企业应当如何应对 SQL 注入式攻击

对于使用 MSSQL 数据库的网站来说，如果没有对 SQL 注入式攻击进行防御那么将受到致命的打击。关于对付 SQL 注入攻击的方法已经有许多讨论，但是为什么还是有大量的网站不断地遭受其魔掌呢?安全研究人员认为，现在正是重新梳理最佳方法来对付大规模的 SQL 注入攻击的时候，从而减轻与注入攻击相关的风险。笔者在此介绍的这些方法未必是革命性的创举，但是又有多少企业真正按照要求全面地实施这些方法呢?

下面，我们将一一谈论这些方法：

应对 SQL 注入式攻击之使用参数化查询

企业应当制定并强化自行开发软件的安全编码指南，要求开发人员使用参数化查询来构建 SQL 查询，这样就可以将数据与代码区分开来。

对于多数 SQL 查询来说，开发人员需要指明某类标准，为此，就需要利用参数化查询，其实就是在运行时可传递的参数。参数化查询就是在 SQL 语句中有一个或多个嵌入参数的查询。这种将参数嵌入到 SQL 语句中的方法与动态构造 SQL 字符串相比，不易产生错误。下面我们看一个在.NET 应用程序中使用参数化查询的例子。假设我们想给张三增加工资 500 元，可参考如下的代码。这些代码范例演示了参数化查询的使用，并展示了如何使用更新语句：

通过利用 SQL 的更新命令，你可以更新记录。在上面的例子中，我们作了如下操作：创建并打开一个数据库链接;创建一个代表执行更新语句的数据库命令;使用 EDBCommand 的 ExecuteNonQuery()方法执行插入命令。

每一个参数都用一个 EDBParameter 对象指明。对于需要在 SQL 语句中指定的每一个参数来说，你需要创建一个 EDBParameter 对象，然后将值指派给这个对象。然后，将 EDBParameter 对象添加到 EDBCommand 命令的参数集中。

对于多数开发平台来说，应当使用参数化的语句而不是将用户输入嵌入到语句中。在许多情况下，SQL 语句是固定的，每一个参数都是一个标量，而不是一个表。用户输入会被指派给一个参数。下面再给出一个使用 Java 和 JDBC API 的例子：

```
PreparedStatement prep = conn.prepareStatement("SELECT * FROM
 USERS WHERE USERNAME=? AND PASSWORD=?");

prep.setString(1, username);

prep.setString(2, password);

prep.executeQuery();
```

笔者用这些例子只是想告诉开发人员，应当确保在查询数据库之前对输入进行净化。要保障用户输入到网站的内容就是你正要查找的数据类型，所以

说，如果你正在寻找一个数字，就要努力保障这种输入一定是一个数字而非字符。

应对 SQL 注入式攻击之实施过滤和监视工具

在 Web 应用程序和数据库这个水平上的过滤和监视工具可有助于阻止攻击并检测攻击行为，从而减轻暴露在大规模的 SQL 注入式攻击中的风险。

在应用程序水平上，企业应当通过实施运行时的安全监视来防御 SQL 注入攻击和生产系统中的漏洞。同样地，Web 应用防火墙也有助于企业部署某些基于行为的规则集，可以在发生损害之前阻止攻击。

在数据库水平上，数据库活动监视还可以从后台过滤攻击。数据库的监视活动是对付 SQL 注入的一种很强大的工具。对于目前所知道的注入攻击而言，应当部署好过滤器，以便向数据库管理员发出警告：正在发生不太安全的问题;还要有一些一般的过滤器，用以查找 SQL 注入攻击中的典型伎俩，如破坏 SQL 代码的不规则的数字引用等。

应对 SQL 注入式攻击之精心编制错误消息

黑客可以利用你的错误消息，以便于将来对付你。所以开发团队和数据库管理员都需要考虑：在用户输入某些出乎意料的“数据”时，应当返回的错误消息。

企业应当配置 Web 服务器和数据库服务器，使其不输出错误或警告消息。因为攻击者可以利用“盲目 SQL 注入”等技术来了解你的数据库设计细节。

应对 SQL 注入式攻击之及时打补丁并强化数据

由于没有打补丁或者配置错误，而造成与 Web 应用程序相关联的数据库遭受攻击，那么与 SQL 注入攻击相关的风险也会因之增加。

很显然，只要有补丁可用，你就需要给数据库打补丁，并且还要给 Web 应用程序和 Web 服务打补丁。

此外，别忘了你的数据库是怎样配置的。你需要禁用不必要的服务和功能，目的是为了强化数据库及其赖以运行的操作系统。

应对 SQL 注入式攻击之限制数据库的特权

最后，企业需要更好地管理与 Web 应用程序相关的账户与后台数据库交互的方式。许多问题之所以发生，其原因在于数据库管理员全面开放了一些账户，其目的是为了让开发人员更轻松地工作。但是，这些超级用户账户极易遭受攻击，并会极大地增加由 SQL 注入攻击及其它 Web 攻击给数据库所造成的风险。

一定要正确地管理所有的账户，使其仅能以最低的特权访问后台的数据库，当然前提是能够完成其工作。你一定要保障这些账户不会拥有对数据库作出更改的权利。