Software Security

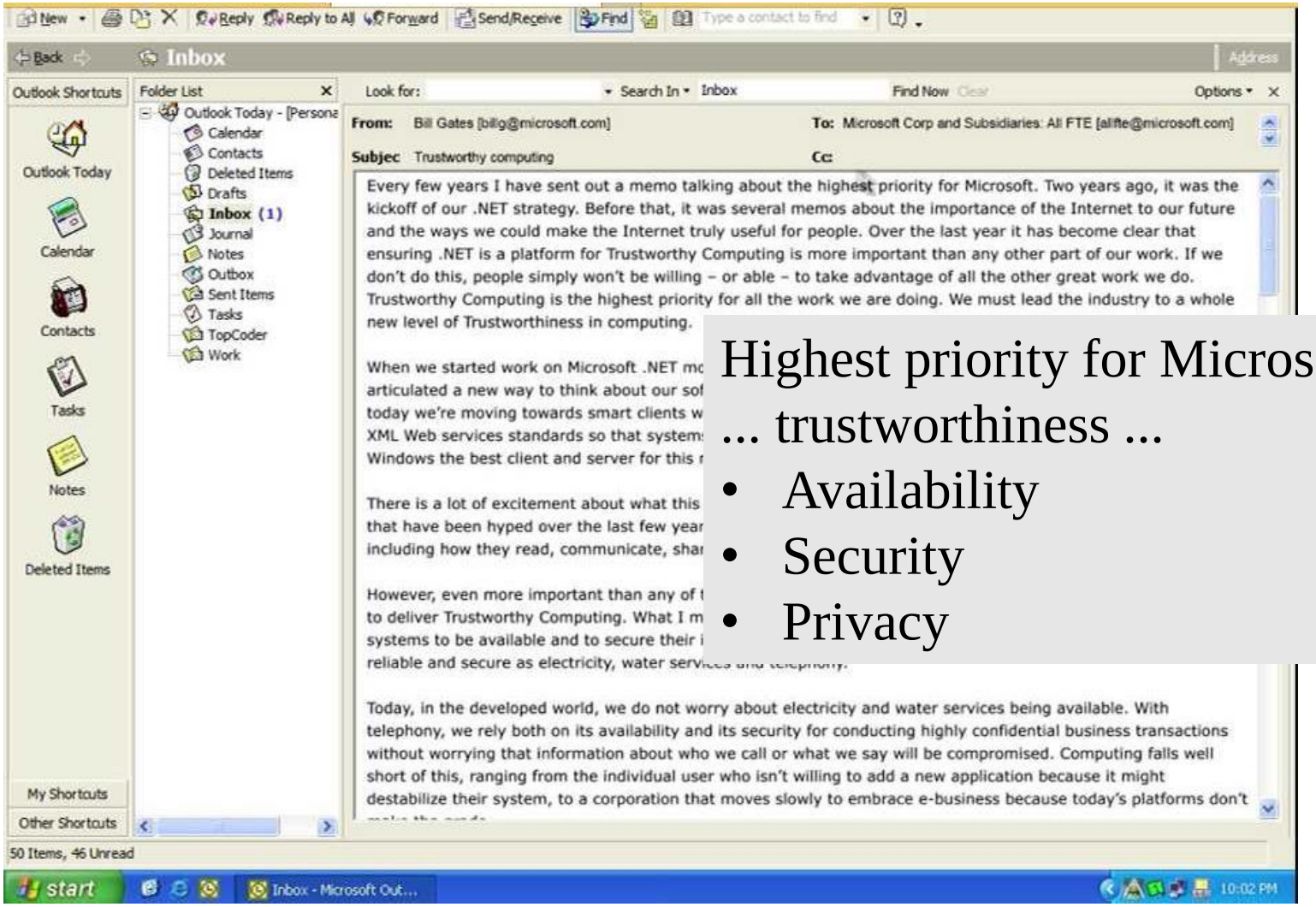
Introduction

Erik Poll

Digital Security

Radboud University Nijmegen

A brief history of software security: January 2002



Highest priority for Microsoft:
... trustworthiness ...

- Availability
- Security
- Privacy

https://news.microsoft.com/2012/01/11/memo-from-bill-gates/

Twenty years later (Sept 2022 & May 2023)

proposed regulation
to complement
NIS2 framework

EU & US announce regulation for software security

https://digital-strategy.ec.europa.eu/en/policies/cyber-resilience-act

https://www.whitehouse.gov/briefing-room/statements-releases/2023/03/02/fact-sheet-biden-harris-administration-announces-national-cybersecurity-strategy

So: problem solved?

https://www.cisa.gov/news-events/bulletins

https://cve.mitre.org/cve/search_cve_list.html

Homework for the coming: check out

a) the latest US-CERT bulletin

b) recent CVEs for the browser, PDF viewer, and other software you

c) some of their CVSS scores

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

- Basic security knowledge
  - TCB (Trusted Computing Base),
    CIA (Confidentiality, Integrity, Availability),
    Authentication, ...

- Basic knowledge of programming, 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'll 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'll 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 object-oriented code you'll 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 & mandatory reading

- slides
- my written lecture notes
- (parts of) some articles

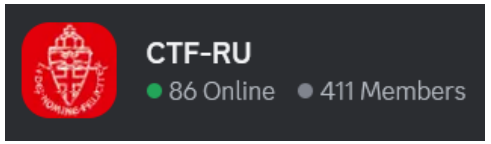
I'll be updating this in Brightspace as we go along

NB keep track of Brightspace announcements

If you do not log into Brightspace regularly,
have these announcements forwarded to your email

Not exam material

- Join the student CTF group if you're interested in the practical side of security
  - in Discord https://discord.gg/bD8D7S5euv
  - IRL Tuesdays at 17:30 in Mercator fishbowl



- I recommend the Risky.Biz podcast to keep up with weekly security news

RISKY.BIZ
It's a jungle out there

Not exam material

- OWASP Netherlands meet-up (i.e. free pizza!!)

  Sept 21 in Utrecht & Oct 19 in Nijmegen

  See https://owasp.org/www-chapter-netherlands/#div-upcoming

Register for the (low-traffic) OWASP-NL mailing list to be informed of such events

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
  - exercise on web site sanitisations
  - project on static analysis with Semmle (individual or in pairs)
- written exam

Bonus point for group project, computed as `(grade-6)/4`

Today

- What is "software security"?
- Some root causes of the problems
- The solution to the problems

Motivation

What is software security?

Intersection of security & software engineering:

- *prevent* design-level & implementation-level security vulnerabilities and pro-actively design & build systems that resist attacks
- *prevent* users from harming themselves & others by bad security choices
  - the same for programmers, sys admins, …
- *detect* vulnerabilities that arise - *accidentally* or *intentionally* - and react to them
- *mitigate* risks
  before and after detecting problems

How do computer systems get ‘hacked’?

By attacking

- **software**

- **humans**

Or: the interaction between software & humans

- crypto
- hardware
- …

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 *big* problem

Alice & her computer are communicating with *another computer*

How to prevent Alice or her computer from getting *hacked*?

Or how to detect this? And then react ?

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

Changing nature of attackers

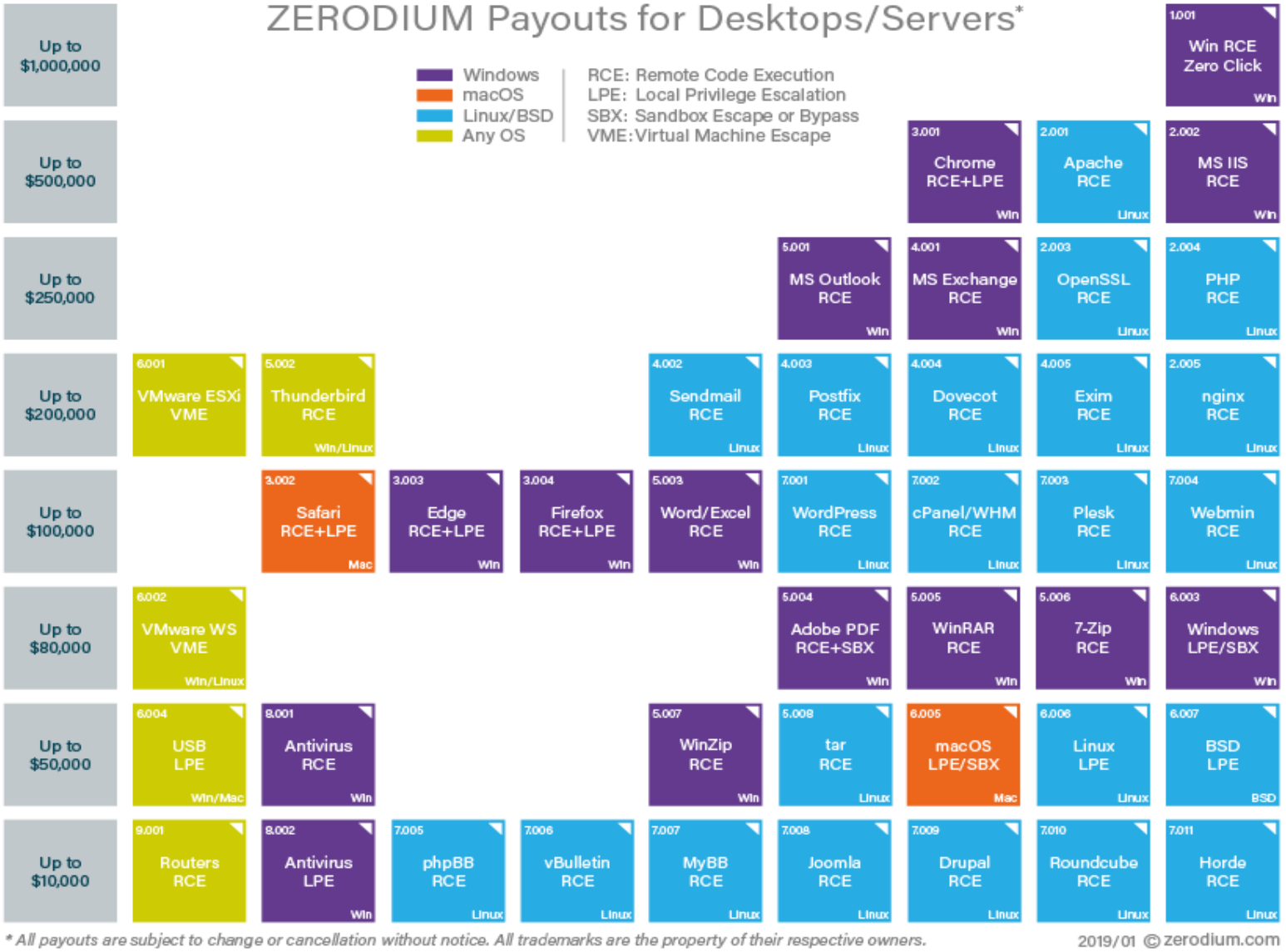
Traditionally, hackers were amateurs motivated by 'fun'

- 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
  - e.g. NSO group, Zerodium, …

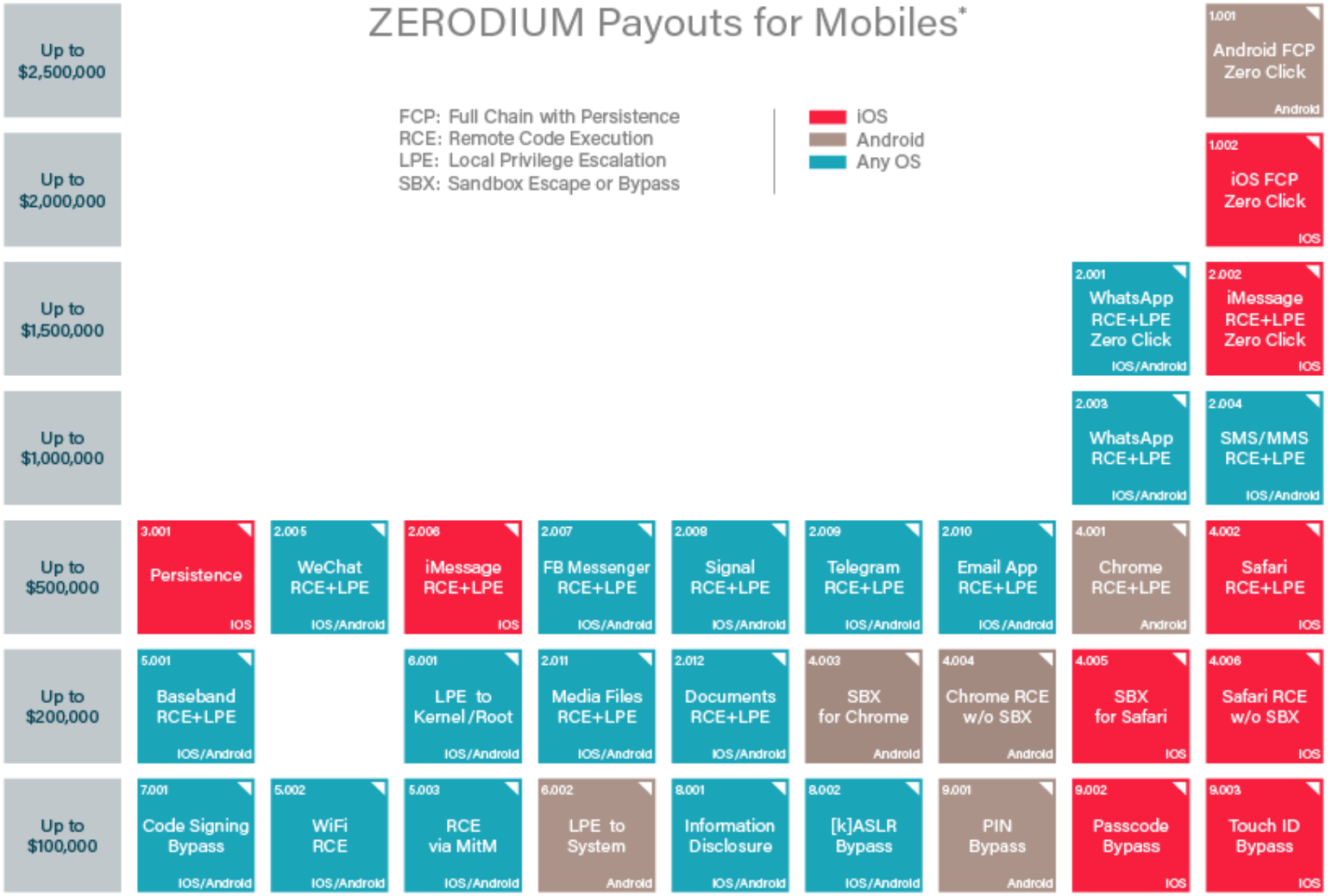
Prices for 0days



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

2019/01 © zerodium.com

Prices for 0days



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

2019/09 © zerodium.com

Apple & Google payouts

Google Offers $1.5M Bug Bounty for Android 13 Beta

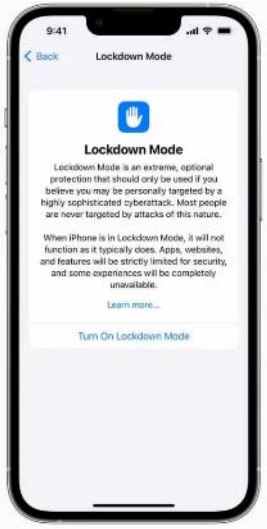
The security vulnerability payout set bug hunters rejoicing, but claiming the reward is much, much easier said than done.

Tara Seals
Managing Editor, News, Dark Reading

May 02, 2022

Apple will pay you $2 million if you can break its new 'Lockdown Mode'

By Joe Wituschek published July 07, 2022



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

- TLS, IPSEC, firewall, VPN, …
- AV (AntiVirus), WAF (Web Application Firewall)
- access contro
- NIDS (Network Intrusion Detection System)
- EDR (Endpoint Detection and Response)
- RASP (Runtime Application Self-Protection)
- …

But *all* software must be secure, not just the security software

- That buffer overflow in your PDF viewer can still be exploited…
- 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?*
  - *warn about format string attacks?*
  - *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**

More root causes: security vs functionality

Primary goal of software is providing functionality & services

Managing associated risks is a secondary concern

When there is often a trade-off/conflict between

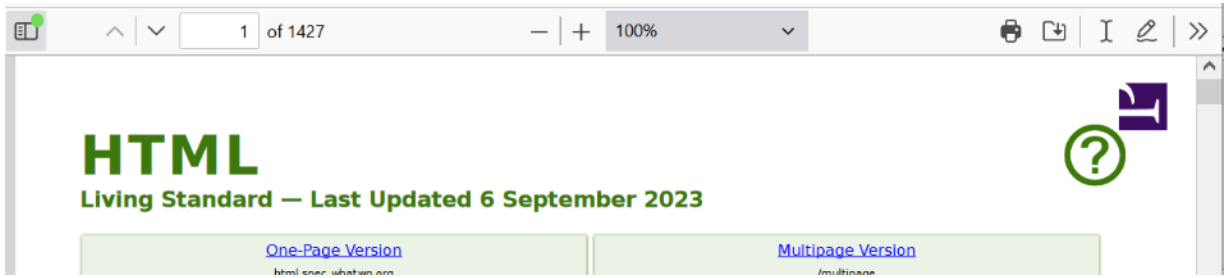
- security
- functionality, convenience, speed , …

then security typically looses out

- Users complain about missing features or broken functionality, but not about insecurity
- Developers like adding features, not thinking about security

Root causes: COMPLEXITY

- *Have anyone here read the HTML specification?*



- *Has anyone here read the URL specification?*

  Which one? There are two!

Updated by: 6874, 7320, 8820 — Errata Exist
Network Working Group — T. Berners-Lee
Request for Comments: 3986 — W3C/MIT
STD: 66 — R. Fielding
Updates: 1738 — Day Software
Obsoletes: 2732, 2396, 1808 — L. Masinter

- **Even security features we add to prevent problems are hopelessly complex**
  - *Has anyone read the TLS specifications?*

FUNCTIONALITY & COMPLEXITY vs security

Lost battles?

- **Programming languages & APIs**

  we want these easy to use, powerful and efficient,
  but they can be insecure, dangerous and error-prone

- **Operating systems (OSs)**

  with huge OS, with huge attack surface

- **Web browsers**

  with ever fancier features, JavaScript, Web APIs to access microphone, web cam, location, …

- **Email clients**

  - which handle with all sorts of formats & attachments

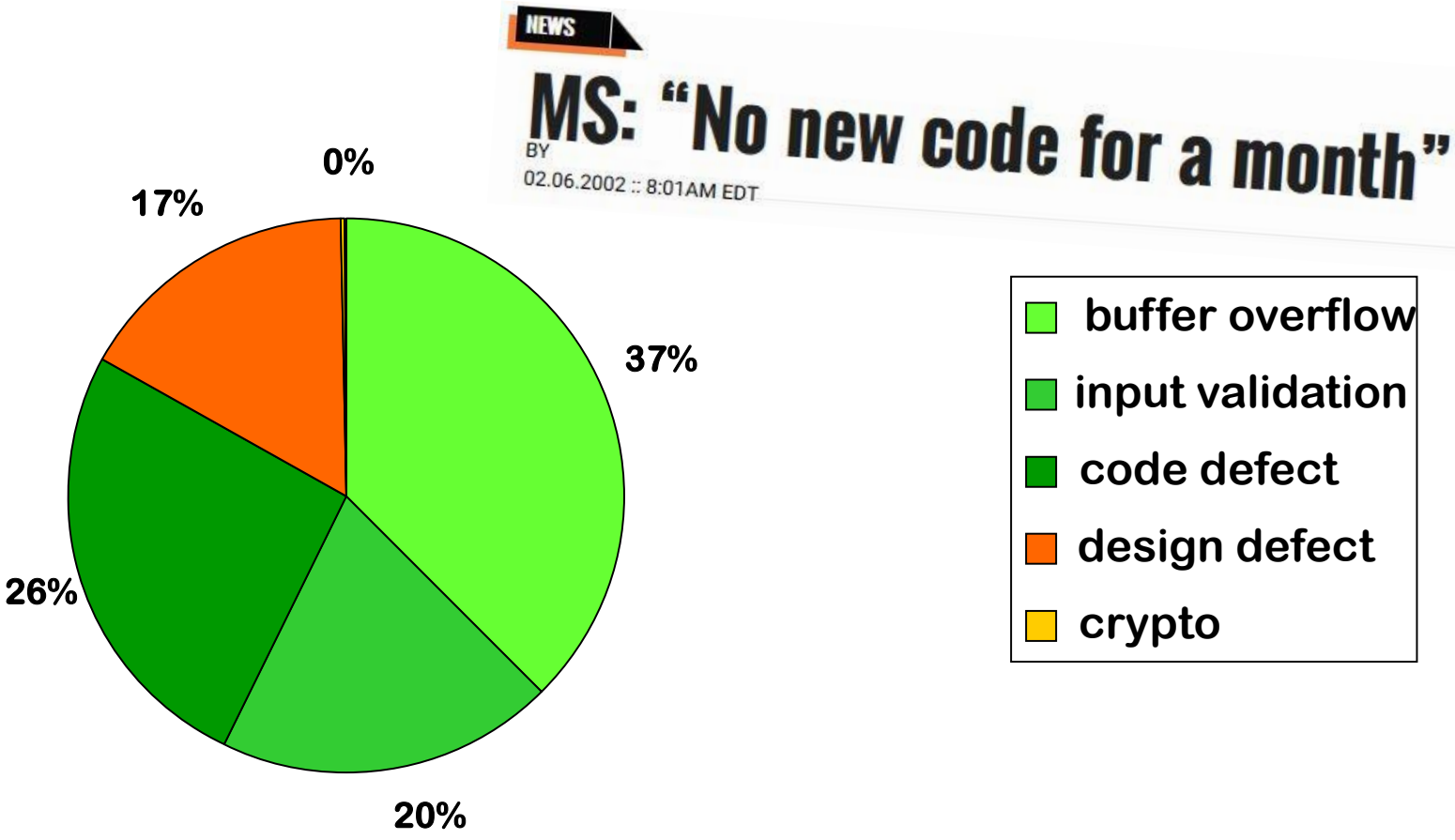
Recap

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

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!

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]

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!

We will always

- **have vulnerabilities that have not been found (yet)**
- **overlook attack vectors**
- **make implicit assumptions that are – or become – invalid**
- **overlook ways in which functionality can be abused**
- **miss security properties that are important**
- **…**

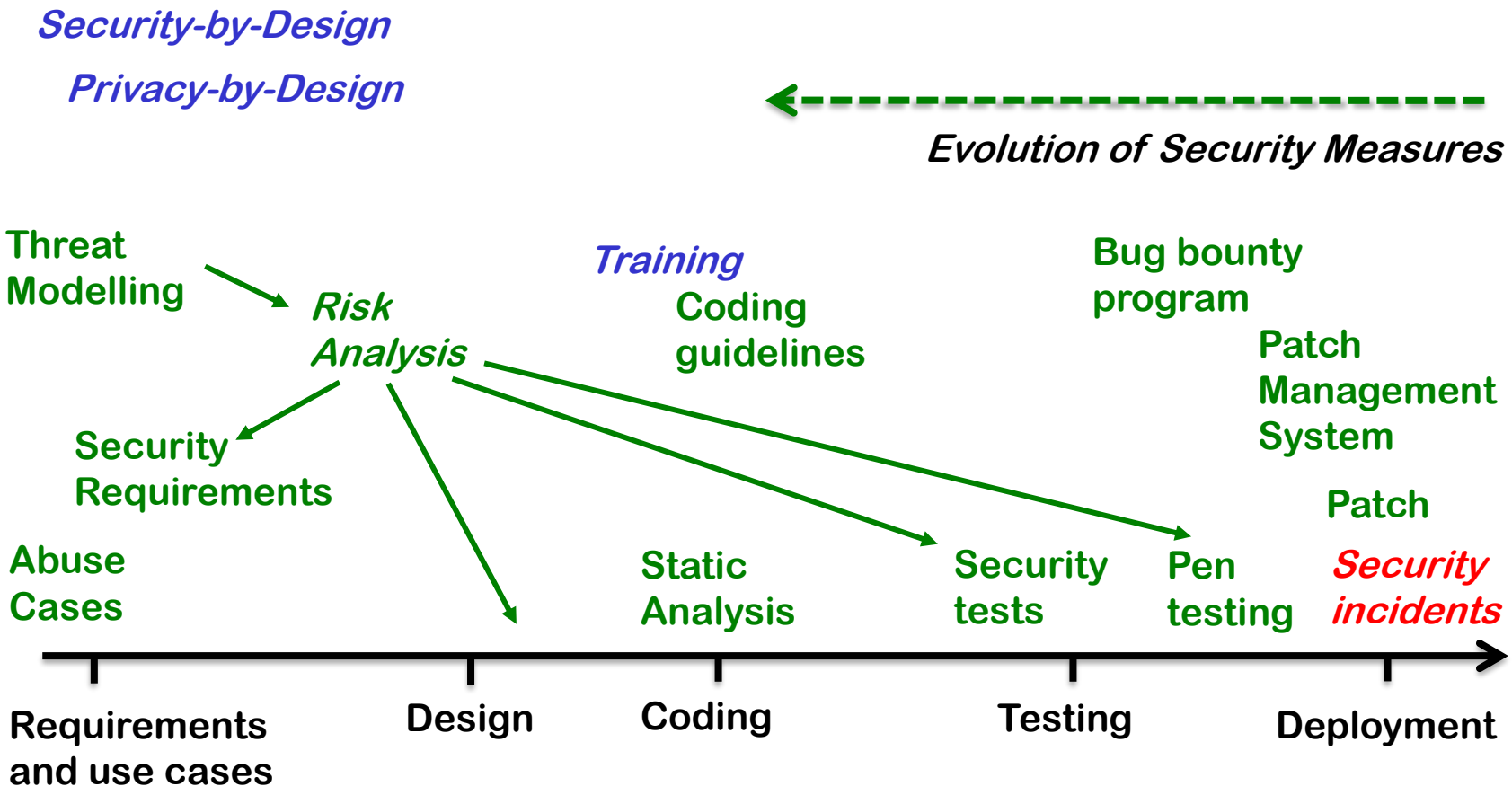
How can we make software *more* secure?

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



"Shifting left"

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

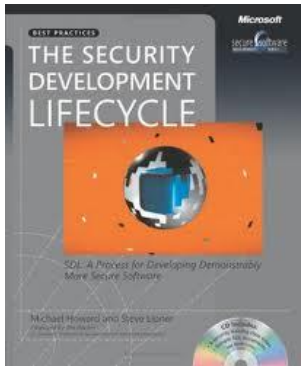
1. First, do nothing
2. Some security issue is discovered:
   a) Still do nothing, if there's no (economic) incentive
   b) Or: patch
3. If this happens often: update mechanism for regular patching
4. Do security testing: eg. hire pen-testers or bug bounty program
5. Use static analysis tools when coding
6. Give security training to programmers
7. Think of abuse cases, and develop security tests for them
8. Think about security *before* you start coding, eg with security architecture review
9. …

DAST, SAST, ...

Security people keep inventing 4 letter new acronyms

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

Methodologies for secure software development



Early ones

- Microsoft SDL

  with extension for Secure DevOps (DevSecOps)
- Touchpoints and BSIMM by Gary McGraw
- OWASP Open SAMM (Software Assurance Maturity Model)

Recent incarnations include

- NIST SSDF
- Grip op SSD (Secure Software Development)

  Ongoing initiative by Dutch government organisations

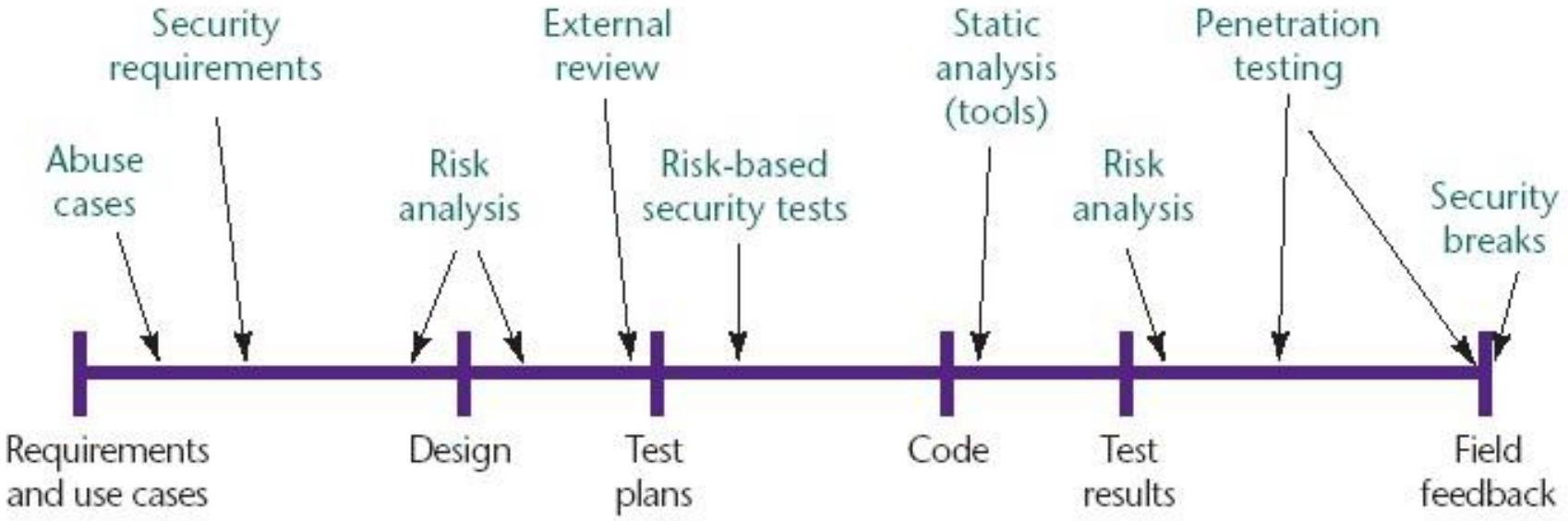
  https://www.cip-overheid.nl/en/category/products/secure-software/
- …

Complemented with Top N lists of dos or don'ts, checklists & cheatsheets, roadmaps, assessment methods, …

Microsoft's SDL Optimisation Model

Security in the software development life cycle

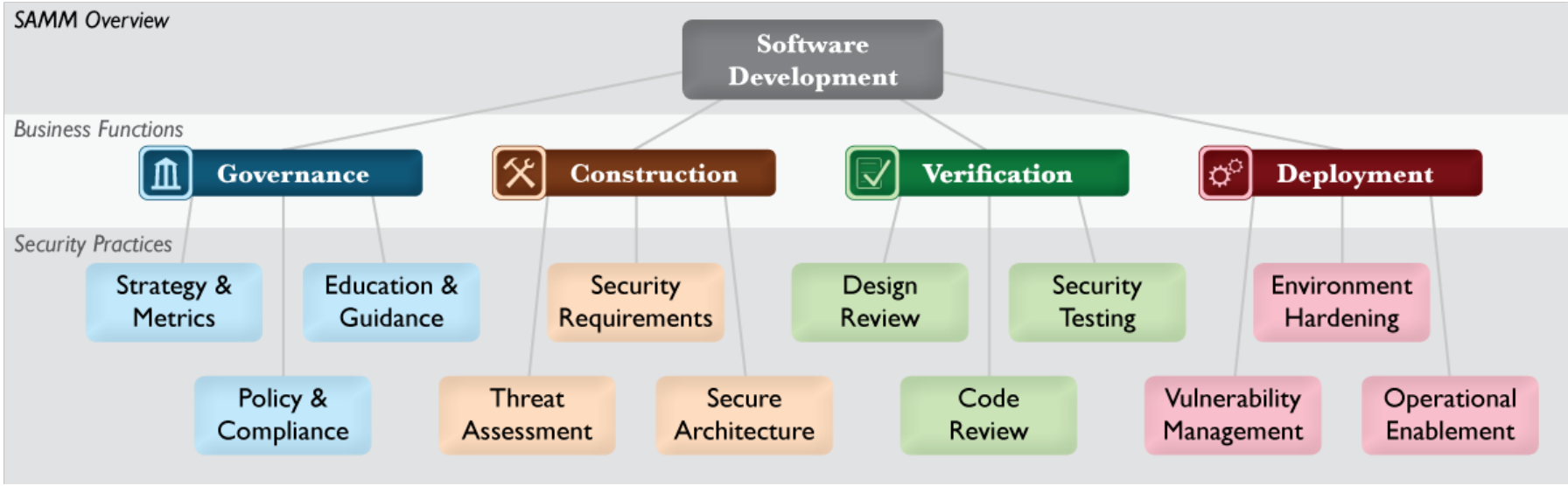
McGraw's Touchpoints



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

OpenSAMM

12 security practices grouped in 4 business functions



BSIMM (Building Security In Maturity Model)

110 activities in 12 practices across 4 domains

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 Manage-ment

Unfortunately, info about this has largely disappeared behind paywall of the corporate website of Synopsys ☹

BSIMM: comparing your security maturity

But first…

Discussing security is meaningless without answering

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
   - Also: what are your security assumptions ?
     - Including: what are the TCBs (Trusted Computing Bases) for specific security properties controls?

Aka threat modelling

Security requirements

a) 'This application cannot be hacked'

- Generic default requirement ☺
- Vague & not actionable ☹
- Negative security model

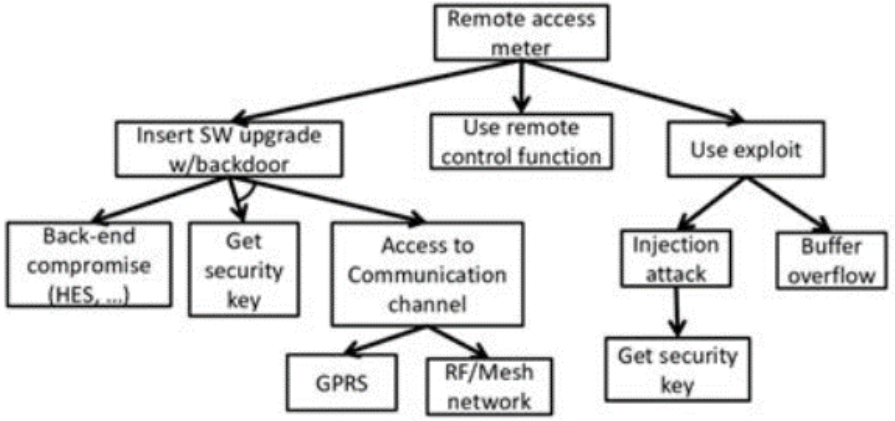
b) More specific security requirements

- In terms of Confidentiality, Integrity and Availability (CIA)
- Or, usually better, in terms of Access Control
  i.e. Authentication & Authorisation
- Not just Prevention but also Detection & Reaction/Response
- Positive security model
- Also thinking in negative terms can be useful

Threat modelling

Draw diagram of the system and then brainstorm about attacks & defenses using e.g. STRIDE or attack trees

- **S**poofing
- **T**ampering
- **R**epudiation
- **I**nformation Disclosure
- **D**enial of Service
- **E**levation of privilege



Read
https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats
if these STRIDE notions are not clear

MITRE ATT&CK is probably too detailed for threat modelling

prevention vs detection & reaction

prevention vs detection & reaction

- Prevention seems to be *the* way to ensure security, but detection & response often 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 to include are
  - doing monitoring
  - having logs for auditing and forensics
  - having someone actually inspecting the logs
  - ...

For you to read & do

1. To read: CyBok chapter on Secure Software Lifecycle

2. To do: check out
   a) the latest US-CERT bulletin
   b) recent CVEs for the browser, PDF viewer and other software you use on a regular basis
   c) some of their CVSS scores

3. To do: brush up on you C(++) knowledge

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;
}
```