利用一个组合检测系统来减少对恶意请求的错误判断

林 捷
(上海应用技术学院 上海 201418)

摘 要 当今Web应用程序定制技术被广泛应用,异常检测技术在预警和实时阻断应用级程序攻击网站等方面是一个很合适的方法。然而,异常检测技术更容易产生误报和漏报的问题。利用一个综合系统来解决问题,这个综合系统包括基于Web的异常检测系统和一个数据库异常检测系统。即一个Web的异常检测系统和SQL查询的异常检测系统的串行结构将能提高系统检测的效率。在现有的几种Web应用程序上对其适用性进行了评估,显示出该算法在减少误报和漏报方面是可行的有效的。

关键词 异常检测,SQL查询,Web应用程序
中图法分类号 TP309.2 **文献标识码** A

Use Combination of Detection Systems to Reduce Errors of Judgment on Malicious Request

LIN Jie
(Shanghai Institute of Technology, Shanghai 201418, China)

Abstract Today's Web application customization technology is widely used. Anomaly detection in early warning and real-time blocking the attack site of application-level program is a very suitable method. However, anomaly detection techniques are easier to detect problems of false positives and false negatives. The use of an integrated system can solve the problem. This integrated system includs the anomaly detection system based anomaly detection system, the Web and a database. Anomaly detection system and the SQL query of a Web serial structure of the anomaly detection system will be able to improve the efficiency of the detection system. The algorithm was evaluated for its applicability on several existing Web application. It is showed that the algorithm is feasible and effective to reduce false positives and false negatives.

Keywords Anomaly detection, SQL queries, Web applications

1 背景介绍

异常检测技术容易出错有两种情况:一种情况,异常网页的请求被屏蔽了是正常的,但是如果一些合法的请求也被否定,那就会造成可用性下降;另一情况,如果恶意的请求被允许访问网页,网页的数据将被一个后台数据库给截获,网页的安全重要信息将被泄露给这个攻击者[1]。而现今技术Web应用程序是利用底层的网络基础设施给客户提供服务的。在许多情况下,Web应用程序包括客户端组件得到了相当多的安全的发展,例如Web服务、语言翻译和数据引擎。但是特定的应用程序代码由于严格的时间限制和程序员很少的安全培训等原因,发展比较缓慢,导致脆弱的Web应用程序完全暴露在整个互联网当中,因此要为整个网络创建容易开发的切入点来方便整个网络和互联网的通讯[2]。

以CVE(Common Vulnerabilities and Exposures)的数据库公布了的相关漏洞网页数量来分析这种发展趋势。1999年到2005年,不安全Web应用程序从15.1%增长到49.1%,并且这个趋势在继续增长[6],那些由于对Web网妥协的网站,已经统给它们的一些客户支付了大量的金钱,因为这些客户注册的资料被黑客攻击而泄露。最理想的是,用一些精密的设计方法和一些安全的测试来解决Web应用程序的安全问题。不幸的是,现实和理想总是有差距的[3],安全意识的发展方法,需要一个对恶意活动能够提供早期预警的入侵检测基础设施。现在的检测系统(IDS)采用数字签名技术。很多黑客网页不容易被数字签名技术发现,因为这些黑客程序是为一个内部开发的应用,而且只为一个单一的组织的需求服务[7]。此外,黑客程序不包含任何共同的特征和特性。一些异常的检测系统经常会把一些黑客程序作为正常程序建议给用户,所以要为一些黑客程序定做一些检测程序。一些系统使用机器学习技术检测正常使用的配置文件关联这些应用程序。这些配置文件经常用来检测一个正常或者不正常的需求。这个方法相当于一个承诺,并且已经成熟,已作为商业产品被投入到市场当中去了[8]。但是,一些检测系统很容易发生错报和漏报的问题,这是因为这些检测系统过度简化的模型和不足的测试。这些检测程序主要是发现一些不正常的需求来判断是否是黑客程序,反之如果黑客以正常的状态出现,这些检测程序就不能发现了。以漏报为代表,这些黑客程序他会攻击一些允许他们去保护的一些Web程序,而造成一些

本文受上海市教委科研创新项目(12YZ166)资助。
林 捷(1975—),女,硕士,讲师,主要研究方向为数据挖掘、软件工程,E-mail:cleverlj@126.com。

敏感信息的泄漏。

为改善这种情况，提出了一个架构，这个架构包括一个基于网络的异常检测系统、反向 HTTP 代理和数据库异常检测系统一个串行组。这个系统在对漏报方面增加了检测的可能性，同时减少了负面影响，增加了 Web 程序的异常探测器的探测能力。提出的异常检测系统除了查询工作，对后端数据库的 Web 应用程序也执行检测。

2 各种异常探测方法

主要介绍了不同情况下的探测方法，讨论了不同的探测方法的优劣，再利用一种组合的形式进行探测。

2.1 一个 IPS 防御系统介绍

下面给出一个潜在的异常探测器组成的正式分析，以产生一种减少整体误报率的方法。所采用的一些符号[4]的定义如下：

- I 为侵入行为，那么 $!I$ 表示非侵入行为。
- A_d，A_w 表示被检测系统检测到黑客攻击后发出的警告，那么 $!A_d$，$!A_w$ 表示没有警告（A_w 代表网页检测，A_d 代表数据库检测）。
- $P(A_w|I)$，$P(A_d|I)$ 表示检测系统的检测率。
- $P(A_w|!I)$，$P(A_d|!I)$ 表示错报率。
- $P(!A_w|I)=1-P(A_w|I)$，$P(!A_d|I)=1-P(A_d|I)$ 表示漏报率。
- $P(!A_w|!I)=1-P(A_w|!I)$，$P(!A_d|!I)=1-P(A_d|!I)$ 表示良性请求率。

在这个 Web 检测和数据库检测框架当中，由此产生的系统的概率如表 1 所列。

表 1

无漏报	良性请求
$P(!A_w\|!I).P(!A_d\|!I)$	$P(A_w\|I)+P(!A_w\|I).P(A_d\|I)$
漏报	错报
$P(!A_w\|I).P(!A_d\|I)$	$P(A_w\|!I)+P(!A_w\|!I).P(A_d\|!I)$

从上面的公式，我们能得出两个结论，首先，异常探测器组成的反馈的序列对良性请求率的增加有潜在的能力。第二个结论，不幸的是这个架构对错报率的增加，也有一定程度的提高。

3 一个组合检测系统

3.1 异常驱动的反向 HTTP 代理系统

黑客的目标是攻击那些无法获得主机控制的 Web 应用程序（例如：通过一个缓存区的溢出）或者窃取一些敏感信息（例如：通过一个 SQL 语句注入攻击，转储数据库表的内容）。

第一种被攻击的类型是，Web 服务器上的一个脆弱的软件或者一个妥协的基本安全 Web 网页的请求得到服务器端的许可。第二种被攻击的类型是可以由一个单一的后端数据库来存储所有的基于 Web 的应用程序的持久性信息。因此，黑客通过利用这些要访问的数据库内容的有限部分的代码来侵入数据库。例如，f_1，f_2 和 f_3，它们都有相同的访问数据库的权限；就算 f_1 不能够访问数据库的表文件，f_2 也能访问表 x。如果一个网站有可能设计服务器和数据库的基础设施，使不同层次的对数据库的访问和主机上运行的服务器进程可以清楚地执行，那么这个网站会被弹性攻击，如图 1(a)所示。

下面构造一个电子商务程序。这个电子商务公司没有敏感和静态的信息（包括公司的联系方式和一些公司的基本信息）通过第一个服务器访问。非敏感产品供应的动态信息通过第二个服务器连接到一个产品数据库访问。关于用户的敏感信息通过第三个服务器被访问，该服务器依赖一个用户数据库，这个数据库是从产品数据库中分离出来的。最后一个服务器是访问信用卡的服务器。这个设计如图 1(b)所示。这样，每个函数根据自己的需要运行在不同的服务器上。

这种设计使提供产品信息的服务器不可能对黑客妥协，不会允许其访问那些存放敏感客户信息的数据库和信用卡通讯服务器。不幸的是这种设计不能运用在这种情况下，即一些网站应用提供一些不同功能的函数，但是这些函数又密切相关。

因此，提出了一个替代设计，亦即将网站的所有内容复制到服务器上，这个方法就不需要对任务分割了。此外，那些被 Web 程序调用的数据库被用于扩展用户帐户在不同层次的特权。每个账户连接的是一个复制的服务器。检测系统用来确定请求攻击的可能性。这个请求信息被一个反向 HTTP 代理转发到 Web 服务器，能够提供最佳的服务水平给异常请求打分。反向 HTTP 代理是个特殊的代理，它被安装在一个 Web 服务器前端。反转代理的使用是为了提高网站的性能，它通过网页缓冲技术或者由几个不同的服务器分配负载。它的作用是根据异常探测器组件的异常值来把请求传输给正确的服务器。

根据这个设计，电子商务程序的实施如图 1(c)所示。在这个例子中，这个网站的 3 个函数 f_1，f_2 和 f_3 分别被复制到 3 个服务器 A，B 和 C 上。这步不需要修改源程序。一个 Web 服务器代理配置一些查询，这些查询把高度异常发送到服务器 A，把中度异常查询发送给服务器 B，最后把正常查询的发给服务器 C。服务器 C 上的查询是唯一能访问信用卡服务器的查询。中度异常查询用户被分为两类，一类为 u_1，它只能访问数据库中的 x 表，另一类是 u_2，它能访问数据库中的 x 和 y 表，u_1 用户和 B 服务器关联，u_2 用户和 C 服务器关联。

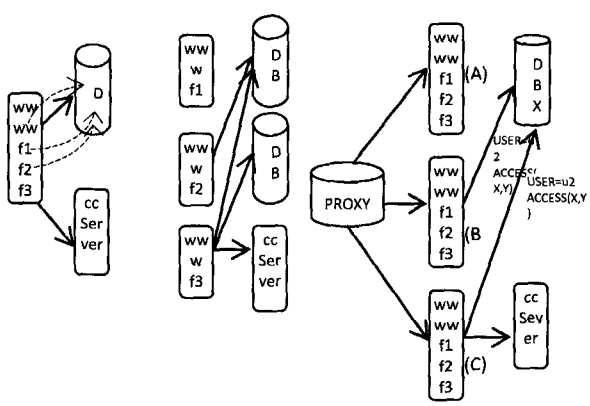

图 1 网站设计

在这个设计中，如果使用到 f_2 这个函数，是需通过服务器 A 的，就算 f_2 这个函数需要访问数据库，这个操作也是失败，因为服务器 A 上没有数据库。另一个方面，一个高度异常的请求也被正确地执行。如果确定是一个高异常的请求，那么用户被拒绝访问。如果这个请求被确定为黑客，有两种

情况是可能的。第一种情况，黑客能够扩张为访问数据库，但是因为服务器上没有数据库，所以访问是失败的。第二种情况是，黑客已经得到主机的妥协[5]。

在这种情况下，如果黑客成功地攻击了服务器，但是由于这个服务器和信用卡服务器没有关联，也导致攻击失败。注意，这台主机可能是服务器请求的小部分(只有那些标记为异常)，因此，它可能是“硬化”，对于某些更有弹性的攻击类型表现出性能下降的特点。

中度的异常请求被发送到 B 服务器上。使用 f_2 函数的用户，将能被正确地执行，因为 B 服务器允许访问数据库中的表文件。也应该注意，某一时刻如果发送过多的请求给服务器 A，请求也会失败的。然而，函数 f_3 通过 B 服务器的访问是失败的，因为这个服务器没有访问敏感信息的权限。最后，所有的正常的请求在服务器 C 上正确执行。因此这个模型提供了足够的便利，在请求正确的执行上也提供了足够的便利。

在减少误报情况这方面，这种体系结构的有效性取决于请求的数量，不需要访问 f_2 和 f_3 函数。例如：Web 程序中对图片的请求大致 40%不需要查询数据库。如果我们假设，其错误地标记为恶意转发到服务器的请求平均分配，则其中 40%可以适当提供给用户。

3.2 敏感的路径覆盖

估计可以由一个受限制的服务器的请求计算敏感路径的程序的分数。路径覆盖的概念是一个非常著名的软件测试度量，这个方法是执行以下部分程序，即通过测试用例覆盖的程序。关键的思想是这个度量代表了有可能执行那些在数据库上访问敏感信息的分数。我们把这个叫做敏感的路径覆盖的分数。应该注意到敏感路径覆盖技术在精确估算即将执行的敏感操作的数量时，假设一个应用程序所有的路径以相同概率出现。因此，其结果更常用于衡量在访问不敏感信息的时候穿插的访问敏感数据库的代码数量。这些敏感操作干净地从应用程序的其余部分分离是非常罕见的，所以期待反向代理方法能够成功地减少误报率。

不幸的是，路径覆盖的精确计算是不切实际的。指数路径爆炸造成的事实，与 k 的决定继承的代码片段包含多达 2^k 个不同的执行路径。计算一个应用程序的敏感覆盖路径应该用图来解决，图中每个节点代表一个函数。例如两个函数 v、w 分别为节点 v 和节点 w，当函数对应节点 v 调用 w 时，利用图论技术先确定一下 v 有多少个可能的路径调用 w。这个就是敏感路径覆盖的度量，也是这个应用程序的分数。

不执行内部程序分析的情况下，将失去保护每个函数调用的条件信息。假定在运行时它可以调用任何可能的组合。不幸的是，再次导致指数爆炸的路径数，下面公式中的 $P(N)$ 表示 N 个子节点的路径总和，s_i 表示通过节点的路径。

$$p(N)=1+\sum_{s\in p(s)i\in s}\prod_{i\in s}p(s_i)$$

为了减少许多儿子节点的影响，采取了不计算路径的方法。相反地，用式(2)来计算出我们的路径集合。

$$p(N)=1+k*\sum_{i=1}^{k}p(s_i)$$

这个公式中的 k 代表所有 N 的孩子节点，计算所得路径仅仅是通过它子孙节点的路径，调整系数为 k。这是基于后继节点的所有可能组合的路径数相乘和总结的结果，与实际路径相反。该方法计算出很多路径编号，可以在这个路径集合中找到一些有用的路径，即一个函数节点成功到其子孙节点，并且能被应用程序很好地执行。这个通过调整系数来实现。当然，当一个节点拥有一个或更多个子孙节点也拥有较多的路径数量，这个信息也会向上传播，并导致预期的结果，该节点本身也有大量的路径。在调用图中，$p(R)$是 R 这个根节点的路径数量，也表示了这个程序通过可能的路径总数。值得注意的是，当调用中还有直接或间接的递归调用循环的时候，这个算法不会停止。因此这个问题的解决方法是，在生成调用图前，采用深度优先搜索的方法把这个循环从调用图中移掉。图 2 展示一个小的调用图的路径情况。根据递归程序的定义，当计算一个节点的路径的时候，这个路径数是基于其子孙节点的基数决定的，这个算法也是用自底向上的方法评估这个调用图的。如图 3 中所有叶子节点的值都是 1，根据图可知这个应用程序的总路径为 31。如预期，根节点的左孩子拥有更多的路径数量，因为它有更多的子孙。一旦聚合的路径被计算出来，就能确定可通过函数的路径数量，这些函数对应的都是执行敏感操作的，把它称作敏感路径。给定了敏感路径数量和总路径数量，就能计算出敏感路径覆盖。

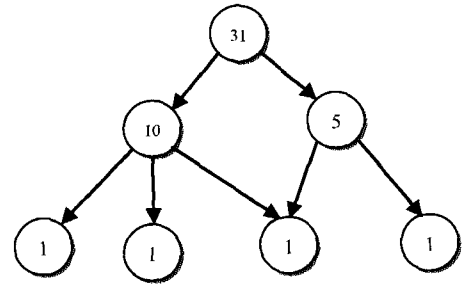

图 2 路径计算

要知道哪些是敏感路径，就要理解什么是敏感操作。通过分析，发现敏感操作可以在检查 Web 应用程序的源代码时检查字符串常量，其代表在那些特权表的数据库操作。一旦找到了敏感操作，调用图的节点对应的就是包括这些敏感操作的函数，并把它标记起来。如前所述，确定操作类型和数据库表的敏感性，取决于应用程序和数据库环境的设计。后面将详细讨论敏感操作的问题。

下面认为所有的 SQL 操作都是写在函数体中的字符串。这意味着，该方案既不利用全局变量来存储 SQL 语句，也不是一个完全动态的方式瞬时产生。当然，这些情况都是可能的，例如一个 WHERE 语句的参数，或者 SELECT 语句的声明，或者是 INSERT 一个值，这些操作都得依赖于变量。然而，我们假设这些操作和表文件的操作是静态的。幸运的是在 Web 应用程序的脚本语言中，直接内嵌的 SQL 语句是一个非常常见的编程惯例，例如 PHP。事实上，目前所有程序还是采用访问数据库的语句直接嵌入指定的操作和访问的表的类型的字符串。

计算敏感路径的算法和上面介绍的路径的求法是类似的，即通过节点标记。当一个节点被标记，那么这个路径就被认为是敏感路径。如果一个节点自己没有敏感操作，但是它的子孙节点有敏感路径，那么就用 $ps(N)$表示这个敏感覆盖是它的子孙节点派生出来的。算法如下面的公式。

$$ps(N)=k*\sum_{i=1}^{k}ps(s_i)$$

上个例子的调用图如图 3 所示，图中有阴影的节点是一个包含敏感操作的节点。除了在每个节点上的总路径，括号中的数字显示敏感路径的每个节点的数量。

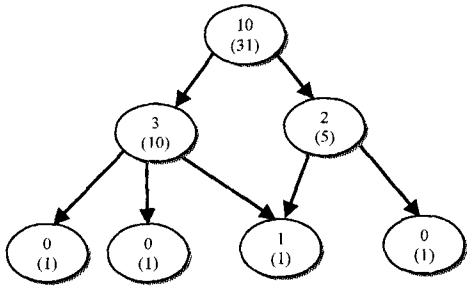

图 3　敏感路径计算

值得注意的是敏感信息的覆盖只是个度量，使用这个方法来支持那些能够在服务器上成功执行的 Web 请求，这些需求是不访问敏感信息的。换句话说，路经覆盖技术是一种用来验证假设的方法，它不是入侵检测工具的一部分。例如，某个应用程序路径覆盖率为 75%，期望用反向 HTTP 代理技术把这个错报率降低到 25%。如果一个程序设计的路径覆盖率为 100%，那么这个程序设计得很差，意味着它所有的 HTTP 关于访问的请求都被归结为访问敏感信息。这种情况下用反向代理技术也不能减少错报率，Web 和 SQL 的组合异常检测器将有效地降低误报率。

3.3　路由 Web 请求

基于代理的方式融入现有的异常检测系统来开发一个反向 Web 代理。这个异常检测组件是基于一个 Web 将异常检测系统的。这个检测器首先从申请 URL 路径中提取一些值，这些路径是被 Web 应用程序给调用的，然后作为参数传递给检测器。检测器查找和 Web 应用程序相关的配置文件。配置文件是一种统计模型的集合，是与一个特定的 Web 应用程序相关联的。这个异常检测模型包含在一个配置文件中，是一套程序，用于评估一个查询属性的某些功能，以及工作在获取和检测两个模型下的一个。在获取阶段，模型建立的“正常”的一个给定功能的属性特征的个人资料，设置一个属性的动态检测阈值。在检测阶段，模型为每个属性值的观察返回一个异常得分。

例如在[0,1]这个时间概率上如何观测这个属性值是异常的呢？要将其与为这个属性建立的配置文件相关联。因为通常一个 Web 应用程序的每个属性关联多个模型，在检测阶段的得分是一个观察到的属性值的最后异常的每个模型的分数计算的加权总和。这个异常分数被用来决定 Web 服务器是否执行这个请求。更多的信息来源于信息自身，且异常检测是个整体，并且是对真实世界的评估。每个后端 Web 服务器都有一个不同的权限级别，并且执行请求的时候是依照不同的异常分值范围。例如，一个正常请求发送给 Web 服务器，这个请求的权限最高，而一个异常请求的权限最低。根据服务器的权限级别，可以访问敏感信息的量根据不同的用户而定，因为每个不同的用户都有不同的权限和一个后台 Web 服务器连接。

为了防止潜在的攻击者，简单地绕过代理部，直接连接到受保护的 Web 服务器，必须部署有效的防火墙机制。实现了分配非路由的 IP 地址的 Web 服务器，并且布置一个服务器来阻断和互联网的直接连接。在代理服务器上，将一个单独的接口的代理连接到隔离网络。

4　SQL 查询的异常检测系统

已经开发一个入侵检测系统，其利用多种检测模式来检测攻击后端 SQL 数据库。下面简要地描述系统的体系结构，然后讨论如何将检测结果反馈用于基于 Web 的异常检测组件。系统能够进入基于 Web 的应用程序和后台数据库之间的通讯通道。应用程序上的 SQL 查询语句将被截获，并且发送给 IDS 上分析。

4.1　特征提取

这个 SQL 异常检测器解析每个传入的 SQL 查询，以产生一个高层次的事件。这个解析输出作为一个系列的令牌。每个令牌都是一个旗帜，且必须用文字对其进行表示。文字是 SQL 查询的唯一元素，它应该包含用户提供的输入。令牌代表了数据库的字段名并增加了这个属性的数据类型。数据类型在寻找字段名时被发现，并且映射到数据库中相应的表名。当查询数据库所有表文件和字段的时候，这个映射是自动产生的。如果希望更准确地描述一个字段的数据类型，这个映射可以被用户下载。例如数据库中有个 VARCHAR 数据类型，这个类型意味着字符串的长度是任意的，但是用户可以改变这个类型给 XML，目的是通知 IDS 这个字段包含一个 XML 对象。可用数据类型的设置是用户可扩展的，并且 IDS 提供了一个简短的借口去指示一个新的数据类型如何被系统处理。

执行类型推断是在查询中包含的文字，使用以下规则：

- 使这段和使用 SQL 操作符字段进行比较的文字的数据类型与给这个比较的字段的类型一致。
- 一段插入表文件的文字与插入表中的字段数据类型一致。

在检测执行之前，该模型将输入事件进行处理，目的是变换查询，产生一个适合的形式去分析。首先，创建一个特征向量是通过提取所有令牌标注的文字，并且插入一个它们在查询中出现的顺序清单。然后一个查询骨架产生了，即所有查询文字被一个空占位令牌给替换掉。这个查询骨架就是一个 SQL 查询的基本结构。因此用户的输入仅仅需要文字表达，不同用户的输入不同，最后都是一种查询骨架。但是一系列的 SQL 的注入将会改变查询结构，并且产生一个不同的查询骨架。

下面我们以一个查询为例：

SELECT firstname FROM users WHERE userid=“foo” AND passed=“bar”;

一个 SQL 异常检测器解析这个查询并且提前一系列令牌。尤其是，foo 和 bar 是文字令牌，而 firstname，userid，passed 被确定为字段令牌。分析这个数据库的架构，这 3 个字段自动地和它们对应的类型关联（这里的类型是 Varchar）。数据类型通过文字传播，得出结论：用户提供的“foo”和“bar”是不是 varch 类型的，一定要和 userid 和 passwd 这两个字段的类型进行比较才能确定。最后，SQL 异常检测器产生特征向量。

(varchar:“foo”,varchar:“bar”)

并且产生了一个骨架查询，在原来的查询使用占位符代文字令牌。

SELECT firstname FROM users WHERE useride=〈INPUT〉 AND passwd=〈INPUT〉

4.2 检测引擎

无论系统处于训练或检测模式，检测引擎均运作在所有不同的入侵检测系统的状态。在训练模式中生成查询的脚本的名称和骨架查询是寻找配置文件的钥匙。配置文件是一个静态文件，并且是一个能确定其功能的模型的映射。如果配置文件是当前脚本文件的名称或者是框架组合，每个元素的特征向量被送入对应的模型中，目的是更新这个模型的"正常"的感觉，如果没有找到配置协议，那么一个新的配置协议将创建并且写入配置文件数据库中。一个配置文件的创建包括一些列的模型，这些模型就是每个元素的特征向量。模型的类型取决于元素的数据类型。例如，如果一个元素的数据类型是 varchar，和它相关联的模型就是字符串模型，如果一个原始是 int，那么和它相关联的数据类型是数值型。

在检测模型中，一个异常分数被观测到并计算出来。如果这个异常分数超过一个阈值，那么即将产生警报。警报产生的原因是找不到这个事件的配置文件，或者这个事件包含的 SQL 语句解析错误。解析错误是企图用 SQL 注入攻击的迹象或一个破碎的应用程序。

5 系统反馈传播

如前所述，组成异常检测系统的体系结构，需要一个 SQL 异常检测元件之间的沟通渠道的存在。在当一个恶意的请求没被前端的一个异常检测器（检测漏报）检测到的时候——这个检测器带着一个恶意查询数据库，这个通道会被 SQL 异常探测器利用。在该系统中，SQL 异常探测器可以选择更新的 Web 请求异常探测器，以进一步阻止类似的恶意请求。更新过程分为 3 个阶段：1. 异常特征；2. 建设和传输模式更新消息；3. 模型更新。下面具体描述这个几个阶段。

5.1 异常特征

模型更新第一步处理包括特征检测到异常的性质。组成异常特征的反常现象有几个特点：包括异常 Web 请求调用、异常值本身、异常检测模型的具体参数，所有这一切都是由 SQL 异常探测器输出。说明异常表征的过程，假设 SQL 适用于字符串长度的模型基础上的切比雪夫不等式的数据字段的查询。在训练阶段，探测器了解到，这一特定领域观察到的平均长度为 μ，一个方差 σ 和一个阈值 t。在测试阶段，假设一定的查询中包含一个字段值 X，其长度是标记为异常，因为切比雪夫不等式返回的 $P(x)<t$。系统的特点是一个被调用的资源被提取出异常特点，用 r 表示，则 μ，σ 和 t 从警告的模型中提取，联合模型的类型标识符 m，建立一个元组 (m,μ,σ,t)。最后异常的特点是这样的元组 $A=(r,x,M)$，r 为资源，x 为异常的值，M 为这个模型的一系列关于 x 的参数。这个公式中的 $M=((m,m,\mu,\sigma,t))$，这个特征元组可以写为 $A=(r,x,((m,m,\mu,\sigma,t)))$。

5.2 模型跟新消息

给出的一个元组 A 是基于 SQL 的 IDS 检测到异常的性质。接下来的步骤是在异常检测系统建立的基础上，封装这方面的资料和消息并转发给 Web 应用程序。在这个系统上，这个元组 A 被翻译成一个 XML，然后通过加密套接字连接发送到 Web 请求异常探测器。

5.3 模型下载

Web 请求异常探测器收到的模型更新消息后，面临的任务是选择正确的模型或模型更新。因为通常多个模型对应多个模型参数，下载产生异常的 SQL 查询的资源模型，以确定正确的模型设置规范。关联异常值，需要一个或多个资源参数模型集算法。这个过程是复杂的事实，至少在理论上，Web 请求异常探测器据悉的参数可能有实际的 SQL 查询请求处理程序适用于它们任意变换前的构造。然而，在实践中，这种转变往往很简单，许多资源参数是一对一映射到表的字段数据库中（用户名、电子邮件地址）。

根据这些考虑，在原型实现中，我们采取的模式选择算法如下。资源 r 设置每个参数的模型，选择相应的模型 M 的成员，设为 M'。然而，每个 $M_i' \in M'$，异常分数 $P_{M_i'}(x)$ 被计算。如果 x 的概率表示 x 是反常的事实，该模型不更新。例如，A 的元组，Web 请求异常探测器构造连接到资源 R 上面所述的参数字符串的长度模型的集合 M'。然而，每个模型 $M_i' \in M'$，它的方差 $\sigma_{M_i'}$ 是减少的，X 不再被认为是正常的，当切比雪夫不等式应用到该字符串的长度。一旦 M' 里的所有模型被更新，就反映异常的 x，这个算法停止。

6 实验结果

这 3 个程序运行平台是：pentium4，2GHz，内存 1G，运行在 linux 系统下。配置 Web 服务器上运行两个基于域名的虚拟主机：聪明和愚蠢。Apache 确定实际的主机被用来服务传入的请求，观测 HTTP 标头部分的主机字段的值。要进一步完善条块分割，我们安装 suphd 模块，它扩展了 Apache 来执行不同的用户权限在两个虚拟主机的 PHP 页面。

表 2 训练和误报的实验结果

程序名		训练集	干净的测试集	异常情况	失败的请求
myblogggie	Web 请求	55k	6k	19	0
	SQL 查询	550k	190k	0	
Phpay	Web 请求	272k	5k	0	0
	SQL 查询	10.4M	186k	2	
punbb	Web 请求	360k	9k	2	1
	SQL 查询	2.1M	51k	2	

表 3 恶意流量的实验结果

程序名		恶意请求	Web 异常	SQL 异常	成功的攻击
myblogggie	Web 请求 400 / SQL 查询 8k	100	99	0	1
Phpay	Web 请求 1.1k / SQL 查询 61k	100	90	10	0
punbb	Web 请求 560 / SQL 查询 3.5k	100	67	32	1

参 考 文 献

[1] Akritidis P, Anagnostakis K, Markatos E. Efficient content-based of zero-day worms[C]// Proceedings of the International Conference on Communications (ICC). eoul, Korea, May 2005

[2] Andersson R. punnBBB-fast and lightweight PHP-powered discussion board[OL]http://www.punbb.org, 2005.

（下转第 380 页）

仿真隧道开挖过程、隧道围岩地层结构特征、分布规律和预警开挖前方可能发生的地质灾害风险为目的。仿真开挖方法思路是：以隧道开挖设计模型为开挖裁剪多边形，隧道设计线路为漫游方向，空间动态地裁剪三维地质模型的空间地层属性元素重构隧道开挖模型空间，揭示隧道围岩地质结构特征分布规律和地质灾害风险。具体实现步骤：1)由用户交互式建立隧道开挖设计模型并以此作为裁剪多边形模型；2)沿隧道设计线路，用多边形裁剪算法[15]，对三维地质模型作空间剖面裁剪生成地质属性多边形序列剖面；3)依据地层结构关系、断层结构关系和属性位置关系作同地质属性单元连接，生成隧道虚拟开挖场景模型；4)漫游仿真隧道开挖施工，依据开挖位置与地质灾害风险数据库同步处理计算，预警报告当前位置的地质灾害分险信息。图7给出了某里程位置的隧道虚拟开挖的场景，隧道底部为隧道里程标记，开挖前方标出了该里程段的地质灾害类别与级别；用虚线标注断层发生的部位。同时这些信息在右部窗口显示列出。

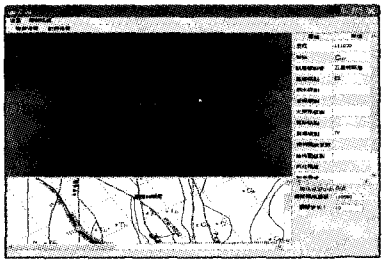

图7　隧道虚拟开挖

结束语　紧密结合隧道工程建设的实践，有效地应用计算机信息处理、数据库管理、可视化和虚拟现实等技术，研发了一套适应解决隧道工程3D复杂地质结构的隧道仿真开挖软件系统。系统提供的三维地质建模与隧道仿真开挖功能，能可视化地揭示隧道围岩内部地质结构特征及空间分布规律，清晰地标记出隧道围岩分类、地质灾害风险(灾害源类型、位置、强度级别和影响规模等)等信息。系统提供的各种可视化图形工具和交互编辑工具，能够实现地质专家扩展逻辑思维空间和知识推理，且能实现地质空间分布的一致性和合理性解释。系统运行在几个大型铁路隧道开挖工程中，结果证明：其对于隧道开挖科学规划与安全施工起到了极为重要的作用。

虚拟仿真技术在地下隧道工程的应用有着极大发展与应用空间，更进一步将地球物理勘探资料、隧道超前预报资料等综合于三维地质模型同步建模并融入系统一体化，无疑将对地下工程地质状态的动态分析、风险评估和隧道实时施工的安全措施规划与决策发挥更大作用。

参 考 文 献

[1] Gutierrez M, Doug B, Joseph D, et al. An IT-based system for planning designing and comstructing tunnels in rocks[J]. Tunneling and Underground space Technology, 2006, 21(3/4): 221

[2] 钟登华，王忠耀，李明超，等. 复杂地下洞室群工程地质三维建模与动态仿真分析[J]. 计算机辅助设计与图形学学报，2007，19(11)：1436-1441

[3] 朱良峰，任开蕾，潘信，等. 地质实体模型的三维交互与分析技术研究[J]. 岩土力学，2007，28(9)：1959-1963

[4] 王宝军，施斌，宋震. 基于GIS与虚拟现实的三维地质建模方法[J]. 岩石力学与工程学报，2008，27(增2)：3563-3568

[5] 钟登华，李明超，杨建敏. 复杂工程岩体结构三维可视化构造及其应用[J]. 岩石力学与工程学报，2005，24(4)：575-580

[6] 黄地龙，邓飞. 复杂地层结构模型三维重构与可视化方法研究[J]. 成都理工大学学报，2008，35(5)：553-558

[7] 张煜，白世伟. 一种基于三棱柱体体元的三维地层建模方法及应用[J]. 中国图像图形学报，2001，6(3)：285-290

[8] 柴贺军，黄地龙，黄润秋，等. 岩体结构三维可视化及其工程应用研究[J]. 岩土工程学报，2001，23(2)：217-220

[9] 王李管，尚晓明，曾庆田，等. 三维可视化建模技术在岩石质量评价中应用[J]. 岩石力学与工程学报，2008，27(增2)：3554-3559

[10] 李邵军，冯夏庭，王威，等. 基于地层信息的三维洞室可视化仿真技术研究[J]. 岩土力学，2008，1(29)：235-239

[11] 戴会超，田斌. 科学计算可视化仿真及其在水电行业中的应用[J]. 水利发电学报，2005，24(6)：88-94

[12] 张建斌，朱合华，朱岳明，等. 厦门翔安海底隧道数字化建模技术[J]. 岩石力学与工程学报，2007，26(6)：1237-1242

[13] 刘涛，沈明荣，陶履彬，等. 连拱隧道动态施工模型试验与三维数值仿真模拟研究[J]. 岩石力学与工程学报，2006，25(9)：1802-1808

[14] 谭德宝，张煜，孙家柄. 滑坡区域的真三维数字仿真[J]. 长江科学学院院报，2005，22(6)：67-70

[15] 刘勇奎，高云，黄有群. 一个有效的多边形裁减算法[J]. 软件学报，2003，14(4)：845-856

(上接第348页)

[3] Roesh M. Snort-lightweight intrusion detection for networks[C]// Proceedings of the USENIX LISA'99 Conference. Seattle, WA, November 1999

[4] Axelsson S. punBB-fast fallacy and its implications for the difficulty of intrusion detection[C]// Proceedings of the 6tth ACM Conference on Computer and Computer and Communications Security. Singapore, 1999

[5] Sidiroglou K A S, Akritidis P, Xinidis K, et al. Detecting targeted attacks using shadow honeypots[C]// Proceeding of the USENIX Security Symposium Baltimore. MD, August 2005

[6] Common vulnerabilities and exposures[OL]. http://www.cve.mitre.org, 2003

[7] Breache Security Breachgate [OL]. http://www.breach.com, 2006-08

[8] Citrix. Citrix application firewall [OL]. http://www.citrix.com, 2006-08