第 5 章
第三方库安全

每个程序员都知道要“避免重复发明轮子”的道理，尽可能使用优秀的第三方框架或库。常见开发语言，如 Java、Python、C 语言、C++、PHP 等，都有对应的第三方库支持。

第三方库被各大系统广泛应用，但是每年都会有许多高危的第三方库漏洞披露，各大互联网公司如果不及时更新第三方库至安全版本，就可能被这些已经公开的第三方库漏洞攻击成功。

本章主要讲解第三方库开源协议和常见的第三方库及其漏洞。

5.1 开源协议选择

每个第三方库都会有属于自己的协议，如今的开源协议有 60 多种，其中，主流的包括 BSD、AL2.0、MPL、MIT、EPL、GPL 和 LGPL 等协议。

5.1.1 BSD

BSD(Berkly software distribution，伯克利软件套装)是 UNIX 的衍生系统，BSD 许可证原先是用在加州大学伯克利分校发表的各个 4.4BSD/4.4BSD-Lite 版本上面的，后来逐渐沿用下来。1979 年，加州大学伯克利分校发布了 BSD UNIX，被称为开放源代码的先驱，BSD 许可证就是随着 BSD UNIX 发展起来的。BSD 许可证被 Apache 和 BSD 操作系统等开源软件采纳。

BSD 开源协议主要包括 FreeBSD license 和 Original BSD license，是一个给予使用者很大自由的协议。使用者可以自由地使用，修改源代码，也可以将修改后的代码作为开源或专有软件再发布。

使用 BSD 开源协议必须满足以下条件：

(1) 如果再发布的产品中包含源代码,则源代码中必须带有原来代码中的 BSD 协议。

(2) 如果再发布的只是二进制类库/软件,则需要在类库/软件的文档和版权声明中包含原来代码中的 BSD 开源协议。

(3) 不可以用开源代码的作者/机构名称和原来产品的名称做市场推广。

BSD 开源协议鼓励代码共享,但需要尊重代码作者的著作权。由于 BSD 开源协议允许使用者修改和重新发布代码,也允许使用或在 BSD 代码上开发商业软件发布和销售,因此它是对商业集成很友好的协议。很多公司在选用开源产品时首选 BSD 开源协议,因为可以完全控制这些第三方的代码,在必要的时候可以修改或二次开发。

5.1.2　AL2.0

AL2.0 是 Apache License 2.0 的简写,AL2.0 开源协议主要包括 Apache License Version 2.0、Apache License Version 1.1 和 Apache License Version 1.0。Apache License 是著名的非营利开源组织 Apache 采用的协议。该协议和 BSD 开源协议类似,同样鼓励代码共享和尊重原作者的著作权,同样允许代码修改,可以作为(开源或商业软件)再发布。

使用 AL2.0 开源协议必须满足以下条件:

(1) 需要给代码的用户一份 Apache License。

(2) 如果修改了代码,需要在被修改的文件中说明。

(3) 在延伸的代码中(修改和有源代码衍生的代码中)需要带有原来代码中的协议、商标、专利声明和其他原来作者规定需要包含的说明。

(4) 如果再发布的产品中包含一个 Notice 文件,则在 Notice 文件中需要带 Apache License。使用者可以在 Notice 中增加自己的许可,但不可以对 Apache License 构成更改。

Apache License 也是对商业应用友好的许可。使用者也可以在需要的时候修改代码来满足需要,并作为开源或商业产品发布/销售。

5.1.3　GPL

GPL(general public license,GUN 通用公共许可证)协议和 BSD、AL2.0 等鼓励代码重用的许可很不一样。GPL 的出发点是代码的开源/免费使用和引用/修改/衍生代码的开源/免费使用,但不允许修改后和衍生的代码作为闭源的商业软件发布和销售。这也就是为什么用户能免费使各种 Linux,包括商业公司的 Linux 和 Linux 上各种各样的由个人、组织,以及商业软件公司开发的免费软件。

GPL 协议的主要内容是只要在一个软件中使用(“使用”指类库引用,修改后的代码或衍生代码)GPL 协议的产品,则该软件产品必须也采用 GPL 协议,即必须也是开源和免费的,这就是所谓的“传染性”。GPL 协议的产品作为一个单独的产品使用没有任何问题,还可以享受免费的优势。

由于 GPL 严格要求使用了 GPL 类库的软件产品必须使用 GPL 协议,商业软件或对代码有保密要求的部门就不适合集成/采用 GPL 协议的开源代码作为类库和二次开发的基础。其他细节,如再发布时需要伴随 GPL 协议等和 BSD/Apache 等类似。

GPL 是最严格和最彻底的开源协议,使用该协议最重要的代表就是 Linux 操作系统,当然也包括在 Linux 上的软件,如图像处理软件 GIMP 等也采用了 GPL 协议。

5.1.4 LGPL

LGPL(lesser general public license,GUN 宽通用公共许可证)是更宽松的 GPL。由于 GPL 太严格,限制了很多商用软件使用 GPL 组件,才推出了这个 LGPL。

与 GPL 要求任何使用/修改/衍生 GPL 类库的软件必须采用 GPL 协议不同,LGPL 允许商业软件通过类库引用(link)方式使用 LGPL 类库而不需要开源商业软件的代码。这样采用 LGPL 协议的开源代码可以被商业软件作为类库引用并发布和销售。

如果修改 LGPL 协议的代码或衍生,则所有修改的代码、涉及修改部分的额外代码和衍生的代码都必须采用 LGPL 协议。因此 LGPL 协议的开源代码很适合作为第三方类库被商业软件引用,但不适合以 LGPL 协议代码为基础,通过修改和衍生的方式做二次开发的商业软件采用。

GPL/LGPL 都保障原作者的知识产权,避免有人利用开源代码复制并开发类似的产品。

5.1.5 MPL

MPL(the mozilla public license)是 1998 年初 Netscape 的 Mozilla 小组为其开源软件项目设计的软件许可证。MPL 许可证出现的最重要原因是 Netscape 公司认为 GPL 许可证没有很好地平衡开发者对源代码的需求和他们利用源代码获得的利益。同著名的 GPL 许可证和 BSD 许可证相比,MPL 在许多权利与义务的约定方面与它们相同(因为都是符合 OSIA 认定的开源软件许可证)。

MPL 还有以下几个显著的不同之处。

(1) MPL 虽然要求对于经 MPL 许可证发布的源代码的修改也要以 MPL 许可证的方式再许可出来,以保证其他人可以在 MPL 的条款下共享源代码。但是,在 MPL 许可证中对“发布”的定义是“以源代码方式发布的文件”,这就意味着 MPL 允许一个企业在自己已有的源代码库上加一个接口,除了接口程序的源代码以 MPL 许可证的形式对外许可,源代码库中的源代码就可以不用 MPL 许可证的方式强制对外许可。这些就为借鉴别人的源代码用作自己商业软件开发的行为留了一个豁口。

(2) MPL 许可证第三条第七款中允许被许可人将经过 MPL 许可证获得的源代码同自己其他类型的代码混合得到自己的软件程序。

(3) 对软件专利的态度。MPL 许可证不像 GPL 许可证那样明确表示反对软件专利,但是却明确要求源代码的提供者不能提供已经受专利保护的源代码(除非他本人是专利权人,并书面向公众免费许可这些源代码),也不能在将这些源代码以开放源代码许可证形式许可后再去申请与这些源代码有关的专利。

5.1.6　MIT

MIT(Massachusetts institute of technology)是和 BSD 一样宽泛的许可协议,作者只想保留版权,而无任何其他的限制。使用者必须在发行版中包含原许可协议的声明,无论是以二进制发布的还是以源代码发布的。

被授权人的权利和义务如下。

(1) 被授权人有权利使用、复制、修改、合并、出版发行、散布、再授权及贩售软件及软件的副本。

(2) 被授权人可根据程序的需要修改授权条款为适当的内容。

(3) 在软件和软件的所有副本中都必须包含版权声明和许可声明。

5.1.7　EPL

EPL(eclipse public license)协议需要遵守以下规则。

(1) 当一个 Contributors 将源码的整体或部分再次开源发布时,必须继续遵循 EPL 协议来发布,而不能改用其他协议发布,除非得到了“源码”所有者的授权。

(2) EPL 协议下,可以将源码不做任何修改来商业发布。但如果要发布修改后的源码,或者当再发布的是目标代码时,必须声明它的源代码是可以获取的,而且要告知获取方法。

(3) 当需要将 EPL 协议下的源码作为一部分跟其他私有的源码混合成为一个项目发

布时，可以将整个项目/产品以私人的协议发布，但要声明哪一部分代码是 EPL 协议下的，而且声明那部分代码继续遵循 EPL 协议。

(4) 独立的模块(separate module)，不需要开源。

(5) EPL 协议允许任意使用、复制、分发、传播、展示、修改及改后闭源的二次商业发布。商业软件可以使用，也可以修改 EPL 协议的代码，但要承担代码产生的侵权责任。

5.1.8 Public Domain

公有领域(public domain)是人类的一部分作品与一部分知识的总汇，可以包括文章、艺术品、音乐、科学和发明等。对于领域内的知识财产，任何个人或团体都不具有所有权益(所有权益通常由版权或专利体现)。这些知识发明属于公有文化遗产，任何人可以不受限制地使用和加工它们(此处不考虑有关安全、出口等的法律)。创立版权制度的初衷是借由给予创作者一段时期的专有权利作为(经济)刺激，以鼓励作者从事创作。当专有权利期间截止，作品便进入公有领域。由于公有领域的作品没有专属权利人，因此公众有权自由使用它们。

对于开源第三方库的使用，建议使用 public domain、BSD、AL2.0 和 MIT，谨慎选择 MPL 和 EPL，避免选择 GPL 和 LGPL。

5.2 Java 常用第三方库

在选择第三方库给应用开发带来便捷的同时，常有各种各样的第三方库的安全漏洞披露，所以如果使用第三方库，就要及时更新第三方包至安全的版本。

5.2.1 Java 核心扩展

Java 标准库虽然提供了那些最基本的数据类型操作方法，但仍然对一些常见的需求场景，缺少实用的工具类。而另一些则是 Java 标准库本身不够完善，需要第三方库去加以补充的。

1. Apache Commons Lang

Apache Commons Lang 是 Apache 最著名的 Java 库，它是对 java.lang 的很好的扩展，包含了大量非常实用的工具类，其中用得较多的有 StringUtils、DateUtils 和 NumberUtils 等。

除了 Apache Commons Lang，还有一些其他的 Apache 库也是对 Java 本身的很好的补充，如 Apache Commons Collection、Apache Commons IO 和 Apache Commons Math。

在 Maven 项目中加入 Apache Commons Lang 这个库的方法：

```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.4</version>
</dependency>
```

2. Google Guava

Google Guava 包含 Google 在自己的 Java 项目中所使用的一些核心 Java 库，包含对集合、缓存、并发库、字符串处理和 I/O 等各方面的支持。另外 Google 开发的库以性能著称。

在 Maven 项目中加入 Google Guava 这个库的方法：

```
<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava</artifactId>
  <version>19.0</version>
</dependency>
```

5.2.2　Web 框架

Web 框架是一个应用最核心的部分，因此推荐使用那些有良好社区支持的框架，如 Spring 和 Struts。

1. Spring

Spring 是一个开源的应用框架，它包含很多子项目，如 Spring MVC、Spring Security、Spring Data 和 Spring Boot 等，几乎可以满足项目上的所有需要。

在 Spring MVC 项目中加入这个库方法（以下举例引入 Spring Core 的支持）：

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>4.2.5.RELEASE</version>
</dependency>
```

2. Struts

Struts 2 是 Apache 中有名的 Web 框架，它也是一个免费开源的 MVC 框架。Struts 也能很好地支持 REST、SOAP 和 AJAX 等最新技术。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts2-core</artifactId>
    <version>2.3.28</version>
</dependency>
```

5.2.3 数据库（持久层）

持久层框架的选择对一个项目的成败同样非常关键，它会直接影响系统的性能、质量、安全及稳定性。

1. MyBatis

MyBatis 是数据库（持久层）框架，它完全是基于 SQL 语句的（通过 SQL 来提取数据并自动映射为所需的数据对象），有足够的灵活性。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.0</version>
</dependency>
```

2. Hibernate

Hibernate 是国内用得较广泛的持久层框架，它非常强大，但用好它并不容易，需要了解它的内部机制，否则可能会出现一些无法预见的性能问题，特别是在数据量特别大的时候。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.1.0.Final</version>
</dependency>
```

5.2.4　日志

Java 中也包含日志记录功能，但它在处理日志分级、日志的存储，以及日志的备份、归档方面都不够出色，因此在项目中一般会使用第三方日志库来处理日志。

1. SLF4J- Simple Logging Facade for Java

SLF4J 提供了一个日志服务的抽象层，基于它开发人员可以选择不同的日志实现，如 java. util. logging、logback 和 log4j，当开发人员需要改变日志实现组件时，不需要修改任何代码，只需要更改一些相应的配置即可。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.21</version>
</dependency>
```

2. Apache Log4j

Log4j 是有名的日志组件，通过简单的配置后就能在程序中方便地记录各个级别的日志，它的日志文件能够根据不同的规则进行命名及归档。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.5</version>
</dependency>
```

5.2.5　单元测试

JUnit 是目前使用极广泛的 Java 单元测试库。通过它可以非常方便地编写自己的单元测试代码，并进行自动化测试。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
```

5.2.6 Office 文档处理

1. Apache POI

Apache POI 是一个免费的开源库,用于处理 Microsoft Office 文档。使用它可以使用 Java 读取和创建,修改 MS Excel 文件、MS Word 和 MSPowerPoint 文件。

在 Maven 项目中加入这个库方法:

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.14</version>
</dependency>
```

2. docx4j

docx4j 是另一套基于 JAXB 的 Office 文档(docx、pptx、xlsx)处理库。

在 Maven 项目中加入这个库方法:

```
<dependency>
    <groupId>org.docx4j</groupId>
    <artifactId>docx4j</artifactId>
    <version>3.3.0</version>
</dependency>
```

5.2.7 XML 解析

1. JDOM

JDOM 是一个开源项目,它基于树形结构,利用纯 Java 的技术对 XML 文档进行解析、生成、序列化等多种操作。在 JDOM 中,XML 元素用 Element 表示,XML 属性用 Attribute 表示,XML 文档本身用 Document 表示。因此这些 API 都非常直观易用。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>org.jdom</groupId>
    <artifactId>jdom</artifactId>
    <version>2.0.2</version>
</dependency>
```

2. DOM4J

DOM4J 是一个处理 XML 的开源框架，它整合了对于 XPath，并且完全支持 DOM、SAX 和 JAXP 等技术。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>dom4j</groupId>
    <artifactId>dom4j</artifactId>
    <version>1.6.1</version>
</dependency>
```

3. Xerces

Xerces 是一个开放源代码的 XML 语法分析器。从 JDK1.5 以后，Xerces 就成了 JDK 的 XML 默认实现。

在 Maven 项目中加入这个库方法：

```
<dependency>
    <groupId>xerces</groupId>
    <artifactId>xercesImpl</artifactId>
    <version>2.11.0</version>
</dependency>
```

一个大型的 Java 应用可能要引用数百个第三方库，如何有效地管理这些第三方库，定期扫描第三方库漏洞，并及时修复这些第三方库存在的漏洞，各大软件公司都应该有自己的一套可重复执行的方案。

5.3　近年第三方库相关漏洞披露

近年披露的服务器相关漏洞如表 5.1 所示。

表 5.1 近年第三方库相关漏洞披露

漏 洞 号	影 响 产 品	漏 洞 描 述
CNVD-2018-10064	Google Guava 11.0-24.x(<24.1.1)	Google Guava 11.0～24.1.1 版本(不包括 24.1.1 版本)中存在安全漏洞,该漏洞源于程序未能正确地检测客户端发送的内容及数据大小是否合理。远程攻击者可以利用该漏洞造成拒绝服务
CNVD-2020-03821	Spring Framework 4.1.4	Spring Framework 4.1.4 版本中存在代码问题漏洞,攻击者可利用该漏洞执行代码
CNVD-2020-03854	Spring Framework 5.2.*,<5.2.3	Spring Framework 5.2.3 之前的版本中存在跨站请求伪造漏洞,该漏洞源于 Web 应用未充分验证请求是否来自可信用户,攻击者可以利用该漏洞通过受影响客户端向服务器发送非预期的请求
CNVD-2019-44960	Apache Struts2	Apache Struts2 中存在安全漏洞。攻击者可以利用该漏洞借助畸形的 XSLT 文件上传并执行任意文件
CNVD-2018-15894	Apache Struts2>=2.3,<=2.3.34 Apache Struts2>=2.5,<=2.5.16	Apache Struts2 存在 S2-057 远程代码执行漏洞。漏洞触发条件:①定义 XML 配置时 namespace 值未设置且上层动作配置(action configuration)中未设置或用通配符 namespace。②URL 标签未设置 value 和 action 值且上层动作未设置或用通配符 namespace。攻击者可以利用漏洞执行 RCE 攻击
CNVD-2018-06262	SLF4J<1.8.0-beta2	SLF4J 中的 slf4j-ext 模块的 org.slf4j.ext.EventData 存在安全漏洞。远程攻击者可以借助特制的数据利用该漏洞绕过访问限制
CNVD-2020-00502	Apache Log4j 1.2	Apache Log4j 1.2 版本中存在安全漏洞。攻击者可以利用该漏洞执行代码
CNVD-2019-41291	Apache POI<=4.1.0	Apache POI 4.1.0 及更早版本存在信息泄露漏洞。当使用 XSSFExportToXml 工具转换用户提供的 Microsoft Excel 文档时,攻击者可以通过特制文档利用该漏洞从本地文件系统或内部网络资源读取文件
CNVD-2020-33467	dom4j<2.1.3	dom4j 2.1.3 之前版本中存在代码问题漏洞。该漏洞源于网络系统或产品的代码开发过程中存在设计或实现不当的问题

说明:如果想查看各个漏洞的细节,或者查看更多的同类型漏洞,可以访问国家信息安全漏洞共享平台:https://www.cnvd.org.cn/。

5.4　习题

1. 简述为什么需要第三方库、常见的开源协议及如何选择这些协议。
2. 简述 Java 常用的第三方库及其主要功能。
3. 为什么第三方库会有安全漏洞，如何保证系统中第三方库的安全？