Security Through the Software Lifecycle (Lecture 2 part 1)

Prof. Mathy Vanhoef

DistriNet – KU Leuven – Belgium

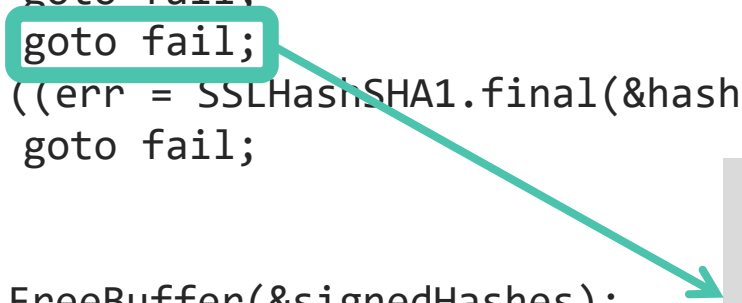
KU LEUVEN DistriNet

More motivation: Avoidable security issues…

Vulnerabilities due to bad development processes

Selected example is the Apple “goto fail” vulnerability:

```
static OSStatus SSLVerifySignedServerKeyExchange(...) {
    ...
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;
    ...
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
}
```



This “goto fail” is always executed

Vulnerabilities due to bad development processes

Selected example is the Apple “goto fail” vulnerability

- Machine-in-the-Middle can decrypt TLS connections

Rather surprising how this “goto fail” ended up there

- Several processes could have prevented this
  - Coding guidelines (e.g., always use braces)
  - Static code analysis (e.g., detect unreachable code)
  - Reviewing patches, negative tests, etc.
- Software lifecycle should include more security checks?!

Log4Shell vulnerability in Log4j

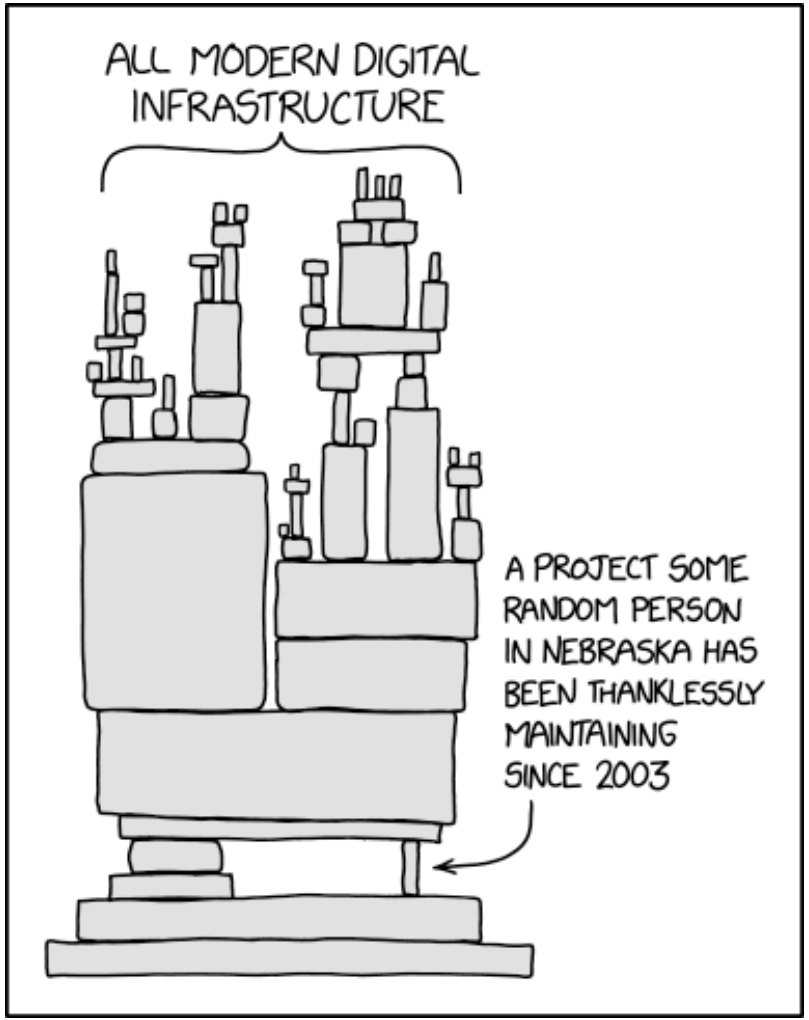
- Attacker-controlled data in log output can be abused to exfiltrate data and execute code

This highlights:

- Importance of **threat modelling**: input should be untrusted
- Disable **unused features** when possible
- **Track/audit/update** your 3rd party libraries

3rd party libraries

- In general, it's hard to be aware of all dependencies
- And some of them may be insecure…
- …and maintained by just a single person



Source: https://xkcd.com/2347/

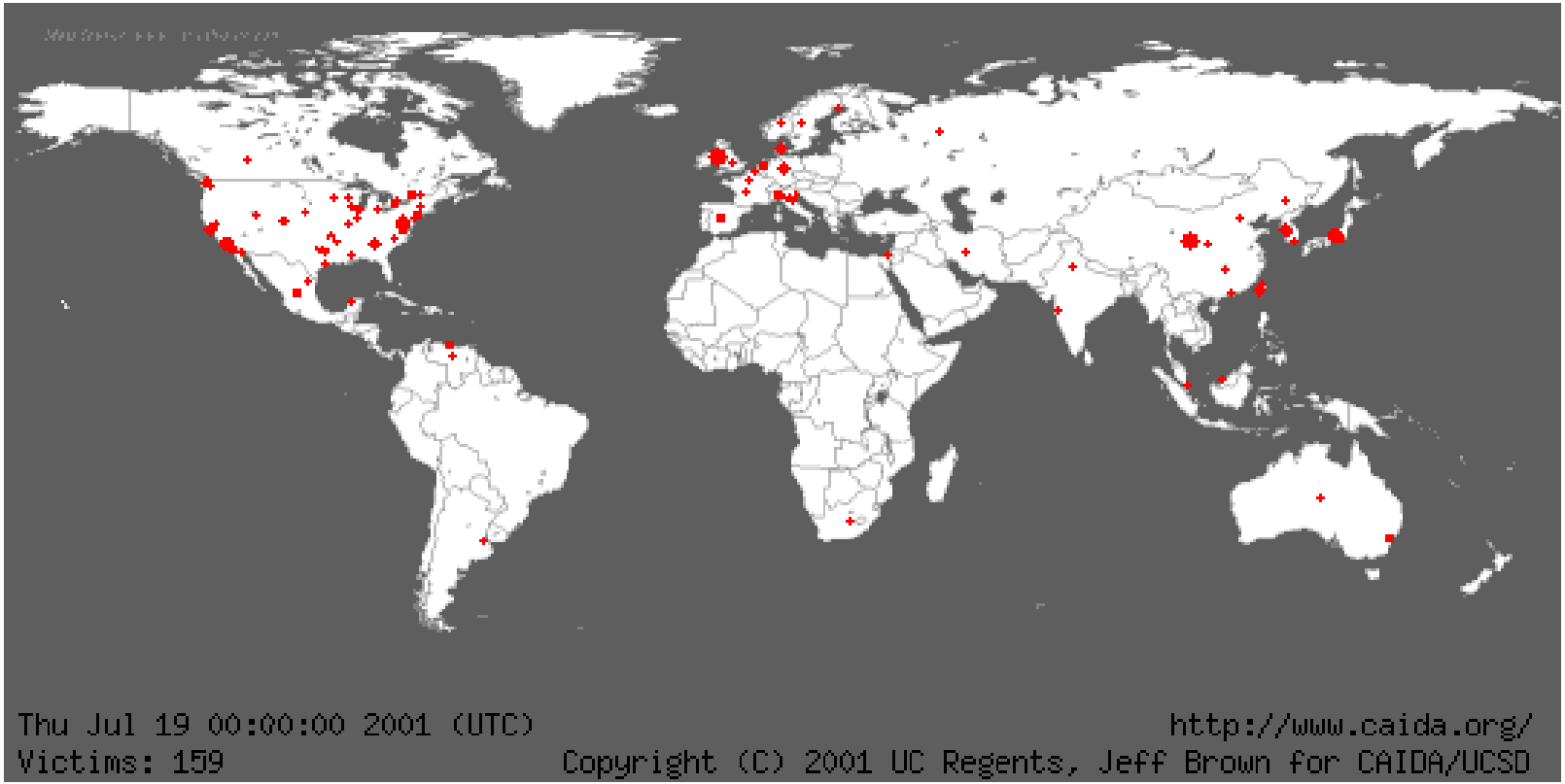
History behind Microsoft SDL

Microsoft: Code Red

July 2001: Code Red computer worm

- Exploited a buffer overflow in Microsoft IIS web servers
- Named Code Red because it defaced websites with “Hacked by Chinese” & because analysts were drinking “Code Red” Mountain Dew
- 1st version (CRv1) detected on July 12 and caused little damage
- On July 19 at 10:00 UTC, a 2nd version (CRv2) was detected that fixed a bug so it could spread more widely
- Launches a denial-of-service against www1.whitehouse.gov from the 20-28th of each month

Code Red infection rate



Source: https://www.caida.org/archive/code-red/coderedv2_analysis/

August 2001: CodeRedII

- Also exploits the buffer overflow in Microsoft IIS web servers
- Entirely **new code**. Source code contained the string “CodeRedII”.
- **Installs a backdoor** on the system that gives remote administrator-level access to the infected machine

→ All combined, estimated to have cost $2.6 billion in damage

Optional more info / sources:

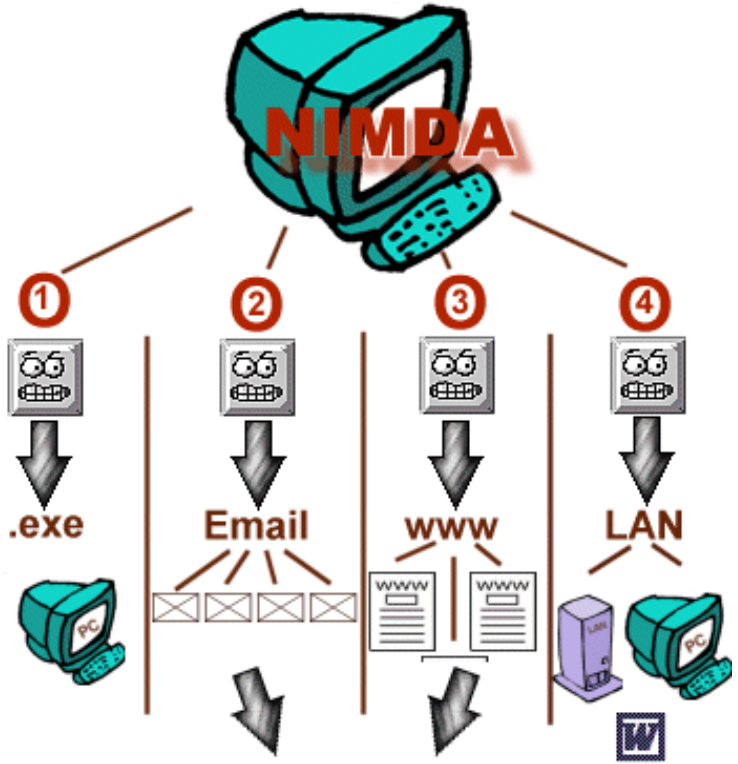
- Moore, David, Colleen Shannon, and K. Claffy. "Code-Red: a case study on the spread and victims of an Internet worm." Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment. 2002.
- CAIDA Analysis of Code-Red
- The Code Red Worm | Hacking History Documentary

Microsoft: Nimda

September 2001: Nimda worm

- Uses multiple methods and vulnerabilities to spread itself
- Its **only goal is to spread itself**. Doesn't delete files and doesn't try to cause any other harmful effects.
- But it **did slowdown computers** and caused network traffic.
- Name "Nimda" is Admin backwards. Because it creates an admin account and creates a file named "Admin" on infected computers.
- Caused roughly $635 million in damage

NIMDA used multiple spreading techniques



1. Infects .exe programs. When they are shared the virus spreads.
2. E-mails itself to all your contacts.
3. Scans the internet for servers & tries to infect them using several vulnerabilities. Infects selected files hosted on the webserver.
4. Scans for file shares in the local network and drops infected files

Source: https://www.f-secure.com/v-descs/nimda.shtml

Memo from Bill Gates in early 2002

After Code Red and Nimda, Bill Gates wrote a famous memo:

- “emphasize **security right out of the box** [..] and constantly refine and improve that security as threats evolve.”
- “vision for **Trustworthy Computing**”
- “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.”

→ Evolved into the Security Development Lifecycle (SDL)

Microsoft SDL history

2002 - 2003	• Bill Gates writes “Trustworthy Computing” memo early 2002 • “Windows security push” for Windows 2003
2004	• SDL is required for all products that process meaningful risk and/or process sensitive data
2005 - 2006	• SDL improvements: fuzzing, more static code analysis, crypto design requirements, privacy, banned APIs, and more. • Increased external visibility of SDL. Customers are asking more info about SDL (education, tools, support, etc).
2019	• Optimize process through feedback, analysis, and automation • Evangelize the SDL

Our previous lecture!

Source: https://www.slideshare.net/RaziRais/microsoft-security-development-lifecycle-175687514 , Sept. 23, 2019. At Nigel Frank International.

But security remains an issue

Help Net Security
July 27, 2022

Share

Amazon's Twitch blames

data

The data time

INVESTMENT BANKING | LEGAL/REGULATORY

JPMorgan Chase Hacking Affects 76 Million Households

BY JESSICA SILVER-GREENBERG, MATTHEW GOLDSTEIN AND NICOLE PERLROTH OCTOBER 2, 2014 12:50 PM

528

Marriott Hotels fined £18.4m for

data breach that hit millions

Attacks/Breaches | 4 MIN READ | ARTICLE

Blizzard Battle.net Security Breached, Passwords Accessed

ram,

be User

n Massive

Data Leak

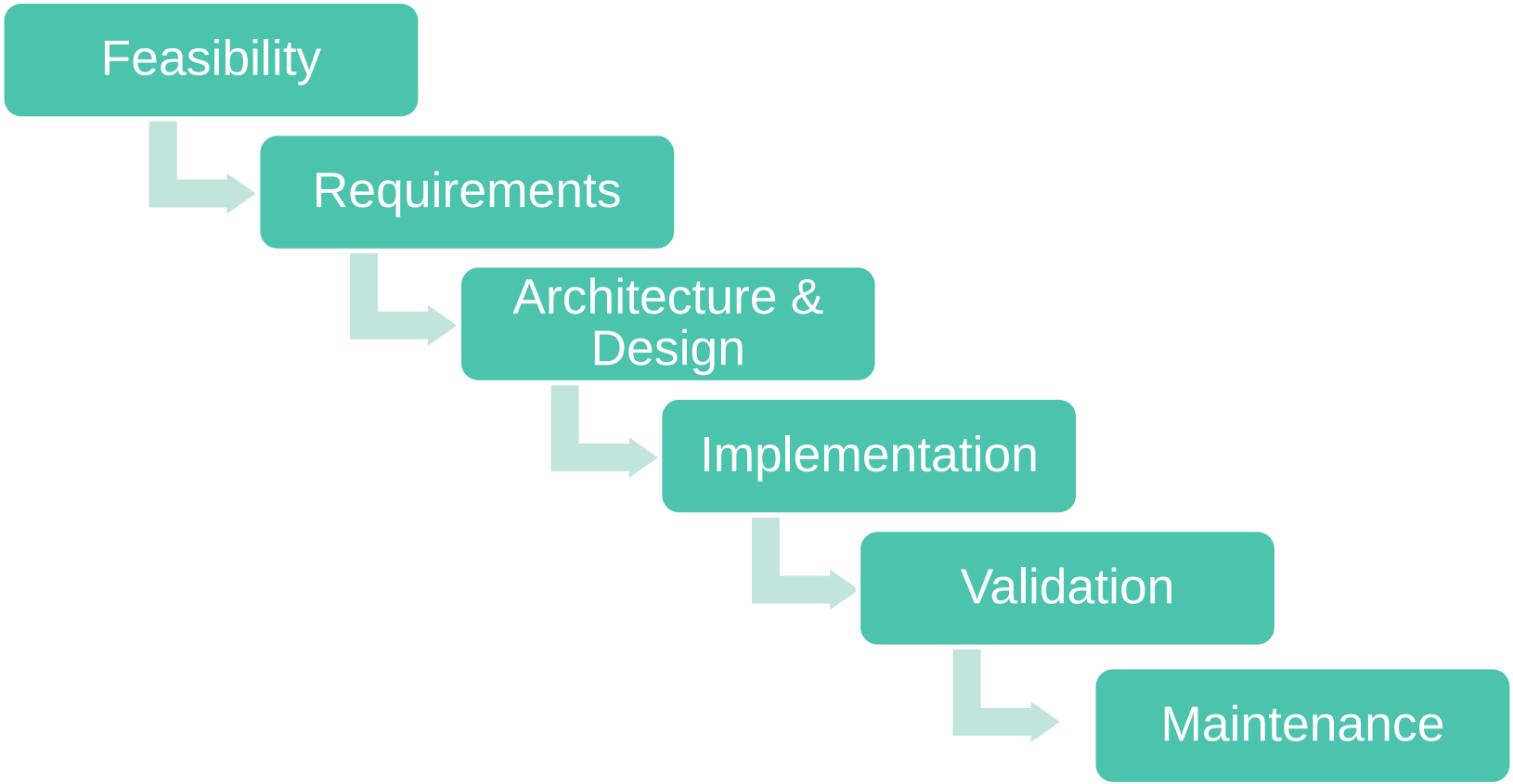
Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC)

Before securing the software development life cycle, we need to know what it is. There are three major alternatives:

- Sequential models (e.g., waterflow model)
  - Complete each step before moving to the next
- Iterative models (e.g., iterative refinement)
  - Create rough system, then repeat to improve the system
- Incremental models (e.g., agile development)
  - Small increments of software are placed in production (sprints)
  - Consists of several elements: sprints, SCRUM, test-first design,…

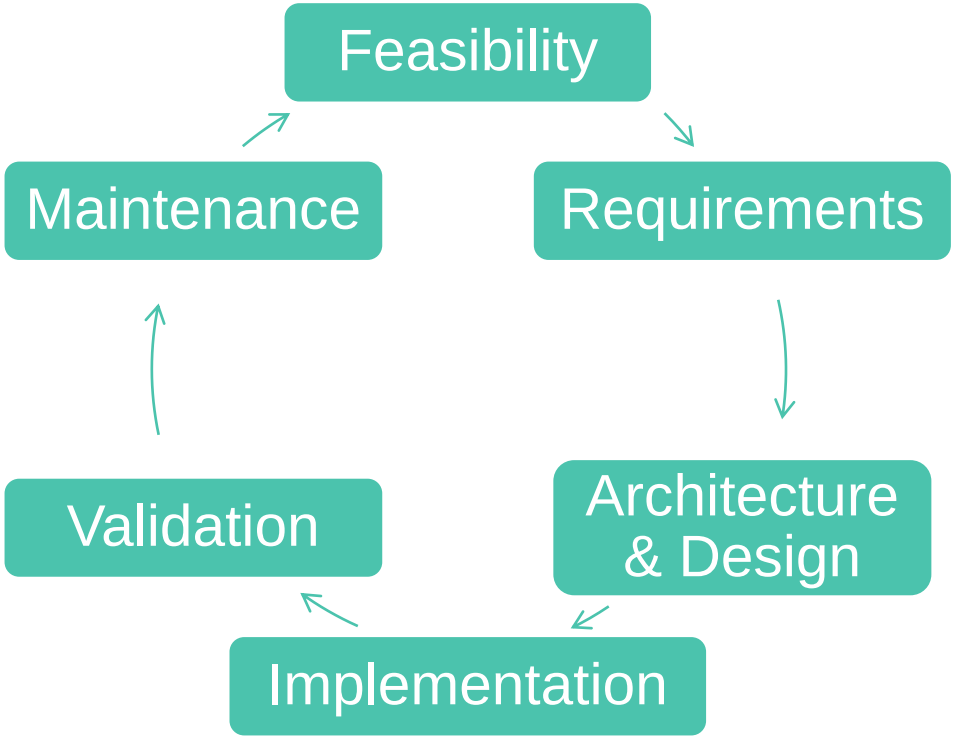
Sequential models (e.g., waterflow model)



Sequential models (e.g., waterflow model)

- One step (must) complete before next one starts
- Rational process that enables careful planning
  - Good for systems that cannot be easily changed (e.g., hardware)
- In practice you rarely know everything up front
  - Hard to state all requirements explicitly
  - Typically a slow process
  - Any blunder can be disastrous
  - Real projects rarely follow a sequential model

Iterative models (e.g., iterative refinement)



- System is created by successive versions.
- Lowers the cost of changing requirements
- Allows some **client/user feedback** to be considered
- A release is created sooner
- Changes can lead to **messy designs & implementations**

Agile development

Development lifecycle consists of a **series of sprints**:

- Short periods (15-60 days) to design, develop, test, and then potentially delivered a set of features to customers
- **Product backlog**: features to add to the product
- **Sprint backlog**: features selected from the product backlog to implement during the next sprint

More broadly, Agile development is **a set of best practices**

- The details are not important for us

All typically include the following phases

1. Feasibility & planning
2. Requirements (analysis / specification)
3. Architecture and design
4. Implementation
5. Validation (acceptance and release)
6. Maintenance (and operations)

Sometimes different terms are used, but the idea behind the phases stays the same.