Software Security

Introduction

Erik Poll

Digital Security

Radboud University Nijmegen

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: form & examination

- 2-hrs lecture every week
  - read associated papers & ask questions!

- mandatory project work
  - group project (with 4 people) on fuzzing
  - smaller exercises (individual or in pairs)
    - static analysis with PREfast for C/C++
    - static analysis with semgrep
- written exam

Group project grade counts toward final grade:
exam is 70%, project is 30%

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, Python, 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;
}
```

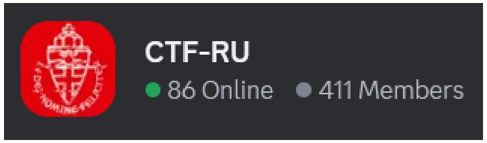
Exam material & mandatory reading

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

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

Not exam material

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



CTF-RU

86 Online 411 Members

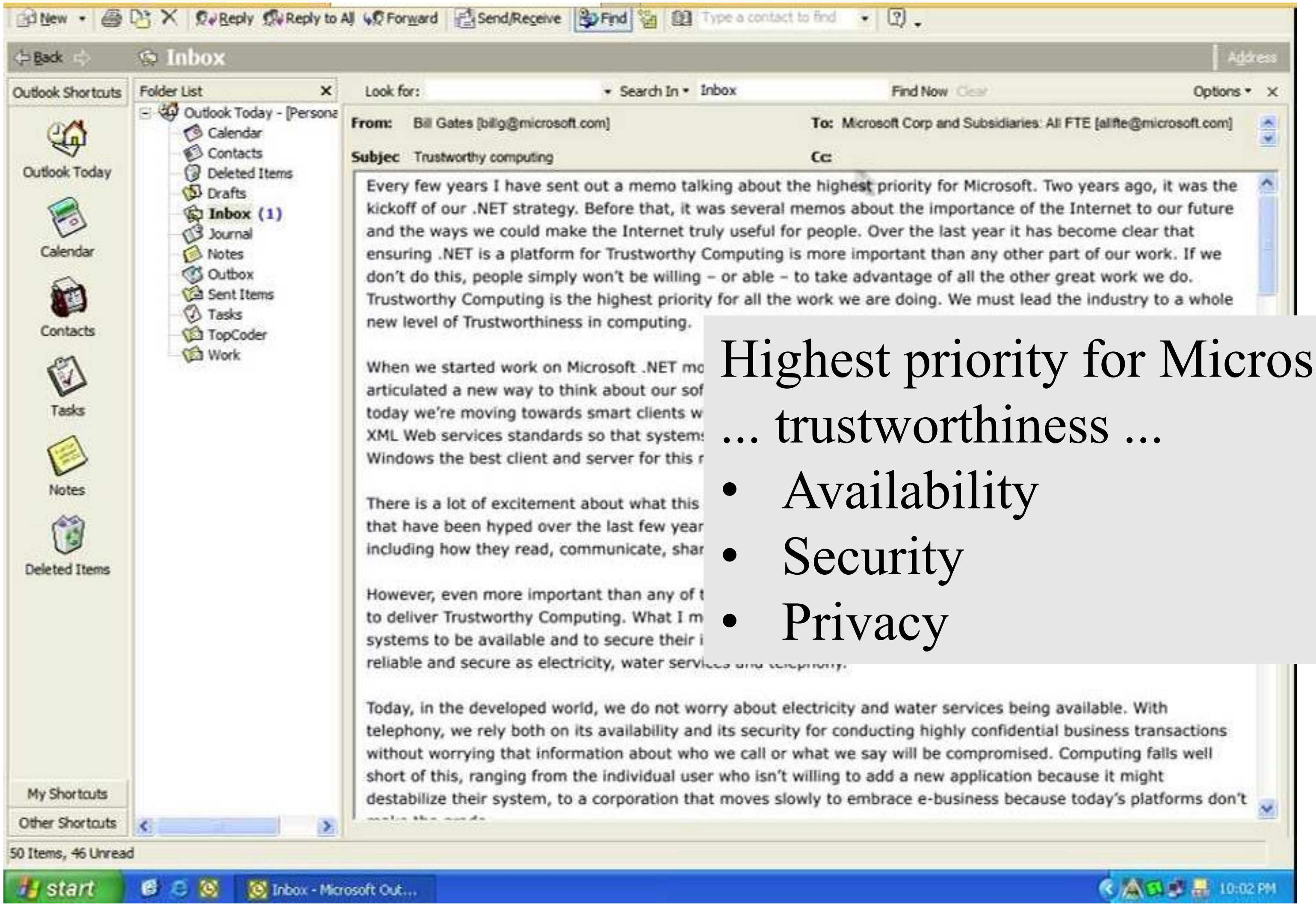
And then maybe participate in HALON or NymaCon

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

RISKY.BIZ
It's a jungle out there

Motivation & Background

A brief history of software security: January 2002



From: Bill Gates [billg@microsoft.com]

To: Microsoft Corp and Subsidiaries: All FTE [allfte@microsoft.com]

Subjec Trustworthy computing

Cc:

Every few years I have sent out a memo talking about the highest priority for Microsoft. Two years ago, it was the kickoff of our .NET strategy. Before that, it was several memos about the importance of the Internet to our future and the ways we could make the Internet truly useful for people. Over the last year it has become clear that ensuring .NET is a platform for Trustworthy Computing is more important than any other part of our work. If we don't do this, people simply won't be willing – or able – to take advantage of all the other great work we do. Trustworthy Computing is the highest priority for all the work we are doing. We must lead the industry to a whole new level of Trustworthiness in computing.

When we started work on Microsoft .NET mo
articulated a new way to think about our sof
today we're moving towards smart clients w
XML Web services standards so that system
Windows the best client and server for this r

There is a lot of excitement about what this
that have been hyped over the last few year
including how they read, communicate, shar

However, even more important than any of t
to deliver Trustworthy Computing. What I m
systems to be available and to secure their i
reliable and secure as electricity, water services and telephony.

Today, in the developed world, we do not worry about electricity and water services being available. With telephony, we rely both on its availability and its security for conducting highly confidential business transactions without worrying that information about who we call or what we say will be compromised. Computing falls well short of this, ranging from the individual user who isn't willing to add a new application because it might destabilize their system, to a corporation that moves slowly to embrace e-business because today's platforms don't

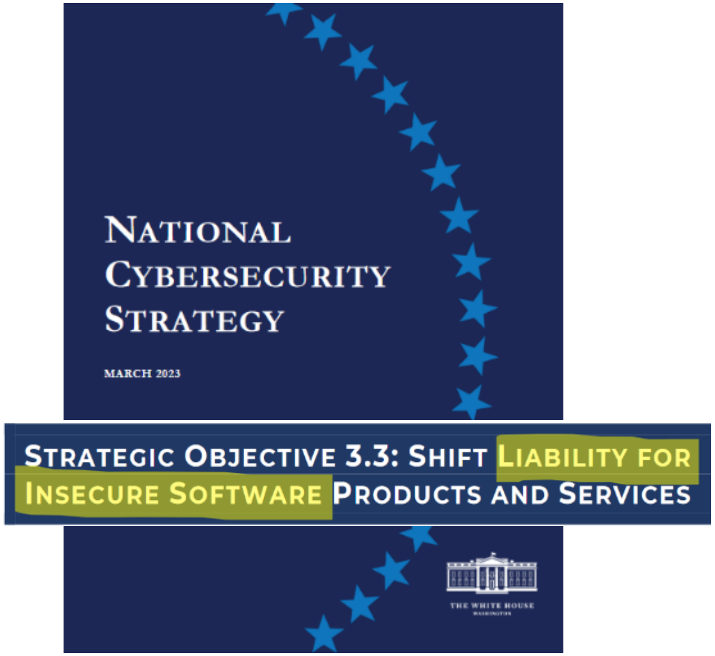
Highest priority for Microsoft:
… trustworthiness …

- Availability
- Security
- Privacy

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

Twenty years later (Sept 2022 & May 2023)

EU & US announce regulation for software security



STRATEGIC OBJECTIVE 3.3: SHIFT LIABILITY FOR INSECURE SOFTWARE PRODUCTS AND SERVICES

"Products with digital elements shall be made available on the market *without known exploitable vulnerabilities* "

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*

How do computer systems get ‘hacked’?

By attacking

- software

```
require [illegible] 'ylaggoam/ylag_styless.php'
$renderer     = $document [illegible] 'module'
$options      = array( 'style' => 'raw' )
$module       = [illegible] 'mod_menu'
$topmenu      = false  $subnav = false  $sidenav = false
Main Menu
if ($default_menu_style == 1 or $default_menu_style == 2)
        $module [illegible] = 'menutype=$menu_name [illegible]
        $topmenu = $renderer [illegible] $module, $options
        $menuclass = 'horiznav'
        $topmenuclass = 'top_menu'
elseif ($default_menu_style == 3 or $default_menu_style == 4)
        $module [illegible] = 'menutype=$menu_name [illegible]
        $topmenu = $renderer [illegible] $module, $options
        $menuclass = 'horiznav_d'
        $topmenuclass = 'top_menu_d'
SPLIT MENU  NO SUBS
elseif ($default_menu_style == 5)
        $module [illegible] = 'menutype=$menu_name [illegible]
        $topmenu = $renderer [illegible] $module, $options
        $menuclass = 'horiznav'
        [illegible] = 'top_menu'
```

- humans

Blaming 'stupid users' is victim blaming: if users do not use a system securely, this is an IT design flaw

Or: interaction between software & humans

- crypto
- hardware

What is software security?

Intersection of security engineering & software engineering:

- *prevent* *design*-level & *implementation*-level security vulnerabilities and pro-actively design & build systems that resist attacks
- *reduce the chance* of users harming themselves & others by bad security choices
  - NB programmers and sys admins are also users
- *detect* vulnerabilities that arise - *accidentally* or *intentionally* - and react to them
- *mitigate* risks before and after detecting problems

Changing nature of attackers

Originally, hackers were amateurs motivated by ‘fun’

- by script kiddies & more skilled hobbyists

Nowadays, hackers are professional:

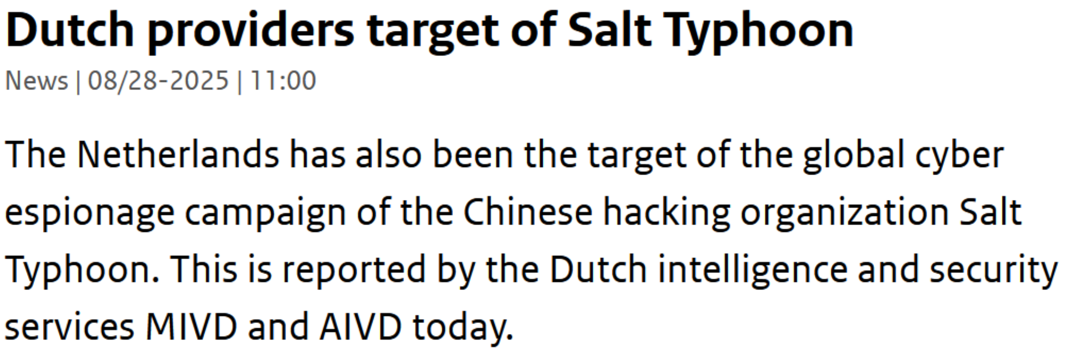
- cyber criminals

  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, ...

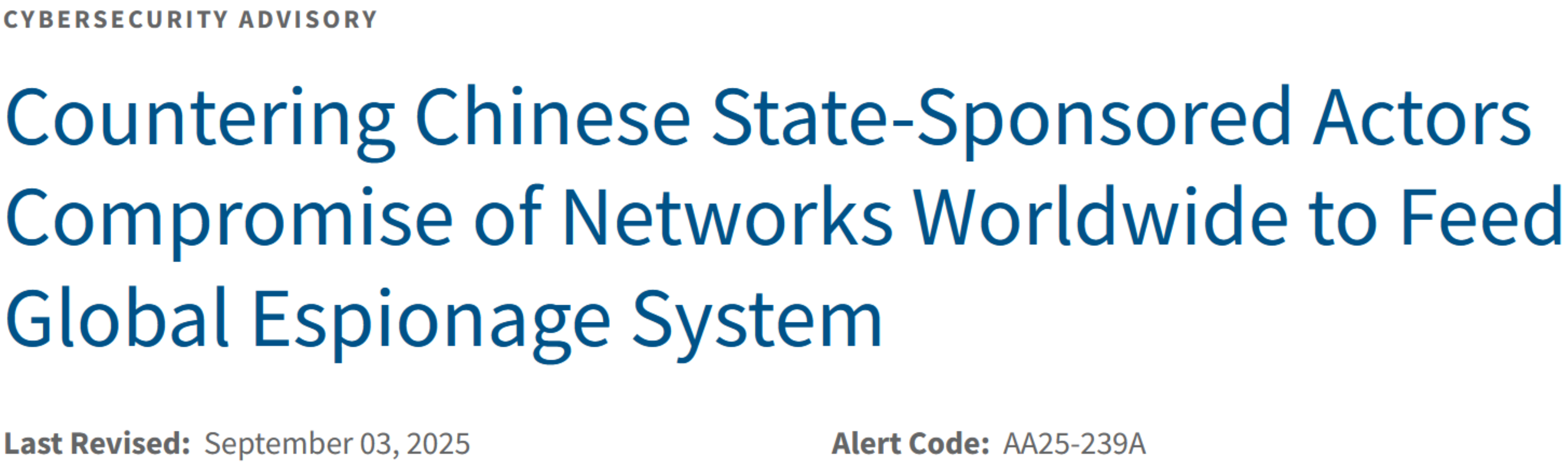


Dutch providers target of Salt Typhoon

News | 08/28-2025 | 11:00

The Netherlands has also been the target of the global cyber espionage campaign of the Chinese hacking organization Salt Typhoon. This is reported by the Dutch intelligence and security services MIVD and AIVD today.

Algemene Inlichtingen- en Veiligheidsdienst
Ministerie van Binnenlandse Zaken en Koninkrijksrelaties

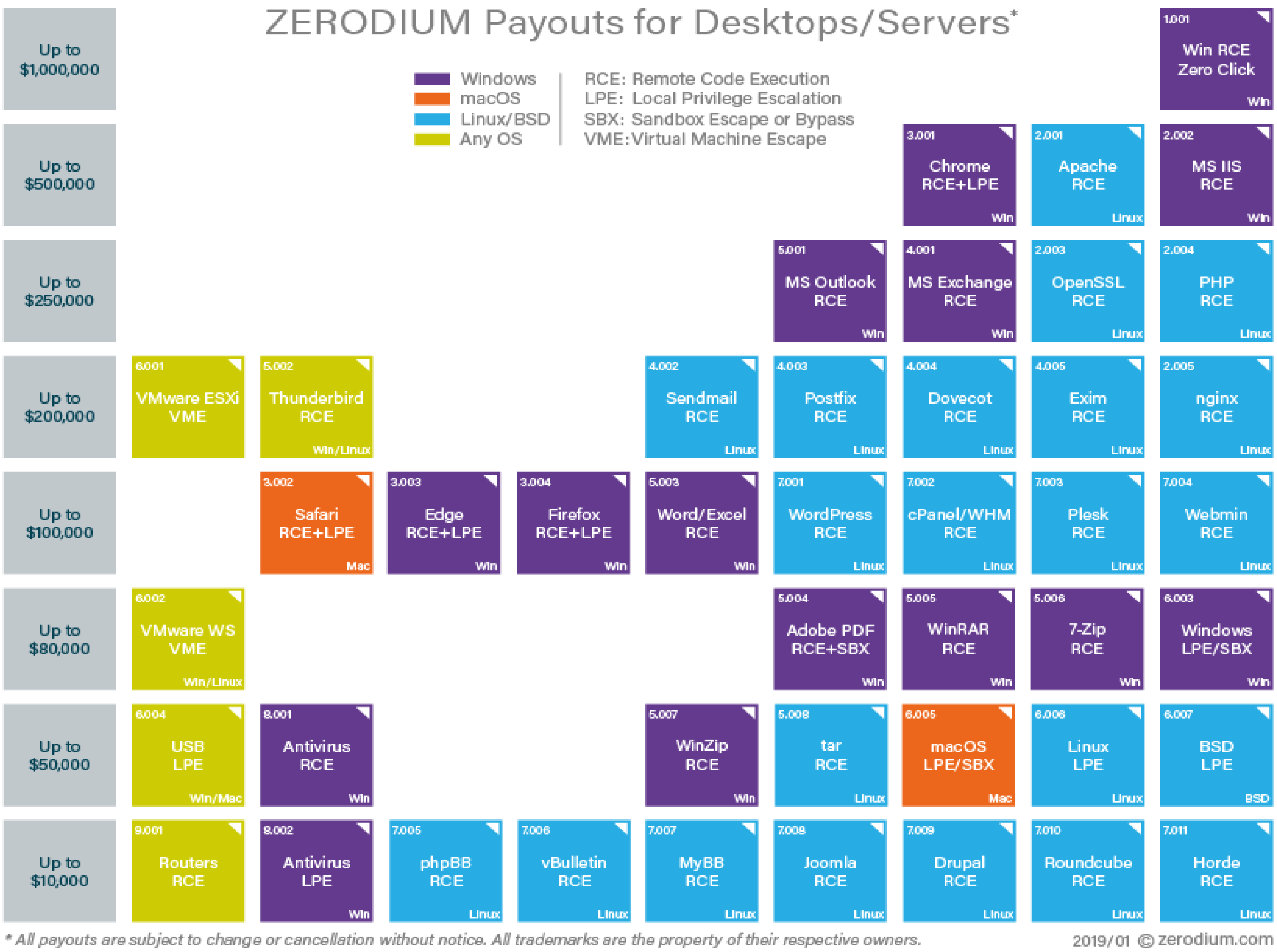


CYBERSECURITY ADVISORY

Countering Chinese State-Sponsored Actors Compromise of Networks Worldwide to Feed Global Espionage System

Last Revised: September 03, 2025 **Alert Code:** AA25-239A

Prices for 0days



ZERODIUM Payouts for Desktops/Servers*

🟪 Windows
🟧 macOS
🟦 Linux/BSD
🟨 Any OS

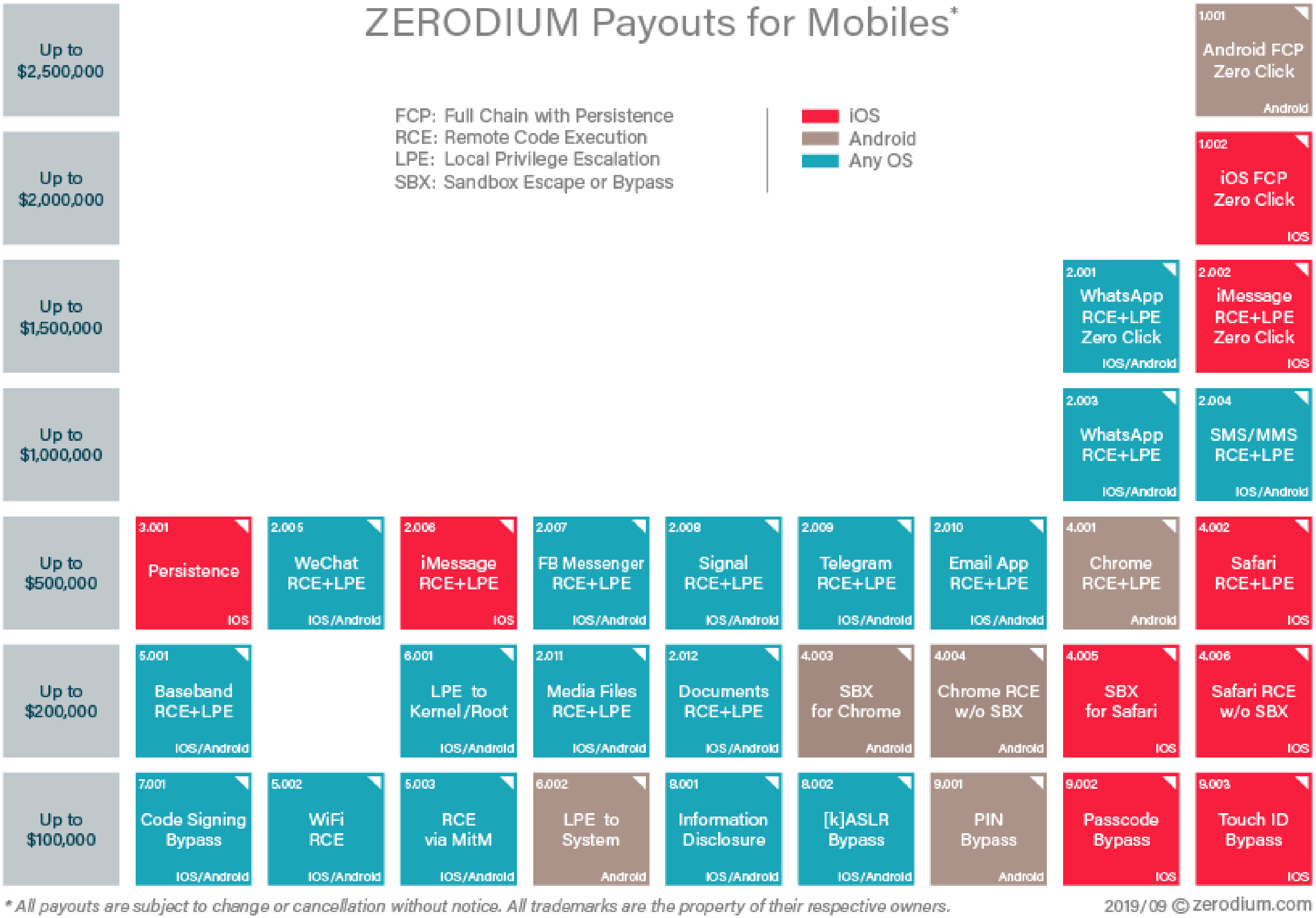
RCE: Remote Code Execution
LPE: Local Privilege Escalation
SBX: Sandbox Escape or Bypass
VME: Virtual Machine Escape

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

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

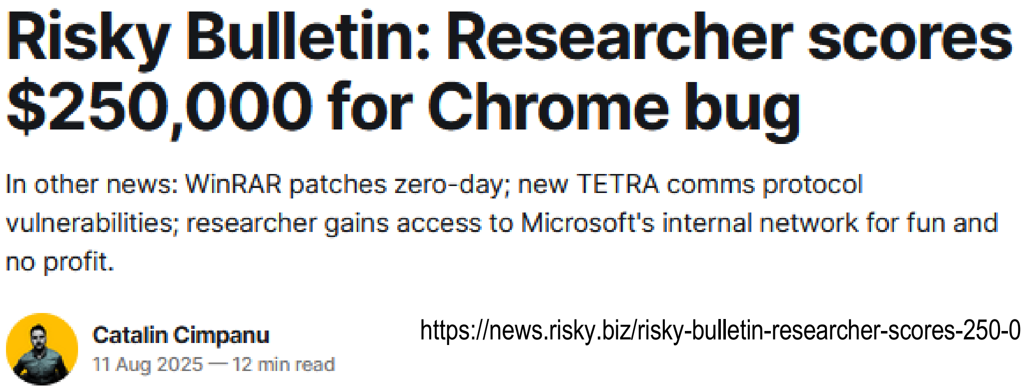
🟥 iOS
🟫 Android
🟦 Any OS

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.*

2019/09 © zerodium.com

Google Chrome bug bounty payouts

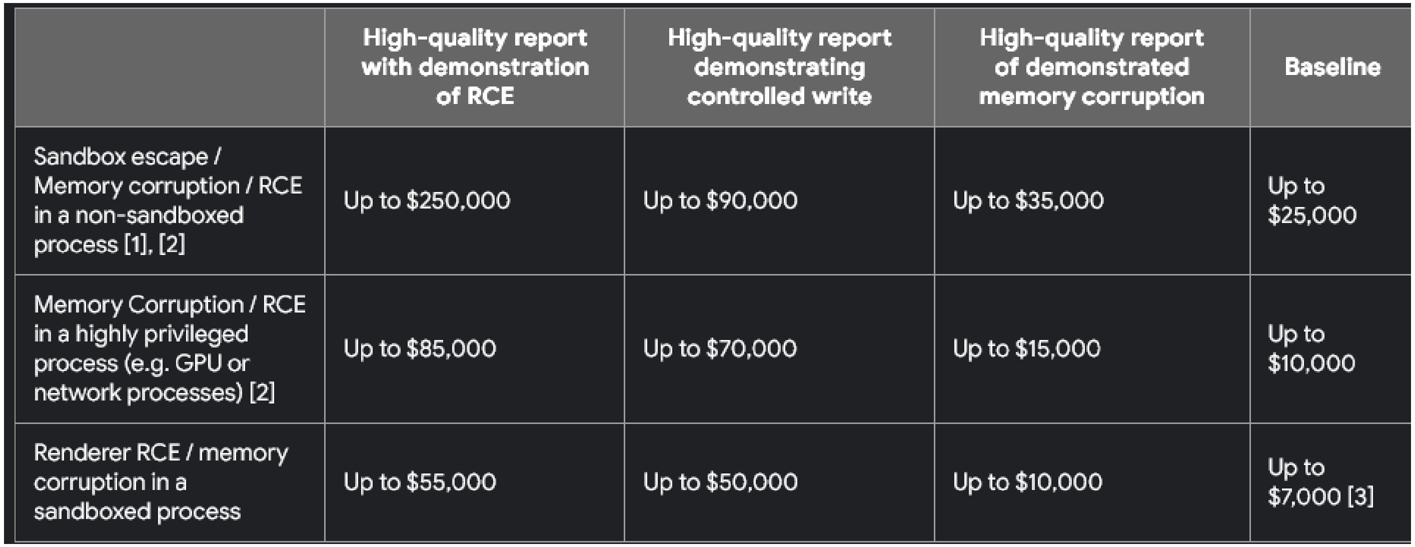


Risky Bulletin: Researcher scores $250,000 for Chrome bug

In other news: WinRAR patches zero-day; new TETRA comms protocol vulnerabilities; researcher gains access to Microsoft's internal network for fun and no profit.

Catalin Cimpanu
11 Aug 2025 — 12 min read

https://news.risky.biz/risky-bulletin-researcher-scores-250-000-for-chrome-bug/



	High-quality report with demonstration of RCE	High-quality report demonstrating controlled write	High-quality report of demonstrated memory corruption	Baseline
Sandbox escape / Memory corruption / RCE in a non-sandboxed process [1], [2]	Up to $250,000	Up to $90,000	Up to $35,000	Up to $25,000
Memory Corruption / RCE in a highly privileged process (e.g. GPU or network processes) [2]	Up to $85,000	Up to $70,000	Up to $15,000	Up to $10,000
Renderer RCE / memory corruption in a sandboxed process	Up to $55,000	Up to $50,000	Up to $10,000	Up to $7,000 [3]

https://bughunters.google.com/blog/5302044291629056/chrome-vrp-reward-updates-to-incentivize-deeper-research

Software security: crucial facts

- *There are no silver bullets!*

  Firewalls, anti-virus, 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 by Design: security should be considered right from the start & throughout the development lifecycle*

Security software ≠ Software security

Adding security software can make a system more secure

i.e. software specifically for security, such as

- access control,with authentication & authorisation
- TLS, IPSEC, VPN, …
- AV (AntiVirus), firewall, WAF (Web Application Firewall)
- access control
- NIDS (Network Intrusion Detection System)
- EDR (Endpoint Detection & Response, eg CrowdStrike)
- …

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

Root cause: 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

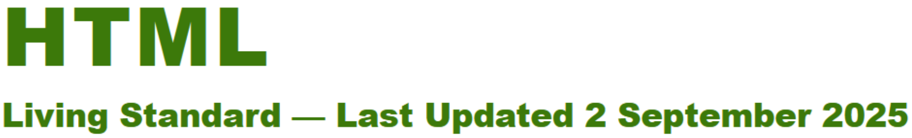
- security
- functionality, convenience, speed, …

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?*

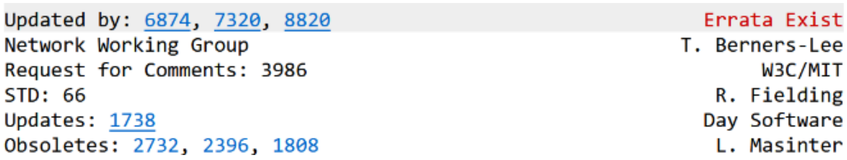


HTML

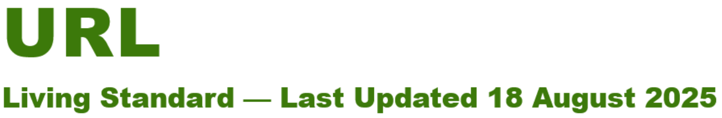
Living Standard — Last Updated 2 September 2025

- *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
```



URL

Living Standard — Last Updated 18 August 2025

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

FUNCTIONALITY & COMPLEXITY vs security
Lost battles?

- Programming languages & APIs

  we want these to be 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

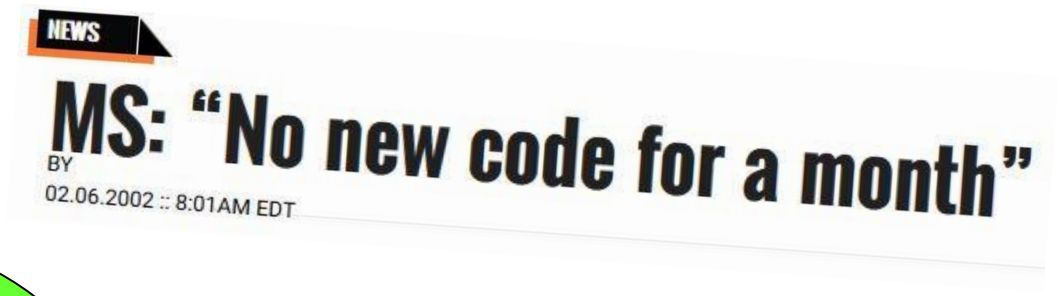
Recap

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

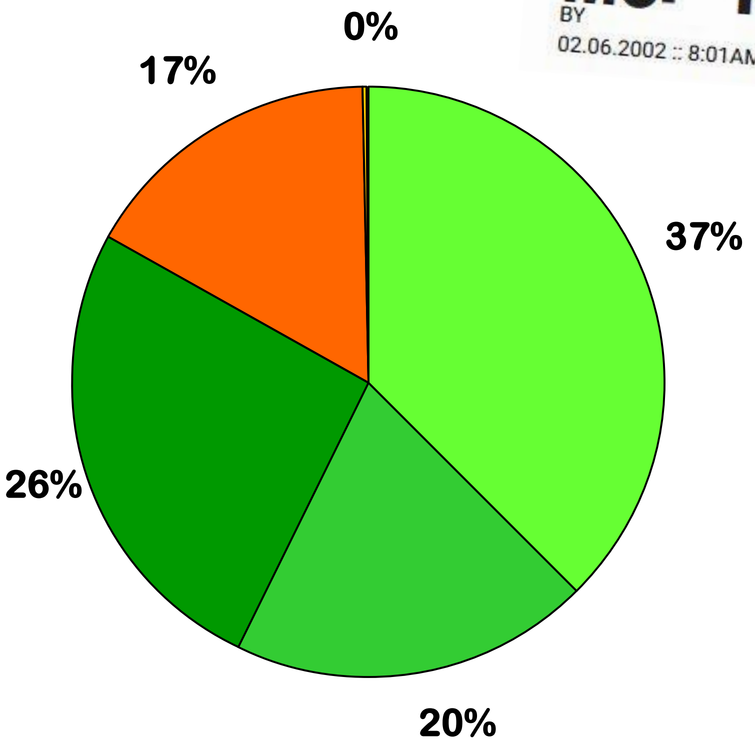


NEWS

MS: “No new code for a month”

BY

02.06.2002 :: 8:01AM EDT



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 *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 covered 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

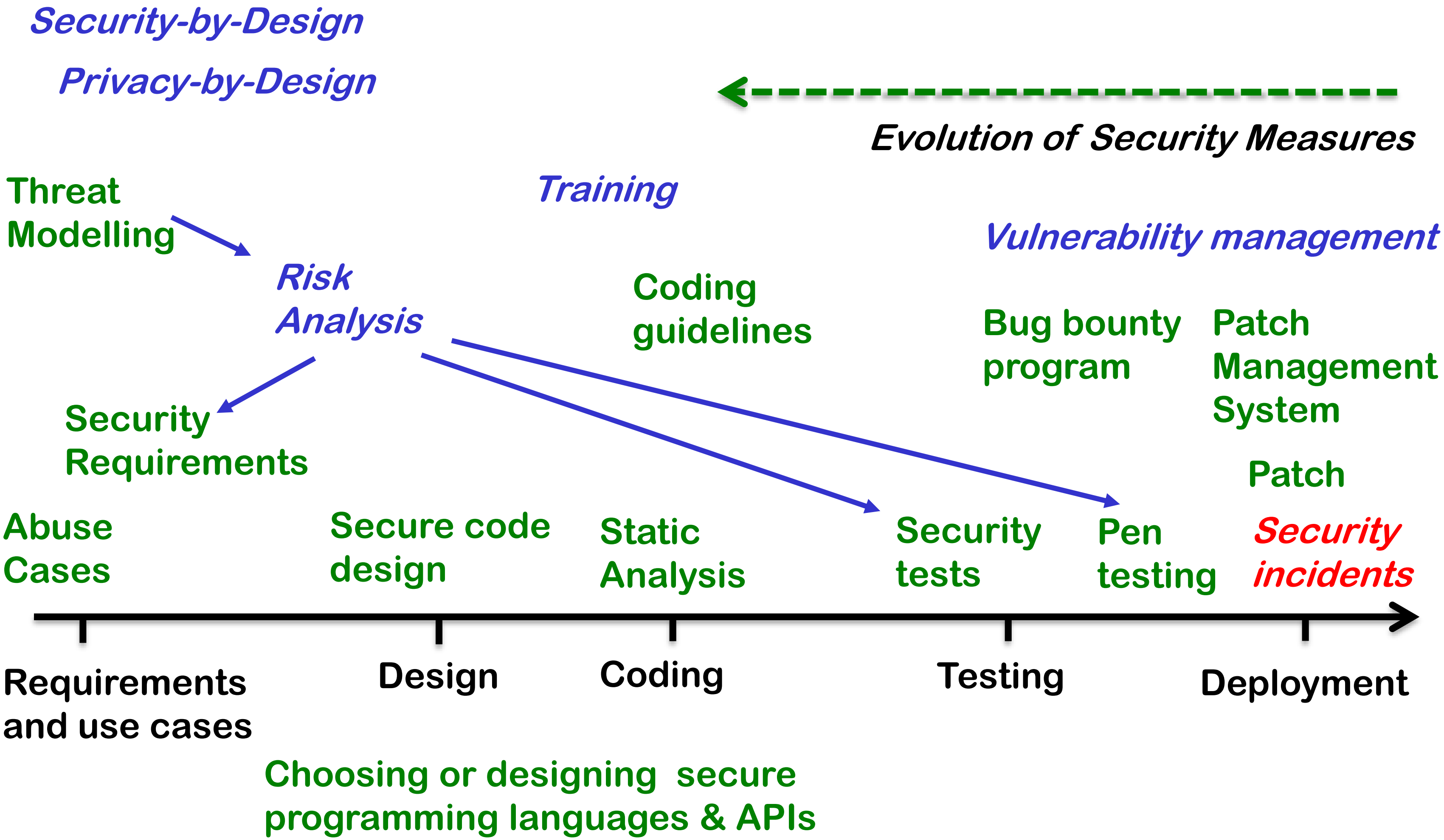
- have vulnerabilities that have not been found (yet)
- overlook attack vectors
- make implicit assumptions that are – or become – invalid
- overlook ways in which functionality can be abused
- miss security properties that are important
- …

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



Security-by-Design

Privacy-by-Design

Evolution of Security Measures

Threat Modelling

Training

Vulnerability management

Risk Analysis

Coding guidelines

Bug bounty program

Patch Management System

Security Requirements

Patch

Abuse Cases

Secure code design

Static Analysis

Security tests

Pen testing

Security incidents

Requirements and use cases

Design

Coding

Testing

Deployment

Choosing or designing secure programming languages & APIs

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. If security issue is discovered, then a) still do nothing, if there's no (economic) incentive; b) sue the people who reported; or c) patch
3. If this happens often: make update mechanism for regular patching
4. Do security testing, maybe hire pen-testers or have bug bounty program
5. Use static analysis tools when coding
6. Give security training to programmers
7. Think of security in software design
8. Think of security when choosing programming language & APIs
9. Think of security when *designing* programming languages & APIs
10. Think of abuse cases, and develop security tests for them
11. Think about security *before* you start coding, eg with security architecture review
12. ...

Ever more acronyms for tools

- DAST (Dynamic Application Security Testing)

  ie. security testing
- SAST (Static Application Security Testing)

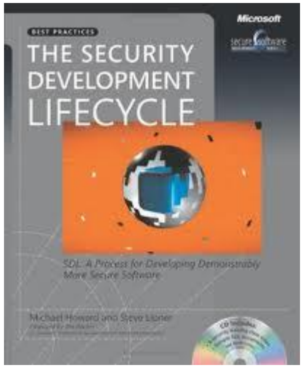
  ie. static analysis
- SCA (Software Composition Analysis)

  looking for known flawed software components
- Secret Scanners

  for leaked credentials (eg API keys) in cloud infra or code repos

- IAST (Interactive Application Security Testing)
  - tools to help in manual pen-testing
- RASP (Run-time Application Security Protection)
  - instrumentation to do runtime monitoring

Secure software development lifecycles


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

Methodologies

- Microsoft SDL [2004]

  with extension for secure DevOps (DevSecOps)
- Touchpoints by Gary McGraw [2004]
- NIST SSDF (Secure Software Development Framework) [2022]
- Grip op SSD (Secure Software Development) by Dutch government organisations https://www.cip-overheid.nl/en/category/products/secure-software

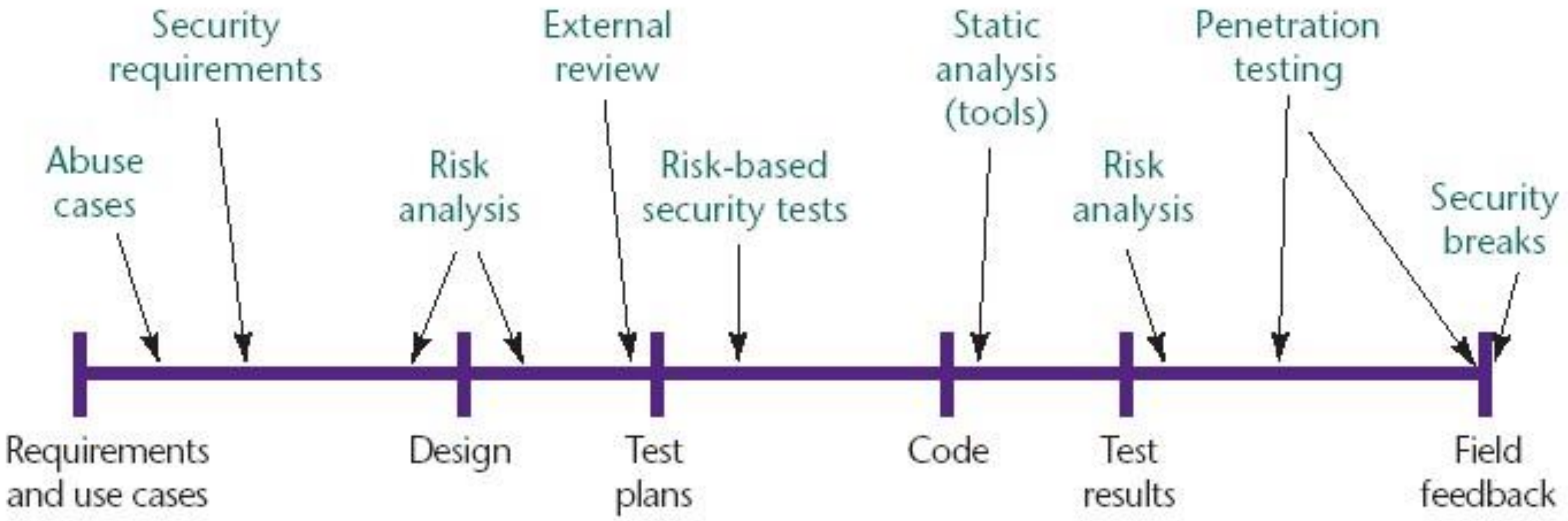
Maturity models

- SAMM (Software Assurance Maturity Model) by OWASP
- BSIMM by Synopsys

These security guidelines for the process are then complemented with security guidelines for the product : Top N lists of common security flaws, coding guidelines, security design patterns, …

Security in the software development life cycle

McGraw’s Touchpoints

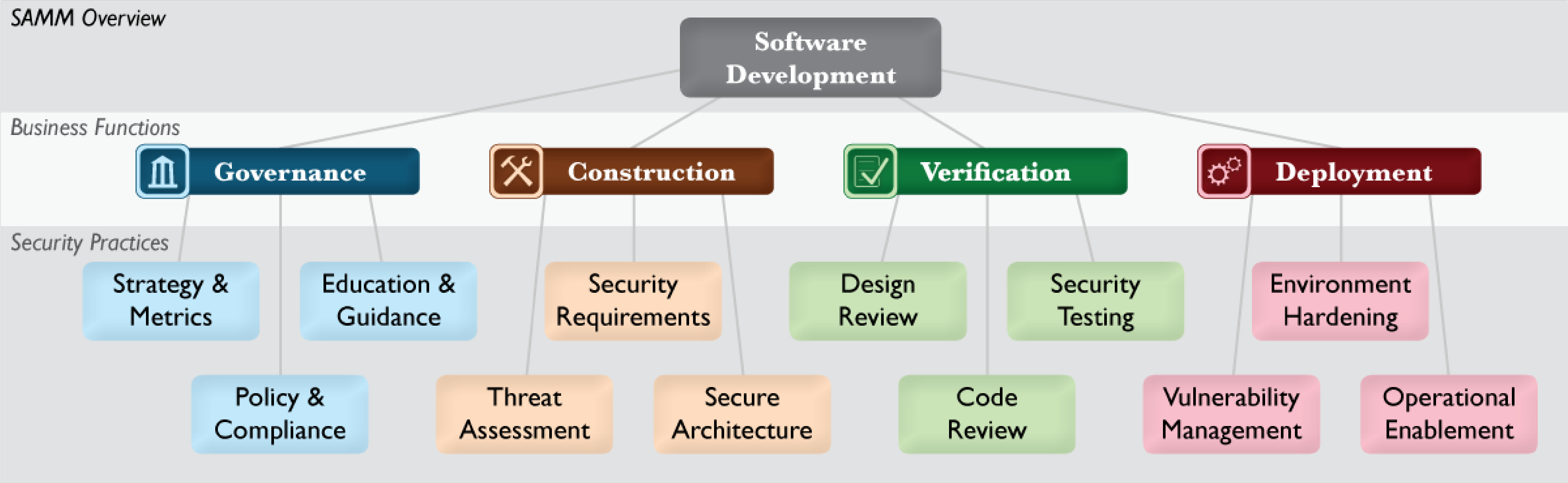


[Source: Gary McGraw, *Software security*, Security & Privacy Magazine, IEEE, Vol 2, No. 2, pp. 80-83, 2004.]

OpenSAMM

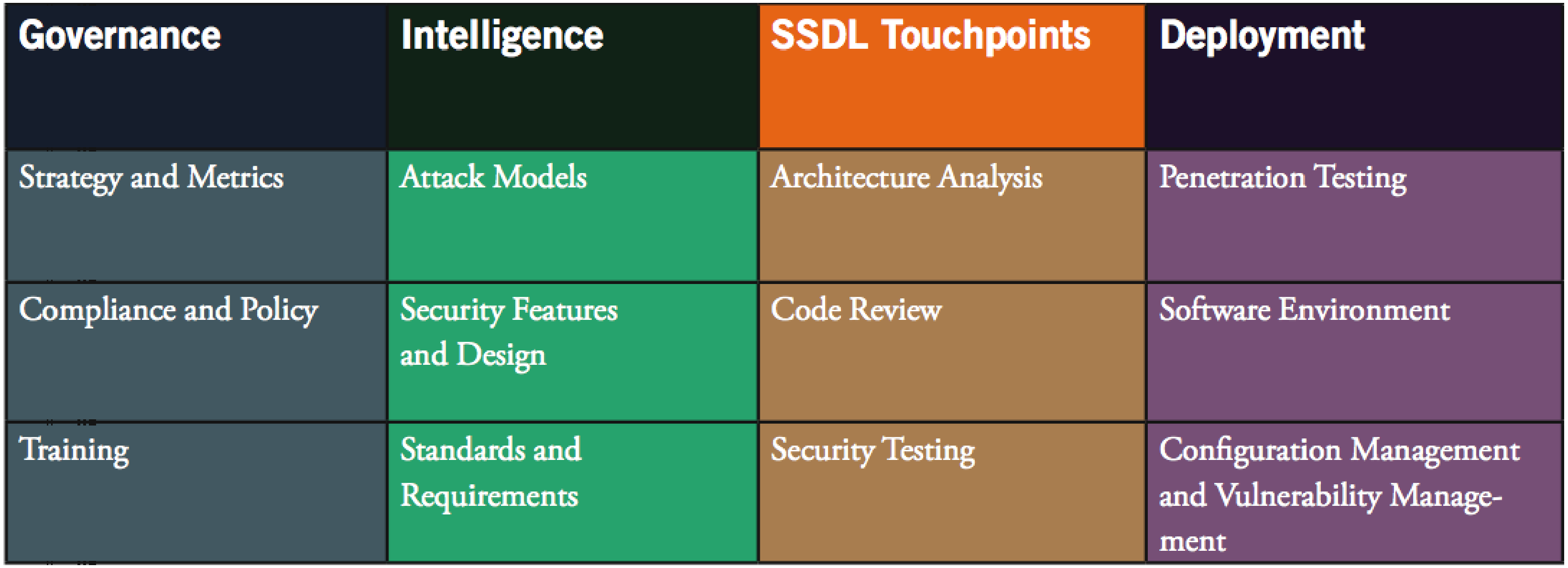
OPENSAMM

12 security practices grouped in 4 business functions



BSIMM (Building Security In Maturity Model)

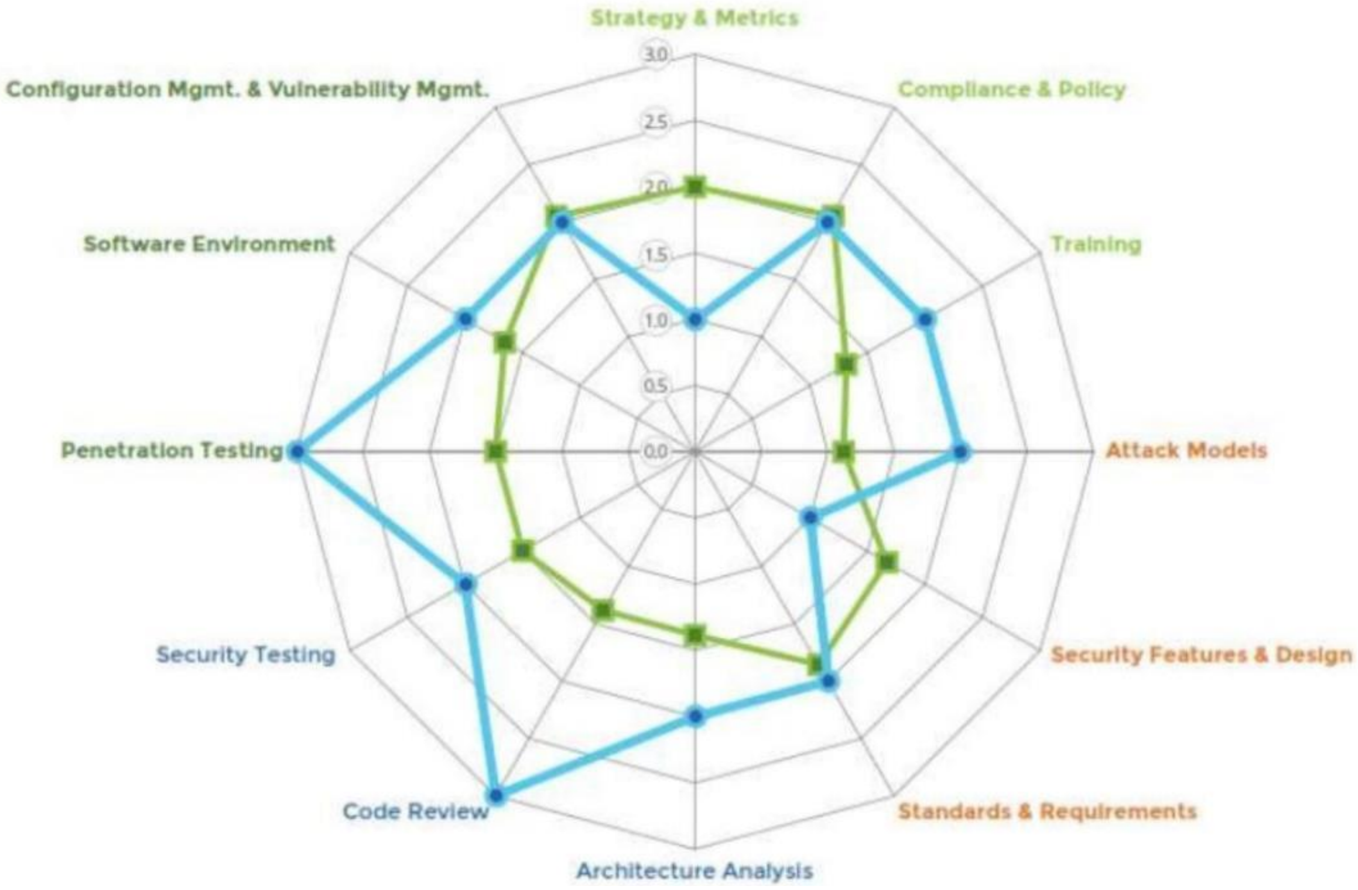
126 activities in 12 practices across 4 domains



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 Management

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 or controls?

Aka threat modelling

Security requirements

a) ‘This application cannot be hacked’

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

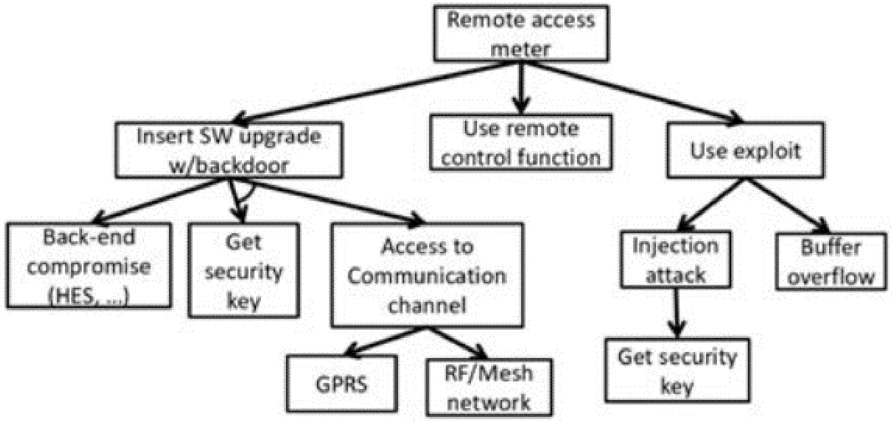
b) More specific security requirements

- In terms of Confidentiality, Integrity Availability (CIA)
- Or, usually better, in terms of Access Control
  - i.e. Authentication & Authorisation
  - also Monitoring & Response, so not just prevention
  - mnemonic: AAAA for Authentication, Authorisation, Auditing, Action
- ‘Positive’ security model

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


Remote access meter
Insert SW upgrade w/backdoor
Use remote control function
Use exploit
Back-end compromise (HES, ...)
Get security key
Access to Communication channel
Injection attack
Buffer overflow
GPRS
RF/Mesh network
Get security key

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:
   - Section 2 & 4.1 of Secure Software Lifecycle by Laurie Williams
   - Sections 1-3 of Twenty Years of Secure Software Development

2. To do: check out
   - the latest US-CERT bulletin
   - recent CVEs for the browser, PDF viewer and other software you use on a regular basis

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;
}
```