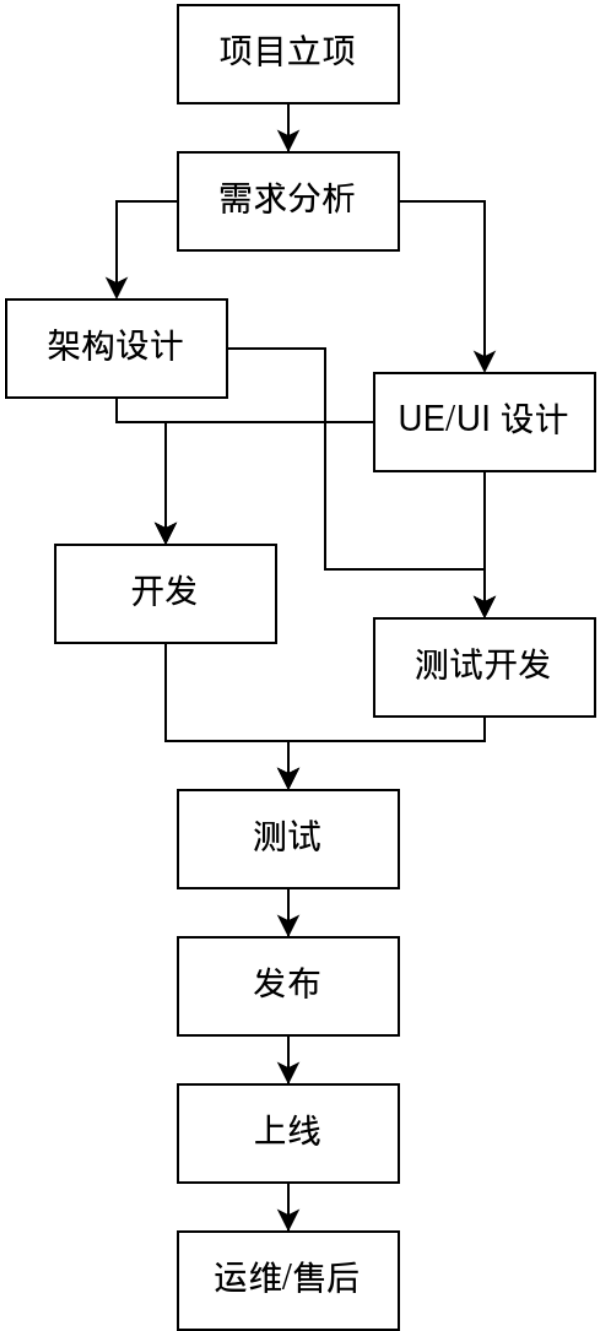
软件工程分享

目录

软件开发流程示例



市场调研，公司战略考量

具体需求调研与细化，还要考虑竞品和客户

架构设计考虑技术选型，组件间依赖与接口等

设计产品交互，观感

模块内详细设计与编码开发

测试用例，自动化测试等

不仅要考虑代码功能，还要考虑运行环境，部署等

根据实际情况，可能要发布alpha、beta等版本

上线到测试环境或生产环境

可能发生硬件、软件故障；用户不会用等各种情况

何谓工程

- 注重实践和实际应用
- 系统化的思维
  - 模型和抽象
- 跨学科协作和组织
- 项目管理
  - 用户和需求驱动
  - 时间/人员规划
  - 质量控制
  - 规范化流程

开源软件中的软件工程实例

	Linux Kernel	Openstack
模块化设计	内核子系统；驱动	计算、网络、存储等模块
版本控制	Git	Git
代码审查	邮件列表	Gerrit
自动化测试	LKDTM, LTP 等	tox, rally 等
持续集成	kernel CI 等	Zuul，Gerrit 等
文档	自动构建	自动构建
项目管理	Linux 基金会	Openstack 基金会
需求管理	邮件列表	Blueprint
代码行数	3000 万	主要项目 130 万

目录

案例：VCF 软件项目 - 概述

- 属于 Trilogy 项目的一部分，开发一套软件提高信息处理和共享能力，并实现无纸化办公作业。
- 历时 5 年开发，总共耗资 1.7 亿美元
- 项目需求数度剧变，历任四任 CIO
- 最终项目完全无法使用

过程

- 立项
  - 时间：2001 年 6 月
  - 目标：升级案件文档管理系统
  - 计划三年，预算 1400 万美元
- 需求变化
  - 开始一年之内，需求从升级变成重新开发
  - 预算增加，时间延长
  - 利好软件承包商 SAIC 公司

过程

- 混乱：NRC 对 VCF 项目出具的评估报告显示：该项目的开发工作一片混乱，可能从开发伊始就缺乏整体规划。甚至是在项目完工日期之后的几个月，仍然存在下列明显的问题：
  - 探员无法通过该系统将案件资料带到现场进行参考。
  - 系统缺乏最基本的人性化操作特性，连书签和历史记录功能都没有，难以查找有用的资料
  - 系统的排序功能不正常
  - 系统在上线前几乎没有做过测试，上线成败与否完全是随机的
  - 对系统上线可能失败的情况没有做预案，整个系统的上线计划就是一场豪赌。一旦系统上线失败，将彻底失去信息化运作能力。
- 试图挽回
  - 2004 年 6 月，雇佣另一家软件研发公司试图修正项目
  - 该公司出具的报告显示项目已无法挽回
  - 2005 年正式宣告失败

失败原因

- 项目从一开始就缺乏完整的构思，从而导致架构设计的失败
- 频繁的需求变更
- 项目管理上频繁往复，导致系统规格混乱
- 对具体软件开发人员管理过于死板
- 项目中的很多经理级别管理人员，缺乏基本的计算机科学背景，造成外行领导内行，甚至干扰项目的进行
- 项目进度严重滞后的情况下，依然不停地添加新的需求
- 项目需求变更和范围扩大导致的代码膨胀问题
- 奢望项目能够快速上线投入使用，造成项目无法通过使用磨合提高软件的可用性

启示

- 一个软件项目的成败因素是多方面的
- 明确需求与管理变更
- 有效的项目管理
- 技术评估与选择
- 选择合适的开发模型
- 供应商管理

案例：XZ Utils 后门事件 - 概述

- XZ Utils 项目概述
  - 是.xz 文件格式的实现，基于 lzma 算法，在 Linux 系统中有着非常广泛的应用
  - 项目维护者 Lasse Collin

后门基本信息：

- CVE-2024-3094
- XZ Utils 5.6.0 和 5.6.1 的发布 tar 包包含后门。这些 tar 包由 Jia Tan 创建并签名
- 攻击者即使没有有效账号，也能通过 sshd 登陆系统
- 具体攻击细节仍在调查中
- 进入了 Debian 和 Fedora 的测试版本

发现过程

- 最初发现该问题的是 Postgres 数据库的维护者之一 Andres Freund。
  - 不是专门的安全研究员
  - 来自微软
- 在 Debian 测试版本测试 Postgres 时发现了 openssh-server 的性能问题
  - 登陆时 CPU 占用高
  - 登陆时间慢了 0.5s 左右
- 几周的调查之后，基本弄清楚来龙去脉
  - 经过精心设计的供应链投毒

https://www.openwall.com/lists/oss-security/2024/03/29/4

后门植入时间线

- 2021 JiaTan Github 账号建立，11.16 在 libarchive 提交了第一个 PR
- 之后两年，陆续在 xz 等项目提交代码，代码质量看起来还比较高
- 期间有多个不同邮件地址向 XZ 项目维护者施压，最终 JinTan 取得了 XZ 项目的权限
- 2024 年 2 月 23 日，Jia Tan 将隐藏的后门二进制代码合并到一些二进制测试输入文件中。
- 2024 年 2 月 24 日：Jia Tan 标记并构建 v5.6.0，并发布带有额外恶意 build-to-host.m4 的 xz-5.6.0.tar.gz 发行版。
  - 这个 m4 文件不存在于源代码库中，但打包过程中另外其他合法文件也被添加了，因此它本身并不可疑。但脚本已被改动，添加了后门。
- 5.6.0 版本被发现有 ifunc 漏洞，该漏洞似乎与攻击无关。
- 但是发行版另外一些可能修改使得 5.6.0 中的后门可能失效，JiaTan 加速发布了可能包含新后门的 5.6.1 版本。
- JiaTan 推动新版本进入各大发行版。

XZ Outbreak (CVE-2024-3094)

XZ Utils is a collection of open-source tools and libraries for the XZ compression format, that are used for high compression ratios with support for multiple compression algorithms, notably LZMA2.

On Friday 29th of March, Andres Freund (principal software engineer at Microsoft) emailed oss-security informing the community of the discovery of a backdoor in xz/liblzma version 5.6.0 and 5.6.1.

Github Activity Summary (user: JiaT75)

Repository: https://github.com/tukaani-project/xz

- **2021** — User Jia Tan (JiaT75) creates his Github Account
- **2022-02-06** — JiaT75's first commit to the XZ repo
- **2023-06-28** — Potential infrastructure testing: liblzma: "Add ifunc implementation to crc64_fast.c."
- **2023-07-08** — PR opened in oss-fuzz to disable ifunc for fuzzing builds. Allegedly to mask the malicious changes.
- **2024-02-16** — Malicious "build-to-host.m4" file added to .gitignore, later incorporated to the package release.
- **2024-03-09** — Obfuscated/encrypted stages binary backdoor hidden in two test files:
  - tests/files/bad-3-corrupt_lzma2.xz
  - tests/files/good-large_compressed.lzma.

xz/libzma v5.6.0 & v5.6.1

Packaged in the final releases

m4/build-to-host.m4

The M4 macro is executed during the build process and runs the malicious code below.

```
...
63 gl_[$1]_config='sed \"r\n\" $gl_am_configmake |
eval $gl_path_map | $gl_[$1]_prefix -d 2>/dev/null'
...
95 gl_path_map='tr "\t \-_" " \t_\-"'
...
```

Reads Bytes

tests/files/bad-3-corrupt_lzma2.xz

Substitution to uncorrupt malformed XZ file

- 0x09 (\t) are replaced with 0x20
- 0x20 (whitespace) are replaced with 0x09
- 0x2d (-) are replaced with 0x5f
- 0x5f (_) are replaced with 0x2d

Decode Data

Uncorrupted bad-3-corrupt_lzma2.xz

Stage 1 - Bash File

v5.6.0
- Bytes in comment: 86 F9 5A F7 2E 68 6A BC
- Custom substitution (byte value mapping)

v5.6.1
- Bytes in comment: E5 55 89 B7 24 04 D8 17
- Check if script running on Linux
- Custom substitution (byte value mapping)

tests/files/good-large_compressed.lzma

1. Decompress the file with xz -dc
2. Remove junk data from the file using multiple head tool calls
3. Portion of the file is discarded (contains the binary backdoor)
4. Use custom substitution cipher to decipher the data
5. Deciphered data is decompressed using xz -F raw --lzma1 -dc

Bash script

Stage 2 - Bash File

v5.6.0 Backdoor extraction

An .o file extracted & integrated into compilation/linking

1. Extract & decipher tests/files/good-large_compressed.lzma
2. Manipulate output with: LC_ALL=C sed "s/\(.\)/\1\n/g"
3. Decrypt using AWK script (RC4-like)
4. Decompress with xz -dc --single-stream
5. Binary backdoor stored as liblzma_la-crc64-fast.o

liblzma_la-crc64-fast.o is then added to the compilation/linking process!

v5.6.1 Extension Mechanism

1. Search Files: use grep -broaF in tests/files/ for signatures:
   a. "~!:_ W", "|_!{ -" output: "file_name:offset:signature"
   b. "jV!^%", "%.8.IZ"
2. If Found:
   a. Save first offset + 7 as $start
   b. Save second file's offset as $end
3. Next Steps:
   a. Merge found segments
   b. Decipher with custom byte mapping
   c. Decompress & execute data

No files with the signatures were found, however it highlights the framework's potential modularity for future updates.

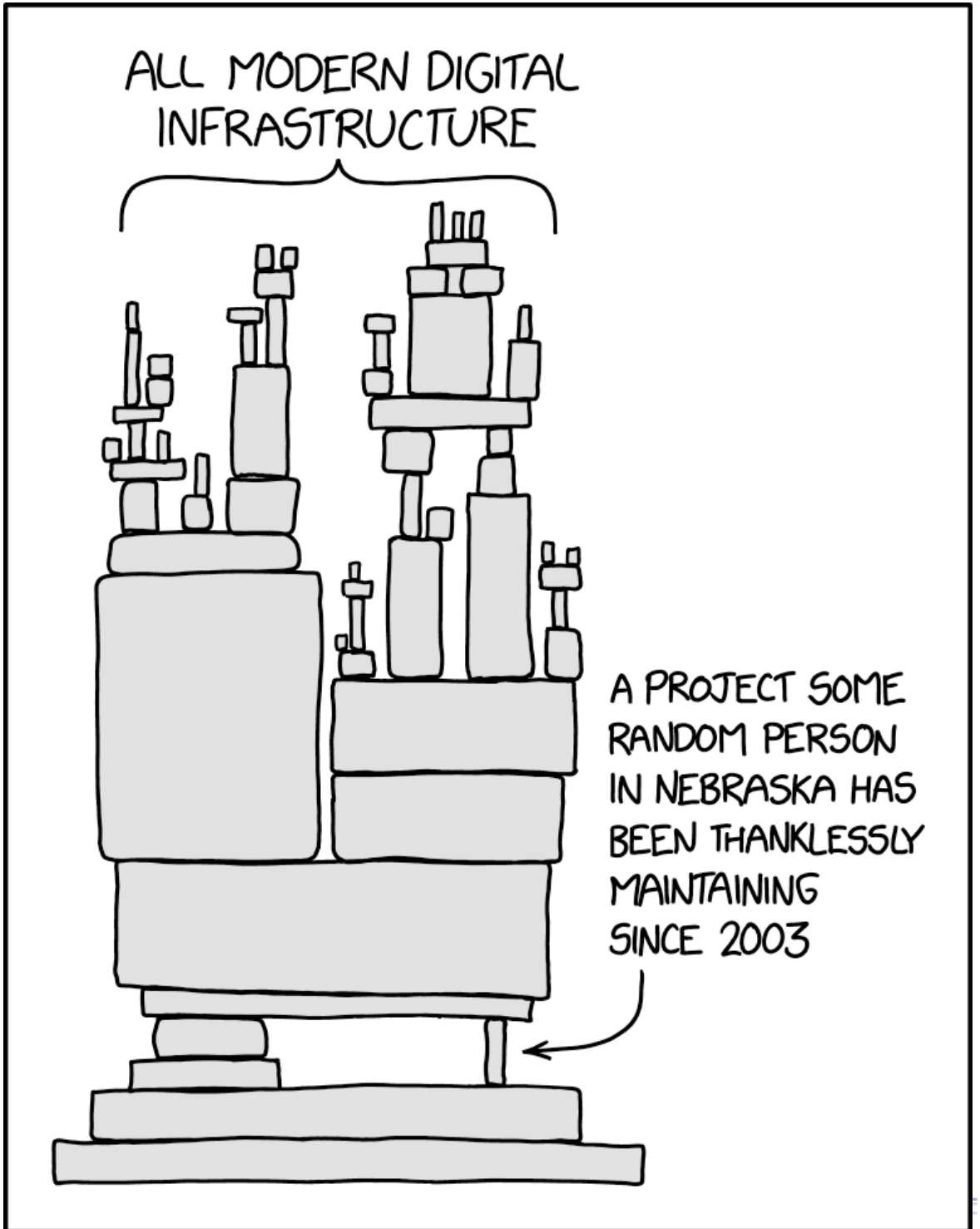
@FRØGGER_
THOMAS ROCCIA

后续

- 漏洞被公开后，Lasse 和 JiaTan 的 Github 账户被暂停。Lasse 的账户后来恢复
- 各大发行版撤下有问题的版本，并发布新版本的包
- xz 项目的官网，邮件列表等去除 JiaTan 的权限或回退到 JiaTan 无权限的备份
- xz 源码的 Github 仓库没有强制 push，也就是 JiaTan 的提交还保留在 Git 历史中
  - 但是触发 build 后门的代码没有包含在其中

启示

- 开源软件维护的难度
- 代码审查的重要性
- 软件中的小瑕疵带来的影响可能很大
- 软件工程实践可能在其中发挥作用
- 开源是提供安全软件的有效途径
  - With enough eyes,all bugs are shallow.（足够多的关注能让所有问题浮现）- Linus
- 没有完美方案


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

目录

云计算为软件工程提供支撑

- 开发与部署环境：
  - 云计算提供了灵活的开发和部署环境，开发人员可以随时随地访问和使用计算资源。
  - Online IDE 可以更方便标准化配置，提高安全性。
- CICD：
  - 利用云平台提供的能力，开发团队可以更高效地进行持续集成和持续交付（CI/CD）
- 协作和管理：
  - 集成的协作工具和平台，如 GitHub、Bitbucket 和 Jira，支持分布式团队的协作开发。
- 专注核心功能与组件
  - 通过利用各种云服务，降低开发者的心智负担

云计算为软件工程提供支撑

- 资源管理：
  - 通过虚拟化技术提供按需资源分配，用户可以根据需要动态调整计算资源的使用。
- 成本效益：
  - 通过按需付费和资源共享，显著降低了硬件和维护成本。
- 安全与合规：
  - 提供了各种安全服务，如身份验证、数据加密和合规性管理，帮助用户保护数据和应用安全。

云计算本身就是一种软件工程实践

云计算产品

- 以开源生态为基础
  - Linux, Kubernetes, Openstack, Ceph, Qemu, Openvswitch etc.
  - 全球协作
- 商业公司提供产品化方案

软件工程在云计算软件开发中发挥着关键作用

- 通过软件工程的系统性思维来解决产品化过程中的问题
- 着眼于整个产品和方案，而不只是具体的代码实现
- 可维护性是软件生命周期的一个重要而关键的阶段

两者互相促进，提升现代软件开发的效率和效果

Thank You!

Q&A