44 | “代码安全篇”答疑汇总

2019-04-15 范学雷

代码精进之路

进入课程 >

讲述：刘飞

时长 10:58　大小 20.09M

到这一篇文章，意味着专栏第三模块“安全的代码”也更新完毕了。今天，我来集中解答一下留言区里的一些疑问。

@醉侠

> 希望老师后面能多讲讲安全编码的例子或者推荐好的书籍，这块儿确实是很大的弱点。

答：不同于其他的编码技术，编码安全是一个在攻与防的拉锯战中持续发展的领域。新的攻击技术花样翻新，防守技术也跟着变化。因此，编码安全的技术和技巧也纷繁复杂。CWE 的常见安全问题列表，到目前为止，已经列举了 1131 种问题。其中每一个问题，都会给软

件带来难以预料的安全风险。由于安全问题和相应的技巧数量如此巨大，学习几种或者十几种编码安全的技巧，能起到的作用并不是很大。

但坐以待毙显然不是我们的风格。安全问题的数量和技巧虽然庞大，但是基本的原理数量并不是很多，而且都很直观。比如，只要你记住，“跨界的数据不可信任”这条简单的原理，并且去校验每一个跨界数据的有效性，就消除了代码的很大一部分风险。至于什么是有效的数据，什么是有害的数据，每一个场景都有不同的定义。当你想要践行这条原理的时候，你总能定义数据的有效性，找到检查的办法。记住了这一条原理，面对跨界数据的时候，你就会加倍警觉，跟踪数据检查点，想办法校验数据。

所以，我能给的第一个建议就是：**记住最基本的安全编码原理。**

遗憾的是，我们掌握每一样技术的过程，都是先有量变，才会有质变。就比如说，“跨界的数据不可信任”这条原理，想要把它变成我们编码时的下意识行为，就需要很多的锻炼。刚开始的时候，要去学一些技巧，然后照葫芦画瓢。慢慢地，不需要葫芦，自己也能画好瓢了。再慢慢地，你想画啥画啥，不管是瓢还是壶。到最后，引领你的思路的，就是基本的原理，而不再是纷繁复杂的技巧。你也就从按照技巧来编码的阶段，转变到根据问题和基本的原理，去寻找技巧，解决问题的新阶段。

比方说吧，你知道了一个数据校验的技巧，整数不能太大。于是，遇到跨界的整数数据，你都会检查这个整数是不是太大。这就是一个很好的实践。如果遇到浮点数呢？如果遇到用户密码呢？如果遇到 SQL 查询语句呢？你会不会想到浮点数不能太大，用户密码不能太长，SQL 查询语句也不能太长？遇到不同的场景，不同的数据，这时候最能给你提供帮助的，就是记住“跨界的数据不可信任”这条原理。然后去想办法去检验整数、浮点数、用户密码、SQL 语句。

从哪里开始积累这个“量变”的量呢？我能给的第二个建议是：**学习编程语言的编码规范以及安全编码指南。**

一门编码语言的规范，通常会告诉大家编码的最佳实践。这些最佳实践，就包括安全编码的内容。比如C/C++语言的编码规范里，就有如何处理整数、浮点数、内存的很多技术和技巧。

如果你学习的是 Java，Java 的安全编码指南是一定要掌握的内容。Java 的安全编码指南是 Java 安全组对潜在的 Java 编码安全问题的总结，既有原则，又有示例。无论是设计还是实

现，我们都要把安全编码指南考虑进来。

Java 语言已经有二十多年的历史了，每更新一个版本，它的安全编码指南都会加入新的内容。这也是安全攻防发展的表现之一。每隔一段时间，总会有新的攻击技术出现，总会有新的安全编码实践。所以，每一个 JDK 版本推出的时候，我们还要检查一下新版的安全编码指南，看看有没有新加的内容。然后，看看我们的代码，有没有需要调整的地方。

一般情况下，每一门语言都会有编码规范和安全指南。其中，一个常用的资源是卡内基梅隆大学的SEI CERT 系列的总结。你自己也找找看，有没有你最喜欢、最适合的资源。

如果你掌握了编码规范和安全编码指南，那就是一个了不起的成就了。但是，对于一个你编写的软件的安全性而言，这还不够。因为，安全的攻防技术是一个持续发展的技术。今天还是安全的系统，明天也许就不安全了。所以，我们也要跟得上变化。

因此，我能给的第三个建议是：**跟踪、学习、使用最新的安全编码进展。**

由于安全漏洞的保密性要求，关于安全编码的最新技术的资源相对来说是稀缺的。一般来说，安全补丁出来的时候，我们大部分人知道的就是：安全补丁出来了，系统需要升级，运维有的忙了。对于研发人员来说，最重要的信息其实是安全补丁补了啥？安全问题在哪儿？是怎么修复的？我们的系统有没有类似的问题？不是所有的安全补丁都会披露安全问题的细节，所以有时候，我们需要去看源代码，去推测到底有什么安全问题，去推测为什么补丁会起作用。

但是这样做太花费时间，幸运的是，有很多人这样做，并且分享了他们的研究成果。我们要做的就是，根据安全问题，使用搜索引擎，搜索相关的研究，然后分析、检查、使用到我们自己的代码里。如果没有现成的研究成果，我们就需要自己动手分析这些安全补丁。

知晓安全问题存在的最重要资源，是 NIST 的安全漏洞数据库，我建议你一定要用好这个数据库。

我知道这并不容易。我们都期望的模式是，培训几个小时，然后天下行走。一个好的程序员是时间堆积起来的。**只有持续地了解、积累、训练，才能慢慢地到达一个期望的水准，才能建立、巩固自己的技术优势。**

我在学习金融课程的时候，经常被各种复杂的数字设计弄得迷迷糊糊。老师最常用的鼓励的话就是：**要学习一点难的东西，这样才能走到更远的地方**。我把这句话也送给你。

@轻歌赋

> 有个问题，案例中 hashtable 增加了一个 entryset 后，攻击者如何直接访问对象的 entryset 呢?
> 以 web 程序为例的话，我想不出用户如何传入可以执行的代码，能过直接让权限检查的调用对象直接执行 entryset，也看不出对方如何能够重写我服务端的代码或者继承并且被 jvm 加载。
> 老师能给个实际的例子吗?

@hua168

> 如果是 web 程序的话，攻击者是怎么查看我们内部程序？ 如果是 API 接口的话，这些方法我们不是隐藏起来，不公开，它怎么绕过漏洞攻击?

答：上面的这两个问题，是一个比较典型的容易迷惑的地方。要想了解这个问题，我们还需要了解我们曾经讲到的边界问题，以及公开接口本身的问题。

比如 Web 服务吧，用户通过浏览器访问 Web 服务，传输使用的 HTTP 或者 HTTPS 协议，并没有机会直接获取服务器端提供服务的 Java 对象。这样的话，服务器端提供应用服务的 Java 对象，只是系统的一个内部实现，外部接口其实是在 HTTP 层。这就是一个很好的边界隔离。

公开接口的问题在于，我们并不知道一个公开接口到底会在什么环境下使用，也许不是 Web 环境。我们也不知道使用公开接口的代码是不是恶意代码，也许它是不可信任的代码。公开接口要做的就是，不管调用者有什么意愿，接口的实现都按照规范执行，调用者不能改变接口的规范。要不然，就是一大堆的问题。

就拿 VM 虚拟机来说吧，一个 VM 上，可能运行了多个用户，每个用户之间并不相互信任，而且虚拟机也不会完全信任每个用户；可能运行多个应用，每个应用之间也不相互信任，而且虚拟机也不会完全信任每个应用。这些用户、应用，为什么能够运行在虚拟机上呢？唯一的办法，就是调用虚拟机提供的接口，来获得虚拟机的资源。而虚拟机要把用户隔离开来，把应用隔离开来，就需要严格的权限安排和调度。而要想实现这些权限的安排和调

度，同样需要通过接口来实现。如果可以越过虚拟机的权限管理，虚拟机上的用户和应用就会面临巨大的安全威胁。

虚拟机及其权限管理的思想，不仅仅只是应用在 JVM 或者云计算。像我们日常使用的浏览器（Firefox、Chrome）、服务器 (nginx、Apache HTTP Server)、打印机，都需要考虑多租户、多任务的问题。

@hua168

> 像我们开发是直接调用框架函数，如果是安全问题，一般是框架自身的问题吧？

这是一个很好的问题。我们的系统每次都及时升级到最新的安全修复版，是不是就够了呢？

如果所有的代码都能够做到及时地推出安全修复版，包括你自己的代码，这样做就够了。

但是，我们常常遗忘了一点，像框架的代码一样，我们自己写的代码也是代码，也会存在安全问题。成熟框架的安全问题虽然很少，但是出现的安全问题通常引人瞩目。我们自己写的代码，存在的安全问题可能会更多，但是通常没人关注，直到问题突发。

如果你留意一下 NIST 的安全漏洞数据库在一段时间内最新的安全漏洞，你可能会发现，出现问题的代码和应用千奇百怪，涉及的领域和技术非常广泛，不仅仅局限于框架、语言、开源代码。有代码的地方，就有安全问题。

我们勤奋地给系统打补丁，但是很少去审视自己编写的代码是不是存在新的安全问题，很少主动地去修复自己代码里的安全问题。这不是你我的个人问题，这是公司的管理和投入问题。有质量的软件维护是昂贵的。

以上就是这次答疑的内容。如果你还有没有解决的疑问，请在留言区给我留言。

如果你觉得这篇文章解决了你的疑惑，对你有所帮助，欢迎点击“请朋友读”，把这篇文章分享给你的朋友或者同事。

上一篇　43 | 编写安全代码的最佳实践清单

下一篇　45 | 尾声：如何成为一个编程好手？

精选留言 (2)

写留言

拉格朗日的...

2019-05-04

打卡

展开

hua168

2019-04-15

打卡

展开