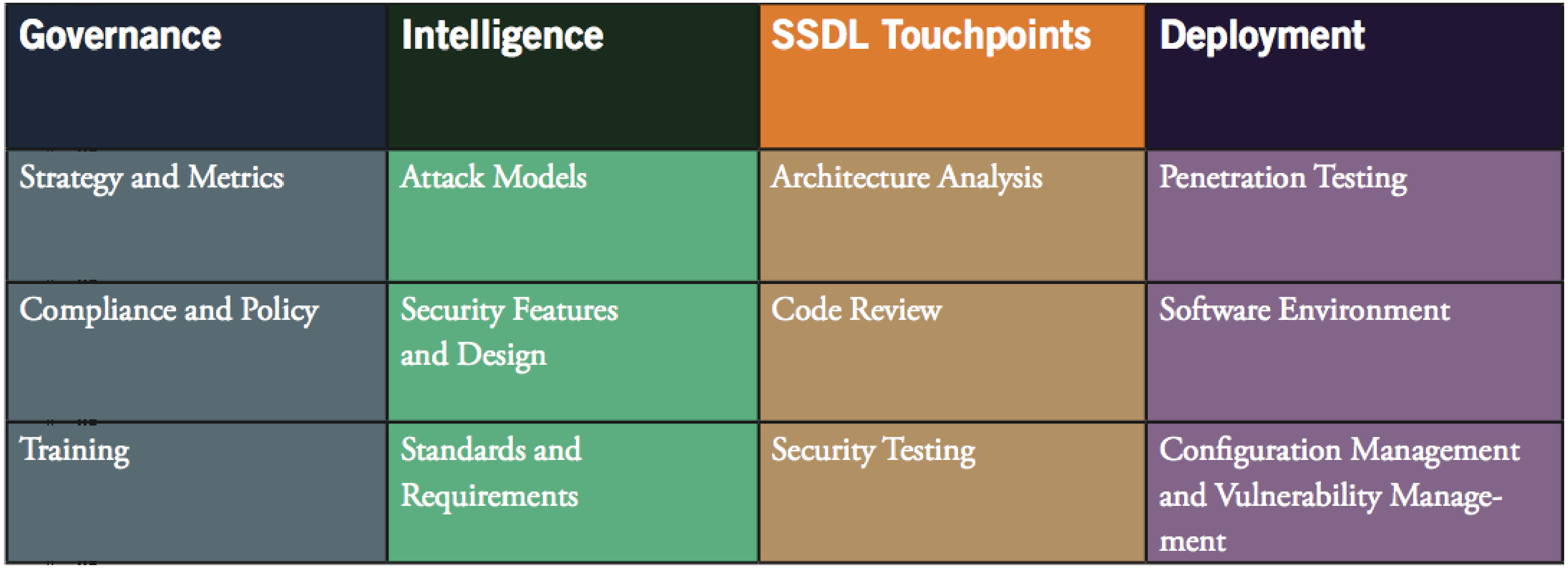
Dynamic: Security is ***specialized*** Customer risk is ***minimized***

The five capability areas of the software development process

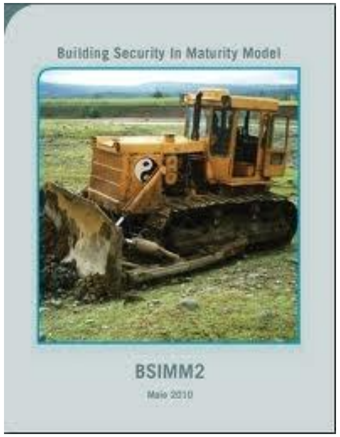
- Training, Policy, and Organizational Capabilities
- Requirements and Design
- Implementation
- Verification
- Release and Response

- Introduction
- Self-assessment guide
- Implementer's guide Basic→Standardized
- Implementer's guide Standardized→Advanced
- Implementer's guide Advanced→Dynamic

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 on security in the SDLC

CyBok chapter on Secure Software Lifecycle
by Laurie Williams, 2019

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

Attacker/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 typically **way** 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, 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 as prevention as way to ensure security, but detection & response are foten 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
  - ...

To read & do for coming week

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

- Check out recent CVEs: see links on course page

- Brush up on your C knowledge