代码安全

概述

软件工程是系统性工程，其中软件代码的安全漏洞是影响软件最终运行安全性的重要因素。安全控制左移，第一步就是保证软件的源代码安全。然而现实并没有想象的乐观，2020 全年公开 CVE 漏洞数为 144431，其中危急漏洞占比 14.07%，高危漏洞占比 42.59%，两者占比达到 56.66%，攻击者利用此类漏洞可以远程执行任意命令或者代码，可见代码安全的整体形势确实不容乐观。

代码产生脆弱性的主要原因是代码开发者缺乏安全经验和安全意识，在编写代码时没有进行必要的安全检查。

例如以下代码没有指定字符串复制长度，容易产生缓冲区溢出漏洞：

```
char dst[14];
strcpy(dst, src);
```

正确的写法应为：

```
char dst[14];
strncpy(dst, src,14)
```

又如在做数据库操作时，以下代码直接拼接 SQL 执行语句，容易产生 SQL 注入漏洞：

```
$query="select name, birth from users where user_id='$id';";
```

正确的写法应是：

```
stmt = $pdo->prepare("select name, birth from users where user_id =?");
$stmt-> execute (array("user_id",$id);
```

[1] https://nvd.nist.gov/vuln/data-feeds

当然，软件出现安全漏洞有各式各样的原因，通常在一个大型系统中，软件存在漏洞是不可避免的，一般使用软件缺陷密度来度量软件的安全性。事实上，可以通过提升开发人员的安全意识和安全水平，或使用相应的安全工具，均可以降低软件缺陷密度。

开源软件安全

自从 Richard Stallman 创立了自由软件基金、Linus 创建 Linux 操作系统后，开源社区迅速扩大。开源软件经过社区孵化成熟度不断提升，良好的生态、开箱即用的功能，丰富的软件库，以及不断更新的功能，使得开源软件备受青睐。现在不仅仅是 DevOps 的敏捷项目中会大量使用到开源软件，一些传统软件开发项目中也都会或多或少地使用到开源软件。

据 Gartner 统计，2016 年，至少 95%的 IT 机构会在关键 IT 产品中使用开源软件，这个数字在 2010 年为 75%，这些软件可能并不被机构所感知；如今新应用的 70%-90%的代码来自外部或第三方组件，每个应用平均有 105 个开源组件。其中最让人不安的是企业对其是否使用了开源软件，使用了什么开源软件并不了解[i]。

事实上，2018 年，超过 16000 个开源软件的漏洞曝光，67%审查的应用包含开源组件的漏洞，每个应用平均有 22.5 个开源组件的漏洞。开源软件每下载 8 次，就有 1 次包括已知安全漏洞[ii]。

可见开源软件的安全问题非常严重，但与之相反的是在 2016 年，不到一半 IT 机构实现了有效的开源软件治理流程，即有效将风险最小化，且最大化投入产出比。

代码审计

审计实施人员对系统重要业务场景进行风险分析并审计源代码，如转账、查询等涉及资金和用户敏感信息的功能场景，在人工分析的过程中，实施人员会通过源代码安全审计工具，对全部代码进行自动化审计，以保证源代码安全审计的全面性。源代码安全审计过程中，除源代码脆弱性审计外，还应参照相关标准和规范，对业务实现的合规性进行审计。

代码审计工具一般以静态分析为主，常见的代码审计工具有静态应用安全测试（static application security testing, SAST）和软件成分分析（software composition analysis，SCA）。

静态应用安全测试（SAST）是指不运行测试程序，仅通过分析代码的语法、结果、过程、参数和接口等检查应用是否存在安全风险和程序的正确性。与人工代码审查相比，具有效率高、结果一致等优点，与动态代码分析相比，具有覆盖度高、运行时间短的优势。但其缺点是实现比较复杂，跟具体的编程语言有关系，很难支持所有的语言。

商用的 SAST 工具有 Fortify[2]、Checkmarx[3]、AppScan[4]和 Coverity[5]等

软件成分分析（SCA）是一种用于发现某应用程序所包含的开源或第三方组件以及其中已知漏洞的应用安全测试技术。软件成分分析可以发现项目中用到的第三方软件库，特别是开源软件，分析相关代码版本库，与漏洞库比较，如有匹配则告知存在漏洞。软件成分分析可以有效地缓解开源软件带来的安全风险，但需要知道软件是不断发展的，不断会有新漏洞出现，所以安全检查需要持续进行。

商用的 SCA 工具有 Snyk[6]，开源的 SCA 工具有 OWASP Dependency Check[7] 、SonaType [8]、Bunder Audit[9]。

通常，在 DevOps 过程中，SAST 和 SCA 可以协同地被集成至 CI/CD 中，以分别检查自有代码和第三方代码。预计 SAST 在整个敏捷开发流程中也会自动化，不仅仅是自动化代码检查，还包括发现漏洞后自动修复。Gartner 预测到

---

[2] https://www.microfocus.com/en-us/portfolio/application-security

[3] https://www.checkmarx.com/

[4] https://www.hcltechsw.com/products/appscan

[5] https://scan.coverity.com/

[6] https://snyk.io/

[7] https://owasp.org/www-project-dependency-check/

[8] https://github.com/sonatype-nexus-community

[9] https://github.com/rubysec/bundler-audit

2022 年, 10%被 SAST 发现的编码漏洞可由自动化方案自动修复，2019 年这个数字还少于 1%[10]。

小结

代码安全是企业安全的第一站，在 DevOps 的大背景下显得更为重要。读者应重点考虑提升开发安全能力，并且灵活使用代码审计工具减少软件安全漏洞。

尽管代码审计可以最大程度上减少代码漏洞，但通常认为静态应用安全测试 SAST 只能覆盖 10%到 20%的代码问题，而运行时检测的动态应用安全测试 DAST 覆盖另外的 10%到 20%。可见代码审计并不能解决全部安全问题，要保证程序在全生命周期的安全，安全左移后还需要考虑重新将安全控制右移，通过运行时检测和响应及时发现和处置威胁。

---

[i] Follow These Four Principles to Effectively Manage, Gartner Security & Risk Summit, 2019
[ii] Gartner, Application Security: Think Big, Start With What Matters, https://www.gartner.com/en/documents/2765517/application-security-think-big-start-with-what-matters

[10] 数据来源于 Gartner 2019 年安全与风险管理峰会