35 | 怎么处理敏感信息？

2019-03-25 范学雷

敏感信息，是一个常见的词汇。如果我们接收到了广告信息，骚扰电话，垃圾邮件等，都表明我们个人的敏感信息或多或少地被泄露了。

有些敏感信息的泄露，也许仅仅会使我们感到不便，比如一小部分的垃圾邮件，但有些敏感信息的泄露，会影响我们的消费倾向或者消费决策，损害我们的经济利益，甚至威胁我们的生命安全，比如医疗记录和行程安排的泄露。

在计算能力已经不再奢侈的今天，看似毫不相干的数据甚至都可以推演出非常关键的隐私信息。比如，你每周三次以上在早晨出发的地方，大概率就是你家的位置。只有几十个人知道你家的位置，这是一件正常的事情；可是如果有10亿人知道，这就麻烦了，几乎不可能发生的小概率事件随时都有可能光临。

我们一定要有保护敏感信息的意识，不管是我们自身的，还是别人的。在互联网世界里，敏感信息真的是无处不在，你知道有哪些编码技术可以保护敏感信息吗？

什么是敏感信息？

要想保护敏感信息，首先要识别敏感信息。什么是敏感信息呢？ 其实，这个问题本身就是一个特别有意思的话题。你要是查阅关于敏感信息定义的不同文献，就可以体会到不同的立场和不同的利益纠葛。敏感信息的界定范围，也透露了游戏规则制定者对于敏感信息保护的力度和态度。

还记得我们在《如何评估代码的安全缺陷？》这篇文章里提到的，信息安全最基本的三个元素

吗？ 私密性、完整性以及可用性是信息安全的三要素。其中，私密性指的是数据未经授权，不得访问，解决的是“谁能看”的问题。在这个框架下，我们可以把敏感信息理解为，未经授权不得泄漏的信息。反过来说，**未经授权不得泄漏的信息，都算是敏感信息**。

让我们一起来看看最常见的一些敏感信息：

1.个人敏感信息

- 个人信息：姓名、性别、年龄、身份证号码、电话号码。
- 健康信息：健康记录、服药记录、健康状况。
- 教育记录：教育经历、学习课程、考试分数。
- 消费记录：所购货物、购买记录、支付记录。
- 账户信息：信用卡号、社交账号、支付记录。
- 隐私信息：家庭成员、个人照片、个人行程。

2.商业敏感信息

- 商业秘密：设计程序、制作工艺、战略规划、商业模式。
- 客户信息：客户基本信息、消费记录、订单信息、商业合作和合同。
- 雇员信息：雇员基本信息、工资报酬。

需要注意的是，上述只是一些常见的、直观的敏感信息。具体到你开发的信息系统，到底什么样的信息是敏感信息，什么样的信息不是敏感信息，还需要进一步分析和界定。

识别出敏感信息之后，就要想办法保护这些信息了。**敏感信息的保护，需要恰当的管理，也需要适合的技术。**

授权，敏感信息谁能看？

敏感信息指的是未经授权不得泄漏的信息。这个概念可以拆分为三部分：

1. 敏感信息是受保护的信息，未经授权，不得访问；
2. 敏感信息是一段有效信息，有信息处理需求，比如产生、传输、存储、读取、更改、废弃等；
3. 敏感信息是有归属的信息，不同的人有不同的权限。经过授权，合适的人可以执行相应的操作。

是否需要授权是敏感信息和普通信息的最关键差异。不同的人有不同的权限，不同的操作需要不同的权限。该如何处理授权呢？

第一件事情，就是定义权限。只有定义了权限，才能知道如何分配和管理权限。在JDK中，权限通过java.security.Permission接口来定义。Permission接口定义权限的名称和操作。比如，java.io.FilePermission把权限名定义为文件名，把操作定义为：

- read
- write
- execute
- delete
- readlink

其中，权限的名称就是抽象了的敏感信息；权限的操作就是对信息的处理。如果把权限的名称和权限的操作结合起来，就可以定义特定的权限了。比如，下面的例子就定义了对于文件目录“/home/myhome”的读操作。

```
permission java.io.FilePermission "/home/myhome", "read";
```

第二件事情，就是定义权限的主体。也就是说，要明确权限可以分派给谁。在JDK中，权限的主体通过java.security.Principal接口来定义。Principal接口可以用来表述个人，组织或者虚拟的账户。比如，com.sun.security.auth.UnixPrincipal可以用来表述Unix的用户。

```
Principal com.sun.security.auth.UnixPrincipal "duke"
```

第三件事情，就是要定义权限的归属。也就是说，有了权限的定义和权限主体的定义，我们就可以分配权限了。下面的例子，就是把对“/home/duke”的读写操作权限，赋予给了Unix用户duke。

```
grant Principal com.sun.security.auth.UnixPrincipal "duke" {
    permission java.io.FilePermission "/home/duke", "read, write";
};
```

上述的三个例子，就是Java权限管理策略文件的最基本概念。更详细的内容，权限管理策略文件的语法以及API的调用，请参阅有关Java的规范。

敏感信息经过授权才可以使用，这看起来是一个漂亮的解决方案。我们是不是可以高枕无忧了？这还远远不够。这套解决方案能够实施下去，还是有很多挑战的。比如说，敏感信息的操作处理过程，也会造成信息的非授权泄漏。

特殊信息，特殊处理

现代信息系统资源，一般都是多用户共享，多应用共享，跨边界合作的，比如内存、硬盘、中央处理器和互联网。而敏感信息是不能共享的，如何在共享的资源内，不留下敏感信息的踪迹？这是一个让人头疼的问题。

比如说吧，要使用敏感信息，就要把敏感信息载入内存，如果发生内存溢出攻击，攻击者就可以绕过权限管理，获取或者修改敏感信息，甚至可以修改对敏感信息的操作。

针对敏感信息的操作，需要特殊的处理和特殊的技术。

敏感信息的无意识泄露是一种比较常见的敏感信息泄露事件。比如说，把敏感信息泄露在抛出异常里，应用日志里，或者序列化对象里。

比如说，如果一个文件不存在，一般的代码实现会倾向于抛出java.io.FileNotFoundException异常。为了使异常信息更加直观，我们常常把文件路径包含在异常的消息里，或者记录在应用日志里。

```
java.io.FileNotFoundException： /home/duke/.ssh was not found
```

这个异常信息有可能绕过权限管理，传达给未授权用户。这个信息里包含了三个重要的敏感信息：

- 当前用户名是“duke”；
- 当前用户没有配置SSH协议；
- 有可能获知特定文件是否存在。

当实现一个文件管理类时，我们可能会习惯于面向对象的机制。比如，给定一个文件的路径，代码就执行一定的读写操作。至于该文件路径是什么，包含什么内容，是否有敏感信息，都不在该类的考虑范围之内。实现这个类时，我们更可能倾向于使用直观友好的异常信息，而不会意识到这些**异常信息可能携带敏感数据，导致敏感数据通过异常信息泄露**。

这和我们一般的面向对象的编程习惯是不符合的，这就要求我们特别小心。从实现者的角度出发，抛出的异常信息尽量做到不包含可能的敏感信息；从调用者的角度出发，截获的异常信息在传递到上层调用之前，如果有必要，需要做净化处理。比如，把上述的java.io.FileNotFoundException转化成更普通的java.io.IOException。

```
java.io.IOException： An IOException was caught!
```

下面的异常堆栈是不是可以接受呢？

```
java.io.IOException： An IOException was caught!
at com.example.myapp.MyHTTPSerer.myMethod(MyHTTPSerer.java:250)
...
Caused by java.io.FileNotFoundException： /home/duke/.ssh was not found
at com.example.myapp.MyFileStream.open(MyFileStream.java:249)
```

这个异常堆栈的“Caused by”部分泄露了同样的敏感信息。所以，**在做异常信息净化处理时，可能还需要避免传递捕获异常的堆栈**。特别是如果调用结果直接面向最终用户，就应当尽量避免使用异常堆栈。比如说，在HTML页面中显示异常信息和异常堆栈，就容易出问题。

在后面的文章中，我们还会讨论敏感信息及时归零的话题。这也是对于高度敏感信息的一种特殊处理方式。

小结

通过对这个案例的讨论，我想和你分享下面两点个人看法：

1. **要建立主动保护敏感信息的意识；**
2. **要识别系统的敏感信息，并且对敏感信息采取必要的、特殊的处理。**

保护敏感信息，是编写安全代码的一个重要内容。下一次，我们接着聊更多关于敏感信息的特殊处理技术。

一起来动手

阅读隐私保护政策，就是一个建立敏感信息保护意识，学习隐私保护策略和技术的一个好办法。你可以试着阅读Google和腾讯的隐私保护政策。

为了获得相应的服务，作为消费者，我们需要做出什么样的妥协，能得到什么样的保护，我们有什么样的权利？为了提供相应的服务，作为服务者，我们需要什么样的信息，需要多大程度的授权，能够提供什么样的保护？

欢迎你把自己的经验和看法写在留言区，我们一起来学习、思考、精进！

如果你觉得这篇文章有所帮助，欢迎点击“请朋友读”，把它分享给你的朋友或者同事。

精选留言

天佑

1

嗯，敏感信息保护要从全生命周期的视角去看待，需要技术与管理双管齐下，有时还要取点巧，还要根据当下权衡效率与安全之间是否平衡，有必要法务介入，法律是安全底线。当然这些过程“撕逼”，“扯皮”少不了的，良好的沟通能力也是不可或缺。。。

2019-03-25

hua168

0

我是说是不是产生的异常写进日志里，然后给用户展示的是友好的错误页面（web的话），如果异常的敏感信息不写进日志文件或日志系统里，那怎么排错？

如果是敏感的数据能放数据库，如mysql，先加密方进去，如果不能放进数据库的数据，如连接数据库的代码配置文件中的用户名和密码，是不是也要加密？我们之前的公司从不加密的。

2019-03-27

作者回复

严肃的敏感信息内部人士也不能够轻易获取。敏感信息写到日志里，是一个很典型的安全漏洞。 想一想，一个开发者看到了日志，或者读取了数据库，就知道了客户的银行账号密码，他要多么高尚自律才可以抵御住这种诱惑？这不是逼着人家想入非非吗？

配置文件里的用户名和密码，一般是系统的账号，不是客户的信息，不过也要严格地设置文件的权限。

非常敏感的数据，比如密码，不建议明文放数据库。一般的敏感信息，比如医疗记录，可以明文放数据库，不过数据库本身的权限管理要做好。

有一些敏感数据，如果和人脱离开来，就没什么意义了。 比如，患有高血压，如果不知道病人特征，数据写到日志也没什么问题。 这也是一种脱敏处理。

2019-03-28

hua168 0

还有数据库中也有敏感信息，一般是用MD5加密

2019-03-27

作者回复

MD5是杂凑函数，不能加密。另外，MD5的破解几分钟的事，安全性不够了，不建议再用了。

2019-03-28

hua168 0

异常信息不是写进日志，然后制定一个友好的页面给用户，这样处理的吗？

2019-03-27

作者回复

没看明白问题。 携带敏感信息的异常信息，也不能写到日志里。

2019-03-27

hua168 0

老师，JDK API文档这么多类，都要看一遍吗？
类和接口最多就是java和C#，上万哪记得这么多？
是不是先大概看下标题，知道有哪些，用到再细看？

2019-03-27

作者回复

没人能都看一遍的。先了解概念，知道有这么回事（这一点非常重要）；选型的时候，再深入一点看架构和设计；用的时候再仔细看接口规范的细节，规范有含糊的地方，甚至还要看源代码。

比如异步接口，知道Java能够提供异步编程能力，异步编程有什么好处，就够了。 需要异步编程的时候，再去研究Java异步编程到底是怎么设计的，应用该怎么做。设计实现自己的异步代码的时候，再去抠接口规范的细节。

2019-03-27